**ASSIGNMENT**

Student name: ĐỖ TIẾN THÀNH | HE161864

Student name: TRẦN THỊ TÚ ANH | HE181730

Student name: PHẠM THỊ HƯƠNG GIANG | HE182352

INSTRUCTOR: BUI THI THU TRANG

DBI202 – DATABASE SYSTEM OF FLIGHTS

SE18100

July 16, 2024

# TABLE OF CONTENTS

## INTRODUCE THE PROBLEM…………………………………………………………………………………………….

### Describe the problem……………………………………………………………………………………………………….

### Management objectives…………………………………………………………………………………………………..

## entity – relationship – er…………………………………………………………………………………………

### difinITION entity – attributE…………………………………………………………………………………………..

### set-up entity – relationship…………………………………………………………………………………………….

## data dictionary………………………………………………………………………………………………………….

### DATABASE AND TABLE……………………………………………………………………………………….………………….

## ENTITY RELATIONSHIP DIAGRAM (ERD)…………………………………………………………………………

### AIR PORTS ………………………………………………………………………………………………………………………….

### FLIGHT\_ROUTES…………………………………………………………………………………………………………………..

### FLIGHT\_CREW………………………………………………………………………………………………………………………

### EMPLOYEES……………………………………………………………………………………………………………………………

### PASSENGERS…………………………………………………………………………………………………………………………

### TICKETS………………………………………………………………………………………………………………………………..

### 7) FLIGHTS………………………………………………………………………………………………………………………………..

## SQL COMMAND

### QUERY USING ORDER BY………………………………………………………………………………………………………

### QUERY USING INNER JOIN……………………………………………………………………………………………………

### QUERY USING AGGREGATE FUNCTIONS…………………………………………………………………………………

### QUERY USING THE GROUP BY AND HAVING CLAUSES……………………………………………………………

### QUERY THAT USES A SUB-QUERY AS A RELATION………………………………………………………………….

### QUERY THAT USES PARTIAL MATCHING IN THE WHERE CLAUSE…………………………………………….

### QUERY THAT USES A SELF-JOIN……………………………………………………………………………………………..

### STORE PROCEDURE……………………………………………………………………………………………………………….

### trigger……………………………………………………………………………………………………………………………….

# INTRODUCE THE PROBLEM

## Describe the problem

In modern aviation management, coordinating and overseeing flights, crew, and passengers has become increasingly complex. Effective flight management is crucial for ensuring safety, efficiency, and customer satisfaction. Several challenges persist in this domain:

* **Flight Scheduling**: Airlines must ensure optimal scheduling of flights to maximize aircraft utilization and minimize delays. This includes managing takeoff and landing slots, coordinating with air traffic control, and dealing with unforeseen circumstances such as weather disruptions.
* **Crew Management**: Each flight requires a specific crew, including pilots, co-pilots, and flight attendants. Managing their schedules, ensuring they meet regulatory rest requirements, and handling last-minute changes can be challenging.
* **Passenger Information**: Accurate and timely information about passengers is essential for safety and service quality. This includes managing passenger lists, seating arrangements, special requirements (such as meals and assistance), and ensuring compliance with security regulations.
* **Data Management**: The absence of a centralized database system to manage the vast amount of data generated by flight operations can lead to inefficiencies and errors. This includes data on flights, crew, passengers, and maintenance activities

**Request:**

* Check the number of tickets issued with a specific TicketCode.
* Retrieve the departure time of a flight.
* Determine the destination airport for a specific flight.
* Find out the airfare price per trip.

## Management objectives

* **Optimize Flight Scheduling**: Ensure efficient utilization of aircraft and minimize delays by optimizing flight schedules.
* **Efficient Crew Management**: Manage crew schedules to meet regulatory requirements and handle last-minute changes effectively.
* **Accurate Passenger Information Management**: Maintain accurate and timely records of passenger information to enhance safety and service quality.
* **Effective Baggage Handling**: Implement robust tracking systems to manage baggage efficiently and prevent loss.
*  **Centralized Data Management**: Develop a centralized database system to manage all data related to flight operations, crew, passengers, baggage, and maintenance activities.

