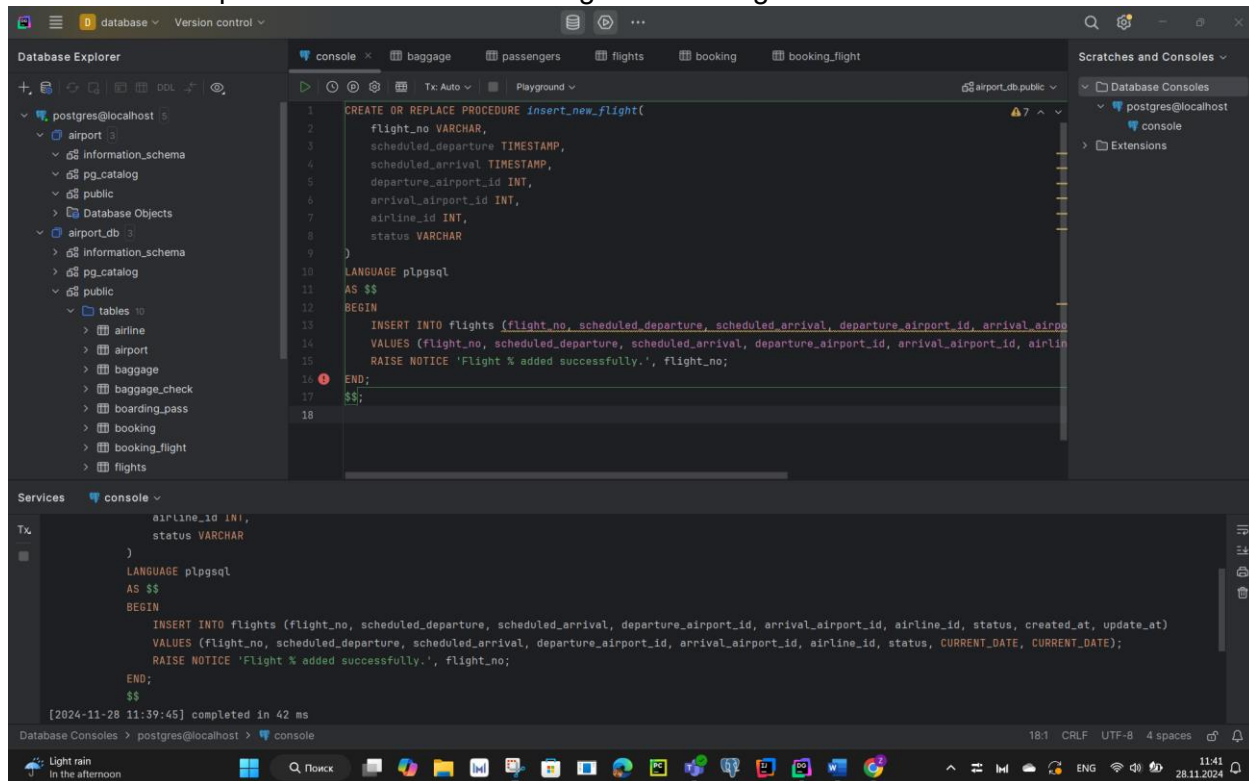


Laboratory work 10

STORED PROCEDURES.

