Airline Reservation System: Queries

Zhirui Yao, Jiacheng Qiu

1. General Usecase:

(1). Search for upcoming flights: since the departure_time in the database includes date and time, we need to use the DATE() function to match the user input

```
-> 'SELECT * FROM flight WHERE departure_airport = %s \
     AND arrival_airport = %s AND DATE(departure_time) = %s\
     AND status = "upcoming"'
```

(2). Check flight status:

```
-> 'SELECT * FROM flight WHERE flight_num = %s \
     AND airline_name = %s  AND DATE(arrival_time) = %s AND
   DATE(departure_time) = %s'
```

(3). Register:

-> Check that the customer has not registered:

```
'SELECT * FROM customer WHERE email = %s'
```

If not, insert this new customer information into the database

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

-> Check that the agent has not registered:

```
'SELECT * FROM booking_agent WHERE email = %s'
```

If not, insert this new agent information into the database

```
'INSERT INTO booking_agent VALUES(%s,%s,%s)'
```

-> Check that the staff has not registered:

'SELECT * FROM airline_staff WHERE username = %s'

If he has not registered yet, check that that the airline the staff inputs exists (otherwise the foreign key constraint is violated):

'SELECT * FROM airline WHERE airline_name = %s'

If this airline exists, insert this new staff information into the database

'INSERT INTO airline_staff VALUES(%s,%s,%s,%s,%s,%s)'

(4). <u>Log in:</u>

-> 'SELECT * FROM booking_agent WHERE email = %s and password = %s'

-> 'SELECT * FROM customer WHERE email = %s and password = %s'

-> 'SELECT * FROM airline_staff WHERE username = %s and password = %s'

(5). <u>Customer can edit their profile information:</u>

```
'UPDATE customer SET customer.building_number = %s,\
        customer.street= %s, customer.city= %s, customer.state= %s,\
            customer.phone_number= %s, customer.passport_number= %s,\
                customer.passport_expiration= %s, customer.passport_country= %s,\
                    customer.date_of_birth=%s WHERE customer.email = %s'
```

2. **Customer Usecase:**

For all the use cases below, the server will first check whether the person carrying out the operations is a customer. Otherwise, his operation will not be permitted.

(1). <u>View my flights:</u>

-> View all the upcoming flights that the customer has purchased:

'SELECT * FROM customer, purchases, ticket NATURAL JOIN flight\

```
        WHERE customer.email = %s AND customer.email = purchases.customer_email\
        AND purchases.ticket_id = ticket.ticket_id AND status = "upcoming"'
```

If it returns nothing, it means the customer hasn't purchased any upcoming flights.

(2). <u>Search for flights:</u>

```
'SELECT * FROM flight WHERE departure_airport = %s \
            AND arrival_airport = %s AND DATE(departure_time) = %s\
            AND status = "upcoming"'
```

If it returns nothing, it means there are no such upcoming flights.

(3). <u>Purchase tickets</u>

-> Check if the input airline and flight does exist and status is upcoming

```
'SELECT * FROM flight WHERE airline_name = %s AND\
                flight_num = %s AND status="upcoming"'
```

-> Check if the tickets have already sold out yet: by counting the number of tickets sold and compare it with the capacity of the airplane executing this flight

```
"SELECT flight_num, COUNT(ticket_ID)\
                FROM ticket as T\
                WHERE airline_name = %s \
                Group BY airline_name, flight_num\
                HAVING flight_num = %s \
                AND COUNT(ticket_id) < (SELECT DISTINCT seats\
                            FROM flight NATURAL JOIN airplane\
                            WHERE flight.flight_num = T.flight_num\
                    AND flight.airline_name = T.airline_name)"
```

-> two scenarios when this query returns empty: 1. all tickets are sold

2. there is no such flight in ticket table yet

To check if it is scenario 2:

```
"SELECT *\
        FROM ticket\
        WHERE airline_name = %s \
        AND flight_num = %s "
```

-> The tickets have not sold out, make sure the randomly-generated ticket number does not already exist

```
'SELECT * FROM ticket WHERE ticket_id = %s AND airline_name = %s AND flight_num
    = %s'
```

