«بـــــــه نام خـــالق آرامـــش»

نام کتاب: SQL Server۲۰۱۹(بخش دوم)

نام نویـــنده: فرشید بابجانی زاده

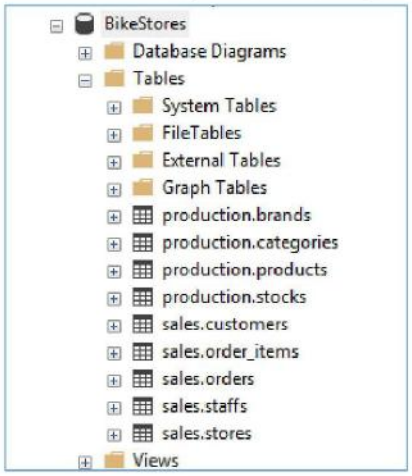
تعداد صفحات: ۱۵۴ صفحه

تاریخ انتشار: ــــــــــــــــ

```
CREATE TABLE sales.stores (
	store_id INT IDENTITY (1, 1) PRIMARY KEY,
	store_name VARCHAR (255) NOT NULL,
	phone VARCHAR (25),
	email VARCHAR (255),
	street VARCHAR (255),
	city VARCHAR (255),
	state VARCHAR (10),
	zip_code VARCHAR (5)
);

CREATE TABLE sales.staffs (
	staff_id INT IDENTITY (1, 1) PRIMARY KEY,
	first_name VARCHAR (50) NOT NULL,
	last_name VARCHAR (50) NOT NULL,
	email VARCHAR (255) NOT NULL UNIQUE,
	phone VARCHAR (25),
	active tinyint NOT NULL,
	store_id INT NOT NULL,
	manager_id INT,
	FOREIGN KEY (store_id) REFERENCES sales.stores (store_id) ON DELETE CASCADE ON
UPDATE CASCADE,


FOREIGN KEY (manager_id) REFERENCES sales.staffs (staff_id) ON DELETE NO ACTION ON
UPDATE NO ACTION
);

CREATE TABLE sales.orders (
	order_id INT IDENTITY (1, 1) PRIMARY KEY,
	customer_id INT,
	order_status tinyint NOT NULL,
	-- Order status: 1 = Pending; 2 = Processing; 3 = Rejected; 4 = Completed
	order_date DATE NOT NULL,
	required_date DATE NOT NULL,
	shipped_date DATE,
	store_id INT NOT NULL,
	staff_id INT NOT NULL,
	FOREIGN KEY (customer_id) REFERENCES sales.customers (customer_id) ON DELETE
CASCADE ON UPDATE CASCADE,
	FOREIGN KEY (store_id) REFERENCES sales.stores (store_id) ON DELETE CASCADE ON
UPDATE CASCADE,
	FOREIGN KEY (staff_id) REFERENCES sales.staffs (staff_id) ON DELETE NO ACTION
ON UPDATE NO ACTION
);

CREATE TABLE sales.order_items (
	order_id INT,
	item_id INT,
	product_id INT NOT NULL,
	quantity INT NOT NULL,
	list_price DECIMAL (10, 2) NOT NULL,
	discount DECIMAL (4, 2) NOT NULL DEFAULT 0,
	PRIMARY KEY (order_id, item_id),
	FOREIGN KEY (order_id) REFERENCES sales.orders (order_id) ON DELETE CASCADE ON
UPDATE CASCADE,
	FOREIGN KEY (product_id) REFERENCES production.products (product_id) ON DELETE
CASCADE ON UPDATE CASCADE
);

CREATE TABLE production.stocks (
	store_id INT,
```

```
        product_id INT,
        quantity INT,
        PRIMARY KEY (store_id, product_id),
        FOREIGN KEY (store_id) REFERENCES sales.stores (store_id) ON DELETE CASCADE ON
UPDATE CASCADE,
        FOREIGN KEY (product_id) REFERENCES production.products (product_id) ON DELETE
CASCADE ON UPDATE CASCADE
);
```

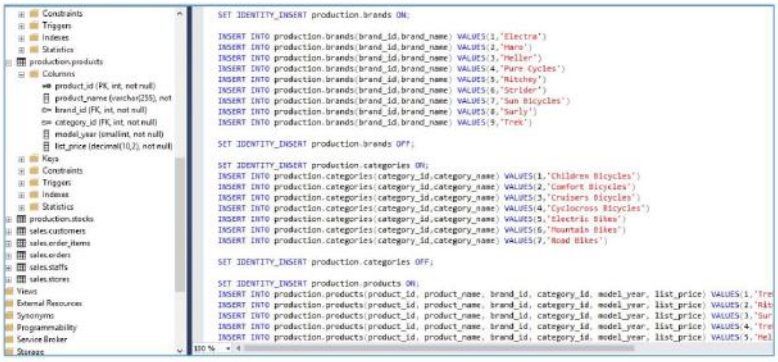
همان‌طور که در شکل زیر مشاهده می‌کنید جداول مورد نظر ایجاد شده‌اند:



در ادامه کار باید اطلاعات این جداول را پر کنیم، برای این کار باید از دستورت زیر استفاده کنیم:

به‌خاطر اینکه تعداد دستورات زیاد است بهتر است از لینک زیر آن را دانلود و در SQL اجرا کنید:

http://3isco.ir/Books/Load-data.sql



توابع ایجاد شده توسط کاربر در SQL Server به شما کمک می‌کنند تا با کپسوله‌سازی منطق‌های کسب‌وکار پیچیده و ایجاد قابلیت استفاده مجدد از آن‌ها در هر query، فرایند توسعه نرم‌افزار خود را تسهیل کنید.

در این بخش مباحث زیر مورد بررسی قرار می‌گیرند:

- توابع اسکالر تعریف شده توسط کاربر: توابع اسکالر تعریف شده توسط کاربر را پوشش می‌دهد. این توابع به شما امکان می‌دهند تتا فرمول‌ها یا منطق‌های کسب‌وکار پیچیده را کپسوله‌سازی کنید و در همه queryها مجدداً از آن‌ها استفاده کنید.
- متغیرهای جدول: چگونگی استفاده از متغیرهای جدول به‌عنوان یک مقدار بازگشتی از توابع تعریف شده توسط کاربر را خواهید آموخت.
- توابع دارای مقدار جدول: شما را با تابع دارای مقدار جدول تک خطی و تابع دارای مقدار جدول چند کدی آشنا می‌کند تا بتوانید توابع تعریف شده توسط کاربر ایجاد کنید که داده‌های نوع‌های جدول را بازگردانند.
- حذف توابع تعریف شده توسط کاربر: چگونگی حذف یک یا چند تابع تعریف شده توسط کاربر موجود از پایگاه‌داده را توضیح می‌دهند.

۱-۱۲-۱-۳ توابع اسکالر در SQL Server

تابع اسکار در SQL Server یک یا چند پارامتر می‌گیرد و یک مقدار بازمی‌گرداند.
توابع اسکالر به شما کمک می‌کنند کدتان را ساده‌سازی کنید. برای مثال، ممکن است یک محاسبه پیچیده داشته باشید که در queryهای زیادی از آن استفاده شده است.
به‌جای درج فرمول در همه queryها، می‌توانید یک تابع اسکالر ایجاد کنید که فرمول را کپسوله‌سازی کرده و از آن در queryها استفاده کند.
ایجاد یک تابع اسکالر

برای ایجاد یک تابع اسکالر، می‌توانید از کد CREATE FUNCTION به شکل زیر استفاده کنید:

```
CREATE FUNCTION [schema_name.]function_name (parameter_list)
RETURN data_type AS
BEGIN
    statements
    RETURN value
END
```

در این syntax:

- اول، نام تابع بعد از کلیدواژه‌های CREATE FUNCTION مشخص شده است. نام شما اختیاری است. اگر صریحاً نام شما را مشخص نکنید، SQL Server به‌صورت پیش‌فرض از dbo استفاده می‌کند.
- دوم، لیستی از پارامترها در بین پرانتزها بعد از نام تابع مشخص شده‌اند.
- سوم، نوع داده از مقدار بازگشتی در کد RETURN مشخص شده است.

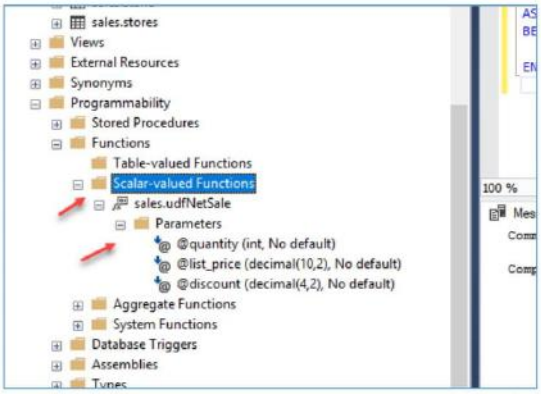
- چهارم، داخل بدنه تابع، از دستور RETURN باید استفاده شود تا یک مقدار بازگردانده شود.

مثال زیر یک تابع ایجاد می‌کند که فروش خالص را بر اساس کمیت، قیمت و تخفیف محاسبه می‌کند:

```
CREATE FUNCTION sales.udfNetSale(
    @quantity INT,
    @list_price DEC(10,2),
    @discount DEC(4,2)
)
RETURNS DEC(10,2)
AS
BEGIN
    RETURN @quantity * @list_price * (1 - @discount);
END;
```

سپس، می‌توانید از این کد برای محاسبه فروش خالص هر سفارش فروش در جدول order_items از پایگاه‌داده Bikestores استفاده کنید.

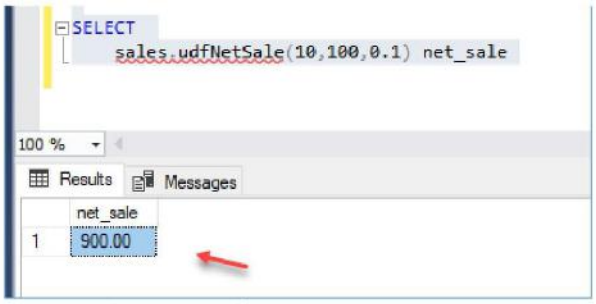
بعد از ایجاد تابع اسکالر، می‌توانید آن را در آدرس programmability > Functions > Scalar-valued Functions که در تصویر زیر مشاهده می‌کنید بیابید:



فراخوانی تابع اسکالر دقیقاً همانند فراخوانی یک تابع معمولی است. برای مثال، کد زیر چگونگی فراخوانی تابع udfNetSale را نشان می‌دهد:

```
SELECT
    sales.udfNetSale(10,100,0.1) net_sale
```

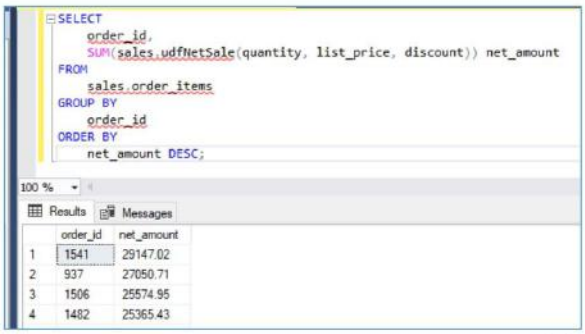
خروجی به این شکل است:



مثال زیر، چگونگی استفاده از تابع Sales.udfNetSale برای به‌دست‌آوردن فروش خالص سفارش‌های فروش در جدول order_items را نشان می‌دهد:

```
SELECT
    order_id,
    SUM(sales.udfNetSale(quantity, list_price, discount)) net_amount
FROM
    sales.order_items
GROUP BY
    order_id
ORDER BY
    net_amount DESC;
```

تصویر زیر بخشی از خروجی را نمایش می‌دهد:



تغییر دادن یک تابع اسکالر

برای تغییر دادن یک تابع اسکالر، می‌توانید به‌جای کلیدواژه CREATE از دستور ALTER استفاده کنید. بقیه کدها دست‌نخورده باقی می‌مانند:

```
ALTER FUNCTION [schema_name.]function_name (parameter_list)
    RETURN data_type AS
    BEGIN
        statements
        RETURN value
    END
```

توجه کنید که اگر تابع تعریف شده توسط کاربر وجود ندارد، می‌توانید از دستور CREATE OR ALTER برای ایجاد یک تابع تعریف شده توسط کاربر یا برای تغییر دادن یک تابع اسکالر از پیش موجود استفاده کنید:

```
CREATE OR ALTER FUNCTION [schema_name.]function_name (parameter_list)
        RETURN data_type AS
        BEGIN
            statements
            RETURN value
        END
```

حذف یک تابع اسکالر

برای حذف یک تابع اسکالر از پیش موجود، می‌توانید از دستور DROP FUNCTION استفاده کنید:

```
DROP FUNCTION [schema_name.]function_name;
```

برای مثال، برای حذف تابع Sales.udfNetSale می‌توانید از کد زیر استفاده کنید:

```
DROP FUNCTION sales.udfNetSale;
```

نکات تابع اسکالر در SQL Server

در لیست زیر نکات کلیدی توابع اسکالر را مشاهده می‌کنید:

- توابع اسکالر را تقریباً می‌توان در همه‌جا درون کدهای T-SQL به کار برد.
- توابع اسکالر یک یا چند پارامتر می‌پذیرند، اما فقط یک مقدار بازمی‌گردانند، در نتیجه باید شامل یک دستور RETURN باشند.

- توابع اسکالر می‌توانند از منطق‌هایی مانند قطعات کد IF یا حلقه‌های WHILE استفاده کنند.
- توابع اسکالر نمی‌توانند داده‌ها را update کنند. می‌توانند به داده‌ها دسترسی داشته باشند، اما این کار اصلاً توصیه نمی‌شود.
- توابع اسکالر می‌توانند توابع دیگر را فراخوانی کنند.

به این صورت، چگونگی استفاده از توابع اسکالر برای کپسوله‌سازی فرمول‌ها یا منطق‌های کسب‌وکار پیچیده و استفاده مجدد از آن‌ها در Queryها را آموختید.

۲-۱۲-۱-۳ توابع تاریخ یا Date

در این قسمت به کار با توابع تاریخ یا Date خواهیم پرداخت. در ادامه لیستی از توابع تاریخ در SQL Server را مشاهده می‌کنید که به شما اجازه می‌دهند داده‌های تاریخ و زمان را به‌صورت مؤثر و کاربردی مدیریت کنید.

توابع بازگرداننده تاریخ و زمان فعلی

تابع	توضیح
CURRENT_TIMESTAMP	تاریخ و زمان فعلی سیستم را بدون بخش محدوده زمانی بازمی‌گرداند.

GETUTCDATE	بخش تاریخ را به‌صورت یک عدد integer بازمی‌گرداند.
GETDATE	تاریخ و زمان فعلی سیستم عاملی که SQL Server در حال اجرا شدن روی آن است را بازمی‌گرداند.
SYSDATETIME	تاریخ و زمان فعلی سیستم را همراه با دقت ثانیه‌های اعشار بیشتر نسبت به تابع GETDATE()بازمی‌گرداند.
SYSUTCDATETIME	تاریخ و زمان سیستم فعلی در زمان UTC را بازمی‌گرداند.
SYSDATETIMEOFFSET	تاریخ و زمان سیستم فعلی همراه با محدوده زمانی را بازمی‌گرداند.

توابع بازگرداننده بخش‌های تاریخ و زمان

تابع	توضیح
DATENAME	بخش تاریخ را به‌عنوان یک کاراکتر رشته بازمی‌گرداند.
DATEPART	بخش تاریخ را به‌عنوان یک عدد integer بازمی‌گرداند.
DAY	روز یک تاریخ خاص را به‌عنوان یک عدد integer بازمی‌گرداند.
MONTH	ماه یک تاریخ خاص را به‌عنوان یک عدد integer بازمی‌گرداند.
YEAR	سال یک تاریخ را به‌عنوان یک عدد integer بازمی‌گرداند.

تابع بازگرداننده اختلاف بین دو تاریخ

تابع	توضیح
DATEADD	یک مقدار به بخش تاریخ اضافه می‌کند و مقدار تاریخ جدید را بازمی‌گرداند.
EOMONTH	آخرین روز ماه موجود در تاریخ مشخص شده را همراه با یک offset اختیاری بازمی‌گرداند.
SWITCHOFFSET	Offsetمنطقه زمانی یک مقدار DATETIMEOFFSET را تغییر می‌دهد و مقدار UTC را حفظ می‌کند.
TODATETIMEOFFSET	یک مقدار DATETIME2را به یک مقدار DATETIMEOFFSET تبدیل می‌کند.

توابع ایجاد تاریخ و زمان از بخش‌های مخصوص به خودشان

تابع	توضیح
DATEFROMPARTS	یک مقدار DATE از سال، ماه و روز بازمی‌گرداند.

DATETIME2FROMPARTS	یک مقدار DATETIME2 از آرگومان‌های تاریخ و زمان بازمی‌گرداند.
DATETIMEOFFSETFROMPARTS	یک مقدار DATETIMEOFFSET از آرگومان‌های تاریخ و زمان بازمی‌گرداند.
TIMEFROMPARTS	یک مقدار TIME از بخش‌های زمان همراه با دقت بیشتر بازمی‌گرداند.

تابع ارزیابی مقادیر تاریخ و زمان

تابع	توضیح
ISDATE	بررسی می‌کند که آیا یک مقدار یک تاریخ، زمان یا Datetime معتبر هست یا نه.

تابع CURRENT_TIMESTAMP در SQL Server

تابع CURRENT_TIMESTAMP برچسب زمان سیستم‌عامل سروری که SQL Server Database روی آن اجرا می‌شود را بازمی‌گرداند. برچسب زمان بازگشتی یک مقدار DATETIME بدون offset محدوده زمانی است.

تابع CURRENT_TIMESTAMP هیچ آرگومانی نمی‌گیرد:

```
CURRENT_TIMESTAMP
```

CURRENT_TIMESTAMP یک معادل ANSI SQL برای ()GETDATE است.

می‌توانید از تابع CURRENT_TIMESTAMP در هرجایی که یک عبارت DATETIME قابل‌قبول باشد، استفاده کنید.

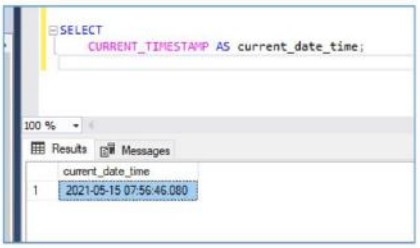
اجازه دهید چند مثال برای استفاده از تابع CURRENT_TIMESTAMP حل کنیم.

الف) مثال ساده CURRENT_TIMESTAMP

مثال زیر از تابع CURRENT_TIMESTAMP برای بازگرداندن تاریخ و زمان فعلی استفاده می‌کند:

```
SELECT
    CURRENT_TIMESTAMP AS current_date_time;
```

خروجی به‌این‌ترتیب است:



ب) مثال استفاده از تابع CURRENT_TIMESTAMP به‌عنوان یک مقدار پیش‌فرض برای ستون‌های جدول

ابتدا، یک جدول جدید به نام current_timestamp_demos ایجاد می‌کنیم که ستون created_at در آن یک مقدار پیش‌فرض را به‌عنوان برچسب زمان می‌پذیرد که در آن یک تابع اضافه شده است:

```
CREATE TABLE current_timestamp_demos
(
    id         INT IDENTITY,
    msg        VARCHAR(255) NOT NULL,
    created_at DATETIME NOT NULL
               DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY(id)
);
```

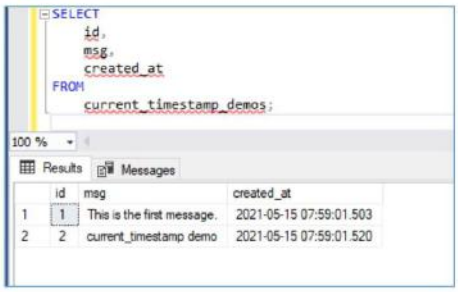
سپس،دو ردیف به جدول اضافه می‌کنیم:

```
INSERT INTO current_timestamp_demos(msg)
VALUES('This is the first message.');

INSERT INTO current_timestamp_demos(msg)
VALUES('current_timestamp demo');
```

در آخر، داده‌ها را از جدول current_timestamp_demos به دست می‌آوریم:

```
SELECT
    id,
    msg,
    created_at
FROM
    current_timestamp_demos;
```

خروجی به این شکل است:



همان‌طور که به‌وضوح در خروجی مشاهده می‌کنید، مقادیر موجود در ستون created_at مقدار برچسب زمان بازگشتی توسط تابع CURRENT_TIMESTAMP را می‌گیرند.

در نتیجه، چگونگی استفاده از تابع CURRENT_TIMESTAMP برای بازگرداندن برچسب زمان سیستم پایگاه‌داده به‌عنوان یک مقدار DATETIME را آموختید.

تابع GETUTCDATE در SQL Server

تابع GETUTCDATE() زمان UTC فعلی را بازمی‌گرداند. تابع GETUTCDATE() این مقدار را از سیستم عاملی که SQL Server در آن اجرا می‌شود، محاسبه می‌کند.

تابع GETUTCDATE() دارای Syntax زیر است:

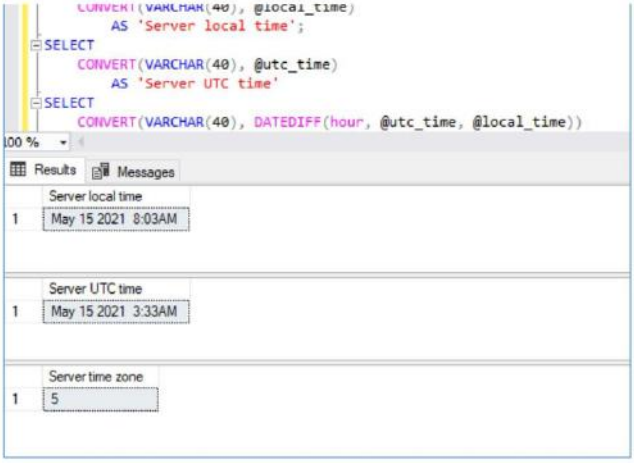
```
GETUTCDATE()
```

مثال GETUTCDATE() در SQL Server

این مثال از توابع GETDATE()، GETUTCDATE() و DATEDIFF() برای بازگرداندن زمان محلی، زمان UTC و محدوده زمانی سرور استفاده می‌کند:

```
DECLARE
    @local_time DATETIME,
    @utc_time DATETIME;

SET @local_time = GETDATE();
SET @utc_time = GETUTCDATE();

SELECT
    CONVERT(VARCHAR(40), @local_time)
        AS 'Server local time';
SELECT
    CONVERT(VARCHAR(40), @utc_time)
        AS 'Server UTC time'
SELECT
    CONVERT(VARCHAR(40), DATEDIFF(hour, @utc_time, @local_time))
        AS 'Server time zone';
GO
```

خروجی دستورات بالا را در شکل زیر مشاهده می‌کنید، در این مثال، چگونگی استفاده از تابع GETUTCDATE() برای به‌دست‌آوردن زمان UTC فعلی را آموختید.



تابع GETDATE در SQL Server

تابع GETDATE() برچسب زمان فعلی سیستم را به‌عنوان یک مقدار DATETIME بدون offset منطقه زمانی پایگاه‌داده بازمی‌گرداند. مقدار DATETIME از سیستم‌عامل سرور به دست می‌آید که نمونه SQL Server در آن اجرا می‌شود.

کد زیر syntax تابع GETDATE() را نشان می‌دهد:

```
GETDATE()
```

توجه کنید که تابع GETDATE() یک تابع غیرمتمرکز است، ازاین‌رو، نمی‌توانید برای ستون‌هایی که این تابع را رفرنس کرده‌اند یک index در viewها ایجاد کنید.

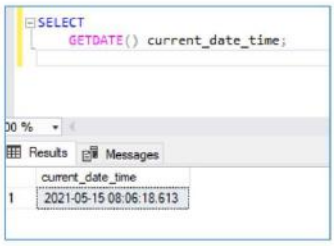
اجازه دهید چند مثال برای استفاده از تابع GETDATE() حل کنیم.

الف) مثال استفاده از تابع GETDATE() برای به‌دست‌آوردن تاریخ و زمان فعلی

این مثال از تابع GETDATE() برای بازگرداندن تاریخ و زمان فعلی سیستم عاملی که SQL Server روی آن اجرا می‌شود، استفاده می‌کند:

```
SELECT
    GETDATE() current_date_time;
```

خروجی به این شکل است:

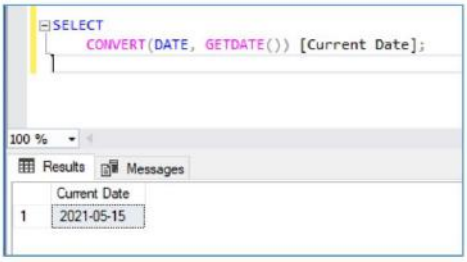


ب) مثال استفاده از تابع GETDATE() برای به‌دست‌آوردن تاریخ فعلی سیستم

برای به‌دست‌آوردن تاریخ فعلی، می‌توانید از تابع CONVERT() برای تبدیل مقدار DATETIME به یک مقدار DATE استفاده کنید که در کد زیر مشاهده می‌کنید:

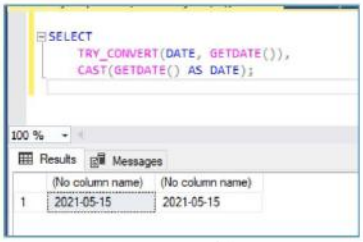
```
SELECT
    CONVERT(DATE, GETDATE()) [Current Date];
```

کد زیر نیز خروجی را نشان می‌دهد:



به طور مشابه، می‌توانید از توابع TRY_CONVERT() و CAST() برای تبدیل نتیجه تابع GETDATE() به یک تاریخ استفاده کنید:

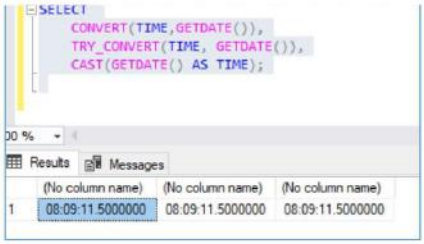
```
SELECT
    TRY_CONVERT(DATE, GETDATE()),
    CAST(GETDATE() AS DATE);
```



پ) مثال استفاده از تابع GETDATE() برای به‌دست‌آوردن زمان فعلی سیستم

برای به‌دست‌آوردن زمان فعلی، می‌توانید از تابع CONVERT()، TRY_CONVERT یا CAST() برای تبدیل نتیجه تابع GETDATE() به یک‌زمان استفاده کنید:

```
SELECT
    CONVERT(TIME,GETDATE()),
    TRY_CONVERT(TIME, GETDATE()),
    CAST(GETDATE() AS TIME);
```



به این شکل، چگونگی استفاده از تابع ()GETDATE برای بازگرداندن تاریخ و زمان فعلی سیستم را نیز آموختید.

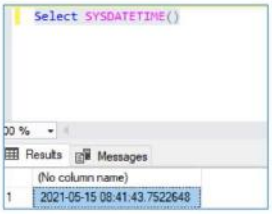
تابع SYSDATETIME در SQL Server

تابع ()SYSDATETIME یک مقدار DATETIME2 را بازمی‌گرداند که نشان‌دهنده تاریخ و زمان فعلی سروری است که نمونه SQL Server در آن اجرا می‌شود.

تابع ()SYSDATETIME هیچ پارامتری نمی‌گیرد:

```
SYSDATETIME()
```

خروجی به این شکل است:



توجه کنید که تابع ()SYSDATETIME دارای دقت ثانیه‌های اعشاری بیشتری نسبت به تابع ()GETDATE است.

()SYSDATETIME یک تابع غیرمتمرکز است، ازاین‌رو، viewها و ستون‌هایی که دارای عبارات رفرنسی به سمت این تابع هستند را نمی‌توان index گذاری کرد.

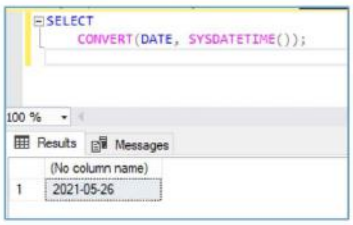
بیایید چند مثال برای استفاده از تابع ()SYSDATETIME حل کنیم.
الف) بازگرداندن تاریخ فعلی سیستم

این مثال از تابع ()CONVERT برای تبدیل نتیجه تابع ()SYSDATETIME به تاریخ فعلی استفاده می‌کند:

```
SELECT
```

```
CONVERT(DATE, SYSDATETIME());
```

نتیجه به این صورت است:

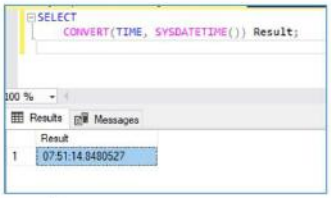


ب) بازگرداندن زمان فعلی سیستم

مثال زیر نتیجه تابع SYSDATETIME() را به زمان فعلی تبدیل می‌کند:

```
SELECT
    CONVERT(TIME, SYSDATETIME()) Result;
```

خروجی به این شکل است:



به این شکل، چگونگی استفاده از تابع SYSDATETIME() برای به‌دست‌آوردن تاریخ و زمان فعلی را نیز آموختید.

تابع SYSUTCDATETIME در SQL Server

تابع SYSTCDATETIME() یک مقدار DATETIME2 بازمی‌گرداند که تاریخ و زمان فعلی سروری که نمونه SQL Server روی آن اجرا می‌شود را نشان می‌دهد. Datetime به فرمت زمان هماهنگی جهانی (UTC) می‌باشد.

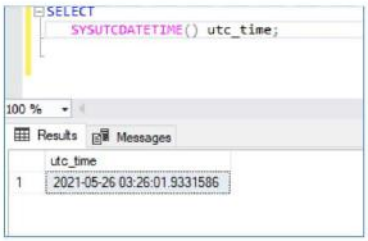
Syntax تابع SYSTCDATETIME() به‌صورت زیر است:

```
SYSUTCDATETIME()
```

برای مثال، برای به‌دست‌آوردن تاریخ و زمان فعلی در UTC، می‌توانید از کد زیر استفاده کنید:

```
SELECT
   SYSUTCDATETIME() utc_time;
```

خروجی به این صورت است:



توجه کنید که تابع SYSTCDATETIME() دارای دقت ثانیه‌های اعشاری بیشتری نسبت به تابع GETUTCDATE() است.

مثال‌های تابع SYSTCDATETIME() در SQL Server

الف) مثال بازگرداندن تاریخ فعلی به فرم زمان UTC

مثال زیر از تابع CONVERT() برای تبدیل نتیجه تابع SYSTCDATETIME() به تاریخ فعلی به فرم زمان UTC استفاده می‌کند:

```
SELECT
    CONVERT(DATE, SYSUTCDATETIME()) utc_date;
```

خروجی به این شکل است:

ب) مثال بازگرداندن زمان فعلی به فرم زمان UTC

این مثال نتیجه تابع SYSTCDATETIME() را به تاریخ فعلی به فرم زمان UTC تبدیل می‌کند:

```
SELECT
    CONVERT(DATE, SYSUTCDATETIME()) utc_time;
```

خروجی به این شکل است:

به این صورت چگونگی استفاده از تابع SYSTCDATETIME() برای به‌دست‌آوردن تاریخ و زمان فعلی به فرم زمان UTC را آموختید.

تابع SYSDATETIMEOFFSET در SQL Server

تابع SYSDATETIMEOFFSET() یک مقدار DATETIMEOFFSET(۷) را بازمی‌گرداند که تاریخ و زمان فعلی را نشان می‌دهد و همچنین شامل منطقه زمانی رایانه‌ای که نمونه SQL Server روی آن اجرا می‌شود نیز هست.

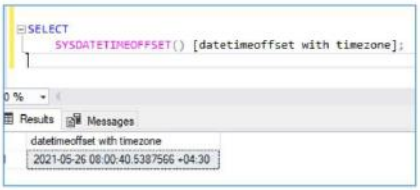
کد زیر syntax تابع SYSDATETIMEOFFSET() را نشان می‌دهد:

```
SYSDATETIMEOFFSET()
```

برای مثال، برای به‌دست‌آوردن تاریخ و زمان فعلی همراه با منطقه زمانی سروری که به آن متصل هستید، می‌توانید از کد زیر استفاده کنید:

```
SELECT
    SYSDATETIMEOFFSET() [datetimeoffset with timezone];
```

خروجی به این شکل است:



مثال‌های تابع SYSDATETIMEOFFSET() در SQL Server

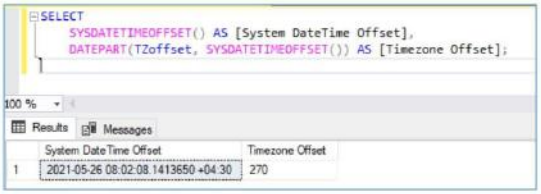
بیایید چند مثال برای استفاده از تابع SYSDATETIMEOFFSET() حل کنیم.

الف) مثال استخراج offset منطقه زمانی

مثال زیر از تابع DATEPART() برای بازگرداندن offset منطقه زمانی استفاده کرده است. این مثال یک integer بازمی‌گرداند که نشان‌دهنده offset منطقه زمانی به دقیقه است.

```
SELECT
    SYSDATETIMEOFFSET() AS [System DateTime Offset],
    DATEPART(TZoffset, SYSDATETIMEOFFSET()) AS [Timezone Offset];
```

خروجی به این شکل است:

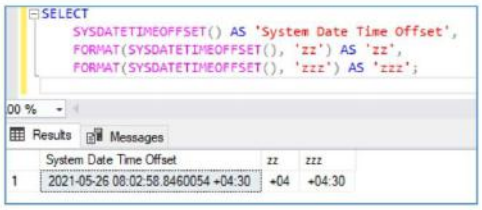


ب) مثال فرمت‌بندی offset منطقه زمانی

این مثال offset منطقه زمانی بازگشتی را با استفاده از تابع ()FORMAT و آرگومان‌های zz و zzz به شکل رشته بازمی‌گرداند:

```
SELECT
    SYSDATETIMEOFFSET() AS 'System Date Time Offset',
    FORMAT(SYSDATETIMEOFFSET(), 'zz') AS 'zz',
    FORMAT(SYSDATETIMEOFFSET(), 'zzz') AS 'zzz';
```

تصویر زیر، خروجی را نشان می‌دهد:



به این شکل، چگونگی استفاده از تابع SYSDATETIMEOFFSET() برای به‌دست‌آوردن تاریخ و زمان فعلی، همراه با منطقه زمانی را آموختید.

تابع DATENAME در SQL Server

تابع ()DATENAME یک‌رشته (نوع NVARCHAR) بازمی‌گرداند که نشان‌دهنده یک بخش تاریخ خاص است (یعنی سال، ماه و روز یک تاریخ خاص).

کد زیر syntax تابع ()DATENAME را نشان می‌دهد:

```
DATENAME(date_part,input_date)
```

تابع ()DATENAME دو آرگومان می‌پذیرد:

- Date_part بخشی از تاریخ است که می‌خواهید بازگشت داده شود. جدول زیر تمام مقادیر بخش تاریخ معتبر را لیست کرده است.
- Input_date یک تاریخ لفظی یا یک عبارت است که می‌توانیم آن را به یک مقدار DATETIME2، DATETIME، SMALLDATETIME، DATE، TIME یا DATETIMEOFFSET تبدیل کنیم.

Date_Part	اختصارات
سال	yy، yyyy
ربع سال	qq، q

ماه	mm, m
روز از سال	dy, y
روز	dd, d
هفته	wk, ww
روز از هفته	dw
ساعت	hh
دقیقه	mi, n
ثانیه	ss, s
میلی‌ثانیه	ms
میکروثانیه	mcs
نانوثانیه	ns
Tzoffset	tz
ISO_WEEK	isowk, isoww

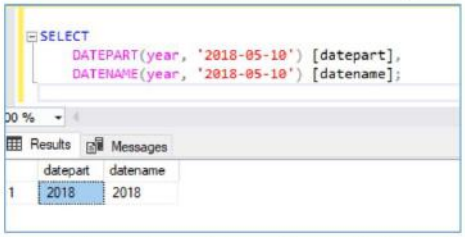
DATENAME() در برابر DATEPART()

توجه کنید که DATENAME() از لحاظ نوع بازگشتی، شبیه به DATEPART() است. تابع DATENAME() بخش تاریخ را به‌عنوان یک کاراکتر رشته بازمی‌گرداند، درحالی‌که DATEPART() بخش تاریخ را به‌عنوان یک integer بازمی‌گرداند.

مثال زیر را ببینید:

```
SELECT
    DATEPART(year, '2018-05-10') [datepart],
    DATENAME(year, '2018-05-10') [datename];
```

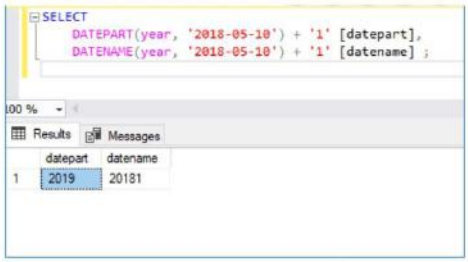
خروجی به‌مانند شکل زیر خواهد بود.



به‌هرحال، نوع‌های داده‌ای آن‌ها همان‌طور که در مثال زیر مشاهده می‌کنید، متفاوت هستند:

```
SELECT
    DATEPART(year, '2018-05-10') + '1' [datepart],
    DATENAME(year, '2018-05-10') + '1' [datename] ;
```

کد زیر نتیجه را نشان می‌دهد:



چون تابع DATEPART() یک integer بازمی‌گرداند، پس مقدار عبارت برابر با ۲۰۱۹ (۱ + ۲۰۱۸) می‌شود. به‌هرحال، تابع DATENAME() یک‌رشته کاراکتر بازمی‌گرداند، ازاین‌رو، علامت + به‌عنوان یک عملگر تلفیقی عمل می‌کند و نتیجه آن '۲۰۱۸۱' (یعنی ۱ + ۲۰۱۸) است.

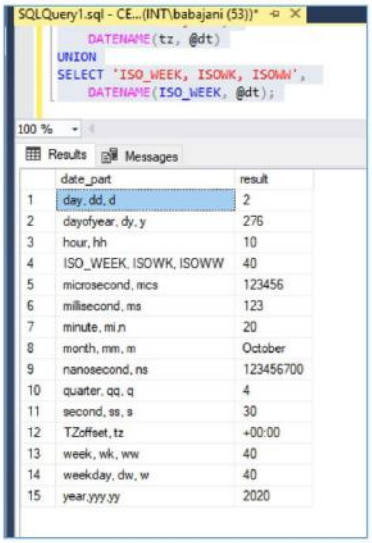
مثال تابع DATENAME() در SQL Server

این مثال از تابع DATENAME() برای بازگرداندن بخش‌های تاریخ مختلف از '۰۸:۱۰ + ۱۲۳٤٥٦٧. ۱۰:۲۰:۳۰ ۰۲-۱۰-۲۰۲۰' استفاده می‌کند:

```
DECLARE @dt DATETIME2= '2020-10-02 10:20:30.1234567 +08:10';

SELECT 'year,yyy,yy' date_part,
    DATENAME(year, @dt) result
UNION
SELECT 'quarter, qq, q',
    DATENAME(quarter, @dt)
UNION
SELECT 'month, mm, m',
    DATENAME(month, @dt)
UNION
SELECT 'dayofyear, dy, y',
    DATENAME(dayofyear, @dt)
UNION
SELECT 'day, dd, d',
    DATENAME(day, @dt)
UNION
SELECT 'week, wk, ww',
    DATENAME(week, @dt)
UNION
SELECT 'weekday, dw, w',
    DATENAME(week, @dt)
UNION
SELECT 'hour, hh' date_part,
    DATENAME(hour, @dt)
UNION
SELECT 'minute, mi,n',
    DATENAME(minute, @dt)
UNION
SELECT 'second, ss, s',
    DATENAME(second, @dt)
UNION
SELECT 'millisecond, ms',
    DATENAME(millisecond, @dt)
UNION
SELECT 'microsecond, mcs',
    DATENAME(microsecond, @dt)
```

```
UNION
SELECT 'nanosecond, ns',
    DATENAME(nanosecond, @dt)
UNION
SELECT 'TZoffset, tz',
    DATENAME(tz, @dt)
UNION
SELECT 'ISO_WEEK, ISOWK, ISOWW',
    DATENAME(ISO_WEEK, @dt);
```

خروجی به‌این‌ترتیب است:



	date_part	result
1	day, dd, d	2
2	dayofyear, dy, y	276
3	hour, hh	10
4	ISO_WEEK, ISOWK, ISOWW	40
5	microsecond, mcs	123456
6	millisecond, ms	123
7	minute, mi,n	20
8	month, mm, m	October
9	nanosecond, ns	123456700
10	quarter, qq, q	4
11	second, ss, s	30
12	TZoffset, tz	+00:00
13	week, wk, ww	40
14	weekday, dw, w	40
15	year,yyy,yy	2020

در نتیجه، چگونگی استفاده از تابع ()DATENAME برای استخراج یک بخش تاریخ به‌عنوان یک‌رشته کاراکتر از یک تاریخ را آموختید.

تابع DATEPART در SQL Server

تابع ()DATEPART یک integer بازمی‌گرداند که بخشی از یک تاریخ مانند روز، ماه و سال است.

کد زیر syntax تابع ()DATEPART را نشان می‌دهد:

```
DATEPART ( date_part , input_date )
```

()DATEPART دو آرگومان می‌گیرد:

- Date_part بخشی از یک تاریخ است که باید استخراج شود. (بخش‌های معتبر تاریخ را در جدول زیر مشاهده می‌کنید).

- Input_date تاریخی است که از آن بخش تاریخ استخراج می‌شود.

Date_part	اختصارات
سال	yy, yyyy
ربع سال	qq, q
ماه	mm, m
روز از سال	dy, y
روز	dd, d
هفته	wk, ww
روز از هفته	dw
ساعت	hh
دقیقه	mi, n
ثانیه	ss, s
میلی‌ثانیه	ms
میکروثانیه	mcs
نانوثانیه	ns
Tzoffset	tz
ISO_WEEK	isowk, isoww

مثال‌های DATEPART() در SQL Server

اجازه دهید چند مثال برای استفاده از تابع DATEPART() حل کنیم.

