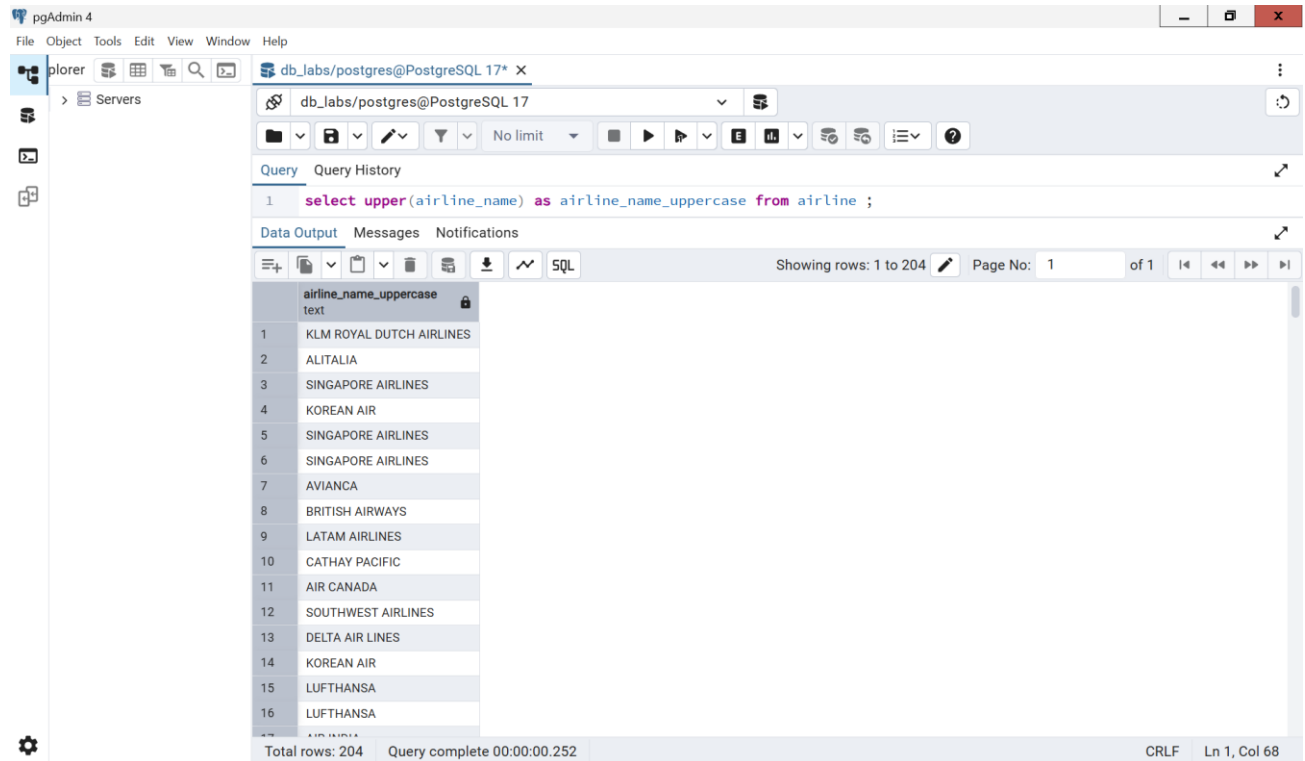


Lab4

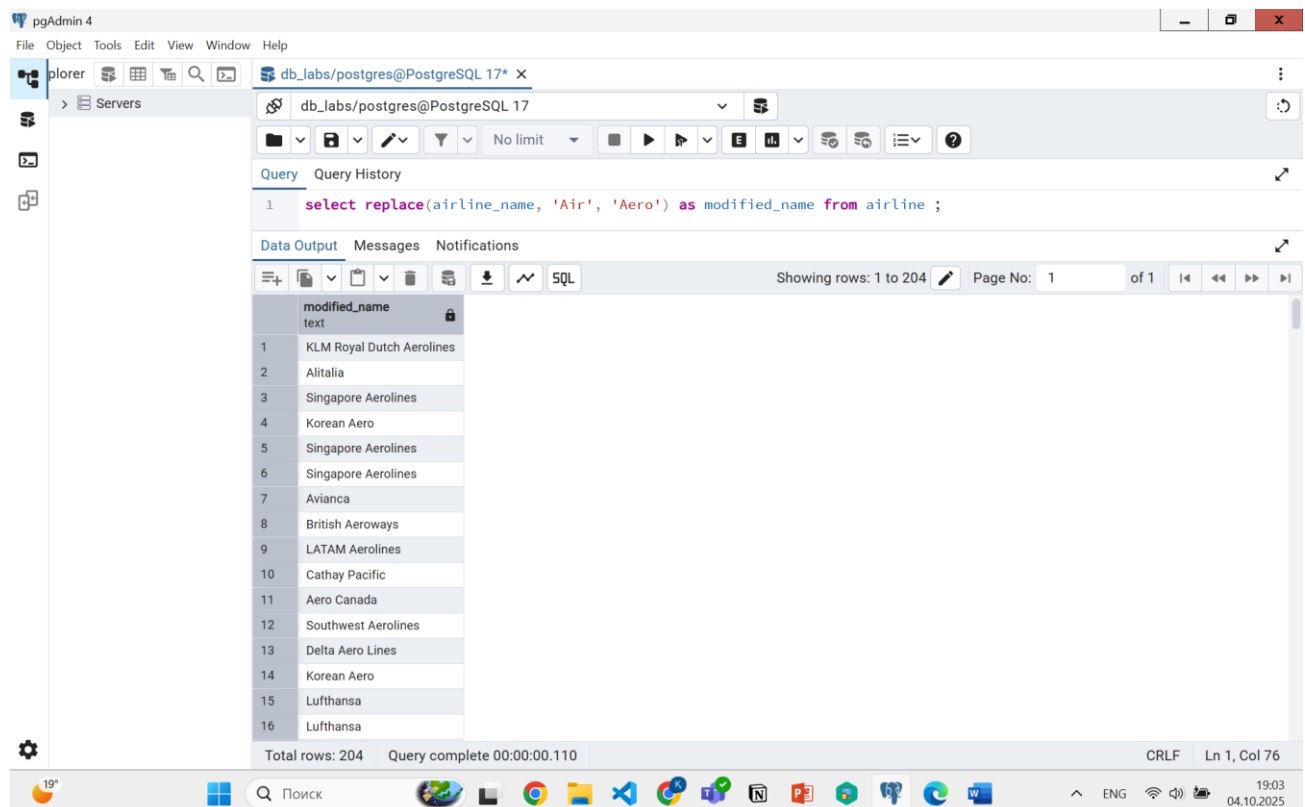
1. Retrieve all airline names in uppercase.



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the 'Servers' tree. The main pane shows a query window for 'db_labs/postgres@PostgreSQL 17'. The query is: `select upper(airline_name) as airline_name_uppercase from airline ;`. The 'Data Output' tab is active, showing a table with 204 rows. The first 16 rows are visible, showing the airline names in uppercase. The status bar at the bottom indicates 'Total rows: 204' and 'Query complete 00:00:00.252'.

airline_name_uppercase
1 KLM ROYAL DUTCH AIRLINES
2 ALITALIA
3 SINGAPORE AIRLINES
4 KOREAN AIR
5 SINGAPORE AIRLINES
6 SINGAPORE AIRLINES
7 AVIANCA
8 BRITISH AIRWAYS
9 LATAM AIRLINES
10 CATHAY PACIFIC
11 AIR CANADA
12 SOUTHWEST AIRLINES
13 DELTA AIR LINES
14 KOREAN AIR
15 LUFTHANSA
16 LUFTHANSA