-> If the above query returns nothing, insert the purchase to ticket

```
'INSERT INTO ticket VALUES(%s,%s,%s)'
```

-> Insert the purchase information to purchases

```
'INSERT INTO purchases(ticket_id, customer_email, purchase_date) VALUES(%s,%s,%s)'
```

(3). Track my spending:

-> View total spending:

```
'SELECT customer_email, SUM(price) AS total_spending\
    FROM (purchases NATURAL JOIN ticket) NATURAL JOIN flight WHERE
    customer_email = %s\
    AND purchase_date BETWEEN %s AND %s GROUP BY customer_email;'
```

If it returns nothing, it means the customer didn't purchase any flights within the time period.

-> View monthly spending:

```
'select customer_email, year(purchase_date) AS year,\
    month(purchase_date) AS month, SUM(price) AS monthly_spending\
    from (purchases NATURAL JOIN ticket) NATURAL JOIN flight\
    WHERE customer_email = %s AND purchase_date BETWEEN %s AND %s\
    group by year(purchase_date), month(purchase_date), customer_email;'
```

(4). <u>Logout:</u> Our customer's session is destroyed and redirects our user to the login page.

3. **Agent Usecase:**

(1). <u>View my customers:</u>

```
'SELECT DISTINCT name, customer_email FROM (booking_agent \
    NATURAL JOIN purchases), customer WHERE customer_email = customer.email\
    AND booking_agent.email = %s'
```

If it returns nothing, it means there is no current booking requests from its customers

(2). <u>View my customers' flights:</u>

-> A list of all my customers' flight purchases

```
'SELECT *, booking_agent.email AS agent_email,\
    customer.email AS cus_email FROM booking_agent, customer, purchases, ticket\
    NATURAL JOIN flight WHERE booking_agent.email = %s AND\
    customer.email = purchases.customer_email AND\
    booking_agent.booking_agent_id = purchases.booking_agent_id AND\
    purchases.ticket_id = ticket.ticket_id'
```

(3). Search for flights:

```
'SELECT * FROM flight WHERE departure_airport = %s \
        AND arrival_airport = %s AND DATE(departure_time) = %s\
        AND status = "upcoming"'
```

If it returns nothing, it means there are no such upcoming flights.

(4). Purchase tickets

-> Check if the input airline and flight does exist and status is upcoming

```
'SELECT * FROM flight WHERE airline_name = %s AND\
            flight_num = %s AND status="upcoming"'
```

-> Check if the tickets have already sold out yet: same implementation as customer use cases

```
"SELECT flight_num, COUNT(ticket_ID)\
             FROM ticket as T\
             WHERE airline_name = %s \
             Group BY airline_name, flight_num\
             HAVING flight_num = %s \
             AND COUNT(ticket_id) < (SELECT DISTINCT seats\
                         FROM flight NATURAL JOIN airplane\
                         WHERE flight.flight_num = T.flight_num\
                         AND flight.airline_name = T.airline_name)"
```

-> two scenarios when this query returns empty: 1. all tickets are sold

2. there is no such flight in ticket table yet

To check if it is scenario 2:

```
                "SELECT *\
                        FROM ticket\
                        WHERE airline_name = %s \
                        AND flight_num = %s "
```

-> The tickets have not sold out, make sure the randomly-generated ticket number does not already exist

```
'SELECT * FROM ticket WHERE ticket_id = %s AND airline_name = %s AND flight_num
    = %s'
```

-> Insert the purchase to ticket

```
'INSERT INTO ticket VALUES(%s,%s,%s)'
```

-> Insert the purchase to purchase

```
query3 = 'INSERT INTO purchases VALUES(%s,%s,%s,%s)'
```

(5). View my commission:

-> We assume the commission rate is 5% of the ticket price

```
'SELECT booking_agent_id, SUM(price)*0.05 AS total_com, \
    SUM(price)*0.05/COUNT(purchases.ticket_id) AS avg_com,
    COUNT(purchases.ticket_id) \
    AS ticket_sales FROM (purchases NATURAL JOIN ticket) NATURAL JOIN flight \
    WHERE booking_agent_id = %s AND purchases.purchase_date BETWEEN %s AND
    %s \
    GROUP BY booking_agent_id'
```

