User cases & SQL queries

######----------public info—----------
-Search based on city:

```
        search = 'SELECT * from flight\
         where status = "upcoming" and departure_time like %s\
          and departure_airport in \
          (select airport_name from airport where airport_city = %s)\
           and arrival_airport in (select airport_name from airport where airport_city = %s)'
   cursor.execute(search,(date, dep, arr))
```

-Search based on airport:

```
        search = 'SELECT * from flight where status = "upcoming" and departure_time \
            like %s and departure_airport = %s and arrival_airport = %s'
   cursor.execute(search,(date, dep, arr))
```

-Find flight status based on departure info:

```
        query = "select airline_name, flight_num, STATUS from flight where departure_time like
%s and flight_num = %s"
   cursor.execute(query,(date, flight_num))
```

-Find flight status based on arrival info:

```
        query = "select airline_name, flight_num, STATUS from flight where \
      arrival_time like %s and flight_num = %s"
   cursor.execute(query,(date, flight_num))
```

######----------register—----------
For customers:

```
        query = 'SELECT * FROM booking_agent WHERE email = %s'
        cursor.execute(query, (username))
        ##If the username doesn't exist, then we are good to go.
        ins = "INSERT INTO customer VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
%s)"
        cursor.execute(ins,(username, name, md5_pw, building_number,street, city,
state,phone_number, passport_number,  passport_expiration, passport_country,
date_of_birth))
```

For booking agents:

```
        query = 'SELECT * FROM customer WHERE email = %s'
        cursor.execute(query, (username))
        ##If the username doesn't exist, then we are good to go.
        ins = 'INSERT INTO booking_agent VALUES(%s, %s, %s)'
        cursor.execute(ins, (username, md5_pw,booking_id))
```

For airline staff:

```
	query = 'SELECT * FROM airline_staff WHERE username = %s'
	cursor.execute(query, (username))
	##If the username doesn't exist, then we are good to go.
	query = 'SELECT * FROM airline WHERE airline_name = %s'
	cursor.execute(query, (airlineName))
	##If the airline name is valid, then we good to go
	ins = "INSERT INTO airline_staff VALUES(%s, %s, %s,%s, %s, %s)"
	cursor.execute(ins,(username, md5_pw, first_name, last_name, date_of_birth,
airline_name))
```

#####----------login—----------

```
	##Based on usertype, check if the username and pwd match anyone in the database
	if usertype == "customer":
		query = 'SELECT * FROM customer WHERE email = %s and password = %s'
		cursor.execute(query, (username, md5_pw))
	elif usertype == "booking_agent":
		query = 'SELECT * FROM booking_agent WHERE email = %s and password = %s'
		cursor.execute(query, (username, md5_pw))
	elif usertype == "staff":
		query = 'SELECT * FROM airline_staff WHERE username = %s and password =
%s'
		cursor.execute(query, (username, md5_pw))
```

#####----------customer—----------

- View My flight(default)

query = "Select * from flight\
where status = 'upcoming' and flight_num in \
(select flight_num from (purchases join ticket using(ticket_id)) where customer_email = %s)"
cursor.execute(query,(username,))

- View my flight(specific) by date

querymap = {"customer": "Select * from flight\
where (departure_time between %s and %s) and flight_num in \
(select flight_num from (purchases join ticket using(ticket_id)) where customer_email = %s)"}
query = querymap[usertype]
cursor.execute(query,(start, end, username))

- View my flight(specific) by city

querymap = {"customer": "Select * from flight\
where departure_airport in (select airport_name from airport \
where airport_city like %s) \
and arrival_airport in \
(select airport_name from airport where airport_city like %s) and flight_num in \
(select flight_num from (purchases join ticket using(ticket_id)) where customer_email = %s)"}
query = querymap[usertype]
cursor.execute(query,(d_name, a_name, username))

- purchase

#First find seat counts:
seatcount = "select seats from airplane natural join flight where flight.airline_name = %s and flight.flight_num = %s"
cursor.execute(seatcount,(airline_name, flight_num))

```
#Next find ticket counts:
ticketcount = "select count(*) from ticket where airline_name = %s and flight_num =
%s"
   cursor.execute(ticketcount,(airline_name, flight_num))

#If seat counts > ticket counts, generate ticket id
query_ticket_id = "SELECT MAX(ticket_id) + 1 FROM ticket;"
   cursor.execute(query_ticket_id)

#then purchase
        q1 = "Insert into ticket (ticket_id, airline_name, flight_num) values(%s, %s, %s)"
   q2 = "Insert into purchases (ticket_id, customer_email, purchase_date) values(%s, %s,
CURDATE())"
   cursor.execute(q1,(ticket_id, airline_name, flight_num))
   conn.commit()
   cursor.execute(q2,(ticket_id, username))
   conn.commit()
```

