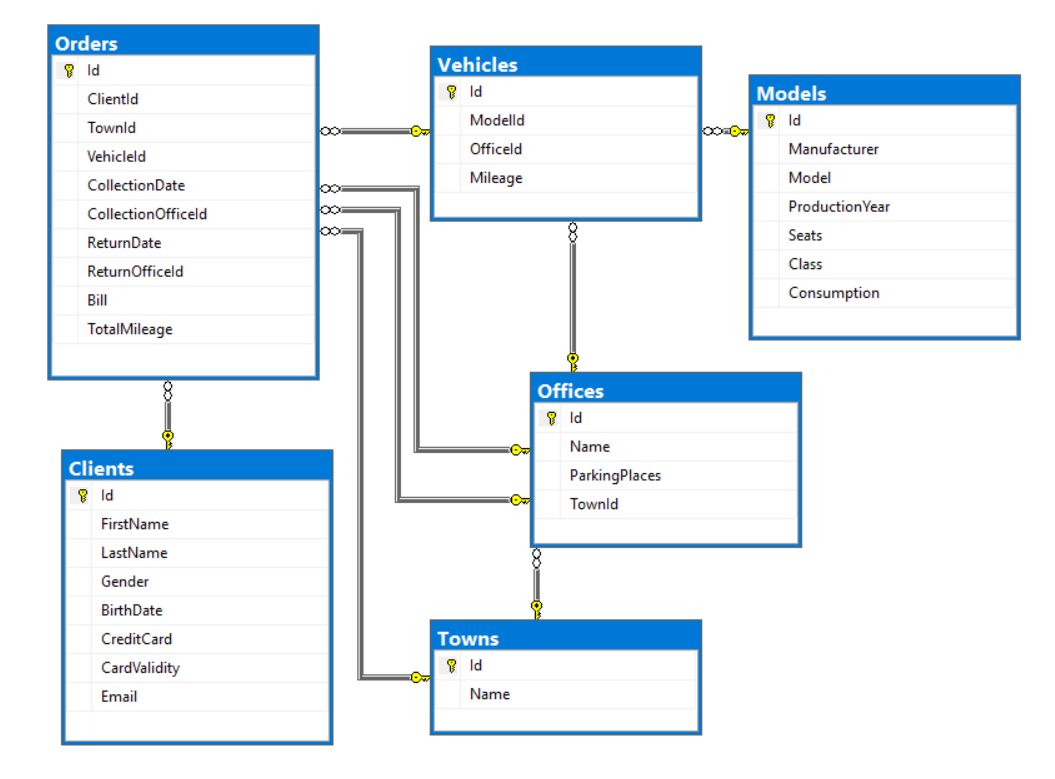
# Database Basics MS SQL Exam – 10 Dec 2019

Exam problems for the ["Database Basics" course @ SoftUni](https://softuni.bg/courses/databases-basics-ms-sql-server)

# Car Showroom

# Section 1. DDL (30 pts)

You have been given the E/R Diagram of the Report Service:



Crеate a database called **CarShowroom**. You need to create **6 tables**:

* **Clients** – contains information about the people who get cars on rent
* **Orders** - contains information about the client’s orders
* **Towns** – contains information about the towns
* **Offices** - contains information about the offices in the different towns
* **Vehicles** – contains information about the vehicles which could be rented
* **Models** – contains information about the

**Clients**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| FirstName | **String** up to 30 symbols, Unicode | **NULL** is **not** allowed |
| LastName | **String** up to 30 symbols, Unicode | **NULL** is **not** allowed |
| Gender | **Character** with **exactly** **1** symbol | Could be: '**M**' or '**F'** |
| BirthDate | **DateTime** |  |
| CreditCard | **String** up to 30 symbols, Unicode | **NULL** is **not** allowed |
| CardValidity | **DateTime** |  |
| Email | **String** up to **50** symbols, Unicode | **NULL** is **not** allowed |

**Towns**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| Name | **String** up to **50** symbols, Unicode | **NULL** is **not** allowed |

