**Лабораторна робота №2**

## Пакети. Команда GO

У попередніх випадках спочатку створювалася база даних, а потім в цю БД додавалася таблиця за допомогою окремих команд SQL. Але можна відразу поєднати в одному скрипті кілька команд. У цьому випадку окремі набори команд називаються пакетами (batch). Кожен пакет складається з одного або декількох SQL-виразів, які виконуються як воно ціле. Як сигнал завершення пакета і виконання його виразів служить команда GO. Сенс поділу SQL-виразів на пакети полягає в тому, що одні вирази повинні успішно виконатися до запуску інших виразів. Наприклад, при додаванні таблиць ми повинні б впевнені, що була створена база даних, в якій ми збираємося створити таблиці.

Наприклад, визначимо наступний скрипт:

CREATE DATABASE productsdb;

GO

USE productsdb;

CREATE TABLE Products

(

Id INT IDENTITY PRIMARY KEY,

ProductName NVARCHAR (30) NOT NULL,

Manufacturer NVARCHAR (20) NOT NULL,

ProductCount INT DEFAULT 0,

Price MONEY NOT NULL

);

CREATE TABLE Customers

(Id INT IDENTITY PRIMARY KEY,

FirstName NVARCHAR (30) NOT NULL

);

CREATE TABLE Orders

(Id INT IDENTITY PRIMARY KEY,

ProductId INT NOT NULL REFERENCES Products (Id) ON DELETE CASCADE,

CustomerId INT NOT NULL REFERENCES Customers (Id) ON DELETE CASCADE,

CreatedAt DATE NOT NULL,

ProductCount INT DEFAULT 1,

Price MONEY NOT NULL

);

Спочатку створюється БД productsdb. Потім йде команда GO, яка сигналізує, що можна виконувати наступний пакет виразів. І далі виконується другий пакет, який підключає базу і додає в неї таблиці: Products, Customers і Orders.

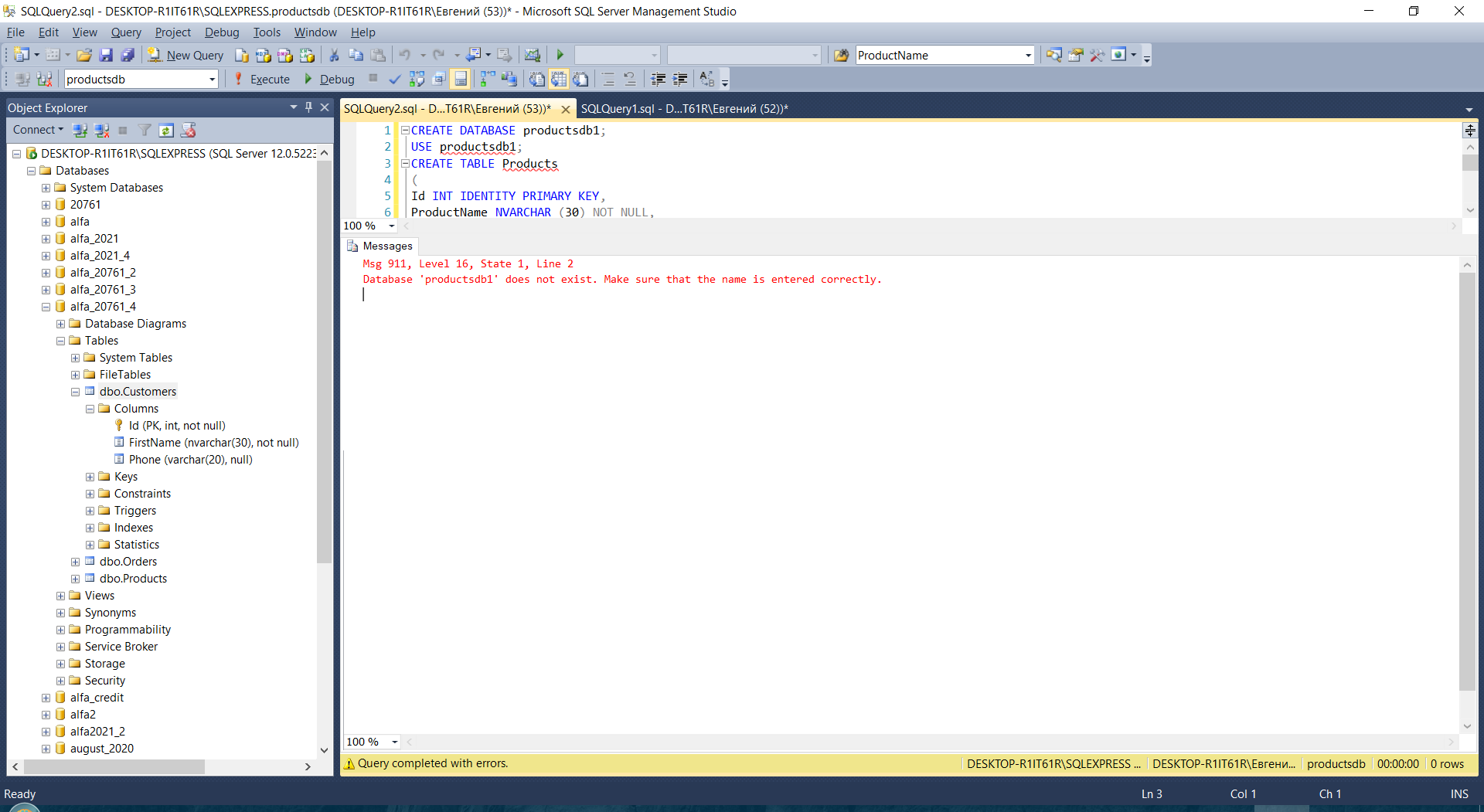
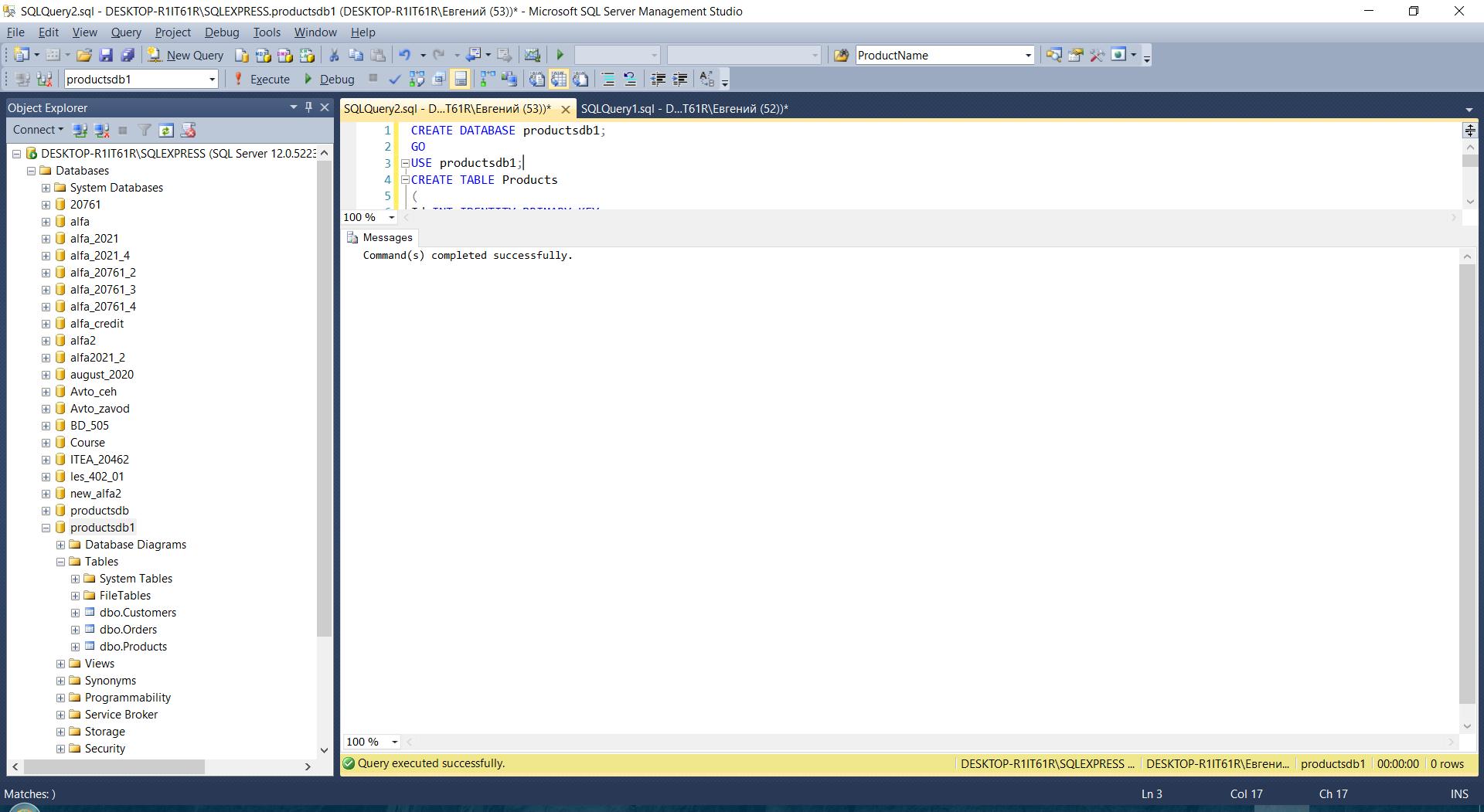


