Group 14

John Keller

7 December 2018

Database System Concepts

Database Systems Group 14 Project Report

**Introduction:**

For our final project in Database Systems Concepts, our group decided to model a chain of car dealerships. This model includes a table of dealerships and the employees, addresses, sales, customers, cars, and car features associated with each dealership. Our final projects allows potential users to get information about customers, get information about employees, search details about particular car makes and models, search features associated with individual cars, analyze the sales performance of each individual employee, analyze the sales performance of each individual dealership, organize employees by the employees with the greatest total sales, and organize customers by the greatest total purchases.

Our project is designed to be used by both dealership employees (both individuals involved in upper management and salespeople who wish to assist customers in purchasing a vehicle). Managers at dealerships could use our application to analyze employee or customer information to determine the most consistently high-performing employees or most consistently loyal customers so that they reward such individuals. Additionally, higher-level managers could use our application to determine the highest-performing dealerships so that they can invest more resources at these locations. A salesperson, on the other hand, could use our application for basic car-searching needs. Customers can use the application to search for cars based on some of the most commonly searched details of a car including, make, model, color, year, mileage, previous owner counts, and the current asking price of the vehicle. Once the salesperson has identified a car that suits the customer’s particular needs, the car can be purchased and the purchase is added to our database.

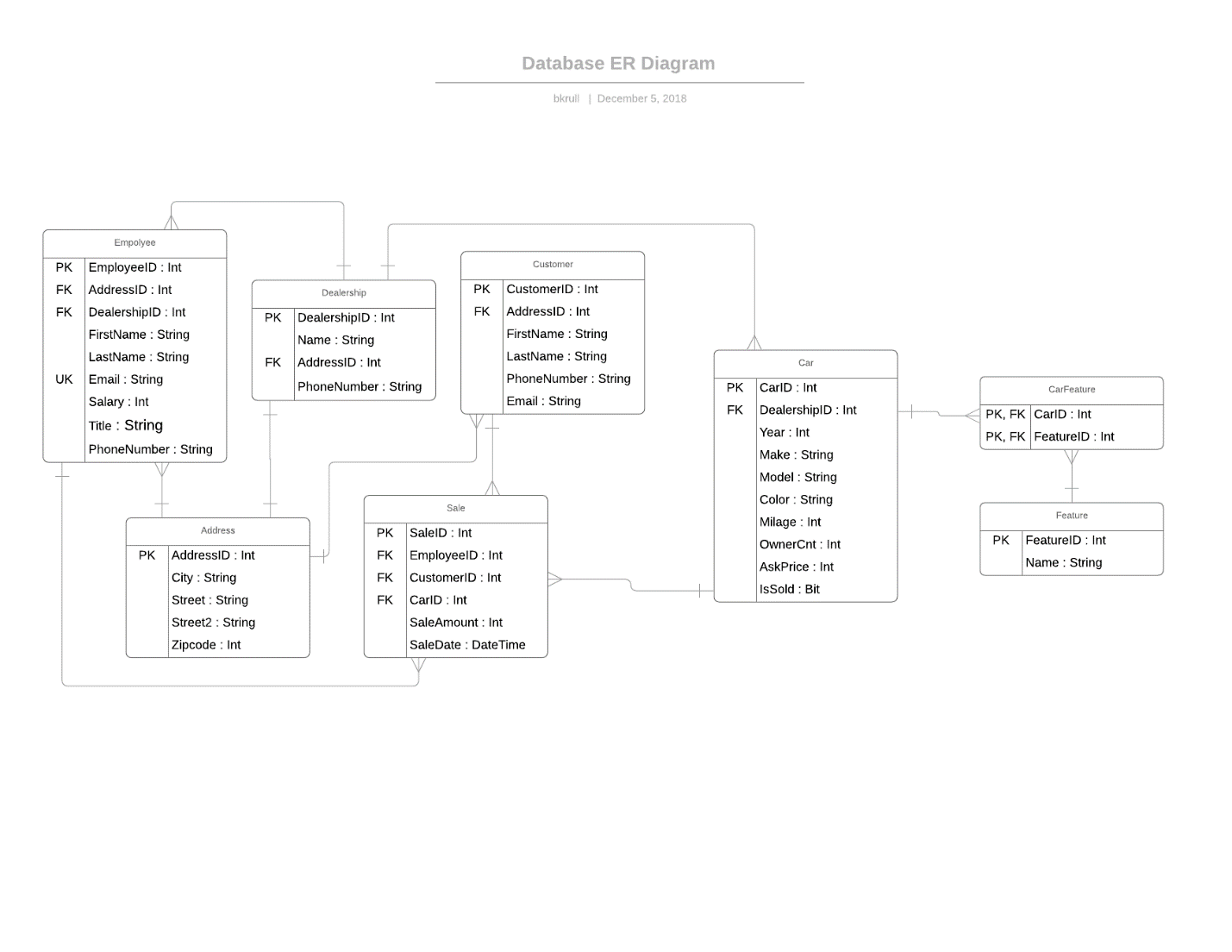
**Technical Description:**

To begin work on our project, we first created eight tables within Microsoft SQL Server Management Studio. The tables included are as follows: Demo. Feature, Demo.[Address], Demo. Dealership, Demo.Car, Demo.CarFeature, Demo.Employee, Demo.Customer, and Demo.Sale. These tables provide a foundation for our database.

