All use cases:

- General
    - Login
        - Customer
            - Fetch the hashed data from the database and judge whether it matches with input password
                - query = 'SELECT password FROM customer WHERE email = %s'
            - Display Invalid login name if the user does not exist; Display error message Wrong password if the password does not match
        - Staff
            - Fetch the hashed data from the database and judge whether it matches with input password
                - query = 'SELECT * FROM airline_staff WHERE username = %s'
            - Display Invalid login name if the user does not exist; Display error message Wrong password if the password does not match
            - Set session[“status”] to a list of all permission types:
                - "SELECT permission_type from permission where permission.username = %s"
            - Set session[“company”] to the airline the staff works for:
                - query_company = "select airline_name from airline_staff where airline_staff.username = %s "
        - Agent
            - Fetch the hashed data from the database and judge whether it matches with input password
                - query = 'SELECT * FROM booking_agent WHERE email = %s '
            - Display Invalid login name if the user does not exist; Display error message Wrong password if the password does not match
            - Set session[“company”] to a list of all airlines the agent works for:
                - query = 'SELECT airline_name from booking_agent_work_for WHERE email = %s'
    - Register
        - Customer
            - Check whether the email already exists; Display error message “The customer already exists”; Display “password does not match” if the two input passwords do not correct
                - query = 'SELECT * FROM customer WHERE email = %s'
            - If all input information is correct, insert the results into database:
                - ins = 'INSERT INTO customer VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)'

- Staff
    - Fetch all the airline names and send them back to staff_register.html for people to choose from:
        - query = "select airline_name from airline"
    - Check whether the email already exists; Display error message “The customer already exists”; Display “password does not match” if the two input passwords do not correct
        - query = 'SELECT * FROM airline_staff WHERE username = %s'
    - If all input information is correct, insert the results into database:
        - ins1 = 'INSERT INTO airline_staff VALUES(%s, %s, %s, %s, %s, %s)'
        - ins2 = 'INSERT INTO permission VALUES(%s, %s)'
- Agent
    - Check whether the email already exists; Display error message “The customer already exists”; Display “password does not match” if the two input passwords do not correct:
        - query_email = 'SELECT * FROM booking_agent WHERE email = %s'
    - Select the max_id from the database and add one to it, return the assigned booking_agent_id to the user:
        - query = "select max(booking_agent_id) from booking_agent"
    - If all input information is correct, insert the results into database:
        - ins1 = 'INSERT INTO booking_agent VALUES(%s, %s, %s)'

- View Upcoming flights
    - Default show all upcoming flights (people who have not log in):
        - query = "SELECT * FROM flight where status = 'Upcoming'"
    - Default search function; Add corresponding information to the first query to filter the results
        - add = "and departure_airport in (select airport_name from airport where airport_city ='"+ d_city +"')"
        - add = "and arrival_airport in (select airport_name from airport where airport_city = '"+ a_city +"')"
    - Customer purchase (login as a customer)
        - Check whether there is available seats for a particular flight
            - query = """select * from purchases, ticket where purchases.customer_email = %s and purchases.ticket_id = ticket.ticket_id and ticket.flight_num = %s and ticket.airline_name = %s"""
            - query = """select seats from airplane where airline_name = %s and airplane_id in (select airplane_id from flight where airline_name = %s and flight_num = %s)"""

