

1. Write a query that displays all flights of a specific airline.

The screenshot shows the DBeaver 25.2.0 interface. The left sidebar displays the database schema, including the 'airline' table with columns: airline_id (int4), airline_code (varchar(30)), airline_name (varchar(50)), airline_country (varchar(50)), created_at (timestamp), updated_at (timestamp), and info (varchar(50)). The main editor shows a SQL query:

```

--Write a query that displays all flights of a specific airline.
select flights.flight_id, airline.airline_name, airline.airline_country from flights
join airline on flights.airline_id=airline.airline_id
order by airline.airline_name
  
```

The query results are displayed in a table with columns: flight_id, airline_name, and airline_country. The results show 15 rows of data, including flights from Airline 10 (Russia), Airline 100 (Russia), Airline 101 (United States), Airline 102 (Germany), Airline 103 (France), Airline 104 (United Kingdom), Airline 105 (Canada), Airline 106 (Japan), Airline 107 (Australia), Airline 108 (China), Airline 109 (Brazil), Airline 11 (United States), Airline 110 (Russia), Airline 111 (United States), and Airline 112 (Germany).

2. Compose a query to obtain a list of all flights with the names of departure airports.

The screenshot shows the DBeaver 25.2.0 interface. The left sidebar displays the database schema, including the 'airport' table with columns: airport_id (int4), airport_name (varchar(50)), country (varchar(50)), state (varchar(50)), city (varchar(50)), created_at (timestamp), updated_at (timestamp), and info (varchar(50)). The main editor shows a SQL query:

```

SELECT
    flights.flight_id,
    flights.sch_departure_time,
    airport.airport_name AS departure_airport,
    airport.city AS departure_city,
    airport.country AS departure_country,
    airline.airline_name
FROM
    flights
INNER JOIN airport ON flights.departing_airport_id = airport.airport_id
INNER JOIN airline ON flights.airline_id = airline.airline_id
ORDER BY
    flights.sch_departure_time
  
```

The query results are displayed in a table with columns: flight_id, sch_departure_time, departure_airport, departure_city, departure_country, and airline_name. The results show 20 rows of data, including flights from Sydney Airport, City 10, City 11, City 12, City 13, City 14, City 15, City 16, City 17, City 18, City 19, and City 20.

3. Create a query that finds all airlines that have no flights scheduled for the next month.

The screenshot shows the DBeaver interface with a SQL query executed against a PostgreSQL database. The query is as follows:

```
SELECT
    airline.airline_id,
    airline.airline_code,
    airline.airline_name,
    airline.airline_country,
    flights.flight_id,
    flights.sch_departure_time
FROM
    airline
LEFT JOIN
    flights ON airline.airline_id = flights.airline_id
    AND flights.sch_departure_time >= CURRENT_DATE
    AND flights.sch_departure_time < CURRENT_DATE + INTERVAL '1 month'
WHERE
    flights.airline_id IS NULL
ORDER BY
    airline.airline_name
```

The result set, titled "airline(+)" and containing 11 rows, displays the following data:

	airline_id	airline_code	airline_name	airline_country	flight_id	sch_departure_time
1	10	A10	Airline 10	Russia	[NULL]	[NULL]
2	100	A10	Airline 100	Russia	[NULL]	[NULL]
3	101	A10	Airline 101	United States	[NULL]	[NULL]
4	102	A10	Airline 102	Germany	[NULL]	[NULL]
5	103	A10	Airline 103	France	[NULL]	[NULL]
6	104	A10	Airline 104	United Kingdom	[NULL]	[NULL]
7	105	A10	Airline 105	Canada	[NULL]	[NULL]
8	106	A10	Airline 106	Japan	[NULL]	[NULL]
9	107	A10	Airline 107	Australia	[NULL]	[NULL]
10	108	A10	Airline 108	China	[NULL]	[NULL]
11	109	A10	Airline 109	Brazil	[NULL]	[NULL]
12	11	A11	Airline 11	United States	[NULL]	[NULL]

