Exam Basics DB 24.VI.2018 TriptService

/\*PART I DDL \*/

/\* 01DatabaseDesign\*/

CREATE TABLE Cities

(

Id INT NOT NULL IDENTITY PRIMARY KEY,

[Name] NVARCHAR(20) NOT NULL,

CountryCode CHAR(2) NOT NULL

)

CREATE TABLE Hotels

(

Id INT NOT NULL IDENTITY PRIMARY KEY,

[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 NOT NULL IDENTITY PRIMARY KEY,

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 NOT NULL IDENTITY PRIMARY KEY,

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

BookDate DATE NOT NULL,

ArrivalDate DATE NOT NULL ,

ReturnDate DATE NOT NULL,

CancelDate DATE,

CONSTRAINT CH\_BookDate\_Befor\_ArrivalDate CHECK(BookDate < ArrivalDate),

CONSTRAINT CH\_ArrivalDate\_Befor\_ReturnDate CHECK(ArrivalDate < ReturnDate)

)

CREATE TABLE Accounts

(

Id INT NOT NULL IDENTITY PRIMARY KEY,

FirstName NVARCHAR(50) NOT NULL,

MiddleName NVARCHAR(30),

LastName NVARCHAR(50) NOT NULL,

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

BirthDate DATE NOT NULL,

Email VARCHAR(100) NOT NULL UNIQUE

)

CREATE TABLE AccountsTrips

(

AccountId INT NOT NULL CONSTRAINT FK\_AccountsTrips\_Accounts FOREIGN KEY REFERENCES Accounts(Id),

TripId INT NOT NULL CONSTRAINT FK\_AccountsTrips\_Trips FOREIGN KEY REFERENCES Trips(Id),

Luggage INT NOT NULL CHECK(Luggage >= 0),

CONSTRAINT PK\_AccountsTrips PRIMARY KEY(AccountId, TripId)

)

/\*PART II DML \*/

/\* 02Insert\*/

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)

/\* 03Update\*/

UPDATE Rooms

SET Price \*= 1.14

WHERE HotelId IN(5, 7, 9)

/\* 04Delete\*/

DELETE FROM AccountsTrips

WHERE AccountId = 47

/\*PART III Querying \*/

/\* 05BulgarianCities\*/

SELECT Id, [Name]

FROM Cities

WHERE CountryCode = 'BG'

ORDER BY [Name]

/\* 06PeopleBornAfter1991\*/

SELECT

CASE WHEN MiddleName IS NOT NULL THEN CONCAT(FirstName, ' ', MiddleName, ' ', LastName)

ELSE CONCAT(FirstName, ' ', LastName)

END AS [Full Name],

YEAR(BirthDate) AS BirthYear

FROM Accounts

WHERE YEAR(BirthDate) > '1991'

ORDER BY BirthYear DESC, [Full Name]

/\* 07EEEMailsL\*/

SELECT a.FirstName, a.LastName, FORMAT(a.BirthDate, 'MM-dd-yyyy') AS BirthDate,

c.[Name] AS Hometown, a.Email

FROM Accounts AS a

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

WHERE Email LIKE 'e%'

ORDER BY Hometown DESC

/\* 08CityStatistics\*/

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

FROM Cities AS c

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

GROUP BY c.[Name]

ORDER BY Hotels DESC, City

/\*09ExpensiveFirstClassRooms\*/

SELECT r.Id, r.Price, h.[Name] AS Hotel, c.[Name] AS City

FROM Rooms AS r

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

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

WHERE r.[Type] = 'First Class'

ORDER BY r.Price DESC, r.Id

/\*10LongestAndShortestTrips\*/

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 aCt.AccountId = a.Id

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

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

GROUP BY a.Id, CONCAT(a.FirstName, ' ', a.LastName)

ORDER BY LongestTrip DESC, AccountId

/\* 11Metropolis\*/

SELECT TOP(5) c.Id, c.[Name] AS City, c.CountryCode AS Country,

COUNT(a.Id) 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 Accounts DESC

/\* 12RomanticGetaways\*/

SELECT a.Id, a.Email, c.[Name] AS City, COUNT(t.Id) AS Trips

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 = a.CityId

WHERE a.CityId = h.CityId

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

ORDER BY Trips DESC, a.Id

/\* 13LucrativeDestinations\*/

SELECT TOP (10) c.Id, c.[Name],

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

COUNT(t.Id) AS Trips

FROM Cities AS c

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

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

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

WHERE YEAR(t.BookDate) = 2016

GROUP BY c.Id, c.[Name]

