

**Министерство науки и высшего образования Российской Федерации  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«Национальный исследовательский университет ИТМО»  
(Университет ИТМО)**

**Факультет прикладной информатики**

**Образовательная программа Мобильные и сетевые технологии**

**Направление подготовки 09.03.03 Мобильные и сетевые технологии**

**О Т Ч Е Т  
ЛАБОРАТОРНАЯ РАБОТА №4**

**"ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ.  
ПРЕДСТАВЛЕНИЯ. РАБОТА С ИНДЕКСАМИ"**

**Обучающийся:** Майстренко Анастасия К3241

**Преподаватель:** Говорова Марина Михайловна

Санкт-  
Петербург,  
2025

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

## 2. Схема базы данных (ЛР 3).



### 3. Выполнение:

#### 3.1 Запросы к базе данных.

В рамках выполнения лабораторной работы были составлены и выполнены SQL-запросы в соответствии с индивидуальным заданием (часть 2). Каждый запрос формировался исходя из требований задания, отражающих конкретные задачи по выборке данных из базы.

Запрос 1: Составить список всех заданий каждого проекта с указанием организаций, отделов и исполнителей, занятых в его выполнении.

Query

Query History

```
1  ✓ SELECT
2      pr.name AS "Проект",
3      t.task_name AS "Задание",
4      org.name AS "Организация",
5      d.name AS "Отдел исполнителя",
6      e.last_name || ' ' || e.first_name AS "Исполнитель"
7  FROM lab3_schema.project pr
8  JOIN lab3_schema.organization org ON pr.organization_id = org.organization_id
9  JOIN lab3_schema.task t ON t.project_id = pr.project_id
10 JOIN lab3_schema.task_participation tp ON t.task_id = tp.task_id
11 JOIN lab3_schema.employee e ON tp.employee_id = e.employee_id
12 JOIN lab3_schema.department d ON e.department_id = d.department_id
13 ORDER BY pr.name, t.task_name, e.last_name;
14
15
16
17
18
19
```

Data Output

Messages

Notifications

⊞

📄

▼

📋

🗑

🗑

📥

⬇

📈

SQL

Showing rows: 1 to 2

	Проект character varying (100) 🔒	Задание character varying (100) 🔒	Организация character varying (100) 🔒	Отдел исполнителя character varying (100) 🔒	Исполнитель text 🔒
1	Проект А	Анализ требований	ООО Альфа	Отдел аналитики	Петров Пётр
2	Проект А	Разработка прототипа	ООО Альфа	Отдел аналитики	Петров Пётр

Этот запрос соединяет таблицы проектов, заданий, организаций, сотрудников и отделов.

Каждое задание связано с проектом, сотрудником-исполнителем и его отделом.

Используем JOIN, чтобы объединить все эти данные в один список.

ORDER BY упорядочивает вывод по проекту и фамилии исполнителя.

Результат — полный перечень **кто над каким заданием какого проекта работает**, с указанием отдела и организации.

Запрос 2: Составить список проектов, работа над которыми была начата больше месяца назад.

Query Query History

```
1  SELECT
2      project_id,
3      name AS "Проект",
4      start_date,
5      end_date
6  FROM lab3_schema.project
7  WHERE start_date < CURRENT_DATE - INTERVAL '1 month';
8
9
10
11
12
13
14
15
16
```

Data Output Messages Notifications

	project_id [PK] integer	Проект character varying (100)	start_date date	end_date date
1	5	Проект В	2025-04-01	2025-08-01

Мы выбираем все проекты, дата начала которых (start\_date) была **раньше, чем месяц назад от текущей даты**.

CURRENT\_DATE - INTERVAL '1 month' автоматически вычисляет дату месяц назад.

Это полезно для выявления **долгосрочных или «старых» проектов**, которые требуют контроля или завершения.

В запросе указываются ID, название, дата начала и окончания таких проектов.

Запрос 3: Вывести список сотрудников, оклад которых превышает средний оклад сотрудников своего отдела.

Query

Query History

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

SELECT

e.employee\_id,

e.last\_name || ' ' || e.first\_name AS "Сотрудник",

d.name AS "Отдел",

e.salary AS "Оклад",

ROUND(avg\_salary\_by\_dept, 2) AS "Средний по отделу"

FROM (

SELECT \*,

AVG(salary) OVER (PARTITION BY department\_id) AS avg\_salary\_by\_dept

FROM lab3\_schema.employee

) e

JOIN lab3\_schema.department d ON e.department\_id = d.department\_id

WHERE e.salary > e.avg\_salary\_by\_dept

ORDER BY d.name, e.salary DESC;

Data Output

Messages

Notifications

+

📄

▼

📋

▼

🗑️

🗑️

📥

⬇️

📈

SQL

Showing 1

	employee_id integer	Сотрудник text	Отдел character varying (100)	Оклад numeric	Средний по отделу numeric
1	21	Богатый Иван	Отдел разработки	100000	80000.00

Этот запрос позволяет найти сотрудников, **чей оклад выше среднего значения среди сотрудников их отдела**.

