**День 3 - 8588a611c**

**User:**

Ты учитель и ментор, учи меня по плану "detailed\_plan\_days\_1". Начни с третьего дня.

**Abacus AI ChatLLM:**

Отлично! Начинаем с третьего дня. Сегодня мы изучаем \*\*базовый SQL\*\* - это фундаментальный навык для любого аналитика данных.  
  
## День 3: Базовый SQL (SELECT, FROM, WHERE, ORDER BY)  
  
### 🎯 Цель дня  
Освоить основы SQL и создать свою "кулинарную книгу" запросов для будущего использования.  
  
### ⏰ План на 3 часа  
  
#### 1. Теория (30-45 минут, 9:00-9:45)  
  
\*\*Что изучаем:\*\*  
- Основные команды SQL: `SELECT`, `FROM`, `WHERE`, `ORDER BY`  
- Синтаксис и правила написания запросов  
- Операторы сравнения (`>`, `=`, `<`, `!=`)  
- Логические операторы (`AND`, `OR`, `NOT`)  
  
\*\*Ключевые правила:\*\*  
- `WHERE` всегда идет перед `ORDER BY`  
- Каждый запрос заканчивается точкой с запятой `;`  
- SQL не чувствителен к регистру, но принято писать команды ЗАГЛАВНЫМИ  
  
\*\*Задание:\*\* Запишите в блокнот основной синтаксис:  
```sql  
SELECT column1, column2  
FROM table\_name  
WHERE condition  
ORDER BY column1 ASC/DESC;  
```  
  
#### 2. Практика (90-120 минут, 9:45-11:45)  
  
\*\*Шаг 1 (30 минут): Настройка базы данных\*\*  
  
Для macOS рекомендую SQLite (уже

установлен):  
  
1. Скачайте тестовую базу данных Chinook: [chinook.db](https://www.sqlitetutorial.net/wp-content/uploads/2018/03/chinook.zip)  
2. Откройте терминал и перейдите в папку с файлом  
3. Запустите: `sqlite3 chinook.db`  
4. Проверьте таблицы: `.tables`  
  
\*\*Шаг 2 (30 минут): Базовые запросы\*\*  
  
Выполните эти 5 запросов:  
  
```sql  
-- 1. Посмотреть все данные из таблицы tracks  
SELECT \* FROM tracks LIMIT 10;  
  
-- 2. Выбрать только название и цену треков  
SELECT Name, UnitPrice FROM tracks;  
  
-- 3. Найти треки дороже $1  
SELECT Name, UnitPrice FROM tracks WHERE UnitPrice > 1.0;  
  
-- 4. Отсортировать треки по цене (по убыванию)  
SELECT Name, UnitPrice FROM tracks ORDER BY UnitPrice DESC;  
  
-- 5. Комбинированный запрос  
SELECT Name, UnitPrice FROM tracks   
WHERE UnitPrice > 0.99   
ORDER BY UnitPrice ASC;  
```  
  
\*\*Шаг 3 (30-60 минут): Практика на 10 примерах\*\*  
  
Попробуйте самостоятельно написать запросы для:  
  
1. Все треки длительностью больше 300000 миллисекунд  
2. Треки, отсортированные по названию (алфавитный порядок)  
3. Треки

с ценой от 0.99 до 1.99  
4. Первые 5 самых дорогих треков  
5. Треки, в названии которых есть слово "Love" (используйте `LIKE '%Love%'`)  
  
#### 3. Фиксация результатов (15-30 минут, 11:45-12:15)  
  
Создайте файл `SQL\_cookbook.md` в вашем репозитории и добавьте:  
  
```markdown  
# SQL Cookbook - День 3  
  
## Базовые команды  
  
### SELECT - выбор данных  
```sql  
SELECT column1, column2 FROM table\_name;  
```  
  
### WHERE - фильтрация  
```sql  
SELECT \* FROM tracks WHERE UnitPrice > 1.0;  
```  
  
### ORDER BY - сортировка  
```sql  
SELECT \* FROM tracks ORDER BY UnitPrice DESC;  
```  
  
## Мои 10 запросов  
  
1. \*\*Все треки дороже $1:\*\*  
 ```sql  
 SELECT Name, UnitPrice FROM tracks WHERE UnitPrice > 1.0;  
 ```  
 \*Результат: X треков найдено\*  
  
[Добавьте остальные запросы с комментариями]  
  
## Что я понял сегодня  
- SQL читается слева направо  
- WHERE фильтрует ДО сортировки  
- Всегда проверяю результат на логичность  
  
## Ошибки, которые делал  
- Забывал точку с запятой  
- Путал порядок WHERE и ORDER BY  
```  
  
