Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
«Российский экономический университет имени Г.В. Плеханова»
МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ

специальность 09.02.07 «Информационные системы и программирование»

Квалификация: Программист

ПРАКТИЧЕСКАЯ РАБОТА №1

ПО МДК 11.01 «Технология разработки и защиты баз данных»

Выполнила студентка
группы П50-4-22
Н.И. Григоренко

Проверил преподаватель
_____ К.А. Перевалов
«___» _____ 2024 года

Москва 2024

«Базы данных. SQL»

Цель работы: создать базу данных по теме «Гитарный магазин», провести три запроса SELECT с использованием команд (ORDER BY, GROUP BY, HAVING, WHERE, DISTINCT, LIMIT, LIKE, CASE), продемонстрировать работу агрегатных функций, провести запрос с использованием оконной функции на языке SQL.

Ход работы:

1. Создание базы данных и её использование.

Для начала необходимо зайти в «Microsoft SQL Server Management Studio» и создать новый SQL-запрос. После справа на рабочей области появится окно для того, чтобы писать в нем скрипт для базы данных. Если нажать на синюю кнопку в верхней панели – файл сохранится в указанное пользователем место.

Первое, что надо сделать после создания файла скрипта – создать базу данных, для чего используется оператор «CREATE», далее нужно прописать «DATABASE» и придумать название для базы данных. Чтобы выполнить часть скрипта, нужно её выделить курсором мышки и нажать на кнопку «Выполнить» в верхней панели. Для системы необходимо уточнить, с какой именной БД будет происходить работа, поэтому ниже нужно прописать команду USE, указав название базы данных и выполнить данную часть кода. Над обозревателем объектов появится название используемой в данный момент базы данных. Для разделения кода в MS SQL используется команда «GO», она помогает сообщить базе данных об окончании конкретной операции, чтобы ничего не сливалось в сплошную большую команду (см. Рисунок 1 – Создание базы данных).

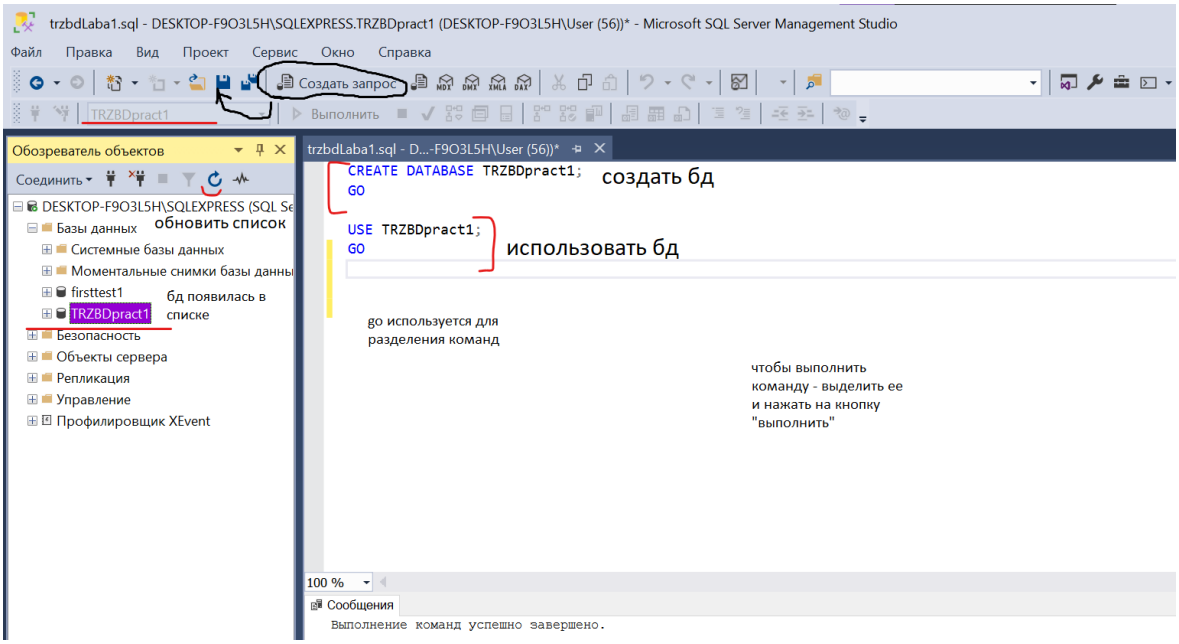


Рисунок 1 – Создание базы данных

2. Создание и заполнение таблиц.

Для того чтобы создать таблицу, необходимо прописать оператор «CREATE» с ключевым словом «TABLE» и указанием названия таблицы. В круглых скобках после названия таблицы прописываются все поля с типами данных через запятую (столбцы), после ставится точка с запятой. Первая таблица называется «VidGuitar», она включает в себя два столбца – «ID_VidGuitar» целочисленного типа INT (так как это идентификатор), который является первичным ключом. С помощью IDENTITY(1,1) устанавливаем автоматическое выставление идентификатора атрибута, при добавлении новой записи он будет увеличиваться на число, прописанное вторым в скобках, то есть здесь - на единицу; «NameOfVidGuitar» - название гитары, имеет символьный тип данных «VARCHAR» с ограничением в скобках в 40 символов. Заполнение таблицы происходит через оператор INSERT и ключевое слово INTO, далее указывается название заполняемой таблицы, а в скобках прописываются поля для заполнения. После ключевого слова VALUES в скобках и одинарных кавычках пишутся значения – добавляемые в таблицу строки (см. Рисунок 2 – Таблица «VidGuitar»: создание и заполнение).