- Track spending

```
1.track by default
year = "select sum(price) from (purchases natural join ticket) natural join flight where
customer_email = %s\
      and purchase_date > date_sub(now(),INTERVAL "+str(x)+" MONTH) AND
purchase_date < now()"

   month = "select sum(price), convert(purchase_date, varchar(7)) as S from (purchases
natural join ticket) natural join\
      flight where purchases.customer_email = %s and purchase_date >
date_sub(now(),INTERVAL "+str(y)+" MONTH) AND purchase_date < now() group by
S"
```

```
    cursor.execute(year,(username,))
    spending = cursor.fetchone()[0]
    cursor.execute(month,(username,))
    spent = cursor.fetchall()
```

2.track by specific input

```
month = "select sum(price) as S from purchases natural join ticket natural join flight
where customer_email = %s\
        and purchase_date between %s and %s"

    monthwise = "select sum(price) as S,convert(purchase_date, varchar(7)) as T from
purchases natural join ticket natural join\
      flight where purchases.customer_email = %s and purchase_date between %s and
%s group by T"
    cursor.execute(month,(username, start, end))
    spending = cursor.fetchone()[0]
    cursor.execute(monthwise,(username, start, end))
    spent = cursor.fetchall()
```

#####----------booking agent—----------

- View My flight(default)

```
query = "Select * from flight\
      where status = 'upcoming' and flight_num in \
        (select flight_num from (purchases join ticket using(ticket_id)) join
booking_agent\
          using(booking_agent_id) where booking_agent.email = %s)"
    cursor.execute(query,(username,))
```

- View my flight(specific) by date

```
querymap = {"booking_agent": "Select * from flight\
        where (departure_time between %s and %s) and flight_num in \
          (select flight_num from (purchases join ticket using(ticket_id))\
           join booking_agent using(booking_agent_id) where \
             booking_agent.email = %s)"
}
query = querymap[usertype]
cursor.execute(query,(start, end, username))
```

- View my flight(specific) by city

```
querymap = {"booking_agent": "Select * from flight\
        where departure_airport in (select airport_name from airport \
          where airport_city like %s) and arrival_airport in \
          (select airport_name from airport where airport_city like %s) \
          and flight_num in \
          (select flight_num from (purchases join ticket using(ticket_id)) join
booking_agent using(booking_agent_id) where \
          booking_agent.email = %s)"
}
query = querymap[usertype]
   cursor.execute(query,(d_name, a_name, username))
```

- purchase

Same as customer except for the last step:

```
query = "select booking_agent_id from booking_agent where email = %s"
   q1 = "Insert into ticket (ticket_id, airline_name, flight_num) values(%s, %s, %s)"
   q2 = "Insert into purchases (ticket_id, customer_email, booking_agent_id,
purchase_date) values(%s, %s, %s, CURDATE())"
   cursor.execute(query,(username,))
   booking_agent_id = cursor.fetchone()
   cursor.execute(q1,(ticket_id, airline_name, flight_num))
```

```
    conn.commit()
    cursor.execute(q2,(ticket_id, customer_email, booking_agent_id[0]))
    conn.commit()
```

- View commission

  1. Default view

```
# compute total commission
    com = "SELECT sum(flight.price * 0.1) from flight natural join ticket natural join
purchases where booking_agent_id = %s \
            and purchase_date >= date_sub(now(),INTERVAL " + str(x) +" MONTH)"
# compute total ticket sold
    tick = "SELECT count(ticket_id) from ticket natural join purchases where
booking_agent_id = %s and \
            purchase_date >= date_sub(now(),INTERVAL "+str(x)+" MONTH)"
    cursor.execute(com,(booking_agent_id,))
    total_commission = cursor.fetchone()[0]
    cursor.execute(tick,(booking_agent_id,))
    total_ticket = cursor.fetchone()[0]
```

  2. View specifically

```
# compute total commission
    com = "SELECT sum(flight.price * 0.1) as S from (flight natural join ticket) natural
join purchases where booking_agent_id = %s \
            and purchase_date between %s and %s"
# compute total ticket sold
    tick = "SELECT count(ticket_id) as S from ticket natural join purchases where
booking_agent_id = %s and \
            purchase_date between %s and %s"

    cursor.execute(com,(booking_agent_id, start, end))
    total_commission = cursor.fetchone()[0]
    cursor.execute(tick,(booking_agent_id, start, end))
    total_ticket = cursor.fetchone()[0]
```

