GFT

PRZETWARZANIE DANYCH – BIG DATA

Zaawansowany SQL

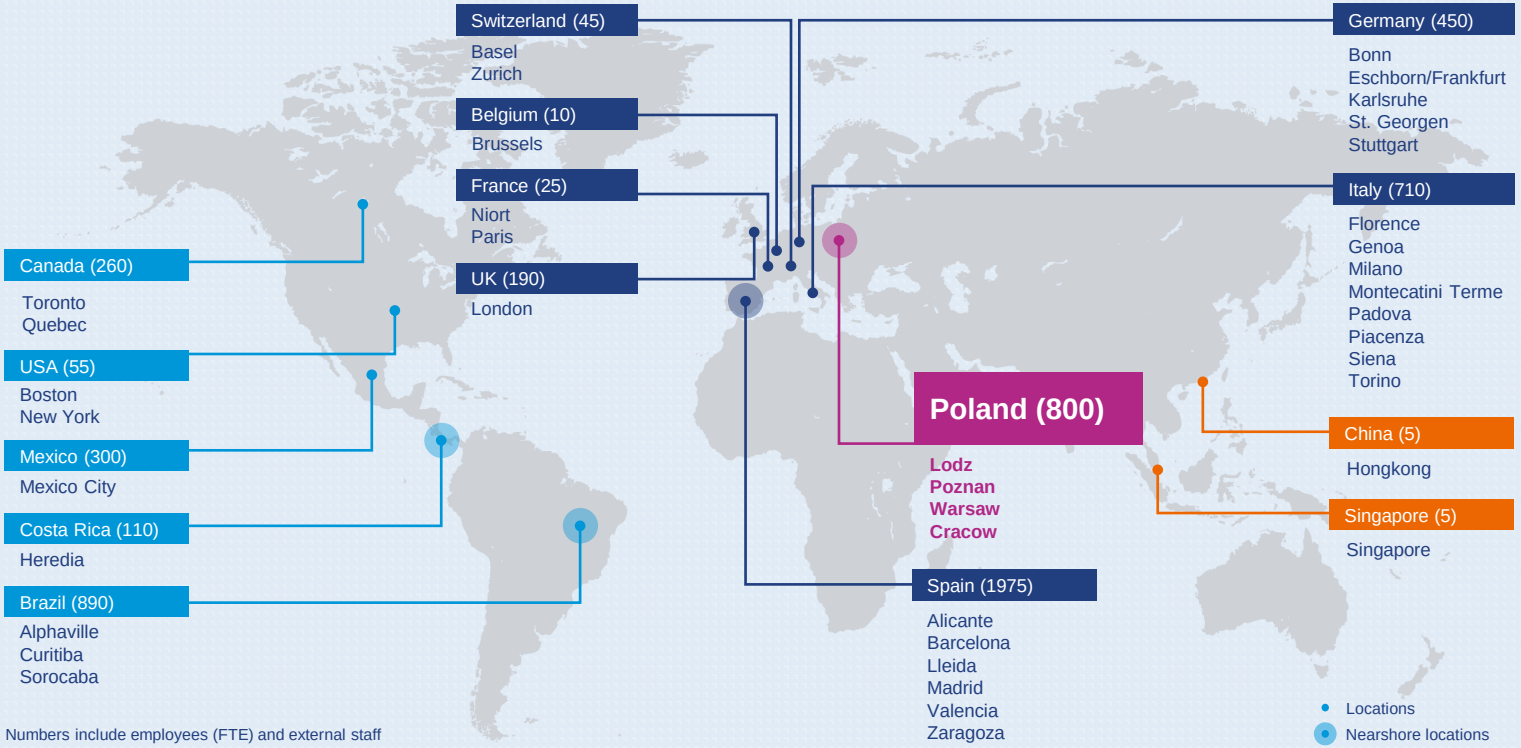
Arkadiusz Kasprzak
07.11.2020

GFT

AGENDA

1. Sprawy organizacyjne
2. Podstawy SQL – krótkie przypomnienie
3. Funkcje grupowania i agregacji
4. Common Table Expression (CTE)
5. Funkcje analityczne
6. Funkcje użytkownika
7. Operator APPLY
8. Co jeszcze warto wiedzieć…

GFT Group: 5700 experts in 15 countries



Numbers include employees (FTE) and external staff

Arkadiusz Kasprzak – o mnie

GFT

- Head of Data Poland w GFT
- Ponad 10 lat doświadczenia w IT
- Autor bloga o przetwarzaniu danych: https://oceandanych.pl
- Współtwórca GDG Cloud Poznań
- Namiary na mnie:
  - akasprzak@wmi.amu.edu.pl / arek@oceandanych.pl
  - LinkedIn

Sprawy organizacyjne

- Warunki zaliczenia
  - 4.0 – x*0.5 = ocena, gdzie x – liczba nieobecności (dla x=1,2)
  - Przy >=3 nieobecnościach - ndst
  - Aktywność na wykładach i/lub laboratoriach +1.0
- Korzystamy z bazy Northwind – skrypty na MS Teams w katalogu Baza danych
- Ankieta

Zasady pracy na zajęciach

GFT

- Wypracujmy kontrakt. Moja propozycja:
  - Włączone kamery
  - Wyciszone mikrofony
  - Zasada Vegas
  - Nie nagrywamy wykładów
  - Materiały zostały udostępnione*
  - Nie oszukujemy z obecnością
  - Zwracamy się do siebie formalnie czy nie?

GFT

Podstawy SQL – krótkie przypomnienie

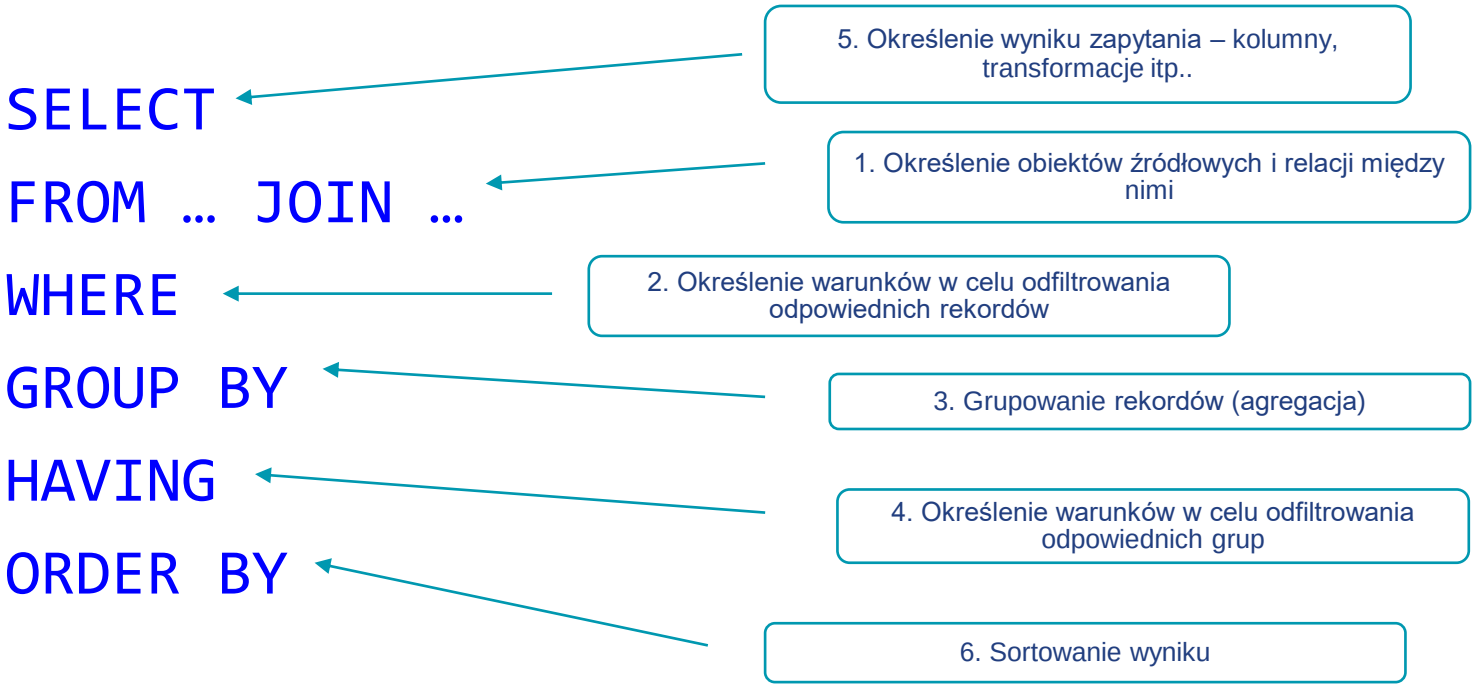
Pojęcia związane z przetwarzaniem zapytań SQL - teoria