ORDER BY [Total Revenue] DESC, Trips DESC

/\* 14TripRevenues\*/

SELECT t.Id,

h.[Name] AS HotelName,

r.[Type] AS RoomType,

CASE

WHEN t.CancelDate IS NULL THEN SUM(r.Price + h.BaseRate)

ELSE 0

END AS Revenue

FROM Trips AS t

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

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

JOIN AccountsTrips AS [at] ON [at].TripId = t.Id

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

ORDER BY RoomType, t.Id

/\* 15TopTravelers\*/

SELECT AccountId, Email, CountryCode, Trips

FROM (SELECT a.Id AS AccountId, a.Email, c.CountryCode,

COUNT(t.Id) AS Trips,

RANK() OVER (PARTITION BY c.CountryCode ORDER BY COUNT(t.Id) DESC, a.Id) AS CountryTripsRank

FROM Accounts AS a

JOIN AccountsTrips AS [at] ON [at].AccountId = a.Id

JOIN Trips AS t ON t.Id = [at].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 c.CountryCode, a.Email, a.Id

) AS TrippersPerCountry

WHERE CountryTripsRank = 1

ORDER BY Trips DESC, AccountId

/\* 16LuggageFees\*/

SELECT t.Id AS TripId,

SUM([at].Luggage) AS Luggage,

CONCAT('$', CASE WHEN SUM([at].Luggage) > 5 THEN SUM([at].Luggage) \* 5 ELSE 0 END) AS Fee

FROM Trips AS t

JOIN AccountsTrips AS [at] ON [at].TripId = t.Id

GROUP BY t.Id

HAVING SUM([at].Luggage) > 0

ORDER BY Luggage DESC

/\* 17GDPRViolation\*/

SELECT t.Id,

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

cA.[Name] AS [From],

cH.[Name] AS [To],

CASE WHEN t.CancelDate IS NOT NULL THEN 'Canceled'

ELSE CONCAT(DATEDIFF(DAY, t.ArrivalDate, t.ReturnDate), ' days')

END AS Duration

FROM Trips AS t

JOIN AccountsTrips AS [at] ON [at].TripId = t.Id

JOIN Accounts AS a ON a.Id = [at].AccountId

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

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

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

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

ORDER BY [Full Name], t.Id

/\*PART IV Programmability\*/

/\* 18AvailableRoom\*/

CREATE FUNCTION udf\_GetAvailableRoom(@HotelId INT, @Date DATE, @People INT)

RETURNS NVARCHAR(MAX) AS

BEGIN

DECLARE @BookedRooms TABLE(Id INT)

INSERT INTO @BookedRooms

SELECT DISTINCT r.Id

FROM Rooms AS r

LEFT JOIN Trips AS t ON t.RoomId = r.Id

WHERE r.HotelId = @HotelId AND (@Date BETWEEN t.ArrivalDate AND t.ReturnDate) AND t.CancelDate IS NULL

DECLARE @Rooms TABLE(Id INT, Price DECIMAL(15, 2), Type VARCHAR(20), Beds INT, TotalPrice DECIMAL(15, 2))

INSERT INTO @Rooms

SELECT TOP(1) r.Id, r.Price, r.[Type], r.Beds, @People \* (h.BaseRate + r.Price) AS TotalPrice

FROM Rooms AS r

LEFT JOIN Hotels AS h on h.Id = r.HotelId

WHERE r.HotelId = @HotelId AND r.Beds >= @People AND r.Id NOT IN (SELECT Id FROM @BookedRooms)

ORDER BY TotalPrice DESC

DECLARE @RoomCount INT = (SELECT COUNT(\*) FROM @Rooms)

IF (@RoomCount < 1)

BEGIN

RETURN 'No rooms available'

END

DECLARE @Result NVARCHAR(MAX) = (SELECT CONCAT('Room ', Id, ': ', [Type], ' (', Beds, ' beds) - ',

'$', TotalPrice) FROM @Rooms)

RETURN @Result

END

--In Judge must be paste without this below

SELECT dbo.udf\_GetAvailableRoom(112, '2011-12-17', 2)

SELECT dbo.udf\_GetAvailableRoom(94, '2015-07-26', 3)