Рис. Реакція обробки запиту з та без GO

# Основи T-SQL. DML

## Додавання даних. команда Insert

Для додавання даних застосовується команда INSERT, яка має наступний формальний синтаксис:

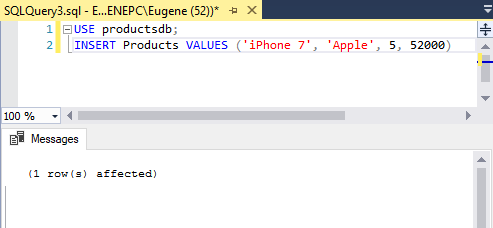
|  |
| --- |
| INSERT [INTO] ім'я\_таблиці [(список\_стовбців)] VALUES (Значення1, значення2, ... значенняN) |

Спочатку йде вираз INSERT INTO, потім в дужках можна вказати список стовпців через кому, в які треба додавати дані, і в кінці після слова VALUES дужках перераховують додаються для стовпців значення.

Додамо в Products один рядок за допомогою команди INSERT:

|  |
| --- |
| INSERT Products VALUES ( 'IPhone 7', 'Apple', 5, 5200) |

Після вдалого виконання в SQL Server Management Studio в поле повідомлень повинно з'явитися повідомлення "1 row (s) affected":



Варто враховувати, що значення для стовпців в дужках після ключового слова VALUES передаються по порядку їх оголошення. Наприклад, в вираженні CREATE TABLE вище можна побачити, що першим стовпцем йде Id. Але так як для нього заданий атрибут IDENTITY, то значення цього стовпця автоматично генерується, і його можна не вказувати. Другий стовпець представляє ProductName, тому перше значення - рядок "iPhone 7" буде передано саме цього стовпцю. Друге значення - рядок "Apple" буде передана третій стовпці Manufacturer і так далі. Тобто значення передаються стовпцями наступним чином:

* ProductName: 'iPhone 7'
* Manufacturer: 'Apple'
* ProductCount: 5
* Price: 52000

Також при введенні значень можна вказати безпосередні стовпці, в які будуть додаватися значення:

|  |
| --- |
| INSERT INTO Products (ProductName, Price, Manufacturer)  VALUES ( 'IPhone 6S', 4100, 'Apple') |

Тут значення вказується тільки для трьох стовпців. Причому тепер значення передаються в порядку проходження стовпців:

* ProductName: 'iPhone 6S'
* Manufacturer: 'Apple'
* Price: 4100

Для невказаних стовпців (в даному випадку ProductCount) буде додаватися значення за замовчуванням, якщо заданий атрибут DEFAULT, або значення NULL. При цьому не вказані стовпці повинні допускати значення NULL або мати атрибут DEFAULT.

Також ми можемо додати відразу кілька рядків:

|  |
| --- |
| INSERT INTO Products VALUES  ( 'IPhone 6', 'Apple', 3, 3600),  ( 'Galaxy S8', 'Samsung', 2, 4600),  ( 'Galaxy S8 Plus', 'Samsung', 1, 5600) |

В даному випадку в таблицю будуть додані три рядки.

Також при додаванні ми можемо вказати, щоб для стовпця використовувалося значення за замовчуванням за допомогою ключового слова DEFAULT або значення NULL:

|  |
| --- |
| INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)  VALUES ( 'Mi6', 'Xiaomi', DEFAULT, 2800) |

В даному випадку для стовпця ProductCount буде використано значення за замовчуванням (якщо воно встановлено, якщо його немає - то NULL).

Якщо всі стовпці мають атрибут DEFAULT, що визначає значення за замовчуванням, або допускають значення NULL, то можна для всіх стовпців вставити значення за замовчуванням:

|  |
| --- |
| INSERT INTO Products  DEFAULT VALUES |

Але якщо брати таблицю Products, то подібна команда завершиться з помилкою, так як кілька полів не мають атрибута DEFAULT і при цьому не допускають значення NULL.

## Вибірка даних. команда SELECT

Для отримання даних застосовується команда SELECT. У спрощеному вигляді вона має наступний синтаксис:

|  |
| --- |
| SELECT список\_стовбців FROM ім'я\_таблиці |

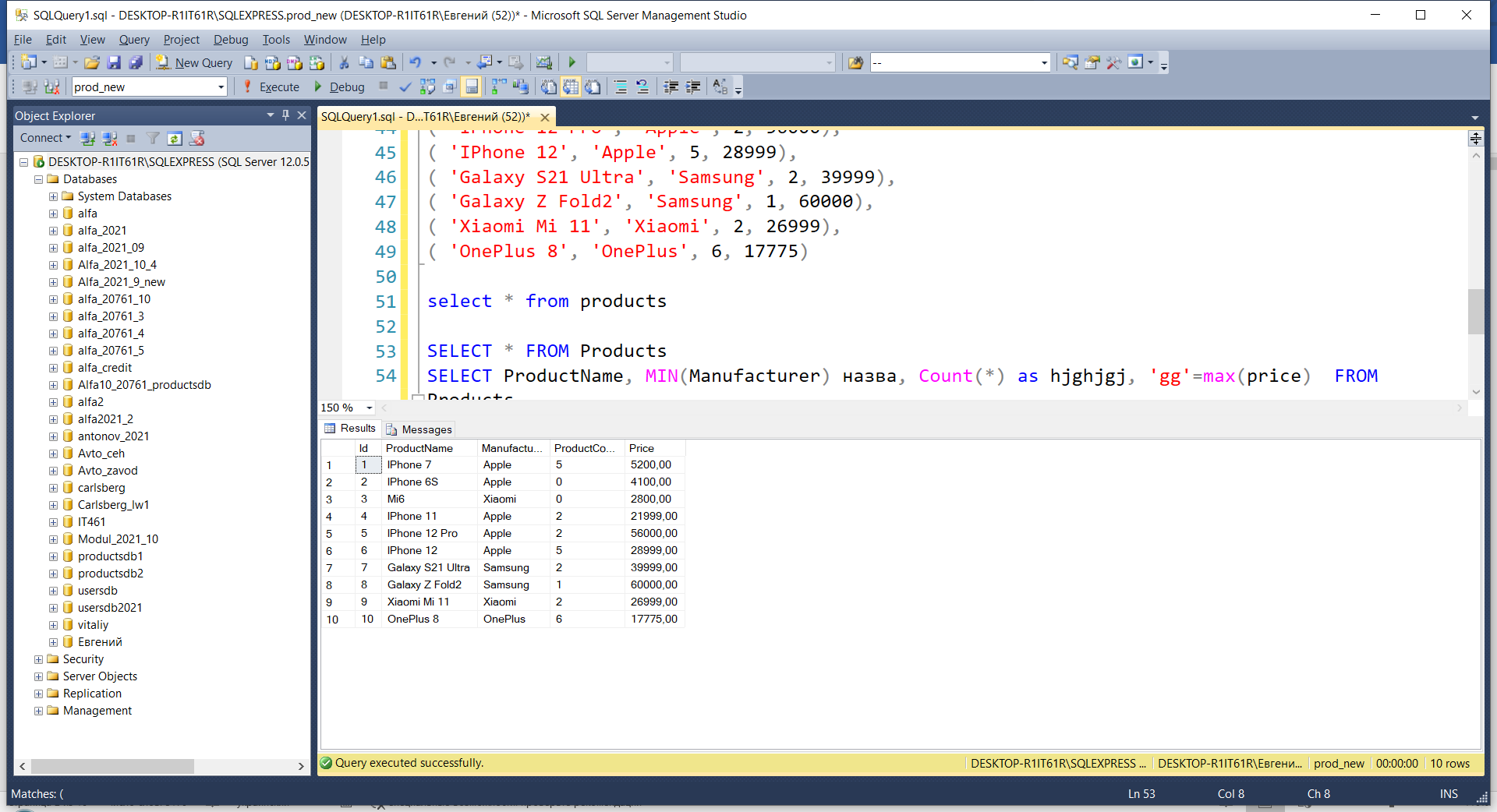
Наприклад, нехай раніше була створена таблиця Products, і в неї додані деякі початкові дані:

|  |
| --- |
| INSERT INTO Products  VALUES ( 'IPhone 11', 'Apple', 2, 21999),  ( 'IPhone 12 Pro', 'Apple', 2, 56000),  ( 'IPhone 12', 'Apple', 5, 28999),  ( 'Galaxy S21 Ultra', 'Samsung', 2, 39999),  ( 'Galaxy Z Fold2', 'Samsung', 1, 60000),  ( 'Xiaomi Mi 11', 'Xiaomi', 2, 26999),  ( 'OnePlus 8', 'OnePlus', 6, 17775) |

Отримаємо всі об'єкти з цієї таблиці:

|  |
| --- |
| SELECT \* FROM Products |

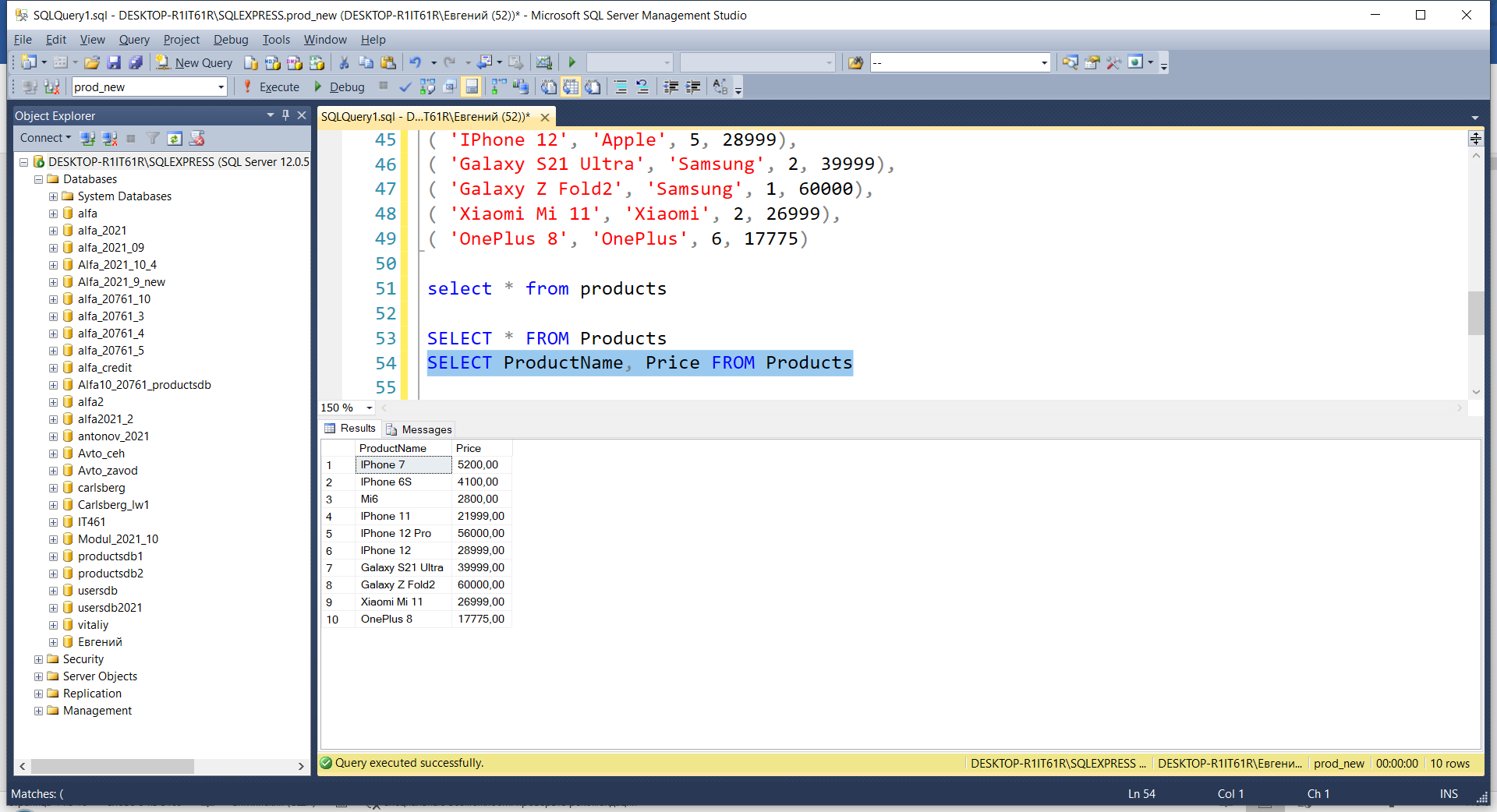
Символ зірочка \* вказує, що нам треба отримати всі стовпці.