**Important output**

* Revenue and Cost Reports
* Available Seats and Flights
* Total Fines Collectect

# entity – relationship – er

## difinITION entity – attributE

Base on the problem description and management objectives, we can present several entities and attributes of the entity as follow:

* Passenger: **PassengerID**, Passenger Name, Date Of Birth, Email, Gender, Nationality.
* Ticket: **TicketID**, TicketCode, Price, Class, Seat, PassengerID.
* Flight Route: **FlightRouteID**, Flight Time, Departing Airport, Arriving Airport.
* Flight: **FlightID**, FlightCode, Airline, Departure Time, Arrival Time, Airplane Type, FlightRouteID, FlightCrewID.
* Flight Crew: **FlightCrewID**, Flight Crew Code.
* Employees: **EmployeeID**, EmployeeName, FlightCrewID.
* Airport: **AirportID**, AirportCode, Airport Name, Country.

## set-up entity – relationship

\* Some symbols used in the model

**Attibute**

|  |  |
| --- | --- |
| * Key / identifier attribute |  |
| * Attribute description / description | **ENTITY**  Attribute |
| * Entity | **WEAK ENTIRY** |
| * Weak entity | *Relationship* |
| * Relationship |  |
| * Connectivity (force) = 1 |  |
| * Connectivity = N |  |

A diagram of flight routes

Description automatically generated

# data dictionary

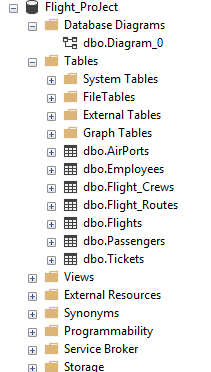
Just for example on some tables (other table are similar, you have to define all the tables in your database). Note: to run the query you have to define the table 1 first then go to the side tables much

## database and table

### cREATE DATABASE **PROJECTDBI202**

--create database Flight\_Project

CREATE DATABASE Flight\_Project



### **Create table Airports**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| AirportId | Int |  |  | Pk,unique |
| AirportCode | Char(10) |  |  | Not null |
| AirportName | Nvarchar(100) |  |  | Not null |
| Country | Nvarchar(100) |  |  | Not null |
| City | Nvarchar(100) |  |  | Not null |

***Code:***

--create table Airports

create table AirPorts(

AirportId int Primary key,

AirportCode char(10) not null,

AirportName nvarchar(100) not null,

Country Nvarchar(100) not null,

City nvarchar(100) not null

)

***Example:***

insert into Airports values (1,'DEN',N'Denver International Airport',N'United States',N'Denver')

insert into Airports values (2,'ATL',N'Atlanta Hartsfield-Jackson Airport',N'United States',N'Atlanta')

insert into Airports values (3,'IST',N'Istanbul International Airport',N'Turkey',N'Istanbul')

insert into Airports values (4,'DFW',N'Dallas / Fort Worth International Airport',N'United States',N'Dallas')

| AirportId | AirportCode | AirportName | Country | City |
| --- | --- | --- | --- | --- |
| 1 | DEN | Denver International Airport | United States | Denver |
| 2 | ATL | Atlanta Hartsfield-Jackson Airport | United States | Atlanta |
| 3 | IST | Istanbul International Airport | Turkey | Istanbul |
| 4 | DFW | Dallas / Fort Worth International Airport | United States | Dallas |

### **Create table Flight\_Routes**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| FlightRouteId | int |  |  | PK, Not null |
| Departing\_airport | int |  |  | FK reference Airports(AirportId) |
| Arriving\_airport | int |  |  | FK reference Airports(AirportId) |

***Code:***

--create table Flight\_Routes

Create table Flight\_Routes(

FlightRouteId int primary key,

Departing\_airport int foreign key references AirPorts(AirportID),

Arriving\_airport int foreign key references AirPorts(AirportID)

)

***Example:***

insert into Flight\_Routes values (1,12,9)

insert into Flight\_Routes values (2,18,17)

insert into Flight\_Routes values (3,5,9)

insert into Flight\_Routes values (4,19,14)

| FlightRouteId | Departing\_Airport | Arrving\_Airport |
| --- | --- | --- |
| 1 | 12 | 9 |
| 2 | 18 | 17 |
| 3 | 5 | 9 |
| 4 | 19 | 14 |

