Mei Shin Lee, Ruhejami Mustari
Databases Final Project

Task: A separate file that lists all of the use cases and the queries executed by them (with brief explanation). This should be well organized and readable. It should be detailed enough to give readers a good idea of how your application works, without making them dig through all the code.

Public Actions

When a user clicks on Search Flights, they are given the option to search for future flights without being logged into the system. A user is given a choice to search for one-way or round trip flights. For one-way flights a user is prompted to enter a source and destination airport, departure date, along with its arrival and departure airports. The query filters for only One-Way flights, and the results are populated in a table with relevant attributes.

- **Search for One-Way Future Flights**

```
SELECT AirlineName, FlightNumber, DepartureDate, DepartureTime, ArrivalDate, FlightStatus
FROM Flight AS f
NATURAL JOIN updates
INNER JOIN airport AS a1 ON a1.AirportName = f.DepartureAirport
INNER JOIN airport AS a2 ON a2.AirportName = f.ArrivalAirport
WHERE a1.AirportCity = %s AND f.DepartureAirport = %s AND a2.AirportCity = %s AND
f.ArrivalAirport = %s AND DepartureDate = %s AND DepartureDate > CURRENT_DATE AND
f.FlightNumber NOT IN #not round trip
 (SELECT FlightNumber
 FROM flight as f2
 GROUP BY FlightNumber
 HAVING COUNT(f2.FlightNumber) > 1)
```

Similarly, a user is also allowed to search for round trip flights. The entered criteria should be the same as searching for one-way flights, with an additional field for return date. Note that round-trip flights are identified by two flights having the same flight number, but have different departure dates and times.

- **Search for Round Trip Future Flights**

```
SELECT f.AirlineName, f.FlightNumber, f.DepartureDate, f3.DepartureDate AS ReturnDate,
f.DepartureTime, f3.DepartureTime AS ReturnTime, f.ArrivalDate, FlightStatus, COUNT(ticketID)
as booked, numberOfSeats FROM flight as f
LEFT JOIN purchasedfor AS p
ON p.FlightNumber = f.FlightNumber AND p.DepartureDate = f.DepartureDate AND
p.DepartureTime = f.DepartureTime
INNER JOIN updates AS u
ON u.FlightNumber = f.FlightNumber AND u.DepartureDate = f.DepartureDate AND
u.DepartureTime = f.DepartureTime
INNER JOIN airplane
ON f.AirplaneID = airplane.AirplaneID
INNER JOIN airport AS a1 ON a1.AirportName = f.DepartureAirport
INNER JOIN airport AS a2 ON a2.AirportName = f.ArrivalAirport
```

```
INNER JOIN flight AS f3
ON f.FlightNumber = f3.FlightNumber AND f.ArrivalAirport = f3.DepartureAirport
WHERE f.FlightNumber IN #check that its round trip
 (SELECT FlightNumber
 FROM flight as f2
 GROUP BY FlightNumber
 HAVING COUNT(f2.FlightNumber) > 1)
AND f3.DepartureDate > f.DepartureDate AND a1.AirportCity = %s AND f.DepartureAirport = %s
AND a2.AirportCity = %s AND f.ArrivalAirport = %s AND f.DepartureDate = %s AND
f3.DepartureDate = %s AND DepartureDate > CURRENT_DATE
GROUP BY f.AirlineName, f.FlightNumber, f.DepartureDate, f.DepartureTime, f.ArrivalDate,
FlightStatus, ReturnDate, ReturnTime
HAVING COUNT(ticketID) < NumberOfSeats
```

When a user clicks on View Flights, all flights, regardless of flight date (past or future) are shown with flight identifiers (FlightNumber, DepartureDate, DepartureTime) and their status. The query below helps us filter for corresponding flights and retrieves their status. For convenience, the flights are sorted by an ascending departure date.

- **View Flight Status**

```
SELECT AirlineName, FlightNumber, DepartureDate, DepartureTime, ArrivalDate, FlightStatus
FROM Flight AS f
NATURAL JOIN updates
ORDER BY DepartureDate
```

Register:

Customer Registration

In order for a customer to register for an account, we need to ensure that the customer does not have an existing account with the same username. A customer is identified by their email address, so when a customer enters their email address, the first thing that should be done is to check if it already exists in the account.

- **Find if customer has an existing account**

```
SELECT CustomerEmail
FROM customer
WHERE CustomerEmail = %s
```

In the case that the user does not have an existing account, we will add the customer to the 'customer' table in the database, along with other customer attributes, such as password, address and passport information.

- **If customer does not exist in the database, add customer attributes to the database**

```
INSERT INTO customer VALUES(%s, %s, md5(%s), %s, %s, %s, %s, %s, %s, %s, %s, %s)
```

Booking Agent Registration

To register a booking agent we need to make sure that the booking agent's account does not already exist. We check if the email that is used for the registration already exists in one of the registered accounts.

