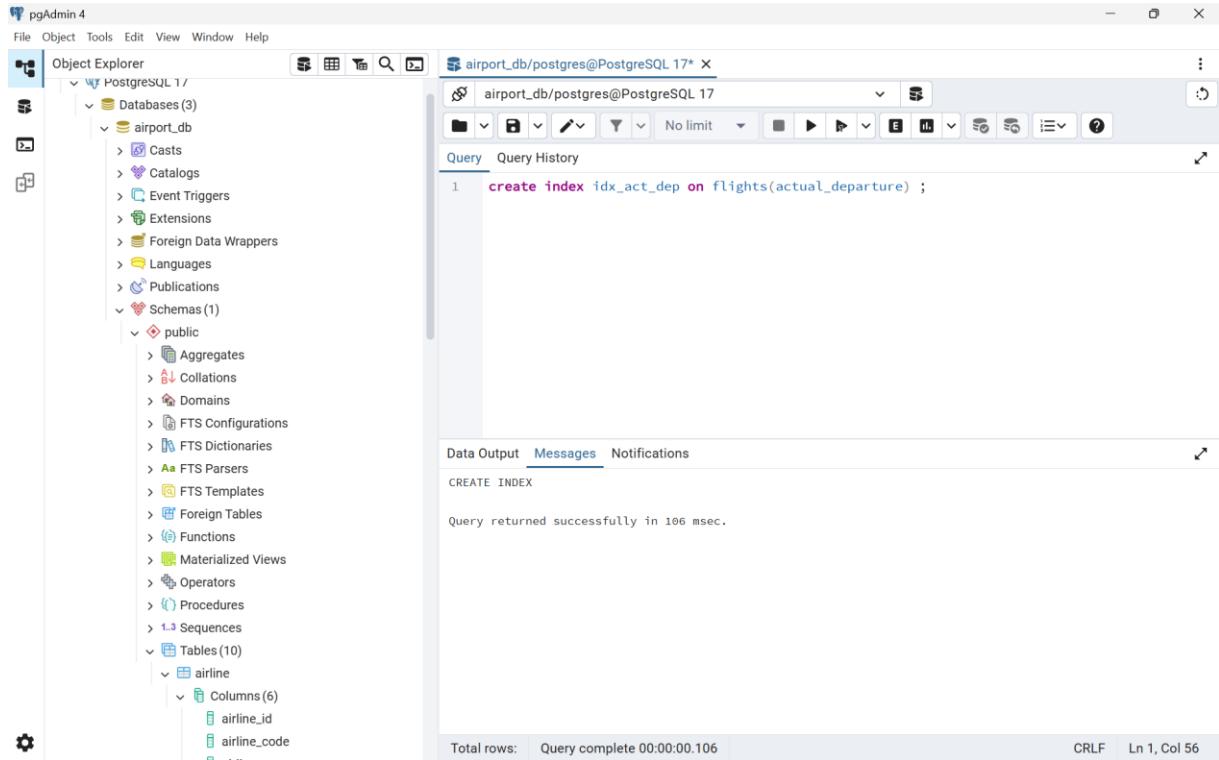


Laboratory work 7

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



The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the PostgreSQL 17 database structure. The right pane is the Query Editor, showing the command to create an index:

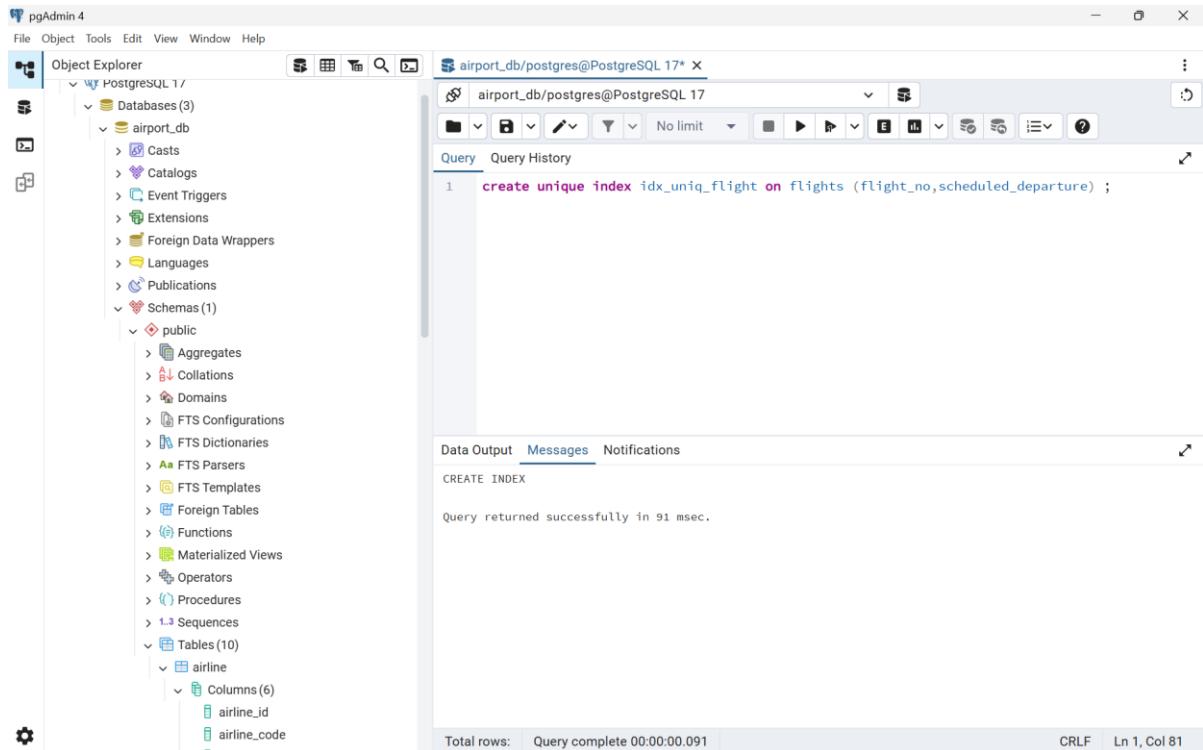
```
create index idx_act_dep on flights(actual_departure);
```

The Messages tab shows the result of the query:

CREATE INDEX
Query returned successfully in 106 msec.

Total rows: Query complete 00:00:00.106 CRLF Ln 1, Col 56

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



The screenshot shows the pgAdmin 4 interface. The left pane is the Object Explorer, displaying the PostgreSQL 17 database structure. The right pane is the Query Editor, showing the command to create a unique index:

```
create unique index idx_uniq_flight on flights (flight_no,scheduled_departure);
```

The Messages tab shows the result of the query:

CREATE INDEX
Query returned successfully in 91 msec.

Total rows: Query complete 00:00:00.091 CRLF Ln 1, Col 81

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

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer pane displays the database structure for 'airport_db'. In the center, the main window shows a query editor with the following SQL command:

```
create index idx_airport on flights(departure_airport_id, arrival_airport_id);
```

Below the query, the Data Output tab shows the result of the query execution:

CREATE INDEX

Query returned successfully in 93 msec.

