

**Министерство науки и высшего образования Российской Федерации** федеральное  
государственное автономное образовательное учреждение высшего образования  
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**О Т Ч Е Т** по

лабораторной работе

«Запросы на выборку и модификацию данных, представления и индексы в PostgreSQL»

по дисциплине «Проектирование и реализация баз данных»

Автор: Глушков Кирилл Георгиевич

Факультет: ИКТ

Группа: К32422

Преподаватель: Говорова Марина Михайловна

Дата сдачи: 15.04.23



Санкт-Петербург 2023

## СОДЕРЖАНИЕ

1 Описание работы .....	2
2 Схема базы данных.....	3
3 Выполнение запросов .....	4

### 1 Описание работы

Цель работы: овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

Практическое задание:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.
3. Изучить графическое представление запросов и просмотреть историю запросов.

Dashboard

Properties

SQL

Statistics

Dependencies

Dependents

Processes

postgres/post...

Untitled\*

postgres/post...

Search

Table name	Tuples inserted	Tuples updated	Tuples deleted	Tuples HOT updated	Live tuples	Dead tuples	Last vacuum
airport	11	0	0	0	11	0	
aviacompany	16	5	3	5	5	16	
crew	22	0	0	0	15	7	
flight	16	0	0	0	15	1	
passenger	5	0	0	0	5	0	
plane	30	5	10	5	30	5	
plane_type	5	0	0	0	5	0	
rank	14	0	0	0	9	5	
schedule	10	16	0	16	5	21	
seat	10	0	0	0	5	5	
staff	10	0	0	0	5	5	
ticket	27	0	0	0	5	22	
ticket_office	8	0	0	0	5	3	
transit	23	12	2	12	14	21	

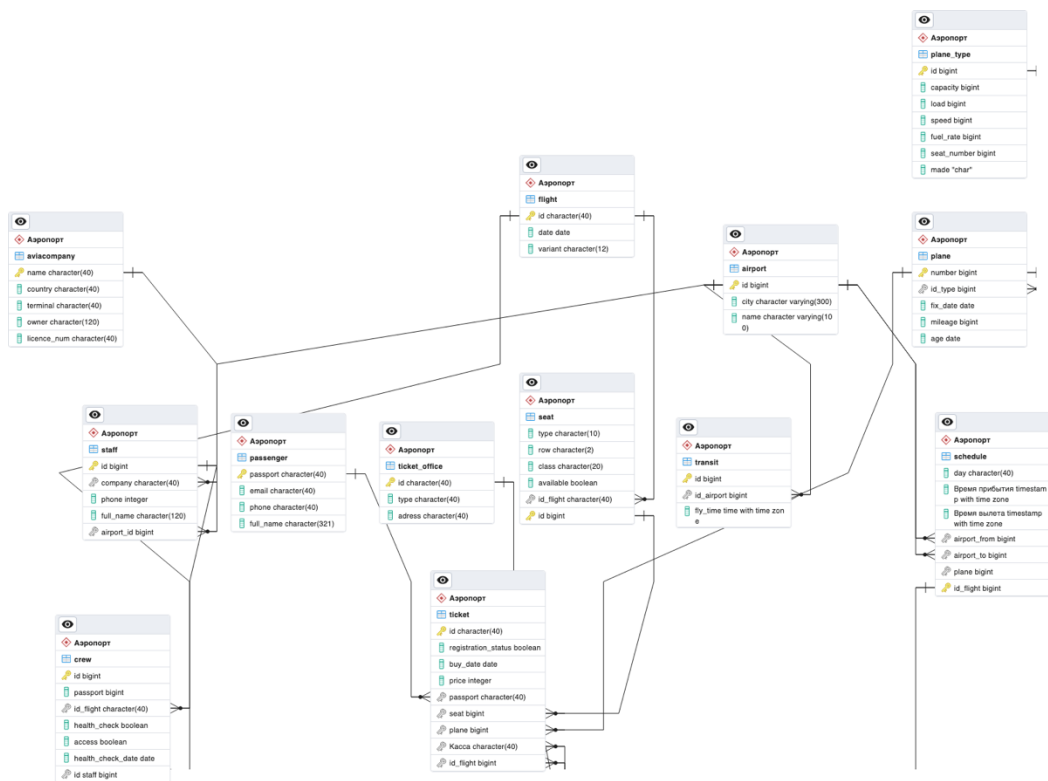
4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

Индивидуальное задание по варианту «Аэропорт».