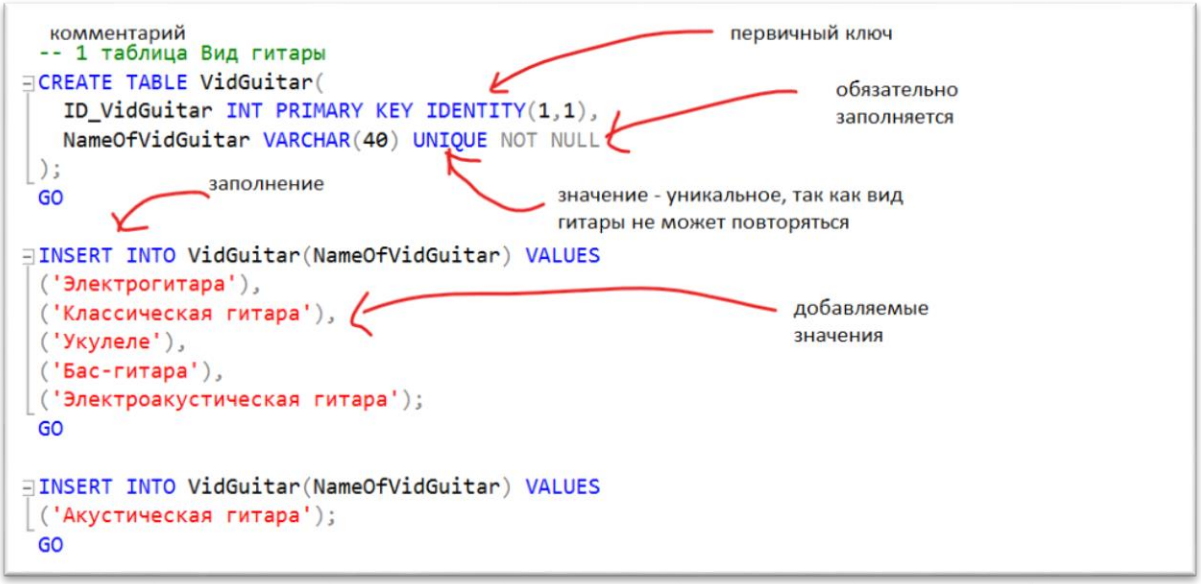


Рисунок 3 – Таблица «VidGuitar»: создание и заполнение

Таким же образом заполняется вторая таблица – должности сотрудников (см. Рисунок 3 – Таблица «PositionsEmployees»).

```
-- 2 таблица Должность сотрудника
CREATE TABLE PositionsEmployees(
  ID_PositionsEmployees INT PRIMARY KEY IDENTITY(1,1),
  NameOfPositionEmployees VARCHAR(40) UNIQUE NOT NULL
);
GO

INSERT INTO PositionsEmployees (NameOfPositionEmployees)
VALUES
('Генеральный директор'),
('Менеджер по продажам'),
('Кассир'),
('Менеджер по закупкам'),
('Продавец-консультант');
GO
```

Рисунок 2 – Таблица «PositionsEmployees»

Внешние таблицы создаются первыми, так как пока не имеют ни с чем другим связей. Далее создаются основные таблицы, в которых будут содержаться внешние ключи, ссылающиеся на другие таблицы. Третья таблица называется «Employees» и содержит в себе информацию о сотрудниках гитарного магазина, имеет столбцы: «ID_Employees» - первичный ключ, «NameEmployee» типа «VARCHAR» - имя сотрудника, «SurnameEmployee» - фамилия сотрудника, «PatronymicEmployee»

отчество сотрудника. «PositionsEmployees_ID int» - внешний ключ. С помощью «FOREIGN KEY» и «REFERENCES» связывается внешний ключ с первичным «ID_PositionsEmployees» (то есть сотрудник с должностью) из ранее созданной таблицы «PositionsEmployees». У сотрудника может быть одна должность, но эта должность может при этом быть и у других сотрудников: например, один человек – кассир, занимает только одну должность, но в магазине работает несколько кассиров, то есть должность встречается не один раз, поэтому здесь реализована связь один ко многим.

После создается таблица клиентов «Clients» – в ней первичный ключ, имя, фамилия и отчество клиента. Также реализуется еще одна внешняя таблица «OrderStatus» – статус заказа, которая имеет всего два поля – название статуса и первичный ключ.

Таблица «Guitars» содержит в себе информацию про гитары: название, цена, внешний ключ к таблице «Вида гитары», который связывается с таблицей «VidGuitar». Почти все заполняемые поля в этой БД имеют ограничение «NOT NULL», что значит обязательность их заполнения, они не могут оставаться пустыми (см. Рисунок 4 – Таблицы сотрудников, клиентов, гитар и статуса заказа).

