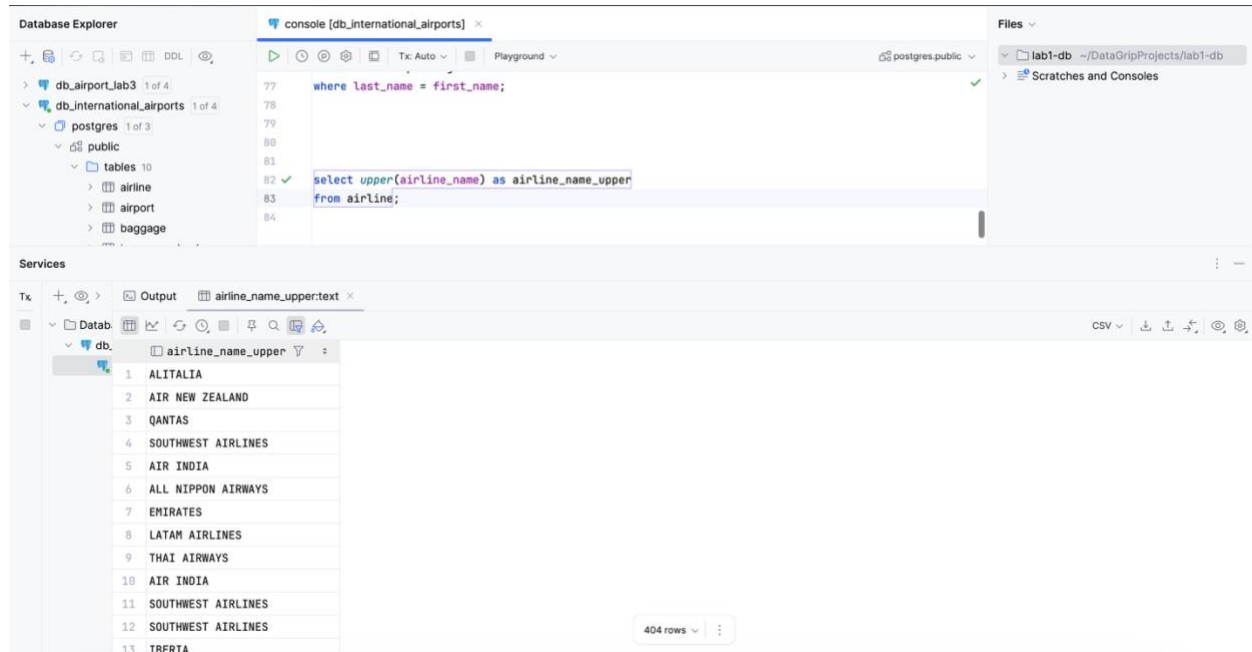


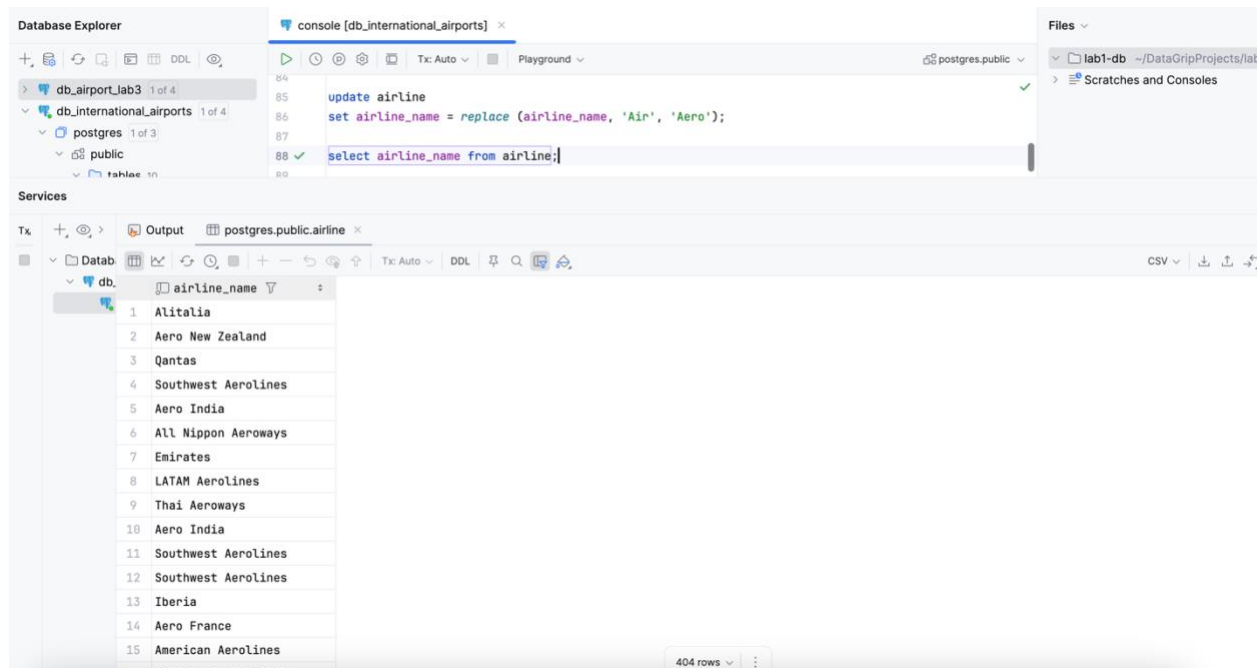
Lab 4

1. Retrieve all airline names in uppercase.



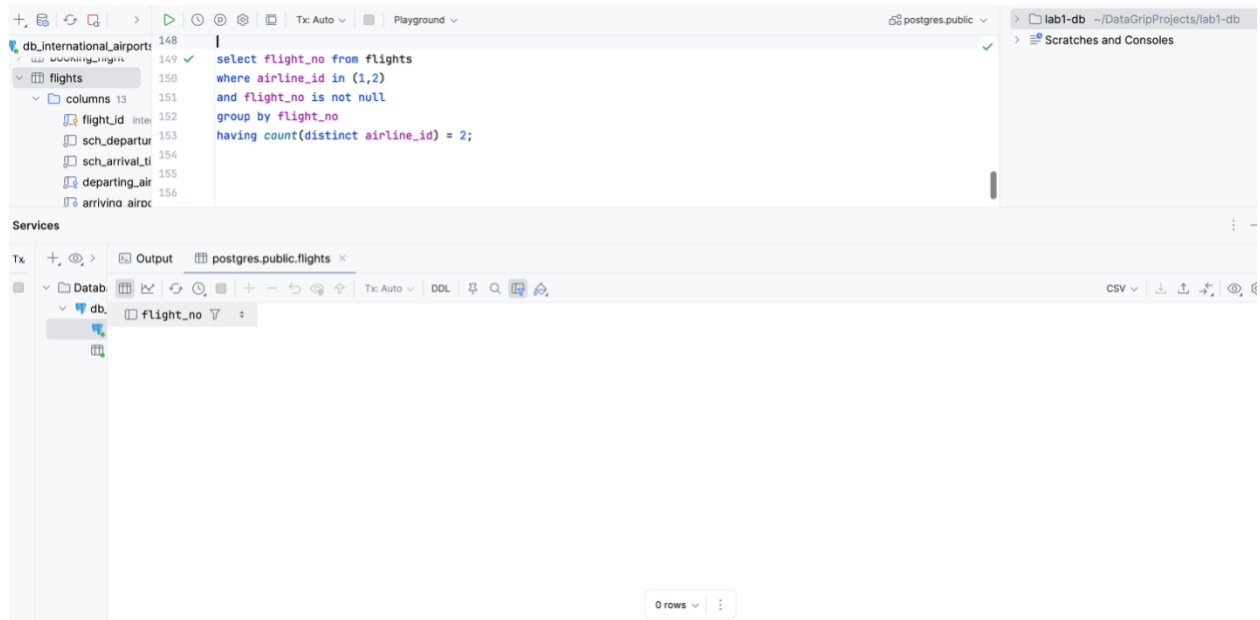
The screenshot shows a database console interface. On the left, the 'Database Explorer' pane shows a tree view of the database structure, including 'db_international_airports' and 'public' schema. The main console area displays a SQL query: `select upper(airline_name) as airline_name_upper from airline;`. The 'Output' pane on the right shows the results of the query, listing 13 airline names in uppercase: ALITALIA, AIR NEW ZEALAND, QANTAS, SOUTHWEST AIRLINES, AIR INDIA, ALL NIPPON AIRWAYS, EMIRATES, LATAM AIRLINES, THAI AIRWAYS, AIR INDIA, SOUTHWEST AIRLINES, SOUTHWEST AIRLINES, and IBERIA. The console also shows a 'where last_name = first_name;' query above the main one.

