# Database Basics MS SQL Exam – 24 Jun 2018

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

Submit your solutions in the SoftUni Judge system at <https://judge.softuni.bg/>

# 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**.

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

**Solution:**

CREATE TABLE Cities

(

Id INT PRIMARY KEY IDENTITY,

[Name] NVARCHAR(20) NOT NULL,

CountryCode CHAR(2) NOT NULL

)

CREATE TABLE Hotels

(

Id INT PRIMARY KEY IDENTITY,

[Name] NVARCHAR(30) NOT NULL,

CityId INT NOT NULL FOREIGN KEY REFERENCES Cities(Id),

EmployeeCount INT NOT NULL,

BaseRate DECIMAL(15,2)

)

CREATE TABLE Rooms

(

Id INT PRIMARY KEY IDENTITY,

Price DECIMAL(15,2) NOT NULL,

[Type] NVARCHAR(20) NOT NULL,

Beds INT NOT NULL,

HotelId INT NOT NULL FOREIGN KEY REFERENCES Hotels(Id)

)

CREATE TABLE Trips

(

Id INT PRIMARY KEY IDENTITY,

RoomId INT NOT NULL FOREIGN KEY REFERENCES Rooms(Id),

BookDate DATE NOT NULL,

CONSTRAINT CK\_BEFOREARR CHECK (BookDate < ArrivalDate),

ArrivalDate DATE NOT NULL,

CONSTRAINT CK\_BEFORERET CHECK (ArrivalDate < ReturnDate),

ReturnDate DATE NOT NULL,

CancelDate DATE

)

CREATE TABLE Accounts

(

Id INT PRIMARY KEY IDENTITY,

FirstName NVARCHAR(50) NOT NULL,

MiddleName NVARCHAR(20),

lastName NVARCHAR(50) NOT NULL,

CityId INT NOT NULL FOREIGN KEY REFERENCES Cities(Id),

BirthDate DATE NOT NULL,

Email VARCHAR(100) UNIQUE -- OR UNIQUE BEFORE THE DATATYPE

)

CREATE TABLE AccountsTrips

(

AccountId INT NOT NULL,

TripId INT NOT NULL,

CONSTRAINT PK\_AccountIdTripId

PRIMARY KEY(AccountId, TripId),

CONSTRAINT FK\_AccountIdTripId\_AccountId

FOREIGN KEY(AccountId)

REFERENCES Accounts(Id),

CONSTRAINT FK\_AccountIdTripId\_TripId

FOREIGN KEY(TripId)

REFERENCES Trips(Id),

Luggage INT NOT NULL ,

CONSTRAINT CK\_LUGGAGEABOVEZERO CHECK(Luggage>=0)

)

# 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 have to 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**.

**Solution:**

INSERT INTO Accounts (FirstName, MiddleName, LastName, CityId, BirthDate, Email)

VALUES

('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')

INSERT INTO Trips(RoomId, BookDate, ArrivalDate, ReturnDate, CancelDate)

VALUES

(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)

**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**.

**Solution:**

UPDATE Rooms

SET Price \*= 1.14

WHERE HotelId IN(5,7,9)

## Delete

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

**Solution:**

DELETE FROM AccountsTrips

WHERE AccountId = 47

# 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**).**

## Bulgarian Cities

Select all **cities** in **Bulgaria**. Order them by **city name**.

**Solution:**

SELECT Id, [Name] FROM Cities

WHERE CountryCode = 'BG'

ORDER BY [Name]

### Examples

|  |  |
| --- | --- |
| **Id** | **Name** |
| 15 | Blagoevgrad |
| 4 | Burgas |
| 8 | Dobrich |
| 18 | Gabrovo |
| … | … |

## People Born After 1991

Select all **full names** and **birth years** from accounts,who are born after **1991**.

