МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра радиотехники и связи

**Отчет по лабораторной работе № 5**

**“ Обработка временных рядов, база данных InfluxDB"**

Выполнил: обучающийся гр. ИСТ – 23

Алексеев П.В.

Проверил: ст. преподаватель каф. РТиС Конкин Н.А.

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Йошкар-Ола

2024

**Цель:**

Исследование применимости специальной базы данных для работы с временными рядами - InfluxDB.

**Задачи:**

# Изучение и установка Docker и InfluxDB.

# Изучение синтаксиса InfluxDB.

# Получение временного хода данных сенсора.

# Загрузка временного хода данных сенсора в БД InfluxDB.

# Визуализация временного ряда загруженных данных сенсора средствами InfluxDB.

# Прогнозирование временного хода данных сенсора.

# Загрузка результов прогнозирования в InfluxDB, анализ результатов.

# Теоретическая часть

Установка InfluxDB через Docker на операционной системе Windows.

Инструкция по установке InfluxDB с использованием Docker:

**Шаг 1: Установка Docker Desktop**

1. Скачайте Docker Desktop: Перейдите на официальный сайт Docker и скачайте установочный файл для Windows.
2. Установите Docker Desktop: Запустите загруженный установщик и следуйте инструкциям на экране. После установки вам может потребоваться перезагрузить компьютер.
3. Запустите Docker Desktop: После перезагрузки запустите Docker Desktop и убедитесь, что он работает. В правом нижнем углу экрана должно отображаться иконка Docker.
4. Создайте учетную запись Docker, принеобходимости.

**Шаг 2: Запуск InfluxDB в Docker**

1. Откройте командную строку (cmd) или PowerShell.
2. Загрузите и запустите контейнер InfluxDB: Введите следующую команду в терминале:

* docker run-d --name=influxdb -p 8086:8086 -v influxdb\_storage:/var/lib/influxdb influxdb

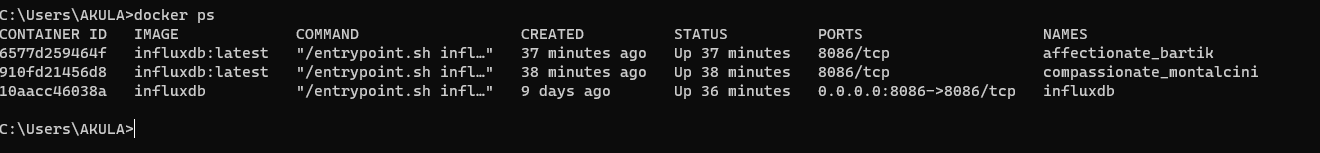
Здесь:

* -d - запускает контейнер в фоновом режиме.
* --name=influxdb - задает имя контейнера.
* -p 8086:8086 - пробрасывает порт 8086 на хост-машину.
* -v influxdb\_storage:/var/lib/influxdb - подключает объем для хранения данных, чтобы данные сохранялись после перезапуска контейнера.
* influxdb - имя образа, который будет загружен из Docker Hub.

1. Проверьте, что контейнер запущен: Введите следующую команду, чтобы увидеть список запущенных контейнеров введите:

docker ps

Убедитесь, что контейнер influxdb отображается в списке.



*Рис. 1*

**Шаг 3: Подключение к InfluxDB**

1. Доступ к InfluxDB: Вы можете подключиться к InfluxDB через командную строку, используя клиент InfluxDB или через HTTP API на http://localhost:8086.

Чтобы получить доступ к интерфейсу командной строки InfluxDB, выполните следующую команду:

* docker exec -it influxdb influx

1. Создание базы данных:

После запуска клиента вы можете создать базу данных с помощью команды:

SQL:

* CREATE DATABASE mydb

Теперь InfluxDB установлен и запущен на вашем Windows! Вы можете взаимодействовать с ним, используя как командную строку, так и HTTP API.

Примечания

Для управления контейнерами вы можете использовать Docker Desktop, который предоставляет графический интерфейс для управления контейнерами и образами.

Если вы хотите остановить контейнер, вы можете выполнить команду:

* docker stop influxdb

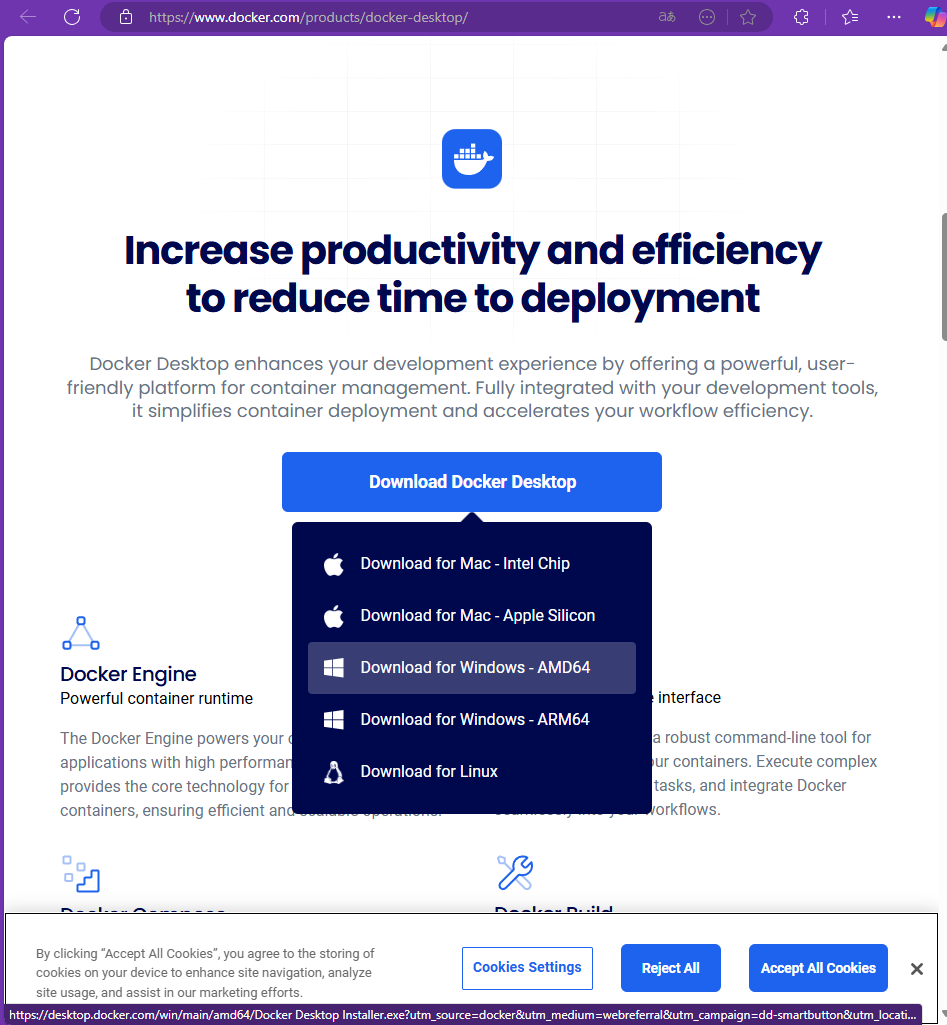
Чтобы удалить контейнер, выполните:

* docker rm influxdb

**Практическая часть**

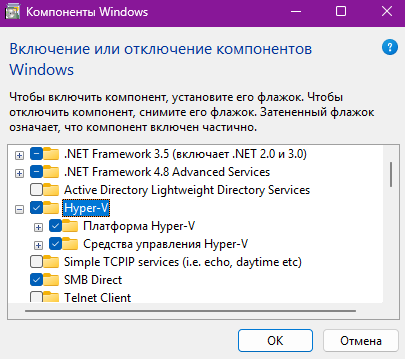
**Шаг 1: Установка Docker Desktop**

1. Убеждаемся, что у нас 11 винда иначе нужно обновиться до 11 версии. В 10 версии нет по умолчанию WSL 2 из-за этого будут трудности.
2. Переходим на сайт выбираем нашу операционную систему и скачиваем наш файл



*Рис. 2*

По умолчанию Докер предлагает установить WSL но чаще всего это не помогает по этому переходим в ‘Панель управления’ > “Удаление программ” > “Включение или отключение компонентов Windows” и находим там Hyper-V



*Рис. 3*

Статья для чего нужен - [Hyper-V](https://winitpro.ru/index.php/2019/08/07/nastrojka-free-windows-hyper-v-server/)

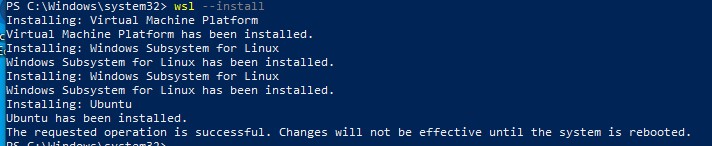
Так -же важно отметить, что у меня AMD процессор и мне нужно включить в настройках BIOS/UEFI компьютера поддержка аппаратной виртуализации: Intel VT (Intel Virtualization Technology) или AMD-V (SVM Mode).

**Теперь перейдём к установке WSL2**

Компонент WSL по умолчанию отключен в Windows. Современных дистрибутивах Windows 10 и Windows 11 для установки среды WSL достаточно выполнить команду:

* wsl –install

Этак команда автоматически включит все необходимые компоненты Windows, необходимые для работы WSL, установить обновление ядра Linux для WSL2, загрузит дистрибутив Ubuntu (по-умолчанию) и установит его в WSL.



*Рис. 4*

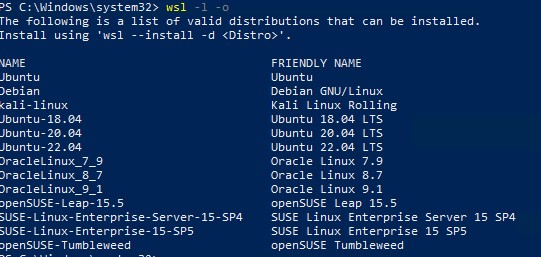
Осталось перезагрузить компьютер, и можем запускать среду WSL!

Мы можем установить для WSL другой дистрибутив Linux. Выведите список доступных дистрибутивов:

* wsl --list --online

Укажем имя дистрибутива Linux, который установим в WSL. Например:

* wsl --install -d kali-linux



*Рис. 5*

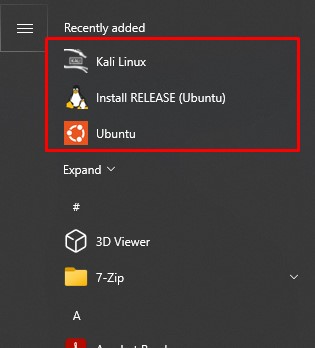
Еще раз перезагрузите компьютер и назначьте версию WSL 2 в качестве среды по-умолчанию:

* wsl --set-default-version 2

wsl2 назначить по умолчанию 

*Рис. 6*

После установки мы можем зайти в меню Пуск и увидеть что появится отдельная программа для запуска Linux.