(6). My top customers

-> Top 5 customers based on number of tickets bought from the booking agent in the past 6 months

```
'SELECT customer.name, COUNT(purchases.ticket_id) AS num_purchased \
    FROM purchases,customer WHERE purchases.booking_agent_id = %s AND \
    purchases.customer_email = customer.email AND purchases.purchase_date BETWEEN
    %s\
    AND %s GROUP BY \
    customer.name ORDER BY num_purchased DESC LIMIT 5'
```

-> Top 5 customers based on amount of commission received in the last year

```
'SELECT customer.name, SUM(flight.price)*0.05 AS commission\
    FROM (purchases NATURAL JOIN ticket) NATURAL JOIN flight, customer \
    WHERE purchases.booking_agent_id = %s AND customer.email =
    purchases.customer_email \
    AND purchases.purchase_date BETWEEN %s AND %s\
    GROUP BY purchases.customer_email ORDER BY commission DESC LIMIT 5'
```

4. **Staff Usecase:**

For all the use cases below, the server will first check whether the person carrying out the operations is an airline staff. Otherwise, his operation will not be permitted.

(1). View my flights:

-> View flights in the upcoming 30 days:

```
        "SELECT * FROM flight WHERE airline_name = %s AND\
                DATE(departure_time) >= %s AND DATE(departure_time) <= %s\
                    AND status = 'upcoming'''
```

-> Search for the information of a flight based on flight number and arrival/departure date:

```
    'SELECT * FROM flight WHERE flight_num = %s \
        AND DATE(arrival_time) = %s AND DATE(departure_time) = %s'
```

-> Search for all flights that take off during a certain time period based on their arrival/departure airport.

```
    'SELECT * FROM flight WHERE airline_name = %s AND\
    DATE(departure_time) >= %s AND DATE(departure_time) <= %s AND\
    departure_airport = %s AND arrival_airport = %s'
```

-> Check the customer list of a flight:

First, check that this flight exists:

```
    "SELECT * FROM flight\
        WHERE airline_name = %s AND flight_num = %s"
```

If it exists, return the customer information list:

```
    "SELECT DISTINCT customer_email, purchase_date\
            FROM flight NATURAL JOIN ticket, purchases\
            WHERE airline_name = %s AND flight_num = %s\
            AND purchases.ticket_id = ticket.ticket_id"
```

(2). Add new flights:

First, check that this flight does not already exist:

```
    'SELECT * FROM flight WHERE airline_name = %s AND flight_num = %s'
```

If not, insert the information of the new flight into database:

```
'INSERT INTO flight VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s)'
```

(3). Change status of flight:

First, insert the updated information of the flight into database:

```
'UPDATE flight SET status = %s WHERE airline_name = %s \
        AND flight_num = %s AND DATE(arrival_time) = %s AND
DATE(departure_time) = %s'
```

Then, use the below query to check if this updated flight exists in the system:

```
'SELECT * FROM flight WHERE airline_name = %s AND flight_num = %s \
                AND DATE(arrival_time) = %s AND DATE(departure_time) = %s'
```

If it returns nothing, then it means that this flight does not exist, and thus no update is made.

(4). Add airplane:

First, check that the airplane does not already exist:

```
'SELECT * FROM airplane WHERE airline_name = %s AND\
                airplane_id = %s AND seats = %s'
```

If not, insert:

```
'INSERT INTO airplane VALUES(%s,%s,%s)'
```

Lastly, display all airplanes of this company in the confirmation page:

```
'SELECT * FROM airplane WHERE airline_name = %s'
```

(5). Add airport:

First, check whether this airport already exists:

```
'SELECT * FROM airport WHERE airport_name = %s'
```

If not, insert this new airport information to the database:

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

(6). <u>View top booking agent:</u>

-> Query top 5 agents based on sales within a time period:

```
        "SELECT booking_agent.email AS email, COUNT(distinct ticket_id) AS tickets \
        FROM purchases NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN
booking_agent\
  WHERE purchase_date >= %s AND purchase_date <= %s AND airline_name = %s\
  GROUP BY booking_agent.email ORDER BY tickets DESC LIMIT 5"
```