## 2 Схема базы данных

Схема была создана с использованием генератора ERD-схемы в pgadmin.

Рисунок 1 – Схема базы данных «Аэропорт»



### 3 Выполнение запросов

- Определить расчетное время полета по всем маршрутам.

set search\_path to Аэропорт;

```
SELECT s.id_flight, s."Время прибытия", s."Время вылета", EXTRACT(EPOCH FROM
(s."Время прибытия" - s."Время вылета"))/60 AS "Разница в минутах"
FROM schedule s
JOIN plane p ON s.plane = p.number;
```

pgAdmin 4 Object Tools Edit Window Help

pgAdmin 4

Object Explorer

- Aggregates
- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (13)
  - airport
  - aviacompany
  - crew
  - flight
  - passenger
  - plane
  - plane\_type
  - schedule
  - seat
  - staff
  - ticket
  - ticket\_office
  - transit
  - Trigger Functions
  - Types
  - Views
  - Subscriptions

postgres/glu@postgres\*

Query

```

1 set search_path to Аэропорт;
2
3 SELECT s.id_flight, s."Время прибытия", s."Время вылета", EXTRACT(EPOCH FROM (s
4 FROM schedule s
5 JOIN plane p ON s.plane = p.number;

```

Data Output

	id_flight [PK] bigint	Время прибытия timestamp with time zone	Время вылета timestamp with time zone	Разница в минутах numeric
1	6	2023-05-27 08:30:00+03	2023-05-27 09:30:00+03	-60.000000000000000000
2	1	2022-01-11 08:30:00+03	2022-01-11 09:30:00+03	-60.000000000000000000
3	7	2023-05-27 10:15:00+03	2023-05-27 11:15:00+03	-60.000000000000000000
4	2	2022-01-12 10:15:00+03	2022-01-12 11:15:00+03	-60.000000000000000000
5	9	2023-05-26 14:45:00+03	2023-05-26 15:45:00+03	-60.000000000000000000
6	8	2023-05-27 12:20:00+03	2023-05-27 13:20:00+03	-60.000000000000000000
7	4	2022-01-14 14:45:00+03	2022-01-14 15:45:00+03	-60.000000000000000000
8	3	2022-01-13 12:20:00+03	2022-01-13 13:20:00+03	-60.000000000000000000
9	10	2023-05-26 16:30:00+03	2023-05-26 17:30:00+03	-60.000000000000000000
10	5	2022-01-15 16:30:00+03	2022-01-15 17:30:00+03	-60.000000000000000000

Total rows: 10 of 10 Query complete 00:00:00.056 Ln 5, Col 35

- Определить расход топлива по всем маршрутам.

pgAdmin 4 Object Tools Edit Window Help

pgAdmin 4

Object Explorer

- Aggregates
- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (13)
  - airport
  - aviacompany
  - crew
  - flight
  - passenger
  - plane
  - plane\_type
  - schedule
  - seat
  - staff
  - ticket
  - ticket\_office
  - transit
  - Trigger Functions
  - Types
  - Views
  - Subscriptions
- Columns (5)
  - number
  - id\_type
  - fix\_date
  - mileage
  - age
- Constraints
- Indexes
- RLS Policies
- Rules
- Triggers

postgres/glu@postgres\*

Query

```

1 set search_path to Аэропорт;
2
3 SELECT SUM(p.mileage) AS total_count
4 FROM plane p
5 JOIN plane_type m ON p.id_type = m.id;

