

Query Query History

```

1 SELECT UPPER(airline_name) AS airline_name_upper
2 FROM airline;
3

```

Select All

Data Output Messages Notifications

airline_name_upper	text
5	AEROFLOT
6	BRITISH AIRWAYS
7	LUFTHANSA
8	AIR FRANCE
9	EMIRATES
10	QANTAS
11	AMERICAN AIRLINES
12	SINGAPORE AIRLINES
13	CHINA AIRLINES

Total rows: 13 Query complete 00:00:00.097

1)SELECT UPPER(airline\_name) AS airline\_name\_upper

FROM airline;

2) SELECT REPLACE(airline\_name, 'Air', 'Aero') AS modified\_airline\_name

FROM airline;

ed\* X public.airport/airp... X public.airline/airpo... X airport\_db/postgres@Postg

airport\_db/postgres@PostgreSQL 17

No limit

Query Query History

```

1 SELECT REPLACE(airline_name, 'Air', 'Aero') AS airline_name
2 FROM airline;
3

```

Data Output Messages Notifications

airline_name	text
5	Aeroflot
6	British Aeroways
7	Lufthansa
8	Aero France
9	Emirates
10	Qantas
11	American Aerolines
12	Singapore Aerolines
13	China Aerolines

3)

SELECT flight\_id

FROM flight

WHERE airline\_id IN (1, 2)

GROUP BY flight\_id

HAVING COUNT(DISTINCT airline\_id) = 2;

The screenshot shows a SQL query editor with two tabs: "Query" and "Query History". The "Query" tab is active, displaying a SQL query with line numbers 1 through 6. Below the query, there are tabs for "Data Output", "Messages", and "Notifications". The "Messages" tab is active, showing the text "INSERT 0 5" and "Query returned successfully in 85 msec."

```
1 SELECT flight_id
2 FROM flight
3 WHERE airline_id IN (1, 2)
4 GROUP BY flight_id
5 HAVING COUNT(DISTINCT airline_id) = 2
6
```

INSERT 0 5

Query returned successfully in 85 msec.

4)

The screenshot shows a SQL query editor with two tabs: "Query" and "Query History". The "Query" tab is active, displaying a SQL query with line numbers 1 through 5. Below the query, there are tabs for "Data Output", "Messages", and "Notifications". The "Data Output" tab is active, showing a table with one column, "airport\_name", and one row with the value "airport\_name character varying (50)".

```
1 SELECT airport_name
2 FROM airport
3 WHERE airport_name LIKE '%Reginal%'
4 AND airport_name LIKE '%Air%';
5
```

airport\_name  
character varying (50)

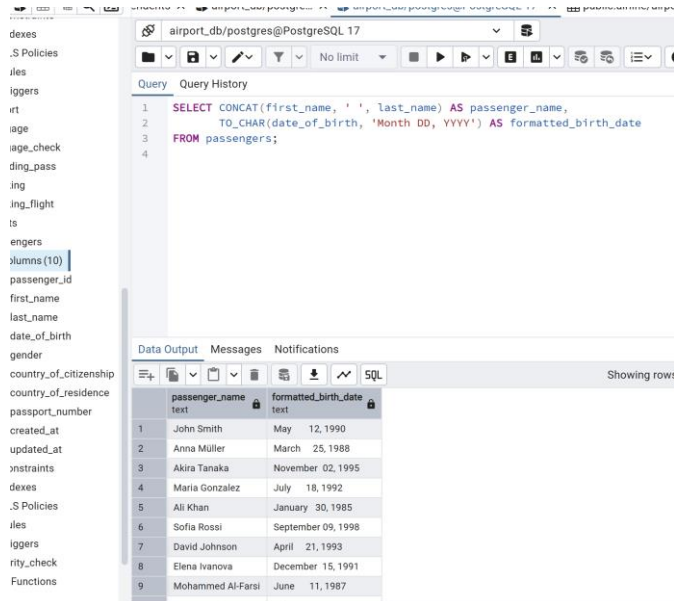
SELECT airport\_name

FROM airport

WHERE airport\_name LIKE '%Reginal%' AND airport\_name LIKE '%Air%';

