Exam Basics DB 3.I.2018 RentACar

/\*PART I DDL \*/

/\* 01DatabaseDesign\*/

CREATE TABLE Clients

(

Id INT NOT NULL IDENTITY CONSTRAINT PK\_Clients PRIMARY KEY,

FirstName NVARCHAR(30) NOT NULL,

LastName NVARCHAR(30) NOT NULL,

Gender CHAR(1) CONSTRAINT CH\_Gender\_MorF CHECK(Gender IN('M', 'F')),

BirthDate DATETIME,

CreditCard NVARCHAR(30) NOT NULL,

CardValidity DATETIME,

Email NVARCHAR(50) NOT NULL

)

CREATE TABLE Towns

(

Id INT NOT NULL IDENTITY CONSTRAINT PK\_Towns PRIMARY KEY,

[Name] NVARCHAR(50) NOT NULL

)

CREATE TABLE Offices

(

Id INT NOT NULL IDENTITY CONSTRAINT PK\_Offices PRIMARY KEY,

[Name] NVARCHAR(40),

ParkingPlaces INT,

TownId INT NOT NULL CONSTRAINT FK\_Offices\_Towns FOREIGN KEY REFERENCES Towns(Id)

)

CREATE TABLE Models

(

Id INT NOT NULL IDENTITY CONSTRAINT PK\_Models PRIMARY KEY,

Manufacturer NVARCHAR(50) NOT NULL,

Model NVARCHAR(50) NOT NULL,

ProductionYear DATETIME,

Seats INT,

Class NVARCHAR(10),

Consumption DECIMAL(14, 2)

)

CREATE TABLE Vehicles

(

Id INT NOT NULL IDENTITY CONSTRAINT PK\_Vehicles PRIMARY KEY,

ModelId INT NOT NULL CONSTRAINT FK\_Vehicles\_Models FOREIGN KEY REFERENCES Models(Id),

OfficeId INT NOT NULL CONSTRAINT FK\_Vehicles\_Offices FOREIGN KEY REFERENCES Offices(Id),

Mileage INT

)

CREATE TABLE Orders

(

Id INT NOT NULL IDENTITY CONSTRAINT PK\_Orders PRIMARY KEY,

ClientId INT NOT NULL CONSTRAINT FK\_Orders\_Clients FOREIGN KEY REFERENCES Clients(Id),

TownId INT NOT NULL CONSTRAINT FK\_Orders\_Towns FOREIGN KEY REFERENCES Towns(Id),

VehicleId INT NOT NULL CONSTRAINT FK\_Orders\_Vehicles FOREIGN KEY REFERENCES Vehicles(Id),

CollectionDate DATETIME NOT NULL,

CollectionOfficeId INT NOT NULL CONSTRAINT FK\_Orders\_Offices\_Collection FOREIGN KEY REFERENCES Offices(Id),

ReturnDate DATETIME,

ReturnOfficeId INT CONSTRAINT FK\_Orders\_Offices\_Return FOREIGN KEY REFERENCES Offices(Id),

Bill DECIMAL(14, 2),

TotalMileage INT

)

/\*PART II DML \*/

/\* 02Insert\*/

INSERT INTO Models(Manufacturer, Model, ProductionYear, Seats, Class, Consumption)

VALUES

('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)

INSERT INTO Orders(ClientId, TownId, VehicleId, CollectionDate, CollectionOfficeId, ReturnDate,

ReturnOfficeId, Bill, TotalMileage)

VALUES

(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)

/\* 03Update\*/

UPDATE Models

SET Class = 'Luxury'

WHERE Consumption > 20

/\* 04Delete\*/

DELETE FROM Orders

WHERE ReturnDate IS NULL

/\*PART III Querying \*/

/\* 05Showroom\*/

SELECT Manufacturer, Model

FROM Models

ORDER BY Manufacturer, Id DESC

/\* 06YGeneration\*/

SELECT FirstName, LastName

FROM Clients

WHERE YEAR(BirthDate) BETWEEN '1977' AND '1994'

ORDER BY FirstName, LastName, Id

/\* 07SpaciousOffice\*/

SELECT t.[Name] AS TownName, o.[Name] AS OfficeName, o.ParkingPlaces

FROM Towns AS t

JOIN Offices AS o ON o.TownId = t.Id

WHERE o.ParkingPlaces > 25

ORDER BY TownName, o.Id

/\* 08AvailableVehicles\*/

SELECT m.Model, m.Seats, v.Mileage

FROM Models AS m

JOIN Vehicles AS v ON v.ModelId = m.Id

WHERE v.Id NOT IN

(SELECT o.VehicleId FROM Orders AS o WHERE o.ReturnDate IS NULL)

