**Лабораторна робота №3**

Створимо таблиці для товарів, покупців і замовлень (УВАГА! Якщо ви виконували всі роботи підряд, то у вас частина таблиць вже є – або створіть нову базу, або внесіть коригування в запити, що надаються):

|  |
| --- |
| 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),  CustomerId INT NOT NULL REFERENCES Customers (Id),  CreatedAt DATE NOT NULL,  ProductCount INT DEFAULT 1,  Price MONEY NOT NULL  ); |

Таблиця Orders містить посилання на дві інші таблиці через поля ProductId і CustomerId. Додамо в таблиці деякі дані (УВАГА! Якщо ви виконували всі роботи підряд, то у вас частина таблиць вже є – або створіть нову базу, або внесіть коригування в запити, що надаються):

|  |
| --- |
| 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)  INSERT INTO Customers VALUES ('Tom'), ('Bob'), ('Sam')  INSERT INTO Orders  VALUES  (4,2,'2021-02-11',2,39999),  (2,2,'2021-02-11',2,56000),  (4,1,'2021-02-13',1,39999),  (7,1,'2021-02-14',5,17775) |

# Групування

## Агрегатні функції

Агрегатні функції виконують обчислення над значеннями в наборі рядків. В T-SQL є наступні агрегатні функції:

* **AVG**: Знаходить середнє значення
* **SUM**: Знаходить суму значень
* **MIN**: Знаходить найменше значення
* **MAX**: Знаходить найбільше значення
* **COUNT**: Знаходить кількість рядків в запиті

Як аргумент все агрегатні функції приймають вираз, яке представляє критерій дя визначення значень. Найчастіше, в якості вираження виступає назва стовпчика, над значеннями якого треба проводити обчислення.

Вирази в функціях AVG і SUM має представляти числове значення. Вираз у функціях MIN, MAX і COUNT може представляти числове або рядківове значення або дату.

Все агрегатні функції за винятком COUNT (\*)ігнорують значення NULL.

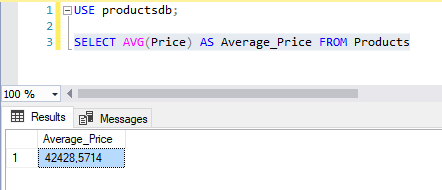
**Avg**

Функція Avg повертає середнє значення на діапазоні значень стовпця таблиці.

Знайдемо середню ціну товарів з бази даних:

|  |
| --- |
| SELECT AVG (Price) AS Average\_Price FROM Products |

Для пошуку середнього значення в якості вираження в функцію передається стовпець Price. Для одержуваного значення встановлюється псевдонім Average\_Price, хоча можна його і не встановлювати (**УВАГА!!!** Результат може відрізнятись від отриманого у вашій БД, у зв’язку з різним змістом таблиць).



Також ми можемо застосувати фільтрацію. Наприклад, знайти середню ціну для товарів якогось певного виробника:

|  |
| --- |
| SELECT AVG (Price) FROM Products  WHERE Manufacturer = 'Apple' |

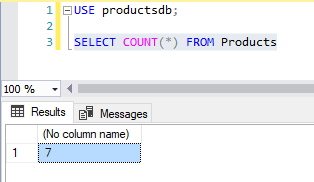
І, крім того, ми можемо знаходити середнє значення для більш складних виразів. Наприклад, знайдемо середню суму всіх товарів, з огляду на їх кількість:

|  |
| --- |
| SELECT AVG (Price \* ProductCount) FROM Products |

**Count**

Функція Count обчислює кількість рядків у вибірці. Є дві форми цієї функції. перша форма COUNT (\*)підраховує число рядків у вибірці:

|  |
| --- |
| SELECT COUNT (\*) FROM Products |



Друга форма функції обчислює кількість рядків за певним стовпцем, при цьому рядки зі значеннями NULL ігноруються:

|  |
| --- |
| SELECT COUNT (Manufacturer) FROM Products |

**Min і Max**

Функції Min і Max повертають відповідно мінімальне і максимальне значення за стовпцем. Наприклад, знайдемо мінімальну ціну серед товарів:

|  |
| --- |
| SELECT MIN (Price) FROM Products |

Пошук максимальної ціни:

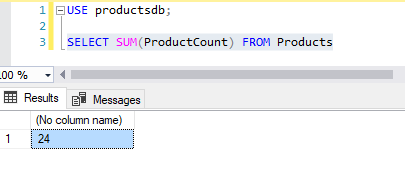
|  |
| --- |
| SELECT MAX (Price) FROM Products |

Дані функції також ігнорують значення NULL і не враховують їх при підрахунку.