```

Data Output

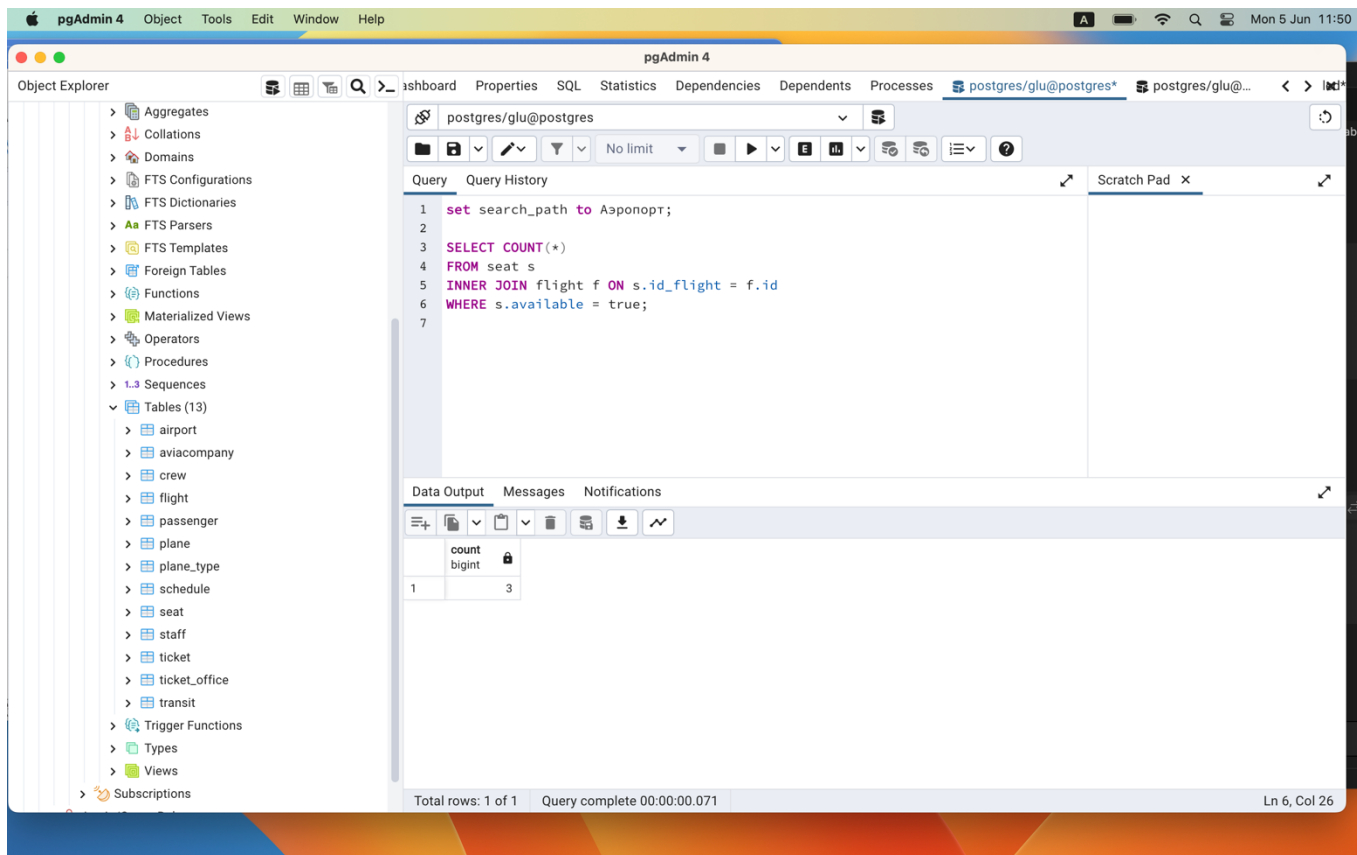
	total_count numeric
1	2313000

Successfully run. Total query runtime: 67 msec. 1 rows affected.