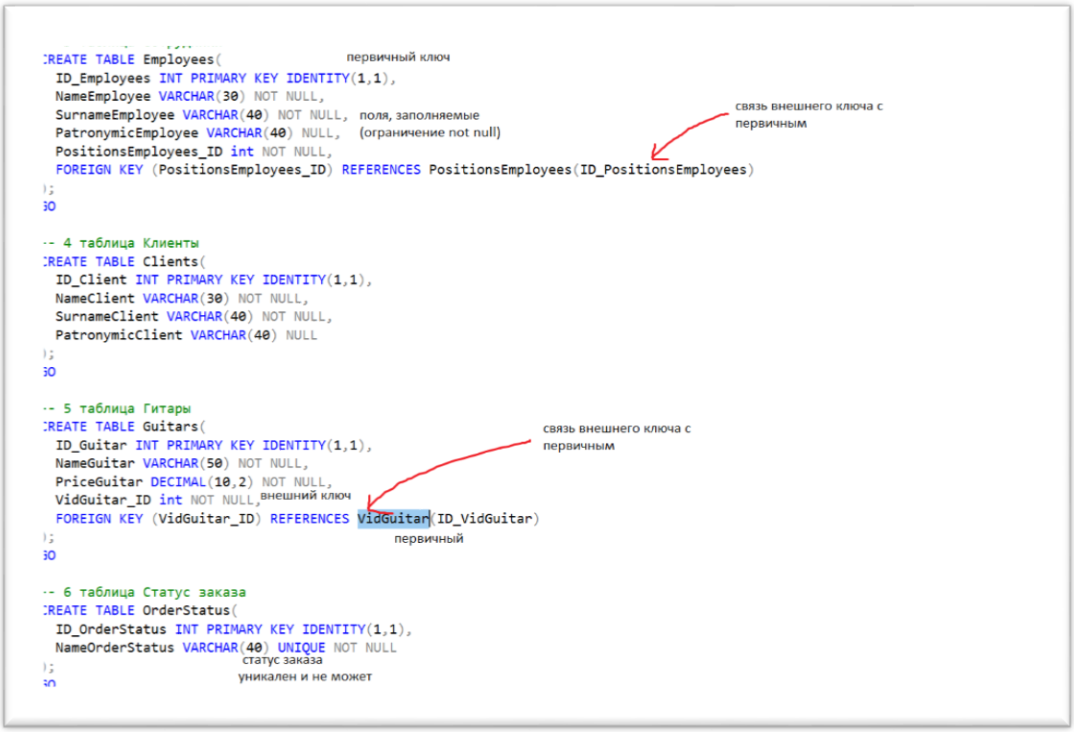


Рисунок 4 – Таблицы сотрудников, клиентов, гитар и статуса заказа

Таблица, в которой больше всего связей с остальными таблицами – таблица заказов «Orders». Она включает в себя первичный ключ, поле количества гитар типа данных int, внешний ключ для связи с таблицей гитар, внешний ключ для связи с клиентами (у одного клиента может быть несколько заказов, один ко многим), дата совершения заказа типа VARCHAR с ограничением до десяти символов – формата даты «гггг-мм-дд», итоговая стоимость заказа десятичного числового типа данных decimal с ограничением до двух цифр после запятой, номер заказа типа int, внешний ключ для связи со статусом заказа, далее идет связывание через «FOREIGN KEY» и «REFERENCES» (см. Рисунок 5 – Таблица заказов).

```
-- 7 таблица Заказы
CREATE TABLE Orders(
  ID_Order INT PRIMARY KEY IDENTITY(1,1),
  AmountGuitars int NOT NULL,
  Guitar_ID int NOT NULL,
  Client_ID int NOT NULL,
  DateOfMadeOrder VARCHAR(10) NOT NULL,
  TotalPrice DECIMAL(10,2) NOT NULL,
  Employee_ID int NOT NULL,
  NumberOfOrder int NOT NULL,
  OrderStatus_ID int NOT NULL,
  FOREIGN KEY (Guitar_ID) REFERENCES Guitars(ID_Guitar),
  FOREIGN KEY (Client_ID) REFERENCES Clients(ID_Client),
  FOREIGN KEY (Employee_ID) REFERENCES Employees(ID_Employees),
  FOREIGN KEY (OrderStatus_ID) REFERENCES OrderStatus(ID_OrderStatus),
);
GO
```

Рисунок 5 – Таблица заказов

После создания всех таблиц, они заполняются данными по несколько строк для наглядности с помощью оператора «INSERT» (см. Рисунок 6 – Заполнение таблиц данными).

```
INSERT INTO Employees (NameEmployee, SurnameEmployee, PatronymicEmployee, PositionsEmployees_ID) VALUES
('Елизавета', 'Парамонова', 'Михайловна', 1),
('Ангус', 'Уильям', 'Янг', 2),
('Френсис', 'Бин', 'Кобейн', 3),
('Евгений', 'Скиллет', 'Нурланович', 4),
('Кристина', 'Электрофорез', 'Рустамовна', 5);

-- Клиенты
INSERT INTO Clients (NameClient, SurnameClient, PatronymicClient) VALUES
('Ярослав', 'Шеметов', 'Дмитриевич'),
('Ксения', 'Школьникова', 'Васильевна'),
('Тимофей', 'Мурашов', 'Александрович'),
('Илья', 'Артемюк', 'Ростиславович');

-- Гитары
INSERT INTO Guitars (NameGuitar, PriceGuitar, VidGuitar_ID) VALUES
('IBANEZ GRG121DX-BKF', 26799.99, 1),
('GIBSON SG Special Vintage Cherry', 256500.00, 1),
('MARTINEZ FAW-702 BL', 11300.00, 7),
('FENDER SQUIER Affinity 2021 Stratocaster MN Black', 40000.00, 1),
('FLIGHT NUT 310', 8390.00, 3),
('Schecter SGR C-4 BASS BLK', 33087.99, 4),
('IBANEZ TCY10E-BK', 28500.00, 5),
('ALHAMBRA 3C', 70739.99, 2);

-- Заказы
INSERT INTO Orders (AmountGuitars, Guitar_ID, Client_ID, DateOfMadeOrder, TotalPrice, Employee_ID, NumberOfOrder, OrderStatus_ID) VALUES
(1, 1, 1, '2024-10-07', 30799.00, 2, 123, 1),
(3, 3, 2, '2024-09-14', 90090.00, 5, 456, 2),
(2, 4, 3, '2024-10-27', 80000.00, 2, 789, 4),
(1, 5, 4, '2024-12-31', 270500.00, 3, 101, 3);
-- Статус заказа
INSERT INTO OrderStatus (NameOrderStatus) VALUES
('Обработан'),
('В пути'),
('Доставлен'),
('Задерживается')
```

заполняются все поля, первичный ключ заполняется автоматически

цена - десятичное число. точность - два знака после запятой

дата в текстовом формате для удобства

Рисунок 7 – Заполнение таблиц данными

Для наглядности результата можно создать диаграмму базы данных, кликнув по соответствующему пункту правой кнопкой мышки и выбрав пункт «Создать диаграмму базы данных» (см. Рисунок 7 – Диаграмма базы данных).

