МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

#### Институт среднего профессионального образования

А.О. Резанова, Д.А. Тузова

**Технологии разработки**

**и защиты баз данных**

Изображение выглядит как рисунок, зарисовка, строительство, на открытом воздухе

Автоматически созданное описание

Санкт-Петербург

2021

**СОДЕРЖАНИЕ**

[**Лабораторная работа №1.** 3](#_Toc148723130)

[**Лабораторная работа №2.** 3](#_Toc148723131)

[**Лабораторная работа №3.** 3](#_Toc148723132)

[**Лабораторная работа №4.** 3](#_Toc148723133)

[**Лабораторная работа №5.** 3](#_Toc148723134)

[**Лабораторная работа №6.** 3](#_Toc148723135)

[**Лабораторная работа №7.** 3](#_Toc148723136)

[**Лабораторная работа №8.** 3](#_Toc148723137)

[**Лабораторная работа №9.** 3](#_Toc148723138)

[**Лабораторная работа №10.** 3](#_Toc148723139)

[**Лабораторная работа №11.** 3](#_Toc148723140)

[**Лабораторная работа №12.** 3](#_Toc148723141)

[**Лабораторная работа №13.** 3](#_Toc148723142)

[**Лабораторная работа №14.** 3](#_Toc148723143)

# **Лабораторная работа №1.**

**Установка соединения с сервером Microsoft SQL Server. Создание, копирование и удаление базы данных.**

Цель: Познакомиться с основными принципами создания базы данных в MS SQL Server. Получить навыки для создания, удаления, резервного копирования и восстановления базы данных. Изучить SQL-операторы для создания, подключения и удаления базы данных.

Задание на лабораторную работу:

1. Откройте SQL Server Management Studio.
2. Подключитесь к серверу.
3. Создайте базу данных с помощью контекстного меню с именем Ваша\_Фамилия\_Номер\_группы\_номер\_в\_журнале.
4. Извлеките команду для создания БД.
5. Удалите БД с помощью контекстного меню.
6. Создайте БД с помощью скрипта с именем ВашаФамилия\_ПредметнаяОбластьИзЛР1.
7. Создайте резервную копию этой БД.
8. Удалите БД с помощью SQL-кода.
9. Восстановите БД из файла с резервной копией.
10. Сохраните файл сценария.

Подключение к серверу (Рисунок 1).

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 1. Рисунок 2.

Создайте базу данных с помощью контекстного меню (Рисунок 2).



Рисунок 3. Созданная база данных.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4. Команда для создания БД.



Рисунок 5. Удалите БД с помощью SQL-кода.

Создание резервной копии этой БД (Рисунок 6-7).

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 6.

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рисунок 7.

Восстановите БД из файла с резервной копией (Рисунок 8-9).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 8.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 9.

# **Лабораторная работа №2.**

**Импорт данных. Работа с csv и txt файлам.**

Цель: Научиться правильно импортировать данные разных форматов, а также экспортировать данные.

Задание на лабораторную работу:

1. Спросите у преподавателя, где находятся файлы для импорта.
2. Импортируйте каждый и них в вашу БД.
3. Следуя указаниям «мастера импорта и экспорта», экспортируйте любую таблицу в csv или txt файл.
4. Вставьте в отчет выбранные вами свойства и результат экспорта.

Изображение выглядит как текст, электроника, снимок экрана, число

Автоматически созданное описание Изображение выглядит как текст, электроника, снимок экрана, компьютер

Автоматически созданное описание

Рисунок 1. Рисунок 2.

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 3. Рисунок 4.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 5. Рисунок 6.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание Изображение выглядит как электроника, текст, дисплей, снимок экрана

Автоматически созданное описание

Рисунок 7. Рисунок 8.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 9. Рисунок 10.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 11. Рисунок 12.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 13. Рисунок 14.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Рисунок 15. Рисунок 16.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 17. Рисунок 18.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

Рисунок 19. Рисунок 20.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 21.

# **Лабораторная работа №3.**

**Создание и заполнение таблиц.**

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

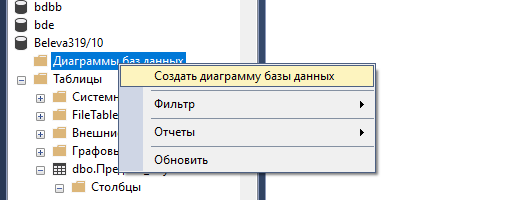
Автоматически созданное описание

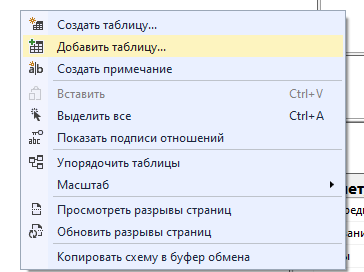
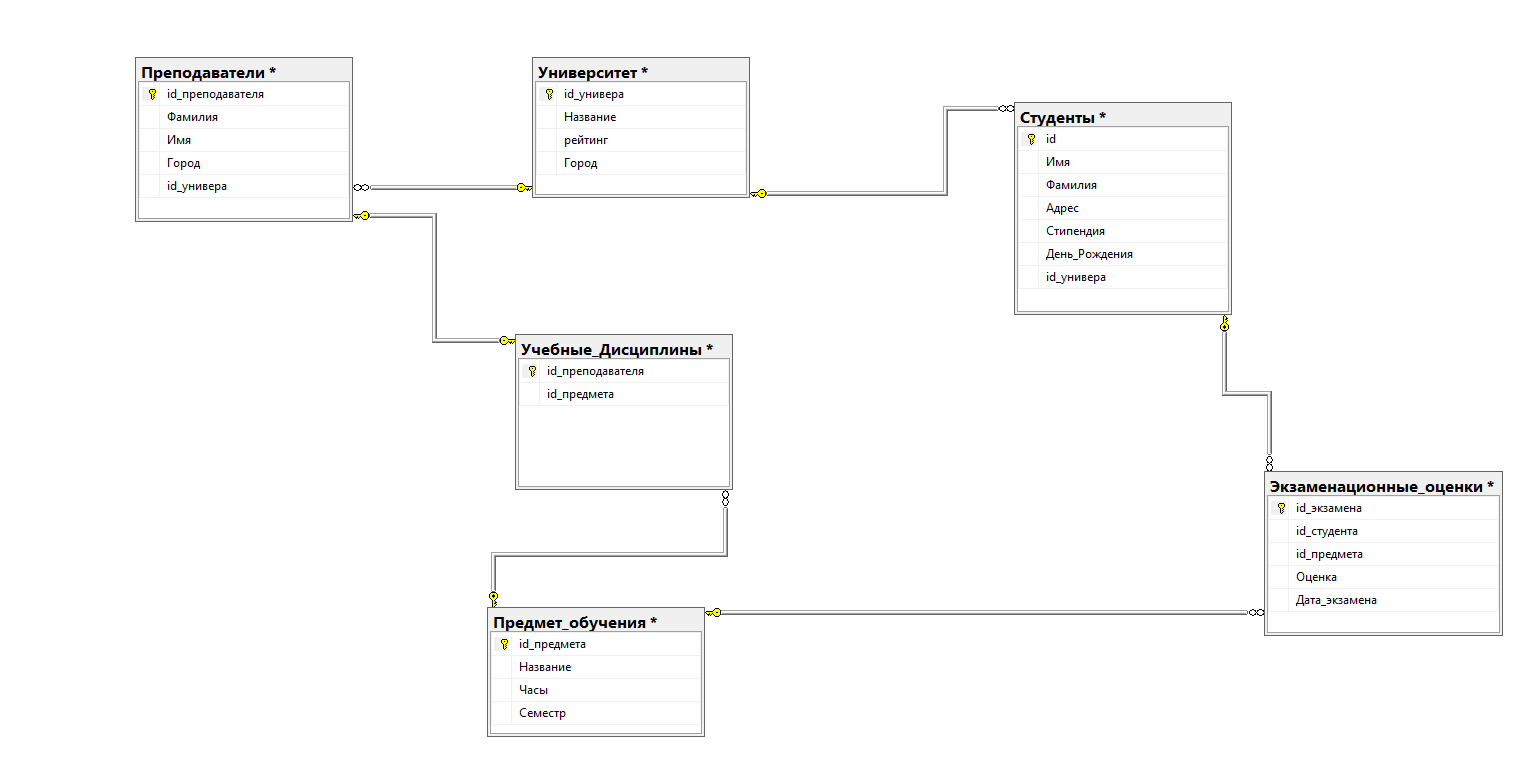
Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

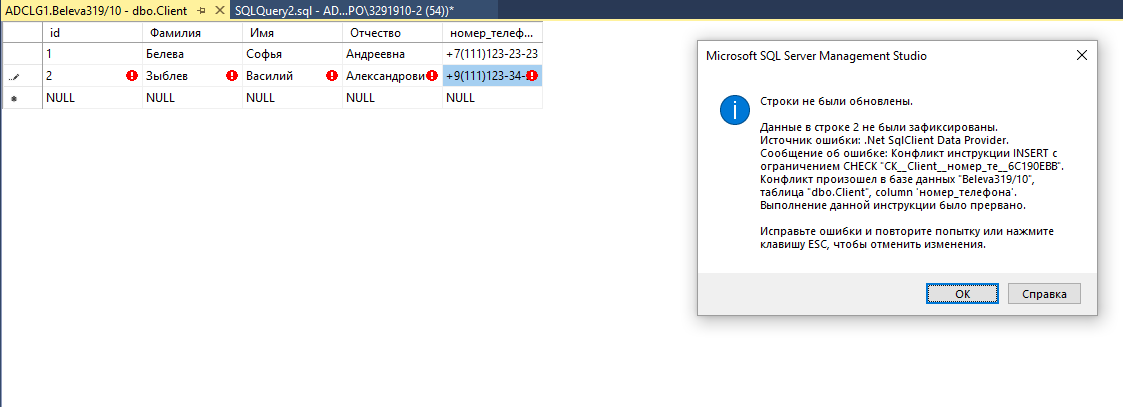
Автоматически созданное описание

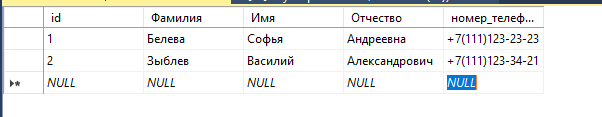
# **Лабораторная работа №4.**

**Разработка ограничений.**

****

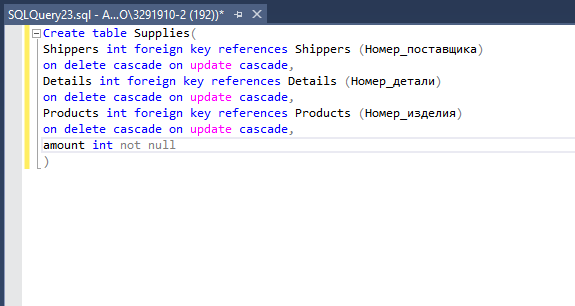
****

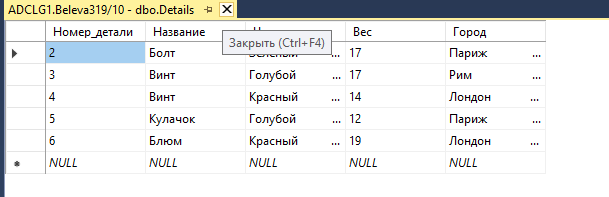


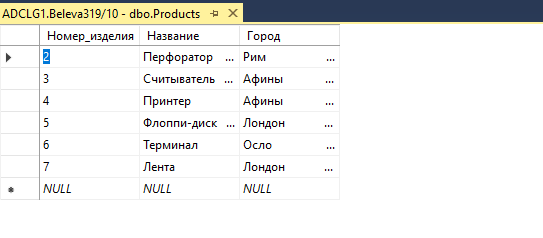


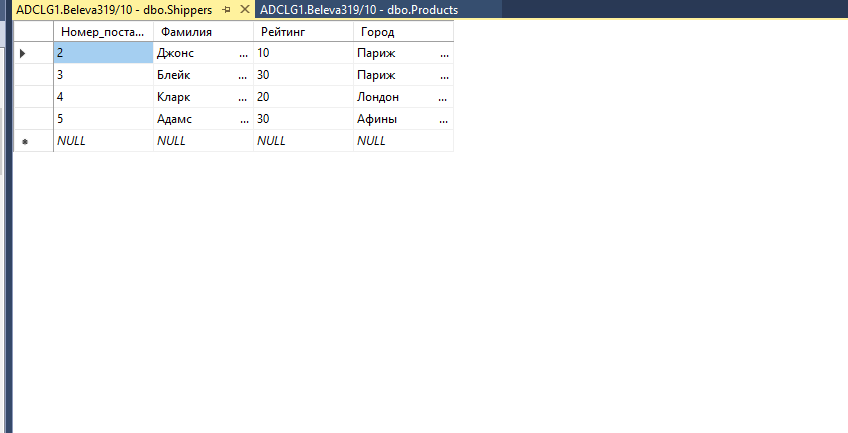
# **Лабораторная работа №5.**

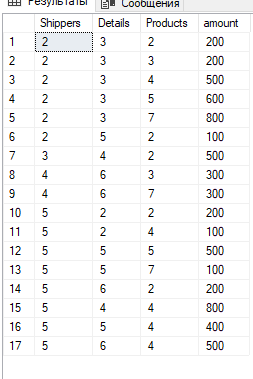
**Работа с внешними ключами**

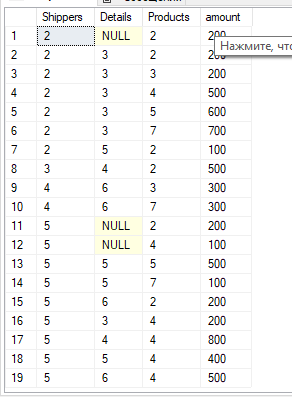
****

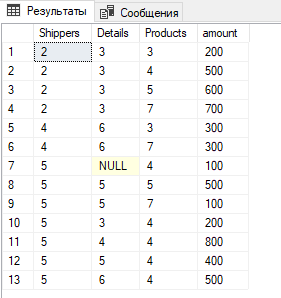
****

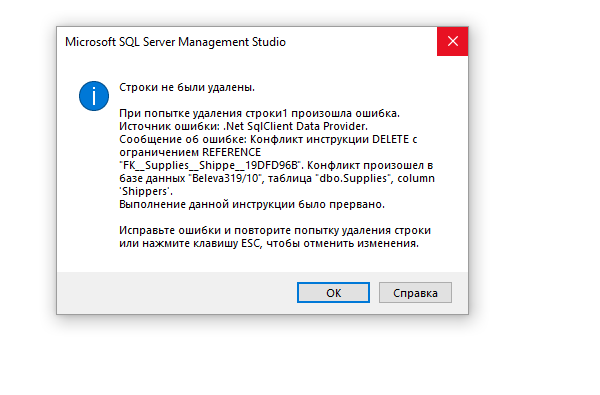
****

****



****



****

**Вывод:**

**Целесообразнее использовать on delete т.к. у нас идет операция удаления из таблицы, а не обновления, например с «1» на «2».**

**Контрольные вопросы:**

1. Какие типы ограничений вы знаете? Какие у них назначения?

* ограничения целостности

используются для обеспечения корректности данных в базе данных, например, проверка правил целостности при вставке или обновлении данных

* ограничения уникальности

определяют, что значения в определенном столбце или комбинации столбцов должны быть уникальными в пределах таблицы

* ограничения внешнего ключа

связывают две таблицы по значениям столбцов, обеспечивая целостность ссылочной целевой таблицы

1. Зачем нужны ограничения?

Ограничения необходимы, чтобы обеспечить целостность и корректность данных в базе данных. Они помогают предотвратить вставку некорректных данных, обеспечить уникальность значений и обеспечить целостность связей между таблицами.

1. Можно ли обойтись без ограничений в БД?

Невозможно, так как они являются неотъемлемой частью обеспечения целостности данных и поддержки правил базы данных

1. Существуют ли какие-либо ограничения в таблице по умолчанию?

В таблице по умолчанию может быть только ограничение первичного ключа (PRIMARY KEY), что означает, что столбец или комбинация столбцов должны быть уникальными и не могут содержать значения NULL.

1. Какие условия необходимо соблюсти, чтобы связать две таблицы связью

Две таблицы могут быть связаны связью, если определенный столбец в одной таблице соответствует столбцу с тем же типом данных в другой таблице. Обычно это реализуется через ограничение внешнего ключа.

1. Какие типы ключей вы знаете?

* Первичный ключ (PRIMARY KEY)

идентификатор для каждой записи в таблице

* Внешний ключ (FOREIGN KEY)

ссылка на первичный ключ в другой таблице

* Уникальный ключ (UNIQUE KEY)

уникальное значение, которое не является первичным ключом

* Внешний ключ составной (COMPOSITE KEY)

комбинация нескольких столбцов, которая служит внешним ключом

1. При связи двух таблиц, накладываются ли какие-либо ограничения на дочернюю или родительскую таблицу?

При связи двух таблиц накладываются определенные ограничения на дочернюю и родительскую таблицы. Дочерняя таблица должна иметь внешний ключ, который ссылается на первичный ключ родительской таблицы. Внешний ключ также может быть с обязательными или необязательными ограничениями на ссылочную целевую таблицу. Если применяется ограничение "ON DELETE CASCADE", то при удалении записи в родительской таблице также будут удалены соответствующие записи из дочерней таблицы.

1. Зачем нужны ключевые слова on update on delete?

ON UPDATE и ON DELETE используются для определения действий, которые будут выполнены при обновлении или удалении записи в родительской таблице, которая связана с дочерней таблицей через внешний ключ.

1. Какие параметры on update on delete существуют?

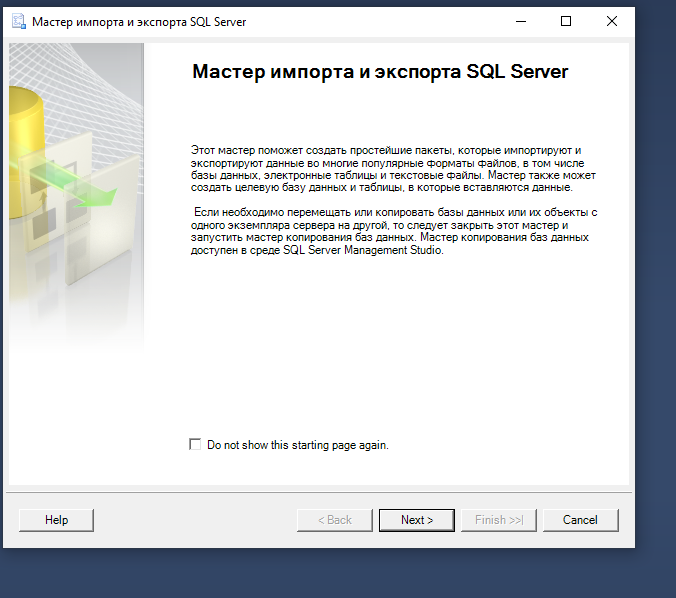
Для ON UPDATE и ON DELETE доступны следующие параметры: - CASCADE: обновление или удаление записи в родительской таблице приведет к обновлению или удалению соответствующих записей в дочерней таблице. - SET NULL: обновление или удаление записи в родительской таблице приведет к установке значения NULL в столбце внешнего ключа в дочерней таблице. - SET DEFAULT: обновление или удаление записи в родительской таблице приведет к установке значения по умолчанию в столбце внешнего ключа в дочерней таблице. - NO ACTION: запрещает обновление или удаление записи в родительской таблице, если на нее есть ссылки из дочерней таблицы. - RESTRICT: аналогично NO ACTION.

