**Федеральное государственное образовательное бюджетное**

**учреждение высшего образования**

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

ОТЧЁТ

По учебной практике

Специальность 09.02.07 «Информационные системы и программирование»

(код) (наименование)

Профессиональный модуль ПМ.11 Разработка, администрирование и защита баз данных

(код) (наименование)

Междисциплинарный курс МДК.11.01 Технология разработки и защиты баз данных

(код) (наименование)

Выполнил:

Студент ­3 курса 3ИСИП-422 учебной группы

(номер) (номер)

\_\_\_\_\_\_\_\_\_\_ А. А. Сидаков

(подпись) (инициалы, фамилия)

Проверил:

Руководитель практики от Колледжа информатики и программирования

Преподаватель 1КК Н.А. Хасанова (квалификационная (инициалы, фамилия)

категория или звание,

должность)

Москва – 20­25

**Перечень работ, выполненных в ходе учебной практики**

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Виды работ** | **Оценка** |
| 1 | Подготовка, обработка и импорт данных в SQL Server |  |
| 2 | Восстановление БД из скрипта. Подготовка, обработка и импорт данных в SQL Server |  |
| 3 | Восстановление БД из скрипта. Создание таблиц базы данных. Изменение существующей таблицы |  |
| 4 | Подготовка, обработка и импорт данных в SQL Server |  |
| 5 | Ограничения на множество допустимых значений |  |
| 6 | Операторы манипулирования данными. Создание запросов на выборку данных |  |
| 7 | Внутренние соединения. Внешние соединения |  |
| 8 | Определение цели и области применения, определение сущностей, определение связей между ними, добавление атрибутов, указание типов связей. Нормализация |  |
| 9 | Создание резервной копии, восстановление БД из резервной копии |  |
| 10 | Табличные выражения, представления, функции, производные таблицы. Хранимые процедуры, триггеры. Применение стандартных методов для защиты объектов базы данных |  |
| 11 | Шифрование |  |

**СОДЕРЖАНИЕ**

[Отчет о выполнении работы 1 4](#_Toc198411020)

[Отчет о выполнении работы 2 10](#_Toc198411021)

[Отчет о выполнении работы 3 18](#_Toc198411022)

[Отчет о выполнении работы 4 22](#_Toc198411023)

[Отчет о выполнении работы 5 31](#_Toc198411024)

[Отчет о выполнении работы 6 34](#_Toc198411025)

[Отчет о выполнении работы 7 42](#_Toc198411026)

[Отчет о выполнении работы 8 52](#_Toc198411027)

## **Отчет о выполнении работы 1**

1. Сначала, создаем базу данных. Создание показано на рисунке 1.

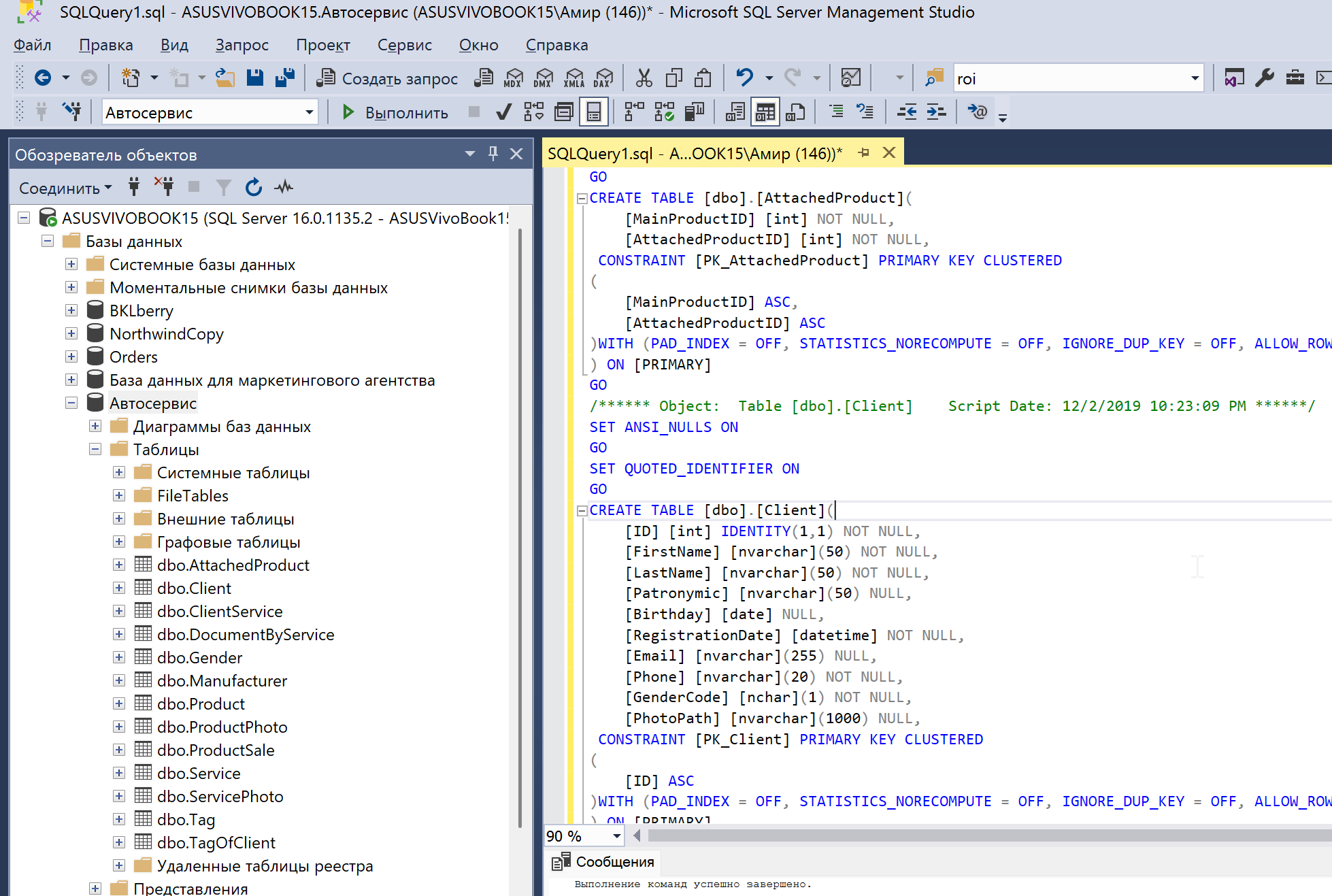


Рисунок 1 — Создание бд

1. Затем, подготавливаем данные для импорта из Manufacturer: добавляем столбец и сохраняем файл в формате Excel 97-2003.

Подготовленные данные показаны на рисунке 2.



Рисунок 2 — Подготовка Manufacturer

1. Затем, подготавливаем данные для импорта из Product: добавляем столбец, приводим к нормальному виду столбцы: «Производитель», «Активен» и «Цена». После сохраняем файл в формате Excel 97-2003.

Подготовленные данные показаны на рисунке 3.



Рисунок 3 — Подготовка Product

1. Почти такие же, действия мы проводим с данными из ProductSale.

Подготовленные данные показаны на рисунке 4.

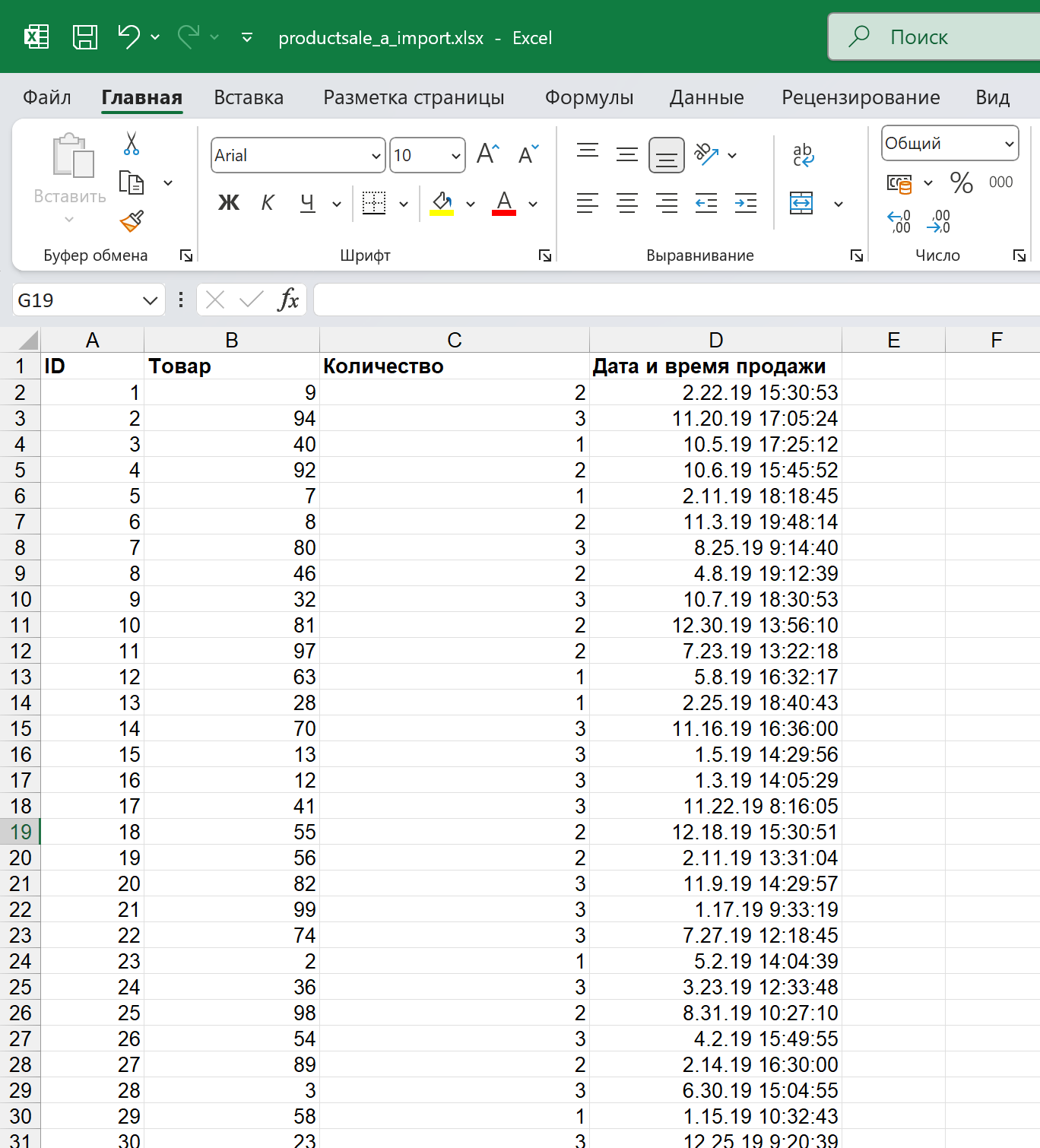


Рисунок 4 — Подготовка ProductSale

1. Импортируем данные и проверяем все ли правильно вставилось. Не возникло ли никаких ошибок.

Проверяем данные на рисунках 5-7.

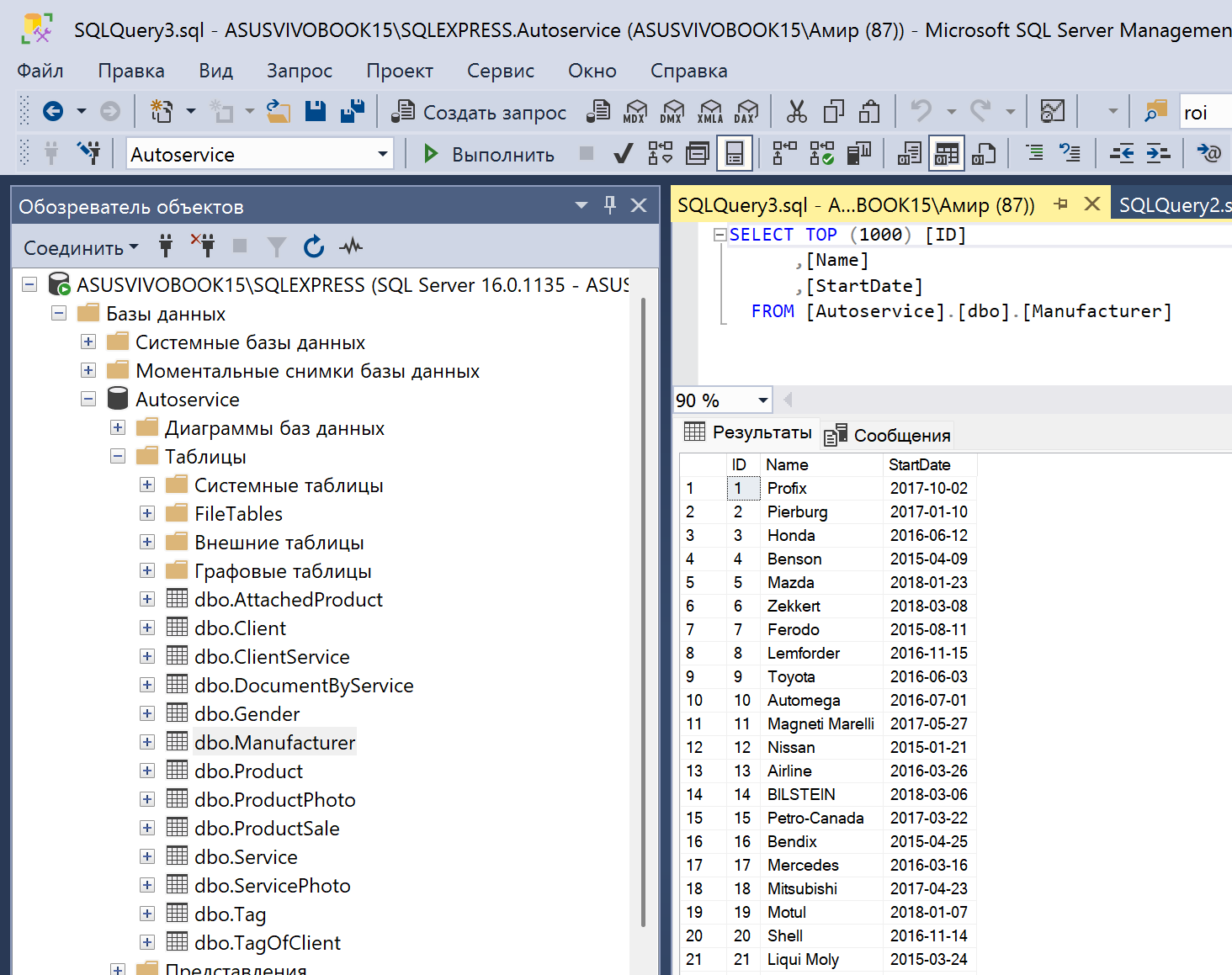


Рисунок 5 — Проверка Manufacturer

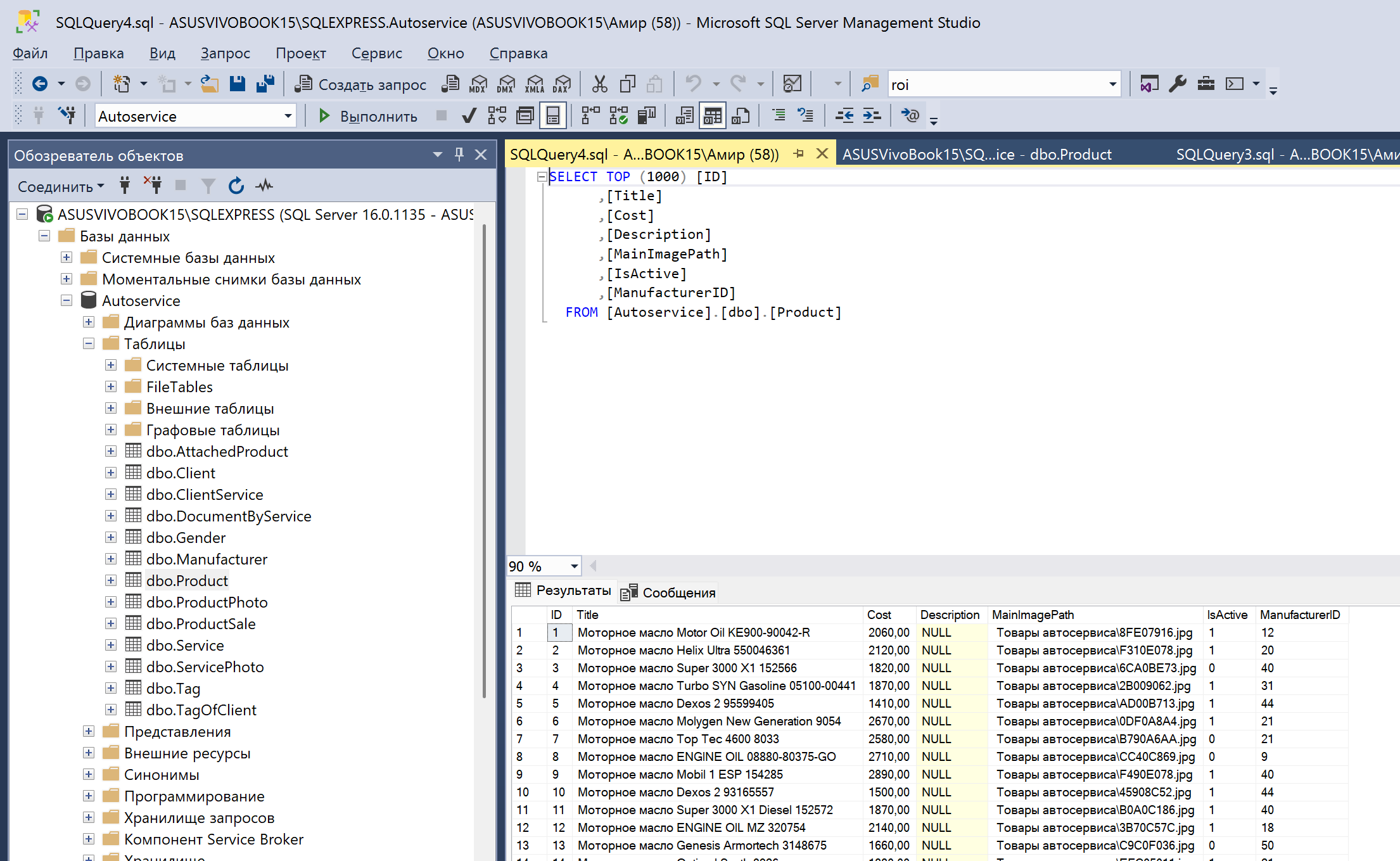


Рисунок 6 — Проверка Product

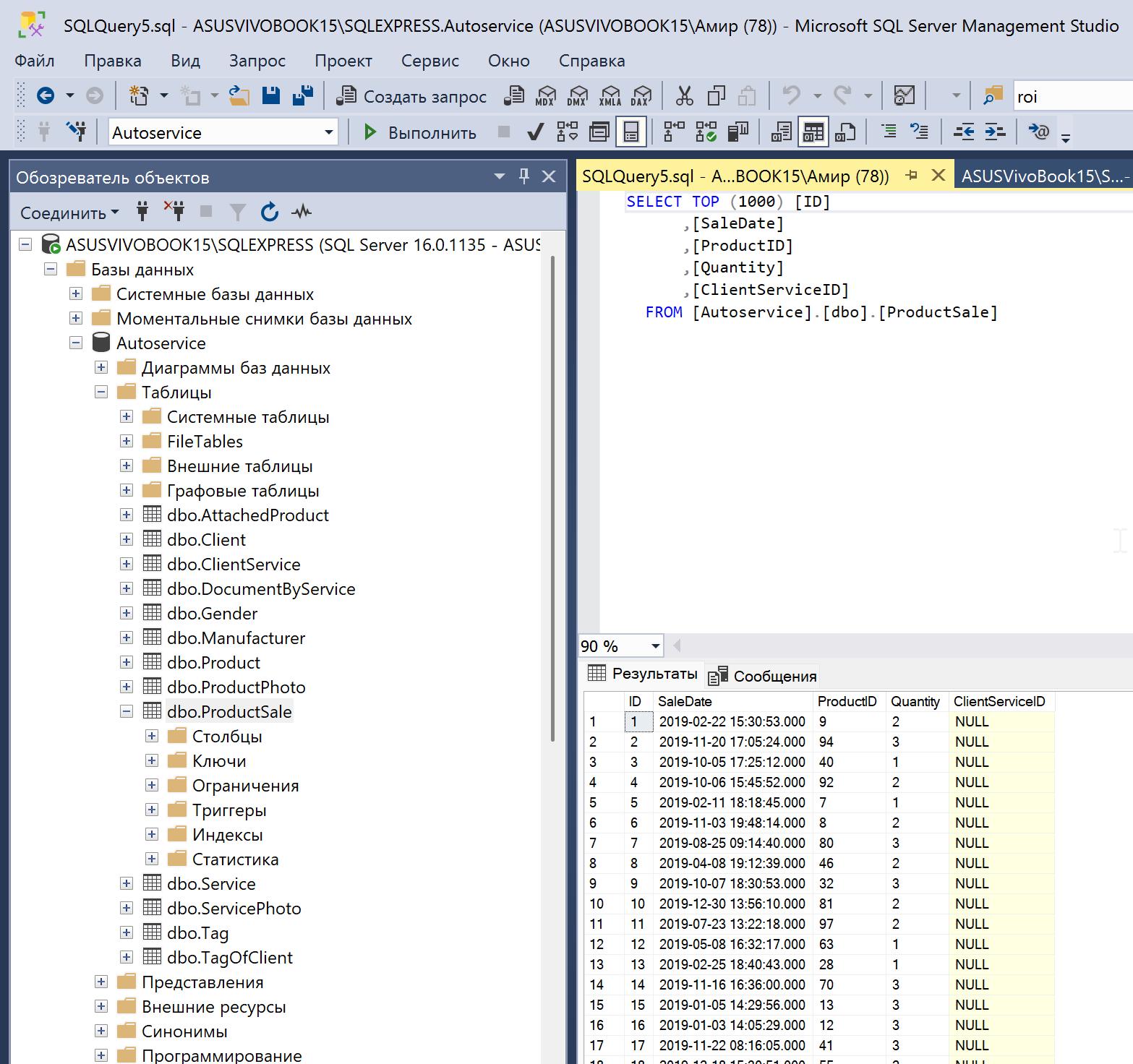


Рисунок 7 — Проверка ProductSale

1. Затем перейдем к написанию Запросов.

Листинг 1 — Посчитать количество продаж каждого товара.

|  |
| --- |
| SELECT  ProductID,  SUM(Quantity) as TotalSales  FROM [AutoService].[dbo].[ProductSale]  GROUP BY ProductID  ORDER BY TotalSales DESC; |

Результат показан на рисунке 8.

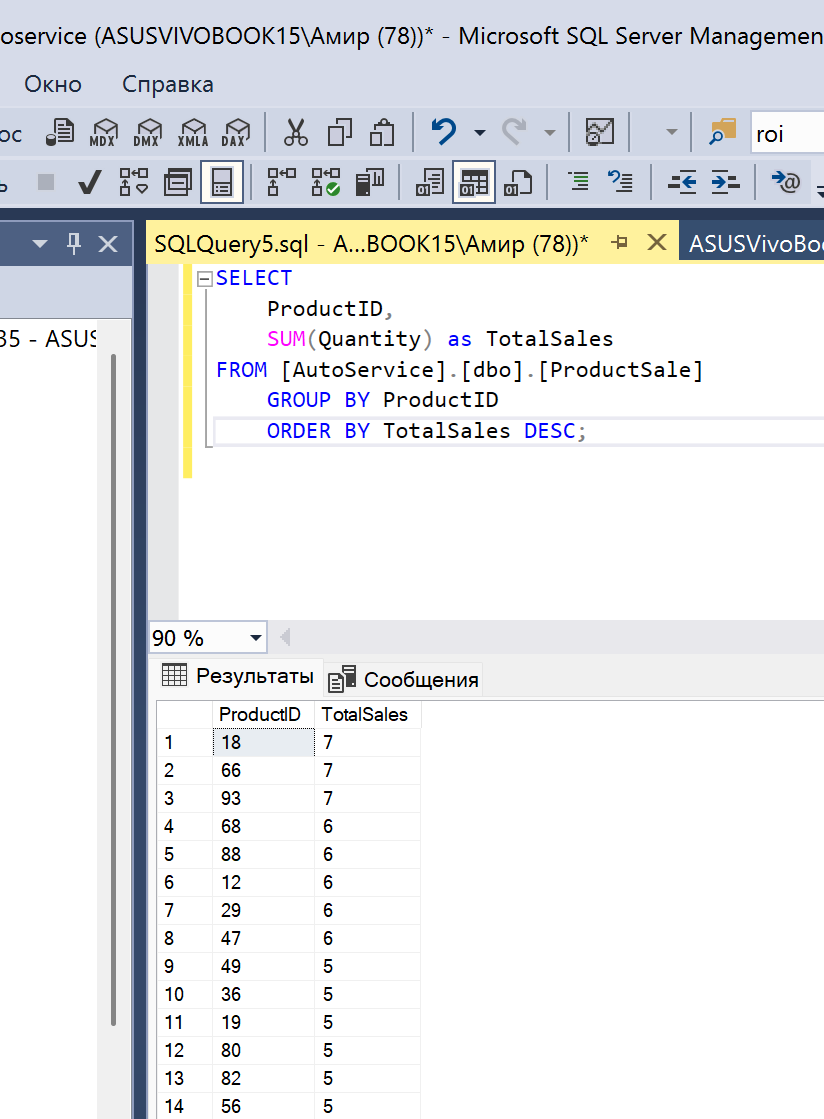


Рисунок 8 — количество продаж каждого товара

Листинг 2 — Вывести товары, которые не были проданы.

|  |
| --- |
| SELECT  p.ID,  p.Title,  p.Cost,  p.Description  FROM dbo.Product p  LEFT JOIN dbo.ProductSale ps  ON p.ID = ps.ProductID WHERE ps.ProductID IS NULL; |

Результат показан на рисунке 9.

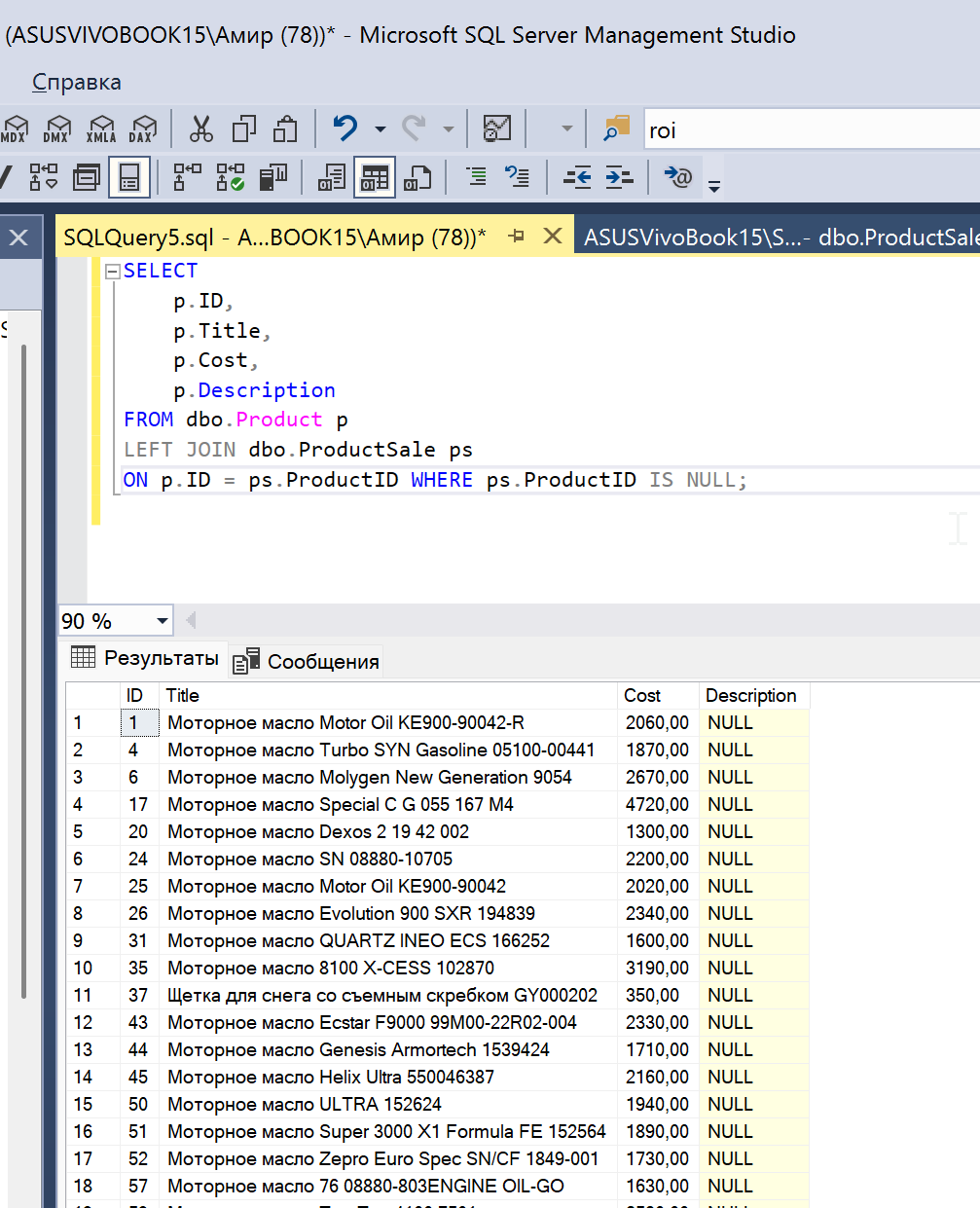


Рисунок 9 ­— Не проданные товары

Листинг 3 — Вывести среднюю стоимость товаров каждого производителя.

|  |
| --- |
| SELECT  m.Name AS ManufacturerName,  AVG(p.Cost) AS AverageCost  FROM dbo.Product p  LEFT JOIN dbo.Manufacturer m  ON p.ManufacturerID = m.ID GROUP BY m.Name; |

Результат показан на рисунке 10.

