

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

The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

```
1 create index f_actual_departure on flights(actual_departure)
2
3
4
5
6
```

The Data Output pane at the bottom displays an error message:

```
ERROR: relation "f_actual_departure" already exists
SQL state: 42P07
```

The status bar at the bottom indicates "Total rows: 1000" and "Query complete 00:00:00.065".

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

The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

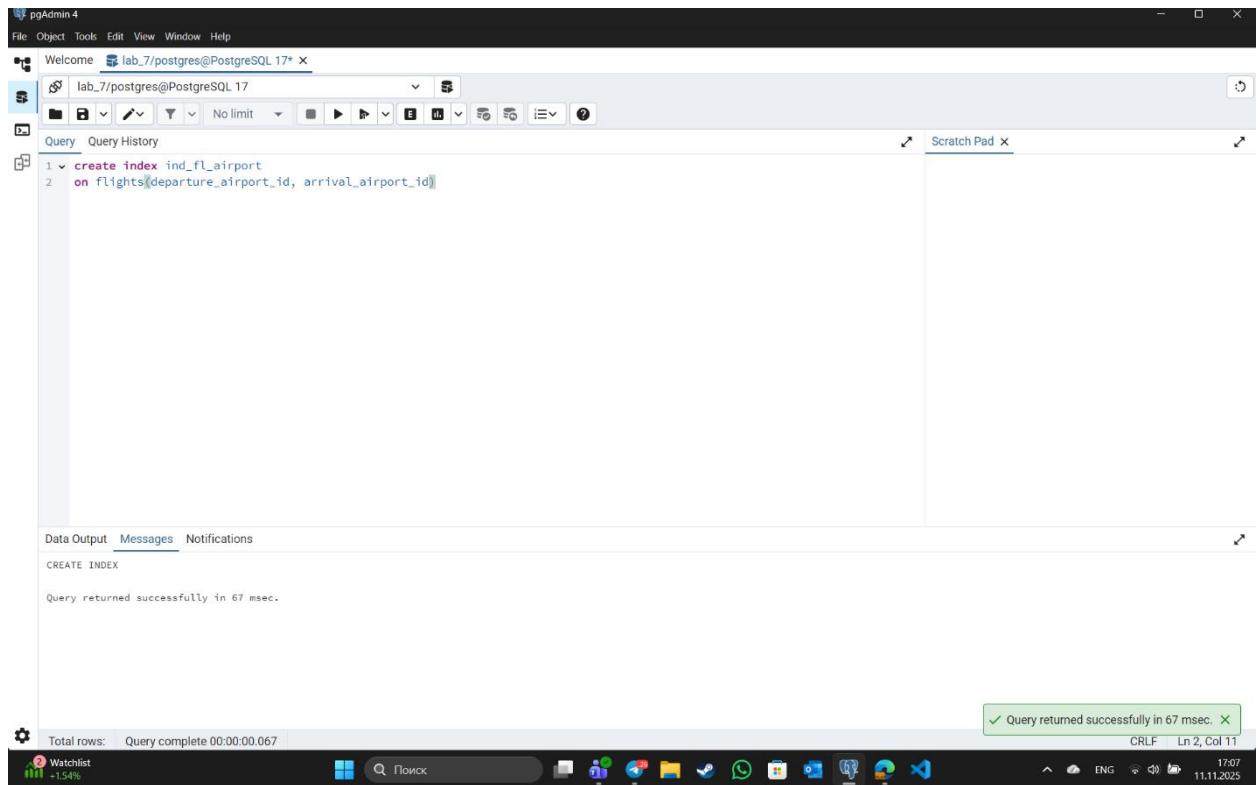
```
1 CREATE UNIQUE INDEX f_no_sch_dep
2 ON flights (flight_no, scheduled_departure);
3
4
```

The Data Output pane at the bottom displays an error message:

```
ERROR: could not create unique index "f_no_sch_dep"
Key (flight_no, scheduled_departure)=(US-KS, 2023-09-04) is duplicated.
SQL state: 23505
Detail: Key (flight_no, scheduled_departure)=(US-KS, 2023-09-04) is duplicated.
```

The status bar at the bottom indicates "Total rows: 2" and "Query complete 00:00:00.062".

