# Use Cases & Queries

## Staff use Cases

### Display flight info at homepage

```
SELECT DISTINCT
dept_date,dept_time,flight_num,dept_airport,arri_date,arri_time,base_price,flight_status,a
rri_airport
FROM Working Natural JOIN Flight
WHERE Flight.dept_date >= (select CURDATE())
or ( Flight.dept_date = (select CURDATE()) and Flight.dept_time > (select CURTIME()))
and Flight.dept_date <= (SELECT DATE_ADD(CURDATE(), INTERVAL 30 DAY))
```

display flight information that will depart in 30 days

### Search flight info at homepage

```
query = 'SELECT DISTINCT
dept_date,dept_time,flight_num,dept_airport,arri_date,arri_time,base_price,flight_status,a
rri_airport ' \
            'FROM Flight ' \
            'WHERE dept_date >= %s and dept_date <= %s and dept_airport = %s and
arri_airport = %s'
    cursor.execute(query, (beginDate, endDate, sourceAirport, destinationAirport))
```

search flight information according to user input

### Login authorization

```
    query = 'SELECT * FROM Airline_Staff WHERE username = %s and air_password = %s'
    cursor.execute(query, (username, password))
    query = 'SELECT * FROM customer WHERE email = %s and customer_password = %s'
    cursor.execute(query, (username, password))
```

search in both staff and customer database to determine if the user is a staff or a customer. Based on the result of searching, different sessions will be assigned to the user. The password is md5 hashed.

## Staff register

```
ins = 'INSERT INTO Airline_Staff VALUES(%s, %s, %s, %s, %s, %s)'
cursor.execute(ins, (username, password, firstname, lastname, dateOfBirth, airline))
ins = 'INSERT INTO Working VALUES(%s, %s)'
cursor.execute(ins, (airline, username))

  for email in emails:
    cursor.execute('INSERT INTO staff_email VALUES(%s, %s)', (username, email))
  for phone in phones:
    cursor.execute('INSERT INTO staff_phone_num VALUES(%s, %s)', (username, phone))
```

If a staff is successfully register, information in 'Airline_staff', 'Working', 'staff_email', and 'staff_phone_num' should all be updated.

## Customer register

```
ins = 'INSERT INTO customer VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)'
cursor.execute(ins, (email, firstname, lastname, password, building, street, apartment, city, state, zipcode, passport_num, passport_exp_date, passport_country, date_of_birth))

for phone in phones:
  cursor.execute('INSERT INTO customer_phone_number VALUES(%s, %s)', (email, phone))
```

If a customer is successfully register, information in 'customer' and 'customer_phone_num' should all be updated.

## View flight

```
query = 'SELECT DISTINCT
dept_date,dept_time,flight_num,dept_airport,arri_date,arri_time,base_price,flight_status,a
rri_airport ' \
'FROM Working Natural JOIN Flight ' \
'WHERE Working.username =  %s ' \
'and Flight.dept_date >= (select CURDATE()) or ( Flight.dept_date = (select CURDATE()) and
Flight.dept_time > (select CURTIME())) ' \
'and Flight.dept_date <= (SELECT DATE_ADD(CURDATE(), INTERVAL 30 DAY))'

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

Display all flights that depart in 30 days and belongs to the staff's airline.

## Search Flight

```
query = 'SELECT DISTINCT
dept_date,dept_time,flight_num,dept_airport,arri_date,arri_time,base_price,flight_status,a
rri_airport ' \
'FROM Working Natural JOIN Flight ' \
'WHERE Working.username = %s and Flight.dept_date >= %s and Flight.dept_date <= %s and
dept_airport = %s and arri_airport = %s'

cursor.execute(query, (username, beginDate, endDate, sourceAirport, destinationAirport))
```

Search flights based on the information provided.

## Search customer

```
query = 'Select of.flight_num,customer.email, customer.first_name, customer.last_name ' \
            'From customer natural join purchase ' \
            'join of on of.id_num_ticket = purchase.id_num ' \
            'join Working on Working.name = of.name ' \
            'where Working.username = %s and of.flight_num = %s and of.dept_date = %s and
of.dept_time = %s'

cursor.execute(query, (username, flight_num, Departure_Date, Departure_Time))
```

Search customer based on the information provided

## Create flights

