

The screenshot displays a database management tool interface. On the left, the 'Database Explorer' pane shows a tree structure with 'postgres@localhost' expanded, then 'airport\_lab20', 'public', and 'tables' expanded, with 'airline\_info' selected. Below this, the 'Services' pane shows a list of services: 'Database', 'postgres@localhost', 'default' (105 ms), and 'console' (22 ms). The main area is a 'console' window with a SQL editor. The SQL code being entered is: 

```
CREATE TABLE Airline_info (  
    airline_id INT PRIMARY KEY NOT NULL,  
    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  
);
```

 The console output shows the command being executed: 

```
[2025-09-24 08:31:52] airport_lab20.public> CREATE TABLE Airline_info (  
    airline_id INT PRIMARY KEY NOT NULL,  
    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  
)  
[2025-09-24 08:31:52] completed in 5 ms
```

 The bottom status bar indicates the current path: 'Database > postgres@localhost > airport\_lab20 > public > tables > airline\_info' and shows settings: '9:3 CRLF UTF-8 4 spaces'.

## DDL

### 1. Create following tables with corresponding attributes:

**Airline\_info:** airline\_id (int), airline\_code (varchar (30)), airline\_name (varchar(50)), airline\_country (varchar(50)), created\_at (timestamp), updated\_at (timestamp), info (varchar(50)) ;

On this step I created all required tables such as airline\_info, airport, passengers, flights, booking and others. For each table I defined primary keys and set all attributes with the NOT NULL constraint.



The screenshot displays a database management tool interface. The top section, 'Database Explorer', shows a tree view of the database structure: 'postgres@localhost' > 'airport\_lab20' > 'public' > 'tables' > 'airport'. The 'airport' table is selected. The 'console' tab is active, showing the SQL command to create the 'airport' table:

```
1 CREATE TABLE airport (  
2     airport_id INT PRIMARY KEY,  
3     airport_name VARCHAR(50) NOT NULL,  
4     country VARCHAR(50) NOT NULL,  
5     state VARCHAR(50) NOT NULL,  
6     city VARCHAR(50) NOT NULL,  
7     created_at TIMESTAMP NOT NULL,  
8     updated_at TIMESTAMP NOT NULL  
9 );
```

The bottom section, 'Services', shows a list of services: 'default' (105 ms) and 'console' (49 ms). The 'console' service is selected, and the SQL command to create the 'Airline\_info' table is shown:

```
[2025-09-24 08:31:52] airport_lab20.public> CREATE TABLE Airline_info (  
    airline_id INT PRIMARY KEY NOT NULL,  
    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  
);
```

The command is completed in 5 ms. The bottom status bar shows the current path: 'Database > postgres@localhost > airport\_lab20 > public > tables > airport' and the encoding settings: '9:3 CRLF UTF-8 4 spaces'.

## DDL

### 1. Create following tables with corresponding attributes:

**Airport:** airport\_id(int), airport\_name (varchar(50)),country (varchar(50)), state (varchar(50)), city(varchar(50)), created\_at(timestamp), updated\_at(timestamp) ;

On this step I created all required tables such as airline\_info, airport, passengers, flights, booking and others. For each table I defined primary keys and set all attributes with the NOT NULL constraint.



The screenshot displays a database IDE interface. On the left, the 'Database Explorer' pane shows a tree structure with 'postgres@localhost' expanded, containing 'airport\_lab20' (1 of 3), which has a 'public' schema. Inside 'public', there is a 'tables' folder (3) containing 'airline\_info', 'airport', and 'passengers' (highlighted). Below this are 'Database Objects', 'postgres' (1 of 3), and 'Server Objects'. The main editor area is titled 'console' and shows a SQL query to create a table named 'passengers'. The query is as follows:

```
1 CREATE TABLE passengers(  
2     passenger_id INT PRIMARY KEY,  
3     first_name VARCHAR(50) NOT NULL,  
4     last_name VARCHAR(50) NOT NULL,  
5     date_of_birth DATE NOT NULL,  
6     gender VARCHAR(50) NOT NULL,  
7     country_of_citizenship VARCHAR(50) NOT NULL,  
8     country_of_residence VARCHAR(50) NOT NULL,  
9     passport_number VARCHAR(20) NOT NULL,  
10    created_at TIMESTAMP NOT NULL,  
11    updated_at TIMESTAMP NOT NULL  
12 );
```

Below the query editor, the 'Services' pane shows a list of services. Under 'Database', there is 'postgres@localhost' with two connections: 'default' (105 ms) and 'console' (76 ms, highlighted). The bottom status bar indicates the current path: 'Database > postgres@localhost > airport\_lab20 > public > tables > passengers'. The status bar also shows '12:3 CRLF UTF-8 4 spaces' and a bell icon.

## DDL

### 1. Create following tables with corresponding attributes:

**Passengers:** passenger\_id(int), first\_name(varchar(50)), last\_name(varchar(50)), date\_of\_birth(date), gender(varchar(50)), country\_of\_citizenship(varchar(50)), country\_of\_residence(varchar(50)), passport\_number(varchar(20)), created\_at(timestamp), updated\_at(timestamp) ;

On this step I created all required tables such as airline\_info, airport, passengers, flights, booking and others. For each table I defined primary keys and set all attributes with the NOT NULL constraint.



