General Pre-login Usage

1) Initial homepage (HTML: index)

A series of videos about traveling is displayed (all the ‘read more’ and social platform buttons are not working at the current stage).

2) View Public Info — search upcoming flights (search bar; FlightSearch, HTML: FlightSearch)

Search for forthcoming flights based on source city/airport name, destination city/airport name, and date. You can use arbitrary combinations of city and airport name, as long as the city is not in upper case (not all in upper case) and the airport is in upper case.
After clicking the button and submitting the form, airline name, flight number, departure airport/time, arrival airport/time, price, and status.
Query (this deals with arbitrary inputs of arrival/departure city and airport. We distinguish airport and city by using .isupper().

```
# enter departure airport and arrival airport
query = "SELECT airline_name, flight_num, departure_airport,
arrival_airport, departure_time, arrival_time, price, status FROM flight
WHERE departure_airport = \'{}\' AND arrival_airport = \'{}\' AND
departure_time LIKE \'{}\'"

# enter departure airport and arrival city
query = "SELECT airline_name, flight_num, departure_airport,
arrival_airport, departure_time, arrival_time, price, status \
               FROM airport AS a, flight AS b\
               WHERE a.airport_name = b.arrival_airport\
               AND b.departure_airport = \'{}\' \
               AND a.airport_city = \'{}\' \
               AND departure_time LIKE \'{}\'"

# enter departure city and arrival airport
query = "SELECT airline_name, flight_num, departure_airport,
arrival_airport, departure_time, arrival_time, price, status \
               FROM airport AS a, flight AS b\
               WHERE a.airport_name = b.departure_airport\
               AND a.airport_city = \'{}\' \
               AND b.arrival_airport = \'{}\' \
               AND departure_time LIKE \'{}\'"

# enter arrival and departure cities
query = "SELECT airline_name, flight_num, departure_airport,
arrival_airport, departure_time, arrival_time, price, status \
               FROM airport AS a, flight AS b, airport AS c \
               WHERE a.airport_name = b.departure_airport\
               AND b.arrival_airport = c.airport_name\
               AND a.airport_city= \'{}\' \
               AND c.airport_city = \'{}\' \
               AND departure_time LIKE \'{}\'"
```

3) View Public Info — check status (search bar: StatusCheck, HTML: StatusCheck)

Be able to see the flight status based on airline name, flight number, and departure date.
Query:

```
query = "SELECT airline_name, flight_num, departure_airport,
arrival_airport, departure_time, arrival_time, status \
           FROM flight WHERE airline_name = \'{}\' AND flight_num = \'{}\'
AND departure_time LIKE \'{}\'"
```

4) View Public Info — about (search bar: About, HTML: About)

Some information about the authors, Pinyu and Qianyu.

5) Register (search bar: Register, HTML: Register)

We implement 3 types of user registrations (Customer, Booking agent, Airline Staff) option via forms on the same page. If the registration succeeds, the user will be redirected to the login page; an error message will be displayed if registration does not succeed.
The passwords are encoded using md5(), which means we canot see the proper password even inside the database!
Query:
First we check whether there is an existing account for the email/username by running

```
query = "SELECT * FROM booking_agent WHERE email = %s"
```

If not existing, we insert a new record:

```
ins = "INSERT INTO booking_agent VALUES(%s,%s,%s)"
```

6) Login (search bar: Login, HTML: Login)

We implement 3 types of user logins (Customer, Booking agent, Airline Staff) option via forms again on the same page. If the login succeeds, the user will be redirected to the user page; an error message will be displayed if the login does not succeed.
Customer login Query:

```
query = "SELECT * FROM customer WHERE email = %s and password = %s"
```

Agent login Query:

```
query = "SELECT * FROM booking_agent WHERE email = %s and password = %s"
```

Staff login Query:

```
query = "SELECT * FROM airline_staff WHERE username = %s and password = 
%s"
```

One highlight about our registration and login is we managed to arrange the three types of login/registration on the same page, which looks good and is more convenient.