### 🤔 Вопросы для самопроверки:  
1. Чем

отличается `ASC` от `DESC`?  
2. Что произойдет, если написать `WHERE` после `ORDER BY`?  
3. Как найти треки с ценой точно $0.99?  
  
### 📝 Домашнее задание  
Попробуйте написать запрос, который найдет 3 самых дешевых трека и отсортирует их по названию.  
  
---  
  
\*\*Готовы начать? Какой у вас опыт с SQL? Есть ли уже установленная база данных или начнем с установки?\*\*

**User:**

Готов начать. Опыта работы с SQL не имею. База данных по-моему уже установлена. Но точно не помню.

**Abacus AI ChatLLM:**

Отлично! Давайте проверим, что у вас есть, и начнем с самого начала.  
  
## Проверяем SQLite на macOS  
  
SQLite обычно уже установлен на macOS. Давайте проверим:  
  
1. \*\*Откройте Terminal\*\* (Finder → Программы → Утилиты → Terminal)  
2. \*\*Введите команду:\*\*  
 ```bash  
 sqlite3 --version  
 ```  
  
Если увидите что-то вроде `3.39.5 2022-10-14...` - отлично, SQLite есть!  
  
Если команда не найдена, установим через Homebrew:  
```bash  
brew install sqlite3  
```  
  
## Скачиваем тестовую базу данных  
  
Нам нужна база данных для практики. Сделаем это пошагово:  
  
1. \*\*Создайте папку для проекта:\*\*  
 ```bash  
 mkdir ~/sql\_practice  
 cd ~/sql\_practice  
 ```  
  
2. \*\*Скачайте файл chinook.db\*\* - это популярная тестовая база с данными о музыке:  
 - Перейдите по ссылке: https://www.sqlitetutorial.net/wp-content/uploads/2018/03/chinook.zip  
 - Скачайте и разархивируйте файл `chinook.db` в папку `sql\_practice`  
  
3. \*\*Или создадим простую базу сами\*\* (если скачивание не работает):  
 ```bash  
 sqlite3 test.db  
 ```  
  
## Первый запуск  
  
Когда

файл готов, запустите:  
```bash  
cd ~/sql\_practice  
sqlite3 chinook.db  
```  
  
Должно появиться приглашение: `sqlite>`  
  
\*\*Проверьте, что все работает:\*\*  
```sql  
.tables  
```  
  
Должен показать список таблиц: albums, artists, tracks и другие.  
  
---  
  
\*\*Напишите, что получилось на каждом шаге! Если что-то не работает - разберем вместе. Как только база будет готова, начнем писать первые запросы! 🚀\*\*

**User:**

Все получилось. Но базу SQL я установил не в рабочую папку "data-analyst-journey", а в другое место. Может переместить ее в папку "data-analyst-journey"?

**Abacus AI ChatLLM:**

Отлично! Да, давайте переместим базу данных в правильное место - это хорошая практика держать все материалы проекта в одном репозитории.  
  
## Перемещаем базу данных в проект  
  
1. \*\*Сначала найдем, где сейчас находится ваша база:\*\*  
 ```bash  
 pwd  
 ```  
 Это покажет текущую папку. Запомните путь.  
  
2. \*\*Перейдем в папку проекта:\*\*  
 ```bash  
 cd ~/data-analyst-journey  
 ```  
  
3. \*\*Создадим папку для SQL-практики:\*\*  
 ```bash  
 mkdir sql  
 cd sql  
 ```  
  
4. \*\*Переместим файл базы данных:\*\*  
 ```bash  
 mv ~/sql\_practice/chinook.db .  
 ```  
 (Точка в конце означает "в текущую папку")  
  
5. \*\*Проверим, что файл на месте:\*\*  
 ```bash  
 ls -la  
 ```  
 Должны увидеть `chinook.db`  
  