The screenshot displays a database management tool interface. On the left, the 'Database Explorer' pane shows a tree structure: 'postgres@localhost' (2) > 'airport\_lab20' (1 of 3) > 'public' > 'tables' (4). The 'tables' folder is expanded, showing 'airline\_info', 'airport', 'flights' (selected), and 'passengers'. Below this, 'Database Objects', 'postgres' (1 of 3), and 'Server Objects' are listed. The main editor area, titled 'console', contains a SQL script to create a table named 'flights'. The script is as follows:

```
1 CREATE TABLE flights(  
2     flight_id INT PRIMARY KEY,  
3     sch_departure_time TIMESTAMP NOT NULL,  
4     sch_arrival_time TIMESTAMP NOT NULL,  
5     departing_airport_id INT NOT NULL,  
6     arriving_airport_id INT NOT NULL,  
7     departing_gate TEXT NOT NULL,  
8     arriving_gate VARCHAR(50) NOT NULL,  
9     airline_id INT NOT NULL,  
10    act_departure_time TIMESTAMP NOT NULL,  
11    act_arrival_time TIMESTAMP NOT NULL,  
12    created_at TIMESTAMP NOT NULL,  
13    updated_at TIMESTAMP NOT NULL  
14 );
```

The script is executed, and the results are shown in the bottom right pane. The results display the table structure for 'flights':

```
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  
)
```

Below the results, a status message indicates: '[2025-09-24 08:44:00] completed in 9 ms'. The bottom status bar shows the current path: 'Database > postgres@localhost > airport\_lab20 > public > tables > flights', along with settings: '5:39 CRLF UTF-8 4 spaces'.

## DDL

### 1. Create following tables with corresponding attributes:

**Flights:** flight\_id(int), sch\_departure\_time(timestamp), sch\_arrival\_time(timestamp), departing\_airport\_id(int), arriving\_airport\_id(int), departing\_gate(varchar(50)), arriving\_gate(varchar(50)), airline\_id(int), act\_departure\_time(timestamp), act\_arrival\_time(timestamp), created\_at(timestamp), updated\_at(timestamp) ;

On this step I created all required tables such as airline\_info, airport, passengers, flights, booking and others. For each table I defined primary keys and set all attributes with the NOT NULL constraint.



The screenshot displays a database management tool interface. On the left, the 'Database Explorer' pane shows a tree structure with 'postgres@localhost' expanded, then 'airport\_lab20', and finally 'public' expanded to show a list of tables: 'airline\_info', 'airport', 'booking' (highlighted), 'flights', and 'passengers'. Below this is the 'Services' pane, which shows a list of database connections: 'default' (105 ms) and 'console' (19 ms, highlighted). The main area is a 'console' window with a toolbar at the top containing icons for running queries, saving, and other functions. The console shows a SQL command being executed: `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, ticket_price DECIMAL(7,2) NOT NULL );`. The command is numbered 1 to 10. Below the command, the execution result is shown: `[2025-09-24 08:45:22] airport_lab20.public> CREATE TABLE booking( ... )` followed by `[2025-09-24 08:45:22] completed in 5 ms`. The bottom status bar indicates the current settings: 'Database Consoles > postgres@localhost > console', '10:3', 'CRLF', 'UTF-8', '4 spaces', and a notification icon.

Database Explorer

postgres@localhost 2

airport\_lab20 1 of 3

public

tables 5

- airline\_info
- airport
- booking
- flights
- passengers

Database Objects

Services All Services

Database

- postgres@localhost
  - default 105 ms
  - console 19 ms

console

```
1 CREATE TABLE booking(  
2     booking_id INT PRIMARY KEY,  
3     flight_id INT NOT NULL,  
4     passenger_id INT NOT NULL,  
5     booking_platform VARCHAR(50) NOT NULL,  
6     created_at TIMESTAMP NOT NULL,  
7     updated_at TIMESTAMP NOT NULL,  
8     status VARCHAR(50) NOT NULL,  
9     ticket_price DECIMAL(7,2) NOT NULL  
10 );
```

[2025-09-24 08:45:22] airport\_lab20.public> 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,  
 ticket\_price DECIMAL(7,2) NOT NULL  
 )

[2025-09-24 08:45:22] completed in 5 ms

Database Consoles > postgres@localhost > console

10:3 CRLF UTF-8 4 spaces

## DDL

### 1. Create following tables with corresponding attributes:

**Booking:** booking\_id(int), flight\_id(int), passenger\_id(int), booking\_platform(varchar(50)), created\_at(timestamp), updated\_at(timestamp), status(varchar(50)), price(decimal(7,2));

On this step I created all required tables such as airline\_info, airport, passengers, flights, booking and others. For each table I defined primary keys and set all attributes with the NOT NULL constraint.



The screenshot displays a database management tool interface. On the left, the 'Database Explorer' pane shows a tree structure: 'postgres@localhost' (2) > 'airport\_lab20' (1 of 3) > 'public' > 'tables' (6). The 'tables' folder is expanded, showing 'airline\_info', 'airport', 'baggage' (selected), 'booking', 'flights', and 'passengers'. Below this is the 'Services' pane, showing 'Database' > 'postgres@localhost' > 'default' (105 ms) and 'console' (14 ms, selected). The main area is a 'console' window with a toolbar and a dropdown menu set to 'airport\_lab20.public'. The console shows the execution of a SQL command to create a table:

```
1 ✓ CREATE TABLE baggage(  
2     baggage_id INT PRIMARY KEY,  
3     weight_in_kg DECIMAL(4,2) NOT NULL,  
4     created_at TIMESTAMP NOT NULL,  
5     updated_at TIMESTAMP NOT NULL,  
6     booking_id INT NOT NULL  
7 );
```

The console output shows the command was completed in 5 ms. Below this, the command is repeated with the schema details:

```
[2025-09-24 08:46:32] airport_lab20.public> 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  
)
```

The command was completed in 4 ms. The bottom status bar shows the breadcrumb: 'Database > postgres@localhost > airport\_lab20 > public > tables > baggage', along with settings: '7:3 CRLF UTF-8 4 spaces'.

