«بـــــــه نام خـــالق آرامـــش»

نام کتاب: SQL Server ۲۰۱۹ (بخش اول)

نام نویسنده: فرشید بابجانی زاده

تعداد صفحات: ۱۵۴ صفحه

تاریخ انتشار: ــــــــــــــــ

FARSHID BABAJANI

Microsoft® SQL Server® 2019

کتاب آموزشی
SQL Server 2019

مؤلف: فرشید بابجانی زاده

سرشناسه: بابجانی زاده، فرشید، ۱۳۶۵
عنوان و نام پدیدآور: کتاب آموزشی SQL Server 2019 / فرشید بابجانی زاده
مشخصات نشر: 3isco.ir
مشخصات ظاهری: ۴۶۳ ص
شابک:
وضعیت فهرست‌نویسی: فیپا
موضوع:
موضوع:
رده‌بندی کنگره PIR:
رده‌بندی دیویی:
شماره کتاب‌شناسی ملی:

تقدیم به

روح بلند مادرم...

فهرست

مقدمه

با تشکر از انتخاب این کتاب به‌عنوان مرجع کامل آموزش **SQL Server 2019** که به‌عنوان کامل‌ترین کتاب در این زمینه ارائه شده است، در این کتاب سعی کردیم همه‌ی جنبه‌های **SQL Server** را بررسی و آموزش دهیم و امیدواریم با موفقیت همراه بوده باشد، البته برای این که کتاب به صورت کامل ارائه شود از منابع مختلف در کتاب استفاده شده است که لیست آن در آخر کتاب ذکر شده است.

مباحث کلی که در این کتاب بررسی شده است شامل:

۱- بررسی بنیادی نرم‌افزار **SQL Server**
۲- پیاده‌سازی **SQL Server 2019**
۳- کار با پایگاه‌داده و دستورات آن
٤- امنیت در **SQL Server 2019**
٥- پشتیبانگیری و بازیابی اطلاعات
٦- بررسی **SQL Replication**
۷- پیاده‌سازی سرویس **SQL Failover Clustring**
۸- هوش تجاری – **Business Intelligence**

فصل اول

مباحث بنیادی پایگاه‌داده

تقریباً بیشتر برنامه‌های کاربردی، امروزه به روش پایگاه‌داده طراحی می‌شوند. پایگاه‌داده امکان ذخیره‌سازی متمرکز داده با حداقل افزونگی و استفاده‌ی اشتراکی توسط کاربران مختلف را می‌دهد.

۱-۱ بررسی اولیه

۱-۱-۱ داده چیست

شکل ۱-۱

داده (data) دارای تعاریف مختلفی است، به طور کلی داده‌ها کلمات و ارزش‌های واقعی هستند که از طریق مشاهده و تحقیق بدست می‌آیند، به عبارت دیگر داده نمودی از وقایع، معلومات، رخدادها، پدیده‌ها و مفاهیم هستند.

در محاسبات، داده به اطلاعاتی گفته می‌شود که به شکلی مناسب برای انتقال و پردازش ترجمه شود. در کامپیوتر و رسانه‌های ارتباطاتی امروزی داده به شکل باینری تبدیل می‌شود. بنابراین داده یک نمایش باینری از یک موجودیت منطقی ذخیره شده در حافظه کامپیوتر است.

ریشه کلمه‌ی داده از عبارت لاتین datum گرفته شده که به معنی اطلاع است و data فرم جمع آن است. اما datum به ندرت استفاده می‌شود و اکثراً ترجیح می‌دهند data را به‌صورت یک کلمه مفرد استفاده کنند.

۲-۱-۱ اطلاعات چیست

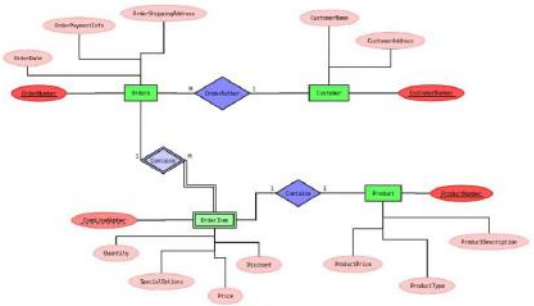
شکل ۲-۱

اطلاعات (information) مفهومی است که برای گیرنده آن قابل‌درک بوده و با دانستن آن می‌تواند برای حل یک مسئله تصمیم‌گیری یا ارزیابی کند.

وقتی اطلاعات وارد کامپیوتر شود، ذخیره می‌شود و به داده تبدیل می‌شود. بعد از پردازش، داده خروجی مجدداً به‌عنوان اطلاعات دریافت می‌شود.

هنگامی‌که اطلاعات در یک بسته خاص قرار می‌گیرند و برای درک موضوعی یا انجام کاری استفاده می‌شود به دانش (knowledge) تبدیل می‌شود.

۳-۱-۱ موجودیت چیست



شکل ۳-۱

موجودیت (entity) مفهوم کلی پدیده، شیء یا فردی است که در مورد آن می‌خواهیم اطلاع داشته باشیم و در کامپیوتر ذخیره کنیم.

هر نوع موجودیت دارای مجموعه‌ای از صفات خاصه (attribute) است که ویژگی جداکننده یک نوع موجودیت از نوع دیگر محسوب می‌شود.

اگر در نظر داریم یک سیستم پایگاه‌داده برای یک دبیرستان پیاده‌سازی کنیم مواردی چون دانشجویان، استادان، دروس، کلاس‌ها و غیره جزء موجودیت‌های سیستم به شمار می‌روند.

موجودیت دانشجو در سیستم دانشگاه می‌تواند دارای صفات خاصه: نام، نام خانوادگی، سن، سال تولد، رشته تحصیلی، سال ورود و ... باشد و یا موجودیت درس صفات خاصه: کد درس، نام درس و تعداد واحد باشد.

٤-١-١ پایگاه‌داده چیست

DATABASE

شکل ۴-۱

یک بانک اطلاعاتی یا پایگاه‌داده (database) مجموعه‌ای سازمان‌یافته و بدون افزونگی از داده‌های مرتبط به هم است که می‌تواند توسط سیستم‌های کاربردی مختلف به اشتراک گذاشته شود و به‌راحتی دسترسی، مدیریت و بهنگام شود. وقتی داده به‌صورت پایگاه‌داده سازماندهی می‌شود، کاربر و برنامه‌نویس نیازی به دانستن جزئیات ذخیره‌سازی داده ندارند. علاوه بر این داده می‌تواند بدون تأثیر روی اجزای دیگر سیستم تغییر کند. برای مثال از اعداد حقیقی به صحیح، از یک ساختار فایل به دیگری یا از دستگاه ذخیره‌سازی نوری به مغناطیسی تغییر کند.

١-٤-١-١ ویژگی‌های داده در پایگاه‌داده

ویژگی‌هایی که داده در پایگاه‌داده باید داشته باشند در لیست زیر آمده است:

١. اشتراکی شدن (Shared)

داده در پایگاه‌داده بین چندین کاربر و برنامه کاربردی به اشتراک گذاشته می‌شود.

٢. ماندگاری (Persistence)

وقتی داده در پایگاه‌داده ذخیره شد پایدار است و از بین نمی‌رود، مگر آنکه توسط سیستم پایگاه‌داده تغییر کند.

٣. امنیت (Security)

داده در پایگاه‌داده از فاش شدن، تغییر و تخریب بدون مجوز محافظت می‌شود. مدیر سیستم توسط سطوح دسترسی و قیدهای امنیتی نحوه دستیابی به داده را مشخص می‌کند و اطمینان می‌دهد که دستیابی از طریق مناسب انجام می‌شود.

٤. اعتبار (Validity)

یا جامعیت (Integrity) و یا صحت (Correctness) به‌درستی داده در پایگاه‌داده نسبت به موجودیت دنیای واقعی معتبر اشاره دارد. مثلاً موجودی بانک نباید منفی باشد.

۵. سازگاری (Consistency)

داده در پایگاه‌داده با مقدار واقعی داده در دنیای خارج سازگار است. وقتی یک فقره اطلاع در بیش از یک نقطه ذخیره شود و لازم باشد به‌هنگام شود، اگر به‌هنگام‌سازی در همه نقاط انجام نشود ناسازگاری ایجاد می‌شود.

٦. کاهش افزونگی (Non redundancy)

داده در پایگاه‌داده دارای حداقل افزونگی است. افزونگی به این معناست که هیچ دو فقره داده در بانک معرف یک موجودیت در دنیای خارج نباشد.

۷. استقلال (independence)

تغییر در نمایش فیزیکی، تکنیک‌های دستیابی و سازماندهی داده تأثیری روی برنامه‌های کاربردی ندارد.

۲-٤-۱-۱ روش‌های ذخیره داده

دو روش کلی برای ذخیره و بازیابی خودکار داده‌ها وجود دارد: سیستم فایل ساده و سیستم پایگاه‌داده.

سیستم فایل (file system)

در این روش، داده‌ها در فایل‌های مجزا قرار گرفته و سیستم‌های جداگانه‌ای به نام سیستم پردازش فایل برای استفاده از فایل‌های داده‌ای طراحی می‌شوند. در این سیستم‌ها هر برنامه‌ی کاربردی تنها به فایل داده‌ای مربوط به خود می‌تواند مراجعه کند.

اشکالات چنین طراحی در ذخیره داده به طور خلاصه عبارت‌اند:

۱. افزونگی و ناسازگاری داده به دلیل چندین فرمت فایل و تکرار اطلاعات در فایل‌های مختلف.

۲. مشکل در دستیابی داده و نیاز به نوشتن برنامه جدیدی برای انجام هر کار.

۳. قیدهای جامعیت به‌جای اینکه صریحاً بیان شوند در کد برنامه از نظر پنهان می‌شوند. اضافه‌کردن قیدهای جدید یا تغییر قیدهای موجود به‌سختی صورت می‌گیرد.

٤. ایجاد ناسازگاری به دلیل وجود چندین کپی از فقره‌های داده.

۵. مشکلات امنیتی به دلیل دسترسی همروند و بدون کنترل توسط چند کاربر

۳-٤-۱-۱ سیستم پایگاه‌داده (database system)

در این روش کلیه داده‌ها به‌صورت مجتمع در پایگاه‌داده ذخیره می‌شود، ولی هر کاربر دید خاص خود را نسبت به داده‌ها دارد. کاربران مختلف می‌توانند به طور مشترک با پایگاه‌داده کار کنند. به دلیل تجمع داده افزونگی به حداقل ممکن کاهش می‌یابد.

نرم‌افزاری به نام سیستم مدیریت پایگاه‌داده (DBMS) به‌عنوان واسطه بین برنامه‌های کاربردی و پایگاه‌داده ایفای نقش می‌کند لذا امنیت داده‌ها در این روش بیشتر است.

چند نمونه از کاربردهای سیستم پایگاه‌داده موارد زیر هستند:

- انجام کلیه تراکنش‌های بانکداری.
- رزرواسیون و زمان‌بندی خطوط هوائی.
- ثبت‌نام دانشجویان، واحدگیری و ثبت نمرات در مراکز آموزشی.
- ثبت اطلاعات مشتریان، محصولات و فاکتورهای خریدوفروش.
- پیگیری سفارشات و پیشنهادها در فروش online.
- ثبت رکوردهای کارمندان و محاسبات حقوق، کسورات مالیاتی در سازمان‌ها.

٤-٤-۱-۱ عناصر اصلی سیستم پایگاه‌داده

اجزاء اصلی سیستم بانک اطلاعاتی عبارت‌اند از:

۱. داده‌ها

شامل داده‌هایی درباره موجودیت‌های مختلف محیط و ارتباط بین موجودیت‌ها.

۲. سخت‌افزار

شامل عناصر پردازشی، رسانه‌های ذخیره‌سازی داده، دستگاه‌های جانبی، سخت‌افزارهای ارتباطی و غیره.

۳. نرم‌افزار

شامل سیستم‌عامل و نرم‌افزارهای ارتباطی شبکه، نرم‌افزار سیستم مدیریت پایگاه‌داده و برنامه‌های کاربردی.

٤. رویه‌های عملیاتی

شامل کلیه عملیاتی که روی پایگاه‌داده انجام می‌شود، نظیر تهیه پشتیبان، آمارگیری و ...

۵. کاربر

شامل کاربران یا کسانی که به نحوی با سیستم در ارتباط هستند نظیر مدیر پایگاه‌داده (DBA)، طراحان پایگاه‌داده (DBD)، برنامه‌نویسان پایگاه‌داده (DBP) و کاربران نهائی (end users).

۵-٤-۱-۱ مزایا و معایب سیستم‌های پایگاه‌داده

مزایای مهم سیستم پایگاه‌داده

۱- تجمع، وحدت ذخیره‌سازی و کنترل متمرکز داده‌ها.

کاهش افزونگی. تجمع داده و وحدت ذخیره‌سازی باعث کاهش افزونگی می‌شود. مثلاً آدرس‌های مختلف برای یک مشتری در قسمت‌های مختلف اداره ثبت نمی‌شود.

۲- به اشتراک گذاشتن داده‌ها

چند کاربر می‌توانند هم‌زمان به یک پایگاه‌داده دسترسی داشته باشند. برنامه‌های کاربردی موجود قادر به اشتراک گذاردن داده‌ها در پایگاه‌داده بوده و برنامه‌های کاربردی جدید نیز می‌توانند از این داده‌ها استفاده کنند.

۳- پرهیز از ناسازگاری

با کاهش افزونگی، کنترل متمرکز و جامعیت، سازگاری و یکپارچگی داده‌ها تضمین می‌شود.

٤- اعمال محدودیت‌های امنیتی

سیستم‌های امنیتی در پایگاه‌داده امکان اعمال کنترل‌های مختلف را برای هر نوع دسترسی (بازیابی، اصلاح، حذف و غیره) بر روی پایگاه‌داده فراهم می‌کند.

۵- صحت بیشتر داده و استقلال از برنامه‌های کاربردی.

٦- راحتی پیاده‌سازی برنامه‌های کاربردی جدید.

معایب سیستم پایگاه‌داده

۱- طراحی سیستم‌های پایگاه‌داده پیچیده‌تر، دشوارتر و زمان برتر است.

۲- هزینه قابل‌توجه، صرف سخت‌افزار و نصب نرم‌افزار می‌شود.

۳- آسیب‌دیدن پایگاه‌داده، روی کلیه‌ی برنامه‌های کاربردی تأثیر می‌گذارد.

٤- هزینه زیاد برای تبدیل از سیستم فایلی به سیستم پایگاه‌داده نیاز است.

۵- نیازمند تعلیم اولیه برنامه‌نویسان و کاربران و استخدام کارمندان خاص پایگاه‌داده است.

٦- نیاز به تهیه چندین کپی پشتیبان از پایگاه‌داده می‌باشد.

۷- خطاهای برنامه می‌توانند فاجعه برانگیز باشند.

۸- زمان اجرای هر برنامه طولانی‌تر می‌شود.

۹- بسیار وابسته به عملیات سیستم مدیریت پایگاه‌داده است.

۵-۱-۱ سیستم مدیریت پایگاه‌داده

شکل ۵-۱

سیستم مدیریت پایگاه‌داده یا به طور خلاصه DBMS (DataBase Management System) مهم‌ترین نرم‌افزاری در سیستم پایگاه‌داده است که به‌عنوان رابط بین پایگاه‌داده و کاربر و برنامه‌های کاربردی عمل می‌نماید.

کلیه فایل‌های پایگاه‌داده فقط در اختیار این نرم‌افزار قرار گرفته و دستیابی به آنها تنها از طریق DBMS امکان‌پذیر است.

DBMS سرویس‌هایی جهت دسترسی داده در پایگاه‌داده فراهم می‌کند به‌نحوی‌که از کلیه خواص داده محافظت شود.

۱-۵-۱-۱ وظایف سیستم مدیریت پایگاه‌داده

وظایف DBMS در سیستم‌های مختلف تا حدودی متفاوت بوده و بستگی به نوع کاربران آن دارد. اما به‌طورکلی این وظایف عبارت‌اند از:

۱- امکان تعریف پایگاه‌داده.
۲- امکان ایجاد پایگاه‌داده.
۳- امکان دست‌کاری در داده‌ها.
٤- بازیابی پایگاه‌داده.
۵- به‌هنگام‌سازی پایگاه‌داده (عملیات درج، حذف و جایگزینی).
٦- تأمین تسهیلاتی برای کاربر به‌منظور توسعه سیستم.
۷- امکان سازماندهی مجدد.
۸- کنترل امنیت و جامعیت داده‌ها.
۹- ایجاد دیکشنری داده‌ها.
۱۰- امکان کنترل کارایی.

۲-۵-۱-۱ تراکنش

شکل ۶-۱

تراکنش (transaction) یک برنامه فعال است که دنباله‌ای از دستورات را شامل می‌شود و به طور خاص بعضی عملیات آن روی پایگاه‌داده است.

سه عمل تراکنشی خاص وجود دارد:

Start که نشان می‌دهد یک تراکنش در حال شروع شدن است.

Commit که دلالت بر اتمام عادی تراکنش دارد.

Abort که بیان‌کننده پایان یافتن تراکنش به دلیل لغو آن است و کلیه اثرات تراکنش لغو شده باید rollback یا بی‌اثر شود. وقتی تراکنش commit می‌شود تأثیرش روی پایگاه‌داده باید دائمی شود.

هر تراکنش باید پایگاه‌داده را از یک حالت سازگار به حالت سازگار بعدی ببرد. تراکنش باید دارای خواص ACID باشد تا پایگاه‌داده را در حالت سازگار باقی نگهدارند. خواص ACID حروف اول چهار خاصیت زیر است:

۱.اتمی بودن (Atomicity)

تراکنش‌ها اتمیک هستند یا اصلاً شروع نمی‌شوند یا وقتی آغاز شدند حتماً به پایان می‌رسند. یا تمام عملیات انجام می‌شود یا هیچ‌کدام.

نگه‌داشتن خاصیت اتمیک به عهده‌ی کنترل هم‌روندی و ترمیم است.

۲. سازگاری (Consistency)

یک تراکنش یا پایگاه‌داده را به حالت سازگار جدیدی می‌برد یا اگر شکستی رخ داد کلیه داده‌ها به حالت قبل از شروع تراکنش برمی‌گردند.

۳. ایزوله بودن (Isolation)

تراکنشی که در حال اجراست و هنوز به پایان نرسیده تأثیرش از بقیه مخفی است مگر اینکه commit شده باشند. اجرای هم‌روند تراکنش‌ها باید به‌صورتی باشد که انگار پشت‌سرهم اجرا شده‌اند. حفظ این خاصیت برعهده کنترل هم‌روندی است.

٤. ماندگاری (Durability)

از وقتی تراکنشی commit شد تأثیرش دائمی است؛ حتی اگر سیستم خراب شود، داده در حالت درست خود باقی می‌ماند.

۳-۵-۱-۱ اجزای سیستم مدیریت پایگاه‌داده

وظایف DBMS توسط تعدادی مؤلفه نرم‌افزاری انجام می‌شود. هرکدام از این مؤلفه‌ها ممکن است مرکب از چند واحد کوچک‌تر باشند. تعدادی از سرویس‌های که توسط مؤلفه‌های DBMS داده می‌شوند در زیر لیست شده است:

پردازش تراکنش (Transaction Processing)

پردازش تراکنش، عملیاتی که از منابع مختلف می‌رسد را روی پایگاه‌داده اجرا می‌کند به‌نحوی‌که خواص مطلوب تراکنش خدشه‌دار نشود. سرویس‌های کنترل هم‌روندی و ترمیم به این مؤلفه برای برقراری خواص ACID کمک می‌کنند. به‌این‌ترتیب اجرای هم‌روند تراکنش‌ها و سازگاری پایگاه‌داده حتی در صورت وقوع شکستی در سیستم تضمین می‌شود.

کنترل هم‌روندی (Concurrency Control)

مدیریت اجرای همروند تراکنش‌ها روی پایگاه‌داده در حین برقراری سازگاری را به عهده دارد.

ترمیم (Recovery)

ترمیم تضمین می‌کند که اگر اجرای تراکنش با عدم موفقیت یا لغو روبرو شد، تأثیر نامطلوبی بر روی پایگاه‌داده یا تراکنش‌های دیگر نگذارد و حالت پایگاه‌داده را همیشه سازگار نگه دارد.

مدیریت ثبت (Log Management)

هر اتفاقی در سیستم در یک فایل ذخیره می‌شود و توسط مدیریت ترمیم برای حفظ صحت و اعتبار پایگاه‌داده هنگام خرابی یا لغو سیستم استفاده می‌شود.

واسطه زبانی (Language Interface)

دستوراتی را برای تعریف داده، کارکردن با آن اختیار کاربران و برنامه‌های کاربردی قرار می‌دهد.

تحمل‌پذیری خطا (Fault Tolerancy)

توانایی ارائه سرویس‌های قابل‌اطمینان توسط DBMS حتی در صورت بروز نقص را تحمل‌پذیری خطا می‌گویند. انواع خطاهایی که ممکن است پیش بیاید عبارت‌اند از:

- خطای منطقی: تراکنش موفق نمی‌شود مثلاً به دلیل ورودی بد، سرریزی.

• خطای سیستمی: تراکنش موفق نمی‌شود مثلاً به دلیل بن‌بست.
• لغو سیستم: قطع برق، پاک شدن حافظه اصلی، پرشدن دیسک.
• ناتوانی دیسک: خرابی هد، خرابکاری عمدی، آتش‌سوزی.

کاتالوگ داده (Data Catalog)

یک دیکشنری داده یک پایگاه‌داده سیستمی شامل اطلاعاتی درباره داده، ارتباطات و قیدها در پایگاه اصلی است. گاهی به آن متا داده هم گفته می‌شود.

امنیت (Security)

امنیت به محافظت داده در مقابل افشا شدن، تغییر و خرابی اشاره دارد. هر کاربر و برنامه کاربردی امتیاز ویژه‌ای برای دسترسی به داده دارند. کاربران ممکن است دیدگاه‌های مختلفی نسبت به داده‌های پایگاه‌داده با توجه امتیازات ویژه خود داشته باشند. سیستم امنیتی همچنین، توسط رویه‌های شناسایی و مجوز، دسترسی به پایگاه‌داده را محدود می‌کند.

مدیریت ذخیره‌سازی (Storage Management)

DBMS مکانیسم‌های خاصی برای ذخیره دائمی داده و دسترسی به منبع فیزیکی و بازیابی داده دارد. مدیر ذخیره‌سازی بین داده ذخیره شده در پایگاه‌داده و برنامه کاربردی و پرس‌وجوهای ارسال شده به سیستم واسطه می‌شود.

مدیریت قفل (Lock Management)

هنگام استفاده اشتراکی از داده انواع مختلفی از قفل روی داده گذاشته می‌شود (مثل Read Lock و Write Lock).

مدیریت بن‌بست (Deadlock Management)

بن‌بست وقتی اتفاق می‌افتد که تراکنش‌ها برای به‌دست‌آوردن منابع در یک دایره بسته قرار گیرند یعنی هر یک منبعی در اختیار دارد که مورد تقاضای دیگری است و درخواست منبعی را می‌کند که در اختیار تراکنش منتظر منبع است. در پایگاه‌داده منابع رکوردها هستند. مدیریت منبع مسئول رفع این مشکل هستند.

۱-۱-۶ انواع سیستم‌های مدیریت پایگاه‌داده

انواع مختلفی از سیستم‌های پایگاه‌داده وجود دارند که هرکدام به‌منظور خاصی طراحی و پیاده‌سازی شده‌اند. دسته‌بندی سیستم‌های پایگاه‌داده به‌صورت زیر انجام‌گرفته است:

۱-۱-۶-۱ سیستم مدیریت پایگاه‌داده توزیع شده

سیستم‌های توزیع شده (Distributed DataBase Management System) بر توزیع داده و همچنین همبستگی فعالیت‌ها و کنترل روی اجزای توزیع شده سیستم دلالت دارند. اکثر سیستم‌های توزیع شده برای تقسیم کردن بار کاری یا برای انتقال عملکردهای پردازش داده به نزدیکی محل انجام این وظایف است. در هر دو حالت هدف نامحسوس بودن توزیع‌شدگی از دید کاربر است.

۲-٦-۱-۱ سیستم مدیریت پایگاه‌داده بلادرنگ

سیستم‌های بلادرنگ (Real-Time DataBase Management System) سیستم‌های سریع با سرعت پاسخگویی بالا هستند که زمان انجام کلیه عملیات نقش مهمی در آنها دارد. سیستم بلادرنگ در تعامل با دنیای واقعی پاسخ قابل پیش‌بینی را در قاب زمان می‌دهد. ورودی، پردازش و پاسخ‌ها همگی از قبل تعریف شده هستند و حد زمانی مشخصی دارند و به نحوی بهینه می‌شوند که هر حالت ورودی یک حالت خروجی قابل‌پیش‌بینی دارد که همیشه در یک‌زمان و به یک روش اتفاق می‌افتد.

۳-٦-۱-۱ سیستم مدیریت پایگاه‌داده تحمل‌پذیر خطا

سیستم تحمل‌پذیر خطا (Fault Tolerance DataBase Management System) سرویس‌هایی را دارد که با ناتوانی‌های اجزای سخت‌افزاری و نرم‌افزاری برخورد می‌کند. برای رسیدن به این منظور باید کلیه نقاطی که احتمال نقصی در آنها وجود دارد از قبل بررسی شده، ابزارهایی برای تشخیص، اصلاح و یا ترمیم آنها به نحوی طراحی شود که کمترین تأثیر را روی برنامه‌های کاربردی بگذارند. مکانیسم‌های RAID، Shadow Memory و کپی از جمله روش‌هایی هستند که استفاده می‌شوند.

٤-٦-۱-۱ سیستم مدیریت پایگاه‌داده امن

در یک سیستم پایگاه‌داده مطمئن (Secure DataBase Management System) کلیه اعمالی که کاربران و برنامه‌های کاربردی اجازه دارند انجام دهند همچنین زمان و مقدار انجام آنها کنترل می‌شوند. به‌عنوان مثال یک سیستم پرسنلی ممکن است درنظرداشته باشد به کلیه کاربران اجازه دستیابی به سابقه پرسنلی خودشان و استخراج اطلاعات شغلی‌شان را بدهد اما دسترسی به سابقه کارمندان دیگر یا حتی برخی اطلاعات مربوط به خودشان امکان‌پذیر نباشد. برای دادن چنین سرویسی سیستم پایگاه‌داده باید قابلیت تعریف حقوق دسترسی و رسیدگی به آنها را درقبال کاربرانی که به داده دسترسی دارند داشته باشد.

۵-۶-۱-۱ سیستم مدیریت پایگاه‌داده ناهمگون

سیستم ناهمگون (Hetrogenouse DataBase Management System) از DBMSهای مختلف تشکیل شده است. برای مثال شعب یک شرکت هرکدام منحصراً نیازهای پردازشی خود را با سخت‌افزار و نرم‌افزار جداگانه برطرف می‌کنند. اگر نیاز باشد این سیستم‌ها با هم فعل‌وانفعال داشته باشند و از طریق شبکه به هم‌پیوند داده شوند یک HTDBMS ایجاد می‌شود تا پایگاه‌داده‌های مختلف با هم ارتباط برقرار کند.

۶-۶-۱-۱ سیستم مدیریت پایگاه‌داده چندرسانه‌ای

سیستم‌های محاسباتی چندرسانه‌ای (Multimedia DataBase Management System) انواع متنوعی از منابع داده‌ای گرافیکی، تصاویر ویدئویی، صوت و متن را استفاده و یا با هم ترکیب می‌کنند. این منابع داده‌ای پیچیده باید برای سیستم محاسباتی به‌سهولت قابل‌دسترس باشند. برای استفاده در برنامه‌های کاربردی interactive چنین سیستم‌هایی از ترکیب الزامات پایگاه‌داده‌های بلادرنگ با سیستم‌های گرافیکی تعاملی استفاده می‌کنند تا ارائه اطلاعات سنکرون شده و بلادرنگ حاصل شود.

۷-۶-۱-۱ سیستم مدیریت پایگاه‌داده متحد

Federated DataBase Management System، نسل جدید سیستم‌های مدیریت پایگاه‌داده سعی دارند اطلاعات جمع‌آوری‌شده از سنسورها را مستقیماً ذخیره کنند. این سیستم‌ها از پایگاه‌های دانش نیز حمایت می‌کنند.

۷-۱-۱ کاربران پایگاه‌داده

کاربران یک سیستم پایگاه‌داده توسط روش‌هایی که با سیستم تعامل می‌کنند از هم تفکیک می‌شوند.

۱-۷-۱-۱ تحلیل گران سیستم

تحلیل گران سیستم (system analysts) با گروه کاربران پایگاه‌داده به‌منظور درک نیازهای اطلاعاتی و پردازشی آنها ارتباط دارند. نیازهای اطلاعاتی و پردازشی هر گروه را مجتمع می‌کنند و مستندسازی می‌کنند.

۲-۷-۱-۱ طراحان پایگاه‌داده

طراحان پایگاه‌داده (database designers) ساختار مناسبی را برای نمایش اطلاعات مشخص شده توسط تحلیل‌گر سیستم به طریق نرمال‌سازی شده به‌منظور تضمین جامعیت و سازگاری داده انتخاب می‌کنند و با استفاده از DDL داده‌های پایگاه‌داده را تعریف می‌کنند.

۳-۷-۱-۱ پیاده سازان برنامه‌های کاربردی

برنامه‌نویسان برنامه‌های کاربردی (Application Developers) برای برآوردن نیازهای کاربران و کار با پایگاه‌داده برنامه‌هایی را آماده می‌کنند. تست، اشکال‌زدایی و مستندسازی برنامه و پایگاه‌داده از وظایف برنامه‌نویسان است. برنامه‌نویسان با سیستم توسط احکام DML ارتباط برقرار می‌کنند.

٤-۷-۱-۱ مدیر پایگاه‌داده

مدیر پایگاه‌داده (database administrator) یا به طور خلاصه DBA فردی است که مسئول کنترل عملیات کل سیستم پایگاه‌داده است. DBA کلیه فعالیت‌های سیستم پایگاه‌داده را هماهنگ می‌کنند. این فرد باید درک خوبی از منابع و نیازهای اطلاعاتی کل سازمان داشته باشد و برای حصول اطمینان از اینکه داده موردنیاز قابل‌دسترس کاربران قرار می‌گیرد با آنها در ارتباط باشد.

بعضی از وظایف DBA شامل:

- تعریف شماها توسط DDL
- تعریف ساختار ذخیره‌سازی و متدهای دسترسی توسط DDL
- اصلاح شما و سازماندهی فیزیکی
- اعطای مجوز دسترسی پایگاه‌داده به کاربران
- تعیین قیدهای جامعیت
- عامل ارتباطی کاربران
- نظارت اجرا و واکنش برای تغییر درصورت نیاز
- برقراری دیکشنری داده

۵-۷-۱-۱ کاربران نهائی

کاربران نهائی (End Users) شامل:

- کاربران پارامتری: که توسط برنامه‌های کاربردی نوشته شده با سیستم سروکار دارند. مانند تحویلدار بانک و کارکنان دفتری.
- کاربران ماهر: که نیازهای پیچیده‌تری دارند و باقابلیت‌های DBMS آشنائی کامل دارند. درخواست‌های خود از پایگاه‌داده را توسط یک زبان پرس‌وجو می‌سازند.
- کاربران نهایی اتفاقی: کسانی که دسترسی گاه‌وبیگاه به پایگاه‌داده دارند اما ممکن است هر بار نیازهای متفاوتی داشته باشند. از زبان‌های پرس‌وجوی و مرورگرهای حرفه‌ای‌تر استفاده می‌کنند.

۸-۱-۱ دیکشنری داده

دیکشنری داده‌ها (Data Catalog) یکی از امکاناتی است که در سیستم پایگاه‌داده در اختیار DBA قرار می‌گیرد. دیکشنری داده‌ها که به آن راهنمای سیستم نیز می‌گویند یک متا داده است یعنی اطلاعاتی درباره خود پایگاه‌داده و داده‌های ذخیره شده در آن را نگهداری می‌کند.

دیکشنری داده تعیین می‌کند چه داده‌ای موجود است و چه معنی دارد، داده چگونه ذخیره می‌شود و در کجا قرار دارد، مالک آن چه کسی است و چه کسانی اجازه دسترسی به داده را دارند، تاریخچه و آمار استفاده از داده را در بردارد.

۹-۱-۱ پایگاه‌دادهXML

یک پایگاه‌داده XML سیستم نرم‌افزاری است که اجازه می‌دهد داده در فرمت XML وارد، پردازش و ارسال شود .

دودسته اصلی پایگاه‌داده XML وجود دارد:

XML-enabled - پایگاه داده‌ای که مستند XML را به‌عنوان ورودی گرفته به یک پایگاه‌داده دیگر نظیر رابطه‌ای تبدیل می‌کند و پس از انجام عملیات آنها را مجددا به XML بر می گردند .

Native XML (NXD) - مدل داخلی چنین پایگاه‌داده ای بر پایه XML است و مستندات XML را به‌عنوان منبع ذخیره‌سازی مستقیماً استفاده می‌کند.

دلیل استفاده XML در پایگاه‌داده شفافیت داده است. داده از پایگاه‌داده استخراج می‌شود و در مستندات XML قرار می‌گیرد و برعکس. به این صورت هزینه ذخیره داده در فرمت XML هم کمتر می‌شود .

فصل دوم

پیاده‌سازی SQL Server 2019

۲-۱ نیازمندی‌های SQL Server 2019

برای اینکه سرور SQL خود را راه‌اندازی کنیم، نیاز داریم بدانیم که این سرور چقدر از منابع سخت‌افزاری ما را قرار است مصرف کند، برای همین به جدول ۲-۱ توجه کنید.

جدول ۲-۱ نیازمندی سخت‌افزاری

سخت‌افزار	نیازمندی
هارد دیسک	نرم‌افزار SQL برای نصب اولیه نیازمند ٦ گیگابایت فضا در هارد دیسک است، اما برای یک محیط عملیاتی باید مشخص شود آن سازمان نیازمند چقدر از فضای هارد دیسک است؛ مثلاً برای محیطی که دارای ۱۰۰ کارمند است باید حداقل ۲۰۰ گیگ فضا برای آن در نظر گرفت. البته همه‌ی اینها مربوط به کاری است که انجام می‌دهید، توجه داشته باشید حتماً از RAIDبندی در سرور خود برای هارد دیسک استفاده کنید تا در صورت خرابی هاردها اطلاعات شما از دست نرود.
مانیتور	سرور SQL برای اجرا نیازمند رزولیشن Super-VGA (800x600) است.
اینترنت	برای آپدیت نرم‌افزار SQL و استفاده از بخش‌های دیگر نیازمند اینترنت هستیم.
رم	حداقل نیازمندی یک گیگابایت است، برای عملکرد بهتر باید حداقل ٤ گیگابایت رم برای این سرور در نظر بگیرید، البته در محیط عملیاتی واقعی امروزه، حداقل ۱۰ گیگابایت یک انتخاب ایده‌ال است.
سرعت پردازنده	حداقل پردازنده X64 با سرعت 1.4 GHz است، اما برای عملکرد بهتر باید از پردازنده ۲ گیگاهرتز به بالا استفاده شود.
نوع پردازنده	پردازنده‌های امروزی از نوع Intel و AMD به خوبی پاسخگوی نیاز این نرم‌افزار هستند.

نکته:

نصب SQL Server فقط در پردازنده‌های x64 پشتیبانی می‌شود و در پردازنده‌های x86 پشتیبانی نمی‌شود که باید به این نکته توجه کنید.

جدول ۲-۲ نشان می‌دهد که کدام نسخه های SQL Server 2019 با کدام نسخه‌های Windows سازگار است:

جدول ۲-۲ سیستم‌عامل

پشتیبانی سیستم‌عامل					
Express	Web	Standard	Developer	Enterprise	نسخه‌های مختلف SQL
Yes	Yes	Yes	Yes	Yes	**Windows Server 2019 Datacenter**
Yes	Yes	Yes	Yes	Yes	**Windows Server 2019 Standard**
Yes	Yes	Yes	Yes	Yes	**Windows Server 2019 Essentials**
Yes	Yes	Yes	Yes	Yes	**Windows Server 2016 Datacenter**
Yes	Yes	Yes	Yes	Yes	**Windows Server 2016 Standard**
Yes	Yes	Yes	Yes	Yes	**Windows Server 2016 Essentials**
Yes	No	Yes	Yes	No	**Windows 10 IoT Enterprise**
Yes	No	Yes	Yes	No	**Windows 10 Enterprise**
Yes	No	Yes	Yes	No	**Windows 10 Professional**
					Windows 10 Home

از نیازمندی‌های دیگر SQL Server می‌توان به NET Framework . اشاره کرد که باید نسخه‌ی مورد نظر آن را که در خود ویندوز ارائه می‌شود نصب کنید، البته این نرم‌افزار به‌صورت پیش‌فرض بر روی سرور نصب خواهد شد.

توجه داشته باشید در هنگام نصب SQL Server اجزای زیر بر روی سرور نصب خواهد شد:

- SQL Server Native Client
- SQL Server Setup support files

اگر بخواهید SQL را به همراه اجزای کامل آن بر روی سرور نصب کنید، هرکدام از اجزا طبق جدول ۳-۲ نیازمند فضای موردنیاز هستند.

توجه داشته باشید که این مورد را در موقع نصب SQL انتخاب خواهیم کرد.

جدول ۳-۲ اجزای SQL

ویژگی مورد نظر	مقدار فضای موردنیاز
Database Engine and data files, Replication, Full-Text Search, and Data Quality Services	1480 MB
Database Engine (as above) with R Services (In-Database)	2744 MB
Database Engine (as above) with PolyBase Query Service for External Data	4194 MB
Analysis Services and data files	698 MB
Reporting Services	967 MB
Microsoft R Server (Standalone)	280 MB
Reporting Services - SharePoint	1203 MB
Reporting Services Add-in for SharePoint Products	325 MB
Data Quality Client	121 MB
Client Tools Connectivity	328 MB
Integration Services	306 MB
Client Components (other than SQL Server Books Online components and Integration Services tools)	445 MB
Master Data Services	280 MB
SQL Server Books Online Components to view and manage help content*	27 MB
All Features	8030 MB

نکته:

SQL Server از دیسک با سکتورهای ۵۱۲ بایت تا ٤ کیلوبایت پشتیبانی می‌کند و اگر بخواهید از سکتورهای بالای ٤ کیلوبایت استفاده کنید، مطمئناً با خطا روبرو خواهید شد، برای دریافت اطلاعات بیشتر می‌توانید از لینک زیر استفاده کنید.

https://support.microsoft.com/en-us/topic/hard-disk-drive-sector-size-support-boundaries-in-sql-server-4d5b73fa-7dc4-1d8a-2735-556e6b60d046

SQL Server برای ذخیره‌ی اطلاعات از منابع ذخیره‌سازی زیر می‌تواند استفاده کند:

۱- حافظه داخلی سرور یا همان هارددیسک محلی.

۲- محل ذخیره‌سازی به اشتراک گذاشته شده در شبکه.

۳- SQL Server failover cluster.

٤- Storage Spaces Direct (S2D) (تکنولوژی شبیه به RAID که اطلاعات دیتابیس در چند سرور قابل‌دسترس است، این ویژگی همان Fault Tolerance است).

۵- فضای ذخیره‌سازی SMB (می‌توانید از یک Windows Server به‌عنوان FileServer برای انکار استفاده کنید).

به دلایل امنیتی که مایکروسافت اعلام کرده، بهتر است که SQL Server را بر روی Domain Controller نصب نکنید، چون موارد زیر را در پی خواهد داشت:

۱- نمی‌توانید سرویس‌های SQL Server را در یک Domain Controller تحت اکانت local service اجرا کنید که بسیار اذیت‌کننده خواهد بود.

۲- پس از نصب SQL Server بر روی سیستم مورد نظر، نمی‌توانید سیستم مورد نظر را از یک عضو دامنه به یک کنترل‌کننده‌ی دامنه تغییر دهید؛ قبل از تغییر سیستم میزبان به یک کنترل‌کننده‌ی دامنه، باید SQL Server را حذف نصب کنید.

۳- پس از نصب SQL Server بر روی رایانه، نمی‌توانید رایانه را از یک کنترل‌کننده‌ی دامنه به یک عضو دامنه تغییر دهید. قبل از تغییر رایانه میزبان به عضو دامنه، باید SQL Server را حذف و بعد نصب کنید.

٤- SQL Server failover cluster در یک دومین کنترل خواندنی پشتیبانی نمی‌شوند.

۵- SQL Server در یک کنترل‌کننده‌ی دامنه فقط خواندنی پشتیبانی نمی‌شود. SQL Server Setup نمی‌تواند گروه‌های امنیتی یا حساب‌های ارائه‌دهنده‌ی خدمات SQL Server را در یک کنترل‌کننده‌ی دامنه‌ی فقط خواندنی ایجاد کند. در این سناریو، نصب ناموفق است.

٦- یک نمونه خوشه‌ی شکست‌خورنده‌ی SQL Server در محیطی که فقط یک کنترل‌کننده‌ی دامنه‌ی فقط خواندنی قابل‌دسترسی است پشتیبانی نمی‌شود.

۲-۲ نصب و راه‌اندازی SQL Server 2019 در ویندوز

قبل از نصب SQL Server 2019 بهتر است یک موضوع بسیار مهم را بررسی کنیم، همان‌طور که می‌دانید دو نوع سیستم داریم:

۱- فیزیکی

۲- مجازی

به طور معمول یک سیستم فیزیکی با رم و هارد و فضای ذخیره‌سازی مناسب را برای نصب SQL Server در نظر می‌گیرید و فکر می‌کنید بهترین عملکرد را ارائه دادید، اما اگر در این بین، اطلاعات شما از دست برود و هارددیسک شما خراب شود، آن‌وقت چه کاری باید انجام دهید، آیا به نظر شما استفاده از سیستم فیزیکی برای استفاده‌ی مستقیم از SQL Server کار درستی خواهد بود؟

هرچند می‌توانید با روش‌هایی مانند Raid بندی و روش‌های دیگر جلوی ازبین‌رفتن اطلاعات را بگیرید، اما بهترین کار این است که از مجازی‌سازی استفاده کنیم و ازروی ماشینی که ایجاد می‌کنیم، پشتیبان تهیه کنیم تا در موقع ازدست‌رفتن سرور در سریع‌ترین زمان ممکن بتوانیم آن را برگردانیم، البته روش‌های پشتیبان‌گیری در SQL یک موضوع مفصل خواهد بود که در فصل مربوط به پشتیبان‌گیری به‌صورت کامل به آن خواهیم پرداخت.

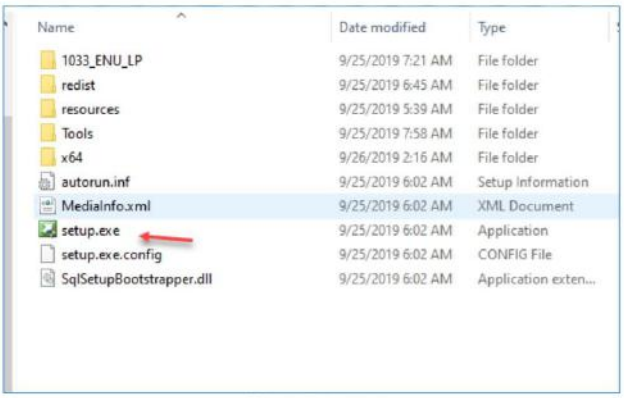
ما برای این کتاب سرور مجازی را انتخاب می‌کنیم، بهترین عملکرد در مجازی‌سازی را شرکت VMware ارائه می‌دهد و شما می‌توانید ماشین مورد نظر خود را توسط نرم‌افزار VMware Workstation و یا با سیستم‌عامل ESXi ایجاد کنید که کار با این نرم‌افزارها را در کتاب VMware Systems به طور کامل توضیح دادیم و می‌توانید نسخه‌ی الکترونیکی را از سایت بنده دریافت کنید.

