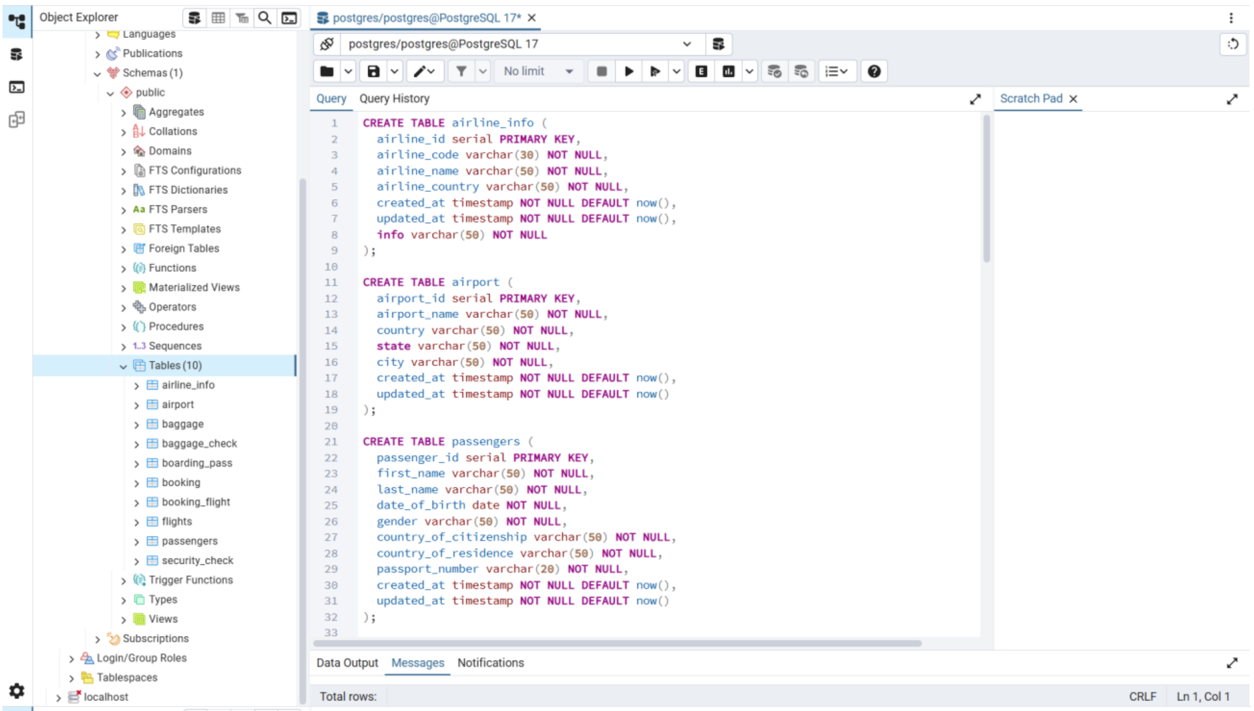
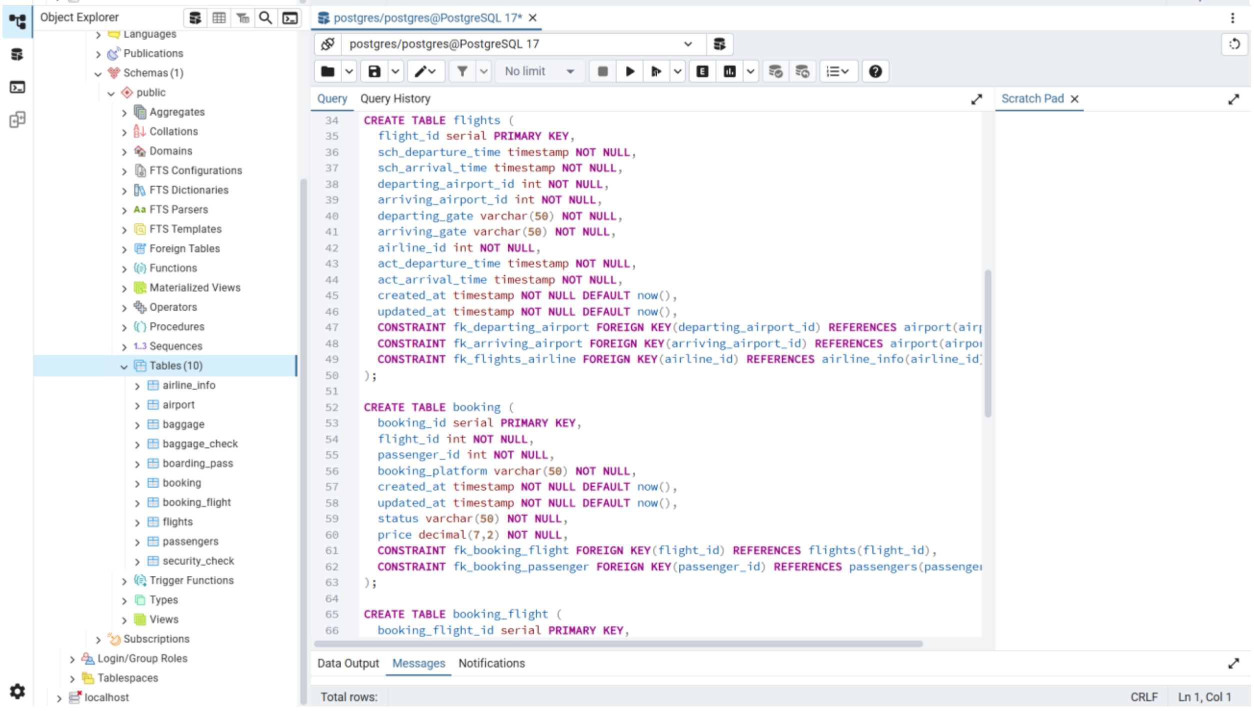
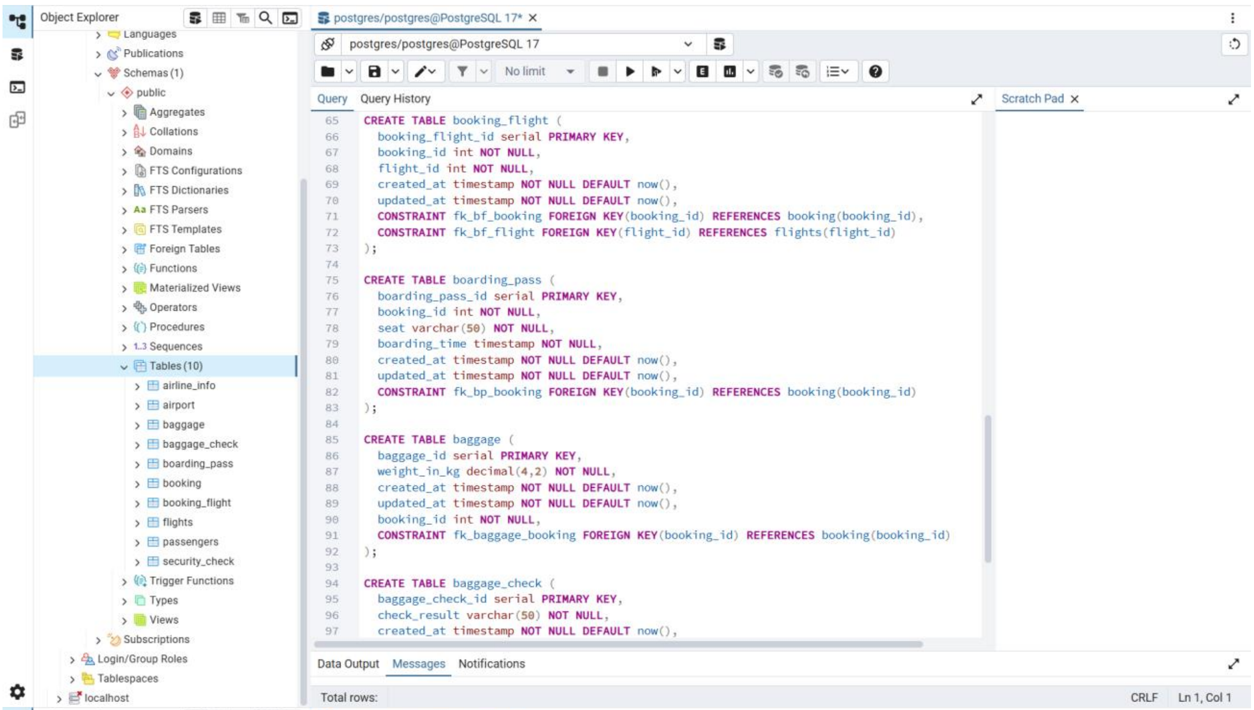
Here, I've created all the necessary tables, defined their columns, and established the primary keys. This forms the foundation of my database.