## DDL

### 1. Create following tables with corresponding attributes:

**Baggage:** baggage\_id(int), weight\_in\_kg (decimal(4,2)), created\_at(timestamp, updated\_at(timestamp) , booking\_id(int);

On this step I created all required tables such as airline\_info, airport, passengers, flights, booking and others. For each table I defined primary keys and set all attributes with the NOT NULL constraint.



The screenshot displays a database management tool interface. On the left, the 'Database Explorer' pane shows a tree structure with 'postgres@localhost' expanded, then 'airport\_lab20', and finally 'public' and 'tables'. The 'baggage\_check' table is highlighted. Below this, the 'Services' pane shows a list of services, with 'console' selected, indicating a 17 ms execution time. The main editor area, titled 'console', shows a SQL query being executed: `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 );`. The query is successful, as indicated by a green checkmark. Below the query, the execution log shows: `[2025-09-24 08:47:29] airport_lab20.public> 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 ); [2025-09-24 08:47:29] completed in 4 ms`. The bottom status bar shows the current path: 'Database > postgres@localhost > airport\_lab20 > public > tables > baggage\_check', along with settings like '8:3', 'CRLF', 'UTF-8', and '4 spaces'.

Database Explorer

console

```
1 ✓ CREATE TABLE baggage_check(  
2     baggage_check_id INT PRIMARY KEY,  
3     check_result VARCHAR(50) NOT NULL,  
4     created_at TIMESTAMP NOT NULL,  
5     updated_at TIMESTAMP NOT NULL,  
6     booking_id INT NOT NULL,  
7     passenger_id INT NOT NULL  
8 );
```

Services

Database

postgres@localhost

default 105 ms

console 17 ms

[2025-09-24 08:47:29] airport\_lab20.public> 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  
)

[2025-09-24 08:47:29] completed in 4 ms

Database > postgres@localhost > airport\_lab20 > public > tables > baggage\_check

8:3 CRLF UTF-8 4 spaces

## DDL

### 1. Create following tables with corresponding attributes:

**Baggage\_check:** baggage\_check\_id(int), check\_result (varchar(50)), created\_at(timestamp), updated\_at(timestamp), booking\_id(int), passenger\_id(int);

On this step I created all required tables such as airline\_info, airport, passengers, flights, booking and others. For each table I defined primary keys and set all attributes with the NOT NULL constraint.



The screenshot displays a database management interface with three main panels. The top-left panel, 'Database Explorer', shows a tree view of the database structure: 'postgres@localhost' (2) contains 'airport\_lab20' (1 of 3), which contains a 'public' schema. Under 'public', there is a 'tables' folder with 8 tables: 'airline\_info', 'airport', 'baggage', 'baggage\_check', 'boarding\_pass' (highlighted), and 'booking'. The top-right panel, 'console', shows a SQL query being executed: `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 );`. The bottom-left panel, 'Services', shows a list of services: 'Database' (105 ms) and 'console' (13 ms). The bottom-right panel shows the execution log: `[2025-09-24 08:48:30] airport_lab20.public> 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 ); [2025-09-24 08:48:30] completed in 5 ms`. The bottom status bar shows the current path: 'Database > postgres@localhost > airport\_lab20 > public > tables > boarding\_pass' and settings: '8:3 CRLF UTF-8 4 spaces'.

Database Explorer

console

postgres@localhost 2

airport\_lab20 1 of 3

public

tables 8

airline\_info

airport

baggage

baggage\_check

boarding\_pass

booking

Services All Services

Database

postgres@localhost

default 105 ms

console 13 ms

[2025-09-24 08:48:30] airport\_lab20.public> 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

)

[2025-09-24 08:48:30] completed in 5 ms

Database > postgres@localhost > airport\_lab20 > public > tables > boarding\_pass

8:3 CRLF UTF-8 4 spaces

## DDL

### 1. Create following tables with corresponding attributes:

**Boarding\_pass:** boarding\_pass\_id(int), booking\_id(int), seat (varchar(50)), boarding\_time(timestamp), created\_at(timestamp), updated\_at(timestamp);

On this step I created all required tables such as airline\_info, airport, passengers, flights, booking and others. For each table I defined primary keys and set all attributes with the NOT NULL constraint.



The screenshot shows a database IDE interface. On the left, the 'Database Explorer' pane shows a tree structure for 'postgres@localhost' with tables like 'airline\_info', 'airport', 'baggage', 'baggage\_check', 'boarding\_pass', 'booking', 'booking\_flight' (selected), 'flights', and 'passengers'. The main editor shows a SQL query to create a table 'booking\_flight' with columns: 'booking\_flight\_id' (INT PRIMARY KEY), 'booking\_id' (INT NOT NULL), 'flight\_id' (INT NOT NULL), 'created\_at' (TIMESTAMP NOT NULL), and 'updated\_at' (TIMESTAMP NOT NULL). The query is executed in the 'console' pane, showing the command and its completion time. The bottom status bar indicates the current context: 'Database > postgres@localhost > airport\_lab20 > public > tables > booking\_flight'.

Database Explorer

postgres@localhost

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

console

```
1 CREATE TABLE booking_flight(  
2     booking_flight_id INT PRIMARY KEY,  
3     booking_id INT NOT NULL,  
4     flight_id INT NOT NULL,  
5     created_at TIMESTAMP NOT NULL,  
6     updated_at TIMESTAMP NOT NULL  
7 );
```

Services

All Services

Database

- postgres@localhost
  - default 105 ms
  - console 15 ms

[2025-09-24 08:48:30] completed in 5 ms

[2025-09-24 08:49:13] airport\_lab20.public> 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  
)

[2025-09-24 08:49:13] completed in 4 ms

Database > postgres@localhost > airport\_lab20 > public > tables > booking\_flight

7:3 CRLF UTF-8 4 spaces

## DDL

### 1. Create following tables with corresponding attributes:

**Booking\_flight:** booking\_flight\_id(int), booking\_id(int), flight\_id(int), created\_at(timestamp), updated\_at(timestamp);

On this step I created all required tables such as airline\_info, airport, passengers, flights, booking and others. For each table I defined primary keys and set all attributes with the NOT NULL constraint.



