*PROJECT*

*Theme:* A Simple Airway Database design using Microsoft SQL Server Environment

Index

[Project Requirements: 2](#_Toc525630049)

[About Project: 2](#_Toc525630050)

[T-SQL Codes; 6](#_Toc525630051)

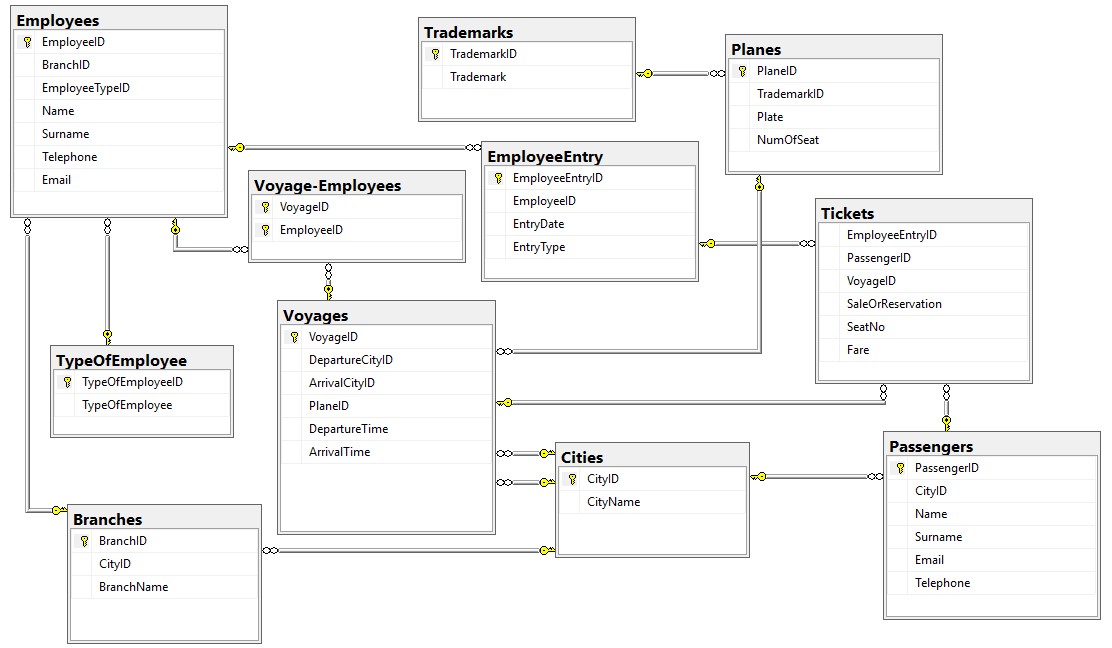
# **Project Requirements:**

* Create a database logical structure (4 or more tables, depends on topic specification).
* Follow table normalization rules to avoid data redundancy in tables.
* Table relations have to be created.
* Submit data to tables, as many records as is needed for evaluation of your database design.
* Create 15 unique queries, where 5 of them have to select records multiple tables.
* Queries have to deal with: select (joins, aggregation, where condition, etc.), update, delete, and insert commands.

# About Project:

The Airway project is a simulation of an Airway database system that is aiming a database model in Microsoft SQL Server using T-SQL commands. Database contains 10 tables which are related each other like employees ,voyages,passengers etc.

2.1. Database diagram and declaration and creation of tables:



2.1 Database Diagram

1. Table ‘Passengers’

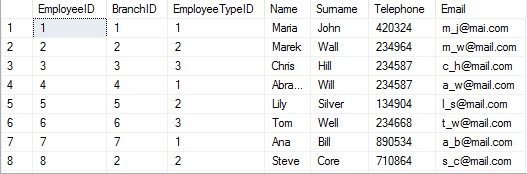
It contains basic knowledges about registered passengers like name,surname.telephone number etc.



2.2 Table ‘Passengers’

b) Table ‘Employees’

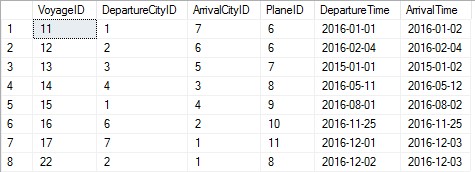
It contains knowledges about all general employees.



2.3 Table ‘Employees

c) Table ‘Voyages’

It contains knowledges about a voyage plan like departure city, plane,times etc.



2.4 Table ‘Voyages’

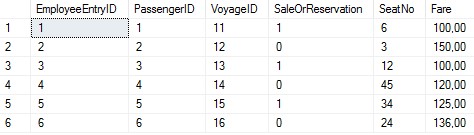
d) Tables ‘Cities’, ‘Tickets’, ‘Trademarks’, ‘Planes’, ‘Branches’ ,‘EmployeeEntry’, ‘TypeOfEmployee’,’Voyage-Employee’

d.1 ) Cities ; All cities which are available for flying.



