МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ 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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9  
  
По дисциплине: Системы хранения и обработки данных.

Тема: Формирование запросов к базе данных  
 Выполнил работу студент группы мИИВТ-231 Киселев А.А. (подпись) Фамилия, инициалы

Принял \_\_\_\_\_\_\_\_\_\_\_ Короленко В.В  
 (подпись) Фамилия, инициалы

Защищена\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

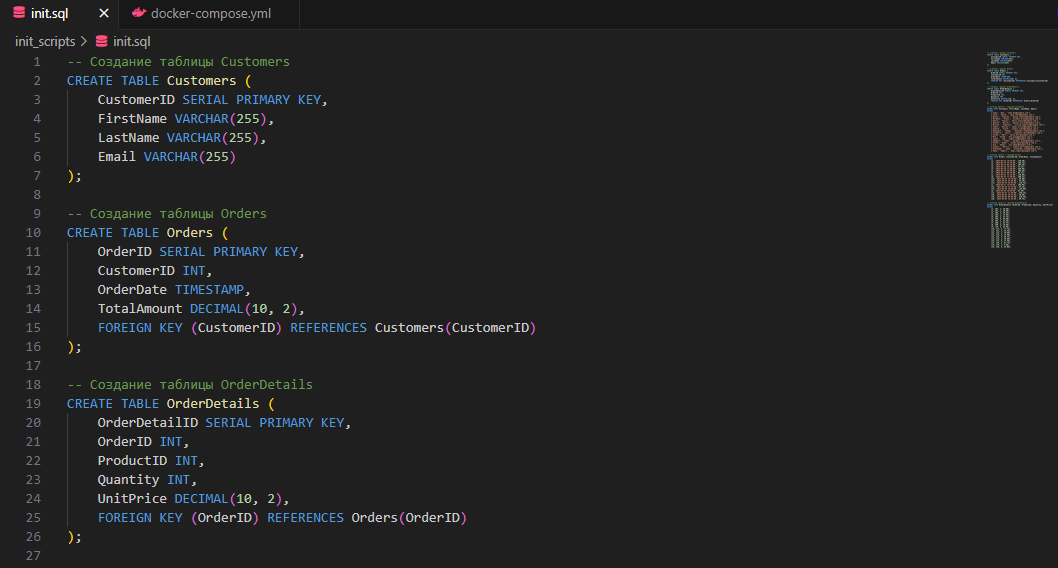
Воронеж 2023

**Цель лабораторной работы:** изучить синтаксис и основные команды для формирования запросов к базе данных, освоить процесс формирования SQL-запросов

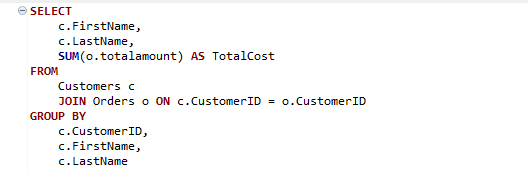
**Основные задачи:**

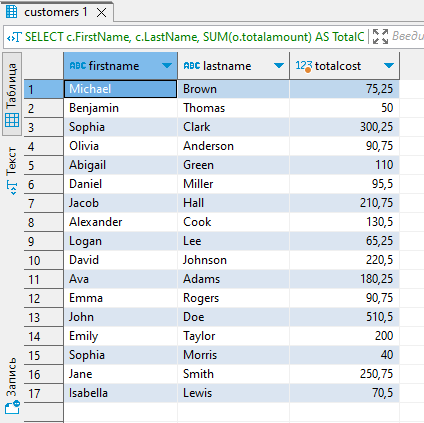
* Разработать структуру базы данных в СУБД Postgres.
* Наполнить базу данных данными.
* Написать ряд запросов к базе данных для отработки навыка формирования SQL-запросов.
* Научиться ставить задачу по формированию выборки необходимых данных из базы данных и решать её с помощью SQL-запросов.