1. Create a stored procedure to insert a new flight into the flights table.

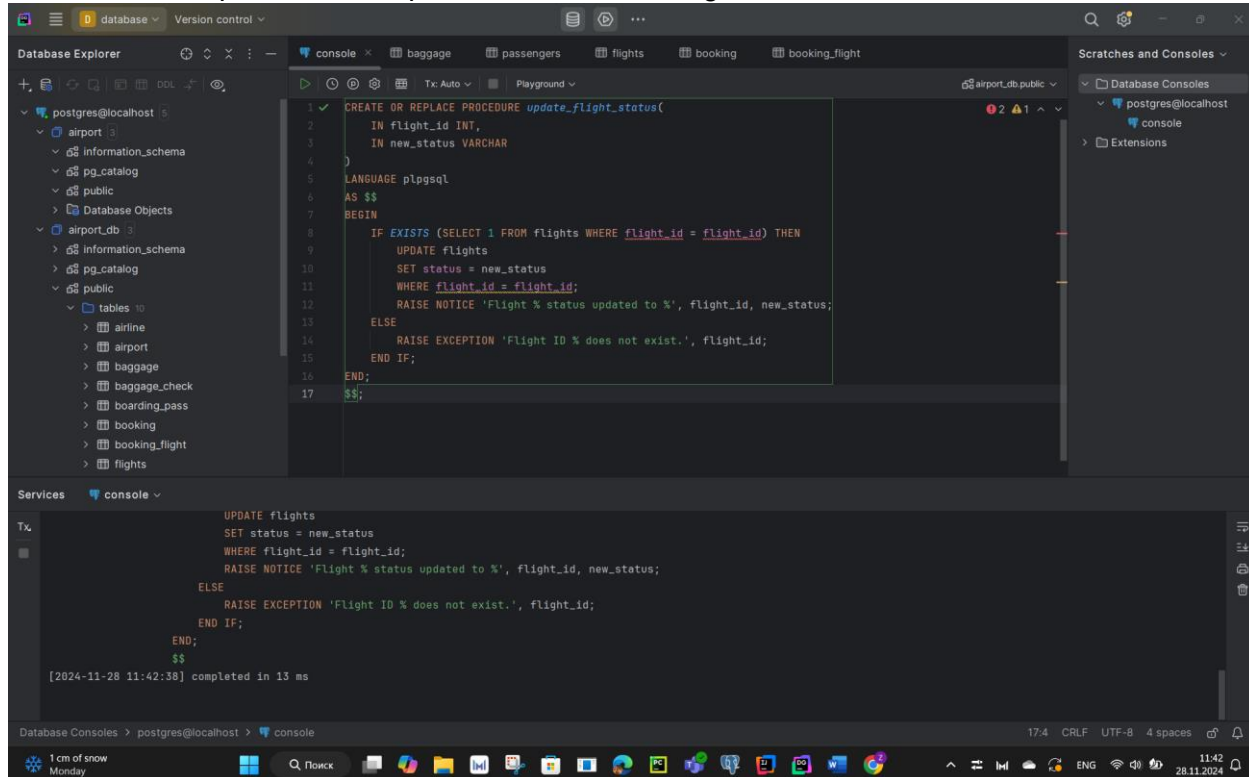


The screenshot shows a database IDE with a 'Database Explorer' on the left and a 'console' window in the center. The 'Database Explorer' shows a database named 'airport_db' with a schema 'public' containing a table 'flights'. The 'console' window displays the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE insert_new_flight(  
2     flight_no VARCHAR,  
3     scheduled_departure TIMESTAMP,  
4     scheduled_arrival TIMESTAMP,  
5     departure_airport_id INT,  
6     arrival_airport_id INT,  
7     airline_id INT,  
8     status VARCHAR  
9 )  
10 LANGUAGE plpgsql  
11 AS $$  
12 BEGIN  
13     INSERT INTO flights (flight_no, scheduled_departure, scheduled_arrival, departure_airport_id, arrival_airport_id, airline_id, status, created_at, update_at)  
14     VALUES (flight_no, scheduled_departure, scheduled_arrival, departure_airport_id, arrival_airport_id, airline_id, status, CURRENT_DATE, CURRENT_DATE);  
15     RAISE NOTICE 'Flight % added successfully.', flight_no;  
16 END;  
17 $$;
```

The console window also shows the execution result: [2024-11-28 11:39:45] completed in 42 ms.

2. Create a stored procedure to update the status of a flight.



The screenshot shows a database IDE with a 'Database Explorer' on the left and a 'console' window in the center. The 'Database Explorer' shows a database named 'airport_db' with a schema 'public' containing a table 'flights'. The 'console' window displays the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE update_flight_status(  
2     IN flight_id INT,  
3     IN new_status VARCHAR  
4 )  
5 LANGUAGE plpgsql  
6 AS $$  
7 BEGIN  
8     IF EXISTS (SELECT 1 FROM flights WHERE flight_id = flight_id) THEN  
9         UPDATE flights  
10        SET status = new_status  
11        WHERE flight_id = flight_id;  
12        RAISE NOTICE 'Flight % status updated to %', flight_id, new_status;  
13     ELSE  
14        RAISE EXCEPTION 'Flight ID % does not exist.', flight_id;  
15     END IF;  
16 END;  
17 $$;
```

The console window also shows the execution result: [2024-11-28 11:42:38] completed in 13 ms.

