# Database Basics MS SQL Exam – 21 June 2020

Submit your solutions in the SoftUni Judge system at [Judge](https://judge.softuni.org/Contests/2479/Databases-MSSQL-Server-Exam-21-Jun-2020)

# Trip Service

You've been an intern at **Krivodol Trip Service LLC** ever since you finished high school. The Krivodol Trip Service doesn't really pay much, but it's the only trip company within a 50km radius in northwestern Bulgaria.

You've recently been appointed as **Chief Database Engineer**. The Chief Database Engineer's job is to keep track of every single city, account, trip, hotel and hotel room – all on a giant ledger (paper, not blockchain), which has been passed down for the last 3 generations.

Word around the office is that the company is going to hire a few programmers to try and automate the entire process. As you know, the holidays are coming up, and of course, you want to go to the overpopulated, overpriced and overbuilt seaside just to post a couple of photos of your rakia-hardened beach body on your Instagram. So, you decided to **design a relational database** in **SQL Server** and let the new code monkeys take care of everything else.

# Section 1. DDL (30 pts)

You are given an E/R Diagram of the Trip Service:



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

* Cities – contains information about cities and their countries;
* Hotels – contains information about the hotels in the system;
* Rooms– contains information about the rooms each hotel has;
* Trips– contains information about each trip;
* Accounts – contains information about the trip service users;
* AccountsTrips– contains information about all **accounts** and their **trips**.

**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. For example, data type Date means that you have to use Date, DateTime means that you have to use DateTime. If you don't use the correct type, the Judge system won't accept your submission as correct.**

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

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

### Cities

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| **Name** | **String** up to **20** symbols, **Unicode** | **NULL** is **not** allowed |
| **CountryCode** | **String** with **exactly 2** symbols | **NULL** is **not** allowed |

### Hotels

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| **Name** | **String** up to **20** symbols, **Unicode** | **NULL** is **not** allowed |
| **CityId** | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Cities** |
| **EmployeeCount** | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed |
| **BaseRate** | **Decimal** number with **two-digit** precision |  |

### Rooms

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| **Price** | **Decimal** number with **two-digit** precision | **NULL** is **not** allowed |
| **Type** | **String** up to **20** symbols, **Unicode** | **NULL** is **not** allowed |
| **Beds** | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed |
| **HotelId** | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Hotels |

### Trips

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| **RoomId** | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed , Relationship with table **Rooms** |
| **BookDate** | **Date** | **NULL** is **not** allowed , **must be** **before ArrivalDate** |
| **ArrivalDate** | **Date** | **NULL** is **not** allowed , **must be** **before ReturnDate** |
| **ReturnDate** | **Date** | **NULL** is **not** allowed |
| **CancelDate** | **Date** |  |

### Accounts

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **Id** | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| **FirstName** | **String** up to **50** symbols, **Unicode** | **NULL** is **not** allowed |
| **MiddleName** | **String** up to **20** symbols, **Unicode** |  |
| **LastName** | **String** up to **50** symbols, **Unicode** | **NULL** is **not** allowed |
| **CityId** | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Cities** |
| **BirthDate** | **Date** | **NULL** is **not** allowed |
| **Email** | **String** up to **100** symbols | **NULL** is **not** allowed, **Unique** |

### AccountsTrips

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| **AccountId** | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Accounts** |
| **TripId** | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Trips** |
| **Luggage** | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, must be **at least** **0** |

## Database design

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

# Section 2. DML (10 pts)

**Before you start, you must import "**DataSet-TripService.sql**". If you have created the structure correctly, the data should be successfully inserted without any errors.**

In this section, you must do some data manipulations:

## Insert

**Insert** some sample data into the database. Write a query to add the following records into the corresponding tables. **All Ids should be auto-generated**.

**Accounts**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FirstName | MiddleName | LastName | CityId | BirthDate | Email |
| John | Smith | Smith | 34 | 1975-07-21 | j\_smith@gmail.com |
| Gosho | NULL | Petrov | 11 | 1978-05-16 | g\_petrov@gmail.com |
| Ivan | Petrovich | Pavlov | 59 | 1849-09-26 | i\_pavlov@softuni.bg |
| Friedrich | Wilhelm | Nietzsche | 2 | 1844-10-15 | f\_nietzsche@softuni.bg |

**Trips**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RoomId | BookDate | ArrivalDate | ReturnDate | CancelDate |
| 101 | 2015-04-12 | 2015-04-14 | 2015-04-20 | 2015-02-02 |
| 102 | 2015-07-07 | 2015-07-15 | 2015-07-22 | 2015-04-29 |
| 103 | 2013-07-17 | 2013-07-23 | 2013-07-24 | NULL |
| 104 | 2012-03-17 | 2012-03-31 | 2012-04-01 | 2012-01-10 |
| 109 | 2017-08-07 | 2017-08-28 | 2017-08-29 | NULL |

## Update

Make all rooms’ prices **14% more expensive** where the **hotel ID** is either **5**, **7** or **9**.

## Delete

Delete **all** of Account ID **47**'s **account's** **trips** from the mapping table.

# Section 3. Querying (40 pts)

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

## EEE-Mails

Select **accounts** whose **emails** **start** with the **letter** '**e**'. Select their **first and last name**, their **birthdate** in the format "MM-dd-yyyy", their **city name,** and their **Email**.

