

1)

The screenshot shows the Database Explorer in VS Code. A new procedure named `insert_flight` has been created in the `flights` schema. The code defines the procedure to insert data into the `flights` table.

```
CREATE OR REPLACE PROCEDURE insert_flight(
    flight_id INT, flight_no VARCHAR(50), scheduled_departure DATE,
    scheduled_arrival DATE, departure_airport_id INT, arrival_airport_id INT, departing_gate VARCHAR(50),
    arriving_gate VARCHAR(50), airline_id INT, status VARCHAR(50),
    actual_departure DATE, actual_arrival DATE, created_at DATE, update_at DATE
)
AS $$
BEGIN
    INSERT INTO flights (flight_id, flight_no, scheduled_departure,
        scheduled_arrival, departure_airport_id, arrival_airport_id, arrival_airport_id,
        departing_gate, arriving_gate, airline_id, status, actual_departure, actual_arrival, created_at, update_at)
    VALUES (flight_id, flight_no, scheduled_departure,
        scheduled_arrival, departure_airport_id, arrival_airport_id, arrival_airport_id,
        departing_gate, arriving_gate, airline_id, status, actual_departure, actual_arrival, created_at, update_at);
    RAISE NOTICE 'Flight % successfully added', flight_no;
END;
$$ LANGUAGE plpgsql;
```

The procedure uses the `plpgsql` language and includes a `RAISE NOTICE` statement to log the successful addition of the flight.

2)

The screenshot shows the Database Explorer in VS Code. A new procedure named `update_flight_status` has been created in the `flights` schema. The code defines the procedure to update the `status` column in the `flights` table based on the provided `p_flight_id` and `p_new_status`.

```
CREATE OR REPLACE PROCEDURE update_flight_status(
    p_flight_id INT,
    p_new_status VARCHAR(20)
)
AS $$
BEGIN
    UPDATE flights
    SET status = p_new_status
    WHERE flight_id = p_flight_id;
    RAISE NOTICE 'Flight % status updated to: %', p_flight_id, p_new_status;
END;
$$ LANGUAGE plpgsql;
```

The procedure uses the `plpgsql` language and includes a `RAISE NOTICE` statement to log the updated status.

3)

4)

```

CREATE OR REPLACE FUNCTION get_flights_by_departure_airport(
    p_departure_airport VARCHAR(50)
)
RETURNS TABLE (
    Flight_id INT, flight_no VARCHAR(50), scheduled_departure DATE,
    scheduled_arrival DATE, departure.airport_id INT, arrival.airport_id INT, departing.pate VARCHAR(50),
    arriving_gate VARCHAR(50), airline_id INT, status VARCHAR(50),
    actual_departure DATE, actual_arrival DATE, created_at DATE, update_at DATE
)
AS $$
BEGIN
    RETURN QUERY
    SELECT
        f.Flight_id,
        f.Flight_no,
        f.Scheduled_departure,
        f.Scheduled_arrival,
        air.Airport_id,
        air.Airport_name
    FROM flights f
    JOIN airport air ON f.departure_airport_id = air.Airport_id
    WHERE air.Airport_name = 'Fort Worth Alliance Airport';
END;
$$ LANGUAGE plpgsql;

```

4)

```

CREATE OR REPLACE FUNCTION calculate_average_delay()
RETURNS DECIMAL(10,2)
AS $$
DECLARE
    avg_delay DECIMAL(10,2);
BEGIN
    SELECT AVG(
        EXTRACT(EPOCH FROM (actual_arrival - scheduled_arrival)) / 60
    ) INTO avg_delay
    FROM flights
    WHERE arrival.airport_id = p_arrival_airport_id
    AND actual_arrival > scheduled_arrival;
    RETURN COALESCE(avg_delay, 0);
END;
$$ LANGUAGE plpgsql;

```

5)

6)

```

CREATE OR REPLACE FUNCTION get_passenger_by_flight(
    p_flight_number VARCHAR(20)
)
RETURNS TABLE (
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100),
    seat_number VARCHAR(10)
)
AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.first_name,
        p.last_name
    FROM passengers p
    JOIN booking b ON p.passenger_id = b.passenger_id
    JOIN booking_flight bf ON b.booking_id = bf.booking_id
    JOIN flights f ON bf.flight_id = f.flight_id
    WHERE f.flight_no= p_flight_number;
END;
$$ LANGUAGE plpgsql;

```

The screenshot shows the PostgreSQL Database Explorer interface. The 'public' schema is selected. A new function 'get\_passenger\_by\_flight' is being created. The code defines a function that takes a flight number as input and returns a table of passenger details (first name, last name, email, seat number) who have booked that flight. The function uses joins between the 'passenger', 'booking', 'booking\_flight', and 'flights' tables.

