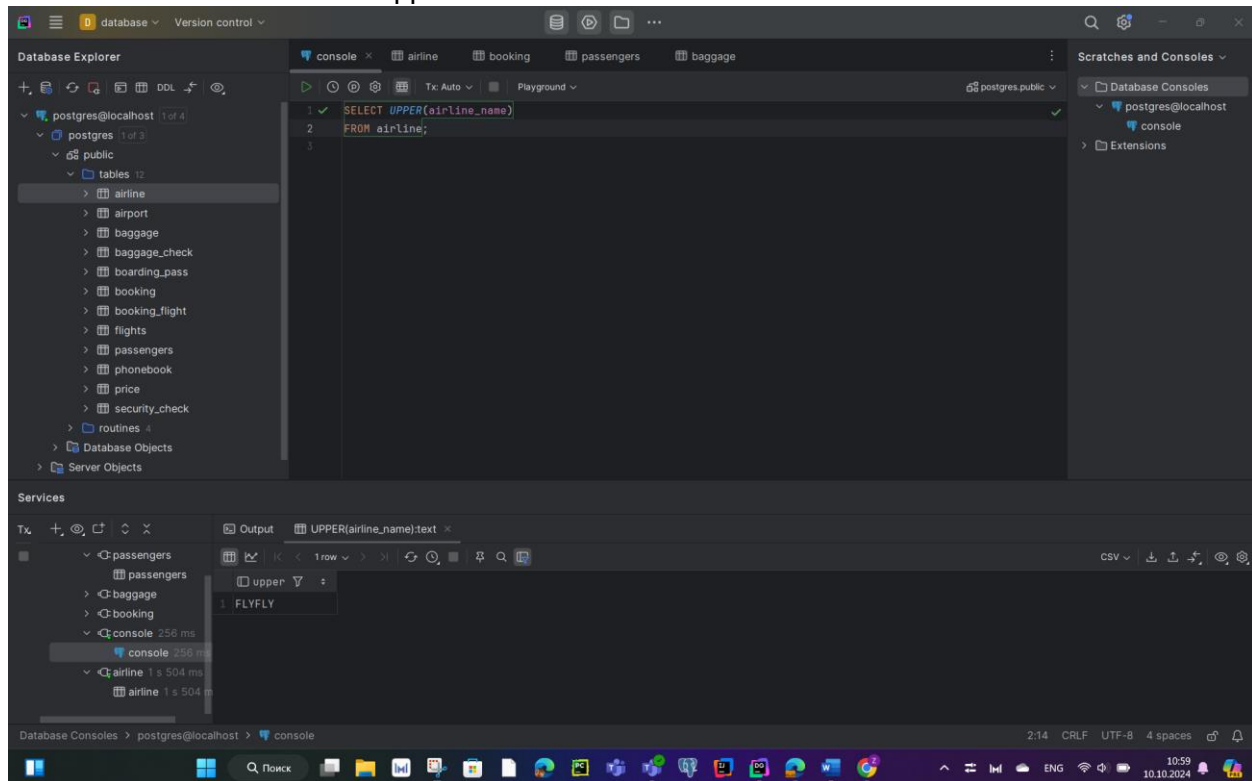
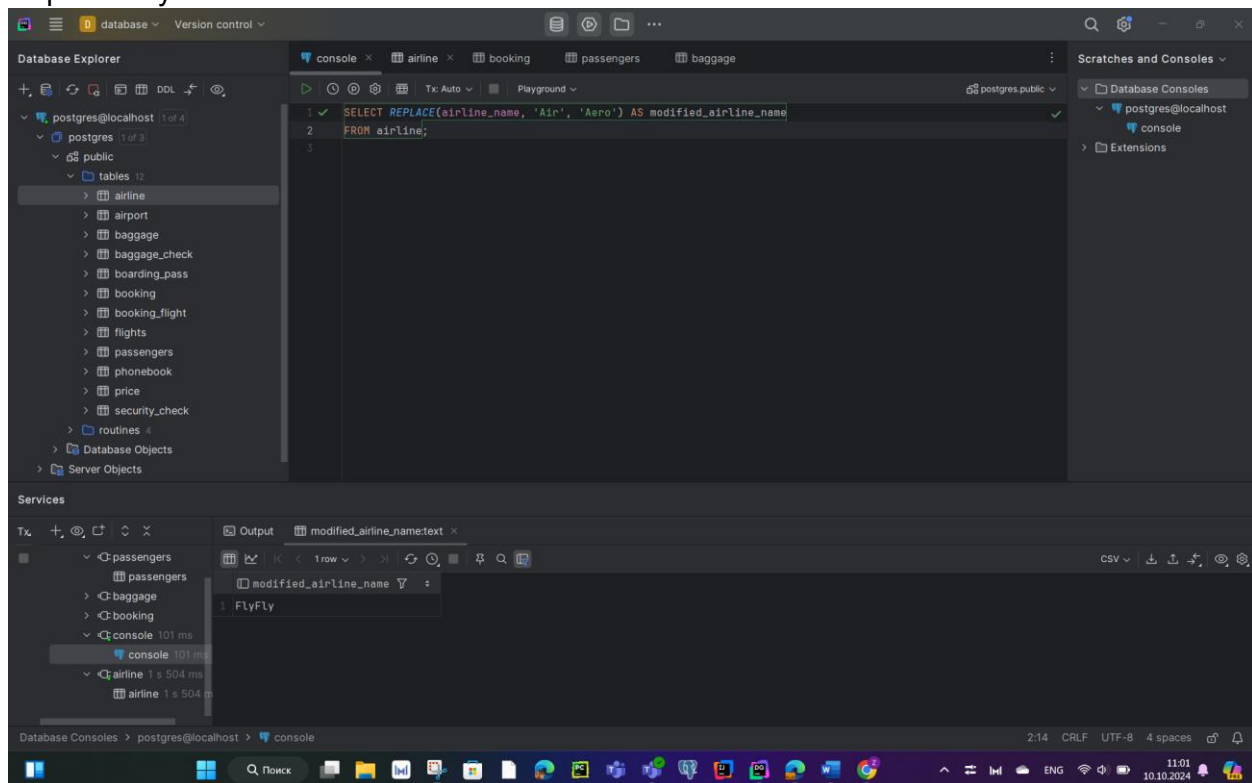