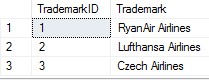
2.5 Table ‘Cities’

d.2)Tickets ; General informations about passengers and voyages.



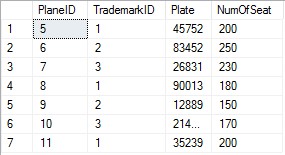
2.6 Table ‘Tickets’

d.3)Trademarks; All available airline companies.



2.7 Table ‘Trademarks’

d.4)Planes;



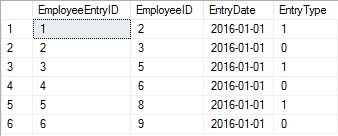
2.8 Table ‘Planes’

d.5)Branches; Airline companys’ branches in cities.



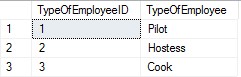
2.9 Table ‘Branches’

d.6)EmployeeEntry; Entry records of employees.



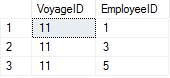
2.10 Table ‘Employee Entries

d.7)TypeOfEmployee; Types of employees in an Airway.



2.11 Table ‘Types of Employees’

d.8)Voyage-Employees;Many to many table between voyages and employees.



2.12 Table ‘Voyage-Employees’

# T-SQL Codes;

Creating database;

CREATE DATABASE Airlines

Creating tables;

USE Airlines

CREATE TABLE Employees(

EmployeeID int PRIMARY KEY NOT NULL IDENTITY,

BranchID int NULL,

EmployeeTypeID int NULL,

Name varchar(20) NULL,

Surname varchar(20) NULL,

Telephone int NULL,

Email nvarchar(30) NULL CONSTRAINT UC\_Employees\_Email UNIQUE (Email)

)

CREATE TABLE Branches(

BranchID int PRIMARY KEY NOT NULL IDENTITY,

CityID int NULL,

BranchName varchar(20)

)

CREATE TABLE Planes (

PlaneID int PRIMARY KEY NOT NULL IDENTITY,

TrademarkID int NULL,

Plate int NULL,

NumOfSeat int NULL,

)

CREATE TABLE Cities(

CityID int PRIMARY KEY NOT NULL IDENTITY,

CityName varchar(20) NULL

)

CREATE TABLE Trademarks (

TrademarkID int PRIMARY KEY NOT NULL IDENTITY,

Trademark nvarchar(20) NULL

)

CREATE TABLE TypeOfEmployee(

TypeOfEmployeeID INT PRIMARY KEY NOT NULL IDENTITY,

TypeOfEmployee varchar(20) NULL

)

CREATE TABLE EmployeeEntry(

EmployeeEntryID INT PRIMARY KEY NOT NULL IDENTITY,

EmployeeID int NULL,

EntryDate date NULL,

EntryType nvarchar(20) NULL

)

CREATE TABLE Passenger(

PassengerID INT PRIMARY KEY NOT NULL IDENTITY,

CityID int NULL,

Name varchar(20) NULL,

Surname varchar(20) NULL,

Email nvarchar(30) NULL CONSTRAINT UC\_Passenger\_Email UNIQUE (Email),

Telephone int NULL

)

CREATE TABLE Voyages (

VoyageID INT PRIMARY KEY NOT NULL IDENTITY,

DepartureCityID int NULL,

ArrivalCityID int NULL,

PlaneID int NULL,

DepartureTime time NULL,

ArrivalTime time NULL,

NumOfSeat int NULL

)

CREATE TABLE Tickets (

EmployeeEntryID int NULL,

PassengerID int NULL,

VoyageID int NULL,

SaleOrReservation bit NULL,

SeatNo tinyint NULL,

Fare money NULL )

CREATE TABLE Voyage-Employees

{

VoyageID int PRIMARY KEY NOT NULL,

EmployeeID int PRIMARY KEY NOT NULL

}

Creating Relationships between tables;

USE Airlines

ALTER TABLE Voyages

ADD CONSTRAINT FK\_Voyages\_Cities1 FOREIGN KEY (DepartureCityID)

REFERENCES Cities(CityID),

CONSTRAINT FK\_Voyages\_Cities2 FOREIGN KEY (ArrivalCityID)

REFERENCES Cities(CityID),

CONSTRAINT FK\_Voyages\_Planes FOREIGN KEY (PlaneID)

REFERENCES Planes(PlaneID) ON DELETE CASCADE,

ALTER TABLE Branches

ADD CONSTRAINT FK\_Branches\_Cities FOREIGN KEY (CityID)