6)

```

CREATE OR REPLACE FUNCTION get_frequent_flyer()
RETURNS TABLE (
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    total_flights BIGINT
)
AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.first_name,
        p.last_name,
        COUNT(b.booking_id)::BIGINT
    FROM passengers p
    JOIN booking b ON p.passenger_id = b.passenger_id
    GROUP BY p.passenger_id
    ORDER BY COUNT(b.booking_id) DESC
    LIMIT 1;
END;
$$ LANGUAGE plpgsql;

```

The screenshot shows the PostgreSQL Database Explorer interface. The 'public' schema is selected. A new function 'get\_frequent\_flyer' is being created. The code defines a function that returns the most frequent flyer (passenger with the highest number of bookings). It groups passengers by ID and counts their bookings, then orders by count and limits the result to one row.

7)

8)

```

CREATE OR REPLACE FUNCTION get_severely_delayed_flights()
RETURNS TABLE (
    flight_number VARCHAR(20),
    departure_time TIMESTAMP,
    arrival_time TIMESTAMP,
    arrival_code VARCHAR(10),
    delay_hours DECIMAL(8,2)
)
AS $$
BEGIN
    RETURN QUERY
    SELECT
        f.flight_no,
        f.departure_time,
        f.arrival_time,
        f.arrival_code,
        EXTRACT(EPOCH FROM (f.actual_departure - f.scheduled_departure)) / 3600
    FROM flights f
    JOIN airport dep ON f.departure_airport_id = dep.airport_id
    JOIN airport arr ON f.arrival_airport_id = arr.airport_id
    WHERE (f.actual_departure - f.scheduled_departure) > INTERVAL '24 hours'
    ORDER BY delay_hours DESC;
END;
$$ LANGUAGE plpgsql;

```

The screenshot shows a PostgreSQL database interface with the 'console' tab selected. The code editor displays a function named 'get\_severely\_delayed\_flights'. The function uses a RETURN TABLE clause to select flight numbers, departure times, arrival times, arrival codes, and calculated delay hours from the 'flights' table. It joins with 'airport' tables for departure and arrival locations and filters for flights delayed by more than 24 hours. The code is written in PL/pgSQL.

8)

```

CREATE OR REPLACE FUNCTION count_flights_per_airline(
    p_airline_id INT
)
RETURNS INT
AS $$
DECLARE
    flight_count INT;
BEGIN
    SELECT COUNT(*) INTO flight_count
    FROM flights
    WHERE airline_id = p_airline_id;
    RETURN flight_count;
END;
$$ LANGUAGE plpgsql;

```

The screenshot shows a PostgreSQL database interface with the 'console' tab selected. The code editor displays a function named 'count\_flights\_per\_airline'. The function takes a parameter 'p\_airline\_id' of type INT and returns an integer representing the count of flights for that airline. It uses a SELECT INTO statement to count the number of flights where the airline ID matches the input parameter. The code is written in PL/pgSQL.

9)

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

- Database Explorer:** Shows the schema structure with tables: `airline`, `airport`, `baggage`, `baggage_check`, `boarding_pass`, `booking`, `booking_flight`, `flights`, `passenger`, and `security_check`.
- Console:** Displays the SQL code for creating a function `get_average_ticket_price`. The function takes a flight number as input and returns the average price of tickets for that flight.
- Services:** Shows two transactions: `booking` (479) and `console` (26). The `console` transaction was completed in 4 ms at [2025-12-02 18:28:45].
- Bottom Status:** Includes icons for file operations and status information: 19:1 CRLF, UTF-8, 4 spaces.

10)

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

- Database Explorer:** Shows the schema structure with tables: `airline`, `airport`, `baggage`, `baggage_check`, `boarding_pass`, `booking`, `booking_flight`, `flights`, `passenger`, and `security_check`.
- Console:** Displays the SQL code for creating a function `get_most_expensive_flight`. The function returns the flight with the highest price, along with departure and arrival airport names.
- Services:** Shows two transactions: `booking` (479) and `console` (26). The `console` transaction was completed in 4 ms at [2025-12-02 18:30:44].
- Bottom Status:** Includes icons for file operations and status information: 25:21 CRLF, UTF-8, 4 spaces.