1. Для каждого отдела мы находим **средний оклад** всех сотрудников.
2. Затем проверяем каждого сотрудника: если его оклад **выше среднего**, то выводим его в результат.

Он **считает средний оклад** в каждом отделе (например, если в отделе 5 сотрудников с разными окладами, он вычислит среднее).

Потом **сравнивает оклад каждого сотрудника с этим средним**.

Если оклад больше, то сотрудник попадает в результат.

Запрос 4: Найти отдел, работающий над максимальным количеством проектов.

The screenshot shows a SQL query editor with a query that selects the department name and the count of projects, ordered by the count in descending order, limited to 1 result. Below the query, the 'Data Output' tab shows a single row of results.

```
1 SELECT
2     d.name AS "Отдел",
3     COUNT(pp.project_id) AS "Количество проектов"
4 FROM lab3_schema.department d
5 JOIN lab3_schema.employee e ON d.department_id = e.department_id
6 JOIN lab3_schema.project_participation pp ON e.employee_id = pp.employee_id
7 GROUP BY d.name
8 ORDER BY COUNT(pp.project_id) DESC
9 LIMIT 1;
```

	Отдел character varying (100)	Количество проектов bigint
1	Отдел аналитики	2

**JOIN** объединяет таблицы:

- department (отделы),
- employee (сотрудники),
- project\_participation (сотрудники, участвующие в проектах).

**COUNT(pp.project\_id)** подсчитывает, сколько проектов связано с каждым отделом.

Мы **сортируем** по количеству проектов и **выводим только 1** отдел с наибольшим количеством (LIMIT 1).

Запрос 5: Составить список сотрудников, проектов, заданий, в выполнении которых они участвуют и дат предполагаемого выполнения ими заданий. Учесть сотрудников, не участвующих в проектах.

Query Query History

```
1 SELECT
2   e.last_name || ' ' || e.first_name AS "Сотрудник",
3   pr.name AS "Проект",
4   t.task_name AS "Задание",
5   t.start_date AS "Дата начала задания",
6   tp.end_date AS "Дата окончания задания"
7 FROM lab3_schema.employee e
8 LEFT JOIN lab3_schema.task_participation tp ON e.employee_id = tp.employee_id
9 LEFT JOIN lab3_schema.task t ON tp.task_id = t.task_id
10 LEFT JOIN lab3_schema.project pr ON t.project_id = pr.project_id
11 ORDER BY e.last_name, pr.name, t.task_name;
```

Data Output Messages Notifications

Showing rows: 1 to 4

	Сотрудник text	Проект character varying (100)	Задание character varying (100)	Дата начала задания date	Дата окончания задания date
1	Богатый Иван	[null]	[null]	[null]	[null]
2	Иванов Иван	[null]	[null]	[null]	[null]
3	Петров Пётр	Проект А	Анализ требований	2025-06-17	2025-07-17
4	Петров Пётр	Проект А	Разработка прототипа	2025-07-22	2025-08-21

Мы хотим получить **список сотрудников**, а также информацию о **проектах и заданиях**, в которых они участвуют.

Если сотрудник **не участвует** в проекте, то его **имя** будет в списке, но проект и задание — пустые (используем LEFT JOIN).

- **e.last\_name || ' ' || e.first\_name AS "Сотрудник"**:

Это объединяет фамилию и имя сотрудника в один столбец, например, "**Иванов Иван**".

|| — это оператор конкатенации строк в SQL.

- **pr.name AS "Проект"**:

Получаем название проекта, в котором участвует сотрудник.

- **t.task\_name AS "Задание"**:

Получаем название задания, в котором участвует сотрудник.

**- t.start\_date AS "Дата начала задания" и tp.end\_date AS "Дата окончания задания":**

Выводим даты начала и окончания задания, для того чтобы понять сроки выполнения.

**- LEFT JOIN lab3\_schema.task\_participation tp ON e.employee\_id = tp.employee\_id:**

LEFT JOIN означает, что мы **включим всех сотрудников**, даже если они **не участвуют** в заданиях.

Мы соединяем таблицу сотрудников с таблицей task\_participation, которая связывает сотрудников с заданиями.

**- LEFT JOIN lab3\_schema.task t ON tp.task\_id = t.task\_id:**

Соединяем таблицу с заданиями, чтобы для каждого сотрудника получить название задания.