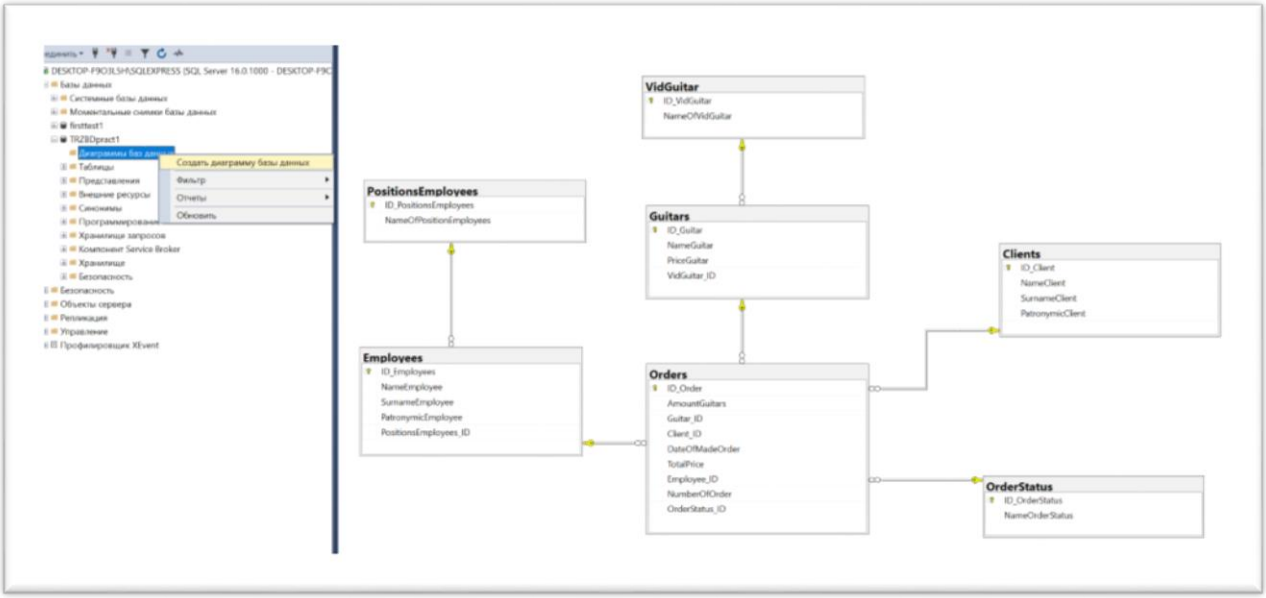


Рисунок 6 – Диаграмма базы данных

3. SELECT-запросы.

Чтобы вывести данные из таблиц используется оператор SELECT. Символ звездочки перед «FROM НАЗВАНИЕ_ТАБЛИЦЫ» позволяет вывести сразу все данные из всех столбцов. Для конкретизации вместо звездочки вводят название нужного столбца (см. Рисунок 8 – SELECT-запросы).

