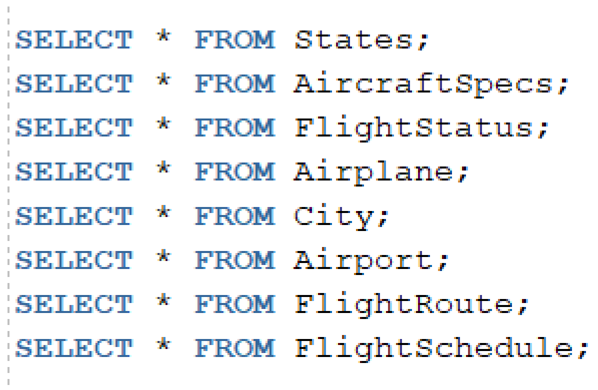
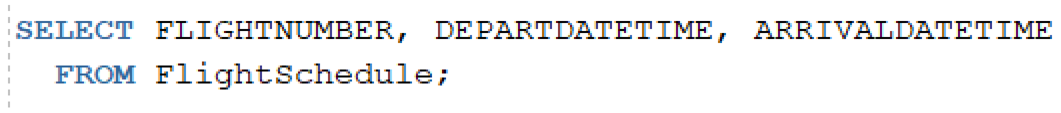
1. **Run the Airline_DB.sql script to create the database tables.**



```
SELECT * FROM States;
SELECT * FROM AircraftSpecs;
SELECT * FROM FlightStatus;
SELECT * FROM Airplane;
SELECT * FROM City;
SELECT * FROM Airport;
SELECT * FROM FlightRoute;
SELECT * FROM FlightSchedule;
```

First of all, we ran the Airline_DB.sql script to create all the needed tables to work on. Then, we used the “SELECT *” command for each of the tables separately to have an idea of what each table looks like and contains. The ”SELECT ” command selects a specific part of the table and the “*” with the “SELECT” command basically means to select all the components of the table.

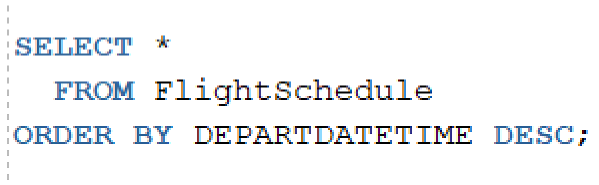
2. **Write a query that displays Flight Number, Depart DateTime, Arrival DateTime for all scheduled flights.**



```
SELECT FLIGHTNUMBER, DEPARTDATETIME, ARRIVALDATETIME
  FROM FlightSchedule;
```

Here, we used the “SELECT” command along with the “FLIGHTNUMBER”, “DEPARTDATETIME”, and “ARRIVALDATETIME” columns from the “FlightSchedule” table meaning that we want to select these 3 columns(and all their rows) from the “FlightSchedule” table specifically. By running this query, we can find all the scheduled flights by their numbers,  and what time they depart and arrive. The business question in this case would be: What is the departure and arrival time of each scheduled flight?

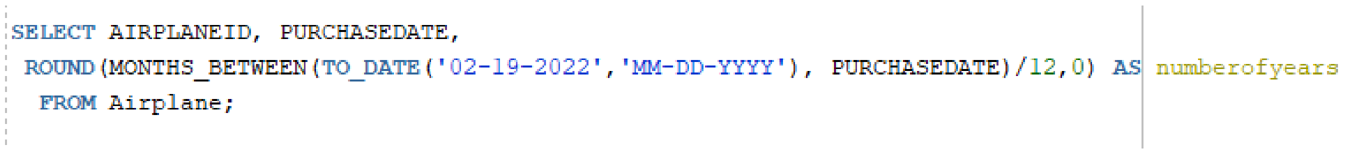
3. **Write a query that displays information for all scheduled flights sorted by Depart DateTime in descending order.**



```
SELECT *
  FROM FlightSchedule
ORDER BY DEPARTDATETIME DESC;
```

In this question, we selected all columns (Select *) from the “FlightSchedule” table ordered in descending order of departure time (“ORDER BY DEPARTTIME DESC”). By running this query, we can see all the scheduled flights, ordered from the last to first, in the month of October (all flights in this table are in October). The business question would be:When are the latest up to the earliest flights in the month of October, and what are all their attributes (flight number, ID,...)?