الف) استفاده از تابع DATEPART() همراه با متغیرها

این مثال از تابع DATEPART() برای استخراج بخش‌های مختلف یک مقدار تاریخ ذخیره شده در یک متغیر استفاده می‌کند:

```
DECLARE @d DATETIME = '2019-01-01 14:30:14';
SELECT
   DATEPART(year, @d) year,
   DATEPART(quarter, @d) quarter,
   DATEPART(month, @d) month,
   DATEPART(day, @d) day,
   DATEPART(hour, @d) hour,
   DATEPART(minute, @d) minute,
   DATEPART(second, @d) second;
```

خروجی به‌این‌ترتیب است:

```
DECLARE @d DATETIME = '2019-01-01 14:30:14';
SELECT
   DATEPART(year, @d) year,
   DATEPART(quarter, @d) quarter,
   DATEPART(month, @d) month,
   DATEPART(day, @d) day,
   DATEPART(hour, @d) hour,
   DATEPART(minute, @d) minute,
   DATEPART(second, @d) second;
```

100 %

Results | Messages

	year	quarter	month	day	hour	minute	second
1	2019	1	1	1	14	30	14

ب) مثال استفاده از تابع DATEPART() همراه با ستون‌های جدول

از جداول sales.orders و sales.order_items در پایگاه‌داده Bikestores برای این مثال استفاده می‌کنیم:

مثال زیر از تابع DATEPART() برای به‌دست‌آوردن نرخ رشد فروش بر اساس سال، ربع سال، ماه و روز استفاده می‌کند.

```
SELECT DATEPART(year, shipped_date) [year],
       DATEPART(quarter, shipped_date) [quarter],
       DATEPART(month, shipped_date) [month],
       DATEPART(day, shipped_date) [day],
       SUM(quantity * list_price) gross_sales
FROM sales.orders o
     INNER JOIN sales.order_items i ON i.order_id = o.order_id
WHERE shipped_date IS NOT NULL
GROUP BY DATEPART(year, shipped_date),
         DATEPART(quarter, shipped_date),
         DATEPART(month, shipped_date),
         DATEPART(day, shipped_date)
ORDER BY [year] DESC,
         [quarter],
         [month],
         [day];
```

بخشی از خروجی به شکل زیر است:

```
SELECT DATEPART(year, shipped_date) [year],
       DATEPART(quarter, shipped_date) [quarter],
       DATEPART(month, shipped_date) [month],
       DATEPART(day, shipped_date) [day],
       SUM(quantity * list_price) gross_sales
FROM sales.orders o
     INNER JOIN sales.order_items i ON i.order_id = o.order_id
WHERE shipped_date IS NOT NULL
GROUP BY DATEPART(year, shipped_date),
```

00 %

Results | Messages

	year	quarter	month	day	gross_sales
1	2018	1	1	1	3259.96
2	2018	1	1	2	11963.92
3	2018	1	1	3	20810.85
4	2018	1	1	5	8819.93
5	2018	1	1	6	5126.94
6	2018	1	1	7	17117.84
7	2018	1	1	8	11869.93
8	2018	1	1	9	2909.94
9	2018	1	1	10	21789.92
10	2018	1	1	12	20839.92
11	2018	1	1	13	20919.92
12	2018	1	1	14	26655.85
13	2018	1	1	15	40799.80
14	2018	1	1	16	21097.95
15	2018	1	1	17	31089.89

در این مثال، ما از تابع DATEPART() برای استخراج سال، ربع سال، ماه و روز از مقادیر موجود در ستون shipped_date استفاده کردیم. در دستور GROUP BY، ما نرخ‌های رشد فروش (quuantity * list_price) را با استفاده از این بخش‌های تاریخ جمع‌آوری کردیم.

توجه کنید که می‌توانید از تابع DATEPART() در دستورهای GROUP BY، HAVING، WHERE، SELECT و ORDER BY استفاده کنید.

تابع DAY در SQL Server

تابع DAY() یک مقدار integer بازمی‌گرداند که روز یک ماه (از ۱ تا ۳۱) از یک تاریخ مشخص شده را نشان می‌دهد.

کد زیر syntax تابع DAY() را نشان می‌دهد:

```
DAY(input_date)
```

تابع DAY() یک آرگومان می‌گیرد که این آرگومان می‌تواند یک تاریخ یا یک عبارت باشد که به‌صورت مقدار DATETIME2، DATETIME، SMALLDATETIME، DATE، TIME یا DATETIMEOFFSET استفاده شود.

تابع DAY() مقدار مشابه با تابع DATEPART() که در زیر مشاهده می‌کنید بازمی‌گرداند:

```
DATEPART(day,input_date)
```

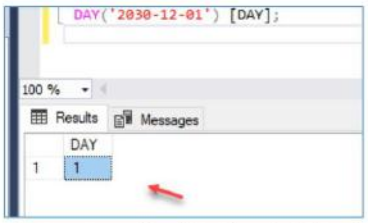
مثال‌های تابع DAY() در SQL Server

الف) مثال استفاده از تابع DAY() با یک مقدار تاریخ

این مثال از تابع DAY() برای استخراج روز از تاریخ '۲۰۳۰-۱۲-۰۱' استفاده می‌کند:

```
SELECT
 DAY('2030-12-01') [DAY];
```

خروجی به این شکل است:



ب) مثال استفاده از تابع DAY() با یک مقدار تاریخ که فقط دارای بخش زمان است

تابع DAY() در صورتی که تاریخ ورودی شامل فقط بخش زمان باشد، مقدار ۱ را بازمی‌گرداند:

```
SELECT
 DAY('10:20:30') [DAY];
```

کد زیر، خروجی را نشان می‌دهد:

پ) مثال استفاده از تابع DAY() با ستون‌های جدول

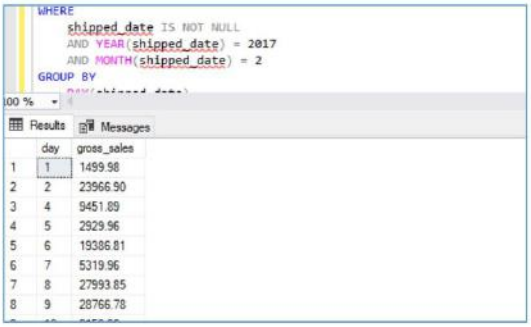
از جداول sales.orders و sales.order_items برای مثال استفاده می‌کنیم:

در این مثال از تابع DAY() برای استخراج داده روز از مقادیر موجود در ستون shipperd_date استفاده شده است. این مثال با استفاده از تابع SUM() و دستور GROUP BY، نرخ‌های رشد فروش را بر اساس روز در فوریه ۲۰۱۷ بازمی‌گرداند:

```
SELECT
    DAY(shipped_date) [day],
    SUM(list_price * quantity) gross_sales
```

```
FROM
    sales.orders o
    INNER JOIN sales.order_items i ON i.order_id = o.order_id
WHERE
    shipped_date IS NOT NULL
    AND YEAR(shipped_date) = 2017
    AND MONTH(shipped_date) = 2
GROUP BY
    DAY(shipped_date)
ORDER BY
    [day];
```

تصویر زیر، خروجی را نشان می‌دهد:



تابع MONTH در SQL Server

تابع MONTH() یک مقدار integer بازمی‌گرداند که نشان‌دهنده ماه از یک تاریخ مشخص شده است.

کد زیر syntax تابع MONTH() را نشان می‌دهد:

```
MONTH(input_date)
```

تابع MONTH() یک آرگومان می‌گیرد که می‌تواند یک مقدار تاریخ یا یک عبارت باشد که دارای مقدار DATETIME2، DATETIME، SMALLDATETIME، DATE، TIME یا DATETIMEOFFSET است.

تابع MONTH() مقدار مشابهی با تابع DATEPART() بازمی‌گرداند:

```
DATEPART(month,input_date)
```

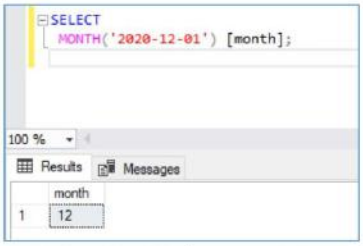
مثال‌های تابع MONTH() در SQL Server

الف) استفاده از تابع MONTH() با یک مقدار تاریخ

این مثال از تابع MONTH() برای استخراج ماه از تاریخ '۲۰۲۰-۱۲-۰۱' استفاده می‌کند:

```
SELECT
 MONTH('2020-12-01') [month];
```

خروجی به این شکل است:

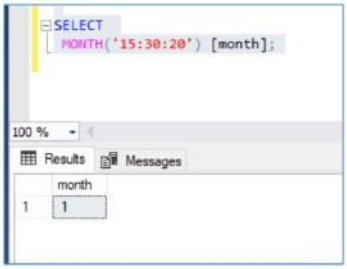


ب) استفاده از تابع MONTH() با یک مقدار تاریخ که فقط دارای داده زمان است

تابع MONTH() در صورتی که مقدار تاریخ شامل فقط بخش زمان باشد، مقدار ۱ را بازمی‌گرداند:

```
SELECT
 MONTH('15:30:20') [month];
```

خروجی به‌این‌ترتیب است:



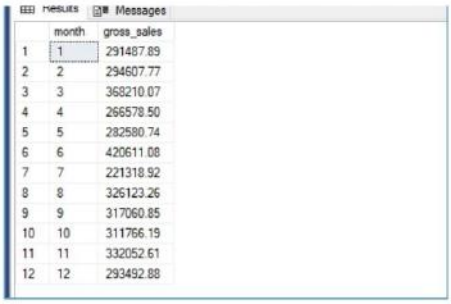
پ) استفاده از تابع MONTH() با ستون‌های جدول

طبق معمول از جداول Sales.orders و Sales.order_items برای مثال استفاده می‌کنیم.

این مثال از تابع MONTH() برای استخراج داده ماه از مقادیر موجود در ستون shipped_date استفاده می‌کند. نتیجه این مثال، نرخ رشد فروش بر اساس ماه در سال ۲۰۱۸ است که با استفاده از تابع SUM() و دستور GROUP BY به دست می‌آیند:

```
SELECT MONTH(shipped_date) [month],
    SUM(list_price * quantity) gross_sales
FROM sales.orders o
    INNER JOIN sales.order_items i ON i.order_id = o.order_id
WHERE shipped_date IS NOT NULL
    AND YEAR(shipped_date) = 2017
GROUP BY MONTH(shipped_date)
ORDER BY [month];
```

تصویر زیر، خروجی را نشان می‌دهد:



	month	gross_sales
1	1	291487.89
2	2	294607.77
3	3	368210.07
4	4	266578.50
5	5	282580.74
6	6	420611.08
7	7	221318.92
8	8	326123.26
9	9	317060.85
10	10	311766.19
11	11	332052.61
12	12	293492.88

تابع YEAR در SQL Server

تابع ()YEAR یک مقدار integer بازمی‌گرداند که نشان‌دهنده سال از یک تاریخ مشخص شده است.

کد زیر syntax تابع ()YEAR را نشان می‌دهد:

```
YEAR(input_date)
```

تابع ()YEAR یک آرگومان می‌گیرد که می‌تواند یک مقدار تاریخ یا یک عبارت باشد که دارای مقدار DATETIME2، DATETIME، SMALLDATETIME، DATE، TIME یا DATETIMEOFFSET است.

تابع ()YEAR مقدار مشابهی با تابع ()DATEPART بازمی‌گرداند:

```
DATEPART(year,input_date)
```

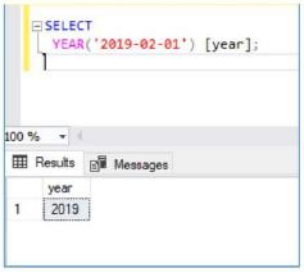
مثال‌های تابع ()YEAR در SQL Server

الف) استفاده از تابع ()YEAR با یک مقدار تاریخ

این مثال از تابع ()YEAR برای استخراج سال از تاریخ '۲۰۱۹-۰۲-۰۱' استفاده می‌کند:

```
SELECT
 YEAR('2019-02-01') [year];
```

خروجی به این شکل است:

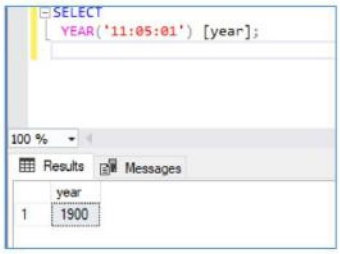


ب) استفاده از تابع YEAR() با یک مقدار تاریخ که فقط دارای بخش زمان است

اگر مقدار داده ورودی فقط دارای داده زمان باشد، تاب YEAR() مقدار ۱۹۰۰ را بازمی‌گرداند:

```
SELECT
 YEAR('11:05:01') [year];
```

خروجی به‌این‌ترتیب است:



پ) استفاده از تابع YEAR() با ستون‌های جدول

باز هم از جداول Sales.orders و Sales.order_items برای مثال استفاده می‌کنیم.

این مثال از تابع YEAR() برای استخراج داده سال از مقادیر موجود در ستون shipped_date استفاده می‌کند. نتیجه این مثال، نرخ رشد فروش بر اساس سال است که با استفاده از تابع SUM() و دستور GROUP BY به دست می‌آیند:

```
SELECT YEAR(shipped_date) [year],
    SUM(list_price * quantity) gross_sales
FROM sales.orders o
 INNER JOIN sales.order_items i ON i.order_id = o.order_id
WHERE shipped_date IS NOT NULL
GROUP BY YEAR(shipped_date)
order by [year];
```

کد زیر، خروجی را نشان می‌دهد:

```
SELECT YEAR(shipped_date) [year],
    SUM(list_price * quantity) gross_sales
FROM sales.orders o
 INNER JOIN sales.order_items i ON i.order_id = o.order_id
WHERE shipped_date IS NOT NULL
GROUP BY YEAR(shipped_date)
order by [year];
```

100 %

Results | Messages

	year	gross_sales
1	2016	2649649.97
2	2017	3725890.76
3	2018	1062469.33

تابع DATEDIFF در SQL Server

برای محاسبه تفاوت بین دو تاریخ از لحاظ سال، ماه، هفته و غیره از تابع ()DATEDIFF استفاده می‌کنیم:

DATEDIFF(date_part , start_date , end_date)

تابع ()DATEDIFF سه آرگومان می‌پذیرد: start_date، date_part و end_date.

- Date_part بخشی از تاریخ (مثلاً یک سال، یک ربع سال، یک ماه، یک هفته و غیره) است که می‌خواهید اختلاف بر اساس آن بین start_date و end_date محاسبه شود. می‌توانید بخش‌های تاریخ معتبر و قابل‌قبول را در جدول زیر مشاهده کنید.

- Start_date و end_date تاریخ‌هایی هستند که باید با هم مقایسه شوند. آن‌ها باید دارای مقادیری از نوع SMALLATETIME، DATETIME2، DATETIMEOFFSET، DATETIME، DATE یا TIME باشند.

جدول زیر لیستی از مقادیر معتبر و قابل‌قبول برای date_part را نشان می‌دهد:

Date_part	اختصارات
سال	yy, yyyy
ربع سال	qq, q
ماه	mm, m
روز از سال	dy, y
روز	dd, d
هفته	wk, ww
ساعت	hh

دقیقه	mi, n
ثانیه	ss, s
میلی‌ثانیه	ms
میکروثانیه	mcs
نانوثانیه	ns

تابع DATEDIFF() یک مقدار integer را بازمی‌گرداند که این مقدار نشان‌دهنده تفاوت بین start_date و end_date است که واحد مشخص‌کننده نوع این تفاوت توسط Date_part مشخص می‌شود.

اگر نتیجه به‌دست‌آمده خارج از بازه اعداد integer باشد (یعنی خارج از بازه ۲,۱۴۷,۴۸۳,۶۴۸- الی ۲,۱۴۷,۴۸۳,۶۴۷+)، آنگاه تابع DATEDIFF() ارور می‌دهد. در چنین وضعیتی، باید در عوض از تابع DATEDIFF_BIG() استفاده کنید.

مثال‌های تابع DATEDIFF() در SQL Server

الف) استفاده از تابع DATEDIFF() برای مقایسه تفاوت‌های بین دو مقدار تاریخ

این مثال از تابع DATEDIFF() برای مقایسه تفاوت بین دو تاریخ که دارای بخش‌های مختلفی از تاریخ هستند، استفاده می‌کند:

```
DECLARE
    @start_dt DATETIME2= '2019-12-31 23:59:59.9999999',
    @end_dt DATETIME2= '2020-01-01 00:00:00.0000000';

SELECT
    DATEDIFF(year, @start_dt, @end_dt) diff_in_year,
    DATEDIFF(quarter, @start_dt, @end_dt) diff_in_quarter,
    DATEDIFF(month, @start_dt, @end_dt) diff_in_month,
    DATEDIFF(dayofyear, @start_dt, @end_dt) diff_in_dayofyear,
    DATEDIFF(day, @start_dt, @end_dt) diff_in_day,
    DATEDIFF(week, @start_dt, @end_dt) diff_in_week,
    DATEDIFF(hour, @start_dt, @end_dt) diff_in_hour,
    DATEDIFF(minute, @start_dt, @end_dt) diff_in_minute,
    DATEDIFF(second, @start_dt, @end_dt) diff_in_second,
    DATEDIFF(millisecond, @start_dt, @end_dt) diff_in_millisecond;
```

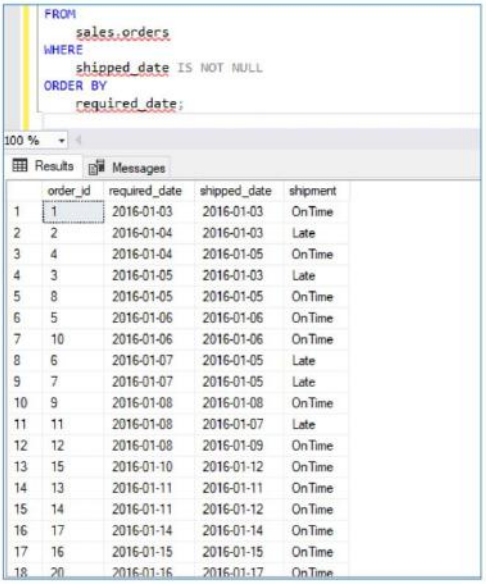
خروجی به این شکل است:

ب) مثال استفاده از تابع DATEDIFF() همراه با ستون جدول

مثال زیر از تابع DATEDIFF() برای مقایسه تاریخ دریافت درخواستی با تاریخ ارسال (به‌روز) استفاده کرده و نتیجه‌ای که بازمی‌گرداند بیان می‌کند که آیا سفارش به‌موقع تحویل داده شده یا خیر.

```
SELECT
    order_id,
    required_date,
    shipped_date,
    CASE
        WHEN DATEDIFF(day, required_date, shipped_date) < 0
        THEN 'Late'
        ELSE 'OnTime'
    END shipment
FROM
    sales.orders
WHERE
    shipped_date IS NOT NULL
ORDER BY
    required_date;
```

خروجی به‌این‌ترتیب است:



	order_id	required_date	shipped_date	shipment
1	1	2016-01-03	2016-01-03	OnTime
2	2	2016-01-04	2016-01-03	Late
3	4	2016-01-04	2016-01-05	OnTime
4	3	2016-01-05	2016-01-03	Late
5	8	2016-01-05	2016-01-05	OnTime
6	5	2016-01-06	2016-01-06	OnTime
7	10	2016-01-06	2016-01-06	OnTime
8	6	2016-01-07	2016-01-05	Late
9	7	2016-01-07	2016-01-05	Late
10	9	2016-01-08	2016-01-08	OnTime
11	11	2016-01-08	2016-01-07	Late
12	12	2016-01-08	2016-01-09	OnTime
13	15	2016-01-10	2016-01-12	OnTime
14	13	2016-01-11	2016-01-11	OnTime
15	14	2016-01-11	2016-01-12	OnTime
16	17	2016-01-14	2016-01-14	OnTime
17	16	2016-01-15	2016-01-15	OnTime
18	20	2016-01-16	2016-01-17	OnTime

تابع DATEADD در SQL Server

تابع DATEADD() یک عدد به یک بخش خاص از تاریخ یک مقدار تاریخ ورودی اضافه می‌کند و مقدار اصلاح شده را بازمی‌گرداند.

کد زیر syntax تابع DATEADD() را نشان می‌دهد:

DATEADD (date_part , value , input_date)

تابع ()DATEADD سه آرگومان می‌پذیرد:

- Date_part بخشی از تاریخ است که تابع ()DATEADD به آن value اضافه می‌کند. اگر value یک مقدار اعشاری یا float باشد، آنگاه تابع ()DATEADD بخش اعشاری را نادیده می‌گیرد. در چنین وضعیتی، تابع عدد را گرد نمی‌کند.

- Input_date یک تاریخ ورودی یا یک عبارت است که می‌تواند یک مقدار با نوع SMALLATETIME، DATETIME2، DATETIMEOFFSET، DATETIME، DATE یا TIME باشد.

جدول زیر مقادیر معتبر و قابل‌قبول date_part را نمایش می‌دهد:

Date_part	اختصارات
سال	yy, yyyy
ربع سال	qq, q
ماه	mm, m
روز از سال	dy, y
روز	dd, d
هفته	wk, ww
ساعت	hh
دقیقه	mi, n
ثانیه	ss, s
میلی‌ثانیه	ms
میکروثانیه	mcs
نانوثانیه	ns

تابع ()DATEADD بعد از اضافه‌کردن value به date_part، یک مقدار تاریخ جدید را بازمی‌گرداند.

مثال‌های تابع ()DATEADD در SQL Server

اضافه‌کردن ۱ ثانیه به ۲۰۱۸-۱۲-۳۱ ۲۳:۵۹:۵۹

```
SELECT
```

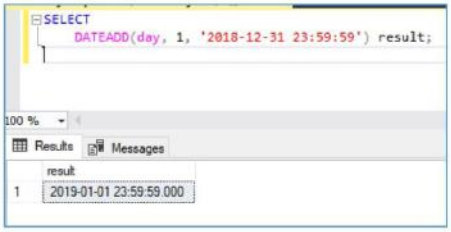
```
    DATEADD(second, 1, '2018-12-31 23:59:59') result;
```

خروجی به این شکل است:

اضافه‌کردن ۱ روز به ۲۰۱۸-۱۲-۳۱ ۰۰:۰۰:۰۰s

```
SELECT
    DATEADD(day, 1, '2018-12-31 23:59:59') result;
```

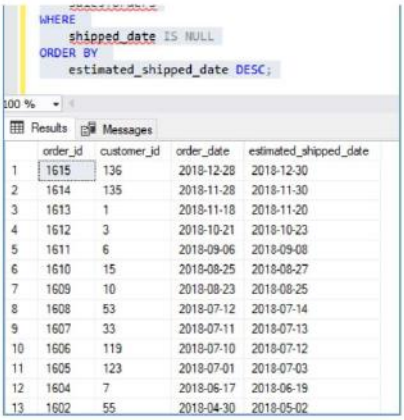
خروجی:



این مثال از تابع DATEADD() برای محاسبه تاریخ ارسال تخمینی بر اساس تاریخ سفارش، استفاده می‌کند:

```
SELECT
    order_id,
    customer_id,
    order_date,
    DATEADD(day, 2, order_date) estimated_shipped_date
FROM
    sales.orders
WHERE
    shipped_date IS NULL
ORDER BY
    estimated_shipped_date DESC;
```

تصویر زیر، بخشی از خروجی را نمایش می‌دهد:



	order_id	customer_id	order_date	estimated_shipped_date
1	1615	136	2018-12-28	2018-12-30
2	1614	135	2018-11-28	2018-11-30
3	1613	1	2018-11-18	2018-11-20
4	1612	3	2018-10-21	2018-10-23
5	1611	6	2018-09-06	2018-09-08
6	1610	15	2018-08-25	2018-08-27
7	1609	10	2018-08-23	2018-08-25
8	1608	53	2018-07-12	2018-07-14
9	1607	33	2018-07-11	2018-07-13
10	1606	119	2018-07-10	2018-07-12
11	1605	123	2018-07-01	2018-07-03
12	1604	7	2018-06-17	2018-06-19
13	1602	55	2018-04-30	2018-05-02

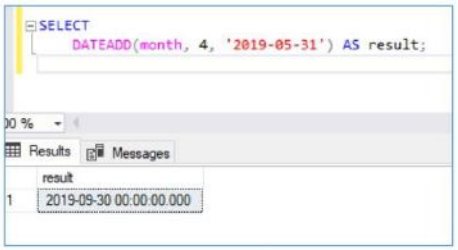
مثال‌های کار با ماه

اگر یک رقم ماه به یک تاریخ اضافه کنید و مقدار روز تاریخ به‌دست‌آمده وجود نداشته باشد، آنگاه تابع DATEADD() آخرین روز ماه بازگشتی را بازمی‌گرداند.

به مثال زیر توجه کنید:

```
SELECT
    DATEADD(month, 4, '2019-05-31') AS result;
```

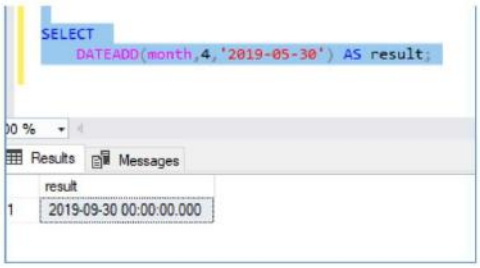
در این مثال، مقدار ماه از تاریخ بازگشتی، ماه سپتامبر است. به‌هرحال، روز ۳۱ در ماه سپتامبر وجود ندارد، ازاین‌رو، تابع DATEADD() آخرین روز ماه سپتامبر (یعنی ۳۰) را به‌عنوان مقدار روز برای تاریخ بازگشتی نمایش می‌دهد:



توجه کنید که query زیر نیز همان نتیجه بالا را بازمی‌گرداند:

```
SELECT
    DATEADD(month,4,'2019-05-30') AS result;
```

خروجی:



تابع EOMONTH در SQL Server

تابع ()EOMONTH آخرین روز ماه یک تاریخ خاص را همراه با یک offset اختیاری، بازمی‌گرداند.

کد زیر syntax تابع ()EOMONTH را نشان می‌دهد:

```
EOMONTH(start_date [, offset]);
```

تابع ()EOMONTH دو آرگومان می‌پذیرد:

- Start_date یک عبارت تاریخ است که مقدار داخل آن نیز یک تاریخ است. تابع ()EOMONTH آخرین روز ماه را برای این تاریخ بازمی‌گرداند.
- Offset یک integer است که عدد ماه‌هایی که باید به Start_date اضافه شوند را مشخص می‌کند.

اگر نتیجه اضافه‌شدن offset به start_date یک مقدار غیرمعتبر یا غیرقابل‌قبول باشد، آنگاه تابع ()EOMONTH ارور می‌دهد.

مثال‌های ()EOMONTH در SQL Server

الف) استفاده از تابع ()EOMONTH برای یک تاریخ

این مثال از تابع ()EOMONTH برای بازگرداندن آخرین روز ماه برای تاریخ ۱۵-۰۲-۲۰۱۹ استفاده می‌کند:

```
SELECT
    DATEADD(month,4,'2019-05-30') AS result;
```

خروجی به این شکل است:

اگر تاریخ یک سال کبیسه را به تابع EOMONTH() بدهید، بازهم مقدار صحیح را بازمی‌گرداند:

```
SELECT
    EOMONTH('2020-02-09') end_of_month_feb2020;
```

خروجی به این صورت است:

همان‌طور که به‌وضوح در خروجی مشاهده می‌کنید، آخرین روز ماه فوریه ۲۰۱۹ روز ۲۹ام است، نه ۲۸ام.

ب) استفاده از تابع EOMONTH() برای به‌دست‌آوردن تعداد روزها در یک ماه خاص

برای به‌دست‌آوردن تعداد روزهای یک ماه خاص، مراحل زیر را انجام دهید:

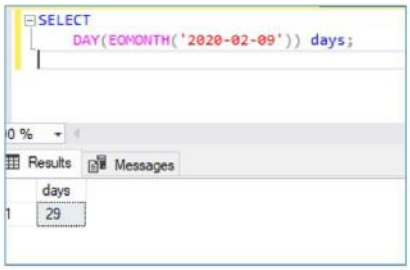
ابتدا، از تابع EOMONTH() برای به‌دست‌آوردن آخرین روز ماه استفاده کنید.

سپس، آخرین روز ماه را به تابع DAY() بفرستید.

این مثال تعداد روزهای ماه فوریه سال ۲۰۱۸ را بازمی‌گرداند:

```
SELECT
    DAY(EOMONTH('2020-02-09')) days;
```

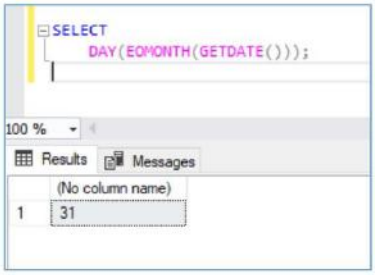
خروجی به این شکل است:



برای به‌دست‌آوردن تعداد روزها در ماه جاری، می‌توانید از کد زیر استفاده کنید:

```
SELECT
```

```
DAY(EOMONTH(GETDATE()));
```

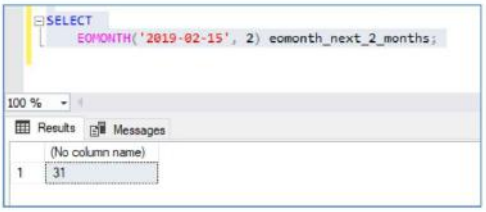


پ) مثال استفاده از تابع EOMONTH() با یک offset

مثال زیر از تابع EOMONTH() همراه با یک offset دوماهه استفاده می‌کند:

```
SELECT
    EOMONTH('2019-02-15', 2) eomonth_next_2_months;
```

خروجی به شکل زیر است:



تابع SWITCHOFFSET در SQL Server

تابع SWITCHOFFSET() یک DATETIMEOFFSET که از offset منطقه زمانی ذخیره شده به یک offset منطقه زمانی جدید تغییریافته است را بازمی‌گرداند.

کد زیر syntax تابع SWITCHOFFSET() را نشان می‌دهد:

```
SWITCHOFFSET(expression, time_zone )
```

تابع SWITCHOFFSET() دو آرگومان می‌پذیرد:

- Expression یک عبارت است که می‌تواند یک مقدار DATETIMEOFFSET باشد.
- Time_zone می‌توانید یک‌رشته کاراکتر به فرمت {+|-}TZH:TZM یا یک integer علامت‌دار از دقیقه‌ها باش. برای مثال، time_zone می‌تواند به شکل ۰۸:۰۰+,۰۷:۰۰- یا ۱۲۰ باشد.

تابع SWITCHOFFSET() یک تابع بسیار مفید و کارآمد برای به‌روزرسانی مقادیر در یک ستون DATETIMEOFFSET است.

مثال تابع SWITCHOFFSET() در SQL Server

ابتدا، یک جدول جدید ایجاد می‌کنیم که دارای ستون DATETIMEOFFSET باشد:

```
CREATE TABLE dbo.switchoffset_demo(
    dtz DATETIMEOFFSET
);
```

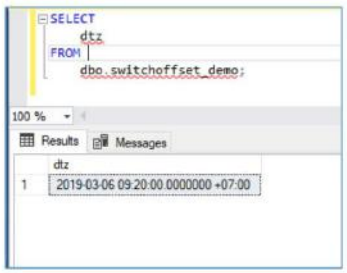
دوم، یک مقدار DATETIMEOFFSET جدید به جدول اضافه می‌کنیم:

```
INSERT INTO dbo.switchoffset_demo
VALUES('2019-03-06 9:20:00 +07:00');
```

سوم، برای به‌دست‌آوردن مقدار جدول dbo.switchoffset_demo یک query می‌نویسیم:

```
SELECT
    dtz
FROM
    dbo.switchoffset_demo;
```

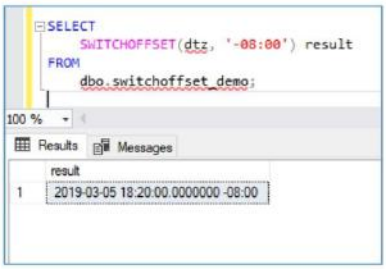
خروجی به این شکل است:



چهارم، از تابع SWITCHOFFSET() برای تغییر منطقه زمانی به ۰۸:۰۰- استفاده می‌کنیم:

```
SELECT
    SWITCHOFFSET(dtz, '-08:00') result
FROM
    dbo.switchoffset_demo;
```

خروجی به‌این‌ترتیب می‌شود:



تابع TODATETIMEOFFSET در SQL Server

تابع ()TODATETIMEOFFSET یک مقدار DATETIME2 را به یک مقدار DATETIMEOFFSET ترجمه می‌کند.

کد زیر syntax تابع ()TODATETIMEOFFSET را نشان می‌دهد:

```
TODATETIMEOFFSET(expression,time_zone)
```

تابع ()TODATETIMEOFFSET دو آرگومان می‌پذیرد:

- Expression یک عبارت است که می‌تواند به مقدار DATETIME2 باشد.
- Time_zone می‌توانید یک integer یا string باشد. اگر از یک integer استفاده کنید، آنگاه time_zone یک offset منطقه زمانی به دقیقه می‌باشد. در غیر این صورت، یک ساعت یا دقیقه به فرم 'TZH:THM+' یا 'TZH:THM-' است که در آن TZH ساعت و THM دقیقه است. بازه ساعت از ١٤- تا ١٤+ است.

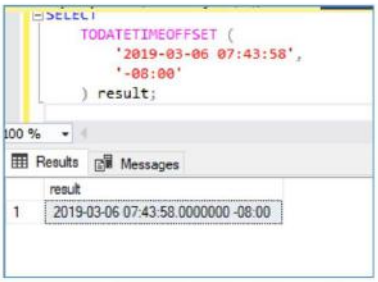
مثال‌های تابع ()TODATETIMEOFFSET در SQL Server

الف) مثال استفاده از تابع ()TODATETIMEOFFSET برای تغییر offset محدوده زمانی یک تاریخ و زمان

این مثال، offset منطقه یک تاریخ و زمان را به منطقه ٠٨:٠٠- تغییر می‌دهد.

```
SELECT
    TODATETIMEOFFSET (
        '2019-03-06 07:43:58',
        '-08:00'
    ) result;
```

خروجی به‌این‌ترتیب است:

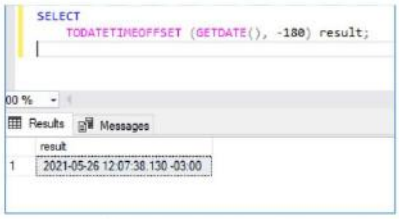


ب) مثال استفاده از تابع ()TODATETIMEOFFSET برای تغییر offset محدوده زمانی به دقیقه

این مثال، محدوده زمانی فعلی را به ١٨٠- دقیقه تغییر می‌دهد:

```
SELECT
    TODATETIMEOFFSET (GETDATE(), -180) result;
```

خروجی به این شکل است:

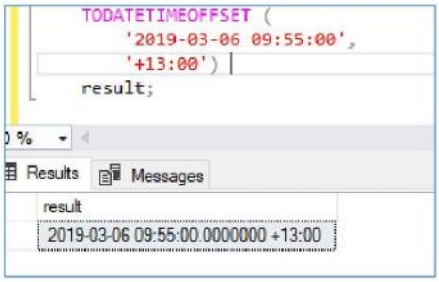


پ) مثال استفاده از تابع ()TODATETIMEOFFSET برای اضافه‌کردن یک offset منطقه زمانی ساعت به تاریخ و زمان فعلی

مثال زیر یک offset منطقه زمانی ۱۲ ساعته را به تاریخ و زمان '۲۰۱۹-۰۳-۰۶ ۰۹:۵۵:۰۰' اضافه می‌کند:

```
SELECT
    TODATETIMEOFFSET (
        '2019-03-06 09:55:00',
        '+13:00')
    result;
```

خروجی به‌این‌ترتیب است:



تابع DATEFROMPARTS در SQL Server

تابع ()DATEFROMPARTS یک مقدار DATE بازمی‌گرداند که متناظر با مقادیر سال، ماه و روز است.

کد زیر، syntax تابع ()DATEFROMPARTS را نمایش می‌دهد:

```
DATEFROMPARTS(year, month, day)
```

تابع ()DATEFROMPARTS سه آرگومان می‌پذیرد:

- Year یک عبارت integer است که از نوع سال است.
- Month یک عبارت integer است که مقدار آن یک ماه از بازه ۱ تا ۱۲ است.
- Day یک عبارت integer است که یک روز را از بازه ۱ تا ۳۱ مشخص می‌کند.

تابع ()DATEFROMPARTS یک مقدار DATE بازمی‌گرداند. اگر هرکدام از آرگومان‌ها NULL باشند، تابع مقدار NULL بازمی‌گرداند.

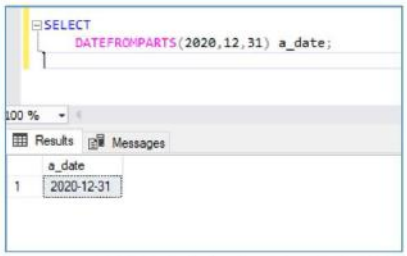
مثال‌های تابع ()DATEFROMPARTS در SQL Server

الف) مثال استفاده از تابع ()DATEFROMPARTS برای بازگرداندن یک تاریخ

این مثال از تابع ()DATEFROMPARTS برای ساخت یک تاریخ متشکل از مقادیر سال، ماه و روز استفاده می‌کند:

```
SELECT
    DATEFROMPARTS(2020,12,31) a_date;
```

خروجی به این شکل است:

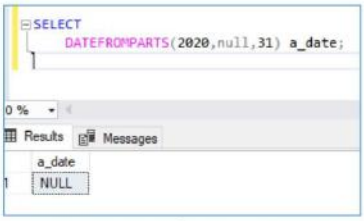


ب) مثال استفاده از تابع ()DATEFROMPARTS همراه با مقدار NULL

مثال زیر مقدار NULL بازمی‌گرداند، چون آرگومان ماه برابر با NULL است:

```
SELECT
    DATEFROMPARTS(2020,null,31) a_date;
```

خروجی به این صورت است:

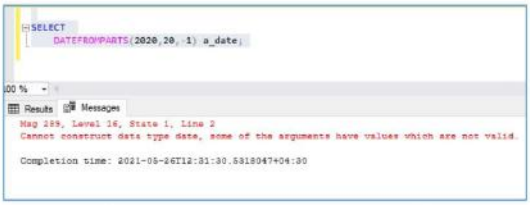


پ) مثال استفاده از تابع ()DATEFROMPARTS همراه با آرگومان‌های خارج از بازه

این مثال آرگومان‌های غیرمعتبر و غیرقابل‌قبول را به تابع ()DATEFROMPARTS ارسال می‌کند که در نتیجه سیستم ارور می‌دهد:

```
SELECT
    DATEFROMPARTS(2020,20,-1) a_date;
```

خط زیر، پیغام ارور را نشان می‌دهد:



به این شکل، چگونگی ساخت یک مقدار DATE از بخش‌های سال، ماه و روز با استفاده از تابع DATEFROMPARTS() را آموختید.

تابع DATETIME2FROMPARTS در SQL Server

تابع DATETIME2FROMPARTS() یک مقدار تاریخ بازمی‌گرداند که از مقادیر سال، ماه، روز، ساعت، دقیقه، ثانیه، اعشار و دقت است. در واقع این تابع یک مقدار DATETIME2 ایجاد می‌کند.
کد زیر، syntax تابع DATETIME2FROMPARTS() را نشان می‌دهد:

DATETIME2FROMPARTS (year, month, day, hour, minute, seconds, fractions, precision)

تابع DATETIME2FROMPARTS() هشت آرگومان می‌پذیرد:

- Year یک عبارت integer است که از نوع سال است.
- Month یک عبارت integer است که از نوع ماه و در بازه ۱ تا ۱۲ است.
- Day یک عبارت integer است که یک روز از بازه ۱ تا ۳۱ را مشخص می‌کند.
- Hour یک عبارت integer است که ساعت را مشخص می‌کند.
- Minute یک عبارت integer است که دقیقه را مشخص می‌کند.
  - Seconds یک عبارت integer است که ثانیه‌ها را مشخص می‌کند.
  - Fractions یک عبارت integer است که اعداد قسمت اعشار را نمایش می‌دهد.
  - Precision یک عبارت integer است که دقت مقدار DATETIME2 را مشخص می‌کند.

تابع DATETIME2FROMPARTS() یک مقدار از نوع DATETIME2 بازمی‌گرداند. اگر هرکدام از آرگومان‌های بالا NULL باشند، تابع مقدار NULL را بازمی‌گرداند.
اگر یک آرگومان دارای یک مقدار غیرمعتبر باشد، تابع DATETIME2FROMPARTS() خطا می‌دهد.
مثال‌های تابع DATETIME2FROMPARTS() در SQL Server

الف) مثال استفاده از تابع DATETIME2FROMPARTS() برای بازگرداندن یک DATETIME2

مثال زیر از تابع DATETIME2FROMPARTS() برای ساخت یک DATETIME2 متشکل از سال، ماه، روز، ساعت، دقیق، ثانیه، اعشار و دقت استفاده می‌کند:

```
SELECT
    DATETIME2FROMPARTS ( 2020, 12, 31, 11, 59, 59, 0, 0 ) result;
```

خروجی به‌این‌ترتیب است:

ب) مثال استفاده از تابع DATETIME2FROMPARTS() همراه با آرگومان‌های NULL

این مثال مقدار NULL را بازمی‌گرداند، چون آرگومان month برابر با NULL است:

```
SELECT
    DATETIME2FROMPARTS(2020, NULL, 31, 11, 59, 59, 0, 0) result;
```

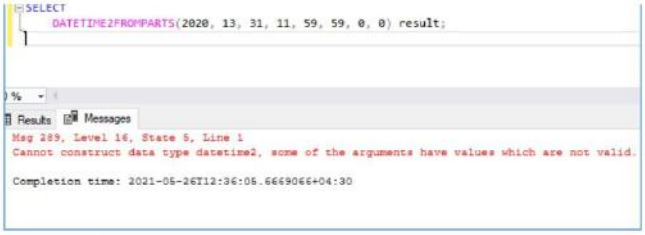
خروجی به این شکل است:

پ) مثال استفاده از تابع DATETIME2FROMPARTS() با مقادیر غیر معتبر در آرگومان‌ها

مثال زیر آرگومان‌های غیر معتبر را به تابع DATETIME2FROMPARTS() ارسال می‌کند که در نتیجه سیستم ارور می‌دهد:

```
SELECT
    DATETIME2FROMPARTS(2020, 13, 31, 11, 59, 59, 0, 0) result;
```

پیغام ارور به این صورت است:



به‌این‌ترتیب، چگونگی استفاده از تابع ()DATETIME2FROMPARTS برای ساخت یک مقدار DATETIME2 از اجزای تشکیل‌دهنده آن یعنی سال، ماه، روز، ساعت، دقیقه، ثانیه، اعشار و دقت را آموختید.

