

1. Retrieve all airline names in uppercase

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

```
1 select upper(airline_name) from airline
```

The Data Output pane shows the results of the query:

upper
IPC
PDN
KLE
KHS
YLO
NGL
O
QIG
NOX
SOZ
IVA

The status bar at the bottom indicates the query was successfully run with a total runtime of 151 msec and 0 rows affected.

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

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

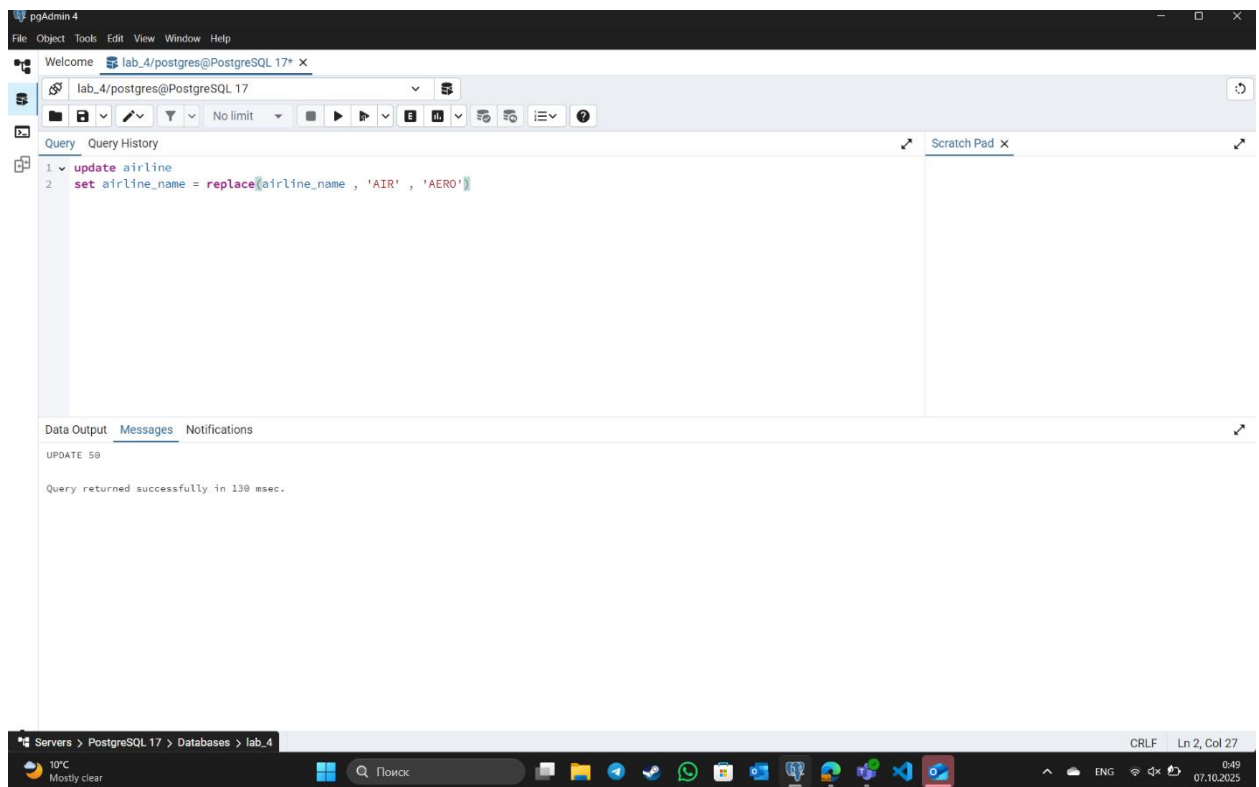
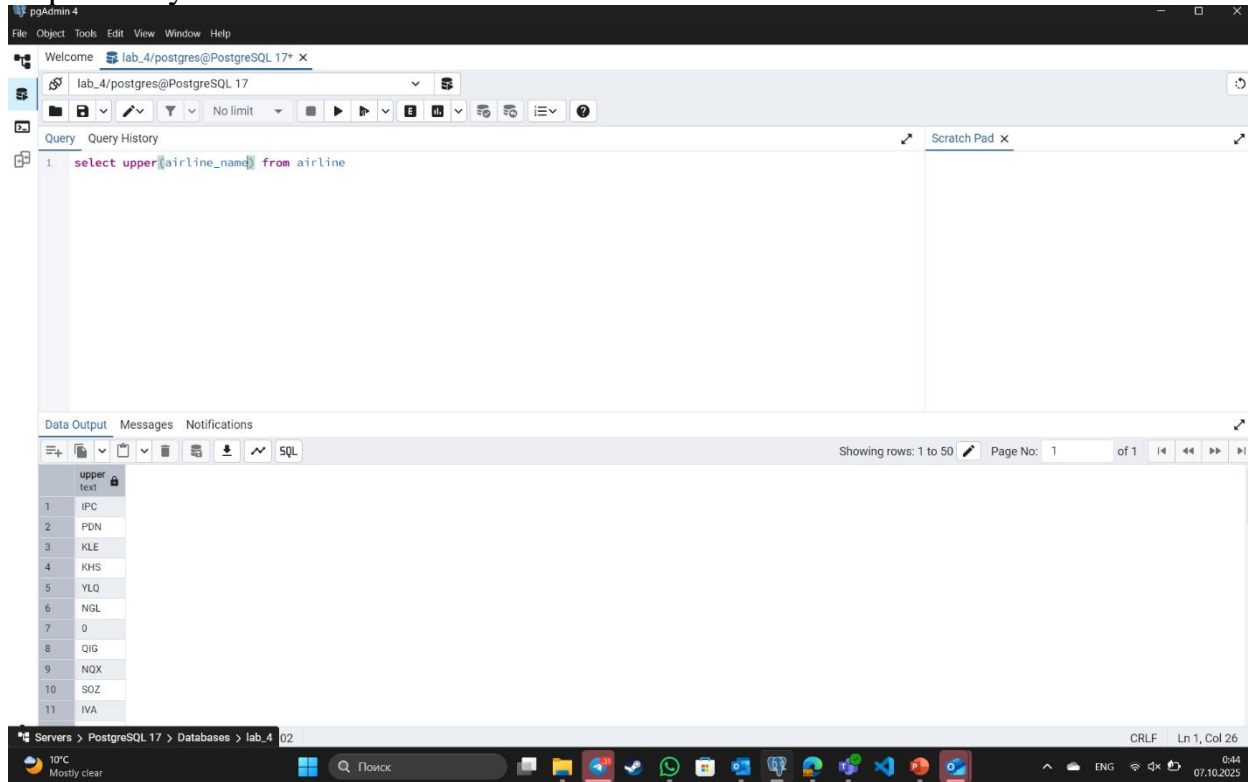
```
1 SELECT flight_no
2 FROM flights
3 WHERE airline_id IN (1, 2)
4 GROUP BY flight_no
5 HAVING COUNT(DISTINCT airline_id) = 2;
```