6. \*\*Запустим базу из новой папки:\*\*  
 ```bash  
 sqlite3 chinook.db  
 ```  
  
7. \*\*Проверим таблицы:\*\*  
 ```sql  
 .tables  
 ```  
  
## Структура проекта теперь:  
```  
data-analyst-journey/  
├── sql/  
│ └── chinook.db  
├── week1/  
├── python/  
├── reports/  
└── goals.md  
```  
  
\*\*Готово! Теперь все в одном месте. Можете удалить старую

папку:\*\*  
```bash  
rm -rf ~/sql\_practice  
```  
  
---  
  
\*\*Все получилось? Если да, то теперь мы готовы к первым SQL-запросам! 🎯\*\*

**User:**

Получилось?

**Abacus AI ChatLLM:**

Отлично! 🎉 Теперь у нас все настроено правильно и мы готовы к изучению SQL.  
  
## Начинаем практику - первые SQL запросы  
  
Вы сейчас в sqlite с базой chinook.db. Давайте начнем с самых простых команд:  
  
### 1. Изучаем структуру базы  
  
\*\*Посмотрим, какие таблицы есть:\*\*  
```sql  
.tables  
```  
  
\*\*Посмотрим структуру таблицы tracks:\*\*  
```sql  
.schema tracks  
```  
  
### 2. Первый запрос - посмотреть данные  
  
\*\*Выберем первые 5 записей из таблицы tracks:\*\*  
```sql  
SELECT \* FROM tracks LIMIT 5;  
```  
  
Что вы видите? Должны появиться колонки: TrackId, Name, AlbumId, MediaTypeId, GenreId, Composer, Milliseconds, Bytes, UnitPrice.  
  
### 3. Выбираем конкретные колонки  
  
\*\*Посмотрим только названия треков и цены:\*\*  
```sql  
SELECT Name, UnitPrice FROM tracks LIMIT 10;  
```  
  
---  
  
\*\*Попробуйте эти команды и напишите:\*\*  
1. Сколько всего таблиц в базе?  
2. Какие колонки есть в таблице tracks?  
3. Какая цена у первого трека в списке?  
  
Как только выполните эти запросы, продолжим с фильтрацией и сортировкой! 🚀

**User:**

Получилось?

**Abacus AI ChatLLM:**

Отлично! Вижу, что вы успешно подключились к базе данных и видите все таблицы. Теперь давайте выполним первые запросы.  
  
## Выполняем первые SQL запросы  
  
Вы сейчас в sqlite с приглашением `sqlite>`. Давайте пошагово:  
  
### 1. Изучаем структуру таблицы tracks  
```sql  
.schema tracks  
```  
  
### 2. Смотрим первые записи  
```sql  
SELECT \* FROM tracks LIMIT 5;  
```  
  
### 3. Выбираем только название и цену  
```sql  
SELECT Name, UnitPrice FROM tracks LIMIT 10;  
```  
  
---  
  
\*\*Выполните эти три команды по очереди и скажите:\*\*  
  
1. \*\*После `.schema tracks`\*\* - какие колонки вы видите в таблице?  
2. \*\*После `SELECT \* FROM tracks LIMIT 5;`\*\* - сколько колонок показалось и какая цена у первого трека?  
3. \*\*После `SELECT Name, UnitPrice FROM tracks LIMIT 10;`\*\* - какие названия треков вы видите?  
  
Просто скопируйте результат или опишите, что получилось. Это поможет убедиться, что все работает правильно, и мы сможем перейти к более интересным запросам с фильтрацией! 🎯

**User:**

1. Вижу колонки: [TrackId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,  
 [Name] NVARCHAR(200) NOT NULL,  
 [AlbumId] INTEGER,  
 [MediaTypeId] INTEGER NOT NULL,  
 [GenreId] INTEGER,  
 [Composer] NVARCHAR(220),  
 [Milliseconds] INTEGER NOT NULL,  
 [Bytes] INTEGER,  
 [UnitPrice] NUMERIC(10,2) NOT NULL,  
 FOREIGN KEY ([AlbumId]) REFERENCES "albums" ([AlbumId])   
 ON DELETE NO ACTION ON UPDATE NO ACTION,  
 FOREIGN KEY ([GenreId]) REFERENCES "genres" ([GenreId])   
 ON DELETE NO ACTION ON UPDATE NO ACTION,  
 FOREIGN KEY ([MediaTypeId]) REFERENCES "media\_types" ([MediaTypeId])   
 ON DELETE NO ACTION ON UPDATE NO ACTION  
  
