Use Cases

General Cases

1. View Public Flights

Default View: List All the Upcoming Flights
Query: "SELECT * FROM flight WHERE status = ‘upcoming'"

Search: Based On Departure/Arrival Airport, Departure Date
Query:

```
query = "SELECT * FROM flight WHERE 1=1"
if departure_airport:
    query += f" AND depart_airport = '{departure_airport}'"
if departure_time:
    query += f" AND depart_time LIKE '%{departure_time}%'"
if arrival_airport:
    query += f" AND arrive_airport = '{arrival_airport}'"
query += " AND status = 'upcoming'"
```

2. View Flight Status

Default View: List All the Flights
Query: "SELECT * FROM flight"

Search: Based On FlightNumber, Departure/Arrival Date
Query:

```
query = "SELECT * FROM flight WHERE 1=1"
if flight_number:
    query += f" AND flight_num LIKE '%{flight_number}%'"
if departure_time:
    query += f" AND depart_time LIKE '%{departure_time}%'"
if arrival_time:
    query += f" AND arrive_time LIKE ‘%{arrival_time}%’"
```

3. Register

Choices: Customer/Booking Agent/Airline Staff

Users can choose from a dropdown selector. Once they choose which role to register, the required information form to fill would pop up.

Query (Customer as Example):

```
data = (
    request.form.get('customer_email'),
    request.form.get('customer_name'),
    password,
    request.form.get('building_number'),
    request.form.get('street'),
    request.form.get('city'),
    request.form.get('state'),
    request.form.get('phone_number'),
    request.form.get('passport_number'),
    request.form.get('passport_expiration'),
    request.form.get('passport_country'),
    request.form.get('customer_date_of_birth')
)    #get information from the form that the user fills
query = '''
INSERT INTO customer
(email, name, password, building_number, street, city, state, phone_number,
    passport_number, passport_expiration, passport_country, date_of_birth)
VALUES ( %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
'''
```

4. Login

System would automatically check which kind of role the account is. If the password is wrong, it would alert ‘wrong password’. Once logged in, a session would be created, storing all the needed information of the user. The user would be redirected to the user homepage.

Session Code:
```
if user[1] == password:
    session['username'] = user[0]
    session['name'] = user[2]
    session['role'] = role
    if role == 'airline_staff':
        session['permission'] = user[3]
        session['airline'] = user[4]
```

Customer Cases

1. Customer Homepage

Default View: List All the Upcoming Flights That He Purchased

Query:
```
"SELECT f.*
FROM flight f
NATURAL JOIN purchase p
NATURAL JOIN ticket t
WHERE p.customer_email = '{customer_email}'
AND f.status = 'upcoming' "
```

Search: Based On Departure/Arrival Airport, Departure Date

Query:
```
"SELECT f.*
FROM flight f
NATURAL JOIN purchase p
NATURAL JOIN ticket t
WHERE 1=1"
if departure_airport:
    query += f" AND depart_airport = '{departure_airport}'"
if departure_time:
    query += f" AND depart_time LIKE '%{departure_time}%'"
if arrival_airport:
    query += f" AND arrive_airport = '{arrival_airport}'"
query += f" AND p.customer_email = '{customer_email}'"
```

2. Search for Upcoming Flights

Same As in the General Case. ‘View Public Flights’

3. Purchase Tickets

Search: Based On Departure/Arrival Airport, Departure Date

Query: Same As for ‘Search for Upcoming Flights’

Purchase: 3 Cases. If the flight is already bought by the customer, there would be a red ‘Purchased’ button. (Once you click it, it would shake once :D) If there is no more seats for the flight, there would be a red ‘LIMIT’ button (It can also shake :P) If the flight can be purchased, there should be a green ‘Purchase’ button. Once you click it, an alert ‘purchase succeed!’ would pop up, and the purchase record would be stored in the database. A ticket would automatically generated and stored in the database.

