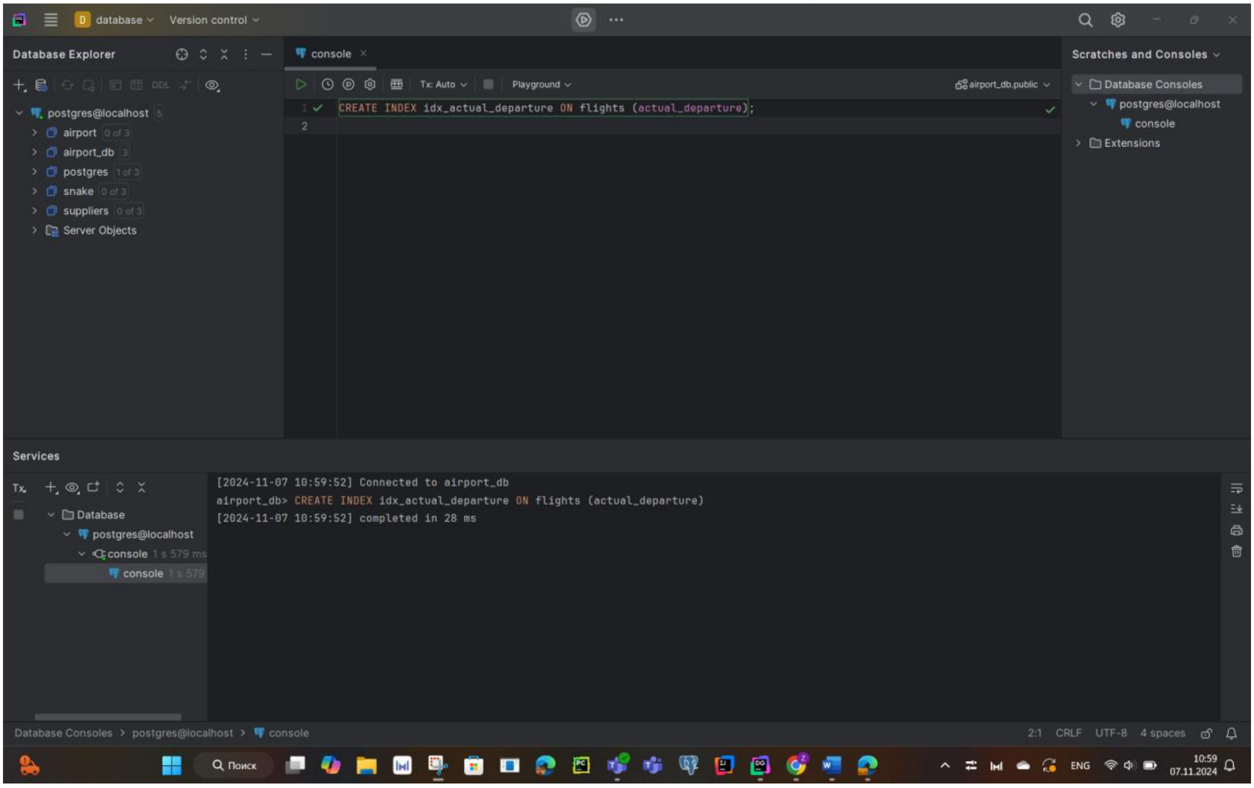
Laboratory work 7

Tasks:

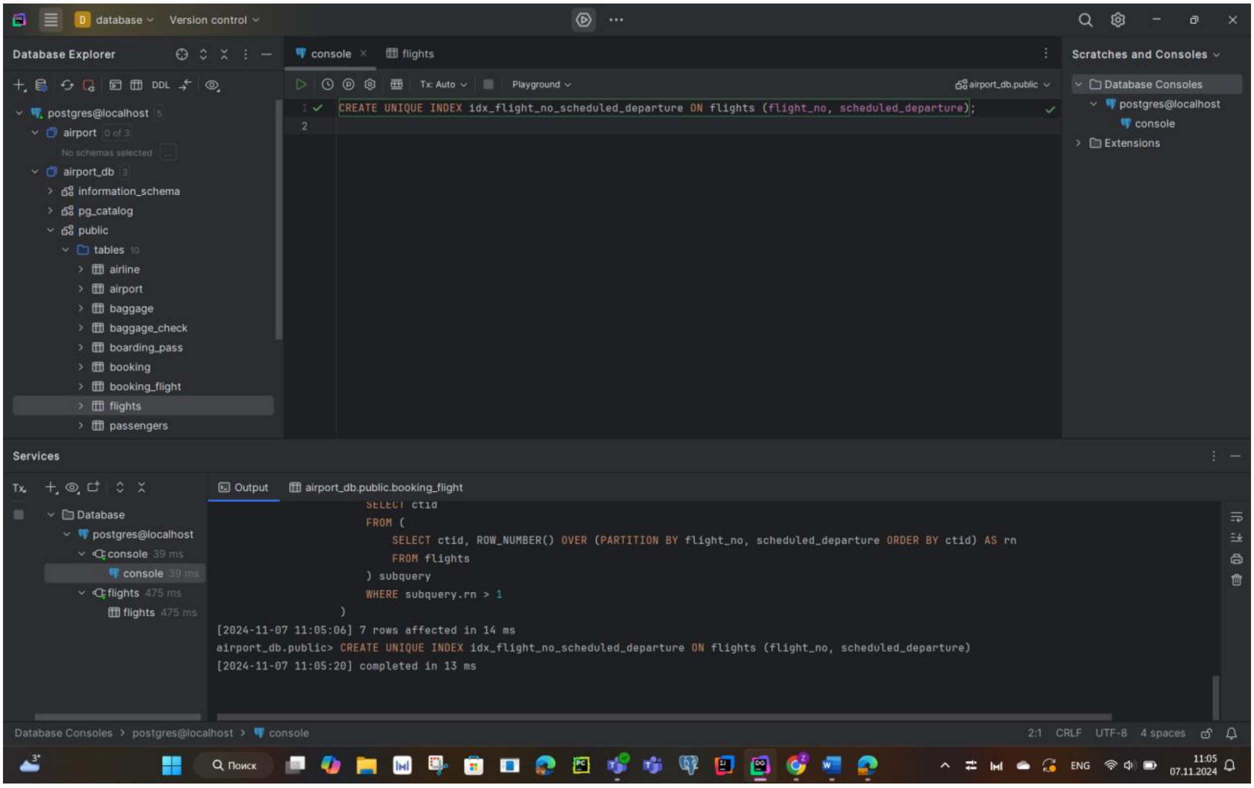
1. Create an index on the actual_departure column in the flights table.



```
CREATE INDEX idx_actual_departure ON flights (actual_departure);
```

```
[2024-11-07 10:59:52] Connected to airport_db
airport_db> CREATE INDEX idx_actual_departure ON flights (actual_departure)
[2024-11-07 10:59:52] completed in 28 ms
```

