**Лабораторна робота № 12. Віконні функції**

## Інструкція OVER в Transact-SQL

**OVER**- це інструкція T-SQL, яка визначає вікно для застосування віконної функції. «Вікно» в Microsoft SQL Server - це контекст, в якому працює функція з певним набором рядків, що відносяться до поточного рядка.

**Віконна функція**- це функція, яка відповідно працює з вікном, тобто набором рядків, і повертає значення на основі деяких обчислень.

Віконні функції використовують в аналітичних звітах, наприклад, для обчислення якихось статистичних значень (суми, ковзаючі середні, проміжні підсумки і так далі) для кожного рядка результуючого набору даних.

Вперше підтримка віконних функцій з'явилася у версії Microsoft SQL Server 2005, в якій була реалізовані базова функціональність. У Microsoft SQL Server 2012 функціонал віконних функцій було розширено, і тепер він з легкістю вирішує багато завдань, які до цього вирішувалися написанням додаткового, в деяких випадках, складного, незрозумілого коду (вкладені запити і т.д.).

### Спрощений синтаксис інструкції OVER

***віконна функція***(Стовпець для обчислень) OVER (

[PARTITION BY стовпець для угруповання]

[ORDER BY стовпець для сортування]

[ROWS або RANGE вираз для обмеження рядків в межах групи]

)

У вираженні для обмеження рядків в групі можна використовувати такі ключові слова:

* ROWS - обмежує рядки;
* RANGE - логічно обмежує рядки за рахунок вказівки діапазону значень відносно до значення поточного рядка;
* UNBOUNDED PRECEDING - вказує, що вікно починається з першого рядка групи. Дана інструкція використовується тільки як початкова точка вікна;
* UNBOUNDED FOLLOWING - за допомогою даної інструкції можна вказати, що вікно закінчується на останньому рядку групи, відповідно, вона може бути вказана тільки як кінцева точка вікна;
* CURRENT ROW - інструкція вказує, що вікно починається або закінчується на поточному рядку, вона може бути задана як початкова або як кінцева точка;
* BETWEEN «межа вікна» AND «межа вікна» - вказує нижню і верхню межу вікна, при цьому верхня межа не може бути менше нижньої межі;
* «Значення» PRECEDING - визначає число рядків перед поточним рядком. Ця інструкція не допускається в реченні RANGE;
* «Значення» FOLLOWING - визначає число рядків після поточного рядка. Якщо FOLLOWING використовується як початкова точка вікна, то кінцева точка повинна бути також вказана за допомогою FOLLOWING. Ця інструкція не допускається в реченні RANGE.

**Примітка!**Щоб вказати вираз для додаткового обмеження рядків (ROWS або RANGE) у вікні повинна бути вказана інструкція ORDER BY.

## Віконні функції в Transact-SQL

В T-SQL віконні функції можна поділити на такі групи:

* Агрегатні функції;
* Ранжирующие функції;
* Функції зміщення;
* Аналітичні функції.

В одній інструкції SELECT з одним пропозицією FROM можна використовувати кілька віконних функцій. Якщо інструкція PARTITION BY не вказана, функція буде обробляти всі рядки результуючого набору. Деякі функції не підтримують інструкцію ORDER BY, ROWS або RANGE.

### Вихідні дані для прикладів

Перед тим як перейти до розгляду використання віконних функцій, давайте спочатку створимо тестові дані, для того щоб виконувати приклади. Таблиця TestTable містить список товарів з деякими характеристиками.

--Созданіе таблиці

CREATE TABLE TestTable (

[ProductId] [INT] IDENTITY (1,1) NOT NULL,

[CategoryId] [INT] NOT NULL,

[ProductName] [VARCHAR] (100) NOT NULL,

[Price] [Money] NULL

)

GO

INSERT INTO TestTable

VALUES (1, 'Клавіатура', 100),

(1, 'Миша', 50),

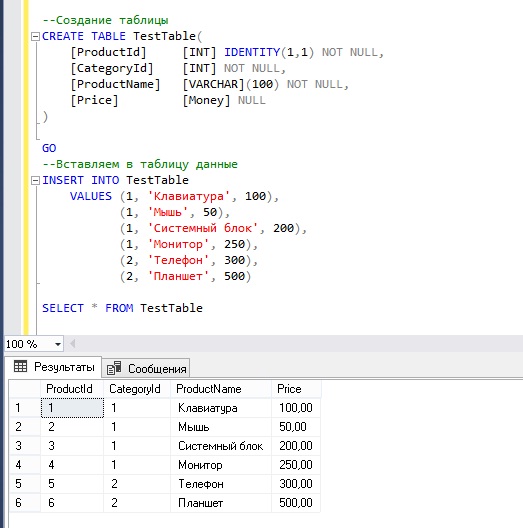
(1, 'Системний блок', 200),

(1, 'Монітор', 250),

(2, 'Телефон', 300),

(2, 'Планшет', 500)

SELECT \* FROM TestTable



### Агрегатні віконні функції

**агрегатні функції**- це функції, які виконують на наборі даних обчислення і повертають підсумкове значення. Агрегатні функції, я думаю, всім відомі - це, наприклад:

* SUM - повертає суму значень в стовпці;
* AVG - визначає середнє значення в стовпці;
* MAX - визначає максимальне значення в стовпці;
* MIN - визначає мінімальне значення в стовпці;
* COUNT - обчислює кількість значень в стовпці (значення NULL не враховуються). Якщо написати COUNT (\*), то будуть враховуватися всі записи, тобто всі рядки. Повертає тип даних INT;
* COUNT\_BIG - працює також як COUNT, тільки повертає тип даних BIGINT.