GFT

Pojęcia związane z przetwarzaniem zapytań SQL:

- Selekcja – wybór odpowiednich wierszy `WHERE`
- Projekcja – wybór odpowiednich kolumn `SELECT`
- Złączenie – operacja pozwalająca pobrać dane z wielu tabel `JOIN`
- Operatory algebraiczne – suma, różnica… `UNION, UNION ALL, EXCEPT, INTERSECT`
- Agregacja – agregacja danych przy użyciu dostępnych funkcji agregujących `MIN, MAX, SUM, …`

Pojęcia związane z przetwarzaniem zapytań SQL - praktyka



Klauzula WHERE

GFT

- Klauzula **WHERE** umożliwia odfiltrowanie rekordów za pomocą warunku lub kilka warunków połączonych operatorami logicznymi

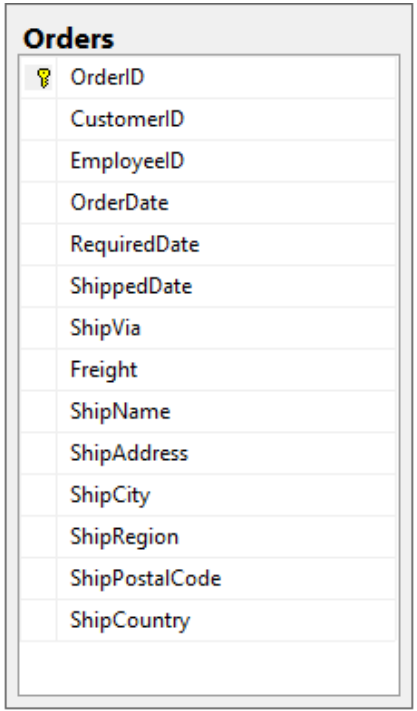
- Operatory logiczne: **AND**, **OR**

- Operatory porównania: **=, != , <>, <, >, LIKE, IN**…

Klauzula WHERE

GFT

```
SELECT OrderID, CustomerID, ShipName, …
SELECT *
FROM   Orders
WHERE  DATEPART(YEAR, OrderDate) > 1997
```

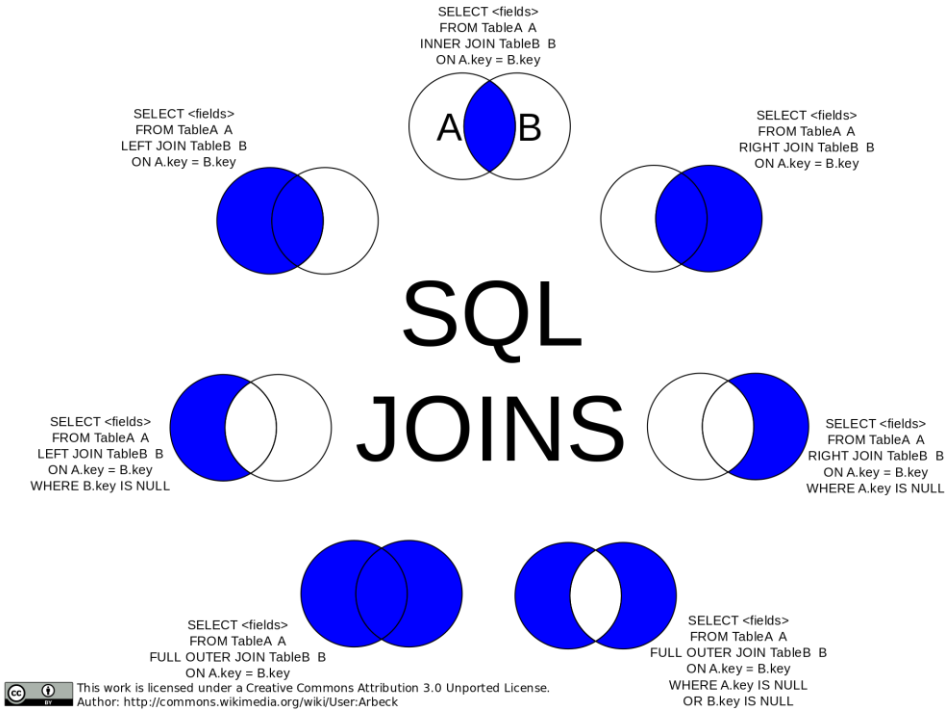


Złączenia - JOINs

GFT

- Rodzaje złączeń:
  - Połączenia wewnętrzne – **INNER JOIN**
  - Połączenia zewnętrzne – **OUTER JOIN**:
    - LEFT [OUTER] JOIN
    - RIGHT [OUTER] JOIN
    - FULL [OUTER] JOIN
  - Iloczyn kartezjański - **CROSS JOIN**

Złączenia - JOINs



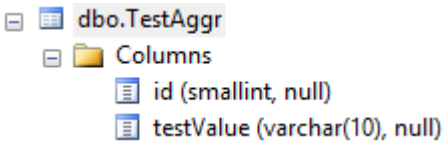
Funkcje grupowania i agregacji

GFT

- Funkcje agregacji służą do wykonywania kalkulacji na zbiorze danych i zwracają pojedynczą wartość

- Za pomocą klauzuli **GROUP BY** możemy grupować rekordy i na każdej z grup przeprowadzić kalkulację. Grupy są wyznaczane unikalne wartości atrybutów podanych w **GROUP BY**

Funkcje agregujące – krótkie przypomnienie



	id	testValue
1	1	test1
2	2	test2
3	NULL	test3
4	7	NULL
5	NULL	NULL

```
SELECT  COUNT(*) AS CNT1, COUNT(1) AS CNT2, COUNT(id) AS CNT3,
        COUNT(testValue) AS CNT4
FROM    TestAggr;
```

	CNT1	CNT2	CNT3	CNT4
1	5	5	3	3

```
SELECT  AVG(id) AS AVG FROM TestAggr;
```

	AVG
1	3

```
SELECT  AVG(CAST(id as DECIMAL(10,2))) AS AVG
FROM    TestAggr;
```

	AVG
1	3.333333

Funkcje agregujące – krótkie przypomnienie

GFT

```
SELECT   ISNULL(MAX(id),0)+1 AS ID
FROM     EmptyTable;
```

	ID
1	1

```
SELECT   MAX(ISNULL(id,0))+1 AS ID
FROM     EmptyTable;
```

	ID
1	NULL

```
SELECT id, value
FROM   NotEmptyTable
```

	id	value
1	1	ExistingValue

```
SELECT   ISNULL(MAX(id),0)+1 AS ID
FROM     NotEmptyTable
WHERE    value = 'NonExistingValue';
```

	ID
1	1

```
SELECT   MAX(ISNULL(id,0))+1 AS ID
FROM     NotEmptyTable
WHERE    value = 'NonExistingValue';
```

	ID
1	NULL

GROUP BY – krótkie przypomnienie

GFT

Zadanie:

Korzystając z tabeli ***Orders*** wyświetl wszystkie identyfikatory klientów (wraz z liczbą dokonanych przez nich zamówień), którzy dokonali co najmniej 10-ciu zamówień pomiędzy majem 1997 a czerwcem 1998 (***OrderDate*** *[datetime]*). Wyniki posortuj malejąco zgodnie z liczbą zamówień.

```
SELECT    CustomerID, COUNT(*) AS CNT
FROM      Orders
WHERE     (DATEPART(YEAR, OrderDate)*100
          +DATEPART(MM, OrderDate)) BETWEEN 199705 AND 199806
GROUP BY CustomerID
HAVING    CNT > 10
HAVING    COUNT(*) > 10
ORDER BY CNT DESC;
```

	CustomerID	CNT
1	SAVEA	25
2	QUICK	19
3	ERNSH	19
4	FOLKO	14
5	BERGS	13
6	BONAP	12
7	HANAR	12
8	HILAA	12
9	HUNGO	11
10	GREAL	11

Orders

- OrderID
- CustomerID
- EmployeeID
- OrderDate
- RequiredDate
- ShippedDate
- ShipVia
- Freight
- ShipName
- ShipAddress
- ShipCity
- ShipRegion
- ShipPostalCode
- ShipCountry

