Non Logged in users:

Use Case	Implementation
View Public Info	Getting flight information based on arrival/departure airport: "SELECT * FROM `flight` WHERE departure_airport = \"" + request.form['depAirport'] + "\" AND arrival_airport = \"" + request.form['arrAirport'] + "\"AND departure_date = \"" + \ request.form['depDate'] + "\"" Getting flight information based on airline name and flight number: "SELECT * FROM `flight` WHERE airline_name = \"" + request.form['airlineName'] + "\" AND flight_number = \"" + request.form['flightNumber'] + "\"AND departure_date = \"" + request.form['depDate'] + "\"" + "AND arrival_date = \"" + request.form['arrDate'] + "\"" Getting flight information based on the airport names: "Select * from flight where departure_date = \"" + request.form['depDate'] + "\" and (departure_airport = \"" + dep.get('airport_name') + "\" and arrival_airport = \"" + arr.get('airport_name') + "\")
Register	Query for customer registration: (storing password as hex representation of hashed values) query = f'''INSERT INTO customer VALUES (\'{request.form['name']}\', \'{request.form['email']}\', \'{hex_hashed}\', {request.form['buildingNumber']}, \'{request.form['street']}\', \'{request.form['city']}\', \'{request.form['state']}\', {request.form['phoneNumber']}, {request.form['passportNumber']}, \'{request.form['expDate']}\', \'{request.form['passportCountry']}\', \'{request.form['dateOfBirth']}\')''' Query for agent registration: f'''INSERT INTO BookingAgent VALUES (\'{request.form['email']}\', \'{hex_hashed}\', {agentID}, 0)'''

	Query for staff registration: f'''INSERT INTO staff VALUES (\'{request.form['email']}\', \'{hex_hashed}\', \'{request.form['name']}\', \'{request.form['dateOfBirth']}\', \'{numbers[0]}\', \'{request.form['airlineName']}\')'''
Login	Query for customer login: "SELECT * FROM customer WHERE customer_email = \"" + request.form['username'] + "\" AND password = \"" + hex_hashed + "\"" Query for staff login: "SELECT * FROM staff WHERE username = \"" + request.form['username'] + "\" AND password = \"" + hex_hashed + "\"" Query for agent login: "SELECT * FROM BookingAgent WHERE booking_agent_email = \"" + request.form['username'] + "\" AND password = \"" + hex_hashed + "\""

Customer:

Use Case	Implementation
View My Flights	Find their purchased ticket_id(s): "Select ticket_id from purchases where customer_email = \"" + session['username'] + "\"" Find the corresponding flight numbers: "Select flight_number from ticket where ticket_id = {str(item.get(‘ticket_id’))}" Find the flight information: query = "Select * from flight where (CURRENT_DATE < flight.departure_date OR (CURRENT_DATE = flight.departure_date " \"AND CURRENT_TIME < departure_time)) and (flight_number = {str(item.get(‘flight_number’))}"
Search for Flights	same queries as non logged in users at the top of the page

Purchase Tickets	Find the airplane id: f'''select airplane_id from flight where flight_number = {flight_number}and departure_date = \'{depDate}\' and departure_time = \'{depTime}\'''' Find the total number of seats: f'''select num_seats from airplane where airplane_id = {airplane_id}''' Find the number of tickets bought (to calculate if the base price needs to be increased): f'''select count(*) from ticket where flight_number = {flight_number}''' Insert new purchase into purchases: f'''insert into purchases values ({ticket_id}, \'{request.form['email']}\', null, {base_price}, {date}, {time}, \'{request.form['cardType']}\', {request.form['cardNumber']}, \'{request.form['cardName']}\', \'{request.form['expDate']}\')''' Insert new ticket into ticket: f'''insert into ticket values ({ticket_id}, \'{airline_name}\', {flight_number})'''
Rate & Comment	Check if already rated / commented: f'''select customer_email, flight_number from rates where customer_email = \'{customer_email}\' and flight_number = {flightNumber}''' Insert new rating / comment: f'''insert into rates values(\"{customer_email}\", {flightNumber}, \"{comment}\", {rating})'''
Track my Spending	Get spending from last year: f'''select sold_price from purchases where customer_email = \'{session['username']}\'and purchase_date > \'{old_date}\'''' Get monthly spending: query = f'''select sold_price from purchases where customer_email = \'{session['username']}\' and month(purchase_date) = {x} and year(purchase_date) = {current_year}'''

	Getting spending from a specific time range: f''' select sold_price from purchases where customer_email = \'{session['username']}\' and purchase_date > \'{date1}\' and purchase_date < \'{date2}\''''
Logout	no query, the session is destroyed and the user is redirected back to the login page

Booking Agents:

Use Case	Implementation

View My Flights	Get all the ticket ids where the booking agent help buy:
	"Select ticket_id from purchases where booking_agent_id = " + bookingID
	
