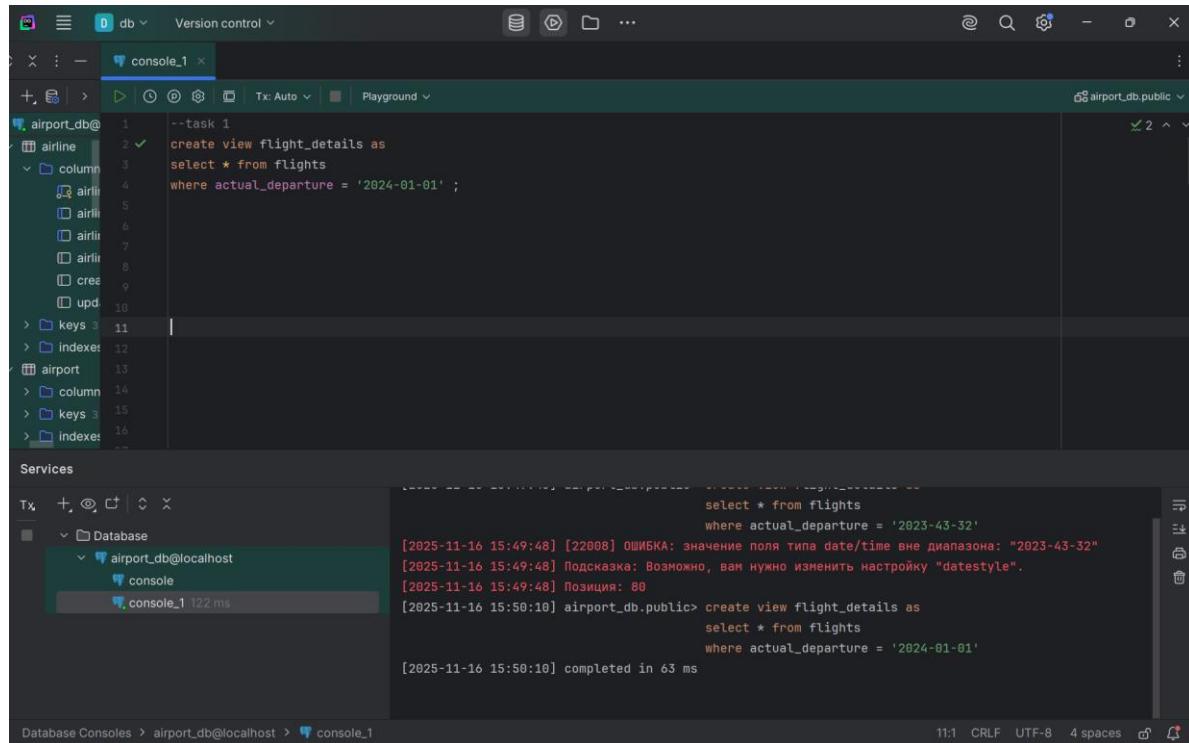


Laboratory work 8

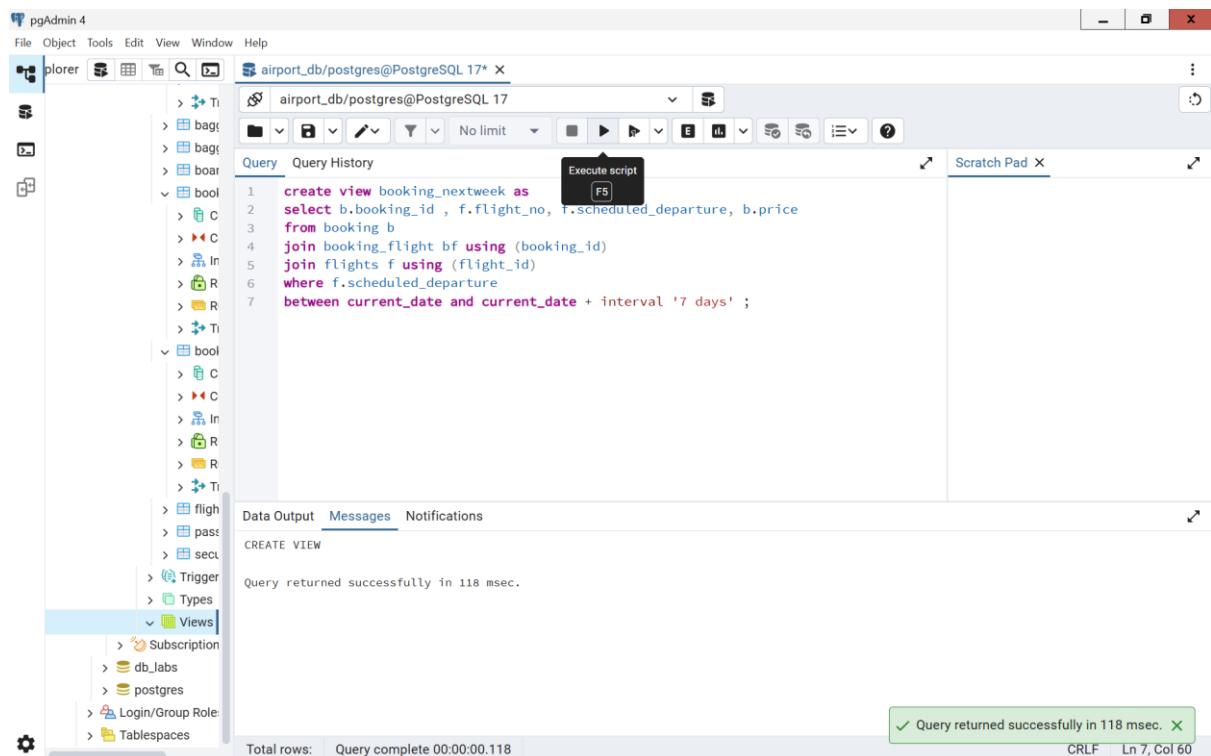
1. Create a view to show details of all flights that are departing on a specific date.



```
--task 1
1  create view flight_details as
2  select * from flights
3  where actual_departure = '2024-01-01' ;
```

The screenshot shows a PostgreSQL database console window. The top bar indicates the connection is to 'airport_db.public'. The left sidebar shows the schema structure under 'airport_db@public'. The main area contains a code editor with the above SQL command. Below the code editor is a 'Services' section showing a transaction 'Tx' and a database connection 'Database' to 'airport_db@localhost'. The right side of the window displays the execution results of the query, including a warning about the 'datestyle' setting and the successful creation of the view. The bottom status bar shows the command was completed in 63 ms.

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



```
create view booking_nextweek as
select b.booking_id , f.flight_no, f.scheduled_departure, b.price
from booking b
join booking_flight bf using (booking_id)
join flights f using (flight_id)
where f.scheduled_departure
between current_date and current_date + interval '7 days' ;
```

The screenshot shows the pgAdmin 4 interface. The left sidebar shows the database structure for 'airport_db/postgres@PostgreSQL 17'. The main area has a 'Query' tab open with the above SQL code. The 'Messages' tab at the bottom shows a green message indicating the query was executed successfully in 118 msec. The bottom status bar shows the total rows affected and the completion time.

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

The screenshot shows the pgAdmin 4 interface with the 'airport_db/postgres@PostgreSQL 17*' connection selected. The left sidebar displays the database schema with 'Views' highlighted. The main query editor window contains the following SQL code:

```
1 create view most_popular as
2 select f.departure_airport_id, f.arrival_airport_id, count(bf.booking_id) as total
3 from booking_flight bf
4 join flights f on bf.flight_id = f.flight_id
5 group by f.departure_airport_id, f.arrival_airport_id
6 order by total desc
7 limit 5;
```

The 'Messages' tab below the query editor shows the message: 'CREATE VIEW'. The status bar at the bottom indicates 'Query returned successfully in 102 msec.' and 'Total rows: Query complete 00:00:00.102'.

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

The screenshot shows the pgAdmin 4 interface with the same connection. The left sidebar shows 'Views' highlighted. The main query editor window contains the following SQL code:

```
1 create view spec_airline as
2 select f.flight_no, a.airline_name
3 from flights f join airline a on f.airline_id = a.airline_id
4 where a.airline_name = 'ICP';
```

The 'Messages' tab shows 'CREATE VIEW'. The status bar at the bottom indicates 'Query returned successfully in 117 msec.' and 'Total rows: Query complete 00:00:00.117'.