Total rows: 1 of 1 Query complete 00:00:00.067 Ln 5, Col 39

- Вывести данные о том, сколько свободных мест оставалось в самолетах, совершавших полет по заданному из рейсов за вчерашний день.

```
SELECT COUNT(*)
FROM "Аэропорт".seat s
INNER JOIN "Аэропорт".flight f ON s.flight_no = f.flight_no
WHERE s.available = true;
```



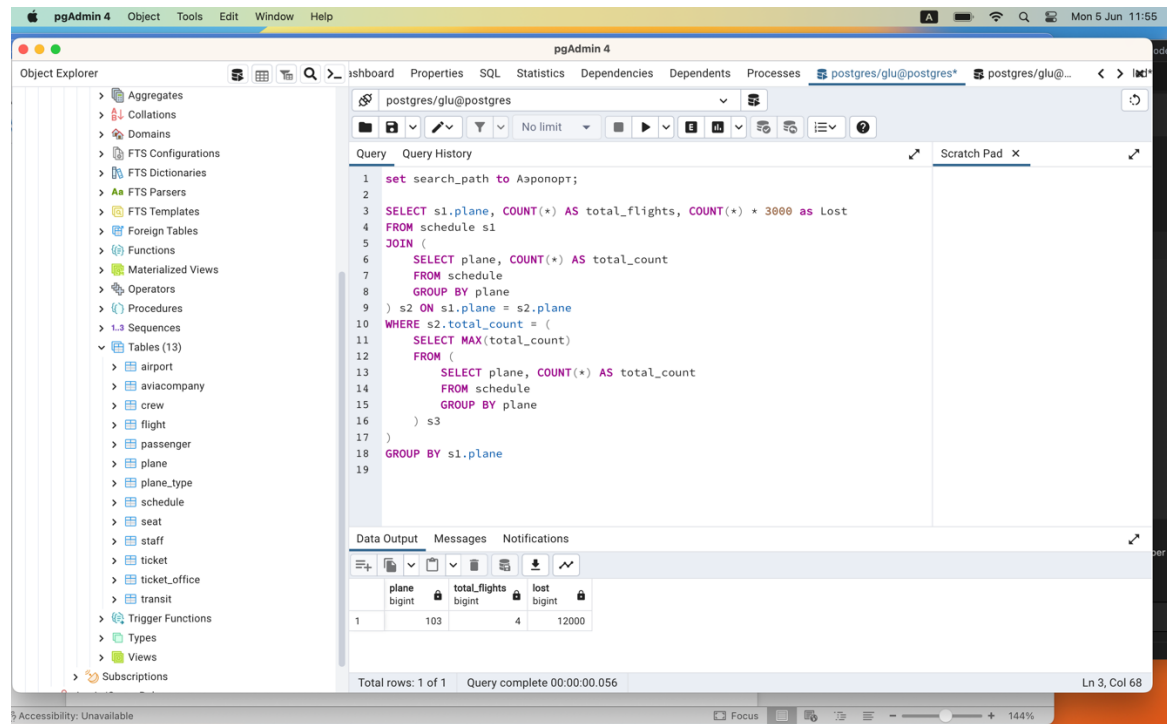
- Рассчитать убытки компании за счет непроданных билетов за вчерашний день.

```
SELECT s1.plane, COUNT(*) AS total_flights, COUNT(*) * 3000 as
Lost
FROM schedule s1
JOIN (
    SELECT plane, COUNT(*) AS total_count
    FROM schedule
    GROUP BY plane
) s2 ON s1.plane = s2.plane
WHERE s2.total_count = (
```

```

SELECT MAX(total_count)
FROM (
    SELECT plane, COUNT(*) AS total_count
    FROM schedule
    GROUP BY plane
) s3
)
GROUP BY s1.plane