برای این کتاب ما از یک سرور ESXi استفاده کردیم که روی آن یک ماشین مجازی ایجاد کردیم و سخت‌افزار مناسب را برای آن در نظر گرفتیم، بعد از این کار بر روی آن ویندوز سرور ۲۰۱۹ نصب کردیم تا همه چیز برای نصب نرم‌افزار SQL آماده باشد.

برای اینکه نرم‌افزار SQL Server 2019 را دانلود کنید می‌توانید به‌صورت مستقیم از سایت مایکروسافت دانلود کنید و یا اینکه آن را از سایت‌های ایرانی دانلود کنید که آدرس آن هم در زیر قرار دارد:

https://soft98.ir/software/programming/3594-microsoft-sql-server-all-2017-full-1.html

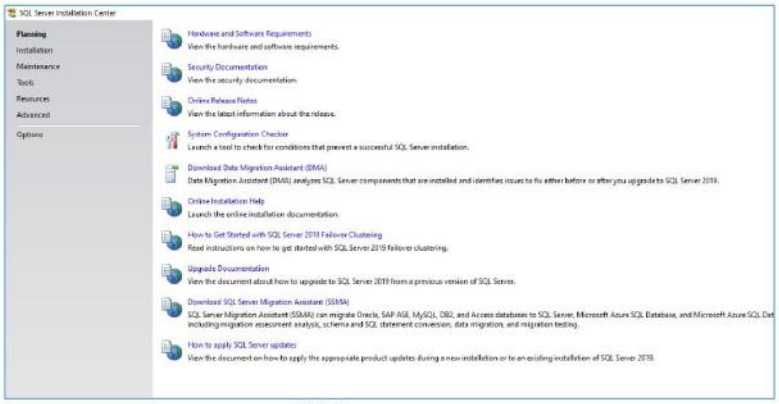
بعد از دانلود فایل مورد نظر، به‌مانند شکل ۱-۲ بر روی فایل Setup.exe دو بار کلیک کنید.



شکل ۲-۱ فایل *Setup*

اولین صفحه‌ای که به شما نمایش داده می‌شود، شکل ۲-۲ است که گزینه‌های مختلفی را به شما ارائه می‌دهد، مثلاً در گزینه‌ی اول اطلاعاتی را در مورد سخت‌افزار و نرم‌افزار موردنیاز برای نصب SQL Server توضیح می‌دهد و همچنین در گزینه‌های دیگر می‌توانید موارد دیگر را هم بررسی کنید.

برای اینکه متوجه شویم که سیستمی که در حال نصب SQL بر روی آن هستیم آیا مناسب است یا نه باید در شکل ۲-۲ بر روی لینک System Configuration Checker کلیک کنید، تا به‌مانند شکل ۳-۲ همه‌ی گزینه‌ها اوکی باشد.

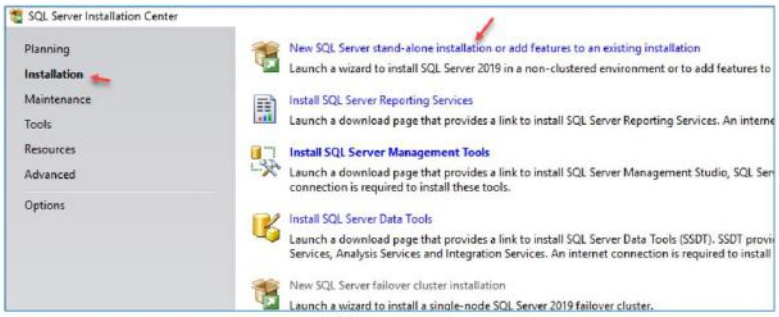


شکل ۲-۲ نصب *SQL Server*

Result	Rule	Status
✓	Setup administrator	Passed
✓	Restart computer	Passed
✓	Windows Management Instrumentation (WMI) service	Passed
✓	Consistency validation for SQL Server registry keys	Passed
✓	Long path names to files on SQL Server installation media	Passed
✓	SQL Server Setup Product Incompatibility	Passed
✓	Computer domain controller	Passed
✓	Edition WOW64 platform	Passed

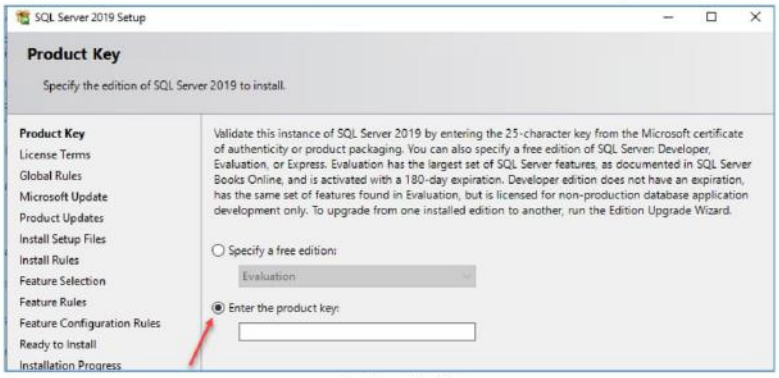
شکل ۲-۳ بررسی نصب *SQL*

برای اینکه نصب SQL Server را آغاز کنید، باید به‌مانند شکل ٤-٢ وارد Installation شوید و بر روی New SQL Server stand-alone installation or add fearure to an existing installation کلیک کنید.



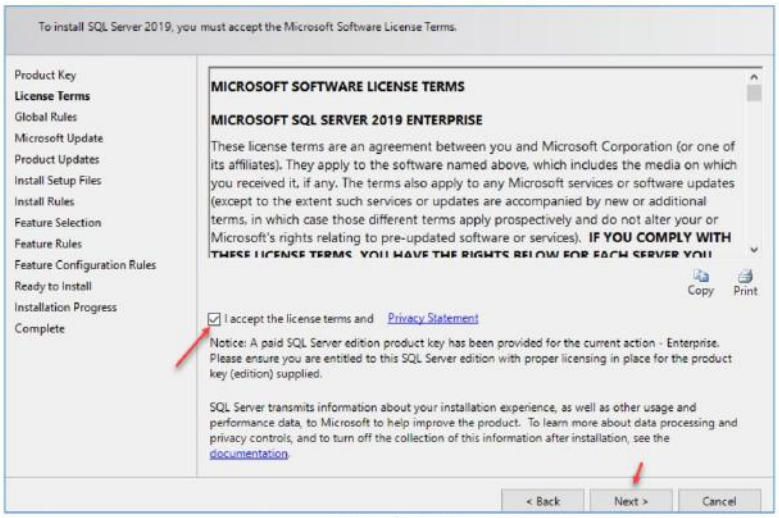
شکل ۴-۲ نصب *SQL*

در شکل ٥-٢ باید سریال نرم‌افزار را وارد و بر روی Next کلیک کنید.



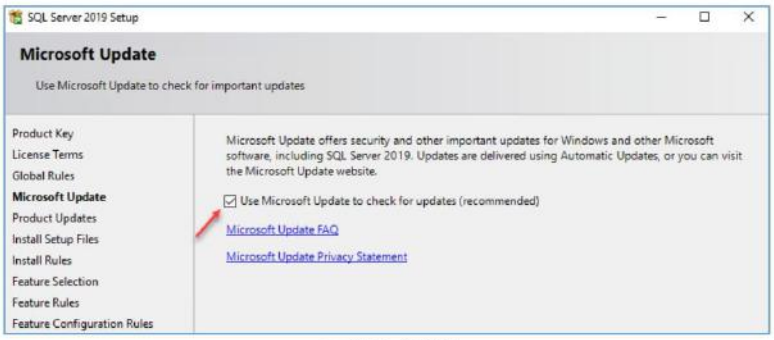
شکل ٥-٢ سریال برنامه

در شکل ٦-٢ باید توافقنامه‌ی استفاده از این نرم‌افزار را مطالعه و درصورتی‌که آن را قبول دارید، تیک گزینه‌ی I accept the license terms را انتخاب و بر روی Next کلیک کنید.



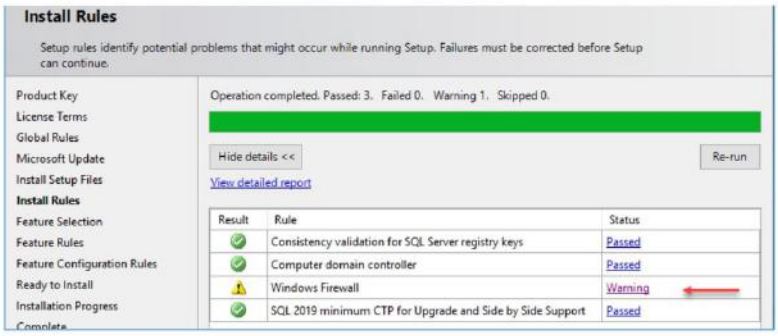
شکل ۲-۶ تأیید توافقنامه

به‌مانند شکل ۲-۷ می‌توانید با انتخاب تیک گزینه‌ی Use Microsoft Updates To ... آخرین آپدیت‌های نرم‌افزار SQL را از سایت مایکروسافت دریافت کنید.



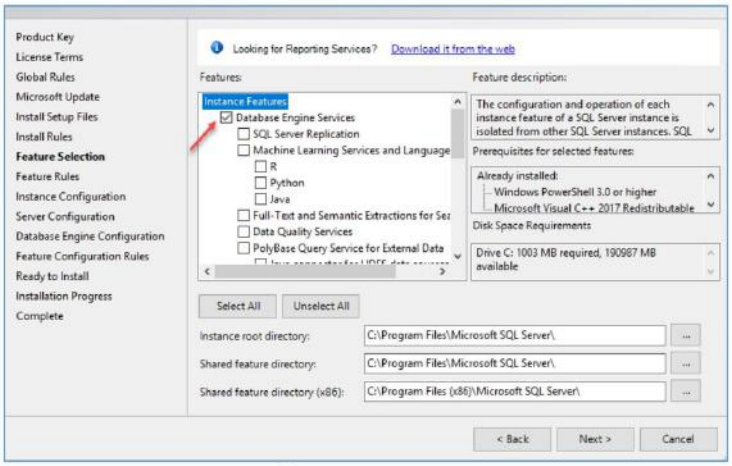
شکل ۲-۷ دریافت آپدیت

در شکل ۲-۸ بررسی اولیه انجام می‌شود و در این قسمت همه چیز باید Passed باشد، اگر به شکل ۲-۸ توجه کنید، متوجه خواهید شد که قسمت Firewall با یک اخطار روبرو شده است و به این موضوع اشاره دارد که برای استفاده از SQL در شبکه باید پورت‌های مورد نظر آن در فایروال باز باشد که با یاری خدا در ادامه این کار را انجام خواهیم داد.



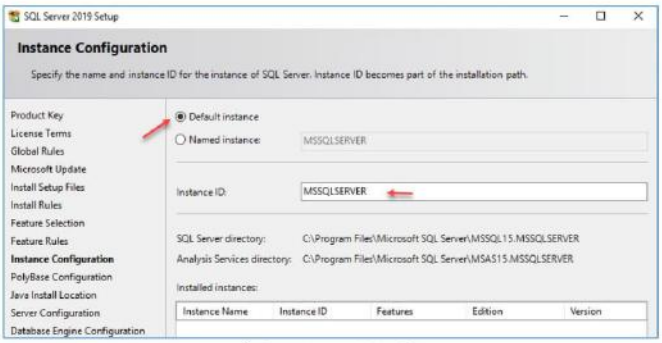
شکل ۸-۲ بررسی نیازمندی‌های اولیه

در شکل ۹-۲ باید تیک گزینه‌ی Database Engine را انتخاب کنید و همچنین می‌توانید مسیر نصب را تغییر دهید.



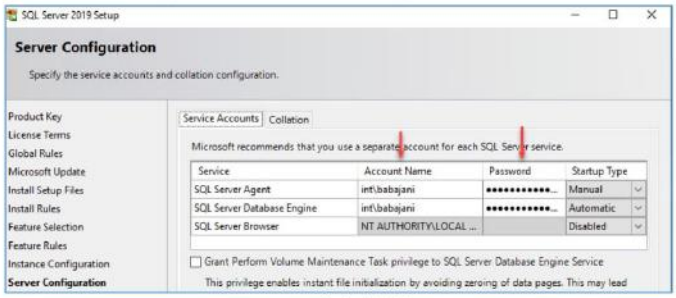
شکل ۹- ۲ انتخاب Feature

در شکل ۱۰-۲ باید Instance را مشخص کنید، Instance را به‌عنوان یک ظرف در نظر بگیرید که داخل آن می‌توانید دیتابیس خود را قرار دهید و نرم‌افزارهای خاص خود را داشته باشید، مثلاً اطلاعاتی که در Instance با نام DB1 قرار دارد با اطلاعاتی که در Instance با نام DB2 قرار دارد متفاوت است، اصولاً Instance را به‌عنوان یک مزرعه‌ی جدا می‌شناسند که داخل آن می‌توانید دیتابیس‌ها و سرویس‌های خود را داشته باشید، در شکل ۱۰-۲ Instance با نام پیش‌فرض MSSQLSERVER قرار دارد که سرویس را بر روی آن فعال می‌کنیم.



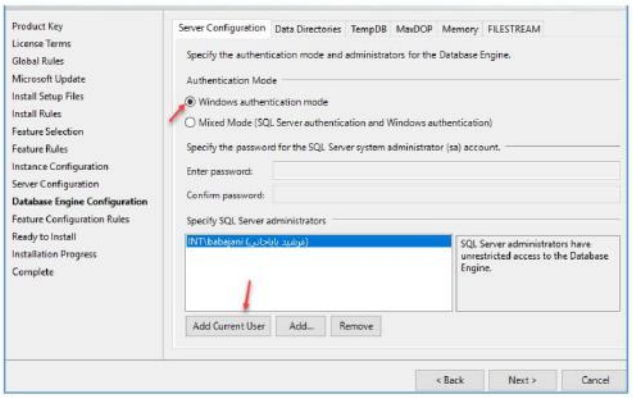
شکل ۱۰-۲ بررسی *Instance*

در شکل ۱۱-۲ باید یک کاربر را برای اجرای سرویس‌های SQL در نظر بگیرید، توجه داشته باشید که آن کاربر دسترسی لازم در شبکه را داشته باشد، برای همین منظور کاربر babajani که یک کاربر تحت دومین است با دسترسی کامل به شبکه وارد شده است و با همین کاربر هم باید در ادامه وارد SQL شویم.



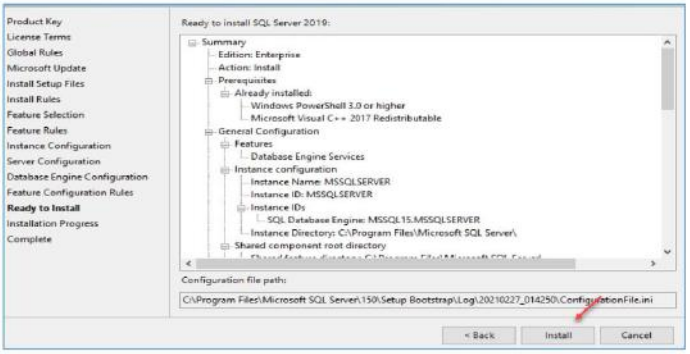
شکل ۱۱-۲ تنظیم سرور

در شکل ۱۲-۲ باید مشخص کنید که چه کاربرانی دسترسی به Database داشته باشند که با کلیک بر روی Add Current User می‌توانید با کاربری که در حال نصب SQL هستید، آن را به لیست اضافه کنید و یا اینکه هر کاربر دیگری که مورد نظر شماست با کلیک بر روی Add آن را به لیست اضافه کنید، توجه داشته باشید دو حالت احراز هویت وجود دارد که به‌صورت پیش‌فرض Windows aythentication mode انتخاب شده است و اگر بخواهید کاربر sa که کاربر پیش‌فرض در SQL است باید در این قسمت گزینه Mixed Mode را انتخاب و یک رمز عبور برای آن در نظر بگیرید.



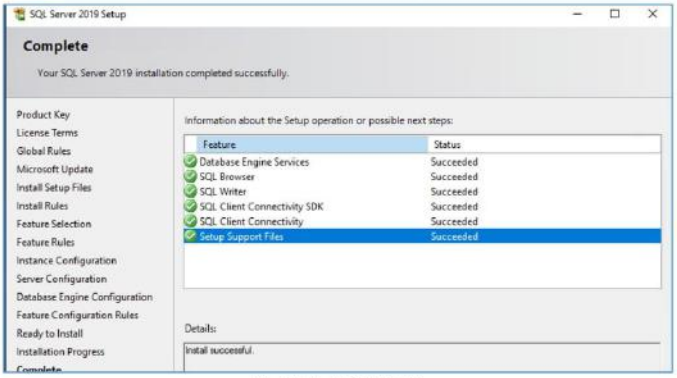
شکل ۱۲-۲ تنظیم دسترسی به دیتابیس

در شکل ۱۳-۲ بر روی Install کلیک کنید تا کار نصب آغاز شود.



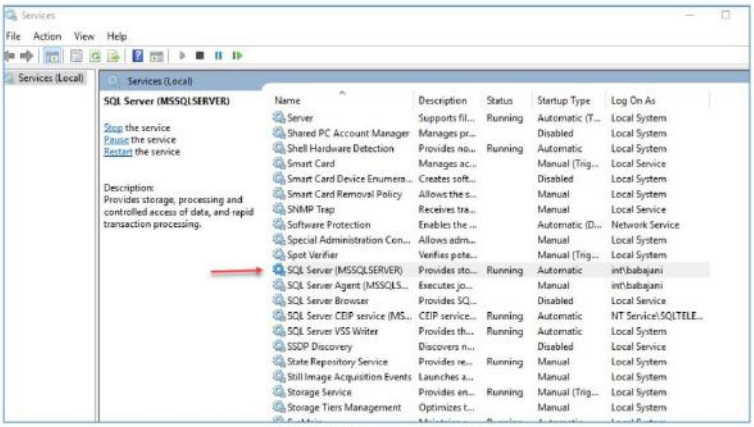
شکل ۱۳-۲ *Install SQL*

همان‌طور که در شکل ۱۴-۲ مشاهده می‌کنید نرم‌افزار SQL Server به‌درستی بر روی سرور نصب شده است.



شکل ۱۴-۲ *Finish Install*

اگر بعد از نصب به‌مانند شکل ۱۵-۲ وارد Services شوید، مشاهده خواهید کرد که سرویس SQL Server به‌درستی در حال اجرا است.

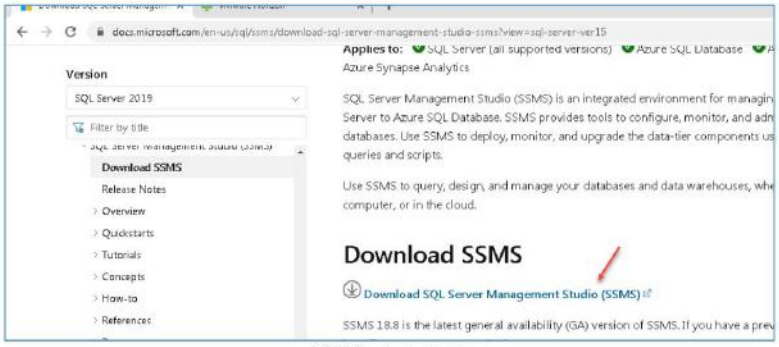


شکل ۱۵-۲ سرویس *SQL Server*

بعد از نصب نرم‌افزار SQL Server نیاز به یک نرم‌افزاری داریم تا بتوانیم SQL را مدیریت کنیم، یعنی کاربر جدید تعریف کنیم، دسترسی آن را مشخص کنیم، دیتابیس جدید و موارد دیگر؛ همه‌ی این کارها توسط نرم‌افزار SQL Server Management Tools یا به‌اختصار SSMS انجام خواهد گرفت، در نسخه‌های جدید این نرم‌افزار به‌صورت جداگانه ارائه می‌شود و می‌توانید از طریق لینک زیر آن را دانلود کنید:

https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15

بعد از ورود به صفحه شکل ۱۶-۲ بر روی لینک دانلود کلیک کنید تا نرم‌افزار مورد نظر دانلود شود و بعد از دانلود آن را بر روی سرور و یا هر سیستمی که می‌خواهید نصب کنید.



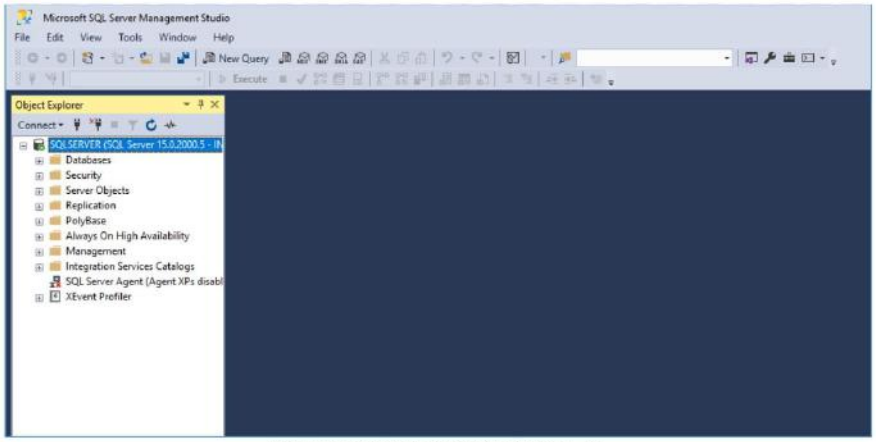
شکل ۱۶ -۲ دانلود *SSMS*

بعد از نصب و از طریق منوی Start نرم‌افزار SSMS را اجرا کنید که به‌مانند شکل ۱۷-۲ صفحه مورد نظر برای شما ظاهر خواهد شد. در این صفحه شما باید در قسمت Server Type، گزینه‌ی Database Engine را انتخاب کنید که شامل

دیتابیس شما خواهد بود؛ در قسمت Server Name باید نام سرور SQL خود را وارد کنید، توجه داشته باشید اگر داخل سرور هستید می‌توانید از نام localhost نیز استفاده کنید، اما برای دسترسی از بیرون باید از نام سرور استفاده کنید. در قسمت Authentication نیز اگر بر روی Windows Authentication قرار دهید با همان نام کاربری که Login کردید وارد خواهد شد که مسلماً باید دسترسی لازم را داشته باشد و یا اینکه می‌توانید از SQL Server Authentication استفاده کنید.

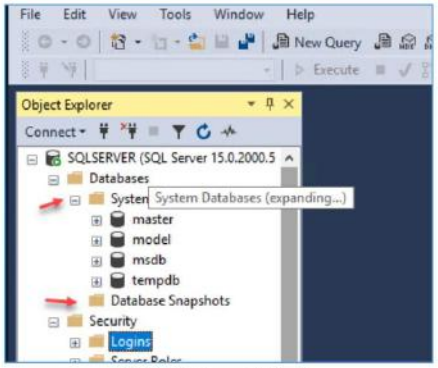
شکل ۱۷-۲ ورود به *SQL Server*

در شکل ۱۸-۲ یک نمای کلی از نرم‌افزار SSMS را که به Database Engine متصل شده است را مشاهده می‌کنید، در قسمت Databases می‌توانید دیتابیس‌های خود و دیتابیس‌هایی که توسط نرم‌افزارهای دیگر به‌صورت اتوماتیک ایجاد می‌شود را مشاهده کنید. در قسمت Security باید کاربران خود را معرفی و دسترسی‌های لازم برای آنها را مشخص کنید؛ در قسمت Server Objects یک سری اشیا یا همان Object وجود دارد که برای مانیتور کردن عملکرد سرور خواهد بود. در قسمت Replication نیز یک سری ابزار وجود دارد تا بتوانید دیتابیس‌ها را از یک پایگاه‌داده به یک پایگاه‌داده به‌صورت درست و امن انتقال دهید، PolyBase نیز یک ابزار برای انتقال اطلاعات از یک پایگاه‌داده متفاوت، مانند Oracle به SQL است؛ گزینه‌ی Always On High Availability برای ایجاد یک گروه از سرورها برای پایدار نگه‌داشتن دیتابیس‌ها و سرورها است که یک روش جدید در میان روش‌های دیگر است. در قسمت Managemnt یک سری ابزار وجود دارد، مانند پشتیبان‌گیری از دیتابیس‌ها، LOG گیری و موارد دیگر که در مدیریت SQL بسیار کمک‌کننده خواهد بود؛ قسمت Inetgaration Service Catalog نیز برای یکپارچه‌سازی داده‌ها در سازمان شما است که با یاری خدا همه‌ی این گزینه‌ها را در ادامه توضیح خواهیم داد و در آخر نیز سرویس SQL Agent برای انجام پشتیبان‌گیری و کارهایی دیگر موردنیاز است.



شکل ۱۸-۲ *SQL Server Management Studio*

اولین قسمتی که بررسی می‌کنیم Databases است، به‌مانند شکل ۱۹-۲، بعد از بازکردن قسمت Databases، دو گزینه را مشاهده می‌کنید، یکی Systems و دیگری Database Snapshots است؛ در قسمت Systems، چهار دیتابیس را مشاهده می‌کنید که به‌صورت پیش‌فرض ایجاد می‌شوند.



شکل ۱۹-۲ بررسی *Database*

جدول ۴-۲ دیتابیس‌های سیستم

دیتابیس سیستم	توضیحات
master	پایگاه‌داده Master به‌عنوان قلب تپنده SQL است و اگر از دست برود با مشکل مواجه خواهید شد، دیتابیس Master شامل اطلاعاتی حیاتی زیر است: • زمانی که یک کاربر ایجاد می‌کنید، ID آن در این دیتابیس قرار می‌گیرد. • تمام رویدادها یا همان Logها در این دیتابیس قرار می‌گیرد. • نام و اطلاعات مربوط به پایگاه داده‌ها.

	• خطاهای سیستم و تمام پیام‌های موجود در سرور. • مقداردهی اولیه‌ی SQL Server . • پایگاه داده‌های محلی. • جدول‌های خاص پایگاه داده‌ها.
tempdb	همان‌طور که از اسم آن مشخص است، یک پایگاه‌داده موقت است، زمانی که شما Server SQL ۲۰۱۹ را اجرا می‌کنید، اطلاعات موقت در این پایگاه‌داده قرار می‌گیرد، مثالی که در این رابطه می‌توان زد، مانند RAM سیستم شما که زمانی نرم‌افزاری را اجرا می‌کنید، اطلاعات به‌صورت موقت در این حافظه قرار می‌گیرد و بعد از بستن نرم‌افزار، اطلاعات نیز از حافظه پاک می‌شوند. زمانی که یک دستور را در SQL اجرا می‌کنید، اطلاعات این دستور به‌صورت موقت در این پایگاه‌داده قرار می‌گیرد و پردازش می‌شود و طول عمر آن به کار کاربر مورد نظر برمی‌گردد.
model	این پایگاه‌داده به‌عنوان یک الگو در نظر گرفته می‌شود، یعنی اینکه یک سری استانداردهایی در آن تعریف شده است که همه‌ی پایگاه داده‌ها از آن استاندارد پیروی می‌کنند. مجموعه‌های از پیش تعیین‌شده در این پایگاه وجود دارد که برای ساخت پایگاه داده‌های دیگر به کار می‌رود، مانند حجم پایگاه داده‌ها، اندازه‌ی جدول‌ها و موارد دیگر.
Resource	یک پایگاه‌داده‌ی فقط خواندنی که اطلاعات System Object در آن نگهداری می‌شود، البته در لیست ظاهری وجود ندارد.
msdb	در این پایگاه‌داده یک سری کارهای از پیش تعیین‌شده قرار دارد این کارها می‌تواند پشتیبان‌گیری و یا بازگردانی اطلاعات باشد که این کار بدون دخالت کسی و به‌صورت خودکار انجام می‌شود.

در بالای نرم‌افزار چندین منو قرار دارد که با هم آنها را بررسی می‌کنیم:

در منوی فایل می‌توانید با کلیک بر روی Connect Object Explorer به دیتابیس جدید خود متصل شوید و یا با انتخاب گزینه‌ی Disconnect، ارتباط را قطع کنید؛ گزینه‌های دیگر برای ایجاد پروژه‌ی جدید و یا بازکردن پروژه‌های قبلی است؛ در قسمت Recent Projects and Solutuin می‌توانید آخرین پروژه‌هایی را که باز کردید را مشاهده کنید.

در منوی Edit می‌توانید اطلاعات را جستجو، کپی، حذف و... کنید.

در منوی View، گزینه‌ی Explorer Object ابزاری برای نمایش کلی دیتابیس‌ها، سرویس‌ها و... است که اگر به نرم‌افزار توجه کنید در سمت چپ، این ابزار را می‌توانید ببینید؛ گزینه‌ی Details Explorer Object ابزاری است زیرمجموعه‌ی ابزار Explorer Object که اطلاعات داخلی آن را نمایش می‌دهد.

گزینه‌ی Explorer Solution، ابزاری که در سمت راست برنامه ظاهر می‌شود و برای نمایش اطلاعات پروژه‌ی شما ایجاد شده است؛ این پروژه‌ها می‌توانند پروژه‌های در Visual Studio شما باشد.

گزینه‌ی Window Bookmark، این ابزار مفید برای ایجاد Bookmarkهایی در کد است که شما را سریع به کد مورد نظر در پروژه می‌رساند.

گزینه‌ی Explorer Utility، ابزاری برای مدیریت پایگاه‌داده‌ها و نظارت کلی بر روی آنها در داخل سازمان و یا سازمان‌هایی در فواصل دورتر از آن است.

منوی Debug، این منو برای کنترل پروژه است و برای بررسی و اشکال‌زدایی پروژه کاربرد دارد که در خلال کار بیشتر با آن آشنا خواهیم شد.

منوی Tools، با استفاده از Profiler Server SQL می‌توانیم تمام Logهای مربوط به پروژه را در مسیر مشخصی ذخیره کنیم تا بتوانیم در صورت مواجه‌شدن با مشکل آنها را بررسی کنیم.

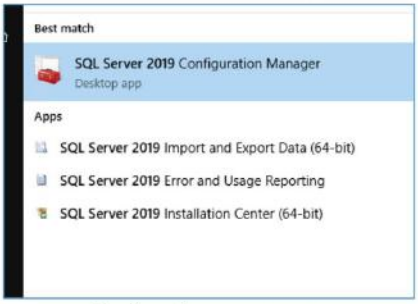
اگر در منوی فایل بر روی Options کلیک کنید، می‌توانید تنظیمات کلی نرم‌افزار SQL را مشاهده و آنها را تغییر دهید.

۳-۲ دسترسی از طریق شبکه به SQL Server

شاید شما در سازمان خود احتیاج داشته باشید که هم‌زمان چندین کاربر به SQL سرور شما متصل شوند و این کار نیاز به فعال‌کردن دسترسی از طریق شبکه به SQL است.

شما می‌توانید سرور SQL خود را عضو شبکه دومین خود کنید و به کاربرانی که نیاز به دسترسی به پایگاه‌داده دارند، دسترسی لازم را اعمال کنید.

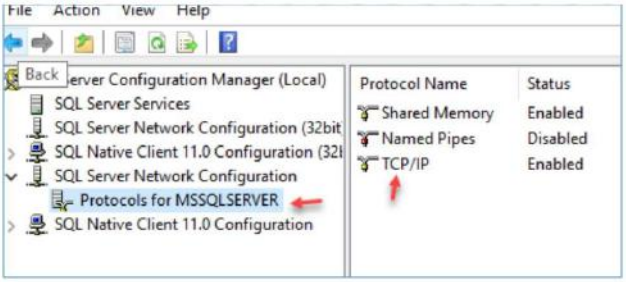
برای شروع باید وارد سرور SQL خود شوید و گزینه‌ی Configuration را در جستجو وارد کنید و ابزار SQL Server 2019 Configuration Manager را به‌مانند شکل ۲۰-۲ اجرا کنید.



شکل ۲۰-۲ سرویس *Configuration*

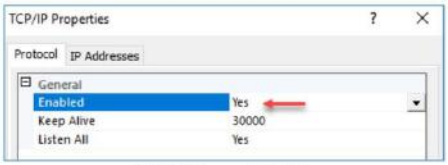
در شکل ۲۱-۲، از سمت چپ وارد SQL Server network Configuration شوید و بر روی گزینه‌ی مورد نظر کلیک کنید تا لیست آن باز شود.

در لیست مورد نظر بر روی TCP/IP دو بار کلیک کنید.



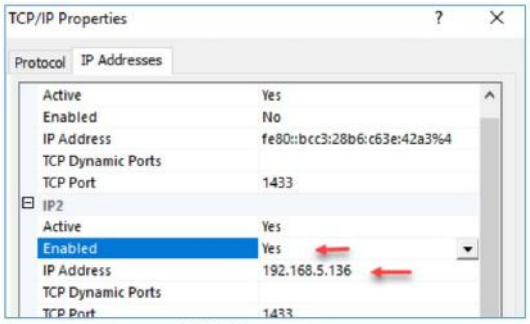
شکل ۲۱-۲ تنظیم پروتکل دسترسی

در شکل ۲۲-۲ و در تب Protocol، گزینه‌ی Enabled را در حالت Yes قرار دهید و وارد تب IP Addresses شوید.



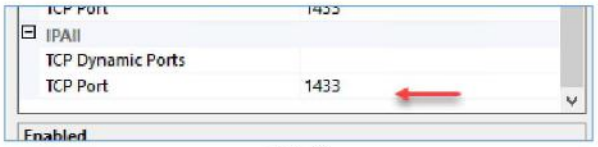
شکل ۲۲-۲ بررسی *TCP/IP*

به‌مانند شکل ۲۳-۲، در تب IP addresses و در قسمتی که IP سرور شما مشخص شده است باید دو گزینه‌ی اول را در حالت Yes قرار دهید و در قسمت TCP Port، شماره‌ی پورت ۱٤۳۳ را وارد کنید، زمانی‌که این پورت را وارد می‌کنید باید آن را در Firewall سیستم خود باز کنید تا کاربران بتوانند از طریق شبکه به SQL دسترسی داشته باشند. (در صورت خاموش بودن Firewall نیاز به این کار نیست).



شکل ۲۳-۲ تنظیم *TCP/IP*

برای اینکه تنظیمات پورت بر روی همه‌ی گزینه‌ها انجام شود، بهتر است در پایین شکل ۲۳-۲ گزینه‌ی TCP Port را به‌مانند شکل ۲٤-۲ بر روی 1433 قرار دهید.



شکل ۲-۲۴

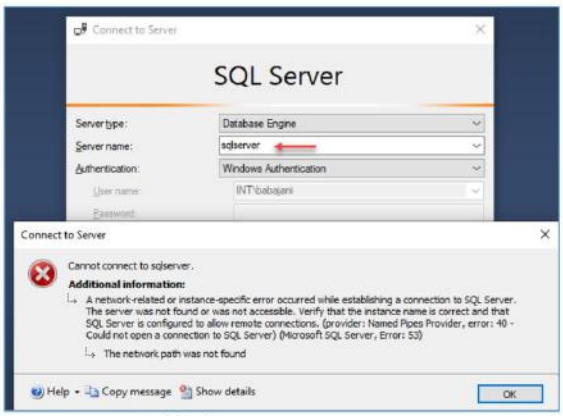
برای تست عملکرد ارتباطی با SQL از طریق شبکه، در یکی از کلاینت‌هایی که عضو شبکه است نرم‌افزار SSMS را نصب و اجرا می‌کنیم.

همان‌طور که در شکل ۲-۲۵ مشاهده می‌کنید در قسمت Server Name، نام سرور SQL خود را وارد کردیم و بعد از کلیک بر روی Connect با خطای مورد نظر روبرو شدیم؛ این خطا چند دلیل می‌تواند داشته باشد:

۱- شبکه در دسترس نباشد.

۲- Firewall مربوط به سرور SQL روشن باشد و جلوی ورود را بگیرد.

۳- تنظیمات سرور SQL مشکلی داشته باشد.



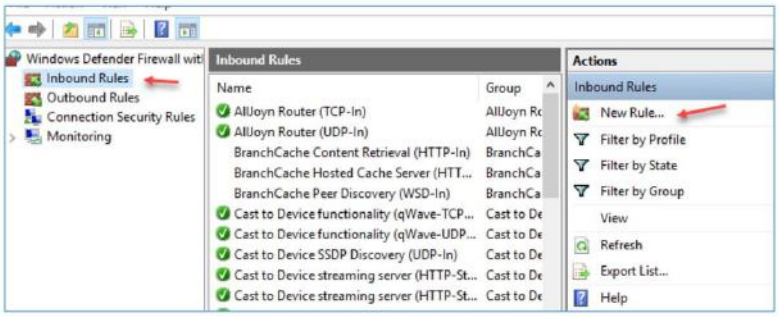
شکل ۲-۲۵ خطای ورود به *SQL Server*

در گزینه‌ی اول که باید برای تست یکی از سرورهای دیگر در شبکه را تست بگیرید، توجه کنید که آن را می‌بینید یا نه، اگر اوکی بود که هیچ، اگر نه که باید شبکه را به‌صورت فیزیکی بررسی کنید، شاید کابل یا کارت شبکه ایراد دارد و یا مشکل نرم‌افزاری است.

در گزینه‌ی دوم باید وارد سرور SQL شوید و سرویس Firewall را اجرا کنید، برای این کار باید به‌مانند شکل ۲-۲۶ بر روی Advanced settings کلیک کنید.

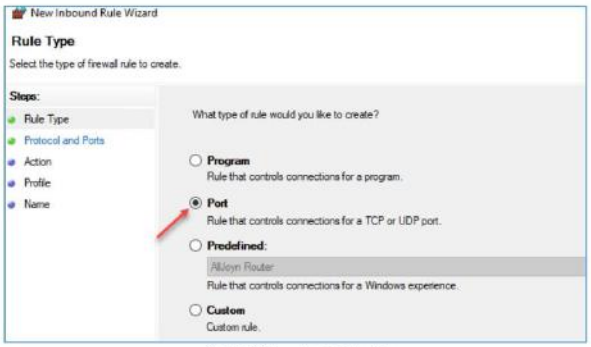
شکل ۲۶-۲ تنظیم فایروال

در شکل ۲۷-۲ برای اینکه به ترافیک ورودی به سرور SQL دسترسی لازم دهید باید بر روی Inbound Rules کلیک کنید و در صفحه‌ی باز شده بر روی New Rule کلیک کنید.



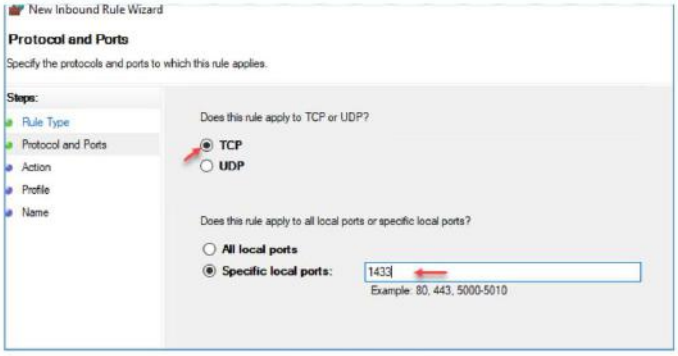
شکل ۲۷-۲ تنظیم فایروال

در شکل ۲۸-۲ باید گزینه‌ی Port را انتخاب و بر روی Next کلیک کنید.



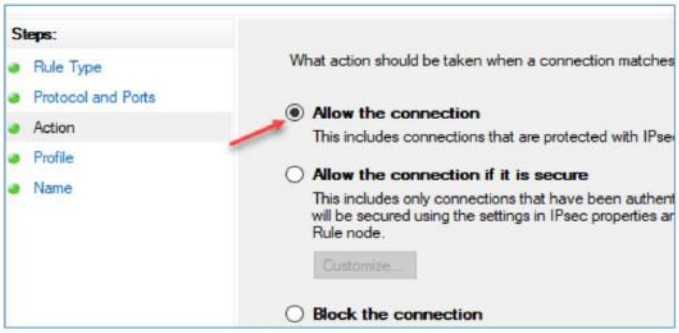
شکل ۲۸-۲ تنظیم *Port SQL*

در شکل ۲۹-۲ باید گزینه‌ی TCP را انتخاب کنید و در قسمت Spicific Local ports، پورت ۱۴۳۳ مربوط به SQL را وارد کنید تا درخواست‌هایی که از بیرون به داخل سرور با این پورت وارد می‌شود، مجوز دسترسی داشته باشد.



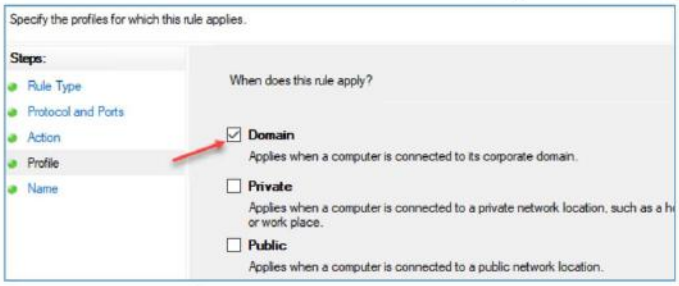
شکل ۲۹-۲ تنظیم *Port SQL*

در شکل ۳۰-۲ باید گزینه‌ی Allow the connection را انتخاب کنید.



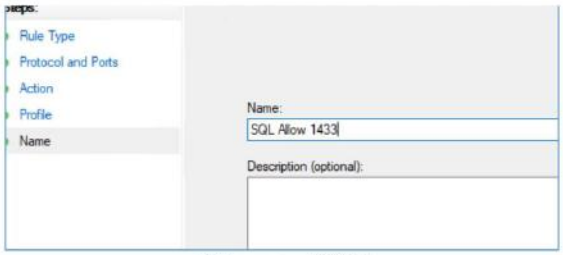
شکل ۳۰-۲ دسترسی به *Port*

در شکل ۳۱-۲ باید مشخص کنید که این دسترسی در چه قسمتی اعمال شود، اگر چنانچه کلاینت شما در منطقه‌ی دومین قرار دارد باید فقط گزینه‌ی دومین را انتخاب کنید، اما اگر کلاینت در منطقه‌ی دیگر، مثلاً در Workgroup قرار داشته باشد، نمی‌تواند به SQL متصل شود.



شکل ۳۱-۲ دسترسی به *Port*

یک نام به‌مانند شکل ۳۲-۲ وارد کنید و بر روی Finish کلیک کنید تا Rule مورد نظر ایجاد شود.



