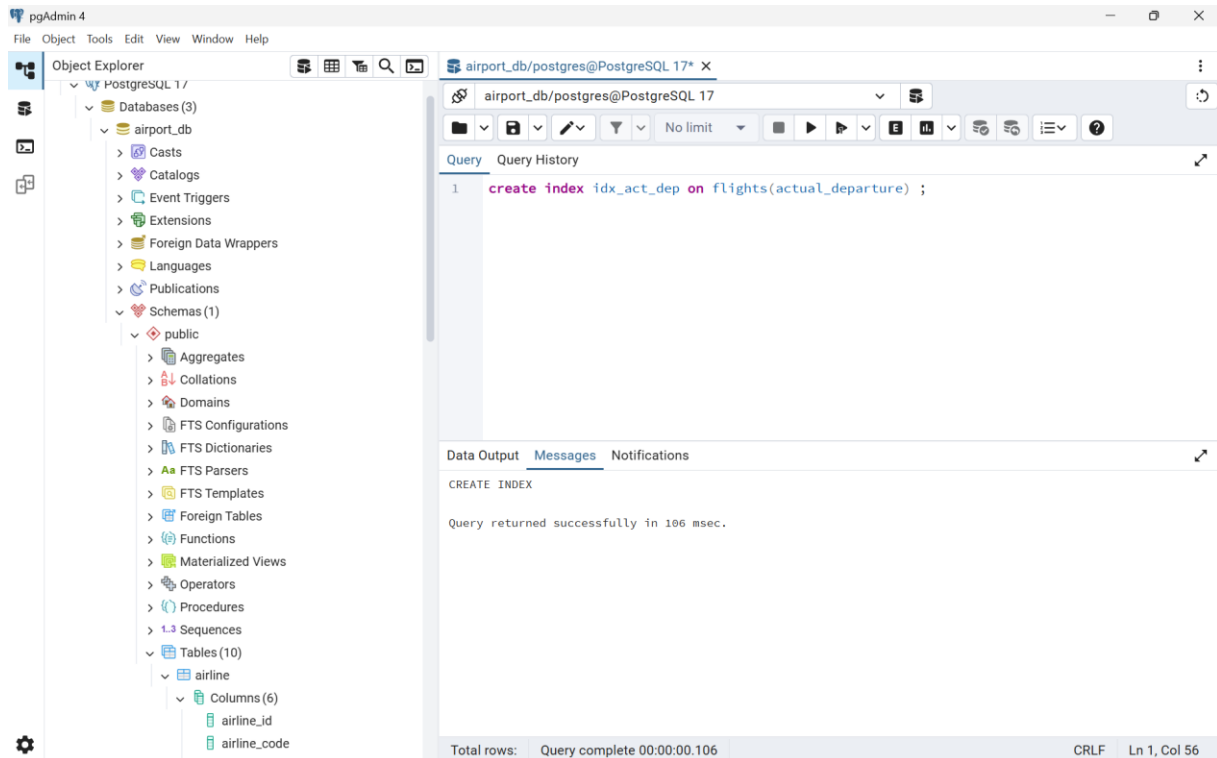
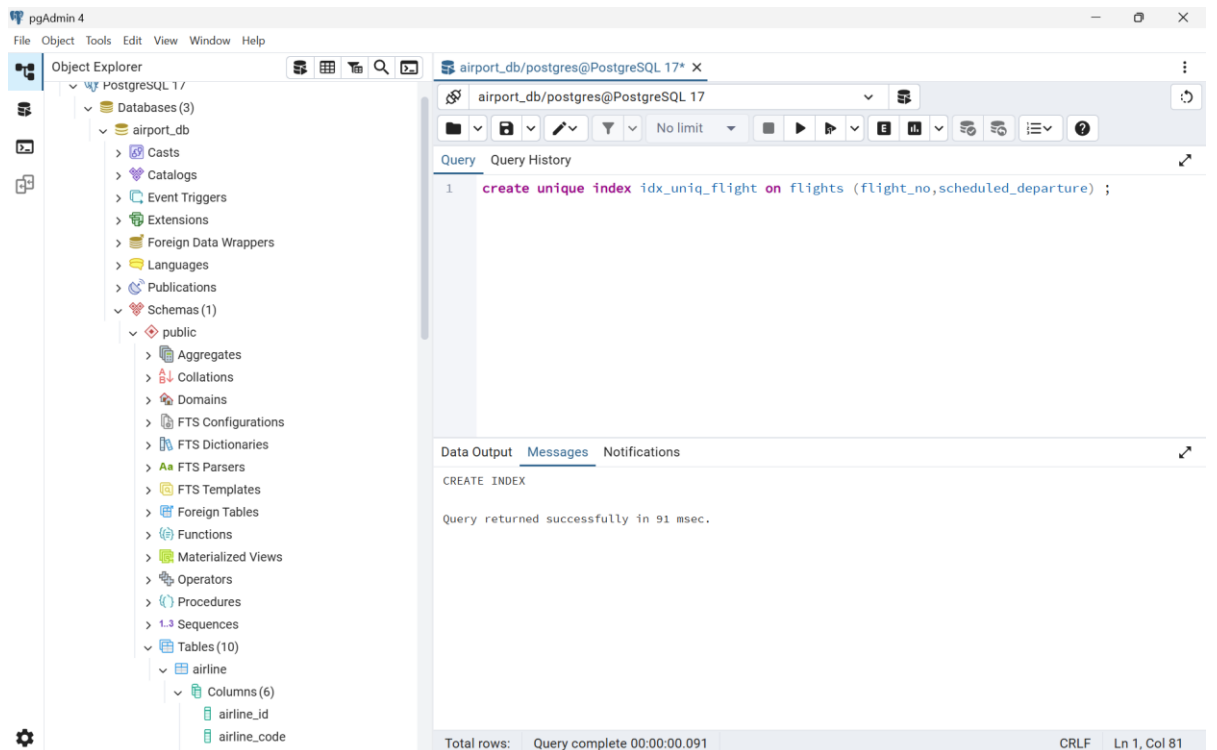