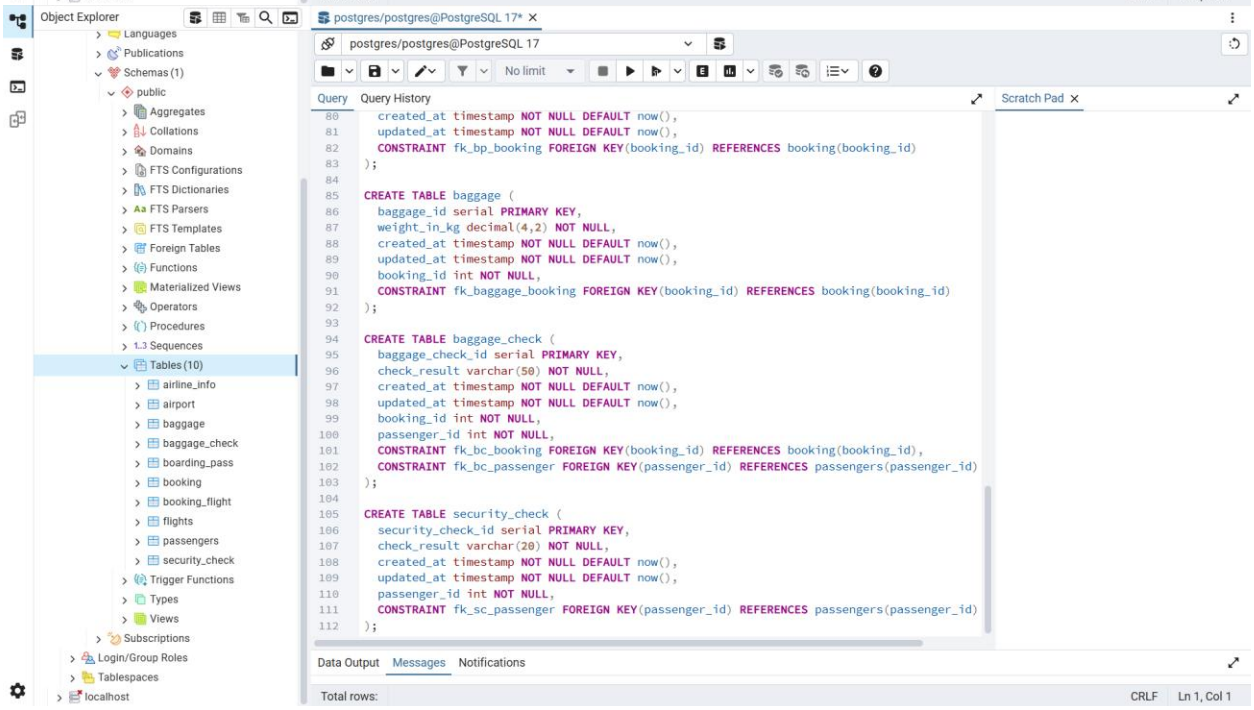
```sql
CREATE TABLE airline_info (
  airline_id serial PRIMARY KEY,
  airline_code varchar(30) NOT NULL,
  airline_name varchar(50) NOT NULL,
  airline_country varchar(50) NOT NULL,
  created_at timestamp NOT NULL DEFAULT now(),
  updated_at timestamp NOT NULL DEFAULT now(),
  info varchar(50) NOT NULL
);

CREATE TABLE airport (
  airport_id serial 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 DEFAULT now(),
  updated_at timestamp NOT NULL DEFAULT now()
);

CREATE TABLE passengers (
  passenger_id serial 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 DEFAULT now(),
  updated_at timestamp NOT NULL DEFAULT now()
);

```



```sql
CREATE TABLE flights (
  flight_id serial PRIMARY KEY,
  sch_departure_time timestamp NOT NULL,
  sch_arrival_time timestamp NOT NULL,
  departing_airport_id int NOT NULL,
  arriving_airport_id int NOT NULL,
  departing_gate varchar(50) NOT NULL,
  arriving_gate varchar(50) NOT NULL,
  airline_id int NOT NULL,
  act_departure_time timestamp NOT NULL,
  act_arrival_time timestamp NOT NULL,
  created_at timestamp NOT NULL DEFAULT now(),
  updated_at timestamp NOT NULL DEFAULT now(),
  CONSTRAINT fk_departing_airport FOREIGN KEY(departing_airport_id) REFERENCES airport(air
  CONSTRAINT fk_arriving_airport FOREIGN KEY(arriving_airport_id) REFERENCES airport(airpo
  CONSTRAINT fk_flights_airline FOREIGN KEY(airline_id) REFERENCES airline_info(airline_id
);

CREATE TABLE booking (
  booking_id serial PRIMARY KEY,
  flight_id int NOT NULL,
  passenger_id int NOT NULL,
  booking_platform varchar(50) NOT NULL,
  created_at timestamp NOT NULL DEFAULT now(),
  updated_at timestamp NOT NULL DEFAULT now(),
  status varchar(50) NOT NULL,
  price decimal(7,2) NOT NULL,
  CONSTRAINT fk_booking_flight FOREIGN KEY(flight_id) REFERENCES flights(flight_id),
  CONSTRAINT fk_booking_passenger FOREIGN KEY(passenger_id) REFERENCES passengers(passenge
);

CREATE TABLE booking_flight (
  booking_flight_id serial PRIMARY KEY,
```

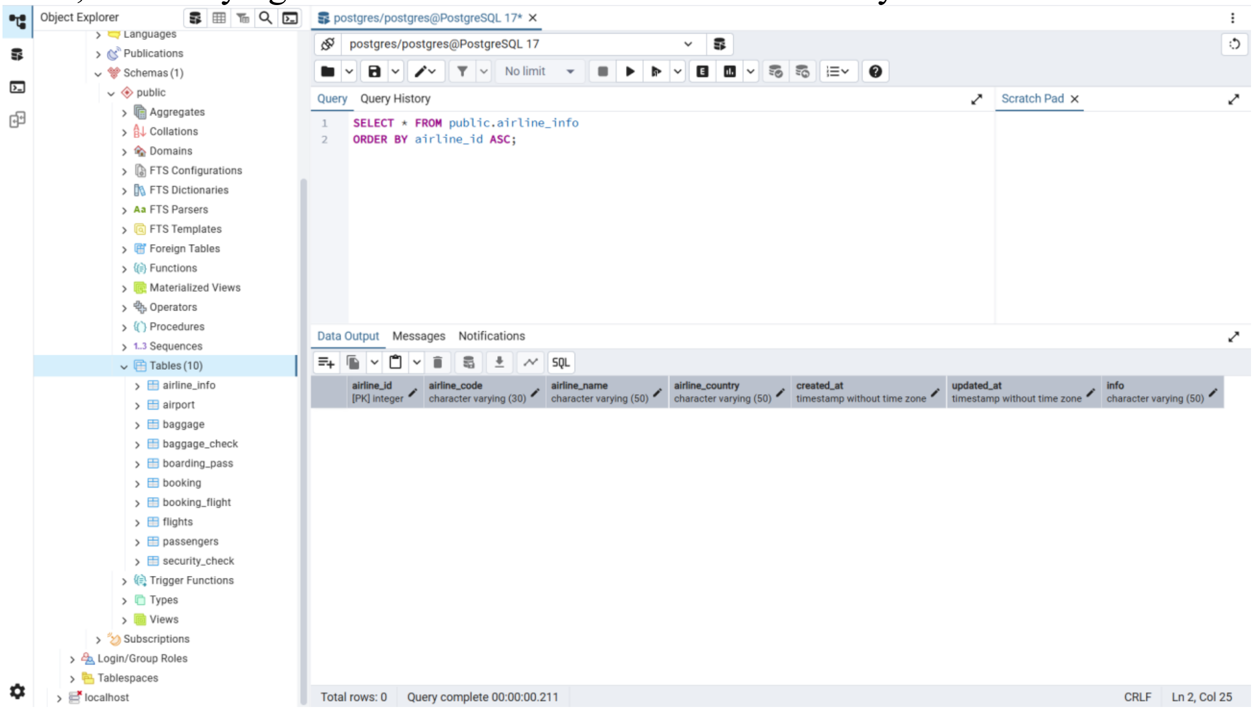


```
CREATE TABLE booking_flight (
  booking_flight_id serial PRIMARY KEY,
  booking_id int NOT NULL,
  flight_id int NOT NULL,
  created_at timestamp NOT NULL DEFAULT now(),
  updated_at timestamp NOT NULL DEFAULT now(),
  CONSTRAINT fk_bf_booking FOREIGN KEY(booking_id) REFERENCES booking(booking_id),
  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_baggage_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(),
```



```
  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_baggage_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)
);
```

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

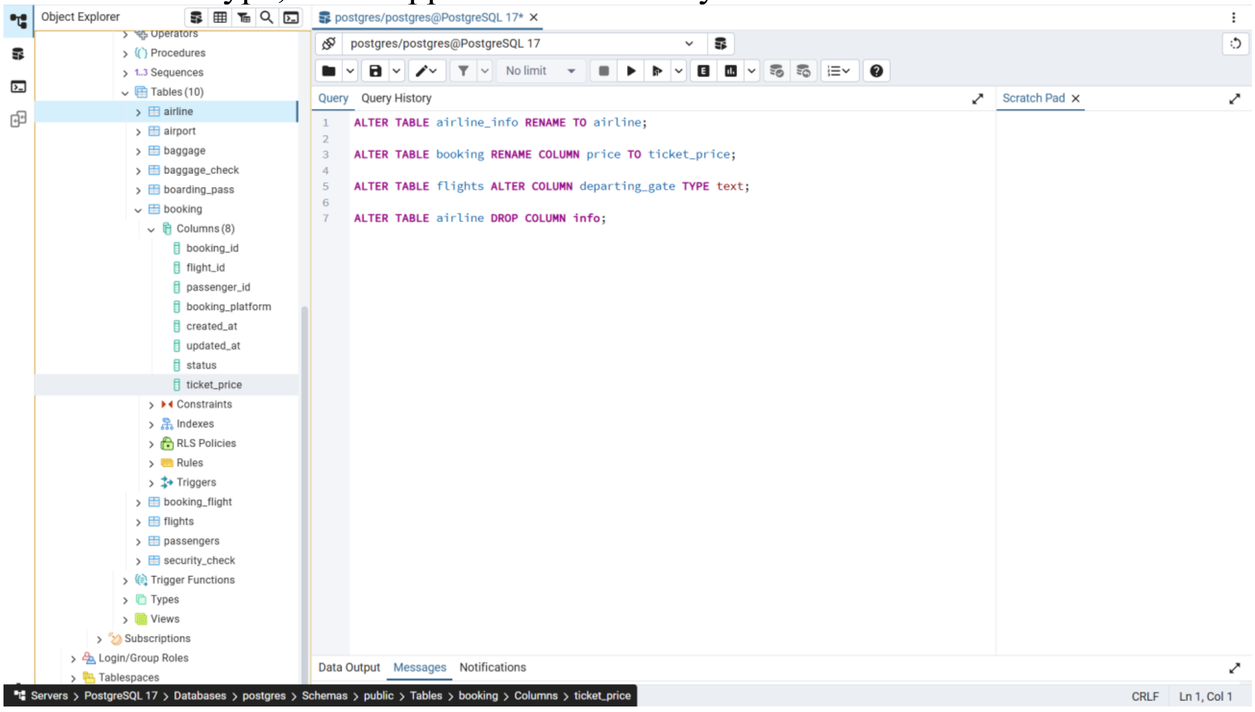


```sql
SELECT * FROM public.airline_info
ORDER BY airline_id ASC;
```