REFERENCES Cities(CityID) ON DELETE CASCADE

ALTER TABLE Passenger

ADD CONSTRAINT FK\_Passenger\_Cities FOREIGN KEY (CityID)

REFERENCES Cities(CityID) ON DELETE CASCADE

ALTER TABLE Planes

ADD CONSTRAINT FK\_Planes\_Trademarks FOREIGN KEY (TrademarkID)

REFERENCES Trademarks(TrademarkID) ON DELETE CASCADE

ALTER TABLE Tickets

ADD CONSTRAINT FK\_Tickets\_EmployeeEntry FOREIGN KEY (EmployeeEntryID)

REFERENCES EmployeeEntry (EmployeeEntryID) ON DELETE CASCADE,

CONSTRAINT FK\_Tickets\_Passenger FOREIGN KEY (PassengerID)

REFERENCES Passenger(PassengerID) ON DELETE CASCADE,

CONSTRAINT FK\_Tickets\_Voyages FOREIGN KEY (VoyageID)

REFERENCES Voyages(VoyageID) ON DELETE CASCADE

ALTER TABLE Employees

ADD CONSTRAINT FK\_Employees\_Branches FOREIGN KEY (BranchID)

REFERENCES Branches(BranchID) ON DELETE CASCADE,

CONSTRAINT FK\_Employees\_TypeOfEmployee FOREIGN KEY (EmployeeTypeID)

REFERENCES TypeOfEmployee(TypeOfEmployeeID) ON DELETE CASCADE

ALTER TABLE EmployeeEntry

ADD CONSTRAINT FK\_EmployeeEntry\_Employees FOREIGN KEY (EmployeeID)

REFERENCES Employees(EmployeeID)

ALTER TABLE Voyage-Employees

ADD CONSTRAINT FK\_Voyage-Employees\_EmployeeID FOREIGN KEY (EmployeeID)

REFERENCES Employees(EmployeeID),

ADD CONSTRAINT FK\_Voyage-Employees\_VoyageID FOREIGN KEY (VoyageID)

REFERENCES Voyages(VoyageID)

Filling tables with data;

USE Airlines

INSERT INTO Trademarks

VALUES ('RyanAir Airlines'),('Lufthansa Airlines'),('Czech Airlines')

INSERT INTO Cities

VALUES ('Prague'),('Ostrava'),('Brno'),('Zlin'),('Pardubice'),('Karlovy Vary'),('Olomouc')

INSERT INTO TypeOfEmployee

VALUES ('Pilot'),('Hostess'),('Cooker')

INSERT INTO Passenger

VALUES (1,'John','William','j\_w@mail.com',420123123),(2,'Adam','Steward','a\_s@mail.com',420123987),

(3,'Maria','Hill','m\_h@mail.com',420456789),(4,'Bill','Brown','b\_b@mail.com',420543765),

(5,'Rose','Taylor','r\_t@mail.com',420264195),(6,'Johanna','Ben','y\_b@mail.com',420963172),

(7,'Josef','Stone','j\_s@mail.com',420200876)

INSERT INTO Branches

VALUES (1,'PragueBranch'),(2,'Ostrava Branch'),(3,'Brno Branch'),(4,'Zlin Branch'),(5,'Pardubice Branch'),

(6,'Karlovy Vary Branch'),(7,'Olomouc Branch')

INSERT INTO Planes

VALUES (1,45752,200),(2,83452,250),(3,26831,230),(1,90013,180),(2,12889,150),(3,214906,170),(1,35239,200)

INSERT INTO Employees

VALUES (1,1,'Maria','John',420324,'m\_j@mai.com'),(2,2,'Marek','Wall',234964,'m\_w@mail.com'),

(3,3,'Chris','Hill',234587,'c\_h@mail.com'),

(4,1,'Abraham','Will',234587,'a\_w@mail.com'),(5,2,'Lily','Silver',134904,'l\_s@mail.com'),

(6,3,'Tom','Well',234668,'t\_w@mail.com'),

(7,1,'Ana','Bill',890534,'a\_b@mail.com'),(2,2,'Steve','Core',710864,'s\_c@mail.com'),

(5,3,'Jousef','Brown',905123,'j\_b@mail.com')

INSERT INTO Voyages

VALUES (1,7,5,'2016-01-01','2016-01-02'),(2,6,6,'2016-01-01','2016-01-02'),(3,5,7,'2016-01-01','2016-01-02'),

(4,3,8,'2016-01-01','2016-01-02'),(5,4,9,'2016-01-01','2016-01-02'),

(6,2,10,'2016-01-01','2016-01-02'),(7,1,11,'2016-01-01','2016-01-02')

INSERT INTO EmployeeEntry

VALUES (2,'2016-01-01','True'),(3,'2016-01-01','False'),(5,'2016-01-01','True'),

(6,'2016-01-01','False'),(8,'2016-01-01','True'),(9,'2016-01-01','False')

INSERT INTO Tickets

Values (1,1,11,'True',6,100),(2,2,12,'False',3,150),(3,3,13,'True',12,100),(4,4,14,'False',45,120),

(5,5,15,'True',34,125),(6,6,16,'False',24,136)

2.2 Creation of stored procedures;

I-Adding data;

With this stored procedure we can easily add data into ‘Passengers’ table.