The screenshot displays a database management tool interface. On the left, the 'Database Explorer' pane shows a tree structure with 'postgres@localhost' expanded, listing tables like 'baggage\_check', 'boarding\_pass', 'booking', 'booking\_flight', 'flights', 'passengers', and 'security\_check'. The 'security\_check' table is selected. Below this, the 'Services' pane shows a tree with 'Database' expanded, listing 'default' (105 ms) and 'console' (16 ms). The 'console' service is selected. The main area shows a SQL editor with the following code:

```
1 CREATE TABLE security_check(  
2     security_check_id INT PRIMARY KEY,  
3     check_result VARCHAR(20) NOT NULL,  
4     created_at TIMESTAMP NOT NULL,  
5     updated_at TIMESTAMP NOT NULL,  
6     passenger_id INT NOT NULL  
7 );
```

Below the editor, the 'Tx' pane shows a list of transactions. The first transaction is '[2025-09-24 08:49:13] completed in 4 ms'. The second transaction is '[2025-09-24 08:50:00] airport\_lab20.public> CREATE TABLE security\_check( security\_check\_id INT PRIMARY KEY, check\_result VARCHAR(20) NOT NULL, created\_at TIMESTAMP NOT NULL, updated\_at TIMESTAMP NOT NULL, passenger\_id INT NOT NULL )', which is highlighted. The third transaction is '[2025-09-24 08:50:00] completed in 5 ms'. The bottom status bar shows the path 'Database > postgres@localhost > airport\_lab20 > public > tables > security\_check' and the settings '7:3 CRLF UTF-8 4 spaces'.

## DDL

### 1. Create following tables with corresponding attributes:

**Security\_check:** security\_check\_id(int),  
check\_result(varchar(20)), created\_at(timestamp),  
updated\_at(timestamp), passenger\_id(int);

On this step I created all required tables such as airline\_info, airport, passengers, flights, booking and others. For each table I defined primary keys and set all attributes with the NOT NULL constraint.



The screenshot shows a database management tool interface. On the left, the 'Database Explorer' pane shows a tree view of the database structure: postgres@localhost > airport\_lab20 > public > tables > airline. The 'Services' pane below it shows the 'console' service is active. The main 'console' pane displays a SQL query: `ALTER TABLE airline_info RENAME TO airline;`. The query has been executed successfully, as indicated by a green checkmark and the message: `[2025-09-24 08:52:16] airport_lab20.public> ALTER TABLE airline_info RENAME TO airline`. The right side of the console shows the table definition for 'airline':

```
security_check_id INT PRIMARY KEY,  
check_result VARCHAR(20) NOT NULL,  
created_at TIMESTAMP NOT NULL,  
updated_at TIMESTAMP NOT NULL,  
passenger_id INT NOT NULL  
)
```

The bottom status bar shows the current path: Database > postgres@localhost > airport\_lab20 > public > tables > airline, along with formatting options: 1:44 CRLF UTF-8 4 spaces.

## DDL

### 5. Rename airline\_info table to airline;

After creating the schema I applied modifications. I renamed airline\_info to airline, changed column price to ticket\_price in booking, altered departing\_gate from varchar(50) to text in flights, and dropped the info column from airline.



The screenshot displays a database management tool interface. On the left, the 'Database Explorer' shows a PostgreSQL database at 'localhost' with a schema named 'airport\_lab20'. The schema contains a table 'booking' with columns: 'passenger\_id' (integer), 'booking\_platform' (varchar(50)), 'created\_at' (timestamp), 'updated\_at' (timestamp), 'status' (varchar(50)), and 'ticket\_price' (numeric(7,2)). Below the schema explorer, the 'Services' section shows a list of database services, including 'default' (105 ms) and 'console' (13 ms).

The main console area shows a SQL query being executed: `ALTER TABLE booking RENAME COLUMN price TO ticket_price;`. The query is marked with a green checkmark, indicating successful execution. The console output shows the following messages:

```
[2025-09-24 08:50:00] completed in 5 ms
[2025-09-24 08:52:16] airport_lab20.public> ALTER TABLE airline_info RENAME TO airline
[2025-09-24 08:52:16] completed in 5 ms
[2025-09-24 08:55:45] airport_lab20.public> ALTER TABLE booking RENAME COLUMN ticket_price TO price
[2025-09-24 08:55:45] completed in 3 ms
[2025-09-24 08:56:04] airport_lab20.public> ALTER TABLE booking RENAME COLUMN price TO ticket_price
[2025-09-24 08:56:04] completed in 3 ms
```

The bottom status bar indicates the current session is 'postgres@localhost' in the 'console' service, with a timestamp of 1:57, CRLF line endings, UTF-8 encoding, and 4 spaces indentation.

## DDL

### 6. Rename column price to ticket\_price in booking table;

After creating the schema I applied modifications. I renamed airline\_info to airline, changed column price to ticket\_price in booking, altered departing\_gate from varchar(50) to text in flights, and dropped the info column from airline.



