DBMS ASSIGNMENT

Nikki Gautam

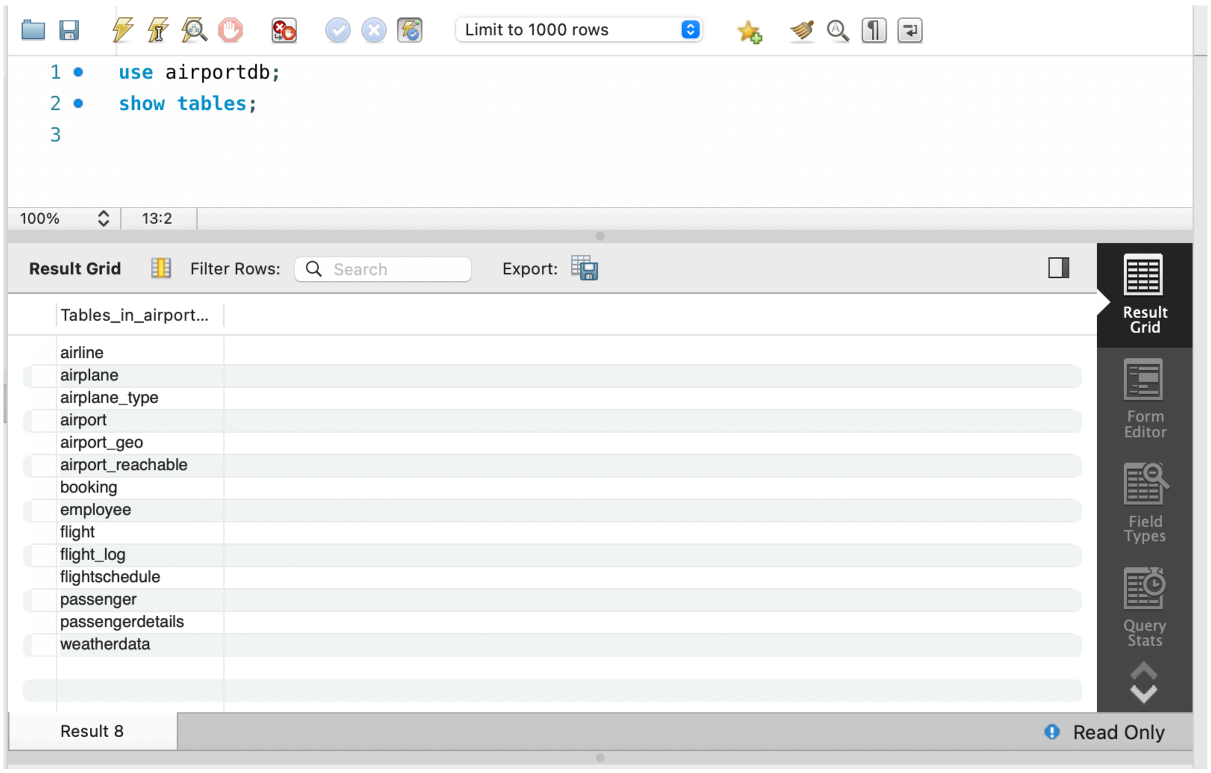
Question 3:

Language Used (for querying): **SQL**

Part A:

Produce the queries for each item in the list

IMPORTED DATABASE-



```
use airportdb;
show tables;
```

```
Tables_in_airport...
airline
airplane
airplane_type
airport
airport_geo
airport_reachable
booking
employee
flight
flight_log
flightschedule
passenger
passengerdetails
weatherdata
```

Relations and Attributes in it: (relation_name : attributes)

1. **airline**:
   airline_id, iata, airlinename, base_airport
2. **airplane**:
   airplane_id, capacity, type_id, airline_id