۳-۱۲-۱-۳ توابع رشته‌ای

در جدول زیر توابع رشته‌ای در SQL Server را مشاهده می‌کنید که روی یک‌رشته ورودی عمل می‌کنند و یک مقدار رشته یا عددی بازمی‌گردانند:

تابع	توضیح
ASCII	مقدار کد ASCII یک کاراکتر را بازمی‌گرداند.
CHAR	یک مقدار ASCII را به یک کاراکتر تبدیل می‌کند.
CHATINDEX	درون یک‌رشته که از یک موقعیت خاص شروع می‌شود به دنبال یک زیررشته می‌گردد و موقعیت زیررشته را بازمی‌گرداند.
CONCAT	دو یا تعداد بیشتری رشته را به یک‌رشته تبدیل می‌کند.
CONCAT_WS	چندین رشته را با یک علامت مجزا کننده، به یک‌رشته واحد تبدیل می‌کند.
DEFFERENCE	مقادیر SOUNDEX()از دو رشته را با هم مقایسه می‌کند.
FORMAT	یک مقدار فرمت شده با یک فرمت خاص و رفتار (اختیاری) را بازمی‌گرداند.
LEFT	تعداد مشخص شده‌ای کاراکتر را از یک‌رشته کاراکتر که از سمت چپ آغاز می‌شوند، استخراج می‌کند.
LEN	تعدادی کاراکتر را از یک‌رشته کاراکتر بازمی‌گرداند.
LOWER	کاراکترهای یک‌رشته را به حروف کوچک تبدیل می‌کند.
LTRIM	یک‌رشته جدید از یک‌رشته مشخص شده را پس از حذف جاهای خالی بازمی‌گرداند.
NCHAR	کاراکتر Unicode با کد integer مشخص شده را بازمی‌گرداند، همان‌طور که توسط استاندارد UNICODE تعریف شده است.
PATINDEX	موقعیت آغازین اولین وقوع الگو در یک‌رشته را بازمی‌گرداند.

QUOTENAME	یک‌رشته UNICODE همراه با فاصله دهنده را بازمی‌گرداند تا رشته ورودی را تبدیل به یک شناساگر دارای فاصله معتبر تبدیل کند.
REPLACE	تمام وقوع‌های یک زیررشته را درون یک‌رشته، با یک زیررشته دیگر جایگزین می‌کند.
REPLICATE	یک‌رشته که به تعداد دفعات خاصی تکرار شده است را بازمی‌گرداند.
REVERSE	ترتیب برعکس یک‌رشته کاراکتر را بازمی‌گرداند.
RIGHT	تعداد خاصی از کاراکترها را از یک‌رشته کاراکتر که از سمت راست شروع می‌شود، بازمی‌گرداند.
RTRIM	یک‌رشته جدید از یک‌رشته مشخص شده را پس از حذف تمام جاهای خالی، بازمی‌گرداند.
SOUNDEX	یک کد) یا (SOUNDEX چهار کاراکتری از یک‌رشته را بر اساس اینکه چگونه گفته شده است، بازمی‌گرداند.
SPACE	یک‌رشته، از فضاها یا فاصله‌های تکرارشونده را بازمی‌گرداند.
STR	داده‌های کاراکتری که از داده‌های عددی تبدیل شده است را بازمی‌گرداند.
STRING_AGG	ردیف‌های رشته‌های دارای یک مجزا کننده خاص را به یک‌رشته جدید متصل می‌کند.
STRING_ESCAPE	کاراکترهای خاصی از یک‌رشته را رها می‌کند و یک‌رشته جدید از کاراکترهای رها شده بازمی‌گرداند.
STRING_SPLIT	یک تابع با مقدار جدولی که یک‌رشته را بر اساس یک مجزا کننده مشخص شده، به‌ردیف‌های زیررشته مجزا می‌کند.
STUFF	بخشی از یک‌رشته را حذف می‌کند و سپس یک زیررشته دیگر را با شروع از یک موقعیت خاص، به رشته اضافه می‌کند.
SUBSTRING	یک زیررشته از درون یک‌رشته که از یک موقعیت خاص آغاز می‌شود و دارای طول خاصی است، استخراج می‌کند.
TRANSLATE	چندین تک کاراکتر را در یک عملیات جایگزین کرده و تک‌به‌تک ترجمه می‌کند.
TRIM	یک‌رشته جدید از یک‌رشته مشخص شده را پس از حذف تمام جاهای خالی ابتدا و انتها و بین حروف، بازمی‌گرداند.
UNICODE	مقدار integer از یک کاراکتر را بازمی‌گرداند) همان‌طور که توسط استاندارد Unicode تعریف شده است.(
UPPER	یک‌رشته را به حروف بزرگ تبدیل می‌کند.

همان‌طور که در جدول بالا مشاهده کردید تعداد توابع زیاد هست و سعی می‌کنیم در این قسمت بعضی از آنها را بررسی کنیم.

تابع ASCII در SQL Server

تابع ()ASCII یک عبارت کاراکتر می‌پذیرد و مقدار کد ASCII از اولین کاراکتر سمت چپ در یک عبارت کاراکتری را بازمی‌گرداند.

کد زیر، syntax تابع ()ASCII را نشان می‌دهد:

```
ASCII ( input_string )
```

Input_string می‌تواند یک کاراکتر حقیقی، یک عبارت رشته‌ای کاراکتری یا یک ستون باشد. اگر input_string دارای بیش از یک کاراکتر باشد، آنگاه تابع مقدار کد ASCII از اولین کاراکتر سمت چپ آن را بازمی‌گرداند.

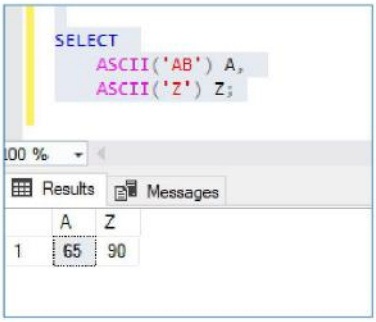
همان‌طور که احتمالاً می‌دانید، ASCII مخفف کد استاندارد آمریکایی برای تبادل اطلاعات (American Standard Code for Information Interchange) است. ASCII به‌عنوان یک استاندارد رمزگذاری کاراکتر برای رایانه‌های مدرن به کار می‌رود.

مثال‌های تابع ()ASCII در SQL Server

مثال زیر مقادیر کد ASCII از کاراکتر A تا Z را بازمی‌گرداند:

```
SELECT
    ASCII('AB') A,
    ASCII('Z') Z;
```

خروجی به این شکل است:



قطعه کد زیر از تابع ()ASCII برای تولید ۲۶ حرف الفبا استفاده می‌کند:

```
WITH cte AS(
    SELECT
        CHAR(ASCII('A')) [char],
        1 [count]
    UNION ALL
    SELECT
        CHAR(ASCII('A') + cte.count) [char],
        cte.count + 1 [count]
    FROM
        cte
)
SELECT
    TOP(26) cte.char
FROM
    cte;
```

خروجی به‌این‌ترتیب است:

	char
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	N
15	O
16	P

تابع CHAR در SQL Server

تابع ()CHAR یک مقدار کد ASCII را به یک مقدار کاراکتر تبدیل می‌کند. کد زیر، syntax تابع ()CHAR را نشان می‌دهد:

```
CHAR ( int_exp )
```

در این syntax، کد inte_expr یک عبارت integer است که مقدار آن یک integer است که بازه آن بین ۰ تا ۲۵۵ است.

اگر عبارت integer دارای مقداری باشد که خارج از این بازه است، آنگاه تابع ()CHAR مقدار NULL را بازمی‌گرداند.

تابع ()CHAR کاراکتری را بازمی‌گرداند که نوع داده آن CHAR(۱) است.

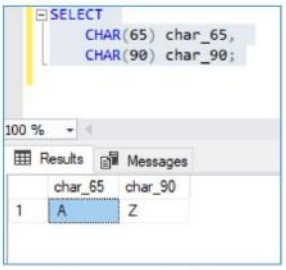
توجه کنید که برای تبدیل یک کاراکتر به یک مقدار ASCII، از تابع ()ASCII استفاده می‌کنیم.

مثال تابع CHAR() در SQL Server

مثال زیر از تابع CHAR() برای به‌دست‌آوردن کاراکترهای اعداد ٦٥ و ٩٠ استفاده می‌کند:

```
SELECT
    CHAR(65) char_65,
    CHAR(90) char_90;
```

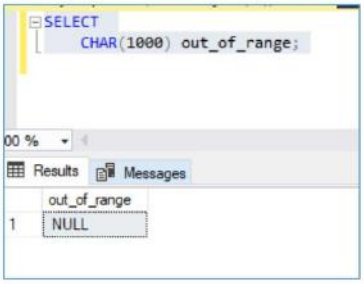
خروجی به این شکل است:



کد زیر مقدار NULL را بازمی‌گرداند، چون مقدار آرگومان خارج از بازه ۰ تا ۲۵۵ است.

```
SELECT
    CHAR(1000) out_of_range;
```

کد بالا، خروجی زیر را نشان می‌دهد:



تابع CHARINDEX در SQL Server

تابع CAHRINDEX() به جستجوی یک زیررشته درون یک‌رشته که از یک موقعیت خاص شروع می‌شود، می‌پردازد. این تابع موقعیت زیررشته یافته شده در رشته مورد جستجو را بازمی‌گرداند، یا اگر زیررشته اصلاً یافته نشود، مقدار صفر را بازمی‌گرداند.

موقعیت آغازین از ۱ شمرده می‌شود نه از ۰.

کد زیر، syntax تابع CAHRINDEX() را نشان می‌دهد:

CHARINDEX(substring, string [, start_location])

در این syntax:

- Substring زیر رشته‌ای است که به دنبال آن هستیم. طول آن به ۸۰۰۰ کاراکتر محدود است.
- String می‌تواند یک‌رشته، عبارت یا ستون باشد. این رشته‌ای است که جستجو می‌شود.
- Start_location موقعیتی است که جستجو از آنجا آغاز می‌شود. Start_location می‌تواند یک integer، یک big integer یا یک عبارت باشد که مقدار آن ممکن است هر نوع داده‌ای دیگری باشد.
- پارامتر start_location اختیاری است. اگر آن را مشخص نکنید یا مقدار آن صفر یا یک مقدار منفی باشد، آنگاه جستجو از ابتدای رشته آغاز می‌شود.

توجه کنید که تابع CAHRINDEX() بر اساس نوع قیاس، می‌تواند هم به‌صورت حساس به بزرگی یا کوچکی حروف و هم غیرحساس به این موضوع عمل کند.

مثال‌های تابع CAHRINDEX() در SQL Server

الف) استفاده از تابع CAHRINDEX() برای انجام یک جستجو

مثال زیر از تابع CAHRINDEX() برای انجام یک جستجوی ساده برای یافتن رشته 'SQL' درون رشته 'SQL Server CHARINDEX' استفاده می‌کند.

```
SELECT
    CHARINDEX('SQL', 'SQL Server CHARINDEX') position;
```

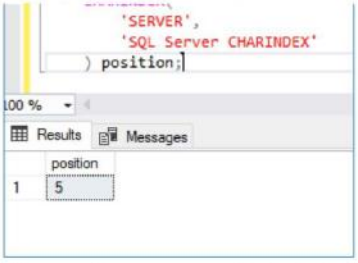
خروجی به این شکل است:

ب) استفاده از تابع CAHRINDEX() برای انجام یک جستجوی غیرحساس به بزرگی و کوچکی حروف

کد زیر یک جستجوی غیرحساس به بزرگی و کوچکی حروف برای یافتن رشته 'SERVER' درون رشته 'SQL Server CHARINDEX' را نشان می‌دهد:

```
SELECT
    CHARINDEX(
        'SERVER',
        'SQL Server CHARINDEX'
    ) position;
```

خروجی به‌این‌ترتیب است:

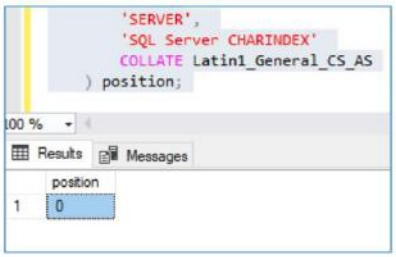


پ) استفاده از تابع CAHRINDEX() برای انجام یک جستجوی حساس به بزرگی و کوچکی حروف

مثال زیر، یک جستجوی حساس به بزرگی و کوچکی حروف برای یافتن رشته 'SERVER' در رشته 'SQL Server CHARINDEX' را نشان می‌دهد.

```
SELECT
    CHARINDEX(
        'SERVER',
        'SQL Server CHARINDEX'
        COLLATE Latin1_General_CS_AS
    ) position;
```

خروجی به این شکل است:

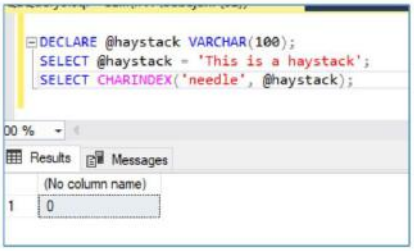


ت) استفاده از تابع CAHRINDEX() برای جستجوی یک زیررشته ناموجود

مثال زیر یک جستجو برای یافتن زیررشته 'needle' در رشته 'This is a haystack' را نشان می‌دهد:

```
DECLARE @haystack VARCHAR(100);
SELECT @haystack = 'This is a haystack';
SELECT CHARINDEX('needle', @haystack);
```

خروجی به این شکل است:

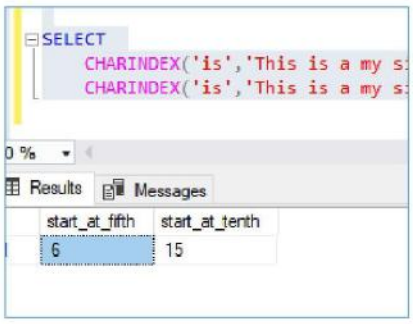


ث) استفاده از تابع CAHRINDEX() برای جستجو از یک موقعیت خاص

این مثال از پارامتر start_location برای آغاز جستجوی رشته 'is' از پنجمین و دهمین کاراکتر از رشته 'This is a my sister' استفاده می‌کند:

```
SELECT
    CHARINDEX('is','This is a my sister',5) start_at_fifth,
    CHARINDEX('is','This is a my sister',10) start_at_tenth;
```

خروجی به‌این‌ترتیب است:



تابع CONCAT در SQL Server

برای تبدیل دو یا تعداد بیشتری رشته به یک‌رشته، از تابع CONCAT() با syntax زیر استفاده می‌کنیم:

```
CONCAT ( input_string1, input_string2 [, input_stringN ] );
```

تابع CONCAT() دو رشته ورودی با حداکثر ۲۵۵ کاراکتر را می‌گیرد و آن‌ها را به یک‌رشته تبدیل می‌کند. این تابع حداقل به دو رشته ورودی نیاز دارد. اگر یکی از رشته‌های ورودی را وارد نکنید، تابع CONCAT() ارور می‌دهد.
اگر مقادیر رشته‌ای غیر کاراکتری وارد کنید، آنگاه تابع CONCAT() قبل از اینکه آن دو مقدار را با هم ادغام کند، به‌صورت ضمنی آن‌ها را به رشته تبدیل می‌کند.
تابع CONCAT() همچنین مقدار NULL را به یک‌رشته خالی با نوع VARCHAR(1) تبدیل می‌کند.
توجه کنید که برای اضافه‌کردن یک مجزا کننده در طول فرایند ادغام، باید از تابع CONCAT_WS() استفاده کنید (که در قسمت بعدی همین جلسه آن را توضیح خواهیم داد).
مثال‌های تابع CONCAT() در SQL Server

الف) استفاده از تابع CONCAT() برای چند رشته
مثال زیر از تابع CONCAT() برای ادغام سه رشته John فاصله یا space و Doe استفاده می‌کند:

```
SELECT
    'John' + ' ' + 'Doe' AS full_name;
```

خروجی به این شکل است:

ب) استفاده از تابع CONCAT() برای ستون‌های جدول

این مثال از جدول sales.customers از پایگاه‌داده نمونه Bikestores استفاده می‌کند:

جدول sales.customers

کد زیر از تابع CONCAT() برای ادغام مقادیر در ستون‌های first_name و last_name از جدول Sales.customers استفاده می‌کند:

```
SELECT
    customer_id,
    first_name,
    last_name,
    CONCAT(first_name, ' ', last_name) full_name
FROM
    sales.customers
ORDER BY
    full_name;
```

تصویر زیر، خروجی را نشان می‌دهد:

Results Messages

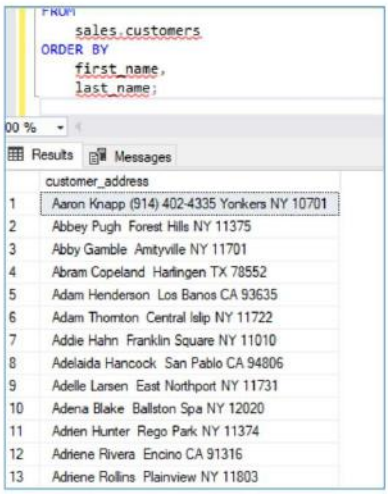
	customer_id	first_name	last_name	full_name
1	1174	Aaron	Knapp	Aaron Knapp
2	338	Abbey	Pugh	Abbey Pugh
3	75	Abby	Gamble	Abby Gamble
4	1224	Abram	Copeland	Abram Copeland
5	673	Adam	Henderson	Adam Henderson
6	1085	Adam	Thornton	Adam Thornton
7	195	Addie	Hahn	Addie Hahn
8	1261	Adelaida	Hancock	Adelaida Hancock
9	22	Adelle	Larsen	Adelle Larsen
10	1023	Adena	Blake	Adena Blake
11	1412	Adrien	Hunter	Adrien Hunter
12	574	Adriene	Rivera	Adriene Rivera
13	1252	Adriene	Rollins	Adriene Rollins
14	527	Afton	Juarez	Afton Juarez
15	1353	Agatha	Daniels	Agatha Daniels
16	769	Agatha	Melton	Agatha Melton
17	771	Agnes	Sims	Agnes Sims
18	1181	Agustina	Lawrence	Agustina Lawrence
19	1322	Ai	Forbes	Ai Forbes
20	937	Aida	Koch	Aida Koch

پ)استفاده از تابع CONCAT() برای مقدار NULL

مثال زیر از تابع CONCAT() برای فرمت کردن آدرس‌ها برای مشتریان استفاده می‌کند:

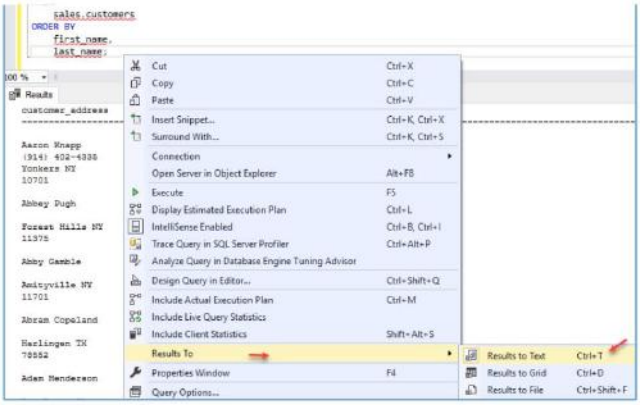
```
SELECT
    CONCAT(
        CHAR(13),
        CONCAT(first_name,' ',last_name),
        CHAR(13),
        phone,
        CHAR(13),
        CONCAT(city,' ',state),
        CHAR(13),
        zip_code
    ) customer_address
FROM
    sales.customers
ORDER BY
    first_name,
    last_name;
```

بخشی از خروجی در فرمت متن به این شکل است:



	customer_address
1	Aaron Knapp (914) 402-4335 Yonkers NY 10701
2	Abbey Pugh Forest Hills NY 11375
3	Abby Gamble Amityville NY 11701
4	Abram Copeland Harlingen TX 78552
5	Adam Henderson Los Banos CA 93635
6	Adam Thornton Central Islip NY 11722
7	Addie Hahn Franklin Square NY 11010
8	Adelaida Hancock San Pablo CA 94806
9	Adelle Larsen East Northport NY 11731
10	Adena Blake Ballston Spa NY 12020
11	Adrien Hunter Rego Park NY 11374
12	Adriene Rivera Encino CA 91316
13	Adriene Rollins Plainview NY 11803

برای اینکه دقیق‌تر این دستور را درک کنید به‌مانند شکل زیر بر روی صفحه کد کلیک راست کنید و از قسمت Results To گزینه‌ی Results to Test را انتخاب کنید و بعد از آن با کلیک بر روی F5 خروجی را به‌صورت Test مشاهده خواهید کرد، البته می‌توانستید از کلیدهای ترکیبی Ctrl + T و Ctrl+D استفاده کنید.



همان‌طور که به‌وضوح در خروجی نشان‌داده شده است، اگر مشتری شماره‌تلفن نداشته باشد، آنگاه تابع CONCAT() از یک فضای خالی برای ادغام استفاده می‌کند.

توجه کنید که از تابع CHAR() برای به‌دست‌آوردن کاراکتر خط جدید در این مثال استفاده کردیم.

تابع CONCAT_WS در SQL Server

تابع CONCAT_WS() دو یا تعداد بیشتری رشته را با استفاده از یک مجزا کننده، به یک‌رشته تبدیل می‌کند. CONCAT_WS() یعنی ادغام با مجزا کنند (concatenate with separator).

کد زیر، syntax تابع CONCAT_WS() را نشان می‌دهد:

```
CONCAT_WS(separator,input_string1,input_string2,[...input_stringN]);
```

در این syntax:

- Separator یک عبارت کاراکتر محور است که مقدار آن می‌تواند از یکی از نوع‌های VARCHAR، NCHAR، CHAR یا NVARCHAR باشد.
- Input_string1 تا input_stringN عباراتی با هر نوع معتبر هستند. تابع CONCAT_WS() قبل از انجام عمل ادغام، به‌صورت ضمنی مقادیری که دارای نوع غیر کاراکتری هستند را به نوع کاراکتری تبدیل می‌کند.

تابع CONCAT_WS() رشته‌های ورودی را به یک‌رشته تبدیل می‌کند. این تابع، رشته‌های در حال ادغام را با separator مشخص شده در اولین آرگومان با هم ادغام می‌کند.
توجه کنید که CONCAT_WS() حداقل به دو رشته ورودی نیاز دارد. یعنی اینکه اگر صفر یا فقط یک‌رشته ورودی وارد کنید، تابع ارور می‌دهد.

تابع CONCAT_WS() با مقدار NULL همانند یک‌رشته خالی از نوع VARCHAR(۱) برخورد می‌کند. این تابع همچنین در بین مقادیر NULL از مجزا کننده استفاده نمی‌کند.

در نتیجه، تابع CONCAT_WS() می‌تواند بدون اضافه‌کردن چیزی، رشته‌های دارای مقادیر خالی را به هم متصل کند.

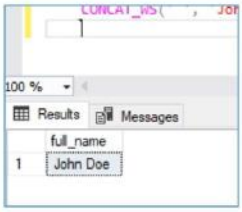
مثال‌های تابع CONCAT_WS() در SQL Server

الف) استفاده از تابع CONCAT_WS() برای اتصال چند رشته با یک مجزا کننده

مثال زیر از تابع CONCAT_WS() برای تبدیل دو رشته به یک‌رشته استفاده می‌کند. در این مثال، دو رشته ادغام شده با استفاده از یک فضای خالی یا Space از هم جدا شده‌اند:

```
SELECT
    CONCAT_WS(' ', 'John', 'Doe') full_name
```

خروجی به این شکل است:

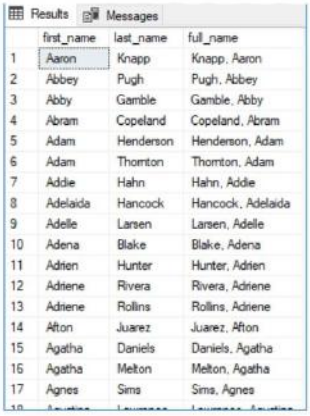


ب) استفاده از تابع CONCAT_WS() برای ستون‌های جدول

کد زیر از تابع CONCAT_WS() برای اتصال مقادیر ستون‌های last_name و first_name از جدول sales.customers استفاده می‌کند. نام خانوادگی و نام با استفاده از ویرگول و Space از هم جدا شده‌اند:

```
SELECT
    first_name,
    last_name,
    CONCAT_WS(', ', last_name, first_name) full_name
FROM
    sales.customers
ORDER BY
    first_name,
    last_name;
```

تصویر زیر، بخشی از خروجی را نشان می‌دهد:



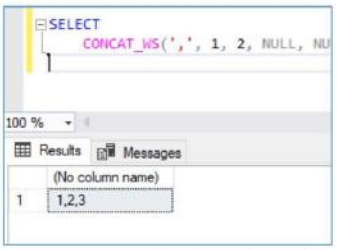
	first_name	last_name	full_name
1	Aaron	Knapp	Knapp, Aaron
2	Abbey	Pugh	Pugh, Abbey
3	Abby	Gamble	Gamble, Abby
4	Abram	Copeland	Copeland, Abram
5	Adam	Henderson	Henderson, Adam
6	Adam	Thornton	Thornton, Adam
7	Addie	Hahn	Hahn, Addie
8	Adelaida	Hancock	Hancock, Adelaida
9	Adelle	Larsen	Larsen, Adelle
10	Adena	Blake	Blake, Adena
11	Adrien	Hunter	Hunter, Adrien
12	Adriene	Rivera	Rivera, Adriene
13	Adriene	Rollins	Rollins, Adriene
14	Afton	Juarez	Juarez, Afton
15	Agatha	Daniels	Daniels, Agatha
16	Agatha	Melton	Melton, Agatha
17	Agnes	Sims	Sims, Agnes

پ) استفاده از تابع CONCAT_WS() برای مقدار NULL

کد زیر، چگونگی عملکرد تابع CONCAT_WS() برای رشته‌های ورودی که دارای مقادیر NULL هستند را نشان می‌دهد:

```
SELECT
    CONCAT_WS(',', 1, 2, NULL, NULL, 3);
```

خروجی به این شکل است:



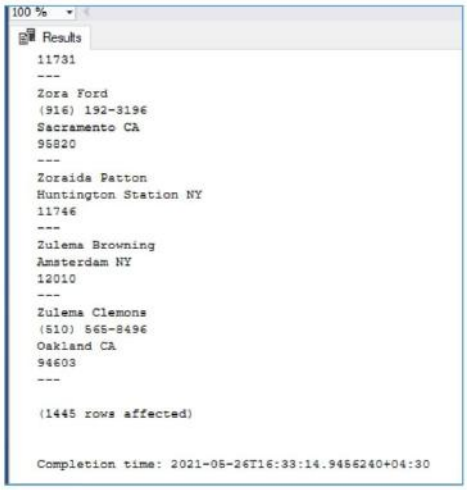
همان‌طور که به‌وضوح در خروجی مشاهده می‌کنید، تابع CONCAT_WS() مقدار NULL را نادیده می‌گیرد و هیچ مجزا کننده‌ای بین مقادیر NULL اضافه نمی‌کند.

مثال زیر داده‌های مشتری را ادغام می‌کند تا آدرس‌های مشتریان را فرمت کند. اگر یک مشتری دارای شماره‌تلفن نباشد، تابع آن را نادیده می‌گیرد:

```
SELECT
    CONCAT_WS
    (
        CHAR(13),
        CONCAT(first_name, ' ', last_name),
        phone,
        CONCAT(city, ' ', state),
        zip_code,
        '---'
    ) customer_address
FROM
```

```
    sales.customers
ORDER BY
    first_name,
    last_name;
```

این تصویر، بخشی از خروجی را نشان می‌دهد:

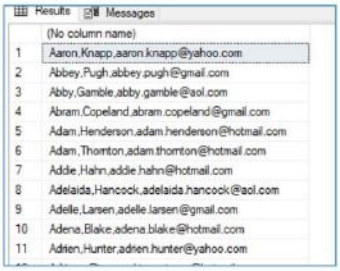


ت) استفاده از تابع CONCAT_WS() برای ایجاد فایل CSV

این کد از یک ویرگول و عنوان مجزا کننده استفاده می‌کند و مقادیر موجود در ستون‌های first_name ،last_name و email را با هم ادغام می‌کند تا یک فایل CSV ایجاد کند:

```
SELECT
    CONCAT_WS(',', first_name, last_name, email)
FROM
    sales.customers
ORDER BY
    first_name,
    last_name;
```

بخشی از خروجی به این شکل است:



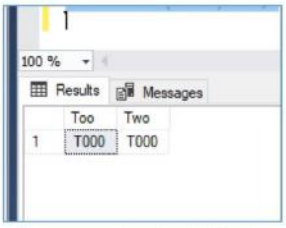
تابع DEFFERENCE در SQL Server

تابع SOUNDEX() بر اساس اینکه وقتی یک‌رشته به زبان می‌آید چگونه تلفظ می‌شود، رشته را به کد چهار کاراکتری تبدیل می‌کند.

برای مثال، هر دو واژه‌های Two و Too تلفظ یکسانی دارند، بنابراین باید دارای مقادیر SOUNDEX() مشابه باشند:

```
SELECT
    SOUNDEX('Too') Too,
    SOUNDEX('Two') Two;
```

خروجی به‌این‌ترتیب است:



تابع DEFFERENCE() یک مقدار integer بازمی‌گرداند که تفاوت بین مقادیر SOUNDEX() دو رشته را می‌سنجد.

کد زیر، syntax تابع DEFFERENCE() را نشان می‌دهد:

```
DIFFERENCE(input_string1, input_string2)
```

نتیجه DEFFERENCE()، تفاوت بین دو مقدار SOUNDEX() را در مقیاس ۰ تا ٤ نشان می‌دهد. مقدار ۰ یعنی ضعیف یا بدون وجود تشابه بین مقادیر SOUNDEX() و مقدار ٤ یعنی مقادیر SOUNDEX() بسیار مشابه یا همسان.

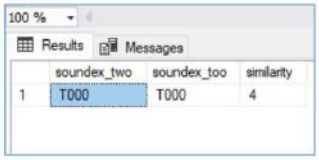
مثال‌های تابع DEFFERENCE() در SQL Server

الف) استفاده از تابع DEFFERENCE() با مقادیر SOUNDEX() مشابه

این مثال از تابع DEFFERENCE() برای مقایسه مقادیر SOUNDEX() از دو رشته Two و Too استفاده می‌کند:

```
SELECT
    SOUNDEX('Two') soundex_two,
    SOUNDEX('Too') soundex_too,
    DIFFERENCE('Two', 'Too') similarity;
```

خروجی به این شکل است:



کد زیر یک مقال دیگر است که تفاوت موجود در مقادیر SOUNDEX() از رشته‌های John و Johny را مقایسه می‌کند:

```
SELECT
    SOUNDEX('Johny') soundex_johny,
    SOUNDEX('John') soundex_john,
    DIFFERENCE('Johny', 'John') similarity;
```

خروجی به‌این‌ترتیب است:

	soundex_johny	soundex_john	similarity
1	J500	J500	4

ب) استفاده از تابع ()DEFFERENCE برای مقادیر ()SOUNDEX مختلف

مثال زیر، چگونگی عملکرد ()DEFFERENCE در هنگام مقایسه دو رشته که دارای تلفظ‌های ضعیف یا غیرمشابه هستند را نشان می‌دهد:

```
SELECT
    SOUNDEX('Coffee') soundex_coffee,
    SOUNDEX('Laptop') soundex_laptop,
    DIFFERENCE('Coffee', 'Laptop') similarity;
```

خروجی به این شکل است:

	soundex_coffee	soundex_laptop	similarity
1	C100	L131	1

تابع LEFT در SQL Server

تابع ()LEFT تعداد مشخصی کاراکتر را از سمت چپ یک‌رشته ارائه شده استخراج می‌کند. برای مثال، کد LEFT('SQL Server', 3) خروجی SQL را نمایش می‌دهد.

Syntax تابع ()LEFT به این صورت است:

```
LEFT ( input_string , number_of_characters )
```

در این syntax:

- Input_string می‌تواند یک‌رشته، متغیر یا ستون باشد. نوع داده نتیجه input_string می‌تواند هر نوع داده‌ای به جز نوع‌های TEXT یا NTEXT باشد، این دو نوع به‌صورت ضمنی به VARCHAR و NVARCHAR تبدیل می‌شوند.
- Number_of_characters یک integer مثبت است که تعداد کاراکترهای input_string که بازگشت داده می‌شوند را مشخص می‌کند.

وقتی input_string یک نوع داده‌ای کاراکتر غیر Unicode باشد، آنگاه تابع ()LEFT یک مقدار VARCHAR و اگر input_string یک نوع داده کاراکتر Unicode باشد، یک مقدار NVARCHAR بازمی‌گرداند.

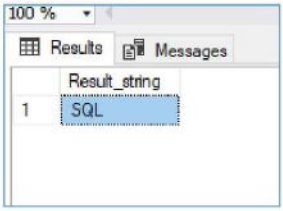
مثال‌های تابع LEFT() در SQL Server

الف) استفاده از تابع LEFT() برای یک‌رشته کاراکتر

کد زیر از تابع LEFT() برای بازگرداندن سه کاراکتر اول از سمت چپ رشته کاراکتر SQL Server را نشان می‌دهد:

```
SELECT LEFT('SQL Server',3) Result_string;
```

خروجی به این شکل است:



ب) استفاده از تابع LEFT() برای یک ستون جدول

مثال زیر ۷ کاراکتر اول سمت چپ هر نام محصول در جدول production.products را بازمی‌گرداند:

```
SELECT
    product_name,
    LEFT(product_name, 7) first_7_characters
FROM
    production.products
ORDER BY
    product_name;
```

تصویر زیر بخشی از خروجی را نشان می‌دهد:

Results Messages

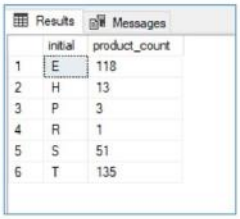
	product_name	first_7_characters
1	Electra Amsterdam Fashion 3i Ladies' - 2017/2018	Electra
2	Electra Amsterdam Fashion 7i Ladies' - 2017	Electra
3	Electra Amsterdam Original 3i - 2015/2017	Electra
4	Electra Amsterdam Original 3i Ladies' - 2017	Electra
5	Electra Amsterdam Royal 8i - 2017/2018	Electra
6	Electra Amsterdam Royal 8i Ladies - 2018	Electra
7	Electra Cruiser 1 (24-Inch) - 2016	Electra
8	Electra Cruiser 1 (24-Inch) - 2016	Electra
9	Electra Cruiser 1 - 2016/2017/2018	Electra
10	Electra Cruiser 1 Ladies' - 2018	Electra
11	Electra Cruiser 1 Tall - 2016/2018	Electra
12	Electra Cruiser 7D (24-Inch) Ladies' - 2016/2018	Electra
13	Electra Cruiser 7D (24-Inch) Ladies' - 2016/2018	Electra
14	Electra Cruiser 7D - 2016/2017/2018	Electra
15	Electra Cruiser 7D Ladies' - 2016/2018	Electra
16	Electra Cruiser 7D Tall - 2016/2018	Electra
17	Electra Cruiser Lux 1 - 2016/2018	Electra

پ) استفاده از تابع LEFT() برای دستور GROUP BY

مثال زیر از تابع LEFT() برای بازگرداندن مجموعه‌ای از حروف اول نام محصول و تعداد هر محصول برای هر حرف اول، استفاده می‌کند:

```
SELECT
 LEFT(product_name, 1) initial,
 COUNT(product_name) product_count
FROM
 production.products
GROUP BY
 left(product_name, 1)
ORDER BY
 initial;
```

خروجی به این شکل است:



	initial	product_count
1	E	118
2	H	13
3	P	3
4	R	1
5	S	51
6	T	135

تابع LEN در SQL Server

تابع LEN() تعداد کاراکترهای یک‌رشته ورودی (به جز فضاهای خالی بین حروف) را بازمی‌گرداند.

کد زیر، syntax تابع LEN() را نشان می‌دهد:

```
LEN(input_string)
```

در این syntax، کد input_string می‌تواند یک‌رشته کاراکتر، عبارت رشته‌ای یا یک ستون از داده‌های رشته‌ای یا دودویی باشد.

تابع LEN() مقداری را بازمی‌گرداند که اگر input_string از نوع داده (NVARCHART(max)، VARCHAR(max) یا VARBINARY(max) باشد، نوع داده آن BIGINT است و در غیر این صورت، نوع آن INT است.

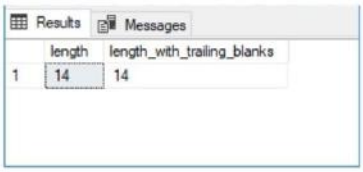
مثال‌های تابع LEN() در SQL Server

الف) استفاده از تابع LEN() برای یک‌رشته

مثال زیر از تابع LEN() برای بازگرداندن تعداد کاراکترهای رشته SQL Server LEN و همین رشته با فضاهای خالی بین حروف استفاده می‌کند.

```
SELECT
    LEN('SQL Server LEN') length,
    LEN('SQL Server LEN   ') length_with_trailing_blanks;
```

خروجی به این شکل است:



	length	length_with_trailing_blanks
1	14	14

همان‌طور که مشاهده می‌کنید، تابع LEN() از فضاهای خالی بین حروف صرف‌نظر می‌کند.

ب) استفاده از تابع LEN() برای یک ستون

برای این مثال از جدول production.products از پایگاه‌داده نمونه استفاده می‌کنیم.

کد زیر از تابع LEN() برای بازگرداندن نام محصول و طول آن استفاده کرده و محصولات را بر اساس طول نام محصول مرتب می‌کند.

```
SELECT
    product_name,
    LEN(product_name) product_name_length
FROM
    production.products
ORDER BY
    LEN(product_name) DESC;
```

تصویر زیر، بخشی از خروجی را نشان می‌دهد:

	product_name	product_name_length
1	Electra Townie Balloon 8D EQ Ladies' - 2016/2017/...	53
2	Electra Townie Balloon 8D EQ Ladies' - 2016/2017/...	53
3	Electra Townie Original 7D EQ Ladies' - 2017/2018	49
4	Pure Cycles Western 3-Speed - Women's - 2015/2016	49
5	Electra Sugar Skulls 1 (20-inch) - Girl's - 2017	48
6	Electra Townie Balloon 7i EQ Ladies' - 2017/2018	48
7	Electra Amsterdam Fashion 3i Ladies' - 2017/2018	48
8	Electra Cruiser 7D (24-Inch) Ladies' - 2016/2018	48
9	Electra Cruiser 7D (24-Inch) Ladies' - 2016/2018	48
10	Electra Townie Balloon 7i EQ Ladies' - 2017/2018	48
11	Electra Tiger Shark 3i (20-inch) - Boys' - 2018	47
12	Electra Sweet Ride 3i (20-inch) - Girls' - 2018	47
13	Electra Tiger Shark 1 (20-inch) - Boys' - 2018	46
14	Electra Sweet Ride 1 (20-inch) - Girl's - 2018	46
15	Electra Soft Serve 1 (16-inch) - Girl's - 2018	46
16	Electra Cyclosaurus 1 (16-inch) - Boy's - 2018	46
17	Sun Bicycles Boardwalk (24-inch Wheels) - 2017	46

تابع LOWER در SQL Server

تابع LOWER() یک‌رشته را به حروف کوچک تبدیل می‌کند. کد زیر، syntax تابع LOWER() را نشان می‌دهد:

```
LOWER(input_string)
```

در این syntax، کد input_string می‌تواند یک‌رشته کاراکتر، متغیر، عبارت رشته کاراکتر یا ستون جدول باشد.

نوع input_string باید به‌صورت ضمنی قابل تبدیل‌شدن به VARCHAR باشد. در غیر این صورت، باید از تابع CAST() برای تبدیل غیر ضمنی input_string استفاده کنید.

تابع LOWER() حروف کوچک input_string را بازمی‌گرداند.

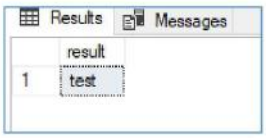
مثال‌های تابع LOWER() در SQL Server

الف) استفاده از تابع LOWER() برای رشته‌ها

این مثال از تابع LOWER() برای تبدیل رشته 'TEST' به 'test' استفاده می‌کند:

```
SELECT
    LOWER('TEST') result;
```

خروجی به این شکل است:



ب) استفاده از تابع LOWER() برای ستون جدول

از جدول customers از پایگاه‌داده Bikestores در این مثال استفاده می‌کنیم:

کد زیر قبل از انجام عمل ادغام، از تابع LOWER() برای تبدیل نام و نام خانوادگی مشتریان به حروف کوچک استفاده می‌کند:

```
SELECT
    first_name,
    last_name,
    CONCAT_WS(
        ' ',
        LOWER(first_name),
        LOWER(last_name)
    ) full_name_lowercase
FROM
    sales.customers
ORDER BY
    first_name,
    last_name;
```

تصویر زیر بخشی از خروجی را نشان می‌دهد:

	first_name	last_name	full_name_lowercase
1	Aaron	Knapp	aaron knapp
2	Abbey	Pugh	abbey pugh
3	Abby	Gamble	abby gamble
4	Abram	Copeland	abram copeland
5	Adam	Henderson	adam henderson
6	Adam	Thornton	adam thornton
7	Addie	Hahn	addie hahn
8	Adelaida	Hancock	adelaida hancock
9	Adelle	Larsen	adelle larsen
10	Adena	Blake	adena blake
11	Adrien	Hunter	adrien hunter
12	Adriene	Rivera	adriene rivera
13	Adriene	Rollins	adriene rollins

به این صورت، چگونگی استفاده از تابع LOWER() برای تبدیل یک‌رشته به حروف کوچک را نیز آموختید.

تابع LTRIM در SQL Server

تابع LTRIM() پس از حذف فضاهای خالی ابتدا و انتهای رشته، یک‌رشته بازمی‌گرداند. کد زیر، syntax تابع LTRIM() را نشان می‌دهد:

```
LTRIM(input_string)
```

در این syntax، رشته ورودی یک عبارت کاراکتری یا داده‌های باینری است. این عبارت می‌تواند رشته، متغیر یا ستون باشد.

مقدار Input_string باید یک مقدار با نوع داده‌ای (به جز نوع‌های داده‌ای NTEXT، TEXT و IMAGE) باشد که بتوانیم آن را به‌صورت ضمنی به VARCHAR تبدیل کنیم.

در غیر این صورت، باید از تابع CAST() برای تبدیل غیر ضمنی آن به یک‌رشته کاراکتر استفاده کنیم.

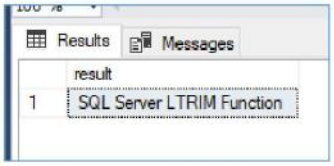
مثال‌های تابع LTRIM() در SQL Server

الف) استفاده از تابع LTRIM() برای رشته‌ها

این مثال از تابع LTRIM() برای حذف فضاهای خالی ابتدا و انتهای رشته 'SQl Server LTRIM function' استفاده می‌کند:

```
SELECT
    LTRIM('   SQL Server LTRIM Function') result;
```

خروجی به‌این‌ترتیب است:



	result
1	SQL Server LTRIM Function

ب) استفاده از تابع LTRIM() برای حذف فضاهای خالی

مثال زیر پس از تبدیل رشته‌های به چند زیررشته، از تابع LTRIM() برای حذف فضاهای خالی ابتدا و انتهای رشته استفاده می‌کند:

```
SELECT
    LTRIM(value) part
FROM
    STRING_SPLIT('Doe, John', ',');
```

خروجی به این شکل است:

توجه کنید که اگر از تابع LTRIM() استفاده نکنید، دومین ردیف یک فضای خالی ابتدا یا انتهای رشته خواهد داشت.