**- LEFT JOIN lab3\_schema.project pr ON t.project\_id = pr.project\_id:**

Присоединяем таблицу с проектами, чтобы вывести название проекта.

**- ORDER BY e.last\_name, pr.name, t.task\_name:**

Сортируем результат по фамилии сотрудника, названию проекта и названию задания.









Запрос 6: Составить список сотрудников, не выполнивших задания в срок с указанием проектов и заданий, которые они должны были выполнить и количества дней просрочки выполнения заданий.

Query Query History

1 SELECT  
2 e.last\_name || ' ' || e.first\_name AS "Сотрудник",  
3 pr.name AS "Проект",  
4 t.task\_name AS "Задание",  
5 tp.end\_date AS "Дата окончания задания",  
6 CURRENT\_DATE - tp.end\_date AS "Просрочка (дни)"  
7 FROM lab3\_schema.employee e  
8 JOIN lab3\_schema.task\_participation tp ON e.employee\_id = tp.employee\_id  
9 JOIN lab3\_schema.task t ON tp.task\_id = t.task\_id  
10 JOIN lab3\_schema.project pr ON t.project\_id = pr.project\_id  
11 WHERE tp.end\_date < CURRENT\_DATE  
12 ORDER BY e.last\_name, pr.name, t.task\_name;  
13  
14  
15

Data Output Messages Notifications

      SQL

Showing rows: 1 of 1

	Сотрудник text	Проект character varying (100)	Задание character varying (100)	Дата окончания задания date	Просрочка (дни) integer
1	Петров Пётр	Проект А	Анализ требований	2025-06-10	10

- **e.last\_name || ' ' || e.first\_name AS "Сотрудник":**

Это объединяет фамилию и имя сотрудника в одну строку, например: **Иванов Иван**.

Используем || для **конкатенации** строк.

- **pr.name AS "Проект":**

Показывает название проекта, в котором участвует сотрудник.

- **t.task\_name AS "Задание":**

Выводит название задания, над которым работает сотрудник.

- **tp.end\_date AS "Дата окончания задания":**

Отображает **дату окончания задания**. Это дата, когда сотрудник должен был завершить задание.

9

**- CURRENT\_DATE - tp.end\_date AS "Просрочка (дни)":**

Вычисляем, сколько дней прошло с момента **плановой даты завершения задания** (tp.end\_date).

Если дата выполнения задания была в прошлом (например, tp.end\_date = 2025-06-10, а сегодня 2025-06-20), то результат будет **10 дней просрочки**.

**- JOIN lab3\_schema.task\_participation tp ON e.employee\_id = tp.employee\_id:**

Мы соединяем таблицу **сотрудников** с таблицей **участия в заданиях**, чтобы понять, какие сотрудники участвовали в каких заданиях.

**- JOIN lab3\_schema.task t ON tp.task\_id = t.task\_id:**

Соединяем таблицу с заданиями, чтобы получить информацию о заданиях, которые выполняются сотрудниками.

**- JOIN lab3\_schema.project pr ON t.project\_id = pr.project\_id:**

Присоединяем таблицу с проектами, чтобы вывести название проекта, к которому относится задание.

