Use case summary

Team members:
Yanglin Tao (yt2061@nyu.edu)
Tianzuo Liu (tl3119@nyu.edu)
Yuting (Justin) Li (yl7685@nyu.edu)
Repository on GitHub: reservation-system

****some overly repetitive queries may be omitted for clarity purposes*

use case: /general_show_flights
author: Yanglin Tao
description: Allows all users to search for future flights based on departure airport, arrival airport, departure date, and
arrival date.
query:
```
SELECT flight_number, departure_date, departure_time, departure_airport, arrival_date,
arrival_time, arrival_airport FROM Flight WHERE departure_date = %s AND
departure_airport = %s AND arrival_date = %s AND arrival_airport = %s AND %s >
CURRENT_DATE()
```
explanation:
Select flight_number, departure_date, departure_time, departure_airport, arrival_date, arrival_time, and arrival_airport based on the given departure airport, arrival airport, departure date, and arrival date. The departure date will have to be greater than the current date for the results should be future flights.

use case: /general_check_status
author: Yanglin Tao
description: Allows all users to check the status of a flight based on airline name, flight number, departure date, and arrival date.
query:
```
SELECT airline_name, flight_number, departure_date, arrival_date, flight_status FROM
Flight WHERE airline_name = %s AND flight_number = %s AND departure_date = %s AND
arrival_date = %s
```
explanation:
Select airline_name, flight_number, departure_date, arrival_date and flight_status based on airline name, flight number, departure date, and arrival date.

use case: /customer_login_auth
author: Yanglin Tao
description: Authenticates a customer to login with registered email and password.
query:
```
SELECT * FROM Customer WHERE customer_email = %s and customer_password = %s
```
explanation: If the customer provided email and password matches a value in the database, he/she will be allowed to login.

use case: /staff_login_auth
author: Tianzuo Liu
description: Authenticates a staff to login with registered email and password.
query:
```
SELECT * FROM Airline_Staff WHERE user_name = %s and staff_password = %s
```
explanation: If the staff provided user_name, and the staff_password matches a value in the database, he/she will be allowed to login.

use case: /customer_register_auth
author: Tianzuo Liu
description: Authenticates a customer to register with all the information that is needed.
query:
```
SELECT * FROM Customer WHERE customer_email = %s and customer_password = %s
INSERT INTO Customer VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
```
explanation: Check whether the database has the customer's account first.
query:
```
INSERT INTO Customer VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
```
explanation: If it doesn't have, insert the value into the customer table to let the customer register successfully.

use case: /staff_register_auth
author: Tianzuo Liu
description: Authenticates a staff to register with all the information that is needed.
query:
```
SELECT * FROM Airline_Staff WHERE user_name = %s and staff_password = %s
```
explanation: Check whether the database has the staff's account first.
query:
```
INSERT INTO Airline_Staff VALUES(%s, %s, %s, %s, %s, %s)
```
explanation: If it doesn't have, insert the value into the Airline_staff table to let the staff register successfully.

use case: /customer_view_flights
author: Tianzuo Liu
Description: Customer can see flight information which he/she purchased.
query:
```
SELECT * FROM Ticket NATURAL JOIN Flight WHERE customer_email = %s and departure_date
>= CURDATE()
```
explanation: User table Ticket natural join with table flight, and show all the future flights the customer has.
query:
```
SELECT * FROM Ticket NATURAL JOIN Flight WHERE customer_email = %s and
departure_date >= %s and departure_date <= %s and departure_airport = %s and
arrival_airport = %s
```
explanation: search by start/end date and departure/arrival airport
query:
```
SELECT * FROM Ticket NATURAL JOIN Flight, Airport D, Airport A WHERE
customer_email = %s AND Flight.departure_airport = D.airport_name AND
```

```
Flight.arrival_airport = A.airport_name AND D.airport_city = %s AND A.airport_city =
%s AND departure_date >= %s AND departure_date <= %s
```
explanation: search by start/end date and source/destination city