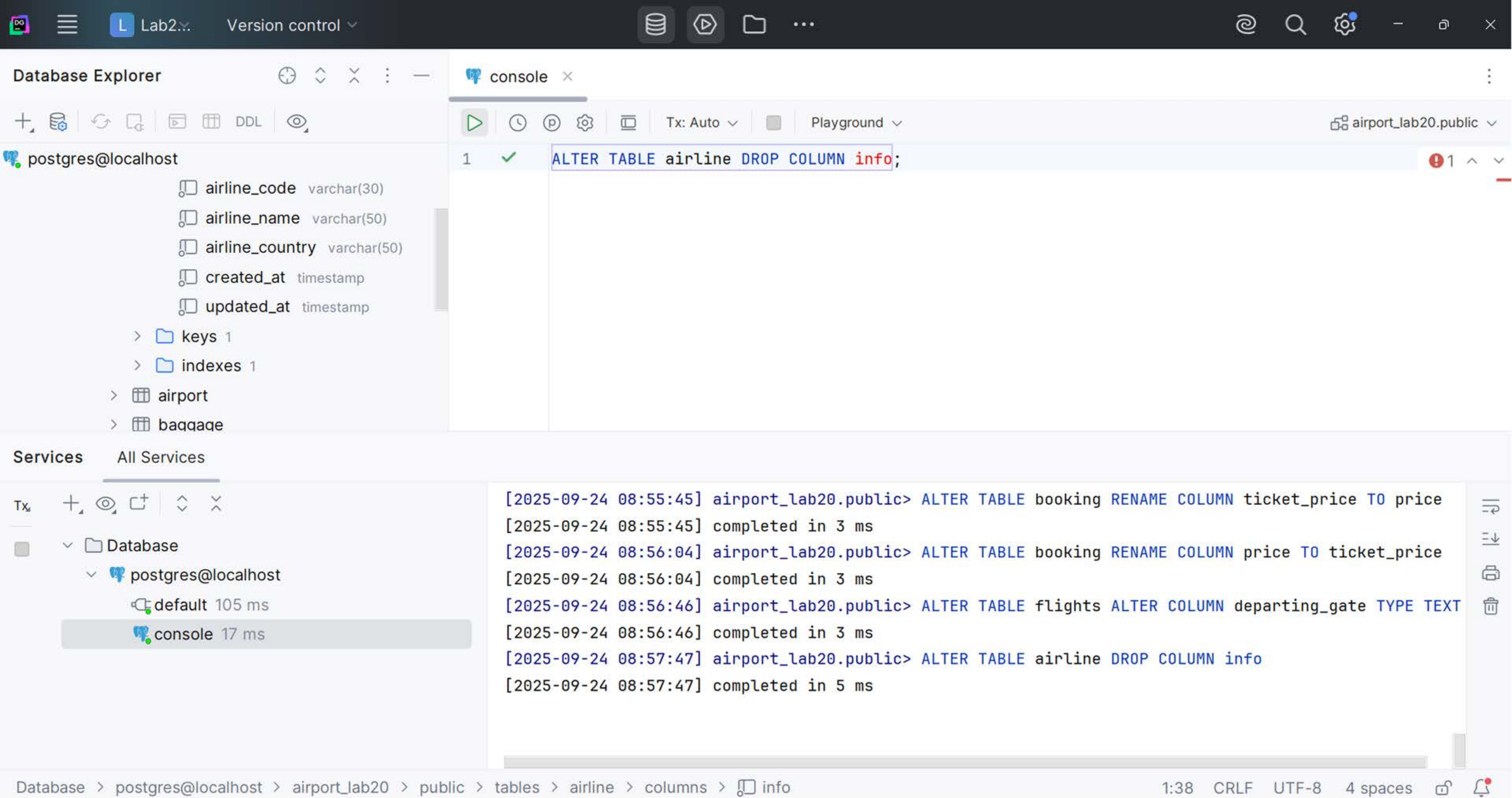
The screenshot displays a database management tool interface. On the left, the 'Database Explorer' shows a schema diagram for a table named 'flights'. The columns listed are: flight\_id (integer), sch\_departure\_time (timestamp), sch\_arrival\_time (timestamp), departing\_airport\_id (integer), arriving\_airport\_id (integer), departing\_gate (text), arriving\_gate (varchar(50)), airline\_id (integer), and act\_departure\_time (timestamp). The 'departing\_gate' column is highlighted. In the center, the 'console' tab shows a SQL statement: `ALTER TABLE flights ALTER COLUMN departing_gate TYPE TEXT;` with a green checkmark indicating successful execution. On the right, the 'Services' panel shows a list of services, including 'default' (105 ms) and 'console' (14 ms). Below this, a log of SQL commands and their execution times is visible: `[2025-09-24 08:50:00] completed in 5 ms`, `[2025-09-24 08:52:16] airport_lab20.public> ALTER TABLE airline_info RENAME TO airline`, `[2025-09-24 08:52:16] completed in 5 ms`, `[2025-09-24 08:55:45] airport_lab20.public> ALTER TABLE booking RENAME COLUMN ticket_price TO price`, `[2025-09-24 08:55:45] completed in 3 ms`, `[2025-09-24 08:56:04] airport_lab20.public> ALTER TABLE booking RENAME COLUMN price TO ticket_price`, `[2025-09-24 08:56:04] completed in 3 ms`, `[2025-09-24 08:56:46] airport_lab20.public> ALTER TABLE flights ALTER COLUMN departing_gate TYPE TEXT`, and `[2025-09-24 08:56:46] completed in 3 ms`. The bottom status bar shows the current path: 'Database > postgres@localhost > airport\_lab20 > public > tables > flights > columns > departing\_gate', along with formatting options like '1:59', 'CRLF', 'UTF-8', and '4 spaces'.

## DDL

### 7. Change data type of departing\_gate from varchar(50) to text;

After creating the schema I applied modifications. I renamed airline\_info to airline, changed column price to ticket\_price in booking, altered departing\_gate from varchar(50) to text in flights, and dropped the info column from airline.





