

## **HOMEWORK - 3**

**MIS686 - Enterprise Database Management**

**Name: Umadevi Betageri**

**Red ID: 827194401**

**Q.No 1.****SQL code for creating database table for Airport:**

# Create Airplanes Table for Airplanes Database

```
CREATE TABLE IF NOT EXISTS Airplanes (
```

```
    Tail_Number INT,
```

```
    Manufacturer VARCHAR(100) NOT NULL,
```

```
    Model_Number INT NOT NULL,
```

```
    Seat_Count INT,
```

```
    PRIMARY KEY (Tail_Number)
```

```
);
```

# Create Flight Crews Table for Airplanes Database

```
CREATE TABLE IF NOT EXISTS Flight_Crews (
```

```
    Flight_Crew_ID INT AUTO_INCREMENT,
```

```
    Flight_Crew_Name VARCHAR(200) NOT NULL,
```

```
    Crew_Count INT,
```

```
    PRIMARY KEY (Flight_Crew_ID)
```

```
);
```

# Create Flights Table for Airplanes Database

```
CREATE TABLE IF NOT EXISTS Flights (
```

```
    Flight_ID INT,
```

```
    Tail_Number INT,
```

```
    Flight_Crew_ID INT,
```

```
    Flight_Date DATE NOT NULL,
```

```
    Origin CHAR(3) NOT NULL,
```

```
    Departure_Time TIME NOT NULL UNIQUE,
```

```
    Destination CHAR(3),
```

```
    Arrival_Time TIME UNIQUE,
```

```
    PRIMARY KEY (Flight_ID),
```

```
    FOREIGN KEY (Tail_Number) REFERENCES Airplanes
```

```
(Tail_Number) ON DELETE SET NULL,
```

```
    FOREIGN KEY (Flight_Crew_ID) REFERENCES Flight_Crews
```

```
(Flight_Crew_ID) ON DELETE SET NULL
```

```
);
```