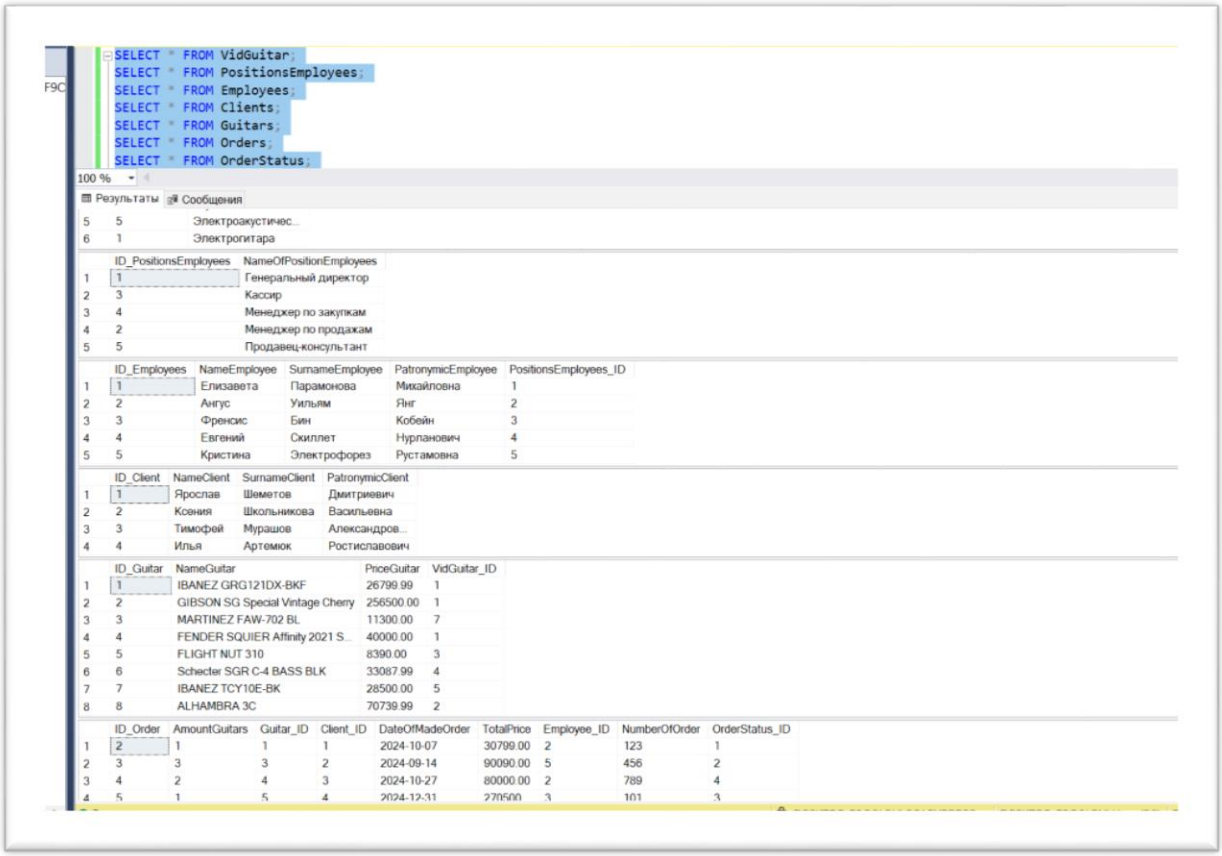


Рисунок 8 – SELECT-запросы

SELECT-запросы можно совершать вместе с командами. Например, для первого к таблице клиентов будет использована команда DISTINCT, исключающая повторения в результате выборки, с помощью оператора «AS» для столбцов написаны «псевдонимы». Для вывода не внешних ключей, а данных, используется оператор «INNER JOIN» с условием «on», выводящий только те строки, для которых были найдены совпадения. «WHERE» - условие к запросу, то есть вывести какие-то данные, которые удовлетворяют требованию, в данном случае выведутся только те данные, где в фамилии клиента есть сочетание букв

«ов» - это сделано через оператор «LIKE» и шаблон «%то что надо найти%», осуществляющий поиск по содержанию сочетания букв в любой части слова. Команда «ORDER BY» сортирует результат по столбцу общей стоимости заказа по возрастанию - ASC (см. Рисунок 9 – Первый SELECT-запрос).

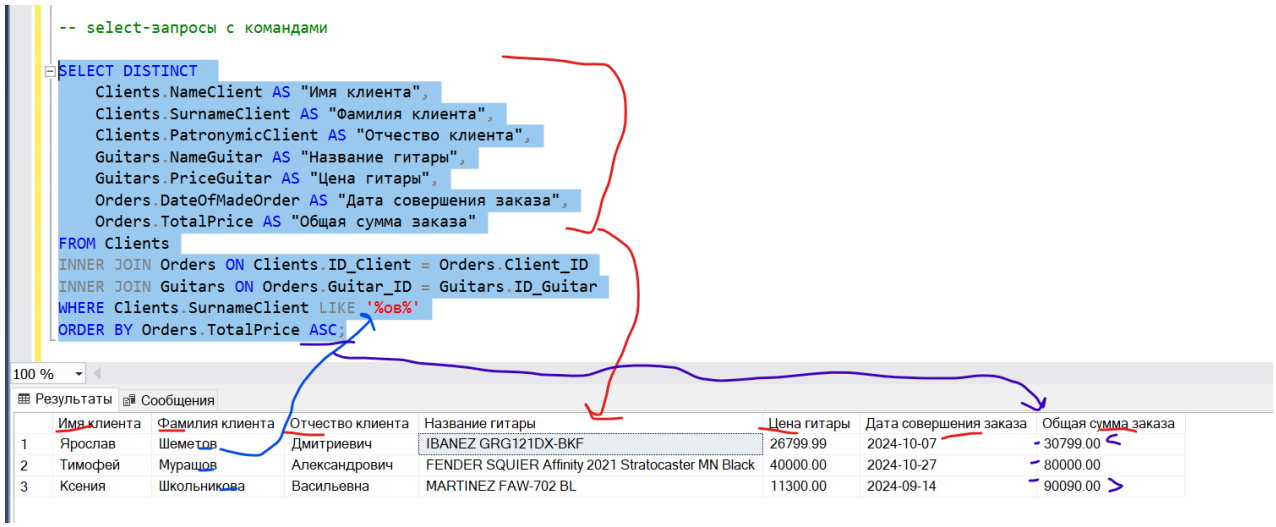


Рисунок 9 – Первый SELECT-запрос

Второй SELECT-запрос выводит имя и фамилию сотрудника из таблицы сотрудников, которые приняли больше одного заказа (сделано через условие HAVING и COUNT – считает количество строк) и сортирует их по количеству принятых заказов по убыванию (см. Рисунок 10 – Второй SELECT-запрос).

