Functions & Queries in app.py

Yuhan Yao (yy2564) & Xue Bai (xb347)

Note:
Red are names of views.
Black background are use cases/functions.
Cyan are explanations.
Orange are user inputs in the query.

First of all, in the database, we created five views for convenience:

```
CREATE VIEW agent_view_flight AS
   SELECT
      booking_agent.email, purchases.booking_agent_id, purchases.customer_email,
      purchases.purchase_date, purchases.ticket_id, flight.airline_name, flight.flight_num,
      D.airport_city AS departure_city, departure_airport, departure_time,
      A.airport_city AS arrival_city, arrival_airport, arrival_time, price, status, airplane_id
   FROM booking_agent NATURAL RIGHT OUTER JOIN purchases
                  NATURAL JOIN ticket  NATURAL JOIN flight, airport AS D, airport AS A
   WHERE D.airport_name = departure_airport AND A.airport_name = arrival_airport;

CREATE VIEW staff_flight AS
   SELECT airplane_id, flight_num, departure_airport, arrival_airport, departure_time,
                  arrival_time, status
   FROM flight NATURAL JOIN airline_staff;

CREATE VIEW public_search_flight AS
   SELECT flight_num, airline_name, airplane_id, D.airport_city AS departure_city,
                  departure_airport, departure_time, A.airport_city AS arrival_city, arrival_airport,
                  arrival_time, price, status
   FROM airport AS D, flight, airport AS A
   WHERE D.airport_name = departure_airport AND A.airport_name = arrival_airport;

CREATE VIEW agent_commission AS
   SELECT email, purchases.ticket_id, customer_email, purchase_date, price AS ticket_price
   FROM booking_agent NATURAL JOIN purchases NATURAL JOIN ticket
                  NATURAL JOIN flight;

CREATE VIEW customer_spending AS
   SELECT *
   FROM purchases NATURAL JOIN ticket NATURAL JOIN flight;
```

Use cases & queries for public access:

```
@app.route('/')
def publicHome()
```

Render publicHome.html

```
@app.route('/publicSearchFlight', methods=['GET', 'POST'])
def publicSearchFlight()
```

Render search results after clicking the search button (search flight) on the public home page.

```
SELECT airline_name, flight_num, departure_city, departure_airport, departure_time,
        arrival_city, arrival_airport, arrival_time, price, airplane_id
FROM public_search_flight
WHERE departure_airport = if (user_input_departure_airport = '', departure_airport,
        user_input_departure_airport) AND
        arrival_airport = if (user_input_arrival_airport = '', arrival_airport,
        user_input_arrival_airport) AND status = 'upcoming' AND
        departure_city = if (user_input_departure_city = '', departure_city,
        user_input_departure_city) AND
        arrival_city = if (user_input_arrival_city = '',arrival_city, user_input_arrival_city) AND
        date(departure_time) = if (user_input_departure_time = '',date(departure_time),
        user_input_departure_time) AND
        date(arrival_time) = if (user_input_arrival_time = '',date(arrival_time),
        user_input_arrival_time)
ORDER BY airline_name, flight_num
```

Query to select search result. Each user input box can be empty. If all empty, then return all results.

```
@app.route('/publicSearchStatus', methods=['GET', 'POST'])
def publicSearchStatus()
```

Render search results after clicking the second search button (search status) on the public home page.

```
SELECT *
FROM public_search_flight
WHERE flight_num = if (user_input_flight_num = '', flight_num, user_input_flight_num) AND
        date(departure_time) = if (user_input_departure_time = '', date(departure_time),
        user_input_departure_time) AND
        date(arrival_time) = if (user_input_arrival_time = '', date(arrival_time),
        user_input_arrival_time) AND
        airline_name = if (user_input_airline_name = '', airline_name, user_input_airline_name)
ORDER BY airline_name, flight_num
```

Query to select search result. Each user input box can be empty. If all empty, then return all results.

```
@app.route('/logout')
def logout():
```
Clear session and render customer login page: cuslogin.html.