Once we had created the tables for our database, we populated the table with data included in our data.sql file. The majority of the data that we created was retrieved from Mockaroo.com. With the mock data provided on this website, we were able to populate our tables with 500 customers and 500 employees along with first names, last names, various IDs, phone numbers, email addresses, employee salaries, and employee job titles that relate to each person within our database. In addition to the information that we generated for each person in the database, we generated physical address information (city, street name, and zip code) information where necessary. Finally, we generated information about each car, including year information, car makes, car models, car colors, car mileage, asking prices, and dealership IDs for the dealerships associated with each car.

Once we had populated all of the tables in our database with mock information, we began to create procedures. In our procedures.sql file, we wrote 16 SQL procedures with which we could modify and organize the information within our database. Among the functions of these procedures are the ability to gather customer purchases, get the features associated with particular cars, search for cars based upon their attributes, and gather information about the performance of dealerships.

Finally, once we had established the structure of our database and created procedures with which we could modify the data in the database, we developed the functionality of our user interface with C# code in Microsoft Visual Studio 2017. We began by creating a central Windows Form that acts as our homepage and gives users access to the various functionalities of our application. Branching off of our homepage, we included eight other windows forms that bring users to windows that allow them to search for cars, assess dealership performance, assess employee performance, search cars by feature, gather customer information, organize top employees, search employees, and make purchases. Each form includes a corresponding page of code written in C# that controls the functionality of the page. These functionalities include calling database procedures through an established connection, creating data grids, and organizing information retrieved from procedure calls so that it is more easily interpreted by users.

**Database Design: **

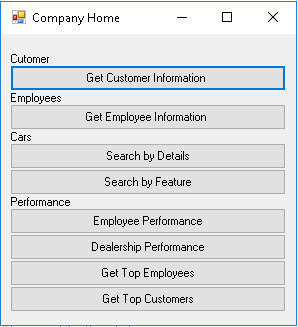
**System Design:**

Our system had many mostly independent forms that talked to each other very little. They all shared a common object, Group14Connection, which help them connect to the database and retrieve data.