GROUP BY – krótkie przypomnienie

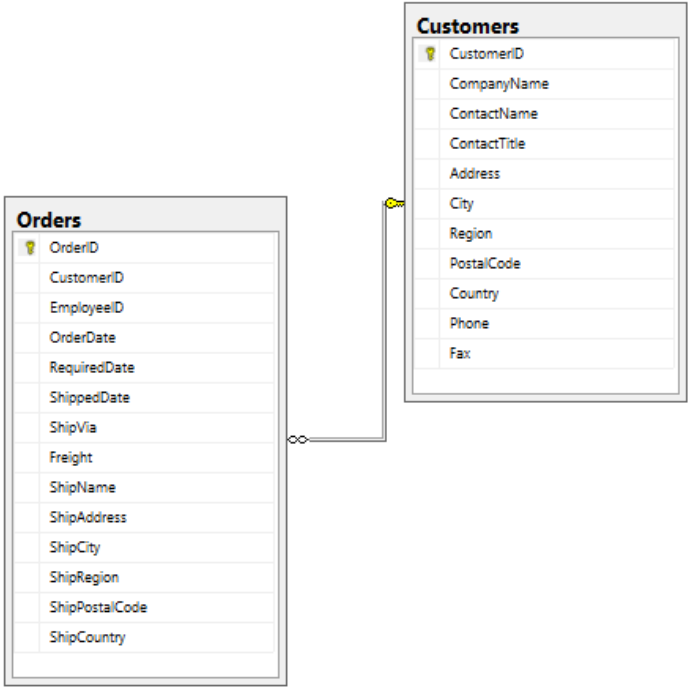
GFT

Zadanie:

Korzystając z tabeli ***Customers*** rozbuduj i zaktualizuj poprzednie zapytanie tak, aby zamiast identyfikatora klienta wyświetlała się jego nazwa (***Customers.CompanyName*** *[nvarchar]*).

```
SELECT    C.CompanyName, COUNT(*) AS CNT
FROM      Orders O JOIN Customers C
ON        (O.CustomerID = C.CustomerID)
WHERE     (DATEPART(YEAR, OrderDate)*100
          +DATEPART(MM, OrderDate)) BETWEEN 199705 AND
          199806
GROUP BY C.CompanyName
HAVING    COUNT(*) > 10
ORDER BY CNT DESC;
```

	CompanyName	CNT
1	Save-a-lot Markets	25
2	QUICK-Stop	19
3	Ernst Handel	19
4	Folk och fä HB	14
5	Berglunds snabbköp	13
6	Bon app'	12
7	Hanari Carnes	12
8	HILARION-Abastos	12
9	Hungry Owl All-Night Grocers	11
10	Great Lakes Food Market	11



Common Table Expression – krótkie przypomnienie

GFT

- **CTE** to zapytanie reprezentujące tymczasowy zestaw rekordów
- Po konstrukcji **WITH** należy użyć jednego z poleceń **SELECT, INSERT UPDATE** lub **DELETE**
- Można definiować więcej niż jedno **CTE** w ramach polecenia **WITH**, natomiast nie można definiować kolejnej klauzuli **WITH** w ramach **CTE**
- Można definiować wiele zapytań **CTE** nierekursywnie, korzystając z operatów: **UNION, UNION ALL, INTERSECT lub EXCEPT**
- W zapytaniu głównym można wielokrotnie odwoływać się do **CTE**

Common Table Expression – krótkie przypomnienie

GFT

- W konstrukcji **CTE** nie można używać poleceń:
  - ORDER BY (z wyjątkiem gdy klauzula TOP jest podana)
  - INTO
  - OPTION
  - FOR BROWSE

- … -> dokumentacja: https://msdn.microsoft.com/pl-pl/library/ms175972(v=sql.110).aspx

- Składnia:

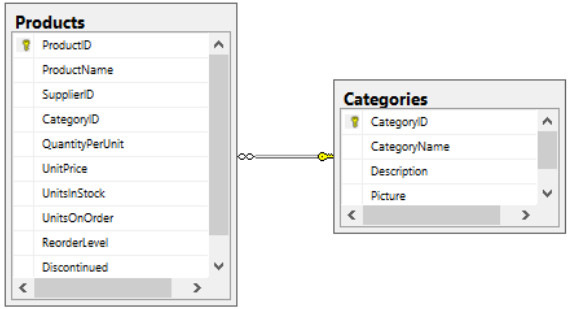
```
[ WITH <common_table_expression> [ ,...n ] ]

<common_table_expression>::=
    expression_name [ ( column_name [ ,...n ] ) ]
    AS
    ( CTE_query_definition )
```

CTE - przykład

GFT

```
WITH
ProdAvgUnitPrice (AvgUnitPrice)
AS
(
          SELECT AVG(UnitPrice)
          FROM   Products
),
GreaterThanAvg (ProductName, CategoryID, UnitPrice)
AS
(
          SELECT ProductName, CategoryID, UnitPrice
          FROM   Products P
          WHERE  UnitPrice > (SELECT AvgUnitPrice FROM ProdAvgUnitPrice)
)
SELECT     G.ProductName, C.CategoryName, G.UnitPrice
FROM       GreaterThanAvg G JOIN Categories C
ON         (C.CategoryID = G.CategoryID);
```



	ProductName	CategoryName	UnitPrice
1	Uncle Bob's Organic Dried Pears	Produce	30,00
2	Northwoods Cranberry Sauce	Condiments	40,00
3	Mishi Kobe Niku	Meat/Poultry	97,00
4	Ikura	Seafood	31,00
5	Queso Manchego La Pastora	Dairy Products	38,00
6	Alice Mutton	Meat/Poultry	39,00
7	Carnarvon Tigers	Seafood	62,50
8	Sir Rodney's Marmalade	Confections	81,00
9	Gumbär Gummibärchen	Confections	31,23
10	Schoggi Schokolade	Confections	43,90
11	Rössle Sauerkraut	Produce	45,60
12	Thüringer Rostbratwurst	Meat/Poultry	123,79
13	Mascarpone Fabioli	Dairy Products	32,00
14	Côte de Blaye	Beverages	263,50
15	Ipoh Coffee	Beverages	46,00

Podzapytania

GFT

- Podzapytania mogą być:
  - **skorelowane** oraz **nieskorelowane**

- Podzapytania mogą zwracać:
  - Pojedynczą wartość: zapytania **skalarne**
  - Listę wartości
  - Dane tabelaryczne

- Przy zapytaniach skalarnych możemy używać do porównania operatorów =, <, >, <>, !=…
- Przy zapytaniach zwracających więcej wartości musimy użyć dodatkowo jednego z operatorów:
  - **ALL**
  - **ANY (SOME)**

Podzapytania - przykłady

GFT

```
SELECT  COUNT(1) AS CNT
FROM    Products
WHERE   UnitPrice >
        (SELECT AVG(UnitPrice) FROM Products)
```

	CNT
1	25

```
SELECT  p1.ProductName, p1.CategoryID
FROM    Products p1
WHERE   UnitPrice =
        (SELECT MAX(p2.UnitPrice) FROM Products p2
        WHERE p1.CategoryID = p2.CategoryID)
```

	ProductName	CategoryID
1	Carnarvon Tigers	8
2	Manjimup Dried Apples	7
3	Thüringer Rostbratwurst	6
4	Gnocchi di nonna Alice	5
5	Raclette Courdavault	4
6	Sir Rodney's Marmalade	3
7	Vegie-spread	2
8	Côte de Blaye	1

Podzapytania - przykłady

GFT

```
SELECT   p1.ProductName, p1.CategoryID
FROM     Products p1
WHERE    UnitPrice > ALL
         (SELECT AVG(p2.UnitPrice) FROM Products p2
                 WHERE    p1.CategoryID !=
                          p2.CategoryID
                 GROUP BY p2.CategoryID)
```

	ProductName	CategoryID
1	Mishi Kobe Niku	6
2	Alice Mutton	6
3	Carnarvon Tigers	8
4	Sir Rodney's Marmalade	3
5	Thüringer Rostbratwurst	6
6	Côte de Blaye	1
7	Raclette Courdavault	4

Podzapytania - przykłady

GFT

```
SELECT ProductName, (SELECT CategoryName
                     FROM Categories c
                     WHERE c.CategoryID =
                           p.CategoryID)
               AS CategoryName
FROM Products p
```

