# Лабораторна робота №4. Вбудовані функції

Створимо таблиці для товарів, покупців і замовлень (УВАГА! Якщо ви виконували всі роботи підряд, то у вас ці таблиці вже є):

|  |
| --- |
| 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:

|  |
| --- |
| 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) |

# Вбудовані функції

## Функції для роботи з рядками

Для роботи з рядками в T-SQL можна застосовувати такі функції:

* **LEN**: Повертає кількість символів в рядку. Як параметр у функцію передається рядок, для якої треба знайти довжину:

|  |
| --- |
| SELECT LEN ( 'Apple') – 5 |

* **LTRIM**: Видаляє початкові пробіли з рядка. Як параметр приймає рядок:

|  |
| --- |
| SELECT LTRIM ( 'Apple') |

* **RTRIM**: Видаляє кінцеві пробіли з рядка. Як параметр приймає рядок:

|  |
| --- |
| SELECT RTRIM ( 'Apple') |

* **CHARINDEX**: Повертає індекс, за яким знаходиться перше входження підрядка в рядку. В якості першого параметра передається подстрока, а в якості другого - рядок, в якій треба вести пошук:

|  |
| --- |
| SELECT CHARINDEX ( 'pl', 'Apple') - 3 |

* **PATINDEX**: Повертає індекс, за яким знаходиться перше входження певного шаблону в рядку:

|  |
| --- |
| SELECT PATINDEX ( '%p\_e%', 'Apple') - 3 |

* **LEFT**: Вирізає з початку рядка певну кількість символів. Перший параметр функції - рядок, а другий - кількість символів, які треба вирізати спочатку рядки:

|  |
| --- |
| SELECT LEFT ( 'Apple', 3) – App |

* **RIGHT**: Вирізає з кінця рядка певну кількість символів. Перший параметр функції - рядок, а другий - кількість символів, які треба вирізати спочатку рядки:

|  |
| --- |
| SELECT RIGHT ( 'Apple', 3) - ple |

* **SUBSTRING**: Вирізає з рядка підрядок певною довжиною, починаючи з певного індексу. Певий параметр функції - рядок, другий - початковий індекс для вирізки, і третій параметр - кількість вирізаних символів:

|  |
| --- |
| SELECT SUBSTRING ( 'Galaxy S21 Ultra', 8, 2) - S2 |

* **REPLACE**: Замінює одну подстроку інший в рамках рядку. Перший параметр функції - рядок, другий - підрядок, яку треба замінити, а третій - підрядок, на яку треба замінити:

|  |
| --- |
| SELECT REPLACE ( 'Galaxy S21 Ultra', 'S21 Ultra', 'Note 8') - Galaxy Note 8 |

* **REVERSE**: Перевертає рядок навпаки:

|  |
| --- |
| SELECT REVERSE ( '123456789') – 987654321 |

* **CONCAT**: Об'єднує два рядки в одну. Як параметр приймає від 2-х і більше рядків, які треба з'єднати:

|  |
| --- |
| SELECT CONCAT ( 'Tom', '', 'Smith') - Tom Smith |

* **LOWER**: Переводить рядок в нижній регістр:

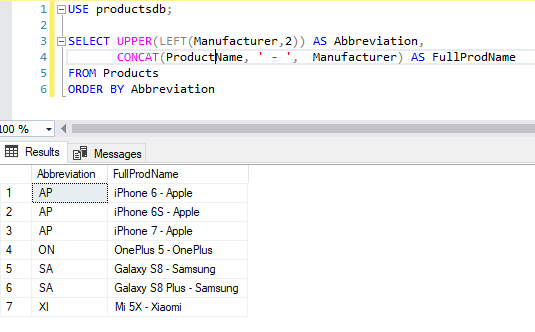
|  |
| --- |
| SELECT LOWER ( 'Apple') – apple |

* **UPPER**: Переводить рядок у верхній регістр

|  |
| --- |
| SELECT UPPER ( 'Apple') – APPLE |

* **SPACE**: Повертає рядок, яка містить певну кількість пропусків
* І при отриманні даних застосуємо рядкові функції:

|  |
| --- |
| SELECT UPPER (LEFT (Manufacturer, 2)) AS Abbreviation,  CONCAT (ProductName, '-', Manufacturer) AS FullProdName  FROM Products  ORDER BY Abbreviation |



## Функції для роботи з числами

Для роботи з числовими даними T-SQL надає ряд функцій:

* **ROUND**: Округлює число. В якості першого параметра передається число. Другий параметр вказує на довжину. Якщо довжина становить позитивне число, то воно вказує, до якої цифри після коми йде округлення. Якщо довжина становить негативне число, то воно вказує, до якої цифри з кінця числа до коми йде округлення

|  |
| --- |
| SELECT ROUND (1342.345, 2) - 1342.350  SELECT ROUND (1342.345, -2) - 1300.000 |

* **ISNUMERIC**: Визначає, чи є значення числом. Як параметр функція приймає вираз. Якщо вираз є числом, то функція повертає 1. Якщо не є, то повертається 0.

|  |
| --- |
| SELECT ISNUMERIC (1342.345) - 1  SELECT ISNUMERIC ( '1342.345') - 1  SELECT ISNUMERIC ( 'SQL') - 0  SELECT ISNUMERIC ('13 -04-2020 ') - 0 |

* **ABS**: Повертає абсолютне значення числа.

|  |
| --- |
| SELECT ABS (-123) – 123 |

* **CEILING**: Повертає найменше ціле число, яке більше або дорівнює поточному значенню.

|  |
| --- |
| SELECT CEILING (-123.45) - -123  SELECT CEILING (123.45) - 124 |

* **FLOOR**: Повертає найбільше ціле число, яке менше або дорівнює поточному значенню.

|  |
| --- |
| SELECT FLOOR (-123.45) - -124  SELECT FLOOR (123.45) - 123 |

* **SQUARE**: Зводить число в квадрат.

|  |
| --- |
| SELECT SQUARE (5) - 25 |

* **SQRT**: Отримує квадратний корінь числа.

|  |
| --- |
| SELECT SQRT (225) - 15 |

* **RAND**: Генерує випадкове число з плаваючою точкою в діапазоні від 0 до 1.

|  |
| --- |
| SELECT RAND () - 0.707365088352935  SELECT RAND () - 0.173808327956812 |

* **COS**: Повертає косинус кута, вираженого в радіанах

|  |
| --- |
| SELECT COS (1.0472) - 0.5 - 60 градусів |

* **SIN**: Повертає синус кута, вираженого в радіанах

|  |
| --- |
| SELECT SIN (1.5708) - 1 - 90 градусів |

* **TAN**: Повертає тангенс кута, вираженого в радіанах

|  |
| --- |
| SELECT TAN (0.7854) - 1 - 45 градусів |

Округлимо твір ціни товару на кількість цього товару:

|  |
| --- |
| SELECT ProductName, ROUND (Price \* ProductCount, 2)  FROM Products |

## Функції по роботі з датами і часом

T-SQL надає ряд функцій для роботи з датами і часом:

* **GETDATE**: Повертає поточну локальну дату і час на основі системного годинника у вигляді об'єкта datetime

|  |
| --- |
| SELECT GETDATE () - 2020-07-28 21: 34: 55.830 |

* **GETUTCDATE**: Повертає поточну локальну дату і час за Гринвічем (UTC / GMT) у вигляді об'єкта datetime

|  |
| --- |
| SELECT GETUTCDATE () - 2020-07-28 18: 34: 55.830 |

* **SYSDATETIME**: Повертає поточну локальну дату і час на основі системного годинника, але відміну від GETDATE полягає в тому, що дата і час повертаються у вигляді об'єкта datetime2

|  |
| --- |
| SELECT SYSDATETIME () - 2020-07-28 21: 02: 22.7446744 |

* **SYSUTCDATETIME**: Повертає поточну локальну дату і час за Гринвічем (UTC / GMT) у вигляді об'єкта datetime2

|  |
| --- |
| SELECT SYSUTCDATETIME () - 2020-07-28 18: 20: 27.5202777 |

* **SYSDATETIMEOFFSET**: Повертає об'єкт datetimeoffset (7), який містить дату і час щодо GMT

|  |
| --- |
| SELECT SYSDATETIMEOFFSET () - 2020-07-28 21: 02: 22.7446744 +03: 00 |

* **DAY**: Повертає день дати, який передається в якості параметра

|  |
| --- |
| SELECT DAY (GETDATE ()) - 28 |

* **MONTH**: Повертає місяць дати

|  |
| --- |
| SELECT MONTH (GETDATE ()) - 7 |

* **YEAR**: Повертає рік з дати

|  |
| --- |
| SELECT YEAR (GETDATE ()) - 2020 |

* **DATENAME**: Повертає частину цієї дати у вигляді рядка. Параметр вибору частині дати передається в якості першого параметра, а сама дата передається в якості другого параметра:

|  |
| --- |
| SELECT DATENAME (month, GETDATE ()) - July |

* Для визначення частини дати можна використовувати наступні параметри (в дужках вказані їх скорочені версії):
  + year (yy, yyyy): рік
  + quarter (qq, q): квартал
  + month (mm, m): місяць
  + dayofyear (dy, y): День року
  + day (dd, d): День місяця
  + week (wk, ww): тиждень
  + weekday (dw): день тижня
  + hour (hh): годину
  + minute (mi, n): хвилина
  + second (ss, s): секунда
  + millisecond (ms): мілісекунда
  + microsecond (mcs): мікросекунда
  + nanosecond (ns): наносекунд
  + tzoffset (tz): Змішання в хвилинах щодо Гринвічем (для об'єкта datetimeoffset)
* **DATEPART**: Повертає частину цієї дати у вигляді числа. Параметр вибору частині дати передається в якості першого параметра (використовуються ті ж параметри, що і для DATENAME), а сама дата передається в якості другого параметра:

|  |
| --- |
| SELECT DATEPART (month, GETDATE ()) - 7 |

* **DATEADD**: Повертає дату, яка є результатом складання числа до певного компоненту дати. Перший параметр представляє компонент дати, описаний вище для функції DATENAME. Другий параметр - додається кількість. Третій параметр - сама дата, до якої треба зробити додаток:

|  |
| --- |
| SELECT DATEADD (month, 2, '2020-7-28') - 2020-09-28 00: 00: 00.000  SELECT DATEADD (day, 5, '2020-7-28') - 2020-08-02 00: 00: 00.000  SELECT DATEADD (day, -5, '2020-7-28') - 2020-07-23 00: 00: 00.000 |

* Якщо додається кількість представляє негативне число, то фактично відбувається зменшення дати.
* **DATEDIFF**: Повертає різницю між двома датами. Перший параметр - компонент дати, який вказує, в яких одиницях варто вимірювати різницю. Другий і третій параметри - порівнювані дати:

|  |
| --- |
| SELECT DATEDIFF (year, '2020-7-28', '2021-9-28')  - різниця 1 рік  SELECT DATEDIFF (month, '2020-7-28', '2021-9-28')  - різниця 14 місяців  SELECT DATEDIFF (day, '2020-7-28', '2021-9-28')  - різниця 427 днів |

* **TODATETIMEOFFSET**: Повертає значення datetimeoffset, яке є результатом складання тимчасового зсуву з іншим об'єктом datetimeoffset

|  |  |
| --- | --- |
| 1 | SELECT TODATETIMEOFFSET ( '2020-7-28 1:10:22', '+03: 00') |

* **SWITCHOFFSET**: Повертає значення datetimeoffset, яке є результатом складання тимчасового зсуву з об'єктом datetime2

|  |
| --- |
| SELECT SWITCHOFFSET (SYSDATETIMEOFFSET (), '+02: 30') |

* **EOMONTH**: Повертає дату останнього дня для місяця, який використовується в переданої в якості параметра датою.

|  |
| --- |
| SELECT EOMONTH ( '2020-02-05') - 2020-02-28  SELECT EOMONTH ( '2020-02-05', 3) - 2020-05-31 |

* Як необов'язкового другого параметра можна передавати кількість місяців, які необхідно додати до дати. Тоді останній день місяця буде обчислюватися для нової дати.
* **DATEFROMPARTS**: За роком, місяцем і днем створює дату

|  |
| --- |
| SELECT DATEFROMPARTS (2020, 7, 28) - 2020-07-28 |

* **ISDATE**: Перевіряє, чи є вираз датою. Якщо є, то повертає 1, інакше повертає 0.

|  |
| --- |
| SELECT ISDATE ( '2020-07-28') - 1  SELECT ISDATE ( '2020-28-07') - 0  SELECT ISDATE ('28-07-2020') - 0  SELECT ISDATE ( 'SQL') – 0 |

Знайдемо замовлення, які були зроблені 16 днів тому:

|  |
| --- |
| SELECT \* FROM Orders  WHERE DATEDIFF (day, CreatedAt, GETDATE ()) = 16 |

## Перетворення даних

Коли ми присвоюємо значення одного одного типу колонки, який зберігає дані іншого типу, або виконуємо операції, які втягують дані різних типів, SQL Server намагається виконати перетворення і привести використовується значення до потрібного типу. Але не всі перетворення SQL Server може виконати автоматично. SQL Server може виконувати неявні перетворення від типу з меншим пріоритетом до типу з великим пріоритетом. Таблиця пріоритетів (чим вище, тим більший пріоритет):

|  |
| --- |
| datetime |
| smalldatetime |
| float |
| real |
| decimal |
| money |
| smallmoney |
| int |
| smallint |
| tinyint |
| bit |
| nvarchar |
| nchar |
| varchar |
| char |

Тобто SQL Server автоматично може перетворити число 100.0 (float) в дату і час (datetime).

