Public info

Check flight:

Departure flight:

```
query = "select * from flight where airline_name=%s and flight_num=%s"\
"and date(departure_time)=%s"
cur.execute(query, (airline_name, flight_num, departure_date ))
```

Return flight:

```
query = "select * from flight where departure_airport=%s and
arrival_airport=%s"\
"and date(arrival_time)=%s"
cur.execute(query, (airline_name, flight_num, return_date ))
```

Search flight:

Departure flight:

```
query = "select * from flight where departure_airport=%s and arrival_airport=%s
and date(departure_time)=%s"
cur.execute(query, (departure_airport, arrival_airport, departure_date ))
```

Return flight:

```
query = "select * from flight where departure_airport=%s and
arrival_airport=%s"\
"and date(arrival_time)=%s"
cur.execute(query, (departure_airport, arrival_airport, return_date))
```

Customer Section

First check using a query if the user already exists:

```
query = 'SELECT * FROM customer WHERE email = %s'
cursor.execute(query, (str(email)))
```

Register:

```
q = 'INSERT INTO customer VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
%s)'
cursor.execute(q, (email, name, password, building_number, street, city, state,
```

View my flights:

```
query = "select * from ticket natural join purchases join flight  where
customer_email = %s and departure_time > %s"

cursor.execute(query, (email, current_date))
```

Search for flights:

```
query = "select * from flight natural join ticket where departure_airport=%s
and arrival_airport=%s and date(departure_time)= %s and ticket_id not in "\
"(select ticket_id from purchases)"
cursor.execute(query,(departure_airport,arrival_airport, dept_date))
```

Purchase Tickets:
Check if ticket is available:

```
query = "select * from ticket natural join flight where flight_num = %s and
airline_name= %s and ticket_id not in (select ticket_id from purchases)"
cursor.execute(query, (int(flight_num), airline_name))
```

Then proceed to buy it if it is:

```
query = "insert into purchases(ticket_id, customer_email, purchase_date) values
(%s,%s,%s)"
cursor.execute(query, (int(ticket_id), customer_email, now))
```

Track my spending:
Select total:

```
"SELECT COAL( SUM(price), 0) as total FROM flight natural join purchases WHERE
purchase_date > %s AND purchase_date < %s AND customer_email = %s"
```

For loop for every month that fall in the range the user inputted:

```
query = "SELECT COAL( SUM(price), 0)  as monthly FROM flight natural join
purchases  WHERE purchase_date > %s and purchase_date < %s AND customer_email =
%s"
```

Logout

Agent Section

First check if the user exists:

```
query = 'SELECT * FROM booking_agent WHERE email = %s'
cursor.execute(query, (str(email)))
```

If yes, then register:

```
q = 'INSERT INTO booking_agent VALUES(%s, %s, %s)'
cursor.execute(q, (email,  password, booking_agent_id))
```

View my commission:
The amount of commission received and the number of tickets sold for a specific range of dates.

```
query = "SELECT IFNULL(SUM(price) , 0) as total_price, IFNULL(COUNT(*) , 0) as ticket_num FROM ticket NATURAL JOIN flight NATURAL JOIN purchases NATURAL JOIN booking_agent WHERE DATE(purchase_date) BETWEEN %s AND %s AND email = %s "
cursor.execute(query, (from_date, to_date, session['email']))
```

The amount of commission received and the number of tickets sold in the past 30 days.

```
query = "SELECT IFNULL(SUM(price) , 0) as total_price, IFNULL(COUNT(*) ,0) as ticket_num FROM ticket NATURAL JOIN flight NATURAL JOIN purchases NATURAL JOIN booking_agent WHERE DATE(purchase_date) BETWEEN NOW() - INTERVAL 30 DAY AND NOW() + INTERVAL 1 DAY AND email = %s"
cursor.execute(query, (session['email']))
```

View Top Customer:

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

```
query = "SELECT customer_email, count(*) as num FROM purchases NATURAL JOIN booking_agent WHERE DATE(purchase_date) BETWEEN NOW() - INTERVAL 6 MONTH AND NOW() + INTERVAL 1 DAY AND email = %s GROUP BY customer_email ORDER BY num DESC LIMIT 5"
cursor.execute(query, (session['email']))
```

