

1. Create a view to show details of all flights that are departing on a specific date.

The screenshot shows the DBeaver interface with the SQL tab selected. A new view named 'all\_flights' is being created with the following SQL code:

```

CREATE VIEW all_flights AS
SELECT f.*
FROM flights f
WHERE CAST(f.sch_departure_time AS date) = '2024-01-02'

```

Below the code, a query is run:

```

select * from all_flights

```

The results are displayed in a grid:

	l23 flight_id	sch_departure_time	sch_arrival_time	l23 departing_airport_id	l23 arriving_airport_id	AZ depart
1	6	2024-01-02 14:00:00.000	2024-01-02 18:00:00.000	6	7	G07
2	5	2024-01-02 08:00:00.000	2024-01-02 12:00:00.000	5	6	G06
3	7	2024-01-02 20:00:00.000	2024-01-03 00:00:00.000	7	8	G08
4	4	2024-01-02 02:00:00.000	2024-01-02 06:00:00.000	4	5	G05

2. Create a view that shows bookings for flights scheduled to depart within the next week.

The screenshot shows the DBeaver interface with the SQL tab selected. A new view named 'bookings' is being created with the following SQL code:

```

--2. Create a view that shows bookings for flights scheduled to depart within the next week.
create view bookings as
select b.*, f.sch_departure_time
from booking b
join flights f on f.flight_id=b.flight_id
WHERE DATE(f.sch_departure_time) BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '7 day'

```

Below the code, a query is run:

```

SELECT * FROM bookings;

```

The results are displayed in a grid:

	l23 booking_id	l23 flight_id	l23 passenger_id	AZ booking_platform	created_at	updated_at	AZ status
							No data- 0.0s, on 2025-11-18 at 12:47:38

3. Create a view to show the top 5 most popular flight routes based on the number of bookings.

The screenshot shows the DBeaver interface with the SQL tab selected. A CREATE VIEW statement is being typed into the editor:

```

CREATE VIEW top_popular AS
SELECT
    f.departing_airport_id,
    f.arriving_airport_id,
    COUNT(*) AS bookings_count
FROM booking b
JOIN flights f ON b.flight_id = f.flight_id
GROUP BY f.departing_airport_id, f.arriving_airport_id
ORDER BY bookings_count DESC
LIMIT 5;

SELECT * FROM top_popular;

```

Below the editor, a results grid titled "top\_popular 1" displays the data:

	departing_airport_id	arriving_airport_id	bookings_count
1	193	194	1
2	7	8	1
3	196	197	1
4	186	187	1
5	17	18	1

4. Create a view that lists all flights for a specific airline.

The screenshot shows the DBeaver interface with the SQL tab selected. A CREATE VIEW statement is being typed into the editor:

```

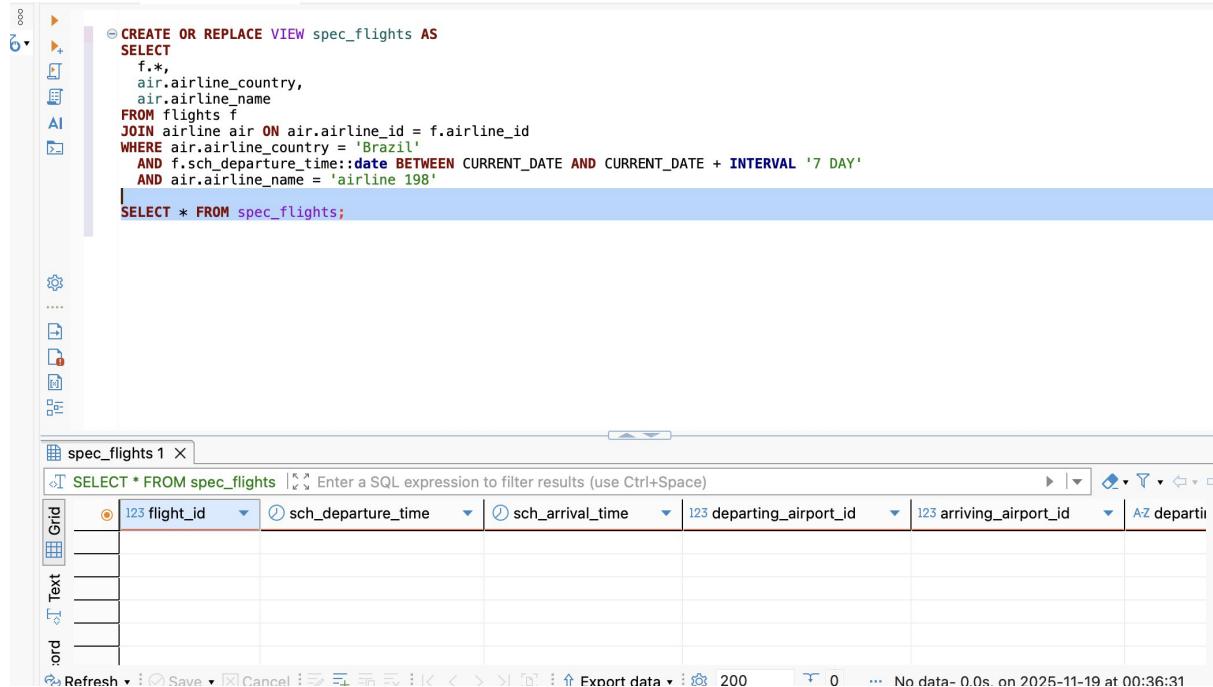
CREATE VIEW spec_flights AS
SELECT
    f.*,
    air.airline_country,
    air.airline_name
FROM flights f
JOIN airline air ON air.airline_id=f.airline_id
WHERE airline_country='Brazil'

