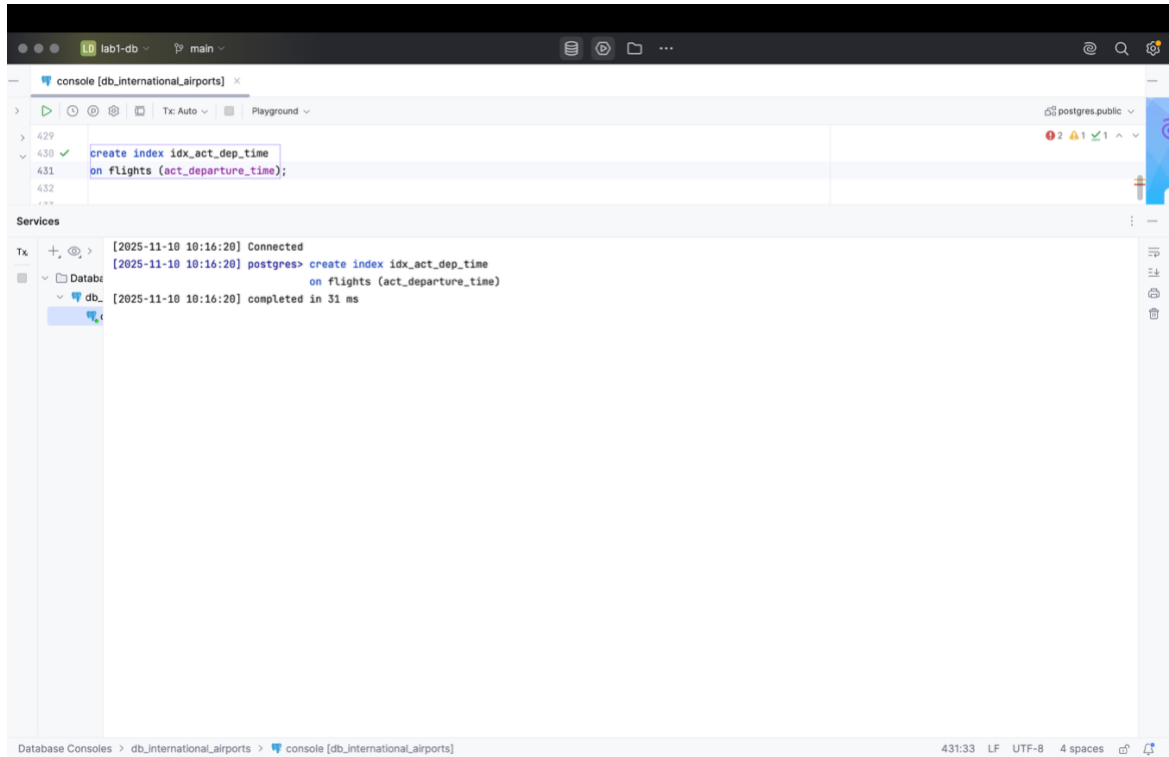


Lab 7

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

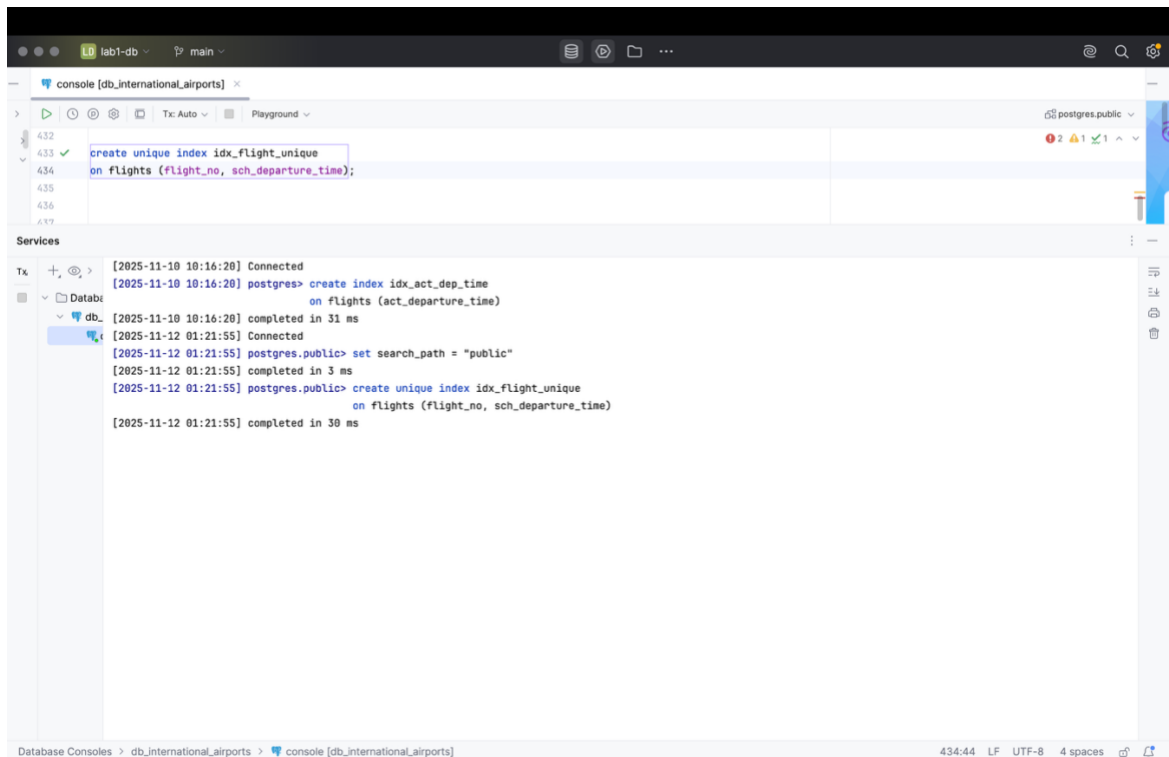


The screenshot shows a database console interface with a dark theme. The top bar indicates the current database is 'lab1-db' and the main view is 'main'. The console is titled 'console [db_internationalAirports]'. The SQL editor shows the following code:

```
create index idx_act_dep_time
on flights (act_departure_time);
```

The 'Services' panel on the right shows a list of transactions. The first transaction is a connection at [2025-11-10 10:16:20]. The second transaction is a successful execution of the SQL command at [2025-11-10 10:16:20], with the output 'postgres> create index idx_act_dep_time on flights (act_departure_time)'. The third transaction is a completion message at [2025-11-10 10:16:20] stating 'completed in 31 ms'.