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

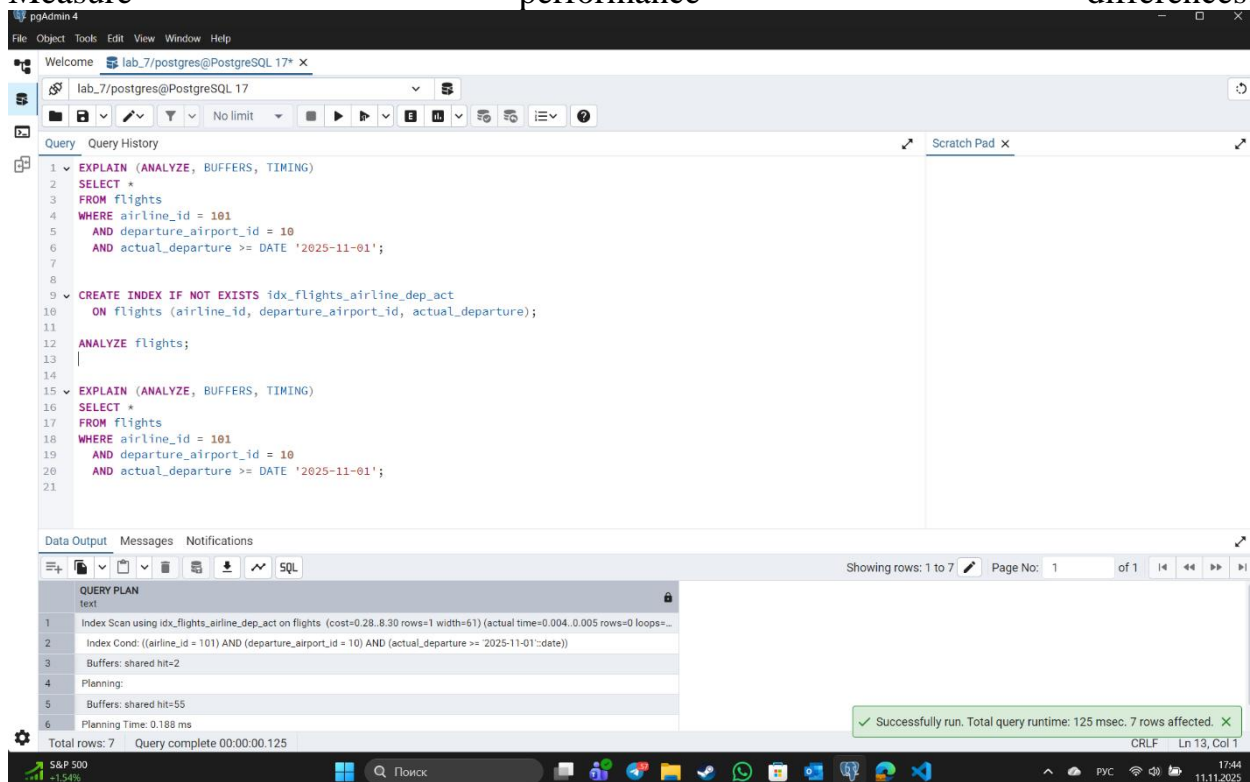


4. Evaluate the difference in query performance with and without indexes.

Measure

performance

differences.



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

The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

```
1 create index if not exists fl_airport
2 on flights(departure_airport_id, arrival_airport_id);
3
4 analyze flights;
5
6 explain(analyze)
7 select *
8 from flights
9 where departure_airport_id = 10
10 and arrival_airport_id = 25;
```

The Data Output tab shows the query plan:

QUERY PLAN
text
1 Index Scan using fl_airport on flights (cost=0.28..8.29 rows=1 width=61) (actual time=0.188..0.188 rows=0 loops=...)
2 Index Cond: ((departure_airport_id = 10) AND (arrival_airport_id = 25))
3 Planning Time: 0.421 ms
4 Execution Time: 0.223 ms

The status bar at the bottom indicates "Total rows: 4" and "Query complete 00:00:00.080".

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?