1. **Изучить синтаксис SQL-запросов в СУБД Postgres**
2. **Изучить основы формирования SQL-запросов, включая такие вопросы, как: подзапросы, агрегатные функции, объединение таблиц**
3. **Изучить способы оптимизации запросов, план запроса, оконные функции**
4. **Сформировать SQL-запросы для создания таблиц в СУБД Postgres**



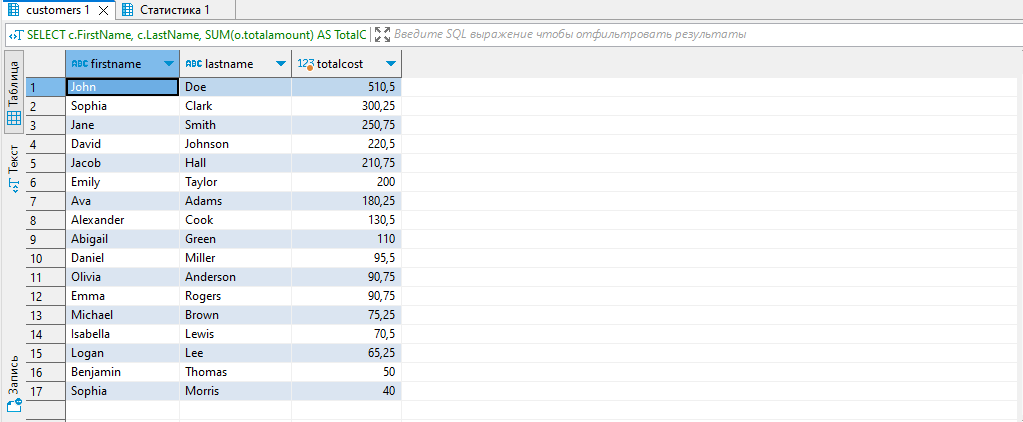
1. **При необходимости создать дополнительные атрибуты и таблицы**
2. **Наполнить базу данных данными (таблицы по 20 – 50 строк)**
3. **Сформировать SQL-запрос, который возвращает список клиентов (имя и фамилия) и суммарную стоимость заказов каждого клиента (должны отобразиться столбцы с именем, фамилией и стоимостью). То есть все заказы каждого клиента надо просуммировать**



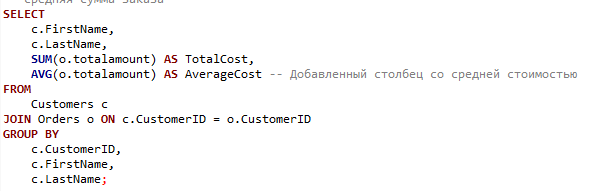


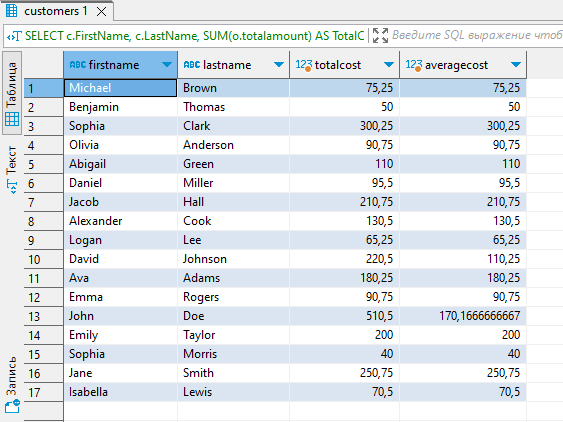
1. **Отсортировать с полученный список по убыванию суммарной стоимости заказов клиента**



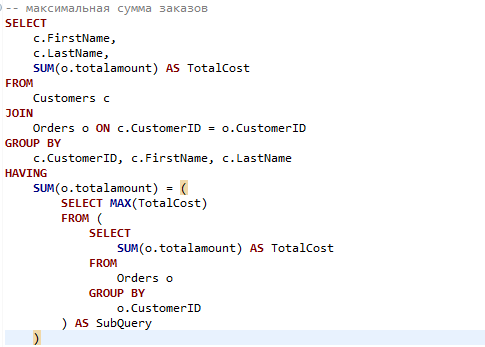


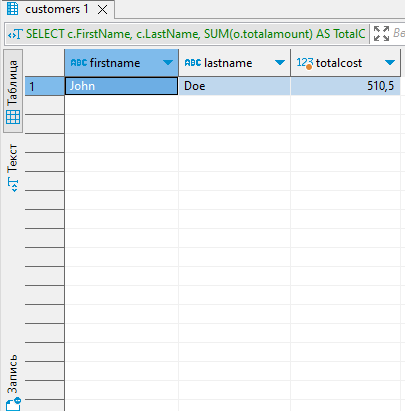
1. **Добавить столбец со средней суммарной стоимостью заказов**