5)

```
SELECT CONCAT(first_name, ' ', last_name) AS passenger_name,  
        TO_CHAR(date_of_birth, 'Month DD, YYYY') AS formatted_birth_date  
FROM passengers;
```



The screenshot shows a PostgreSQL query editor with a query window and a results window. The query window contains the following SQL code:

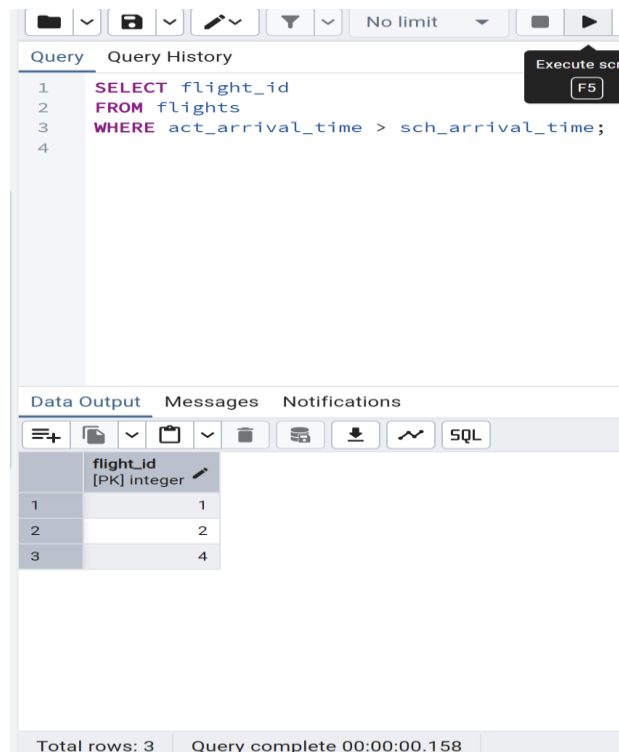
```
1 SELECT CONCAT(first_name, ' ', last_name) AS passenger_name,  
2     TO_CHAR(date_of_birth, 'Month DD, YYYY') AS formatted_birth_date  
3 FROM passengers;  
4
```

The results window displays the output of the query, showing two columns: **passenger\_name** and **formatted\_birth\_date**. The results are as follows:

	passenger_name	formatted_birth_date
1	John Smith	May 12, 1990
2	Anna Müller	March 25, 1988
3	Akira Tanaka	November 02, 1995
4	Maria Gonzalez	July 18, 1992
5	Ali Khan	January 30, 1985
6	Sofia Rossi	September 09, 1998
7	David Johnson	April 21, 1993
8	Elena Ivanova	December 15, 1991
9	Mohammed Al-Farsi	June 11, 1987

6)

```
SELECT flight_id  
FROM flights  
WHERE act_arrival_time > sch_arrival_time;
```



The screenshot shows a PostgreSQL query editor with a query window and a results window. The query window contains the following SQL code:

```
1 SELECT flight_id  
2 FROM flights  
3 WHERE act_arrival_time > sch_arrival_time;  
4
```

The results window displays the output of the query, showing one column: **flight\_id**. The results are as follows:

flight_id
1
2
3

Total rows: 3    Query complete 00:00:00.158

7)

SELECT

first\_name || ' ' || last\_name AS passenger\_name,

CASE

WHEN EXTRACT(YEAR FROM AGE(date\_of\_birth)) BETWEEN 18 AND 35 THEN 'Young'

WHEN EXTRACT(YEAR FROM AGE(date\_of\_birth)) BETWEEN 36 AND 55 THEN 'Adult'

END AS age\_group

FROM passengers;

(|| – in PostgreSQL, this is the string concatenation operator. That is, first\_name || ' ' || last\_name takes the first name (first\_name), adds a space (' '), and then the last name (last\_name). For example: "Ivan" || ' ' || "Petrov" → "Ivan Petrov". AS passenger\_name – this is a column alias. It sets the name of the resulting column.)