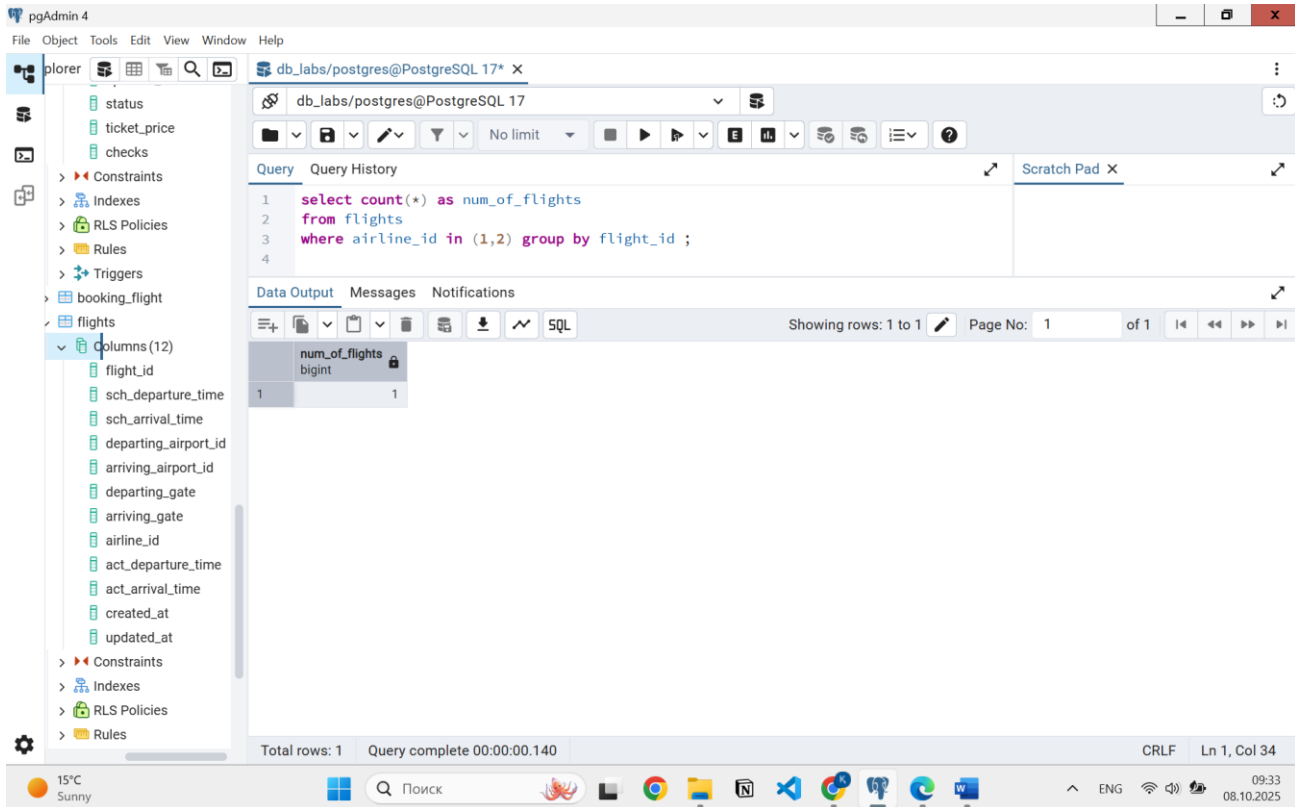
2. Replace any occurrence of the word "Air" in airline names with "Aero".



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the 'Servers' tree. The main pane shows a query window for 'db_labs/postgres@PostgreSQL 17'. The query is: `select replace(airline_name, 'Air', 'Aero') as modified_name from airline ;`. The 'Data Output' tab is active, showing a table with 204 rows. The first 16 rows are visible, showing the modified airline names. The status bar at the bottom indicates 'Total rows: 204' and 'Query complete 00:00:00.110'.

modified_name
1 KLM Royal Dutch Aerolines
2 Alitalia
3 Singapore Aerolines
4 Korean Aero
5 Singapore Aerolines
6 Singapore Aerolines
7 Avianca
8 British Aeroways
9 LATAM Aerolines
10 Cathay Pacific
11 Aero Canada
12 Southwest Aerolines
13 Delta Aero Lines
14 Korean Aero
15 Lufthansa
16 Lufthansa

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



The screenshot shows the pgAdmin 4 interface. On the left, the 'flights' table is selected under the 'columns(12)' view. The main pane displays a SQL query:

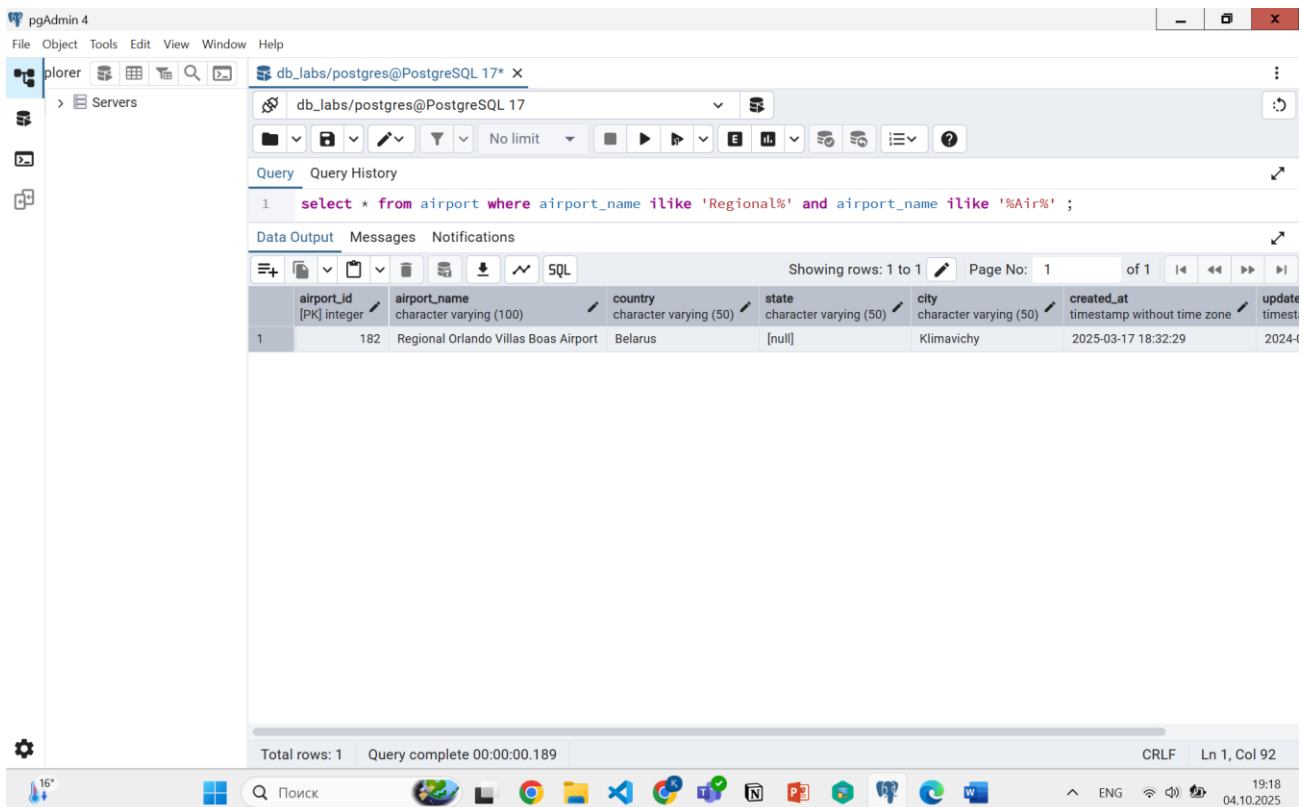
```
1 select count(*) as num_of_flights
2 from flights
3 where airline_id in (1,2) group by flight_id ;
4
```

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

num_of_flights
1

The status bar at the bottom indicates 'Total rows: 1' and 'Query complete 00:00:00.140'.

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



The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree is visible. The main pane displays a SQL query:

```
1 select * from airport where airport_name ilike 'Regional%' and airport_name ilike '%Air%' ;
```

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

airport_id [PK] integer	airport_name character varying (100)	country character varying (50)	state character varying (50)	city character varying (50)	created_at timestamp without time zone	update timestamp
1	182 Regional Orlando Villas Boas Airport	Belarus	[null]	Klimavichy	2025-03-17 18:32:29	2024-4

The status bar at the bottom indicates 'Total rows: 1' and 'Query complete 00:00:00.189'.