*Рис. 7*

Чтобы подключиться к нашему дистрибутиву Linux в среде WSL, выполним:

wsl.exe

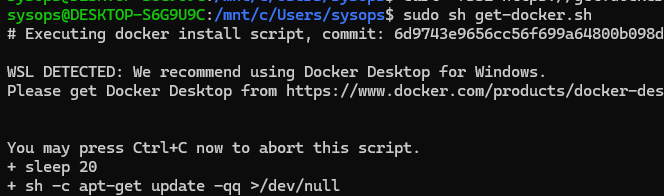
Обновим пакеты в дистрибутиве Linux (Ubuntu в этом случае):

* $ sudo apt-get update && sudo apt-get upgrade -y

Затем выполним установку Docker Engine в Linux с помощью универсального официального скрипта:

* $ curl -fsSL https://get.docker.com -o get-docker.sh
* $ sudo sh get-docker.sh

Игнорируем появившееся предупреждение об установке Docker в среде WSL.

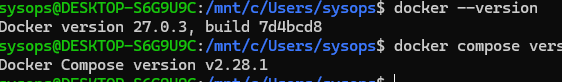
 *Рис. 8*

Добавим пользователя в группу docker чтобы получить возможность запускать команды docker без повышения привилегий и ввода пароля через sudo:

* $ sudo usermod -aG docker $USER

Проверьте, что Docker desktop и compose установлены успешно:

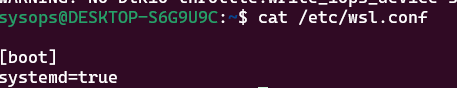
* $ docker –version
* $ docker compose version



*Рис. 9*

Теперь нужно настроить автоматически запуск docker в WSL. В современных дистрибутивах Linux для WSL2 добавлена полноценная поддержка **systemd**. Проверим, что он включен:

* $ cat /etc/wsl.conf

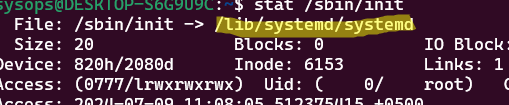


*Рис. 10*

Затем проверим, что systemd используется в качестве системы инициализации Linux:

* $ stat /sbin/init

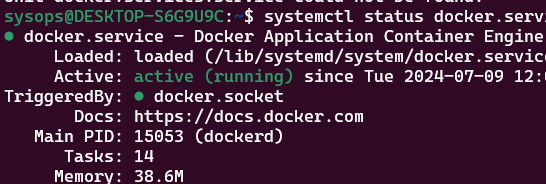
В данном используется systemd, т.к. процесс **/sbin/init** это символьная ссылка на **/lib/systemd/systemd.**



*Рис. 11*

Запустим сервис docker engine и включим автоматический запуск:

* $ sudo systemctl enable --now docker.service
* $ systemctl status docker.service



*Рис. 12*

Если systemd не используется по той или иной причине, можно добавить его в автозагрузку WSL:

* $ nano /etc/wsl.conf

[boot]

command = "/usr/sbin/service docker start"

WSL2 по-умолчанию может использовать следующие ресурсы компьютера:

* 50% оперативной памяти
* 25% файла подкачки (swap)
* 100% ресурсов процессора

Если мы хотим ограничить использование ресурсов хоста подсистемой WSL2, создадим в профиле текущего пользователя текстовый файл %UserProfile%\.wslconfig

.В этом файле можно задать глобальные ограничения для всех дистрибутивов в WSL. Например:

* [wsl2]
* memory=8GB
* processors=4
* swap=2GB

Перезапустим образ WSL из командной строки Windows:

* wsl --shutdown

Попробуем запустить тестовый docker образ в WSL:

* $ docker run hello-world

Docker Engine скачает и запустит демонстрационный контейнер hello-world из Docker Hub.

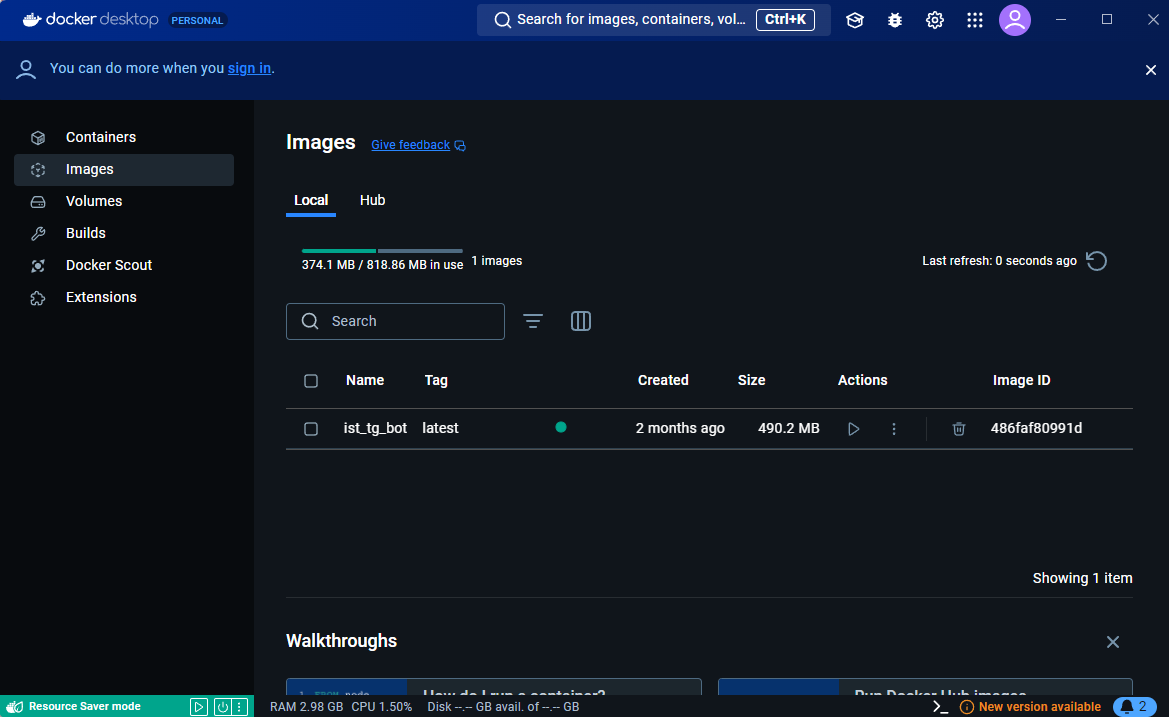
Теперь мы можем выполнять команды к docker в WSL из командной строки Windows. Перед командами docker нужно указывать wsl. Например, вывести список доступных образов в Docker:

* wsl docker images

Управление docker контейнерами из Windows

Теперь мы можем разрабатывать и запускать Docker контейнеры непосредственно на своем Windows компьютере.

Но нам всё это не нужно наш пользователь автоматически запускает докер , по этому запустим наш Docker Desktop и проверим что всё работает.

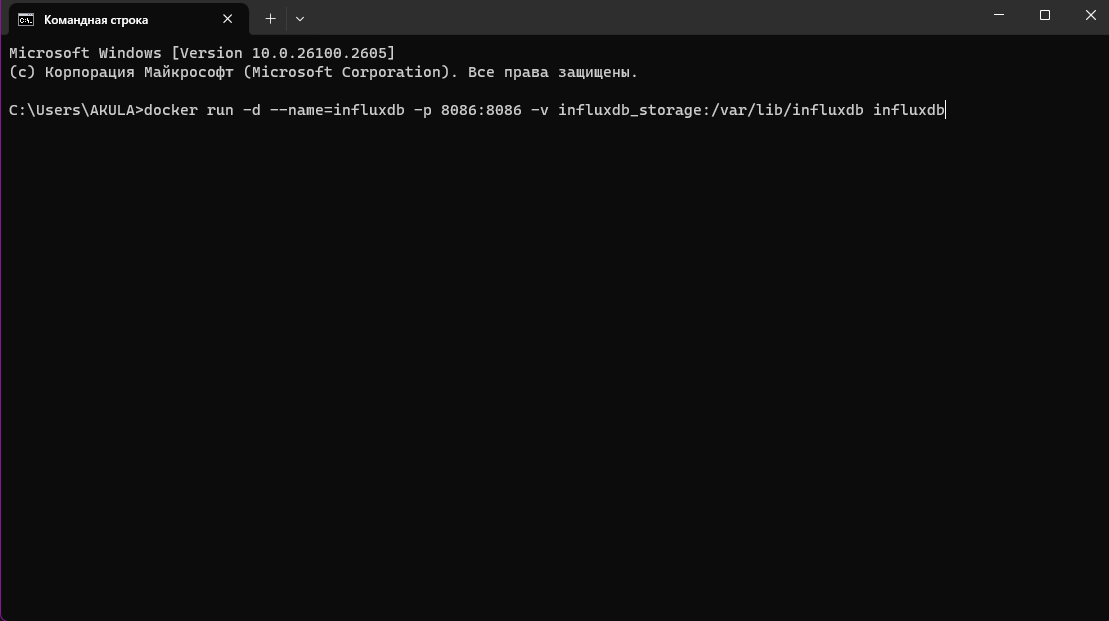


*Рис. 13*

**Шаг 2: Запуск InfluxDB в Docker**

Открываем CMD и водим :