# Create Passengers Table for Airplanes Database

```
CREATE TABLE IF NOT EXISTS Passengers (
```

```
    Passenger_ID INT AUTO_INCREMENT,
```

```
    Passenger_Name VARCHAR(200) NOT NULL,
```

```

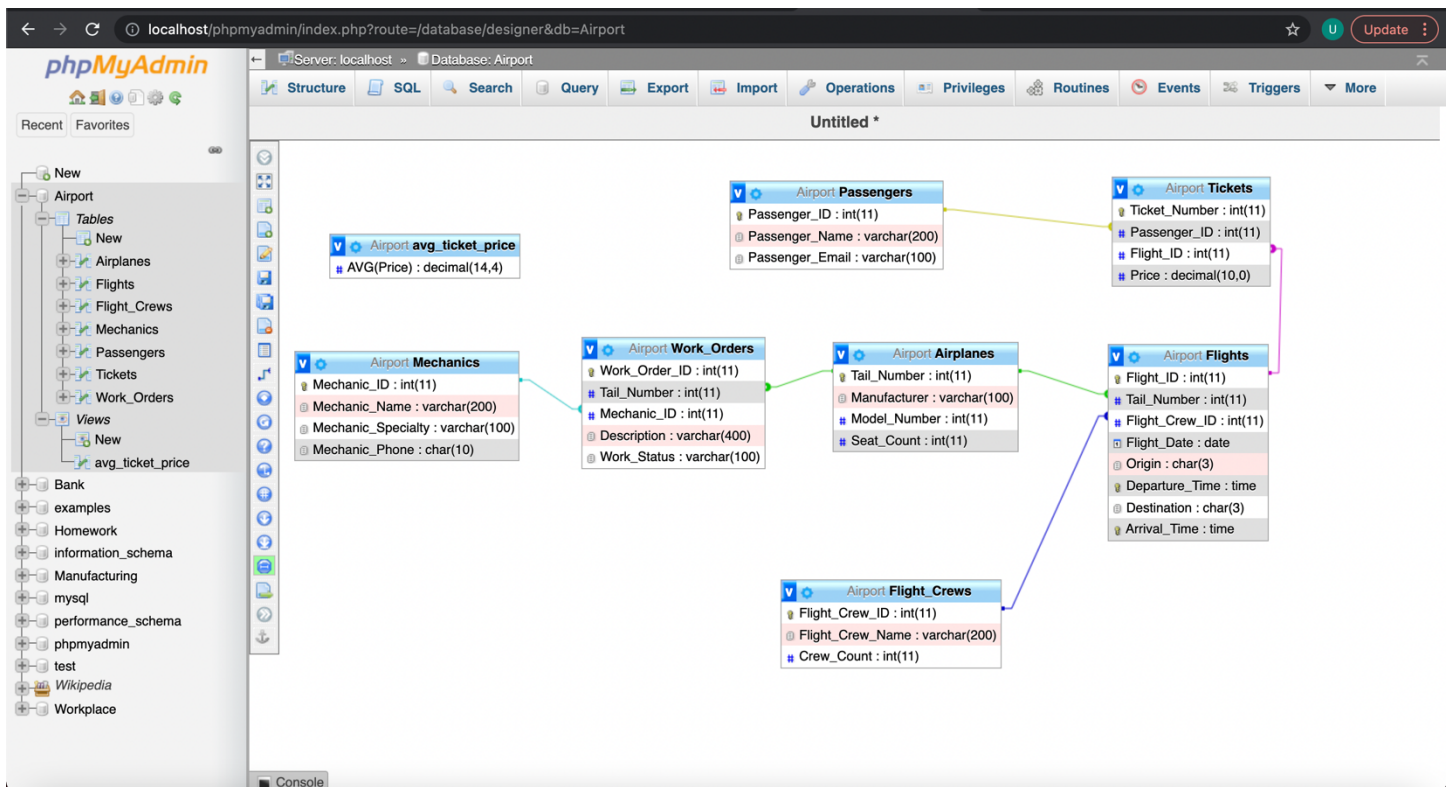
    Passenger_Email VARCHAR(100),
    PRIMARY KEY (Passenger_ID)
);

# Create Tickets Table for Airplanes Database
CREATE TABLE IF NOT EXISTS Tickets (
    Ticket_Number INT AUTO_INCREMENT,
    Passenger_ID INT,
    Flight_ID INT,
    Price DECIMAL NOT NULL DEFAULT 200.00,
    PRIMARY KEY (Ticket_Number),
    FOREIGN KEY (Passenger_ID) REFERENCES Passengers
    (Passenger_ID) ON DELETE SET NULL,
    FOREIGN KEY (Flight_ID) REFERENCES Flights
    (Flight_ID) ON DELETE SET NULL
);

# Create Mechanics Table for Airplanes Database
CREATE TABLE IF NOT EXISTS Mechanics (
    Mechanic_ID INT AUTO_INCREMENT,
    Mechanic_Name VARCHAR(200) NOT NULL,
    Mechanic_Specialty VARCHAR(100),
    Mechanic_Phone CHAR(10),
    PRIMARY KEY (Mechanic_ID)
);

#Create Work Orders Table for Airplanes Database
CREATE TABLE IF NOT EXISTS Work_Orders (
    Work_Order_ID INT AUTO_INCREMENT,
    Tail_Number INT,
    Mechanic_ID INT,
    Description VARCHAR(400),
    Work_Status VARCHAR(100),
    PRIMARY KEY (Work_Order_ID),
    FOREIGN KEY (Tail_Number) REFERENCES Airplanes
    (Tail_Number) ON DELETE SET NULL,
    FOREIGN KEY (Mechanic_ID) REFERENCES Mechanics
    (Mechanic_ID) ON DELETE SET NULL
);

```



## QUERIES

**# 1)** List the flight ID, tail number, departure time, and origin for all flights between January 1, 2021 and March 1, 2021 with a destination of "DCA". Sort the result by the flight ID in ascending order

```
SELECT Flight_ID, Tail_Number, Departure_Time, Origin
FROM Flights
WHERE Flight_Date BETWEEN "2021-01-01" AND "2021-03-01" AND Destination = "DCA"
ORDER BY Flight_ID ASC;
```

