Билет 7.

Теория. Вопрос 15. Экосистема Big Data. Распределенная файловая система Hadoop (HDFS). Инструменты работы с большими данными: Hive и HiveQL.

Дополнительные источники:

- Github по дисциплине - теория: https://github.com/BosenkoTM/Big-Data-Storage-and-Processing-Tools/tree/main

1. Экосистема Big Data.

Экосистема Big Data представляет собой множество технологий, инструментов и платформ, разработанных для обработки, хранения, анализа и визуализации огромных объемов данных.

Big Data — это крупные массивы разнообразной информации и стек специальных технологий, инструментов и платформ, разработанных для её обработки, хранения, анализа и визуализации. Термин применяется к таким объемам данных, с которыми пользовательский компьютер и офисные программы не справятся.

Существует шесть основных критериев (6V), которые помогут определить Big Data:

- Volume (объем) — информации должно поступать более 150 Гб в сутки.
- Velocity (скорость) — для работы с массивами информации в режиме реального времени требуются повышенные вычислительные мощности.
- Variety (разнообразие) — поступающая информация имеет разные форматы или степень структурированности.
- Veracity (достоверность) — источникам данных можно доверять, а результат их обработки обладает достоверностью, достаточной для принятия решений.
- Variability (вариативность) — поток данных изменчив, на него может влиять даже время суток или погода.
- Value (ценность) — данные могут иметь разное значение для компании.

Возникла необходимость придумать новые типы хранилищ данных, поскольку стандартных уже не хватало. Первой платформой, которая взяла на себя работу с такими объемами данных, стала Hadoop. К настоящему времени она обладает мощным стеком инструментов. Вот некоторые из основных компонентов этой экосистемы:

- Apache Hadoop: Фреймворк для распределенного хранения и обработки структурированных и неструктурированных данных через кластеры компьютеров.
- Apache Spark: Система обработки данных в памяти, которая обеспечивает быструю обработку данных в пакетном и потоковом режимах.
- Apache Hive: Инфраструктура для анализа, запросов и обработки данных в распределенных хранилищах, таких как Hadoop.
- Apache HBase: Распределенная система управления базами данных NoSQL, обеспечивающая быстрый доступ к большим объемам данных.
- Apache Kafka: Платформа для обработки данных в реальном времени и передачи данных между приложениями.
- Apache Flink: Фреймворк для обработки данных в потоковом режиме с низкой задержкой, поддерживающий сложные вычисления.

Эти и другие инструменты объединены для создания мощной и гибкой инфраструктуры обработки и анализа больших данных.

2. Распределенная файловая система Hadoop (HDFS).

Распределенная файловая система Hadoop (Hadoop Distributed File System – HDFS) – это распределенная файловая система, предназначенная для хранения и обработки больших объемов данных на кластерах вычислительных узлов.

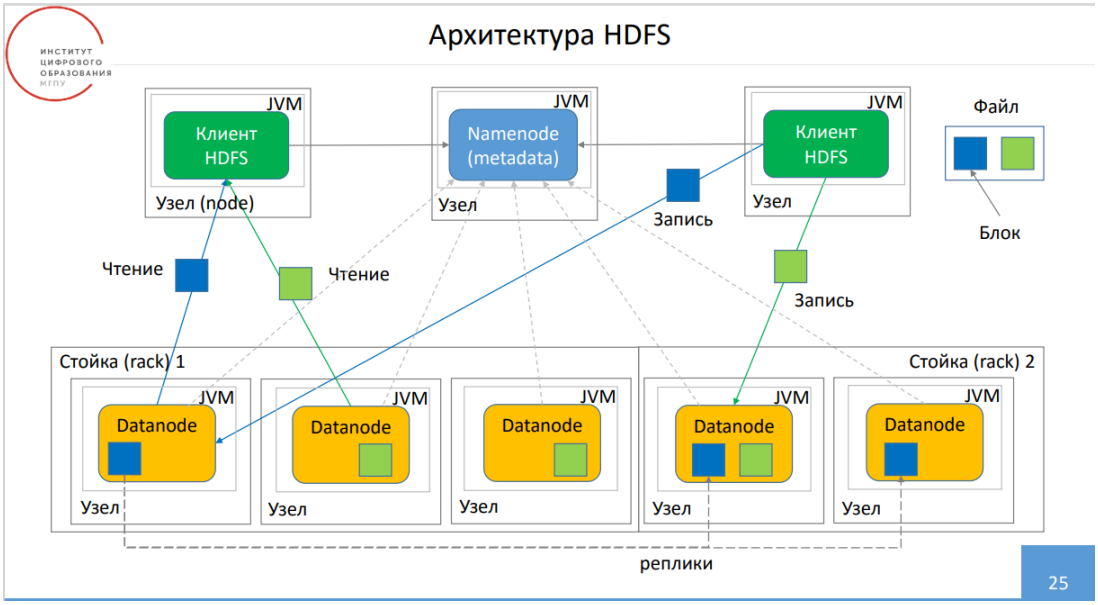
HDFS была разработана как часть проекта Apache Hadoop, который представляет собой платформу с открытым исходным кодом, написанную на Java и позволяющую распределять данные для анализа по кластеру компьютеров. *Кластер* — группа компьютеров, которые представляют собой аппаратный ресурс, выполняющий работу как единое целое. То есть задача одновременно выполняется на всех из них, что ускоряет её решение.

