Public View:

1. Get airport and city
   1. Input is airport

DELIMITER //

CREATE PROCEDURE GetAirportWithCity()

BEGIN

SELECT airport\_city, airport\_name FROM `airport`;

END //

DELIMITER ;

* 1. Input is city

DELIMITER //

CREATE PROCEDURE GetUniqueAirportCity()

BEGIN

SELECT DISTINCT airport\_city FROM `airport`;

END //

DELIMITER ;

1. Get flights by location

SELECT airline\_name, flight\_num, departure\_airport, SRC.airport\_city, departure\_time, arrival\_airport, DST.airport\_city, arrival\_time, price, status, airplane\_id

FROM flight AS F JOIN airport AS SRC ON (F.departure\_airport = SRC.airport\_name) JOIN airport AS DST ON (F.arrival\_airport = DST.airport\_name)

WHERE DATE(F.departure\_time) = %s AND F.status = 'Upcoming' AND SRC.airport\_city = %s AND DST.airport\_city= %s AND (F.departure\_airport = %s OR %s = '') AND (F.arrival\_airport = %s OR %s = '')

ORDER BY departure\_time

1. Get flight status: Today or Specify a range of date

SELECT airline\_name, flight\_num, status, airport\_city,departure\_time

FROM flight JOIN airport ON (flight.arrival\_airport = airport.airport\_name)

WHERE (flight\_num = %s OR %s = '') AND (DATE(departure\_time) = %s OR %s = '') AND (DATE(arrival\_time) = %s OR %s = '')

1. Get upcoming flights

cursor = conn.cursor(prepared=True)

email = email.replace("\'", "\'\'")

query = """SELECT airline\_name, flight\_num, departure\_airport, SRC.airport\_city, departure\_time,arrival\_airport, DST.airport\_city, arrival\_time, price, status, airplane\_id

FROM flight AS F JOIN airport AS SRC ON (F.departure\_airport = SRC.airport\_name)JOIN airport AS DST ON (F.arrival\_airport = DST.airport\_name)

WHERE status = 'Upcoming' AND """

if identity == "airline\_staff":

query += """DATE(departure\_time) <= \'%s\' AND """ % next\_thirty\_days

query += """ airline\_name IN (

SELECT airline\_name

FROM airline\_staff

WHERE username = %s

)"""

elif identity == "customer":

query += """ flight\_num IN (

SELECT flight\_num

FROM purchases NATURAL JOIN ticket

WHERE customer\_email = %s

)"""

else:

query += """ flight\_num IN (

SELECT flight\_num

FROM purchases NATURAL JOIN ticket NATURAL JOIN booking\_agent

WHERE email = %s

)"""

# print("get upcoming flights query is:\n", query % email)

cursor.execute(query, (email,))

1. Get time flight

cursor = conn.cursor(prepared=True)

query = """SELECT airline\_name, flight\_num, departure\_airport, SRC.airport\_city, departure\_time, arrival\_airport, DST.airport\_city, arrival\_time, price, status, airplane\_id

FROM flight AS F JOIN airport AS SRC ON (F.departure\_airport = SRC.airport\_name)JOIN airport AS DST ON (F.arrival\_airport = DST.airport\_name)

WHERE

"""

if identity == "airline\_staff":

query += """ airline\_name IN (

SELECT airline\_name

FROM airline\_staff

WHERE username = %s

)"""

elif identity == "customer":

query += """ flight\_num IN (

SELECT flight\_num

FROM purchases NATURAL JOIN ticket

WHERE customer\_email = %s

)"""

else:

query += """ flight\_num IN (

SELECT flight\_num

FROM purchases NATURAL JOIN ticket NATURAL JOIN booking\_agent

WHERE email = %s

)"""

query += """ AND (DATE(departure\_time) >= %s OR %s = '') AND (DATE(departure\_time) <= %s OR %s = '')

AND (SRC.airport\_city = %s OR %s = '') AND (DST.airport\_city = %s OR %s = '')

AND (departure\_airport = %s OR %s = '') AND (arrival\_airport = %s OR %s = '')

ORDER BY departure\_time

"""

cursor.execute(query, (email, start\_date, start\_date, end\_date, end\_date, src\_city, src\_city, dst\_city, dst\_city, src\_airport, src\_airport, dst\_airport, dst\_airport))

data = cursor.fetchall()

cursor.close()

1. Get specific flights

cursor = conn.cursor(prepared=True)

query = """SELECT airline\_name, flight\_num, departure\_airport, SRC.airport\_city, departure\_time,

arrival\_airport, DST.airport\_city, arrival\_time, price, status, airplane\_id

FROM flight AS F JOIN airport AS SRC ON (F.departure\_airport = SRC.airport\_name)