Отримання всіх стовпців за допомогою символу зірочки \* вважається не дуже гарною практикою, так як, як правило, не всі стовпці бувають потрібні. І більш оптимальний підхід полягає у прописуванні всіх необхідних стовпців після команди SELECT. Виняток становить той випадок, коли треба отримати дані за абсолютно всіма стовпцями таблиці. Також використання символу \* може бути виправдано в таких ситуаціях, коли точно не відомі назви стовпців.

Якщо нам треба отримати дані не по всіх, а по конкретним стовпцям, то тоді всі ці специфікації стовпців перераховуються через кому після SELECT:

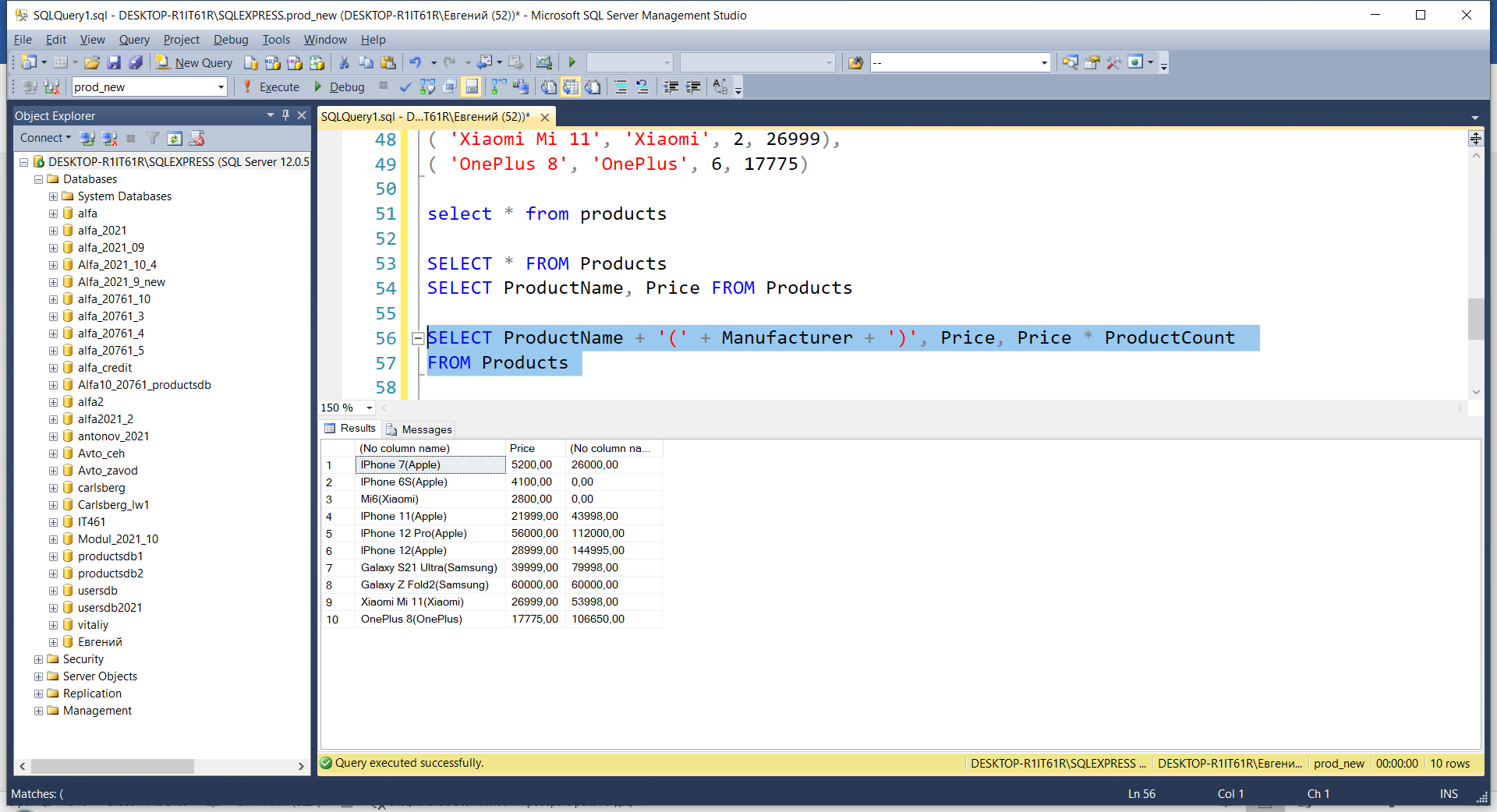
|  |
| --- |
| SELECT ProductName, Price FROM Products |



Специфікація стовпця необов'язково повинна представляти його назву. Це може бути будь-який вираз, наприклад, результат арифметичної операції. Виконаємо наступний запит:

|  |
| --- |
| SELECT ProductName + '(' + Manufacturer + ')', Price, Price \* ProductCount  FROM Products |

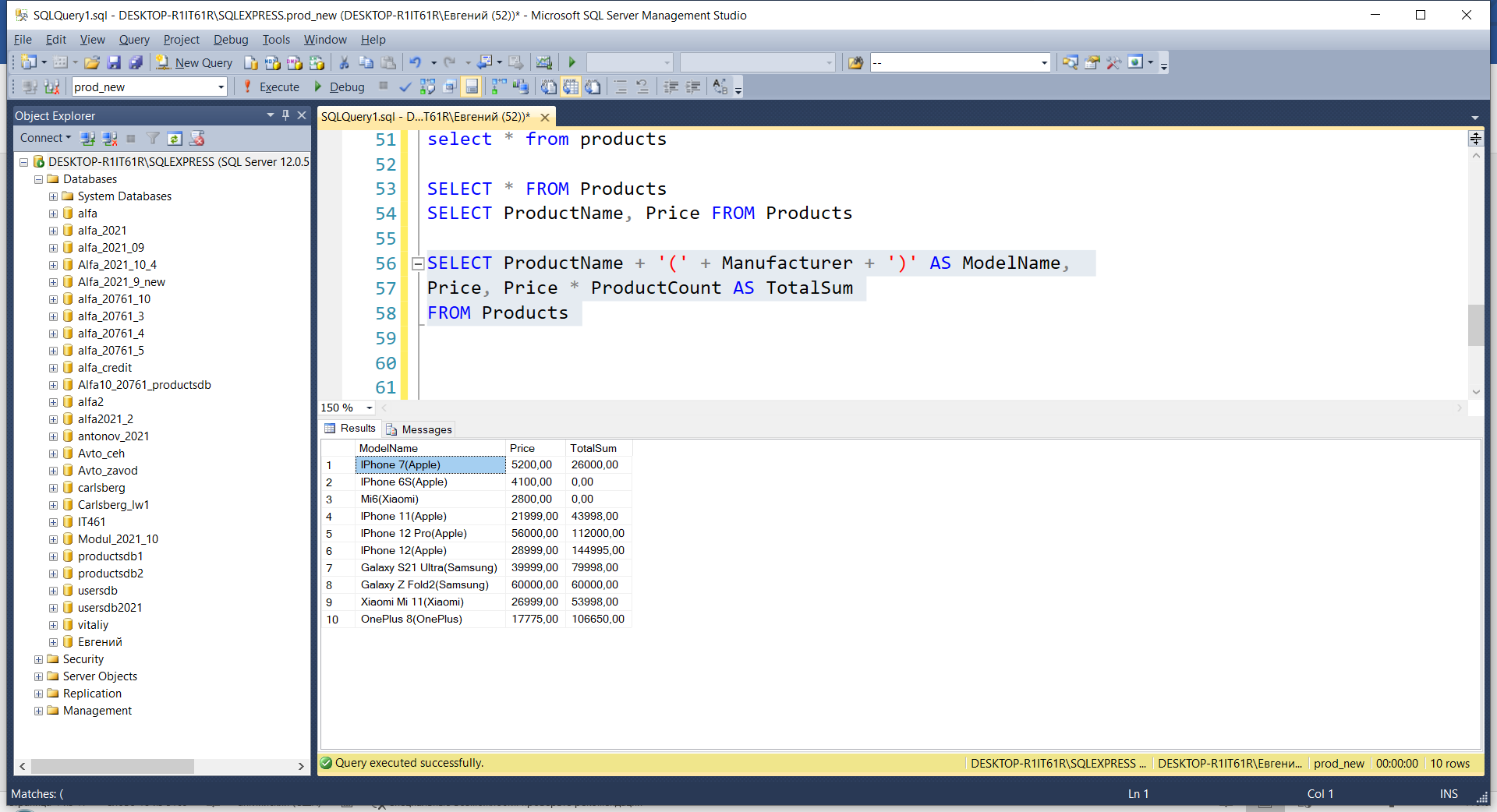
Тут при вибірці будуть створюватися три стовпці. Перший стовпець представляє результат об'єднання двох стовпців ProductName і Manufacturer. Другий стовпець - стандартний стовпець Price. А третій стовпець представляє значення стовпця Price, помножене на значення стовпця ProductCount.



