**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА**

ФедеральноеГосударственное Бюджетное Образовательное Учреждение

Высшего Образования

**«Петербургский Государственный Университет Путей Сообщения**

**Императора Александра I»**

**(ФГБОУ ВО ПГУПС)**

Факультет «Автоматизация и интеллектуальные технологии»

Кафедра «Информационные и вычислительные системы»

**ЛАБОРАТОРНАЯ РАБОТА №7**

по дисциплине

**«Программирование»**

ТЕМА РАБОТЫ

**«Создание таблиц и выполнение основных SQL-запросов. Управление данными (вставка, обновление удаление). Выполнение сложных SQL-запросов. Использование подзапросов и объединение таблиц»**

Выполнил:

|  |  |  |  |
| --- | --- | --- | --- |
| Обучающийся группы ИВБ-417 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | | М.Д. Карванен\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  И. О. Фамилия |
| Исправить замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |  | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | к.в.н. доцент В.Е.Петров\_\_\_\_\_\_\_  учёное звание, И.О. Фамилия |
| Защита: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Зачтено/не зачтено | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | | к.в.н. доцент В.Е.Петров\_\_\_\_\_  учёное звание, И.О. Фамилия |

**Санкт-Петербург**

**2025г.**

**Целями** (целью) выполнения лабораторной работы являются:

1. Изучить концепцию ключей в реляционной модели данных и их типы.
2. Освоить основные принципы реляционной модели данных.
3. Приобрести практические навыки работы с SQL-запросами:
   * Создание и заполнение таблиц.
   * Выполнение базовых операций (SELECT, INSERT, UPDATE, DELETE).
   * Использование сортировки, фильтрации и агрегации данных.
   * Применение подзапросов и объединение таблиц (JOIN).
   * Создание сложных запросов для анализа данных.

**Вариант индивидуального задания**

**Задание:**

1. Теоретическая часть.

Поясните концепцию ключей, которая является ключевым элементом реляционной модели данных. Основные типы ключей.

Раскройте основные принципы реляционной модели данных.

1. Практическая часть.

А) Создайте таблицу студентов и выполните запрос для выбора студентов с оценкой выше 4.

Б) Создайте таблицу товаров и выполните запрос для сортировки по цене.

В) Напишите запрос для объединения данных из таблиц клиентов и заказов.

Г) Создайте таблицу работников и выполните запрос для обновления их должностей.

Д) Напишите запрос для удаления всех заказов, сделанных более года назад.

Е) Создайте таблицу заказов и выполните запрос для подсчета количества заказов каждого клиента.

Ж) Напишите запрос для выборки всех товаров, у которых цена выше среднего.

З) Создайте таблицу с информацией о книгах и выполните запрос для нахождения всех книг конкретного товара

И) Напишите запрос для создания новой таблицы с объединенными данными из нескольких других таблиц.

**Выполнение задания**

1. **Теоретическая часть**

***Поясните концепцию ключей, которая является ключевым элементом реляционной модели данных. Основные типы ключей.***

Ключи являются фундаментальным элементом реляционной модели данных, обеспечивая уникальность записей и устанавливая связи между таблицами. Они играют важную роль в поддержании целостности данных и организации структуры базы данных.

Основные типы ключей:

1. Первичный ключ (Primary Key)

Это один или несколько атрибутов, которые однозначно идентифицируют каждую строку в таблице. Первичный ключ гарантирует уникальность каждой записи и не может содержать значения NULL. Например, в таблице "Сотрудники" первичным ключом может быть атрибут "ID сотрудника";

1. Внешний ключ (Foreign Key)

Это атрибут или набор атрибутов, которые ссылаются на первичный ключ другой таблицы. Внешние ключи используются для установления связей между таблицами. Например, в таблице "Проекты" внешний ключ "ID сотрудника" может ссылаться на первичный ключ таблицы "Сотрудники", связывая проекты с ответственными за них сотрудниками;

1. Кандидатный ключ (Candidate Key)

Это атрибут или набор атрибутов, которые могут быть использованы для однозначной идентификации строк в таблице. В отличие от первичного ключа, в таблице может быть несколько кандидатных ключей, и один из них выбирается в качестве первичного. Например, в таблице "Сотрудники" кандидатными ключами могут быть "Идентификатор сотрудника" и "Номер паспорта", так как оба атрибута могут однозначно идентифицировать сотрудника. Однако только один из них станет первичным ключом, а остальные останутся кандидатными.