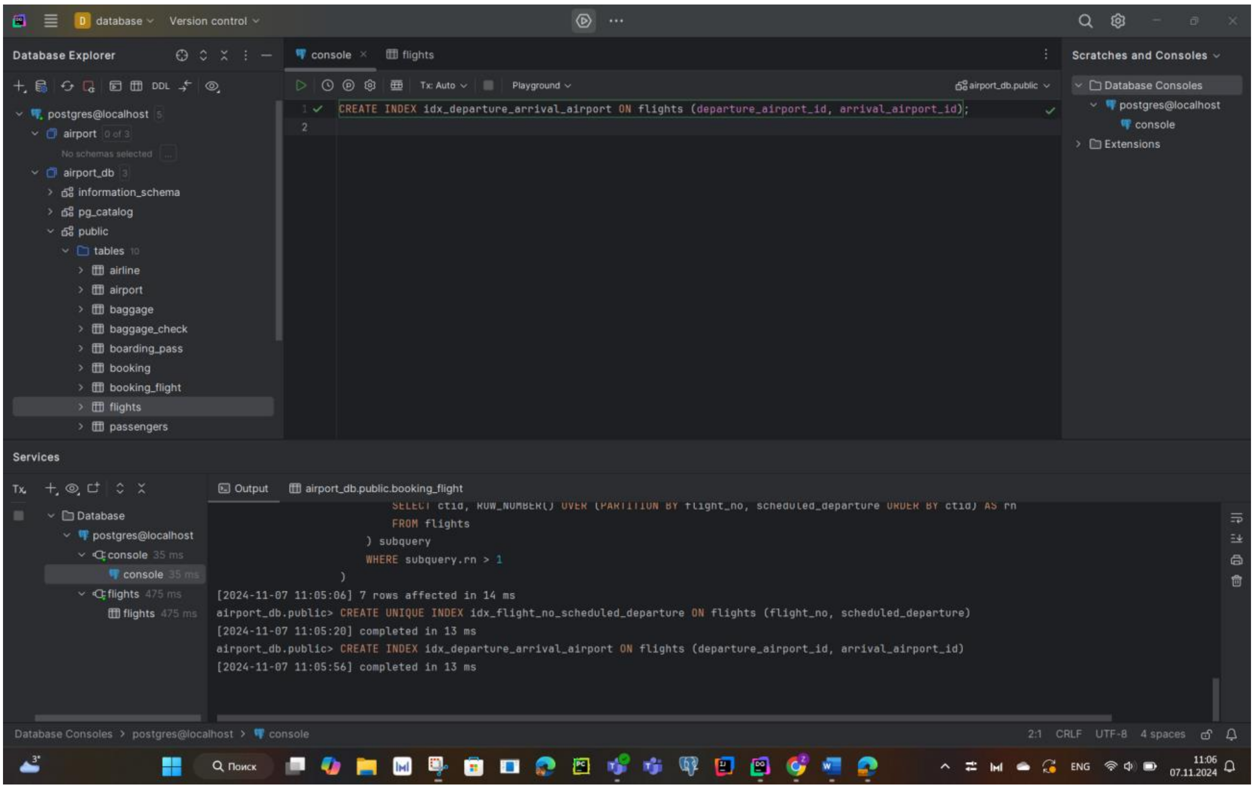
2. Create a unique index to ensure flight_no and scheduled_departure combinations are unique.



```
CREATE UNIQUE INDEX idx_flight_no_scheduled_departure ON flights (flight_no, scheduled_departure);
```

```
                SELECT ctid
                FROM (
                    SELECT ctid, ROW_NUMBER() OVER (PARTITION BY flight_no, scheduled_departure ORDER BY ctid) AS rn
                    FROM flights
                ) subquery
                WHERE subquery.rn > 1
            )
[2024-11-07 11:05:06] 7 rows affected in 14 ms
airport_db.public> CREATE UNIQUE INDEX idx_flight_no_scheduled_departure ON flights (flight_no, scheduled_departure)
[2024-11-07 11:05:20] completed in 13 ms
```

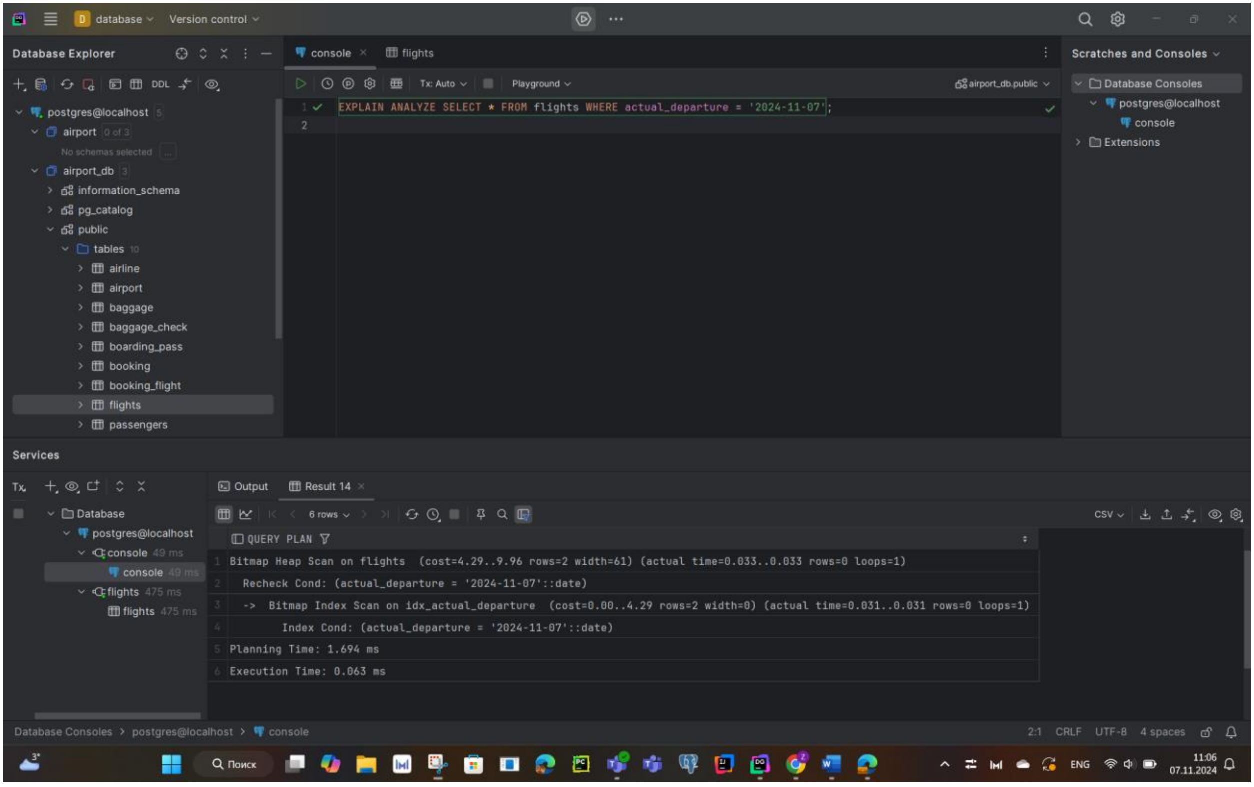
3. Create a composite index on the departure_airport_id and arrival_airport_id columns.



```sql
CREATE INDEX idx_departure_arrival_airport ON flights (departure_airport_id, arrival_airport_id);
```