2. Количество колонок - 5. Цена первого трека - 0.99  
3. Названия треков - Balls to the Wall|0.99  
Fast As a Shark|0.99  
Restless and Wild|0.99  
Princess of the Dawn|0.99  
Put The Finger On You|0.99  
Let's Get It Up|0.99  
Inject The Venom|0.99  
Snowballed|0.99  
Evil Walks

**Abacus AI ChatLLM:**

Отлично! 🎉 Вы успешно выполнили первые SQL запросы. Теперь переходим к более интересной части - \*\*фильтрации и сортировке\*\*.  
  
## Изучаем WHERE и ORDER BY  
  
### 4. Фильтрация данных (WHERE)  
  
\*\*Найдем треки дороже $1:\*\*  
```sql  
SELECT Name, UnitPrice FROM tracks WHERE UnitPrice > 1.0;  
```  
  
\*\*Найдем треки с точной ценой $0.99:\*\*  
```sql  
SELECT Name, UnitPrice FROM tracks WHERE UnitPrice = 0.99 LIMIT 5;  
```  
  
### 5. Сортировка данных (ORDER BY)  
  
\*\*Отсортируем треки по цене (от дешевых к дорогим):\*\*  
```sql  
SELECT Name, UnitPrice FROM tracks ORDER BY UnitPrice ASC LIMIT 10;  
```  
  
\*\*Отсортируем по цене (от дорогих к дешевым):\*\*  
```sql  
SELECT Name, UnitPrice FROM tracks ORDER BY UnitPrice DESC LIMIT 10;  
```  
  
### 6. Комбинируем WHERE и ORDER BY  
  
\*\*Найдем треки дороже $1 и отсортируем по цене:\*\*  
```sql  
SELECT Name, UnitPrice FROM tracks   
WHERE UnitPrice > 1.0   
ORDER BY UnitPrice DESC;  
```  
  
---  
  
\*\*Выполните эти запросы и ответьте:\*\*  
  
1. \*\*Сколько треков дороже $1?\*\* (посчитайте строки в результате первого запроса)  
2.

\*\*Какая самая высокая цена трека?\*\* (из запроса с DESC)  
3. \*\*Какой трек самый дорогой?\*\* (название из того же запроса)  
  
После этого перейдем к созданию вашей "кулинарной книги" SQL запросов! 📚

**User:**

Можно это делать не в командной строке а в чем-то более удобном?

**Abacus AI ChatLLM:**

Конечно! Командная строка не очень удобна для изучения SQL. Давайте установим что-то более дружелюбное.  
  
## Варианты для macOS:  
  
### 1. \*\*DB Browser for SQLite\*\* (рекомендую для начинающих)  
- Бесплатный, простой интерфейс  
- Скачать: https://sqlitebrowser.org/dl/  
- Или через Homebrew: `brew install --cask db-browser-for-sqlite`  
  
### 2. \*\*DBeaver\*\* (более профессиональный)  
- Бесплатный, поддерживает много баз данных  
- Скачать: https://dbeaver.io/download/  
- Или через Homebrew: `brew install --cask dbeaver-community`  
  
### 3. \*\*VS Code с расширением SQLite\*\*  
- Если у вас уже есть VS Code  
- Установите расширение "SQLite Viewer"  
  
## Рекомендую DB Browser for SQLite  
  
\*\*Установка:\*\*  
```bash  
brew install --cask db-browser-for-sqlite  
```  
  
\*\*Как использовать:\*\*  
1. Откройте DB Browser for SQLite  
2. File → Open Database → выберите ваш `chinook.db`  
3. Перейдите на вкладку "Execute SQL"  
4. Пишите запросы в верхнем окне, результаты внизу  
  
\*\*Преимущества:\*\*  
- Видите структуру таблиц слева  
- Автодополнение SQL  
- Красивое

отображение результатов  
- Можете сохранять запросы  
  
---  
  
\*\*Какой вариант предпочитаете? Или установим DB Browser и продолжим изучение в более удобном интерфейсе?\*\* 🖥️

**User:**

У меня есть VS Code

**Abacus AI ChatLLM:**

Отлично! VS Code - отличный выбор. Давайте настроим его для работы с SQL.  
  