Зазвичай агрегатні функції використовуються в поєднанні з інструкцією GROUP BY, яка групує рядки, але їх також можна використовувати і без GROUP BY, наприклад, з використанням інструкції OVER, і в даному випадку вони будуть обчислювати значення в певному вікні (наборі даних) для кожної поточної рядки. Це дуже зручно, якщо Вам необхідно отримати якусь величину по відношенню до загальної суми, наприклад.

**Приклад використання агрегатних віконних функцій з інструкцією OVER.**

У цьому прикладі продемонстровано просте застосування деяких агрегатних віконних функцій.

SELECT ProductId, ProductName, CategoryId, Price,

SUM (Price) OVER (PARTITION BY CategoryId) AS [SUM],

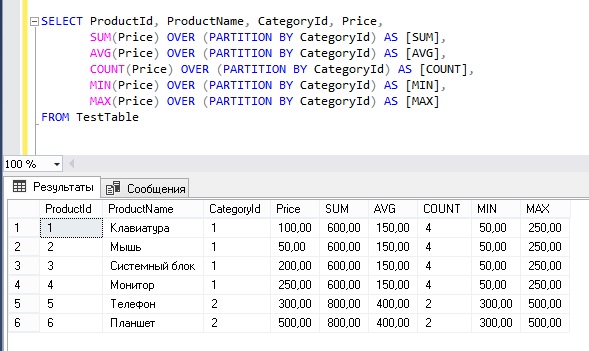
AVG (Price) OVER (PARTITION BY CategoryId) AS [AVG],

COUNT (Price) OVER (PARTITION BY CategoryId) AS [COUNT],

MIN (Price) OVER (PARTITION BY CategoryId) AS [MIN],

MAX (Price) OVER (PARTITION BY CategoryId) AS [MAX]

FROM TestTable



Як бачите, у нас вивелися всі рядки, включаючи стовпчики з агрегованими даними, згрупованими по категорії.

### Ранжуючі віконні функції

**Ранжуючі функції**- це функції, які ранжирують значення для кожного рядка в групі. Наприклад, їх можна використовувати для того, щоб пронумерувати рядки по групах або виставити ранг і скласти рейтинг.

У Microsoft SQL Server існують такі ранжирующие функції:

* **ROW\_NUMBER**- функція повертає номер рядка, використовується для нумерації рядків в секції результуючого набору;
* **RANK**- функція повертає ранг кожного рядка. В даному випадку значення вже аналізуються і, в разі знаходження однакових, повертає однаковий ранг з пропуском наступного;
* **DENSE\_RANK**- функція повертає ранг кожного рядка. Але на відміну від функції RANK, вона для однакових значень повертає ранг, не пропускаючи наступний;
* **NTILE**- це функція, яка повертає результуючий набір, розділений на групи за певною колонки.

## ROW\_NUMBER

ROW\_NUMBER () OVER ([PARTITION BY стовпи угруповання] ORDER BY стовпець сортування)

де, partition by - це не обов'язкове ключове слово, після якого вказується стовпець або стовпці, по яким групувати дані, а order by стовпець для сортування, тобто за даним колонки будуть відсортовані дані, а потім пронумеровані, він вже обов'язковий.

SELECT Productid, ProductName, price, categoryid,

ROW\_NUMBER() over (order by categoryid desc) as [ROW\_NUMBER\_by\_category],

ROW\_NUMBER() over (PARTITION BY categoryid order by categoryid desc) as [ROW\_NUMBER\_category]

FROM TestTable



Перше поле ранжирування тільки з сортуванням по категорії в порядку убування, друге - з угрупованням по категорії і з сортуванням по категорії в порядку убування

## RANK

Приклад без угруповання з сортуванням по категорії (2 записи (категорії 2) і 4 записи категорії один), з угрупованням по категорії і сортуванням за ціною і відміну від row\_number ()