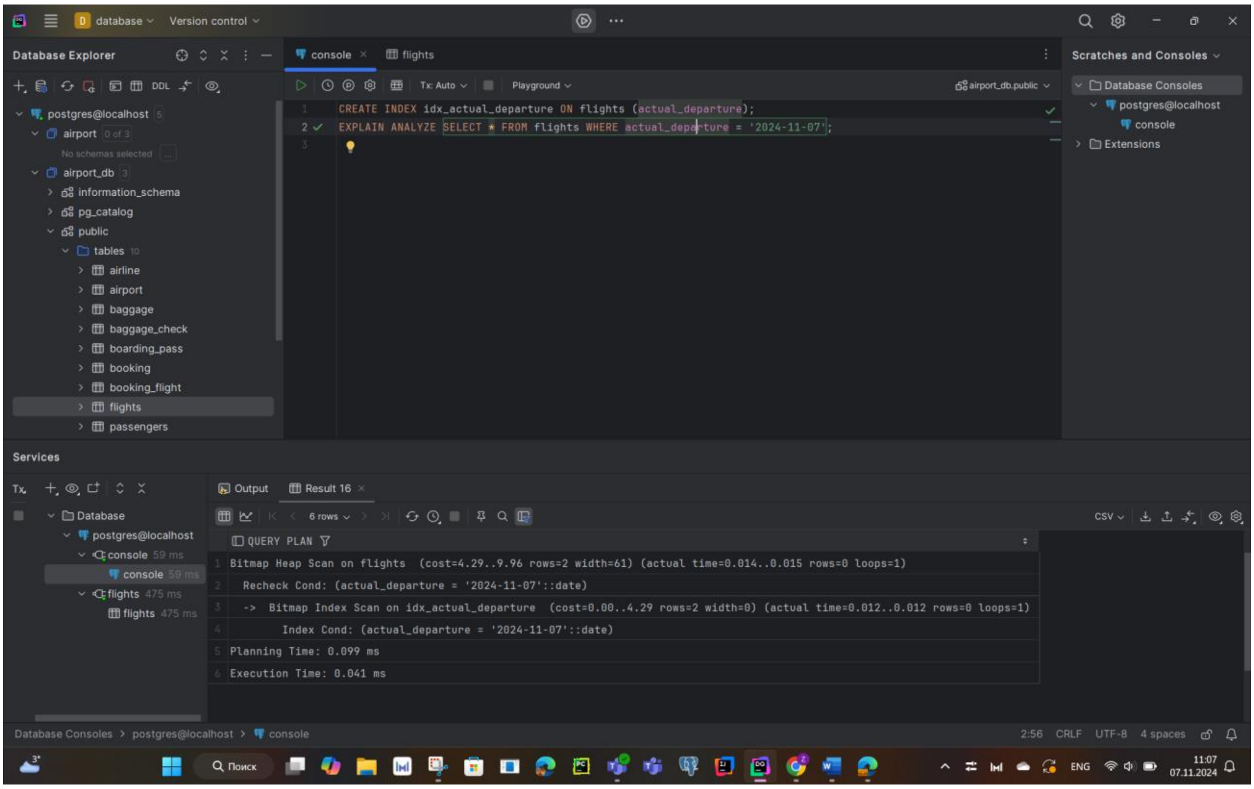
```
                            SELECT ctid, ROW_NUMBER() OVER (PARTITION BY flight_no, scheduled_departure ORDER BY ctid) AS rn
                            FROM flights
                        ) subquery
                        WHERE subquery.rn > 1
                    )
[2024-11-07 11:05:06] 7 rows affected in 14 ms
airport_db.public> CREATE UNIQUE INDEX idx_flight_no_scheduled_departure ON flights (flight_no, scheduled_departure)
[2024-11-07 11:05:20] completed in 13 ms
airport_db.public> CREATE INDEX idx_departure_arrival_airport ON flights (departure_airport_id, arrival_airport_id)
[2024-11-07 11:05:56] completed in 13 ms
```

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



```sql
EXPLAIN ANALYZE SELECT * FROM flights WHERE actual_departure = '2024-11-07';
```

