Part I. REQUIRED Application Use Cases.
All users, whether logged in or not, can have four functions to do: view public info (search for future flights and see flight status); register and login.

1a. public flight search:
Users can either search single-trip flight or round-trip flight. This search function enables fuzzy search where not all parameters are required. Flight information will be displayed after click the "search" button and users will be guided to log in first before purchase tickets. Price of each flight will be changed due to number of tickets sold.

For both single-trip and round-trip flight:

```
        query = 'SELECT * FROM flight as F, airport as A, airport as D WHERE D.name = F.departure_airport_name and A.name = F.arrival_airport_name and F.departure_airport_date=%s'
```

fuzzy search to get flight information

```
        query1 = 'SELECT count(*) as C FROM ticket as T WHERE T.flight_number = %s and T.airline_name = %s and T.ticket_id not in (SELECT P.ticket_id FROM purchase as P WHERE P.ticket_id = T.ticket_id)'
```

check the number of tickets sold for each flight

```
        query2 = 'SELECT base_price FROM flight WHERE flight_number = %s and airline_name = %s'
```

return base price to change the sold price for each flight

```
        query3 = 'SELECT count(*) as C FROM ticket WHERE flight_number = %s and airline_name = %s'
```

return the total amounts of tickets for each flight

1b. flight status search:
Users must input airline name and flight number to search for flight status. They can choose to input departure and arrival dates but they are not required.

```
        query = 'SELECT * FROM flight WHERE airline_name = %s and flight_number = %s'
```

select flights with given airline name and flight number

2. Register
Three types of users will be led to three web pages. For each register page, will first check whether the user already exists. For security reason, staffs are given the special permission code "staffcode" to register.

1) Staff:
To register as a staff, one should input username, password, permission_code ("staffcode"), first_name, last_name, date_of_birth and airline_name

```
        query1 = 'SELECT * FROM staff WHERE username = %s'
```

```
    if the staff already exists, return error.
    query2 = 'SELECT * from airline where name = %s
    if no airline found, return error.
    else:
    ins = 'INSERT INTO staff VALUES(%s, %s, %s, %s, %s, %s)'
    insert this new staff into database
```

2) Customer:
To register as a customer, one should input customer_email, password, customer_name, building_number, street, city, state, phone_number, passport_number, passport_expiration, passport_country and date_of_birth.

```
    query = 'SELECT * FROM customer WHERE customer_email = %s'
    if the customer already exists, return error.
    ins = 'INSERT INTO customer VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)'
    insert this new customer into database
```

3) Booking agent:
To register as a booking agent, one should input agent_email, password and booking_agent_id.

```
    query = 'SELECT * FROM booking_agent WHERE agent_email = %s'
    if the agent already exists, return error.
    ins = 'INSERT INTO booking_agent VALUES(%s, %s, %s)'
    insert this new agent into database
```

3. Login
To log in, users need to input username, password and user_type. User_type should only be staff, customer or booking agent.

```
    if user_type == "staff":
        query = 'SELECT * FROM staff WHERE username = %s and password = %s'
        check if the staff exists in the database
    elif user_type == "customer":
        query = 'SELECT * FROM customer WHERE customer_email = %s and password = %s'
        check if the customer exists in the database
    elif user_type == "booking_agent":
        query = 'SELECT * FROM booking_agent WHERE agent_email = %s and password = %s'
        check if the booking agent exists in the database
```

Part II. Customer use cases.
The customer home page will first welcome the customer and show his username and then display all the functions. There are five functions: view my flights; search for flights; purchase tickets; track my spending and log out.
4. View my flights:

The default will be showing all future flights for the customer and he/she could also search with specified parameters.

query = 'SELECT * FROM flight natural join ticket natural join purchase where departure_airport_date >= NOW() AND customer_email = %s'

the default will be showing all the future flights that customer is going to take

query = 'SELECT * FROM flight natural join ticket natural join purchase WHERE customer_email = %s and arrival_airport_name = %s and departure_airport_name = %s and departure_airport_date = %s'

this query enables customer to specify dates and airports for future flight search

5. search for flights:
Customers can either search single-trip flight or round-trip flight. This search function enables fuzzy search where not all parameters are required. Flight information will be displayed after clicking the "search" button and users can directly purchase the tickets clicking the "buy" button. Price of each flight will be changed due to number of tickets sold.
Queries of flight search are the same with public flight search.