- **Check booking agent is not already registered**
  SELECT AgentEmail
  FROM bookingagent
  WHERE AgentEmail = %s

After it is confirmed that the account does not already exist, we then know we can register the booking agent account as a new account. So we add a new record using a unique booking agent email, password and agentID.

- **If booking agent not registered, insert info into system**
  INSERT INTO bookingagent VALUES(%s, md5(%s), %s)

Airline Staff Registration

To register an airline staff we first have to make sure that the airline staff is not already in the system. We do this by checking if the staff username already exists in the system.

- **Check if airline staff username exists in system**
  SELECT username
  FROM airlinestaff
  WHERE username = %s

After it is confirmed that the airline staff's username is not already in the system, then we can create an airline staff account using the values username, password, first and last name, date of birth, and phone number(s).

- **If airline staff is not registered then add staff into system**
  INSERT INTO airlinestaff VALUES(%s, md5(%s), %s, %s, %s, %s)

Additionally, we are given that airline staff are allowed to have multiple phone numbers. In the registration form, airline staff are instructed to enter multiple phone numbers separated by a semicolon. An airline staff's phone number is not stored in the airlinestaff table. It is stored (along with the airline staff's username) and inserted in a different table, phonenumber.

- **Add airline staff phone numbers**
  INSERT INTO phonenumber VALUES (%s, %s)

Login:

Customer Login

In order for a customer to log in, a customer only needs to enter their username and password. For protection, their passwords are hashed using MD5. Next, the customers table is queried using the username (email) and password that the customer entered. If the username and password entered by the customer matches a result in the customer table, the corresponding username and password pair is returned, which will allow the customer to access their home page.

- **Check if customer Email+Password combination match and exist in the customer table**
  SELECT CustomerEmail, CustomerPassword
  FROM customer
  WHERE CustomerEmail = %s AND CustomerPassword = md5(%s)

Booking Agent Login

Booking agents need to enter their email, password and Agent ID to log in. The 'bookingagent' table is searched for a matching combination. If all three fields are correct, the data is returned, which will allow the booking agent to log in and bring them to their home page.

- **Check if booking agent login credentials match the one in the booking agent table**

```
SELECT AgentEmail, AgentPassword, AgentID
FROM bookingagent
WHERE AgentEmail = %s and AgentPassword = md5(%s) AND AgentID = %s
```

Airline Staff Login

The airline staff login by entering their username and password. If the password is correctly matched with the username then the airline staff can successfully login and access the airline staff homepage.

- **Check if the airline staff credentials match with the account registered into the system**

```
SELECT Username, StaffPassword
FROM airlinestaff
WHERE Username = %s and StaffPassword = md5(%s)
```

<u>Customer use cases</u>

View My Flights:

For a customer to view their flights along with their status, we will query for useful information that a customer might need to identify a flight. Most importantly, we know that the FlightNumber, DepartureDate and the DepartureTime of a flight is the primary key for a flight, so it must be included in the output. Other attributes, such as AirlineName, ArrivalDate, ArrivalTime and FlightStatus were included for convenience. Given all the different flights in the system, we needed to filter out the flights the customer (who is logged in) must take. Additionally the query returns all future flights (Default View) that a customer has.

- **Displays flight attributes of future flights that a customer has bought a ticket for**

```
SELECT AirlineName, FlightNumber, DepartureDate, DepartureTime, ArrivalDate, ArrivalTime,
FlightStatus
FROM Flight
NATURAL JOIN updates
NATURAL JOIN purchasedfor
NATURAL JOIN ticket
NATURAL JOIN customer
WHERE CustomerEmail = %s AND DepartureDate >CURRENT_DATE  OR (DepartureDate =
CURRENT_DATE AND DepartureTime > CURRENT_TIME)
ORDER BY DepartureDate
```

Next, to take care of how the customer can view their flights in various ways, a method for the customer to view their past flights was implemented to complement the 'View Future Flights' function. The query is mostly the same as the query above, except in this case, we query for past flights by changing the condition in the WHERE clause. We change the filter to finding departure dates that are less than the current date.

- **Displays flight attributes of past flights that a customer has bought a ticket for**

```
SELECT AirlineName, FlightNumber, DepartureDate, DepartureTime, ArrivalDate, ArrivalTime,
FlightStatus
FROM Flight
NATURAL JOIN updates
NATURAL JOIN purchasedfor
NATURAL JOIN ticket
NATURAL JOIN customer
WHERE CustomerEmail = %s AND DepartureDate < CURRENT_DATE OR (DepartureDate =
CURRENT_DATE AND DepartureTime < CURRENT_TIME)
ORDER BY DepartureDate
```

Search for Flights: (Query is also reused for Booking Agents Search Flight)

Both customers and booking agents are allowed to search for flights (to eventually purchase). The customer/booking agent is prompted to enter source/destination airports and cities and a departure date. Flights that are fully booked or are classified as round trips are filtered out by the query.