Laboratory work 4

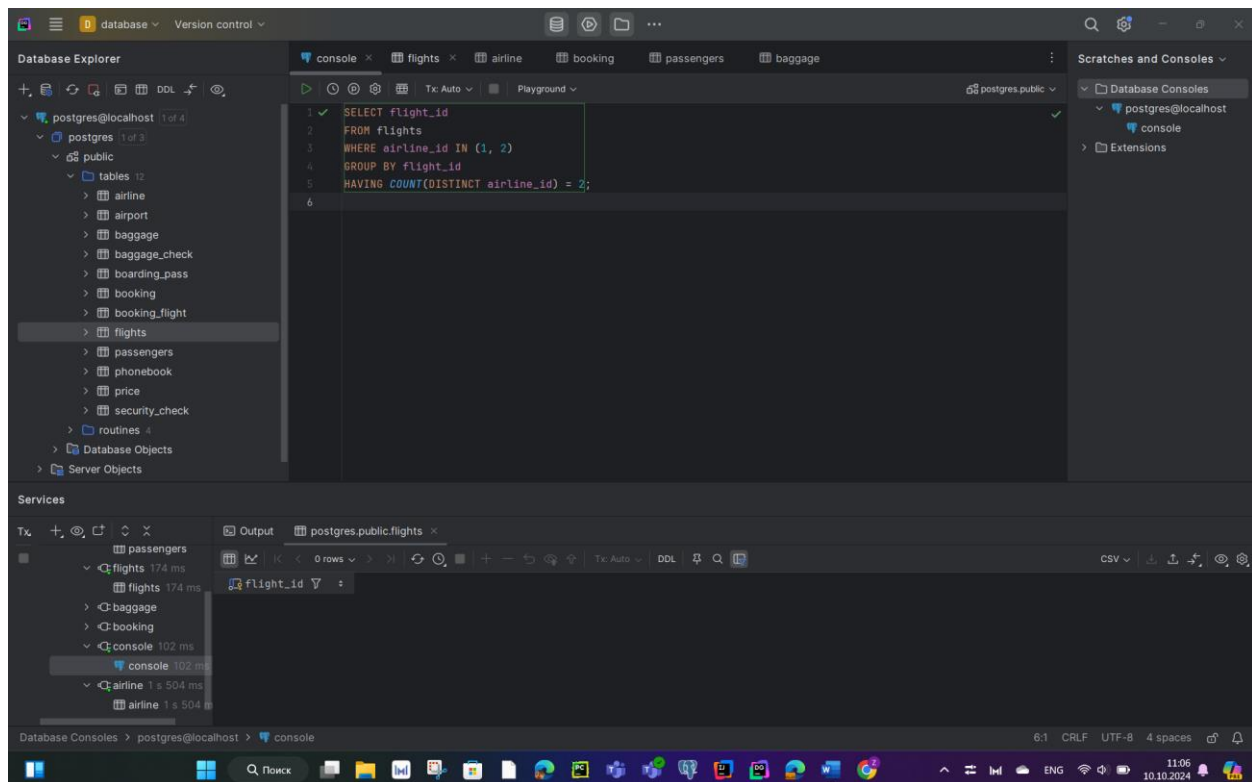
1. Retrieve all airline names in uppercase.