6. purchase tickets:
Customers can directly purchase tickets in the flight search page. When they purchase tickets, the agent_email will be "dummy".

query = 'SELECT * FROM ticket natural join flight WHERE flight_number = %s and airline_name = %s and ticket_id not in (SELECT P.ticket_id FROM purchase as P WHERE P.ticket_id = ticket_id)'

check whether there are remaining tickets to purchase.

query2 = 'SELECT * FROM booking_agent where agent_email = %s'

check if the "dummy" agent exists. If customer buy tickets without agents, the agent_email will be "dummy"

query3 = 'INSERT INTO booking_agent (agent_email, password, booking_agent_id) values(%s,%s,%s)'

insert the "dummy" agent into database if it does not exist.

query1 = 'INSERT INTO purchase (ticket_id,customer_email,agent_email,card_type,card_number,card_name,expiration_date,sold_price) VALUES(%s, %s, %s, %s, %s, %s, %s, %s)'

insert the purchase information into the database, with agent_email being "dummy"

7. track my spending:
Customers are able to see their total spending in the past year and monthly spending for the past 6 months.

query = 'SELECT customer_email, sum(sold_price) as total_spending from purchase where customer_email = %s and purchase_ts <= NOW() and purchase_ts > NOW() - INTERVAL 1 YEAR'
return their total spending in the past year

query = 'SELECT customer_email, sum(sold_price) as monthly_spending FROM purchase where customer_email = %s and purchase_ts >= NOW() - INTERVAL %s MONTH AND purchase_ts < NOW() - INTERVAL %s MONTH'
return their month wise spending for the past 6 months

Part III. Booking agent use cases.
The booking agent home page will first welcome the agent and show his agent email and then display all the functions. There are six functions: view my flights; search for flights; purchase tickets; view my commission; view top customers and log out.
4. view my flights:
The default will be showing all future flights the agent has booked for customers and he/she could also search with specified parameters.
query = 'SELECT customer_email, ticket_id, flight_number, airline_name, arrival_airport_name, arrival_airport_date, arrival_airport_time, departure_airport_name, departure_airport_date, departure_airport_time, status FROM flight natural join ticket natural join purchase where departure_airport_date >= NOW() and agent_email = %s'
return all future flights the agent has booked for all customers

query = 'SELECT customer_email, ticket_id, flight_number, airline_name, arrival_airport_name, arrival_airport_date, arrival_airport_time, departure_airport_name, departure_airport_date, departure_airport_time, status FROM flight natural join ticket natural join purchase where agent_email = %s and departure_airport_name = %s and arrival_airport_name = %s and departure_airport_date >= %s and arrival_airport_date<= %s '
the agent could also search for booked flights with specified airports or dates

5. search for flights:
Agents can either search single-trip flight or round-trip flight. This search function enables fuzzy search where not all parameters are required. Flight information will be displayed after clicking the "search" button and agents can directly purchase the tickets clicking the "buy" button. Price of each flight will be changed due to number of tickets sold.
Queries of flight search are the same with public flight search and customer flight search.

6. purchase tickets:
Agents can directly purchase tickets for customers in the flight search page. When they purchase tickets, they need to input the customer email and other required information.
query = 'SELECT * FROM ticket natural join flight WHERE flight_number = %s and airline_name = %s and ticket_id not in (SELECT P.ticket_id FROM purchase as P WHERE P.ticket_id = ticket_id)'

first check if there are remaining tickets for the flight

query1 = 'INSERT INTO purchase (ticket_id,customer_email,agent_email,card_type,card_number,card_name,expiration_date,sold_price) VALUES(%s, %s, %s, %s, %s, %s, %s, %s)'

insert purchase information into the database

7. View my commission:
Default will be showing total amount of commission, average commission and tickets sold in the past month. Agents can also specify range of dates to view the commission.

query = 'SELECT agent_email, sum(sold_price * 0.1) as total_commission, sum(sold_price*0.1)/count(ticket_id) as average_commission, count(ticket_id) as tickets_sold from purchase where agent_email = %s and purchase_ts <= NOW() and purchase_ts > NOW() - INTERVAL 1 MONTH'

view total commission; average commission and tickets sold in the past month

query = 'SELECT agent_email, sum(sold_price * 0.1) as total_commission, sum(sold_price*0.1)/count(ticket_id) as average_commission, count(ticket_id) as tickets_sold from purchase where agent_email = %s and purchase_ts >= %s and purchase_ts <= %s '

view total commission; average commission and tickets sold within specified range of dates