- **Customer and booking agents search one way flights**

```
SELECT f.AirlineName, f.FlightNumber, f.DepartureDate, f.DepartureTime, ArrivalDate,
FlightStatus, COUNT(ticketID) as booked, numberOfSeats
FROM flight as f
LEFT JOIN purchasedfor AS p
ON p.FlightNumber = f.FlightNumber AND p.DepartureDate = f.DepartureDate AND
p.DepartureTime = f.DepartureTime
INNER JOIN updates AS u
ON u.FlightNumber = f.FlightNumber AND u.DepartureDate = f.DepartureDate AND
u.DepartureTime = f.DepartureTime
INNER JOIN airplane
ON f.AirplaneID = airplane.AirplaneID
INNER JOIN airport AS a1
ON a1.AirportName = f.DepartureAirport
INNER JOIN airport AS a2
ON a2.AirportName = f.ArrivalAirport
WHERE f.FlightNumber NOT IN
  (SELECT FlightNumber
  FROM flight as f2
  GROUP BY FlightNumber
  HAVING COUNT(f2.FlightNumber) > 1)
AND a1.AirportCity = %s AND f.DepartureAirport = %s AND a2.AirportCity = %s AND
f.ArrivalAirport = %s AND f.DepartureDate = %s
GROUP BY f.AirlineName, f.FlightNumber, f.DepartureDate, f.DepartureTime, ArrivalDate,
FlightStatus
HAVING booked < NumberOfSeats
```

Round trip flights that are in the database are fully searchable by a customer or booking agent. Similar to the query above, the search criteria remains the same, with an additional 'Return Date' feature. All one-way flights are filtered out and user-specified conditions are set.

- **Customer and booking agents search round trip flights**

```
SELECT f.AirlineName, f.FlightNumber, f.DepartureDate, f3.DepartureDate AS ReturnDate,
f.DepartureTime, f3.DepartureTime AS ReturnTime, f.ArrivalDate, FlightStatus, COUNT(ticketID)
as booked, numberOfSeats
FROM flight as f
LEFT JOIN purchasedfor AS p
ON p.FlightNumber = f.FlightNumber AND p.DepartureDate = f.DepartureDate AND
p.DepartureTime = f.DepartureTime
INNER JOIN updates AS u
ON u.FlightNumber = f.FlightNumber AND u.DepartureDate = f.DepartureDate AND
u.DepartureTime = f.DepartureTime
INNER JOIN airplane
ON f.AirplaneID = airplane.AirplaneID
INNER JOIN airport AS a1
ON a1.AirportName = f.DepartureAirport
INNER JOIN airport AS a2
ON a2.AirportName = f.ArrivalAirport
INNER JOIN flight AS f3
ON f.FlightNumber = f3.FlightNumber AND f.ArrivalAirport = f3.DepartureAirport
WHERE f.FlightNumber IN
  (SELECT FlightNumber
  FROM flight as f2
  GROUP BY FlightNumber
  HAVING COUNT(f2.FlightNumber) > 1)
AND f3.DepartureDate > f.DepartureDate AND a1.AirportCity = %s AND f.DepartureAirport = %s
AND a2.AirportCity = %s AND f.ArrivalAirport = %s AND f.DepartureDate = %s AND
f3.DepartureDate = %s
GROUP BY f.AirlineName, f.FlightNumber, f.DepartureDate, f.DepartureTime, f.ArrivalDate,
FlightStatus, ReturnDate, ReturnTime
HAVING COUNT(ticketID) < NumberOfSeats
```

Purchase Tickets:

Customers and booking agents are allowed to purchase tickets. The query below ensures that the one-way flight exists. If the flight exists, it will also check to see how many seats are filled to determine the price of the flight.

- **Customer and booking agent purchase one way flights**

```
SELECT f.AirlineName, f.FlightNumber, f.DepartureDate, f.DepartureTime, f.BasePrice,
f.ArrivalDate, f.ArrivalTime, f.ArrivalAirport, f.DepartureAirport, COUNT(ticketID) as booked,
numberOfSeats
FROM flight as f
LEFT JOIN purchasedfor AS p
```

```
ON p.FlightNumber = f.FlightNumber AND p.DepartureDate = f.DepartureDate AND
p.DepartureTime = f.DepartureTime
INNER JOIN updates AS u
ON u.FlightNumber = f.FlightNumber AND u.DepartureDate = f.DepartureDate AND
u.DepartureTime = f.DepartureTime
INNER JOIN airplane
ON f.AirplaneID = airplane.AirplaneID
INNER JOIN airport AS a1
ON a1.AirportName = f.DepartureAirport
INNER JOIN airport AS a2
ON a2.AirportName = f.ArrivalAirport
WHERE f.FlightNumber NOT IN
  (SELECT FlightNumber
  from flight as f2
  GROUP BY FlightNumber
  HAVING COUNT(f2.FlightNumber) > 1)
AND f.DepartureDate = %s AND f.DepartureTime = %s AND f.FlightNumber = %s
GROUP BY f.AirlineName, f.FlightNumber, f.DepartureDate, f.DepartureTime, ArrivalDate,
FlightStatus
HAVING booked < NumberOfSeats
```

