Non Logged in users:

|  |  |
| --- | --- |
| Use Case | Implementation |
| View Public Info | Getting flight information based on arrival/departure airport and departure date:  "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 and departure/arrival dates:  "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 and departure date:  "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 (querying with 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 (querying with password as hex representation of hashed values):  f'''INSERT INTO BookingAgent VALUES (\'{request.form['email']}\', \'{hex\_hashed}\', {agentID}, 0)'''  Query for staff registration (querying with password as hex representation of hashed values):  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 (if anything is returned, the login was successful):  "SELECT \* FROM customer WHERE customer\_email = \"" + request.form['username'] + "\" AND password = \"" + hex\_hashed + "\""  Query for staff login (if anything is returned, the login was successful):  "SELECT \* FROM staff WHERE username = \"" + request.form['username'] + "\" AND password = \"" + hex\_hashed + "\""  Query for agent login (if anything is returned, the login was successful):  "SELECT \* FROM BookingAgent WHERE booking\_agent\_email = \"" + request.form['username'] + "\" AND password = \"" + hex\_hashed + "\"" |

Customer:

|  |  |
| --- | --- |
| Use Case | Implementation |
| View My Flights | Find ticket\_id(s) that was were purchased by the user with the current session:  "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 for future purchased flights:  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 given the flight number and departure date/time:  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 given an airplane id:  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 | Get all flights that this customer has purchased tickets for that departed already:  Check if already rated / commented (if anything is returned, then the customer 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, where old date is exactly 1 year ago:  f'''select sold\_price from purchases where customer\_email = \'{session['username']}\'and purchase\_date > \'{old\_date}\''''  Get monthly spending (this is ran in a loop, where x is the index of the loop from 1-12):  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 (date1 – date2):  f''' select sold\_price from purchases where customer\_email = \'{session['username']}\' and purchase\_date > \'{date1}\' and purchase\_date < \'{date2}\''''  Get monthly spending (this is ran in a loop as well.) Also make sure that the purchase dates are between the specified range of dates:  f'''select sold\_price from purchases where customer\_email = \'{session['username']}\' and month(purchase\_date) = {month\_num} and year(purchase\_date) = {year\_num} and  purchase\_date <= \'{datetime2.strftime("%Y-%m-%d")}\' and purchase\_date >= \'{datetime1.strftime("%Y-%m-%d")}\'''' |
| 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 helped 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 for this booking agent in the session:  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 for this staff member’s airline:  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 the 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 for a specific flight:  "SELECT AVG(`rating`) FROM `rates` WHERE flight\_number =" + flight\_number  Get actual rating info for a specific flight:  "SELECT customer\_email, comment, rating FROM `rates` WHERE flight\_number =" + flight\_number |
| View Booking Agents | Get the IDs of the top 5 booking agents sorted by the number of purchases which they made on behalf of a customer from 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"  Get the IDs of the top 5 booking agents sorted by the number of purchases which they made on behalf of a customer from the 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"  Get the IDs and commission of the top 5 booking agents from the past year  "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 flight which the customer bought a ticket for:  f'''select flight\_number from ticket natural join purchases  where airline\_name = \'{session['airline\_name']}\' and customer\_email = \'{email}\'''' |
| View Reports | Get the number of tickets sold in a particular month and year (this is ran in a loop across the range of dates) Also make sure that the purchase date is in the specified range of dates:  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} and purchase\_date <= \'{datetime2}\' and purchase\_date >= \'{datetime1}\'''' |
| Comparison of revenue earned | Get the direct revenue from last month for a specific airline:  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")}\''''  Get the indirect revenue from last month for a specific airline:  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")}\''''  Get the direct revenue from last year for a specific airline:  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")}\''''  Get the indirect revenue from last year for a specific airline:  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 |