1. Составной ключ (Composite Key)

Это разновидность ключа, который состоит из двух или более атрибутов, совместно обеспечивающих уникальность записи в таблице. Составной ключ используется в тех случаях, когда ни один из атрибутов по отдельности не может гарантировать уникальность строки. Например, в таблице "Расписание занятий" составной ключ может состоять из атрибутов "Дата" и "Время", так как только их комбинация может однозначно идентифицировать конкретное занятие. Составные ключи особенно полезны в таблицах, которые отражают отношения "многие ко многим", где уникальность записи обеспечивается комбинацией нескольких атрибутов.

Они позволяют не только организовать данные в таблицах, но и обеспечить их целостность, а также создать эффективные связи между таблицами, что критически важно для работы с большими объемами данных и сложными структурами.

***Раскройте основные принципы реляционной модели данных.***

Реляционные базы данных основаны на строгой структурированной модели, где данные организованы в таблицы с четко определенными связями между ними, тогда как нереляционные базы данных предоставляют более гибкий подход к организации данных, ориентированный на хранение сложных, слабо структурированных или неструктурированных данных.

В основе реляционной модели лежит концепция таблиц, где данные упорядочены в строки и столбцы. Каждая строка представляет собой запись, а каждый столбец – поле с определенным типом данных. Это обеспечивает четкость и предсказуемость структуры данных, что особенно важно при работе с большими объемами информации, требующей строгой организации.

Для работы с реляционными базами данных используется стандартный язык SQL (Structured Query Language), который позволяет выполнять мощные запросы, манипулировать данными, создавать сложные выборки и выполнять агрегатные операции.

Реляционная модель данных строится на нескольких принципах, которые определяют её структуру и правила взаимодействия с данными:

* Уникальность строк – каждая строка в таблице должна быть уникальной, что достигается использованием первичного ключа. Это исключает возможность дублирования данных и облегчает поиск конкретной информации.
* Отношения между таблицами – таблицы в реляционной базе данных не изолированы, они связаны друг с другом через внешние ключи. Это позволяет представлять сложные связи между данными, например, отношения "один-к-одному", "один-ко-многим" или "многие-ко-многим".
* Целостность данных – помимо ссылочной целостности, реляционная модель поддерживает доменную целостность, которая гарантирует, что данные в каждом атрибуте соответствуют определенным ограничениям. Например, числовое поле не может содержать текстовые значения.

1. **Практическая часть**

***А)***

**Запрос создания таблицы:**

CREATE TABLE "students" (

"id" INTEGER,

"name" TEXT NOT NULL,

"group\_name" TEXT,

"grade" REAL CHECK(grade BETWEEN 1 AND 5),

PRIMARY KEY("id" AUTOINCREMENT)

)

**Запрос на заполнение данных:**

INSERT INTO students (name, group\_name, grade) VALUES

('Иванов Иван', 'Группа 101', 4.5),

('Петрова Мария', 'Группа 102', 3.8),

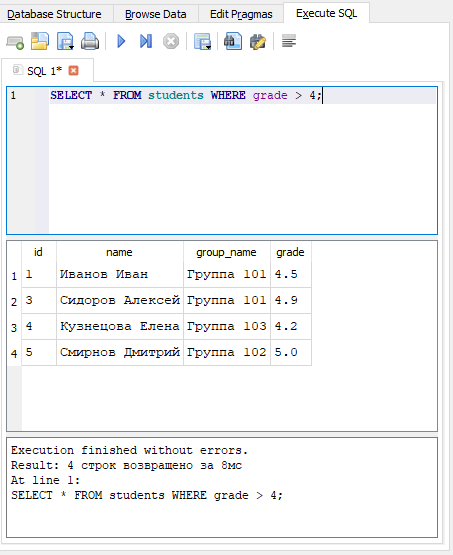
('Сидоров Алексей', 'Группа 101', 4.9),

('Кузнецова Елена', 'Группа 103', 4.2),

('Смирнов Дмитрий', 'Группа 102', 5.0);

