Практика TCL, DML, DDL, and DCL Statements

Задание 1: DML

Цель: Практика базовой вставки, выборки, простого обновления и удаления данных.

Таблица Employees до:

```
select * from Employees
```

Введите SQL выражение чтобы отфильтровать результаты

	employeeid	firstname	lastname	department	salary
1	1	Alice	Smith	HR	60 000
2	2	Bob	Johnson	IT	75 000
3	3	Charlie	Brown	Finance	62 000
4	4	Diana	Prince	IT	80 000
5	5	Eve	Davis	HR	58 000

Выполнение заданий

1. Вставить двух новых сотрудников в таблицу Employees.

```
-- 1. Вставка двух новых сотрудников в таблицу Employees
insert into  Employees (FirstName, LastName, Department, Salary)
values ('John', 'Doe', 'Marketing', 55000.00),
       ('Jane', 'Taylor', 'IT', 72000.00);
```

2. Выбрать всех сотрудников из таблицы Employees.

```
-- 2. Выбор всех сотрудников из таблицы Employees
select  * from Employees;
```

	employeeid	firstname	lastname	department	salary
1	1	Alice	Smith	HR	60 000
2	2	Bob	Johnson	IT	75 000
3	3	Charlie	Brown	Finance	62 000
4	4	Diana	Prince	IT	80 000
5	5	Eve	Davis	HR	58 000
6	6	John	Doe	Marketing	55 000
7	7	Jane	Taylor	IT	72 000

3. Выбрать только FirstName и LastName сотрудников из отдела 'IT'.

```
-- 3. Выбор только FirstName и LastName сотрудников из отдела IT
select FirstName, LastName from  Employees where Department = 'IT';
```

	firstname	lastname
1	Bob	Johnson
2	Diana	Prince
3	Jane	Taylor

4. Обновить Salary 'Alice Smith' до 65000.00.

```
-- 4. Обновление Salary (з/п) 'Alice Smith' до 65000.00
update Employees set Salary = 65000.00 where FirstName = 'Alice' and LastName = 'Smith';
```

1	Alice	Smith	HR	65 000

5. Удалить сотрудника, чья LastName — 'Prince'

```
-- 5. Удаление сотрудника с фамилией Prince
delete from Employees where LastName = 'Prince';
```

6. Проверить все изменения, используя SELECT * FROM Employees;

```
-- 6. Проверка всех изменений
select * from Employees;
```

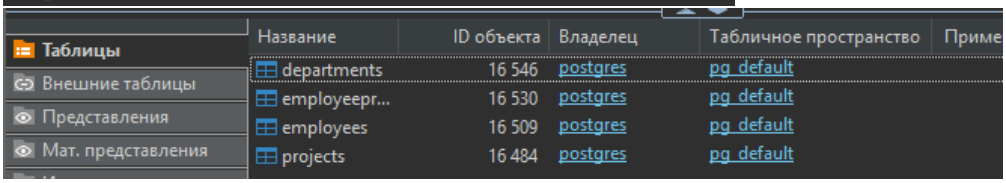
	employeeid	firstname	lastname	department	salary
1	1	Alice	Smith	HR	65 000
2	2	Bob	Johnson	IT	75 000
3	3	Charlie	Brown	Finance	62 000
4	5	Eve	Davis	HR	58 000
5	6	John	Doe	Marketing	55 000
6	7	Jane	Taylor	IT	72 000

Задание 2: DDL
Цель: Практика создания и изменения структуры таблиц.

Выполнение заданий:

1. Создать новую таблицу с именем Departments со столбцами: DepartmentID (SERIAL PRIMARY KEY), DepartmentName (VARCHAR(50), UNIQUE, NOT NULL), Location (VARCHAR(50)).

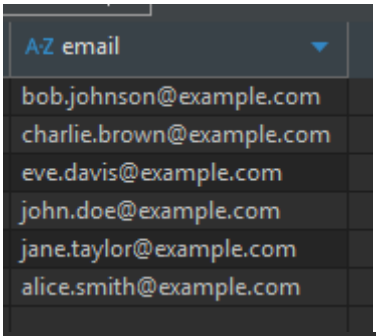
```
-- 1. Создание таблицы Departments
create table Departments (
    DepartmentID SERIAL primary key,
    DepartmentName VARCHAR(50) unique not null,
    Location VARCHAR(50)
);
```



2. Изменить таблицу Employees, добавив новый столбец с именем Email (VARCHAR(100)).

```
-- 2. Добавление столбца Email в  таблицу Employees
alter table Employees add column Email VARCHAR(100);
```