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	info character varying (50)

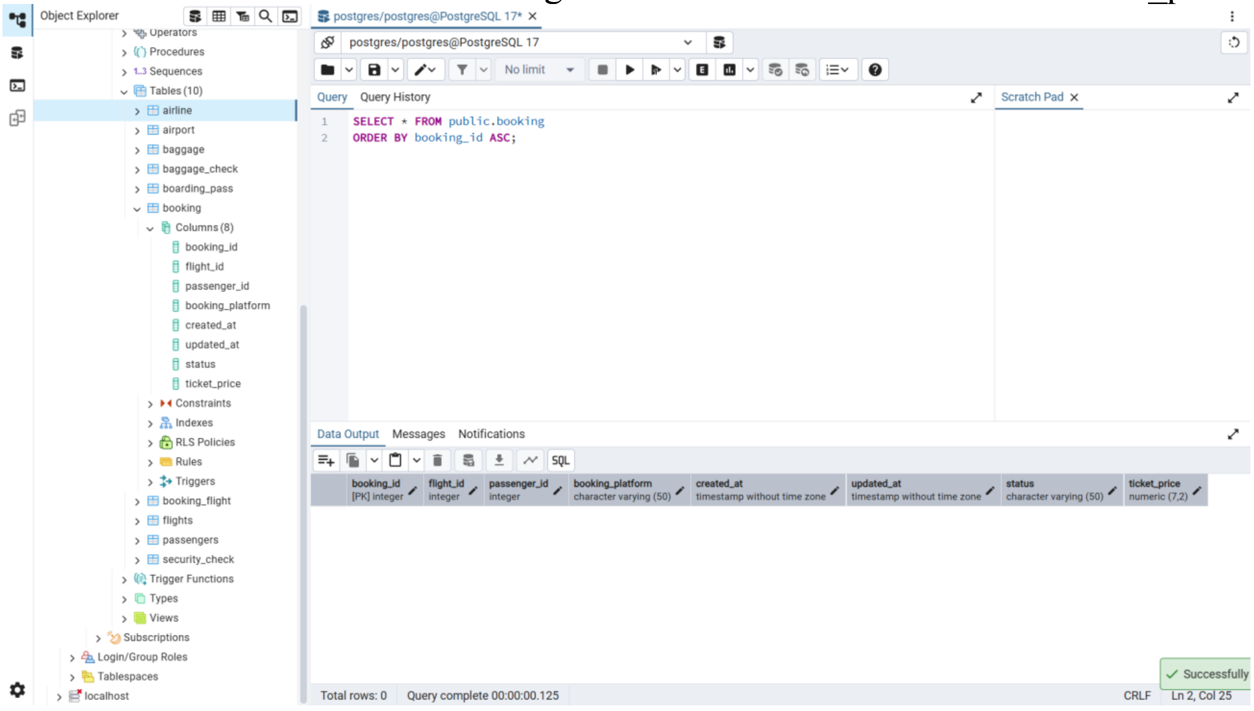
Total rows: 0 Query complete 00:00:00.211

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.



```sql
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;
```

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

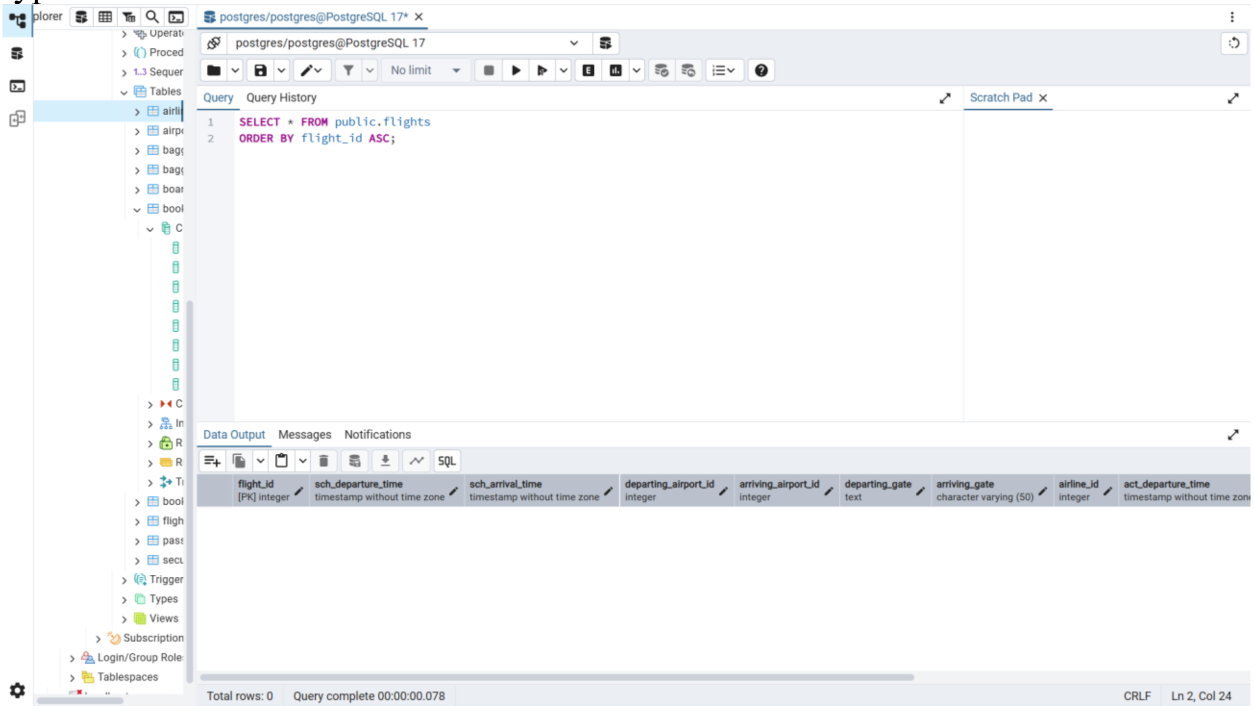


```sql
SELECT * FROM public.booking
ORDER BY booking_id ASC;
```

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)

Total rows: 0 Query complete 00:00:00.125

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.

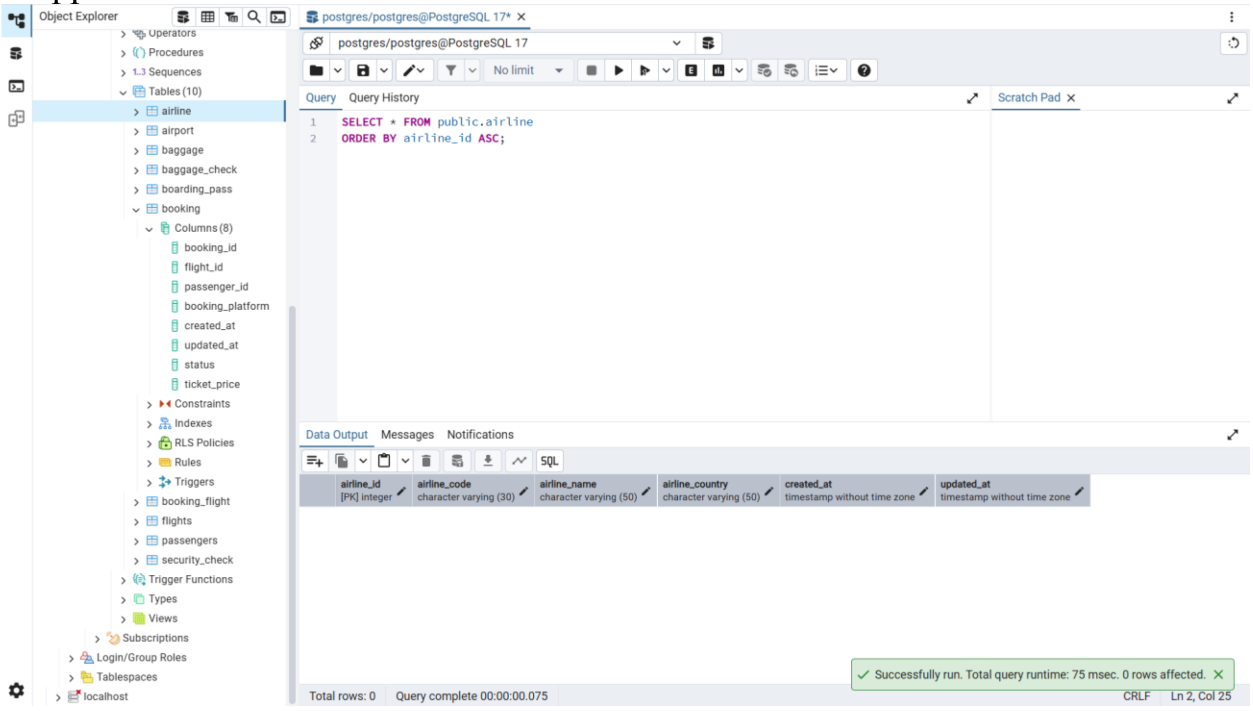


```sql
SELECT * FROM public.flights
ORDER BY flight_id ASC;
```

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_departure_time timestamp without time zon

Total rows: 0 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.

