

1)

The screenshot shows a database console interface with two tabs: 'console' and 'console_2'. The 'console_2' tab is active. The code area contains the following SQL command:

```
1  1. Create an index on the actual_departure column in the flights table.  
2  
3 ✓ CREATE INDEX flights_act_departure_time_idx  
4   on flights(act_departure_time);  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15
```

The output window below shows the execution results:

```
[2025-11-19 05:41:13] completed in 4 ms  
[2025-11-19 05:41:24] slabwork.public> CREATE INDEX flights_act_departure_time_idx  
on flights(act_departure_time)  
[2025-11-19 05:41:24] completed in 5 ms
```

At the bottom left, it says 'localhost > console_2'. At the bottom right, it shows '16:11 CRL'.

2)

The screenshot shows a database console interface with two tabs: 'console' and 'console_2'. The 'console_2' tab is active. The code area contains the following SQL command:

```
1  2. Create a unique index to ensure flight_no and scheduled_departure cor  
2  
3 ✓ CREATE UNIQUE INDEX idx_flights_flightno_sched  
4   ON Flights (flight_no, sch_departure_time);  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15
```

The output window below shows the execution results:

```
[2025-11-19 05:43:42] completed in 3 ms  
[2025-11-19 05:43:54] slabwork.public> CREATE UNIQUE INDEX idx_flights_flightno_sched  
ON Flights (flight_no, sch_departure_time)  
[2025-11-19 05:43:54] completed in 4 ms
```

At the bottom left, it says 'localhost > console_2'. At the bottom right, it shows '3:1 0'.

3)

```
console  console_2 ×
Tx: Auto  Playground ×

1 3. Create a composite index on the departure_airport_id and arrival_airport_i
2
3 ✓ CREATE INDEX idx_flights_departure_arrival
4   ON Flights (departing_airport_id, arriving_airport_id);
5
6 |
7
8
9
10
11
12
13
14
15
```

[2025-11-19 05:43:54] completed in 4 ms
[2025-11-19 05:48:17] slabwork.public> CREATE INDEX idx_flights_departure_arrival
 ON Flights (departing_airport_id, arriving_airport_id)
[2025-11-19 05:48:17] completed in 5 ms

4)

```
console  console_2 ×
Tx: Auto  Playground ×

1 4. Evaluate the difference in query performance with and without
2
3 ✓ EXPLAIN ANALYZE
4
5   SELECT *
6     FROM Passengers
7   WHERE country_of_citizenship = 'Philippines' AND
8       date_of_birth BETWEEN '1984-01-01' AND '2004-12-31';
9
10
11
12
13
14
15
```

Output Result 85 ×
QUERY PLAN ▾

1 Seq Scan on passengers (cost=0.00..6.50 rows=1 width=66) (actual time=0.033..0.033 rows=0 loops=1)
2 Filter: ((date_of_birth >= '1984-01-01'::date) AND (date_of_birth <= '2004-12-31'::date) AND ((country_of_citizen
3 Rows Removed by Filter: 200
4 Planning Time: 1.059 ms
5 Execution Time: 0.044 ms

```
1  4. Evaluate the difference in query performance with and without index
2
3 ✓ CREATE INDEX idx_passengers_cs_dob ON passengers(country_of_citizenship)
4
5
6
7
8
9
10
11
12
13
14 5. Use EXPLAIN ANALYZE to check index usage in a query filtering by country
15 6. Create a unique index for the passport_number of the Passengers table
```

Output Result 83

```
date_of_birth BETWEEN '1984-01-01' AND '2004-12-31'
[2025-11-19 05:58:29] 5 rows retrieved starting from 1 in 331 ms (execution: 3 ms, fetch: 328 ms)
[2025-11-19 06:00:51] slabwork.public> CREATE INDEX idx_passengers_cs_dob ON passengers(country_of_citizenship)
[2025-11-19 06:00:51] completed in 4 ms
```

```
--4. Evaluate the difference in query performance with and without indexes. Measure execution time
1
2
3 ✓ EXPLAIN ANALYZE
4
5 SELECT *
6 FROM Passengers
7 WHERE country_of_citizenship = 'Philippines' AND
8       date_of_birth BETWEEN '1984-01-01' AND '2004-12-31';
9
10
11
12
13
14
15
```

Output Result 113

QUERY PLAN

```
1 Index Scan using idx_passengers on passengers  (cost=0.14..8.14 rows=1 width=60) (actual time=0.049..0.050 rows=0 loops=1)
2   Index Cond: ((country_of_citizenship)::text = 'Philippines'::text) AND (date_of_birth >= '1984-01-01'::date) AND (date_of_birth <= '2004-12-31'::date)
3 Planning Time: 0.148 ms
4 Execution Time: 0.063 ms
```