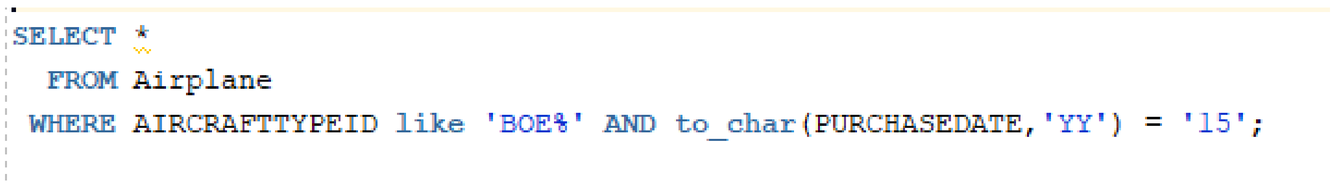
4. **Write a query that displays each AirplaneID, purchase date, and how many years the airplane has been in service since purchased. You need to generate the Number of Years value based on other information located in the Airplane table.**



```
SELECT AIRPLANEID, PURCHASEDATE,
 ROUND(MONTHS_BETWEEN(TO_DATE('02-19-2022','MM-DD-YYYY'), PURCHASEDATE)/12,0) AS numberofyears
  FROM Airplane;
```

Here we used the “Select” command to take “AirplaneID” column, “PurchaseDate” column, and the “numberofyears” column which we created in the “Airplane” table by calculate the number of months between the date purchased and today (and then converting it to years). We used the “To_Date” function to specify today's date. Afterwards, we divide the value by 12 to get the number in years, and then we rounded it and gave the column  the name “numberofyears”. By running this query, we get the requested columns from the “Airplane” table, along with the “numberofyears” column which we created that enabled us to see how long each airplane has been in use. The business question in this case would be: For how many years has each airplane been utilized?

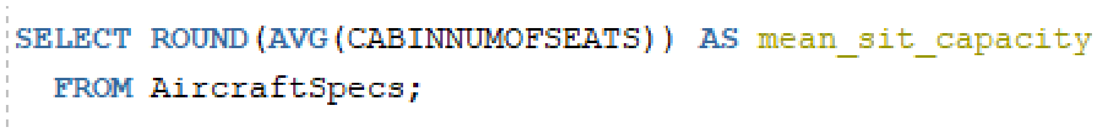
5. **Write a query that filters and displays all rows from the Airplane table such that the following parameters are true:**
   **-The aircraftTypeID start with 'BOE'**
   **-The purchaseDate is in the year 2015**



```
SELECT *
  FROM Airplane
 WHERE AIRCRAFTTYPEID like 'BOE%' AND to_char(PURCHASEDATE,'YY') = '15';
```

Here we selected all the rows and columns (Select *) from the “Airplane” table where the aircrafttypeID starts with BOE (like ‘BOE%’ which means that this word has to start with BOE and it doesn’t matter how it ends) and the purchase date is 2015 (to_char(purchasedate, ‘YY’) = ‘15’). The result we get from the query is all the columns and rows of airplanes that have an ID starting with “BOE” (as in BOEING airplanes) and were purchased in 2015. The business question would be: How many and what BOEING aircrafts were purchased in 2015?

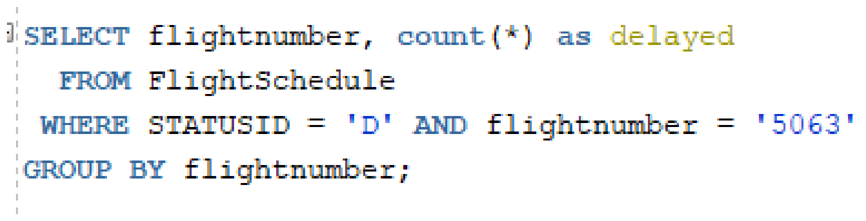
- Write a query to find the average of the aircrafts seating capacity.



```
SELECT ROUND(AVG(CABINNUMOFSEATS)) AS mean_sit_capacity
  FROM AircraftSpecs;
```

We used Select query to take average of “cabinnumofseats” columns and then rounded it by giving name as “mean_sit_capacity”, obtaining a result of approximately 236 seats on average.The business question would be: What is the average number of seats in all aircrafts or What is the average number of customers that can travel in the aircrafts.

- Write a query to find how often flights on the route with flight number ‘5063’ were delayed.

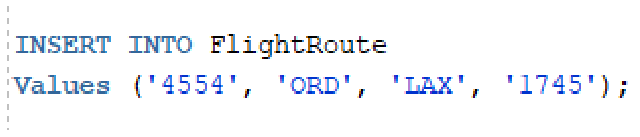


```
SELECT flightnumber, count(*) as delayed
  FROM FlightSchedule
 WHERE STATUSID = 'D' AND flightnumber = '5063'
GROUP BY flightnumber;
```

We selected “flightnumber” column from “FlightSchedule” table, filtered it with “statusid” that equals to ‘D’ (corresponding to Delay) and “flightnumber” equals to ‘5063’ and we grouped by “flightnumber” column to count the number of delayed flights. By running this query, we were able to see how many times flight ‘5063’ was delayed. The business question is: How many times has flight number 5063 been delayed.