**Запрос на выборку студентов с оценкой выше 4:**

SELECT \* FROM students WHERE grade > 4;

****

***Б)***

**Создание таблицы товаров:**

CREATE TABLE "products" (

"id" INTEGER,

"name" TEXT NOT NULL,

"price" REAL NOT NULL,

"category" TEXT,

PRIMARY KEY("id" AUTOINCREMENT)

);

**Запрос на заполнение таблицы данными:**

INSERT INTO products (name, price, category) VALUES

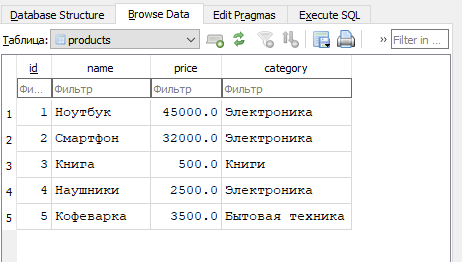
('Ноутбук', 45000, 'Электроника'),

('Смартфон', 32000, 'Электроника'),

('Книга', 500, 'Книги'),

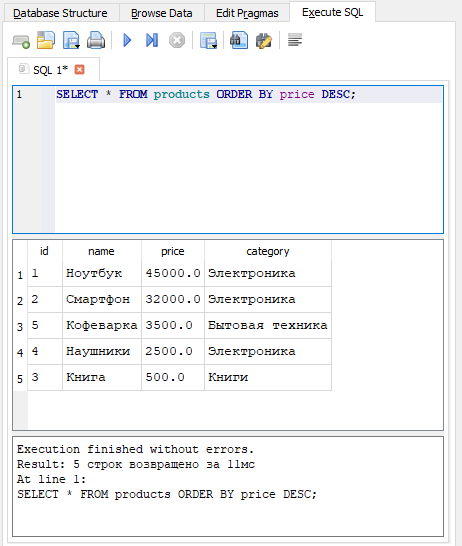
('Наушники', 2500, 'Электроника'),

('Кофеварка', 3500, 'Бытовая техника');



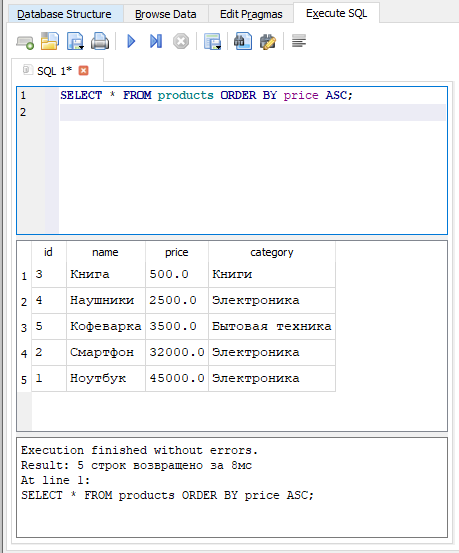
**Запрос на сортировку по цене (по возрастанию):**

SELECT \* FROM products ORDER BY price DESC;



**Запрос на сортировку по цене (по убыванию):**

SELECT \* FROM products ORDER BY price ASC;



***В)***

**Запрос на создание таблиц с клиентами и заказами, и их заполнение данными**CREATE TABLE "customers" (

"id" INTEGER,

"name" TEXT NOT NULL,

"email" TEXT,

"phone" TEXT,

PRIMARY KEY("id" AUTOINCREMENT)

);

CREATE TABLE orders (

id INTEGER PRIMARY KEY AUTOINCREMENT,

customer\_id INTEGER,

order\_date TEXT,

amount REAL,

FOREIGN KEY (customer\_id) REFERENCES customers(id)

)