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

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure for 'db_labs/postgres@PostgreSQL 17'. The main pane shows a SQL query in the 'Query' tab:

```
1 select first_name, last_name, to_char(date_of_birth, 'Month DD, YYYY') as formatted_birth_date from passengers ;
```

The 'Data Output' tab shows the results of the query, displaying 16 rows of passenger data. The columns are 'first_name', 'last_name', and 'formatted_birth_date'. The data is as follows:

	first_name	last_name	formatted_birth_date
1	Jacqui	Rous	February 17, 2025
2	Averill	Machin	May 02, 2025
3	Rosalind	Critoph	February 07, 2025
4	Torey	Towe	April 23, 2025
5	Tye	Dulling	August 10, 2025
6	Fabio	Nutbeem	June 15, 2025
7	Carlie	Millions	December 27, 2024
8	Tara	Whannel	March 26, 2025
9	Obadiah	Acome	April 16, 2025
10	Binky	Figg	May 05, 2025
11	Karalynn	Boler	May 19, 2025
12	Abraham	Blunkett	January 09, 2025
13	Heath	Manach	December 18, 2024
14	Cornell	Yacobsohn	December 31, 2024
15	Alonso	Iacapucci	October 28, 2024
16	Zulema	Toomer	April 16, 2025

The status bar at the bottom indicates 'Total rows: 200' and 'Query complete 00:00:00.175'.

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

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure for 'db_labs/postgres@PostgreSQL 17'. The main pane shows a SQL query in the 'Query' tab:

```
1 select count(*) as delayed_flights
2 from flights
3 where act_arrival_time > sch_arrival_time;
4
```

The 'Data Output' tab shows the results of the query, displaying 1 row of data. The column is 'delayed_flights' and the value is 64.

delayed_flights
64

The status bar at the bottom indicates 'Total rows: 1' and 'Query complete 00:00:00.134'.

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 the pgAdmin 4 interface with a SQL query executed against the 'db_labs/postgres@PostgreSQL 17' database. The query categorizes passengers into age groups based on their birth date.

```

1 select first_name , last_name , date_part('year', age(current_date,date_of_birth)) as age , case
2 when date_part('year', age(current_date, date_of_birth)) between 18 and 35 then 'young'
3 when date_part('year', age(current_date, date_of_birth)) between 36 and 55 then 'adult'
4 else 'other'
5 end as age_group
6 from passengers ;

```

The results are displayed in a table with the following columns: first_name, last_name, age, and age_group. The table shows 12 rows of data.

	first_name	last_name	age	age_group
1	Jacqui	Rous	0	other
2	Averill	Machin	0	other
3	Rosalind	Critoph	0	other
4	Torey	Towe	0	other
5	Tye	Dulling	0	other
6	Fabio	Nutbeem	0	other
7	Carlie	Millions	0	other
8	Tara	Whannel	0	other
9	Obadiah	Acome	0	other
10	Binky	Figg	0	other
11	Karalynn	Boler	0	other
12	Abrahan	Blunkett	0	other

