Customer Use Cases

1. View My Flights
    a. SELECT ticket_ID, flight_num, departure_date, departure_time, airline_name, first_name, last_name, flight_status FROM ticket_bought_by natural join ticket natural join flight WHERE email_address = ? AND (CURRENT_DATE < DEPARTURE_DATE OR (CURRENT_DATE = departure_date AND CURRENT_TIME < departure_time));
        i. This query finds all the future flights that the user has purchased tickets for.
    b. SELECT ticket_ID, flight_num, departure_date, departure_time, airline_name, first_name, last_name, flight_status FROM Ticket NATURAL JOIN Ticket_Bought_By NATURAL JOIN Flight WHERE email_address = ? AND (CURRENT_DATE > DEPARTURE_DATE);
        i. This query finds all the past flights that the user has purchased tickets for.
    c. SELECT ticket_ID, flight_num, departure_date, departure_time, airline_name, first_name, last_name, flight_status FROM Ticket NATURAL JOIN Ticket_Bought_By NATURAL JOIN Flight WHERE email_address = ? AND (CURRENT_DATE = DEPARTURE_DATE);
        i. This query finds all the flights that are happening on today's date.
2. Search for flights
    a. SELECT * FROM ticket where ticket_id not in (SELECT ticket_id FROM ticket_bought_by) AND flight_num IN (SELECT flight_num FROM flight where departure_airport = ? and arrival_airport = ? and departure_date = ?);

i. This query finds all the tickets that haven't been bought based on the date, departure, and arrival airport.

b. SELECT flight_num, count(*) as unsold, num_of_seats FROM flight NATURAL JOIN airplane NATURAL JOIN ticket WHERE (departure_airport = ? and arrival_airport = ? and departure_date = ?) AND (ticket_id NOT IN (select ticket_id from ticket_bought_by)) GROUP BY flight_num"

i. This query counts the number of unsold tickets and how many seats are on the plane. Used for checking for 80% capacity

c. UPDATE ticket SET price = ? WHERE (ticket_ID = ?) and (flight_num = ?) and (departure_date = ?) and (departure_time = ?) and ? not in (SELECT ticket_id FROM ticket_bought_by)

i. This query multiplies the current price by 1.25 if the flight is at least 80% filled up already

3. Purchase Tickets

a. INSERT INTO Ticket_Bought_By VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, CURRENT_DATE, CURRENT_TIME);

i. This query inserts the information regarding the tickets bought into the ticket_bought_by table along with the current_date and current_time.

4. Cancel Trip

a. DELETE FROM Ticket_Bought_By WHERE ticket_id = ?;

i. This query deletes the entry from the ticket_bought_by table based on the ticke_id of the ticket selected to cancel by the user.

5. Give Ratings and Comments on previous flights

a. INSERT INTO Review VALUES (?,?,?,?,?,?);
    i. This query inserts into the review table, the information given by the front-end when a review is inserted.
b. SELECT flight_num, rating, comments FROM Review WHERE email_address = ?;
    i. This query gets all the reviews made by the user based on their email_address

6. Track My Spending
    a. select sum(price) as total from ticket_bought_by natural join ticket where email_address = ? and purchase_date BETWEEN DATE_SUB(current_date(), INTERVAL 1 YEAR) AND current_date()
        i. This query gets the total price of all the tickets that were purchased in the past year
    b. select price as market_price, purchase_date, MONTH(purchase_date) as purchase_month, YEAR(purchase_date) as purchase_year FROM flight JOIN airplane ON flight.airplane_id = airplane.airplane_id JOIN ticket as t ON t.flight_num = flight.flight_num JOIN ticket_bought_by as tbb on t.ticket_id = tbb.ticket_ID WHERE email_address = ? AND tbb.purchase_date BETWEEN DATE_SUB(current_date(), INTERVAL 6 MONTH) AND current_date();
        i. This query gets all the purchase information from the last 6 months including price, purchase date, purchase month, and purchase year. Used for mapping out each month for the monthly spending reports.