	ProductName	CategoryName
1	Chai	Beverages
2	Chang	Beverages
3	Aniseed Syrup	Condiments
4	Chef Anton's Cajun Seasoning	Condiments
5	Chef Anton's Gumbo Mix	Condiments
6	Grandma's Boysenberry Spread	Condiments
7	Uncle Bob's Organic Dried Pears	Produce
8	Northwoods Cranberry Sauce	Condiments
9	Mishi Kobe Niku	Meat/Poultry
10	Ikura	Seafood
11	Queso Cabrales	Dairy Products
12	Queso Manchego La Pastora	Dairy Products
13	Konbu	Seafood
14	Tofu	Produce
15	Genen Shouyu	Condiments
16	Pavlova	Confections
17	Alice Mutton	Meat/Poultry
18	Carnarvon Tigers	Seafood
19	Teatime Chocolate Biscuits	Confections

Podzapytania - przykłady

GFT

```
SELECT   ProductName, CategoryID,
         (SELECT MAX(UnitPrice) FROM Products)
         AS MaxUnitPrice
FROM Products
WHERE CategoryID != 1
```

	ProductName	CategoryID	MaxUnitPrice
1	Aniseed Syrup	2	263,50
2	Chef Anton's Cajun Seasoning	2	263,50
3	Chef Anton's Gumbo Mix	2	263,50
4	Grandma's Boysenberry Spread	2	263,50
5	Uncle Bob's Organic Dried Pears	7	263,50
6	Northwoods Cranberry Sauce	2	263,50
7	Mishi Kobe Niku	6	263,50
8	Ikura	8	263,50
9	Queso Cabrales	4	263,50
10	Queso Manchego La Pastora	4	263,50

```
SELECT MAX(UnitPrice)
                AS MaxUnitPricePerCategory,
         CategoryID
FROM     Products
GROUP BY CategoryID
```

	MaxUnitPricePerCategory	CategoryID
1	263,50	1
2	43,90	2
3	81,00	3
4	55,00	4
5	38,00	5
6	123,79	6
7	53,00	7
8	62,50	8

Podzapytania – EXISTS vs. IN

GFT

- Operator **[NOT] EXISTS** jest używany aby zweryfikować czy istnieje jakiś rekord w podzapytaniu
- Operator **[NOT] IN** pozwala wyspecyfikować wiele wartości w klauzuli WHERE (wpisanych ręcznie lub też poprzez zapytanie)

```
SELECT    COUNT(1) AS CNT
FROM      Products
WHERE     CategoryID IN (1,3)
```

	CNT
1	25

```
SELECT    COUNT(1) AS CNT
FROM      Products
WHERE     CategoryID IN (
                  SELECT    CategoryID
                  FROM      Products P
                  JOIN      Suppliers S ON (P.SupplierID = S.SupplierID)
                  WHERE     S.Country = 'UK'
)
```

	CNT
1	37

Podzapytania – EXISTS vs. IN

GFT

```
SELECT COUNT(*) AS CNT
FROM   Orders
WHERE  ShipRegion IN
       (SELECT ShipRegion
        FROM Orders
        WHERE CustomerID = 'HANAR')
```

	CNT
1	25

```
SELECT COUNT(*) AS CNT
FROM   Orders O
WHERE  EXISTS (
              SELECT 1
              FROM   Orders P
              WHERE P.CustomerID =
                    'HANAR'
              AND O.ShipRegion =
                    P.ShipRegion)
```

NOT EXISTS ≠ NOT IN

Logika trójwartościowa

GFT

p	q	p AND q	p OR q	p = q	NOT p
True	True				
True	False				
True	Unknown				
False	True				
False	False				
False	Unknown				
Unknown	True				
Unknown	False				
Unknown	Unknown				

Logika trójwartościowa

GFT

p	q	p AND q	p OR q	p = q	NOT p
True	True	**True**	**True**	**True**	**False**
True	False	**False**	**True**	**False**	**False**
True	Unknown				**False**
False	True	**False**	**True**	**False**	**True**
False	False	**False**	**False**	**True**	**True**
False	Unknown				**True**
Unknown	True				
Unknown	False				
Unknown	Unknown				

Logika trójwartościowa

GFT

p	q	p AND q	p OR q	p = q	NOT p
True	True	**True**	**True**	**True**	**False**
True	False	**False**	**True**	**False**	**False**
True	Unknown	**Unknown**	**True**	**Unknown**	**False**
False	True	**False**	**True**	**False**	**True**
False	False	**False**	**False**	**True**	**True**
False	Unknown				**True**
Unknown	True	**Unknown**	**True**	**Unknown**	
Unknown	False				
Unknown	Unknown				

Logika trójwartościowa

GFT

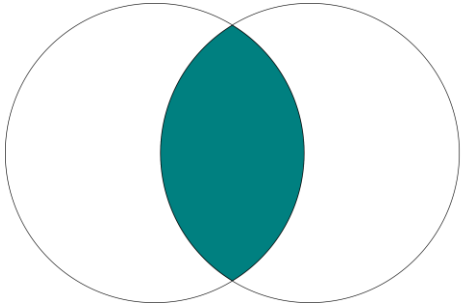
p	q	p AND q	p OR q	p = q	NOT p
True	True	**True**	**True**	**True**	**False**
True	False	**False**	**True**	**False**	**False**
True	Unknown	**Unknown**	**True**	**Unknown**	**False**
False	True	**False**	**True**	**False**	**True**
False	False	**False**	**False**	**True**	**True**
False	Unknown	**False**	**Unknown**	**Unknown**	**True**
Unknown	True	**Unknown**	**True**	**Unknown**	
Unknown	False	**False**	**Unknown**	**Unknown**	
Unknown	Unknown				

Logika trójwartościowa

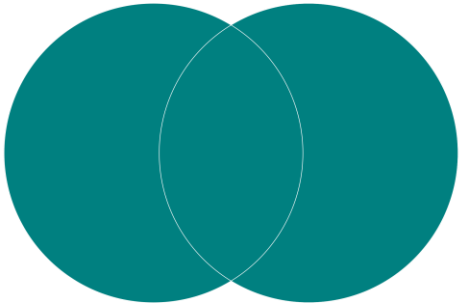
GFT

p	q	p AND q	p OR q	p = q	NOT p
True	True	**True**	**True**	**True**	**False**
True	False	**False**	**True**	**False**	**False**
True	Unknown	**Unknown**	**True**	**Unknown**	**False**
False	True	**False**	**True**	**False**	**True**
False	False	**False**	**False**	**True**	**True**
False	Unknown	**False**	**Unknown**	**Unknown**	**True**
Unknown	True	**Unknown**	**True**	**Unknown**	**Unknown**
Unknown	False	**False**	**Unknown**	**Unknown**	**Unknown**
Unknown	Unknown	**Unknown**	**Unknown**	**Unknown**	**Unknown**

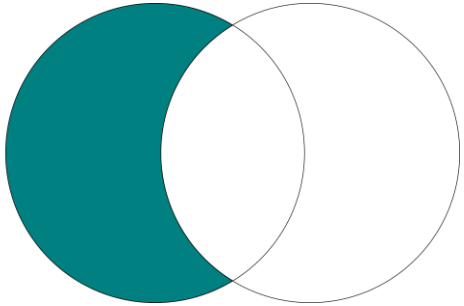
Operatory na zbiorach



INTERSECT



UNION
UNION ALL



MINUS
EXCEPT

CASE


```
Simple CASE expression:
CASE input_expression
     WHEN when_expression THEN result_expression [ ...n ]
     [ ELSE else_result_expression ]
END
Searched CASE expression:
CASE
     WHEN Boolean_expression THEN result_expression [ ...n ]
     [ ELSE else_result_expression ]
END
```

CASE - przykłady

GFT

```
SELECT C.CategoryName,
       COUNT(*) AS NumberOfProducts,
       CASE
              WHEN COUNT(*) > 10 THEN
                     'High'
              WHEN COUNT(*) BETWEEN 6 AND
                     10 THEN 'Average'
              ELSE 'Low'
       END AS Level
FROM   Products P JOIN Categories C
ON     P.CategoryID = C.CategoryID
GROUP BY C.CategoryID, C.CategoryName
```

	CategoryName	NumberOfProducts	Level
1	Beverages	12	High
2	Condiments	12	High
3	Confections	13	High
4	Dairy Products	10	Average
5	Grains/Cereals	7	Average
6	Meat/Poultry	6	Average
7	Produce	5	Low
8	Seafood	12	High

