

LAB 8

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

The screenshot shows a database console interface with a left sidebar displaying a database explorer. The main area contains SQL code for creating a view and querying it. The view is named `flight_on_date` and is created using a hash partition on the `country` column. The view selects flight details for a specific date. The output shows a single row of flight data.

```
on airport using hash (country);
create view flight_on_date as
select flight_id,
       sch_departure_time,
       sch_arrival_time,
       arriving_airport_id,
       departing_airport_id,
       act_departure_time,
       act_arrival_time,
       flight_no,
       airline_id
from flights;
select * from flight_on_date
where date(act_departure_time) = '2005-08-10';
```

flight_id	sch_departure_time	sch_arrival_time	arriving_airport_id	departing_airport_id	act_departure_time	act_arrival_time
1	23 2006-04-23 05:02:55.000000	2013-03-04 02:51:38.000000	187	68	2005-08-10 19:59:13.000000	2004-04

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

The screenshot shows a database console interface with a left sidebar displaying a database explorer. The main area contains SQL code for creating a view and querying it. The view is named `bookings_by_flight` and is created by joining the `booking` and `flights` tables. The view selects booking details for flights scheduled to depart within the next week. The output shows 0 rows.

```
create view bookings_by_flight as
select b.booking_id,
       f.flight_id,
       f.sch_departure_time,
       b.passenger_id,
       b.booking_platform,
       b.status,
       b.ticket_price,
       b.ticket_discount,
       b.created_at as booking_creation_time
from booking b join flights f
on b.flight_id = f.flight_id
where f.sch_departure_time >= now()
and f.sch_departure_time <= now() + interval '7 days';
select * from bookings_by_flight;
```

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

The screenshot shows a database console interface with a SQL query editor and an output window. The query editor contains the following SQL code:

```
521
522
523 create view top5_routes as
524     select f.departing_airport_id,
525            f.arriving_airport_id,
526            count(b.booking_id) as total_bookings
527 from flights f join booking b 1<->1 on b.flight_id = f.flight_id
528 group by f.departing_airport_id, f.arriving_airport_id
529 order by total_bookings desc
530 limit 5;
531
532 select * from top5_routes;
```

The output window displays the results of the query, showing 5 rows of data:

	departing_airport_id	arriving_airport_id	total_bookings
1	168	106	7
2	181	48	6
3	28	118	6
4	193	174	6
5	77	146	5

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

The screenshot shows a database console interface with a SQL query editor and an output window. The query editor contains the following SQL code:

```
534
535 create view airline_flights as
536     select airline_id,
537            flight_id,
538            departing_airport_id,
539            arriving_airport_id,
540            act_arrival_time,
541            act_departure_time,
542            sch_arrival_time,
543            sch_departure_time
544 from flights;
545
546 select * from airline_flights
547 where airline_id = '75';
```

The output window displays the results of the query, showing 1 row of data:

airline_id	flight_id	departing_airport_id	arriving_airport_id	act_arrival_time	act_departure_time	sch_arrival_time
75	163	17	127	2020-07-25 10:44:51.000000	2005-12-26 15:01:20.000000	2017-01-17 02:2

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 a database console interface with a query editor and a results pane. The query editor contains the following SQL code:

```
create or replace view airline_flights as
select
    airline_id,
    flight_id,
    departing_airport_id,
    arriving_airport_id,
    act_arrival_time,
    act_departure_time,
    sch_arrival_time,
    sch_departure_time
from flights where airline_id = 75
and sch_departure_time >= now()
and sch_departure_time <= now() + interval '7 days';
select * from airline_flights;
```

The results pane shows the output of the query, which is currently empty (0 rows).

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

The screenshot shows a database console interface with a query editor and a results pane. The query editor contains the following SQL code:

```
create view delayed_flights as
select
    flight_id,
    airline_id,
    departing_airport_id,
    arriving_airport_id,
    sch_arrival_time,
    act_arrival_time,
    act_arrival_time - sch_arrival_time as delay_period
from flights
where act_arrival_time - sch_arrival_time > interval '24 hours';
select* from delayed_flights;
```