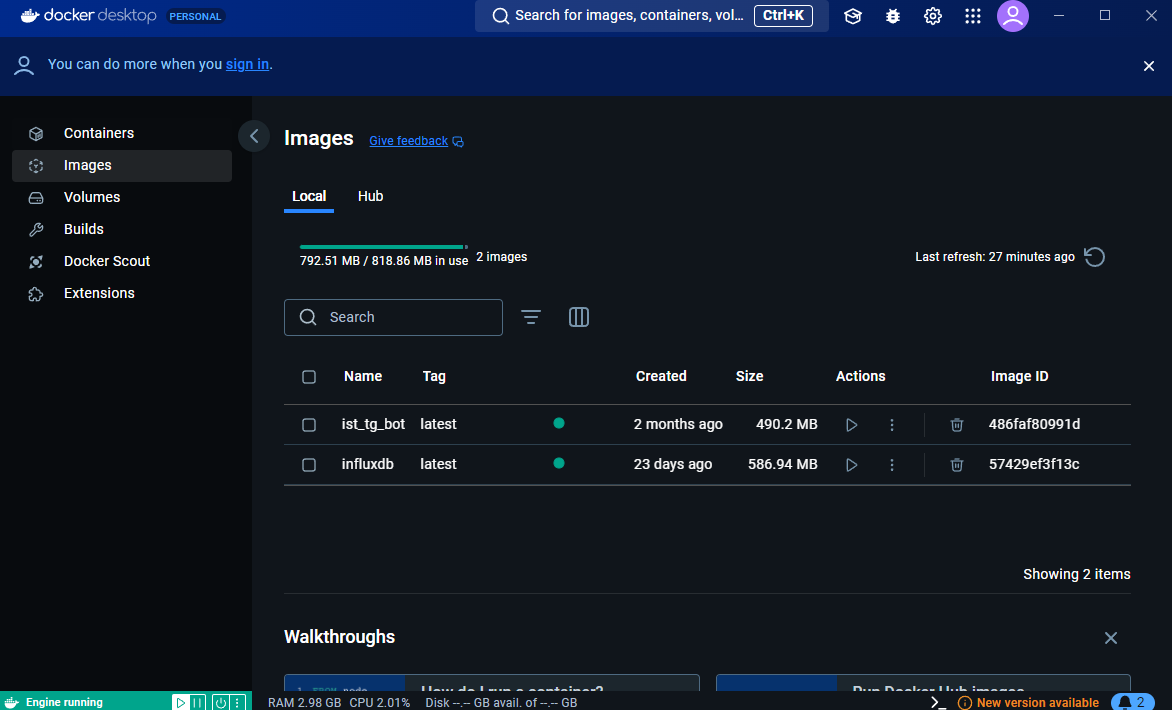
* docker run -d --name=influxdb -p 8086:8086 -v influxdb\_storage:/var/lib/influxdb influxdb



*Рис. 14*

* -d - запускает контейнер в фоновом режиме.
* --name=influxdb - задает имя контейнера.
* -p 8086:8086 - пробрасывает порт 8086 на хост-машину.
* -v influxdb\_storage:/var/lib/influxdb - подключает объем для хранения данных, чтобы данные сохранялись после перезапуска контейнера.
* influxdb - имя образа, который будет загружен из Docker Hub.

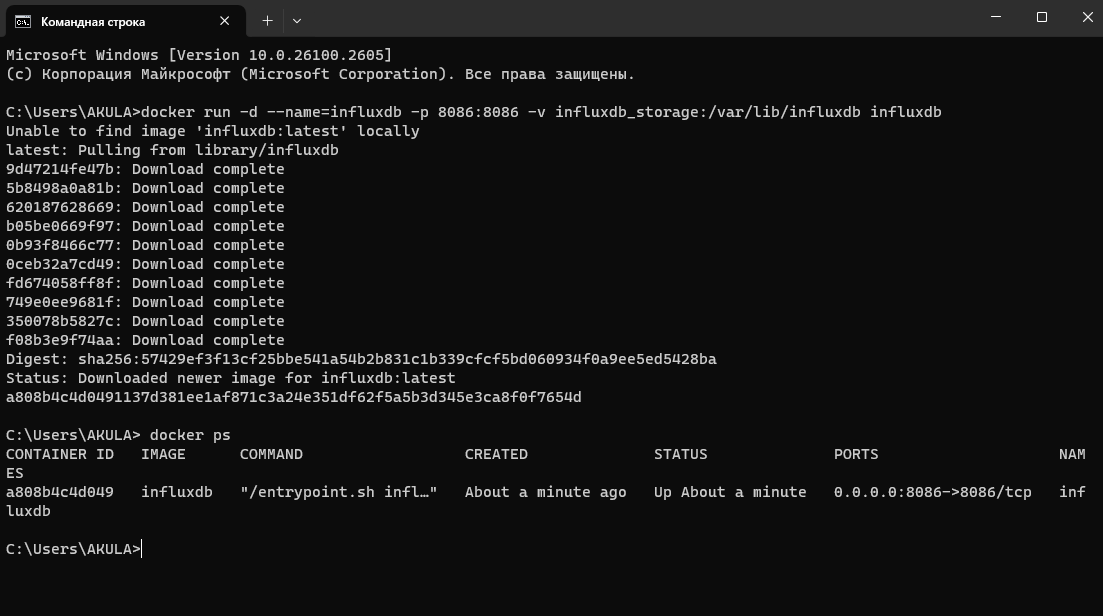
Зайдём в Докер десктоп и увидим, что всё есть



*Рис. 15*

Теперь так-же введём консольную команду и посмотрим писок запущенных контейнеро

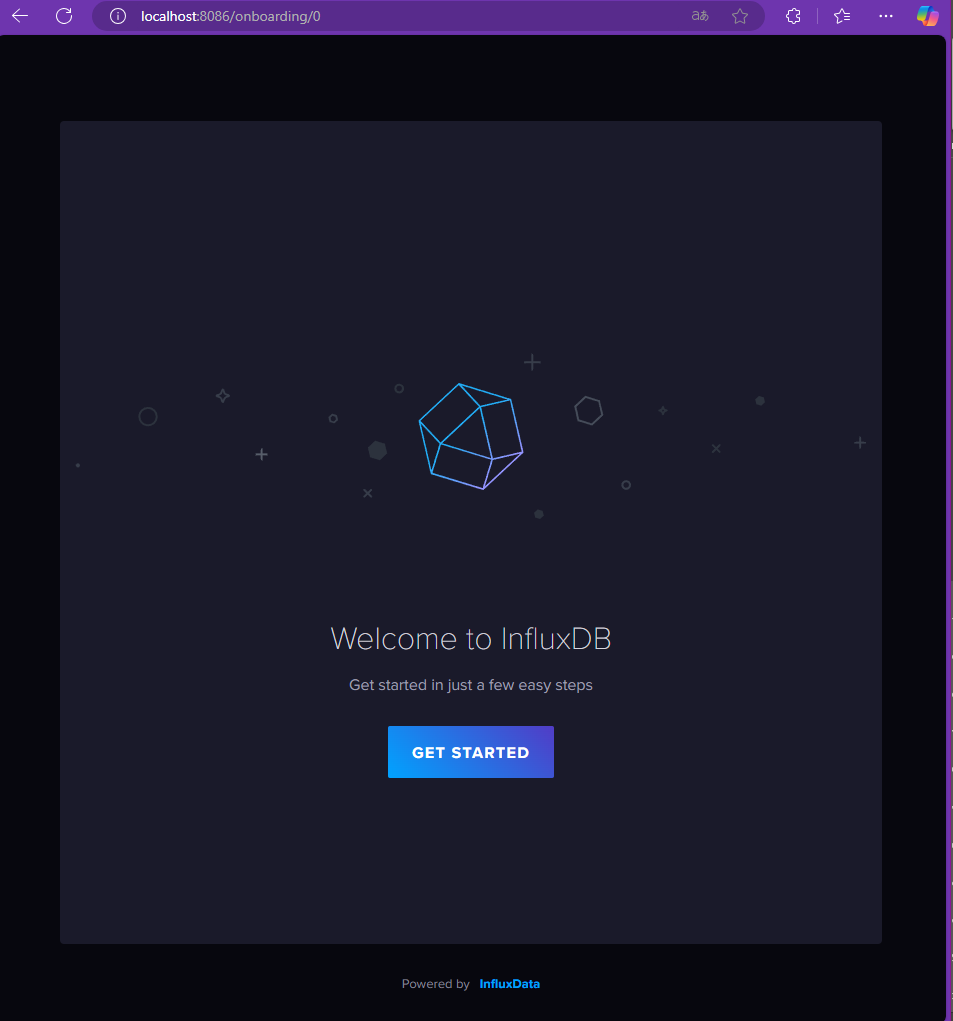
* docker ps



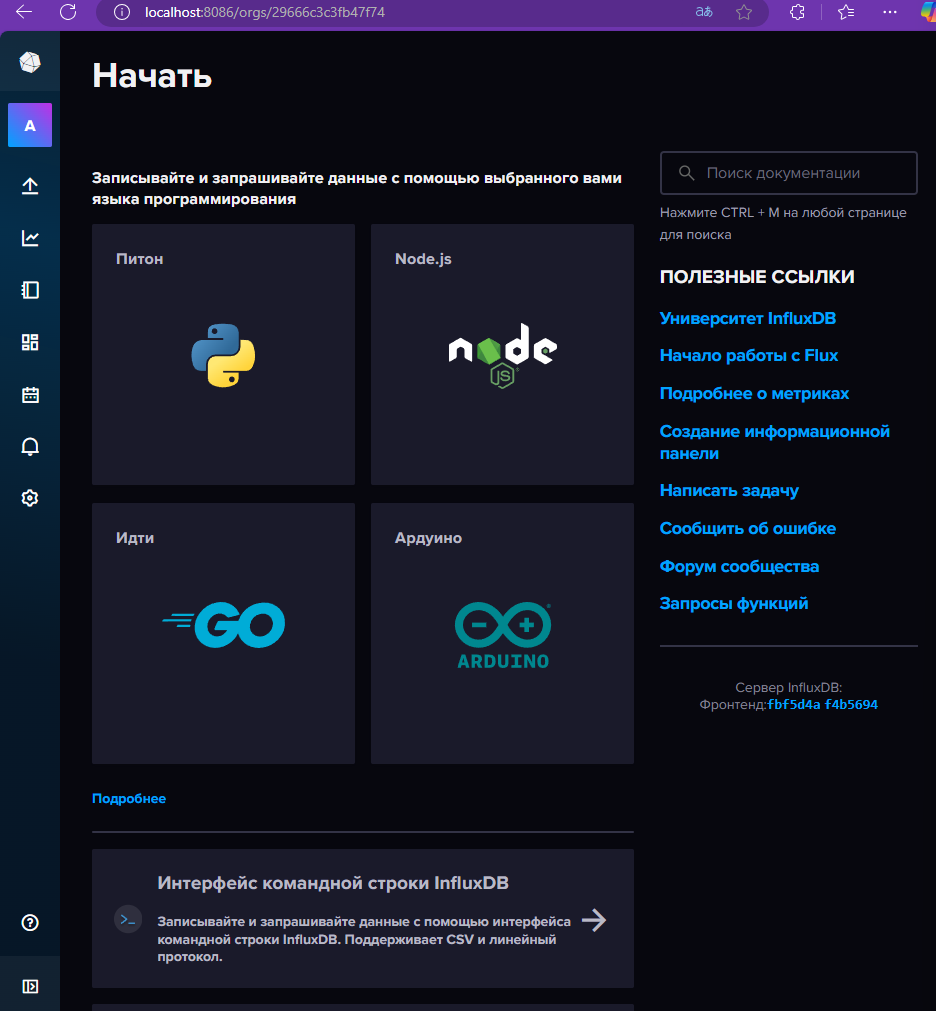
*Рис. 16*

**Шаг 3: Подключение к InfluxDB**

Проверим получили ли мы доступ к HTTP API на http://localhost:8086.