CASE - przykład

GFT

```
WITH NumberOfProductsInCategory AS
(
            SELECT      C.CategoryName,
                        COUNT(*) AS NumberOfProducts
            FROM        Products P JOIN Categories C
            ON          P.CategoryID = C.CategoryID
            GROUP BY    C.CategoryID, C.CategoryName
)
SELECT      CategoryName,
            NumberOfProducts,
            CASE NumberOfProducts
                WHEN (SELECT MAX(NumberOfProducts)
                      FROM   NumberOfProductsInCategory )
                      THEN  'Best'
                WHEN (SELECT MIN(NumberOfProducts)
                      FROM   NumberOfProductsInCategory )
                      THEN   'Worst'
                ELSE  'Not too bad'
            END AS Level
FROM NumberOfProductsInCategory;
```

	CategoryName	NumberOfProducts	Level
1	Beverages	12	Not too bad
2	Condiments	12	Not too bad
3	Confections	13	Best
4	Dairy Products	10	Not too bad
5	Grains/Cereals	7	Not too bad
6	Meat/Poultry	6	Not too bad
7	Produce	5	Worst
8	Seafood	12	Not too bad

Przykładowe funkcje/polecenia

GFT

- ISNULL
- COALESCE
- DISTINCT
- DATEPART / YEAR / MONTH / DAY / DATEADD
- …

GFT

Funkcje grupowania i agregacji

GROUP BY – krótkie przypomnienie

GFT

Zadanie:

Korzystając z tabeli ***Orders*** wyświetl wszystkie identyfikatory klientów (wraz z liczbą dokonanych przez nich zamówień), którzy dokonali co najmniej 10-ciu zamówień pomiędzy majem 1997 a czerwcem 1998 (***OrderDate*** *[datetime]*). Wyniki posortuj malejąco zgodnie z liczbą zamówień.

```
SELECT    CustomerID, COUNT(*) AS CNT
FROM      Orders
WHERE     (DATEPART(YEAR, OrderDate)*100
          +DATEPART(MM, OrderDate)) BETWEEN 199705 AND 199806
GROUP BY CustomerID
HAVING    CNT > 10
HAVING    COUNT(*) > 10
ORDER BY CNT DESC;
```

	CustomerID	CNT
1	SAVEA	25
2	QUICK	19
3	ERNSH	19
4	FOLKO	14
5	BERGS	13
6	BONAP	12
7	HANAR	12
8	HILAA	12
9	HUNGO	11
10	GREAL	11

Orders

- OrderID
- CustomerID
- EmployeeID
- OrderDate
- RequiredDate
- ShippedDate
- ShipVia
- Freight
- ShipName
- ShipAddress
- ShipCity
- ShipRegion
- ShipPostalCode
- ShipCountry

GROUP BY – krótkie przypomnienie

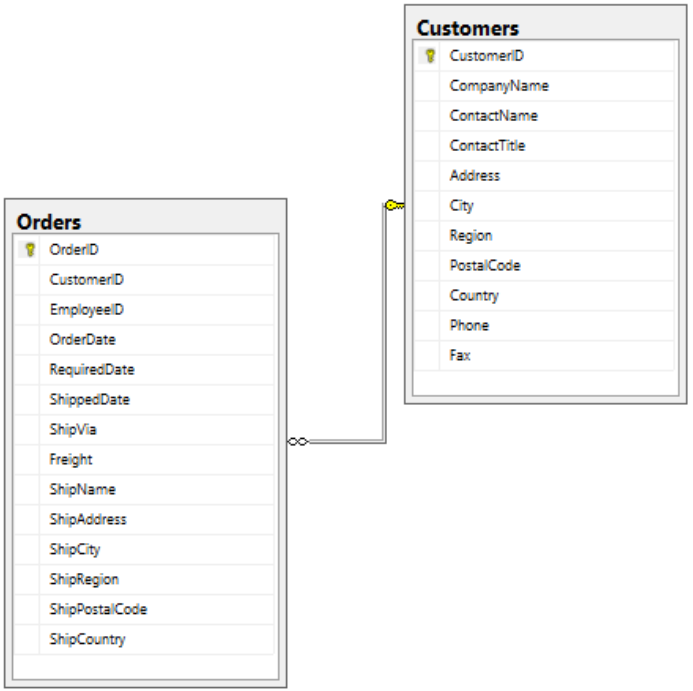
GFT

Zadanie:

Korzystając z tabeli ***Customers*** rozbuduj i zaktualizuj poprzednie zapytanie tak, aby zamiast identyfikatora klienta wyświetlała się jego nazwa (***Customers.CompanyName*** *[nvarchar]*).

```
SELECT    C.CompanyName, COUNT(*) AS CNT
FROM      Orders O JOIN Customers C
ON        (O.CustomerID = C.CustomerID)
WHERE     (DATEPART(YEAR, OrderDate)*100
          +DATEPART(MM, OrderDate)) BETWEEN 199705 AND
          199806
GROUP BY C.CompanyName
HAVING    COUNT(*) > 10
ORDER BY CNT DESC;
```

	CompanyName	CNT
1	Save-a-lot Markets	25
2	QUICK-Stop	19
3	Ernst Handel	19
4	Folk och fä HB	14
5	Berglunds snabbköp	13
6	Bon app'	12
7	Hanari Carnes	12
8	HILARION-Abastos	12
9	Hungry Owl All-Night Grocers	11
10	Great Lakes Food Market	11



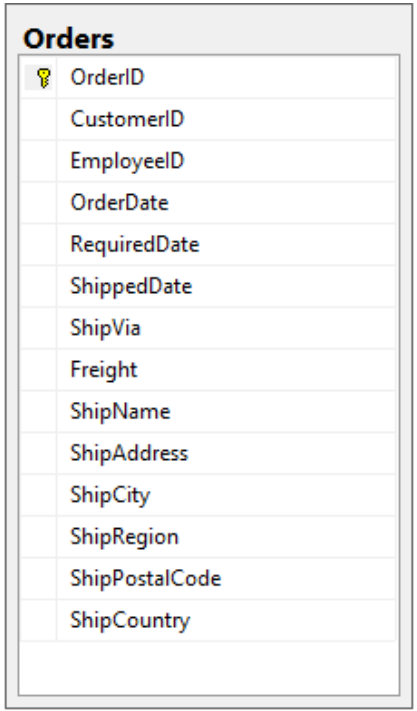
Zadanie

GFT

Zadanie:

Korzystając z tabeli ***Orders*** zaprojektuj zapytanie przedstawiające sumaryczną liczbą zamówień w roku 1997 (***OrderDate*** *[datetime]*) dla poszczególnych państw (***ShipCountry*** *[nvarchar]*), i miast (***ShipCity*** *[nvarchar]*), samych państw oraz całościowe podsumowanie. Wynik posortuj rosnąco tak, aby w pierwszej kolejności były prezentowane wyniki dla danego kraju i miast, następnie tylko dla danego kraju - podsumowanie jako ostatnia pozycja.

	ShipCountry	ShipCity	CNT
1	Argentina	Buenos Aires	6
2	Argentina	NULL	6
3	Austria	Graz	15
4	Austria	Salzburg	6
5	Austria	NULL	21
6	Belgium	Bruxelles	3
7	Belgium	Charleroi	4
8	Belgium	NULL	7
	...		
90	Venezuela	NULL	20
91	NULL	NULL	408



ROLLUP

GFT

- **ROLLUP** jest rozwinięciem polecenia GROUP BY, które pozwala wyliczyć dodatkowe podsumowania częściowe i ogólne dla podgrup generowanych „od prawej do lewej”
- Zapytanie:

```
SELECT
...
GROUP BY ROLLUP (a,b,c)
```

Wygeneruje grupowania:

```
(a,b,c)
(a,b)
(a)
()
```

Wykonane zostaną grupowania:

- `GROUP BY a,b,c`
- `GROUP BY a,b`
- `GROUP BY a`
- `GROUP BY ()`

Rekord agregujący cały zbiór

Co jest tożsame z brakiem grupowania

N+1 grupowań

ROLLUP

GFT

- Czy możemy ten sam efekt uzyskać nie korzystając z polecenia ROLLUP?

```
SELECT 1 as Level, NULL as colA,      NULL as colB,      NULL as colC,      COUNT(...)
...

UNION ALL

SELECT 2,          a,               NULL,              NULL,              COUNT(...)
...
GROUP BY a

UNION ALL

SELECT 3,          a,               b,                 NULL,              COUNT(...)
...
GROUP BY a, b

UNION ALL

SELECT 4,          a,               b,                 c,                 COUNT(...)
...
GROUP BY a, b, c
```

