Final Report - Airline Project

Group Members: Luiza Menezes, Daniel Hunte

A separate PDF 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 codes.

Users whether logged in or not Use Cases:

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 following query returns all the info on flights that are departing within the specified date range, is departing and arriving from and to a specified airport/city and still have seats available.

```
“’SELECT *
FROM flight_expanded
WHERE (departure_airport = %s OR departure_city = %s) AND (arrival_airport = %s OR arrival_city = %s)
AND departure_date = %s AND number_of_passengers < number_of_seats
ORDER BY departure_time ASC”’
```

b. Will be able to see the flights status based on airline name, flight number, arrival/departure date.

The following query returns all the info (including the flight status) on a specific flight based on flight number, arrival and departure date:

```
'''SELECT * FROM flight_expanded WHERE airline = %s AND flight_number = %s AND (arrival_date = %s
OR departure_date = %s)'''
```

2. Register: 3 types of user registrations (Customer, Booking Agent and Airline Staff) option via forms.

The following query was used to check if there was a customer already present in the system with that email. If there was, the user would be redirected to the registration page and an error would be shown:

```
“’SELECT * FROM customer WHERE email = %s’’
```

The following query inserts a new customer into the system:

```
'''INSERT INTO customer
(name,email,password,building_number,street,city,state,phone_number,passport_number,passport_ex
piration,passport_country,date_of_birth)
VALUES(%s,%s,MD5(%s),%s,%s,%s,%s,%s,%s,%s,%s,%s)'''
```

The queries for registering booking agents and airline staff would be very similar.

3. Login: 3 types of user login (Customer, Booking Agent and Airline Staff). Users enters their username (email address will be used as username), x, and password, y, via forms on login page. This

data is sent as POST parameters to the login-authentication component, which checks whether there is a tuple in the corresponding user's table with username=x and the password = md5(y).

The following query checks to see if the email and password the user entered is present in the database. If it is the user (customer) is then redirected to the customer home page.

'SELECT * FROM customer WHERE email = %s and password = MD5(%s)'

The queries for registering booking agents and airline staff would be very similar.

Customer use cases:

5. 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).

The following query returns all the info on flights that are departing within the specified date range, is departing and arriving from and to a specified airport/city and still have seats available:

'''SELECT * FROM flight_expanded WHERE (departure_airport = %s OR departure_city = %s) AND (arrival_airport = %s OR arrival_city = %s) AND departure_date = %s AND number_of_passengers < number_of_seats ORDER BY departure_time ASC'''

6. 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 following query inserts the ticket info in the ticket table when a customer purchases a ticket

'''INSERT INTO ticket(flight_number,airline,customer_email,sold_price,card_type,card_number,name_on_card,expiration_date,purchase_date,purchase_time)
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,CURDATE(),CURTIME())'''
This query would be replicated for purchasing a round trip ticket (two tickets)

The following query inserts the ticket ID and customer email into the cus_purchase table. Since the ticket ID was autoincremented from the previous insert, the LAST_INSERT_ID() function was used:

'''INSERT INTO cus_purchase(ticket_id,customer_email)
VALUES(LAST_INSERT_ID(),%s)'''

6. 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) for the airline they logged in.

The following query inserts the rating the customer enters into the ratings table. If the customer has already rated that flight in the past it will update the rate and the comment with the most recent entry:

```
'''INSERT INTO rating
(customer_email,flight_number,airline,rate,comment)
   VALUES(%s,%s,%s,%s,%s)
   ON DUPLICATE KEY UPDATE rate=%s, comment=%s'''
```

7.Track My Spending: Default view will be total amount of money spent in the past year and a bar chart 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 showing month wise money spent within that range.

The following query gives the total amount of money a customer spent between a specified time period. By default this time period is the past year:

```
'''SELECT IFNULL(SUM(sold_price),0) AS total
  FROM ticket
  WHERE customer_email = %s AND purchase_date >= %s AND purchase_date <= %s'''
```

Booking Agent Queries:

4. View My flights: Provide various ways for the booking agents to see flights information for which he/she purchased on behalf of customers. 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 and/or city name etc to show all the flights for which he/she purchased tickets.