٤-١٢-١-٣ توابع سیستمی

توابع سیستمی عبارت‌اند از:

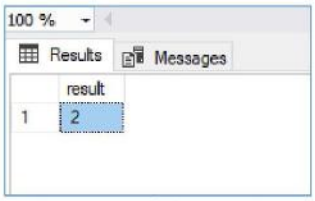
- CAST – یک مقدار با یک نوع را به یک نوع دیگر cast (یعنی تبدیل صریح)می‌کند.
- CONVERT – یک مقدار با یک نوع را به یک نوع دیگر convert (یعنی تبدیل ضمنی) می‌کند.
- CHOOSE – یکی از دو مقدار را بر اساس نتیجه اولین آرگومان بازمی‌گرداند.
- ISNULL – مقدار NULL را با یک مقدار مشخص شده جایگزین می‌کند.
- ISNUMERIC – بررسی می‌کند که آیا یک عبارت از نوع عددی معتبر هست یا نه.
- IIF – منطق if-else را به یک query اضافه می‌کند.
- TRY-CAST – یک مقدار با یک نوع را به یک نوع دیگر cast می‌کند و اگر cast موفقیت‌آمیز نبود، مقدار NULL بازمی‌گرداند.
- TRY-CONVERT – یک مقدار با یک نوع را به یک نوع دیگر convert می‌کند و مقداری را بازمی‌گرداند که به یک نوع خاص ترجمه شده است. اگر cast یا تبدیل موفقیت‌آمیز نباشد، مقدار NULL را بازمی‌گرداند.
- TRY-PARSE – یک‌رشته را به یک date/time یا یک عدد تبدیل می‌کند و اگر تبدیل موفقیت‌آمیز نباشد، مقدار NULL بازمی‌گرداند.
- تبدیل Datetime به string – چگونگی تبدیل یک مقدار datetime به یک‌رشته با یک فرمت خاص را نشان می‌دهد.
- تبدیل رشته به datetime – چگونگی تبدیل یک‌رشته به یک مقدار datetime را نشان می‌دهد.
- تبدیل datetime به date – یک Datetime را به date تبدیل می‌کند.

تابع CAST در SQL Server

به query زیر دقت کنید:

```
SELECT 1 + '1' AS result;
```

مقدار ۲ را بازمی‌گرداند:

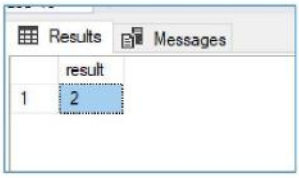


در این کد، SQL Server به‌صورت ضمنی کاراکتر رشته '۱' را به عدد ۱ تبدیل می‌کند.

وقتی از دو مقدار با نوع‌های داده‌ای مختلف استفاده می‌کنیم، SQL Server سعی می‌کند قبل از اینکه محاسبه را پردازش کند، نوع داده‌ای کمتر را به نوع داده‌ای بیشتر و بالاتر تبدیل کند. به این نوع تبدیل در SQL Server، تبدیل ضمنی گفته می‌شود.

در مقابل تبدیل‌های ضمنی، تبدیل‌های صریح یا غیر ضمنی را داریم که در آن‌ها تابع ()CAST را فراخوانی می‌کنیم تا به‌صورت صریح یک مقدار با یک نوع را به یک نوع دیگر تبدیل کند:

```
SELECT 1 + CAST(1 AS INT) result;
```



Syntax تابع ()CAST به این شکل است:

```
CAST ( expression AS target_type [ ( length ) ] )
```

در این syntax:

- Expression می‌تواند یک‌رشته یا یک عبارت معتبر با هر نوعی (که قابل تبدیل‌شدن است) باشد.
- Target_type نوع داده مورد نظر است که می‌خواهید عبارت به آن تبدیل شود. این نوع می‌تواند SQL-VARIANT، BIT، INT و غیره باشد. توجه کنید که این کد نمی‌تواند یک نوع داده‌ای alias باشد.
- Length یک integer اختیاری است که طول نوع داده‌ای مورد نظر را مشخص می‌کند. مقدار پیش‌فرض length برابر با ۳۰ است.

تابع ()CAST عبارت تبدیل شده به نوع داده‌ای مورد نظر را بازمی‌گرداند.

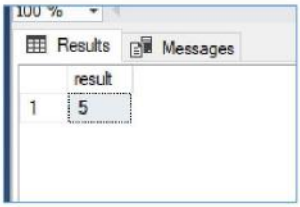
مثال‌های تابع ()CAST در SQL Server

الف) استفاده از تابع CAST() برای تبدیل یک عدد اعشاری به یک عدد صحیح

این مثال از تابع CAST() برای تبدیل عدد اعشاری ۵.۹۵ به یک عدد صحیح استفاده می‌کند:

```
SELECT CAST(5.95 AS INT) result;
```

خروجی به‌این‌ترتیب است:

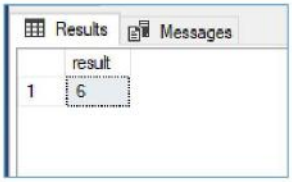


ب) استفاده از تابع CAST() برای تبدیل یک عدد اعشاری به یک عدد اعشاری دیگر با طول متفاوت

مثال زیر از تابع CAST() برای تبدیل عدد اعشاری ۵.۹۵ به یک عدد اعشاری دیگر با صفر عدد اعشار تبدیل می‌کند:

```
SELECT CAST(5.95 AS DEC(3,0)) result;
```

خروجی به این شکل است:



وقتی یک مقدار با نوع‌های داده‌ای را به حالت‌های مختلف تبدیل می‌کنید، SQL Server بر اساس قوانین زیر، یک نتیجه کوتاه شده یا یک مقدار گرد شده بازمی‌گرداند:

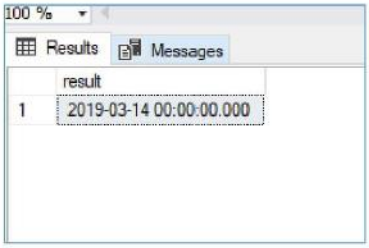
از نوع داده	به نوع داده	رفتار
numeric	numeric	گرد کردن
numeric	int	کوتاه شدن
Numeric	Money	گرد شدن
Money	Int	گرد شدن
Money	Numeric	گرد شدن
Float	Int	کوتاه شدن
Float	Numeric	گرد شدن
Float	Datetime	گرد شدن
Datetime	Int	گرد شدن

پ) استفاده از تابع CAST() برای تبدیل یک‌رشته به یک مقدار datetime

این مثال از تابع CAST() برای تبدیل رشته '۲۰۱۹-۰۳-۱٤' به datetime استفاده می‌کند:

```
SELECT
    CAST('2019-03-14' AS DATETIME) result;
```

خروجی به‌این‌ترتیب است:



ت) استفاده از تابع CAST() با عملگرهای ریاضی

برای این مثال از جداول sales.orders و sales.order_items از پایگاه‌داده Bikestores استفاده می‌کنیم:

کد زیر از تابع CAST() برای تبدیل فروش‌های ماهیانه در سال ۲۰۱۷ به مقادیر integer استفاده می‌کند.

```
SELECT
    MONTH(order_date) month,
    CAST(SUM(quantity * list_price * (1 - discount)) AS INT) amount
FROM sales.orders o
    INNER JOIN sales.order_items i ON o.order_id = i.order_id
WHERE
    YEAR(order_date) = 2017
GROUP BY
    MONTH(order_date)
ORDER BY
    month;
```

تصویر زیر، خروجی را نشان می‌دهد:

Results | Messages

	month	amount
1	1	285616
2	2	312923
3	3	308911
4	4	227290
5	5	268233
6	6	378865
7	7	229995
8	8	290553
9	9	293405
10	10	310328
11	11	281577
12	12	259505

به این صورت چگونگی استفاده از تابع CAST() برای تبدیل یک مقدار با یک نوع به یک نوع دیگر را آموختید.

تابع CONVERT در SQL Server

تابع CONVERT() به شما امکان می‌دهد تا یک مقدار با یک نوع را به یک نوع دیگر تبدیل کنید.
کد زیر، syntax تابع CONVERT() را نشان می‌دهد:

```
CONVERT ( target_type [ ( length ) ] , expression [ , style ] )
```

در این syntax:

- Target_type نوع داده‌ای مورد نظر است که می‌خواهید عبارت به آن تبدیل شود. این نوع می‌تواند SQL_VARIANT، BIT، INT و غیره باشد. توجه کنید که این کد نمی‌تواند یک نوع داده‌ای Alias باشد.
- Length یک integer است که طول نوع داده‌ای مورد نظر را مشخص می‌کند. length اختیاری است و مقدار پیش‌فرض آن ۳۰ است.
- Expression یک عبارت معتبر با هر نوع داده‌ای قابل تبدیل‌شدن است.
- Style یک integer اختیاری است که مشخص می‌کند تابع CONVERT() چگونه عبارت را ترجمه می‌کند. اگر مقدار style برابر با NULL باشد، تابع CONVERT() مقدار NULL را بازمی‌گرداند.

تابع CONVERT() مقدار ترجمه شده expression به target_type را با یک style مشخص شده بازمی‌گرداند.

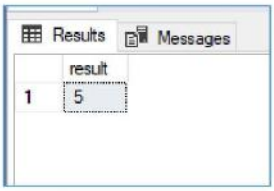
تابع CONVERT() مشابه تابع CAST() است. به‌هرحال، برای SQL Server خاص و متفاوت است. برخلاف تابع CONVERT()، تابع CAST() بخشی از توابع ANSI-SQL است که به طور وسیع در بسیاری از محصولات پایگاه داده‌ای دیگر در دسترس است.
مثال‌های تابع CONVERT() در SQL Server

الف) استفاده از تابع CONVERT() برای تبدیل یک عدد اعشاری به یک integer

این مثال از تابع CONVERT() برای تبدیل عدد اعشاری ۹.۹۵ به یک integer استفاده می‌کند:

```
SELECT CONVERT(INT, 5.95) result;
```

خروجی به این شکل است:

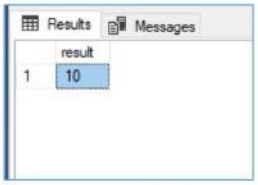


ب) استفاده از تابع CONVERT() برای تبدیل یک عدد اعشاری به یک عدد اعشاری دیگر با طول متفاوت

این مثال از تابع CONVERT() برای تبدیل عدد اعشاری ۹.۹۵ به یک عدد اعشاری دیگر با صفر عدد اعشار استفاده می‌کند:

```
SELECT CAST(9.95 AS DEC(2,0)) result;
```

خروجی به این صورت است:



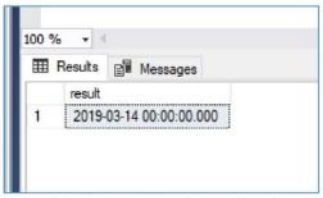
توجه کنید که رفتارهای گرد کردن و کوتاه کردن تابع CONVERT() دقیقاً مشابه رفتارهای تابع CAST() است.

پ) استفاده از تابع CONVERT() برای تبدیل یک‌رشته به یک مقدار datetime

این مثال از تابع CONVERT() برای تبدیل رشته '۲۰۱۹-۰۳-۱۴' به یک مقدار Datetime استفاده می‌کند:

```
SELECT
    CONVERT(DATETIME, '2019-03-14') result;
```

خروجی به این صورت است:

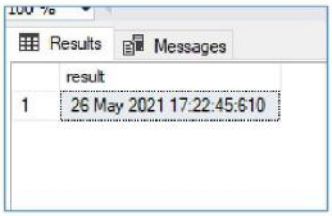


ت) استفاده از تابع CONVERT() برای تبدیل یک مقدار datetime به یک‌رشته

این مثال از تابع CONVERT() برای تبدیل تاریخ و زمان فعلی به یک‌رشته با style مشخص استفاده می‌کند:

```
SELECT
    CONVERT(VARCHAR, GETDATE(),13) result;
```

خروجی به‌این‌ترتیب است:



در نتیجه، چگونگی استفاده از تابع CONVERT() برای تبدیل یک مقدار با یک نوع داده‌ای به یک نوع داده دیگر را نیز آموختید.

تابع CHOOSE در SQL Server

تابع CHOOSE() یک آیتم را از لیست آیتم‌های موجود در یک index مشخص شده بازمی‌گرداند.

کد زیر، syntax تابع CHOOSE() را نشان می‌دهد:

```
CHOOSE ( index, elem_1, elem_2 [, elem_n ] )
```

در این syntax:

- Index یک عبارت integer است که index عنصری که باید بازگردانده شود را مشخص می‌کند. توجه کنید که indexهای عناصر بر اساس ۱ هستند.
- Elem_1, elem_2, … elem_n لیستی از مقادیر مجزا شده با ویرگول هستند که هر نوعی می‌توانند داشته باشند.

اگر index برابر با ۱ باشد، تابع CHOOSE() مقدار elem_1 را بازمی‌گرداند. اگر index برابر با ۲ باشد، تابع CHOOSE() مقدار elem_2 را بازمی‌گرداند و الی آخر.

اگر index یک مقدار integer نباشد، به یک integer تبدیل یا cast می‌شود. درصورتی‌که index خارج از بازه لیست باشد، تابع CHOOSE() مقدار NULL را بازمی‌گرداند.

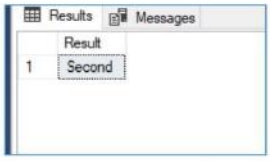
مثال‌های تابع CHOOSE() در SQL Server

الف) استفاده از تابع CHOOSE() برای مقادیر رشته

این مثال، دومین آیتم از لیست مقادیر را بازمی‌گرداند:

```
SELECT
    CHOOSE(2, 'First', 'Second', 'Third') Result;
```

خروجی به این شکل است:



ب) استفاده از تابع CHOOSE() برای ستون جدول

مثال زیر از تابع CHOOSE() برای بازگرداندن وضعیت سفارش بر اساس مقدار موجود در ستون order_status از جدول sales.orders استفاده می‌کند:

```
SELECT
    order_id,
```

```
    order_date,
    CHOOSE(order_status,
        'Pending',
        'Processing',
        'Rejected',
        'Completed') AS order_status
FROM
    sales.orders
ORDER BY
    order_date DESC;
```

تصویر زیر بخشی از خروجی را نشان می‌دهد:

Results | Messages

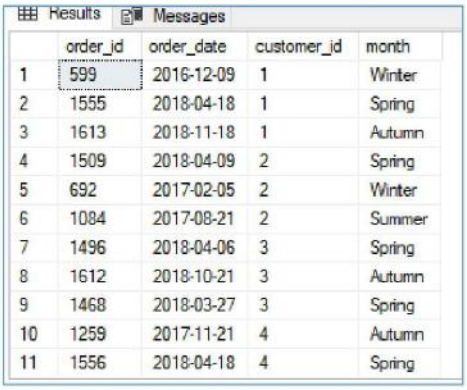
	order_id	order_date	order_status
1	1615	2018-12-28	Rejected
2	1614	2018-11-28	Rejected
3	1613	2018-11-18	Rejected
4	1612	2018-10-21	Rejected
5	1611	2018-09-06	Rejected
6	1610	2018-08-25	Rejected
7	1609	2018-08-23	Rejected
8	1608	2018-07-12	Rejected
9	1607	2018-07-11	Rejected
10	1606	2018-07-10	Rejected
11	1605	2018-07-01	Rejected

پ) استفاده از تابع CHOOSE() همراه با تابع MONTH

مثال زیر از تابع MONTH() برای بازگرداندن فصل‌هایی که در آن‌ها مشتریان محصول خریداری کرده‌اند، استفاده می‌کند. نتیجه تابع MONTH() در تابع CHOOSE() استفاده می‌شود تا فصل متناظر بازگردانده شود:

```
SELECT
    order_id,
    order_date,
    customer_id,
    CHOOSE(
        MONTH(order_date),
        'Winter',
        'Winter',
        'Spring',
        'Spring',
        'Spring',
        'Summer',
        'Summer',
        'Summer',
        'Autumn',
        'Autumn',
        'Autumn',
        'Winter') month
FROM
    sales.orders
ORDER BY
    customer_id;
```

تصویر زیر بخشی از خروجی را نشان می‌دهد:



Results | Messages

	order_id	order_date	customer_id	month
1	599	2016-12-09	1	Winter
2	1555	2018-04-18	1	Spring
3	1613	2018-11-18	1	Autumn
4	1509	2018-04-09	2	Spring
5	692	2017-02-05	2	Winter
6	1084	2017-08-21	2	Summer
7	1496	2018-04-06	3	Spring
8	1612	2018-10-21	3	Autumn
9	1468	2018-03-27	3	Spring
10	1259	2017-11-21	4	Autumn
11	1556	2018-04-18	4	Spring

تابع ISNULL در SQL Server

تابع ()ISNULL، مقدار NULL را با یک مقدار مشخص شده جایگزین می‌کند. کد زیر، Syntax تابع ()ISNULL را نشان می‌دهد:

```
ISNULL(expression, replacement)
```

تابع ()ISNULL دو آرگومان می‌پذیرد:

- Expression یک عبارت با هر نوعی است که بررسی می‌شود که NULL هست یا خیر.
- Replacement مقداری است که اگر عبارت NULL باشد بازگردانده می‌شود. Replacement باید قابل تبدیل‌شدن به یک مقدار از نوع مقدار موجود در expression باشد.

تابع ()ISNULL درصورتی‌که مقدار expression برابر با NULL باشد، مقدار replacement را بازمی‌گرداند. اگر نوع‌های دو آرگومان متفاوت باشند، این تابع قبل از بازگرداندن یک مقدار، به‌صورت ضمنی نوع replacement را به نوع expression تبدیل می‌کند.

درصورتی‌که مقدار expression برابر با NULL نباشد، تابع ()ISNULL مقدار expression را بازمی‌گرداند.

مثال‌های تابع ()ISNULL در SQL Server

الف) استفاده از تابع ISNULL() برای داده‌های عددی

مثال زیر از تابع ISNULL() برای بازگرداندن دومین آرگومان استفاده می‌کند، چون اولین آرگومان برابر با NULL است:

```
SELECT
    ISNULL(NULL,20) result;
```

خروجی به این شکل است:

ب) استفاده از تابع ISNULL() برای رشته کاراکتر

مثال زیر از تابع ISNULL() برای بازگرداندن رشته 'Hello' استفاده می‌کند، چون اولین آرگومان است و NULL نیست:

```
SELECT
    ISNULL('Hello', 'Hi') Result;
```

خروجی به‌این‌ترتیب است:

پ) استفاده از تابع ISNULL() برای جایگزینی مقادیر NULL با مقادیر معنادار

ابتدا، یک جدول جدید به نام divisions ایجاد می‌کنیم که دسته‌بندی‌های ورزشکار را بر اساس سن ذخیره می‌کند:

```
CREATE TABLE divisions
(
    id      INT
    PRIMARY KEY IDENTITY,
    min_age INT DEFAULT 0,
    max_age INT
);
```

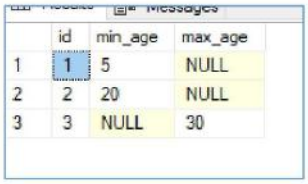
دوم، چند ردیف به جدول divisions اضافه می‌کنیم:

```
INSERT INTO divisions(min_age, max_age)
VALUES(5,null),
        (20,null),
        (null,30);
```

سوم، داده‌ها را از جدول divisions به دست می‌آوریم:

```
SELECT
    id,
    min_age,
    max_age
FROM
    divisions;
```

تصویر زیر، خروجی را نشان می‌دهد:



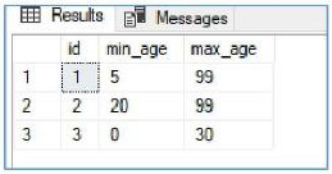
	id	min_age	max_age
1	1	5	NULL
2	2	20	NULL
3	3	NULL	30

اگر یک دسته‌بندی دارای محدودیت حداقل سن نباشد، پس ستون min_age دارای NULL خواهد بود. به طور مشابه، اگر یک دسته‌بندی نیز به حداکثر سن نداشته باشد، ستون max_age نیز دارای NULL خواهد بود.

در آخر، از تابع ISNULL() برای تبدیل NULL در ستون min_age به ۰ و تبدیل NULL در ستون max_age به ۹۹ استفاده می‌کنیم:

```
SELECT
    id,
    ISNULL(min_age,0) min_age,
    ISNULL(max_age,99) max_age
FROM
    divisions;
```

تصویر زیر، خروجی را نشان می‌دهد:



Results Messages

	id	min_age	max_age
1	1	5	99
2	2	20	99
3	3	0	30

تابع ISNUMERIC در SQL Server

تابع ISNUMERIC() یک عبارت می‌پذیرد و اگر عبارت از نوع عددی معتبری باشد، مقدار ۱ را بازمی‌گرداند، در غیر این صورت، مقدار ۰ را بازمی‌گرداند.

کد زیر، syntax تابع ISNUMERIC() را نشان می‌دهد:

```
ISNUMERIC ( expression )
```

در این syntax، کد expression می‌تواند هر عبارت معتبری باشد.

توجه داشته باشید که یک نوع عددی معتبر، یکی از نوع‌های زیر است:

اعداد صحیح: TINYINT، SMALLINT، INT، BIGINT و BIT

دارای دقت ثابت: DECIMAL و NUMERIC

تقریبی: FLOAT و REAL

مقادیر پولی: MONEY و SMALLMONEY

تابع ISNUMERIC() در واقع بررسی می‌کند که آیا یک مقدار می‌تواند به یک نوع داده‌ای عددی تبدیل شود یا خیر، و سپس پاسخ صحیح را بازمی‌گرداند. به‌هرحال، این تابع به شما نمی‌گوید که کدام نوع داده‌ای، به‌خوبی می‌تواند جریان یا overflow را کنترل کند.

به همین دلیل از زمان انتشار SQL Server 2012 توابع TRY_PARSE()، TRY_CAST() و TRY_CONVERT() معرفی شدند.

مثال‌های تابع ISNUMERIC() در SQL Server

این مثال، از تابع ISNUMERIC() استفاده می‌کند تا بررسی کند که آیا رشته '$۱۰' می‌تواند به یک عدد تبدیل شود یا خیر:

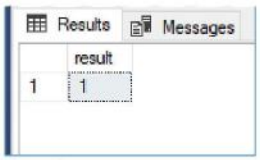
```
SELECT
    ISNUMERIC('$10') result;
```

خروجی به این صورت است:

مثال زیر بررسی می‌کند که آیا رشته '-2.23E-308' یک عدد هست یا خیر:

```
SELECT
    ISNUMERIC('-2.23E-308') result;
```

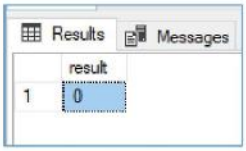
خروجی عبارت است از:



مثال زیر مقدار ۰ را بازمی‌گرداند که نشان‌دهنده این است که رشته '+ABC' یک عدد نیست:

```
SELECT
    ISNUMERIC('+ABC') result;
```

خروجی به این شکل است:



تابع IIF در SQL Server

تابع IIF() سه آرگومان می‌پذیرد. سپس اولین آرگومان را ارزیابی می‌کند و اگر مقدار آن true باشد، آنگاه دومین آرگومان را بازمی‌گرداند، در غیر این صورت، سومین آرگومان را بازمی‌گرداند.

کد زیر، syntax تابع IIF() را نشان می‌دهد:

```
IIF(boolean_expression, true_value, false_value)
```

در این syntax:

- Boolean_expression یک عبارت است که باید ارزیابی شود. این عبارت باید یک عبارت boolean معتبر باشد، در غیر این صورت تابع ارور می‌دهد.
  - True_value مقداری است که اگر مقدار boolean_expression برابر با true باشد، بازگردانده می‌شود.
  - False_value مقداری است که اگر مقدار boolean_expressiom برابر با false باشد، بازگردانده می‌شود.

در واقع، تابع IIF() خلاصه شده یا short hand یک عبارت CASE است:

```
CASE
    WHEN boolean_expression
        THEN true_value
    ELSE
        false_value
END
```

مثال‌های تابع IIF() در SQL Server

الف) استفاده از تابع IIF() برای یک‌رشته

این مثال از تابع IIF() استفاده می‌کند تا بررسی کند که آیا ۲۰ > ۱۰ هست یا نه و رشته True را بازمی‌گرداند:

```
SELECT
    IIF(10 < 20, 'True', 'False') Result ;
```

خروجی به این شکل است:

Results | Messages

	Result
1	True

ب) استفاده از تابع IIF() برای ستون جدول

مثال زیر تابع IIF() را به‌صورت تودرتو درون توابع IIF() به کار می‌برد و وضعیت سفارش متناظر را بر اساس تعداد سفارش بازمی‌گرداند:

```
SELECT
    IIF(order_status = 1,'Pending',
        IIF(order_status=2, 'Processing',
            IIF(order_status=3, 'Rejected',
                IIF(order_status=4,'Completed','N/A')
            )
        )
    ) order_status,
    COUNT(order_id) order_count
FROM
    sales.orders
WHERE
    YEAR(order_date) = 2018
GROUP BY
    order_status;
```

تصویر زیر، خروجی را نشان می‌دهد:

Results | Messages

	order_status	order_count
1	Pending	62
2	Processing	63
3	Rejected	13
4	Completed	154

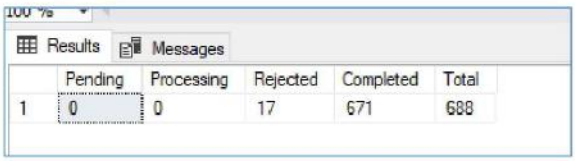
پ) استفاده از تابع IIF() برای توابع تجمعی

این مثال از تابع IIF() برای تابع SUM() استفاده می‌کند تا تعداد سفارش‌ها را بر اساس وضعیت سفارش در سال ۲۰۱۸ به دست آورد:

```
SELECT
    SUM(IIF(order_status = 1, 1, 0)) AS 'Pending',
    SUM(IIF(order_status = 2, 1, 0)) AS 'Processing',
    SUM(IIF(order_status = 3, 1, 0)) AS 'Rejected',
    SUM(IIF(order_status = 4, 1, 0)) AS 'Completed',
    COUNT(*) AS Total
FROM
    sales.orders
WHERE
    YEAR(order_date) = 2017;
```

در این مثال، اگر وضعیت متناظری پیدا شد، تابع IIF() مقدار ۱ یا ۰ را بازمی‌گرداند. تابع SUM() تعداد سفارش‌ها را برای هر وضعیت بازمی‌گرداند.

خروجی به این شکل است:



	Pending	Processing	Rejected	Completed	Total
1	0	0	17	671	688

فصل چهارم

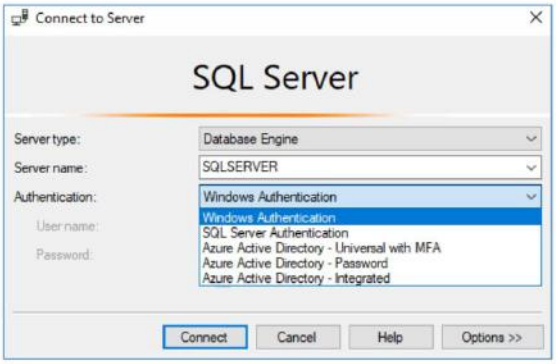
امنیت در SQL Server

یکی از مهم‌ترین بخش‌هایی که باید به آن توجه کرد امنیت در SQL است که اگر درست به آن توجه نکنیم می‌تواند باعث ایجاد مشکلات امنیتی و درز اطلاعات مهم سازمان شود..

زمانی که یک دیتابیس را ایجاد می‌کنید، امنیت زیادی نخواهد داشت و اگر روش ایجاد امنیت بر روی دیتابیس را ندانید، شاید تمام اطلاعت خود را از دست بدهید.

٤-١ روش‌های احراز هویت

برای اینکه روش‌های احراز هویت در SQL Server 2019 را بررسی کنیم بهتر است صفحه‌ی اول نرم‌افزار SSMS را مشاهده کنید که در شکل ١-٤ مشخص شده است.



شکل ۱-۴ احراز هویت

همان‌طور که در شکل ١-٤ مشاهده می‌کنید احراز هویت SQL از چند قسمت تشکیل شده است که در زیر همه‌ی آنها را بررسی خواهیم کرد:

Windows Authentication

یک روش احراز هویت است که در سرویس‌گیرنده‌های ویندوزی تعبیه شده است، این یک روش پیش‌فرض است که خود ویندوز بر روی آن تأکید می‌کند، در زمان نصب SQL این روش به‌صورت پیش‌فرض اجرا خواهد شد و کاربر موردنظر برای ورود به SQL تأیید خواهد شد.

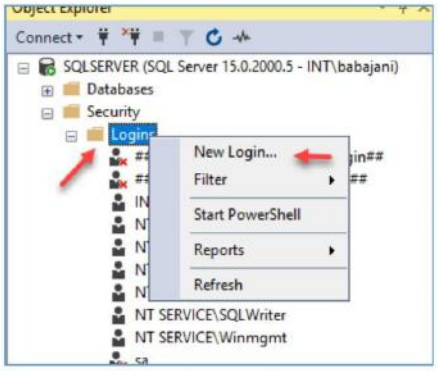
اصولاً سازمان‌ها برای مدیریت آسان‌تر کاربران یک سرویس Active directory در شبکه خود ایجاد می‌کنند و کاربران را به‌صورت متمرکز در آن تعریف می‌کنند و بعد می‌توان از طریق احراز هویت Windows Athentication به این کاربران دسترسی داد، اصولاً هر کاربر در ویندوز یک شماره‌ی خاص با نام SID دارد که با بقیه متمایز خواهد بود.

SQL Server Authentication

این روش از احراز هویت مختص نرم‌افزار SQL است و رمز عبور و نام کاربری در داخل دیتابیس نرم‌افزار تعریف می‌شود.

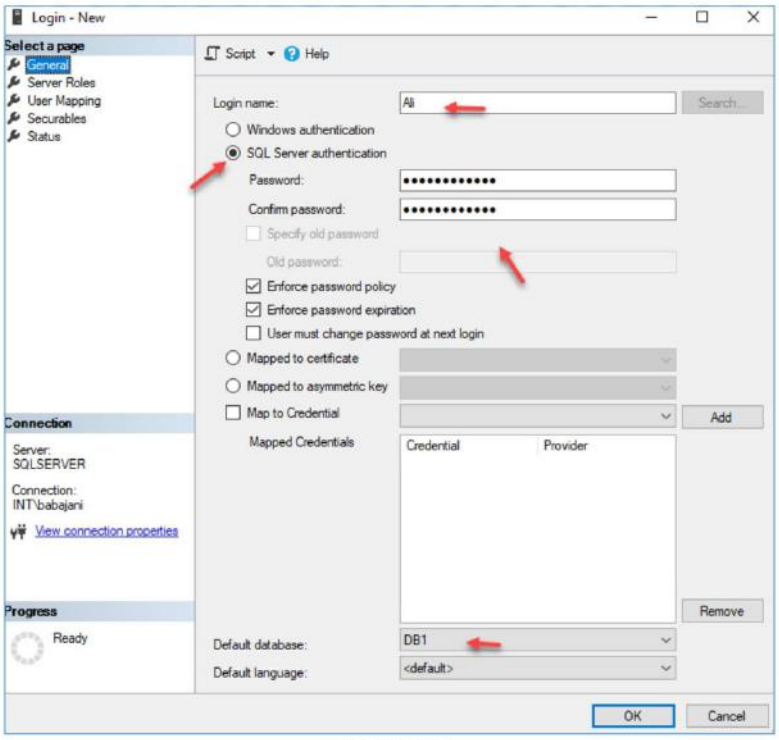
۲-٤ ایجاد کاربر و اعطای دسترسی به آن

همان‌طور که در اوایل کتاب گفتیم، زمانی که سرور SQL را عضو شبکه دامین می‌کنید، می‌توانید از کاربران Active Directory برای دسترسی به SQL استفاده کنید و مجوزهای لازم را برای آن در نظر بگیرید، اما اگر بخواهید کاربر محلی در SQL ایجاد کنید باید به‌مانند شکل ۲-٤ این کار را انجام دهید:



شکل ۲-۴ ایجاد کاربر

در شکل ۳-٤ باید کاربر خود را ایجاد کنید، در قسمت Login Name باید نام کاربر خود را وارد کنید و در زیر آن SQL Server Authentication را انتخاب کنید، با انتخاب این گزینه شما یک کاربر در داخل SQL Server ایجاد می‌کنید.



شکل ۳-۴ / ایجاد کاربر

یک رمز عبور پیچیده وارد کنید مانند Test@12345، اگر می‌خواهید رمز عبور شما با خط مش سازمان شما یکی باشد باید تیک گزینه‌ی Enforce password policy را انتخاب کنید، خط مش سازمان برای رمز عبور می‌تواند موارد زیر باشد:

زمانی که این گزینه را انتخاب می‌کنید باید موارد زیر رعایت شود:

۱- رمز عبور نباید شامل نام کاربری باشد.

۲- رمز عبور حداقل هشت کاراکتر باشد.

۳- رمز عبور باید شامل حروف (a-z)، (A-Z) و اعداد (0-9) باشد.

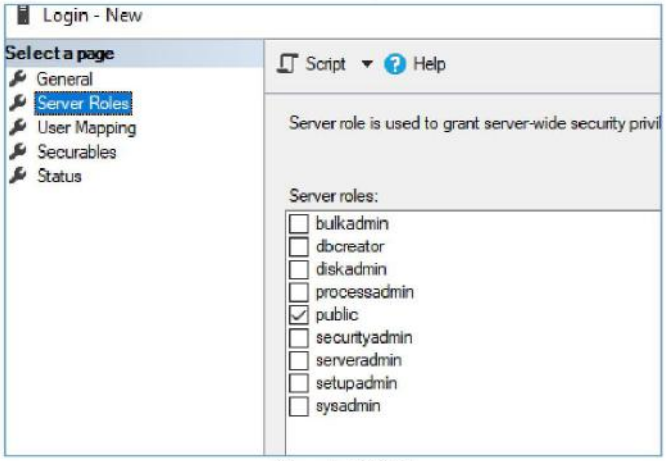
۴- رمزهای غیرالفبایی مانند: تعجب (!) ، علامت دلار ($) ، علامت عدد (#) یا درصد (٪).

در ادامه کار اگر تیک گزینه‌ی Enforce Password Expiration را انتخاب کنید به این معنا است که یک تاریخ برای انقضاء رمز عبور کاربران در نظر گرفته خواهد شد و قبل از اینکه تاریخ انقضاء برسد به کاربران هشدارهای لازم داده خواهد شد تا رمز عبور خود را تغییر دهند.

گزینه‌ی User Must Change Passeowrd at next Login را انتخاب کنید کاربر بعد از ورود باید رمز عبور را خودش تغییر دهد.

در آخر شکل ۳-٤ می‌توانید کاربر موردنظر را به دیتابیس موردنظر خود تخصیص دهید، تا دسترسی اولیه به آن را داشته باشد.

در ادامه کار به‌مانند شکل ٤-٤ وارد تب Server Role شوید، در این قسمت گزینه‌های مختلف دسترسی وجود دارد که در جدول ۱-٤ به طور کامل آن را بررسی می‌کنیم.



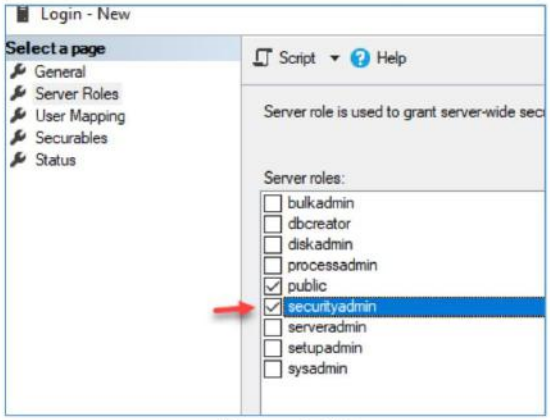
شکل ۴-۴ دسترسی به کاربر

جدول ۱-۴ بررسی *Role* دسترسی

نام Role	توضیحات
sysadmin	کاربران عضو sysadmin می‌توانند هر فعالیتی را در سرور انجام دهند .یعنی آزاد و رها هستند.
serveradmin	کاربران عضو serveradmin می‌توانند پیکربندی سرور موردنظر را تغییر دهند و همچنین می‌توانند سرور را خاموش کنند.
securityadmin	اعضای گروه securityadmin می‌توانند قسمت Login و جزئیات آنها را مدیریت کنند، کاربرانی که عضو این نقش هستند توانایی این را دارند که مجوزهای سطح سرور را اعطا کنند مانند GRANT و DENY و یا REVOKE ، آنها همچنین می‌توانند در صورت دسترسی به دیتابیس مجوزهای آن را هم مدیریت کنند، علاوه بر این هم آنها می‌توانند رمزهای عبور کاربران را ریست کنند. به این نکته توجه کنید که این نقش بسیار قدرتمند است و باید مواظب باشید که به هر کاربری آن را اعطا نکنید، این نقش معادل نقش sysadmin در سرور است.
processadmin	کاربرانی که عضو نقش processadmin باشند می‌توانند فرایندهای در حال اجرا را خاتمه دهند.

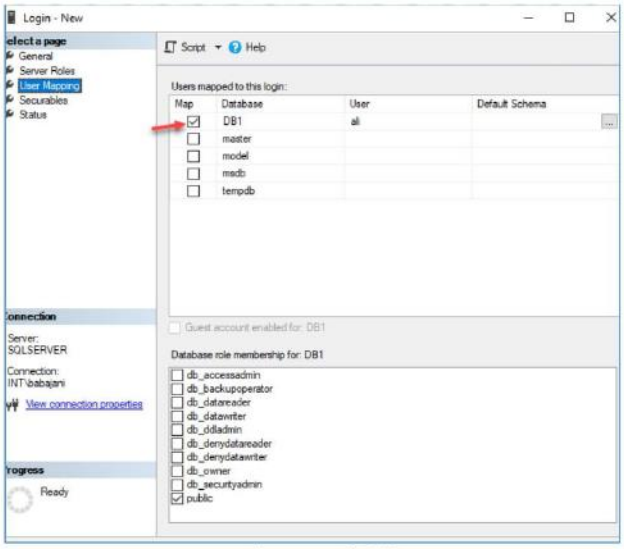
setupadmin	کاربران این نقش می‌توانند با استفاده از زبان پرس‌وجو یا همان Transact-SQL سرورها SQL را اضافه یا حذف کنند، البته در صورت استفاده از Management Studio کاربر موردنظر حتماً باید عضو sysadmin هم باشد.
bulkadmin	اعضای نقش bulkadmin می‌توانند دستور BULK INSERT را اجرا کنند (دستور BLUNK INSERT می‌تواند با استفاده از فایل‌های txt یا csv مقادیر زیادی از اطلاعات را به جداول اضافه کند).
diskadmin	کاربران عضو نقش diskadmin می‌توانند فایل‌های موجود بر روی دیسک را مدیریت کنند.
dbcreator	کاربران دارای نقش dbcreator می‌توانند هر پایگاه داده‌ای را ایجاد، مدیریت و حذف کنند و حتی می‌توانند پایگاه‌داده‌ی موردنظر را بازیابی کنند.
public	زمانی که یک کاربر ایجاد می‌کنید به‌صورت پیش‌فرض نقش public به آن تعلق می‌گیرد، این نقش به این صورت است که زمانی که مدیر یک object ایجاد می‌کند می‌تواند مشخص کند که کاربرانی که نقش public را دارند بتوانند به آن object دسترسی داشته باشند.

برای اینکه به کاربر ali دسترسی کامل دهیم تیک گزینه‌ی securityadmin را به‌مانند شکل ٥-٤ انتخاب می‌کنیم.



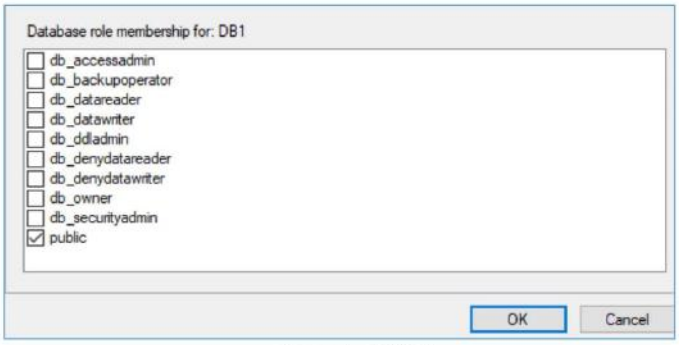
شکل ٥-٤ دسترسی به کاربر

در ادامه کار وارد تب User Mapping شوید در این صفحه که در شکل ٦-٤ مشخص شده است، شما می‌توانید مشخص کنید که کاربر موردنظر بر روی کدام دیتابیس دسترسی لازم را داشته باشد.



شکل ۴-۶ دسترسی به دیتابیس

در پایین شکل ٦-٤ که در شکل ٧-٤ مشخص شده است چندین Role وجود دارد که می‌توانید برای دسترسی به دیتابیس توسط کاربر موردنظر مشخص کنید، برای اینکه بیشتر با گزینه‌های موردنظر آشنا شویم به جدول ٢-٤ توجه کنید.



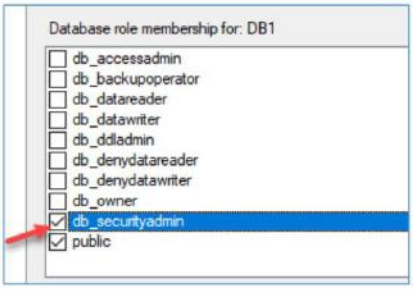
شکل ۴-۷ دسترسی به دیتابیس

جدول ۴-۲ دسترسی Role

نام Role	توضیحات
db_owner	کاربرانی که نقش db_owner را دریافت می‌کنند می‌توانند بر کلیه‌ی تنظیمات و نگهداری پایگاه‌داده موردنظر را انجام دهند و همچنین می‌توانند پایگاه‌داده را در sql حذف کنند،

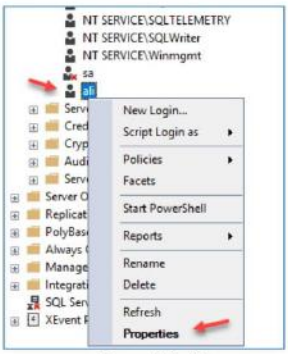
	البته در بعضی از مواقع این نوع دسترسی کارایی ندارد و باید مجوز سطح سرور به آنها داده شود.
db_securityadmin	اعضای این نقش توانایی تغییر عضویت در نقش‌ها را دارند و همچنین می‌توانند وظیفه مدیریت اختیارات را برعهده دارند.
db_accessadmin	این نقش به کاربران این اجازه را می‌دهد که بر روی Login کردن در SQL نظارت و دسترسی داشته باشد.
db_backupoperator	این قابلیت را به کاربران می‌دهد تا بتوانند از پایگاه‌داده پشتیبان تهیه کنند.
db_ddladmin	اعضای این گروه توانایی اجرای دستورات از نوع DDL را دارند.
db_datawriter	اعضای این گروه توانایی تغییر، حذف، و اضافه‌کردن هیچ اطلاعاتی را در جدول‌های تعریف شده توسط کاربر ندارند.
db_datareader	اعضای این نقش، توانایی خواندن اطلاعات از جدول‌های تعریف شده توسط کاربر را دارد.
db_denydatawriter	اعضای این نقش نمی‌توانند داده‌های موجود در جداول کاربران موجود در یک پایگاه‌داده را اضافه، اصلاح یا حذف کنند.
db_denydatareader	کاربرانی که این نقش را دارند نمی‌توانند هیچ داده‌ای را از جدول‌های تعریف شده توسط کاربر بخوانند.

