

Here, I've created all the necessary tables, defined their columns, and established the primary keys. This forms the foundation of my database.

The screenshot displays a PostgreSQL database management interface. On the left, the 'Object Explorer' shows the database structure, including schemas, tables, and sequences. The 'public' schema is expanded, showing a list of tables: airline\_info, airport, baggage, baggage\_check, boarding\_pass, booking, booking\_flight, flights, passengers, security\_check, Trigger Functions, Types, Views, Subscriptions, Login/Group Roles, and Tablespaces. The 'Tables (10)' folder is selected, and the 'airline\_info' table is highlighted.

The main window shows the SQL query editor with the following SQL code:

```
1 CREATE TABLE airline_info (  
2     airline_id serial PRIMARY KEY,  
3     airline_code varchar(30) NOT NULL,  
4     airline_name varchar(50) NOT NULL,  
5     airline_country varchar(50) NOT NULL,  
6     created_at timestamp NOT NULL DEFAULT now(),  
7     updated_at timestamp NOT NULL DEFAULT now(),  
8     info varchar(50) NOT NULL  
9 );  
10  
11 CREATE TABLE airport (  
12     airport_id serial PRIMARY KEY,  
13     airport_name varchar(50) NOT NULL,  
14     country varchar(50) NOT NULL,  
15     state varchar(50) NOT NULL,  
16     city varchar(50) NOT NULL,  
17     created_at timestamp NOT NULL DEFAULT now(),  
18     updated_at timestamp NOT NULL DEFAULT now()  
19 );  
20  
21 CREATE TABLE passengers (  
22     passenger_id serial PRIMARY KEY,  
23     first_name varchar(50) NOT NULL,  
24     last_name varchar(50) NOT NULL,  
25     date_of_birth date NOT NULL,  
26     gender varchar(50) NOT NULL,  
27     country_of_citizenship varchar(50) NOT NULL,  
28     country_of_residence varchar(50) NOT NULL,  
29     passport_number varchar(20) NOT NULL,  
30     created_at timestamp NOT NULL DEFAULT now(),  
31     updated_at timestamp NOT NULL DEFAULT now()  
32 );  
33
```

The 'Data Output' tab shows the results of the query, indicating that the tables were created successfully. The 'Messages' tab shows the execution progress, and the 'Notifications' tab shows any system messages.

The bottom window shows the same SQL query editor with the following SQL code:

```
34 CREATE TABLE flights (  
35     flight_id serial PRIMARY KEY,  
36     sch_departure_time timestamp NOT NULL,  
37     sch_arrival_time timestamp NOT NULL,  
38     departing_airport_id int NOT NULL,  
39     arriving_airport_id int NOT NULL,  
40     departing_gate varchar(50) NOT NULL,  
41     arriving_gate varchar(50) NOT NULL,  
42     airline_id int NOT NULL,  
43     act_departure_time timestamp NOT NULL,  
44     act_arrival_time timestamp NOT NULL,  
45     created_at timestamp NOT NULL DEFAULT now(),  
46     updated_at timestamp NOT NULL DEFAULT now(),  
47     CONSTRAINT fk_departing_airport FOREIGN KEY(departing_airport_id) REFERENCES airport(airport_id),  
48     CONSTRAINT fk_arriving_airport FOREIGN KEY(arriving_airport_id) REFERENCES airport(airport_id),  
49     CONSTRAINT fk_flights_airline FOREIGN KEY(airline_id) REFERENCES airline_info(airline_id)  
50 );  
51  
52 CREATE TABLE booking (  
53     booking_id serial PRIMARY KEY,  
54     flight_id int NOT NULL,  
55     passenger_id int NOT NULL,  
56     booking_platform varchar(50) NOT NULL,  
57     created_at timestamp NOT NULL DEFAULT now(),  
58     updated_at timestamp NOT NULL DEFAULT now(),  
59     status varchar(50) NOT NULL,  
60     price decimal(7,2) NOT NULL,  
61     CONSTRAINT fk_booking_flight FOREIGN KEY(flight_id) REFERENCES flights(flight_id),  
62     CONSTRAINT fk_booking_passenger FOREIGN KEY(passenger_id) REFERENCES passengers(passenger_id)  
63 );  
64  
65 CREATE TABLE booking_flight (  
66     booking_flight_id serial PRIMARY KEY,
```

Object Explorer

Languages

Publications

Schemas (1)

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (10)