Order them by **city name** **(ascending)**

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FirstName** | **LastName** | **BirthDate** | **Hometown** | **Email** |
| Edgardo | Slessar | 12-29-1983 | Glasgow | e\_slessar@outlook.com |
| Eadith | Gull | 03-03-1983 | Haskovo | e\_gull@outlook.com |
| Eward | Prigg | 12-10-1982 | Shumen | e\_prigg@gmail.com |
| Evvie | Covolini | 01-11-1979 | Wolverhampton | e\_covolini@softuni.bg |

## City Statistics

Select all **cities** with the **count of hotels** in them. Order them by the **hotel count (descending)**, then by **city name**. Do not include cities, which have **no** **hotels** in them.

### Example

|  |  |
| --- | --- |
| **City** | **Hotels** |
| Belfast | 11 |
| Cardiff | 11 |
| Chelyabinsk | 11 |
| Phoenix | 11 |
| San Francisco | 11 |
| Seattle | 11 |
| Veliko Tarnovo | 11 |
| Houston | 10 |
| … | … |

## Longest and Shortest Trips

Find the **longest** and **shortest** **trip** for each **account**, in **days**. Filter the results to **accounts** with **no middle name** and **trips, which are not cancelled** (**CancelDate** is **null**).

Order the results by **Longest Trip days** (**descending),** then by **Shortest Trip (ascending).**

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **AccountId** | **FullName** | **LongestTrip** | **ShortestTrip** |
| 40 | Winna Maisey | 7 | 1 |
| 56 | Tillie Windress | 7 | 1 |
| 57 | Eadith Gull | 7 | 1 |
| 66 | Sargent Rockhall | 7 | 1 |
| 69 | Jerome Flory | 7 | 2 |
| … | … | … | … |

## Metropolis

Find the **top 10** cities, which have the most registered accounts in them. Order them by the **count of** **accounts (descending)**.

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **City** | **Country** | **Accounts** |
| 76 | Tyumen | RU | 5 |
| 12 | Haskovo | BG | 4 |
| 33 | Belfast | UK | 4 |
| … | … | … | … |

## Romantic Getaways

Find all accounts, which have had **one or more** trips to **a hotel in** **their hometown**.

Order them by the **trips count (descending)**, then by **Account ID**.

### Example

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Email** | **City** | **Trips** |
| 50 | t\_joules@mail.com | New York | 2 |
| 19 | m\_stango@yahoo.com | Burgas | 1 |
| 48 | n\_revitt@softuni.bg | Bradford | 1 |
| … | … | … | … |

## GDPR Violation

Retrieve the following information about each trip:

* Trip ID
* Account Full Name
* From – Account hometown
* To – Hotel city
* Duration – the **duration** between the **arrival date** and **return date** in **days**. If a trip is cancelled, the value is "Canceled"

Order the results by **full name**, then by **Trip ID**.

### Example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **Full Name** | **From** | **To** | **Duration** |
| 273 | Adah Douglass Lathaye | Stara Zagora | Cardiff | Canceled |
| 491 | Adah Douglass Lathaye | Stara Zagora | Houston | 4 days |
| 776 | Adah Douglass Lathaye | Stara Zagora | Chelyabinsk | 3 days |
| 133 | Allissa Rickey Gigg | Austin | Veliko Tarnovo | 6 days |
| … | … | … | … | … |

# Section 4. Programmability (14 pts)

## Available Room

Create a **user defined function**, named **udf\_GetAvailableRoom(@HotelId, @Date, @People)**, that receives a **hotel ID**, a desired **date** and the count of **people** that are going to be signing up.

The total price of the room can be calculated by using this formula:

* (HotelBaseRate + RoomPrice) \* PeopleCount

The function should find a suitable room in the provided hotel, based on these conditions:

* The room must **not be already occupied**. A room is occupied if the **date** the customers want to book is **between** the **arrival** and **return dates** of a trip to that room and the trip is **not canceled**.
* The room must be **in** the provided **hotel**.
* The room must have enough **beds** for all the **people**.

If any rooms in the desired hotel **satisfy** the customers’ conditions, find the **highest priced room (by total price)** of all of them and provide them with that room.

The function must return a **message** in the **format**:

* "Room {Room Id}: {Room Type} ({Beds} beds) - ${Total Price}"

If no room could be found, the function should return "No rooms available".

### Example

|  |
| --- |
| **Query** |
| SELECT dbo.udf\_GetAvailableRoom(112, '2011-12-17', 2) |
| **Output** |
| Room 211: First Class (5 beds) - $202.80 |

|  |
| --- |
| **Query** |
| SELECT dbo.udf\_GetAvailableRoom(94, '2015-07-26', 3) |
| **Output** |
| No rooms available |

## Switch Room

Create a **user defined stored procedure**, named **usp\_SwitchRoom(@TripId, @TargetRoomId)**, that receives a **trip** and a **target room** and attempts to **move** the **trip** to **the target room**. A room will only be switched if all of these conditions are true:

* If the **target room** ID is in **a different hotel**, than the **trip** is in, **raise an exception** with the message "Target room is in another hotel!".
* If the **target room** doesn't have **enough beds** for all the **trip's accounts**, **raise an exception** with the message "Not enough beds in target room!".

If all the above conditions pass, **change** the **trip's room ID** to the **target room ID**.

### Example usage

|  |  |
| --- | --- |
| **Query** | **Output** |
| EXEC usp\_SwitchRoom 10, 11  SELECT RoomId FROM Trips WHERE Id = 10 | 11 |
| EXEC usp\_SwitchRoom 10, 7 | Target room is in another hotel! |
| EXEC usp\_SwitchRoom 10, 8 | Not enough beds in target room! |