

**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Тульский государственный университет»**

**Интернет-институт**

**ОТЧЕТ ПО ЛАБОРАТОРНЫМ РАБОТАМ**

по дисциплине

«Базы данных 1»

Семестр 5

Вариант 3

Выполнил: студент гр. ИБ262521-ф

Артемов Александр Евгеньевич

Проверил: канд. техн. наук, доц.

Французова Юлия Вячеславовна

Тула 2024

# Лабораторная работа № 1.

**Название работы:** Установка и настройка СУБД.

**Цели работы:** Получение навыков установки и настройки СУБД.

**Задание:**

1. Установите СУБД PostgreSQL 9.6 (или новее) и клиент управления СУБД pgAdmin4. Приведите скриншоты шаговой установки.
2. Подключитесь к базе данных через текстовый терминал (с помощью программы psql) от имени пользователя PostgreSQL и создайте базу данных. Приведите в отчете скриншоты терминала с выполненными командами и их результатом.

**Выполнение лабораторной работы.**

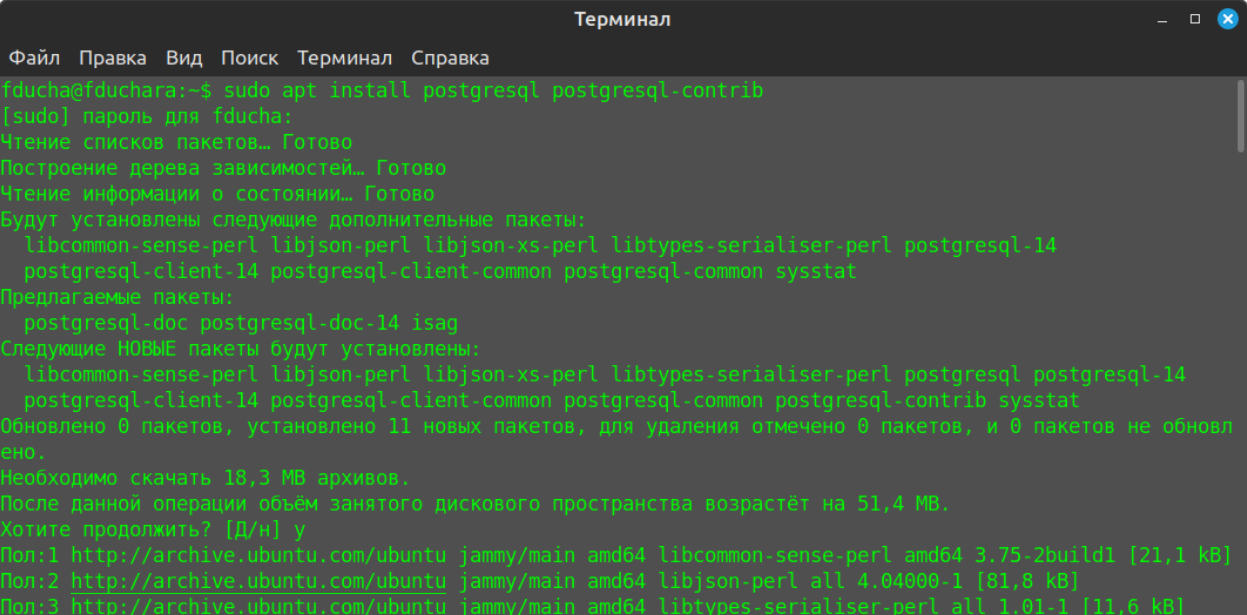
1. Изучены теоретические сведения лабораторной работы по запуску и остановке базы данных PostgreSQL.
2. Установка СУБД PostgreSQL и клиента управления СУБД pgAdmin4.

Установка PostgreSQL и pgAdmin4 производится в среде операционной системы Linux Mint 21.3 Cinnamon по статье автора Winnie Ondara «How to Install PostgreSQL with pgAdmin4 on Linux Mint 21/20» размещенной по адресу <https://www.tecmint.com/install-postgresql-with-pgadmin4-on-linux-mint/>.

Установка PostgreSQL.

В эмуляторе терминала ОС выполняем команду установки:

```
sudo apt install postgresql postgresql-contrib
```



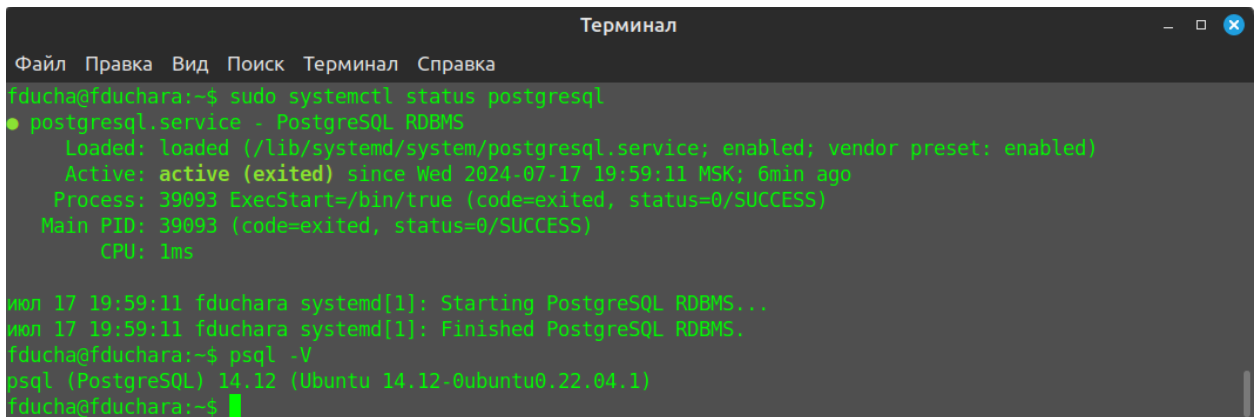
```
Терминал
Файл Правка Вид Поиск Терминал Справка
fducha@fduchara:~$ sudo apt install postgresql postgresql-contrib
[sudo] пароль для fducha:
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Будут установлены следующие дополнительные пакеты:
  libcommon-sense-perl libjson-perl libjson-xs-perl libtypes-serialiser-perl postgresql-14
  postgresql-client-14 postgresql-client-common postgresql-common sysstat
Предлагаемые пакеты:
  postgresql-doc postgresql-doc-14 isag
Следующие НОВЫЕ пакеты будут установлены:
  libcommon-sense-perl libjson-perl libjson-xs-perl libtypes-serialiser-perl postgresql postgresql-14
  postgresql-client-14 postgresql-client-common postgresql-common postgresql-contrib sysstat
Обновлено 0 пакетов, установлено 11 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновлено.
Необходимо скачать 18,3 МБ архивов.
После данной операции объем занятого дискового пространства возрастет на 51,4 МБ.
Хотите продолжить? [Д/н] y
Пон:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 libcommon-sense-perl amd64 3.75-2build1 [21,1 kB]
Пон:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjson-perl all 4.04000-1 [81,8 kB]
Пон:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 libtypes-serialiser-perl all 1.01-1 [11,6 kB]
```

Проверяем работоспособность установленной СУБД командой

```
sudo systemctl status postgresql
```

и версию СУБД командой

```
psql -V
```



```
Терминал
Файл Правка Вид Поиск Терминал Справка
fduchara@fduchara:~$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; enabled; vendor preset: enabled)
   Active: active (exited) since Wed 2024-07-17 19:59:11 MSK; 6min ago
     Process: 39093 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
    Main PID: 39093 (code=exited, status=0/SUCCESS)
       CPU: 1ms

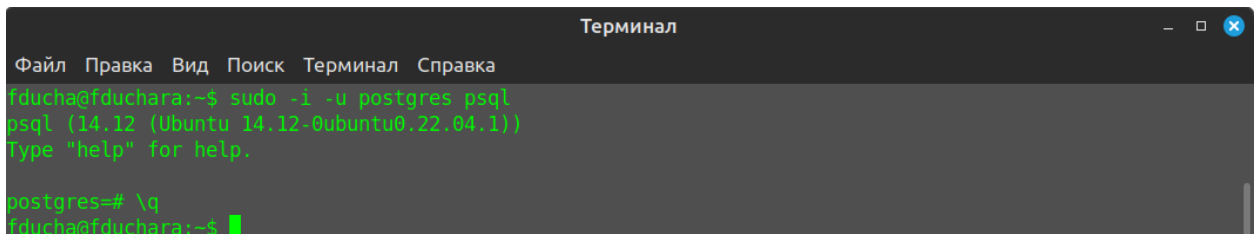
июл 17 19:59:11 fduchara systemd[1]: Starting PostgreSQL RDBMS...
июл 17 19:59:11 fduchara systemd[1]: Finished PostgreSQL RDBMS.
fduchara@fduchara:~$ psql -V
psql (PostgreSQL) 14.12 (Ubuntu 14.12-0ubuntu0.22.04.1)
fduchara@fduchara:~$
```

Как видно, установлена СУБД PostgreSQL версии 14.12.

Для соединения с сервером СУБД и входа в консоль используем команду

```
sudo -u postgres psql
```

где postgres — имя пользователя по умолчанию. Используем команду `\q` для выхода из консоли.



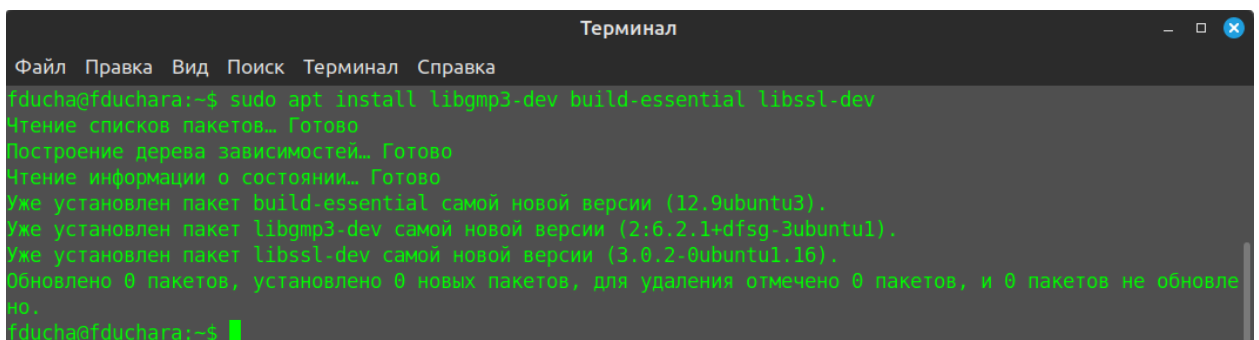
```
Терминал
Файл Правка Вид Поиск Терминал Справка
fduchara@fduchara:~$ sudo -i -u postgres psql
psql (14.12 (Ubuntu 14.12-0ubuntu0.22.04.1))
Type "help" for help.

postgres=# \q
fduchara@fduchara:~$
```

Установка pgAdmin4.

Устанавливаем необходимые зависимости пакетов

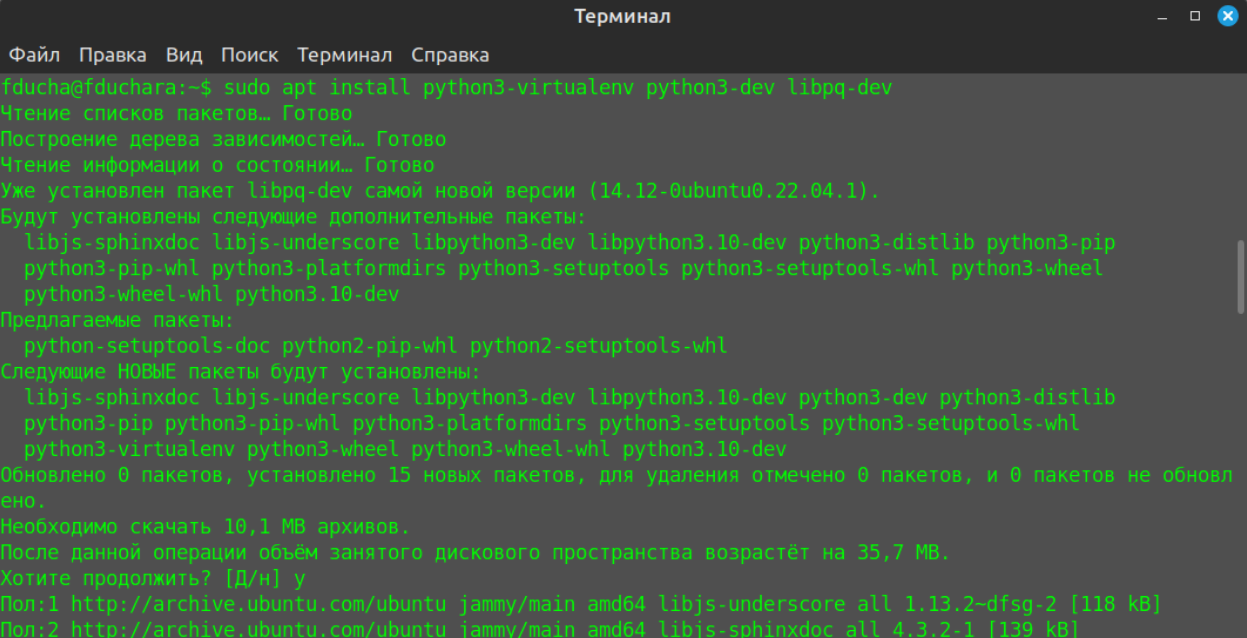
```
sudo apt install libgmp3-dev build-essential libssl-dev
```



```
Терминал
Файл Правка Вид Поиск Терминал Справка
fduchara@fduchara:~$ sudo apt install libgmp3-dev build-essential libssl-dev
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Уже установлен пакет build-essential самой новой версии (12.9ubuntu3).
Уже установлен пакет libgmp3-dev самой новой версии (2:6.2.1+dfsg-3ubuntu1).
Уже установлен пакет libssl-dev самой новой версии (3.0.2-0ubuntu1.16).
Обновлено 0 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновле
но.
fduchara@fduchara:~$
```