airline\_info

airport

baggage

baggage\_check

boarding\_pass

booking

booking\_flight

flights

passengers

security\_check

Trigger Functions

Types

Views

Subscriptions

Login/Group Roles

Tablespaces

localhost

postgres/postgres@PostgreSQL 17\* X

postgres/postgres@PostgreSQL 17

Query

Query History

Scratch Pad x

65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97

```
CREATE TABLE booking_flight (  
    booking_flight_id serial PRIMARY KEY,  
    booking_id int NOT NULL,  
    flight_id int NOT NULL,  
    created_at timestamp NOT NULL DEFAULT now(),  
    updated_at timestamp NOT NULL DEFAULT now(),  
    CONSTRAINT fk_bf_booking FOREIGN KEY(booking_id) REFERENCES booking(booking_id),  
    CONSTRAINT fk_bf_flight FOREIGN KEY(flight_id) REFERENCES flights(flight_id)  
);  
  
CREATE TABLE boarding_pass (  
    boarding_pass_id serial PRIMARY KEY,  
    booking_id int NOT NULL,  
    seat varchar(50) NOT NULL,  
    boarding_time timestamp NOT NULL,  
    created_at timestamp NOT NULL DEFAULT now(),  
    updated_at timestamp NOT NULL DEFAULT now(),  
    CONSTRAINT fk_bp_booking FOREIGN KEY(booking_id) REFERENCES booking(booking_id)  
);  
  
CREATE TABLE baggage (  
    baggage_id serial PRIMARY KEY,  
    weight_in_kg decimal(4,2) NOT NULL,  
    created_at timestamp NOT NULL DEFAULT now(),  
    updated_at timestamp NOT NULL DEFAULT now(),  
    booking_id int NOT NULL,  
    CONSTRAINT fk_bagg_booking FOREIGN KEY(booking_id) REFERENCES booking(booking_id)  
);  
  
CREATE TABLE baggage_check (  
    baggage_check_id serial PRIMARY KEY,  
    check_result varchar(50) NOT NULL,  
    created_at timestamp NOT NULL DEFAULT now(),
```

Data Output Messages Notifications

Total rows:

CRLF Ln 1, Col 1

postgres/postgres@PostgreSQL 17\* X

postgres/postgres@PostgreSQL 17

Query

Query History

Scratch Pad x

80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112

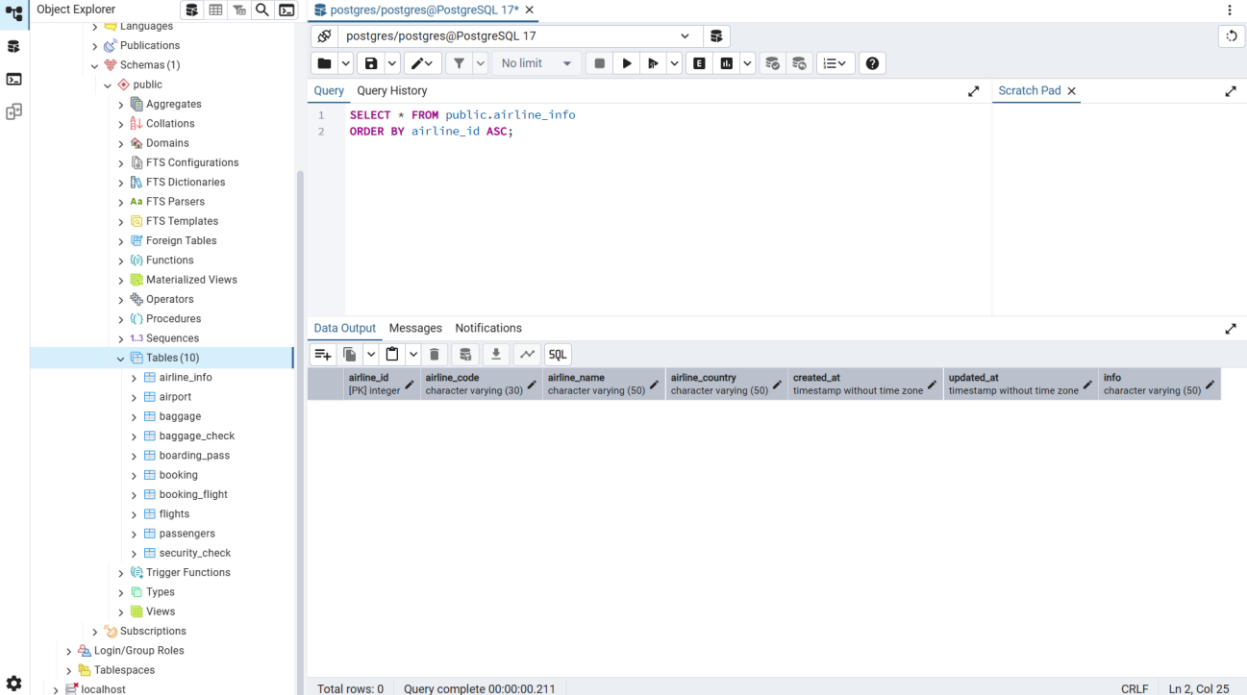
```
created_at timestamp NOT NULL DEFAULT now(),  
updated_at timestamp NOT NULL DEFAULT now(),  
CONSTRAINT fk_bp_booking FOREIGN KEY(booking_id) REFERENCES booking(booking_id)  
);  
  
CREATE TABLE baggage (  
    baggage_id serial PRIMARY KEY,  
    weight_in_kg decimal(4,2) NOT NULL,  
    created_at timestamp NOT NULL DEFAULT now(),  
    updated_at timestamp NOT NULL DEFAULT now(),  
    booking_id int NOT NULL,  
    CONSTRAINT fk_bagg_booking FOREIGN KEY(booking_id) REFERENCES booking(booking_id)  
);  
  
CREATE TABLE baggage_check (  
    baggage_check_id serial PRIMARY KEY,  
    check_result varchar(50) NOT NULL,  
    created_at timestamp NOT NULL DEFAULT now(),  
    updated_at timestamp NOT NULL DEFAULT now(),  
    booking_id int NOT NULL,  
    passenger_id int NOT NULL,  
    CONSTRAINT fk_bc_booking FOREIGN KEY(booking_id) REFERENCES booking(booking_id),  
    CONSTRAINT fk_bc_passenger FOREIGN KEY(passenger_id) REFERENCES passengers(passenger_id)  
);  
  
CREATE TABLE security_check (  
    security_check_id serial PRIMARY KEY,  
    check_result varchar(20) NOT NULL,  
    created_at timestamp NOT NULL DEFAULT now(),  
    updated_at timestamp NOT NULL DEFAULT now(),  
    passenger_id int NOT NULL,  
    CONSTRAINT fk_sc_passenger FOREIGN KEY(passenger_id) REFERENCES passengers(passenger_id)  
);
```

