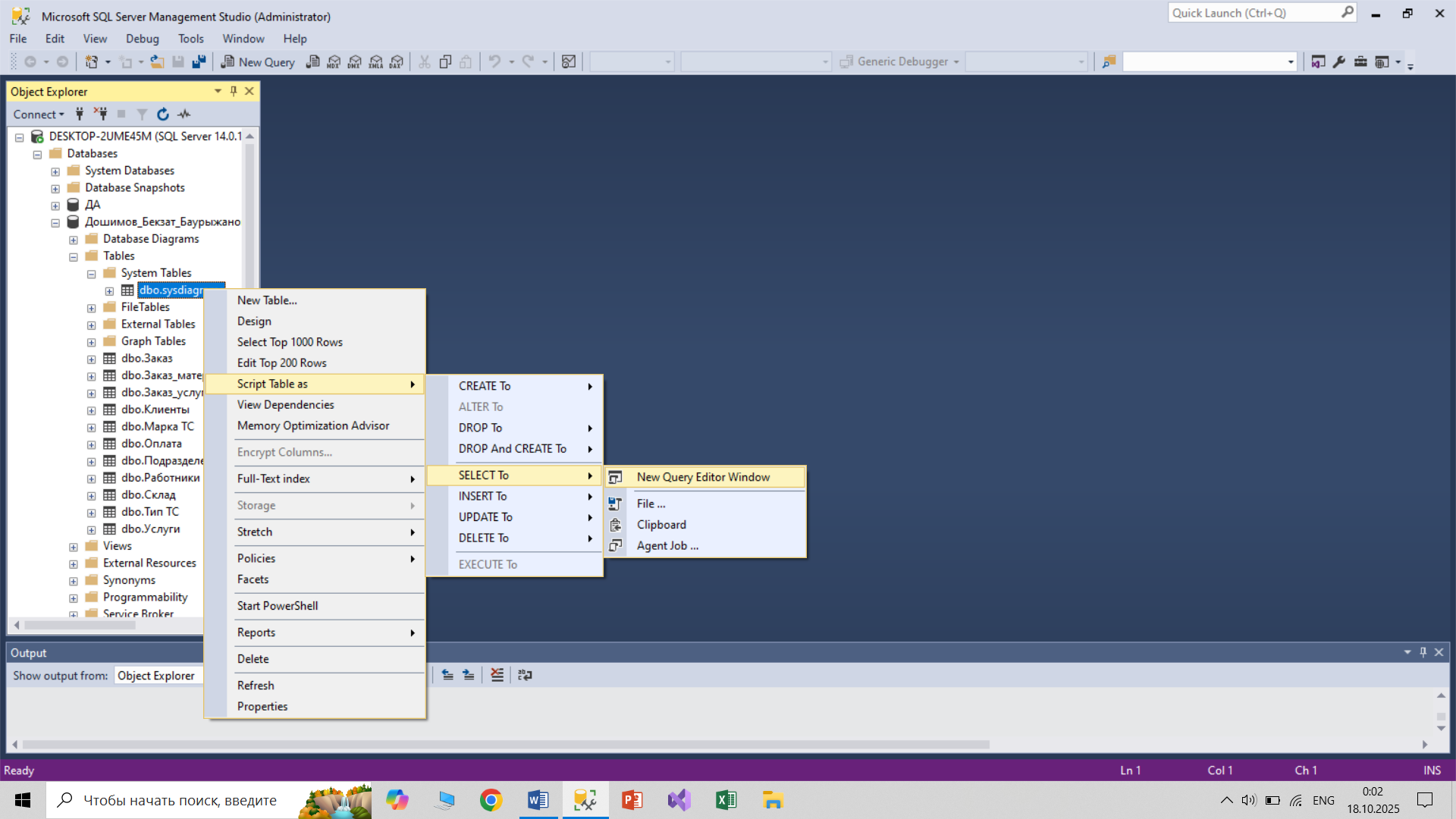