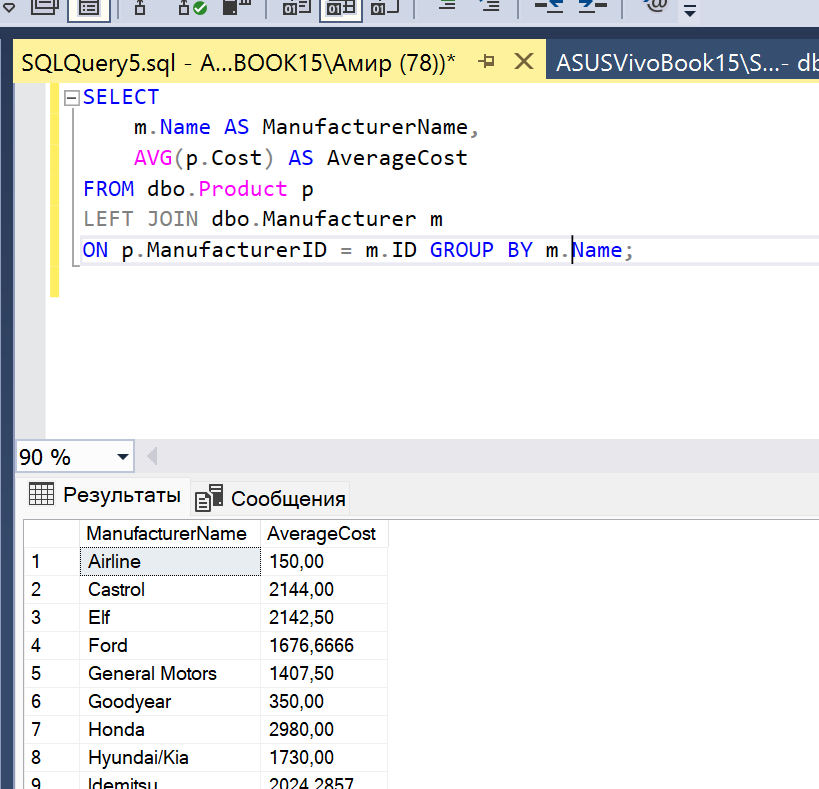


Рисунок 10 ­— Средняя стоимость товаров у производителей

Листинг 4 — Найти месяц, в котором было продано больше всего товаров.

|  |
| --- |
| SELECT TOP 1  YEAR(ps.SaleDate) AS SaleYear,  MONTH(ps.SaleDate) AS SaleMonth,  SUM(ps.Quantity) AS TotalSold  FROM dbo.ProductSale ps  GROUP BY YEAR(ps.SaleDate), MONTH(ps.SaleDate)  ORDER BY TotalSold DESC; |

Результат показан на рисунке 11.

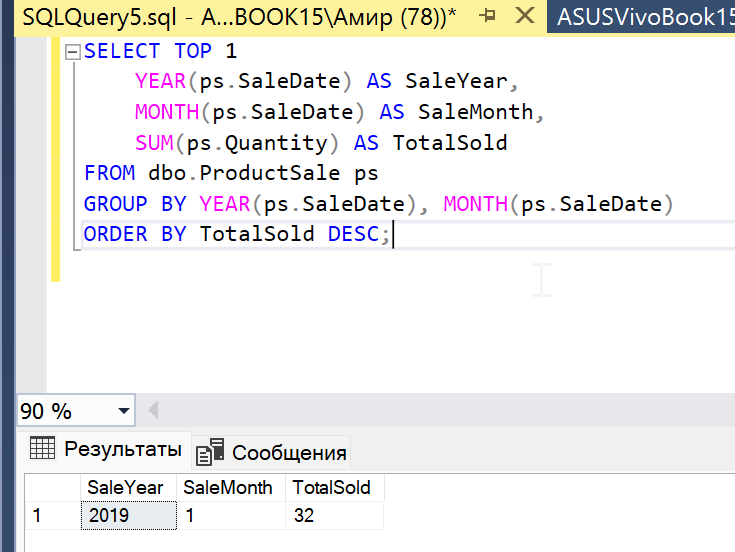


Рисунок 11 — месяц, в котором было продано больше всего товаров

## **Отчет о выполнении работы 2**

1. Сначала, создаем базу данных. Создание показано на рисунке 1.

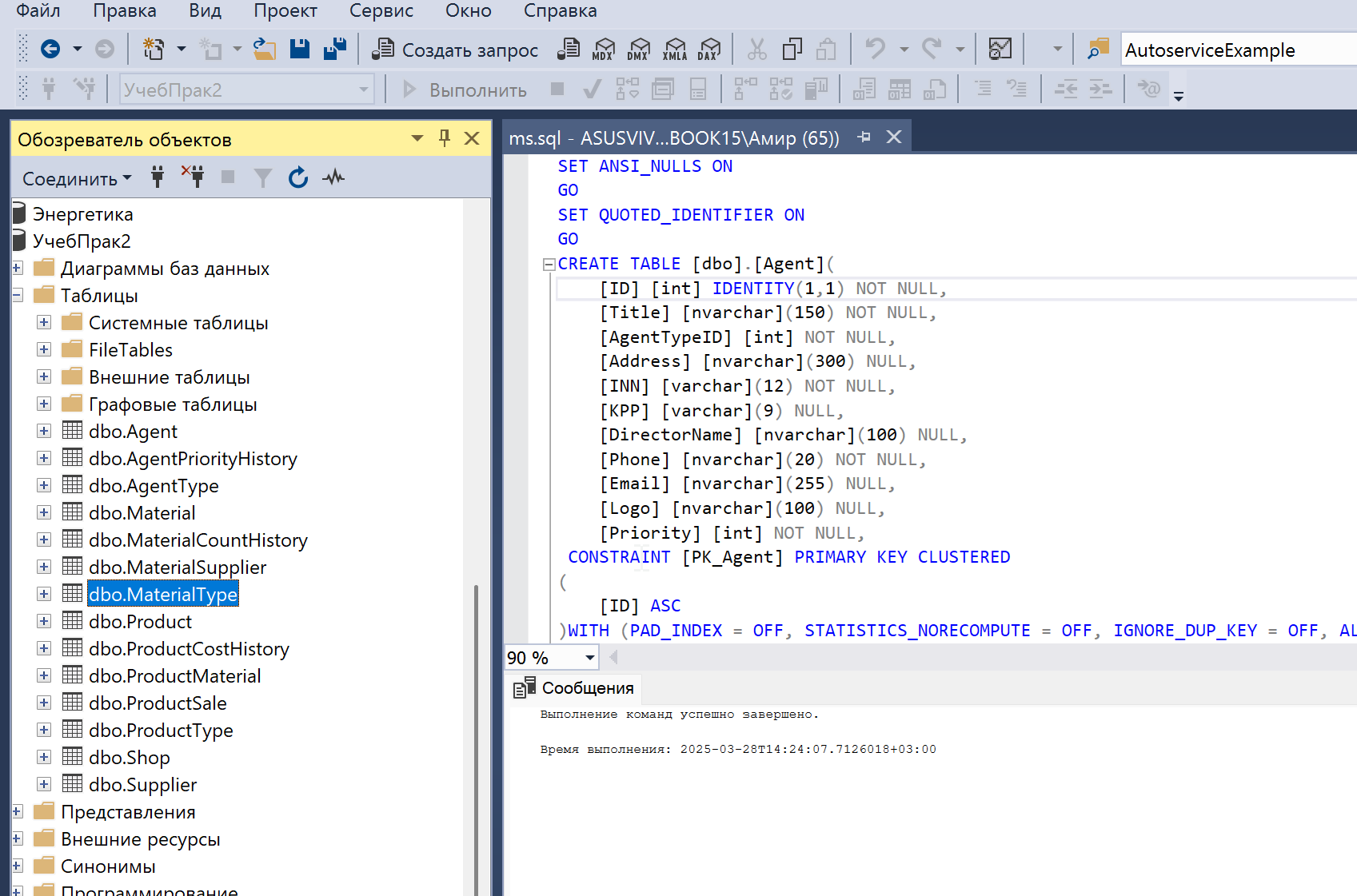


Рисунок 1 — Создание бд

1. Затем, подготавливаем данные Material для импорта и сохраняем файл в формате Excel 97-2003.

Подготовленные данные показаны на рисунке 2.

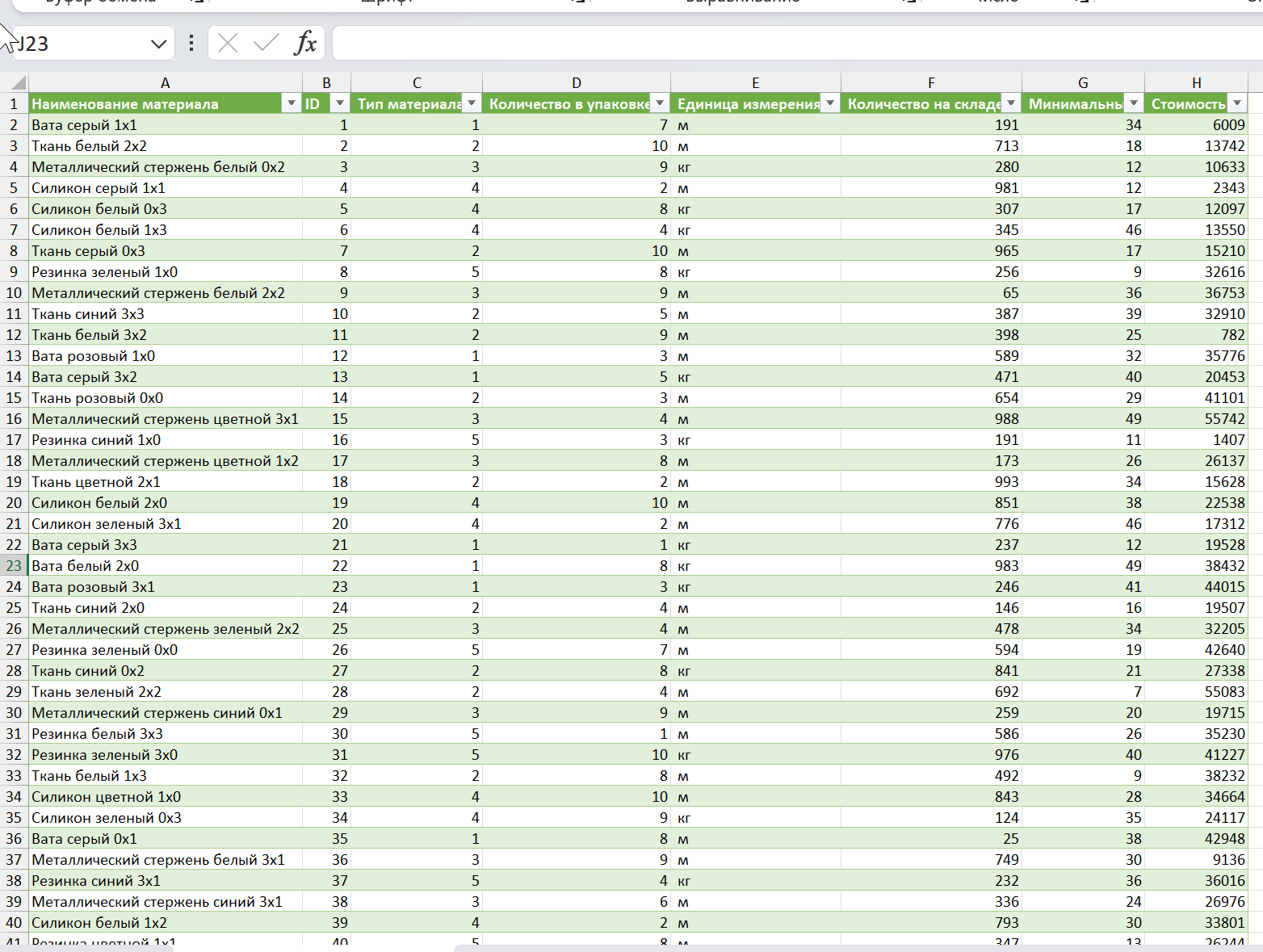


Рисунок 2 — Подготовка Material

Вспомогательная таблица для подготовки основных данных, показана на рисунке 3.

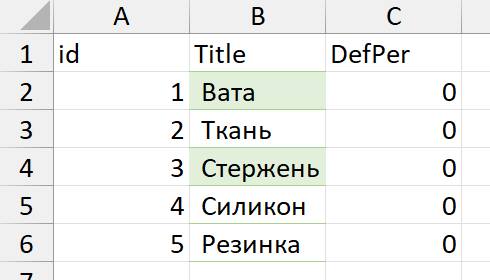


Рисунок 3 – Вспомогательная таблица для Material

1. Затем, подготавливаем данные для импорта из Product: добавляем столбец, приводим к нормальному виду столбцы. После сохраняем файл в формате Excel 97-2003.

Подготовленные данные показаны на рисунке 4.

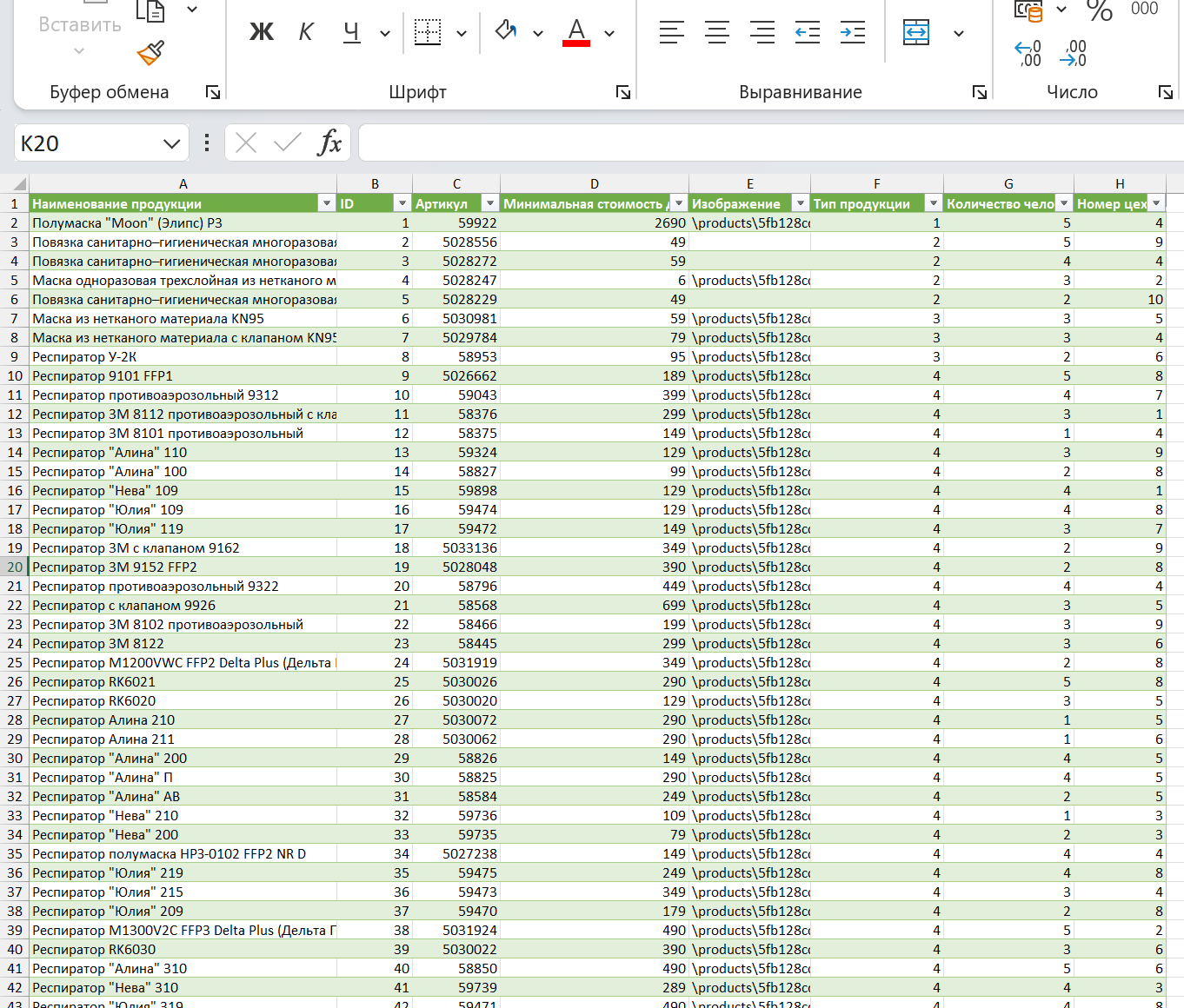


Рисунок 4 — Подготовка Product

Вспомогательная таблица для подготовки основных данных, показана на рисунке 5.

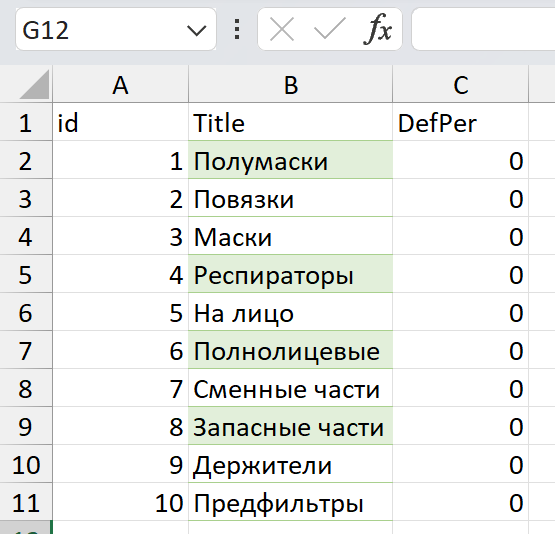


Рисунок 5 – Вспомогательная таблица для Product

1. Почти такие же, действия мы проводим с данными из ProductSale.

Подготовленные данные показаны на рисунке 4.

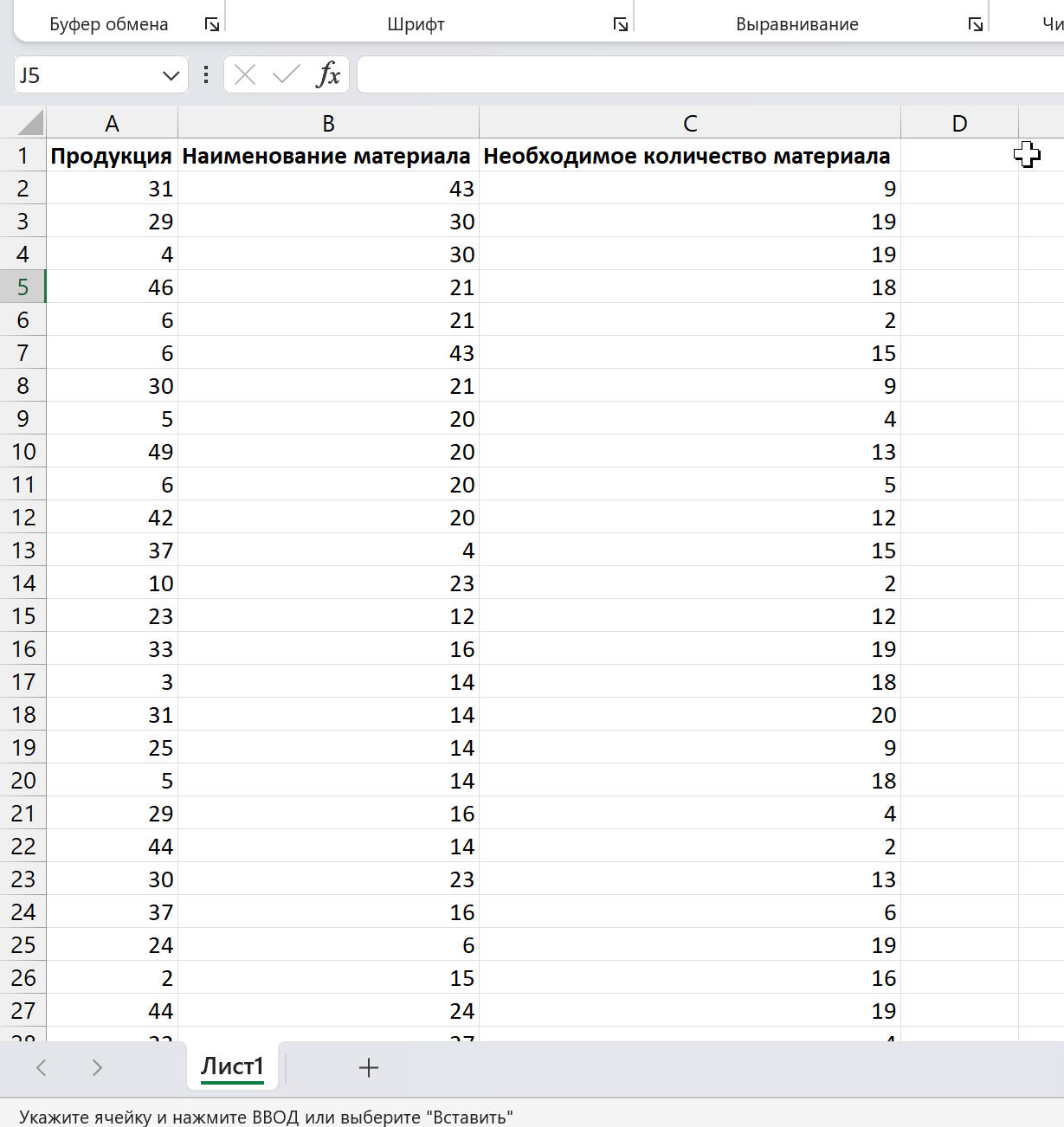


Рисунок 6 — Подготовка ProductMaterial

1. Импортируем данные и проверяем все ли правильно вставилось. Не возникло ли никаких ошибок.

Проверяем правильность импортированных данных.

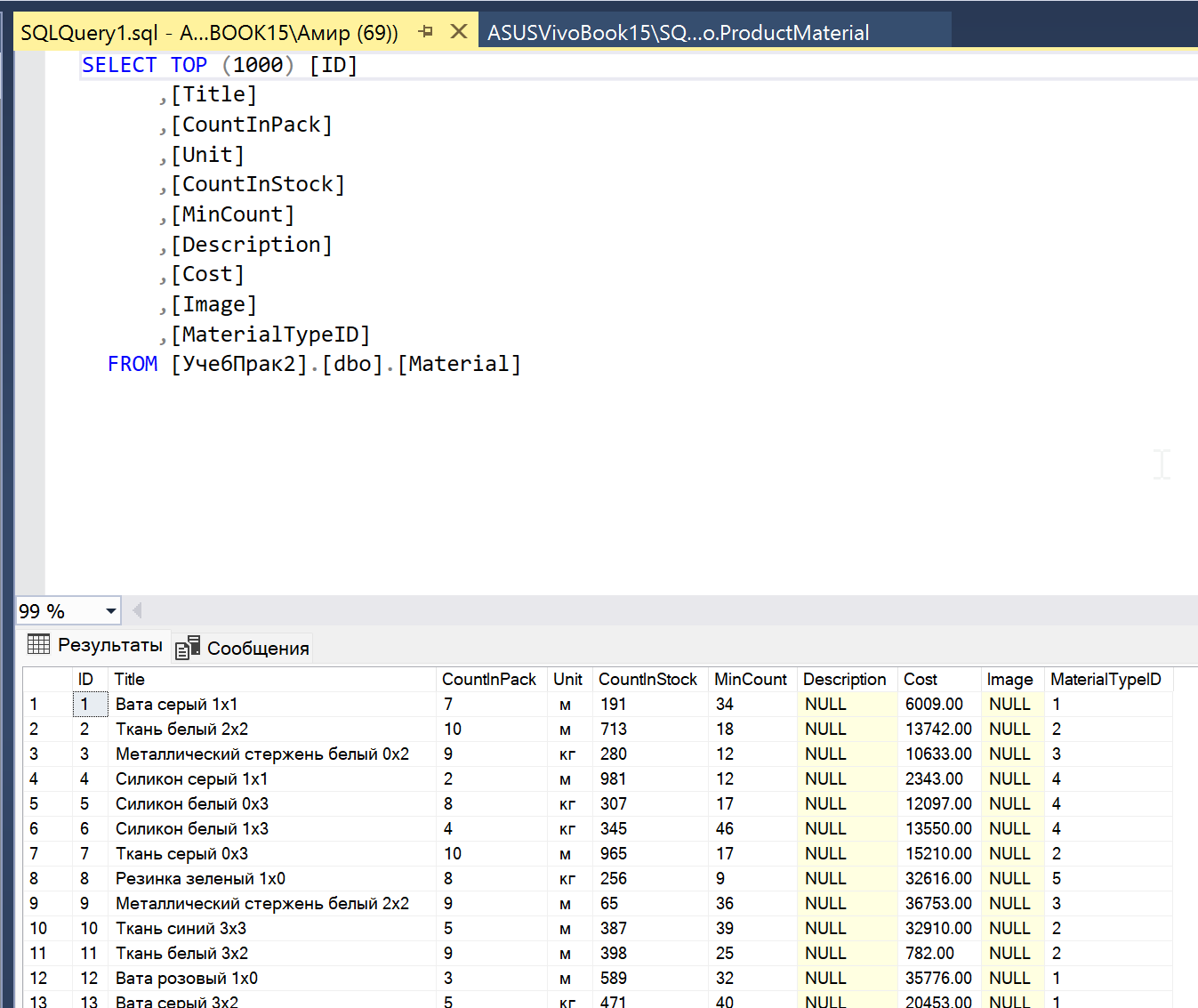


Рисунок 7 — Проверка Material

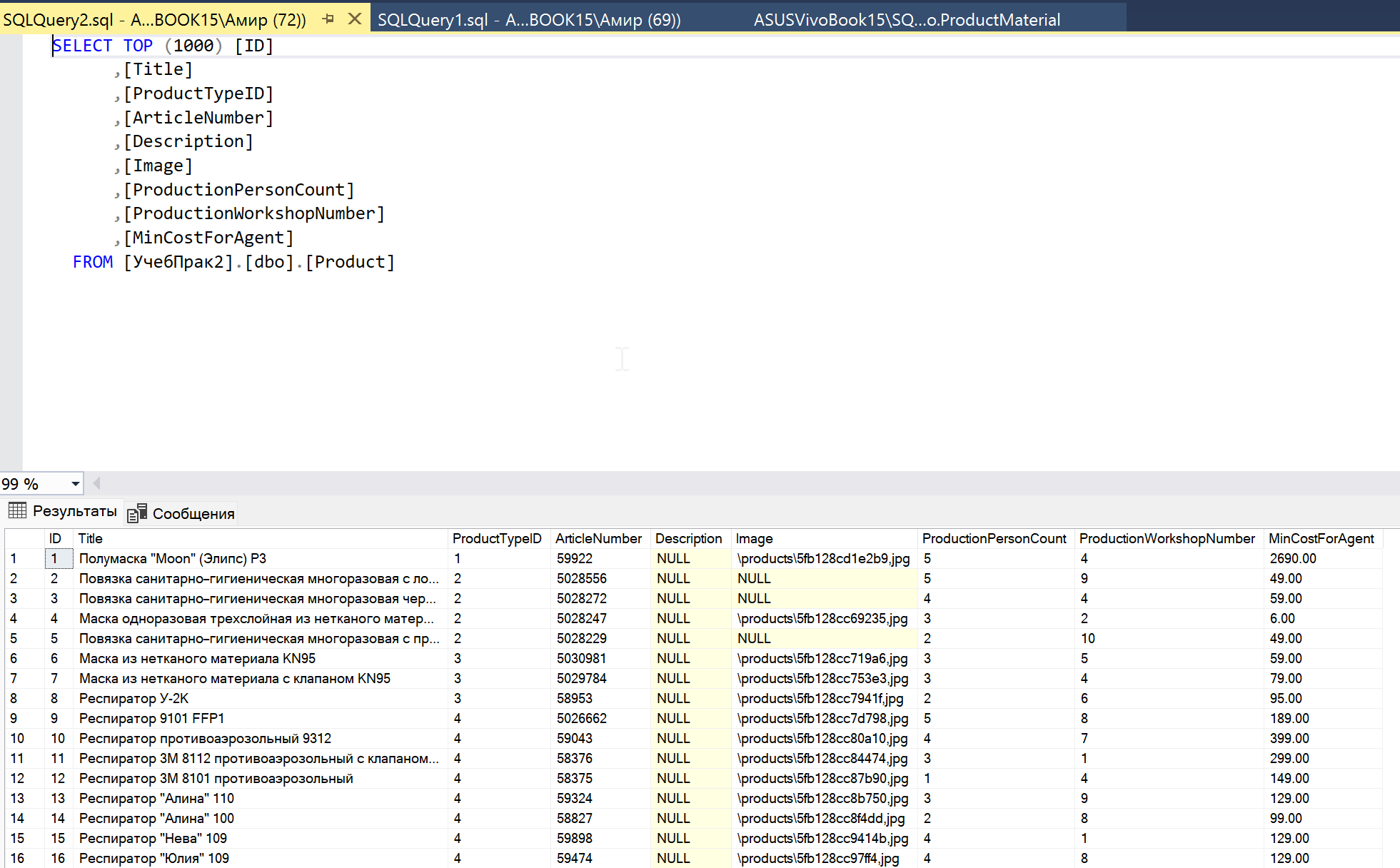


Рисунок 8 — Проверка Product

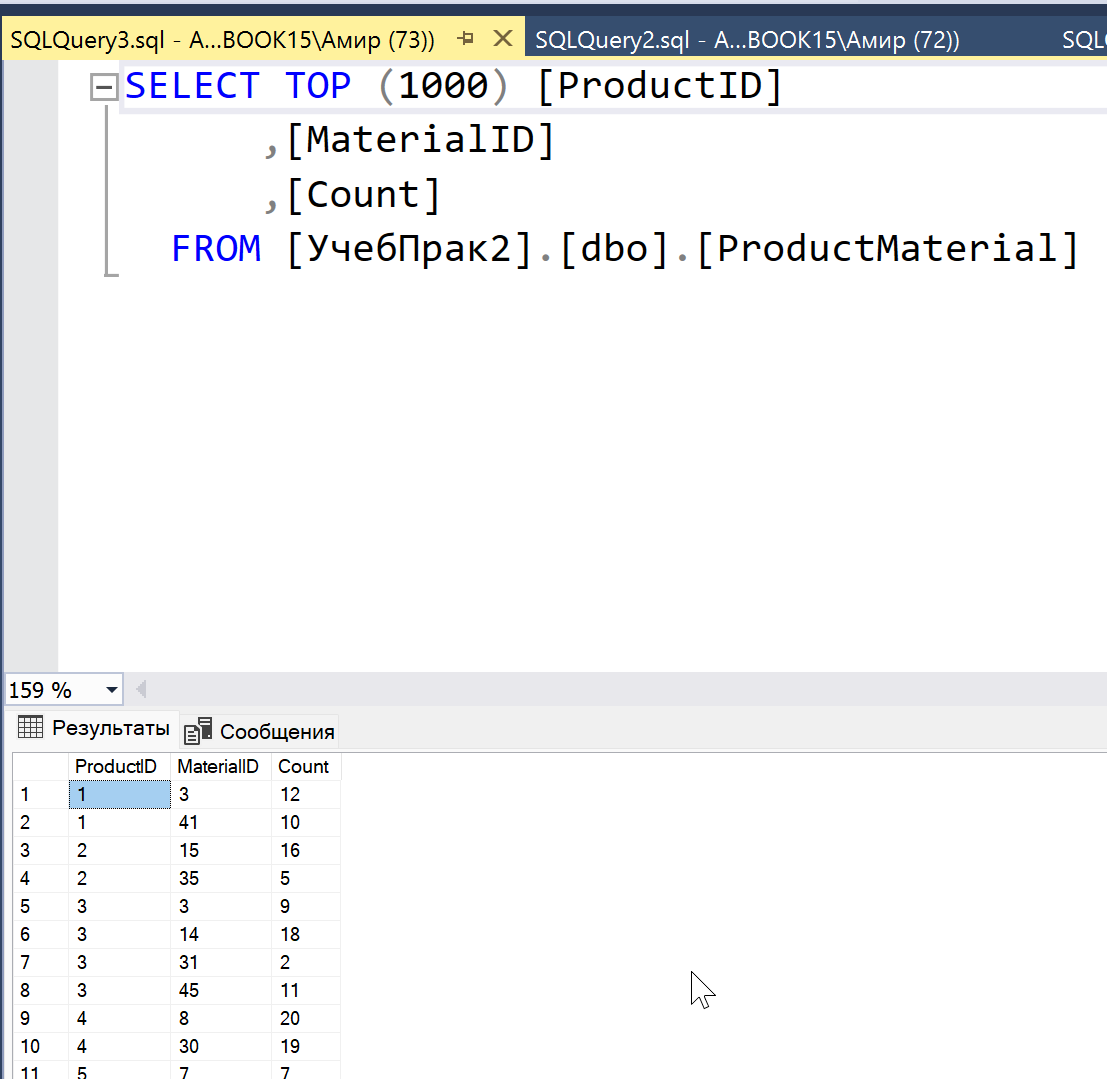


Рисунок 9 — Проверка ProductMaterial

1. Затем перейдем к написанию Запросов.
2. Листинг 1 — Получение всех продуктов с их типами.

|  |
| --- |
| SELECT  p.ID AS ProductID,  p.Title AS ProductName,  p.ArticleNumber,  p.Description AS ProductDescription,  p.MinCostForAgent,  pt.ID AS ProductTypeID,  pt.Title AS ProductTypeName,  pt.DefectedPercent  FROM  [dbo].[Product] p  LEFT JOIN  [dbo].[ProductType] pt ON p.ProductTypeID = pt.ID  ORDER BY  p.Title |

Результат показан на рисунке 10.