HDFS разделяет данные на множество блоков и хранит их на серверах в кластере. Блок — это кусочек файла стандартного размера. Во второй версии HDFS, которая сейчас считается основной, это 128 МБ. Соответственно, исходный файл разделяется на блоки по 128 МБ, которых может быть десять, а несколько тысяч. Каждый блок с информацией реплицируется на несколько узлов для обеспечения отказоустойчивости.

Архитектура HDFS включает в себя несколько ключевых компонентов, каждый из которых выполняет свою уникальную роль в распределенном хранении данных.

1. Клиент (HDFS Client). Клиенты представляют собой приложения, которые взаимодействуют с HDFS для чтения, записи и обработки данных. Они отправляют запросы на операции с файлами HDFS, такие как чтение, запись, создание или удаление файлов.
2. Главный узел (NameNode). Центральный компонент HDFS, отвечающий за управление метаданными файловой системы. Хранит информацию о том, где расположены блоки данных, и предоставляет эту информацию клиентам. NameNode не хранит фактические данные, а только метаданные, что делает его критически важным компонентом для целостности файловой системы. Его отказ может привести к недоступности файловой системы, поэтому он является одним из узких мест в архитектуре HDFS.
3. Вторичный главный узел (Secondary NameNode) Выполняет задачу резервного копирования метаданных от NameNode. Несмотря на название, Secondary NameNode не является резервной NameNode; он предназначен для создания периодических снимков метаданных и их объединения с текущим состоянием, чтобы уменьшить время восстановления в случае сбоя. Помогает предотвратить потерю данных и ускоряет процесс восстановления NameNode в случае сбоя.
4. Узел данных (DataNode). Datanodes – узлы, на которых хранятся фактические данные. Они отвечают за чтение и запись данных в файлы, а также передачу блоков данных NameNode для регистрации метаданных. HDFS использует репликацию для обеспечения отказоустойчивости данных, и DataNode отвечают за создание и управление репликами блоков данных.

Вместе эти компоненты обеспечивают эффективное и отказоустойчивое хранение данных в распределенной среде, что делает HDFS подходящей для обработки больших объемов данных на кластерах серверов.



Преимущества:

- Распределённое хранение. HDFS разбивает файлы на небольшие блоки и хранит их на разных узлах в кластере серверов. Это равномерно распределяет нагрузку на кластер и позволяет ускорить работу с данными за счёт одновременной обработки сотен и тысяч файловых блоков.
- Репликация данных. Каждый блок данных в HDFS дублируется на несколько узлов. Если один узел выходит из строя, информация может быть восстановлена из других.
- Работа в формате потока данных. Обработка данных может идти в режиме реального времени в процессе их получения, что ускоряет работу. Серверу не нужно ждать, пока поступление данных закончится.
- Простота обслуживания и устойчивость. Благодаря репликации и системе сообщений узлов данных HDFS автоматически обнаруживает сбои и восстанавливает данные из реплицированных узлов.
- Масштабируемость. HDFS легко масштабируется по горизонтали. Если объём данных или нагрузка увеличиваются, то можно просто добавить больше серверов в вычислительный кластер. Система будет автоматически использовать их для хранения и обработки данных.
- Поддержка различных типов данных. HDFS поддерживает хранение разнообразных данных — структурированных (таблицы), полуструктурированных (JSON, XML) и неструктурированных (видео и изображения).
- Интеграция с экосистемой Hadoop. HDFS плотно интегрирована с другими компонентами экосистемы Hadoop — Apache Spark, Apache Hive, Apache Pig и другими. Вместе они обеспечивают полный цикл обработки данных — хранение, распределение, загрузку, анализ, визуализацию и прочие способы представления данных.

