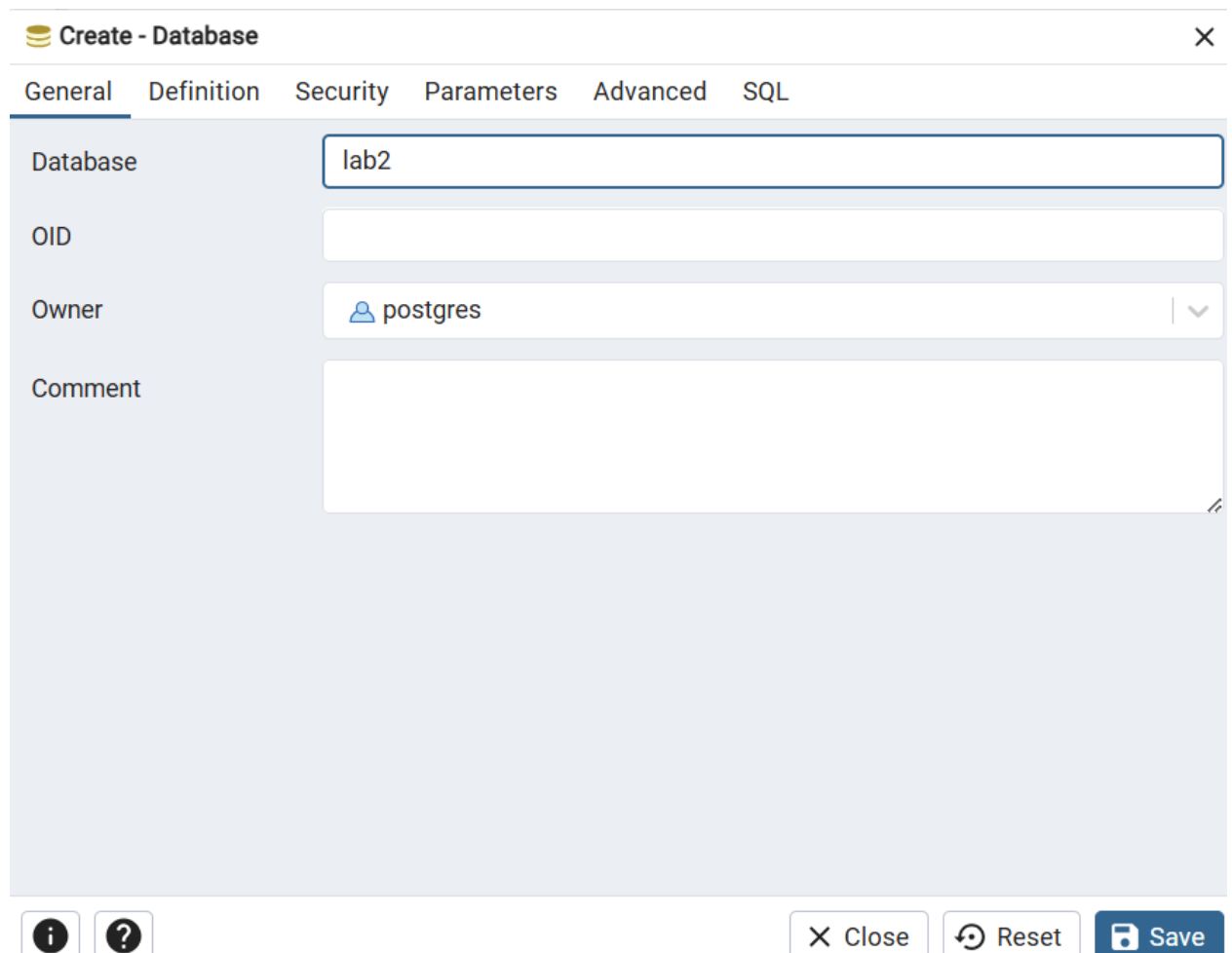
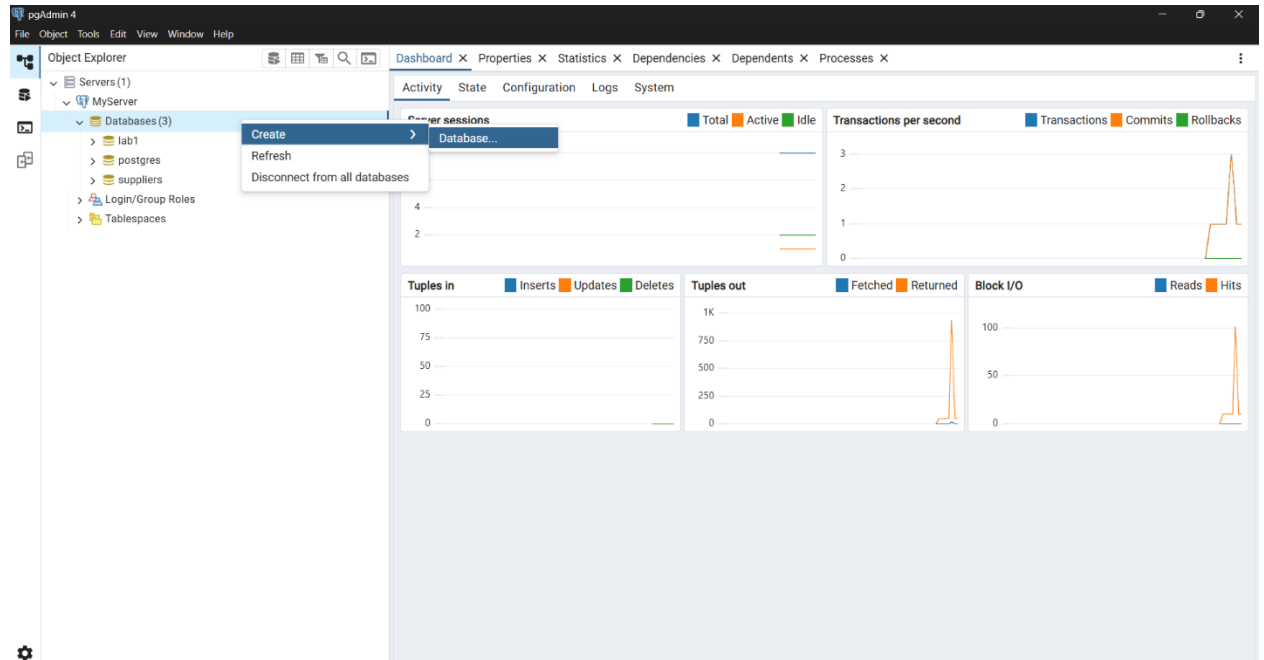