**Offices**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| Name | **String** up to **40** symbols, Unicode |  |
| ParkingPlaces | **Integer** from **0** to **2,147,483,647** |  |
| TownId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Towns |

**Models**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| Manufacturer | **String** up to **50** symbols, Unicode | **NULL** is **not** allowed |
| Model | **String** up to **50** symbols, Unicode | **NULL** is **not** allowed |
| ProductionYear | **DateTime** |  |
| Seats | **Integer** from **0** to **2,147,483,647** |  |
| Class | **String** up to **10** symbols, Unicode |  |
| Consumption | **Decimal** up to **14** digitswith **2** digits **precision** after the decimal point |  |

**Vehicles**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| ModelId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Models |
| OfficeId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Offices |
| Mileage | **Integer** from **0** to **2,147,483,647** |  |

**Orders**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| ClientId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Clients |
| TownId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Towns |
| VehicleId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Vehicles |
| CollectionDate | DateTime | **NULL** is **not** allowed |
| CollectionOfficeId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Offices |
| ReturnDate | DateTime | Relationship with table Offices |
| ReturnOfficeId | **Integer** from **0** to **2,147,483,647** | Relationship with table Offices |
| Bill | **Decimal** up to **14** digitswith **2** digits **precision** after the decimal point |  |
| TotalMileage | **Integer** from **0** to **2,147,483,647** |  |

## Database design

Submit all of yours **create** **statements** to Judge (**only creation of tables**).

# Section 2. DML (10 pts)

**Before you start you have to import "Dataset-CarShowroom.sql". If you have created the structure correctly the data should be successfully inserted.**

In this section, you have to do some data manipulations:

## Insert

Let's **insert** some sample data into the database. Write a query to add the following records into the corresponding tables. All Id's should be auto-generated.

**Models**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Manufacturer** | **Model** | **ProductionYear** | **Seats** | **Class** | **Consumption** |
| Chevrolet | Astro | 2005-07-27 00:00:00.000 | 4 | Economy | 12.60 |
| Toyota | Solara | 2009-10-15 00:00:00.000 | 7 | Family | 13.80 |
| Volvo | S40 | 2010-10-12 00:00:00.000 | 3 | Average | 11.30 |
| Suzuki | Swift | 2000-02-03 00:00:00.000 | 7 | Economy | 16.20 |

**Orders**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Client Id** | **Town Id** | **Vehicle Id** | **Collection Date** | **Collection Office Id** | **Return Date** | **Return Office Id** | **Bill** | **Total Mileage** |
| 17 | 2 | 52 | 2017-08-08 | 30 | 2017-09-04 | 42 | 2360.00 | 7434 |
| 78 | 17 | 50 | 2017-04-22 | 10 | 2017-05-09 | 12 | 2326.00 | 7326 |
| 27 | 13 | 28 | 2017-04-25 | 21 | 2017-05-09 | 34 | 597.00 | 1880 |

## Update

Set **all Model's** **class** to "**Luxury"** where the consumption is over **20**.

## Delete

Delete **all orders** which don't have a **Return Date**.

# Section 3. Querying (40 pts)

**You need to start with a fresh dataset, so recreate your DB and import the sample data again (Dataset-CarShowroom.sql).**

## My Gen

Find all **clients** who are born between 1977 and 1994. **Order** the clients by **First Name** and then by **Last Name** in **ascending** order, and finally by **Id** (**ascending**).

Required columns:

* First Name
* Last Name

### Example:

|  |  |
| --- | --- |
| **FirstName** | **LastName** |
| Agretha | Bumphries |
| Alene | Bernocchi |
| Alicea | Elphinston |
| … | … |

## Office

Select **all offices** which have a **parking lot** with **more than 25** places**.** Order them by **their Town's name** (ascending) and **then** by **Office Id** (ascending).