**6. Define and execute SQL commands to insert, update, and delete table records:**

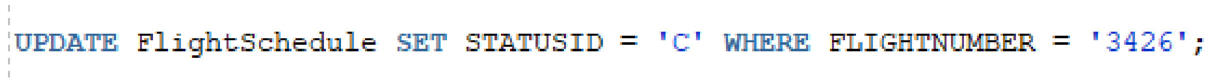
- Define and execute a SQL command to insert a new flight route using the following values: flightnumber = '4554', departairport = 'ORD', arriveairport = 'LAX', distance = 1745.



```
INSERT INTO FlightRoute
Values ('4554', 'ORD', 'LAX', '1745');
```

Here, we inserted a new row with all the demanded values by using the “INSERT INTO… VALUES” meaning that we want to insert a new row into the “FlightRoute” table with the values “4554” for flight number, “ORD” for departure airport, “LAX ” for arrival airport, and “1745” for distance. We write the values in the exact respective order that we want to add for each column. By writing this query, we have a new row with all desired values inserted into the table “FlightRoute”.

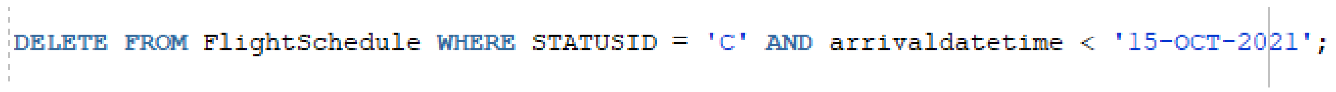
- Define and execute a SQL command to update the status of scheduled flights with route number ‘3426’ to ‘C’ (canceled).



```
UPDATE FlightSchedule SET STATUSID = 'C' WHERE FLIGHTNUMBER = '3426';
```

Here, we updated the “FlightSchedule" table to change the status id to “C” (SET STATUSID = ‘C’) where we have a flight number of “3426” (WHERE FLIGHTNUMBER = ‘3426’). By running this query we change the status ID of flight number 3426 into ‘C’ (canceled).

- Define and execute a SQL command to delete all the canceled flights with a departure date less than ‘15-OCT-2021’.

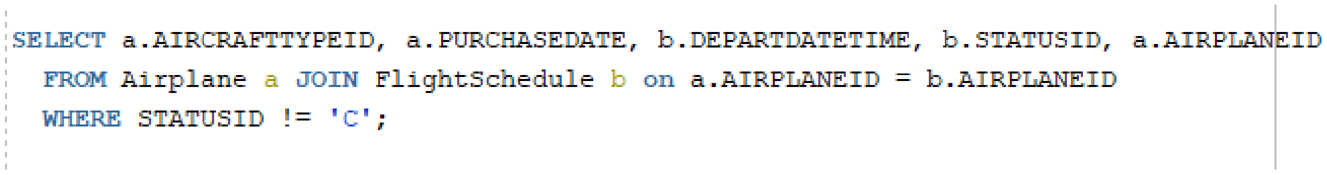


```
DELETE FROM FlightSchedule WHERE STATUSID = 'C' AND arrivaldatetime < '15-OCT-2021';
```

Here, we delete all the rows from the table “Flight Schedule” (DELETE FROM) where the flights are canceled or the arrival date time is before October 15 (WHERE STATUSID = ‘C’ AND ARRIVALDATETIME < ‘15-OCT-2021’). By running this query, we cleaned the table to be free of canceled flights or flights arriving before the 15th of October. The “Delete” function helps in cleaning the data for analysis.

These commands (Insert, Update, and Delete) are very important Data Manipulation Language commands. They are necessary to add data, change data, or omit unneeded data from a table. Basically without these commands, we wouldn't be able to create a table in the first place to use or work on. Also, we might have some errors in data that can be updated with those commands.

**7. Define and execute a SQL query to display all flights which were on time or delayed. You need to display the flight Aircraft Type ID, Aircraft purchase date, flight depart DateTime, and statusID. What business question do you think this SQL query answers?**



```
SELECT a.AIRCRAFTTYPEID, a.PURCHASEDATE, b.DEPARTDATETIME, b.STATUSID, a.AIRPLANEID
  FROM Airplane a JOIN FlightSchedule b on a.AIRPLANEID = b.AIRPLANEID
  WHERE STATUSID != 'C';
```

We selected “AirplaneID”, “AircraftTypeID” and “purchasedate” from “Airplane” table and flight “departDateTime” with “statusID” from “FlightSchedule” table. We then used the JOIN command to inner join the two tables based on the same “airplaneid” and gave them names “a” for “Airplane” and “b” for “Flightschedule” , filtering the tables where statusid is not “C” (not canceled). The business question the query answers is: What are the airplanes that were scheduled for flight and have not been canceled?