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

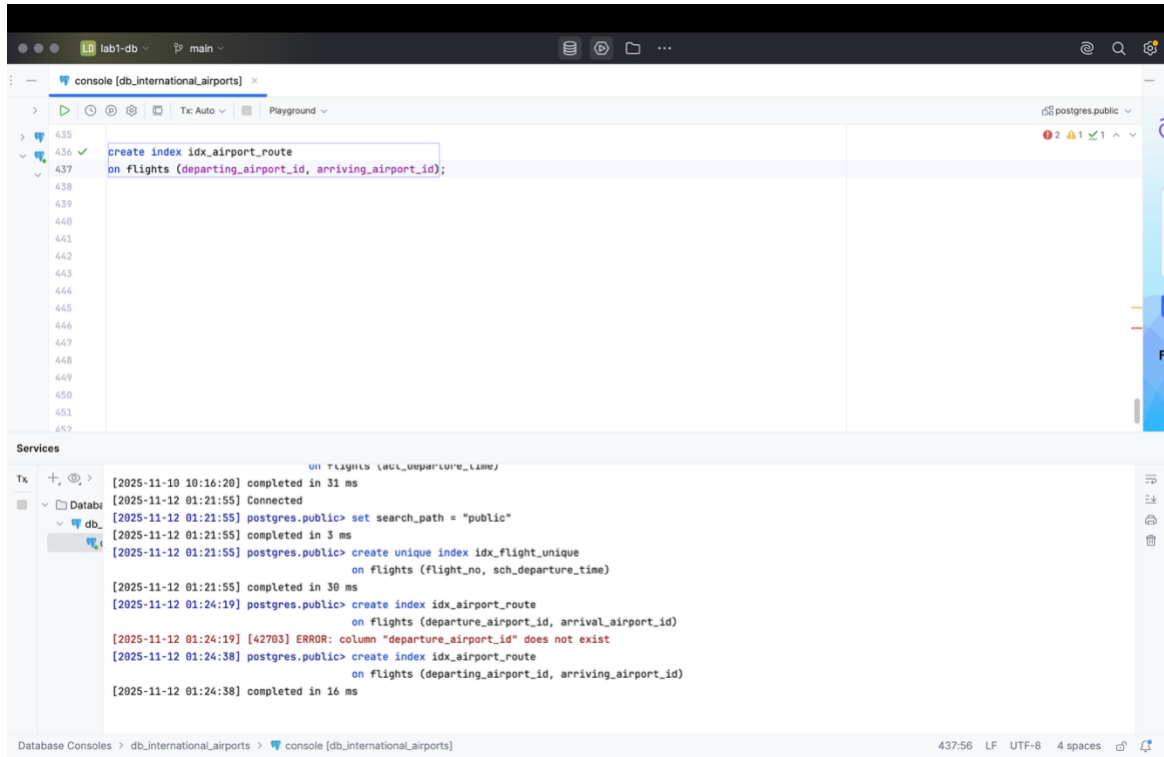


The screenshot shows the same database console interface. The SQL editor now shows the following code:

```
create unique index idx_flight_unique
on flights (flight_no, sch_departure_time);
```

The 'Services' panel on the right shows a list of transactions. The first transaction is a connection at [2025-11-10 10:16:20]. The second transaction is a successful execution of the SQL command at [2025-11-10 10:16:20], with the output 'postgres> create index idx_act_dep_time on flights (act_departure_time)'. The third transaction is a completion message at [2025-11-10 10:16:20] stating 'completed in 31 ms'. The fourth transaction is a connection at [2025-11-12 01:21:55]. The fifth transaction is a successful execution of the SQL command at [2025-11-12 01:21:55], with the output 'postgres> set search_path = "public"'. The sixth transaction is a successful execution of the SQL command at [2025-11-12 01:21:55], with the output 'postgres> create unique index idx_flight_unique on flights (flight_no, sch_departure_time)'. The seventh transaction is a completion message at [2025-11-12 01:21:55] stating 'completed in 30 ms'.

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