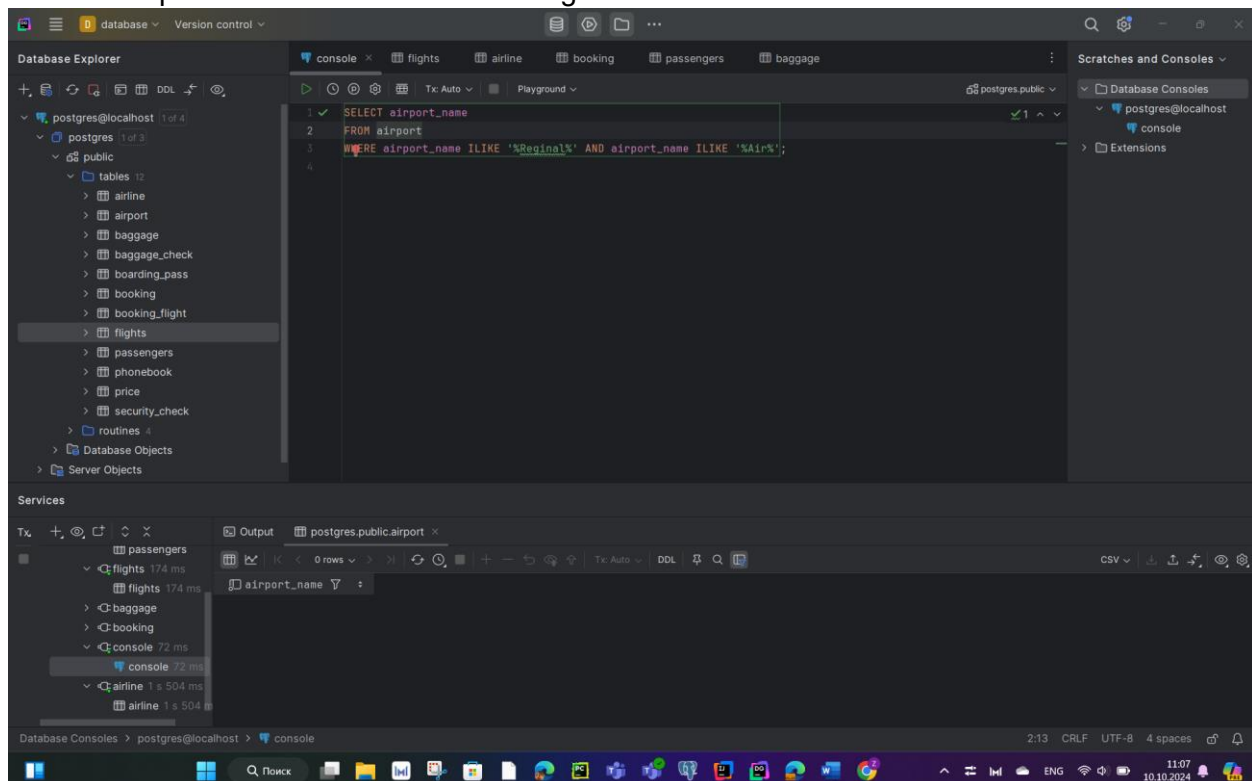
2. Replace any occurrence of the word "Air" in airline names with "Aero".