The Data Output pane shows the results of the query:

flight_no
character varying (50)

The status bar at the bottom indicates the query was successfully run with a total runtime of 151 msec and 0 rows affected.

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



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

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

```
1 select * from airport
2 where airport_name ~*'Reginal' and airport_name ~*'Air' |
```

The Data Output pane shows the following columns:

airportId	airport_name	country	state	city	created_at	update_at
[PK] integer	character varying (50)	character varying (50)	character varying (50)	character varying (50)	date	date

A status message at the bottom right indicates: "Successfully run. Total query runtime: 126 msec. 0 rows affected."

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

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

```
1 SELECT
2 first_name, last_name,
3 TO_CHAR(date_of_birth, 'FMMonth DD, YYYY') AS formatted_birth_date
4 FROM passengers;
5 |
```

The Data Output pane shows the following columns:

first_name	last_name	formatted_birth_date	
character varying (50)	character varying (50)	text	
1	Hilde	Irnis	January 03, 2000
2	Anvy	Sparsholt	June 09, 1974
3	Reinald	Pococke	June 07, 1982
4	Con	Borrel	October 17, 1986
5	Wayne	Bangs	April 22, 1996
6	Tildy	Shackleford	April 15, 2004
7	Byrle	Oram	July 07, 1985
8	Howard	Igo	March 29, 1984
9	Alphonso	Phillipou	August 30, 1975
10	Fairfax	Needham	July 08, 1981
11	Vanda	Prime	May 17, 1977

A status message at the bottom right indicates: "Successfully run. Total query runtime: 156 msec. 200 rows affected."