        - query = """SELECT count(*) FROM ticket WHERE airline_name = %s AND flight_num = %s"""
    - If there are available seats, buy for the customer. Automatically return the ticket_id to the customer by adding 1 to the current max ticket_id.
        - query = "select max(ticket_id) from purchases"
        - query1 = "insert into ticket values(%s, %s, %s)" cursor.execute(query1,(ticket_id,airline_name,flight_num)) query2 = "INSERT INTO purchases (ticket_id,customer_email,purchase_date) VALUES(%s,%s,%s)"
- Agent purchase (login as a agent)
    - Check whether the agent's input is a valid customer email input:
        - query = """select email from customer WHERE email = %s"""
    - Acquire the booking agent id:
        - query = """select booking_agent_id from booking_agent WHERE booking_agent.email = %s"""
    - Check whether the customer has already bought the ticket:
        - query = """select * from purchases, ticket where purchases.ticket_id = ticket.ticket_id and ticket.flight_num = %s AND customer_email = %s"""
    - Check how many seats have been purchased on the flight:
        - query = """select seats from airplane where airline_name = %s and airplane_id in (select airplane_id from flight where airline_name = %s and flight_num = %s)"""
    - Check how many seats that the plane has originally, (then compared with the seats from above, so decide whether there is enough seats to purchase)
        - query = """SELECT count(*) FROM ticket WHERE airline_name = %s AND flight_num = %s"""
    - Get the current ticket id. This indicates what should be the next ticket_id. Once a ticket is purchased, a record in the ticket table will be generated, with a record in the purchase table. They two share the same ticket id. In other words, if a ticket is not purchased, even if there are still empty seats on the plane, there will not be a ticket record (which does not have a corresponding purchase record) in the ticket table.
        - query = "select max(ticket_id) from purchases"
    - Insert the newly generated ticket_id into the ticket table.
        - query1 = "insert into ticket values(%s, %s, %s)" cursor.execute(query1,(ticket_id, airline_name, flight_num))
    - Insert a corresponding purchase record into the purchase table

        - query2 = "INSERT INTO purchases(ticket_id,customer_email,purchase_date, booking_agent_id) VALUES(%s,%s,%s,%s)"
- Customer
    - Default show all the flights bought by the customer
        - query = 'select * from flight where status ="Upcoming" and (airline_name,flight_num) in (select airline_name, flight_num from ticket where ticket_id in (select purchases.ticket_id from purchases where customer_email = %s))'
        - The search function is also available. The system will loop through all the tickets that the customer has bought to check whether it satisfies the input requirements and display the filtered records.
    - Default spending based on last year records
        - query = 'select * from flight where (airline_name,flight_num) in (select airline_name, flight_num from ticket where ticket_id in (select purchases.ticket_id from purchases where customer_email = %s and purchase_date <= %s and purchase_date >= %s)) '
    - Default Draw barchart for the consumption
        - query = "SELECT price, purchase_date FROM ticket NATURAL JOIN purchases NATURAL JOIN flight WHERE customer_email = '%s'"
        - The search function is available and the system will loop through all the tickets and select those satisfying input requirements to draw the barchart
    - Search the customer's flight based on input
        - query = 'select * from flight where status ="Upcoming" and (airline_name,flight_num) in (select airline_name, flight_num from ticket where ticket_id in (select purchases.ticket_id from purchases where customer_email = %s))'
        - add = "and departure_airport in (select airport_name from airport where airport_city ='"+ d_city +"')"
        - add = "and arrival_airport in (select airport_name from airport where airport_city = '"+ a_city +"')"
        - If users specify departure date in the input, we will add this to the above query:
            - "and '"+ str(d_start)[:10] +"' <=departure_time"
        - If users specify arrival date in the input, we will add this to the above query:
            - "and '"+ str(a_start)[:10] +"' >=arrival_time"
        - If users specify flight number in the input, we will add this to the above query:
            - appendix += "and flight_num = "
            - appendix += flight_num
        - If users specify departure city in the input, we will add this to the above query:
            - appendix += "and departure_airport = '"