Data Output Messages Notifications

Total rows:

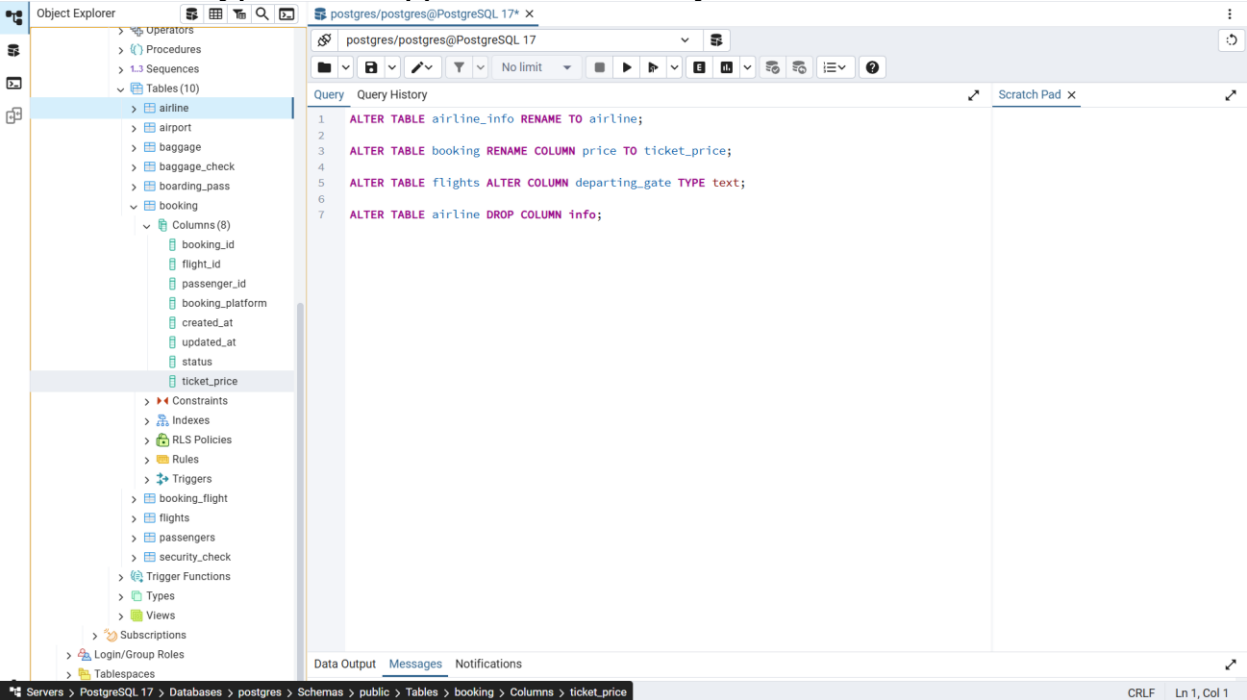
CRLF Ln 1, Col 1

Here, I'm verifying that all the tables have been successfully created in the database.



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays the database structure. Under the 'public' schema, the 'Tables (10)' folder is expanded, showing a list of tables: airline\_info, airport, baggage, baggage\_check, boarding\_pass, booking, booking\_flight, flights, passengers, security\_check, and Trigger Functions. The 'airline\_info' table is selected. On the right, the 'Data Output' pane shows the table's structure with the following columns: airline\_id (PK) integer, airline\_code character varying (30), airline\_name character varying (50), airline\_country character varying (50), created\_at timestamp without time zone, updated\_at timestamp without time zone, and info character varying (50). The 'Query' pane at the top shows a query: `SELECT * FROM public.airline_info ORDER BY airline_id ASC;`. The status bar at the bottom indicates 'Total rows: 0' and 'Query complete 00:00:00.211'.

Here, I've made changes to the table structure: I renamed a table and a column, altered a data type, and dropped an unnecessary column.



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays the database structure. Under the 'public' schema, the 'Tables' folder is expanded, and the 'booking' table is selected. The 'Columns (8)' folder is expanded, showing a list of columns: booking\_id, flight\_id, passenger\_id, booking\_platform, created\_at, updated\_at, status, and ticket\_price. The 'ticket\_price' column is selected. On the right, the 'Data Output' pane shows the table's structure with the following columns: booking\_id integer, flight\_id integer, passenger\_id integer, booking\_platform character varying (50), created\_at timestamp without time zone, updated\_at timestamp without time zone, status character varying (50), and ticket\_price text. The 'Query' pane at the top shows a query: `ALTER TABLE airline_info RENAME TO airline;`  
`ALTER TABLE booking RENAME COLUMN price TO ticket_price;`  
`ALTER TABLE flights ALTER COLUMN departing_gate TYPE text;`  
`ALTER TABLE airline DROP COLUMN info;`. The status bar at the bottom indicates 'Servers > PostgreSQL 17 > Databases > postgres > Schemas > public > Tables > booking > Columns > ticket\_price'.

Here, I'm verifying that all the structural changes have been applied successfully. The column in the booking table is now named ticket\_price.