در شکل ۸-٤ تیک گزینه‌ی db_securityadmin را انتخاب کنید تا کاربر مورد نظر دسترسی کامل به دیتابیس داشته باشد



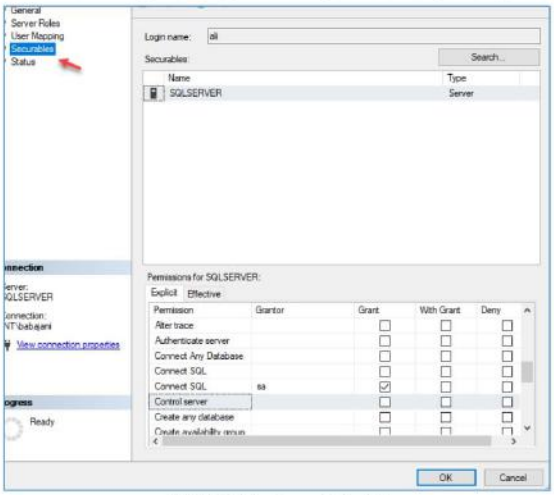
شکل ۸-۴ دسترسی به دیتابیس

بعد از انجام کار می‌توانید بر روی OK کلیک کنید تا کاربر مورد نظر ایجاد شود، بعد از ایجاد کاربر به‌مانند شکل ۹-٤ بر روی کاربرمورد نظر کلیک راست کنید و گزینه‌ی Properties را انتخاب کنید.



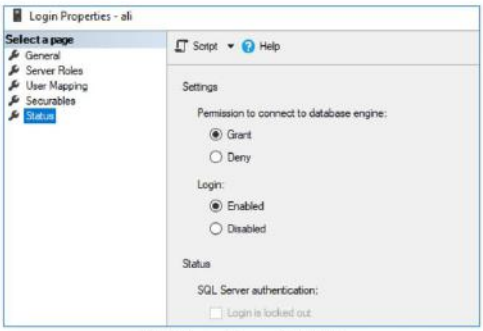
شکل ۹-۴ بررسی کاربر

در شکل ۱۰-٤ جزئیات مربوط به کاربر ali را مشاهده می‌کنید، در تب Securables یک سری دسترسی‌های پیشرفته وجود دارد که می‌توانید از طریق مدیر SQL به کاربران و دیتابیس مورد نظر اعطا کنید.



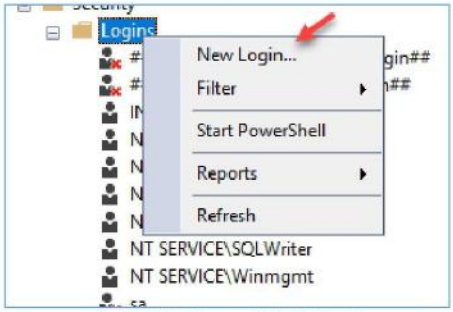
شکل ۱۰-۴ بررسی تب *Securables*

در تب Status که در شکل ۱۱-٤ مشخص شده است می‌توانید مشخص کنید که کاربر مورد نظر توانایی متصل شدن به Database Engine را داشته باشد یا نه و یا اینکه کاربر مورد نظر را فعال و یا غیرفعال کنید.



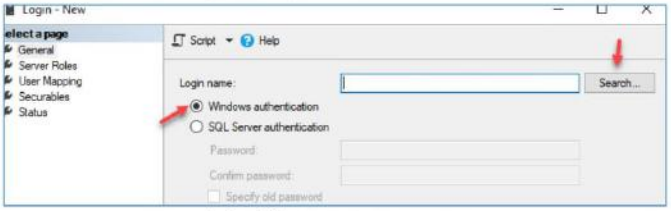
شکل ۱۱-۴ بررسی قسمت *Status*

اگر بخواهیم یک کاربر از طریق Active Directory را به لیست Logins اضافه کنیم باید به‌مانند شکل ۱۲-٤ بر روی Logins کلیک راست کنید و گزینه‌ی New Login را انتخاب کنید.



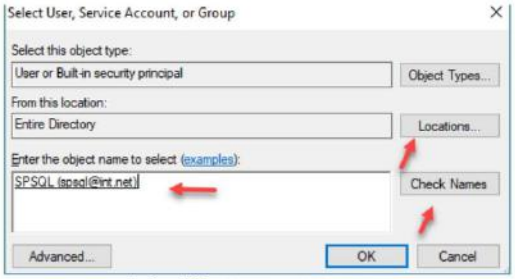
شکل ۱۲-۴ ایجاد کاربر *Active Directory*

در شکل ۱۳-٤ باید گزینه‌ی Windows authentication را انتخاب کنید و بر روی Search کلیک کنید.



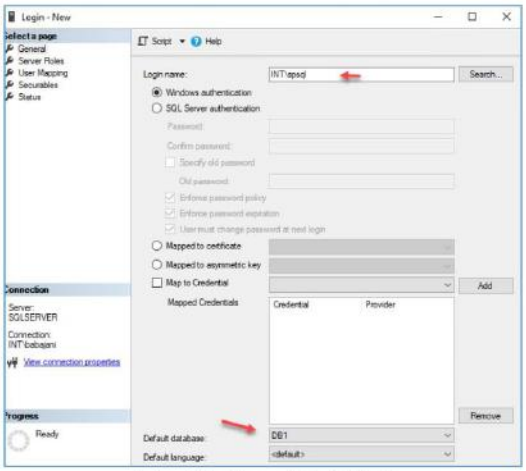
شکل ۱۳-۴ ایجاد کاربر *Active Directory*

در شکل ۱٤-٤ باید از قسمت Location دومین خود را انتخاب کنید و کاربر مورد نظر خود را در کادر مشخص شده وارد کنید تا با کلیک بر روی Check Names آن را پیدا کنید.



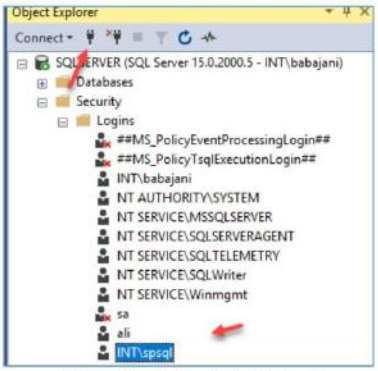
شکل ۴-۱۴ ایجاد کاربر *Active Directory*

اگر به شکل ۱۵-٤ توجه کنید بعد از انتخاب کاربر از نوع دومین دیگر نمی‌توانید رمز عبور و دیگر گزینه‌ها را خودتان وارد و انتخاب کنید بلکه این اطلاعات از طریق سرویس Active Directory انجام می‌گیرد.



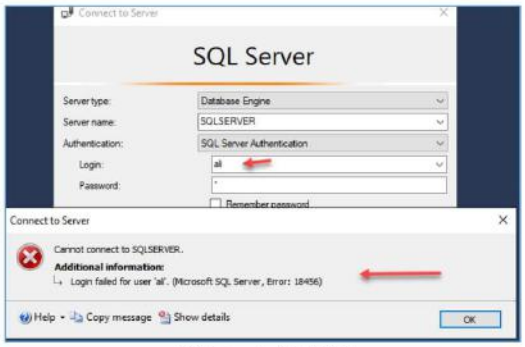
شکل ۴-۱۵ ایجاد کاربر *Active Directory*

همان‌طور که در شکل ۱۶-٤ مشاهده می‌کنید هم کاربر ali و هم کاربر spsql که از نوع دومین بوده به لیست SQL اضافه شده است، برای اینکه با این کاربران وارد SQL شویم باید بر روی آیکون مورد نظر کلیک کنید.



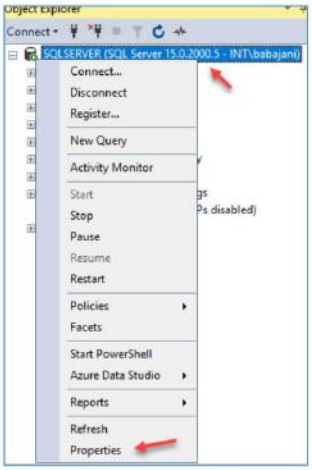
شکل ۴-۱۶ ایجاد کاربر *Active Directory*

قبل از اینکه ادامه دهیم باید این نکته را متذکر شویم که برای ورود با کاربر تحت دومین باید با همان کاربر اول وارد ویندوز شوید و بعد از آن می‌توانید با انتخاب windows Authentication از آن کاربر استفاده کنید ولی برای ورود با کاربر ali که از نوع SQL Authentication است باید گزینه‌ی SQL Server Authentication را انتخاب کنید و طبق شکل ٤-١٧ کاربر ALI را وارد و بر روی Connect کلیک کنید، بعد از کلیک با خطایی روبرو خواهید شد که اجازه ورود به سرور را به کاربر ali نمی‌دهد.



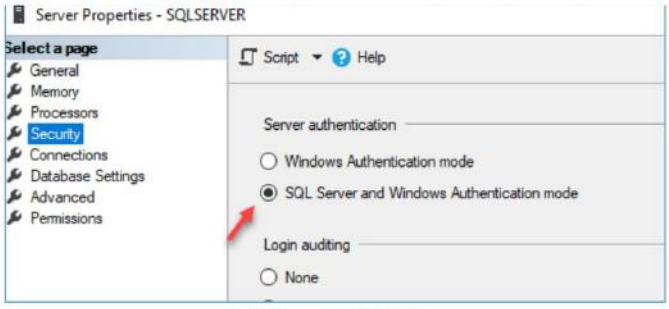
شکل ١٧-٤ احراز هویت SQL

برای حل خطای شکل ١٧-٤ باید به‌مانند شکل ١٨-٤ بر روی سرور کلیک راست کنید و گزینه‌ی Properties را انتخاب کنید.



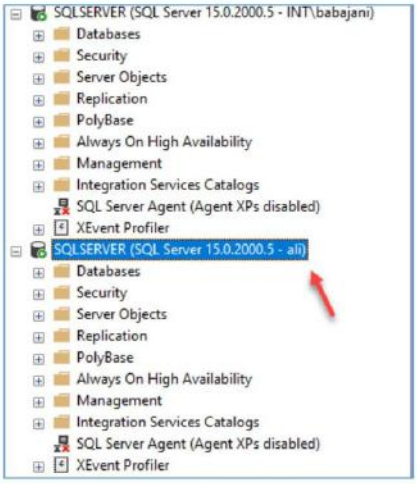
شکل ١٨-٤ احراز هویت SQL

در شکل ١٩-٤ وارد تب Security شوید و گزینه‌ی SQL Server and Windows Authentication mode را انتخاب کنید، با این کار هم با احراز هویت SQL و هم Windows می‌توانیم وارد سرور SQL شویم.



شکل ۴-۱۹ احراز هویت SQL

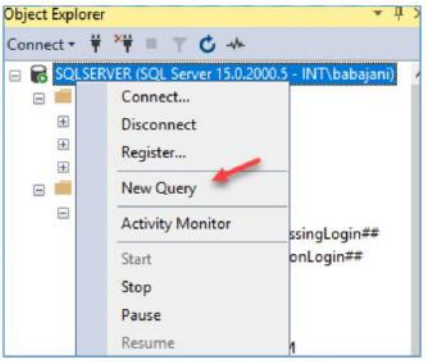
در شکل ۴-۲۰ مشخص شده است که کاربر ali که از نوع SQL Authentication است توانسته وارد سرور SQL شود.



شکل ۴-۲۰ احراز هویت SQL

۴-۳ ایجاد کاربر از طریق Query

یکی دیگر از راه‌های ایجاد کاربر یا موارد خاص دیگر استفاده از دستورات SQL است که در این قسمت می‌خواهیم این کار را انجام دهیم، برای شروع باید بر روی نام سرور SQl کلیک راست کنید و گزینه‌ی New Query را به‌مانند شکل ۴-۲۱ انتخاب کنید.



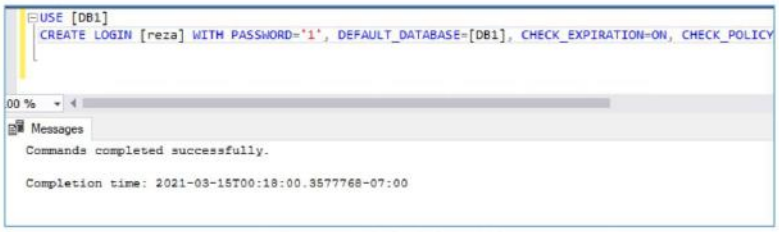
شکل ۴-۲۱ ایجاد کاربر از طریق *Query*

در صفحه بازشده‌ی شکل ٤-۲۲ دستورات زیر را وارد کنید و بعد از آن کلید F5 را فشار دهید:

```
USE [DB1]
CREATE LOGIN [reza] WITH PASSWORD='1', DEFAULT_DATABASE=[DB1], CHECK_EXPIRATION=ON,
CHECK_POLICY=ON
```

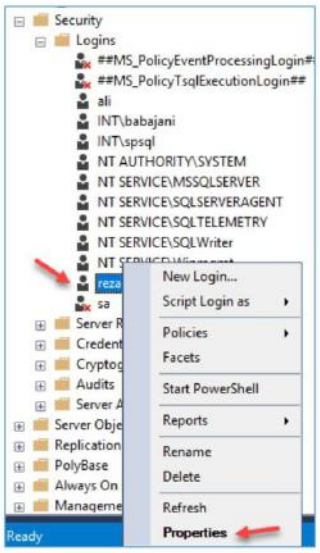
در دستور بالا و در خط اول باید مشخص کنیم که بر روی چه دیتابیسی قرار است کار کنیم که در اینجا دیتابیس DB1 انتخاب شده است، در خط بعد باید دستورات اصلی را وارد کنیم، با دستور CREATE LOGIN مشخص می‌کنیم که می‌خواهیم یک LOGIN جدید ایجاد کنیم که بعد از آن هم در [] نام کاربر را که reza است وارد می‌کنیم و با دستور WITH PASSWORD رمز عبور آن را که حتماً هم باید در '' باشد را مشخص می‌کنیم، در ادامه باید ویژگی‌های این کاربر را مشخص کنیم، مثلاً برای اینکه مشخص کنیم که دیتابیس پیش‌فرضی که کاربر بر روی آن کار می‌کند را باید با دستور DEFAULT_DATABASE=[DB1] مشخص کنیم و بعد از آن می‌توانیم ویژگی‌های دیگر آن مانند Enforce Password Policy و Enforce Password Expiration را فعال کنید.

بعد از درست واردکردن دستور مورد نظر بر روی کلید F5 فشار دهید تا به‌مانند شکل ٤-۲۲ کاربر مورد نظر ایجاد شود.



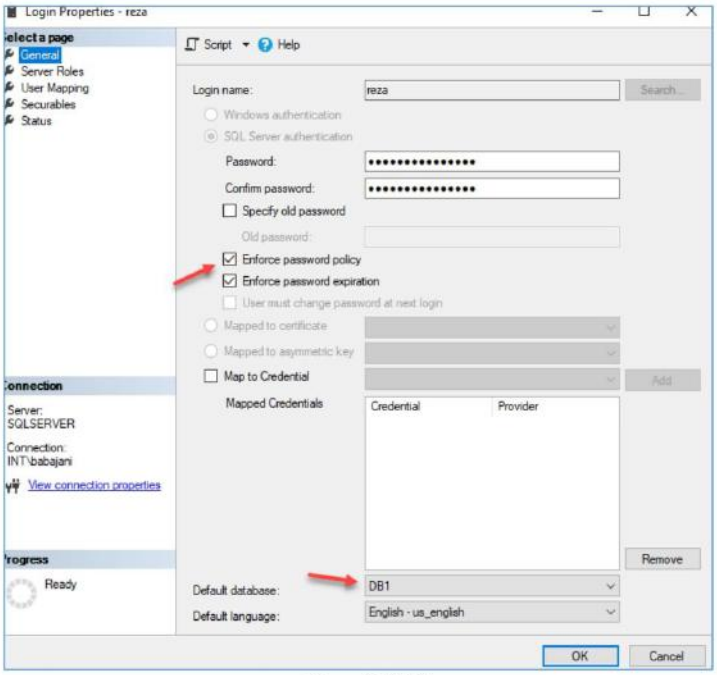
شکل ۴-۲۲ ایجاد کاربر از طریق *Query*

اگر به شکل ٤-۲۳ توجه کنید، کاربر مورد نظر در لیست Logins قرار گرفته است و برای اینکه بررسی بیشتری داشته باشیم بر روی آن کلیک راست کنید و گزینه‌ی Properties را انتخاب کنید.



شکل ۲۳-۴ بررسی کاربر

همان‌طور که در شکل ٢٤-٤ مشاهده می‌کنید اطلاعات کاربر reza دقیقاً همان چیزی است که در دستورات وارد کردیم.



شکل ۲۴-۴ بررسی کاربر

اگر بخواهیم با استفاده از دستور مشخص کنیم که کاربر مورد نظر فعال باشد و یا غیرفعال باید دستور زیر را وارد کنیم:

```
ALTER LOGIN reza ENABLE;
```

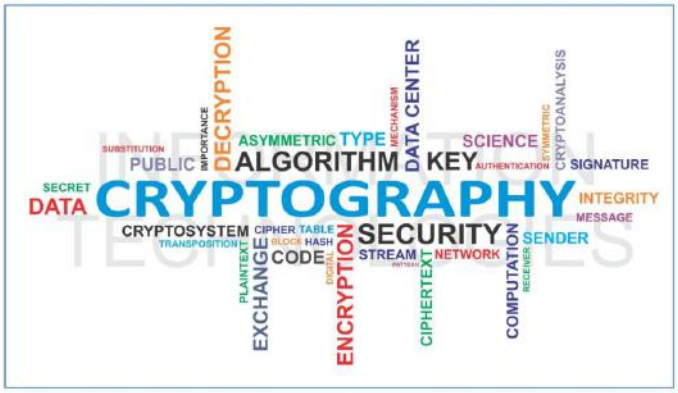
در دستور بالا کاربر reza فعال می‌شود و اگر بخواهید غیرفعال کنید باید از دستور DISABLE استفاده کنید.

٤-٤ رمزگذاری بر روی دیتابیس

یکی از مهم‌ترین کارها در SQL ایجاد امنیت برای دیتابیس‌ها و ستون‌های جدول آن است که با این کار دسترسی به اطلاعات دیتابیس کار هرکسی نخواهد بود و طرف باید رمز عبور و گواهینامه موردنیاز را داشته باشد.

مثلاً شما در جدول خود یک ستون دارید که نمی‌خواهید کس دیگری بتواند اطلاعات آن را مشاهده کند و شما با داشتن رمز عبور و گواهینامه مورد نظر میتونید اطلاعات را مشاهده کنید.

رمزنگاری یا همان Cryptography به دانشی گفته می‌شود که در آن اطلاعات به‌صورت یک کلید رمزنگاری می‌شود که این کار توسط الگوریتم مربوط به آن انجام می‌شود و فقط کسی می‌تواند از این اطلاعات استفاده کند که اطلاعات لازم مانند کلید و الگوریتم مربوط به آن را بداند.



شکل ۴-۲۵ رمزنگاری

بررسی برخی از اصطلاحات

Decryption

برای آشکارسازی اطلاعات Encryption شده مورداستفاده قرار می‌گیرد و نام آن را رمزگشایی هم می‌نامند.

Plain text

متن اولیه که رمزنگاری نشده و یک رمز آشکار است و مهاجمان به‌راحتی می‌توانند به آن دست پیدا کنند.

Cipher

الگوریتمی برای رمزگذاری و رمزشکنی است و از سرعت عمل خوبی برخوردار است.

Cryptanalysis

به بازکردن قفل‌های Cipher گفته می‌شود یا خواندن متن قفل‌شده‌ی آن.

۱-٤-٤ کلیدهای متقارن (Symmetric) و نامتقارن (Asymmetric)

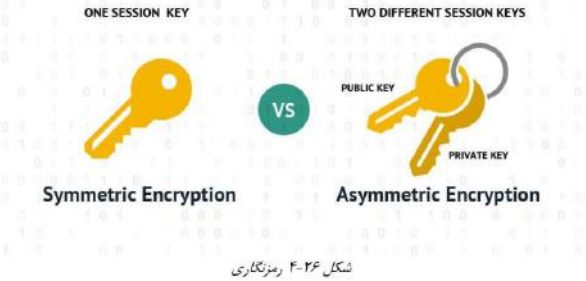
رمزنگاری کلید متقارن

رمزنگاری کلید متقارن یا تک کلیدی، به آن دسته از الگوریتم‌ها، پروتکل‌ها و سیستم‌های رمزنگاری گفته می‌شود که در آن هر دو طرف ردوبدل اطلاعات از **یک کلید رمز یکسان** برای عملیات رمزگذاری و رمزگشایی استفاده می‌کنند. در این قبیل سیستم‌ها، یا کلیدهای رمزگذاری و رمزگشایی یکسان هستند یا با رابطه‌ای بسیار ساده از یکدیگر قابل استخراج هستند.

واضح است که در این نوع از رمزنگاری، باید یک کلید رمز مشترک بین دو طرف تعریف گردد. چون کلید رمز باید کاملاً محرمانه باقی بماند، برای ایجاد و ردوبدل کلید رمز مشترک باید از کانال امن استفاده نمود یا از روش‌های رمزنگاری نامتقارن استفاده کرد. نیاز به وجود یک کلید رمز به‌ازای هر دو نفرِ درگیر در رمزنگاری متقارن، موجب بروز مشکلاتی در مدیریت کلیدهای رمز می‌گردد.

الگوریتم‌هایی که در Symmetric به کار می‌رود عبارت‌اند از:

- ✓ DES
- ✓ DES۳
- ✓ AES
- ✓ IDEA
- ✓ RC2, RC4, RC5, RC6
- ✓ Blowfish



شکل ۲۶-۴ رمزنگاری

رمزنگاری کلید نامتقارن

رمزنگاری کلید نامتقارن، در ابتدا با هدف حل مشکل انتقال کلید در روش متقارن پیشنهاد شد. در این نوع از رمزنگاری، به‌جای یک کلید مشترک، از یک زوج کلید به نام‌های کلید عمومی و کلید خصوصی استفاده می‌شود. کلید خصوصی تنها در اختیار دارندهٔ آن قرار دارد و امنیت رمزنگاری به محرمانه بودن کلید خصوصی بستگی دارد. کلید عمومی در اختیار کلیهٔ کسانی که با دارندهٔ آن در ارتباط هستند قرار داده می‌شود.

به‌مرورزمان، به‌غیراز حل مشکل انتقال کلید در روش متقارن، کاربردهای متعددی برای این نوع از رمزنگاری مطرح گردیده است. در سیستم‌های رمزنگاری نامتقارن، بسته به کاربرد و پروتکل مورد نظر، گاهی از کلید عمومی برای

رمزگذاری و از کلید خصوصی برای رمزگشایی استفاده می‌شود و گاهی نیز، برعکس، کلید خصوصی برای رمزگذاری و کلید عمومی برای رمزگشایی به کار می‌رود.

دو کلید عمومی و خصوصی با یکدیگر متفاوت هستند و با استفاده از روابط خاص ریاضی محاسبه می‌گردند. رابطهٔ ریاضی بین این دو کلید به‌گونه‌ای است که کشف کلید خصوصی با در اختیار داشتن کلید عمومی، عملاً ناممکن است.

الگوریتم‌هایی که در Asymmetric به کار می‌روند عبارت‌اند از:

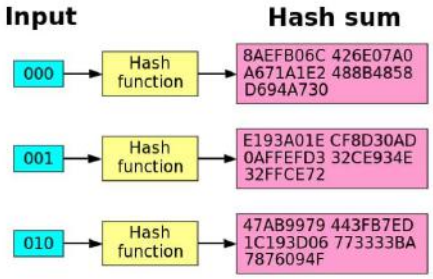
- RSA ✓
- DH ✓
- ElGamal ✓
- DSA ✓
- ECC ✓

مقایسه رمزنگاری کلید متقارن و کلید نامتقارن

اصولاً رمزنگاری کلید متقارن و کلید نامتقارن دارای دو ماهیت متفاوت هستند و کاربردهای متفاوتی نیز دارند؛ بنابراین مقایسه این دو نوع رمزنگاری بدون توجه به کاربرد و سیستم مورد نظر کار دقیقی نخواهد بود. اما اگر معیار مقایسه، به‌طور خاص، حجم و زمان محاسبات موردنیاز باشد، باید گفت که با درنظرگرفتن مقیاس امنیتی معادل، الگوریتم‌های رمزنگاری متقارن خیلی سریع‌تر از الگوریتم‌های رمزنگاری نامتقارن می‌باشند.

۲-٤-٤ هش کردن (Hashing)

در این روش یک ورودی از اطلاعات دریافت می‌شود و بعد از اجرای یک الگوریتم بر روی آن ورودی تبدیل به اعداد و حروف خواهد شد که در شکل ۲۷-٤ این موضوع را مشاهده می‌کنید که مثلاً با ورود عدد ۰۰۰ و اعمال الگوریتم هش روی آن کد نهایی آن به‌صورت کامل تغییر کرده و هک کردن آن کاملاً سخت شده است.



شکل ۲۷-۴ هشینگ

انواع الگوریتم‌های هش عبارت‌اند از:

نوع الگوریتم	اندازه
BLAKE-256	256 bits
BLAKE-512	512 bits
BLAKE2s	Up to 256 bits
BLAKE2b	Up to 512 bits

ECOH	224 to 512 bits
FSB	160 to 512 bits
GOST	256 bits
Grøstl	Up to 512 bits
HAS-160	160 bits
HAVAL	128 to 256 bits
JH	224 to 512 bits
MD2	128 bits
MD4	128 bits
MD5	128 bits
MD6	Up to 512 bits
RadioGatún	Up to 1216 bits
RIPEMD	128 bits
RIPEMD-128	128 bits
RIPEMD-160	160 bits
RIPEMD-320	320 bits
SHA-1	160 bits
SHA-224	224 bits
SHA-256	256 bits
SHA-384	384 bits
SHA-512	512 bits
SHA-3 (originally known as Keccak)	arbitrary
Skein	arbitrary
Snefru	128 or 256 bits
Spectral Hash	512 bits
Streebog	256 or 512 bits
SWIFFT	512 bits
Tiger	192 bits
Whirlpool	512 bits

از بین این الگوریتم‌ها بیشترین استفاده از الگوریتم‌های MD5، SHA1، SHA2 می‌شود.

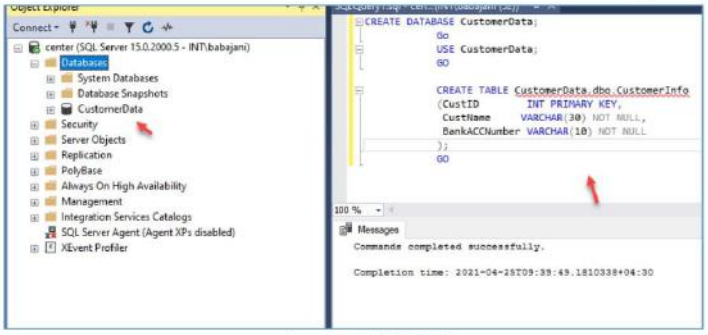
۳-٤-٤ رمزگذاری بر روی ستون‌های جداول در SQL

امنیت داده برای هر سازمانی یک کار اساسی و مهم است، به‌خصوص اگر اطلاعات شخصی مشتری مانند شماره تماس، آدرس ایمیل، شماره تأمین اجتماعی، شماره کارت‌های بانکی و اعتباری را ذخیره کنید. هدف اصلی ما محافظت از دسترسی غیرمجاز به داده‌ها در داخل و خارج از سازمان است. برای دستیابی به این هدف SQL Server راه‌حل‌های رمزگذاری ارائه می‌دهد. ما می‌توانیم از این رمزگذاری‌ها استفاده کنیم و از داده‌ها محافظت کنیم.

برای شروع کار می‌خواهیم یک دیتابیس جدید ایجاد کنیم و یکی از ستون‌های آن را رمزنگاری کنیم، برای اینکه یک دیتابیس جدید ایجاد کنیم می‌توانید از دستور زیر استفاده کنید:

```
CREATE DATABASE CustomerData;
            Go
            USE CustomerData;
            GO

            CREATE TABLE CustomerData.dbo.CustomerInfo
            (CustID        INT PRIMARY KEY,
             CustName      VARCHAR(30) NOT NULL,
             BankACCNumber VARCHAR(10) NOT NULL
            );
            GO
```

با اجرای دستورات بالا یک دیتابیس جدید با نام CoustomerData ایجاد شده است و در ادامه دستورات یک جدول با نام CustomerInfo هم داخل دیتابیس CoustomerData ایجاد شده که دارای سه ستون است که در شکل ۲۸-٤ این موضوع را مشاهده می‌کنید.

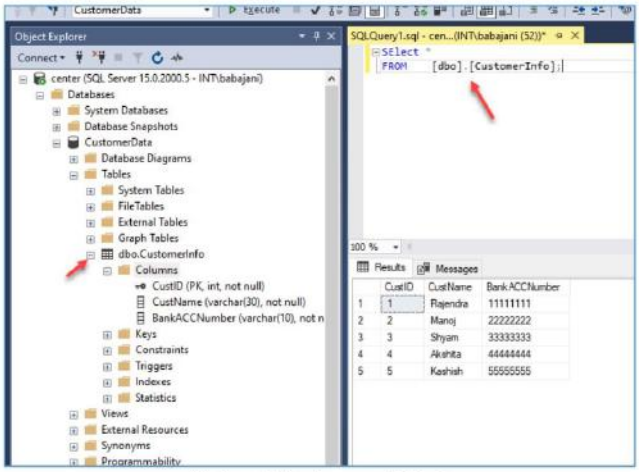


شکل ۲۸-۴ ایجاد جدول و دیتابیس

برای اینکه اطلاعات جدول را تکمیل کنیم می‌توانید از دستورات زیر استفاده کنید:

```
Insert into CustomerData.dbo.CustomerInfo (CustID,CustName,BankACCNumber)
          Select 1,'Rajendra',11111111 UNION ALL
          Select 2, 'Manoj',22222222 UNION ALL
          Select 3, 'Shyam',33333333 UNION ALL
          Select 4,'Akshita',44444444 UNION ALL
          Select 5, 'Kashish',55555555
```

با دستور بالا اطلاعات جدول CustomerInfo کامل خواهد شد که در شکل ۲۹-۲ این موضوع را مشاهده می‌کنید.



شکل ۲۹-۴ خروجی جدول *CustomerInfo*

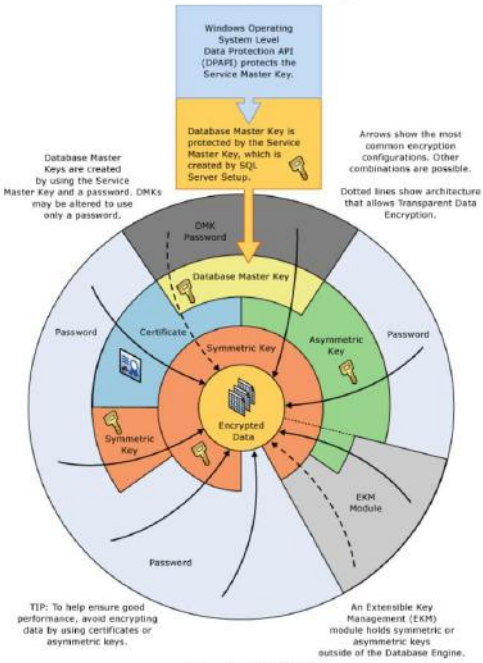
از مراحل زیر برای رمزگذاری سطح ستون استفاده می‌کنیم :

- یک کلید اصلی یا همان master key برای پایگاه‌داده ایجاد کنید.

- برای SQL Server یک گواهینامه خود امضا ایجاد کنید.
- یک کلید متقارن را برای رمزگذاری پیکربندی کنید.
- رمزگذاری داده‌های ستون.
- رمزگذاری را جستجو و تأیید کنید.

برای انجام این مراحل دقیقاً طبق دستورات زیر پیش بروید تا مشکلی در کار پیش نیاید.

اگر به شکل ٣٦-٤ توجه کنید، یک نمای کلی از ایجاد رمزگذاری در SQL Server را مشاهده می‌کنید.



شکل ٣٦-٤ نمودار رمزگذاری

١-٣-٤-٤ ایجاد Master Key برای رمزگذاری رو ستون

برای شروع یک Master Key تعریف و یک رمز عبور برای محافظت آن مشخص می‌کنیم که این کلید یک کلید متقارن است که برای محافظت از کلیدهای خصوصی و کلید نامتقارن تعریف می‌شود که در شکل ٣٦-٤ هم مشخص شده است.

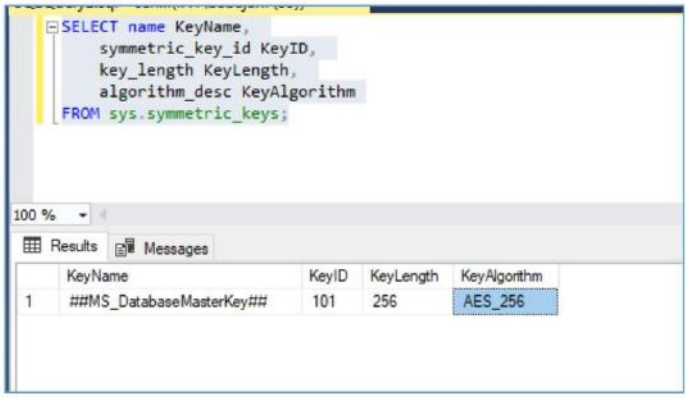
برای ایجاد یک کلید اصلی (Master Key) پایگاه‌داده از عبارت CREATE MASTER KEY استفاده می‌کنیم:

```
USE CustomerData;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'Test@12345';
```

برای اینکه متوجه شویم دستور بالا به‌درستی اجرا شده است باید از دستور زیر استفاده کنیم:

```
SELECT name KeyName,
    symmetric_key_id KeyID,
    key_length KeyLength,
    algorithm_desc KeyAlgorithm
FROM sys.symmetric_keys;
```

همان‌طور که در شکل ۳۷-٤ مشاهده می‌کنید با اجرای دستورات بالا یک View با ستون‌های مشخص شده در خروجی به نمایش گذاشته شد که مقدار sys.symetric_keys را نمایش می‌دهد که اگر به ستون الگوریتم توجه کنید الگوریتم ما از نوع AES و با اندازه 256 است که یک رمزنگاری قدرتمند را ارائه می‌دهد، به این نکته هم توجه کنید که خود SQL نوع الگوریتم به همراه طول آن را به‌صورت اتوماتیک ایجاد می‌کند.



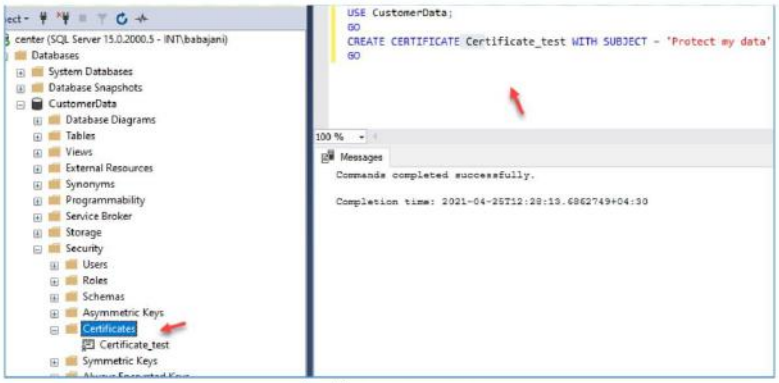
شکل ۳۷-۴

۲-۳-٤-٤ ایجاد Certificate در SQL

در ادامه کار باید یک گواهینامه خود امضا یا همان self-signed ایجاد کنیم منظور از گواهینامه‌های خود امضاء این است که ای گواهینامه داخل خود SQL ایجاد می‌شود و هیچ سازمان دیگری آن را تولید نمی‌کند یعنی دیگر SQL نیاز ندارد گواهینامهٔ مرجع تولید شده از سازمان دیگر استفاده کند بلکه خودش تولید و استفاده می‌کند.

```
USE CustomerData;
GO
CREATE CERTIFICATE Certificate_test WITH SUBJECT = 'Protect my data';
GO
```

در دستور بالا اول به دیتابیس CustomerData متصل شدیم و بعد یک گواهینامه‌ی جدید با نام Certificate_test ایجاد کردیم که موضوع آن Protect my data است و اگر به شکل ۳۸-٤ توجه کنید این موضوع را مشاهده خواهید کرد.

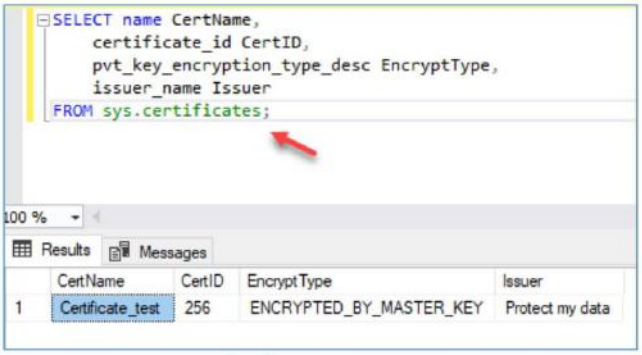


شکل ۳۸-۴ ایجاد *Certificate*

برای اینکه مطمئن شویم کار به‌درستی انجام‌گرفته می‌توانیم از دستورات زیر استفاده کنیم:

```
SELECT name CertName,
    certificate_id CertID,
    pvt_key_encryption_type_desc EncryptType,
    issuer_name Issuer
FROM sys.certificates;
```

اگر به شکل ۳۹-٤ توجه کنید متوجه خواهید شد که Certificate مورد نظر به‌درستی ایجاد شده است.



شکل ۳۹-۴ نمایش *Certificate*

پس اگر به ستون‌های شکل ۳۹-٤ توجه کنید در قسمت Encrypt Type مقدار ENCRYPTED_BY_MASTER_KEY قرار گرفته است که نشان می‌دهد SQL از کلیدی که ایجاد کردیم در حال استفاده است، در ستون Certname هم که نام Certificate قرار می‌گیرد، و در ستون Issuer باید نام سازمان صادرکننده گواهینامه نوشته شود که در اینجا چون خود SQL صادرکننده آن است یک نوشته خودمان قرار دادیم.

۳-۳-٤-٤ ایجاد کلید متقارن

در مرحله بعد کار باید یک کلید متقارن ایجاد کنیم، در مورد کلید متقارن در قسمت‌های قبل توضیح دادیم، در کل کلید متقارن برای رمزگذاری و رمزگشایی استفاده می‌کند.

برای شروع باید از دستور زیر استفاده کنید:

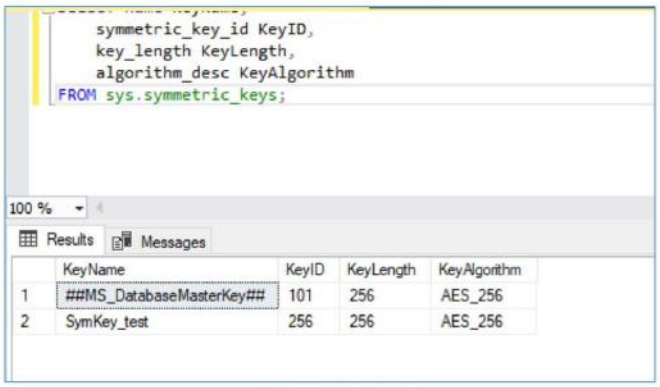
```
CREATE SYMMETRIC KEY SymKey_test WITH ALGORITHM = AES_256 ENCRYPTION BY CERTIFICATE
Certificate_test;
```

در دستور بالا SymKey_test نام کلید متقارن است که باید ایجاد کنیم و AES_256 همان طول و نوع رمزنگاری است که مشخص شده است و در آخر باید نام Cerfitficate که در قسمت قبل ایجاد کردیم را وارد کنید.

برای اینکه متوجه شویم دستور به‌درستی اجرا شده می‌توانیم از دستورات زیر استفاده کنیم:

```
SELECT name KeyName,
    symmetric_key_id KeyID,
    key_length KeyLength,
    algorithm_desc KeyAlgorithm
FROM sys.symmetric_keys;
```

همان‌طور که در شکل ٤٠-٤ مشاهده می‌کنید کلید متقارن هم به همراه کلید Master ایجاد شده است.



شکل ۴۰-۴ نمایش کلید متقارن

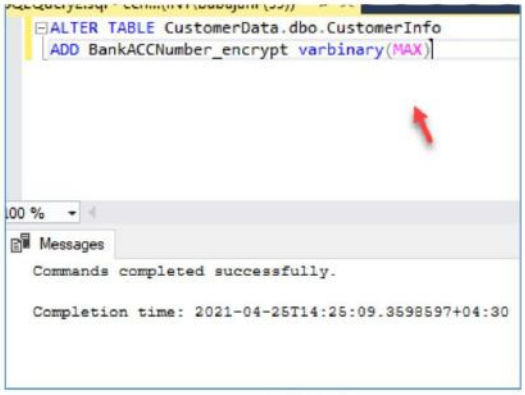
خوب تا به اینجای کار توانستیم کلیدهای رمزگذاری مورد نظر را برای این نسخه از دیتابیس ایجاد کنیم و روش ایجاد آن به‌مانند شکل ٣٦-٤ است یعنی اینکه SQL Server یک Service Master Key (SMK) ایجاد می‌کند و بعد از آن سیستم‌عامل Windows Data Protection API (DPAPI) از کلید Service Master Key (SMK) محافظت می‌کند، توجه داشته باشیم که کلید اصلی سرویس (SMK) از کلید اصلی پایگاه‌داده(DMK) محافظت می‌کند کلید اصلی پایگاه‌داده (DMK)از گواهی خود امضا شده که همان Certificate باشد محافظت می‌کند و این گواهی‌نامه یا همان Certificate از کلید Symmetric محافظت می‌کند. این شد مراحل کار تا به اینجا.