To display select one way flight attributes to the user, specific flight information is queried again and shown to the user as they enter their card information.

```
SELECT *
FROM flight
WHERE FlightNumber = %s AND DepartureDate = %s AND DepartureTime =%s
```

- **Customer and Booking Agent Purchase Round Trip Flight (get the prices)**

Since each individual trip for a round trip flight does not have to be in the same airplane, the number of seats in the departure and return airplane can vary. Therefore, we must query for the number of seats booked in both the departure and arrival flights to determine the correct prices.

```
SELECT f.FlightNumber, f.DepartureDate, f.DepartureTime, COUNT(ticketID) AS booked,
numberOfSeats
FROM flight AS f
NATURAL JOIN airplane
LEFT JOIN purchasedfor AS p
ON f.DepartureDate = p.DepartureDate AND f.DepartureTime = p.DepartureTime AND
f.FlightNumber = p.FlightNumber
WHERE f.flightNumber = %s AND f.departureDate = %s AND f.departureTime = %s
GROUP BY  f.FlightNumber, f.DepartureDate, f.DepartureTime, numberOfSeats
```

Next, we will use the information from above to determine the price we should charge the customer. The following query is executed twice (one for outgoing and incoming).

```
SELECT *
```

FROM flight
WHERE FlightNumber = %s AND DepartureDate = %s AND DepartureTime =%s

- **Customer and booking agent enter card information to purchase flight**

In order for a customer to make a purchase, they will have to have a credit or debit card on file. Each card stored on file can be identified by their card number. If we find that a customer's card is already on file, we will not add it to the database. Else, we will add it to the database.

SELECT *
FROM cardinfo
WHERE cardNumber = %s

- **Add card information into the cardinfo**
  INSERT INTO cardinfo VALUES(%s, %s, %s, %s)

The ticket numbers are not auto-incremented in SQL, so we must query to find the maximum ticket number in the ticket table so far. We will increment this number in Python individually, so we will have to change its type (varchar) to int to perform numeric operations.

- **Find max ticket number in ticket**
  SELECT MAX(CAST(TicketID AS INT)) AS currTicket
  FROM ticket

Once the ticket number is determined, customer and flight information is added to the ticket, where each record is identified by a unique ticket number.

- **Add a ticket record into ticket**
  INSERT INTO ticket VALUES(%s, %s, %s, %s, %s, %s, %s, %s)

After creating a ticket for the requested flight, the payment method for the ticket will have to be logged into a table called paymentmethod.

- **Add customer email and card number used for purchase in paymentmethod table**
  INSERT INTO paymentmethod VALUES (%s, %s)

The purchasefor table lists all flights that customers have purchased/or booking agents have purchased for customers using ticketID and flight identifiers. A new flight that is purchased by the customer will have to be inserted into the purchasedfor table as a tracker.

- **Add ticket number and flight identifiers into purchasedfor**
  INSERT INTO purchasedfor VALUES (%s, %s, %s, %s)

In the case that a booking agent has bought the ticket for the customer, the booking agent will receive a 10% commission. The creates table keeps track of the amount of commission a booking agent receives from booking customer flights.

- **Insert booking agent commission into creates**
  INSERT INTO creates VALUES (%s, %s, %s)

Give Ratings and Comment on previous flights:
We prompt the user to enter a unique ticket number and a corresponding rating and suggestion. The system first checks if the flight's departure date is in the past.

- **Check ticket/flight exists and bought by customer and date is in the past**
  SELECT FlightNumber, DepartureDate, DepartureTime
  FROM ticket
  NATURAL JOIN purchasedfor
  NATURAL JOIN customer
  WHERE CustomerEmail = %s AND TicketID = %s AND (CURRENT_DATE > DepartureDate OR (CURRENT_DATE = DepartureDate AND CURRENT_TIME > DepartureTime))

We also query for ratings that are associated with the ticketID/flight. A customer should not be able to rate a flight more than once.

- **Check that a flight is not rated more than once (based on return value)**
  SELECT FlightNumber, DepartureDate, DepartureTime, TicketID
  FROM suggested
  NATURAL JOIN ticket
  WHERE CustomerEmail = %s AND TicketID = %s

If we see that there are no ratings from the customer for the flight that customer has already taken, we will add the customer's rating to the suggested table.