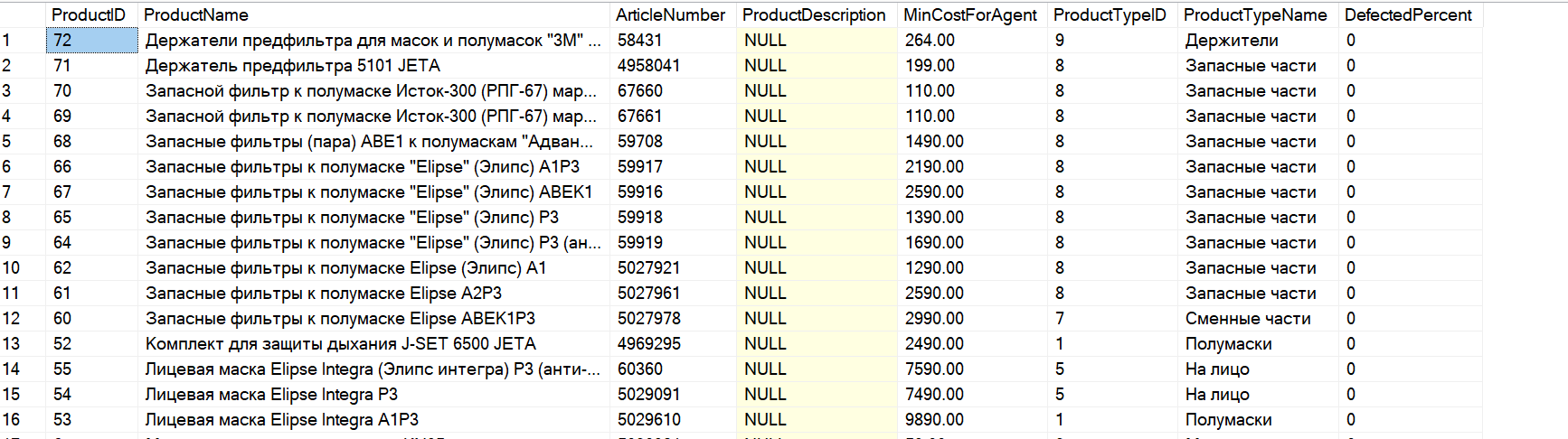


Рисунок 10 — Продукты и их тип

Листинг 2 — Получение списка продуктов и их материалов.

|  |
| --- |
| SELECT  p.ID AS ProductID,  P.Title AS ProductName,  P.ArticleNumber,  Mt.Title AS MaterialType,  M.Title AS MaterialName,  M.Unit,  M.Cost AS MaterialCost  FROM [dbo].[Product] P  JOIN [dbo].[ProductMaterial] Pm  ON Pm.ProductID = P.ID  JOIN [dbo].[Material] M  ON M.ID = Pm.MaterialID  JOIN [dbo].[MaterialType] Mt  ON Mt.ID = M.MaterialTypeID  ORDER BY  P.Title, M.Title |

Результат показан на рисунке 11.

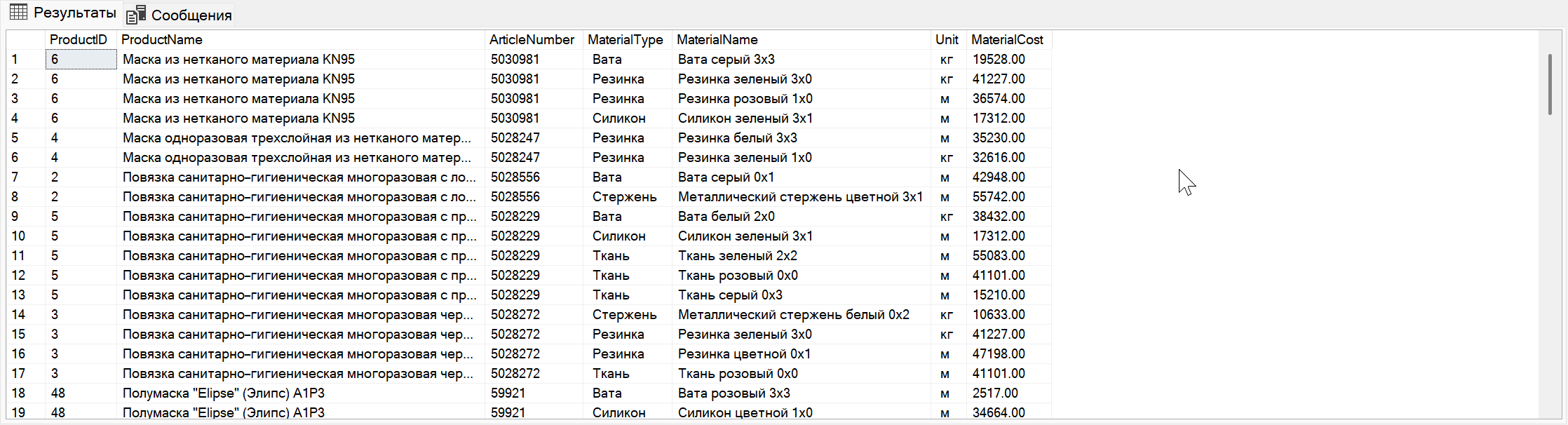


Рисунок 11 ­— Продукты и их материал

Листинг 3 — Подсчет количества материалов в каждом продукте.

|  |
| --- |
| SELECT  p.ID AS 'ID продукта',  p.Title AS 'Название продукта',  p.ArticleNumber AS 'Артикул',  COUNT(pm.MaterialID) AS 'Количество материалов',  SUM(pm.Count) AS 'Общее количество материалов (сумма)'  FROM  [dbo].[Product] p  INNER JOIN  [dbo].[ProductMaterial] pm ON p.ID = pm.ProductID  GROUP BY  p.ID, p.Title, p.ArticleNumber  ORDER BY  p.Title |

Результат показан на рисунке 12.

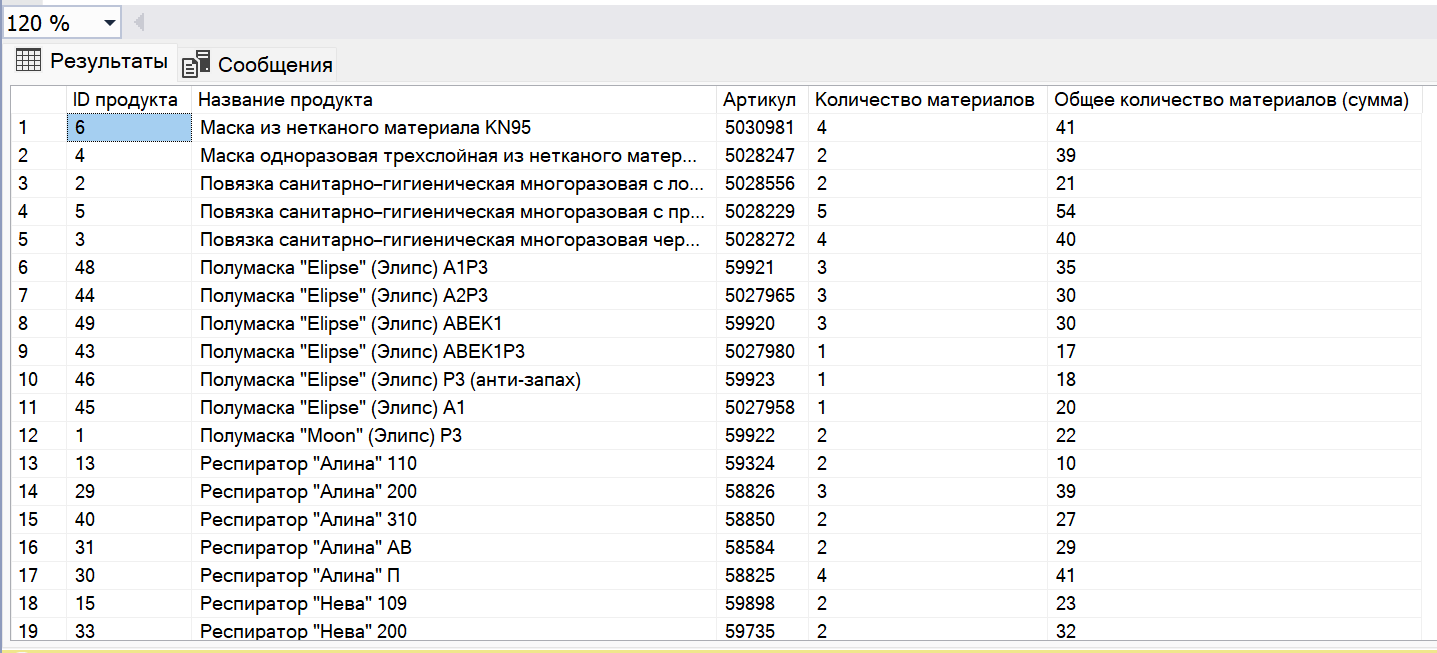


Рисунок 12 ­— Количество материалов в каждом продукте

Листинг 4 — Поиск материалов с минимальным запасами

|  |
| --- |
| SELECT  \*  FROM [dbo].[Material] M  WHERE M.CountInStock < M.MinCount  ORDER BY  (M.MinCount - M.CountInStock) DESC |

Результат показан на рисунке 13.

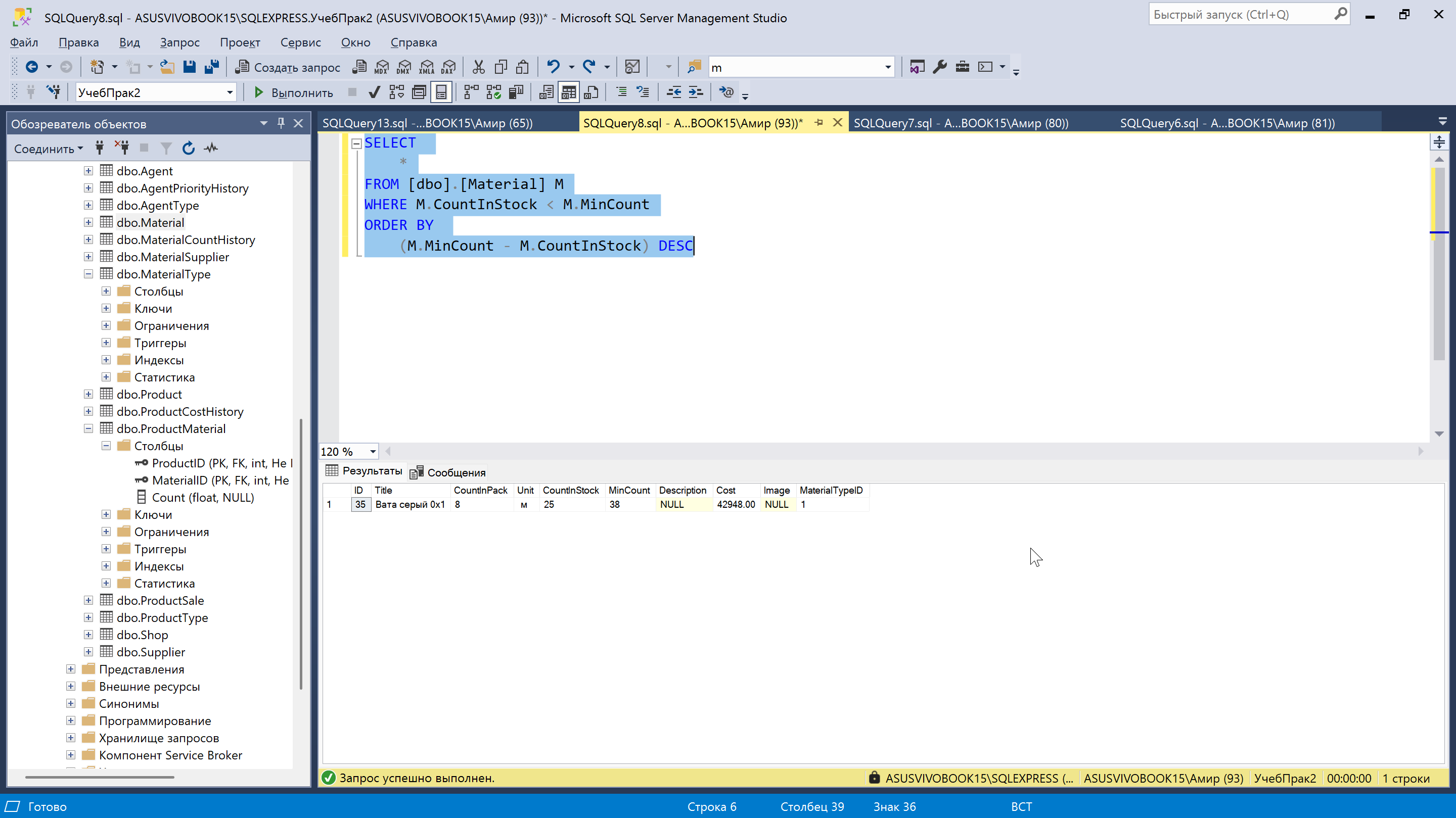


Рисунок 13 — Материалы с минимальными запасами

Листинг 5 — Получение продуктов, у которых нет материалов

|  |
| --- |
| SELECT  p.ID AS ProductID,  P.Title AS ProductName,  P.ArticleNumber,  Mt.Title AS MaterialType,  M.Title AS MaterialName,  M.Unit,  M.Cost AS MaterialCost  FROM [dbo].[Product] P  LEFT JOIN [dbo].[ProductMaterial] Pm  ON Pm.ProductID = P.ID  LEFT JOIN [dbo].[Material] M  ON M.ID = Pm.MaterialID  LEFT JOIN [dbo].[MaterialType] Mt  ON Mt.ID = M.MaterialTypeID  WHERE  M.Title IS NULL |

Результат показан на рисунке 14.

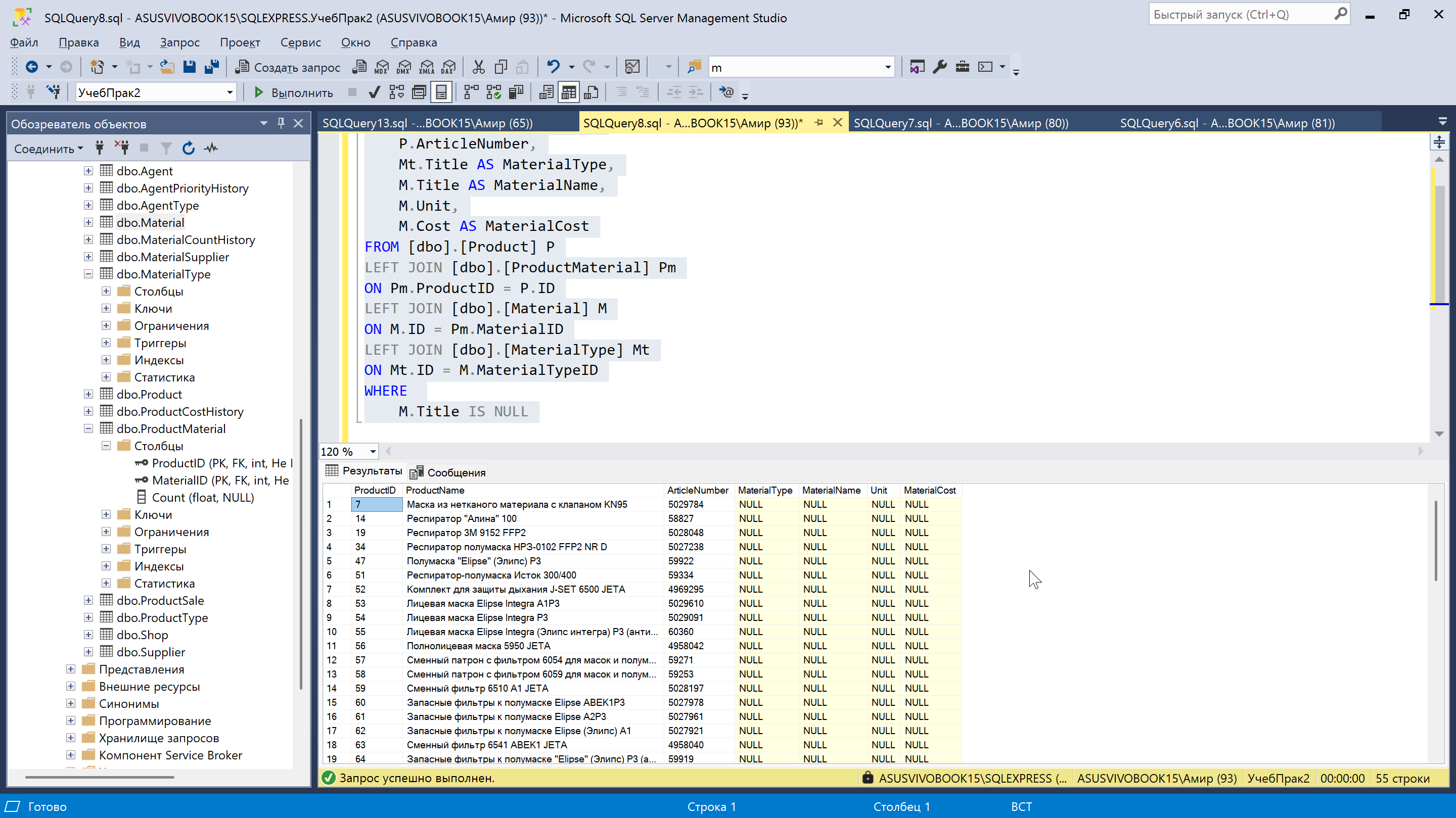


Рисунок 14 — Продукты без материалов

## **Отчет о выполнении работы 3**

Вариант 24 Логистическая компания

1. Сначала, создаем базу данных и диаграмму для него. Показано на рисунке 1.

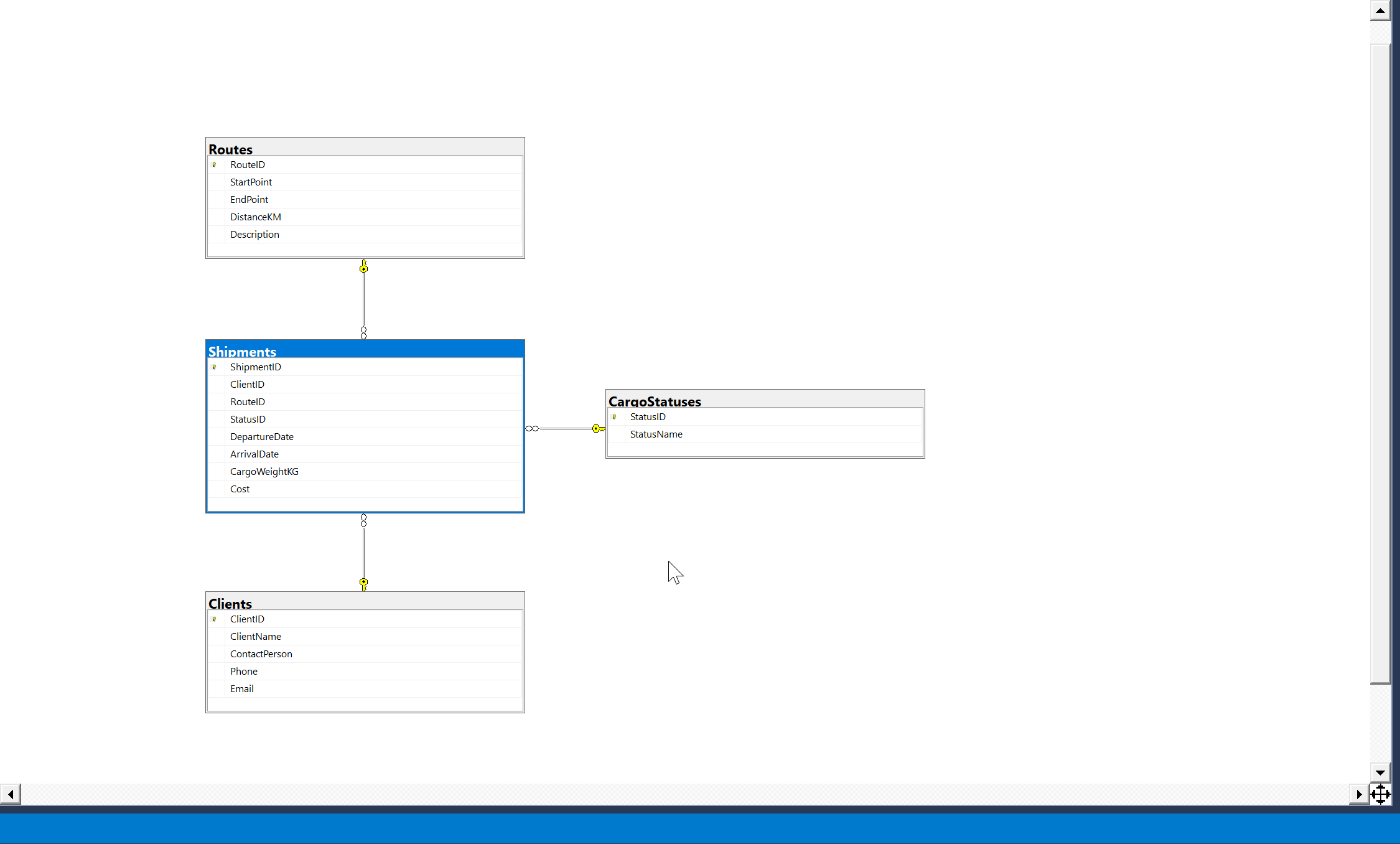


Рисунок 1 — Диаграмма бд

Листинг 1 — Создание таблиц.

|  |
| --- |
| -- Создание базы данных  CREATE DATABASE LogisticsCompany;  USE LogisticsCompany;  -- Создание таблицы клиентов  CREATE TABLE Clients (  ClientID INT PRIMARY KEY IDENTITY(1,1),  ClientName NVARCHAR(100) NOT NULL,  ContactPerson NVARCHAR(100),  Phone NVARCHAR(20),  Email NVARCHAR(100)  );  -- Создание таблицы маршрутов  CREATE TABLE Routes (  RouteID INT PRIMARY KEY IDENTITY(1,1),  StartPoint NVARCHAR(100) NOT NULL,  EndPoint NVARCHAR(100) NOT NULL,  DistanceKM INT NOT NULL,  Description NVARCHAR(200)  );  -- Создание справочника статусов груза  CREATE TABLE CargoStatuses (  StatusID INT PRIMARY KEY,  StatusName NVARCHAR(50) NOT NULL  );  -- Создание таблицы грузоперевозок  CREATE TABLE Shipments (  ShipmentID INT PRIMARY KEY IDENTITY(1,1),  ClientID INT NOT NULL,  RouteID INT NOT NULL,  StatusID INT NOT NULL,  DepartureDate DATETIME NOT NULL,  ArrivalDate DATETIME,  CargoWeightKG DECIMAL(10,2) NOT NULL,  Cost DECIMAL(10,2) NOT NULL,  FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),  FOREIGN KEY (RouteID) REFERENCES Routes(RouteID),  FOREIGN KEY (StatusID) REFERENCES CargoStatuses(StatusID)  ); |

Результат показан на рисунке 2.



Рисунок 2 — Просмотр бд.

Листинг 2 — Найти маршруты с наибольшей загруженностью.

|  |
| --- |
| -- Запрос 1: Найти маршруты с наибольшей загруженностью  SELECT  r.RouteID,  r.StartPoint + ' - ' + r.EndPoint AS Route,  COUNT(s.ShipmentID) AS ShipmentsCount,  SUM(s.CargoWeightKG) AS TotalWeightKG  FROM  Routes r  JOIN Shipments s  ON r.RouteID = s.RouteID  GROUP BY  r.RouteID, r.StartPoint, r.EndPoint  ORDER BY  ShipmentsCount DESC, TotalWeightKG DESC; |

Результат показан на рисунке 3.

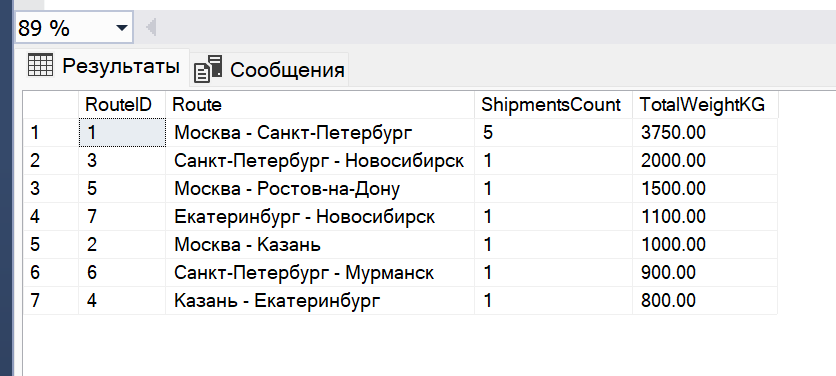


Рисунок 3 — Маршруты с наибольшей загруженностью

Листинг 3 — Определить клиентов с наибольшим числом отправлений.

|  |
| --- |
| -- Запрос 2: Определить клиентов с наибольшим числом отправлений  SELECT  c.ClientID,  c.ClientName,  COUNT(s.ShipmentID) AS ShipmentsCount  FROM  Clients c  JOIN Shipments s  ON c.ClientID = s.ClientID  GROUP BY  c.ClientID, c.ClientName  ORDER BY  ShipmentsCount DESC; |

Результат показан на рисунке 4.

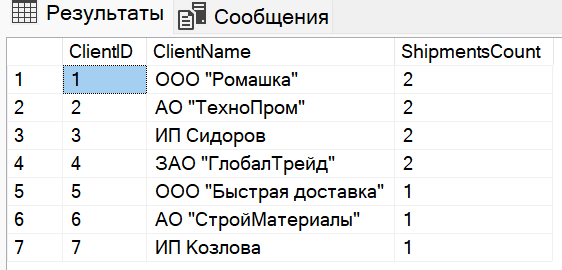


Рисунок 4 ­— Клиенты с наибольшим числом отправлений

Листинг 4 — Вывести среднюю стоимость перевозки.

|  |
| --- |
| -- Запрос 3: Вывести среднюю стоимость перевозки  SELECT  AVG(Cost) AS AverageCost,  AVG(Cost / NULLIF(CargoWeightKG, 0)) AS AverageCostPerKG  FROM  Shipments; |

Результат показан на рисунке 5.

