

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИТМО»**

**Факультет инфокоммуникационных технологий**

**Дисциплина:**

**«Проектирование и реализация баз данных»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

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

**Выполнил:**

студент группы К32391

Микитчак Иван Михайлович

---

(подпись)

**Проверил:**

Говорова Марина Михайловна

---

(отметка о выполнении)

---

(подпись)

Санкт-Петербург

2023 г.

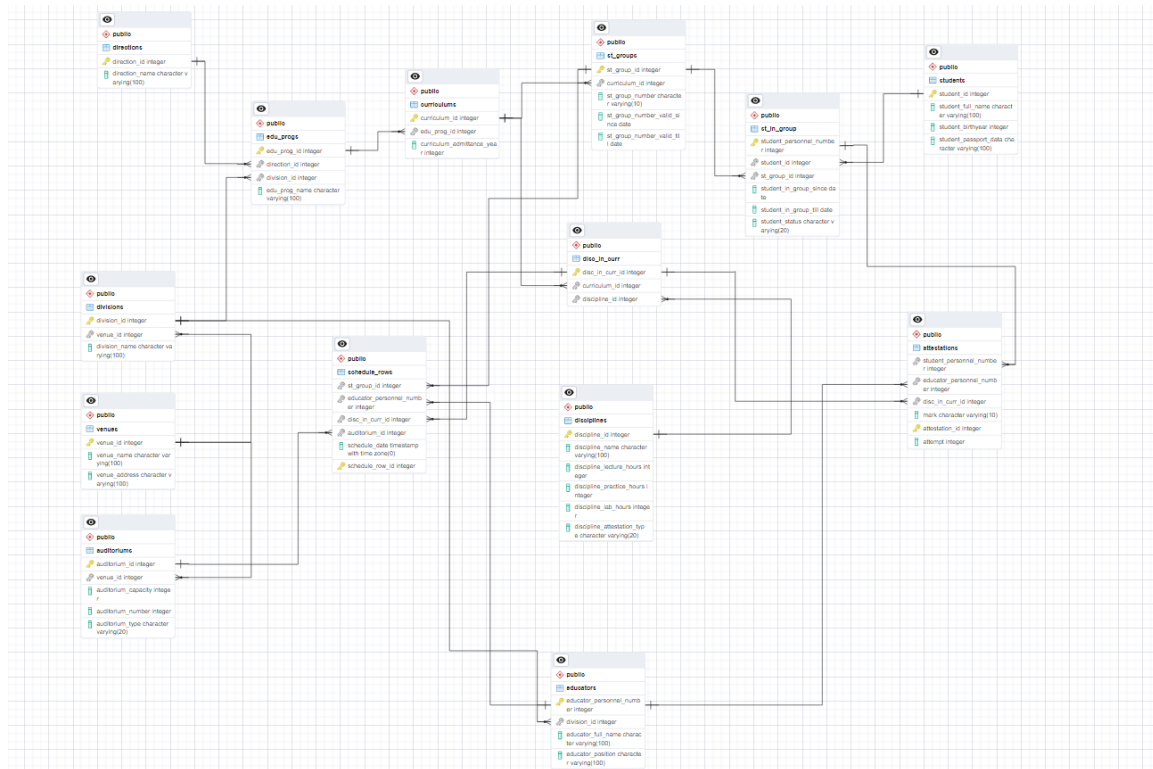
## **Цель работы:**

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.
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.

## Схема базы данных



## Выполнение

### Запросы к базе данных

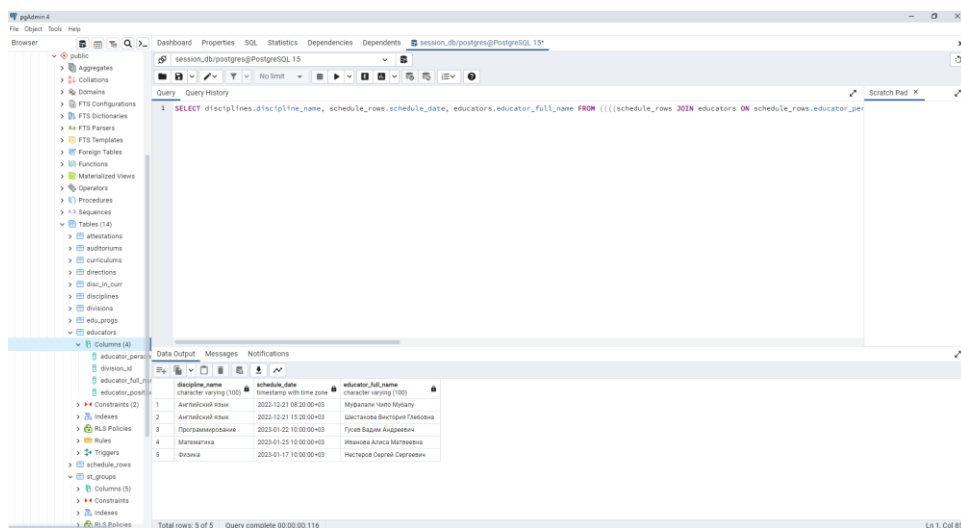
#### 1

Составить список дисциплин, которые должны быть сданы заданной группой с указанием дат сдачи и фамилий преподавателей.