### **Create table Flight\_Crews**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| FlightCrewId | int |  |  | Primary key |
| FlightCrewCode | Char(10) |  |  | Not null |

***Code:***

--create table Flight\_Crews

Create table Flight\_Crews(

FlightCrewId int primary key,

FlightCrewCode char(10) not null

)

***Example:***

insert into Flight\_Crews values (1,'Group1')

insert into Flight\_Crews values (2,'Group2')

insert into Flight\_Crews values (3,'Group3')

insert into Flight\_Crews values (4,'Group4')

| FlightCrewId | FlightCrewCode |
| --- | --- |
| 1 | Group1 |
| 2 | Group2 |
| 3 | Group3 |
| 4 | Group4 |

### Create TABLE **Employees**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| EmployeeId | int |  |  | PK |
| EmployeeName | Nvarchar(100) |  |  | Not null |
| Email | Nvarchar(100) |  |  | Not null |
| Gender | Char(1) |  | Gender in (‘M’,’F’) | Not null |
| Supervisor | int |  |  | Fk references Employees(EmployeeID) |
| Role | Nvarchar(100) |  |  | Not null |
| FlightCrewId | int |  |  | Fk references Flight\_Crews(FlightCrewID) |

***Code:***

--create table Employees

Create table Employees(

EmployeeId int primary key,

EmployeeName nvarchar(100) not null,

Email nvarchar(100) not null,

Gender char(1) not null check(Gender in ('M','F')),

SuperVisor int foreign key references Employees(EmployeeId),

[Role] nvarchar(100) not null,

FlightCrewId int foreign key references Flight\_Crews(FlightCrewId)

)

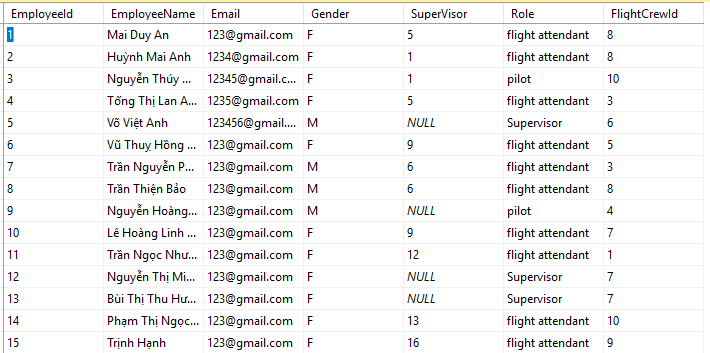
***Example:***

insert into Employees values(5,N'Võ Việt Anh',N'123456@gmail.com','M',null,N'Supervisor',6)

insert into Employees values(9,N'Nguyễn Hoàng Dũng',N'123@gmail.com','M',null,N'pilot',4)

insert into Employees values(12,N'Nguyễn Thị Minh Hưng',N'123@gmail.com','F',null,N'Supervisor',7)

insert into Employees values(13,N'Bùi Thị Thu Hương',N'123@gmail.com','F',null,N'Supervisor',7)

******

### Create TABLE **Flights**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| FlightId | int |  |  | PK |
| FlightCode | char(10) |  |  | Not null |
| Departure\_Time | Date |  |  | Not null |
| Arrival\_Time | Date |  |  | Not null |
| AirplaneType | Nvarchar(100) |  |  | Not null |
| Airline | Nvarchar(100) |  |  | Not null |
| FlightRouteId | int |  |  | FK references Flight\_Routes(FlightRouteId) |
| FlightCrewId | int |  |  | FK references Flight\_Crews(FlightCrewId) |
| Departing\_gate | Nvarchar(10) |  |  | Not null |
| Arriving\_gate | Nvarchar(10) |  |  | Not null |