Total rows: Query complete 00:00:00.093 CRLF Ln 1, Col 80

A green message bar at the bottom right indicates: ✓ Query returned successfully in 93 msec. X

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

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer pane displays the database structure for 'airport_db'. In the center, the main window shows a query editor with the following SQL command:

```
explain analyze
select * from flights
where departure_airport_id = 3 and arrival_airport_id = 7;
```

Below the query, the Data Output tab shows the result of the query execution, including the query plan:

Showing rows: 1 to 5 Page No: 1 of 1

QUERY PLAN
text
1 Seq Scan on flights (cost=0.00..36.00 rows=2 width=63) (actual time=0.280..0.394 rows=1 loops=..)
2 Filter: ((departure_airport_id = 3) AND (arrival_airport_id = 7))
3 Rows Removed by Filter: 999
4 Planning Time: 1.072 ms
5 Execution Time: 0.423 ms

Total rows: 5 Query complete 00:00:00.201 CRLF Ln 4, Col 1

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer tree displays database objects like Constraints, Indexes, RLS Policies, Rules, Triggers, booking_flight, and flights. The flights node is expanded, showing columns such as flight_id, flight_no, scheduled_departure, scheduled_arrival, departure_airport_id, arrival_airport_id, departing_gate, arriving_gate, airline_id, status, and actual_departure. In the center, a query editor window titled 'airport_db/postgres@PostgreSQL 17*' contains the following SQL code:

```
1 explain analyze
2 select * from flights
3 where departure_airport_id = 3 and arrival_airport_id = 7;
```

Below the query editor is a 'Data Output' tab showing the results of the query. The results table has one row with the following data:

flight_id	flight_no	scheduled_departure	scheduled_arrival	departure_airport_id	arrival_airport_id	departing_gate	arriving_gate	airline_id	status	actual_departure
1	AA123	2023-10-15 08:00:00	2023-10-15 09:00:00	3	7	A1	B2	AA	On Time	2023-10-15 08:00:00

The 'Data Output' tab also includes a 'QUERY PLAN' section with the following details:

Step	Operation	Details
1	Bitmap Heap Scan on flights	(cost=4.30..10.47 rows=2 width=63) (actual time=0.401..0.402 rows=1 loops=1)
2	Recheck Cond:	((departure_airport_id = 3) AND (arrival_airport_id = 7))
3	Heap Blocks:	exact=1
4	-> Bitmap Index Scan on idx_airport_pair	(cost=0.00..4.29 rows=2 width=0) (actual time=0.364..0.364 rows=1 loops=1)
5	Index Cond:	((departure_airport_id = 3) AND (arrival_airport_id = 7))
6	Planning Time:	0.264 ms
7	Execution Time:	0.447 ms

At the bottom of the pgAdmin interface, status messages indicate 'Total rows: 7' and 'Query complete 00:00:00.130'. The status bar also shows 'CRLF' and 'Ln 4, Col 1'.

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

```

explain analyze
select * from flights where departure_airport_id = 1 and arrival_airport_id = 1;

```

QUERY PLAN

- 1 Bitmap Heap Scan on flights (cost=4.31..12.95 rows=3 width=63) (actual time=0.794..0.803 rows=3 loops=1)
 - 2 Recheck Cond: ((departure_airport_id = 1) AND (arrival_airport_id = 1))
 - 3 Heap Blocks: exact=3
 - 4 -> Bitmap Index Scan on idx_airport (cost=0.00..4.31 rows=3 width=0) (actual time=0.126..0.126 rows=3 loops=1)
 - 5 Index Cond: ((departure_airport_id = 1) AND (arrival_airport_id = 1))
 - 6 Planning Time: 0.326 ms
 - 7 Execution Time: 2.658 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`

```

create unique index idx_passport_un on passengers(passport_number);

```

CREATE INDEX

Query returned successfully in 132 msec.

Check

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer, which lists the PostgreSQL 17 database 'airport_db'. Under 'Tables' (10), there is a 'airline' table with columns: airline_id, airline_code, and others. The main window shows a query in the 'Query' tab:

```
1 select indexname, indexdef from pg_indexes where tablename = 'passengers';
```

The results table shows three indexes:

indexname	indexdef
passengers_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
uq_passport_number	CREATE UNIQUE INDEX uq_passport_number ON public.passengers USING btree (passport_number)
idx_passport_uni	CREATE UNIQUE INDEX idx_passport_uni ON public.passengers USING btree (passport_number)