**Sum**

Функція Sum обчислює суму значень стовпця. Наприклад, підрахуємо загальна кількість товарів:

|  |
| --- |
| SELECT SUM (ProductCount) FROM Products |



Також замість імені стовпця може передаватися обчислюється вираз. Наприклад, знайдемо загальну вартість всіх наявних товарів:

|  |
| --- |
| SELECT SUM (ProductCount \* Price) FROM Products |

**All і Distinct**

За замовчуванням всі вищеперелічених п'ять функцій враховують всі рядки вибірки для обчислення результату. Але вибірка може містити значення, що повторюються. Якщо необхідно виконати обчислення тільки над унікальними значеннями, виключивши з набору значень повторювані дані, то для цього застосовується оператор DISTINCT.

|  |
| --- |
| SELECT AVG (DISTINCT ProductCount) AS Average\_Price FROM Products |

За замовчуванням замість DISTINCT застосовується оператор ALL, який вибирає всі рядки:

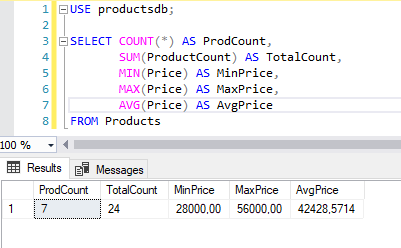
|  |
| --- |
| SELECT AVG (ALL ProductCount) AS Average\_Price FROM Products |

Так як цей оператор неявно мається на увазі при відсутності DISTINCT, то його можна не вказувати.

**Комбінування функцій**

Об'єднаймо застосування декількох функцій:

|  |
| --- |
| SELECT COUNT (\*) AS ProdCount, SUM (ProductCount) AS TotalCount,  MIN (Price) AS MinPrice, MAX (Price) AS MaxPrice,  AVG (Price) AS AvgPrice  FROM Products |



## Оператори GROUP BY і HAVING

Для групування даних в T-SQL застосовуються оператори GROUP BY і HAVING, для використання яких застосовується наступний формальний синтаксис:

|  |
| --- |
| SELECT стовпці  FROM таблиця  [WHERE умова\_фільтрації\_рядків]  [GROUP BY стовбці\_для\_групування]  [HAVING умова\_фільтрації\_груп]  [ORDER BY стовбці\_для\_сортування] |

**GROUP BY**

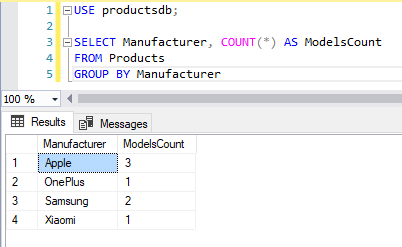
Оператор GROUP BY визначає, як рядки будуть групуватися.

Наприклад, згрупуємо товари по виробнику

|  |
| --- |
| SELECT Manufacturer, COUNT (\*) AS ModelsCount  FROM Products  GROUP BY Manufacturer |

Перший стовпець в виразі SELECT - Manufacturer представляє назву групи, а другий стовпець - ModelsCount представляє результат функції Count, яка обчислює кількість рядків в групі.

Варто враховувати, що будь-який стовпець, який використовується в виразі SELECT (не рахуючи стовпців, які зберігають результат агрегатних функцій), повинні бути вказані після оператора GROUP BY.



Так, наприклад, в разі вище стовпець Manufacturer вказано і в виразі SELECT, і в виразі GROUP BY.

І якщо в виразі SELECT проводиться вибірка по одному або кількох стовпців і також використовуються агрегатні функції, то необхідно вжити вислів GROUP BY. Так, наступний приклад працювати не буде, так як він не містить вираз групування:

|  |
| --- |
| SELECT Manufacturer, COUNT (\*) AS ModelsCount  FROM Products |

Інший приклад, додамо групування за кількістю товарів:

|  |
| --- |
| SELECT Manufacturer, ProductCount, COUNT (\*) AS ModelsCount  FROM Products  GROUP BY Manufacturer, ProductCount |

Оператор GROUP BY може виконувати групування по безлічі стовпців.

