**فاز سوم**

اعضای گروه:

محمدمهدی احمدی

حوری دهش

سارا سلطانی گردفرامرزی

موضوع پروژه:

سامانه مدیریت کارواش

برای ران شدن پروژه به ترتیب فایل های زیر را ران کنید:

create\_tables

functions

procedures

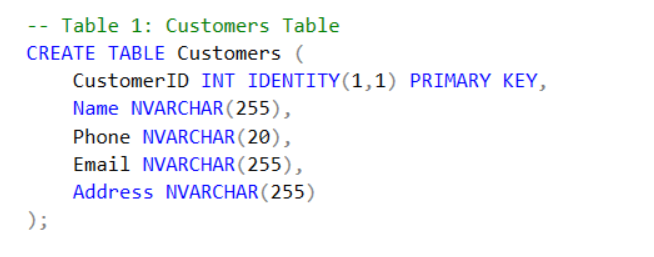
triggers

insert

views

**create\_tables**

1. جدول Customers:



CustomerID: یک ID یکتا برای هر مشتری است که به صورت خودکار افزایش می یابد (Identity) و این کلید اصلی جدول می باشد

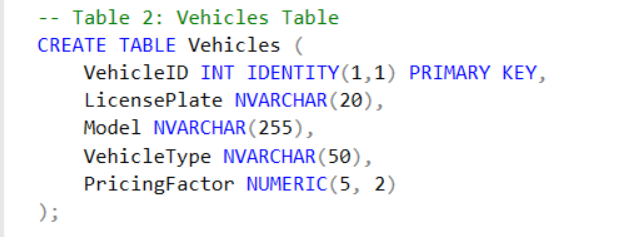
Name: نام مشتری

Phone: شماره تلفن مشتری

Email: آدرس ایمیل مشتری

Address: آدرس مشتری

2. جدول Vehicles:



VehicleID: یک ID یکتا برای هر خودرو است که به صورت خودکار افزایش می‌ یابد (Identity) و این کلید اصلی جدول می باشد

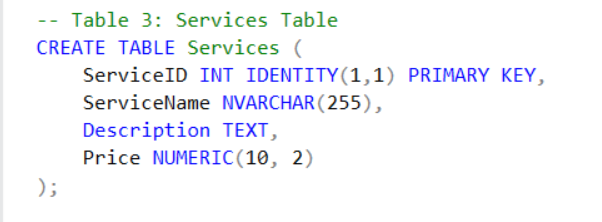
LicensePlate: پلاک خودرو

Model: مدل خودرو

VehicleType: نوع خودرو

PricingFactor: یک ضریب قیمتی برای هر خودرو است که متناسب با خودروها متفاوت می باشد

3. جدول Services:



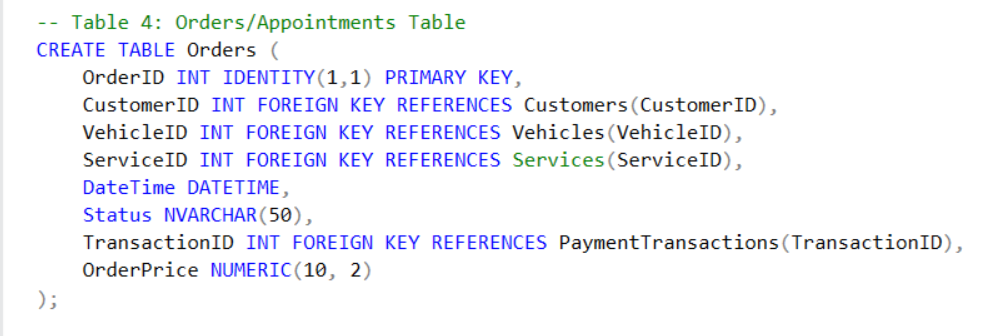
ServiceID: یک ID یکتا برای هر خدماتی است که ارائه میشود و به صورت خودکار افزایش می ‌یابد (Identity) و این کلید اصلی جدول می باشد

ServiceName: نام آن سرویسی است که ارائه شده است

Description: توضیحات بیشتر درباره سرویس ها یا خدمات

Price: قیمت آن سرویس است

4. جدول Orders/Appointments:



OrderID: یک ID یکتا برای هر سفارش یا قرار ملاقات هر مشتری است که به صورت خودکار افزایش می ‌یابد (Identity) و این کلید اصلی جدول می باشد

CustomerID: کلید خارجی به جدول Customers است

VehicleID: کلید خارجی به جدول Vehicles است

ServiceID: کلید خارجی به جدول Services است

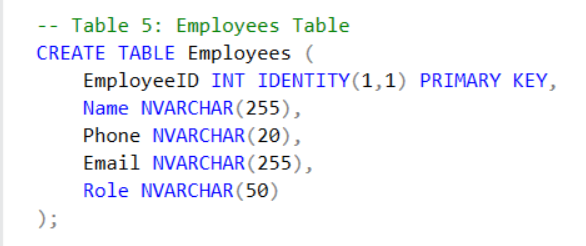
DateTime: تاریخ و زمان سفارش یا قرار ملاقات

Status: وضعیت سفارش (مثلا در حالت pending یا complete است)

TransactionID: کلید خارجی به جدول PaymentTransactions است

OrderPrice: قیمت کل سفارش می باشد

5. جدول Employees:



EmployeeID: یک ID یکتا برای هر کارمند است که به صورت خودکار افزایش می‌ یابد (Identity)و این کلید اصلی جدول می باشد

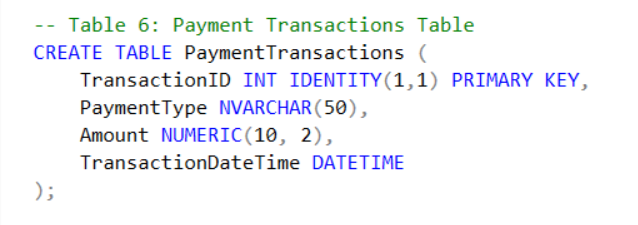
Name: نام کارمند

Phone: شماره تلفن کارمند

Email: آدرس ایمیل کارمند

Role: نقش کارمند در کارواش

6. جدول Payment Transactions:



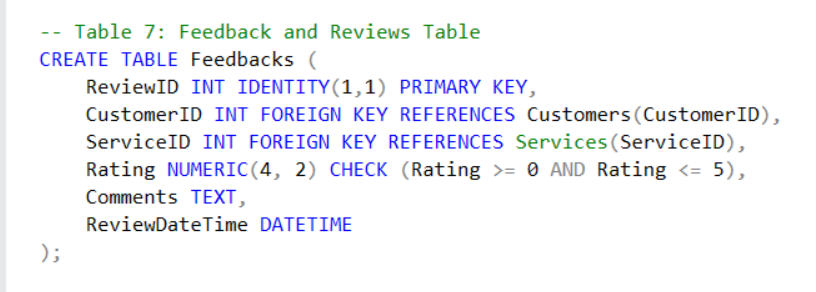
TransactionID: یک ID یکتا برای هر تراکنش پرداختی است که به صورت خودکار افزایش می ‌یابد (Identity) و این کلید اصلی جدول می باشد