## Настройка VS Code для SQL  
  
### 1. Установите расширение SQLite  
1. Откройте VS Code  
2. Нажмите `Cmd+Shift+X` (Extensions)  
3. Найдите и установите \*\*"SQLite Viewer"\*\* от alexcvzz  
4. Также установите \*\*"SQL Highlight"\*\* для подсветки синтаксиса  
  
### 2. Откройте вашу базу данных  
1. В VS Code откройте папку `data-analyst-journey`  
2. В папке `sql` вы увидите файл `chinook.db`  
3. Щелкните правой кнопкой на `chinook.db` → \*\*"Open Database"\*\*  
4. Слева появится панель SQLITE EXPLORER с таблицами  
  
### 3. Создайте файл для SQL запросов  
1. В папке `sql` создайте новый файл: `day3\_practice.sql`  
2. VS Code автоматически подсветит SQL синтаксис  
  
### 4. Выполнение запросов  
1. Напишите запрос в файле `.sql`  
2. Выделите запрос  
3. Нажмите `Cmd+Shift+P` → найдите \*\*"SQLite: Run Query"\*\*  
4. Результат появится в новой вкладке  
  
## Первый тест  
  
Создайте файл `day3\_practice.sql` и напишите:  
  
```sql  
-- Мой первый запрос в VS Code  
SELECT Name, UnitPrice FROM

tracks LIMIT 5;  
```  
  
Выделите этот запрос и выполните через Command Palette.  
  
---  
  
\*\*Получилось настроить? Если да, то продолжим изучение SQL в удобном интерфейсе! 🚀\*\*

**User:**

Расширения установил, базу открыл, файл для SQL запросов создал.  
Остановился на 4-м пункте. Не понял где и что писать.

**Abacus AI ChatLLM:**

Отлично! Давайте разберем 4-й пункт пошагово.  
  
## Как выполнять SQL запросы в VS Code  
  
### Шаг 1: Напишите запрос  
В вашем файле `day3\_practice.sql` напишите:  
  
```sql  
-- Мой первый запрос в VS Code  
SELECT Name, UnitPrice FROM tracks LIMIT 5;  
```  
  
### Шаг 2: Выделите запрос  
- Выделите \*\*весь запрос\*\* (от SELECT до точки с запятой)  
- Или поставьте курсор в любое место запроса  
  
### Шаг 3: Выполните запрос  
\*\*Способ 1 (через Command Palette):\*\*  
1. Нажмите `Cmd+Shift+P` (откроется Command Palette)  
2. Начните печатать "sqlite run"  
3. Выберите \*\*"SQLite: Run Query"\*\*  
  
\*\*Способ 2 (через правую кнопку):\*\*  
1. Щелкните правой кнопкой на выделенном запросе  
2. Выберите \*\*"Run Query"\*\*  
  
### Шаг 4: Смотрите результат  
- Откроется новая вкладка с результатами  
- Вы увидите таблицу с данными  
  
---  
  
\*\*Попробуйте сейчас выполнить этот простой запрос. Что получилось? Видите ли результаты в новой вкладке?\*\* 🎯

**User:**

Все получилось

**Abacus AI ChatLLM:**

Отлично! 🎉 Теперь у нас есть удобная среда для изучения SQL. Продолжаем практику!  
  
## Практикуем WHERE и ORDER BY  
  
Добавьте в ваш файл `day3\_practice.sql` следующие запросы и выполняйте их по одному:  
  
```sql  
-- Мой первый запрос в VS Code  
SELECT Name, UnitPrice FROM tracks LIMIT 5;  
  
-- Запрос 2: Треки дороже $1  
SELECT Name, UnitPrice FROM tracks WHERE UnitPrice > 1.0;  
  
-- Запрос 3: Треки с точной ценой $0.99  
SELECT Name, UnitPrice FROM tracks WHERE UnitPrice = 0.99 LIMIT 5;  
  
-- Запрос 4: Сортировка по цене (от дешевых к дорогим)  
SELECT Name, UnitPrice FROM tracks ORDER BY UnitPrice ASC LIMIT 10;  
  
-- Запрос 5: Сортировка по цене (от дорогих к дешевым)  
SELECT Name, UnitPrice FROM tracks ORDER BY UnitPrice DESC LIMIT 10;  
  
-- Запрос 6: Комбинируем WHERE и ORDER BY  
SELECT Name, UnitPrice FROM tracks  
WHERE UnitPrice > 1.0  
ORDER BY UnitPrice DESC;  
```  
  