- View top customers

  1.Top 5 customers based on number of tickets bought from the booking agent in the past 6 months

```
q = "select customer_email, name, count(*) as S from (purchases natural join
booking_agent) inner join customer on customer_email\
```

= customer.email where purchase_date >= date_sub(now(),INTERVAL " + str(x) + " MONTH) and booking_agent.email = %s \
group by customer.email order by S DESC limit 5"
cursor.execute(q,(username,))

2.Top 5 customers based on amount of commission received in the last year

q = "select customer_email, name, sum(price * 0.1) as S from (purchases natural join ticket natural join flight natural join booking_agent)\
inner join customer on customer_email = customer.email where purchase_date >= date_sub(now(),INTERVAL " + str(x) + " MONTH) and booking_agent.email = %s \
group by customer.email order by S DESC limit 5"

cursor.execute(q,(username,))

#####----------airline staff—----------

1. Create new flight.

```
ins = "INSERT INTO flight VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)"
cursor.execute(ins,
(airline_name,flight_num,departure_airport,departure_time,arrival_airport,
arrival_time,price,status,airplane_id))
##This is included in a try-except statement. Will ask the user to retry if
anything goes wrong
```

2. Change status of flight.

```
update = "UPDATE flight SET status = %s WHERE flight_num = %s AND airline_name
= %s"
cursor.execute(update,(status,flight_num,airline_name))
##Update flight info according to flight_num and new desired status that user
enters
```

3. Add airplane.

```
ins = "INSERT INTO airplane VALUES (%s,%s,%s)"
cursor.execute(ins,(airline_name,airplane_id,seats))
##Insert a new airplane to the database given airline_name and airplane_id and
seats
```

4. Add new airport.

```
query = "SELECT * FROM airport WHERE airport_name = %s"
```

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

5. View booking agents

```
##Get booking agent lists order by sales descending for last 30 days
   query = "SELECT email, booking_agent_id, count(ticket_id) as sales\
       FROM (booking_agent NATURAL JOIN ticket NATURAL JOIN purchases)\
       WHERE airline_name = %s AND DateDiff(CURDATE(), purchase_date) <= 30 \
       GROUP BY email, booking_agent_id\
       ORDER BY sales DESC"
   cursor.execute(query,(airline_name))
##Get booking agent lists order by sales descending for last 365 days
   query = "SELECT email, booking_agent_id, count(ticket_id) as sales\
       FROM booking_agent NATURAL JOIN purchases NATURAL JOIN ticket\
       WHERE airline_name = %s AND DateDiff(CURDATE(), purchase_date) <= 365 \
       GROUP BY email, booking_agent_id\
       ORDER BY sales DESC"
   cursor.execute(query,(airline_name))
##Get booking agent lists order by commissions earned for last 365 days
   query = "SELECT email, booking_agent_id, sum(price)*0.1 as commission \
       FROM ((booking_agent NATURAL JOIN purchases) NATURAL JOIN ticket)
NATURAL JOIN flight\
       WHERE airline_name = %s AND DateDiff(CURDATE(), purchase_date) <= 365 \
       GROUP BY email, booking_agent_id\
       ORDER BY commission DESC"
   cursor.execute(query,(airline_name))
```

6. View frequent customers.

```
##Get customer lists order by purchases for last 365 days
query = "SELECT email, name, count(purchases.ticket_id) as sales\
       FROM customer, ticket, purchases\
       WHERE airline_name = %s AND customer.email = purchases.customer_email\
       AND purchases.ticket_id = ticket.ticket_id AND DateDiff(CURDATE(),
purchase_date) <= 365 \
       GROUP BY email, name \
       ORDER BY sales DESC"
   cursor.execute(query,(airline_name))
##Get the info of the flights that a certain customer has
query = "SELECT flight_num, departure_airport, departure_time, arrival_airport,
arrival_time,status,price\
       FROM flight NATURAL JOIN purchases NATURAL JOIN ticket\
       WHERE customer_email = %s AND airline_name = %s"
   cursor.execute(query,(customer_email,airline_name))
```

