# Database Basics MS SQL Exam – 16 Apr 2019

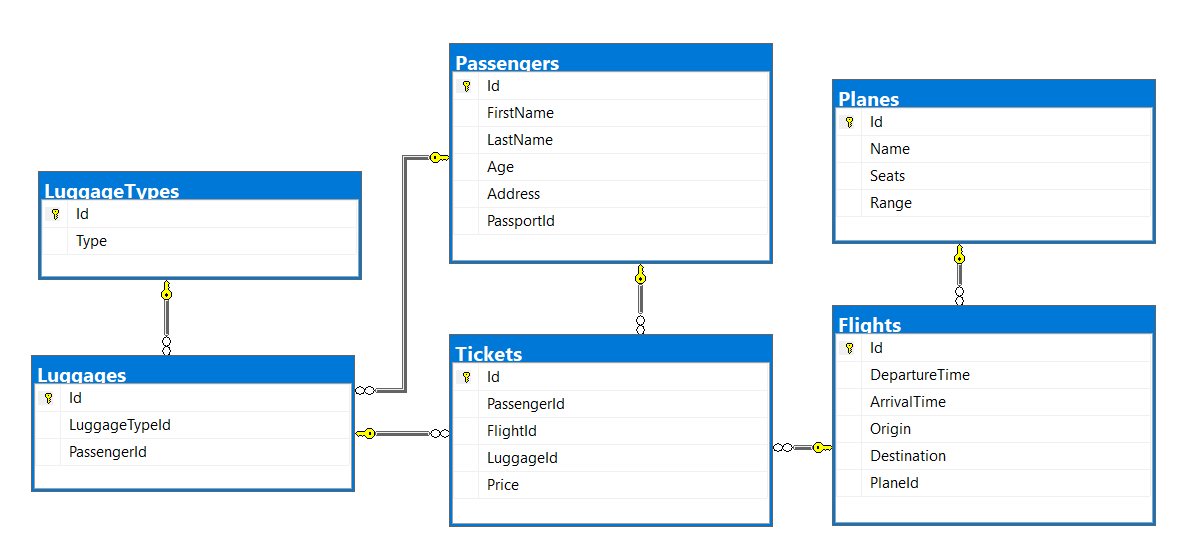
Exam problems for the [“Database Basics” course @ SoftUni](https://softuni.bg/courses/databases-basics-ms-sql-server).

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

# Airport

# Section 1. DDL (30 pts)

You are given an E/R Diagram of the Airport:



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

* Planes – contains information about the **planes**.
* Flights – contains information about the **flights**.
* Passеngers – contains information about the **passengers**
* LuggageTypes – contains information about the **type of luggage's**.
* Flights – contains information about the **flights**.
  + Each flight has a plane.
* Luggages – contains information about the **luggage's**.
  + Each luggage has a luggage type.
* Tickets – contains information about the tickets.
  + Each ticket has a passenger.
  + Each ticket has a flight.
  + Each ticket has a luggage.

**Planes**

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

**Flights**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| DepartureTime | **Datetime** | None |
| ArrivalTime | **Datetime** | None |
| Origin | **String** up to 50 symbols | **NULL** is **not** allowed |
| Destination | **String** up to 50 symbols | **NULL** is **not** allowed |
| PlaneId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table Planes |

**Passengers**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| FirstName | **String** up to 30 symbols | **NULL** is **not** allowed |
| LastName | **String** up to 30 symbols | **NULL** is **not** allowed |
| Age | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed |
| Address | **String** up to 30 symbols | **NULL** is **not** allowed |
| PassportId | **String** with **exactly 11** symbols | **NULL** is **not** allowed |

**LuggageTypes**

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

**Luggages**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| LuggageTypeId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **LuggageTypes** |
| PassengerId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Passengers** |

**Tickets**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Constraints** |
| Id | **Integer** from **0** to **2,147,483,647** | Unique table **identificator**, **Identity** |
| PassеngerId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Passengers** |
| FlightId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Flights** |
| LuggageId | **Integer** from **0** to **2,147,483,647** | **NULL** is **not** allowed, Relationship with table **Luggages** |
| Price | **Decimal** number with **two-digit** precision | **NULL** is **not** allowed |

## Database Design

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

CREATE TABLE Planes(

[Id] INT PRIMARY KEY IDENTITY NOT NULL,

[Name] NVARCHAR(30) NOT NULL,

[Seats] INT NOT NULL,

[Range] INT NOT NULL

)

CREATE TABLE Flights(

[Id] INT PRIMARY KEY IDENTITY NOT NULL,

[DepartureTime] DATETIME,

[ArrivalTime] DATETIME,

[Origin] NVARCHAR(50) NOT NULL,

[Destination] NVARCHAR(50) NOT NULL,

[PlaneId] INT FOREIGN KEY REFERENCES Planes(Id) NOT NULL

)

CREATE TABLE Passengers(

[Id] INT PRIMARY KEY IDENTITY NOT NULL,

[FirstName] NVARCHAR(30) NOT NULL,

[LastName] NVARCHAR(30) NOT NULL,

[Age] INT NOT NULL,

[Address] NVARCHAR(30) NOT NULL,

[PassportId] CHAR(11) NOT NULL

)