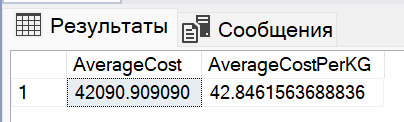


Рисунок 5 ­— Средняя стоимость перевозки

Листинг 5 — Триггер: Автоматически меняет статус груза на "В пути".

|  |
| --- |
| -- Задание 4: Создание триггера для автоматического изменения статуса груза на "В пути"  CREATE TRIGGER trg\_ShipmentDeparture  ON Shipments  AFTER UPDATE  AS  BEGIN  SET NOCOUNT ON;    -- Если изменилась дата отправления и статус был "Ожидает отправки"  UPDATE s  SET s.StatusID = 2 -- "В пути"  FROM Shipments s  INNER JOIN inserted i ON s.ShipmentID = i.ShipmentID  INNER JOIN deleted d ON s.ShipmentID = d.ShipmentID  WHERE i.DepartureDate IS NOT NULL  AND (d.DepartureDate IS NULL OR d.DepartureDate <> i.DepartureDate)  AND i.StatusID = 1;  END; |

Результат показан на рисунке 6.

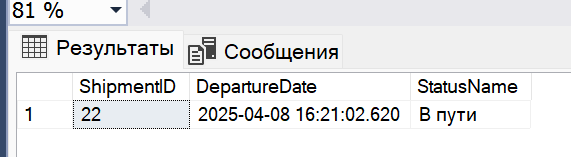


Рисунок 6 — Проверка триггера.

Запись имела в StatusName «Ожидает отправки», после изменения даты отправления срабатывает триггер.

## **Отчет о выполнении работы 4**

Вариант 24 Океанариум

1. Сначала, создаем базу данных. Показано на рисунке 1.

Листинг 1 ­– Создание бд и заполнение его данными

|  |
| --- |
| ---- Создание базы данных  --CREATE DATABASE Oceanarium;  --GO  --USE Oceanarium;  --GO  -- 1. Таблица "Сотрудники"  CREATE TABLE Employees (  EmployeeID INT PRIMARY KEY IDENTITY(1,1),  FirstName NVARCHAR(50) NOT NULL,  LastName NVARCHAR(50) NOT NULL,  Position NVARCHAR(50) NOT NULL,  Salary DECIMAL(10,2) CHECK (Salary > 0),  HireDate DATE DEFAULT GETDATE(),  Email NVARCHAR(100) UNIQUE,  Phone NVARCHAR(20) UNIQUE,  IsActive BIT DEFAULT 1  );  -- 2. Таблица "Водоемы"  CREATE TABLE Aquariums (  AquariumID INT PRIMARY KEY IDENTITY(1,1),  Name NVARCHAR(100) NOT NULL,  Volume DECIMAL(10,2) CHECK (Volume > 0),  WaterType NVARCHAR(20) CHECK (WaterType IN ('Пресная', 'Морская', 'Солоноватая')),  Temperature DECIMAL(5,2) DEFAULT 25.00,  LastCleaningDate DATE  );  -- 3. Таблица "Морские обитатели"  CREATE TABLE MarineLife (  MarineLifeID INT PRIMARY KEY IDENTITY(1,1),  Species NVARCHAR(100) NOT NULL,  CommonName NVARCHAR(100),  DateOfArrival DATE DEFAULT GETDATE(),  HealthStatus NVARCHAR(20) DEFAULT 'Здоров' CHECK (HealthStatus IN ('Здоров', 'Болен', 'На лечении', 'Умер')),  DietType NVARCHAR(50),  IsEndangered BIT DEFAULT 0,  AquariumID INT FOREIGN KEY REFERENCES Aquariums(AquariumID)  );  -- 4. Таблица "Расписание кормления"  CREATE TABLE FeedingSchedule (  ScheduleID INT PRIMARY KEY IDENTITY(1,1),  FeedingTime TIME NOT NULL,  FoodType NVARCHAR(100) NOT NULL,  Quantity DECIMAL(8,2) CHECK (Quantity > 0),  DaysOfWeek NVARCHAR(20) DEFAULT 'Ежедневно',  MarineLifeID INT FOREIGN KEY REFERENCES MarineLife(MarineLifeID),  EmployeeID INT FOREIGN KEY REFERENCES Employees(EmployeeID)  );  -- 5. Таблица "Посетители"  CREATE TABLE Visitors (  VisitorID INT PRIMARY KEY IDENTITY(1,1),  FirstName NVARCHAR(50) NOT NULL,  LastName NVARCHAR(50) NOT NULL,  VisitDate DATE DEFAULT GETDATE(),  TicketType NVARCHAR(30) DEFAULT 'Стандартный' CHECK (TicketType IN ('Стандартный', 'Детский', 'Пенсионный', 'VIP')),  GroupSize INT DEFAULT 1 CHECK (GroupSize > 0),  GuideNeeded BIT DEFAULT 0,  EmployeeID INT FOREIGN KEY REFERENCES Employees(EmployeeID)  );  -- Вставка данных в таблицу Employees  INSERT INTO Employees (FirstName, LastName, Position, Salary, Email, Phone)  VALUES  ('Иван', 'Петров', 'Биолог', 65000.00, 'ivan.petrov@oceanarium.ru', '+79161234567'),  ('Мария', 'Сидорова', 'Аквариумист', 58000.00, 'maria.sidorova@oceanarium.ru', '+79162345678'),  ('Алексей', 'Козлов', 'Гид', 52000.00, 'alex.kozlov@oceanarium.ru', '+79163456789'),  ('Елена', 'Николаева', 'Ветеринар', 75000.00, 'elena.nikolaeva@oceanarium.ru', '+79164567890'),  ('Дмитрий', 'Волков', 'Техник', 48000.00, 'dmitry.volkov@oceanarium.ru', '+79165678901'),  ('Ольга', 'Зайцева', 'Администратор', 60000.00, 'olga.zaytseva@oceanarium.ru', '+79166789012'),  ('Сергей', 'Белов', 'Охранник', 45000.00, 'sergey.belov@oceanarium.ru', '+79167890123'),  ('Анна', 'Морозова', 'Кассир', 50000.00, 'anna.morozova@oceanarium.ru', '+79168901234'),  ('Павел', 'Семенов', 'Уборщик', 40000.00, 'pavel.semenov@oceanarium.ru', '+79169012345'),  ('Наталья', 'Васнецова', 'Менеджер', 70000.00, 'natalya.vasnetsova@oceanarium.ru', '+79160123456');  -- Вставка данных в таблицу Aquariums  INSERT INTO Aquariums (Name, Volume, WaterType, Temperature, LastCleaningDate)  VALUES  ('Главный океанский резервуар', 5000.00, 'Морская', 24.50, '2023-05-15'),  ('Тропический риф', 1200.00, 'Морская', 26.00, '2023-05-20'),  ('Пресноводная река', 800.00, 'Пресная', 22.00, '2023-05-18'),  ('Акулий бассейн', 3000.00, 'Морская', 23.50, '2023-05-10'),  ('Мангровый залив', 750.00, 'Солоноватая', 25.50, '2023-05-22'),  ('Арктическая зона', 1500.00, 'Морская', 10.00, '2023-05-05'),  ('Коралловая лагуна', 2000.00, 'Морская', 25.00, '2023-05-25'),  ('Глубоководный каньон', 3500.00, 'Морская', 18.00, '2023-05-12'),  ('Детский контактный бассейн', 300.00, 'Морская', 26.00, '2023-05-28'),  ('Экзотические медузы', 600.00, 'Морская', 24.00, '2023-05-08');  -- Вставка данных в таблицу MarineLife  INSERT INTO MarineLife (Species, CommonName, AquariumID, HealthStatus, DietType, IsEndangered)  VALUES  ('Carcharodon carcharias', 'Белая акула', 4, 'Здоров', 'Рыба, кальмары', 1),  ('Tursiops truncatus', 'Афалина', 1, 'Здоров', 'Рыба, кальмары', 0),  ('Hippocampus hippocampus', 'Морской конек', 2, 'Здоров', 'Мелкие ракообразные', 1),  ('Manta birostris', 'Манта', 1, 'На лечении', 'Планктон', 1),  ('Chelonia mydas', 'Зеленая черепаха', 5, 'Здоров', 'Водоросли, медузы', 1),  ('Aptenodytes forsteri', 'Императорский пингвин', 6, 'Здоров', 'Рыба, кальмары', 0),  ('Physeter macrocephalus', 'Кашалот', 1, 'Здоров', 'Кальмары, рыба', 0),  ('Aurelia aurita', 'Лунная медуза', 10, 'Здоров', 'Планктон', 0),  ('Pygoscelis adeliae', 'Пингвин Адели', 6, 'Болен', 'Рыба, криль', 0),  ('Dasyatis pastinaca', 'Морской кот', 5, 'Здоров', 'Моллюски, ракообразные', 0),  ('Paracanthurus hepatus', 'Хирург синий', 2, 'Здоров', 'Водоросли, планктон', 0),  ('Eretmochelys imbricata', 'Бисса', 5, 'Здоров', 'Губки, медузы', 1),  ('Orcinus orca', 'Косатка', 1, 'Здоров', 'Рыба, тюлени', 0),  ('Phoca vitulina', 'Обыкновенный тюлень', 6, 'Здоров', 'Рыба, моллюски', 0),  ('Octopus vulgaris', 'Обыкновенный осьминог', 3, 'Здоров', 'Крабы, моллюски', 0);  -- Вставка данных в таблицу FeedingSchedule  INSERT INTO FeedingSchedule (MarineLifeID, EmployeeID, FeedingTime, FoodType, Quantity, DaysOfWeek)  VALUES  (1, 2, '10:00', 'Свежая рыба', 5.00, 'Пн,Ср,Пт'),  (2, 1, '09:30', 'Сельдь', 8.00, 'Ежедневно'),  (3, 2, '11:00', 'Артемия', 0.50, 'Ежедневно'),  (4, 4, '12:00', 'Криль', 3.00, 'Вт,Чт,Сб'),  (5, 2, '14:00', 'Морские водоросли', 2.00, 'Ежедневно'),  (6, 3, '15:00', 'Скумбрия', 1.50, 'Ежедневно'),  (7, 1, '16:00', 'Кальмары', 10.00, 'Пн,Ср,Пт,Вс'),  (8, 2, '13:30', 'Планктон', 0.30, 'Ежедневно'),  (9, 4, '15:30', 'Криль', 0.80, 'Ежедневно'),  (10, 2, '11:30', 'Мидии', 1.20, 'Вт,Чт,Сб'),  (11, 2, '12:30', 'Спирулина', 0.40, 'Ежедневно'),  (12, 4, '14:30', 'Медузы', 1.50, 'Пн,Ср,Пт'),  (13, 1, '17:00', 'Лосось', 15.00, 'Вт,Чт,Сб,Вс'),  (14, 3, '16:30', 'Сельдь', 3.00, 'Ежедневно'),  (15, 2, '18:00', 'Крабы', 1.00, 'Пн,Ср,Пт,Сб');  -- Вставка данных в таблицу Visitors  INSERT INTO Visitors (FirstName, LastName, VisitDate, TicketType, GroupSize, GuideNeeded, EmployeeID)  VALUES  ('Андрей', 'Иванов', '2023-05-01', 'Стандартный', 1, 0, 3),  ('Екатерина', 'Смирнова', '2023-05-02', 'VIP', 2, 1, 3),  ('Михаил', 'Попов', '2023-05-03', 'Стандартный', 4, 1, 6),  ('Алина', 'Васильева', '2023-05-04', 'Детский', 1, 0, NULL),  ('Виктор', 'Федоров', '2023-05-05', 'Пенсионный', 1, 0, NULL),  ('Дарья', 'Новикова', '2023-05-06', 'Стандартный', 2, 1, 3),  ('Артем', 'Морозов', '2023-05-07', 'VIP', 5, 1, 6),  ('София', 'Волкова', '2023-05-08', 'Детский', 1, 0, NULL),  ('Кирилл', 'Алексеев', '2023-05-09', 'Стандартный', 3, 0, NULL),  ('Полина', 'Лебедева', '2023-05-10', 'Стандартный', 2, 1, 3),  ('Глеб', 'Семенов', '2023-05-11', 'Детский', 1, 0, NULL),  ('Вероника', 'Егорова', '2023-05-12', 'Пенсионный', 2, 0, NULL),  ('Роман', 'Павлов', '2023-05-13', 'VIP', 4, 1, 6),  ('Юлия', 'Козлова', '2023-05-14', 'Стандартный', 1, 0, NULL),  ('Никита', 'Степанов', '2023-05-15', 'Стандартный', 2, 0, NULL); |

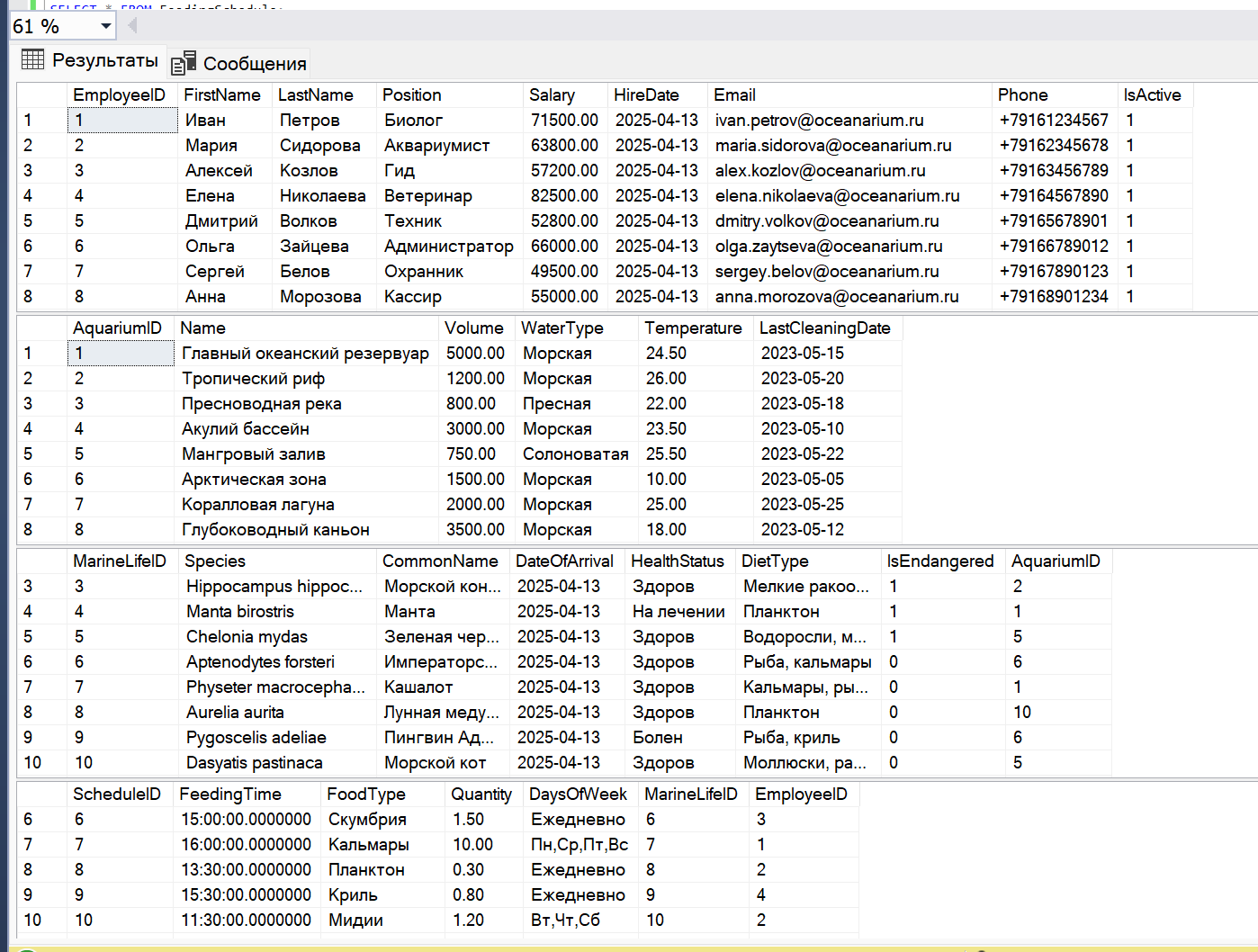


Рисунок 1 —база данных

Если смотреть скрипт, то заметите, что там используются ограничения (Check, Default, Unique и Foreign Key).

1. Employees - информация о сотрудниках океанариума
   * Ограничения: CHECK на зарплату, DEFAULT для даты приема и статуса, UNIQUE для email и телефона
2. Aquariums - данные о водоемах/аквариумах
   * Ограничения: CHECK на объем и тип воды, DEFAULT для температуры
3. MarineLife - информация о морских обитателях
   * Ограничения: FOREIGN KEY на аквариум, CHECK на статус здоровья, DEFAULT для даты прибытия и статуса исчезающего вида
4. FeedingSchedule - расписание кормления животных
   * Ограничения: FOREIGN KEY на животное и сотрудника, CHECK на количество корма, DEFAULT для дней недели
5. Visitors - данные о посетителях
   * Ограничения: FOREIGN KEY на сотрудника-гида, CHECK на тип билета и размер группы, DEFAULT для даты посещения и типа билета
6. Затем попробовал изменить данные, а некоторые вообще удалить. Результат после изменений отследить было не очень удобно, поэтому приложу скриншот выполнения запроса. Результат показан на рисунке 2.

Листинг 2 — Скрипт для изменения и удаления данных.

|  |
| --- |
| --Запросы на обновление (UPDATE)  --1. Повышение зарплаты сотрудникам на 10% (кроме уборщиков)  UPDATE Employees  SET Salary = Salary + (Salary \* 0.1);  --2. Изменение статуса здоровья всех животных в "Акульем бассейне" на "Здоров"  UPDATE MarineLife  SET HealthStatus = 'Здоров'  WHERE AquariumID = (SELECT AquariumID FROM Aquariums WHERE Name = 'Акулий бассейн');  --3. Перенос всех кормлений с понедельника на вторник (изменение дней недели)  UPDATE FeedingSchedule  SET DaysOfWeek = REPLACE(DaysOfWeek, 'Пн', 'Вт')  WHERE DaysOfWeek LIKE '%Пн%';  --Запросы на удаление (DELETE)  --1. Удаление всех посетителей, которые пришли до 2023-05-10  DELETE FROM Visitors  WHERE VisitDate < '2023-05-10';  --2. Удаление кормлений с количеством еды меньше 0.5 кг  DELETE FROM FeedingSchedule  WHERE Quantity < 0.5;  --3. Удаление всех аквариумов с объемом меньше 1000 литров (если нет связанных животных)  DELETE FROM Aquariums  WHERE Volume < 1000  AND AquariumID NOT IN (SELECT DISTINCT AquariumID FROM MarineLife); |

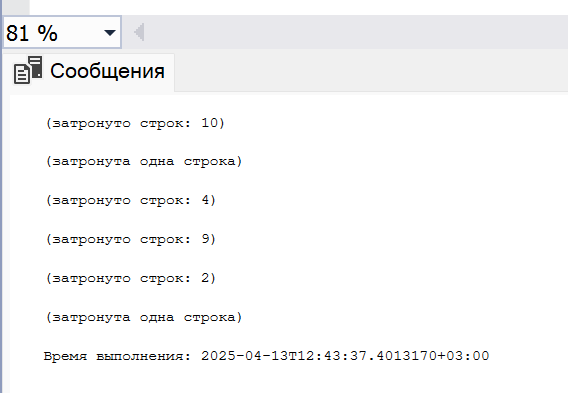


Рисунок 2 — Изменение и удаление.

Листинг 3 — Запросы на выборку данных.

|  |
| --- |
| -- Запросы на выборку (SELECT)  --1.1. Список всех морских обитателей с их местом обитания  SELECT  m.MarineLifeID,  m.Species AS 'Вид',  m.CommonName AS 'Название',  a.Name AS 'Аквариум',  a.WaterType AS 'Тип воды',  m.HealthStatus AS 'Состояние здоровья'  FROM  MarineLife m  JOIN  Aquariums a ON m.AquariumID = a.AquariumID;  -- 1.2. Список сотрудников и их обязанности (кормление животных)  SELECT  e.EmployeeID,  e.FirstName + ' ' + e.LastName AS 'Сотрудник',  e.Position AS 'Должность',  COUNT(f.ScheduleID) AS 'Количество кормлений'  FROM  Employees e  LEFT JOIN  FeedingSchedule f ON e.EmployeeID = f.EmployeeID  GROUP BY  e.EmployeeID, e.FirstName, e.LastName, e.Position;  --1.3. Посетители с экскурсоводами за последнюю неделю  SELECT  v.VisitorID,  v.FirstName + ' ' + v.LastName AS 'Посетитель',  v.VisitDate AS 'Дата посещения',  v.TicketType AS 'Тип билета',  e.FirstName + ' ' + e.LastName AS 'Экскурсовод'  FROM  Visitors v  LEFT JOIN  Employees e ON v.EmployeeID = e.EmployeeID |

Результат показан на рисунке 3.

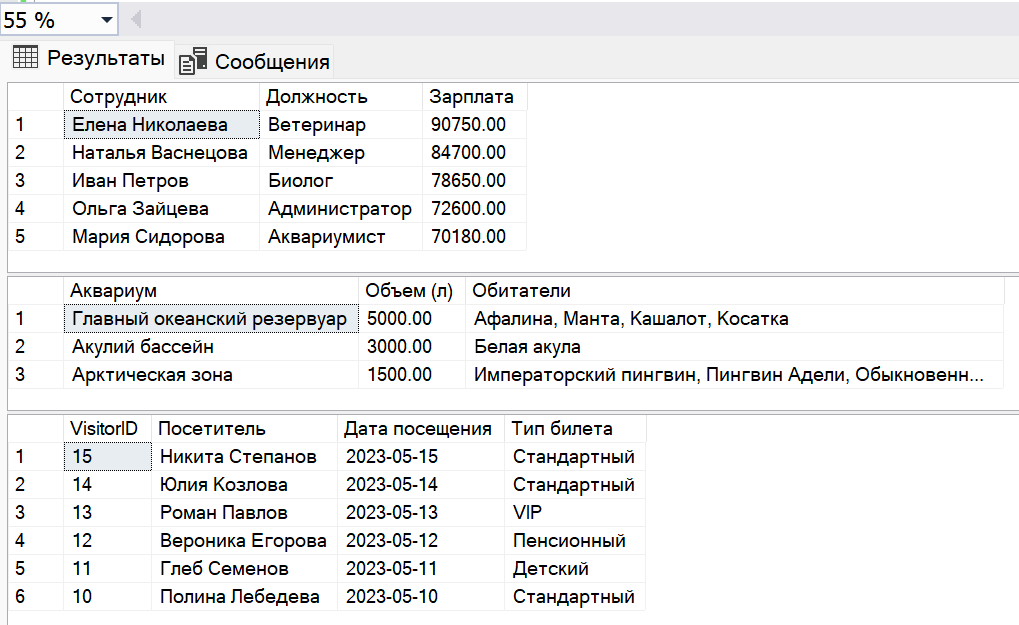


Рисунок 3 — Выборка данных.

Листинг 4 — Запросы с объединением таблиц (JOIN)

|  |
| --- |
| --2. Запросы с объединением таблиц (JOIN)  --2.1. Полное расписание кормления с деталями  SELECT  f.ScheduleID,  m.CommonName AS 'Животное',  a.Name AS 'Аквариум',  f.FeedingTime AS 'Время кормления',  f.FoodType AS 'Тип корма',  f.Quantity AS 'Количество (кг)',  f.DaysOfWeek AS 'Дни недели',  e.FirstName + ' ' + e.LastName AS 'Ответственный'  FROM  FeedingSchedule f  JOIN  MarineLife m ON f.MarineLifeID = m.MarineLifeID  JOIN  Aquariums a ON m.AquariumID = a.AquariumID  JOIN  Employees e ON f.EmployeeID = e.EmployeeID;    --2.2. Больные животные и их ответственные сотрудники  SELECT  m.MarineLifeID,  m.CommonName AS 'Животное',  m.HealthStatus AS 'Состояние',  a.Name AS 'Аквариум',  e.FirstName + ' ' + e.LastName AS 'Ответственный биолог',  e.Email AS 'Контакт'  FROM  MarineLife m  JOIN  Aquariums a ON m.AquariumID = a.AquariumID  JOIN  Employees e ON a.AquariumID = e.EmployeeID  WHERE  m.HealthStatus IN ('Болен', 'На лечении'); |

Результат показан на рисунке 4.

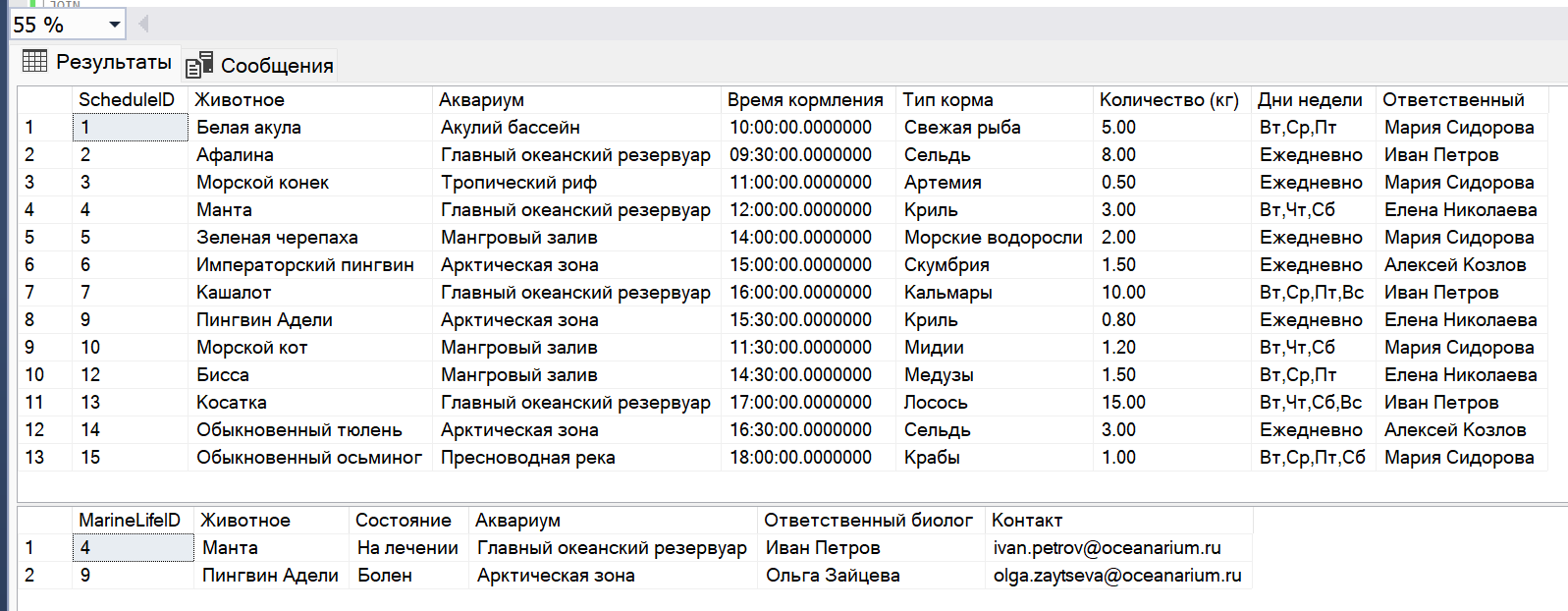


Рисунок 4 — Объединение таблиц

Листинг 5 — Запросы с группировкой (GROUP BY) и условиями (HAVING).

|  |
| --- |
| --3. Запросы с группировкой (GROUP BY) и условиями (HAVING)  --3.1. Аквариумы с количеством животных (только с 3+ животными)  SELECT  a.Name AS 'Аквариум',  a.Volume AS 'Объем (л)',  COUNT(m.MarineLifeID) AS 'Количество животных',  SUM(CASE WHEN m.IsEndangered = 1 THEN 1 ELSE 0 END) AS 'Исчезающие виды'  FROM  Aquariums a  LEFT JOIN  MarineLife m ON a.AquariumID = m.AquariumID  GROUP BY  a.Name, a.Volume  HAVING  COUNT(m.MarineLifeID) >= 3  ORDER BY  COUNT(m.MarineLifeID) DESC;  --3.2. Средняя зарплата по должностям (только где средняя > 50000)  SELECT  Position AS 'Должность',  COUNT(\*) AS 'Количество',  AVG(Salary) AS 'Средняя зарплата',  SUM(Salary) AS 'Общий фонд'  FROM  Employees  GROUP BY  Position  HAVING  AVG(Salary) > 50000  ORDER BY  AVG(Salary) DESC; |

Результат показан на рисунке 5.