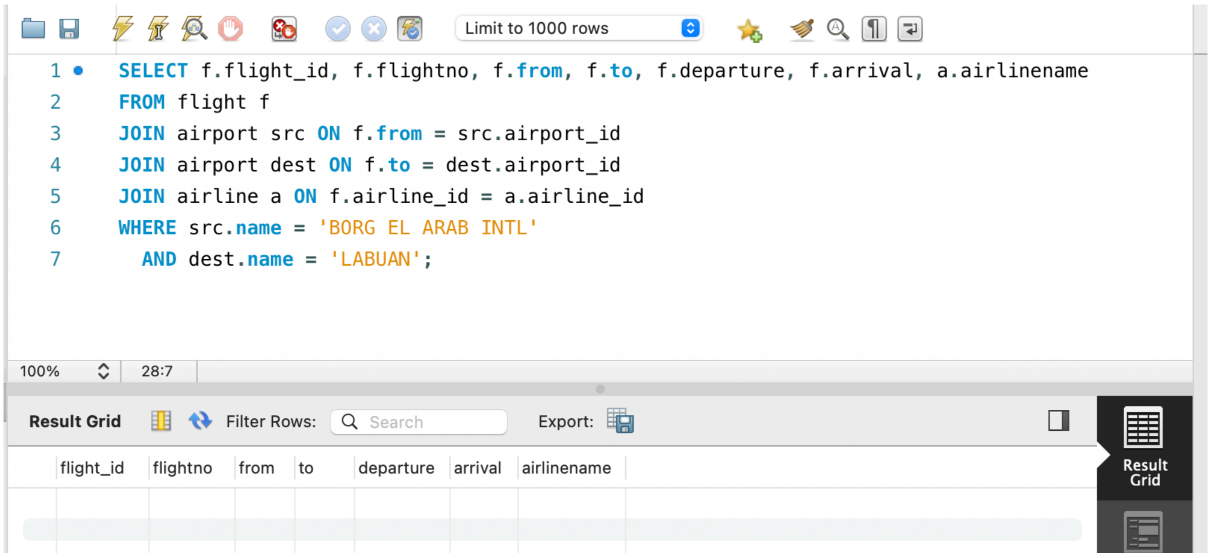
3. **airplane_type**:
   type_id, identifier, description
4. **airport**:
   airport_id, iata, icao, name
5. **airport_geo**:
   airport_id, name, city, country, latitude, longitude, geolocation
6. **airport_reachable**:
   airport_id, hops
7. **booking**:
   booking_id, flight_id, seat, passenger_id, price
8. **employee**:
   employee_id, firstname, lastname, birthdate, sex, street, city, zip, country, emailaddress, telephoneno, salary, department, username, password
9. **flight**:
   flight_id, flightno, from, to, departure, arrival, airline_id, airplane_id
10. **flight_log**:
    flight_log_id, log_date, user, flight_id, flightno_old, flightno_new, from_old, to_old, from_new, to_new, departure_old, arrival_old, departure_new, arrival_new, airplane_id_old, airplane_id_new, airline_id_old, airline_id_new, comment
11. **flightschedule**:
    flightno, from, to, departure, arrival, airline_id, monday, tuesday, wednesday, thursday, friday, saturday, sunday
12. **passenger**:
    passenger_id, passportno, firstname, lastname
13. **passengerdetails**:
    passenger_id, birthdate, sex, street, city, zip, country, emailaddress, telephoneno
14. **weatherdata**:
    log_date, time, station, temp, humidity, airpressure, wind, weather, winddirection

Retrieve the following information using SQL Queries:

1. Given a source Airport Name (Ex : 'BORG EL ARAB INTL') and a destination airport name (Ex :'LABUAN') retrieve all the available flights from the source to the destination

SQL Query:

```
SELECT   f.flight_id,   f.flightno,   f.from,   f.to,
f.departure, f.arrival, a.airlinename
FROM flight f
JOIN airport src ON f.from = src.airport_id
JOIN airport dest ON f.to = dest.airport_id
JOIN airline a ON f.airline_id = a.airline_id
WHERE src.name = 'BORG EL ARAB INTL'
  AND dest.name = 'LABUAN';
```



```
SELECT f.flight_id, f.flightno, f.from, f.to, f.departure, f.arrival, a.airlinename
FROM flight f
JOIN airport src ON f.from = src.airport_id
JOIN airport dest ON f.to = dest.airport_id
JOIN airline a ON f.airline_id = a.airline_id
WHERE src.name = 'BORG EL ARAB INTL'
  AND dest.name = 'LABUAN';
```