The screenshot shows the PostgreSQL Object Explorer on the left, with the 'booking' table selected under the 'public' schema. The table's columns are listed: booking\_id, flight\_id, passenger\_id, booking\_platform, created\_at, updated\_at, status, and ticket\_price. The Query Editor on the right displays a query: `SELECT * FROM public.booking ORDER BY booking_id ASC;`. The Data Output pane at the bottom shows the table's schema: booking\_id (PK) integer, flight\_id integer, passenger\_id integer, booking\_platform character varying (50), created\_at timestamp without time zone, updated\_at timestamp without time zone, status character varying (50), and ticket\_price numeric (7,2). A green 'Successfully' message is visible in the bottom right corner.

Here, I'm verifying that all the structural changes have been successfully applied, and the flights table now includes the departing\_gate column with the new text data type.

The screenshot shows the PostgreSQL Object Explorer on the left, with the 'flights' table selected under the 'public' schema. The table's columns are listed: 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 Query Editor on the right displays a query: `SELECT * FROM public.flights ORDER BY flight_id ASC;`. The Data Output pane at the bottom shows the table's schema: flight\_id (PK) integer, sch\_departure\_time timestamp without time zone, sch\_arrival\_time timestamp without time zone, departing\_airport\_id integer, arriving\_airport\_id integer, departing\_gate text, arriving\_gate character varying (50), airline\_id integer, and act\_departure\_time timestamp without time zone. The status bar at the bottom indicates 'Total rows: 0' and 'Query complete 00:00:00.078'.

Here, I'm verifying that all the structural changes to the airline table have been successfully applied. I'm also confirming that the info column has been successfully dropped.

The screenshot shows the PostgreSQL Object Explorer on the left, displaying the 'airline' table under the 'public' schema. The table has 8 columns: `booking_id`, `flight_id`, `passenger_id`, `booking_platform`, `created_at`, `updated_at`, `status`, and `ticket_price`. The main query editor shows a query that selects all columns from the `public.airline` table, ordered by `airline_id` in ascending order. The query is executed successfully, and the Data Output pane shows the table structure with data types: `airline_id` (integer), `airline_code` (character varying (30)), `airline_name` (character varying (50)), `airline_country` (character varying (50)), `created_at` (timestamp without time zone), and `updated_at` (timestamp without time zone). A green message at the bottom indicates: 'Successfully run. Total query runtime: 75 msec. 0 rows affected.'

Here, I've generated 200 random records and inserted them into the airport table, which is necessary for future data operations.

The screenshot shows the PostgreSQL Object Explorer on the left, displaying the 'airport' table under the 'public' schema. The table has 7 columns: `airport_id`, `airport_name`, `country`, `state`, `city`, `created_at`, and `updated_at`. The main query editor shows a PL/pgSQL function `generate_random_string` that generates a random string of a specified length. The function is then used to insert 200 random records into the `airport` table. The query is executed successfully, and the Data Output pane shows the insertion of 200 records. A message at the bottom indicates: 'Query returned successfully in 183 msec.'

Here, I'm verifying that 200 random records have been successfully added to the airport table and that the data has been correctly generated.

The screenshot shows a PostgreSQL client interface with a query window and a data output window. The query window contains the following SQL query:

```
SELECT * FROM airport LIMIT 10;
```

The data output window displays the results of the query, showing 10 rows of data. The columns are: airport\_id, airport\_name, country, state, city, created\_at, and updated\_at. The data is as follows:

airport_id	airport_name	country	state	city	created_at	updated_at
1	Airport 0kgpy	Country JQI6O	State edylj	City cmPwm	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
2	Airport 9zBnv	Country FthWK	State KKKUuk	City V0ny3	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
3	Airport 0brgp	Country FHNSu	State AEXJS	City FQzvu	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
4	Airport lI92t	Country Ebz9W	State DJU7C	City TrfSa	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
5	Airport 59WBh	Country Pkm8v	State gGbTZ	City hsmzn	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
6	Airport W0IOt	Country FMFbs	State mTIPX	City HYlww	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
7	Airport 83JcF	Country Rnl7I	State EsLj7	City 6CR0k	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
8	Airport 1I9Y1	Country 8GVqe	State QxDBy	City Fk813	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
9	Airport Llpqm	Country FsgCy	State qtuZl	City 1hmfp	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
10	Airport z05hj	Country 8ba8N	State pldHe	City Rfl1l	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458

The status bar at the bottom indicates: Total rows: 10 Query complete 00:00:00.100 CRLF Ln 1, Col 32.

Here, I'm adding a new airline, "KazAir," to the airline table.

The screenshot shows a PostgreSQL client interface with a query window and a data output window. The query window contains the following SQL query:

```
INSERT INTO airline (airline_code, airline_name, airline_country)
VALUES ('KZA', 'KazAir', 'Kazakhstan');
```

The data output window displays the results of the query, showing 1 row of data. The columns are: airline\_code, airline\_name, and airline\_country. The data is as follows:

airline_code	airline_name	airline_country
KZA	KazAir	Kazakhstan

The status bar at the bottom indicates: Total rows: Query complete 00:00:00.195 CRLF Ln 2, Col 38.

In the previous step, I added a new airline, and here I updated the data, changing its country to "Turkey".