PaymentType: نوع پرداخت است

Amount: مقدار پرداخت

TransactionDateTime: تاریخ و زمان انجام تراکنش پرداختی

7. جدول Feedback and Reviews:



ReviewID: یک ID یکتا برای هر بازخورد و نقد است که به صورت خودکار افزایش می‌ یابد (Identity) و این کلید اصلی جدول می باشد

CustomerID: کلید خارجی به جدول Customers است

ServiceID: کلید خارجی به جدول Services است

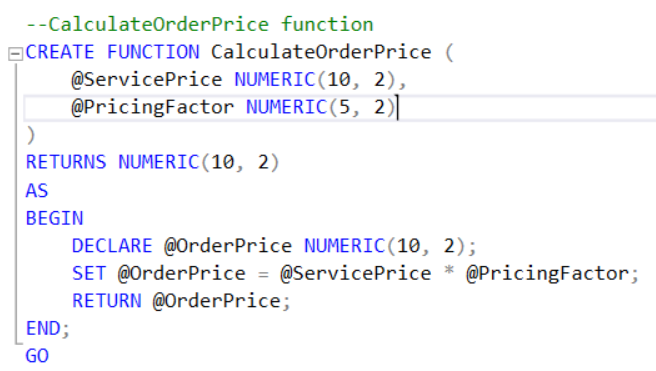
Rating: امتیاز داده شده برای آن خدماتی که گرفته اند

Comments: نظرات و توضیحات اضافی

ReviewDateTime: تاریخ و زمان ارسال بازخورد

**functions**

1. CalculateOrderPrice function:



از تابع CalculateOrderPrice برای محاسبه قیمت نهایی یک سفارش، بر اساس قیمت سرویسی که استفاده کرده و فاکتور قیمت‌ ارائه شده، استفاده میشود.

ورودی ها:

ServicePrice: یک عدد اعشاری با دقت 10 رقم در کل و 2 رقم در ممیز، که نمایانگر قیمت خدمت مورد نظر در جدول Services است.

PricingFactor: یک عدد اعشاری با دقت 5 رقم در کل و 2 رقم در ممیز، که نمایانگر فاکتور قیمت مورد استفاده برای محاسبه قیمت سفارش است.

عملیات:

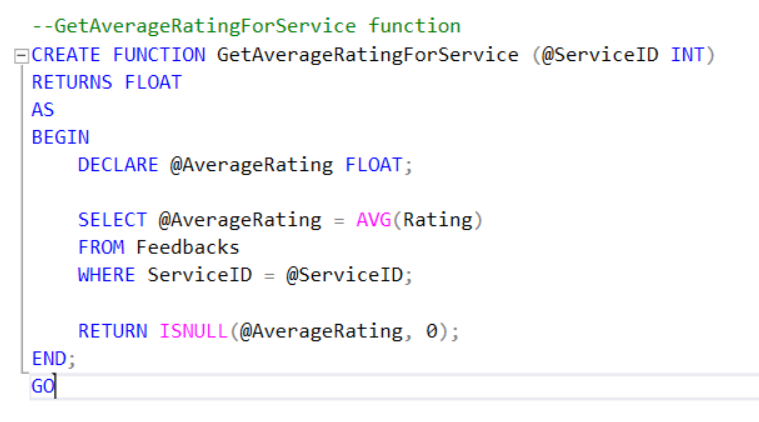
تابع ابتدا یک متغیر محلی به نام OrderPrice ایجاد می ‌کند که برای ذخیره قیمت نهایی سفارش استفاده می ‌شود.

سپس با استفاده از فرمول ServicePrice \* PricingFactor، قیمت سفارش محاسبه می ‌شود.

خروجی:

تابع مقدار محاسبه شده برای قیمت سفارش را به عنوان خروجی باز می‌ گرداند یعنی OrderPrice را که این مقدار یک عدد اعشاری با دقت 10 رقم در کل و 2 رقم در ممیز است.

2. GetAverageRatingForService function:



از تابع GetAverageRatingForService برای محاسبه میانگین امتیازها برای یک سرویس مشخص استفاده می شود.

ورودی:

ServiceID: یک عدد صحیح که نشان‌ دهنده ID یک سرویس در جدول Services است.

عملیات:

تابع ابتدا یک متغیر محلی به نام AverageRating ایجاد می ‌کند که برای ذخیره میانگین امتیازها استفاده می ‌شود.

سپس با استفاده از دستور

SELECT @AverageRating = AVG(Rating) FROM Feedbacks

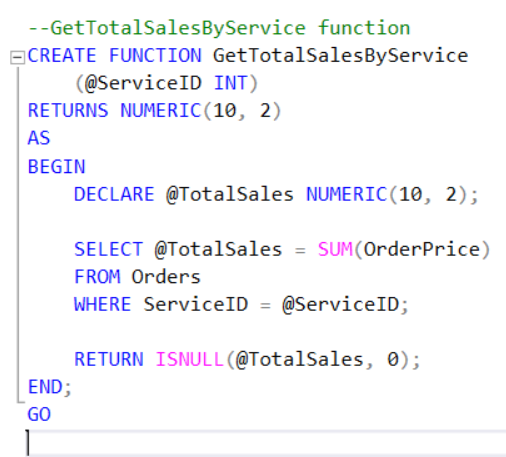
WHERE ServiceID = @ServiceID

میانگین امتیازها برای سرویس مشخص شده در ورودی محاسبه می ‌شود.

خروجی:

تابع مقدار محاسبه شده برای میانگین امتیازها را به عنوان خروجی باز می‌ گرداند و اگر هیچ امتیازی برای سرویس مورد نظر وجود نداشته باشد یعنی سرویس امتیازی دریافت نکرده باشد از تابع ISNULL(@AverageRating, 0) استفاده میشود که یک مقدار پیش فرض صفر را برمی گرداند.

3. GetTotalSalesByService function:



از تابع GetTotalSalesByService برای محاسبه کل فروش بر اساس یک سرویس مشخص شده استفاده میشود.

ورودی:

ServiceID: یک عدد صحیح که نشان‌ دهنده ID یک سرویس در جدول Services است.

عملیات:

تابع ابتدا یک متغیر محلی به نام TotalSales ایجاد می‌ کند که برای ذخیره مجموع فروش استفاده می ‌شود.

