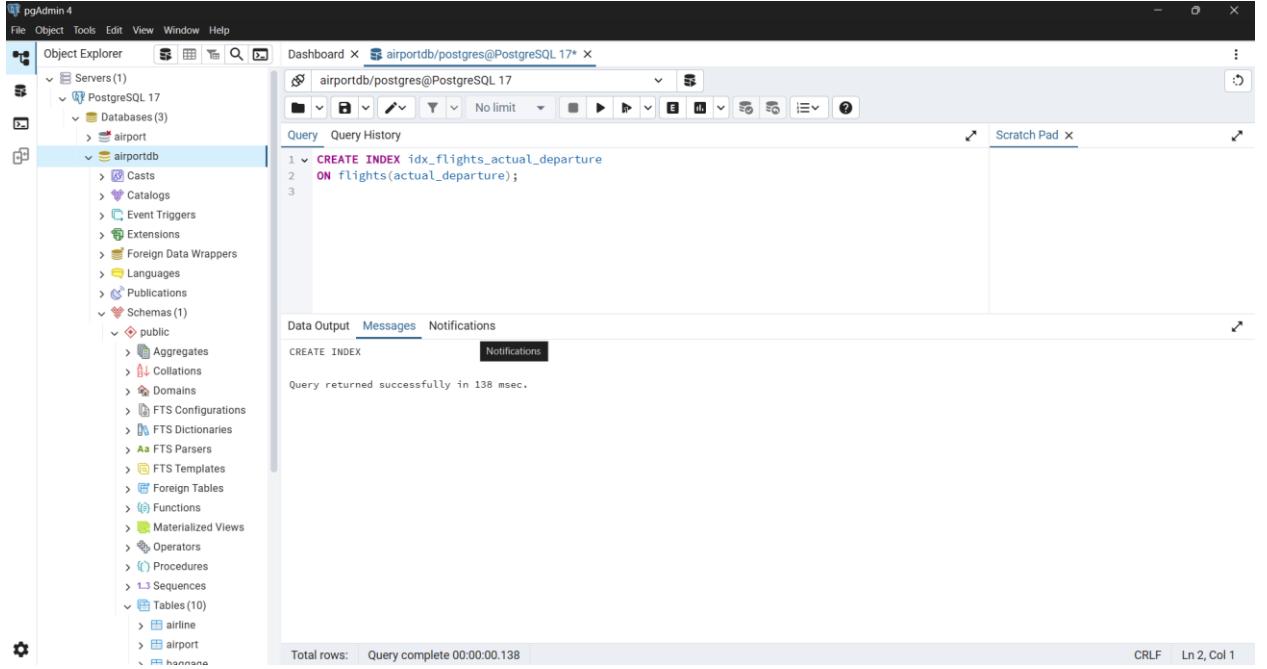


Laboratory work 7

1. Create an index on the actual_departure column in the flights table.

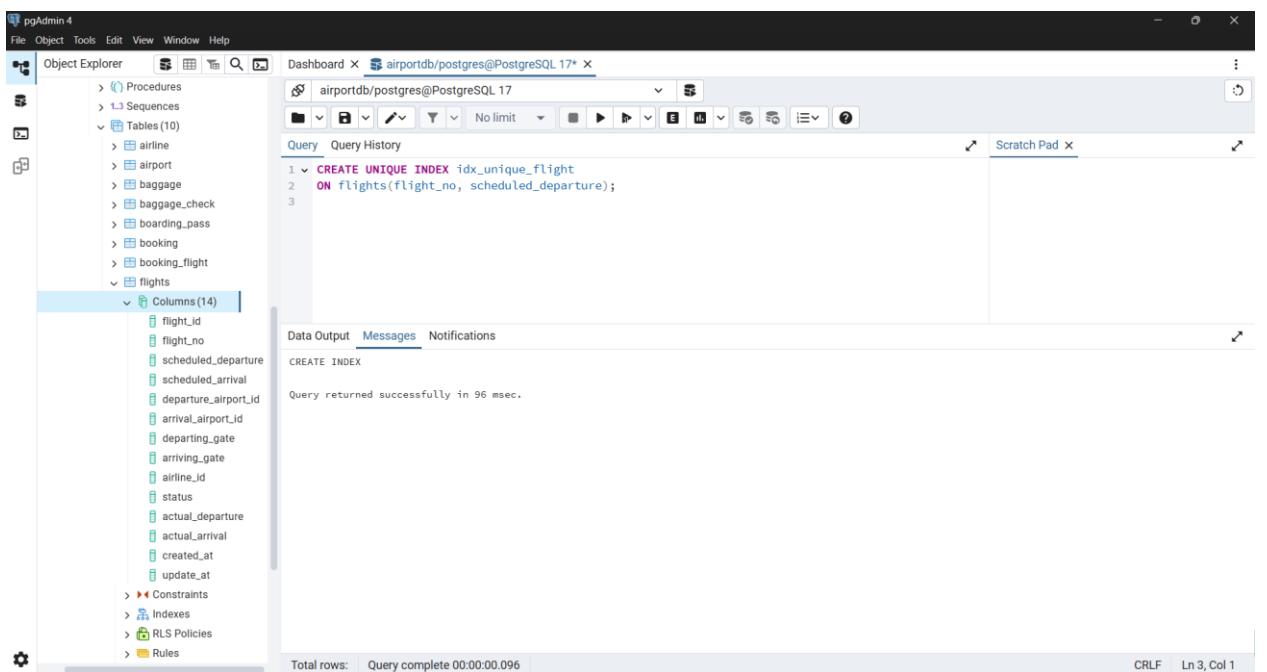


The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'airportdb' database, the 'Tables' node is expanded, showing the 'flights' table. The 'Columns' node for the 'flights' table is also expanded, showing the 'actual_departure' column. In the main query editor, the following SQL command is run:

```
CREATE INDEX idx_flights_actual_departure  
ON flights(actual_departure);
```

The 'Messages' tab shows the result: "Query returned successfully in 138 msec."

2. Create a unique index to ensure flight_no and scheduled_departure combinations are unique.



The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'airportdb' database, the 'Tables' node is expanded, showing the 'flights' table. The 'Columns' node for the 'flights' table is also expanded, showing the 'flight_no' and 'scheduled_departure' columns. In the main query editor, the following SQL command is run:

```
CREATE UNIQUE INDEX idx_unique_flight  
ON flights(flight_no, scheduled_departure);
```

The 'Messages' tab shows the result: "Query returned successfully in 96 msec."

3. Create a composite index on the departure_airport_id and arrival_airport_id columns.

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'flights' table, the 'Columns (14)' section is selected. A query window titled 'Query History' contains the following SQL code:

```

CREATE INDEX idx_flights_airports
ON flights(departure_airport_id, arrival_airport_id);

```

The status bar at the bottom right indicates: 'Query returned successfully in 50 msec.' and 'CRLF Ln 3, Col 1'.

4. Evaluate the difference in query performance with and without indexes.
Measure performance differences.

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, under the 'flights' table, the 'Columns (14)' section is selected. A query window titled 'Query History' contains the following SQL code:

```

EXPLAIN ANALYZE
SELECT * FROM flights
WHERE departing_gate = 'DME'
    AND arriving_gate = 'ALA';

```

The results pane displays the 'QUERY PLAN' for the EXPLAIN ANALYZE command, showing the execution plan and statistics. The status bar at the bottom right indicates: 'Showing rows: 1 to 5' and 'CRLF Ln 4, Col 20'.