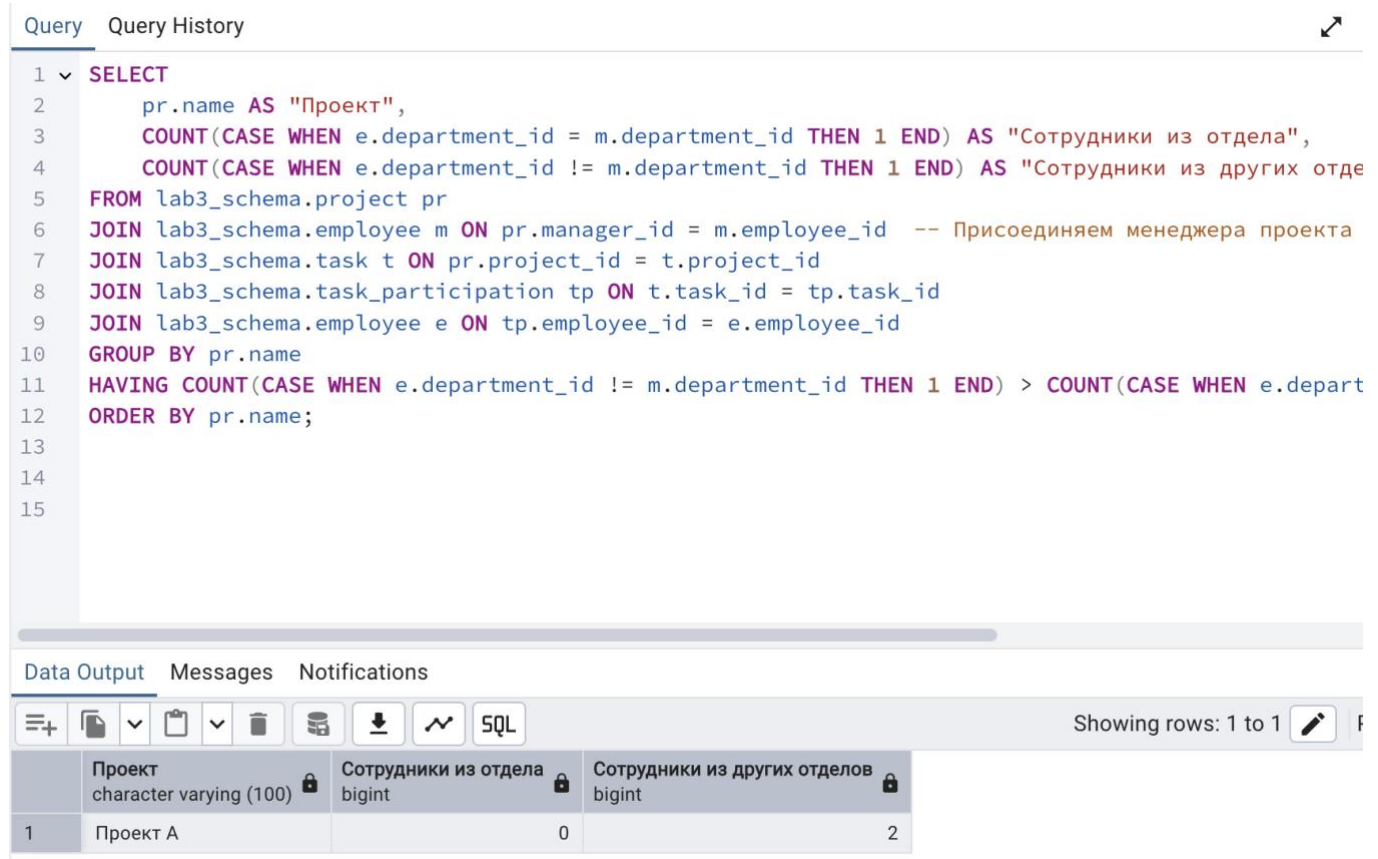
**- WHERE tp.end\_date < CURRENT\_DATE:**

Отбираем только те задания, которые не были завершены в срок, т.е. где **end\_date < CURRENT\_DATE**.

**- ORDER BY e.last\_name, pr.name, t.task\_name:**

Сортируем результат по фамилии сотрудника, названию проекта и задания, чтобы был **удобный порядок**.

Запрос 7: Вывести список проектов, в которых количество сотрудников сторонних отделов участвующих в проекте, превышает количество сотрудников отдела, за которым закреплен проект.



The screenshot shows a SQL IDE interface. The top panel displays a SQL query. The bottom panel shows the results of the query in a table format.

**Query:**

```
1 SELECT
2     pr.name AS "Проект",
3     COUNT(CASE WHEN e.department_id = m.department_id THEN 1 END) AS "Сотрудники из отдела",
4     COUNT(CASE WHEN e.department_id != m.department_id THEN 1 END) AS "Сотрудники из других отде
5 FROM lab3_schema.project pr
6 JOIN lab3_schema.employee m ON pr.manager_id = m.employee_id -- Присоединяем менеджера проекта
7 JOIN lab3_schema.task t ON pr.project_id = t.project_id
8 JOIN lab3_schema.task_participation tp ON t.task_id = tp.task_id
9 JOIN lab3_schema.employee e ON tp.employee_id = e.employee_id
10 GROUP BY pr.name
11 HAVING COUNT(CASE WHEN e.department_id != m.department_id THEN 1 END) > COUNT(CASE WHEN e.depart
12 ORDER BY pr.name;
```

**Data Output:**

	Проект character varying (100)	Сотрудники из отдела bigint	Сотрудники из других отделов bigint
1	Проект А	0	2

### Присоединение таблиц:

Мы соединяем таблицы **project**, **task**, **task\_participation**, **employee**.

Каждая таблица содержит информацию о:

- **Проекте**,
- **Заданиях** в этом проекте,
- **Сотрудниках**, которые выполняют эти задания.

### Определение отдела проекта:

Мы считаем, что проект связан с **отделом менеджера** проекта. Менеджер хранится в таблице **employee** и имеет свой **department\_id**, который и будет указывать на отдел, к которому принадлежит проект.

### Считаем сотрудников:

Мы подсчитываем:

- **Сотрудников, которые принадлежат отделу менеджера** (считаем их для каждого проекта).

- **Сотрудников из других отделов** (считаем сотрудников, чей department\_id не совпадает с отделом менеджера).