The screenshot shows the PostgreSQL IDE interface. On the left, the 'Object Explorer' pane displays a tree view of the database schema. The 'airline' table is selected under the 'Tables (10)' category. The 'Columns (7)' for the 'airline' table are listed: 'airport\_id', 'airport\_name', 'country', 'state', 'city', 'created\_at', and 'updated\_at'. The main query editor displays the following SQL statement:

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

The 'Data Output' pane shows the result of the query: 'UPDATE 1'. A status bar at the bottom indicates 'Query returned successfully in 105 msec.' and 'Total rows: Query complete 00:00:00.105'.

Here, I've added three new airlines to the airline table in a single operation.

The screenshot shows the PostgreSQL IDE interface. On the left, the 'Object Explorer' pane displays a tree view of the database schema. The 'airline' table is selected under the 'Tables (10)' category. The 'Columns (7)' for the 'airline' table are listed: 'airport\_id', 'airport\_name', 'country', 'state', 'city', 'created\_at', and 'updated\_at'. The main query editor displays the following SQL statement:

```
1 INSERT INTO airline (airline_code, airline_name, airline_country)
2 VALUES
3 ('AER', 'AirEasy', 'France'),
4 ('FLH', 'FlyHigh', 'Brazil'),
5 ('FLY', 'FlyFly', 'Poland');
```

The 'Data Output' pane shows the result of the query: 'INSERT 0 3'. A status bar at the bottom indicates 'Query returned successfully in 85 msec.' and 'Total rows: Query complete 00:00:00.085'.

Here, I'm verifying the result, and we can see that all four airlines—"KazAir," "AirEasy," "FlyHigh," and "FlyFly"—have been successfully added to the table.

The screenshot shows the PostgreSQL IDE interface. On the left, the 'Object Explorer' pane displays the database structure, including the 'airline' table. The main query editor shows a simple query: `SELECT * FROM airline;`. Below the query editor, the 'Data Output' pane displays the results of the query as a table with 7 columns: `airline_id` (PK integer), `airline_code` (character varying (30)), `airline_name` (character varying (50)), `airline_country` (character varying (50)), `created_at` (timestamp without time zone), and `updated_at` (timestamp without time zone). The results show four rows of data for the airlines: KazAir (Turkey), AirEasy (France), FlyHigh (Brazil), and FlyFly (Poland).

airline_id	airline_code	airline_name	airline_country	created_at	updated_at
1	KZA	KazAir	Turkey	2025-10-01 02:35:29.782219	2025-10-01 02:35:29.782219
2	AER	AirEasy	France	2025-10-01 02:41:41.026088	2025-10-01 02:41:41.026088
3	FLH	FlyHigh	Brazil	2025-10-01 02:41:41.026088	2025-10-01 02:41:41.026088
4	FLY	FlyFly	Poland	2025-10-01 02:41:41.026088	2025-10-01 02:41:41.026088

Here, I've added several flights to the flights table. Now I have the data needed to perform the deletion and update tasks.

The screenshot shows the PostgreSQL IDE interface. The main query editor contains an `INSERT INTO` statement: `INSERT INTO flights (sch_departure_time, sch_arrival_time, departing_airport_id, arriving_airport_id, departi`. The 'Data Output' pane shows the message: `INSERT 0 5`. A green notification box at the bottom right states: 'Query returned successfully in 107 msec.'.



Here, I'm deleting all flights scheduled to arrive in 2024. The query successfully deleted 3 records from the table.

The screenshot shows the PostgreSQL IDE interface. On the left, the Object Explorer displays the database schema, including tables like 'airline', 'airport', 'baggage', 'baggage\_check', 'boarding\_pass', and 'booking'. The 'airline' table is selected. The main query editor displays the following SQL query:

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

The 'Data Output' tab shows the result of the query:

```
DELETE 3
Query returned successfully in 80 msec.
```

A green status bar at the bottom right indicates: "Query returned successfully in 80 msec." The status bar also shows "Total rows: Query complete 00:00:00.080" and "CRLF Ln 2, Col 50".

Here, I'm checking the result after deleting the 2024 flights. As we can see, only two flights remain in the table, arriving in 2025 and 2023.

The screenshot shows the PostgreSQL IDE interface. The 'Data Output' tab displays the result of a SELECT query:

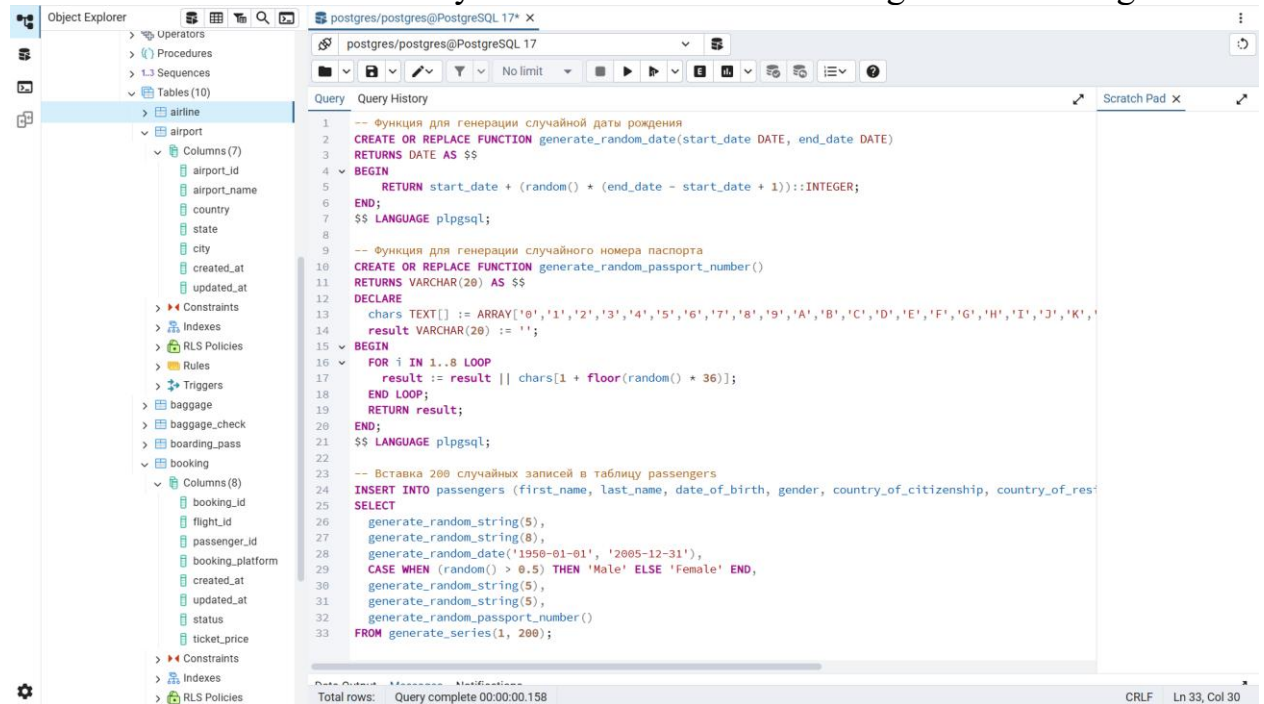
```
1 SELECT * FROM public.flights
2 ORDER BY flight_id ASC;
```

The query result is shown in a table with the following columns: flight\_id (PK integer), sch\_departure\_time (timestamp without time zone), sch\_arrival\_time (timestamp without time zone), departing\_airport\_id (integer), arriving\_airport\_id (integer), departing\_gate (text), arriving\_gate (character varying (50)), airline\_id (integer), and act\_d (timestamp). The table contains two rows:

flight_id (PK integer)	sch_departure_time (timestamp without time zone)	sch_arrival_time (timestamp without time zone)	departing_airport_id (integer)	arriving_airport_id (integer)	departing_gate (text)	arriving_gate (character varying (50))	airline_id (integer)	act_d (timestamp)
3	2025-01-20 09:00:00	2025-01-20 12:15:00	5	6	E3	F3	3	2025
5	2023-11-25 10:00:00	2023-11-25 13:20:00	9	10	I5	J5	4	2023

The status bar at the bottom shows "Total rows: 2" and "Query complete 00:00:00.151". The status bar also shows "CRLF Ln 2, Col 24".

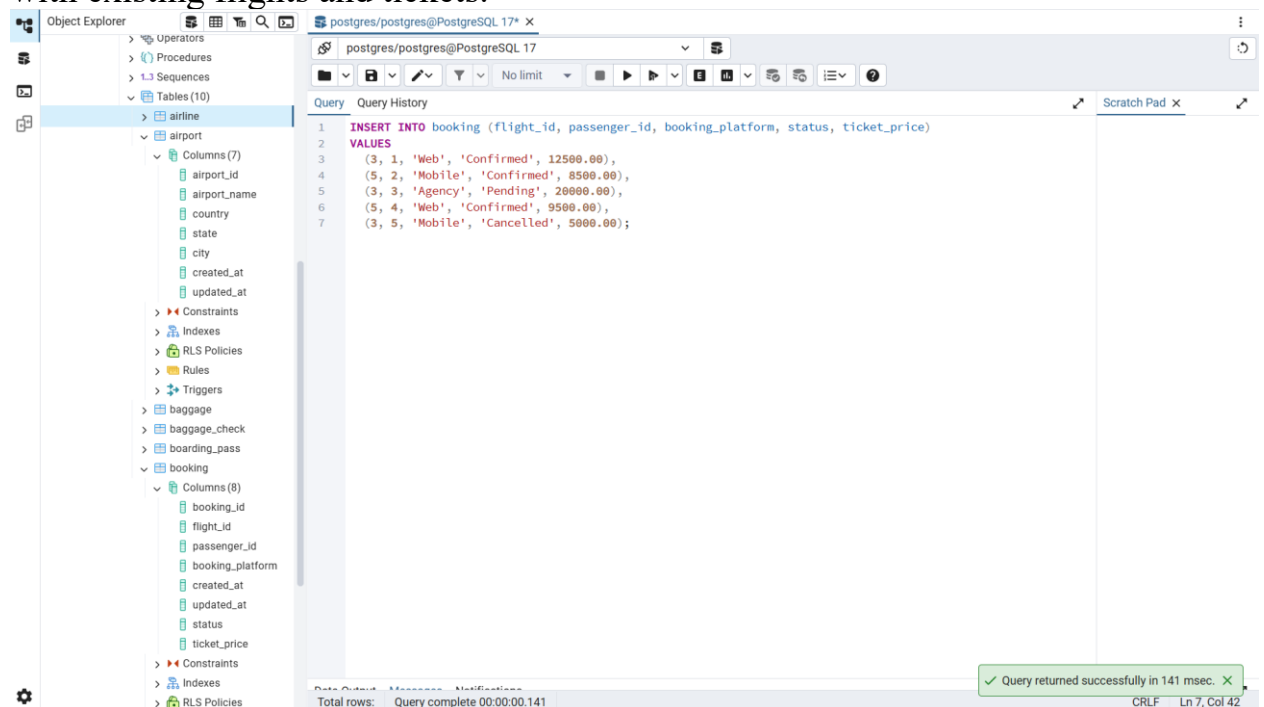
Here, I've generated 200 random records and inserted them into the passengers table. This ensures that the necessary data is available for creating future bookings.