Query (for purchase check):
```
SELECT count(1)
FROM purchase p
JOIN ticket t ON p.ticket_id = t.ticket_id
WHERE p.customer_email = %s
AND t.flight_num = %s
```

Query (for seat check):
```
SELECT COUNT(t.ticket_id), a.seat
FROM ticket t
JOIN flight f ON t.flight_num = f.flight_num
JOIN airplane a ON a.id = f.plane_id
WHERE f.flight_num = '{flight_num}'
```

Query (for success purchase):

```
SELECT MAX(ticket_id) FROM ticket.      #get the current max ticket id
ticket_id = int(max_ticket_id) + 1              #generate ticket id automatically
cursor.execute('INSERT INTO ticket (ticket_id, flight_num)
 VALUES (ticket_id, flight_number)')
cursor.execute('INSERT INTO purchase (ticket_id, customer_email, agent_email, date)
 VALUES (ticket_id, customer_email, NULL, purchase_date)')
```

4. My Spending

Default View: total amount of money spent in the past year and a bar chart showing month wise money spent for last 6 months

Query:

```
query = """
SELECT DATE_FORMAT(date, '%Y-%m') as month, SUM(flight.price) as total_spent
FROM purchase
JOIN ticket ON purchase.ticket_id = ticket.ticket_id
JOIN flight ON ticket.flight_num = flight.flight_num
WHERE purchase.customer_email = %s AND purchase.date BETWEEN %s AND %s
GROUP BY DATE_FORMAT(date, '%Y-%m')
ORDER BY month
"""
```

p.s. if none of the date is filled, then the date would be set to default.

Search: Based on Date Range (if one of them is not filled, then there would be no restriction for the start/end date)

Query: Same as default. (The date is automatically judged)

5. Logout

The session would be destroyed. An alert 'Goodbye!' would be pop up, and the user wold be redirected to the index homepage.

Booking Agent Cases

1. Booking Agent Homepage

Default View: List All the Upcoming Flights Of All his Customers

Query:

```
f"SELECT f.*,p.customer_email
FROM flight f
NATURAL JOIN purchase p
NATURAL JOIN ticket t
WHERE p.agent_email = '{agent_email}'
AND f.status = 'upcoming'"
```

Search: Based On Departure/Arrival Airport, Departure Date

Query:

```
query = f"SELECT f.*,p.customer_email
 FROM flight f
 NATURAL JOIN purchase p
 NATURAL JOIN ticket t
 WHERE p.agent_email = '{agent_email}'"
   if departure_airport:
      query += f" AND depart_airport = '{departure_airport}'"
   if departure_time:
      query += f" AND depart_time LIKE '%{departure_time}%'"
   if arrival_airport:
      query += f" AND arrive_airport = '{arrival_airport}'"
   query += f" AND p.agent_email = '{agent_email}'"
```

2. Search for Upcoming Flights

Same As in the General Case. 'View Public Flights'

3. Purchase Tickets

Search: Based On Departure/Arrival Airport, Departure Date

Query: Same As for 'Search for Upcoming Flights'

p.s. but showing only the flights of the airline that he works for

Purchase: 2 Cases. If there is no more seats for the flight, there would be a red 'LIMIT' button (Once you click it, it would shake once :D) If the flight can be purchased, there should be a green 'Purchase' button. Once you click it, a search window would pop up. Agent can search for the email of the customers (p.s. customers who already purchased the ticket would not show up in the results). And by clicking one of the results, a flight is purchased successfully. An alert 'purchase succeed!' would pop up, and the purchase record would be stored in the database. A ticket would automatically generated and stored in the database.

