

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

The screenshot shows a database interface with a SQL editor at the top and a results grid below. The SQL editor contains the following code:

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

select * from all_flights
```

The results grid displays the following data:

	123 flight_id	sch_departure_time	sch_arrival_time	123 departing_airport_id	123 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 a database interface with a SQL editor at the top and a results grid below. The SQL editor contains the following 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'

SELECT * FROM bookings;
```

The results grid displays the following data:

	123 booking_id	123 flight_id	123 passenger_id	AZ booking_platform	created_at	updated_at	AZ status

At the bottom of the grid, a message states: "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 SQL editor with a PostgreSQL connection. The SQL editor contains the following code:

```
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;
```

The results window shows the following data:

	123 departing_airport_id	123 arriving_airport_id	123 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 SQL editor with a PostgreSQL connection. The SQL editor contains the following code:

```
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'

SELECT * FROM spec_flights;
```

The results window shows the following data:

	123 flight_id	sch_departure_time	sch_arrival_time	123 departing_airport_id	123 arriving_airport_id	AZ depa
1	189	2024-02-17 08:00:00.000	2024-02-17 12:00:00.000	189	190	G40
2	89	2024-01-23 08:00:00.000	2024-01-23 12:00:00.000	89	90	G40
3	199	2024-02-19 20:00:00.000	2024-02-20 00:00:00.000	199	200	G50
4	99	2024-01-25 20:00:00.000	2024-01-26 00:00:00.000	99	100	G50
5	179	2024-02-14 20:00:00.000	2024-02-15 00:00:00.000	179	180	G30
6	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.

The screenshot shows the DBeaver SQL editor with the following SQL code:

```
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;
```

Below the editor, the results of the query are displayed in a table grid. The table has columns: flight_id, sch_departure_time, sch_arrival_time, departing_airport_id, arriving_airport_id, and departure. The first row shows flight_id 123 with a departure time of 2025-11-19 00:36:31.

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

The screenshot shows the DBeaver SQL editor with the following SQL code:

```
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
```

Below the editor, the results of the query are displayed in a table grid. The table has columns: flight_id, act_departure_time, and sch_departure_time. The first row shows flight_id 189 with an actual departure time of 2024-02-17 08:00:00.000 and a scheduled departure time of 2024-01-23 08:00:00.000.

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 PostgreSQL IDE with a SQL editor and a results grid. The SQL editor contains the following 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
```

The results grid, titled 'lt_platform 1', displays the following data:

AZ first_name	AZ last_name	AZ country_of_citizenship	AZ booking_platform

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

The screenshot shows a PostgreSQL IDE with a SQL editor and a results grid. The SQL editor contains the following 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 results grid, titled 'top10 1', displays the following data:

AZ country	123 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 database IDE with a SQL editor at the top and a results pane below. The SQL editor contains the following code:

```
create view upcoming_week_bookings as
  h.departure_time,
  en fact departure_time > f.sch_departure_time
  en 'delayed' else 'on time' END AS flight_status
ON b.flight_id = f.flight_id
parture_time::date BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '7 day';

```

The results pane shows a table with the following columns: booking_id, flight_id, passenger_id, booking_platform, created_at, updated_at, and status. The table is currently empty.

10. Drop all existing views.

The screenshot shows a database IDE with a SQL editor at the top and an output pane below. The SQL editor contains the following code:

```
DROP VIEW IF EXISTS
all_flights,
bookings,
top_popular,
spec_flights,
delayed_fly,
lt_platform,
upcoming_week_bookings,
top10;

```

The output pane shows the following message:

```
view "bookings" does not exist, skipping
view "top_popular" does not exist, skipping
view "all_flights" does not exist, skipping
view "bookings" does not exist, skipping
view "top_popular" does not exist, skipping
view "spec_flights" does not exist, skipping
view "delayed_fly" does not exist, skipping
view "lt_platform" does not exist, skipping
view "upcoming_week_bookings" does not exist, skipping
view "top10" does not exist, skipping

```

The results pane shows a table with the following columns: Name, Value. The table is currently empty.