Team member: Kevin Zhou , Junwen Zhong

Below is a list of all the use cases and corresponding SQL queries in our project:

Home page when not logged-in:

1. Public Info
   a. Search for future flights based on source city/airport name, destination city/airport name, departure date for one way (departure and return dates for round trip):
   The user would have the option to choose whether to search by airport or by city. The system would then store the information user entered and use them to run the query `'SELECT airline_name, flight_number, departure_date, departure_time, arrival_date, arrival_time, flight_status FROM Flight where departure_airport in (select airport_code from Airport where city=%s) and arrival_airport in (select airport_code from Airport where city=%s) and departure_date=%s'`
   Assuming the user is searching by city. The results of the query would contain all the flights that fit the description. If we switch the input sequence of source city/airport and destination city/airport, and substitute departure date with return date, the results would then be all the available returning flights for the users. The results would be displayed on a separate html page, and each row represents a flight. There will also be a "purchase" option for each row, which will redirect the user to the customer sign-in page if they are not yet logged in.

   b. See the flights status based on airline name, flight number, arrival/departure date:
   This feature is achieved through a "view" generated in the database: `"create view public_info as select airline_name, flight_number, departure_date, arrival_date, flight_status from Flight;"`.
   The system would then query all the information within that "view" and display it on the main page.

2. Register
   a. Customer register:
   The customer need to fill in a form. Email, password, first name, last name and zipcode are required, while other information like address and phone number are not required. If the customer does not fill this part, it simply store a null value.
   The system first check if the customer already has an account by `'SELECT * FROM Customer WHERE email = %s'`. If it already exists, the system will show an error

message. Otherwise, the system add the new customer into database by `"INSERT INTO Customer VALUES(...)"`depending on whether the values are null or not.
The system also stores the customer's phone numbers in database by `"INSERT INTO Customer_phone VALUES('" + email + "', " + phones[i] + ")"`.
After register, the system redirects the customer to customer home page, and a session is initiated with the member's email stored as a session variable. Also, the system sets the log in status to "online" for this customer in the database by `"UPDATE Customer SET log_in_status='online' WHERE email='" + email + "';"`.

b. Staff register:
The staff need to fill in a form with username, password, etc. All information are required. The system also asks the staff for an airline authentication code from their airline, so unauthorized staffs are not able to register an account. If the authentication code is wrong, the system will show an error message.
The system first checks if the staff already has an account by `'SELECT * FROM Airline_staff WHERE username = %s'`. If it already exists, the system will show an error message. Otherwise, the system add the new customer into database by `"INSERT INTO Airline_staff VALUES(...)"`depending on whether the values are null or not.
The system also stores the staff's phone numbers and email addresses by `"INSERT INTO Staff_phone VALUES('" + username + "', " + thephones[i] + ")"` and `"INSERT INTO Staff_email VALUES('" + username + "', '" + theemails[i] + "')"`

3. Login:

a. Customer login:
The customer need to fill in a form with email and password. The system checks if the pair is correct by `'SELECT * FROM Customer WHERE email = %s and customer_password = %s'`. If it is incorrect, the system will show an error message. Otherwise, it redirects the customer to customer home page, and a session is initiated with the member's email stored as a session variable. Also, the system sets the log in status to "online" for this customer in the database by `"UPDATE Customer SET log_in_status='online' WHERE email='" + email + "';"`.

b. Staff login:
The staff need to fill in a form with username and password. The system checks if the pair is correct by `'SELECT * FROM Airline_staff WHERE username = %s and staff_password = %s'`. If it is incorrect, the system will show an error message. Otherwise, it redirects the customer to staff home page, and a session is initiated with the staff's username stored as a session variable.

Customer use cases:

In all customer use cases, the system first check if the customer is logged in. If the customer hasn't logged in yet, it will redirect to the customer login page.

1. View My flights:
   Provide various ways for the user to see flights information which he/she purchased. The default should be showing for the future flights. Optionally you may include a way for the user to specify a range of dates, specify destination and/or source airport name or city name etc.
   The system runs a default query when user enter the page `"SELECT airline_name, flight_number, departure_date, departure_time FROM Purchase NATURAL JOIN Ticket where ((departure_date = NOW() and departure_time > NOW()) or (departure_date > NOW())) and email=%s"` with the user's email address which is stored in the session as they log in as a customer. The results would display all the future flights the customer has purchased. However, the drop-down bar at the top of the page also lets the customer choose to see the past flights they have purchased, or all the flights in their purchase history, through similar queries.