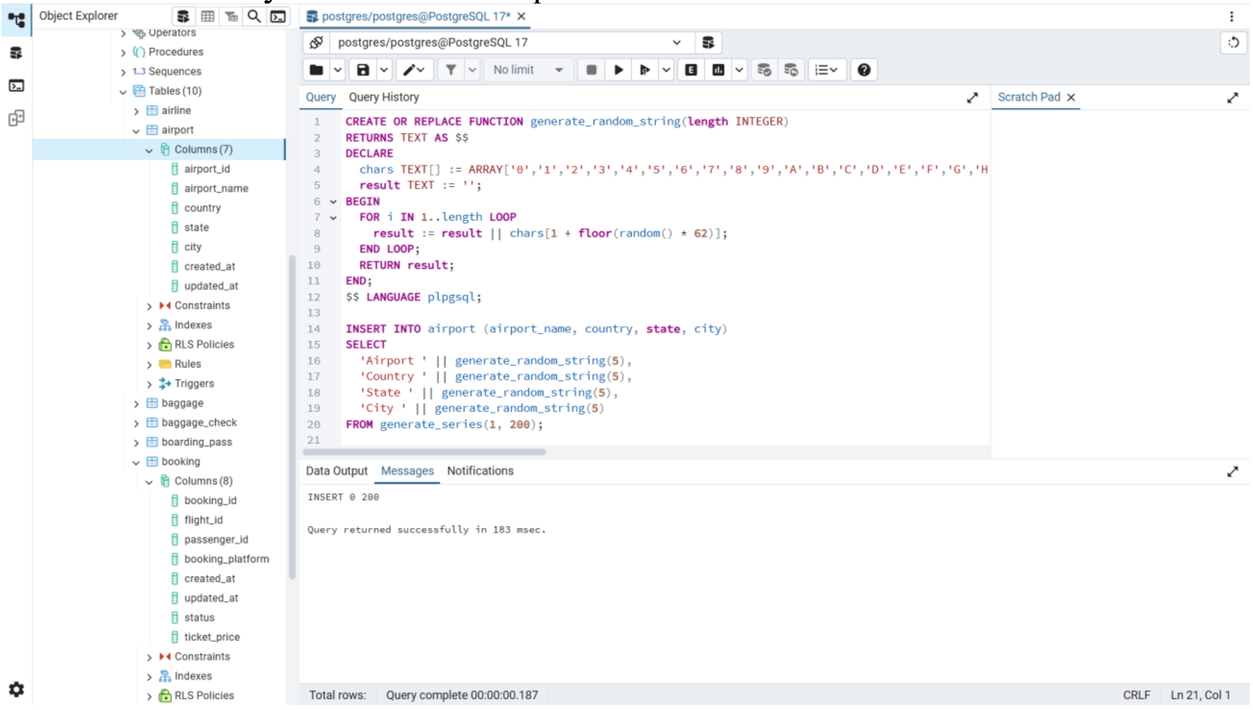


```
SELECT * FROM public.airline
ORDER BY airline_id ASC;
```

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

```
Successfully run. Total query runtime: 75 msec. 0 rows affected.
Total rows: 0    Query complete 00:00:00.075
```

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

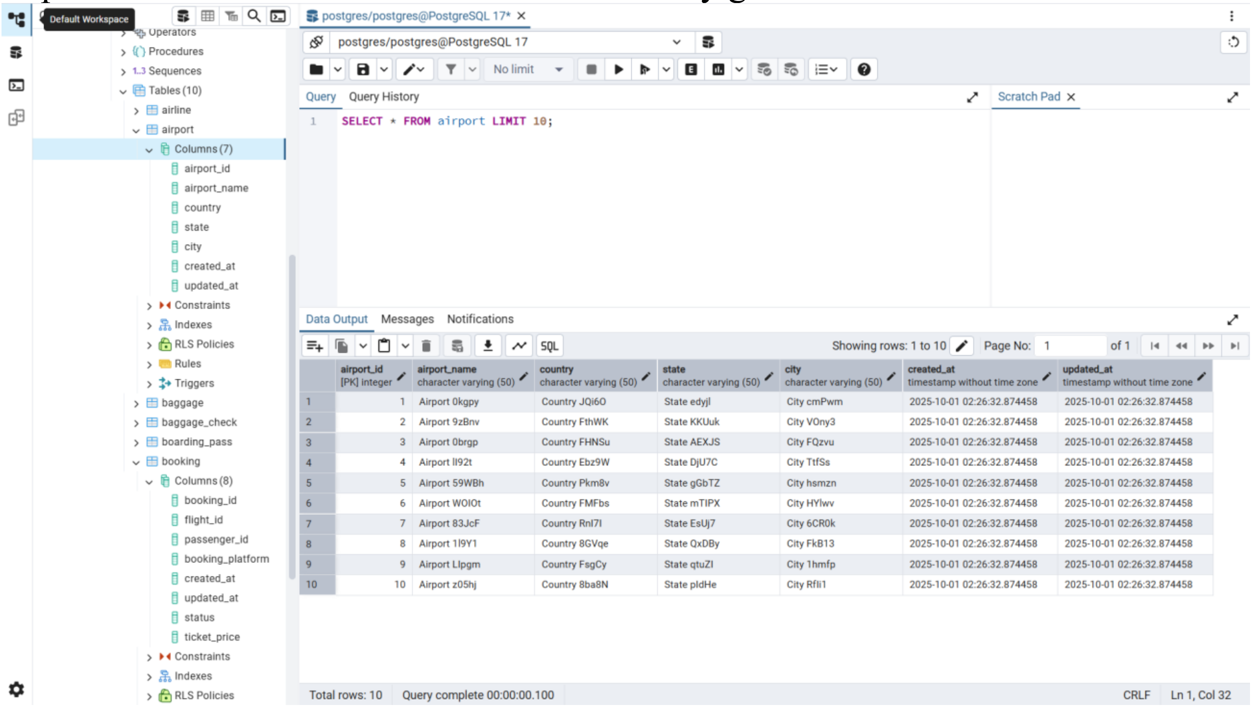


```
CREATE OR REPLACE FUNCTION generate_random_string(length INTEGER)
RETURNS TEXT AS $$
DECLARE
  chars TEXT[] := ARRAY['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H
  result TEXT := '';
BEGIN
  FOR i IN 1..length LOOP
    result := result || chars[1 + floor(random() * 62)];
  END LOOP;
  RETURN result;
END;
$$ LANGUAGE plpgsql;

INSERT INTO airport (airport_name, country, state, city)
SELECT
  'Airport ' || generate_random_string(5),
  'Country ' || generate_random_string(5),
  'State ' || generate_random_string(5),
  'City ' || generate_random_string(5)
FROM generate_series(1, 200);
```

```
INSERT 0 200

Query returned successfully in 183 msec.
```

```
Total rows:    Query complete 00:00:00.187
```

