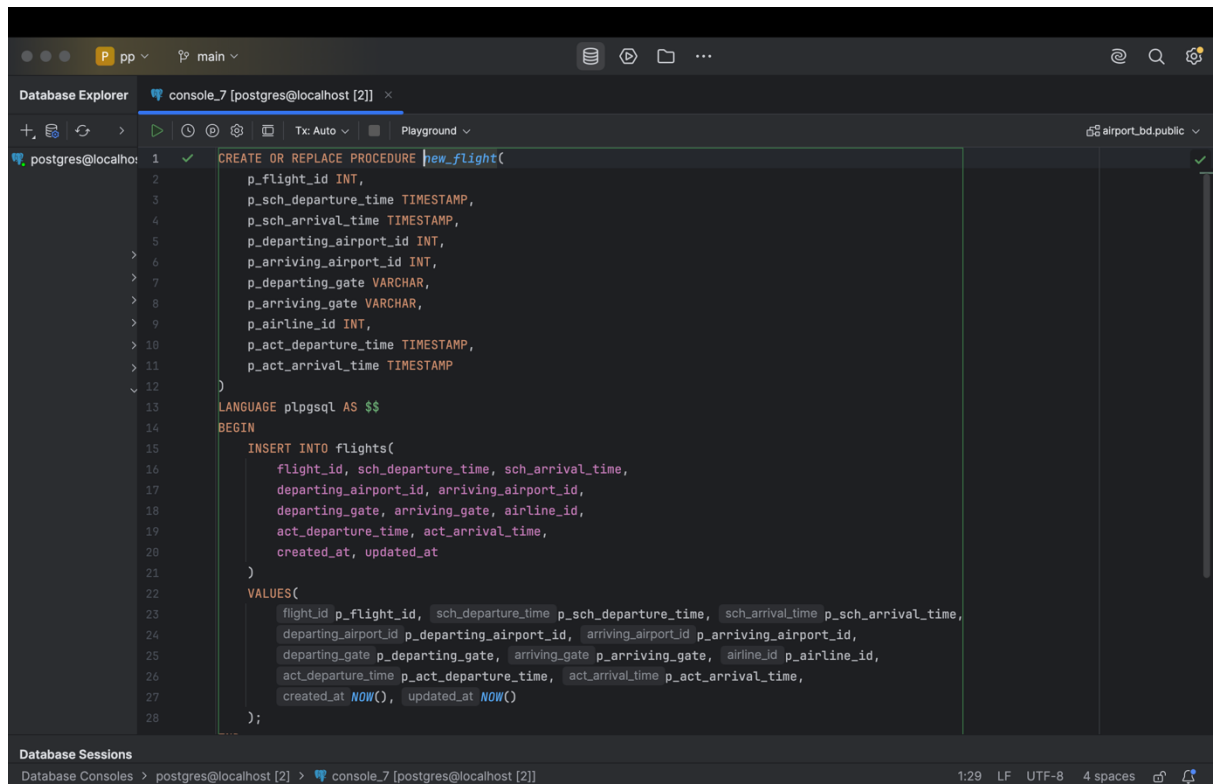


Laboratory work 10

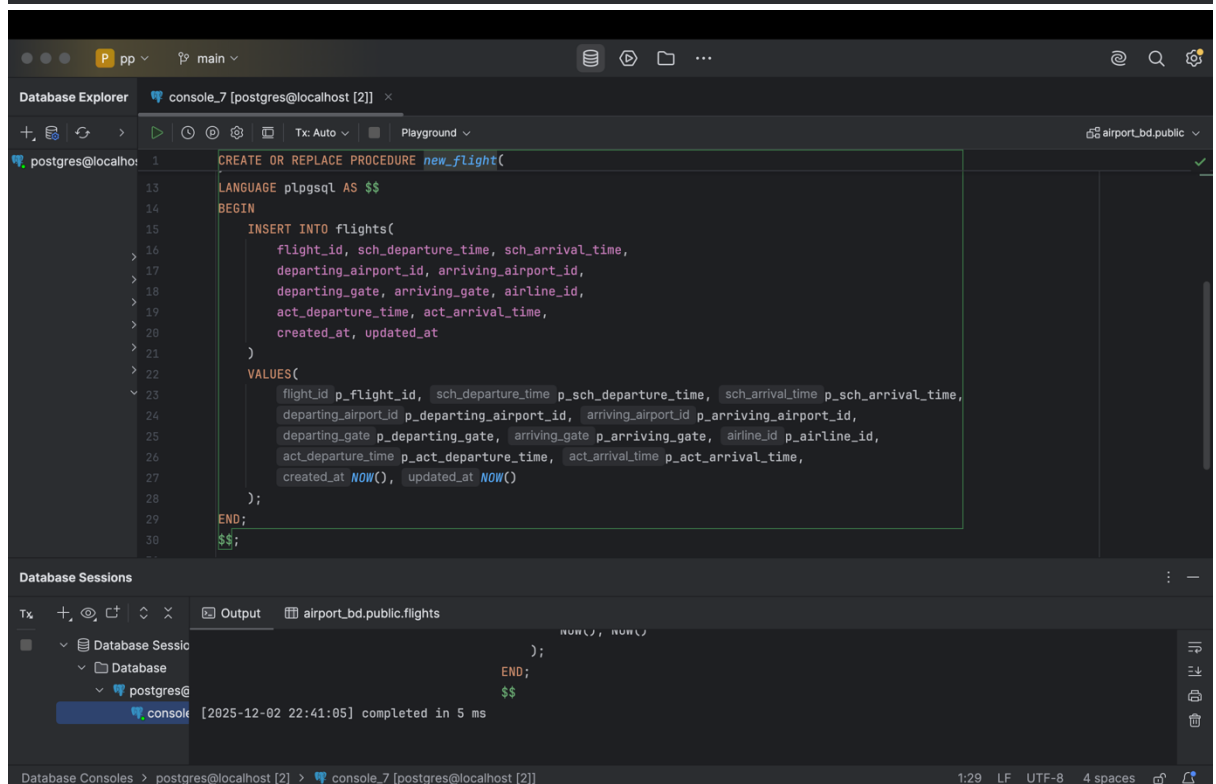
1.



The screenshot shows a PostgreSQL console window with the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE new_flight(  
2     p_flight_id INT,  
3     p_sch_departure_time TIMESTAMP,  
4     p_sch_arrival_time TIMESTAMP,  
5     p_departing_airport_id INT,  
6     p_arriving_airport_id INT,  
7     p_departing_gate VARCHAR,  
8     p_arriving_gate VARCHAR,  
9     p_airline_id INT,  
10    p_act_departure_time TIMESTAMP,  
11    p_act_arrival_time TIMESTAMP  
12 )  
13 LANGUAGE plpgsql AS $$  
14 BEGIN  
15     INSERT INTO flights(  
16         flight_id, sch_departure_time, sch_arrival_time,  
17         departing_airport_id, arriving_airport_id,  
18         departing_gate, arriving_gate, airline_id,  
19         act_departure_time, act_arrival_time,  
20         created_at, updated_at  
21     )  
22     VALUES(  
23         flight_id p_flight_id, sch_departure_time p_sch_departure_time, sch_arrival_time p_sch_arrival_time,  
24         departing_airport_id p_departing_airport_id, arriving_airport_id p_arriving_airport_id,  
25         departing_gate p_departing_gate, arriving_gate p_arriving_gate, airline_id p_airline_id,  
26         act_departure_time p_act_departure_time, act_arrival_time p_act_arrival_time,  
27         created_at NOW(), updated_at NOW()  
28     );  
29 END;  
30 $$;
```

The console also shows the Database Explorer on the left and the Database Sessions panel at the bottom.