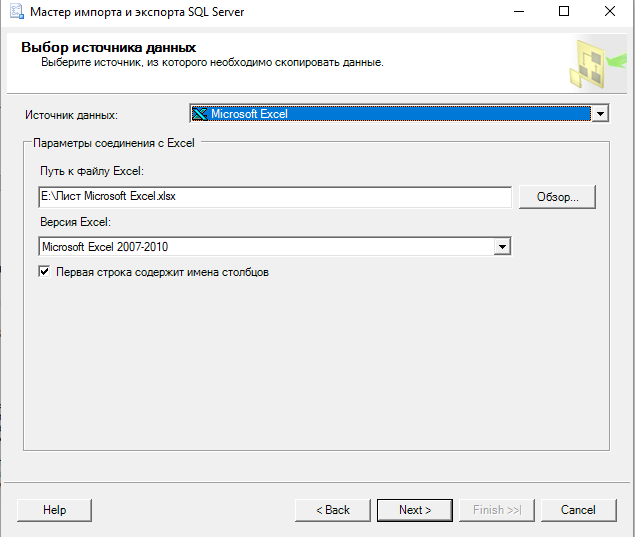
1. Какой синтаксис у ограничения типа CHECK?

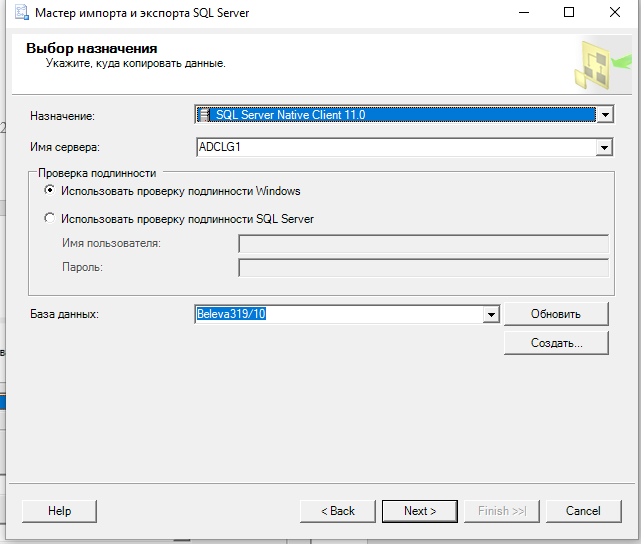
CHECK (условие), где условие - логическое выражение, которое должно быть выполнено для всех записей в таблице. Например, CHECK (возраст >= 18) ограничивает возраст записей в таблице, требуя, чтобы они были равны или больше 18.

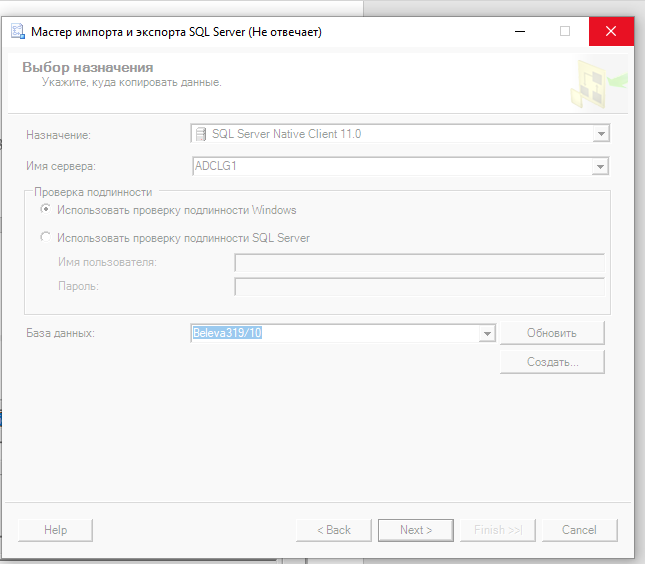
# **Лабораторная работа №6.**

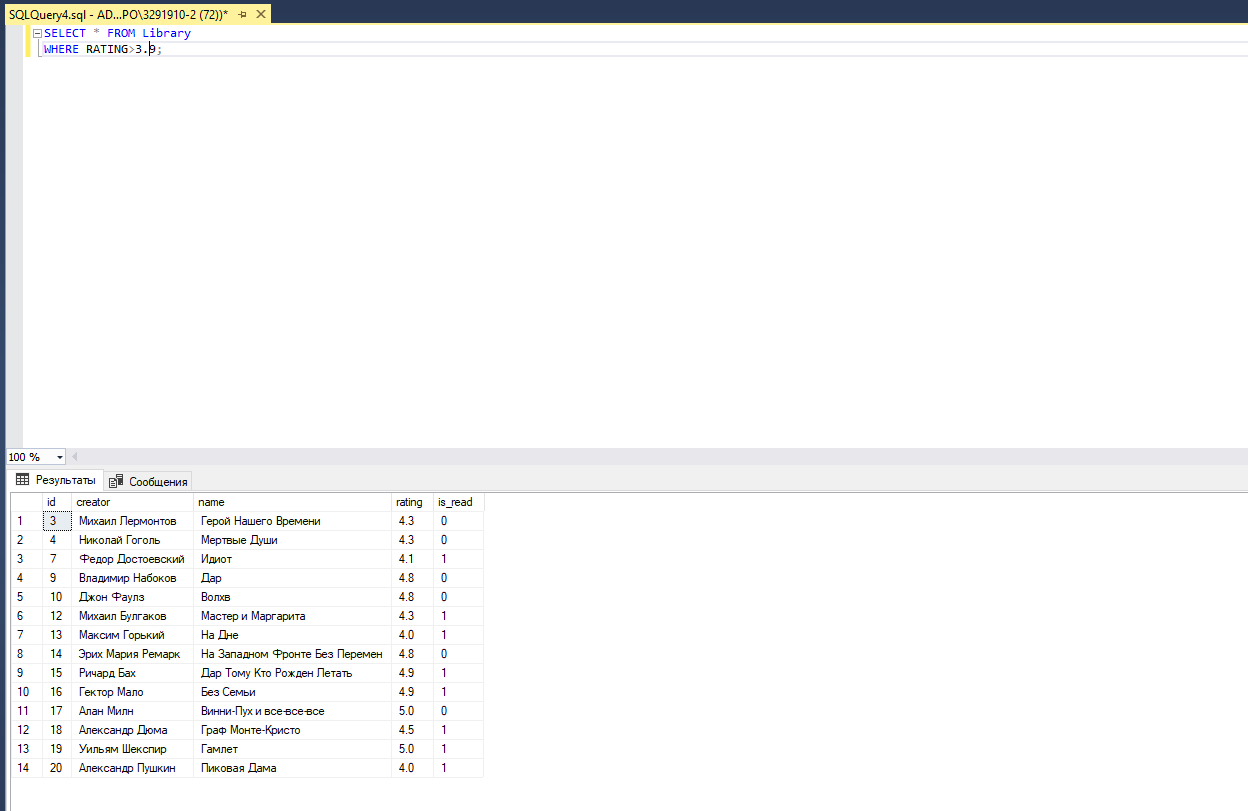
**Выборка данных. Выбор строк по условию**

