Коментарий что сделали   
Вот резюме того, что мы сделали:

1. **Создание базы данных в Neon**:
   * Мы выбрали базу данных **Neon** для хранения информации о событиях на рынке криптовалют.
   * Настроили базу данных с использованием **PostgreSQL**, подключившись к ней через URL, который включает данные для аутентификации.
2. **Создание файла для работы с базой данных**:
   * В файле database.py был реализован функционал для работы с базой данных:
     + **Подключение к базе**: Мы создали функцию connect\_db(), которая подключается к базе данных с использованием ранее созданного DATABASE\_URL.
     + **Создание таблицы**: В функции create\_table() создаётся таблица market\_events, если она ещё не существует. Таблица содержит следующие поля: id, timestamp, symbol, event\_type, price, volume, и price\_change\_5min.
     + **Сохранение данных о событии**: Функция save\_market\_event() позволяет сохранять информацию о событиях в таблице, таких как криптовалютная пара (symbol), тип события (event\_type), цена (price), объём (volume) и изменение цены за 5 минут (price\_change\_5min).
     + **Получение всех событий**: Функция get\_all\_events() позволяет извлекать все события из таблицы market\_events и возвращать их в виде списка.
3. **Основной принцип работы**:
   * Каждое событие (например, обнаружение спуфинга на криптовалютной бирже) может быть записано в базу данных с указанием криптовалютной пары, типа события, цены, объёма и изменения цены.
   * Мы можем использовать функцию для добавления новых данных в таблицу и затем запросить все события, чтобы провести анализ.

Таким образом, мы создали простую структуру для хранения данных о рыночных событиях и их дальнейшего анализа через запросы к базе данных.

Команды в шел

|  |  |
| --- | --- |
| C:\Users\user>psql "postgresql://neondb\_owner:npg\_A8UYieglXhJ9@ep-tiny-wildflower-a8bf72pn-pooler.eastus2.azure.neon.tech/neondb?sslmode=require" | Захожу в neondb |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Crypto Signal Bot — это бот, который анализирует **стакан ордеров (Order Book)** на Binance и отправляет уведомления в Telegram о важных рыночных событиях.  
Бот умеет отслеживать **Spoofing (манипуляции с ордерами), резкие скачки цен и другие рыночные движения.**

**2. Структура файлов проекта**

Проект состоит из нескольких файлов, каждый из которых отвечает за свою часть логики:

📂 crypto\_signal\_bot.py — основной файл бота.  
📂 database.py — работа с базой данных Neon PostgreSQL.  
📂 config.py — хранит настройки, такие как API-токены.  
📂 requirements.txt — список зависимостей для установки библиотек.

# **3. Как файлы взаимодействуют**

Бот работает по следующему принципу:

1️⃣ crypto\_signal\_bot.py **запрашивает данные с Binance**.  
2️⃣ Обнаруженные события **передаются в database.py**, где записываются в базу данных.  
3️⃣ database.py **сохраняет их в таблицу market\_events в Neon PostgreSQL**.  
4️⃣ При необходимости **бот запрашивает данные из базы и отправляет их в Telegram**.

# NEON

 Host (например, your-db-name.neon.tech)

 Database (например, your\_database)

 User (например, your\_username)

 Password (ваш пароль)

 Port (обычно 5432)

postgresql://neondb\_owner:npg\_A8UYieglXhJ9@ep-tiny-wildflower-a8bf72pn-pooler.eastus2.azure.neon.tech/neondb?sslmode=require

### **1. Экспорт структуры (без данных)**

Если вам нужна только структура (таблицы, колонки, индексы), используйте pg\_dump:

sh (Shell) — это командный интерпретатор в Linux и macOS, а также в терминале Windows с установленным **Git Bash** или **WSL** (Windows Subsystem for Linux).

Как ввести команды sh?

В зависимости от вашей системы:

🔹 В Windows

Откройте **Командную строку (cmd)**

Нажмите Win + R, введите cmd, нажмите Enter

Либо используйте **PowerShell** (Win + X → Windows Terminal)

Команды, которые я привел выше (pg\_dump, pg\_restore), запускаются **в командной строке**, а не в SQL-редакторе.

sh

pg\_dump -h <your-neon-host> -U <your-username> -d <your-database> -s -f schema.sql

🔹 Флаг -s означает **"schema-only"**, то есть экспортируется только структура без данных.

Вы получите файл schema.sql, который можно мне отправить.