-> Query top 5 agents based on commission fee within a time period:

```
        "SELECT booking_agent.email AS email, SUM(price)*0.05 AS commission\
          FROM purchases NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN
booking_agent WHERE purchase_date >= %s AND purchase_date <= %s\
  AND airline_name = %s\
  GROUP BY booking_agent.email ORDER BY commission DESC LIMIT 5"
```

(7). <u>View top customers:</u>

-> View top 1 customer based on the number of tickets purchased:

```
        "SELECT customer_email AS email, name, COUNT(ticket_ID) as ticket\
  FROM purchases NATURAL JOIN ticket NATURAL JOIN customer\
  WHERE purchase_date >= %s AND purchase_date <= %s AND airline_name=%s\
  GROUP BY customer_email, name ORDER BY COUNT(ticket_ID) DESC LIMIT 1"
```

-> View all flights he/she has purchased:

```
        "SELECT * FROM purchases NATURAL JOIN ticket NATURAL JOIN flight\
            WHERE ticket.airline_name = %s\
            AND purchases.customer_email = ( SELECT customer_email AS email\
            FROM purchases NATURAL JOIN ticket\
           WHERE purchase_date >= %s AND purchase_date <= %s  AND airline_name= %s\
           GROUP BY customer_email ORDER BY COUNT(ticket_ID) DESC LIMIT 1)"
```

(8) Top destinations:

-> View top 3 destinations during different time periods: need to find the arrival airport of flights and the corresponding arrival cities. Many arrival airports may be located in the same city, and we need to count the number of flights that arrive in the same city, not the same airport.

```
"SELECT airport_city, COUNT(customer_email) AS ticket_sold\
   FROM flight, purchases NATURAL JOIN ticket NATURAL JOIN airport\
   WHERE ticket.airline_name = %s AND flight.arrival_airport = airport.airport_name\
   AND flight.airline_name = ticket.airline_name AND ticket.flight_num  = flight.flight_num\
   AND purchase_date >= %s AND purchase_date <= %s \
   GROUP BY airport_city ORDER BY COUNT(airport_city)  DESC LIMIT 3"
```

-> View top 3 destinations in each month of a calendar year

```
subquery_select = "(SELECT year(departure_time) AS year, month(departure_time) AS month,
airport_city, COUNT(customer_email) AS ticket_sold\
   FROM flight, purchases NATURAL JOIN ticket NATURAL JOIN airport\
   WHERE ticket.airline_name = %s AND flight.arrival_airport = airport.airport_name\
   AND flight.airline_name = ticket.airline_name AND ticket.flight_num  = flight.flight_num
AND MONTH(departure_time) = %s\
   GROUP BY airport_city, year, month ORDER BY COUNT(airport_city) DESC LIMIT 3)"
   query_select = ''
 for i in range(11):
      query_select += subquery_select+'union all'
```

(9). Sales Report:

-> Get a list of all the sales

```
        'select DATE_FORMAT(concat(concat(year(purchase_date), "-" ,
        month(purchase_date)),"-01"), "%%Y-%%m") AS yearmonth,\
                year(purchase_date) AS year , month(purchase_date) AS month,\
                airline_name, COUNT(ticket.ticket_id) AS ticketsales\
                from ((purchases NATURAL JOIN ticket) NATURAL JOIN flight) NATURAL
        JOIN airline\
                WHERE purchase_date BETWEEN %s AND %s AND airline_name = %s\
                GROUP BY airline_name, yearmonth, year, month;'
```

-> To fulfill the empty months:

```
      idx = pd.date_range(start_date,end_date_1,freq='M').to_period('m')
      years = np.arange(int(start_date[0:4]),int(end_date[0:4])+1,1)
      df = pd.DataFrame(data)
      df["yearmonth"] = pd.to_datetime(df["yearmonth"])
      df = df.set_index('yearmonth')
      df['year']= df['year'].astype(int)
      df.index = df.index.to_period('m')
```

```
o_d = df.loc[df['year'].isin(years)]

o_d = o_d.reindex(idx,fill_value=np.nan)

o_d['ticketsales'] = o_d['ticketsales'].fillna(0)

o_d = o_d.reset_index()

sales_result =o_d.to_dict('records')
```