use case: /customer_search_flights
author: Tianzuo Liu
description: Search for future flights (one way or round trip) based on source city/airport name, destination city/airport name, dates (departure or return).
query:
```
SELECT flight_number, departure_date, departure_time, departure_airport, arrival_date,
arrival_time, arrival_airport, airline_name, base_price FROM Flight WHERE
departure_date = %s AND departure_airport = %s AND arrival_airport = %s AND %s >
CURRENT_DATE()
```
explanation: check whether customers search for a city or airport. If customers search for an airport, use this query, and then check whether customers search for one way or round trip. Then show all the information for the flights if there are flights that customers searched for.
query:
```
SELECT flight_number, departure_date, departure_time, departure_airport, arrival_date,
arrival_time, arrival_airport, airline_name, base_price FROM Flight, Airport D,
Airport A WHERE Flight.departure_airport = D.airport_name AND Flight.arrival_airport =
A.airport_name AND D.airport_city = %s AND A.airport_city = %s AND departure_date = %s
AND %s > CURRENT_DATE()
```
explanation: If a customer searches for a city, use this query, and then check whether the customer searches for one way or round trip. Then show all the information for the flights if there are flights that customers searched for.

use case: /purchase_ticket
author: Tianzuo Liu
description: Customer chooses a flight and purchases a 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.
query:
```
SELECT * FROM Flight WHERE flight_number = %s AND departure_date = %s AND
departure_time = %s AND airline_name = %s AND departure_date > CURRENT_DATE()
```
explanation: Customer choose a flight first
```
SELECT base_price FROM Flight WHERE flight_number = %s AND departure_date = %s AND
departure_time = %s AND airline_name = %s AND departure_date > CURRENT_DATE()
```
explanation: and if this flight is valid, then check the base_price of the flight.
```
SELECT COUNT(ticket_ID) FROM Ticket NATURAL JOIN Airplane NATURAL JOIN Flight WHERE
flight_number = %s AND departure_date = %s AND departure_time = %s AND airline_name =
%s
```
explanation: and then check how many tickets the airline sold.
```
SELECT number_seats FROM Flight NATURAL JOIN Airplane WHERE flight_number = %s AND
departure_date = %s AND departure_time = %s AND airline_name = %s
```
explanation: then check how many seats the airline has. If 60% of the capacity is already booked/reserved for that flight, an extra 20% will be added with the minimum/base price.
```
INSERT INTO Ticket VALUES (%s, %s, %s, %s, %s, %s, %s, CURRENT_DATE(), CURRENT_DATE(),
%s, %s, %s, %s)
```

explanation: Next, insert all the information for the Ticket table and make the customer purchase the ticket successfully if there is nothing wrong.
`INSERT INTO Purchase VALUES(%s, %s)`
explanation: insert all the information for the Purchase table and make the customer purchase the ticket successfully if there is nothing wrong.

use case: /cancel_trip
author: Tianzuo Liu
description: Customer chooses a purchased ticket for a flight that will take place more than 24 hours in the future and cancels 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.
query:
```
SELECT ticket_ID FROM Ticket NATURAL JOIN Flight WHERE customer_email = %s and
departure_date = %s and flight_number = %s and departure_time = %s and airline_name =
%s and departure_date >= CURDATE()
```
explanation: find the ticket that the customer wants to cancel first.
`DELETE FROM Purchase WHERE ticket_ID = %s`
explanation: if there are nothing wrong, then delete the ticket in the Purchase table
`DELETE FROM Ticket WHERE ticket_ID = %s`
explanation: if there are nothing wrong, then delete the ticket in the Ticket table

use case: /customer_rating
author: Yanglin Tao
description: Customers are allowed to rate and comment on a previous flight.
query:
```
SELECT * FROM Customer NATURAL JOIN Ticket WHERE customer_email = %s AND
flight_number = %s AND departure_date = %s AND departure_time = %s AND airline_name =
%s AND %s < CURRENT_DATE()
```
explanation: Check if the input flight is among the customer's previous flights.
query:
```
SELECT * FROM Taken WHERE customer_email = %s AND flight_number = %s AND
departure_date = %s AND departure_time = %s AND airline_name = %s
```
explanation: Check if the customer already commented on this particular flight.
query: `INSERT INTO Taken VALUES(%s, %s, %s, %s, %s, %s, %s)`
explanation: insert the values into the Taken records