*Рис. 17*

Дальше мы создали пользователя, скопировали ключ доступа и вошли в приложение

*Рис. 18*

Для дальнейшей работы, нам нужно создать среду conda

Откроем командную сточку

Введём:

* conda create --name inf

Создали среду с название inf

* conda activate inf

Активировали среду

Теперь нужно установить пакеты для работы с докером InfluxDB

* pip install influxdb\_client

Также не забываем про прогнозирование временного хода данных сенсора.

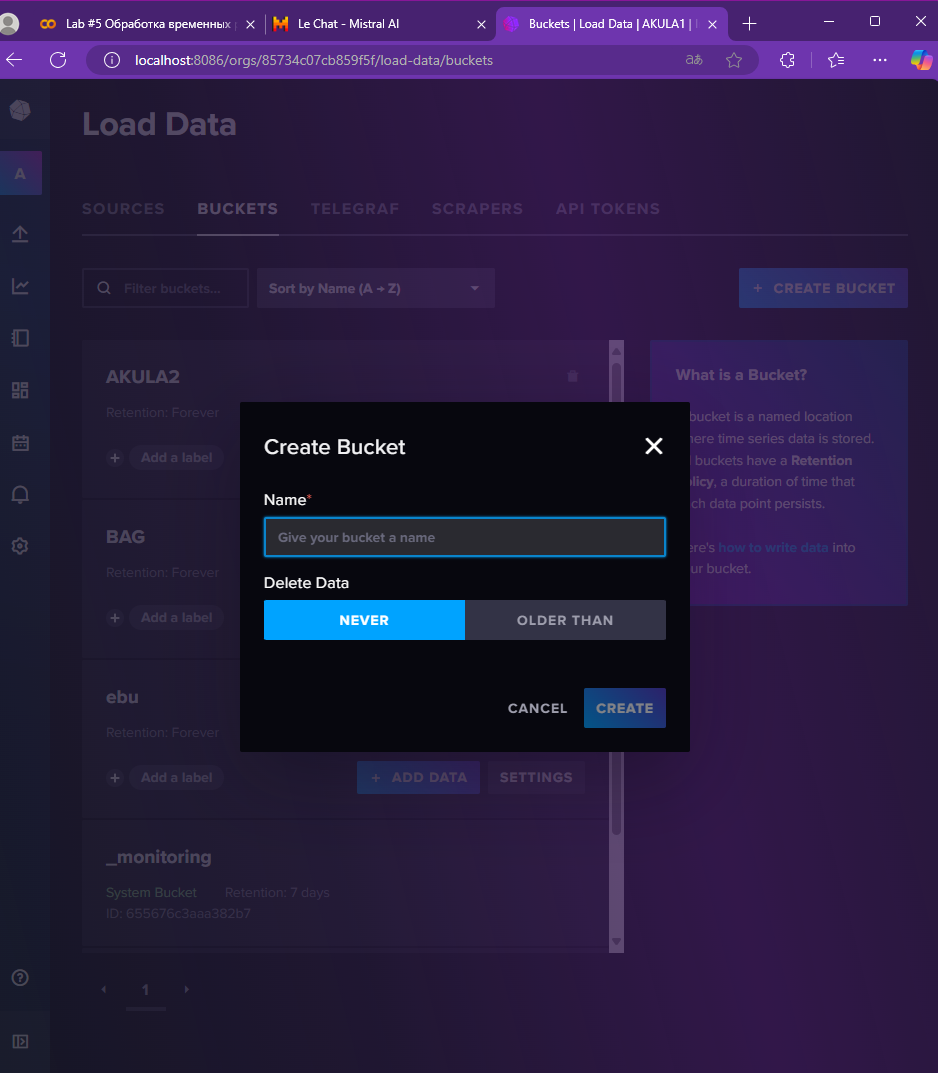
* conda install scikit-learn matplotlib
* conda install -c conda-forge xgboost

Перейдём в браузер выберем Load Data и нажмём кнопку “+ CREATE BUKET” укажем имя и нажмём создать тем самым мы создадим.

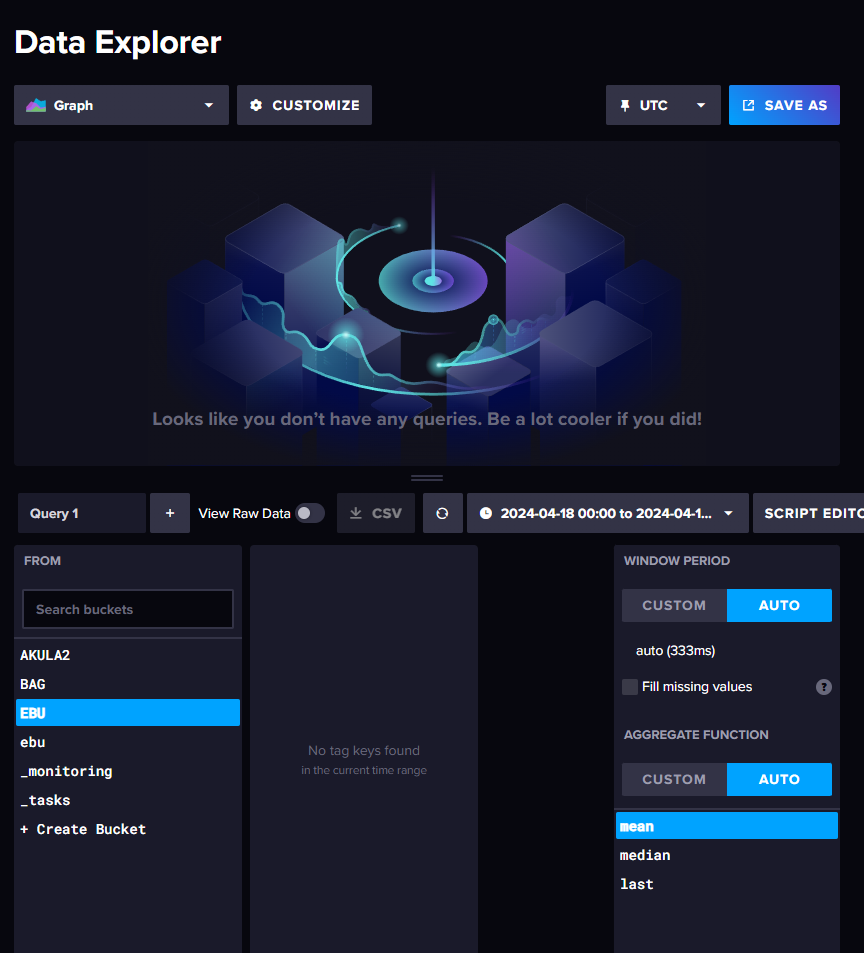
В InfluxDB бакет (bucket) — это основная единица хранения данных. Бакеты используются для организации и управления данными, а также для настройки политик хранения и сжатия. Вот основные аспекты бакетов в InfluxDB:

1. Хранение данных: Бакеты хранят временные ряды данных (time series data). Каждый бакет может содержать множество временных рядов, организованных в виде измерений (measurements), тегов (tags) и полей (fields).
2. Политики хранения: Бакеты позволяют настраивать политики хранения данных, такие как время хранения (retention period) и сжатие данных. Это помогает управлять объемом данных и оптимизировать производительность.
3. Управление доступом: Бакеты могут быть настроены с различными уровнями доступа, что позволяет контролировать, кто и как может взаимодействовать с данными в бакете.
4. Индексация: InfluxDB использует индексы для быстрого поиска и извлечения данных. Бакеты помогают организовать эти индексы для оптимальной производительности запросов.
5. Репликация: Бакеты могут быть настроены для репликации данных между несколькими узлами кластера InfluxDB, что обеспечивает высокую доступность и отказоустойчивость.

Назовём наш bucket “EBU” перейдём в Data Explorer и посмотрим, что у нас там ничего нет.

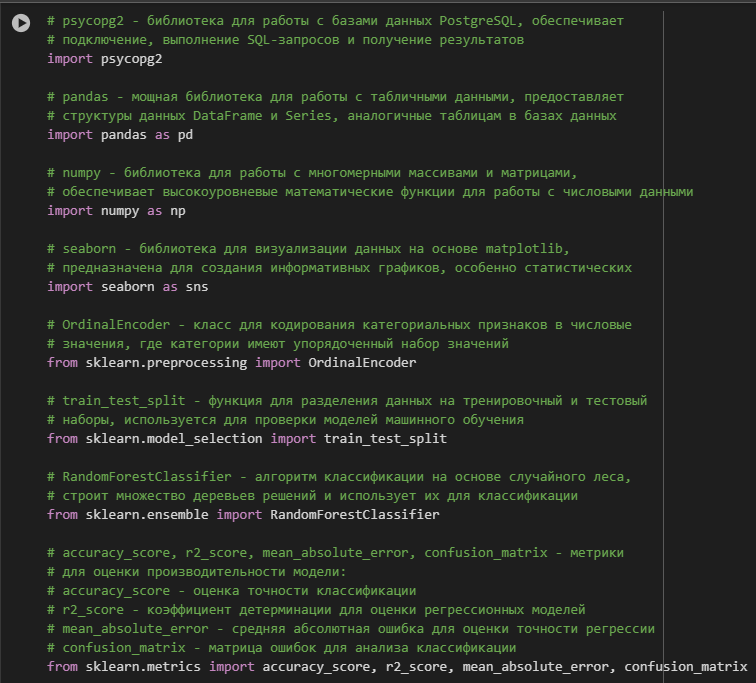


*Рис. 19*

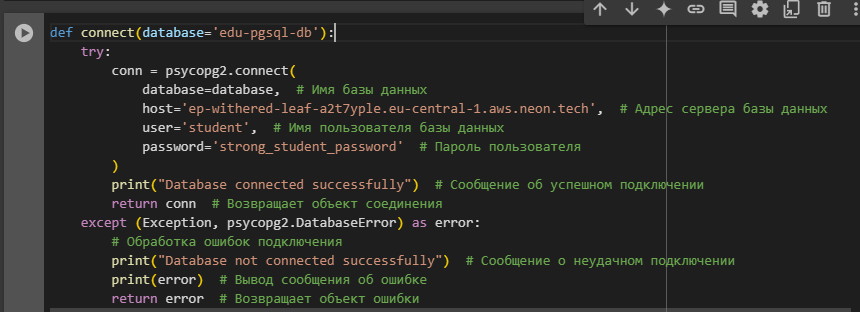


*Рис. 20*

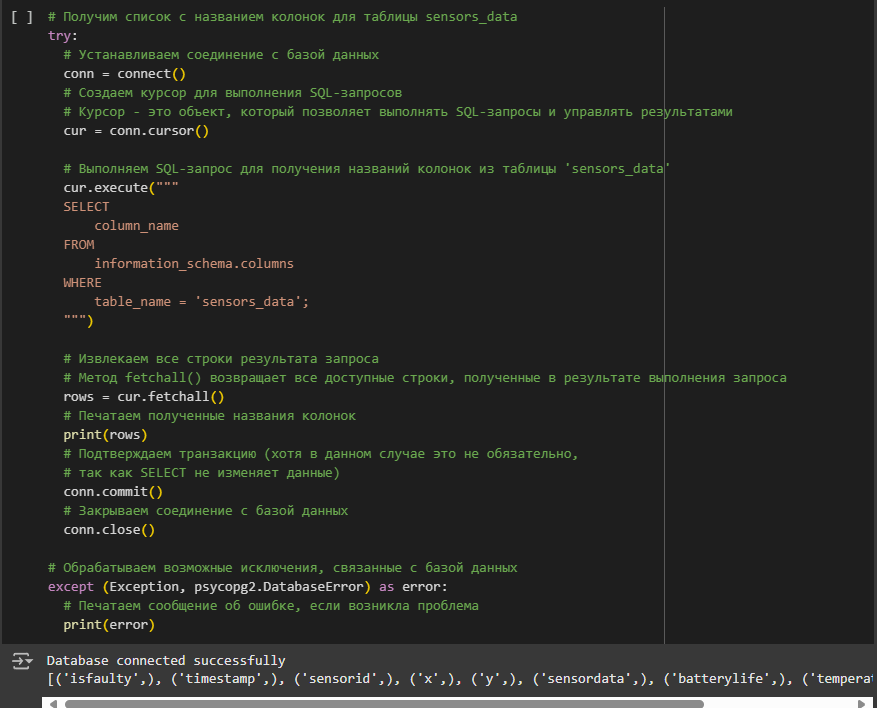
И так чтобы получить наши данные нам нужно повторить действия из 3-ей работы, но с небольшим изменением, приступим.