3. Добавить ограничение UNIQUE к столбцу Email в таблице Employees, предварительно заполнив любыми значениями

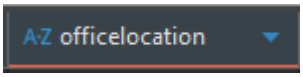



```
-- 3. Добавление ограничения UNIQUE к столбцу Email в таблице Employees и предварительное заполнение
update Employees
set Email = CONCAT(LOWER(FirstName), '.', LOWER(LastName), '@example.com')
where Email is null;

alter table Employees
add constraint unique_email unique (Email);
```

4. Переименовать столбец Location в таблице Departments в OfficeLocation.

```
-- 4. Переименование столбца Location
alter table Departments rename column Location to OfficeLocation;
```



Задание 3: DCL

Цель: Научиться создавать пользователей (роли в PostgreSQL) и предоставлять/отзывать базовые разрешения

Выполнение заданий:

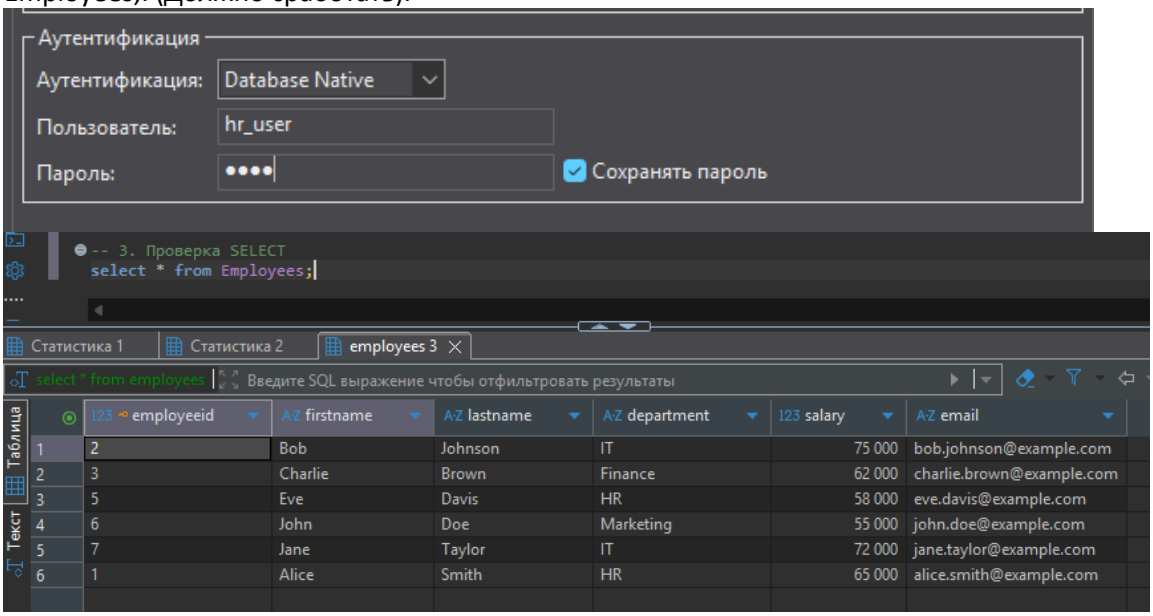
1. Создать нового пользователя PostgreSQL (роль) с именем hr_user и простым паролем.

```
-- 1. Создание нового пользователя PostgreSQL (роль) с именем hr_user и простым паролем
create role hr_user with LOGIN password '1234';
```