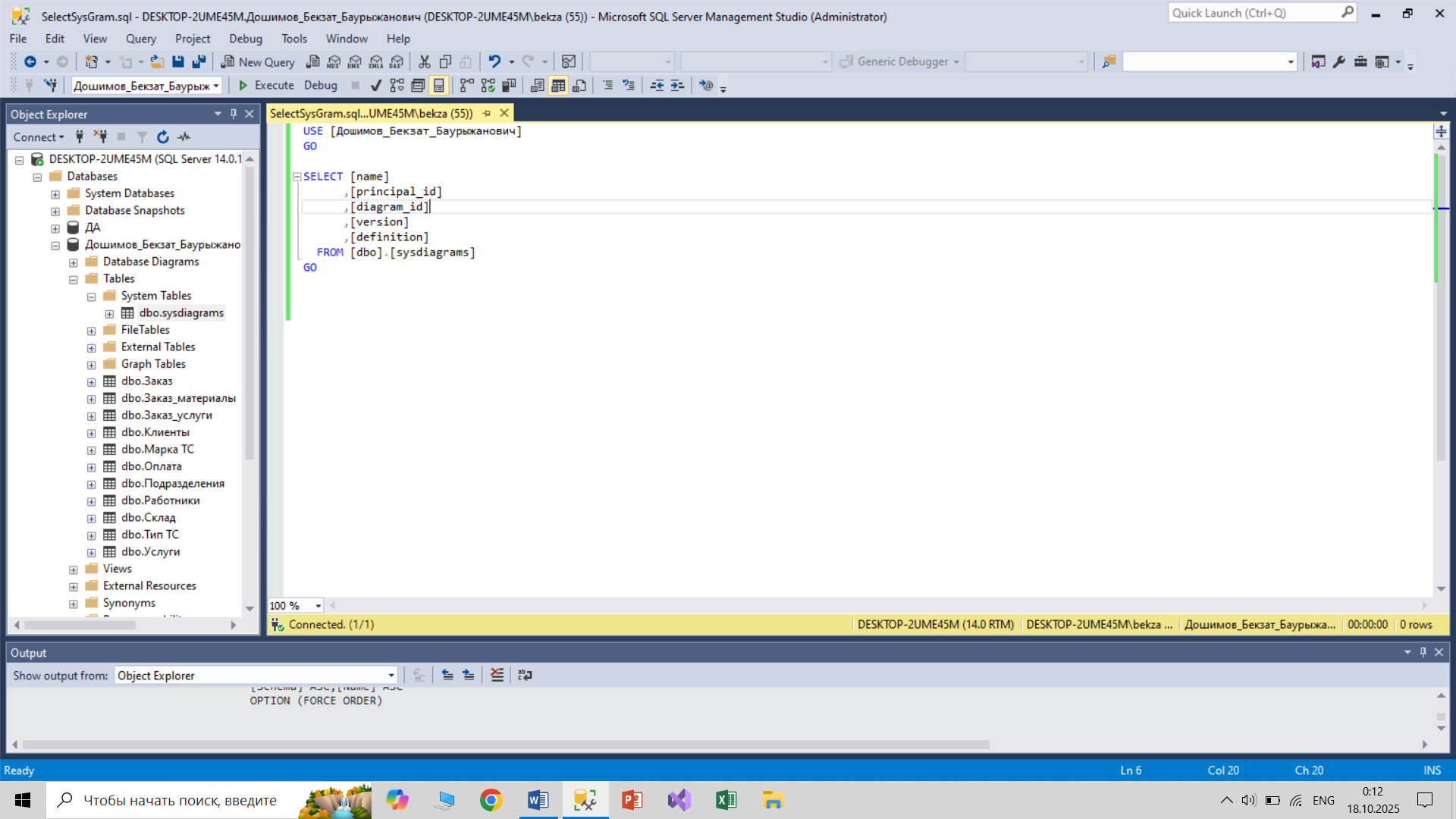
Лабораторная работа № 3