```
query0 = 'SELECT * ' \
             'FROM maintainance natural join Working ' \
             'WHERE username = %s and id_num = %s ' \
             'AND (((start_date < %s) OR (start_date = %s AND start_time <= %s)) ' \
             'AND ((end_date > %s) OR (end_date = %s AND end_time >= %s)) ' \
             'OR ((start_date < %s) OR (start_date = %s AND start_time <= %s)) ' \
             'AND ((end_date > %s) OR (end_date = %s AND end_time >= %s))) '
    cursor.execute(query0, (username, id_num, dept_date, dept_date, dept_time, arri_date,
arri_date, arri_time, arri_date, arri_date, arri_time, dept_date, dept_date, dept_time))

  query = 'INSERT INTO Flight (name, id_num, dept_date, dept_time, flight_num,
dept_airport, arri_date, arri_time, base_price, flight_status, arri_airport) ' \
            'VALUES((SELECT name FROM Working WHERE username =
%s),%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)'

  cursor.execute(query, (
                username, id_num, dept_date, dept_time, flight_num, dept_airport,
arri_date,                  arri_time, base_price, flight_status, arri_airport))
```

First check if the airplane is under maintenance. If not, update 'Flight'.

## Change flight status

```
query = 'UPDATE Flight ' \
            'SET flight_status = %s ' \
            'WHERE flight_num = %s and dept_date = %s and dept_time = %s and name =
(select name from Working where username = %s) '

    cursor.execute(query, (status, flight_num, Departure_Date, Departure_Time, username))
```

Change flight status to on-time/delayed/canceled

## Create Airplane

```
query = 'INSERT INTO AirPlane (name, id_num, num_of_seats, manu_company, model_num,
manu_date, age) ' \
                'VALUES((SELECT name FROM Working WHERE username = %s),
%s,%s,%s,%s,%s,%s)'
cursor.execute(query, (username, id_num, num_of_seats, manu_company, model_num, manu_date,
age))
```

Create new airplane and update 'AirPlane'

## View airplane

```
query = 'SELECT DISTINCT name,id_num,num_of_seats,manu_company,model_num,manu_date,age ' \
            'FROM Working Natural JOIN AirPlane WHERE Working.username = %s'

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

Display all current airplanes.

## Create Airport

```
query = 'INSERT INTO Airport (air_code, name, city, country, num_of_terminal, type) ' \
                'VALUES(%s,%s,%s,%s,%s,%s)'
cursor.execute(query, (air_code, name, city, country, number_of_terminal, type))
query = 'INSERT INTO INSIDE (air_code, name) VALUES (%s, (SELECT name FROM Working WHERE
username = %s))'
cursor.execute(query, (air_code, username))
```

Create new airport and update 'Airport' and 'Inside', which is the relationship between airport and airline

## View Flight Ratings

```
query = 'SELECT of.flight_num, AVG(cus_comment.rate) as average_rating ' \
            'FROM of JOIN cus_comment ON of.id_num_ticket = cus_comment.id_num_ticket ' \
            'JOIN Working ON of.name = Working.name WHERE Working.username = %s ' \
            'GROUP BY of.flight_num'

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

Display the average rating of all flights

## Search flight ratings

```
query = 'SELECT of.flight_num, of.dept_date, of.dept_time, cus_comment.comments,
cus_comment.rate ' \
            'FROM cus_comment natural join of ' \
            'JOIN Working ON of.name = Working.name ' \
            'WHERE of.flight_num = %s and of.dept_date = %s and of.dept_time =%s and
Working.username = %s'

cursor.execute(query, (flight_num, Departure_Date, Departure_Time, username))
```

Search the rating of flight based on the information provided

## Create maintenance

```
query = 'INSERT INTO maintainance (name, id_num, start_date, start_time, end_date,
end_time) ' \
        'VALUES((SELECT name FROM Working WHERE username = %s), %s,%s,%s,%s,%s)'
cursor.execute(query, (username, id_num, start_date, start_time, end_date, end_time))
```

Create a new maintenance for airplane and update 'maintainance'

## View Frequent Customer

```
query = 'SELECT email, COUNT(*) as flight_count ' \
        'FROM of natural JOIN Working ' \
        'JOIN purchase on purchase.id_num = of.id_num_ticket ' \
        'WHERE Working.username = %s and of.dept_date >= CURDATE() - INTERVAL 1 YEAR ' \
        'GROUP BY purchase.email ' \
        'ORDER BY flight_count DESC ' \
        'LIMIT 1 '