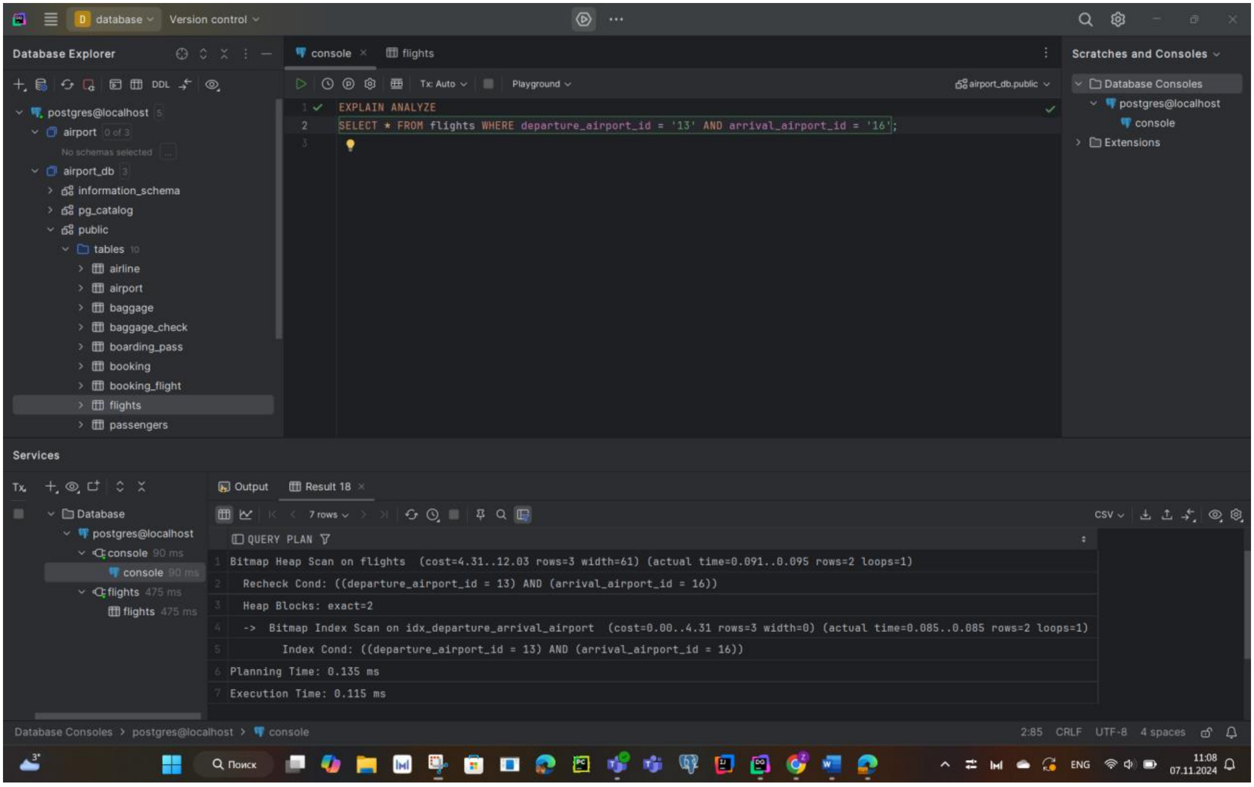
```
QUERY PLAN
Bitmap Heap Scan on flights  (cost=4.29..9.96 rows=2 width=61) (actual time=0.033..0.033 rows=0 loops=1)
  Recheck Cond: (actual_departure = '2024-11-07'::date)
  ->  Bitmap Index Scan on idx_actual_departure  (cost=0.00..4.29 rows=2 width=0) (actual time=0.031..0.031 rows=0 loops=1)
        Index Cond: (actual_departure = '2024-11-07'::date)
Planning Time: 1.694 ms
Execution Time: 0.063 ms
```



```sql
CREATE INDEX idx_actual_departure ON flights (actual_departure);
EXPLAIN ANALYZE SELECT * FROM flights WHERE actual_departure = '2024-11-07';
```

```
QUERY PLAN
Bitmap Heap Scan on flights  (cost=4.29..9.96 rows=2 width=61) (actual time=0.014..0.015 rows=0 loops=1)
  Recheck Cond: (actual_departure = '2024-11-07'::date)
  ->  Bitmap Index Scan on idx_actual_departure  (cost=0.00..4.29 rows=2 width=0) (actual time=0.012..0.012 rows=0 loops=1)
        Index Cond: (actual_departure = '2024-11-07'::date)
Planning Time: 0.099 ms
Execution Time: 0.041 ms
```

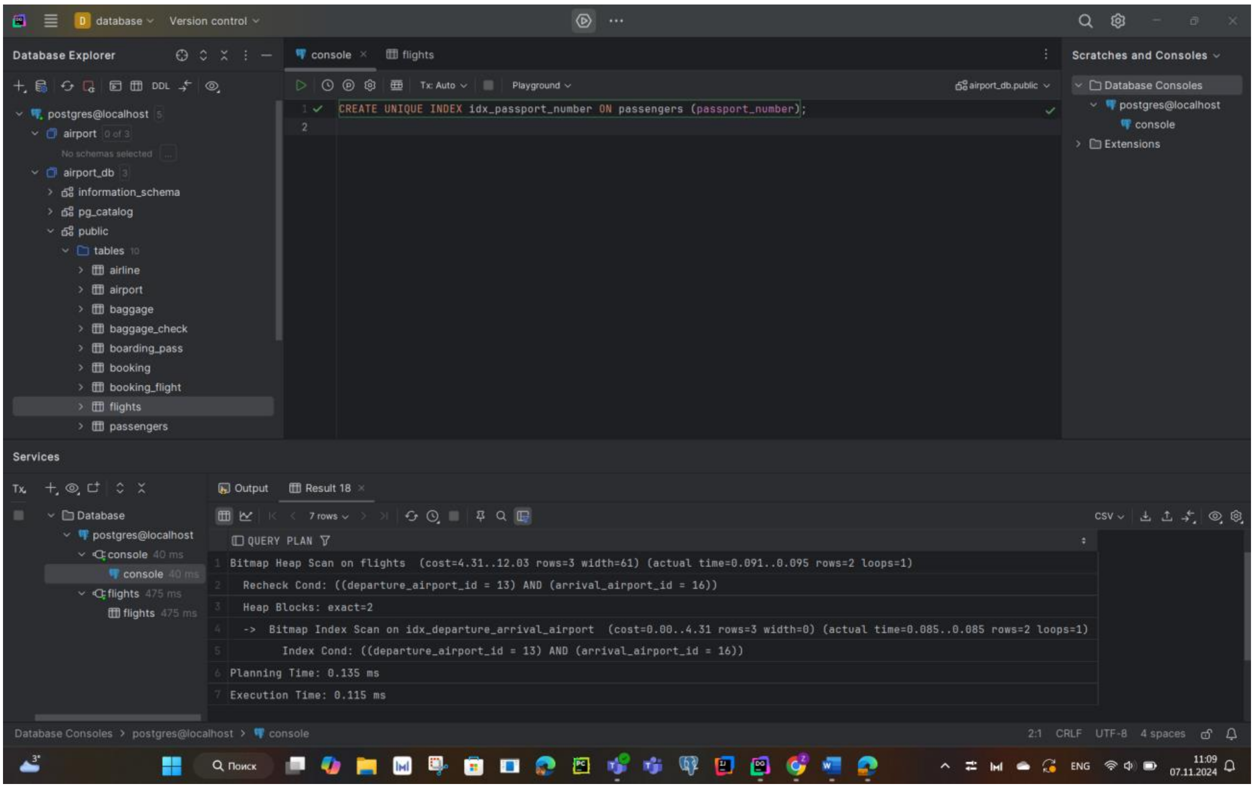
5. Use EXPLAIN ANALYZE to check index usage in a query filtering by departure_airport and arrival_airport.



```sql
EXPLAIN ANALYZE
SELECT * FROM flights WHERE departure_airport_id = '13' AND arrival_airport_id = '16';
```