```



- Вывести список самолетов, “возраст” которых превышает средний “возраст” самолетов этого типа.

```

1 SELECT *
2 FROM Аэропорт.plane p
3 JOIN (
4     SELECT id_type, AVG(EXTRACT(YEAR FROM AGE(CURRENT_DATE, age))) AS avg_age
5     FROM Аэропорт.plane
6     GROUP BY id_type
7 ) AS avg ON p.id_type = avg.id_type
8 WHERE EXTRACT(YEAR FROM AGE(CURRENT_DATE, p.age)) > avg.avg_age;

```

Data Output Messages Notifications

	number [PK] bigint	id_type bigint	fix_date date	mileage bigint	age date	id_type bigint	avg_age numeric
1	101	1	2021-08-01	100000	1999-01-01	1	23.166666666666667
2	102	2	2021-07-15	75000	2000-02-10	2	19.833333333333333
3	103	3	2021-08-05	40000	2010-07-03	3	10.833333333333333
4	104	4	2021-06-30	25000	2015-09-21	4	4.666666666666667
5	106	1	2021-09-15	80000	1998-11-20	1	23.166666666666667
6	108	1	2021-07-31	95000	1997-09-12	1	23.166666666666667
7	110	1	2021-09-25	110000	1996-12-18	1	23.166666666666667
8	111	2	2021-09-10	67000	2001-03-14	2	19.833333333333333
9	112	2	2021-08-16	60000	1999-04-28	2	19.833333333333333
10	118	3	2021-09-18	35000	2009-06-17	3	10.833333333333333
11	120	3	2021-09-13	25000	2011-01-24	3	10.833333333333333
12	122	4	2021-09-21	22000	2017-07-19	4	4.666666666666667
13	124	4	2021-08-17	28000	2016-08-12	4	4.666666666666667
14	127	5	2021-08-12	155000	2004-02-19	5	17.000000000000000
15	129	5	2021-07-25	148000	2002-05-16	5	17.000000000000000