Total rows: 3 Query complete 00:00:00.408 CRLF Ln 1, Col 22

Insert 2 passengers

The screenshot shows the pgAdmin 4 interface. The Object Explorer lists the 'airport_db' database. In the main window, a query is being run in the 'Query' tab:

```
1 insert into passengers
2 (passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship,
3 country_of_residence, passport_number, created_at, update_at)
4 values
5 (201, 'Kuralay', 'Kassym', '2001-06-15', 'female', 'Kazakhstan', 'Kazakhstan', '4657385', '2024
6 (202, 'Dana', 'Kairat', '1998-02-20', 'female', 'Kazakhstan', 'Kazakhstan', '4657385', '2024
```

An error message is displayed in the 'Messages' tab:

ERROR: повторяющееся значение ключа нарушает ограничение уникальности "uq_passport_number"
Ключ "(passport_number)=(4657385)" уже существует.

OШИБКА: повторяющееся значение ключа нарушает ограничение уникальности "uq_passport_number"
SQL state: 23505
Detail: Ключ "(passport_number)=(4657385)" уже существует.

Total rows: 200 Query complete 00:00:00.117 CRLF Ln 7, Col 1

The unique index allows only one unique passport number per passenger, when I inserted 2 new passengers, it gave an error, because these 2 passengers have the same passport number. This happens because unique index prevents duplicate passport numbers in the table.

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. On the left is the Object Explorer pane, which displays the database structure. In the center is the main query editor window. A message bar at the bottom right indicates "Query returned successfully in 109 msec.".

```
idx_passengerinfo ON passengers(first_name, last_name, date_of_birth, country_of_citizenship)
```

CREATE INDEX

Query returned successfully in 109 msec.

Total rows: Query complete 00:00:00.109 CRLF Ln 1, Col 92

✓ Query returned successfully in 109 msec. X

```

explain analyze
select * from passengers where country_of_citizenship = 'Philippines' and date_of_birth between '1984-01-01' and '1984-12-31';

```

QUERY PLAN

1	Seq Scan on passengers (cost=0.00..6.50 rows=1 width=64) (actual time=0.041..0.102 rows=1 loops=1)
2	Filter: ((date_of_birth >= '1984-01-01'::date) AND (date_of_birth <= '1984-12-31'::date) AND ((country_of_citizenship)::text = 'Philippines'::text))
3	Rows Removed by Filter: 199
4	Planning Time: 1.491 ms
5	Execution Time: 0.132 ms

The query used seq scan , so it checked every row in the table to find this passengers. It didn't use index that was created,because index is used from left to right,starting with the first columns that were defined in the index,also the table is small,so it would be faster than using the index , to scan all rows.

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

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Servers (1)
PostgreSQL 17
Databases (3)
airport_db
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (10)
airline
Columns (6)
airline_id

airport_db/postgres@PostgreSQL 17*

Query Query History

```
1 select * from pg_indexes where tablename = 'passengers' ;
```