У тих випадках, коли необхідно виконати перетворення від типів з вищим пріоритетом до типам з нижчим пріоритетом, то треба виконувати явне приведення типів. Для цього в T-SQL визначені дві функції: CONVERT і CAST.

Функція CAST перетворює вираз одного типу до іншого. Вона має таку форму:

|  |
| --- |
| CAST (вираз AS тип даних) |

при поданні інформації про замовлення перетворює числове значення і дату в рядок:

|  |
| --- |
| SELECT Id, CAST (CreatedAt AS nvarchar) + '; total: ' + CAST (Price \* ProductCount AS nvarchar)  FROM Orders |

# Функция CAST в T-SQL и MS SQL Server

**Convert**

Більшу частину перетворень охоплює функція CAST. Якщо ж необхідно якесь додаткове форматування, то можна використовувати функцію CONVERT. Вона має таку форму:

|  |
| --- |
| CONVERT (тип\_даних, вираз [, стиль]) |

Третій необов'язковий параметр задає стиль форматування даних. Цей параметр представляє числове значення, яке для різних типів даних має різну інтерпретацію. Наприклад, деякі значення для форматування дат і часу:

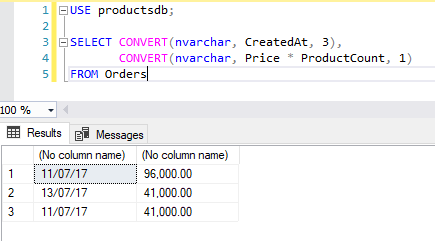
* 0 або 100 - формат дати "Mon dd yyyy hh: miAM / PM" (значення за замовчуванням)
* 1 або 101 - формат дати "mm / dd / yyyy"
* 3 або 103 - формат дати "dd / mm / yyyy"
* 7 або 107 - формат дати "Mon dd, yyyy hh: miAM / PM"
* 8 або 108 - формат дати "hh: mi: ss"
* 10 або 110 - формат дати "mm-dd-yyyy"
* 14 або 114 - формат дати "hh: mi: ss: mmmm" (24-годинний формат часу)

Деякі значення для форматування даних типу money в рядок:

* 0 - в дробової частини числа залишаються тільки дві цифри (за замовчуванням)
* 1 - в дробової частини числа залишаються тільки дві цифри, а для поділу розрядів застосовується кома
* 2 - в дробової частини числа залишаються тільки чотири цифри

**Наприклад, виведемо дату і вартість замовлень з форматуванням:**

|  |
| --- |
| SELECT CONVERT (nvarchar, CreatedAt, 3),  CONVERT (nvarchar, Price \* ProductCount, 1)  FROM Orders |



**TRY\_CONVERT**

При використанні функцій CAST і CONVERT SQL Server викидає виключення, якщо дані не можна привести до певного типу. наприклад:

|  |  |
| --- | --- |
| 1 | SELECT CONVERT (int, 'sql') |

Щоб уникнути генерації виключення можна використовувати функцію TRY\_CONVERT. Її використання аналогічно функції CONVERT за тим винятком, що якщо вираз не вдається перетворити до потрібного типу, то функція повертає NULL:

|  |
| --- |
| SELECT TRY\_CONVERT (int, 'sql') - NULL  SELECT TRY\_CONVERT (int, '22') - 22 |

**Додаткові функції**

Крім CAST, CONVERT, TRY\_CONVERT є ще ряд функцій, які можуть використовуватися для перетворення в ряд типів:

* **STR (float [, length [, decimal]])**: Перетворює число в рядок. Другий параметр вказує на довжину рядка, а третій - скільки знаків у дробовій частині числа треба залишати
* **CHAR (int)**: Перетворює числовий код ASCII в символ. Нерідко використовується для тих ситуацій, коли необхідний символ, який не можна ввести з клавіатури
* **ASCII (char)**: Перетворює символ в числовий код ASCII
* **NCHAR (int)**: Перетворює числовий код UNICODE в символ
* **UNICODE (char)**: Перетворює символ в числовий код UNICODE

|  |
| --- |
| SELECT STR (123.4567, 6,2) - 123.46  SELECT CHAR (219) - И  SELECT ASCII ( 'И') - 219  SELECT NCHAR (1067) - И  SELECT UNICODE ( 'И') - 1067 |

## Функції CASE і IIF

**CASE**

Функція CASE перевіряє значення деякого вираження, і в залежності від результату перевірки може повертати той чи інший результат.

