

# Задание 1

```
select airport_name from airports
except
select city from airports
order by airport_name;
```

The screenshot shows a PostgreSQL client interface with a script editor and a results pane. The script editor contains four SQL queries. The first query is selected, and its results are displayed in the results pane. The results pane shows a table with two columns: 'airport\_name' and 'city'. The first row is highlighted, showing 'Байкал' in both columns.

```
-- select airport_name from airports
-- except
-- select city from airports
-- order by airport_name;
```

airport_name	city
Байкал	Байкал
Баратаевка	
Бегишево	
Беслан	
Бесовец	
Богашёво	
Витязево	
Внуково	
Гумрак	
Домодедово	
Донское	
Елизovo	
Емельяново	

## Задание 2

**select** airport\_name **from** airports

**intersect**

**select** city **from** airports

**order by** airport\_name;

The screenshot shows a PostgreSQL IDE with a script editor and a results panel. The script editor contains four SQL queries. The first query is selected, and its results are displayed in the results panel. The results panel shows a table with 13 rows of airport names, sorted alphabetically. The first row is highlighted, showing 'Абакан'.

```
select airport_name from airports
except
select city from airports
order by airport_name;

select airport_name from airports
intersect
select city from airports
order by airport_name;

select departure_airport, actual_departure, aircraft_code from flights
where
departure_airport = 'KZN'
and extract(month from actual_departure) = 8
and extract(year from actual_departure) = 2017;

select f1.aircraft_code, count(f1.aircraft_code) from flights f1
inner join(
select flight_id, departure_airport, actual_departure, aircraft_code from flights
```

Results 1 x

select airport\_name from airports in Enter a SQL expression to filter results (use Ctrl+Space)

airport_name
Абакан
Анадырь
Астрахань
Барнаул
Белгород
Белоярский
Братск
Брянск
Бугульма
Владивосток
Воркута
Воронеж
Геленджик

Value x

Абакан

Refresh Save Cancel Export data 200 66

YEKT en Writable Smart Insert 9 : 23 : 191 Sel: 0 | 0

## Задание 3

```
select f1.aircraft_code, count(f1.flight_id) from flights f1
inner join(
    select flight_id, departure_airport, scheduled_departure, aircraft_code, status from flights
    where
        departure_airport = 'KZN'
        and extract(month from scheduled_departure) = 8
        and extract(year from scheduled_departure) = 2017
        and (status = 'Arrived' or status = 'On Time')
) f2 on f2.flight_id = f1.flight_id
group by f1.aircraft_code
having count(f1.flight_id) > 50
order by count(f1.flight_id) desc, f1.aircraft_code;
```

The screenshot shows a database management tool interface. The top panel displays a SQL query that filters flights by departure airport 'KZN', month 8, year 2017, and status ('Arrived' or 'On Time'). It then counts the number of flights for each aircraft code, filtering for those with more than 50 flights, and orders them by count in descending order.

The bottom panel shows the results of the query in a table with two columns: 'aircraft\_code' and 'count'. The table is currently empty. The status bar at the bottom indicates 'YEKT en Writable', 'Smart Insert', and '29 : 53 : 1185'.

## Задание 4

```
select * from boarding_passes
inner join ticket_flights on true
inner join flights on true
inner join airports_data on true
inner join aircrafts_data on true
inner join seats on true
inner join tickets on true
inner join bookings on true;
```

The screenshot shows a database management interface with two SQL queries in the top panel and a data grid in the bottom panel.

**SQL Queries:**

```
select * from boarding_passes
inner join ticket_flights on true
inner join flights on true
inner join airports_data on true
inner join aircrafts_data on true
inner join seats on true
inner join tickets on true
inner join bookings on true;

select f1.aircraft_code, count(f1.flight_id) from flights f1
inner join(
```

**Data Grid:**

	ticket_no	flight_id	boarding_no	seat_no	ticket_no
1	0005435212351	30,625	1	2D	0005433534496
2	0005435212351	30,625	1	2D	0005433534496
3	0005435212351	30,625	1	2D	0005433534496
4	0005435212351	30,625	1	2D	0005433534496
5	0005435212351	30,625	1	2D	0005433534496
6	0005435212351	30,625	1	2D	0005433534496
7	0005435212351	30,625	1	2D	0005433534496
8	0005435212351	30,625	1	2D	0005433534496
9	0005435212351	30,625	1	2D	0005433534496
10	0005435212351	30,625	1	2D	0005433534496
11	0005435212351	30,625	1	2D	0005433534496
12	0005435212351	30,625	1	2D	0005433534496

**Dictionary (ticket flights):**

Value	Description
0005435212349	Economy
0005435212350	Economy
0005435212350	Economy
0005435212351	Business
<b>0005435212351</b>	<b>Business</b>
0005435212352	Economy

The interface includes a status bar at the bottom with the text: YEKT en Writable Smart Insert 18 : 29 : 440 Sel: 0 | 0

## Задание 5

```
select f.flight_id from flights f
inner join(
    select flight_id from ticket_flights
) tf on f.flight_id != tf.flight_id;
```

The screenshot shows a PostgreSQL IDE interface. The top pane displays a SQL script with two queries. The first query filters flights by departure airport 'KZN', month 8, and year 2017, then counts aircraft codes. The second query, which is highlighted, is the one shown in the task description. The bottom pane shows the results of the second query in a grid view. The grid has two columns: 'flight\_id' and 'count'. The first row is highlighted, showing flight\_id 1185 and count 1. The right pane shows the value 1185. The bottom status bar indicates 37 rows, 37 columns, and 1369 bytes.

```
departure_airport = 'KZN'
and extract(month from actual_departure) = 8
and extract(year from actual_departure) = 2017;

select f1.aircraft_code, count(f1.aircraft_code) from flights f1
inner join(
    select flight_id, departure_airport, actual_departure, aircraft_code from flights
    where
        departure_airport = 'KZN'
        and extract(month from actual_departure) = 8
        and extract(year from actual_departure) = 2017
) f2 on f2.flight_id = f1.flight_id
group by f1.aircraft_code;

select f.flight_id from flights f
inner join(
    select flight_id from ticket_flights
) tf on f.flight_id != tf.flight_id;
```

flight_id	count
1185	1
3979	1
4739	1
5502	1
6938	1
7784	1
9478	1
11085	1
11847	1
12012	1
13113	1
14806	1
16837	1