Команда:

```
SELECT disciplines.discipline_name, schedule_rows.schedule_date,  
educators.educator_full_name FROM (((schedule_rows JOIN educators ON  
schedule_rows.educator_personnel_number = educators.educator_personnel_number) JOIN  
st_groups ON schedule_rows.st_group_id = st_groups.st_group_id) JOIN disc_in_curr ON  
schedule_rows.disc_in_curr_id = disc_in_curr.disc_in_curr_id) JOIN disciplines on  
disc_in_curr.discipline_id = disciplines.discipline_id) WHERE st_groups.st_group_number =  
'K32391';
```

Результат:



The screenshot shows the pgAdmin 4 interface. The SQL editor contains the query from the previous block. The 'Data Output' pane at the bottom displays the results of the query, which are 5 rows of data. The columns are discipline\_name, schedule\_date, and educator\_full\_name.

discipline_name	schedule_date	educator_full_name
Автоматизация	2023-12-21 09:20:00+03	Медведева Анна Михайловна
Автоматизация	2023-12-21 13:20:00+03	Щербатова Валерия Геннадьевна
Программирование	2023-01-22 10:00:00+03	Гусев Вадим Александрович
Математика	2023-01-23 10:00:00+03	Иванова Анна Михайловна
Физика	2023-01-17 10:00:00+03	Исупов Сергей Сергеевич

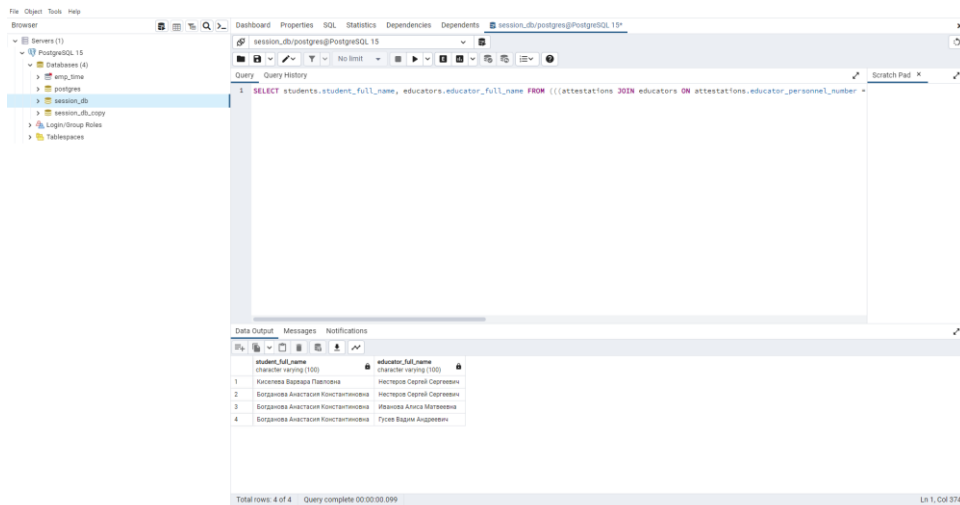
#### 2

Вывести список студентов, получивших двойки на первой попытке с указанием фамилии преподавателя, которым они должны пересдать экзамен.

Команда:

```
SELECT students.student_full_name, educators.educator_full_name FROM (((attestations  
JOIN educators ON attestations.educator_personnel_number =  
educators.educator_personnel_number) JOIN st_in_group ON  
attestations.student_personnel_number = st_in_group.student_personnel_number) JOIN  
students on st_in_group.student_id = students.student_id) WHERE attestations.mark = '2F';
```

## Результат:



The screenshot shows a PostgreSQL query editor with a query window and a results window. The query is:

```
SELECT students.student_full_name, educators.educator_full_name FROM (((attestations JOIN educators ON attestations.educator_personnel_number = educators.educator_personnel_number
```

The results window shows the following data:

student_full_name	educator_full_name
Иванова Александра Павловна	Иванова Мария Сергеевна
Борданова Анастасия Константиновна	Иванова Мария Сергеевна
Борданова Анастасия Константиновна	Иванова Мария Сергеевна
Борданова Анастасия Константиновна	Гусев Вадим Андреевич