use case: /track_spending
author: Yanglin Tao
description: Customers are allowed to see their spending in the past year and monthly spending in the past 6 months. They will also be allowed to view total spending within a date range.
query:
```
SELECT %s AS start_date, %s AS end_date, SUM(sold_price) AS total FROM Ticket
NATURAL JOIN Customer WHERE customer_email = %s AND purchase_date >= %s AND
purchase_date <= %s
```
explanation: If a start date and an end date is specified, the sum of the sold prices will be selected in this customer’s tickets between that date range.

query: `SELECT DATE_SUB(CURRENT_DATE(), INTERVAL 6 MONTH) AS start_date, DATE_SUB(CURRENT_DATE(), INTERVAL 5 MONTH) AS end_date, SUM(sold_price) AS total FROM Ticket NATURAL JOIN Customer WHERE customer_email = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 6 MONTH) AND purchase_date <= DATE_SUB(CURRENT_DATE(), INTERVAL 5 MONTH)`
explanation: Select the sum of sold prices in the customer's tickets of the sixth month in the past 6 months.
query: `SELECT DATE_SUB(CURRENT_DATE(), INTERVAL 5 MONTH) AS start_date, DATE_SUB(CURRENT_DATE(), INTERVAL 4 MONTH) AS end_date, SUM(sold_price) AS total FROM Ticket NATURAL JOIN Customer WHERE customer_email = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 5 MONTH) AND purchase_date <= DATE_SUB(CURRENT_DATE(), INTERVAL 4 MONTH)`
explanation: Select the sum of sold prices in the customer's tickets of the fifth month in the past 6 months.
query: `SELECT DATE_SUB(CURRENT_DATE(), INTERVAL 4 MONTH) AS start_date, DATE_SUB(CURRENT_DATE(), INTERVAL 3 MONTH) AS end_date, SUM(sold_price) AS total FROM Ticket NATURAL JOIN Customer WHERE customer_email = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 4 MONTH) AND purchase_date <= DATE_SUB(CURRENT_DATE(), INTERVAL 3 MONTH)`
explanation: Select the sum of sold prices in the customer's tickets of the fourth month in the past 6 months.
query: `SELECT DATE_SUB(CURRENT_DATE(), INTERVAL 3 MONTH) AS start_date, DATE_SUB(CURRENT_DATE(), INTERVAL 2 MONTH) AS end_date, SUM(sold_price) AS total FROM Ticket NATURAL JOIN Customer WHERE customer_email = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 3 MONTH) AND purchase_date <= DATE_SUB(CURRENT_DATE(), INTERVAL 2 MONTH)`
explanation: Select the sum of sold prices in the customer's tickets of the third month in the past 6 months.
query: `SELECT DATE_SUB(CURRENT_DATE(), INTERVAL 2 MONTH) AS start_date, DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH) AS end_date, SUM(sold_price) AS total FROM Ticket NATURAL JOIN Customer WHERE customer_email = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 2 MONTH) AND purchase_date <= DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH)`
explanation: Select the sum of sold prices in the customer's tickets of the second month in the past 6 months.
query: `SELECT DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH) AS start_date, CURRENT_DATE() AS end_date, SUM(sold_price) AS total FROM Ticket NATURAL JOIN Customer WHERE customer_email = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH) AND purchase_date <= CURRENT_DATE()`
explanation: Select the sum of sold prices in the customer's tickets of the first month in the past 6 months.
query: `'SELECT SUM(sold_price) AS total_last_year FROM Ticket NATURAL JOIN Customer WHERE customer_email = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 YEAR) AND purchase_date <= CURRENT_DATE()`
explanation: Select the sum of sold prices in the customer's tickets in the past year.

use case: /customer_logout
author: Yanglin Tao
description: Customers will be able to log out their accounts and their session will be ended.