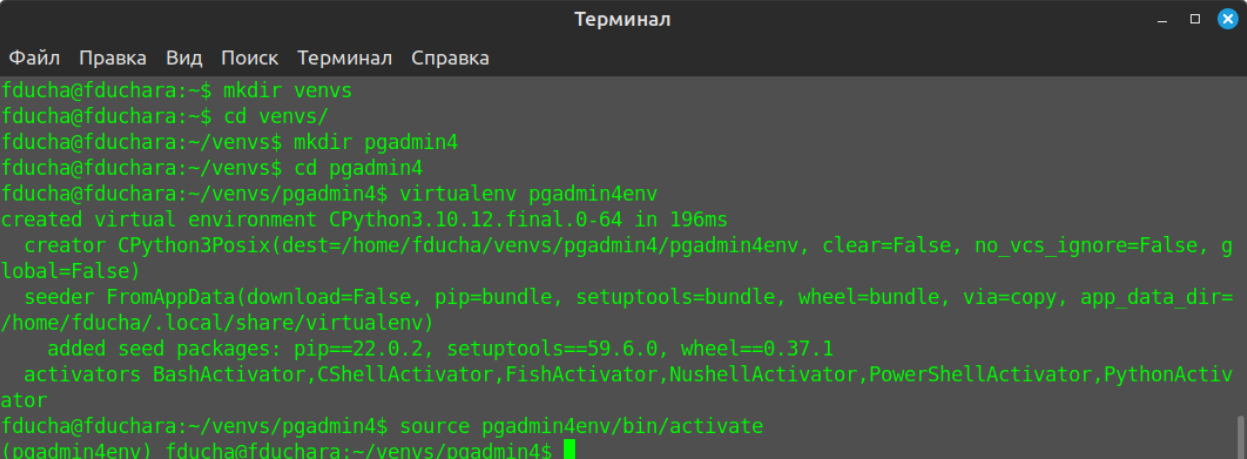
Устанавливаем виртуальное окружение для Python3 и необходимые пакеты разработчика:

```
sudo apt install python3-virtualenv python3-dev libpq-dev
```



```
Терминал
Файл Правка Вид Поиск Терминал Справка
fducha@fduchara:~$ sudo apt install python3-virtualenv python3-dev libpq-dev
Чтение списков пакетов... Готово
Построение дерева зависимостей... Готово
Чтение информации о состоянии... Готово
Уже установлен пакет libpq-dev самой новой версии (14.12-0ubuntu0.22.04.1).
Будут установлены следующие дополнительные пакеты:
  libjs-sphinxdoc libjs-underscore libpython3-dev libpython3.10-dev python3-distlib python3-pip
  python3-pip-whl python3-platformdirs python3-setuptools python3-setuptools-whl python3-wheel
  python3-wheel-whl python3.10-dev
Предлагаемые пакеты:
  python-setuptools-doc python2-pip-whl python2-setuptools-whl
Следующие НОВЫЕ пакеты будут установлены:
  libjs-sphinxdoc libjs-underscore libpython3-dev libpython3.10-dev python3-dev python3-distlib
  python3-pip python3-pip-whl python3-platformdirs python3-setuptools python3-setuptools-whl
  python3-virtualenv python3-wheel python3-wheel-whl python3.10-dev
Обновлено 0 пакетов, установлено 15 новых пакетов, для удаления отмечено 0 пакетов, и 0 пакетов не обновл
ено.
Необходимо скачать 10,1 MB архивов.
После данной операции объем занятого дискового пространства возрастет на 35,7 MB.
Хотите продолжить? [Д/н] y
Пон:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjs-underscore all 1.13.2-dfsg-2 [118 kB]
Пон:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjs-sphinxdoc all 4.3.2-1 [139 kB]
```

Создаем каталог `~/venvs/pgadmin4` в домашней директории и переходим в него. Создаем в этом каталоге виртуальное окружение `pgadmin4env` и активируем его:



```
Терминал
Файл Правка Вид Поиск Терминал Справка
fducha@fduchara:~$ mkdir venvs
fducha@fduchara:~$ cd venvs/
fducha@fduchara:~/venvs$ mkdir pgadmin4
fducha@fduchara:~/venvs$ cd pgadmin4
fducha@fduchara:~/venvs/pgadmin4$ virtualenv pgadmin4env
created virtual environment CPython3.10.12.final.0-64 in 196ms
  creator CPython3Posix(dest=/home/fducha/venvs/pgadmin4/pgadmin4env, clear=False, no_vcs_ignore=False, g
  lobal=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=
  /home/fducha/.local/share/virtualenv)
  added seed packages: pip==22.0.2, setuptools==59.6.0, wheel==0.37.1
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActiv
  ator
fducha@fduchara:~/venvs/pgadmin4$ source pgadmin4env/bin/activate
(pgadmin4env) fducha@fduchara:~/venvs/pgadmin4$
```

Устанавливаем в виртуальном окружении клиент `pgAdmin4` командой

```
pip install pgadmin4
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
(pgadmin4env) fducha@fduchara:~/venvs/pgadmin4$ pip install pgadmin4
Collecting pgadmin4
  Downloading pgadmin4-8.9-py3-none-any.whl (101.2 MB)
    101.2/101.2 MB 10.6 MB/s eta 0:00:00
Collecting azure-mgmt-resource==23.1.1
  Downloading azure_mgmt_resource-23.1.1-py3-none-any.whl (2.6 MB)
    2.6/2.6 MB 19.9 MB/s eta 0:00:00
Collecting jsonformatter==0.3.2
  Downloading jsonformatter-0.3.2.tar.gz (15 kB)
```

Помимо клиента pgAdmin4 пакетный менеджер установит требуемые зависимости.

Создаем конфигурационный файл `config_local.py` в каталоге `pgadmin4env/lib/python3.10/site-packages/pgadmin4` виртуального окружения следующего содержания:

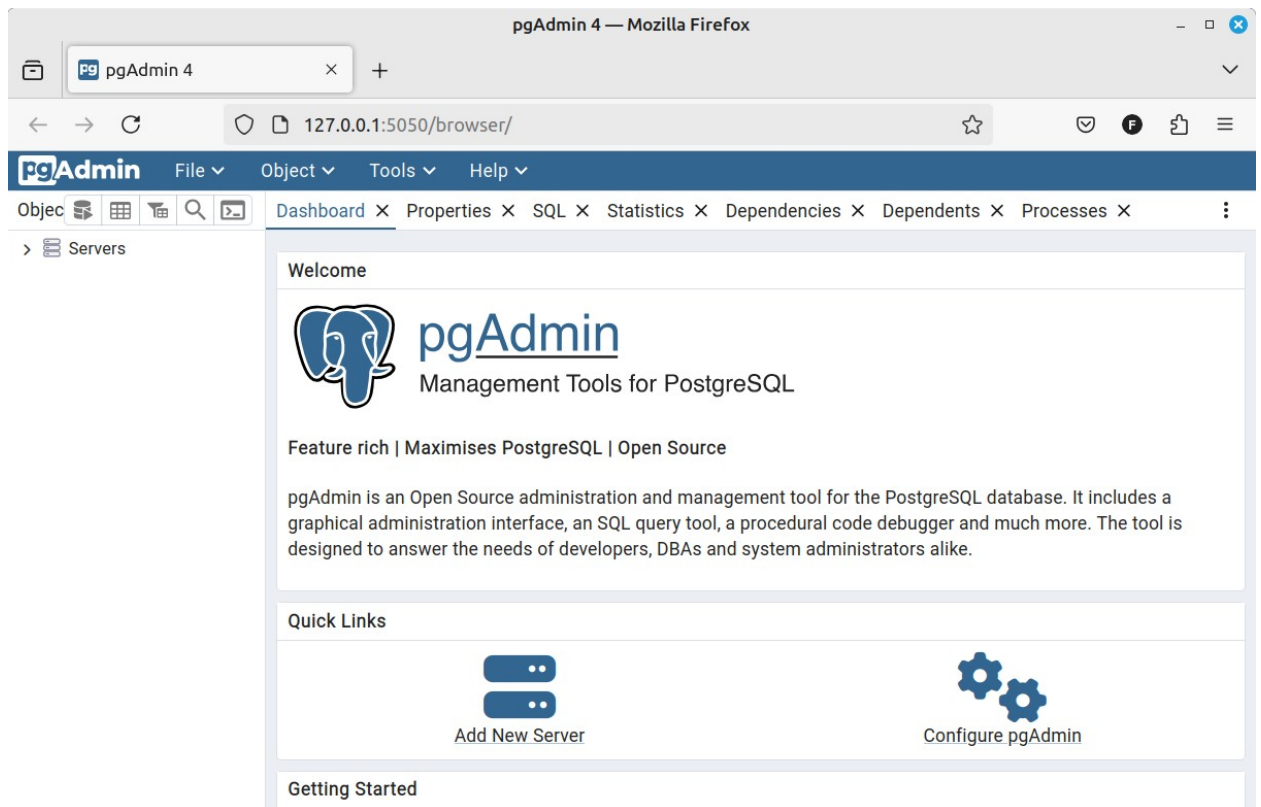
```
import os
DATA_DIR = os.path.realpath(os.path.expanduser(u'~/.pgadmin/'))
LOG_FILE = os.path.join(DATA_DIR, 'pgadmin4.log')
SQLITE_PATH = os.path.join(DATA_DIR, 'pgadmin4.db')
SESSION_DB_PATH = os.path.join(DATA_DIR, 'sessions')
STORAGE_DIR = os.path.join(DATA_DIR, 'storage')
SERVER_MODE = False
AZURE_CREDENTIAL_CACHE_DIR = os.path.join(DATA_DIR, 'azurecredentialcache')
```

Для запуска клиента pgAdmin4 выполняем команду

```
python
pgadmin4env/lib/python3.10/site-packages/pgadmin4/pgAdmin4.py
```

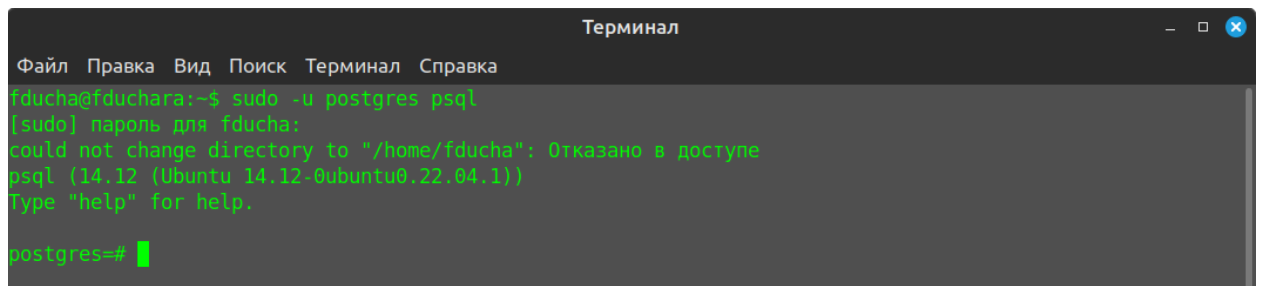
```
Терминал
Файл Правка Вид Поиск Терминал Справка
(pgadmin4env) fducha@fduchara:~/venvs/pgadmin4$ python pgadmin4env/lib/python3.10/site-packages/pgadmin4/pgAdmin4.py
NOTE: Configuring authentication for DESKTOP mode.
pgAdmin 4 - Application Initialisation
=====
Starting pgAdmin 4. Please navigate to http://127.0.0.1:5050 in your browser.
* Serving Flask app 'pgadmin'
* Debug mode: off
```