شکل ۳۲-۲ دسترسی به *Port*

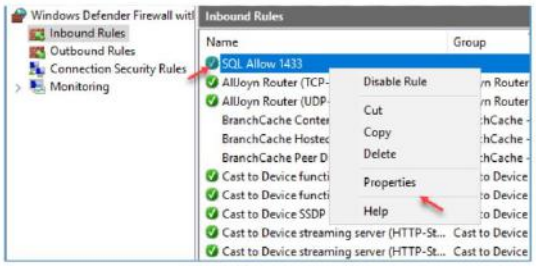
بعد از ایجاد Rule در فایروال سرور SQL باید از طریق کلاینت تست بگیریم تا ببینیم می‌توانیم به سرور SQL از طریق شبکه متصل بشویم یا نه؛ برای این کار SQL Server Management Studio را طبق شکل ۳۳-۲ اجرا کنید و در قسمت Server Name باید نام سرور SQL خود را وارد و بر روی Connect کلیک کنید.

شکل ۳۳-۲ ورود به *SQL Server*

اگر مشکلی در ارتباط شبکه و دومین شما وجود نداشته باشد به‌راحتی به سرور SQL متصل خواهید شد، اما اگر طبق شکل ۳٤-۲ با خطا روبرو شدید، باید دوباره وارد فایروال سرور SQL شوید و منطقه‌ی کاری را تغییر بدهید.

شکل ۳۴-۲ خطای ورود به *SQL*

طبق شکل ۳۵-۲ وارد فایروال SQL شوید و بر روی Rule مورد نظر خود کلیک راست و گزینه‌ی Properties را انتخاب کنید.

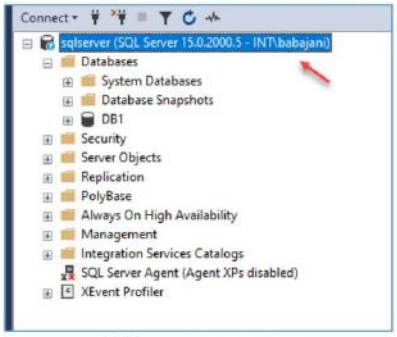


شکل ۲-۳۵ بررسی *Firewall*

در شکل ۲-۳۶ وارد تب Advanced شوید و تیک گزینه‌ی Private را انتخاب و بر روی OK کلیک کنید.

شکل ۲-۳۶ دسترسی *Private*

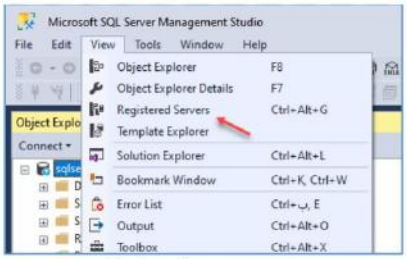
همان‌طور که در شکل ۲-۳۷ مشاهده می‌کنید به‌درستی توانستیم به سرور SQL از طریق شبکه متصل شویم.



شکل ۲-۳۷ متصل شدن به *SQL*

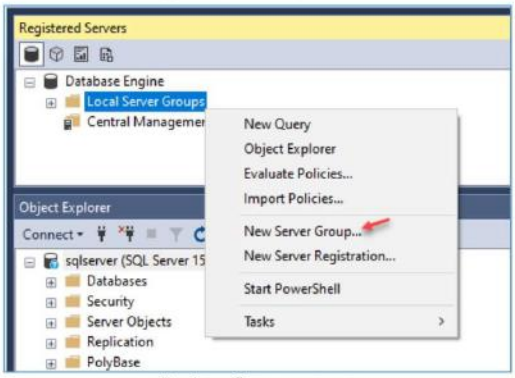
شاید در سازمان خود چندین سرور SQL داشته باشید و به همه‌ی آنها از طریق شبکه متصل می‌شوید، برای راحتی کار خود بهتر است یک گروه ایجاد کنید و همه‌ی آنها را در گروه مورد نظر خود قرار دهید.

برای این کار در نرم‌افزار SQL Management Studio به‌مانند شکل ۲-۳۸ وارد منوی View شوید و بر روی گزینه‌ی Registered Servers کلیک کنید.



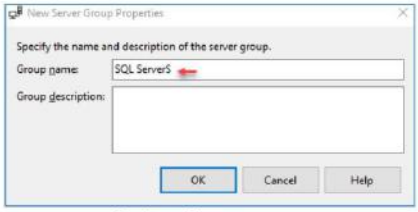
شکل ۲-۳۸ *Register Server*

در شکل ۲-۳۹ بر روی Local Server Groups کلیک راست کنید و گزینه‌ی New Server Group را انتخاب کنید.



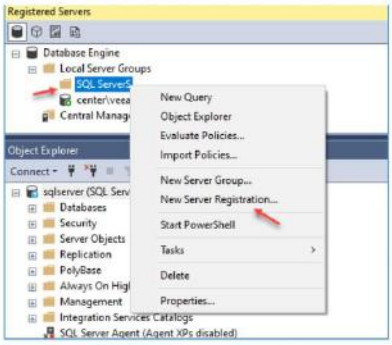
شکل ۲-۳۹ *Register Server*

در شکل ۲-۴۰ باید یک اسم برای گروه خود وارد کنید و بر روی OK کلیک کنید.



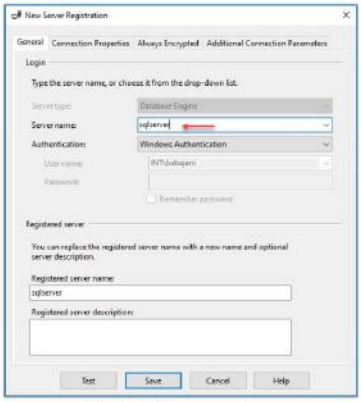
شکل ۲-۴۰ *Register Server*

بر روی نام گروهی که ایجاد کردید، طبق شکل ۲-۴۱ کلیک راست کنید و گزینه‌ی New Server Registration را انتخاب کنید.



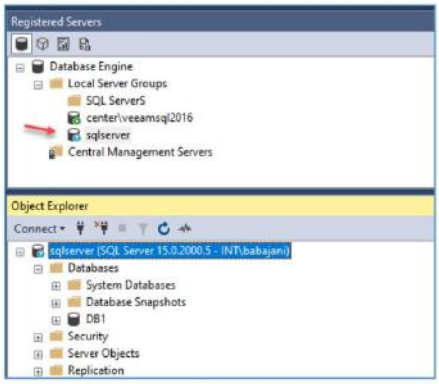
Register Server شکل ۲-۴۱

در شکل ۲-۴۲ باید نام سرور را در قسمت Server name وارد کنید و بر روی Save کلیک کنید.



Register Server شکل ۲-۴۲

همان‌طور که در شکل ۲-۴۳ مشاهده می‌کنید، سرور به لیست اضافه شده است و با کلیک بر روی آن به‌راحتی به سرور SQL متصل خواهید شد.



Register Server شکل ۲-۴۳

۲-٤ نصب و راه‌اندازی SQL Server 2019 در لینوکس Ubuntu

تا به اینجا توانستیم نرم‌افزار SQL را بر روی ویندوز نصب کنیم و آن را اجرا و از راه دور به آن متصل شویم، اما شرکت مایکروسافت در نسخه‌های جدید خود، توانایی نصب SQL را بر روی سیستم‌عامل لینوکس فعال کرده است که با هم در این قسمت نحوه‌ی راه‌اندازی آن را فرا خواهیم گرفت.

برای شروع کار باید یک نسخه از سیستم عامل لینوکس که واقعاً هم زیاد است را انتخاب کنید؛ برای این قسمت، Linux Ubuntu را انتخاب می‌کنیم. برای دانلود **Linux Ubuntu** می‌توانید از لینک زیر استفاده کنید.

https://ubuntu.com/download/desktop

بعد از دانلود لینوکس **Ubuntu** باید آن را پیاده‌سازی کنید، بهترین کار این است که از یک نرم‌افزار مجازی‌سازی، مانند VMware Workstation و یا برنامه‌ی دیگر برای این کار استفاده کنید تا سهولت دسترسی و عملکرد بهتری را شاهد باشید، اگر برنامه VMware Workstation را در دسترس ندارید می‌توانید از لینک زیر آن را دانلود کنید:

https://soft98.ir/os/virtual-machine/1232-vmware-workstation.html

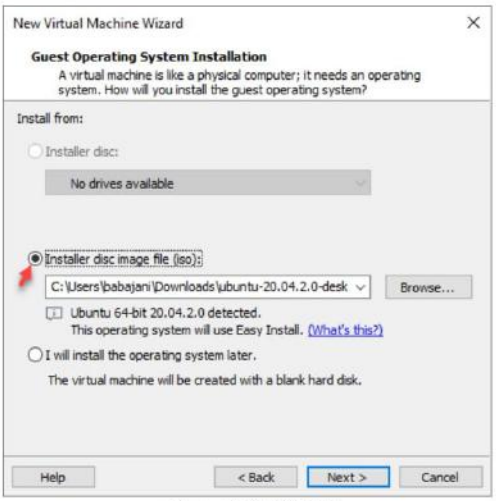
بعد از دانلود، آن را نصب کنید و برای آغاز کار باید به‌مانند شکل ٤٤-۲ وارد منوی **File** شوید و گزینه‌ی New Virtual Machine را انتخاب کنید.

شکل ۴۴-۲ /ایجاد ماشین مجازی

در شکل ٤٥-۲ گزینه‌ی Typical را انتخاب و بر روی Next کلیک کنید.

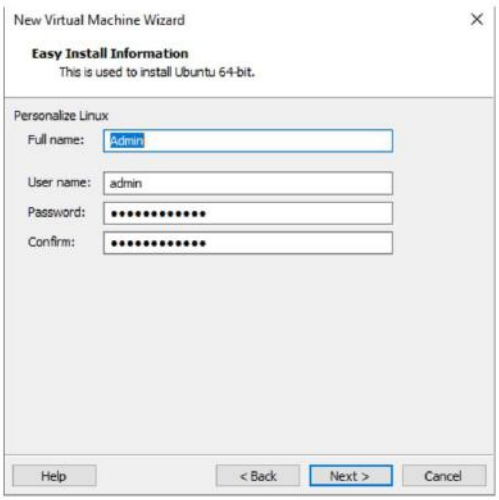
شکل ۴۵-۲ ایجاد ماشین مجازی

در شکل ٤٦-۲ باید فایل ISO مربوط به لینوکس Ubuntu را که لینک آن را در قسمت پیش قرار دادیم معرفی کنید، بعد از معرفی به‌صورت اتوماتیک نوع سیستم‌عامل در زیر آن مشخص خواهد شد.



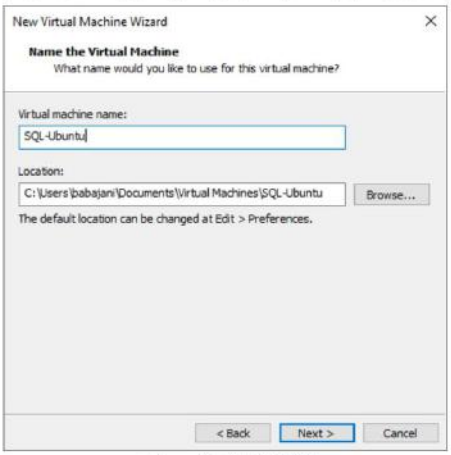
شکل ۴۶-۲ ایجاد ماشین مجازی

در شکل ٤٧-۲ باید اطلاعات تکمیلی را وارد کنید، مانند نام کاربری و رمز عبور برای لینوکس Ubuntu.



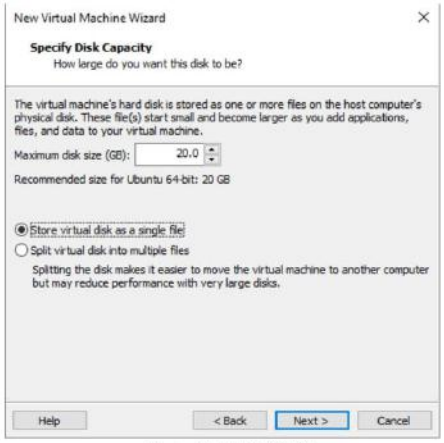
شکل ۴۷-۲ /ایجاد ماشین مجازی

در شکل ۴۸-۲ باید نام ماشین مجازی خود به همراه آدرس ذخیره‌سازی آن در هارددیسک را مشخص کنید.



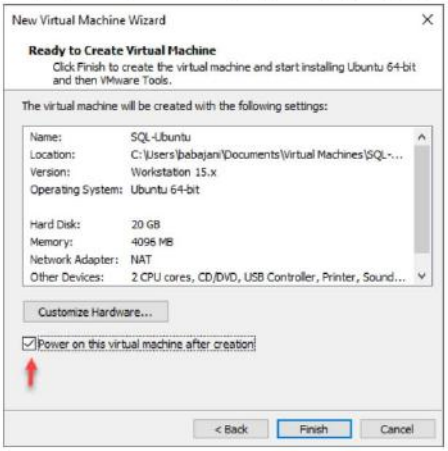
شکل ۴۸-۲ /ایجاد ماشین مجازی

در شکل ۴۹-۲ باید مقدار فضای هارددیسک را مشخص کنید که برای این سیستم‌عامل، ۲۰ گیگابایت کفایت می‌کند و برای اینکه هارددیسک مجازی فقط یک فایل تکی باشد باید گزینه‌ی Store Virtual disk as single Files را انتخاب کنید.



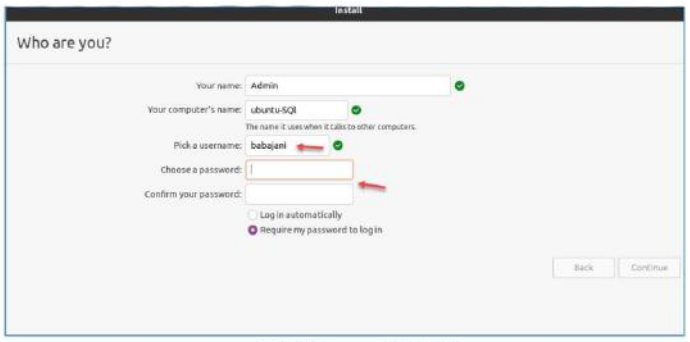
شکل ۲-۴۹ ایجاد ماشین مجازی

در شکل ۲-۵۰ اطلاعات کلی را مشاهده می‌کنید که اگر نیاز باشد تغییراتی در سخت‌افزار اعمال کنید و یا اینکه بخواهید سخت‌افزار جدید به این ماشین اضافه کنید باید بر روی Customize Hradware کلیک کنید، در پایان برای اینکه ماشین بعد از ایجاد شدن، روشن شود تیک گزینه‌ی .Power on را انتخاب کنید.



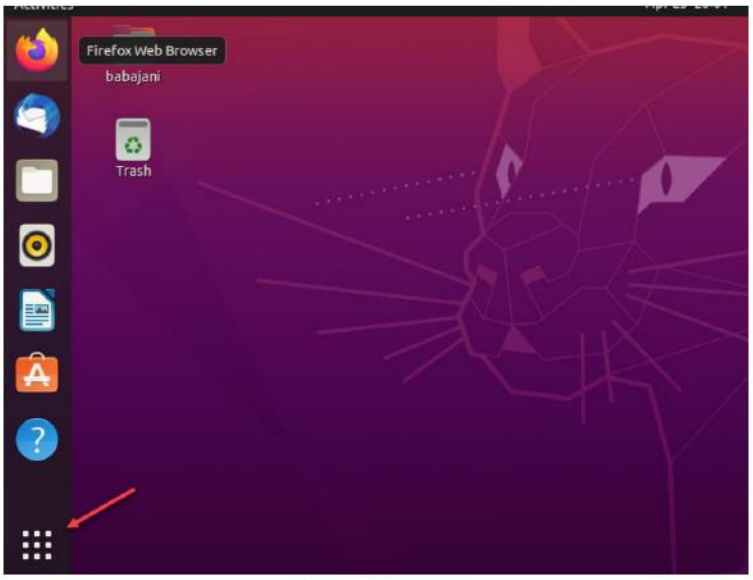
شکل ۲-۵۰ ایجاد ماشین مجازی

بعد از اجرای ماشین مورد نظر مراحل نصب به‌صورت اتوماتیک انجام خواهد شد، اما به‌مانند شکل ۲-۵۱ چون نام کاربری admin یک نام رزرو شده است باید یک نام جدید، وارد و رمز آن را مشخص کنید و بر روی Continue کلیک کنید تا کار نصب به پایان برسد.



شکل ۵۱-۲ نصب سیستم‌عامل لینوکس

در ادامه‌ی کار و بعد از نصب کامل لینوکس Ubuntu باید به‌مانند شکل ۵۲-۲ بر روی Show Applications کلیک کنید و در کادر جستجو، Terminal را اجرا کنید.



شکل ۵۲-۲ لینوکس *Ubuntu*

بعد از اجرا شدن سرویس باید به‌مانند شکل ۵۳-۲ کاربر Root را فعال کنید، برای این کار از دو دستور زیر استفاده کنید:

```
sudo passwd root
sudo passwd -u root
```

توجه داشته باشید بعد از اجرای دستور اول باید یک رمز عبور جدید برای کاربر Root وارد کنید تا بتوانید از آن استفاده کنید.

شکل ۵۳-۲ فعال‌سازی کاربر root

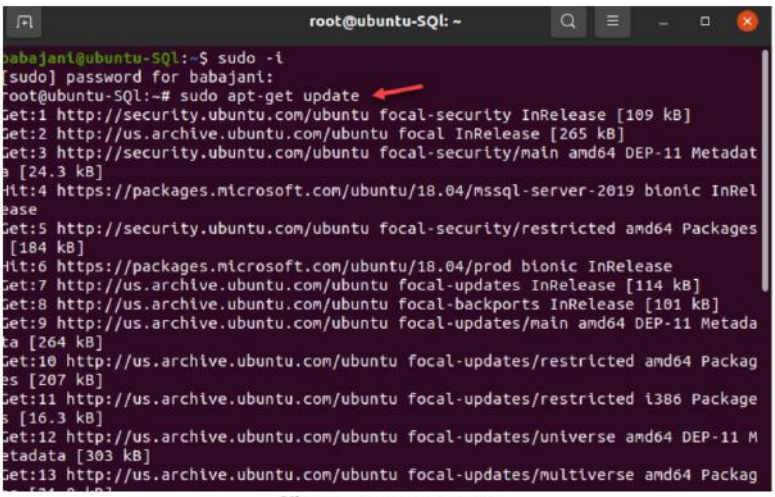
در ادامه‌ی کار با دستور sudo –i، کاربر را به root تغییر دهید تا بتوانید دستورات مورد نظر خود را با بالاترین دسترسی اجرا کنید.

شکل ۵۴-۲ اجرای کاربر root

برای فعال‌سازی SQL Server بر روی لینوکس Ubuntu باید دستورات زیر را وارد کنید.

دو دستور زیر آخرین آپدیت‌ها را بر روی لینوکس Ubuntu نصب می‌کند تا در ادامه بتوانید، دستورات را به‌راحتی اجرا کنید که نتیجه‌ی آن را در شکل ۵۵-۲ مشاهده می‌کنید.

```
sudo apt-get update
sudo apt-get -y upgrade
```



شکل ۵۵-۲ آپدیت لینوکس Ubuntu

با دستور زیر، بعد از اینکه آپدیت‌ها فعال شدند لینوکس را یک‌بار Restart کنید.

```
sudo reboot
```

بعد از اجرا شدن سرور مجازی لینوکس، وارد ترمینال شوید و دستور زیر را برای فعال‌سازی repository اجرا کنید.

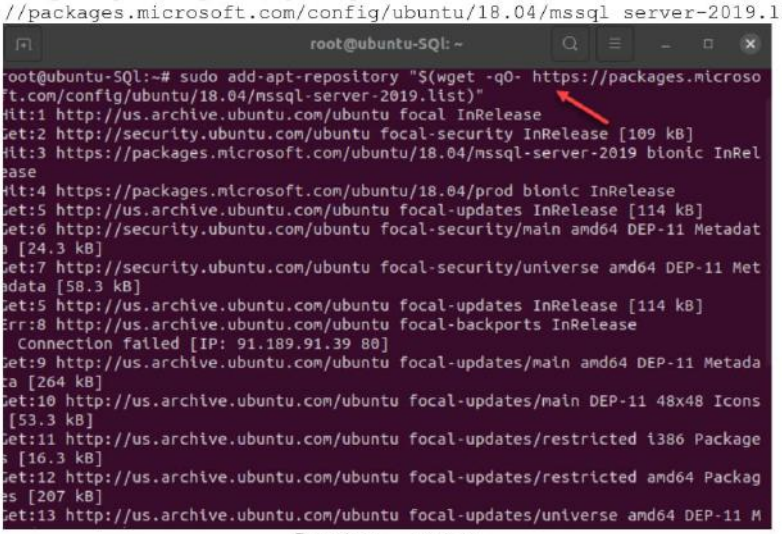
```
sudo wget -qO- https://packages.microsoft.com/keys/microsoft.asc | sudo
apt-key add -
```

همان‌طور که در شکل ۵۶-۲ مشاهده می‌کنید، دستور مورد نظر به‌درستی اجرا شده است.

شکل ۵۶-۲ فعال‌سازی *Repository*

در ادامه‌ی کار با دستور زیر، Repository یا همان مخزن مربوط به SQL Server را دانلود و فعال می‌کنیم، اگر به شکل ۵۷-۲ دقت کنید این دستور به‌درستی اجرا شده است.

```
sudo add-apt-repository "$ (wget -qO-
https://packages.microsoft.com/config/ubuntu/18.04/mssql server-2019.list)"
```

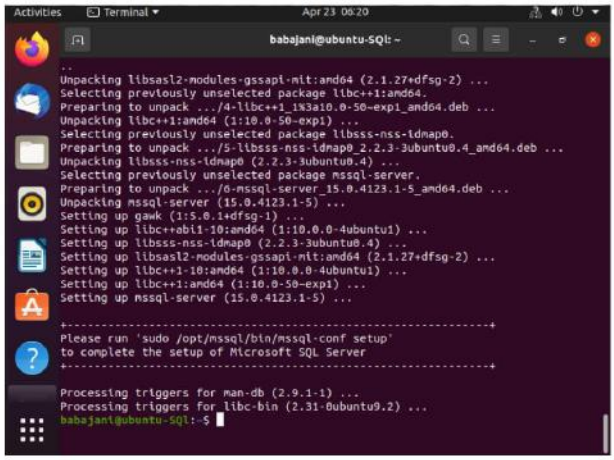


شکل ۵۷-۲ نصب *Repository*

در ادامه‌ی کار با دستور زیر، MSSQL Server را بر روی لینوکس Ubuntu نصب کنید:

```
sudo apt update
sudo apt install mssql-server
```

همان‌طور که در شکل ۵۸-۲ مشاهده می‌کنید، دستورات بالا اجرا و MSSQL نصب شده است که در ادامه باید آن را تنظیم کنید.

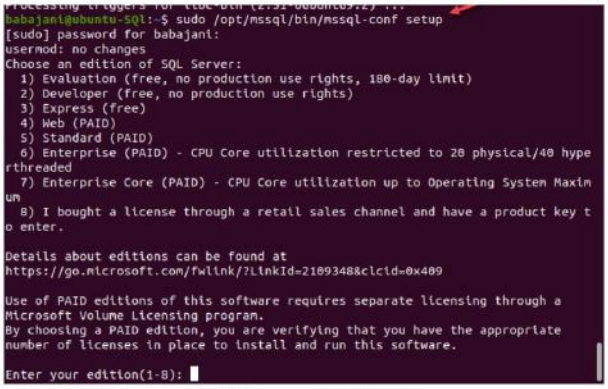


شکل ۵۸-۲ نصب *MSSQL*

برای **تنظیم و استارت سرویس** MSSQL **باید دستور زیر را وارد کنید:**

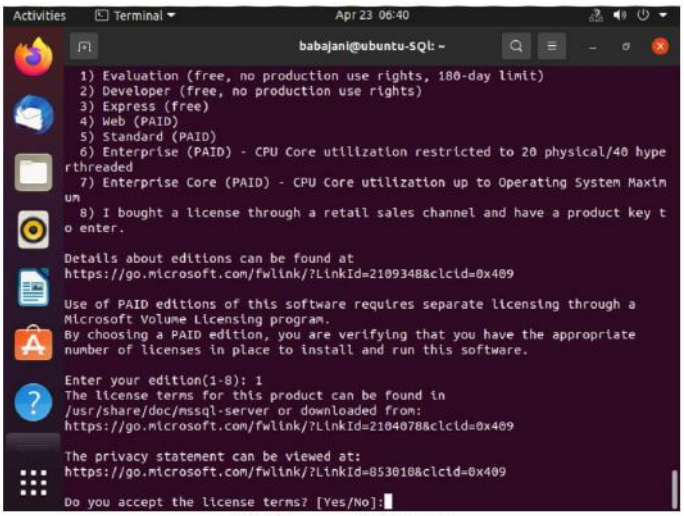
```
/opt/mssql/bin/mssql-conf setup
```

با اجرای دستورر بالا، شکل ۵۹-۲ ظاهر خواهد شد که باید یک نسخه از SQL **سرور را انتخاب کنید که در اینجا گزینه‌ی یک را انتخاب می‌کنیم.**



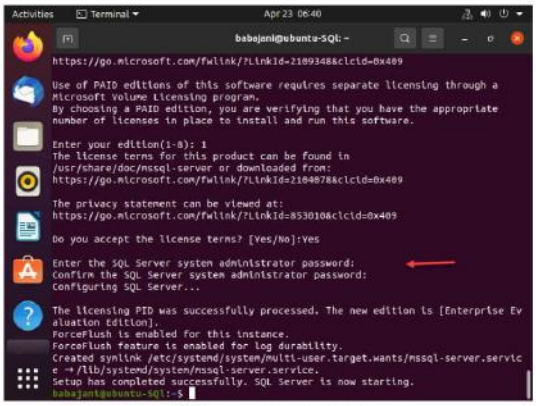
شکل ۵۹-۲ اجرای سرویس *MSSQL*

بعد از وارد کردن عدد یک باید کلمه‌ی Yes را به‌مانند شکل ۶۰-۲ برای تأیید لایسنس برنامه وارد کنید تا کار ادامه پیدا کند.



شکل ۶۰-۲ اجرای سرویس *MSSQL*

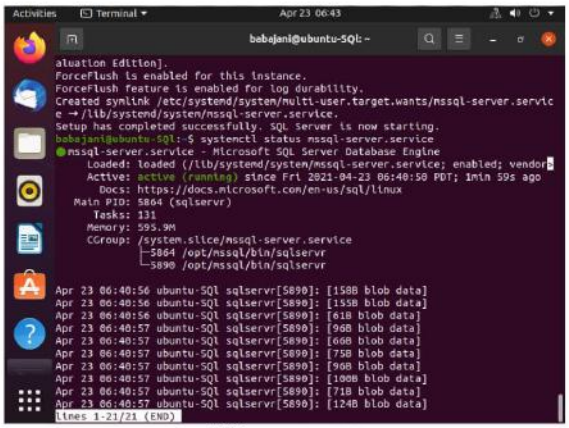
در ادامه به‌مانند شکل ۶۱-۲ باید یک رمز عبور برای مدیریت SQL وارد کنید، بعد از آن سرویس SQL اجرا خواهد شد.



شکل ۶۱-۲ اجرای سرویس *SQL*

برای اینکه متوجه شویم، سرویس فعال شده است یا نه باید دستور زیر را وارد کنید، تا شکل ۶۲-۲ ظاهر شود.

```
systemctl status mssql-server.service
```

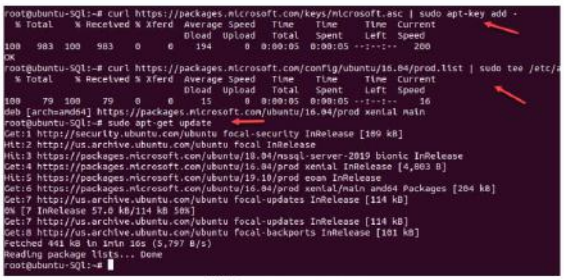


شکل ۲-۶۲ سرویس *SQL*

در ادامه‌ی کار باید ابزار مورد نیاز را بر روی سرور لینوکس برای کار با SQL نصب کنید؛ برای این کار دستورات زیر را به صورت متوالی اجرا کنید:

```
curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -

curl https://packages.microsoft.com/config/ubuntu/19.10/prod.list >
/etc/apt/sources.list.d/mssql-release.list

sudo apt update

sudo ACCEPT_EULA=Y apt-get install mssql-tools unixodbc-dev
```

با اجرای دستورات بالا به‌مانند شکل ۲-۶۳ ابزار مربوط به SQL بر روی لینوکس نصب می‌شود.



شکل ۲-۶۳ نصب ابزار *SQL*

در ادامه‌ی دستورات در قسمتی که در شکل ۲-٦٤ مشخص شده است باید Yes را وارد کنید تا ابزار MSSQL که حجمی حدود ۱۰۲ مگابایت دارد بر روی سرور لینوکس نصب شود.

شکل ۲-۶۴ نصب ابزار SQL

در ادامه‌ی کار، دو دستور زیر را وارد کنید تا کار نصب و تنظیم SQL به پایان برسد:

```
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bash_profile


echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc
source ~/.bashrc
```

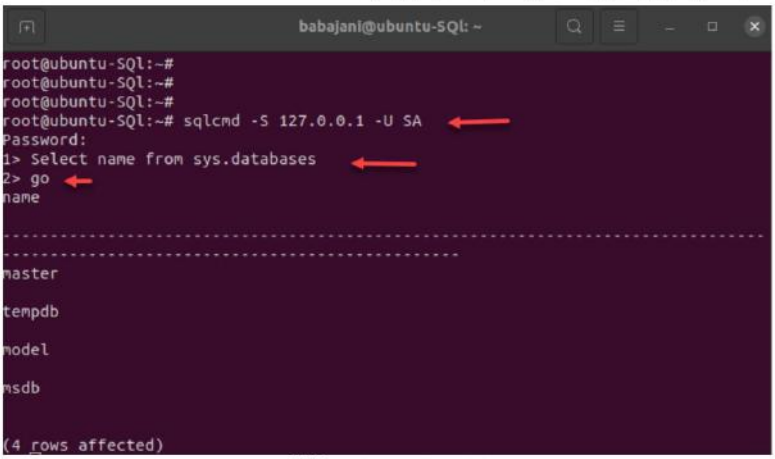
بعد از نصب SQL و ابزارهای آن بر روی سرور لینوکس، حال می‌توانید بر روی SQL کار کنید، برای تست کارایی سرویس SQL دستور زیر را وارد کنید تا به آن متصل شوید:

```
sqlcmd -S 127.0.0.1 -U SA
```

همان‌طور که در شکل ۲-۶۵ مشاهده می‌کنید، بعد از اجرای دستور بالا از ما رمز عبوری را می‌خواهد که در هنگام نصب SQL وارد کردیم، یعنی شکل شماره‌ی ۲-۶۱؛ بعد از وارد کردن رمز عبور حالا وارد خط فرمان SQL شدیم و می‌توانیم دستورات مورد نظر را اجرا کنیم. برای تست دستور زیر را اجرا می‌کنیم:

```
select name from sys.databases;
go
```

با اجرای دستور بالا، لیست دیتابیس‌های اصلی و مربوط به تنظیمات خود SQL را مشاهده می‌کنید که در مورد این دیتابیس‌ها در فصل سوم به‌صورت کامل صحبت خواهیم کرد.



شکل ۲-۶۵ لیست دیتابیس SQL

فصل سوم

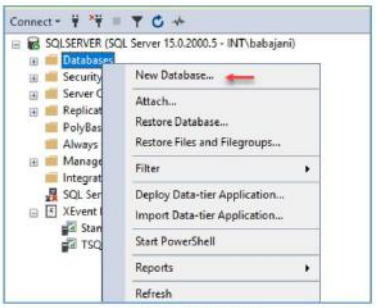
آشنایی با پایگاه‌داده

۱-۳ ایجاد پایگاه‌داده و کار با آن

۱-۱-۳ ایجاد پایگاه‌داده

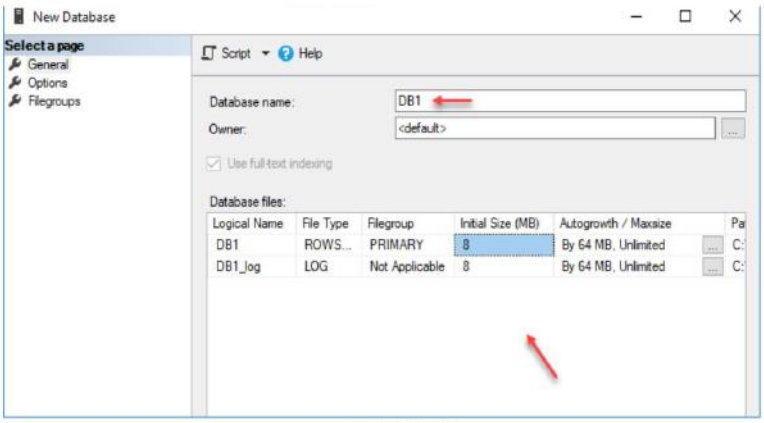
در ادامه‌ی کار می‌خواهیم برای شروع، یک دیتابیس ایجاد و بر روی آن کارهای مختلفی را انجام می‌دهیم، ایجاد دیتابیس را می‌توانیم به دو صورت انجام دهیم، یکی به‌صورت گرافیکی و از طریق نرم‌افزار SSMS و یکی از طریق کد T-SQL که در ادامه هر دو روش را بررسی خواهیم کرد.

برای شروع به‌مانند شکل ۱-۳ وارد SSMS شوید و بر روی Databases کلیک راست کنید و بر روی New Database کلیک کنید.



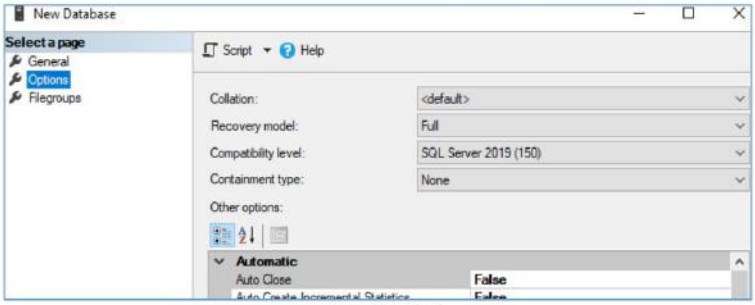
شکل ۱-۳ ایجاد دیتابیس

در شکل ۲-۳ باید نام دیتابیس مورد نظر خود را در قسمت Database Name وارد کنید و در قسمت Owner باید یک کاربر را به‌عنوان صاحب این دیتابیس انتخاب کنید که اگر انتخاب نکنید، همین کاربری که با آن در حال ایجاد دیتابیس هستید به‌عنوان صاحب آن در نظر گرفته خواهد شد؛ بعد از واردکردن نام در قسمت Database name، دو گزینه با نام دیتابیس شما ایجاد می‌شود که اولی (DB1) نام دیتابیس و دومی (DB1_Log) دیتابیسی برای ثبت Log است، این دو مکمل هم هستند و حتماً برای اجرا به همدیگر نیاز دارند.



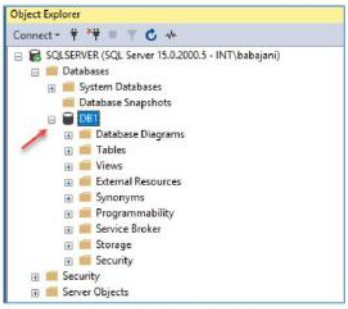
شکل ۲-۳ / ایجاد دیتابیس

در قسمت Option که در شکل ۳-۳ مشخص شده است، گزینه‌هایی وجود دارد، گزینه‌ی Collation به روشی برای مقایسه‌ی اطلاعات با هم اشاره دارد؛ قسمت Recovery model که مربوط به عملیات پشتیبان‌گیری و بازیابی اطلاعات است که در موقع مناسب توضیح خواهیم داد. در قسمت Compatibility level هم می‌توانید مشخص کنید که این دیتابیس با کدام نسخه از SQL سازگاری داشته باشد.



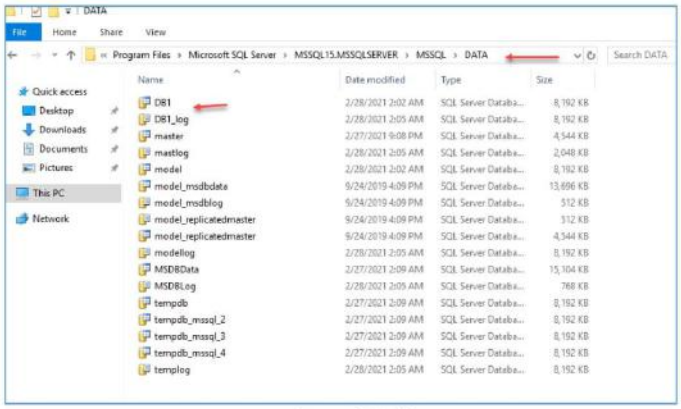
شکل ۳-۳ / ایجاد دیتابیس

بعد از ایجاد دیتابیس می‌توانید به‌مانند شکل ٤-۳ در قسمت Databases، این دیتابیس جدید را مشاهده کنید.



شکل ۴-۳

اگر وارد مسیر مورد نظر در شکل ۵-۳ شوید، فایل دیتابیس تولید شده را مشاهده می‌کنید که همان دو فایلی است که اشاره کردیم.



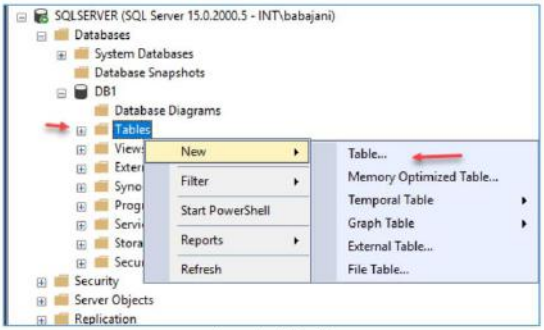
شکل ۵-۳ لیست دیتابیس

نکته:

سعی کنید در اول کار زمانی‌که می‌خواهید دیتابیس خود را ایجاد کنید، آدرس آن را در یک مکان مطمئن و با حجم بالا قرار دهید تا در آینده با اضافه شدن داده، حجم و اطلاعات شما حفظ شود.

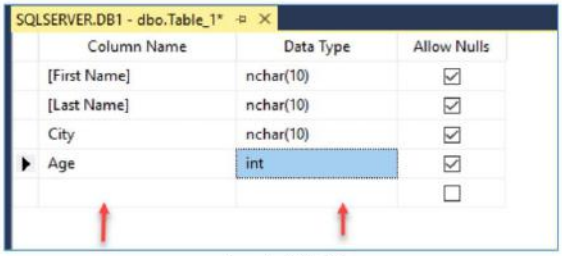
۲-۱-۳ ایجاد جدول در دیتابیس

برای ایجاد جدول در دیتابیس مورد نظر باید به‌مانند شکل ۲۵-۲ بر روی پوشه‌ی Tables کلیک راست کنید و گزینه‌ی Table را از قسمت New انتخاب کنید.



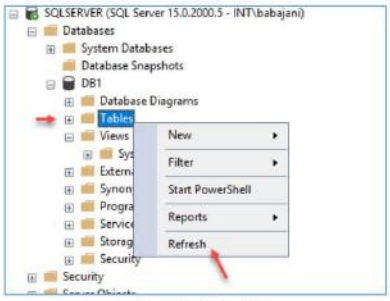
شکل ۶-۳ ایجاد جدول

در شکل ۷-۳ باید در قسمت Column Name، نام ستون خود را وارد کنید که چهار مورد وارد شده است و در قسمت Data Type باید نوع ستون مورد نظر را از نظر عددی، حرفی و ... انتخاب کنید، برای ذخیره کردن این جدول باید بر روی عنوان جدول کلیک راست کنید و گزینه‌ی Save را انتخاب کنید.



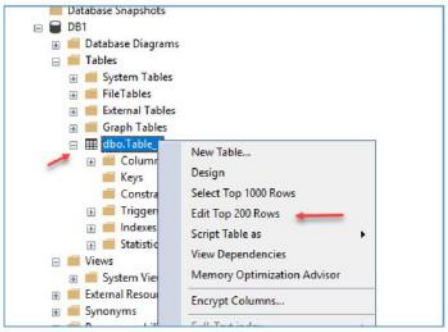
شکل ۷-۳ / ایجاد جدول

بعد از ایجاد جدول باید جدول مورد نظر در قسمت Table نمایش داده شود، اگر این‌چنین نشد باید به‌مانند شکل ۸-۳ بر روی Tables کلیک راست کنید و گزینه‌ی Refresh را انتخاب کنید تا جدول مورد نظر مشخص شود.



شکل ۸-۳ / ایجاد جدول

بعد از ایجاد جدول می‌خواهیم اطلاعاتی در آن وارد کنیم؛ برای این کار در شکل ۹-۳ بر روی dbo.Table_1 کلیک راست کنید و گزینه‌ی Edit Top 200 Rows را انتخاب کنید.



شکل ۹-۳ ورود اطلاعات در جدول

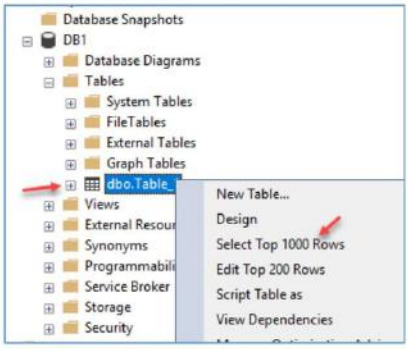
در شکل ۱۰-۳ اطلاعاتی را وارد کنید و صفحه‌ی مورد نظر را ببندید.

SQLSERVER.DB1 - dbo.Table_1

	First Name	Last Name	City	Age
	ali	modares	babol	25
	reza	hazir	dezfol	32
	azadeh	kordi	amol	33
	meysam	moham	shiraz	35

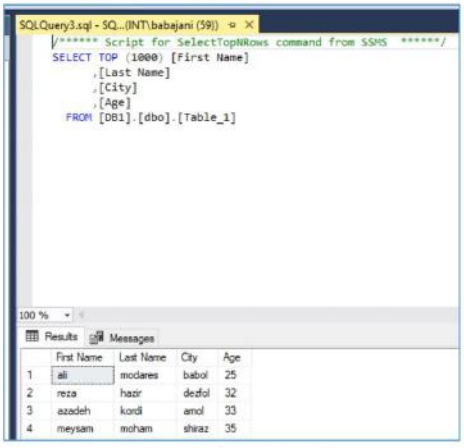
شکل ۱۰-۳ ورود اطلاعات در جدول

برای اینکه اطلاعات موجود در جدول را مشاهده کنید، می‌توانید به‌مانند شکل ۱۱-۳ بر روی جدول مورد نظر کلیک راست کنید و گزینه‌ی Select Top 1000 Rows را انتخاب کنید.



شکل ۱۱-۳ نمایش اطلاعات جدول

همان‌طور که در شکل ۱۲-۳ مشاهده می‌کنید، اطلاعات جدول توسط یک اسکریپت در خروجی به نمایش گذاشته شده است که در ادامه، در مورد کد مورد نظر توضیحاتی خواهیم داد.



شکل ۱۲-۳ نمایش اطلاعات جدول

بعد از ایجاد اولیه‌ی جدول و واردکردن مقادیری در آن، می‌خواهیم در مورد نوع داده‌ی ورودی صحبت کنیم که بسیار مهم است.