## DDL

7. Drop the column info(varchar(50)) from the airline table.

After creating the schema I applied modifications. I renamed `airline_info` to `airline`, changed column `price` to `ticket_price` in `booking`, altered `departing_gate` from `varchar(50)` to `text` in `flights`, and dropped the `info` column from `airline`.



Database Explorer

postgres@localhost

airline\_code varchar(30)

airline\_name varchar(50)

airline\_country varchar(50)

created\_at timestamp

updated\_at timestamp

keys 1

indexes 1

airport

baggage

baggage\_check

boarding\_pass

booking

booking\_flight

flights

passengers

security\_check

Database Objects

postgres 1 of 3

Server Objects

console

Tx: Auto

Playground

1 ALTER TABLE security\_check

2 ADD CONSTRAINT fk\_security\_check\_passenger

3 FOREIGN KEY (passenger\_id) REFERENCES passengers(passenger\_id);

4

5 ALTER TABLE booking

6 ADD CONSTRAINT fk\_booking\_passenger

7 FOREIGN KEY (passenger\_id) REFERENCES passengers(passenger\_id);

8

9 ALTER TABLE baggage\_check

10 ADD CONSTRAINT fk\_baggage\_check\_passenger

11 FOREIGN KEY (passenger\_id) REFERENCES passengers(passenger\_id);

12

13 ALTER TABLE baggage\_check

14 ADD CONSTRAINT fk\_baggage\_check\_booking

15 FOREIGN KEY (booking\_id) REFERENCES booking(booking\_id);

16

17 ALTER TABLE baggage

18 ADD CONSTRAINT fk\_baggage\_booking

19 FOREIGN KEY (booking\_id) REFERENCES booking(booking\_id);

20

21 ALTER TABLE boarding\_pass

22 ADD CONSTRAINT fk\_boarding\_pass\_booking

23 FOREIGN KEY (booking\_id) REFERENCES booking(booking\_id);

24

25 ALTER TABLE booking\_flight

26 ADD CONSTRAINT fk\_booking\_flight\_booking

27 FOREIGN KEY (booking\_id) REFERENCES booking(booking\_id);

28

29 ALTER TABLE booking\_flight

30 ADD CONSTRAINT fk\_booking\_flight\_flight

31 FOREIGN KEY (flight\_id) REFERENCES flights(flight\_id);

32

33 ALTER TABLE flights

34 ADD CONSTRAINT fk\_flights\_departing\_airport

35 FOREIGN KEY (departing\_airport\_id) REFERENCES airport(airport\_id);

36

37 ALTER TABLE flights

38 ADD CONSTRAINT fk\_flights\_arriving\_airport

39 FOREIGN KEY (arriving\_airport\_id) REFERENCES airport(airport\_id);

40

41 ALTER TABLE flights

42 ADD CONSTRAINT fk\_flights\_airline

43 FOREIGN KEY (airline\_id) REFERENCES airline(airline\_id);

DDL

9. Make a relationship between following tables:

- Passengers with Secuitiry\_check, Booking, Baggage\_check by passenger\_id;
- Booking with Baggage\_check, Baggage, Boarding\_pass, Booking\_flight Flights with Booking\_flight by flight\_id;
- Airport with Flights by departing\_airport\_id;
- Airport with Flights by arriving\_airport\_id;
- Airline with Flights by airline\_id;

Next I established relationships between tables using foreign keys: passengers with security\_check, booking, and baggage\_check; booking with baggage, baggage\_check, boarding\_pass and booking\_flight; flights with booking\_flight; and airline and airport with flights.

tables 10

airline

columns 6

keys 1

indexes 1

airport

columns 7

keys 1

indexes 1

baggage

columns 5

keys 1

foreign keys 1

indexes 1

baggage\_check

columns 6

keys 1

foreign keys 2

indexes 1

boarding\_pass

columns 6

keys 1

foreign keys 1

indexes 1

booking

columns 8

keys 1

foreign keys 1

indexes 1

booking\_flight

columns 5

keys 1

foreign keys 2

indexes 1

flights

columns 12

keys 1

foreign keys 3

indexes 1

passengers

columns 10

keys 1

indexes 1

security\_check

columns 5

keys 1

foreign keys 1

indexes 1



Preview

```
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Carmen De Patagones Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-09-23', '2025-09-23');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Portoroz Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-05-14', '2025-02-25');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Deputado Luiz Eduardo Magalhães International Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-05-14', '2025-02-25');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('San Nicolas Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-04-01', '2025-05-19');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Fonte Boa Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-07-24', '2025-02-22');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Greater Binghamton/Edwin A Link field', 'Turkey', 'Manisa', 'Yeniköy', '2025-07-06', '2025-02-22');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Atatürk International Airport', 'Turkey', 'Manisa', 'Yeniköy', '2024-10-22', '2024-10-22');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Senador Petrônio Portela Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-06-19', '2025-06-19');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Columbia County Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-06-10', '2025-09-09');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Humbert River Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-07-28', '2024-09-25');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Lusk Municipal Airport', 'Turkey', 'Manisa', 'Yeniköy', '2024-11-12', '2025-05-08');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Bolling Air Force Base', 'Turkey', 'Manisa', 'Yeniköy', '2024-11-05', '2025-05-04');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Woitape Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-01-05', '2025-09-23');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Ileg Airport', 'Turkey', 'Manisa', 'Yeniköy', '2024-10-18', '2025-01-07');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Chisasibi Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-02-14', '2024-09-25');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Lubang Airport', 'Turkey', 'Manisa', 'Yeniköy', '2024-10-23', '2025-09-04');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('L'Aquila-Preturo Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-07-05', '2024-11-12');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Maintirano Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-06-28', '2025-04-04');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Kasos Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-05-06', '2025-09-17');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Piatã Airport', 'Turkey', 'Manisa', 'Yeniköy', '2024-10-20', '2025-07-12');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Sharjah International Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-07-12', '2025-07-12');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Malacca Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-07-17', '2025-03-30');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Helenvale Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-01-01', '2024-12-26');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Baie Comeau Airport', 'Turkey', 'Manisa', 'Yeniköy', '2025-06-01', '2025-05-17');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Caribou Municipal Airport', 'Turkey', 'Manisa', 'Yeniköy', '2024-11-26', '2025-02-22');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Rhinelander Oneida County Airport', 'Turkey', 'Manisa', 'Yeniköy', '2024-11-05', '2024-11-05');
```

