МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности

Кафедра компьютерных интеллектуальных технологий проектирования

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9

По дисциплине: «Формирование запросов к базе данных»

Тема: «Формирование запросов к базе данных»

Выполнил работу студент группы змИИВТ-241: Быкова А.О.

подпись, дата

Принял: Короленко В.В.

подпись, дата

Воронеж 2024

***Задание***

*Цель лабораторной работы:*

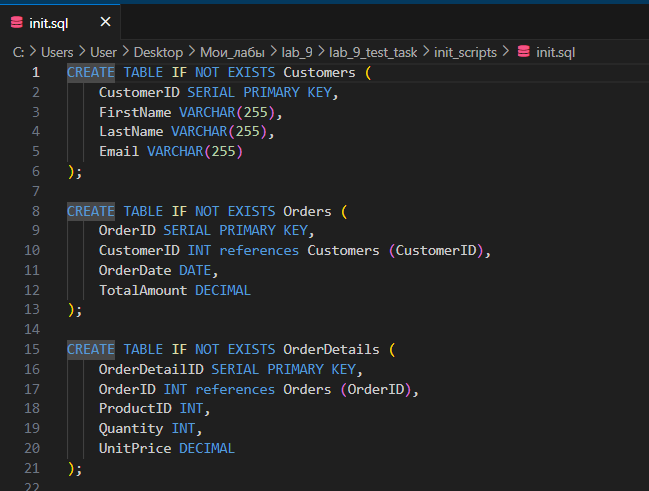
изучить синтаксис и основные команды для формирования запросов к базе данных, освоить процесс формирования SQL-запросов.

*Основные задачи:*

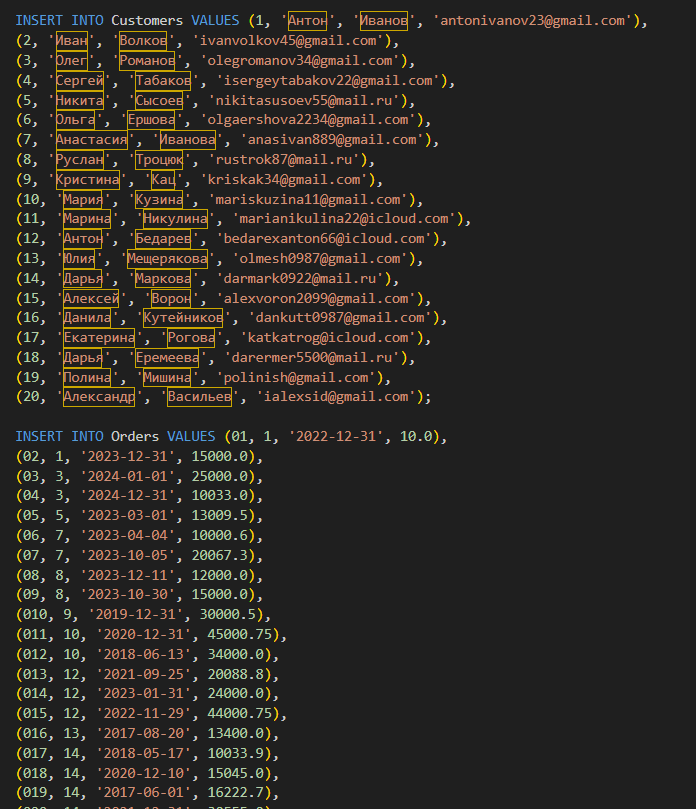
* Разработать структуру базы данных в СУБД Postgres.
* Наполнить базу данных данными.
* Написать ряд запросов к базе данных для отработки навыка формирования SQL-запросов.
* Научиться ставить задачу по формированию выборки необходимых данных из базы данных и решать её с помощью SQL-запросов.