Customer login usage

1) View my flights (search bar: ViewMyFlights, HTML: CustomerViewMyFlights)

Defaults is showing all the upcoming flight information which he/she purchased.
Additionally, he/she can specify a range of dates, arrival and departure airport name or city name.
Queries are the same as public flight search (4 situations are considered based on input type), but with customer email specified. One example is

```
q = 'SELECT airline_name, flight_num, departure_airport, arrival_airport,
departure_time, arrival_time, price, status, ticket_id \
               FROM flight NATURAL JOIN ticket NATURAL JOIN purchases \
               WHERE customer_email = \'{}\' AND departure_time <= \'{}\'
AND departure_time >= \'{}\' \
               AND departure_airport = \'{}\' AND arrival_airport = \'{}\'
'
```

2) Search for flights (search bar: SearchFlights, HTML: CustomerSearchFlights)

Search for upcoming flights based on source city/airport name, destination city/airport name, and departure date.
Queries are the same as the public search.

3) Purchase tickets (search bar: PurchaseTickets, HTML: CustomerPurchaseTickets)

After searching for flights, use the unique combination of airline name and flight number to purchase tickets. You could buy multiple tickets for a flight, and the SQL query will check whether there are enough tickets left. Success/failure messages will be displayed.
Query: we first use this query to check available tickets based on input information,

```
query = "SELECT airline_name, flight_num, departure_airport,
arrival_airport, departure_time, arrival_time, price, status,
seats-COUNT(*) as remaining_seats FROM flight join airplane using
(airline_name,airplane_id) join ticket using (airline_name,flight_num)
```

```
WHERE airline_name = \'{}\' AND flight_num = \'{}\' GROUP BY airline_name,
flight_num"
```
Then we insert tickets if there is available tickets left.
```
INSERT INTO ticket (ticket_id, airline_name, flight_num) VALUES (%s, %s,
%s)
```

4) Track my spending (search bar: TrackSpending, HTML: CustomerTrackSpending)

The default view is the total amount of money spent in the past year and a bar chart showing month-wise money spent for the last 6 months.
He/she will also have the option to specify a range of dates to view the total amount of money spent within that range and a bar chart showing month-wise money spent within that range.
We made beautiful bar charts and used calendars for date range selection.
We use this query to get the total money spent by the customer in a particular time period:
```
query = "SELECT  IFNULL(sum(price),0)  as sum_spending from ticket NATURAL
JOIN purchases NATURAL JOIN flight WHERE customer_email = \'{}\'  and
purchase_date between \'{}\' and \'{}\'"
```

5) Logout (search bar: Logout)

Returned to the pre-login homepage.
We use session to store login information such as the username/email. Stored info will be thrown after the user logout.

Booking agent login usage

1) View my flights

The default is showing for the upcoming flights the agent booked the ticket from.
Additionally, the user may specify a range of dates, departure and arrival airport name or city name to show all the flights for which he/she purchased tickets.
Default query:

```
query = 'SELECT airline_name, flight_num, departure_airport,
arrival_airport, departure_time, arrival_time, price, status, ticket_id \
           FROM flight NATURAL JOIN ticket NATURAL JOIN purchases \
           WHERE departure_time >= \'{}\' AND departure_time <= \'{}\' AND
booking_agent_id = \'{}\''
```

The queries specifying departure time, departure, and arrival places are similar to the customer view my flights, but using booking_agent_id instead of customer_email.

2) Search for flights (search bar: SearchFlights, HTML: AgentSearchFlights)

Search for upcoming flights based on source city/airport name, destination city/airport name, and departure date.
The same as customer search.

3) Purchase tickets (search bar: PurchaseFlights, HTML: AgentPurchaseTickets)

Booking agent chooses a flight based on the search result and purchases tickets for other customers using their email.
Check the agent's airline to see whether the agent is eligible to book this flight:

```
query = 'SELECT airline_name FROM works_for WHERE email = \'{}\' AND
airline_name = \'{}\''
```

Then add the ticket to the database:

```
query_insert_ticket = "INSERT INTO ticket (ticket_id, airline_name,
flight_num) VALUES (%s, %s, %s)"
```

4) View my commission (search bar: MyCommission, HTML: AgentMyCommission)

The 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 the option to specify a range of dates to view total amount of commission received and total numbers of tickets sold.
We made beautiful parallel bar charts for information visualization.