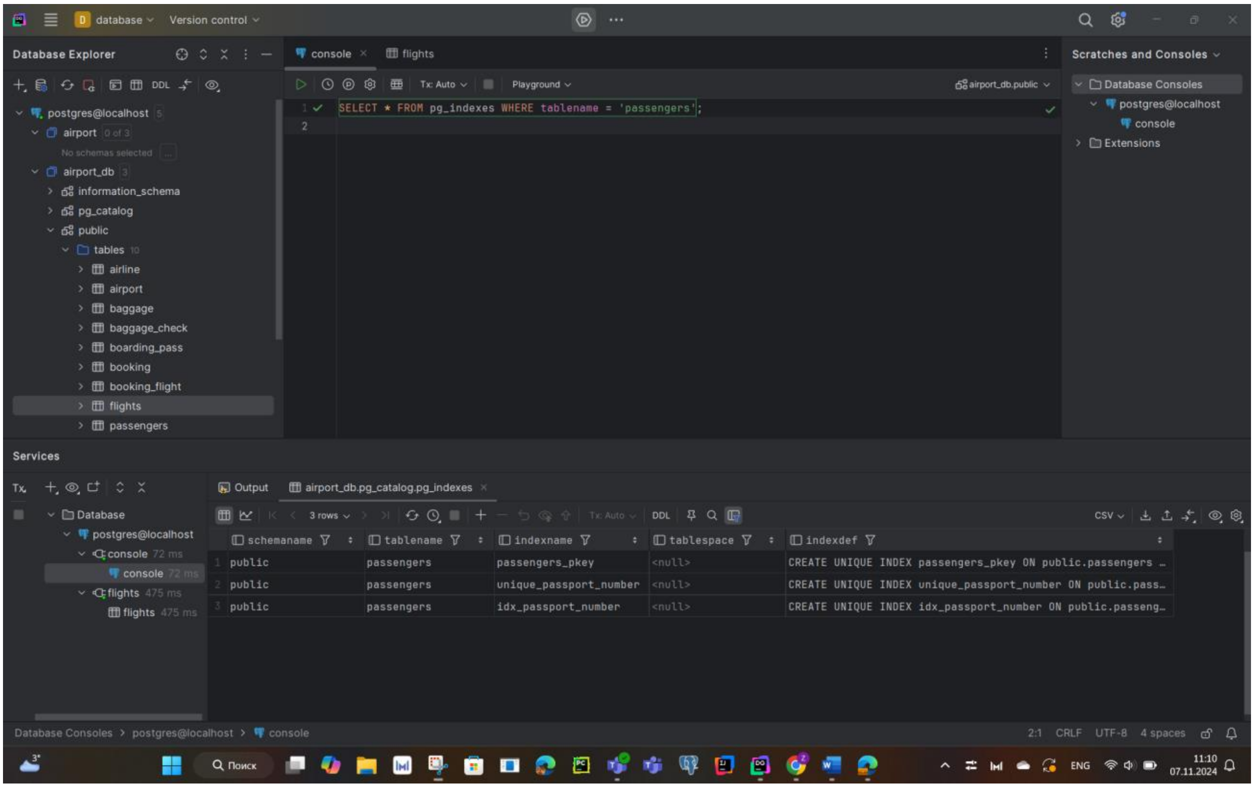
```
QUERY PLAN
Bitmap Heap Scan on flights  (cost=4.31..12.03 rows=3 width=61) (actual time=0.091..0.095 rows=2 loops=1)
  Recheck Cond: ((departure_airport_id = 13) AND (arrival_airport_id = 16))
  Heap Blocks: exact=2
  ->  Bitmap Index Scan on idx_departure_arrival_airport  (cost=0.00..4.31 rows=3 width=0) (actual time=0.085..0.085 rows=2 loops=1)
        Index Cond: ((departure_airport_id = 13) AND (arrival_airport_id = 16))
Planning Time: 0.135 ms
Execution Time: 0.115 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.



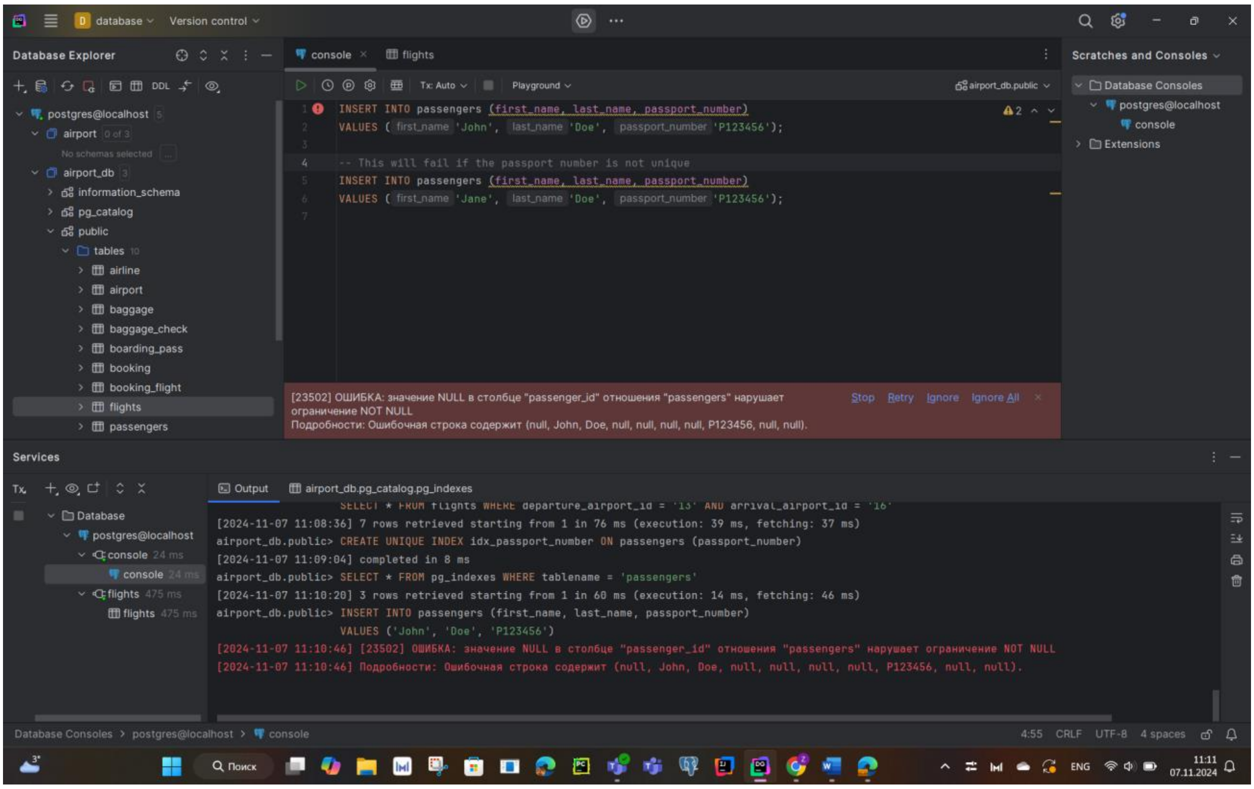
```sql
CREATE UNIQUE INDEX idx_passport_number ON passengers (passport_number);
```