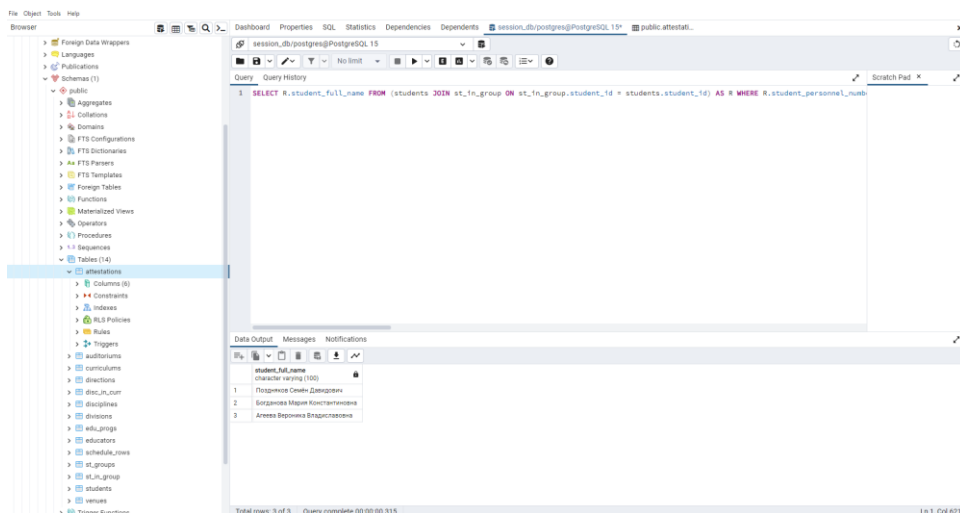
3

Вывести фамилии студентов, получивших оценки по дисциплине, которые выше среднего балла по этой дисциплине.

Команда:

```
SELECT R.student_full_name FROM (students JOIN st_in_group ON st_in_group.student_id = students.student_id) AS R WHERE R.student_personnel_number IN (SELECT T.student_personnel_number FROM (SELECT *, CASE WHEN mark='2F' THEN 2 WHEN mark='3D' THEN 3 WHEN mark='4C' THEN 4 WHEN mark='4B' THEN 4 WHEN mark='5A' THEN 5 ELSE NULL END mark_to_num FROM attestations) AS T WHERE T.disc_in_curr_id = 2 AND T.mark_to_num > (SELECT AVG(mark_to_num) FROM (SELECT *, CASE WHEN mark='2F' THEN 2 WHEN mark='3D' THEN 3 WHEN mark='4C' THEN 4 WHEN mark='4B' THEN 4 WHEN mark='5A' THEN 5 ELSE NULL END mark_to_num FROM attestations) AS T))
```

## Результат:



The screenshot shows a PostgreSQL query editor with a query window and a results window. The query is:

```
SELECT R.student_full_name FROM (students JOIN st_in_group ON st_in_group.student_id = students.student_id) AS R WHERE R.student_personnel_number
```

The results window shows the following data:

student_full_name
Иванова Александра Павловна
Борданова Анастасия Константиновна
Алексей Александрович

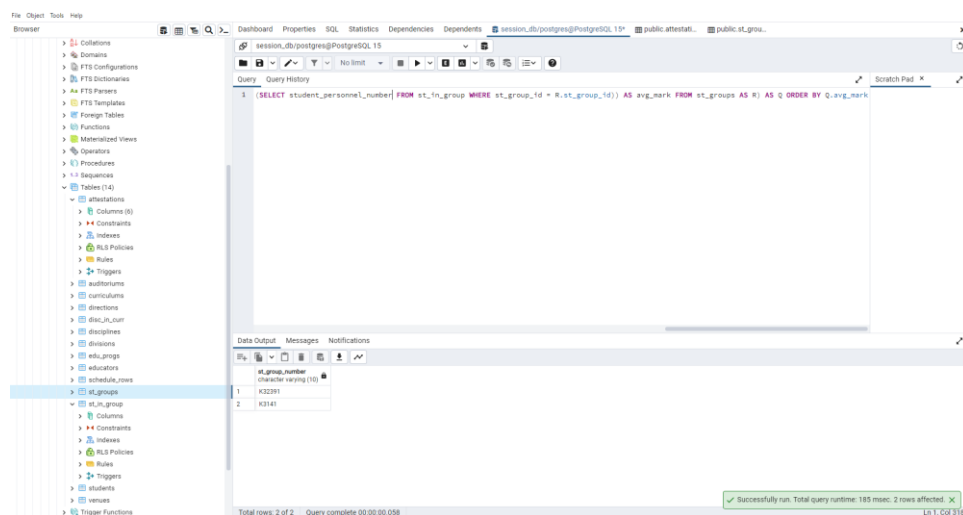
## 4

Формулировка: Создать рейтинговый список групп по заданному направлению по результатам сдачи сессии, упорядочить его по убыванию.

Команда:

```
SELECT Q.st_group_number FROM (SELECT *, (SELECT AVG(T.mark_to_num) FROM
(SELECT *, CASE WHEN mark='2F' THEN 2 WHEN mark='3D' THEN 3 WHEN
mark='4C' THEN 4 WHEN mark='4B' THEN 4 WHEN mark='5A' THEN 5 ELSE NULL
END mark_to_num FROM attestations) AS T WHERE T.student_personnel_number IN
(SELECT student_personnel_number FROM st_in_group WHERE st_group_id =
R.st_group_id)) AS avg_mark FROM st_groups AS R) AS Q ORDER BY Q.avg_mark
```

Результат:



## 5

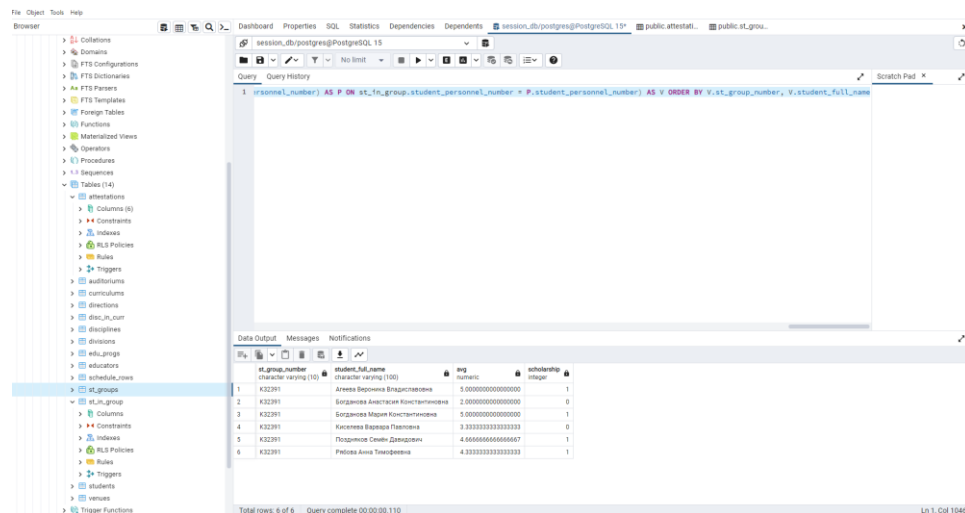
Формулировка: Создайте списки студентов, упорядоченные по группам и фамилиям студентов, содержащие данные о средних баллах и назначении на стипендии. Студент получает стипендию, если он сдал сессию без троек. Если студент не назначен на стипендию, указать 0, если назначен – 1.

Команда:

```
SELECT V.st_group_number, V.student_full_name, V.avg, CASE WHEN V.sum = 0 THEN
1 ELSE 0 END scholarship FROM (((st_in_group JOIN students ON st_in_group.student_id
= students.student_id) JOIN st_groups ON st_in_group.st_group_id = st_groups.st_group_id)
JOIN (SELECT T.student_personnel_number, SUM(T.f) FROM (SELECT *, CASE WHEN
mark='2F' THEN 1 WHEN mark='3D' THEN 1 WHEN mark='4C' THEN 0 WHEN
mark='4B' THEN 0 WHEN mark='5A' THEN 0 WHEN mark='Незачёт' THEN 1 WHEN
mark='Зачёт' THEN 0 ELSE NULL END f FROM attestations) AS T GROUP BY
T.student_personnel_number) AS R ON st_in_group.student_personnel_number =
R.student_personnel_number) JOIN (SELECT Q.student_personnel_number,
```

AVG(Q.mark\_to\_num) FROM (SELECT \*, CASE WHEN mark='2F' THEN 2 WHEN mark='3D' THEN 3 WHEN mark='4C' THEN 4 WHEN mark='4B' THEN 4 WHEN mark='5A' THEN 5 ELSE NULL END mark\_to\_num FROM attestations) AS Q GROUP BY Q.student\_personnel\_number) AS P ON st\_in\_group.student\_personnel\_number = P.student\_personnel\_number) AS V ORDER BY V.st\_group\_number, V.student\_full\_name

Результат:



The screenshot shows a PostgreSQL query editor with a query window and a data output window. The query window contains the following SQL query:

```
1. personnel_number) AS P ON st_in_group.student_personnel_number = P.student_personnel_number) AS V ORDER BY V.st_group_number, V.student_full_name
```

The data output window shows the following results:

st_group_number	student_full_name	avg	scholarship
1	Алексеева Мария Константиновна	5.0000000000000000	1
2	Борисова Александра Константиновна	5.0000000000000000	0
3	Борисова Мария Константиновна	5.0000000000000000	1
4	Киселева Мария Павловна	3.3333333333333333	0
5	Полосинская Елена Дмитриевна	4.6666666666666667	1
6	Петрова Анна Тимуровна	4.3333333333333333	1