```

Below the editor, a results grid titled "spec\_flights 1" displays the data:

flight_id	sch_departure_time	sch_arrival_time	departing_airport_id	arriving_airport_id	depa
189	2024-02-17 08:00:00.000	2024-02-17 12:00:00.000	189	190	G40
89	2024-01-23 08:00:00.000	2024-01-23 12:00:00.000	89	90	G40
199	2024-02-19 20:00:00.000	2024-02-20 00:00:00.000	199	200	G50
99	2024-01-25 20:00:00.000	2024-01-26 00:00:00.000	99	100	G50
179	2024-02-14 20:00:00.000	2024-02-15 00:00:00.000	179	180	G30
79	2024-01-20 20:00:00.000	2024-01-21 00:00:00.000	79	80	G30

5. Modify the view created in task 4 to show only flights departing within the next 7 days for a specific airline.



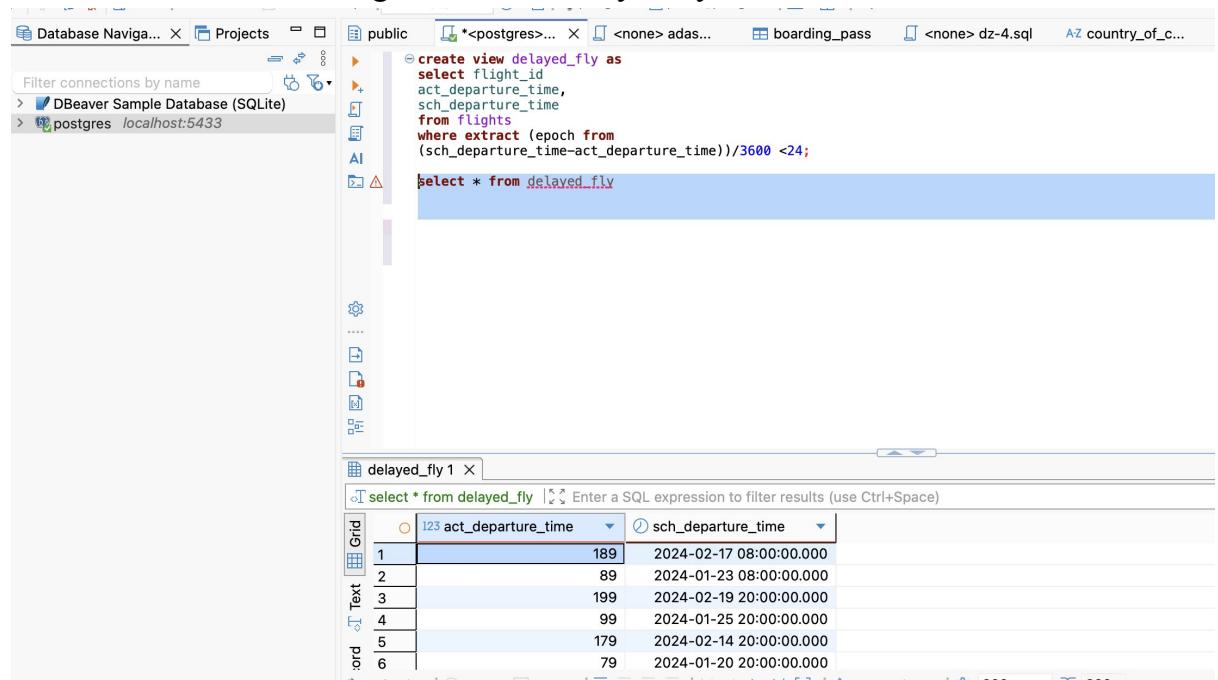
```

CREATE OR REPLACE VIEW spec_flights AS
SELECT
    f.*,
    air.airline_country,
    air.airline_name
FROM flights f
JOIN airline air ON air.airline_id = f.airline_id
WHERE air.airline_country = 'Brazil'
    AND f.sch_departure_time::date BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '7 DAY'
    AND air.airline_name = 'airline 198'
SELECT * FROM spec_flights;

```

The screenshot shows the DBeaver interface with a SQL editor window containing the creation of a view named 'spec\_flights'. The query selects all columns from the 'flights' table and joins it with the 'airline' table to filter flights departing from Brazil within the next 7 days for the airline 'airline 198'. Below the editor is a results grid titled 'spec\_flights 1' which is currently empty, indicating no data has been returned.