Use cases & queries for customer:
```
@app.route('/cuslogin')
def cuslogin():
```
Render customer login page: cuslogin.html.

```
@app.route('/cusregister')
def cusregister():
```
Render customer registration page: cusregister.html.

```
@app.route('/cusloginAuth', methods=['GET', 'POST'])
def cusloginAuth():
```
Handle customer login details. Check identity. If the user is a customer, then render the customer home page: cushome.html. If the user is not a customer, then show an error message and ask him or her to login again.

SELECT *
FROM customer
WHERE email = user_input_email and password = md5(user_input_password)
Check if given the email and password, there is such a customer in the database system.

SELECT ticket_id, airline_name, airplane_id, flight_num, D.airport_city, departure_airport, A.airport_city, arrival_airport, departure_time, arrival_time, status
FROM flight NATURAL JOIN purchases NATURAL JOIN ticket, airport as D, airport as A
WHERE customer_email = user_input_email and status = 'upcoming' and D.airport_name = departure_airport and A.airport_name = arrival_airport
Select the customer's flights for default view on customer home page.

```
@app.route('/cusregisterAuth', methods=['GET', 'POST'])
def cusregisterAuth():
```
Handle customer registration details. Check identity. If the user is a new customer, then render the customer home page: cushome.html. If the user email is used, then show an error message and ask him or her to register again.

SELECT *
FROM customer
WHERE email = user_input_email
Check that this email has not been registered before.

INSERT INTO customer

```
VALUES(user_input_email, user_input_name, user_input_password,
user_input_building_number, user_input_street, user_input_city, user_input_state,
user_input_phone_number, user_input_passport_number, user_input_passport_expiration,
user_input_passport_country, user_input_date_of_birth)
```
Insert new customer info into customer table.

```
SELECT ticket_id, airline_name, airplane_id, flight_num, D.airport_city, departure_airport,
A.airport_city, arrival_airport, departure_time, arrival_time, status
FROM flight NATURAL JOIN purchases NATURAL JOIN ticket, airport as D, airport as A
WHERE customer_email = user_input_email and status = 'upcoming' and D.airport_name =
departure_airport and A.airport_name = arrival_airport
```
Select the customer's flights for default view on customer home page.

```python
@app.route('/cushome')
def cushome():
```
Render customer home page: cushome.html.

```
SELECT ticket_id, airline_name, airplane_id, flight_num, D.airport_city, departure_airport,
A.airport_city, arrival_airport, departure_time, arrival_time, status
FROM flight NATURAL JOIN purchases NATURAL JOIN ticket, airport as D, airport as A
WHERE customer_email = user_input_email and status = 'upcoming' and D.airport_name =
departure_airport and A.airport_name = arrival_airport
```
Select the customer's flights for default view on customer home page.

```python
@app.route('/cusSearchPurchase')
def cusSearchPurchase():
```
Render cusSearchPurchase.html.

```python
@app.route('/cusSpending', methods=['POST', 'GET'])
def cusSpending()
```
Show the customer spending bar chart.

```
SELECT SUM(price)
FROM customer_spending
WHERE customer_email = user_input_email
        AND (purchase_date BETWEEN DATE_ADD(NOW(),
        INTERVAL - user_input_duration DAY) AND NOW())
```
Select the total spending of the current customer.

```
SELECT YEAR(purchase_date) AS year, MONTH(purchase_date) AS month,
        SUM(price) AS monthly_spending
FROM customer_spending
WHERE customer_email = user_input_email AND purchase_date >= user_input_past_date
GROUP BY YEAR(purchase_date) , MONTH(purchase_date)
```

Select monthly spending information of the customer to draw the bar chart.

```
@app.route('/cusSearchFlight', methods=['GET', 'POST'])
def cusSearchFlight():
```

Show the results of searching flights.