Odp.: Tak, ale po co? ☺

ROLLUP jest znacząco bardziej wydajny

ROLLUP – przykład

GFT

Bez ROLLUP:

```
SELECT s.* FROM (
  SELECT   ShipCountry, ShipCity, COUNT(OrderID) as CNT
  FROM     Orders
  WHERE    DATEPART(yyyy, OrderDate) = 1997
  GROUP BY ShipCountry, ShipCity
  UNION ALL
  SELECT   ShipCountry, NULL , COUNT(OrderID) as CNT
  FROM     Orders
  WHERE    DATEPART(yyyy, OrderDate) = 1997
  GROUP BY ShipCountry
  UNION ALL
  SELECT   NULL, NULL, COUNT(OrderID) as CNT
  FROM     Orders
  WHERE    DATEPART(yyyy, OrderDate) = 1997) s
ORDER BY CASE WHEN ShipCountry IS NULL THEN 1 ELSE 0 END ASC,
ShipCountry ASC, CASE WHEN ShipCity IS NULL THEN 1 ELSE 0 END
ASC, ShipCity ASC
```

ROLLUP:

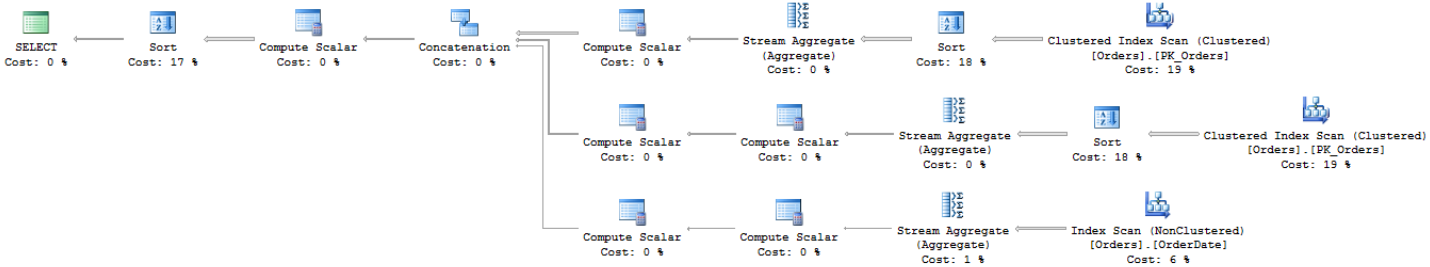
```
SELECT    ShipCountry, ShipCity, COUNT(OrderID) as CNT
FROM      Orders
WHERE     DATEPART(yyyy, OrderDate) = 1997
GROUP BY  ROLLUP (ShipCountry, ShipCity)
ORDER BY  CASE WHEN ShipCountry IS NULL THEN 1 ELSE 0 END ASC,
          ShipCountry ASC,
          CASE WHEN ShipCity IS NULL THEN 1 ELSE 0 END ASC,
          ShipCity ASC
```

	ShipCountry	ShipCity	CNT
1	Argentina	Buenos Aires	6
2	Argentina	NULL	6
3	Austria	Graz	15
4	Austria	Salzburg	6
5	Austria	NULL	21
6	Belgium	Bruxelles	3
7	Belgium	Charleroi	4
8	Belgium	NULL	7
	...		
90	Venezuela	NULL	20
91	NULL	NULL	408

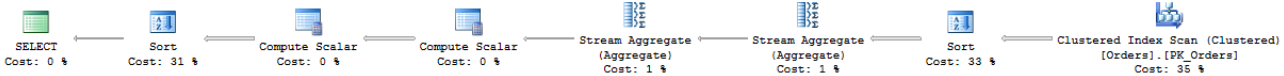
ROLLUP – plany zapytań

GFT

Query 1: Query cost (relative to the batch): 64%
SELECT s.* FROM (SELECT ShipCountry, ShipCity, COUNT(OrderID) as CNT FROM Orders WHERE DATEPART(yyyy, OrderDate) = 1997 GROUP BY ShipCountry, ShipCity UNION ALL SELECT ShipCountry, NULL , COUNT(…



Query 2: Query cost (relative to the batch): 36%
SELECT ShipCountry, ShipCity, COUNT(OrderID) as CNT FROM Orders WHERE DATEPART(yyyy, OrderDate) = 1997 GROUP BY ROLLUP (ShipCountry, ShipCity) ORDER BY CASE WHEN ShipCountry IS NULL THEN 1 ELSE 0…



ROLLUP – przykłady grupowań

GFT

- Przykłady (źródło: *https://technet.microsoft.com/pl-pl/library/bb522495(v=sql.105).aspx*):

Operacja	Grupowanie

`ROLLUP (DATEPART(yyyy,OrderDate)`
`,DATEPART(mm,OrderDate)`
`,DATEPART(dd,OrderDate))`
	`year, month, day`
	`year, month`
	`year`
	`()`

ROLLUP – przykłady grupowań

GFT

Operacja	Grupowanie

`ROLLUP (region, city),`
`ROLLUP (DATEPART(yyyy,OrderDate)`
`,DATEPART(mm,OrderDate)`
`,DATEPART(dd,OrderDate))`
	region, city, year, month, day
	region, city, year, month
	region, city, year
	region, city
	region, year, month, day
	region, year, month
	region, year
	region
	year, month, day
	year, month
	year
	()

CUBE

GFT

- Polecenie **CUBE** działa w podobny sposób do polecenia **ROLLUP**, z tą różnicą, iż przy tworzeniu poszczególnych grupowań uwzględniane są wszystkie kombinacje wskazanych kolumn.
- Zapytanie

```
SELECT
...
GROUP BY CUBE (a,b,c)
```

Wygeneruje grupowania:

```
(a, b, c)
(a, b)
(a, c)
(a)
(b, c)
(b)
(c)
()
```

Wykonane zostaną grupowania:

- `GROUP BY a,b,c`
- `GROUP BY a,b`
- `GROUP BY a,c`
- `GROUP BY a`
- `GROUP BY b,c`
- `GROUP BY b`
- `GROUP BY c`
- `GROUP BY ()`

2^N grupowań

CUBE – przykłady grupowań

GFT

Operacja	Grupowanie	Grupowanie

`CUBE (region, city`
`,DATEPART(yyyy,OrderDate)`
`,DATEPART(mm,OrderDate)`
`,DATEPART(dd,OrderDate))`
	region, city, year, month, day	city, year, month, day

	region, city, year, month	city, year, month
	region, city, year	city, year
	region, city	city, month, day
	region, city, month, day	city, month
	region, city, month	city, year, day
	region, city, day	city, day
	region, city, year, day	year, month, day
	region, city, day	year, month
	region, year, month, day	year
	region, year, month	year, day
	region, year	month, day
	region, month, day	month
	region, month	day
	region, year, day	()
	region, day	
	region	

CUBE – zadanie

GFT

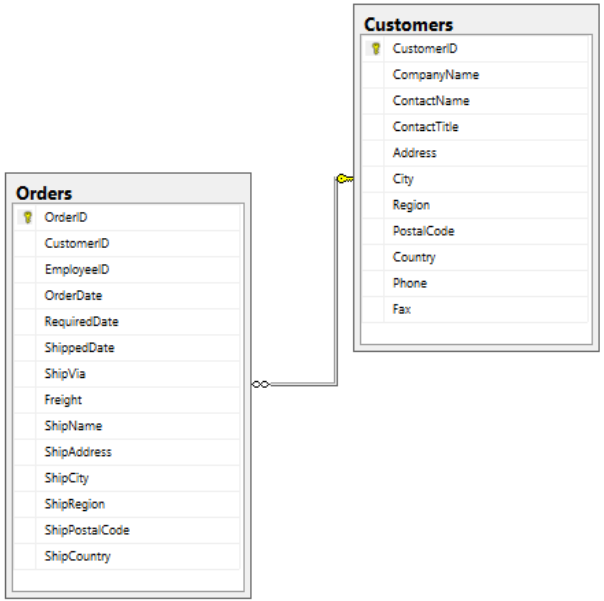
?