Переходим по предлагаемому адресу `http://127.0.0.1:5050` в браузере:



3. Подключение к СУБД через терминал и создание базы данных. Подключаемся к серверу СУБД и входим в консоль:

```
sudo -u postgres psql
```



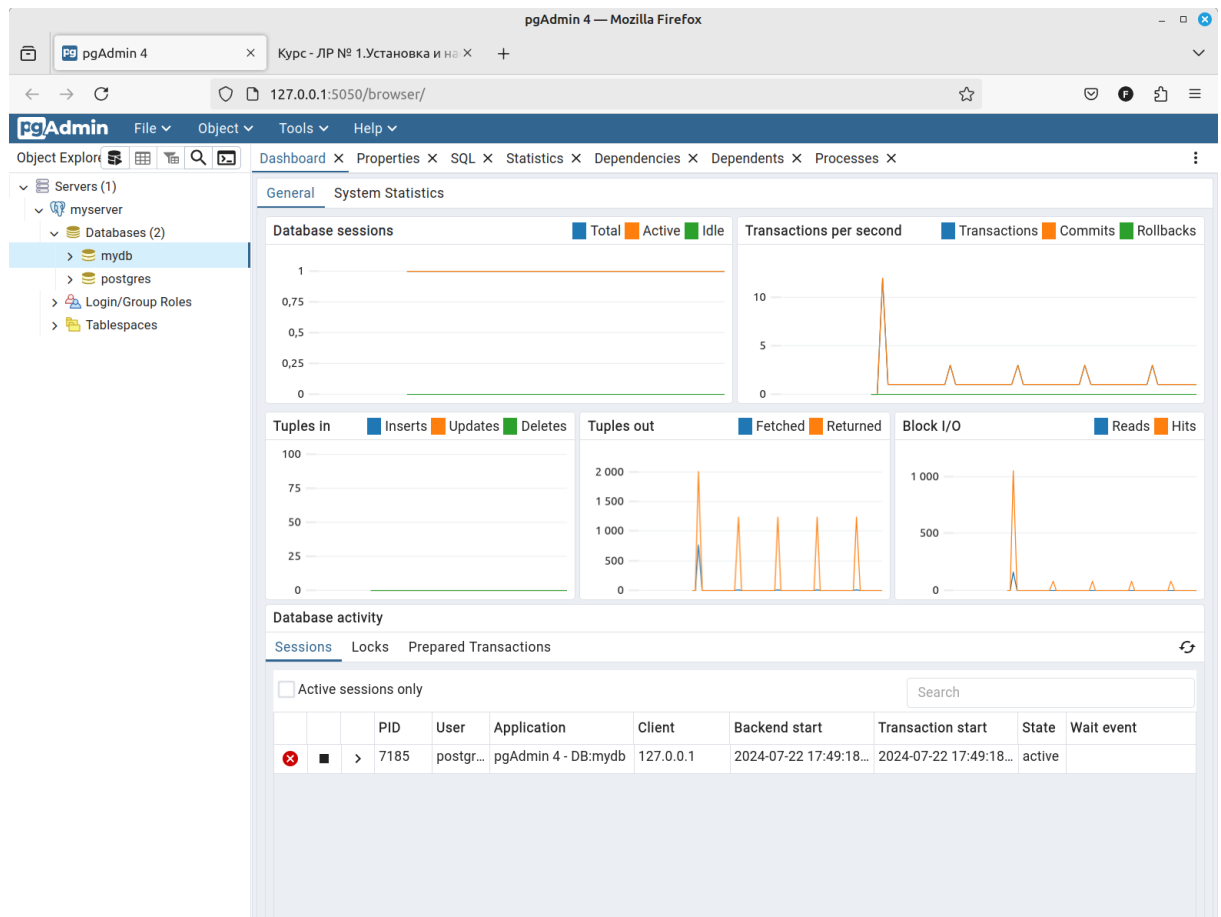
Создаем базу данных с именем mydb командой

```
CREATE DATABASE mydb;
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# CREATE DATABASE mydb;
CREATE DATABASE
postgres=#
```

Вывод сообщения «CREATE DATABASE» говорит об успешном выполнении команды.

При просмотре существующих БД через pgAdmin4 видно, что база с именем mydb присутствует.



## Ответы на контрольные вопросы.

### 1. Каковы основные функции СУБД?

Основные функции СУБД:

- управление данными во внешней памяти;
- управление транзакциями;
- восстановление базы данных;
- поддержка языков БД;
- поддержка языков БД;
- словарь данных;
- управление параллельным доступом;
- управление буферами оперативной памяти;

- контроль доступа к данным;
- поддержка целостности данных.

## 2. Назовите отличительные черты реляционных баз данных?

Одним из основных преимуществ реляционной модели является ее однородность. Все данные рассматриваются как хранимые в таблицах и только в таблицах. Каждая строка такой таблицы имеет один и тот же формат.

К числу достоинств реляционного подхода можно отнести:

- наличие небольшого набора абстракций, которые позволяют сравнительно просто моделировать большую часть распространенных предметных областей и допускают точные формальные определения, оставаясь интуитивно понятными;
- наличие простого и в то же время мощного математического аппарата, опирающегося главным образом на теорию множеств и математическую логику и обеспечивающего теоретический базис реляционного подхода к организации БД;
- возможность не навигационного манипулирования данными без необходимости знания конкретной физической организации баз данных во внешней памяти.

## 3. Что представляют собой базы данных?

База данных (БД) – именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, или иначе БД – это совокупность взаимосвязанных данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений в определенной предметной области. БД состоит из множества связанных файлов. БД, как правило, создается как общий ресурс всего предприятия, где данные являются интегрированными и общими. Интегрированные данные – это возможность представить базу данных как объединение нескольких отдельных файлов данных. Общие данные – это возможность использования отдельных областей данных в БД несколькими различными пользователями для разных целей.

БД содержит не только данные, всесторонне характеризующие деятельность самой организации, фирмы, процесса или другой предметной области, но и описания этих данных. Информацию о данных принято называть «метаданными», т.е. «данными о данных». В совокупности описания всех данных образуют словарь данных.

В БД должны храниться данные, логически связанные между собой. Для того чтобы данные можно было связать между собой, и связать так,



чтобы эти связи соответствовали реально существующим в данной предметной области, последнюю подвергают детальному анализу, выделяя сущности или объекты. Сущность или объект – это то, о чем необходимо хранить информацию. Сущности имеют некоторые характеристики, называемые атрибутами, которые тоже необходимо сохранять в БД. Атрибуты по своей внутренней структуре могут быть простыми, а могут быть сложными. Простые атрибуты могут быть представлены простыми типами данных. Различного рода графические изображения, являющиеся атрибутами сущностей, – это пример сложного атрибута. Определив сущности и их атрибуты, необходимо перейти к выявлению связей, которые могут существовать между некоторыми сущностями. Связь – это то, что объединяет две или более сущностей. Связи между сущностями также являются частью данных, и они также должны храниться в базе данных.

#### 4. Назовите преимущества архитектуры «клиент-сервер»?

В модели файлового сервера преимущество заключается в том, что в ней уже осуществлено разделение монопольного приложения на два взаимодействующих процесса. При этом сервер может обслуживать множество клиентов, обращающихся к нему с запросами.

В модели удаленного доступа преимущество заключается в том, что в ней сервер принимает и обрабатывает запросы со стороны клиентов, проверяет полномочия пользователей, гарантирует соблюдение ограничений целостности, выполняет обновление данных, выполняет запросы и возвращает результаты клиенту, поддерживает системный каталог, обеспечивает параллельный доступ к базе данных и ее восстановление. К тому же резко уменьшается загрузка сети, так как по ней от клиентов к серверу передаются не файловые команды, а запросы на SQL, и их объем существенно меньше. В ответ на запросы клиент получает только данные, соответствующие запросу, а не блоки файлов, как в модели файлового сервера.

В модели сервера баз данных преимущество заключается в том, что в ней существует механизм хранимых процедур, который позволяет создавать подпрограммы, работающие на сервере и управляющие его процессами.

Таким образом, размещение на сервере хранимых процедур означает, что прикладные функции приложения разделены между клиентом и сервером. Трафик обмена информацией между клиентом и сервером резко уменьшается.

Централизованный контроль целостности базы данных в модели сервера баз данных выполняется с использованием механизма триггеров. Триггеры также являются частью БД.

5. Какие функции перешли к среднему уровню обработки данных в трехуровневой архитектуре «клиент-сервер»?

Среднему уровню, который может содержать один или несколько серверов приложений, выделяются общие не загружаемые функции для клиентов: наиболее общие прикладные функции клиента, функции, поддерживающие сетевую доменную операционную среду, каталоги с данными, функции, обеспечивающие обмен сообщениями и поддержку запросов.

6. Определите роль сервера в двухуровневой архитектуре «клиент-сервер»

Сервер принимает и обрабатывает запросы со стороны клиентов, проверяет полномочия пользователей, гарантирует соблюдение ограничений целостности, выполняет обновление данных, выполняет запросы и возвращает результаты клиенту, поддерживает системный каталог, обеспечивает параллельный доступ к базе данных и ее восстановление.

## Лабораторная работа № 2.

**Название работы:** Создание таблиц.

**Цели работы:** Изучение и практическое применение функций для создания таблиц.

### Задание:

1. Напишите три SQL запроса, создающие таблицы в соответствии с вашим вариантом. В работе вы должны продемонстрировать использование различных типов данных: целочисленные, числа с плавающей точкой, числа с фиксированной точкой, строковые типы (с фиксированной и с переменной длиной строки), дата. Чем больше разных типов данных вы используете, тем лучше. Приведите в отчете скриншоты терминала с выполненными командами и их результатом.

2. С помощью pgAdmin заполните таблицы данными: в каждую таблицу добавьте 10 строк.

### Выполнение лабораторной работы.

Изучены теоретические сведения лабораторной работы по применению функций для создания таблиц в PostgreSQL.

#### 1. Создание таблиц.

В данной работе создадим 3 связанные таблицы: Клиенты (clients), Товары (products) и Заказы (orders). Используем для это следующие данные:

#### Клиенты

Имя поля	Назначение поля	Тип данных
id	Уникальный идентификатор клиента	Целое, serial
name	Имя клиента	Строка длиной 256 символов, varchar(255)
phone	Телефон клиента	Строка длиной 11 символов, char(11)

#### Товары

Имя поля	Назначение поля	Тип данных
articul	Уникальный идентификатор товара	Строка длиной 10 символов, char(10)
name	Название товара	Строка длиной 255 символов, varchar(255)

price	Стоимость	Денежная сумма с фиксированной дробной частью, money
-------	-----------	--

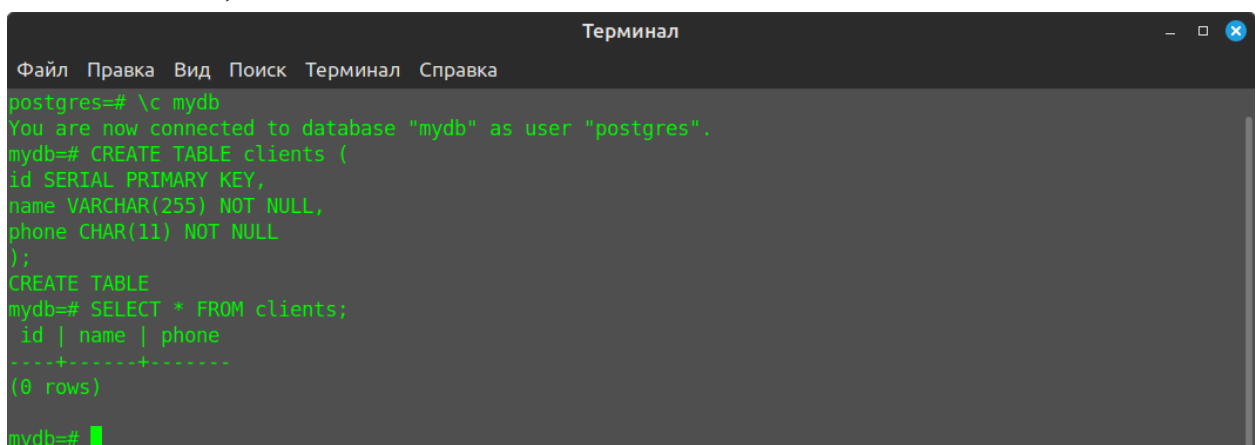
### Заказы

Имя поля	Назначение поля	Тип данных
id	Уникальный идентификатор заказа	Целое, serial
client_id	Уникальный идентификатор клиента	Целое (внешний индекс таблицы клиентов), integer
articul	Уникальный идентификатор товара	Строка длиной 10 символов (внешний индекс таблицы товаров), char(10)
count	Количество единиц товара в заказе	Вещественное число, real
order_date	Дата создания заказа	Дата, date