۳-۱-۳ انواع Data Type در جداول

زمانی‌که می‌خواهیم یک Filed جدید در جدول، مانند: نام، نام خانوادگی، شماره‌ی دانشجویی و... ایجاد کنیم باید از انواع مختلف داده‌ای استفاده کنیم، مثلاً برای نوشته‌ای مانند نام باید از نوع داده‌ی Char/nchar استفاده کنید و به همین ترتیب از انواع مختلف دیگر می‌توان استفاده کرد. در زیر این نوع داده‌ها را بررسی می‌کنیم:

انواع داده‌های رشته‌ای

جدول ۱-۳ انواع داده‌های رشته‌ای

نوع داده	اندازه داده	توضیحات
CHAR(*size*)	حداکثر اندازه ۸۰۰۰ کاراکتر.	طول داده ثابت است.
VARCHAR(*size*) or VARCHAR(max)	حداکثر اندازه ۸۰۰۰ کاراکتر با افزایش حجم.	طول داده متغیر است و توانایی افزایش حجم تا ۲ گیگابایت را داراست.
TEXT	حداکثر اندازه ۲ گیگابایت.	داده‌های غیر یونیک با طول متغیر.
NCHAR(*size*)	حداکثر اندازه ٤۰۰۰ کاراکتر.	داده‌های غیر یونیک با طول ثابت.
NVARCHAR(*size*) or NVARCHAR(max)	حداکثر اندازه ٤۰۰۰ کاراکتر با افزایش حجم.	طول داده متغیر است و توانایی افزایش حجم تا ۲ گیگابایت را داراست.
NTEXT	حداکثر اندازه ۱,۰۷۳,۷٤۱,۸۲۳ bytes است.	داده‌های غیر یونیک با طول متغیر.
BINARY(*size*)	حداکثر اندازه ۸۰۰۰ کاراکتر.	طول داده ثابت است.
VARBINARY(*size*) or VARBINARY(max)	حداکثر اندازه ۸۰۰۰ کاراکتر با افزایش حجم.	طول داده متغیر است و توانایی افزایش حجم تا ۲ گیگابایت را داراست.

IMAGE	حداکثر اندازه ۲ گیگابایت.	داده‌های غیر یونیک با طول متغیر.

انواع داده‌های عددی

جدول ۲-۳ داده‌های عددی

نوع داده عددی	حداکثر اندازه	توضیحات
BIT	عدد صحیح است که می‌تواند ۰ یا ۱ یا NULL باشد.	
TINYINT	از ۰ تا ۲۵۵	
SMALLINT	از -۳۲۷۶۸ تا ۳۲۷۶۷	
INT	از -۲,۱۴۷,۴۸۳,۶۴۸ تا ۲,۱۴۷,۴۸۳,۶۴۷	
BIGINT	از -۹,۲۲۳,۳۷۲,۰۳۶,۸۵۴,۷۷۵,۸۰۸ تا ۹,۲۲۳,۳۷۲,۰۳۶,۸۵۴,۷۷۵,۸۰۷	
DECIMAL(*m*,*d*)	M اگر مشخص نشده باشد به‌صورت پیش‌فرض ۱۸ در نظر گرفته می‌شود.	M تعداد ارقام و d تعداد ارقام اعشار بعد از ممیز است.
	d اگر مشخص نشده باشد، به‌صورت پیش‌فرض ۰ در نظر گرفته می‌شود.	
DEC(*m*,*d*)	M اگر مشخص نشده باشد به‌صورت پیش‌فرض ۱۸ در نظر گرفته می‌شود.	M تعداد ارقام و d تعداد ارقام اعشار بعد از ممیز.

	d اگر مشخص نشده باشد، به‌صورت پیش‌فرض ۰ در نظر گرفته می‌شود.	
NUMERIC(*m*,*d*)	M اگر مشخص نشده باشد به‌صورت پیش‌فرض ۱۸ در نظر گرفته می‌شود.	M تعداد ارقام و d تعداد ارقام اعشار بعد از ممیز.
	d اگر مشخص نشده باشد، به‌صورت پیش‌فرض ۰ در نظر گرفته می‌شود.	
FLOAT(*n*)	اعداد شناور	که در آن n تعداد تعداد بیت برای ذخیره به‌صورت نماد علمی است.
	N به‌صورت پیش‌فرض ۵۳ در نظر گرفته خواهد شد.	
SMALLMONEY	از – ۲۱۴,۷۴۸.۳۶۴۸ تا ۲۱۴,۷۴۸.۳۶۴۷	
MONEY	از –۹۲۲,۳۳۷,۲۰۳,۶۸۵,۴۷۷.۵۸۰۸ تا ۹۲۲,۳۳۷,۲۰۳,۶۸۵,۴۷۷.۵۸۰۷	

انواع داده‌های زمان و تاریخ

جدول ۳-۳ داده‌های زمان و تاریخ

نوع داده	حداکثر اندازه کاراکتر	توضیحات
DATE	دامنه‌ی اعداد از '0001-01-01' تا '۹۹۹۹-۱۲-۳۱'.	فرمت نمایش 'YYYY-MM-DD'"

DATETIME	دامنه‌ی اعداد از '۱۷۵۳-۰۱-۰۱ ۰۰:۰۰:۰۰' تا '9999-12-31 23:59:59'.	فرمت نمایش 'YYYY-MM-DD hh:mm:ss[.mmm]'
	دامنه‌ی زمان از '۰۰:۰۰:۰۰' تا '۲۳:۵۹:۵۹:۹۹۷'	
DATETIME2(fractional seconds precision)	دامنه‌ی تاریخ از '0001-01-01' تا '۹۹۹۹-۱۲-۳۱'.	فرمت نمایش 'YYYY-MM-DD hh:mm:ss[.fractional seconds]'
	دامنه‌ی زمان از '00:00:00' تا '23:59:59:9999999'.	
SMALLDATETIME	دامنه‌ی تاریخ از '1900-01-01' تا '۲۰۷۹-۰۶-۰۶'.	فرمت نمایش 'YYYY-MM-DD hh:mm:ss'
	دامنه‌ی زمان از '۰۰:۰۰:۰۰' تا '۲۳:۵۹:۵۹'.	
TIME	دامنه‌ی زمان از'۰۰:۰۰:۰۰.۰۰۰۰۰۰۰' تا '23:59:59.9999999'	فرمت نمایش 'YYYY-MM-DD hh:mm:ss[.nnnnnnn]'
DATETIMEOFFSET(fractional seconds precision)	دامنه‌ی تاریخ از '0001-01-01' تا '۹۹۹۹-۱۲-۳۱'.	فرمت نمایش 'YYYY-MM-DD hh:mm:ss[.nnnnnnn] [{+|-}hh:mm]'
	دامنه‌ی زمان از '۰۰:۰۰:۰۰' تا '23:59:59:9999999'.	
	دامنه‌ی منطقه از -۱۴:۰۰ تا +۱۴:۰۰.	

در جدول‌هایی که بررسی کردیم، تمام داده‌های رشته‌ای، عددی، زمان و تاریخ تعریف شده است و توضیحات مربوط به آن داده شده است.

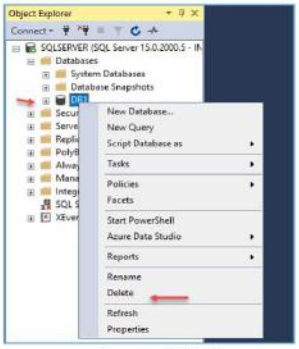
Data Typeهای دیگری نیز وجود دارد که در نسخه‌های جدید SQL می‌توانید از آنها استفاده کنید:

جدول ۴-۳ داده‌های دیگر

نوع داده	توضیحات
sql_variant	حداکثر ۸۰۰۰ بایت داده از انواع مختلف داده، به جز متن، ntext و زمان‌سنج ذخیره می‌کند.
uniqueidentifier	شناسه منحصربه‌فرد جهانی (GUID) را ذخیره می‌کند.
xml	داده‌های قالب‌بندی شده XML را ذخیره می‌کند، حداکثر ۲ گیگابایت.
cursor	این نوع داده برای ذخیره‌ی متغیرها و یا پارامترهای OUTPUT مربوط به store proceduere کاربرد دارد.
table	مجموعه‌ای از نتایج را برای پردازش بعدی ذخیره می‌کند.

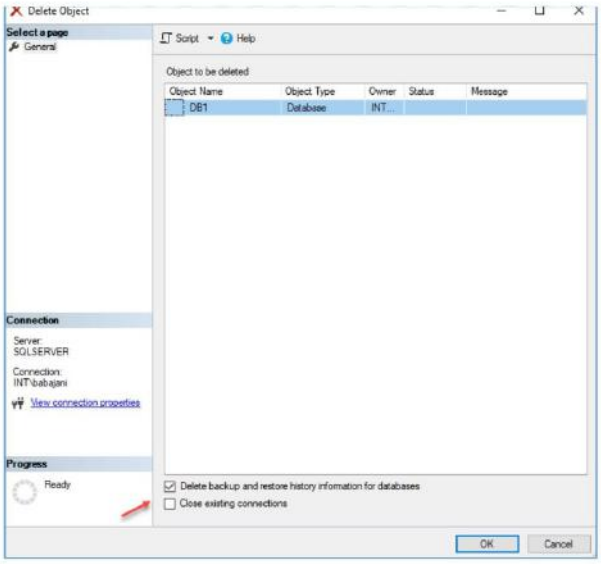
۳-۱-٤ حذف دیتابیس در SQL Server

بعد از اینکه توانستیم یک دیتابیس را ایجاد کنیم نوبت به حذف آن می‌رسد، برای حذف دیتابیس کافی است به‌مانند شکل ۱۳-۳ بر روی دیتابیس مورد نظر کلیک راست کنید و گزینه‌ی Delete را انتخاب کنید.



شکل ۱۳-۳ حذف دیتابیس

در شکل ۱٤-۳ نام دیتابیس DB1 در لیست مشخص شده است و اگر بر روی Delete کلیک کنید، دیتابیس حذف خواهد شد، اما دو گزینه در زیر این صفحه وجود دارد.



شکل ۳-۱۴ حذف دیتابیس

گزینه‌ی Delete backup and restore history information for databases: این گزینه به‌صورت پیش‌فرض فعال است و تاریخ و اطلاعات این دیتابیس را از دیتابیس MSDB حذف می‌کند.

گزینه‌ی Close existing connections، به‌صورت پیش‌فرض فعال نیست و اگر آن را فعال کنید، قبل از حذف دیتابیس، اول بررسی می‌کند که این دیتابیس در جایی دیگر باز شده و یا در حال استفاده است، ابتدا آن ارتباط را می‌بندد و بعد اقدام به حذف می‌کند، سعی کنید این گزینه را فعال نکنید تا حداقل در زمان حذف مشخص شود که در جایی دیگر در حال استفاده است.

۳-۱-۵ تعریف پرس‌وجو یا Query در SQL

یکی از اصلی‌ترین موضوعاتی که در SQL باید بررسی شود بحث پرس‌وجو و یا همان Query است، هر پرس‌وجو می‌تواند شامل چندین دستور باشد که در زیر آنها را بررسی می‌کنیم.

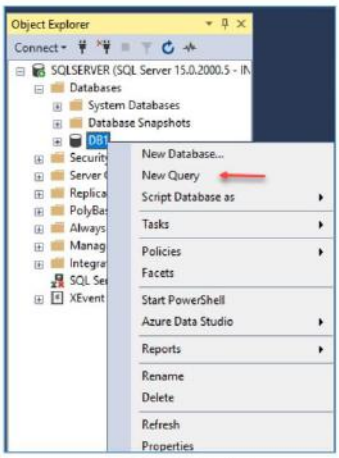
۳-۱-۵-۱ بررسی دستور SELECT

برای اینکه در SQL اطلاعات را از جدول‌های بازیابی به قولی فِچ کنیم از دستور SELECT استفاده می‌کنیم، در زیر نمونه‌ای از این دستور را مشاهده می‌کنید:

```
SELECT column1, column2, ...
FROM table_name;
```

در دستور بالا، دستور SELECT را با حروف بزرگ می‌نویسیم و بعد از آن باید ستون (*column1, column2*) مربوط به جدولی را که می‌خواهید داده‌ها را از آن استخراج کنید می‌نویسید، در سطر دوم با دستور FROM، اسم جدول را به‌جای *table_name* می‌نویسید.

برای اینکه این دستورات را در سرور SQL تست بگیریم باید به‌مانند شکل ۱۵-۳ وارد Management شوید و بر روی دیتابیس مورد نظر خود کلیک راست کنید و گزینه‌ی New Query را انتخاب کنید.



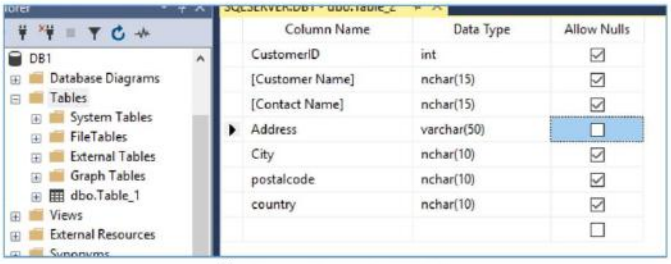
شکل ۱۵-۳ ایجاد *Query*

برای شروع کار به جدول ۵-۳ توجه کنید، می‌خواهیم با استفاده از این جدول عملیات مختلف خود را انجام دهیم و داده‌ها را فراخوانی کنیم.

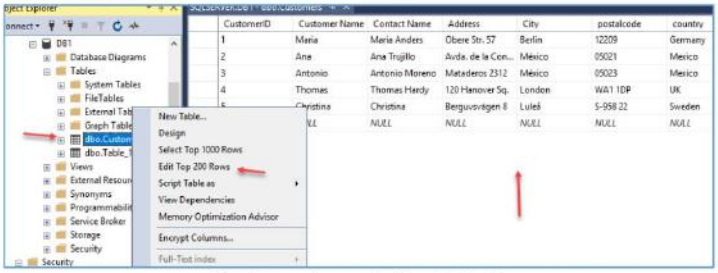
جدول ۵-۳ *Customers*

CustomerID	Customer Name	ContactName	Address	City	PostalCode	Country
1	Maria	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana	Ana Trujillo	Avda. de la Constitución 2222	México	05021	Mexico
3	Antonio	Antonio Moreno	Mataderos 2312	México	05023	Mexico
4	Thomas	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Christina	Christina	Berguvsväg en 8	Luleå	S-958 22	Sweden

برای ایجاد این جدول از طریق SSMS یک جدول ایجاد کنید و آن را با اسم Customers ذخیره کنید و بعد از ایجاد، اطلاعات جدول ۵-۳ را درون آن وارد کنید تا در ادامه بتوانید از آنها استفاده کنید، این موضوع را در شکل ۱۶-۳ مشاهده می‌کنید.



شکل ۱۶-۳ ویرایش جدول *Customers*



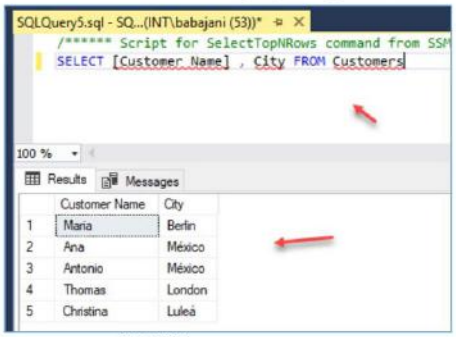
شکل ۱۷-۳ ورود اطلاعات در جدول *Customers*

مثال اول – انتخاب ستون

```
SELECT [Customer Name] , City FROM Customers
```

در دستور بالا، ستون Customer Name و City از جدول Customers انتخاب شده است، توجه داشته باشید علامت [] را بهتر است زمانی قرار دهید که یک جمله‌ی دو حرفی یا بیشتر دارید، مثلاً در مثال بالا اگر city بود، نیاز به علامت [] نبود، اما چون Customer Name از دو قسمت با فاصله تعریف شده است باید حتماً علامت [] را قرار دهید.

به‌مانند شکل ۱۸-۳ دستور مورد نظر را بنویسید و برای دریافت خروجی باید کلید F5 را فشار دهید، این دستور دو ستون Customer Name و City را در خروجی نمایش می‌دهد.



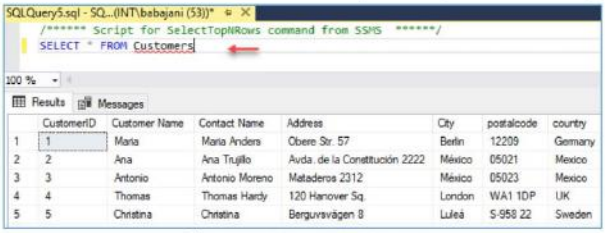
شکل ۱۸-۳ خروجی دستور *SELECT*

مثال دوم – نمایش تمام اطلاعات جدول

برای اینکه بتوانید همه‌ی اطلاعات جدول مورد نظر خود را مشاهده کنید باید از دستور زیر استفاده کنید:

```
SELECT * FROM Customers
```

همان‌طور که در شکل ۱۹-۳ مشاهده می‌کنید، تمام اطلاعات جدول در خروجی نمایش‌داده شده است.



	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country
1	1	Maria	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	2	Ana	Ana Trujillo	Avda. de la Constitución 2222	México	05021	Mexico
3	3	Antonio	Antonio Moreno	Mataderos 2312	México	05023	Mexico
4	4	Thomas	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	5	Christina	Christina	Berguvsvägen 8	Luleå	S-958 22	Sweden

شکل ۱۹-۳ نمایش جدول *Customers*

اگر بخواهیم در یک جدول، تنها اطلاعات متمایز شده و نه تکراری را نمایش دهیم باید از دستور زیر استفاده کنیم:

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

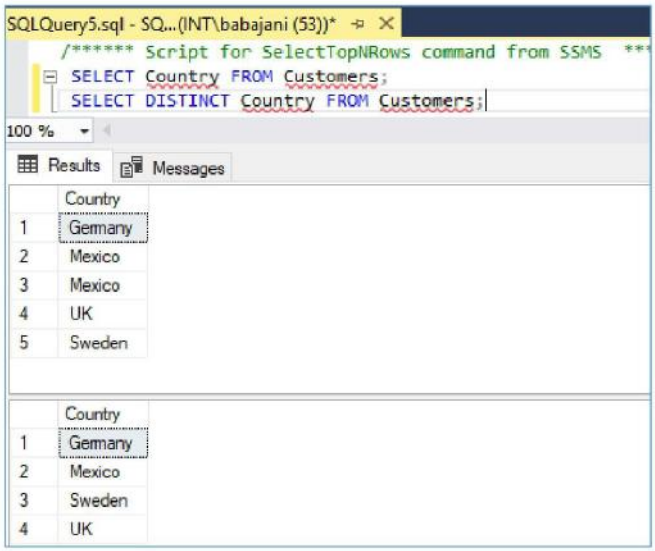
در دستور بالا به‌مانند قبل باید نام ستون‌ها را جلوی دستور SELECT بنویسید، اما تنها باید بعد از دستور SELECT از DISTINCT استفاده کنید تا اطلاعات شبیه به هم را تنها یک‌بار در خروجی نمایش دهد.

مثال سوم – نمایش اطلاعات شهرهای مشابه

دستورات زیر را اجرا کنید:

```
SELECT Country FROM Customers;
SELECT DISTINCT Country FROM Customers;
```

خروجی دستورات بالا را در شکل ۲۰-۳ مشاهده می‌کنید، دستور اول تمام شهرها را به شما نمایش داده است، اما در دستور دوم تنها یکی از شهرهای Mexico را نمایش داده است و این مورد می‌تواند در نمایش بهتر اطلاعات کمک کند.



	Country
1	Germany
2	Mexico
3	Mexico
4	UK
5	Sweden

	Country
1	Germany
2	Mexico
3	Sweden
4	UK

شکل ۲۰-۳ نمایش جدول

۲-۵-۱-۳ بررسی دستور Insert

این دستور برای واردکردن اطلاعات در جدول کاربرد دارد و می‌توانید با استفاده از این دستور، اطلاعات خود را وارد جدول کنید.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...) ;
```

در دستور بالا، ابتدا باید INSERT INTO را بنویسیم و بعد از آن، نام جدول را به‌جای table_name بنویسیم، سپس باید در پرانتز مورد نظر نام ستون‌هایی که قرار است اطلاعات در آن وارد بشود را بنویسیم.

روش دیگری هم برای ورود اطلاعات به‌صورت زیر وجود دارد:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...) ;
```

در دستور بالا، ورود اطلاعات به ترتیب ستون‌های موجود انجام می‌شود.

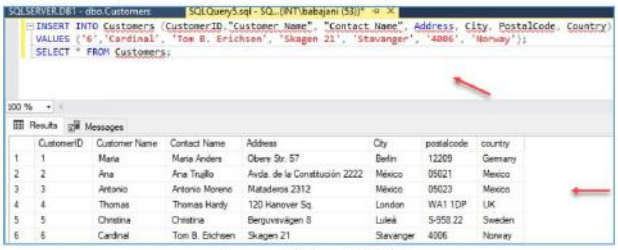
مثال اول – ورود اطلاعات با استفاده از نام ستون

```
INSERT INTO Customers (CustomerID,"Customer Name", "Contact Name", Address, City,
PostalCode, Country)
VALUES ('6','Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway')
;
SELECT * FROM Customers;
```

در دستور بالا، ابتدا نام ستون‌ها ذکر شده است و بعد از آن، اطلاعاتی که قرار است وارد جدول شود را در قسمت VALUES وارد می‌کنیم.

به این نکته توجه کنید که ستون‌هایی که دوقسمتی هستند، حتماً باید بین دو تا علامت " " قرار بگیرند تا در خروجی با خطا مواجه نشوید.

در آخر، دستور SELECT * FROM Customers; را برای نمایش اطلاعات جدول وارد کردیم که در شکل ۳-۲۱ خروجی نهایی را مشاهده می‌کنید که یک سطر جدید به جدول اضافه شده است.



شکل ۳-۲۱ ورود اطلاعات در جدول

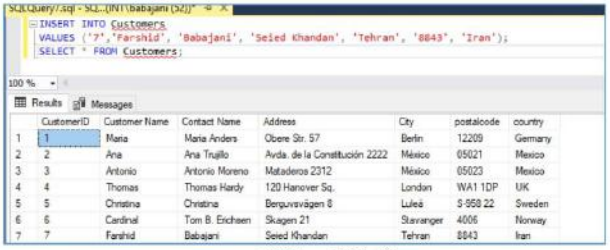
مثال دوم – ورود اطلاعات به ترتیب ستون

دستور زیر را اجرا کنید:

```
INSERT INTO Customers
```

```
VALUES ('7','Farshid', 'Babajani', 'Seied Khandan', 'Tehran', '8843', 'Iran') ;
SELECT * FROM Customers;
```

در دستور بالا، اطلاعات ورودی به ترتیب ستون‌های جدول قرار می‌گیرند و باید در ورود اطلاعات دقت کنید، اگر به شکل ۲۲-۳ توجه کنید خروجی مشخص شده است.



شکل ۲۲-۳ ورود اطلاعات

۳-۱-۵-۳ بررسی دستور Update

با استفاده از این دستور می‌توانید اطلاعات موجود در هر ستون را تغییر دهید؛ برای این کار باید از دستور زیر استفاده کنید:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

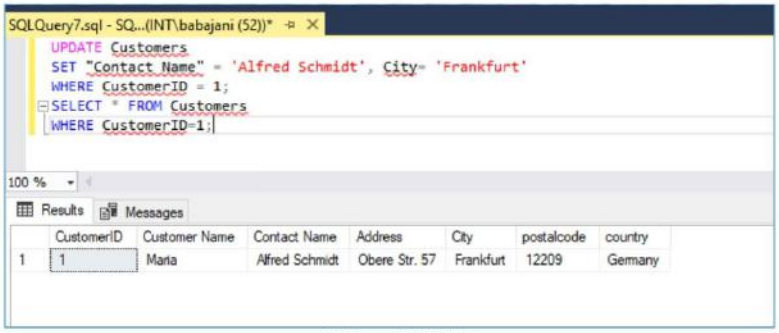
در دستور بالا، ابتدا باید نام جدول را بعد از دستور UPDATE وارد کنید. در قسمت SET باید مشخص کنید که چه گزینه‌هایی باید تغییر کند و در آخر نیز باید از دستور شرطی WHERE استفاده کنید تا مشخص شود که به کدام سطر اشاره می‌کنید.

مثال اول – آپدیت کردن اطلاعات جدول

```
UPDATE Customers
SET "Contact Name" = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
SELECT * FROM Customers
WHERE CustomerID=1;
```

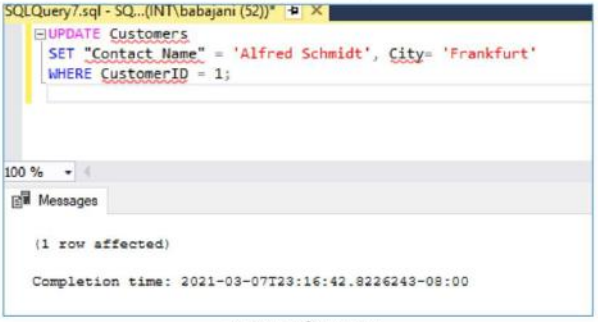
در دستور بالا، جدول Customers انتخاب می‌شود و در قسمت SET ستون‌ها به همراه مقدار آن مشخص می‌شود و در قسمت شرطی WHERE باید شرط مورد نظر را وارد کنید که در اینجا از شماره‌ی CustomerID استفاده کرده است؛ توجه داشته باشید دو خط آخر برای نمایش سطر اول جدول به کار برده شده است.

اگر به شکل ۲۳-۳ توجه کنید، سطر مورد نظر آپدیت شده است و در خروجی به نمایش گذاشته شده است.



شکل ۳-۲۳ / پدیت اطلاعات

اگر دو خط آخر دستور بالا را وارد نکنید، خروجی مانند شکل ۳-۲٤ خواهد شد.



شکل ۳-۲۴ / پدیت اطلاعات

کارهای زیادی می‌توان با دستور UPDATE انجام داد، مثلاً می‌توانید با دستور زیر نام ContactName مربوط به شهرهای Mexico را با استفاده از شرط تغییر دهید:

```
UPDATE Customers
SET "Contact Name"='Juan'
WHERE Country='Mexico';
```

نکته‌ی مهم: اگر چنانچه شرط WHERE را در دستور UPDATE قرار ندهید، تمام ستون‌های انتخاب‌شده آپدیت خواهد شد؛ در دستور زیر تمام ContactName ها به Juan تغییر خواهند کرد.

```
UPDATE Customers
SET ContactName='Juan';
```

۳-۱-۵-٤ بررسی دستور DELETE

این دستور همان‌طور که از نامش پیداست، برای حذف اطلاعات جدول‌ها کاربرد دارد که شکل کلی آن به‌صورت زیر است:

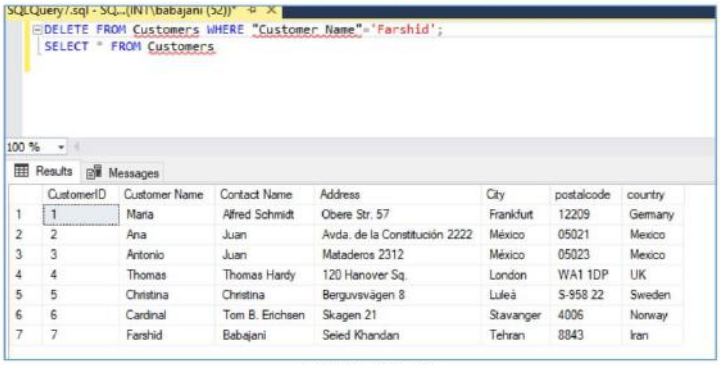
DELETE FROM *table_name* WHERE *condition*;

در دستور بالا به‌جای table_name باید نام جدول و به‌جای condition، شرط مورد نظر خود را وارد کنید.

مثال اول – حذف اطلاعات جدول

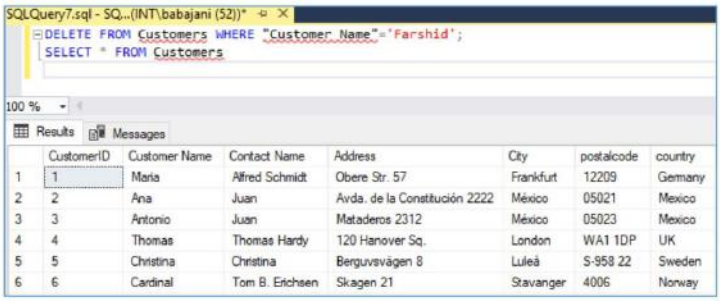
```
DELETE FROM Customers WHERE "Customer Name"='Farshid';
SELECT * FROM Customers
```

در دستور بالا، جدول Customers انتخاب شده است و در شرط آن اعلام شده است که CustomersName مشخص‌شده باید حذف شود که در شکل ۲۵-۳ قبل از اینکه دستور مورد نظر را اجرا کنید، نام Farshid را در قسمت ۷ مشاهده می‌کنید، اما بعد از اجرای دستور در شکل ۲۶-۳ سطر مورد نظر حذف شد.



	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country
1	1	Maria	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico
3	3	Antonio	Juan	Mataderos 2312	México	05023	Mexico
4	4	Thomas	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	5	Christina	Christina	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	6	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway
7	7	Farshid	Babajani	Seied Khandan	Tehran	8843	Iran

شکل ۲۵-۳ حذف اطلاعات



	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country
1	1	Maria	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico
3	3	Antonio	Juan	Mataderos 2312	México	05023	Mexico
4	4	Thomas	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	5	Christina	Christina	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	6	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

شکل ۲۶-۳ حذف اطلاعات

برای حذف اطلاعات کامل جدول باید از دستور زیر استفاده کنید:

```
DELETE FROM table_name;
```

در دستور بالا باید به‌جای table_name، نام جدول خود را وارد کنید.

۳-۱-۵-۵ بررسی دستور WHERE

یک دستور شرطی که در قسمت دستور UPDATE مقداری بررسی کردیم، شکل کلی دستور به‌صورت زیر خواهد بود

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

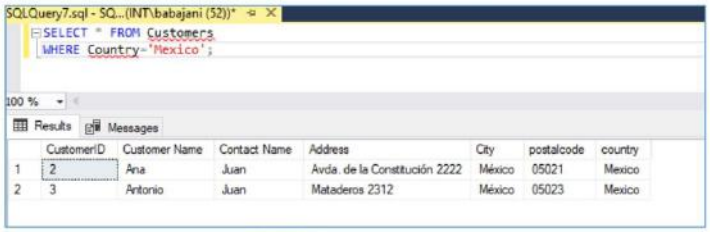
در دستور بالا، SELECT نوشتیم که ستون‌های column1, column2 را در خروجی نمایش دهد، اما در آخر از دستور WHERE استفاده شده و شرطی بر خروجی قرار داده است.

مثال یک – خروجی دستور SELECT با استفاده از شرط

```
SELECT * FROM Customers
```

```
WHERE Country='Mexico';
```

در دستور بالا، اطلاعات جدول Customers در خروجی نمایش داده می‌شود، به شرطی که ستون Country برابر Mexico باشد، خروجی دستور مورد نظر را در شکل ۲۷-۳ مشاهده می‌کنید.



شکل ۲۷-۳ بررسی دستور WHERE

نکته:

در دستور WHERE Country='Mexico'، کلمه‌ی Mexico بین نقل قول قرار گرفته است، اما اگر به جای Mexico، از عدد استفاده می‌کردید، دیگر نیاز به نقل قول نبود، پس به این نکته توجه کنید.

از عملگرهای جدول ٦-۳ می‌توان در بند WHERE استفاده کرد.

جدول ۶-۳ عملگر

عملگر	توضیحات
=	مساوی
>	بزرگ‌تر از
<	کوچک‌تر از
> =	بزرگ‌تر مساوی
<=	کوچک‌تر مساوی
<>	نابرابر، توجه داشته باشید در بعضی از نسخه‌های SQL این علامت شاید به‌صورت =! باشد.
BETWEEN	بین یک محدوده‌ی مشخص
LIKE	جستجوی یک الگو
IN	برای تعیین چندین مقدار ممکن برای یک ستون

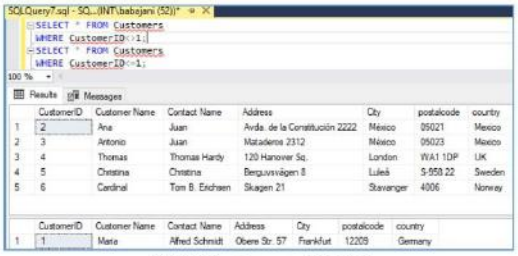
مثال دوم – استفاده از عملگر

اگر بخواهیم از این عملگرها در دستور شرطی استفاده کنید، می‌توانید از دستور زیر استفاده کنید:

```
SELECT * FROM Customers
WHERE CustomerID<>1;
```

```
SELECT * FROM Customers
WHERE CustomerID<=1;
```

در دستور بالا از یک عملگر نابرابر و یک کوچک‌تر مساوی استفاده شده است که خروجی این دستور را در شکل ۳-۲۸ مشاهده می‌کنید؛ در قسمت اول، سطر یک نمایش داده نشده و در قسمت دوم تنها سطر یک نمایش‌داده شده است.



شکل ۲۸-۳ بررسی دستور WHERE

۶-۵-۱-۳ بررسی دستور AND، OR و NOT

همان‌طور که در درس قبلی خواندید، دستور WHERE یک دستور شرطی است، شما می‌توانید با استفاده از دستورات AND، OR و NOT یک ترکیب جدید در شرط ایجاد کنید که در این قسمت به طور کامل آن را بررسی می‌کنیم.

دستور AND

فرم کلی دستور AND به‌صورت زیر است:

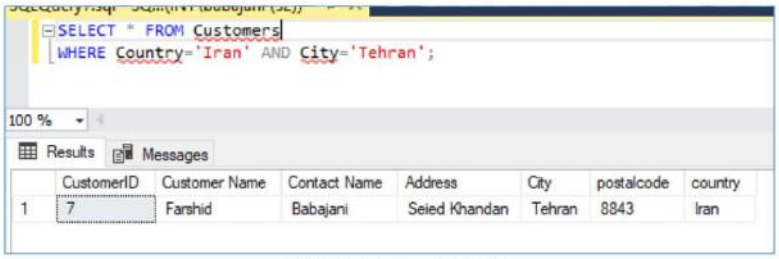
```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ... ;
```

در دستور بالا با استفاده از SELECT، نام ستون‌ها را انتخاب می‌کنیم و در قسمت WHERE باید شرط خود را با استفاده از AND وارد کنید، منظور از AND این است که یک شرط حتماً باید این چند مورد که با AND تعریف می‌شود را داشته باشد.

مثال اول – AND

```
SELECT * FROM Customers
WHERE Country='Iran' AND City='Tehran';
```

در دستور بالا، خروجی اطلاعات جدول Customers در خط اول نمایش داده خواهد شد، اما در خط دوم شرطی قرار داده که حتماً باید نام Country و City به ترتیب برابر Farshid و Tehran باشد، اگر به شکل ۳-۲۹ توجه کنید خروجی به‌درستی نمایش‌داده شده است.



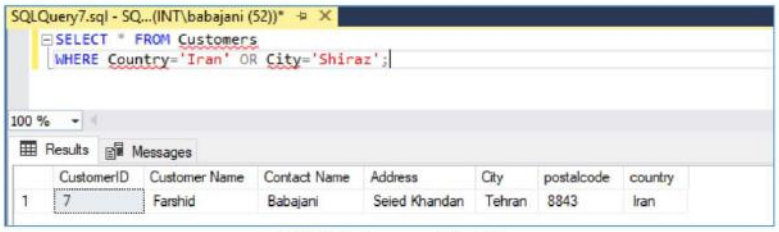
شکل ۲۹-۳ بررسی دستور *WHERE*

توجه داشته باشید که تمام مثال‌های این فصل از جدول ۱-۳ برداشته شده است و دقیقاً طبق همان جدول عمل کنید.

مثال دوم- دستور OR

```
SELECT * FROM Customers
WHERE Country='Iran' OR City='Shiraz';
```

برای دستور OR نیز همان مثال قبلی را اجرا می‌کنیم با این تفاوت که شهر را به شیراز تغییر می‌دهیم، دستور OR به این نکته اشاره دارد که اگر یکی از این شرایط درست باشد، پس شرط درست است و خروجی آن سطر باید نمایش داده شود که در شکل ۳۰-۳ این موضوع را مشاهده می‌کنید؛ با اینکه شهر Shiraz در دستور وارد شده، چون یکی از شرط‌ها، یعنی Iran درست بوده، خروجی نمایش داده شده است.



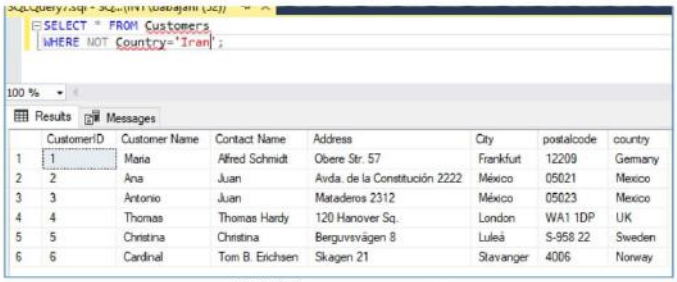
شکل ۳۰-۳ بررسی دستور *WHERE*

مثال سوم دستور NOT

این دستور، یعنی منفی کردن شرط، یعنی اینکه اگر NOT قبل از آن قرار دهیم، یعنی اینکه آن کلید نباید در خروجی باشد.

```
SELECT * FROM Customers
WHERE NOT Country='Iran';
```

در دستور بالا، اگر کشور Iran در یکی از سطرها باشد آن سطر در خروجی نمایش داده نخواهد شد که این موضوع را در شکل ۳۱-۳ مشاهده می‌کنید.



	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country
1	1	Maria	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico
3	3	Antonio	Juan	Mataderos 2312	México	05023	Mexico
4	4	Thomas	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	5	Christina	Christina	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	6	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

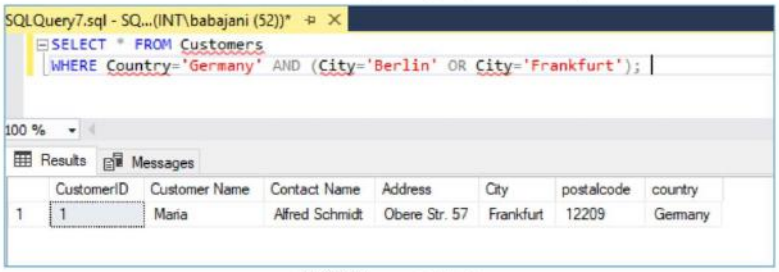
شکل ۳۱-۳ بررسی دستور *WHERE*

مثال چهارم- ترکیب سه دستور AND, OR, NOT

برای اینکه درک کامل‌تری از این سه دستور داشته باشیم، می‌خواهیم یک مثال کلی برای این سه دستور بزنیم.

```
SELECT * FROM Customers
WHERE Country='Germany' AND (City='Berlin' OR City='Frankfurt') ;
```

در دستور بالا، کشور مورد نظر Germany در نظر گرفته شده است، اما در شرطی که در داخل پرانتز قرار داده شده، گفته که یکی از شهرها می‌تواند Berlin و یا Frankfurt باشد که چون شهر Frunkfurt در جدول وجود داشت به این دلیل شرط، درست و به‌مانند شکل ۳۲-۳ در خروجی نمایش داده شده است.



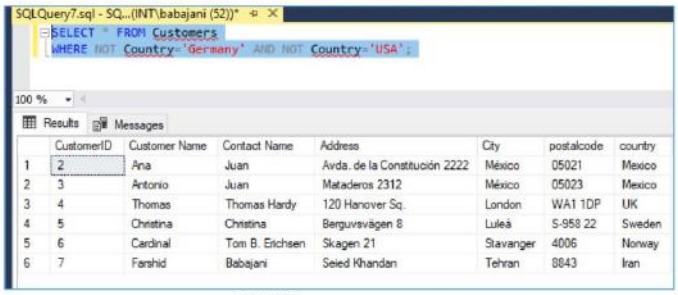
	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country
1	1	Maria	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany

شکل ۳۲-۳ بررسی *WHERE*

اگر بخواهیم مثالی از NOT بزنیم می‌توانیم از دستور زیر استفاده کنیم:

```
SELECT * FROM Customers
WHERE NOT Country='Germany' AND NOT Country='USA';
```

در این دستور، اگر کشور مورد نظر آمریکا و آلمان نباشد باید لیست نهایی در خروجی چاپ شود که این موضوع را در شکل ۳۳-۳ مشاهده می‌کنید.



	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country
1	2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico
2	3	Antonio	Juan	Mataderos 2312	México	05023	Mexico
3	4	Thomas	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
4	5	Christina	Christina	Berguvsvägen 8	Luleå	S-958 22	Sweden
5	6	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway
6	7	Farshid	Babajani	Seied Khandan	Tehran	8843	Iran

شکل ۳۳-۳ بررسی دستور *WHERE*

۳-۱-۵-۷ بررسی دستور LIKE

با این دستور می‌توانید یک Pattern ایجاد کنید، یعنی اینکه مثلاً مشخص کنید که ستون‌هایی که با حرف P شروع می‌شوند در خروجی چاپ شوند؛ در زیر شمای کلی این دستور را مشاهده می‌کنید:

```
SELECT column1, column2, ...
FROM table_name
WHERE column LIKE pattern;
```

در این دستور و در شرط مورد نظر به این نکته اشاره شده است که خروجی کار باید طبق Pattern مورد نظر باشد، در لیست زیر چند نمونه از عملگرهای دستور LIKE را مشاهده می‌کنید که با % و _ اجرا شده‌اند.

جدول ۳-۷ عملگر *Like*

عملگر LIKE	توضیحات
LIKE 'a%'	این دستور *مقادیری* را پیدا می‌کند که با حرف "a" شروع می‌شوند.
LIKE '%a'	این دستور *مقادیری* را پیدا می‌کند که به حرف "a" ختم می‌شوند.
LIKE '%or%'	مقادیری را پیدا می‌کند که or داشته باشند.
LIKE '_r%'	این دستور به مقادیری اشاره دارد که حرف دوم آنها r باشد.
LIKE 'a_%'	به مقادیری اشاره دارد که با حرف a شروع شده و دارای حداقل دو حرف است.
LIKE 'a__%'	به مقادیری اشاره دارد که با حرف a شروع شده و حداقل دارای سه کاراکتر هستند.
LIKE 'a%o'	مقادیری را پیدا می‌کنند که با حرف a شروع شده و با حرف o پایان می‌یابد.
LIKE '_2%3'	مقادیری را که مقدار دوم آنها، ۲ و مقدار آخر، ۳ باشد.