Order them by **birth year** **(descending)**, then by first name **(ascending)**. Keep in mind that middle names can be **NULL** 😊

**Solution:**

SELECT CONCAT (FirstName+' ', MiddleName + ' ', lastName ) AS [Full Name],

DATEPART(YEAR, BirthDate) AS BirthYear

FROM Accounts

WHERE DATEPART(YEAR, BirthDate) > 1991

ORDER BY DATEPART(YEAR, BirthDate) DESC, FirstName

### Examples

|  |  |
| --- | --- |
| **Full Name** | **BirthYear** |
| Claudia Keely Lotze | 1994 |
| Jourdan Marketa Fawcitt | 1994 |
| Nealson Waiter Villalta | 1994 |
| Palm Van | 1994 |
| Xever Leoine Santi | 1994 |
| Cornell Alidia Grieg | 1993 |
| … | … |

## 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", and their **city name**.

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

**Solution:**

SELECT a.FirstName,

a.lastName,

CONVERT(varchar, a.BirthDate, 110) AS BirthDate,

c.[Name],

a.Email

FROM Accounts AS a

JOIN Cities AS c ON c.Id = a.CityId

WHERE a.Email LIKE 'e%'

ORDER BY c.[Name] DESC

### Examples

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

## City Statistics

Select all **cities** with the **count of hotels** in them. Order them by the **hotel count (descending)**, then by **city name**. Include cities, which have **no** **hotels** in them as well.

**Solution:**

SELECT c.[Name] AS City, ISNULL(COUNT(h.Id) , 0) AS Hotels FROM Hotels AS h

RIGHT JOIN Cities AS c ON h.CityId = c.Id

GROUP BY c.[Name]

ORDER BY COUNT(h.Id) DESC, c.[Name]

### Examples

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

## Expensive First-Class Rooms

Find all First-Class rooms and select the **Id**, **Price**, **Hotel name** and **City name**.

Order them by **Price (descending)**, then by **Room ID**.  
**Solution:**

SELECT r.Id, r.Price, h.[Name], c.[Name] FROM Rooms AS r

JOIN Hotels AS h ON h.Id = r.HotelId

JOIN Cities AS c ON c.Id = h.CityId

WHERE [Type] = 'First class'

ORDER BY r.Price DESC, r.Id

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Price** | **Hotel** | **City** |
| 51 | 90.90 | Recreation Hotel & Spa | Ruse |
| 863 | 90.90 | Exalted Resort & Spa | Volgograd |
| 427 | 90.20 | Stargaze Hotel & Spa | New York |
| 727 | 90.20 | Enterprise Hotel | Yekaterinburg |
| 777 | 90.00 | Nimbus Hotel & Spa | Chelyabinsk |
| 729 | 89.90 | History Resort | Yekaterinburg |
| 512 | 89.70 | Nimbus Hotel & Spa | San Diego |
| 901 | 89.70 | Diorama Resort | Tyumen |
| 116 | 89.50 | Holiday Resort | Blagoevgrad |
| … | … | … | … |

## 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 aren’t cancelled** (**CancelDate** is **null**).

Order the results by **Longest Trip days** (**descending),** then by **Account ID.**

**Solution:**

SELECT a.Id AS AccountId,

CONCAT(a.FirstName, ' ', a.lastName) AS FullName,

MAX(DATEDIFF(DAY, t.ArrivalDate, t.ReturnDate)) AS LongestTrip,

MIN(DATEDIFF(DAY, t.ArrivalDate, t.ReturnDate)) AS ShortestTrip

FROM Accounts AS a

JOIN AccountsTrips AS act ON a.Id = act.AccountId

JOIN Trips AS t ON act.TripId = t.Id

GROUP BY a.Id, t.CancelDate, a.MiddleName, a.FirstName, a.lastName

HAVING a.MiddleName IS NULL AND t.CancelDate IS NULL

ORDER BY MAX(DATEDIFF(DAY, t.ArrivalDate, t.ReturnDate)) DESC, a.Id

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **AccountId** | **FullName** | **LongestTrip** | **ShortestTrip** |
| 40 | Winna Maisey | 7 | 1 |
| 47 | Evvie Covolini | 7 | 2 |
| 56 | Tillie Windress | 7 | 1 |
| 57 | Eadith Gull | 7 | 1 |
| … | … | … | … |

## Metropolis

Find the **top 5** cities, which have the most registered accounts in them. Order them by the **count of** **accounts (descending)**.

**Solution:**

SELECT TOP(5) c.Id,

c.[Name] AS City,

c.CountryCode AS Country ,

COUNT(a.CityId) AS Accounts

FROM Cities AS c

JOIN Accounts AS a ON a.CityId = c.Id

GROUP BY c.Id, c.[Name], c.CountryCode

ORDER BY COUNT(a.CityId) DESC

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **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**.