Zadanie

Korzystając z tabeli ***Orders*** oraz ***Customers*** przedstaw pełną analizę przedstawiającą liczbą zamówień w 3 wymiarach: rok (***Orders.OrderDate*** *[datetime]*), kraj oraz miasto zamieszkania klienta (***Customers.Country*** *[nvarchar]*, ***Customers.City*** *[nvarchar]*). Wyświetl dane dla liczby zamówień większej od 15.

```
SELECT  DATEPART(yyyy,O.OrderDate) as Year, C.Country,
        C.City, COUNT(*) as NumberOfOrders
FROM    Orders O JOIN Customers C
ON      (O.CustomerID = C.CustomerID)
GROUP BY CUBE(DATEPART(yyyy,O.OrderDate),
                C.Country,
                C.City)
HAVING COUNT(*) > 15
ORDER BY NumberOfOrders DESC;
```

	Year	Country	City	NumberOfOrders
1	NULL	NULL	NULL	830
2	1997	NULL	NULL	408
3	1998	NULL	NULL	270
4	1996	NULL	NULL	152
5	NULL	Germany	NULL	122
6	NULL	USA	NULL	122
7	NULL	Brazil	NULL	83
8	NULL	France	NULL	77
9	1997	Germany	NULL	64
10	1997	USA	NULL	60
11	NULL	UK	NULL	56
12	NULL	Venezuela	NULL	46
13	NULL	UK	London	46
14	NULL	NULL	London	46
15	1997	Brazil	NULL	42



ROLLUP & CUBE – kolumny złożone

GFT

```
SELECT
...
GROUP BY ROLLUP ((a, b), c)
```

```
(a, b, c)
(a, b)
()
```

Brak:

```
(a)
```

```
SELECT
...
GROUP BY CUBE ((a, b), c)
```

```
(a, b, c)
(a, b)
(c)
()
```

Brak:

```
(a, c)
(a)
(b, c)
(b)
```

ROLLUP & CUBE – redukcja grup

GFT

```
SELECT
...
GROUP BY a, ROLLUP (b, c, d)

(a, b, c, d)
(a, b, c)
(a, b)
(a)
```

```
SELECT
...
GROUP BY a, CUBE (b, c, d)

(a, b, c, d)
(a, b, c)
(a, b, d)
(a, b)
(a, c, d)
(a, c)
(a, d)
(a)
```

GROUPING SETS

GFT

- Polecenie **GROUPING SETS** pozwala określić konkretne poziomy grupowania:
  - Poszczególne grupy wymieniamy po przecinku.
  - Grupy złożone z kilku kolumn ujmujemy w nawiasy ()
- Za pomocą **GROUPING SETS** możemy opisać podzbiory tworzone przez CUBE i ROLLUP:

```
ROLLUP(ShipCountry, ShipCity) <=> GROUPING SETS((ShipCountry, ShipCity), (ShipCountry), ())
```

  - Jest to np. pomocne, gdy chcemy zrezygnować, z niektórych poziomów grupowania wymuszonych przez ROLLUP i/lub CUBE
- **GROUPING SETS** może być użyte razem z poleceniami ROLLUP i CUBE w klauzuli GROUP BY: np.:

```
GROUP BY ROLLUP          (DATEPART(yyyy, OrderDate),
                          DATEPART(qq, OrderDate)),
         GROUPING SETS(  (ShipCountry,   ShipCity),
                         (ShipCountry))
```

ROLLUP/CUBE/GROUPING SETS - potencjalnie problemy?

GFT

Zadanie:

Korzystając z tabeli ***Orders*** zaprojektuj zapytanie przedstawiające sumaryczną liczbą zamówień dla poszczególnych państw (***ShipCountry*** *[nvarchar]*), regionów (***ShipRegion*** *[nvarchar]*) oraz miast (***ShipCity*** *[nvarchar]*) wraz z pośrednimi oraz pełnym podsumowaniem (ROLLUP).

```
SELECT   ShipCountry, ShipRegion, ShipCity,
         COUNT(OrderID) as NumberOfOrders
FROM     Orders
GROUP BY ROLLUP (ShipCountry, ShipRegion, ShipCity)
```

	ShipCountry	ShipRegion	ShipCity	NumberOfOrders
90	UK	NULL	London	33
91	UK	NULL	NULL	33
92	UK	Essex	Colchester	13
93	UK	Essex	NULL	13
94	UK	Isle of Wi...	Cowes	10
95	UK	Isle of Wi...	NULL	10
96	UK	NULL	NULL	56

GROUPING

GFT

- **GROUPING** jest funkcją wskazującą czy dana kolumna/wyrażenie jest agregowane
- Składnia:

  GROUPING (kolumna/wyrażenie)

- Zwracane wartości [tinyint]:
  - 1 – kolumna/wyrażenie jest agregacją
  - 0 – w przeciwnym razie
- Funkcja **GROUPING** może być użyta jedynie w poleceniu **SELECT**, **HAVING** oraz **ORDER BY** (zakładając, że użyto polecenia grupującego **GROUP BY**)

GROUPING – przykład użycia

GFT

```
SELECT    ShipCountry,
          ShipRegion,
          ShipCity,
          COUNT(OrderID) AS NumberOfOrders,
          GROUPING(ShipCountry) AS
                    ShipCntryGrp,
          GROUPING(ShipRegion) AS
                    ShipRegGrp,
          GROUPING(ShipCity) AS
                    ShipCityGrp
FROM      Orders
GROUP BY  ROLLUP (ShipCountry,
                    ShipRegion,
                    ShipCity)
```

	ShipCountry	ShipRegion	ShipCity	NumberOfOrders	ShipCntryGrp	ShipRegGrp	ShipCityGrp
90	UK	NULL	London	33	0	0	0
91	UK	NULL	NULL	33	0	0	1
92	UK	Essex	Colchester	13	0	0	0
93	UK	Essex	NULL	13	0	0	1
94	UK	Isle of Wi...	Cowes	10	0	0	0
95	UK	Isle of Wi...	NULL	10	0	0	1
96	UK	NULL	NULL	56	0	1	1

GROUPING_ID

GFT

- **GROUPING_ID** jest funkcją wyliczającą poziom grupowania dla poszczególnych kolumn (wyrażeń) w postaci wektora bitowego. Zwracana wartość jest decymalną reprezentacją tego wektora.

- Składnia:

GROUPING_ID (kolumna/wyrażenie [,…n])

- Zwracany typ: **int**

- Funkcja **GROUPING_ID** może być użyta jedynie w poleceniu **SELECT**, **HAVING** oraz **ORDER BY** (zakładając, że użyto polecenia grupującego **GROUP BY**)

GROUPING_ID - przykłady

GFT

Podzbiory	Agregowane kolumny	GROUPING_ID (a, b, c)	Wynik funkcji GROUPING_ID()
(a, b, c)	-	0 0 0	0
(a, b)	c	0 0 1	1
(a, c)	b	0 1 0	2
(b, c)	a	1 0 0	4
(a)	b c	0 1 1	3
(b)	a c	1 0 1	5
(c)	a b	1 1 0	6
()	a b c	1 1 1	7

Przykład

GFT

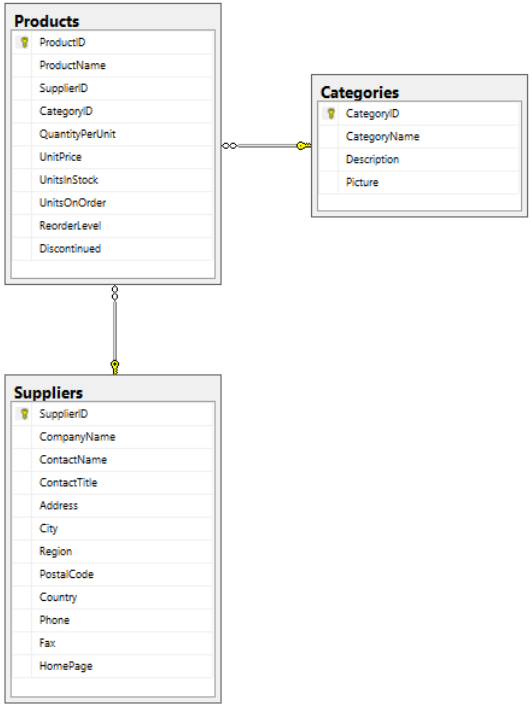
Zadanie

Korzystając z tabeli ***Products***, ***Categories***, ***Suppliers*** przeanalizuj średnią jednostkową, minimalną oraz maksymalną cenę produktu w 3 wymiarach: kategoria produktu (***Categories.CategoryName*** *[nvarchar]*), kraj (**Suppliers.City** *[nvarchar]*) wraz z regionem dostawcy (***Suppliers.Region*** *[nvarchar]*) oraz wyłącznie dla kraju.

W rozwiązaniu uwzględnij jedynie produkty, które posiadają wskazanie na dostawcę oraz przypisaną kategorie. Wartości puste w polu region, które wynikają z danych zamień na wartość – *„Not provided”*.