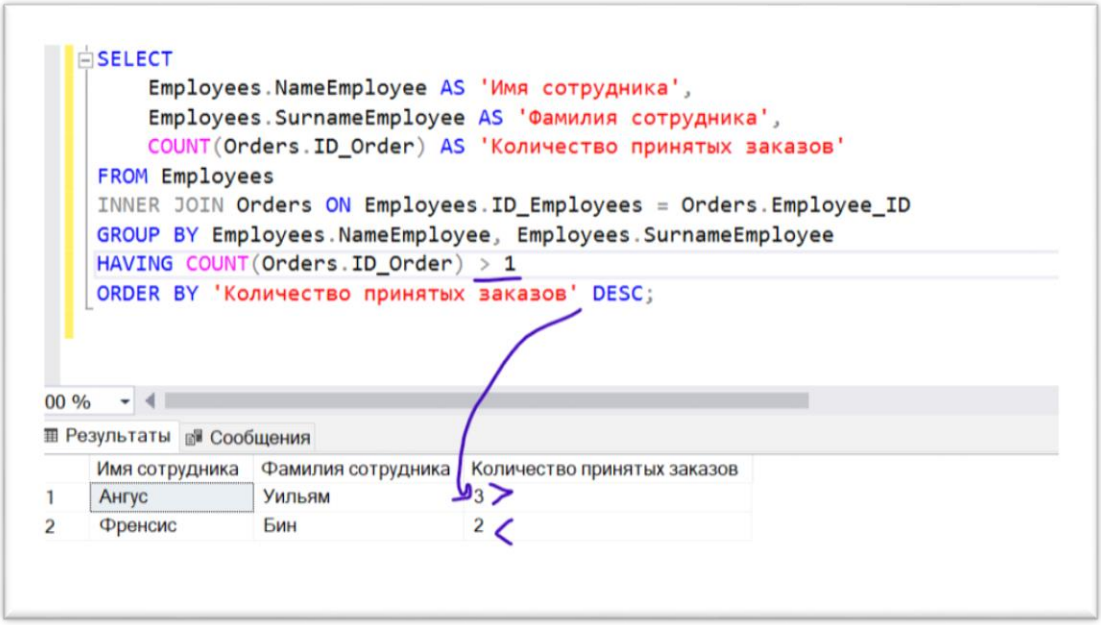
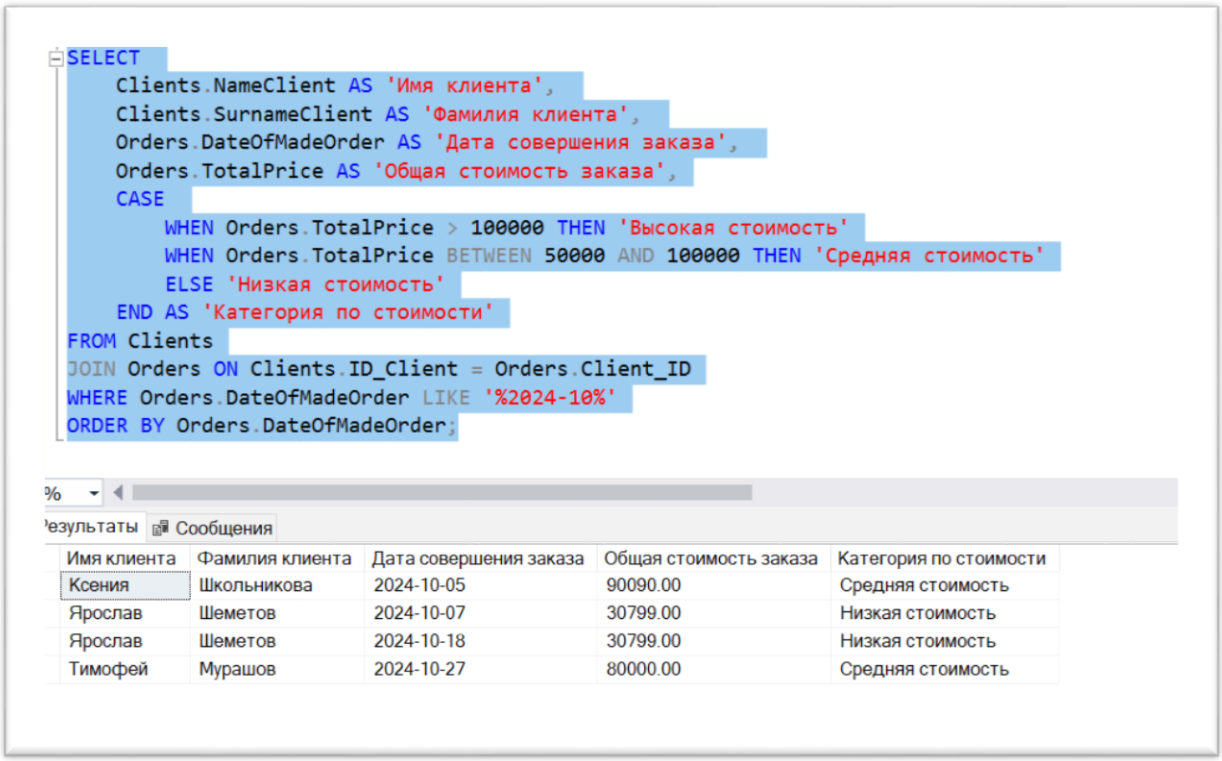


Рисунок 10 – Второй SELECT-запрос

Третий SELECT-запрос с помощью оператора CASE выводит определенный результат: если цена гитары больше ста тысяч – это высокая стоимость, если между 50000 и 100000 тысячами – средняя, все остальные ситуации – низкая стоимость. Столбцы также имею псевдонимы, заказы выводятся только за октябрь 2024 года (см. Рисунок 11 – Третий SELECT-запрос).



```
SELECT
    Clients.NameClient AS 'Имя клиента',
    Clients.SurnameClient AS 'Фамилия клиента',
    Orders.DateOfMadeOrder AS 'Дата совершения заказа',
    Orders.TotalPrice AS 'Общая стоимость заказа',
    CASE
        WHEN Orders.TotalPrice > 100000 THEN 'Высокая стоимость'
        WHEN Orders.TotalPrice BETWEEN 50000 AND 100000 THEN 'Средняя стоимость'
        ELSE 'Низкая стоимость'
    END AS 'Категория по стоимости'
FROM Clients
JOIN Orders ON Clients.ID_Client = Orders.Client_ID
WHERE Orders.DateOfMadeOrder LIKE '%2024-10%'
ORDER BY Orders.DateOfMadeOrder;
```

