

## Laboratory work 8

### VIEW.

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

The screenshot displays the PostgreSQL Labworks interface. On the left, the 'Object Explorer' pane shows a database schema with tables like 'airport', 'baggage', 'boarding\_pass', 'booking', 'booking\_flight', 'flights', and 'passengers'. The 'flights' table is selected, showing its columns: flight\_id, flight\_no, scheduled\_departure, scheduled\_arrival, departure\_airport\_id, arrival\_airport\_id, departing\_gate, arriving\_gate, airline\_id, status, actual\_departure, actual\_arrival, created\_at, and update\_at. The 'scheduled\_departure' column is highlighted. The main query editor on the right contains the following SQL code:

```
1 CREATE VIEW date_flight AS
2 SELECT * FROM flights
3 WHERE DATE(scheduled_departure) = '2025-01-01';
4
5
```

Below the query editor, the 'Data Output' pane shows the message: 'CREATE VIEW' and 'Query returned successfully in 115 msec.' A green status bar at the bottom right confirms: 'Query returned successfully in 115 msec.' The bottom status bar indicates 'Total rows: Query complete 00:00:00.115'.

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

The screenshot shows a PostgreSQL IDE interface. On the left, the 'Object Explorer' pane displays a database schema with tables like 'airports', 'baggage', 'booking', 'booking\_flight', and 'flights'. The 'booking' table is selected, showing its columns: booking\_id, passenger\_id, booking\_platform, created\_at, update\_at, status, and price. The main SQL editor contains the following query:

```

1 CREATE VIEW bookings_next_week AS
2 SELECT b.booking_id, b.created_at, f.flight_no, f.scheduled_departure
3 FROM booking b
4 JOIN booking_flight bf ON b.booking_id = bf.booking_id
5 JOIN flights f ON bf.flight_id = f.flight_id
6 WHERE EXTRACT(YEAR FROM f.scheduled_departure) = EXTRACT(YEAR FROM CURRENT_DATE)
7       AND EXTRACT(WEEK FROM f.scheduled_departure) = EXTRACT(WEEK FROM CURRENT_DATE) + 1;

```

The 'Data Output' pane at the bottom shows the message: 'CREATE VIEW' and 'Query returned successfully in 56 msec.' A status bar at the bottom indicates 'Total rows: Query complete 00:00:00.056'.

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

The screenshot shows the same PostgreSQL IDE interface. The 'Object Explorer' pane is still open. The main SQL editor contains the following query:

```

1 CREATE VIEW top5_routes AS
2 SELECT f.departure_airport_id, f.arrival_airport_id,
3 COUNT(*) AS total_bookings
4 FROM flights f
5 JOIN booking_flight bf ON f.flight_id = bf.flight_id
6 GROUP BY f.departure_airport_id, f.arrival_airport_id
7 ORDER BY total_bookings DESC
8 LIMIT 5;

```

The 'Data Output' pane at the bottom shows the message: 'CREATE VIEW' and 'Query returned successfully in 95 msec.' A status bar at the bottom indicates 'Total rows: Query complete 00:00:00.095'.

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

The screenshot shows a database management tool interface. On the left, the 'Object Explorer' displays a schema with tables like 'airports', 'baggage', 'booking', and 'flights'. The 'flights' table is selected, showing its columns: flight\_id, flight\_no, scheduled\_departure, scheduled\_arrival, departure\_airport\_id, arrival\_airport\_id, departing\_gate, and arriving\_gate. The main pane shows a SQL query: `CREATE VIEW airline_flights AS SELECT * FROM flights WHERE airline_id = 1;`. The 'Data Output' pane shows the result: 'CREATE VIEW' and 'Query returned successfully in 56 msec.'.

Object Explorer

- Columns (7)
  - airport\_id
  - airport\_name
  - country
  - state
  - city
  - created\_at
  - update\_at
- Constraints
- Indexes
- RLS Policies
- Rules
- Triggers
- baggage
- baggage\_check
- boarding\_pass
- booking
  - Columns (7)
    - booking\_id
    - passenger\_id
    - booking\_platform
    - created\_at
    - update\_at
    - status
    - price
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
- booking\_flight
- flights
  - Columns (14)
    - flight\_id
    - flight\_no
    - scheduled\_departure
    - scheduled\_arrival
    - departure\_airport\_id
    - arrival\_airport\_id
    - departing\_gate
    - arriving\_gate

LABWORKS/postgres@PostgreSQL 17

Query

```
1 CREATE VIEW airline_flights AS SELECT * FROM flights WHERE airline_id = 1;
```

Data Output

CREATE VIEW

Query returned successfully in 56 msec.