За допомогою оператора AS можна змінити назву вихідного стовпця або визначити його псевдонім:

|  |
| --- |
| SELECT ProductName + '(' + Manufacturer + ')' AS ModelName,  Price, Price \* ProductCount AS TotalSum  FROM Products |

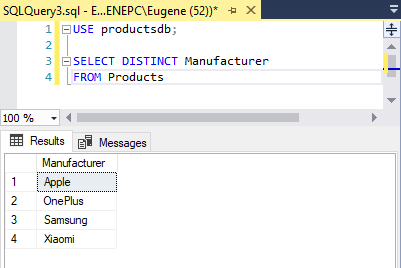
В даному випадку результатом вибірки є дані по 3-м стовпцях. Перший стовпець ModelName об'єднує стовпці ProductName і Manufacturere, другий пропонує стандартний стовпець Price. Третій стовпець TotalSum зберігає твір стовпців ProductCount і Price. При цьому, як у випадку зі стовпцем Price, необов'язково визначати назву результуючого стовпця за допомогою AS.



### DISTINCT

Оператор DISTINCT дозволяє вибрати унікальні рядки. Наприклад, в нашому випадку в таблиці може бути по кілька товарів від одних і тих же виробників. Виберемо всіх виробників:

|  |
| --- |
| SELECT DISTINCT Manufacturer  FROM Products |



В даному випадку критерієм розмежування рядків є стовпець Manufacturer. Тому в результуючої вибіркою будуть тільки унікальні значення Manufacturer. І якщо, наприклад, в базі даних є два товари з виробником Apple, то ця назва буде зустрічатися в результуючої вибіркою тільки один раз.

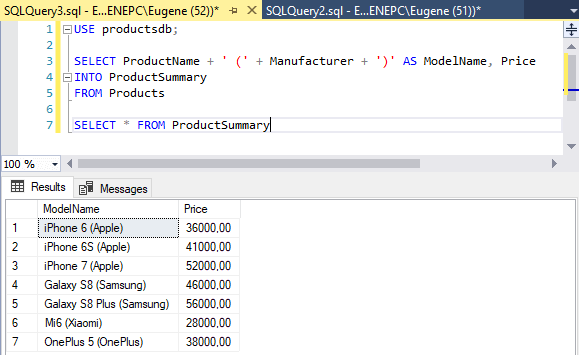
### Вибірка з додаванням

#### SELECT INTO

Вираз SELECT INTO дозволяє вибрати з однієї таблиці деякі дані в іншу таблицю, при цьому друга таблиця створюється автоматично. наприклад:

|  |
| --- |
| SELECT ProductName + '(' + Manufacturer + ')' AS ModelName, Price  INTO ProductSummary  FROM Products  SELECT \* FROM ProductSummary |

Після виконання цієї команди в базі даних буде створена ще одна таблиця ProductSummary, яка буде мати два стовпці ModelName і Price, а дані для цих стовпців будуть взяті з таблиці Products:



При виконанні цієї команди таблиця, в яку йде вибірка (в даному випадку ProductSummary), не повинна існувати в базі даних.

Але, припустимо, ми потім вирішили додати всі дані з таблиці Products в уже існуючу таблицю ProductSummary. В цьому випадку можна знову ж використовувати команду INSERT:

|  |
| --- |
| INSERT INTO ProductSummary  SELECT ProductName + '(' + Manufacturer + ')' AS ModelName, Price  FROM Products |

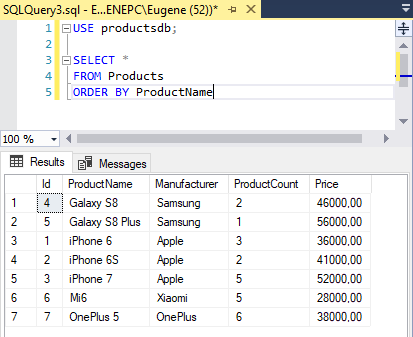
Тут додаються значення фактично представляють результат вибірки з таблиці Products.

## Сортування. ORDER BY

Оператор ORDER BY дозволяє впорядкувати видобувні значення за певним стовпцем:

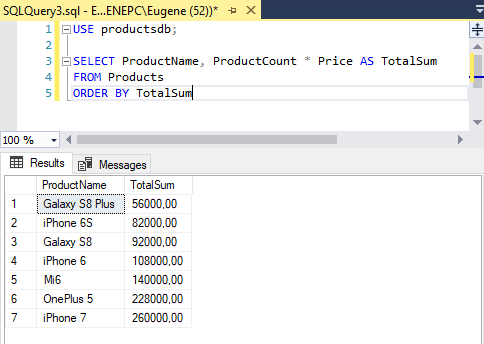
|  |
| --- |
| SELECT \*  FROM Products  ORDER BY ProductName |

В даному випадку рядки сортуються за зростанням значення стовпця ProductName:



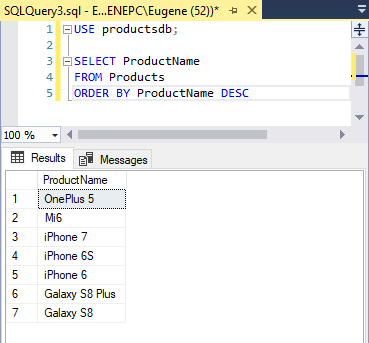
Сортування також можна проводити за псевдонімом стовпчика, який визначається за допомогою оператора AS:

|  |
| --- |
| SELECT ProductName, ProductCount \* Price AS TotalSum  FROM Products  ORDER BY TotalSum |



За замовчуванням застосовується сортування за зростанням. За допомогою додаткового оператора DESC можна задати сортування по спадаючій.

|  |
| --- |
| SELECT ProductName  FROM Products  ORDER BY ProductName DESC |



За замовчуванням замість DESC використовується оператор ASC:

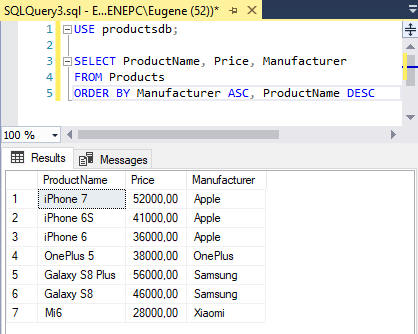
|  |
| --- |
| SELECT ProductName  FROM Products  ORDER BY ProductName ASC |