**# 2)** List the passenger IDs of all passengers that have purchased tickets over \$1,000 in price. If the same passenger ID has purchased multiple tickets that meet this criterion, then only list the passenger ID once

```
SELECT DISTINCT Passenger_ID
FROM Tickets
WHERE Price > 1000;
```

**# 3)** List all fields for work orders whose descriptions mention “oil”. Order these fields by status descending, then by tail number ascending.

```
SELECT *  
FROM Work_Orders  
WHERE Description LIKE “%oil%”  
ORDER BY Work_Status DESC, Tail_Number ASC;
```

**# 4)** For each passenger name, list their name in addition to the date, origin, departure time, destination, and arrival time of matching all flights. Sort by the passenger name (A-Z), then by the date ascending.

```
SELECT P.Passenger_Name, F.Flight_Date, F.Origin, F.Departure_Time, F.Destination, F.Arrival_Time  
FROM Flights F  
JOIN Tickets T  
ON F.Flight_ID = T.Flight_ID  
JOIN Passengers P  
ON P.Passenger_ID = T.Passenger_ID  
ORDER BY P.Passenger_Name ASC, F.Flight_Date ASC;
```

**# 5) a)** Create a view that computes the average ticket price

```
CREATE VIEW Avg_Ticket_Price AS  
SELECT AVG(Price)  
FROM Tickets
```

**# 5) b)** Based on the view created in (a), list each passenger name alongside the number of tickets that the passenger has purchased with above-average ticket prices.

```
SELECT P.Passenger_Name, COUNT(T.Ticket_Number) AS Num_of_Tickets  
FROM Tickets T  
JOIN Passengers P  
ON P.Passenger_ID = T.Passenger_ID  
WHERE T.Price > (SELECT * FROM Avg_Ticket_Price)  
GROUP BY P.Passenger_Name;
```

**# 6)** List all airplane tail numbers alongside the number of work orders pertaining to that airplane that have a “pending” status. Order by the number of work orders from high to low.

```
SELECT Tail_Number, COUNT(Work_Order_ID) AS Num_of_WO  
FROM Work_Orders  
WHERE Work_Status LIKE “pending”  
GROUP BY Tail_Number  
ORDER BY Num_of_WO DESC;
```

**# 7)** List all flight crew names alongside all flight IDs and tail numbers to which those flight crews are assigned with an origin of "SAN". If a flight crew has not been assigned to a flight yet, then still list it in the results. Sort by the date in descending order, then by the departure time in descending order.

```
SELECT C.Flight_Crew_Name, F.Flight_ID, F.Tail_Number
FROM Flights F
LEFT JOIN Flight_Crews C
ON F.Flight_Crew_ID = C.Flight_Crew_ID
WHERE F.Origin = "SAN"
ORDER BY F.Flight_Date DESC, F.Departure_Time DESC;
```

**# 8)** For each passenger name, list their name, all flight destinations that they have flown to, and the sum of the ticket prices for each destination. Sort by the sum of ticket prices in descending order.

```
SELECT P.Passenger_Name, F.Destination, SUM(T.Price) AS Total_Price
FROM Flights F
JOIN Tickets T
ON F.Flight_ID = T.Flight_ID
JOIN Passengers P
ON T.Passenger_ID = P.Passenger_ID
GROUP BY F.Destination, P.Passenger_Name
ORDER BY Total_Price DESC;
```

**# 9)** List all descriptions of work orders with a status of "completed" assigned to mechanics with a name that starts with "Eric".

```
SELECT Description
FROM Work_Orders
WHERE Work_Status = "completed" AND Mechanic_ID IN
    (SELECT Mechanic_ID
     FROM Mechanics
     WHERE Mechanic_Name LIKE "Eric%");
```

**# 10)** For each flight destination, list each airplane manufacturer whose airplanes have flown to that destination alongside the number of flights that the manufacturer's flights have flown to that destination. Sort by the manufacturer in ascending order, then the number of flights in descending order.

```
SELECT F.Destination, A.Manufacturer, COUNT(F.Flight_ID) AS No_of_Flights
FROM Flights F
JOIN Airplanes A
ON F.Tail_Number = A.Tail_Number
GROUP BY F.Destination, A.Manufacturer
ORDER BY A.Manufacturer ASC, No_of_Flights DESC;
```