use case: /staff_view_flights
author: Yanglin Tao
description: By default the staff will be able to see all flights within the next 30 days. Staff will also be able to specify a range of dates and destination/source airports/cities.
query: `SELECT airline_name FROM Airline_staff WHERE user_name = %s`
explanation: Find out which airline it is from an airline staff's username.
query: `SELECT flight_number, departure_date, departure_time, departure_airport, arrival_date, arrival_time, arrival_airport FROM Flight WHERE departure_date >= %s AND departure_date <= %s AND departure_airport = %s AND arrival_airport = %s AND airline_name = %s`
explanation: Select flights with specified date range and source/destination airports within that airline.
query: `SELECT customer_email FROM Ticket WHERE flight_number = %s AND departure_date = %s AND departure_time = %s AND airline_name = %s`
explanation: Select all customer emails from customers who took the specific flight.
query: `SELECT flight_number, departure_date, departure_time, departure_airport, arrival_date, arrival_time, arrival_airport FROM Flight, Airport D, Airport A WHERE Flight.departure_airport = D.airport_name AND Flight.arrival_airport = A.airport_name AND D.airport_city = %s AND A.airport_city = %s AND departure_date >= %s AND departure_date <= %s AND airline_name = %s`
explanation: Select flights with specified date range and source/destination cities within that airline.
query: `SELECT flight_number, departure_date, departure_time, departure_airport, arrival_date, arrival_time, arrival_airport FROM Flight WHERE departure_date >= CURRENT_DATE() AND departure_date <= DATE_ADD(CURRENT_DATE(), INTERVAL 30 DAY) AND airline_name = %s`
explanation: Select flights within the next 30 days.

use case: /add_flight
author: Yanglin Tao
description: Staff will be able to give all the information and add a new flight to the system.
query: `SELECT airline_name FROM Airline_staff WHERE user_name = %s`
explanation: Find out which airline it is from an airline staff's username.
query: `SELECT * FROM Flight WHERE flight_number = %s AND departure_date = %s AND departure_time = %s AND airline_name = %s`
explanation: Check if the flight he/she wants to add is an existing flight.
query: `SELECT * FROM Airplane WHERE airplane_identification_number = %s AND airline_name = %s`
explanation: Check if the airplane exists in the system.
query: `INSERT INTO Flight (flight_number, departure_airport, departure_date, departure_time, arrival_airport, arrival_date, arrival_time, airplane_identification_number, base_price, airline_name, flight_status) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, NULL)`
explanation: Insert the flight into the system. By default the flight status will be set to NULL.

use case: /change_status
author: Yanglin Tao
description: Airline staff will be allowed to change the status of a flight.
query: `SELECT airline_name FROM Airline_staff WHERE user_name = %s`
explanation: Find out which airline it is from an airline staff's username.
query: `SELECT * FROM Flight WHERE flight_number = %s AND departure_date = %s AND departure_time = %s AND airline_name = %s`
explanation: Check if the flight exists in the system.
query: `UPDATE Flight SET flight_status = %s WHERE flight_number = %s AND departure_date = %s AND departure_time = %s AND airline_name = %s`
explanation: If the flight exists in the system, the flight status will be updated with the given value.

use case: /add_airplane
author: Yanglin Tao
description: Airline staff will be able to add a new airplane to the system by providing all the information.
query: `SELECT airline_name FROM Airline_staff WHERE user_name = %s`
explanation: Find out which airline it is from an airline staff's username.
query: `SELECT * FROM Airplane WHERE airplane_identification_number = %s AND airline_name = %s`
explanation: Check if an airplane already exists in the system.
query: `INSERT INTO Airplane (airplane_identification_number, number_seats, manufacture_company, age, airline_name) VALUES (%s, %s, %s, %s, %s)`
explanation: If the airplane does not exist in the system yet, it will be added to the airplanes.