SELECT airline_name, airplane_id, flight_num, D.airport_city, departure_airport, A.airport_city, arrival_airport, departure_time, arrival_time, price, status, num_tickets_left
FROM airport AS D, flight, airport AS A
WHERE D.airport_city = IF (user_input_airport_city = '', D.airport_city, user_input_airport_city)
        AND D.airport_name = departure_airport AND
        departure_airport = IF (user_input_departure_airport = '', departure_airport, user_input_departure_airport) AND
        A.airport_city = IF (user_input_airport_city = '', A.airport_city, user_input_airport_city)
        AND A.airport_name = arrival_airport AND
        arrival_airport = IF (user_input_arrival_airport = '', arrival_airport, user_input_arrival_airport) AND
        DATE(departure_time) = IF (user_input_departure_time = '', date(departure_time), user_input_departure_time) AND
        DATE(arrival_time) = IF (user_input_arrival_time = '', date(arrival_time), user_input_arrival_time)
ORDER BY airline_name, flight_num

Query to select search result. Each user input box can be empty. If all empty, then return all results.

```
@app.route('/cusBuyTickets', methods=['GET', 'POST'])
def cusBuyTickets():
```

Buy tickets.

SELECT *
FROM flight
WHERE airline_name = user_input_airplane_name AND flight_num = user_input_flight_num
        AND num_tickets_left > 0

Check if there are still tickets left for the flight the customer is trying to buy.

SELECT ticket_id
FROM ticket
ORDER BY ticket_id DESC
LIMIT 1

Get the maximum ticket id for now so that we can add 1 to generate a new ticket id.

INSERT INTO ticket VALUES (new_ticket_id, user_input_airline_name, user_input_flight_num)

Create a new ticket.

INSERT INTO purchases
VALUES (new_ticket_id, user_input_customer_email, NULL,
CURDATE())
Create a new purchase tuple.

Use cases & queries for booking agent:

```
@app.route('/agentlogin')
def agentlogin()
```
Render agent login page: agentlogin.html.

```
@app.route('/agentregister')
def agentregister()
```
Render agent registration page: agentregister.html.

```
@app.route('/agentloginAuth', methods=['GET', 'POST'])
def agentloginAuth()
```
Handle agent login details. Check identity. If the user is an agent, then render the agent home page: agenthome.html. If the user is not an agent, then show an error message and ask him or her to login again.

SELECT *
FROM booking_agent
WHERE email = user_input_email and password = md5(user_input_password)
Check if given the email and password, there is such an agent in the database system.

SELECT booking_agent_id
FROM booking_agent
WHERE email = user_input_email
Select the booking agent's booking_agent_id to show in the frontend.

SELECT *
FROM agent_view_flight
WHERE email = user_input_email
Select the agent's flights for default view on booking agent home page.

```
@app.route('/agentregisterAuth', methods=['GET', 'POST'])
def agentregisterAuth()
```
Handle agent registration details. Check identity. If the user is a new agent, then render the agent home page: agenthome.html. If the user email is used, then show an error message and ask him or her to register again.

SELECT *

FROM booking_agent
WHERE email = user_input_email
Check that this email has not been registered before.

INSERT INTO booking_agent
VALUES(user_input_email, user_input_password, user_input_booking_agent_id)
Insert new booking agent info into booking_agent table.

SELECT booking_agent_id
FROM booking_agent
WHERE email = user_input_email
Select the booking agent's booking_agent_id to show in the frontend.

SELECT *
FROM agent_view_flight
WHERE email = user_input_email
Select the agent's flights for default view on booking agent home page.

```
@app.route('/agentHome')
def agentHome()
```
Render agent home page: agenthome.html.

SELECT booking_agent_id
FROM booking_agent
WHERE email = user_input_email
Select the booking agent's booking_agent_id to show in the frontend.

SELECT *
FROM agent_view_flight
WHERE email = user_input_email
Select the agent's flights for default view on booking agent home page.

```
@app.route('/agentSearchPurchase')
def agentSearchPurchase()
```
Render agentSearchPurchase.html.

```
@app.route('/agentCommission', methods=['POST', 'GET'])
def agentCommission()
```
Render agentCommission.html.