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



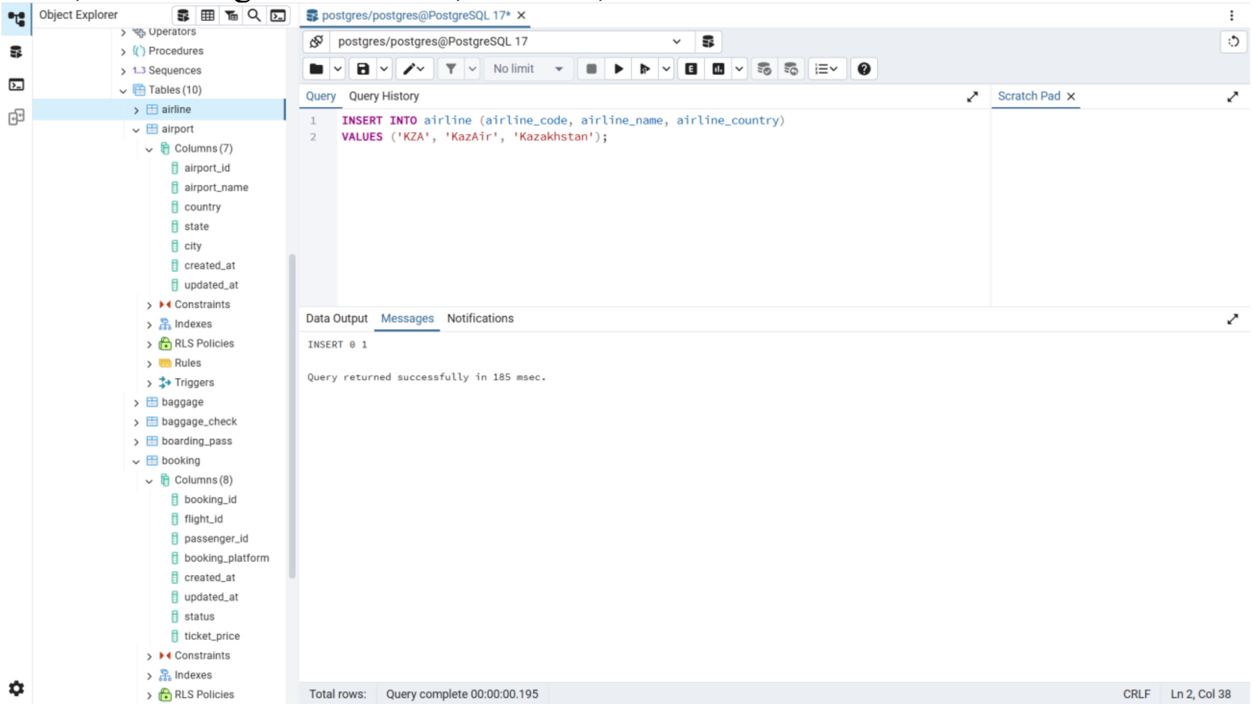
```
SELECT * FROM airport LIMIT 10;
```

	airport_id [PK] integer	airport_name character varying (50)	country character varying (50)	state character varying (50)	city character varying (50)	created_at timestamp without time zone	updated_at timestamp without time zone
1	1	Airport 0kgpy	Country JQi6O	State edyjl	City cmPwm	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
2	2	Airport 9zBnv	Country FthWK	State KKUuk	City VOny3	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
3	3	Airport 0brgp	Country FHNSu	State AEXJS	City FQzvu	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
4	4	Airport ll92t	Country Ebz9W	State DjU7C	City TtfSs	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
5	5	Airport 59WBh	Country Pkm8v	State gGbTZ	City hsmzn	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
6	6	Airport WOIOt	Country FMFbs	State mTIPX	City HYlwv	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
7	7	Airport 83JcF	Country RnI7I	State EsUj7	City 6CR0k	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
8	8	Airport 1l9Y1	Country 8GVqe	State QxDBy	City FkB13	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
9	9	Airport Llpgm	Country FsgCy	State qtuZI	City 1hmfp	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458
10	10	Airport z05hj	Country 8ba8N	State pldHe	City RfIi1	2025-10-01 02:26:32.874458	2025-10-01 02:26:32.874458

Showing rows: 1 to 10 Page No: 1 of 1

Total rows: 10 Query complete 00:00:00.100

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

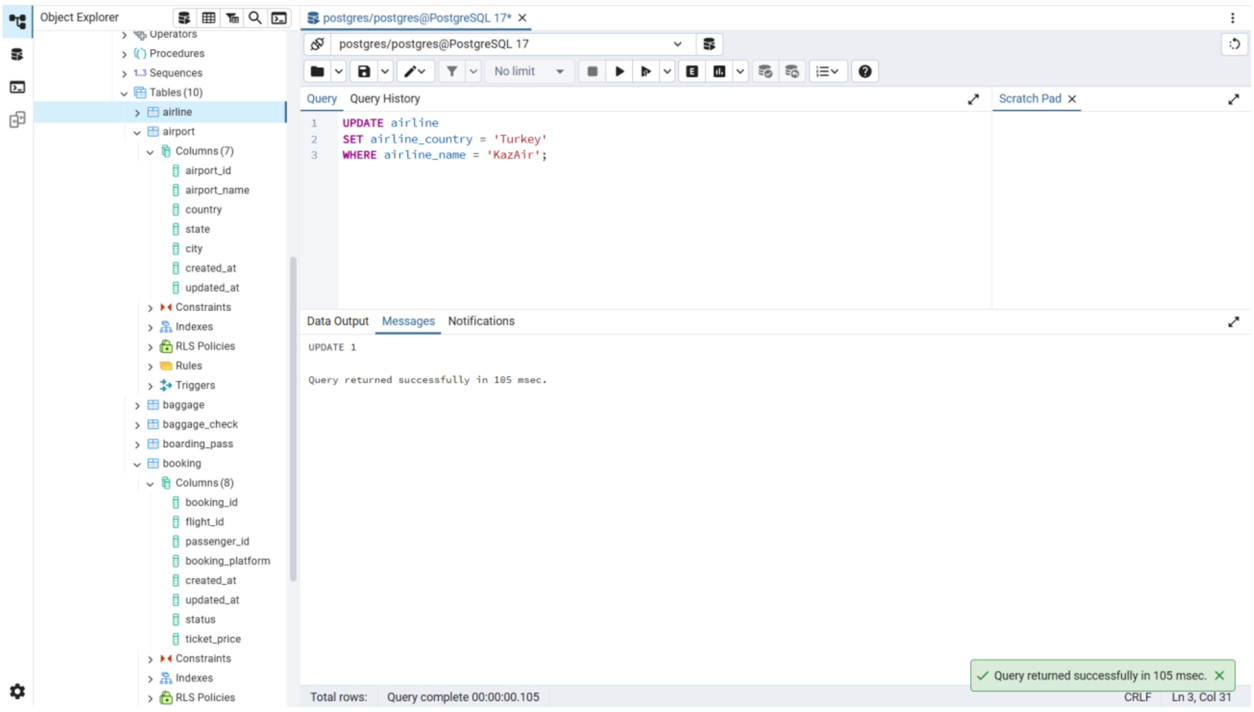


```
INSERT INTO airline (airline_code, airline_name, airline_country)
VALUES ('KZA', 'KazAir', 'Kazakhstan');
```

```
INSERT 0 1

Query returned successfully in 185 msec.
```

Total rows: Query complete 00:00:00.195

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



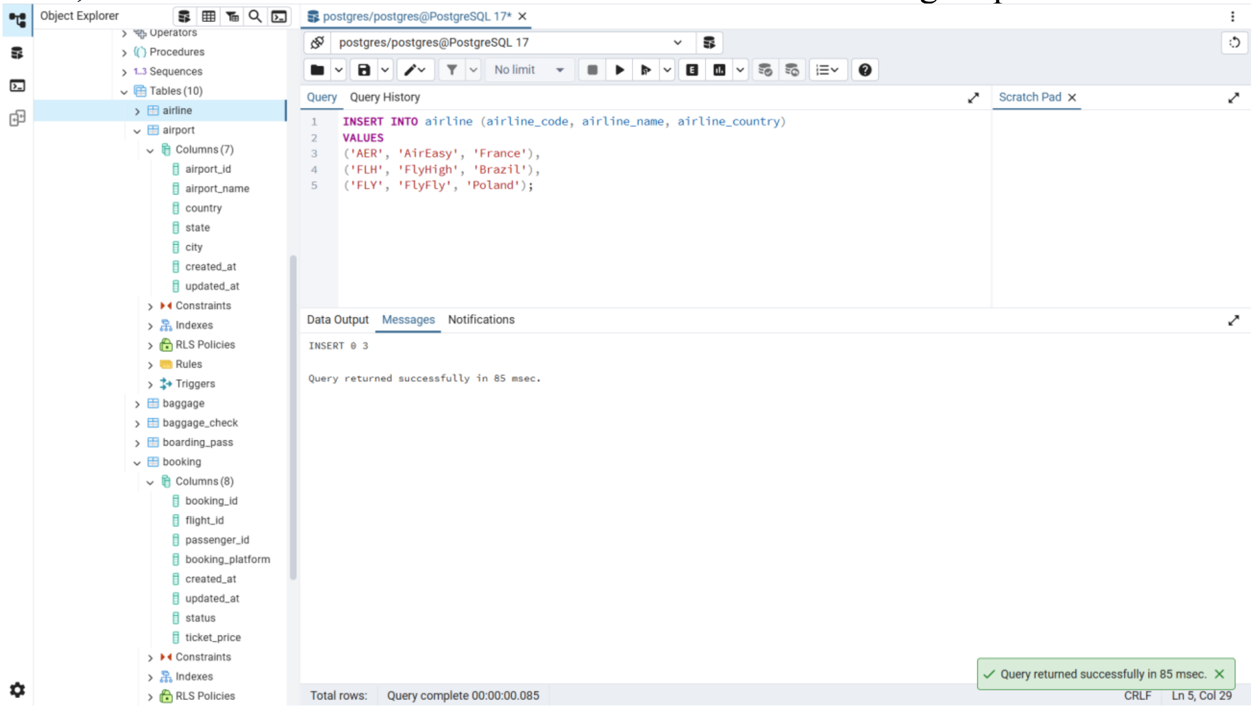
```
UPDATE airline
SET airline_country = 'Turkey'
WHERE airline_name = 'KazAir';
```

```
UPDATE 1

Query returned successfully in 105 msec.
```

Query returned successfully in 105 msec.