The screenshot shows the PostgreSQL IDE interface. On the left, the Object Explorer displays the database schema, including tables like 'airline', 'airport', 'baggage', 'boarding\_pass', and 'booking'. The 'passengers' table is highlighted. The main query editor displays a SQL script that defines two functions: 'generate\_random\_date' and 'generate\_random\_passport\_number'. The script then inserts 200 random records into the 'passengers' table. The status bar at the bottom indicates 'Query complete 00:00:00.158'.

```
1 -- Функция для генерации случайной даты рождения
2 CREATE OR REPLACE FUNCTION generate_random_date(start_date DATE, end_date DATE)
3 RETURNS DATE AS $$
4 BEGIN
5     RETURN start_date + (random() * (end_date - start_date + 1))::INTEGER;
6 END;
7 $$ LANGUAGE plpgsql;
8
9 -- Функция для генерации случайного номера паспорта
10 CREATE OR REPLACE FUNCTION generate_random_passport_number()
11 RETURNS VARCHAR(20) AS $$
12 DECLARE
13     chars TEXT[] := ARRAY['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'];
14     result VARCHAR(20) := '';
15 BEGIN
16     FOR i IN 1..8 LOOP
17         result := result || chars[1 + floor(random() * 36)];
18     END LOOP;
19     RETURN result;
20 END;
21 $$ LANGUAGE plpgsql;
22
23 -- Вставка 200 случайных записей в таблицу passengers
24 INSERT INTO passengers (first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence)
25 SELECT
26     generate_random_string(5),
27     generate_random_string(8),
28     generate_random_date('1956-01-01', '2005-12-31'),
29     CASE WHEN (random() > 0.5) THEN 'Male' ELSE 'Female' END,
30     generate_random_string(5),
31     generate_random_string(5),
32     generate_random_passport_number()
33 FROM generate_series(1, 200);
```

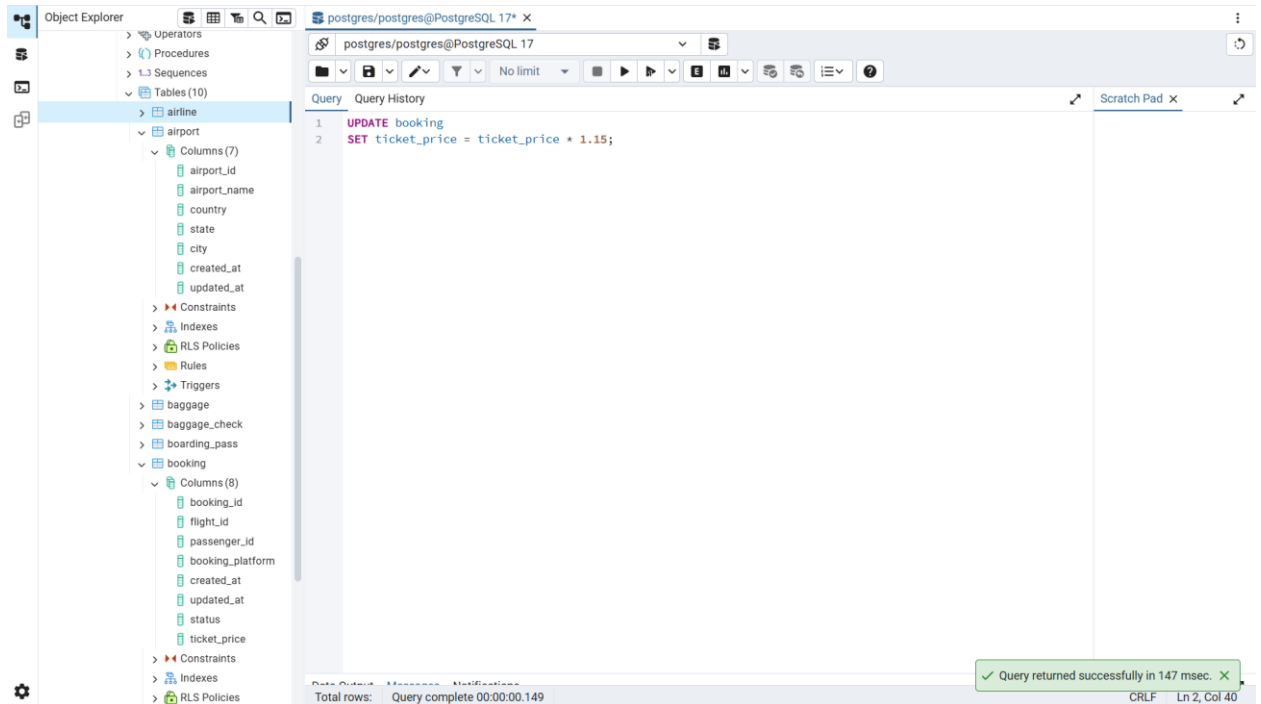
Here, I'm adding five records to the booking table. These records link passengers with existing flights and tickets.