٤-٣-٤-٤ رمزگذاری داده

برای اینکه یک ستون در SQL رمزگذاری شود باید نوع داده را VARBINARY(max) در نظر بگیریم، برای این کار و در دیتابیس CustomerData یک ستون جدید با نام BankACCNumber_encrypt ایجاد و نوع آن را varbinary(MAX) در نظر می‌گیریم.

```
ALTER TABLE CustomerData.dbo.CustomerInfo
ADD BankACCNumber_encrypt varbinary(MAX)
```

همان‌طور که در شکل ٤١-٤ مشاهده می‌کنید ستون مورد نظر با موفقیت ایجاد شده است.

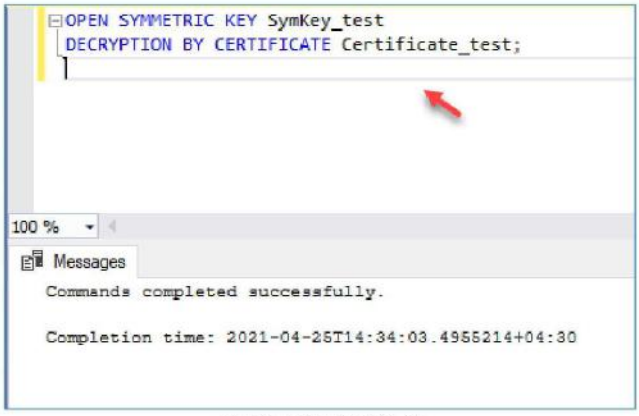


شکل ۴۱-۴ ایجاد ستون در جدول

در ادامه می‌خواهیم کار اصلی را انجام دهیم، یعنی اینکه داده‌هایی که در ستون BankACCNumber_encrypt که در بالا ایجاد کردیم قرار می‌گیرند را رمزنگاری کنیم.

پس برای این کار باید از همان کلید متقارن و گواهینامه قبلی که ایجاد کردیم استفاده کنیم، با دستور زیر کلید و گواهینامه را فعال می‌کنیم که در شکل ٤٢-٤ این موضوع مشخص شده است.

```
OPEN SYMMETRIC KEY SymKey_test
DECRYPTION BY CERTIFICATE Certificate_test;
```



شکل ۴۲-۴ بازکردن کلید و گواهینامه

در ادامه کار اطلاعات موجود در ستون BankACCNumber را درون ستون جدیدی که با نام BankACCNumber_encrypt است قرار می‌دهیم البته با استفاده از کلید متقارن که ایجاد کردیم این کار را انجام می‌دهیم.

```
UPDATE CustomerData.dbo.CustomerInfo
        SET BankACCNumber_encrypt = EncryptByKey (Key_GUID('SymKey_test'),
BankACCNumber)
        FROM CustomerData.dbo.CustomerInfo;
        GO
```

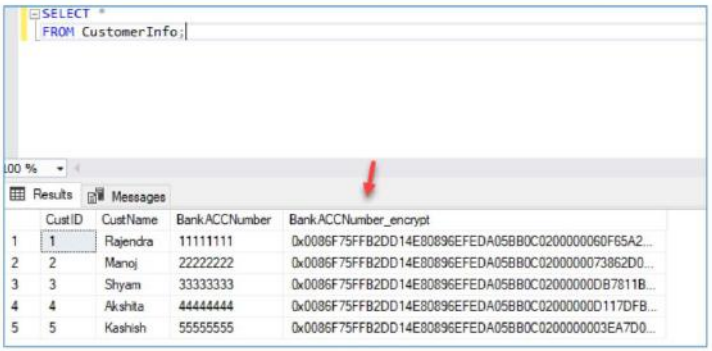
در ادامه باید کلید متقارن که باز کردید را ببندید که باید از این دستور استفاده کنید:

```
CLOSE SYMMETRIC KEY SymKey_test;
        GO
```

حالا می‌توانیم با استفاده از دستور زیر خروجی کار را مشاهده کنیم:

```
SELECT *
FROM CustomerInfo;
```

همان‌طور که در شکل ٤٣-٤ مشاهده می‌کنید ستون جدید ایجاد شده و اطلاعاتی که داخل آن قرار گرفته‌اند رمزنگاری شده و کسی نمی‌تواند از اطلاعات را مشاهده کند.



	CustID	CustName	BankACCNumber	BankACCNumber_encrypt
1	1	Rajendra	11111111	0x0086F75FFB2DD14E80896EFEDA05BB0C0200000060F65A2...
2	2	Manoj	22222222	0x0086F75FFB2DD14E80896EFEDA05BB0C0200000073862D0...
3	3	Shyam	33333333	0x0086F75FFB2DD14E80896EFEDA05BB0C02000000DB7811B...
4	4	Akshita	44444444	0x0086F75FFB2DD14E80896EFEDA05BB0C02000000D117DFB...
5	5	Kashish	55555555	0x0086F75FFB2DD14E80896EFEDA05BB0C0200000003EA7D0...

شکل ۴۳-۴ نمایش جدول

تا اینجا توانستیم یک ستون از جدول را رمزگذاری کنیم تا کسی نتواند اطلاعات آن را مشاهده کند، اما اگر بخواهیم این اطلاعات را مشاهده کنیم یعنی رمزگشایی یا همان Decrypt کنیم چه کاری باید انجام دهیم؟

برای این کار باید از دستور DecryptByKey استفاده کنیم، برای تست این موضوع به دستورات زیر دقت کنید:

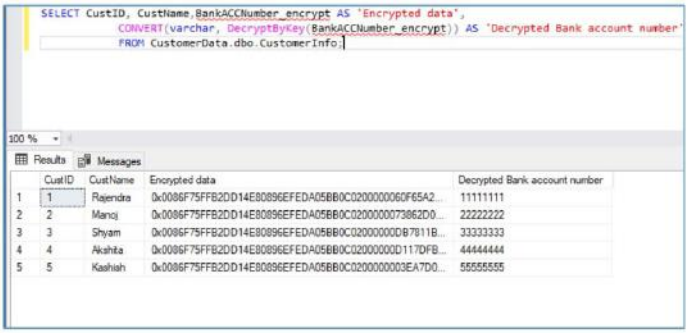
اولین کاری که انجام می‌دهیم کلید متقارن و گواهینامه مورد نظر را صدا می‌زنیم:

```
OPEN SYMMETRIC KEY SymKey_test
DECRYPTION BY CERTIFICATE Certificate_test;
```

بعد از بازکردن موارد مورد نظر باید از دستورات زیر استفاده کنید:

```
SELECT CustID, CustName,BankACCNumber_encrypt AS 'Encrypted data',
            CONVERT(varchar, DecryptByKey(BankACCNumber_encrypt)) AS 'Decrypted Bank
account number'
            FROM CustomerData.dbo.CustomerInfo;
```

در دستورات بالا ستون‌های جدول مورد نظر انتخاب شده‌اند ولی در ستون BankACCNumber_encrypt با استفاده از دستور AS به ستون Encrypted data تغییر نام پیدا کرد و همین ستون با استفاده از دستور DecryptByKey اطلاعات آن به ستون جدید با نام Decrypted Bank account number رمزگشایی شد که این موضوع را در شکل ٤٤-٤ مشاهده می‌کنید:

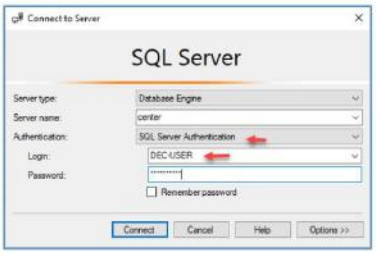


شکل ۴۴-۴ رمزگشایی ستون

بعد از رمزگذاری و رمزگشایی، حالا می‌خواهیم تست بگیریم که هر کاربری با هر دسترسی می‌تواند این عملیات را انجام دهد یا نه، برای تست این موضوع با استفاده از دستورات زیر یک کاربر جدید با نام DEC-USER ایجاد می‌کنیم که به دیتابیس CustomerData دسترسی db_datareader دارد یعنی فقط می‌تواند اطلاعات را بخواند.

```
USE [master]
GO
CREATE LOGIN [DEC-USER] WITH PASSWORD=N'Test@12345', DEFAULT_DATABASE=[CustomerData],
CHECK_EXPIRATION=OFF, CHECK_POLICY=OFF
GO
USE [CustomerData]
GO
CREATE USER [DEC-USER] FOR LOGIN [DEC-USER]
GO
USE [CustomerData]
GO
ALTER ROLE [db_datareader] ADD MEMBER [DEC-USER]
GO
```

بعد از ایجاد کاربر DEC-USER باید با این کاربر به‌مانند شکل ٤-٤٥ با کاربر جدید وارد SQL شوید.

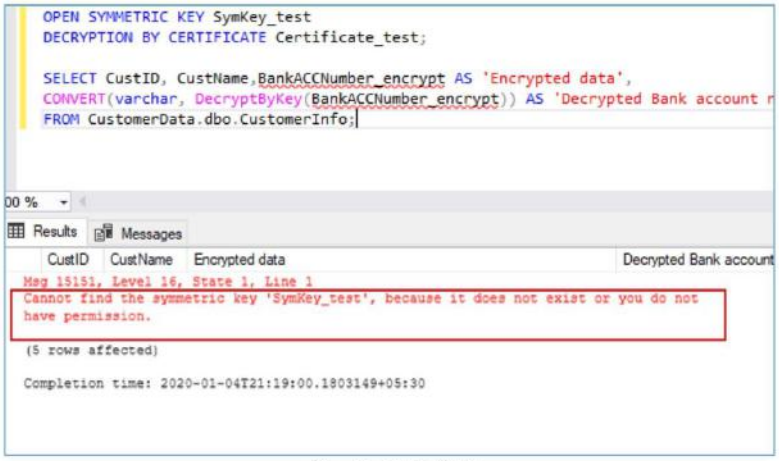


شکل ۴۵-۴ ورود به SQL

بعد از ورود دستور زیر را در New Query اجرا کنید:

```
OPEN SYMMETRIC KEY SymKey_test
DECRYPTION BY CERTIFICATE Certificate_test;

SELECT CustID, CustName,BankACCNumber_encrypt AS 'Encrypted data',
CONVERT(varchar, DecryptByKey(BankACCNumber_encrypt)) AS 'Decrypted Bank account
number'
FROM CustomerData.dbo.CustomerInfo;
```

با اجرای دستور بالا با خطای شکل ٤٦-٤ مواجه خواهید شد که اشاره به دسترسی نداشتن کاربر مورد نظر دارد.

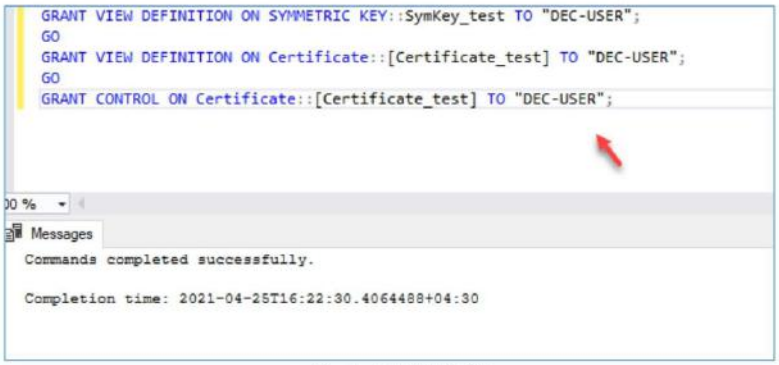


شکل ۴۶-۴ تست دسترسی کاربر

برای حل این مشکل باید به کاربر مورد نظر دسترسی لازم را بدهیم تا بتواند هم به کلید متقارن و هم به گواهینامه مورد نظر دسترسی داشته باشد.

```
GRANT VIEW DEFINITION ON SYMMETRIC KEY::SymKey_test TO "DEC-USER";
GO
GRANT VIEW DEFINITION ON Certificate::[Certificate_test] TO "DEC-USER";
GO
GRANT CONTROL ON Certificate::[Certificate_test] TO "DEC-USER";
```

نتیجه‌ی دستورات را در شکل ٤٧-٤ مشاهده می‌کنید.



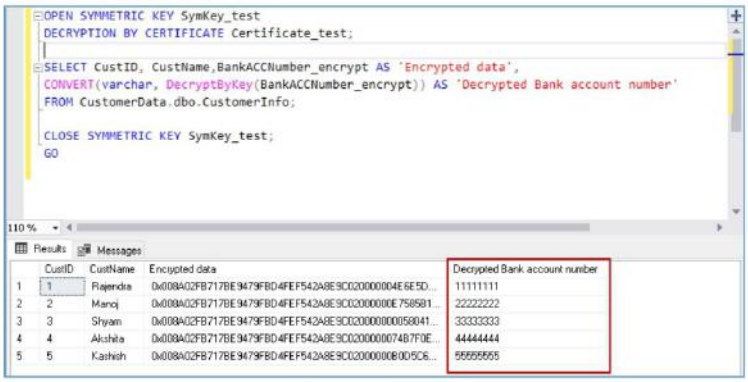
شکل ۴۷-۴ افزایش دسترسی کاربر

بعد از دادن دسترسی‌های لازم دوباره دستورات بالا را در این قسمت وارد و اجرا می‌کنیم:

```
OPEN SYMMETRIC KEY SymKey_test
DECRYPTION BY CERTIFICATE Certificate_test;
```

```
SELECT CustID, CustName,BankACCNumber_encrypt AS 'Encrypted data',
CONVERT(varchar, DecryptByKey(BankACCNumber_encrypt)) AS 'Decrypted Bank account
number'
FROM CustomerData.dbo.CustomerInfo;
```

همان‌طور که در شکل ٤٨-٤ مشاهده می‌کنید دستور مورد نظر با موفقیت اجرا شده و خروجی به نمایش گذاشته شده است.



شکل ۴۸-۴ تست دسترسی کاربر

فصل پنجم

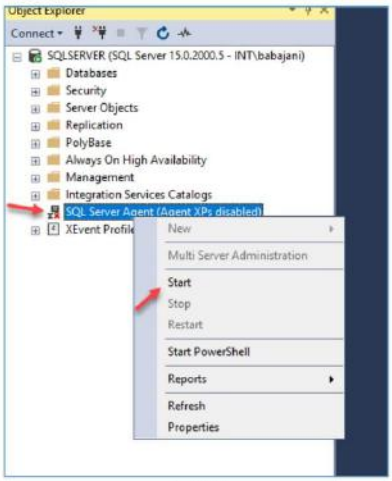
پشتیبان‌گیری و بازیابی

می‌توان یکی از مهم‌ترین بخش‌های یک سیستم نرم‌افزاری را بخش پشتیبانی و بازیابی آن بیان کرد. در سازمان‌های بزرگ مانند بانک‌ها در یک ثانیه چندین هزار رکورد در دیتابیس و جداول ثبت می‌شود و اگر چنانچه از دیتابیس مورد نظر پشتیبان نداشته باشید با ازدست‌رفتن اطلاعات اصلی دیگر نمی‌توانید به آن اطلاعات دسترسی داشته باشید پس باید مسیر را دقیقاً مشخص کرد تا بتوانیم در سریع‌ترین زمان ممکن اطلاعات را برگردانیم.

روش‌های پشتیبان‌گیری از دیتابیس بسیار زیاد است که می‌توانید از خود نرم‌افزار SQL استفاده کنید و یا اینکه از نرم‌افزارهای جانبی استفاده کنید که در اینجا همه‌ی آنها را بررسی می‌کنیم.

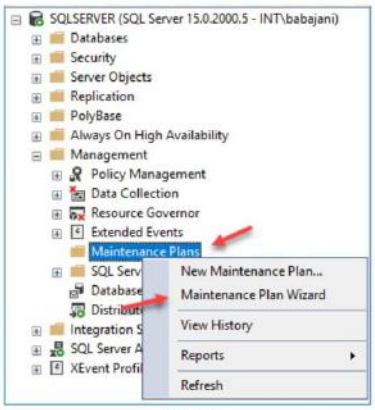
۵-۱ پشتیبان‌گیری از طریق نرم‌افزار SQL

برای شروع کار باید سرویس SQL Server Agent را فعال کنیم، برای این کار باید وارد SQL شوید و به‌مانند شکل ۵-۱ بر روی سرویس Agent کلیک راست کنید و گزینه‌ی Start را انتخاب کنید، بعد از آن پنجره‌ای باز خواهد شد که باید بر روی Yes کلیک کنید.



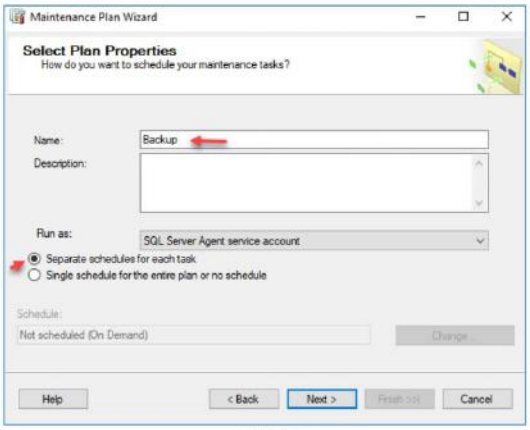
شکل ۱-۵

در ادامه می‌خواهیم ابزارهای Database Maintenance را با هم بررسی کنیم، برای اینکه سرویس Backup را فعال کنیم باید به‌مانند شکل ۲-۵ وارد قسمت Management شوید و بر روی Maintenace Plans کلیک راست کنید و گزینه‌ی Maintenace Plan Wizard را انتخاب کنید.



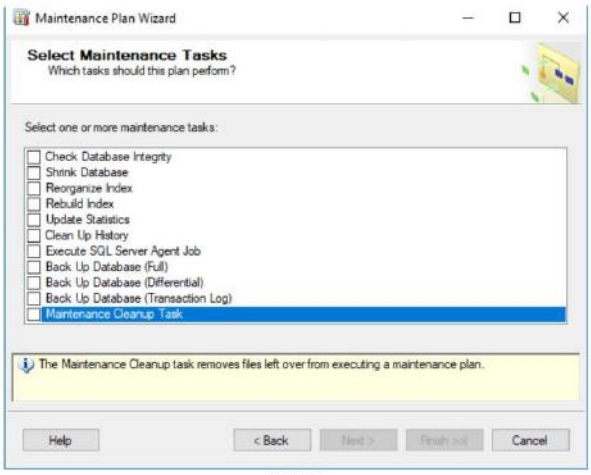
شکل ۲-۵

در شکل ۳-۵ باید یک نام به‌دلخواه وارد کنید و در قسمت زیری آن برای زمان‌بندی فعلاً گزینه‌ی Separate را انتخاب کنید.



شکل ۳-۵

در شکل ٤-۵ گزینه‌های مختلفی را مشاهده می‌کنید که هرکدام برای یک کار طراحی شده‌اند که در زیر آنها را بررسی می‌کنیم.



شکل ۴-۵

۱- Chech Database Integrity

این گزینه برای این است که بفهمیم یک دیتابیس مشکلی ندارد و می‌توانیم از آن پشتیبان تهیه کنیم که این موضوع واقعاً برای پشتیبان‌گیری مهم است اگر چنانچه دیتابیس مورد نظر مشکل داشته باشد حتی بعد از انجام پشتیبان‌گیری نمی‌توانید از آن دیتابیس در آینده استفاده کنید، پس حتماً باید از این گزینه استفاده کنید.

۲- Shrink Database

زمانی که یک دیتابیس ایجاد می‌شود دو فایل در محل ذخیره‌سازی ایجاد می‌شود که یکی فایل اصلی و دیگری فایل log است، این فایل Log بسته به نوع کار در طی زمان حجم آن افزایش پیدا خواهد کرد که باید توسط این گزینه حجم آن را کاهش دهید تا حجم اضافه در خروجی پشتیبان‌گیری ایجاد نشود.

۳- Reorganize index و Rebuld index

زمانی که یا دستورت مختلف بر روی دیتابیس خودکار می‌کنید استفاده از این دستورات مانند Insert, Delete و... به‌مرورزمان باعث ایجاد Fragmentation یا همان پارگی می‌شوند و همین امر باعث می‌شود که اطلاعات به‌درستی در هارددیسک یا محل ذخیره‌سازی قرار نگیرند به طور ساده‌تر باید گفت که مثلاً از یک محل ذخیره‌سازی با حجم ۱ گیگابیت داشته باشید، زمانی که Fragmentation یا پارگی ایجاد شود یک دیتابیس ۵۰۰ مگابایت در یک این فضای یک گیگابیت ذخیره خواهد شد و استفاده درستی از آن نخواهد شد و مهم‌ترین مشکلی که پارگی ایجاد می‌کند این است که به شدت سرعت دسترسی و استفاده از دیتابیس را کاهش می‌دهد.

۴- Update Statistics

این گزینه برای بهبود عملکرد پرس‌وجو طراحی شده است و باعث به‌روزرسانی اطلاعات برای انجام پرس‌وجو خواهد شد.

۵- Clean Up History

زمانی که دیتابیس‌های مختلفی را در طول زمان ایجاد می‌کنید یک سری داده در جدول سیستم ذخیره می‌کند و این داده‌ها به‌مرورزمان منسوخ و ازکارافتاده خواهند شد و باید با دستوری آن اطلاعات قدیمی را از دیتابیس پاک کرد، البته اگر این داده‌ها را حذف نکنید به‌مرورزمان فضای ذخیره‌سازی را پر خواهند کرد.

۶- execute sql server agent job

این دستور برای اجرای کارهایی است که در agent ایجاد کردید.

۷- Backup Database (FULL)

برای اینکه به طور کامل از دیتابیس‌های خود پشتیبان تهیه کنیم، باید از این گزینه استفاده کنیم، داشتن حداقل یک پشتیبان Full برای هر یک از دیتابیس‌ها ضروری است و اگر Full وجود نداشته باشد شما توانایی برگرداندن دیتابیس خراب شده را نخواهید داشت.

۸- Backup Database (Differential)

این روش یک روش برای پشتیبان‌گیری از تغییرات است، مثلاً اگر یک پشتیبان Full تهیه کرده باشد با حجم ۱۰ گیگابایت در موقعی که از پشتیبان Differential استفاده کنید حجم آن بسیار کم خواهد شد چون فقط از تغییراتی که بعد از Full ایجاد شده پشتیبان تهیه می‌شود، البته اگر تنها از روش Differential استفاده کنید در اولین باری که از دیتابیس پشتیبان تهیه می‌کند از تمام اطلاعات پشتیبان می‌گیرد.

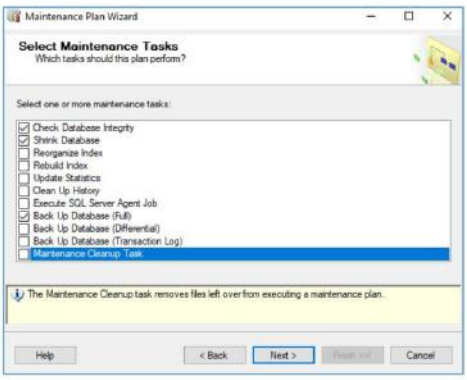
۹- Backup Database (Transaction Log)

زمانی که یک دیتابیس ایجاد می‌کنید، باید حالت Recovery Model آن را مشخص کنید، اگر بر روی حالت Full قرار داشته باشد از دیتابیس مورد نظر یک فایل Log هم گرفته می‌شود که قبلاً این موضوع را توضیح دادیم، اگر از این روش پشتیبان‌گیری استفاده کنید از داده‌های log هم پشتیبان تهیه خواهد شد و یک مزیت آن این است که بعد از انجام این نوع پشتیبان‌گیری حجم فایل log دیتابیس کاهش پیدا خواهد کرد و دیگر نیاز نیست خودتان آن را کم کنید.

۱۰- Maintenance Cleanup Task

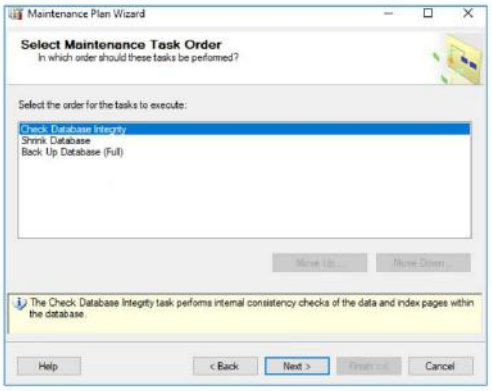
این گزینه برای حذف پرونده‌های مربوط به برنامه‌های نگهداری مانند پرونده‌های دیتابیس و... کاربرد دارد.

بعد از بررسی گزینه‌های مورد نظر برای تست کار به‌مانند شکل ۵-۵ سه گزینه‌ی مورد نظر را انتخاب کنید و بر روی Next کلیک کنید.



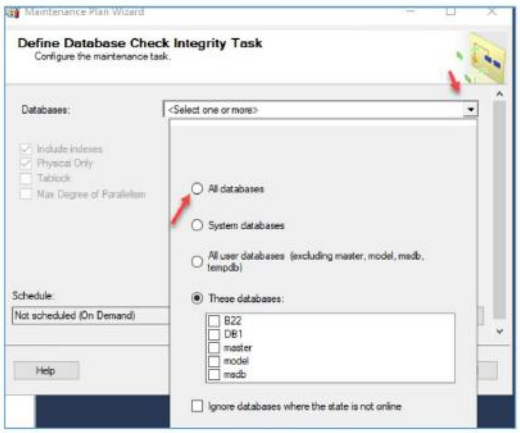
شکل ۵-۵

همان‌طور که در شکل ٦-٥ مشاهده می‌کنید سه گزینه‌ی مورد نظر انتخاب شده است و برای ادامه بر روی Next کلیک کنید.



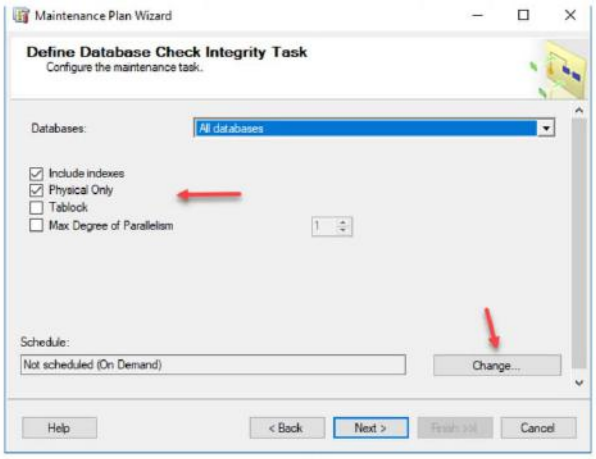
شکل ۶-۵

به‌مانند شکل ۷-۵ باید مشخص کنید که چه دیتابیسی باید انتخاب شود تا عملیات Check Integrity بر روی آن اعمال شود که می‌توانید با انتخاب گزینه‌ی All databases همه‌ی آنها را انتخاب کنید و یا System Database و یا خودتان انتخاب کنید که بهتر است گزینه‌ی All databases را انتخاب کنید.



شکل ۷-۵

بعد از انتخاب All database چند گزینه را در شکل ۸-۵ مشاهده می‌کنید که در ادامه آنها را معرفی می‌کنیم.



شکل ۸-۵

Include Indexes

این گزینه برای بررسی درستی و یکپارچگی ایندکس‌های جداول در دیتابیس است که انجام می‌شود.

Physical Only

این گزینه بعد از اجرا فقط ساختار Pageها و سرصفحه‌های رکوردهای موجود را بررسی می‌کند.

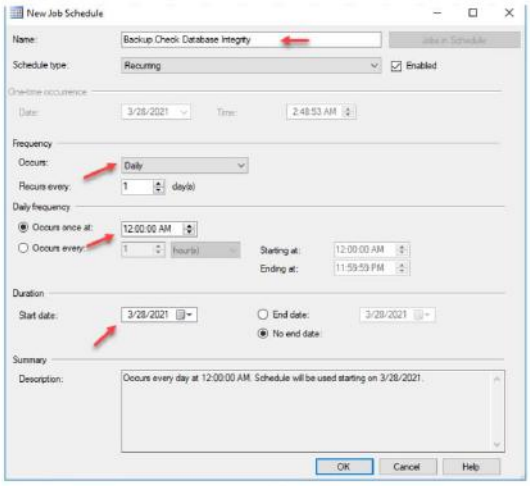
Tablock

با فعال‌کردن این گزینه دیتابیس مورد نظر برای مدت‌زمان کوتاهی قفل شده و دستور بررسی سلامت دیتابیس یعنی DBCC CHECKDB فعال می‌شود و بعد از اجرا دستور دیتابیس مورد نظر از قفل باز می‌شود، یکی از ویژگی‌های این روش بعد از اجرا باعث افزایش سرعت بررسی دیتابیس خواهد شد.

Max Degree Of Parallelism

زمانی که از دستور بررسی سلامت دیتابیس یعنی DBCC CHECKDB استفاده می‌کنید SQL از تمام توان CPU استفاده خواهد کرد، با این گزینه می‌توانید مشخص کنید که در زمان اجرا از چند هسته‌ی CPU استفاده شود.

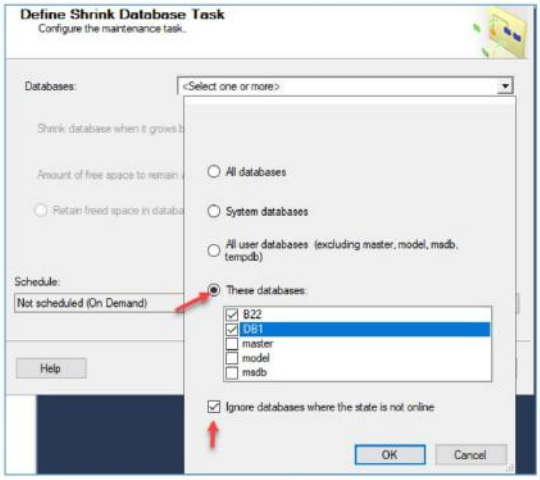
اگر به پایین شکل ۹-۵ توجه کنید، شما می‌توانید برای این وظیفه یک زمان‌بندی مشخص ایجاد کنید، برای این کار بر روی Change کلیک کنید.



شکل ۹-۵

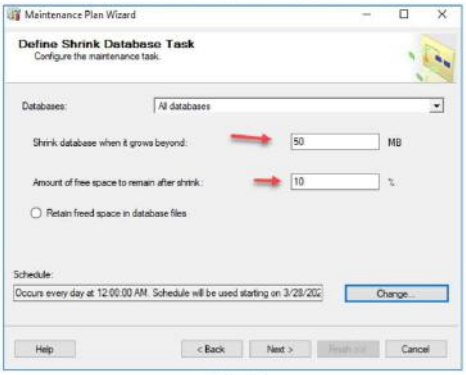
در شکل ۹-۵ باید یک نام برای زمان‌بندی خود وارد کنید و بعد باید مشخص کنید که این وظیفه در چه زمان‌هایی اجرا شود مثلاً می‌توانید هفتگی انتخاب کنید یا به‌صورت روزانه که در اینجا روزانه را انتخاب می‌کنیم، در قسمت سوم می‌توانیم ساعت اجرا این وظیفه را مشخص کنیم که ساعت ۱۲ بامداد انتخاب شده است و در قسمت آخر می‌توانید مشخص کنید که این زمان‌بندی از چه تاریخی اجرا شود که به‌صورت پیش‌فرض تاریخ امروز انتخاب خواهد شد، بر روی OK کلیک کنید.

در شکل ۱۰-۵ باید مشخص کنید که چه دیتابیس‌هایی را می‌خواهید بر روی آن عملیات shrink انجام دهید که دو دیتابیس که با هم ایجاد کردیم را انتخاب می‌کنیم، اگر در پایین صفحه تک گزینه‌ی ignore.. را انتخاب کنید در زمان Shrink از دیتابیس‌هایی که آفلاین هستند صرف‌نظر خواهد شد.



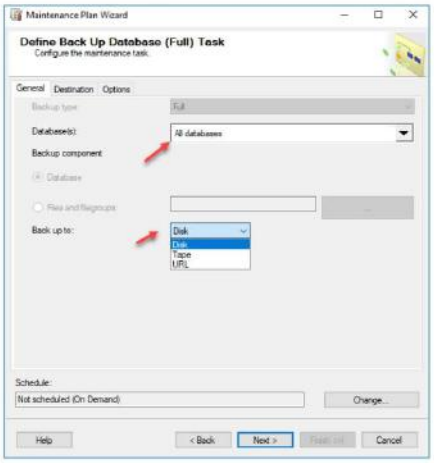
شکل ۱۰-۵

بعد از انتخاب دیتابیس به‌مانند شکل ۱۱-۵ دو گزینه وجود دارد، گزینه‌ی اول عدد ۵۰ را نشان می‌دهد و این نشان‌دهنده‌ی این است که روی دیتابیس‌هایی که بیشتر از ۵۰ مگابایت حجم داشته باشند عملیات Shrink انجام خواهد شد و اگر کمتر از این مقدار باشد عملیات Shrink انجام نخواهد شد، در قسمت بعدی عدد ۱۰ درصد را مشاهده می‌کنید که نشان‌دهنده‌ی این است که چند درصد از فضای دیتابیس بعد از Shrink خالی بماند، گزینه‌ی Retain هم اگر انتخاب شود فقط صفحه‌های انتهایی را به ابتدای فایل انتقال خواهد داد و با صفحات میانی کاری ندارد، در آخر صفحه هم می‌توانید زمان‌بندی اجرا را به‌مانند قبل مشخص کنید.



شکل ۱۱-۵

در شکل زیر نحوه انجام Full backups را باید مشخص کنید، برای این کار باید دیتابیس مورد نظر خود را انتخاب کنید که بهترین حالت این است که از گزینه‌ی All Databases استفاده کنید تا همه‌ی دیتابیس‌ها اعم از سیستمی و کاربر را تخصیص دهد. در پایین شکل ۱۲-۵ نوع ذخیره‌سازی فایل را باید مشخص کنید که دارای سه حالت است که در زیر بررسی می‌کنیم.



شکل ۱۲-۵

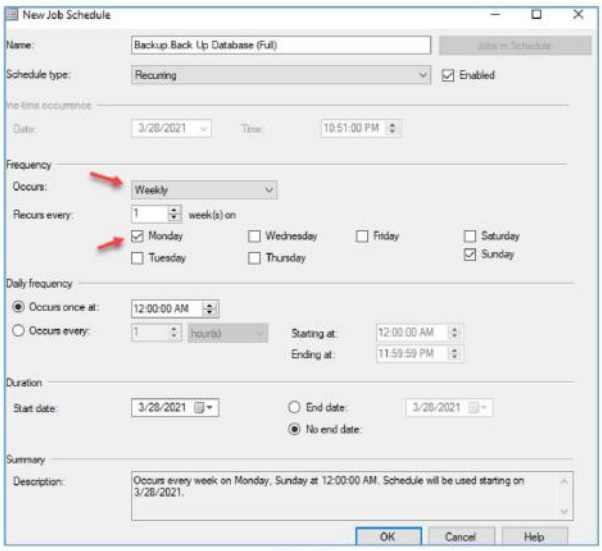
گزینه‌ی Disk برای ذخیره کردن فایل‌های پشتیبان در هارددیسک سیستم خود و یا در یک ادرس شبکه است که در ادامه بررسی می‌کنیم.

گزینه‌ی tape برای ذخیره اطلاعات بر روی نوار مغناطیسی می‌شد که یک روش بسیار مقرون‌به‌صرفه است، هرچند که بسیار قدیمی شده ولی به علت ظرفیت بالای آن و قیمت پایین آن حتی در شرکت‌های بزرگ هم در حال استفاده است.

گزینه‌ی URL برای ذخیره کردن اطلاعات در یک آدرس تحت وب طراحی شده است مانند آدرس زیر:

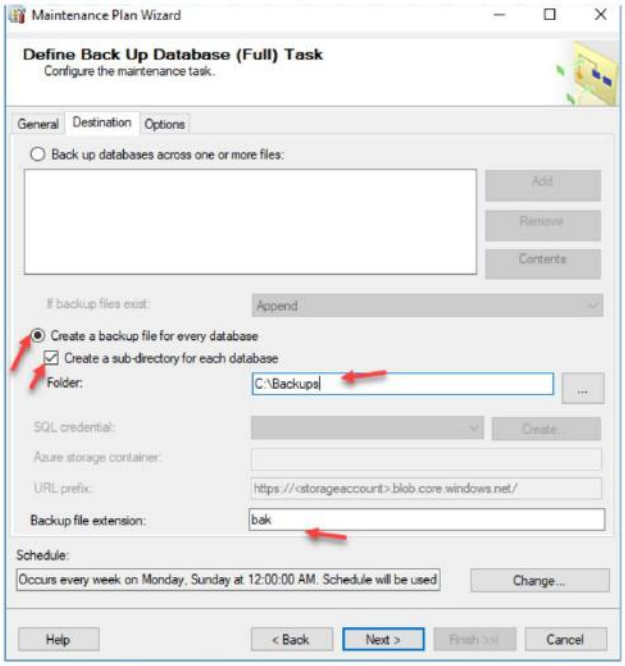
```
'https://mystorage.blob.core.windows.net/mycontainer/TestDbBackupSetNumber2_0.bak'
```

در ادامه کار بعد از انتخاب گزینه‌ی Disk در پایین صفحه می‌توانید زمان‌بندی مورد نظر خود را مشخص کنید، در شکل ۱۳-۵ باید زمان‌بندی انجام Full Backup را مشخص کنید، برای انجام Full Backup بهتر است گزینه‌ی هفتگی را انتخاب کنید دلیل آن هم این است که اگر تعداد زیادی دیتابیس با حجم بالا داشته باشید انجام Full Backup می‌تواند حجم زیادی را از حافظه‌ی شما پر کند و بهترین کار این است که به‌صورت هفتگی این کار را انجام دهید و به Differential Backup را به‌صورت روزانه انجام دهید.



شکل ۱۳-۵

در ادامه کار به‌مانند شکل ۱۴-۵ وارد تب Destination شوید و در این تب باید مشخص کنید که فایل‌های Backup در کدام مسیر ذخیره شود که در اینجا مسیر C:\Backups انتخاب شده است توجه داشته باشید برای اینکه نرم‌افزار دسترسی ایجاد SubFolder را داشته باشید بهتر است تیک گزینه‌ی Create a sub-directory را انتخاب کنید، در پایین صفحه می‌توانید پسوند فایل Backup خود را به‌دلخواه وارد کنید که به‌صورت پیش‌فرض پسوند bak وارد شده است.

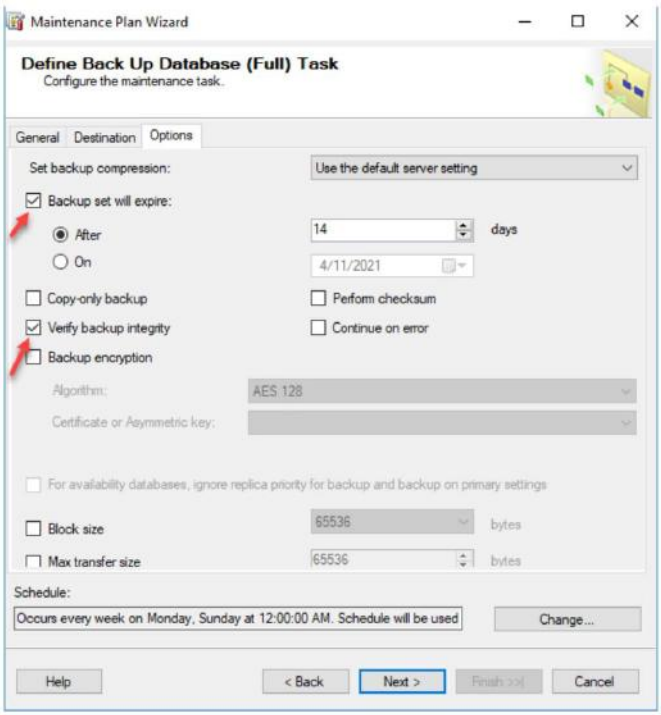


شکل ۱۴-۵

در ادامه به‌مانند شکل ۱۵-۵ وارد تب Options شوید، در این تب می‌توانید با انتخاب گزینه‌ی Backup set will Expore مشخص کنید که فایل‌های Backup بعد از مدت مشخص‌شده‌ی ۱٤ روز ازروی هارد پاک شوند و فایل Backup جدید جایگزین آن شود.

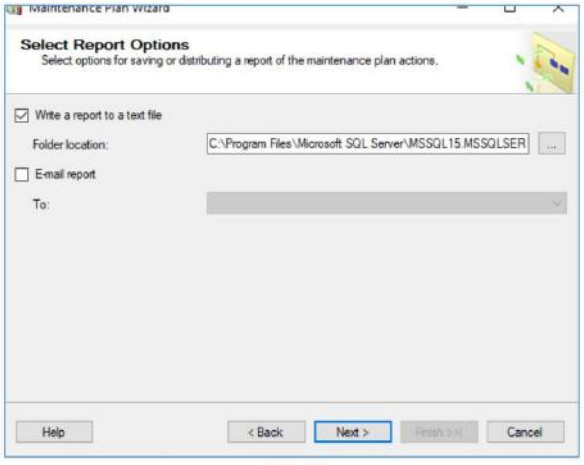
یکی از مهم‌ترین قسمت‌هایی که باید مراقب آن باشد این است که تیک گزینه‌ی Verify Backup Integirity را انتخاب کنید، این گزینه بعد از انجام Backup فایل Backup را تست می‌کند تا سلام باشد و بعد از آن پردازش به پایان می‌رسد، در جاهایی دیده شده که فایل‌های Backup قابل بازگردانی نبودن و مدیر شبکه با مشکل بزرگی روبرو شده است، پس حتماً تیک این گزینه را بزنید، البته اگر حجم Backup زیاد باشد مقدار زمان انجام عملیات بیشتر خواهد شد.

گزینه‌ی Backup Encryption را اگر انتخاب کنید بر روی فایل‌های Backup یک رمز عبور به همراه هشینگ قدرتمند قرار می‌دهد تاکسی نتواند اطلاعات را به سرقت ببرد.