Total rows: Query complete 00:00:00.105 CRLF Ln 3, Col 31

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



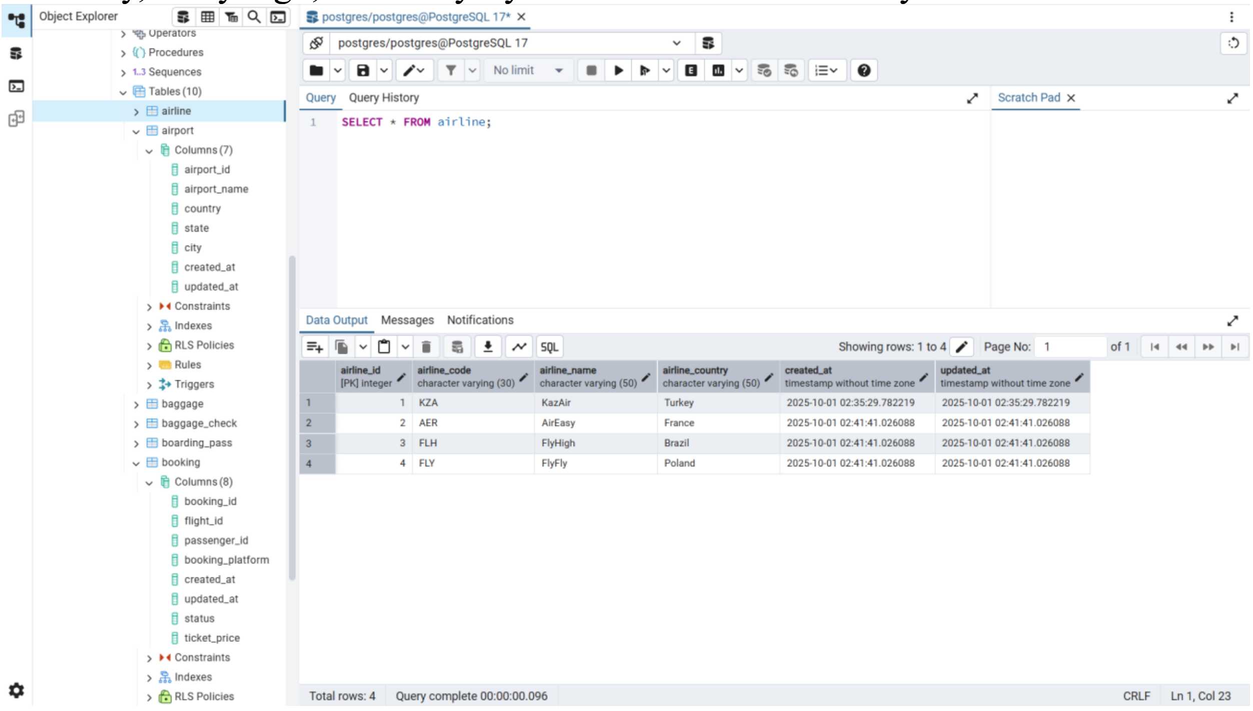
```
INSERT INTO airline (airline_code, airline_name, airline_country)
VALUES
('AER', 'AirEasy', 'France'),
('FLH', 'FlyHigh', 'Brazil'),
('FLY', 'FlyFly', 'Poland');
```

```
INSERT 0 3

Query returned successfully in 85 msec.
```

Query returned successfully in 85 msec.

Total rows: Query complete 00:00:00.085 CRLF Ln 5, Col 29

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.

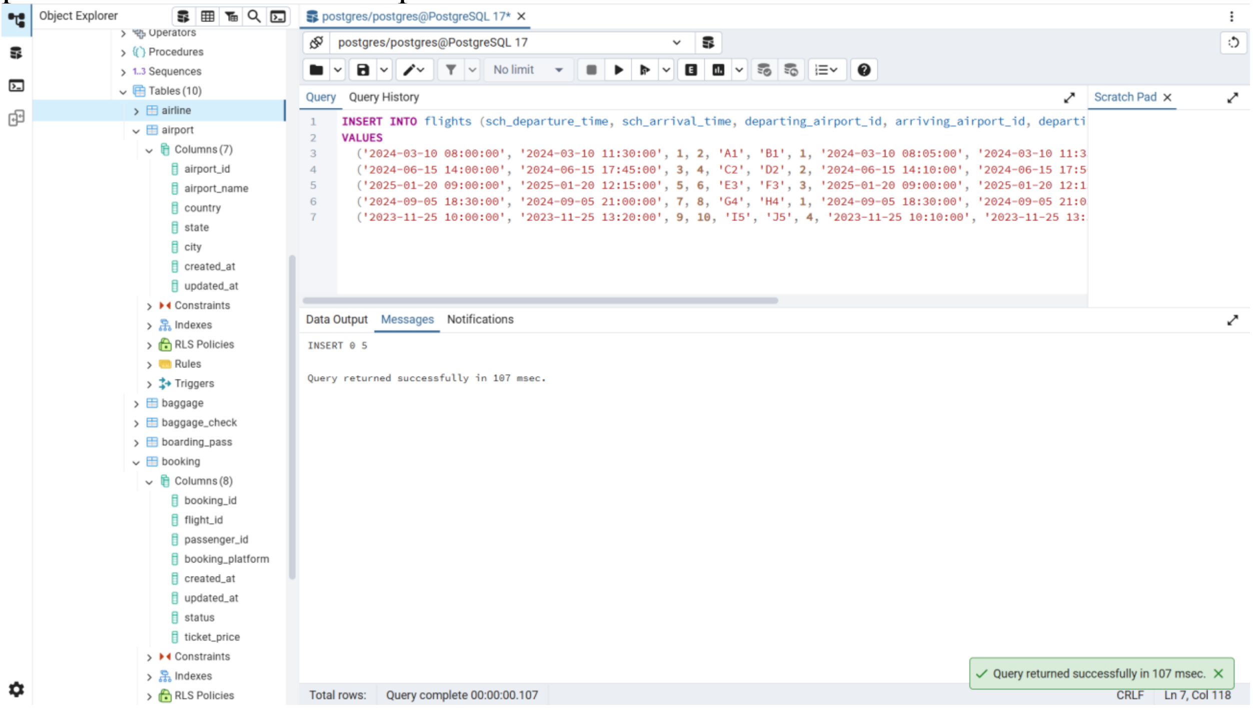


```sql
SELECT * FROM airline;
```

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

Total rows: 4 Query complete 00:00:00.096

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

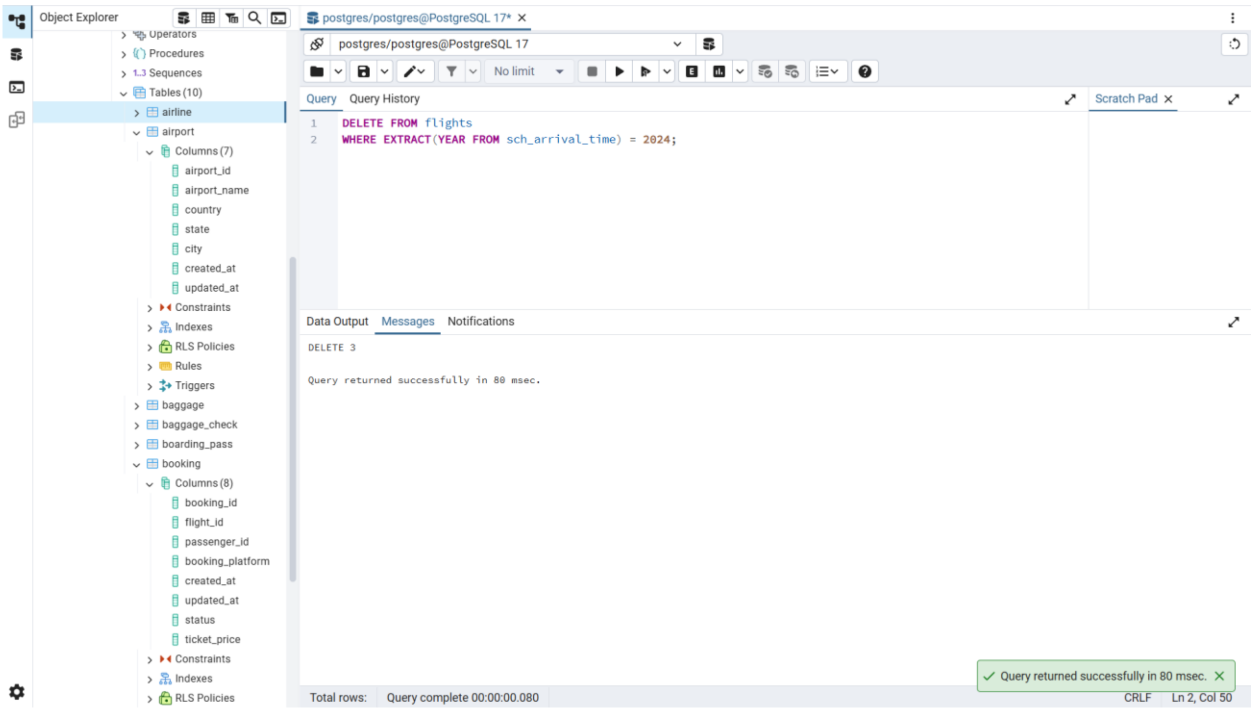


```sql
INSERT INTO flights (sch_departure_time, sch_arrival_time, departing_airport_id, arriving_airport_id, departi
VALUES
  ('2024-03-10 08:00:00', '2024-03-10 11:30:00', 1, 2, 'A1', 'B1', 1, '2024-03-10 08:05:00', '2024-03-10 11:3
  ('2024-06-15 14:00:00', '2024-06-15 17:45:00', 3, 4, 'C2', 'D2', 2, '2024-06-15 14:10:00', '2024-06-15 17:5
  ('2025-01-20 09:00:00', '2025-01-20 12:15:00', 5, 6, 'E3', 'F3', 3, '2025-01-20 09:00:00', '2025-01-20 12:1
  ('2024-09-05 18:30:00', '2024-09-05 21:00:00', 7, 8, 'G4', 'H4', 1, '2024-09-05 18:30:00', '2024-09-05 21:0
  ('2023-11-25 10:00:00', '2023-11-25 13:20:00', 9, 10, 'I5', 'J5', 4, '2023-11-25 10:10:00', '2023-11-25 13:
```

```
INSERT 0 5

Query returned successfully in 107 msec.
```