SELECT Productid, ProductName, price, categoryid,

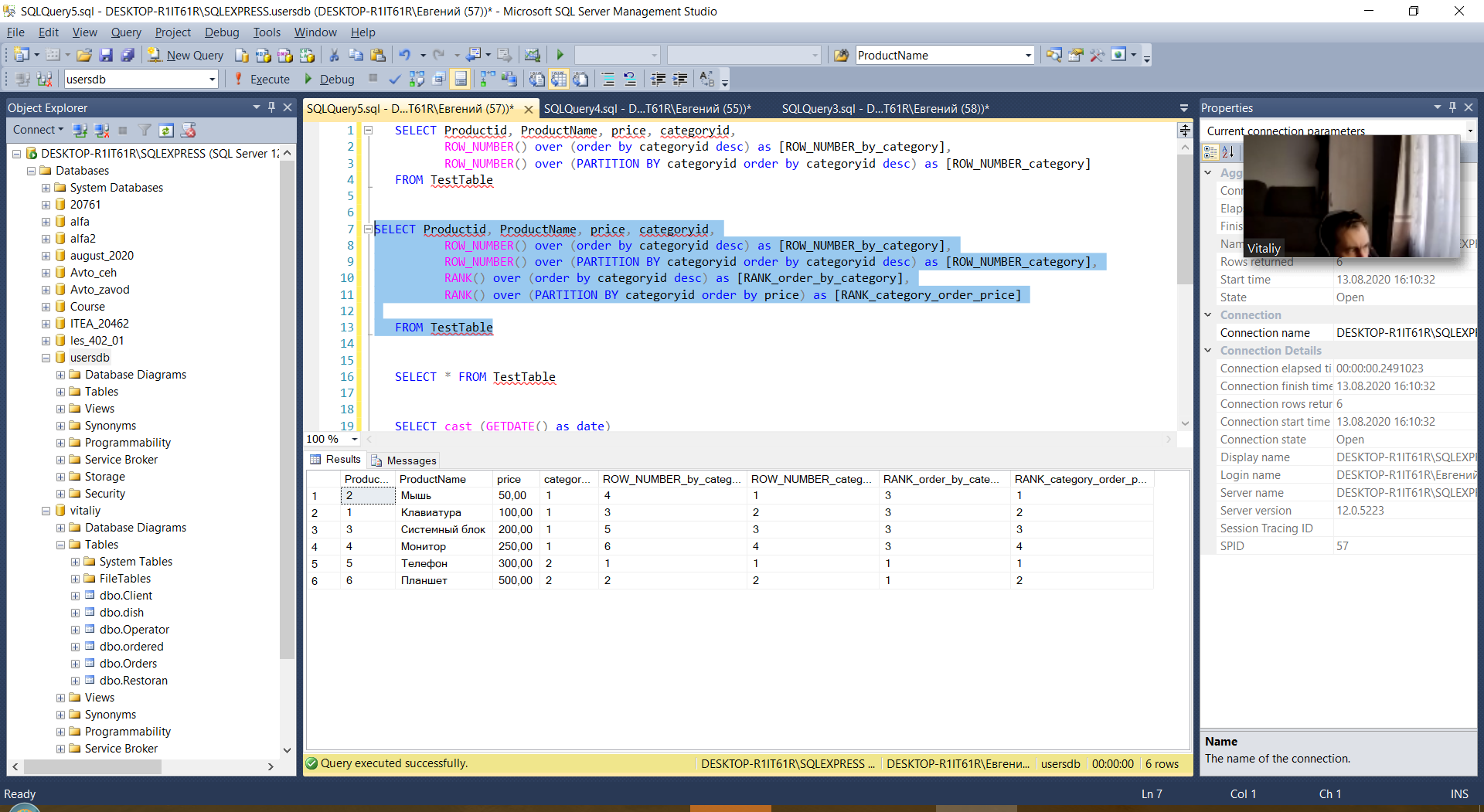
ROW\_NUMBER() over (order by categoryid desc) as [ROW\_NUMBER\_by\_category],

ROW\_NUMBER() over (PARTITION BY categoryid order by categoryid desc) as [ROW\_NUMBER\_category],

RANK() over (order by categoryid desc) as [RANK\_order\_by\_category],

RANK() over (PARTITION BY categoryid order by price) as [RANK\_category\_order\_price]