flight_id	flightno	from	to	departure	arrival	airlinename

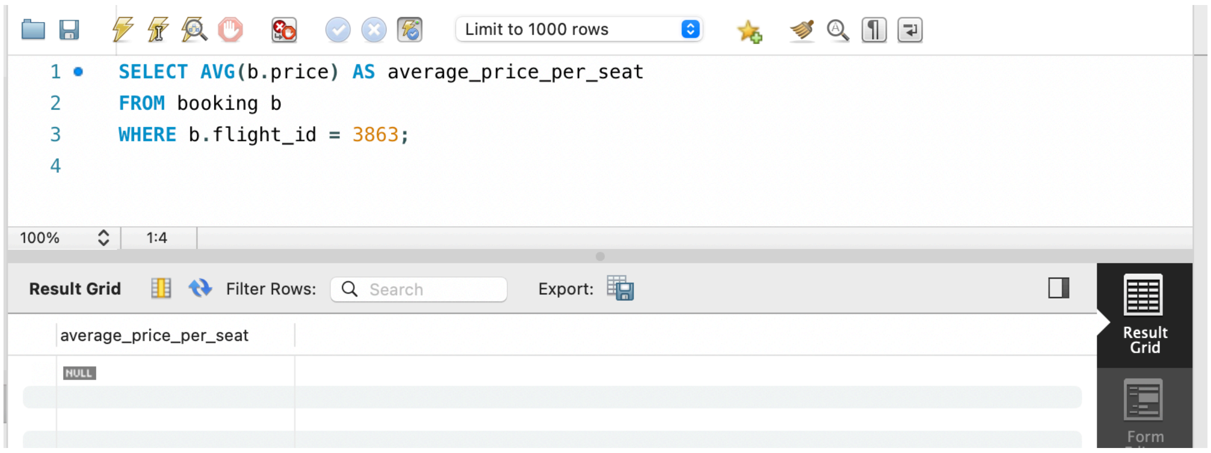
Description:

1. Joined the "**flight**" table with the "**airport**" table twice to map the source (from) and destination (to) airports
2. Then, joined the "**airline**" table to get the airline information
3. Lastly, filtered the flights where the source airport is 'BORG EL ARAB INTL' and the destination airport is 'LABUAN' (for example).

2. Given a flight id (Ex: 3863), what is the average price of booking for every seat

SQL Query:

```
SELECT AVG(b.price) AS average_price_per_seat
FROM booking b
WHERE b.flight_id = 3863;
```



```
SELECT AVG(b.price) AS average_price_per_seat
FROM booking b
WHERE b.flight_id = 3863;
```

```
average_price_per_seat
NULL
```

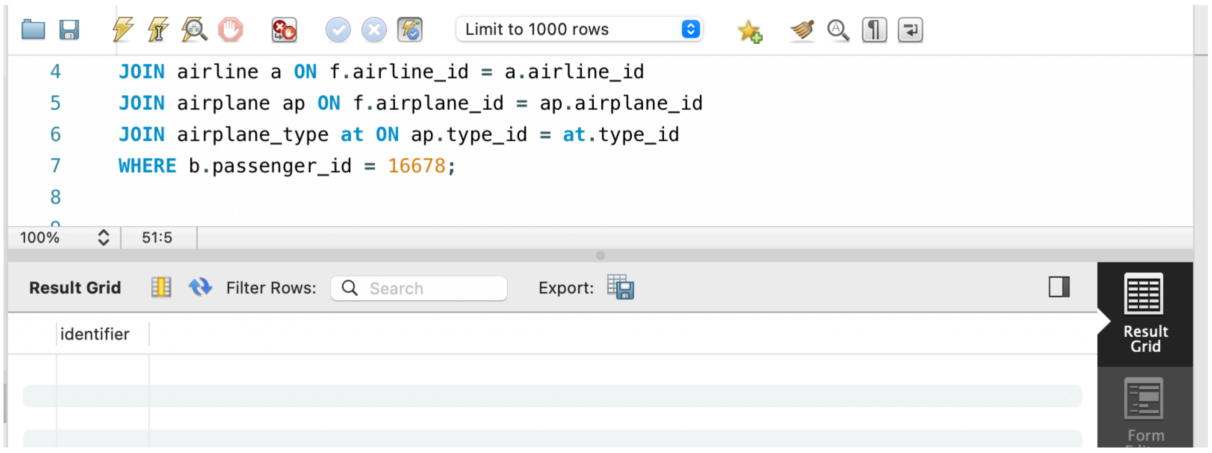
Description:

1. Filtered the "**booking**" table for the given "**flight_id**" (given 3863)
2. Then, calculated the average price (**AVG(b.price)**) for all the bookings associated with that flight.