Total rows: Query complete 00:00:00.107

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

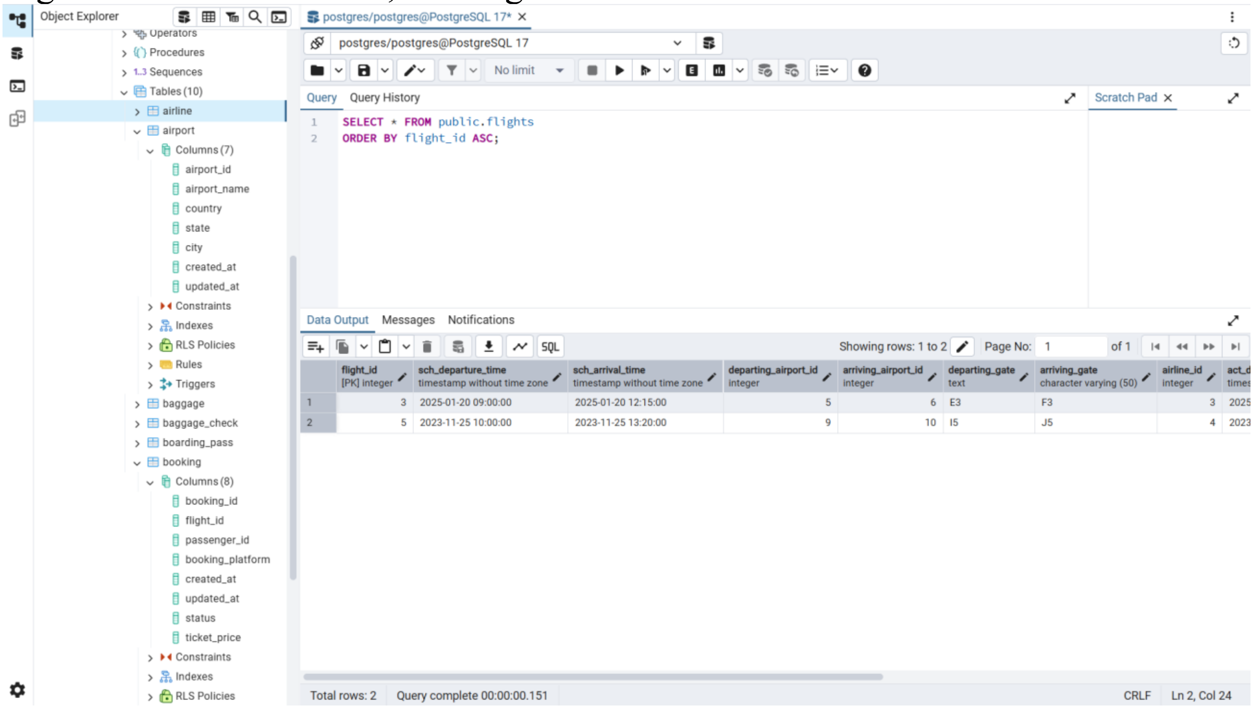


```sql
DELETE FROM flights
WHERE EXTRACT(YEAR FROM sch_arrival_time) = 2024;
```

```
DELETE 3

Query returned successfully in 80 msec.
```

Total rows: Query complete 00:00:00.080

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.

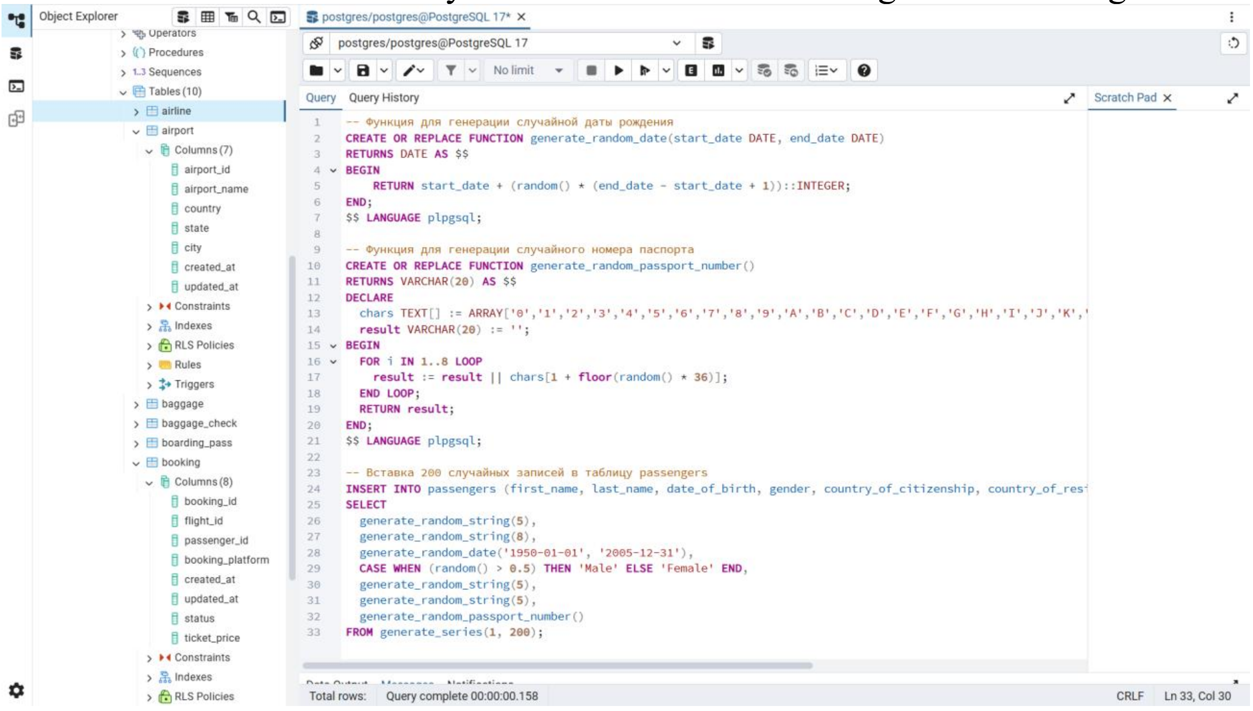


```sql
SELECT * FROM public.flights
ORDER BY flight_id ASC;
```

	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 times
1	3	2025-01-20 09:00:00	2025-01-20 12:15:00	5	6	E3	F3	3	2025
2	5	2023-11-25 10:00:00	2023-11-25 13:20:00	9	10	I5	J5	4	2023

Total rows: 2 Query complete 00:00:00.151

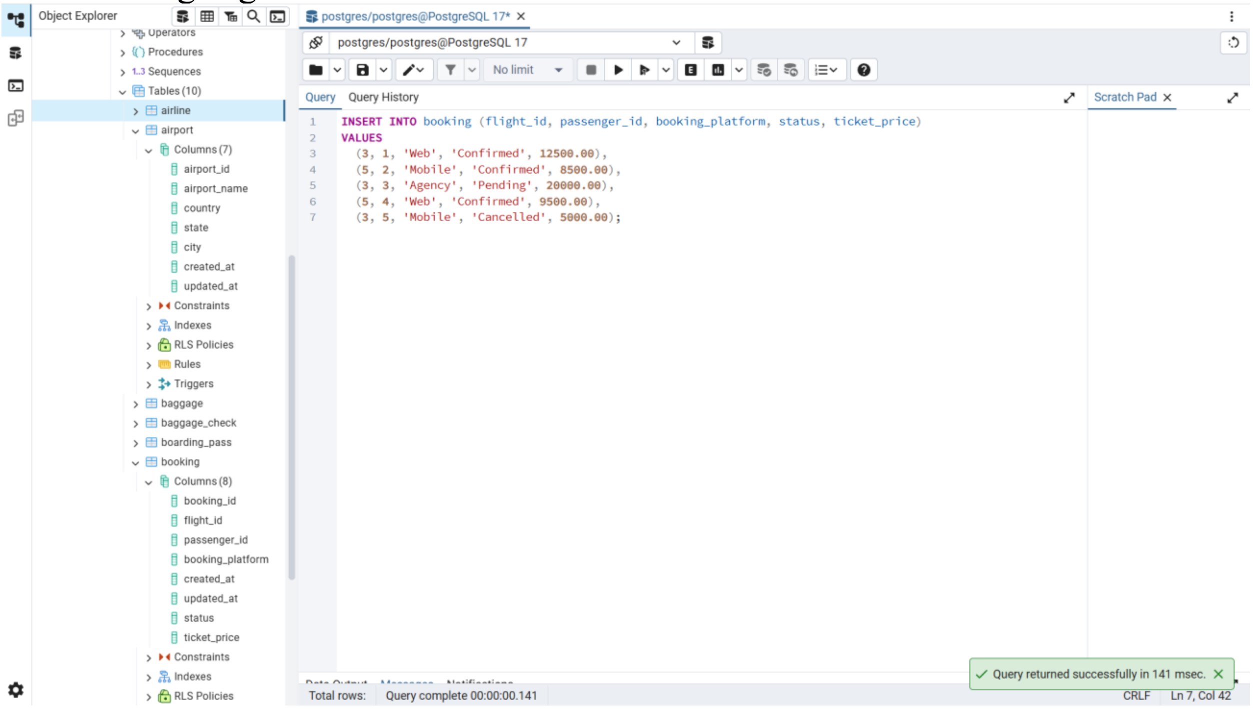
Here, I've generated 200 random records and inserted them into the passengers table. This ensures that the necessary data is available for creating future bookings.



```
-- Функция для генерации случайной даты рождения
CREATE OR REPLACE FUNCTION generate_random_date(start_date DATE, end_date DATE)
RETURNS DATE AS $$
BEGIN
    RETURN start_date + (random() * (end_date - start_date + 1))::INTEGER;
END;
$$ LANGUAGE plpgsql;

-- Функция для генерации случайного номера паспорта
CREATE OR REPLACE FUNCTION generate_random_passport_number()
RETURNS VARCHAR(20) AS $$
DECLARE
  chars TEXT[] := ARRAY['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F','G','H','I','J','K','
  result VARCHAR(20) := '';
BEGIN
  FOR i IN 1..8 LOOP
    result := result || chars[1 + floor(random() * 36)];
  END LOOP;
  RETURN result;
END;
$$ LANGUAGE plpgsql;

-- Вставка 200 случайных записей в таблицу passengers
INSERT INTO passengers (first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_res
SELECT
  generate_random_string(5),
  generate_random_string(8),
  generate_random_date('1950-01-01', '2005-12-31'),
  CASE WHEN (random() > 0.5) THEN 'Male' ELSE 'Female' END,
  generate_random_string(5),
  generate_random_string(5),
  generate_random_passport_number()
FROM generate_series(1, 200);
```

```
Total rows:    Query complete 00:00:00.158                    CRLF    Ln 33, Col 30
```