مثال اول – **a%**

در این قسمت می‌خواهیم گزینه‌هایی را پیدا کنیم که با حرف a شروع می‌شوند، در دستور زیر قسمت Customer Name بررسی می‌شود و اگر کلمه‌ای با a شروع شود را در خروجی به‌مانند شکل ۳-۳٤ چاپ می‌کند:

```
SELECT * FROM Customers
WHERE "Customer Name" LIKE 'a%';
```

	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country
1	2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico
2	3	Antonio	Juan	Mataderos 2312	México	05023	Mexico

شکل ۳۴-۳ بررسی دستور *LIKE*

مثال دوم – a%

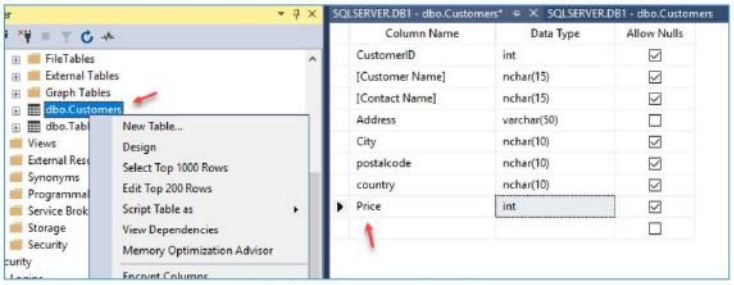
در این مثال در خروجی در ستون Customers Name، کلماتی که به a ختم می‌شوند در خروجی چاپ خواهند شد.

۸-۵-۱-۳ بررسی دستور BETWEEN

این دستور مقدار مشخص‌شده بین دو محدوده را در خروجی نمایش می‌دهید، شکل کلی این دستور به‌صورت زیر می‌باشد:

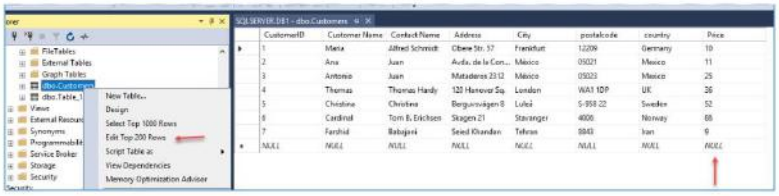
SELECT *column_name(s)*
FROM *table_name*
WHERE *column_name* BETWEEN *value1* AND *value2;*

برای اینکه دستور BETWEEN را اجرا کنیم باید یک ستون جدید به جدول اصلی خود اضافه کنیم؛ برای این کار طبق شکل ۳۵-۳ بر روی جدول مورد نظر خود کلیک راست کنید و قسمت Design را انتخاب کنید و یک ستون با عنوان Price و از نوع int وارد کنید.



شکل ۳۵-۳ بررسی دستور *BETWEEN*

در ادامه باید به‌مانند شکل ۳۶-۳ اطلاعات ستون Price را وارد کنید؛ برای تمرین می‌توانید از طریق دستور INSERT این کار را انجام دهید.

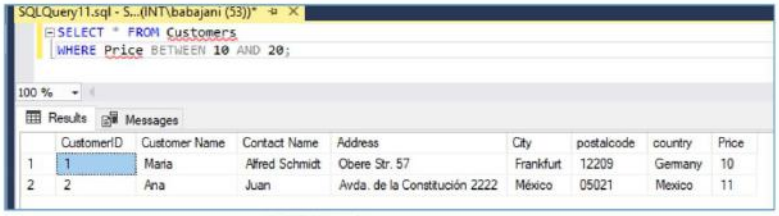


شکل ۳۶-۳ نمایش جدول *Price*

مثال اول –دستور BETWEEN

```
SELECT * FROM Customers
WHERE Price BETWEEN 10 AND 20;
```

در دستور بالا، قیمت بین ۱۰ تا ۲۰ در خروجی نمایش داده خواهد شد که این موضوع را در شکل ۳۷-۳ مشاهده می‌کنید:



	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country	Price
1	1	Maria	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany	10
2	2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico	11

شکل ۳۷-۳ بررسی دستور *BETWEEN*

مثال دوم

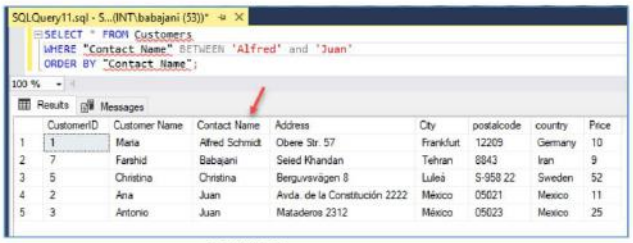
```
SELECT * FROM Customers
WHERE "Contact Name" BETWEEN 'A' and 'Z';
```

در دستور بالا و در ستون Contact Name، اسامی که با حروف A تا Z شروع شوند در خروجی چاپ خواهد شد.

مثال سوم – منظم کردن خروجی

```
SELECT * FROM Customers
WHERE "Contact Name" BETWEEN 'Alfred' and 'Juan'
ORDER BY "Contact Name";
```

در دستور بالا، یک دستور جدید اضافه شده است با نام ORDER BY که این دستور برای منظم کردن ستون خروجی است که این موضوع را در شکل ۳۸-۳ مشاهده می‌کنید.



	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country	Price
1	1	Maria	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany	10
2	7	Farshid	Babajani	Seied Khandan	Tehran	8843	Iran	9
3	5	Christina	Berguvsvägen 8	Luleå	S-958 22	Sweden	52	
4	2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico	11
5	3	Antonio	Juan	Mataderos 2312	México	05023	Mexico	25

شکل ۳۸-۳ بررسی دستور *BETWEE*

۳-۱-۵-۹ بررسی دستور IN

این دستور اصولاً در یک شرط به کار گرفته می‌شود و مشخص می‌کند چه گزینه‌هایی باید در خروجی نمایش داده شود، شمای کلی دستور IN به‌صورت زیر است:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...) ;
```

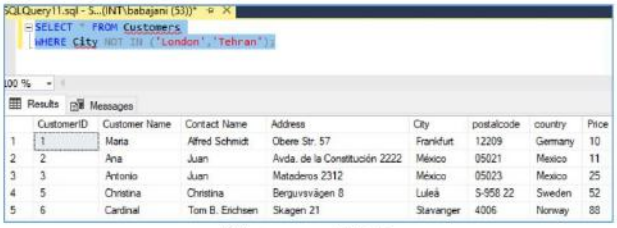
روش دیگری نیز برای به‌کارگیری دستور IN وجود دارد که شکل کلی آن به‌صورت زیر است:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

مثال اول – بررسی دستور IN

```
SELECT * FROM Customers
WHERE City NOT IN ('London','Tehran') ;
```

در مثال بالا، ستون‌هایی در خروجی چاپ خواهد شد که دارای شهرهای London و Tehran نباشند، چون پیش از دستور IN، دستور NOT به کار برده شده است و به این دلیل شرط منفی خواهد شد.



	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country	Price
1	1	Maria	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany	10
2	2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico	11
3	3	Antonio	Juan	Mataderos 2312	México	05023	Mexico	25
4	5	Christina	Christina	Berguvsvägen 8	Luleå	S-958 22	Sweden	52
5	6	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway	88

شکل ۳۹-۳ بررسی دستور *IN*

۳-۱-۵-۱۰ بررسی دستور TOP

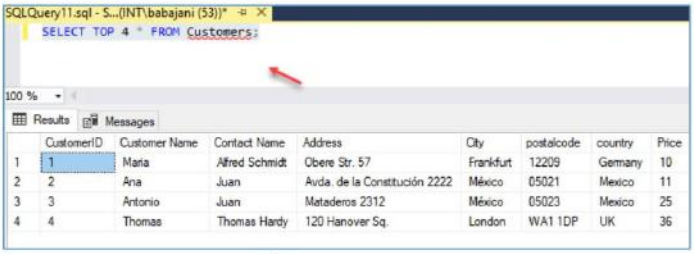
این دستور برای نمایش تعداد سطر در خروجی کاربرد دارد که دارای گزینه‌های مختلفی است و شکل کلی آن را در زیر مشاهده می‌کنید:

```
SELECT TOP number|percent column_name(s)
FROM table_name
WHERE condition;
```

مثال اول – بررسی دستور TOP

```
SELECT TOP 4 * FROM Customers;
```

در دستور بالا، ٤ سطر از جدول Customers به‌مانند شکل ٤٠-٣ در خروجی چاپ خواهد شد.



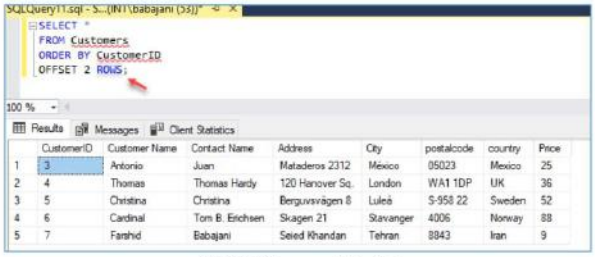
	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country	Price
1	1	Maria	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany	10
2	2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico	11
3	3	Antonio	Juan	Mataderos 2312	México	05023	Mexico	25
4	4	Thomas	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	36

شکل ۴۰-۳ بررسی دستور *TOP*

مثال دوم – استفاده از OFFSET

```
SELECT * FROM Customers
ORDER BY CustomerID
OFFSET 2 ROWS;
```

برای استفاده از دستور OFFSET حتماً باید قبل از آن از دستور ORDER BY استفاده کنید تا ID مورد نظر منظم شود و بعد می‌توانید از دستور OFFSET استفاده کنید؛ توجه داشته باشید که این دستور از شماره‌ای که در مقابل آن نوشته می‌شود در جدول مورد نظر شماره‌های بعد از آن را در خروجی چاپ می‌کنند، مانند دستور بالا که شماره‌ی ۲ نوشته شده است، اما از شماره‌ی ۳ به بعد در خروجی چاپ خواهد شد، اگر به شکل ٤١-٣ توجه کنید این موضوع را مشاهده می‌کنید؛ توجه داشته باشد که دستور ROWS به ردیف‌ها اشاره دارد.



	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country	Price
1	3	Antonio	Juan	Mataderos 2312	México	05023	Mexico	25
2	4	Thomas	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	36
3	5	Christina	Christina	Berguvsvägen 8	Luleå	S-958 22	Sweden	52
4	6	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway	88
5	7	Farshid	Babajani	Seied Khandan	Tehran	8843	Iran	9

شکل ۴۱-۳ بررسی *OFFSET*

مثال سوم – استفاده از دستورات FETCH و PRESENT

```
SELECT * FROM Customers
ORDER BY CustomerID
OFFSET 2 ROWS
FETCH FIRST 3 ROWS ONLY;
```

این دستور از چند قسمت تشکیل شده است؛ در خط اول با دستور SELECT، جدول Customers انتخاب شده و در خط دوم، CustomerID با دستور ORDER BY مرتب می‌شود، بعد از مرتب شدن CustomerID با دستور OFFSET

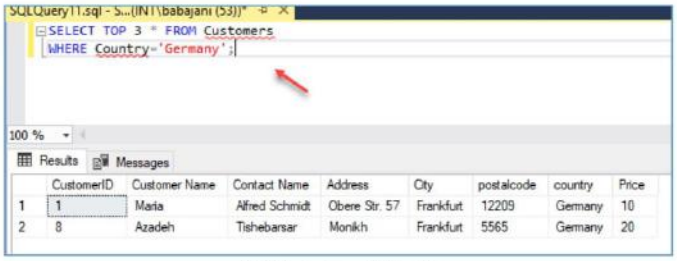
مشخص می‌کنیم که از سطر سوم به بعد در خروجی چاپ شود و در خط آخر نیز مشخص می‌کنیم که چند سطر در خروجی چاپ شود که این موضوع را در شکل ۴۲-۳ مشاهده می‌کنید.

شکل ۴۲-۳ بررسی دستور *FETCH* و *PRESENT*

مثال چهارم – استفاده از WHERE به همراه TOP

```
SELECT TOP 3 * FROM Customers
WHERE Country='Germany';
```

در دستور بالا با استفاده از TOP تعداد خروجی کار مشخص شده است، یعنی اگر سطر مورد نظر کشورش Germany باشد در خروجی چاپ خواهد شد، اما تنها ۳ سطر آن چاپ خواهد شد و اگر بیشتر باشد، چاپ نخواهد شد که در شکل ۴۳-۳ این موضوع را مشاهده می‌کنید.



شکل ۴۳-۳ بررسی دستور *TOP*

۳-۱-۵-۱۱ بررسی دستور MIN and MAX

این دستور کمترین و بیشترین مقدار در یک ستون را برمی‌گرداند که در زیر شکل کلی آن را مشاهده می‌کنید:

MIN

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

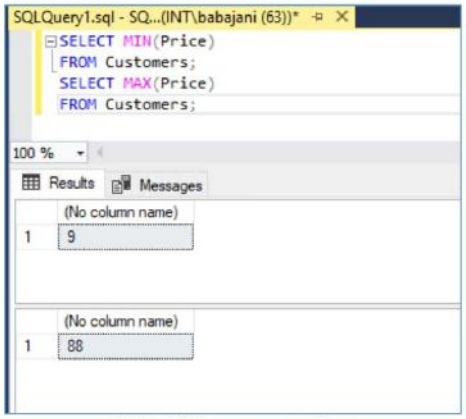
MAX

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

مثال یک – دستور MIN & MAX

```
SELECT MIN(Price)
FROM Customers;
SELECT MAX(Price)
FROM Customers;
```

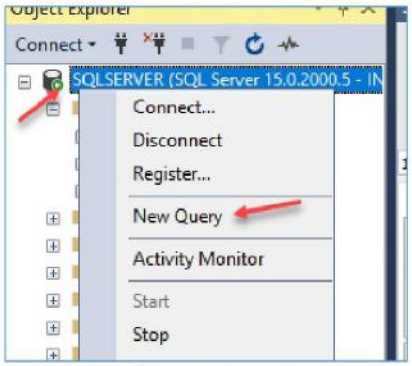
در دستور بالا، دو عدد MIN و MAX از ستون Price در خروجی چاپ خواهد شد که این موضوع را در شکل ٤٤-٣ مشاهده می‌کنید.



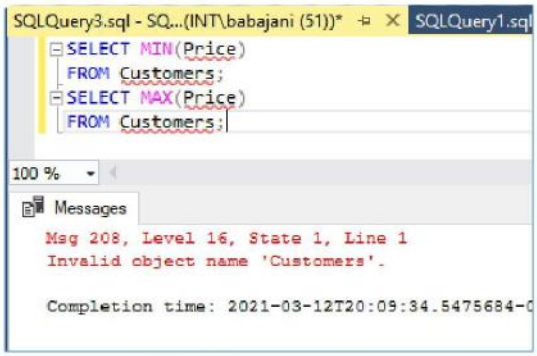
شکل ۴۴-۳ بررسی دستور *MIN* و *MAX*

نکته‌ی مهم:

توجه کنید که ایجاد Query باید در مسیر درست انجام شود، اگر به‌مانند شکل ٤٥-٣ بر روی SQLSERVER کلیک راست کنید و گزینه‌ی New Query را انتخاب کنید و بعد دستورات بالا را در آن اجرا کنید با خطای شکل ٦٤-٣ مواجه خواهید شد که دلیل آن نیز این است که جدول Customers در زیرمجموعه‌ی دیگری قرار دارد و حتماً باید بر روی دیتابیس مورد نظر این دستور را اجرا کنید.



شکل ۴۵-۳ ایجاد *Query*



شکل ۴۶-۳ بررسی دستور *MIN* و *MAX*

۱۲-۵-۱-۳ بررسی دستور COUNT، AVG و SUM

شکل کلی دستور Count به‌صورت زیر است:

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

شکل کلی دستور AVG به‌صورت زیر است:

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

شکل کلی دستور SUM به‌صورت زیر است:

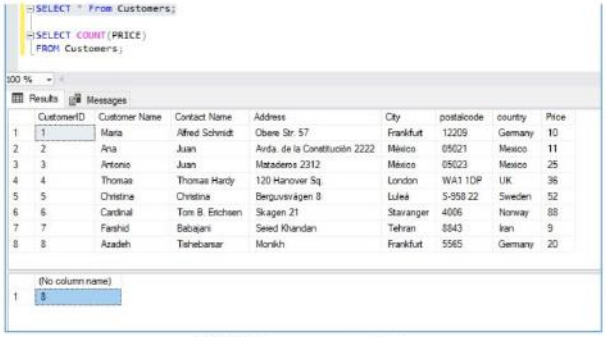
```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

برای هر یک از دستورات بالا، یک مثال را با هم بررسی می‌کنیم:

مثال اول – بررسی دستور Count

```
SELECT * From Customers;

SELECT COUNT(PRICE)
FROM Customers;
```

در خط اول، کل جدول Customers در خروجی چاپ خواهد شد و در خط بعدی با دستور COUNT، تعداد موجودی در ستون Price مشخص خواهد شد که در شکل ۴۷-۳ این موضوع را مشاهده می‌کنید؛ در قسمت دوم عدد ۸ نوشته شده است که تعداد موجودی در ستون Price است.

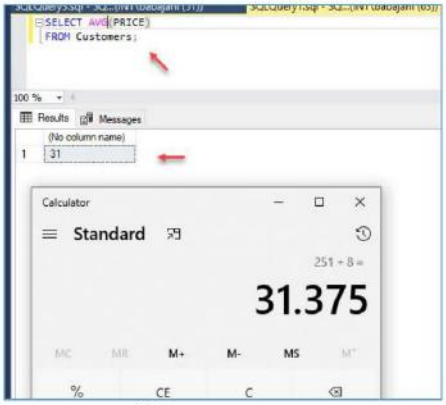


شکل ۴۷-۳ بررسی دستور *COUNT*

مثال دوم – بررسی دستور AVG

```
SELECT AVG(PRICE)
FROM Customers;
```

در دستور بالا، میانگین اعدادی ستون Price حساب شده است و در خروجی چاپ خواهد شد که این موضوع را در شکل ۴۸-۳ مشاهده می‌کنید، اگر توجه کنید با ماشین حساب جمع کل ستون Price را محاسبه و تقسیم بر ۸ کردیم که نتیجه‌ی آن عدد ۳۱ شده است.

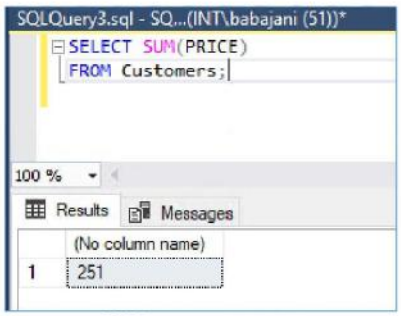


شکل ۴۸-۳ بررسی دستور *AVG*

مثال سوم – بررسی دستور SUM

```
SELECT SUM(PRICE)
FROM Customers;
```

این دستور، کل ستون Price را جمع خواهد کرد که در شکل ۴۹-۳ این موضوع را مشاهده می‌کنید.



شکل ۴۹-۳ بررسی دستور *SUM*

۱۳-۵-۱-۳ بررسی Wildcards

این دستور در کنار دستور LIKE استفاده می‌شود و برای جستجو و جایگزینی در یک رسته کاربرد دارند، در زیر کاراکترهایی که می‌توان به‌عنوان Wildcard در دستورات استفاده کرد.

جدول ۸-۳ بررسی *Wildcard*

Symbol	توضیحات	مثال
%	صفر یا اکثر نویسه‌ها را نشان می‌دهد.	bl% کلماتی مانند bl, black, blue, و blob را پیدا می‌کند.
_	نماینده‌ی یک شخصیت واحد است	دستور h_t کلماتی مانند hot, hat, و hit را پیدا می‌کند.
[]	هر کاراکتر را در داخل براکت نشان می‌دهد.	دستور h[oa]t کلمات hot و hat را نشان می‌دهد، اما کلمه‌ی hit را نمایش نمی‌دهد.
^	هر کاراکتری که در پرانتز نیست را نشان می‌دهد.	دستور h[^oa]t هر کلمه‌ای که در کروشه نباشد را نشان می‌دهد، مانند hit .
-	طیف وسیعی از نویسه‌ها را نشان می‌دهد.	دستور c[a-b]t کلمات cat و cbt را پیدا می‌کند.

مثال اول

```
SELECT * FROM Customers
WHERE City LIKE 'fr%';
```

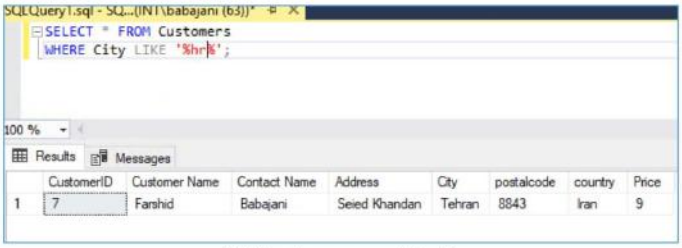
در دستور بالا، شهرهایی که با نام fr شروع می‌شوند در خروجی چاپ خواهند شد که در شکل ۵۰-۳ مشاهده می‌کنید.

شکل ۵۰-۳ بررسی *Wildcards*

مثال دوم

```
SELECT * FROM Customers
WHERE City LIKE '%hr%';
```

دستور بالا، شهرهایی که بین کلماتشان hr دارند در خروجی چاپ خواهد شد که در شکل ۵۱-۳ این موضوع را مشاهده می‌کنید:

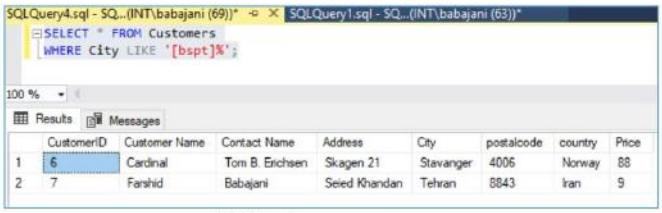


شکل ۵۱-۳ بررسی دستور *Wildcards*

مثال سوم

```
SELECT * FROM Customers
WHERE City LIKE '[bspt]%';
```

در دستور بالا، شهرهایی که با یکی از حروف bspt شروع شوند در خروجی چاپ خواهد شد که شکل ۵۲-۳ نمایانگر این موضوع است.

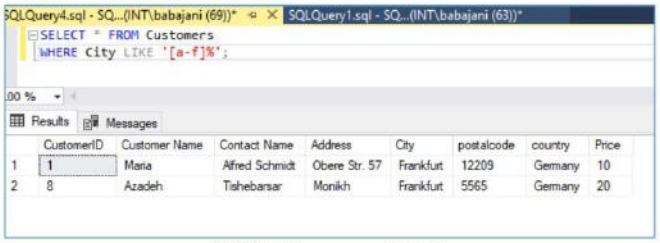


شکل ۵۲-۳ بررسی دستور *Wildcards*

مثال چهارم

```
SELECT * FROM Customers
WHERE City LIKE '[a-f]%';
```

در دستور بالا، شهرهایی که با حروف a تا f شروع می‌شوند در خروجی چاپ خواهد شد که در شکل ۵۳-۳ مشخص شده است.



شکل ۵۳-۳ بررسی دستور *Wildcards*

۱۴-۵-۱-۳ بررسی دستور Aliass

این دستور برای ایجاد یک نام مستعار برای ستون‌های یک جدول کاربرد دارد تا خوانایی آن راحت‌تر شود؛ این نام تنها در زمان جستجو ایجاد می‌شود و بعد از آن حذف خواهد شد، این دستور را باید با AS اجرا کرد که شکل کلی آن به‌صورت زیر است:

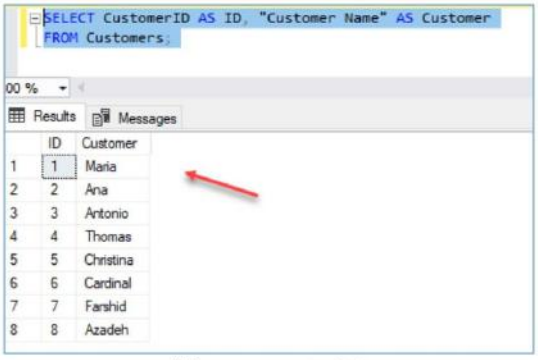
SELECT *column_name* AS *alias_name*
FROM *table_name;*

در دستور بالا، alias_name یک نام مستعار برای جدول مورد نظر است خواهد بود.

مثال اول

```
SELECT CustomerID AS ID, "Customer Name" AS Customer
FROM Customers;
```

در دستور بالا، دو ستون CustomerID و Customer Name با دو نام جدید در خروجی چاپ خواهند شد که در شکل ۵۴-۳ این موضوع را مشاهده می‌کنید.

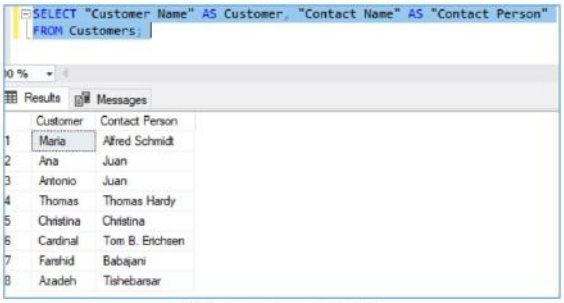


شکل ۵۴-۳ بررسی دستور *Aliass*

مثال دوم

```
SELECT "Customer Name" AS Customer, "Contact Name" AS "Contact Person"
FROM Customers;
```

این دستور نیز به‌مانند دستور قبلی است و نکته‌ای که در این دستور وجود دارد، این است که برای ایجاد یک ستون دو حرفی باید کلمه‌ی مورد نظر را در یک کروشه قرار دهید؛ این موضوع را در شکل ۵۵-۳ مشاهده می‌کنید.

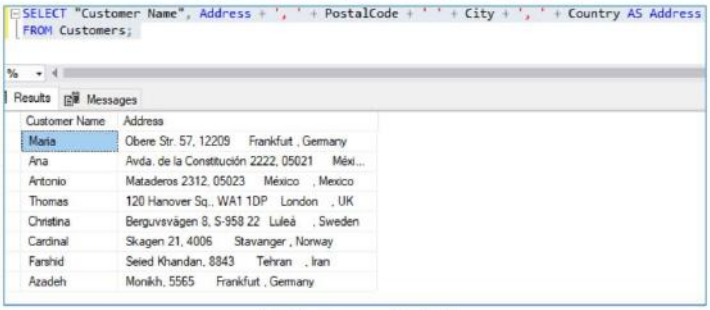


شکل ۵۵-۳ بررسی دستور *Aliass*

مثال سوم

```
SELECT "Customer Name", Address + ', ' + PostalCode + ' ' + City + ', ' + Country AS
Address
FROM Customers;
```

در دستور بالا، دو ستون در خروجی چاپ خواهد شد که در ستون Address، چند گزینه چاپ خواهد شد که شامل Address, PostalCode, City و کشور خواهد بود که در شکل ۵۶-۳ این موضوع را مشاهده می‌کنید.



شکل ۵۶-۳ بررسی دستور *Aliass*

۱۵-۵-۱-۳ بررسی دستور GROUP BY

دستور GROUP BY برای جمع‌بندی مقادیر یکسان کاربرد دارد و آنها را توسط تابع‌های COUNT(), MAX(), MIN(), SUM(), AVG() در خروجی چاپ خواهد کرد، شکل کلی این دستور به‌صورت زیر خواهد بود:

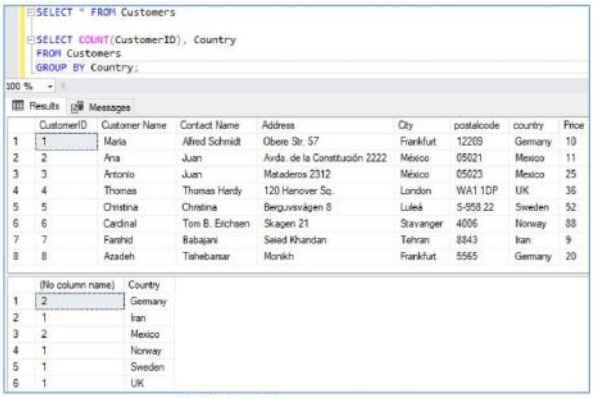
```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

مثال اول

```
SELECT * FROM Customers
```

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

در دستور بالا، در خط اول یک خروجی کلی از اطلاعات جدول Customers را خواهیم داشت و در ادامه با استفاده از دستور COUNT، تعداد مشتریان هر کشور شمارش می‌شود و در آخر با دستورGROUP BY، تعداد مشتریان به همراه کشورهای آنها چاپ شده است؛ در شکل ۵۷-۳ این موضوع را مشاهده می‌کنید.



	CustomerID	Customer Name	Contact Name	Address	City	postalcode	country	Price
1	1	Maria	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany	10
2	2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico	11
3	3	Antonio	Juan	Mataderos 2312	México	05023	Mexico	25
4	4	Thomas	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	36
5	5	Christina	Christina	Berguvsvägen 8	Luleå	S-958 22	Sweden	52
6	6	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway	88
7	7	Farshid	Babajani	Seied Khandan	Tehran	8843	Iran	9
8	8	Azadeh	Tshebarsar	Monikh	Frankfurt	5565	Germany	20

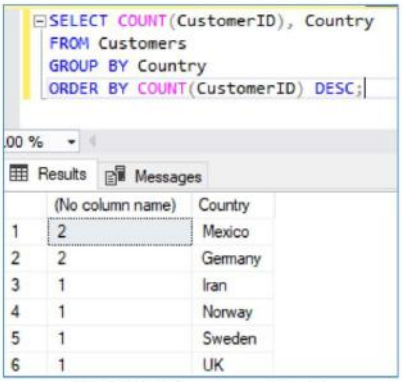
	(No column name)	Country
1	2	Germany
2	1	Iran
3	2	Mexico
4	1	Norway
5	1	Sweden
6	1	UK

شکل ۵۷-۳ بررسی دستور *GROUP BY*

مثال دوم

```
SELECT * FROM Customers

SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country;
```

در دستور بالا به‌مانند دستورات قبلی مشتریان مورد نظر چاپ خواهد شد، اما در آخر خط با دستور ORDER BY به همراه دستور DESC از بیشتر به کمتر مشتریان مرتب خواهند شد که در شکل ۵۸-۳ این موضوع را مشاهده می‌کنید.



	(No column name)	Country
1	2	Mexico
2	2	Germany
3	1	Iran
4	1	Norway
5	1	Sweden
6	1	UK

شکل ۵۸-۳ بررسی دستور *GROUP BY*

۳-۱-۵-۱۶ بررسی دستور HAVING

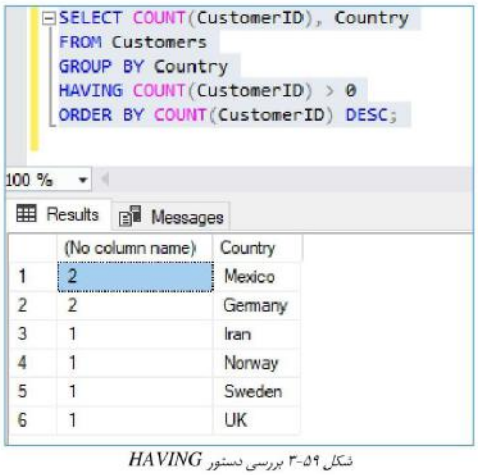
این دستور، یک دستور کمکی است که می‌تواند در دستورات اضافه شود؛ شکل کل این دستور به‌صورت زیر است:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

مثال اول

```
SELECT COUNT(CustomerID), Country
FROM Customers
GROUP BY Country
HAVING COUNT(CustomerID) > 0
ORDER BY COUNT(CustomerID) DESC;
```

در مثال بالا، عدد صفری که مقابل دستور HAVING قرار داده شده به این معنی است که تعداد مشتریان هر کشور اگر بیشتر از صفر بود، در خروجی چاپ کند. شما می‌توانید این عدد را تغییر دهید و یا اینکه عملگر آن را عوض کنید، در شکل ۳-۵۹ این مثال را مشاهده می‌کنید.

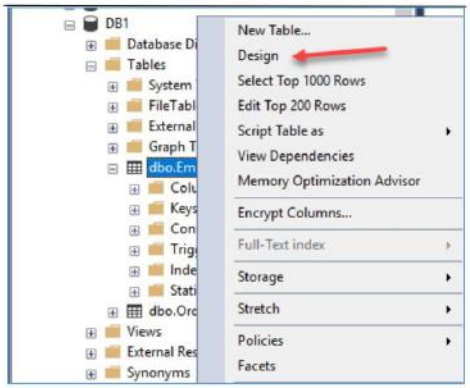


	(No column name)	Country
1	2	Mexico
2	2	Germany
3	1	Iran
4	1	Norway
5	1	Sweden
6	1	UK

شکل ۳-۵۹ بررسی دستور *HAVING*

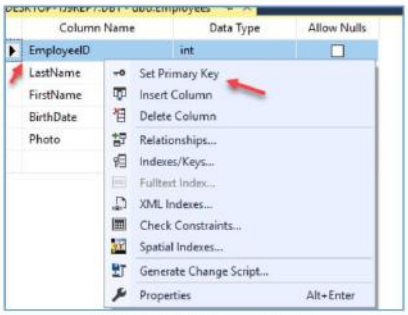
۳-۱-۵-۱۷ بررسی PRIMARY KEY

زمانی که یک جدول ایجاد می‌کنید، می‌توانید مشخص کنید که کدام ستون به‌عنوان PRIMARY KEY انتخاب شود؛ انتخاب یک ستون به‌عنوان PRIMARY KEY باعث می‌شود که مقدار آن دیگر، Null یا خالی نباشد و حتماً باید آن را پر کنید؛ برای این کار به‌مانند شکل ۳-۶۰ بر روی جدول مورد نظر کلیک راست کنید و گزینه‌ی Design را انتخاب کنید.



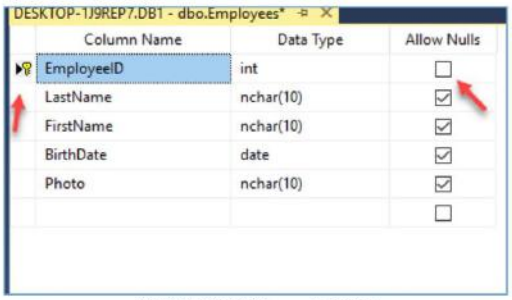
شکل ۶۰-۳ طراحی

به‌مانند شکل ٦١-٣ بر روی ستونی که می‌خواهید به‌عنوان Primary Key انتخاب شود، کلیک راست کنید و گزینه‌ی Set Primary Key را انتخاب کنید.



شکل ۶۱-۳ بررسی *PRIMARY KEY*

در بعد از انتخاب Primary Key در شکل ٦٢-٣ یک کلید در کنار ستون مورد نظر قرار خواهد گرفت و در قسمت Allow Nulls، تیک آن برداشته خواهد شد؛ به دلیل اینکه این ستون اصلاً نباید خالی باشد، مانند شماره‌ی دانشجویی در دانشگاه‌ها.

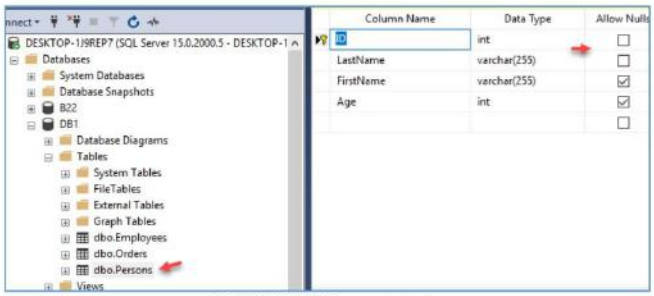


شکل ۶۲-۳ بررسی *PRIMARY KEY*

برای اینکه بتوانیم از طریق دستورات T-SQL یک جدول ایجاد کنیم که دارای Primary Key باشد باید دستورات زیر را اجرا کنیم:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (ID)
) ;
```

در دستورات بالا، یک جدول جدید با نام Persons ایجاد کردیم که ستون‌های ID و LastName نباید خالی باشد و در قسمت آخر PRIMARY KEY را نیز برابر ID قرار دادیم، اگر به شکل ۳-۶۳ توجه کنید نتیجه‌ی دستور را مشاهده خواهید کرد.



شکل ۳-۶۳ بررسی *PRIMARY KEY*

اگر بخواهیم دستور بالا را تغییر دهیم، می‌توانیم از دستور زیر استفاده کنیم:

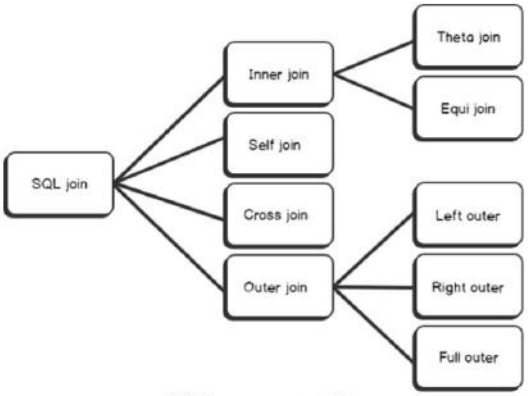
```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
) ;
```

در دستور بالا، Primary Key در کنار ID قرار گرفته شده است.

۳-۱-۵-۱۸ بررسی JOIN در SQL

یکی از مهم‌ترین دستوراتی که در SQL می‌توان نام برد، دستور JOIN است که این دستور اساساً برای ترکیب و بازیابی داده‌ها از دو یا چند جدول مورد استفاده قرار می‌گیرد. در یک پایگاه‌داده، رابطه‌ای در دنیای واقعی، داده‌ها در تعداد زیادی جداول ساختار می‌یابند و به این دلیل، دائماً باید به این جداول متصل شد. چهار نوع اساسی عضویت در SQL Server وجود دارد: Inner، Outer، Self و Cross join. برای اینکه یک مرور سریع در مورد همه‌ی این پیوستن‌ها داشته باشید در زیر آنها را بررسی می‌کنیم:

در شکل ۳-۶۴ یک شمای کلی از نحوه‌ی ارتباط دستورات JOIN را مشاهده می‌کنید که هر کدام توضیح مخصوص به خود را دارند.



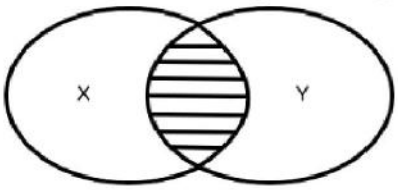
شکل ۶۴-۳ بررسی دستور *JOIN*

توانایی ترکیب نتایجِ حاصل از سطرهای مرتبط از چندین جدول، قسمت مهمی در طراحی سیستم پایگاه داده‌ای رابطه‌ای است. در SQL Server این امر با دستور join انجام می‌شود. این ماهیت سیستم‌های پایگاه‌داده‌ای رابطه‌ای سنتی است که برخی از جدول‌ها حاوی اطلاعات مربوط به جداول دیگر با مقدار کلیدی مشترک هستند. با استفاده از پیوستن SQL، می‌توانید به‌راحتی از طریق چندین جدول با این کلیدهای مشترک، پرس‌وجوهایی را در مجموعه داده‌های مرتبط انجام دهید.

در ادامه نوع‌های مختلف SQL JOIN را که در شکل ۴۳-۳ مشخص شده است را با هم بررسی می‌کنیم:

۱- SQL inner join

این دستور یکی از ساده‌ترین دستورات JOIN است که نتیجه‌ی خروجی آن، سطری از هر دو جدول است که در آن شرایط پیوستگی یکی باشد؛ در شکل ۶۵-۳ شمای کلی این دستور را مشاهده می‌کنید.



شکل ۶۵-۳ *SQL inner join*

دستورات کلی Inner Join به‌صورت زیر است:

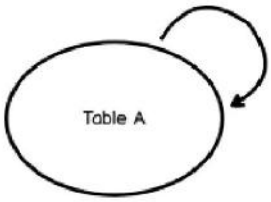
```
SELECT ColumnList from LeftTable L
INNER join  RightTable R
ON L.Column=R.Column
```

۲- SQL self join

این دستور باعث می‌شود، یک جدول با خودش Join شود که در ادامه، نحوه‌ی کار آن را بررسی خواهیم کرد.

شکل ۶۶-۳ نمای کلی دستور به‌صورت زیر است:

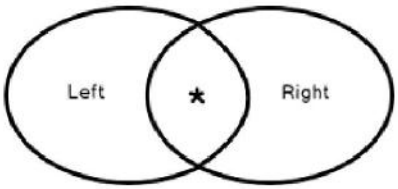
```sql
SELECT column_name(s)
FROM table1 T1, table1 T2
WHERE condition
```



شکل ۳-۶۶ *SQL self join*

۳- SQL cross join

این نوع JOIN حاصل‌ضرب دکارتی سطرهای تعیین‌شده در دستور join را در خروجی نمایش می‌دهد. این دستور، سطرهایی را تولید می‌کند که ترکیبی از سطر جدول اول و دوم است که نمای کلی آن را در شکل ۳-۶۷ مشاهده می‌کنید.



شکل ۳-۶۷ *SQL cross Join*

۴- SQL outer join

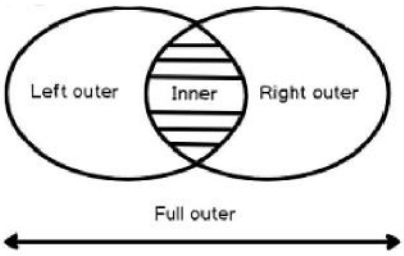
این دستور شامل سه دستور LEFT OUTER, RIGHT OUTER, FULL است که هرکدام برای کار خاص طراحی شده‌اند.

دستور LEFT OUTER JOIN

شکل کلی این دستور به‌صورت زیر است:

```sql
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name = table2.column_name;
```

این دستور، تمام اطلاعات را از جدول سمت چپ، یعنی جدول اول و اطلاعات منطبق را از جدول دوم در خروجی چاپ می‌کند و اگر هم عدم تطابق ایجاد شود، نتیجه‌ی صفر را از جدول سمت راست برمی‌گرداند؛ در شکل ۳-۶۸ این موضوع مشخص شده است.



شکل ۶۸-۳ *SQL Full outer*

مثال دستور INNER JOIN

برای اینکه بیشتر با این دستورات آشنا شویم یک مثال را با هم بررسی می‌کنیم؛ برای شروع نیاز به دو جدول داریم تا بتوانیم عملیات JOIN را بر روی آنها انجام دهیم.

جدول ۹-۳ *Customers*

CustomerID	Customer Name	Contact Name	Address	City	postalcode	country	Price
1	Maria	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany	10
2	Ana	Juan	Avda. de la Constitución 2222	México	05021	Mexico	11
3	Antonio	Juan	Mataderos 2312	México	05023	Mexico	25
4	Thomas	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK	36
5	Christina	Christina	Berguvsvägen 8	Luleå	S-95822	Sweden	52
6	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway	88
7	Farshid	Babajani	Seied Khandan	Tehran	8843	Iran	9
8	Azadeh	Tishebarsar	Monikh	Frankfurt	5565	Germany	20

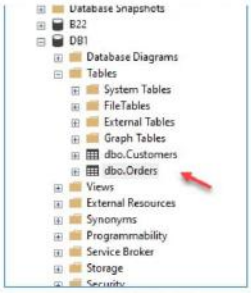
جدول *Order*

جدول ۱۰-۳ *Order*

OrderID	CustomerID	OrderDate
10308	8	1996-09-18
10309	6	1996-09-19
10310	4	1996-09-20

دو جدول ۹-۳ و ۱۰-۳ را مشاهده می‌کنید که باید در SQL آنها را ایجاد کنیم تا بتوانیم عملیات Join را بر روی آنها انجام دهیم.

همان طور که در شکل ۶۹-۳ مشاهده می‌کنید، دو جدول مورد نظر در SQL ایجاد شده‌اند.