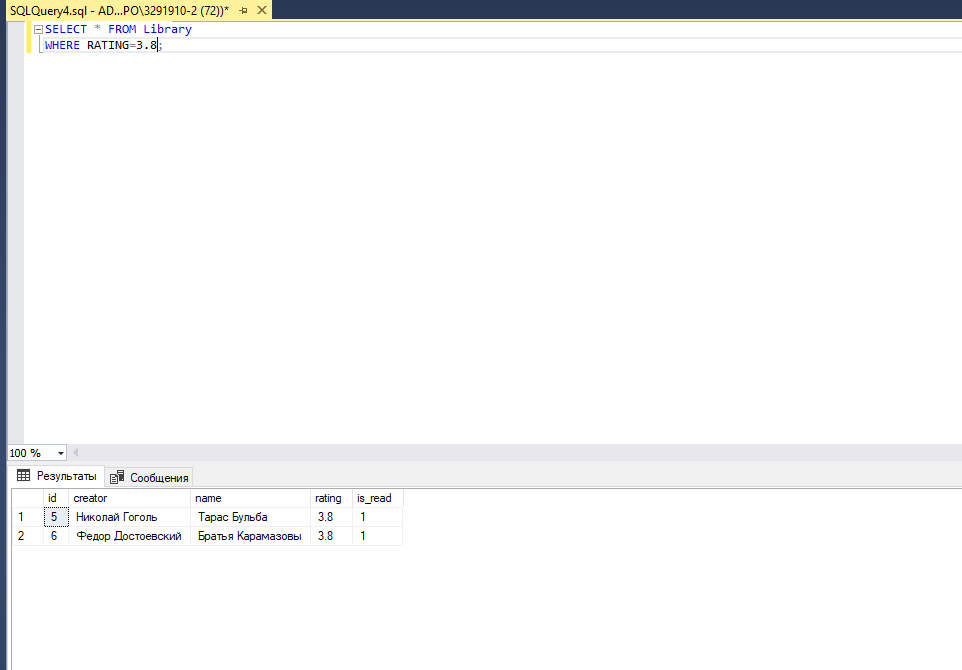
****

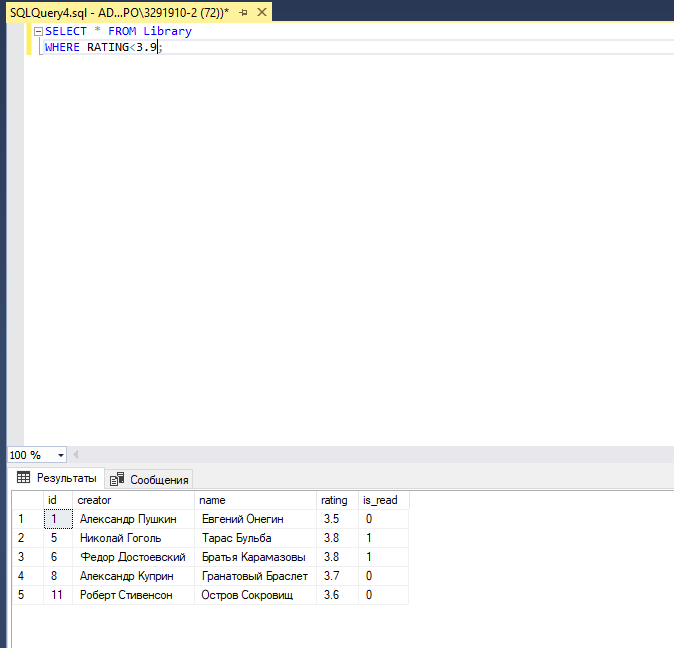
****

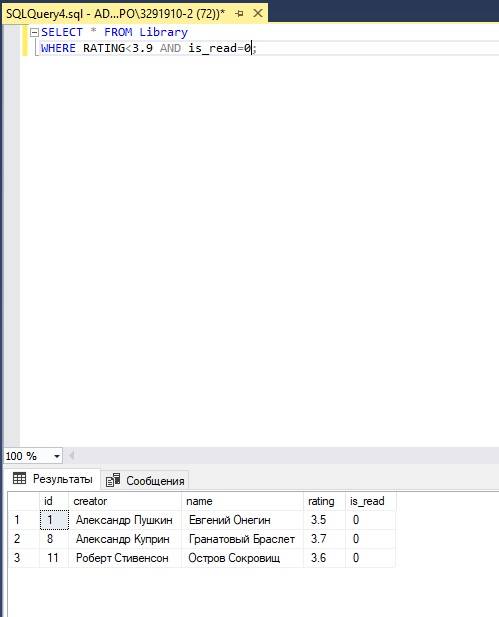
****

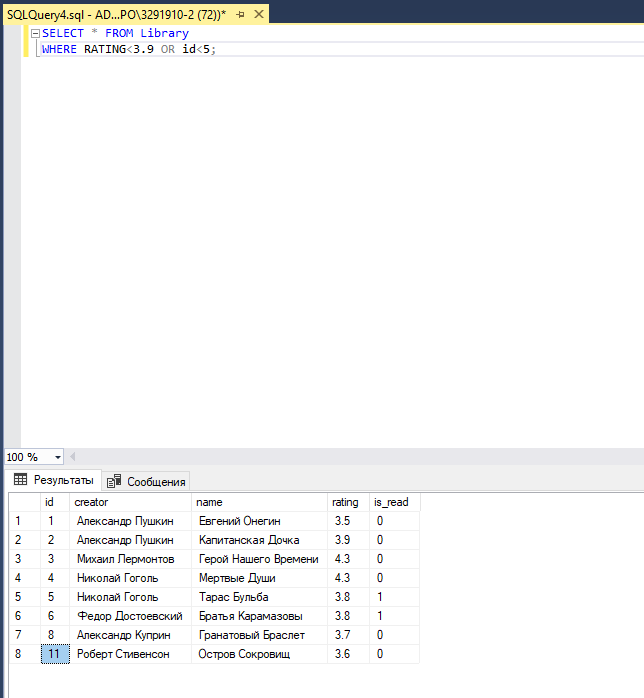
****

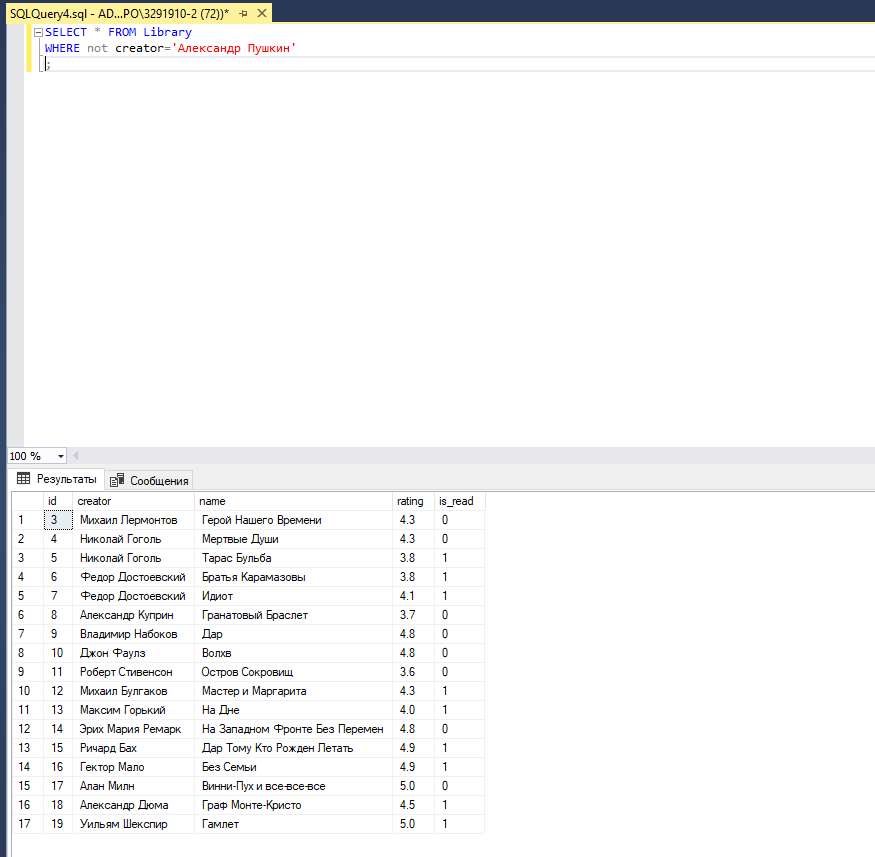
****

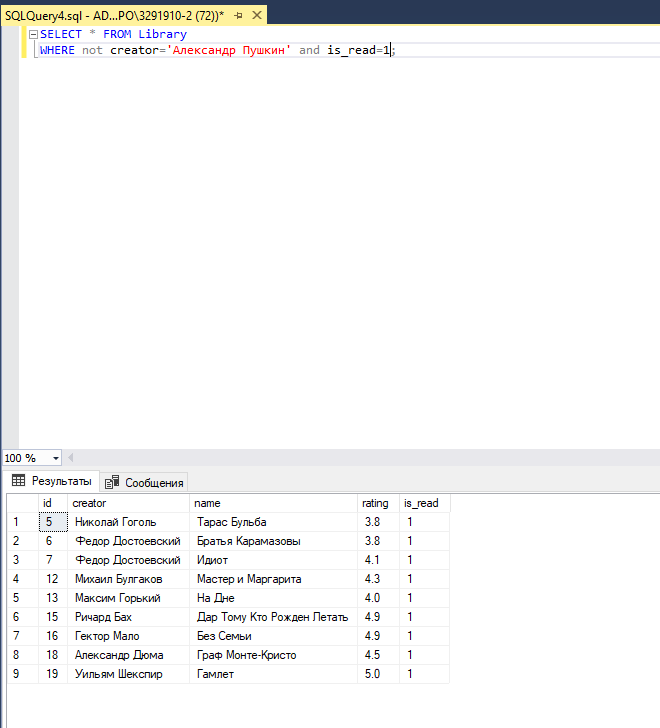
****

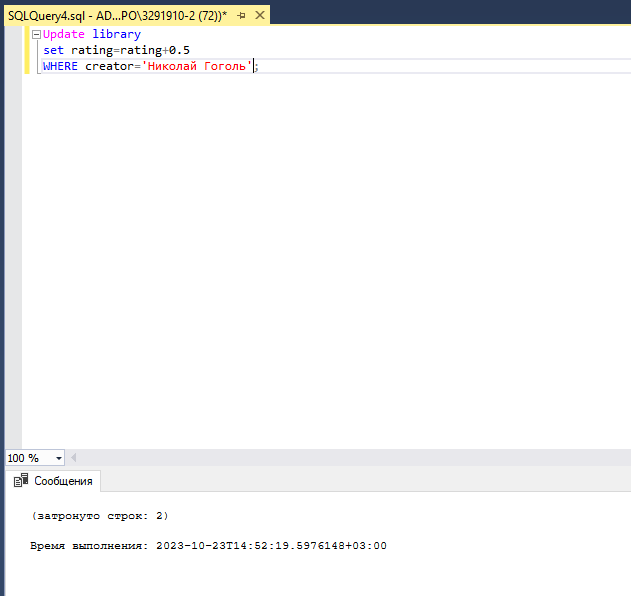
****

****

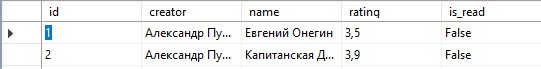
****

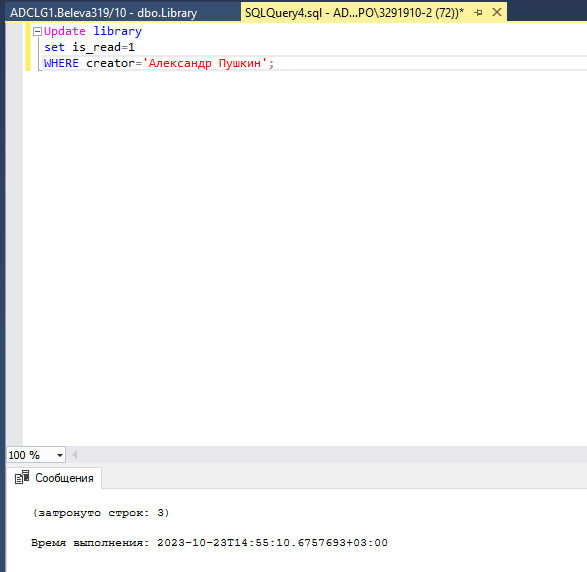
****

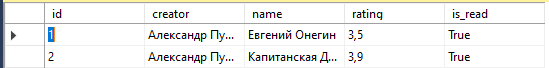
****

****

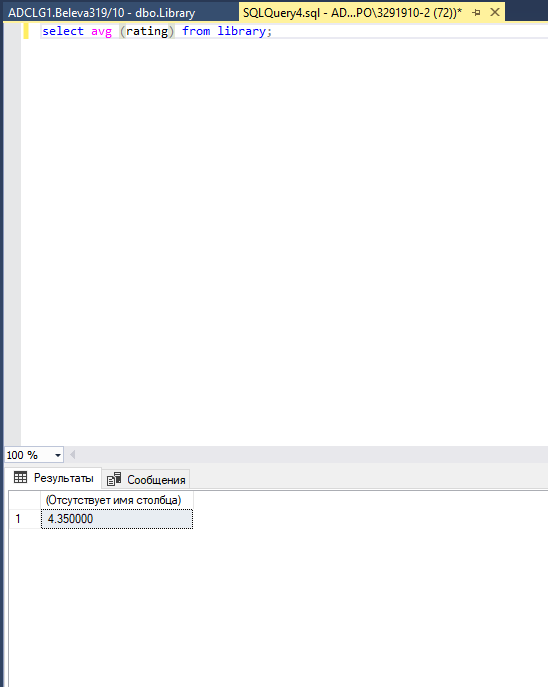
****

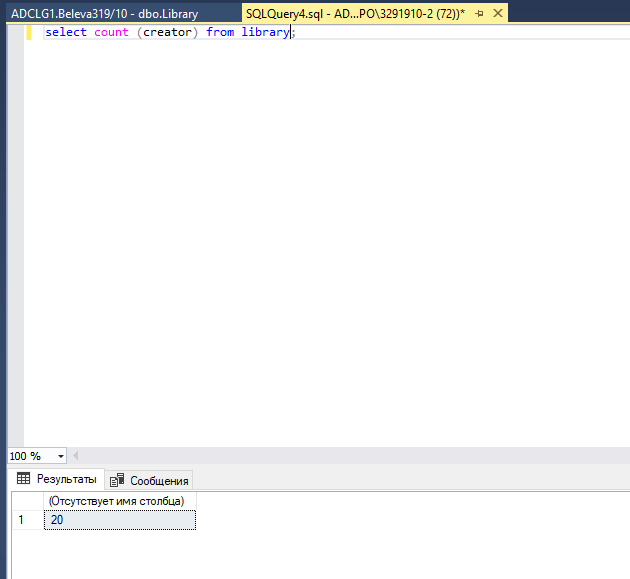
****

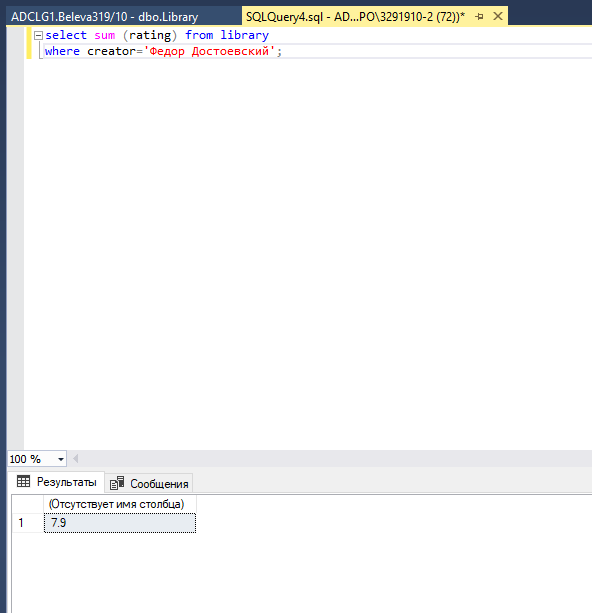
****

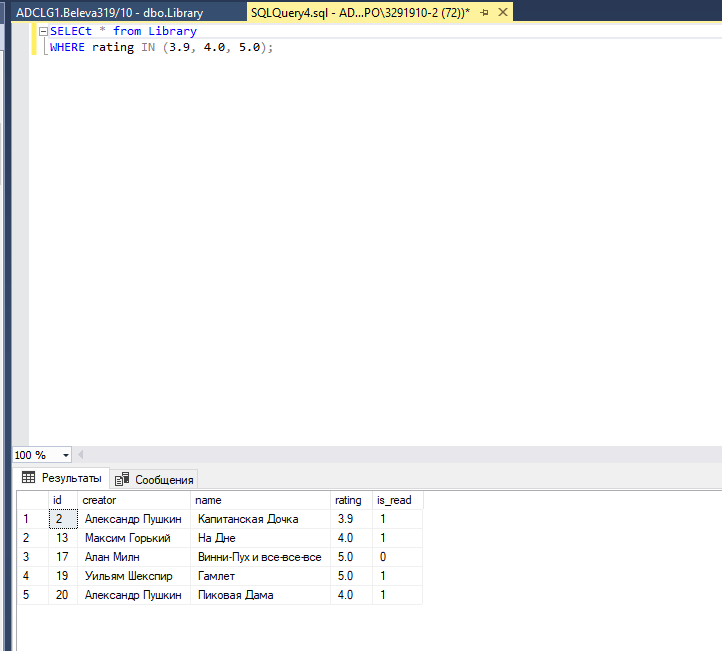
****

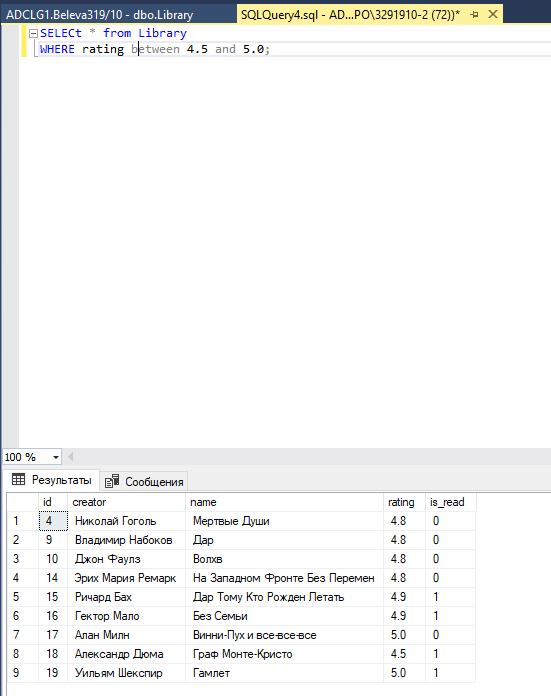
****

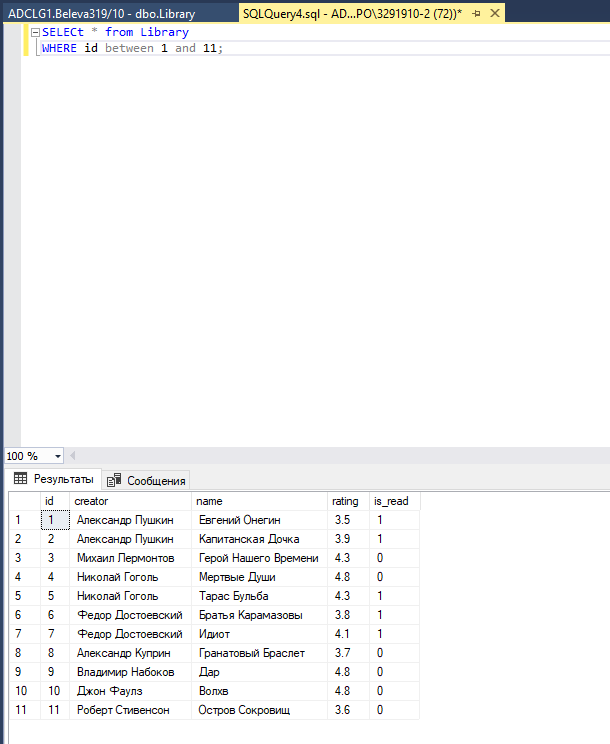
****

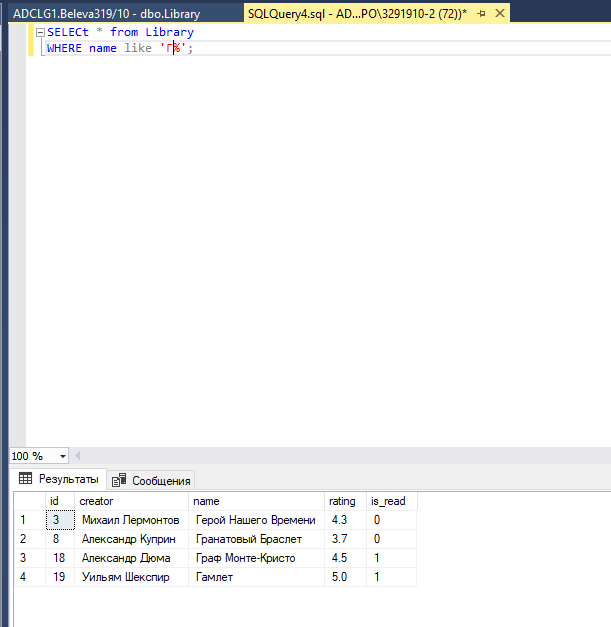
****

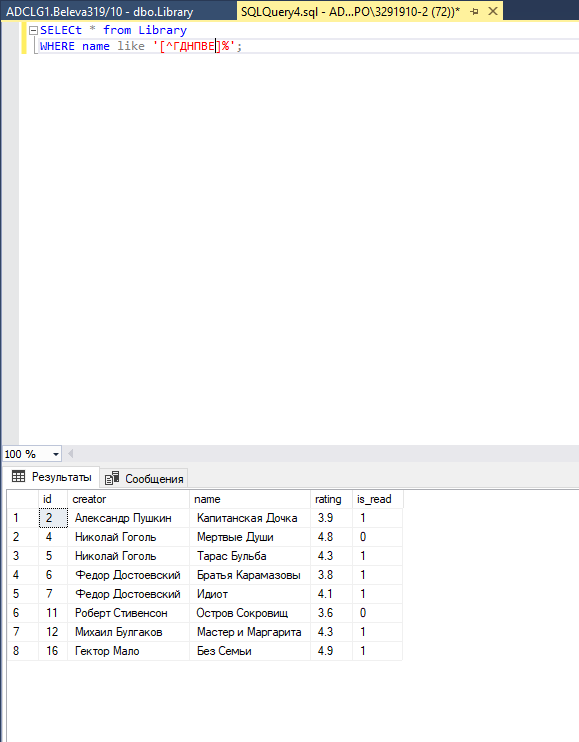
****

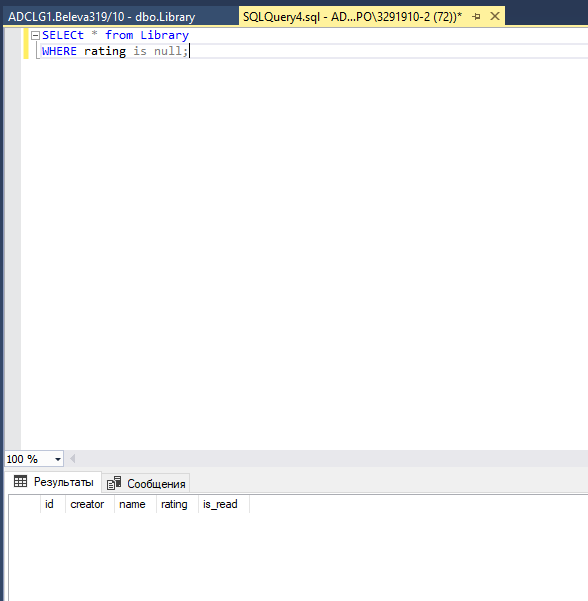
****

****

****

****

****

****

# **Лабораторная работа №7.**

**Выборка данных. Многотабличные запросы, запросы на соединения.**

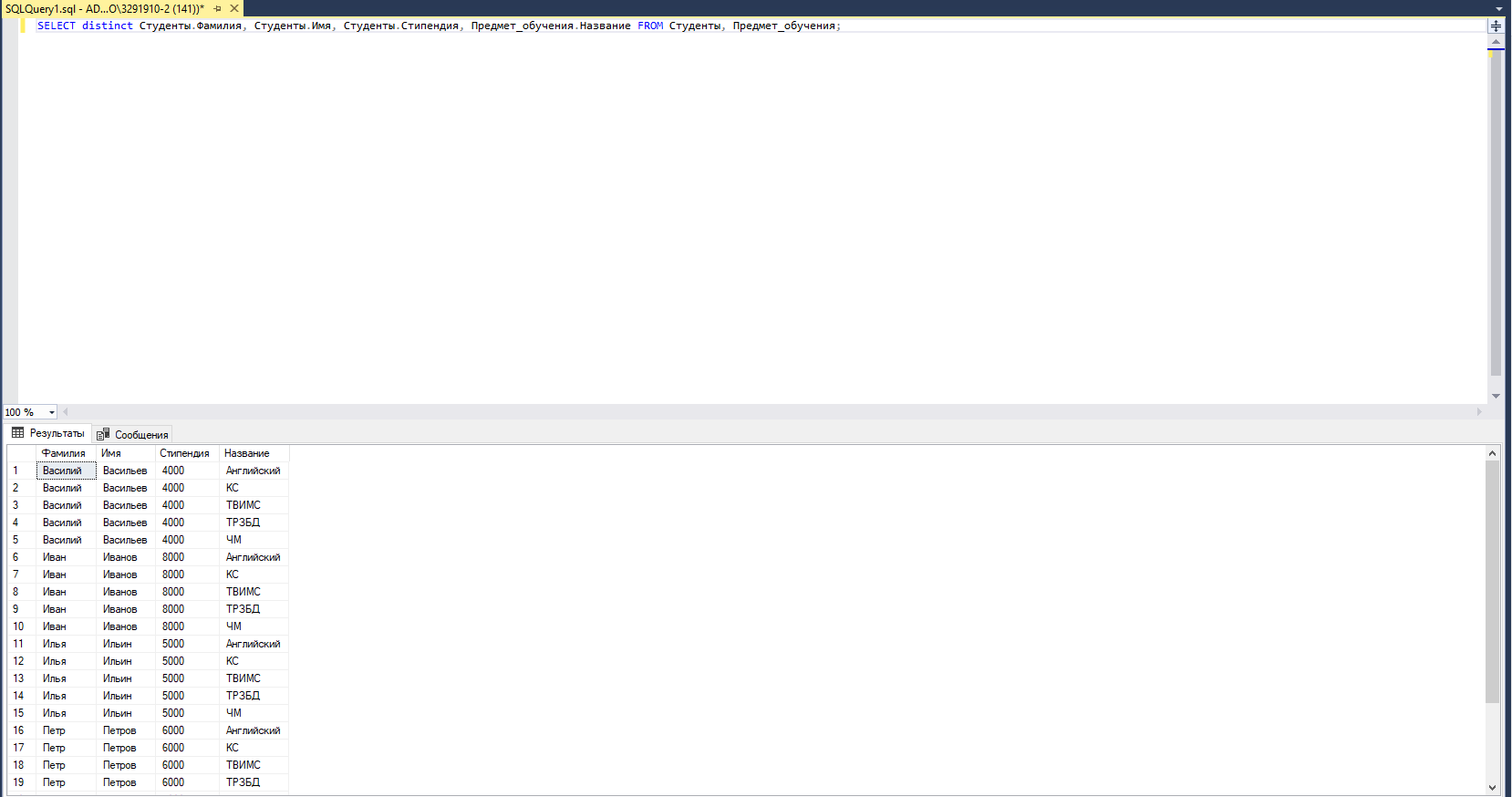
Задания на лабораторную работу.

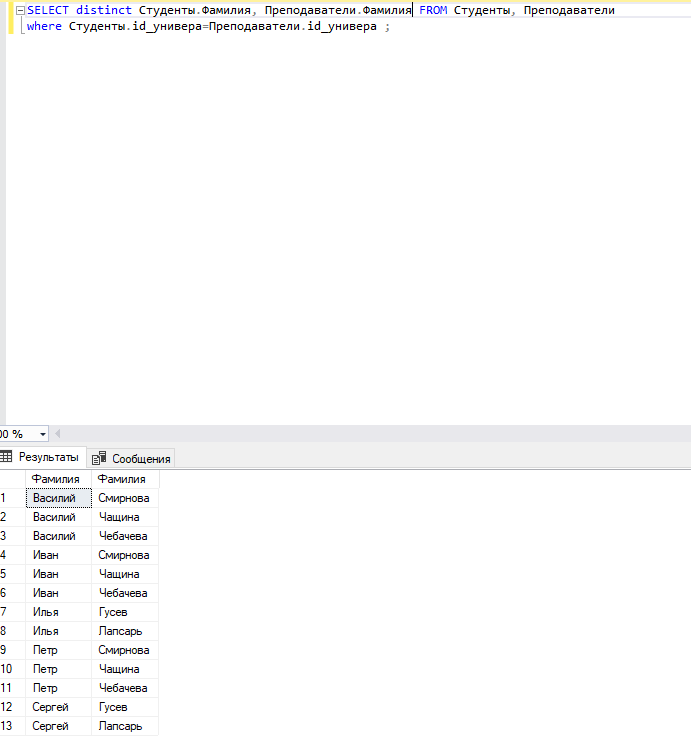
Используйте БД, которая была создана в лабораторной работе №3.