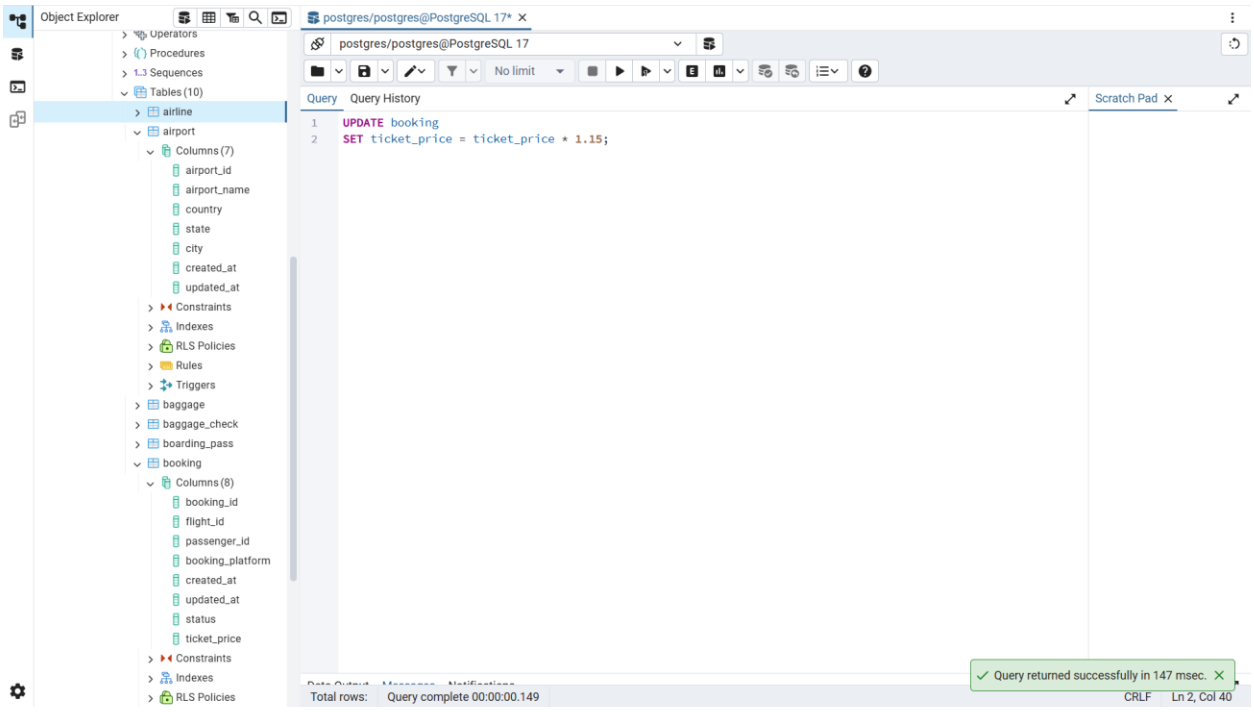
Here, I'm adding five records to the booking table. These records link passengers with existing flights and tickets.



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

```
Query returned successfully in 141 msec.
Total rows:    Query complete 00:00:00.141                    CRLF    Ln 7, Col 42
```

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

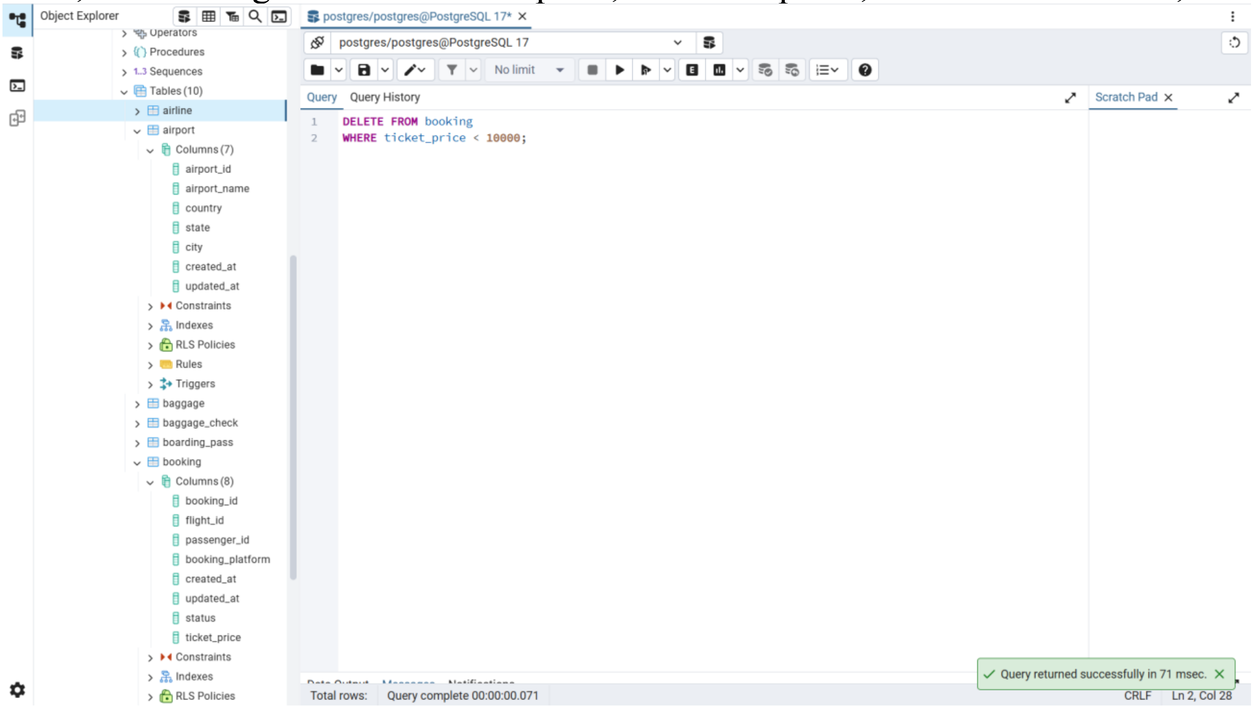


```
UPDATE booking
SET ticket_price = ticket_price * 1.15;
```

Query returned successfully in 147 msec.

Total rows: Query complete 00:00:00.149

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

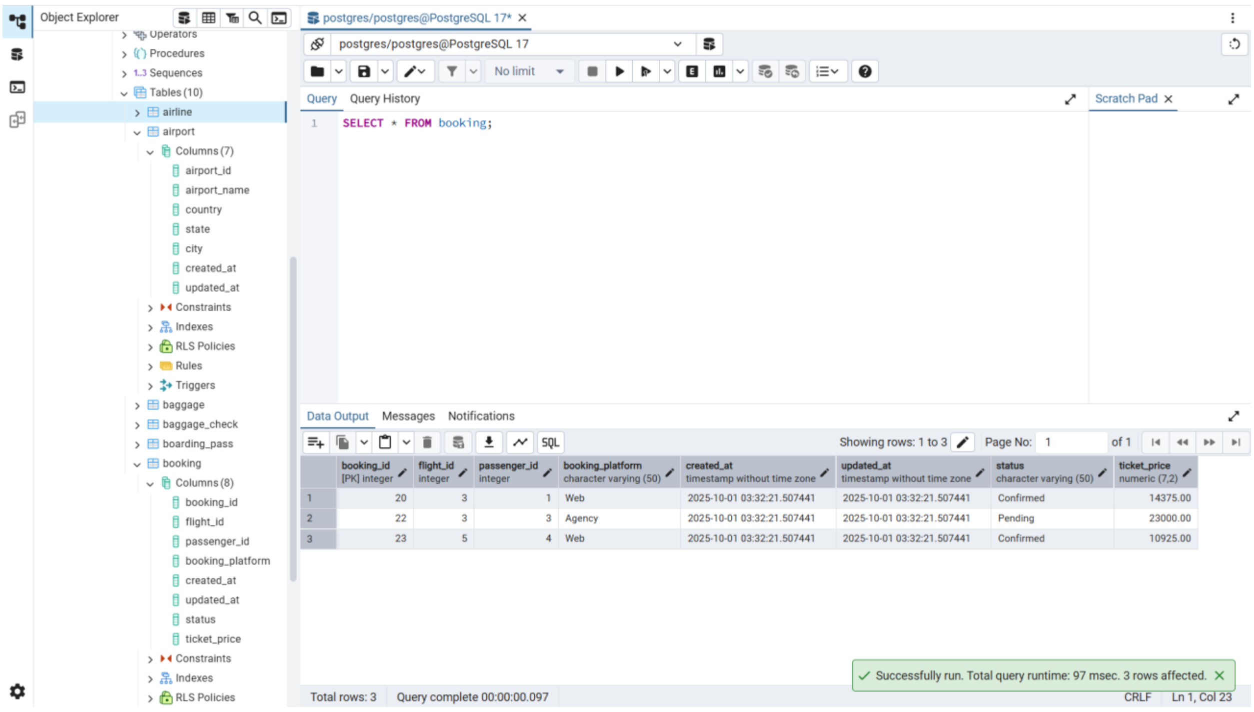


```
DELETE FROM booking
WHERE ticket_price < 10000;
```

Query returned successfully in 71 msec.

Total rows: Query complete 00:00:00.071

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.



```
SELECT * FROM booking;
```

	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
```