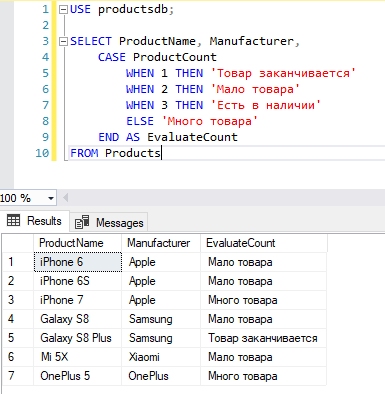
CASE приймає наступну форму:

|  |
| --- |
| CASE вираз  WHEN значення\_1 THEN результат\_1  WHEN значення\_2 THEN результат\_2  .................................  WHEN значення\_N THEN результат\_N  [ELSE альтернативний\_результат]  END |

Виконаємо запит до таблиці Products і використовуємо функцію CASE:

|  |
| --- |
| SELECT ProductName, Manufacturer,  CASE ProductCount  WHEN 1 THEN 'Товар закінчується'  WHEN 2 THEN 'Мало товару'  WHEN 3 THEN 'Є в наявності'  ELSE 'Багато товару'  END AS EvaluateCount  FROM Products |

Тут значення стовпця ProductCount послідовно порівнюється зі значеннями після операторів WHEN. Залежно від значення стовпця ProductCount функція CASE буде повертати одну з рядків, яка йде після відповідного оператора THEN. Для повертається результату визначено стовпець EvaluateCount:



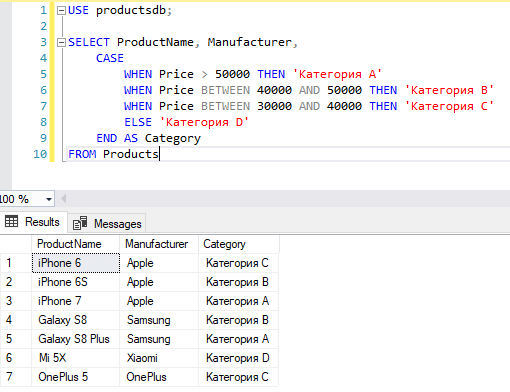
Також функція CASE може приймати ще одну форму:

|  |
| --- |
| CASE  WHEN вираз\_1 THEN результат\_1  WHEN вираз\_2 THEN результат\_2  .................................  WHEN вираз\_N THEN результат\_N  [ELSE альтернатівний\_результат]  END |

Наприклад, стосовно таблиці Products:

|  |
| --- |
| SELECT ProductName, Manufacturer,  CASE  WHEN Price> 50000 THEN 'Категорія A'  WHEN Price BETWEEN 40000 AND 50000 THEN 'Категорія B'  WHEN Price BETWEEN 30000 AND 40000 THEN 'Категорія C'  ELSE 'Категорія D'  END AS Category  FROM Products |

Фактично все те ж саме, що і в попередньому прикладі, тільки після CASE не вказується значення для порівняння. А самі вирази порівняння стоять після оператора WHEN. І якщо вираз після оператора WHEN було це слово, то повертається значення, яке йде після відповідного оператора THEN.



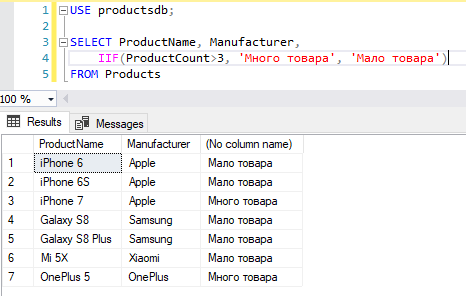
**IIF**

Функція IIF в залежності від результату умовного виразу повертає одне з двох значень. Загальна форма функції виглядає наступним чином:

|  |  |
| --- | --- |
| 1 | IIF (умова, значення\_1, значення\_2) |

Якщо умова в функції IIF правдиве повертається значення\_1, якщо помилково, то повертається значення\_2. наприклад:

|  |
| --- |
| SELECT ProductName, Manufacturer,  IIF (ProductCount> 3, 'Багато товару', 'Мало товару')  FROM Products |