3. Create a stored procedure that returns a list of flights departing from a specific airport.

The screenshot shows a database console interface with a dark theme. On the left, the 'Database Explorer' pane shows a tree view of the database structure, including schemas like 'information_schema', 'pg_catalog', and 'public', and tables like 'airline', 'airport', 'baggage', 'baggage_check', 'boarding_pass', 'booking', 'booking_flight', and 'flights'. The main console area displays a SQL script for creating a stored procedure named 'list_flights_by_airport'. The script is as follows:

```
1 CREATE OR REPLACE PROCEDURE list_flights_by_airport(  
2     IN airport_code VARCHAR  
3 )  
4 LANGUAGE plpgsql  
5 AS $$  
6 BEGIN  
7     RAISE NOTICE 'Flights departing from airport %:', airport_code;  
8     PERFORM * FROM flights WHERE update_at = airport_code;  
9 END;  
10 $$;
```

Below the script, the 'Services' pane shows the execution details: 'Tx: [2024-11-28 11:43:13] completed in 9 ms'. The bottom status bar indicates the console is connected to 'postgres@localhost' and shows various settings like '10:4 CRLF UTF-8 4 spaces'.

4. Create a stored procedure to calculate the average delay time of flights arriving at a specific airport.

The screenshot shows the same database console interface. The main console area displays a SQL script for creating a stored procedure named 'calculate_avg_delay'. The script is as follows:

```
1 CREATE OR REPLACE PROCEDURE calculate_avg_delay(  
2     IN airport_code VARCHAR,  
3     OUT avg_delay INTERVAL  
4 )  
5 LANGUAGE plpgsql  
6 AS $$  
7 BEGIN  
8     SELECT AVG(actual_arrival - scheduled_arrival)  
9     INTO avg_delay  
10    FROM flights  
11    WHERE arrival_airport_id = airport_code AND actual_arrival IS NOT NULL;  
12    RAISE NOTICE 'Average delay for airport % is %', airport_code, avg_delay;  
13 END;  
14 $$;
```

Below the script, the 'Services' pane shows the execution details: 'Tx: [2024-11-28 11:43:42] completed in 9 ms'. The bottom status bar indicates the console is connected to 'postgres@localhost' and shows various settings like '14:4 CRLF UTF-8 4 spaces'.

5. Create a stored procedure that lists all passengers for a given flight number.

The screenshot shows a database console interface with a dark theme. On the left, the 'Database Explorer' pane shows a tree view of the database structure, including 'airport_db' and its tables. The main console area displays the SQL code for creating a stored procedure named `list_passengers_for_flight`. The code is as follows:

```
1 CREATE OR REPLACE PROCEDURE list_passengers_for_flight(  
2     IN flight_id INT  
3 )  
4 LANGUAGE plpgsql  
5 AS $$  
6 BEGIN  
7     RAISE NOTICE 'Passengers for flight %:', flight_id;  
8     PERFORM * FROM passengers p  
9     JOIN booking b ON p.passenger_id = b.passenger_id  
10    WHERE b.booking_id = flight_id;  
11 END;  
12 $$;
```

Below the code editor, the 'Services' pane shows the execution details of the procedure, including the SQL code and the execution time: [2024-11-28 11:44:41] completed in 9 ms.

6. Create a stored procedure to find the passenger who has taken the greatest number of flights.

The screenshot shows a database console interface with a dark theme. On the left, the 'Database Explorer' pane shows a tree view of the database structure, including 'airport_db' and its tables. The main console area displays the SQL code for creating a stored procedure named `find_top_passenger`. The code is as follows:

```
1 CREATE OR REPLACE PROCEDURE find_top_passenger()  
2 LANGUAGE plpgsql  
3 AS $$  
4 BEGIN  
5     RAISE NOTICE 'Passenger with the most flights:';  
6     PERFORM passenger_id, COUNT(*) AS flight_count  
7     FROM booking  
8     GROUP BY passenger_id  
9     ORDER BY flight_count DESC  
10    LIMIT 1;  
11 END;  
12 $$;
```

Below the code editor, the 'Services' pane shows the execution details of the procedure, including the SQL code and the execution time: [2024-11-28 11:45:09] completed in 19 ms.