  INSERT INTO suggested VALUES(%s, %s, %s, %s, %s, %s)

Track My Spending:
The customer is able to see how much they spent in the past year. A simple query is used to find the amount.

- **Yearly spending of customer**
  SELECT SUM(SoldPrice) AS spend
  FROM ticket
  NATURAL JOIN purchasedfor
  WHERE CustomerEmail = %s AND PurchaseDate >= CURRENT_DATE - INTERVAL 1 YEAR

To find the amount a customer spent in the last 6 months:

- **Amount spent in the last 6 month, grouped by month**
  SELECT MONTHNAME(PurchaseDate) AS month, SUM(soldPrice) as spent
  FROM ticket
  NATURAL JOIN purchasedfor
  WHERE CustomerEmail = %s AND PurchaseDate >= CURRENT_DATE - INTERVAL 6 MONTH
  GROUP BY MONTHNAME(PurchaseDate)

And if the customer would like to specify custom dates to see their spending habits, additional constraints can be added:

- **Custom range of customer spending, grouped by month**

SELECT MONTHNAME(PurchaseDate) AS month, YEAR(PurchaseDate) AS year, SUM(soldPrice) as spent
FROM ticket
NATURAL JOIN purchasedfor
WHERE CustomerEmail = %s AND PurchaseDate >= %s AND PurchaseDate <= %s
GROUP BY MONTHNAME(PurchaseDate)

Booking Agent use cases

View My Flights:

Before viewing all the flights the booking agent booked for the customer, we first check if the customer exists in the system.

- **Check if the customer exists in the customer table**
  SELECT customeremail
  FROM customer
  WHERE customeremail = %s

After verifying the customer exists in the system, we then find the flights that the booking agent has booked for the customer.

- **If the customer exists, a list of their flights are returned with flight attributes and status**
  SELECT AirlineName, FlightNumber, DepartureDate, DepartureTime, ArrivalDate, ArrivalTime, FlightStatus
  FROM ticket
  NATURAL JOIN purchasedfor
  NATURAL JOIN customer
  NATURAL JOIN flight
  NATURAL JOIN updates
  WHERE CustomerEmail = %s

View my commision:

To find the average commission per ticket, we take the sum of the commission the booking agent received in the past 30 days then divide by the total number of tickets sold by the booking agent in the past 30 days.

- **Find total commission, average commission and number of tickets sold by a particular booking agent in the past 30 days**
  SELECT SUM(commissionAmount) AS totalcom, SUM(commissionAmount)/COUNT(*) as avgcom, COUNT(ticketID) AS numtickets
  FROM creates
  NATURAL JOIN ticket
  WHERE AgentEmail = %s AND CURRENT_DATE - 30 <= purchaseDate

To find the total commission received, average commission per ticket, and the total number of tickets sold by a specific booking agent, we find the sum of the commission received within the specified date range, then find the total number of tickets within the date range, and we divide the total commission by the number of tickets sold to find the average commission per ticket within the specified date range.

- **Find total commission, average commission and number of tickets sold by a particular booking agent for a specific range of dates**
  SELECT SUM(commissionAmount) AS totalcom, SUM(commissionAmount)/COUNT(*) as avgcom, COUNT(ticketID) AS numtickets
  FROM creates
  NATURAL JOIN ticket
  WHERE AgentEmail = %s AND PurchaseDate >= %s AND PurchaseDate <= %s
  Airline Staff Login & Registration
  #check if airline staff login credentials match the ones listed on the database
  SELECT Username, StaffPassword
  FROM airlinestaff
  WHERE Username = %s and StaffPassword = md5(%s)

View Top Customers:
The top customers are determined by the number of tickets bought from the booking agent in the past 6 months. We count the number of tickets purchased by customers using a booking agent then filter by the ones that were purchased within the last 6 months then we get the top 5 customers with the highest number of purchases.

- **Find top 5 customers by amount of tickets (in 6 months)**
  SELECT customerEmail, COUNT(ticketID) AS tickets
  FROM customer
  NATURAL JOIN ticket
  NATURAL JOIN creates
  WHERE agentemail = %s AND PurchaseDate >= CURRENT_DATE - INTERVAL 6 MONTH
  GROUP BY customerEmail
  ORDER BY COUNT(ticketID) DESC
  LIMIT 5

Similar to the previous query, instead of finding the top 5 customers by amount of tickets bought, in this query we find the top customers by the amount of commission the booking agent makes off of them.