showing first 100 rows

# Rows: 200

GENERATE DATA

CLOSE

## DML

### 1. Generate and insert 200 random rows in your airport database.

I generated and inserted 200 rows into the airport table. Then I added a new airline named KazAir in Kazakhstan, and later updated its country to Turkey. After that I inserted three airlines at once: AirEasy (France), FlyHigh (Brazil), and FlyFly (Poland).

Database Explorer

console

```
1 SELECT count(*) FROM airport;
2 SELECT * FROM airport LIMIT 10;
3
```

Services

All Services

Output

count(\*):bigint

airport\_lab20.public.airport

CSV

10 rows

airport_id	airport_name	country	state	city	created_at	updated_at
1	154 Tarempa Airport	Bolivia	<null>	Sotomayor	2025-07-04 00:00:00.000000	2024-11-26 00:00:00.000000
2	155 Unalakleet Airport	Egypt	<null>	Sumuștă as Sultānī	2025-04-07 00:00:00.000000	2024-11-26 00:00:00.000000
3	156 Achutupo Airport	Russia	<null>	Krasnogvardeyets	2025-02-18 00:00:00.000000	2025-02-18 00:00:00.000000
4	157 Chistochina Airport	Czech Republic	<null>	Nová Role	2024-12-18 00:00:00.000000	2024-12-18 00:00:00.000000
5	158 Qaanaaq Airport	China	<null>	Matou	2025-04-01 00:00:00.000000	2025-04-01 00:00:00.000000
6	159 Blackall Airport	China	<null>	Pingpu	2025-06-22 00:00:00.000000	2025-06-22 00:00:00.000000
7	160 Kilkenny Airport	Vietnam		Phú Lộc	2025-01-31 00:00:00.000000	2024-11-26 00:00:00.000000
8	161 Majeed Bin Abdulaziz Airport	Portugal		São Miguel Mosteiros	2025-02-11 00:00:00.000000	2025-02-11 00:00:00.000000

Database Consoles > postgres@localhost > console

3:1 CRLF UTF-8 4 spaces

Database Explorer

console

```
1 SELECT count(*) FROM airport;
2 SELECT * FROM airport LIMIT 10;
3
```

Services

All Services

Output

count(\*):bigint

airport\_lab20.public.airport

CSV

1 row

count
200

Database Consoles > postgres@localhost > console

3:1 CRLF UTF-8 4 spaces



Database Explorer

postgres@localhost 2

airport\_lab20 1 of 3

public

tables 10

sequences 1

Database Objects

postgres 1 of 3

console

```
1 INSERT INTO airline (airline_code, airline_name, airline_country, created_at, updated_at)
2 VALUES ( airline_code 'KZR', airline_name 'KazAir', airline_country 'Kazakhstan', created_at now(), updated_at now());
3
4 SELECT airline_id, airline_name, airline_country FROM airline WHERE airline_name='KazAir';
```

Services All Services

Output airport\_lab20.public.airline

airline_id	airline_name	airline_country
1	KazAir	Kazakhstan
2	KazAir	Kazakhstan

2 rows

Database Consoles > postgres@localhost > console 4:91 CRLF UTF-8 4 spaces

## DML

2. Add a new airline named "KazAir" based in "Kazakhstan" to the airline table.

I generated and inserted 200 rows into the airport table. Then I added a new airline named KazAir in Kazakhstan, and later updated its country to Turkey. After that I inserted three airlines at once: AirEasy (France), FlyHigh (Brazil), and FlyFly (Poland).



Database Explorer

postgres@localhost 2

airport\_lab20 1 of 3

public

tables 10

sequences 1

Database Objects

postgres 1 of 3

console

```
1 UPDATE airline
2 SET airline_country = 'Turkey', updated_at = now()
3 WHERE airline_name = 'KazAir';
4
5 ✓ SELECT airline_id, airline_name, airline_country FROM airline WHERE airline_name='KazAir';
```

Services All Services

Output airport\_lab20.public.airline

	airline_id	airline_name	airline_country
1	1	KazAir	Turkey
2	2	KazAir	Turkey

2 rows

Database Consoles > postgres@localhost > console 5:91 CRLF UTF-8 4 spaces

## DML

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

I generated and inserted 200 rows into the airport table. Then I added a new airline named KazAir in Kazakhstan, and later updated its country to Turkey. After that I inserted three airlines at once: AirEasy (France), FlyHigh (Brazil), and FlyFly (Poland).



The screenshot shows a database management interface. On the left, the 'Database Explorer' pane shows a connection to 'postgres@localhost' with a database 'airport\_lab20'. Under 'public', there are 'tables 10', 'sequences 1', and 'Database Objects'. The 'Services' pane shows 'All Services'. The main area is split into two panes. The top pane, titled 'console', shows a SQL query being executed: 

```
( airline_code 'FFY', airline_name 'FlyFly', airline_country 'Poland', created_at now(), updated_at now());
```

 followed by 

```
SELECT airline_name, airline_country FROM airline
```

 and 

```
WHERE airline_name IN ('AirEasy', 'FlyHigh', 'FlyFly');
```

. The bottom pane, titled 'Output', shows the result of the query as a table with two columns: 'airline\_name' and 'airline\_country'. The table contains three rows: 'AirEasy' from France, 'FlyHigh' from Brazil, and 'FlyFly' from Poland. The status bar at the bottom indicates 'Database Consoles > postgres@localhost > console' and shows settings for '7:54', 'CRLF', 'UTF-8', and '4 spaces'.

