Public View:

1. Get airport and city
   a. Input is airport

```
DELIMITER //
CREATE PROCEDURE GetAirportWithCity()
BEGIN
   SELECT airport_city, airport_name FROM `airport`;
END //
DELIMITER ;
```

   b. Input is city

```
DELIMITER //
CREATE PROCEDURE GetUniqueAirportCity()
BEGIN
   SELECT DISTINCT airport_city FROM `airport`;
END //
DELIMITER ;
```

2. Get flights by location

```
SELECT airline_name, flight_num, departure_airport, SRC.airport_city, departure_time,
arrival_airport, DST.airport_city, arrival_time, price, status, airplane_id
FROM flight AS F JOIN airport AS SRC ON (F.departure_airport = SRC.airport_name) JOIN
airport AS DST ON (F.arrival_airport = DST.airport_name)
WHERE DATE(F.departure_time) = %s AND F.status = 'Upcoming' AND SRC.airport_city = %s
AND DST.airport_city= %s AND (F.departure_airport = %s OR %s = '') AND
(F.arrival_airport = %s OR %s = '')
ORDER BY departure_time
```

3. Get flight status: Today or Specify a range of date

```
SELECT airline_name, flight_num, status, airport_city,departure_time
FROM flight JOIN airport ON (flight.arrival_airport =
airport.airport_name)
WHERE (flight_num = %s OR %s = '') AND (DATE(departure_time) = %s OR
%s = '') AND (DATE(arrival_time) = %s OR %s = '')
```

4. Get upcoming flights

```
cursor = conn.cursor(prepared=True)
   email = email.replace("\'", "\'\'")
   query = """SELECT airline_name, flight_num, departure_airport, SRC.airport_city,
departure_time,arrival_airport, DST.airport_city, arrival_time, price, status,
airplane_id
              FROM flight AS F JOIN airport AS SRC ON (F.departure_airport =
SRC.airport_name)JOIN airport AS DST ON (F.arrival_airport = DST.airport_name)
              WHERE status = 'Upcoming' AND """
   if identity == "airline_staff":
       query += """DATE(departure_time) <= \'%s\' AND """ % next_thirty_days
       query += """ airline_name IN (
                       SELECT airline_name
                       FROM airline_staff
                       WHERE username = %s
       )"""
   elif identity == "customer":
       query += """ flight_num IN (
                       SELECT flight_num
                       FROM purchases NATURAL JOIN ticket
                       WHERE customer_email = %s
       )"""
   else:
       query += """ flight_num IN (
                       SELECT flight_num
                       FROM purchases NATURAL JOIN ticket NATURAL JOIN booking_agent
                       WHERE email =  %s
       )"""

   # print("get upcoming flights query is:\n", query % email)
   cursor.execute(query, (email,))
```

5. Get time flight

```
cursor = conn.cursor(prepared=True)
   query = """SELECT airline_name, flight_num, departure_airport, SRC.airport_city,
departure_time, arrival_airport, DST.airport_city, arrival_time, price, status,
airplane_id
              FROM flight AS F JOIN airport AS SRC ON (F.departure_airport =
SRC.airport_name)JOIN airport AS DST ON (F.arrival_airport = DST.airport_name)
              WHERE
           """

   if identity == "airline_staff":
       query += """ airline_name IN (
                       SELECT airline_name
                       FROM airline_staff
                       WHERE username = %s
       )"""
   elif identity == "customer":
       query += """ flight_num IN (
                       SELECT flight_num
                       FROM purchases NATURAL JOIN ticket
                       WHERE customer_email = %s
       )"""
   else:
       query += """ flight_num IN (
                       SELECT flight_num
                       FROM purchases NATURAL JOIN ticket NATURAL JOIN booking_agent
                       WHERE email = %s
       )"""

   query += """ AND (DATE(departure_time) >= %s OR %s = '') AND (DATE(departure_time)
<= %s OR %s = '')
                AND (SRC.airport_city = %s OR %s = '') AND (DST.airport_city = %s OR
%s = '')
                AND (departure_airport = %s OR %s = '') AND (arrival_airport = %s OR
%s = '')
                ORDER BY departure_time
            """
   cursor.execute(query, (email, start_date, start_date, end_date, end_date, src_city,
src_city, dst_city, dst_city, src_airport, src_airport, dst_airport, dst_airport))
   data = cursor.fetchall()
   cursor.close()
```

6. Get specific flights

```
cursor = conn.cursor(prepared=True)
   query = """SELECT airline_name, flight_num, departure_airport, SRC.airport_city,
departure_time,
                     arrival_airport, DST.airport_city, arrival_time, price, status,
airplane_id
              FROM flight AS F JOIN airport AS SRC ON (F.departure_airport =
SRC.airport_name)
                               JOIN airport AS DST ON (F.arrival_airport =
DST.airport_name)
              WHERE airline_name = %s AND flight_num = %s
           """
   cursor.execute(query, (airline_name, flight_num))
```