- **Find top 5 customers by amount of commission (in a year)**
  SELECT customerEmail, SUM(commissionAmount) AS commission
  FROM customer
  NATURAL JOIN ticket
  NATURAL JOIN creates
  WHERE agentemail = %s AND PurchaseDate >= CURRENT_DATE - INTERVAL 1 YEAR
  GROUP BY customerEmail
  ORDER BY SUM(commissionAmount) DESC
  LIMIT 5

- **Booking Agent Search for Flights and Purchase Ticket**

The queries used in this section are exactly the same as the ones presented in customer, since both customers and booking agents use the same templates and protocols to purchase tickets.

Airline Staff use cases

View Flights:

An airline staff can only work for one airline. We will first identify the airline that an airline staff works for.

- **Find airline thar airline staff works for**

  ```
  SELECT AirlineName
  FROM airlinestaff
  WHERE Username = %s
  ```

After finding the airline the airline staff is associated with, we will filter for flights that are operated from that airline that will depart in the next 30 days. The results are intentionally sorted in ascending order so that sooner departing flights will show up at the top of the list.

- **View flights for staff in the next 30 days**

  ```
  SELECT DISTINCT FlightNumber, DepartureDate, DepartureTime, DepartureAirport, ArrivalDate, ArrivalTime
  FROM airlinestaff
  NATURAL JOIN flight
  WHERE AirlineName = %s AND (DepartureDate <= CURRENT_DATE + INTERVAL 30 DAY AND (DepartureDate > CURRENT_DATE)) OR (DepartureDate = CURRENT_DATE AND DepartureTime > CURRENT_TIME)
  ORDER BY DepartureDate ASC
  ```

Optionally, airline staff also has the choice to look for flights that depart in a specific time frame. The query below shows the constraints placed on flight departure dates (given a start and end date) to find specific flights.

- **View flights for staff in specific time frame**

  ```
  SELECT FlightNumber, DepartureDate, DepartureTime
  FROM flight
  WHERE airlineName = %s AND (DepartureDate > %s AND DepartureDate < %s OR DepartureDate = %s) #last one is start date
  ```

Airline staff can also view flights that arrive in a custom arrival or departure city. Airline staff can only view the flights for the airline they work with. The flight number, departure date and departure time is shown as an output.

- **View flights for airline staff with custom arrival airport or departure city**

  ```
  SELECT FlightNumber, DepartureDate, DepartureTime
  FROM flight AS f
  INNER JOIN airport AS a ON f.ArrivalAirport = a.AirportName
  INNER JOIN airport AS a2 ON f.DepartureAirport = a2.AirportName
  WHERE airlineName = %s AND (a.airportCity = %s OR a2.airportCity = %s)
  ORDER BY arrivalDate
  ```

Airline staff may also choose to view flights that arrive or depart from a specific airport. The following queries aid finding those flights.

- **Airline staff get flights from custom departure airport**
  SELECT FlightNumber, DepartureDate, DepartureTime
  FROM flight
  WHERE airlineName = %s AND DepartureAirport = %s

- **Airline staff get flights from custom arrival airport**
  SELECT FlightNumber, DepartureDate, DepartureTime
  FROM flight
  WHERE airlineName = %s AND ArrivalAirport = %s

Additionally, airline staff are able to see a list of customers that are taking a specific flight. To query for customers that are taking a specific flight, we use the flight number as an identifier to join to the customer and purchasedfor table.

- **Airline Staff: see list of customers on a flight**
  SELECT DISTINCT customerEmail
  FROM ticket
  NATURAL JOIN purchasedfor
  WHERE FlightNumber = %s AND DepartureDate = %s AND DepartureTime = %s AND AirlineName = %s

Note that in each of the airplane staff view flight use cases, we always make sure to check that a specific airline staff only has access to their own airline's flight.

View all the Booking Agents:
To be able to view the top 5 booking agents with the most sales in the last 6 months and year, we count the number of tickets sold by each booking agent in the past year then rank them by most tickets sold and taking only the top 5. We sort it by descending order to show the highest ranking first.

- **Find top 5 agents by ticket sales in past 6 months**
  SELECT AgentEmail, COUNT(*) AS ticketSold
  FROM ticket
  NATURAL JOIN creates
  WHERE purchaseDate >= CURRENT_DATE - INTERVAL 1 MONTH
  GROUP BY AgentEmail
  ORDER BY COUNT(AgentID)
  DESC LIMIT 5

- **Find top 5 agents by ticket sales in last year**
  SELECT AgentEmail, COUNT(*) AS ticketSold
  FROM ticket
  NATURAL JOIN creates
  WHERE purchaseDate >= CURRENT_DATE - INTERVAL 1 YEAR
  GROUP BY AgentEmail
  ORDER BY COUNT(AgentID) DESC
  LIMIT 5

To be able to view the top 5 booking agents with the most commission earned in the last year, we sum up the commission earned by each booking agent in the past year then rank them by most commission earned and take only the top 5.