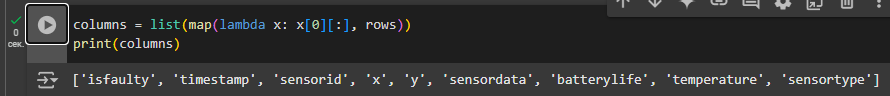
*Рис. 21*



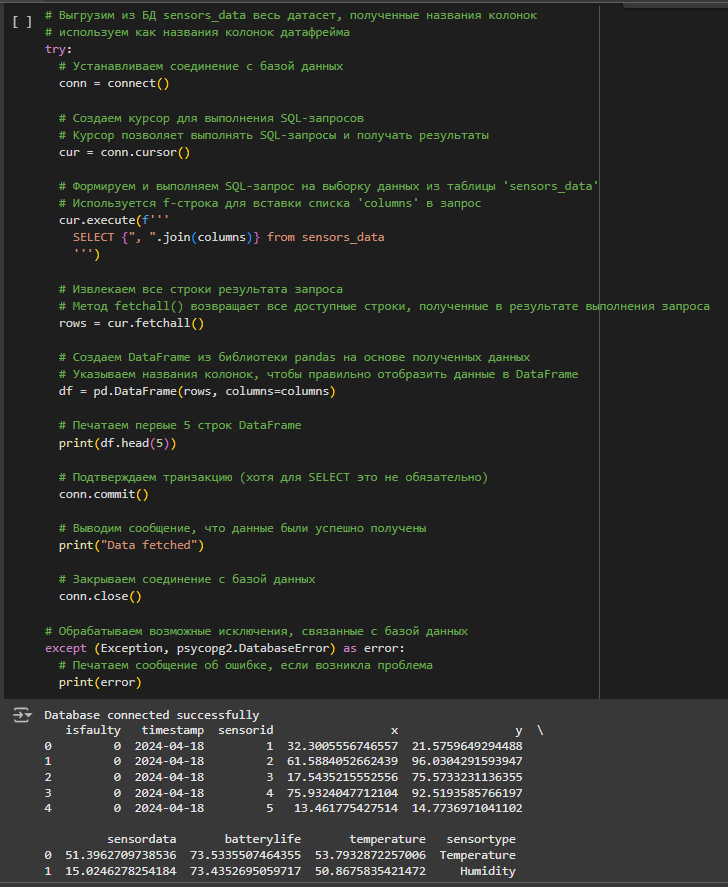
*Рис. 22*



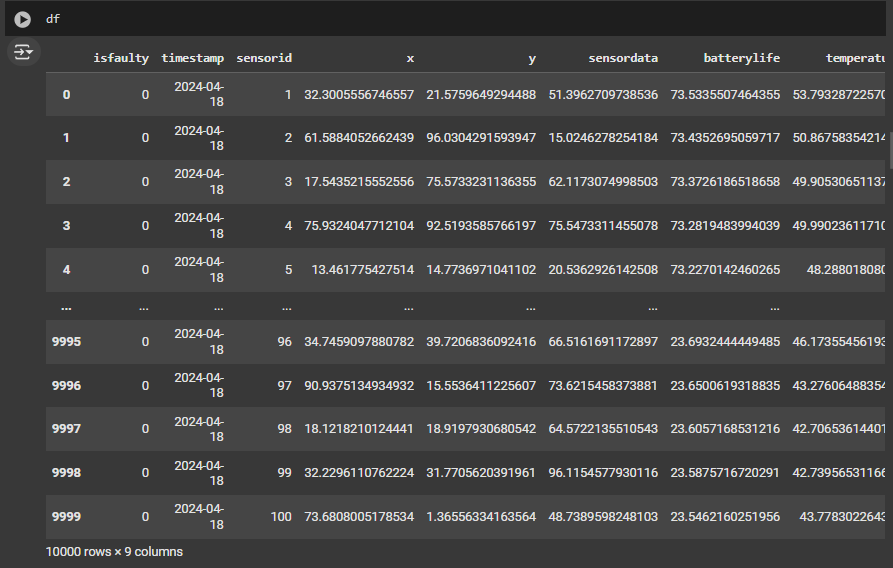
*Рис. 23*



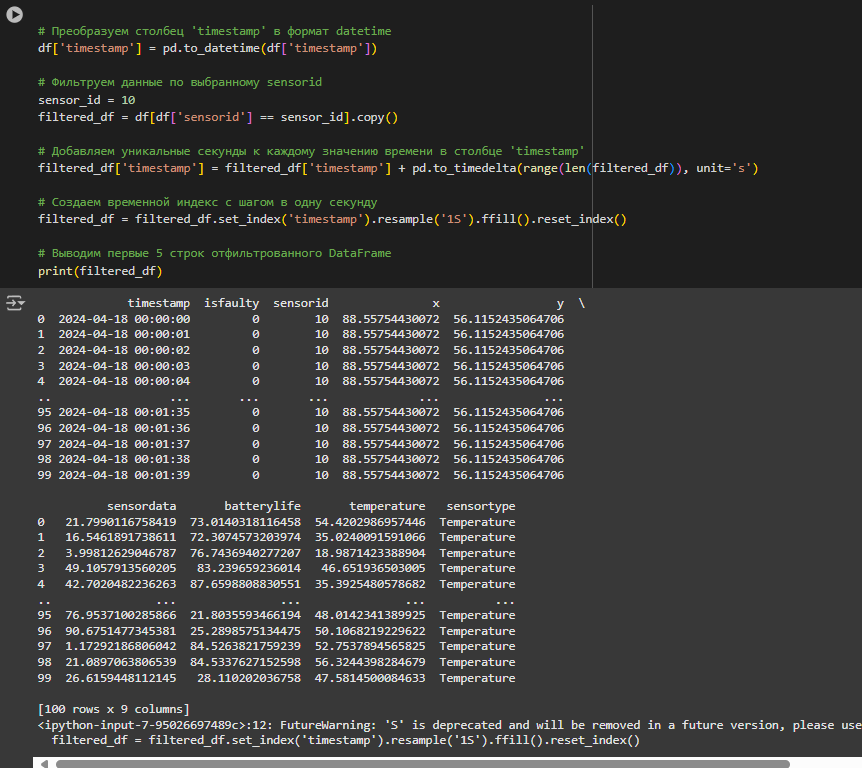
*Рис. 24*



*Рис. 25*

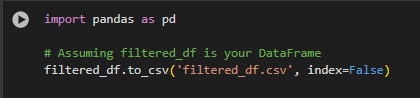


*Рис. 26*



*Рис. 27*

Мы достали сначала названия колонок, а потом достали столбцы.

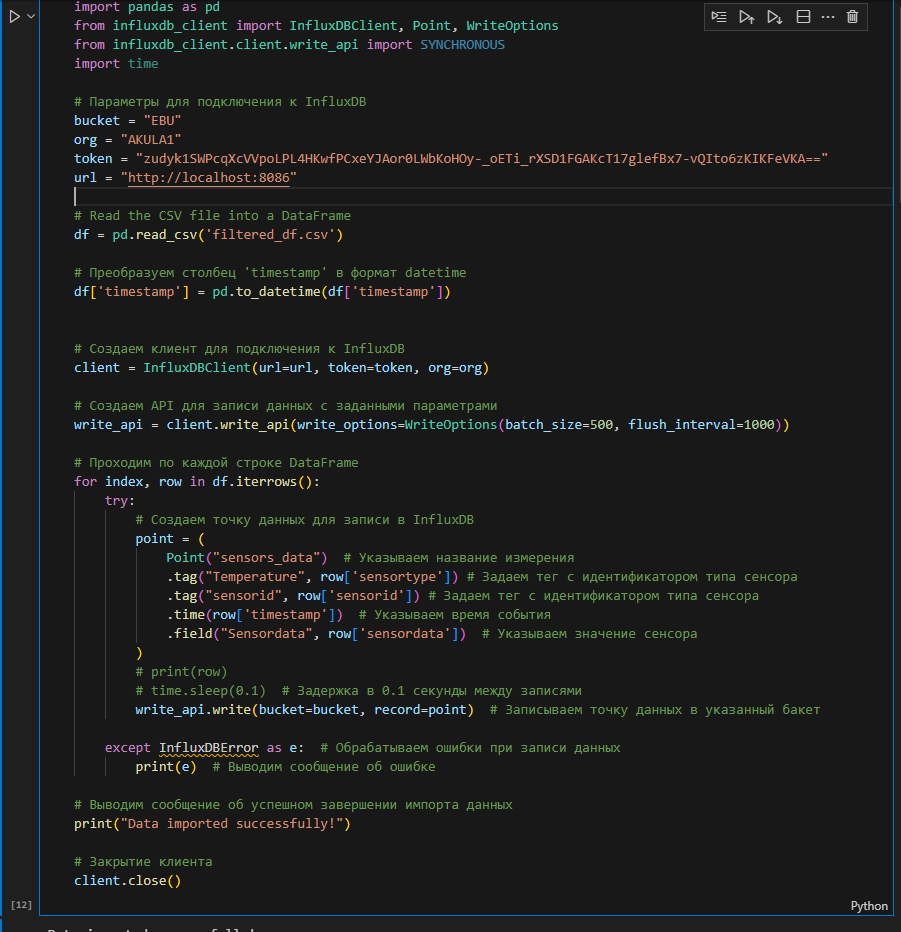


*Рис. 28*

Здесь мы скачиваем нашу БД с выборным “sensor\_id = 10”

