Airport Database

CREATING TABLES

#Creating Mechanics table

```
CREATE TABLE IF NOT EXISTS Mechanics (
Mechanic_ID INT (AUTO_INCREMENT),
Mechanic_Name VARCHAR(100) NOT NULL,
Mechanic_Specialty VARCHAR(50),
Mechanic_Phone CHAR(10),
PRIMARY KEY(Mechanic_ID),
);
```

 # Creating Work_Orders table

```
CREATE TABLE IF NOT EXISTS Work_Orders(
Work_Orders ID INT (AUTO_INCREMENT),
Tail_Number INT,
Mechanic_ID INT,
Description VARCHAR(100),
Status VARCHAR(30),
PRIMARY KEY(Work_Orders ID),
FOREIGN KEY(Tail_Number) REFERENCES Airplanes (Tail_Number) ON DELETE SET NULL ON UPDATE SET NULL,
FOREIGN KEY(Mechanic_ID) REFERENCES Mechanics (Mechanic_ID) ON DELETE SET NULL ON UPDATE SET NULL
);
```

#Creating Airplanes table

```
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)
);
```

#Creating Flights table

```
CREATE TABLE IF NOT EXISTS Flights(
Flight_ID INT,
Tail_Number INT,
Flight_Crew_ID INT,
Date DATE NOT NULL,
Origin CHAR(3) NOT NULL,
Departure_Time TIME NOT NULL UNIQUE,
Destination CHAR(3),
Arrival_Time TIME NOT NULL UNIQUE,
PRIMARY KEY(Flight_ID),
FOREIGN KEY(Tail_Number) REFERENCES Airplanes (Tail_Number) ON DELETE SET NULL ON UPDATE SET NULL,
FOREIGN KEY(Flight_Crew_ID) REFERENCES Flight_Crews (Flight_Crew_ID) ON DELETE SET NULL ON UPDATE SET NULL
);
```

#Creating Flight_Crews table

```
CREATE TABLE IF NOT EXISTS Flight_Crews (
Flight_Crew_ID INT AUTO_INCREMENT,
Flight_Crew_Name VARCHAR(100) NOT NULL,
Crew_Count INT ,
PRIMARY KEY(Flight_Crew ID)
);
```

#Creating Tickets table

```
CREATE TABLE IF NOT EXISTS Tickets (
Ticket_Number INT AUTO_INCREMENT,
Passenger_ID INT,
Flight_ID INT,
Price DECIMAL(10,3) DEFAULT 200.0 NOT NULL,
PRIMARY KEY(Ticket_Number),
FOREIGN KEY(Passenger_ID) REFERENCES Passengers (Passenger_ID) ON DELETE SET NULL ON UPDATE SET NULL
FOREIGN KEY(Flight_ID) REFERENCES Flights (Flight_ID) ON DELETE SET NULL ON UPDATE SET NULL
);
```

#Creating Passenger table

```
CREATE TABLE IF NOT EXISTS Passengers (
Passenger_ID INT AUTO_INCREMENT,
Passenger_Name VARCHAR(100) NOT NULL,
Passenger_Email VARCHAR(120),
PRIMARY KEY(Passenger_ID)
);
```

QUERIES

1. **List the flight ID, tail number, departure time, and origin for all flights between January 1,2022 and March 1, 2022 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 Date BETWEEN '2022-01-01' AND '2022-03-01' 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 thepassenger ID once.

```
SELECT DISTINCT P.Passenger_ID
FROM Passenger P
JOIN Tickets T
ON P.Passenger_ID = T.Passenger_ID
WHERE T.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 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 bythe date ascending.

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

5. a) Create a view that computes the average ticket price.
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.

```
CREATE VIEW IF NOT EXISTS Avg_Ticket_Price AS
    SELECT AVG(Price)
    FROM Tickets;

SELECT P.Passenger_Name, P.Passenger_ID, COUNT(*) AS Number_of_tickets
FROM Passenger P
JOIN Tickets T
ON P.Passenger_ID = T.Passenger_ID
WHERE T.Ticket_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(*) AS number_of_work_orders
```

```
FROM Work_Orders
WHERE Status LIKE "%pending%"
GROUP BY Tail_Number
ORDER BY number_of_work_orders DESC;
```

7. List all flight crew names alongside all flight IDs and tail numbers to which those flight crewsare assigned with an origin of "SAN". If a flight crew has not been assigned to a flight yet, thenstill 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 Flight_Crew C
LEFT JOIN Flights F
ON C.Flight_Crew ID=F.Flight_Crew_ID
WHERE F.Origin="SAN" OR F.Origin IS NULL
ORDER BY F.Date DESC, F.Departure_Time DESC;
```

8. For each passenger name, list their name, all flight destinations that they have flown to, andthe sum of the ticket prices for each destination. Sort by the sum of ticket prices in descendingorder.

```
SELECT P.Passenger_Name, F.Destination, SUM(T.Price) AS Total_Price
FROM Passengers P
JOIN Tickets T
ON P.Passenger_ID=T.Passenger_ID
JOIN Flights F
ON T.Flight_ID=F.Flight_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 W.Description
FROM Work_Order W
JOIN Mechanics M
ON W.Mechanic_ID=M.Mechanic_ID
WHERE (W.Status="completed") AND M.Mechanic 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 A.Manufacturer, COUNT(*) AS Flight_Count
FROM `Flights` F
JOIN Airplanes A
ON F.Tail_Number = A.Tail_Number
GROUP BY F.Destination, A.Manufacturer
ORDER BY A.Manufacturer ASC, Flight_Count DESC;
```