- **Find top agents in last year by total commission**

```
SELECT AgentEmail, SUM(commissionAmount) AS commission
FROM ticket
NATURAL JOIN creates
WHERE purchaseDate >= CURRENT_DATE - INTERVAL 1 YEAR
GROUP BY AgentEmail
ORDER BY commission DESC
LIMIT 5
```

View Frequent Customers:

To view the customers based on the customers that took the most flights, we count the number of flights each customer has for that specific airline in the past year then rank them based on the number of flights and taking only the top customer with the most flights.

- **Find top customer by number of flights taken in past year**

```
SELECT CustomerEmail, COUNT(*) AS numFlights
FROM ticket
WHERE AirlineName = %s AND purchaseDate >= CURRENT_DATE - INTERVAL 1 YEAR
GROUP BY CustomerEmail
ORDER BY numFlights DESC
LIMIT 1
```

We specify a customer and an airline to find all the flights a customer has taken on that specific airline. The flights are specified by their flight number, departure date, and departure time.

- **Find all flights customer has taken on a particular airline**

```
SELECT FlightNumber, DepartureDate, DepartureTime
FROM Flight
NATURAL JOIN updates
NATURAL JOIN purchasedfor
NATURAL JOIN ticket
NATURAL JOIN customer
WHERE CustomerEmail = %s AND airlineName = %s
ORDER BY DepartureDate
```

Create New Flights:

To create a new flight we first need to check if the flight already exists in the system. We do this by checking if the flight number, departure date, and departure time matches with a flight already in the system.

- **Query to see if flight already exists in system**

```
SELECT *
FROM flight
```

WHERE FlightNumber = %s AND DepartureDate = %s AND DepartureTime = %s

If the flight does not already exist in the system then we can add it in with the values for the airline name, flight number, departure airport, departure date, departure time, arrival airport, arrival date, arrival time, base price, and airplane ID.

- **Insert flight with given attributes into the flight table**
  INSERT INTO flight VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)

After creating a flight the flight arrival information needs to be put into the system.

- **Insert flight information into the arrives table**
  INSERT INTO arrives VALUES (%s, %s, %s, %s)

After creating the flight the flight departure information needs to be put into the system.

- **Insert flight information into the departs table**
  INSERT INTO departs VALUES (%s, %s, %s, %s)

After creating the flight the flight needs to be entered into the updates table to be able to update the flight status.

- **Insert flight entry into the updates table**
  INSERT INTO updates VALUES (%s, %s, %s, %s, %s)

Change Status of Flights:

To create a new flight we first need to check if the flight exists in the system. We do this by checking if the flight number, departure date, and departure time matches with a flight already in the system.

- **Check if flight exists in flight table (same as previous queries)**
  SELECT *
  FROM updates
  WHERE FlightNumber = %s AND DepartureDate = %s AND DepartureTime = %s

After making sure the flight is in the system then we can update the status of the flight. We set the status of a flight that is specified by the flight number, departure date, and departure time.

- **Update the value in updates table**
  UPDATE updates SET FlightStatus = %s, Username = %s WHERE FlightNumber = %s AND DepartureDate = %s AND DepartureTime = %s

Add Airplane in the System:

Before adding an airplane in the system we first need to make sure that the airplane does not exist in the system. We check the airplane by the airline name and the airplane ID.

- **Check if airplane exists**
  SELECT *
  FROM airplane
  WHERE AirlineName = %s AND AirplaneID = %s

After making sure the airplane is not in the system we can now add the airplane in the system by the values airline name, airplane ID, and number of seats.

- **Add airplane to system**
  INSERT INTO airplane VALUES(%s, %s, %s)

To display the list of airplanes in the confirmation page, we will need another query to get the list of airplanes that operate under the airline. We filter on airlineName to only retrieve the airplanes that exist on the airline the staff work for.

- **View airplanes of the airline the airline staff works for**
  SELECT *
  FROM airplane
  WHERE airlineName = %s

Add New Airport in the System:
Before adding an airport in the system we first need to make sure the airport does not exist. We do this by checking if the airport name exists in the system.

- **Check if specific airport already exists**
  SELECT * FROM airport WHERE AirportName = %s

After making sure the airport does not exist in the system we can add in the airport by the values airport name and airport city.

- **Insert airport into airport table**
  INSERT INTO airport VALUES(%s, %s)

View Flight Ratings:
To view the average rating for a specific flight, we specify the airline, flight number, departure date, and departure time then take the average rating by averaging the ratings that were received for that specific flight. We then display the flight and its airline with the average rating it received.

- **Get average flight ratings**
  SELECT AirlineName, FlightNumber, DepartureDate, DepartureTime, AVG(Rate) as averageRating
  FROM suggested
  NATURAL JOIN Flight
  WHERE AirlineName = %s
  GROUP BY AirlineName, FlightNumber, DepartureDate, DepartureTime

To view the ratings of a specific flight, the user input the flight that is identified by its flight number, departure date, and departure time and returns all the rates and customer comments for that specific flight.