#### **Условие HAVING:**

Мы отбираем только те проекты, где **сотрудников из других отделов** больше, чем из отдела менеджера.

#### **COUNT(CASE ...):**

Это **условное подсчётное выражение**, которое считает количество сотрудников:

- **Из отдела менеджера** — если e.department\_id = m.department\_id.

- **Из других отделов** — если e.department\_id != m.department\_id.

#### **Сортировка по проекту:**

Мы выводим список проектов, отсортированных по **названию проекта**.

## 3.2 Представления

Представление 1: для руководителей проектов, содержащее сведения об исполнителях, отделах, сроках выполнения заданий, включенных в проект.

Query Query History

```
1 CREATE VIEW lab3_schema.project_leaders AS
2 SELECT
3     pr.name AS "Проект",
4     e.last_name || ' ' || e.first_name AS "Исполнитель",
5     d.name AS "Отдел исполнителя",
6     t.task_name AS "Задание",
7     t.start_date AS "Дата начала задания",
8     tp.end_date AS "Дата окончания задания", -- Используем tp.end_date
9     m.last_name || ' ' || m.first_name AS "Руководитель проекта"
10 FROM lab3_schema.project pr
11 JOIN lab3_schema.employee m ON pr.manager_id = m.employee_id -- Руководитель проекта
12 JOIN lab3_schema.task t ON pr.project_id = t.project_id -- Задания в проекте
13 JOIN lab3_schema.task_participation tp ON t.task_id = tp.task_id -- Участники заданий
14 JOIN lab3_schema.employee e ON tp.employee_id = e.employee_id -- Исполнители заданий
15 JOIN lab3_schema.department d ON e.department_id = d.department_id -- Отделы исполнителей
16 ORDER BY pr.name, t.task_name, e.last_name;
17
18
19
```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 49 msec.

Data Output Messages Notifications

Showing rows: 1 to 2 Page No: 1 of 1

	Проект character varying (100)	Исполнитель text	Отдел исполнителя character varying (100)	Задание character varying (100)	Дата начала задания date	Дата окончания задания date	Руководитель проекта text
1	Проект А	Петров Пётр	Отдел аналитики	Анализ требований	2025-06-17	2025-06-10	Иванов Иван
2	Проект А	Петров Пётр	Отдел аналитики	Разработка прототипа	2025-07-22	2025-08-21	Иванов Иван

Это представление собирает информацию о **сотрудниках**, которые участвуют в **задачах** проектов.

Оно **выводит фамилию и имя сотрудника**, название проекта, название задания, даты начала и окончания задания.

Также рассчитывается **просрочка выполнения задания**, т.е. сколько дней прошло с момента окончания задания (на основе текущей даты).

Используется **LEFT JOIN** для того, чтобы включить все задания, а не только те, которые имеют выполнения (чтобы учесть сотрудников, которые могут не участвовать в каких-то заданиях).

Представление 2: список проектов, срок выполнения которых истекает сегодня и которые включают больше трех невыполненных заданий.

Query Query History

```
1 CREATE VIEW lab3_schema.projects_due_today AS
2 SELECT
3     pr.name AS "Проект",
4     COUNT(t.task_id) AS "Количество заданий",
5     SUM(CASE WHEN t.status != 'Завершено' THEN 1 ELSE 0 END) AS "Невыполненные задания"
6 FROM lab3_schema.project pr
7 JOIN lab3_schema.task t ON pr.project_id = t.project_id
8 JOIN lab3_schema.task_participation tp ON t.task_id = tp.task_id -- Добавляем JOIN с task_parti
9 WHERE tp.end_date = CURRENT_DATE -- Теперь используем tp.end_date
10 GROUP BY pr.project_id, pr.name
11 HAVING COUNT(t.task_id) > 3 AND
12     SUM(CASE WHEN t.status != 'Завершено' THEN 1 ELSE 0 END) > 3;
13
14
```

Data Output Messages Notifications

CREATE VIEW

Query returned successfully in 22 msec.

Data Output Messages Notifications

	Проект character varying (100)	Количество заданий bigint	Невыполненные задания bigint
1	Проект D	4	4

Это представление собирает информацию о проектах, которые имеют **больше трёх невыполненных заданий**, срок выполнения которых **истекает сегодня**.

Используется **JOIN** для объединения таблиц: проект, задание и участие сотрудников.

Фильтрация происходит по тому, что **end\_date** заданий равна **CURRENT\_DATE**, то есть срок окончания заданий — **сегодня**.

Также присутствует условие в **HAVING**, что в проекте должно быть **больше 3 заданий** и **больше 3 невыполненных заданий**.