Тема: " Создание запросов и модификация таблиц базы данных "

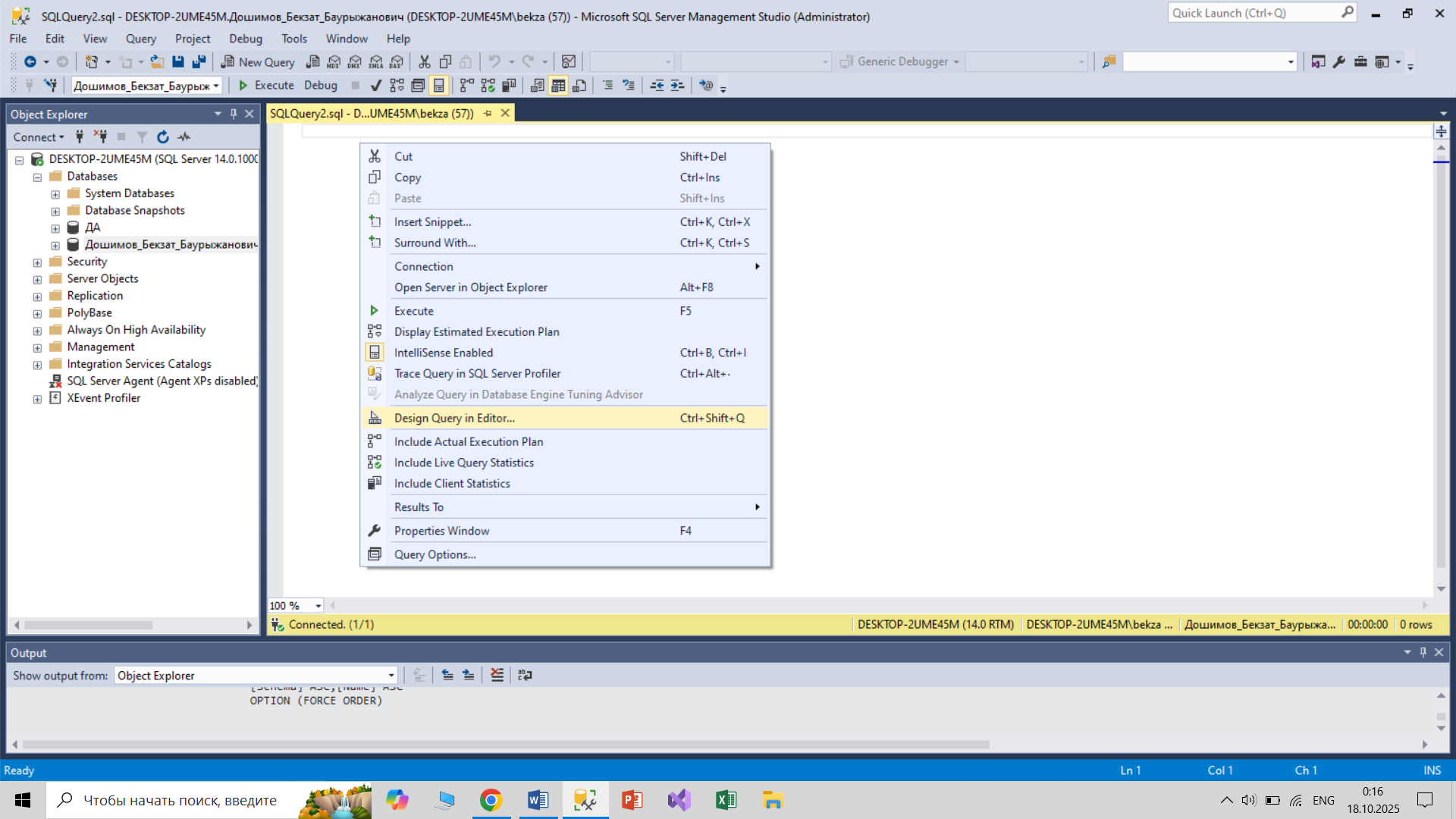
Цель работы: усвоить способы создания выборки данных в среде СУБД MS SQL Server.