### Задания для самопроверки:  
  
Выполните каждый запрос и ответьте:  
  
1. \*\*Запрос 2:\*\* Сколько треков дороже $1? (посчитайте строки)  
2. \*\*Запрос 5:\*\* Какая самая

высокая цена трека?  
3. \*\*Запрос 6:\*\* Какой самый дорогой трек в базе?  
  
---  
  
\*\*Выполните эти запросы и поделитесь результатами. После этого перейдем к более интересным задачам! 🚀\*\*

**User:**

Запрос 2: 213 треков  
Запрос 5: 1.99  
Запрос 6: Их 213 по цене 1.99

**Abacus AI ChatLLM:**

Отлично! 🎯 Вы правильно выполнили все запросы. Теперь переходим к более интересным задачам.  
  
## Практические задания для закрепления  
  
Попробуйте самостоятельно написать запросы для этих задач:  
  
### Задание 1: Работа с длительностью треков  
```sql  
-- Найдите треки длительностью больше 300000 миллисекунд (5 минут)  
-- Подсказка: используйте колонку Milliseconds  
```  
  
### Задание 2: Сортировка по названию  
```sql  
-- Отсортируйте треки по названию в алфавитном порядке  
-- Покажите первые 10  
```  
  
### Задание 3: Диапазон цен  
```sql  
-- Найдите треки с ценой от 0.99 до 1.50  
-- Подсказка: используйте AND или BETWEEN  
```  
  
### Задание 4: Поиск по названию  
```sql  
-- Найдите треки, в названии которых есть слово "Love"  
-- Подсказка: используйте LIKE '%Love%'  
```  
  
### Задание 5: Топ самых коротких треков  
```sql  
-- Найдите 5 самых коротких треков  
-- Отсортируйте по Milliseconds по возрастанию  
```  
  
---  
  
\*\*Попробуйте написать эти запросы самостоятельно. Если застрянете - подскажу! После выполнения создадим вашу "кулинарную книгу"

SQL запросов. 📚\*\*  
  
Какое задание попробуете первым?

**User:**

Мне нужно больше информации о том как формировать запросы SQL, чтобы знать как выполнять это задание. Ты можешь мне указать где взять эту информацию?

**Abacus AI ChatLLM:**

Конечно! Вам нужна теоретическая база перед практикой. Вот лучшие ресурсы для изучения SQL:  
  
## 📚 Рекомендуемые ресурсы  
  
### 1. \*\*W3Schools SQL Tutorial\*\* (лучший для начинающих)  
- \*\*Ссылка:\*\* https://www.w3schools.com/sql/  
- \*\*Что изучить сначала:\*\*  
 - SQL Intro  
 - SQL Syntax   
 - SQL SELECT  
 - SQL WHERE  
 - SQL ORDER BY  
 - SQL AND, OR, NOT  
 - SQL LIKE  
 - SQL BETWEEN  
  
### 2. \*\*SQLBolt\*\* (интерактивные уроки)  
- \*\*Ссылка:\*\* https://sqlbolt.com/  
- Пошаговые уроки с практикой прямо в браузере  
  
### 3. \*\*SQL Tutorial от SQLite\*\*  
- \*\*Ссылка:\*\* https://www.sqlitetutorial.net/  
- Специально для SQLite (как у нас)  
  
