# Database Basics MS SQL Exam – 16 Apr 2019

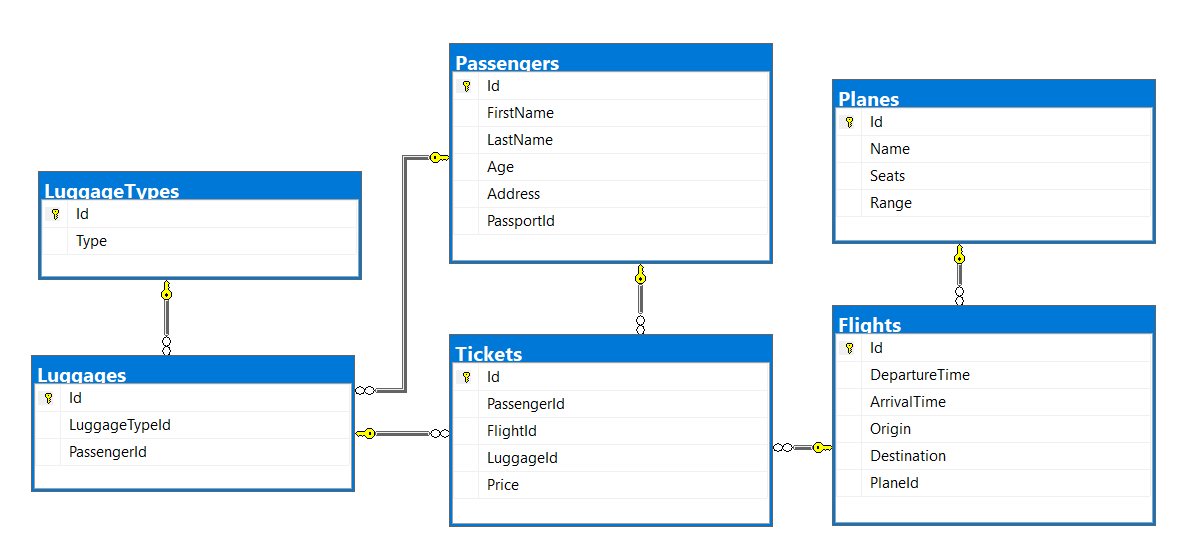
Exam problems for the [“Database Basics” course @ SoftUni](https://softuni.bg/courses/databases-basics-ms-sql-server).

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 |

**Solution:**

CREATE TABLE Planes

(

Id INT PRIMARY KEY IDENTITY,

[Name] VARCHAR(30) NOT NULL,

Seats INT NOT NULL,

[Range] INT NOT NULL

)

CREATE TABLE Flights

(

Id INT PRIMARY KEY IDENTITY,

DepartureTime DATETIME,

ArrivalTime DATETIME,

Origin VARCHAR(50) NOT NULL,

Destination VARCHAR(50) NOT NULL,

PlaneID INT FOREIGN KEY REFERENCES Planes(Id) NOT NULL

)

CREATE TABLE Passengers

(

Id INT PRIMARY KEY IDENTITY,

FirstName VARCHAR(30) NOT NULL,

LastName VARCHAR(30) NOT NULL,

Age INT NOT NULL,

[Address] VARCHAR(30) NOT NULL,

PassportId CHAR(11) NOT NULL

)

CREATE TABLE LuggageTypes

(

Id INT PRIMARY KEY IDENTITY,

[Type] VARCHAR(30) NOT NULL

)

CREATE TABLE Luggages

(

Id INT PRIMARY KEY IDENTITY,

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,

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(15,2) NOT NULL

)

## Database Design

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

# 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**.

**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 |

**Solution:**

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')

**Luggage Types**

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

## Update

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

**Solution:**

UPDATE Tickets

SET Price = Price\* 1.13

WHERE FlightId = 41

## Delete

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

**Solution:**

DELETE FROM Tickets

WHERE FlightId = 30

DELETE FROM Flights

WHERE Destination = '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**).**

## Trips

Select all **flights** from the database. Order them by **origin** (ascending) and **destination** (ascending).

