

LAB 8

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 'db_international_airports' database selected. In the Database Explorer, a new view named 'flight_on_date' is being created. The code for the view is:

```
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;
```

Below the code, a query is run to select all rows from the view where the departure date is '2005-08-10':

```
select * from flight_on_date
where date(act_departure_time) = '2005-08-10';
```

The results of this query are displayed in the Services tab, showing one row of flight information for that specific date.

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

The screenshot shows the DBeaver interface with the 'db_international_airports' database selected. A new view named 'bookings_by_flight' is being created. The code for the view is:

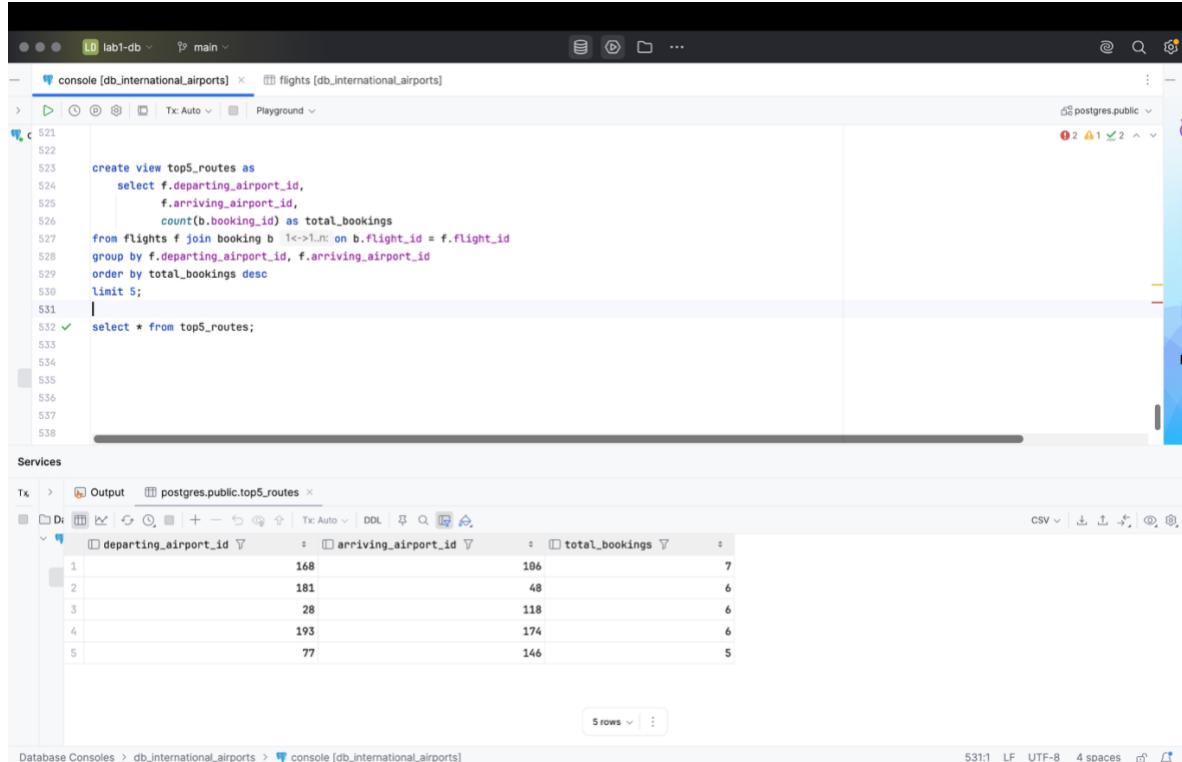
```
create view bookings_by_flight as
select b.booking_id,
       f.flight_id,
       f.sch_departure_time,
       b.passenger_id,
       b.booking_platform,
       b.status,
       b.ticket_price,
       b.ticket_discount,
       b.created_at as booking_creation_time
  from booking b join flights f 1..n->1.. on b.flight_id = f.flight_id
 where f.sch_departure_time >= now()
   and f.sch_departure_time <= now() + interval '7 days';
```

Below the code, a query is run to select all rows from the view:

```
select * from bookings_by_flight;
```

The results of this query are displayed in the Services tab, showing multiple rows of booking information for flights departing within the next week.