SELECT SUM(ticket_price * 0.1), AVG(ticket_price * 0.1), COUNT(ticket_price * 0.1)
FROM agent_commission
WHERE email = user_input_email AND

(purchase_date BETWEEN DATE_ADD(NOW(), INTERVAL - user_input_duration DAY) AND NOW())

Select commission details for the current booking agent where the duration can be a range of dates chosen by the agent.

```
@app.route('/agentTopCustomers')
def agentTopCustomers()
```

Render agentTopCustomers.html and show a bar chart.

```
SELECT customer_email, COUNT(ticket_id)
FROM agent_commission
WHERE email = user_input_email AND DATEDIFF(CURDATE(), DATE(purchase_date)) < 183
GROUP BY customer_email
ORDER BY COUNT(ticket_id) DESC
```

Select how many tickets have each customer bought in the past six months.

```
SELECT customer_email, SUM(ticket_price) * 0.1
FROM agent_commission
WHERE email = user_input_email AND DATEDIFF(CURDATE(), DATE(purchase_date)) < 365
GROUP BY customer_email
ORDER BY SUM(ticket_price) DESC
```

Select how much commission has each customer contributed in the past year.

```
@app.route('/agentSearchFlight', methods=['GET', 'POST'])
def agentSearchFlight()
```

Show the results of searching flights.

```
SELECT booking_agent_id
FROM booking_agent
WHERE email = user_input_email
```

Validate booking agent's identity before he/she can search and purchase any tickets.

```
SELECT airplane_id, flight_num, D.airport_city, departure_airport, A.airport_city, arrival_airport,
        departure_time, arrival_time, status, price, airline_name, num_tickets_left
FROM airport AS D, flight, airport AS A
WHERE D.airport_city = IF (user_input_airport_city = '', D.airport_city, user_input_airport_city)
        AND D.airport_name = departure_airport AND
        departure_airport = IF (user_input_departure_airport = '', departure_airport,
        user_input_departure_airport) AND
        A.airport_city = IF (user_input_airport_city = '', A.airport_city, user_input_airport_city)
        AND A.airport_name = arrival_airport AND
        arrival_airport =  IF (user_input_arrival_airport = '', arrival_airport,
        user_input_arrival_airport) AND
```

DATE(departure_time) = IF (user_input_departure_time = '', date(departure_time), user_input_departure_time) AND
DATE(arrival_time) = IF (user_input_arrival_time = '', date(arrival_time), user_input_arrival_time)
ORDER BY airline_name, flight_num
Query to select search result. Each user input box can be empty. If all empty, then return all results.

```
@app.route('/agentBuyTickets', methods=['GET', 'POST'])
def agentBuyTickets()
```

Buy tickets.

SELECT booking_agent_id
FROM booking_agent
WHERE email = user_input_email
Select the booking agent's booking_agent_id for identity validation and later insertion.

SELECT *
FROM customer
WHERE email = user_input_email
Validate customer email.

SELECT *
FROM flight
WHERE airline_name = user_input_airplane_name AND flight_num = user_input_flight_num
AND num_tickets_left > 0
Check if there are still tickets left for the flight the booking agent is trying to buy.

SELECT ticket_id
FROM ticket
ORDER BY ticket_id DESC
LIMIT 1
Get the maximum ticket id for now so that we can add 1 to generate a new ticket id.

INSERT INTO ticket VALUES (new_ticket_id, user_input_airline_name, user_input_flight_num)
Create a new ticket.

INSERT INTO purchases
VALUES (new_ticket_id, user_input_customer_email, user_input_booking_agent_id, CURDATE())
Create a new purchase tuple.

Use cases & queries for airline staff:

```
@app.route('/stafflogin')
def stafflogin():
```

Render staff login page: stafflogin.html.

```
@app.route('/staffregister')
def staffregister():
```

Render staff registration page: staffregister.html.

```
@app.route('/staffloginAuth', methods=['GET', 'POST'])
def staffloginAuth():
```

Handle staff login details. Check identity. If the user is a staff, then render the staff home page: staffhome.html. If the user is not a staff, then show an error message and ask him or her to login again.

SELECT *
FROM airline_staff
WHERE username = user_input_username and password = md5(user_input_password)
Check if given the email and password, there is such a staff in the database system.