            - appendix += d_airport
            - appendix += """
        - If users specify arrival city in the input, we will add this to the above query:
            - appendix += "and arrival_airport = '"
            - appendix += a_airport
            - appendix += """
- Agent
- (Here we set the fixed commission for agents to be 0.3. In the following steps, we always get the original amount first, and then multiply it by 0.3 using python later.)
    - Default: Show the Upcoming flights that THIS agent purchased for his/her customers:
        - query = '"select flight.airline_name, flight.flight_num, flight.departure_airport, flight.departure_time, flight.arrival_airport, flight.arrival_time, flight.price, flight.status, flight.airplane_id, purchases.customer_email from flight join ticket join purchases where status ="Upcoming" and ticket.flight_num = flight.flight_num AND ticket.ticket_id = purchases.ticket_id and (ticket.airline_name, ticket.flight_num) in (select airline_name, flight_num from ticket where ticket_id in (select purchases.ticket_id from purchases where booking_agent_id in (select booking_agent_id from booking_agent WHERE booking_agent.email = %s)))'"
    - Default commission (Commission = ticket price * 0.3) : show all the tickets that THIS agent purchased in the last 30 days, and then process the data by Python to get the total number and total commission based on a 30% rate of the original price of the ticket:
        - query = '"select * from flight where (airline_name,flight_num) in (select airline_name, flight_num from ticket where ticket_id in (select purchases.ticket_id from purchases where purchase_date <= %s and purchase_date >= %s and booking_agent_id in (SELECT booking_agent_id from booking_agent where email = %s)))'"
    - View top customers, get the Top 5 customers based on the number of tickets bought from the booking agent in the past 6 months:
        - query = """select count(flight.price) AS 'totnum', purchases.customer_email from flight join purchases join ticket where flight.airline_name = ticket.airline_name AND flight.flight_num = ticket.flight_num AND purchases.ticket_id = ticket.ticket_id AND purchases.booking_agent_id in (SELECT booking_agent_id FROM booking_agent WHERE booking_agent.email = %s) AND purchases.purchase_date >= %s AND purchases.purchase_date <= %s GROUP BY purchases.customer_email ORDER by totnum desc LIMIT 5"""
    - View top customers, get the Top 5 customers based on the amount of commission (Commission = ticket price * 0.3) received in the last year. Here we

first get the original revenue earned from the ticket price. Then we multiply it by 0.3 (which is our commission rate) when returning the variable value to html through python:

- query = """select sum(flight.price) AS 'totprice', purchases.customer_email from flight join purchases join ticket where flight.airline_name = ticket.airline_name AND flight.flight_num = ticket.flight_num AND purchases.ticket_id = ticket.ticket_id AND purchases.booking_agent_id in (SELECT booking_agent_id FROM booking_agent WHERE booking_agent.email = %s) AND purchases.purchase_date >= %s AND purchases.purchase_date <= %s GROUP BY purchases.customer_email ORDER by totprice desc LIMIT 5"""

- if user specify time span in View Commission (Commission = ticket price * 0.3), get the total number of tickets and the total commission from the tickets that are bought from the booking agent in the specific time span. Here we first get the original revenue earned from the ticket price. Then we multiply it by 0.3 (which is our commission rate) when returning the variable value to html through python:
  - query = """select * from flight where (airline_name,flight_num) in (select airline_name, flight_num from ticket where ticket_id in (select purchases.ticket_id from purchases where purchase_date <= %s and purchase_date >= %s and booking_agent_id in (SELECT booking_agent_id from booking_agent where email = %s)))"""
- If the user inputs the checking flights form, we will run this query:
  - query = '"select flight.airline_name, flight.flight_num, flight.departure_airport, flight.departure_time, flight.arrival_airport, flight.arrival_time, flight.price, flight.status, flight.airplane_id, purchases.customer_email from flight join ticket join purchases where status ="Upcoming" and ticket.flight_num = flight.flight_num AND ticket.ticket_id = purchases.ticket_id and (ticket.airline_name, ticket.flight_num) in (select airline_name, flight_num from ticket where ticket_id in (select purchases.ticket_id from purchases where booking_agent_id in (select booking_agent_id from booking_agent WHERE booking_agent.email = %s)))"'
  - If users specify departure date in the input, we will add this to the above query:
    - "and '"+ str(d_start)[:10] +"' <=departure_time"
  - If users specify arrival date in the input, we will add this to the above query:
    - "and '"+ str(a_start)[:10] +"' >=arrival_time"
  - If users specify flight number in the input, we will add this to the above query:
    - appendix += "and flight_num = "
    - appendix += flight_num