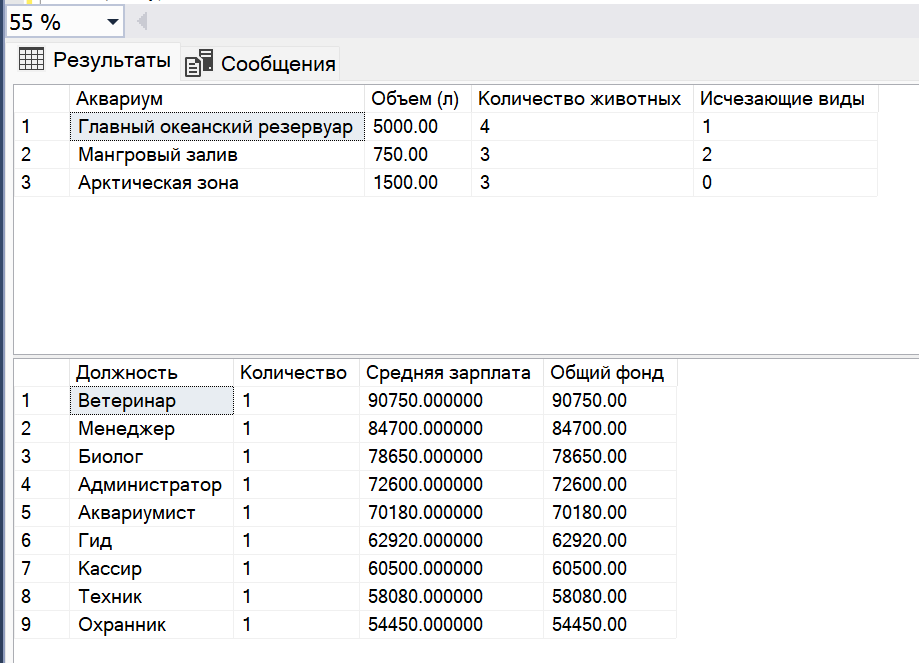


Рисунок 5 ­—Группировки и условия

Листинг 6 — Запросы с сортировкой (ORDER BY) и ограничением (LIMIT/OFFSET)

|  |
| --- |
| --4. Запросы с сортировкой (ORDER BY) и ограничением (LIMIT/OFFSET)  --4.1. ТОП-5 самых дорогих сотрудников  SELECT TOP 5  FirstName + ' ' + LastName AS 'Сотрудник',  Position AS 'Должность',  Salary AS 'Зарплата'  FROM  Employees  ORDER BY  Salary DESC;  --4.2. 3 самых больших аквариума с животными  SELECT TOP 3  a.Name AS 'Аквариум',  a.Volume AS 'Объем (л)',  STRING\_AGG(m.CommonName, ', ') AS 'Обитатели'  FROM  Aquariums a  JOIN  MarineLife m ON a.AquariumID = m.AquariumID  GROUP BY  a.Name, a.Volume  ORDER BY  a.Volume DESC;  --4.3. Последние 10 посетителей (пагинация. Грубо говоря срез строк)  SELECT  VisitorID,  FirstName + ' ' + LastName AS 'Посетитель',  VisitDate AS 'Дата посещения',  TicketType AS 'Тип билета'  FROM  Visitors  ORDER BY  VisitDate DESC  OFFSET 0 ROWS FETCH NEXT 10 ROWS ONLY; |

Результат показан на рисунке 6.

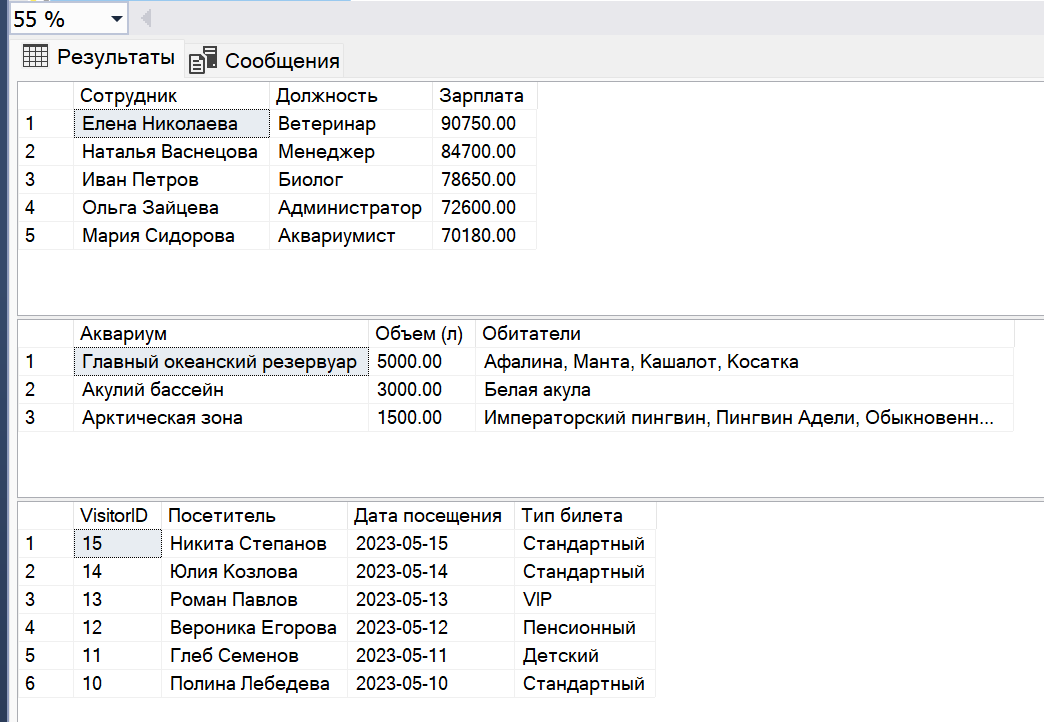


Рисунок 6 ­— Сортировки и ограничения

## **Отчет о выполнении работы 5**

Вариант 24. Учёт музыкантов и

концертов музыкального агентства

Цель: Создание системы учёта музыкантов и концертов музыкального агентства для автоматизации управления артистами, мероприятиями, расписанием и финансами.

Область применения:

* Учёт музыкантов (личные данные, контакты, репертуар).
* Планирование концертов (дата, место, участники).
* Контроль финансовых расчетов (гонорары, расходы).
* Анализ загруженности артистов.

Ключевые сущности

1. Музыкант (артист, участник группы).
2. Концерт (мероприятие, выступление).
3. Место проведения (концертный зал, клуб).
4. Группа (музыкальный коллектив).
5. Договор (финансовые условия, обязательства).

Таблицы сущностей

Музыкант

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Тип данных** | **Описание** | **Ограничения** |
| musician\_id | INT | Уникальный ID музыканта | PK, AUTO\_INCREMENT |
| full\_name | VARCHAR(50) | ФИО музыканта | NOT NULL |
| instrument | VARCHAR(30) | Инструмент |  |
| phone | VARCHAR(20) | Контактный телефон | UNIQUE |
| birth\_date | DATE | Дата рождения |  |

Группа

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Тип данных** | **Описание** | **Ограничения** |
| band\_id | INT | Уникальный ID группы | PK, AUTO\_INCREMENT |
| name | VARCHAR(50) | Название группы | NOT NULL, UNIQUE |
| genre | VARCHAR(30) | Музыкальный жанр |  |
| foundation\_year | INT | Год основания | CHECK (foundation\_year > 1900) |

Концерт

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Тип данных** | **Описание** | **Ограничения** |
| concert\_id | INT | Уникальный ID концерта | PK, AUTO\_INCREMENT |
| date\_time | DATETIME | Дата и время проведения | NOT NULL |
| venue\_id | INT | ID места проведения | FK – Место проведения |
| status | ENUM('planned', 'completed') | Статус | DEFAULT 'planned' |
| ticket\_price | DECIMAL(10,2) | Цена билета | CHECK (ticket\_price >= 0) |

Место проведения

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Тип данных** | **Описание** | **Ограничения** |
| venue\_id | INT | Уникальный ID места | PK, AUTO\_INCREMENT |
| name | VARCHAR(50) | Название зала/клуба | NOT NULL |
| address | VARCHAR(100) | Адрес |  |
| capacity | INT | Вместимость (человек) | CHECK (capacity > 0) |

Договор

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Тип данных** | **Описание** | **Ограничения** |
| contract\_id | INT | Уникальный ID договора | PK, AUTO\_INCREMENT |
| band\_id | INT | ID группы (если с группой) | FK – Группа, NULL |
| musician\_id | INT | ID музыканта (если сольный) | FK – Музыкант, NULL |
| fee | DECIMAL(10,2) | Гонорар | NOT NULL |
| sign\_date | DATE | Дата подписания | DEFAULT CURRENT\_DATE |

Связи между сущностями

1. Музыкант — Группа:
   * Один-ко-многим (один музыкант может быть в нескольких группах, одна группа включает многих музыкантов).
2. Группа — Концерт:
   * Многие-ко-многим (одна группа может выступать на многих концертах, один концерт может включать несколько групп).
3. Концерт — Место проведения:
   * Многие-к-одному (один концерт проходит в одном месте, одно место принимает много концертов).
4. Договор — Группа/Музыкант:
   * Один-к-одному (один договор оформляется на одну группу или сольного музыканта).

ER-диаграмма показана на рисунке 1.



Рисунок 1 – ER-диагрмма

## **Отчет о выполнении работы 6**

Платформа для аренды инструментов и оборудования

1. Создайте структуру БД с 3–4 связанными таблицами.

Листинг 1 ­– Создание бд и Диаграмма

|  |
| --- |
| -- 1. Создаем базу данных  CREATE DATABASE ToolRental;  GO  USE ToolRental;  GO  -- 2. Таблица пользователей (арендодатели и арендаторы)  CREATE TABLE Users (  Id INT PRIMARY KEY IDENTITY(1,1),  Name NVARCHAR(100) NOT NULL,  Phone NVARCHAR(20) NOT NULL  );  -- 3. Таблица инструментов  CREATE TABLE Tools (  Id INT PRIMARY KEY IDENTITY(1,1),  OwnerId INT NOT NULL, -- кто сдает  Name NVARCHAR(100) NOT NULL,  PricePerDay DECIMAL(10,2) NOT NULL,  IsAvailable BIT DEFAULT 1,  FOREIGN KEY (OwnerId) REFERENCES Users(Id)  );  -- 4. Таблица аренды  CREATE TABLE Rentals (  Id INT PRIMARY KEY IDENTITY(1,1),  ToolId INT NOT NULL,  UserId INT NOT NULL, -- кто берет  StartDate DATE NOT NULL,  EndDate DATE NOT NULL,  TotalCost DECIMAL(10,2) NOT NULL,  FOREIGN KEY (ToolId) REFERENCES Tools(Id),  FOREIGN KEY (UserId) REFERENCES Users(Id)  ); |

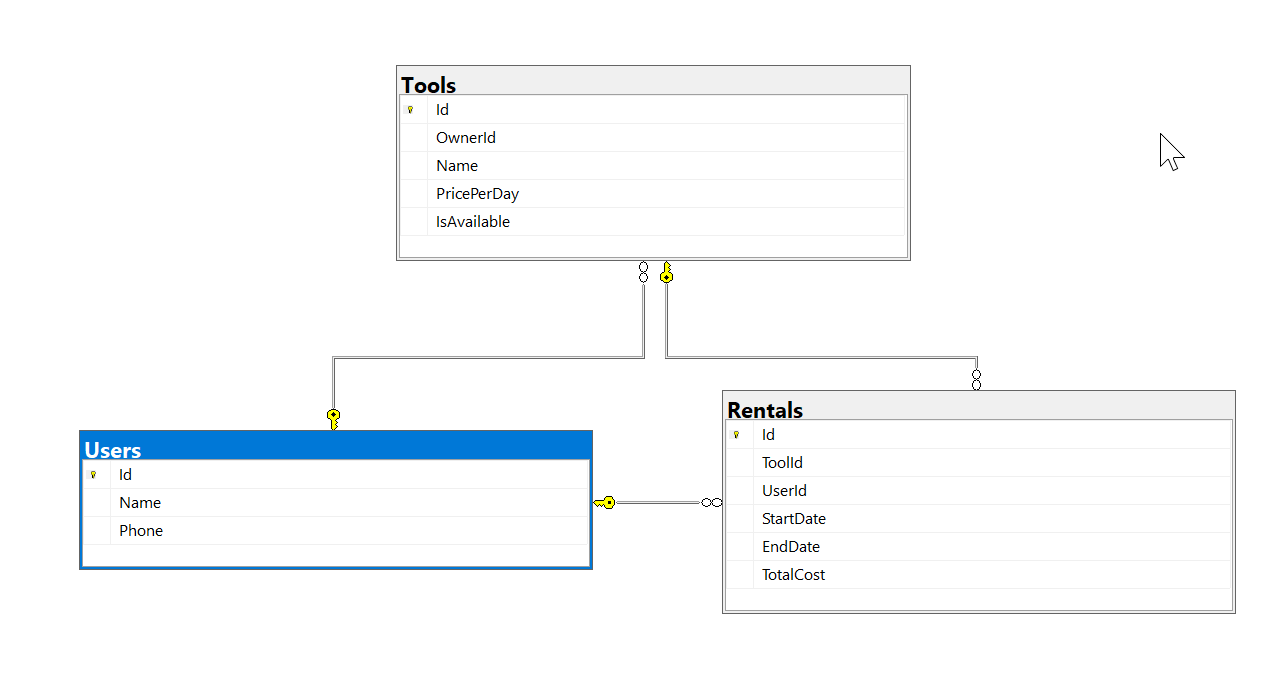


Рисунок 1 —база данных

1. Наполните таблицы тестовыми данными (минимум по 10 строк).

Листинг 2 – Заполнение базы данных

|  |
| --- |
| -- 1. Добавляем 10 пользователей  INSERT INTO Users (Name, Phone) VALUES  ('Иван Петров', '79161234567'),  ('Мария Сидорова', '79167654321'),  ('Алексей Иванов', '79169998877'),  ('Елена Кузнецова', '79162223344'),  ('Дмитрий Смирнов', '79165556677'),  ('Ольга Васильева', '79168889900'),  ('Сергей Попов', '79161112233'),  ('Анна Морозова', '79164445566'),  ('Павел Новиков', '79167778899'),  ('Наталья Федорова', '79160001122');  GO  -- 2. Добавляем 10 инструментов  INSERT INTO Tools (OwnerId, Name, PricePerDay, IsAvailable) VALUES  (1, 'Перфоратор Bosch GBH 2-26', 800.00, 1),  (2, 'Шуруповерт Makita DF347DWE', 500.00, 1),  (3, 'Бензопила Husqvarna 135', 1200.00, 1),  (4, 'Газонокосилка электрическая', 900.00, 1),  (5, 'Бетономешалка 160л', 1500.00, 1),  (6, 'Лестница алюминиевая 6м', 300.00, 1),  (7, 'Триммер бензиновый', 700.00, 1),  (8, 'Строительный пылесос', 400.00, 1),  (9, 'Шлифмашина DeWalt', 600.00, 1),  (10, 'Генератор 3 кВт', 2000.00, 1);  GO  -- 3. Добавляем аренды с АКТУАЛЬНЫМИ датами  INSERT INTO Rentals (ToolId, UserId, StartDate, EndDate, TotalCost) VALUES  (1, 2, GETDATE()-1, GETDATE()+1, 1600.00), -- Аренда активна (вчера-завтра)  (3, 4, GETDATE()+2, GETDATE()+5, 3600.00), -- Аренда начнется послезавтра  (2, 5, GETDATE()-3, GETDATE()-1, 1000.00), -- Аренда завершена  (5, 7, GETDATE(), GETDATE()+4, 6000.00), -- Аренда началась сегодня  (4, 3, GETDATE()-5, GETDATE()-2, 2700.00), -- Аренда завершена  (6, 8, GETDATE()+1, GETDATE()+3, 600.00), -- Аренда начнется завтра  (7, 10, GETDATE()-2, GETDATE()+2, 2800.00), -- Аренда активна  (8, 1, GETDATE()-7, GETDATE()-4, 1200.00), -- Аренда завершена  (9, 6, GETDATE()+3, GETDATE()+6, 1800.00), -- Аренда начнется через 3 дня  (10, 9, GETDATE()-1, GETDATE()+1, 4000.00); -- Аренда активна  GO  -- Обновляем статус инструментов, которые сейчас арендованы  UPDATE Tools SET IsAvailable = 0 WHERE Id IN (  SELECT ToolId FROM Rentals  WHERE GETDATE() BETWEEN StartDate AND EndDate  );  GO  -- Посмотреть всех пользователей  SELECT \* FROM Users;  -- Посмотреть все инструменты  SELECT \* FROM Tools;  -- Посмотреть все аренды  SELECT \* FROM Rentals;  -- Посмотреть доступные инструменты  SELECT \* FROM Tools WHERE IsAvailable = 1;  -- Посмотреть текущие аренды (активные на сегодня)  SELECT \* FROM Rentals  WHERE GETDATE() BETWEEN StartDate AND EndDate; |

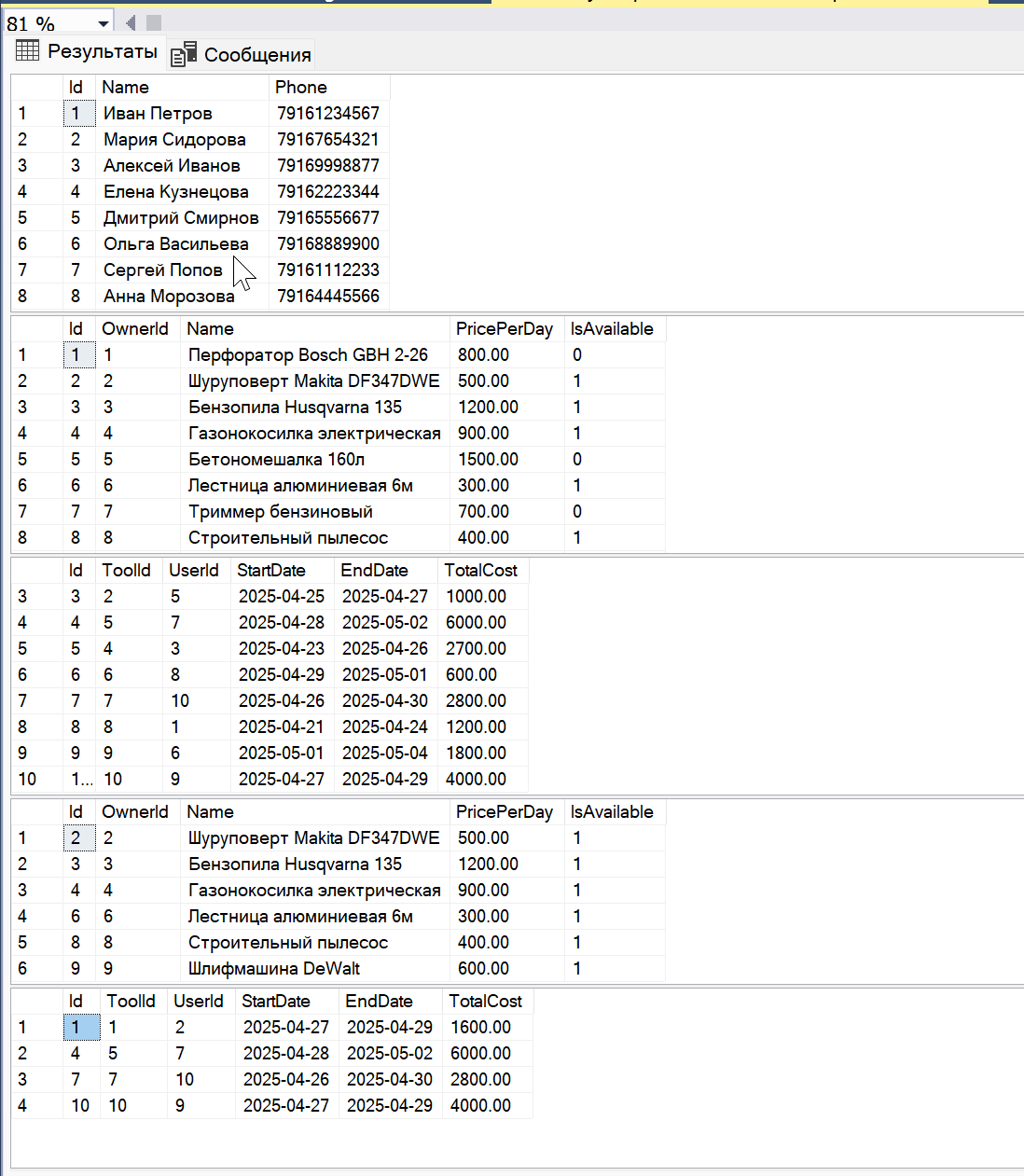


Рисунок 2 – Заполнение бд

1. Создайте следующие объекты:
   * как минимум 1 табличное выражение (CTE)
   * как минимум 1 представление (VIEW)
   * как минимум 1 пользовательскую табличную или скалярную функцию
   * как минимум 1 временную таблицу в SELECT-запросе

Листинг 3 – Табличные выражения

|  |
| --- |
| -- табличные выражения (CTE)  -- CTE для активных аренд  WITH ActiveRentals AS (  SELECT  r.Id AS RentalId,  t.Name AS ToolName,  u.Name AS RenterName,  r.StartDate,  r.EndDate,  r.TotalCost,  DATEDIFF(day, GETDATE(), r.EndDate) AS DaysRemaining  FROM Rentals r  JOIN Tools t ON r.ToolId = t.Id  JOIN Users u ON r.UserId = u.Id  WHERE GETDATE() BETWEEN r.StartDate AND r.EndDate  )  SELECT \* FROM ActiveRentals  ORDER BY DaysRemaining;  -- CTE для расчета доходов по месяцам  with MonthlyRevenue as (  SELECT  YEAR(r.StartDate) AS Year,  MONTH(r.StartDate) AS Month,  SUM(r.TotalCost) AS Revenue,  COUNT(r.Id) AS RentalCount  FROM Rentals r  GROUP BY YEAR(r.StartDate), MONTH(r.StartDate)  )  SELECT  Year,  Month,  Revenue,  RentalCount  FROM MonthlyRevenue  ORDER BY Year, Month; |

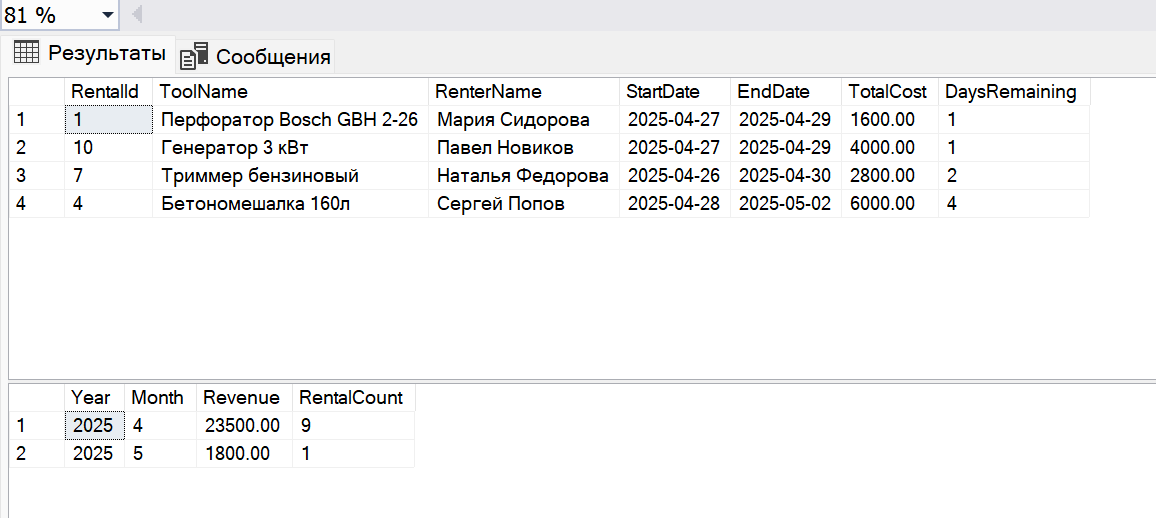


Рисунок 3 – Табличные выражения

Листинг 4 – Табличные представления

|  |
| --- |
| --Табличные представления (VIEW)  -- Представление доступных инструментов  CREATE VIEW AvailableToolsView AS  SELECT  t.Id,  t.Name,  t.PricePerDay,  u.Name AS OwnerName,  u.Phone AS OwnerPhone  FROM Tools t  JOIN Users u ON t.OwnerId = u.Id  WHERE t.IsAvailable = 1;  -- Представление статистики по пользователям  CREATE VIEW UserStatsView AS  SELECT  u.Id,  u.Name,  u.Phone,  COUNT(r.Id) AS TotalRentals,  SUM(r.TotalCost) AS TotalSpent,  AVG(r.TotalCost) AS AvgRentalCost,  MAX(r.EndDate) AS LastRentalDate  FROM Users u  LEFT JOIN Rentals r ON u.Id = r.UserId  GROUP BY u.Id, u.Name, u.Phone;  --Получить список доступных инструментов  SELECT \* FROM AvailableToolsView;  --Показать статистику пользователей  SELECT \* FROM [UserStatsView] ORDER BY [TotalSpent] DESC; |

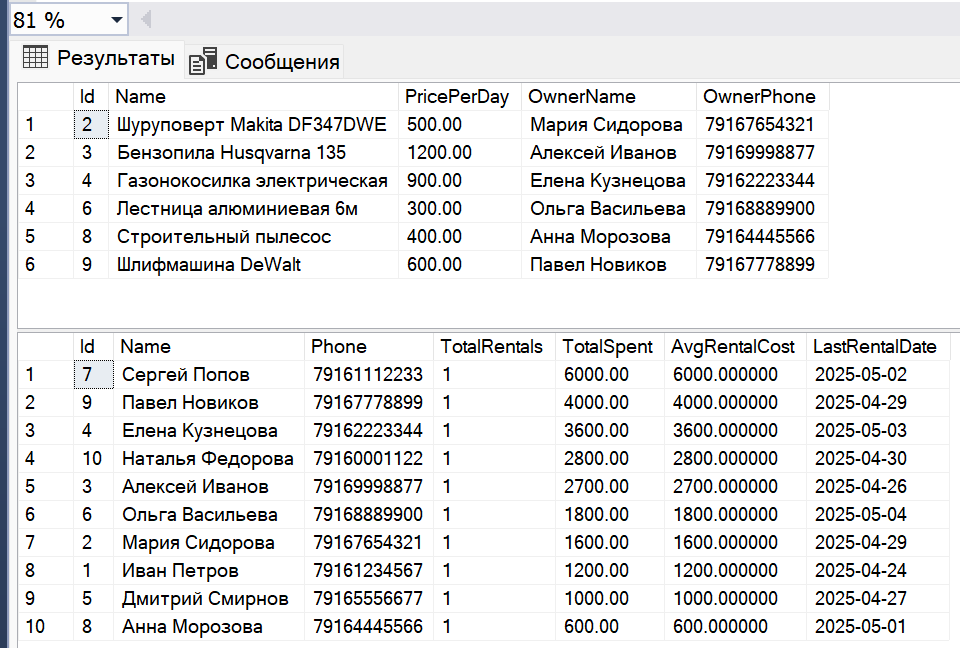


Рисунок 4 – Табличные представления

Листинг 5 – Пользовательские функции

|  |
| --- |
| -- Пользовательские табличные функции  -- Табличная функция: Поиск доступных инструментов в период  CREATE FUNCTION GetAvailableToolsInPeriod (  @StartDate DATE,  @EndDate DATE  )  RETURNS TABLE  AS  RETURN  (  SELECT  t.Id,  t.Name,  t.PricePerDay,  u.Name AS OwnerName,  u.Phone AS OwnerPhone  FROM Tools t  JOIN Users u ON t.OwnerId = u.Id  WHERE t.IsAvailable = 1  AND t.Id NOT IN (  SELECT ToolId FROM Rentals  WHERE (@StartDate BETWEEN StartDate AND EndDate)  OR (@EndDate BETWEEN StartDate AND EndDate)  OR (StartDate BETWEEN @StartDate AND @EndDate)  )  );  SELECT \* FROM [GetAvailableToolsInPeriod]('2023-10-15', '2023-10-20'); |

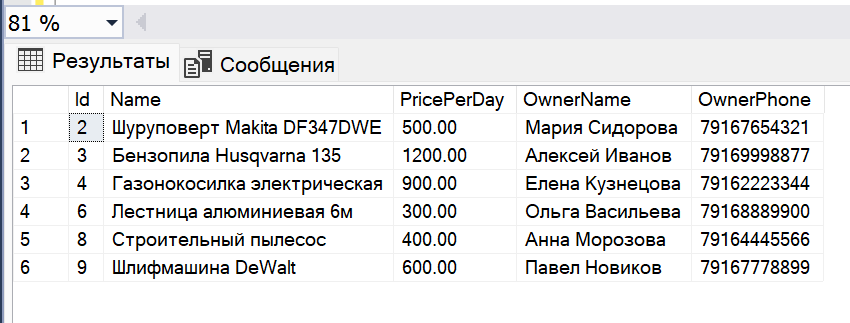


Рисунок 5 – Поиск инструментов в период

Листинг 6 – Временные таблицы

|  |
| --- |
| -- Создаем временную таблицу для обработки аренд  CREATE TABLE #PendingRentals (  RentalId INT,  ToolName VARCHAR(100),  RenterName VARCHAR(100),  StartDate DATE,  EndDate DATE,  TotalCost DECIMAL(10,2),  IsAvailable BIT  );  -- Заполняем данными (используем флаг IsAvailable из Tools вместо Status)  INSERT INTO #PendingRentals (RentalId, ToolName, RenterName, StartDate, EndDate, TotalCost, IsAvailable)  SELECT  r.Id,  t.Name,  u.Name,  r.StartDate,  r.EndDate,  r.TotalCost,  t.IsAvailable  FROM Rentals r  JOIN Tools t ON r.ToolId = t.Id  JOIN Users u ON r.UserId = u.Id  WHERE t.IsAvailable = 0; -- Берем только аренды, где инструмент сейчас занят  -- Обновляем статус доступности инструментов  UPDATE t  SET t.IsAvailable = 1  FROM Tools t  JOIN Rentals r ON t.Id = r.ToolId  JOIN #PendingRentals pr ON r.Id = pr.RentalId  WHERE r.EndDate < GETDATE(); -- Делаем инструменты доступными, если аренда завершена  -- Просмотр результатов  SELECT  RentalId,  ToolName,  RenterName,  StartDate,  EndDate,  TotalCost,  CASE  WHEN EndDate < GETDATE() THEN 'Завершена'  WHEN StartDate > GETDATE() THEN 'Запланирована'  ELSE 'Активна'  END AS RentalStatus  FROM #PendingRentals;  -- Очистка  DROP TABLE #PendingRentals; |

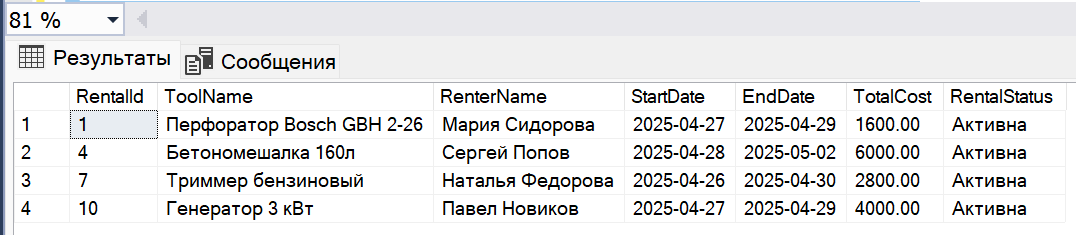


Рисунок 6 – Временные таблицы

1. Настройте создание резервной копии базы средствами SQL Server. ПКМ – Задачи – Создание резервной копии – Ок.