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

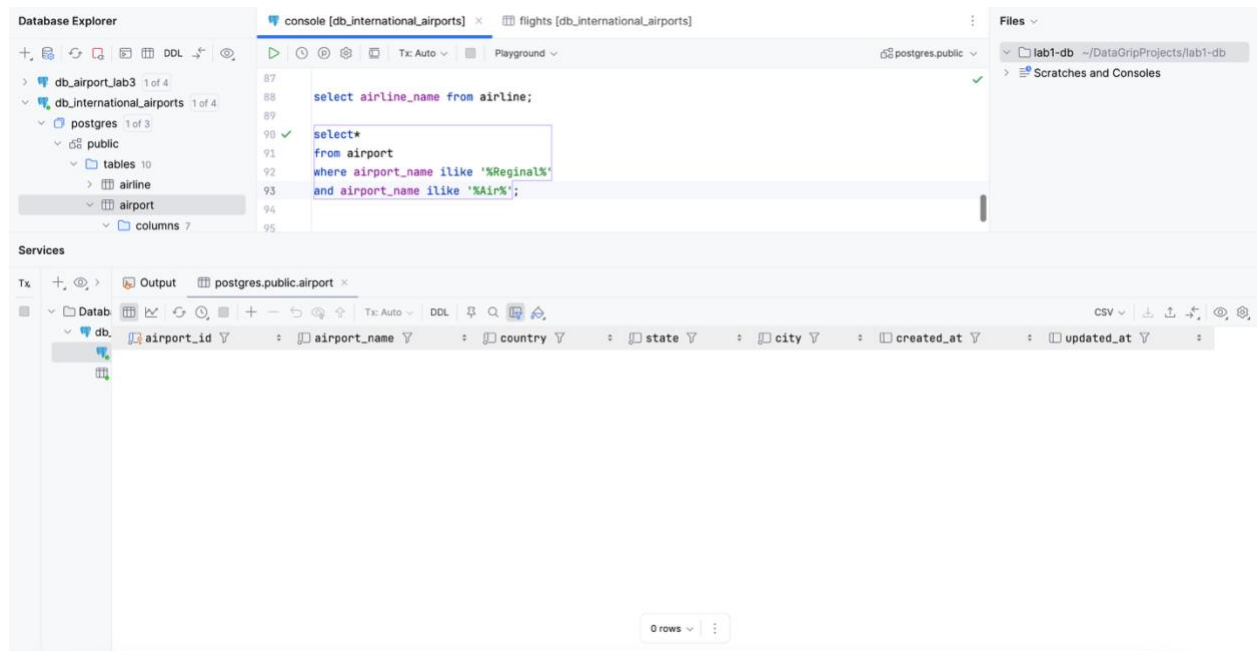


The screenshot shows a database console interface. On the left, the 'Database Explorer' pane shows a tree view of the database structure, including 'db_international_airports' and 'public' schema. The main console area displays a SQL query: `update airline set airline_name = replace (airline_name, 'Air', 'Aero'); select airline_name from airline;`. The 'Output' pane on the right shows the results of the query, listing 15 airline names with 'Air' replaced by 'Aero': Alitalia, Aero New Zealand, Qantas, Southwest Aerolines, Aero India, All Nippon Aeroways, Emirates, LATAM Aerolines, Thai Aeroways, Aero India, Southwest Aerolines, Southwest Aerolines, Iberia, Aero France, and American Aerolines. The console also shows a 'where last_name = first_name;' query above the main one.