INSERT INTO customers (name, email, phone) VALUES

('ООО "Ромашка"', 'romashka@mail.com', '+79151234567'),

('ИП Сидоров', 'sidorov@mail.com', '+79059876543'),

('ЗАО "Вектор"', 'vector@mail.com', '+74951234567');

INSERT INTO orders (customer\_id, order\_date, amount) VALUES

(1, '2023-05-15', 12000),

(2, '2023-06-20', 8500),

(1, '2023-07-10', 23500),

(3, '2023-08-05', 15000);

**Запрос на объединение таблиц через INNER JOIN:**

SELECT

c.name AS customer\_name,

c.email,

o.order\_date,

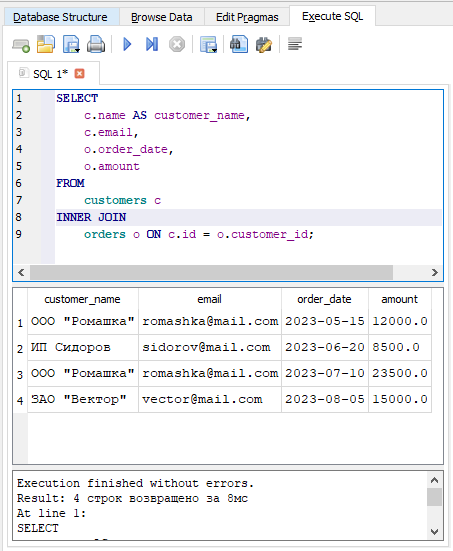
o.amount

FROM

customers c

INNER JOIN

orders o ON c.id = o.customer\_id;



***Г)***

**Создание таблицы работников и заполнение её данными:**

CREATE TABLE "employees" (

"id" INTEGER,

"name" TEXT NOT NULL,

"position" TEXT,

"department" TEXT,

"salary" REAL,

PRIMARY KEY("id" AUTOINCREMENT)

);

INSERT INTO employees (name, position, department, salary) VALUES

('Иванов И.И.', 'Менеджер', 'Продажи', 50000),

('Петрова П.П.', 'Разработчик', 'IT', 80000),

('Сидоров С.С.', 'Аналитик', 'Маркетинг', 60000),

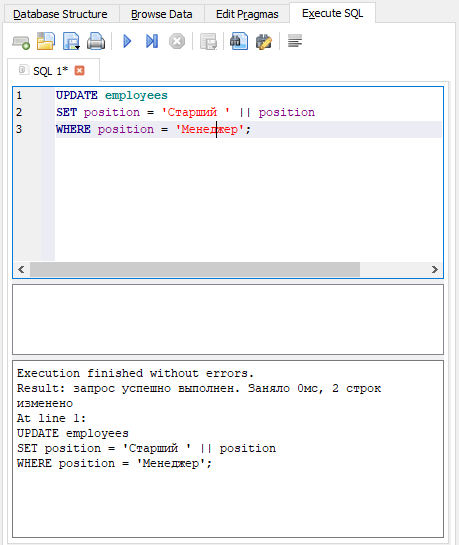
('Кузнецова Е.В.', 'Менеджер', 'Продажи', 55000);

**Запрос на обновление данных (Менеджеров повышаем до старших менеджеров):**

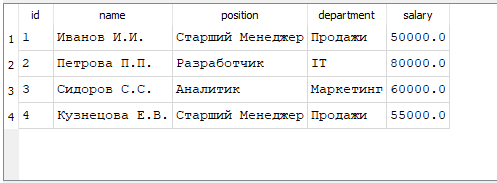
UPDATE employees

SET position = 'Старший ' || position

WHERE position = 'Менеджер';

******

***Результат:***

******

***Д)***

***Добавим в раннее созданную таблицу «orders» данные:***

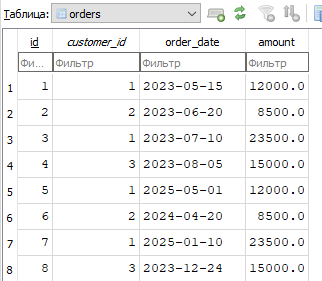
INSERT INTO orders (customer\_id, order\_date, amount) VALUES