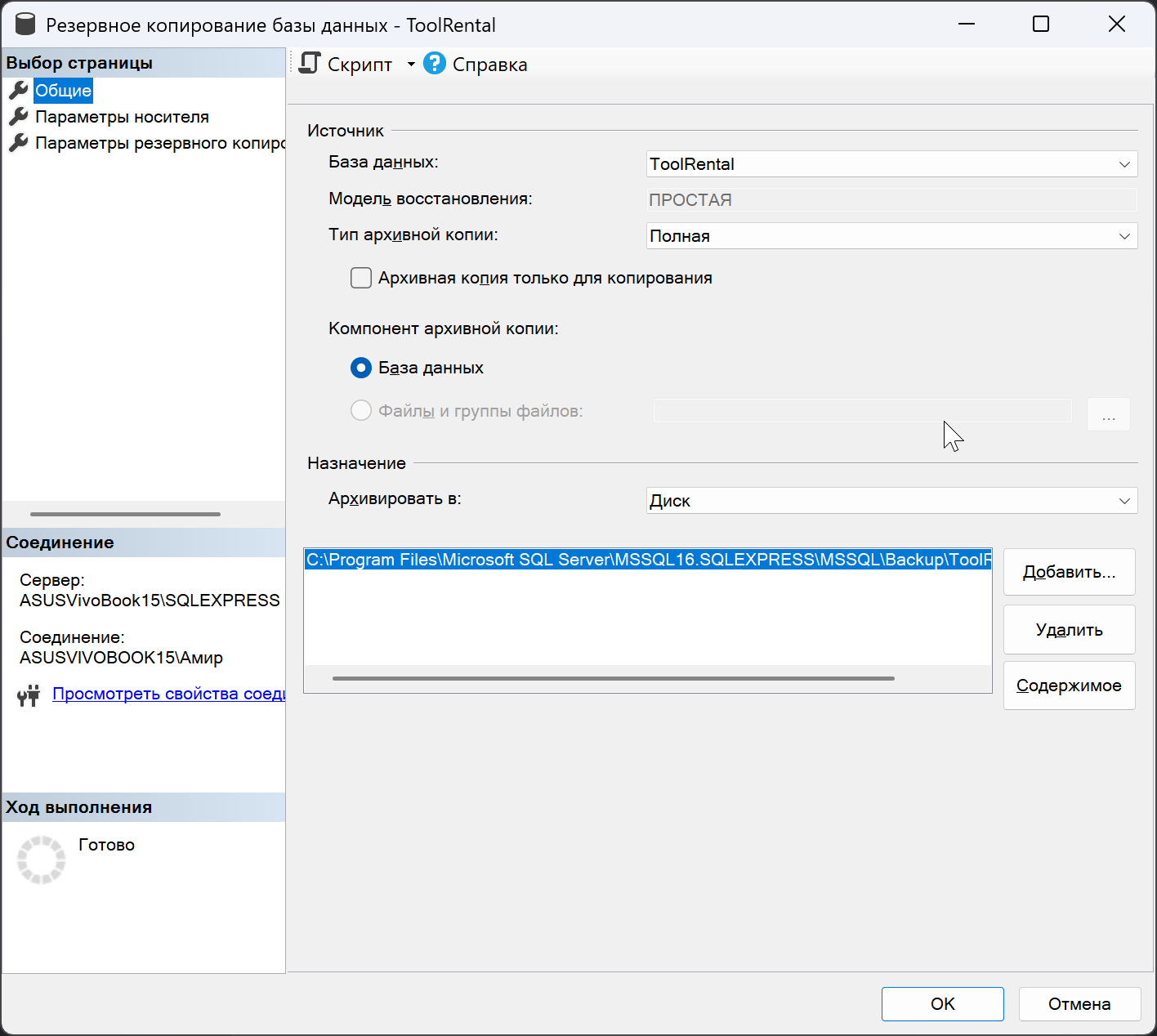


Рисунок 7 –Создание резервной копии

1. Покажите процесс восстановления БД из резервной копии. ПКМ по «базы данных» – «восстановить бд» – «Устройство» – «….» – «Добавить». Выбираем свой файл.bak и нажимаем «Ок».

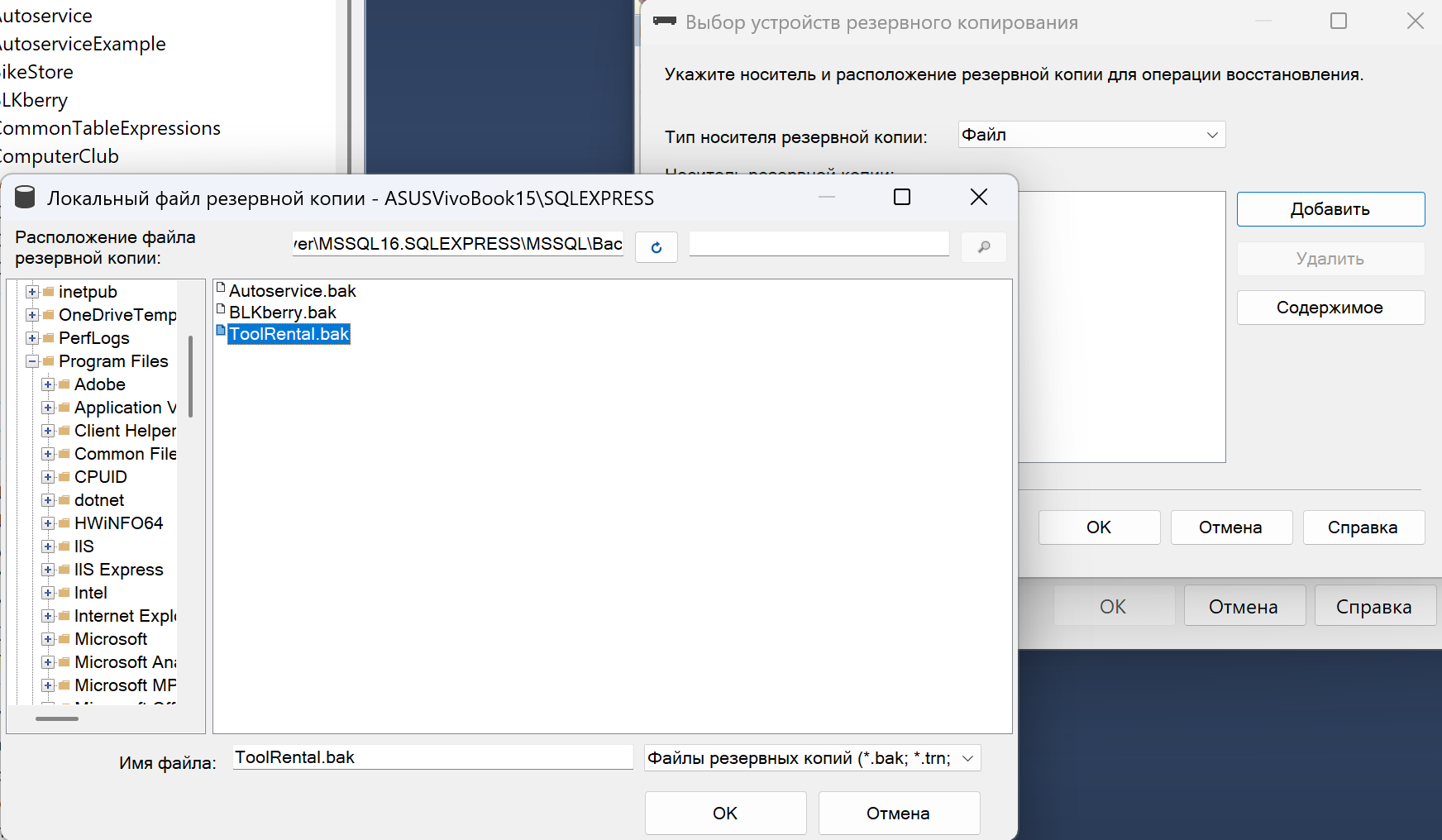


Рисунок 8 – Восстановление бд

## **Отчет о выполнении работы 7**

Сервис по доставке готовой еды с подпиской

1. Создайте структуру БД с 3–5 связанными таблицами.

Листинг 1 ­– Создание бд и Диаграмма

|  |
| --- |
| -- 1. Клиенты  CREATE TABLE Clients (  ClientID INT PRIMARY KEY IDENTITY(1,1),  Name NVARCHAR(100) NOT NULL,  Phone NVARCHAR(20) UNIQUE NOT NULL,  Address NVARCHAR(200) NOT NULL,  RegDate DATE DEFAULT GETDATE()  );  -- 2. Подписки  CREATE TABLE Subscriptions (  SubID INT PRIMARY KEY IDENTITY(1,1),  Name NVARCHAR(50) NOT NULL,  Price DECIMAL(10,2) NOT NULL,  MealsPerWeek INT NOT NULL,  DurationDays INT NOT NULL  );  -- 3. Активные подписки клиентов  CREATE TABLE ClientSubs (  ClientSubID INT PRIMARY KEY IDENTITY(1,1),  ClientID INT NOT NULL FOREIGN KEY REFERENCES Clients(ClientID),  SubID INT NOT NULL FOREIGN KEY REFERENCES Subscriptions(SubID),  StartDate DATE NOT NULL,  EndDate DATE NOT NULL,  IsActive BIT DEFAULT 1  );  -- 4. Доставки  CREATE TABLE Deliveries (  DeliveryID INT PRIMARY KEY IDENTITY(1,1),  ClientSubID INT NOT NULL FOREIGN KEY REFERENCES ClientSubs(ClientSubID),  DeliveryDate DATE NOT NULL,  MealDescription NVARCHAR(500) NOT NULL,  Status NVARCHAR(20) DEFAULT 'Ожидает' -- Ожидает, В пути, Доставлено  ); |

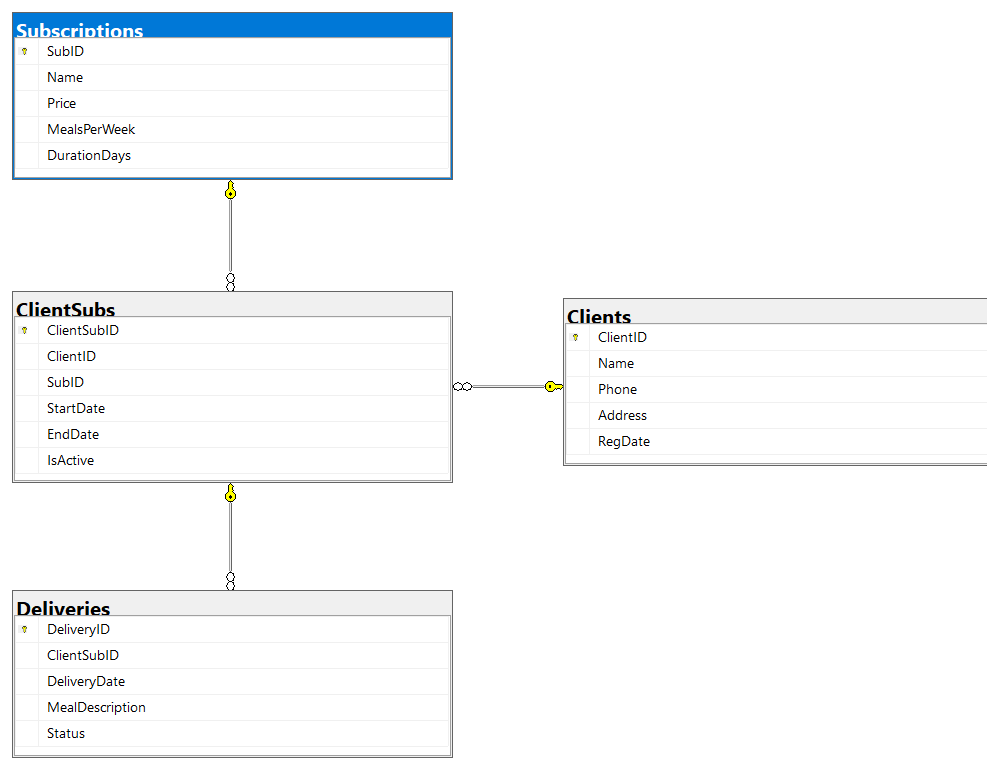


Рисунок 1 —база данных

1. Заполните таблицы тестовыми данными (минимум по 10 строк).

Листинг 2 – Заполнение базы данных

|  |
| --- |
| -- 1. Заполняем таблицу Clients (Клиенты)  INSERT INTO Clients (Name, Phone, Address, RegDate)  VALUES  ('Иванов Алексей', '79161234567', 'ул. Ленина, 10, кв. 5', '2023-01-15'),  ('Петрова Мария', '79262345678', 'пр. Мира, 25, кв. 12', '2023-02-20'),  ('Сидоров Дмитрий', '79373456789', 'ул. Гагарина, 7, кв. 33', '2023-03-05'),  ('Кузнецова Анна', '79484567890', 'ул. Садовая, 15, кв. 8', '2023-03-10'),  ('Федоров Игорь', '79595678901', 'пр. Победы, 3, кв. 17', '2023-04-22'),  ('Морозова Елена', '79606789012', 'ул. Центральная, 1, кв. 9', '2023-05-18'),  ('Николаев Павел', '79717890123', 'ул. Школьная, 14, кв. 21', '2023-06-30'),  ('Васнецова Ольга', '79828901234', 'пр. Строителей, 8, кв. 4', '2023-07-11'),  ('Белов Сергей', '79939012345', 'ул. Лесная, 12, кв. 7', '2023-08-09'),  ('Козлова Наталья', '79040123456', 'ул. Речная, 5, кв. 15', '2023-09-25');  -- 2. Заполняем таблицу Subscriptions (Подписки)  INSERT INTO Subscriptions (Name, Price, MealsPerWeek, DurationDays)  VALUES  ('Легкий перекус', 1500.00, 5, 30),  ('Стандарт', 2500.00, 10, 30),  ('Премиум', 4000.00, 14, 30),  ('Фитнес', 3500.00, 12, 30),  ('Вегетарианский', 3000.00, 10, 30),  ('Бизнес-ланч', 2000.00, 5, 30),  ('Семейный', 5000.00, 21, 30),  ('Экспресс', 1800.00, 7, 14),  ('Пробный', 1000.00, 3, 14),  ('VIP', 6000.00, 28, 30);  -- 3. Заполняем таблицу ClientSubs (Активные подписки клиентов)  INSERT INTO ClientSubs (ClientID, SubID, StartDate, EndDate, IsActive)  VALUES  (1, 2, '2023-10-01', '2023-10-31', 1),  (2, 1, '2023-10-05', '2023-11-04', 1),  (3, 3, '2023-09-15', '2023-10-15', 0),  (4, 4, '2023-10-10', '2023-11-09', 1),  (5, 2, '2023-09-20', '2023-10-20', 0),  (6, 5, '2023-10-15', '2023-11-14', 1),  (7, 7, '2023-10-03', '2023-11-02', 1),  (8, 6, '2023-09-25', '2023-10-25', 0),  (9, 8, '2023-10-12', '2023-10-26', 1),  (10, 9, '2023-10-20', '2023-11-03', 1);  -- 4. Заполняем таблицу Deliveries (Доставки)  INSERT INTO Deliveries (ClientSubID, DeliveryDate, MealDescription, Status)  VALUES  (1, '2023-10-02', 'Куриная грудка с овощами, греческий салат, хлеб', 'Доставлено'),  (1, '2023-10-05', 'Лосось на гриле с брокколи, булгур', 'Доставлено'),  (2, '2023-10-06', 'Омлет с шампиньонами, тосты, фрукты', 'Доставлено'),  (3, '2023-09-16', 'Стейк из говядины с картофельным пюре, салат Цезарь', 'Доставлено'),  (4, '2023-10-11', 'Кускус с овощами, хумус, лаваш', 'В пути'),  (5, '2023-09-21', 'Тыквенный суп-пюре, сэндвич с индейкой', 'Доставлено'),  (6, '2023-10-16', 'Фалафель с тахини, овощной салат, пита', 'Ожидает'),  (7, '2023-10-04', '3 бизнес-ланча: суп, второе, салат, компот', 'Доставлено'),  (8, '2023-09-26', 'Салат с креветками, куриный бульон, хлебцы', 'Доставлено'),  (9, '2023-10-13', 'Смузи-боул с гранолой и ягодами, тост с авокадо', 'Ожидает'); |

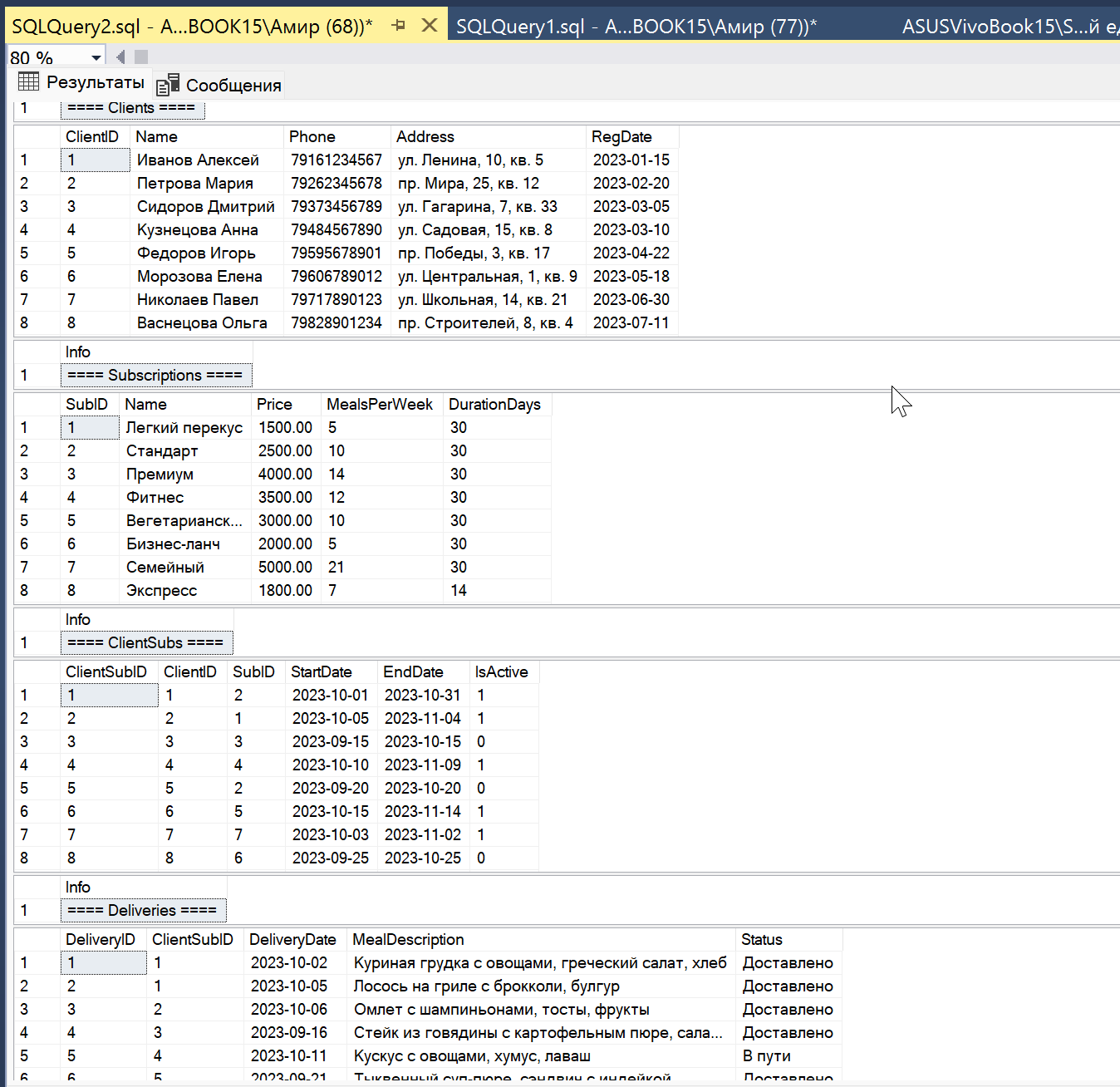


Рисунок 2 – Заполнение бд

1. Реализуйте:
   * 1. Минимум 3 хранимые процедуры, выполняющие сложную операцию (вставка, обновление, логика с условиями и параметрами);

Листинг 3 – Процедура добавления нового клиента и подписки

|  |
| --- |
| -- 1. Процедура добавления нового клиента и подписки (вставка)  CREATE PROCEDURE AddNewClientWithSubscription  @Name NVARCHAR(100),  @Phone NVARCHAR(20),  @Address NVARCHAR(200),  @SubID INT  AS  BEGIN  BEGIN TRY  BEGIN TRANSACTION;    -- Вставляем нового клиента  INSERT INTO Clients (Name, Phone, Address)  VALUES (@Name, @Phone, @Address);    DECLARE @NewClientID INT = SCOPE\_IDENTITY();  DECLARE @DurationDays INT;    -- Получаем длительность подписки  SELECT @DurationDays = DurationDays FROM Subscriptions WHERE SubID = @SubID;    -- Активируем подписку  INSERT INTO ClientSubs (ClientID, SubID, StartDate, EndDate, IsActive)  VALUES (  @NewClientID,  @SubID,  GETDATE(),  DATEADD(DAY, @DurationDays, GETDATE()),  1  );    COMMIT TRANSACTION;  SELECT 'Клиент и подписка успешно добавлены' AS Result;  END TRY  BEGIN CATCH  ROLLBACK TRANSACTION;  SELECT 'Ошибка: ' + ERROR\_MESSAGE() AS Result;  END CATCH;  END;  -- 1. Добавление нового клиента с подпиской  EXEC AddNewClientWithSubscription  @Name = 'Смирнов Александр',  @Phone = '79150001122',  @Address = 'ул. Новая, 15, кв. 3',  @SubID = 3; |

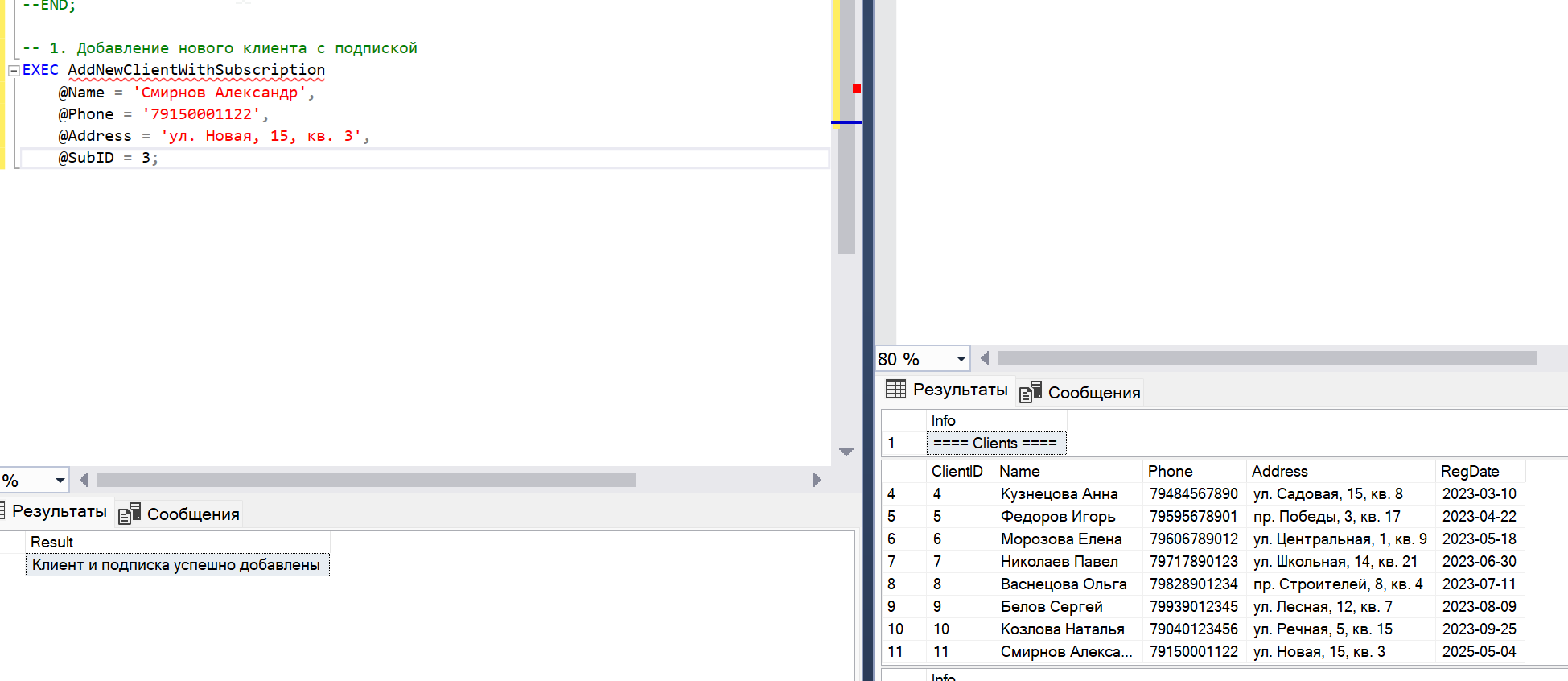


Рисунок 3 – Добавление нового клиента и подписки

Листинг 4 – Процедура обновления статуса доставки с проверками

|  |
| --- |
| -- 2. Процедура обновления статуса доставки с проверками  CREATE PROCEDURE UpdateDeliveryStatus  @DeliveryID INT,  @NewStatus NVARCHAR(20)  AS  BEGIN  -- Проверка существования доставки  IF NOT EXISTS (SELECT 1 FROM Deliveries WHERE DeliveryID = @DeliveryID)  BEGIN  SELECT 'Ошибка: Доставка с указанным ID не найдена' AS Result;  RETURN;  END    -- Проверка валидности статуса  IF @NewStatus NOT IN ('Ожидает', 'В пути', 'Доставлено', 'Отменена')  BEGIN  SELECT 'Ошибка: Неверный статус доставки' AS Result;  RETURN;  END    -- Получаем текущий статус  DECLARE @CurrentStatus NVARCHAR(20);  SELECT @CurrentStatus = [Status] FROM Deliveries WHERE DeliveryID = @DeliveryID;    -- Логика проверки переходов статусов  IF @CurrentStatus = 'Доставлено' AND @NewStatus != 'Доставлено'  BEGIN  SELECT 'Ошибка: Нельзя изменить статус уже доставленного заказа' AS Result;  RETURN;  END    IF @CurrentStatus = 'Отменена' AND @NewStatus != 'Отменена'  BEGIN  SELECT 'Ошибка: Нельзя изменить статус отмененного заказа' AS Result;  RETURN;  END    -- Обновляем статус  UPDATE Deliveries  SET [Status] = @NewStatus  WHERE DeliveryID = @DeliveryID;    SELECT 'Статус доставки успешно обновлен' AS Result;  END;  EXEC UpdateDeliveryStatus  @DeliveryID = 7,  @NewStatus = 'В пути'; |

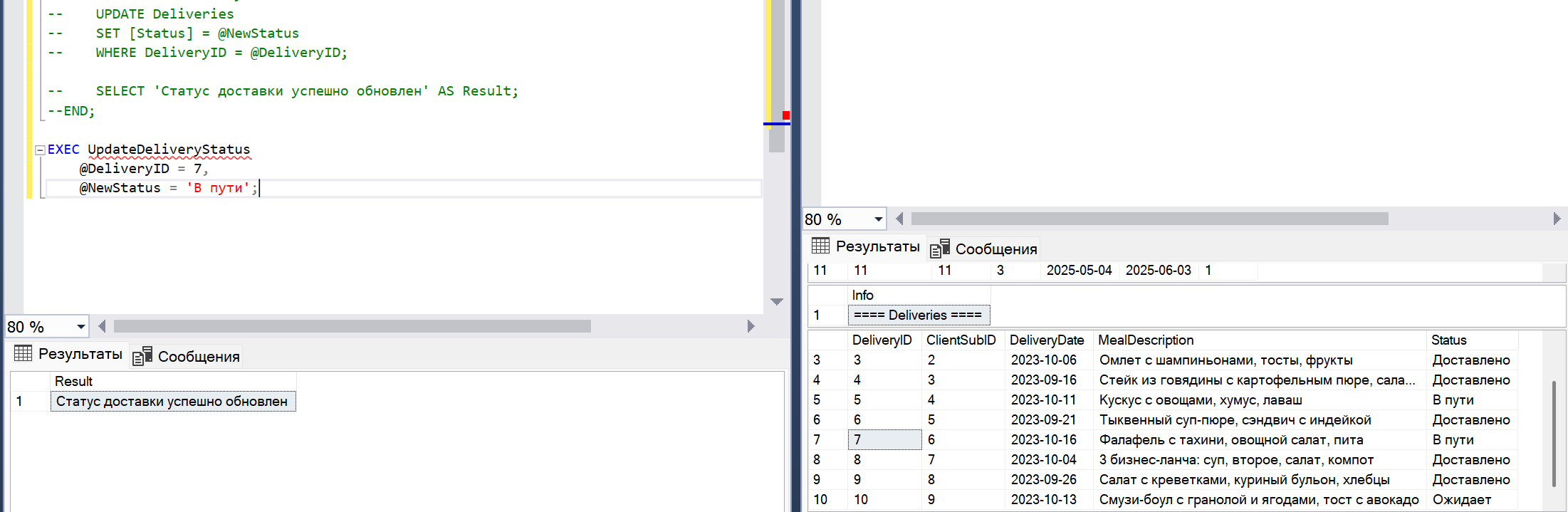


Рисунок 4 – Обновления статуса доставки

Листинг 5 – Процедура продления подписки с условиями

|  |
| --- |
| -- 3. Процедура продления подписки с условиями  CREATE PROCEDURE RenewSubscription  @ClientSubID INT,  @RenewalWeeks INT = 4 -- По умолчанию продлеваем на 4 недели  AS  BEGIN  SET NOCOUNT ON;    -- Проверка существования подписки  IF NOT EXISTS (SELECT 1 FROM ClientSubs WHERE ClientSubID = @ClientSubID)  BEGIN  SELECT 'Ошибка: Подписка не найдена' AS Result;  RETURN;  END    DECLARE @CurrentEndDate DATE;  DECLARE @IsActive BIT;    -- Получаем текущие данные подписки  SELECT  @CurrentEndDate = EndDate,  @IsActive = IsActive  FROM ClientSubs  WHERE ClientSubID = @ClientSubID;    -- Проверка активности подписки  IF @IsActive = 0  BEGIN  SELECT 'Ошибка: Нельзя продлить неактивную подписку' AS Result;  RETURN;  END    -- Проверка на слишком раннее продление (больше 7 дней до окончания)  IF DATEDIFF(DAY, GETDATE(), @CurrentEndDate) > 7  BEGIN  SELECT 'Ошибка: Подписку можно продлить не ранее чем за 7 дней до окончания' AS Result;  RETURN;  END    -- Продлеваем подписку  UPDATE ClientSubs  SET EndDate = DATEADD(WEEK, @RenewalWeeks, @CurrentEndDate)  WHERE ClientSubID = @ClientSubID;    SELECT  'Подписка успешно продлена' AS Result,  @ClientSubID AS SubscriptionID,  DATEADD(WEEK, @RenewalWeeks, @CurrentEndDate) AS NewEndDate;  END;  EXEC RenewSubscription  @ClientSubID = 2,  @RenewalWeeks = 2; |