Якщо необхідно впорядкувати відразу за кількома стовпцями, то всі вони перераховуються після оператора ORDER BY:

|  |
| --- |
| SELECT ProductName, Price, Manufacturer  FROM Products  ORDER BY Manufacturer, ProductName |

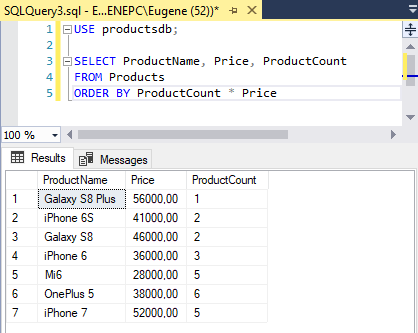
У цьому випадку спочатку рядки упорядковано відповідно до колонки Manufacturer по зростанню. Потім якщо є два рядки, в яких стовпець Manufacturer має однакове значення, то вони сортуються по стовпцю ProductName також по зростанню. Але знову ж таки за допомогою ASC і DESC можна окремо для різних стовпців визначити сортування по зростанню і зменшенням:

|  |
| --- |
| SELECT ProductName, Price, Manufacturer  FROM Products  ORDER BY Manufacturer ASC, ProductName DESC |



Як критерій сортування також можна використовувати складно вираз на основі стовпців:

|  |
| --- |
| SELECT ProductName, Price, ProductCount  FROM Products  ORDER BY ProductCount \* Price |

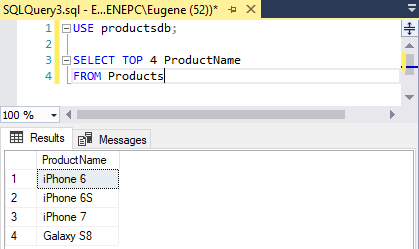


## Виведення діапазону рядків

**Оператор TOP**

Оператор TOP дозволяє вибрати певну кількість рядків з таблиці:

|  |
| --- |
| SELECT TOP 4 ProductName  FROM Products |



Додатковий оператор PERCENT дозволяє вибрати процентне кількість рядків з таблиці. Наприклад, виберемо 75% рядків:

|  |
| --- |
| SELECT TOP 75 PERCENT ProductName FROM Products |

**OFFSET і FETCH**

Оператор TOP дозволяє витягти певну кількість рядків, починаючи з початку таблиці. Для вилучення набору рядків з будь-якого місця, застосовуються оператори OFFSET і FETCH. Важливо, що ці оператори застосовуються тільки в відсортованому наборі даних після висловлення ORDER BY.

|  |
| --- |
| ORDER BY вираз  OFFSET зсув\_відносно\_початку {ROW | ROWS}  [FETCH {FIRST | NEXT} кількість\_рядків {ROW | ROWS} ONLY] |

Наприклад, виберемо всі рядки, починаючи з третьої:

|  |
| --- |
| SELECT \* FROM Products  ORDER BY Id  OFFSET 2 ROWS; |

Число після ключового слова OFFSET вказує, скільки рядків необхідно пропустити.



Тепер виберемо тільки три рядки, починаючи з третьої:

|  |
| --- |
| SELECT \* FROM Products  ORDER BY Id  OFFSET 2 ROWS  FETCH NEXT 3 ROWS ONLY; |

Після оператора FETCH вказується ключове слово FIRST або NEXT і потім вказується кількість рядків, яке треба отримати.



Дана комбінація операторів, як правило, використовується для посторінкового навігації, коли необхідно отримати певну сторінку з даними.

## Фільтрація. WHERE

Для фільтрації в команді SELECT застосовується оператор WHERE. Після цього оператора ставиться умова, якому повинна відповідати рядок:

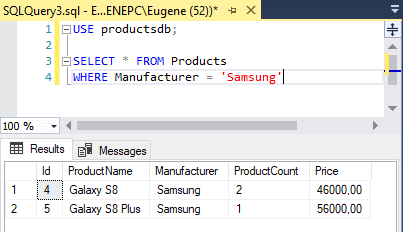
|  |
| --- |
| WHERE умова |

Якщо умова є істинною, то рядок потрапляє в результуючу вибірку. Як можна використовувати операції порівняння. Ці операції порівнюють два вирази. В T-SQL можна застосовувати такі операції порівняння:

* **=**: Порівняння на рівність (на відміну від сі-подібних мов в T-SQL для порівняння на рівність використовується один знак дорівнює)
* **<>**: Порівняння на нерівність (також допускається **!=**)
* **<**: менше ніж
* **>**: більше ніж
* **!<**: не менше ніж
* **!>**: не більш ніж
* **<=**: Менше ніж або дорівнює
* **>=**: Більше ніж або дорівнює

Наприклад, знайдемо всіх товари, виробником яких є компанія Samsung:

|  |
| --- |
| SELECT \* FROM Products  WHERE Manufacturer = 'Samsung' |



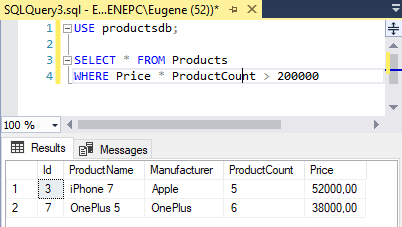
Варто відзначити, що в даному випадку регістр не має значення, і ми могли б використовувати для пошуку і рядок "Samsung", і "SAMSUNG", і "samsung". Всі ці варіанти давали б еквівалентний результат вибірки.

Інший приклад - знайдемо всі товари, у яких ціна більше 45000:

|  |
| --- |
| SELECT \* FROM Products  WHERE Price> 45000 |

Як умова можуть використовуватися і більш складні вирази. Наприклад, знайдемо всі товари, у яких сукупна вартість понад 200 000:

|  |
| --- |
| SELECT \* FROM Products  WHERE Price \* ProductCount> 200000 |



**Логічні оператори**

Для об'єднання декількох умов в одне можуть використовуватися логічні оператори. В T-SQL є такі логічні оператори:

* **AND**: Операція логічного І. Вона об'єднує два вирази:

|  |
| --- |
| вираз1 AND вираз2 |

* Тільки якщо обидва цих вираз одночасно істина, то і загальне умова оператора AND також буде істина. Тобто якщо і перша умова істина, і друга.
* **OR**: Операція логічного АБО. Вона також об'єднує два вирази:

|  |
| --- |
| вираз1 OR вираз2 |

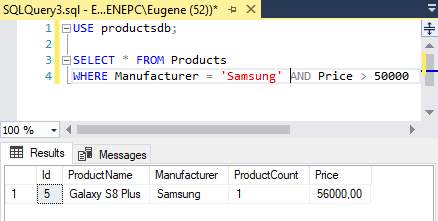
* Якщо хоча б одне з цих виразів істина, то загальне умова оператора OR також буде істина. Тобто якщо або перша умова істина, або друга.
* **NOT**: Операція логічного заперечення. Якщо вираз в цій операції помилковий, то загальна умова істина.

|  |
| --- |
| NOT вираз |

Якщо ці оператори зустрічаються в одному вираpі, то спочатку виконується NOT, потім AND і в кінці OR.