4. Create a query to display a list of passengers on a specific flight.

The screenshot shows the DBeaver interface with a SQL query executed against a PostgreSQL database. The query is as follows:

```
SELECT
    p.passenger_id,
    p.first_name,
    p.last_name,
    p.date_of_birth,
    p.gender,
    p.country_of_citizenship,
    p.passport_number,
    booking.booking_id,
    booking.status,
    bp.seat,
    bp.boarding_time
FROM
    passengers p
INNER JOIN booking b ON p.passenger_id = booking.passenger_id
INNER JOIN flights f ON booking.flight_id = f.flight_id
LEFT JOIN boarding_pass bp ON booking.booking_id = bp.booking_id
WHERE
    f.flight_id = 11 AND booking.status != 'CANCELLED'
ORDER BY
    p.last_name, p.first_name
```

The result set, titled "passengers(+)" and containing 1 row, displays the following data:

	passenger_id	first_name	last_name	date_of_birth	gender	country_of_citizenship	passport_number
1	11	Alexander	Surname10	1980-01-01	Male	Russia	RU0000010

5. Write a query that calculates the average, total, maximum and minimum price of tickets for each flight.

The screenshot shows the DBeaver SQL editor with a query that calculates the average, total, maximum, and minimum price of tickets for each flight. The query is as follows:

```
SELECT
    f.flight_id,
    f.sch_departure_time,
    dep_airport.airport_name AS departure_airport,
    arr_airport.airport_name AS arrival_airport,
    COUNT(b.booking_id) AS total_bookings,
    ROUND(AVG(b.price), 2) AS average_price,
    SUM(b.price) AS total_revenue,
    MAX(b.price) AS maximum_price,
    MIN(b.price) AS minimum_price
FROM
    flights f
    INNER JOIN booking b ON f.flight_id = b.flight_id
    INNER JOIN airport dep_airport ON f.departing_airport_id = dep_airport.airport_id
    INNER JOIN airport arr_airport ON f.arriving_airport_id = arr_airport.airport_id
    INNER JOIN airline a ON f.airline_id = a.airline_id
GROUP BY
    f.flight_id,
    dep_airport.airport_name,
    arr_airport.airport_name
```

The result set shows the following data:

flight_id	sch_departure_time	departure_airport	arrival_airport	total_bookings	average_price
27	2024-01-07 20:00:00.000	Airport 26	Airport 27	1	1,300.34
18	2024-01-05 14:00:00.000	Airport 17	Airport 18	1	998.48
96	2024-01-25 02:00:00.000	Airport 95	Airport 96	1	1,258.55
26	2024-01-07 14:00:00.000	Airport 25	Airport 26	1	747.59
16	2024-01-05 02:00:00.000	Airport 15	Airport 16	1	1,036.02
93	2024-01-06 08:00:00.000	Airport 92	Airport 93	1	276.84
42	2024-01-11 14:00:00.000	Airport 41	Airport 42	1	1,320.23
76	2024-01-20 02:00:00.000	Airport 75	Airport 76	1	529.68
182	2024-02-15 14:00:00.000	Airport 181	Airport 182	1	712.81
143	2024-02-05 20:00:00.000	Airport 142	Airport 143	1	643.74
20	2024-01-06 02:00:00.000	Airport 19	Airport 20	1	320.44
144	2024-02-06 08:00:00.000	Airport 144	Airport 144	1	214.1

6. Create a query that shows all flights flying to a specific country by combining flights, airports and airline, and using the condition on the country name.