The results pane shows the output of the query, which is a table with 10 rows and 7 columns. The columns are: flight_id, airline_id, departing_airport_id, arriving_airport_id, sch_arrival_time, act_arrival_time, and delay_period. The data is as follows:

flight_id	airline_id	departing_airport_id	arriving_airport_id	sch_arrival_time	act_arrival_time	delay_period
1	4	195	165	199 2002-03-30 22:46:52.000000	2014-10-20 21:51:39.000000	0 years 0 mons
2	7	159	67	102 2003-04-12 18:50:18.000000	2014-04-12 16:38:56.000000	0 years 0 mons
3	9	113	138	84 2012-03-23 17:59:28.000000	2016-04-16 17:26:11.000000	0 years 0 mons
4	13	125	196	42 2005-12-21 12:07:13.000000	2023-12-04 17:32:12.000000	0 years 0 mons
5	15	5	177	25 2005-09-24 08:30:03.000000	2014-12-17 21:10:33.000000	0 years 0 mons
6	20	144	186	56 2008-06-22 04:20:38.000000	2010-11-08 18:47:08.000000	0 years 0 mons
7	21	42	89	51 2008-03-19 04:11:05.000000	2017-07-22 18:55:25.000000	0 years 0 mons
8	24	82	118	137 2001-01-30 18:04:22.000000	2008-09-13 11:19:20.000000	0 years 0 mons
9	28	43	46	3 2013-10-15 15:30:30.000000	2025-05-19 02:51:42.000000	0 years 0 mons
10	29	94	193	174 2012-12-11 07:09:58.000000	2023-04-21 04:48:30.000000	0 years 0 mons

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 database console interface with a SQL editor and an output window. The SQL editor contains the following code:

```
585
586 create view leffler_t_passengers as
587     select b.booking_platform,
588           p.passenger_id,
589           p.first_name,
590           p.last_name,
591           p.country_of_citizenship
592 from booking b join passengers p 1.n<>1: on b.passenger_id = p.passenger_id
593 where b.booking_platform = 'Leffler Thompson';
594
595 select * from leffler_t_passengers;
```

The output window displays the results of the query, showing a single row of data:

booking_platform	passenger_id	first_name	last_name	country_of_citizenship
Leffler Thompson	454	Laura	John	Bulgaria

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

The screenshot shows a database console interface with a SQL editor and an output window. The SQL editor contains the following code:

```
597
598 create or replace view most_visited as
599     select a.country as visited_country,
600           count(b.booking_id) as total_visits
601 from flights f join booking b 1.<>1.n: on b.flight_id = f.flight_id
602 join airport a 1.n<>1: on f.arriving_airport_id = a.airport_id
603 group by a.country
604 order by total_visits desc
605 limit 10;
606
607 select * from most_visited;
```

The output window displays the results of the query, showing the top 10 most visited countries:

visited_country	total_visits
1 China	60
2 Indonesia	57
3 Philippines	28
4 Russia	27
5 Poland	20
6 Argentina	15
7 Thailand	14
8 United States	13
9 Czech Republic	12
10 Slovenia	9

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

The screenshot shows a database console interface with a SQL query executed. The query creates or replaces a view named 'most_visited' and then selects all data from it. The results are displayed in a table with three columns: 'visited_country', 'total_visits', and 'airline_name'.

```
create or replace view most_visited as
select a.country as visited_country,
       count(b.booking_id) as total_visits,
       al.airline_name
from flights f join booking b on b.flight_id = f.flight_id
join airport a on a.arriving_airport_id = a.airport_id
join airline al on al.airline_id = a.airline_id
group by a.country, al.airline_name
order by total_visits desc
limit 10;

select * from most_visited;
```

visited_country	total_visits	airline_name
Indonesia	11	United Aerolines
Indonesia	8	British Aeroways
China	7	South African Aeroways
Indonesia	7	South African Aeroways
Argentina	7	Ethiopian Aerolines
Poland	6	Avianca
China	6	SAS Scandinavian
China	6	Avianca
Slovenia	6	Aero Canada
Indonesia	5	KLM Royal Dutch Aer

10. Drop all existing views.

The screenshot shows a database console interface with a SQL query that lists and drops all existing views in the 'public' schema. The results show an empty table with columns 'schemaname' and 'viewname'.

```
select schemaname, viewname
from pg_views
where schemaname = 'public';

drop view airline_flights;
drop view flight_on_date;
drop view delayed_flights;
drop view bookings_by_flight;
drop view leffler_t_passengers;
drop view top5_routes;
drop view most_visited;
```

schemaname	viewname
------------	----------