Недостатки:

- Низкая эффективность работы с файлами меньше размера одного стандартного блока — 128 МБ. Работа с ними приведёт к замедлению работы из-за многократного повышения нагрузки на NameNode, хранящего пространство имён в HDFS.
- Работа системы полностью зависит от главного узла. Если по какой-либо причине он перестанет работать, то вся HDFS выйдет из строя. Восстановить его из вторичного главного узла невозможно.
- Низкая безопасность данных, так как при получении доступа к главному узлу можно получить доступ ко всей хранящейся в файловой системе информации.

3. Инструменты работы с большими данными: Hive и HiveQL.

Hive – это система управления базами данных в рамках платформы Hadoop с SQL-подобным языком запросов, позволяет выполнять запросы, агрегировать и анализировать данные; компонент экосистемы Hadoop. Работает напрямую с HDFS, поддерживает основные форматы Hadoop. Запросы могут выполняться через Tez, Spark или Hadoop MapReduce.

Язык запросов — HiveQL — приближен к SQL, при этом не реализует все возможности стандарта SQL-92. В язык встроены функции для работы с форматами XML и JSON, поддержка нескалярных типов данных, таких как массивы, структуры, ассоциативные массивы, реализован достаточно широкий набор агрегатных функций, поддерживаются определяемые пользователем функции, блокировки.

**MapReduce* – это модель распределённых вычислений от компании Google, используемая в технологиях Big Data для параллельных вычислений над очень большими (до нескольких петабайт) наборами данных в компьютерных кластерах, и фреймворк для вычисления распределенных задач на узлах (node) кластера.

Ключевыми преимуществами Apache Hive являются следующие:

- масштабируемость — динамическое расширение при добавлении машины к кластеру Hadoop;
- расширяемость за счет MapReduce и определяемых пользователем функций (UDF/UDAF/UDTF);
- отказоустойчивость благодаря сохранению всех промежуточных результатов;
- поддержка разных форматов данных — TEXTFILE, Sequence, ORC, RCFILE, а также Parquet (с помощью плагина в версиях позже 0.10).

В Apache Hive запросы к данным, хранящимся в Hadoop, реализуются на SQL-подобном декларативном языке Hive Query Language (HiveQL), который является подмножеством SQL92. Однако, в ряде случаев HiveQL отличается от стандартного SQL, в частности:

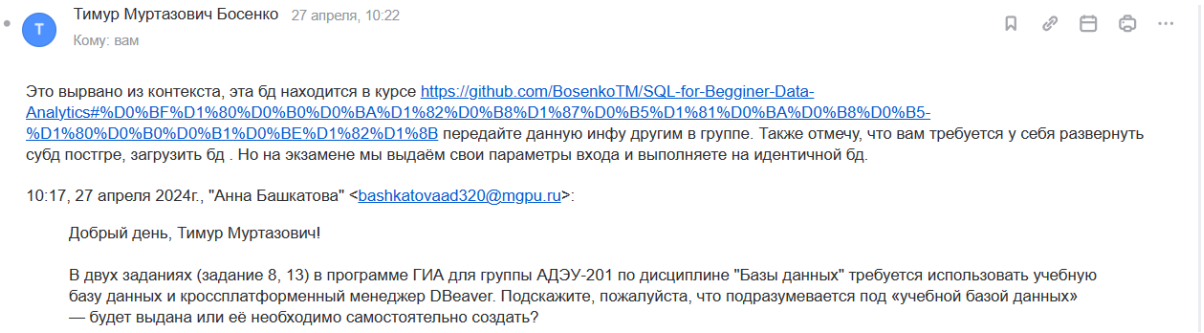
- разные способы определения операций join для максимальной производительности;
- в HiveQL нет некоторых функций, операций и операторов SQL (UPDATE и DELETE statements, INSERT для отдельных строк);
- HiveQL позволяет вставлять пользовательский код для ситуаций, которые не вписываются в типовой SQL, предоставляя соответствующие инструменты для обработки входа и выхода – определенные пользователем функции: User Defined Function (UDF), User Defined Aggregate Function (UDAF), User Defined Tabular Function (UDTF);
- HiveQL не поддерживает типы данных даты и времени, т. к. они рассматриваются как строки.