(1, '2025-05-01', 12000),

(2, '2024-04-20', 8500),

(1, '2025-01-10', 23500),

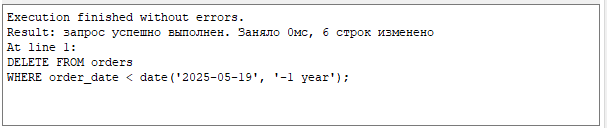
(3, '2023-12-24', 15000);

******

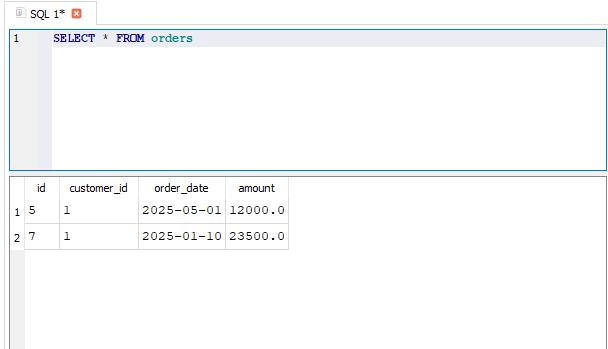
**Удалим заказы, сделанные более 1 года назад:**

DELETE FROM orders

WHERE order\_date < date('2025-05-19', '-1 year');



***Результат:***

******

***Е)***

**Используем раннее созданные таблицы «customers» и «orders»**

**Заполним их данными:**

INSERT INTO orders (customer\_id, order\_date, amount) VALUES

(1, '2023-05-15', 12000),

(2, '2023-06-20', 8500),

(1, '2023-07-10', 23500),

(3, '2023-08-05', 15000);

**Запрос на подсчет заказов для каждого клиента:**

SELECT

c.id,

c.name AS customer\_name,

COUNT(o.id) AS orders\_count,

SUM(o.amount) AS total\_amount

FROM

customers c

LEFT JOIN

orders o ON c.id = o.customer\_id

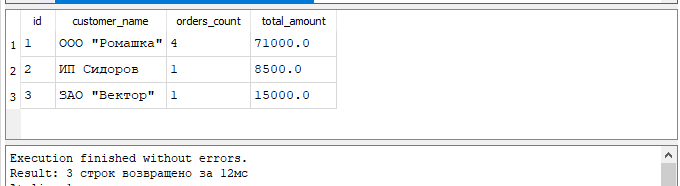
GROUP BY

c.id, c.name

ORDER BY

orders\_count DESC;

**Результат:**

******

***Ж)***

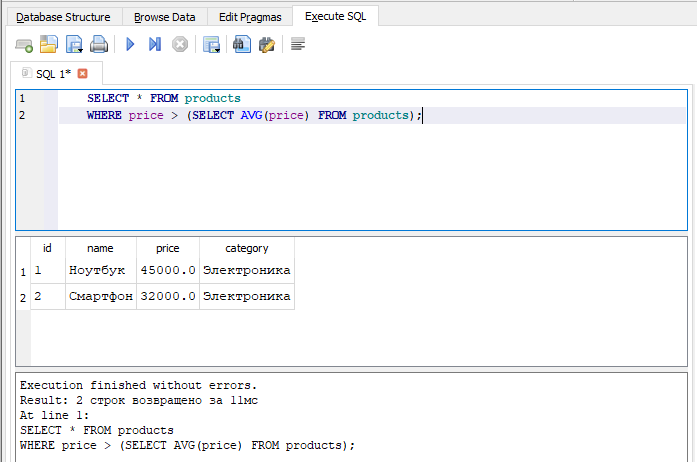
***Используем раннее созданную таблицу «products»***

***Запрос на выборку товаров с ценой выше среднего:***

SELECT \* FROM products

WHERE price > (SELECT AVG(price) FROM products);

**Результат:**



***З)***

***Запрос на создание таблицы «books»***

CREATE TABLE books (

id INTEGER PRIMARY KEY AUTOINCREMENT,

title TEXT NOT NULL,

author TEXT NOT NULL,

year INTEGER,

price REAL,

genre TEXT

)

