

Laboratory work 10

1.

The screenshot shows a PostgreSQL database console interface. The code in the main editor window is:

```
CREATE OR REPLACE PROCEDURE new_flight(
    p_flight_id INT,
    p_sch_departure_time TIMESTAMP,
    p_sch_arrival_time TIMESTAMP,
    p_departing_airport_id INT,
    p_arriving_airport_id INT,
    p_departing_gate VARCHAR,
    p_arriving_gate VARCHAR,
    p_airline_id INT,
    p_act_departure_time TIMESTAMP,
    p_act_arrival_time TIMESTAMP
)
LANGUAGE plpgsql AS $$

BEGIN
    INSERT INTO flights(
        flight_id, sch_departure_time, sch_arrival_time,
        departing_airport_id, arriving_airport_id,
        departing_gate, arriving_gate, airline_id,
        act_departure_time, act_arrival_time,
        created_at, updated_at
    )
    VALUES(
        flight_id p_flight_id, sch_departure_time p_sch_departure_time, sch_arrival_time p_sch_arrival_time,
        departing_airport_id p_departing_airport_id, arriving_airport_id p_arriving_airport_id,
        departing_gate p_departing_gate, arriving_gate p_arriving_gate, airline_id p_airline_id,
        act_departure_time p_act_departure_time, act_arrival_time p_act_arrival_time,
        created_at NOW(), updated_at NOW()
    );
END;
$$;
```

The code defines a PostgreSQL stored procedure named `new_flight`. It takes several parameters: `p_flight_id`, `p_sch_departure_time`, `p_sch_arrival_time`, `p_departing_airport_id`, `p_arriving_airport_id`, `p_departing_gate`, `p_arriving_gate`, `p_airline_id`, `p_act_departure_time`, and `p_act_arrival_time`. The procedure uses the `plpgsql` language and contains a single `INSERT` statement into the `flights` table. The inserted values are derived from the procedure's parameters. The `VALUES` clause includes the `flight_id` parameter followed by the other parameters in a specific order: `p_sch_departure_time`, `p_sch_arrival_time`, `p_departing_airport_id`, `p_arriving_airport_id`, `p_departing_gate`, `p_arriving_gate`, `p_airline_id`, `p_act_departure_time`, `p_act_arrival_time`, `created_at` set to `NOW()`, and `updated_at` set to `NOW()`.

2.

The screenshot shows a PostgreSQL database interface. In the top-left pane, the Database Explorer lists a connection to 'console_7 [postgres@localhost [2]]'. In the main area, a code editor window displays the following PL/pgSQL code:

```
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;
$$;
```

Below the code editor, the Database Sessions pane shows the execution of the command:

```
[2025-12-02 22:41:05] completed in 5 ms
```

After the procedure is created, another session is shown:

```
[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;
$$
```

Execution time: [2025-12-02 22:46:07] completed in 4 ms

3.

The screenshot shows a PostgreSQL database interface. In the top-left pane, the Database Explorer lists a connection to 'console_7 [postgres@localhost [2]]' and shows the structure of the 'flights' table, including columns like flightId, sch_departure_time, sch_arrival_time, departing_airport_id, arriving_airport_id, arriving_gate, departing_gate, airline_id, act_departure_time, act_arrival_time, created_at, and updated_at.

In the main area, a code editor window displays the following PL/pgSQL code:

```
CREATE OR REPLACE FUNCTION avg_arrival_delay_time(
    p_airport_id INT
)
RETURNS NUMERIC
LANGUAGE plpgsql
AS $$

DECLARE
    avg_delay NUMERIC;
BEGIN
    SELECT AVG(EXTRACT(EPOCH FROM (act_arrival_time - sch_arrival_time)) / 60)
    INTO avg_delay
    FROM flights
    WHERE arriving_airport_id = p_airport_id;

    RETURN avg_delay;
END;
$$;
```

Below the code editor, the Database Sessions pane shows the execution of the command:

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

4.

The screenshot shows a PostgreSQL database interface. In the top right, there's a tab labeled "console_7 [postgres@localhost [2]]". Below it, the "Database Explorer" pane shows a table named "booking_flight" with columns "booking_flight_id", "booking_id", and "flight_id". The main area contains the following PL/pgSQL code:

```
CREATE OR REPLACE PROCEDURE pass_flight(
    p_flight_id INT
)
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT p.* FROM passengers p
    JOIN booking b ON p.passenger_id=b.passenger_id
    WHERE b.flight_id=p_flight_id;
END;
$$;
```

In the "Database Sessions" pane, a transaction is running under the session "postgres@console". The same PL/pgSQL code is visible in the "Output" tab. The status bar at the bottom indicates the command was completed in 6 ms on 2025-12-02 23:01:07.

5.