***Code:***

--create table flights

Create table Flights(

FlightId int primary key,

FlightCode char(10) not null,

Departure\_Time Date not null,

Arrival\_Time Date not null,

AirplaneType nvarchar(100) not null,

Airline nvarchar(100) not null,

FlightRouteId int Foreign key references Flight\_Routes(FlightRouteId),

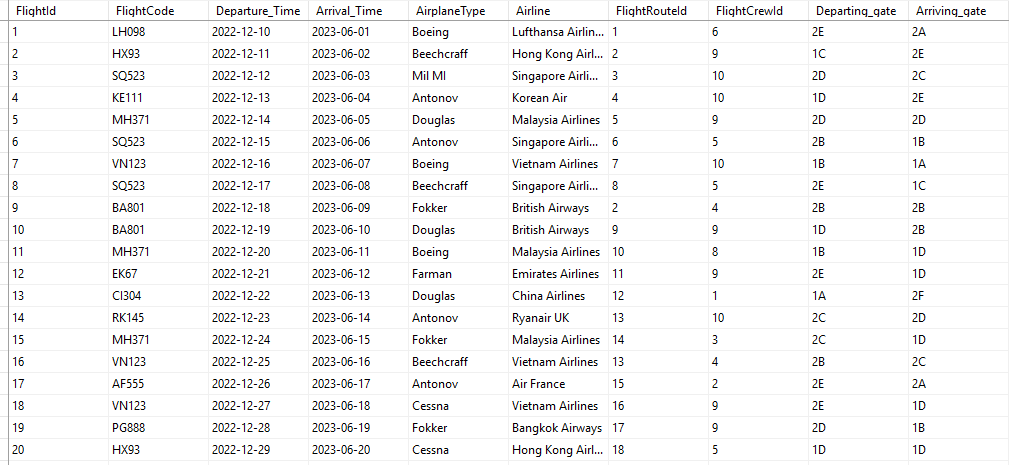
FlightCrewId int Foreign key references Flight\_Crews(FlightCrewID),

Departing\_gate nvarchar(10) not null,

Arriving\_gate nvarchar(10) not null,

)

***Example:***



### create table **Passengers**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| PassengerId | Int |  |  | PK, Not null |
| PasssengerName | Nvarchar(100) |  |  | Not null |
| DateOfBirth | Date |  |  | Not null |
| Email | Nvarchar(100) |  |  | Not null |
| Gender | Char(1) |  | Check Gender in (‘M’,’F’) | Not null |
| Nationality | Nvarchar(100) |  |  | Not null |

***Code:***

--create table passengers

Create table Passengers(

PassengerId int primary key,

PasssengerName nvarchar(100) not null,

DateOfBirth Date not null,

Email nvarchar(100) not null,

Gender char(1) not null check(Gender in ('M','F')),

Nationality nvarchar(100) not null

)

***Example:***

******

### create table **Tickets**

| Column Name | Data Type | Default | Check | Key/ Index/ Constraint |
| --- | --- | --- | --- | --- |
| TicketId | int |  |  | Pk |
| TicketCode | Char(10) |  |  | Not null |
| Seat | Nvarchar(50) |  |  | Not null |
| Class | Nvarchar(50) |  |  | Not null |
| PassengerId | Int |  |  | fk references Passengers(PassengerId) |
| FlightId | Int |  |  | fk references Flights(FlightId) |
| Price | float |  |  | Not null |

***Code:***

--create table Tickets

Create table Tickets(

TicketId int primary key,

TicketCode char(10) not null,

Seat nvarchar(50) not null,

Class nvarchar(50) not null,

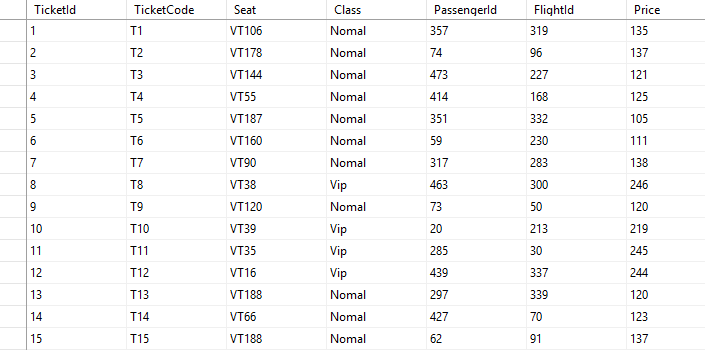
PassengerId int foreign key references Passengers(PassengerId),