The screenshot shows a PostgreSQL query editor interface. The query is as follows:

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

The results are displayed in a table with two columns: passenger\_name and age\_group. The table contains 9 rows of data.

	passenger_name	age_group
1	John Smith	Young
2	Anna Müller	Adult
3	Akira Tanaka	Young
4	Maria Gonzalez	Young
5	Ali Khan	Adult
6	Sofia Rossi	Young
7	David Johnson	Young
8	Elena Ivanova	Young
9	Mohammed Al-Farsi	Adult

Total rows: 10 Query complete 00:00:00.169

8)

SELECT

ticket\_price,

CASE

WHEN ticket\_price < 10000 THEN 'Cheap'

WHEN ticket\_price BETWEEN 10000 AND 30000 THEN 'Medium'

ELSE 'Expensive'

END AS price\_category

FROM booking ;

The screenshot shows a SQL query editor with a query window and a data output window. The query window contains the following SQL code:

```
1 SELECT
2     ticket_price,
3     CASE
4         WHEN ticket_price < 10000 THEN 'Cheap'
5         WHEN ticket_price BETWEEN 10000 AND 30000 THEN 'Medium'
6         ELSE 'Expensive'
7     END AS price_category
8 FROM booking ;
9
```

The data output window shows the results of the query. It has a toolbar with icons for adding, deleting, and refreshing data, as well as a 'SQL' button. The results are displayed in a table with two columns: 'ticket\_price' (numeric (7,2)) and 'price\_category' (text). The table contains three rows of data:

	ticket_price numeric (7,2)	price_category text
1	15000.50	Medium
2	14500.00	Medium
3	12000.75	Medium

9)

SELECT airline\_country, COUNT(\*) AS airline\_count

FROM airline

GROUP BY airline\_country;

“ COUNT(\*) is an aggregate function that counts the number of rows.\* means "count all rows."If you just write COUNT(\*), it counts all rows in each group (or in the entire table if there is no GROUP BY).Example: if there are 5 airlines from Kazakhstan in a table, then COUNT(\*) for the "Kazakhstan" group will return 5. ”

Query	Query History
1	SELECT airline_country, COUNT(*) AS airline_count
2	FROM airline
3	GROUP BY airline_country;
4	

Data Output	Messages	Notifications
<div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>📦</div> <div>⬇️</div> <div>📈</div> <div>SQL</div> </div>		
airline_country	airline_count	
character varying (50)	bigint	
1	Singapore	1
2	France	2
3	United States	1
4	Turkey	1
5	China	1
6	Australia	1
7	United Arab Emirates	1
8	United Kingdom	1
9	Germany	1

Total rows: 12	Query complete 00:00:00.199
----------------	-----------------------------

10)

```
SELECT flight_id, sch_arrival_time, act_arrival_time
FROM flights
WHERE act_arrival_time > sch_arrival_time;
```

Query	Query History
1	SELECT flight_id, sch_arrival_time, act_arrival_time
2	FROM flights
3	WHERE act_arrival_time > sch_arrival_time;
4	

Data Output	Messages	Notifications
<div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>📦</div> <div>⬇️</div> <div>📈</div> <div>SQL</div> <div>Show</div> </div>		
flight_id	sch_arrival_time	act_arrival_time
[PK] integer	timestamp without time zone	timestamp without time zone
1	1 2025-10-05 11:00:00	2025-10-05 11:20:00
2	2 2025-10-06 18:00:00	2025-10-06 18:05:00
3	4 2025-10-09 00:30:00	2025-10-09 00:45:00

**Filtering: we display only those flights where the actual arrival time is later than the scheduled time (that is, the flight arrived behind schedule).**