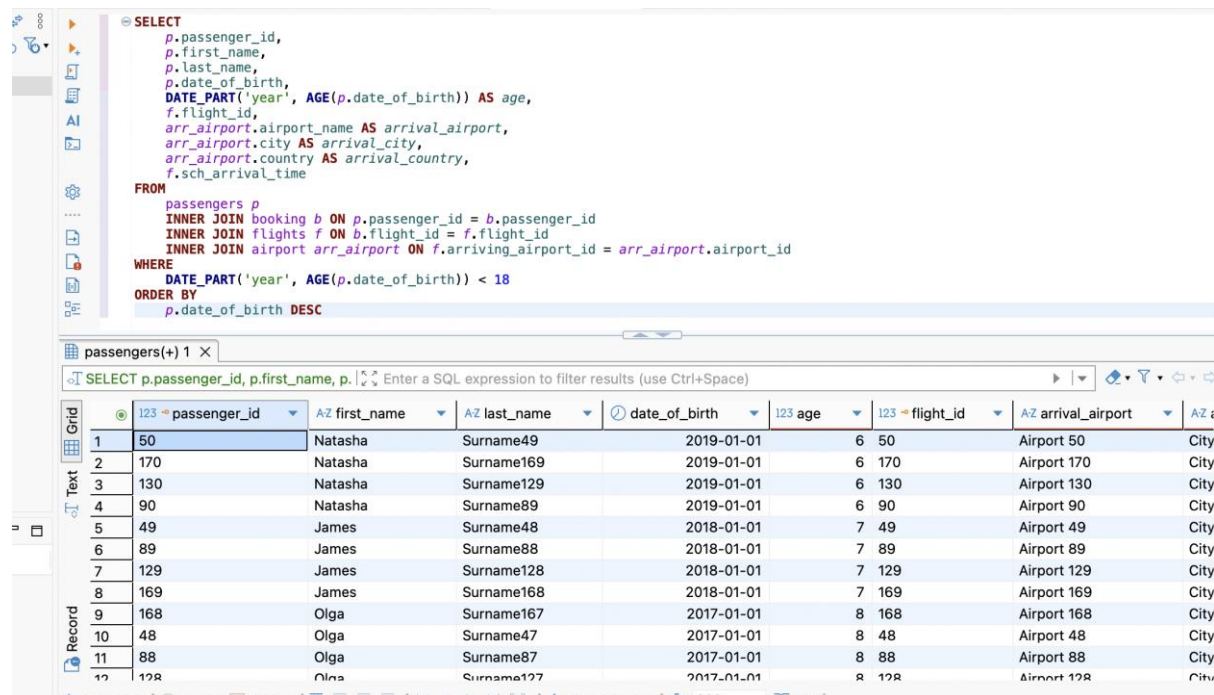
The screenshot shows the DBeaver SQL editor with a query that shows all flights flying to a specific country by combining flights, airports and airline. The query is as follows:

```
SELECT
    f.flight_id,
    f.sch_departure_time,
    f.sch_arrival_time,
    a.airline_name,
    dep_airport.airport_name AS departure_airport,
    dep.country AS departure_country,
    arr_airport.airport_name AS arrival_airport,
    arr.country AS arrival_country
FROM
    flights f
    INNER JOIN airline a ON f.airline_id = a.airline_id
    INNER JOIN airport dep ON f.departing_airport_id = dep.airport_id
    INNER JOIN airport arr ON f.arriving_airport_id = arr.airport_id
ORDER BY
    arr.country
```

The result set shows the following data:

flight_id	sch_departure_time	sch_arrival_time	airline_name	departure_airport	departure_country
187	2024-02-16 20:00:00.000	2024-02-17 00:00:00.000	Airline 187	Airport 186	Japan
37	2024-01-10 08:00:00.000	2024-01-10 12:00:00.000	Airline 37	Airport 36	Japan
137	2024-02-04 08:00:00.000	2024-02-04 12:00:00.000	Airline 137	Airport 136	Japan
107	2024-01-27 20:00:00.000	2024-01-28 00:00:00.000	Airline 107	Airport 106	Japan
67	2024-01-17 20:00:00.000	2024-01-18 00:00:00.000	Airline 67	Airport 66	Japan
167	2024-02-11 20:00:00.000	2024-02-12 00:00:00.000	Airline 167	Airport 166	Japan
27	2024-01-07 20:00:00.000	2024-01-08 00:00:00.000	Airline 27	Airport 26	Japan
127	2024-02-01 20:00:00.000	2024-02-02 00:00:00.000	Airline 127	Airport 126	Japan
47	2024-01-12 20:00:00.000	2024-01-13 00:00:00.000	Airline 47	Airport 46	Japan
147	2024-02-06 20:00:00.000	2024-02-07 00:00:00.000	Airline 147	Airport 146	Japan
57	2024-01-15 08:00:00.000	2024-01-15 12:00:00.000	Airline 57	Airport 56	Japan
157	2024-02-09 08:00:00.000	2024-02-09 12:00:00.000	Airline 157	Airport 156	Japan