2. Search for flights:
   Search for future flights (one way or round trip) based on source city/airport name, destination city/airport name, dates (departure or return).
   Merged with "Purchase tickets" feature

3. Purchase tickets:
   Customer chooses a flight and purchase ticket for this flight, providing all the needed data, via forms. You may find it easier to implement this along with a use case to search for flights.
   The system would first redirect the user to the "search flight" page, which allows the user to choose from a range of flights which have the source, destination, and date that the user entered. After the user clicks on the purchase button of a flight, they would be directed to a page where they are shown the ticket price and asked for their information as well as the payment information. The price of the ticket would be calculated based on the seat occupancy of the flight. This information is checked through the query `"SELECT base_price, tickets_booked, seats from Flight NATURAL JOIN Airplane where airline_name=%s and flight_number=%s and departure_date=%s and departure_time=%s"`; we then calculate the ratio of "tickets_booked" and "seats", if more than 80% of the seats are booked, then the price would be 1.25 times the base price.

After all the information is collected, we run the query `"insert into Ticket values(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s);"` to insert a row with the information into the "Ticket" table, and we increment the largest ticket id by 1 as the new id for the ticket the user is purchasing. Eventually, we use the query `"insert into Purchase values (%s, %s, null, null);"` to insert the ticket into "Purchase" table with the customer's email.

4. Cancel Trip:
   Customer chooses a purchased ticket for a flight that will take place more than 24 hours in the future and cancel the purchase. After cancellation, the ticket will no longer belong to the customer. The ticket will be available again in the system and purchasable by other customers.
   The system would first run the query `"SELECT id, airline_name, flight_number, departure_date, departure_time FROM Purchase NATURAL JOIN Ticket where timestamp(concat(departure_date, ' ', departure_time)) > (NOW() + INTERVAL 24 HOUR) and email=%s"` which displays all the flights that departs at least 24 hours from now. The customer would be able to choose a flight to delete. The system first deletes from "Purchase" table: `"DELETE FROM Purchase where email=%s and id=%s"`, and then deletes the ticket: `"DELETE FROM Ticket where id=%s"`

5. Give Ratings and Comment on previous flights:
   Customer will be able to rate and comment on their previous flights (for which he/she purchased tickets and already took that flight).
   System would run the query `"SELECT T.id, F.airline_name, F.flight_number, F.departure_date, F.departure_time, rate, comments FROM Purchase NATURAL JOIN Ticket as T INNER JOIN Flight as F where (T.airline_name=F.airline_name and T.flight_number=F.flight_number and T.departure_date=F.departure_date and T.departure_time=F.departure_time) and timestamp(concat(arrival_date, ' ', arrival_time)) < NOW() and email=%s"` to display all the past flights the customer has taken. Then, the user can choose one to rate and comment. They may give a rating out of 10 points, and write a review comment before posting. After that, the system will update the "Purchase" table with their rating and comment: `"UPDATE Purchase SET rate=%s, comments=%s where id=%s and email=%s"`, and they will see their review being displayed on the front page.

6. Track My Spending:
   Default view will be total amount of money spent in the past year and a bar chart/table showing month wise money spent for last 6 months. He/she will also have option to specify a range of dates to view total amount of money spent within that range and a bar chart/table showing month wise money spent within that range.
   System would execute the query `"SELECT MONTHNAME(purchase_date) as month, sum(price) as spending FROM Purchase NATURAL JOIN Ticket where purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR) and email=%s GROUP BY MONTH(purchase_date) ORDER by MONTH(purchase_date) ASC;"` when a user enter the page, which display the monthly spending of the user within the past year in a table, along with the total amount of spending at the bottom. The customer can set a range of dates at the top of the page, and with the information given, system would execute the query `"SELECT MONTHNAME(purchase_date) as month, sum(price) as spending FROM Purchase NATURAL JOIN Ticket where purchase_date >= %s and purchase_date <= %s and email=%s GROUP BY MONTH(purchase_date) ORDER by MONTH(purchase_date) ASC;"` which displays the monthly spending within that range in a table and the total spending at the bottom.

7. Logout:
   If the customerclick "logout", the system will redirect he/she to the customer login page, where he/she can login again or go to the main page. Also, the system will delete the customer session from session dictionary. Addistionally, the system sets the log in status to "offline" for this customer in the database by `"UPDATE Customer SET log_in_status='offline' WHERE email='" + email + "';"`.

Airline Staff use cases:

In all airline staff use cases, the system first check if the person is logged in as a staff. If the person hasn't logged in yet, it will redirect to the staff login page. We do so to prevent customer or other people to access staff use cases.

Also, in most cases, we use `'SELECT airline_name FROM Airline_staff WHERE username = %s'` to get the airline name that the logged in staff works for. We need it to search in the database or add new rows to the database.

1. View flights:
   Defaults will be showing all the future flights operated by the airline he/she works for the next 30 days. The system gets the information by `"SELECT * FROM Flight WHERE airline_name=%s and (departure_date BETWEEN CURDATE() AND DATE_ADD(CURDATE(), INTERVAL 30 DAY))"`.
   On the page, there are also two links "search flight" and "search customers in flight". They lead to the following two use cases:
   (a) Search flight:
   On this page, the staff is able to search for flights operated by the airline he/she works. He can select to search by range of dates, departure/arrival airports, or both of them. For example, it the staff selected both, the system will get the information by `"SELECT * from Flight WHERE airline_name=%s and departure_airport=%s and arrival_airport=%s and (departure_date BETWEEN %s AND %s); "`, and show the search result on the page.