ORDER BY v.Mileage ASC, m.Seats DESC, m.Id ASC

/\* 09OfficesPerTown\*/

SELECT t.[Name] AS TownName, COUNT(t.Id) AS OfficesNumber

FROM Towns AS t

JOIN Offices AS o ON o.TownId = t.Id

GROUP BY o.TownId, t.[Name]

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

/\* 10BuyersBestChoice\*/

SELECT Manufacturer, Model, SUM(CountOfOrdersById) AS TimesOrdered

FROM (

SELECT m.Manufacturer, m.Model, COUNT(v.Id) AS CountOfOrdersById

FROM Orders As o

LEFT JOIN Vehicles AS v ON v.Id = o.VehicleId

RIGHT JOIN Models AS m ON m.Id = v.ModelId

GROUP BY m.Manufacturer, m.Model, v.Id

) AS ovm

GROUP BY Manufacturer, Model

ORDER BY TimesOrdered DESC, Manufacturer DESC, Model ASC

/\* 11KindaPerson\*/

SELECT Names, Class

FROM (

SELECT c.FirstName + ' ' + c.LastName AS Names, m.Class,

RANK() OVER (PARTITION BY (c.FirstName + ' ' + c.LastName) ORDER BY COUNT(m.Class) DESC) AS [Rank]

FROM Orders AS o

JOIN Clients AS c ON c.Id = o.ClientId

JOIN Vehicles AS v ON v.Id = o.VehicleId

JOIN Models AS m ON m.Id = v.ModelId

GROUP BY c.FirstName + ' ' + c.LastName, m.Class

) AS ocvm

WHERE [Rank] = 1

ORDER BY Names, Class

/\* 12AgeGroupsRevenue\*/

SELECT AgeGroup =

CASE

WHEN YEAR(c.BirthDate) BETWEEN '1970' AND '1979' THEN '70''s'

WHEN YEAR(c.BirthDate) BETWEEN '1980' AND '1989' THEN '80''s'

WHEN YEAR(c.BirthDate) BETWEEN '1990' AND '1999' THEN '90''s'

ELSE 'Others'

END,

SUM(o.Bill) AS Revenue,

AVG(o.TotalMileage) AS AverageMileage

FROM Clients AS c

JOIN Orders AS o ON o.ClientId = c.Id

GROUP BY

CASE

WHEN YEAR(c.BirthDate) BETWEEN '1970' AND '1979' THEN '70''s'

WHEN YEAR(c.BirthDate) BETWEEN '1980' AND '1989' THEN '80''s'

WHEN YEAR(c.BirthDate) BETWEEN '1990' AND '1999' THEN '90''s'

ELSE 'Others'

END

--ORDER BY AgeGroup

/\* 13ConsumpionInMind\*/

SELECT Manufacturer, AverageConsumption

FROM

(SELECT TOP(7) m.Model, m.Manufacturer, AVG(m.Consumption) AS AverageConsumption,

COUNT(m.Model) AS [Counter]

FROM Orders AS o

JOIN Vehicles AS v ON v.Id = o.VehicleId

JOIN Models AS m ON m.Id = v.ModelId

GROUP BY m.Manufacturer, m.Model

ORDER BY [Counter] DESC

) AS ovm

WHERE AverageConsumption BETWEEN 5 AND 15

ORDER BY Manufacturer, AverageConsumption

/\* 14DeptHunter\*/

SELECT [Category Name], Email, Bill, Town

FROM

(SELECT ROW\_NUMBER() OVER(PARTITION BY t.[Name] ORDER BY o.Bill DESC) AS BillsByTownDesc,

CONCAT(c.FirstName, ' ', c.LastName) AS [Category Name],

c.Id AS ClientId, c.Email, o.Bill, t.[Name] AS Town

FROM Orders AS o

JOIN Clients AS c ON c.Id = o.ClientId

JOIN Towns AS t ON t.Id = o.TownId

WHERE o.CollectionDate > c.CardValidity AND o.Bill IS NOT NULL

) AS oct

WHERE BillsByTownDesc IN(1, 2)

ORDER BY Town, Bill, ClientId

/\* 15TownStatistics\*/

SELECT t.[Name] AS TownName,

(SUM(oc.M) \* 100) / (ISNULL(SUM(oc.M), 0) + ISNULL(SUM(oc.F), 0)) AS MailPercent,

(SUM(oc.F) \* 100) / (ISNULL(SUM(oc.M), 0) + ISNULL(SUM(oc.F), 0)) AS FemalePercent