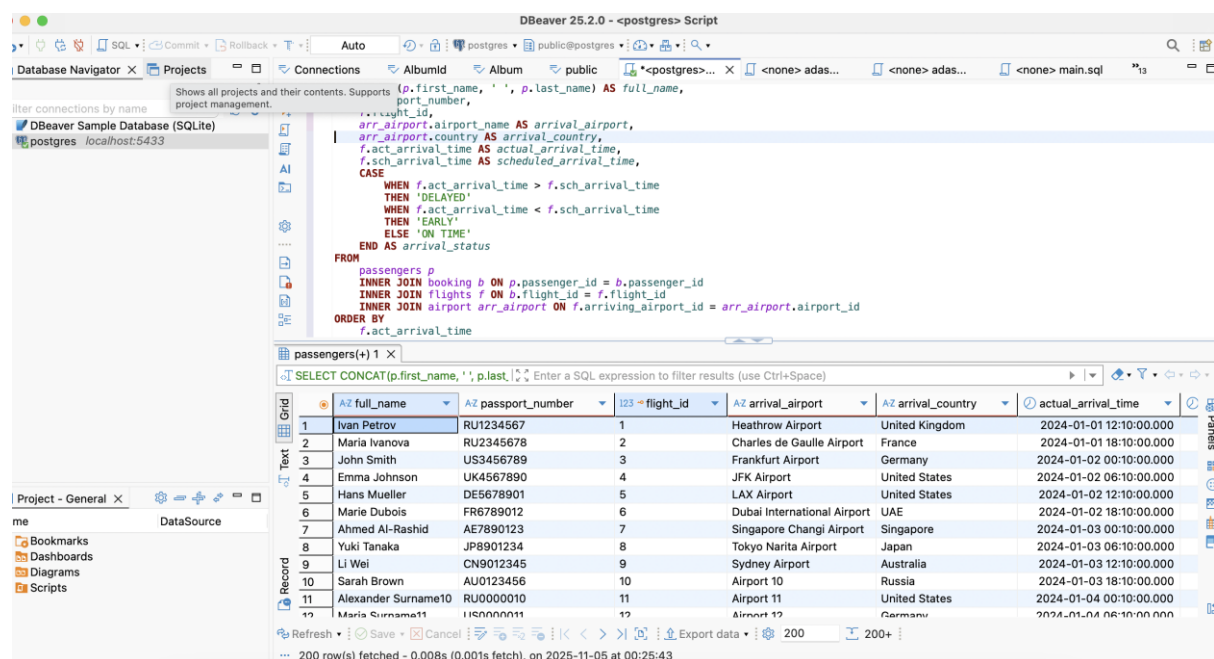
7. Display a list of minor passengers and their arrival destination.



```
SELECT
    p.passenger_id,
    p.first_name,
    p.last_name,
    p.date_of_birth,
    DATE_PART('year', AGE(p.date_of_birth)) AS age,
    f.flight_id,
    arr_airport.airport_name AS arrival_airport,
    arr_airport.city AS arrival_city,
    arr_airport.country AS arrival_country,
    f.sch_arrival_time
FROM
    passengers p
    INNER JOIN booking b ON p.passenger_id = b.passenger_id
    INNER JOIN flights f ON b.flight_id = f.flight_id
    INNER JOIN airport arr_airport ON f.arriving_airport_id = arr_airport.airport_id
WHERE
    DATE_PART('year', AGE(p.date_of_birth)) < 18
ORDER BY
    p.date_of_birth DESC
```

	passenger_id	first_name	last_name	date_of_birth	age	flight_id	arrival_airport	arrival_city	arrival_country
1	50	Natasha	Surname49	2019-01-01	6	50	Airport 50	City	
2	170	Natasha	Surname169	2019-01-01	6	170	Airport 170	City	
3	130	Natasha	Surname129	2019-01-01	6	130	Airport 130	City	
4	90	Natasha	Surname89	2019-01-01	6	90	Airport 90	City	
5	49	James	Surname48	2018-01-01	7	49	Airport 49	City	
6	89	James	Surname88	2018-01-01	7	89	Airport 89	City	
7	129	James	Surname128	2018-01-01	7	129	Airport 129	City	
8	169	James	Surname168	2018-01-01	7	169	Airport 169	City	
9	168	Olga	Surname167	2017-01-01	8	168	Airport 168	City	
10	48	Olga	Surname47	2017-01-01	8	48	Airport 48	City	
11	88	Olga	Surname87	2017-01-01	8	88	Airport 88	City	
12	128	Olga	Surname127	2017-01-01	8	128	Airport 128	City	