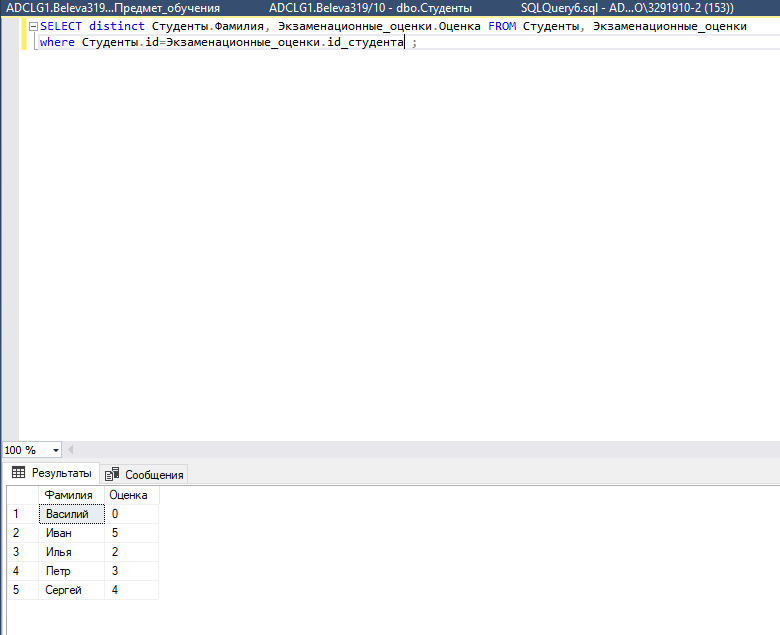
Самостоятельно создать следующие запросы:

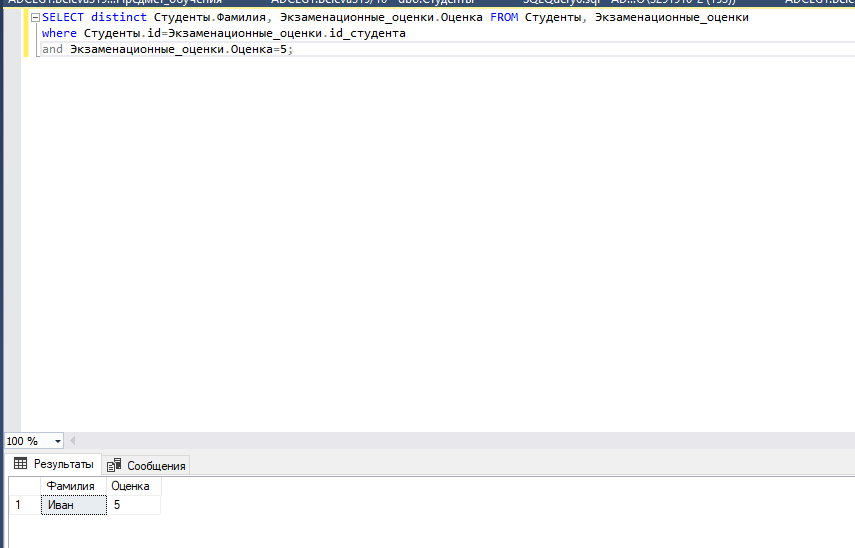
1. один запрос с использованием декартового произведения двух таблиц;
2. два запроса с использованием соединения двух таблиц по равенству;
3. один запрос с использованием соединения двух таблиц по равенству и условием отбора;
4. один запрос с использованием соединения по трем таблицам с условием;
5. создать копии ранее созданных запросов на соединение по равенству на запросы с использованием внешнего полного соединения таблиц (JOIN).
6. один запрос с использованием левого внешнего соединения;
7. один запрос на использование правого внешнего соединения;
8. один запрос с использованием внутреннего соединения.
9. один запрос на проверку какого-либо условия через самосоединение.

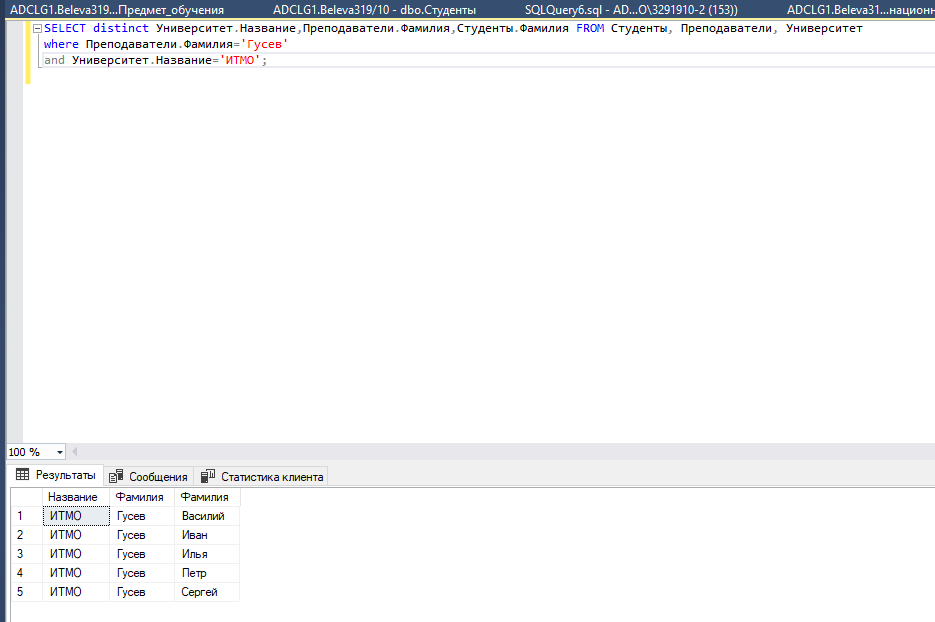
Отчет должен содержать код или скриншот запросов и результат их выполнения. На скриншоте результатов должно быть видно имя вашей БД.

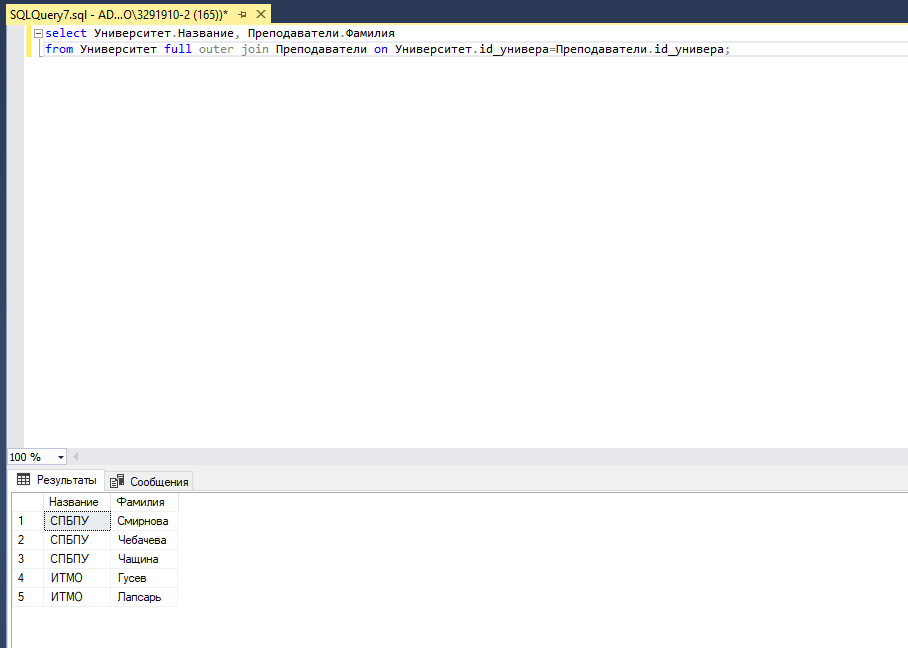
****

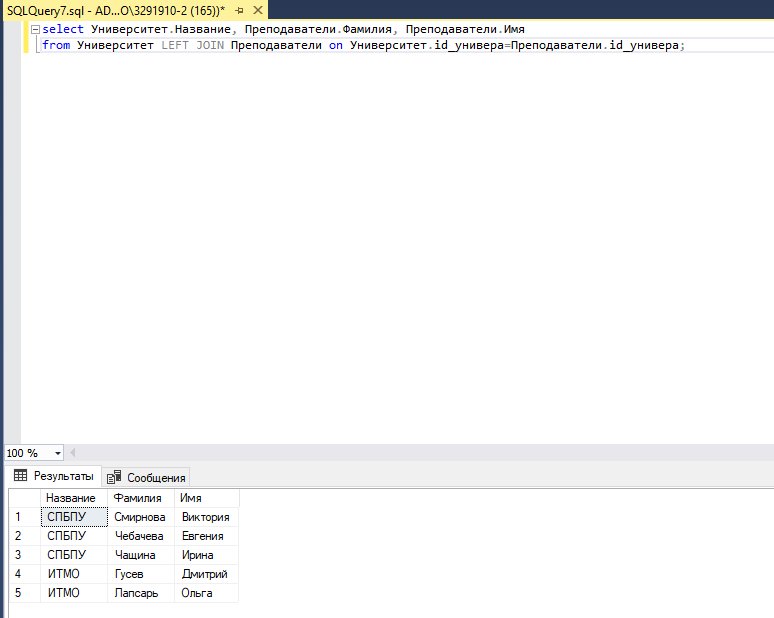
****

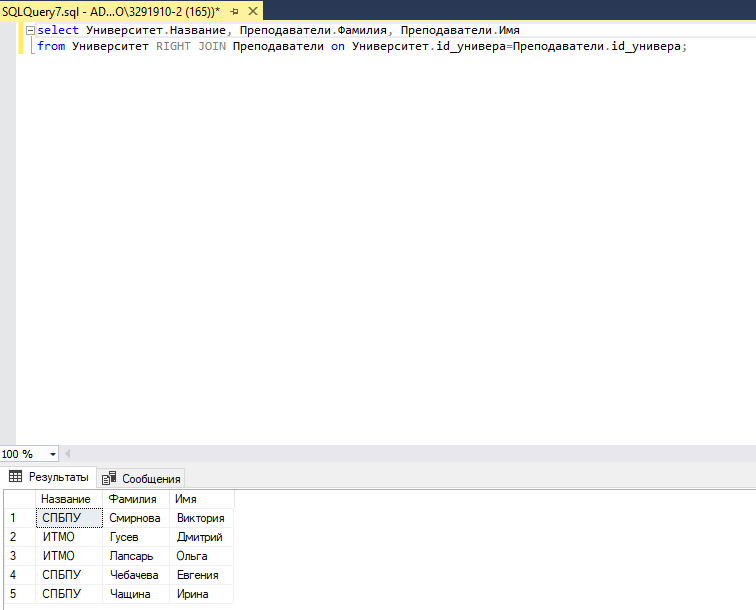
****

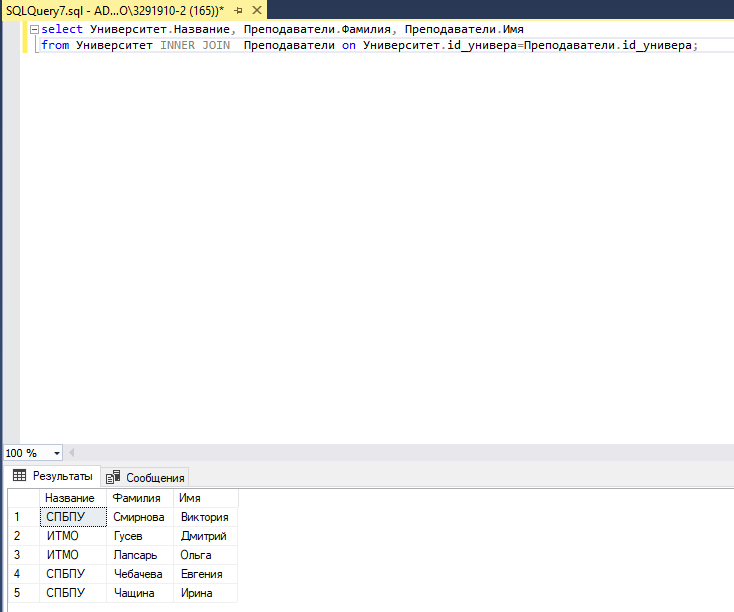
****

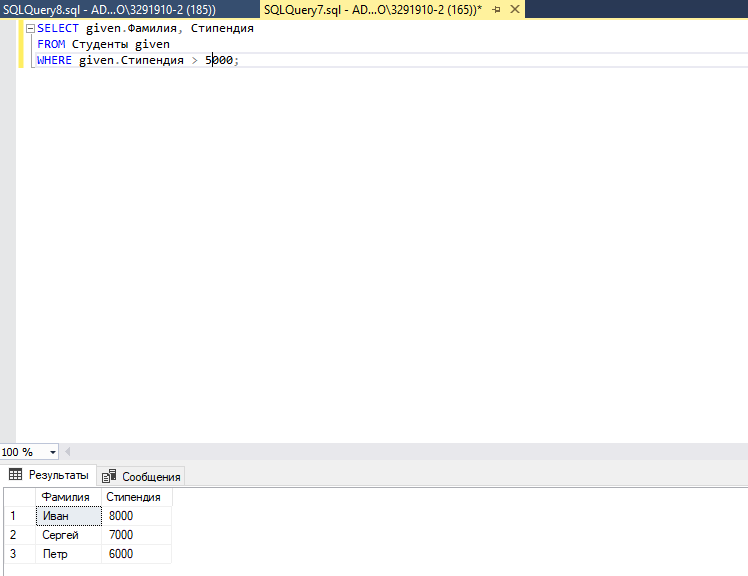
****

****

****

****

****

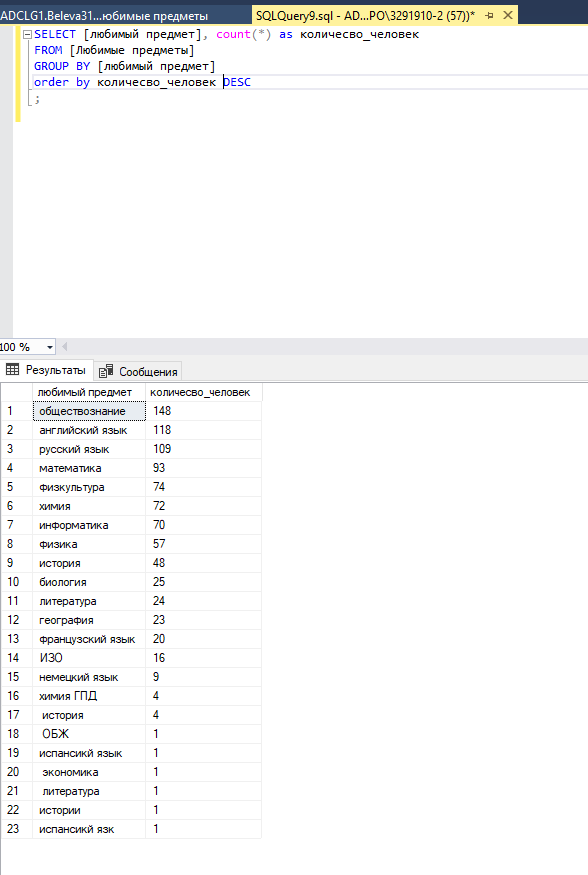
****

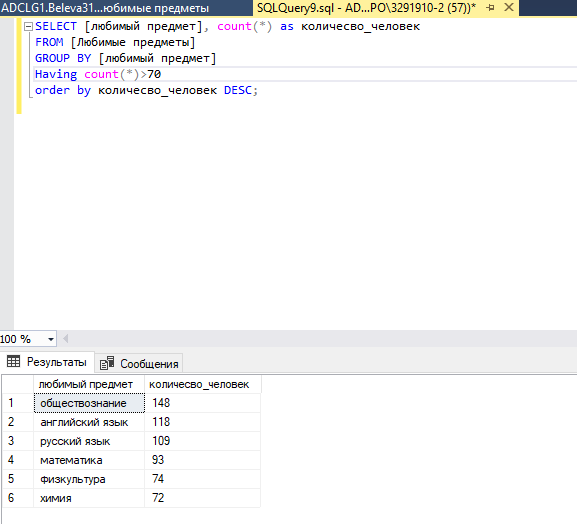
# **Лабораторная работа №8.**

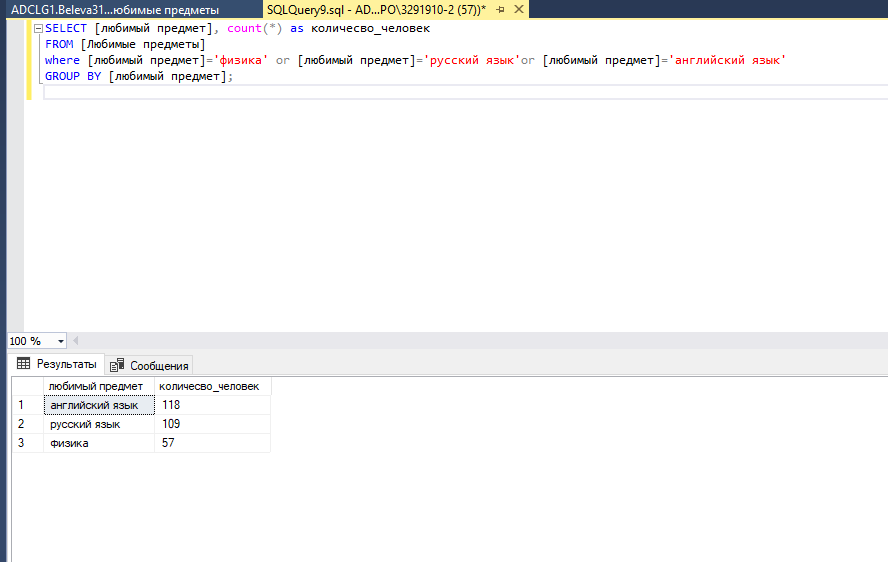
**Выборка данных. Группировка и сортировка данных.**

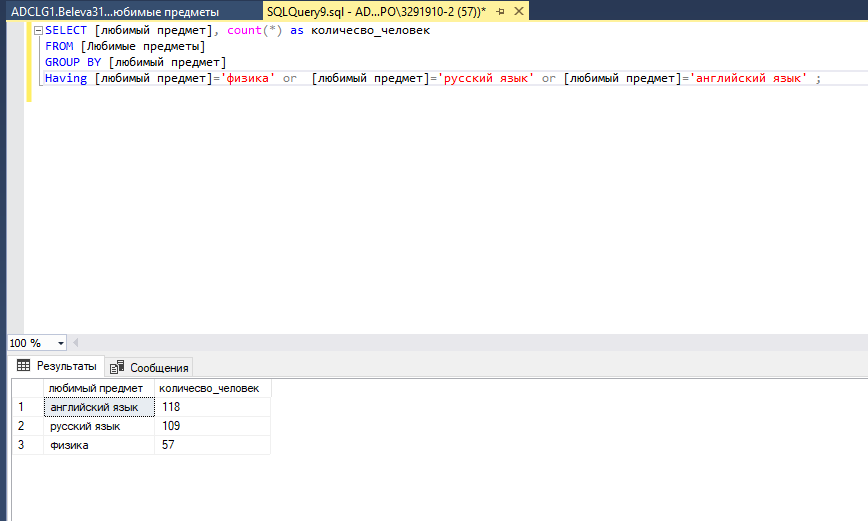
Для 2 варианта:

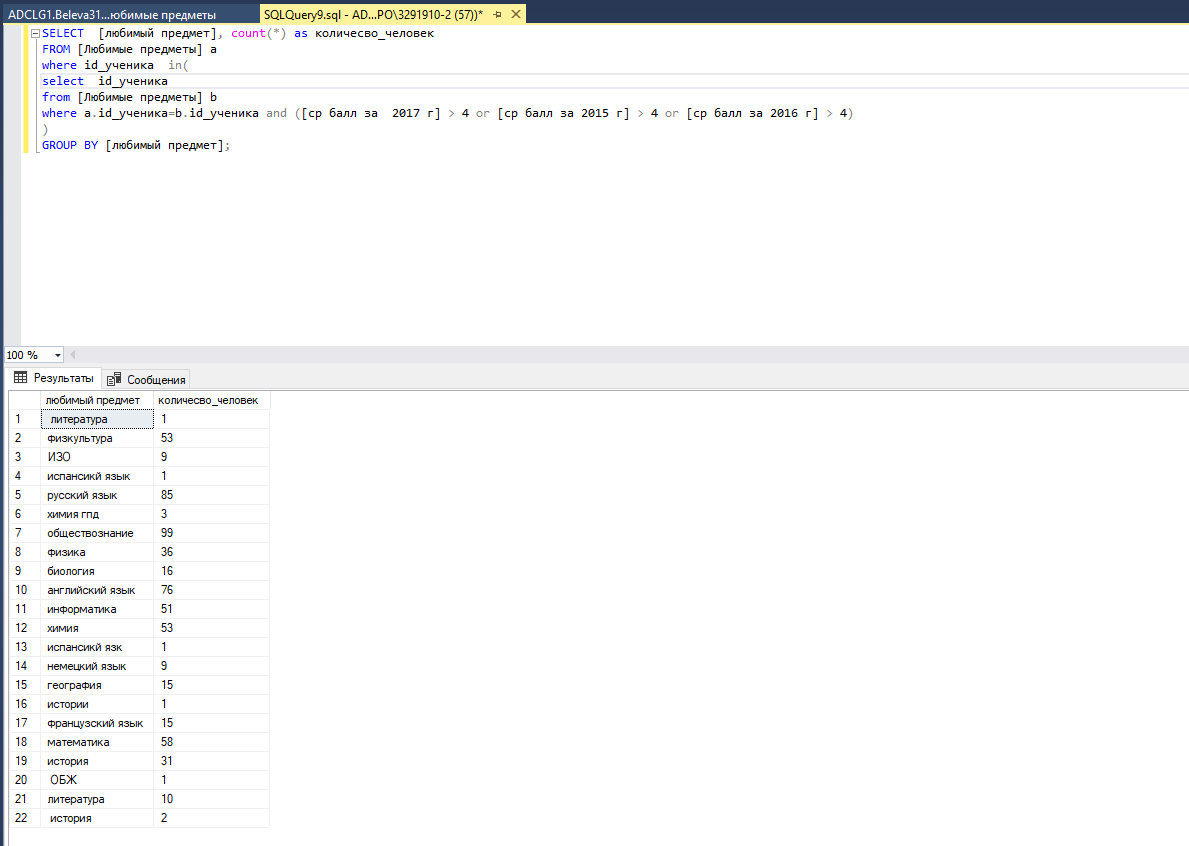
1. Запрос, который покажет, сколько человек выбрало тот или иной предмет в качестве любимого, сделайте сортировку по убыванию.
2. Запрос, который покажет, какие предметы в качестве любимого выбрало более 70 человек.
3. Запрос, который покажет сколько студентов выбрало в качестве любимого физику, русский или английский. Создайте два запроса один с помощью WHERE, другой используя HAVING.
4. Запрос, который покажет, сколько студентов выбрало тот или иной предмет в качестве любимого и имеющих средний балл хотя бы за один год более 4.
5. Запрос, который покажет сколько студентов имею средний бал за 2015 или 2016 год менее 3.

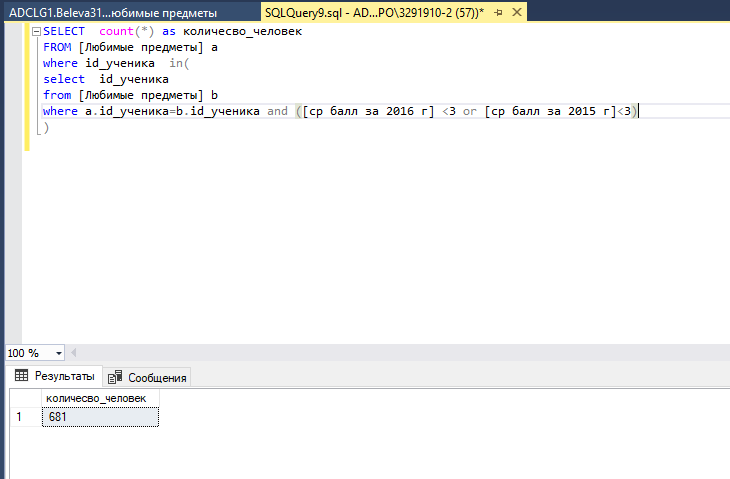
****

****

****

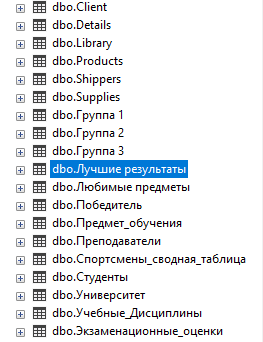
****

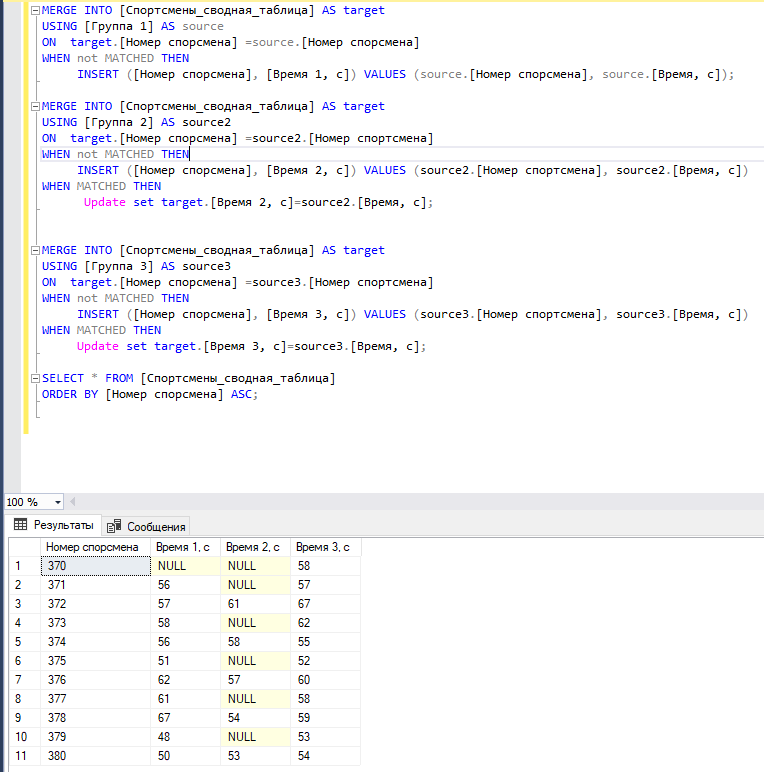
****

****

# **Лабораторная работа №9.**

**Инструкция MERGE**





MERGE INTO [Спортсмены\_сводная\_таблица] AS target

USING [Группа 1] AS source

ON target.[Номер спорсмена] =source.[Номер спорсмена]

WHEN not MATCHED THEN

INSERT ([Номер спорсмена], [Время 1, с]) VALUES (source.[Номер спорсмена], source.[Время, с]);

MERGE INTO [Спортсмены\_сводная\_таблица] AS target

USING [Группа 2] AS source2

ON target.[Номер спорсмена] =source2.[Номер спортсмена]

WHEN not MATCHED THEN

INSERT ([Номер спорсмена], [Время 2, с]) VALUES (source2.[Номер спортсмена], source2.[Время, с])

WHEN MATCHED THEN

Update set target.[Время 2, с]=source2.[Время, с];

MERGE INTO [Спортсмены\_сводная\_таблица] AS target

USING [Группа 3] AS source3

ON target.[Номер спорсмена] =source3.[Номер спортсмена]

WHEN not MATCHED THEN

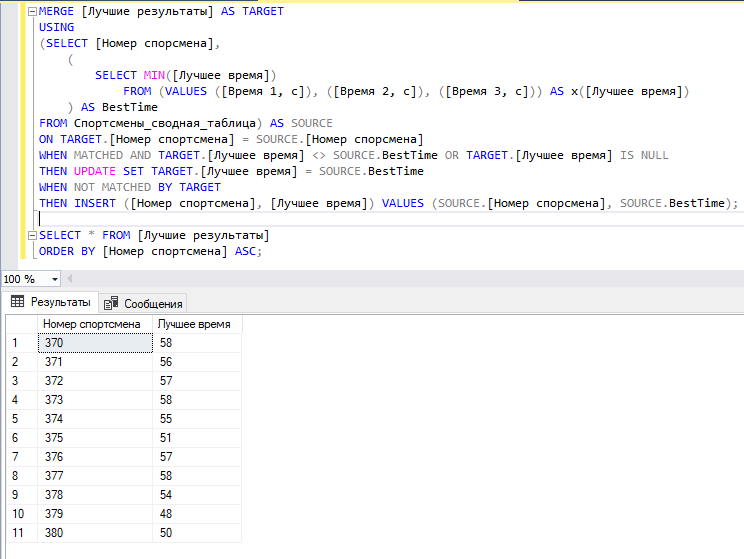
INSERT ([Номер спорсмена], [Время 3, с]) VALUES (source3.[Номер спортсмена], source3.[Время, с])

WHEN MATCHED THEN

Update set target.[Время 3, с]=source3.[Время, с];

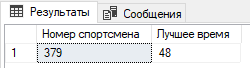
SELECT \* FROM [Спортсмены\_сводная\_таблица]

ORDER BY [Номер спорсмена] ASC;



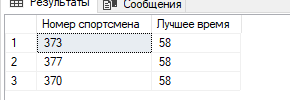
SELECT \* FROM [Лучшие результаты]

WHERE [Лучшее время]= (Select MIN([Лучшее время]) From [Лучшие результаты]);



SELECT \* FROM [Лучшие результаты]

WHERE [Лучшее время]= (Select MAX([Лучшее время]) From [Лучшие результаты]);



# **Лабораторная работа №10.**

**Функции**

Задания на лабораторную работу.

1. Создайте функцию, выполняющую роль калькулятора, со следующими возможностями: сложение, вычитание, умножение, деление, sin, cos, tg, ctg, возведение в заданную степень.

2. Протестируйте функцию.

3. Создайте функцию, которая при вводе даты, будет возвращать название сезона (Зима, Весна, Лето, Осень).

4. Протестируйте функцию.

5. Создайте функцию, которая будет разбивать входную строку на отдельные строки. Символ пробела является разделителем строк.

6. Протестируйте функцию.

7. Импортируйте документ Doc.xsl.

8. Работа с таблицей StudList. Создайте функцию, возвращающую список студентов, обучающихся или отчисленных, в зависимости от входного параметра. Предполагается, что, если в столбце Studying стоит No, студент отчислен, в противном случае – обучается.

9. Протестируйте функцию.

Задание 1.

create FUNCTION [dbo].[calculator5] (@operation varchar(10), @a float, @b float)

RETURNS float

AS

BEGIN

DECLARE @c float

if (@operation='+')

set @c=@a+@b;

else if (@operation='-')

set @c=@a-@b;

else if (@operation='\*')

set @c=@a\*@b;

else if (@operation='/')

set @c=@a/@b;

else if (@operation='sin')

set @c=Sin(@a);

else if (@operation='cos')

set @c=Cos(@a);

else if (@operation='tan')

set @c=Tan(@a);

else if (@operation='ctg')

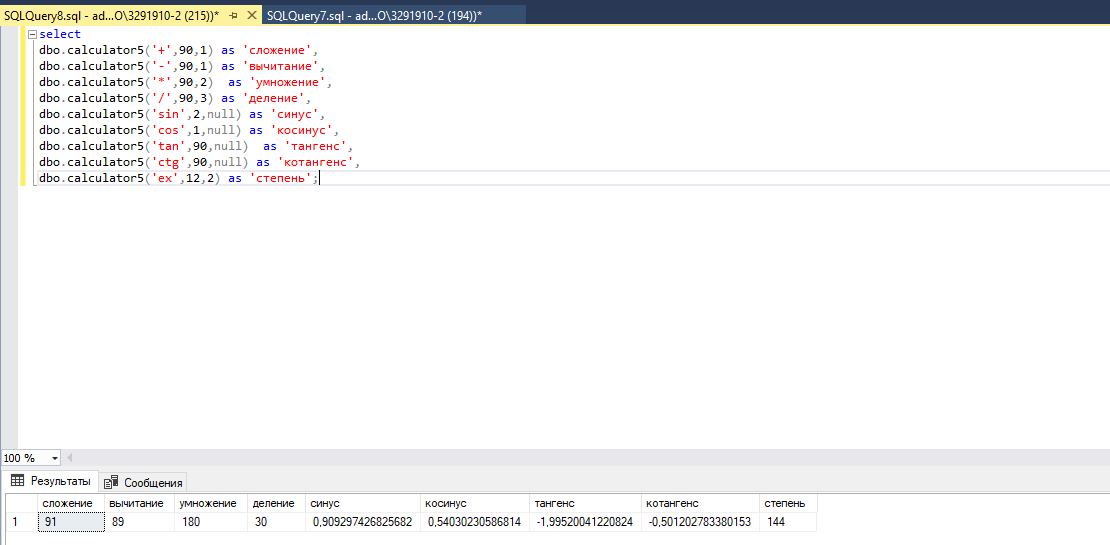
set @c=1.0/(Tan(@a));

else if (@operation='ex')

set @c=Power(@a,@b);

return @c;

END;



Задание 2.

CREATE FUNCTION [dbo].[ISOweek] (@DATE date)

RETURNS CHAR(15)

WITH EXECUTE AS CALLER

AS

BEGIN

DECLARE @man int;

DECLARE @ISOweek char(15);

SET @man= MONTH(@DATE)

IF (@man=1 or @man=12 or @man=2) SET @ISOweek='Зима';

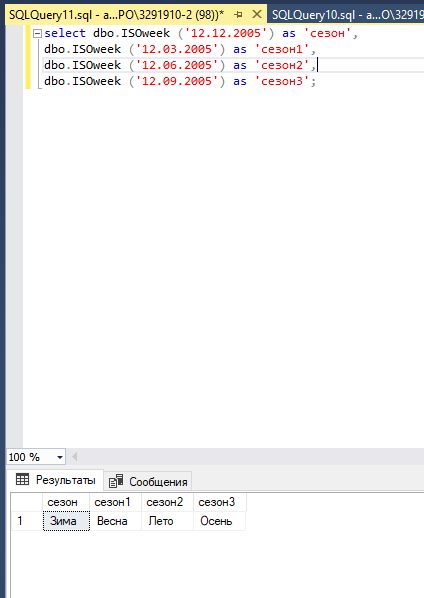
IF (@man=3 or @man=4 or @man=5) SET @ISOweek='Весна';

IF (@man=8 or @man=6 or @man=7 ) SET @ISOweek='Лето';

IF (@man=11 or @man=9 or @man=10 ) SET @ISOweek='Осень';

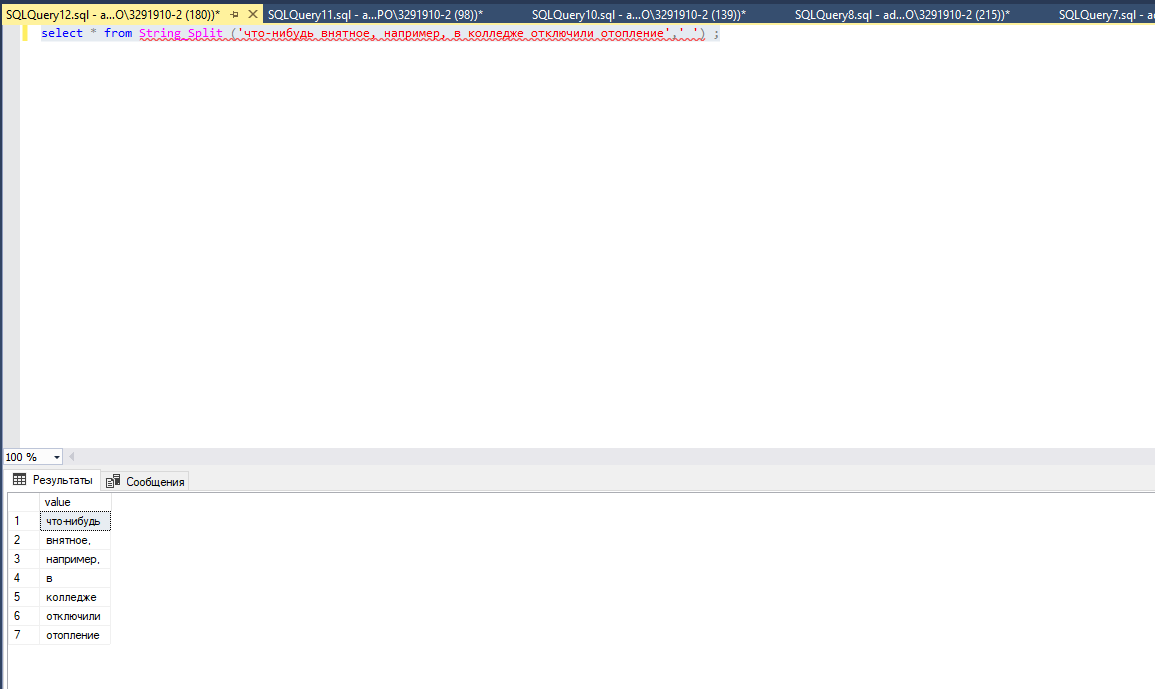
RETURN(@ISOweek);

END;



Задание 3.

select \* from String\_Split ('что-нибудь внятное, например, в колледже отключили отопление',' ') ;



Задание 4.

create function [dbo].[F\_student] (@iS\_IsNot\_St varchar(20))

returns table

as

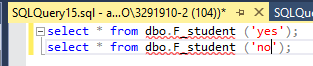
return

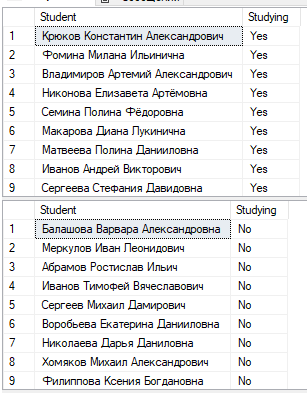
(

Select \* from dbo.study\_list

where Studying=@iS\_IsNot\_St

)





# **Лабораторная работа №11.**

**Процедуры**

Задания на лабораторную работу.

1. Работа с таблицей StudList из предыдущей ЛР. Создать процедуру с входным параметром и return. Процедура должна получать входной параметр – Имя студента, а на выходе выдать 1 или 0, в зависимости от того обучается он или нет.

2. Проверить процедуру.

3. Работа с таблицей Shop. Импортируйте таблицу Shop. В магазине проходит акция. Для всех, кто оставил комментарий на сайте, скидка 5%. Создайте процедуру, которая уменьшит цены в таблице Shop на 5% (атрибут Price), в тех строках, где значение столбца Comment yes.

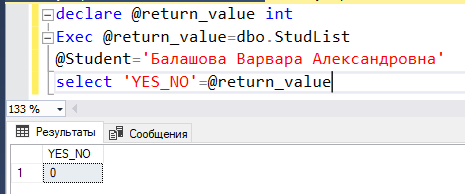
4. Проверить процедуру.