The following query gives all the flight info based on the tickets the booking agent has purchased on behalf of cusomters, based on the specified departing and arrival airport/city and departure date.

```
'''SELECT * FROM agent_purchase JOIN ticket ON (agent_purchase.ticket_ID = ticket.ID) JOIN
flight_expanded ON (ticket.flight_number = flight_expanded.flight_number) JOIN customer ON
(ticket.customer_email = customer.email)
WHERE agent_email = %s AND departure_date >= %s AND departure_date <= %s
ORDER BY departure_date ASC'''
```

5. 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 arrival).

The following query gives all the info on flight based on specified city/airport and the departure date (by default, this shows future flights).

```
'''SELECT * FROM flight_expanded WHERE (departure_airport = %s OR departure_city = %s) AND
(arrival_airport = %s OR arrival_city = %s) AND departure_date = %s AND number_of_passengers <
number_of_seats
 ORDER BY departure_time ASC'''
```

6. Purchase tickets: Booking agent chooses a flight and purchases tickets for other customers giving customer information and payment information, providing all the needed data, via forms. You may find it easier to implement this along with a use case to search for flights.

This query inserts all the ticket info after booking agent
purchases a ticket on behalf of the customer.

'''INSERT INTO ticket
(flight_number,airline,customer_email,sold_price,card_type,card_number,name_on_card,expiration_date,purchase_date,purchase_time,booking_agent_ID)
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,CURDATE(),CURTIME(),(SELECT booking_agent_ID FROM
booking_agent WHERE email = %s))'''

7. View my commission: Default view will be total amount of commission received in the past 30 days and the average commission he/she received per ticket booked in the past 30 days and total number of tickets sold by him in the past 30 days. He/she will also have option to specify a range of dates to view total amount of commission received and total numbers of tickets sold.

This query sums the price of the tickets that the booking agent has sold, and then finds the average price of the tickets they have sold between within a thirty-day price period.

'''SELECT IFNULL(0.1 * SUM(sold_price), 0) as total, IFNULL(0.1 * AVG(sold_price), 0) AS average, COUNT(ticket_ID) AS num_tickets FROM agent_purchase JOIN ticket ON (agent_purchase.ticket_ID = ticket.ID) WHERE agent_email = %s AND purchase_date >= %s AND purchase_date <= %s'''

8. View Top Customers: Top 5 customers based on number of tickets bought from the booking agent in the past 6 months and top 5 customers based on amount of commission received in the last year. Show a bar chart showing each of these 5 customers in x-axis and number of tickets bought in y-axis. Show another bar chart showing each of these 5 customers in x-axis and amount commission received in y- axis.

The following query selects the top five customers that booked a flight with the help of a booking agent. The top five customers have been determined by how many tickets they have purchased with the help of a booking agent within the past six months.

'''SELECT customer.name AS name, customer.email AS email,COUNT(ticket_ID) AS num_tickets FROM agent_purchase JOIN ticket ON (ticket_ID = ID) JOIN customer ON (customer.email = agent_purchase.customer_email) WHERE agent_email = %s GROUP BY customer.email ORDER BY num_tickets DESC LIMIT 5'''

The following query selects the top five customers that booked a flight with the help of a booking agent and helped the booking agent receive the most commission as possible. These customers averaged the most expensive flight purchases and total sold prices.

'''SELECT customer.name AS name, customer.email AS email, IFNULL((0.1 * SUM(sold_price)), 0) AS commission FROM agent_purchase JOIN ticket ON (ticket_ID = ID) JOIN customer ON (customer.email = agent_purchase.customer_email) WHERE agent_email = %s GROUP BY customer.email ORDER BY commission DESC LIMIT 5'''

<u>**Airline Staff Queries:**</u>

4. View flights: Defaults will be showing all the future flights operated by the airline he/she works for the next 30 days. He/she will be able to see all the current/future/past flights operated by the airline

**he/she works for based range of dates,
source/destination airports/city etc. He/she will be able to see all the customers of a particular flight.**

The following query displays all flights that airline will be conducting. By default, these values are set to a thirty day interval between the current date and 30 days from now. Otherwise, the airline staff will be able to put the range of dates from where they want to see flights by their airline.