### 3.3 Запросы на модификацию данных

Выполнение запросов на модификацию данных (INSERT, UPDATE, DELETE)

#### Запрос на добавление данных (INSERT)

Задача: Добавить нового сотрудника в таблицу **employee**. Мы будем использовать подзапрос, чтобы добавить сотрудника только в том случае, если в проекте есть **невыполненные задания**.

ДО

	employee_id [PK] integer	last_name character varying (50)	first_name character varying (50)	middle_name character varying (50)	department_id integer	salary numeric	internal_number integer
1	1	Иванов	Иван	Иванович	1	60000	1001
2	2	Петров	Пётр	Петрович	2	55000	1002
3	21	Богатый	Иван	Сергеевич	1	100000	999
4	23	Петров	Алексей	Сергеевич	1	50000	1003
5	24	Иванов	Иван	Петрович	2	60000	1004
6	27	Сидоров	Алексей	Петрович	1	45000	1005
7	28	Кузнецова	Ирина	Михайловна	2	48000	1006
8	30	Сидоров	Алексей	Петрович	1	45000	1007
9	31	Кузнецова	Ирина	Михайловна	2	48000	1008

Query	Query History
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11

**INSERT INTO lab3\_schema.employee:** Мы добавляем нового сотрудника в таблицу **employee**.

**SELECT 'Новиков', 'Алексей', 'Юрьевич', 2, 55000, 1009:** Указываем данные нового сотрудника (ФИО, отдел, зарплата, внутренний номер).

**WHERE EXISTS (...):** Этот подзапрос проверяет, есть ли **невыполненные** задания в проекте с **project\_id = 4**.

**SELECT 1:** Мы проверяем существование хотя бы одного невыполненного задания.

**WHERE project\_id = 4 AND status != 'Завершено':** Подзапрос фильтрует задания, которые относятся к проекту **Проект D** (с **project\_id = 4**) и имеют статус **не завершено**

ПОСЛЕ

	employee_id [PK] integer	last_name character varying (50)	first_name character varying (50)	middle_name character varying (50)	department_id integer	salary numeric	internal_number integer
1	1	Иванов	Иван	Иванович	1	60000	1001
2	2	Петров	Пётр	Петрович	2	55000	1002
3	21	Богатый	Иван	Сергеевич	1	100000	999
4	23	Петров	Алексей	Сергеевич	1	50000	1003
5	24	Иванов	Иван	Петрович	2	60000	1004
6	27	Сидоров	Алексей	Петрович	1	45000	1005
7	28	Кузнецова	Ирина	Михайловна	2	48000	1006
8	30	Сидоров	Алексей	Петрович	1	45000	1007
9	31	Кузнецова	Ирина	Михайловна	2	48000	1008

## Запрос на обновление данных (UPDATE)

Мы будем обновлять зарплату сотрудников, чья зарплата ниже **средней** по их отделу. Для этого используем подзапрос.

ДО

	employee_id [PK] integer	last_name character varying (50)	first_name character varying (50)	middle_name character varying (50)	department_id integer	salary numeric	internal_number integer
1	1	Иванов	Иван	Иванович	1	60000	1001
2	2	Петров	Пётр	Петрович	2	55000	1002
3	21	Богатый	Иван	Сергеевич	1	100000	999
4	23	Петров	Алексей	Сергеевич	1	50000	1003
5	24	Иванов	Иван	Петрович	2	60000	1004
6	27	Сидоров	Алексей	Петрович	1	45000	1005
7	28	Кузнецова	Ирина	Михайловна	2	48000	1006
8	30	Сидоров	Алексей	Петрович	1	45000	1007
9	31	Кузнецова	Ирина	Михайловна	2	48000	1008



## Query Query History

```

1  ✓ UPDATE lab3_schema.employee
2     SET salary = salary * 1.1
3     WHERE department_id IN (
4         SELECT department_id
5         FROM lab3_schema.employee
6         GROUP BY department_id
7         HAVING AVG(salary) < 50000
8     );
9

```

**UPDATE lab3\_schema.employee:** Мы обновляем данные в таблице **employee**.

**SET salary = salary \* 1.1:** Увеличиваем зарплату сотрудников на **10%**.

**WHERE department\_id IN (...):** Мы обновляем только сотрудников тех отделов, где средняя зарплата **меньше 50000**.

**Подзапрос:**

**SELECT department\_id:** Извлекаем ID всех отделов.

**HAVING AVG(salary) < 50000:** Фильтруем только те отделы, где **средняя зарплата меньше 50000**.

**ПОСЛЕ**

	employee_id [PK] integer	last_name character varying (50)	first_name character varying (50)	middle_name character varying (50)	department_id integer	salary numeric	internal_number integer
1	1	Иванов	Иван	Иванович	1	60000	1001
2	2	Петров	Пётр	Петрович	2	55000	1002
3	21	Богатый	Иван	Сергеевич	1	100000	999
4	23	Петров	Алексей	Сергеевич	1	50000	1003
5	24	Иванов	Иван	Петрович	2	60000	1004
6	27	Сидоров	Алексей	Петрович	1	45000	1005
7	28	Кузнецова	Ирина	Михайловна	2	48000	1006
8	30	Сидоров	Алексей	Петрович	1	45000	1007
9	31	Кузнецова	Ирина	Михайловна	2	48000	1008

