**Постановка задачи для BI-AI-agent**

**Проект: Разработка MVP ИИ-агента для анализа данных торговой сети**

**Общее описание**

Требуется разработать MVP интеллектуального агента, который позволит пользователям получать табличные данные из корпоративного хранилища данных торговой сети продуктов питания посредством запросов на естественном языке. Прототип должен демонстрировать возможность перевода естественно-языковых запросов в SQL-запросы к данным с последующей визуализацией результатов в табличном формате.

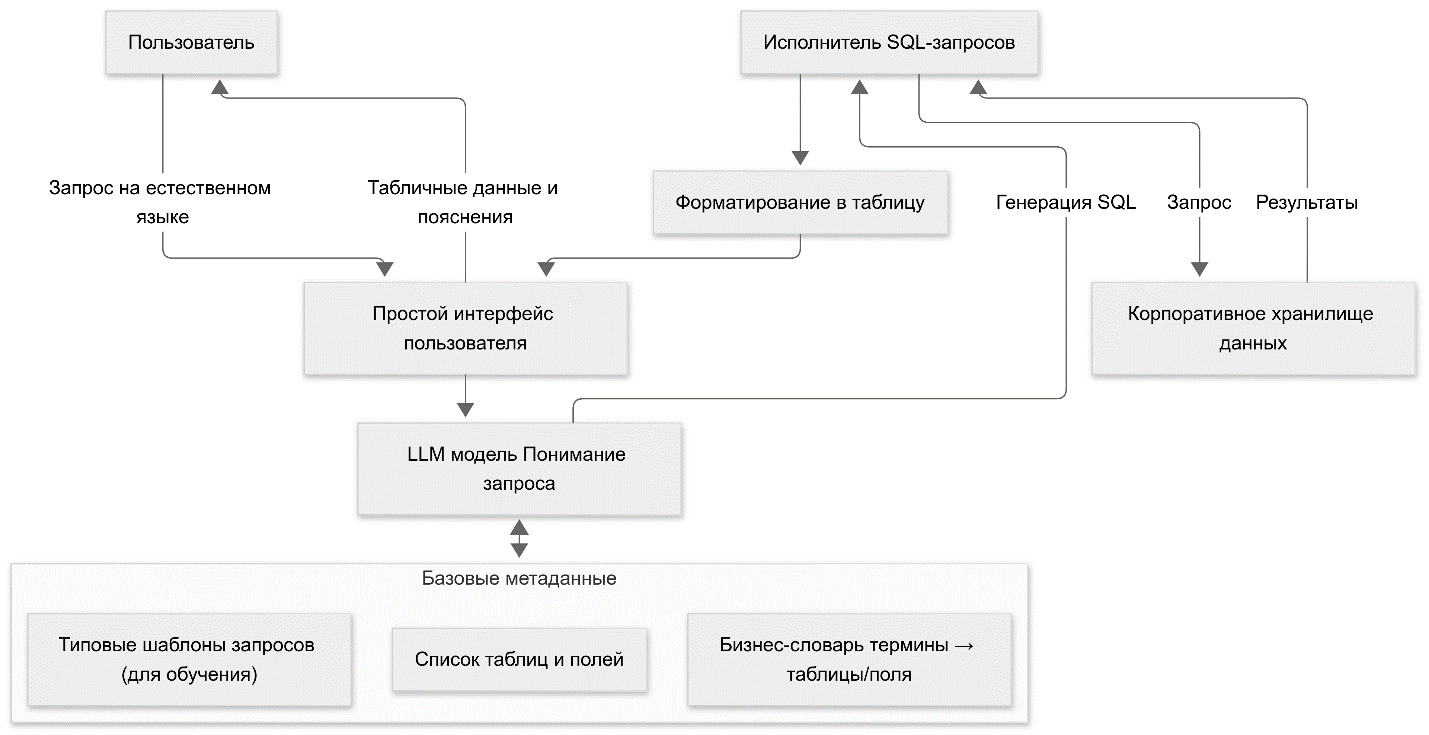
**Технологический стек**

* **Язык программирования:** Python 3.9+
* **Фреймворк интерфейса:** Streamlit
* **База данных:** DuckDB
* **LLM API:** OpenAI API (GPT-4о mini)
* **Хостинг:** share.streamlit.io
* **Вспомогательные библиотеки:** pandas, numpy, plotly (опционально)