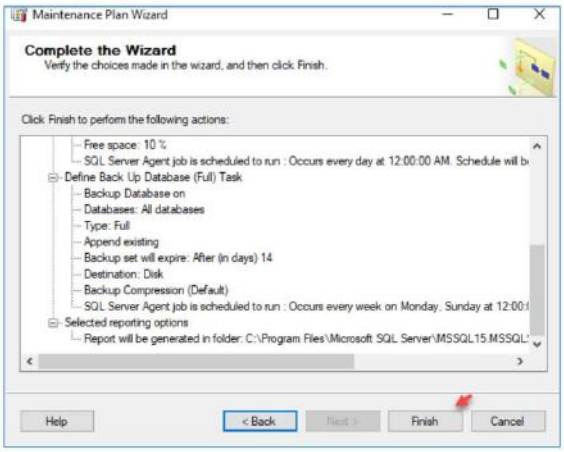
شکل ۵-۱۵

در ادامه بر روی Next کلیک کنید، در شکل ۵-۱۶ می‌توانید مشخص کنید که گزارشگیری از اطلاعات این کار در چه مسیری ذخیره شود، البته این فایل به‌صورت TXT است و اگر هم بخواهید به یک آدرس خاص ایمیل شود باید تیک گزینه‌ی E-mail report را انتخاب کنید.



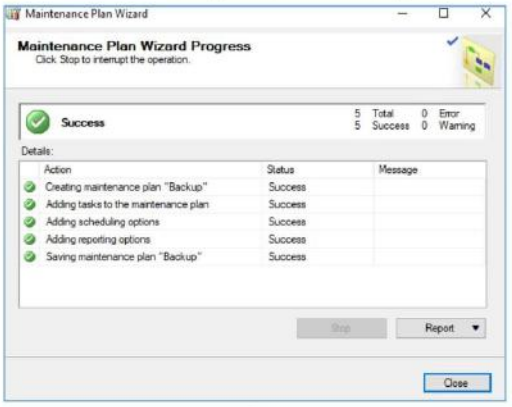
شکل ۵-۱۶

اگر اطلاعاتی در شکل ۵-۱۷ مورد قبول بود می‌توانید بر روی Finish کلیک کنید.



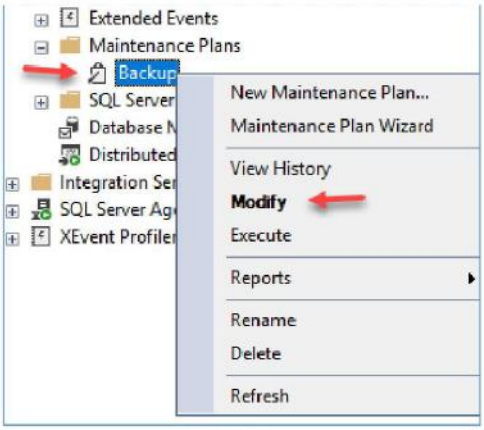
شکل ۵-۱۷

همان‌طور که در شکل ۵-۱۸ مشاهده می‌کنید اطلاعات به‌درستی تأیید و ایجاد شده است.



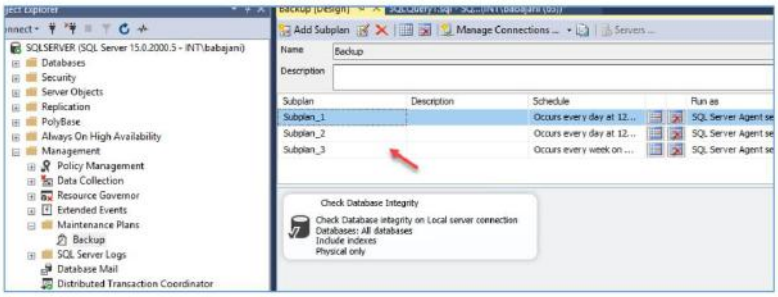
شکل ۵-۱۸

در شکل ۵-۱۹ Plan مورد نظر ایجاد شده است و برای اینکه بررسی کامل‌تری داشته باشیم بر روی آن کلیک راست کنید و گزینه‌ی Modify را انتخاب کنید.



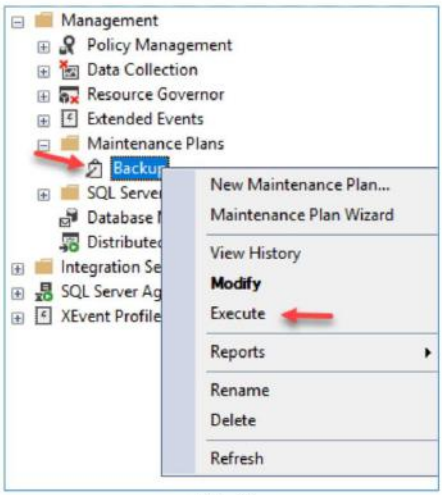
شکل ۱۹-۵

در شکل ۲۰-۵ هر سه قسمتی که در قسمت‌های قبل ایجاد کردیم را مشاهده می‌کنید اگر بر روی آنها کلیک کنید عملکرد آنها را مشاهده خواهید کرد، در جلوی آنها زمان‌بندی را می‌توانید تغییر یا حذف کنید و حتی می‌توانید نام هر یک از Ruleها را تغییر دهید.



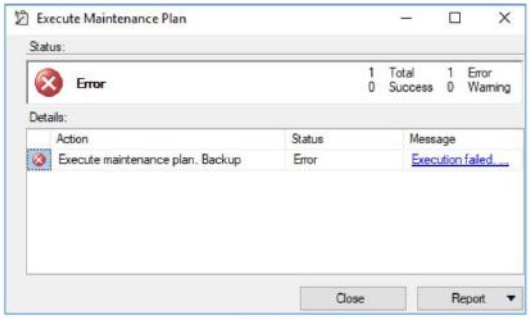
شکل ۲۰-۵

بعد از ایجاد Plan باید آن را اجرا کنیم، برای اجرا به‌مانند شکل ۲۱-۵ بر روی Plan مورد نظر کلیک راست کنید و گزینه‌ی Execute را انتخاب کنید.



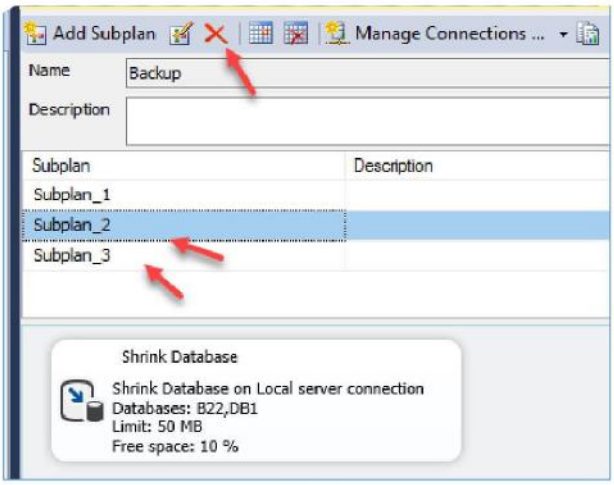
شکل ۲۱-۵

بعد از اجرا به‌احتمال زیاد با خطای شکل ۲۲-۵ مواجه خواهید شد که آن هم به این خاطر است که ما برای هر یک از قسمت‌ها یک زمان‌بندی تعریف کردیم و به‌خاطر همین این سه قسمت از هم جداشدن، برای حل این مشکل باید آنها را در یک قسمت قرار دهیم.



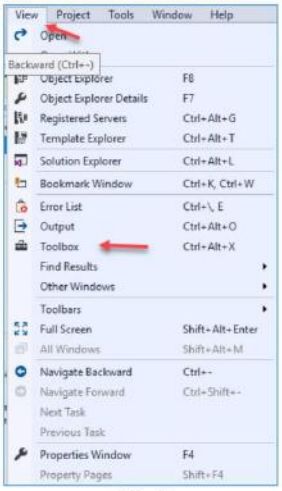
شکل ۲۲-۵

برای حل مشکل به‌مانند شکل ۲۳-۵ وارد Modify شوید و Subplan_2 و Subplan_3 را از لیست حذف کنید.



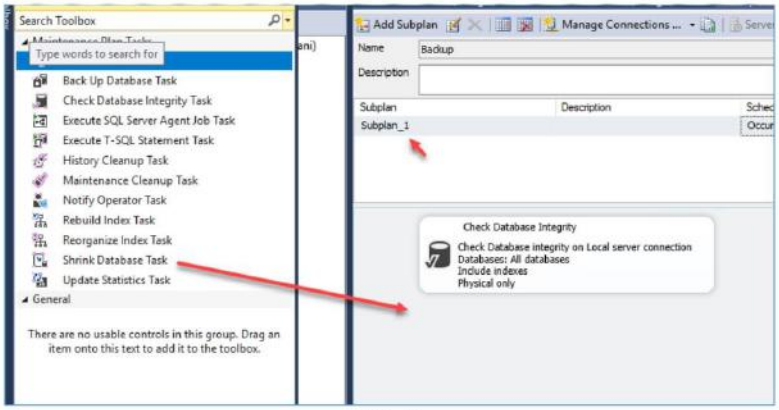
شکل ۲۳-۵

در ادامه به‌مانند شکل ۲۴-۵ وارد منوی View شوید و بر روی Toolbox کلیک کنید، البته از سمت چپ هم می‌توانستید ToolBox را اجرا کنید.



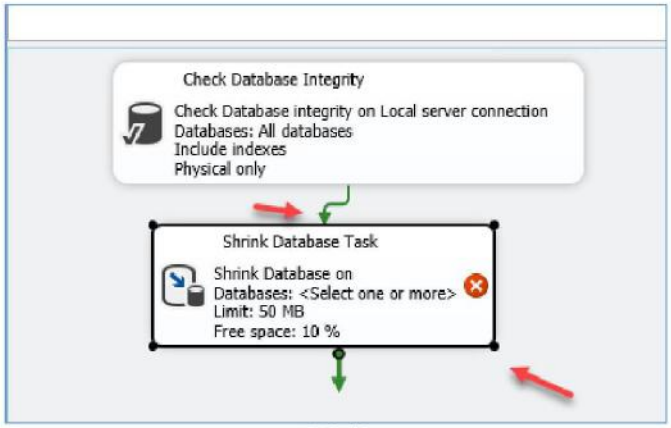
شکل ۲۴-۵

به‌مانند شکل ۲۵-۵ در subplan_1 باید گزینه‌ی Shrink را کشیده و در محل مورد نظر رها کنید.



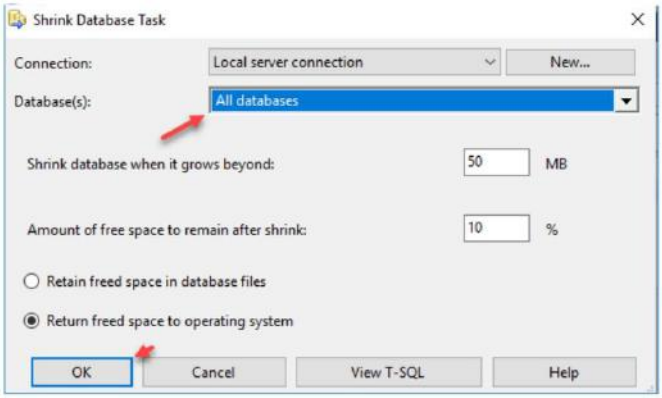
شکل ۵-۲۵

بعد از اضافه‌کردن Shrink باید فلش بالایی را به آن متصل کنید و بعد برای تنظیم آن دو بار برروی Dhrink کلیک کنید.



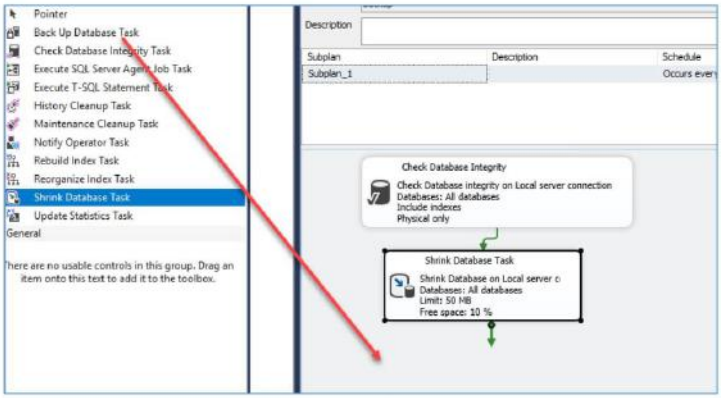
شکل ۵-۲۶

در شکل ۵-۲۷ باید All databases را انتخاب کنید و بر روی OK کلیک کنید.



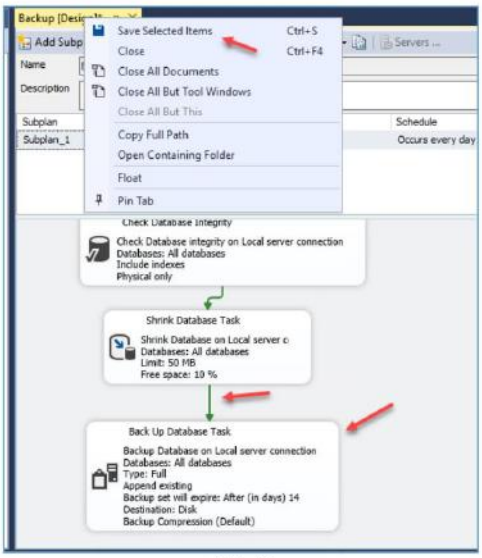
شکل ۵-۲۷

در ادامه باید گزینه‌ی Back UP Database Task را هم به لیست اضافه کنیم که این کار را باید به‌مانند شکل ۵-۲۸ انجام دهید.



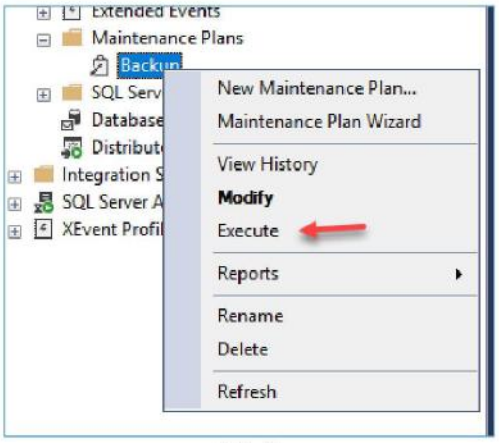
شکل ۵-۲۸

به‌مانند شکل ۵-۲۹ گزینه‌ی آخر را هم متصل کنید و با دو بار کلیک بر روی آن تنظیمات آن را هم به نسبت قبل انجام دهید و اطلاعات را ذخیره کنید.

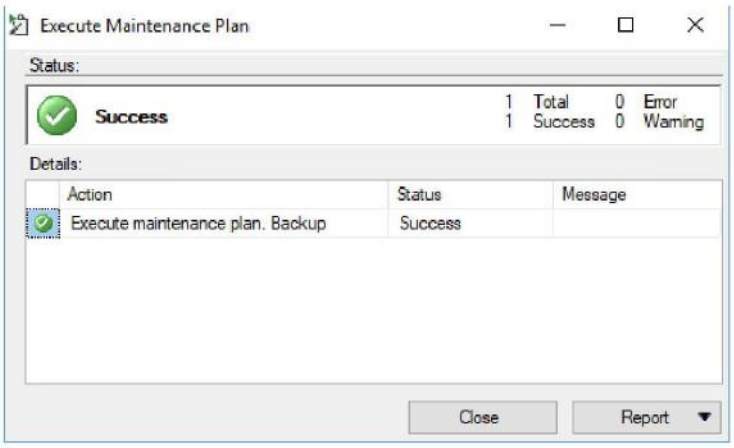


شکل ۲۹-۵

بعد از اجرای مراحل بالا اگر به‌مانند شکل ۳۰-۵ بر روی Execute کلیک کنید عملیات به‌مانند شکل ۳۱-۵ با موفقیت انجام شود.

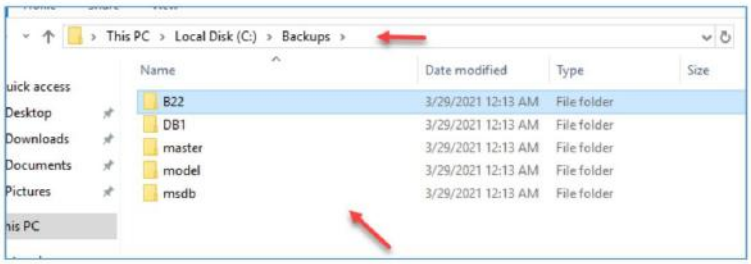


شکل ۳۰-۵



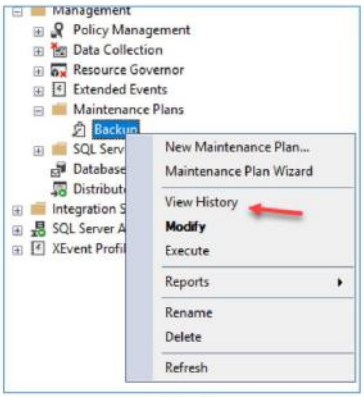
شکل ۳۱-۵

به‌مانند شکل ۳۲- ۵ وارد آدرس ذخیره‌سازی Backup شوید و مشاهده خواهید کرد فایل‌های Backup به‌درستی ایجاد شده‌اند.



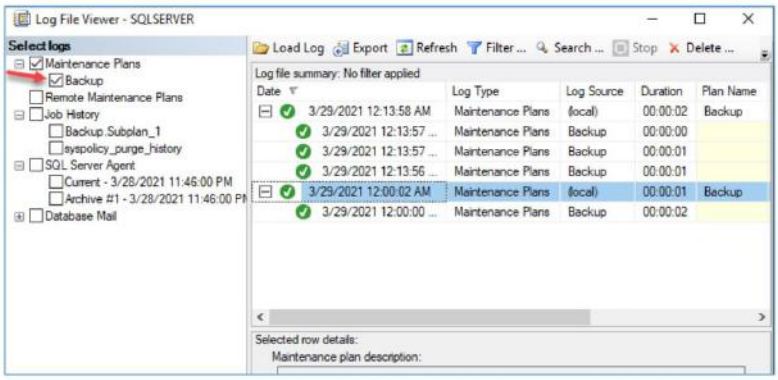
شکل ۳۲-۵

برای اینکه متوجه شوید Maintenance Plans چند بار اجرا شده و درست کار کرده یا نه باید به‌مانند شکل ۳۳-۵ باید بر روی Plan مورد نظر کلیک راست و گزینه‌ی View History را انتخاب کنید.



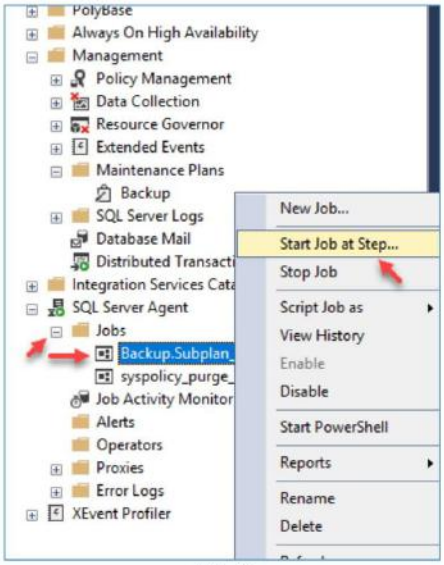
شکل ۳۳-۵

در شکل ٣٤-٥ نتیجه کار را مشاهده می‌کنید، توجه داشته باشید برای اینکه فقط Plan مورد نظر شما نمایش داده شود باید تیک آن را انتخاب کنید.



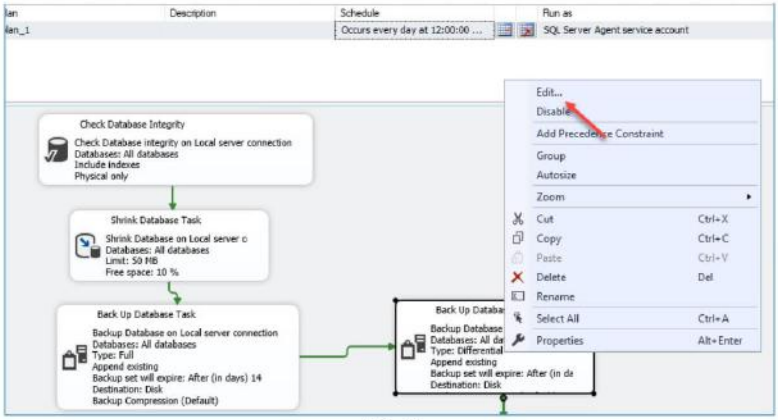
شکل ٣٤-٥

همان‌طور که گفتیم عملیات Maintenance Plans توسط سرویس Agent انجام می‌شود اگر به‌مانند شکل ٣٥- ٥ وارد قسمت SQL Server agent شوید می‌توانید Plan مورد نظر را مشاهده و آن را اجرا و تنظیمات آن را تغییر دهید.



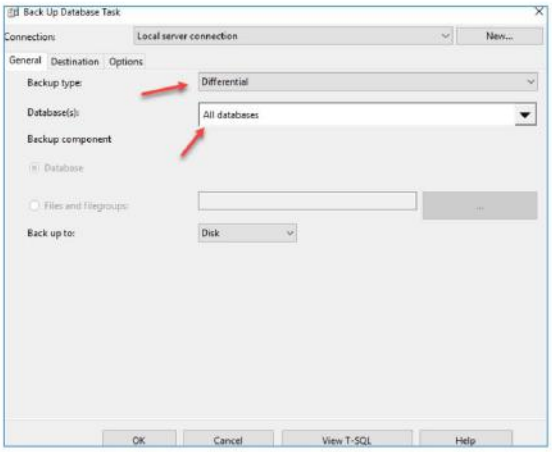
شکل ٣٥-٥

برای اینکه بتوانیم از Differential Backup استفاده کنیم باید به‌مانند شکل ٣٦-٥ یک Back UP Database به لیست اضافه کنید و بر روی آن کلیک راست کنید و گزینه‌ی Edit را انتخاب کنید، یا اینکه دوبار کلیک کنید.



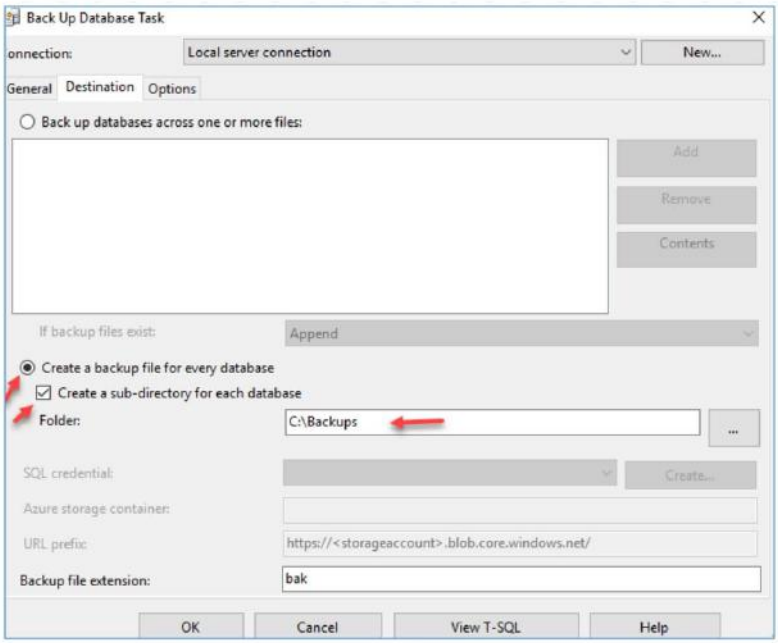
شکل ۵-۳۶

در شکل ۵-۳۷ باید در قسمت Backups Type گزینه‌ی Differential را انتخاب کنید و All Database را هم انتخاب کنید.



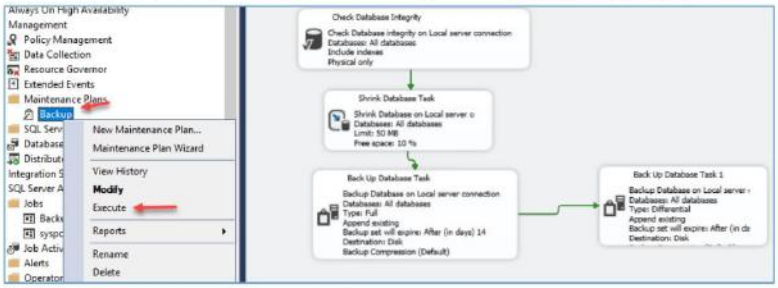
شکل ۵-۳۷

به‌مانند شکل ۵-۳۸ در تب Destination هم همان آدرس قبلی را انتخاب و تیک گزینه‌ی مورد نظر را انتخاب کنید، در تب Options هم همان گزینه‌های قبلی را انتخاب و بر روی OK کلیک کنید.



شکل ۳۸-۵

بعد از انجام مراحل بالا می‌توانید به‌مانند شکل ۳۹-۵ Plan مورد نظر خود را اجرا کنید.

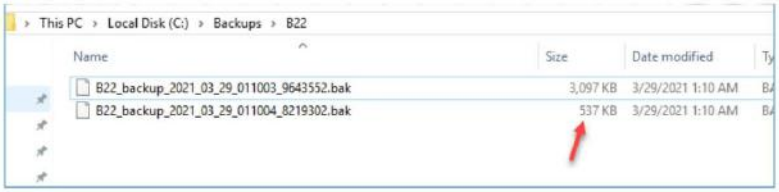


شکل ۳۹-۵

بعد از اجرا این سرویس به‌مانند شکل ۴۰-۵ در هر اجرا دو فایل پشتیبان ایجاد می‌کند که یکی Full است و دیگری Differenctail که حجم آنها مشخص‌کننده فایل مورد نظر است.

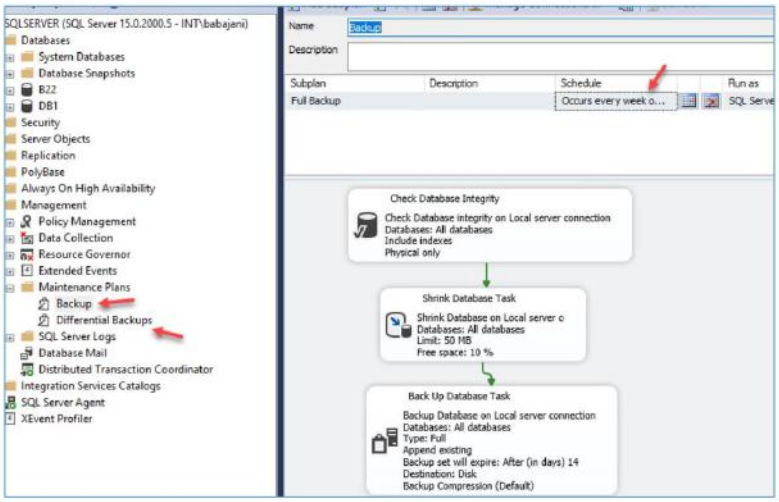
آیا به نظر شما این روش پشتیبان‌گیری درست است؟

نه این روش کاملاً اشتباه است، به‌خاطر اینکه این Plan که ایجاد کردیم هر دو فایل را در هر بار اجرا تولید می‌کند و این کار بسیار بر حجم فضای ذخیره‌سازی تاثیر گذار خواهد بود، برای حل این مشکل بهتر است هر کدام را جداگانه در یک Plan مختلف با زمانبندی مختلف قرار دهیم.



شکل ۴۰-۵

در شکل ۴۱-۵ دو Plan متفاوت ایجاد کردیم و در زمانبندی آنها برای Full Backup هفگی را مشخص کردیم و برای Differenctial روزانه را مشخص کردیم، با این کار هر هفته یک Full و هر روز یک Differenctial گرفته خواهد شد.

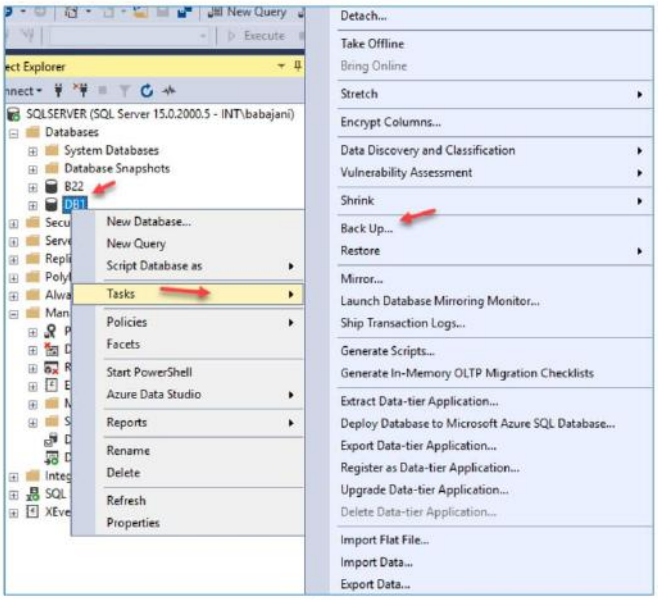


شکل ۴۱-۵

۵-۱-۱ پشتیبانگیری به‌صورت دستی در SQL

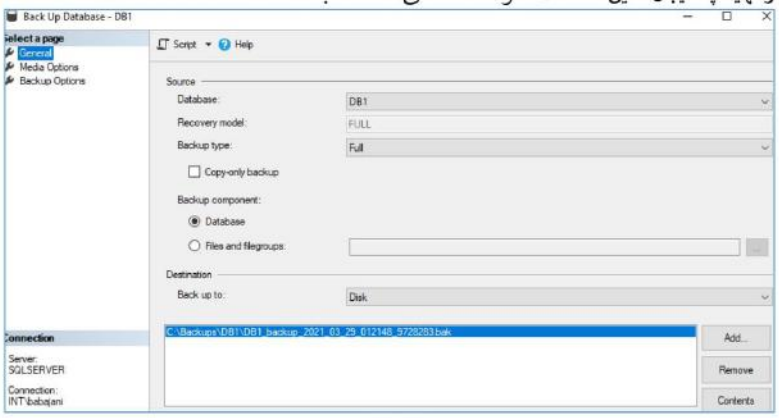
تا به اینجا توانستیم از طریق Maintenance Plans و به‌صورت اتوماتیک از دیتابیس‌ها پشتیبان تهیه کنیم، اما اگر بخواهید به‌صورت دستی و سریع از دیتابیس مورد نظر پشتیبان تهیه کنید باید به‌صورت زیر عمل کرد.

به‌مانند شکل ۴۲-۵ بر روی دیتابیس مورد نظر خود کلیک راست کنید و از قسمت Tasks گزینه‌ی BackUP را انتخاب کنید.



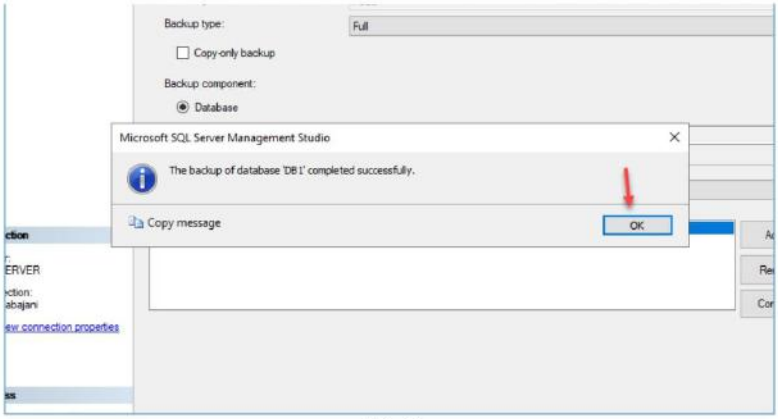
شکل ۴۲-۵

در شکل ۴۳-۵ باید نوع Backup را انتخاب کنید که گزینه‌ی Full را انتخاب می‌کنیم و در پایین صفحه باید مسیر ذخیره شدن فایل Backup را مشخص کنید که به‌صورت پیش‌فرض در مسیر نصب SQL قرار خواهد گرفت، ولی چون در قسمت قبلی از Plan استفاده کردیم مسیر را تغییر داده است، در تب‌های دیگر هم می‌توانید مشخص کنید که بعد از تهیه پشتیبان فایل‌ها تست شوند تا مشکلی نداشته باشند.



شکل ۴۳-۵

بعد از کلیک بر روی OK در شکل ۴۳-۵ عملیات شروع شده و تایید آن در شکل ۴۴-۵ مشخص می‌شود.

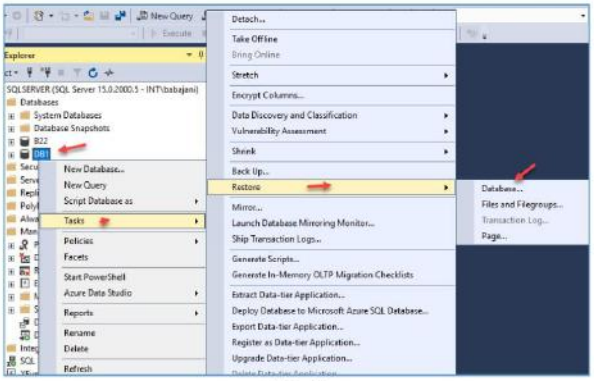


شکل ۴۴-۵

۵-۱-۲ نحوه‌ی بازگرداندن فایل‌های Backup

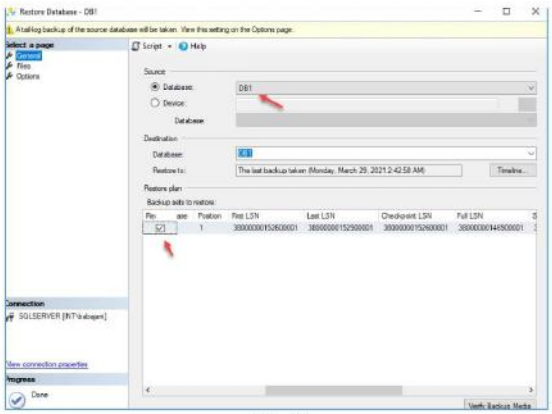
بعد از انجام پشتیبانگیری از دیتابیس‌های SQL باید توانایی این را داشته باشید در صورت خرابی هر یک از دیتابیس‌ها بتوانید آن را برگردانید.

برای اینجام دستور Restore باید به‌مانند شکل ٤٥-٥ بر روی دیتابیس مورد نظر که مشکل دارد کلیک راست کنید و از قسمت Tasks و بعد Restore گزینه‌ی Database را انتخاب کنید.



شکل ۴۵-۵

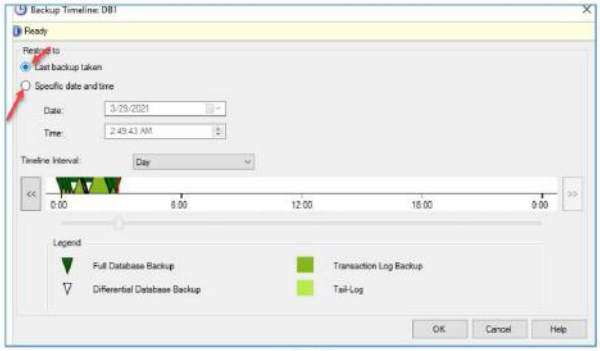
در شکل ٤٦-٥ باید نام دیتابیس به همراه پشتیبانی که از قبل تهیه کردید را انتخاب کنید.



شکل ۴۶-۵

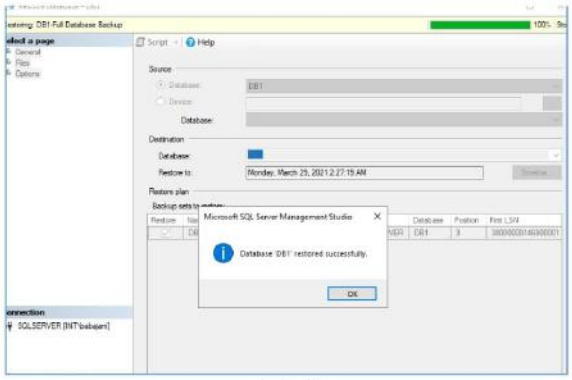
توجه داشته باشید که برای اینکه فایل پشتیبان برای تاریخ خاصی را انتخاب کنید باید در شکل ٤٦-٥ بر روی Timeline کلیک کنید.

در شکل ٤٧-٥ می‌توانید با انتخاب گزینه‌ی Last Backup taken از آخرین Backup گرفته شده استفاده کنید و یا اینکه اگر تاریخ خاصی مد نظر شما هست باید گزینه‌ی Specific date and time را انتخاب کنید و بعد تاریخ مورد نظر خود را مشخص کنید، البته در قسمت Timeline می‌توانید به‌صورت گرافیکی بر روی آیکون‌های مورد نظر کلیک کنید که آیکون سبز پر رنگ نشان‌دهنده‌ی Full Backup است.



شکل ۴۷-۵

بعد از اینکه بر روی OK کلیک کردید باید عملیات Restore انجام شود که در شکل ٤٨-٥ این موضوع را مشاهده می‌کنید.

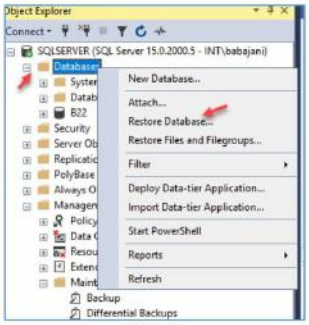


شکل ۵-۴۸

۵-۱-۳ بازگرداندن دیتابیس حذف شده

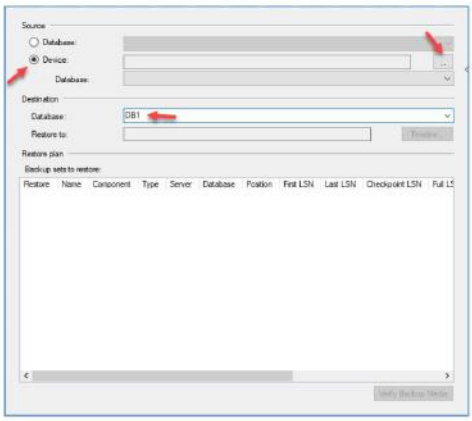
در بعضی مواقع امداً یا سهواً دیتابیس از داخل SQL حذف می‌شود و دیگر نمی‌توانیم به ان دسترسی داشته باشیم، در این زمان باید نسخه پشتیبان را بازگردانیم.

به‌مانند شکل ۵-۴۹ بر روی پوشه‌ی Databases کلیک راست کنید و گزینه‌ی Restore Databases را انتخاب کنید.



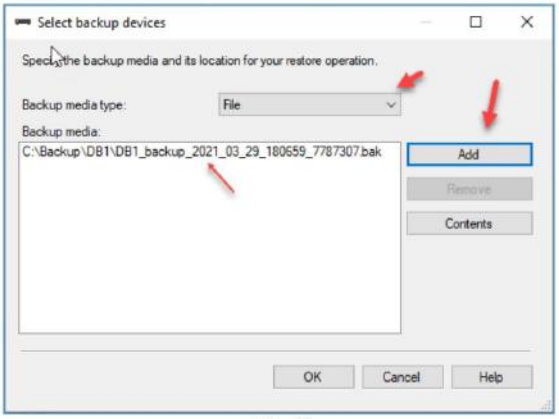
شکل ۵-۴۹

در شکل ۵-۵۰ شما باید دیتابیسی که قرار است برگردانید را از لیست انتخاب کنید که این کار به علت حذف ان دیتابیس امکان پذیر نیست به‌خاطر همین موضوع باید در قسمت Database نام دیتابیس خود را وارد کنید و در ادامه باید گزینه‌ی Device کلیک کنید و بعد بر روی Browse کلیک کنید.



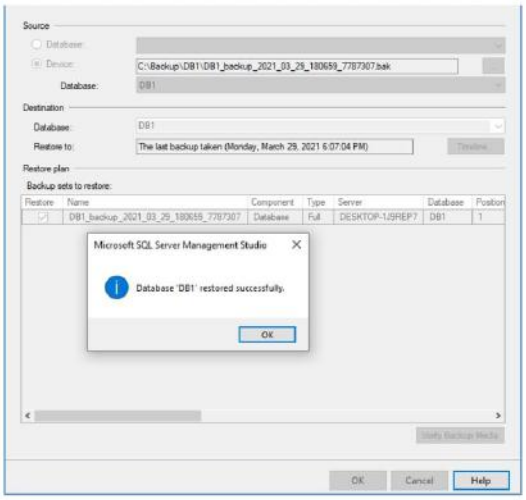
شکل ۵۰-۵

در شکل ۵۱-۵ باید بر روی Add کلیک کنید و فایل پشتیبان Full را به لیست اضافه و بر روی OK کلیک کنید.



شکل ۵۱-۵

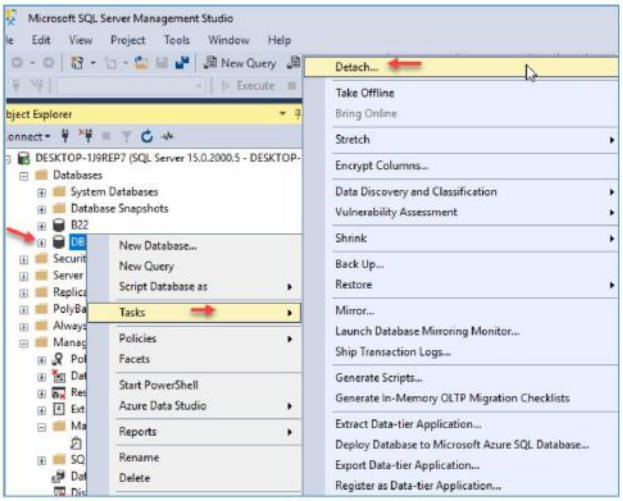
به‌مانند شکل ۵۲-۵ بعد از کلیک بر روی OK به‌درستی دیتابیس مورد نظر به لیست اضافه خواهد شد.



شکل ۵-۵۲

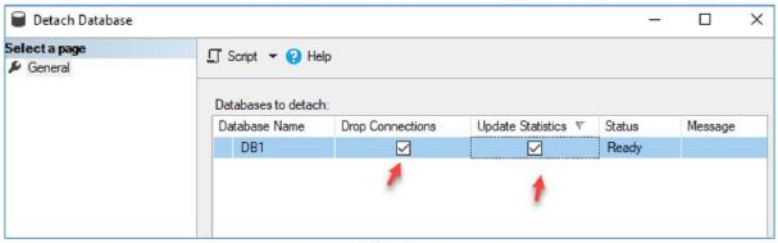
۵-۱-٤ Attach و Detach کردن دیتابیس

برای اینکه بتوانید یک دیتابیس را از لیست SQL مخفی و یا آن را اضافه کنید باید از این دستورات استفاده کنید، برای این کار به‌مانند شکل ۵-۵۳ بر روی دیتابیس مورد نظر کلیک راست کنید و از قسمت Tasks گزینه‌ی Detach را انتخاب کنید.