Якщо стовпець, по якому проводиться групування, містить значення NULL, то рядки зі значенням NULL складуть окрему групу.

Слід враховувати, що вираз GROUP BY має йти після висловлення WHERE, але до виразу ORDER BY:

|  |  |
| --- | --- |
| SELECT Manufacturer,  COUNT (\*) AS ModelsCount  FROM Products  WHERE Price> 30000  GROUP BY Manufacturer  ORDER BY ModelsCount DESC | GROUP BY в MS SQL Server |

**Фільтрація груп. HAVING**

Оператор HAVING визначає, які групи будуть включені в вихідний результат, тобто виконує фільтрацію груп.

Застосування HAVING багато в чому подібне до використання WHERE. Тільки є WHERE застосовується до фільтрації рядків, то HAVING використовується для фільтрації груп.

Наприклад, знайдемо всі групи товарів по виробникам, для яких визначено більше 1 моделі:

|  |  |
| --- | --- |
| SELECT Manufacturer,  COUNT (\*) AS ModelsCount  FROM Products  GROUP BY Manufacturer  HAVING COUNT (\*)> 1 | HAVING в T-SQL |

При цьому в одній команді ми можемо використовувати вирази WHERE і HAVING:

|  |
| --- |
| SELECT Manufacturer, COUNT (\*) AS ModelsCount  FROM Products  WHERE Price \* ProductCount> 80000  GROUP BY Manufacturer  HAVING COUNT(\*)> 1 |

Тобто в даному випадку спочатку фільтруються рядки: вибираються ті товари, загальна вартість яких більше 80000. Потім вибрані товари групуються по виробникам. І далі фільтруються самі групи - вибираються ті групи, які містять більше 1 моделі.

Якщо при цьому необхідно провести сортування, то вираз ORDER BY йде після висловлення HAVING:

|  |  |
| --- | --- |
| SELECT Manufacturer,  COUNT (\*) AS Models,  SUM (ProductCount) AS Units  FROM Products  WHERE Price \* ProductCount> 80000  GROUP BY Manufacturer  HAVING SUM (ProductCount)> 2  ORDER BY Units DESC | HAVING с фильтрацией и сортировкой в MS SQL Server |

В даному випадку групування йде по виробникам, і також вибирається кількість моделей для кожного виробника (Models) і загальна кількість всіх товарів за усіма цими моделями (Units). В кінці групи упорядковано відповідно до кількості товарів по спадаючій.

## Розширення SQL Server для групування

Додатково до стандартних операторів GROUP BY і HAVING SQL Server підтримує ще чотири спеціальні розширення для групування даних: ROLLUP, CUBE, GROUPING SETS і OVER.

**ROLLUP**

Оператор ROLLUP додає підсумовує рядок в результуючий набір:

|  |  |
| --- | --- |
| SELECT Manufacturer,  COUNT (\*) AS Models,  SUM (ProductCount) AS Units  FROM Products  GROUP BY Manufacturer WITH ROLLUP | Оператор ROLLUP в MS SQL Server |

Як видно з скріншоту, в кінці таблиці була додано додатковий рядок, який підсумовує значення стовпців.

Альтернативний синтаксис запиту, який можна використовувати, починаючи з версії MS SQL Server 2008.

|  |
| --- |
| SELECT Manufacturer, COUNT (\*) AS Models, SUM (ProductCount) AS Units  FROM Products  GROUP BY ROLLUP (Manufacturer) |

При групуванню за кількома критеріями ROLLUP створюватиме підсумовує рядок для кожної з підгруп:

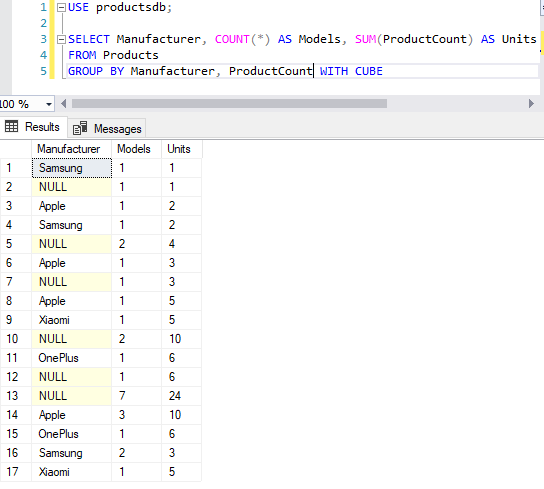
|  |  |
| --- | --- |
| SELECT  Manufacturer,  COUNT (\*) AS Models,  SUM (ProductCount) AS Units  FROM Products  GROUP BY Manufacturer, ProductCount WITH ROLLUP | WITH ROLLUP in T-SQL |

