

The screenshot shows a PostgreSQL database management interface. In the Database Explorer, under the 'airport\_lab20' database's 'public' schema, the 'airline\_info' table is selected. In the console, the CREATE TABLE command for 'airline\_info' is displayed, showing columns: 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), and info (VARCHAR(50) NOT NULL). The command ends with a semicolon. Below the console, the execution log shows the command being run at [2025-09-24 08:31:52] and completed in 5 ms. The bottom status bar indicates the path Database > postgres@localhost > airport\_lab20 > public > tables > airline\_info.

```
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] 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
```

## 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 shows the PostgreSQL Database Explorer in the VS Code interface. The left sidebar displays the database structure under 'postgres@localhost' and 'airport\_lab20'. The 'Tables' section under 'airport\_lab20' contains 'airline\_info' and 'airport'. The 'airport' table is selected, showing its schema:

```
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
);
```

The bottom right pane shows the command history:

```
[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
[2025-09-24 08:35:21] airport_lab20.public> CREATE TABLE airport (
```

The status bar at the bottom indicates the current encoding is CRLF.

## 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 shows a PostgreSQL database management interface. In the Database Explorer, under the 'public' schema of the 'airport\_lab20' database, the 'passengers' table is selected. In the console, the CREATE TABLE command for the 'passengers' table is displayed, defining columns for 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), and updated\_at (TIMESTAMP NOT NULL). The transaction status in the Services panel shows a completed transaction in the 'console' tab.

```
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
);
```

## 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 shows a PostgreSQL database management interface. On the left, the Database Explorer pane displays the schema of the 'airport\_lab20' database, specifically the 'public' schema, which contains four tables: 'airline\_info', 'airport', 'flights', and 'passengers'. The 'flights' table is currently selected. In the center, the console pane shows the SQL DDL command for creating the 'flights' table:

```
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 TEXT 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
);
```

The command is numbered from 1 to 14. A tooltip for the 'departing\_airport\_id' column indicates it is an INT type with a NOT NULL constraint. The status bar at the bottom shows the command was completed in 9 ms on 2025-09-24 at 08:44:00.

## 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 shows the DataGrip IDE interface. In the top navigation bar, there are icons for file, edit, and database management. The title bar says "Lab2... Version control". On the left, the "Database Explorer" sidebar shows a connection to "postgres@localhost" with a database "airport\_lab20" selected. Inside "airport\_lab20", there's a "public" schema containing five tables: "airline\_info", "airport", "booking" (which is currently selected), "flights", and "passengers". Below the schema list is a "Database Objects" section. On the right, the main area is titled "console" and contains a code editor with the following SQL DDL statement:

```
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
);
```

Below the code editor, the "Services" tab is active, showing "All Services". Under "Tx", there are two entries: "default" (105 ms) and "console" (19 ms). The "Database" section shows the same connection details as the sidebar. The main console window displays the command and its execution results:

```
[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
```

At the bottom, the status bar indicates "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 shows the VS Code interface with the Database Explorer and Console extensions. In the Database Explorer, under the 'airport\_lab20' database's 'public' schema, the 'baggage' table is selected. In the Console, the following SQL code is shown:

```
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 execution history in the Console shows:

- [2025-09-24 08:45:22] completed in 5 ms
- [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  
)
- [2025-09-24 08:46:32] completed in 4 ms

The status bar at the bottom indicates the path: Database > postgres@localhost > airport\_lab20 > public > tables > baggage.

## 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 shows the VS Code interface with the Database Explorer and Console extensions active. In the Database Explorer, under the 'airport\_lab20' database's 'public' schema, a new table named 'baggage\_check' is being created. The table structure is defined by the following DDL:

```
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
);
```

In the Console tab, the command is run and completed successfully:

```
[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 status bar at the bottom indicates the file is saved with 8:3 CRLF encoding, in UTF-8, and has 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 shows the Visual Studio Code interface with the Database Explorer and SQL console extensions active. In the Database Explorer, under the 'airport\_lab20' database, the 'public' schema contains a table named 'boarding\_pass'. The SQL console shows the DDL command used to create this table:

```
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 command was run at [2025-09-24 08:48:30] and completed in 5 ms. The transaction history shows it took 105 ms for the default connection and 13 ms for the current 'console' connection.

Services: All Services

Tx: + ⏪ ⏴ ⏵ ⏳ ⏷ ⏸

Database: 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 the DataGrip IDE interface. On the left, the Database Explorer pane lists tables under the schema 'postgres@localhost'. The 'booking\_flight' table is selected and highlighted with a blue background. In the center, the console pane displays the SQL code for creating the 'booking\_flight' table:

```
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
);
```

Below the code, the execution log shows the command was completed in 5 ms. The transaction history on the left shows a transaction named 'console' took 15 ms. The bottom status bar indicates the current database path is 'Database > postgres@localhost > airport\_lab20 > public > tables > booking\_flight'.

## 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 shows the Microsoft SQL Server Management Studio (SSMS) interface. In the top left, there's a 'Database Explorer' pane showing a database structure with several tables like baggage\_check, boarding\_pass, booking, booking\_flight, flights, passengers, and security\_check. The security\_check table is currently selected. The top right contains a 'console' tab where DDL (Data Definition Language) code is being run. The code is:

```
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
);
```

Below the code, the execution results are shown in the console window:

```
[2025-09-24 08:49:13] completed in 4 ms
[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
)
[2025-09-24 08:50:00] completed in 5 ms
```

The bottom left shows a 'Services' section with 'All Services'. The bottom right shows file details: 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 the DBeaver Database Explorer interface. On the left, the Database Explorer tree shows a connection to 'postgres@localhost' with a database 'airport\_lab20'. Under 'public' schema, there are ten tables: airline, airport, baggage, baggage\_check, boarding\_pass, booking, etc. The 'airline' table is currently selected. In the center, the 'console' tab displays the command: 'ALTER TABLE airline\_info RENAME TO airline;'. The output pane shows the execution details: [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, and [2025-09-24 08:52:16] completed in 5 ms. The right side of the interface shows the modified schema for the 'airline' table, which includes columns like security\_check\_id, check\_result, created\_at, updated\_at, and passenger\_id.

```
ALTER TABLE airline_info RENAME TO 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
```

```
[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
```

## 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 shows the VS Code interface with the Database Explorer and Console extensions installed. In the Database Explorer, a connection to 'postgres@localhost' is selected, showing a table named 'booking' with columns: passenger\_id, booking\_platform, created\_at, updated\_at, status, ticket\_price, keys (1), indexes (1), and booking\_flight. In the Console tab, a command is run: `ALTER TABLE booking RENAME COLUMN price TO ticket_price;`. The command is successful, indicated by a green checkmark and a warning icon. The output window shows the command and its execution details: [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.

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

```
[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
```

## 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 shows the DataGrip IDE interface. The Database Explorer pane on the left lists the schema for the 'flights' table, including columns like flight\_id, sch\_departure\_time, sch\_arrival\_time, departing\_airport\_id, arriving\_airport\_id, departing\_gate, arriving\_gate, airline\_id, and act\_departure\_time. The console pane in the center shows the execution of an SQL command: `ALTER TABLE flights ALTER COLUMN departing_gate TYPE TEXT;`. The Services pane at the bottom displays a list of database connections, with 'postgres@localhost' selected.

```
[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
[2025-09-24 08:56:46] completed in 3 ms
```

## 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.

The screenshot shows the Visual Studio Code interface with the Database Explorer and Console extensions active. In the Database Explorer, a connection to 'postgres@localhost' is selected, showing tables like 'airline', 'keys', 'indexes', 'airport', and 'baqqae'. The 'airline' table is expanded, showing columns: 'airline\_code' (varchar(30)), 'airline\_name' (varchar(50)), 'airline\_country' (varchar(50)), 'created\_at' (timestamp), and 'updated\_at' (timestamp). In the Console tab, a transaction is running with ID 1, which contains the command: `ALTER TABLE airline DROP COLUMN info;`. The status bar at the bottom indicates the path: Database > postgres@localhost > airport\_lab20 > public > tables > airline > columns > info.

```
[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
[2025-09-24 08:56:46] completed in 3 ms
[2025-09-24 08:57:47] airport_lab20.public> ALTER TABLE airline DROP COLUMN info
[2025-09-24 08:57:47] completed in 5 ms
```

## 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

+	File	Table	DDL	Eye
+	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			

## DDL

### 9. Make a relationship between following tables:

- Passengers with Security\_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.

```

 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);

```

tables 10		
airline	boarding_pass	flights
columns 6	columns 6	columns 12
keys 1	keys 1	keys 1
indexes 1	foreign keys 1	foreign keys 3
airport	indexes 1	indexes 1
columns 7	booking	passengers
keys 1	columns 8	columns 10
indexes 1	keys 1	keys 1
baggage	foreign keys 1	indexes 1
columns 5	indexes 1	security_check
keys 1	booking	columns 5
foreign keys 1	columns 5	keys 1
indexes 1	keys 1	foreign keys 1
baggage_check	foreign keys 2	indexes 1
columns 6	indexes 1	
keys 1		
foreign keys 2		
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', '2025-04-01', '2025-05-19');
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', '2025-07-06');
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', '2024-11-11');
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', '2025-02-28');
insert into airport (airport_name, country, state, city, created_at, updated_at) values ('Rhinelander Oneida County Airport', 'Turkey', 'Manisa', 'Yeniköy', '2024-11-05', '2025-02-28');
```