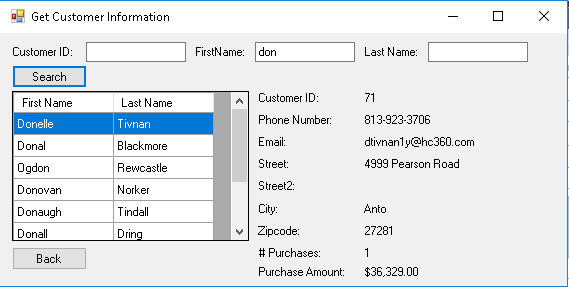
**System Features and Usage:**

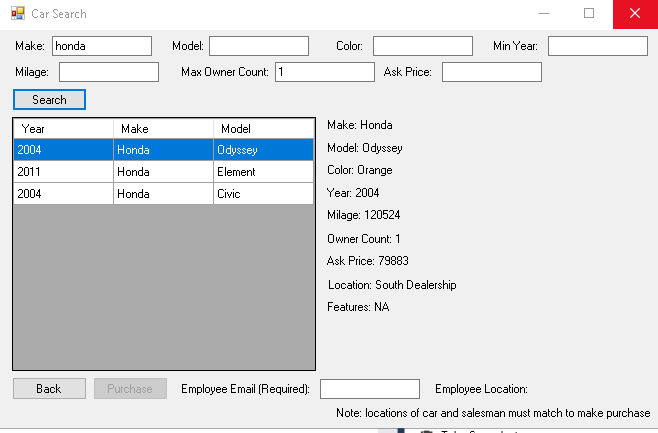
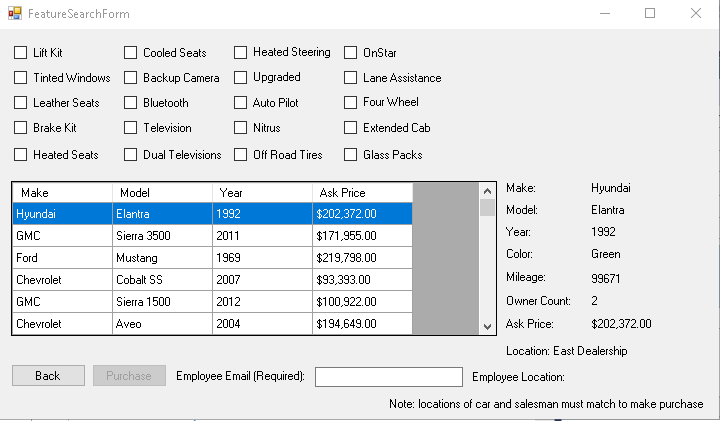
The usage of our system can be broken down into two main user interactions: salesperson interactions and managerial interactions.

*Salesperson Interactions:*

Our system is ideal for salespeople in situations where a salesperson would like to help a customer find a car based upon specific car features and attributes. When opening the application, both salespeople and managers are greeted by the home screen, which allows users to access all of the application’s functions. Our current design is flawed because it allows salespeople to access functions that only mangers should be able to access, but we should operate under the assumption that salespeople are only using three functions to assist a customer: get customer information, search by details, and search by feature.

Using “Get Customer Information” could be useful when a salesperson needs personal information about a customer who has already been involved with the car dealership in the past. Instead of requesting all of the customer’s information, the salesperson may simply request the first or last name of the customer and once identified in the database, all of the customer’s other information will be filled in automatically.

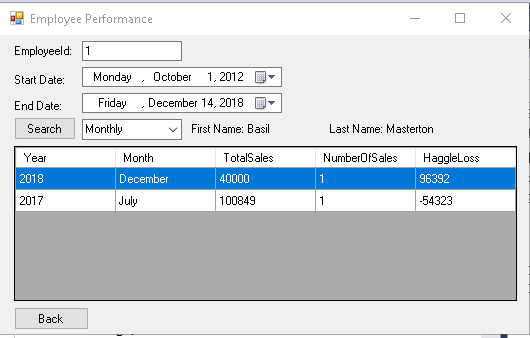


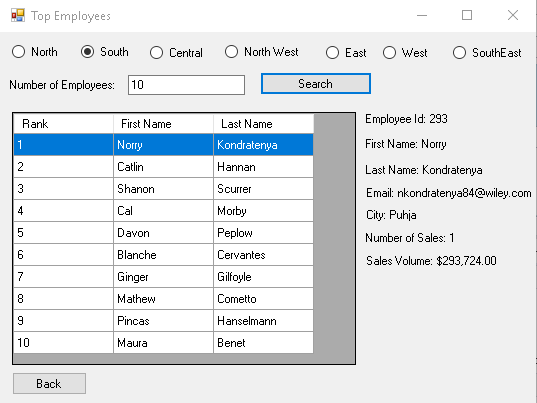
When trying to locate a suitable vehicle for a customer’s needs, the salesperson may either search cars by details or by features. The customer can specify information about the make, model, color, milage, and asking price of a vehicle, and our application will return a table that lists all of the results that match the customer’s request. Once a car is chosen from the table, additional details about the car will appear on the right hand side of the application interface. Alternatively, customers may request cars based on the features that they have. Customers may choose from a list of twenty features, and all cars that have the features selected will be returned in a table. Just like in the search by details function, information about a selected car will appear on the right hand side of the application. If a suitable car is located, the salesperson may make a purchase for the customer, and the car will be removed from the list of cars available in the chain of dealerships.