The screenshot shows the same PostgreSQL console window, but with the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE new_flight(  
2     p_flight_id INT,  
3     p_sch_departure_time TIMESTAMP,  
4     p_sch_arrival_time TIMESTAMP,  
5     p_departing_airport_id INT,  
6     p_arriving_airport_id INT,  
7     p_departing_gate VARCHAR,  
8     p_arriving_gate VARCHAR,  
9     p_airline_id INT,  
10    p_act_departure_time TIMESTAMP,  
11    p_act_arrival_time TIMESTAMP  
12 )  
13 LANGUAGE plpgsql AS $$  
14 BEGIN  
15     INSERT INTO flights(  
16         flight_id, sch_departure_time, sch_arrival_time,  
17         departing_airport_id, arriving_airport_id,  
18         departing_gate, arriving_gate, airline_id,  
19         act_departure_time, act_arrival_time,  
20         created_at, updated_at  
21     )  
22     VALUES(  
23         flight_id p_flight_id, sch_departure_time p_sch_departure_time, sch_arrival_time p_sch_arrival_time,  
24         departing_airport_id p_departing_airport_id, arriving_airport_id p_arriving_airport_id,  
25         departing_gate p_departing_gate, arriving_gate p_arriving_gate, airline_id p_airline_id,  
26         act_departure_time p_act_departure_time, act_arrival_time p_act_arrival_time,  
27         created_at NOW(), updated_at NOW()  
28     );  
29 END;  
30 $$;
```

The console also shows the Database Explorer on the left and the Database Sessions panel at the bottom. The Database Sessions panel shows a session named 'console_7' with the status 'completed in 5 ms'.