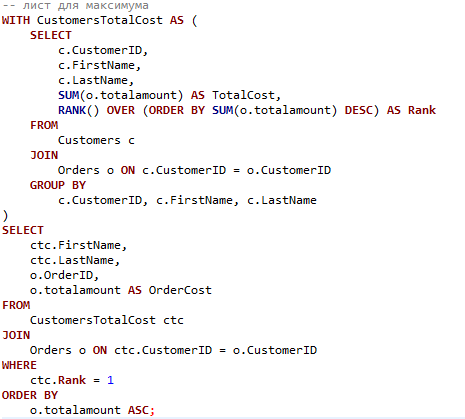


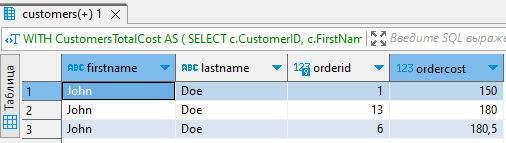
1. **Вывести клиента (список клиентов) с наибольшей суммарной стоимостью заказов (имя, фамилия, стоимость)**



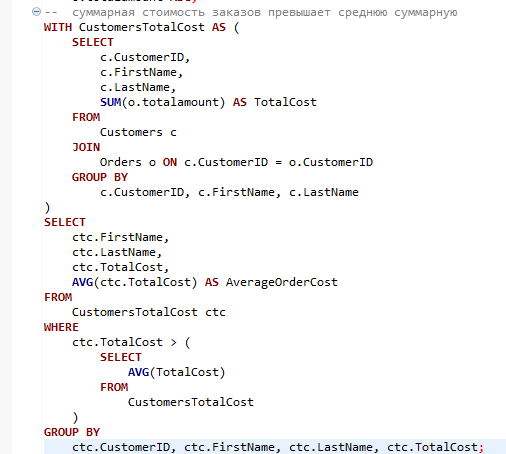


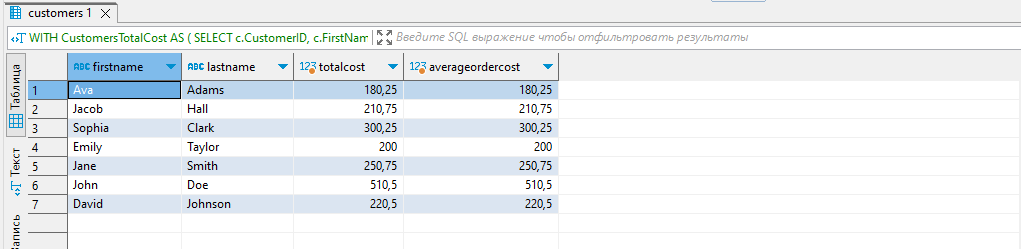
1. **Для каждого клиента с наибольшей суммарной стоимостью заказов вывести список его заказов (номер заказа и стоимость) в порядке возрастания стоимости заказа**





1. **Вывести только тех клиентов, у которых суммарная стоимость заказов превышает среднюю суммарную стоимость заказов клиентов (имя, фамилия, суммарная стоимость заказов клиента, средняя стоимость заказа)**

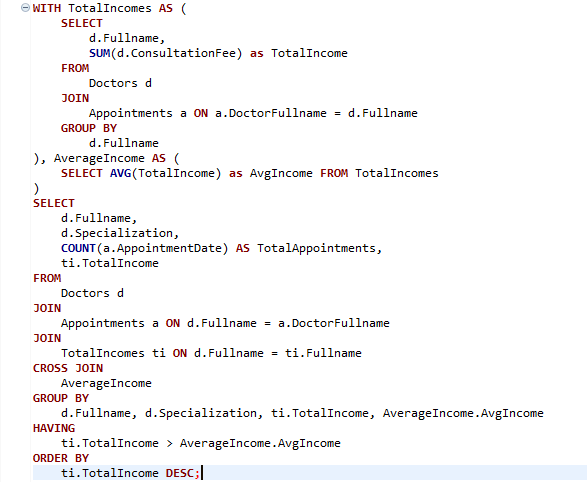


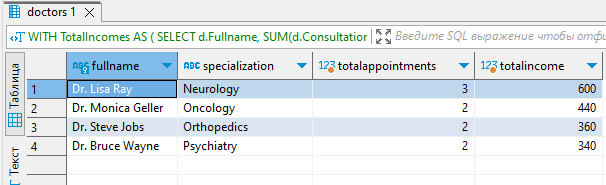


1. **Сохранить SQL-запросы в соответствующем файле со скриптами SQL**
2. **Сформулировать задачу, аналогичную изложенной выше (в пунктах 4 – 12), на основе базы данных, разработанной в предыдущей лабораторной работе в соответствии с индивидуальным заданием. Для своей задачи разрешается создать новую базу данных либо дополнить уже имеющуюся**

