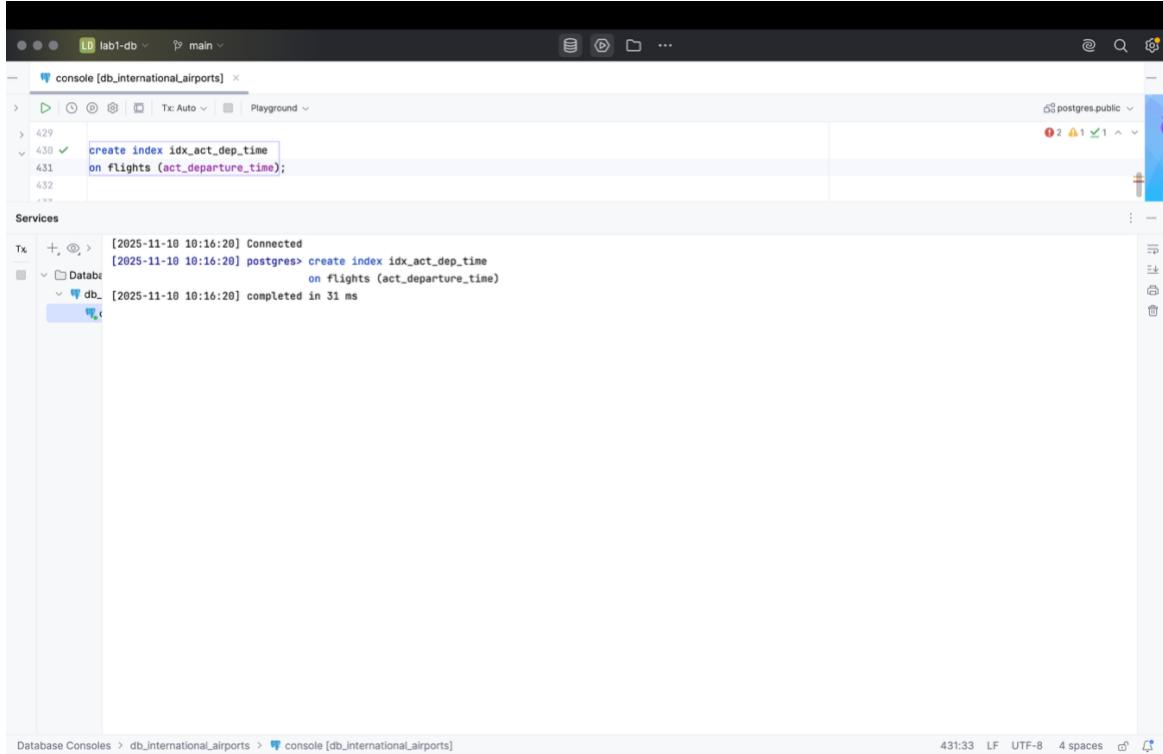


Lab 7

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



The screenshot shows a DBeaver database console for the 'db_international_airports' database. The current tab is 'console [db_international_airports]'. The query window contains the following SQL command:

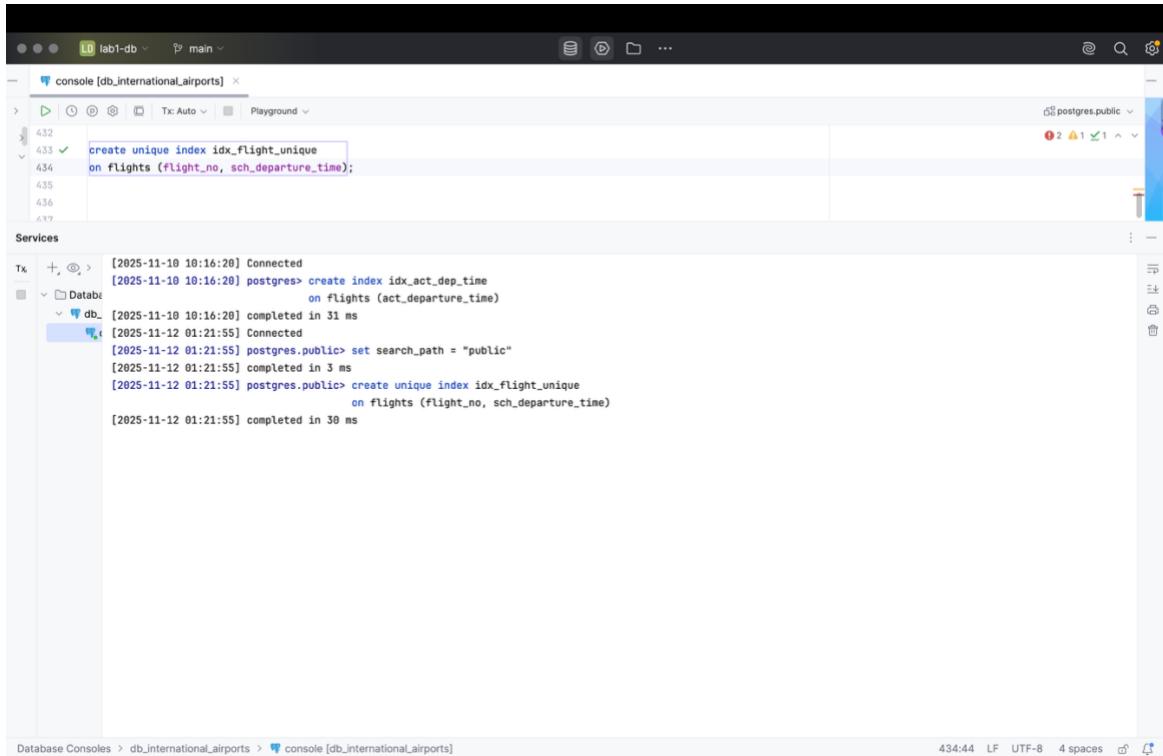
```
429
430 ✓ create index idx_act_dep_time
431 on flights (act_departure_time);
432
```

The execution log shows the following output:

```
[2025-11-10 10:16:20] Connected
[2025-11-10 10:16:20] postgres> create index idx_act_dep_time
[2025-11-10 10:16:20] on flights (act_departure_time)
[2025-11-10 10:16:20] completed in 31 ms
```

The status bar at the bottom indicates the session is connected to 'lab1-db' with 'main' selected, and the encoding is 'UTF-8'.

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



The screenshot shows a DBeaver database console for the 'db_international_airports' database. The current tab is 'console [db_international_airports]'. The query window contains the following SQL command:

```
432
433 ✓ create unique index idx_flight_unique
434 on flights (flight_no, sch_departure_time);
435
436
```

The execution log shows the following output:

```
[2025-11-10 10:16:20] Connected
[2025-11-10 10:16:20] postgres> create index idx_act_dep_time
[2025-11-10 10:16:20] on flights (act_departure_time)
[2025-11-10 10:16:20] completed in 31 ms
[2025-11-12 01:21:55] Connected
[2025-11-12 01:21:55] postgres.public> set search_path = "public"
[2025-11-12 01:21:55] completed in 3 ms
[2025-11-12 01:21:55] postgres.public> create unique index idx_flight_unique
[2025-11-12 01:21:55] on flights (flight_no, sch_departure_time)
[2025-11-12 01:21:55] completed in 30 ms
```

The status bar at the bottom indicates the session is connected to 'lab1-db' with 'main' selected, and the encoding is 'UTF-8'.

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

The screenshot shows a PostgreSQL database console interface. The top bar indicates the session is connected to 'lab1-db' on the 'main' database. The current query in the main pane is:

```
create index idx_airport_route
on flights (departing_airport_id, arriving_airport_id);
```

The 'Services' panel on the left shows the transaction history:

- [2025-11-10 10:16:20] completed in 31 ms
- [2025-11-12 01:21:55] Connected
- [2025-11-12 01:21:55] postgres.public> set search_path = "public"
- [2025-11-12 01:21:55] completed in 3 ms
- [2025-11-12 01:21:55] postgres.public> create unique index idx_flight_unique
on flights (flight_no, sch_departure_time)
[2025-11-12 01:21:55] completed in 30 ms
- [2025-11-12 01:24:19] postgres.public> create index idx_airport_route
on flights (departure_airport_id, arrival_airport_id)
- [2025-11-12 01:24:19] [42703] ERROR: column "departure_airport_id" does not exist
- [2025-11-12 01:24:38] postgres.public> create index idx_airport_route
on flights (departing_airport_id, arriving_airport_id)
- [2025-11-12 01:24:38] completed in 16 ms

The bottom status bar shows the session details: Database Consoles > db_international_airports > console [db_international_airports].

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

The screenshot displays two vertically stacked pgAdmin database consoles, both connected to the 'lab1-db' database and the 'main' schema.

Top Console (Without Index):

- Database Explorer:** Shows the 'flights' table with 13 columns: flight_id, sch_departure_time, sch_arrival_time, and others.
- Query:** `explain analyze select* from flights where departing_airport_id = 133 and arriving_airport_id = 125;`
- Result:** A query plan showing a Seq Scan on the 'flights' table. The plan details:
 - Seq Scan on flights (cost=0.00..11.93 rows=1 width=146) (actual time=0.033..0.101 rows=1 loops=1)
 - Filter: ((departing_airport_id = 133) AND (arriving_airport_id = 125))
 - Rows Removed by Filter: 394
 - Planning Time: 0.117 ms
 - Execution Time: 0.118 ms

Bottom Console (With Index):

- Database Explorer:** Shows the 'flights' table with 13 columns, including the newly created index 'idx_airport_route' on the 'departing_airport_id' and 'arriving_airport_id' columns.
- Query:** `create index idx_airport_route on flights (departing_airport_id, arriving_airport_id);` followed by the same query as the top console.
- Result:** A query plan showing an Index Scan using 'idx_airport_route' on the 'flights' table. The plan details:
 - Index Scan using idx_airport_route on flights (cost=0.27..8.29 rows=1 width=146) (actual time=0..)
 - Index Cond: ((departing_airport_id = 133) AND (arriving_airport_id = 125))
 - Planning Time: 0.929 ms
 - Execution Time: 0.231 ms

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

The screenshot shows the pgAdmin interface with the following details:

- Database Explorer:** Shows the `db_international_airports` database with the `flights` table selected. The `columns` section lists `flight_id`, `sch_departure_time`, `sch_arrival_time`, `departing_airport_id`, and `arriving_airport_id`.
- Console:** The query entered is:


```
explain analyze
select flight_id, flight_no, departing_airport_id, arriving_airport_id
from flights
where departing_airport_id = 160 and arriving_airport_id = 193;
```
- Services - Query Plan:** The results show the execution plan:


```
1 Index Scan using idx_airport_route on flights (cost=0.27..8.29 rows=1 width=90) (actual time=0.108..0.110 rows=1 loops=1)
  2  Index Cond: ((departing_airport_id = 160) AND (arriving_airport_id = 193))
  3  Planning Time: 1.067 ms
  4  Execution Time: 0.165 ms
```
- Bottom Status:** 448:1 LF UTF-8 4 spaces

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 interface with the following details:

- Database Explorer:** Shows the `db_international_airports` database with the `flights` table selected. The `columns` section lists `flight_id`, `sch_departure_time`, `sch_arrival_time`, `departing_airport_id`, `arriving_airport_id`, `departing_gate`, `arriving_gate`, and `airline_id`.
- Console:** The query entered is:


```
create unique index idx_passport_unique
on passengers(passport_number);
```
- Services - Query Plan:** The results show the execution plan:


```
1 Index Scan using idx_airport_route on flights (cost=0.27..8.29 rows=1 width=90) (actual time=0.108..0.110 rows=1 loops=1)
  2  Index Cond: ((departing_airport_id = 160) AND (arriving_airport_id = 193))
  3  Planning Time: 1.067 ms
  4  Execution Time: 0.165 ms
```
- Bottom Status:** 454:1 LF UTF-8 4 spaces