Total rows: 6 of 6. Query complete 00:00:00.110. Ln 1, Col 1045

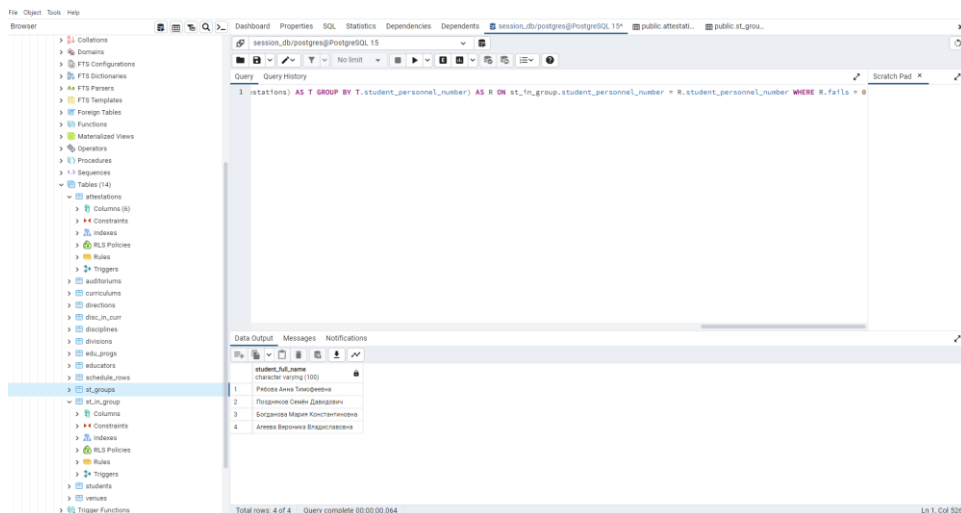
6

Формулировка: Вывести список студентов, сдавших все положенные экзамены.

Команда:

SELECT students.student\_full\_name FROM (st\_in\_group JOIN students ON st\_in\_group.student\_id = students.student\_id) JOIN (SELECT T.student\_personnel\_number, SUM(T.f) AS fails FROM (SELECT \*, CASE WHEN mark='2F' THEN 1 WHEN mark='3D' THEN 1 WHEN mark='4C' THEN 0 WHEN mark='4B' THEN 0 WHEN mark='5A' THEN 0 WHEN mark='Незачёт' THEN 1 WHEN mark='Зачёт' THEN 0 ELSE NULL END f FROM attestations) AS T GROUP BY T.student\_personnel\_number) AS R ON st\_in\_group.student\_personnel\_number = R.student\_personnel\_number WHERE R.fails = 0

Результат:



7

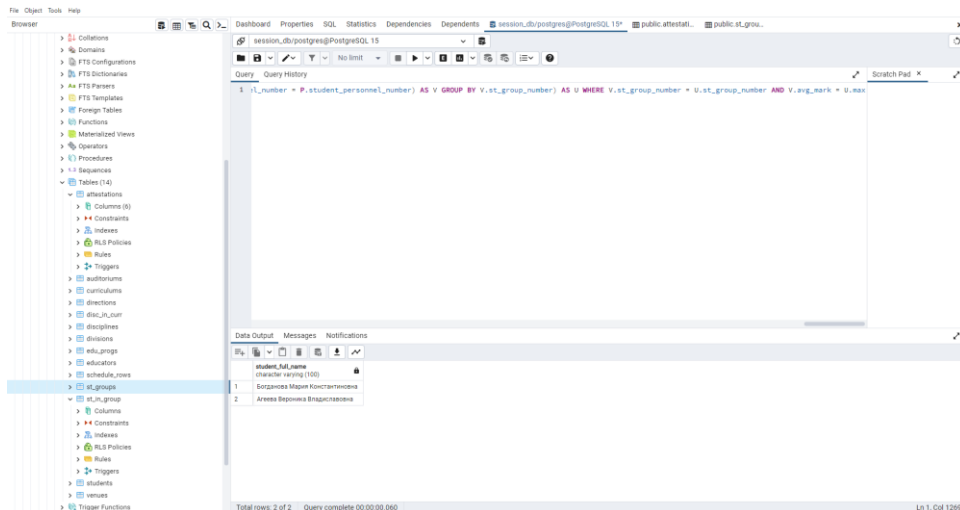
Формулировка: Вывести список студентов, получивших максимальный средний балл в своей группе.

Команда:

```
SELECT V.student_full_name FROM (SELECT st_groups.st_group_number,
students.student_full_name, P.avg_mark FROM ((st_in_group JOIN students ON
st_in_group.student_id = students.student_id) JOIN st_groups ON st_in_group.st_group_id =
st_groups.st_group_id) JOIN (SELECT Q.student_personnel_number,
AVG(Q.mark_to_num) AS avg_mark FROM (SELECT *, CASE WHEN mark='2F' THEN 2
WHEN mark='3D' THEN 3 WHEN mark='4C' THEN 4 WHEN mark='4B' THEN 4 WHEN
mark='5A' THEN 5 ELSE NULL END mark_to_num FROM attestations) AS Q GROUP BY
Q.student_personnel_number) AS P ON st_in_group.student_personnel_number =
P.student_personnel_number) AS V, (SELECT V.st_group_number, MAX(V.avg_mark)
FROM (SELECT st_groups.st_group_number, students.student_full_name, P.avg_mark
FROM ((st_in_group JOIN students ON st_in_group.student_id = students.student_id) JOIN
st_groups ON st_in_group.st_group_id = st_groups.st_group_id) JOIN (SELECT
Q.student_personnel_number, AVG(Q.mark_to_num) AS avg_mark FROM (SELECT *,
CASE WHEN mark='2F' THEN 2 WHEN mark='3D' THEN 3 WHEN mark='4C' THEN 4
WHEN mark='4B' THEN 4 WHEN mark='5A' THEN 5 ELSE NULL END mark_to_num
FROM attestations) AS Q GROUP BY Q.student_personnel_number) AS P ON
st_in_group.student_personnel_number = P.student_personnel_number) AS V GROUP BY
V.st_group_number) AS U WHERE V.st_group_number = U.st_group_number AND
V.avg_mark = U.max
```