3. Given a passenger id (Ex : 16678) return all the different airline types he has travelled in

SQL Query:

```
SELECT DISTINCT at.identifier
FROM booking b
JOIN flight f ON b.flight_id = f.flight_id
JOIN airline a ON f.airline_id = a.airline_id
JOIN airplane ap ON f.airplane_id = ap.airplane_id
JOIN airplane_type at ON ap.type_id = at.type_id
WHERE b.passenger_id = 16678;
```



```
JOIN airline a ON f.airline_id = a.airline_id
JOIN airplane ap ON f.airplane_id = ap.airplane_id
JOIN airplane_type at ON ap.type_id = at.type_id
WHERE b.passenger_id = 16678;
```

Description:

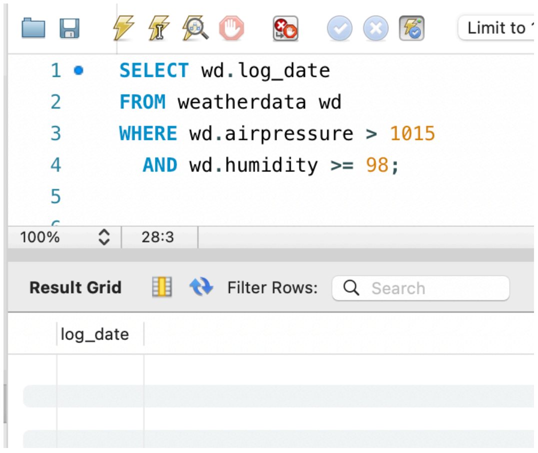
1. Joined the "**booking**", "**flight**", "**airline**", "**airplane**", and "**airplane_type**" tables to connect the passenger with the flights they have taken, and the types of airplanes they have travelled on
2. Then, filtered for the given "**passenger_id**" (16678)
3. Finally, selected distinct airline types (**at.identifier**) the passenger has flown with

4. The weather is considered as not suitable for flight if the air-pressure is greater 1015 and humidity is greater than or equal to 98. Calculate the number of dates which were not suitable for flight & list the dates as well

SQL Query:

- Displaying dates which were not suitable for flying:

```
SELECT wd.log_date
FROM weatherdata wd
WHERE wd.airpressure > 1015
  AND wd.humidity >= 98;
```



```
SELECT wd.log_date
FROM weatherdata wd
WHERE wd.airpressure > 1015
  AND wd.humidity >= 98;
```

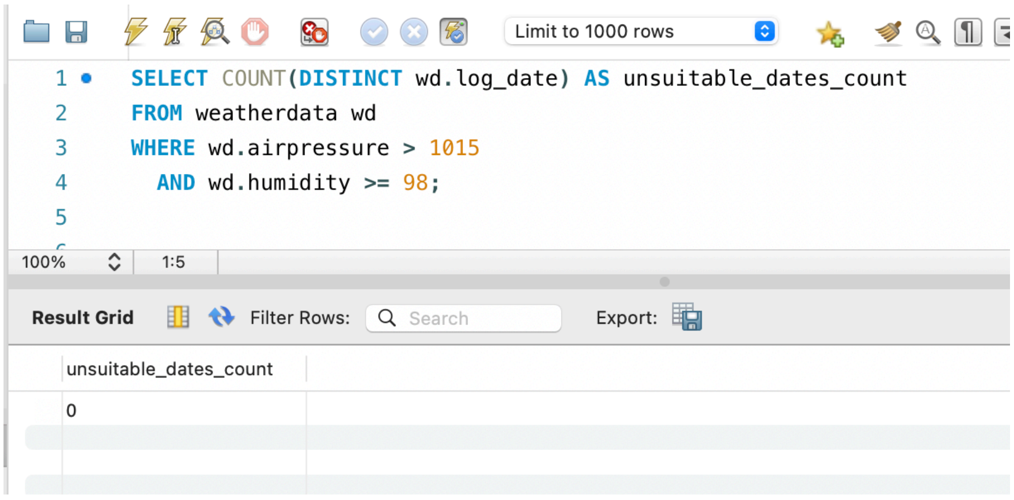
```
log_date
```

Description:

1. Filtered the “**weatherdata**” table on the condition for unsuitable weather: air pressure greater than 1015 and humidity greater than or equal to 98
2. Selected the “**log_date**” of those records

- Displayig the total number of such dates:

```
SELECT COUNT(DISTINCT wd.log_date) AS
unsuitable_dates_count
FROM weatherdata wd
WHERE wd.airpressure > 1015
  AND wd.humidity >= 98;
```