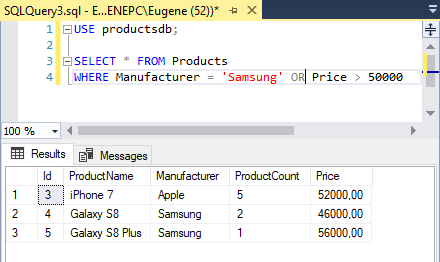
Наприклад, виберемо усі товари, у яких виробник Samsung і одночасно ціна більше 50000:

|  |
| --- |
| SELECT \* FROM Products  WHERE Manufacturer = 'Samsung' AND Price> 50000 |



Тепер змінимо оператор на OR. Тобто виберемо усі товари, у яких або виробник Samsung, або ціна більше 50000:

|  |
| --- |
| SELECT \* FROM Products  WHERE Manufacturer = 'Samsung' OR Price> 50000 |



Застосування оператора NOT - виберемо усі товари, у яких виробник не Samsung:

|  |
| --- |
| SELECT \* FROM Products  WHERE NOT Manufacturer = 'Samsung' |



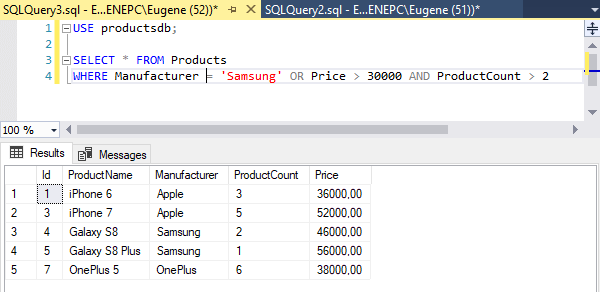
Але в більшості випадків цілком можна обійтися без оператора NOT. Так, в попередній приклад ми можемо переписати таким чином:

|  |
| --- |
| SELECT \* FROM Products  WHERE Manufacturer <> 'Samsung' |

Також в одній команді SELECT можна використовувати відразу кілька операторів:

|  |
| --- |
| SELECT \* FROM Products  WHERE Manufacturer = 'Samsung' OR Price> 30000 AND ProductCount> 2 |

Так як оператор AND має більш високий пріоритет, то спочатку буде виконуватися підвираз Price> 30000 AND ProductCount> 2, І тільки потім оператор OR. Тобто тут вибираються товари, коториех на складі більше 2 і у яких одночасно ціна більше 30000, або ті товари, виробником яких є Samsung.



За допомогою дужок ми також можемо перевизначити порядок операцій:

|  |
| --- |
| SELECT \* FROM Products  WHERE (Manufacturer = 'Samsung' OR Price> 30000) AND ProductCount> 2 |

**IS NULL**

Ряд стовпців може допускати значення NULL. Це значення не еквівалентне порожній рядку ''. NULL являє повна відсутність будь-якого значення. І для перевірки на наявність подібного значення застосовується оператор IS NULL.

Наприклад, виберемо усі товари, у яких не встановлено поле ProductCount:

|  |
| --- |
| SELECT \* FROM Products  WHERE ProductCount IS NULL |

Якщо, навпаки, необхідно отримати рядки, у яких поле ProductCount не дорівнює NULL, то можна використовувати оператор NOT:

|  |
| --- |
| SELECT \* FROM Products  WHERE ProductCount IS NOT NULL |

## Оператори фільтрації

**оператор IN**

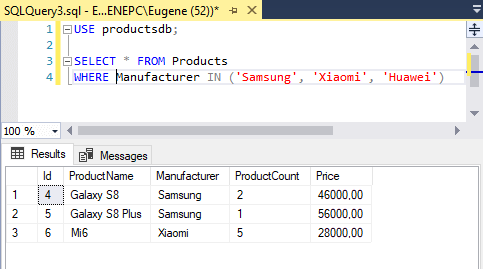
Оператор IN дозволяє визначити набір значень, які повинні мати стовпці:

|  |
| --- |
| WHERE вираз [NOT] IN (Вираз) |

Вираз в дужках після IN визначає набір значень. Цей набір може обчислюватися динамічно на підставі, наприклад, ще одного запиту, або це можуть бути константні значення.

Наприклад, виберемо товари, у яких виробник або Samsung, або Xiaomi, або Huawei:

|  |
| --- |
| SELECT \* FROM Products  WHERE Manufacturer IN ( 'Samsung', 'Xiaomi', 'Huawei') |



Ми могли б всі ці значення перевірити і через оператор OR:

|  |
| --- |
| SELECT \* FROM Products  WHERE Manufacturer = 'Samsung' OR Manufacturer = 'Xiaomi' OR Manufacturer = 'Huawei' |

Але використання оператора IN набагато зручніше, особливо якщо подібних значень дуже багато.

За допомогою оператора NOT можна знайти всі рядки, які, навпаки, не відповідають набору значень:

|  |
| --- |
| SELECT \* FROM Products  WHERE Manufacturer NOT IN ( 'Samsung', 'Xiaomi', 'Huawei') |

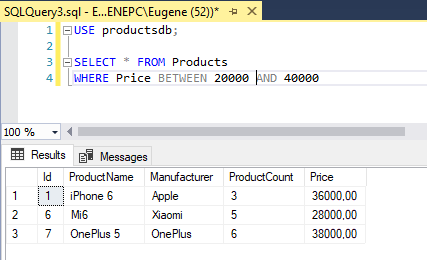
**оператор BETWEEN**

Оператор BETWEEN визначає діапазон значень за допомогою початкового і кінцевого значення, яким має відповідати вираз:

|  |
| --- |
| WHERE вираз [NOT] BETWEEN початкове\_значення AND кінцеве\_значення |

Наприклад, отримаємо усі товари, у яких ціна від 20 000 до 40 000 (початкове і кінцеве значення також включаються в діапазон):

|  |
| --- |
| SELECT \* FROM Products  WHERE Price BETWEEN 20000 AND 40000 |



Якщо треба, навпаки, вибрати ті рядки, які не потрапляють в даний діапазон, то застосовується оператор NOT:

|  |
| --- |
| SELECT \* FROM Products  WHERE Price NOT BETWEEN 20000 AND 40000 |

Також можна використовувати більш складні вирази. Наприклад, отримаємо товари, запаси яких на певну суму (ціна \* кількість):

|  |
| --- |
| SELECT \* FROM Products  WHERE Price \* ProductCount BETWEEN 100000 AND 200000 |

**Оператор LIKE**

Оператор LIKE приймає шаблон рядки, якому має відповідати вираз.

|  |
| --- |
| WHERE вираз [NOT] LIKE шаблон\_рядку |

Для визначення шаблону можуть застосовуватися ряд спеціальних символів підстановки:

* **%**: Відповідає будь-якому рядку, який може мати будь-яку кількість символів, при цьому рядок може і не містити жодного символу
* **\_**: Відповідає будь-якому одиночному символу
* **[]**: Відповідає одному символу, який вказаний в квадратних дужках
* **[-]**: Відповідає одному символу з певного діапазону
* **[^]**: Відповідає одному символу, який не вказано після символу ^

Деякі приклади використання підстановок:

* WHERE ProductName LIKE 'Galaxy%'

Відповідає таким значенням як "Galaxy Ace 2" або "Galaxy S7"

* WHERE ProductName LIKE 'Galaxy S\_'

Відповідає таким значенням як "Galaxy S7" або "Galaxy S8"

* WHERE ProductName LIKE 'iPhone [78]'

Відповідає таким значенням як "iPhone 7" або "iPhone8"

* WHERE ProductName LIKE 'iPhone [6-8]'