Database Explorer

```
db_international_airports
  flights
    columns 13
      flight_id integer = nextval
      sch_departure_time timestamp
      sch_arrival_time timestamp
      departing_airport_id integer
      arriving_airport_id integer
      departing_gate text
      arriving_gate varchar(45)
      airline_id integer
      act_departure_time timestamp
      act_arrival_time timestamp
      created_at timestamp
      updated_at timestamp
      flight_no varchar(30)
```

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

Services

Database

indexname	indexdef
passenger_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
passenger_passport_number_key	CREATE UNIQUE INDEX passengers_passport_number_key ON public.passengers USING btree (passport_number)
u_people	CREATE UNIQUE INDEX u_people ON public.passengers USING btree (first_name, last_name)
idx_passport_unqive	CREATE UNIQUE INDEX idx_passport_unqive ON public.passengers USING btree (passport_number)

Database Consoles > db_international_airports > console [db_international_airports]

464:32 LF UTF-8 4 spaces ⌂ ⌂

Database Explorer

```
db_international_airports
  passengers
    insert into passengers(passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number)
    values ( passenger_id 449, first_name 'John', last_name 'Doe', date_of_birth '2006-01-28', gender 'Male', country_of_citizenship 'Georgia', country_of_residence 'Georgia', passport_number )
    insert into passengers(passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number)
    values ( passenger_id 451, first_name 'Jane', last_name 'Smith', date_of_birth '1987-05-13', gender 'Female', country_of_citizenship 'Singapore', country_of_residence 'China', passport_number )
```

[23505] ERROR: duplicate key value violates unique constraint "idx_passport_unqive"
Detail: Key (passport_number)=(-764) already exists.

Services

Database

indexname	indexdef
passenger_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
passenger_passport_number_key	CREATE UNIQUE INDEX passengers_passport_number_key ON public.passengers USING btree (passport_number)
u_people	CREATE UNIQUE INDEX u_people ON public.passengers USING btree (first_name, last_name)
idx_passport_unqive	CREATE UNIQUE INDEX idx_passport_unqive ON public.passengers USING btree (passport_number)

Database Consoles > db_international_airports > console [db_international_airports]

474:1 LF UTF-8 4 spaces ⌂ ⌂

We created a unique index for passport column ensuring that no duplicates can be added into the table passengers. This error displays that index we created is working. The first time we successfully inserted a new passenger but then when we tried to add a passenger with the same passport number as the previous one, the passport index is not letting us to do it.

7. Create an index for the Passengers table. Use for that first name, last name, date of birth, 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 explanation of the results.

The screenshot shows a PostgreSQL database interface with the following details:

- Console Tab:** Shows the command to create an index and a query to select passengers born in the Philippines in 1984.

```
470
471
472     create index idx_passenger_info
473     on passengers(first_name, last_name, date_of_birth, country_of_citizenship);
474
475 ✓   select * from passengers
476     where country_of_citizenship = 'Philippines'
477     and extract(year from date_of_birth) = 1984;
478
```
- Output Tab:** Shows the result of the query, which returns a single row of data:

passenger_id	first_name	last_name	date_of_birth	gender	country_of_citizenship	country_of_residence
453	Kate	McAddams	1984-01-28	Female	Philippines	India
- Services Tab:** Shows the passenger table structure and the data row.

The screenshot shows a PostgreSQL database console interface. The top bar displays the connection name 'lab1-db' and the schema 'main'. Below the bar, there are tabs for 'console [db_international_airports]', 'passengers [db_international_airports]', and 'flights [db_international_airports]'. The main area contains a code editor with the following SQL query:

```
470
471     create index idx_passenger_info
472     on passengers(first_name, last_name, date_of_birth, country_of_citizenship);
473     explain analyze
474     select * from passengers
475     where country_of_citizenship = 'Philippines'
476     and extract(year from date_of_birth) = 1984;
477
478
```

The 'explain analyze' command has been run, and the results are shown in the 'Services' panel under the 'db.' database. The 'QUERY PLAN' section shows the following details:

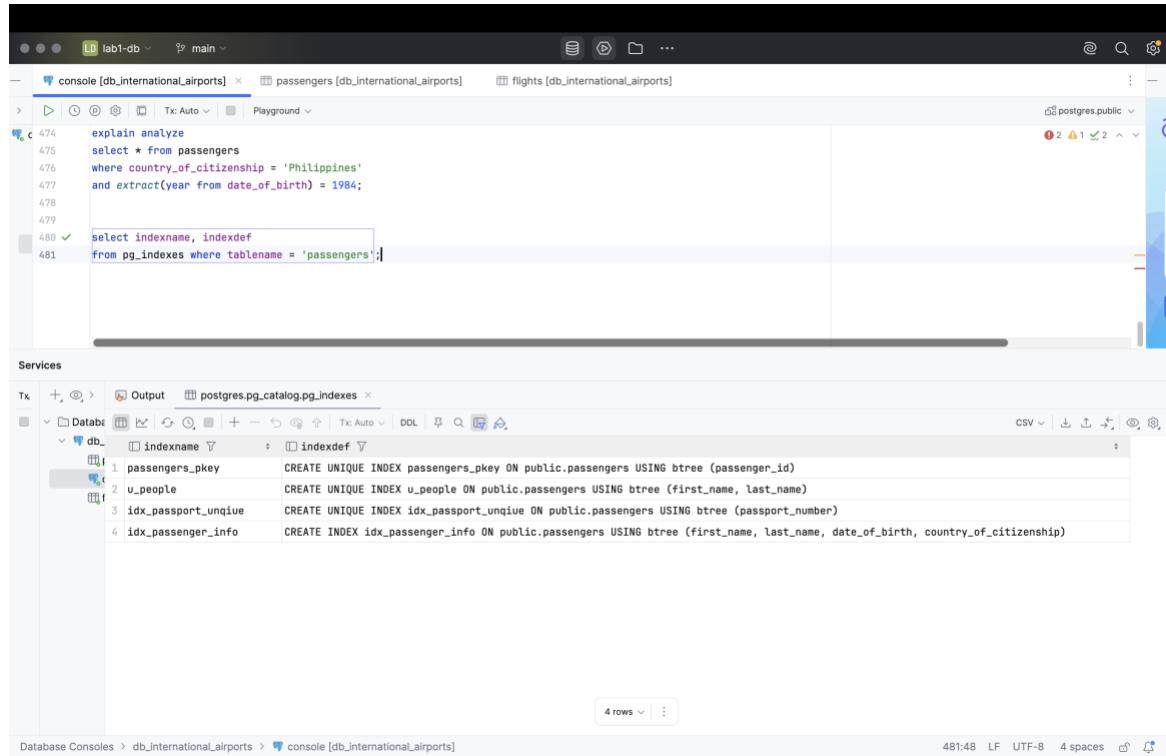
- Seq Scan on passengers (cost=0.00..14.88 rows=1 width=73) (actual time=0.220..0.221 rows=1 loops=1)
- Filter: (((country_of_citizenship)::text = 'Philippines'::text) AND (EXTRACT(year FROM date_of_birth) = '1984'::numeric))
- Rows Removed by Filter: 449

Planning Time: 0.287 ms
Execution Time: 0.244 ms

At the bottom of the interface, it says 'Database Consoles > db_international_airports > console [db_international_airports]' and shows the line number '478:1'.

As we can see here a query did not use a composite index, we created to search for a passenger who was born in Philippines in 1984. Probably because the filter does not use the first indexed columns but instead uses 'first_name', 'last_name'. Also, we used the function extract to find the year of birth so the query cannot use the index on that column.

