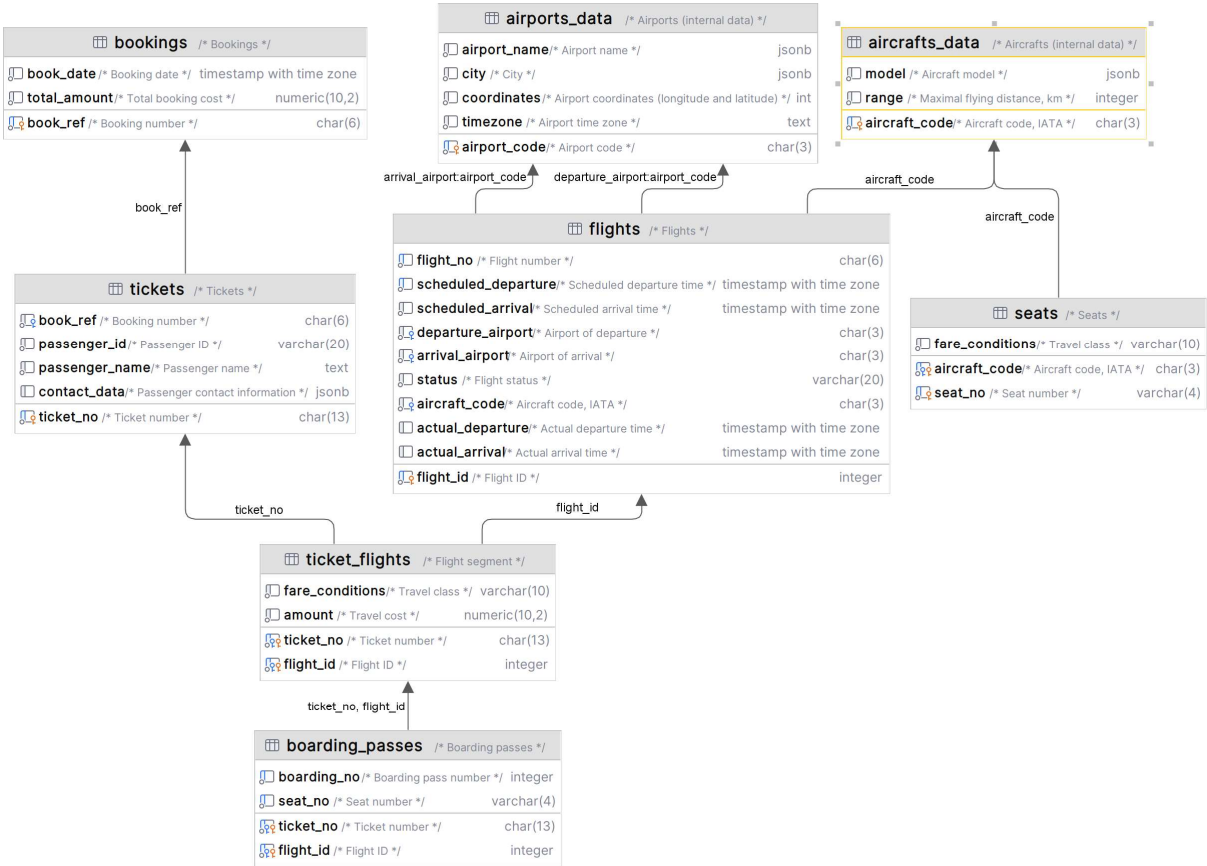


1. Применение dump

```
C:\WINDOWS\system32\cmd. X + v
C:\Users\Камила\Downloads>psql -f demo-medium-20170815.sql -U postgres
Пароль пользователя postgres:
SET
SET
SET
SET
SET
SET
SET
SET
psql:demo-medium-20170815.sql:17: P?PĖP?P'P?P?: P+P°P·P° P?P°P?P?C<C: "demo" P?Pч C?C?C%PчC?C'P?C?PчC'
CREATE DATABASE
Вы подключены к базе данных "demo" как пользователь "postgres".
SET
SET
SET
SET
SET
SET
SET
SET
CREATE SCHEMA
COMMENT
CREATE EXTENSION
COMMENT
SET
CREATE FUNCTION
CREATE FUNCTION
COMMENT
SET
SET
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT
CREATE VIEW
COMMENT
COMMENT
COMMENT
COMMENT
CREATE TABLE
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
COMMENT
CREATE VIEW
COMMENT
```