Пример сеанса работы с Hive с применением HiveQL — удаление таблицы, создание таблицы, загрузка в неё данных из текстового файла и запрос для подсчёта, сколько раз каждое слово встречалось в файле:

```
DROP TABLE IF EXISTS docs;
CREATE TABLE docs (line STRING);
LOAD DATA INPATH 'input_file' OVERWRITE INTO TABLE docs;
CREATE TABLE word_counts AS
SELECT word, count(1) AS count FROM
 (SELECT explode(split(line, '\s')) AS word FROM docs) temp
GROUP BY word
ORDER BY word;
```

Практика. Задание 13. Используя учебную базу данных и кроссплатформенный менеджер DBeaver, выполните запрос, который позволит отобрать продукты, которые продавались партиями более 65 шт. Выведите id продуктов, их наименование и количество проданных продуктов.

Уточнение по заданию от Босенко:



Тимур Муртазович Босенко 27 апреля, 10:22
Кому: вам

Это вырвано из контекста, эта бд находится в курсе https://github.com/BosenkoTM/SQL-for-Begginer-Data-Analytics#%D0%BF%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B передайте данную инфу другим в группе. Также отмечу, что вам требуется у себя развернуть субд постгре, загрузить бд . Но на экзамене мы выдаём свои параметры входа и выполняете на идентичной бд.

10:17, 27 апреля 2024г., "Анна Башкатова" <bashkatovaad320@mgpu.ru>:

Добрый день, Тимур Муртазович!

В двух заданиях (задание 8, 13) в программе ГИА для группы АДЭУ-201 по дисциплине "Базы данных" требуется использовать учебную базу данных и кроссплатформенный менеджер DBeaver. Подскажите, пожалуйста, что подразумевается под «учебной базой данных» — будет выдана или её необходимо самостоятельно создать?

Ссылка из скрина: https://github.com/BosenkoTM/SQL-for-Begginer-Data-Analytics#%D0%BF%D1%80%D0%B0%D0%BA%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5-%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D1%8B

Ход действий (работала в VirtualBox, Ubuntu 20.04):

1. Установить СУБД PostgreSQL;
2. Установить Dbeaver;
3. Работа с СУБД через Dbeaver;
4. Запрос.

Установка DBeaver:

```
sudo apt install snapd
sudo snap install dbeaver-ce
sudo apt install postgresql
sudo -u postgres psql
```

Т.к. у пользователя postgres по умолчанию нет пароля, то устанавливаем его самостоятельно:

```
\password
sudo su postgres
createdb sqlda
```

Необходимо скачать data.dump (https://disk.yandex.ru/d/p3ga3WZpmAw8-Q) и зайти в папку с этим файлом (в данном случае “Загрузки”)

```
psql -U postgres -d sqlda < data.dump
```