Результат:





## Создание представлений

### 1

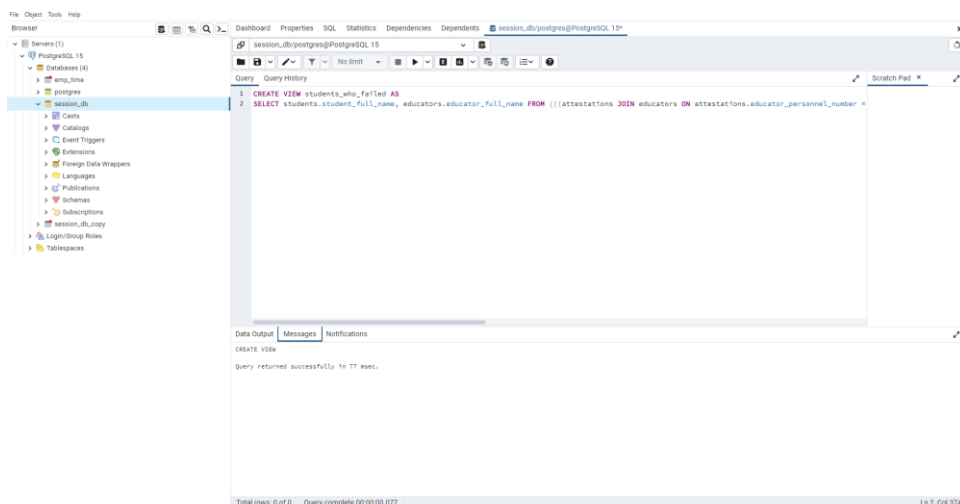
Формулировка: список студентов, получивших двойки на первой попытке с указанием фамилии преподавателя, которым они должны пересдать экзамен;

Команда:

CREATE VIEW students\_who\_failed AS

SELECT students.student\_full\_name, educators.educator\_full\_name FROM (((attestations JOIN educators ON attestations.educator\_personnel\_number = educators.educator\_personnel\_number) JOIN st\_in\_group ON attestations.student\_personnel\_number = st\_in\_group.student\_personnel\_number) JOIN students ON st\_in\_group.student\_id = students.student\_id) WHERE attestations.mark = '2F';

Результат:



Содержимое:

	student_full_name character varying (100)	educator_full_name character varying (100)
1	Киселева Варвара Павловна	Нестеров Сергей Сергеевич
2	Богданова Анастасия Константиновна	Нестеров Сергей Сергеевич
3	Богданова Анастасия Константиновна	Гусев Вадим Андреевич
4	Богданова Анастасия Константиновна	Иванова Алиса Матвеевна

2

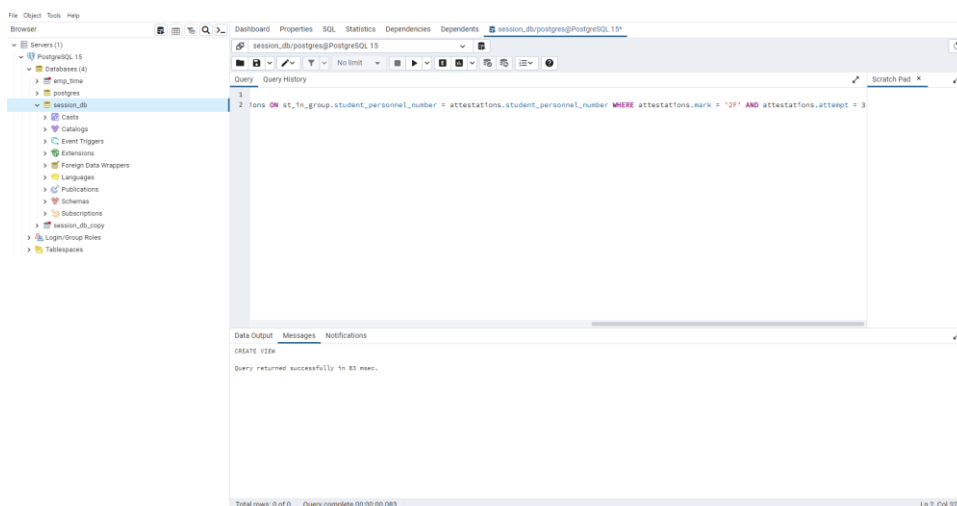
Формулировка: данных о студентах при получении ими хотя бы одной оценки 2 (после 3-й попытки).