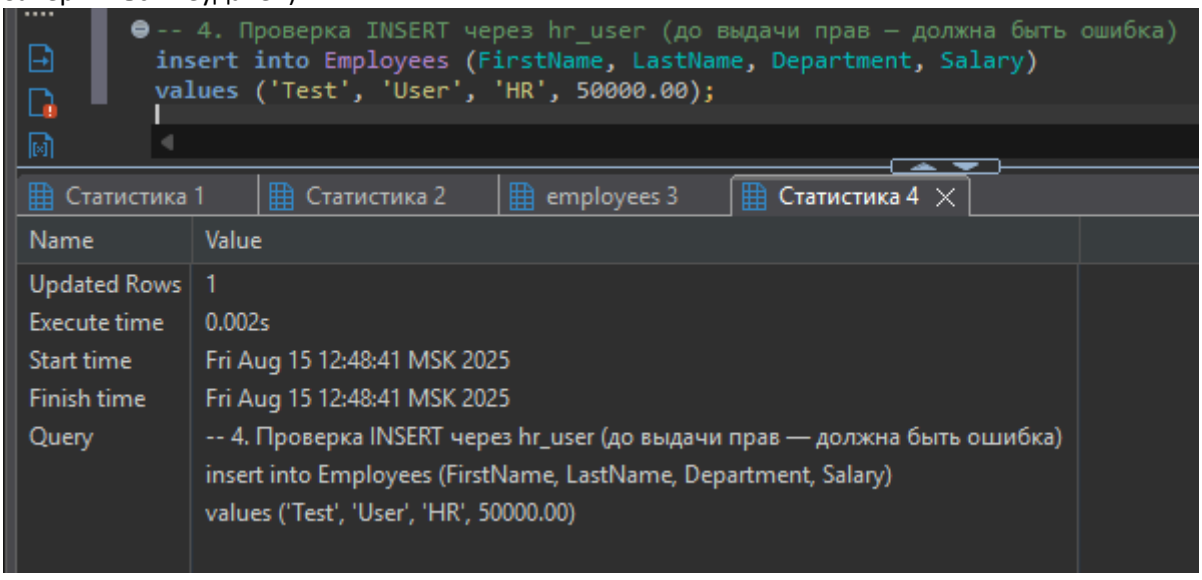
2. Предоставить hr_user право SELECT на таблицу Employees.

```
-- 2. Предоставление hr_user право SELECT на таблицу Employees
grant select on Employees to hr_user;
```

3. Тест: В новой сессии подключиться как hr_user и попытаться выполнить SELECT * FROM Employees;. (Должно сработать).



4. Как hr_user, попытаться выполнить INSERT нового сотрудника в Employees. (Должно завершиться неудачей).



5. Как пользователь-администратор, предоставить hr_user права INSERT и UPDATE на таблицу Employees.

```
-- 5. Предоставление прав INSERT и UPDATE как пользователь-админ
grant insert, update on employees to hr_user;
```

6. Тест: Как hr_user, попробовать выполнить INSERT и UPDATE сотрудника. (Теперь должно сработать).

```
-- 6. Финальный тест: INSERT и UPDATE от имени hr_user. Ожидаемый результат: оба запроса выполняются успешно

-- INSERT нового сотрудника
insert into Employees (FirstName, LastName, Department, Salary)
values ('Frank', 'Miller', 'Marketing', 55000.00);

-- UPDATE существующего сотрудника
update Employees
set Salary = 77000.00
where FirstName = 'Bob' and LastName = 'Johnson';

-- Проверка результатов
select * from Employees
where FirstName = 'Frank' or (FirstName = 'Bob' and LastName = 'Johnson');
```

	123 employeeid	AZ firstname	AZ lastname	AZ department	123 salary
1	10	Frank	Miller	Marketing	55 000
2	2	Bob	Johnson	IT	77 000

Задание 4: DML/DCL

Цель: Практика более сложных DML-операций с использованием предложений WHERE, множественных обновлений

Выполнение заданий:

1. Увеличить Salary всех сотрудников в отделе 'HR' на 10%.

```
-- 1. Увеличение зарплаты в HR на 10%
update Employees set Salary = Salary * 1.10 where Department = 'HR';
```

Вот пример, что было у Alise

1	Alice	Smith	HR	65 000

Стало:

1	Alice	Smith	HR	71 500

2. Обновить Department любого сотрудника с Salary выше 70000.00 на 'Senior IT'

```
-- 2. Обновление отдела при зарплате > 70000
update Employees set Department = 'Senior IT' where Salary > 70000.00;
```

7	Jane	Taylor	Senior IT	72 000	jane.taylor@example.com
2	Bob	Johnson	Senior IT	77 000	bob.johnson@example.com
1	Alice	Smith	Senior IT	71 500	alice.smith@example.com

3. Удалить всех сотрудников, которые не назначены ни на один проект в таблице EmployeeProjects. Подсказка: Используйте подзапрос NOT EXISTS или LEFT JOIN

```
-- 3. Удаление сотрудников без проектов
delete from Employees
where not exists (
    select 1 from EmployeeProjects ep where ep.EmployeeID = Employees.EmployeeID
);
```

4. Вставить новый проект и назначить на него двух существующих сотрудников с определенным количеством HoursWorked в EmployeeProjects, и все это в одном блоке BEGIN/COMMIT.

```
-- 4. Вставка проекта и назначение сотрудников
begin;
insert into Projects (ProjectName, Budget, StartDate, EndDate)
values ('New Initiative', 100000.00, '2025-08-01', '2025-12-31');

insert into EmployeeProjects (EmployeeID, ProjectID, HoursWorked)
values (1, 4, 90), (3, 4, 100);
commit;
```

Задание 5: Функции и представления
Цель: Понять и создать простые SQL-функции и представления в PostgreSQL.

