

LAB8 Abdykamat Adilet. VIEW

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

```
CREATE VIEW flights_on_date AS SELECT * FROM flights  
WHERE DATE(sch_departure_time) = '2025-01-15';
```

The screenshot shows a database interface with two main sections: 'Query' and 'Messages'. In the 'Query' section, two lines of SQL code are displayed:

```
1 ✓ CREATE VIEW flights_on_date AS SELECT * FROM flights  
2 WHERE DATE(sch_departure_time) = '2025-01-15';
```

In the 'Messages' section, the output of the query is shown:

```
CREATE VIEW  
Query returned successfully in 101 msec.
```

A green success message at the bottom right of the 'Messages' section states: "✓ Query returned successfully in 101 msec. ✎".

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

```
CREATE VIEW ext_week AS  
SELECT b.booking_id, bf.flight_id, f.sch_departure_time FROM booking b  
JOIN booking_flight bf on b.booking_id = bf.booking_id  
JOIN flights f on f.flight_id = bf.flight_id  
WHERE f.sch_departure_time BETWEEN now() AND now() + INTERVAL '7 days';
```

The screenshot shows a MySQL Workbench interface. In the top-left pane, under the 'Query' tab, there is a code editor containing the following SQL code:

```

1 ✓ CREATE VIEW ext_week AS
2   SELECT b.booking_id, bf.flight_id, f.sch_departure_time FROM booking b
3   JOIN booking_flight bf ON b.booking_id = bf.booking_id
4   JOIN flights f ON f.flight_id = bf.flight_id
5   WHERE f.sch_departure_time BETWEEN now() AND now() + INTERVAL '7 days';

```

In the bottom-left pane, under the 'Messages' tab, the output is:

```

CREATE VIEW

Query returned successfully in 104 msec.

Total rows: 0 Query complete 00:00:00 104

```

A green message box at the bottom right indicates: "✓ Query returned successfully in 104 msec. X". Below the message box, the status bar shows: "CREATE 1 In 5 Col 62".

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

```

CREATE VIEW top5_routes AS
SELECT
    dep.airport_name AS from_airport,
    arr.airport_name AS to_airport,
    COUNT(*) AS total_bookings
FROM booking_flight bf
JOIN flights f ON bf.flight_id = f.flight_id
JOIN airport dep ON f.departing_airport_id = dep.airport_id
JOIN airport arr ON f.arriving_airport_id = arr.airport_id
GROUP BY dep.airport_name, arr.airport_name
ORDER BY total_bookings DESC
LIMIT 5;

```

Query History

```

1 ✓ CREATE VIEW top5_routes AS
2   SELECT
3     dep.airport_name AS from_airport,
4     arr.airport_name AS to_airport,
5     COUNT(*) AS total_bookings
6   FROM booking_flight bf
7   JOIN flights f ON bf.flight_id = f.flight_id
8   JOIN airport dep ON f.departing_airport_id = dep.airport_id
9   JOIN airport arr ON f.arriving_airport_id = arr.airport_id
10  GROUP BY dep.airport_name, arr.airport_name
11  ORDER BY total_bookings DESC
12  LIMIT 5;

```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 113 msec.

Total rows: 0 Query complete 00:00:00.113 ✓ CRLF In 12 Col 9

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

```

CREATE VIEW flights_by_airline AS
SELECT f.flight_id, a.airline_name, f.sch_departure_time, f.sch_arrival_time
FROM flights f
JOIN airline a on f.airline_id = a.airline_id
WHERE a.airline_name = 'Air Astana';

```

Query History

```

1 ✓ CREATE VIEW flights_by_airline AS
2   SELECT f.flight_id, a.airline_name, f.sch_departure_time, f.sch_arrival_time
3   FROM flights f
4   JOIN airline a on f.airline_id = a.airline_id
5   WHERE a.airline_name = 'Air Astana';

```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 108 msec.

✓ Query returned successfully in 108 msec. ✘

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 flights_by_airline AS  
  
SELECT f.flight_id, a.airline_name,  
       f.sch_departure_time, f.sch_arrival_time  
  
FROM flights f  
  
JOIN airline a ON f.airline_id = a.airline_id  
  
WHERE a.airline_name = 'Air Astana' AND f.sch_departure_time  
      BETWEEN now() AND now() + INTERVAL '7 days';
```

The screenshot shows a database interface with two main sections: 'Query History' and 'Messages'. In the 'Query History' section, a SQL script is displayed with numbered lines 1 through 7. Lines 1-6 define the view, and line 7 specifies the time range. In the 'Messages' section, the output of the query is shown, starting with 'CREATE VIEW'. Below it, a message states 'Query returned successfully in 93 msec.' A green success icon is present in the status bar at the bottom right.