CREATE PROCEDURE InsertPassenger

(@CityID int,@Name varchar(20),@Surname varchar(20),@Email nvarchar(30),@Telephone int)

AS

BEGIN

IF NOT EXISTS (SELECT \* FROM Passenger WHERE Email=@Email)

INSERT INTO Passenger (CityID,Name,Surname,Email,Telephone) VALUES

(@CityID,@Name,@Surname,@Email,@Telephone)

ELSE PRINT 'This E-mail adress have already taken.Please try another one !'

END

EXEC InsertPassenger @CityID=1,@Name='Michael',@Surname='Green',@Email='j\_w@mail.com',@Telephone=324646

EXEC InsertPassenger @CityID=1,@Name='Michael',@Surname='Green',@Email='m\_g@mail.com',@Telephone=324646

II-Deleting data;

CREATE PROCEDURE DeletePassenger

(@PassengerID int)

AS

BEGIN

IF EXISTS (SELECT \* FROM Passenger WHERE PassengerID=@PassengerrecordsID)

DELETE FROM Passenger WHERE PassengerID=@PassengerID

END

EXEC DeletePassenger @PassengerID= 12

III-Listing all records;

CREATE PROCEDURE ListPassengers

AS

BEGIN

SELECT \* FROM Passenger

END

EXEC ListPassengers

IV-Detail about a passenger;

CREATE PROCEDURE DetailPassenger

(@PassengerID int)

AS

BEGIN

IF EXISTS (SELECT \* FROM Passenger WHERE PassengerID=@PassengerID )

SELECT \* FROM Passenger WHERE PassengerID=@PassengerID

END

EXEC DetailPassenger @PassengerID=3

V-Updating a passenger;

CREATE PROCEDURE UpdatePassenger

(@PassengerID int,@CityID int,@Name varchar(20),@Surname varchar(20),@Email nvarchar(30),@Telephone int)

AS

BEGIN

UPDATE Passenger SET CityID=@CityID ,Name=@Name,Surname=@Surname,

Email=@Email,Telephone=@Telephone WHERE PassengerID=@PassengerID

END

EXEC UpdatePassenger @PassengerID=1,@CityID=2,@Name='John',@Surname='William',@Email='j\_w@mail.com',

@Telephone=2135436

2.3 Creation of triggers;

Triggers are [procedural code](https://en.wikipedia.org/wiki/Procedural_code)s that is automatically executed in response to certain [events](https://en.wikipedia.org/wiki/Event_(computing)) on a particular [table](https://en.wikipedia.org/wiki/Table_(database)).Trigger is mostly used for maintaining the [integrity](https://en.wikipedia.org/wiki/Database_integrity) of the information on the database i.e. if we make some changes in a table then we can change the other related tables’ information using trigger.In this project it is used for changes after crud operations

I-After insert;

CREATE TRIGGER Add\_Voyage

ON Voyages

AFTER INSERT

AS

DECLARE @NumOfSeat int,@PlaneID int,@VoyageID int

SELECT @PlaneID=PlaneID, @VoyageID=VoyageID from inserted

UPDATE Voyages SET NumOfSeat = (select NumOfSeat from Planes where PlaneID=@PlaneID) where VoyageID=@VoyageID

INSERT INTO Voyages

VALUES (1,2,5,'2016-02-02','2016-01-01')

Note : After inserting new voyage to ‘Voyages’ table then the number of seats of plane which will be used is being copied to the voyage information.

II-When a passenger taking ticket;

CREATE TRIGGER Buying\_ticket ON Tickets

AFTER INSERT

AS

BEGIN

DECLARE @VoyageID int

SELECT @VoyageID=VoyageID FROM inserted

UPDATE Voyages SET NumOfSeat -=1 WHERE VoyageID=@VoyageID

END

III-When a passenger cancelling ticket;

CREATE TRIGGER Cancelling\_Ticket ON Tickets

AFTER DELETE

AS

BEGIN

DECLARE @VoyageID int

SELECT @VoyageID=VoyageID FROM deleted

UPDATE Voyages SET NumOfSeat +=1 WHERE VoyageID=@VoyageID

END