5. Modify the view created in task 4 to show only flights departing within the next 7 days for a specific airline.

pgAdmin 4

File Object Tools Edit View Window Help

Explorer Query airport_db/postgres@PostgreSQL 17*

```

1 create or replace view spec_airline as
2 select f.flight_no, a.airline_name
3 from flights f
4 join airline a on f.airline_id = a.airline_id
5 where a.airline_name = 'ICP'
6 and f.scheduled_departure between current_date and current_date + interval '7 days';

```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 100 msec.

Total rows: Query complete 00:00:00.100 CRLF Ln 6, Col 27

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

pgAdmin 4

File Object Tools Edit View Window Help

Explorer Query airport_db/postgres@PostgreSQL 17*

```

1 create view delayed as
2 select flight_id , scheduled_departure, actual_departure,age(actual_departure,scheduled_
3 from flights
4 where extract(day from age(actual_departure,scheduled_departure)) >= 1 ;

```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 200 msec.

Total rows: Query complete 00:00:00.200 CRLF Ln 4, Col 73

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.

pgAdmin 4

File Object Tools Edit View Window Help

Explorer Query airport_db/postgres@PostgreSQL 17*

Query History

```
1 create view bookingplatform as
2 select concat(p.first_name, ' ', p.last_name) as fullname , p.country_of_citizenship
3 from passengers p
4 join booking b using(passenger_id)
5 where booking_platform = 'Leffler-Thompson platform' ;
```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 148 msec.

Total rows: 0 Query complete 00:00:00.148 CRLF Ln 4, Col 1

pgAdmin 4

File Object Tools Edit View Window Help

Explorer Query airport_db/postgres@PostgreSQL 17*

Query History

```
1 create view bookingplatform as
2 select concat(p.first_name, ' ', p.last_name) as fullname , p.country_of_citizenship
3 from passengers p
4 join booking b using(passenger_id)
5 where booking_platform = 'Leffler-Thompson platform' ;
6
7 select * from bookingplatform ;
```

Data Output Messages Notifications

fullname	country_of_citizenship

Total rows: 0 Query complete 00:00:00.254 CRLF Ln 7, Col 1

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

pgAdmin 4

File Object Tools Edit View Window Help

Explorer Query History Scratch Pad

airport_db/postgres@PostgreSQL 17*

```

1 create view top10 as
2 select a.country, count(bf.booking_id) as total
3 from flights f
4 join airport a on f.arrival_airport_id = a.airport_id
5 join booking_flight bf using(flight_id)
6 group by a.country
7 order by total desc limit 10 ;

```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 133 msec.

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

pgAdmin 4

File Object Tools Edit View Window Help

Explorer Query History Scratch Pad

airport_db/postgres@PostgreSQL 17*

```

1 create or replace view top10 as
2 select a.country, count(bf.booking_id) as total, now() as update
3 from flights f
4 join airport a on f.arrival_airport_id = a.airport_id
5 join booking_flight bf using(flight_id)
6 group by a.country
7 order by total desc limit 10 ;

```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 101 msec.

Total rows: Query complete 00:00:00.101

✓ Query returned successfully in 101 msec. X

CRLF Ln 7, Col 31

10. Drop all existing views.

pgAdmin 4

File Object Tools Edit View Window Help

Explorer airport_db/postgres@PostgreSQL 17*

Query History

```
1 drop view if exists flight_details, booking_nextweek, most_popular,bookingplatform,
2 spec_airline,delayed, top10 ;
```

Data Output Messages Notifications

DROP VIEW

Query returned successfully in 154 msec.

Total rows: Query complete 00:00:00.154

✓ Query returned successfully in 154 msec. X

CRLF Ln 2, Col 1

A screenshot of the pgAdmin 4 interface. The left sidebar shows a tree view of database objects under 'airport_db/postgres@PostgreSQL 17*'. The 'Views' node is selected. The main pane contains a query editor with two lines of SQL: 'drop view if exists flight_details, booking_nextweek, most_popular,bookingplatform, spec_airline,delayed, top10 ;'. Below the editor is a 'Data Output' tab showing the result of the query: 'DROP VIEW' followed by the message 'Query returned successfully in 154 msec.'. A green status bar at the bottom right indicates 'Query returned successfully in 154 msec.' with a checkmark icon. The bottom status bar also shows 'CRLF' and 'Ln 2, Col 1'.