7. Purchase tickets

```
cursor = conn.cursor(prepared=True)
   query = """SELECT *
                  FROM flight
                  WHERE airline_name = %s AND flight_num = %s AND status = 'Upcoming'
AND departure_time > %s"""
   cursor.execute(query, (airline_name, flight_num,
datetime.today().strftime("%Y-%m-%d")))
   check_flight = cursor.fetchall()
   if not check_flight:
       cursor.close()
       return False, "Don't cheat! Can't buy this flight!"

   query = """SELECT COUNT(ticket_id)
              FROM ticket
              WHERE flight_num = %s"""
   # print(query)
   cursor.execute(query, (flight_num,))
   ticket_num = cursor.fetchall()
   ticket_num = ticket_num[0][0] if ticket_num else 0
   query = """SELECT seats
              FROM airplane NATURAL JOIN flight
              WHERE flight_num = %s"""
   # print(query)
   cursor.execute(query, (flight_num,))
   seat_num = cursor.fetchall()
   seat_num = seat_num[0][0] if seat_num else 0
   # print(ticket_num, seat_num)
   if ticket_num >= seat_num:
       cursor.close()
       return False, "No ticket! This flight is full!"

   query = """SELECT * FROM customer WHERE email = %s"""
   cursor.execute(query, (customer_email,))
   data = cursor.fetchall()
   if not data:
       cursor.close()
       return False, "Customer does not exists!"
```

```
ticket_id = flight_num[:2] + datetime.now().strftime("%Y%m%d%H%M%S")
today = datetime.today().strftime("%Y-%m-%d")
query = """INSERT INTO ticket
           VALUES (%s, %s, %s)"""
cursor.execute(query, (ticket_id, airline_name, flight_num))
# conn.commit()

if identity == "customer":
    query = """INSERT INTO purchases
               VALUES (%s, %s, NULL, %s)"""
    t = (ticket_id, customer_email, today)
else:
    query = """SELECT booking_agent_id
               FROM booking_agent
               WHERE email = %s"""
    cursor.execute(query, (agent_email,))
    agent_id = cursor.fetchall()[0][0]
    query = """INSERT INTO purchases
               VALUES (%s, %s, %s, %s)"""
    t = (ticket_id, customer_email, agent_id, today)
cursor.execute(query, t)
conn.commit()
cursor.close()
```

Login:

1. Airline staff

```
SELECT password
FROM %s
WHERE username = %s

SELECT airline_name
FROM airline_staff
WHERE username = %s
```

2. Customer and booking agent

```
SELECT password
FROM %s
WHERE email = %s
```

Register:

1. Airline staff: check airline name exists; check email is unique;

```
SELECT airline_name
FROM airline
WHERE airline_name = %s


SELECT username FROM %s
WHERE username = %s
```

2. Booking agent: check agent id unique; check email is unique

```
SELECT email FROM %s
WHERE email = %s


SELECT email FROM %s
WHERE email = %s
```

3. Customer: check email is unique

```
SELECT email FROM %s
WHERE email = %s
```

```
cursor = conn.cursor(prepared=True)
   query = """INSERT INTO %s""" % identity
   if identity == "customer":
       query += """ VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s) """
       cursor.execute(query, (info["email"].replace("\'", "\'\'"),
info["name"].replace("\'", "\'\'"),
                              generate_password_hash(info["password"], PASSWORD_HASH),
info["building_number"],
                              info["street"].replace("\'", "\'\'"),
info["city"].replace("\'", "\'\'"), info["state"],
                              info["phone_number"], info["passport_number"],
                              info["passport_expiration"], info["passport_country"],
info["date_of_birth"]))
   elif identity == "booking_agent":
       query += " VALUES (%s, %s, %s)"
       cursor.execute(query, (info["email"].replace("\'", "\'\'"),
                              generate_password_hash(info["password"], PASSWORD_HASH),
info["booking_agent_id"]))
   else:
       query += """ VALUES (%s, %s, %s, %s, %s, %s)"""
```

```
        cursor.execute(query, (info["email"].replace("\'", "\'\'"),
                               generate_password_hash(info["password"], PASSWORD_HASH),
                               info["first_name"].replace("\'", "\'\'"),
info["last_name"].replace("\'", "\'\'"),
                               info["date_of_birth"], info["airline_name"]))
   conn.commit()
   cursor.close()
```