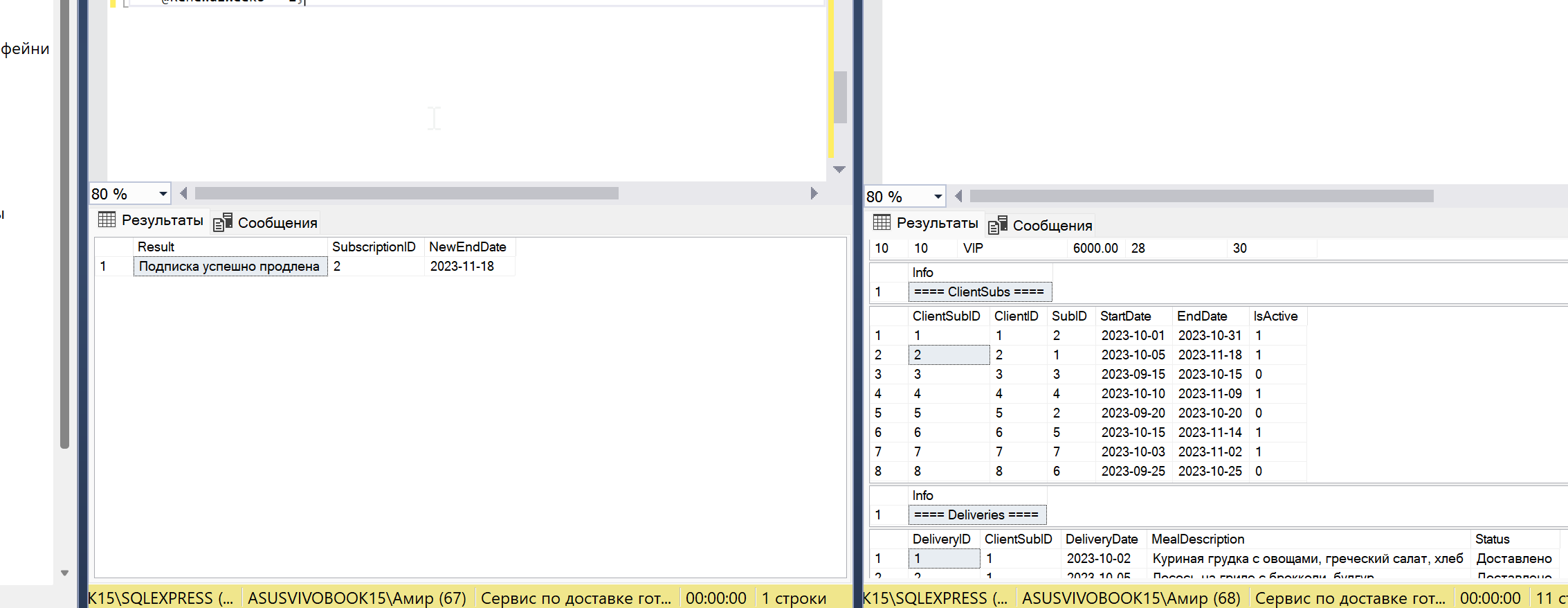


Рисунок 5 – продления подписки

* + 1. Минимум 3 триггера, срабатывающие на INSERT, UPDATE или DELETE, и реализующие контроль, логирование или автоматическое изменение других данных.

Листинг 6 – Триггер для автоматического обновления статуса подписки при изменении даты

|  |
| --- |
| -- 1. Триггер для автоматического обновления статуса подписки при изменении даты  CREATE TRIGGER trg\_UpdateSubscriptionStatus  ON ClientSubs  AFTER UPDATE  AS  BEGIN  UPDATE cs  SET IsActive =  CASE  WHEN GETDATE() BETWEEN i.StartDate AND i.EndDate THEN 1  ELSE 0  END  FROM ClientSubs cs  INNER JOIN inserted i ON cs.ClientSubID = i.ClientSubID  INNER JOIN deleted d ON cs.ClientSubID = d.ClientSubID  WHERE DATEDIFF(DAY, d.StartDate, d.EndDate) <> DATEDIFF(DAY, i.StartDate, i.EndDate);  END;  -- При изменении даты окончания подписки триггер автоматически обновит статус  UPDATE ClientSubs  SET EndDate = '2023-11-30'  WHERE ClientSubID = 1; |

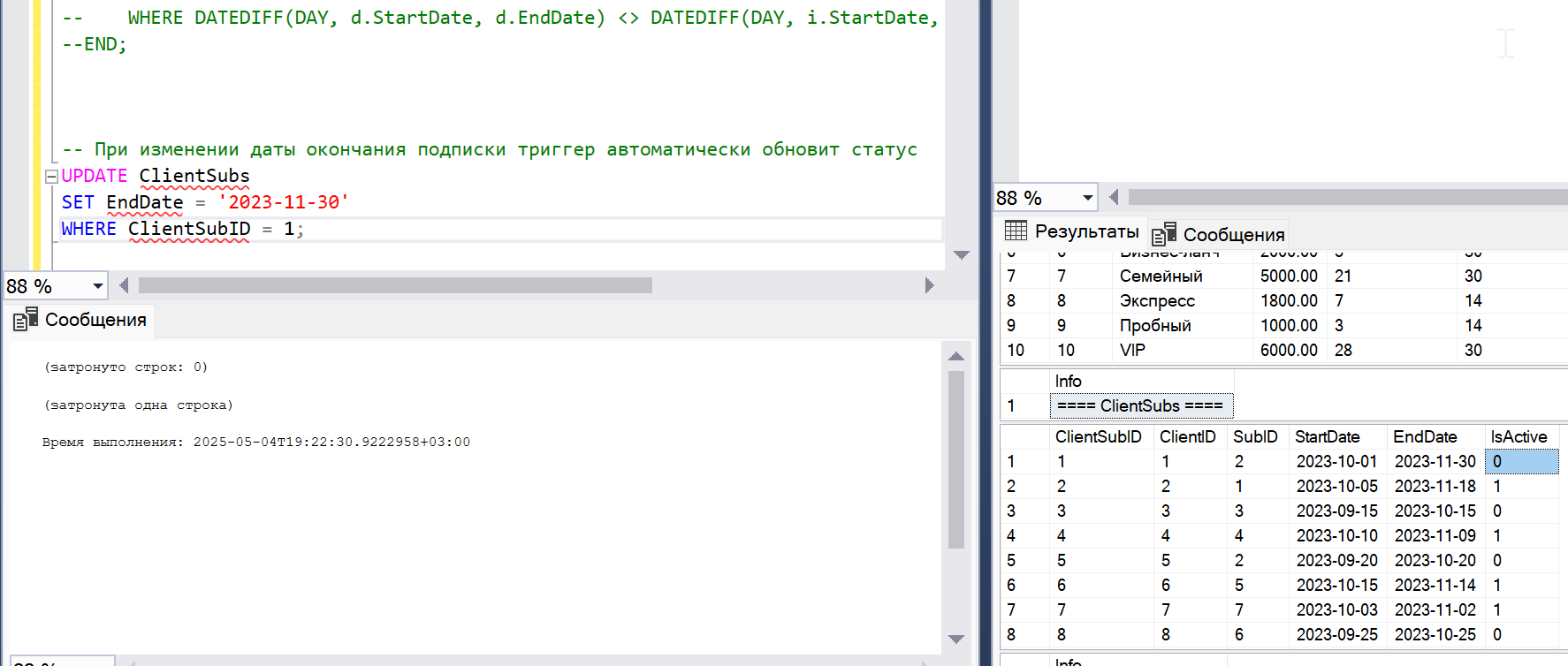


Рисунок 6 – Обновление статуса подписки при изменении даты

Листинг 7 – Триггер для логирования изменений в таблице Deliveries

|  |
| --- |
| -- 2. Триггер для логирования изменений в таблице Deliveries  -- Сначала создадим таблицу для логов  CREATE TABLE DeliveryLogs (  LogID INT IDENTITY(1,1) PRIMARY KEY,  DeliveryID INT,  OldStatus NVARCHAR(20),  NewStatus NVARCHAR(20),  ChangeDate DATETIME DEFAULT GETDATE(),  OperationType NVARCHAR(10) -- INSERT, UPDATE, DELETE  );  GO  -- Создаем триггер  CREATE TRIGGER trg\_LogDeliveryChanges  ON Deliveries  AFTER INSERT, UPDATE, DELETE  AS  BEGIN  -- Для операции INSERT  IF EXISTS (SELECT \* FROM inserted) AND NOT EXISTS (SELECT \* FROM deleted)  BEGIN  INSERT INTO DeliveryLogs (DeliveryID, OldStatus, NewStatus, OperationType)  SELECT  i.DeliveryID,  NULL,  i.Status,  'INSERT'  FROM inserted i;  END    -- Для операции UPDATE  IF EXISTS (SELECT \* FROM inserted) AND EXISTS (SELECT \* FROM deleted)  BEGIN  INSERT INTO DeliveryLogs (DeliveryID, OldStatus, NewStatus, OperationType)  SELECT  i.DeliveryID,  d.Status,  i.Status,  'UPDATE'  FROM inserted i  JOIN deleted d ON i.DeliveryID = d.DeliveryID  WHERE i.Status <> d.Status;  END    -- Для операции DELETE  IF NOT EXISTS (SELECT \* FROM inserted) AND EXISTS (SELECT \* FROM deleted)  BEGIN  INSERT INTO DeliveryLogs (DeliveryID, OldStatus, NewStatus, OperationType)  SELECT  d.DeliveryID,  d.Status,  NULL,  'DELETE'  FROM deleted d;  END  END;  -- Все изменения в Deliveries будут записаны в DeliveryLogs  UPDATE Deliveries  SET Status = 'Доставлено'  WHERE DeliveryID = 1;  -- Проверим логи  SELECT \* FROM DeliveryLogs; |

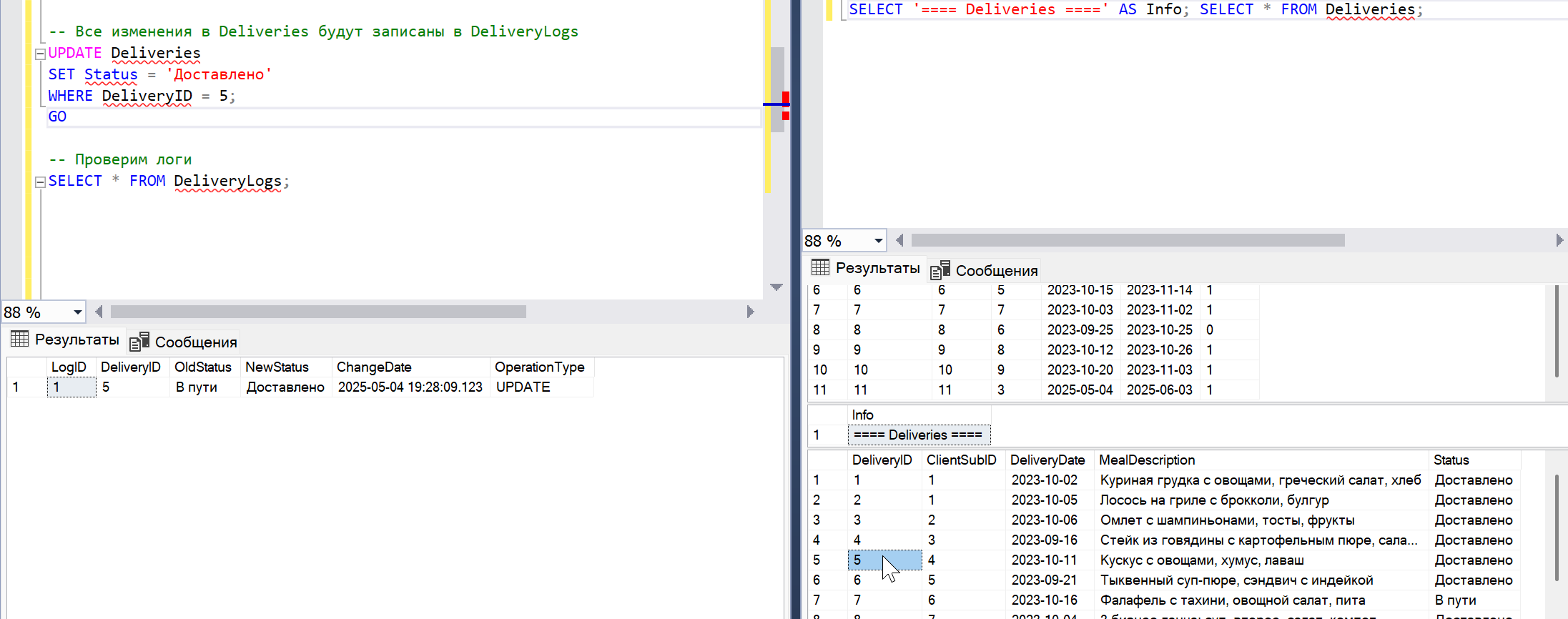


Рисунок 7 – Логирование изменений

Листинг 8 – Триггер для предотвращения пересечение дат активных подписок у одного клиента

|  |
| --- |
| -- 3. Триггер для предотвращения пересечение дат активных подписок у одного клиента  CREATE TRIGGER trg\_PreventSubscriptionOverlap  ON ClientSubs  AFTER INSERT, UPDATE  AS  BEGIN  -- Проверяем наличие пересекающихся подписок  IF EXISTS (  SELECT 1  FROM inserted i  JOIN ClientSubs cs ON i.ClientID = cs.ClientID  AND cs.ClientSubID != i.ClientSubID -- исключаем текущую запись  AND cs.IsActive = 1  WHERE  -- Проверяем пересечение дат: начало новой подписки внутри существующей  (i.StartDate BETWEEN cs.StartDate AND cs.EndDate)  OR  -- Или конец новой подписки внутри существующей  (i.EndDate BETWEEN cs.StartDate AND cs.EndDate)  OR  -- Или существующая подписка полностью внутри новой  (i.StartDate <= cs.StartDate AND i.EndDate >= cs.EndDate)  )  BEGIN  -- Отменяем операцию и выводим сообщение об ошибке  RAISERROR('Ошибка: У клиента уже есть активная подписка в указанный период', 16, 1);  ROLLBACK TRANSACTION;  RETURN;  END  END;  -- Пытаемся изменить даты существующей подписки так, чтобы она пересекалась с другой  UPDATE ClientSubs  SET StartDate = '2023-12-20', EndDate = '2024-01-20'  WHERE ClientSubID = 1; |

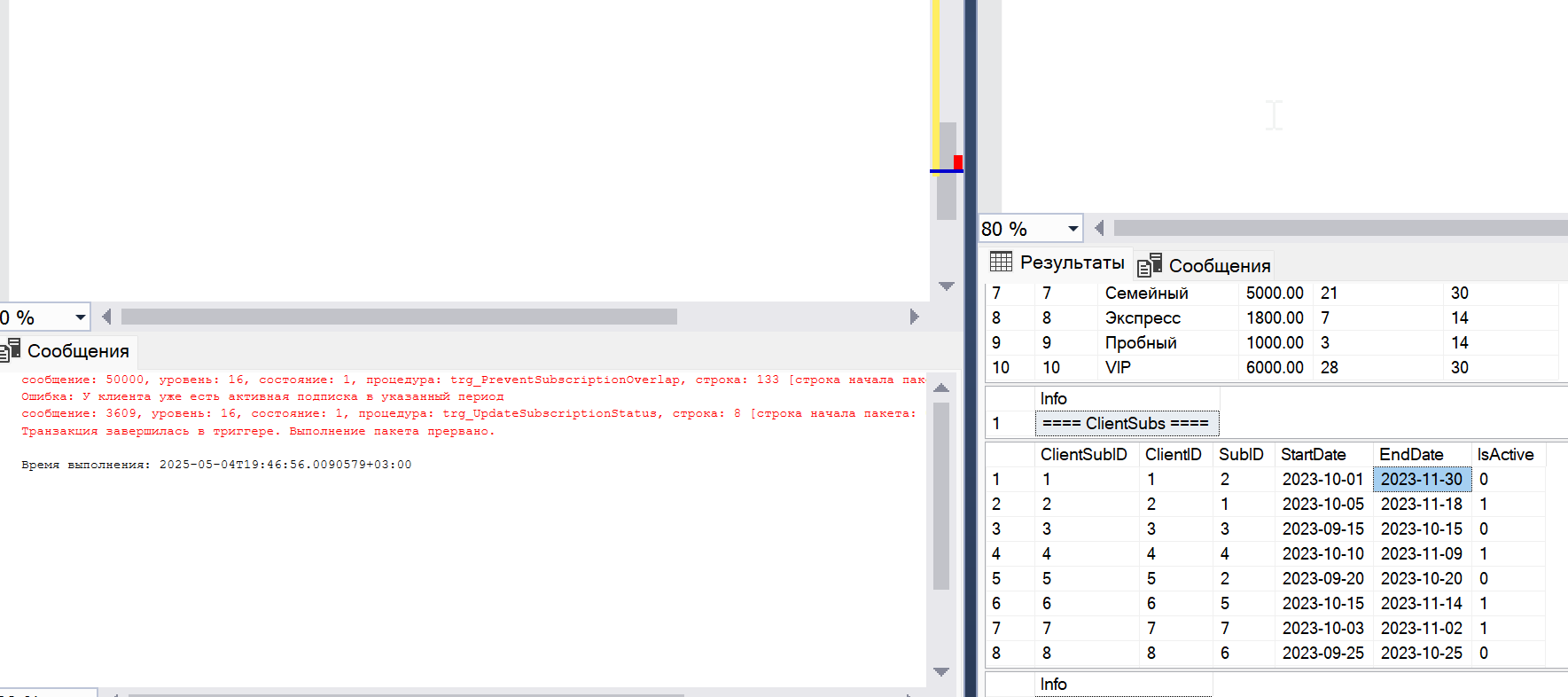


Рисунок 8 – Предотвращение пересечения дат

## **Отчет о выполнении работы 8**

Хранилище судов и исков для адвокатского бюро

1. Создание структуры базы данных:

* Минимум 3–4 связанные таблицы
* Нормализация до 3НФ
* Представления и CTE (минимум по одному)

Листинг 1 – Создание структуры бд и наполнение их данными

|  |
| --- |
| -- Создаем базу данных  -- Таблицы нормализованы до 3НФ  CREATE DATABASE LawFirmDB;  GO  USE LawFirmDB;  GO  -- 1. Таблица клиентов  CREATE TABLE Clients (  ClientID INT PRIMARY KEY IDENTITY(1,1),  FirstName NVARCHAR(50) NOT NULL,  LastName NVARCHAR(50) NOT NULL,  Phone NVARCHAR(20) NOT NULL,  Email NVARCHAR(100),  Address NVARCHAR(200)  );  -- 2. Таблица судов (нормализована до 3НФ)  CREATE TABLE Courts (  CourtID INT PRIMARY KEY IDENTITY(1,1),  CourtName NVARCHAR(100) NOT NULL,  CourtLevel NVARCHAR(50) NOT NULL,  Address NVARCHAR(200) NOT NULL  );  -- 3. Таблица адвокатов (нормализована до 3НФ)  CREATE TABLE Lawyers (  LawyerID INT PRIMARY KEY IDENTITY(1,1),  FirstName NVARCHAR(50) NOT NULL,  LastName NVARCHAR(50) NOT NULL,  LicenseNumber NVARCHAR(20) NOT NULL UNIQUE,  Specialization NVARCHAR(100)  );  -- 4. Таблица дел/исков (нормализована до 3НФ, связана с другими таблицами)  CREATE TABLE Cases (  CaseID INT PRIMARY KEY IDENTITY(1,1),  CaseNumber NVARCHAR(50) NOT NULL UNIQUE,  Description NVARCHAR(500) NOT NULL,  CaseType NVARCHAR(50) NOT NULL,  Status NVARCHAR(30) NOT NULL DEFAULT 'Открыто',  OpenDate DATE NOT NULL DEFAULT GETDATE(),  CloseDate DATE,  CourtID INT NOT NULL,  ClientID INT NOT NULL,  LawyerID INT NOT NULL,  FOREIGN KEY (CourtID) REFERENCES Courts(CourtID),  FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),  FOREIGN KEY (LawyerID) REFERENCES Lawyers(LawyerID)  );  -- Заполняем таблицу Клиентов  INSERT INTO Clients (FirstName, LastName, Phone, Email, Address)  VALUES  ('Иван', 'Петров', '+79161234567', 'ivan.petrov@mail.ru', 'г. Москва, ул. Ленина, д. 10, кв. 5'),  ('Елена', 'Сидорова', '+79262345678', 'elena.sidorova@gmail.com', 'г. Москва, ул. Пушкина, д. 15, кв. 12'),  ('Алексей', 'Кузнецов', '+79373456789', 'alex.kuznetsov@yandex.ru', 'г. Москва, пр. Мира, д. 25, кв. 30'),  ('Ольга', 'Васильева', '+79484567890', 'olga.vasileva@mail.ru', 'г. Москва, ул. Гагарина, д. 7, кв. 8'),  ('Дмитрий', 'Смирнов', '+79595678901', 'dmitry.smirnov@gmail.com', 'г. Москва, ул. Тверская, д. 20, кв. 15'),  ('Анна', 'Попова', '+79606789012', 'anna.popova@yandex.ru', 'г. Москва, ул. Чехова, д. 12, кв. 9'),  ('Сергей', 'Федоров', '+79717890123', 'sergey.fedorov@mail.ru', 'г. Москва, ул. Дзержинского, д. 5, кв. 3');  -- Заполняем таблицу Судов  INSERT INTO Courts (CourtName, CourtLevel, Address)  VALUES  ('Тверской районный суд г. Москвы', 'Районный', 'г. Москва, ул. Тверская, д. 18'),  ('Мещанский районный суд г. Москвы', 'Районный', 'г. Москва, ул. Мещанская, д. 25'),  ('Московский городской суд', 'Городской', 'г. Москва, ул. Богородский Вал, д. 8'),  ('Арбитражный суд г. Москвы', 'Арбитражный', 'г. Москва, ул. Большая Тульская, д. 17'),  ('Верховный суд РФ', 'Верховный', 'г. Москва, ул. Поварская, д. 15'),  ('Гагаринский районный суд г. Москвы', 'Районный', 'г. Москва, ул. Вавилова, д. 47'),  ('Преображенский районный суд г. Москвы', 'Районный', 'г. Москва, ул. Буженинова, д. 23');  -- Заполняем таблицу Адвокатов  INSERT INTO Lawyers (FirstName, LastName, LicenseNumber, Specialization)  VALUES  ('Александр', 'Иванов', 'АБ123456', 'Уголовное право'),  ('Мария', 'Соколова', 'АБ654321', 'Семейное право'),  ('Павел', 'Лебедев', 'АБ789012', 'Гражданское право'),  ('Екатерина', 'Новикова', 'АБ345678', 'Налоговое право'),  ('Артем', 'Морозов', 'АБ901234', 'Арбитражные споры'),  ('Наталья', 'Волкова', 'АБ567890', 'Трудовое право'),  ('Михаил', 'Павлов', 'АБ234567', 'Жилищное право');  -- Заполняем таблицу Дел/Исков  INSERT INTO Cases (CaseNumber, Description, CaseType, Status, OpenDate, CloseDate, CourtID, ClientID, LawyerID)  VALUES  ('А40-12345/2023', 'Взыскание долга по договору займа', 'Гражданское', 'Открыто', '2023-01-15', NULL, 1, 1, 3),  ('А41-67890/2023', 'Раздел имущества при разводе', 'Семейное', 'В процессе', '2023-02-20', NULL, 2, 2, 2),  ('А42-54321/2023', 'Оспаривание увольнения', 'Трудовое', 'Закрыто', '2023-03-10', '2023-06-15', 3, 3, 6),  ('А43-98765/2023', 'Налоговая проверка', 'Налоговое', 'Открыто', '2023-04-05', NULL, 4, 4, 4),  ('А44-12309/2023', 'Банкротство предприятия', 'Арбитражное', 'В процессе', '2023-05-12', NULL, 5, 5, 5),  ('А45-67543/2023', 'Уголовное дело о мошенничестве', 'Уголовное', 'Открыто', '2023-06-18', NULL, 6, 6, 1),  ('А46-23456/2023', 'Спор о праве собственности на квартиру', 'Жилищное', 'Открыто', '2023-07-22', NULL, 7, 7, 7); |

Листинг 2 – Создание представлений и CTE

|  |
| --- |
| -- Создаем представление для активных дел  CREATE VIEW ActiveCasesView AS  SELECT  c.CaseNumber,  c.Description,  c.CaseType,  c.OpenDate,  cl.FirstName + ' ' + cl.LastName AS ClientName,  l.FirstName + ' ' + l.LastName AS LawyerName,  crt.CourtName  FROM  Cases c  JOIN Clients cl ON c.ClientID = cl.ClientID  JOIN Lawyers l ON c.LawyerID = l.LawyerID  JOIN Courts crt ON c.CourtID = crt.CourtID  WHERE  c.Status = 'Открыто';  -- Пример CTE (Common Table Expression) для анализа загрузки адвокатов  WITH LawyerWorkload AS (  SELECT  l.LawyerID,  l.FirstName + ' ' + l.LastName AS LawyerName,  COUNT(c.CaseID) AS ActiveCasesCount  FROM  Lawyers l  LEFT JOIN Cases c ON l.LawyerID = c.LawyerID AND c.Status = 'Открыто'  GROUP BY  l.LawyerID, l.FirstName, l.LastName  )  SELECT  LawyerName,  ActiveCasesCount,  CASE  WHEN ActiveCasesCount > 5 THEN 'Высокая загрузка'  WHEN ActiveCasesCount > 2 THEN 'Средняя загрузка'  ELSE 'Низкая загрузка'  END AS WorkloadStatus  FROM  LawyerWorkload  ORDER BY  ActiveCasesCount DESC; |

Результат выполнения запроса, показан на рисунке 1.

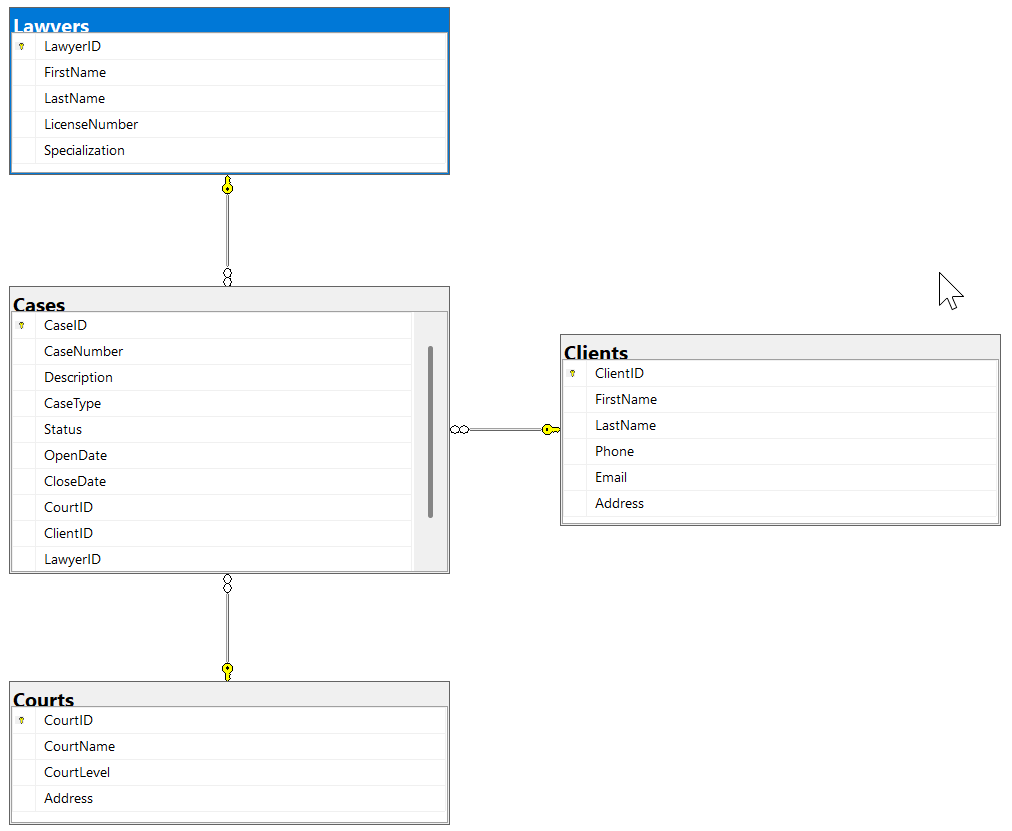


Рисунок 1 – Структура БД

Проверка работоспособности представления и CTE показано на рисунке 2.