FlightId int foreign key references Flights(FlightId),

Price float not null

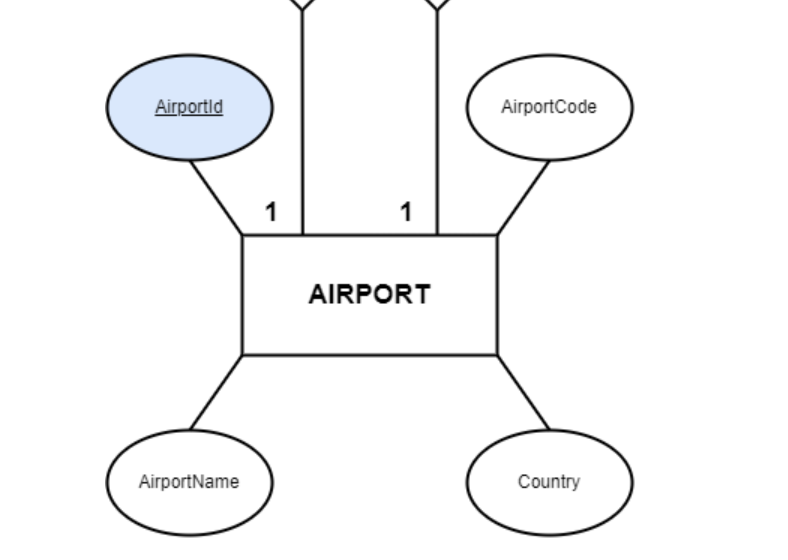
)

***Example:***



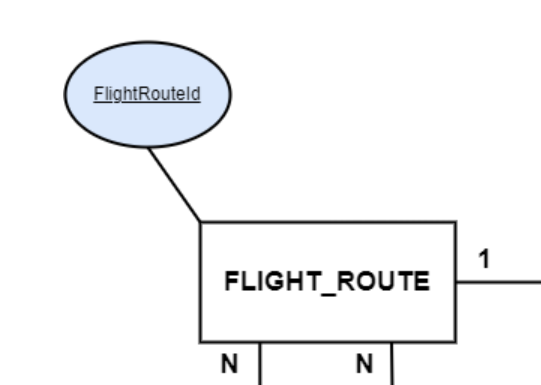
# IV. entity relationship diagram (erd)

## AIR PORTS



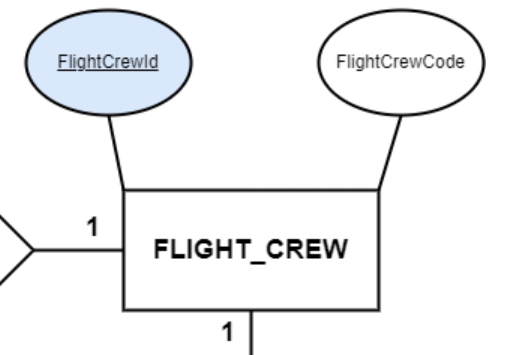
|  |  |
| --- | --- |
| The AirPorts entity contains information about each airport**. Attributes:** AirportId: The primary key for this entity. AirportCode: A unique code assigned to each airport. AirportName: The name of the airport. Country: The country where the airport is located. City: The city where the airport is located. |  |

## Flight\_Routes



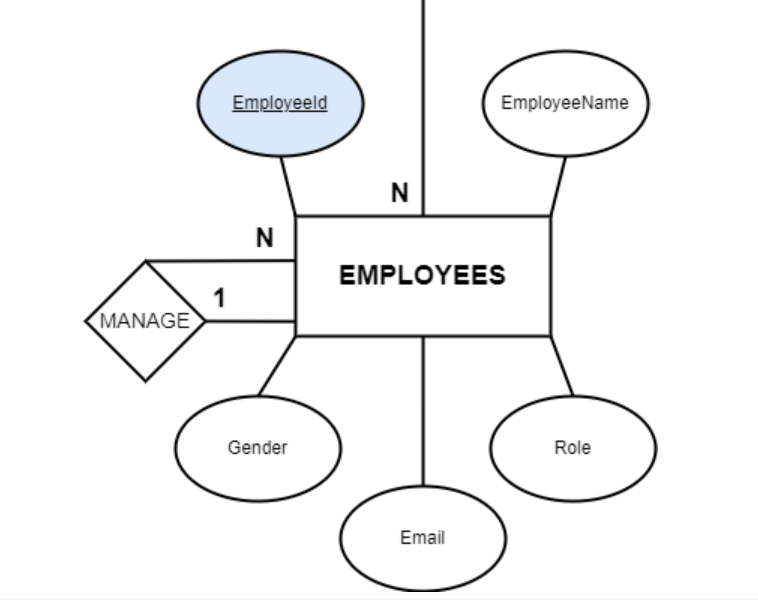
|  |  |
| --- | --- |
| The Flight\_Routes entity defines the routes that flights will take. **Attributes:** FlightRouteId: The primary key for this entity. Departing\_airport: The airport code of the departure airport. Arriving\_airport: The airport code of the arrival airport. |  |
|  |  |