Для начала подключаемся к базе данных mydb командой «\c mydb», в результате чего меняется приглашение консоли. Далее, создаем таблицу clients командой CREATE TABLE:

```
CREATE TABLE clients (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    phone CHAR(11) NOT NULL
);
```

После создания таблицы проверяем результат командой «SELECT \* FROM clients;».



```

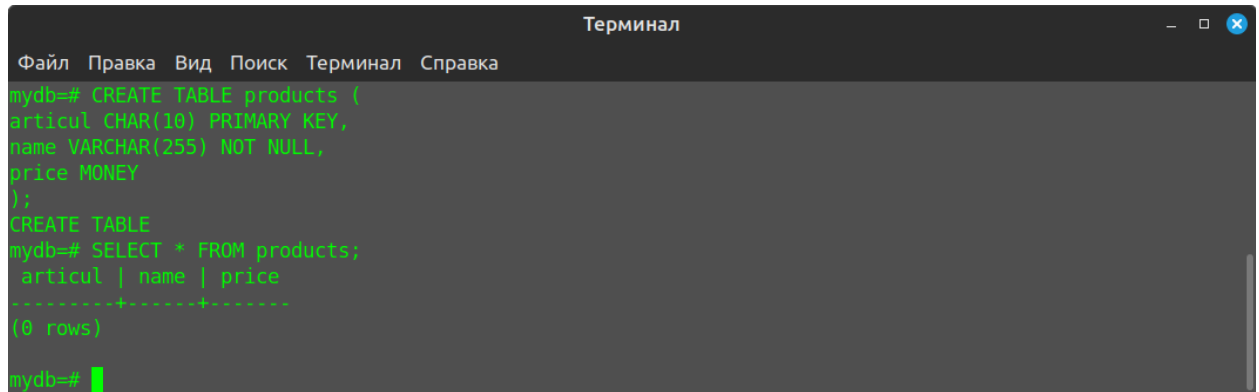
Терминал
Файл Правка Вид Поиск Терминал Справка
postgres=# \c mydb
You are now connected to database "mydb" as user "postgres".
mydb=# CREATE TABLE clients (
id SERIAL PRIMARY KEY,
name VARCHAR(255) NOT NULL,
phone CHAR(11) NOT NULL
);
CREATE TABLE
mydb=# SELECT * FROM clients;
 id | name | phone
-----+-----+-----
(0 rows)

mydb=#

```

Далее, создаем таблицу products:

```
CREATE TABLE products (  
    articul CHAR(10) PRIMARY KEY,  
    name VARCHAR(255) NOT NULL,  
    price MONEY  
);
```



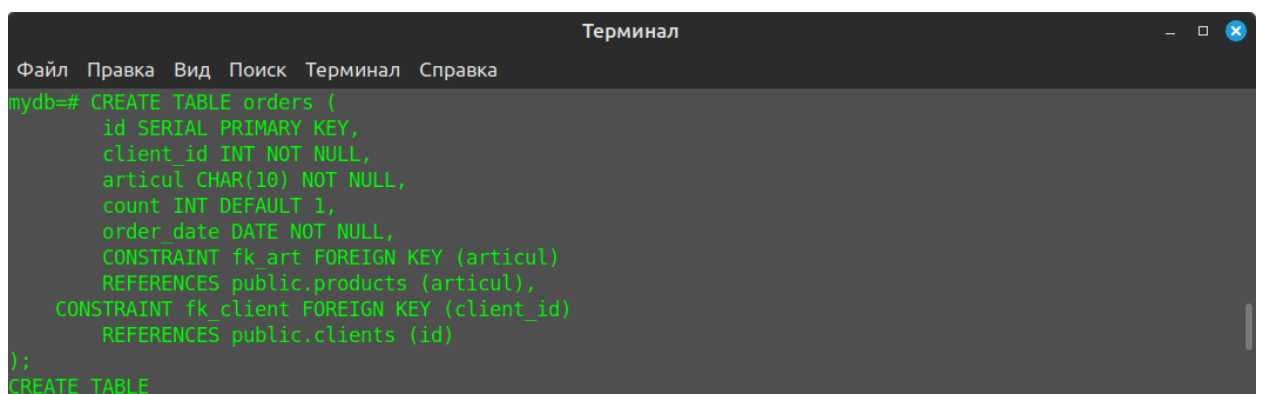
```
Терминал  
Файл  Правка  Вид  Поиск  Терминал  Справка  
mydb=# CREATE TABLE products (  
    articul CHAR(10) PRIMARY KEY,  
    name VARCHAR(255) NOT NULL,  
    price MONEY  
);  
CREATE TABLE  
mydb=# SELECT * FROM products;  
 articul | name | price  
-----+-----+-----  
(0 rows)  
mydb=#
```

После создания таблицы проверяем результат командой «SELECT \* FROM products;».

Создаем таблицу orders, в которой, в отличие от двух предыдущих, добавим два внешних ключа: client\_id — указывает идентификатор клиента и articul — идентификатор товара.

```
CREATE TABLE orders (  
    id SERIAL PRIMARY KEY,  
    client_id INT NOT NULL,  
    articul CHAR(10) NOT NULL,  
    count INT DEFAULT 1,  
    order_date DATE NOT NULL,  
    CONSTRAINT fk_art FOREIGN KEY (articul)  
        REFERENCES public.products (articul),  
    CONSTRAINT fk_client FOREIGN KEY (client_id)  
        REFERENCES public.clients (id)  
);
```

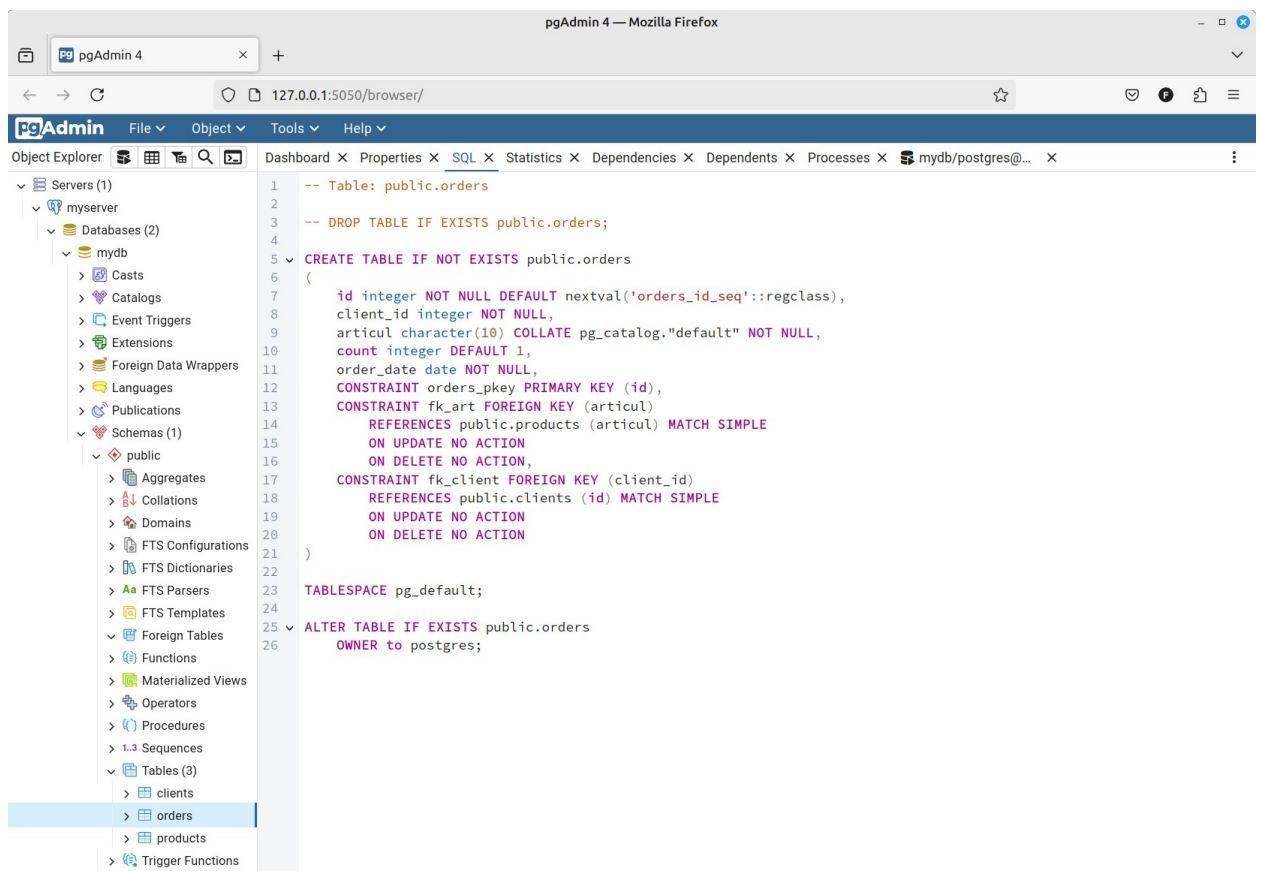
Выполняем и проверяем наличие таблицы в БД командой «\d orders»:



```
Терминал  
Файл  Правка  Вид  Поиск  Терминал  Справка  
mydb=# CREATE TABLE orders (  
    id SERIAL PRIMARY KEY,  
    client_id INT NOT NULL,  
    articul CHAR(10) NOT NULL,  
    count INT DEFAULT 1,  
    order_date DATE NOT NULL,  
    CONSTRAINT fk_art FOREIGN KEY (articul)  
        REFERENCES public.products (articul),  
    CONSTRAINT fk_client FOREIGN KEY (client_id)  
        REFERENCES public.clients (id)  
);  
CREATE TABLE
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# \d orders
Table "public.orders"
Column | Type          | Collation | Nullable | Default
-----+-----+-----+-----+-----
id      | integer       |           | not null | nextval('orders_id_seq'::regclass)
client_id | integer       |           | not null |
articul | character(10) |           | not null |
count   | integer       |           |          | 1
order_date | date          |           | not null |
Indexes:
    "orders_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "fk_art" FOREIGN KEY (articul) REFERENCES products(articul)
    "fk_client" FOREIGN KEY (client_id) REFERENCES clients(id)
mydb=#
```

Проверяем наличие таблиц в pgAdmin4: в боковой панели находим нашу БД mydb, далее Schemas → Tables. В контекстном меню Tables выбираем команду Refresh и убеждаемся в наличии 3 таблиц: clients, orders и products. При выборе вкладки SQL в верхней панели при выборе таблицы можно просмотреть SQL-запрос создания таблицы. Как можно убедиться на рисунке ниже, СУБД PostgreSQL несколько изменяет SQL-запрос введенный в консоли, указывая больше различных деталей.



```
pgAdmin 4 — Mozilla Firefox
127.0.0.1:5050/browser/
pgAdmin
File Object Tools Help
Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes mydb/postgres@...
Servers (1)
  myserver
    Databases (2)
      mydb
        Casts
        Catalogs
        Event Triggers
        Extensions
        Foreign Data Wrappers
        Languages
        Publications
        Schemas (1)
          public
            Aggregates
            Collations
            Domains
            FTS Configurations
            FTS Dictionaries
            FTS Parsers
            FTS Templates
            Foreign Tables
            Functions
            Materialized Views
            Operators
            Procedures
            Sequences
            Tables (3)
              clients
              orders
              products
            Trigger Functions

1  -- Table: public.orders
2
3  -- DROP TABLE IF EXISTS public.orders;
4
5  CREATE TABLE IF NOT EXISTS public.orders
6  (
7      id integer NOT NULL DEFAULT nextval('orders_id_seq'::regclass),
8      client_id integer NOT NULL,
9      articul character(10) COLLATE pg_catalog."default" NOT NULL,
10     count integer DEFAULT 1,
11     order_date date NOT NULL,
12     CONSTRAINT orders_pkey PRIMARY KEY (id),
13     CONSTRAINT fk_art FOREIGN KEY (articul)
14         REFERENCES public.products (articul) MATCH SIMPLE
15         ON UPDATE NO ACTION
16         ON DELETE NO ACTION,
17     CONSTRAINT fk_client FOREIGN KEY (client_id)
18         REFERENCES public.clients (id) MATCH SIMPLE
19         ON UPDATE NO ACTION
20         ON DELETE NO ACTION
21 )
22
23 TABLESPACE pg_default;
24
25 ALTER TABLE IF EXISTS public.orders
26     OWNER to postgres;
```

## 2. Заполнение таблиц данными с помощью pgAdmin4.

Первой заполним таблицу Клиенты, пусть это будут современные российские актеры и актрисы. Для этого вызываем контекстное меню таблицы clients, выбираем пункты «View/Edit Data» → «All Rows».

В верхней части во вкладке «Query» уже находится SQL-запрос, выводящий все записи таблицы clients, отсортированные по возрастанию по полю id.