The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL code:

```
1
2 CREATE unique INDEX idx_passport_number
3 ON passengers (passport_number);
4
5 SELECT indexname, indexdef
6 FROM pg_indexes
7 WHERE tablename = 'passengers';
8
9 SELECT *
10 FROM passengers;
```

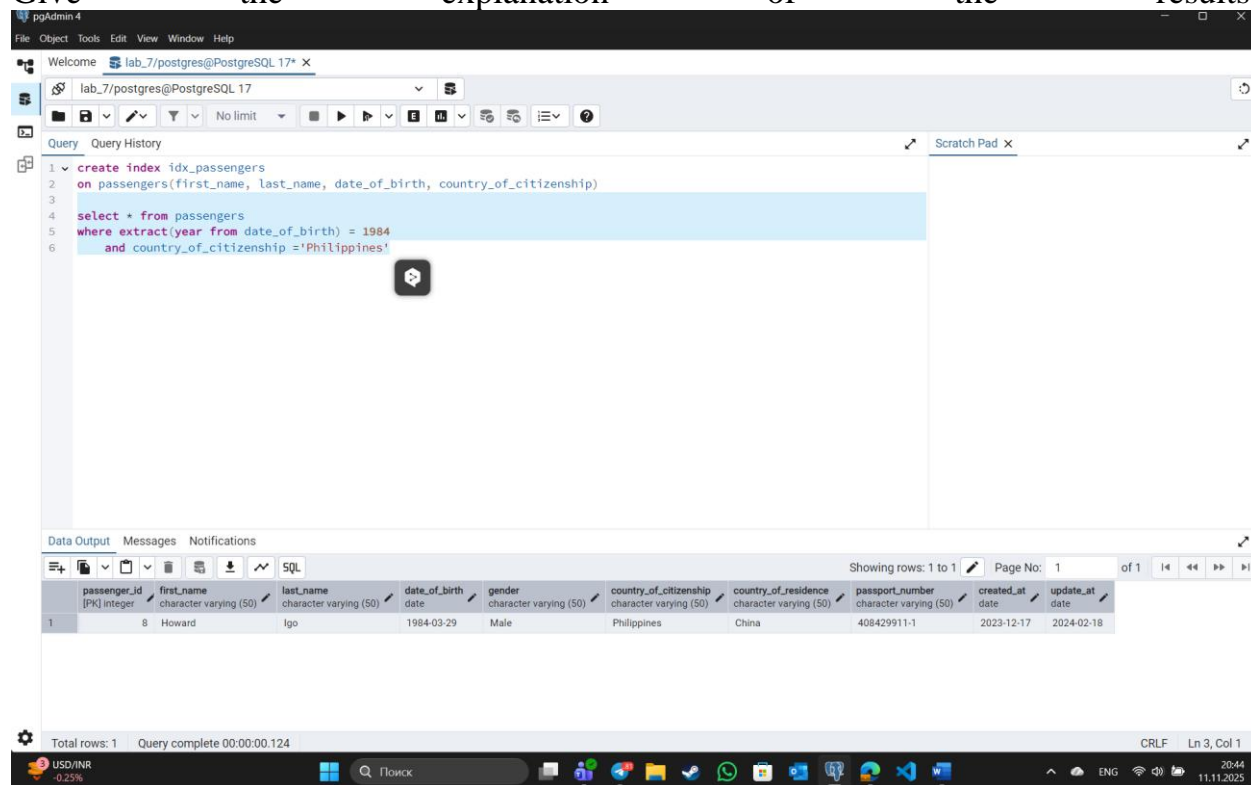
The Data Output tab shows the following error message:

```
ERROR: duplicate key value violates unique constraint "idx_passport_number"
Key (passport_number)=(382385347-3) already exists.

SQL state: 23505
Detail: Key (passport_number)=(382385347-3) already exists.
```

The status bar at the bottom indicates "Total rows: 4" and "Query complete 00:00:00.052".