JOIN airport AS DST ON (F.arrival\_airport = DST.airport\_name)

WHERE airline\_name = %s AND flight\_num = %s

"""

cursor.execute(query, (airline\_name, flight\_num))

1. Purchase tickets

cursor = conn.cursor(prepared=True)

query = """SELECT \*

FROM flight

WHERE airline\_name = %s AND flight\_num = %s AND status = 'Upcoming' AND departure\_time > %s"""

cursor.execute(query, (airline\_name, flight\_num, datetime.today().strftime("%Y-%m-%d")))

check\_flight = cursor.fetchall()

if not check\_flight:

cursor.close()

return False, "Don't cheat! Can't buy this flight!"

query = """SELECT COUNT(ticket\_id)

FROM ticket

WHERE flight\_num = %s"""

# print(query)

cursor.execute(query, (flight\_num,))

ticket\_num = cursor.fetchall()

ticket\_num = ticket\_num[0][0] if ticket\_num else 0

query = """SELECT seats

FROM airplane NATURAL JOIN flight

WHERE flight\_num = %s"""

# print(query)

cursor.execute(query, (flight\_num,))

seat\_num = cursor.fetchall()

seat\_num = seat\_num[0][0] if seat\_num else 0

# print(ticket\_num, seat\_num)

if ticket\_num >= seat\_num:

cursor.close()

return False, "No ticket! This flight is full!"

query = """SELECT \* FROM customer WHERE email = %s"""

cursor.execute(query, (customer\_email,))

data = cursor.fetchall()

if not data:

cursor.close()

return False, "Customer does not exists!"

ticket\_id = flight\_num[:2] + datetime.now().strftime("%Y%m%d%H%M%S")

today = datetime.today().strftime("%Y-%m-%d")

query = """INSERT INTO ticket

VALUES (%s, %s, %s)"""

cursor.execute(query, (ticket\_id, airline\_name, flight\_num))

# conn.commit()

if identity == "customer":

query = """INSERT INTO purchases

VALUES (%s, %s, NULL, %s)"""

t = (ticket\_id, customer\_email, today)

else:

query = """SELECT booking\_agent\_id

FROM booking\_agent

WHERE email = %s"""

cursor.execute(query, (agent\_email,))

agent\_id = cursor.fetchall()[0][0]

query = """INSERT INTO purchases

VALUES (%s, %s, %s, %s)"""

t = (ticket\_id, customer\_email, agent\_id, today)

cursor.execute(query, t)

conn.commit()

cursor.close()

Login:

1. Airline staff

SELECT password

FROM %s

WHERE username = %s

SELECT airline\_name

FROM airline\_staff

WHERE username = %s

1. Customer and booking agent

SELECT password

FROM %s

WHERE email = %s

Register:

1. Airline staff: check airline name exists; check email is unique;

SELECT airline\_name

FROM airline

WHERE airline\_name = %s

SELECT username FROM %s

WHERE username = %s

1. Booking agent: check agent id unique; check email is unique

SELECT email FROM %s

WHERE email = %s

SELECT email FROM %s

WHERE email = %s

1. Customer: check email is unique

SELECT email FROM %s

WHERE email = %s

cursor = conn.cursor(prepared=True)

query = """INSERT INTO %s""" % identity

if identity == "customer":

query += """ VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s) """

cursor.execute(query, (info["email"].replace("\'", "\'\'"), info["name"].replace("\'", "\'\'"),

generate\_password\_hash(info["password"], PASSWORD\_HASH), info["building\_number"],

info["street"].replace("\'", "\'\'"), info["city"].replace("\'", "\'\'"), info["state"],

info["phone\_number"], info["passport\_number"],

info["passport\_expiration"], info["passport\_country"], info["date\_of\_birth"]))

elif identity == "booking\_agent":

query += " VALUES (%s, %s, %s)"

cursor.execute(query, (info["email"].replace("\'", "\'\'"),

generate\_password\_hash(info["password"], PASSWORD\_HASH), info["booking\_agent\_id"]))

else:

query += """ VALUES (%s, %s, %s, %s, %s, %s)"""

cursor.execute(query, (info["email"].replace("\'", "\'\'"),

generate\_password\_hash(info["password"], PASSWORD\_HASH),

info["first\_name"].replace("\'", "\'\'"), info["last\_name"].replace("\'", "\'\'"),

info["date\_of\_birth"], info["airline\_name"]))

conn.commit()

cursor.close()

Customer

1. Track my spending
   1. Total

SELECT SUM(price)