We use this query to get the total commission obtained by the agent in a particular time period:

```
query = 'SELECT CAST(0.1*SUM(price) as decimal(10,2)) as TotalPrice,
CAST((0.1*price)/COUNT(*) as decimal(10,2)) as AvePrice, COUNT(*) as
NumTicket\
       FROM purchases NATURAL JOIN ticket NATURAL JOIN flight \
   WHERE booking_agent_id = \'{}\' AND purchase_date <= \'{}\' AND
purchase_date >= \'{}\''
```

5) View top customers (search bar: TopCustomers, HTML: AgentTopCustomer)

Two parallel bar charts are displayed to show the top 5 customers based on the number of tickets bought from the booking agent in the past 6 months and the top 5 customers based on the amount of commission received in the last year.
These 5 customers are on the x-axis, and the number of tickets bought/commissioned in the y-axis. By default, the commission rate is 10% of the ticket price.
If there is fewer than 5 customers whom the agent has booked a ticket for, the bar chart will have fewer columns.

We use this query to get the top 5 customers (below are one of the queries):

```
query = "SELECT name, count(ticket_id) as TotalTicket FROM purchases RIGHT
JOIN customer ON (customer.email = purchases.customer_email)\
       WHERE booking_agent_id = \'{}\'\
       AND purchase_date <= \'{}\' AND purchase_date >= \'{}\'\
       GROUP BY customer_email\
       ORDER BY COUNT(ticket_id) DESC LIMIT 5"
```

6) View Customer Info (search bar: CusInfo, HTML: AgentCustomerInfo)

We made an additional function, letting the booking agent to view partial information of the customers he/she booked tickets for previously. Input email and name, address, phone number, birthday are displayed. Sensitive data like passport is hidden.

We use this query to get specific information about a customer based on his/her email:

```
query = 'SELECT email, name, building_number, street, city, state,
phone_number, date_of_birth FROM customer WHERE email = \'{}\''
```

7) Logout (search bar: Logout)

Airline staff login usage

1) View my flights (search bar: ViewFlights, HTML: StaffViewMyFlights)

Default is showing all the upcoming flights operated by the airline he/she works for the next 30 days. Staff will be able to see all the current/future/past flights operated by the airline he/she works for a range of dates, arrival/departure airports/cities.

Here we set a procedure in the Air database to extract the airline information from staff.

```
#Use procedure to get staff company
cursor.callproc('get_staff_company',(username,))
airline_name = cursor.fetchone()['airline_name']
```

We use this query to get all the incoming flights in the particular airline:

```
query = "SELECT airline_name, flight_num, departure_airport,
arrival_airport, departure_time, arrival_time, price, status FROM flight
NATURAL join airline_staff WHERE departure_time >= \'{}\' and
departure_time <= \'{}\' and airline_name = (SELECT airline_name from
airline_staff WHERE username = \'{}\')"
```

2) View Flight passengers (search bar: ListFlightCus, HTML: StaffViewFlightCustomer)

Staff is able to see all the customers of a particular flight based on flight number. They could only see customers on flights owned by his/her airline! Otherwise, the request will be denied with an error message.

We use this query to fetch all the customer name in the particular flight:

```
query2 = "SELECT name from customer where email IN (SELECT customer_email
from purchases where ticket_id IN (SELECT ticket_id from ticket WHERE
airline_name = (SELECT airline_name from airline_staff WHERE username=
\'{}\') and flight_num = \'{}\'))"
```

3) Change the status of flights (search bar: Change, HTML: StaffChange)

He or she changes a flight status (from upcoming to in progress, in progress to delayed, etc) via forms. Request from unauthorized staff without "Operator" permission from doing this action.
We use the following query to change flight status:

```
query4 = "UPDATE flight SET status = %s WHERE airline_name = %s and flight_num=%s"
```

4) Add new flights/airplanes/airport/new permission/new agents (search bar: Add, HTML: StaffAdd)

Staff with “Admin” permission could use forms to:
1) create new flights for his/her airline;

```
ins = "INSERT INTO flight VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s)"
```

2) add airplane for his/her airline;

```
ins = "INSERT INTO airplane VALUES(%s,%s,%s)"
```

3) add a new airport to the system;

```
ins = "INSERT INTO airport VALUES(%s,%s)"
```

4) add booking agents to his/her airline;

```
ins = "INSERT INTO works_for VALUES(%s,%s)"
```

5) grant permissions for staff working for the same airline.

```
ins = f"UPDATE permission SET permission_id = 3 WHERE username = \'{staff_to_grant}\'"
```

All the request need “Admin” and the query will be denied if validation fails.
We use triggers to prevent illegal insert: for example, wrong format for airport or wrong email format.