5. Работа с таблицей Shop. Создать процедуру с входным параметром. Входной параметр – номер клиента. Процедура удаляет все строки с заказами этого клиента.

6. Работа с таблицей Shop. Создать процедуру с входным и выходным параметрами, где входной параметр – номер клиента, а выходной – самый дорогой заказ выбранного клиента.

7. Проверить процедуру.

Задание 1



create procedure StudList @Student char(40)

As

DECLARE @Studying nchar(3)

set @Studying=(select Studying from study\_list where Student=@Student)

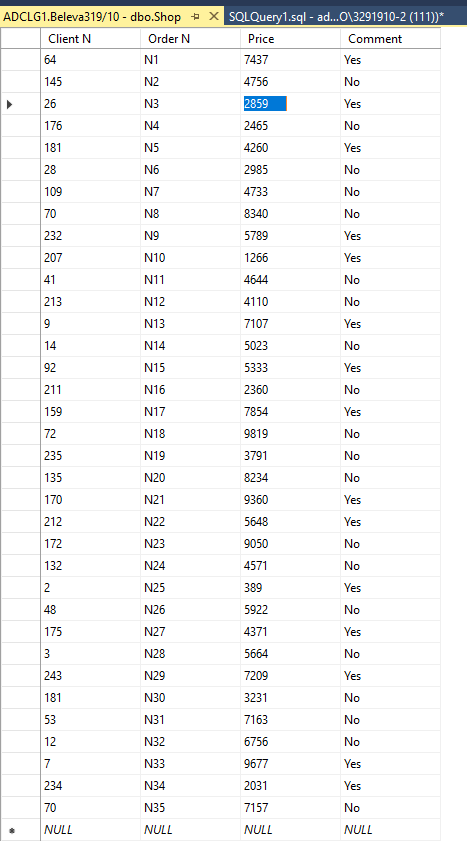
if @Studying = 'No'

return(0);

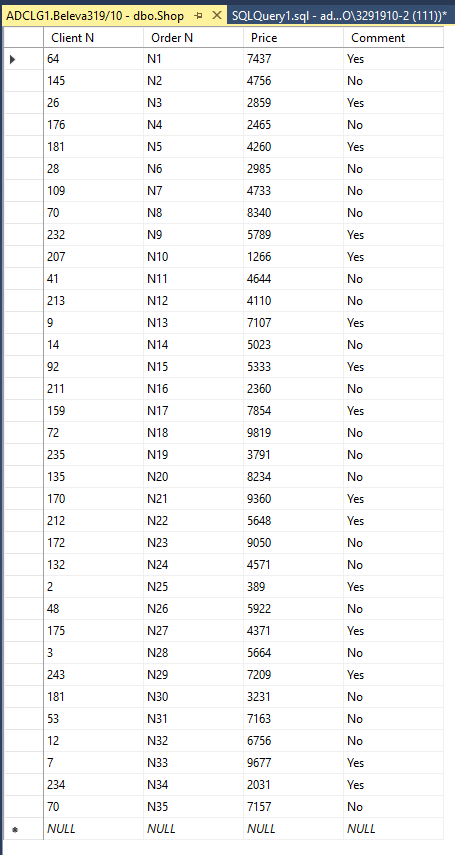
else if

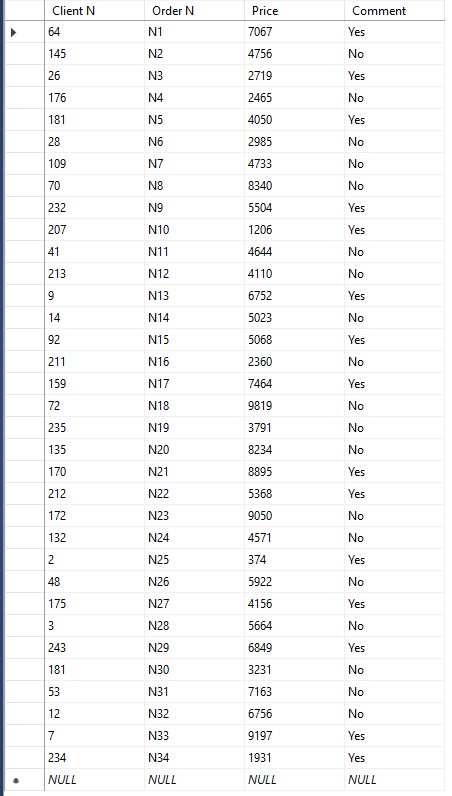
@Studying = 'Yes'

return (1);



Задание 3





create procedure ShopList

As

select Comment from Shop

update Shop

set Price=Price-((Price/100)\*5)

where Comment = 'Yes'

exec ShopList;

задание 5

create procedure DeleteOrder

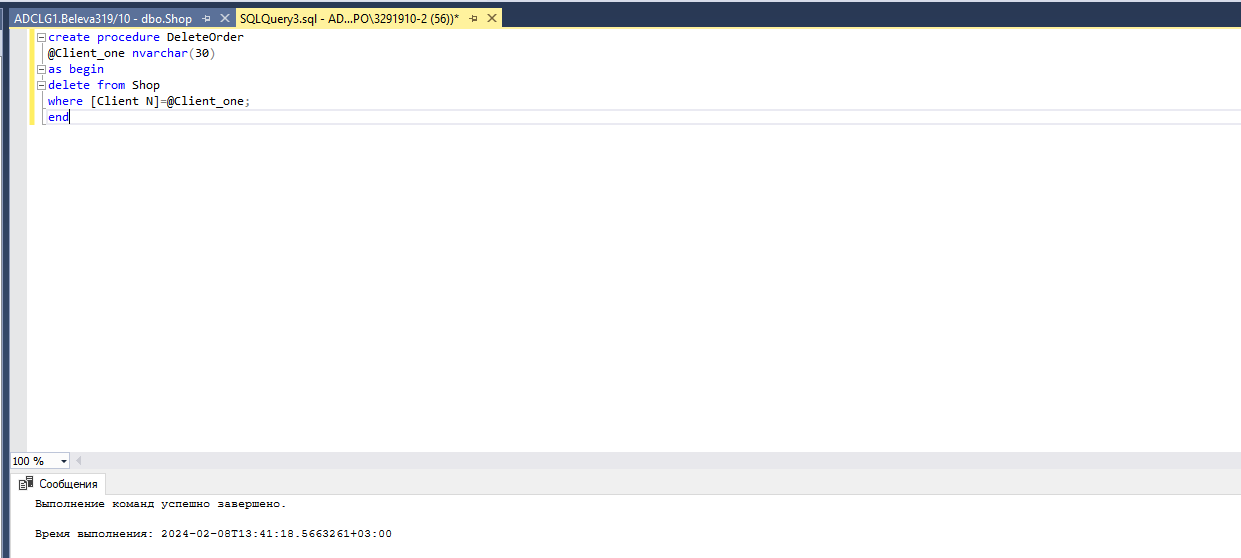
@Client\_one nvarchar(30)

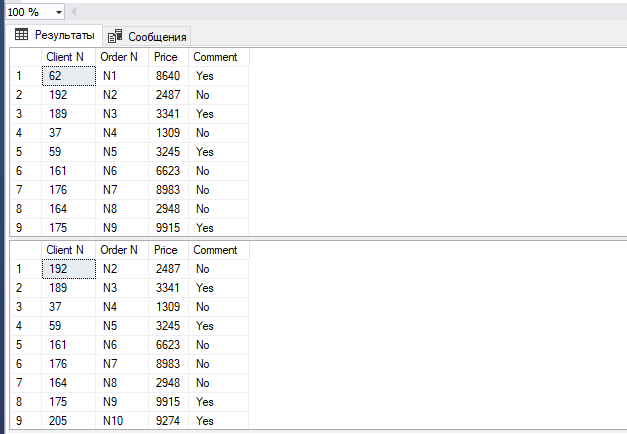
as begin

delete from Shop

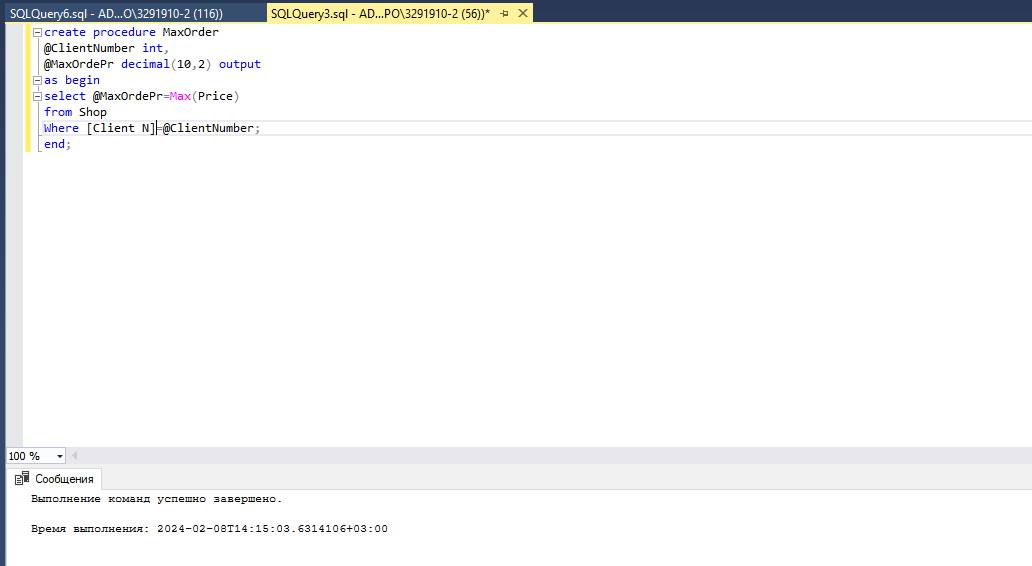
where [Client N]=@Client\_one;

end





Задание 7



create procedure MaxOrder

@ClientNumber int,

@MaxOrdePr decimal(10,2) output

as begin

select @MaxOrdePr=Max(Price)

from Shop

Where [Client N]=@ClientNumber;

end;



declare @MaxOrder decimal (10,2);

exec MaxOrder 202, @MaxOrder output;

select @MaxOrder as 'самое дорогое';



Контрольные вопросы:   
1. Какие типы процедур вы знаете?   
Существуют различные типы процедур, включая хранимые процедуры, пользовательские процедуры и системные процедуры.   
2. Где хранятся процедуры?   
Процедуры могут храниться в различных местах, в зависимости от конкретной среды или языка программирования. В случае баз данных, хранимые процедуры обычно хранятся в базе данных, а для обычных процедур может использоваться файловая система или другое хранилище.   
3. Какой синтаксис у создания процедур?   
Синтаксис создания процедур зависит от языка программирования или среды разработки.   
CREATE PROCEDURE procedure\_name   
parameter1 datatype,   
parameter2 datatype   
AS   
BEGIN   
-- тело процедуры   
END   
4. Какой синтаксис у выполнения процедур?   
Синтаксис выполнения процедуры также зависит от языка программирования или среды разработки. В SQL вызов процедуры может выглядеть следующим образом: EXEC procedure\_name param1, param2   
5. В чем разница между функцией и процедурой?   
Функция возвращает результат, а процедура выполняет определенные действия. Функция может быть использована для вычисления и возврата значения, тогда как процедура может выполнять различные операции без необходимости возвращать результат.   
6. Зачем нужны процедуры?   
Процедуры могут использоваться для организации и структурирования кода, повторного использования фрагментов кода, улучшения производительности, уменьшения дублирования кода, а также для реализации сложных бизнес-логик и управления данными в базах данных.

# **Лабораторная работа №12.**

**Создание, программирование и управление триггерами**

Задания на лабораторную работу.

1.Импортируйте данные из файла AUTH.xls

2.Создайте триггер Trigger\_1, который после изменения данных будет выводить строки, со старыми и новыми значениями, используйте триггер AFTER UPDATE.

Протестируйте ваш триггер.

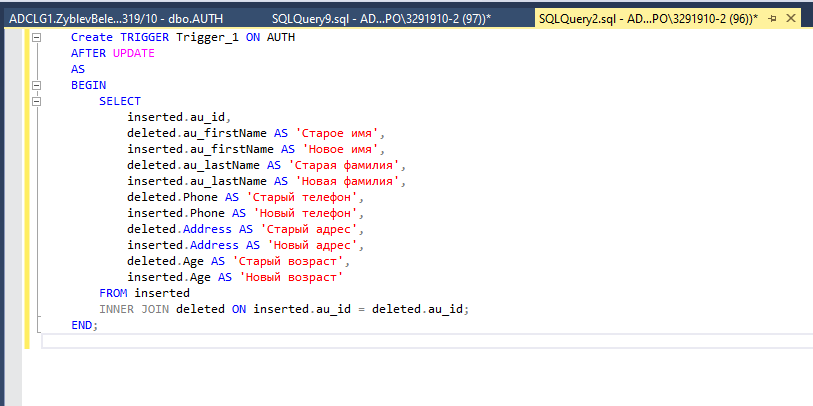
3.Создайте триггер Trigger\_2, который вместо удаления выведет следующую информацию: «попытка удаления информации об au\_id». Используйте триггер INSTEAD OF – триггеры вызывается вместо выполнения команд списка..

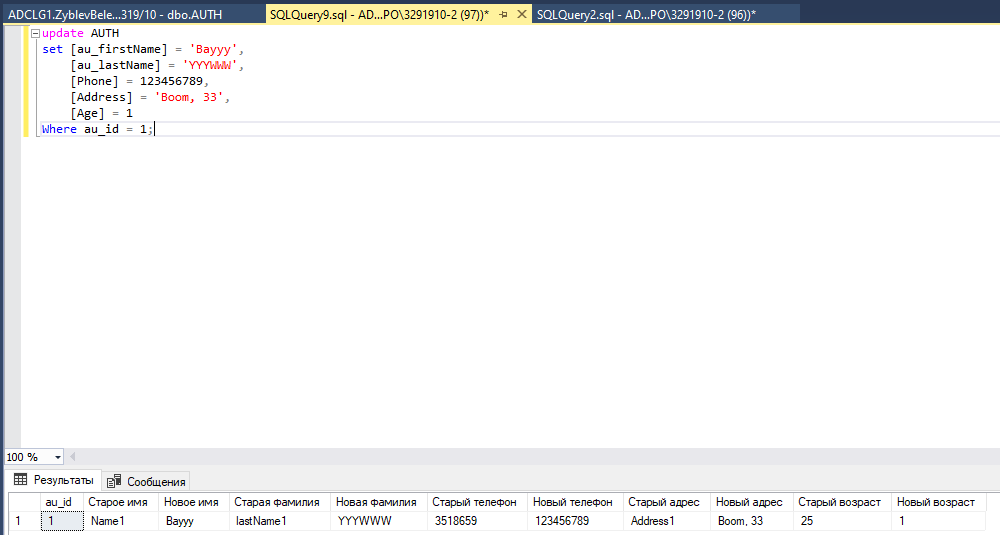
Протестируйте ваш триггер.

4.Создайте триггер Trigger\_3, который будет считать и выводить количество удаленных строк

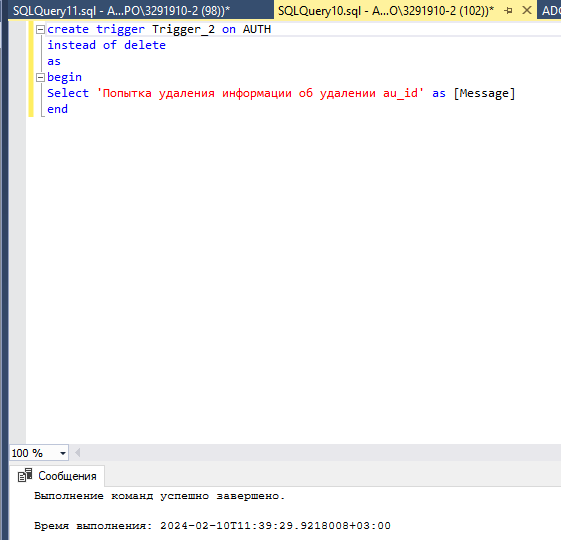
Протестируйте ваш триггер.

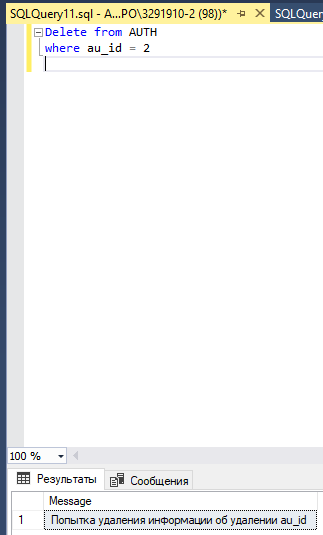
Задание 2



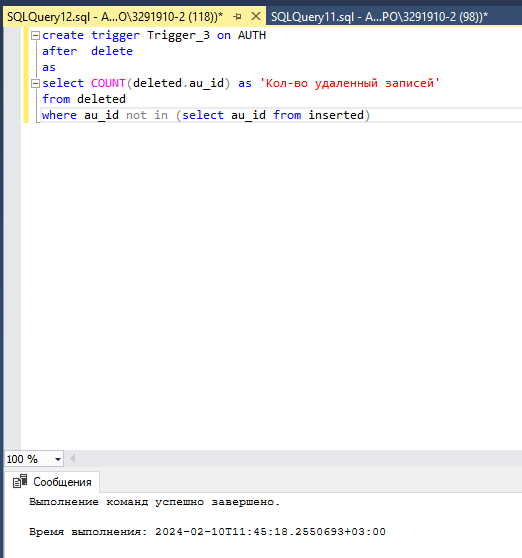


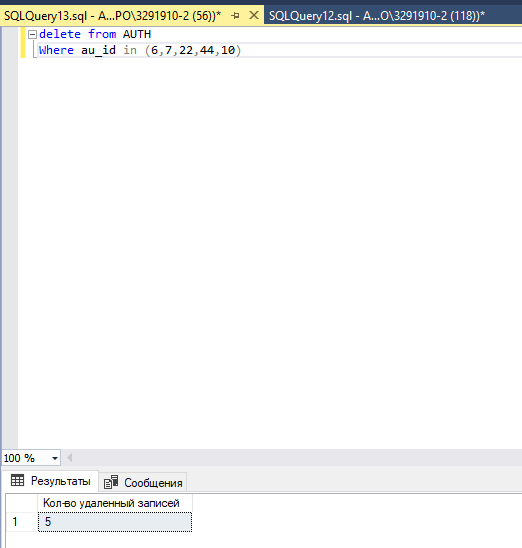
Задание 3





Задание 4





Контрольные вопросы:   
1. Какие типы триггеров вы знаете?   
Существуют разные типы триггеров в зависимости от события, которое вызывает их выполнение. Некоторые из наиболее распространенных типов триггеров включают триггеры до и после INSERT, UPDATE или DELETE операции, а также триггеры, выполняющиеся после команд CREATE, ALTER или DROP.   
2. Можем ли мы обращаться к журналу транзакций при создании триггеров?   
Да, во многих СУБД можно обращаться к журналу транзакций при создании триггеров. Журнал транзакций содержит информацию о всех операциях, выполняемых в базе данных, и может использоваться для получения информации, необходимой для работы триггера.   
3. Как можно протестировать ваш триггер?   
Для тестирования триггера можно создать тестовые данные в базе данных, которые вызовут выполнение триггера. Затем можно проверить результаты выполнения триггера, например, проверив изменения в других таблицах или выводя сообщение об успешном выполнении триггера.   
4. Сколько триггеров может иметь одно отношение?   
Обычно каждое отношение может иметь не более одного триггера на каждый тип операции (INSERT, UPDATE, DELETE, CREATE, ALTER, DROP). Однако, в некоторых СУБД существуют ограничения на количество триггеров для отдельного отношения.   
5. Можно ли создать один триггер на несколько операций?   
Да, один триггер может реагировать на несколько операций. Например, можно создать триггер, который будет выполняться как после INSERT, так и после UPDATE операции.   
6. Как можно увидеть созданные триггеры?   
В каждой СУБД есть специальные системные таблицы или представления, которые содержат информацию о созданных триггерах. Например, в PostgreSQL можно использовать представление "information\_schema.triggers" для просмотра списка созданных триггеров.   
7. Как можно изменить тело триггера?   
Чтобы изменить тело триггера, нужно внести соответствующие изменения в его определение. В большинстве СУБД - это можно сделать с помощью команды ALTER TRIGGER.