**Solution:**

SELECT Origin, Destination FROM Flights

ORDER BY Origin, Destination

### Examples

|  |  |
| --- | --- |
| **Origin** | **Destination** |
| Abelheira | Sabanitas |
| Adirejo | Koblain |
| Alfena | Makariv |
| Aubagne | Kitahama |
| … | … |

## The "Tr" Planes

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

**Solution:**

SELECT \* FROM Planes

WHERE [Name] LIKE '%TR%'

ORDER BY Id, [Name], Seats, Range

### 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).

**Solution:**

SELECT t.FlightId, SUM(t.Price) AS Price FROM Tickets AS t

JOIN Passengers AS p ON t.PassengerId = p.Id

JOIN Flights AS f ON t.FlightId = f.Id

GROUP BY t.FlightId

ORDER BY Price DESC, t.FlightId

### Examples

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

## Passengers and Prices

Select top **10** records from passengers along with the price for their tickets. Order them by **price** (descending), **first name** (ascending) and **last name** (ascending).  
**Solution:**

SELECT TOP (10) p.FirstName, p.LastName, t.Price FROM Passengers AS p

JOIN Tickets AS t ON t.PassengerId = p.Id

ORDER BY t.Price DESC,p.FirstName, p.LastName

### Examples

|  |  |  |
| --- | --- | --- |
| **FirstName** | **LastName** | **Price** |
| Brittne | Leggin | 447.82 |
| Adolphe | Juste | 440.12 |
| Rudyard | Kaveney | 439.96 |
| … | … | … |

## Most Used Luggage's

**Solution:**

SELECT lt.[Type], COUNT(l.LuggageTypeId) AS MostUsedLuggage FROM LuggageTypes AS lt

JOIN Luggages AS l ON l.LuggageTypeId = lt.Id

GROUP BY lt.[Type], l.LuggageTypeId

ORDER BY COUNT(l.LuggageTypeId) DESC, lt.[Type]

### Examples

Select luggage type and how many times was used by **persons**. Sort by **count** (descending) and **luggage type** (ascending).

|  |  |
| --- | --- |
| **Type** | **MostUsedLuggage** |
| Garment Bag | 19 |
| Wheeled Business Case | 19 |
| Duffel Bag | 16 |
| … | .. |

## 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).

**Solution:**

SELECT CONCAT (p.FirstName, ' ', p.LastName) AS FullName, f.Origin, f.Destination FROM Tickets AS t

JOIN Passengers AS p ON t.PassengerId = p.Id

JOIN Flights AS f ON t.FlightId = f.Id

ORDER BY FullName, Origin, Destination

### 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).

**Solution:**

SELECT p.FirstName, p.LastName, p.Age FROM Passengers AS p

LEFT JOIN Tickets AS t ON p.Id = t.PassengerId

WHERE t.PassengerId IS NULL

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

### Examples

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

## Lost Luggage's

Select **all** **passengers** who **don't** have luggage's. Select their **passport id** and **address**. Order the results by **passport id** (ascending) and **address** (ascending).

**Solution:**

SELECT p.PassportId, p.Address FROM Luggages AS l

RIGHT JOIN Passengers AS p ON l.PassengerId = p.Id

WHERE l.Id IS NULL

ORDER BY p.PassportId, p.Address

### Examples

|  |  |
| --- | --- |
| **Passport Id** | **Address** |
| 105-40-7273 | 4 Haas Park |
| 135-11-2922 | 435 Marquette Terrace |
| 165-12-7011 | 2056 Kedzie Pass |
| … | .. |

## Count of Trips

Select **all** **passengers** and their count of trips. Select **the first name**, **last name** and **count of trips**. Order the results by **total trips** (descending), **first name** (ascending) and **last name** (ascending).

**Solution:**

SELECT p.FirstName, p.LastName, COUNT(t.PassengerId) AS [Total Trips] FROM Tickets AS t

RIGHT JOIN Passengers AS p ON t.PassengerId = p.Id

GROUP BY p.FirstName, p.LastName

ORDER BY [Total Trips] DESC, p.FirstName, p.LastName

### Examples

|  |  |  |
| --- | --- | --- |
| **First Name** | **Last Name** | **Total Trips** |
| Adolphe | Juste | 5 |
| Neddie | Hugill | 4 |
| Ashley | Peterkin | 3 |
| … | … | .. |

## 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).

**Solution:**

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

pln.[Name] AS [Plane Name],

CONCAT( f.Origin, ' - ' ,f.Destination) AS Trip,

lgtp.[Type] AS [Luggage Type]

FROM Passengers AS p

JOIN Tickets AS t ON t.PassengerId = p.Id

JOIN Flights AS f ON t.FlightId = f.Id

JOIN Planes AS pln ON f.PlaneID = pln.Id

JOIN Luggages AS lg ON t.LuggageId = lg.Id

JOIN LuggageTypes AS lgtp ON lg.LuggageTypeId = lgtp.Id

ORDER BY [Full Name], [Plane Name], f.Origin, f.Destination, [Luggage Type]

### 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 |
| … | … | … | … |

## Most Expensive Trips

Select **all passengers** who have **flights**. Select their **first name**, **last name**, **destination** and **price for the ticket**. Take only the ticket with highest price for user. Order the results by **price** (descending), **first name** (ascending), **last name** (ascending) and **destination** (ascending).