5) View top booking agents (search bar: TopAgt, HTML: StaffViewTopAgents)

Listed the top 5 booking agents based on the number of ticket sales/commissions received for the past 1 and 6 months.
4 tables are shown in total.
We create view to store this information:

```
reate_view_query = "CREATE or REPLACE VIEW commission as ( SELECT
booking_agent_id,sum(price*0.1) as total_commission, count(*) as
num_of_tickets from purchases NATURAL join ticket NATURAL join flight
WHERE purchase_date BETWEEN \'{}\' AND \'{}\' GROUP BY booking_agent_id
HAVING booking_agent_id is NOT Null)"
```

Select top 5 agents:

```
query = "SELECT booking_agent_id from commission c1 WHERE 4>=(SELECT
COUNT(DISTINCT booking_agent_id) from commission c2 WHERE
c2.total_commission > c1.total_commission)"
```

6) View frequent customers (search bar: FreqCust, HTML: StaffViewFrequentCustomer)

Staff is 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.

We create another view to store the flight frequency information:

```
create_view_query = "CREATE or REPLACE VIEW top_customer as (SELECT
name,customer_email,phone_number,passport_country, count(*) as
total_flights from purchases NATURAL JOIN ticket JOIN customer ON
purchases.customer_email=customer.email WHERE purchase_date BETWEEN \'{}\'
AND \'{}\' and airline_name = \'{}\' GROUP BY customer_email)"
```

And the most frequent customer is selected by this following query:

```
query = "SELECT * from top_customer t1 WHERE 0=(SELECT COUNT(DISTINCT
customer_email) from top_customer t2 WHERE t2.total_flights>
t1.total_flights)"
```

The flight list is obtained by

```
query="SELECT
name,customer_email,airline_name,flight_num,ticket_id,departure_time,depar
ture_airport,arrival_time,arrival_airport, price, status FROM ticket
NATURAL join purchases NATURAL join flight JOIN customer on
purchases.customer_email=customer.email WHERE airline_name = %s AND
customer_email = %s"
```

7) View reports (search bar: Report, HTML: StaffReport)

Total tickets sold based on the range of dates/last year/month etc. Month-wise tickets sold in a bar chart.
The total amount in the past year is obtained by

```
query = "SELECT count(*) as ticket_sold\
           from purchases NATURAL JOIN ticket WHERE airline_name = \'{}\'\
           and purchase_date BETWEEN \'{}\' and \'{}\'"
```

Using different date ranges, we get the data for the bar chart.

8) Comparison of revenue earned (search bar: Compare, HTML: StaffComparison)

Two pie charts is displayed to show 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 direct revenue is obtained by this query

```
query2 = "SELECT IFNULL(sum(price),0) as direct_sum from purchases NATURAL
JOIN ticket NATURAL JOIN flight WHERE airline_name = \'{}\'  and
purchase_date BETWEEN \'{}\' and \'{}\' GROUP BY booking_agent_id HAVING
booking_agent_id is Null"
```

Indirect revenue is obtained by

```
query3 = "SELECT IFNULL(sum(price),0) as indirect_sum from purchases
NATURAL JOIN ticket NATURAL JOIN flight WHERE airline_name = \'{}\'  and
purchase_date BETWEEN \'{}\' and \'{}\' GROUP BY booking_agent_id HAVING
booking_agent_id is NOT Null"
```

9) Vew Top Destinations (search bar: Dest, HTML: StaffViewPopularDestinations)

Find the top 3 most popular destinations for the last 3 months and last year.

We create a view to store the information:

```
create_view_query = "CREATE or REPLACE view popular_dest as (SELECT airport_city, COUNT(*) AS num_flights FROM airport P JOIN flight F on P.airport_name=F.arrival_airport JOIN ticket T on F.airline_name=T.airline_name and F.flight_num=T.flight_num WHERE F.departure_time BETWEEN \'{}\' and \'{}\' GROUP BY P.airport_city)"
```

The top destination is obtained by

```
query = "SELECT airport_city from popular_dest p1 WHERE 2>=(SELECT COUNT(DISTINCT airport_city) from popular_dest p2 WHERE p2.num_flights > p1.num_flights)"
```

10) Logout (search bar: Logout)