سپس با استفاده از دستور

SELECT @TotalSales = SUM(OrderPrice) FROM Orders

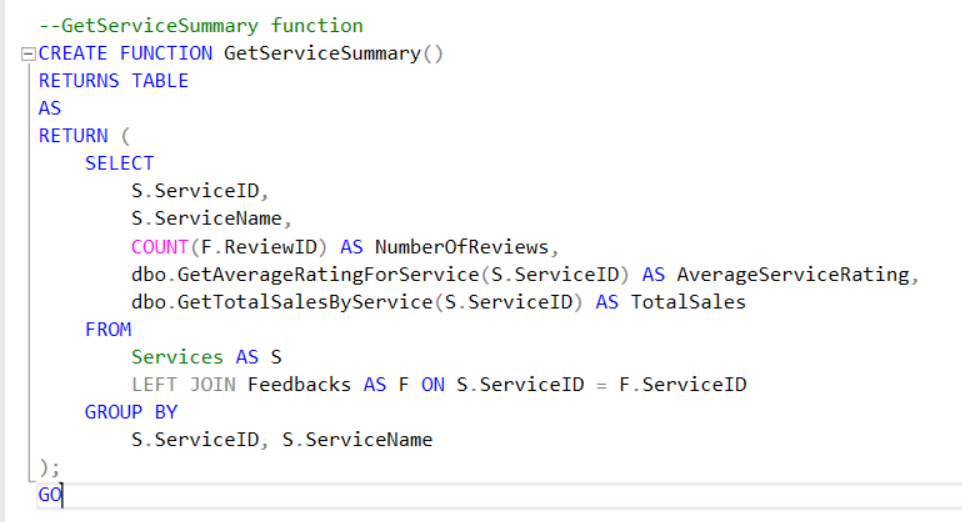
WHERE ServiceID = @ServiceID

مجموع فروش برای سرویس مشخص شده در ورودی محاسبه می ‌شود.

خروجی:

تابع مقدار محاسبه شده برای مجموع فروش را به عنوان خروجی باز می ‌گرداند و اگر هیچ فروشی برای سرویس مورد نظر وجود نداشته باشد یعنی هیچ سفارشی برای سرویس مورد نظر در جدول Orders وجود نداشته باشد، از تابع ISNULL(@TotalSales, 0) استفاده میشود که یک مقدار پیش فرض صفر را برمی گرداند.

4. GetServiceSummary function:



از تابع GetServiceSummary برای دریافت خلاصه‌ای از اطلاعات مرتبط با خدمات شامل تعداد نظرات، میانگین امتیازات و کل فروش استفاده میشود.

تابع GetServiceSummary یک جدول خروجی ایجاد می ‌کند که شامل ستون‌های زیر است:

ServiceID: ID سرویس از جدول Services

ServiceName: نام سرویس از جدول Services

NumberOfReviews: تعداد نظرات مرتبط با سرویس از جدول Feedbacks

AverageServiceRating: میانگین امتیازهای مرتبط با سرویس که با استفاده از تابع GetAverageRatingForService محاسبه می ‌شود.

TotalSales: کل فروش مرتبط با سرویس که با استفاده از تابع GetTotalSalesByService محاسبه میشود.

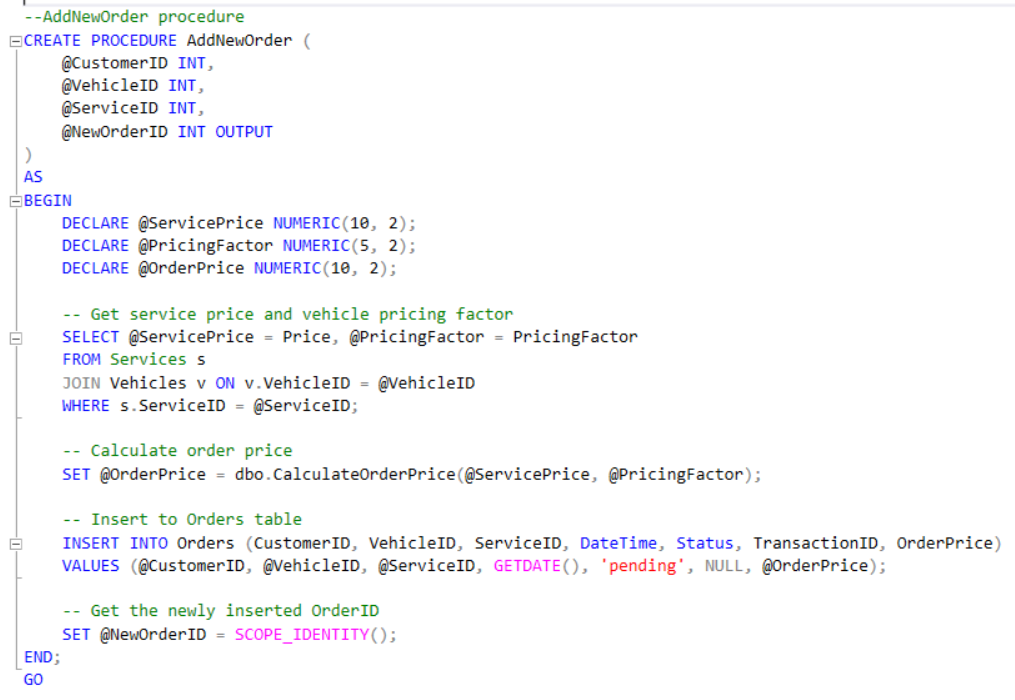
عملیات:

در داخل تابع یک جدول موقت به نام S برای جدول Services و یک جدول موقت به نام F برای جدول Feedbacks تعریف می‌ شود. سپس از یک LEFT JOIN بین جدول Services و جدول Feedbacks با شرط ارتباط بین ServiceID در هر دو جدول استفاده می‌ شود در نهایت با group by کردن بر اساس ستون ‌های ServiceID و ServiceName در جدول Services، تعداد نظرات، میانگین امتیازات و کل فروش برای هر سرویس محاسبه می ‌شود.

و در آخر این تابع یک جدول به عنوان خروجی برمی گرداند.

**procedures**

1. AddNewOrder procedure:



از پروسیجر AddNewOrder برای افزودن یک سفارش جدید به جدول Orders استفاده میشود.