CREATE TABLE LuggageTypes(

[Id] INT PRIMARY KEY IDENTITY NOT NULL,

[Type] NVARCHAR(30) NOT NULL

)

CREATE TABLE Luggages(

[Id] INT PRIMARY KEY IDENTITY NOT NULL,

[LuggageTypeId] INT FOREIGN KEY REFERENCES LuggageTypes(Id) NOT NULL,

[PassengerId] INT FOREIGN KEY REFERENCES Passengers(Id) NOT NULL

)

CREATE TABLE Tickets(

[Id] INT PRIMARY KEY IDENTITY NOT NULL,

[PassengerId] INT FOREIGN KEY REFERENCES Passengers(Id) NOT NULL,

[FlightId] INT FOREIGN KEY REFERENCES Flights(Id) NOT NULL,

[LuggageId] INT FOREIGN KEY REFERENCES Luggages(Id) NOT NULL,

[Price] DECIMAL(18, 2) NOT NULL

)

# Section 2. DML (10 pts)

**Before you start, you must import “**DataSet-Airport.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**.

INSERT INTO Planes([Name], [Seats], [Range])

VALUES

('Airbus 336', 112, 5132),

('Airbus 330', 432, 5325),

('Boeing 369', 231, 2355),

('Stelt 297', 254, 2143),

('Boeing 338', 165, 5111),

('Airbus 558', 387, 1342),

('Boeing 128', 345, 5541)

INSERT INTO LuggageTypes([Type])

VALUES

('Crossbody Bag'),

('School Backpack'),

('Shoulder Bag')

**Planes**

|  |  |  |
| --- | --- | --- |
| Name | Seats | Range |
| Airbus 336 | 112 | 5132 |
| Airbus 330 | 432 | 5325 |
| Boeing 369 | 231 | 2355 |
| Stelt 297 | 254 | 2143 |
| Boeing 338 | 165 | 5111 |
| Airbus 558 | 387 | 1342 |
| Boeing 128 | 345 | 5541 |

**Luggage Types**

|  |
| --- |
| Type |
| Crossbody Bag |
| School Backpack |
| Shoulder Bag |

## Update

Make all flights to "**Carlsbad**" 13% more expensive.

UPDATE Tickets

SET Price \*= 1.13

WHERE FlightId IN(

SELECT Id

FROM Flights

WHERE Destination LIKE 'Carlsbad'

)

## Delete

Delete all flights to "**Ayn Halagim**".

DELETE

FROM Tickets

WHERE FlightId IN(

Select Id

FROM Flights

WHERE Destination LIKE 'Ayn Halagim')

DELETE

FROM Flights

WHERE Destination LIKE 'Ayn Halagim'

# Section 3. Querying (40 pts)

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

## The "Tr" Planes

Select all of the **planes,** which name contains "**tr**". Order them by **id** (ascending), **name** (ascending), **seats** (ascending) and **range** (ascending).

SELECT p.Id, p.Name, p.Seats, p.Range

FROM Planes as p

WHERE p.Name LIKE '%tr%'

ORDER BY p.Id ASC, p.Name ASC, p.Seats ASC, p.Range ASC

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Name** | **Seats** | **Range** |
| 31 | Trunyx cpp | 195 | 2653 |
| 86 | Yakitri | 321 | 1360 |
| 87 | Trilith | 223 | 4375 |
| … | … | … | … |

## Flight Profits

Select the total profit for each flight from database. Order them by **total price** (descending), **flight id** (ascending).

SELECT t.FlightId, Sum(t.Price) AS Price

FROM Tickets as t

RIGHT JOIN Flights as f

ON f.Id = t.FlightId

GROUP BY t.FlightId

ORDER BY SUM(t.Price) DESC, t.FlightId ASC

### Examples

|  |  |
| --- | --- |
| **FlightId** | **Price** |
| 58 | 828.43 |
| 43 | 819.84 |
| … | … |

## Passenger Trips

Select the **full name** of the passengers with their trips (**origin** - **destination**). Order them by **full name** (ascending), **origin** (ascending) and **destination** (ascending).

SELECT CONCAT(p.FirstName, ' ', p.LastName) AS [Full Name], f.Origin, f.Destination

FROM Passengers AS p

JOIN Tickets AS t

ON p.Id = t.PassengerId

JOIN Flights AS f

ON f.Id = t.FlightId

ORDER BY [Full Name] ASC, f.Origin ASC, f.Destination ASC

### Examples

|  |  |  |
| --- | --- | --- |
| **Full Name** | **Origin** | **Destination** |
| Adina Uvedale | Lawa-an | Hulei |
| Adolphe Juste | Boto | Pantubig |
| Adolphe Juste | Codrington | Kasiyan |
| … | … | .. |

## Non Adventures People

Select all people who don't have tickets. Select their **first name**, **last name** and **age** .Order them by **age** (descending), **first name** (ascending) and **last name** (ascending).