2.

The screenshot shows a PostgreSQL IDE interface. The top pane displays the SQL code for creating a procedure named `fly_airport`. The code is as follows:

```

1 CREATE OR REPLACE PROCEDURE fly_airport(
2   p_airport_id INT
3 )
4 LANGUAGE plpgsql AS $$
5 BEGIN
6   SELECT * FROM flights WHERE departing_airport_id = p_airport_id;
7 END;
8 $$;
9

```

The bottom pane shows the Database Sessions tab with the output of the procedure execution. The output indicates that the procedure was completed successfully in 5 ms.

```

[2025-12-02 22:41:05] completed in 5 ms
[2025-12-02 22:46:07] airport_bd.public> CREATE OR REPLACE PROCEDURE fly_airport(
    p_airport_id INT
  )
  LANGUAGE plpgsql AS $$
  BEGIN
    SELECT * FROM flights WHERE departing_airport_id = p_airport_id;
  END;
  $$;
[2025-12-02 22:46:07] completed in 4 ms

```

3.

The screenshot shows a PostgreSQL IDE interface. The top pane displays the SQL code for creating a function named `avg_arrival_delay_time`. The code is as follows:

```

1 CREATE OR REPLACE FUNCTION avg_arrival_delay_time(
2   p_airport_id INT
3 )
4 RETURNS NUMERIC
5 LANGUAGE plpgsql
6 AS $$
7 DECLARE
8   avg_delay NUMERIC;
9 BEGIN
10  SELECT AVG(EXTRACT(EPOCH FROM (act_arrival_time - sch_arrival_time)) / 60)
11    INTO avg_delay
12  FROM flights
13  WHERE arriving_airport_id = p_airport_id;
14
15  RETURN avg_delay;
16 END;
17 $$;
18
19

```

The bottom pane shows the Database Sessions tab with the output of the function execution. The output indicates that the function was completed successfully in 4 ms.

```

[2025-12-02 22:54:01] completed in 4 ms

```

4.

The screenshot shows a PostgreSQL IDE interface. On the left, the 'Database Explorer' pane displays the schema for 'airport_bd.public', including tables like 'passengers', 'bookings', and 'flights'. The main editor pane shows a SQL script to create a procedure named 'pass_flight'. The script is as follows:

```

1 CREATE OR REPLACE PROCEDURE pass_flight(
2   p_flight_id INT
3 )
4 LANGUAGE plpgsql
5 AS $$
6 BEGIN
7   SELECT p.* FROM passengers p
8   JOIN booking b ON p.passenger_id=b.passenger_id
9   WHERE b.flight_id=p_flight_id;
10 END;
11 $$;

```

The 'Database Sessions' pane at the bottom shows the execution of the script, which completed successfully in 6 ms.

5.

The screenshot shows the same PostgreSQL IDE interface. The main editor pane now shows a SQL script to create a procedure named 'top_passenger'. The script is as follows:

```

1 CREATE OR REPLACE PROCEDURE top_passenger()
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5   SELECT p.passenger_id,
6         p.first_name,
7         p.last_name,
8         COUNT(*) AS total_flights
9   FROM passengers p
10  JOIN booking b ON p.passenger_id = b.passenger_id
11  GROUP BY p.passenger_id
12  ORDER BY total_flights DESC
13  LIMIT 1;
14 END;
15 $$;

```

The 'Database Sessions' pane at the bottom shows the execution of the script, which completed successfully in 5 ms.

6.