        - If users specify departure airport in the input, we will add this to the above query:
            - add = "and departure_airport in (select airport_name from airport where airport_city ='"+ d_city +"')"
        - If users specify arrival airport in the input, we will add this to the above query:
            - add = "and arrival_airport in (select airport_name from airport where airport_city = '"+ a_city +"')"
        - If users specify departure city in the input, we will add this to the above query:
            - appendix += "and departure_airport = '"
            - appendix += d_airport
            - appendix += "'"
        - If users specify arrival city in the input, we will add this to the above query:
            - appendix += "and arrival_airport = '"
            - appendix += a_airport
            - appendix += "'"
- Airline staff
    - Default show all the flights for the airline and send back to html
        - query = "select * from flight where airline_name = %s"
    - Search flights based on input data
        - query = "select * from flight where airline_name = %s"
        - add = "and departure_airport in (select airport_name from airport where airport_city ='"+ d_city +"')"
        - add = "and arrival_airport in (select airport_name from airport where airport_city = '"+ a_city +"')"
        - Other features are directly added to the query
    - Default view all the booking_agent based on their commission in the last month and last year and also total tickets sold for the airline
        - query ="select purchases.booking_agent_id from purchases where purchases.purchase_date <= %s and purchases.purchase_date >= %s and purchases.booking_agent_id is NOT NULL and purchases.ticket_id in (select ticket_id from ticket where airline_name = %s) group by purchases.booking_agent_id ORDER BY count(purchases.ticket_id) DESC LIMIT 5 "
        - query = "select purchases.booking_agent_id from purchases, flight,ticket where ticket.airline_name = %s and purchases.booking_agent_id is NOT NULL and (ticket.airline_name, ticket.flight_num) = (flight.airline_name,flight.flight_num) and purchases.ticket_id = ticket.ticket_id and purchases.purchase_date <= %s and purchases.purchase_date >= %s group by purchases.booking_agent_id ORDER BY sum(flight.price) DESC LIMIT 5"

- Default View Frequent Customer based on the total number of tickets they bought from the airline
    - query = "select purchases.customer_email,count(purchases.ticket_id) from purchases, ticket where ticket.ticket_id = purchases.ticket_id and ticket.airline_name = %s and purchases.purchase_date <= %s and %s <= purchases.purchase_date GROUP BY purchases.customer_email ORDER BY count(purchases.ticket_id) LIMIT 10"
    - A button that direct to “view frequent customer” url that can display all the tickets that particular customer bought from the airline
- Default View Reports based on total tickets sold last year and last month
    - query = "select COUNT(ticket.ticket_id) from purchases, ticket where ticket.airline_name = %s and ticket.ticket_id = purchases.ticket_id and purchases.purchase_date <= %s and %s <= purchases.purchase_date"
    - A button direct to “view detailed report” url with search function
- Default view Destination, the most three visited airport and city based on last three month and last years’ records
    - query = "select airport.airport_city from airport, flight where flight.arrival_airport = airport.airport_name AND flight.departure_time <= %s and flight.departure_time >= %s GROUP BY airport_city ORDER BY count(flight.arrival_time) DESC LIMIT 3"’
- View Revenue of last year and last month
    - Here the query is for staff to make a comparison of revenue earned. This is for revenue without an agent. We will run this query twice to get the data for last month and last year. The result will be compared with the result of the query below, with respect to last month and last year.
        - query = """SELECT SUM(flight.price) AS 'totalprice' FROM flight, ticket, purchases WHERE flight.airline_name = ticket.airline_name AND flight.flight_num = ticket.flight_num AND ticket.ticket_id = purchases.ticket_id AND ticket.airline_name = %s AND purchases.booking_agent_id is NULL AND purchases.purchase_date <= %s AND purchases.purchase_date >= %s"""
    - Here the query is for staff to make a comparison of revenue earned. This is for revenue with an agent. We will run this query twice to get the data for last month and last year. The result will be compared with the result of the query above, with respect to last month and last year.
        - query = """SELECT SUM(flight.price) AS 'totalprice' FROM flight, ticket, purchases WHERE flight.airline_name = ticket.airline_name AND flight.flight_num = ticket.flight_num AND ticket.ticket_id = purchases.ticket_id AND ticket.airline_name = %s AND purchases.booking_agent_id is not NULL AND purchases.purchase_date <= %s AND purchases.purchase_date >= %s"""
- Create new flight

