# Database Basics MS SQL Regular Exam – 05 Dec 2023

Exam problems for the ["Database Basics" course @ SoftUni](https://softuni.bg/trainings/4182/ms-sql-september-2023).  
Submit your solutions in the SoftUni Judge system at [Judge](https://judge.softuni.org/Contests/4483/Databases-MSSQL-Server-Retake-Exam-05-December-2023).

# Railways Database

# Section 1. DDL (30 pts)

You have been given the E/R Diagram of the **RailwaysDb** database.



Create a database called **RailwaysDb**. You need to create **7 tables**:

* **Passengers** – contains information about the names of the passengers;
* **Towns** – contains information about the names of the towns;
* **RailwayStations** – holds data about railway stations, such as their names and associated town IDs, indicating the towns where these stations are located.;
* **Trains**– details about trains, including departure and arrival times and their corresponding town IDs;
* **TrainsRailwayStations** - Manages the many-to-many relationship between trains and railway stations, indicating which trains stop at which stations;
* **MaintenanceRecords** – records maintenance activities for trains, including dates and detailed descriptions of maintenance work;
* **Tickets** – contains information about tickets, including price, departure and arrival date, and associated train and passenger IDs;

**NOTE: Keep in mind that Judge doesn't accept "ALTER" statement and square brackets naming (when the names are not keywords).**

**NOTE: Please keep in mind that in case you have to work with a date, you have to use the exact same data type, described in the models tables. If you don't use the correct type, the Judge system won't accept your submission as correct.**

You have been tasked to **create the tables in the database by the following models**:

### ****Passengers****

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | PK, Unique table identification, Identity |
| **Name** | **String** up to **80** symbols, Unicode | **Null** is **not** allowed |

### Towns

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | PK, Unique table identification, Identity |
| **Name** | **String** up to **30** symbols | **Null** is **not** allowed |

### RailwayStations

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | PK, Unique table identification, Identity |
| **Name** | **String** up to **50** symbols | **Null** is **not** allowed |
| **TownId** | **Integer** from **0** to **2,147,483,647** | **Relationship** with table **Towns**, **Null** is **not** allowed |

### ****Trains****

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | PK, Unique table identification, Identity |
| **HourOfDeparture** | **String** up to **5** symbols | **Null** is **not** allowed |
| **HourOfArrival** | **String** up to **5** symbols | **Null** is **not** allowed |
| **DepartureTownId** | **Integer** from **0** to **2,147,483,647** | **Relationship** with table **Towns**, **Null** is **not** allowed |
| **ArrivalTownId** | **Integer** from **0** to **2,147,483,647** | **Relationship** with table **Towns**, **Null** is **not** allowed |

### TrainsRailwayStations

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **TrainId** | **Integer** from **0** to **2,147,483,647** | PK, Unique table identification, Relationship with table **Trains**, **Null** is **not** allowed |
| **RailwayStationId** | **Integer** from **0** to **2,147,483,647** | PK, Unique table identification, Relationship with table **RailwayStations**, **Null** is **not** allowed |

### ****MaintenanceRecords****

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | PK, Unique table identification, Identity |
| **DateOfMaintenance** | **Date** | **Null** is **not** allowed |
| **Details** | **String** up to **2000** symbols | **Null** is **not** allowed |
| **TrainId** | **Integer** from **0** to **2,147,483,647** | Relationship with table **Trains**, **Null** is **not** allowed |

### ****Tickets****

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | PK, Unique table identification, Identity |
| **Price** | **Decimal** number with two-digit precision | **Null** is **not** allowed |
| **DateOfDeparture** | **Date** | **Null** is **not** allowed |
| **DateOfArrival** | **Date** | **Null** is **not** allowed |
| **TrainId** | **Integer** from **0** to **2,147,483,647** | Relationship with table **Trains**, **Null** is **not** allowed |
| **PassengerId** | **Integer** from **0** to **2,147,483,647** | Relationship with table **Passengers**, **Null** is **not** allowed |

## Database design

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

# Section 2. DML (10 pts)

**Before you start, you have to import "Dataset.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 IDs (Primary Keys) should be **auto-generated**.

### ****Trains****

|  |  |  |  |
| --- | --- | --- | --- |
| **HourOfDeparture** | **HourOfArrival** | **DepartureTownId** | **ArrivalTownId** |
| '07:00' | '19:00' | 1 | 3 |
| '08:30' | '20:30' | 5 | 6 |
| '09:00' | '21:00' | 4 | 8 |
| '06:45' | '03:55' | 27 | 7 |
| '10:15' | '12:15' | 15 | 5 |

### ****TrainsRailwayStations****

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TrainId** | **RailwayStationId** | **TrainId** | **RailwayStationId** | **TrainId** | **RailwayStationId** |
| 36 | 1 | 37 | 60 | 39 | 3 |
| 36 | 4 | 37 | 16 | 39 | 31 |
| 36 | 31 | 38 | 10 | 39 | 19 |
| 36 | 57 | 38 | 50 | 40 | 41 |
| 36 | 7 | 38 | 52 | 40 | 7 |
| 37 | 13 | 38 | 22 | 40 | 52 |
| 37 | 54 | 39 | 68 | 40 | 13 |

### ****Tickets****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Price** | **DateOfDeparture** | **DateOfArrival** | **TrainId** | **PassengerId** |
| 90.00 | '2023-12-01' | '2023-12-01' | 36 | 1 |
| 115.00 | '2023-08-02' | '2023-08-02' | 37 | 2 |
| 160.00 | '2023-08-03' | '2023-08-03' | 38 | 3 |
| 255.00 | '2023-09-01' | '2023-09-02' | 39 | 21 |
| 95.00 | '2023-09-02' | '2023-09-03' | 40 | 22 |

## Update

Due to technical reasons, every ticket with a **DateOfDeparture after October 31st** will be postponed with one week. That means that **both** **DateOfDeparture** and **DateOfArrival** should be changed for **7 days later**.