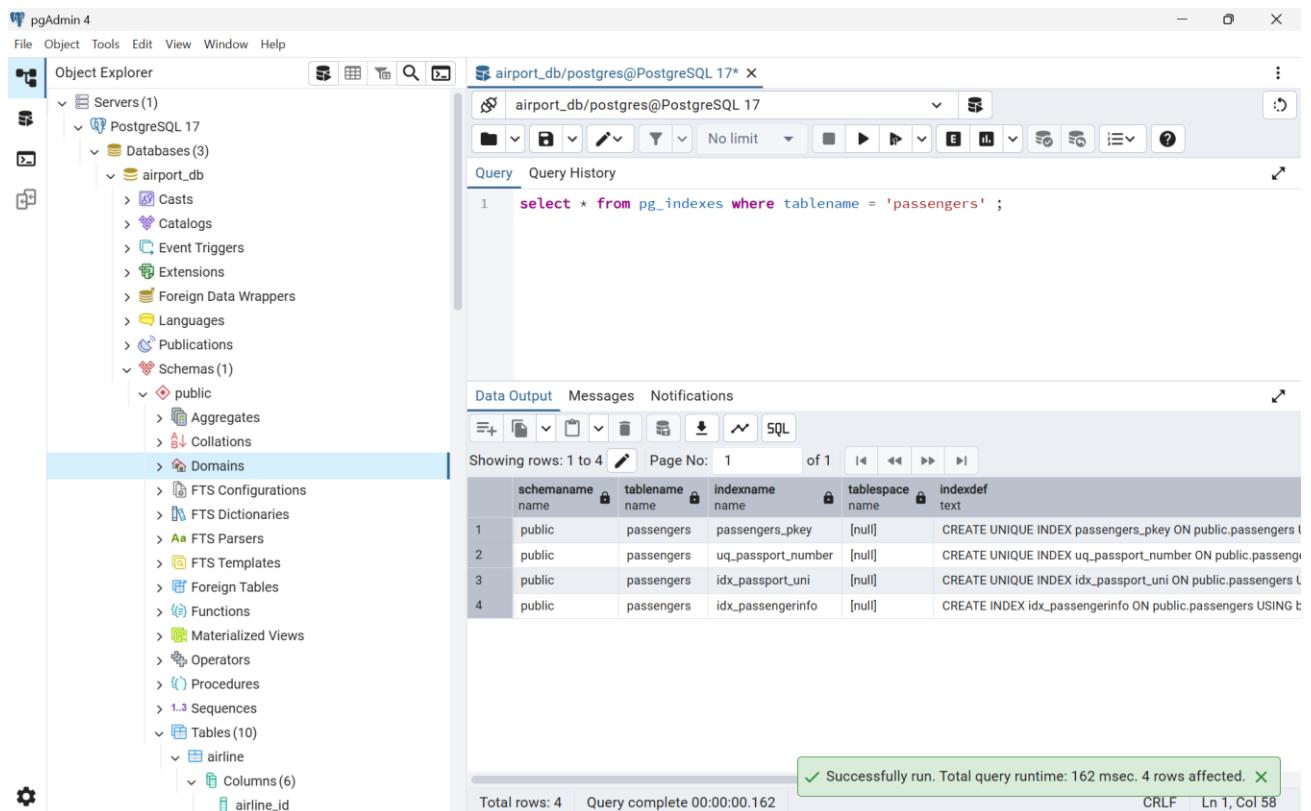
Data Output Messages Notifications

Showing rows: 1 to 4 Page No: 1 of 1

	schemaname	tablename	indexname	tablespace	indexdef
1	public	passengers	passengers_pkey	[null]	CREATE UNIQUE INDEX passengers_pkey ON public.passengers
2	public	passengers	uq_passport_number	[null]	CREATE UNIQUE INDEX uq_passport_number ON public.passengerinfo
3	public	passengers	idx_passport_uni	[null]	CREATE UNIQUE INDEX idx_passport_uni ON public.passengerinfo
4	public	passengers	idx_passengerinfo	[null]	CREATE INDEX idx_passengerinfo ON public.passengerinfo USING btree

Successfully run. Total query runtime: 162 msec. 4 rows affected.

Total rows: 4 Query complete 00:00:00.162 CRLF Ln 1, Col 58



pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

Servers (1)
PostgreSQL 17
Databases (3)
airport_db
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (10)
airline
Columns (6)
airline_id

airport_db/postgres@PostgreSQL 17*

Query Query History

```
1 drop index idx_passport_uni;
2 drop index idx_passengerinfo ;
```

Data Output Messages Notifications

DROP INDEX

Query returned successfully in 122 msec.

✓ Query returned successfully in 122 msec.

Total rows: 0 Query complete 00:00:00.122 CRLF Ln 2, Col 31