## 🎯 План изучения на сегодня  
  
\*\*Потратьте 30-45 минут на изучение:\*\*  
  
1. \*\*W3Schools:\*\* прочитайте разделы SELECT, WHERE, ORDER BY  
2. \*\*Запишите шпаргалку\*\* в файл `sql\_cheatsheet.md`:  
  
```markdown  
# SQL Шпаргалка  
  
## Основной синтаксис  
SELECT column1, column2 FROM table\_name WHERE condition ORDER BY column1;  
  
## Операторы сравнения  
- `=` равно  
- `>` больше   
- `<` меньше  
- `>=` больше или равно  
- `

<=` меньше или равно  
- `!=` не равно  
  
## Логические операторы  
- `AND` - оба условия должны быть истинными  
- `OR` - одно из условий должно быть истинным  
- `BETWEEN` - значение в диапазоне  
  
## Поиск по тексту  
- `LIKE '%текст%'` - содержит текст  
- `LIKE 'текст%'` - начинается с текста  
```  
  
\*\*После изучения теории возвращайтесь - разберем задания вместе! 📖\*\*  
  
Сколько времени потратите на изучение?

**User:**

Ответы на задание:  
SELECT Name, Milliseconds FROM tracks WHERE Milliseconds > 300000;  
SELECT Name FROM tracks ORDER BY Name ASC LIMIT 10;  
SELECT Name, UnitPrice FROM tracks WHERE UnitPrice BETWEEN 0.99 AND 1.50;  
SELECT Name FROM tracks WHERE Name LIKE '%Love%';  
SELECT Name, Milliseconds FROM tracks ORDER BY Milliseconds ASC LIMIT 5;

**Abacus AI ChatLLM:**

Отлично! 🎉 Все запросы написаны правильно! Вы быстро освоили основы SQL.  
  
## Проверим результаты  
  
Выполните свои запросы в VS Code и проверьте:  
  
1. \*\*Длинные треки:\*\* Сколько треков длиннее 5 минут нашлось?  
2. \*\*Алфавитный порядок:\*\* Какой трек первый в алфавитном порядке?  
3. \*\*Диапазон цен:\*\* Есть ли треки с ценой между 0.99 и 1.50?  
4. \*\*Поиск "Love":\*\* Сколько треков с "Love" в названии?  
5. \*\*Короткие треки:\*\* Какой самый короткий трек и сколько он длится?  
  
## Создаем SQL Cookbook  
  
Теперь создайте файл `SQL\_cookbook.md` в папке `sql` и добавьте:

# SQL Cookbook - День 3  
  
## Основной синтаксис  
  
SELECT column1, column2 FROM table\_name WHERE condition ORDER BY column1;  
  
## Операторы сравнения  
  
\* `=` равно  
  
\* `>` больше  
  
\* `<` меньше  
  
\* `>=` больше или равно  
  
\* `<=` меньше или равно  
  
\* `!=` не равно  
  
## Логические операторы  
  
\* `AND` - оба условия должны быть истинными  
  
\* `OR` - одно из условий должно быть истинным  
  
\* `BETWEEN` - значение в диапазоне  
  
## Поиск по тексту  
  
\* `LIKE '%текст%'` - содержит текст  
  
\* `LIKE 'текст%'` - начинается с текста  
  
## Мои запросы - День 3  
  
### 1\. Базовые запросы  
  
```sql  
-- Посмотреть структуру таблицы  
.schema tracks  
  
-- Выбрать все данные (первые 5 записей)  
SELECT \* FROM tracks LIMIT 5;  
  
-- Выбрать конкретные колонки  
SELECT Name, UnitPrice FROM tracks LIMIT 10;  
```  
  
### 2\. Фильтрация (WHERE)  
  
```sql  
-- Треки дороже $1  
SELECT Name, UnitPrice FROM tracks WHERE UnitPrice > 1.0;  
  
-- Треки с точной ценой  
SELECT Name, UnitPrice FROM tracks WHERE UnitPrice = 0.99;  
  
-- Длинные треки (больше 5 минут)  
SELECT Name, Milliseconds FROM tracks WHERE Milliseconds > 300000;  
  
-- Диапазон цен  
SELECT Name, UnitPrice FROM tracks WHERE UnitPrice BETWEEN 0.99 AND 1.50;  
  
-- Поиск по названию  
SELECT Name FROM tracks WHERE Name LIKE '%Love%';  
```  
  