**Solution:**

SELECT a.Id,

a.Email,

c.[Name] AS City,

COUNT(a.Id) AS Trips

FROM Accounts AS a

JOIN AccountsTrips AS act ON act.AccountId = a.Id

JOIN Trips AS t ON act.TripId = t.Id

JOIN Rooms AS r ON t.RoomId = r.Id

JOIN Hotels AS h ON r.HotelId = h.Id

JOIN Cities AS c ON h.CityId = c.Id

WHERE a.CityId = c.Id

GROUP BY a.Id, a.Email, c.[Name]

ORDER BY COUNT(a.Id) DESC , a.Id

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |
| … | … | … | … |

## Lucrative Destinations

Find the **top 10** cities’ **Total Revenue Sum** (**Hotel Base Rate + Room Price**) and **trip count**.

Count only trips, which were **booked** **in 2016**.

Order them by **Total Revenue (descending)**, then by **Trip count (descending)**

**Solution:**

SELECT TOP(10) c.Id,

c.[Name],

SUM(h.BaseRate + r.Price) AS [Total Revenue],

COUNT(c.Id) AS Trips

FROM Cities AS c

JOIN Hotels AS h ON c.Id = h.CityId

JOIN Rooms AS r ON h.Id = r.HotelId

JOIN Trips AS tr ON r.Id = tr.RoomId

WHERE DATEPART(YEAR,tr.BookDate) = 2016

GROUP BY c.Id,c.[Name]

ORDER BY SUM(h.BaseRate + r.Price) DESC, COUNT(c.Id) DESC

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Name** | **Total Revenue** | **Trips** |
| 56 | Seattle | 795.80 | 6 |
| 47 | Dallas | 649.40 | 6 |
| 17 | Vratsa | 536.60 | 4 |
| 28 | Cardiff | 504.30 | 4 |
| 65 | Chelyabinsk | 386.30 | 3 |
| … | … | … | … |

## Trip Revenues

Find all trips’ **revenue** **(hotel base rate + room price)**. If a trip is **canceled**, its **revenue** is **always 0**. Extract the **trip’s ID**, the **hotel’s name**, the **room type** and the **revenue**.

Order the results by **Room type**, then by the **Trip ID**.

**Solution:**

SELECT t.Id,

h.[Name] AS HotelName,

r.[Type] AS RoomType,

IIF(( t.CancelDate IS NULL), SUM(h.BaseRate + r.Price), 0) AS Revenue

FROM Trips AS t

JOIN Rooms AS r ON t.RoomId = r.Id

JOIN Hotels AS h ON r.HotelId = h.Id

JOIN AccountsTrips AS act ON act.TripId = t.Id

GROUP BY t.Id, h.[Name], r.[Type], t.CancelDate

ORDER BY r.[Type], t.Id

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **HotelName** | **RoomType** | **Revenue** |
| 9 | Cloud Resort | Economy | 51.10 |
| 14 | Lethargy Hotel & Spa | Economy | 39.90 |
| 43 | Courtyard Hotel | Economy | 82.20 |
| 49 | Ranch Hotel | Economy | 0.00 |
| … | … | … | … |

## Top Travelers

Find the **top** **traveler** for **each country**. The top traveler is the **account**, which has the **most trips** to that **country**.

Order the results by the **count of trips (descending)**, then by **Account ID**.  
**Solution:**

SELECT r.Id, r.Email ,r.CountryCode, r.Trips FROM (SELECT a.Id,

a.Email,

c.CountryCode,

COUNT(c.CountryCode) AS Trips,

ROW\_NUMBER()OVER(PARTITION BY c.CountryCode

ORDER BY COUNT(c.CountryCode) DESC) AS Ranking

FROM Accounts AS a

JOIN AccountsTrips AS act ON act.AccountId = a.Id

JOIN Trips AS t ON t.Id = act.TripId

JOIN Rooms AS r ON r.Id = t.RoomId

JOIN Hotels AS h ON h.Id = r.HotelId

JOIN Cities AS c ON c.Id = h.CityId

GROUP BY a.Id, c.CountryCode, a.Email) AS r

WHERE r.Ranking =1

ORDER BY r.Trips DESC, r.Id

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **AccountId** | **Email** | **CountryCode** | **Trips** |
| 80 | a\_flucks@gmail.com | RU | 6 |
| 14 | t\_ludwikiewicz@outlook.com | UK | 5 |
| 32 | a\_roskell@softuni.bg | US | 5 |
| … | … | … | … |

## Luggage Fees

Apart from its base rate and room price, each hotel also has a hidden “luggage fee”. It’s in the terms and conditions, but nobody reads those…

The luggage fee only comes into action if a trip has **more than 5 items of luggage** and it’s equal to the **number of luggage items,** **multiplied by 5**.