В нижней части во вкладке «Data Output» отображаются данные таблицы. Для добавления новой строки нажимаем кнопку «Add row» (крайняя слева) и осуществляем ввод данных. Когда все данные введены, нажимаем кнопку «Save Data Changes» (третья справа) или клавишу F6.

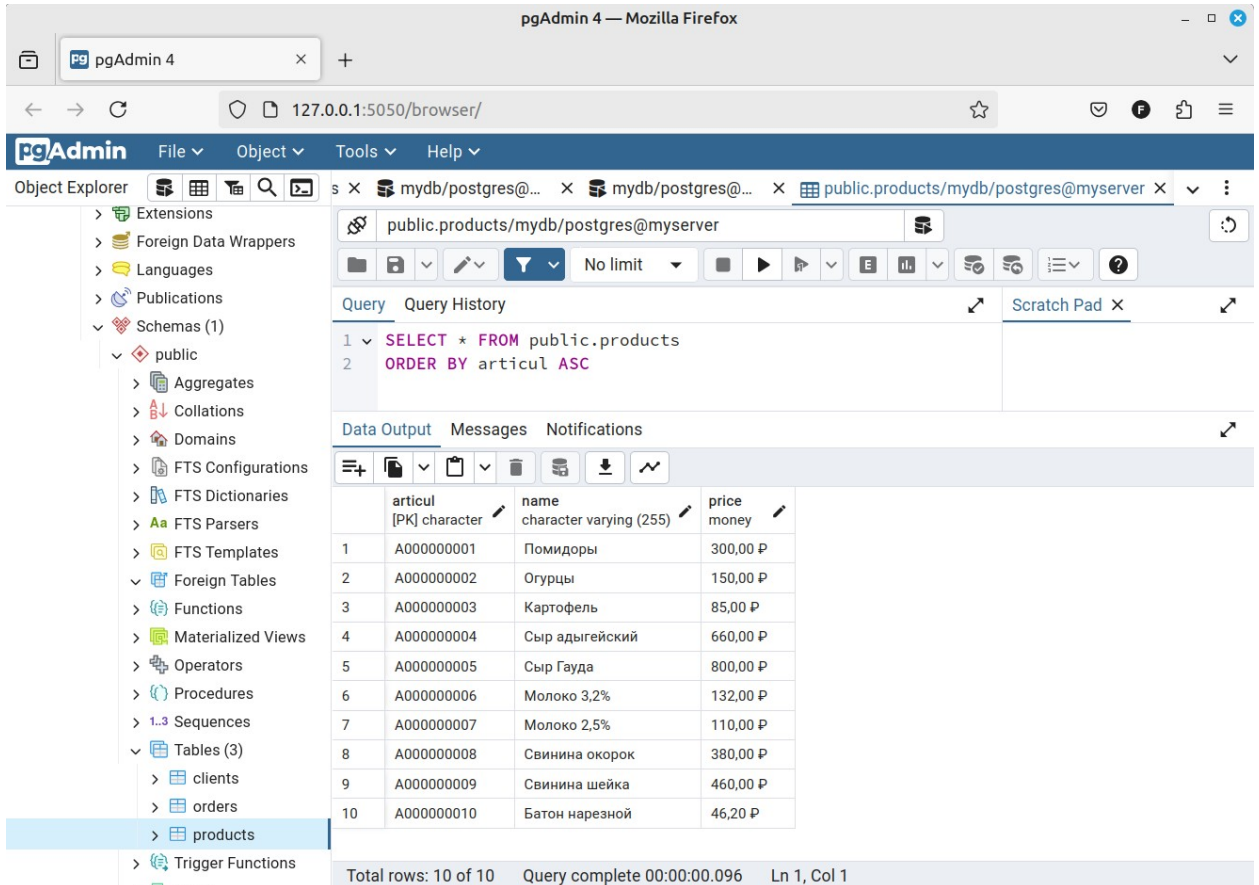
Для представления данных в отсортированном виде запускаем имеющийся SQL-запрос кнопкой «Execute script» или клавишей F5.

The screenshot shows the pgAdmin 4 web interface in a Mozilla Firefox browser. The browser address bar shows the URL `127.0.0.1:5050/browser/`. The pgAdmin 4 interface has a top menu bar with 'File', 'Object', 'Tools', and 'Help'. On the left is the 'Object Explorer' tree, which is expanded to show the 'public' schema and the 'clients' table. The main panel is divided into three tabs: 'Query', 'Data Output', and 'Messages'. The 'Query' tab is active, showing a SQL query: `SELECT * FROM public.clients ORDER BY id ASC`. The 'Data Output' tab is also visible, showing a table with 10 rows of data. The table has three columns: 'id' (integer, primary key), 'name' (character varying (255)), and 'phone' (character). The data is as follows:

id	name	phone
1	Сергей Пускепалис	79001111111
2	Таисия Вилкова	79002222222
3	Евгений Миронов	79003333333
4	Екатерина Климова	79004444444
5	Алексей Серебряков	79005555555
6	Кристина Асмус	79006666666
7	Виктор Хориняк	79007777777
8	Фёдор Добронравов	79008888888
9	Аглая Тарасова	79009999999
10	Сергей Безруков	79001234567

At the bottom of the interface, a status bar indicates 'Total rows: 10 of 10', 'Query complete 00:00:00.114', and 'Ln 2, Col 16'.

Далее заполняем таблицу Товары аналогичным способом.



pgAdmin 4 — Mozilla Firefox

127.0.0.1:5050/browser/

pgAdmin File Object Tools Help

Object Explorer

- Extensions
- Foreign Data Wrappers
- Languages
- Publications
- Schemas (1)
  - public
    - Aggregates
    - Collations
    - Domains
    - FTS Configurations
    - FTS Dictionaries
    - FTS Parsers
    - FTS Templates
    - Foreign Tables
    - Functions
    - Materialized Views
    - Operators
    - Procedures
    - Sequences
    - Tables (3)
      - clients
      - orders
      - products
    - Trigger Functions

public.products/mydb/postgres@myserver

Query Query History

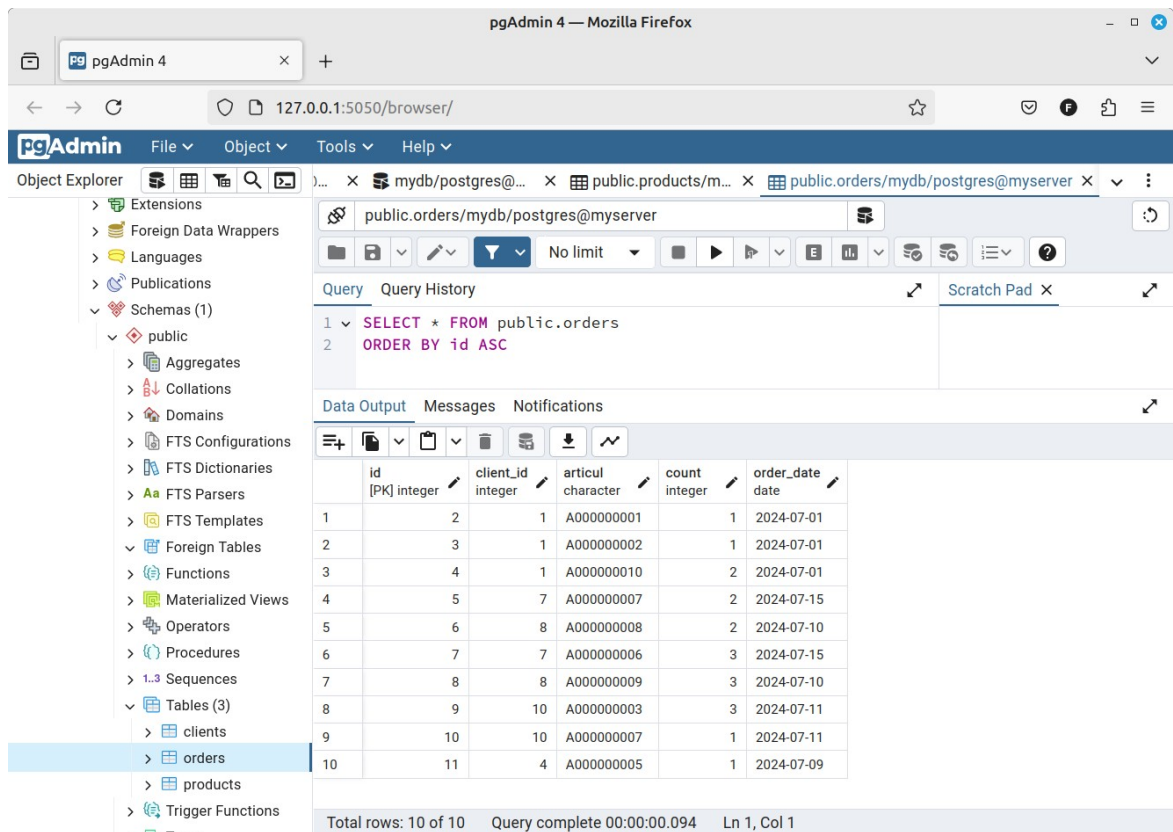
```
1 SELECT * FROM public.products
2 ORDER BY artikul ASC
```

Data Output Messages Notifications

	artikul [PK] character	name character varying (255)	price money
1	A000000001	Помидоры	300,00 P
2	A000000002	Огурцы	150,00 P
3	A000000003	Картофель	85,00 P
4	A000000004	Сыр адыгейский	660,00 P
5	A000000005	Сыр Гауда	800,00 P
6	A000000006	Молоко 3,2%	132,00 P
7	A000000007	Молоко 2,5%	110,00 P
8	A000000008	Свинина окорок	380,00 P
9	A000000009	Свинина шейка	460,00 P
10	A000000010	Батон нарезной	46,20 P

Total rows: 10 of 10 Query complete 00:00:00.096 Ln 1, Col 1

Заполняем таблицу Заказы.



pgAdmin 4 — Mozilla Firefox

127.0.0.1:5050/browser/

pgAdmin File Object Tools Help

Object Explorer

- Extensions
- Foreign Data Wrappers
- Languages
- Publications
- Schemas (1)
  - public
    - Aggregates
    - Collations
    - Domains
    - FTS Configurations
    - FTS Dictionaries
    - FTS Parsers
    - FTS Templates
    - Foreign Tables
    - Functions
    - Materialized Views
    - Operators
    - Procedures
    - Sequences
    - Tables (3)
      - clients
      - orders
      - products
    - Trigger Functions

public.orders/mydb/postgres@myserver

Query Query History

```
1 SELECT * FROM public.orders
2 ORDER BY id ASC
```

Data Output Messages Notifications

	id [PK] integer	client_id integer	artikul character	count integer	order_date date
1	2	1	A000000001	1	2024-07-01
2	3	1	A000000002	1	2024-07-01
3	4	1	A000000010	2	2024-07-01
4	5	7	A000000007	2	2024-07-15
5	6	8	A000000008	2	2024-07-10
6	7	7	A000000006	3	2024-07-15
7	8	8	A000000009	3	2024-07-10
8	9	10	A000000003	3	2024-07-11
9	10	10	A000000007	1	2024-07-11
10	11	4	A000000005	1	2024-07-09

Total rows: 10 of 10 Query complete 00:00:00.094 Ln 1, Col 1



## Ответы на контрольные вопросы.

1. Создать базу данных basa1, причем для данных на приросте 10%, для журнала транзакций – на диске E один файл с начальным размером 50 Мб, но не более 100 Мб, с величиной прироста 10 Мб.

```
CREATE DATABASE basa1
ON PRIMARY
(NAME=b1, FILENAME='D:\user\b1.mdf', FILEGROWTH=10%)
LOG ON
(NAME=l1, FILENAME='E:\user\l1.ldf',
SIZE=50MB, MAXSIZE=100, FILEGROWTH=10);
```

2. Создать базу данных basa1, причем для данных на диске D определить два файла с начальным размером по 100 Мб, но не более 500 Мб, с величиной прироста 10%, для журнала транзакций - на диске E один файл с начальным размером 50 Мб, но не более 100 Мб, с величиной прироста 10 Мб.

```
CREATE DATABASE basa1
ON PRIMARY
(NAME=b1, FILENAME='D:\user\b1.mdf',
SIZE=100MB, MAXSIZE=500, FILEGROWTH=10%),
(NAME=b2, FILENAME='D:\user\b2.mdf',
SIZE=100MB, MAXSIZE=500, FILEGROWTH=10%)
LOG ON
(NAME=l1, FILENAME='E:\user\l1.ldf',
SIZE=50MB, MAXSIZE=100, FILEGROWTH=10););
```

3. Создать базу данных basa1, причем для данных на диске D определить два файла с начальным размером по 100 Мб, но не более 500 Мб, с величиной прироста 10%, для журнала транзакций - на диске E один файл с начальным размером 50 Мб, но не более 100 Мб, с величиной прироста 10 %.