'''SELECT * FROM flight_expanded WHERE departure_date BETWEEN %s AND %s ORDER BY departure_date DESC'''

The following query allows airline staff to displays all customer's names and emails that will be on a specific flight.

'''SELECT customer.name, customer.email FROM ticket JOIN customer ON (ticket.customer_email = customer.email) WHERE airline = %s AND flight_number = %s'''

5. Create new flights: He or she creates a new flight, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action. Defaults will be showing all the future flights operated by the airline he/she works for the next 30 days.

The following query inserts a new flight into our database using forms.

'''INSERT INTO flight(flight_number,airline,departure_airport,departure_date,departure_time,
arrival_airport,arrival_date,arrival_time,base_price,status,airplane_id)
VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)'''

6. Change Status of flights: He or she changes a flight status (from on-time to delayed or vice versa) via forms.

The following query updates our database to reflect the changes in flight status, arrival date, departure dates, and times.

'''UPDATE flight SET status = %s WHERE flight_number = %s AND airline = %s'''

7. Add airplane in the system: He or she adds a new airplane, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action. In the confirmation page, she/he will be able to see all the airplanes owned by the airline he/she works for.

The following query inserts a new airplane into our database using request forms and information given by the airline staff.

'''INSERT INTO airplane (id,airline, number_of_seats) VALUES(%s, %s, %s)'''

8. Add new airport in the system: He or she adds a new airport, providing all the needed data, via forms. The application should prevent unauthorized users from doing this action.

The following query inserts a new airport into our database using request forms and information given by the airline staff.

'''INSERT INTO airport (city, name) VALUES (%s, %s)'''

9. View flight ratings: Airline Staff will be able to see each flight's average ratings and all the comments and ratings of that flight given by the customers.

The following query collects all ratings from a particular flight that belongs to the airline the airline staff worker works for. This is displayed as a button next to the past flights.

'''SELECT rate, comment FROM rating WHERE airline = %s AND flight_number = %s'''

10. View all the booking agents: Top 5 booking agents based on number of tickets sales for the past month and past year. Top 5 booking agents based on the amount of commission received for the last year.

The following finds the top five booking agents that have had the most ticket sales during the past year.

'''SELECT booking_agent.email AS email, booking_agent.booking_agent_ID as booking_agent_ID, COUNT(ticket.ID) AS num_tickets FROM booking_agent JOIN agent_purchase ON (booking_agent.email = agent_purchase.agent_email) JOIN ticket ON (agent_purchase.ticket_ID = ticket.ID) WHERE ticket.airline = %s GROUP BY booking_agent.email ORDER BY num_tickets DESC LIMIT 5'''

The following finds the top five booking agents with the highest commission over the last year.

'''SELECT booking_agent.email AS email, booking_agent.booking_agent_ID as booking_agent_ID, SUM(0.1 * sold_price) AS commission FROM booking_agent JOIN agent_purchase ON (booking_agent.email = agent_purchase.agent_email) JOIN ticket ON (agent_purchase.ticket_ID = ticket.ID) WHERE ticket.airline = %s GROUP BY booking_agent.email ORDER BY commission DESC LIMIT 5'''

11. View frequent customers: Airline Staff will also be able to see the most frequent customer within the last year. In addition, Airline Staff will be able to see a list of all flights a particular Customer has taken only on that particular airline.

The following query runs through tickets to select which customers have flown most frequently throughout the last year.

'''SELECT name, email, MAX(num_tickets) FROM (SELECT customer.name AS name, customer_email AS email, COUNT(ticket.ID) AS num_tickets FROM ticket JOIN customer ON (customer.email = ticket.customer_email) WHERE (airline = %s) AND purchase_date BETWEEN (CURDATE() - INTERVAL 1 YEAR) AND CURDATE() GROUP BY ticket.customer_email ORDER BY num_tickets DESC) AS x'''

12. View reports: Total amounts of ticket sold based on range of dates/last year/last month etc. Month wise tickets sold in a bar chart.