# Завдання (10 завдань)

* 1. Вивести товари, довжина назв яких більше 15.
  2. Вивести замовлення тільки за суботи і неділі останнього місяця
  3. Вивести замовлення з датою у форматі dd-mm-yy (реалізувати декілька варіантів: А) через функції DAY, MONTH, YEAR, Б) через функцію DATEPART, В) через функцію FORMAT (приклад дивитись у лекції)).
  4. \*Вивести в окремі поля назву товару та його модель (модель починається з першого пробілу у повній назві).
  5. \*Вивести товари, які не мають моделі (модель починається з пробілу і має число у повній назві)
  6. \* Вивести лише ті місяці років, де сумарний виторг був більше 100000.
  7. \* Додати в таблицю продуктів поле PriceDate NVARCHAR (30). Записати в це поле ціну і поточну дату за прикладом: якщо була ціна 25000, а дата 2022-01-11, то вписати в PriceDate «25000 грн. на 11-01-2022».
  8. \* Написати запит для виведення випадкових чисел в діапазоні [-50;50].
  9. \* Додати в таблицю продуктів поля дати випуску та днів гарантії. Вивести товари, до закінчення гарантії яких залишилось менше 2 днів.
  10. \*Вивести в додатковому стовбці інформацію, щодо сумарної вартості залишку по позиції. Якщо вартість більше 200000, то написати "Вартість > 200000", якщо менша 200000, то написати "Вартість < 200000", якщо дорівнює 0, то написати "ПЕРЕВІРИТИ КІЛЬКІСТЬ ЗАЛИШКІВ". Для товарів компанії ‘Apple’ в кінці кожної фрази додати у текст "(ЦЕ APPLE!!!)".
  11. **Проробити розв’язок задач:**

11.1. Розділити назву компанії на дві частини (до першого пробіла і решта), використовуючи LEN, CHARINDEX, LEFT, RIGHT

РОЗВ’ЯЗОК

1) Представити, яким буде результат (які поля, які приблизні значення)

2) Якщо можливо, то використати вбудовану функцію, інакше п. 3

3) Розділити задачу на підзадачі і до кожної виконати п.2

SELECT Manufacturer FROM Products;

--1) 2 стовпчики: в першому частина поля до першого пробілу,

--в другому - після першого пробілу

--2) Вбудована функція відсутня

--3) Розділимо на 2 задачі:

--А.1) отримати частину до першогопробілу,

--А.2) отримати частину після першого пробілу

-- А.1.1) отримати частину до першого пробілу

-- А.1.2) Вбудована функція відсутня

-- А.1.3) Розділимо на 2 задачі:

-- А.1.3.1) Знайти місце пробілу (де N - індекс пробілу)

-- А.1.3.2) Вивести N перших символів

SELECT IIF(CHARINDEX (' ',Manufacturer)=0, Manufacturer, LEFT (Manufacturer, CHARINDEX (' ',Manufacturer)-1)) 'Перша частина'

FROM Products;

-- А.2.1) отримати частину після першого пробілу

-- А.2.2) Вбудована функція відсутня

-- А.2.3) Розділимо на 2 задачі:

-- А.2.3.1) Знайти місце пробілу

-- А.2.3.2) Вивести LEN-N правих символів (де N - індекс пробілу)

SELECT IIF(CHARINDEX(' ',Manufacturer)=0,' ',

RIGHT( Manufacturer,

LEN(Manufacturer)-CHARINDEX (' ',Manufacturer))

) 'Друга частина'

FROM Products;

SELECT IIF(CHARINDEX (' ',Manufacturer)=0,

Manufacturer,

LEFT (Manufacturer, CHARINDEX (' ',Manufacturer)-1)) 'Перша частина' ,

IIF(CHARINDEX(' ',Manufacturer)=0,

' ',

RIGHT( Manufacturer,

LEN(Manufacturer)-CHARINDEX (' ',Manufacturer))) 'Друга частина'

FROM Products;

Позбутися IIF можна за рахунок додавання в кінець рядку пробілу:

SELECT LEFT(Manufacturer,

CHARINDEX(' ',Manufacturer+' ',1)),

RIGHT(Manufacturer,

LEN(Manufacturer) - CHARINDEX(' ',Manufacturer+' ',1)+1)