# **Лабораторная работа №13.**

**Создание и управление транзакциями**

Задания на лабораторную работу.

1.Проверьте работу транзакции следующим образом. Выберите любое отношение в вашей базе данных. Создайте транзакцию, в которой будет содержаться один запрос вставки. Запустите ее дважды. В первом случае завершите транзакцию командой COMMIT, второй раз ROLLBACK. Проанализируйте результаты. Вставьте скриншот результата в отчет.

2.Работа с вложенными транзакциями.

Создать вложенные транзакции, выполнив следующие команды:

CREATE TABLE table\_name (a1 int) -- 0-й уровень

BEGIN TRANSACTION tr1-- 1-й уровень

INSERT INTO table\_name VALUES (11)

BEGIN TRANSACTION tr2-- 2-й уровень

INSERT INTO table\_name VALUES (22)

BEGIN TRANSACTION tr3 -- 3-й уровень

INSERT INTO table\_name VALUES (33)

SELECT \* FROM table\_name

SELECT 'Вложенность транзкций', @@TRANCOUNT

ROLLBACK TRANSACTION

SELECT \* FROM table\_name -- откат на 0-й уровень

SELECT 'Вложенность транзакций', @@TRANCOUNT

Проанализируйте результаты.

3. Работа с данными из файла groups. Импортируйте все листы. В первом листе находятся данные студентов, поступивших в 2018 году. В других листах содержаться списки групп. Задача импортировать данные из списков групп в сводную таблицу всех поступивших. Создайте явную транзакцию, в которой данные о студентах будут вноситься, а, если эти данные уже были внесены раннее, то транзакция будет откатываться. Результаты продемонстрируйте в отчете.

4.Изучите самостоятельно команду SET IMPLICIT\_TRANSACTIONS ON/OFF.

Продемонстрируйте разницу между этими режимами на примере любой таблицы в вашей базе данных.

Контрольные вопросы:

1. Зачем нужны транзакции?
2. С какими режимами транзакции вы работали?
3. Какие команды являются признаком начала неявной транзакции?
4. Зачем нужны контрольные точки транзакций? Продемонстрируйте их работу.
5. Можно ли узнать количество активных транзакций?

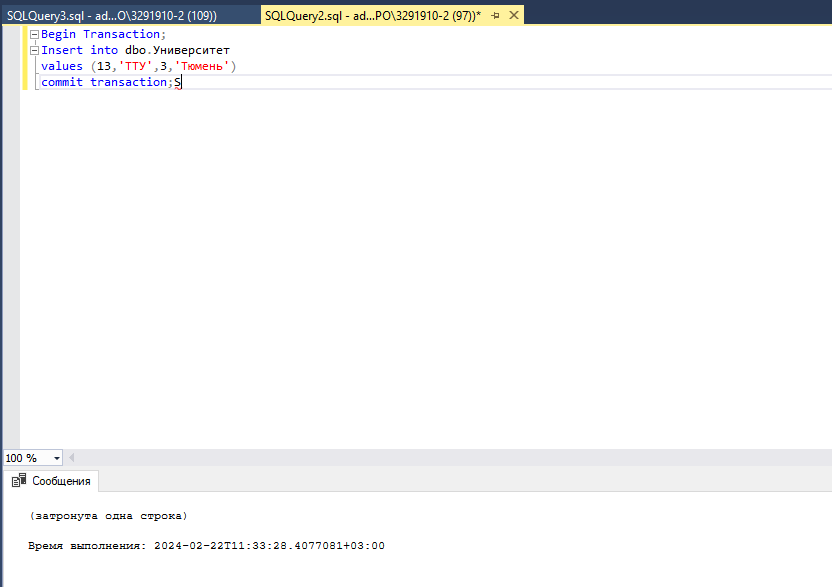
Задание 1

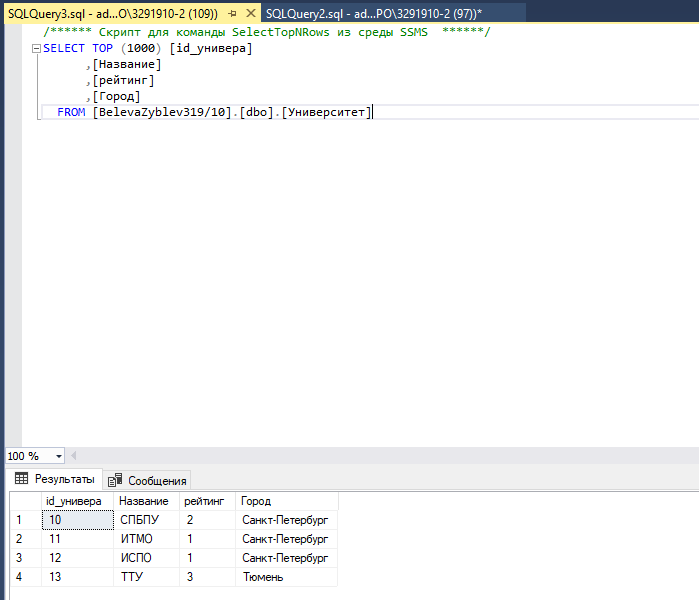
Begin Transaction;

Insert into dbo.Университет

values (13,'ТТУ',3,'Тюмень')

commit transaction;



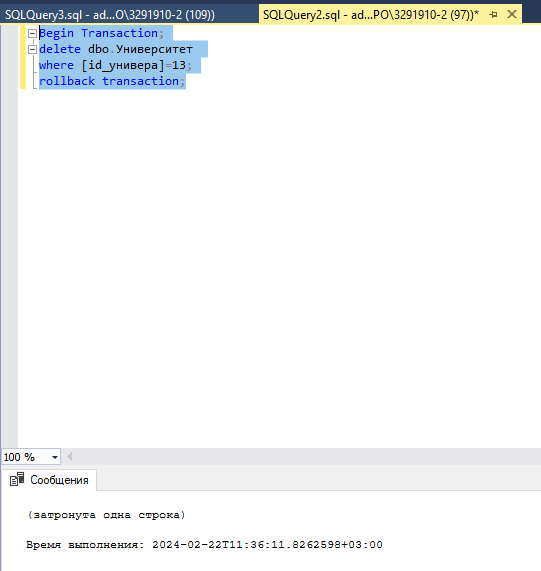


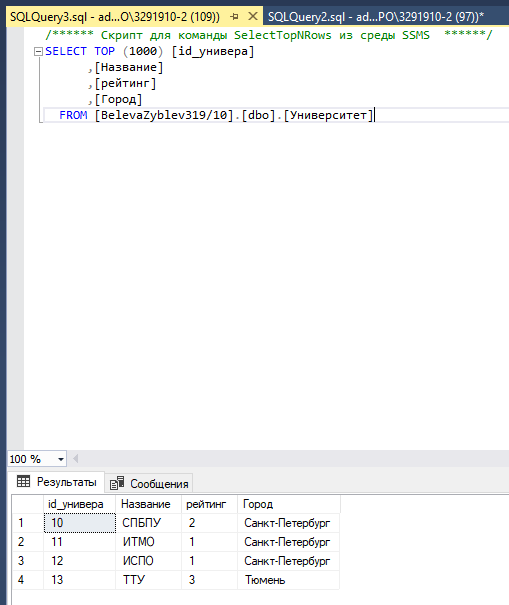
Begin Transaction;

delete dbo.Университет

where [id\_универа]=13;

rollback transaction;





Задание 2

create table test\_transaction (a1 int)

begin transaction tr1

insert into test\_transaction values (11)

begin transaction tr2

INSERT INTO test\_transaction VALUES (22)

begin transaction tr3

INSERT INTO test\_transaction VALUES (33)

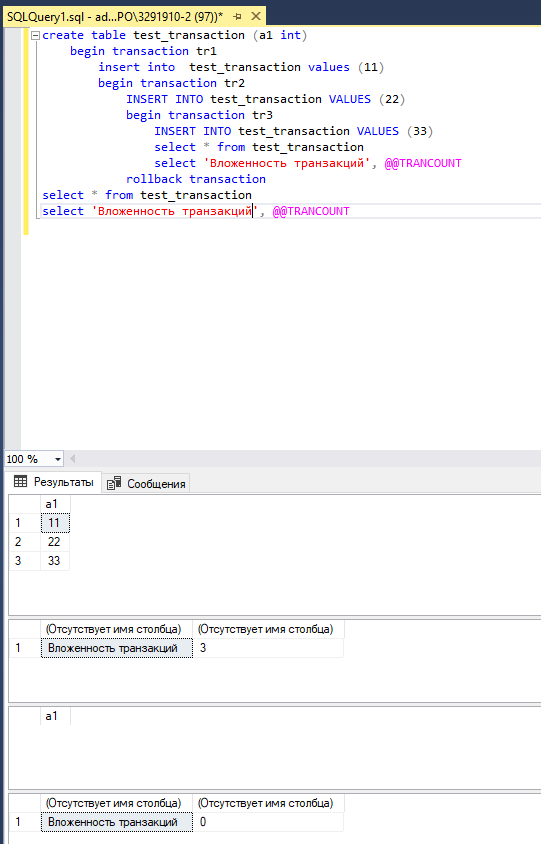
select \* from test\_transaction

select 'Вложенность транзакций', @@TRANCOUNT

rollback transaction

select \* from test\_transaction

select 'Вложенность транзакций', @@TRANCOUNT



Задание 3

declare @groups table (id int identity (0,1), ФИО nvarchar(255),Группа int)

insert into @groups select \* from group212$

insert into @groups select \* from group213$

declare @i int

set @i=0;

while @i<(select count (\*)from @groups)

begin

declare @name nvarchar(255)

select @name = ФИО from @groups where id=@i

begin transaction tr3 insert into nabor2018$ select ФИО, группа from @groups where id = @i

if (select count(\*) from nabor2018$ where ФИО = @name) > 1

rollback tran tr3

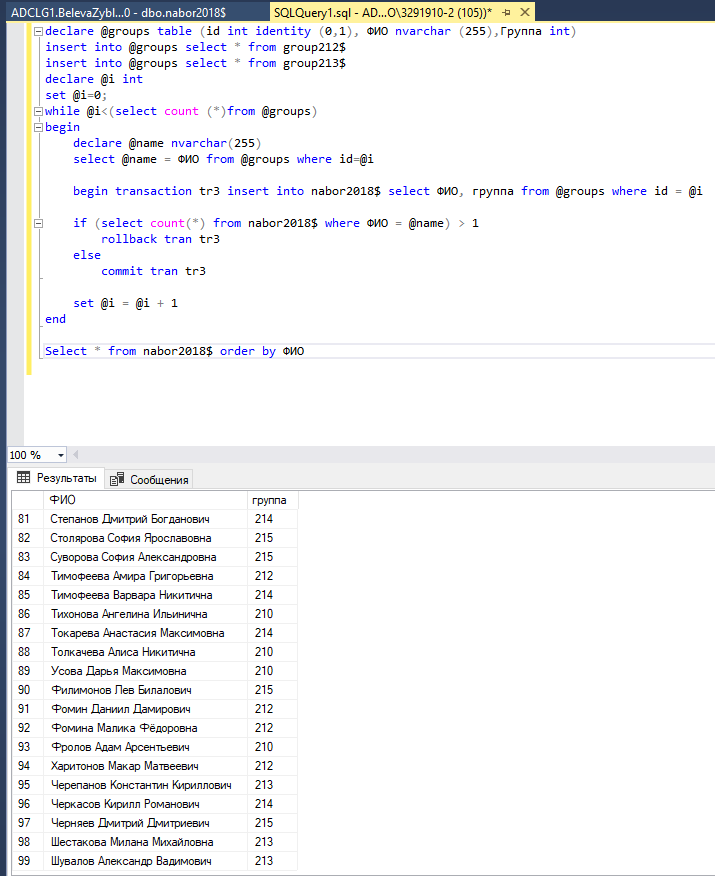
else

commit tran tr3

set @i = @i + 1

end

Select \* from nabor2018$ order by ФИО

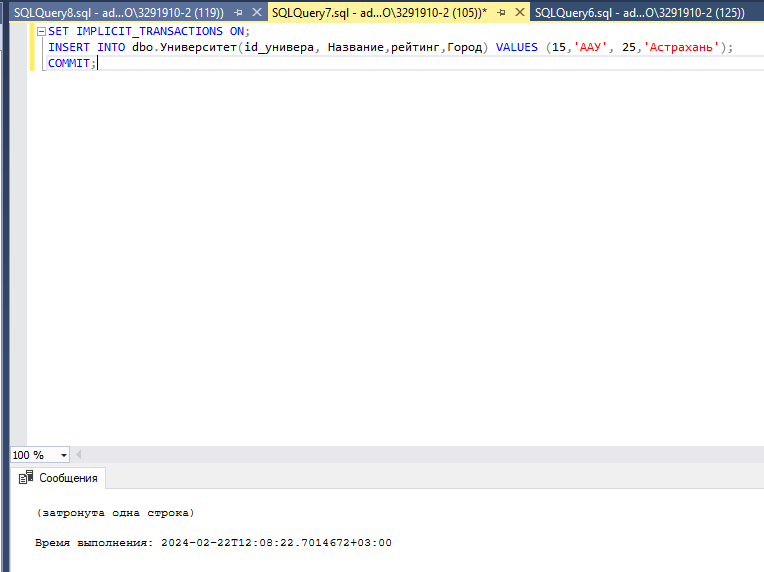


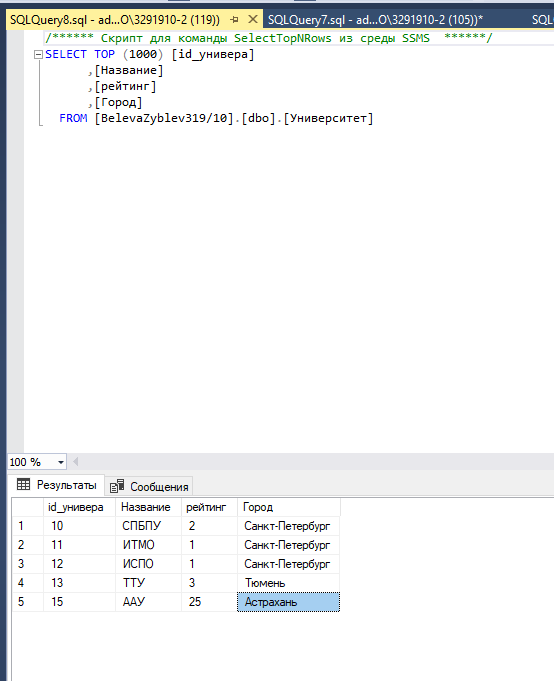
Задание 4

SET IMPLICIT\_TRANSACTIONS ON;

INSERT INTO dbo.Университет(id\_универа, Название,рейтинг,Город) VALUES (15,'ААУ', 25,'Астрахань');

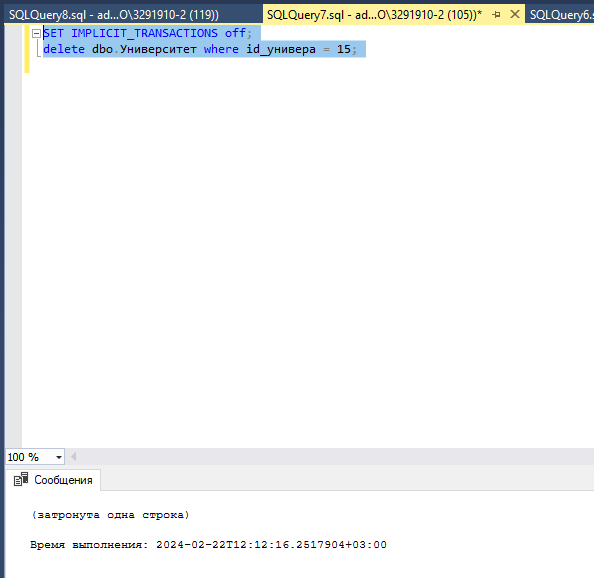
COMMIT;

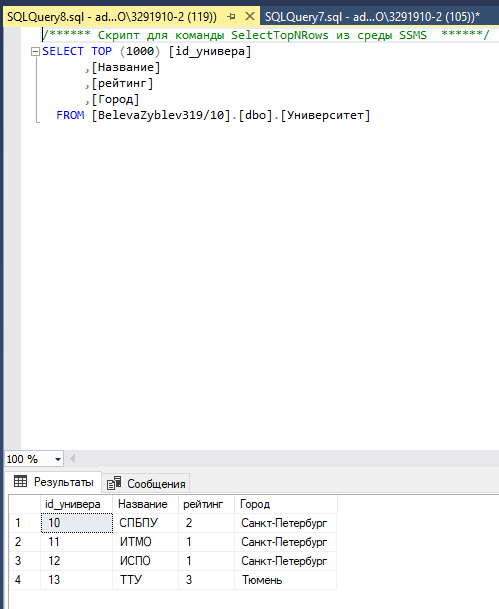




SET IMPLICIT\_TRANSACTIONS off;

delete dbo.Университет where id\_универа = 15;





Вывод:

Если **SET IMPLICIT\_TRANSACTIONS установлено в OFF (по умолчанию в sql), то происходит автоматическое выполнение команды**

**Если же SET IMPLICIT\_TRANSACTIONS установлено в ON, то нужно указывать commit или roolback**

1. Зачем нужны транзакции?   
Транзакции нужны для обеспечения целостности данных и поддержки соответствующих операций в базе данных. Они гарантируют, что набор операций будет выполнен целиком или не выполнен вовсе, чтобы избежать непредвиденных и неправильных результатов.   
2. С какими режимами транзакции вы работали?   
Я работала со следующими режимами транзакции:   
   - Автоматический режим:   
Транзакции автоматически начинаются и завершаются при выполнении операций. Если операция завершается успешно, изменения сохраняются, иначе откатываются.   
   - Ручной режим:   
Транзакции начинаются и завершаются с помощью командных операторов (BEGIN, COMMIT, ROLLBACK).   
3. Какие команды являются признаком начала неявной транзакции?   
Команды, являющиеся признаком начала неявной транзакции, в зависимости от СУБД могут варьироваться. В SQL Server и Oracle, например, BEGIN TRANSACTION или зарезервированные ключевые слова (INSERT, UPDATE, DELETE) автоматически начинают транзакцию, если она еще не активна.   
4. Зачем нужны контрольные точки транзакций? Продемонстрируйте их работу.   
Контрольные точки транзакций используются для сохранения изменений данных в базе данных. Они позволяют вернуться к сохраненному состоянию базы данных при возникновении сбоев или ошибок в процессе выполнения транзакции. Контрольные точки обычно сопровождаются командами CHECKPOINT в СУБД, таких как SQL Server или Oracle.   
- Создание таблицы для демонстрации   
CREATE TABLE TestTable (ID INT, Name VARCHAR(50))   
- Вставка данных   
INSERT INTO TestTable (ID, Name) VALUES (1, 'John')   
INSERT INTO TestTable (ID, Name) VALUES (2, 'Jane')   
- Создание контрольной точки   
CHECKPOINT   
- Изменение данных   
UPDATE TestTable SET Name = 'Mike' WHERE ID = 2   
- Восстановление состояния до контрольной точки   
ROLLBACK   
- Проверка результата   
SELECT \* FROM TestTable   
5. Можно ли узнать количество активных транзакций?   
В SQL можно узнать количество активных транзакций с помощью следующего запроса:   
SELECT COUNT(\*) FROM sys.dm\_tran\_active\_transactions   
Этот запрос дает общее количество активных транзакций в базе данных.