Query (for seat check): same as for customers

Query (for customer search):

```
SELECT c.email FROM customer c
WHERE c.email LIKE %s AND NOT EXISTS (
SELECT 1 FROM purchase p
JOIN ticket t ON p.ticket_id = t.ticket_id
WHERE p.customer_email = c.email AND t.flight_num = %s
```

Query (for success purchase):

```
SELECT MAX(ticket_id) FROM ticket.     #get the current max ticket id
ticket_id = int(max_ticket_id) + 1     #generate ticket id automatically
cursor.execute('INSERT INTO ticket (ticket_id, flight_num)
 VALUES (ticket_id, flight_number)')
cursor.execute('INSERT INTO purchase (ticket_id, customer_email, agent_email, date)
 VALUES (ticket_id, customer_email, agent_email, purchase_date)')
```

4. View My Commission

Default View: 1. total amount of commission received in the past 30 days
2. the average commission received per ticket booked in the past 30 days
3. total number of tickets sold by him in the past 30 days

Query:

```
SELECT SUM(flight.price), AVG(flight.price), COUNT(ticket.ticket_id) as
    total_commission
FROM purchase
JOIN ticket ON purchase.ticket_id = ticket.ticket_id
JOIN flight ON ticket.flight_num = flight.flight_num
WHERE purchase.agent_email = %s AND purchase.date BETWEEN %s AND %s
```

p.s. if none of the date is filled, then the date would be set to default.

Search: Based on Date Range (if one of them is not filled, then there would be no restriction for the start/end date)

Query: Same as default. (The date is automatically judged)

5. View Top Customers

Default View: 1. Show a bar chart showing each of these 5 customers in x-axis and number of tickets bought in y-axis.
2. Show another bar chart showing each of these 5 customers in x-axis and amount commission received in y- axis.

Query (for chart 1):

```
SELECT purchase.customer_email, COUNT(ticket.ticket_id) as total_ticket
FROM purchase
JOIN ticket ON purchase.ticket_id = ticket.ticket_id
JOIN flight ON ticket.flight_num = flight.flight_num
WHERE purchase.agent_email = %s
AND purchase.date BETWEEN %s AND %s
GROUP BY purchase.customer_email
ORDER BY total_ticket DESC LIMIT 5
```

Query (for chart 2):

```
SELECT purchase.customer_email, SUM(flight.price) as total_commission
FROM purchase
JOIN ticket ON purchase.ticket_id = ticket.ticket_id
JOIN flight ON ticket.flight_num = flight.flight_num
```

```
WHERE purchase.agent_email = %s
AND purchase.date BETWEEN %s AND %s
GROUP BY purchase.customer_email
ORDER BY total_commission DESC LIMIT 5
```

p.s. if none of the date is filled, then the date would be set to default.

Search: Based on Date Range (if one of them is not filled, then there would be no restriction for the start/end date)

Query: Same as default. (The date is automatically judged)

6. Logout

The session would be destroyed. An alert 'Goodbye!' would be pop up, and the user wold be redirected to the index homepage.

Airline Staff Cases

1. Airline Staff Homepage

Default View: show all the upcoming flights operated by the airline he works for the next 30 days.

p.s. the flight information would be duplicated to show all customers

Query:
```
f"SELECT f.*,p.customer_email
FROM flight f
NATURAL JOIN purchase p
NATURAL JOIN ticket t
WHERE f.name_airline = '{staff_airline}'
AND f.status = 'upcoming'
AND f.depart_time >= '{today}'
AND f.depart_time <= '{date_30_days_later}'
ORDER BY f.flight_num"
```

Search: Based On Departure/Arrival Airport, Date Range (if one of them is not filled, then there would be no restriction for the start/end date)

Query:
```
f"SELECT f.*,p.customer_email
FROM flight f
NATURAL JOIN purchase p
NATURAL JOIN ticket t
WHERE f.name_airline = '{staff_airline}'"
if departure_airport:
   query += f" AND depart_airport = '{departure_airport}'"
if start_date:
   query += f" AND depart_time >= '{start_date}'"
if end_date:
   query += f" AND depart_time <= '{end_date}'"
if arrival_airport:
   query += f" AND arrive_airport = '{arrival_airport}'"
query += " ORDER BY f.flight_num"
```

2. Create New Flights

Permission Check: System would check in the session and get his permission id. If he doesn't have admin permission, an alert 'You have no permission!' Would pop up, and redirect him back to homepage.