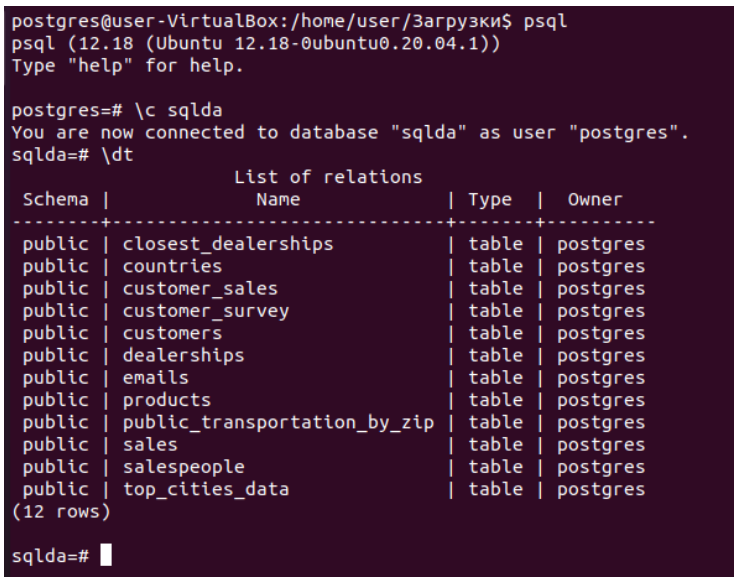
```
user@user-VirtualBox:~$ sudo apt install snapd
[sudo] пароль для user:
Чтение списков пакетов… Готово
Построение дерева зависимостей
Чтение информации о состоянии… Готово
Следующие пакеты будут обновлены:
  snapd
Обновлено 1 пакетов, установлено 0 новых пакетов, для удаления отмечено 0 пакетов, и 299 пакетов не обновлено.
Необходимо скачать 0 B/24,4 MB архивов.
После данной операции объём занятого дискового пространства уменьшится на 70,1 MB.
(Чтение базы данных … на данный момент установлено 183522 файла и каталога.)
Подготовка к распаковке …/snapd_2.61.3+20.04_amd64.deb …
Распаковывается snapd (2.61.3+20.04) на замену (2.58+20.04) …
Настраивается пакет snapd (2.61.3+20.04) …
Устанавливается новая версия файла настройки /etc/apparmor.d/usr.lib.snapd.snap-confine.real …
snapd.failure.service is a disabled or a static unit not running, not starting i
t.
snapd.snap-repair.service is a disabled or a static unit not running, not starti
ng it.
Failed to restart snapd.mounts-pre.target: Operation refused, unit snapd.mounts-
pre.target may be requested by dependency only (it is configured to refuse manua
l start/stop).
See system logs and 'systemctl status snapd.mounts-pre.target' for details.
Обрабатываются триггеры для mime-support (3.64ubuntu1) …
Обрабатываются триггеры для gnome-menus (3.36.0-1ubuntu1) …
Обрабатываются триггеры для man-db (2.9.1-1) …
Обрабатываются триггеры для dbus (1.12.16-2ubuntu2.3) …
Обрабатываются триггеры для desktop-file-utils (0.24-1ubuntu3) …
user@user-VirtualBox:~$ sudo snap install dbeaver-ce
dbeaver-ce 24.0.3.202404211624 от DBeaver (dbeaver-corp) установлен
user@user-VirtualBox:~$
```

```
user@user-VirtualBox:~$ sudo apt install postgresql
Чтение списков пакетов… Готово
Построение дерева зависимостей
Чтение информации о состоянии… Готово
Будут установлены следующие дополнительные пакеты:
  libllvm10 libpq5 postgresql-12 postgresql-client-12 postgresql-client-common
  postgresql-common sysstat
Предлагаемые пакеты:
  postgresql-doc postgresql-doc-12 libjson-perl isag
Следующие НОВЫЕ пакеты будут установлены:
  libllvm10 libpq5 postgresql postgresql-12 postgresql-client-12
  postgresql-client-common postgresql-common sysstat
Обновлено 0 пакетов, установлено 8 новых пакетов, для удаления отмечено 0 пакетов, и 299 пакетов не обновлено.
Необходимо скачать 30,7 MB архивов.
После данной операции объём занятого дискового пространства возрастёт на 122 MB.
Хотите продолжить? [Д/н] y
```

```
user@user-VirtualBox:~$ sudo -u postgres psql
psql (12.18 (Ubuntu 12.18-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# \password
Enter new password for user "postgres":
Enter it again:
postgres=# \q
```

```
user@user-VirtualBox:~$ sudo su postgres
postgres@user-VirtualBox:/home/user$ ls
 Видео   Документы   Загрузки   Изображения   Музыка   Общедоступные  'Рабочий стол'   Шаблоны
postgres@user-VirtualBox:/home/user$ cd Загрузки
postgres@user-VirtualBox:/home/user/Загрузки$ ls
data.dump
postgres@user-VirtualBox:/home/user/Загрузки$ createdb sqlda
postgres@user-VirtualBox:/home/user/Загрузки$ psql
psql (12.18 (Ubuntu 12.18-0ubuntu0.20.04.1))
Type "help" for help.

postgres=# \l
                                 List of databases
   Name    |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
-----------+----------+----------+-------------+-------------+-----------------------
 postgres  | postgres | UTF8     | ru_RU.UTF-8 | ru_RU.UTF-8 |
 sqlda     | postgres | UTF8     | ru_RU.UTF-8 | ru_RU.UTF-8 |
 template0 | postgres | UTF8     | ru_RU.UTF-8 | ru_RU.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | ru_RU.UTF-8 | ru_RU.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
(4 rows)

postgres=# \q
```

```
postgres@user-VirtualBox:/home/user/Загрузки$ psql -U postgres -d sqlda < data.dump
SET
SET
SET
SET
SET
 set_config
------------

(1 row)

SET
SET
SET
SET
CREATE EXTENSION
```