## Запрос на удаление данных (DELETE)

Задание: Удалим всех сотрудников, которые **не участвуют в заданиях**. Мы будем использовать подзапрос, чтобы удалить только тех сотрудников, которые не имеют записей в таблице **task\_participation** (то есть, не участвуют в заданиях).

до

	employee_id [PK] integer	last_name character varying (50)	first_name character varying (50)	middle_name character varying (50)	department_id integer	salary numeric	internal_number integer
1	1	Иванов	Иван	Иванович	1	60000	1001
2	2	Петров	Пётр	Петрович	2	55000	1002
3	21	Богатый	Иван	Сергеевич	1	100000	999
4	23	Петров	Алексей	Сергеевич	1	50000	1003
5	24	Иванов	Иван	Петрович	2	60000	1004
6	27	Сидоров	Алексей	Петрович	1	45000	1005
7	28	Кузнецова	Ирина	Михайловна	2	48000	1006
8	30	Сидоров	Алексей	Петрович	1	45000	1007
9	31	Кузнецова	Ирина	Михайловна	2	48000	1008

### Query Query History

```
1  ✓ DELETE FROM lab3_schema.employee
2  WHERE employee_id NOT IN (
3      SELECT DISTINCT employee_id
4      FROM lab3_schema.task_participation
5  );
6
```

**DELETE FROM lab3\_schema.employee:** Удаляем сотрудников из таблицы **employee**.

**WHERE employee\_id NOT IN (...):** Удаляем только тех сотрудников, чьи **employee\_id** отсутствуют в таблице **task\_participation**.

**Подзапрос:**

**SELECT DISTINCT employee\_id FROM lab3\_schema.task\_participation:** Получаем уникальные **employee\_id** из таблицы **task\_participation**, чтобы увидеть, какие сотрудники участвуют в заданиях.

ПОСЛЕ

Showing rows: 1 to 2								Page No:	
	employee_id [PK] integer	last_name character varying (50)	first_name character varying (50)	middle_name character varying (50)	department_id integer	salary numeric	internal_number integer		
1	1	Иванов	Иван	Иванович	1	60000	1001		
2	2	Петров	Пётр	Петрович	2	55000	1002		

### 3.4 Создание индексов

В данном пункте лабораторной работы были проведены следующие действия:

#### 1. Выполнение тестовых запросов без индексов

##### Добавление данных (INSERT)

Для начала выполнила запрос на добавление данных в таблицу **employee**

Query

Query History

1

2

3

4

5

EXPLAIN ANALYZE

INSERT INTO lab3\_schema.employee (last\_name, first\_name, middle\_name, department\_id, salary, int

VALUES

('Иванов', 'Алексей', 'Владимирович', 1, 50000, 1024);

Data Output

Messages

Notifications

Showing rows: 1 to 4

QUERY PLAN

text

1

2

3

4

Insert on employee (cost=0.00..0.01 rows=0 width=0) (actual time=0.180..0.181 rows=0 loops=...

-> Result (cost=0.00..0.01 rows=1 width=398) (actual time=0.049..0.049 rows=1 loops=1)

Planning Time: 0.105 ms

Execution Time: 0.220 ms

Этот запрос выполнил добавление нового сотрудника и замерил время выполнения с помощью **EXPLAIN ANALYZE**. Он показал план выполнения запроса и фактическое время выполнения.

## Обновление данных (UPDATE)

The screenshot shows a database query editor with two tabs: "Query" and "Query History". The "Query" tab is active, displaying the following SQL code:

```
1  EXPLAIN ANALYZE
2  UPDATE lab3_schema.employee
3  SET salary = salary * 1.1
4  WHERE department_id = 1;
5
```

Below the query editor, there are three tabs: "Data Output", "Messages", and "Notifications". The "Data Output" tab is active, showing a table with 6 rows. The first row is the title "QUERY PLAN" with a lock icon. The subsequent rows show the execution plan details:

	QUERY PLAN
1	Update on employee (cost=0.00..12.38 rows=0 width=0) (actual time=0.310..0.311 rows=0 loops=1)
2	-> Seq Scan on employee (cost=0.00..12.38 rows=1 width=38) (actual time=0.075..0.081 rows=9 loop...)
3	Filter: (department_id = 1)
4	Rows Removed by Filter: 7
5	Planning Time: 0.210 ms
6	Execution Time: 0.356 ms

Этот запрос увеличил **оклады сотрудников** в отделе с **department\_id = 1** на **10%** и замерил время выполнения.