## Flight\_Crews

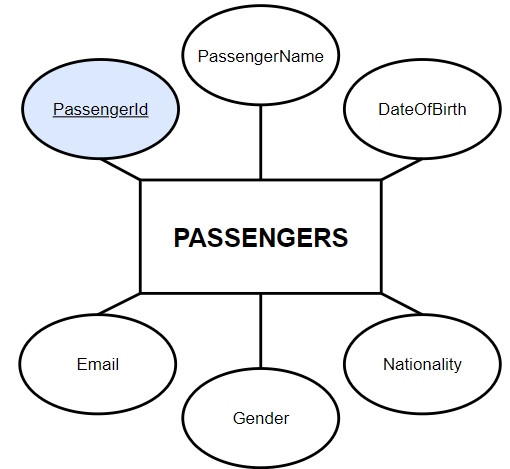


|  |  |
| --- | --- |
| The Flight\_Crews entity manages information about flight crews. **Attributes:** FlightCrewId: The primary key for this entity. FlightCrewCode: A unique code assigned to each flight crew. |  |
|  |  |

## Employees



|  |  |
| --- | --- |
| The Employees entity stores personal and professional information about each airline employee**. Attributes:** EmployeeId: The primary key for this entity. EmployeeName: The name of the employee, consisting of FirstName and LastName. Email: The employee's email address. Gender: The employee's gender. Supervisor: The ID of the employee's supervisor (if applicable). Role: The role of the employee (e.g., pilot, flight attendant). FlightCrewId: A foreign key referencing the FlightCrews entity. |  |
| Passengers |  |
|  |  |



|  |  |
| --- | --- |
| This is the Passengers entity, the root of whole diagram. Tournament entity has 5 attributes.  The attribute PassengerId also is the primary key of this entity. Each Passenger has also Name, Birthday, Email, Gender and Nationality. |  |
| TICKETS  |  |  | | --- | --- | |  |  |     This is the Tickets entity. This has 7 attributes.  Each Ticket has TicketId is primary key, FlightId and PassengerId are foreign keys. The information of ticket include TicketCode, Seat, Class, Price   fLIGHTS   Flights entity has 10 attributes.  Each Flight has FlightId is primary key,  FlightCrewId and FlightRouteId are foreign key.  The informations of flight include FlightCode, Departure\_Time, ArrivalTime,  AirplaneType, Airline, Departing\_gate, Arriving\_gate   full diagram |  |
|  |  |

# V. sql command

I using Microsoft SQL Server 2019, this server build intelligent, mission-critical applications using a scalable, hybrid database platform that has everything built in—from in-memory performance and advanced security to in-database analytics.