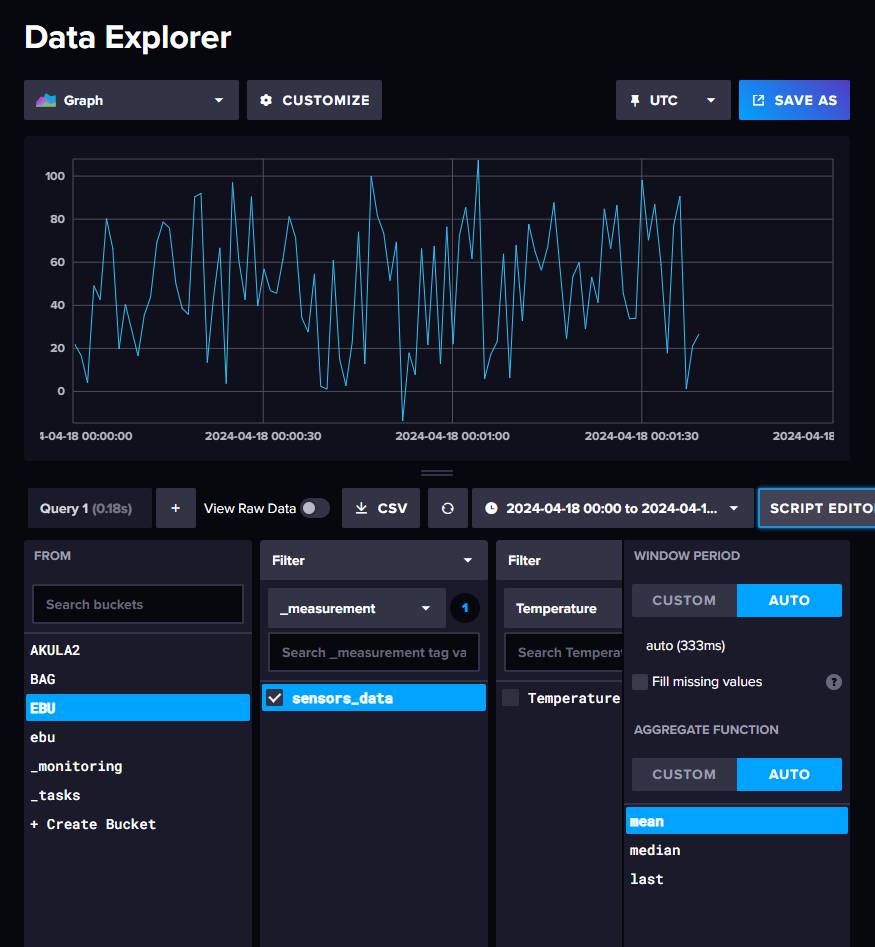
Скачиваем БД в формате csv “filtered\_df.csv”

Теперь скаченный файл нужно загрузить в InfluxDB



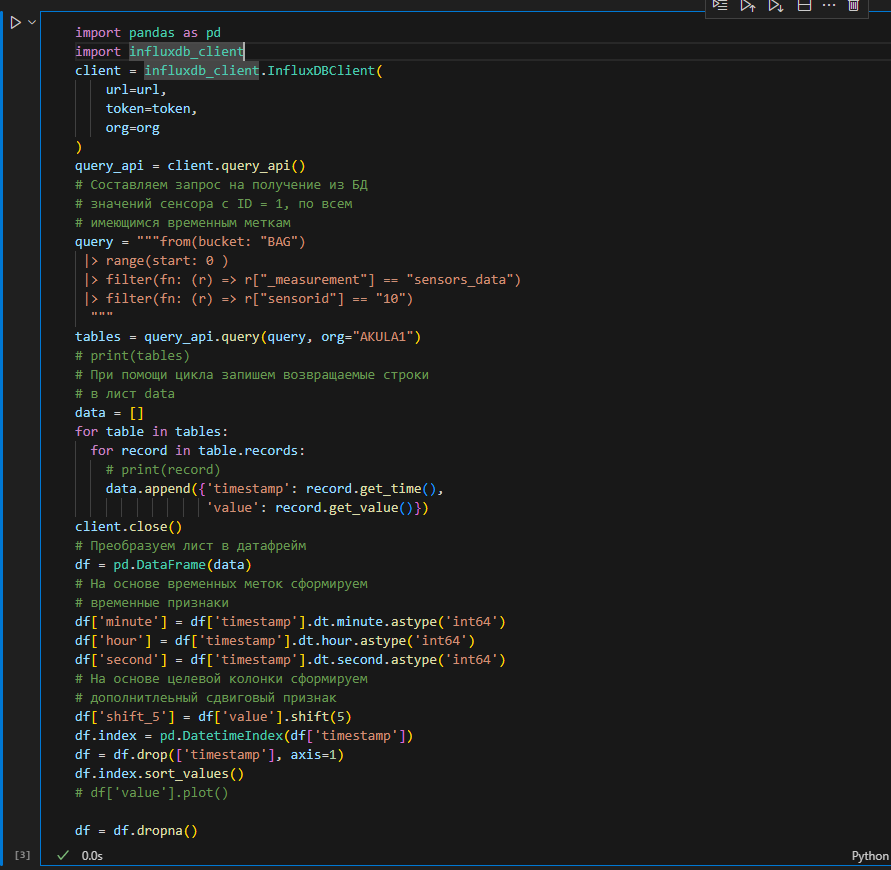
*Рис. 29*

* bucket = "BAG"
* org = "AKULA1"
* token = "zudyk1SWPcqXcVVpoLPL4HKwfPCxeYJAor0LWbKoHOy-\_oETi\_rXSD1FGAKcT17glefBx7-vQIto6zKIKFeVKA=="
* url = "http://localhost:8086"

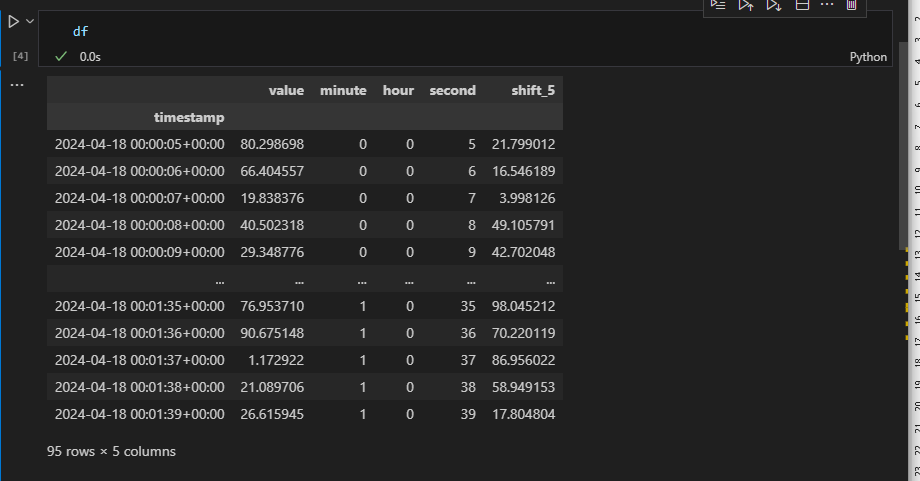


*Рис. 30*

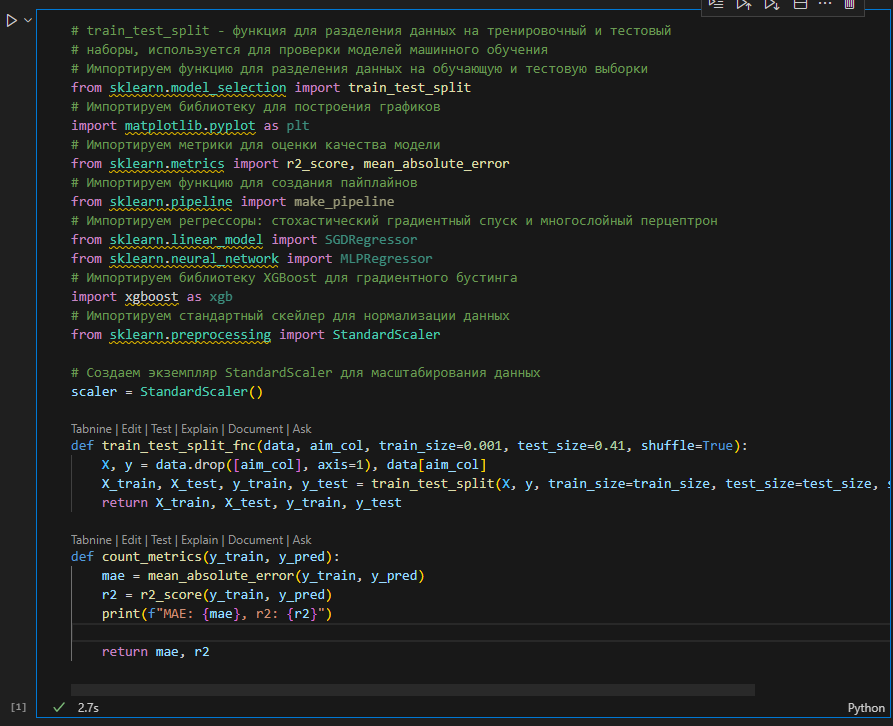
Теперь мы видим данные, загруженные из “filtered\_df.csv”.