SELECT p.FirstName, p.LastName, p.Age

FROM Passengers AS p

LEFT JOIN Tickets AS t

ON t.PassengerId = p.Id

WHERE t.Id IS NULL

ORDER BY p.Age DESC, p.FirstName ASC, p.LastName ASC

### Examples

|  |  |  |
| --- | --- | --- |
| **First Name** | **Last Name** | **Age** |
| Felipa | Wabe | 89 |
| Darius | Ellissen | 87 |
| Eleen | Ummfrey | 86 |
| … | … | .. |

## Full Info

Select all passengers who have **trips**. Select their **full name** (first name – last name), **plane name**, trip (in format {**origin**} - {**destination**}) and luggage type. Order the results by **full name** (ascending), **name** (ascending), **origin** (ascending), **destination** (ascending) and **luggage type** (ascending).

SELECT CONCAT(p.FirstName, ' ', p.LastName) AS [Full Name], pl.Name AS [Plane Name], CONCAT(f.Origin, ' - ', f.Destination), lt.Type AS [Luggage Type]

FROM Passengers AS p

JOIN Tickets AS t

ON p.Id = t.PassengerId

JOIN Flights AS f

ON f.Id = t.FlightId

JOIN Planes AS pl

ON pl.Id = f.PlaneId

JOIN Luggages AS l

ON l.Id = t.LuggageId

JOIN LuggageTypes AS lt

ON lt.Id = l.LuggageTypeId

ORDER BY [Full Name] ASC, pl.Name ASC, f.Origin ASC, f.Destination ASC, lt.Type ASC

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Full Name** | **Plane Name** | **Trip** | **Luggage Type** |
| Adina Uvedale | Feedspan | Lawa-an - Hulei | Wheeled Business Case |
| Adolphe Juste | Babbleopia | Usagara - Ikhtiman | Upright Luggage |
| Adolphe Juste | Feednation | Le Mans - Grazhdanka | Duffel Bag |
| … | … | … | … |

## PSP

Select **all planes** with their **name**, **seats count** and **passengers count**. Order the results by **passengers count** (descending), **plane name** (ascending) and **seats** (ascending)

SELECT p.Name, p.Seats, COUNT(t.PassengerId)

FROM Planes AS p

LEFT JOIN Flights AS fl

ON p.Id = fl.PlaneId

LEFT JOIN Tickets AS t

ON t.FlightId = fl.Id

GROUP BY p.Name, p.Seats

ORDER BY COUNT(t.PassengerId) DESC, p.Name ASC, p.Seats ASC

### Examples

|  |  |  |
| --- | --- | --- |
| **Name** | **Seats** | **Passengers Count** |
| Jabberbean | 56 | 6 |
| Youbridge | 159 | 5 |
| Yoveo | 247 | 5 |
| … | … | … |

# Section 4. Programmability (20 pts)

## Vacation

Create a **user defined function**, named **udf\_CalculateTickets(@origin, @destination, @peopleCount)** that receives an origin (town name), destination (town name) and people count.

The function must return the total price in format "**Total price {price}**"

* If people count is less or equal to zero return – "**Invalid people count!**"
* If flight is invalid return – "**Invalid flight!**"

CREATE FUNCTION udf\_CalculateTickets(@origin NVARCHAR(MAX), @destination NVARCHAR(MAX), @peopleCount INT)

RETURNS NVARCHAR(MAX)

AS

BEGIN

DECLARE @doesFlightExist INT = (SELECT f.Id

FROM Flights AS f

WHERE @origin = f.Origin AND @destination = f.Destination)

IF(@peopleCount <= 0)

BEGIN

RETURN 'Invalid people count!'

END

IF(@doesFlightExist IS NULL)

BEGIN

RETURN 'Invalid flight!'

END

DECLARE @price DECIMAL(18,2) = (SELECT t.Price

FROM Tickets AS t

JOIN Flights AS f

ON f.Id = t.FlightId

WHERE t.FlightId = @doesFlightExist)

RETURN CONCAT('Total price ', @peopleCount\*@price)

END

### Example:

|  |
| --- |
| **Query** |
| **SELECT** **dbo.udf\_CalculateTickets**(**'Kolyshley'**,**'Rancabolang'**, **33**) |
| **Output** |
| **Total price 2419.89** |

|  |
| --- |
| **Query** |
| **SELECT** **dbo.udf\_CalculateTickets**(**'Kolyshley'**,**'Rancabolang'**, **-1**) |
| **Output** |
| **Invalid people count!** |

|  |
| --- |
| **Query** |
| **SELECT** **dbo.udf\_CalculateTickets**(**'Invalid'**,**'Rancabolang'**, **33**) |
| **Output** |
| **Invalid flight!** |

## Wrong Data

Create a **user defined stored procedure**, named **usp\_CancelFlights**  
The procedure must cancel all flights on which the arrival time is before the departure time. Cancel means you need to leave the departure and arrival time empty.

CREATE PROCEDURE usp\_CancelFlights

AS

BEGIN

UPDATE Flights

SET ArrivalTime = NULL, DepartureTime = NULL

WHERE ArrivalTime > DepartureTime

END

### Example:

|  |
| --- |
| **Query** |
| **EXEC** **usp\_CancelFlights** |
| **Output** |
| (49 rows affected) |