## query using order by

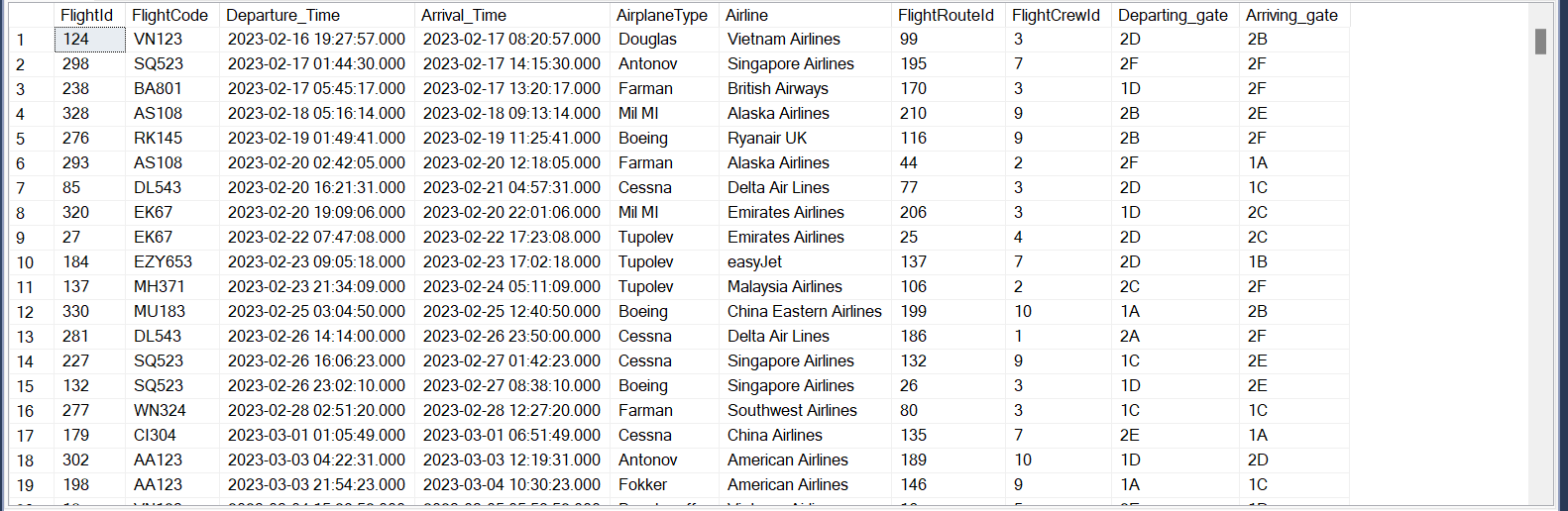
***Code:***

SELECT \*

FROM Flights

ORDER BY Departure\_Time

***Result:***

******

We use query containing **ORDER BY** to sort the list ascending or descending by the values of a domain. **SELECT \* FROM FLIGHTS** command give us all record in Flight table and then sort the records ascending by Departure\_Time

## query using inner join

***Code:***

SELECT P.PassengerName, P.Email, F.FlightCode, F.Departure\_Time, F.Arrival\_Time   
FROM Passengers P

INNER JOIN Tickets T

ON P.PassengerId = T.PassengerId

INNER JOIN Flights F

ON T.FlightId = F.FlightId

***Result:***

|  |
| --- |
|  |
| We use a query containing **INNER JOIN** to combine rows from two or more tables based on a related column between them. In this query, we are retrieving the list of passengers along with their flight details. The **INNER JOIN** operation matches rows in the Passengers table with rows in the Tickets table where the PassengerId values are equal, and further matches these rows with the Flights table where the FlightId values are equal. This way, we get passenger names, emails, and corresponding flight codes, departure times, and arrival times. |

## query using aggregate functions

***Code:***

SELECT TOP 3 F.FlightCode, F.AirplaneType,F.Airline,

SUM(T.Price) AS TotalRevenue

FROM Tickets T JOIN FlightS F

ON T.FlightId = F.FlightId

WHERE F.AirplaneType = 'Boeing'

GROUP BY F.FlightCode, F.AirplaneType, F.Airline  
***Result:***



Displays the flight code, aircraft type, airline and total revenue from tickets sold for the three flights with the aircraft type 'Boeing' with the highest revenue.

## query using the group by and having clauses

***Code:***

select f.FlightCode, COUNT(p.PassengerId) AS NumberOfPassengers

from Flights f

left join Tickets t

on t.FlightId = f.FlightId

left join Passengers p

on p.PassengerId = t.PassengerId

group by f.FlightCode

having COUNT(p.PassengerId) >= all (select COUNT(p.PassengerId)

from Flights f

left join Tickets t

on t.FlightId = f.FlightId

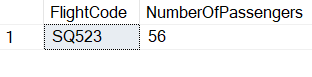
left join Passengers p

on p.PassengerId = t.PassengerId

group by f.FlightCode

)

***Result:***



Find the flight with the most passengers.