FROM ticket NATURAL JOIN purchases NATURAL JOIN flight

WHERE customer\_email = %s AND purchase\_date BETWEEN %s AND %s

* 1. Specific time range

SELECT purchase\_date, price

FROM ticket NATURAL JOIN purchases NATURAL JOIN flight

WHERE customer\_email = %s AND purchase\_date BETWEEN %s AND %s

Booking agent

1. Get my commission

SELECT SUM(price) \* 0.1, COUNT(ticket\_id), SUM(price) \* 0.1 / COUNT(ticket\_id)

FROM ticket NATURAL JOIN purchases NATURAL JOIN booking\_agent NATURAL JOIN flight

WHERE email = %s AND (purchase\_date >= %s OR %s = '') AND (purchase\_date <= %s OR %s = '')

SELECT SUM(price) \* 0.1, COUNT(ticket\_id), SUM(price) \* 0.1 / COUNT(ticket\_id)

FROM ticket NATURAL JOIN purchases NATURAL JOIN booking\_agent NATURAL JOIN flight

WHERE email != %s AND (purchase\_date >= %s OR %s = '') AND (purchase\_date <= %s OR %s = '')

1. View top customers

CREATE OR REPLACE VIEW top\_customers\_ticket AS (

SELECT customer\_email, COUNT(ticket\_id) AS num\_of\_ticket

FROM ticket NATURAL JOIN purchases NATURAL JOIN booking\_agent

WHERE email = \'%s\' and purchase\_date >= \'%s\'

GROUP BY customer\_email

)

CREATE OR REPLACE VIEW top\_customers\_commission AS (

SELECT customer\_email, SUM(price) \* 0.1 AS amount\_of\_commission

FROM ticket NATURAL JOIN purchases NATURAL JOIN booking\_agent NATURAL JOIN flight

WHERE email = \'%s\' and purchase\_date >= \'%s\'

GROUP BY customer\_email

)

DELIMITER //

CREATE PROCEDURE GetTopFiveCustomerTicket()

BEGIN

SELECT \*

FROM top\_customers\_ticket

ORDER BY num\_of\_ticket DESC, customer\_email

LIMIT 5;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE GetTopFiveCustomerCommission()

BEGIN

SELECT \*

FROM top\_customers\_commission

ORDER BY amount\_of\_commission DESC, customer\_email

LIMIT 5;

END //

DELIMITER ;

Airline staff

1. Create new flights

SELECT flight\_num FROM flight WHERE flight\_num = %s

SELECT airport\_name FROM airport WHERE airport\_name = %s

SELECT airport\_name FROM airport WHERE airport\_name = %s

SELECT airplane\_id FROM airplane WHERE airplane\_id = %s

INSERT INTO flight

VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)

1. Change flight status

SELECT flight\_num, status

FROM flight

WHERE flight\_num = %s AND airline\_name = %s

UPDATE flight

SET status = %s

WHERE flight\_num = %s AND airline\_name = %s

1. Add airplane

SELECT \*

FROM airplane

WHERE airline\_name = %s AND airplane\_id = %s

INSERT INTO airplane

VALUES (%s, %s, %s)

SELECT \*

FROM airplane

WHERE airline\_name = %s

1. Add airport

SELECT \*

FROM airport

WHERE airport\_name = %s

INSERT INTO airport

VALUES (%s, %s)

1. View booking agent

CREATE OR REPLACE VIEW top\_agents\_ticket AS (

SELECT email, COUNT(ticket\_id) AS num\_of\_ticket

FROM booking\_agent NATURAL JOIN purchases NATURAL JOIN ticket NATURAL JOIN flight

WHERE purchase\_date >= \'%s\' AND airline\_name = \'%s\'

GROUP BY email

DELIMITER //

CREATE PROCEDURE GetTopFiveAgentsTicket()

BEGIN

SELECT \*

FROM top\_agents\_ticket

ORDER BY num\_of\_ticket DESC, email

LIMIT 5;

END //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE GetTopFiveAgentsCommission()

BEGIN

SELECT \*

FROM top\_agents\_commission

ORDER BY t1.amount\_of\_commission DESC, email

LIMIT 5;

END //

DELIMITER ;

CREATE OR REPLACE VIEW top\_agents\_commission AS (

SELECT email, SUM(price \* 0.1) AS amount\_of\_commission

FROM booking\_agent NATURAL JOIN purchases NATURAL JOIN ticket NATURAL JOIN flight

WHERE purchase\_date >= \'%s\' AND airline\_name = \'%s\'

GROUP BY email

)