*Manager Interactions*

Managers can use our application to get a better understanding of the performance of employees and dealerships. They can also identify repeat customers to ensure that the needs of such customers are being met.

Manages may first wish to determine the performance of individuals within the company so that these individuals may be recognized for their services to the company. To determine individual employee performance, a manager could choose “Employee Performance” from the home screen. The manager can then select the range of dates over which he or she wishes to analyze employee performance, and can determine whether they would like a monthly or yearly view of performance data. Once the terms of the search have been selected, performance data such as the number of sales made, the total amount of sales, and the haggle loss of particular sales is displayed.

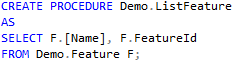
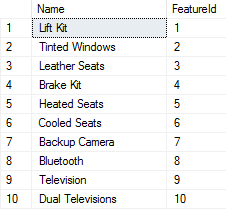
The “Dealership Performance” form takes the same constraints and behaves similarly to the “Employee Performance” form, the major difference being that sales numbers are calculated by dealership instead of by individual.

Finally, the “Get Top...” functions can be used by managers to determine major employees and customers so that special focus can be placed upon these individuals. “Get Top Employees” allows managers to determine the number of top performing individuals that they would like to observe from a particular dealership location. A search for the top 10 employees, for example, would return a table containing ten employees with the highest volume of sales. Details about these individuals will appear on the right hand side of the application when a name is selected, and managers can clearly and easily locate their top employees.

Another similar form is available to find “Top Customers.” Managers may use this function if they wish to identify particular consistent customers so that the dealership may continue to appeal to these customers in the future.

**Queries and Reports:**

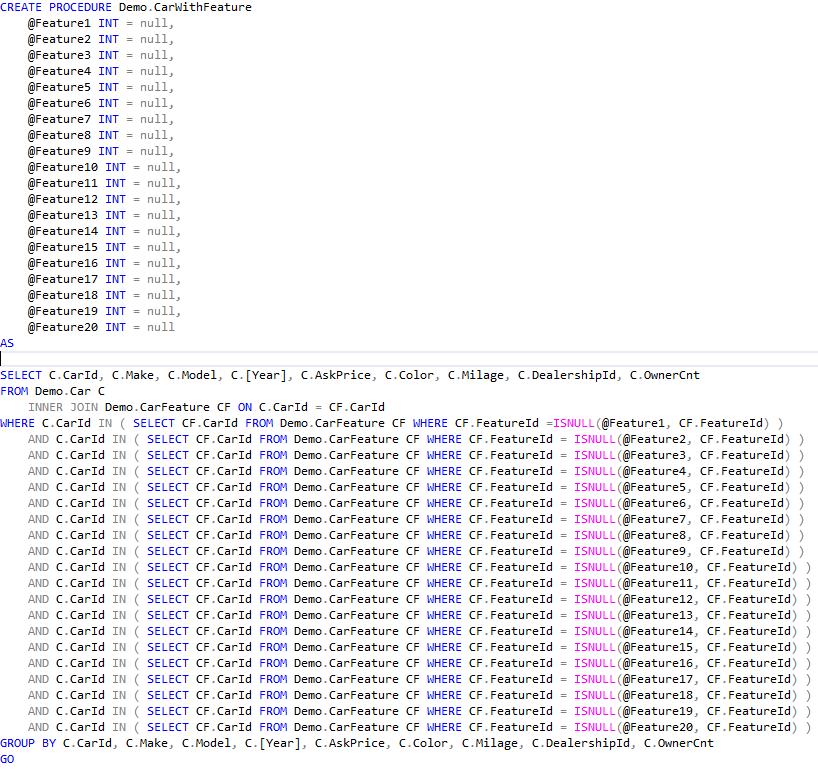
Demo.ListFeature is a question-type query. This procedure retrieves the names of all features so that they can be listed in the FeatureSearchForm.