SELECT \*

FROM Аэропорт.plane p

JOIN (

SELECT id\_type, AVG(EXTRACT(YEAR FROM AGE(CURRENT\_DATE, age))) AS avg\_age

FROM Аэропорт.plane

GROUP BY id\_type

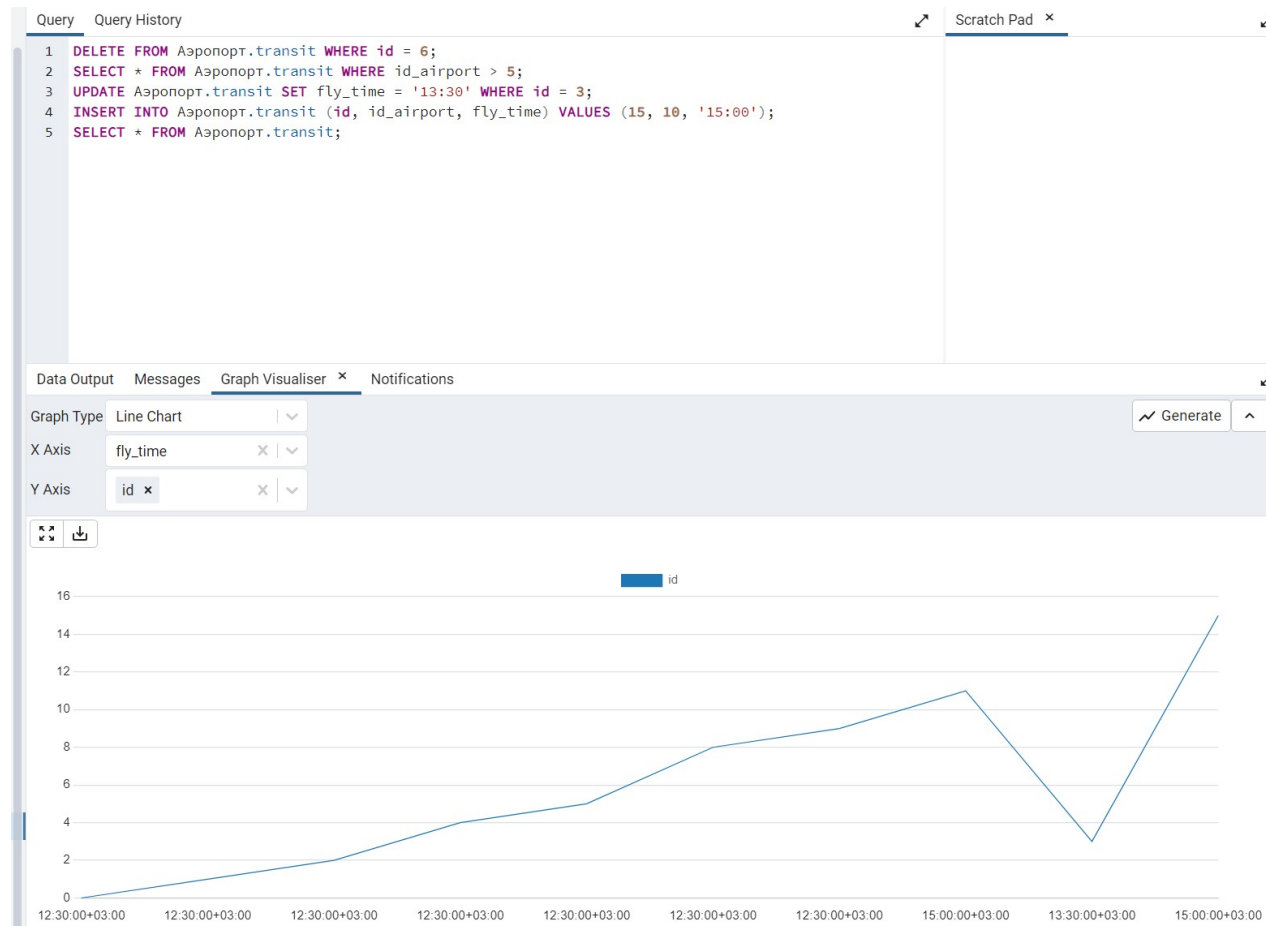
) AS avg ON p.id\_type = avg.id\_type

WHERE EXTRACT(YEAR FROM AGE(CURRENT\_DATE, p.age)) > avg.avg\_age;



- Определить тип самолетов, летающих во все аэропорты назначения.

## Изучить графическое представление запросов и просмотреть историю запросов



**Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов.**

**INSERT**

```

32
33 ((SELECT 'Cathay Pacific Airways' EXCEPT SELECT name FROM "Аэропорт".aviacompany LIMIT 1),
34 'Hong Kong',
35 'Terminal F',
36 'Cathay Pacific Group',
37 'ER145'),
38
39 ((SELECT 'Qantas' EXCEPT SELECT name FROM "Аэропорт".aviacompany LIMIT 1),
40 'Australia',
41 'Terminal G',
42 'Qantas Airways Limited',
43 'CE372'),
44
45 ((SELECT 'Turkish Airlines' EXCEPT SELECT name FROM "Аэропорт".aviacompany LIMIT 1),
46 'Turkey',
47 'Terminal H',
48 'Turkish Airlines Inc.',
49 'LA124'),
50
51 ((SELECT 'Lufthansa' EXCEPT SELECT name FROM "Аэропорт".aviacompany LIMIT 1),
52 'Germany',
53 'Terminal I',
54 'Deutsche Lufthansa AG',
55 'PM439'),
56
57 ((SELECT 'British Airways' EXCEPT SELECT name FROM "Аэропорт".aviacompany LIMIT 1),
58 'UK',
59 'Terminal J',
60 'International Airlines Group',
61 'GA508');
62
63 select * from Аэропорт.aviacompany