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



```

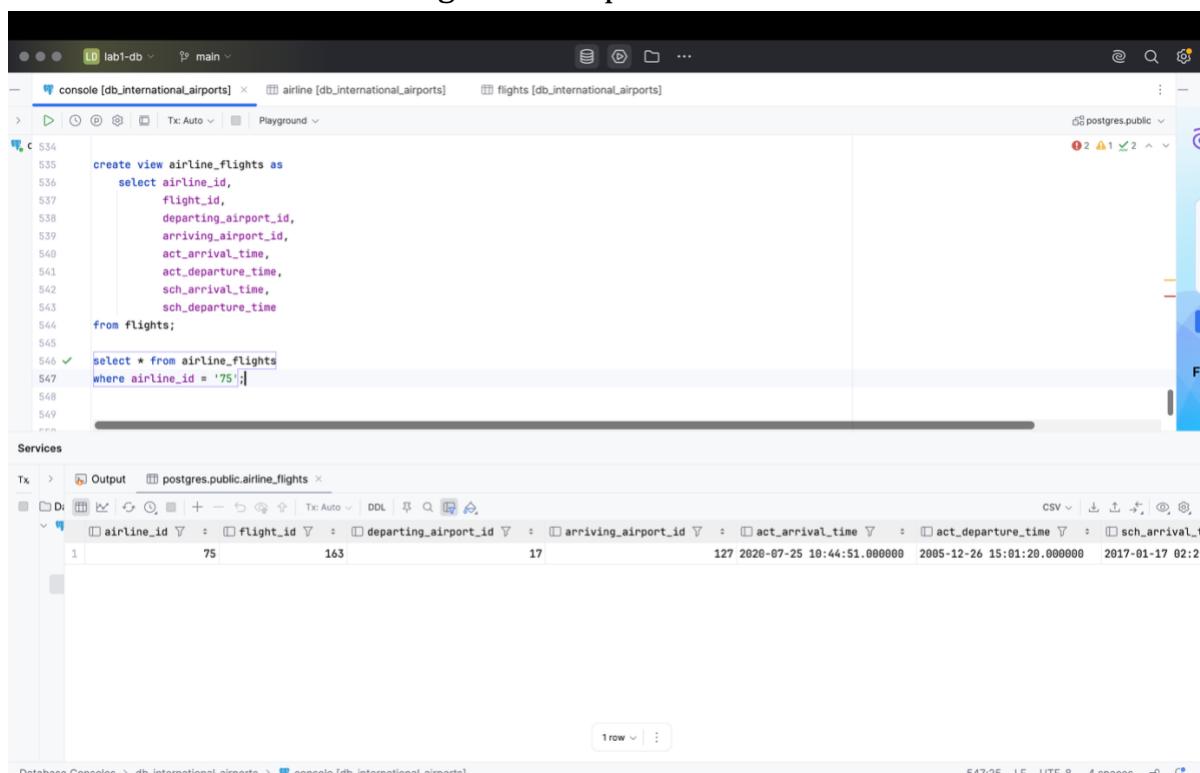
521
522
523 create view top5_routes as
524     select f.departing_airport_id,
525             f.arriving_airport_id,
526             count(b.booking_id) as total_bookings
527     from flights f join booking b 1->1.m on b.flight_id = f.flight_id
528     group by f.departing_airport_id, f.arriving_airport_id
529     order by total_bookings desc
530     limit 5;
531
532 ✓ select * from top5_routes;
533
534
535
536
537
538

```

The screenshot shows a PostgreSQL database console interface. The code in the main pane creates a view named 'top5_routes' that selects the top 5 flight routes based on the number of bookings. The output pane displays the results of the query, showing five rows of data:

	departing_airport_id	arriving_airport_id	total_bookings
1	168	186	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



```

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';
548
549

```

The screenshot shows a PostgreSQL database console interface. The code in the main pane creates a view named 'airline_flights' that lists all flights for a specific airline (airline_id 75). The output pane displays the results of the query, showing one row of data:

airline_id	flight_id	departing_airport_id	arriving_airport_id	act_arrival_time	act_departure_time	sch_arrival_time	sch_departure_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 the pgAdmin interface with a code editor and a results pane. The code editor contains the following SQL:

```

554
555 create or replace view airline_flights as
556     select
557         airline_id,
558         flight_id,
559         departing_airport_id,
560         arriving_airport_id,
561         act_arrival_time,
562         act_departure_time,
563         sch_arrival_time,
564         sch_departure_time
565     from flights where airline_id = 75
566     and sch_departure_time >= now()
567     and sch_departure_time <= now() + interval '7 days';
568
569 ✓    select * from airline_flights;
570
571

```

The results pane shows the structure of the view:

airline_id	flight_id	departing_airport_id	arriving_airport_id	act_arrival_time	act_departure_time	sch_arrival_time

Below the results pane, it says "0 rows".