use case: /add_airport
author: Justin Li
description: Airline staff will be able to add a new airport to the system by providing the name, city, country, and type/
query: `SELECT * FROM Airport WHERE airport_name = %s`
explanation: this query will check if the airport already exists in the system.
query: `INSERT INTO Airport( airport_name, airport_city, airport_cout, airport_type) VALUES (%s, %s, %s, %s)`
explanation: this query will insert the new airport into the database, depending on the input values that airline staff provided.

use case: /view_rating
author: Justin Li
description: Airline staff will be able to view the comments and the ratings from a specific plan, and the average rate will be present as well.
query: `SELECT airline_name FROM Airline_staff WHERE user_name = %s`
explanation: this query will check which airline the airline staff is from.
query: `SELECT rating, comment FROM Taken WHERE flight_number = %s AND departure_date = %s AND departure_time = %s AND airline_name = %s`

explanation: this query will get the comments and the ratings on that specific flight, with input flight_number, departure_date, departure_time, and airline_name
query: `SELECT AVG(rating) as average_rating FROM Taken WHERE flight_number = %s AND departure_date = %s AND departure_time = %s AND airline_name = %s`
explanation: This query will get the average value of the rating of that specific flight.

use case: /frequent_customer
author: Justin Li
description: this will help the airline staff to view the most frequent customer
query: `SELECT airline_name FROM Airline_staff WHERE user_name = %s`
explanation: this query will check which airline the staff is from
query: `Select customer_email, COUNT(ticket_ID) as frequency from NATURAL JOIN Ticket WHERE airline_name = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 YEAR) GROUP BY customer_email ORDER BY frequency DESC LIMIT1`
explanation: this query will help to select the customer with the most ticket_id in the past year. The number of tickets he bought and the customer email will be provided to staff.

use case: /view_customer_flights
author: Justin Li
description: this will help the airline staff to view all the flights that a customer bought.
query: `SELECT airline_name FROM Airline_staff WHERE user_name = %s`
explanation: this query will check which airline the staff is from
query: `SELECT * FROM Customer NATURAL JOIN Ticket NATURAL JOIN Flight WHERE customer_email = %s AND airline_name = %s`
explanation: this query will provide all the flight information that a specific customer has taken.

use case: /view_reports
author: Justin Li
description: this method will enable the staff to view all of the reports that have been published in the past year and month.
query: `SELECT airline_name FROM Airline_staff WHERE user_name = %s`
explanation: this query will check which airline the staff is from
query: `SELECT COUNT(ticket_id) AS num_tickets FROM Ticket WHERE airline_name = %s AND purchase_date >= %s AND purchase_date <= %s`
explanation: this query will execute if the user provided a specific start time and a specific end time. This will provide the number of ticket_id reported between this time period to the airline staff.
query: `SELECT COUNT(ticket_id) AS num_tickets FROM Ticket WHERE airline_name = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 YEAR) AND purchase_date <= CURRENT_DATE()`
explanation: this query will be executed first if the user does not provide a specific start and end time period. This will provide the user with the number of ticket_id reported within a year.
query:`SELECT COUNT(ticket_id) AS num_tickets FROM Ticket WHERE airline_name = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH) AND purchase_date <= CURRENT_DATE()`

explanation: this query will be executed next if the user does not provide a specific start and end time period. This will provide the user with the number of ticket_id reported within a month.

use case: /view_revenue
author: Justin Li
description: This will help the airline staff to view the total revenue that the airline earned by selling tickets over the past year and month.
query: `SELECT airline_name FROM Airline_staff WHERE user_name = %s`
explanation: this query will check which airline the staff is from.
query: `SELECT SUM(sold_price) AS year_revenue FROM Ticket WHERE airline_name = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 YEAR) AND purchase_date <= CURRENT_DATE()`
explanation: this query will provide the sum of the sold_price that has been sold in the past year.
query: `SELECT SUM(sold_price) AS year_revenue FROM Ticket WHERE airline_name = %s AND purchase_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH) AND purchase_date <= CURRENT_DATE()`
explanation: this query will provide the sum of the sold_price that has been sold in the past month.

use case: /staff_logout
author: Justin Li
description: staff will be able to log out their session account.