The screenshot shows a database console interface with a dark theme. The top bar indicates the database is 'lab1-db' and the current schema is 'main'. The console window is titled 'console [db_international_airports]'. It shows a SQL command being executed: `create index idx_airport_route on flights (departure_airport_id, arriving_airport_id);`. The command is highlighted in blue. Below the command, the 'Services' panel shows the execution log. The log includes the following entries:

- [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 current line is 437:56, the encoding is UTF-8, and there are 4 spaces.

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

The image displays two screenshots of a PostgreSQL database console interface, illustrating the performance difference between a sequential scan and an indexed query.

Top Screenshot (Without Index):

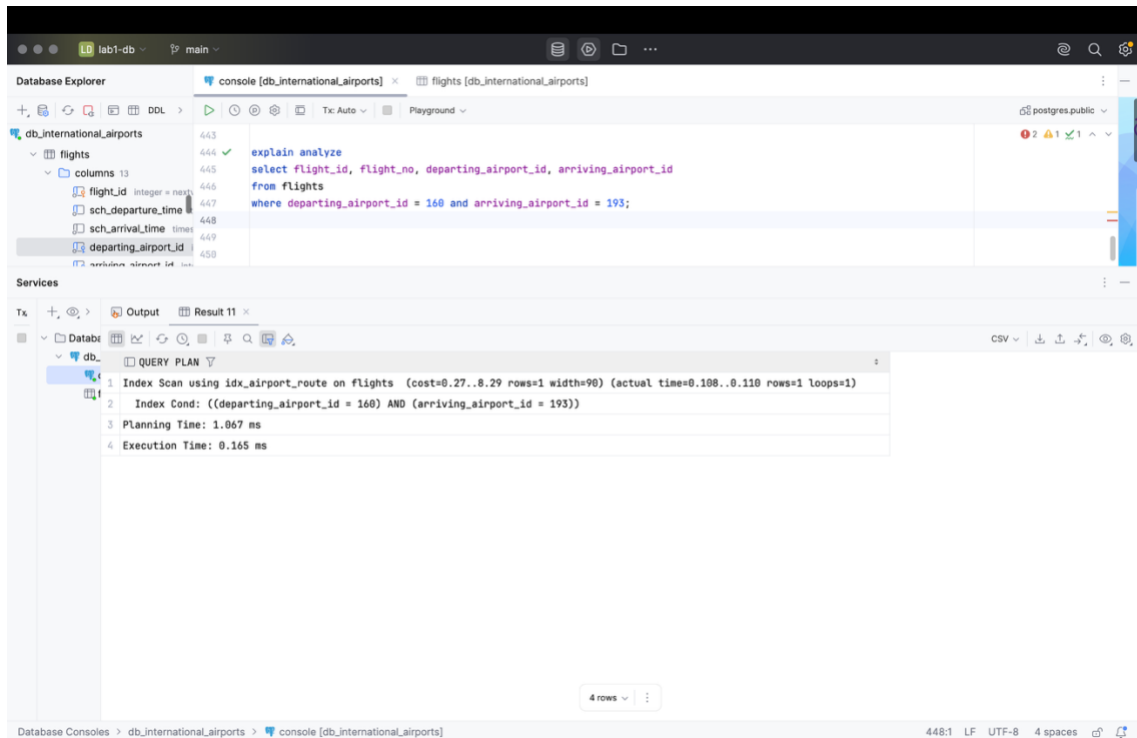
- Query:** `explain analyze select * from flights where departing_airport_id = 133 and arriving_airport_id = 125;`
- Query Plan:**
 - 1 Seq Scan on flights (cost=0.00..11.93 rows=1 width=146) (actual time=0.033..0.101 rows=1 loops=1)
 - 2 Filter: ((departing_airport_id = 133) AND (arriving_airport_id = 125))
 - 3 Rows Removed by Filter: 394
 - 4 Planning Time: 0.117 ms
 - 5 Execution Time: 0.118 ms