5)

```
console  console_2 ×
▶ ⌂ Tx: Auto ▶ Playground ▶
1 --5.      Use EXPLAIN ANALYZE to check index usage in a query
2
3 ✓ EXPLAIN ANALYZE
4
5   SELECT
6     flight_id,
7       departing_airport_id,
8       arriving_airport_id,
9       sch_departure_time,
10      sch_arrival_time
11 FROM Flights
12 WHERE departing_airport_id = 3
13   AND arriving_airport_id = 7;
14
15
```

Output Result 387 Result 389 ×

QUERY PLAN

```
1 Bitmap Heap Scan on flights  (cost=182.95..19271.12 rows=13124 width=28) (actual time=0.049..0.049 rows=6)
  2  Recheck Cond: ((departing_airport_id = 3) AND (arriving_airport_id = 7))
  3    -> Bitmap Index Scan on idx_fl_daiaai  (cost=0.00..179.67 rows=13124 width=0) (actual time=0.039..0.039 rows=6)
  4      Index Cond: ((departing_airport_id = 3) AND (arriving_airport_id = 7))
  5 Planning Time: 1.283 ms
  6 Execution Time: 0.077 ms
```

```
console console_2
Tx: Auto | Playground ▾

1 --5. Use EXPLAIN ANALYZE to check index usage in a query filtering by dep
2
3 ✓ EXPLAIN ANALYZE
4
5 SELECT
6   flight_id,
7   departing_airport_id,
8   arriving_airport_id,
9   sch_departure_time,
10  sch_arrival_time
11 FROM Flights
12 WHERE departing_airport_id = 3
13   AND arriving_airport_id = 7;
14
15
```

The screenshot shows a PostgreSQL terminal window with two tabs: 'console' and 'console_2'. The code in the terminal is an 'EXPLAIN ANALYZE' command for a query that selects flight details from the 'Flights' table where the departing airport ID is 3 and the arriving airport ID is 7. The output pane displays the query plan, which shows a 'Seq Scan on flights' with a cost of 10000000000.00..10000040144.00, scanning 13124 rows with a width of 28. The actual time for the execution was 72.392..72.392 ms. A filter was applied to the results, removing 1321200 rows. The planning time was 0.060 ms, and the execution time was 72.415 ms.

	Value
Cost	10000000000.00..10000040144.00
Rows	13124
Width	28
Actual Time	72.392..72.392 ms
Planning Time	0.060 ms
Execution Time	72.415 ms

```
  console  console_2 ×
▶ ⏴ ⏵ ⏴ Tx: Auto ⏴ ⏴ Playground ⏴
1   --5.    Use EXPLAIN ANALYZE to check index usage in a q
2
3   CREATE INDEX idx_fl_daiaai ON flights(departing_airport
4
5
6
7
8
9
10
11
12
13
14
15
16
```

Output Result 387 Result 389

```
WHERE departing_airport
      AND arriving_airport_
[2025-11-19 06:27:57] 5 rows retrieved starting from 1 in 402 ms
[2025-11-19 06:30:13] slabwork.public> CREATE INDEX idx_fl_dai
[2025-11-19 06:30:13] completed in 596 ms
```

6)

The screenshot shows a database console interface with two tabs: 'console' and 'console_2'. The 'console_2' tab is active and displays a SQL command to create a unique index on the 'Passengers' table:

```
--6.      Create a unique index for the passport_number of the P
1
2
3 ✓ CREATE UNIQUE INDEX idx_passengers_passport_number
4   ON Passengers (passport_number);
5
6
7
8
9
10
11
12
13
14
15
```

Below the code editor, the results pane shows the execution log:

```
Output Result 387 Result 389
[2025-11-19 06:38:21] 6 rows retrieved starting from 1 in 350 ms (executed in 350 ms)
[2025-11-19 06:38:38] slabwork.public> CREATE UNIQUE INDEX idx_passengers_
ON Passengers (passport_number)
[2025-11-19 06:38:38] completed in 5 ms
```

```
1 --6.      Create a unique index for the passport_number of the Passen
2
3 ✓ INSERT INTO Passengers (
4     passenger_id, first_name, last_name, date_of_birth,
5     gender, country_of_citizenship, country_of_residence,
6     passport_number, created_at, updated_at
7 ) VALUES (
8     passenger_id 1001, first_name 'Test', last_name 'One', date_of_birth '1995-05-05',
9     gender 'Male', country_of_citizenship 'USA', country_of_residence 'USA',
10    passport_number 'P999999999', created_at NOW(), updated_at NOW()
11 );
12
13
14
15
```

