

TASK 1

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Materialized Views
- Operators
- Procedures
- 1.3 Sequences
- Tables (10)
 - airline
 - airport
 - baggage
 - baggage_check
 - boarding_pass
 - booking
 - booking_flight
 - flights
- Columns (12)
 - flight_id
 - sch_departure_time
 - sch_arrival_time
 - departing_airport_id
 - arriving_airport_id
 - departing_gate
 - arriving_gate
 - airline_id
 - act_departure_time
 - act_arrival_time
 - created_at
 - updated_at
- Constraints
- Indexes
- RLS Policies
- Rules

Airport/postgres@PostgreSQL 17*

Airport/postgres@PostgreSQL 17

Query Query History

```
1 CREATE INDEX act_departure_time_index ON flights(act_departure_time)
```

Scratch Pad

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 50 msec.

Total rows: Query complete 00:00:00.050

CRLF Ln 1, Col 68

TASK 2

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Tables (10)
 - airline
 - airport
 - baggage
 - baggage_check
 - boarding_pass
 - booking
 - booking_flight
 - flights**
 - Columns (13)
 - flight_id
 - sch_departure_time
 - sch_arrival_time
 - departing_airport_id
 - arriving_airport_id
 - departing_gate
 - arriving_gate
 - airline_id
 - act_departure_time
 - act_arrival_time
 - created_at
 - updated_at
 - flight_no
 - Constraints
 - Indexes
 - RLS Policies
 - Rules
 - Triggers
 - passengers
 - security_check

Airport/postgres@PostgreSQL 17*

Airport/postgres@PostgreSQL 17

No limit

Query Query History

```
1 CREATE UNIQUE INDEX unique_schdep_flightno_index ON flights(flight_no, sch_departure_time)
```

Scratch Pad

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 53 msec.

Total rows: Query complete 00:00:00.053

CRLF Ln 1, Col 91

16:31 07.11.2025

TASK 3

pgAdmin 4

File Object Tools Edit View Window Help

Object Explorer

- Tables (10)
 - airline
 - airport
 - baggage
 - baggage_check
 - boarding_pass
 - booking
 - booking_flight
 - flights
- Columns (13)
 - flight_id
 - sch_departure_time
 - sch_arrival_time
 - departing_airport_id
 - arriving_airport_id
 - departing_gate
 - arriving_gate
 - airline_id
 - act_departure_time
 - act_arrival_time
 - created_at
 - updated_at
 - flight_no
- Constraints
- Indexes
- RLS Policies
- Rules
- Triggers
- passengers
- security_check

Airport/postgres@PostgreSQL 17* x

Airport/postgres@PostgreSQL 17

No limit

Query Query History

```
1 CREATE INDEX flights_departure_arrival_index ON flights(departing_airport_id, arriving_airport_id)
```

Scratch Pad x

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 55 msec.

Total rows: Query complete 00:00:00.055 CRLF Ln 1, Col 99



Поиск



50

PYC



16:35
07.11.2025

TASK 4

The screenshot displays the pgAdmin 4 web interface. On the left, the 'Object Explorer' pane shows a tree structure of the database: Servers (1) > PostgreSQL 17 > Databases (2) > Airport. The main pane shows the 'Query Editor' for the 'Airport/postgres@PostgreSQL 17' connection. The query being executed is:

```
SELECT * FROM flights
WHERE act_departure_time BETWEEN '2020-01-01' AND '2025-02-01';
```

The 'Query Output' pane at the bottom shows the results of the query. The table has 10 columns: flight_id (PK) integer, sch_departure_time timestamp without time zone, sch_arrival_time timestamp without time zone, departing_airport_id integer, arriving_airport_id integer, departing_gate text, arriving_gate character varying (50), airline_id integer, and act_deptime timestamp. The results show 6 rows of flight data.