SELECT username, airline_name, airplane_id, flight_num, departure_airport, arrival_airport, departure_time, arrival_time
FROM flight NATURAL JOIN airline_staff
WHERE username = user_input_username and status = 'upcoming' and datediff(CURDATE(), DATE(departure_time)) < 30
Select the airline's upcoming flights in 30 days for default view on staff home page.

```
@app.route('/staffregisterAuth', methods=['GET', 'POST'])
def staffregisterAuth():
```

Handle staff registration details. Check identity. If the user is a new staff, then render the staff home page: staffhome.html. If the username is used, then show an error message and ask him or her to register again.

SELECT *
FROM airline_staff
WHERE username = user_input_username
Check that this username has not been registered before.

SELECT airline_name
FROM airline
WHERE airline_name = user_input_airline_name
Check that the airline exists.

INSERT INTO airline_staff

VALUES(user_input_username, user_input_password, user_input_first_name, user_input_last_name, user_input_date_of_birth, user_input_airline_name)
Insert new staff info staff table if the airline exists.

SELECT username, airline_name, airplane_id, flight_num, departure_airport, arrival_airport, departure_time, arrival_time
FROM flight NATURAL JOIN airline_staff
WHERE username = user_input_username and status = 'upcoming' and datediff(CURDATE(), DATE(departure_time)) < 30
Select the airline's upcoming flights in 30 days for default view on staff home page.

```
@app.route('/staffhome')
def staffhome():
```

Render staff home page: staffhome.html.

SELECT username, airline_name, airplane_id, flight_num, departure_airport, arrival_airport, departure_time, arrival_time
FROM flight NATURAL JOIN airline_staff
WHERE username = user_input_username and status = 'upcoming' and datediff(CURDATE(), DATE(departure_time)) < 30
Select the airline's upcoming flights in 30 days for default view on staff home page.

```
@app.route('/staffflight')
def staffflight():
```

Render staffflight.html.

SELECT username, airline_name
FROM airline_staff
WHERE username = user_input_username
Validate staff's identity before he/she can search flights and edit flight status.

```
@app.route('/staffSearchFlight', methods=['GET', 'POST'])
def staffSearchFlight():
```

Show the results of searching flights.

SELECT airline_name, airplane_id, flight_num, D.airport_city, departure_airport, A.airport_city, arrival_airport, departure_time, arrival_time, status, price
FROM airport AS D, flight NATURAL JOIN airline_staff, airport AS A
WHERE D.airport_city = IF (user_input_airport_city = '', D.airport_city, user_input_airport_city)
        AND D.airport_name = departure_airport AND
        departure_airport = IF (user_input_departure_airport = '', departure_airport, user_input_departure_airport) AND
        A.airport_city = IF (user_input_airport_city = '', A.airport_city, user_input_airport_city)
        AND A.airport_name = arrival_airport AND

arrival_airport = IF (user_input_arrival_airport = '', arrival_airport, user_input_arrival_airport) AND
DATE(departure_time) = IF (user_input_departure_time = '', date(departure_time), user_input_departure_time) AND
DATE(arrival_time) = IF (user_input_arrival_time = '', date(arrival_time), user_input_arrival_time) AND
username = user_input_username
ORDER BY airline_name, flight_num
Query to select search result. Each user input box can be empty. If all empty, then return all results.

```
@app.route('/edit_status', methods=['GET', 'POST'])
def edit_status():
```
Edit flight status.

UPDATE flight
SET status = user_input_status
WHERE flight_num = user_input_flight_num
Change the status of selected flight.

```
@app.route('/staffaddinfo')
def staffaddinfo():
```
Render staffaddinfo.html

SELECT airplane_id, seats
FROM airplane NATURAL JOIN airline_staff
WHERE username = user_input_username
Select the airline id and seats for default view on staff addinfo page.

```
@app.route('/create_flight', methods=['GET', 'POST'])
def create_flight():
```
Add flight information.

SELECT airline_name
FROM airline_staff
WHERE username = user_input_username
Select the airline of the staff.

SELECT airport_name
FROM airport
WHERE airport_name = user_input_airport
Check that the airport input by staff exists, error if it doesn't.

SELECT airplane_id

FROM airplane
WHERE airline_name = airline_name and airplane_id =user_input_airplane_id
Check that the airplane input by staff exists in the airline, error if it doesn't.