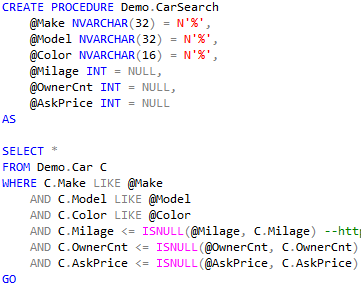
Demo.CarWithFeatures is a question-type query. This procedure allows users to lookup cars based on which features they have.



This table shows the results for determining cars that have feature 19 and feature 20.

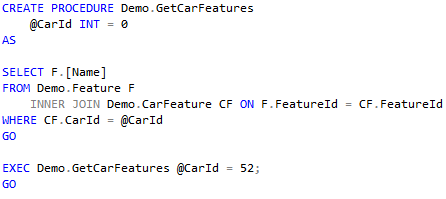


Demo.CarSearch is a question-type query. This procedure allows user to lookup cars based on features that the user specifies.

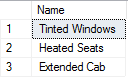


This table shows the result of the query with @make set to Ford and @Color set to Teal.

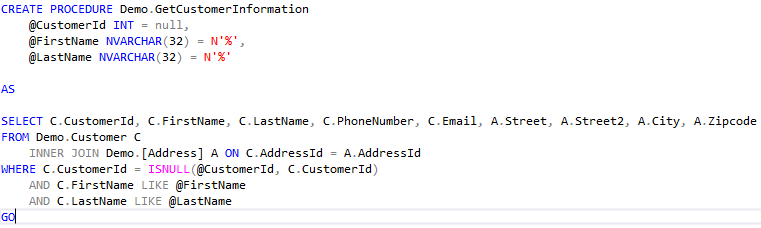
Demo.GetCarFeatures is a question-type query. This procedure allows the user to find a particular car’s features.



This table shows the features belonging to the car with ID = 52.

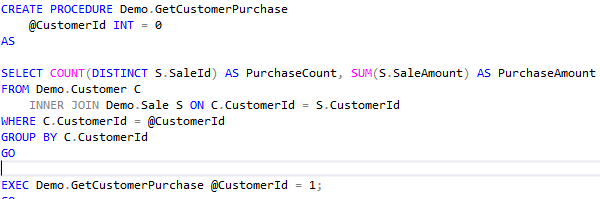


Demo.GetCustomerInformation is a question-type query. This procedure allows the user to search a customer and proceeds to retrieve all information about a customer.



This table shows customer information based on searching customerId =1.

Demo.GetCustomerPurchase is a question-type query. This retrieves a customer’s purchase count and purchase amount based upon a customer ID.

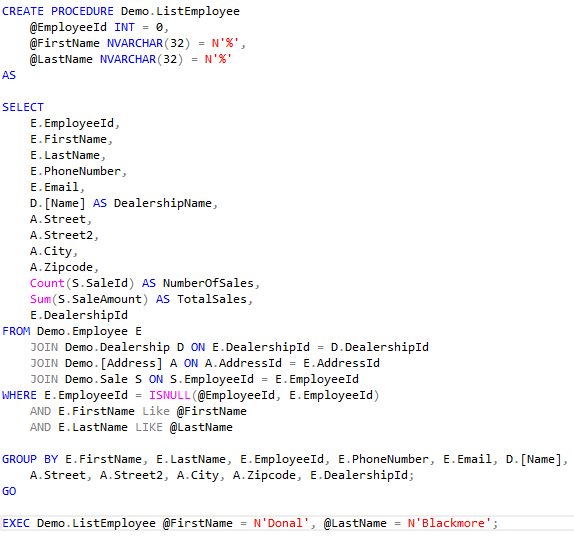


This table shows purchase information based on searching customerId =1.



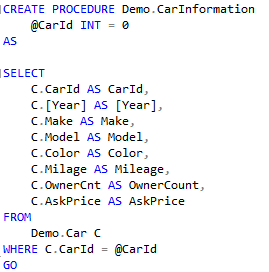
Demo.ListEmployee is a question-type query. This procedure allows the user to search an employee and provides information about that employee.

This table shows the employee information of EmployeeId = 2.