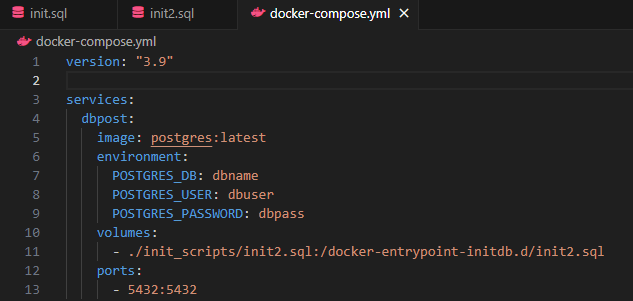
Задача запроса состоит в получении информации о врачах, их специализации, количестве назначений и общем доходе, а также выводе только тех записей, где общий доход превышает средний доход. Результаты должны быть упорядочены по убыванию общего дохода.

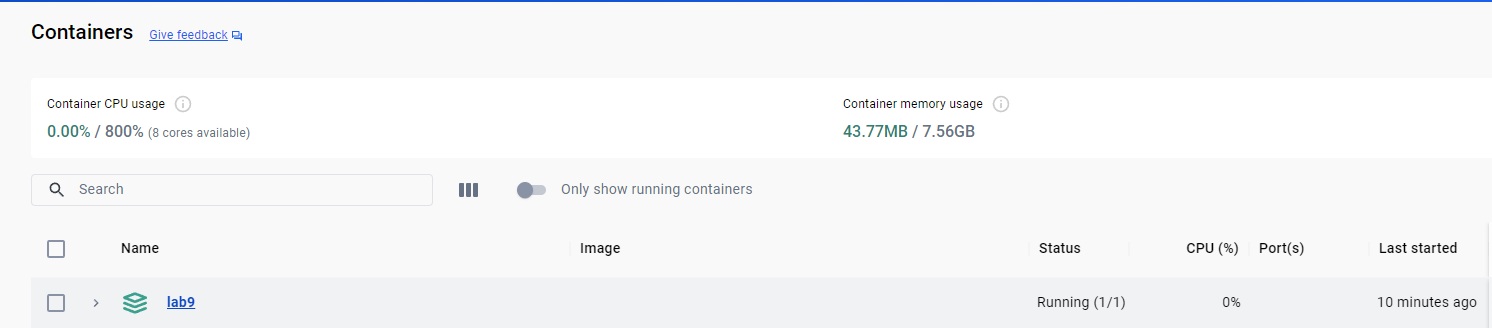
1. **Сохранить SQL-запросы для решения задачи согласно индивидуальному заданию в соответствующем файле со скриптами SQL**





1. **Описать каждый запрос (что делает запрос, из каких элементов (составляющих) состоит запрос, роль этих составляющих) (как для тренировочной задачи, так и для задачи в соответствии с индивидуальным заданием) в форме комментариев в файле со скриптами.**
2. **Обернуть созданную базу данных в docker-контейнер (файл docker-compose.yml)**





**Контрольные вопросы:**

1) Подзапрос - это запрос, который включается внутри другого запроса. Он используется для выполнения вычислений или получения данных, которые затем используются во внешнем запросе. Подзапросы могут быть использованы в различных частях запроса, таких как SELECT, FROM, WHERE, HAVING и других, в зависимости от того, где требуется использовать результаты подзапроса.

2) Для заполнения таблицы данными в PostgreSQL вы можете использовать оператор INSERT. Синтаксис команды INSERT выглядит следующим образом:

INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...);

Вы указываете имя таблицы и столбцы, в которые вы хотите вставить данные, а затем указываете значения для каждого столбца. Если вы хотите вставить несколько строк данных одновременно, вы можете использовать одну команду INSERT с несколькими наборами значений.

3) Агрегатные функции - это функции, которые применяются к набору значений и возвращают единственное значение в результате. Они используются для выполнения вычислений на группах данных, таких как сумма, среднее значение, минимум, максимум и т. д. Некоторые из основных агрегатных функций в PostgreSQL включают SUM, AVG, MIN, MAX, COUNT и другие.