Відповідає таким значенням як "iPhone 6", "iPhone 7" або "iPhone8"

* WHERE ProductName LIKE 'iPhone [^7]%'

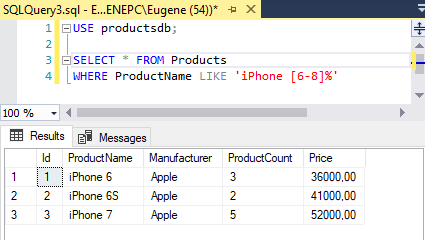
Відповідає таким значенням як "iPhone 6", "iPhone 6S" або "iPhone 8". Але не відповідає значенням "iPhone 7" і "iPhone 7S"

* WHERE ProductName LIKE 'iPhone [^1-6]%'

Відповідає таким значенням як "iPhone 7", "iPhone 7S" і "iPhone 8". Але не відповідає значенням "iPhone 5", "iPhone 6" і "iPhone 6S"

Застосуємо оператор LIKE:

|  |
| --- |
| SELECT \* FROM Products  WHERE ProductName LIKE 'IPhone [6-8]%' |



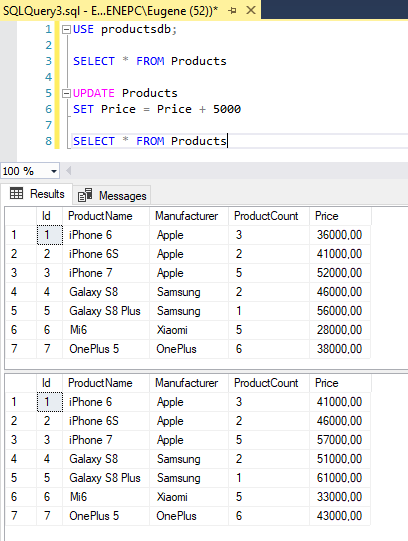
## Команда UPDATE

Для зміни вже наявних рядків в таблиці застосовується команда UPDATE. Вона має наступний формальний синтаксис:

|  |
| --- |
| UPDATE ім'я\_таблиці  SET стовбчик1 = значення1, стовбчик2 = значення2, ... стовбчикN = значенняN  [FROM вибірка AS псевдонім\_вибірки]  [WHERE умова\_оновлення] |

Наприклад, збільшимо у всіх товарів ціну на 5000:

|  |
| --- |
| UPDATE Products  SET Price = Price +5000 |



Використовуємо критерій, і змінимо назву виробника з "Samsung" на "Samsung Inc.":

|  |
| --- |
| UPDATE Products  SET Manufacturer = 'Samsung Inc.'  WHERE Manufacturer = 'Samsung' |

Більш складний запит - замінимо у поля Manufacturer значення "Apple" на "Apple Inc." в перших 2 рядках:

|  |
| --- |
| UPDATE Products  SET Manufacturer = 'Apple Inc.'  FROM  (SELECT TOP 2 Products.Id FROM Products WHERE Manufacturer = 'Apple') AS Selected  WHERE Products.Id = Selected.Id |

За допомогою підзапиту після ключового слова FROM проводиться вибірка перших двох рядків, в яких Manufacturer = 'Apple'. Для цієї вибірки буде визначено псевдонім Selected. Ім'я користувача вказується після оператора AS.

Далі йде умова поновлення Products.Id = Selected.Id. Тобто фактично ми маємо справу з двома таблицями - Products і Selected (яка є похідною від Products). У Selected знаходиться дві перших рядки, в яких Manufacturer = 'Apple'. У Products - взагалі всі рядки. І оновлення проводиться тільки для тих рядків, які є в вибірці Selected. Тобто якщо в таблиці Products десятки товарів з виробником Apple, то оновлення торкнеться тільки двох перших з них.

## Видалення даних. команда DELETE

Для видалення застосовується команда DELETE:

|  |
| --- |
| DELETE [FROM] ім'я\_таблиці  WHERE умова\_видалення |

Наприклад, видалимо рядки, у яких id дорівнює 9:

|  |
| --- |
| DELETE Products  WHERE Id = 9 |

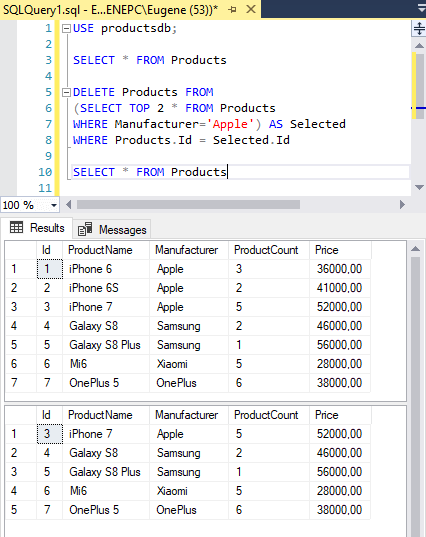
Або видалимо всі товари, виробником яких є Xiaomi і які мають ціну менше 15000:

|  |
| --- |
| DELETE Products  WHERE Manufacturer = 'Xiaomi' AND Price <15000 |

Більш складний приклад - видалимо перші два товари, у яких виробник - Apple:

|  |
| --- |
| DELETE Products FROM  (SELECT TOP 2 \* FROM Products  WHERE Manufacturer = 'Apple]') AS Selected  WHERE Products.Id = Selected.Id |

Після першого оператора FROM йде вибірка двох рядків з таблиці Products. Цією вибірці призначається псевдонім Selected за допомогою оператора AS. Далі встановлюємо умова, що якщо Id в таблиці Products має те ж значення, що і Id у вибірці Selected, то рядок видаляється.



Якщо необхідно зовсім видалити всі рядки незалежно від умови, то умова можна не вказувати:

|  |
| --- |
| DELETE Products |

## **Завдання**

Перевірити роботу запитів з теоретичної частини.

На основі отриманих таблиць Products, Customers, Orders (запит на початку лабораторної роботи) побудувати запити для вирішення наступних завдань:

1. Побудувати запит INSERT для додавання 2 нових користувачів.

2. Побудувати запит INSERT для додавання 2 нових товарів.

3. Побудувати запит INSERT для додавання 3 нових товарів без вказання кількості.

4. Побудувати запит INSERT для додавання 2 нових товарів фірми Apple.

5. Побудувати запит INSERT для додавання 3 нових товарів фірми Apple без вказання кількості.

6. Побудувати запит INSERT для додавання 5 нових замовлень (таблиця Orders) за участю нових товарів (вставлених в завданні 2 та 3) і нових користувачів (вставлених в завданні 1).

7. Вивести два продукти з найнижчими цінами.

8. Вивести другу п'ятірку продуктів з найвищою ціною.

9. Вивести продукти, число залишків яких більше 4.

10. Вивести продукти фірми Samsung з ціною вище 50000

11. Вивести позиції з вартістю залишків більше 200000

12. Вивести всі товари, у яких в найменуванні є 'Plus'

13. Вивести всі товари, у яких в найменуванні немає цифр 7 і 8

14. Вивести всі товари, найменування фірми яких не мають букву 'a'

15. \*Збільшити ціну останніх 3 створених продуктів на 2000 (без підзапиту)

16. \* Видалити останній створений продукт (без підзапиту)

17. \* Вивести таблицю з одним стовпцем, в якому дані будуть в наступному форматі: На складі є N продукції під назвою M фірми F (N, M, F - повинні підставитися з таблиці) (\* - використовувати Google або матеріали лабораторної роботи № 4)