FROM Products

11.2. Розділити назву компанії на дві частини (до першого пробілу і решта)

РОЗВ’ЯЗОК:

1) 2 стовпчики: в першому частина поля до першого пробілу, в другому - після першого пробілу

2) Вбудована функція відсутня

3) Розділимо на 2 задачі:

А.1) вивести підрядок від початку довжиною N (N - індекс першого пробілу),

А.2) вивести підрядок від символу N довжиною LEN-N+1

SELECT SUBSTRING (

Manufacturer,

1,

CHARINDEX (' ',Manufacturer+' '))

'Перша частина',

SUBSTRING (

Manufacturer,

CHARINDEX (' ',Manufacturer+' ')+1,

LEN(Manufacturer)-CHARINDEX (' ',Manufacturer+' ')+1)

'Друга частина'

FROM Products;

Позбутися IIF можна за рахунок додавання в кінець рядку пробілу:

SELECT LEFT(Manufacturer,

CHARINDEX(' ',Manufacturer+' ',1)),

RIGHT(Manufacturer,

LEN(Manufacturer) - CHARINDEX(' ',Manufacturer+' ',1)+1)

FROM Products

* 1. \*\*Побудувати таблицю з полями:

FullName, - повне ім’я (може бути записано як всі три значення - Прізвище Ім'я По батькові, так і два значення - Прізвище Ім'я, так і одне - Прізвище),

FirstName - прізвище,

SecondName – ім’я,

Third\_Name - По батькові.

Вивести на екран 4 стовбці:

1) FullName,

2) Прізвище - це перша частина поля FullName до пробіла,

3) Ім’я – це друга частина поля FullName між пробілами,

4) По батькові - це третя частина поля FullName після другого пробіла.

РОЗВ’ЯЗОК:

CREATE TABLE MyTable

(

FullName NVARCHAR(100), -- Поле для повного імені

FirstName NVARCHAR(50), -- Поле для прізвища

SecondName NVARCHAR(50), -- Поле для імені

Third\_Name NVARCHAR(50) -- Поле для по батькові

);

INSERT INTO MyTable (FullName)

VALUES

('Артамонов'),

('Артамонов Євген'),

('Артамонов Євген Борисович');

Запит сумісний з MS SQL Server 2014

SELECT FullName,

CASE

WHEN CHARINDEX(' ', FullName) > 0

THEN LEFT(FullName, CHARINDEX(' ', FullName) - 1)

ELSE FullName

END 'Прізвище',

CASE

WHEN CHARINDEX(' ', FullName) > 0

THEN CASE

WHEN CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) > 0

THEN SUBSTRING(FullName, CHARINDEX(' ', FullName) + 1, CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) - CHARINDEX(' ', FullName) - 1)

ELSE ''

END

ELSE ''

END 'Ім''я',

CASE

WHEN CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) > 0

THEN SUBSTRING(FullName, CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) + 1, LEN(FullName))

ELSE ''

END 'По батькові'

FROM MyTable;

Пояснення

CASE

WHEN CHARINDEX(' ', FullName) > 0

THEN LEFT(FullName, CHARINDEX(' ', FullName) - 1)

ELSE FullName

END 'Прізвище'

Цей фрагмент коду призначений для виділення Прізвища (FirstName) з поля FullName. Логіка його роботи наступна:

1. CHARINDEX(' ', FullName) > 0: Ця умова перевіряє, чи існує пробіл у полі FullName. Якщо пробіл існує, це може свідчити про те, що поле FullName містить як Прізвище, так і Ім'я, і, можливо, По батькові.

2. LEFT(FullName, CHARINDEX(' ', FullName) - 1): Якщо умова вище виконується, то застосовується функція LEFT, яка виділяє частину рядка FullName до першого пробілу (тобто Прізвище). Функція CHARINDEX визначає позицію першого пробілу, і LEFT витягує всі символи з початку рядка до цієї позиції мінус один символ, оскільки останній символ (пробіл) не потрібно включати до Прізвища.

3. Якщо умова вище не виконується (тобто поле FullName не містить пробіла), то вважається, що FullName вже містить Прізвище, і це значення просто залишається в полі FirstName.

Отже, цей фрагмент коду дозволяє правильно виділити Прізвище (FirstName) з поля FullName навіть у випадках, коли FullName містить ім'я та прізвище в одному рядку або лише прізвище.