### 3\. Сортировка (ORDER BY)  
  
```sql  
-- По цене (от дешевых к дорогим)  
SELECT Name, UnitPrice FROM tracks ORDER BY UnitPrice ASC LIMIT 10;  
  
-- По цене (от дорогих к дешевым)  
SELECT Name, UnitPrice FROM tracks ORDER BY UnitPrice DESC LIMIT 10;  
  
-- По названию (алфавитный порядок)  
SELECT Name FROM tracks ORDER BY Name ASC LIMIT 10;  
  
-- Самые короткие треки  
SELECT Name, Milliseconds FROM tracks ORDER BY Milliseconds ASC LIMIT 5;  
```  
  
### 4\. Комбинированные запросы  
  
```sql  
-- Дорогие треки, отсортированные по цене  
SELECT Name, UnitPrice FROM tracks   
WHERE UnitPrice > 1.0   
ORDER BY UnitPrice DESC;  
```  
  
## Что я понял сегодня  
  
\* SQL читается слева направо  
  
\* WHERE фильтрует данные ДО сортировки  
  
\* ORDER BY всегда идет в конце  
  
\* LIKE с % позволяет искать по части текста  
  
\* BETWEEN включает граничные значения  
  
## Полезные команды SQLite  
  
\* `.tables` - показать все таблицы  
  
\* `.schema table\_name` - показать структуру таблицы  
  
\* `.quit` - выйти из SQLite  
  
## Следующие шаги  
  
\* Изучить JOIN для связывания таблиц  
  
\* Освоить агрегатные функции (COUNT, SUM, AVG)  
  
\* Практиковать GROUP BY

Отлично! Я создал для вас шаблон \*\*SQL Cookbook\*\* - это будет ваша "кулинарная книга" запросов.  
  
## 📋 Что делать дальше:  
  
1. \*\*Выполните свои 5 запросов\*\* в VS Code и запишите результаты  
2. \*\*Скопируйте содержимое Cookbook\*\* в файл `SQL\_cookbook.md` в папке `sql`  
3. \*\*Добавьте свои наблюдения\*\* в раздел "Что я понял сегодня"  
  
## 🎯 Итоги Дня 3:  
  
✅ Настроили удобную среду (VS Code + SQLite)   
✅ Освоили базовые команды: SELECT, WHERE, ORDER BY   
✅ Написали 10+ SQL запросов   
✅ Создали SQL Cookbook для будущего использования   
  
## 📝 Домашнее задание:  
  
Попробуйте написать запрос: \*\*"Найти 3 самых дорогих трека, которые длятся меньше 4 минут, отсортированные по названию"\*\*  
  
---  
  
\*\*Поздравляю! Вы успешно завершили День 3. Готовы ли продолжить завтра с более сложными SQL темами (JOIN, агрегатные функции)? 🚀\*\*

**User:**

SELECT Name, UnitPrice FROM tracks WHERE Milliseconds < 240000 ORDER BY Name DESC LIMIT 4;

**Abacus AI ChatLLM:**

Почти правильно! 🎯 Но есть небольшая ошибка в логике запроса.  
  
## Разбор вашего запроса:  
  
```sql  
SELECT Name, UnitPrice FROM tracks   
WHERE Milliseconds < 240000   
ORDER BY Name DESC   
LIMIT 4;  
```  
  
\*\*Что делает ваш запрос:\*\*  
- ✅ Фильтрует треки короче 4 минут (240000 мс)  
- ❌ Сортирует по названию в \*\*обратном\*\* алфавитном порядке (DESC)  
- ❌ Показывает \*\*4\*\* трека вместо 3  
- ❌ \*\*Не сортирует по цене\*\* для поиска самых дорогих  
  
## Правильное решение:  
  
```sql  
SELECT Name, UnitPrice FROM tracks   
WHERE Milliseconds < 240000   
ORDER BY UnitPrice DESC, Name ASC   
LIMIT 3;  
```  
  
\*\*Логика:\*\*  
1. `WHERE Milliseconds < 240000` - треки короче 4 минут  
2. `ORDER BY UnitPrice DESC` - \*\*сначала\*\* сортируем по цене (самые дорогие первые)  
3. `, Name ASC` - \*\*потом\*\* по названию в алфавитном порядке  
4. `LIMIT 3` - берем только 3 записи  
  
## 🤔 Вопрос для понимания:  
Попробуйте выполнить оба запроса. Видите разницу в результатах?  
  
\*\*Отличная работа! Вы быстро осваиваете SQL. Готовы к следующему дню? 🚀\*\*