/\* 19SwitchRoom\*/

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

AS

BEGIN

DECLARE @targetRoom\_HotelId INT = (SELECT HotelId FROM Rooms WHERE Id = @TargetRoomId)

DECLARE @trip\_RoomId INT = (SELECT RoomId FROM Trips WHERE Id = @TripId)

DECLARE @trip\_Rooms\_HotelId INT = (SELECT HotelId FROM Rooms WHERE Id = @trip\_RoomId)

IF (@targetRoom\_HotelId <> @trip\_Rooms\_HotelId)

BEGIN

RAISERROR('Target room is in another hotel!', 16, 1)

END

DECLARE @targetRoom\_Beds INT = (SELECT Beds FROM Rooms WHERE Id = @TargetRoomId)

DECLARE @trip\_AccountsCount INT = (SELECT COUNT(AccountId) FROM AccountsTrips WHERE TripId = @TripId)

IF (@targetRoom\_Beds < @trip\_AccountsCount)

BEGIN

RAISERROR('Not enough beds in target room!', 16, 1)

END

UPDATE Trips

SET RoomId = @TargetRoomId

WHERE Id = @TripId

END

--In judje must be paste only one query, but this query below does not work good - 4/7 in Judge, because

--exception 'Target room is in another hotel!' don't work in transaction, I don't know way

GO

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

AS

BEGIN

BEGIN TRANSACTION

UPDATE Trips

SET RoomId = @TargetRoomId

WHERE Id = @TripId

DECLARE @targetRoom\_HotelId INT = (SELECT HotelId FROM Rooms WHERE Id = @TargetRoomId)

DECLARE @trip\_RoomId INT = (SELECT RoomId FROM Trips WHERE Id = @TripId)

DECLARE @trip\_Rooms\_HotelId INT = (SELECT HotelId FROM Rooms WHERE Id = @trip\_RoomId)

IF (@targetRoom\_HotelId <> @trip\_Rooms\_HotelId)

BEGIN

ROLLBACK

RAISERROR('Target room is in another hotel!', 16, 1)

RETURN

END

DECLARE @targetRoom\_Beds INT = (SELECT Beds FROM Rooms WHERE Id = @TargetRoomId)

DECLARE @trip\_AccountsCount INT = (SELECT COUNT(AccountId) FROM AccountsTrips WHERE TripId = @TripId)

IF (@targetRoom\_Beds < @trip\_AccountsCount)

BEGIN

ROLLBACK

RAISERROR('Not enough beds in target room!', 16, 1)

RETURN

END

COMMIT

END

--In judje must be paste only one query, but this below is autor's solution

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

AS

BEGIN

DECLARE @SourceHotelId INT = (SELECT H.Id

FROM Hotels H

JOIN Rooms R on H.Id = R.HotelId

JOIN Trips T on R.Id = T.RoomId

WHERE T.Id = @TripId)

DECLARE @TargetHotelId INT = (SELECT H.Id

FROM Hotels H

JOIN Rooms R on H.Id = R.HotelId

WHERE R.Id = @TargetRoomId)

IF (@SourceHotelId <> @TargetHotelId)

THROW 50013, 'Target room is in another hotel!', 1

DECLARE @PeopleCount INT = (SELECT COUNT(\*)

FROM AccountsTrips

WHERE TripId = @TripId)

DECLARE @TargetRoomBeds INT = (SELECT Beds

FROM Rooms

WHERE Id = @TargetRoomId)

IF (@PeopleCount > @TargetRoomBeds)

THROW 50013, 'Not enough beds in target room!', 1

UPDATE Trips

SET RoomId = @TargetRoomId

WHERE Id = @TripId

END

--In Judge must be paste without this below

EXEC usp\_SwitchRoom 10, 11

SELECT RoomId FROM Trips WHERE Id = 10

EXEC usp\_SwitchRoom 10, 7

EXEC usp\_SwitchRoom 10, 8

DROP PROCEDURE usp\_SwitchRoom

/\*PART Bonus \*/

/\* 20CancelTrip\*/

CREATE TRIGGER TR\_CancelTrip ON Trips

INSTEAD OF DELETE AS

BEGIN

UPDATE Trips

SET CancelDate = GETDATE()

WHERE Id IN (SELECT Id FROM deleted WHERE CancelDate IS NULL)

END

--In Judge must be paste without this below

DELETE FROM Trips

WHERE Id IN (48, 49, 50)