Привет! Это Хабр Карьера и Хекслет

Поздравляем, вы добрались до третьего тестового! Это задание самое сложное, но и самое интересное. Главное — не стремитесь к идеалу, лучше прислать решение, чем вовсе его не закончить. Даже если сомневаетесь в некоторых деталях, отправляйте результаты.

Для выполнения тестового вам может пригодиться курс **Python: Pandas**.

Попробуйте сначала разобраться самостоятельно. Если где-то возникнут сложности, курс поможет заполнить пробелы. А если задание пока кажется слишком сложным, запланируйте изучение материалов на ближайшие дни, пока они доступны бесплатно.

📎 Если пропустили предыдущие тестовые, вот ссылки, чтобы наверстать:

- **Первое тестовое**
- **Второе тестовое**

Дедлайн для всех тестовых — 13 декабря, 23:59.

И помните: только те, кто сдадут все три задания, получат шанс выиграть 12 месяцев бесплатного обучения на Хекслете. Удачи!

Описание задания

Разработайте сервис, который анализирует историю покупок всех клиентов и определяет, на какие товары предложить скидки конкретному пользователю.

Цель — рекомендовать пользователю товар, который часто покупают другие клиенты со схожими предпочтениями.

Что нужно учитывать

1. **Сопутствующие товары**:

- Анализируйте, какие товары чаще всего покупают вместе.
- Рекомендуйте пользователю один товар, не входящий в его предыдущие покупки, но популярный у покупателей со схожими предпочтениями.

2. **Популярность позиции**: если рекомендаций несколько, выбирайте самый популярный товар на основе общей частоты покупок.
3. **Асинхронность**: генерация рекомендаций должна быть асинхронной.

Пример структуры базы данных

Таблица "Покупки" (UserPurchases):

Поле	Тип	Описание
id	UUID	Уникальный идентификатор покупки
user_id	UUID	Идентификатор пользователя
item_id	UUID	Идентификатор товара
category	String	Категория товара
purchase_date	Timestamp	Дата покупки

Таблица "Товары" (Items):

Поле	Тип	Описание
id	UUID	Уникальный идентификатор товара
name	String	Название товара
category	String	Категория товара

Таблица "Рекомендации" (Recommendations):

Поле	Тип	Описание
id	UUID	Уникальный идентификатор рекомендации
user_id	UUID	Идентификатор пользователя
item_id	UUID	Идентификатор товара

Пример работы API

1. **Добавление покупки пользователя:**

```
POST /purchases
Content-Type: application/json

{
    "user_id": "user123",
    "cart": [
        {"item_id": "item123", "category": "fruits"},
        {"item_id": "item456", "category": "dairy"}
    ]
}
```

Ответ:

```
{
    "status": "purchases_added"
}
```

2. **Генерация рекомендаций**:

```
POST /generate_recommendations
Content-Type: application/json

{
    "user_id": "user123"
}
```

Ответ:

```
{
    "status": "recommendations_generation_started"
}
```

3. **Получение персонализированной рекомендации:**

```
GET /recommendations?user_id=user123
```

Ответ:

```
{
    "recommendations": [
        {
            "item_id": "item789",
            "category": "snacks"
        }
    ]
}
```

Алгоритм рекомендации

1. **Матрица совместной покупки**: постройте матрицу товаров, где каждая ячейка показывает, как часто два товара покупают вместе.
2. **Рекомендация новых товаров**: для каждого товара из истории покупок пользователя найдите товары с наибольшей связностью, которых нет в его истории покупок.
3. **Учет популярности**: если таких товаров несколько, то в первую очередь нужно рекомендовать более популярный товар на основе общего количества покупок.
4. **Генерация рекомендаций:** на основе анализа покупок выберите одну позицию, которую наиболее вероятно купит пользователь, и добавьте ее в таблицу с рекомендациями.

Требования

1. **Технологии**:

- FastAPI для реализации API.
- PostgreSQL для хранения данных.
- SQLAlchemy для работы с базой данных.
- asyncio для асинхронной генерации рекомендаций.

2. **Функциональные требования**:

- Асинхронная генерация рекомендаций.
- Учет истории покупок и сопутствующих товаров.

3. **Документация**:

- README с инструкцией по запуску и примерами запросов.

4. **Тестирование**:

- Unit-тесты для проверки API и алгоритмов рекомендации.