Required columns:

* TownName
* OfficeName
* ParkingPlaces

### Example:

|  |  |  |
| --- | --- | --- |
| **TownName** | **OfficeName** | **ParkingPlaces** |
| Betrr | Robel, Krajcik and Olson | 26 |
| Czarna Dabrówka | Cassin, Heathcote and Kuhlman | 26 |
| Gaohong | Champlin Inc | 26 |
| … | … | … |

## Vehicles

Show all **available** vehicles. (A vehicle is not available if it is **reserved** **for** an **order** **and** is **not** **turned** **back** yet)

Required columns:

* Model
* Seats
* Mileage

Order the results by **Mileage** (**ascending**), then by the Model’s **number of seats** (**descending**) and finally by **Model Id** (**ascending**).

### Example:

|  |  |  |
| --- | --- | --- |
| **Model** | **Seats** | **Milage** |
| Eldorado | 3 | 23188 |
| J | 9 | 29281 |
| Swift | 7 | 29940 |
| … | … | … |

## Offices per Town

Select all towns and show the **total number** of **offices** per **each** **town**.

Required columns:

* TownName
* OfficesNumber

Order the results by **OfficesNumber** **descending** and then by **TownName** **ascending**.

### Example:

|  |  |
| --- | --- |
| **TownName** | **OfficesNumber** |
| La Escondida | 5 |
| Budapest | 4 |
| Gaohong | 4 |
| … | … |

## Best Choice

Select **all** **vehicle models** and show **how many** **times** each of them have been ordered.

Required columns:

* Manufacturer
* Model
* TimesOrdered

Order by total **TimesOrdered** **descending,** then by **Manufacturer** **descending and** then by **Model** (**ascending**).

### Example:

|  |  |  |
| --- | --- | --- |
| **Manufacturer** | **Model** | **TimesOrdered** |
| Kia | Forte | 15 |
| Chevrolet | Astro | 12 |
| Toyota | Solara | 11 |
| … | … | … |

## Kinda Person

Select the **clients** who **have placed** an **order** and print their **most frequent** choice of **vehicle's class**. If a client's most frequent choice is **equally spread** over **different** vehicle **classes** show **all the choices** on **separate** lines.

Required columns:

* Names - Clients first and last name separated by space
* Class - Most frequent class choice

Order them by **client's Names**, then by **Class** and **then** by **Client Id** (all in **ascending** order).

### Example:

|  |  |
| --- | --- |
| **Names** | **Class** |
| Agnella Adamiec | Family |
| Alayne Morville | Average |
| Albie Pinshon | Economy |
| Albie Pinshon | Luxury |
| … | … |

|  |  |
| --- | --- |
| Volkswagen | 14.900000 |

|  |  |
| --- | --- |
| … |  |

# Section 4. Programmability (20 pts)

## Find My Custom Ride

Create a **user defined function** with the name **udf\_CheckForVehicle(@townName, @seatsNumber)** that receives a **town's name** and a **seats number** and checks if there is **any vehicle** with the **given seats** at an **office** of the **given** **town**.

* If **there is** a **vehicle** print the **output** in the following **format**: "**OfficeName – Model**".
* If there is no vehicle found print the following message: "NO SUCH VEHICLE FOUND"
* If there is **more than one** vehicle available **order** the results by **office name** **ascending** and **return** the **first** **one**

Parameters:

* Town's name
* Seats number

### Example usage:

|  |  |
| --- | --- |
| **Query** | |
| **SELECT** **dbo.udf\_ CheckForVehicle ('La Escondida', 9)** | |
| **Result** | |
| Green, Jaskolski and Blick - TL |

## Move a Vehicle

Create a **user defined stored procedure** with the name **usp\_MoveVehicle(@vehicleId, @officeId)** that receives a **vehicle's Id** and an **office’s** **Id** and changes the vehicle's **OfficeId** to the given value **only if** there are **free** **ParkingPlaces** in the **given office**. If the move is not successful **rollback** any changes and throw an **exception** with message: "Not enough room in this office!"

Parameters:

* Vehicle's Id
* Office's Id

### Example usage:

|  |
| --- |
| **Query** |
| **EXEC usp\_MoveVehicle 7, 32;**  **SELECT OfficeId FROM Vehicles WHERE Id = 7** |
| **Response** |
| 32 |