The screenshot shows a PostgreSQL database interface. In the top right, there's a tab labeled "console_7 [postgres@localhost [2]]". Below it, the "Database Explorer" pane shows a table named "booking_flight" with columns "booking_flight_id", "booking_id", "flight_id", "created_at", and "updated_at". The main area contains the following PL/pgSQL code:

```
CREATE OR REPLACE PROCEDURE top_passenger()
LANGUAGE plpgsql
AS $$
BEGIN
    SELECT p.passenger_id,
           p.first_name,
           p.last_name,
           COUNT(*) AS total_flights
    FROM passengers p
    JOIN booking b ON p.passenger_id = b.passenger_id
    GROUP BY p.passenger_id
    ORDER BY total_flights DESC
    LIMIT 1;
END;
$$;
```

In the "Database Sessions" pane, a transaction is running under the session "postgres@console". The same PL/pgSQL code is visible in the "Output" tab. The status bar at the bottom indicates the command was completed in 5 ms on 2025-12-02 23:02:23.

6.

The screenshot shows a PostgreSQL database interface with the following details:

- Database Explorer:** Shows the schema of the `bookings` table, which has columns: `status`, `ticket_price`, `ticket_discount`, `keys`, `foreign keys`, `indexes`, `checks`, `booking_flight`, and `columns`.
- Console:** A code editor window titled `console_7 [postgres@localhost [2]]` containing the following PL/pgSQL code:

```
CREATE OR REPLACE PROCEDURE delayed_24h()
LANGUAGE plpgsql
AS $$

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;
$$;
```
- Database Sessions:** Shows a single session named `postgres@console` connected to the `flights` table.
- Output:** Displays the execution results of the query, indicating it completed in 4 ms.

7.

The screenshot shows a PostgreSQL database interface with the following details:

- Database Explorer:** Shows the schema of the `bookings` table, which has columns: `status`, `ticket_price`, `ticket_discount`, `keys`, `foreign keys`, `indexes`, `checks`, `booking_flight`, and `columns`.
- Console:** A code editor window titled `console_7 [postgres@localhost [2]]` containing the following PL/pgSQL code:

```
CREATE OR REPLACE FUNCTION flight_airline(
    p_airline_id INT
)
RETURNS INT
LANGUAGE plpgsql
AS $$

DECLARE
    result INT;
BEGIN
    SELECT COUNT(*) INTO result FROM flights WHERE airline_id = p_airline_id;
    RETURN result;
END;
$$;
```
- Database Sessions:** Shows a single session named `postgres@console` connected to the `flights` table.
- Output:** Displays the execution results of the query, indicating it completed in 4 ms.

8.

The screenshot shows a PostgreSQL database interface with the following details:

- Database Explorer:** Shows the schema of the `booking` table, which contains columns: `status`, `ticket_price`, `ticket_discount`, `keys`, `foreign keys`, `indexes`, `checks`, and `booking_flight`. The `booking_flight` table has columns: `booking_flight_id`, `booking_id`, `flight_id`, `created_at`, and `updated_at`.
- Console:** The code being run is a stored procedure named `avg_ticket_price` with the following SQL:

```
CREATE OR REPLACE PROCEDURE avg_ticket_price()
  p_flight_id INT
)
LANGUAGE plpgsql
AS $$

BEGIN
  SELECT AVG(ticket_price) AS average_price FROM booking
  WHERE flight_id = p_flight_id;
END;
$$;
```
- Database Sessions:** Shows a session for `postgres@localhost` running the same SQL query as the stored procedure.
- Output:** Displays the result of the query: [2025-12-02 23:05:57] completed in 4 ms.

9.

The screenshot shows a PostgreSQL database interface with the following details:

- Database Explorer:** Shows the schema of the `booking` table, which contains columns: `status`, `ticket_price`, `ticket_discount`, `keys`, `foreign keys`, `indexes`, `checks`, and `booking_flight`. The `booking_flight` table has columns: `booking_flight_id`, `booking_id`, `flight_id`, `created_at`, and `updated_at`.
- Console:** The code being run is a stored procedure named `most_expensive_flight` with the following SQL:

```
CREATE OR REPLACE PROCEDURE most_expensive_flight()
LANGUAGE plpgsql
AS $$

BEGIN
  SELECT f.flight_id,
         a1.airport_name AS departing_airport,
         a2.airport_name AS arriving_airport,
         b.ticket_price
  FROM booking b
  JOIN flights f ON f.flight_id = b.flight_id
  JOIN airport a1 ON f.departing_airport_id = a1.airport_id
  JOIN airport a2 ON f.arriving_airport_id = a2.airport_id
  ORDER BY b.ticket_price DESC
  LIMIT 1;
END;
$$;
```
- Database Sessions:** Shows a session for `postgres@localhost` running the same SQL query as the stored procedure.
- Output:** Displays the result of the query: [2025-12-02 23:08:33] completed in 16 ms.