**Архитектура решения**

User (запрос) → Streamlit Interface → LLM (понимание и генерация SQL) →

→ QueryExecutor (выполнение в DuckDB) → Formatter → Streamlit Interface → User (результаты)



Метаданные:

* Структура таблиц и полей
* Словарь бизнес-терминов
* Примеры типовых запросов

**Структура синтетических данных торговой сети**

**1. Таблица магазинов (stores)**

* store\_id (INT): Уникальный идентификатор магазина
* store\_name (TEXT): Название магазина
* format (TEXT): Формат магазина (гипермаркет, супермаркет, мини-маркет)
* region (TEXT): Регион расположения
* city (TEXT): Город
* open\_date (DATE): Дата открытия
* size\_sqm (FLOAT): Площадь магазина
* is\_active (BOOLEAN): Активен ли магазин

**2. Таблица товаров (products)**

* product\_id (INT): Уникальный идентификатор товара
* product\_name (TEXT): Наименование товара
* category\_id (INT): Идентификатор категории
* subcategory\_id (INT): Идентификатор подкатегории
* brand (TEXT): Бренд
* supplier\_id (INT): Идентификатор поставщика
* unit\_price (FLOAT): Базовая цена за единицу
* unit\_cost (FLOAT): Себестоимость единицы
* unit\_type (TEXT): Единица измерения (кг, шт, л)
* is\_private\_label (BOOLEAN): Является ли собственной торговой маркой

**3. Таблица категорий (categories)**

* category\_id (INT): Уникальный идентификатор категории
* category\_name (TEXT): Название категории
* department (TEXT): Отдел

**4. Таблица подкатегорий (subcategories)**

* subcategory\_id (INT): Уникальный идентификатор подкатегории
* category\_id (INT): Идентификатор родительской категории
* subcategory\_name (TEXT): Название подкатегории

**5. Таблица продаж (sales)**

* sale\_id (INT): Уникальный идентификатор продажи
* store\_id (INT): Идентификатор магазина
* product\_id (INT): Идентификатор товара
* customer\_id (INT, может быть NULL): Идентификатор клиента
* sale\_date (DATE): Дата продажи
* quantity (FLOAT): Количество проданных единиц
* unit\_price (FLOAT): Фактическая цена продажи за единицу
* discount (FLOAT): Размер скидки
* total\_amount (FLOAT): Итоговая сумма продажи
* payment\_type (TEXT): Тип оплаты (наличные, карта, онлайн)
* promo\_id (INT, может быть NULL): Идентификатор промо-акции

**6. Таблица клиентов (customers)**

* customer\_id (INT): Уникальный идентификатор клиента
* first\_name (TEXT): Имя
* last\_name (TEXT): Фамилия
* email (TEXT): Электронная почта
* phone (TEXT): Телефон
* registration\_date (DATE): Дата регистрации
* loyalty\_level (TEXT): Уровень лояльности
* city (TEXT): Город
* birth\_date (DATE): Дата рождения
* gender (TEXT): Пол

**7. Таблица поставщиков (suppliers)**

* supplier\_id (INT): Уникальный идентификатор поставщика
* supplier\_name (TEXT): Название поставщика
* contact\_person (TEXT): Контактное лицо
* email (TEXT): Электронная почта
* phone (TEXT): Телефон
* country (TEXT): Страна
* rating (FLOAT): Рейтинг поставщика

**8. Таблица запасов (inventory)**

* inventory\_id (INT): Уникальный идентификатор записи инвентаря
* store\_id (INT): Идентификатор магазина
* product\_id (INT): Идентификатор товара
* quantity (FLOAT): Количество на складе
* last\_update (TIMESTAMP): Время последнего обновления
* min\_stock\_level (FLOAT): Минимальный уровень запаса
* max\_stock\_level (FLOAT): Максимальный уровень запаса