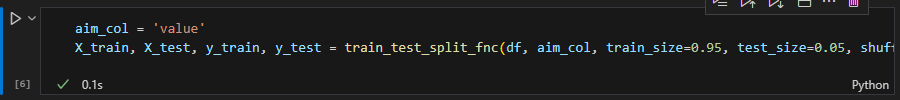
*Рис. 31*



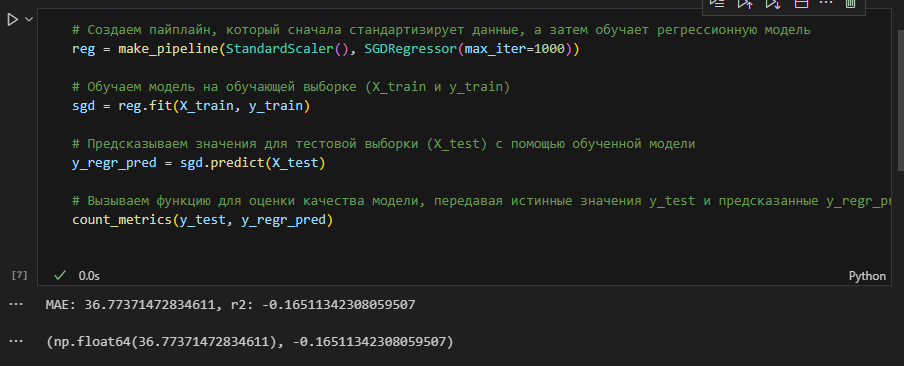
*Рис. 32*



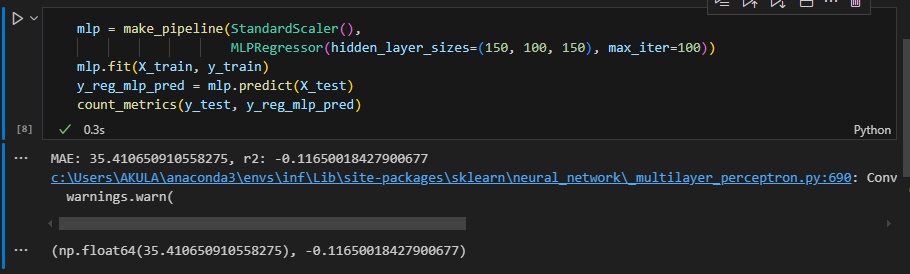
*Рис. 33*



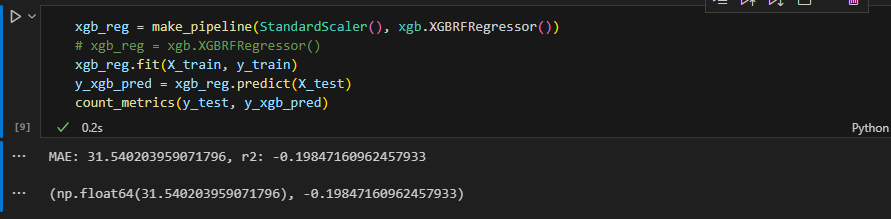
*Рис. 34*



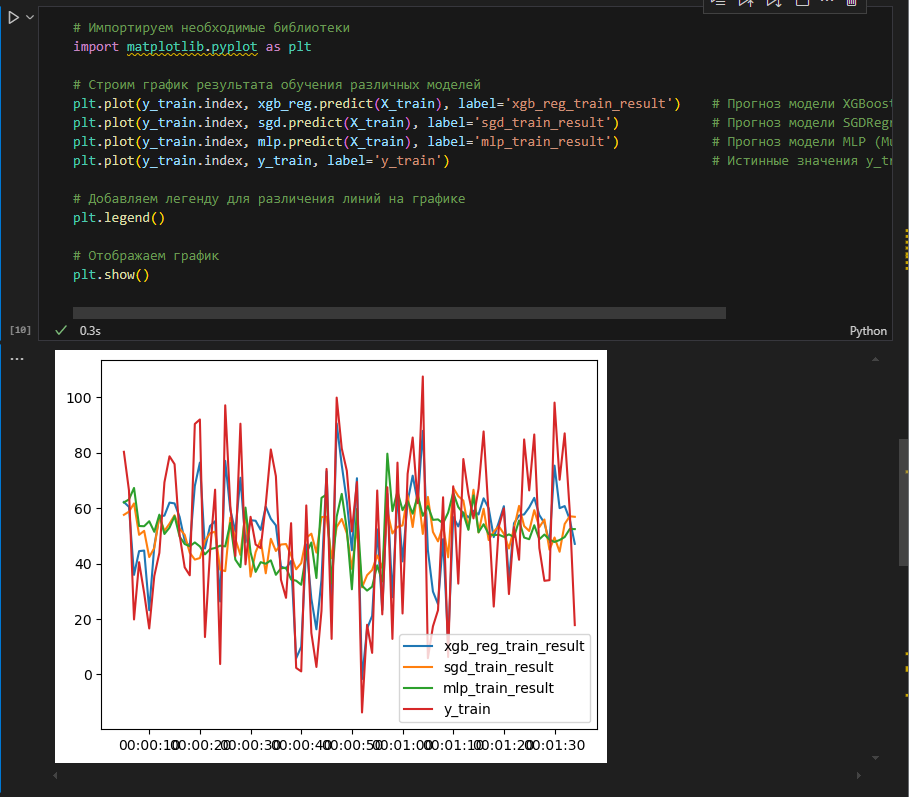
*Рис. 35*



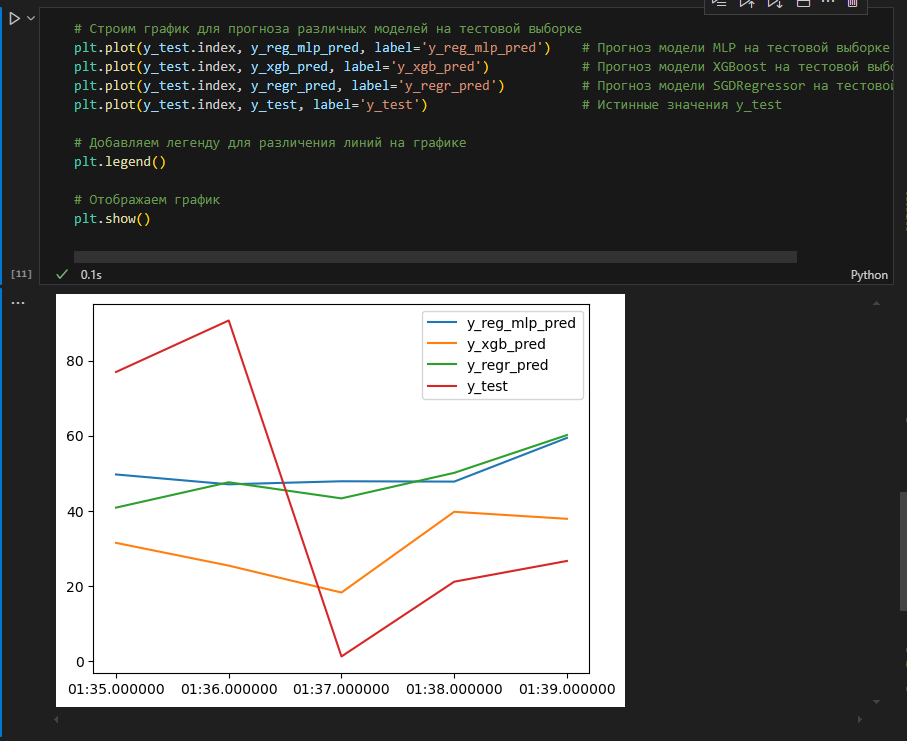
*Рис. 36*



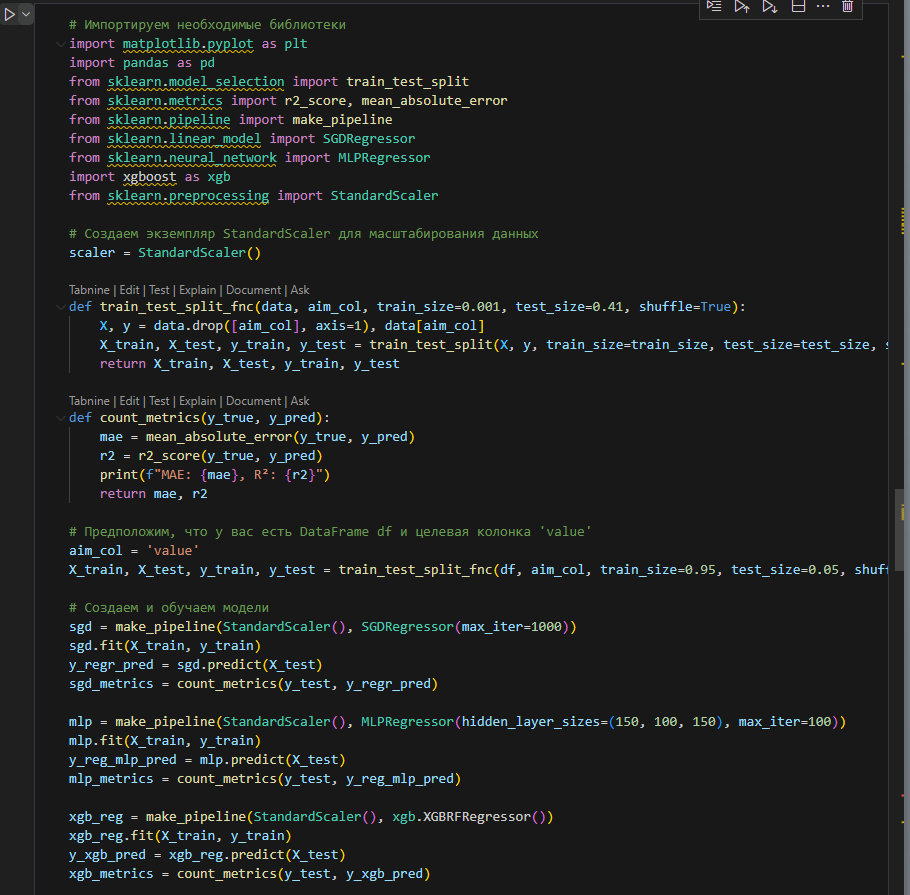
*Рис. 37*



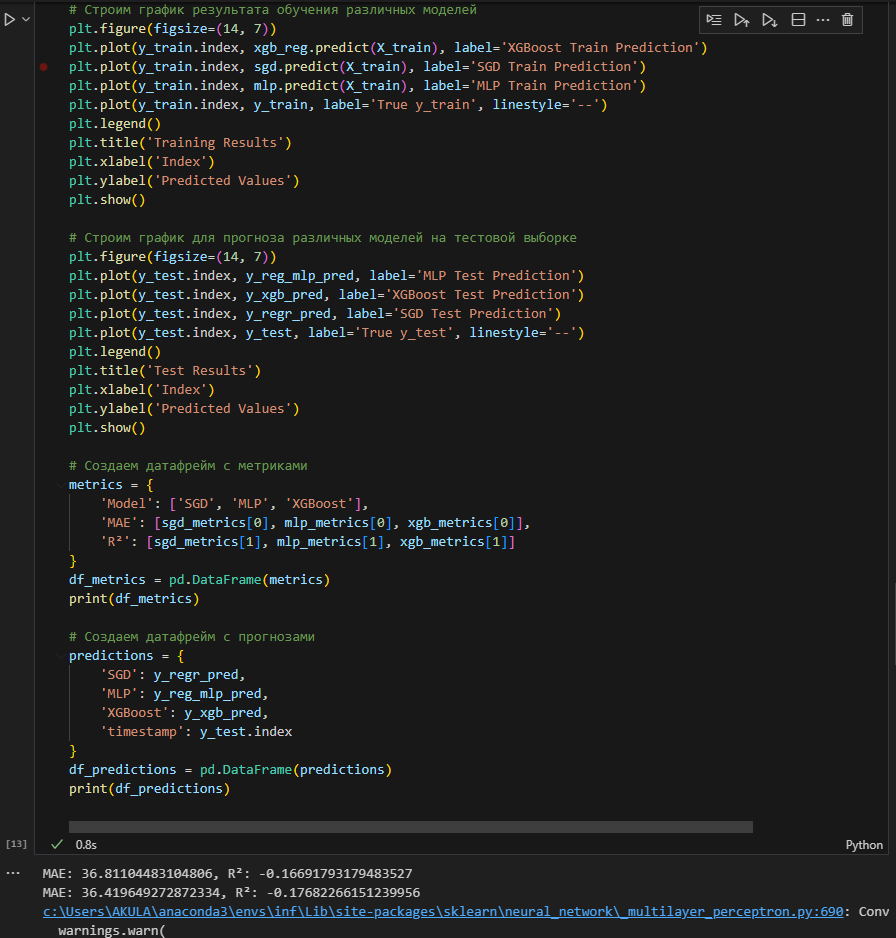
*Рис. 38*



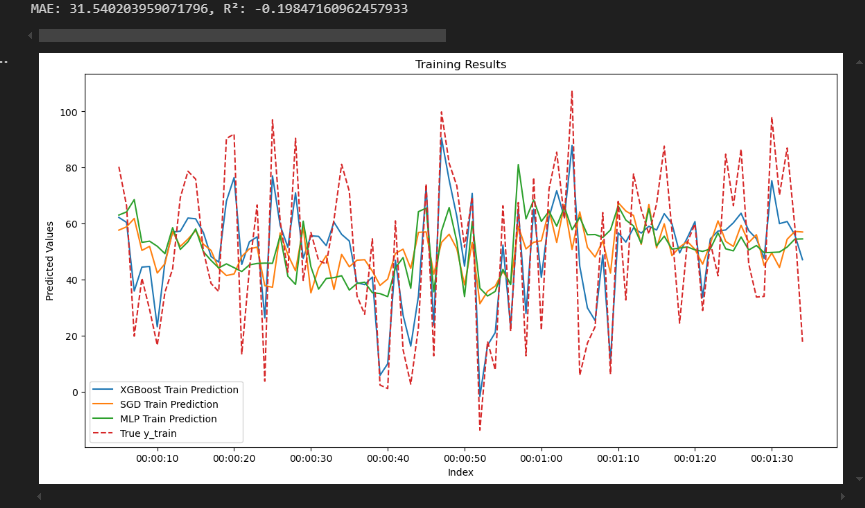
*Рис. 39*



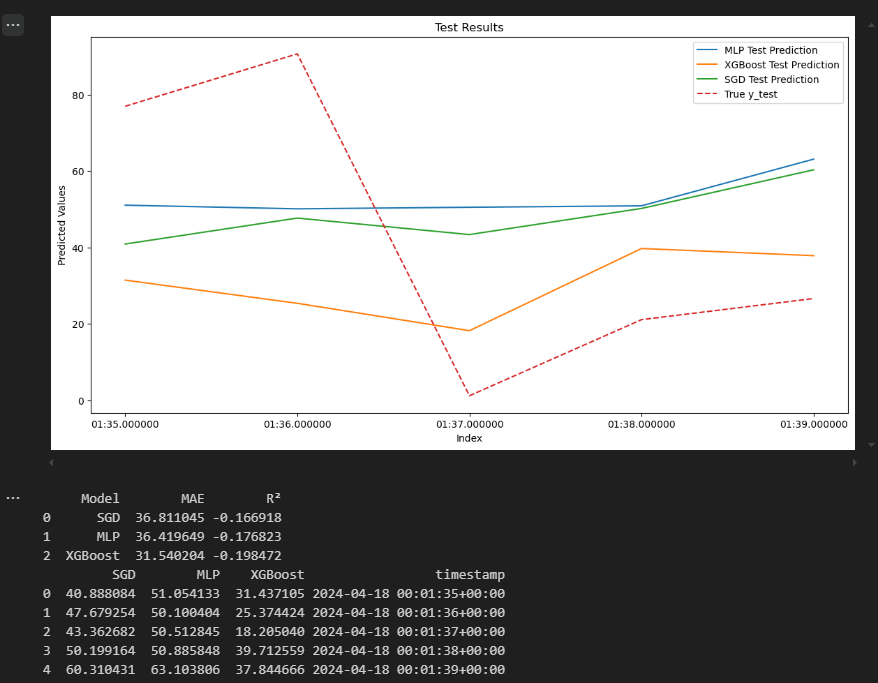
*Рис. 40*



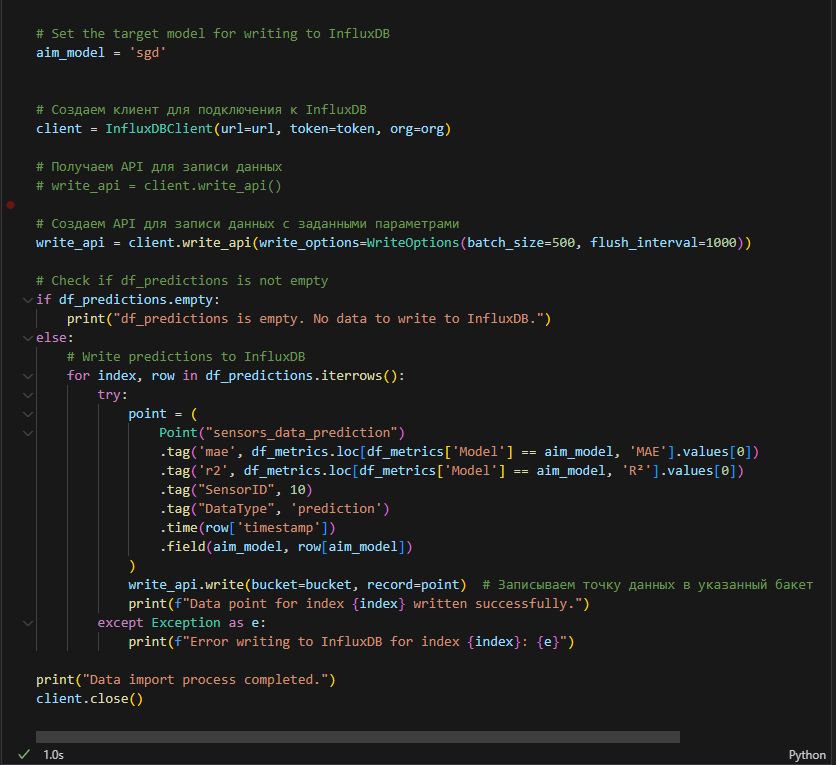