**4. Как работает база данных Neon**

**Neon** — это облачная база данных PostgreSQL, в которой хранятся рыночные события.  
База данных содержит таблицу **market\_events**, в которой есть следующие поля:

id — уникальный идентификатор события.

timestamp — время события.

symbol — торговая пара (например, BTC/USDT).

event\_type — тип события (например, Spoofing).

price — цена в момент события.

volume — объём ордеров.

price\_change\_5min — изменение цены за 5 минут.

### **. Дополнительные данные, которые нужно добавить**

🔹 spoofing\_volume – объём исчезнувших ордеров при Spoofing-е (уже добавили).  
🔹 spoofing\_count – сколько раз Spoofing был зафиксирован на этой цене за последние 10 минут.  
🔹 spread – разница между лучшим бидом и аском в момент события (полезно для анализа ликвидности).  
🔹 liquidation\_volume – если фиксируем ликвидации, записываем сумму ликвидированных позиций.  
🔹 trend\_direction – направление тренда в момент события (UP или DOWN).

# **5. Примеры работы с базой данных**

📌 **Как сохранить событие в базе:**

python

save\_market\_event(

symbo1="BTC/USDT",  
event\_type="Spoofing",  
price=43700,  
volume=2500000,  
price\_change\_5min=0.5

📌 **Как получить последние 10 событий:**

events = get\_all\_events()[-10:]

for event in events:

print(event)

**6. Как добавить новые функции**

Чтобы добавить новую функцию в бота, нужно:  
1️⃣ Добавить команду в crypto\_signal\_bot.py.  
2️⃣ Если нужно хранить данные, добавить запись в database.py.  
3️⃣ Обновить команды Telegram, если бот должен реагировать на новые запросы.

# СПУФФИНГ

**🚀 1. Что бот будет делать при фиксации Spoofing-а?**

Когда бот обнаружит Spoofing, он:  
1️⃣ **Запишет данные в базу Neon** (добавит новый ряд в таблицу market\_events).  
2️⃣ **Проанализирует аномалию** (сколько раз Spoofing был на этой цене).  
3️⃣ **Оценит тренд** (где больше ликвидности, куда двигаются ордера).  
4️⃣ **Отправит сигнал в Telegram** с детальной информацией.

основными командами cmd

| **Команда** | **Описание** |
| --- | --- |
| **🔹 Работа с виртуальным окружением (venv)** |  |
| python -m venv venv | Создать виртуальное окружение venv в текущей папке |
| venv\Scripts\activate | Активировать venv (Windows, cmd) |
| source venv/bin/activate | Активировать venv (Linux/macOS) |
| deactivate | Деактивировать venv и вернуться в глобальную среду |
| **🔹 Проверка установленных библиотек** |  |
| pip list | Показать список установленных библиотек в venv |
| pip show aiogram | Показать информацию о конкретной библиотеке (aiogram) |
| pip freeze | Показать список зависимостей (можно скопировать в requirements.txt) |
| **🔹 Работа с requirements.txt** |  |
| pip freeze > requirements.txt | Сохранить список установленных библиотек в requirements.txt |
| pip install -r requirements.txt | Установить все библиотеки из requirements.txt |
| **🔹 Установка и удаление библиотек** |  |
| pip install aiogram | Установить библиотеку aiogram в venv |
| pip install aiogram==3.1.0 | Установить конкретную версию aiogram |
| pip uninstall aiogram | Удалить aiogram из окружения |
| **🔹 Проверка пути установки библиотек** |  |
| pip show aiogram | Проверить, где установлена библиотека |
| where python (Windows) | Проверить, какой Python используется |
| which python (Linux/macOS) | Проверить путь к используемому Python |
| **🔹 Запуск бота** |  |
| python crypto\_signal\_bot.py | Запустить бота в venv |

Решения

Switch to the server base (PostgreSQL) so you can connect from any device.

Установки

psycopg2 (pip install psycopg2-binary)

Решения

PostgreSQL.

<https://console.neon.tech/>

Connection Details" в Neon.tech

postgresql://neondb\_owner:npg\_A8UYieglXhJ9@ep-tiny-wildflower-a8bf72pn-pooler.eastus2.azure.neon.tech/neondb?sslmode=require

Файлы

database.py

crypto\_signal\_bot.py

### **Короткий план для нового чата**

Если новая версия чата не в курсе дела, объясните ей кратко:

* **Проект:** База данных в Neon для крипто-бота.
* **Главная таблица:** market\_events (анализ рынка).
* **Доступ:** Через psql, pg\_dump и GUI-клиенты (DBeaver, pgAdmin).
* **Основные команды:** Проверка таблиц, восстановление схемы, подключение через psql.