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

Display the email and the flight count of the most frequent customer

## Search Customer Flight

```
query = 'SELECT of.flight_num, of.dept_date, of.dept_time ' \
        'FROM of natural JOIN Working ' \
        'JOIN purchase ON of.id_num_ticket = purchase.id_num ' \
        'WHERE email = %s AND Working.username = %s '

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

Search all flights of a customer

## View Earned Revenue

```
query_last_month = 'SELECT SUM(ticket_price) as last_month_revenue ' \
                   'FROM of natural join Working ' \
                   'join ticket on of.id_num_ticket = ticket.id_num ' \
                   'WHERE Working.username = %s and dept_date BETWEEN CURDATE() - INTERVAL
1 MONTH AND CURDATE()'
cursor.execute(query_last_month, (username))

query_last_year = 'SELECT SUM(ticket_price) as last_year_revenue ' \
                  'FROM of natural join Working ' \
                  'join ticket on of.id_num_ticket = ticket.id_num ' \
                  'WHERE Working.username = %s and dept_date BETWEEN CURDATE() - INTERVAL
1 YEAR AND CURDATE()'
cursor.execute(query_last_year, (username))
```

Display last month revenue and last year revenue of the airline

# Customer use cases

## track spending

```
query = """
    SELECT MONTHNAME(p.date_purchase) AS month, SUM(t.ticket_price) AS total_spent
    FROM purchase as p, ticket as t
    WHERE p.email = %s AND p.date_purchase BETWEEN %s AND %s AND p.id_num = t.id_num
    GROUP BY MONTH(p.date_purchase)
    ORDER BY MONTH(p.date_purchase)
"""
cursor.execute(query, (customer_email, start_date, end_date))
```

display the spending with a specific time period

## Search purchased flight

```
# time range
query = """
    SELECT DISTINCT flight.id_num, flight.name, flight.dept_date, flight.dept_time,
flight.arri_date, flight.arri_time, flight.dept_airport
    , flight.arri_airport, flight.flight_status
    FROM Flight join of on of.flight_num = flight.flight_num
             join purchase on of.id_num_ticket = purchase.id_num
    WHERE email = %s AND flight.dept_date BETWEEN %s AND %s
    ORDER BY flight.dept_date, flight.dept_time
"""
cursor.execute(query, (customer_email, start_date, end_date))

# departure airport
query = """
                SELECT DISTINCT flight.id_num, flight.name, flight.dept_date,
flight.dept_time, flight.arri_date, flight.arri_time, flight.dept_airport
                , flight.arri_airport, flight.flight_status
                FROM Flight join of on of.flight_num = flight.flight_num
                          join purchase on of.id_num_ticket = purchase.id_num
                WHERE email = %s AND flight.dept_airport LIKE %s
                ORDER BY flight.dept_date, flight.dept_time
            """
            cursor.execute(query, (customer_email, f"%{departure_airport}%"))

# arrival airport
query = """
                SELECT DISTINCT flight.id_num, flight.name, flight.dept_date,
flight.dept_time, flight.arri_date, flight.arri_time, flight.dept_airport
                , flight.arri_airport, flight.flight_status
                FROM Flight join of on of.flight_num = flight.flight_num

                          join purchase on of.id_num_ticket = purchase.id_num
```

```
                        join purchase on of.id_num_ticket = purchase.id_num
                    WHERE email = %s AND flight.arri_airport LIKE %s
                    ORDER BY flight.dept_date, flight.dept_time
                """
                cursor.execute(query, (customer_email, f"%{arrival_airport}%"))
```

The customer could either search their purchased flights by time range, departure airport, or arrival airport.

## Rate and comment

```
cursor.execute("INSERT INTO cus_comment (id_num_ticket, email, comments, rate) VALUES (%s, %s, %s, %s)", (flight_id, customer_email, rate_comment, rate))
```

Customers can rate and comment on a flight that they took before. 'cus_comment' will be updated.

## View past comment

```
cursor.execute("SELECT c.comments, c.rate FROM cus_comment c WHERE c.email = %s",
               (customer_email))
```

Display previous comment

## Search for flights that are available for comments