```

Data Output Messages Notifications

	name [PK] character (40)	country character (40)	terminal character (40)	owner character (120)	licence_num character (40)
	Emirates	Australia	Terminal 8	Qantas Airways Limited	QF-377
	Qatar Airways	Singapore	Terminal 10	Singapore Airlines Limited	SQ-992
	Air India	India	Terminal 1	Government of India	AI-1234
	Delta Air Lines	United States	Terminal 2	Richard Anderson	DL-5678
	Lufthansa	Germany	Terminal 4	Carsten Spohr	LH-3456

## UPDATE

До:

Query Query History

```

1 select * from Аэропорт.ticket

```

Data Output Messages Notifications

<div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div></div>										
	id [PK] character (40)	registration_status boolean	buy_date date	price integer	passport character (40)	id_flight character (40)	seat bigint	plane bigint	Kacca character (40)	
1	ABC002	true	2022-01-02	200	AK-234567	2	2	102	TO002	
2	ABC003	true	2022-01-03	300	RS-345678	3	3	103	TO003	
3	ABC004	false	2022-01-04	400	ST-456789	4	4	104	TO004	
4	ABC005	true	2022-01-05	500	BG-567890	5	5	105	TO005	

## После:

```

1 UPDATE Аэропорт.ticket
2 SET registration_status = false
3 WHERE passport IN (
4     SELECT passport FROM Аэропорт.passenger WHERE full_name = 'Bob Smith'
5 );
6
7 select * from Аэропорт.ticket

```

Data Output Messages Notifications

	id [PK] character (40)	registration_status boolean	buy_date date	price integer	passport character (40)	id_flight character (40)	seat bigint	plane bigint	Kacca character (40)
1	ABC002	true	2022-01-02	200	AK-234567	2	2	102	TO002
2	ABC004	false	2022-01-04	400	ST-456789	4	4	104	TO004
3	ABC005	true	2022-01-05	500	BG-567890	5	5	105	TO005
4	ABC003	false	2022-01-03	300	RS-345678	3	3	103	TO003

## DELETE

### До

```
1 select * from Аэропорт.ticket
2
```

Data Output Messages Notifications

	id [PK] character (40)	registration_status boolean	buy_date date	price integer	passport character (40)	id_flight character (40)	seat bigint	plane bigint	Kacca character (40)
1	ABC001	false	2022-01-01	100	CJ-123456	1	1	101	TO001
2	ABC002	true	2022-01-02	200	AK-234567	2	2	102	TO002
3	ABC003	true	2022-01-03	300	RS-345678	3	3	103	TO003
4	ABC004	false	2022-01-04	400	ST-456789	4	4	104	TO004
5	ABC005	true	2022-01-05	500	BG-567890	5	5	105	TO005

### После

Query Query History

```
1 DELETE FROM Аэропорт.ticket
2 WHERE passport IN (
3   SELECT passport FROM Аэропорт.passenger WHERE full_name = 'John Doe'
4 );
5
6 select * from Аэропорт.ticket
```

Data Output Messages Notifications

	id [PK] character (40)	registration_status boolean	buy_date date	price integer	passport character (40)	id_flight character (40)	seat bigint	plane bigint	Kacca character (40)
1	ABC002	true	2022-01-02	200	AK-234567	2	2	102	TO002
2	ABC003	true	2022-01-03	300	RS-345678	3	3	103	TO003
3	ABC004	false	2022-01-04	400	ST-456789	4	4	104	TO004
4	ABC005	true	2022-01-05	500	BG-567890	5	5	105	TO005