ورودی ها:

CustomerID: ID مشتری که سفارش را ثبت می‌ کند

VehicleID: ID وسیله نقلیه مرتبط با سفارش

ServiceID: ID سرویس مرتبط با سفارش

NewOrderID: یک پارامتر OUTPUT که برای دریافت ID سفارش جدید بعد از ثبت استفاده می ‌شود

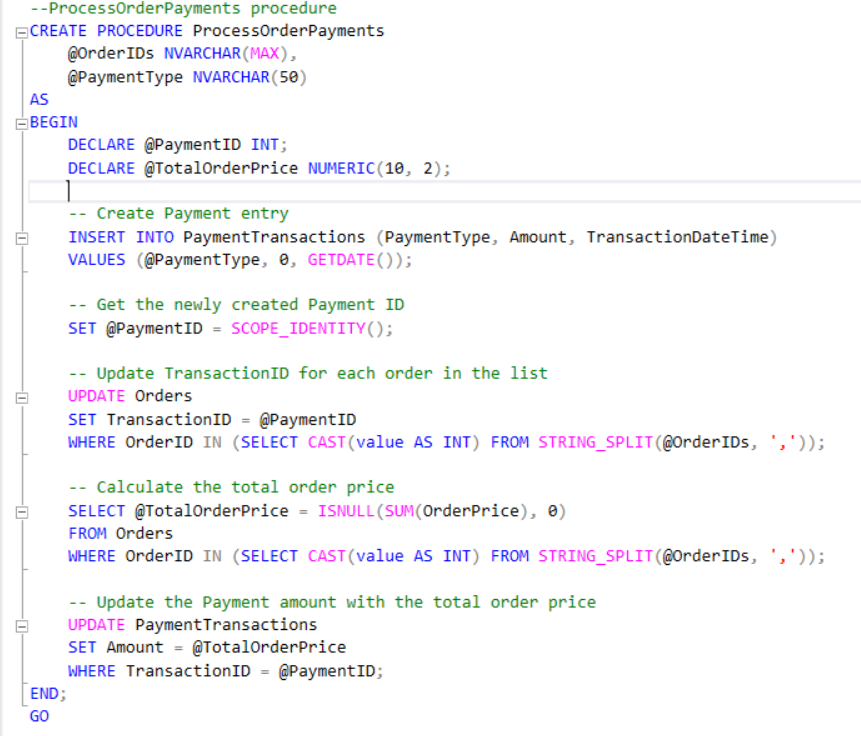
عملیات:

ابتدا متغیرهای محلی ServicePrice و PricingFactor برای ذخیره قیمت سرویس و فاکتور قیمت‌ وسیله نقلیه تعریف می شوند سپس اطلاعات این دو تا متغییر از جدول Services و Vehicles با استفاده از یک JOIN به دست می‌ آیند.

با فراخوانی تابع CalculateOrderPrice قیمت سفارش یعنی OrderPrice براساس قیمت سرویس و فاکتور قیمت‌ وسیله نقلیه محاسبه می ‌شود.

در آخر یک رکورد جدید به جدول Orders اضافه می ‌شود که این رکورد شامل اطلاعات مشتری، وسیله نقلیه، سرویس، تاریخ و وضعیت سفارش و قیمت سفارش است سپس با استفاده از SCOPE\_IDENTITY، شناسه سفارش جدید به عنوان مقدار پارامتر NewOrderID تنظیم می ‌شود. این مقدار به کاربر ارائه میشود تا بتواند شناسه سفارش جدید را دریافت کند.

2. ProcessOrderPayments procedure:



از پروسیجر ProcessOrderPayments برای پردازش پرداخت‌ های مرتبط با یک یا چند سفارش، ایجاد تراکنش پرداخت جدید و به ‌روزرسانی اطلاعات مربوط به پرداخت استفاده میشود.

ورودی ها:

OrderIDs: یک رشته است که حاوی IDهای سفارشات جدا شده با ویرگول است. این ID‌ها با استفاده از تابع STRING\_SPLIT از رشته جدا می‌ شوند و به صورت لیست به پروسیجر وارد می‌ شوند.

PaymentType: نوع پرداخت مورد نظر برای ثبت در جدول PaymentTransactions است.

عملیات:

ابتدا یک رکورد جدید برای پرداخت در جدول PaymentTransactions ایجاد میشود. مقدار پیش فرض برای Amount به صورت صفر است و با استفاده از تابع GETDATE تاریخ و زمان فعلی ثبت میشود.

با استفاده از SCOPE\_IDENTITY، شناسه پرداخت جدید به متغیر PaymentID تنظیم می ‌شود. همچنین با استفاده از UPDATE، شناسه تراکنش برای هر سفارش موجود در لیست OrderIDs در جدول Orders به ID پرداخت جدید تنظیم می‌ شود.

سپس با استفاده از دستور

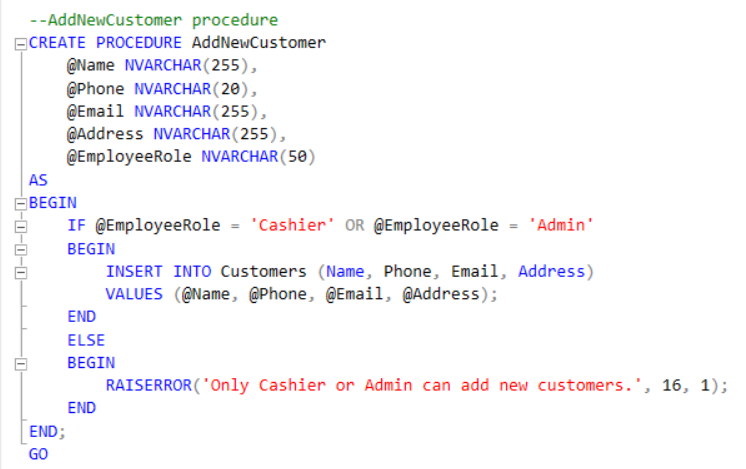
SELECT @TotalOrderPrice = ISNULL(SUM(OrderPrice), 0) FROM Orders

WHERE OrderID IN (SELECT CAST(value AS INT) FROM STRING\_SPLIT(@OrderIDs, ','))

مجموع قیمت سفارشات مرتبط با پرداخت محاسبه می ‌شود.

در نهایت با استفاده از UPDATE دیگری، مقدار Amount در رکورد مربوط به پرداخت در جدول PaymentTransactions به مقدار مجموع قیمت سفارشات تنظیم می ‌شود.

3. AddNewCustomer procedure:



از پروسیجر AddNewCustomer برای افزودن مشتری جدید به جدول Customers استفاده می ‌شود.