```
cursor.execute("""
        SELECT distinct p.id_num, f.name
        FROM Flight f
        JOIN of o ON f.id_num = o.id_num_ap and o.dept_date = f.dept_date
        JOIN ticket t ON o.id_num_ticket = t.id_num
        JOIN purchase p ON t.id_num = p.id_num
        LEFT JOIN cus_comment c ON o.id_num_ticket = c.id_num_ticket and c.email = %s
        WHERE f.arri_date < CURDATE()
            AND (c.id_num_ticket IS NULL OR c.email <> %s)
            AND p.id_num NOT IN (SELECT id_num_ticket FROM cus_comment WHERE email = %s)
    """, (customer_email, customer_email, customer_email))
```

Only flights that already arrived can be rated or commented by the customers

## Cancel Trip

```
cancel_trip_query = """
    DELETE FROM purchase
    WHERE email = %s AND id_num = %s
"""
cursor.execute(cancel_trip_query, (customer_email, ticket_id))

cancel_trip_query2 = """
    DELETE FROM of
    WHERE id_num_ticket = %s
"""
cursor.execute(cancel_trip_query2, (ticket_id))

cancel_trip_query3 = """
    DELETE FROM ticket
    WHERE id_num = %s
"""
cursor.execute(cancel_trip_query3, (ticket_id))
```

If customers want to cancel a trip, 'purchase', 'of', and 'ticket' should all be updated

## Display trips that can be canceled

```
cursor.execute("""
SELECT t.id_num
FROM purchase t, of m
WHERE m.id_num_ticket = t.id_num
AND m.dept_date - 1 > CURDATE()
AND t.email = %s
""", (customer_email))
```

Customers can only cancel a trip 1 day before the departure date of the flight

## Calculate taken seats

```
taken_seats_query = """
    SELECT (Airplane.num_of_seats - COUNT(of.id_num_ap)) AS available_seat
    FROM of
    JOIN Airplane ON of.id_num_ap = Airplane.id_num
    WHERE of.id_num_ap = %s AND Airplane.id_num = %s
"""
cursor.execute(taken_seats_query, (flight_id, flight_id))
```

## Calculate Actual Price

```
query = """
    SELECT flight_num, base_price * CASE
        WHEN (%s / Airplane.num_of_seats) >= 0.8 THEN 1.25
        ELSE 1
    END AS actual_price
    FROM Flight
    JOIN Airplane ON Flight.id_num = Airplane.id_num
    WHERE flight_num = %s;
"""
cursor.execute(query, (taken_seats, flight_id))
```

When the number of taken seats is greater than 80% of the total seats, the actual price is equal to 1.25 times the base price

## Search flights

```
query = """
    SELECT Flight.*,
           AirPlane.num_of_seats,
           (AirPlane.num_of_seats - COUNT(of.id_num_ticket)) AS available_seats
    FROM Flight
    LEFT JOIN AirPlane ON Flight.id_num = AirPlane.id_num
    LEFT JOIN of ON Flight.id_num = of.id_num_ap
    WHERE Flight.dept_airport = %s
    AND Flight.arri_airport = %s
    AND Flight.dept_date = %s
    GROUP BY Flight.id_num
    HAVING available_seats > 0
"""
cursor.execute(query, (departure_airport, arrival_airport, departure_date))
```

Search available flights based on the information provided

Search available flights based on the information provided

## Check if the ticket id is unique

```
check_query = """
    SELECT id_num
    FROM ticket
    WHERE id_num = %s
    UNION
    SELECT id_num_ticket
    FROM of
    WHERE id_num_ap = %s AND id_num_ticket = %s
"""
cursor.execute(check_query, (id_num, flight_id, id_num))
result = cursor.fetchone()
cursor.close()
return result is None
```

If the id is unique, return true, else return false.

## Purchase a ticket

```
insert_ticket_query = """
                    INSERT INTO ticket (id_num, ticket_price)
                    VALUES (%s, %s)
                """
cursor.execute(insert_ticket_query, (id_num, int(ticket_actual_price)))

insert_of_query = """
    INSERT INTO of (name, id_num_ap, dept_date, dept_time, flight_num, id_num_ticket)
    SELECT name, id_num, dept_date, dept_time, flight_num, %s
    FROM Flight
    WHERE flight_num = %s
"""
cursor.execute(insert_of_query, (id_num, flight_id))

insert_query = """
            INSERT INTO purchase (id_num, email, date_purchase, time_purchase, card_type,
card_num, name_on_card, exp_date, passenger_name)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
        """
cursor.execute(insert_query, (
    id_num, customer_email, date_purchase, time_purchase,
```

```
    card_type, card_num, name_on_card, exp_date, passenger_name
))
```

After purchase a ticket, 'ticket', 'of', and 'purchase' should all be updated.