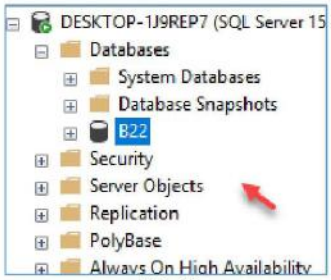
شکل ۵-۵۳

در شکل ٥٤-٥ باید دو تیک مرود نظر را انتخاب کنید و بعد بر روی OK کلیک کنید، گزینه‌ی Drop Connection ارتباط این دیتابیس را با هر نرم‌افزاری قطع می‌کند و گزینه‌ی بعدی برای آپدیت اطلاعات است.



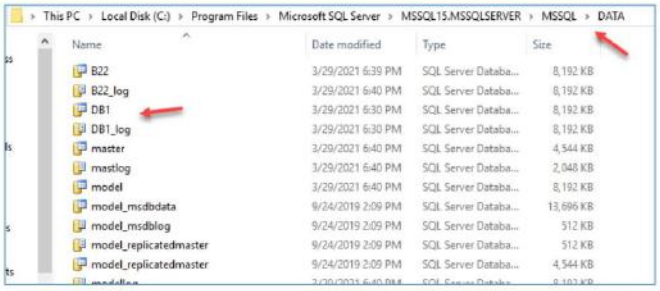
شکل ۵-۵۴

همان‌طور که در شکل ٥٥-٢ مشاهده می‌کنید دیتابیس DB1 از لیست حذف شده است این بدان معنا نیست که دیتابیس از هارد حذف شده باشد بلکه فقط از لیست حذف شده است که این موضوع را در شکل ٥٦-٥ مشاهده می‌کنید



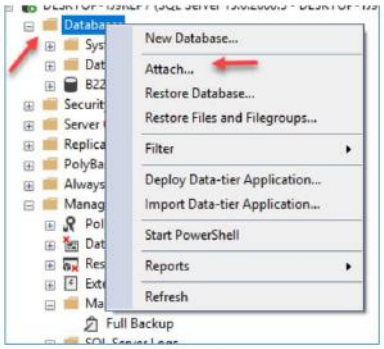
شکل ۵-۵۵

اگر به‌مانند شکل ٥٦-٥ وارد آدرس مورد نظر شوید دیتابیس DB1 را مشاهده خواهید کرد.



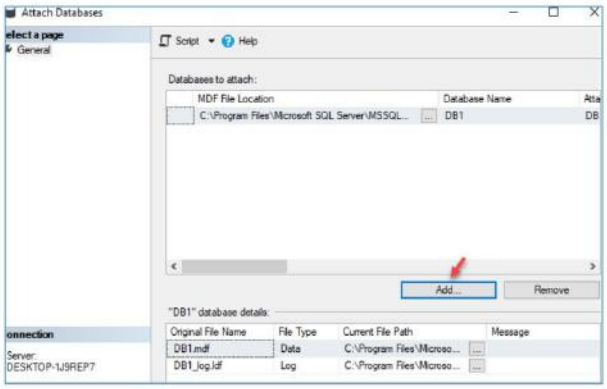
شکل ۵-۵۶

برای اینکه دوباره دیتابیس را به SQL اضافه کنیم به‌مانند شکل ٥٧-٥ بر روی پوشه‌ی Databases کلیک راست کنید و گزینه‌ی Attach را انتخاب کنید.



شکل ۵-۵۷

در شکل ۵-۵۸ بر روی Add کلیک کنید و دیتابیس DB1 را از آدرس مورد نظر که در شکل ۵-۵۶ مشخص کردیم انتخاب کنید و بر روی OK کلیک کنید.



شکل ۵-۵۸

استفاده‌های دیگر این ابزار زمانی است بخواهید سرور SQL خود را تغییر دهید و دیتابیس‌ها را از یک سرور به سرور دیگر انتقال دهید، البته راه‌های بهتر و متنوع‌تری هم برای این کار وجود دارد.

۵-۲ پشتیبان‌گیری از طریق نرم‌افزار Veeam Backup

یک نرم‌افزار قدرتمند در پشتیبان‌گیری است که حداکثر اطمینان را برای شما به ارمغان می‌آورد؛ با این نرم‌افزار فوق‌العاده می‌توانید از نرم‌افزارها، فایل‌ها، سایت‌ها و... پشتیبان تهیّه کنید و کمتر از چند ثانیه آن را برگردانید. نرم‌افزار Veeam، یکی از بهترین‌ها در بازار نرم‌افزارها است و به‌خاطر اطمینان کامل در بازگردانی اطّلاعات، رقیبی مقابل خود نمی‌بیند.

۵-۲-۱ نصب نرم‌افزار Veeam Backup and Replication

این نرم‌افزار به‌مانند دیگر نرم‌افزارها دارای نیازمندی‌هایی برای نصب است که در زیر آنها را مورد بررسی قرار می‌دهیم.

جدول ۱-۵ نیازمندی سخت‌افزاری

پردازنده	پردازنده بهتر است، حداقل دو هسته‌ای و ۲ گیگاهرتز سرعت داشته باشد.
رم	حداقل رم باید ٤ گیگابایت باشد تا عملکرد متوسّط از سرور را داشته باشیم.
کارت شبکه	بهترین حالت استفاده از کارت شبکه، گیگابایت است تا حداکثر سرعت را در انتقال فایل داشته باشیم.
هارددیسک	برای نصب و راه‌اندازی، حداقل باید ٦ گیگابایت محلّ ذخیره‌سازی داشته باشیم، البتّه برای اجرای Backup و ذخیره این فایل‌ها باید هاردهایی با ظرفیّت بالا به نسبت شبکه داشته باشیم تا کلّ اطّلاعات شبکهٔ ما را پشتیبانی کند.

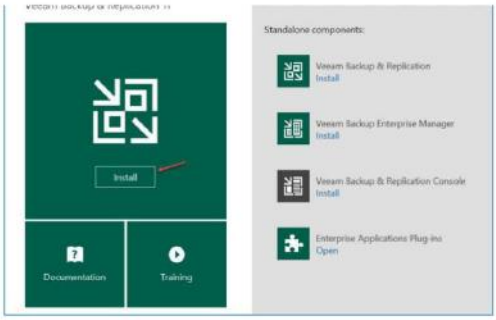
جدول ۲-۵ نیازمندی نرم‌افزاری

سیستم‌عامل	از سیستم‌عامل ویندوز که لیست آن در زیر آمده است، می‌توان استفاده کرد: Microsoft Windows Server 2019 Microsoft Windows Server 2016 Microsoft Windows Server 2012 R2 Microsoft Windows Server 2012 Microsoft Windows Server 2008 R2 SP1 Microsoft Windows Server 2008 SP2 Microsoft Windows 10 Microsoft Windows 8.x Microsoft Windows 7 SP1
نرم‌افزار	این نرم‌افزارهای اوّلیّه برای نصب Veeam موردنیاز است: Microsoft .NET Framework 4.5.2 Microsoft Windows Installer 4.5 Microsoft SQL Server Management Objects Microsoft SQL Server System CLR Types Microsoft Visual C++ 2010 Service Pack 1 redistributable package
دیتابیس	در زیر، لیستی از دیتابیس‌های مورد استفادهٔ نرم‌افزار Veeam را مشاهده می‌کنید: Microsoft SQL Server 2017 Microsoft SQL Server 2016 Microsoft SQL Server 2014 Microsoft SQL Server 2012 (Microsoft SQL Server 2012 SP3 Express Edition is included in the setup) Microsoft SQL Server 2008 R2 Microsoft SQL Server 2008

برای دانلود این نرم‌افزار می‌توانید از لینک زیر استفاده کنید:

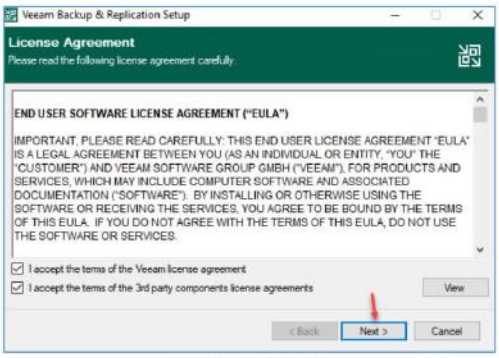
https://soft98.ir/software/backup/2098-veeam-backup.html

بعد از اجرای فایل Setup بر روی Install کلیک کنید، گزینه‌های دیگری نیز وجود دارد که گزینهٔ Enterprise برای مدیریّت چندین Backup & Replication Veeam کاربرد دارد و گزینهٔ Console نیز برای ارتباط از سیستم خودتان با سرورهای Veeam است که فعلاً کاری با آنها نداریم.



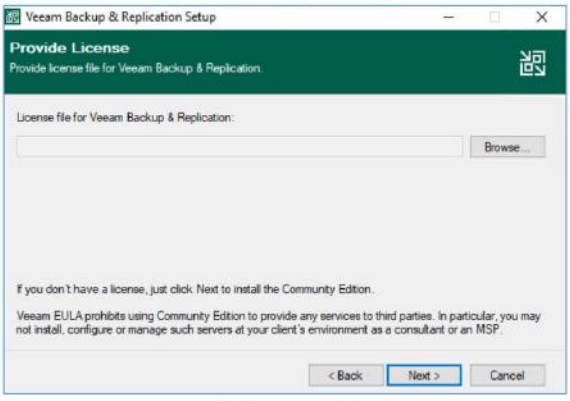
شکل ۵-۵۹ نصب Veeam

در شکل ۵-۶۰، قرار دادنامه را بررسی و تأیید کنید و بر روی Next کلیک کنید.



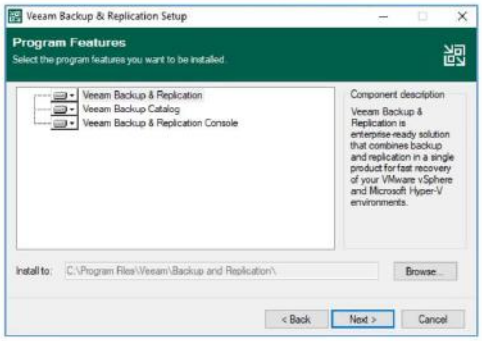
شکل ۵-۶۰ نصب Veew

در شکل ۵-۶۱، اگر لایسنس نرم‌افزار را خریداری کردید، بر روی Browse کلیک کنید و فایل لایسنس را معرّفی کنید و اگر لایسنسی در اختیار ندارید، می‌توانید از آن صرف‌نظر کنید و بر روی Next کلیک کنید که البته یک لایسنس Trial را برای شما در نظر می‌گیرد و بعد از چند روز غیرفعال خواهد شد.



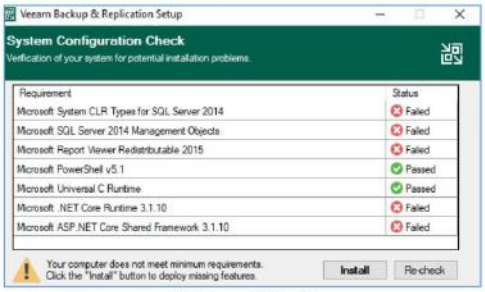
شکل ۵-۶۱ نصب Veew

در شکل ٦٢-٥ می‌توانید مسیر نصب و اجرای نرم‌افزار را تغییر دهید، حتماً مطمئن شوید که مسیر مورد نظر دارای فضای کافی برای نصب باشد، بر روی Next کلیک کنید.



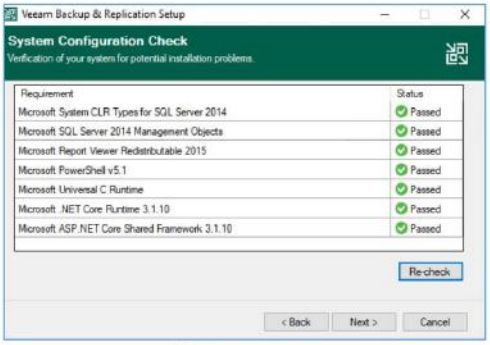
شکل ۶۲-۵ نصب Veew

در شکل ٦٣-٥، اگر پیش‌نیازهای نرم‌افزار بر روی سیستم شما نصب باشد، خطایی نمی‌دهد، امّا اگر با خطا مواجه شدید می‌توانید بر روی Install کلیک کنید تا کار نصب انجام شود.



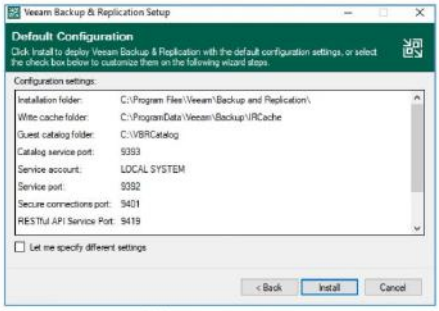
شکل ۶۳-۵ نصب Veew

همان‌طور که در شکل ٦٤-٥ مشاهده می‌کنید، پیش‌نیازهای نرم‌افزار به‌درستی نصب شده است، بر روی Next کلیک کنید.



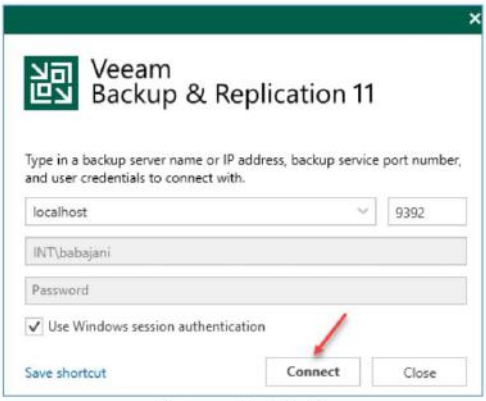
شکل ۶۴-۵ نصب Veew

در شکل ٦٥-٥ بر روی Install کلیک کنید تا کار نصب آغاز شود.



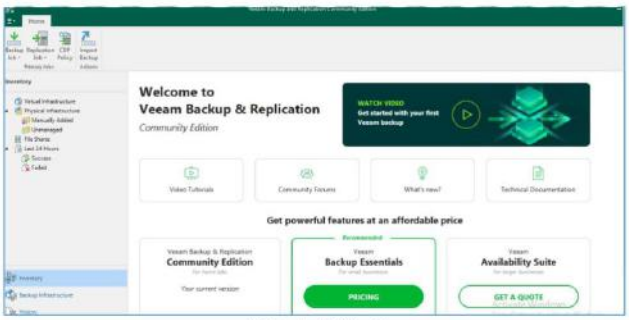
شکل ۶۵-۵ نصب Veew

بعد از نصب، نرم‌افزار Veeam backup & Replication Console را اجرا کنید، به‌مانند شکل ٦٦-٥ در قسمت اوّل، کلمهٔ Localhost نوشته شده است که اشاره می‌کند به سیستمی که در حال کار با آن هستید و نرم‌افزار بر روی آن نصب شده است، اگر نرم‌افزار بر روی سرور دیگری نصب شده باشد باید به‌جای Localhost، نام سرور مورد نظر را وارد کنید، برای ورود از همان کاربری استفاده می‌کند که با آن وارد سیستم شده‌اید؛ بر روی Connect کلیک کنید تا نرم‌افزار اجرا شود.



شکل ۶۶-۵ اجرای Veeam

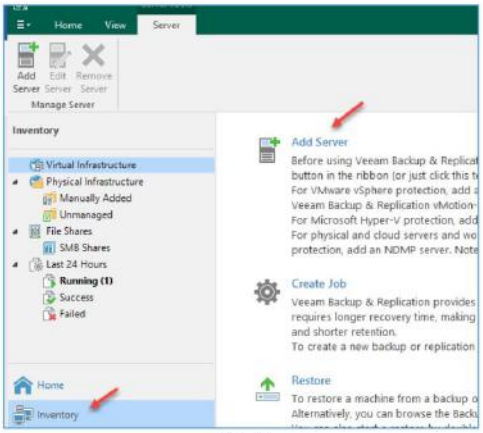
همان‌طور که در شکل ٦٧-٥ مشاهده می‌کنید، نرم‌افزار اجرا شده و آماده کار است، البته لایسنس آن باید فعال شود تا چند روز دیگر انقضا نشود، این نرم‌افزار دارای ابزارهای مختلفی است که می‌تواند عملکرد مشخصی را ارائه دهد، در ادامه نحوه‌ی پشتیبان‌گیری از ماشین مجازی و فایل‌های آنها مخصوصاً دیتابیس SQL را بررسی خواهیم کرد.



شکل ۶۷-۵ اجرای نرم‌افزار

۲-۲-۵ اضافه‌کردن سرورها برای پشتیبان‌گیری

در شکل ۶۸-۵، برای اینکه سرورهای و فایل‌هایی که قرار است از آنها پشتیبان تهیه کنیم را باید در قسمت Inventory تعریف کنیم، حالا می‌تواند یک ماشین مجازی باشد و یا یک فایل خاص در مسیر مشخص از شبکه، برای معرفی سرور خود می‌توانید بر روی Add Server کلیک کنید.



شکل ۶۸-۵

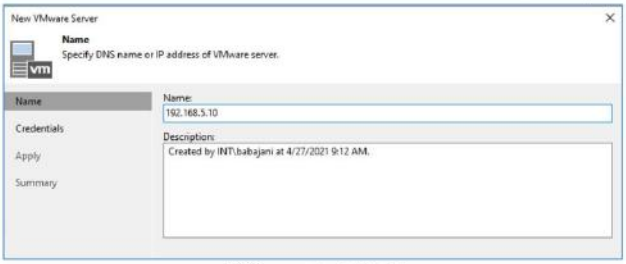
در شکل ۶۹-۵ باید مشخص کنید که سیستم مجازی‌سازی شما مربوط به کدام شرکت است اگر از VMware vSphere استفاده می‌کنید گزینه‌ی اول را انتخاب کنید و یا اگر از Hyper-v شرکت مایکروسافت استفاده می‌کنید گزینه‌ی دوم را انتخاب کنید، در مورد هر دو نرم‌افزار در کتاب‌های مدیر شبکه و Vmware Systems آموزش کافی داده شده و برای دریافت می‌توانید به سایت بنده مراجعه کنید، برای ادامه کار بر روی گزینه‌ی اول کلیک کنید.

شکل ۵-۶۹ معرفی سرور *vSphere*

در صفحه‌ی شکل ۵-۷۰ بر روی vSphere کلیک کنید تا سرور vCenter یا ESXi خود را معرفی کنیم.

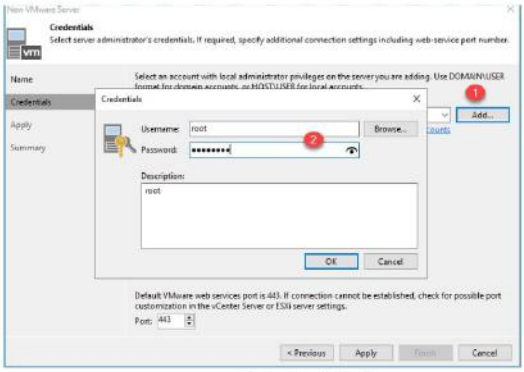
شکل ۵-۷۰ معرفی سرور *vSphere*

در شکل ۵-۷۱ باید آدرس سرور ESXi یا vCenter خود را وارد کنید، توجّه داشته باشید، البته شاید برای راه‌اندازی سرور SQL از نرم‌افزار VMware Workstation یا یک سرور فیزیکی استفاده کرده باشید که در ادامه کتاب نحوه پشتیبان‌گیری از آنها هم آموزش‌داده شده است.



شکل ۵-۷۱ معرفی سرور *ESXi*

در شکل ۵-۷۲ باید نام کاربری که برای ورود به سرور استفاده می‌کنید را وارد کنید، برای این کار بر روی Add کلیک کنید و در قسمت شمارهٔ دو، نام کاربر را وارد و بر روی OK کلیک کنید، پورت پیش‌فرض برای ارتباط با سرور، ٤٤٣ است که در صورت تغییر آن باید شمارهٔ پورت خود را وارد کنید.

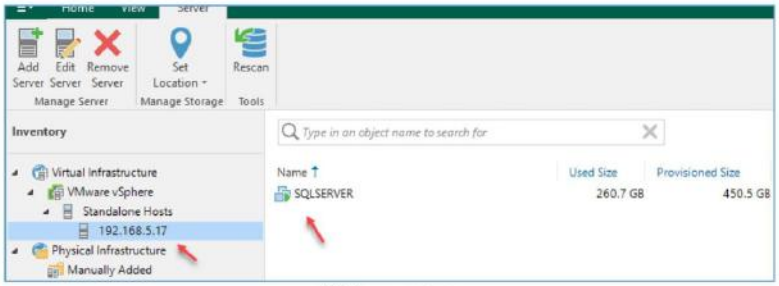


شکل ۷۳-۵ تعریف نام کاربری و رمز عبور

بعد از اینکه بر روی Apply کلیک کنید شکل ۷۳-۵ ظاهر خواهد شد که باید گواهینامه‌ی سرور را مورد تأیید قرار دهید.

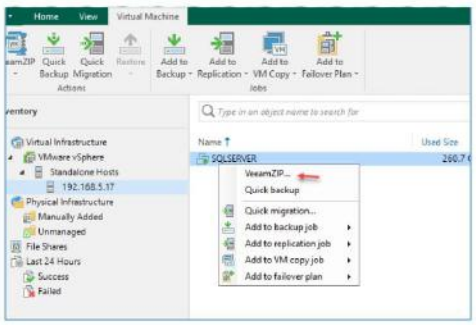
شکل ۷۳-۵ تأیید گواهینامه

همان‌طور که در شکل ۷۴-۵ مشاهده می‌کنید سرور مورد نظر به لیست اضافه شده است و با کلیک بر روی آن همه‌ی ماشین‌های مجازی مورد نظر را مشاهده خواهید کرد.



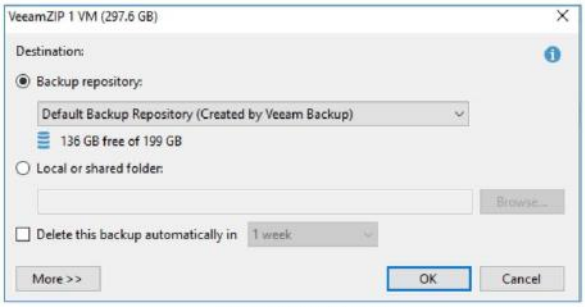
شکل ۷۴-۵ سرور *ESXi*

به‌مانند شکل ۷۵-۵، برای اینکه تستی از نرم‌افزار داشته باشیم، می‌توانیم بر روی ماشین مجازی خود در سرور Vmware کلیک راست کنیم و از آن پشتیبان تهیّه کنیم، قسمت شمارهٔ ۱ و ۲، ابزارهایی برای این کار وجود دارد؛ برای شروع بر روی VeeamZip کلیک کنید.



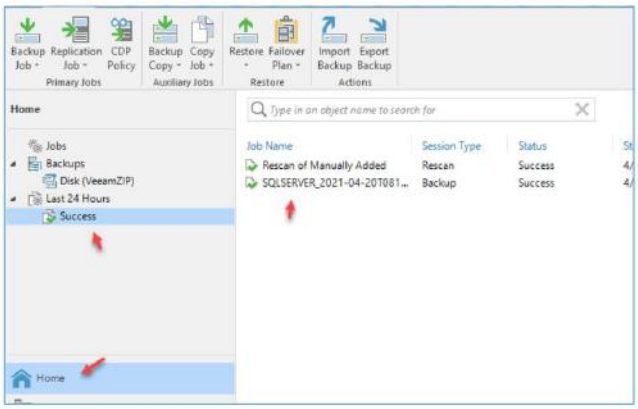
شکل ۷۵-۵ ایجاد پشتیبان دستی

در شکل ۷۶-۵ باید آدرس ذخیره شدن آن را بر روی سیستم خود مشخّص و بر روی ok کلیک کنید، توجه داشته باشید که برای ذخیره اطلاعات و فایل‌های پشتیبان باید در قسمت Repository یک مسیر ذخیره‌سازی مشخص کنیم که در اینجا به‌صورت پیش‌فرض یک مسیر در سروری که Veeam نصب شده است انتخاب شد ولی در ادامه و در قسمت Repository حتماً مراحل را به‌صورت کامل بررسی خ.اهیم کرد.



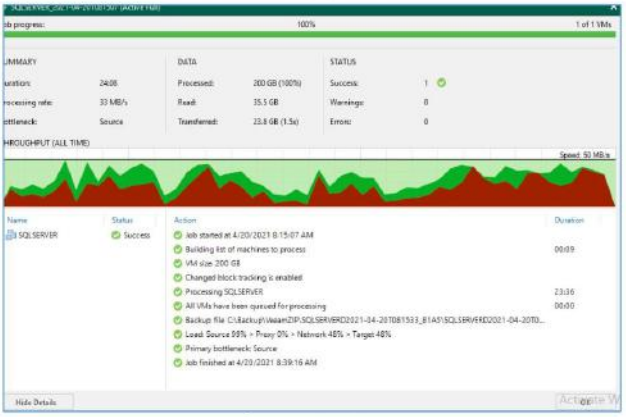
شکل ۷۶-۵ ایجاد پشتیبان دستی

برای اینکه متوجّه شویم کاری که انجام دادیم به‌درستی عمل کرد یا نه باید از سمت چپ بر روی Home کلیک کنید و در قسمت Last 24 Houres می‌توانیم آخرین تغییرات را مشاهده کنیم، در شکل ۷۷-۵ مشخص شده است که عملیات پشتیبان‌گیری از سرور SQL با موفقیت انجام شده است.



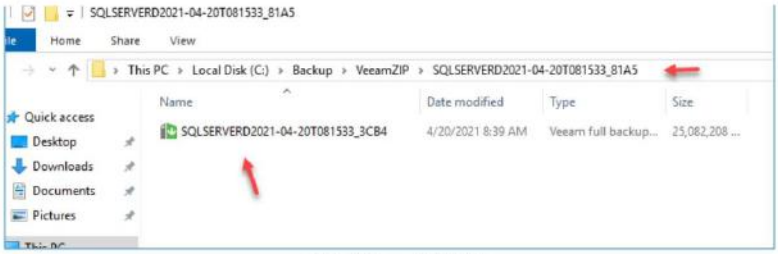
شکل ۷۷-۵ تغییرات فایل

اگر بر روی هرکدام از Log های مورد نظر در شکل ۷۷-۵ کلیک کنید، صفحه‌ای به‌مانند شکل ۷۸-۵ ظاهر می‌شود، در این قسمت، اطّلاعات مربوط به فایلی که از آن پشتیبان تهیّه کردید را مشاهده خواهید کرد، در این قسمت می‌توانید اندازهٔ بک آپ را در قسمت Transfer مشاهده کنید که برابر با ۲۳.۸ گیگابایت است.



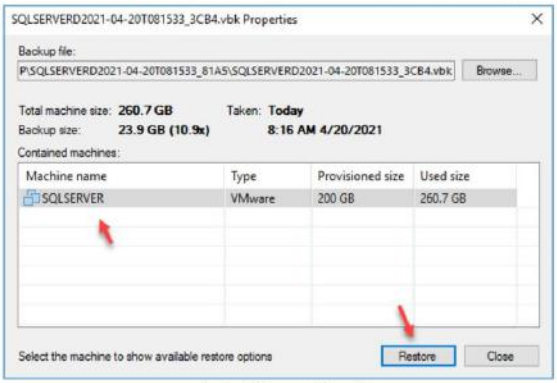
شکل ۷۸-۵ بررسی Log

اگر بخواهیم فایل پشتیبان را مشاهده کنید یا نه وارد مسیر ذخیره‌سازی مورد نظر شوید که به‌مانند شکل ۷۹-۵ به‌صورت یک فایل تکّی در قسمت مشخّص‌شده ذخیره شده است و حجم آن نیز ۲۳.۸ گیگابایت است، برای اینکه از این پشتیبان استفاده کنید باید بر روی آن دو بار کلیک کنید.



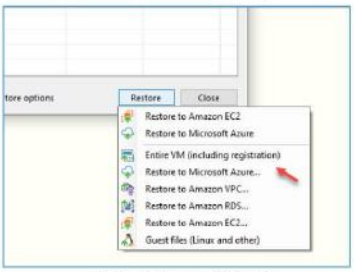
شکل ۷۹-۵ بررسی فایل پشتیبان

باتوجه‌به شکل ۸۰-۵ ، اندازهٔ فایل پشتیبان و تاریخ پشتیبانی آن مشخّص شده است که برای برگشت به آن روز باید فایل مورد نظر را از لیست، انتخاب و بر روی Restore کلیک کنید، با این کار بسته به‌سرعت شبکه و یا هارددیسک سرور ماشین مورد نظر شما برگشت داده خواهد شد



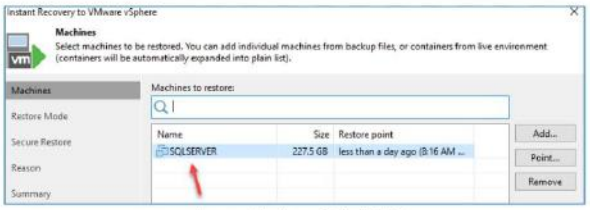
شکل ۸۰-۵ بررسی فایل پشتیبان

بعد از کلیک بر روی Restore منوی شکل ۸۱-۵ ظاهر خواهد شد که برای اینک ماشین مورد نظر را در همان مسیر Restore کنیم باید گزینه‌ی Entire VM را انتخاب کنید.



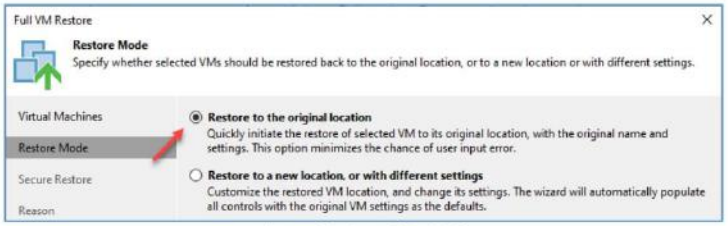
شکل ۸۱-۵ بررسی فایل پشتیبان

در شکل ۸۲-۵، آخرین بک آپ مربوط به ماشین مورد نظر را مشاهده می‌کنید که می‌توانید تاریخ مورد نظر خود را انتخاب کنید و بر روی Next کلیک کنید.



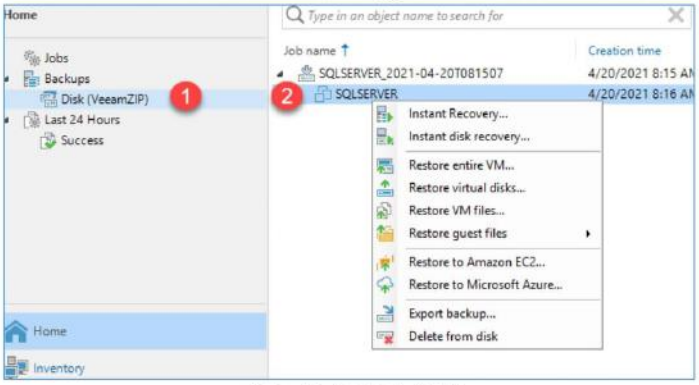
شکل ۸۲-۵ برگشت دادن فایل پشتیبان

در شکل ۸۳-۵ اگر گزینه‌ی اول را انتخاب کنید فایل پشتیبان دقیقاً جای همان ماشین مورد نظر را خواهد گرفت و اگر بخواهید مسیر آن را عوض کنید باید گزینه‌ی دوم را انتخاب کنید.



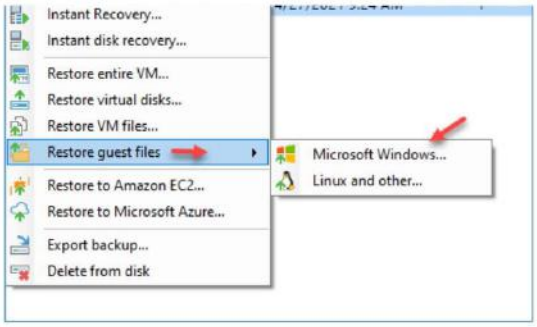
شکل ۸۳-۵ برگشت دادن فایل پشتیبان

روش دیگر برای استفاده از فایل بک آپ، این است که به‌مانند شکل ۸۴-۵ از قسمت شمارهٔ یک وارد Backup & Replication شوید و از قسمت شمارهٔ دو، Disk را انتخاب کنید؛ با انتخاب این گزینه، صفحهٔ مربوط به فایل‌های پشتیبان‌گیری شده را مشاهده خواهید کرد، بر روی فایل مورد نظر کلیک راست کنید؛ در این منو گزینه‌های مختلف وجود دارد، با انتخاب گزینهٔ اوّل می‌توانید خیلی سریع بک آپ مورد نظر را برگردانید.



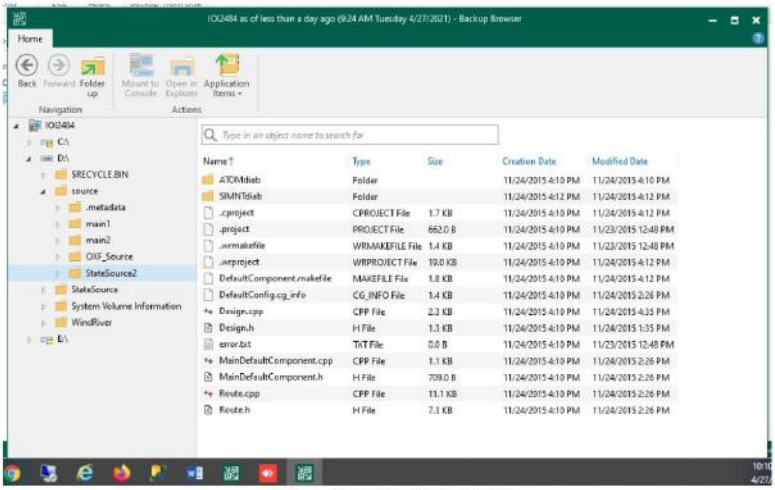
شکل ۸۴-۵ برگشت دادن فایل پشتیبان

اگر بخواهید اطلاعات ماشین مجازی را مشاهده کنید، مثلاً می‌خواهید یک فایل خاصی را از درون ماشین مجازی و از یکی از درایوهایش بردارید، برای این کار باید به‌مانند شکل ۸۵-۵ از قسمت Restore guest files گزینه‌ی Microsoft Windows را انتخاب کنید.



شکل ۵-۸۵ نمایش اطلاعات ماشین مجازی

همان‌طور که در شکل ۵-۸۶ مشاهده می‌کنید، فایل‌های مربوط به ماشین مجازی را مشاهده می‌کنید و اگر بخواهید فایل خاصی را برگردانید باید فایل مورد نظر را انتخاب کنید و در تب بالایی بر روی Restore کلیک کنید.



شکل ۵-۸۶ نمایش اطلاعات ماشین مجازی

این روش‌هایی که با هم بررسی کردیم، کار ساده‌ای است که این نرم‌افزار می‌تواند به‌خوبی انجام دهد، در ادامه بیشتر با ویژگی‌های قدرتمند آن آشنا خواهیم شد.

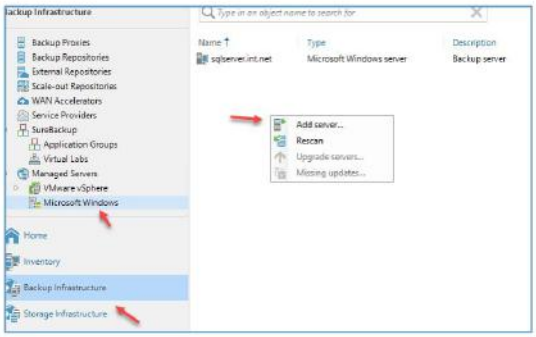
۵-۲-۳ اضافه‌کردن سرور Backup

در این قسمت می‌خواهیم کمی کار را باکیفیت‌تر کنیم؛ کار اصولی و درست این است که شما یک سرور پشتیبان یا همان، Repository را به نرم‌افزار معرّفی کنید و ماشین‌ها و اطّلاعات را به آن سرور انتقال دهید، در این قسمت می‌خواهیم این کار را با هم انجام دهیم. سروری که برای این کار انتخاب می‌کنید باید از سرعت شبکهٔ خوبی برخوردار باشد، یعنی باید حداقل سرعت گیگابایت داشته باشد تا در موقع پشتیبان‌گیری با عملکرد ضعیف روبرو نشوید، در بخش دیگر باید هارددیسک با ظرفیّت بالا برای آن در نظر بگیرید تا با کمبود فضا روبرو نشوید، اگر برای شرکت

شما، هزینه ملاک نیست می‌توانید از هارددیسک‌های SAS با سرعت بالا استفاده کنید که البتّه نیاز به سرورهایی دارد که آنها را پشتیبانی کند.

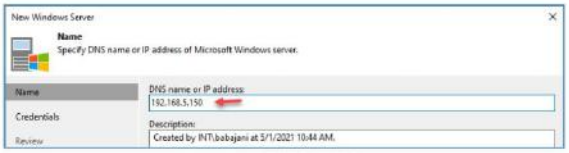
برای شروع کار می‌خواهیم یک سرور معمولی را به نرم‌افزار اضافه کنیم تا در ادامهٔ کار بتوانیم به‌عنوان یک پشتیبان از آن استفاده کنیم.

به‌مانند شکل ۸۷-۵ وارد قسمت Backup Infrastructure شوید و از سمت چپ بر روی Microsoft Windows کلیک کنید، به‌خاطر این موضوع این گزینه را انتخاب کردیم که سیستم‌عامل ما از نوع ویندوز است، بر روی Add Server کلیک کنید.



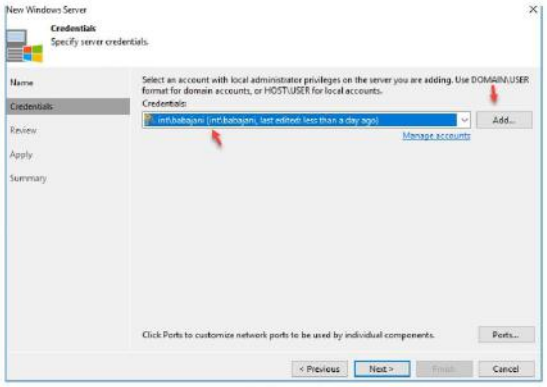
شکل ۸۷-۵ اضافه‌کردن سرور

در شکل ۸۸-۵ باید در قسمت مورد نظر، نام یا آدرس IP سرور خود که قرار است نقش پشتیبان را بازی کند را وارد کنید و بر روی Next کلیک کنید.



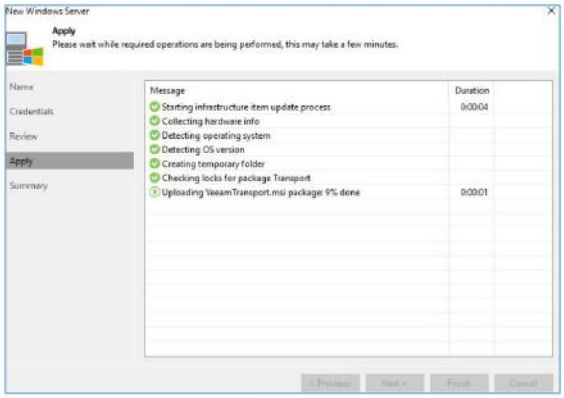
شکل ۸۸-۵ اضافه‌کردن سرور

در شکل ۸۹-۵ با کلیک بر روی Add باید نام کاربری را وارد کنید که دسترسی کامل به شبکه داشته باشد، اگر از قبل وارد کردید، می‌توانید آن را از لیست انتخاب کنید؛ در قسمت پایین صفحه، گزینهٔ Port وجود دارد که پورت‌هایی را که نرم‌افزار نیاز دارد تا با این سرور در ارتباط باشد را مشخّص می‌کند که البتّه قابل تغییر است.



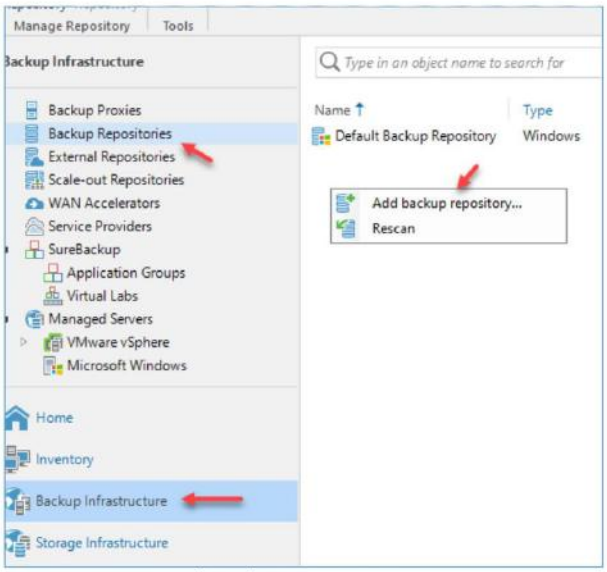
شکل ۹۸-۵ اضافه‌کردن سرور

همان‌طور که در شکل ۹۰-۵ مشاهده می‌کنید، اطّلاعات در سرور FS در حال نصب است، اگر به شکل ۹۰-۵ دقّت کنید، یک پکیج با نام Transport Veeam به سیستم FS انتقال داده شد و در حال نصب است که این پکیج، شامل سرویس‌های مربوط به نرم‌افزار Veeam است، بعد از پایان کار بر روی Finish کلیک کنید.



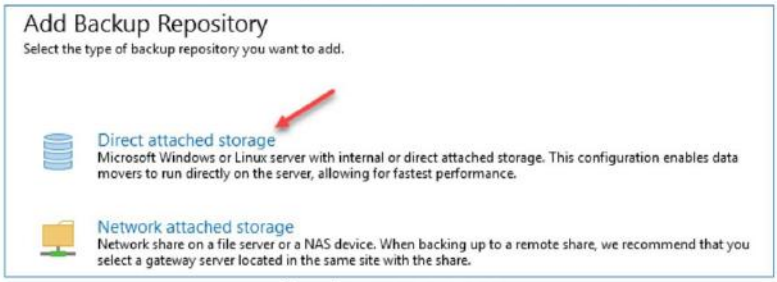
شکل ۹۰-۵ اضافه‌کردن سرور

در مرحلهٔ بعد باید سیستمی را که به‌عنوان پشتیبان اضافه کردید را در لیست Repositories یا همان، مخزن اضافه کنید تا بتوانید اطّلاعات را به آن انتقال دهید، برای این کار به‌مانند شکل ۹۱-۵ وارد Backup Repositories شوید و در صفحه باز شده کلیک راست و گزینه‌ی Add backup repository را انتخاب کنید.



شکل ۹۱-۵ اضافه‌کردن *Repoeitory*

در شکل ۹۲-۵ چند نوع مختلف Repository وجود دارد که برای این قسمت باید گزینه‌ی اول را انتخاب کنید.



شکل ۹۲-۵ اضافه‌کردن *Repoeitory*

در شکل ۹۲-۵ سیستم‌عامل ویندوز را انتخاب و بر روی Next کلیک کنید.

شکل ۹۲-۵ انتخاب سیستم‌عامل