شکل ۳-۶۹ ایجاد جدول

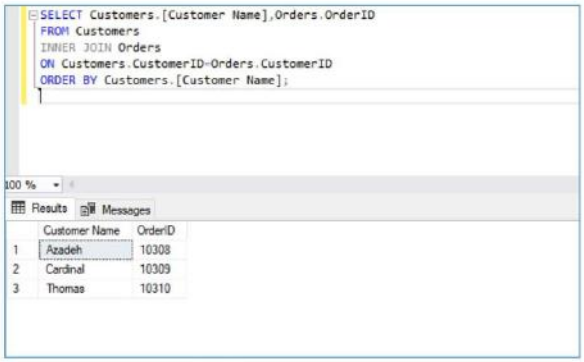
با استفاده از دستور زیر می‌توانیم بین دو جدول ارتباط داخلی برقرار کنیم:

```
SELECT Customers.[Customer Name],Orders.OrderID
FROM Customers
INNER JOIN Orders
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.[Customer Name];
```

در خط اول با دستور SELECT، ستون‌های خروجی را که قرار است چاپ شوند را مشخص می‌کنیم که ستون Customer Name از جدول Customers و ستون OrderID از جدول Orders چاپ خواهد شد.

در خط دوم جدول Customers انتخاب می‌شود و در خط سوم مشخص می‌کنیم که باید با جدول Orders ارتباط INNER JOIN برقرار کند؛ در خط چهارم باید شرط برقراری ارتباط را مشخص کنیم که با دستور ON این کار را انجام می‌دهیم و بعد از آن، CustomerID از جدول Customers را با ستون CustomerID از جدول Orders را مساوی قرار می‌دهیم؛ این بدان معنا است که اگر در این دو ستون، عبارت مساوی هم قرار داشته باشد، آن سطر در خروجی چاپ خواهد شد؛ در خط آخر نیز با دستور ORDER BY، ستون Customer Name را منظم می‌کنیم تا در خروج، نمای بهتری داشته باشد.

اگر به شکل ۳-۷۰ توجه کنید، مشاهده خواهید کرد سه سطر در خروجی اپ شده است و آن هم به دلیل برابر بودن اعداد در ستون‌های Customer ID در هر دو جدول است.



شکل ۳-۷۰ بررسی دستور JOIN

نکته: دستور JOIN و INNER JOIN یک کار را انجام می‌دهند، اما دستور INNER JOIN دارای عملکرد بهتری نسبت به دستور JOIN است و کار را در سریع‌ترین زمان ممکن انجام می‌دهد.

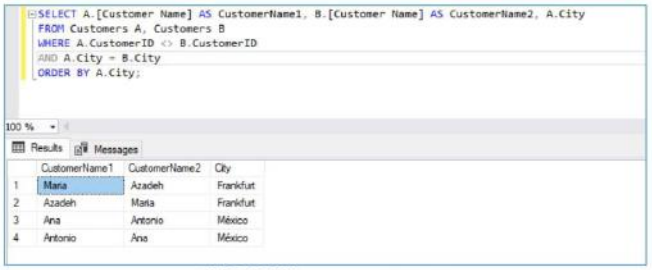
مثال دستور SELF JOIN

همان طور که گفتیم این دستور با خودش ارتباط برقرار می‌کند؛ برای تست این موضوع از جدول Customers استفاده می‌کنیم و دستور زیر را اجرا می‌کنیم:

```
SELECT A.[Customer Name] AS CustomerName1, B.[Customer Name] AS CustomerName2, A.City
FROM Customers A, Customers B
WHERE A.CustomerID <> B.CustomerID
AND A.City = B.City
ORDER BY A.City;
```

در دستور بالا و در خط اول، سه ستون در خروجی چاپ خواهد شد که دستور AS، یک ستون جدید در خروجی نمایش می‌دهد؛ در قسمت FROM، یک جدول به دو قسمت تبدیل شده است که یکی Customers A و دیگری Customers B است؛ در قسمت WHERE شرطی قرار دادیم و گفتیم که در صورتی که A.CustomerID برابر نباشد با B.CustomerID و همچنین شهر جدول A با شهر جدول B یکی باشد شرط درست است و باید در خروجی چاپ شود که در خط آخر نیز طبق ستون A.City خروجی منظم خواهد شد.

در شکل ۷۱-۳ خروجی دستورات بالا را مشاهده می‌کنید که سه ستون ایجاد و خروجی طبق شرطی که قرار دادیم، نمایش‌داده شده است.



شکل ۷۱-۳ بررسی دستور *SELF JOIN*

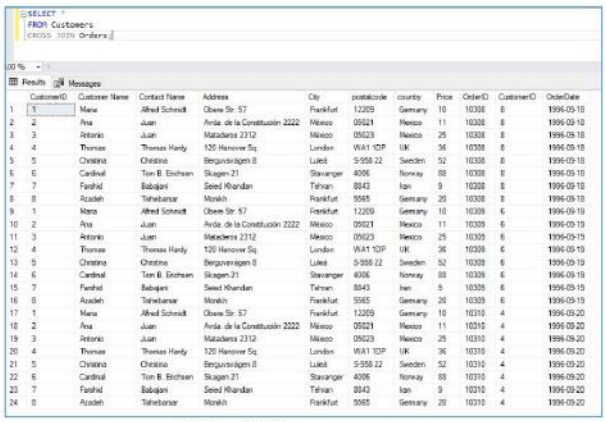
مثال دستور CROSS JOIN

همان طور که گفتیم، این دستور با استفاده از ضرب دکارتی خروجی را مشخص می‌کند.

```
SELECT *
FROM Customers
CROSS JOIN Orders;
```

در دستور بالا با دستور * SELECT باید همه‌ی ستون‌های جدول Customers در خروجی نمایش داده شوند، اما در خط آخر با دستور CROSS JOIN، تمام سطرهای جدول اول را ضرب در جدول Orders می‌کنیم که خروجی مانند شکل ۷۲-۳ خواهد شد.

همان طور که در شکل ۷۲-۳ مشاهده می‌کنید، هر کدام از سطرهای جدول Customers ضرب در هر کدام از سطرهای جدول Orders خواهد شد که نتیجه‌ی آن یک جدول بزرگ از اطلاعات است.



شکل ۷۲-۳ خروجی دستور *CROSS JOIN*

مثال دستور OUTER JOIN

این دستور همان طور که گفتیم از چندین قسمت تشکیل شده:LEFT OUTER, RIGHT OUTER, FULL OUTER که آنها را با هم بررسی می‌کنیم.

مثال دستور LEFT JOIN

دستور زیر را اجرا کنید:

```
SELECT Customers.[Customer Name], Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.[Customer Name];
```

در دستور بالا، در خط اول دستور SELECT ستون Customer Name را از جدول Customers و ستون OrderID را از جدول Orders در خروجی چاپ خواهد کرد؛ در خط دوم جدول Customers انتخاب و در خط سوم با استفاده از دستور LEFT JOIN می‌گوییم که ستون Customers.CustomerID برابر با Orders.CustomerID باشد؛ این دستور تمام اطلاعات جدول Customers را در صورت تطبیق در خروجی چاپ خواهد کرد و اگر در جدول دوم، یعنی Orders مقداری وجود نداشته باشد، کلمه‌ی null را برمی‌گرداند.

همان طور که در شکل ۷۴-۳ مشاهده می‌کنید، خروجی به درستی نمایش داده شده است.

```
SELECT Customers.[Customer Name], Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
ORDER BY Customers.[Customer Name];
```

100 %

Results | Messages

	Customer Name	OrderID
1	Ana	NULL
2	Antonio	NULL
3	Azadeh	10308
4	Cardinal	10309
5	Christina	NULL
6	Farshid	NULL
7	Maria	NULL
8	Thomas	10310

شکل ۳-۷۴ بررسی دستور *OUTER JOIN*

مثال دستور RIGHT JOIN

برای انجام این دستور، دو جدول ۳-۱۱ و ۳-۱۲ را در SQL ایجاد کنید:

جدول ۳-۱۱ *Orders*

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

جدول ۳-۱۲ *Employees*

EmployeeID	LastName	FirstName	BirthDate	Photo
1	Davolio	Nancy	12/8/1968	EmpID1.pic
2	Fuller	Andrew	2/19/1952	EmpID2.pic
3	Leverling	Janet	8/30/1963	EmpID3.pic

بعد از ایجاد جدول باید دستورات زیر را اجرا کنید:

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

در خط اول، سه ستون از دو جدول مشخص شده است و در ادامه به جدول Orders و جدول Employess متصل می‌شود که شرط RIGHT JOIN در جدول Employess پیاده‌سازی شده است، با این شرایط که Orders.EmployeeID = Employees.EmployeeID باشد؛ در شکل ۳-۷۵ این موضوع را مشاهده می‌کنید.

```
SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
FROM Orders
RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY Orders.OrderID;
```

00 %

Results | Messages

	OrderID	LastName	FirstName
1	NULL	Davolio	Nancy
2	NULL	Fuller	Andrew
3	10309	Leverling	Janet

شکل ۳-۷۵ بررسی دستور *RIGHT JOIN*

مثال دستور Full JOIN

این دستور به طور کامل، خروجی هر دو جدول را طبق شرطی که در دستور زیر مشاهده می‌کنید، نمایش می‌دهد.

```
SELECT Employees.FirstName, Orders.OrderID
FROM Employees
FULL OUTER JOIN Orders ON Employees.EmployeeID=Orders.CustomerID
ORDER BY Orders.EmployeeID;
```

```
SELECT Employees.FirstName, Orders.OrderID
FROM Employees
FULL OUTER JOIN Orders ON Employees.EmployeeID=Orders.CustomerID
ORDER BY Orders.EmployeeID;
```

00 %

Results | Messages

	FirstName	OrderID
1	Nancy	NULL
2	Janet	NULL
3	NULL	10309
4	Andrew	10308

شکل ۳-۷۶ بررسی دستور *Full JOIN*

۳-۱-۵-۱۹ بررسی دستور synonym

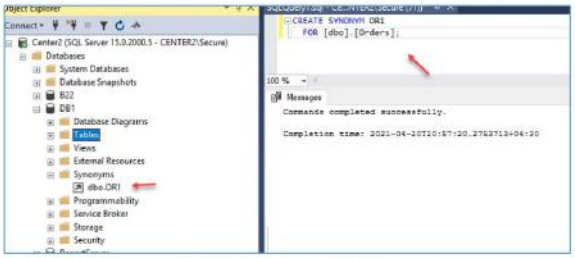
با دستور synonym می‌توانید برای جداول خود مترادف درست کنید، یعنی می‌توانید به جای نام اصلی جداول و View، یک نام جدید به آنها تخصیص دهید و در کد اجرا کنید؛ در زیر شکل کلی این دستور را مشاهده می‌کنید.

```
CREATE SYNONYM synonym_name
FOR object;
```

برای اینکه بیشتر با این دستور آشنا شویم، دستور زیر را در SQL اجرا می‌کنیم:

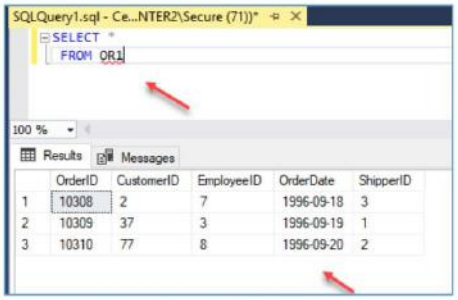
```
CREATE SYNONYM OR1
  FOR [dbo].[Orders];
```

با استفاده از دستور CREATE SYNONYM، یک نام جدید با عنوان OR1 برای جدول Orders که در جلوی دستور For قرار دادیم ایجاد می‌شود، اگر به شکل ۷۷-۳ دقت کنید بعد از اجرای دستور در قسمت Synonyms، نام جدید شما ایجاد شده است و حالا می‌توانید به‌مانند جدول اصلی از آن استفاده کنید.



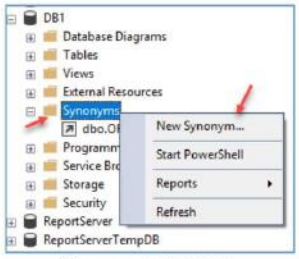
شکل ۷۷-۳ بررسی دستور Synonym

برای اینکه از این جدول خروجی تهیه کنید، می‌توانید به‌مانند شکل ۷۸-۳ عمل کنید؛ همان طور که مشاهده می‌کنید با وارد کردن نام OR1 در جلوی دستور FROM، لیست جدول ORDERS در خروجی چاپ شده است؛ یکی از مزایای استفاده از دستور SYNONYM، خلاصه کردن و مشخص کردن اسم مشخص برای جداول و کوتاه کردن دستورات است.



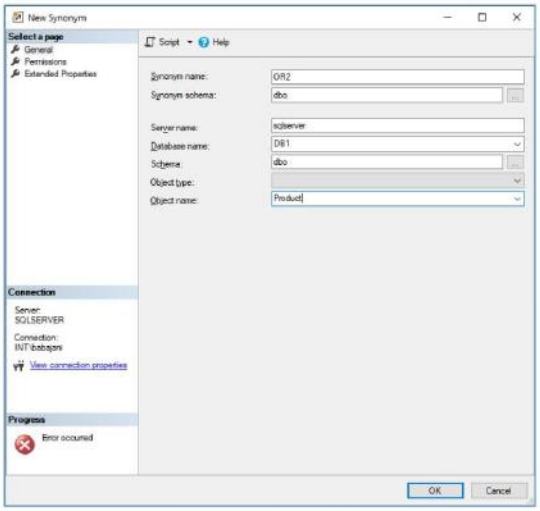
شکل ۷۸-۳ خروجی جدول

خوب اگر بخواهید از طریق GUI، یک SYNONYM ایجاد کنید باید به‌صورت شکل ۷۹-۳ بر روی Synonyms کلیک راست کنید و گزینه‌ی New Synonyms را انتخاب کنید.



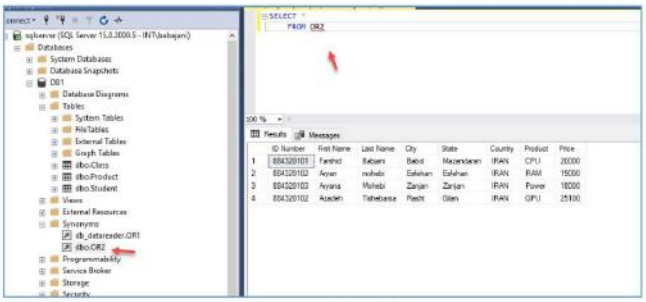
شکل ۷۹-۳ ایجاد Synonym

در شکل ۸۰-۳ و در قسمت Synonyms Name باید نام دلخواه خود را وارد کنید و در قسمت Synonyms schema باید نوع دسترسی آن را مشخص کنید که با وارد کردن dbo، یعنی owner دسترسی کامل به جدول خواهید داشت؛ در قسمت Database name باید نام دیتابیسی را وارد کنید که قرار است جدول مورد نظر را از آن انتخاب کنید و در قسمت Object name نیز باید نام جدول مورد نظر را وارد و بر روی OK کلیک کنید.



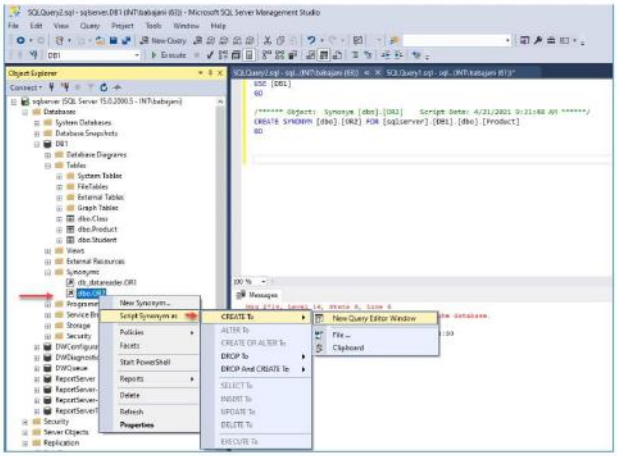
شکل ۸۰-۳ ایجاد *synonym*

همان طور که در شکل ۸۱-۳ مشاهده می‌کنید، Synonyms مورد نظر ایجاد شده و خروجی آن نیز جدول Product است.



شکل ۸۱-۳ خروجی جدول

اگر بخواهید بعد از ایجاد synonyms کد آن را به دست آورید، می‌توانید به‌مانند شکل ۸۲-۳ بر روی synonyms مورد نظر کلیک راست کنید و از قسمت Script و بعد، Create to گزینه‌ی New Query Editor را انتخاب کنید تا کد مورد نظر نمایش داده شود که البته بعد از اجرا به شما خطا خواهد داد که synonyms از قبل وجود دارد.



شکل ۲-۸-۳ ایجاد کد Synonym

۶-۱-۳ کار با View در SQL

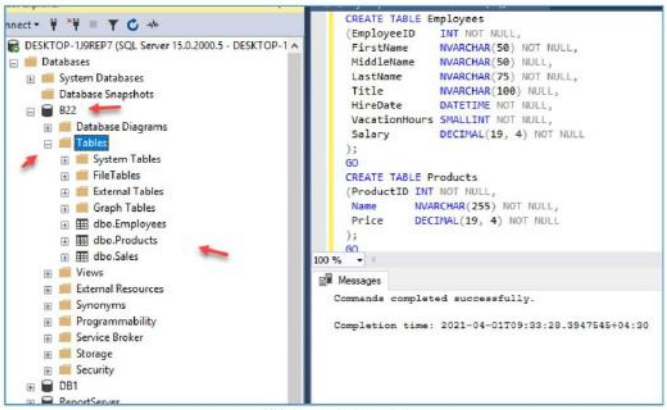
Viewها یک سری از جداول هستند که می‌توانند به‌صورت دقیق‌تر و بهتر نتیجه‌ی دستور SELECT را در خروجی نمایش دهند، البته به این موضوع توجه کنید که VIEW شامل جداول مجازی هستند و ربطی به جدول اصلی در دیتابیس ندارند و به این دلیل هیچ چیزی را در خودشان ذخیره نمی‌کنند، یعنی تنها برای نمایش خروجی دستور SELECT است و اگر هم حذف شوند، تأثیری روی جداول اصلی ندارند.

برای شروع کار نیاز به جداولی داریم تا بتوانیم View را بر روی آنها پیاده‌سازی کنیم؛ برای این کار دستور زیر را اجرا کنید:

```
CREATE TABLE Employees
(EmployeeID    INT NOT NULL,
 FirstName     NVARCHAR(50) NOT NULL,
 MiddleName    NVARCHAR(50) NULL,
 LastName      NVARCHAR(75) NOT NULL,

Title         NVARCHAR(100) NULL,
 HireDate      DATETIME NOT NULL,
 VacationHours SMALLINT NOT NULL,
 Salary        DECIMAL(19, 4) NOT NULL
);
GO
CREATE TABLE Products
(ProductID INT NOT NULL,
 Name      NVARCHAR(255) NOT NULL,
 Price     DECIMAL(19, 4) NOT NULL
);
GO
CREATE TABLE Sales
(SalesID    UNIQUEIDENTIFIER NOT NULL,
 ProductID  INT NOT NULL,
 EmployeeID INT NOT NULL,
 Quantity   SMALLINT NOT NULL,
```

```
 SaleDate    DATETIME NOT NULL
);
GO
```

با اجرای دستورات بالا، سه جدول با نام‌های Employees, Products, Sales ایجاد می‌شود که در شکل ۸۳-۳ این موضوع را مشاهده می‌کنید.



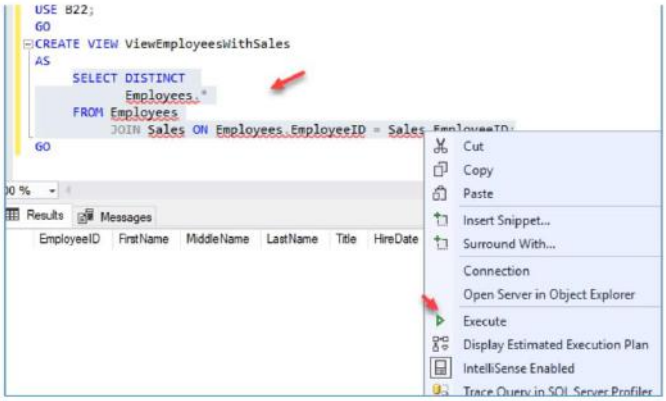
شکل ۸۳-۳ ایجاد View

برای ایجاد View از دو روش می‌توانیم استفاده کنیم؛ می‌توانیم از دستورات T-SQL استفاده کنیم، یا اینکه از Managemennt Studio استفاده کنیم.

برای ایجاد نمای جدید می‌توانیم از دستورات T-SQL زیر استفاده کنیم:

```
USE B22;
GO
CREATE VIEW ViewEmployeesWithSales
AS
     SELECT DISTINCT
            Employees.*
     FROM Employees
          JOIN Sales ON Employees.EmployeeID = Sales.EmployeeID;
GO
```

این یک نمای (view) ساده با یک عبارت ساده SELECT است که لیستی از کارمندان فروش را در جدول Employess و Sales برمی‌گرداند؛ در حقیقت، شما همیشه می‌توانید قبل از ایجاد نمایش، تنها با اجرای قسمت SELECT، عبارت CREATE VIEW SQL، عبارت جستجو را آزمایش کنید و بهتر است ببینید آیا پرسشی چیزی را برمی‌گرداند؛ برای اینکه یک قسمت از دستور در خروجی اجرا کنید باید به‌مانند شکل ۸۴-۳ دستور مورد نظر را انتخاب کنید و بعد، کلیک راست و گزینه‌ی Execute را انتخاب کنید که نتیجه‌ی آن مشخص خواهد شد؛ از آنجایی که جداول مقدار ندارند، یک جدول خالی را در شکل ۸۴-۳ مشاهده می‌کنید.



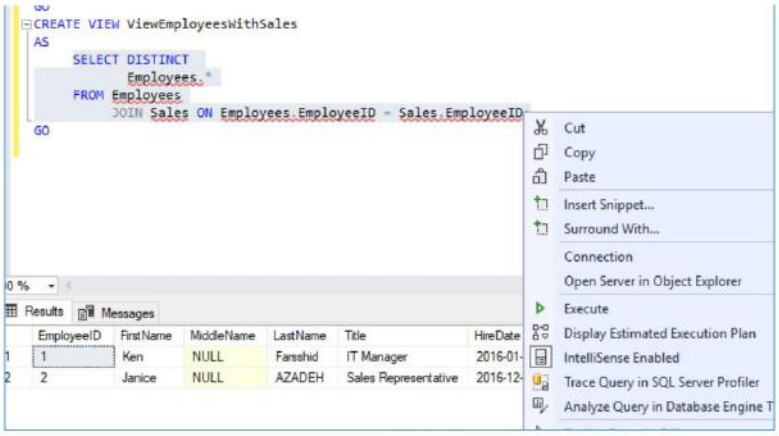
شکل ۸۴-۳ خروجی دستور

برای اینکه مقداری را در جداول وارد کنیم، می‌توانیم از دستورات زیر استفاده کنیم:

```
USE B22;
GO

INSERT INTO Employees VALUES (1,'Ken', NULL, 'Farshid', 'IT Manager', '1/1/2016',
2080, 45000);
INSERT INTO Employees VALUES (2,'Janice', NULL, 'AZADEH', 'Sales Representative',
'12/11/2016', 2080, 45000);

INSERT INTO Products VALUES (1,'Long-Sleeve Logo Jersey, S', 12.99);
INSERT INTO Products VALUES (2, 'Long-Sleeve Logo Jersey, M', 14.99);
INSERT INTO Products VALUES (3, 'Long-Sleeve Logo Jersey, L', 16.99);
INSERT INTO Products VALUES (4, 'Long-Sleeve Logo Jersey, XL', 18.99);

INSERT INTO Sales VALUES (NEWID(), 1, 1, 4, '04/15/2016');
INSERT INTO Sales VALUES (NEWID(), 2, 1, 1, '02/01/2016');
INSERT INTO Sales VALUES (NEWID(), 3, 1, 2, '03/12/2016');
INSERT INTO Sales VALUES (NEWID(), 2, 2, 2, '03/18/2016');
INSERT INTO Sales VALUES (NEWID(), 3, 2, 1, '04/16/2016');
INSERT INTO Sales VALUES (NEWID(), 4, 2, 2, '04/23/2016');
```

در خط اول دستور باید نام دیتابیسی که جداول در آن قرار دارند را وارد کنید، یعنی به‌جای B22، نام دیتابیس خود را بنویسید.

در ادامه برای تست موضوع باید دوباره قسمت SELECT را به‌مانند شکل ۸۵-۳ انتخاب کنید و دستور Execute را اجرا کنید؛ توجه داشته باشید در دستورات از DISTINCT برای جلوگیری از نمایش سوابق تکراری در خروجی استفاده کردیم.

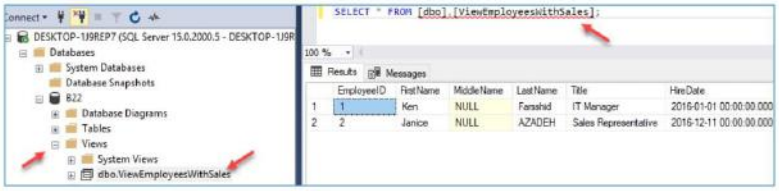


شکل ۳-۱۰۳ خروجی دستور

بعد از اینکه View را ایجاد کردید، می‌توانید به‌مانند شکل ۳-۱۰٤ از سمت چپ وارد قسمت Views شوید و نمای مورد نظر که ایجاد کردید را مشاهده کنید؛ با دستور زیر می‌توانید خروجی این View را مشاهده کنید:

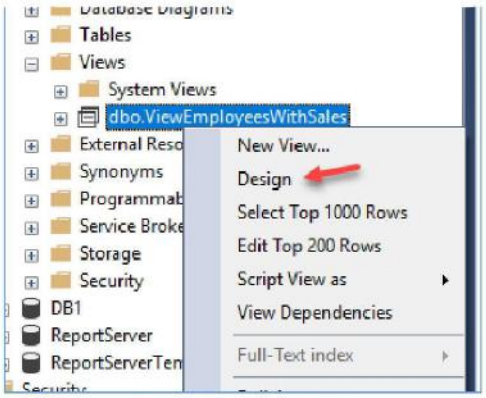
```
SELECT * FROM [dbo].[ViewEmployeesWithSales];
```

توجه داشته باشید به جای اینکه در دستور بالا، نام View را بنویسید، می‌توانید View مورد نظر را بکشید و در دستور رها کنید تا خودش نوشته شود.



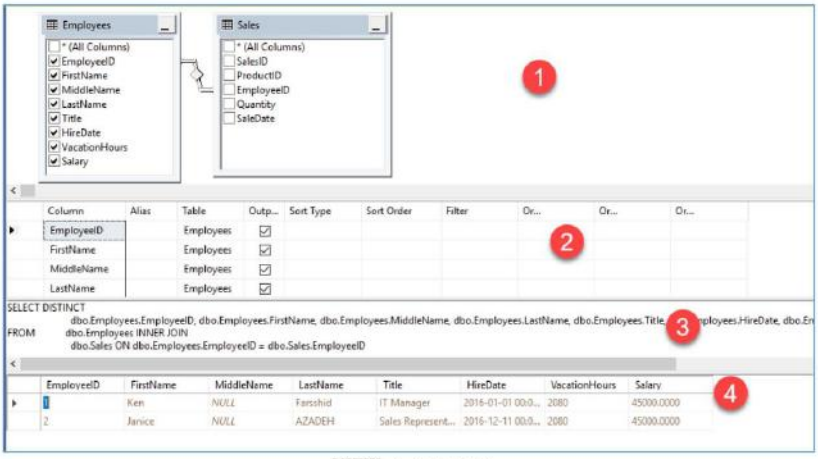
شکل ۳-۱۰۴ خروجی *View*

در ادامه برای اینکه ارتباط بین جداول را در View مورد نظر به‌صورت گرافیکی مشاهده کنید باید به‌مانند شکل ۳-۱۰۵ بر روی View مورد نظر کلیک راست کنید و گزینه‌ی Design را انتخاب کنید؛ توجه داشته باشید که شما می‌توانستید به‌جای وارد کردن دستورات از همین قسمت با انتخاب New View، نمای مورد نظر خود را انتخاب کنید.



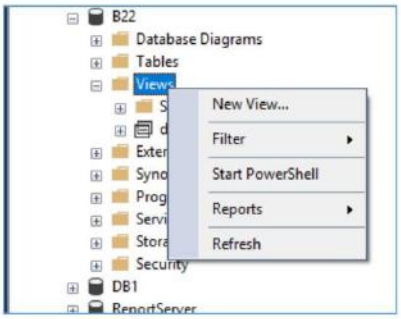
شکل ۱۰۵-۳ بررسی دستور *Design*

همان طور که در شکل ۱۰۶-۳ مشاهده می‌کنید، این صفحه از چهار قسمت تشکیل شده است؛ در قسمت شماره‌ی نمای کلی، دو جدول را مشاهده می‌کنید؛ در قسمت شماره‌ی دو طبق ستون‌هایی که در دستور قبلی وارد کردیم، ستون‌های مورد نظر تیک خورده شدند؛ در قسمت سوم دستور خروجی SELECT را که قبلاً وارد کردیم را مشاهده می‌کنید؛ کلاً در قسمت شماره‌ی یک، اگر هر یک از ستون‌ها را انتخاب کنید در قسمت شماره‌ی سه دستوارت تغییر خواهد کرد؛ در قسمت شماره‌ی چهار نیز خروجی دستور شماره‌ی سوم را مشاهده می‌کنید، البته باید دستور شماره‌ی سوم را انتخاب و کلیک راست کنید و گزینه‌ی Execute را انتخاب کنید؛ توجه داشته باشید در قسمت شماره‌ی یک، پیوند بین جدول اول و دوم ایجاد شده است که آن نیز به دلیل دستور dbo.Employees.EmployeeID = dbo.Sales.EmployeeID که در قسمت شماره‌ی سوم وارد کردید، است؛ به جای = اگر <> قرار دهید شکل آن در قسمت شماره‌ی یک تغییر خواهد کرد.



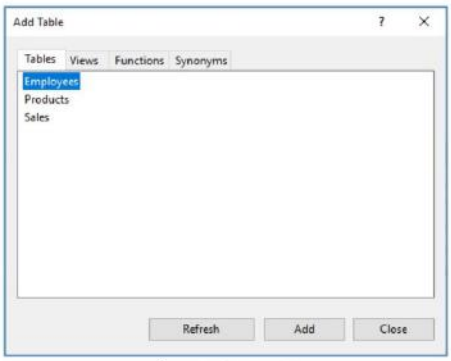
شکل ۱۰۶-۳ ایجاد *VIEW*

اگر بخواهید از طریق محیط گرافیکی، View ایجاد کنید باید به‌مانند شکل ۱۰۷-۳ بر روی View کلیک راست کنید و گزینه‌ی New View را انتخاب کنید.



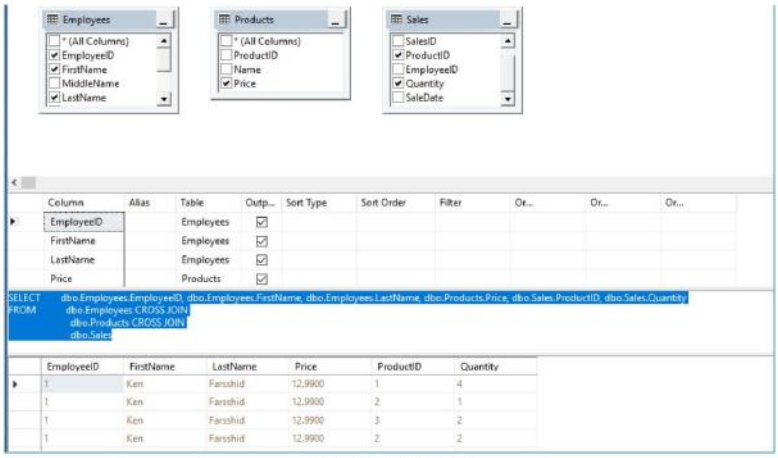
شکل ۱۰۷ -۳ ایجاد View

بعد از ورود، شکل ۱۰۸- ۳ ظاهر خواهد شد که باید انتخاب کنید که از چه جداولی می‌خواهید خروجی بگیرید؛ در این قسمت، هر سه جدول را انتخاب می‌کنیم، البته می‌توانیم از تب View نیز استفاده کنیم.



شکل ۱۰۸ - ۳ ایجاد view

در شکل ۱۰۹ – ۳ و در قسمت اول می‌توانید ستون‌هایی که قرار است در خروجی چاپ شود را انتخاب کنید؛ به این نکته توجه کنید که انتخاب باید به ترتیب باشد، چون هر انتخابی که در جداول انجام می‌دهید، در قسمت دوم ردیف آن مشخص می‌شود و در قسمت سوم نیز دستورات آن به‌صورت اتوماتیک نوشته خواهد شد؛ کلاً View همان دستور SELECT است که قبلاً با هم روی آن کار کردیم.

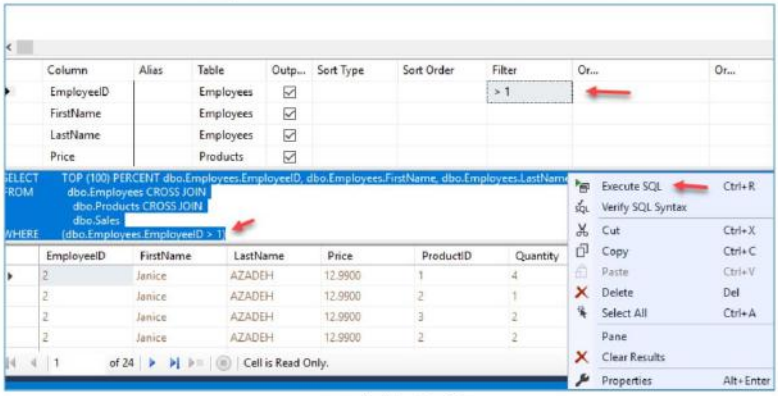


شکل ۱۰۹ – ۳ ایجاد View

برای اینکه بتوانیم شرط در این دستورات قرار دهیم، می‌توانیم به‌مانند شکل ۱۱۰ – ۳ در قسمت دوم و در ستون فیلتر، شرط خود را وارد کنیم که در اینجا 2< وارد شده است، یعنی اینکه در خروجی کار سطرهایی که EmployeeID آنها بزرگ‌تر از 2 باشد در خروجی چاپ خواهد شد؛ توجه کنید زمانی که در قسمت فیلتر، شرط مورد نظر خود را وارد می‌کنید، بعد از خروج از آن، قسمت دستور آن در قسمت شماره‌ی سوم نوشته می‌شود که در شکل ۱۱۰ –۳ مشخص شده است.

برای اینکه دستوری که ایجاد شده را اجرا کنیم باید آن را اجرا کنیم و کلیک راست کنیم و گزینه‌ی Execute SQL را انتخاب کنیم تا در قسمت چهارم، خروجی مشخص شود که همان طور که گفتیم خروجی برای EmployeeIDهایی که بزرگ‌تر از دو باشد، مشخص شده است.

در ستون‌های Sort نیز می‌توانید مشخص کنید که خروجی به چه صورت تنظیم شود.



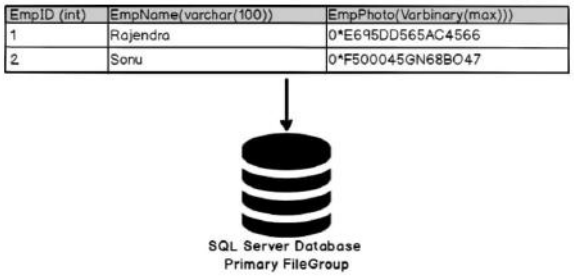
شکل ۱۱۰ - ۳ اجرای دستور

۳-۱-۷ بررسی FileStream در SQL Server

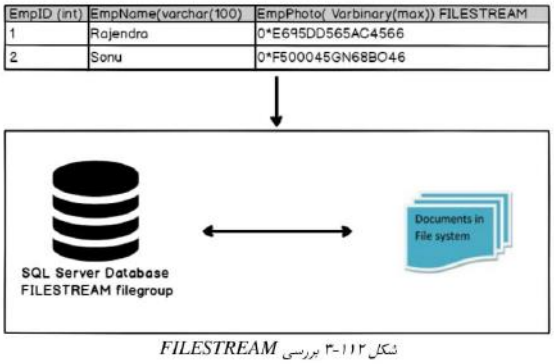
در نرم‌افزار SQL قابلیتی طراحی شده که بتوانید فایل‌هایی با حجم بالا را درآن ذخیره کنید. همیشه این طور نیست که فقط داده‌های متنی در دیتابیس ذخیره شوند، بلکه داده‌هایی مانند: عکس، موسیقی، فیلم نیز می‌توانند در SQL ذخیره شوند، اما برای این کار باید از قابلیت FileStream در SQL استفاده کنید؛ در نسخه‌های قدیمی SQL افراد برای ذخیره‌ی عکس، موسیقی و فیلم از داده‌هایی از نوع BLOB استفاده می‌کردند که این داده‌ها تا حداکثر ۲ گیگابایت را پشتیبانی می‌کرد و بیشتر از این عدد، دیگر نمی‌توانستند ذخیره کنند و یکی از مهم‌ترین مشکلات این نوع داده‌ها این بود که عملکرد سیستم را به شدت کاهش می‌داد و باعث کندی آن می‌شد، اما با استفاده از FileStream دیگر نگران مقدار فضا و کندی سرعت نرم‌افزار نخواهیم بود.

برای اینکه از FILESTREAM استفاده کنیم باید یک گروه برای آن ایجاد کنیم تا اطلاعات در گروه جدید قرار بگیرد و دیگر با گروه اصلی یا همان Primary کاری نداریم؛ برای استفاده از این قابلیت در جداول نیاز به یک رکورد با نوع varbinary داریم.

در شکل ۱۱۱-۳ نوع داده‌های قدیمی برای ذخیره عکس را مشاهده می‌کنید که از یک دیتابیس استفاده می‌کردند، اما در عکس ۱۱۲-۳ این موضوع از دیتابیس اصلی جدا شده است.

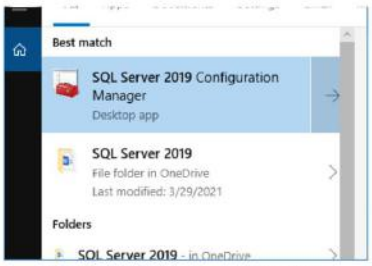


شکل ۱۱۱-۳ بررسی *FILESTREAM*



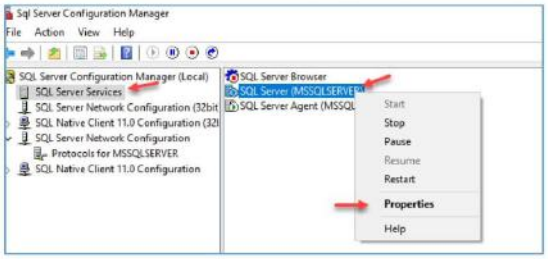
شکل ۱۱۲-۳ بررسی *FILESTREAM*

برای فعال‌سازی قابلیت FILESTREAM باید به‌مانند شکل ۱۱۳ – ۳ سرویس SQL Server 2019 Configuration Manager را اجرا کنید.



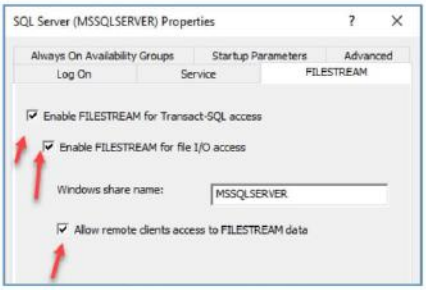
شکل ۳-۱۱۳ اجرای سرویس *Configuration*

در شکل ۳-۱۱٤ از سمت چپ بر روی SQL Server Services کلیک کنید و در صفحه‌ی بازشده بر روی SQL Server کلیک راست کنید و گزینه‌ی Properties را انتخاب کنید.



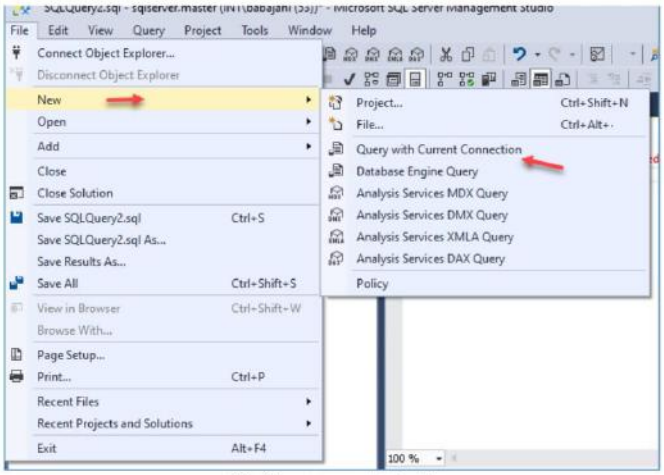
شکل ۳-۱۱۴ بررسی سرویس

در شکل ۳-۱۱۵ وارد تب FILESTREAM شوید و تیک هر سه گزینه را انتخاب کنید؛ گزینه‌ی اول برای دسترسی از طریق دستورات T-SQL است؛ گزینه‌ی دوم برای خواندن و نوشتن از ورودی و خروجی پرونده‌ها است و برای این کار باید یک نام برای به اشتراک‌گذاری آن وارد کنید و گزینه‌ی آخر، اجازه‌ی دسترسی از راه دور به FILESTREAM را می‌دهد.



شکل ۳-۱۱۵ بررسی *FileStream*

برای اینکه FileStream را در SQL فعال کنیم باید تغییری در تنظیمات آن انجام دهیم؛ برای این کار وارد منوی فایل به‌مانند شکل ۳-۱۱٦ شوید و از قسمت New بر روی Query کلیک کنید.

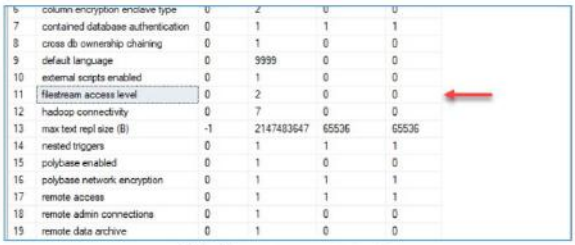


شکل ۱۱۶-۳ بررسی *FileStream*

دستور زیر را اجرا کنید:

```
USE master
  Go
  EXEC sp_configure
  Go
```

این دستور تنظیمات کلی سرور SQL را به شما نشان می‌دهد که مربوط به قسمت Advanced است؛ در این قسمت که در شکل ۱۱۷-۳ مشخص شده است باید به قسمت File Stream Access Level مراجعه کنید و در قسمت ستون config_value عدد ۲ قرار دهید؛ با این کار دسترسی کامل به سرویس FILESTREAM فعال خواهد شد.



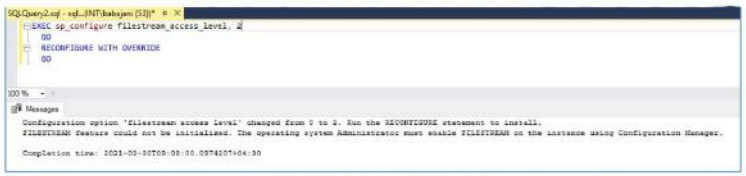
7	contained database authentication	0	1	1	1
8	cross db ownership chaining	0	1	0	0
9	default language	0	9999	0	0
10	external scripts enabled	0	1	0	0
11	filestream access level	0	2	0	0
12	hadoop connectivity	0	7	0	0
13	max text repl size (B)	-1	2147483647	65536	65536
14	nested triggers	0	1	1	1
15	polybase enabled	0	1	0	0
16	polybase network encryption	0	1	1	1
17	remote access	0	1	1	1
18	remote admin connections	0	1	0	0
19	remote data archive	0	1	0	0

شکل ۱۱۷-۳ بررسی *FileStream*

برای قراردادن عدد ۲ باید دستور زیر را وارد کنید.