4) Для объединения таблиц в запросе в PostgreSQL используются операторы JOIN. Оператор JOIN позволяет объединить строки из двух или более таблиц на основе условия сопоставления столбцов. Наиболее распространенные типы JOIN включают INNER JOIN, LEFT JOIN, RIGHT JOIN и FULL JOIN. INNER JOIN возвращает только совпадающие строки из обеих таблиц, LEFT JOIN возвращает все строки из левой таблицы и совпадающие строки из правой таблицы, RIGHT JOIN возвращает все строки из правой таблицы и совпадающие строки из левой таблицы, а FULL JOIN возвращает все строки из обеих таблиц.

5) Способы объединения таблиц включают INNER JOIN, LEFT JOIN, RIGHT JOIN и FULL JOIN:

- INNER JOIN возвращает только совпадающие строки из обеих таблиц.

- LEFT JOIN возвращает все строки из левой таблицы и совпадающие строки из правой таблицы.

- RIGHT JOIN возвращает все строки из правой таблицы и совпадающие строки из левой таблицы.

- FULL JOIN возвращает все строки из обеих таблиц.

6) План запроса (query plan) - это план выполнения запроса, который определяет оптимальный способ получения данных из таблицы или выполнения операций, указанных в запросе. План запроса включает в себя информацию о порядке выполнения операций, использовании индексов, выборе подходящих методов объединения таблиц и других оптимизационных шагах. План запроса может быть сгенерирован оптимизатором запросов в PostgreSQL для повышения производительности выполнения запроса.

7) Оптимизация запроса - это процесс улучшения производительности выполнения запроса путем выбора оптимального плана выполнения и применения различных оптимизаций. Оптимизация запроса включает в себя анализ структуры запроса, выбор оптимальных индексов, определение порядка выполнения операций и другие техники для улучшения производительности выполнения запроса.

8) Оконные функции (window functions) - это функции, которые выполняются над набором строк, определенных окном (window), которое является подмножеством строк из результирующего набора запроса. Оконные функции могут использоваться для выполнения вычислений и агрегаций, которые возвращают результаты для каждой строки окна, а не только для всего набора данных. Они позволяют выполнять сложные вычисления, такие как вычисление суммы, среднего значения, ранга и других значений на основе определенных оконных условий.

9) Для группировки данных в PostgreSQL используется команда GROUP BY. Команда GROUP BY позволяет группировать строки по одному или нескольким столбцам и выполнять агрегатные функции, такие как SUM, AVG, COUNT и другие, на каждой группе. Синтаксис команды GROUP BY выглядит следующим образом:

SELECT column1, column2, ..., aggregate\_function(column)

FROM table

GROUP BY column1, column2, ...;

Вы указываете столбцы, по которым вы хотите сгруппировать данные, а затем указываете агрегатные функции, которые вы хотите применить к каждой группе.

10) Сортировка данных в PostgreSQL выполняется с помощью оператора ORDER BY. Оператор ORDER BY позволяет упорядочивать строки в результирующем наборе по одному или нескольким столбцам. Вы можете указать направление сортировки (ASC - по возрастанию, DESC - по убыванию) для каждого столбца, по которому вы хотите выполнить сортировку. Синтаксис оператора ORDER BY выглядит следующим образом:

SELECT column1, column2, ...

FROM table

ORDER BY column1 ASC, column2 DESC, ...;

Вы указываете столбцы, по которым вы хотите выполнить сортировку, и указываете направление сортировки для каждого столбца.

11) Условия для выборки данных в PostgreSQL могут быть указаны с помощью операторов WHERE и HAVING. Оператор WHERE используется для фильтрации строк в результирующем наборе на основе условий, указанных в запросе. Он применяется перед выполнением группировки и агрегатных функций. Оператор HAVING используется для фильтрации группированных данных на основе условий, указанных в запросе. Он применяется после выполнения группировки и агрегатных функций.

Синтаксис оператора WHERE выглядит следующим образом:

SELECT column1, column2, ...

FROM table

WHERE condition;

Синтаксис оператора HAVING выглядит следующим образом:

SELECT column1, column2, ...

FROM table

GROUP BY column1, column2, ...

HAVING condition;

Вы указываете условия, которые должны быть выполнены для выборки данных в операторах WHERE и HAVING.