# Lab 2 Abdykamat Adilet

## DDL

1) In the first one, i opened the lab2 database.



## 2) Creating tables

**Then I created all the tables specified in the assignment.**

```
CREATE TABLE Airline_info (  
    airline_id INT PRIMARY KEY,  
    airline_code VARCHAR(30) NOT NULL,  
    airline_name VARCHAR(50) NOT NULL,  
    airline_country VARCHAR(50) NOT NULL,  
    created_at TIMESTAMP NOT NULL,  
    updated_at TIMESTAMP NOT NULL,  
    info VARCHAR(50) NOT NULL  
);
```

```
CREATE TABLE Airport (  
    airport_id INT PRIMARY KEY,  
    airport_name VARCHAR(50) NOT NULL,  
    country VARCHAR(50) NOT NULL,  
    state VARCHAR(50) NOT NULL,  
    city VARCHAR(50) NOT NULL,  
    created_at TIMESTAMP NOT NULL,  
    updated_at TIMESTAMP NOT NULL  
);
```

```
CREATE TABLE Baggage_check (  
    baggage_check_id INT PRIMARY KEY,  
    check_result VARCHAR(50) NOT NULL,  
    created_at TIMESTAMP NOT NULL,  
    updated_at TIMESTAMP NOT NULL,  
    booking_id INT NOT NULL,  
    passenger_id INT NOT NULL  
);
```

```
CREATE TABLE Baggage (  
    baggage_id INT PRIMARY KEY,  
    weight_in_kg DECIMAL(4,2) NOT NULL,  
    created_at TIMESTAMP NOT NULL,  
    updated_at TIMESTAMP NOT NULL,  
    booking_id INT NOT NULL  
);
```

```
CREATE TABLE Boarding_pass (  
    boarding_pass_id INT PRIMARY KEY,  
    booking_id INT NOT NULL,  
    seat VARCHAR(50) NOT NULL,  
    boarding_time TIMESTAMP NOT NULL,  
    created_at TIMESTAMP NOT NULL,  
    updated_at TIMESTAMP NOT NULL  
);
```

```
CREATE TABLE Booking_flight (  
    booking_flight_id INT PRIMARY KEY,  
    booking_id INT NOT NULL,  
    flight_id INT NOT NULL,  
    created_at TIMESTAMP NOT NULL,  
    updated_at TIMESTAMP NOT NULL  
);
```

```
CREATE TABLE Booking (  
    booking_id INT PRIMARY KEY,  
    flight_id INT NOT NULL,  
    passenger_id INT NOT NULL,  
    booking_platform VARCHAR(50) NOT NULL,  
    created_at TIMESTAMP NOT NULL,  
    updated_at TIMESTAMP NOT NULL,
```

```
status VARCHAR(50) NOT NULL,  
price DECIMAL(7,2) NOT NULL  
);
```

```
CREATE TABLE Flights (  
    flight_id INT PRIMARY KEY,  
    sch_departure_time TIMESTAMP NOT NULL,  
    sch_arrival_time TIMESTAMP NOT NULL,  
    departing_airport_id INT NOT NULL,  
    arriving_airport_id INT NOT NULL,  
    departing_gate VARCHAR(50) NOT NULL,  
    arriving_gate VARCHAR(50) NOT NULL,  
    airline_id INT NOT NULL,  
    act_departure_time TIMESTAMP NOT NULL,  
    act_arrival_time TIMESTAMP NOT NULL,  
    created_at TIMESTAMP NOT NULL,  
    updated_at TIMESTAMP NOT NULL  
);
```

```
CREATE TABLE Passengers (  
    passenger_id INT PRIMARY KEY,  
    first_name VARCHAR(50) NOT NULL,  
    last_name VARCHAR(50) NOT NULL,  
    date_of_birth DATE NOT NULL,  
    gender VARCHAR(50) NOT NULL,  
    country_of_citizenship VARCHAR(50) NOT NULL,  
    country_of_residence VARCHAR(50) NOT NULL,  
    passport_number VARCHAR(20) NOT NULL,  
    created_at TIMESTAMP NOT NULL,  
    updated_at TIMESTAMP NOT NULL  
);
```