Команда:

CREATE VIEW students\_who\_failed\_really\_hard AS

```
SELECT DISTINCT students.student_full_name, st_in_group.student_personnel_number
FROM (st_in_group JOIN students ON st_in_group.student_id = students.student_id) JOIN
attestations ON st_in_group.student_personnel_number =
attestations.student_personnel_number WHERE attestations.mark = '2F' AND
attestations.attempt = 3
```

Результат:



Содержимое:

<b>student_full_name</b> character varying (100) 🔒	<b>student_personnel_number</b> integer 🔒
---	--

## Запросы на модификацию данных

1

Формулировка: Вставить лишние данные о человеке в attestations

Команда:

INSERT INTO attestations (attestation\_id, student\_personnel\_number, educator\_personnel\_number, disc\_in\_curr\_id, mark, attempt)

VALUES (25, (SELECT st\_in\_group.student\_personnel\_number FROM st\_in\_group JOIN students ON st\_in\_group.student\_id = students.student\_id WHERE students.student\_full\_name = 'Агеева Вероника Владиславовна'), (SELECT educator\_personnel\_number FROM educators WHERE educator\_full\_name = 'Муфалали Чипо Мубалу'), 4, 'Незачёт', 1)

Результат:

The screenshot shows the PostgreSQL pgAdmin interface. The left sidebar displays the database structure, including the 'public' schema and the 'attestations' table. The main window shows the SQL query history with the following query:

```
INSERT INTO attestations (attestation_id, student_personnel_number, educator_personnel_number, disc_in_curr_id, mark, attempt)
VALUES (25, (SELECT st_in_group.student_personnel_number FROM st_in_group JOIN students ON st_in_group.student_id = students.student_id WHERE students.student_full_name = 'Агеева Вероника Владиславовна'), (SELECT educator_personnel_number FROM educators WHERE educator_full_name = 'Муфалали Чипо Мубалу'), 4, 'Незачёт', 1)
```

The query was executed successfully on 21.05.2023 at 14:44:06, affecting 1 row in 53 milliseconds. The 'Messages' pane shows: 'Query returned successfully in 53 msec.' Below the query, the 'Data Output' pane displays the contents of the 'attestations' table:

attestation_id	student_personnel_number	educator_personnel_number	disc_in_curr_id	mark	attempt
17	333005	111111	2	SA	1
18	333005	111222	1	SA	1
19	333005	111333	3	SA	1
20	333005	111444	4	Over	1
21	333005	111111	2	AC	1
22	333005	111222	1	AB	1
23	333005	111333	3	SA	1
24	333005	111444	4	Over	1
25	333005	111555	4	mezachet	1

Total rows: 25 of 25. Query complete 00:00:00.314. Lin 1, Col 1.

2

Формулировка: Обновить попытку конкретного человека в attestations

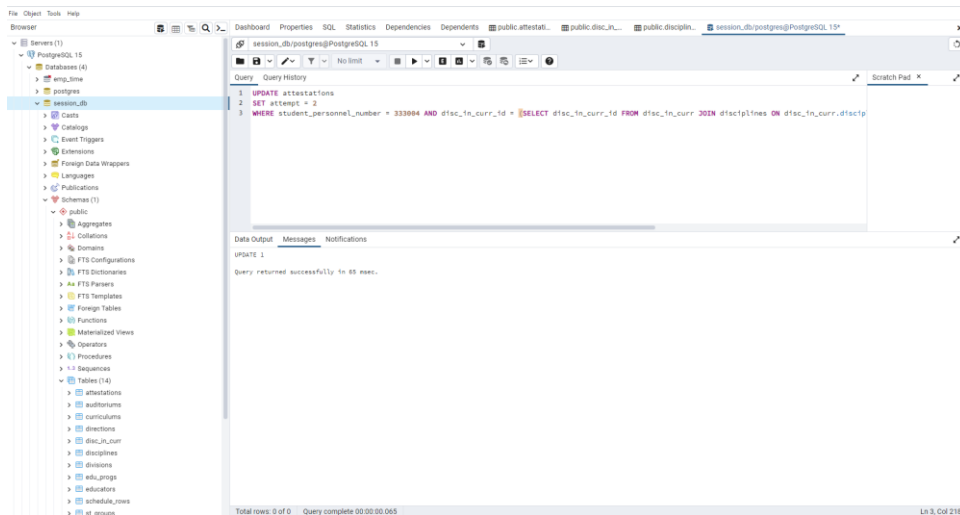
Команда:

UPDATE attestations

SET attempt = 2

WHERE student\_personnel\_number = 333004 AND disc\_in\_curr\_id = SELECT  
disc\_in\_curr\_id FROM disc\_in\_curr JOIN disciplines ON disc\_in\_curr.discipline\_id =  
disciplines.discipline\_id WHERE discipline\_name = 'Математика'

Результат:



3

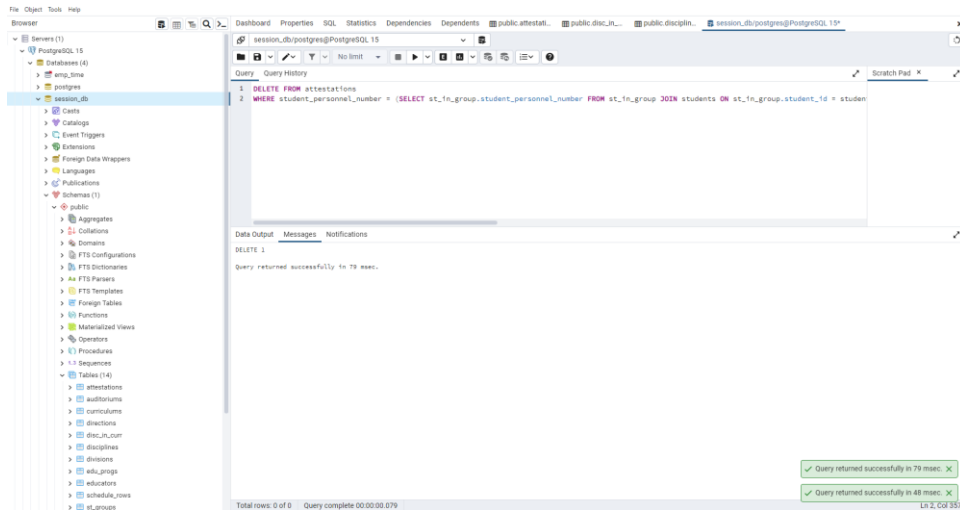
Формулировка: Удалить лишние данные о человеке в attestations

Команда:

DELETE FROM attestations

WHERE student\_personnel\_number = (SELECT st\_in\_group.student\_personnel\_number  
FROM st\_in\_group JOIN students ON st\_in\_group.student\_id = students.student\_id WHERE  
students.student\_full\_name = 'Агеева Вероника Владиславовна') AND  
educator\_personnel\_number = (SELECT educator\_personnel\_number FROM educators  
WHERE educator\_full\_name = 'Муфалали Чипо Мубалу')

Результат:



## Индексы

### 1

Мы попытаемся выполнить следующий запрос без индекса:

```
SELECT * FROM students WHERE student_full_name = 'AAA AAA AAA'
```

План выполнения запроса:



Результат работы:

22.05.2023 13:32:08	6	51 msec
Date	Rows affected	Duration

Copy Copy to Query Editor

```
SELECT * FROM students WHERE student_full_name = 'AAA AAA AAA'
```

Messages

Successfully run. Total query runtime: 51 msec. 6 rows affected.

Создадим индекс:

```
CREATE INDEX idx_student_full_name ON students (student_full_name)
```

План выполнения того же запроса:



Результат выполнения старого запроса:

22.05.2023 13:36:43	6	106 msec
Date	Rows affected	Duration

Copy

Copy to Query Editor

```
SELECT * FROM students WHERE student_full_name = 'AAA AAA AAA'
```

Messages

Successfully run. Total query runtime: 106 msec. 6 rows affected.

2

Мы попытаемся выполнить следующий запрос без индекса:

```
SELECT * FROM auditoriums WHERE auditorium_capacity > 50 AND auditorium_type = 'Лекционный зал'
```

План выполнения запроса:

auditoriums

Successfully run. Total query runtime: 83 msec. 16562 rows affected.

Результат работы:

22.05.2023 13:49:16	16562	83 msec
Date	Rows affected	Duration

Copy

Copy to Query Editor

```
SELECT * FROM auditoriums WHERE auditorium_capacity > 50 AND auditorium_type = 'Лекционный зал'
```

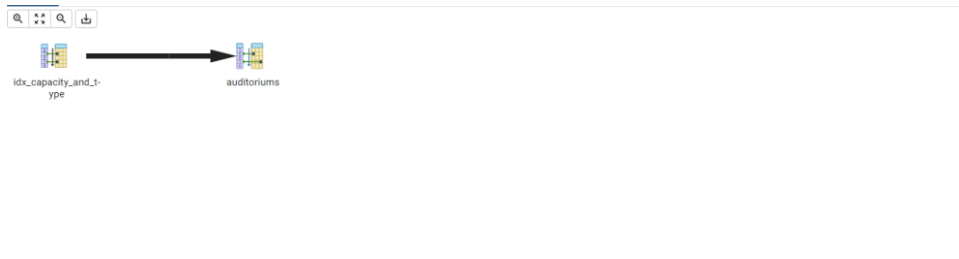
Messages

Successfully run. Total query runtime: 83 msec. 16562 rows affected.

Создадим индекс:

```
CREATE INDEX idx_capacity_and_type ON auditoriums (auditorium_capacity,  
auditorium_type)
```

План выполнения того же запроса:



Результат выполнения старого запроса:

22.05.2023 13:53:00	16562	69 msec
Date	Rows affected	Duration
<div>Copy Copy to Query Editor</div> <pre>SELECT * FROM auditoriums WHERE auditorium_capacity &gt; 50 AND auditorium_type = 'Лект</pre>		
Messages		
Successfully run. Total query runtime: 69 msec. 16562 rows affected.		

Как видно, в первом случае запрос без индекса оказался быстрее. Во втором же случае быстрее оказался запрос с индексом.

Удалим индексы:

Today - 22.05.2023
► DROP INDEX idx_capacity_and_type
13:56:20
► DROP INDEX idx_student_full_name
13:56:07

## **Выводы**

Мы овладели практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.