Bottom Screenshot (With Index):

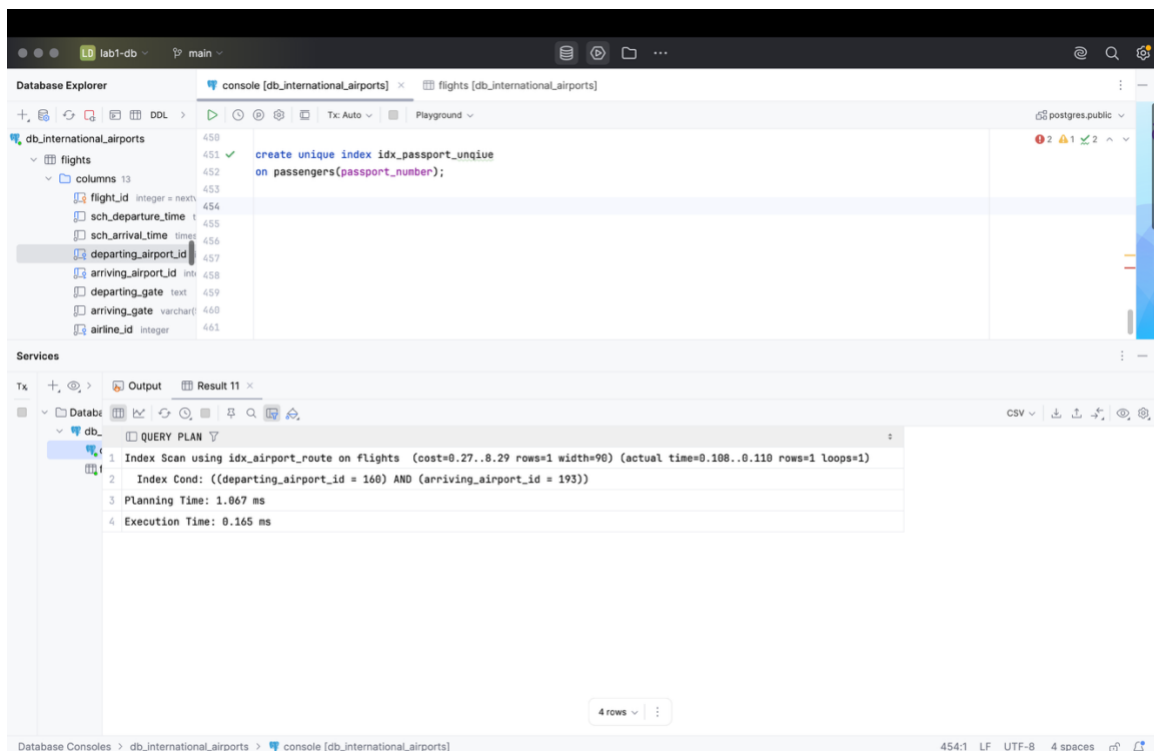
- DDL:** `create index idx_airport_route on flights (departing_airport_id, arriving_airport_id);`
- Query:** `explain analyze select * from flights where departing_airport_id = 133 and arriving_airport_id = 125;`
- Query Plan:**
 - 1 Index Scan using idx_airport_route on flights (cost=0.27..8.29 rows=1 width=146) (actual time=0.000..0.000 rows=1 loops=1)
 - 2 Index Cond: ((departing_airport_id = 133) AND (arriving_airport_id = 125))
 - 3 Planning Time: 0.929 ms
 - 4 Execution Time: 0.231 ms

The comparison shows that the indexed query (bottom) is significantly faster than the sequential scan (top), demonstrating the importance of indexing for query performance.