ورودی ها:

Name: نام مشتری جدید

Phone: شماره تلفن مشتری جدید

Email: آدرس ایمیل مشتری جدید

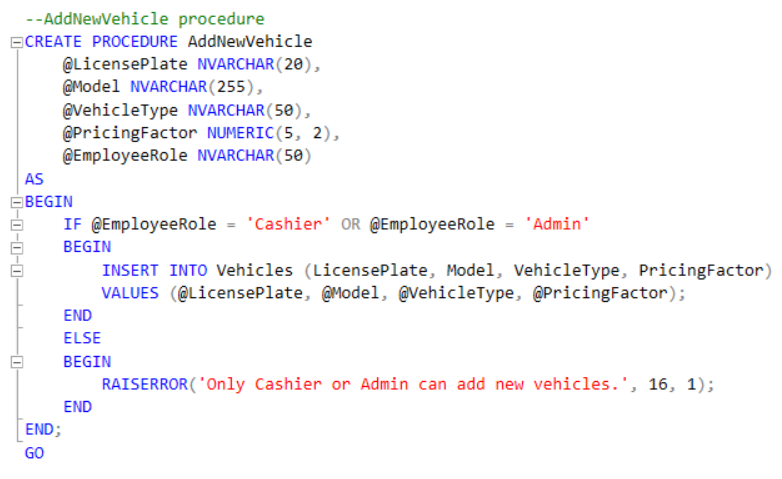
Address: آدرس مشتری جدید

EmployeeRole: نقش کارمند

عملیات:

ابتدا با استفاده از یک شرط IF بررسی می ‌شود که نقش کارمندی که این عملیات را انجام می‌دهد، Cashier یا Admin است یا هیچکدام. اگر کارمند نقش Cashier یا Admin را داشته باشد یک رکورد جدید به جدول Customers اضافه میشود با اطلاعات مشتری جدید که از ورودی ‌های پروسیجر گرفته شده‌اند و در صورتی که نقش کارمند نامعتبر باشد یعنی نه Cashier و نه Admin باشد، با اجرای RAISEERROR یک پیام خطا ایجاد می ‌شود و اجرای پروسیجر متوقف می ‌شود.

4. :AddNewVehicle procedure



از پروسیجر AddNewVehicle برای افزودن وسیله نقلیه جدید به جدول Vehicles استفاده می ‌شود.

ورودی ها:

LicensePlate: پلاک وسیله نقلیه جدید

Model: مدل وسیله نقلیه جدید

VehicleType: نوع وسیله نقلیه جدید

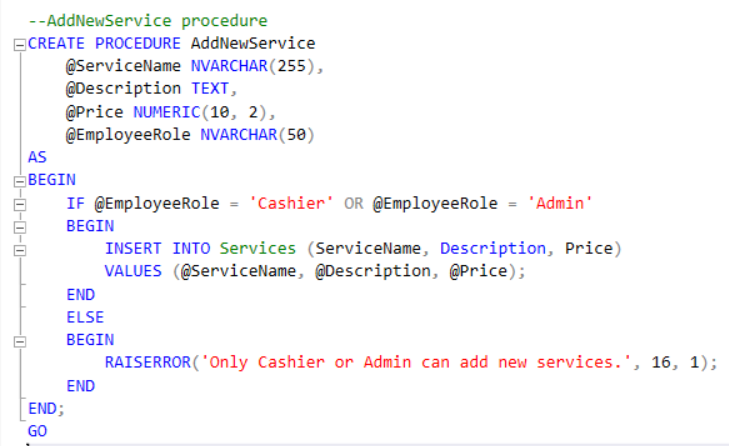
PricingFactor: فاکتور قیمت‌ برای وسیله نقلیه جدید

EmployeeRole: نقش کارمند

عملیات:

ابتدا با استفاده از یک شرط IF بررسی می ‌شود که نقش کارمندی که این عملیات را انجام می‌دهد، Cashier یا Admin است یا هیچکدام. اگر کارمند نقش Cashier یا Admin را داشته باشد یک رکورد جدید به جدول Vehicles اضافه میشود با اطلاعات وسیله نقلیه جدید که از ورودی‌ های پروسیجر گرفته شده‌اند و در صورتی که نقش کارمند نامعتبر باشد یعنی نه Cashier و نه Admin باشد، با اجرای RAISEERROR یک پیام خطا ایجاد می ‌شود و اجرای پروسیجر متوقف می ‌شود.

5. :AddNewService procedure



از پروسیجر AddNewService برای افزودن سرویس جدید به جدول Services استفاده می‌شود.

ورودی ها:

ServiceName: نام سرویس جدید

Description: توضیحات سرویس جدید

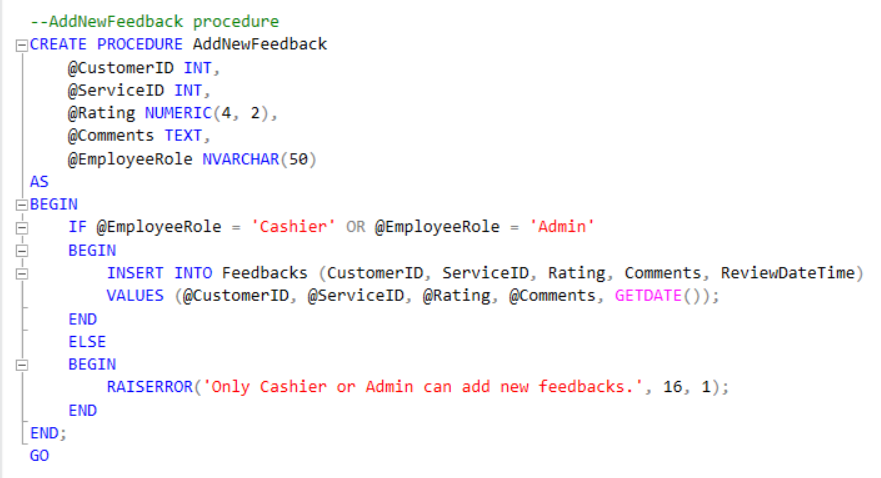
Price: قیمت سرویس جدید

EmployeeRole: نقش کارمند

عملیات:

ابتدا با استفاده از یک شرط IF بررسی می ‌شود که نقش کارمندی که این عملیات را انجام می‌دهد، Cashier یا Admin است یا هیچکدام. اگر کارمند نقش Cashier یا Admin را داشته باشد یک رکورد جدید به جدول Services اضافه میشود با اطلاعات سرویس جدید که از ورودی ‌های پروسیجر گرفته شده‌اند و در صورتی که نقش کارمند نامعتبر باشد یعنی نه Cashier و نه Admin باشد، با اجرای RAISEERROR یک پیام خطا ایجاد می ‌شود و اجرای پروسیجر متوقف می ‌شود.

6. :AddNewFeedback procedure



از پروسیجر AddNewFeedback برای افزودن بازخورد جدید به جدول Feedbacks استفاده می‌ شود.