FROM TestTable



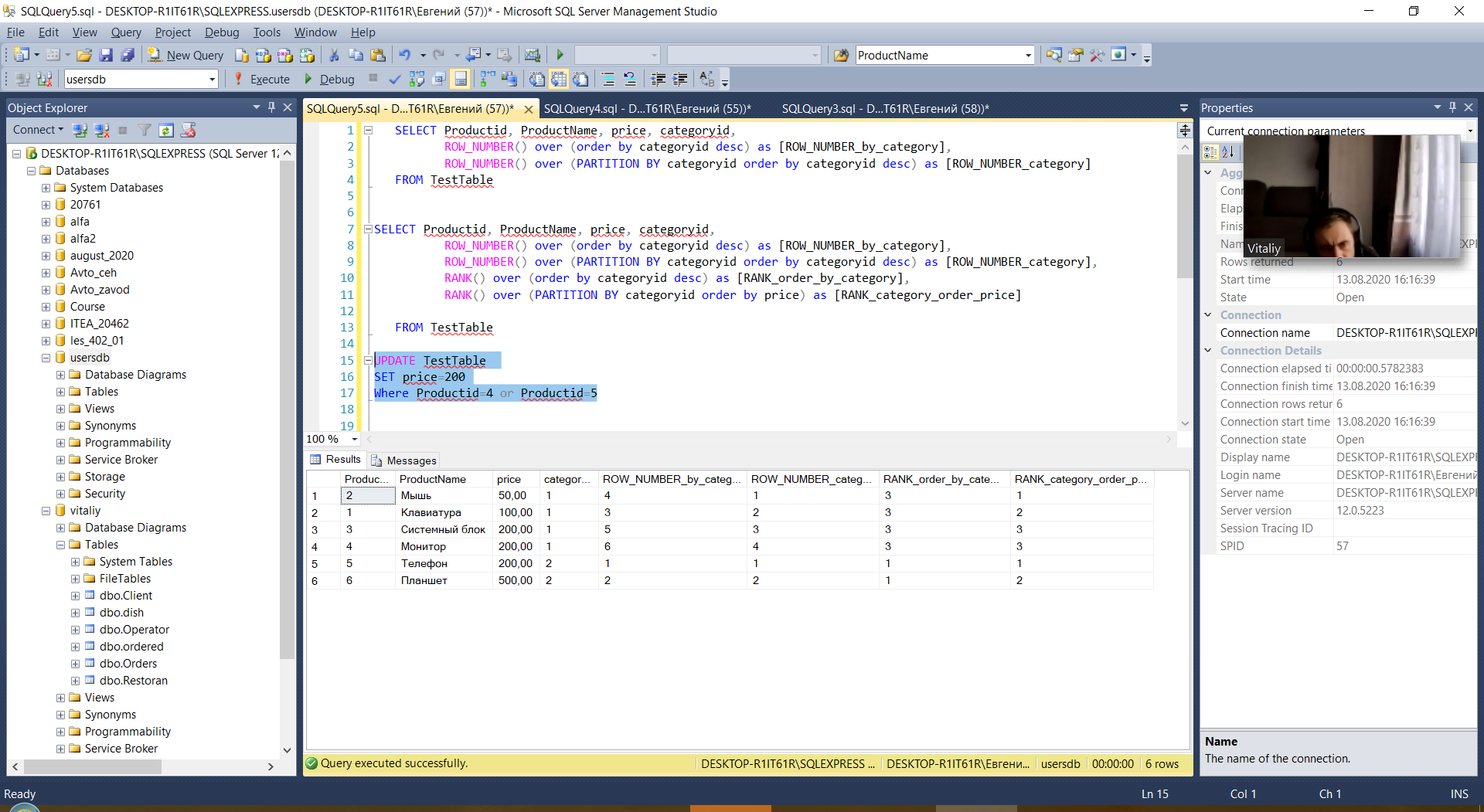
Змінимо ціни у декількох товарів:

UPDATE TestTable

SET price=200

Where Productid=4 or Productid=5

Тепер попередній запит дасть інший результат



## DENSE\_RANK

SELECT Productid, ProductName, price, categoryid,

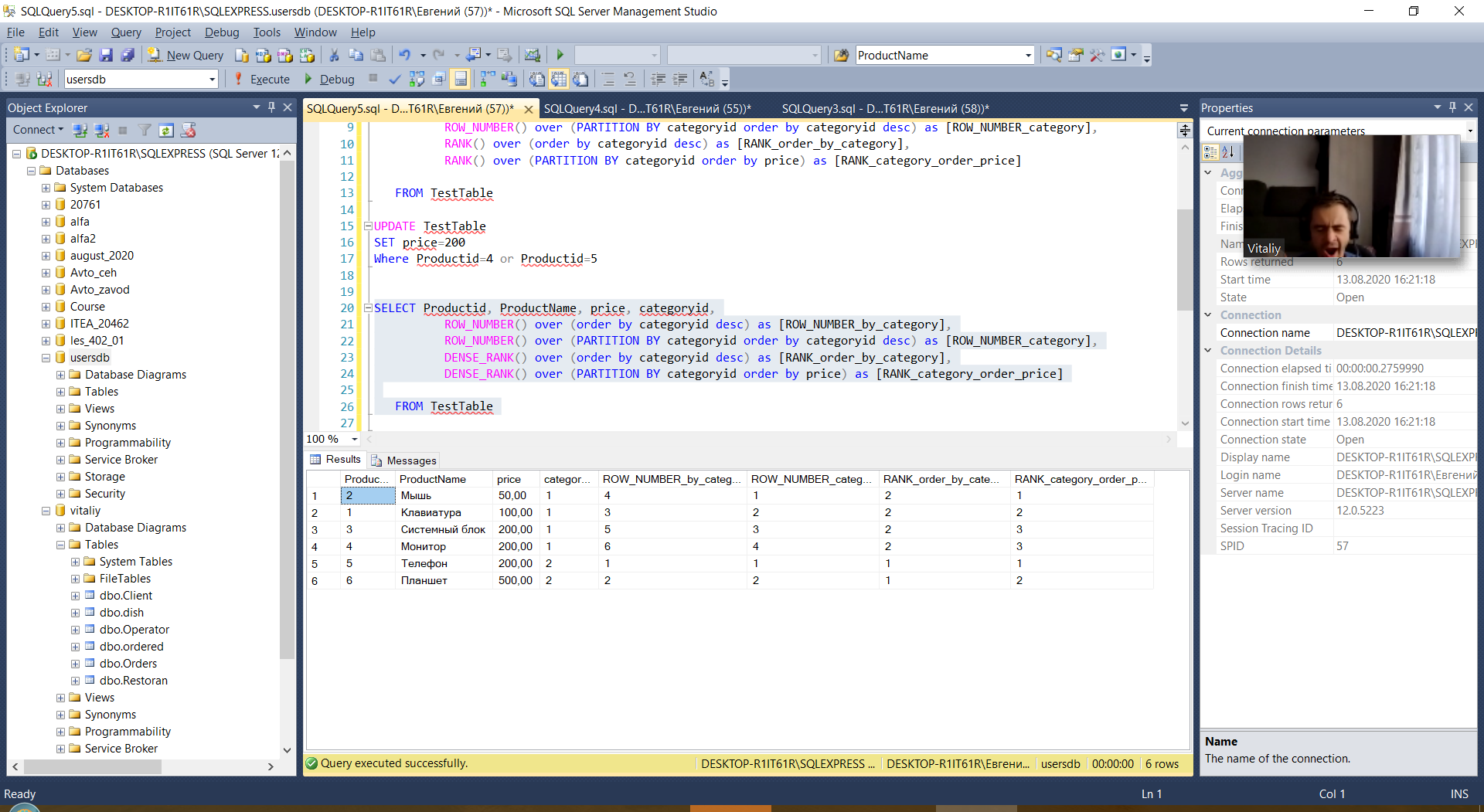
ROW\_NUMBER() over (order by categoryid desc) as [ROW\_NUMBER\_by\_category],

ROW\_NUMBER() over (PARTITION BY categoryid order by categoryid desc) as [ROW\_NUMBER\_category],

DENSE\_RANK() over (order by categoryid desc) as [RANK\_order\_by\_category],

DENSE\_RANK() over (PARTITION BY categoryid order by price) as [RANK\_category\_order\_price]