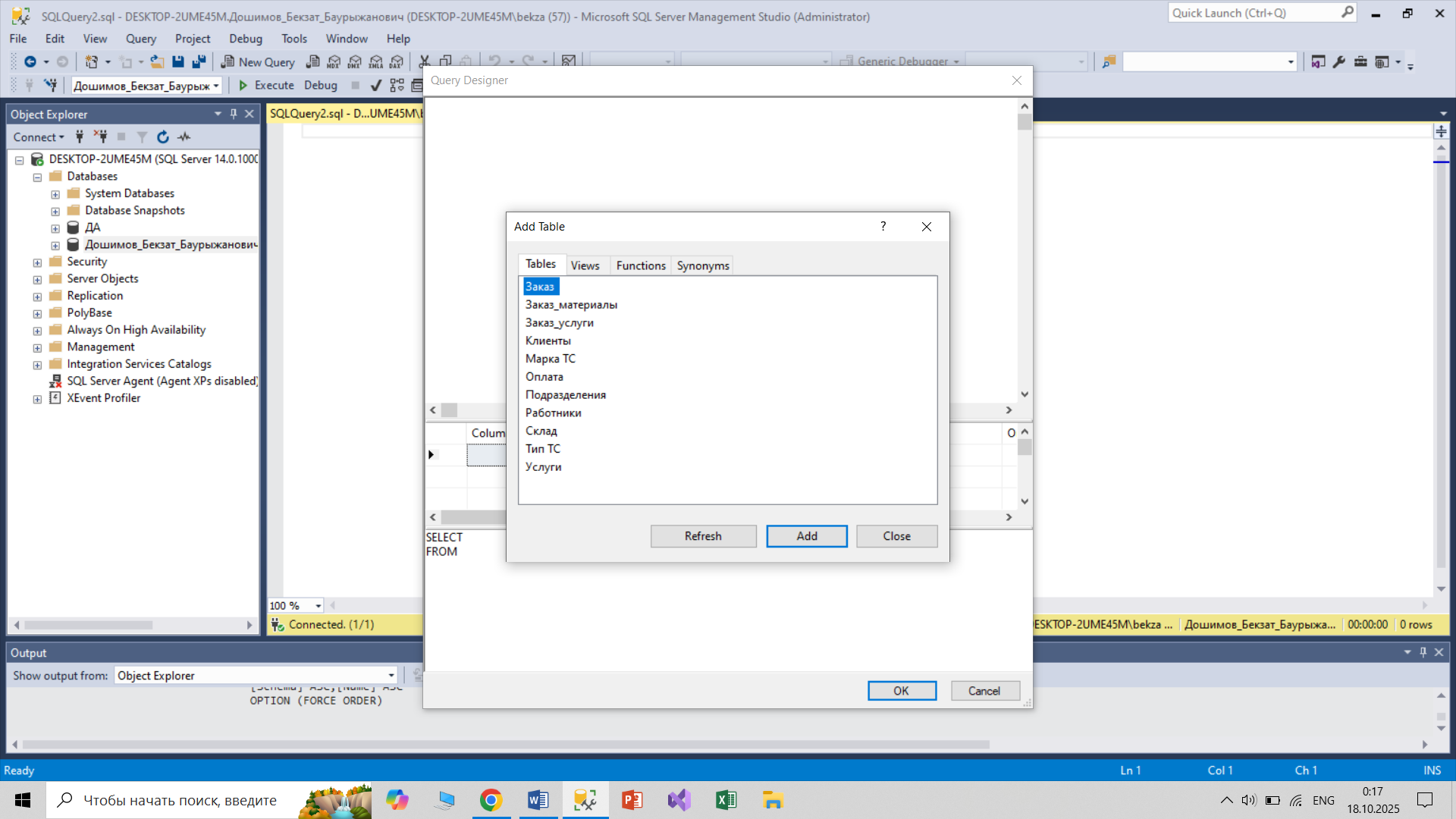
Автоматическая генерация скриптов из Object Explorer