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

The screenshot shows the pgAdmin 4 interface with a SQL query executed against the 'db_labs/postgres@PostgreSQL 17' database. The query categorizes ticket prices into 'cheap', 'medium', or 'expensive' based on their value.

```

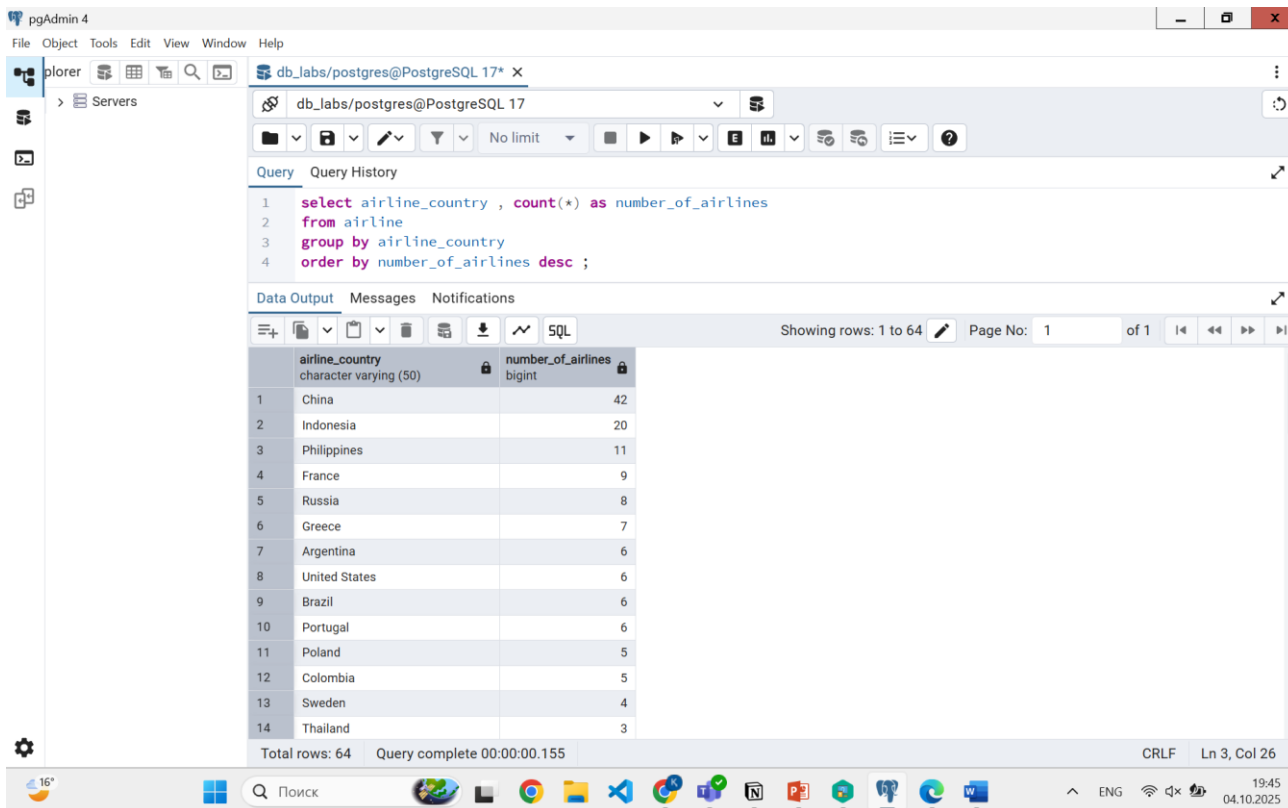
1 select booking_id, ticket_price , case
2 when ticket_price < 100 then 'cheap'
3 when ticket_price between 100 and 300 then 'medium'
4 else 'expensive'
5 end as price_category
6 from booking ;

```

The results are displayed in a table with the following columns: booking_id, ticket_price, and price_category. The table shows 12 rows of data.

	booking_id	ticket_price	price_category
1	2	68.99	cheap
2	22	57.49	cheap
3	41	172.49	medium
4	44	91.99	cheap
5	55	228.85	medium
6	56	57.49	cheap
7	63	68.99	cheap
8	70	51.74	cheap
9	95	149.49	medium
10	73	413.99	expensive
11	106	229.99	medium
12	111	91.99	cheap