FROM TestTable



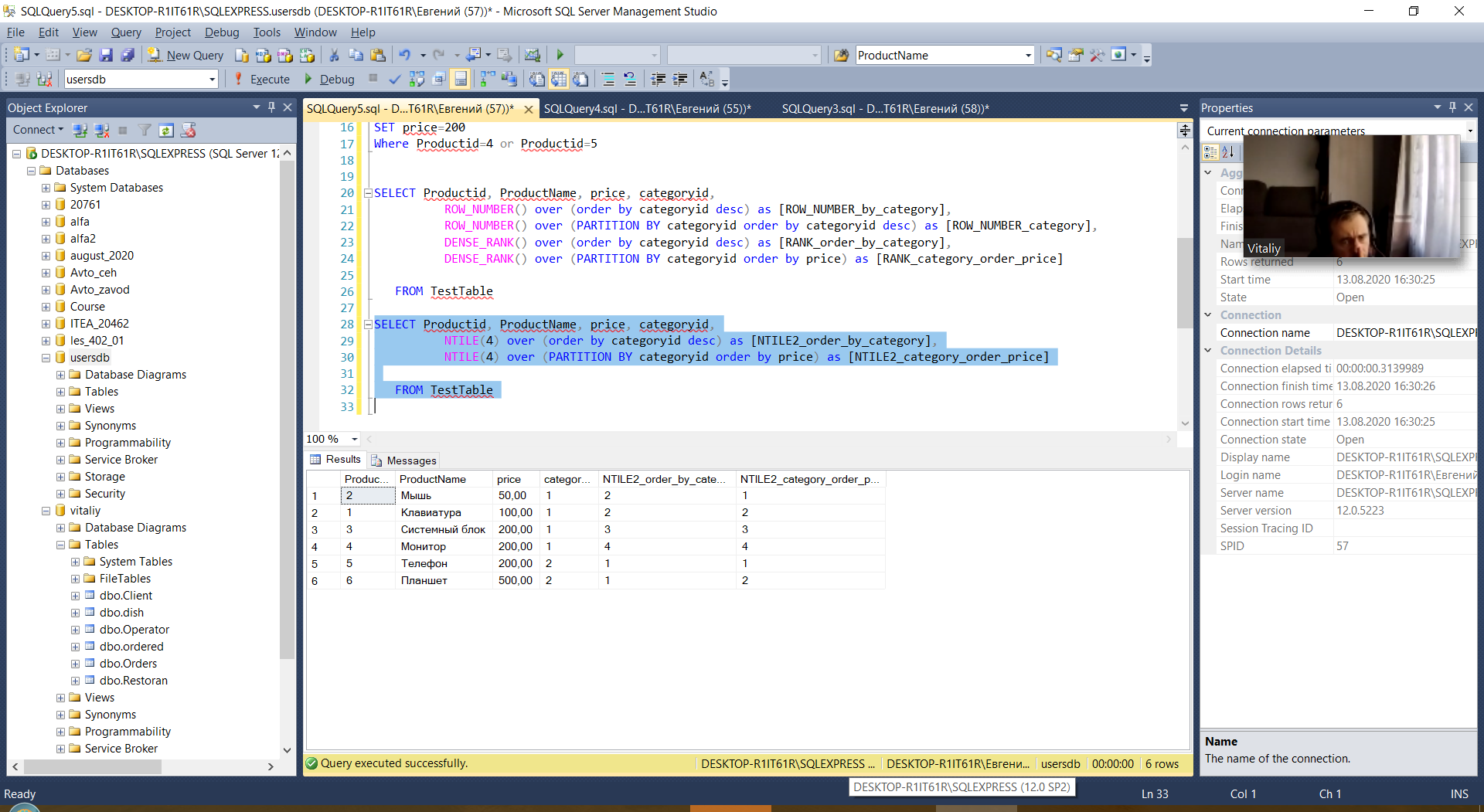
## NTILE

SELECT Productid, ProductName, price, categoryid,

NTILE(4) over (order by categoryid desc) as [NTILE2\_order\_by\_category],

NTILE(4) over (PARTITION BY categoryid order by price) as [NTILE2\_category\_order\_price]

FROM TestTable



В даному прикладі ми пронумеруємо рядки в кожній категорії, при цьому використовуємо сортування по стовпцю ProductId, а також виставимо ранг кожному товару в категорії на основі його ціни.

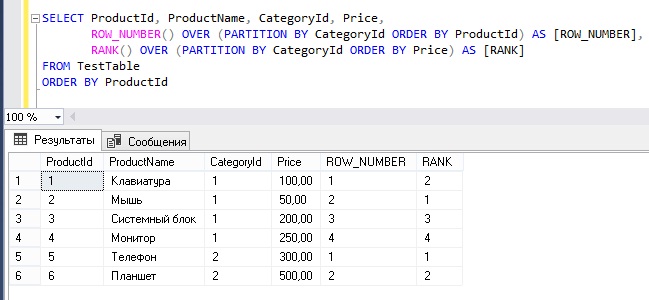
SELECT ProductId, ProductName, CategoryId, Price,

ROW\_NUMBER () OVER (PARTITION BY CategoryId ORDER BY ProductId) AS [ROW\_NUMBER],

RANK () OVER (PARTITION BY CategoryId ORDER BY Price) AS [RANK]

FROM TestTable

ORDER BY ProductId



### Віконні функції зміщення

**функції зміщення**- це функції, які дозволяють переміщатися і, відповідно, звертатися до різних рядках в наборі даних (вікні) щодо поточного рядка або просто звертатися до значень на початку або в кінці вікна. Ці функції з'явилися в Microsoft SQL Server 2012.

До функцій зсуву в T-SQL відносяться:

* **LEAD**- функція звертається до даних з наступного рядка набору даних. Її можна використовувати, наприклад, для того щоб порівняти поточне значення рядка з наступним. Має три параметра: стовпець, значення якого необхідно повернути (обов'язковий параметр), кількість рядків для зсуву (за замовчуванням 1), значення, яке необхідно повернути якщо після зсуву повертається значення NULL;
* **LAG**- функція звертається до даних з попереднього рядка набору даних. В даному випадку функцію можна використовувати для того, щоб порівняти поточне значення рядка з попереднім. Має три параметра: стовпець, значення якого необхідно повернути (обов'язковий параметр), кількість рядків для зсуву (за замовчуванням 1), значення, яке необхідно повернути якщо після зсуву повертається значення NULL;
* **FIRST\_VALUE**- функція повертає перше значення з набору даних, як параметр приймає стовпець, значення якого необхідно повернути;
* **LAST\_VALUE**- функція повертає останнє значення з набору даних, як параметр приймає стовпець, значення якого необхідно повернути.

**Приклад використання віконних функцій зсуву в T-SQL.**

У цьому прикладі спочатку ми повернемо наступне і попереднє значення ідентифікатора товару в категорії. Потім за допомогою FIRST\_VALUE і LAST\_VALUE отримаємо перше і останнє значення ідентифікатора товару в категорії, при цьому в якості прикладу використовуємо синтаксис додаткового обмеження рядків. А потім, використовуючи необов'язкові параметри функцій LEAD і LAG, ми змістилися вже на 2 рядки щодо поточної, при цьому, якщо після зсуву функцією LAG такого рядка не опиниться, нам повернеться 0, так як ми вкажемо третій необов'язковий параметр із значенням 0.