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

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

```

570
571 create view delayed_flights as
572     select
573         flight_id,
574         airline_id,
575         departing_airport_id,
576         arriving_airport_id,
577         sch_arrival_time,
578         act_arrival_time,
579         act_arrival_time - sch_arrival_time as delay_period
580     from flights
581     where act_arrival_time - sch_arrival_time > interval '24 hours';
582
583 ✓    select* from delayed_flights;
584
585
586

```

The results pane shows the structure of the view:

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	2014-04-20 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	21 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 11:19:26.000000	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

Below the results pane, it says "188 rows".

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.

```

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;
596
597
598
599
600
601
602
603
604

```

The screenshot shows a PostgreSQL database interface. In the top navigation bar, the database is set to 'lab1-db'. The current tab is 'main'. Below the tabs, there are several open panes: 'console [db_international_airports]', 'passengers [db_international_airports]', 'booking [db_international_airports]', 'airline [db_international_airports]', and 'flights [db_international_airport]'. The 'Tx' tab is selected in the bottom left. The main pane displays the SQL code for creating a view named 'leffler_t_passengers'. The code uses a join between the 'booking' and 'passenger' tables to filter passengers from the 'Leffler Thompson' platform. The 'Services' pane shows the newly created view 'postres.public.leffler_t_passengers'. A table viewer below it displays one row of data: 'Leffler Thompson' (passenger_id), '454' (first_name), 'Laura' (last_name), and 'Bulgaria' (country_of_citizenship). The bottom status bar indicates the session ID '595:36', file type 'LF', encoding 'UTF-8', and a note about '4 spaces'.

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

```

577
578     create or replace view most_visited as
579         select a.country as visited_country,
580             count(b.booking_id) as total_visits
581         from flights f join booking b 1->1..n: on b.flight_id = f.flight_id
582         join airport a 1..n->1: on f.arriving_airport_id = a.airport_id
583         group by a.country
584         order by total_visits desc
585         limit 10;
586
587 ✓    select * from most_visited;
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610

```

The screenshot shows a PostgreSQL database interface. The setup is identical to the previous one, with 'lab1-db' as the database and 'main' as the tab. The 'Tx' tab is selected. The main pane contains the SQL code for creating a view named 'most_visited'. This view aggregates data from the 'flights', 'booking', and 'airport' tables to count the total visits per country, ordered by the total visits in descending order, with a limit of 10 rows. The 'Services' pane shows the view 'postres.public.most_visited'. A table viewer below it lists the top 10 most visited countries with their respective visit counts. The table has two columns: 'visited_country' and 'total_visits'. The data is as follows:

visited_country	total_visits
China	60
Indonesia	57
Philippines	28
Russia	27
Poland	20
Argentina	15
Thailand	14
United States	13
Czech Republic	12
Slovenia	9

The bottom status bar shows the session ID '608:1', file type 'LF', encoding 'UTF-8', and a note about '4 spaces'.

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

The screenshot shows the pgAdmin interface with the database 'db_international_airports' selected. In the top-left pane, a code editor displays the SQL command to create a view:

```

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             al.airline_name
602         from flights f join booking b 1<->1:n on b.flight_id = f.flight_id
603         join airport a 1..n->1:n on f.arriving_airport_id = a.airport_id
604         join airline al 1..n->1:n on f.airline_id = al.airline_id
605         group by a.country, al.airline_name
606         order by total_visits desc
607         limit 10;
608
609 ✓     select * from most_visited;
610
611
612

```

In the bottom-right pane, the 'Output' tab shows the results of the query:

visited_country	total_visits	airline_name
Indonesia	11	United Aeroline
Indonesia	8	British Aeroways
China	7	South African Aeroways
Indonesia	7	South African Aeroways
Argentina	7	Ethiopian Aeroline
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 the pgAdmin interface with the database 'db_international_airports' selected. In the top-left pane, a code editor displays a script to drop all views in the 'public' schema:

```

612
613     select schemaname, viewname
614     from pg_views
615     where schemaname = 'public';
616
617     drop view airline_flights;
618     drop view flight_on_date;
619     drop view delayed_flights;
620     drop view bookings_by_flight;
621     drop view leffler_t_passengers;
622     drop view top5_routes;
623
624
625
626
627
628
629

```

In the bottom-right pane, the 'Output' tab shows the results of the query:

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

At the bottom of the interface, the status bar indicates 'Database Consoles > db_international_airports > console [db_international_airports]' and '623:1 LF UTF-8 4 spaces'.