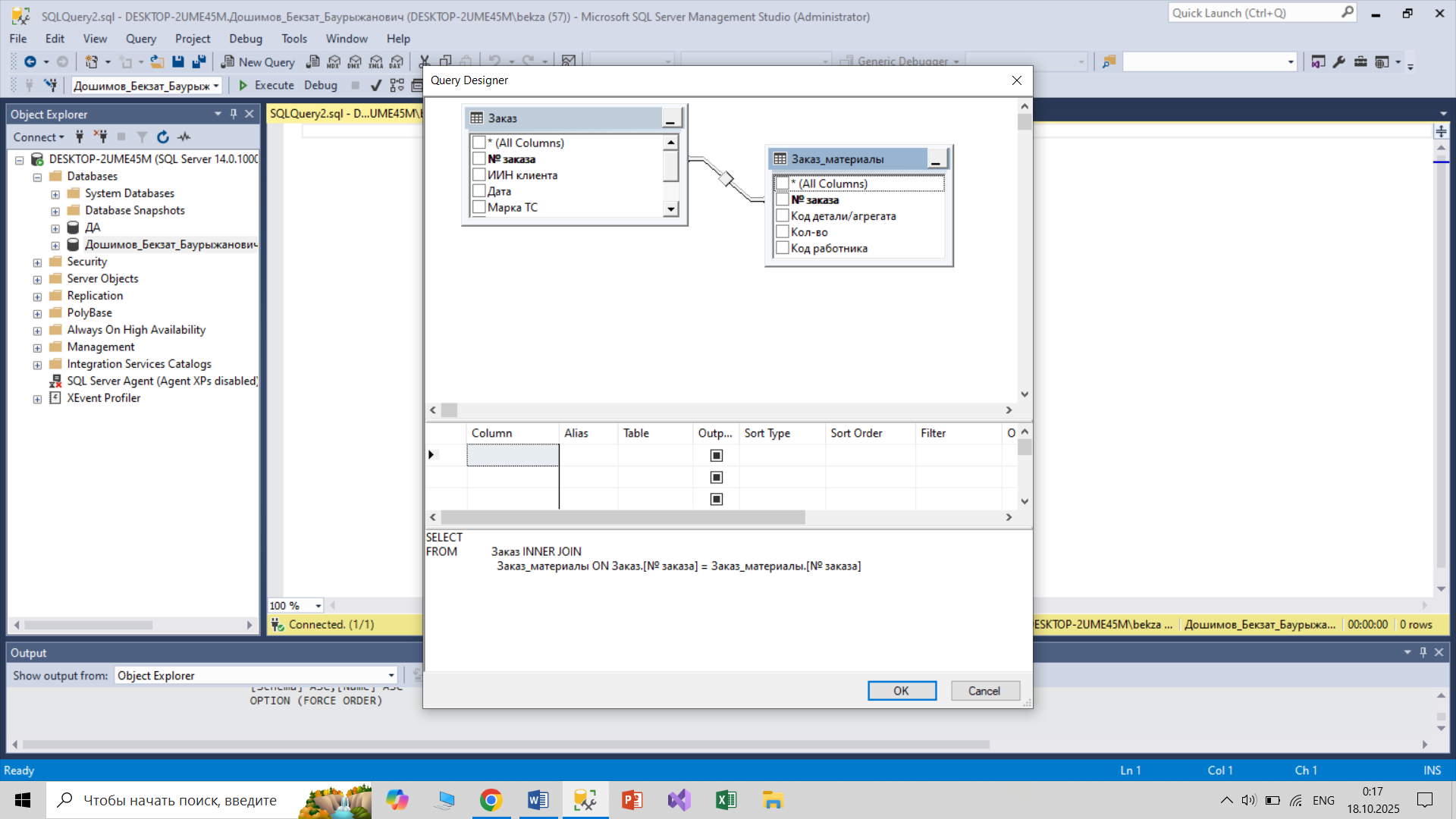
Сгенерированный скрипт



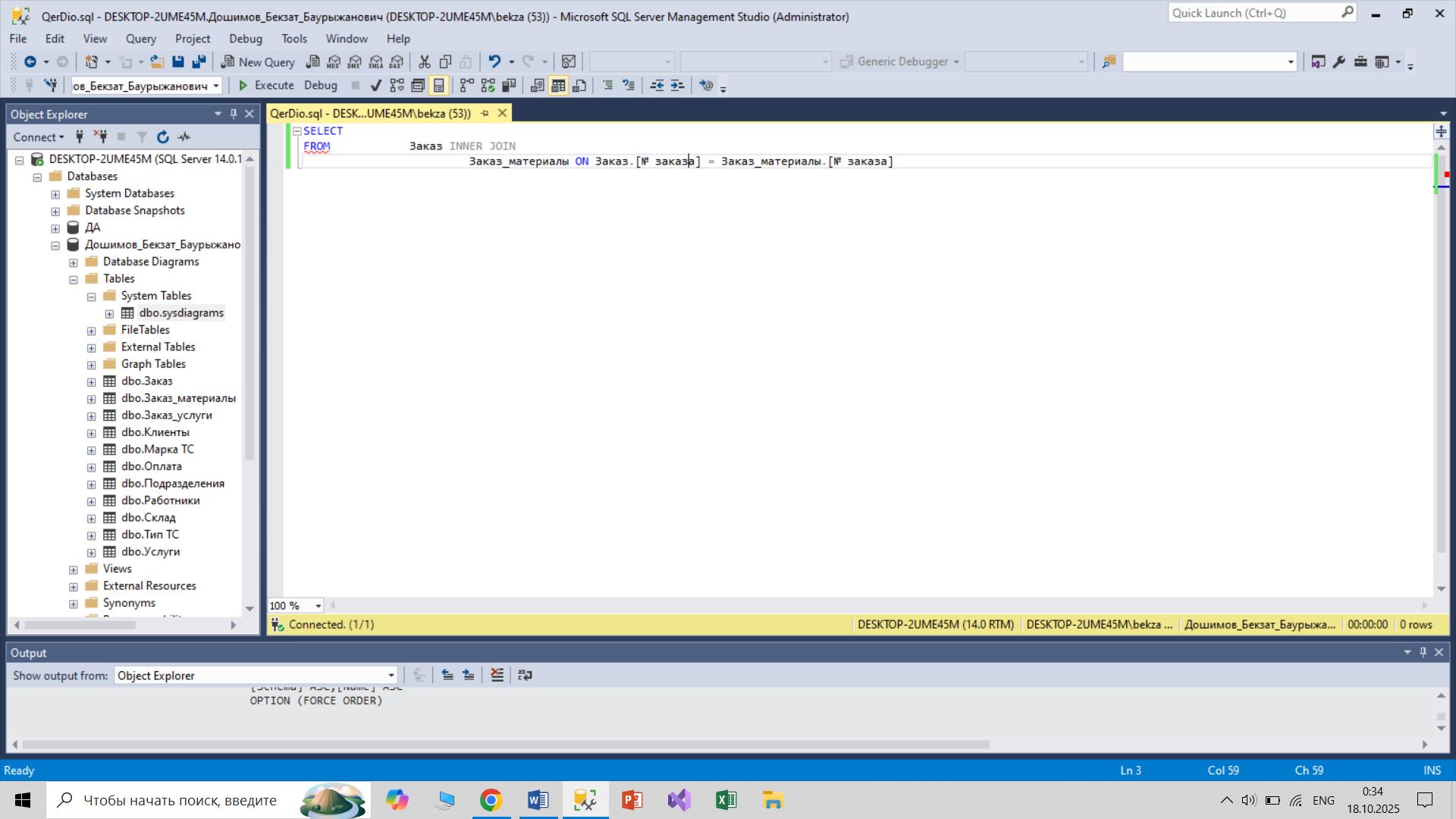
Design Query in Editor



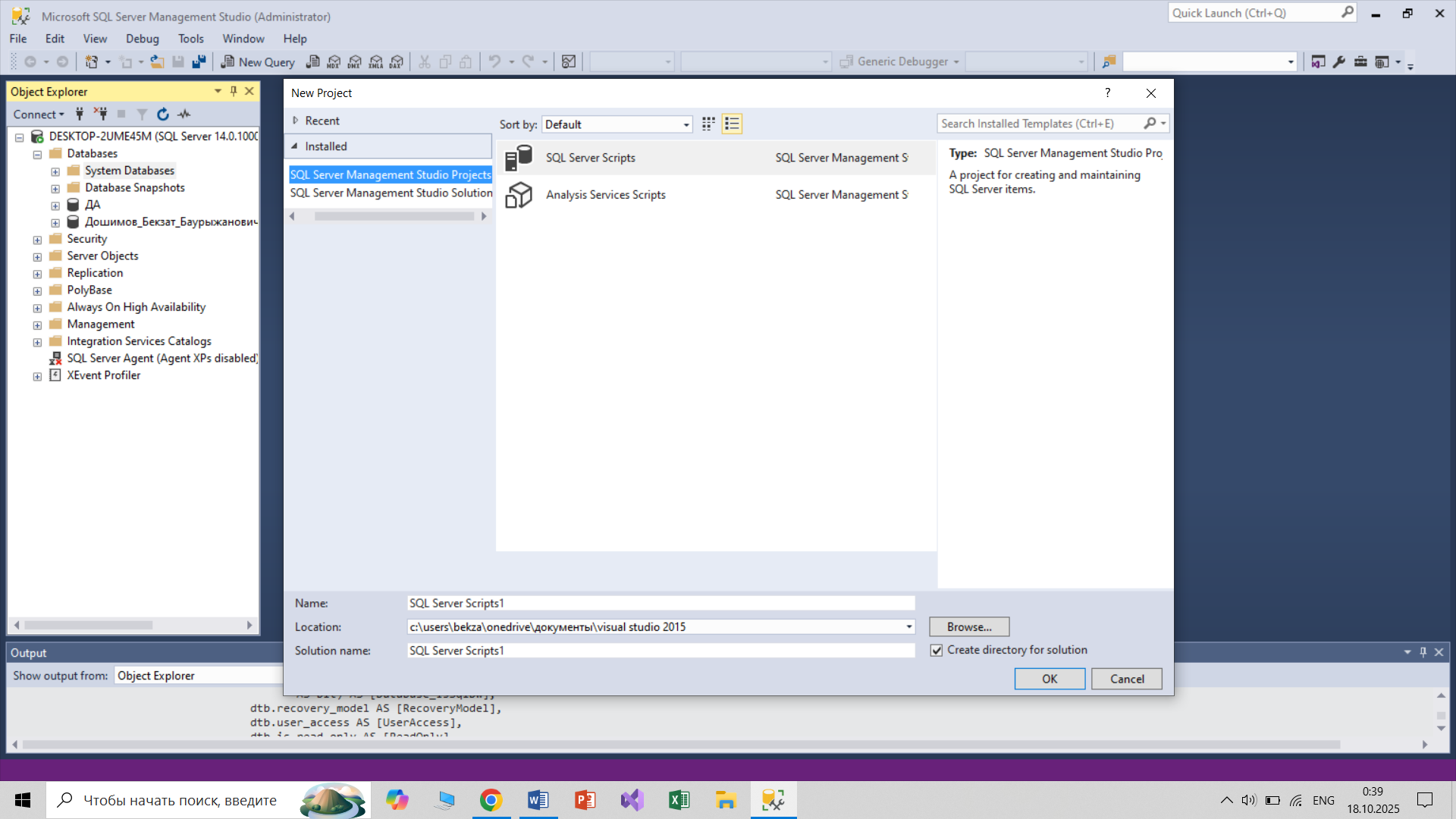
Query Designer выбор таблицы



Query Designer редактор



Query Designer результат



Script action to the new Query

**Контрольные вопросы**

1. Возможности меню Query Editor

|  |  |
| --- | --- |
| **Команды меню Query** | **Описание** |
| Connection | Предоставляет возможности смены рабочей БД или отсоединения запросов от нее. |
| Execute | Выполняет сформированные операторы SQL. |
| Parse | Проверяет правильность запроса, но не выполняет его. |
| Display Estimated Execution Plan | Отображает в нижней части окна вкладку предпологаемый план разбиения выполнения запроса по логическим шагам и машинным ресурсам с указанием степени занятости ресурсов. |
| Analyze Query in Database Engine Tuning Wizard | Запускает мастер оптимизации и настройки индексов базы данных. |
| Design Query in Editor... | Предоставляет возможность визуального создания запросов. |
| Display Actual Execution Plan | Отображает в нижней части окна вкладку действительный план разбиения выполнения запроса по логическим шагам и машинным ресурсам с указанием степени занятости ресурсов. |
| Include Client Statistics | Отображает в нижней части окна вкладку статистику выполнения зароса. |