C:\Users\Камила\Downloads>

Результат применения



SQL запросы

1) Объединить данные из двух произвольных таблиц используя UNION [ALL], отсортировать результат в порядке убывания

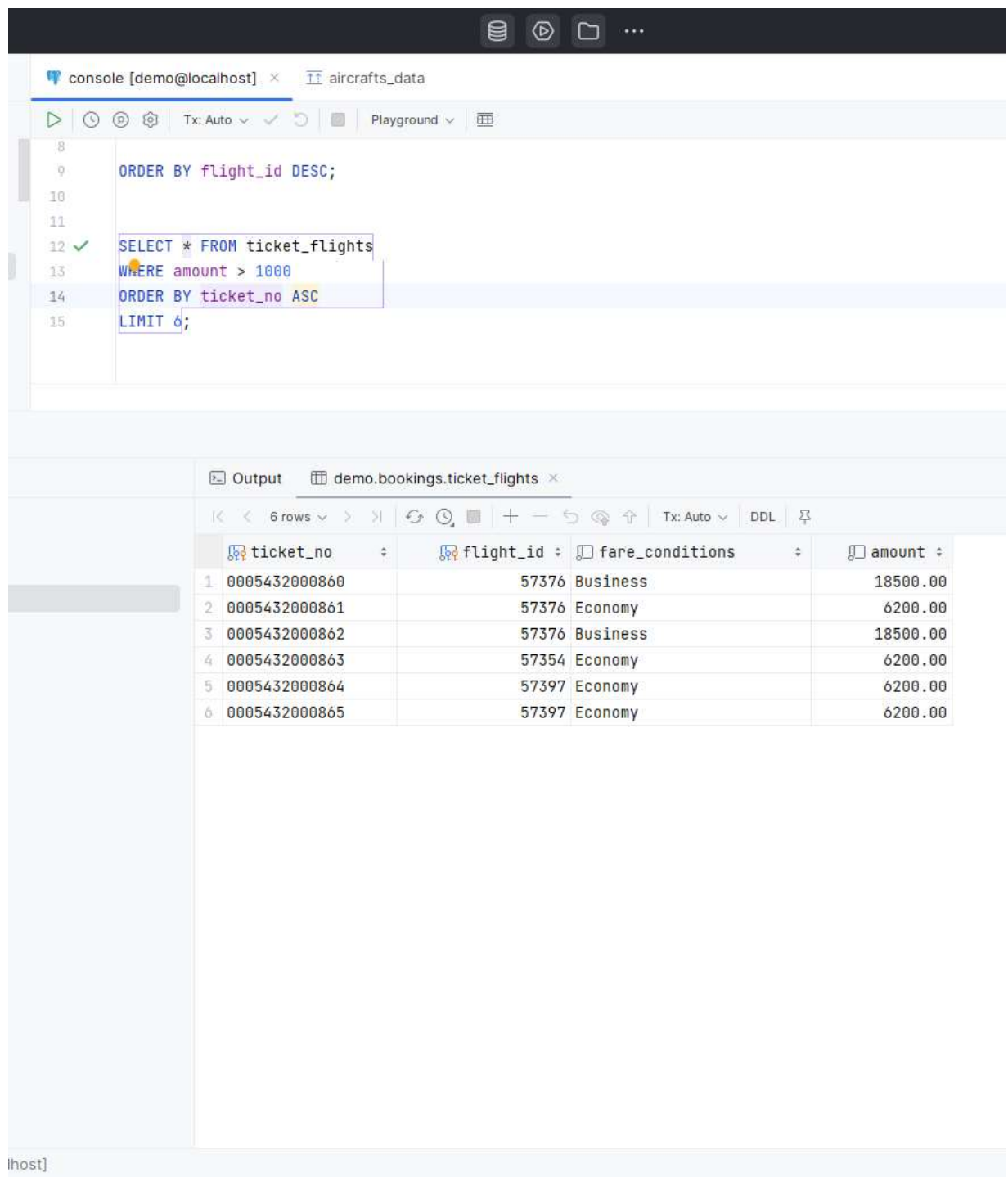
The screenshot shows a SQL playground interface with a query editor and an output table. The query is as follows:

```
1 SELECT fare_conditions, amount, ticket_no, flight_id
2 FROM ticket_flights
3
4 UNION ALL
5
6 SELECT NULL as fare_conditions, NULL as amount, ticket_no, flight_id
7 FROM boarding_passes
8
9 ORDER BY flight_id DESC;
```