```
QUERY PLAN
Bitmap Heap Scan on flights  (cost=4.31..12.03 rows=3 width=61) (actual time=0.091..0.095 rows=2 loops=1)
  Recheck Cond: ((departure_airport_id = 13) AND (arrival_airport_id = 16))
  Heap Blocks: exact=2
  ->  Bitmap Index Scan on idx_departure_arrival_airport  (cost=0.00..4.31 rows=3 width=0) (actual time=0.085..0.085 rows=2 loops=1)
        Index Cond: ((departure_airport_id = 13) AND (arrival_airport_id = 16))
Planning Time: 0.135 ms
Execution Time: 0.115 ms
```



```sql
SELECT * FROM pg_indexes WHERE tablename = 'passengers';
```

	schemaname	tablename	indexname	tablespace	indexdef
1	public	passengers	passengers_pkey	<null>	CREATE UNIQUE INDEX passengers_pkey ON public.passengers …
2	public	passengers	unique_passport_number	<null>	CREATE UNIQUE INDEX unique_passport_number ON public.pass…
3	public	passengers	idx_passport_number	<null>	CREATE UNIQUE INDEX idx_passport_number ON public.passeng…



```
INSERT INTO passengers (first_name, last_name, passport_number)
VALUES ( first_name 'John', last_name 'Doe', passport_number 'P123456');

-- This will fail if the passport number is not unique
INSERT INTO passengers (first_name, last_name, passport_number)
VALUES ( first_name 'Jane', last_name 'Doe', passport_number 'P123456');
```

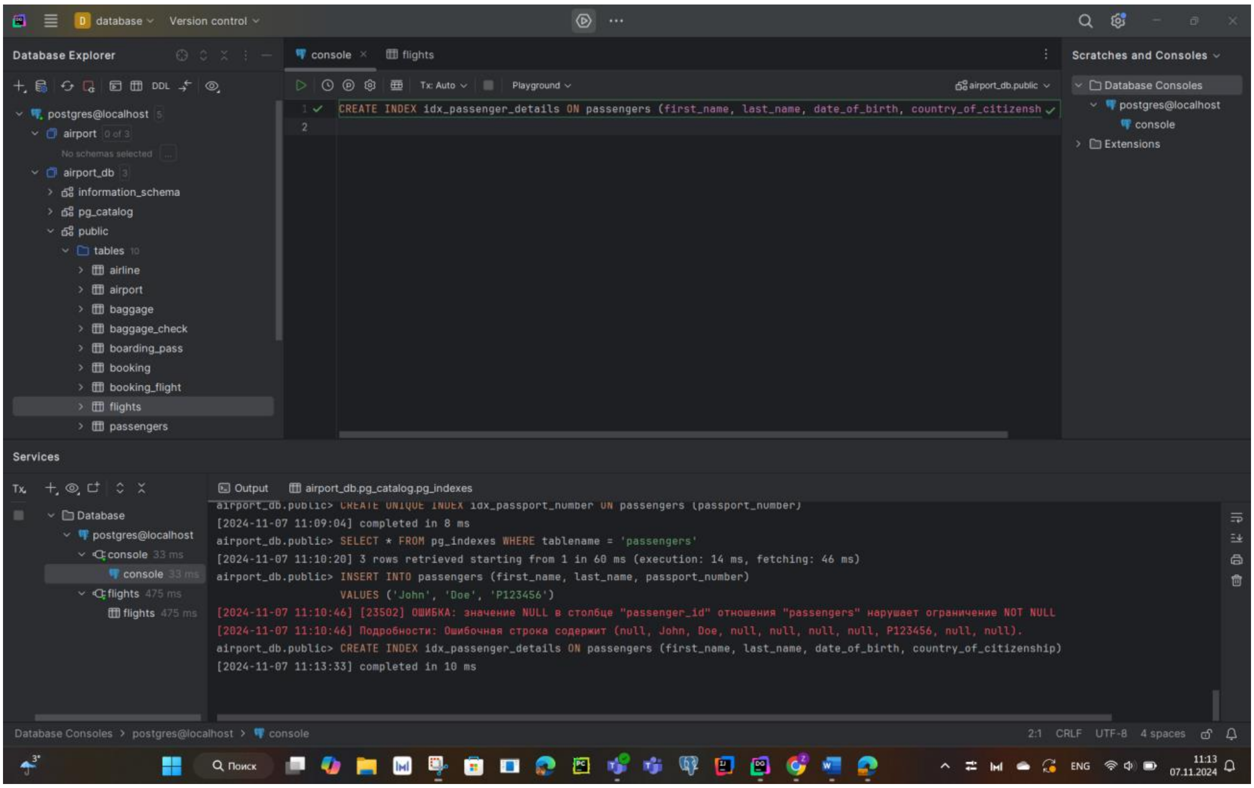
```
[23502] ОШИБКА: значение NULL в столбце "passenger_id" отношения "passengers" нарушает
ограничение NOT NULL
Подробности: Ошибочная строка содержит (null, John, Doe, null, null, null, null, P123456, null, null).
```