Laboratory work 7

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



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



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 displays the database structure: PostgreSQL 17 > Databases (3) > airport_db > Schemas (1) > public > Tables (10) > airline > Columns (6). The main query editor shows the following SQL command:

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

The Messages tab at the bottom shows the execution result:

```
CREATE INDEX  
  
Query returned successfully in 93 msec.
```

A green status bar at the bottom right indicates: "Query returned successfully in 93 msec." The status bar also shows "Total rows: Query complete 00:00:00.093" and "CRLF Ln 1, Col 80".

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

The screenshot shows the pgAdmin 4 interface with the same database structure as before. The main query editor shows the following SQL command:

```
1 explain analyze  
2 select * from flights where actual_departure > '2025-11-10' ;
```

The Messages tab at the bottom shows the execution result, including the query plan:

```
QUERY PLAN  
text  
1  Index Scan using idx_act_dep on flights (cost=0.28..8.29 rows=1 width=63) (actual time=0.005..0.006 rows=0 loops=...  
2  Index Cond: (actual_departure > '2025-11-10':date)  
3  Planning Time: 6.988 ms  
4  Execution Time: 0.040 ms
```

The status bar at the bottom shows "Total rows: 4" and "Query complete 00:00:00.104". The status bar also shows "CRLF Ln 2, Col 62".

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

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure, including the 'public' schema and the 'airline' table. The main pane shows a SQL query: `create unique index idx_passport_uni on passengers(passport_number);`. The 'Execute script' button is highlighted. Below the query, the 'Data Output' tab shows the message: 'CREATE INDEX' and 'Query returned successfully in 132 msec.'.

Check

The screenshot shows the pgAdmin 4 interface. The main pane displays a SQL query: `select indexname, indexdef from pg_indexes where tablename = 'passengers';`. The 'Data Output' tab shows the results of the query, listing three indexes: 'passengers_pkey', 'uq_passport_number', and 'idx_passport_uni'. The status bar at the bottom indicates 'Total rows: 3' and 'Query complete 00:00:00.408'.

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

Insert 2 passengers

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer displays the database structure for 'airport_db', including tables like 'airline' and 'passengers'. The main pane shows a SQL query being executed: `insert into passengers (passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number, created_at, update_at) values (201, 'Kuralay', 'Kassym', '2001-06-15', 'female', 'Kazakhstan', 'Kazakhstan', '4657385', '2024-02-20', '2024-02-20');`. The 'Data Output' tab shows an error message: `ERROR: повторяющееся значение ключа нарушает ограничение уникальности "uq_passport_number"`. The status bar at the bottom indicates 'Total rows: 200' and 'Query complete 00:00:00.117'.

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 the 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.

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.

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer shows the database structure: Servers (1) > PostgreSQL 17 > Databases (3) > airport_db > Schemas (1) > public > Tables (10) > airline > Columns (6) > airline_id. The main pane shows a query window for 'airport_db/postgres@PostgreSQL 17'. The query is: `select * from pg_indexes where tablename = 'passengers' ;`. The Data Output tab shows the results of the query, which are 4 rows. A green message bar at the bottom indicates: 'Successfully run. Total query runtime: 162 msec. 4 rows affected.'

	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.passengers
3	public	passengers	idx_passport_uni	[null]	CREATE UNIQUE INDEX idx_passport_uni ON public.passengers
4	public	passengers	idx_passengerinfo	[null]	CREATE INDEX idx_passengerinfo ON public.passengers

The screenshot shows the pgAdmin 4 interface. On the left, the Object Explorer shows the database structure: Servers (1) > PostgreSQL 17 > Databases (3) > airport_db > Schemas (1) > public > Tables (10) > airline > Columns (6) > airline_id. The main pane shows a query window for 'airport_db/postgres@PostgreSQL 17'. The query is: `drop index idx_passport_uni ;` and `drop index idx_passengerinfo ;`. The Messages tab shows the results of the query, which are 2 rows. A green message bar at the bottom indicates: 'Query returned successfully in 122 msec.'

	Message
1	DROP INDEX
2	Query returned successfully in 122 msec.