Take into account only trips, which have **more than 0** luggage.

Order the results by the **count of luggage** **(descending)**

**Solution:**

SELECT t.Id AS TripId,

SUM(act.Luggage) AS Luggage,

CASE

WHEN (SUM(act.Luggage) > 5) THEN CONCAT('$',(SUM(act.Luggage) \* 5))

ELSE '$0'

END

AS Fee

FROM AccountsTrips AS act

JOIN Trips AS t ON act.TripId = t.Id

WHERE act.Luggage > 0

GROUP BY t.Id

ORDER BY SUM(act.Luggage) DESC

### Examples

|  |  |  |
| --- | --- | --- |
| **TripId** | **Luggage** | **Fee** |
| 632 | 7 | $35 |
| 617 | 6 | $30 |
| 833 | 6 | $30 |
| 264 | 6 | $30 |
| 273 | 6 | $30 |
| 306 | 6 | $30 |
| 323 | 6 | $30 |
| 330 | 6 | $30 |
| 428 | 6 | $30 |
| 457 | 6 | $30 |
| 405 | 5 | $0 |
| … | … | … |

## 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**.

**Solution:**

SELECT t.Id,

CONCAT(a.FirstName + ' ', a.MiddleName + ' ',a.lastName ) AS [Full Name],

ct.[Name] AS [From],

c.[Name] AS [To],

CASE

WHEN (t.CancelDate IS NULL) THEN CONCAT( (DATEDIFF(DAY, t.ArrivalDate, t.ReturnDate)), ' days')

ELSE 'Canceled'

END

AS [Duration]

FROM Trips AS t

JOIN AccountsTrips AS act ON t.Id = act.TripId

JOIN Accounts AS a ON a.Id = act.AccountId

JOIN Cities AS ct ON a.CityId = ct.Id

JOIN Rooms AS r ON r.Id = t.RoomId

JOIN Hotels AS h ON h.Id = r.HotelId

JOIN Cities AS c ON c.Id = h.CityId

ORDER BY CONCAT(a.FirstName + ' ', a.MiddleName + ' ',a.lastName ), t.Id

### Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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)

## 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! |

**Solution:**

CREATE PROCEDURE usp\_SwitchRoom(@TripId INT, @TargetRoomId INT)

AS

BEGIN

DECLARE @hotelName VARCHAR(30)

DECLARE @hotelDestination VARCHAR(30)

DECLARE @firstRoomBedCount INT

DECLARE @secondRoomBedCount INT

SET @hotelName =

(

SELECT TOP(1) h.[Name] FROM Trips AS t

JOIN Rooms AS r ON t.RoomId = r.Id

JOIN Hotels AS h ON r.HotelId = h.Id

WHERE t.Id = @TripId

)

SET @hotelDestination =

(

SELECT TOP(1) h.[Name] FROM Trips AS t

JOIN Rooms AS r ON t.RoomId = r.Id

JOIN Hotels AS h ON r.HotelId = h.Id

WHERE t.RoomId = @TargetRoomId

)

IF(@hotelName <> @hotelDestination AND @hotelDestination IS NOT NULL AND @hotelName IS NOT NULL )

BEGIN

SELECT 'Target room is in another hotel!'

END

ELSE IF(@hotelName = @hotelDestination AND @hotelDestination IS NOT NULL)

BEGIN

SET @firstRoomBedCount = (ISNULL((SELECT TOP(1) r.Beds FROM Trips AS t

JOIN Rooms AS r ON t.RoomId = r.Id

WHERE r.Id = @TargetRoomId), 0))

SET @secondRoomBedCount = (SELECT COUNT(\*) FROM AccountsTrips

WHERE TripId = @TripId)

IF(@firstRoomBedCount < @secondRoomBedCount )

BEGIN

SELECT 'Not enough beds in target room!'

END

ELSE

BEGIN

UPDATE Trips

SET RoomId = @TargetRoomId

WHERE Id = @TripId

END

END

END