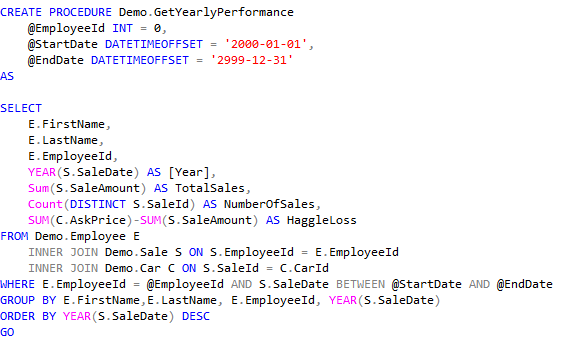


Demo.CarInformation is a question type query. This procedure allows the user to search for the details associated with a car.

This table shows the car information of CarId = 2.

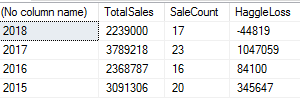
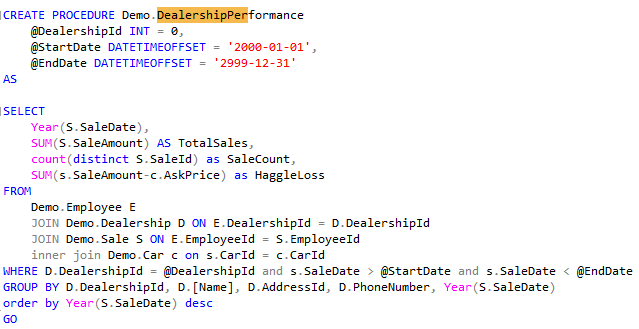


Demo.GetYearlyPerformance is a report type query. This procedure combines all the sales an employee has made in a data range and returns the yearly employee information of sales count, sales amount, and money lost on Ask vs Sale price. (Similar query for monthly results)



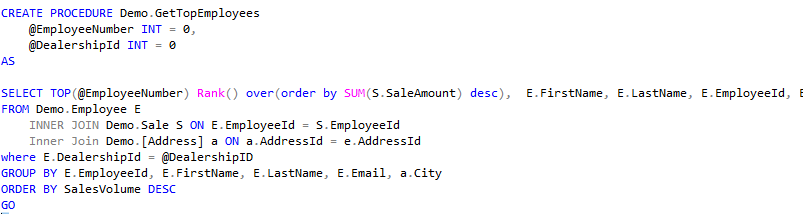
This table shows the employee performance of EmployeeId = 2, 2000-2018.

Demo.DealershipPerformance is a report type query. This procedure combines all the sales a dealership has made in a date range and returns the yearly information of sales count, sales amount, and money lost on Ask vs Sale price. (Similar query for monthly results)



This table shows the dealership performance of Dealerhip = 2, 2000-2018.

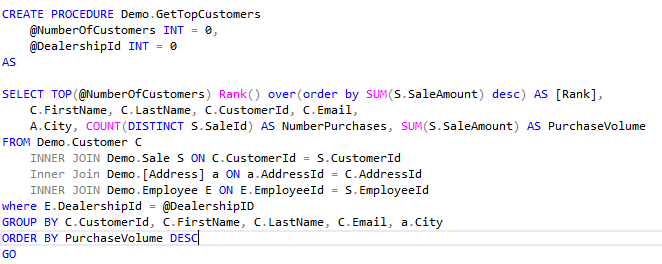
Demo.GetTopEmployees is a report type query. This procedure combines all the sales of employees at a dealership and returns the top X employees.



This table shows the top 3 employees from dealership 2.

Demo.GetTopCustomers is a report type query. This procedure combines all of the purchases of customers at a dealership and returns the top X employees.

**Summary:**



This table shows the top 3 customers from dealership 1.

We had originally planned on using doing a webpage using react with about 4 main pages, however our end result was a C# project with 8 (or 9 with purchasing) forms. We did not change much from our original database design: we added a few members to a couple tables and removed one foreign key from the sales table. We learned how to integrate C# and an SQL database. Given another go at this project we would start earlier and spend more time designing our user interface. Our user interface is functionally complete, but is a bit clunky in some cases, ideally with future work we would improve upon this and possibly add more features.