## Delete

In table **Trains**, delete the **train, that departures from Berlin**. Keep in mind that there could be **foreign key constraint conflicts**.

# Section 3. Querying (40 pts)

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

## Tickets by Price and Date of Departure

Select all **tickets,** ordered by **price** (ascending), then by **departure date** (descending).

Required columns:

* **DateOfDeparture**
* **TicketPrice**

### Example

|  |  |
| --- | --- |
| **DateOfDeparture** | **TicketPrice** |
| 2023-11-06 | 45.00 |
| 2023-10-20 | 45.00 |
| 2023-10-08 | 45.00 |
| 2023-09-11 | 45.00 |
| 2023-09-02 | 45.00 |
| 2023-11-07 | 50.00 |
| … | … |

## Passengers with their Tickets

Select all the tickets purchased, along with the **names of the passengers** who purchased them. For the tickets you will need information for the **price**, **date of departure**, related **train’s id**. The report should be organized in a way that lists the tickets **starting from the highest price to the lowest**. In case of **identical ticket prices**, further **order the entries alphabetically by the passenger's name**.

Required columns:

* **PassengerName**
* **TicketPrice**
* **DateOfDeparture**
* **TrainID**

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **PassengerName** | **TicketPrice** | **DateOfDeparture** | **TrainID** |
| Mary Campbell | 275.00 | 2023-09-24 | 29 |
| Wayne Richardson | 275.00 | 2023-10-29 | 29 |
| Bruce Howard | 180.00 | 2023-11-14 | 10 |
| Matthew Allen | 180.00 | 2023-09-05 | 10 |
| ... | ... | … | … |

## Railway Stations without Passing Trains

Select all railway stations that **do not have any trains scheduled to stop or pass** through them. Each station must be associated **with the town** it is located in. The **town's name should be included** in your result set to understand the geographical distribution of these inactive stations. The results should be **ordered by the name of the town** in **ascending** order, **then by the name of the railway station** in **ascending** order.

Required columns:

* **Town**
* **RailwayStation**

### Example

|  |  |
| --- | --- |
| **Town** | **RailwayStation** |
| Amsterdam | Amstel |
| Amsterdam | Sloterdijk |
| Athens | Larissa Station |
| Barcelona | Passeig de Garcia |
| … | … |

## First 3 Trains Between 8:00 and 8:59

Select the **top 3 trains** departing **between 8:00 and 08:59** with **ticket** **prices above €50.00** in the RailwaysDb. Your query should join trains with **arrival town names**, ordering the results by **ticket price in ascending order**. The output should include **TrainId**, **HourOfDeparture**, **TicketPrice**, and **Destination**. Keep in mind that you cannot compare VARCHAR data, so you will have to approach differently.

Required columns:

* **TrainId**
* **HourOfDeparture**
* **TicketPrice**
* **Destination**

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **TrainId** | **HourOfDeparture** | **TicketPrice** | **Destination** |
| 1 | 08:00 | 55.00 | Paris |
| 17 | 08:00 | 60.00 | Zurich |
| 17 | 08:00 | 85.00 | Zurich |

## Count of Passengers Paid More Than Average With Arrival Towns

The **average price** of all tickets is **€76.99.** Select all passengers who have paid **more than the average price of all tickets**. Your query should generate a table grouped by the **name of arrival town, showing the count of passengers arriving in each town**. **Order** the results **by town name**.

Required columns:

* **TownName**
* **PassengersCount**

### Example

|  |  |
| --- | --- |
| **TownName** | **PassengersCount** |
| Barcelona | 1 |
| Brussels | 1 |
| Budapest | 2 |
| Gdansk | 1 |
| Lisbon | 3 |
| London | 4 |
| … | … |

## Maintenance Inspection with Town

Your task is to generate a table report for train maintenance related to inspections. This report should include the **train ID**, the **name of the town where the train departs from** and the **maintenance record details,** that **include** the word **"inspection"**. **Order** the records **by train ID**.

Required columns:

* **TrainID**
* **DepartureTown**
* **Details**

### Example

|  |  |  |
| --- | --- | --- |
| **TrainID** | **DepartureTown** | **Details** |
| 1 | London | Brake system inspection and overhaul |
| 2 | London | Brake system inspection and overhaul |
| 4 | Paris | Wheel and axle inspection |
| … | … |  |

# Section 4. Programmability (20 pts)

## Towns with Trains

Create a user-defined function, named **udf\_TownsWithTrains(@name)** that receives a town’s name.

The function should return the **total number of trains** that departures or arrives at that town.

### Example

|  |
| --- |
| **Query** |
| **SELECT dbo.udf\_TownsWithTrains('Paris')** |
| **Output** |
| **7** |

## Search Passenger Travelling to Specific Town

Create a **stored procedure**, named **usp\_SearchByTown(@townName)** that receives a town name. The procedure must print full information about all passengers that are travelling to the town with the given **townName.** Order them by **DateOfDeparture** (descending) and **PassengerName**(ascending).

Required columns:

* **PassengerName**
* **DateOfDeparture**
* **HourOfDeparture**

### Example

|  |  |  |
| --- | --- | --- |
| **Query** | | |
| **EXEC usp\_SearchByTown 'Berlin'** | | |
| **Output** | | |
| **PassengerName** | **DateOfDeparture** | **HourOfDeparture** |
| Beverly Ross | 2023-10-30 | 12:00 |
| Kenneth Parker | 2023-09-25 | 12:00 |