1. View most frequent customers

SELECT customer\_email, COUNT(ticket\_id)

FROM purchases NATURAL JOIN ticket

WHERE purchase\_date BETWEEN %s AND %s AND airline\_name = %s

GROUP BY customer\_email

HAVING COUNT(ticket\_id) >= ALL (SELECT COUNT(ticket\_id)

FROM purchases NATURAL JOIN ticket

WHERE purchase\_date BETWEEN %s AND %s AND airline\_name = %s

GROUP BY customer\_email)

ORDER BY customer\_email

SELECT COUNT(ticket\_id)

FROM purchases NATURAL JOIN ticket

WHERE purchase\_date BETWEEN %s AND %s AND airline\_name = %s

SELECT DISTINCT flight\_num, departure\_airport, departure\_time, arrival\_airport, arrival\_time, status, airplane\_id

FROM purchases NATURAL JOIN ticket NATURAL JOIN flight

WHERE customer\_email = %s AND airline\_name = %s

1. View my flight

SELECT DISTINCT email, name, phone\_number, passport\_number, date\_of\_birth

FROM ticket NATURAL JOIN purchases JOIN customer ON (customer.email = purchases.customer\_email)

WHERE airline\_name = %s AND flight\_num = %s

1. View reports

SELECT ticket\_id, purchase\_date

FROM purchases NATURAL JOIN ticket

WHERE airline\_name = %s AND purchase\_date BETWEEN %s AND %s

1. Comparison of revenue earned

SELECT SUM(price)

FROM ticket NATURAL JOIN purchases NATURAL JOIN flight

WHERE airline\_name = %s AND purchase\_date BETWEEN %s AND %s

if t == "direct":

query += """ AND booking\_agent\_id IS NULL"""

else:

query += """ AND booking\_agent\_id IS NOT NULL"""

1. Get top destinations

CREATE OR REPLACE VIEW top\_destinations AS (

SELECT DST.airport\_city AS dst, COUNT(ticket\_id) AS num\_of\_purchase

FROM flight AS F JOIN airport AS DST ON (F.arrival\_airport = DST.airport\_name)

NATURAL JOIN purchases NATURAL JOIN ticket

WHERE airline\_name = \'%s\' AND purchase\_date BETWEEN \'%s\' AND \'%s\'

GROUP BY dst

)

DELIMITER //

CREATE PROCEDURE GetTopThreeDestination()

BEGIN

SELECT \*

FROM top\_destinations

ORDER BY num\_of\_purchase DESC, dst

LIMIT 3;

END //

DELIMITER ;

EXTRA: change info

cursor = conn.cursor()

query = """SELECT \* FROM %s""" % identity

if identity == "airline\_staff":

query += """ WHERE username = %s"""

else:

query += """ WHERE email = %s"""

cursor.execute(query, (email,))

data = list(cursor.fetchall()[0])

if identity == "customer":

data.pop(2)

else:

data.pop(1)

cursor.close()

cursor = conn.cursor(prepared=True)

if info["email"] != old\_email:

query = """SELECT \* FROM %s""" % identity

if identity == "airline\_staff":

query += """ WHERE username = %s"""

else:

query += """ WHERE email = %s"""

cursor.execute(query, (info["email"],))

data = cursor.fetchall()

if data:

cursor.close()

return False, "Email has already been used!"

if old\_agent\_id:

if info["booking\_agent\_id"] != old\_agent\_id:

query = """SELECT \* FROM %s""" % identity

query += """ WHERE booking\_agent\_id = %s"""

cursor.execute(query, (info["booking\_agent\_id"],))

data = cursor.fetchall()

if data:

cursor.close()

return False, "Agent id has already been used!"

query = """UPDATE %s""" % identity

if identity == "airline\_staff":

query += """ SET username = %s, first\_name = %s, last\_name = %s

WHERE username = %s

"""

cursor.execute(query, (info["email"], info["first\_name"], info["last\_name"], old\_email))

elif identity == "booking\_agent":

query += """ SET email = %s, booking\_agent\_id = %s

WHERE email = %s

"""

cursor.execute(query, (info["email"], info["booking\_agent\_id"], old\_email))

else:

query += """ SET email = %s, name = %s, building\_number = %s, street = %s, city = %s, state = %s, phone\_number = %s, passport\_number = %s, passport\_expiration = %s, passport\_country = %s

WHERE email = %s

"""

cursor.execute(query, (info["email"], info["name"], info["building\_number"], info["street"], info["city"], info["state"],

info["phone\_number"], info["passport\_number"], info["passport\_expiration"], info["passport\_country"], old\_email))

conn.commit()

cursor.close()