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



The screenshot shows a PostgreSQL database interface with two panes. The top pane is a SQL console with the following session history:

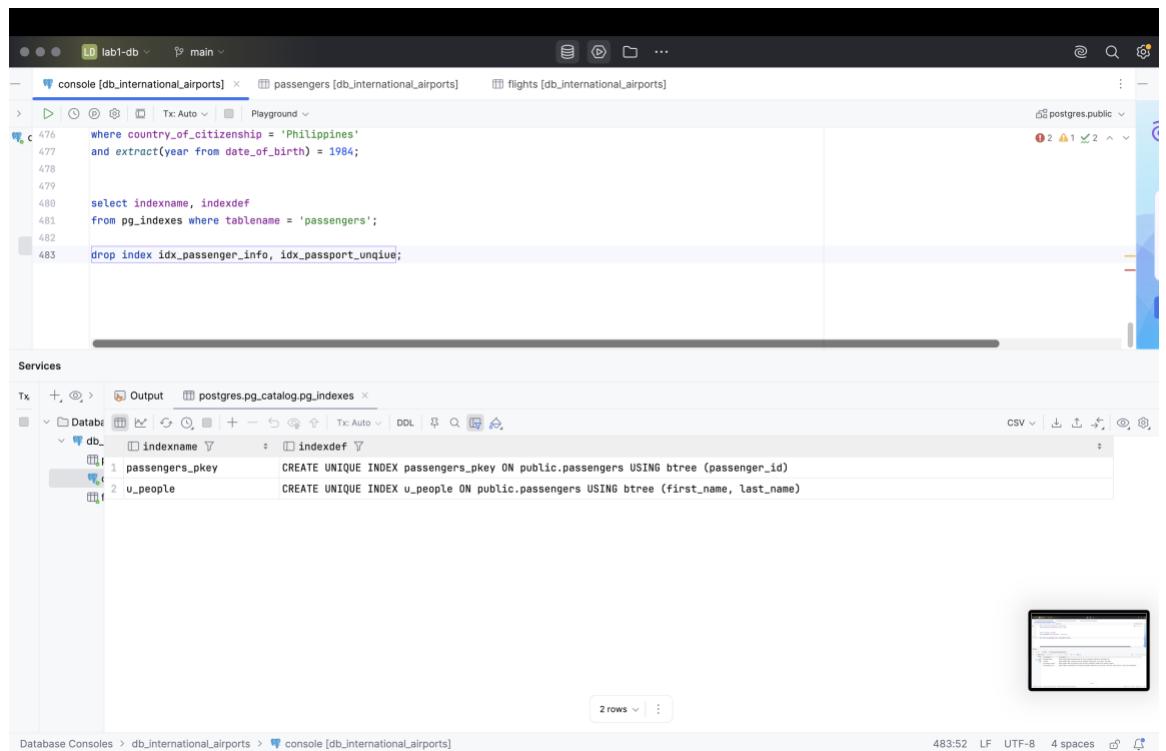
```

474 explain analyze
475 select * from passengers
476 where country_of_citizenship = 'Philippines'
477 and extract(year from date_of_birth) = 1984;
478
479
480 ✓ select indexname, indexdef
481 from pg_indexes where tablename = 'passenger';

```

The bottom pane shows the `pg_catalog.pg_indexes` table with the following data:

indexname	indexdef
passenger_pkey	CREATE UNIQUE INDEX passenger_pkey ON public.passenger USING btree (passenger_id)
u_people	CREATE UNIQUE INDEX u_people ON public.passenger USING btree (first_name, last_name)
idx_passport_unq	CREATE UNIQUE INDEX idx_passport_unq ON public.passenger USING btree (passport_number)
idx_passenger_info	CREATE INDEX idx_passenger_info ON public.passenger USING btree (first_name, last_name, date_of_birth, country_of_citizenship)



The screenshot shows the same PostgreSQL interface with the following session history:

```

476 where country_of_citizenship = 'Philippines'
477 and extract(year from date_of_birth) = 1984;
478
479
480 ✓ select indexname, indexdef
481 from pg_indexes where tablename = 'passenger';
482
483 drop index idx_passenger_info, idx_passport_unq;

```

The bottom pane shows the `pg_catalog.pg_indexes` table with the following data:

indexname	indexdef
passenger_pkey	CREATE UNIQUE INDEX passenger_pkey ON public.passenger USING btree (passenger_id)
u_people	CREATE UNIQUE INDEX u_people ON public.passenger USING btree (first_name, last_name)