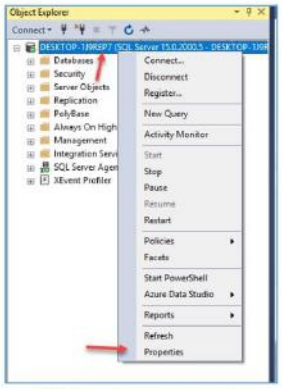
```
EXEC sp_configure filestream_access_level, 2
  GO
  RECONFIGURE WITH OVERRIDE
  GO
```

در دستور بالا، قسمت config_value مربوط به filestream_access_level به عدد ۲ تغییر می‌کند که در شکل ۱۱۸-۳ این موضوع را مشاهده می‌کنید.



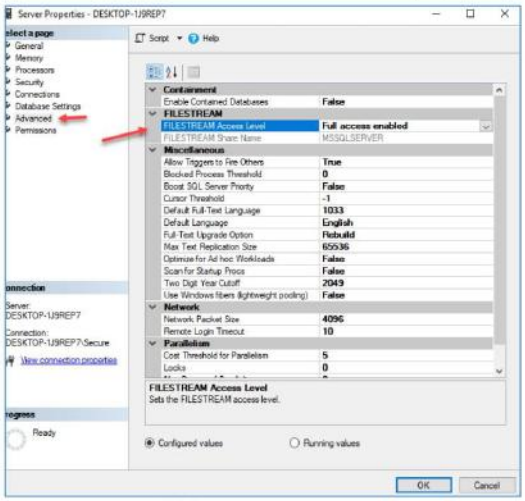
شکل ۱۱۸-۳ بررسی *FileStream*

برای اینکه متوجه شویم که کار به درستی انجام شده است یا نه باید به‌مانند شکل ۱۱۹-۳ بر روی سرور SQL کلیک راست کنید و وارد قسمت Properties شوید.



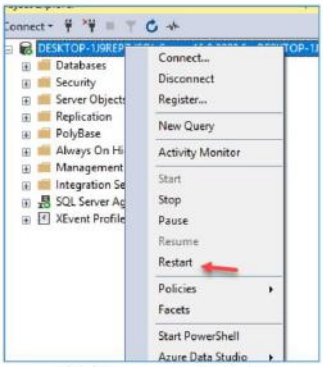
شکل ۱۱۹ -۳ بررسی *FileStream*

در شکل ۱۲۰-۳ از سمت چپ وارد قسمت Advanced شوید و قسمت FILESTREAM Access Level را مشاهده کنید که بر روی FULL access enabled قرار گرفته است، البته شما می‌توانستید دستورات قبلی را وارد نکنید و مستقیم در این قسمت، این گزینه را انتخاب کنید.



شکل ۱۲۰-۳ بررسی *FileStream*

بعد از اتمام کار، حتماً باید سرور را Restart کنید؛ برای این کار باید به‌مانند شکل ۱۲۱-۳ بر روی سرور SQL کلیک راست کنید و گزینه‌ی Restart را انتخاب کنید تا تنظیمات به درستی اعمال شود.



شکل ۱۲۱-۳ بررسی *FileStream*

نکاتی که قبل از استفاده از FILEDTREAM باید بدانیم شامل:

۱- می‌توانیم از عبارات SELECT، INSERT، UPDATE و DELETE مشابه‌ی پرس‌وجو استاندارد پایگاه‌داده در FILESTREAM استفاده کنیم.

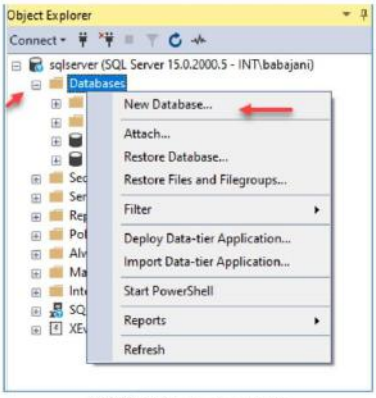
۲- اگر اندازه‌ی شی یا Object بیشتر از ۱ مگابایت باشد باید از FILESTREAM استفاده کنیم.

۳- هر سطر باید یک ID ردیف منحصربه‌فرد داشته باشد تا از این قابلیت استفاده کند و نباید حاوی مقادیر NULLباشد.

۴- نمی‌توانیم داده‌های FILESTREAM را رمزگذاری کنیم.

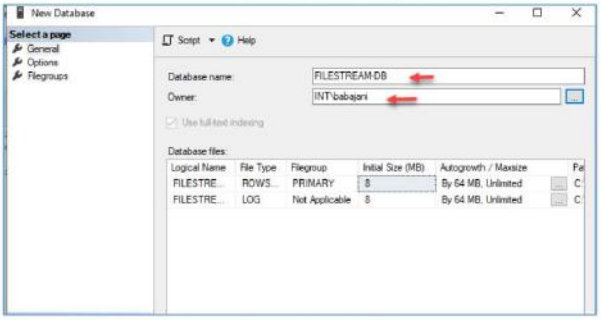
۱-۷-۱-۳ ایجاد دیتابیس برای استفاده از FILESTREAM

برای اینکه بتوانیم از قابلیت FILESTREAM استفاده کنیم باید یک دیتابیس جدید ایجاد و بر روی آن کار کنیم، البته می‌توانیم از دیتابیس‌های قدیمی خود استفاده کنیم؛ به‌مانند شکل ۱۲۲-۳ بر روی Databases کلیک راست کنید و گزینه‌ی New Databases را انتخاب کنید.



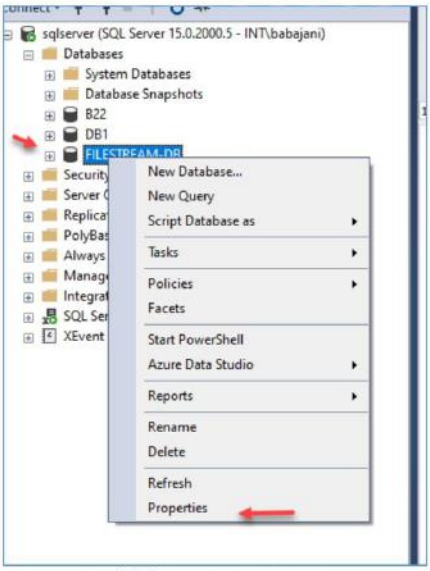
شکل ۱۲۲-۳ ایجاد *FileStream*

به‌مانند شکل ۱۲۳-۳، یک نام برای دیتابیس خود وارد کنید که در اینجا، FILESTREAM-DB وارد شده است، بعد از این کار بر روی ADD کیک کنید تا دیتابیس مورد نظر ایجاد شود.



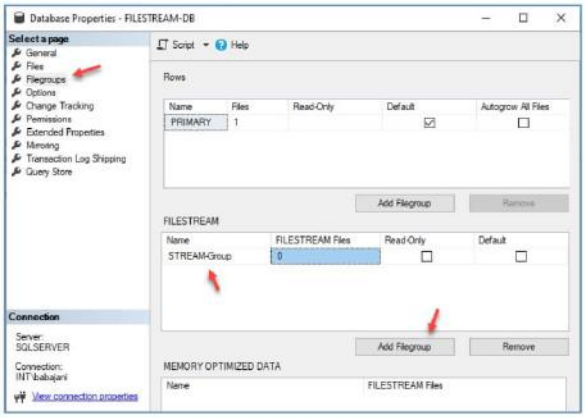
شکل ۱۲۳-۳ ایجاد *FileStream*

بعد از ایجاد دیتابیس، به‌مانند شکل ۱۲٤-۳ بر روی آن کلیک راست کنید و گزینه‌ی Properties را انتخاب کنید.



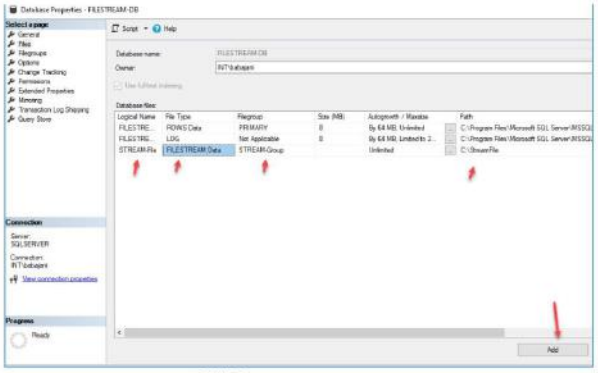
شکل ۱۲۴-۳ ایجاد *FileStream*

در شکل ۱۲۵-۳ باید وارد FileGroup شوید و در قسمت FILESTREAM بر روی Add کلیک کنید و نام گروه مورد نظر خود را برای تخصیص به FILESTREAM وارد کنید.



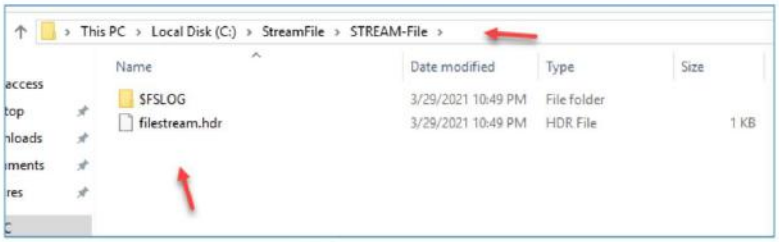
شکل ۱۲۶-۳ ایجاد *FileStream*

در ادامه‌ی کار به‌مانند شکل ۱۲۷-۳ وارد قسمت Files شوید و بر روی Add کلیک کنید؛ در ستونLogical Name یک نام به‌دلخواه وارد کنید؛ در ستون File Type نوع فایل را FILESTREM Data در نظر بگیرید و Filegroup را نیز انتخاب کنید، البته به‌صورت اتوماتیک خودش انتخاب خواهد شد؛ در قسمت Path نیز باید مسیری را انتخاب کنید که از ظرفیت خوبی برخوردار باشد؛ در آخر بر روی OK کلیک کنید تا تنظیمات ذخیره شود.



شکل ۱۲۷-۳ ایجاد *FileStream*

بعد از انجام تنظیمات، اگر به‌مانند شکل ۱۲۸-۳ وارد آدرسی مربوط به Stream شوید، مشاهده خواهید کرد که فایل STREM در این مسیر ایجاد شده است.



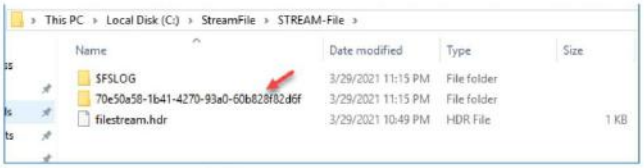
شکل ۱۲۸-۳ ایجاد *FileStream*

۲-۷-۱-۳ ایجاد جدول در دیتابیس FILESTREAM

برای اینکه یک جدول با داده‌های FILESTREAM داشته باشیم باید به‌صورت زیر عمل کنیم؛ همان‌طور که گفتیم برای استفاده از FILESTREAM باید داده از نوع VARBINARY باشد که در دستور زیر مشخص شده است و یکی دیگر از شرط‌های FILESTREAM این است که حتماً باید یک کلید منحصربه‌فرد داشته باشد و حتماً نیز باید NOT NULL باشد؛ در دستور زیر سه فیلد ایجاد شده است که در Fileid نوع داده، UNIQUIDENTIFIER و NOT NULL است؛ داده‌ی FileName نیز از نوع معمولی VARCHAR با حداکثر کاراکتر ۲۵ است و FILE نیز از نوع VARBINARY است.

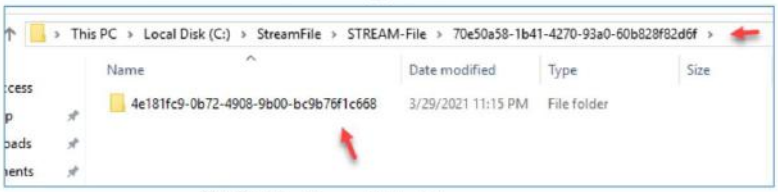
```
Use [FILESTREAM-DB]
Go
CREATE TABLE [FileStreamTable_1] (
[FileId] UNIQUEIDENTIFIER ROWGUIDCOL NOT NULL UNIQUE,
[FileName] VARCHAR (25),
[File] VARBINARY (MAX) FILESTREAM);
GO
```

بعد از اجرای دستور در مسیر مشخص‌شده که در شکل ۱۲۹-۳ مشخص شده است، یک پوشه با مقدار GUID ایجاد شده است.



شکل ۳-۱۲۹ بررسی فایل *FileStream*

اگر وارد این پوشه در شکل ۳-۱۳۰ شوید، یک پوشه‌ی دیگر با GUID جدید مشاهده می‌کنید که مربوط به ستون FILESTREAM در جدول جدید است؛ این موضوع را در شکل ٦٢ مشاهده می‌کنید.

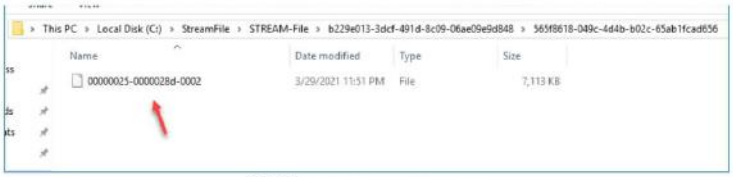


شکل ۳-۱۳۰ بررسی فایل *FileStream*

برای اینکه بیشتر با FILESTREAM کار کنیم، می‌خواهیم یک فایل موسیقی را درون دیتابیس مورد نظر ذخیره کنیم؛ اولین کاری که باید انجام دهید این است که فایل موسیقی را در مسیر C:\Music ذخیره کنیم،
بعد از آن باید از دستور زیر استفاده کنید؛ در این دستور، یک جدول جدید با نام FileStreamTable_2 ایجاد می‌شود و در ادامه با دستور DECLARE، متغیر FILE با نوع varbinary صدا زنده می‌شود، فایل MP3 در آن قرار می‌گیرد.

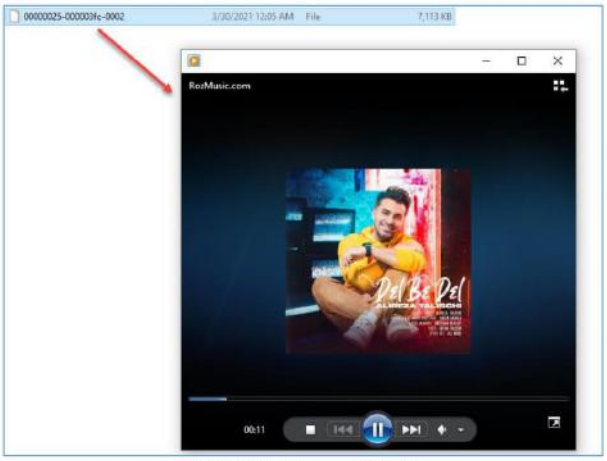
```
Use [FILESTREAM-DB]
Go
CREATE TABLE [FileStreamTable_2] (
[FileId] UNIQUEIDENTIFIER ROWGUIDCOL NOT NULL UNIQUE,
[FileName] VARCHAR (25),
[File] VARBINARY (MAX) FILESTREAM);
GO
DECLARE @File varbinary(MAX);
SELECT
@File = CAST(
bulkcolumn as varbinary(max)
)
FROM
OPENROWSET(BULK 'c:\music\a.mp3', SINGLE_BLOB) as MyData;

INSERT INTO FileStreamTable_2
VALUES
(
NEWID(),
'Sample Music',
@File
)
```

اگر به شکل ۳-۱۳۱ توجه کنید، یک فایل جدید به حجم تقریبی ۷ مگابایت که همان فایل MP3است در مسیر مورد نظر ایجاد شده است.



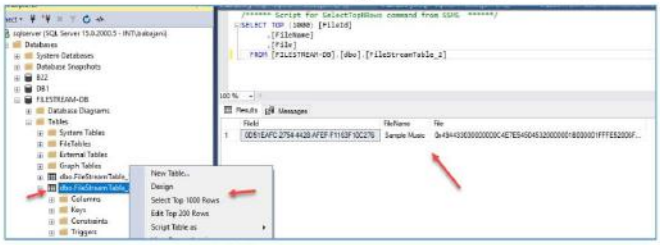
شکل ۱۳۱-۳ بررسی فایل *FileStream*

توجه داشته باشید، اگر فایل موسیقی در شکل ۱۳۱-۳ را در نرم‌افزار مورد نظر قرار دهید به راحتی اجرا خواهد شد؛ این موضوع را در شکل ۱۳۲-۳ مشاهده می‌کنید.



شکل ۱۳۲-۳ بررسی فایل *FileStream*

اگر وارد SQL شوید و بر روی جدول مورد نظر کلیک راست کنید و گزینه‌ی Select Top 1000 Rows را انتخاب کنید، خروجی جدول مورد نظر را نشان می‌دهد که در شکل ۱۳۳-۳ مشخص شده است.



شکل ۱۳۳-۳ بررسی فایل *FileStream*

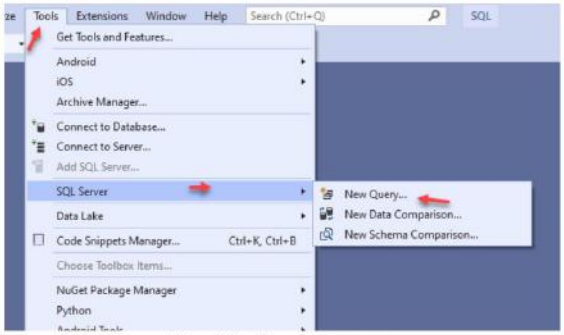
۸-۱-۳ ارتباط با SQL از طریق Visual Studio

یکی از ابزارهای مهم در صنعت برنامه‌نویسی و پایگاه‌داده، نرم‌افزار عالی Visual Studio است که ما را در ارائه‌ی راحت‌تر و بهتر کار یاری می‌کند؛ برای اینکه از آخرین نسخه‌ی این نرم‌افزار استفاده کنید، می‌توانید از لینک زیر آن را دانلود کنید:

https://dl2.soft98.ir/programing/Microsoft.Visual.Studio.2019.16.9.1.html

بعد از دانلود، آن را بر روی سیستم خود نصب کنید؛ توجه داشته باشید برای نصب کامل این نرم‌افزار حداقل نیاز به ۵۰ گیگابیت فضای هارد دارید که واقعاً فضای زیادی را اشغال می‌کند.

بعد از اجرای نرم‌افزار، اولین کاری که انجام می‌دهیم این است که از طریق Visual Studio به SQL Server متصل شویم و Query خود را اجرا کنیم؛ برای این کار به‌مانند شکل ۱۳۴-۳ وارد منوی Tools شوید و از قسمت SQL Server، گزینه‌ی New Query را انتخاب کنید.

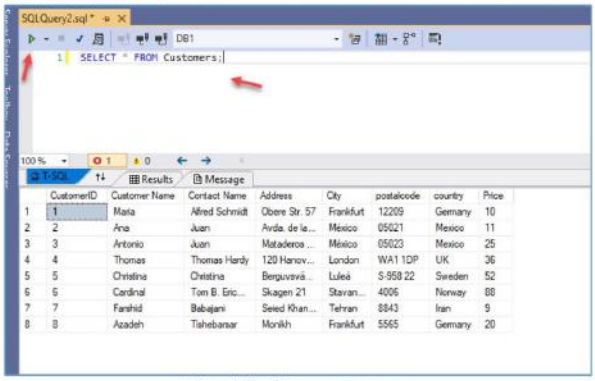


شکل ۱۳۴-۳ بررسی *Visual Studio*

در شکل ۱۳۵-۳ باید در قسمت Server name نام سرور SQL را وارد کنید و بعد از آن، مشخص کنید که نوع احراز هویت به چه صورت باشد، اگر با همان کاربری که در حال کار با visual studio هستید، می‌خواهید با SQL ارتباط برقرار کنید، می‌توانید Windows Authentication را انتخاب کنید و یا اگر می‌خواهید با کاربران موجود در SQL وارد آن شوید باید SQL Authentication را انتخاب کنید؛ در قسمت آخر نیز باید نام دیتابیس خود را انتخاب کنید و بر روی Connect کلیک کنید.

شکل ۱۳۵-۳ بررسی *Visual Studio*

همان طور که در شکل ۱۳۶-۳ مشاهده می‌کنید با استفاده از دستور SELECT توانستیم جدول Customers را در خروجی چاپ کنیم؛ این بدان معنا است که از طریق Visual Studio توانستیم با SQL Server ارتباط برقرار کنیم.



شکل ۱۳۷-۳ بررسی *Visual Studio*

۱-۸-۱-۳ ایجاد دیتابیس از طریق دستورات در Visual Studio

برای اینکه کار خود را گسترش دهیم، می‌خواهیم از طریق خط فرمان، یک دیتابیس ایجاد و در آن جدول دلخواه خود را ایجاد کنیم؛ برای این کار باید در query مورد نظر در Visual Studio دستورات زیر را وارد کنیم:

```
CREATE DATABASE B22
GO
USE B22
CREATE TABLE B22T (
   IDNumber int identity primary key,
   FirstName varchar(50),
   LastName varchar(50),
   City varchar(10),
   Country varchar(10),
   Address varchar(50)
);
INSERT INTO B22T(FirstName, LastName, City, Country, Address) VALUES
('ahmad', 'Mohamad', 'babol', 'IR', 'Kordmahaleh'),
('Azadeh', 'Mohebi', 'Shiraz', 'IR', 'Shiraz1'),
('Negar', 'Sistani', 'Karaj', 'IR', 'MohamadShahr'),
('alireza', 'nafeei', 'khozestan', 'IR', 'khozestan'),
('Elham', 'bozorgi', 'ardebil', 'IR', 'Azarbaijan')
```

در دستورات بالا، در خط اول با دستور CREATE DATABASE، یک دیتابیس با نام B22 ایجاد کردیم و در ادامه، حتماً دستور GO را قرار دهید تا بعد از ایجاد دیتابیس B22، یک refresh صورت بگیرد؛ در ادامه و در خط سوم باید مشخص کنیم که از چه دیتابیسی می‌خواهیم استفاده کنیم تا جدول را داخل آن ایجاد کنیم که این کار را با دستور USE B22 انجام می‌دهیم.

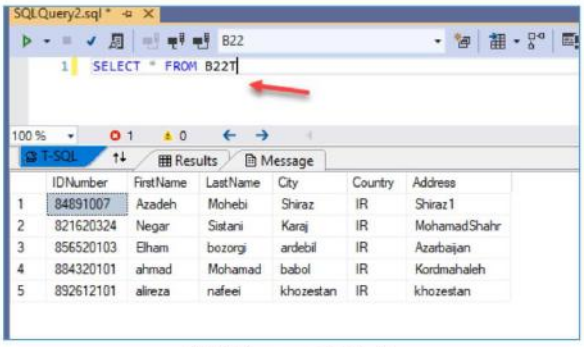
در خط چهارم با استفاده از دستور CREATE TABLE، یک جدول با نام B22T در دیتابیس B22 ایجاد می‌کنیم و اطلاعات آن را نیز در ادامه مشخص می‌کنیم؛ توجه داشته باشید که IDNumber از نوع Primery Key است و نباید خالی باشد.

بعد از ایجاد جدول B22T باید با دستور INSERT INTO، اطلاعات مورد نظر را وارد ستون‌های جدول کنیم؛ به این نکته‌ی مهم توجه کنید که IDNumber در دستور INSERT INTO وجود ندارد؛ این موضوع به این دلیل است که در خط پنجم از دستور identity استفاده کردیم و باعث می‌شود که به‌صورت اتوماتیک در هر سطر، یک شماره به آن سطر اختصاص دهد و اگر بخواهید از IDNumber در داخل جدول استفاده کنید و شماره‌ی دلخواه خود را وارد کنید باید کد بالا را به‌صورت زیر تغییر دهید:

```
CREATE DATABASE B22
GO
USE B22
CREATE TABLE B22T (
   IDNumber int  primary key,
   FirstName varchar(50),
   LastName varchar(50),
   City varchar(10),
   Country varchar(10),
   Address varchar(50)
);
INSERT INTO B22T(IDNumber,FirstName, LastName, City, Country, Address) VALUES
('884320101','ahmad', 'Mohamad', 'babol', 'IR', 'Kordmahaleh'),
('84891007','Azadeh', 'Mohebi', 'Shiraz', 'IR', 'Shiraz1'),
('821620324','Negar', 'Sistani', 'Karaj', 'IR', 'MohamadShahr'),
('892612101','alireza', 'nafeei', 'khozestan', 'IR', 'khozestan'),
('856520103','Elham', 'bozorgi', 'ardebil', 'IR', 'Azarbaijan')
```

در کد بالا و در خط پنجم، دستور identity حذف شده است و حتماً باید در ورودی به IDNumber یک مقدار بدهید تا با خطا مواجه نشوید،؛دلیل آن نیز این است که این گزینه از نوع Primary Key است و نباید خالی باشد؛ به این دلیل در خط دوازده و در داخل پرانتز، IDNumber تعریف و مقدار آن نیز داده شده است.

همان طور که در شکل ۱۳۸-۳ مشاهده می‌کنید، دستورات به درستی اجرا شده و خروجی جدول، B22T را چاپ کرده است؛ توجه داشته باشید که این خروجی مربوط به کد دومی است که داخل آن از IDNumber استفاده شده است.

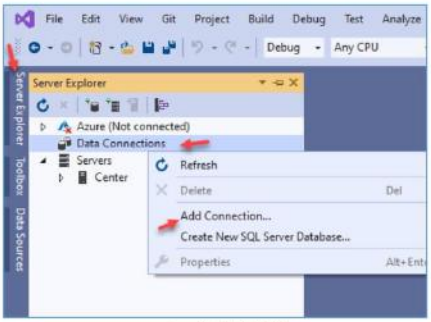


شکل ۱۳۸-۳ خروجی جدول B22T

۲-۸-۱-۳ ایجاد دیتابیس از طریق ابزار Visual Studio

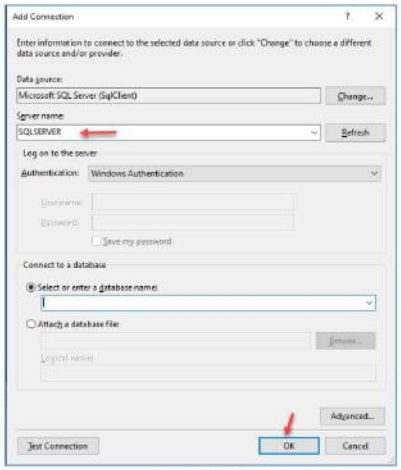
در قسمت قبلی توانستیم با استفاده از Query، یک دیتابیس جدید ایجاد و جدول آن را به همراه مقادیر آن ایجاد کنیم؛ در این قسمت نیز می‌خواهیم از طریق ابزارهای موجود در Visual Studio این کار را انجام دهیم.

برای شروع به‌مانند شکل ۱۳۹-۳ وارد Server Explorer شوید و بر روی Data Connections کلیک راست کنید و بر روی Add Connection کلیک کنید.



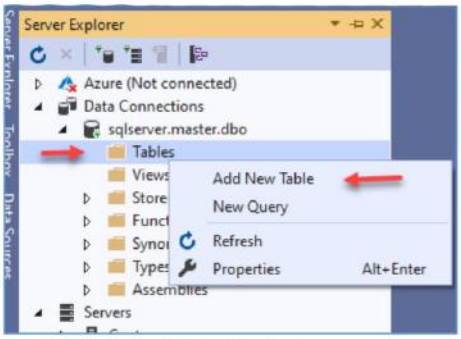
شکل ۱۳۹-۳ / ایجاد دیتابیس

بعد از باز شدن پنجره در شکل ۱۴۰-۳، نام سرور SQL را وارد و بر روی Connect کلیک کنید.



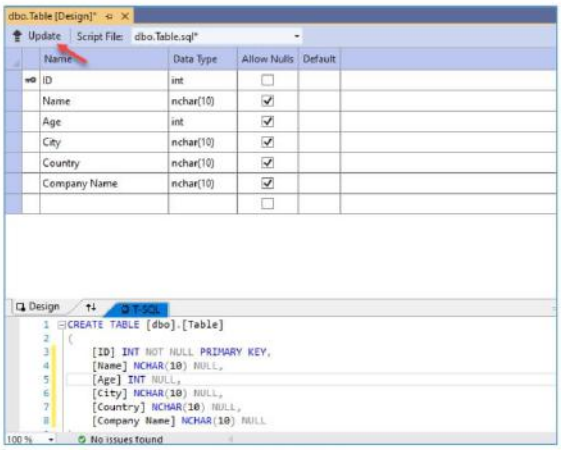
شکل ۱۴۰-۳ / ایجاد دیتابیس

همان طور که در شکل ۱۴۱-۳ مشاهده می‌کنید به SQL مورد نظر متصل شدیم و برای اینکه بتوانیم جدول مورد نظر خود را ایجاد کنیم، می‌توانیم بر روی Tables کلیک راست کنیم و گزینه‌ی Add New Table را انتخاب کنیم.



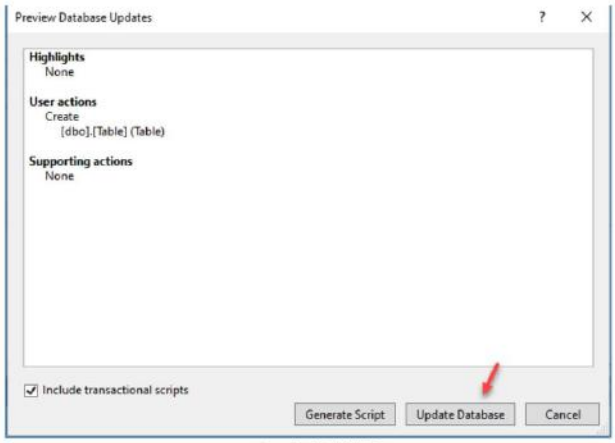
شکل ۱۴۱-۳ / ایجاد جدول

همان طور که در شکل ۱۴۲-۳ مشاهده می‌کنید به راحتی می‌توانید اطلاعات مورد نظر جدول خود را وارد کنید و هم‌زمان که این کار را انجام می‌دهید در زیر شکل ۱۴۲-۳، Query آن نیز نوشته می‌شود؛ برای تایید اطلاعات باید بر روی Update کلیک کنید.



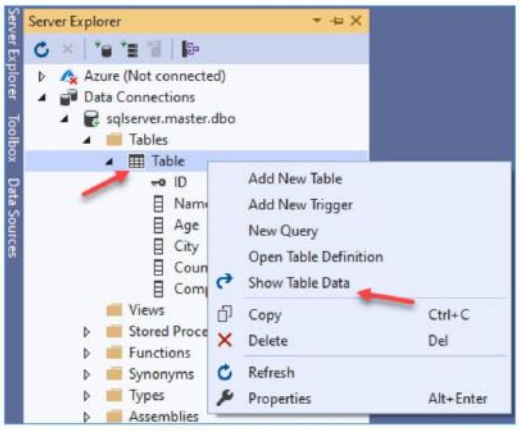
شکل ۱۴۲-۳ / ایجاد جدول

بعد از کلیک بر روی Update، شکل ۱۴۳-۳ ظاهر می‌شود که باید بر روی Update Database کلیک کنید تا اطلاعات به سرور SQL ارسال شود و در آخر پیغام تأیید را برای شما ارسال می‌کند.



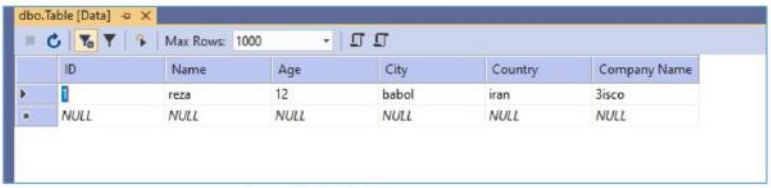
شکل ۱۴۳-۳ / ایجاد دیتابیس

همان طور که در شکل ۱٤٤-۳ مشاهده می‌کنید، جدول مورد نظر ایجاد شده است و برای اینکه اطلاعات خود را وارد کنید باید بر روی جدول کلیک راست کنید و گزینه‌ی Show Table Data را انتخاب کنید.



شکل ۱۴۴-۳ / ایجاد جدول و نمایش

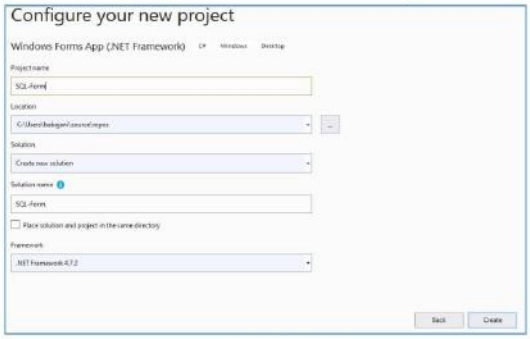
همان‌طور که در شکل ۱٤٥-۳ مشاهده می‌کنید، اطلاعات را می‌توانید وارد ستون‌های جدول کنید.



شکل ۱۴۵-۳ نمایش جدول

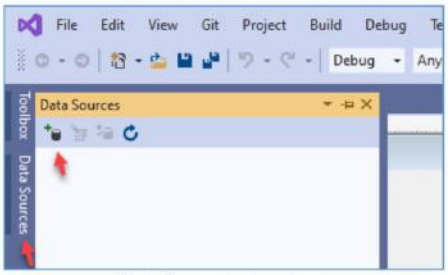
۳-۸-۱-۳ ایجاد فرم در Visual Studio و ثبت اطلاعات در جدول SQL

در این قسمت می‌خواهیم یک فرم طراحی کنیم تا کاربر بعد از وارد کردن اطلاعات، آن را در SQL ثبت کند؛ برای اینکه یک فرم ایجاد کنیم و اطلاعات ورودی کاربر را در SQL ثبت کنیم باید به‌مانند شکل ۱٤٦-۳ یک پروژه‌ی جدید از نوع Windows Forms App ایجاد کنیم.



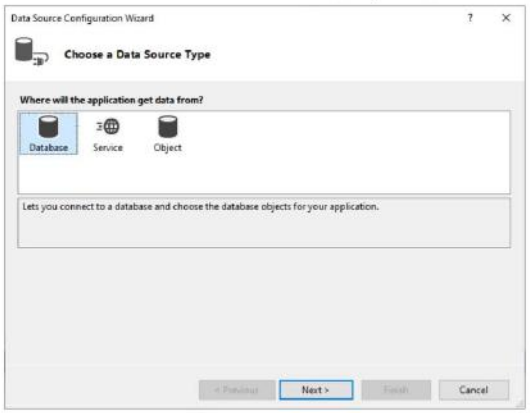
شکل ۱۴۶-۳ ایجاد فایل جدید

در شکل ۱٤۷-۳ وارد Data Sources شوید و بر روی آیکون مورد نظر کلیک کنید.



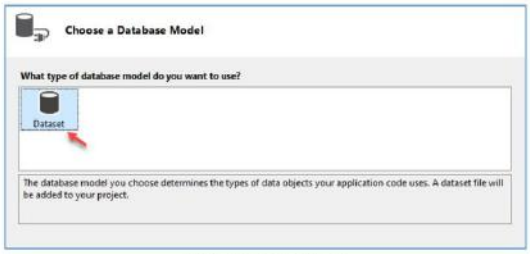
شکل ۱۴۷-۳ ایجاد *Data Source*

در شکل ۱٤۸-۳ گزینه‌ی Database را انتخاب کنید.



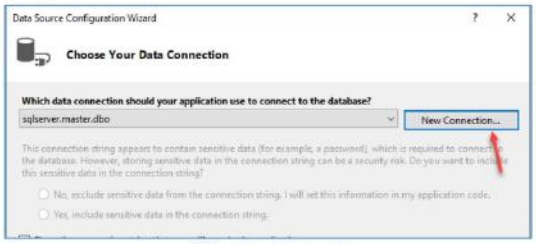
شکل ۱۴۸-۳ ارتباط با *SQL*

در شکل ۱۴۹-۳ گزینه‌ی Dataset را انتخاب و بر روی Next کلیک کنید.



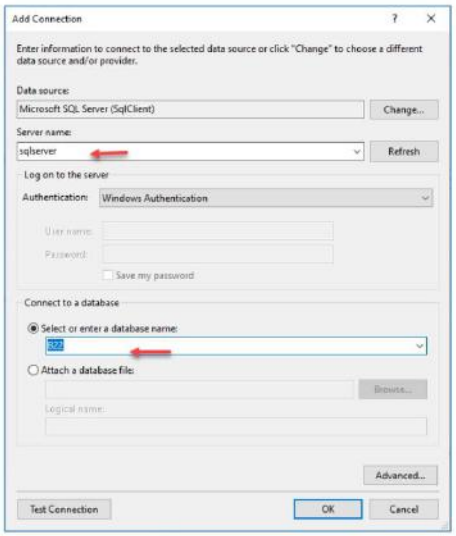
شکل ۱۴۹-۳ ارتباط با SQL

در شکل ۱۵۰-۳ بر روی New Connection کلیک کنید.



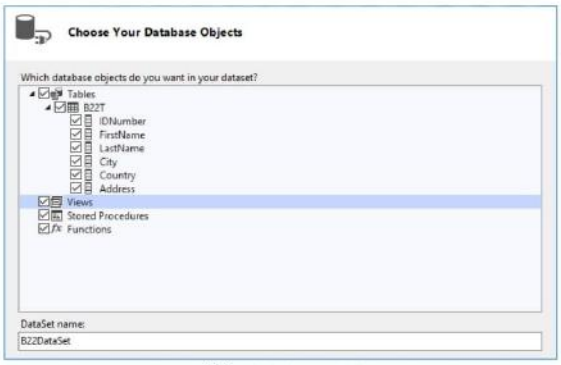
شکل ۱۵۰-۳ ارتباط با SQL

در شکل ۱۵۱-۳ باید نام سرور را وارد و دیتابیس مورد نظر را انتخاب کنید.



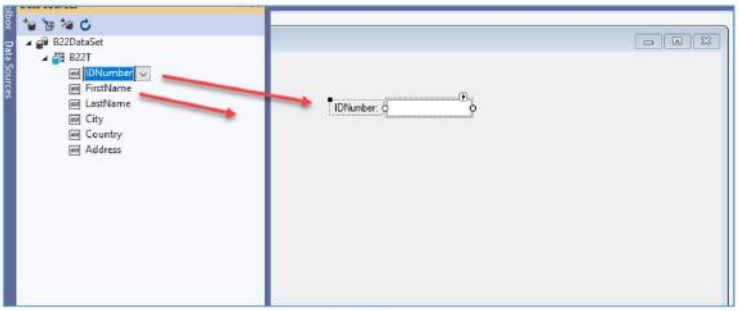
شکل ۱۵۱-۳ ارتباط با SQL

در شکل ۱۵۲-۳ تیک همه‌ی گزینه‌ها را انتخاب و بر روی Next کلیک کنید.



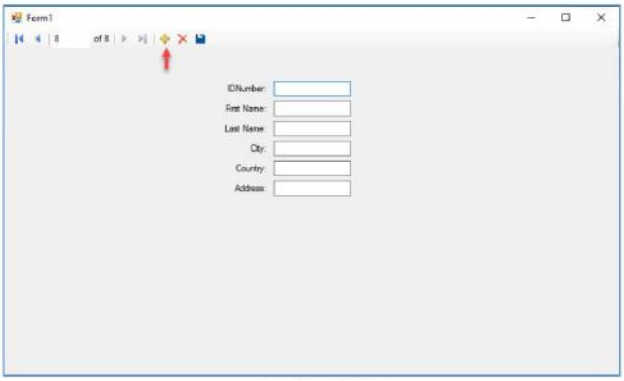
شکل ۱۵۲-۳ ارتباط با SQL

در شکل ۱۵۳-۳ همه‌ی ستون‌های جدول مورد نظر را بکشید و در فرم رها کنید.



شکل ۱۵۳-۳ ایجاد فرم

در شکل ۱۵۴-۳ برای وارد کردن اطلاعات در جدول مورد نظر بر روی + کلیک و اطلاعات را وارد و در آخر بر روی آیکون Save کلیک کنید.



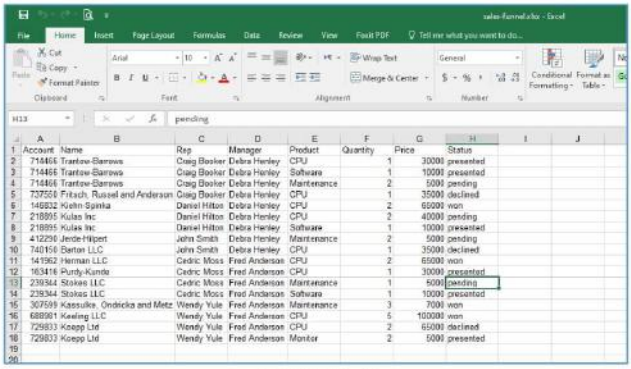
شکل ۱۵۴-۳ ایجاد فرم

۹-۱-۳ واردکردن فایل اکسل در SQL

شما حتماً با فایل‌های اکسل کار کردید و کارهای روزمره‌ی خود را در آن انجام می‌دهید، مانند: اطلاعات حسابداری، ورود و خروج به شرکت و ... که این نرم‌افزار را می‌توان یکی از نرم‌افزارهای پرکاربرد در سطح جهان دانست.

اگر شما یک فایل اکسلی داشته باشید و بخواهید آن را وارد جداول SQL کنید، باید چه کاری انجام دهید؛ آیا این کار شدنی است؟

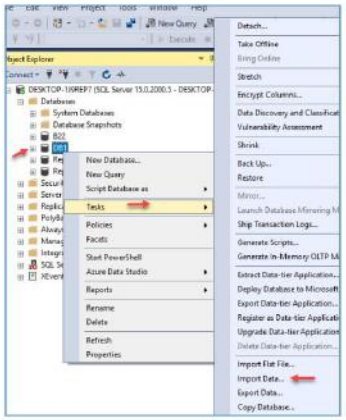
بله، این کار توسط SQL شدنی است و به راحتی می‌توانید فایل‌های اکسل را وارد جداول و دیتابیس SQL کنید؛ برای این کار یک فایل اکسل را در شکل ۱۵۵-۳ مشاهده می‌کنید که دارای ۸ ستون و چندین سطر است که می‌خواهیم آن را وارد SQL Server کنیم.