c. SELECT price as market_price, purchase_date, MONTH(purchase_date) as purchase_month, YEAR(purchase_date) as purchase_year FROM flight JOIN airplane ON flight.airplane_id = airplane.airplane_id JOIN ticket as t ON t.flight_num = flight.flight_num JOIN ticket_bought_by as tbb on t.ticket_id = tbb.ticket_ID WHERE email_address = ? AND tbb.purchase_date BETWEEN ? AND ? GROUP BY t.price, flight.flight_num, tbb.purchase_date, airplane.airplane_id, airplane.num_of_seats;

    i. This query does the same as the previous, except instead of selecting the data from the last 6 months, it selects data in between the inputted range

7. Logout

    a. No queries are needed.

Staff Use Cases

1. View Flights

    a. SELECT * FROM flight WHERE airline_name = ? AND ((departure_date > CURRENT_DATE) OR (departure_date = CURRENT_DATE AND departure_time > CURRENT_TIME))

        i. This query finds all the future flights that the airline has coming up.

    b. SELECT * FROM flight WHERE airline_name = ? AND departure_date < CURRENT_DATE

        i. This query finds all the past flights that the airline has.

    c. SELECT * FROM flight WHERE airline_name = ? AND departure_date = CURRENT_DATE

i. This query finds all the flights that the airline has happening today.

2. Create New Flights
    a. INSERT INTO flight VALUES (?,?,?,?,?,?,?,?,?,?)
        i. This query inserts the flight_num, departure_date, departure_time, arrival_date, arrival_time, departure_airport, arrival_airport, airline_name, airplane_ID, flight_status given by the user into the Flights table
3. Change Status of Flights
    a. UPDATE flight SET flight_status = ? WHERE (flight_num = ?) and (departure_date = ?) and (departure_time = ?);
        i. This query updates the flight status of a given flight based on the flight number, status, and departure date and time given by the user.
4. Add airplane in the system
    a. INSERT INTO airplane VALUES (?,?,?,?,?,?,?);
        i. This query inserts into the airplane table, the new airplane created by the staff user with the data provided.
5. Add new airport in the system
    a. INSERT INTO airport VALUES (?,?,?,?,?);
        i. This query inserts into the airport table, the new airport created by the staff user with the data provided.
6. View Flight Ratings
    a. SELECT avg(rating) as average FROM review WHERE (flight_num = ?) and (departure_date = ?) and (departure_time = ?)

i. This query gets the average rating based on the flight information given by the user from the review table.

b. SELECT * FROM review WHERE (flight_num = ?) and (departure_date = ?) and (departure_time = ?)

i. This query gets all the information related to a specific rating given by the user.

7. View frequent customer

a. SELECT * FROM Ticket_Bought_By NATURAL JOIN Ticket WHERE departure_date >= DATE_SUB(NOW(),INTERVAL 1 YEAR) AND airline_name = ?;

i. This query gets all the information regarding flights in the past year based on the airline the staff works for and where the departure date is greater than or equal to date from a year ago.

8. View Reports and Earned Revenue

a. SELECT COUNT(ticket_ID), SUM(price) FROM Ticket NATURAL JOIN Ticket_Bought_By WHERE airline_name = ? AND purchase_date BETWEEN DATE_SUB(NOW(),INTERVAL 1 YEAR) AND CURRENT_DATE

i. This query counts the number of tickets sold within the past year and sums the prices the tickets sold for in order to see how much money was made

b. SELECT COUNT(ticket_ID), SUM(price) FROM Ticket_Bought_By NATURAL JOIN Ticket WHERE airline_name = ? AND purchase_date BETWEEN DATE_SUB(NOW(),INTERVAL 1 MONTH) AND CURRENT_DATE;

i. This query counts the number of tickets sold within the past month and sums the prices the tickets sold for in order to see how much money was made

c. SELECT purchase_date , COUNT(purchase_date) FROM Ticket_Bought_By NATURAL JOIN Ticket WHERE airline_name = ? AND purchase_date BETWEEN ? AND ? GROUP BY purchase_date

i. This query counts the number of tickets sold within a date range and sums the prices the tickets sold for in order to see how much money was made

9. Logout

a. No queries are needed.