ورودی ها:

CustomerID: ID مشتری که بازخورد را ثبت می‌ کند

ServiceID: ID سرویس مورد بازخورد

Rating: امتیاز اختصاص داده شده به سرویس

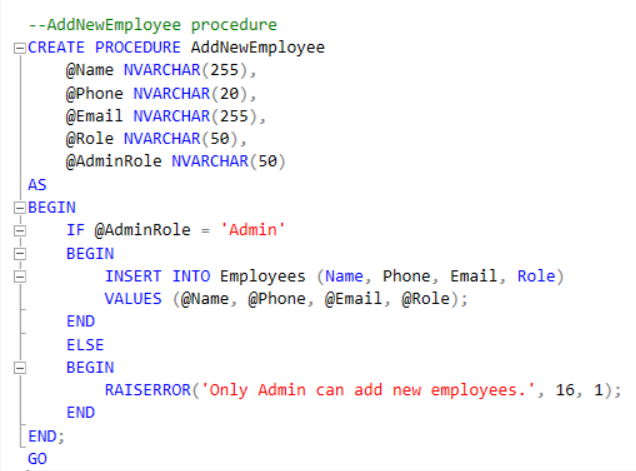
Comments: نظرات یا بازخورد مشتری

EmployeeRole: نقش کارمند

عملیات:

ابتدا با استفاده از یک شرط IF بررسی می ‌شود که نقش کارمندی که این عملیات را انجام می‌دهد، Cashier یا Admin است یا هیچکدام. اگر کارمند نقش Cashier یا Admin را داشته باشد یک رکورد جدید به جدول Feedbacks اضافه میشود با اطلاعات بازخورد که از ورودی ‌های پروسیجر گرفته شده‌اند. همچنین تاریخ و زمان ثبت بازخورد با استفاده از GETDATE تنظیم می ‌شود. در صورتی که نقش کارمند نامعتبر باشد یعنی نه Cashier و نه Admin باشد، با اجرای RAISEERROR یک پیام خطا ایجاد می ‌شود و اجرای پروسیجر متوقف می ‌شود.

7. :AddNewEmployee procedure



از پروسیجر AddNewEmployee برای افزودن کارمند جدید به جدول Employees استفاده می ‌شود.

ورودی ها:

Name: نام کارمند جدید

Phone: شماره تلفن کارمند جدید

Email: آدرس ایمیل کارمند جدید

Role: نقش کارمند جدید

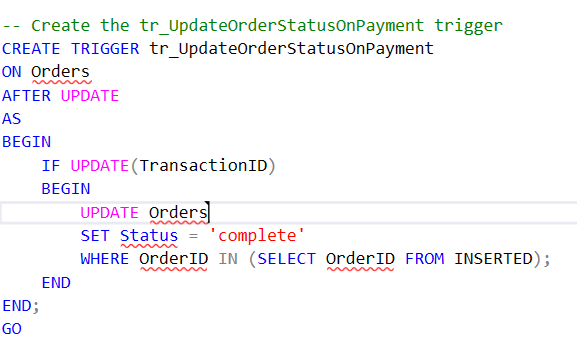
AdminRole: نقش مدیر سیستم که عملیات اضافه کردن کارمند را انجام می‌دهد

عملیات:

ابتدا با استفاده از یک شرط IF بررسی می ‌شود که نقش مدیر سیستمی که این عملیات را انجام می‌دهد، Admin است یا نه. اگر مدیر سیستم نقش Admin را داشته باشد یک رکورد جدید به جدول Employees اضافه میشود با اطلاعات کارمند جدید که از ورودی‌ های پروسیجر گرفته شده‌اند و در صورتی که نقش مدیر سیستم نامعتبر باشد یعنی Admin نباشد، با اجرای RAISEERROR یک پیام خطا ایجاد می ‌شود و اجرای پروسیجر متوقف می ‌شود.

**triggers**

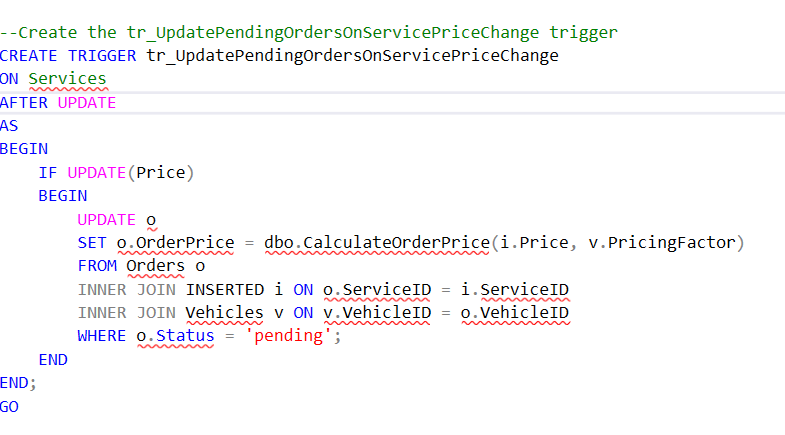
1. : tr\_UpdateOrderStatusOnPayment trigger



عملیات: به طور خلاصه، این تریگر به‌طور خودکار «وضعیت» سفارش‌ها را هنگامی که ستون «TransactionID» آپدیت می‌شود، به مقدار «complete » تغییرمیدهد. فرض بر این است که به‌روزرسانی «TransactionID» به معنای تکمیل تراکنش یا پرداخت مربوط به سفارش است.

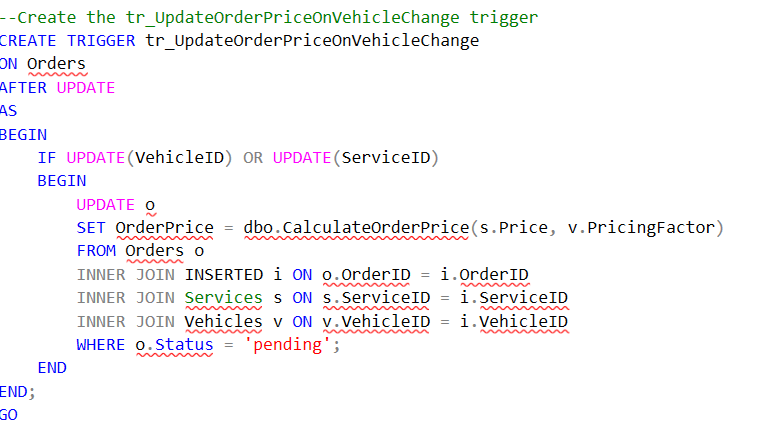
این عملیات از جدول INSERTED استفاده می کند که یک جدول ویژه در SQL Server است که کپی هایی از ردیف های آپدیت شده یا تغییر یافته را در طول عملیات به روز رسانی نگه می دارد.