Customer

1. Track my spending
    a. Total

```
SELECT SUM(price)
              FROM ticket NATURAL JOIN purchases NATURAL JOIN flight
              WHERE customer_email = %s AND purchase_date BETWEEN %s AND %s
```

    b. Specific time range

```
SELECT purchase_date, price
               FROM ticket NATURAL JOIN purchases NATURAL JOIN flight
               WHERE customer_email = %s AND purchase_date BETWEEN %s AND %s
```

Booking agent

1. Get my commission

```
SELECT SUM(price) * 0.1, COUNT(ticket_id), SUM(price) * 0.1 / COUNT(ticket_id)
              FROM ticket NATURAL JOIN purchases NATURAL JOIN booking_agent NATURAL
JOIN flight
              WHERE email = %s AND (purchase_date >= %s OR %s = '') AND (purchase_date
<= %s OR %s = '')
SELECT SUM(price) * 0.1, COUNT(ticket_id), SUM(price) * 0.1 / COUNT(ticket_id)
              FROM ticket NATURAL JOIN purchases NATURAL JOIN booking_agent NATURAL
JOIN flight
              WHERE email != %s AND (purchase_date >= %s OR %s = '') AND
(purchase_date <= %s OR %s = '')
```

2. View top customers

```
CREATE OR REPLACE VIEW top_customers_ticket AS (
                   SELECT customer_email, COUNT(ticket_id) AS num_of_ticket
                   FROM ticket NATURAL JOIN purchases NATURAL JOIN booking_agent
                   WHERE email = \'%s\' and purchase_date >= \'%s\'
                   GROUP BY customer_email
              )
```

```
CREATE OR REPLACE VIEW top_customers_commission AS (
                   SELECT customer_email, SUM(price) * 0.1 AS amount_of_commission
                   FROM ticket NATURAL JOIN purchases NATURAL JOIN booking_agent
NATURAL JOIN flight
                   WHERE email = \'%s\' and purchase_date >= \'%s\'
                   GROUP BY customer_email
           )

DELIMITER //
CREATE PROCEDURE GetTopFiveCustomerTicket()
BEGIN
   SELECT *
   FROM top_customers_ticket
   ORDER BY num_of_ticket DESC, customer_email
   LIMIT 5;
END //
DELIMITER ;

DELIMITER //
CREATE PROCEDURE GetTopFiveCustomerCommission()
BEGIN
   SELECT *
   FROM top_customers_commission
   ORDER BY amount_of_commission DESC, customer_email
   LIMIT 5;
END //
DELIMITER ;
```

Airline staff

1. Create new flights

```
SELECT flight_num FROM flight WHERE flight_num = %s
SELECT airport_name FROM airport WHERE airport_name = %s
SELECT airport_name FROM airport WHERE airport_name = %s
SELECT airplane_id FROM airplane WHERE airplane_id = %s
INSERT INTO flight
              VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
```

2. Change flight status

```
SELECT flight_num, status
```

```
FROM flight
WHERE flight_num = %s AND airline_name = %s

UPDATE flight
              SET status = %s
              WHERE flight_num = %s AND airline_name = %s
```

3. Add airplane

```
SELECT *
FROM airplane
WHERE airline_name = %s AND airplane_id = %s
INSERT INTO airplane
              VALUES (%s, %s, %s)

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

4. Add airport

```
SELECT *
FROM airport
WHERE airport_name = %s
INSERT INTO airport
              VALUES (%s, %s)

```

5. View booking agent

```
CREATE OR REPLACE VIEW top_agents_ticket AS (
                       SELECT email, COUNT(ticket_id) AS num_of_ticket
                       FROM booking_agent NATURAL JOIN purchases NATURAL JOIN ticket
NATURAL JOIN flight
                       WHERE purchase_date >= \'%s\' AND airline_name = \'%s\'
                       GROUP BY email

DELIMITER //
CREATE PROCEDURE GetTopFiveAgentsTicket()
BEGIN
   SELECT *
   FROM top_agents_ticket
   ORDER BY num_of_ticket DESC, email
   LIMIT 5;
END //
```

```
DELIMITER ;


DELIMITER //
CREATE PROCEDURE GetTopFiveAgentsCommission()
BEGIN
   SELECT *
   FROM top_agents_commission
   ORDER BY t1.amount_of_commission DESC, email
   LIMIT 5;
END //
DELIMITER ;


CREATE OR REPLACE VIEW top_agents_commission AS (
                  SELECT email, SUM(price * 0.1) AS amount_of_commission
                  FROM booking_agent NATURAL JOIN purchases NATURAL JOIN ticket
NATURAL JOIN flight
                  WHERE purchase_date >= \'%s\' AND airline_name = \'%s\'
                  GROUP BY email
              )
```

6. View most frequent customers