The output table, titled "demo.bookings.ticket_flights", displays the results of the query. It has four columns: fare_conditions, amount, ticket_no, and flight_id. The results are sorted by flight_id in descending order. The first 18 rows show data from the ticket_flights table, and the remaining 31 rows show data from the boarding_passes table with NULL values for fare_conditions and amount.

| | fare_conditions | amount | ticket_no | flight_id |
|----|-----------------|--------|---------------|-----------|
| 28 | Economy | 5400 | 0005435132302 | 65662 |
| 29 | Economy | 5400 | 0005435382889 | 65662 |
| 30 | Economy | 5400 | 0005435382887 | 65662 |
| 31 | Economy | 5400 | 0005435132337 | 65662 |
| 32 | Economy | 5400 | 0005435132254 | 65662 |
| 33 | Economy | 5400 | 0005435132309 | 65662 |
| 34 | Economy | 5400 | 0005435132269 | 65662 |
| 35 | Economy | 5400 | 0005435132266 | 65662 |
| 36 | Economy | 5400 | 0005435132258 | 65662 |
| 37 | Economy | 5400 | 0005435382891 | 65662 |
| 38 | Economy | 5400 | 0005435132259 | 65662 |
| 39 | <null> | <null> | 0005435132337 | 65662 |
| 40 | <null> | <null> | 0005435132338 | 65662 |
| 41 | <null> | <null> | 0005435382892 | 65662 |
| 42 | <null> | <null> | 0005435382890 | 65662 |
| 43 | <null> | <null> | 0005435132255 | 65662 |
| 44 | <null> | <null> | 0005435132259 | 65662 |
| 45 | <null> | <null> | 0005435132258 | 65662 |
| 46 | <null> | <null> | 0005435382891 | 65662 |
| 47 | <null> | <null> | 0005435132266 | 65662 |
| 48 | <null> | <null> | 0005435132314 | 65662 |
| 49 | <null> | <null> | 0005435132272 | 65662 |

2) Запрос с любым фильтром WHERE к произвольной таблице, отсортировать результат с ограничением вывода по количеству строк (LIMIT)



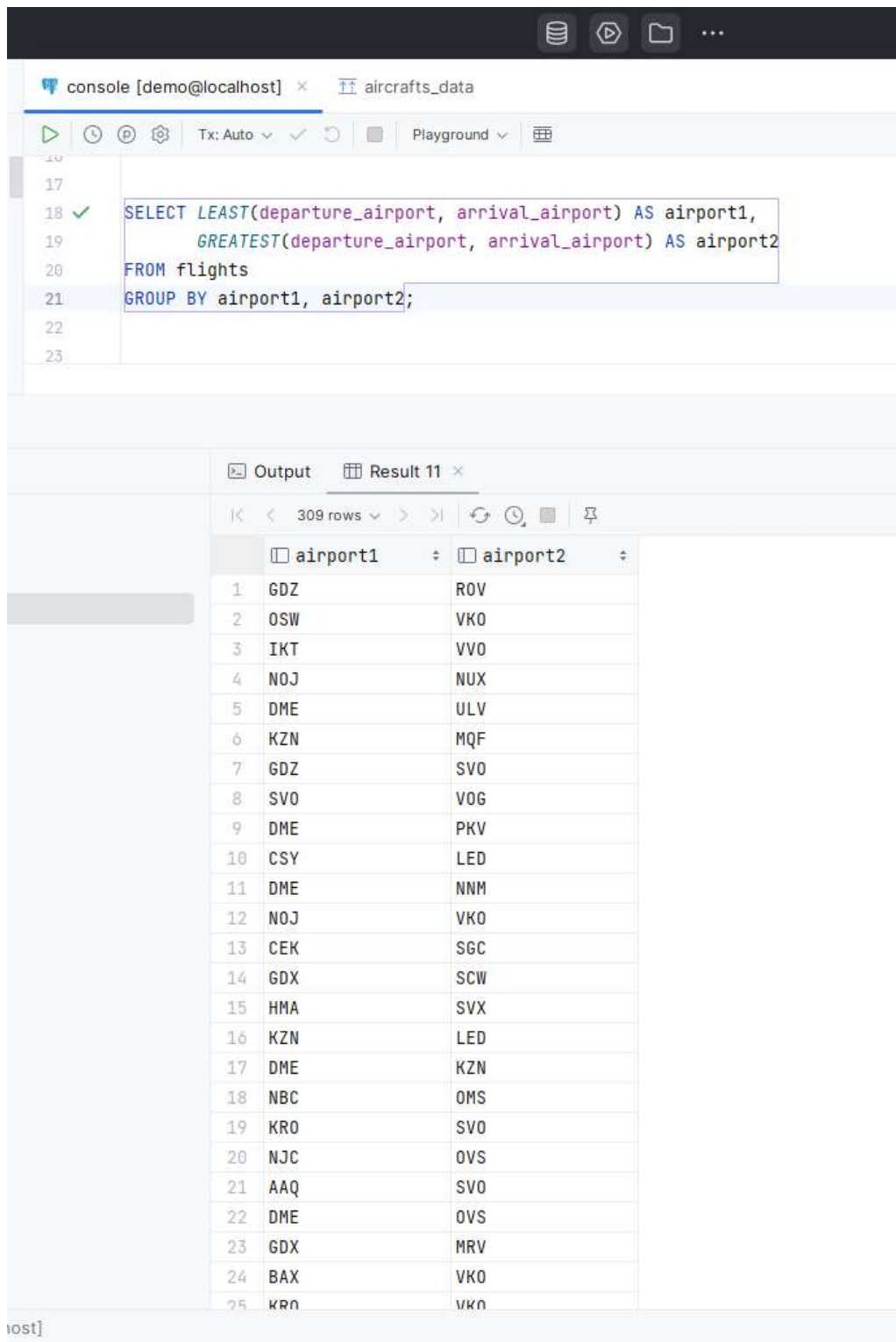
The screenshot shows a database console interface. The top part displays a SQL query being executed. The query is:

```
ORDER BY flight_id DESC;  
  
SELECT * FROM ticket_flights  
WHERE amount > 1000  
ORDER BY ticket_no ASC  
LIMIT 6;
```