The following query counts the amount of tickets sold by the booking agent based off a range of dates.

'''SELECT IFNULL(COUNT(ticket.ID), 0) AS total FROM ticket WHERE airline = %s AND purchase_date BETWEEN %s AND %s'''

The following query takes the previous ticket data and separates the amounts sold into month and date ranges.

'''SELECT YEAR(purchase_date) AS year, MONTH(purchase_date) AS month, SUM(sold_price) AS total FROM ticket WHERE airline = %s AND purchase_date >= %s AND purchase_date <= %s GROUP BY YEAR(purchase_date), MONTH(purchase_date) ORDER BY YEAR(purchase_date), MONTH(purchase_date)'''

13. Comparison of Revenue earned: Draw a pie chart for showing total amount of revenue earned from direct sales (when customer bought tickets without using a booking agent) and total amount of revenue earned from indirect sales (when customer bought tickets using booking agents) in the last month and last year.

The following query sums the ticket sold by booking agents during the past year.

'''SELECT SUM(ticket.sold_price) AS total FROM cus_purchase JOIN ticket ON (cus_purchase.ticket_ID = ticket.ID) WHERE ticket.airline = %s'''

The following query sums the ticket sold by booking agents during the past year.

'''SELECT SUM(ticket.sold_price) AS total FROM agent_purchase JOIN ticket ON (agent_purchase.ticket_ID = ticket.ID) WHERE ticket.airline = %s'''

14. View Top destinations: Find the top 3 most popular destinations for last 3 months and last year (based on tickets already sold).

The following query selects the top travel destinations based off information from the last year based off tickets sales frequency.

'''SELECT flight_expanded.arrival_city AS city, COUNT(ticket.ID) AS num_tickets FROM ticket JOIN flight_expanded ON ((ticket.airline,ticket.flight_number) = (flight_expanded.airline,flight_expanded.flight_number)) WHERE ticket.airline = %s AND ticket.purchase_date BETWEEN (CURDATE() - INTERVAL 1 YEAR) AND CURDATE() GROUP BY city LIMIT 3'''

The following query consists of two with clauses, the first joins flight and airport in order to get the city [flight_city] and the second joins the flight to the airplane to get the number of seats [flight_size]. Finally, these two with clauses are joined to produce flight_expanded.

'''CREATE VIEW IF NOT EXISTS flight_expanded AS WITH flight_city AS (SELECT flight_number,airline,airplane_id,departure_date,departure_time,arrival_date,arrival_time,departure_airport,departure_city,arrival_airport,arrival_city,status,base_price FROM flight JOIN (SELECT name AS departure_airport_name, city AS departure_city FROM airport) as s JOIN (SELECT name AS arrival_airport_name, city AS arrival_city FROM airport) as t WHERE departure_airport = s.departure_airport_name AND arrival_airport = t.arrival_airport_name), flight_size AS (SELECT flight_number as flight_num,airline AS airl,IFNULL(COUNT(ticket.ID),0) as number_of_passengers FROM flight NATURAL LEFT JOIN ticket GROUP BY flight_number,airline) SELECT flight_number,airline,airplane_id,departure_date,departure_time,arrival_date,arrival_time,departure_airport,departure_city,arrival_airport,arrival_city,status,base_price,IF(number_of_passengers <

```
0.7*number_of_seats, base_price, 1.2*base_price) AS
sale_price,number_of_seats,number_of_passengers FROM (SELECT * FROM flight_city AS s JOIN
(SELECT ID,airline AS al,number_of_seats FROM airplane) AS t ON (s.airplane_ID = t.ID AND s.airline =
t.al)) AS u JOIN flight_size ON (flight_size.flight_num = u.flight_number AND flight_size.airl = u.airline)'''
```