CASE

WHEN CHARINDEX(' ', FullName) > 0

THEN CASE

WHEN CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) > 0

THEN SUBSTRING(FullName, CHARINDEX(' ', FullName) + 1, CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) - CHARINDEX(' ', FullName) - 1)

ELSE ''

END

ELSE ''

END 'Ім''я',

Цей фрагмент коду призначений для виділення середнього імені (SecondName) з поля FullName, якщо поле FullName містить два ім'я розділені пробілами. Я поясню кожен крок логіки:

1. CHARINDEX(' ', FullName) > 0: Ця умова перевіряє, чи є пробіл у полі FullName. Якщо пробіл є, це означає, що поле FullName містить два ім'я: Прізвище та Ім'я.

2. CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) > 0: Ця умова перевіряє, чи є ще один пробіл у полі FullName, починаючи з позиції, яка визначається попереднім CHARINDEX. Це означає, що поле FullName містить три імені: Прізвище, Перше Ім'я та Друге Ім'я (По батькові).

3. SUBSTRING(FullName, CHARINDEX(' ', FullName) + 1, CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) - CHARINDEX(' ', FullName) - 1): Якщо обидві умови вище виконуються, то виконується функція SUBSTRING, яка виділяє друге ім'я з поля FullName. Функція SUBSTRING приймає три аргументи: рядок, з якого витягувати підстроку (в даному випадку - FullName), позицію, з якої почати виділення (позиція пробілу після Прізвища), та довжину підстроки (розташування наступного пробілу мінус позиція першого пробілу мінус одиниця). Це допомагає виділити друге ім'я з рядка.

Якщо ж жодна з умов вище не виконується (тобто поле FullName містить або лише Прізвище, або Прізвище та Перше Ім'я, але не містить Другого Імені), то повертається порожній рядок.

Загальна логіка полягає в розділенні рядка FullName на частини, визначені пробілами, і виділенні середнього імені, якщо воно існує.

CASE

WHEN CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) > 0

THEN SUBSTRING(FullName, CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) + 1, LEN(FullName))

ELSE ''

END 'По батькові'

Цей фрагмент коду призначений для виділення По батькові (Third\_Name) з поля FullName, якщо поле FullName містить три імені: Прізвище, Перше Ім'я та Друге Ім'я (По батькові). Я поясню кожен крок логіки:

1. CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) > 0: Ця умова перевіряє, чи є ще один пробіл у полі FullName, починаючи з позиції, яка визначається попереднім CHARINDEX. Це означає, що поле FullName містить три імені: Прізвище, Перше Ім'я та Друге Ім'я (По батькові).

2. SUBSTRING(FullName, CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) + 1, LEN(FullName)): Якщо умова вище виконується, то виконується функція SUBSTRING, яка виділяє По батькові (Third\_Name) з поля FullName. Функція SUBSTRING приймає три аргументи: рядок, з якого витягувати підстроку (в даному випадку - FullName), позицію, з якої почати виділення (позиція другого пробілу після Прізвища плюс одиниця), та довжину підстроки. В цьому випадку довжина рівна довжині рядка FullName, оскільки ми витягуємо По батькові з місця, де починається третє ім'я і до кінця рядка.

Якщо умова вище не виконується (тобто поле FullName містить або лише Прізвище, або Прізвище та Перше Ім'я, але не містить Другого Імені і По батькові), то повертається порожній рядок.

Загальна логіка полягає в розділенні рядка FullName на частини, визначені пробілами, і виділенні По батькові, якщо воно існує.

* 1. \*\* Організувати фізичний запис (UPDATE) з одного поля таблиці до завдання 12. Заповнити в таблиці поля:

FirstName - прізвище,

SecondName – ім’я,

Third\_Name - По батькові.

РОЗВ’ЯЗОК:

Запит сумісний з MS SQL Server 2014

UPDATE MyTable

SET

FirstName = CASE

WHEN CHARINDEX(' ', FullName) > 0

THEN LEFT(FullName, CHARINDEX(' ', FullName) - 1)

ELSE FullName

END,

SecondName = CASE

WHEN CHARINDEX(' ', FullName) > 0

THEN CASE

WHEN CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) > 0

THEN SUBSTRING(FullName, CHARINDEX(' ', FullName) + 1, CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) - CHARINDEX(' ', FullName) - 1)

ELSE ''

END

ELSE ''

END,

Third\_Name = CASE

WHEN CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) > 0

THEN SUBSTRING(FullName, CHARINDEX(' ', FullName, CHARINDEX(' ', FullName) + 1) + 1, LEN(FullName))

ELSE ''

END;