## 2. Создание индексов

Теперь мы создадим индексы для таблиц, которые часто используются в запросах, чтобы ускорить выполнение запросов.

**Индекс на `employee_id` в таблице `task_participation`:**



The screenshot displays a database interface with a query editor and a messages pane. The query editor shows a SQL statement to create an index on the `employee_id` column of the `task_participation` table. The messages pane below shows the successful execution of the query.

```
Query Query History
```

```
1 CREATE INDEX idx_employee_id ON lab3_schema.task_participation(employee_id);
2
```

---

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 42 msec.

Этот индекс поможет ускорить поиск и операции с участниками заданий, особенно если часто фильтруем по **`employee_id`**.

## Индекс на `project_id` в таблице `task`:

The screenshot shows a database query interface with two tabs: "Query" and "Query History". The "Query" tab is active, displaying a SQL command: `CREATE INDEX idx_project_id ON lab3_schema.task(project_id);`. The command is numbered 1 and 2. Below the query, there are three tabs: "Data Output", "Messages", and "Notifications". The "Messages" tab is active, showing the message "CREATE INDEX" and "Query returned successfully in 48 msec."

Этот индекс ускорит операции с заданиями, связанными с проектами, если часто фильтруем по **`project_id`**.

### 3. Выполнение тех же запросов с индексами

Запросы были повторно выполнены, планы запросов с помощью `EXPLAIN ANALYZE` показали изменение стратегии доступа — в том числе использование индексного поиска (`Index Scan` или `Bitmap Index Scan`).

## Запрос на добавление данных (INSERT):

Query

Query History

1

EXPLAIN ANALYZE

2

INSERT INTO lab3\_schema.employee (last\_name, first\_name, middle\_name, department\_id, salary, integer\_1)

3

VALUES ('Иванов', 'Алексей', 'Владимирович', 1, 50000, 1025);

4

Data Output

Messages

Notifications

Showing rows: 1 to 4

QUERY PLAN

text

1

Insert on employee (cost=0.00..0.01 rows=0 width=0) (actual time=0.163..0.164 rows=0 loops=...

2

-> Result (cost=0.00..0.01 rows=1 width=398) (actual time=0.067..0.067 rows=1 loops=1)

3

Planning Time: 0.094 ms

4

Execution Time: 0.203 ms

## Запрос на обновление данных (UPDATE):

Query

Query History

1

EXPLAIN ANALYZE

2

UPDATE lab3\_schema.employee

3

SET salary = salary \* 1.1

4

WHERE department\_id = 1;

5

Data Output

Messages

Notifications

Showing rows: 1 to 6

QUERY PLAN

text

1

Update on employee (cost=0.00..12.38 rows=0 width=0) (actual time=0.265..0.266 rows=0 loops=1)

2

-> Seq Scan on employee (cost=0.00..12.38 rows=1 width=38) (actual time=0.061..0.071 rows=10 loop=1)

3

Filter: (department\_id = 1)

4

Rows Removed by Filter: 7

5

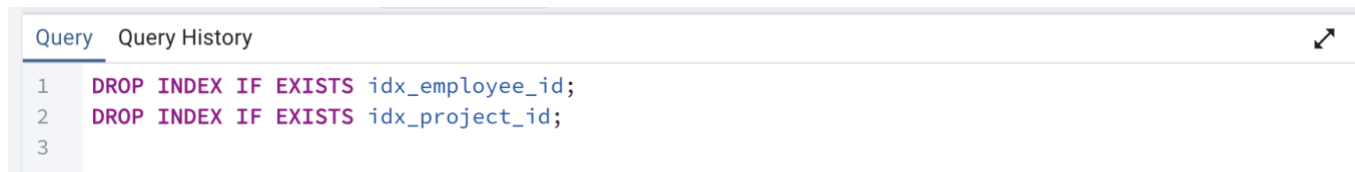
Planning Time: 0.211 ms

6

Execution Time: 0.323 ms

Индекс помог оптимизировать доступ к данным, уменьшив время выборки.

### Удаление индексов



```
Query  Query History  ↗
1  DROP INDEX IF EXISTS idx_employee_id;
2  DROP INDEX IF EXISTS idx_project_id;
3
```

### 4. Вывод по лабораторной работе:

В ходе выполнения лабораторной работы были приобретены практические навыки разработки и выполнения сложных SQL-запросов на выборку и модификацию данных в реляционной базе PostgreSQL.