8. View top customers:

query1 = 'SELECT customer_email, count(*) AS tickets_sold FROM purchase where agent_email = %s AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 6 MONTH GROUP BY customer_email ORDER BY tickets_sold DESC LIMIT 5'

display top 5 customers based on number of tickets sold in the past 6 months

query2 = 'SELECT customer_email, sum(sold_price * 0.1) AS total_commission FROM purchase where agent_email = %s AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 YEAR GROUP BY customer_email ORDER BY total_commission DESC LIMIT 5'

display top 5 customers based on amount of commissions received in the past year

Part IV. Airline Staff use cases.
The staff home page will first welcome the staff and show his username and then display all the functions. There are eleven functions: view future flights; create new flights; change status of flights; add airplane in the system; add new airport in the system; view top 5 booking agents; view frequent customers; view reports; comparison of revenue earned; view top destinations and logout.
4. View future flights:
The default will be showing all future one-month flights of the airline company this staff belong to. He/she can also specify departure date and airports of future flights.

query1 = 'SELECT airline_name FROM staff where username = %s' first get the airline name that the staff belongs to

query2 = 'SELECT * FROM flight where airline_name = %s and departure_airport_date >= NOW() AND departure_airport_date < NOW() + INTERVAL 1 MONTH'

for the default, show all future flights of this airline company in one month

query = 'SELECT * FROM flight WHERE airline_name = %s and arrival_airport_name = %s and departure_airport_name = %s and departure_airport_date = %s'

optional flight search with specified departure date and airport names

5. create new flights:
The create new flights page will first show all future flights in one month (the default) and then enable staff to add flights for his/her airline company. Tickets will also be automatically generated with the same number of seats of the airplane.

1)default page:

query1 = 'SELECT airline_name FROM staff where username = %s'

first get the airline name that the staff belongs to

query2 = 'SELECT * FROM flight where airline_name = %s and departure_airport_date >= NOW() AND departure_airport_date < NOW() + INTERVAL 1 MONTH'

the default will be showing all future flights of this airline company in one month

2) add new flights page:

query4 = 'SELECT * FROM flight WHERE flight_number = %s and airline_name = %s'

first check whether this flight has already existed, if already existed, return error

query7 = 'SELECT airline_name FROM staff WHERE username = %s'

then check whether the airline name of the new flight is the airline this staff works for, if not, return error

query1 = 'SELECT * FROM airplane WHERE airline_name = %s AND airplane_id = %s'
query2 = 'SELECT * FROM airport WHERE name = %s'
query3 = 'SELECT * FROM airport WHERE name = %s'

query1-3 checks whether the airline name, airplane id, airport names exist, if not, return error

query5 = 'INSERT INTO flight VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)'

if all the above conditions don't return error, insert the new flight into database

query6 = 'SELECT seats FROM airplane where airplane_id = %s and airline_name = %s'

get the number of seats of the airplane. Same amount of tickets will be automatically generated

```
    for i in range(number of total seats):
        query = 'INSERT INTO ticket (ticket_id, flight_number, airline_name) VALUES (%s, %s, %s)'
        generate the same amount of tickets as seats into database
```

6. change status of flights:
Staffs are only authorized to update status of flights in his/her airline. Also, the flight should already exist in the database.

```
    query1 = 'SELECT * FROM flight WHERE flight_number = %s AND airline_name = %s'
    first check if the flight exists. if not, return error

    query4 = 'SELECT airline_name FROM staff WHERE username = %s'
    then check if the flight belongs to the airline the staff works for. if not, return error

    query2 = 'UPDATE flight SET status = %s WHERE flight_number = %s AND airline_name = %s'
    if all the above conditions are met and return no error, the staff is able to update flight status
```

7. add airplane in the system:
Staffs are only authorized to add new airplanes for his/her airline company. In the confirmation page, he/she will also be able to see all airplanes owned by the airline.
1)add airplane page

```
    query1 = 'SELECT * FROM airplane WHERE airplane_id = %s AND airline_name = %s'
    first check whether the airplane already exists. if yes, return error.

    query2 = 'SELECT * FROM airline WHERE name = %s'
    query4 = 'SELECT airline_name FROM staff WHERE username = %s'
    then check if the new airplane belongs to the airline company that the staff works for, if not, return error

    query3 = 'INSERT INTO airplane (airplane_id,airline_name,seats) VALUES (%s, %s, %s)'
    if all the above conditions are met and returns no error, the new airplane will be added into the database
```

2)confirmation page

```
    query1 = 'SELECT airline_name FROM staff WHERE username = %s'
    query2 = 'SELECT * FROM airplane WHERE airline_name = %s'
    return all airplanes of the airline company this staff works for
```

8. add new airport in the system:

```
    query1 = 'SELECT * FROM airport WHERE name = %s'
    first check if the new airport already exists in the database, if yes, return error
```

query2 = 'INSERT INTO airport(name, city) VALUES (%s, %s)'
insert the new airport into the database

9. view top 5 booking agents:
Staffs can choose from three ways to view the top 5 agents: tickets sold in past one month; tickets sold in past one year or based on total commission in past one year.