Account	Name	Rep	Manager	Product	Quantity	Price	Status
714466	Trantow-Barrows	Craig Booker	Debra Henley	CPU	1	30000	presented
714466	Trantow-Barrows	Craig Booker	Debra Henley	Software	1	10000	presented
714466	Trantow-Barrows	Craig Booker	Debra Henley	Maintenance	2	5000	pending
737550	Fritsch, Russel and Anderson	Craig Booker	Debra Henley	CPU	1	35000	declined
146832	Kiehn-Spinka	Daniel Hilton	Debra Henley	CPU	2	65000	won
218895	Kulas Inc	Daniel Hilton	Debra Henley	CPU	2	40000	pending
218895	Kulas Inc	Daniel Hilton	Debra Henley	Software	1	10000	presented
412290	Jerde-Hilpert	John Smith	Debra Henley	Maintenance	2	5000	pending
740150	Barton LLC	John Smith	Debra Henley	CPU	1	35000	declined
141962	Herman LLC	Cedric Moss	Fred Anderson	CPU	2	65000	won
163416	Purdy-Kunde	Cedric Moss	Fred Anderson	CPU	1	30000	presented
239344	Stokes LLC	Cedric Moss	Fred Anderson	Maintenance	1	5000	pending
239344	Stokes LLC	Cedric Moss	Fred Anderson	Software	1	10000	presented
307599	Kassulke, Ondricka and Metz	Wendy Yule	Fred Anderson	Maintenance	3	7000	won
688981	Keeling LLC	Wendy Yule	Fred Anderson	CPU	5	100000	won
729833	Koepp Ltd	Wendy Yule	Fred Anderson	CPU	2	65000	declined
729833	Koepp Ltd	Wendy Yule	Fred Anderson	Monitor	2	5000	presented

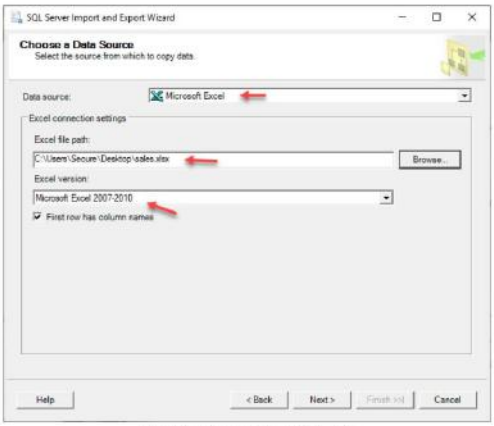
شکل ۱۵۵-۳ *Excel*

در ادامه‌ی کار به‌مانند شکل ۱۵۶-۳ وارد SQL Server شوید و بر روی دیتابیس مورد نظر خود که قرار است فایل اکسل را وارد آن کنید، کلیک راست کنید و از قسمت Tasks، گزینه‌ی Import Data را انتخاب کنید.



شکل ۱۵۶-۳ واردکردن فایل *Excel*

در شکل ۱۵۷-۳ و در قسمت Data source باید Microsoft Excel را انتخاب کنید و بعد بر روی Browse کلیک کنید و فایل مورد نظر خود را انتخاب کنید؛ بعد از انتخاب در قسمت سوم، ورژن آن مشخص خواهد شد؛ برای ادامه‌ی کاربر روی Next کلیک کنید.



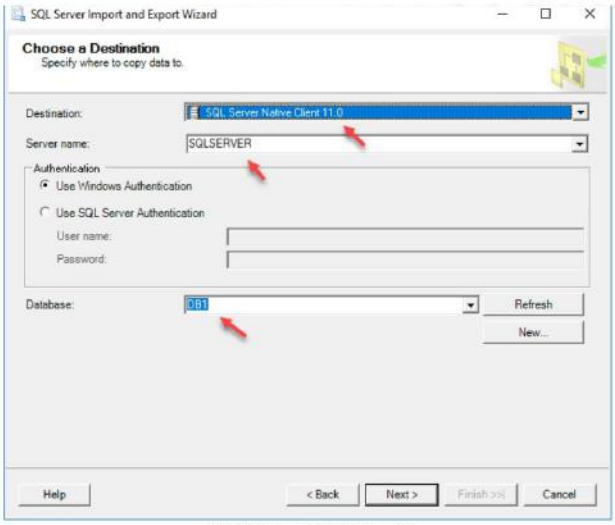
شکل ۱۵۷-۳ واردکردن فایل *Excel*

بعد از کلیک بر روی Next در شکل ۱۵۷-۳ با خطای شکل ۱۵۹-۳ روبرو می‌شوید که برای حل آن باید از لینک زیر، نرم‌افزار Microsoft Access Database Engine 2010 Redistributable را دانلود و نصب کنید تا این خطا برطرف شود.

https://www.microsoft.com/en-us/download/details.aspx?id=13255

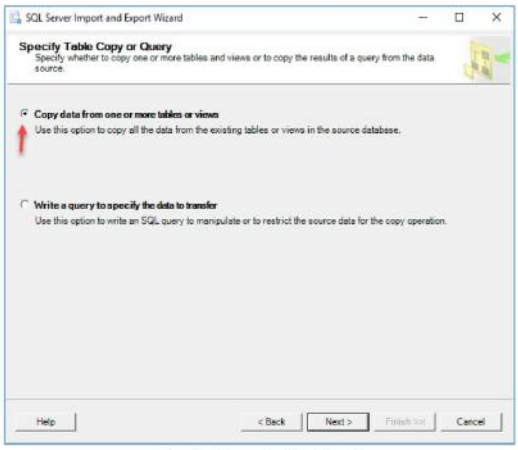
شکل ۱۵۹-۳ خطای فایل *Excel*

بعد از نصب، دوباره مراحل بالا را اجرا کنید و به‌مانند شکل ۱۶۰-۳ در قسمت Destination گزینه‌ی SQL Server Native Client 11.0 را انتخاب کنید؛ در قسمت Server name، نام سرور SQL خود را وارد و در قسمت Database باید دیتابیسی که قرار است اطلاعات اکسل در آن وارد شود را انتخاب و بر روی Next کلیک کنید.



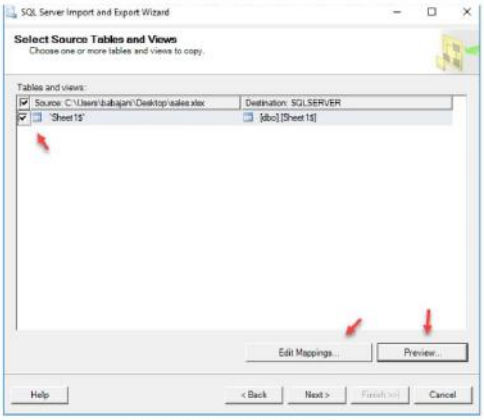
شکل ۱۶۰-۳ واردکردن فایل *Excel*

در شکل ۱۶۱-۳ باید گزینه‌ی اول را انتخاب و بر روی Next کلیک کنید.



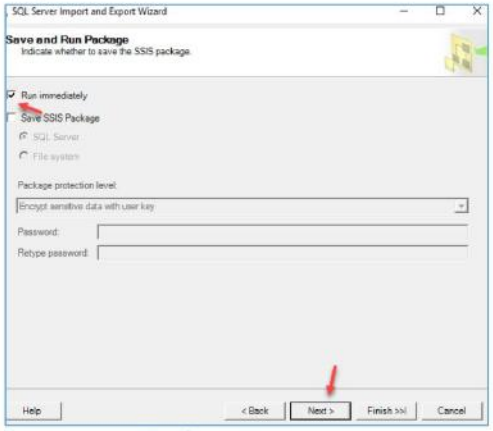
شکل ۱۶۱-۳ واردکردن فایل *Excel*

همان طور که در شکل ۱۶۲-۳ مشاهده می‌کنید، Sheet مورد نظر در اکسل پیدا شده است و شما می‌توانید با کلیک بر روی دکمه‌ی Edit Mappings، ستون‌های آن را ویرایش کنید و همچنین برای نمایش اطلاعات بر روی دکمه‌ی Preview کلیک کنید.



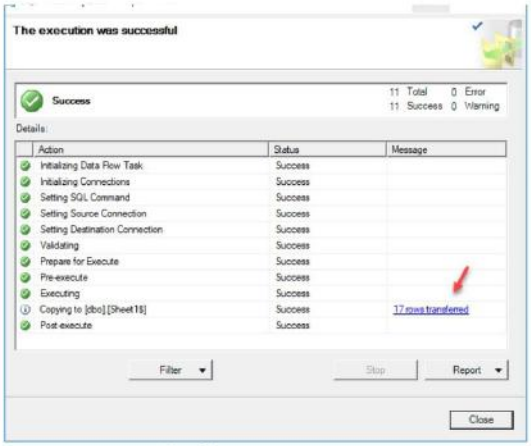
شکل ۱۶۲-۳ واردکردن فایل *Excel*

در شکل ۱۶۳-۳ گزینه‌ی اول را انتخاب کنید و اگر می‌خواهید بر روی اطلاعات، رمز عبور قرار دهید باید گزینه‌ی دوم را انتخاب کنید و در صفحه‌ی آخر بر روی Finish کلیک کنید تا کار انتقال انجام شود.



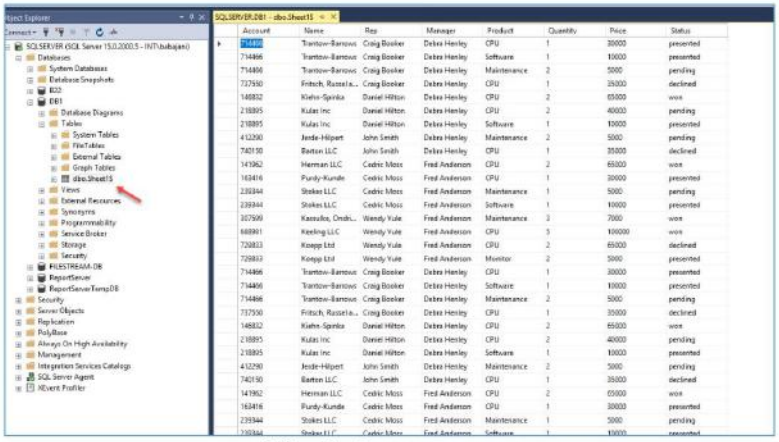
شکل ۱۶۳-۳ واردکردن فایل *Excel*

همان طور که در شکل ۱۶۴-۳ مشاهده می‌کنید، عملیات انتقال با موفقیت انجام شده است.



شکل ۱۶۴-۳ واردکردن فایل *Excel*

همان طور که در شکل ۱۶۵-۳ مشاهده می‌کنید، جدول مورد نظر در دیتابیس DB1 ایجاد شده است و اگر از آن خروجی بگیرید، تمام اطلاعات آن را مشاهده خواهید کرد.



شکل ۱۶۵-۳ اطلاعات فایل *Excel* در *SQL*

۱۰-۱-۳ بررسی دستور Stored Procedures

زمانی که شما یک کد T-SQL برای یک قسمت از بانک اطلاعاتی خود می‌نویسید، شاید دوست داشته باشید همیشه از آن کد استفاده کنید و به این دلیل همیشه باید آن را چندین و چند بار بنویسید یا کپی کنید؛ روشی وجود دارد با عنوان Stored Procedures که دستورات پرکاربرد T-SQL شما را در نام ذخیره می‌کند و دیگر نیاز نیست، کل کد را بنویسید و تنها کافی است آن نام مورد نظر را صدا بزنید.

شکل کلی دستور به‌صورت زیر است:

CREATE PROCEDURE *procedure_name*
AS
sql_statement
GO;

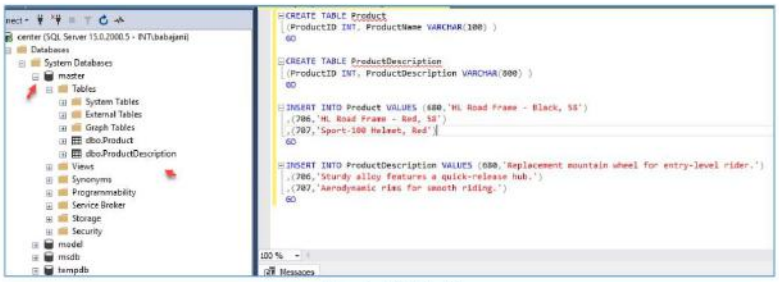
برای اینکه PROCEDURE که ایجاد کردید را فراخوانی کنید باید از دستور زیر استفاده کنید:

EXEC procedure_name;

برای اینکه این دستور را تست بگیریم، یک مثال را با هم بررسی می‌کنیم؛ برای شروع دستورات زیر را اجرا کنید:

```
CREATE TABLE Product
(ProductID INT, ProductName VARCHAR(100) )
GO

CREATE TABLE ProductDescription
(ProductID INT, ProductDescription VARCHAR(800) )
GO

INSERT INTO Product VALUES (680,'HL Road Frame - Black, 58')
,(706,'HL Road Frame - Red, 58')
,(707,'Sport-100 Helmet, Red')
GO

INSERT INTO ProductDescription VALUES (680,'Replacement mountain wheel for entry-level
rider.')
,(706,'Sturdy alloy features a quick-release hub.')
,(707,'Aerodynamic rims for smooth riding.')
GO
```

با اجرای دستورات بالا، دو جدول Product و ProductDescription در دیتابیس master ایجاد می‌شود و داده‌های داخل جدول نیز مشخص شده است که در شکل ۱۶۶-۳ این موضوع نشان داده شده است.



شکل ۱۶۶-۳ ایجاد جدول

بعد از ایجاد جدول، حال می‌خواهیم یک Procedure ایجاد کنیم و اطلاعات را در آن قرار دهیم؛ برای این موضوع از دستور زیر استفاده کنید:

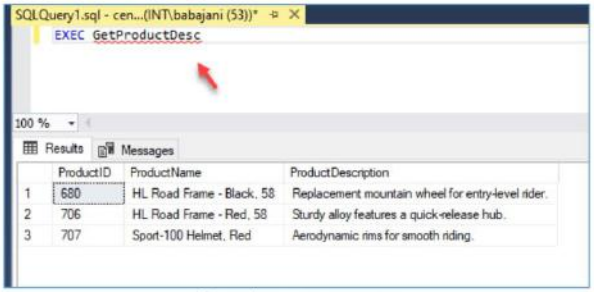
```
CREATE PROCEDURE GetProductDesc
AS
BEGIN
SET NOCOUNT ON

SELECT P.ProductID,P.ProductName,PD.ProductDescription  FROM
Product P
INNER JOIN ProductDescription PD ON P.ProductID=PD.ProductID
```

```
END
```

در دستور بالا، یک Procedure با نام GetProductDesc ایجاد می‌کنیم و در ادامه‌ی دستورات با استفاده از INNER JOIN، دو جدول را به هم متصل می‌کنیم که اطلاعات خروجی در Procedure مورد نظر ذخیره می‌شود.

بعد از ایجاد Procedure باید آن را با دستور زیر فراخوانی کنیم:

```
EXEC GetProductDesc
```



شکل ۱۶۷-۳ اجرای *Procedure*

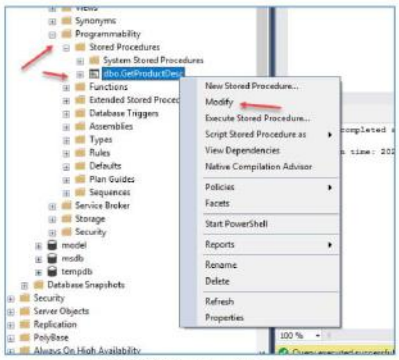
نکته:

به این نکته باید توجه کنید که حتی با بستن پنجره Query و بستن SQL Management Studio باز هم Procedure مورد نظر قابل استفاده است.

۲-۱۰-۱-۳ تغییر یک Stored procedure

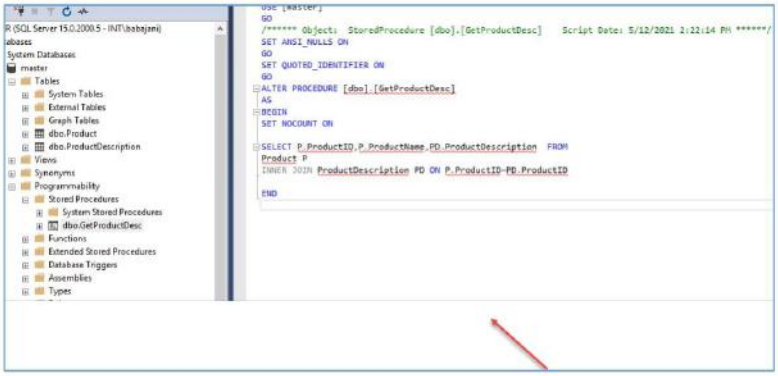
برای تغییر یا اصلاح یک stored procedure موجود از دستور ALTER PROCEDURE استفاده می‌کنیم.

ابتدا فولدر stored procedure را باز کنید تا محتوای آن را ببینید، سپس بر روی نام stored procedure مورد نظر کلیک راست کنید و از آیتم‌های داخل منو، گزینه‌ی Modify را انتخاب کنید:



شکل ۱۷۰-۳ ویرایش *Procedure*

همان طور که در شکل ۱۷۱-۳ مشاهده می‌کنید، می‌توانید کد مورد نظر را تغییر و اطلاعات را ذخیره و اجرا کنید.



شکل۱۷۱-۳ ویرایش *Procedure*

۳-۱۰-۱-۳ حذف یک Stored procedure

برای حذف یک stored procedure می‌توانید از دستور DROP PROCEDURE یا DROP PROC استفاده کنید:

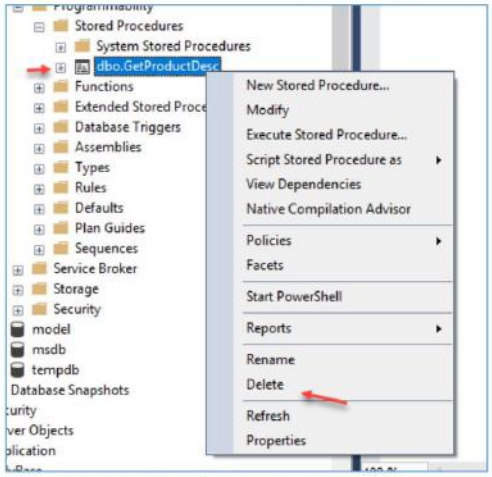
```
DROP PROCEDURE sp_name;
یا
DROP PROC sp_name;
```

که sp_name، نام stored procedure مورد نظر شماست که می‌خواهید حذف شود.

برای مثال، برای حذف stored procedure به نام uspProductList باید دستور زیر را اجرا کنیم:

```
DROP PROCEDURE uspProductList;
```

اگر بخواهید به‌صورت گرافیکی این کار را انجام دهید باید بر روی Procedure مورد نظر کلیک راست کنید و گزینه‌ی DELETE را انتخاب کنید که در شکل ٤٠ مشخص شده است.



شکل ۱۷۲-۳ حذف *Procedure*

در نتیجه، چگونگی مدیریت stored procedureها در SQL Server به‌خصوص اجرا، تغییر و حذف stored procedureها را آموختید.

۱-۱۰-۱-۳ استفاده از پارامتر در دستور Procedure

در قسمت قبل، چگونگی ایجاد یک stored procedure ساده که یک دستور SELECT را پوشش می‌داد را آموختید؛ وقتی stored procedure را فراخوانی می‌کنید، خیلی ساده query داخل آن اجرا می‌شود و یک مجموعه نتیجه را بازمی‌گرداند.

در این قسمت، بحث stored procedure را گسترش می‌دهیم به‌صورتی که خواهیم توانست یک یا چند مقدار را به آن انتقال دهیم؛ نتیجه‌ی stored procedure بر اساس مقادیر پارامترها تغییر خواهد کرد.

ایجاد یک stored procedure با یک پارامتر

Query زیر یک لیست محصول از جدول products در پایگاه‌داده‌ی Bikestores بازمی‌گرداند:

```
SELECT
    product_name,
    list_price
FROM
    production.products
ORDER BY
    list_price;
```

می‌توانید یک stored procedure ایجاد کنید که با استفاده از دستور CREATE PROCEDURE این query را احاطه کند:

```
CREATE PROCEDURE uspFindProducts
AS
BEGIN
    SELECT
        product_name,
        list_price
    FROM
        production.products
    ORDER BY
        list_price;
END;
```

در هر صورت، این دفعه می‌توانیم یک پارامتر به stored procedure اضافه کنیم تا محصولاتی که لیست قیمت‌های آنها بیشتر از یک قیمت ورودی هستند را بیابد:

```
ALTER PROCEDURE uspFindProducts(@min_list_price AS DECIMAL)
AS
BEGIN
    SELECT
        product_name,
        list_price
    FROM
        production.products
    WHERE
```

```
        list_price >= @min_list_price
    ORDER BY
        list_price;
END;
```

در این مثال:

ابتدا، یک پارامتر به نام @min_list_price به stored procedure به نام uspFindProducts اضافه کردیم؛ هر پارامتر باید با علامت @ آغاز شود. کلیدواژه‌های AS DECIMAL، نوع داده‌ی پارامتر @min_list_price را مشخص می‌کنند؛ پارامترها باید با آکولادهای باز و بسته احاطه شوند.

سپس از پارامتر @min_list_price در دستور WHERE، درون دستور SELECT برای فیلتر محصولاتی که لیست قیمت‌های آنها بیشتر یا برابر با @min_list_price هستند، استفاده کردیم.

اجرای یک stored procedure با یک پارامتر

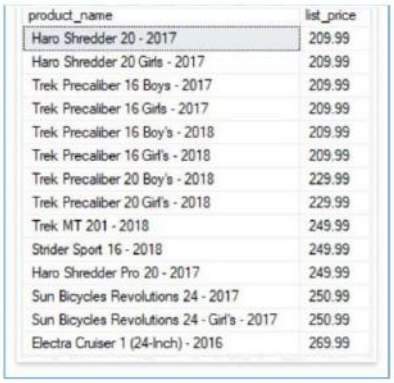
برای اجرای stored procedure باید به‌مانند شکل زیر یک آرگومان به uspFindProducts ارسال کنید:

```
EXEC uspFindProducts 100 ;
```

product_name	list_price
Sun Bicycles Lil Kitt'n - 2017	109.99
Trek Boy's Kickster - 2015/2017	149.99
Trek Girl's Kickster - 2017	149.99
Trek Kickster - 2018	159.99
Trek Precaliber 12 Boys - 2017	189.99
Trek Precaliber 12 Girls - 2017	189.99
Trek Precaliber 12 Boy's - 2018	199.99
Trek Precaliber 12 Girl's - 2018	199.99
Trek Precaliber 16 Boy's - 2018	209.99
Trek Precaliber 16 Girl's - 2018	209.99
Trek Precaliber 16 Boys - 2017	209.99
Trek Precaliber 16 Girls - 2017	209.99
Haro Shredder 20 - 2017	209.99
Haro Shredder 20 Girls - 2017	209.99

stored procedure، تمام محصولاتی که لیست قیمت‌های آنها بیشتر یا برابر با ۱۰۰ هستند را بازمی‌گرداند.

اگر آرگومان را به ۲۰۰ تغییر دهید، مجموعه‌نتایج متفاوتی دریافت خواهید کرد:



product_name	list_price
Haro Shredder 20 - 2017	209.99
Haro Shredder 20 Girls - 2017	209.99
Trek Precaliber 16 Boys - 2017	209.99
Trek Precaliber 16 Girls - 2017	209.99
Trek Precaliber 16 Boy's - 2018	209.99
Trek Precaliber 16 Girl's - 2018	209.99
Trek Precaliber 20 Boy's - 2018	229.99
Trek Precaliber 20 Girl's - 2018	229.99
Trek MT 201 - 2018	249.99
Strider Sport 16 - 2018	249.99
Haro Shredder Pro 20 - 2017	249.99
Sun Bicycles Revolutions 24 - 2017	250.99
Sun Bicycles Revolutions 24 - Girl's - 2017	250.99
Electra Cruiser 1 (24-Inch) - 2016	269.99

ایجاد یک stored procedure با چندین پارامتر

stored procedureها می‌توانند یک یا چند پارامتر بگیرند؛ پارامترها با ویرگول از هم جدا می‌شوند.

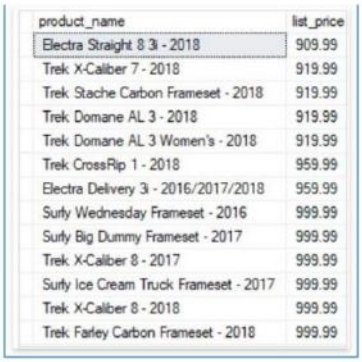
کد زیر stored procedure به نام uspFindProducts را با اضافه کردن یک پارامتر دیگر به نام @max_list_price به آن، تغییر می‌دهد:

```
ALTER PROCEDURE uspFindProducts(
    @min_list_price AS DECIMAL
    ,@max_list_price AS DECIMAL
)
AS
BEGIN
    SELECT
        product_name,
        list_price
    FROM
        production.products
    WHERE
        list_price >= @min_list_price AND
        list_price <= @max_list_price
    ORDER BY
        list_price;
END;
```

وقتی stored procedure با موفقیت تغییر کرد، می‌توانید آن را با ارسال دو آرگومان (یکی برای @min_list_price و دیگری برای @max_list_price) اجرا کنید:

```
EXECUTE uspFindProducts 900, 1000;
```

تصویر زیر خروجی را نشان می‌دهد:



product_name	list_price
Electra Straight 8 3i - 2018	909.99
Trek X-Caliber 7 - 2018	919.99
Trek Stache Carbon Frameset - 2018	919.99
Trek Domane AL 3 - 2018	919.99
Trek Domane AL 3 Women's - 2018	919.99
Trek CrossRip 1 - 2018	959.99
Electra Delivery 3i - 2016/2017/2018	959.99
Surly Wednesday Frameset - 2016	999.99
Surly Big Dummy Frameset - 2017	999.99
Trek X-Caliber 8 - 2017	999.99
Surly Ice Cream Truck Frameset - 2017	999.99
Trek X-Caliber 8 - 2018	999.99
Trek Farley Carbon Frameset - 2018	999.99

استفاده از پارامترهای دارای نام

در صورتی که stored procedureها چندین پارامتر داشته باشند، بهتر و به‌صرفه‌تر است که stored procedureها را با استفاده از پارامترهای دارای نام اجرا کنید.

برای مثال، کد زیر stored procedure به نام uspFindProducts را با استفاده از پارامترهای دارای نام (یعنی @min_list_price و @max_list_price) اجرا می‌کند:

```
EXECUTE uspFindProducts
    @min_list_price = 900,
    @max_list_price = 1000;
```

نتیجه‌ی stored procedure، همان نتیجه‌ی قبلی است، با این تفاوت که حالا کد ما خیلی واضح‌تر و مرتب‌تر است.

ایجاد پارامترهای متنی

کد زیر پارامتر @name را به‌عنوان یک پارامتر رشته‌حروف به stored procedure اضافه می‌کند.

```
ALTER PROCEDURE uspFindProducts(
    @min_list_price AS DECIMAL
    ,@max_list_price AS DECIMAL
    ,@name AS VARCHAR(max)
)
AS
BEGIN
    SELECT
        product_name,
        list_price
    FROM
        production.products
    WHERE
        list_price >= @min_list_price AND
        list_price <= @max_list_price AND
        product_name LIKE '%' + @name + '%'
```

```
    ORDER BY
        list_price;
END;
```

در دستور WHERE از کد SELECT، شرط زیر را اضافه می‌کنیم:

```
product_name LIKE '%' + @name + '%'
```

با این کار، stored procedure محصولاتی که لیست قیمت‌های آن‌ها در بازه‌ی حداقل و حداکثر لیست قیمت‌ها هستند را بازمی‌گرداند و نام محصولات نیز شامل یک قطعه‌متن هستند که ارسال می‌کنید.

وقتی stored procedure با موفقیت تغییر کرد، می‌توانید آن را به‌مانند شکل زیر اجرا کنید:

```
EXECUTE uspFindProducts
    @min_list_price = 900,
    @max_list_price = 1000,
    @name = 'Trek';
```

در این کد از stored procedure به نام uspFindProducts برای یافتن محصولاتی که لیست قیمت آنها در بازه‌ی ۹۰۰ و ۱۰۰۰ و نام آن‌ها شامل واژه Trek باشد استفاده کردیم.

تصویر زیر خروجی را نشان می‌دهد:

product_name	list_price
Trek X-Caliber 7 - 2018	919.99
Trek Stache Carbon Frameset - 2018	919.99
Trek Domane AL 3 - 2018	919.99
Trek Domane AL 3 Women's - 2018	919.99
Trek CrossRip 1 - 2018	959.99
Trek X-Caliber 8 - 2017	999.99
Trek X-Caliber 8 - 2018	999.99
Trek Farley Carbon Frameset - 2018	999.99

ایجاد پارامترهای اختیاری

وقتی uspFinProducts را اجرا می‌کنید، باید تمام سه آرگومان متناظر با سه پارامتر آن را نیز ارسال کنید.

SQL Server به شما اجازه می‌دهد مقادیر پیش‌فرض برای پارامترها مشخص کنید تا وقتی stored procedureها را فراخوانی می‌کنید، می‌توانید پارامترها را با مقادیر پیش‌فرض ارسال کنید.

stored procedure زیر را ببینید:

```
ALTER PROCEDURE uspFindProducts(
    @min_list_price AS DECIMAL = 0
    ,@max_list_price AS DECIMAL = 999999
    ,@name AS VARCHAR(max)
)
AS
BEGIN
    SELECT
        product_name,
```

```
        list_price
    FROM
        production.products
    WHERE
        list_price >= @min_list_price AND
        list_price <= @max_list_price AND
        product_name LIKE '%' + @name + '%'
    ORDER BY
        list_price;
END;
```

در این stored procedure، ما ۰ را به‌عنوان مقدار پیش‌فرض برای پارامتر @min_list_price و ۹۹۹.۹۹۹ را به‌عنوان مقدار پیش‌فرض برای پارامتر @max_list_price قرار داده‌ایم.

وقتی stored procedure کامپایل می‌شود، می‌توانید آن را بدون ارسال آرگومان به پارامترهای @min_list_price و @max_list_price اجرا کنید:

```
EXECUTE uspFindProducts
    @name = 'Trek';
```

product_name	list_price
Trek Boy's Kickster - 2015/2017	149.99
Trek Girl's Kickster - 2017	149.99
Trek Kickster - 2018	159.99
Trek Precaliber 12 Boys - 2017	189.99
Trek Precaliber 12 Girls - 2017	189.99
Trek Precaliber 12 Boy's - 2018	199.99
Trek Precaliber 12 Girl's - 2018	199.99
Trek Precaliber 16 Boy's - 2018	209.99
Trek Precaliber 16 Girl's - 2018	209.99
Trek Precaliber 16 Boys - 2017	209.99
Trek Precaliber 16 Girls - 2017	209.99
Trek Precaliber 20 Boy's - 2018	229.99

در این صورت، stored procedure در زمان اجرای query از ۰ برای پارامتر @min_list_price و از ۹۹۹.۹۹۹ برای پارامتر @max_list_price استفاده می‌کند.

پارامترهای @min_list_price و @max_list_price پارامترهای اختیاری فراخوانی کرده‌اند.

مسلماً، همچنین می‌توانید آرگومان‌ها را به پارامترهای اختیاری نیز ارسال کنیم. برای مثال، کد زیر تمام محصولاتی که لیست قیمت‌های آن‌ها بیشتر یا برابر با ۶۰۰۰ و نام آن‌ها شامل واژه Trek است را بازمی‌گرداند:

```
EXECUTE uspFindProducts
    @min_list_price = 6000,
    @name = 'Trek';
```

product_name	
Trek Silque SLR 8 Women's - 2017	6499.99
Trek Domane SL Frameset - 2018	6499.99
Trek Domane SL Frameset Women's - 2018	6499.99
Trek Emonda SLR 8 - 2018	6499.99
Trek Domane SLR 8 Disc - 2018	7499.99
Trek Domane SLR 9 Disc - 2018	11999.99

استفاده از NULL به‌عنوان مقدار پیش‌فرض

در uspFindProducts، ما از ۹۹۹.۹۹۹ به‌عنوان قیمت حداکثر پیش‌فرض استفاده کردیم. این کار جالبی نیست، چون در آینده ممکن است محصولات با قیمت‌های بیشتر از این مقدار داشته باشید.

یک تکنیک مرسم برای پرهیز از این امر، استفاده از NULL به‌عنوان مقدار پیش‌فرض برای پارامترها است:

```
ALTER PROCEDURE uspFindProducts(
    @min_list_price AS DECIMAL = 0
    ,@max_list_price AS DECIMAL = NULL
    ,@name AS VARCHAR(max)
)
AS
BEGIN
    SELECT
        product_name,
        list_price
    FROM
        production.products
    WHERE
        list_price >= @min_list_price AND
        (@max_list_price IS NULL OR list_price <= @max_list_price) AND
        product_name LIKE '%' + @name + '%'
    ORDER BY
        list_price;
END;
```

در دستور WHERE، ما شرط را تغییر دادیم تا بتوانیم از مقدار NULL برای پارامتر @max_list_price استفاده کنیم:

```
(@max_list_price IS NULL OR list_price <= @max_list_price)
```

کد زیر uspFindProducts را برای یافتن محصولاتی که قیمت آن‌ها بزرگ‌تر یا برابر با ۵۰۰ و نام آن‌ها شامل واژه Haro هستند، اجرا می‌کند.

```
EXECUTE uspFindProducts
    @min_list_price = 500,
    @name = 'Haro';
```

product_name	list_price
Haro SR 1.1 - 2017	539.99
Haro Flightline Two 26 Plus - 2017	549.99
Haro SR 1.2 - 2017	869.99
Haro SR 1.3 - 2017	1409.99
Haro Shift R3 - 2017	1469.99

در این بخش، چگونگی ایجاد و اجرای stored procedure ها با یک یا چند پارامتر را آموختید. همچنین چگونگی ایجاد پارامترهای اختیاری و استفاده از NULL به‌عنوان مقادیر پیش‌فرض برای پارامترها را نیز آموختید.

۱۱-۱-۳ بررسی Trigger در SQL Server

همان طور که می‌دانید SQL معدن Log است و هر کاری که در دیتابیس و دیگر جاهای SQL انجام دهید، یک Log از آن تولید می‌شود که شما می‌توانید با قابلیت Trigger این Logها را بهتر مدیریت کنید و آن چیزی را که بخواهید در خروجی نمایش دهید.

در کل Trigger شامل سه نوع مختلف است:

۱- DML

که این نوع Tigger زمانی اجرا می‌شود که یکی از سه عملیات (Insert, delete, Update) در جدول مورد نظر شما انجام شود.

۲- DDL

در این نوع Trigger زمانی اجرا می‌شود که سه عملیات (Create, alter, drop) اجرا شود.

۳- Logon

این نوع Trigger زمانی فعال خواهد شد که کاربر وارد SQL و خارج شود.

برای اینکه بهتر با موضوع آشنا شویم یک مثال را با هم بررسی می‌کنیم و نحوه کارکرد Trigger را بررسی می‌کنیم.

شکل کلی دستور DML Triggers به‌صورت زیر می‌باشد:

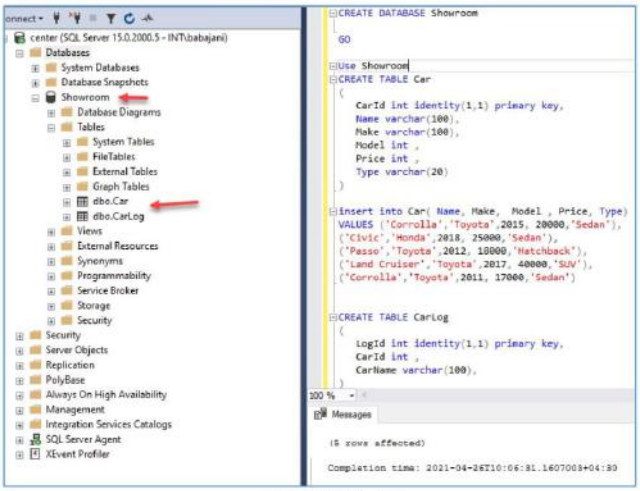
```
CREATE TRIGGER [schema_name.]trigger_name
ON table_name
{FOR | AFTER | INSTEAD OF} {[INSERT] [,] [UPDATE] [,] [DELETE]}
AS
{sql_statements}
```

برای اینکه دستور بالا را بررسی کنیم یک مثال را با هم انجام می‌دهیم، در دستور زیر یک دیتابیس با نام Showroom ایجاد می‌کنیم که این دیتابیس دارای دو جدول CAR و CARLOG است و اطلاعات داخل آنها هم مشخص شده است.

```
CREATE DATABASE Showroom

GO

Use Showroom
CREATE TABLE Car
(
   CarId int identity(1,1) primary key,
   Name varchar(100),
   Make varchar(100),
   Model int ,
   Price int ,
   Type varchar(20)
)
```

```
insert into Car( Name, Make,  Model , Price, Type)
VALUES ('Corrolla','Toyota',2015, 20000,'Sedan'),
('Civic','Honda',2018, 25000,'Sedan'),
('Passo','Toyota',2012, 18000,'Hatchback'),
('Land Cruiser','Toyota',2017, 40000,'SUV'),
('Corrolla','Toyota',2011, 17000,'Sedan')


CREATE TABLE CarLog
(
   LogId int identity(1,1) primary key,
   CarId int ,
   CarName varchar(100),
)
```

دستورات بالا در شکل ۱۷۰-۳ اجرا شده است، و دیتابیس به همراه دو جدول ایجاد شده است.



شکل ۱۷۰-۳ / ایجاد دیتابیس

در ادامه می‌خواهیم از DML Trigger استفاده کنید تا رویدادهایی مانند Insert, Update, Delete در جدول اصلی یعنی در CAR صورت می‌گیرد به‌صورت اتوماتیک در جدول Carlog هم ثبت شود، برای این کار از دستورات زیر استفاده کنید.

```
CREATE TRIGGER [dbo].[CarLOG_INSERT] ON [dbo].[CarLog]
INSTEAD OF INSERT
AS
BEGIN
     DECLARE @car_id INT, @car_name VARCHAR(50)
     SELECT @car_id = INSERTED.CarId,  @car_name = INSERTED.CarName
       FROM INSERTED
       INSERT INTO CarLog
       VALUES(@car_id, @car_name)
    END
```

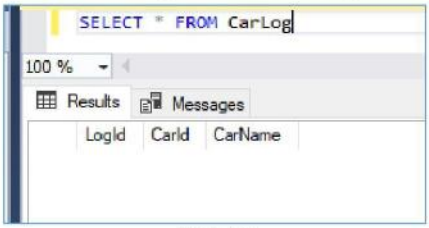
در دستورات بالا یک Trigger جدید با نام CarLOG_INSERT ایجاد می‌شود که زیرمجموعه جدول Carlog است، در قسمت بعدی با استفاده از دستور INSTEAD OF INSERT مشخص می‌کنیم که فقط داده‌هایی که وارد جدول

می‌شوند باید در جدول Carlog ثبت شوند بعدازاین کار با دستور DECLARE دو متغیر با نام‌های @car_id و @car_name تعریف کردیم که و در ادامه با دستور SELECT این دو متغیر را با ستون‌های جدول اصلی برابر قرار دادیم و در ادامه گفتیم که اطلاعات را در جدول CArlog قرار دهد.

برای اینکه عملکرد Trigger را تست بگیریم قبل از هر کاری محتوای جدول Carlog را با دستور زیر مشاهده کنید:

```
SELECT * FROM CarLog
```

همان‌طور که در شکل ۱۷۱-۳ مشاهده می‌کنید جدول کاملاً خالی است



شکل ۱۷۱-۳

برای اینکه کارایی را مشاهده کنیم باید با دستور زیر اطلاعاتی را در جدول Car که جدول اصلی است وارد کنیم و بعد از آن به‌صورت اتوماتیک اطلاعات در جدول Carlog ثبت خواهد شد.

```
insert into Car( Name, Make,  Model , Price, Type)
VALUES ('Pride','Kia',2021, 100000,'Sedan')
```

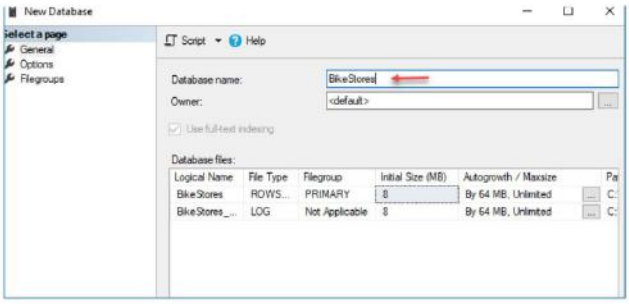
بعد از واردکردن دستورات بالا اطلاعات جدید در جدول Car ثبت خواهد شد و بعد از آن به‌صورت اتوماتیک Car ID و Car Name در جدول Carlog ثبت خواهند شد.

۱۲-۱-۳ توابع در SQL Server 2019

زمانی که یک پروژه سنگین را به اتمام می‌رسانید و این پروژه دارای هزار خط کد است که اگر چنانچه با مشکل روبرو شود پیداکردن خطا در آن بسیار پیچیده خواهد شد، با استفاده از توابع در SQL شما می‌توانید یک برنامه را بهبود ببخشید و برنامه را بهتر کنترل کنید، اگر بخواهیم به‌صورت کلی بگوییم با استفاده از توابع، برنامه ما به قطعه‌های منطقی جداگانه تبدیل می‌شود و یک تابع به این صورت عمل می‌کند که یک سری پارامتر را از ورودی می‌گیرد و عملیاتی را بر روی آن انجام می‌دهد و در خروجی نمایش می‌دهد، تابع را می‌توان یک شی در نظر گرفت که بعد از ایجاد در SQL ذخیره شده و هر زمان بخواهید می‌توانید آن را صدا بزنید.

برای اینکه توابع را بررسی کنیم یک دیتابیس به همراه جداول و مقادیر آن را در SQL ایجاد و در ادامه توابع را بر روی آنها اجرا خواهیم کرد:

به‌مانند شکل زیر یک Database با نام BikeStores ایجاد کنید، توجه کنید که دقیقاً این اسم را وارد کنید تا در ادامه تمرینات با مشکل روبرو نشویم.



بعد از ایجاد Database بالا با دستورات زیر جداول آن را ایجاد کنید:

برای دانلود آن هم می‌توانید به ID من در تلگرام (@farshidbabajani) پیام دهید.

```
CREATE SCHEMA production;
go

CREATE SCHEMA sales;
go

-- create tables
CREATE TABLE production.categories (
	category_id INT IDENTITY (1, 1) PRIMARY KEY,
	category_name VARCHAR (255) NOT NULL
);

CREATE TABLE production.brands (
	brand_id INT IDENTITY (1, 1) PRIMARY KEY,
	brand_name VARCHAR (255) NOT NULL
);

CREATE TABLE production.products (
	product_id INT IDENTITY (1, 1) PRIMARY KEY,
	product_name VARCHAR (255) NOT NULL,
	brand_id INT NOT NULL,
	category_id INT NOT NULL,
	model_year SMALLINT NOT NULL,
	list_price DECIMAL (10, 2) NOT NULL,
	FOREIGN KEY (category_id) REFERENCES production.categories (category_id) ON
DELETE CASCADE ON UPDATE CASCADE,
	FOREIGN KEY (brand_id) REFERENCES production.brands (brand_id) ON DELETE
CASCADE ON UPDATE CASCADE
);

CREATE TABLE sales.customers (
	customer_id INT IDENTITY (1, 1) PRIMARY KEY,
	first_name VARCHAR (255) NOT NULL,
	last_name VARCHAR (255) NOT NULL,
	phone VARCHAR (25),
	email VARCHAR (255) NOT NULL,
	street VARCHAR (255),
	city VARCHAR (50),
	state VARCHAR (25),
	zip_code VARCHAR (5)
);
```