4. Создать базу данных basa1, причем для данных на диске D определить два файла с начальным размером по 100 Мб, но не более 500 Мб, с величиной прироста 10%, для журнала транзакций - на диске E один файл с начальным размером 50 Мб, но не более 100 Мб, с величиной прироста 10 Мб.

**2, 3 и 4 вопросы дублируются.**

## Лабораторная работа № 3.

**Название работы:** Извлечение информации из таблиц.

**Цели работы:** Изучение и практическое применение возможностей SQL для извлечения информации из таблиц.

### Задание:

1. Напишите запрос, выбирающий все поля и все строки из таблицы, используемой в данной лабораторной. При этом таблица должна содержать 10-15 строк. Приведите содержимое данной таблицы.
2. Напишите запрос, демонстрирующий выбор нескольких (не всех) полей таблицы с удалением дубликатов строк (DISTINCT).
3. Напишите запрос, демонстрирующий выбор всех полей (\*) и задание условий выборки в виде операций сравнения (>, <, =) и логических операций (AND, OR, NOT).
4. Напишите запрос, демонстрирующий работу конструкций IN, BETWEEN, IS NULL, IS NOT NULL.
5. Напишите запрос, демонстрирующий работу конструкции LIKE (с символами "%" и "\_").
6. Напишите запрос, демонстрирующий вычисление арифметических выражений как в условиях выборки (после WHERE), так и в списке выбора (после SELECT) с заданием имени для результата выражения.

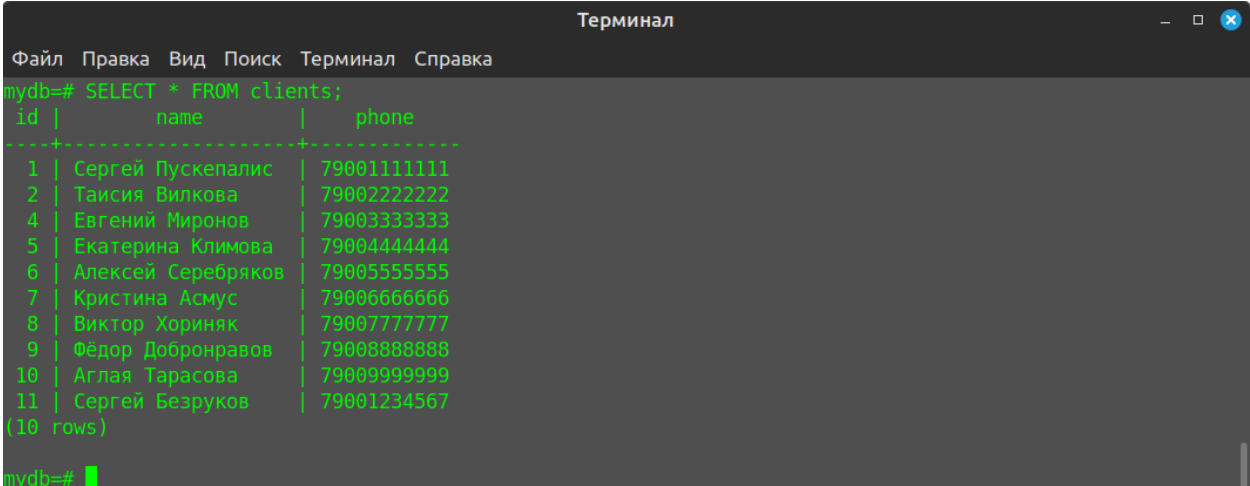
### Выполнение лабораторной работы.

Изучены теоретические сведения лабораторной работы по применению функций для создания таблиц в PostgreSQL.

1. Запрос, выбирающий все поля и все строки из таблицы.

Для вывода всех полей таблицы «clients» используем символ «\*» вместо перечисления всех полей. При отсутствии ограничительных параметров запроса будут выведены все строки таблицы.

```
SELECT * FROM clients;
```



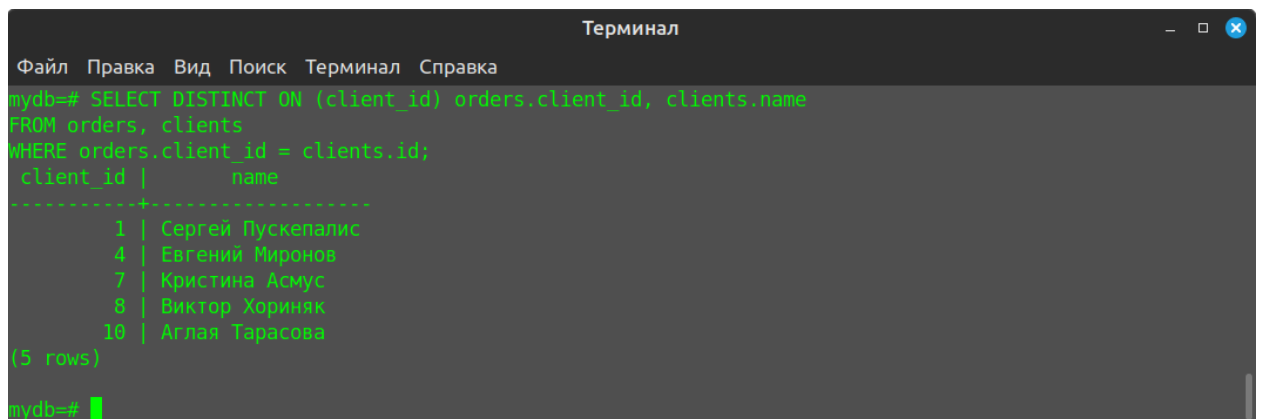
```
Терминал
Файл  Правка  Вид  Поиск  Терминал  Справка
mydb=# SELECT * FROM clients;
 id |      name      |      phone
-----+-----+-----
  1 | Сергей Пускепалис | 79001111111
  2 | Таисия Вилкова   | 79002222222
  4 | Евгений Миронов  | 79003333333
  5 | Екатерина Климова | 79004444444
  6 | Алексей Серебряков | 79005555555
  7 | Кристина Асмус   | 79006666666
  8 | Виктор Хориняк   | 79007777777
  9 | Федор Добронравов | 79008888888
 10 | Аглая Тарасова   | 79009999999
 11 | Сергей Безруков   | 79001234567
(11 rows)

mydb=#
```

2. Запрос, демонстрирующий выбор нескольких (не всех) полей таблицы с удалением дубликатов строк.

Рассмотрим таблицу Заказы: видно, что некоторые клиенты заказывают несколько товаров, а некоторые клиенты не делали ни одного заказа. При помощи ключевого слова DISTINCT в запросе можно узнать, какие клиенты делали заказы.

```
SELECT DISTINCT ON (client_id) orders.client_id,  
clients.name  
FROM orders, clients  
WHERE orders.client_id = clients.id;
```



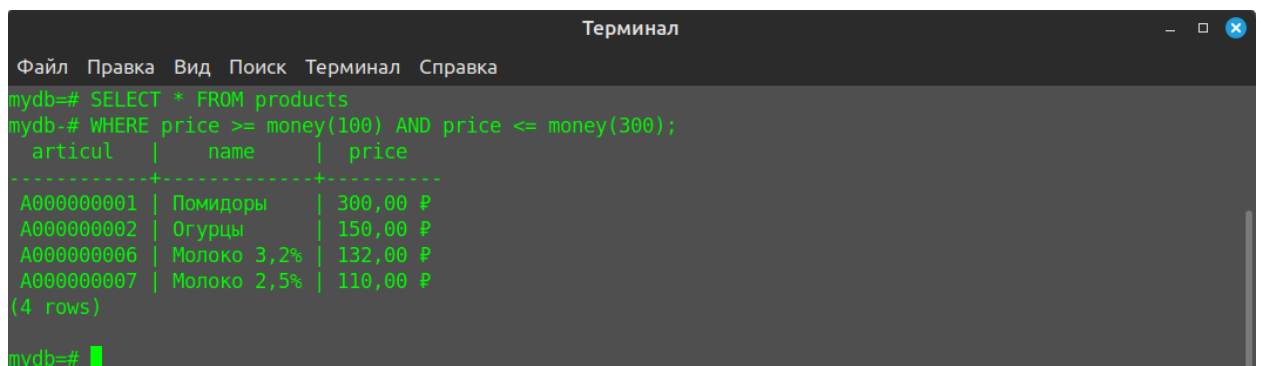
```
Терминал  
Файл Правка Вид Поиск Терминал Справка  
mydb=# SELECT DISTINCT ON (client_id) orders.client_id, clients.name  
FROM orders, clients  
WHERE orders.client_id = clients.id;  
 client_id |      name  
-----+-----  
         1 | Сергей Пускепалис  
         4 | Евгений Миронов  
         7 | Кристина Асмус  
         8 | Виктор Хориняк  
        10 | Аглая Тарасова  
(5 rows)  
mydb=#
```

3. Запрос, демонстрирующий выбор всех полей и задание условий выборки в виде операций сравнения (>, <, =) и логических операций (AND, OR, NOT).

Запрос, выбирающий продукты, стоимостью в диапазоне от 100 до 300 рублей.

```
SELECT * FROM products  
WHERE price >= money(100) AND price <= money(300);
```

Для указания стоимости в запросе используется приведение типа целого числа к денежному типу.



```
Терминал  
Файл Правка Вид Поиск Терминал Справка  
mydb=# SELECT * FROM products  
mydb=# WHERE price >= money(100) AND price <= money(300);  
 article |      name      | price  
-----+-----+-----  
 A000000001 | Помидоры      | 300,00 P  
 A000000002 | Огурцы         | 150,00 P  
 A000000006 | Молоко 3,2%   | 132,00 P  
 A000000007 | Молоко 2,5%   | 110,00 P  
(4 rows)  
mydb=#
```

Запрос, выбирающий продукты, стоимостью до 100 рублей или более 500 рублей.

```
SELECT * FROM products  
WHERE price <= money(100) OR price >= money(500);
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT * FROM products
WHERE price <= money(100) OR price >= money(500);
 articl |      name      | price
-----+-----+-----
 A000000003 | Картофель      | 85,00 P
 A000000004 | Сыр адыгейский | 660,00 P
 A000000005 | Сыр Гауда       | 800,00 P
 A000000010 | Батон нарезной  | 46,20 P
(4 rows)

mydb=#
```

4. Запрос, демонстрирующий работу конструкций IN, BETWEEN, IS NULL, IS NOT NULL.

Запрос, выбирающий продукты, стоимостью 110 и 380 рублей.

```
SELECT * FROM public.products
WHERE price IN (money(110), money(380));
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT * FROM products
WHERE price IN (money(110), money(380));
 articl |      name      | price
-----+-----+-----
 A000000007 | Молоко 2,5%    | 110,00 P
 A000000008 | Свинина окорок | 380,00 P
(2 rows)

mydb=#
```

Запрос, выбирающий продукты, стоимостью в диапазоне от 100 до 300 рублей.

```
SELECT * FROM public.products
WHERE price BETWEEN money(100) AND money(300);
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT * FROM products
WHERE price BETWEEN money(100) AND money(300);
 articl |      name      | price
-----+-----+-----
 A000000001 | Помидоры       | 300,00 P
 A000000002 | Огурцы         | 150,00 P
 A000000006 | Молоко 3,2%    | 132,00 P
 A000000007 | Молоко 2,5%    | 110,00 P
(4 rows)

mydb=#
```

5. Запрос, демонстрирующий работу конструкции LIKE (с символами "%" и "\_").

Запрос, выбирающий клиентов, имя которых начинается с буквы «Е».

```
SELECT * FROM clients
WHERE name LIKE 'E%';
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT * FROM clients
mydb=# WHERE name LIKE 'E%';
 id |      name      |      phone
-----+-----+-----
  4 | Евгений Миронов | 79003333333
  5 | Екатерина Климова | 79004444444
(2 rows)
mydb=#
```

Запрос, выбирающий клиентов, имя которых состоит из 6 букв, начинается с буквы «С» и заканчивается буквой «й».

```
SELECT * FROM clients
WHERE name LIKE 'C_____й%';
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT * FROM clients
mydb=# WHERE name LIKE 'C_____й%';
 id |      name      |      phone
-----+-----+-----
  1 | Сергей Пускепалис | 79001111111
 11 | Сергей Безруков   | 79001234567
(2 rows)
mydb=#
```

6. Запрос, демонстрирующий вычисление арифметических выражений как в условиях выборки (после WHERE), так и в списке выбора (после SELECT) с заданием имени для результата выражения.