Database Explorer

postgres@localhost 2

airport\_lab20 1 of 3

public

tables 10

sequences 1

Database Objects

postgres 1 of 3

Services All Services

Output airport\_lab20.public.airline

	airline_name	airline_country
1	AirEasy	France
2	FlyHigh	Brazil
3	FlyFly	Poland

3 rows

Database Consoles > postgres@localhost > console

7:54 CRLF UTF-8 4 spaces

## DML

4. Add three airlines at once: "AirEasy" in "France", "FlyHigh" in "Brazil" and "FlyFly" in "Poland".

I generated and inserted 200 rows into the airport table. Then I added a new airline named KazAir in Kazakhstan, and later updated its country to Turkey. After that I inserted three airlines at once: AirEasy (France), FlyHigh (Brazil), and FlyFly (Poland).



The screenshot shows a database management tool interface. On the left, the 'Database Explorer' pane shows a connection to 'postgres@localhost' with a database 'airport\_lab20'. Under 'public', there are 'tables 10', 'sequences 1', and 'Database Objects'. The 'Services' pane shows 'All Services'. The main area is a 'console' window with a SQL query: `DELETE FROM flights WHERE act_arrival_time >= '2024-01-01' AND act_arrival_time < '2025-01-01';` followed by `SELECT flight_id, act_arrival_time FROM flights ORDER BY flight_id;`. The query is executed, and the result is shown in a table viewer below. The table viewer shows the columns 'flight\_id' and 'act\_arrival\_time'. The status bar at the bottom indicates '0 rows'.

Database Explorer

postgres@localhost 2

airport\_lab20 1 of 3

public

tables 10

sequences 1

Database Objects

postgres 1 of 3

Services All Services

Output airport\_lab20.public.flights

flight\_id act\_arrival\_time

0 rows

Database Consoles > postgres@localhost > console

5:1 CRLF UTF-8 4 spaces

## DML

### 5. Delete all flights whose arrival in 2024 year.

On this step I removed all flights that had their actual arrival time in the year 2024.

I used a DELETE statement with a date condition to filter the rows.

As a result, only the flights arriving outside of 2024 remained in the table.



The screenshot shows a database management tool interface. On the left is the 'Database Explorer' pane showing a tree view of the database structure: postgres@localhost (2) > airport\_lab20 (1 of 3) > public > tables (10). The main area is a 'console' tab with a SQL editor. The editor contains the following SQL statement:

```
1 UPDATE booking
2 SET ticket_price = ROUND((ticket_price * 1.15)::numeric,2),
3     updated_at = now()
4 WHERE TRUE;
```

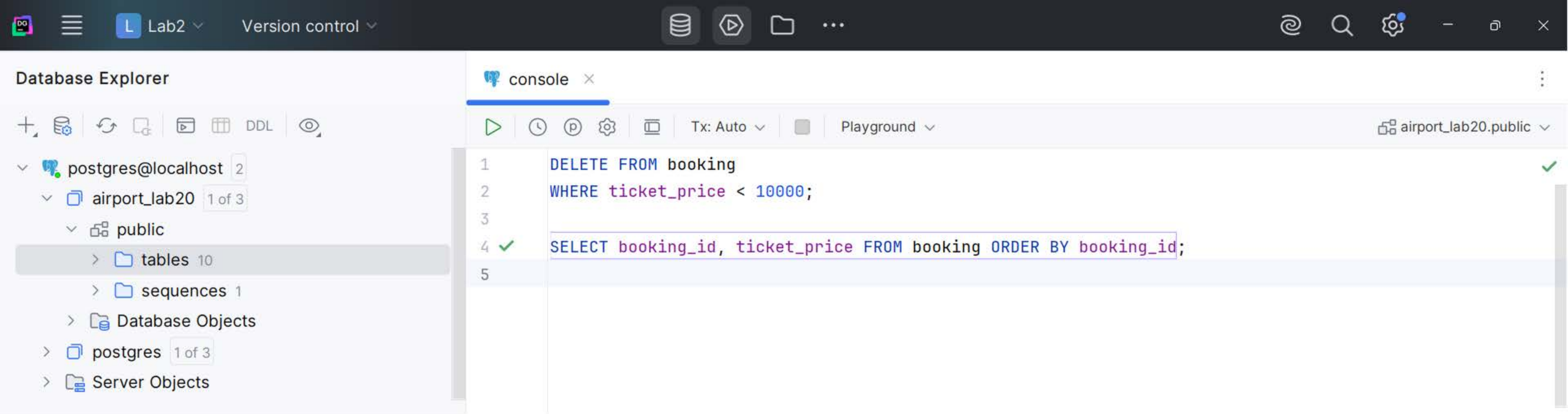
Below the editor is a 'Services' pane showing a log entry: [2025-09-24 11:22:26] completed in 4 ms. The bottom status bar indicates the current context: Database Consoles > postgres@localhost > console, and formatting settings: 5:1 CRLF UTF-8 4 spaces.

## DML

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

On this step I updated the booking table to increase the price of all tickets by 15%. I used an UPDATE statement that multiplied the value in the ticket\_price column by 1.15. As a result, every row in the booking table had its ticket price raised, while the update time column was also refreshed.





## DML

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

On this step I removed all records from the booking table where the ticket price was less than 10000. I used a DELETE query with a condition on the ticket\_price column. As a result, only the tickets with higher prices remained in the table.