Create: Provide All required data.

p.s. After creating, the page would be reloaded (form would be cleared), and wait for the next creation.

Query:
```
data = (
  request.form.get('flight_num'),
  request.form.get('depart_time'),
  request.form.get('arrive_time'),
  request.form.get('price'),
  request.form.get('status'),
  name_airline,
```

```
        request.form.get('plane_id'),
        request.form.get('depart_airport'),
        request.form.get('arrive_airport'),
    )
    query = '''
    INSERT INTO flight
    (flight_num, depart_time, arrive_time, price, status, name_airline, plane_id, depart_airport,
arrive_airport)
    VALUES ( %s,  %s,  %s,  %s,  %s,  %s,  %s,  %s, %s)
    '''
```

3. Change Flight Status

Permission Check: System would check in the session and get his permission id. If he doesn't have Operator permission, an alert 'You have no permission!' Would pop up, and redirect him back to homepage.

Change: By selecting FlightNum and Status

p.s. After changing, the page would be reloaded (form would be cleared), and wait for the next change.

Query:
```
UPDATE flight
SET status = "{status}"
WHERE flight_num = "{flight_num}"
```

4. Add New Airplane

Permission Check: System would check in the session and get his permission id. If he doesn't have admin permission, an alert 'You have no permission!' Would pop up, and redirect him back to homepage.

Add: By typing in Plane ID and Seat Number

p.s. After adding, the page would be reloaded (form would be cleared), and wait for the next add.

Query (for showing existing airplanes owned by the airline):
```
SELECT id, seat
FROM airplane
WHERE name_airline = '{name_airline}'
```
Query (for adding new airplane):
```
INSERT INTO airplane
(id, name_airline, seat)
VALUES (%s, %s, %s)
```

5. Add New Airport

Permission Check: System would check in the session and get his permission id. If he doesn't have admin permission, an alert 'You have no permission!' Would pop up, and redirect him back to homepage.

Add: By typing in Airport Name and City

Query:
```
INSERT INTO airport
(name, city)
VALUES (%s, %s)
```

6. View Top Booking Agents

Showing: 1. Top 5 booking agents based on number of tickets sales for the past month.
2. Top 5 booking agents based on number of tickets sales for the past year.
3. Top 5 booking agents based on the amount of commission received for the last year.

Query (for ticket sales):
```
SELECT agent_email, COUNT(ticket_id) FROM purchase p
NATURAL JOIN ticket t
NATURAL JOIN flight f
NATURAL JOIN works_for w
WHERE airline_name = "{name_airline}"
AND date >= "{date_30/365_before}"
```

```
                    GROUP BY agent_email
                    ORDER BY COUNT(ticket_id) DESC LIMIT 5
Query (for commission):  SELECT agent_email, SUM(Price) FROM purchase p
                      NATURAL JOIN ticket t
                      NATURAL JOIN flight f
                      NATURAL JOIN works_for w
                      WHERE airline_name = “{name_airline}"
                      AND date >= "{date_365_before}"
                      GROUP BY agent_email
                      ORDER BY SUM(Price) DESC LIMIT 5
```

7. View Frequent Customers

Top Customer: the most frequent customer within the last year

```
Query:  SELECT customer_email, COUNT(ticket_id) FROM purchase p
        NATURAL JOIN ticket t
        NATURAL JOIN flight f
        WHERE name_airline = "{name_airline}" AND date >= "{date_365_before}"
        GROUP BY customer_email
        ORDER BY COUNT(ticket_id) DESC LIMIT 1
```

All Customers: A dropdown selectors of all customers, and once a customer is clicked, show a list of all flights this particular Customer has taken only on that particular airline.

```
Query:  SELECT customer_email, flight_num FROM purchase p
        NATURAL JOIN ticket t
        NATURAL JOIN flight f
        WHERE name_airline = "{name_airline}"
```