6. Create a view to show flights that are delayed by more than 24 hours.



```

create view delayed_fly as
select flight_id,
       act_departure_time,
       sch_departure_time
  from flights
 where extract (epoch from
 (sch_departure_time-act_departure_time))/3600 <24;
select * from delayed_fly

```

The screenshot shows the DBeaver interface with a SQL editor window containing the creation of a view named 'delayed\_fly'. The query selects flight\_id, act\_departure\_time, and sch\_departure\_time from the 'flights' table, filtering for flights where the difference between actual and scheduled departure times is less than 24 hours. Below the editor is a results grid titled 'delayed\_fly 1' showing six rows of data, each with a flight ID and two timestamp columns.

	123 act_departure_time	sch_departure_time
1	189	2024-02-17 08:00:00.000
2	89	2024-01-23 08:00:00.000
3	199	2024-02-19 20:00:00.000
4	99	2024-01-25 20:00:00.000
5	179	2024-02-14 20:00:00.000
6	79	2024-01-20 20:00:00.000

7. Create a view in which you can display the full name and country of origin of passengers who made bookings on Leffler-Thompson platform. Then show the list of that passengers.

The screenshot shows a DBeaver interface with the following details:

- Editor Tab Bar:** Contains tabs for public (\*<postgres>...), adas..., boarding\_pass, <none> dz-4.sql, country\_of\_c..., airline, and a file icon with '5'.
- Toolbar:** Includes icons for New, Open, Save, Print, Copy, Paste, Find, Replace, Undo, Redo, and a magnifying glass.
- Left Sidebar:** Shows a tree view with 'public' expanded, containing 'CREATE VIEW lt\_platform AS'. Below it are 'select \* from lt\_platform' and a 'SQL' tab.
- Editor Content:** Displays the following SQL code:

```
CREATE VIEW lt_platform AS
SELECT p.first_name,
       p.last_name,
       p.country_of_citizenship,
       b.booking_platform
  FROM passengers p
 JOIN booking b ON p.passenger_id = b.passenger_id
 WHERE b.booking_platform = 'Leffler-Thompson';

select * from lt_platform
```
- Results Grid:** A table titled 'lt\_platform 1' with four columns: first\_name, last\_name, country\_of\_citizenship, and booking\_platform. The first row is highlighted with a red border, and the column headers are also red.
- Filter Bar:** Below the grid, it says 'Enter a SQL expression to filter results (use Ctrl+Space)'.
- Bottom Navigation:** Includes icons for Grid, Text, and Card, along with a back and forward arrow.

8. Create a view that shows top 10 most visited countries.

The screenshot shows a DBeaver interface with multiple tabs at the top:

- public
- \*<postgres>...
- boarding\_pass
- <none> dz-4.sql
- AZ country\_of\_c...
- airline
- booking

The top-left editor contains the following SQL code:

```
create view top10 as
select ap.country, COUNT(*) as visits
from flights f
join airport ap on f.arriving_airport_id = ap.airport_id
group by ap.country
order by visits desc
limit 10;

select * from top10
```

The bottom-right area displays the results of the query in a grid:

Rank	Country	Visits
1	United States	21
2	Australia	20
3	United Kingdom	20
4	Germany	20
5	Japan	20
6	Russia	20

9. Update any of the created views by adding new information in the view table. Show results.

The screenshot shows a PostgreSQL client interface. At the top, a code editor displays a SQL script defining a view named `upcoming_week_bookings`. The view selects columns from tables `f` and `b`, including `act_departure_time`, `sch_departure_time`, and `flight_status` based on a condition involving `date BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '7 day'`. Below the code editor is a results grid titled "upcoming\_week\_bookings 1". The grid has columns: `booking_id`, `flight_id`, `passenger_id`, `booking_platform`, `created_at`, `updated_at`, and `status`. The results grid is currently empty, showing only the header row.

10. Drop all existing views.

The screenshot shows two separate PostgreSQL client sessions. Both sessions are executing the same SQL command: `DROP VIEW IF EXISTS all_flights, bookings, top_popular, spec_flights, delayed_fly, lt_platform, upcoming_week_bookings, top10;`. The output pane for both sessions displays messages indicating that the views "bookings" and "top\_popular" do not exist and therefore are being skipped. The sessions are identified by session numbers 1 and 2 at the top left of their respective panes.