*Ход работы*

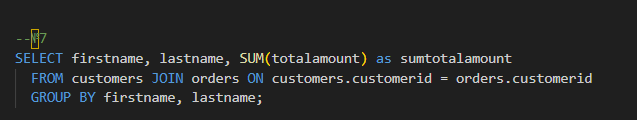
1. Сформировала SQL-запросы для создания таблиц в СУБД Postgres в соответствии с заданной структурой:



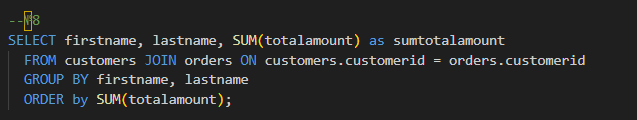
1. Наполнила базу данных данными:

**

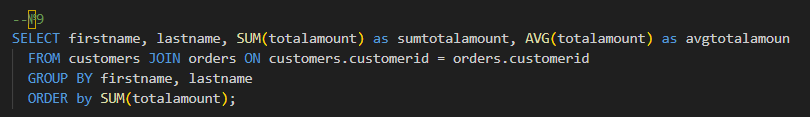
1. Сформировала SQL-запрос, который возвращает список клиентов (имя и фамилия) и суммарную стоимость заказов каждого клиента­­:

**

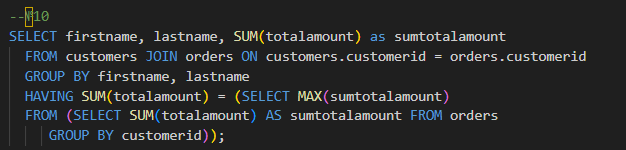
1. Отсортировала полученный список по убыванию суммарной стоимости заказов клиента:

**

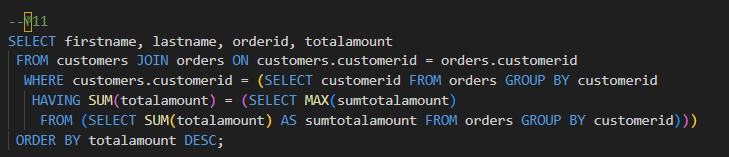
1. Добавила столбец со средней суммарной стоимостью заказов:

**

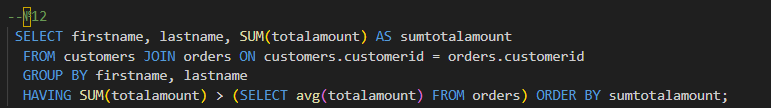
1. Вывела список клиентов с наибольшей суммарной стоимостью заказов (имя, фамилия, стоимость):



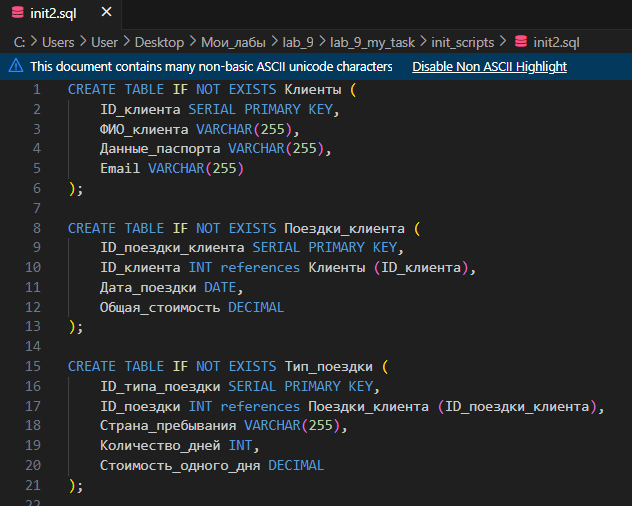
1. Для каждого клиента с наибольшей суммарной стоимостью заказов вывела список его заказов (номер заказа и стоимость) в порядке возрастания стоимости заказа:



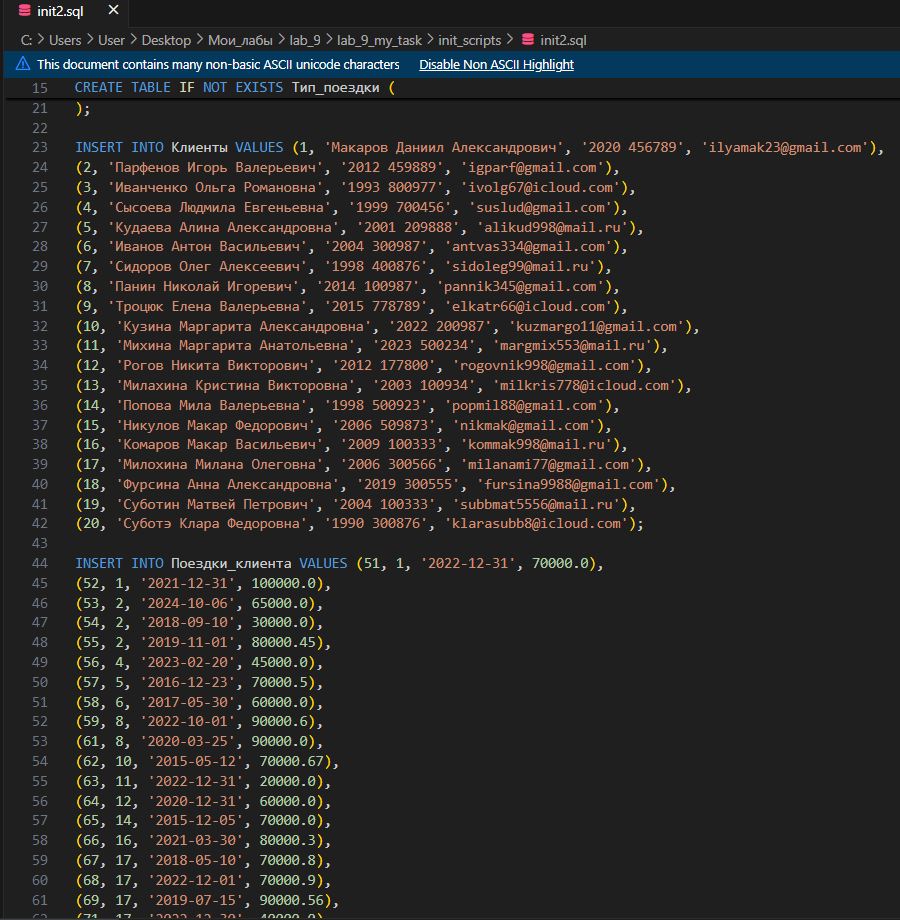
1. Вывела только тех клиентов, у которых суммарная стоимость заказов превышает среднюю суммарную стоимость заказов клиентов:

**

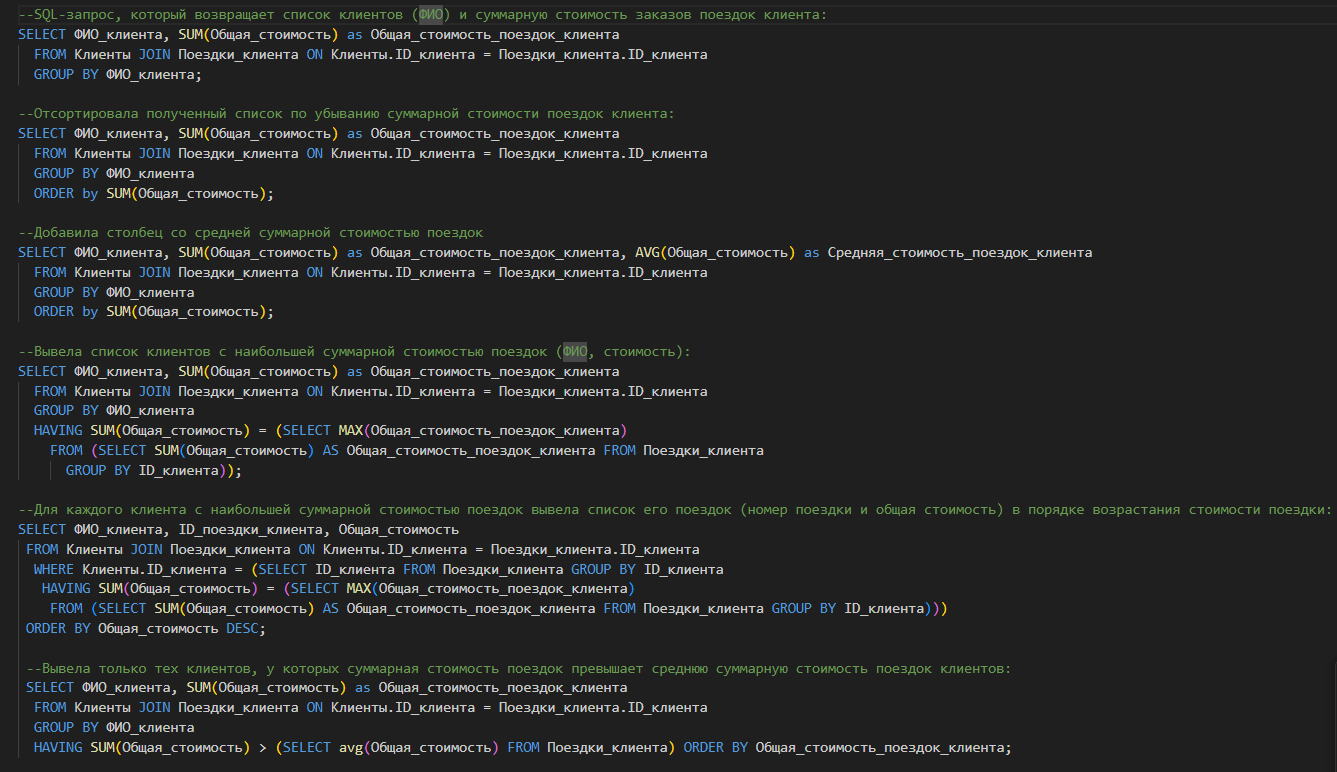
1. Сохранила SQL-запросы в соответствующем файле init.sql
2. Сформулировала задачу, аналогичную изложенной выше, на основе базы данных, разработанной в предыдущей