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

```
query = "SELECT customer_email, SUM(price) as sum FROM ticket NATURAL JOIN flight NATURAL JOIN purchases NATURAL JOIN booking_agent WHERE DATE(purchase_date) BETWEEN NOW() - INTERVAL 1 YEAR AND NOW() + INTERVAL 1 DAY AND email = %s GROUP BY customer_email ORDER BY sum DESC LIMIT 5"
cursor.execute(query, (session['email']))
```

Staff Section

First check if the user exists:

```
query = 'SELECT * FROM airline_staff WHERE email = %s'
cursor.execute(query, (str(email)))
```

If yes, then register:

```
q = 'INSERT INTO airline_staff VALUES(%s, %s, %s, %s, %s))'
```

```
cursor.execute(q, (email,  password, first_name, last_name, dat_of_birth, airline_name))
```

View my Flights:

See all the current/future/past flights operated by the airline he/she works for based on a range of dates, source/destination airports/city etc.

```
query = "SELECT * FROM flight NATURAL JOIN airplane, airport as A, airport as B where airline_name = %s AND date(departure_time) >= %s AND date(departure_time) <= %s AND flight.departure_airport = A.airport_name and flight.arrival_airport = B.airport_name and (A.airport_name = %s or A.airport_city = %s) and (B.airport_name = %s or B.airport_city = %s)"
cursor.execute(query, (airline_name, start_date, end_date, dept_from, dept_from, arrival_airport, arrival_airport))
```

Showing all the upcoming flights operated by the airline he/she works for the next 30 days.

```
query = 'SELECT * FROM flight WHERE airline_name = %s AND DATE(departure_time) BETWEEN DATE(CURRENT_TIMESTAMP) AND DATE(CURRENT_TIMESTAMP) + INTERVAL 30 DAY'
cursor.execute(query, (airline_name))
```

Create new flights:

```
query = "INSERT INTO flight VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
cursor.execute(query, (airline_name, flight_num, dept_from, dept_time, arrival_airport, arr_time,base_price, flight_status, airplane_id))
```

Change Status of flights:

```
query = "UPDATE flight SET status = %s WHERE (airline_name, flight_num) = (%s, %s)"
cursor.execute(query, (new_status, airline_name, flight_num))
```

Add airplane in the system:

See all the airplanes owned by the airline he/she works for.

```
query = "SELECT airplane_id FROM airplane"
cursor.execute(query)
```

Add new airports:

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

Add new airport in the system:

See all the airports owned by the airline he/she works for.

```
query = "SELECT airport_name FROM airport"
cursor.execute(query)
```

Add new airports:

```
query = "INSERT INTO airport VALUES (%s, %s)"
cursor.execute(query, (name, city))
```

View all the booking agents:

Top 5 booking agents based on number of tickets sales for the past month.

```
title = "Top 5 booking agents by ticket sales for the past month"
query = "SELECT  booking_agent_id, COUNT(*) AS total_sales FROM purchases
WHERE booking_agent_id IS NOT NULL AND DATE(purchase_date) BETWEEN NOW()
INTERVAL 30 DAY AND NOW()GROUP BY booking_agent_id ORDER BY total_sales DESC
LIMIT 5"
cursor.execute(query)
```

Top 5 booking agents based on number of tickets sales for the past year.

```
title = "Top 5 booking agents by ticket sales for the past year"
query = "SELECT  booking_agent_id, COUNT(*) AS total_sales FROM purchases \
WHERE booking_agent_id IS NOT NULL AND DATE(purchase_date) BETWEEN NOW() -
INTERVAL 1 YEAR AND NOW()\
GROUP BY booking_agent_id ORDER BY total_sales DESC LIMIT 5"
cursor.execute(query)
```

Top 5 booking agents based on the amount of commission received for the last year.

```
query = "SELECT booking_agent_id, SUM(price) AS commission FROM purchases
NATURAL JOIN ticket NATURAL JOIN flight\
        WHERE booking_agent_id IS NOT NULL AND DATE(purchase_date) BETWEEN
NOW() - INTERVAL 1 YEAR AND NOW()\
        GROUP BY booking_agent_id ORDER BY commission DESC LIMIT 5"
cursor.execute(query)
```

View frequent customers:

See the most frequent customer within the last year.

```
query = "SELECT customer_email, COUNT(*) AS travel_times FROM purchases
NATURAL JOIN ticket WHERE airline_name = %s AND DATE(purchase_date) BETWEEN
NOW() - INTERVAL 1 YEAR AND NOW() GROUP BY customer_email"

cursor.execute(query, (airline_name))
```

```
query2 = "SELECT customer_email, COUNT(*) AS travel_times FROM purchases
NATURAL JOIN ticketWHERE airline_name = %s GROUP BY customer_email HAVING
travel_times = %s"
cursor.execute(query2, (airline_name, max_times))
```

See travel history of the top customers.

```
query = "SELECT airline_name, flight_num, departure_time, purchase_date,
price, customer_email FROM ticket NATURAL JOIN purchases NATURAL JOIN flight
WHERE customer_email = %s"
cursor.execute(query, (email))
```

View reports:

Total amounts of ticket sold based on range of dates/last year/last month etc. Month wise tickets sold in a bar chart.

last month:

```
query = "SELECT DATE(NOW()) - INTERVAL 1 MONTH AS curr_prev, DATE(NOW()) AS
current,  COUNT(*) AS total_sales FROM purchases NATURAL JOIN ticket WHERE
date(purchase_date) between DATE(NOW()) - INTERVAL 1 MONTH AND DATE(NOW()) and
airline_name = %s"
cursor.execute(query, (airline_name))
```

A range of dates:

```
query = "SELECT COUNT(*) as total_sales FROM purchases NATURAL JOIN ticket
WHERE date(purchase_date) >= %s AND date(purchase_date) <= %s and airline_name
= %s"
cursor.execute(query, (from_date, to_date, airline_name))
```

Last year

```
query = "SELECT DATE(NOW()) AS current, DATE(NOW()) - INTERVAL 1 YEAR AS
curr_prev , COUNT(*) as total_sales FROM purchases NATURAL JOIN ticket WHERE
date(purchase_date) between DATE(NOW()) - INTERVAL 1 YEAR AND DATE(NOW()) and
airline_name = %s"
cursor.execute(query, (airline_name))
```

Comparison of Revenue earned:

Draw a pie chart for showing 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).

last month:

```
query_direct = "SELECT SUM(price) as total_price FROM ticket NATURAL JOIN
purchases NATURAL JOIN flight WHERE booking_agent_id IS NULL and
DATE(purchase_date) BETWEEN DATE(NOW()) - INTERVAL 1 MONTH and DATE(NOW())"
```

```
query_indirect = "SELECT SUM(price) as total_price FROM ticket NATURAL JOIN
purchases NATURAL JOIN flight WHERE booking_agent_id IS NOT NULL and
DATE(purchase_date) BETWEEN DATE(NOW()) - INTERVAL 1 MONTH and DATE(NOW())"
```

last year:

```
query_direct = "SELECT SUM(price) as total_price FROM ticket NATURAL JOIN
purchases NATURAL JOIN flight \
   WHERE booking_agent_id IS NULL and DATE(purchase_date) BETWEEN DATE(NOW())
- INTERVAL 1 YEAR and DATE(NOW())"
query_indirect = "SELECT SUM(price) as total_price FROM ticket NATURAL JOIN
purchases NATURAL JOIN flight \
   WHERE booking_agent_id IS NOT NULL and DATE(purchase_date) BETWEEN
DATE(NOW()) - INTERVAL 1 YEAR and DATE(NOW())"
```

ViewTopdestinations:

Find the top 3 most popular destinations.

Last three month:

```
query = "SELECT arrival_airport, airport_city, count(*) as visit_time FROM
purchases NATURAL JOIN ticket NATURAL JOIN flight as S, airport WHERE
S.arrival_airport  = airport.airport_name AND DATE(purchase_date) BETWEEN
NOW() - INTERVAL 3 MONTH and NOW()GROUP BY arrival_airport ORDER BY visit_time
DESC LIMIT 3"
```

Last Year:

```
query = "SELECT arrival_airport, airport_city, count(*) as visit_time FROM
purchases NATURAL JOIN ticket NATURAL JOIN flight as S, airport WHERE
S.arrival_airport  = airport.airport_name AND DATE(purchase_date) BETWEEN
NOW() - INTERVAL 1 YEAR and NOW()GROUP BY arrival_airport ORDER BY visit_time
DESC LIMIT 3"
```