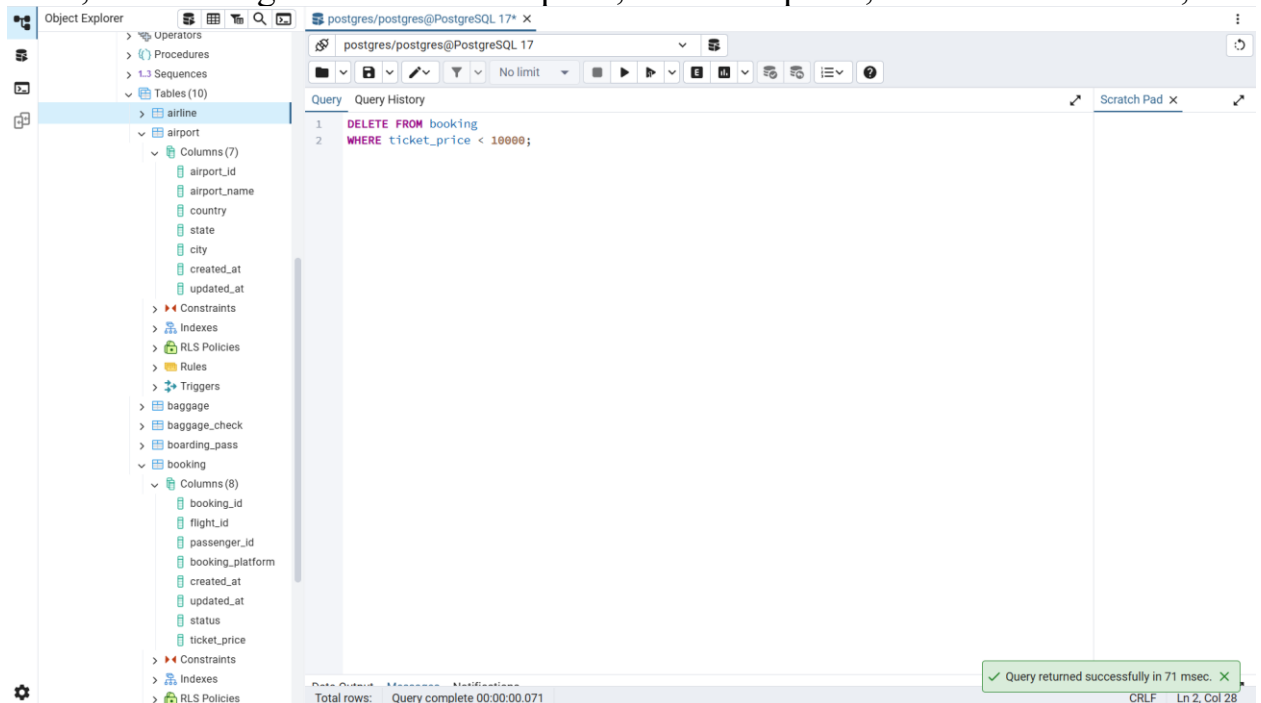
The screenshot shows the PostgreSQL IDE interface. On the left, the Object Explorer displays the database schema, including tables like 'airline', 'airport', 'baggage', 'boarding\_pass', and 'booking'. The 'booking' table is highlighted. The main query editor displays a SQL script that inserts five records into the 'booking' table. The status bar at the bottom indicates 'Query complete 00:00:00.141'.

```
1 INSERT INTO booking (flight_id, passenger_id, booking_platform, status, ticket_price)
2 VALUES
3 (3, 1, 'Web', 'Confirmed', 12500.00),
4 (5, 2, 'Mobile', 'Confirmed', 8500.00),
5 (3, 3, 'Agency', 'Pending', 20000.00),
6 (5, 4, 'Web', 'Confirmed', 9500.00),
7 (3, 5, 'Mobile', 'Cancelled', 5000.00);
```

Here, I'm updating the data in the booking table, increasing the price of each ticket by 15%.



Here, I'm deleting all tickets whose price, after the update, became less than 10,000.



Here, I'm showing the final result. After increasing the prices and deleting tickets that cost less than 10,000, only these three records remain in the table.

Object Explorer

Operators

Procedures

Sequences

Tables (10)

airline

airport

Columns (7)

airport\_id

airport\_name

country

state

city

created\_at

updated\_at

Constraints

Indexes

RLS Policies

Rules

Triggers

baggage

baggage\_check

boarding\_pass

booking

Columns (8)

booking\_id

flight\_id

passenger\_id

booking\_platform

created\_at

updated\_at

status

ticket\_price

Constraints

Indexes

RLS Policies

postgres/postgres@PostgreSQL 17\*

postgres/postgres@PostgreSQL 17

Query

Query History

1 SELECT \* FROM booking;

Data Output

Messages

Notifications

Showing rows: 1 to 3

Page No: 1

of 1

SQL

	booking_id [PK] integer	flight_id integer	passenger_id integer	booking_platform character varying (50)	created_at timestamp without time zone	updated_at timestamp without time zone	status character varying (50)	ticket_price numeric (7,2)
1	20	3	1	Web	2025-10-01 03:32:21.507441	2025-10-01 03:32:21.507441	Confirmed	14375.00
2	22	3	3	Agency	2025-10-01 03:32:21.507441	2025-10-01 03:32:21.507441	Pending	23000.00
3	23	5	4	Web	2025-10-01 03:32:21.507441	2025-10-01 03:32:21.507441	Confirmed	10925.00

✓ Successfully run. Total query runtime: 97 msec. 3 rows affected. ✕

Total rows: 3

Query complete 00:00:00.097

CRLF

Ln 1, Col 23