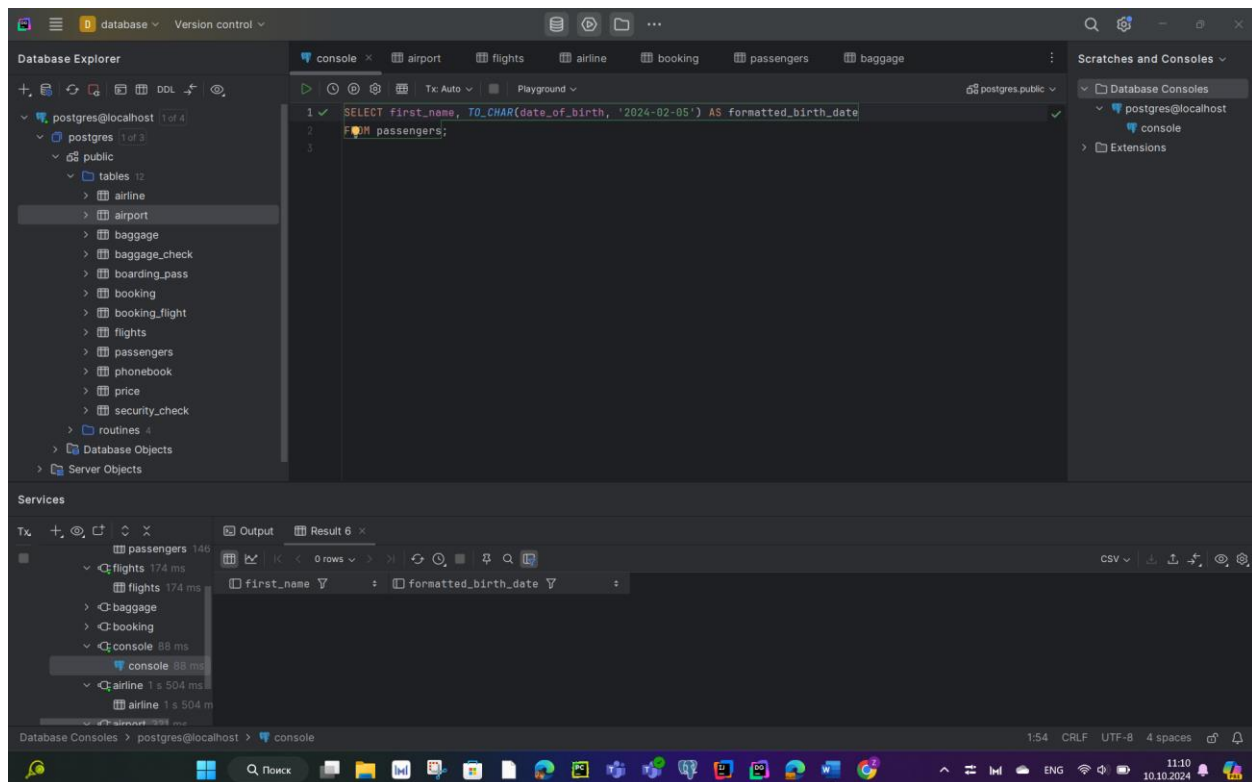
3. Find all flight numbers that coordinates with both airline 1 and airline 2.



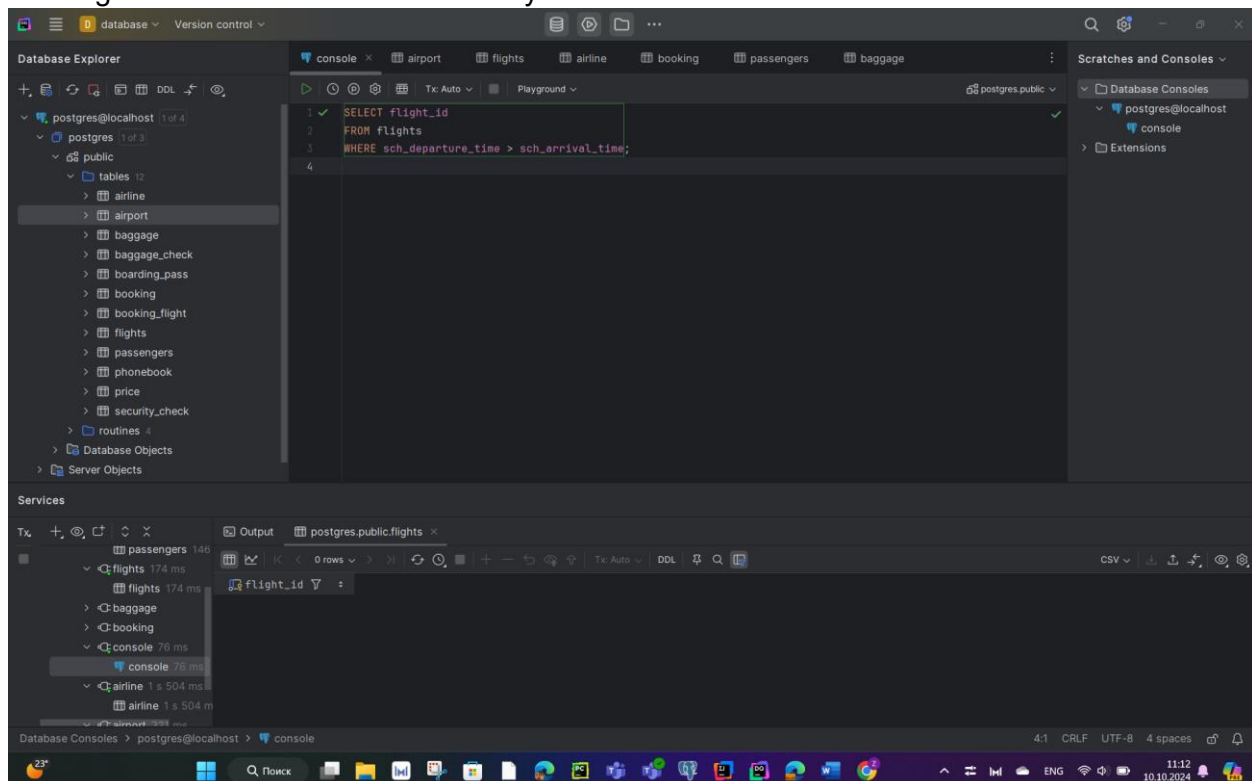
4. Retrieve airports that contain the word "Reginal" and "Air" in their names.