2. tr\_UpdatePendingOrdersOnServicePriceChange trigger:



عملیات: این تریگر روی جدول "Services" ایجاد می شود. این تریگر طوری طراحی شده است که پس از انجام عملیات به روز رسانی در جدول "سرویس ها"، به ویژه زمانی که ستون "قیمت" به روز می شود، اجرا شود. به طور خلاصه، این تریگرهنگام به‌روزرسانی قیمت یک سرویس، «OrderPrice» را در جدول «Orders» برای سفارش‌هایی با وضعیت «در انتظار» به‌روزرسانی می‌کند. قیمت سفارش جدید با استفاده از تابع CalculateOrderPrice محاسبه می شود که قیمت خدمات جدید و فاکتور قیمت گذاری وسیله نقلیه مرتبط را در نظر می گیرد.

3. tr\_UpdateOrderPriceOnVehicleChange trigger:



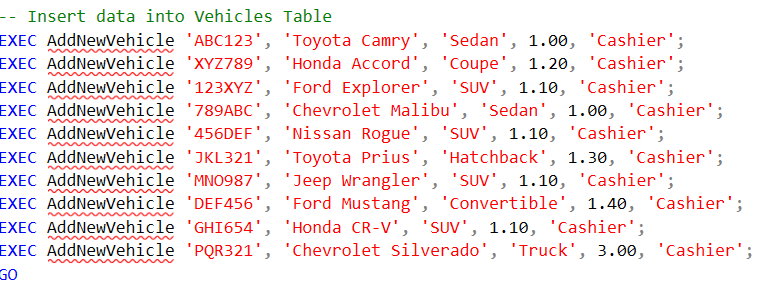
عملیات: به طور خلاصه، زمانی که ستون‌های «VehicleID» یا «ServiceID» به‌روزرسانی می‌شوند، این تریگر «OrderPrice» را در جدول «Orders» برای سفارش‌هایی با وضعیت «در انتظار» به‌روزرسانی می‌کند. قیمت سفارش جدید با استفاده از یک تابع تعریف شده محاسبه می شود که قیمت خدمات و فاکتور قیمت گذاری وسیله نقلیه مرتبط را در نظر می گیرد.

**Inserts**

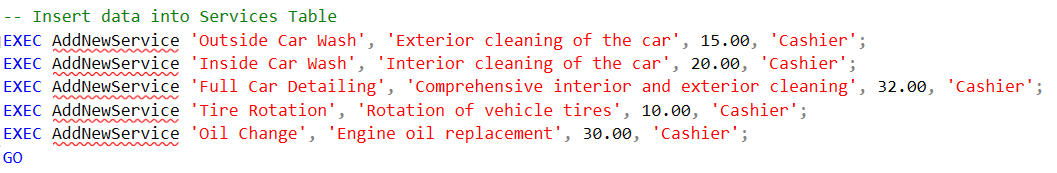
1. اضافه کردن دیتا به جدول customer:



1. اضافه کردن دیتا به جدول vehicles:



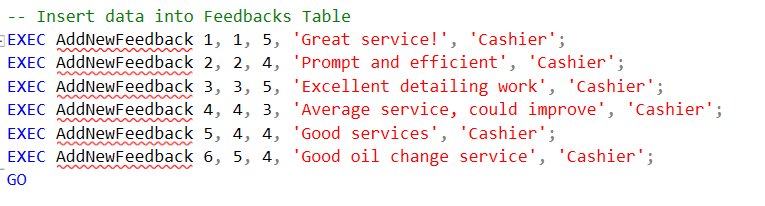
1. اضافه کردن دیتا به جدول Services:



1. اضافه کردن دیتا به جدول :Employees

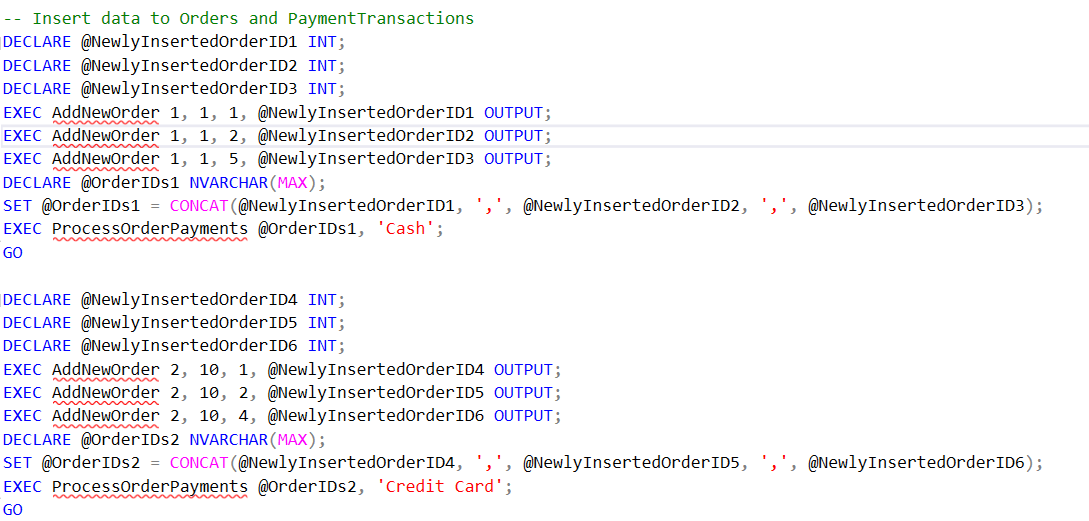


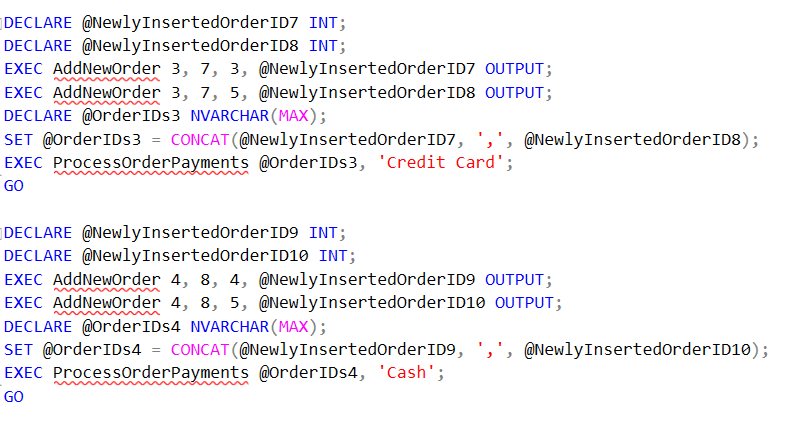
1. اضافه کردن دیتا به جدول Feedbacks:



1. اضافه کردن دیتا به جدول :orders , PaymentTransactions

عملیات: ابتدا سه خط اولیه سه متغیر را برای ذخیره مقادیر خروجی پراسیجر ذخیره شده تعریف میکنند. سپس سه خط بعدی پراسیجر ذخیره شده AddNewOrder را سه بار با پارامترهای مختلف اجرا می کنند. این پراسیجر کارش افزودن سفارشات جدید است. مثلا در اینسرت اولی پارامترها شامل 1 برای ایدی مشتری، 1 برای ایدی وسیله نقلیه، و ایدی های خدمات مختلف (1، 2، و 5) است. ایدی های سفارشات تازه درج شده در پارامترهای خروجی ثبت می شوند. سپس سه ایدی سفارشات تازه درج شده را به هم متصل کرده و در متغیر orderIDS1 ریخته میشود. سپس پراسیجر دیگری را اجرا می کند ProcessOrderPayments)) که این پراسیجر بالاتر توضیح داده شد.