8. Display the passenger's full name, passport number, and the passenger's current time of arrival at the destination.



```
SELECT
    p.first_name,
    p.last_name,
    p.passport_number,
    f.flight_id,
    arr_airport.airport_name AS arrival_airport,
    arr_airport.country AS arrival_country,
    f.act_arrival_time AS actual_arrival_time,
    f.sch_arrival_time AS scheduled_arrival_time,
    CASE
        WHEN f.act_arrival_time > f.sch_arrival_time
        THEN 'DELAYED'
        WHEN f.act_arrival_time < f.sch_arrival_time
        THEN 'EARLY'
        ELSE 'ON TIME'
    END AS arrival_status
FROM
    passengers p
    INNER JOIN booking b ON p.passenger_id = b.passenger_id
    INNER JOIN flights f ON b.flight_id = f.flight_id
    INNER JOIN airport arr_airport ON f.arriving_airport_id = arr_airport.airport_id
ORDER BY
    f.act_arrival_time
```

	full_name	passport_number	flight_id	arrival_airport	arrival_country	actual_arrival_time	arrival_status
1	Ivan Petrov	RU1234567	1	Heathrow Airport	United Kingdom	2024-01-01 12:10:00.000	
2	Maria Ivanova	RU2345678	2	Charles de Gaulle Airport	France	2024-01-01 18:10:00.000	
3	John Smith	US3456789	3	Frankfurt Airport	Germany	2024-01-02 00:10:00.000	
4	Emma Johnson	UK4567890	4	JFK Airport	United States	2024-01-02 06:10:00.000	
5	Hans Mueller	DE5678901	5	LAX Airport	United States	2024-01-02 12:10:00.000	
6	Marie Dubois	FR6789012	6	Dubai International Airport	UAE	2024-01-02 18:10:00.000	
7	Ahmed Al-Rashid	AE7890123	7	Singapore Changi Airport	Singapore	2024-01-03 00:10:00.000	
8	Yuki Tanaka	JP8901234	8	Tokyo Narita Airport	Japan	2024-01-03 06:10:00.000	
9	Li Wei	CN9012345	9	Sydney Airport	Australia	2024-01-03 12:10:00.000	
10	Sarah Brown	AU0123456	10	Airport 10	Russia	2024-01-03 18:10:00.000	
11	Alexander Surname10	RU0000010	11	Airport 11	United States	2024-01-04 00:10:00.000	
12	Maria Surname11	RU0000011	12	Airport 12	Germany	2024-01-04 06:10:00.000	

9. Print a list of flights where the airline's home country and origin country are the same. Group them by the airport country.

The screenshot shows the DBeaver SQL editor interface. The left sidebar displays the 'Database Navigator' with a connection to 'postgres' at 'localhost:5433'. The main editor area contains the following SQL query:

```
select
    f.flight_id,
    f.sch_departure_time,
    f.sch_arrival_time,
    a.airline_name,
    dep.country AS departure_country,
    dep.airport_name AS departure_airport,
    dep.city AS departure_city,
    arr.airport_name AS arrival_airport,
    arr.city AS arrival_city,
    arr.country AS arrival_country
FROM
    flights f
    INNER JOIN airline a ON f.airline_id = a.airline_id
    INNER JOIN airport dep ON f.departing_airport_id = dep.airport_id
    INNER JOIN airport arr ON f.arriving_airport_id = arr.airport_id
WHERE
    a.airline_country = dep.country
ORDER BY
    dep.country
```

Below the query editor, the 'Grid' view shows the results of the query. The columns are: flight_id, sch_departure_time, sch_arrival_time, airline_name, departure_country, and departure_airport. The grid is currently empty, indicating no data was returned by the query.

At the bottom of the interface, a status bar indicates 'No data- 0.004s, on 2025-11-05 at 00:27:38'.