- **View flight ratings**
  SELECT Rate, CustomerComment
  FROM suggested
  WHERE FlightNumber = %s AND DepartureDate = %s AND DepartureTime = %s

View Reports:
Airline staff are allowed to view the behavior of ticket sales. A query is used to find the amount of tickets that were sold in the last month/last year and a custom range of dates. The data is grouped by month as needed.

- **Get amount of tickets sold last month, return a number**
  SELECT COUNT(TicketID) AS tickets, MONTHNAME(PurchaseDate) AS month
  FROM ticket
  WHERE PurchaseDate >= CURRENT_DATE - INTERVAL 1 MONTH
  GROUP BY month

- **Get amount of tickets sold last year, grouped by month**
  SELECT COUNT(TicketID) AS tickets, MONTHNAME(PurchaseDate) AS month
  FROM ticket
  WHERE PurchaseDate >= CURRENT_DATE - INTERVAL 1 YEAR
  GROUP BY month

In this case, we will add a filter to let the staff specify the range of the purchase dates. The query counts the number of tickets sold grouped by month. Due to the unpredictable nature of when tickets will be bought, months where there were no purchases made will not be shown.

- **Get indirect and direct sale prices from custom range of dates**
  SELECT COUNT(TicketID) AS tickets, MONTHNAME(PurchaseDate) AS month,
  YEAR(PurchaseDate) AS year
  FROM ticket
  WHERE PurchaseDate >= %s AND PurchaseDate <= %s
  GROUP BY month, year

Comparison of Revenue Earned:
The indirect revenue is the revenue earned when customers bought tickets using a booking agent. The direct revenue is the revenue earned when customers bought tickets themselves. We find the indirect revenue by calculating the sum of all the ticket prices where a booking agent was used and falls within the last month. We find the direct revenue by calculating the sum of all the ticket prices where a booking agent was not used and falls within the last month.

- **Find indirect and direct revenue in the past month**
  SELECT SUM(SoldPrice) as sale
  FROM ticket #for indirect revenue, bought through booking agent, %s is not NULL
  WHERE AgentID <> %s AND PurchaseDate >= CURRENT_DATE - INTERVAL 1 MONTH
  UNION
  SELECT SUM(SoldPrice) as sale
  FROM ticket #direct sale where %s is NULL
  WHERE AgentID = %s AND PurchaseDate >= CURRENT_DATE - INTERVAL 1 MONTH

We find the indirect revenue by calculating the sum of all the ticket prices where a booking agent was used and falls within the last year. We find the direct revenue by calculating the sum of all the ticket prices where a booking agent was not used and falls within the last year.

- **Find indirect and direct revenue in the past year**
  SELECT SUM(SoldPrice) as sale
  FROM ticket #indirect sale where %s is not NULL
  WHERE AgentID <> %s AND PurchaseDate >= CURRENT_DATE - INTERVAL 1 YEAR
  UNION
  SELECT SUM(SoldPrice) as sale
  FROM ticket #direct sale where %s is NULL
  WHERE AgentID = %s AND PurchaseDate >= CURRENT_DATE - INTERVAL 1 YEAR

View Top Destinations:
To view the top 3 destinations in the last 3 months we first look through the arrival destinations for all the tickets on a certain airline. Then we count the number of tickets that had its destination in an airport in that certain city where the arrival date is within the last year. We then order them by the destinations with the most tickets at that location and take the top 3.

- **Top 3 destinations in the last 3 months**
  SELECT AirportCity
  FROM ticket
  NATURAL JOIN purchasedfor
  NATURAL JOIN flight
  INNER JOIN airport
  ON arrivalAirport = airport.AirportName
  WHERE AirlineName = %s AND arrivalDate >= CURRENT_DATE - INTERVAL 3 MONTH
  GROUP BY AirportName
  ORDER BY COUNT(airportName) DESC
  LIMIT 3

To view the top 3 destinations in the last year we first look through the arrival destinations for all the tickets on a certain airline. Then we count the number of tickets that had its destination in an airport in that certain city where the arrival date is within the last year. We then order them by the destinations with the most tickets at that location and take the top 3.

- **Top 3 destinations in the last year**
  SELECT AirportCity
  FROM ticket
  NATURAL JOIN purchasedfor
  NATURAL JOIN flight
  INNER JOIN airport
  ON arrivalAirport = airport.AirportName
  WHERE AirlineName = %s AND arrivalDate >= CURRENT_DATE - INTERVAL 1 YEAR
  GROUP BY AirportName
  ORDER BY COUNT(airportName) DESC
  LIMIT 3

Logout:
All three logout scenarios do not execute queries; this is done in Python.