        - Fetch all airport name and airplane_id, max(flight_num) from the database and send back to html file:
            - query = "select airport_name from airport"
            - query = "select airplane_id from airplane where airline_name = %s"
            - query = "select max(flight_num) from flight where airline_name = %s"
        - The previous procedure guarantees all input data is valid, insert it into the databases with flight_num automatically increase 1:
            - query1 = "INSERT into flight values (%s,%s,%s,%s,%s,%s,%s,%s,%s) "
            - Display “Successfully created” and the assigned
    - Create new airport
        - Fetch all airplane ids from the database and check whether the input airplane_id already exists:
            - query = "select * from airport where airport_name = %s "
            - Display “The airport already exists”
        - If all input data is valid, insert it into the databases:
            - query = "INSERT INTO airport values (%s,%s)"
            - Display “Successfully created”
    - Add new airplanes
        - Fetch all airplane ids from the database and check whether the input airplane_id already exists:
            - query = "select airplane_id from airplane where airline_name = %s"
            - Display “The airplane_id already exists”
        - If all input data is valid, insert it into the databases:
            - query = "INSERT INTO airplane values (%s,%s,%s)"
            - Display “Successfully created”
    - Grant new permission
        - Get a list of staff with their permissions in the SAME Airline with the Current User.
            - query = """SELECT airline_staff.username, permission.permission_type FROM permission, airline_staff WHERE airline_staff.username = permission.username AND airline_staff.airline_name = %s"""
        - Before assigning permission to this staff, check whether this staff already have this type of permission:
            - query = """SELECT * FROM permission, airline_staff WHERE permission.username = airline_staff.username AND permission.permission_type = %s AND permission.username = %s"""
        - Grant permission to this work:
            - query = """INSERT INTO permission VALUES (%s, %s)"""

        - If the selected worker does not have any type of permission before the granting, he/she will have a blank roll in the permission table (i.e. there is a record saying that his/her permission type is blank). We delete that blank roll.
            - query = """DELETE FROM permission WHERE permission.username = %s AND permission.permission_type = """"
    - Add booking agents
        - Check whether this booking agent exists (check whether the input is valid)
            - query = """SELECT * FROM booking_agent WHERE email = %s"""
        - Check whether this agent has already work for this company:
            - query = """SELECT * FROM booking_agent_work_for WHERE booking_agent_work_for.email = %s AND booking_agent_work_for.airline_name = %s"""
        - Add booking agents that can work for this airline
            - query = """INSERT INTO booking_agent_work_for VALUES (%s, %s)"""
    - Change flight status
        - Get all the flights of the company:
            - query = """SELECT * FROM flight WHERE airline_name = %s"""
        - Check whether the flight is already in this status:
            - query = """SELECT * FROM flight WHERE airline_name = %s AND flight_num = %s AND status = %s"""
        - Change the status of the flight:
            - query = """UPDATE `flight` SET `status` = %s WHERE `flight`.`airline_name` = %s AND `flight`.`flight_num` = %s"""
    - Detailed report
        - Get the total number of ticket of last month of the airline that THIS user works for
            - query = """select COUNT(ticket.ticket_id)   from purchases, ticket where ticket.airline_name = %s and ticket.ticket_id = purchases.ticket_id   and purchases.purchase_date <= %s and %s <= purchases.purchase_date"""
        - Get all tickets of last year of the airline that THIS user works for, and then use python to classify them according to their time
            - query = """select ticket.ticket_id, purchases.purchase_date from purchases, ticket   where ticket.airline_name = %s and ticket.ticket_id = purchases.ticket_id and purchases.purchase_date <= %s   and %s <= purchases.purchase_date"""

        - Get all tickets of the specific user input time span of the airline that THIS user works for, and then use python to classify them according to their time
            - query = """select ticket.ticket_id, purchases.purchase_date from purchases, ticket where ticket.airline_name = %s and ticket.ticket_id = purchases.ticket_id and purchases.purchase_date <= %s and %s <= purchases.purchase_date"""
    - View frequent customer
        - Airline Staff will also be able to see the most frequent customer within the last year. They will be able to see a list of all flights a particular Customer has taken only on that particular airline.
            - query = """SELECT flight.flight_num, flight.departure_airport, flight.departure_time, flight.arrival_airport, flight.arrival_time, flight.price, flight.status, flight.airplane_id, ticket.ticket_id, purchases.booking_agent_id, purchases.purchase_date FROM flight, ticket, purchases WHERE flight.airline_name = ticket.airline_name AND flight.flight_num = ticket.flight_num AND ticket.ticket_id = purchases.ticket_id AND flight.airline_name = %s AND purchases.customer_email = %s"""
    - View customer of a particular flight
        - Select all customers who bought the flight ticket
            - query = "select purchases.customer_email from ticket,purchases where ticket.airline_name = %s and ticket.flight_num = %s and purchases.ticket_id = ticket.ticket_id"