**Заполнение таблицы данными:**

INSERT INTO books (title, author, year, price, genre) VALUES

('Мастер и Маргарита', 'Булгаков М.А.', 1967, 450, 'Роман'),

('Преступление и наказание', 'Достоевский Ф.М.', 1866, 380, 'Роман'),

('Война и мир', 'Толстой Л.Н.', 1869, 600, 'Роман'),

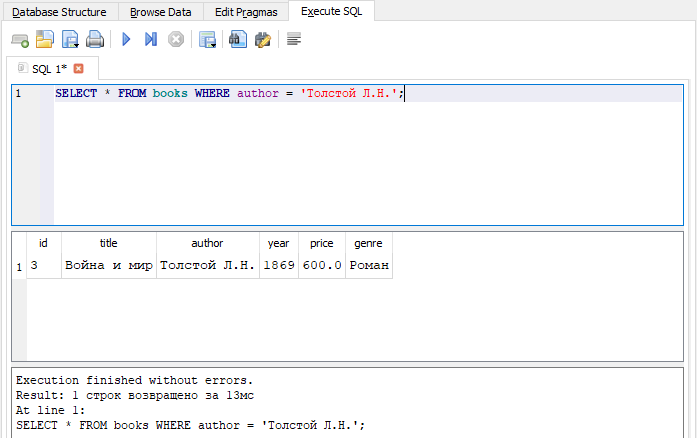
('Мёртвые души', 'Гоголь Н.В.', 1842, 400, 'Поэма'),

('Герой нашего времени', 'Лермонтов М.Ю.', 1840, 350, 'Роман');

**Запрос на нахождения всех книг принадлежащим конкретному автору (Л.Н. Толстой):**

SELECT \* FROM books WHERE author = 'Толстой Л.Н.';

**Результат:**



***И)***

**Запрос на создание новой таблицы «customer\_orders\_summary» с объединенными данными из таблиц «customers» и «orders»:**

CREATE TABLE customer\_orders\_summary AS

SELECT

c.id AS customer\_id,

c.name AS customer\_name,

c.email,

COUNT(o.id) AS orders\_count,

SUM(o.amount) AS total\_amount,

MAX(o.order\_date) AS last\_order\_date

FROM

customers c

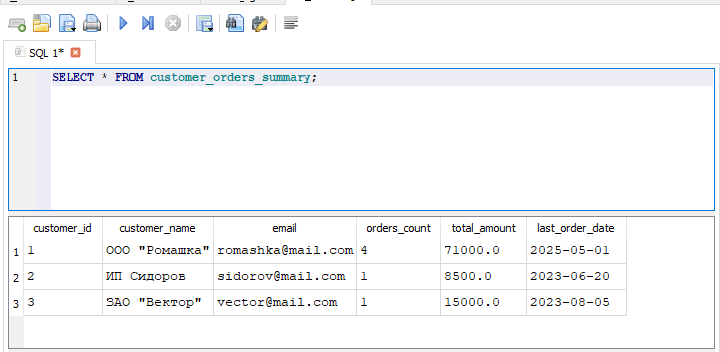
LEFT JOIN

orders o ON c.id = o.customer\_id

GROUP BY

c.id, c.name, c.email;

**Результат:**

****

**База данных с таблицами находится в репозитории по ссылке:**

<https://github.com/MaksimKarvanen/lab7_prog>

**Выводы**

1. В ходе работы изучены ключевые аспекты реляционной модели данных, включая типы ключей (первичный, внешний, кандидатный, составной) и их роль в обеспечении целостности данных.
2. Освоены принципы проектирования реляционных баз данных, включая нормализацию и установление связей между таблицами.
3. На практике закреплены навыки написания SQL-запросов:
   * Созданы и заполнены таблицы (students, products, customers, orders, employees, books).
   * Выполнены выборки с условиями (WHERE), сортировкой (ORDER BY) и агрегацией (COUNT, SUM).
   * Применены операции обновления (UPDATE) и удаления (DELETE) данных.
   * Реализованы сложные запросы с использованием подзапросов и объединения таблиц (JOIN).