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



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.



lab1-db main

Database Explorer

console [db_internationalAirports] flights [db_internationalAirports]

db_internationalAirports

- flights
 - columns 13
 - flight_id integer = next 458
 - sch_departure_time 459
 - sch_arrival_time 460
 - departing_airport_id 461
 - arriving_airport_id 462
 - departing_gate text 463
 - arriving_gate varchar 464
 - airline_id integer 465
 - act_departure_time 466
 - act_arrival_time 467
 - created_at timestamp 468
 - updated_at timestamp 469
 - flight_no varchar(30) 470

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

Services

Output postgres.pg_catalog.pg_indexes

indexname	indexdef
passengers_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
passengers_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_unque	CREATE UNIQUE INDEX idx_passport_unque ON public.passengers USING btree (passport_number)

4 rows

Database Consoles > db_internationalAirports > console [db_internationalAirports]

464:32 LF UTF-8 4 spaces

lab1-db main

console [db_internationalAirports] passengers [db_internationalAirports] flights [db_internationalAirports]

```
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 764);

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 764);
```

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

Services

Output postgres.pg_catalog.pg_indexes

indexname	indexdef
passengers_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
passengers_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_unque	CREATE UNIQUE INDEX idx_passport_unque ON public.passengers USING btree (passport_number)

4 rows

Database Consoles > db_internationalAirports > console [db_internationalAirports]

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 database console interface with the following content:

```
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
```

Below the console, the 'Services' section shows the 'Output' for the query. The table 'postgres.public.passengers' is displayed with the following data:

passenger_id	first_name	last_name	date_of_birth	gender	country_of_citizenship	country_of_residence	
1	453	Kate	McAddams	1984-01-28	Female	Philippines	India

The status bar at the bottom indicates the database console is for 'db_international_airports' and shows the current line and column positions.

The screenshot shows a database console interface with a query editor at the top and a 'Services' panel at the bottom. The query editor contains the following SQL code:

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

The 'Services' panel shows the 'Output' tab with 'Result 26'. It displays a 'QUERY PLAN' for the executed query:

- 1 Seq Scan on passengers (cost=0.00..14.88 rows=1 width=73) (actual time=0.220..0.221 rows=1 loops=1)
- 2 Filter: (((country_of_citizenship)::text = 'Philippines'::text) AND (EXTRACT(year FROM date_of_birth) = '1984'::numeric))
- 3 Rows Removed by Filter: 449
- 4 Planning Time: 0.287 ms
- 5 Execution Time: 0.244 ms

At the bottom of the console, there is a status bar showing 'Database Consoles > db_international_airports > console [db_international_airports]' and '478:1 LF UTF-8 4 spaces'.

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 database console interface with the following SQL query entered:

```
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 = 'passengers';
```

The 'Services' panel at the bottom displays the results of the query in a table:

indexname	indexdef
1 passengers_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
2 u_people	CREATE UNIQUE INDEX u_people ON public.passengers USING btree (first_name, last_name)
3 idx_passport_unquie	CREATE UNIQUE INDEX idx_passport_unquie ON public.passengers USING btree (passport_number)
4 idx_passenger_info	CREATE INDEX idx_passenger_info ON public.passengers USING btree (first_name, last_name, date_of_birth, country_of_citizenship)

The status bar at the bottom indicates: Database Consoles > db_internationalAirports > console [db_internationalAirports] 481:48 LF UTF-8 4 spaces.

The screenshot shows the same database console interface with the following SQL query entered:

```
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 = 'passengers';
482
483 drop index idx_passenger_info, idx_passport_unquie;
```

The 'Services' panel at the bottom displays the results of the query in a table:

indexname	indexdef
1 passengers_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
2 u_people	CREATE UNIQUE INDEX u_people ON public.passengers USING btree (first_name, last_name)

The status bar at the bottom indicates: Database Consoles > db_internationalAirports > console [db_internationalAirports] 483:52 LF UTF-8 4 spaces.