лабораторной работе в соответствии с индивидуальным заданием. Сформировала SQL-запросы для создания таблиц в СУБД Postgres:

**

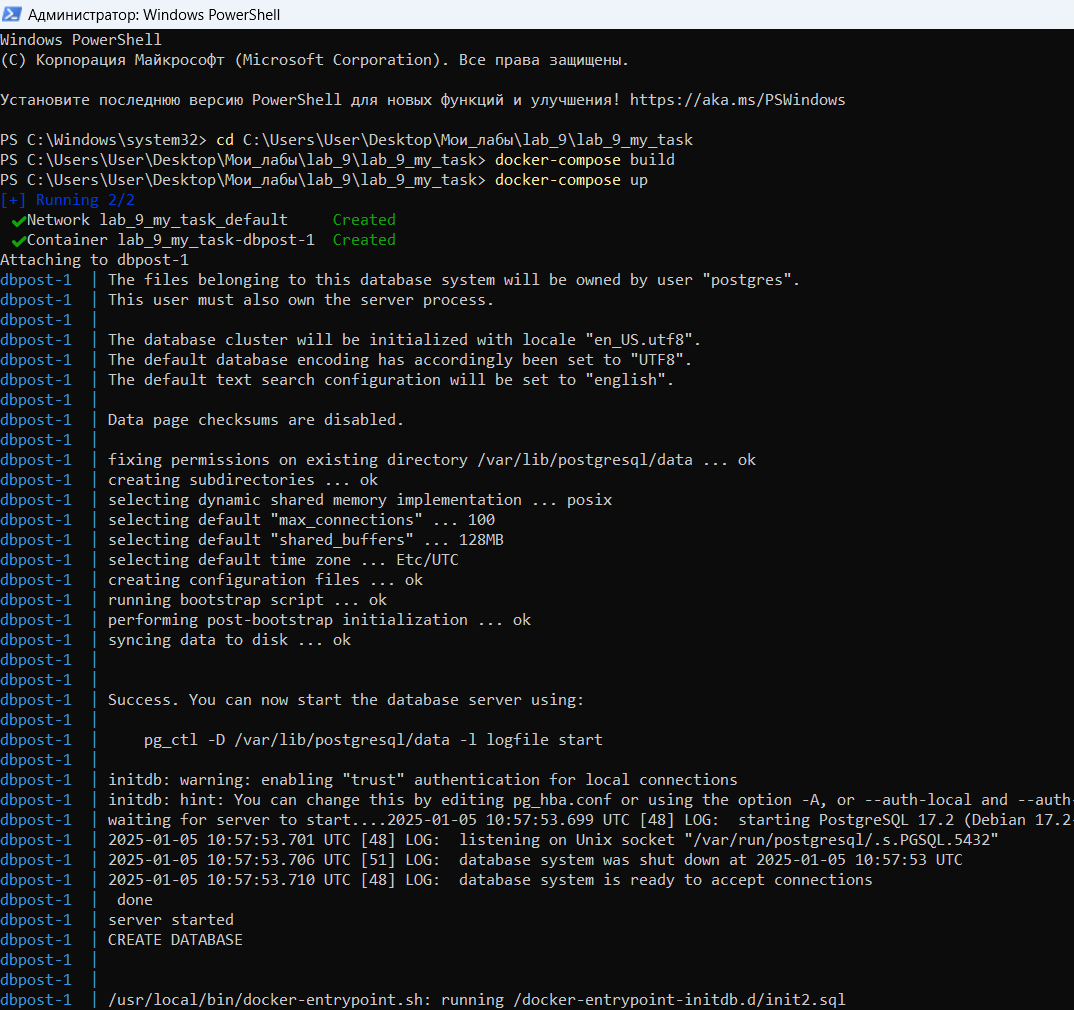
1. Наполнила базу данных данными:

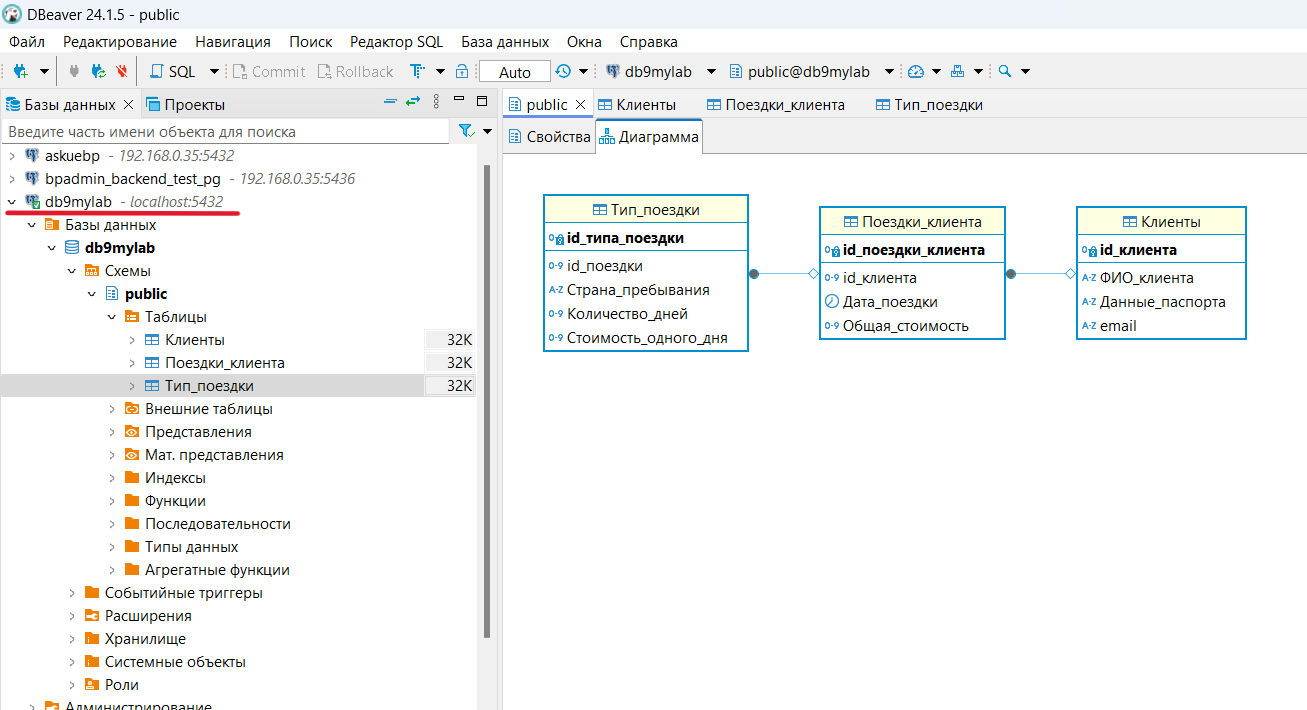
**

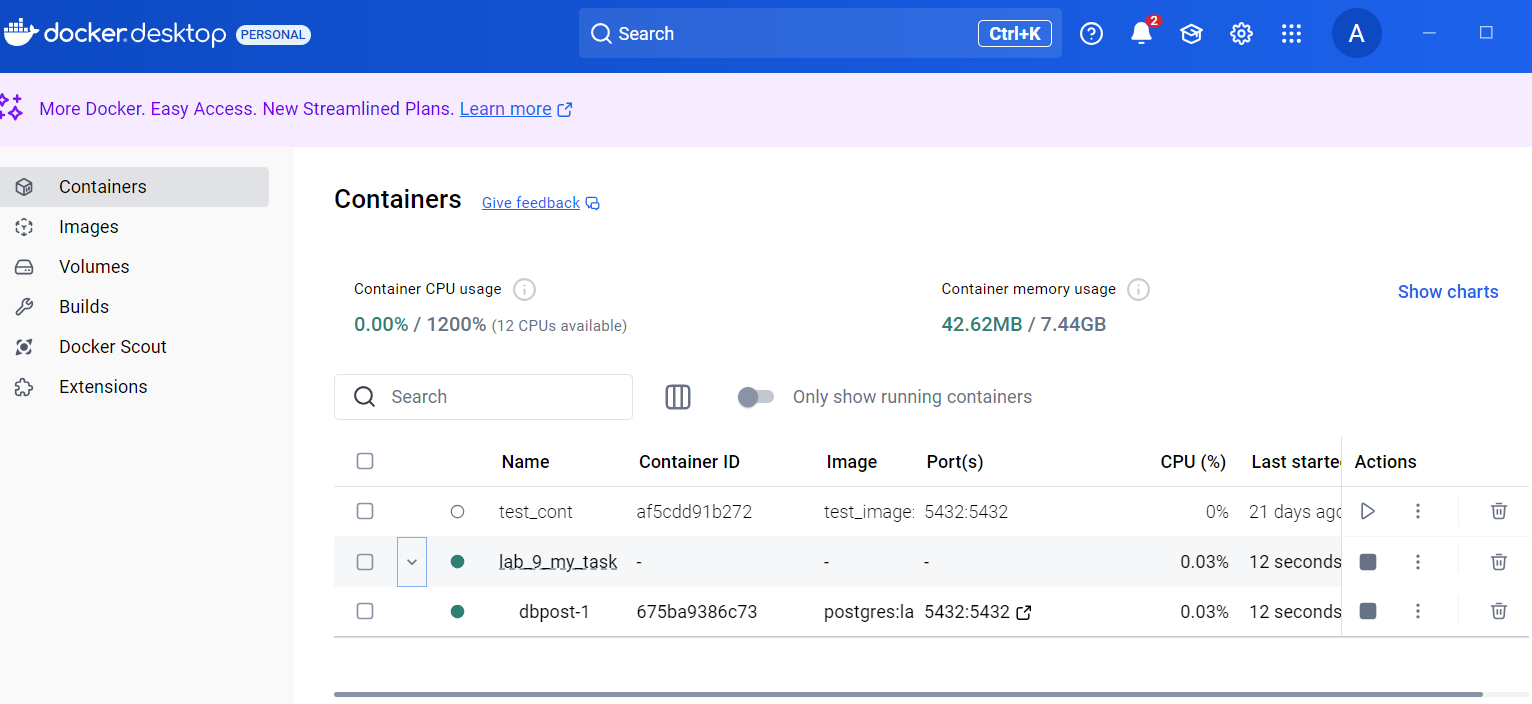
1. Сформировала SQL-запросы для своей задачи. Сохранила SQL-запросы в соответствующем файле init2.sql:

**

1. Обернула созданную базу данных в docker-контейнер (файл docker-compose.yml):





**

***Контрольные вопросы***

1. Что такое подзапрос? - это запрос, который вложен в другой запрос. Он используется для извлечения данных из одной таблицы, которые могут быть связаны с данными из другой таблицы.
2. Как заполнить таблицу данными? – использование операции INSERT INTO
3. Агрегатные функции. Основные сведения:

* SUM () - возвращает итоговое значение числового столбца
* COUNT() – производит подсчет количества строк в таблице
* AVG() - вычисляет среднее значение числового столбца
* MIN() - возвращает наименьшее значение в столбце
* MAX() - возвращает наибольшее значение в столбце

1. Как объединить таблицы в запросе? – с помощью оператора JOIN
2. Способы объединения таблиц. Краткая характеристика:

* INNER JOIN или просто JOIN - внутреннее соединение. В результате остаются только те строки, для которых нашлось соответствие.
* LEFT JOIN - левое внешнее соединение. Работает как JOIN, но если для строки таблицы, находящейся по левую сторону ключевого слова LEFT JOIN, не нашлось ни одной строки в таблице, находящейся по правую сторону LEFT JOIN, то строка все равно добавляется в результат, а значения столбцов правой таблицы равны null.
* RIGHT JOIN - правое внешнее соединение. Работает как JOIN, но если для строки таблицы, находящейся по правую сторону ключевого слова RIGHT JOIN, не нашлось ни одной строки в таблице, находящейся по левую сторону RIGHT JOIN, то строка все равно добавляется в результат, а значения столбцов левой таблицы равны null.
* FULL JOIN - полное внешнее соединение. Если для какой-либо из таблиц не нашлось строки в другой таблице, то строка все равно попадает в результат, а значения столбцов другой таблицы равны null.
* CROSS JOIN - перекрестное (или декартово) произведение. Каждая строка одной таблицы соединяется с каждой строкой второй таблицы, давая тем самым в результате все возможные сочетания строк двух таблиц. Аналогичного результата можно достичь просто перечислив таблицы в FROM через запятую.

1. План запроса. Краткое описание:
2. SELECT – выбирает отдельные столбцы или всю таблицу целиком (обязательный);
3. FROM - из какой таблицы получить данные (обязательный);
4. WHERE - условие, по которому SQL выбирает данные;
5. GROUP BY - столбец, по которому мы будут группироваться данные;
6. HAVING - условие, по которому сгруппированные данные будут отфильтрованы;
7. ORDER BY - столбец, по которому данные будут отсортированы;
8. Оптимизация запроса. Краткое описание.

Оптимизация SQL запросов — это процесс улучшения производительности запросов к базе данных.

1. Оконные функции. Краткое описание.

Синтаксис оконных функций:

* <window\_function\_name> — имя оконной функции, например ROW\_NUMBER(), RANK(), DENSE\_RANK() и другие.
* (arguments) — аргументы функции, например имя столбца, по которому происходит расчёт.
* OVER() — ключевое слово, которое определяет, как оконная функция будет применятся к набору данных.
* [Partitioning] — определяет критерий, по которому строки делятся на подгруппы. Это необязательный компонент.
* [Ordering] — указывает порядок строк в каждой подгруппе, что важно для таких функций, как RANK(), ROW\_NUMBER() и других. Этот элемент тоже необязателен.
* [Frame] — задаёт «фрейм» строк относительно текущей. Как правило, используется с ключевыми словами ROWS или RANGE.

1. Команда и синтаксис для группировки данных – команда GROUP BY объединяет записи с одинаковыми значениями в указанном списке полей в одну запись.

Синтакcис: SELECT \* FROM <имя\_таблицы> GROUP BY <поля\_группировки>

1. Сортировка данных. Краткое описание.

ORDER BY – команда сортировки

Синтаксис: SELECT \* FROM <имя\_таблицы> ORDER BY <столбец>

1. Условия для выборки данных (WHERE, HAVING). Краткое описание:

WHERE – условие, по которому SQL выбирает данные

HAVING - условие, по которому сгруппированные данные будут отфильтрованы