%

езультаты | Сообщения

Имя клиента	Фамилия клиента	Дата совершения заказа	Общая стоимость заказа	Категория по стоимости
Ксения	Школьникова	2024-10-05	90090.00	Средняя стоимость
Ярослав	Шеметов	2024-10-07	30799.00	Низкая стоимость
Ярослав	Шеметов	2024-10-18	30799.00	Низкая стоимость
Тимофей	Мурашов	2024-10-27	80000.00	Средняя стоимость

Рисунок 11 – Третий SELECT-запрос

4. Агрегатные функции.

Агрегатные функции – это функции, которые выполняют вычисление над набором значений и возвращают одно значение. «COUNT» считает количество строк – в данном случае количество клиентов, «SUM» – выводит сумму из столбца – например общая сумма всех заказов, MAX/MIN выведут максимальное и минимальное значение соответственно, «AVG» выведет среднее значение, здесь – среднюю стоимость заказа, результат будет округлен до двух знаков после запятой с помощью функции для преобразования данных из одного типа

в другой при выполнении определенной операции или сравнения (см. Рисунок 12 – Агрегатные функции).

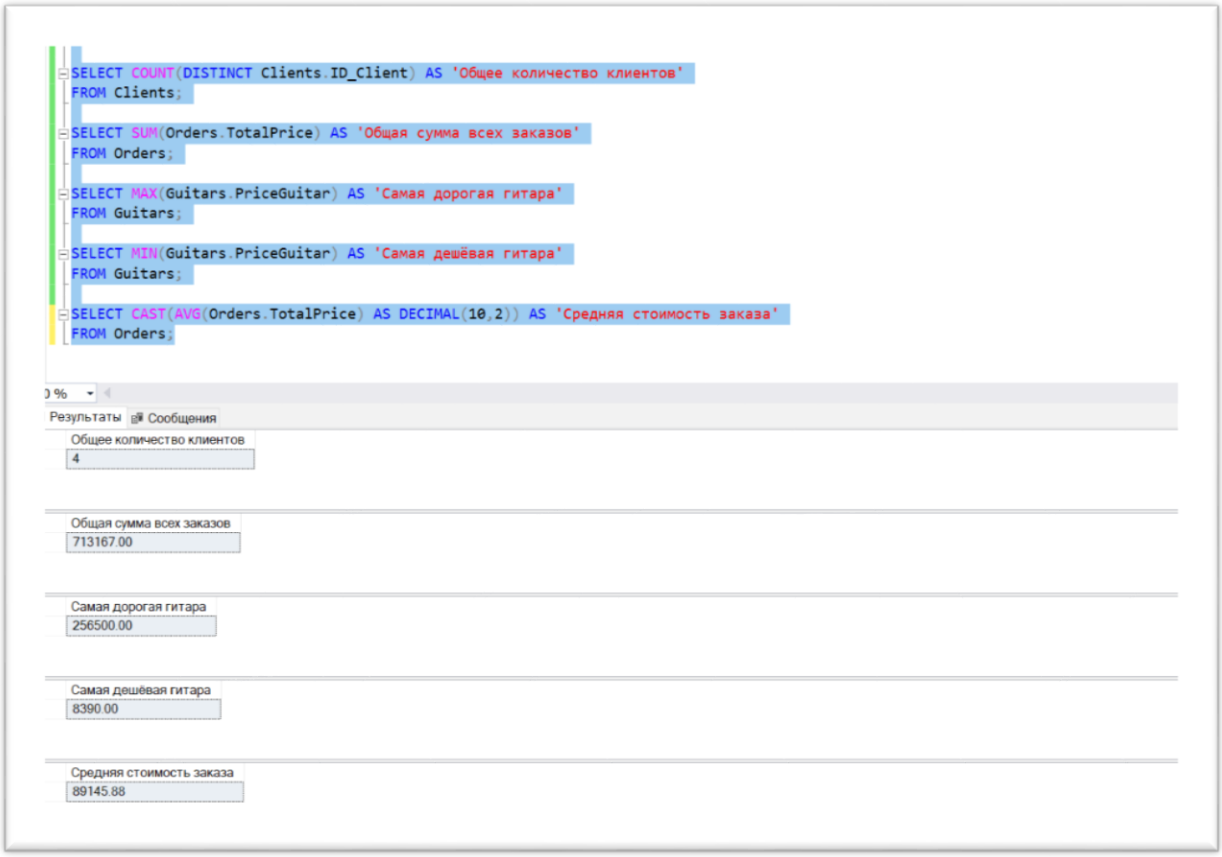
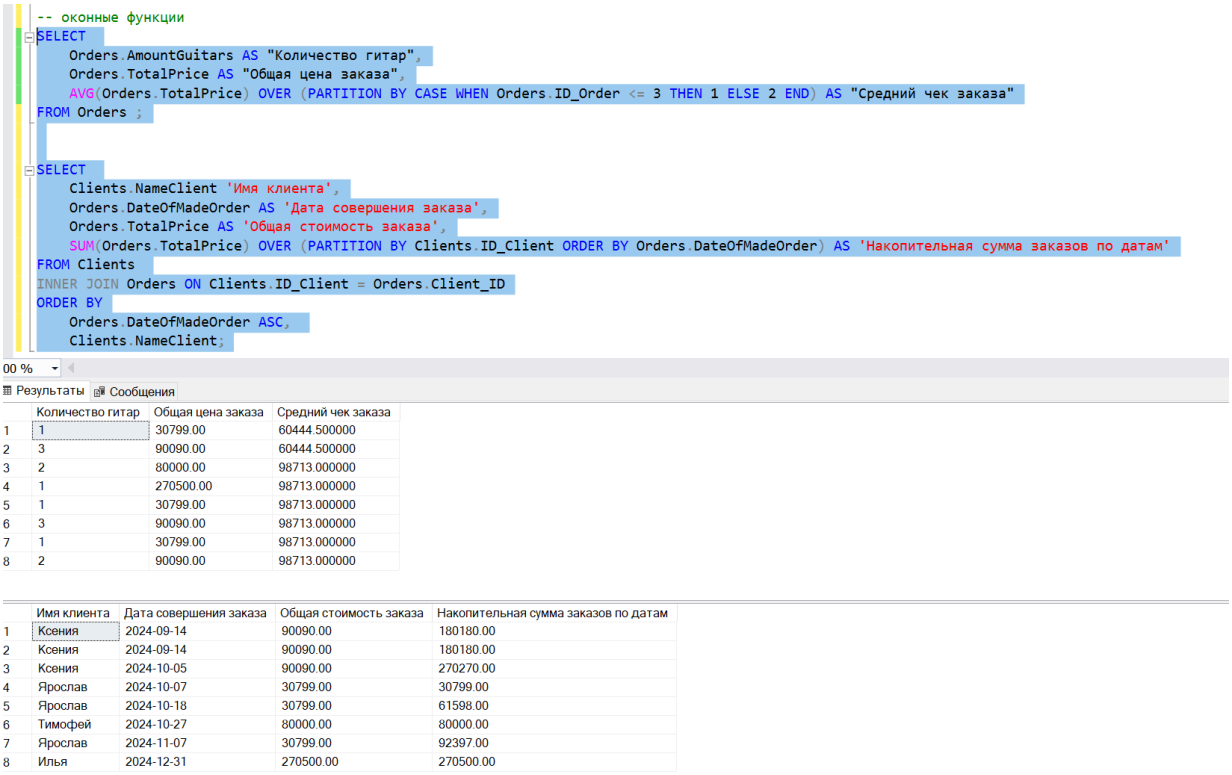