Total rows: Query complete 00:00:00.056

LF Ln 1, Col 32

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 same database management tool interface. The SQL query is now: `CREATE OR REPLACE VIEW airline_flights AS SELECT * FROM flights WHERE airline_id = 1 AND scheduled_departure BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '7 days';`. The 'Data Output' pane shows the result: 'CREATE VIEW' and 'Query returned successfully in 43 msec.'.

Object Explorer

- Columns (7)
  - airport\_id
  - airport\_name
  - country
  - state
  - city
  - created\_at
  - update\_at
- Constraints
- Indexes
- RLS Policies
- Rules
- Triggers
- baggage
- baggage\_check
- boarding\_pass
- booking
  - Columns (7)
    - booking\_id
    - passenger\_id
    - booking\_platform
    - created\_at
    - update\_at
    - status
    - price
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
- booking\_flight
- flights
  - Columns (14)
    - flight\_id
    - flight\_no
    - scheduled\_departure
    - scheduled\_arrival
    - departure\_airport\_id
    - arrival\_airport\_id
    - departing\_gate
    - arriving\_gate

LABWORKS/postgres@PostgreSQL 17

Query

```
1 CREATE OR REPLACE VIEW airline_flights AS
2 SELECT * FROM flights WHERE airline_id = 1
3 AND scheduled_departure BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '7 days';
```

Data Output

CREATE VIEW

Query returned successfully in 43 msec.

Total rows: Query complete 00:00:00.043

LF Ln 3, Col 83

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

The screenshot shows a PostgreSQL IDE interface. On the left, the 'Object Explorer' pane displays a database schema with tables like 'airports', 'baggage', 'boarding\_pass', 'booking', 'booking\_flight', and 'flights'. The 'flights' table is expanded, showing columns: flight\_id, flight\_no, scheduled\_departure, scheduled\_arrival, departure\_airport\_id, arrival\_airport\_id, departing\_gate, and arriving\_gate. The main editor pane shows a SQL query to create a view named 'delayed\_24h'.

```
1 CREATE VIEW delayed_24h AS
2 SELECT flight_id, flight_no, scheduled_departure, actual_departure, actual_departure - scheduled_departure AS delay
3 FROM flights
4 WHERE actual_departure IS NOT NULL AND actual_departure > scheduled_departure + INTERVAL '24 hours';
```

The 'Data Output' pane at the bottom shows the message: 'CREATE VIEW' and 'Query returned successfully in 50 msec.' A status bar at the bottom indicates 'Total rows: Query complete 00:00:00.050'.

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 interface. On the left, the 'Object Explorer' pane displays a database schema with tables like 'airports', 'passengers', 'bookings', and 'flights'. The 'bookings' table is selected, showing its columns: booking\_id, passenger\_id, booking\_platform, created\_at, update\_at, status, and price. The main query editor displays the following SQL code:

```

1 CREATE VIEW lt_passengers AS
2 SELECT DISTINCT p.first_name || ' ' || p.last_name AS full_name, p.country_of_citizenship AS country
3 FROM passengers p
4 JOIN booking b ON p.passenger_id = b.passenger_id
5 WHERE b.booking_platform = 'Leffler-Thompson';

```

The 'Data Output' pane at the bottom shows the message: 'CREATE VIEW' and 'Query returned successfully in 52 msec.' A status bar at the bottom indicates 'Total rows: Query complete 00:00:00.052'.

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

The screenshot shows the same PostgreSQL IDE interface. The 'Object Explorer' pane is still open. The main query editor displays the following SQL code:

```

1 CREATE VIEW top10_countries AS
2 SELECT a.country, COUNT(*) AS visits
3 FROM flights f
4 JOIN airport a ON f.arrival_airport_id = a.airport_id
5 GROUP BY a.country ORDER BY visits DESC
6 LIMIT 10;

```

The 'Data Output' pane at the bottom shows the message: 'CREATE VIEW' and 'Query returned successfully in 55 msec.' A status bar at the bottom indicates 'Total rows: Query complete 00:00:00.055'.

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

Object Explorer

- Columns (7)
  - airport\_id
  - airport\_name
  - country
  - state
  - city
  - created\_at
  - update\_at
- Constraints
- Indexes
- RLS Policies
- Rules
- Triggers
- baggage
- baggage\_check
- boarding\_pass
- booking
  - Columns (7)
    - booking\_id
    - passenger\_id
    - booking\_platform
    - created\_at
    - update\_at
    - status
    - price
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
  - booking\_flight
  - flights
    - Columns (14)
      - flight\_id
      - flight\_no
      - scheduled\_departure
      - scheduled\_arrival
      - departure\_airport\_id
      - arrival\_airport\_id
      - departing\_gate
      - arriving\_gate

LABWORKS/postgres@PostgreSQL 17

Query

```

1 CREATE OR REPLACE VIEW airline_flights AS
2 SELECT f.flight_id, f.flight_no, f.airline_id, f.scheduled_departure, f.departure_airport_id, f.arrival_airport_id,
3 da.airport_name AS departure_airport_name, aa.airport_name AS arrival_airport_name
4 FROM flights f
5 JOIN airport da ON f.departure_airport_id = da.airport_id
6 JOIN airport aa ON f.arrival_airport_id = aa.airport_id;

```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 97 msec.