The screenshot displays the pgAdmin 4 web interface. On the left, the Object Explorer shows a tree structure of the database 'lab2/postgres@MyServer', with 'Databases (4)' expanded to show 'lab1' and 'lab2'. The 'lab2' database is selected. The main pane shows the SQL query editor with the following SQL code:

```

80  gender VARCHAR(20) NOT NULL,
81  country_of_citizenship VARCHAR(50) NOT NULL,
82  country_of_residence VARCHAR(50) NOT NULL,
83  passport_number VARCHAR(20) NOT NULL,
84  created_at TIMESTAMP NOT NULL,
85  updated_at TIMESTAMP NOT NULL
86  );
87
88  CREATE TABLE Security_check (
89    security_check_id INT PRIMARY KEY,
90    check_result VARCHAR(20) NOT NULL,
91    created_at TIMESTAMP NOT NULL,
92    updated_at TIMESTAMP NOT NULL,
93    passenger_id INT NOT NULL
94  );

```

Below the query editor, the 'Data Output' tab is active, showing the message: 'Query returned successfully in 108 msec.' The status bar at the bottom indicates 'Total rows: Query complete 00:00:00.108' and 'CRLF Ln 100, Col 3'.

Tables (10)

- > airline\_info
- > airport
- > baggage
- > baggage\_check
- > boarding\_pass
- > booking
- > booking\_flight
- > flights
- > passengers
- > security\_check

### 3) Rename the airline\_info table to airline

I executed the following query:

```
ALTER TABLE airline_info RENAME TO airline;
```