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

Services

Output count(\*) bigint airport\_lab20.public.airport

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

Database Explorer

Services

Output count(\*) bigint

SELECT count(\*) FROM airport

count
200

Database Explorer

Services

Output count(\*) bigint

SELECT count(\*) FROM airport

1 row

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

The screenshot shows a PostgreSQL database interface with the following details:

- Database Explorer:** Shows the connection to `postgres@localhost` and the database `airport_lab20`. The `public` schema is expanded, showing `tables` (10), `sequences` (1), and `Database Objects`.
- Services:** Shows the `Output` tab for the `airport_lab20.public.airline` table.
- Console:** Displays two SQL statements:
  - `INSERT INTO airline (airline_code, airline_name, airline_country, created_at, updated_at)  
VALUES ('KZR', 'KazAir', 'Kazakhstan', now(), now());`
  - `SELECT airline_id, airline_name, airline_country FROM airline WHERE airline_name='KazAir';`The second statement has a green checkmark indicating it was successful.
- Table Data:** The `airline` table has the following data:

airline_id	airline_name	airline_country
1	KazAir	Kazakhstan
2	KazAir	Kazakhstan

## 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).

The screenshot shows a PostgreSQL database management interface. In the top right, there's a 'console' tab with a blue icon. Below it, a code editor window displays two SQL statements:

```
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';
```

The fifth statement is highlighted with a green checkmark. In the bottom left, a 'Services' section shows 'All Services'. The main area features a table viewer for the 'airline' table, which has columns: airline\_id, airline\_name, and airline\_country. The data shows two rows:

	airline_id	airline_name	airline_country
1	1	KazAir	Turkey
2	2	KazAir	Turkey

At the bottom, the status bar indicates 'Database Consoles > postgres@localhost > console' and shows the current time as 5:91.

## 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 PostgreSQL database interface. In the top navigation bar, there are icons for file, edit, and search, followed by "Lab2" and "Version control". On the right are icons for refresh, search, and exit. Below the navigation bar is the "Database Explorer" sidebar, which lists connections, databases, and tables. A "console" tab is active, showing a query in progress:

```
4  (airline_code 'FFY', airline_name 'FlyFly', airline_country 'Poland', created_at now(), updated_at now());
5
6 ✓ SELECT airline_name, airline_country FROM airline
7 WHERE airline_name IN ('AirEasy','FlyHigh','FlyFly');
8
```

The "Playground" tab is also visible. In the main area, under the "Services" tab, the "Output" tab is selected, showing the results of the query:

	airline_name	airline_country
1	AirEasy	France
2	FlyHigh	Brazil
3	FlyFly	Poland

At the bottom, the status bar shows "Database Consoles > postgres@localhost > console", the time "7:54", and encoding information "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 PostgreSQL database management interface. In the top navigation bar, there are tabs for 'Lab2' and 'Version control'. The main area has a 'Database Explorer' sidebar on the left, which lists connections, databases, and tables. A 'console' tab is active, showing a history of SQL commands. The first command is a DELETE statement:

```
DELETE FROM flights  
WHERE act_arrival_time >= '2024-01-01' AND act_arrival_time < '2025-01-01';
```

The second command is a SELECT statement:

```
SELECT flight_id, act_arrival_time FROM flights ORDER BY flight_id;
```

In the bottom half of the interface, there is a 'Output' tab for the 'airport\_lab20.public.flights' table. It displays two columns: 'flight\_id' and 'act\_arrival\_time'. Below the table, it says '0 rows'.

## 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 PostgreSQL database management interface. On the left, the Database Explorer sidebar lists connections, databases, and objects. The main area is a console window titled "console" showing a successful UPDATE query:

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

Below the console, the Services panel shows the query was completed in 4 ms on 2025-09-24 at 11:22:26. The bottom status bar indicates the console is using 5:1 CRLF encoding, UTF-8 encoding, and 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.

The screenshot shows the DataGrip IDE interface. On the left, the Database Explorer sidebar is open, showing a tree structure of database objects. Under 'postgres@localhost' > 'airport\_lab20' > 'public', there are 'tables' (10), 'sequences' (1), 'Database Objects', and other sections. In the center, the 'console' tab is active, displaying a code editor with the following SQL queries:

```
1  DELETE FROM booking
2  WHERE ticket_price < 10000;
3
4  SELECT booking_id, ticket_price FROM booking ORDER BY booking_id;
```

## 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.