	flight_id [PK] integer	sch_departure_time timestamp without time zone	sch_arrival_time timestamp without time zone	departing_airport_id integer	arriving_airport_id integer	departing_gate text	arriving_gate character varying (50)	airline_id integer	act_deptime timestamp
1	1	2022-06-07 09:11:49	2022-06-07 18:11:49	180	185	G10	H1	2	2022-0
2	4	2021-06-29 14:30:59	2021-06-30 02:30:59	118	15	G16	H19	1	2021-0
3	5	2023-08-24 22:10:45	2023-08-25 09:10:45	118	15	G21	H20	2	2023-0
4	6	2022-10-16 14:46:12	2022-10-16 20:46:12	20	117	G22	H13	150	2022-1
5	8	2022-10-05 10:35:32	2022-10-05 19:35:32	171	35	G19	H18	4	2022-1
6	76	2021-06-17 05:06:08	2021-06-17 07:06:08	164	198	G14	H3	193	2021-0

The status bar at the bottom indicates 'Total rows: 173', 'Query complete 00:00:00.095', and 'Ln 2, Col 39'.

WITH INDEX

The screenshot displays the pgAdmin 4 web interface. On the left, the 'Object Explorer' shows the database structure for 'Airport/postgres@PostgreSQL 17', including 'Databases (2)', 'Airport', 'Catalogs', 'Event Triggers', 'Extensions', 'Foreign Data Wrappers', 'Languages', 'Publications', 'Schemas (1)', and 'public'. The 'public' schema contains 'Aggregates', 'Collations', 'Domains', 'FTS Configurations', 'FTS Dictionaries', 'FTS Parsers', 'FTS Templates', 'Foreign Tables', 'Functions', 'Materialized Views', 'Operators', 'Procedures', 'Sequences', and 'Tables (10)'. The 'Tables (10)' folder is expanded, showing 'airline', 'airport', 'baggage', and 'baggage_check'.

The main pane shows the 'Query' editor with the following SQL query:

```
SELECT * FROM flights
WHERE act_departure_time BETWEEN '2020-01-01' AND '2020-02-01';
```

The 'Query Output' pane displays the results of the query, showing 173 rows. The table has the following columns: flight_id (PK) integer, sch_departure_time timestamp without time zone, sch_arrival_time timestamp without time zone, departing_airport_id integer, arriving_airport_id integer, departing_gate text, arriving_gate character varying (50), airline_id integer, and act_deptime timestamp. The results show flights from 2020-01-01 to 2020-02-01.

The status bar at the bottom indicates 'Total rows: 173', 'Query complete 00:00:00.106', and 'Ln 2, Col 39'.

WITHOUT INDEX

TASK 5

The screenshot displays the pgAdmin 4 web interface. On the left, the 'Object Explorer' pane shows a tree structure of the database: Servers (1) > PostgreSQL 17 > Databases (2) > Airport. The 'public' schema is expanded, showing various database objects like Aggregates, Collations, Domains, etc., and a list of 10 tables including 'airline', 'airport', 'baggage', and 'baggage_check'. The main pane is titled 'Airport/postgres@PostgreSQL 17*'. It contains a SQL query editor with the following text:

```
1  EXPLAIN ANALYZE
2  SELECT * FROM flights
3  WHERE departing_airport_id = 1 AND arriving_airport_id = 3;
4
```

Below the query editor, the 'Data Output' tab is active, showing the 'QUERY PLAN' for the executed query. The plan details are as follows:

Step	Operation
1	Index Scan using flights_departure_arrival_index on flights (cost=0.14..8.16 rows=1 width=76) (actual time=0.038..0.038 rows=0 loops=0)
2	Index Cond: (((departing_airport_id = 1) AND (arriving_airport_id = 3)))
3	Planning Time: 1.529 ms
4	Execution Time: 0.055 ms

At the bottom of the interface, a status bar indicates 'Total rows: 4', 'Query complete 00:00:00.053', and the current cursor position 'Ln 4, Col 1'.