7. Create a stored procedure to find all flights that are delayed by more than 24 hours.

```
1 CREATE OR REPLACE PROCEDURE find_long_delays()
2 LANGUAGE plpgsql
3 AS $$
4 BEGIN
5     RAISE NOTICE 'Flights delayed by more than 24 hours: ';
6     PERFORM * FROM flights
7     WHERE actual_departure - scheduled_departure > INTERVAL '24 hours';
8 END;
9 $$;
```

Services console

airport_db.public> CREATE OR REPLACE PROCEDURE find_long_delays()
LANGUAGE plpgsql
AS \$\$
BEGIN
 RAISE NOTICE 'Flights delayed by more than 24 hours: ';
 PERFORM * FROM flights
 WHERE actual_departure - scheduled_departure > INTERVAL '24 hours';
END;
\$\$
[2024-11-28 11:45:36] completed in 7 ms

8. Create a stored procedure that counts the number of flights for each airline.

```
1 CREATE OR REPLACE FUNCTION CountFlightsByAirline()
2 RETURNS TABLE (airline_name VARCHAR, flight_count INT) AS
3 $$
4 BEGIN
5     RETURN QUERY
6     SELECT a.airline_name, COUNT(f.flight_id) AS flight_count
7     FROM Airline a
8     JOIN Flights f ON a.airline_id = f.airline_id
9     GROUP BY a.airline_name;
10 END;
11 $$ LANGUAGE plpgsql;
```

Services console

airport_db.public> \$\$
BEGIN
 RETURN QUERY
 SELECT a.airline_name, COUNT(f.flight_id) AS flight_count
 FROM Airline a
 JOIN Flights f ON a.airline_id = f.airline_id
 GROUP BY a.airline_name;
END;
\$\$ LANGUAGE plpgsql
[2024-11-28 11:49:44] completed in 10 ms

9. Create a stored procedure to calculate the average ticket price for a specific flight.

The screenshot shows a database console interface with a 'Database Explorer' on the left and a 'console' window in the center. The 'Database Explorer' shows a PostgreSQL database with a schema named 'airport_db' containing a table 'flights'. The 'console' window displays the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE calculate_avg_ticket_price(  
2     IN flight_id INT,  
3     OUT avg_price NUMERIC  
4 )  
5 LANGUAGE plpgsql  
6 AS $$  
7 BEGIN  
8     SELECT AVG(price)  
9     INTO avg_price  
10    FROM booking  
11    WHERE flight_id = flight_id;  
12    RAISE NOTICE 'Average ticket price for flight % is %', flight_id, avg_price;  
13 END;  
14 $$;
```

Below the code, the 'Services' section shows the execution of the procedure:

```
Tx: AS $$  
BEGIN  
    SELECT AVG(price)  
    INTO avg_price  
    FROM booking  
    WHERE flight_id = flight_id;  
    RAISE NOTICE 'Average ticket price for flight % is %', flight_id, avg_price;  
END;  
$$  
[2024-11-28 11:50:59] completed in 8 ms
```

The bottom status bar indicates the console is running on 'postgres@localhost' with a 'console' window, showing 14.4 CRLF, UTF-8, and 4 spaces.

10. Create a stored procedure to find the flight with the highest ticket price. The procedure should return the flight number, the departure and arrival airports, and the ticket price for the most expensive flight.

The screenshot shows a database console interface with a 'Database Explorer' on the left and a 'console' window in the center. The 'Database Explorer' shows a PostgreSQL database with a schema named 'airport_db' containing a table 'flights'. The 'console' window displays the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE find_highest_priced_flight(  
2     OUT flight_info RECORD  
3 )  
4 LANGUAGE plpgsql  
5 AS $$  
6 BEGIN  
7     SELECT f.flight_id, f.update_at, f.created_at, MAX(b.price) AS highest_price  
8     INTO flight_info  
9     FROM flights f  
10    JOIN booking b ON f.flight_id = b.booking_id  
11    GROUP BY f.flight_id, f.update_at, f.created_at  
12    ORDER BY highest_price DESC  
13    LIMIT 1;  
14    RAISE NOTICE 'Flight with highest ticket price: %', flight_info;  
15 END;  
16 $$;
```

Below the code, the 'Services' section shows the execution of the procedure:

```
Tx: INTO flight_info  
FROM flights f  
JOIN booking b ON f.flight_id = b.booking_id  
GROUP BY f.flight_id, f.update_at, f.created_at  
ORDER BY highest_price DESC  
LIMIT 1;  
RAISE NOTICE 'Flight with highest ticket price: %', flight_info;  
END;  
$$  
[2024-11-28 11:51:26] completed in 16 ms
```

The bottom status bar indicates the console is running on 'postgres@localhost' with a 'console' window, showing 16.4 CRLF, UTF-8, and 4 spaces.