query1 = 'SELECT agent_email, count(*) AS count FROM purchase where agent_email <> "dummy" AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 MONTH GROUP BY agent_email ORDER BY count DESC LIMIT 5'
select top 5 agents based on total tickets sold in past one month

query2 = 'SELECT agent_email, count(*) AS count FROM purchase where agent_email <> "dummy" AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 YEAR GROUP BY agent_email ORDER BY count DESC LIMIT 5'
select top 5 agents based on total tickets sold in past one year

query3 = 'SELECT agent_email, sum(sold_price * 0.1) AS count FROM purchase where agent_email <> "dummy" AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 YEAR GROUP BY agent_email ORDER BY count DESC LIMIT 5'
select top 5 agents based on total commission in past one year

10. view frequent customers:
query1 = 'SELECT airline_name FROM staff WHERE username = %s'
first get the airline name the staff works for

query2 = 'DROP VIEW IF EXISTS customer_view'
query3 = 'CREATE VIEW customer_view AS SELECT customer_email, COUNT(flight_number) AS num FROM purchase NATURAL JOIN ticket WHERE airline_name = %s GROUP BY customer_email'
query4 = 'SELECT distinct customer_email, flight_number FROM purchase NATURAL JOIN ticket WHERE airline_name = %s AND customer_email in (SELECT customer_email FROM customer_view WHERE num = (SELECT MAX(num) FROM customer_view))'
first drop the view "customer_view" if it already exists. Then create the view which shows the number of tickets sold for each customer. After that, select the customer who purchases most tickets and all flights he/she takes/will take.

11. view reports:
query1 = 'SELECT count(*) as num_t FROM purchase natural join ticket WHERE airline_name = %s AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 MONTH'
Get the number of tickets sold last month. Use it to plot the bar chart.

query2 = 'SELECT count(*) as num_t FROM purchase natural join ticket WHERE airline_name = %s AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 YEAR'
Get the number of tickets sold last year

query = 'SELECT count(*) as num_t FROM purchase natural join ticket where airline_name = %s and purchase_ts >= NOW() - INTERVAL %s MONTH AND purchase_ts < NOW() - INTERVAL %s MONTH'
Execute this query 12 times to get the month wise number of tickets sold during previous year. Use this to plot bar chart

For view reports optional:
query = 'SELECT count(*) as num_t from purchase natural join ticket where airline_name = %s and purchase_ts >= %s and purchase_ts <= %s'
Get the number of ticket sold during the selected time period

query = 'SELECT count(*) as num_t FROM purchase natural join ticket where airline_name = %s and purchase_ts >= %s - INTERVAL %s MONTH AND purchase_ts < %s - INTERVAL %s MONTH'
Execute this query n times, where n is the number of months during the selected time period, to get the month wise number of tickets sold. Use this to plot bar chart.

12. comparison of revenue earned:
Total amount of revenue earned from direct sales and indirect sales are compared in the last month and last year.
query1 = 'SELECT sum(sold_price) as revenue FROM purchase natural join ticket WHERE airline_name = %s and agent_email = "dummy" AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 MONTH'
for direct sales (no booking agent) in the last month

query2 = 'SELECT sum(sold_price) as revenue FROM purchase natural join ticket WHERE airline_name = %s and agent_email <> "dummy" AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 MONTH'
for indirect sales (with booking agent) in the last month

query3 = 'SELECT sum(sold_price) as revenue FROM purchase natural join ticket WHERE airline_name = %s and agent_email = "dummy" AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 YEAR'
for direct sales (no booking agent) in the last year

query4 = 'SELECT sum(sold_price) as revenue FROM purchase natural join ticket WHERE airline_name = %s and agent_email <> "dummy" AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 YEAR'
for indirect sales (with booking agent) in the last year

13. view top destinations:
Staffs can view top destinations in the past three months or in the past year.

query1 = 'SELECT airport.city as name, count(*) AS count FROM purchase NATURAL JOIN ticket NATURAL JOIN flight, airport WHERE airline_name = %s AND airport.name = arrival_airport_name AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 3 MONTH GROUP BY arrival_airport_name ORDER BY count DESC LIMIT 3'

return top 3 destinations in the past three months

query2 = 'SELECT airport.city as name, count(*) AS count FROM purchase NATURAL JOIN ticket NATURAL JOIN flight, airport WHERE airline_name = %s AND airport.name = arrival_airport_name AND purchase_ts <= NOW() AND purchase_ts > NOW() - INTERVAL 1 YEAR GROUP BY arrival_airport_name ORDER BY count DESC LIMIT 3'

return top 3 destinations in the past one year