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'..o

Konyratbayeva Uldana

The screenshot shows a database console interface with the following components:

- Database Explorer:** Lists databases including `db_internationalAirports`, `flights`, `passengers`, and `columns`.
- Console:** Contains a SQL query:

```
select first_name,
       last_name,
       to_char(date_of_birth, 'Month DD, YYYY') as birth_date
from passengers;
```
- Output:** Displays the results of the query in a table format with 17 rows and 4 columns: `first_name`, `last_name`, `birth_date`, and an implicit row number column.

	first_name	last_name	birth_date
1	Lisetta	Ranyell	March 29, 2025
2	Marco	Grindall	April 08, 2025
3	Ricardo	Handke	October 08, 2024
4	Anetta	Garnsworth	July 24, 2025
5	Eolande	Blitzer	July 12, 2025
6	Anthea	Shotter	April 10, 2025
7	Cornelia	Ruck	January 17, 2025
8	Ryley	Dublin	November 18, 2024
9	Mark	Dubbin	March 01, 2025
10	Adrea	Campos	August 22, 2025
11	Walton	Bayle	March 07, 2025
12	Denni	Liston	December 21, 2024
13	Lucius	Lakeland	July 23, 2025
14	Amelia	Jaggs	June 29, 2025
15	Zollie	Willmot	May 17, 2025
16	Phedra	De Bruin	June 12, 2025
17	Dama11a	Hawine	April 05, 2025

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

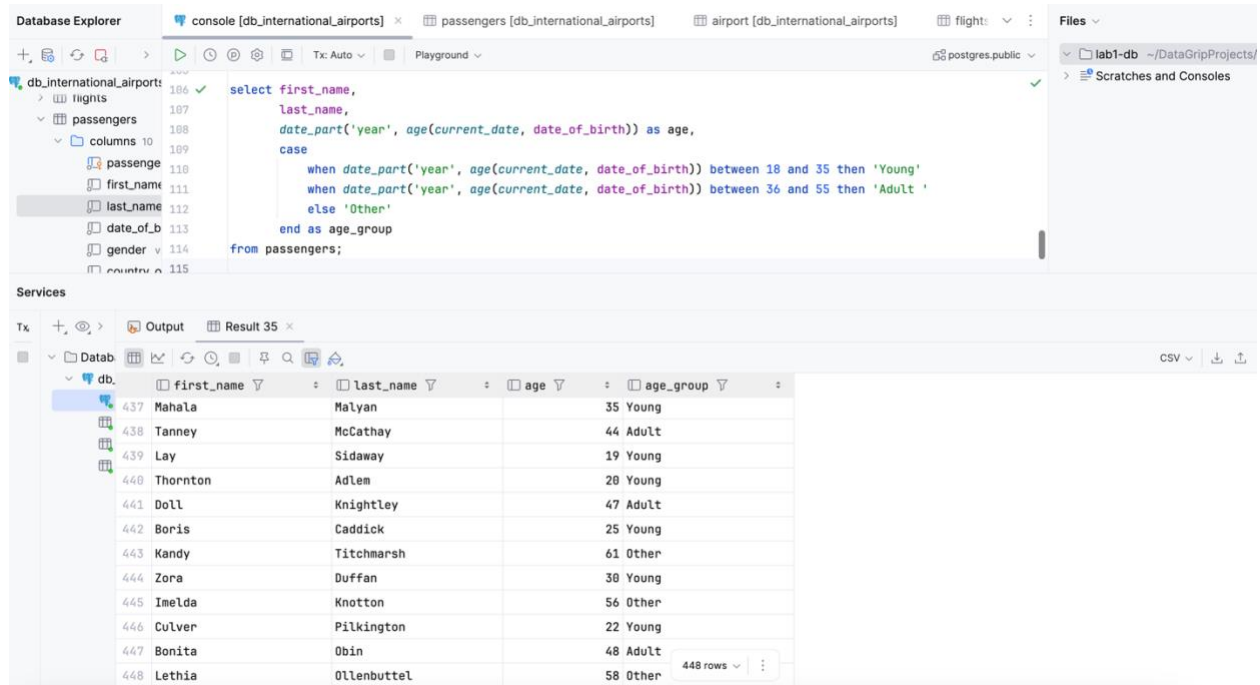
The screenshot shows a database console interface with the following components:

- Database Explorer:** Lists databases including `db_internationalAirports`, `booking`, `passengers`, and `airp`.
- Console:** Contains a SQL query:

```
select flight_id from flights
where sch_arrival_time < act_arrival_time;
```
- Output:** Displays the results of the query in a table format with 19 rows and 1 column: `flight_id`.

flight_id
4
7
9
13
15
20
21
24
28
29
30
31
33
35
36
37
40
41

- 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 IDE with a SQL query in the console. The query selects passenger details and categorizes them into age groups based on their birth date. The result set shows 448 rows of data.

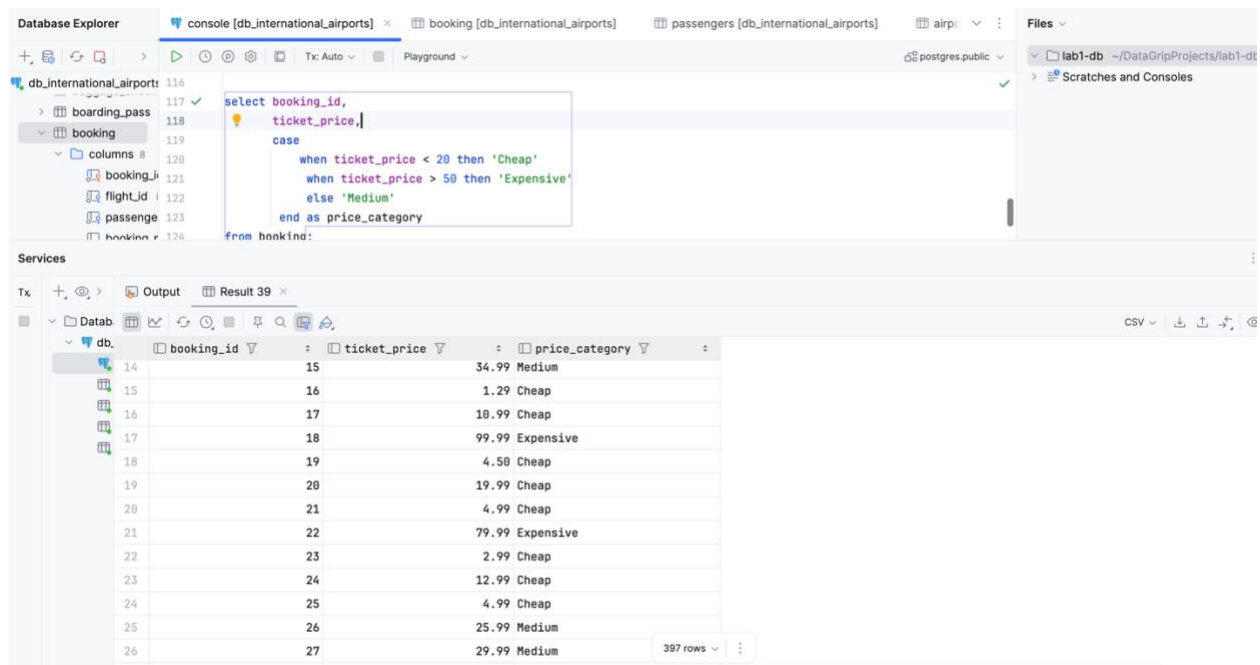
```

select first_name,
       last_name,
       date_part('year', age(current_date, date_of_birth)) as age,
       case
         when date_part('year', age(current_date, date_of_birth)) between 18 and 35 then 'Young'
         when date_part('year', age(current_date, date_of_birth)) between 36 and 55 then 'Adult'
         else 'Other'
       end as age_group
from passengers;