*Рис. 41*



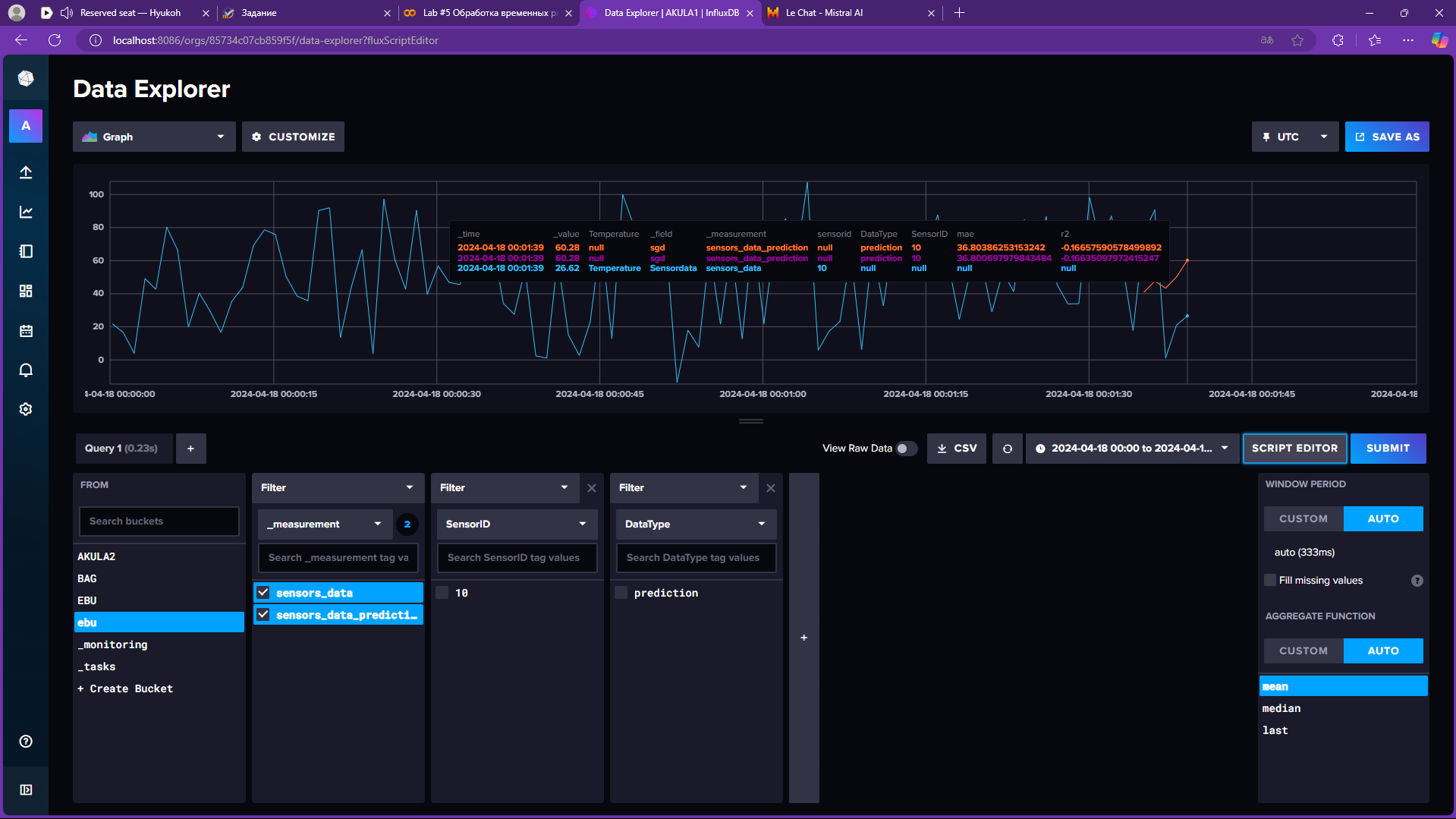
*Рис. 42*



*Рис. 43*



*Рис. 44*



*Рис. 45*

**Вывод:**

Данная работа показала, что InfluxDB является мощным инструментом для работы с временными рядами данных. Она предоставляет удобные средства для загрузки, хранения, визуализации и анализа данных.

В сочетании с моделями машинного обучения, InfluxDB позволяет эффективно прогнозировать временные ряды и анализировать результаты. Успешное выполнение всех задач подтверждает применимость InfluxDB для решения задач, связанных с временными рядами данных.

**Вывод:**