| Reset Client Statistics | Сбрасывает предыдущие статистические данные выполнения зароса. |
| Use SQLCMD mode | Включает или отключает возможность добавления в скрипт системных команд Windows. |
| Result to > Results to Text Result to > Results to Grid Result to > Results to File | Позволяют выбрать способ представления результатов запросов на выборку данных: в виде простого текста, в виде таблицы или с сохранением в файл соответственно. |
| Query Options | Позволяет добавить к окну с результатами запросов вкладки плана выполнения запроса, серверной трассировки запроса, клиентских статистик запроса, а также установить прочие свойства запроса. |

1. Опишите конструкцию оператора SELECT

**КОМАНДА SELECT**

В общем случае, команда SELECT начинается с ключевого слова SELECT, сопровождаемого пробелом. После этого должен следовать список имен столбцов которые вы хотите видеть, отделяемые запятыми. Ключевое слово FROM следующее далее, сопровождается пробелом и именем таблицы запрос к которой делается. Например,

SELECT snum, sname, sity, comm

FROM Salespeople;

Если вы хотите видеть все столбцы таблицы, вы можете заменить список звездочкой (\*).

SELECT \*

FROM Salespeople;

DISTINCT (ОТЛИЧИЕ) - аргумент который обеспечивает Вас способом устранять повторяющиеся значения из вашего предложения SELECT. DISTINCT может указываться только один раз в данном предложении SELECT.

SELECT DISTINCT snum

FROM Orders;

WHERE - предложение команды SELECT, которое позволяет вам устанавливать предикаты, условие которых может быть или верным или неверным для любой строки таблицы. Команда извлекает только те строки из таблицы для которой такое утверждение верно. Когда предложение WHERE представлено, программа базы данных просматривает всю таблицу по одной строке и исследует каждую строку чтобы определить верно ли утверждение.

SELECT \*

FROM Customers

WHERE rating = 100;

Основные Булевы операторы также распознаются в SQL. Выражения Буля - являются или верными или неверными, подобно предикатам. Булевы операторы связывают одно или более верных/неверных значений и производят единственное верное/или/неверное значение. Стандартными операторами Буля распознаваемыми в SQL являются: AND, OR, и NOT. Связывая предикаты с операторами Буля, вы можете значительно увеличить их возможности.

SELECT \*

FROM Customers

WHERE city = " San Jose' AND rating > 200;

NOT может использоваться для инвертирования значений Буля. Имеется пример запроса с NOT:

SELECT \*

FROM Customers

WHERE city = " San Jose' OR NOT rating > 200;

Оператор IN определяет набор значений в которое данное значение может или не может быть включено.