Выполнение заданий:

1. Функция: Создать функцию PostgreSQL с именем CalculateAnnualBonus, которая принимает employee_id и Salary в качестве входных данных и возвращает рассчитанную сумму бонуса (10 % от Salary) для этого сотрудника. Используйте PL/pgSQL для тела функции.

```sql
-- 1. Функция CalculateAnnualBonus
-- Назначение: рассчитывает годовой бонус сотрудника как 10% от его зарплаты.
-- Принимает: emp_id (не используется в теле функции, но может быть полезен для расширения логики)
-- Возвращает: бонус в формате DECIMAL
create or replace function CalculateAnnualBonus(emp_id INT, salary DECIMAL)
returns DECIMAL as $$
begin
    return salary * 0.10;
end;
$$ language plpgsql;
```

2. Использовать эту функцию в операторе SELECT, чтобы увидеть потенциальный бонус для каждого сотрудника.

```sql
-- 2. Использование функции CalculateAnnualBonus
-- Цель: показать потенциальный бонус для каждого сотрудника
-- Выводит: ID, имя, фамилию, зарплату и рассчитанный бонус
select EmployeeID, FirstName, LastName, Salary,
       CalculateAnnualBonus(EmployeeID, Salary) as Bonus
from Employees;
```

3. Представление (View): Создать представление с именем IT_Department_View, которое показывает EmployeeID, FirstName, LastName и Salary только для сотрудников из отдела 'IT'.

```sql
-- 3. Представление IT_Department_View которое показывает EmployeeID, FirstName, LastName и Salary только для сотрудников из отдела 'IT'
create view IT_Department_View as
select EmployeeID, FirstName, LastName, Salary
from Employees where Department = 'IT';
```

4. Выбрать данные из вашего представления IT_Department_View.

```sql
-- 4. Выбор из представления
SELECT * FROM IT_Department_View;
```

Задание 6: DML (Optional)
Цель: Объединение DML-операций с JOIN, подзапросами и условной логикой

Выполнение заданий:

1. Найти ProjectName всех проектов, в которых 'Bob Johnson' работал более 150 часов.

```sql
-- 1. Выбор названий проектов, где Bob Johnson работал более 150 часов
select p.ProjectName
from Projects p
join EmployeeProjects ep on p.ProjectID = ep.ProjectID
join Employees e on ep.EmployeeID = e.EmployeeID
where e.FirstName = 'Bob' and e.LastName = 'Johnson' and ep.HoursWorked > 150;
```

2. Увеличить Budget всех проектов на 10%, если к ним назначен хотя бы один сотрудник из отдела 'IT'.

```sql
-- 2. Увеличение бюджета на 10% для проектов, в которых участвуют сотрудники из отдела 'IT'
update Projects
set Budget = Budget * 1.10
where  ProjectID in (
    select distinct ep.ProjectID
    from EmployeeProjects ep
    join Employees e on ep.EmployeeID = e.EmployeeID
    where e.Department = 'IT'
);
```

3. Для любого проекта, у которого еще нет EndDate (EndDate IS NULL), установить EndDate на один год позже его StartDate.

```sql
-- 3. Установка даты завершения проекта на один год позже даты начала, если EndDate отсутствует
update Projects
set EndDate = StartDate + INTERVAL '1 year'
where EndDate is null;
```

4. Вставить нового сотрудника и немедленно назначить его на проект 'Website Redesign' с 80 отработанными часами, все в рамках одной транзакции. Использовать предложение RETURNING, чтобы получить EmployeeID вновь вставленного сотрудника.

```sql
-- 4. Вставка нового сотрудника и назначение его на проект 'Website Redesign' с 80 часами
-- Используется транзакция и RETURNING для получения нового EmployeeID
begin;
insert into Employees (FirstName, LastName, Department, Salary)
values ('Greg', 'Miller', 'Design', 50000.00)
returning EmployeeID into new_emp_id;

insert into EmployeeProjects (EmployeeID, ProjectID, HoursWorked)
values (new_emp_id, 1, 80);
commit;
```