8. View Ticket Report

Total amounts of ticket sold based on range of dates. Month wise tickets sold in a bar chart.
(if one of them is not filled, then there would be no restriction for the start/end date)

```
Query:  SELECT DATE_FORMAT(date, '%Y-%m') as month, COUNT(ticket.ticket_id) as
        total_spent
        FROM purchase
        JOIN ticket ON purchase.ticket_id = ticket.ticket_id
        JOIN flight ON ticket.flight_num = flight.flight_num
        WHERE flight.name_airline = %s AND purchase.date BETWEEN %s AND %s
        GROUP BY DATE_FORMAT(date, '%Y-%m')
        ORDER BY month
```

(The date is automatically judged)

9. View Revenue Report

1. A pie chart for showing total amount of revenue earned from direct sales and total amount of revenue earned from indirect sales in the last month.
2. A pie chart for showing total amount of revenue earned from direct sales and total amount of revenue earned from indirect sales in the last year.

```
Query:  SELECT
          SUM(CASE WHEN agent_email IS NULL THEN price ELSE 0 END)
          AS Direct_Sales_Revenue,
          SUM(CASE WHEN agent_email IS NOT NULL THEN price ELSE 0 END)
          AS Indirect_Sales_Revenue
        FROM purchase p
        NATURAL JOIN ticket t
        NATURAL JOIN flight f
        WHERE name_airline = "{name_airline}" AND date >= “{date_30/365_before}”
```

10. View Top Destinations

1. the top 3 most popular destinations for last 3 month.
2. the top 3 most popular destinations for last year.

```
Query:  SELECT a.city, COUNT(t.ticket_id) FROM ticket t
        JOIN flight f ON t.flight_num = f.flight_num
```

```
JOIN airport a ON f.arrive_airport = a.name
WHERE name_airline = "{name_airline}" AND depart_time >= "{date_30/365_before}"
GROUP BY a.city
ORDER BY COUNT(t.ticket_id) DESC LIMIT 3
```

11. Grant New Permissions

Permission Check: System would check in the session and get his permission id. If he doesn't have admin permission, an alert 'You have no permission!' Would pop up, and redirect him back to homepage.

Grant: Grant new permissions to other staffs in the same airline. There would be a dropdown selector with all staff from this airline. Once a staff is selected, his current permission would be shown. And an other two selectors would be 'grant operator' and 'grant admin'. After choosing these two, and then click the 'commit' button, his permissions would be updated in the database.

Query (to show permission):
```
SELECT a.email, p.is_operator, p.is_admin
FROM airline_staff a
NATURAL JOIN permission p
WHERE a.name_airline = '{staff_airline}'
AND a.last_name != '{staff_name}'.  #not choosing himself
```

Query (to update data):
```
UPDATE airline_staff
SET permission_id = "{permission}"
WHERE email = "{email}"
```

12. Add Booking Agents

Permission Check: System would check in the session and get his permission id. If he doesn't have admin permission, an alert 'You have no permission!' Would pop up, and redirect him back to homepage.

Add: Add booking agents that can work for this airline, providing their email address. There would be a dropdown selector with all free (who are not in the 'works_for' table) booking agents. After selecting an agent and click 'commit' button, a new data of 'works_for' (with the staff's airline name) would be written in the database.

Query (for all free agents):
```
SELECT a.email
FROM agent a
LEFT JOIN works_for w ON a.email = w.agent_email
WHERE w.agent_email IS NULL;
```

Query (for new data):
```
INSERT INTO works_for
(agent_email, airline_name)
VALUES ('{email}','{staff_airline}')
```

13. Logout

The session would be destroyed. An alert 'Goodbye!' would be pop up, and the user wold be redirected to the index homepage.

Additional Cases

1. UI Design

Better Layout, Mouse Interaction, Nice Pictures, Clean Design

2. More Customization

More search options, Advanced Button Interaction, Nice Charts

3. Security Check

Prevent User to Access Pages by typing in Directions in the web browser.
Solution: Check Session Stage For Every Page