# **Лабораторная работа №14.**

**Создание и управление курсорами**

Задания на лабораторную работу.

1. Создайте таблицу Clients со следующими атрибутами: Id, Name, Address.
2. С помощью курсора создайте запрос, который будет выводить информацию о всех клиентах c нечетным Id.
3. Создайте таблицу OlympMath с результатами Олимпиады по математике за 2019, 2020 и 2021 года. Таблица OlympMath состоит из следующих атрибутов: id, Name, res19, res20, res21.
4. С помощью курсора создайте запрос, который будет находить и выводить лучший результат для каждого участника. То есть максимальный в строке.
5. Создайте две таблицы со следующей структурой (папки и файлы):

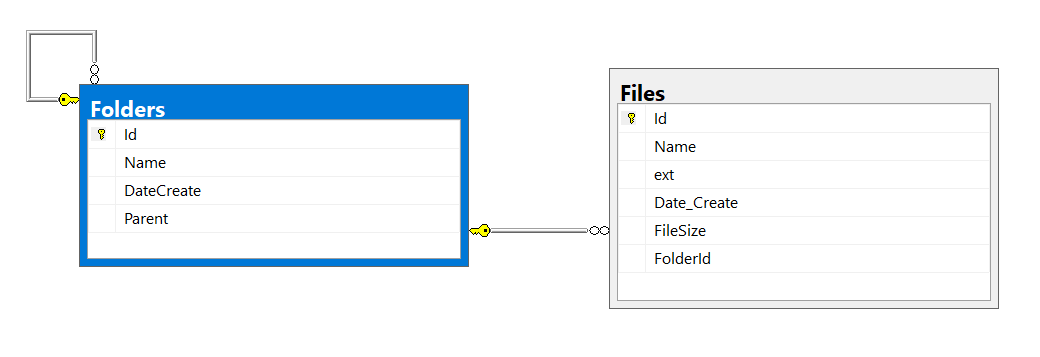


Рис.14 Таблицы Папки и файлы

Реализуйте функцию, которая будет принимать на входе идентификатор каталога и выводить все файлы и каталоги, которые находятся в заданном каталоге и во всех подкаталогах.

1.

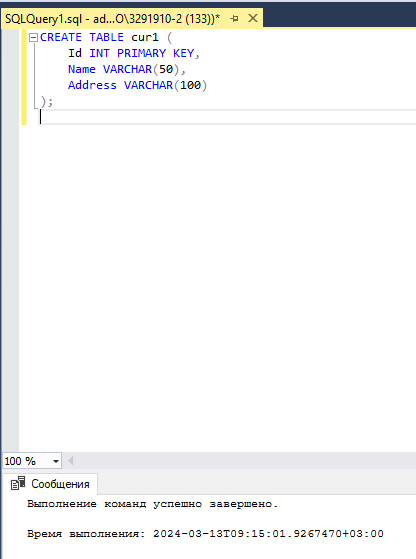
CREATE TABLE cur1 (

Id INT PRIMARY KEY,

Name VARCHAR(50),

Address VARCHAR(100)

);



INSERT INTO cur1 (Id, Name, Address) VALUES

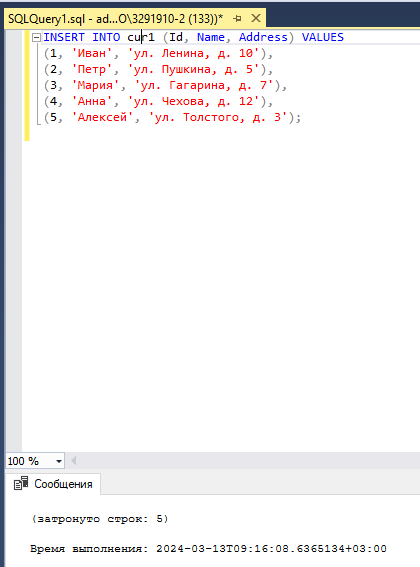
(1, 'Иван', 'ул. Ленина, д. 10'),

(2, 'Петр', 'ул. Пушкина, д. 5'),

(3, 'Мария', 'ул. Гагарина, д. 7'),

(4, 'Анна', 'ул. Чехова, д. 12'),

(5, 'Алексей', 'ул. Толстого, д. 3');



Создаем вторую таблицу для вывода результат работы курсора

CREATE TABLE cur2

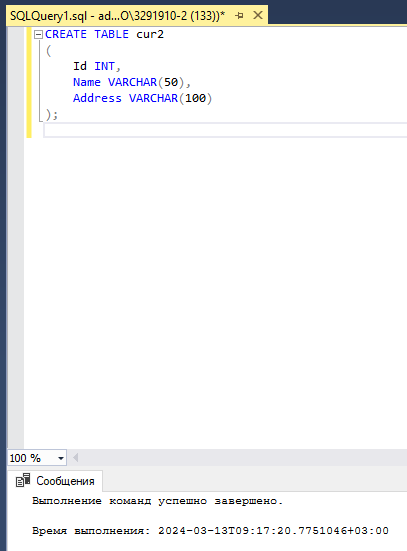
(

Id INT,

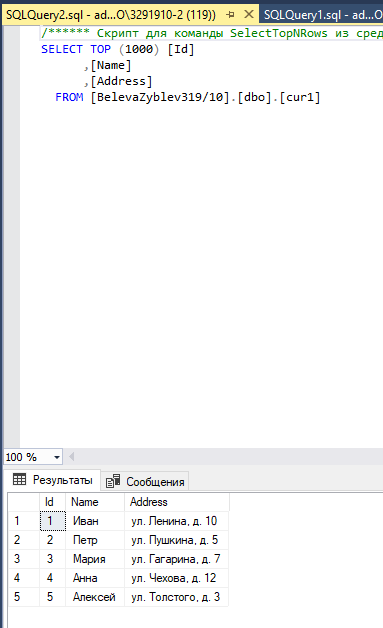
Name VARCHAR(50),

Address VARCHAR(100)

);



Cur 1



Выполняем курсор

DECLARE @ClientId2 INT;

DECLARE @Name2 VARCHAR(50);

DECLARE @Address2 VARCHAR(100);

DECLARE ClientCursor2 CURSOR FOR

SELECT ID, Name, Address

FROM cur1

WHERE ID % 2 != 0;

-- Открытие курсора

OPEN ClientCursor2;

-- Запись данных из курсора во временную таблицу

FETCH NEXT FROM ClientCursor2 INTO @ClientId2, @Name2, @Address2;

WHILE @@FETCH\_STATUS = 0

BEGIN

INSERT INTO cur2 (ID, Name, Address) VALUES (@ClientId2, @Name2, @Address2);

FETCH NEXT FROM ClientCursor2 INTO @ClientId2, @Name2, @Address2;

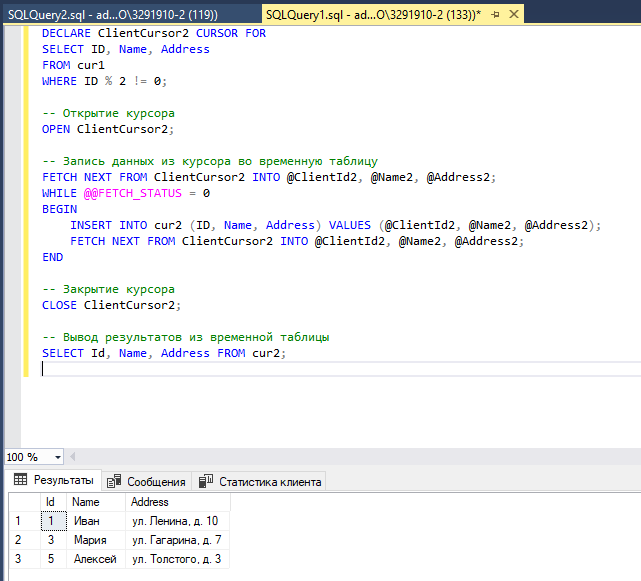
END

-- Закрытие курсора

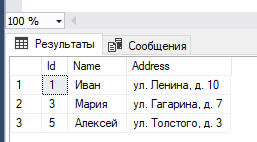
CLOSE ClientCursor2;

-- Вывод результатов из временной таблицы

SELECT Id, Name, Address FROM cur2;



Cur 2



2.

CREATE TABLE OlympMath (

id INT PRIMARY KEY,

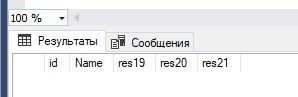
Name VARCHAR(50),

res19 INT,

res20 INT,

res21 INT

);



INSERT INTO OlympMath (id, Name, res19, res20, res21) VALUES

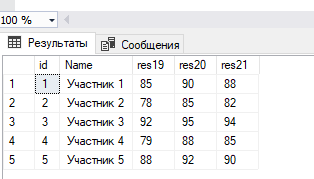
(1, 'Участник 1', 85, 90, 88),

(2, 'Участник 2', 78, 85, 82),

(3, 'Участник 3', 92, 95, 94),

(4, 'Участник 4', 79, 88, 85),

(5, 'Участник 5', 88, 92, 90);



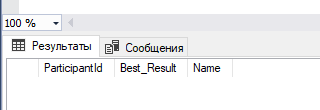
CREATE TABLE BestRES (

ParticipantId INT,

Best\_Result INT,

Name nvarchar(50)

);



declare @id\_player int;

declare @name\_player VARCHAR(50);

declare @best\_res\_player int;

declare @res1\_player int;

declare @res2\_player int;

declare @res3\_player int;

declare Play\_cur cursor for

Select id, Name, res19,res20,res21

from OlympMath;

open Play\_cur

Fetch next from Play\_cur into @id\_player,@name\_player,@res1\_player,@res2\_player,@res3\_player;

while @@FETCH\_STATUS= 0

begin

SET @best\_res\_player = (SELECT MAX(Result) FROM (VALUES (@res1\_player), (@res2\_player), (@res3\_player)) AS AllResults(Result));

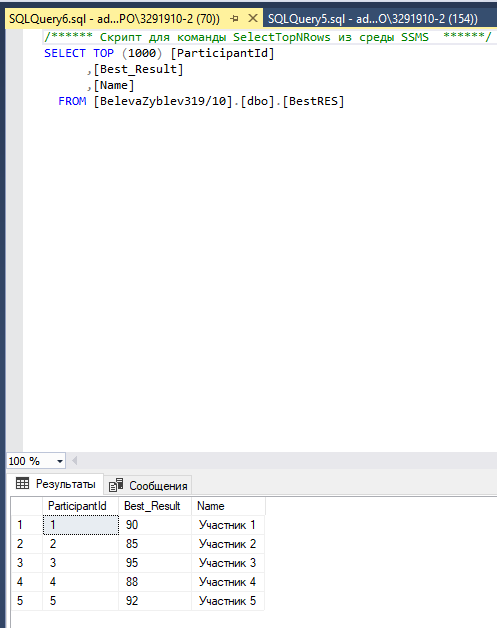
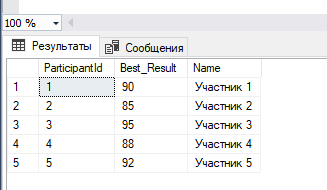
INSERT INTO BestRES (ParticipantId, Best\_Result, Name) VALUES (@id\_player, @best\_res\_player, @name\_player);

FETCH NEXT FROM Play\_cur INTO @id\_player, @name\_player, @res1\_player, @res2\_player, @res3\_player;

END

CLOSE Play\_cur;

SELECT ParticipantId, Best\_Result, Name FROM BestRES;



3.

Create table folders(

id int primary key,

name nvarchar(60),

DateSoz date,

parent int);

Create table files(

id int primary key,

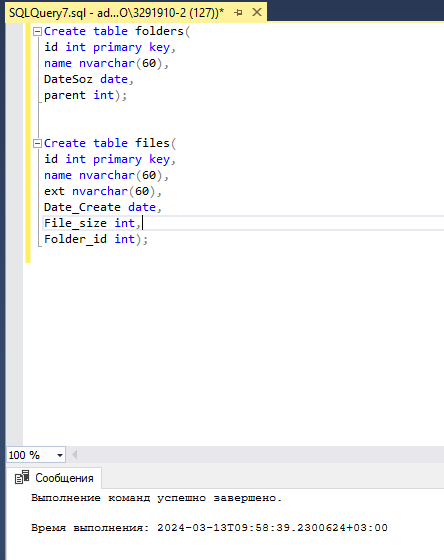
name nvarchar(60),

ext nvarchar(60),

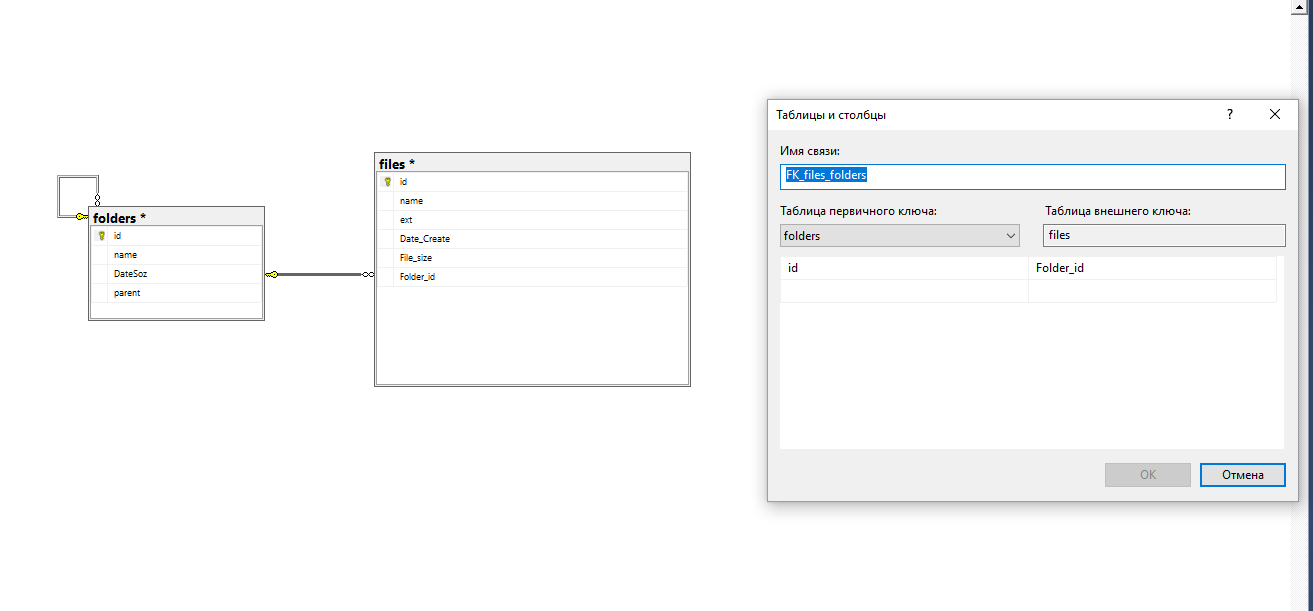
Date\_Create date,

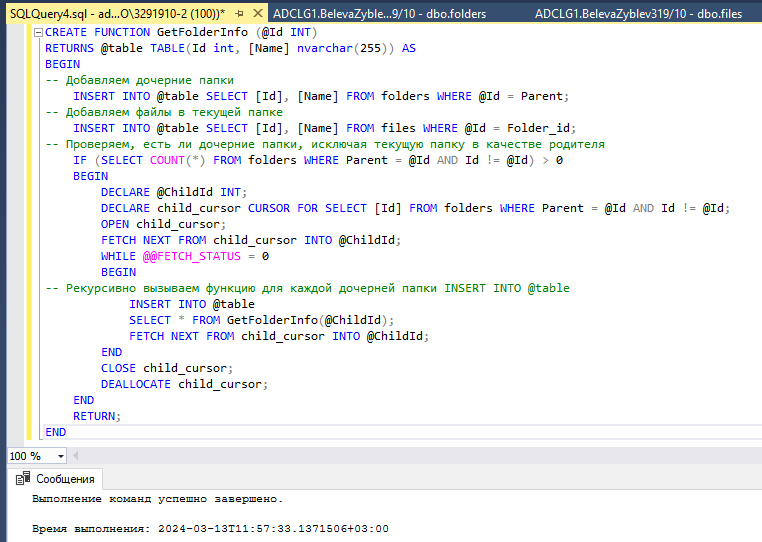
File\_size int,

Folder\_id int);



ДИАГРАММА





CREATE FUNCTION GetFolderInfo (@Id INT)

RETURNS @table TABLE(Id int, [Name] nvarchar(255)) AS

BEGIN

-- Добавляем дочерние папки

INSERT INTO @table SELECT [Id], [Name] FROM folders WHERE @Id = Parent;

-- Добавляем файлы в текущей папке

INSERT INTO @table SELECT [Id], [Name] FROM files WHERE @Id = Folder\_id;

-- Проверяем, есть ли дочерние папки, исключая текущую папку в качестве родителя

IF (SELECT COUNT(\*) FROM folders WHERE Parent = @Id AND Id != @Id) > 0

BEGIN

DECLARE @ChildId INT;

DECLARE child\_cursor CURSOR FOR SELECT [Id] FROM folders WHERE Parent = @Id AND Id != @Id;

OPEN child\_cursor;

FETCH NEXT FROM child\_cursor INTO @ChildId;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- Рекурсивно вызываем функцию для каждой дочерней папки INSERT INTO @table

INSERT INTO @table

SELECT \* FROM GetFolderInfo(@ChildId);

FETCH NEXT FROM child\_cursor INTO @ChildId;

END

CLOSE child\_cursor;

DEALLOCATE child\_cursor;

END

RETURN;

END

