Тестовое задание: Telegram-бот для аналитики по видео на основе задач на естественном языке

1. Контекст

У нас есть внутренний сервис сбора статистики по видео-креаторам публикующим контент в разных соц сетях.
Сервис хранит две таблицы:

- итоговую статистику по каждому видео (просмотры, лайки, комментарии и т.д.);
- почасовые «снапшоты» статистики по каждому видео (чтобы отслеживать динамику).

Вам нужно сделать Telegram-бота, который умеет по запросу на естественном языке считать нужные метрики по этим данным.

---

2. Что нужно сделать

1. **Развернуть базу данных**
2. **Загрузить предоставленный JSON-файл** в базу, разложив его на нормальные таблицы. Json файл скачать тут -> ссылка
3. **Реализовать Telegram-бота**, который:
    - принимает текстовые сообщения от пользователя бота на русском (естественный язык);
    - по этому тексту строит запрос к данным (основная цель тестового посмотреть, как вы реализуете эту часть, здесь есть несколько рабочих подходов, которые не будут галлюцинировать на таком объеме данных);
    - возвращает пользователю ответ - одно число (счётчик, сумму или прирост в зависимости от запроса).
4. Распознавание естественного языка:
    - можно использовать любого API провайдера LLM или локально развернутую LLM
5. **Важно:** вам нужно продумать и написать такой промпт/описание схемы данных, чтобы модель сама понимала, из каких таблиц что брать и как считать.

---

3. Данные

В архиве — один JSON-файл, в котором лежит массив объектов `videos`.
Каждый объект — это одно видео с вложенным списком почасовых снапшотов.

Таблица `videos` (итоговая статистика по ролику)

- `id` — идентификатор видео;
- `creator_id` — идентификатор креатора;
- `video_created_at` — дата и время публикации видео;
- `views_count` — финальное количество просмотров;
- `likes_count` — финальное количество лайков;
- `comments_count` — финальное количество комментариев;
- `reports_count` — финальное количество жалоб;
- служебные поля `created_at`, `updated_at`.

Таблица `video_snapshots` (почасовые замеры по ролику)

Каждый снапшот относится к одному видео и содержит:

- `id` — идентификатор снапшота;
- `video_id` — ссылка на соответствующее видео;
- текущие значения: `views_count`, `likes_count`, `comments_count`, `reports_count` на момент замера;
- приращения: `delta_views_count`, `delta_likes_count`, `delta_comments_count`, `delta_reports_count` — насколько изменилось значение с прошлого замера;
- `created_at` — время замера (раз в час);
- `updated_at` — служебное поле.

---

4. Требования к боту

Бот должен уметь отвечать на вопросы вида (примеры):

- «Сколько всего видео есть в системе?»
- «Сколько видео у креатора с id `...` вышло с 1 ноября 2025 по 5 ноября 2025 включительно?»
- «Сколько видео набрало больше 100 000 просмотров за всё время?»

- «На сколько просмотров в сумме выросли все видео 28 ноября 2025?»
- «Сколько разных видео получали новые просмотры 27 ноября 2025?»

Характер запроса:

- вопросы задаются **на русском, естественным языком**;
- даты могут быть в виде: «28 ноября 2025», «с 1 по 5 ноября 2025» и т.п.;
- в ответе от бота ожидается **одно число**

Мы будем автоматически прогонять по вашему боту набор своих запросов (в духе примеров выше) и сравнивать ответы с эталонными.

---

5. Требования к реализации

Технологии:

- Язык программирования — Python
- База — **PostgreSQL**.
- Telegram-бот — aiogram.
- Весь стек должен быть асинхронный! Запросы к бд, запросы к llm и тд

Внутренняя логика:

- **один запрос к боту → один числовой ответ**
  Хранить контекст диалога не нужно

---

6. Как мы будем проверять

После того как вы развернули бота и он доступен в Telegram, вам нужно запустить проверку через нашего служебного бота **@rlt_test_checker_bot**.

1. Откройте чат с **@rlt_test_checker_bot**.
    - Отправьте ему команду в формате:
      `/check @yourbotnickname https://github.com/yourrepo`
2. Где:
    - `@yourbotnickname` — никнейм вашего Telegram-бота, которого нужно проверить;
    - `https://github.com/yourrepo` — ссылка на репозиторий с исходным кодом;

3. Наш служебный бот автоматически:

    - пройдётся по набору тестовых запросов (промптов) к вашему боту;
    - дождётся ответов;
    - сравнит ответы с эталонными значениями.

    Важно: ваш бот на момент проверки **должен быть запущен и доступен**, иначе тесты просто не пройдут.

Результат проверки:

- если все тестовые запросы прошли корректно — служебный бот напишет вам, что проверка успешна;
- если в каком-то из промптов будет неправильный расчёт — вы получите сообщение:
    - с указанием, на каком именно запросе произошла ошибка;
    - и с фактом, что ответ не совпал с ожидаемым.

После исправления вы можете снова запустить проверку тем же `/check` (количество попыток не ограничено).

---

7. Что нужно подготовить

К моменту запуска команды `/check` у вас должно быть:

1. **Работающий Telegram-бот**

    - корректно настроенный токен;
    - бот развернут и отвечает на сообщения.

2. **Публичный репозиторий с кодом** (GitHub / GitLab / Bitbucket), ссылка на который передаётся в `/check`. В репозитории должны быть:

    - исходный код бота и всего backend-сервиса;
    - SQL-миграции или скрипты для создания таблиц в PostgreSQL;
    - скрипт или инструкция по загрузке предоставленного JSON-файла в базу;
    - `README`, в котором описано:

        - как запустить проект локально (желательно с Docker, но можно и без);
        - как задать токен Telegram-бота;
        - краткое описание архитектуры и подхода к преобразованию текстовых запросов в обращения к БД (SQL / код);

- если используется LLM — как именно вы описываете схему данных и какой промпт применяете.

Важно: если у вас проблемы и что-то не работает — проверяйте со своей стороны. Валидация точно верная, поэтому писать с вопросами насчет этого не нужно.

В случае других вопросов — пишите в тг @tglabs_manager