При сортуванні за допомогою ORDER BY слід враховувати, що вона застосовується вже після додавання підсумовує рядки.

**CUBE**

**CUBE** схожий на ROLLUP за тим винятком, що CUBE додає підсумовуючі рядки для кожної комбінації груп.

|  |
| --- |
| SELECT Manufacturer, COUNT (\*) AS Models, SUM (ProductCount) AS Units  FROM Products  GROUP BY Manufacturer, ProductCount WITH CUBE |



**Завдання**

1. Опрацювати всі наведені запити
2. Показати мінімальну вартість товару
3. Показати максимальну кількість залишків товару (ProductCount)
4. Вивести сумарну вартість залишків (кількість\*вартість) по кожній фірмі
5. Вивести середню вартість залишків (кількість\*вартість) по кожній фірмі
6. Вивести дані, які покажуть скільки є позицій з кількістю залишків 1, скільки з кількістю залишків 2 і т.д.
7. Показати фірми, сумарна вартість товарів яких більше 100000 грн.
8. Показати фірми, сумарна кількість залишків товарів яких менше 10.
9. Вивести кількості позицій з однаковою сумарною вартістю (кількість\*вартість)
10. \*Виведіть вартість залишків (кількість\*вартість) з представленням суми по кожній фірмі (використовувати ROLLUP)
11. Вивести виторг за днями.
12. Показати ті дні, де виторг був більше 50000.
13. Показати id замовників, які робили більше 2 замовлень в один з днів.
14. Показати id замовників, які робили замовлення в один з днів на суму більше 50000 (кількість\*вартість в таблиці Orders).
15. Скільки було продано товарів за останній місяць (останній місяць підібрати з урахуванням дат замовлень).
16. \*Без підзапитів знайти дні, коли було продано більше 3 товарів компанії OnePlus.

**Відповіді (можливий варіант)**

--#2

select MIN(price) from Products;

АБО

select top 1 \* from Products order by Price;

--#3

select MAX(ProductCount) from Products;

АБО

select top 1 \* from Products order by ProductCount desc;

--#4

select Manufacturer, sum(ProductCount\*Price) from Products group by Manufacturer;

--#5

select Manufacturer, avg(ProductCount\*Price) from Products group by Manufacturer;

--#6

select ProductCount, COUNT(\*) from Products group by ProductCount;

--#7

select Manufacturer, sum(ProductCount\*Price) from Products group by Manufacturer having sum(ProductCount\*Price)>100000

--#8

select Manufacturer, sum(ProductCount) from Products group by Manufacturer having sum(ProductCount)<10;

--#9

select ProductCount\*Price, count(\*) from Products group by ProductCount\*Price having count(\*)>1;

--#10

select Manufacturer, sum(ProductCount\*Price) from Products group by Manufacturer with rollup;

--#11

select CreatedAt, sum(Price\*ProductCount) Sum\_At\_Date from Orders group by CreatedAt;

--#12

select CreatedAt, sum(ProductCount\*Price) Sum\_At\_Date from Orders group by CreatedAt having sum(ProductCount\*Price)>50000;

--#13

select CustomerId from Orders group by CustomerId, CreatedAt having count(\*)>=2

--#14

select CustomerId from Orders group by CustomerId, CreatedAt having sum(ProductCount\*Price)>50000

--#15

SELECT SUM(ProductCount) FROM Orders

WHERE CreatedAt<'2021-03-01' AND CreatedAt>='2021-02-01'

--#16

SELECT id FROM Products

WHERE Manufacturer='OnePlus'

--у мене був id=7

SELECT CreatedAt, SUM(ProductCount) FROM Orders

WHERE ProductId=7

GROUP BY ProductId, CreatedAt

HAVING SUM(ProductCount)>3