SELECT seats
FROM airplane
WHERE airline_name = airline_name and airplane_id =user_input_airplane_id
Check that the airplane has enough seats for tickets input by staff, error if it doesn't.

SELECT flight_num
FROM flight
WHERE airline_name = airline_name and flight_num =user_input_flight_num
Check that the flight input by staff does not exist in the airline, error if it does.

INSERT INTO flight
VALUES (airline_name, user_input_flight_num, user_input_departure_airport, user_input_departure_date, user_input_departure_time, user_input_arrival_airport, user_input_arrival_date, user_input_arrival_time, user_input_price, user_input_status, user_input_airplane_id, user_input_number)
Insert new flight info into flight table.

```
@app.route('/add_airplane', methods=['GET', 'POST'])
def add_airplane():
```

Add airplane information.

SELECT airline_name
FROM airline_staff
WHERE username = user_input_username
Select the airline of the staff.

SELECT airplane_id
FROM airplane
WHERE airline_name = airline_name and airplane_id =user_input_airplane_id
Check that the airplane input by staff does not exist in the airline, error if it does.

INSERT INTO airplane
VALUES (airline_name, user_input_airplane_id, user_input_seats)
Insert new airplane info into airplane table.

```
@app.route('/add_airport', methods=['GET', 'POST'])
def add_airport():
```

Add airport information.

SELECT airline_name

FROM airline_staff
WHERE username = user_input_username
Select the airline of the staff.

SELECT airort_name
FROM airport
WHERE airport_name = user_input_airport_name
Check that the airport input by staff does not exist in the airline, error if it does.

INSERT INTO airport
VALUES (user_input_airport_name, user_input_airport_city)
Insert new airport info into airport table.

```
@app.route('/staffagent')
def staffagent():
```

Render staffagent.html and show top agents

SELECT email, booking_agent_id, sum(price) * 0.1 as commission
FROM booking_agent NATURAL JOIN purchases NATURAL JOIN flight NATURAL JOIN ticket AS T, airline_staff
WHERE username = user_input_username and airline_staff.airline_name = T.airline_name and datediff(CURDATE(), DATE(purchase_date)) < 365
GROUP BY email, booking_agent_id
ORDER BY commission DESC
LIMIT 5
Select top 5 agents in the airline based on commission in the past year.

SELECT booking_agent.email, booking_agent_id, count(ticket_id) as ticket FROM booking_agent NATURAL JOIN purchases NATURAL JOIN ticket AS T, airline_staff
WHERE username = user_input_username and airline_staff.airline_name = T.airline_name and datediff(CURDATE(), DATE(purchase_date)) < 30
GROUP BY email, booking_agent_id
ORDER BY ticket DESC
LIMIT 5
Select top 5 agents in the airline based on tickets sold in the past month.

SELECT booking_agent.email, booking_agent_id, count(ticket_id) as ticket FROM booking_agent NATURAL JOIN purchases NATURAL JOIN ticket AS T, airline_staff
WHERE username = user_input_username and airline_staff.airline_name = T.airline_name and datediff(CURDATE(), DATE(purchase_date)) < 365
GROUP BY email, booking_agent_id
ORDER BY ticket DESC
LIMIT 5
Select top 5 agents in the airline based on tickets sold in the past year.

```
@app.route('/staffcus')
def staffcus():
```
Render staffcus.html and show customer info.

SELECT email, name, count(ticket_id) as ticket FROM customer, purchases NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline_staff
WHERE email = customer_email AND username = user_input_username and datediff(CURDATE(), DATE(purchase_date)) < 365
GROUP BY email, name
ORDER BY ticket DESC
LIMIT 1
Select the top customer in the airline based on tickets sold in the past year.

```
@app.route('/staffcusflight', methods=['GET', 'POST'])
def staffcusflight():
```
Show the results of flights bought by the customer.

SELECT DISTINCT airplane_id, flight_num, departure_airport, arrival_airport, departure_time, arrival_time, status
FROM customer, purchases NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline_staff
WHERE email = user_input_email and email = customer_email and username = user_input_username
Select the flights in the airline bought by the customer.