Запрос, выбирающий общую сумму заказов для каждого клиента, сделавшего заказы.

```
SELECT clients.name AS client,
       SUM(orders.count * products.price) AS price
FROM orders
LEFT JOIN clients ON orders.client_id = clients.id
LEFT JOIN products ON orders.articul = products.articul
GROUP BY client;
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT clients.name AS client, SUM(orders.count * products.price) AS price
FROM orders
LEFT JOIN clients ON orders.client_id = clients.id
LEFT JOIN products ON orders.articul = products.articul
GROUP BY client;
      client      | price
-----+-----
 Евгений Миронов  | 800,00 P
 Сергей Пускепалис | 542,40 P
 Кристина Асмус   | 616,00 P
 Виктор Хориняк   | 2 140,00 P
 Аглая Тарасова   | 365,00 P
(5 rows)

mydb=#
```

## Ответы на контрольные вопросы.

1. Дана таблица Рейс. Вывести в убывающем порядке список рейсов, вылетающих не позднее 1 апреля в Москву, Петербург или Самару, стоимость билета не более 1500 р.; в Саратов - не позднее 7 апреля, стоимость билета - от 500 до 800 р.

Пусть таблица Рейс (flights) имеет столбцы Номер рейса (flight\_id), Дата (date), Место назначения (destination), Расстояние (lenght) и Стоимость (price).

```
SELECT * FROM flights
WHERE (date <= '2024-04-01'
AND dectination IN ('Москва', 'Петербург', 'Самара')
AND price <= 1500)
OR (date <= '2024-04-07'
AND dectination = 'Саратов'
AND (price BETWEEN 500 AND 800))
ORDER BY date DESC;
```

2. Дана таблица Город. Вывести в алфавитном порядке список городов Поволжского региона, в коде которых встречается цифра 9, а в названии города на втором месте стоит буква «д» или «ж».

Пусть таблица Город (cities) имеет столбцы Имя (name), Регион (region) и Код (code).

```
SELECT * FROM cities
WHERE region = 'Поволжье'
AND code LIKE '%9%'
AND (name LIKE '_д%' OR name LIKE '_ж%')
ORDER BY name ASC;
```

3. Дана таблица Автор. Вывести в алфавитном порядке фамилии авторов из Самары, в телефонном номере которых на первом или третьем месте стоит цифра от 5 до 8, а последними являются цифры 7 и 8.

Пусть таблица Автор (authors) имеет столбцы Фамилия (name), Город

(city) и Телефон (phone), а номер телефона имеет формат +7 XXX XX XX, где X — цифра от 0 до 9.

```
SELECT * FROM authors
WHERE city = 'Самара'
AND (phone LIKE '%7' OR phone LIKE '%8')
AND (phone LIKE '_[5,8]%' OR phone LIKE '__[5,8]%')
ORDER BY name ASC;
```

4. Дана таблица Блюдо. Вывести в алфавитном порядке фамилии поваров, блюда которых относятся к десерту или выпечке, стоимость не превышает 50 руб., а калорийность не больше 300 ккал.

Пусть таблица Блюдо (dishes) имеет столбцы Повар (chief), Тип (type), Стоимость (price) и Калорийность (value).

```
SELECT * FROM dishes
WHERE type IN ('Десерт', 'Выпечка')
AND price <= 50
AND value <= 300
ORDER BY chief ASC;
```

5. Дана таблица Рейс. Вывести список рейсов, продолжительность маршрутов которых не более 500 км и не менее 100 км, а стоимость билета - от 800 до 1500 руб.

Пусть таблица Рейс (flights) имеет столбцы Номер рейса (flight\_id), Дата (date), Место назначения (destination), Расстояние (lenght) и Стоимость (price).

```
SELECT * FROM flights
WHERE (lenght BETWEEN 100 AND 500)
AND (price BETWEEN 800 AND 1500);
```

## Лабораторная работа № 4.

**Название работы:** Обобщение данных с помощью агрегатных функций.

**Цели работы:** Изучение и практическое применение методов обобщения данных с помощью агрегатных функций.

### Задание:

1. Напишите запрос, выбирающий все поля и все строки из таблицы, используемой в данной лабораторной работе. При этом таблица должна содержать 10-15 строк. Приведите содержимое данной таблицы.
2. Напишите запрос, демонстрирующий возможности функций MAX и MIN.
3. Напишите запрос, демонстрирующий возможности функций AVG и SUM.
4. Напишите запрос, демонстрирующий работу конструкций COUNT, COUNT(\*) и COUNT(DISTINCT).
5. Напишите запрос, демонстрирующий работу конструкции GROUP BY.
6. Напишите запрос, демонстрирующий работу конструкции HAVING.
7. Напишите запрос, демонстрирующий совместную работу конструкций HAVING и WHERE.

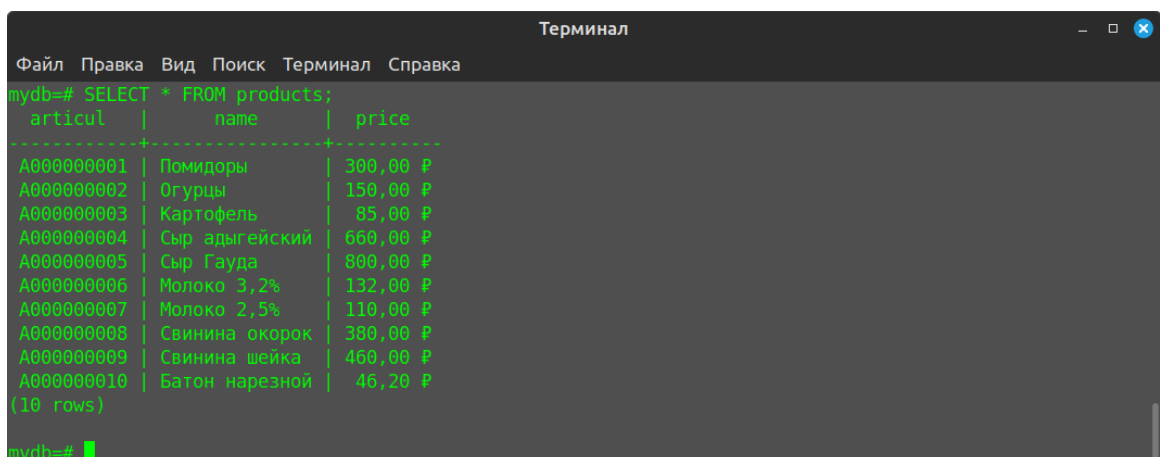
### Выполнение лабораторной работы.

Изучены теоретические сведения лабораторной работы по применению функций для создания таблиц в PostgreSQL.

1. Запрос, выбирающий все поля и все строки из таблицы.

Для вывода всех полей таблицы «products» используем символ «\*» вместо перечисления всех полей. При отсутствии ограничительных параметров запроса будут выведены все строки таблицы.

```
SELECT * FROM products;
```

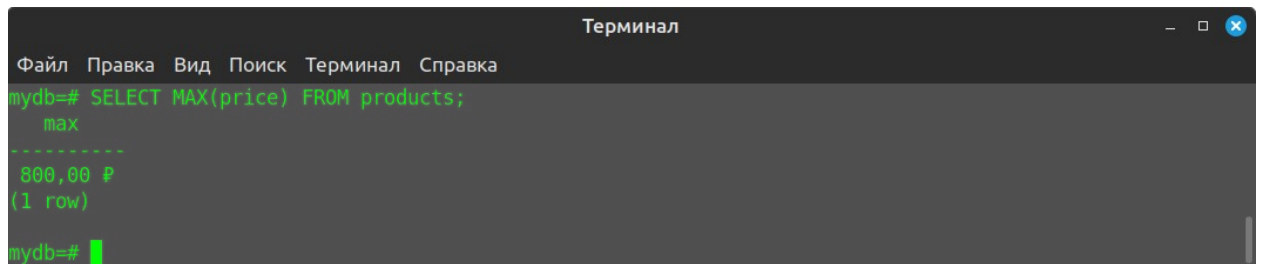


```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT * FROM products;
 article | name          | price
-----+-----+-----
 A000000001 | Помидоры      | 300,00 ₺
 A000000002 | Огурцы        | 150,00 ₺
 A000000003 | Картофель     | 85,00 ₺
 A000000004 | Сыр адыгейский | 660,00 ₺
 A000000005 | Сыр Гауда     | 800,00 ₺
 A000000006 | Молоко 3,2%   | 132,00 ₺
 A000000007 | Молоко 2,5%   | 110,00 ₺
 A000000008 | Свинина окорок | 380,00 ₺
 A000000009 | Свинина шейка | 460,00 ₺
 A000000010 | Батон нарезной | 46,20 ₺
(10 rows)
mydb=#
```



2. Запрос, демонстрирующий возможности функций MAX и MIN.  
Запрос, выбирающий наибольшую стоимость продукта.

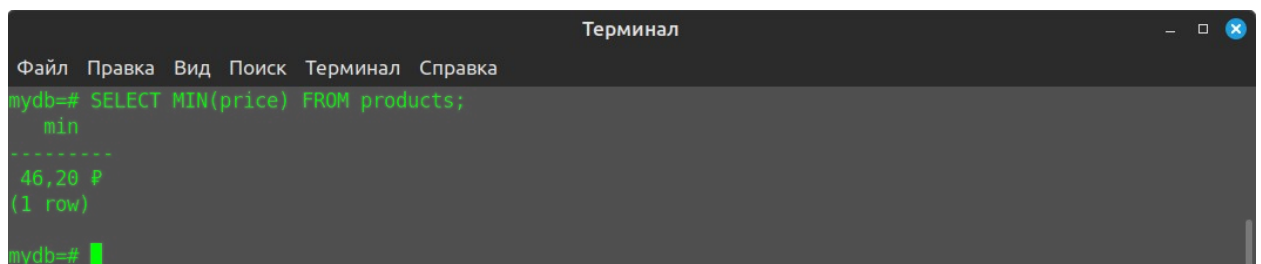
```
SELECT MAX(price) FROM products;
```



```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT MAX(price) FROM products;
      max
-----
 800,00 ₺
(1 row)
mydb=#
```

- Запрос, выбирающий наименьшую стоимость продукта.

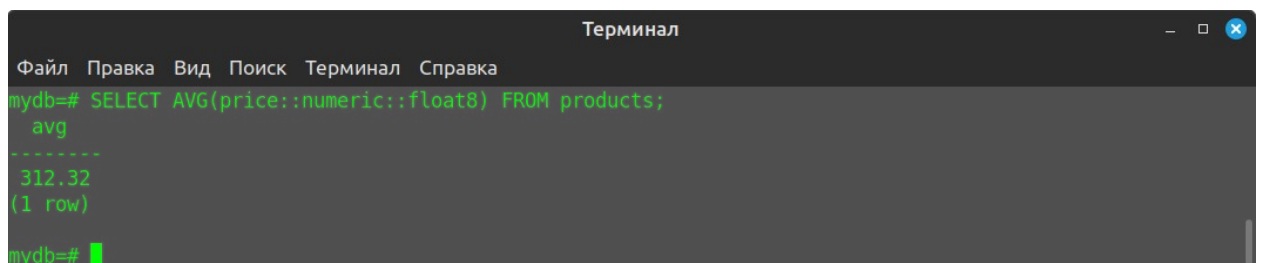
```
SELECT MIN(price) FROM products;
```



```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT MIN(price) FROM products;
      min
-----
 46,20 ₺
(1 row)
mydb=#
```

3. Запрос, демонстрирующий возможности функций AVG и SUM.  
Запрос, выбирающий среднюю стоимость продуктов.

```
SELECT AVG(price::numeric::float8) FROM products;
```



```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT AVG(price::numeric::float8) FROM products;
      avg
-----
 312.32
(1 row)
mydb=#
```

Запрос, выбирающий общую сумму заказов для каждого клиента, сделавшего заказы.

```
SELECT clients.name AS client,
       SUM(orders.count * products.price) AS price
FROM orders
LEFT JOIN clients ON orders.client_id = clients.id
LEFT JOIN products ON orders.articul = products.articul
GROUP BY client;
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT clients.name AS client, SUM(orders.count * products.price) AS price
FROM orders
LEFT JOIN clients ON orders.client_id = clients.id
LEFT JOIN products ON orders.articul = products.articul
GROUP BY client;
  client      | price
-----+-----
Евгений Миронов | 800,00 P
Сергей Пускепалис | 542,40 P
Кристина Асмус | 616,00 P
Виктор Хориняк | 2 140,00 P
Аглая Тарасова | 365,00 P
(5 rows)

mydb=#
```

4. Запрос, демонстрирующий работу конструкций COUNT, COUNT(\*) и COUNT(DISTINCT).

Запрос, выбирающий количество заказов клиента с id = 1.

```
SELECT COUNT(CASE WHEN client_id=1 THEN 1 END) AS
order_count
FROM orders;
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT COUNT(CASE WHEN client_id=1 THEN 1 END) AS order_count
FROM orders;
 order_count
-----
          3
(1 row)

mydb=#
```

Запрос, выбирающий общее количество сделанных заказов всех клиентов.

```
SELECT COUNT(*) AS order_count
FROM orders;
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT COUNT(*) AS order_count
FROM orders;
 order_count
-----
         10
(1 row)

mydb=#
```

Запрос, выбирающий количество клиентов, сделавших заказы.