TASK 6

The screenshot displays the pgAdmin 4 interface with two query windows open. The left window shows a successful execution of a SQL query to create a unique index on the 'passengers' table. The right window shows an error message resulting from an attempt to insert a duplicate key value into the same table.

Left Query Window:

```
CREATE UNIQUE INDEX passenger_passport_number_index ON passengers(passport_number)
```

Right Query Window:

```
INSERT INTO passengers (passenger_id, first_name, last_name, date_of_birth, gender, country_of_citizenship, country_of_residence, passport_number, created_at, updated_at) VALUES (201, 'Asel', 'Nurpeis', '1999-09-09', 'Female', 'Kazakhstan', 'Kazakhstan', 'P00001273', NOW(), NOW())
```

Error Message:

```
ERROR: duplicate key value violates unique constraint "passengers_passport_number_key"
Key (passport_number)=(P00001273) already exists.

SQL state: 23505
Detail: Key (passport_number)=(P00001273) already exists.
```

CHECKING

TASK 7

The image displays two side-by-side screenshots of the pgAdmin 4 interface, demonstrating a PostgreSQL query and its execution plan.

Left Screenshot: The 'Query' tab shows the following SQL code:

```
1 CREATE INDEX passengers_name_birth_country_index
2 ON passengers (first_name, last_name, date_of_birth, country_of_citizenship);
```

The 'Messages' tab shows the result: 'Query returned successfully in 55 msec.'

Right Screenshot: The 'Query' tab shows the following SQL code:

```
1 EXPLAIN ANALYZE
2 SELECT *
3 FROM passengers
4 WHERE country_of_citizenship = 'Philippines'
5 AND EXTRACT(YEAR FROM date_of_birth) = 1984;
```

The 'Data Output' tab shows the 'QUERY PLAN' for the second query:

Step	Plan
1	Seq Scan on passengers (cost=0.00..6.50 rows=1 width=64) (actual time=0.021..0.036 rows=1 loops=1)
2	Filter: (((country_of_citizenship)::text = 'Philippines'::text) AND (EXTRACT(year FROM date_of_birth) = '1984'::num...
3	Rows Removed by Filter: 199
4	Planning Time: 0.100 ms
5	Execution Time: 0.044 ms

The 'Messages' tab shows the result: 'Successfully run. Total query runtime: 62 msec. 5 rows returned.'

PostgreSQL used a sequential scan, not the index. The `EXTRACT(YEAR FROM date_of_birth)` function stops the index from working, and the table is small, so a full scan is faster.

TASK 8

The image displays two side-by-side screenshots of the pgAdmin 4 interface, showing SQL queries and their results.

Left Screenshot:

- Query:**

```
1 SELECT
2     indexname,
3     indexdef
4 FROM
5     pg_indexes
6 WHERE
7     tablename = 'passengers';
8
```
- Data Output:**

	indexname	indexdef
1	passengers_pkey	CREATE UNIQUE INDEX passengers_pkey ON public.passengers USING btree (passenger_id)
2	passengers_passport_number_key	CREATE UNIQUE INDEX passengers_passport_number_key ON public.passengers USING btree (passport_number)
3	unique_passport	CREATE UNIQUE INDEX unique_passport ON public.passengers USING btree (passport_number)
4	passenger_passport_number_index	CREATE UNIQUE INDEX passenger_passport_number_index ON public.passengers USING btree (passport_number)
5	passengers_name_birth_country_in...	CREATE INDEX passengers_name_birth_country_index ON public.passengers USING btree (first_name, last_name, date_of_birth, country_of_citize...
- Status:** Total rows: 5 | Query complete 00:00:00.077

Right Screenshot:

- Query:**

```
1 DROP INDEX IF EXISTS passenger_passport_number_index;
2 DROP INDEX IF EXISTS passengers_name_birth_country_index;
3
```
- Data Output:**

DROP INDEX

Query returned successfully in 56 msec.
- Status:** Total rows: | Query complete 00:00:00.056