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



The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

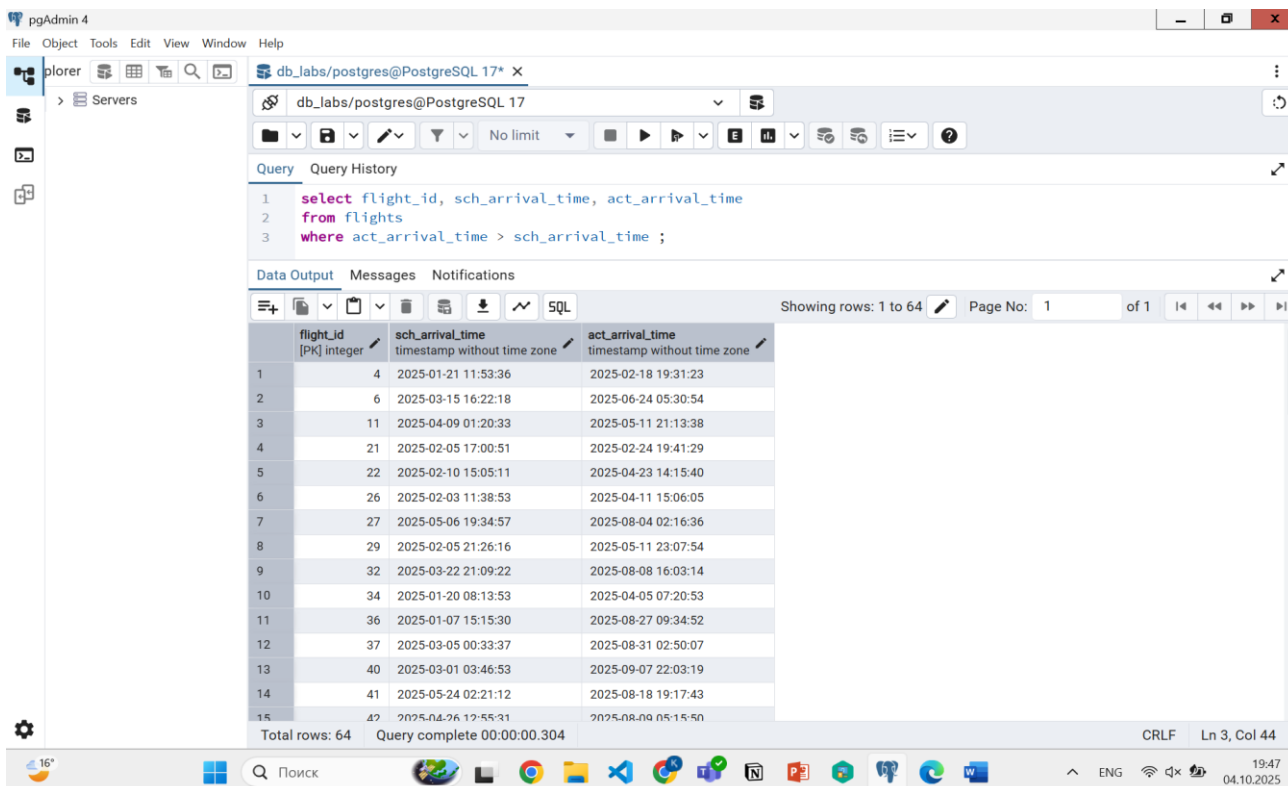
```
1 select airline_country , count(*) as number_of_airlines
2 from airline
3 group by airline_country
4 order by number_of_airlines desc ;
```

The Data Output tab shows the results of the query. The table has two columns: `airline_country` (character varying (50)) and `number_of_airlines` (bigint). The results are as follows:

airline_country	number_of_airlines
China	42
Indonesia	20
Philippines	11
France	9
Russia	8
Greece	7
Argentina	6
United States	6
Brazil	6
Portugal	6
Poland	5
Colombia	5
Sweden	4
Thailand	3

Total rows: 64. Query complete 00:00:00.155. CRLF Ln 3, Col 26.

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



The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

```
1 select flight_id, sch_arrival_time, act_arrival_time
2 from flights
3 where act_arrival_time > sch_arrival_time ;
```

The Data Output tab shows the results of the query. The table has three columns: `flight_id` (integer), `sch_arrival_time` (timestamp without time zone), and `act_arrival_time` (timestamp without time zone). The results are as follows:

flight_id	sch_arrival_time	act_arrival_time
4	2025-01-21 11:53:36	2025-02-18 19:31:23
6	2025-03-15 16:22:18	2025-06-24 05:30:54
11	2025-04-09 01:20:33	2025-05-11 21:13:38
21	2025-02-05 17:00:51	2025-02-24 19:41:29
22	2025-02-10 15:05:11	2025-04-23 14:15:40
26	2025-02-03 11:38:53	2025-04-11 15:06:05
27	2025-05-06 19:34:57	2025-08-04 02:16:36
29	2025-02-05 21:26:16	2025-05-11 23:07:54
32	2025-03-22 21:09:22	2025-08-08 16:03:14
34	2025-01-20 08:13:53	2025-04-05 07:20:53
36	2025-01-07 15:15:30	2025-08-27 09:34:52
37	2025-03-05 00:33:37	2025-08-31 02:50:07
40	2025-03-01 03:46:53	2025-09-07 22:03:19
41	2025-05-24 02:21:12	2025-08-18 19:17:43
42	2025-04-26 12:55:31	2025-08-09 05:15:50

Total rows: 64. Query complete 00:00:00.304. CRLF Ln 3, Col 44.