7. View reports.

```
#Get stats over the past 30 days
query = "SELECT count(*) FROM ticket NATURAL JOIN purchases\
       WHERE airline_name = %s AND DateDiff(CURDATE(), purchase_date) <= 30"
cursor.execute(query,(airline_name))
#Get stats over the past 365 days
query = "SELECT count(*) FROM ticket NATURAL JOIN purchases\
       WHERE airline_name = %s AND DateDiff(CURDATE(), purchase_date) <= 365"
cursor.execute(query,(airline_name))
#Get monthwise stats for last 12 months
   today = str(datetime.date.today())
   curr_year = int(today[:4])
   curr_month = int(today[5:7])
   years = [str(curr_year - 1) for i in range(12-curr_month)] + [str(curr_year)
for i in range(curr_month)]
   months = [str(i) if i >= 10 else "0"+str(i) for i in range(curr_month+1,13)]
+ [str(i) if i >= 10 else "0"+str(i) for i in range(1,curr_month+1)]
   yms = [years[i]+"-"+months[i]+"-%" for i in range(12)]
   query = "SELECT count(*) FROM ticket NATURAL JOIN purchases\
       WHERE airline_name = %s AND purchase_date LIKE %s"
   monthwise_stats = []
   for ym in yms:
       cursor.execute(query,(airline_name,ym))
       monthwise_stats.append(cursor.fetchone()[0])
```

8. Comparisons of revenue earned.

```
#Get direct revenue last month
   query = "SELECT sum(price)\
       FROM flight NATURAL JOIN ticket NATURAL JOIN purchases\
       WHERE airline_name = %s AND booking_agent_id is Null AND
DateDiff(CURDATE(), purchase_date) <= 30"
   cursor.execute(query,(airline_name))
#Get indirect revenue last month
   query = "SELECT 0.9*sum(price)\
       FROM flight NATURAL JOIN ticket NATURAL JOIN purchases\
       WHERE airline_name = %s AND booking_agent_id is not Null AND
DateDiff(CURDATE(), purchase_date) <= 30"
   cursor.execute(query,(airline_name))
#Get direct revenue last year
   query = "SELECT sum(price)\
```

```
        FROM flight NATURAL JOIN ticket NATURAL JOIN purchases\
        WHERE airline_name = %s AND booking_agent_id is Null AND
DateDiff(CURDATE(), purchase_date) <= 365"
    cursor.execute(query,(airline_name))
#Get indirect revenue last year
    query = "SELECT 0.9*sum(price)\
        FROM flight NATURAL JOIN ticket NATURAL JOIN purchases\
        WHERE airline_name = %s AND booking_agent_id is not Null AND
DateDiff(CURDATE(), purchase_date) <= 365"
    cursor.execute(query,(airline_name))
```

9. View top destinations.

```
#Get most popular cities over last 3 months
    query = "SELECT airport.airport_city, count(ticket.ticket_id) as totalnum\
        FROM ticket NATURAL JOIN flight, airport \
        WHERE flight.airline_name = %s AND flight.arrival_airport =
airport.airport_name \
            AND DateDiff(CURDATE(), flight.arrival_time) <= 90 \
        GROUP BY airport.airport_city\
        ORDER BY totalnum DESC"
    cursor.execute(query,(airline_name))
#Get most popular cities over last year
    query = "SELECT airport.airport_city, count(ticket.ticket_id) as totalnum\
        FROM ticket NATURAL JOIN flight, airport \
        WHERE flight.airline_name = %s AND flight.arrival_airport =
airport.airport_name \
            AND DateDiff(CURDATE(), flight.arrival_time) <= 365 \
        GROUP BY airport.airport_city\
        ORDER BY totalnum DESC"
    cursor.execute(query,(airline_name))
```

10. Grant new permissions.

```
#Check if the staff belongs to the same Airline
    query = "SELECT * FROM airline_staff WHERE username = %s AND airline_name =
%s"
    cursor.execute(query,(staff_username,airline_name))
#If he/she is, then…
ins = "INSERT INTO permission VALUES (%s,%s)"
cursor.execute(ins,(staff_username,type))
```

11. Cancel permissions.

```
#Check if the staff belongs to the same Airline
   query = "SELECT * FROM airline_staff WHERE username = %s AND airline_name =
%s"
   cursor.execute(query,(staff_username,airline_name))
#If he/she is, then…
upd = "DELETE FROM permission WHERE username = %s AND permission_type = %s"
cursor.execute(upd,(staff_username,type))
```

12. Add booking agents.

```
ins = "INSERT INTO booking_agent_work_for VALUES (%s,%s)"
cursor.execute(ins,(agent_email,airline_name))
```

13. Remove booking agents.

```
upd = "DELETE FROM booking_agent_work_for where email = %s AND airline_name =
%s"
cursor.execute(upd,(agent_email,airline_name))
```