```
                 SELECT * FROM flights WHERE departure_airport_id = '13' AND arrival_airport_id = '16'
[2024-11-07 11:08:36] 7 rows retrieved starting from 1 in 76 ms (execution: 39 ms, fetching: 37 ms)
airport_db.public> CREATE UNIQUE INDEX idx_passport_number ON passengers (passport_number)
[2024-11-07 11:09:04] completed in 8 ms
airport_db.public> SELECT * FROM pg_indexes WHERE tablename = 'passengers'
[2024-11-07 11:10:20] 3 rows retrieved starting from 1 in 60 ms (execution: 14 ms, fetching: 46 ms)
airport_db.public> INSERT INTO passengers (first_name, last_name, passport_number)
                   VALUES ('John', 'Doe', 'P123456')
[2024-11-07 11:10:46] [23502] ОШИБКА: значение NULL в столбце "passenger_id" отношения "passengers" нарушает ограничение NOT NULL
[2024-11-07 11:10:46] Подробности: Ошибочная строка содержит (null, John, Doe, null, null, null, null, P123456, null, null).
```

Explain in your own words what is going on in the output?

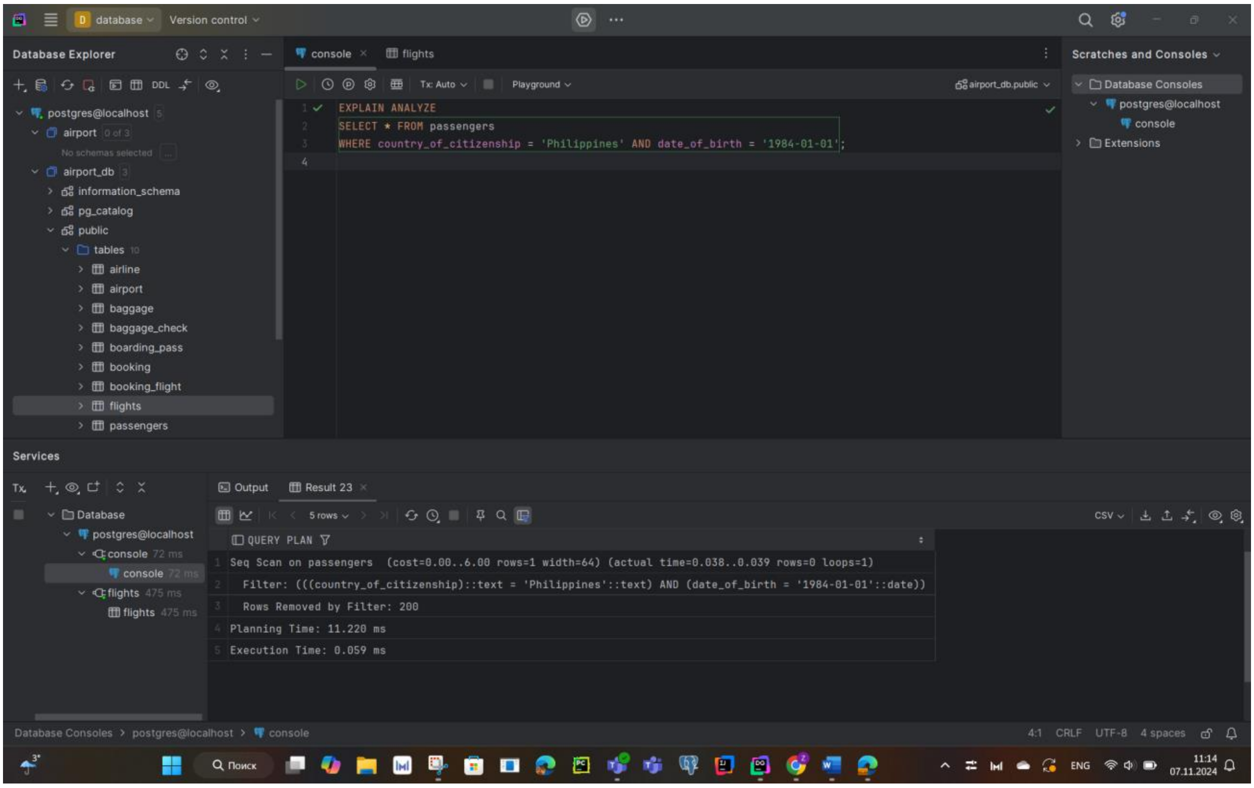
The output will show an error for the second insert due to the unique constraint on passport_number. The purpose of creating a **unique index** is to enforce uniqueness on the passport_number column in the Passengers table. This ensures that no two passengers can have the same passport_number. The first insertion will succeed, but the second one will fail because the unique index will enforce that no duplicates are allowed.

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.