**9. Таблица промо-акций (promotions)**

* promo\_id (INT): Уникальный идентификатор акции
* promo\_name (TEXT): Название акции
* start\_date (DATE): Дата начала
* end\_date (DATE): Дата окончания
* promo\_type (TEXT): Тип акции (скидка, 2+1, и т.д.)
* discount\_amount (FLOAT): Размер скидки
* min\_purchase (FLOAT): Минимальная сумма покупки

**Функциональные требования к MVP**

1. **Обработка запросов на естественном языке:**
   * Понимание намерения пользователя
   * Извлечение ключевых параметров из запроса
   * Перевод в корректный SQL-запрос к базе данных
2. **Выполнение запросов к данным:**
   * Генерация оптимизированных SQL-запросов
   * Исполнение запросов в DuckDB
   * Обработка ошибок и ограничение сложных/долгих запросов
3. **Табличное представление данных:**
   * Форматирование результатов в читаемые таблицы
   * Возможность сортировки и фильтрации в интерфейсе
   * Добавление заголовков и пояснений
4. **Интерфейс пользователя:**
   * Поле ввода запроса
   * Отображение сгенерированного SQL
   * Вывод результатов в табличном формате

**Примеры запросов пользователей, которые система должна обрабатывать**

1. "Покажи топ-10 товаров по продажам за последний месяц"
2. "Какие магазины имеют наибольшую выручку в категории 'Молочные продукты'?"
3. "Сравни продажи по регионам за первый квартал этого года"
4. "Как изменилась средняя маржа по категориям товаров за последние 3 месяца?"
5. "Какие товары чаще всего покупают вместе с хлебом?"
6. "Покажи динамику продаж мороженого по месяцам за прошлый год"
7. "Какие клиенты потратили больше всего в прошлом месяце и что они покупали?"
8. "Какие категории товаров приносят наибольшую прибыль в магазинах формата мини-маркет?"
9. "Сравни эффективность промо-акций за последний квартал"
10. "У каких товаров критический уровень запасов в магазинах Москвы?"

**Структура проекта**

graphql

retail\_data\_assistant/

├── app.py # Основной файл Streamlit приложения

├── llm\_processor.py # Модуль обработки запросов через LLM API

├── metadata/

│ ├── schema.json # Структура таблиц и полей

│ ├── dictionary.json # Словарь бизнес-терминов

│ └── query\_examples.json # Примеры типовых запросов

├── data\_manager/

│ ├── db\_initializer.py # Создание и инициализация DuckDB

│ ├── query\_executor.py # Выполнение SQL-запросов

│ └── formatter.py # Форматирование результатов

├── data/

│ ├

│ └── \*.csv # Сгенерированные CSV-файлы с данными

├── utils/

│ ├── config.py # Конфигурация приложения

│ └── logger.py # Логирование

├── requirements.txt # Зависимости проекта

└── README.md # Документация

**План разработки MVP**

1. **Этап 1: Подготовка данных**
   * Генерация набора данных для всех таблиц
   * Подготовка метаданных (схема, словарь, примеры)
2. **Этап 2: Базовый бэкенд**
   * Настройка DuckDB и загрузка данных
   * Реализация простых SQL-запросов к данным
3. **Этап 3: Интеграция с LLM**
   * Подключение к API выбранной LLM: gpt4o-mini (ключ положить в .streamlit/secerts.toml)
   * Реализация преобразования запросов
   * Примеры типовых запросов для контекста
4. **Этап 4: Интерфейс Streamlit**
   * Создание базового UI
   * Интеграция компонентов

**Дополнительные требования**

1. **Производительность:**
   * Ограничение времени выполнения запросов до 10 секунд
   * Ограничение объема возвращаемых данных (не более 1000 строк)
2. **Метаданные:**
   * Не менее 20 бизнес-терминов в словаре соответствий
   * Не менее 10 примеров типовых запросов разной сложности
3. **Генерация данных:**
   * Реалистичные, но синтетические данные
   * Объем данных: ~30 продуктов, ~5 магазинов, ~100 записей продаж
   * Временной диапазон: последние 2 недели