The bottom part of the console shows the output of the query, which is a table with 6 rows. The table has four columns: ticket_no, flight_id, fare_conditions, and amount. The data is as follows:

| | ticket_no | flight_id | fare_conditions | amount |
|---|---------------|-----------|-----------------|----------|
| 1 | 0005432000860 | 57376 | Business | 18500.00 |
| 2 | 0005432000861 | 57376 | Economy | 6200.00 |
| 3 | 0005432000862 | 57376 | Business | 18500.00 |
| 4 | 0005432000863 | 57354 | Economy | 6200.00 |
| 5 | 0005432000864 | 57397 | Economy | 6200.00 |
| 6 | 0005432000865 | 57397 | Economy | 6200.00 |

4) Найдите все уникальные неупорядоченные пары аэропортов связанные рейсами в таблице flights. В этом пункте нельзя использовать DISTINCT. Неупорядоченная пара, значит что DME-KZN = KZN-DME. Все уникальные пары, значит что вне зависимости от количества рейсов по маршруту KZN-DME, в выводе запроса будет фигурировать только 1 экземпляр данной пары



The screenshot shows a SQL playground interface with a query editor and a results table. The query is as follows:

```
SELECT LEAST(departure_airport, arrival_airport) AS airport1,  
       GREATEST(departure_airport, arrival_airport) AS airport2  
FROM flights  
GROUP BY airport1, airport2;
```

The results table, titled "Result 11", displays 309 rows of unique airport pairs. The columns are "airport1" and "airport2". The first 25 rows are shown in the image:

| | airport1 | airport2 |
|----|----------|----------|
| 1 | GDZ | R0V |
| 2 | OSW | VK0 |
| 3 | IKT | VV0 |
| 4 | NOJ | NUX |
| 5 | DME | ULV |
| 6 | KZN | MQF |
| 7 | GDZ | SVO |
| 8 | SVO | VOG |
| 9 | DME | PKV |
| 10 | CSY | LED |
| 11 | DME | NNM |
| 12 | NOJ | VK0 |
| 13 | CEK | SGC |
| 14 | GDX | SCW |
| 15 | HMA | SVX |
| 16 | KZN | LED |
| 17 | DME | KZN |
| 18 | NBC | OMS |
| 19 | KRO | SVO |
| 20 | NJC | OVS |
| 21 | AAQ | SVO |
| 22 | DME | OVS |
| 23 | GDX | MRV |
| 24 | BAX | VK0 |
| 25 | KRN | VK0 |