SELECT ProductId, ProductName, CategoryId, Price,

LEAD (ProductId) OVER (PARTITION BY CategoryId ORDER BY ProductId) AS [LEAD],

LAG (ProductId) OVER (PARTITION BY CategoryId ORDER BY ProductId) AS [LAG],

FIRST\_VALUE (ProductId) OVER (PARTITION BY CategoryId

ORDER BY ProductId

ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

) AS [FIRST\_VALUE],

LAST\_VALUE (ProductId) OVER (PARTITION BY CategoryId

ORDER BY ProductId

ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING

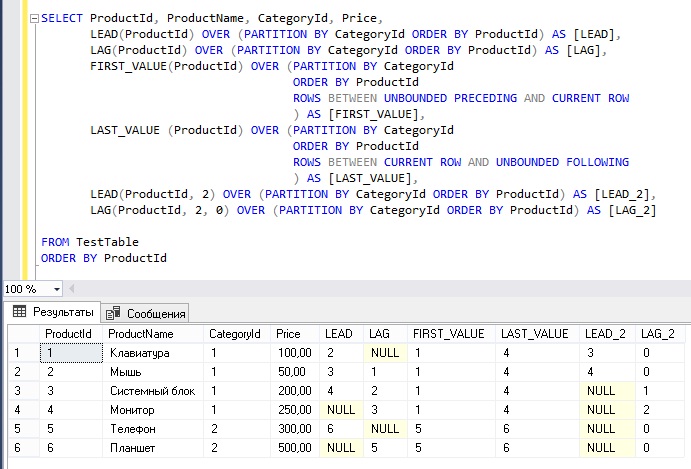
) AS [LAST\_VALUE],

LEAD (ProductId, 2) OVER (PARTITION BY CategoryId ORDER BY ProductId) AS [LEAD\_2],

LAG (ProductId, 2, 0) OVER (PARTITION BY CategoryId ORDER BY ProductId) AS [LAG\_2]

FROM TestTable

ORDER BY ProductId



Для вибору мінімальної і максимальної величини в категорії, а також альтернативний варіант агрегаторних функцій min () і max () по всій таблиці можна скористатися наступним кодом:

SELECT Productid, ProductName, price, categoryid,

FIRST\_VALUE (Price) OVER (PARTITION BY CategoryId ORDER BY Price

ROWS between UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) as [Min\_Price\_by\_category],

LAST\_VALUE (Price) OVER (PARTITION BY CategoryId ORDER BY Price

ROWS between UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) as [Max\_Price\_by\_category],

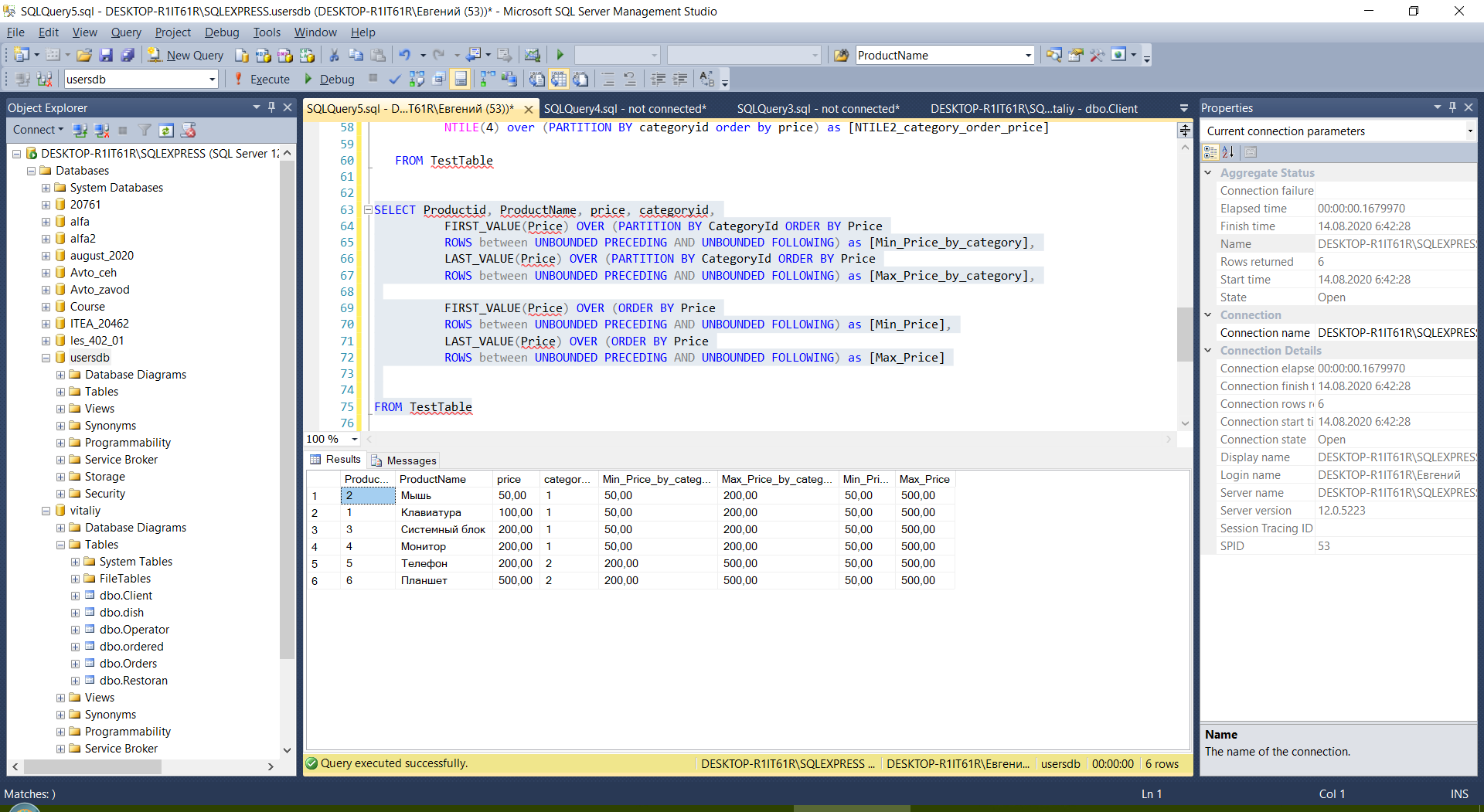
FIRST\_VALUE (Price) OVER (ORDER BY Price