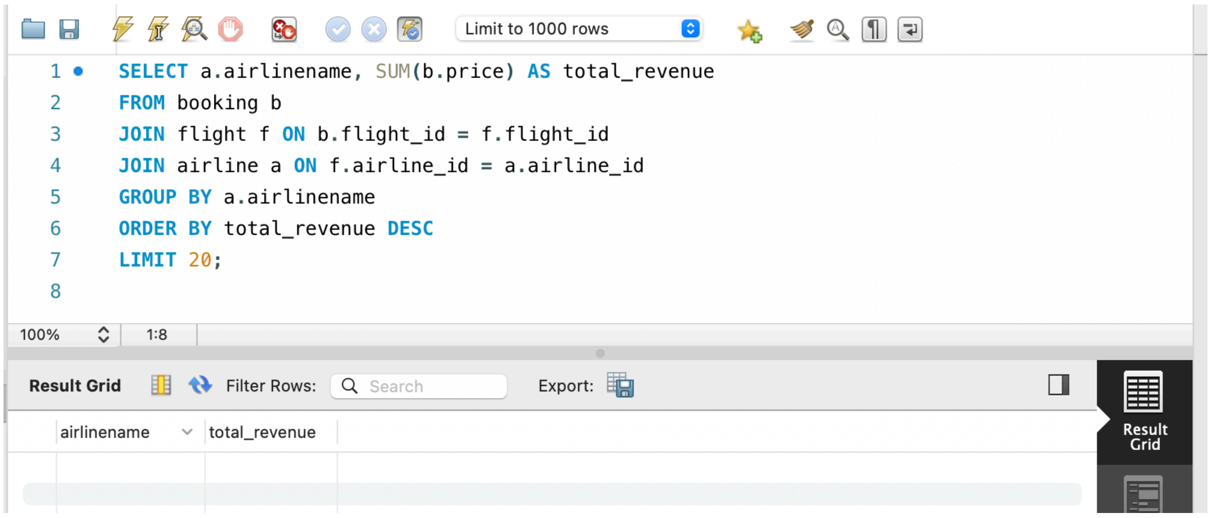
```
SELECT COUNT(DISTINCT wd.log_date) AS unsuitable_dates_count
FROM weatherdata wd
WHERE wd.airpressure > 1015
  AND wd.humidity >= 98;
```

```
unsuitable_dates_count
0
```

5. Identify the top airlines (at-least top 20) ordered by revenue. Revenue is defined as the sum of all the booking price for that airline

SQL Query:

```
SELECT a.airlinename, SUM(b.price) AS total_revenue
FROM booking b
JOIN flight f ON b.flight_id = f.flight_id
JOIN airline a ON f.airline_id = a.airline_id
GROUP BY a.airlinename
ORDER BY total_revenue DESC
LIMIT 20;
```



```
SELECT a.airlinename, SUM(b.price) AS total_revenue
FROM booking b
JOIN flight f ON b.flight_id = f.flight_id
JOIN airline a ON f.airline_id = a.airline_id
GROUP BY a.airlinename
ORDER BY total_revenue DESC
LIMIT 20;
```

airlinename	total_revenue

Description:

1. Firstly, joined the "**booking**" table with the "**flight**" and "**airline**" tables to get the relevant information about the bookings and airlines
2. Then, grouped the results by the airline name (**a.airlinename**)
3. Then, sumed up the booking prices (**SUM(b.price)**) to calculate the total revenue for each airline
4. Finally, ordered the results in descending order of total revenue (**ORDER BY total_revenue DESC**)
5. Limited the results to the top 20 airlines using LIMIT 20 (as mentioned in the question)

Part B:

If you have made some optimizations to make this query run faster, we would like write up on what that optimization was, how did you figure it out and how did it help you.

Query Optimizations:

Given that we may have large datasets in the “**booking**”, “**flight”**, and “**airline”** tables, optimizing the queries can significantly improve performance

1. Explicit INNER JOIN

Replaced implicit joins with explicit INNER JOIN to clarify join conditions. This was done since, it was read through the material that this sometimes helps the database optimizer choose a better execution plan

2. Aggregation

Performed SUM in a subquery before joining with the airline table. This would reduce the data size early, speeding up subsequent joins and final aggregation

3. Limiting Data Early

Filtered unnecessary data by excluding airlines with no bookings early in the query. This would help reduce the processing load, improving efficiency