Рисунок 12 – Агрегатные функции

5. Оконная функция.

Оконная функция в SQL - функция, которая работает с выделенным набором строк (окном, партицией) и выполняет вычисление для этого набора строк в отдельном столбце. Партиции (окна из набора строк) - это набор строк, указанный для оконной функции по одному из столбцов или группе столбцов таблицы, прописывается как «..OVER (PARTITION BY..».

Первая оконная функция считает средний чек для групп заказов – сначала для 1-3 по ID, а потом для остальных, все заказы также выводятся в качестве цены и стоимости. Вторая оконная функция выводит информацию о заказах клиентов, включая накопительную сумму покупок каждого клиента, отсортированную по дате заказа – от более ранней к поздней (см. Рисунок 13 – Оконные функции).



```sql
-- оконные функции
SELECT
    Orders.AmountGuitars AS "Количество гитар",
    Orders.TotalPrice AS "Общая цена заказа",
    AVG(Orders.TotalPrice) OVER (PARTITION BY CASE WHEN Orders.ID_Order <= 3 THEN 1 ELSE 2 END) AS "Средний чек заказа"
FROM Orders ;

SELECT
    Clients.NameClient 'Имя клиента',
    Orders.DateOfMadeOrder AS 'Дата совершения заказа',
    Orders.TotalPrice AS 'Общая стоимость заказа',
    SUM(Orders.TotalPrice) OVER (PARTITION BY Clients.ID_Client ORDER BY Orders.DateOfMadeOrder) AS 'Накопительная сумма заказов по датам'
FROM Clients
INNER JOIN Orders ON Clients.ID_Client = Orders.Client_ID
ORDER BY
    Orders.DateOfMadeOrder ASC,
    Clients.NameClient;
```

00 %

Результаты | Сообщения

	Количество гитар	Общая цена заказа	Средний чек заказа
1	1	30799.00	60444.500000
2	3	90090.00	60444.500000
3	2	80000.00	98713.000000
4	1	270500.00	98713.000000
5	1	30799.00	98713.000000
6	3	90090.00	98713.000000
7	1	30799.00	98713.000000
8	2	90090.00	98713.000000

	Имя клиента	Дата совершения заказа	Общая стоимость заказа	Накопительная сумма заказов по датам
1	Ксения	2024-09-14	90090.00	180180.00
2	Ксения	2024-09-14	90090.00	180180.00
3	Ксения	2024-10-05	90090.00	270270.00
4	Ярослав	2024-10-07	30799.00	30799.00
5	Ярослав	2024-10-18	30799.00	61598.00
6	Тимофей	2024-10-27	80000.00	80000.00
7	Ярослав	2024-11-07	30799.00	92397.00
8	Илья	2024-12-31	270500.00	270500.00

Рисунок 13 – Оконные функции

Вывод: в ходе выполнения данной практической работы была создана база данных по теме «Гитарный магазин», проведены три запроса SELECT с использованием команд (ORDER BY, GROUP BY, HAVING, WHERE, DISTINCT, LIKE, CASE), была продемонстрирована работа агрегатных функций, также был проведён запрос с использованием оконной функции на языке SQL.