Output Result 387 Result 389

'Male', 'USA', 'USA',
'P999999999', NOW(), NOW()
)
[2025-11-19 06:39:15] 1 row affected in 4 ms

```
1 --6.      Create a unique index for the passport_number of the Passengers table. Check
2
3 ⓘ INSERT INTO Passengers (
4     passenger_id, first_name, last_name, date_of_birth,
5     gender, country_of_citizenship, country_of_residence,
6     passport_number, created_at, updated_at
7 ) VALUES (
8     passenger_id 1002, first_name 'Test', last_name 'Two', date_of_birth '1995-05-05',
9     gender 'Female', country_of_citizenship 'USA', country_of_residence 'USA',
10    passport_number 'P999999999', created_at NOW(), updated_at NOW()
11 );
12 |
13
14
```

[23505] ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности "idx_passengers_passport_number"
Подробности: Ключ "(passport_number)=(P999999999)" уже существует.

Output Result 387 Result 389
'P999999999', NOW(), NOW()
)
[23505] ОШИБКА: повторяющееся значение ключа нарушает ограничение уникальности "idx_passengers_passport_number"
Подробности: Ключ "(passport_number)=(P999999999)" уже существует.

7)

```
console  console_2 x
Tx: Auto  Playground v

1 7. Create an index for the Passengers table. Use for that first
2
3 ✓ CREATE INDEX idx_passengers_country_dob_name
4 ON Passengers (
5     country_of_citizenship,
6     date_of_birth,
7     last_name,
8     first_name
9 );
10
11
12
13
14
15
```

Output Result 387 Result 389

```
last_name,
first_name
)
completed in 5 ms
```

```
console  console_2 x
Tx: Auto  Playground v
slabwork.p

1 --7. Create an index for the Passengers table. Use for that first name, last name, country_of_citizenship
2
3 ✓ EXPLAIN ANALYZE
4
5 SELECT
6     passenger_id,
7     first_name,
8     last_name,
9     date_of_birth,
10    country_of_citizenship
11 FROM Passengers
12 WHERE country_of_citizenship = 'Philippines'
13     AND date_of_birth BETWEEN DATE '1984-01-01' AND DATE '1985-01-01';
14
15
```

Output Result 394 Result 389

CSV | ↻ | ⌂ | ■ | ⌂ | ⌂ | ↺

QUERY PLAN

```
1 Index Scan using idx_passengers_dob on passengers (cost=0.14..8.11 rows=1 width=27) (actual time=0.004..0.004 rows=0 loops=1)
2 | Index Cond: ((date_of_birth >= '1984-01-01'::date) AND (date_of_birth <= '1985-01-01'::date) AND ((country_of_citizenship)::text = 'Philippines'::text))
3 Planning Time: 1.170 ms
4 Execution Time: 0.017 ms
```

4 rows | :

8)

```
1 8. Write a SQL query to list indexes for table Passenger
2 ✓ SELECT
3     indexname,
4     indexdef
5 FROM pg_indexes
6 WHERE tablename = 'passenger';
7
8
9
10
11
12
13
14
```

indexname	indexdef
passenger_pkey	CREATE UNIQUE INDEX passenger_pkey ON public.passenger USI...
idx_passenger_cs_dob	CREATE INDEX idx_passenger_cs_dob ON public.passenger USIN...
idx_passenger	CREATE INDEX idx_passenger ON public.passenger USING btree...
idx_passenger_dob	CREATE INDEX idx_passenger_dob ON public.passenger USING b...
idx_passenger_passport_number	CREATE UNIQUE INDEX idx_passenger_passport_number ON public...
idx_passenger_country_dob_name	CREATE INDEX idx_passenger_country_dob_name ON public.passe...

```
console      console_2 ×
Tx: Auto | Playground ▾

1 --8.      Write a SQL query to list indexes for table Passengers.
2
3 ✓  DROP INDEX
4      idx_passengers_cs_dob,
5      idx_passengers,
6      idx_passengers_dob,
7      idx_passengers_passport_number,
8      idx_passengers_country_dob_name;
9
10
11
12
13
14
```

Output Result 396

```
idx_passengers_cs_dob,
idx_passengers,
idx_passengers_dob,
idx_passengers_passport_number,
idx_passengers_country_dob_name
completed in 6 ms
```