   (b) Search customers in flight:
   On this page, the staff is able to search for a specific flight by flight number, departure date and departure time. The system will show all the customers on it. The query is `"SELECT first_name, last_name FROM Ticket WHERE flight_number=%s and departure_date=%s and departure_time=%s and airline_name=%s"`.

2. Create new flights:
   The staff can create a new flight on this page by providing all the needed data via forms. The system first check if the flight already exist by `"SELECT * FROM Flight WHERE flight_number = %s and departure_date = %s and departure_time = %s and airline_name = %s"`. If the same flight already exists, the system will show an error message. Otherwise, the system add the new flight

by `"INSERT INTO Flight VALUES(...)"` with the information provided by the staff. After the staff successfully add the new flight, the system goes to "view flights" page.

3. Change Status of flights:
   The staff can change a flight status (from on-time to delayed or vice versa) via forms on this page. The system do this by `"UPDATE Flight SET flight_status=%s WHERE flight_number=%s and departure_date=%s and departure_time=%s and airline_name=%s"`. After the change, the system goes to "view flights" page, where the staff can see the future flights to see if he/she has changed it successfully, or just to do other searches.

4. Add airplane in the system:
   The staff can add a new airplane by providing all the needed data via forms. Using the information provided, the system first check if the airplane already exists by `"SELECT * FROM Airplane WHERE id = %s"`. If it already exists, the system will show an error message. Otherwise, the airplane can be added successfully by `"INSERT INTO Airplane VALUES(%s, %s, %s, %s, null, %s)"`. (The `null` is for age of the airplane, which is calculated automatically). The staff will be redirected to the confirmation page ("view airplanes"). In the confirmation page, the staff will be able to see all the airplanes owned by the airline he/she works for. The staff can access this "view airplanes" page from staff home page directly as well. This page get all the information for planes from database by `"SELECT id FROM Airplane WHERE airline_name='" + airline_name + "'"`.

5. Add new airport in the system:
   The staff can add a new airport by providing all the needed data via forms. Using the information provided, the system first check if the airport already exists by `"SELECT * FROM Airport WHERE airport_code = %s"`. If it already exists, the system will show an error message. Otherwise, the airport can be added successfully by `"INSERT INTO Airport VALUES(%s, %s, %s, %s, %s)"`. The staff will be redirected to the confirmation page ("view airports"). In the confirmation page, the staff will be able to see all the available airports including the one he/she just added. The staff can access this "view airports" page from staff home page directly as well. This page get all the information for airports from database by `"SELECT * FROM Airport"`.

6. View flight ratings:
   The staff can see each flight's average ratings on this page. The system calculate the average rating for each flight number in the history, rather than calculate the average

rating of a specific flight. The system get the average values by `"SELECT AVG(rate), flight_number FROM Purchase natural join Ticket where airline_name=%s GROUP BY flight_number"`.

7. View frequent customers:
   The staff can see the most frequent customer within the last month or last year on this page. The system get the data for last month by `"SELECT email, COUNT(*) as count FROM Purchase natural join Ticket where airline_name=%s GROUP BY email ORDER BY count DESC; "`, and get the data for last year by `"SELECT email, COUNT(*) as count FROM Purchase NATURAL JOIN Ticket WHERE airline_name = %s AND departure_date BETWEEN DATE_ADD(CURDATE(), INTERVAL -365 DAY) AND CURDATE() GROUP BY email ORDER BY count DESC; "`. The tables are sorted by count of purchases, so that the customer who has purchased the most tickets is on the top.

8. View reports:
   The staff can see total amounts of ticket sold based on range of dates on this page. There is a form for the staff to provide the range of dates he/she want to search. And using the date range, the system get the data by `"SELECT COUNT(*) as count FROM Ticket WHERE airline_name=%s and (purchase_date BETWEEN %s AND %s); "`.

9. View Earned Revenue:
   The staff can see total amount of revenue earned from ticket sales in total, in the last month and in the last year. The system gets these data separately by: `"SELECT SUM(price) as total FROM Purchase NATURAL JOIN Ticket WHERE airline_name = %s; "`; `"SELECT SUM(price) as total FROM Purchase NATURAL JOIN Ticket WHERE airline_name = %s AND departure_date BETWEEN DATE_ADD(CURDATE(), INTERVAL -30 DAY) AND CURDATE(); "`, and `"SELECT SUM(price) as total FROM Purchase NATURAL JOIN Ticket WHERE airline_name = %s AND departure_date BETWEEN DATE_ADD(CURDATE(), INTERVAL -365 DAY) AND CURDATE(); "`.

10. Logout:
    If the staff click “logout”, the system will redirect he/she to the staff login page, where he/she can login again or go to the main page. Also, the system will delete the staff session from session dictionary.