```
SELECT customer_email, COUNT(ticket_id)
              FROM purchases NATURAL JOIN ticket
              WHERE purchase_date BETWEEN %s AND %s AND airline_name = %s
              GROUP BY customer_email
              HAVING COUNT(ticket_id) >= ALL (SELECT COUNT(ticket_id)
                                              FROM purchases NATURAL JOIN ticket
                                              WHERE purchase_date BETWEEN %s AND %s
AND airline_name = %s
                                              GROUP BY customer_email)
              ORDER BY customer_email


SELECT COUNT(ticket_id)
              FROM purchases NATURAL JOIN ticket
              WHERE purchase_date BETWEEN %s AND %s AND airline_name = %s
```

```
SELECT DISTINCT flight_num, departure_airport, departure_time, arrival_airport,
arrival_time, status, airplane_id
              FROM purchases NATURAL JOIN ticket NATURAL JOIN flight
              WHERE customer_email = %s AND airline_name = %s
```

7. View my flight

```
SELECT DISTINCT email, name, phone_number, passport_number, date_of_birth
              FROM ticket NATURAL JOIN purchases JOIN customer ON (customer.email =
purchases.customer_email)
              WHERE airline_name = %s AND flight_num = %s
```

8. View reports

```
SELECT ticket_id, purchase_date
              FROM purchases NATURAL JOIN ticket
              WHERE airline_name = %s AND purchase_date BETWEEN %s AND %s
```

9. Comparison of revenue earned

```
SELECT SUM(price)
              FROM ticket NATURAL JOIN purchases NATURAL JOIN flight
              WHERE airline_name = %s AND purchase_date BETWEEN %s AND %s

if t == "direct":
       query += """ AND booking_agent_id IS NULL"""
   else:
       query += """ AND booking_agent_id IS NOT NULL"""
```

10. Get top destinations

```
CREATE OR REPLACE VIEW top_destinations AS (
                   SELECT DST.airport_city AS dst, COUNT(ticket_id) AS num_of_purchase
                   FROM flight AS F JOIN airport AS DST ON (F.arrival_airport =
DST.airport_name)
                       NATURAL JOIN purchases NATURAL JOIN ticket
                   WHERE airline_name = \'%s\' AND purchase_date BETWEEN \'%s\' AND
\'%s\'
                   GROUP BY dst
           )

DELIMITER //
CREATE PROCEDURE GetTopThreeDestination()
BEGIN
   SELECT *
   FROM top_destinations
   ORDER BY num_of_purchase DESC, dst
   LIMIT 3;
END //
DELIMITER ;
```

EXTRA: change info

```
cursor = conn.cursor()
   query = """SELECT * FROM %s""" % identity
   if identity == "airline_staff":
       query += """ WHERE username = %s"""
   else:
       query += """ WHERE email = %s"""
   cursor.execute(query, (email,))
   data = list(cursor.fetchall()[0])
   if identity == "customer":
       data.pop(2)
   else:
       data.pop(1)
   cursor.close()
cursor = conn.cursor(prepared=True)
   if info["email"] != old_email:
       query = """SELECT * FROM %s""" % identity
       if identity == "airline_staff":
           query += """ WHERE username = %s"""
       else:
           query += """ WHERE email = %s"""
       cursor.execute(query, (info["email"],))
       data = cursor.fetchall()
       if data:
           cursor.close()
           return False, "Email has already been used!"

   if old_agent_id:
       if info["booking_agent_id"] != old_agent_id:
           query = """SELECT * FROM %s""" % identity
           query += """ WHERE booking_agent_id = %s"""
           cursor.execute(query, (info["booking_agent_id"],))
           data = cursor.fetchall()
           if data:
               cursor.close()
               return False, "Agent id has already been used!"
   query = """UPDATE %s""" % identity
   if identity == "airline_staff":
       query += """ SET username = %s, first_name = %s, last_name = %s
                    WHERE username = %s
                """
       cursor.execute(query, (info["email"], info["first_name"], info["last_name"], old_email))
   elif identity == "booking_agent":
       query += """ SET email = %s, booking_agent_id = %s
                    WHERE email = %s
                """
       cursor.execute(query, (info["email"], info["booking_agent_id"], old_email))
   else:
       query += """ SET email = %s, name = %s, building_number = %s, street = %s, city = %s, state = %s, phone_number = %s, passport_number = %s, passport_expiration = %s, passport_country = %s
                    WHERE email = %s
               """
       cursor.execute(query, (info["email"], info["name"], info["building_number"], info["street"], info["city"], info["state"],
                              info["phone_number"], info["passport_number"], info["passport_expiration"], info["passport_country"], old_email))

   conn.commit()
   cursor.close()
```