```

first_name	last_name	age	age_group
437	Mahala	35	Young
438	Tanney	44	Adult
439	Lay	19	Young
440	Thornton	20	Young
441	Doll	47	Adult
442	Boris	25	Young
443	Kandy	61	Other
444	Zora	30	Young
445	Imelda	56	Other
446	Culver	22	Young
447	Bonita	48	Adult
448	Lethia	58	Other

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



The screenshot shows a database IDE with a SQL query in the console. The query selects booking details and categorizes them into price categories based on their ticket price. The result set shows 397 rows of data.

```

select booking_id,
       ticket_price,
       case
         when ticket_price < 20 then 'Cheap'
         when ticket_price > 50 then 'Expensive'
         else 'Medium'
       end as price_category
from booking;

```

booking_id	ticket_price	price_category
14	15	34.99 Medium
15	16	1.29 Cheap
16	17	18.99 Cheap
17	18	99.99 Expensive
18	19	4.50 Cheap
19	20	19.99 Cheap
20	21	4.99 Cheap
21	22	79.99 Expensive
22	23	2.99 Cheap
23	24	12.99 Cheap
24	25	4.99 Cheap
25	26	25.99 Medium
26	27	29.99 Medium

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

The screenshot shows the DataGrip interface with a SQL query executed in the console. The query is as follows:

```
select airline_country,  
       count(airline_name) as number_of_airlines  
from airline  
group by airline_country  
order by number_of_airlines desc;
```

The results are displayed in a table with two columns: `airline_country` and `number_of_airlines`. The table contains 13 rows of data, sorted by the number of airlines in descending order.

airline_country	number_of_airlines
China	91
Indonesia	46
Russia	21
Brazil	20
Philippines	18
Portugal	16
Poland	15
United States	11
France	9
Japan	8
Colombia	8
Ukraine	8
Sweden	6

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

The screenshot shows the DataGrip interface with a SQL query executed in the console. The query is as follows:

```
select flight_id,  
       sch_arrival_time,  
       act_arrival_time from flights  
where sch_arrival_time < act_arrival_time;
```

The results are displayed in a table with three columns: `flight_id`, `sch_arrival_time`, and `act_arrival_time`. The table contains 16 rows of data, showing flights that arrived later than scheduled.

flight_id	sch_arrival_time	act_arrival_time
4	2002-03-30 22:46:52.000000	2014-10-20 21:51:39.000000
7	2003-04-12 18:50:18.000000	2014-04-12 16:38:56.000000
9	2012-03-23 17:59:28.000000	2016-04-16 17:26:11.000000
13	2005-12-21 12:07:13.000000	2023-12-04 17:32:12.000000
15	2005-09-24 08:30:03.000000	2014-12-17 21:10:33.000000
20	2008-06-22 04:20:38.000000	2010-11-08 18:47:08.000000
21	2008-03-19 04:11:05.000000	2017-07-22 18:55:25.000000
24	2001-01-30 18:04:22.000000	2008-09-13 11:19:20.000000
28	2013-10-15 15:30:30.000000	2025-05-19 02:51:42.000000
29	2012-12-11 07:09:58.000000	2023-04-21 04:48:30.000000
30	2005-11-26 17:49:53.000000	2006-03-22 09:58:15.000000
31	2008-03-06 23:02:02.000000	2021-04-07 08:36:03.000000
33	2013-02-01 05:38:43.000000	2014-01-23 04:45:51.000000
35	2007-11-27 00:54:54.000000	2009-07-16 19:31:58.000000
36	2012-08-02 23:04:43.000000	2020-09-14 21:57:00.000000
37	2017-05-22 01:38:55.000000	2019-01-07 03:21:00.000000