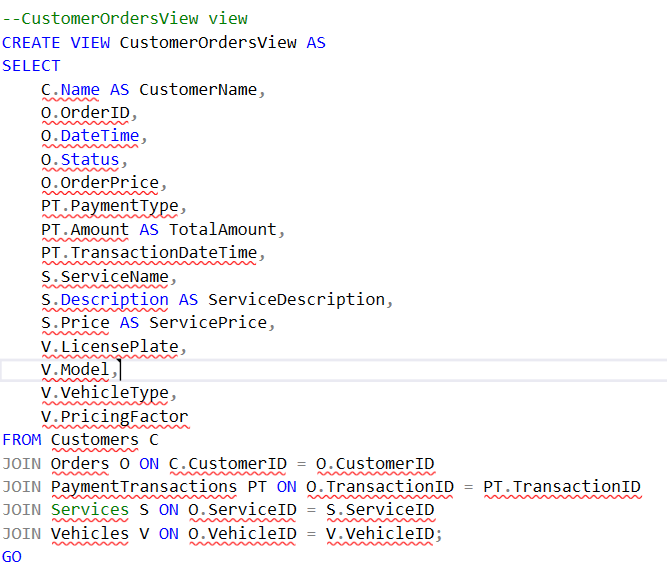




**Views**

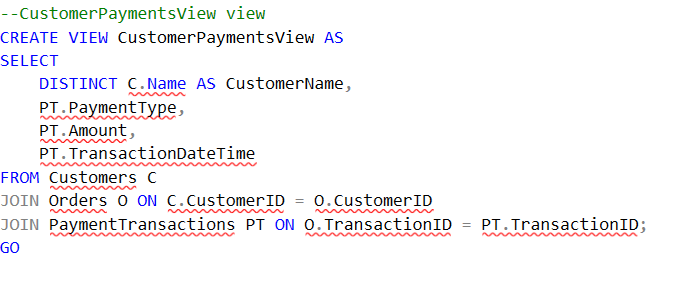
1. CustomerOrderView View:

عملیات: این ویو چندین جدول مرتبط را با یکدیگر جوین کرده تا یک نمای تلفیقی از سفارشات مشتری ارائه دهد. در واقع یک نمای کلی جامع از سفارشات مشتری، از جمله جزئیات مشتری، اطلاعات سفارش، جزئیات تراکنش پرداخت، جزئیات خدمات و جزئیات خودرو را ارائه می دهد.



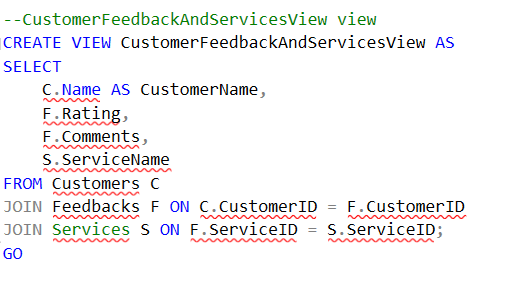
1. CustomerPaymentView View:

عملیات: این ویو با پیوستن جداول Customers، Orders و PaymentTransaction اطلاعاتی در مورد پرداخت های مشتری ارائه می دهد. به طور خلاصه، CustomerPaymentsView یک نمای تلفیقی از پرداخت های مشتری ارائه می دهد که نام مشتری را همراه با نوع پرداخت، مبلغ و تاریخ/زمان تراکنش نشان می دهد. این ویو برای حذف نام‌های تکراری مشتریان در مجموعه نتایج طراحی شده است و نمای کلی واضحی از پرداخت‌های انجام شده توسط مشتریان فردی ارائه می‌کند.



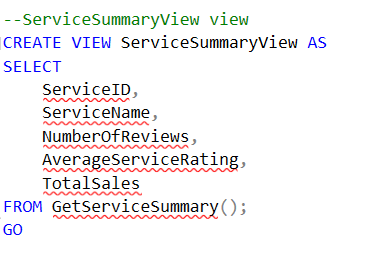
1. CustomerFeedbackAndServicesView View:

عملیات: این ویو نمایی است که اطلاعاتی درباره بازخورد مشتری و خدمات مرتبط با آن بازخورد با پیوستن جداول مشتریان، بازخوردها و خدمات ارائه می دهد. این ویو به طور خلاصه، یک نمای تلفیقی از بازخورد مشتری، از جمله نام مشتری، رتبه بندی داده شده، نظرات ارائه شده و نام سرویس مرتبط با بازخورد ارائه می دهد. این دیدگاه برای تجزیه و تحلیل و گزارش در مورد رضایت مشتری و خدمات خاصی که بازخورد دریافت کرده اند مفید است.



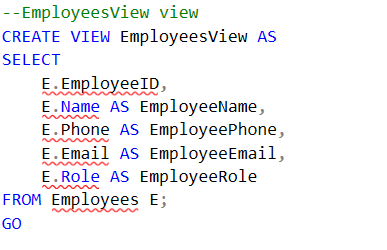
1. ServiceSummaryView View:

عملیات: این ویو خروجی های تابع GetServiceSummary که بالاتر نحوه کار آن توضیح داده شده است را برمیگرداند تا از نتایج آن استفاده شود در صورت نیاز.



1. EmployeesView View:

عملیات: این ویو ستون های مشخص و مورد نیاز از جدول کارمندان را برمیگرداند تا اطلاعات جامعی از کارمندان موجود بدست آورد.



1. DatabaseInfo View:

عملیات: این ویو اطلاعاتی در مورد آبجکت های مختلف پایگاه داده، به ویژه جداول، توابع، پراسیجرها، تریگرها و ویوها ارائه می دهد. این شامل جزئیاتی مانند نام آبجکت و نوع آن و برای جداول، تعداد ردیف است. این ویو یک نمای کلی در سطح بالا از اطلاعات ساختاری پایگاه داده و برخی از آمارهای اولیه در مورد تعداد ردیف جداول ارائه می دهد.