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. The Query editor contains the following SQL code:

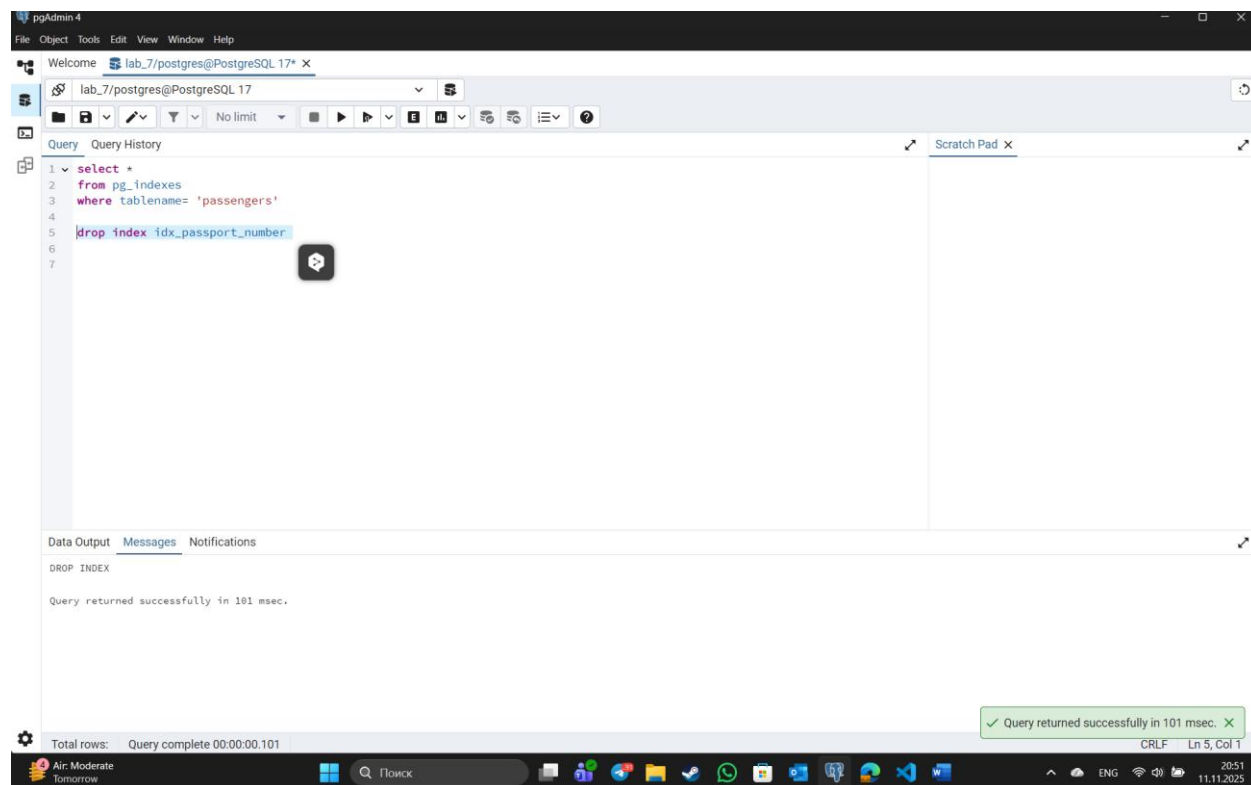
```
1 create index idx_passengers
2 on passengers(first_name, last_name, date_of_birth, country_of_citizenship)
3
4 select * from passengers
5 where extract(year from date_of_birth) = 1984
6 and country_of_citizenship = 'Philippines'
```

The Data Output tab shows the results of the query:

passenger_id [PK] integer	first_name character varying (50)	last_name character varying (50)	date_of_birth date	gender character varying (50)	country_of_citizenship character varying (50)	country_of_residence character varying (50)	passport_number character varying (50)	created_at date	update_at date	
1	8	Howard	Igo	1984-03-29	Male	Philippines	China	408429911-1	2023-12-17	2024-02-18

The status bar at the bottom indicates: Total rows: 1, Query complete 00:00:00.124.

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



The screenshot shows the pgAdmin 4 interface. The Query editor contains the following SQL code:

```
1 select *
2 from pg_indexes
3 where tablename= 'passengers'
4
5 drop index idx_passport_number
6
7
```

The Messages tab shows the following message:

```
DROP INDEX
Query returned successfully in 101 msec.
```

The status bar at the bottom indicates: Total rows: Query complete 00:00:00.101. A green notification box at the bottom right says: "Query returned successfully in 101 msec."