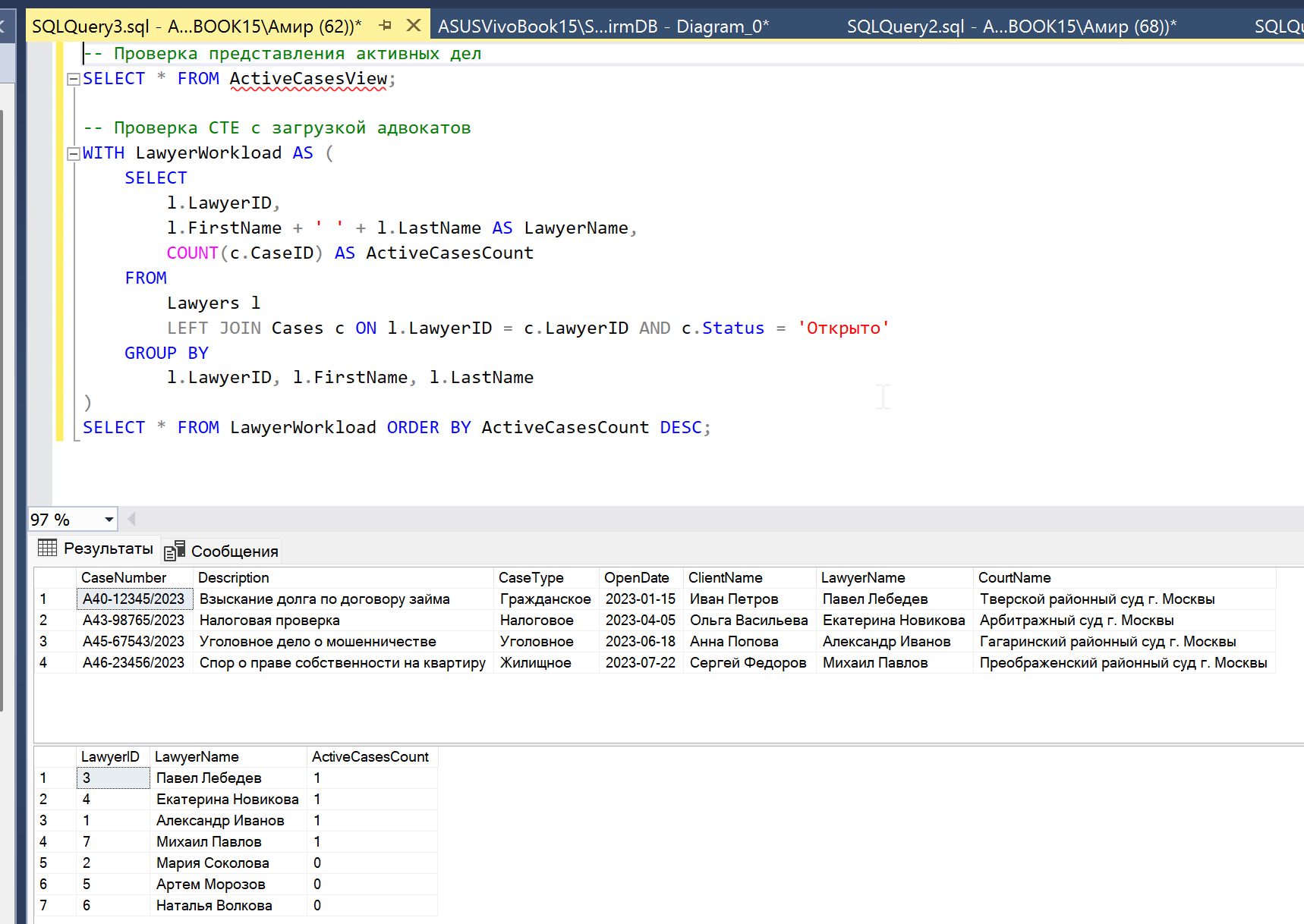


Рисунок 2 – Представления и CTE

1. Шифрование чувствительных данных:

* Выберите 1–2 атрибута для шифрования (с помощью EncryptByPassPhrase)
* Реализуйте процедуру шифрования и дешифрования данных
* Добавьте функцию для расшифровки (используйте DecryptByPassPhrase)

Листинг 3 – Подготовка данных для шифрования.

|  |
| --- |
| -- 1. Добавляем столбцы для хранения зашифрованных данных  ALTER TABLE Clients  ADD  EncryptedPassportSeries VARBINARY(MAX),  EncryptedPassportNumber VARBINARY(MAX);  GO  -- 2. Обновляем данные, шифруя существующие паспортные данные (если они есть)  -- Предположим, что мы добавили эти данные ранее  UPDATE Clients SET  EncryptedPassportSeries = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '1234'),  EncryptedPassportNumber = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '567890')  WHERE ClientID = 1;  UPDATE Clients SET  EncryptedPassportSeries = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '2345'),  EncryptedPassportNumber = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '678901')  WHERE ClientID = 2;  UPDATE Clients SET  EncryptedPassportSeries = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '3456'),  EncryptedPassportNumber = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '789012')  WHERE ClientID = 3;  UPDATE Clients SET  EncryptedPassportSeries = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '4567'),  EncryptedPassportNumber = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '890123')  WHERE ClientID = 4;  UPDATE Clients SET  EncryptedPassportSeries = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '5678'),  EncryptedPassportNumber = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '901234')  WHERE ClientID = 5;  UPDATE Clients SET  EncryptedPassportSeries = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '6789'),  EncryptedPassportNumber = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '012345')  WHERE ClientID = 6;  UPDATE Clients SET  EncryptedPassportSeries = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '7890'),  EncryptedPassportNumber = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', '123456')  WHERE ClientID = 7; |

Результат показан на рисунке 3.

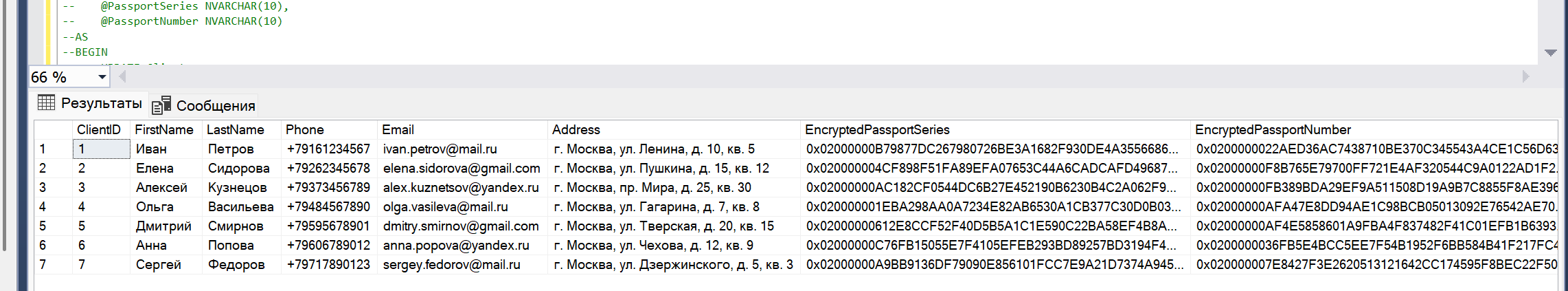


Рисунок 3 – Шифрование данных

Листинг 4 – Реализация процедур шифрования и дешифрования данных.

|  |
| --- |
| -- 3. Создаем процедуру для обновления зашифрованных паспортных данных  CREATE PROCEDURE dbo.UpdateClientPassportData  @ClientID INT,  @PassportSeries NVARCHAR(10),  @PassportNumber NVARCHAR(10)  AS  BEGIN  UPDATE Clients  SET  EncryptedPassportSeries = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', @PassportSeries),  EncryptedPassportNumber = ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', @PassportNumber)  WHERE ClientID = @ClientID;    RETURN @@ROWCOUNT;  END;  GO  -- 4. Создаем процедуру для получения расшифрованных паспортных данных  CREATE PROCEDURE dbo.GetClientPassportData  @ClientID INT  AS  BEGIN  SELECT  ClientID,  FirstName,  LastName,  dbo.DecryptClientPassportData(@ClientID, 'Series') AS PassportSeries,  dbo.DecryptClientPassportData(@ClientID, 'Number') AS PassportNumber  FROM Clients  WHERE ClientID = @ClientID;  END;  GO  -- Проверка процедуры обновления данных  EXEC dbo.UpdateClientPassportData  @ClientID = 1,  @PassportSeries = '9999',  @PassportNumber = '888888';  -- Проверка процедуры получения данных  EXEC [dbo].[GetClientPassportData] @ClientID = 4; |

Результат показан на рисунке 4.

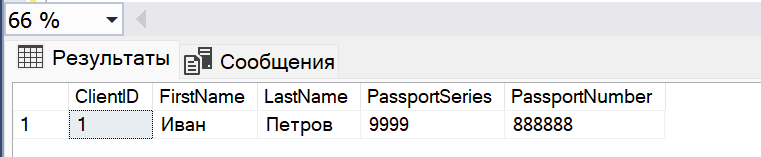


Рисунок 4 – Реализация процедур

Листинг 5 – Функция для расшифровки.

|  |
| --- |
| -- 5. Создаем функцию для расшифровки паспортных данных  CREATE FUNCTION dbo.DecryptClientPassportData(  @ClientID INT,  @DataType VARCHAR(20) -- 'Series' или 'Number'  )  RETURNS NVARCHAR(10)  AS  BEGIN  DECLARE @Result NVARCHAR(10);    IF @DataType = 'Series'  SELECT @Result = CONVERT(NVARCHAR(10), DECRYPTBYPASSPHRASE('LawFirmSecretKey123', EncryptedPassportSeries))  FROM Clients  WHERE ClientID = @ClientID;  ELSE IF @DataType = 'Number'  SELECT @Result = CONVERT(NVARCHAR(10), DECRYPTBYPASSPHRASE('LawFirmSecretKey123', EncryptedPassportNumber))  FROM Clients  WHERE ClientID = @ClientID;    RETURN @Result;  END;  GO  -- Проверка функции расшифровки  SELECT  ClientID,  FirstName,  LastName,  dbo.DecryptClientPassportData(ClientID, 'Series') AS DecryptedSeries,  dbo.DecryptClientPassportData(ClientID, 'Number') AS DecryptedNumber  FROM Clients  WHERE ClientID IN (1, 2, 3); |

Результат показан на рисунке 5.

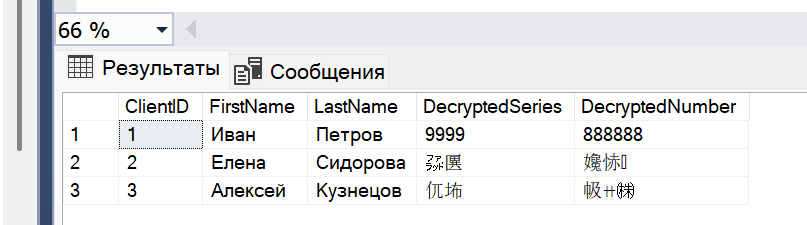


Рисунок 5 – Функция для расшифровки

1. Безопасность. Роли и разграничение доступа:

* Создайте две роли пользователей: DataReader и DataManager
* Ограничьте доступ к таблицам / представлениям / процедурам
* Проверьте выполнение с разных ролей

Листинг 6 – Создание ролей

|  |
| --- |
| -- 1. Создаем роли пользователей  CREATE ROLE DataReader;  GO  CREATE ROLE DataManager;  GO |

Результат показан на рисунке 6.

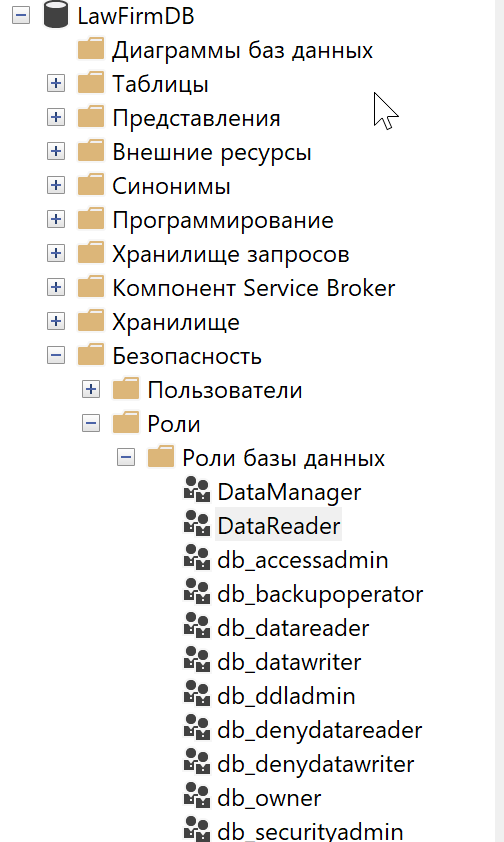


Рисунок 6 – Роли

Листинг 7 – Функция для расшифровки.

|  |
| --- |
| -- 2. Настраиваем права для роли DataReader (только чтение)  -- Разрешаем SELECT для основных таблиц  GRANT SELECT ON OBJECT::dbo.Clients TO DataReader;  GRANT SELECT ON OBJECT::dbo.Courts TO DataReader;  GRANT SELECT ON OBJECT::dbo.Lawyers TO DataReader;  GRANT SELECT ON OBJECT::dbo.Cases TO DataReader;  -- Разрешаем доступ к представлениям  GRANT SELECT ON OBJECT::dbo.ActiveCasesView TO DataReader;  -- Разрешаем выполнение процедуры получения данных (без права изменения)  GRANT EXECUTE ON OBJECT::dbo.GetClientPassportData TO DataReader;  -- 3. Настраиваем права для роли DataManager (чтение и запись)  -- Наследуем все права DataReader  ALTER ROLE DataReader ADD MEMBER DataManager;  -- Дополнительные права на изменение данных  GRANT INSERT, UPDATE, DELETE ON OBJECT::dbo.Clients TO DataManager;  GRANT INSERT, UPDATE, DELETE ON OBJECT::dbo.Cases TO DataManager;  -- Разрешаем выполнение процедуры обновления данных  GRANT EXECUTE ON OBJECT::dbo.UpdateClientPassportData TO DataManager;  -- Создаем тестовых пользователей  IF NOT EXISTS (SELECT \* FROM sys.server\_principals WHERE name = 'reader\_user')  CREATE LOGIN reader\_user WITH PASSWORD = 'Reader@123';  GO  IF NOT EXISTS (SELECT \* FROM sys.database\_principals WHERE name = 'reader\_user')  CREATE USER reader\_user FOR LOGIN reader\_user;  GO  IF NOT EXISTS (SELECT \* FROM sys.server\_principals WHERE name = 'manager\_user')  CREATE LOGIN manager\_user WITH PASSWORD = 'Manager@123';  GO  IF NOT EXISTS (SELECT \* FROM sys.database\_principals WHERE name = 'manager\_user')  CREATE USER manager\_user FOR LOGIN manager\_user;  GO  -- Назначаем роли пользователям  ALTER ROLE DataReader ADD MEMBER reader\_user;  ALTER ROLE DataManager ADD MEMBER manager\_user; |

Результат показан на рисунке 7.

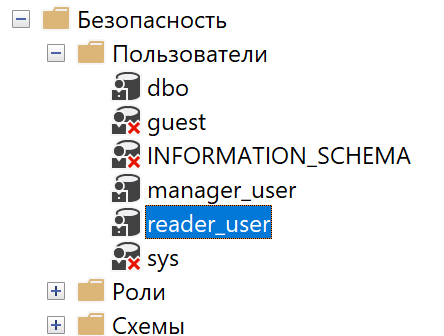
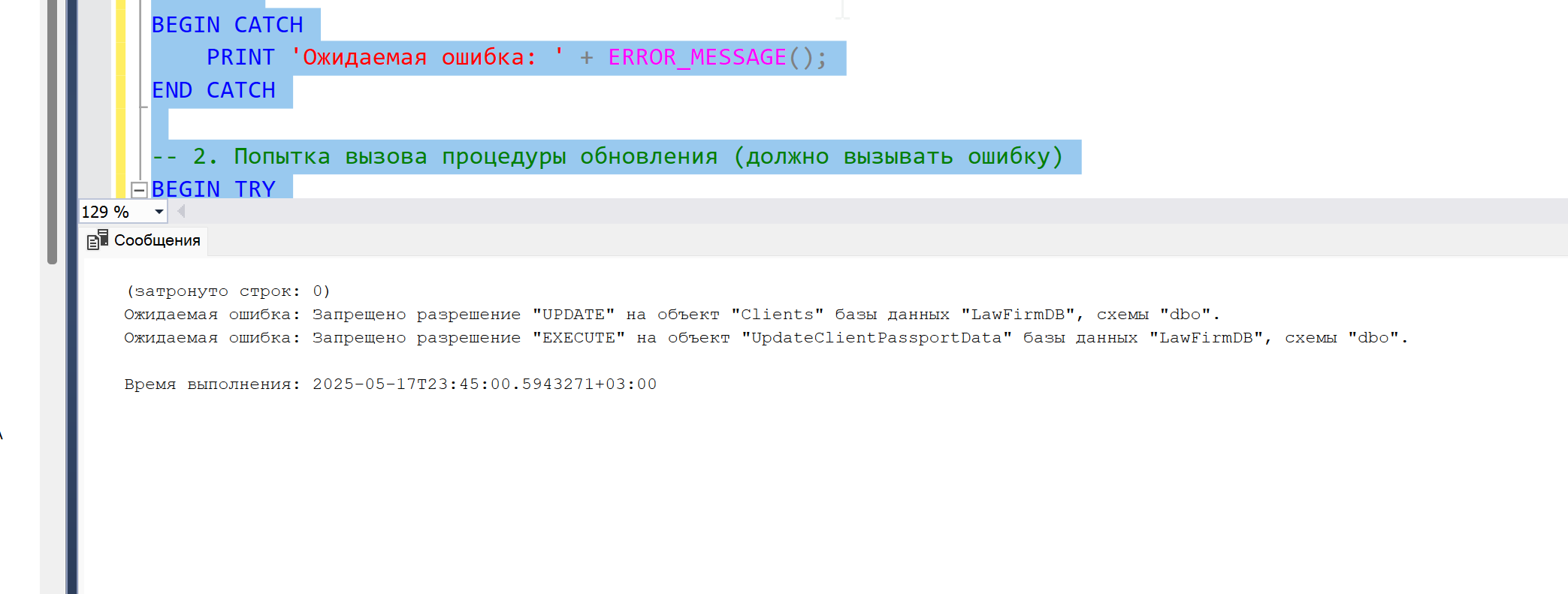


Рисунок 7 – Функция для расшифровки

Листинг 8 – Проверка ролей.

|  |
| --- |
| -- Войдем под пользователем reader\_user  EXECUTE AS USER = 'reader\_user';  GO  -- Проверяем доступные действия:  -- 1. Попытка изменения данных (должно вызывать ошибку)  BEGIN TRY  UPDATE dbo.Clients SET Phone = '123456789' WHERE ClientID = 1;  PRINT 'ОШИБКА: Обновление прошло успешно, но не должно было!';  END TRY  BEGIN CATCH  PRINT 'Ожидаемая ошибка: ' + ERROR\_MESSAGE();  END CATCH  -- 2. Попытка вызова процедуры обновления (должно вызывать ошибку)  BEGIN TRY  EXEC dbo.UpdateClientPassportData  @ClientID = 1,  @PassportSeries = '9999',  @PassportNumber = '888888';  PRINT 'ОШИБКА: Процедура выполнилась, но не должна была!';  END TRY  BEGIN CATCH  PRINT 'Ожидаемая ошибка: ' + ERROR\_MESSAGE();  END CATCH  -- Выходим из контекста пользователя  REVERT;  GO  -- Войдем под пользователем manager\_user  EXECUTE AS USER = 'manager\_user';  GO  -- Проверяем доступные действия:  -- 1. Чтение данных (должно работать)  SELECT TOP 3 \* FROM dbo.Clients;  -- 2. Изменение данных (должно работать)  BEGIN TRY  UPDATE dbo.Clients SET Phone = '+79991112233' WHERE ClientID = 1;  PRINT 'Обновление прошло успешно (как и ожидалось)';    -- Проверяем обновление  SELECT Phone FROM dbo.Clients WHERE ClientID = 1;  END TRY  BEGIN CATCH  PRINT 'НЕОЖИДАННАЯ ошибка: ' + ERROR\_MESSAGE();  END CATCH  -- 3. Вызов процедуры обновления паспортных данных (должно работать)  BEGIN TRY  EXEC dbo.UpdateClientPassportData  @ClientID = 1,  @PassportSeries = '9999',  @PassportNumber = '888888';  PRINT 'Процедура обновления выполнена успешно (как и ожидалось)';    -- Проверяем через процедуру чтения  EXEC dbo.GetClientPassportData @ClientID = 1;  END TRY  BEGIN CATCH  PRINT 'НЕОЖИДАННАЯ ошибка: ' + ERROR\_MESSAGE();  END CATCH  -- 4. Попытка прямого доступа к зашифрованным данным (должно вызывать ошибку)  BEGIN TRY  SELECT EncryptedPassportSeries FROM dbo.Clients WHERE ClientID = 1;  PRINT 'ОШИБКА: Доступ к зашифрованным данным получен, но не должен был!';  END TRY  BEGIN CATCH  PRINT 'Ожидаемая ошибка: ' + ERROR\_MESSAGE();  END CATCH  -- Выходим из контекста пользователя  REVERT;  GO |

Результат показан на рисунке 8.



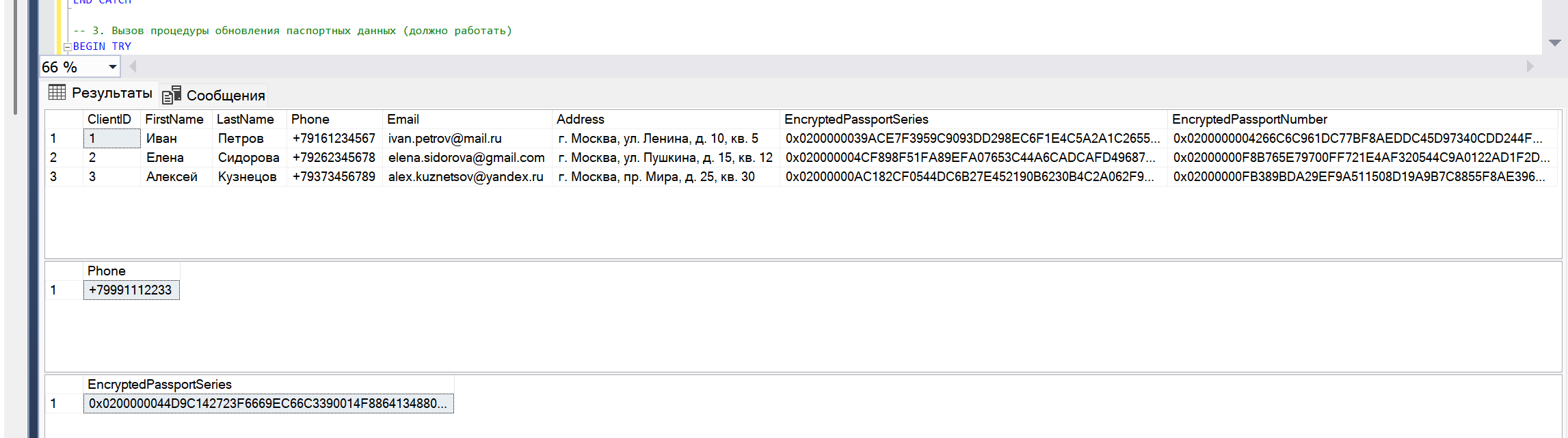


Рисунок 8 – Проверка ролей

Листинг 9 – Процедура вставки зашифрованных данных

|  |
| --- |
| ---- Создаем процедуру для безопасного добавления клиента с шифрованием паспортных данных  --CREATE OR ALTER PROCEDURE dbo.AddClientWithEncryptedData  -- @FirstName NVARCHAR(50),  -- @LastName NVARCHAR(50),  -- @Phone NVARCHAR(20),  -- @Email NVARCHAR(100) = NULL,  -- @Address NVARCHAR(200) = NULL,  -- @PassportSeries NVARCHAR(10),  -- @PassportNumber NVARCHAR(10)  --AS  --BEGIN  -- SET NOCOUNT ON;    -- BEGIN TRY  -- BEGIN TRANSACTION;    -- -- Вставляем данные клиента с шифрованием паспортных данных  -- INSERT INTO Clients (  -- FirstName,  -- LastName,  -- Phone,  -- Email,  -- Address,  -- EncryptedPassportSeries,  -- EncryptedPassportNumber  -- )  -- VALUES (  -- @FirstName,  -- @LastName,  -- @Phone,  -- @Email,  -- @Address,  -- ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', @PassportSeries),  -- ENCRYPTBYPASSPHRASE('LawFirmSecretKey123', @PassportNumber)  -- );    -- -- Возвращаем ID нового клиента  -- DECLARE @NewClientID INT = SCOPE\_IDENTITY();    -- -- Логируем действие  -- INSERT INTO SecurityLog (ActionType, TableName, RecordID, UserName)  -- VALUES ('INSERT', 'Clients', @NewClientID, USER\_NAME());    -- COMMIT TRANSACTION;    -- -- Возвращаем результат  -- SELECT  -- @NewClientID AS ClientID,  -- 'Клиент успешно добавлен' AS Message;  -- END TRY  -- BEGIN CATCH  -- IF @@TRANCOUNT > 0  -- ROLLBACK TRANSACTION;    -- DECLARE @ErrorMessage NVARCHAR(4000) = ERROR\_MESSAGE();  -- RAISERROR('Ошибка при добавлении клиента: %s', 16, 1, @ErrorMessage);  -- RETURN -1;  -- END CATCH;  --END;  --GO  ---- Таблица для логирования действий с зашифрованными данными  --IF NOT EXISTS (SELECT \* FROM sys.tables WHERE name = 'SecurityLog')  --BEGIN  -- CREATE TABLE SecurityLog (  -- LogID INT IDENTITY(1,1) PRIMARY KEY,  -- ActionTime DATETIME NOT NULL DEFAULT GETDATE(),  -- ActionType NVARCHAR(50) NOT NULL, -- INSERT, UPDATE, SELECT\_DECRYPTED, UNAUTHORIZED\_ACCESS  -- TableName NVARCHAR(50) NOT NULL,  -- RecordID INT NULL,  -- UserName NVARCHAR(100) NOT NULL,  -- Details NVARCHAR(MAX) NULL  -- );    -- -- Создаем индекс для быстрого поиска по логам  -- CREATE INDEX IX\_SecurityLog\_ActionTime ON SecurityLog(ActionTime);  -- CREATE INDEX IX\_SecurityLog\_User ON SecurityLog(UserName);  --END  --GO  ---- Тестирование процедуры добавления клиента  --EXEC dbo.AddClientWithEncryptedData  -- @FirstName = 'Андрей1',  -- @LastName = 'Семенов2',  -- @Phone = '+79167778891',  -- @Email = 'andrey1.semenov@example.com',  -- @Address = 'г. Москва, ул. Ленинградская, д. 15, кв. 42',  -- @PassportSeries = '12345',  -- @PassportNumber = '5678900';  --GO |

Результат показан на рисунке 9.

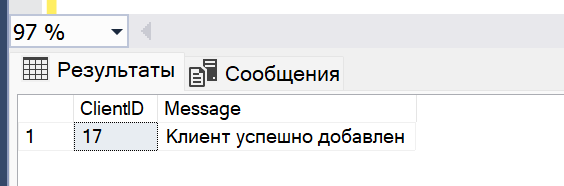


Рисунок 9 –Вставка зашифрованных данных

Листинг 10 – Процедура выборки с расшифровкой

|  |
| --- |
| -- Процедура для безопасного получения данных клиента с расшифровкой паспортных данных  CREATE OR ALTER PROCEDURE dbo.GetClientDecryptedData  @ClientID INT  AS  BEGIN  SET NOCOUNT ON;    -- Просто выбираем данные с расшифровкой без сложных проверок  SELECT  c.ClientID,  c.FirstName,  c.LastName,  c.Phone,  CONVERT(NVARCHAR(10), DECRYPTBYPASSPHRASE('LawFirmSecretKey123', c.EncryptedPassportSeries)) AS PassportSeries,  CONVERT(NVARCHAR(10), DECRYPTBYPASSPHRASE('LawFirmSecretKey123', c.EncryptedPassportNumber)) AS PassportNumber  FROM  Clients c  WHERE  c.ClientID = @ClientID;  END;  GO  -- Вызов процедуры  EXEC dbo.GetClientDecryptedData @ClientID = 1; |

Результат показан на рисунке 10.

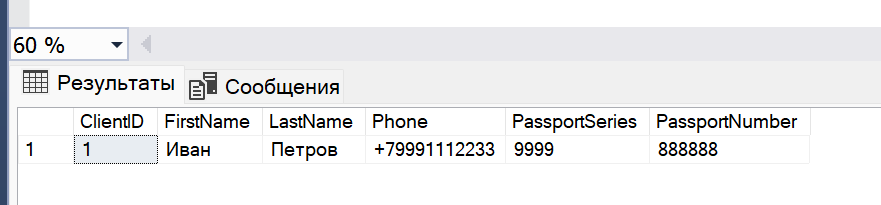


Рисунок 10 – Функция для расшифровки