SELECT email
FROM customer
WHERE email = user_input_email
Check that the customer input by staff exists, error if it doesn't.

```
@app.route('/staffflightcus', methods=['GET', 'POST'])
def staffflightcus():
```
Show the results of customers on particular flight

SELECT DISTINCT email, name
FROM customer, purchases NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline_staff
WHERE flight_num = user_input_flight_num and email = customer_email and username = user_input_username
Select the customer on the flight input by staff in the airline.

SELECT flight_num FROM flight NATURAL JOIN airline_staff
WHERE flight_num = user_input_flight_num AND username = user_input_username

Check that the flight input by staff exists, error if it doesn't.

```
@app.route('/staffDest')
def staffDest():
```

Render staffDest.html and show top destinations.

SELECT airport_city, count(ticket_id) AS ticket
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight, airport
WHERE airport_name = arrival_airport and datediff(CURDATE(), DATE(purchase_date)) < 90
GROUP BY airport_city
ORDER BY ticket DESC
LIMIT 3
Select top 3 destinations in the past three months.

SELECT airport_city, count(ticket_id) AS ticket
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight, airport
WHERE airport_name = arrival_airport and datediff(CURDATE(), DATE(purchase_date)) < 365
GROUP BY airport_city
ORDER BY ticket DESC
LIMIT 3
Select top 3 destinations in the past year.

```
@app.route('/staffReve')
def staffReve():
```

Render staffDest.html and show pie charts.

SELECT sum(price)
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline_staff
WHERE username = user_input_username AND booking_agent_id is NULL AND
datediff(CURDATE(), DATE(purchase_date)) < 30
GROUP BY airline_name
Select sum price bought by all customers in the airline in the past month.

SELECT sum(price)
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline_staff
WHERE username = user_input_username AND booking_agent_id is NOT NULL AND
datediff(CURDATE(), DATE(purchase_date)) < 30
GROUP BY airline_name
Select sum price bought by all agents in the airline in the past month.

SELECT sum(price)
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline_staff
WHERE username = user_input_username AND booking_agent_id is NULL AND
datediff(CURDATE(), DATE(purchase_date)) < 365

GROUP BY airline_name
Select sum price bought by all customers in the airline in the past year.

SELECT sum(price)
FROM purchases NATURAL JOIN ticket NATURAL JOIN flight NATURAL JOIN airline_staff
WHERE username = user_input_username AND booking_agent_id is NOT NULL AND datediff(CURDATE(), DATE(purchase_date)) < 365
GROUP BY airline_name
Select sum price bought by all agents in the airline in the past year.

```
@app.route('/staffTickets')
def staffTickets():
```

Render staffTickets.html.

```
@app.route('/stafffixticket', methods=['GET', 'POST'])
def stafffixticket():
@app.route('/staffticket', methods=['GET', 'POST'])
def staffticket():
```

Show the results of searching the sum of tickets and monthly tickets in the bar chart.

SELECT YEAR(purchase_date) AS year, MONTH(purchase_date) AS month, count(ticket_id)
FROM purchases NATURAL JOIN airline_staff NATURAL JOIN flight NATURAL JOIN ticket
WHERE datediff(CURDATE(), DATE(purchase_date)) < 30 AND username = user_input_username
GROUP BY year, month
ORDER BY year, month
Select the sum of tickets in the airline in the past month.

SELECT YEAR(purchase_date) AS year, MONTH(purchase_date) AS month, count(ticket_id)
FROM purchases NATURAL JOIN airline_staff NATURAL JOIN flight NATURAL JOIN ticket
WHERE datediff(CURDATE(), DATE(purchase_date)) < 365 AND username = user_input_username
GROUP BY year, month
ORDER BY year, month
Select the sum of tickets in the airline in the past year.

SELECT YEAR(purchase_date) AS year, MONTH(purchase_date) AS month, count(ticket_id)
FROM purchases NATURAL JOIN airline_staff NATURAL JOIN flight NATURAL JOIN ticket
WHERE purchase_date > user_input_date and purchase_date < user_input_date AND username = user_input_username
GROUP BY year, month
ORDER BY year, month
Select the sum of tickets in the airline in the range of date input by staff.