## query that uses a sub-query as a relation

***Code:***

select f.FlightCode, f.Airline, f.Departure\_Time, f.Arrival\_Time

from Flights f

join (select fr.FlightRouteId

from Flight\_Routes fr

join AirPorts ap1

on fr.Departing\_airport = ap1.AirportId

join AirPorts ap2

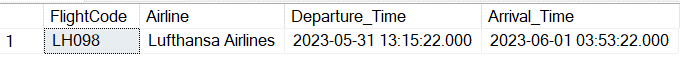
on fr.Arriving\_airport = ap2.AirportId

where ap1.AirportName = 'Charles-de-Gaulle Airport'

and ap2.AirportName = 'Shanghai Pudong International Airport') AS T

on f.FlightRouteId = T.FlightRouteId

***Result:***



Find information of the flight has flight route from Charles-de-Gaulle Airport to Shanghai Pudong International Airport.

## query that uses partial matching in the where clause

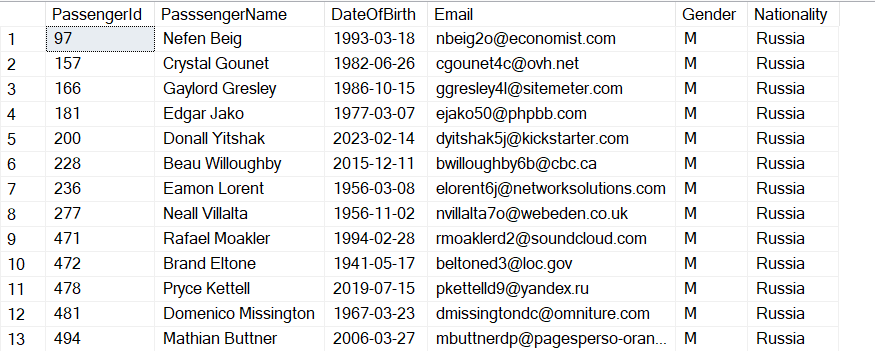
***Code:***

select\*

from Passengers

where Nationality = 'Russia' and Gender = 'M'

***Result:***



Find information of the passengers come from Russia and has Gender is Male

## query that uses a self – join

***Code:***

Select e1.EmployeeId SuperVisorId, e1.EmployeeName Supervisor,

count(e2.EmployeeId) NumberOfEmployees

from Employees e1 left join Employees e2 on e1.EmployeeId = e2.SuperVisor

where e1.SuperVisor is null

group by e1.EmployeeId , e1.EmployeeName

***Result:***

|  |  |
| --- | --- |
| **List supervisors and the number of employees they supervise** |  |

## store procedure

***Code:***

create proc NumberOfTickets

@flightId int,

@NumberOfTickets int output

as

begin

Select @NumberOfTickets = count(t.TicketId)

from FLights f left join Tickets t on f.FlightId = t.FlightId

where f.FlightId = @flightId

group by f.FlightId

end

declare @Numbers int

exec NumberOfTickets 1, @Numbers output

Select @Numbers as NumberOfTickets

***Result:***



**Create a procedure that counts the number of booked tickets for a flight with the flight id as input**

## trigger

***Code:***

create trigger insert\_flight

on Flights

instead of insert

as

begin

if exists (

Select i.FlightId

from inserted i join Flights f on f.FlightId=i.FlightId

)

begin

print('Flight already exist. Cancelled insert.')

rollback transaction;

return

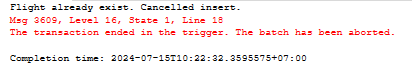
end;

select \* from inserted

end

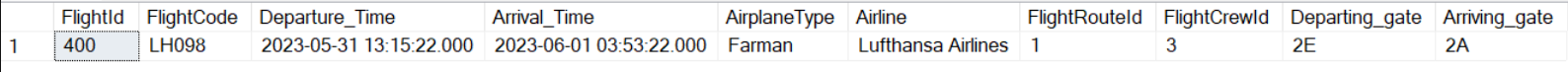
**--insert value flightid duplicate:**

insert into Flights values(1,'LH098','2023/05/31 13:15:22','2023/06/01 03:53:22 AM',N'Farman',N'Lufthansa Airlines',1,3,N'2E',N'2A')



--**insert new value:**

insert into Flights values(400,'LH098','2023/05/31 13:15:22','2023/06/01 03:53:22 AM',N'Farman',N'Lufthansa Airlines',1,3,N'2E',N'2A')



THE END