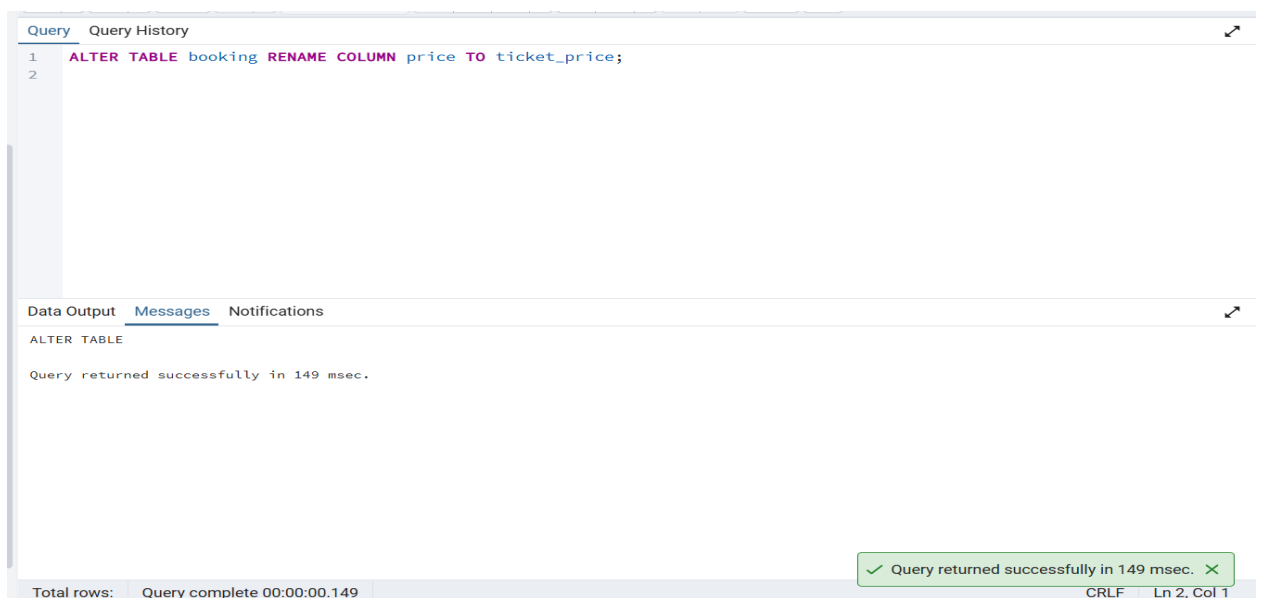
The screenshot shows a database query interface with two tabs: 'Query' and 'Query History'. The 'Query' tab is active, displaying the SQL query: `1 ALTER TABLE airline_info RENAME TO airline;`. Below the query, there are three tabs: 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, showing the text: `ALTER TABLE` and `Query returned successfully in 100 msec.`. At the bottom right, a green status bar indicates: `✓ Query returned successfully in 100 msec. ✕`.

The table was successfully renamed.

### 4) Rename the column price to ticket\_price in the booking table

Query:

```
ALTER TABLE booking RENAME COLUMN price TO ticket_price;
```



The screenshot shows a database query interface with two tabs: 'Query' and 'Query History'. The 'Query' tab is active, displaying the SQL query: `1 ALTER TABLE booking RENAME COLUMN price TO ticket_price;` and `2`. Below the query, there are three tabs: 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is active, showing the text: `ALTER TABLE` and `Query returned successfully in 149 msec.`. At the bottom right, a green status bar indicates: `✓ Query returned successfully in 149 msec. ✕`. The bottom status bar shows: `Total rows: Query complete 00:00:00.149` and `CRLF Ln 2, Col 1`.

The column was successfully renamed.

## 5) Change the data type of departing\_gate from varchar(50) to text

Query:

```
ALTER TABLE flights ALTER COLUMN departing_gate TYPE text;
```

The screenshot shows a PostgreSQL query editor interface. At the top, the connection is 'lab2/postgres@MyServer'. The query editor contains the following SQL statement:

```
1 ALTER TABLE flights ALTER COLUMN departing_gate TYPE text;
```

Below the query editor, the 'Messages' tab is active, displaying the following message:

```
ALTER TABLE  
  
Query returned successfully in 96 msec.
```

A green status bar at the bottom right indicates: '✓ Query returned successfully in 96 msec. X'. The bottom status bar shows 'Total rows: Query complete 00:00:00.096' and 'CRLF Ln 1 Col 50'.

The data type was successfully changed.

## 6) Drop the column info (varchar(50)) from the airline table

Query:

```
ALTER TABLE airline DROP COLUMN info;
```

The screenshot shows a PostgreSQL query editor interface. At the top, the connection is 'lab2/postgres@MyServer'. The query editor contains the following SQL statement:

```
1 ALTER TABLE airline DROP COLUMN info;
```

Below the query editor, the 'Messages' tab is active, displaying the following message:

```
ALTER TABLE  
  
Query returned successfully in 107 msec.
```

A green status bar at the bottom right indicates: '✓ Query returned successfully in 107 msec. X'. The bottom status bar shows 'Total rows: Query complete 00:00:00.107' and 'CRLF Ln 1 Col 38'.