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

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

```
1 SELECT flight_no
2 from flights
3 where actual_arrival > scheduled_arrival
```

The query has been executed successfully, and the results are displayed in the Data Output pane. The results show 11 rows of flight numbers. A status message at the bottom right indicates: "Successfully run. Total query runtime: 132 msec. 509 rows affected."

flight_no
US-CT
US-NM
RU-KR
US-AZ
IN-OR
CA-NL
BR-PE
TH-32
US-MS
GB-ENG
UZ-SU

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.

pgAdmin 4

Welcome lab_4/postgres@PostgreSQL 17

lab_4/postgres@PostgreSQL 17

Query Query History

```

1 SELECT
2   first_name,last_name,
3   DATE_PART('year', AGE(CURRENT_DATE, date_of_birth)) AS age,
4   CASE
5     WHEN DATE_PART('year', AGE(CURRENT_DATE, date_of_birth)) BETWEEN 18 AND 35 THEN 'Young'
6     WHEN DATE_PART('year', AGE(CURRENT_DATE, date_of_birth)) BETWEEN 36 AND 55 THEN 'Adult'
7     ELSE 'Other'
8   END AS age_group
9 FROM passengers;
10

```

Data Output Messages Notifications

Showing rows: 1 to 200 Page No: 1 of 1

	first_name character varying (50)	last_name character varying (50)	age double precision	age_group text
1	Hilde	Irnis	25	Young
2	Anvy	Sparsholt	51	Adult
3	Reinald	Pococke	43	Adult
4	Con	Borrel	38	Adult
5	Wayne	Bangs	29	Young
6	Tildy	Shackleford	21	Young
7	Byrle	Oram	40	Adult
8	Howard	Igo	41	Adult
9	Alphonso	Philippou	50	Adult
10	Fairfax	Needham	44	Adult
11	Vanda	Prime	48	Adult

Servers > PostgreSQL 17 > Databases > lab_4 127

CRLF Ln 2, Col 24

AWOL -0.06%

18°C Sunny

12:18 07.10.2025

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

pgAdmin 4

Welcome lab_4/postgres@PostgreSQL 17

lab_4/postgres@PostgreSQL 17

Query Query History

```

1 SELECT booking_id, passenger_id, price,
2   case
3     when price <2500 then 'cheap'
4     when price between 2500 and 7500 then 'medium'
5     when price > 7500 then 'expensive'
6   end as price_category
7 from booking

```

Data Output Messages Notifications

Showing rows: 1 to 500 Page No: 1 of 1

	booking_id [PK] integer	passenger_id integer	price numeric (7,2)	price_category text
1	1	156	7462.13	medium
2	2	120	4216.60	medium
3	3	42	5782.37	medium
4	4	147	102.96	cheap
5	5	79	6711.04	medium
6	6	120	7813.55	expensive
7	7	14	1346.71	cheap
8	8	197	5711.67	medium
9	9	160	7015.82	medium
10	10	92	1668.38	cheap
11	11	35	3342.95	medium

Servers > PostgreSQL 17 > Databases > lab_4 163

CRLF Ln 7, Col 13

18°C Sunny

12:27 07.10.2025

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,  
2 count(airline_name) as number_of_airline  
3 from airline  
4 group by airline_country
```

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

airline_country	number_of_airline
Yemen	1
Argentina	1
Indonesia	2
Sudan	1
Venezuela	1
Hungary	1
Slovenia	1
Greece	1
China	10
Russia	4
Georgia	1

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_no, actual_arrival , scheduled_arrival  
2 from flights  
3 where actual_arrival > scheduled_arrival
```

The Data Output tab shows the results of the query. The table has three columns: 'flight_no' (character varying (50)), 'actual_arrival' (date), and 'scheduled_arrival' (date). The results are as follows:

flight_no	actual_arrival	scheduled_arrival
US-CT	2023-11-07	2023-09-08
US-NM	2024-01-23	2023-09-17
RU-KR	2023-04-07	2023-03-18
US-AZ	2023-08-01	2023-04-08
IN-OR	2023-12-03	2023-09-19
CA-NL	2023-11-17	2023-06-04
BR-PE	2023-11-09	2023-06-02
TH-32	2023-07-21	2023-03-30
US-MS	2024-03-06	2024-01-22
GB-ENG	2024-01-04	2023-10-11
UZ-SU	2023-12-23	2023-05-24

A green notification bar at the bottom right indicates: "Successfully run. Total query runtime: 99 msec. 509 rows affected."