ROWS between UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) as [Min\_Price],

LAST\_VALUE (Price) OVER (ORDER BY Price

ROWS between UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) as [Max\_Price]

FROM TestTable



**Примітка**: Зверніть увагу на облік початковій і кінцевій рядки вікна ROWS between UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING

### Аналітичні віконні функції

Аналітичні віконні функції - це по суті функції розподілу, які повертають інформацію про розподіл даних. Ці функції дуже специфічні і в основному використовуються для статистичного аналізу:

* **CUME\_DIST**- обчислює і повертає інтегральний розподіл значень в наборі даних. Іншими словами, вона визначає відносне положення значення в наборі;
* **PERCENT\_RANK**- обчислює і повертає відносний ранг рядки в наборі даних;
* **PERCENTILE\_CONT**- обчислює процентиль на основі постійного розподілу значення стовпця. Як параметр приймає процентиль, який необхідно обчислити;
* **PERCENTILE\_DISC**- обчислює певний процентиль для відсортованих значень в наборі даних. Як параметр приймає процентиль, який необхідно обчислити.

У функцій PERCENTILE\_CONT і PERCENTILE\_DISC синтаксис трохи відрізняється, стовпець, по якому сортувати дані, вказується за допомогою ключового слова WITHIN GROUP.

**Приклад використання аналітичних віконних функцій в T-SQL.**

SELECT ProductId, ProductName, CategoryId, Price,

CUME\_DIST () OVER (PARTITION BY CategoryId ORDER BY Price) AS [CUME\_DIST],

PERCENT\_RANK () OVER (PARTITION BY CategoryId ORDER BY Price) AS [PERCENT\_RANK],

PERCENTILE\_DISC (0.5) WITHIN GROUP (ORDER BY ProductId) OVER (PARTITION BY CategoryId) AS [PERCENTILE\_DISC],

PERCENTILE\_CONT (0.5) WITHIN GROUP (ORDER BY ProductId) OVER (PARTITION BY CategoryId) AS [PERCENTILE\_CONT]

FROM TestTable



Завдання

1. Вивести поля з порядковими номерами продуктів (таблиця Products з минулих лабораторних робіт) за наступними параметрами: а) за збільшенням ціни, б) за зменшенням ціни, в) за збільшенням ціни у межах Виробника (Manufacturer), г) за зменшенням ціни у межах Виробника (Manufacturer)
2. Вивести продукти, які є другими за ціною у своїй групі (групування за Виробником).
3. Вивести ціни, які є другими за величиною у кожного виробника (УВАГА!!! Дуже схоже на завдання 2 – різниця у результуючому наборі).
4. Показати назву продукту, його ціну і максимальну ціну у межах виробника (з використанням конструкції OVER)
5. Розбити виробників на 2 (за наявності великої кількості (>10) на 3) групи за ознакою сумарних продажів.
6. Розбити замовників на 2 (за наявності великої кількості (>10) на 3) групи за ознакою сумарної вартості покупок.
7. \*Вивести виробника, сумарна вартість залишків якого друга за величиною.
8. \*Вивести з проданих у цьому місяці продуктів ті, що мають другу за величною ціну у межах виробника.
9. \*Вивести таблицю продуктів з сортуванням за рангом їх популярності (кількість проданих одиниць).
10. \*Вивести таблицю замовників з сортуванням за рангом їх активності (кількість фактів покупок).
11. \*Вивести компанії виробників, у яких сумарні продажі їх товарів були другими зі всього списку компаній.
12. \*Вивести компанії виробників, у яких сумарні продажі їх товарів за 2022 року були другими зі всього списку компаній.