5. Use EXPLAIN ANALYZE to check index usage in a query filtering by departure_airport and arrival_airport.

```

EXPLAIN ANALYZE
SELECT * FROM flights
WHERE departing_gate = 'DME'
    AND arriving_gate = 'ALA';

```

Query Plan text:

- Seq Scan on flights (cost=0.00..27.89 rows=1 width=61) (actual time=0.249..0.249 rows=0 loops=1)
- Filter: (((departing_gate)::text = 'DME'::text) AND ((arriving_gate)::text = 'ALA'::text))
- Rows Removed by Filter: 993
- Planning Time: 0.148 ms
- Execution Time: 0.266 ms

6. Create a unique index for the passport_number of the Passengers table. Check if the index was created or not. Insert into the table two new passengers.
Explain in your own words what is going on in the output?

```

CREATE UNIQUE INDEX idx_passengers_passport
ON passengers(passport_number);

```

Data Output:

CREATE INDEX

Query returned successfully in 66 msec.

7. Create an index for the Passengers table. Use for that first name, last name, date of birth and country of citizenship. Then, write a SQL query to find a passenger who was born in Philippines and was born in 1984 and check if the query uses indexes or not. Give the explanation of the results.

The screenshot shows the pgAdmin 4 interface. In the Object Explorer, the 'flights' table is selected, and its columns are listed. A query window titled 'airportdb/postgres@PostgreSQL 17*' contains the following SQL code:

```

1 v CREATE INDEX idx_passengers_full
2 ON passengers(first_name, last_name, date_of_birth, country_of_citizenship);
3

```

The 'Messages' tab shows the result of the query: 'CREATE INDEX'. Below it, a message states 'Query returned successfully in 71 msec.' The status bar at the bottom right indicates 'Query complete 00:00:00.071'.

8. Write a SQL query to list indexes for table Passengers. After delete the created indexes.

The screenshot shows the pgAdmin 4 interface. The 'flights' table is selected in the Object Explorer. A query window titled 'airportdb/postgres@PostgreSQL 17*' contains the following SQL code:

```

1 SELECT * FROM pg_indexes WHERE tablename = 'passenger';
2
3 DROP INDEX IF EXISTS idx_passenger_passport;
4 DROP INDEX IF EXISTS idx_passenger_full;
5

```

The 'Messages' tab shows the result of the query: 'DROP INDEX'. Below it, a message states 'Query returned successfully in 60 msec.' The status bar at the bottom right indicates 'Query complete 00:00:00.060'.