## 3. Представления

- для пассажиров авиакомпании о рейсах в Москву на ближайшую неделю;

set search\_path to Аэропорт;

```
CREATE VIEW moscow_flight AS
SELECT p.full_name, t.id_flight
FROM passenger p
JOIN ticket t ON t.passport = p.passport
WHERE date(t.buy_date) <= date(now() + interval '7 days')
AND EXISTS (
  SELECT * FROM schedule s JOIN airport a ON s.airport_to = a.id
  WHERE s.id_flight = t.id_flight AND a.city = 'Москва'
);
```

SELECT \* FROM moscow\_flight;

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, including tables like 'airport', 'aviation', 'crew', 'flight', 'passenger', 'plane', 'schedule', 'seat', 'staff', and 'ticket'. The main pane shows a SQL query editor with the following code:

```

1 set search_path to Аэропорт;
2
3 -- CREATE VIEW moscow_flight AS
4 -- SELECT p.full_name, t.id_flight
5 -- FROM passenger p
6 -- JOIN ticket t ON t.passport = p.passport
7 -- WHERE date(t.buy_date) <= date(now() + interval '7 days')
8 -- AND EXISTS (
9 --   SELECT * FROM schedule s JOIN airport a ON s.airport_to = a.id
10 --   WHERE s.id_flight = t.id_flight AND a.city = 'Москва'
11 -- );
12
13 SELECT * FROM moscow_flight;
14

```

The 'Data Output' tab at the bottom shows the results of the query:

full_name	id_flight
Jane Doe	6
Alice Johnson	7
Chris Wilson	8
Bob Smith	9

Total rows: 4 of 4. Query complete 00:00:00.056.

• КОЛИЧЕСТВО САМОЛЕТОВ КАЖДОГО ТИПА, ЛЕТАВШИМИ ЗА ПОСЛЕДНИЙ МЕСЯЦ.

```

1 SET search_path TO Аэропорт;
2
3 CREATE OR REPLACE VIEW plane_type_flying_count AS
4 SELECT pt.id, pt.made, COUNT(*) AS flying_count
5 FROM plane p
6 JOIN plane_type pt ON p.id_type = pt.id
7 JOIN ticket t ON t.plane = p."number"
8 WHERE t.buy_date >= CURRENT_DATE - INTERVAL '1 month'
9 GROUP BY pt.id, pt.made;
10
11 SELECT * FROM plane_type_flying_count;

```

The 'Data Output' tab shows the results of the query:

id	made	flying_count
1	B	1
2	A	1
3	E	1
4	S	1

```
SET search_path TO Аэропорт;
```

```
CREATE OR REPLACE VIEW plane_type_flying_count AS  
SELECT pt.id, pt.made, COUNT(*) AS flying_count  
FROM plane p  
JOIN plane_type pt ON p.id_type = pt.id  
JOIN ticket t ON t.plane = p."number"  
WHERE t.buy_date >= CURRENT_DATE - INTERVAL '1 month'  
GROUP BY pt.id, pt.made;
```

```
SELECT * FROM plane_type_flying_count;
```

## 4. Индексы

### Просто

```
1 SET search_path TO Аэропорт;  
2  
3 CREATE INDEX flight_number_idx ON "Аэропорт".schedule (day) WHERE day = 'Monday';  
4
```

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 55 msec.

### Составной

```
1 SET search_path TO Аэропорт;  
2  
3 CREATE INDEX schedule_request_idx ON schedule (airport_from, airport_to, "Время прибытия", "Время вылета");
```

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 39 msec.

## ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы были созданы запросы и представления на выборку данных к базе данных PostgreSQL согласно индивидуальному заданию, часть 2 и 3. Были созданы 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) с использованием подзапросов. Были изучены графические представления запросов. Были созданы простой и составной индексы для двух произвольных запросов.

Таким образом, за выполнение данной лабораторной работы удалось познакомиться с представлениями и индексами и успешно их реализовать. Также были отработаны навыки выполнения запросов на индивидуальных заданиях. Успешно были реализованы различные модификации данных с подзапросами.

Индексы при больших запросах позволили значительно выиграть время выполнения, план запроса остался тем же.