	Obtain the flight_numbers from the ticket_ids with a for loop:
	query = "Select flight_number from ticket where ticket_id = "
	for item in ticket_ids:
	        query += str(item.get('ticket_id'))
	        query += " or ticket_id = "
	query += " -1 "
	
	Select all the details from the flight_numbers that are in the future with for loop:
	query = "Select * from flight where (CURRENT_DATE < flight.departure_date OR (CURRENT_DATE = flight.departure_date AND CURRENT_TIME < departure_time)) and (flight_number = "
	for item in flight_numbers:
	        query += str(item.get('flight_number'))
	        query += " or flight_number = "
	query += " -1) "

Search for Flights	Same as view public as not logged in users
Purchase Tickets	insert into purchases: f'''insert into purchases values ({ticket_id}, \'{request.form['email']}\',

	{session['agentID']}, {base_price}, {date}, {time}, \'{request.form['cardType']}\', {request.form['cardNumber']}, \'{request.form['cardName']}\', \'{request.form['expDate']}\')''' insert into ticket: query = f'''insert into ticket values ({ticket_id}, \'{airline_name}\', {flight_number})''' get commission: f'''select commission from bookingagent where booking_agent_id = {session['agentID']}''' update commission: f'''update bookingagent set commission = {commission} where booking_agent_email = \'{session['username']}\'''''

View my Commission	This gets the sum of commissions for this booking agent for the last 30 days:
	query = "SELECT sum(`sold_price`)/10 from purchases where (purchase_date > ADDDATE(CURRENT_DATE, INTERVAL -30 DAY)) and booking_agent_id =" + bookingID
	cursor.execute(query)
	commission = cursor.fetchall()[0].get("sum(`sold_price`)/10")
	
	This gets the amount of tickets for this booking agent for the last 30 days:
	query = "SELECT COUNT(*) from purchases WHERE (purchase_date > ADDDATE(CURRENT_DATE, INTERVAL -30 DAY)) and `booking_agent_id` =" + bookingID
	 cursor.execute(query)
	 tickets = cursor.fetchall()[0].get("COUNT(*)")
	
	If they search for a date range we modify both sum of commission and amount of tickets to accommodate that date range:
	query = "SELECT sum(`sold_price`)/10 from purchases where ((purchase_date > \'" + request.form['begDate'] + "\') and ("purchase_date < \'" + request.form['endDate']
	query += "\')) and booking_agent_id =" + bookingID

	query = "SELECT COUNT(*) from purchases WHERE ((purchase_date > \'" + request.form['begDate'] + "\') and ("purchase_date < \'" + request.form['endDate'] query += "\')) and booking_agent_id =" + bookingID

View Top Customers	We query by last 6 months and booking agent id then group it by customer_email then put it in order of the amount of purchases/tickets to get the top 5:
	query = "SELECT `customer_email`, count(*) FROM purchases WHERE purchase_date > ADDDATE(CURRENT_DATE, INTERVAL -6 MONTH) and booking_agent_id = " + bookingID + " GROUP BY `customer_email` ORDER BY COUNT(*) DESC LIMIT 5 "
	
	We query by last year and booking agent id then group it by customer_email then put it in order of the amount of commissions to get the top 5:
	query = "SELECT `customer_email`, sum(sold_price)/10 FROM purchases WHERE purchase_date > ADDDATE(CURRENT_DATE, INTERVAL -1 YEAR) and booking_agent_id = " + bookingID + " GROUP BY `customer_email` ORDER BY sum(sold_price)/10 DESC LIMIT 5"

Logout	Ends session

Airline Staff:

Use Case	Implementation
View Flights	Showing flights in the next 30 days: f"Select flight_number from flight where airline_name = \'{airline_name}\' and ((CURRENT_DATE < " f"flight.departure_date) OR (CURRENT_DATE = flight.departure_date AND CURRENT_TIME < departure_time)) and" f"(flight.departure_date < ADDDATE(CURRENT_DATE, INTERVAL 30 DAY))" Get the names of customers on each flight:

	f'''SELECT customer.name from ticket NATURAL JOIN purchases NATURAL JOIN customer where ticket.flight_number = {num}'''
Create new Flights	Insert a new flight: f'''INSERT into flight values (\'{ session['airline_name'] }\', \'{status}\', \'{request.form['flightNumber']}\', \'{request.form['depAirport']}\', \'{request.form['depDate']}\', \'{request.form['depTime']}\', \'{request.form['arrAirport']}\', \'{request.form['arrDate']}\', \'{request.form['arrTime']}\', \'{request.form['basePrice']}\',\'{request.form['airplaneID']}\')'''
Change status of flights	Change status: query = f'''update flight set status = \'{status}\' where flight_number = \'{request.form['flightNumber']}\' and departure_date = \'{request.form['depDate']}\' and departure_time = \'{request.form['depTime']}\''''
Add new airplane into system	Insert a new airplane: query = f'''INSERT into airplane values (\'{request.form['airplaneID']}\', \'{request.form['numSeats']}\', \'{session['airline_name']}\')'''
Add new airport into system	Insert a new airport: query = f'''INSERT into airport values (\'{request.form['airportName']}\', \'{request.form['city']}\')'''
View Flight Ratings	Get flights from this airline: "SELECT * from flight where airline_name = \"" + session["airline_name"] + "\"" Get average ratings: "SELECT AVG(`rating`) FROM `rates` WHERE flight_number =" + flight_number Get actual rating info: "SELECT customer_email, comment, rating FROM `rates` WHERE flight_number =" + flight_number
View Booking Agents	Top Booking Agents for the past month: "SELECT booking_agent_id, count(*) from purchases natural join ticket where booking_agent_id IS NOT NULL and (purchase_date > ADDDATE(CURRENT_DATE, INTERVAL -1 MONTH)) and ticket.airline_name = \"" + session["airline_name"] + "\" group by booking_agent_id order by count(*) desc limit 5"

	Past Year: "SELECT booking_agent_id, count(*) from purchases natural join ticket where booking_agent_id IS NOT NULL and (purchase_date > ADDDATE(CURRENT_DATE, INTERVAL -1 YEAR)) and ticket.airline_name = \"" + session["airline_name"] + "\" group by booking_agent_id order by count(*) desc limit 5" With Commission: "SELECT booking_agent_id, sum(sold_price)/10 from purchases natural join ticket where booking_agent_id IS NOT NULL and (purchase_date > ADDDATE(CURRENT_DATE, INTERVAL -1 YEAR)) and ticket.airline_name = \"" + session["airline_name"] + "\" group by booking_agent_id order by sum(sold_price)/10 desc limit 5"
View frequent customers	Get top 5 customers and number of tickets bought in the past year: f'''SELECT customer_email, count(ticket_id) FROM purchases NATURAL JOIN ticket WHERE airline_name = \'{session['airline_name']}\' AND purchase_date >= \'{date.strftime("%Y-%m-%d")}\' group by customer_email order by count(ticket_id) desc limit 5''' Getting the flight numbers for each customer: f'''select flight_number from ticket natural join purchases where airline_name = \'{session['airline_name']}\' and customer_email = \'{email}\''''
View Reports	Number of tickets sold, monthly: f'''SELECT count(ticket_id) FROM ticket NATURAL JOIN purchases WHERE ticket.airline_name = \'{session['airline_name']}\' and extract(month from purchases.purchase_date) = {month_num} AND extract(year from purchases.purchase_date) = {year_num}'''
Comparison of revenue earned	Direct revenue from last month: f'''SELECT sum(sold_price) FROM ticket NATURAL JOIN purchases WHERE ticket.airline_name = \'{airline_name}\' AND purchases.booking_agent_id is null and purchases.purchase_date >= \'{one_month_ago.strftime("%Y-%m-%d")}\''''

	Indirect revenue from last month: f'''SELECT sum(sold_price) FROM ticket NATURAL JOIN purchases WHERE ticket.airline_name = \'{airline_name}\' AND purchases.booking_agent_id is not null and purchases.purchase_date >= \'{one_month_ago.strftime("%Y-%m-%d")}\'''' Direct revenue from last year: f'''SELECT sum(sold_price) FROM ticket NATURAL JOIN purchases WHERE ticket.airline_name = \'{airline_name}\' AND purchases.booking_agent_id is null and purchases.purchase_date >= \'{one_year_ago.strftime("%Y-%m-%d")}\'''' Indirect revenue from last year: f'''SELECT sum(sold_price) FROM ticket NATURAL JOIN purchases WHERE ticket.airline_name = \'{airline_name}\' AND purchases.booking_agent_id is not null and purchases.purchase_date >= \'{one_year_ago.strftime("%Y-%m-%d")}\''''
View Top Destinations	List of all destinations from the past 3 months: f'''SELECT DISTINCT airport.city FROM purchases NATURAL JOIN ticket NATURAL JOIN flight, airport WHERE flight.arrival_airport = airport.airport_name and flight.airline_name = \'{session['airline_name']}\' and purchase_date >= \'{three_months_ago.strftime("%Y-%m-%d")}\'''' Get airport name: f'''select airport_name from airport where city = \'{city}\'''' Get number of tickets bought for each destination: f'''SELECT count(ticket_id) FROM ticket NATURAL JOIN flight WHERE airline_name = \'{session['airline_name']}\' and arrival_airport = \'{airports[i]}\''''
Logout	No query, destroy the session and redirect to login page