```
1 CREATE OR REPLACE VIEW flights_by_airline AS  
2   SELECT f.flight_id, a.airline_name,  
3         f.sch_departure_time, f.sch_arrival_time  
4   FROM flights f  
5   JOIN airline a ON f.airline_id = a.airline_id  
6   WHERE a.airline_name = 'Air Astana' AND f.sch_departure_time  
7     BETWEEN now() AND now() + INTERVAL '7 days';
```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 93 msec.

Total rows: 0 Query complete: 00:00:00.000 CPU: 1.5% OS: 1%

✓ Query returned successfully in 93 msec. ✎

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

```
CREATE VIEW delayed_24h AS  
  
SELECT flight_id, sch_departure_time, act_departure_time,  
       act_departure_time - sch_departure_time FROM flights  
  
WHERE act_departure_time - sch_departure_time > INTERVAL '24 hours';
```

```

Query History
1. CREATE VIEW delayed_24h AS
2. SELECT flight_id, sch_departure_time, act_departure_time,
3.        act_departure_time - sch_departure_time FROM flights
4. WHERE act_departure_time - sch_departure_time > INTERVAL '24 hours';

Data Output Messages Notifications
CREATE VIEW

Query returned successfully in 94 msec.

Total rows: 0 Query complete 00:00:00.004 ✓

```

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.

CREATE VIEW leffler_passengers AS

```

SELECT p.first_name || ' ' || p.last_name as full_name, p.country_of_citizenship FROM
passengers p
JOIN tickets t on t.passenger_id = p.passenger_id
JOIN booking b on b.passenger_id = p.passenger_id
WHERE b.booking_platform = 'Leffler-Thompson';

```

```

Query History
1. CREATE VIEW leffler_passengers AS
2. SELECT p.first_name || ' ' || p.last_name as full_name,
3.        p.country_of_citizenship FROM passengers p
4. JOIN tickets t on t.passenger_id = p.passenger_id
5. JOIN booking b on b.passenger_id = p.passenger_id
6. WHERE b.booking_platform = 'Leffler-Thompson';

Data Output Messages Notifications
CREATE VIEW

Query returned successfully in 106 msec.

Total rows: 0 Query complete 00:00:00.106 ✓

```

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

```
CREATE VIEW top10_countries AS  
  
SELECT arr.country, COUNT(*) AS visits FROM booking_flight bf  
  
JOIN flights f ON bf.flight_id = f.flight_id  
  
JOIN airport arr ON arr.airport_id = f.arriving_airport_id  
  
GROUP BY arr.country  
  
ORDER BY visits DESC  
  
limit 10;
```

The screenshot shows a database query interface with two main sections: 'Query' and 'Messages'.

Query: This section contains the SQL code for creating a view named 'top10_countries'. The code is as follows:

```
1 ✓ CREATE VIEW top10_countries AS  
2   SELECT arr.country, COUNT(*) AS visits FROM booking_flight bf  
3   JOIN flights f ON bf.flight_id = f.flight_id  
4   JOIN airport arr ON arr.airport_id = f.arriving_airport_id  
5   GROUP BY arr.country  
6   ORDER BY visits DESC  
7   limit 10;
```

Messages: This section displays the results of the query execution. It shows the message "CREATE VIEW" followed by "Query returned successfully in 147 msec." A green success message box at the bottom right also states "Query returned successfully in 147 msec.".

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

```
CREATE OR REPLACE VIEW flights_on_date AS  
SELECT f.*, dep.city AS departure_city FROM flights f  
JOIN airport dep on dep.airport_id = f.departing_airport_id  
WHERE date(f.sch_departure_time) = '2025-01-15';
```

Query History

```
1 ✓ CREATE OR REPLACE VIEW flights_on_date AS
2   SELECT f.*, dep.city AS departure_city FROM flights f
3   JOIN airport dep ON dep.airport_id = f.departing_airport_id
4   WHERE date(f.sch_departure_time) = '2025-01-15';
```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 70 msec.

✓ Query returned successfully in 70 msec. ✘

RESULT:

Query History

1 SELECT * FROM flights_on_date

Data Output Messages Notifications

flight_id sch_departure_time sch_arrival_time departing_airport_id arriving_airport_id departing_gate arriving_gate airline_id

10. Drop all existing views.

The screenshot shows a database query interface with two main sections: 'Query' and 'Messages'. In the 'Query' section, a script of seven 'DROP VIEW' statements is listed. In the 'Messages' section, the output shows the command 'DROP VIEW' followed by the message 'Query returned successfully in 120 msec.' A green success message box at the bottom right also states 'Query returned successfully in 120 msec.' with a checkmark icon. The status bar at the bottom indicates 'Total rows: 0' and 'Query complete 00:00:00 120'.

```
1 DROP VIEW flights_on_date CASCADE;
2 DROP VIEW ext_week CASCADE;
3 DROP VIEW top5_routes CASCADE;
4 DROP VIEW flights_by_airline CASCADE;
5 DROP VIEW delayed_24h CASCADE;
6 DROP VIEW leffler_passengers CASCADE;
7 DROP VIEW top10_countries CASCADE;
```

Data Output Messages Notifications

DROP VIEW

Query returned successfully in 120 msec.

Total rows: 0 Query complete 00:00:00 120 ✓ CRI E In 2 Col 11