```
CREATE INDEX idx_passenger_details ON passengers (first_name, last_name, date_of_birth, country_of_citizensh
```

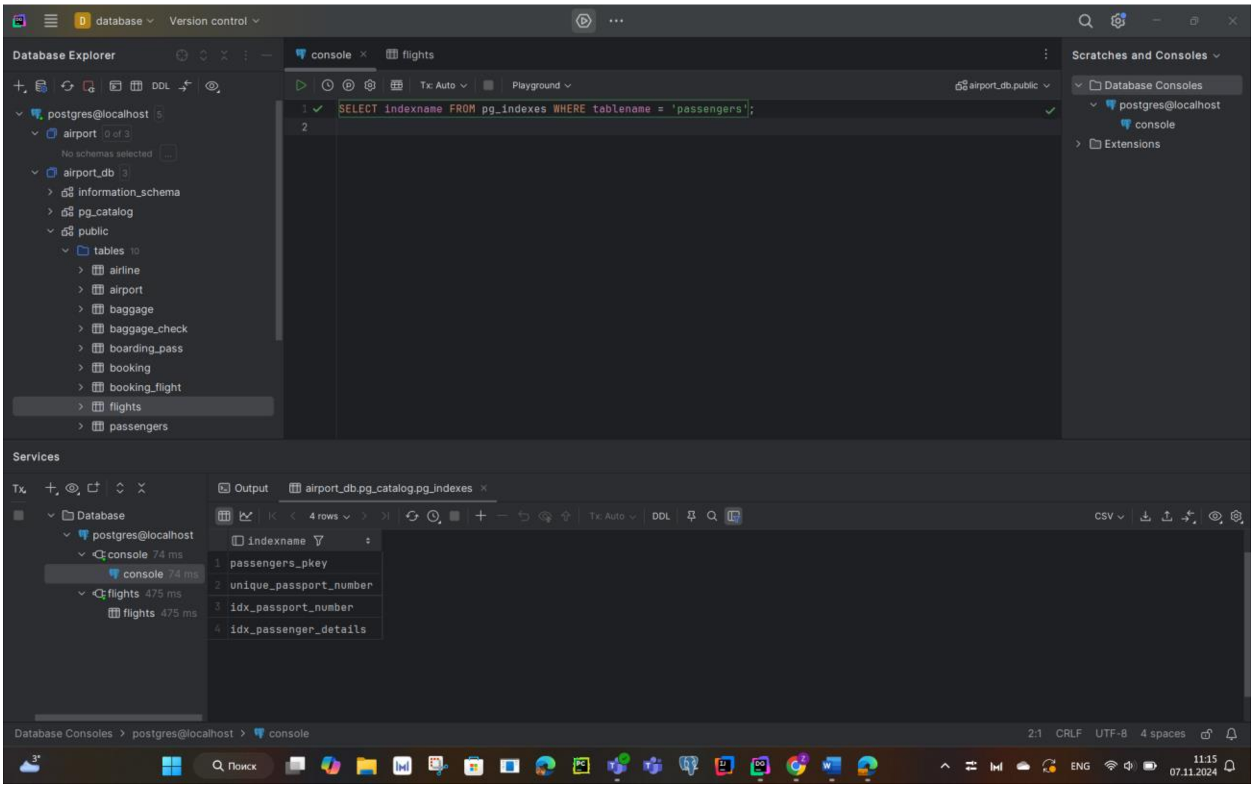
```
airport_db.public> CREATE UNIQUE INDEX idx_passport_number ON passengers (passport_number)
[2024-11-07 11:09:04] completed in 8 ms
airport_db.public> SELECT * FROM pg_indexes WHERE tablename = 'passengers'
[2024-11-07 11:10:20] 3 rows retrieved starting from 1 in 60 ms (execution: 14 ms, fetching: 46 ms)
airport_db.public> INSERT INTO passengers (first_name, last_name, passport_number)
                   VALUES ('John', 'Doe', 'P123456')
[2024-11-07 11:10:46] [23502] ОШИБКА: значение NULL в столбце "passenger_id" отношения "passengers" нарушает ограничение NOT NULL
[2024-11-07 11:10:46] Подробности: Ошибочная строка содержит (null, John, Doe, null, null, null, null, P123456, null, null).
airport_db.public> CREATE INDEX idx_passenger_details ON passengers (first_name, last_name, date_of_birth, country_of_citizenship)
[2024-11-07 11:13:33] completed in 10 ms
```



```
EXPLAIN ANALYZE
SELECT * FROM passengers
WHERE country_of_citizenship = 'Philippines' AND date_of_birth = '1984-01-01';
```

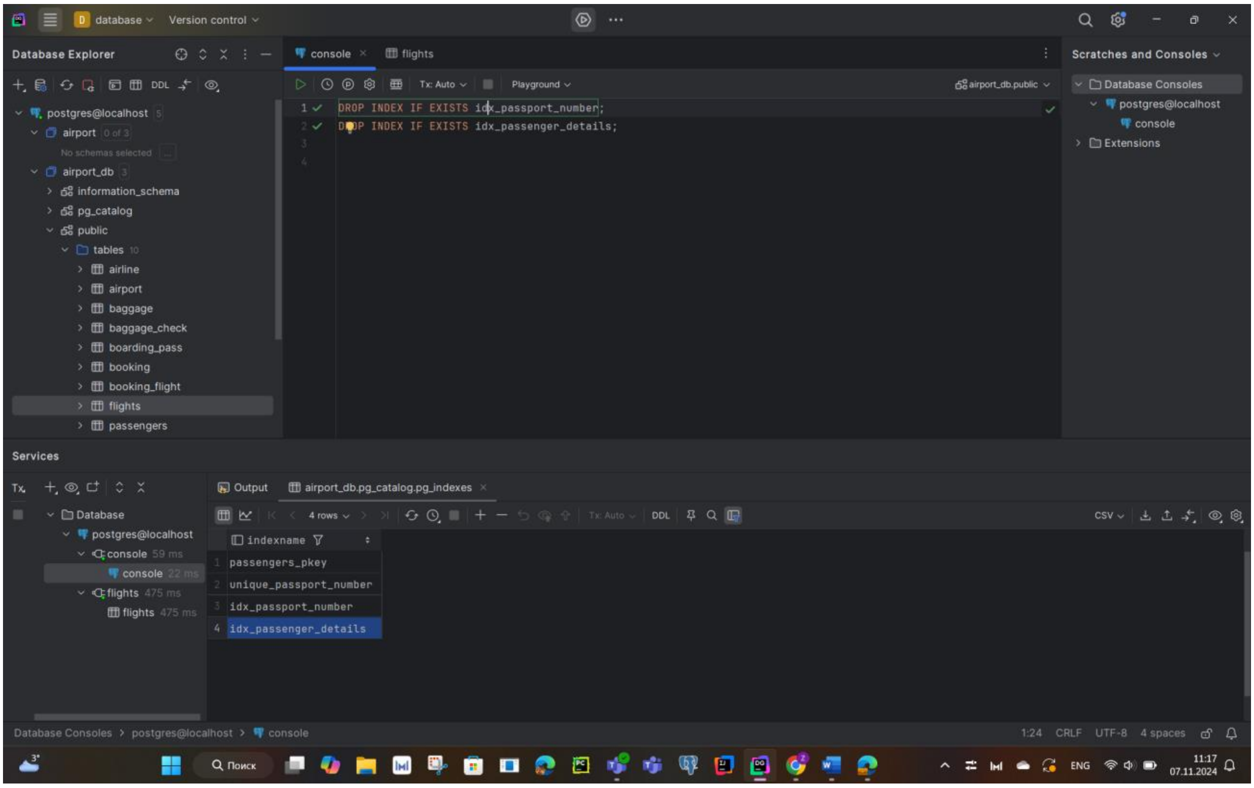
```
QUERY PLAN
Seq Scan on passengers  (cost=0.00..6.00 rows=1 width=64) (actual time=0.038..0.039 rows=0 loops=1)
  Filter: (((country_of_citizenship)::text = 'Philippines'::text) AND (date_of_birth = '1984-01-01'::date))
  Rows Removed by Filter: 200
Planning Time: 11.220 ms
Execution Time: 0.059 ms
```

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



```sql
SELECT indexname FROM pg_indexes WHERE tablename = 'passengers';
```

indexname
passengers_pkey
unique_passport_number
idx_passport_number
idx_passenger_details



```sql
DROP INDEX IF EXISTS idx_passport_number;
DROP INDEX IF EXISTS idx_passenger_details;
```

indexname
passengers_pkey
unique_passport_number
idx_passport_number
idx_passenger_details