Total rows: Query complete 00:00:00.097

LF Ln 3, Col 44

Object Explorer

- Columns (7)
  - airport\_id
  - airport\_name
  - country
  - state
  - city
  - created\_at
  - update\_at
- Constraints
- Indexes
- RLS Policies
- Rules
- Triggers
- baggage
- baggage\_check
- boarding\_pass
- booking
  - Columns (7)
    - booking\_id
    - passenger\_id
    - booking\_platform
    - created\_at
    - update\_at
    - status
    - price
  - Constraints
  - Indexes
  - RLS Policies
  - Rules
  - Triggers
  - booking\_flight
  - flights
    - Columns (14)
      - flight\_id
      - flight\_no
      - scheduled\_departure
      - scheduled\_arrival
      - departure\_airport\_id
      - arrival\_airport\_id
      - departing\_gate
      - arriving\_gate

LABWORKS/postgres@PostgreSQL 17

Query

```

1 SELECT * FROM airline_flights;

```

Data Output Messages Notifications

Showing rows: 1 to 993 Page No: 1 of 1

flight_id	flight_no	airline_id	scheduled_departure	departure_airport_id	arrival_airport_id	departure_airport_name	arrival_airport_name
1	1 US-CT	39	2024-01-22	12	15	Elorza Airport	Ocean Falls Seaplane Base
2	2 US-NM	34	2023-07-21	13	16	Figari Sud-Corse Airport	Zephyrhills Municipal Airport
3	3 FI-OL	34	2023-03-29	18	12	Darchula Airport	Elorza Airport
4	4 RU-KR	33	2024-01-02	3	19	Lime Acres Finsch Mine Airport	Longana Airport
5	5 RO-DJ	14	2023-07-03	6	2	Hana Airport	Alert Bay Airport
6	6 CA-SK	34	2023-07-07	18	1	Darchula Airport	Akunag Heliport
7	7 AU-TAS	10	2023-10-12	15	18	Ocean Falls Seaplane Base	Darchula Airport
8	8 US-AZ	12	2023-07-29	13	6	Figari Sud-Corse Airport	Hana Airport
9	9 IN-OR	13	2023-05-18	6	9	Hana Airport	Pitalito Airport
10	10 AU-NT	29	2023-11-25	19	16	Longana Airport	Zephyrhills Municipal Airport
11	11 TH-57	46	2023-03-28	5	5	Delta County Airport	Delta County Airport
12	12 CA-NL	27	2023-03-22	12	13	Elorza Airport	Figari Sud-Corse Airport
13	13 BR-PE	1	2024-01-16	13	20	Figari Sud-Corse Airport	Mellila Airport
14	14 US-IN	8	2023-07-24	18			
15	15 MI-7	16	2023-11-02	1			

Total rows: 993 Query complete 00:00:00.084

Successfully run. Total query runtime: 84 msec. 993 rows affected.

LF Ln 1, Col 31

10. Drop all existing views.

Object Explorer

airports

Columns (7)

- airport\_id
- airport\_name
- country
- state
- city
- created\_at
- update\_at

> Constraints

> Indexes

> RLS Policies

> Rules

> Triggers

baggage

baggage\_check

boarding\_pass

booking

Columns (7)

- booking\_id
- passenger\_id
- booking\_platform
- created\_at
- update\_at
- status
- price

> Constraints

> Indexes

> RLS Policies

> Rules

> Triggers

booking\_flight

flights

Columns (14)

- flight\_id
- flight\_no
- scheduled\_departure
- scheduled\_arrival
- departure\_airport\_id
- arrival\_airport\_id
- departing\_gate
- arriving\_gate

LABWORKS/postgres@PostgreSQL 17

Query

Query History

```
1 DROP VIEW IF EXISTS date_flight;
2 DROP VIEW IF EXISTS bookings_next_week;
3 DROP VIEW IF EXISTS top5_routes;
4 DROP VIEW IF EXISTS airline_flights;
5 DROP VIEW IF EXISTS delayed_24h;
6 DROP VIEW IF EXISTS lt_passengers;
7 DROP VIEW IF EXISTS top10_countries;
```

Data Output

Messages

Notifications

DROP VIEW

Query returned successfully in 54 msec.

Total rows: Query complete 00:00:00.054

LF Ln 7, Col 37

✓ Query returned successfully in 54 msec. ✕