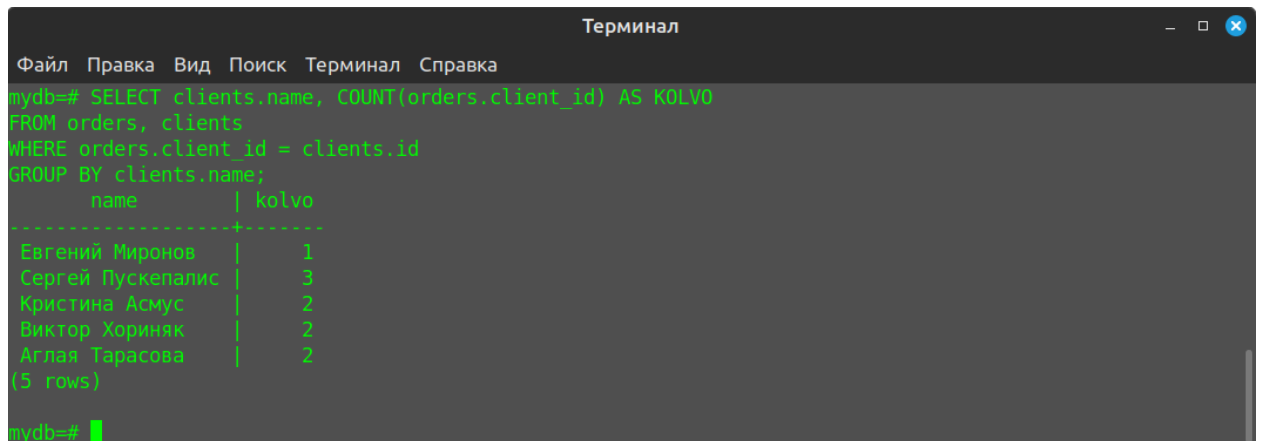
```
SELECT COUNT(DISTINCT client_id) AS client_count
FROM orders;
```

```
Терминал
Файл Правка Вид Поиск Терминал Справка
mydb=# SELECT COUNT(DISTINCT client_id) AS client_count
FROM orders;
 client_count
-----
          5
(1 row)

mydb=#
```

5. Запрос, демонстрирующий работу конструкции GROUP BY.  
Запрос, выбирающий имена клиентов и количество сделанных ими заказов.

```
SELECT clients.name, COUNT(orders.client_id) AS KOLVO
FROM orders, clients
WHERE orders.client_id = clients.id
GROUP BY clients.name;
```



The screenshot shows a terminal window titled "Терминал" with a menu bar containing "Файл", "Правка", "Вид", "Поиск", "Терминал", and "Справка". The terminal displays the following SQL query and its output:

```
mydb=# SELECT clients.name, COUNT(orders.client_id) AS KOLVO
FROM orders, clients
WHERE orders.client_id = clients.id
GROUP BY clients.name;
```

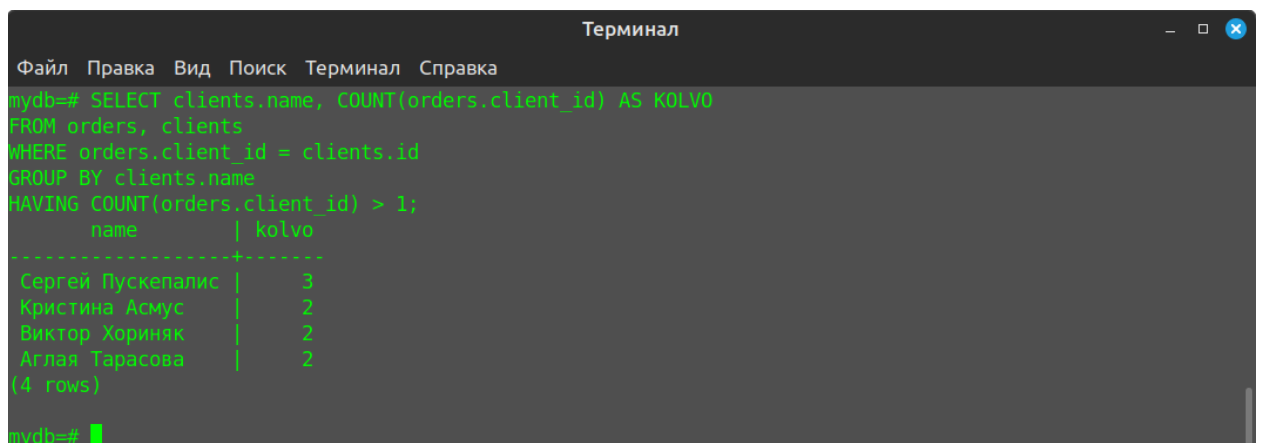
name	kolvo
Евгений Миронов	1
Сергей Пускепалис	3
Кристина Асмус	2
Виктор Хориняк	2
Аглая Тарасова	2

(5 rows)

mydb=#

6. Запрос, демонстрирующий работу конструкции HAVING.  
Запрос, выбирающий имена клиентов и количество сделанных ими заказов, если заказов больше одного.

```
SELECT clients.name, COUNT(orders.client_id) AS KOLVO
FROM orders, clients
WHERE orders.client_id = clients.id
GROUP BY clients.name
HAVING COUNT(orders.client_id) > 1;
```



The screenshot shows a terminal window titled "Терминал" with a menu bar containing "Файл", "Правка", "Вид", "Поиск", "Терминал", and "Справка". The terminal displays the following SQL query and its output:

```
mydb=# SELECT clients.name, COUNT(orders.client_id) AS KOLVO
FROM orders, clients
WHERE orders.client_id = clients.id
GROUP BY clients.name
HAVING COUNT(orders.client_id) > 1;
```

name	kolvo
Сергей Пускепалис	3
Кристина Асмус	2
Виктор Хориняк	2
Аглая Тарасова	2

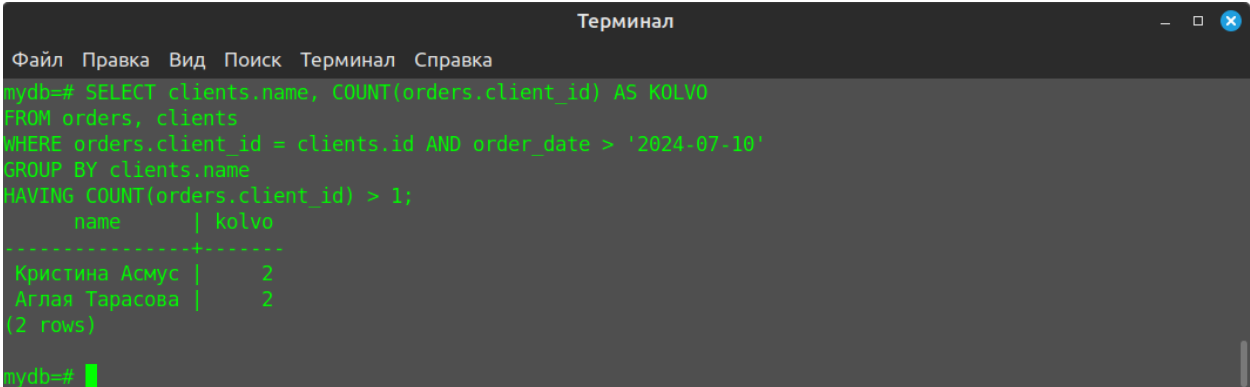
(4 rows)

mydb=#

7. Запрос, демонстрирующий совместную работу конструкций HAVING и WHERE.

Запрос, выбирающий имена клиентов и количество сделанных ими заказов позднее 10 июля 2024 года, если заказов больше одного.

```
SELECT clients.name, COUNT(orders.client_id) AS KOLVO
FROM orders, clients
WHERE orders.client_id = clients.id
AND order_date > '2024-07-10'
GROUP BY clients.name
HAVING COUNT(orders.client_id) > 1;
```



The screenshot shows a terminal window titled "Терминал" with a menu bar containing "Файл", "Правка", "Вид", "Поиск", "Терминал", and "Справка". The terminal displays the following SQL query and its output:

```
mydb=# SELECT clients.name, COUNT(orders.client_id) AS KOLVO
FROM orders, clients
WHERE orders.client_id = clients.id AND order_date > '2024-07-10'
GROUP BY clients.name
HAVING COUNT(orders.client_id) > 1;
 name      | kolvo
-----+-----
Кристина Асмус |      2
Аглая Тарасова |      2
(2 rows)
```

The prompt "mydb=#" is visible at the bottom of the terminal.

## Ответы на контрольные вопросы.

1. Определить количество и общую продолжительность разговоров для каждого региона, с городами которого осуществляли телефонную связь абоненты, чьи фамилии содержат слог «ва».

```
SELECT
    Город.Регион,
    Count(Разговор.Код_Разговор) AS Кол_разговоров,
    Sum(Разговор.Продолжительность) AS
Общ_Продолжительность
FROM Город
INNER JOIN Разговор ON Город.Код_Города =
Разговор.Код_Города
WHERE Разговор.Фамилия LIKE "%ва%" GROUP BY
Город.Регион;
```

2. На какую сумму были проданы билеты на рейс до Москвы в день вылета?

```
SELECT
    Sum(Билет.Стоимость) AS общ_Стоимость,
    Билет.Дата_продажи
FROM Рейс
INNER JOIN Билет ON Рейс.Номер_рейса =
Билет.Номер_рейса
WHERE Билет.Дата_продажи=Рейс.Дата_вылета
AND Рейс.Конечный_пункт='Москва'
GROUP BY Билет.Дата_продажи;
```

3. В каком количестве и на какую сумму издавал свои книги автор Борисов в каждом издательстве?

```
SELECT
    Sum(Книга.Количество) AS Общее_Количество,
    Sum(Книга.Цена*Книга.Количество) AS общ_Стоимость,
    Книга.Издательство
FROM Книга
INNER JOIN Автор ON Книга.Код_Автора = Автор.Код_Автора
GROUP BY Книга.Издательство, Автор.Фамилия
HAVING Автор.Фамилия='Борисов';
```

4. Даны таблицы Город и Разговор.

```
CREATE TABLE Город
(Код_Города INT, Название VARCHAR(20) NOT NULL,
```

Тариф MONEY,  
Регион VARCHAR(20))

и

```
CREATE TABLE Разговор
(Код_Разговор INT ,
Код_Города INTNOTNULL,
Фамилия VARCHAR(20,
Дата DATETIME NOT NULL,
Продолжительность INT NOT NULL))
```

Рассчитать стоимость каждого телефонного разговора с Москвой.

```
SELECT
    Разговор.Фамилия,
    Город.Название,
    (Разговор.Продолжительность *   Город.Тариф)   AS
Стоимость
FROM Разговор, Город
WHERE Город.Код_Города = 'Москва';
```

5. Даны таблицы Автор и Книга.

```
CREATE TABLE Автор
(Код_Автора INT ,
Название VARCHAR(50) NOT NULL)
```

и

```
CREATE TABLE Книга
(Код_Книги INT ,
Название VARCHAR(50) NOT NULL,
Цена MONEY,
Издательство VARCHAR(50) NOT NULL,
Код_Автора INT NOT NULL,
Количество INT)
```

Книги каких авторов были проданы на сумму, превышающую 10000 руб.

```
SELECT
    Автор.Название,
    SUM(Книга.Цена * Книга.Количество) AS SUMMA
FROM Книга
INNER JOIN Автор ON Книга.Код_Автора = Автор.Код_Автора
GROUP BY Автор.Название
HAVING SUM(Книга.Цена * Книга.Количество) > 10000;
```

6. Даны таблицы:

```
CREATE TABLE Город  
(Код_Города INT ,  
Название VARCHAR(20) NOT NULL,  
Тариф MONEY,  
Название VARCHAR(20)).
```

и

```
CREATE TABLE Разговор  
(Код_Разговора INT ,  
Код_Города INT NOT NULL,  
Фамилия VARCHAR(20),  
Дата DATETIME NOT NULL,  
Продолжительность INT NOT NULL).
```

Определить фамилии абонентов, общее время разговоров которых менее 10 мин, а общая стоимость оказалась больше 100 руб.

```
SELECT  
    Разговор.Фамилия,  
    SUM(Разговор.Продолжительность) AS  
ОбщПродолжительность,  
    SUM(Город.Тариф*Разговор.Продолжительность) AS  
Стоимость  
FROM Город  
INNER JOIN Разговор ON Город.Код_Города =  
Разговор.Код_Города  
GROUP BY Разговор.Фамилия  
HAVING SUM(Разговор.Продолжительность) < 10  
AND SUM(Город.Тариф*Разговор.Продолжительность) > 100;
```