## 7) Relationships between tables

I created relationships using foreign keys:

Passenger with Security\_check, Booking, Baggage\_check

```
ALTER TABLE security_check ADD CONSTRAINT fk_passenger FOREIGN KEY (passenger_id) REFERENCES passengers(passenger_id);
```

```
ALTER TABLE booking ADD CONSTRAINT fk_passenger_booking FOREIGN KEY (passenger_id) REFERENCES passengers(passenger_id);
```

```
ALTER TABLE baggage_check ADD CONSTRAINT fk_passenger_baggage FOREIGN KEY (passenger_id) REFERENCES passengers(passenger_id);
```

Booking with Baggage\_check, Baggage, Boarding\_pass, Booking\_flight

```
ALTER TABLE baggage_check ADD CONSTRAINT fk_booking_baggagecheck FOREIGN KEY (booking_id) REFERENCES booking(booking_id);
```

```
ALTER TABLE baggage ADD CONSTRAINT fk_booking_baggage FOREIGN KEY (booking_id) REFERENCES booking(booking_id);
```

```
ALTER TABLE boarding_pass ADD CONSTRAINT fk_booking_boarding FOREIGN KEY (booking_id) REFERENCES booking(booking_id);
```

```
ALTER TABLE booking_flight ADD CONSTRAINT fk_booking_flight FOREIGN KEY (booking_id) REFERENCES booking(booking_id);
```

Flights with Booking\_flight

```
ALTER TABLE booking_flight ADD CONSTRAINT fk_flight FOREIGN KEY (flight_id) REFERENCES flights(flight_id);
```

Airport with Flights

```
ALTER TABLE flights ADD CONSTRAINT fk_depart_airport FOREIGN KEY (departing_airport_id) REFERENCES airport(airport_id);
```

```
ALTER TABLE flights ADD CONSTRAINT fk_arrive_airport FOREIGN KEY (arriving_airport_id) REFERENCES airport(airport_id);
```

Airline with Flights

```
ALTER TABLE flights ADD CONSTRAINT fk_airline FOREIGN KEY (airline_id) REFERENCES airline(airline_id);
```



```
7 ALTER TABLE baggage_check ADD CONSTRAINT booking_baggagecheck FOREIGN KEY (booking_id) REFERENCES booking(booking_id);
8 ALTER TABLE baggage ADD CONSTRAINT booking_baggage FOREIGN KEY (booking_id) REFERENCES booking(booking_id);
9 ALTER TABLE boarding_pass ADD CONSTRAINT fk_booking_boarding FOREIGN KEY (booking_id) REFERENCES booking(booking_id);
10 ALTER TABLE booking_flight ADD CONSTRAINT fk_booking_flight FOREIGN KEY (booking_id) REFERENCES booking(booking_id);
11
12 -- Flights with Booking_flight
13 ALTER TABLE booking_flight ADD CONSTRAINT fk_flight FOREIGN KEY (flight_id) REFERENCES flights(flight_id);
14
15 -- Airport with Flights
16 ALTER TABLE flights ADD CONSTRAINT fk_depart_airport FOREIGN KEY (departing_airport_id) REFERENCES airport(airport_id);
17 ALTER TABLE flights ADD CONSTRAINT fk_arrive_airport FOREIGN KEY (arriving_airport_id) REFERENCES airport(airport_id);
18
19 -- Airline with Flights
20 ALTER TABLE flights ADD CONSTRAINT fk_airline FOREIGN KEY (airline_id) REFERENCES airline(airline_id);
```

ALTER TABLE

Query returned successfully in 129 msec.

✓ Query returned successfully in 129 msec. ✕

## DML

### 1. Generate and insert 200 rows into airport

Query:

```
INSERT INTO airport (airport_id, airport_name, country, state, city, created_at, updated_at)
```

```
SELECT i,
```

```
    'Airport_' || i,
```

```
    'Country_' || (i % 50),
```

```
    'State_' || (i % 20),
```

```
    'City_' || (i % 100),
```

```
    NOW(),
```

```
    NOW()
```

```
FROM generate_series(1,200) AS i;
```

lab2/postgres@MyServer

Query Query History

```
1 INSERT INTO airport (airport_id, airport_name, country, state, city, created_at, updated_at)
2 SELECT i,
3     'Airport_' || i,
4     'Country_' || (i % 50),
5     'State_' || (i % 20),
6     'City_' || (i % 100),
7     NOW(),
8     NOW()
9 FROM generate_series(1,200) AS i;
```

Data Output Messages Notifications

INSERT 0 200

Query returned successfully in 107 msec.