Основные команды при работе в psql:

- \l – список баз данных
- \dt – список таблиц
- \q (или Ctrl+D) – выход с программы

Дополнительно можно ознакомиться с другими: https://bookflow.ru/shpargalka-po-postgresql/?ysclid=lvy01mm7xp848517239

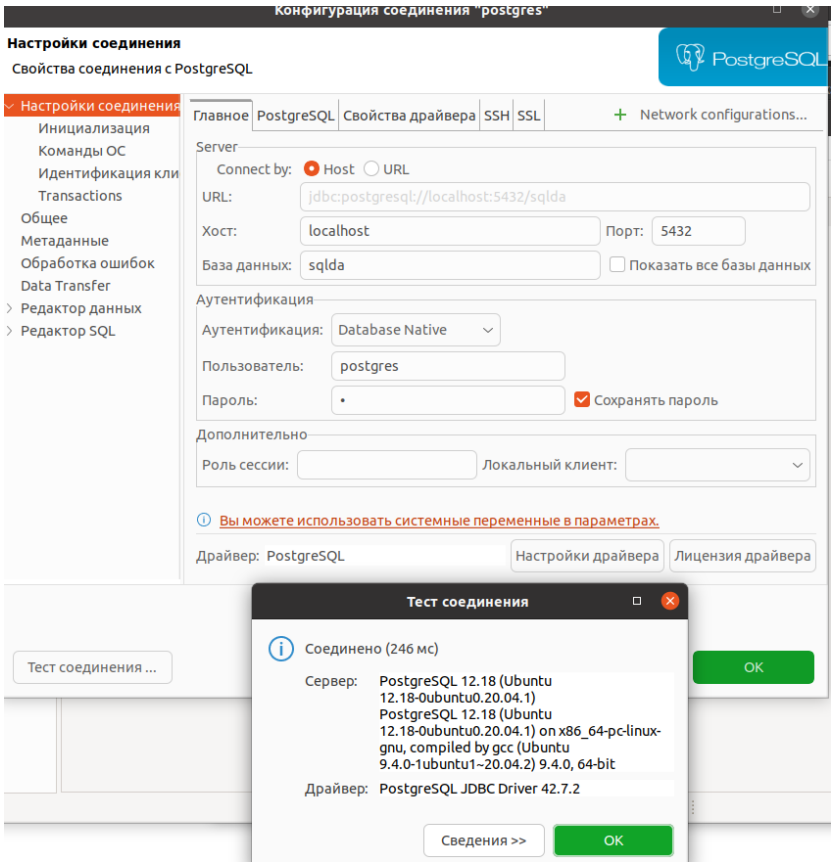
Инструкция по загрузке data.dump от Босенко: https://docs.yandex.ru/docs/view?url=ya-disk-public%3A%2F%2FPJ2TQIxrfbfvbjunimoajsIMV3Hw%2FcT3PQzU4LcrD2WmSBZZQ6O3cOd1PUwvA2poq%2FJ6bpmRyOJonT3VoXnDag%3D%3D%3A%2FLoading_the_sample_datasets_instructions.pdf&name=Loading_the_sample_datasets_instructions.pdf&nosw=1

Работа в DBeaver:

Создаем соединение с PostgreSQL:

БД – та, что указывали в createdb, то есть sqlda

Пароль – тот, что задавали для пользователя postgres

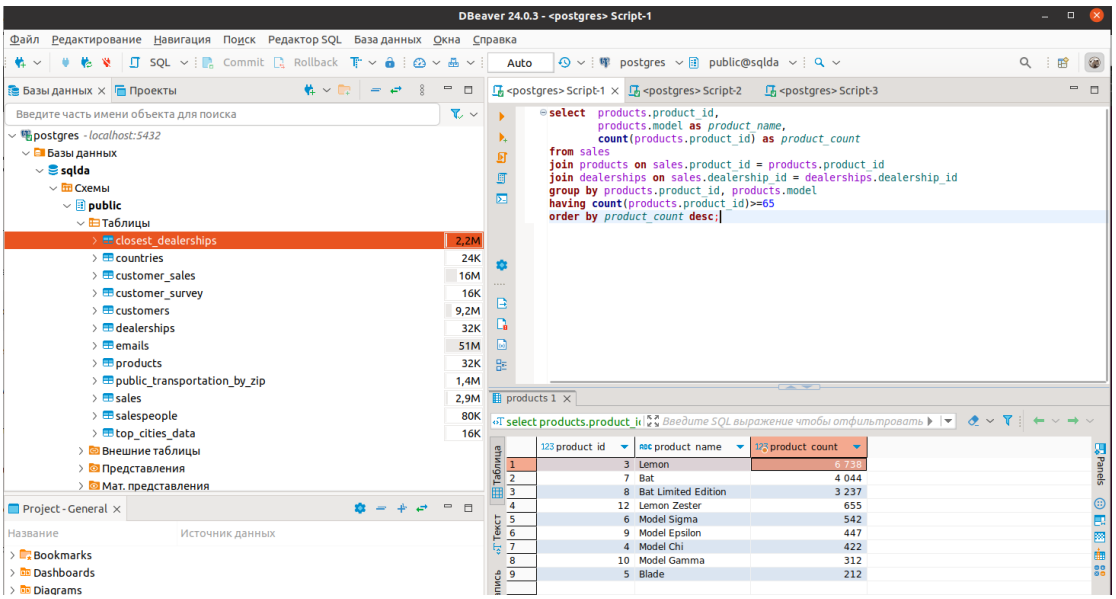


Далее проверяем соединение.

По заданию необходимо выполните запрос, который позволит отобрать продукты, которые продавались партиями более 65 шт. Выведите id продуктов, их наименование и количество проданных продуктов.

В решении этого задания партия рассмотрена как однородная продукция, которая продается конкретными dealerships.

```
SELECT products.product_id, products.model AS product_name, COUNT(sales.product_id) AS product_count

FROM sales

JOIN products ON sales.product_id = products.product_id

JOIN dealerships ON sales.dealership_id = dealerships.dealership_id

GROUP BY products.product_id, products.model

HAVING COUNT(products.product_id) >= 65

ORDER BY product_count DESC;
```


DBeaver 24.0.3 - <postgres> Script-1
Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка
Базы данных
Проекты
Введите часть имени объекта для поиска
postgres - localhost:5432
Базы данных
sqlda
Схемы
public
Таблицы
closest_dealerships 2,2M
countries 24K
customer_sales 16M
customer_survey 16K
customers 9,2M
dealerships 32K
emails 51M
products 32K
public_transportation_by_zip 1,4M
sales 2,9M
salespeople 80K
top_cities_data 16K
Внешние таблицы
Представления
Мат. представления
Project - General
Название
Источник данных
Bookmarks
Dashboards
Diagrams
<postgres> Script-1
<postgres> Script-2
<postgres> Script-3
select products.product_id,
products.model as product_name,
count(products.product_id) as product_count
from sales
join products on sales.product_id = products.product_id
join dealerships on sales.dealership_id = dealerships.dealership_id
group by products.product_id, products.model
having count(products.product_id)>=65
order by product_count desc;
products 1
product_id
product_name
product_count
1 3 Lemon 6 738
2 7 Bat 4 044
3 8 Bat Limited Edition 3 237
4 12 Lemon Zester 655
5 6 Model Sigma 542
6 9 Model Epsilon 447
7 4 Model Chi 422
8 10 Model Gamma 312
9 5 Blade 212