SELECT \*

FROM Salespeople

WHERE city IN ( 'Barcelona', 'London' );

Как вы можете видеть, IN определяет набор значений с помощью имен членов набора заключенных в круглые скобки и отделенных запятыми. Он затем проверяет различные значения указанного пол пытаясь найти совпадение со значениями из набора. Если это случается, то предикат верен

Оператор BETWEEN похож на оператор IN. В отличии от определения по номерам из набора, как это делает IN, BETWEEN определяет диапазон, значения которого должны уменьшаться что делает предикат верным. Вы должны ввести ключевое слово BETWEEN с начальным значением, ключевое AND и конечное значение. В отличие от IN, BETWEEN чувствителен к порядку, и первое значение в предложении должно быть первым по алфавитному или числовому порядку. (Обратите Внимание что, в отличие от Английского языка, SQL не говорит что "значение находится (между) BETWEEN значением и значением", а просто "значение BETWEEN значение значение". Это применимо и к оператору LIKE). Следующий пример будет извлекать из таблицы Продавцов всех продавцов с комиссионными между 10 и 12 (вывод показывается в Рисунке 5.4):

SELECT \*

FROM Salespeople

WHERE comm BETWEEN 10 AND 12;

подобно реляционным операторам, BETWEEN может работать с символьными полями в терминах эквивалентов ASCII. Это означает что вы можете использовать BETWEEN чтобы выбирать ряд значений из упорядоченных по алфавиту значений.

SELECT \*

FROM Customers

WHERE cname BETWEEN 'A' AND 'G';

LIKE применим только к полям типа CHAR или VARCHAR, с которыми он используется чтобы находить подстроки. Т.е. он ищет поле символа чтобы видеть, совпадает ли с условием часть его строки. В качестве условия он использует групповые символы(wildkards) - специальные символы которые могут соответствовать чему-нибудь. Имеются два типа групповых символов используемых с LIKE:

символ подчеркивания ( \_ ) замещает любой одиночный символ. Например, 'b\_t' будет соответствовать словам 'bat' или 'bit', но не будет соответствовать 'brat'.

знак процента (%) замещает последовательность любого числа символов (включая символы нуля). Например '%p%t' будет соответствовать словам 'put', 'posit', или 'opt', но не 'spite'.

Давайте найдем всех заказчиков чьи имена начинаются с G:

SELECT

FROM Customers

WHERE cname LIKE 'G%';

Часто, будут иметься записи в таблице которые не имеют никаких значений для каждого пол, например потому что информация не завершена, или потому что это поле просто не заполнялось. SQL учитывает такой вариант, позволяя вам вводить значение NULL(ПУСТОЙ) в поле, вместо значения. Так как NULL указывает на отсутствие значения, вы не можете знать каков будет результат любого сравнения с использованием NULL. Когда NULL сравнивается с любым значением, даже с другим таким же NULL, результат будет ни верным ни неверным, он - неизвестен. SQL предоставляет специальный оператор IS, который используется с ключевым словом NULL, для размещения значения NULL. Найдем все записи в нашей таблице Заказчиков с NULL значениями в city столбце:

SELECT \*

FROM Customers

WHERE city IS NULL;

Агрегатные функции используются подобно именам полей в предложении SELECT запроса, но с одним исключением, они берут имена полей как аргументы. Только числовые поля могут использоваться с SUM и AVG. С COUNT, MAX, и MIN, могут использоваться и числовые или символьные поля. Когда они используются с символьными полями, MAX и MIN будут транслировать их в эквивалент ASCII, который должен сообщать, что MIN будет означать первое, а MAX последнее значение в алфавитном порядке. Функция COUNT несколько отличается от всех. Она считает число значений в данном столбце, или число строк в таблице. Когда она считает значения столбца, она используется с DISTINCT чтобы производить счет чисел различных значений в данном поле. Чтобы подсчитать общее число строк в таблице, используйте функцию COUNT со звездочкой:

SELECT COUNT (\*)

FROM Customers

COUNT со звездочкой включает и NULL и дубликаты, по этой причине DISTINCT не может быть использован. Предложение GROUP BY позволяет вам определять подмножество значений в особом поле в терминах другого пол, и применять функцию агрегата к подмножеству. Это дает вам возможность объединять пол и агрегатные функции в едином предложении SELECT. Например, предположим что вы хотите найти наибольшую сумму приобретений полученную каждым продавцом.

SELECT snum, MAX (amt)

FROM Orders

GROUP BY snum;

Вы не сможете использовать агрегатную функцию в предложении WHERE (если вы не используете подзапрос, описанный позже), потому что предикаты оцениваются в терминах одиночной строки, а агрегатные функции оцениваются в терминах групп строк. Аргументы в предложении HAVING следуют тем же самым правилам что и в предложении SELECT, состоящей из команд использующих GROUP BY. Они должны иметь одно значение на группу вывода.