FROM

(SELECT o.TownId,

CASE WHEN (Gender = 'M') THEN COUNT(o.Id) ELSE NULL END AS M,

CASE WHEN (Gender = 'F') THEN COUNT(o.Id) ELSE NULL END AS F

FROM Orders AS o

JOIN Clients AS c ON c.Id = o.ClientId

GROUP BY c.Gender, o.TownId

) oc

JOIN Towns AS t ON t.Id = oc.TownId

GROUP BY t.[Name]

/\* 16HomeSeetHome\*/

WITH cte\_Rancs(ReturnOfficeId, OfficeId, Id, Manufacturer, Model)

AS

(SELECT ovm.ReturnOfficeId, ovm.OfficeId, ovm.Id, ovm.Manufacturer, ovm.Model

FROM

(SELECT DENSE\_RANK() OVER(PARTITION BY v.Id ORDER BY o.CollectionDate DESC) AS LatestRentCarsRank,

o.ReturnOfficeId, v.OfficeId, v.Id, m.Manufacturer, m.Model

FROM Orders AS o

RIGHT JOIN Vehicles AS v ON v.Id = o.VehicleId

JOIN Models AS m ON m.Id = v.ModelId

) AS ovm

WHERE LatestRentCarsRank = 1)

SELECT CONCAT(Manufacturer, ' - ', Model) AS Vehicle,

[Location] =

CASE

WHEN(SELECT COUNT(\*) FROM Orders AS o WHERE o.VehicleId = cte\_Rancs.Id) = 0 THEN 'home'

WHEN(cte\_Rancs.ReturnOfficeId IS NULL) THEN 'on a rent'

WHEN(cte\_Rancs.OfficeId <> cte\_Rancs.ReturnOfficeId)

THEN (SELECT CONCAT(t.[Name], ' - ', o.[Name])

FROM Towns AS t JOIN Offices AS o ON o.TownId = t.Id

WHERE o.Id = cte\_Rancs.ReturnOfficeId)

END

FROM cte\_Rancs

ORDER BY Vehicle, cte\_Rancs.Id

/\*PART IV Programmability \*/

/\* 17FindMyRide\*/

CREATE FUNCTION udf\_CheckForVehicle(@townName VARCHAR(50), @seatsNumber INT)

RETURNS NVARCHAR(MAX)

AS

BEGIN

DECLARE @result NVARCHAR(100) = (SELECT TOP(1) CONCAT(o.[Name], ' - ', m.Model)

FROM Towns AS t

JOIN Offices AS o ON o.TownId = t.Id

JOIN Vehicles AS v ON v.OfficeId = o.Id

JOIN Models AS m ON m.Id = v.ModelId

WHERE t.[Name] = @townName AND m.Seats = @seatsNumber

ORDER BY o.[Name])

IF (@result IS NULL)

RETURN 'NO SUCH VEHICLE FOUND'

RETURN @result

END

--In Judge must be paste without this below

SELECT dbo.udf\_CheckForVehicle ('La Escondida', 9)

/\* 18MoveAVehicle\*/

CREATE PROCEDURE usp\_MoveVehicle @vehicleId INT, @officeId INT

AS

BEGIN

BEGIN TRANSACTION

UPDATE Vehicles

SET OfficeId = @officeId

WHERE Id = @vehicleId

DECLARE @countVehiclesById INT = (SELECT COUNT(v.Id) FROM Vehicles AS v WHERE v.OfficeId = @officeId)

DECLARE @parkingPlaces INT = (SELECT ParkingPlaces FROM Offices WHERE Id = @officeId)

IF (@parkingPlaces < @countVehiclesById)

BEGIN

ROLLBACK

RAISERROR('Not enough room in this office!', 16, 1)

RETURN

END

COMMIT

END

--In Judge must be paste without this below

EXEC usp\_MoveVehicle 7, 32;

SELECT OfficeId FROM Vehicles WHERE Id = 7

/\* 19MoveTheTally\*/

CREATE TRIGGER tr\_Orders\_MoveTheTally ON Orders

FOR UPDATE

AS

BEGIN

DECLARE @newTotalMileage INT = (SELECT TotalMileage FROM inserted)

DECLARE @oldTotalMileage INT = (SELECT TotalMileage FROM deleted)

DECLARE @vehicleId INT = (SELECT VehicleId FROM inserted)

IF (@oldTotalMileage IS NULL AND @vehicleId IS NOT NULL)

BEGIN

UPDATE Vehicles

SET Mileage += @newTotalMileage

WHERE Id = @vehicleId

END

END

--In Judge must be paste without this below

UPDATE Orders

SET

TotalMileage = 100

WHERE Id = 40