The screenshot shows a PostgreSQL IDE interface. On the left, the Database Explorer displays the schema of the `airport_bd` database, including the `booking_flight` table with columns `booking_flight_id`, `booking_id`, `flight_id`, `created_at`, and `updated_at`. The main editor shows the SQL code for creating a procedure `delayed_24h()` in PL/pgSQL. The procedure selects all flights from the `flights` table where the difference between actual and scheduled departure or arrival times is greater than 24 hours. The Database Sessions panel at the bottom shows the execution of the procedure, which completed in 4 ms.

```

1 CREATE OR REPLACE PROCEDURE delayed_24h()
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5     SELECT *
6     FROM flights
7     WHERE (act_departure_time - sch_departure_time) > INTERVAL '24 hours'
8           OR (act_arrival_time - sch_arrival_time) > INTERVAL '24 hours';
9 END;
10 $$;

```

```

BEGIN
SELECT *
FROM flights
WHERE (act_departure_time - sch_departure_time) > INTERVAL '24 hours'
      OR (act_arrival_time - sch_arrival_time) > INTERVAL '24 hours';
END;
$$
[2025-12-02 23:03:23] completed in 4 ms

```

7.

The screenshot shows a PostgreSQL IDE interface. On the left, the Database Explorer displays the schema of the `airport_bd` database, including the `booking_flight` table with columns `booking_flight_id`, `booking_id`, `flight_id`, `created_at`, and `updated_at`. The main editor shows the SQL code for creating a function `flight_airline()` in PL/pgSQL. The function takes an `airline_id` as input and returns the count of flights for that airline. The Database Sessions panel at the bottom shows the execution of the function, which completed in 4 ms.

```

1 CREATE OR REPLACE FUNCTION flight_airline(
2     p_airline_id INT
3 )
4 RETURNS INT
5 LANGUAGE plpgsql
6 AS $$
7 DECLARE
8     result INT;
9 BEGIN
10     SELECT COUNT(*) INTO result FROM flights WHERE airline_id = p_airline_id;
11     RETURN result;
12 END;
13 $$;

```

```

DECLARE
result INT;
BEGIN
SELECT COUNT(*) INTO result FROM flights WHERE airline_id = p_airline_id;
RETURN result;
END;
$$
[2025-12-02 23:05:01] completed in 4 ms

```

8.

The screenshot shows a PostgreSQL IDE interface. On the left, the 'Database Explorer' pane displays the schema for the 'airport_bd' database, including tables like 'booking_flight' and 'booking'. The main editor pane shows the SQL code for creating a stored procedure named 'avg_ticket_price'. The procedure takes a flight ID as input and returns the average ticket price for that flight. The 'Database Sessions' pane at the bottom shows the execution of the procedure, which completed successfully in 4 ms.

```

1 CREATE OR REPLACE PROCEDURE avg_ticket_price(
2   p_flight_id INT
3 )
4 LANGUAGE plpgsql
5 AS $$
6 BEGIN
7   SELECT AVG(ticket_price) AS average_price FROM booking
8   WHERE flight_id = p_flight_id;
9 END;
10 $$;

```

```

1 LANGUAGE plpgsql
2 AS $$
3 BEGIN
4   SELECT AVG(ticket_price) AS average_price FROM booking
5   WHERE flight_id = p_flight_id;
6 END;
7 $$

```

[2025-12-02 23:05:57] completed in 4 ms

9.

The screenshot shows the same PostgreSQL IDE interface. The main editor pane now shows the SQL code for creating a stored procedure named 'most_expensive_flight'. This procedure returns the flight with the highest ticket price, including details about the airports and the ticket price. The 'Database Sessions' pane at the bottom shows the execution of the procedure, which completed successfully in 16 ms.

```

1 CREATE OR REPLACE PROCEDURE most_expensive_flight()
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5   SELECT f.flight_id,
6         a1.airport_name AS departing_airport,
7         a2.airport_name AS arriving_airport,
8         b.ticket_price
9   FROM booking b
10  JOIN flights f ON f.flight_id = b.flight_id
11  JOIN airport a1 ON f.departing_airport_id = a1.airport_id
12  JOIN airport a2 ON f.arriving_airport_id = a2.airport_id
13  ORDER BY b.ticket_price DESC
14  LIMIT 1;
15 END;
16 $$;

```

```

1 JOIN flights f ON f.flight_id = b.flight_id
2 JOIN airport a1 ON f.departing_airport_id = a1.airport_id
3 JOIN airport a2 ON f.arriving_airport_id = a2.airport_id
4 ORDER BY b.ticket_price DESC
5 LIMIT 1;
6 END;
7 $$

```

[2025-12-02 23:08:33] completed in 16 ms