✓ Query returned successfully in 107 msec. ✕

Total rows: Query complete 00:00:00.107 CRLF Ln 9, Col 24

## 2. Add a new airline named "KazAir" based in "Kazakhstan"

INSERT INTO airline (airline\_id, airline\_code, airline\_name, airline\_country, created\_at, updated\_at)

VALUES (1, 'KZ', 'KazAir', 'Kazakhstan', NOW(), NOW());

lab2/postgres@MyServer

Query Query History

```
1 INSERT INTO airline (airline_id, airline_code, airline_name, airline_country, created_at, updated_at)
2 VALUES (1, 'KZ', 'KazAir', 'Kazakhstan', NOW(), NOW());
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 95 msec.

✓ Query returned successfully in 95 msec. ✕

Total rows: Query complete 00:00:00.095 CRLF Ln 2, Col 56

### 3. Update the airline country of "KazAir" to "Turkey"

UPDATE airline

SET airline\_country = 'Turkey'

WHERE airline\_name = 'KazAir';

The screenshot shows a SQL IDE interface. The top pane, titled 'Query', contains the following SQL statement:

```
1 UPDATE airline
2 SET airline_country = 'Turkey'
3 WHERE airline_name = 'KazAir';
```

The bottom pane, titled 'Messages', shows the execution result:

```
UPDATE 1
Query returned successfully in 95 msec.
```

A green status bar at the bottom right indicates: '✓ Query returned successfully in 95 msec. ✕'. The status bar also shows 'Total rows: Query complete 00:00:00.095' and 'CRLF Ln 3, Col 31'.

### 4. Add three airlines at once

INSERT INTO airline (airline\_id, airline\_code, airline\_name, airline\_country, created\_at, updated\_at)

VALUES

(2, 'AE', 'AirEasy', 'France', NOW(), NOW()),

(3, 'FH', 'FlyHigh', 'Brazil', NOW(), NOW()),

(4, 'FF', 'FlyFly', 'Poland', NOW(), NOW());

The screenshot shows a SQL IDE interface. The top pane, titled 'Query', contains the following SQL statement:

```
1 INSERT INTO airline (airline_id, airline_code, airline_name, airline_country, created_at, updated_at)
2 VALUES
3 (2, 'AE', 'AirEasy', 'France', NOW(), NOW()),
4 (3, 'FH', 'FlyHigh', 'Brazil', NOW(), NOW()),
5 (4, 'FF', 'FlyFly', 'Poland', NOW(), NOW());
```

The bottom pane, titled 'Messages', shows the execution result:

```
INSERT 6 3
Query returned successfully in 104 msec.
```

A green status bar at the bottom right indicates: '✓ Query returned successfully in 104 msec. ✕'. The status bar also shows 'Total rows: Query complete 00:00:00.104' and 'CRLF Ln 5 Col 45'.

## 5. Delete all flights arriving in the year 2024

DELETE FROM flights

WHERE EXTRACT(YEAR FROM sch\_arrival\_time) = 2024;

The screenshot displays a SQL query execution window. The top pane, titled 'Query', shows the following SQL statement:

```
1 DELETE FROM flights
2 WHERE EXTRACT(YEAR FROM sch_arrival_time) = 2024;
```

The bottom pane, titled 'Messages', shows the execution results:

```
DELETE 0
Query returned successfully in 84 msec.
```

A green status bar at the bottom right indicates: '✓ Query returned successfully in 84 msec. ✕'. The bottom status bar shows 'Total rows: Query complete 00:00:00.084 CRLF Ln 2, Col 50'.

## 6. Increase the price of all tickets in booking by 15%

UPDATE booking

SET ticket\_price = ticket\_price \* 1.15;

The screenshot displays a SQL query execution window. The top pane, titled 'Query', shows the following SQL statement:

```
1 UPDATE booking
2 SET ticket_price = ticket_price * 1.15;
3
```

The bottom pane, titled 'Messages', shows the execution results:

```
UPDATE 0
Query returned successfully in 94 msec.
```

A green status bar at the bottom right indicates: '✓ Query returned successfully in 94 msec. ✕'. The bottom status bar shows 'Total rows: Query complete 00:00:00.094 CRLF Ln 1, Col 1'.

## 7. Delete all tickets where the price is less than 10000

DELETE FROM booking

WHERE ticket\_price < 10000;

Query

Query History

↗

1

▼

DELETE FROM booking

2

WHERE ticket\_price < 10000;

Data Output

Messages

Notifications

↗

DELETE 0

Query returned successfully in 109 msec.

✓ Query returned successfully in 109 msec. ✕

Total rows: Querv complete 00:00:00.109

CRLF Ln 2, Col 28