5. Retrieve passenger names and format their birth dates as 'Month DD, YYYY'.



6. Find flight numbers that have been delayed based on the actual arrival time.



7. Create a query that divides passengers into age groups like 'Young' and 'Adult' based on their birth date. Young passengers age between 18 and 35, Adult passengers age between 36 and 55.

The screenshot shows a database console interface with a query editor and a results pane. The query editor contains the following SQL code:

```

1 SELECT first_name,
2       CASE
3       WHEN AGE(date_of_birth) BETWEEN '18 years' AND '35 years' THEN 'Young'
4       WHEN AGE(date_of_birth) BETWEEN '36 years' AND '55 years' THEN 'Adult'
5       ELSE 'Other'
6       END AS age_group
7 FROM passengers;

```

The results pane shows the output of the query, with columns for first_name and age_group. The results are displayed in a table format.

8. Create a query that categorizes ticket prices based on their price as "Cheap," "Medium" or "Expensive."

The screenshot shows a database console interface with a query editor and a results pane. The query editor contains the following SQL code:

```

1 SELECT ticket_price,
2       CASE
3       WHEN ticket_price < 100 THEN 'Cheap'
4       WHEN ticket_price BETWEEN 100 AND 500 THEN 'Medium'
5       ELSE 'Expensive'
6       END AS price_category
7 FROM price;

```

The results pane shows the output of the query, with columns for ticket_price and price_category. The results are displayed in a table format.

9. Find number of airline names in each airline country.

The screenshot shows a database console interface with a query editor on the left and a results pane on the right. The query editor contains the following SQL code:

```
1 SELECT country, COUNT(airline_name) AS airline_count
2 FROM airline
3 GROUP BY country;
```

The results pane displays a single row of data:

country	airline_count
Poland	1

The interface also shows a list of tables in the 'Database Explorer' on the left, including 'airline', 'airport', 'baggage', 'baggage_check', 'boarding_pass', 'booking', 'booking_flight', 'flights', 'passengers', 'phonebook', 'price', 'security_check', and 'routines'. The 'Services' pane at the bottom shows the execution time for the query as 89 ms.

10. Find flights that arrived late according to their actual arrival time compared to the scheduled arrival time.

The screenshot shows a database console interface with a query editor on the left and a results pane on the right. The query editor contains the following SQL code:

```
1 SELECT flight_id, act_arrival_time, sch_arrival_time
2 FROM flights
3 WHERE act_arrival_time > sch_arrival_time;
```

The results pane displays a table with the following columns: 'flight_id', 'act_arrival_time', and 'sch_arrival_time'. The table is currently empty, showing 0 rows.

The interface also shows a list of tables in the 'Database Explorer' on the left, including 'airline', 'airport', 'baggage', 'baggage_check', 'boarding_pass', 'booking', 'booking_flight', 'flights', 'passengers', 'phonebook', 'price', 'security_check', and 'routines'. The 'Services' pane at the bottom shows the execution time for the query as 113 ms.