**Solution:**

SELECT k.FirstName, k.LastName, k.Destination, k.Price FROM (

SELECT DISTINCT p.FirstName,

p.LastName,

f.Destination,

t.Price,

DENSE\_RANK()OVER(PARTITION BY p.FirstName

ORDER BY t.Price DESC)

AS Ranking

FROM Passengers AS p

JOIN Tickets AS t ON t.PassengerId = p.Id

JOIN Flights AS f ON t.FlightId = f.Id) AS k

WHERE Ranking = 1

ORDER BY Price DESC, k.FirstName, k.LastName, k.Destination

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **First Name** | **Last Name** | **Destination** | **Price** |
| Brittne | Leggin | Quitilipi | 447.82 |
| Adolphe | Juste | Pantubig | 440.12 |
| Rudyard | Kaveney | Kynopiastes | 439.96 |
| … | … | … | … |

## Destinations Info

Select **all destinations** and trips count to them. Sort the result by **trips count** (descending) and **destination name** (ascending).

**Solution:**

SELECT f.Destination, COUNT(p.Id) AS FilesCount FROM Flights AS f

LEFT JOIN Tickets AS t ON t.FlightId = f.Id

LEFT JOIN Passengers AS p ON t.PassengerId = p.Id

GROUP BY f.Destination

ORDER BY FilesCount DESC, f.Destination

### Examples

|  |  |
| --- | --- |
| **Destination** | **FilesCount** |
| Daniwato | 3 |
| Kobenhavn | 3 |
| San Lorenzo | 3 |
| … | … |

## 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)

**Solution:**

SELECT p.[Name],

p.Seats,

ISNULL((COUNT(t.PassengerId)), 0) AS [Passengers Count] FROM Planes AS p

LEFT JOIN Flights AS f ON f.PlaneID = p.Id

LEFT JOIN Tickets AS t ON t.FlightId = f.Id

GROUP BY p.[Name], p.Seats

ORDER BY [Passengers Count] DESC , p.[Name], p.Seats

### 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!**"

**Solution:**

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

RETURNS NVARCHAR(MAX)

AS

BEGIN

DECLARE @output NVARCHAR(MAX)

DECLARE @amountOfTrip DECIMAL(18,2)

IF (@peopleCount < = 0 )

BEGIN

SET @output = 'Invalid people count!'

END

ELSE IF

( EXISTS

(SELECT \* FROM Flights WHERE Origin = @origin AND Destination = @destination )

)

BEGIN

SET @amountOfTrip = @peopleCount \*

(

SELECT t.Price FROM Flights AS f

JOIN Tickets AS t ON f.Id = t.FlightId

WHERE Origin = @origin AND Destination = @destination

)

SET @output = CONCAT('Total price ', CAST(@amountOfTrip AS NVARCHAR))

END

ELSE IF

( NOT EXISTS

( SELECT \* FROM Flights WHERE Origin = @origin AND Destination = @destination)

)

BEGIN

SET @output = 'Invalid flight!'

END

RETURN @output

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.

**Solution:**

CREATE PROCEDURE usp\_CancelFlights

AS

BEGIN

UPDATE Flights

SET DepartureTime = NULL , ArrivalTime = NULL

WHERE DATEDIFF(DAY, DepartureTime , ArrivalTime ) >= 0

END

### Example:

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

## Deleted Planes

Create a new table **"DeletedPlanes**" with columns **(Id,Name,Seats, Range)**. Create a **trigger**, which fires when planes are deleted. After deleting the planes, **insert all of the data into the new table** "**DeletedPlanes"**.

Note: Submit only your **CREATE TRIGGER** statement!

### Example usage:

|  |
| --- |
| **Query** |
| **DELETE** **Tickets**  **WHERE** **FlightId** **IN** (**SELECT** **Id** **FROM** **Flights** **WHERE** **PlaneId** = **8**) **DELETE** **FROM** **Flights**  **WHERE** **PlaneId** **= 8**  **DELETE** **FROM** **Planes**  **WHERE** **Id** **= 8** |
| **Response** |
| **(1 rows affected)**  **(1 rows affected)**  **(1 rows affected)**  **(1 rows affected)** |

**Solution:**

CREATE TRIGGER tr\_DeletedPlanes ON Planes AFTER DELETE

AS

INSERT INTO DeletedPlanes (Id, [Name],Seats, [Range])

(SELECT Id, [Name],Seats, [Range] FROM deleted)