Do rozwiązania dodaj kolumnę, która dla poszczególnych wymiarów przyjmie następujące wartości:

- Kategoria produktu – *„Category”*
- Kraj wraz z regionem – *„Country & Region”*
- Kraj – *„Country”*



Przykład - rozwiązanie

GFT

```
SELECT          C.CategoryName                  AS Category,
                S.Country                       AS Country,
                CASE
                  WHEN          GROUPING(S.Region) = 0
                                AND S.Region IS NULL THEN 'Not Provided'
                  ELSE          S.Region
                END                             AS Region,
                ROUND(AVG(P.UnitPrice), 2)      AS AvgUnitPrice,
                MIN(UnitPrice)                  AS MinUnitPrice,
                MAX(UnitPrice)                  AS MaxUnitPrice,
                CASE GROUPING_ID (C.CategoryName, S.Country, S.Region)
                                WHEN 5 THEN 'Country'
                                WHEN 4 THEN 'Country & Region'
                                WHEN 3 THEN 'Category'
                 END                            AS GroupingLevel
FROM            Products P
JOIN            Categories C
ON              P.CategoryID = C.CategoryID
JOIN            Suppliers S
ON              P.SupplierID = S.SupplierID
GROUP BY GROUPING SETS ((C.CategoryName),
                        (S.Country, S.Region),
                        (S.Country))
```

	Category	Country	Region	AvgUnitPrice	MinUnitPrice	MaxUnitPrice	GroupingLevel
28	NULL	Sweden	Not Provided	18,00	9,00	26,00	Country & Region
29	NULL	Sweden	NULL	18,00	9,00	26,00	Country
30	NULL	UK	Not Provided	22,81	9,20	81,00	Country & Region
31	NULL	UK	NULL	22,81	9,20	81,00	Country
32	NULL	USA	LA	20,35	17,00	22,00	Country & Region
33	NULL	USA	MA	14,03	9,65	18,40	Country & Region
34	NULL	USA	MI	31,67	25,00	40,00	Country & Region
35	NULL	USA	OR	15,33	14,00	18,00	Country & Region
36	NULL	USA	NULL	20,87	9,65	40,00	Country
37	Bevera...	NULL	NULL	37,98	4,50	263,50	Category
38	Condim...	NULL	NULL	23,06	10,00	43,90	Category
39	Confec...	NULL	NULL	25,16	9,20	81,00	Category
40	Dairy P...	NULL	NULL	28,73	2,50	55,00	Category
41	Grains/...	NULL	NULL	20,25	7,00	38,00	Category
42	Meat/P...	NULL	NULL	54,01	7,45	123,79	Category

PIVOT

GFT

- Polecenie **PIVOT** pozwala przekształcić wyniki z jednego układu tabelarycznego w inny układ tabelaryczny
- Celem jest transformacja danych z układu wierszowego na kolumnowy, w celu czytelniejszej prezentacji danych
- Podczas przekształcenia wykonywana jest agregacja
- Wartości NULL nie są uwzględniane przy obliczaniu agregacji

PIVOT

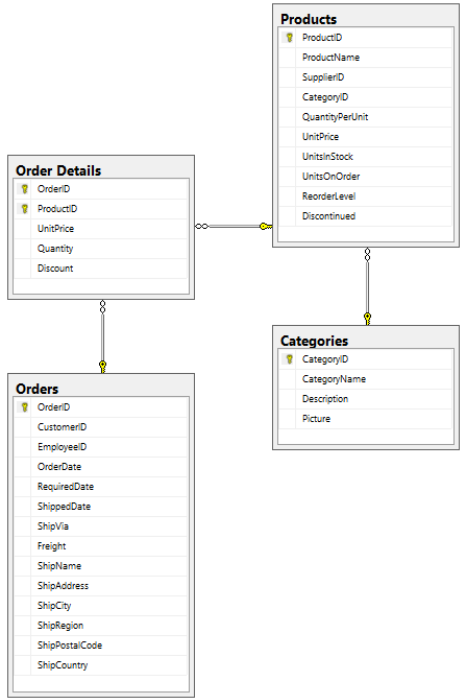
GFT

Zadanie

Korzystając z tabel ***Products***, ***Categories***, ***Order Details*** oraz ***Orders*** przedstaw sumę wartości zamówień (***Order Details.UnitPrice*** [money], ***Order Details.Quantity*** [smallint]) produktów w danej kategorii (***Categories.CategoryName*** [nvarchar]) w poszczególnych latach (***Orders.OrderDate*** [datetime]), uwzględniając dane, które posiadają wszystkie wymagane informacje.

```
SELECT    C.CategoryName AS CategoryName,
          DATEPART(yyyy,O.OrderDate) AS Year,
          SUM(OD.UnitPrice*OD.Quantity) AS Amount
FROM      [Products] P
JOIN      [Order Details] OD
ON        P.ProductID = OD.ProductID
JOIN      [Orders] O
ON        OD.OrderID = O.OrderID
JOIN      [Categories] C
ON        C.CategoryID = P.CategoryID
GROUP BY C.CategoryName, DATEPART(yyyy,OrderDate)
```

	CategoryName	Year	Amount
1	Produce	1996	15134,20
2	Confections	1998	58359,73
3	Grains/Cereals	1996	9817,60
4	Beverages	1997	110424,00
5	Seafood	1996	21589,60
6	Dairy Products	1996	44615,80
7	Condiments	1997	59679,00
8	Meat/Poultry	1998	60275,57
9	Produce	1997	57718,55
10	Grains/Cereals	1997	60486,95
11	Confections	1996	31511,60
12	Beverages	1998	122223,75
13	Meat/Poultry	1996	30292,20
14	Produce	1998	32415,85
15	Grains/Cereals	1998	30422,25



PIVOT

GFT

Zadanie*

Zaktualizuj poprzednie zapytanie, tak aby jako wiersze otrzymać kategorie, a lata jako kolumny. Na przecięciu tych wartości powinna być zaprezentowana kwota zamówień dla danej kategorii w danym roku.

	CategoryName	1996	1997	1998
1	Seafood	21589,60	71320,65	48712,84
2	Meat/Poultry	30292,20	87621,03	60275,57
3	Condiments	19458,30	59679,00	34557,45
4	Confections	31511,60	87227,77	58359,73
5	Produce	15134,20	57718,55	32415,85
6	Dairy Products	44615,80	123910,80	82803,90
7	Beverages	53879,20	110424,00	122223,75
8	Grains/Cereals	9817,60	60486,95	30422,25

PIVOT

GFT

```sql
SELECT [CategoryName], [1996], [1997], [1998]
FROM (
        SELECT   C.CategoryName    AS CategoryName,
        DATEPART(yyyy,OrderDate)   AS Year,
        (OD.UnitPrice*OD.Quantity) AS Amount
        FROM     [Products] P
        JOIN     [Order Details] OD
        ON       P.ProductID = OD.ProductID
        JOIN     [Orders] O
        ON       OD.OrderID = O.OrderID
        JOIN     [Categories] C
        ON       C.CategoryID = P.CategoryID) S
PIVOT
(   SUM(S.Amount)
    FOR Year IN ([1996], [1997], [1998])
) AS AMT
```

```
SELECT <non-pivoted column>,

    [first pivoted column] AS <column name>,

    [second pivoted column] AS <column name>,

    ...

    [last pivoted column] AS <column name>

FROM

    (<SELECT query that produces the data>)

    AS <alias for the source query>

PIVOT

(

    <aggregation function>(<column being aggregated>)

FOR

[<column that contains the values that will become column headers>]

    IN ( [first pivoted column], [second pivoted column],

... [last pivoted column])

) AS <alias for the pivot table>

<optional ORDER BY clause>;
```

PIVOT

GFT

```
SELECT [CategoryName], [1996], [1997], [1998]
FROM (
        SELECT   C.CategoryName    AS CategoryName,
        DATEPART(yyyy,OrderDate)   AS Year,
        (OD.UnitPrice*OD.Quantity) AS Amount
        FROM     [Products] P
        JOIN     [Order Details] OD
        ON       P.ProductID = OD.ProductID
        JOIN     [Orders] O
        ON       OD.OrderID = O.OrderID
        JOIN     [Categories] C
        ON       C.CategoryID = P.CategoryID) S
PIVOT
(   SUM(S.Amount)
    FOR Year IN ([1996], [1997], [1998])
) AS AMT
```

	CategoryName	1996	1997	1998
1	Seafood	21589,60	71320,65	48712,84
2	Meat/Poultry	30292,20	87621,03	60275,57
3	Condiments	19458,30	59679,00	34557,45
4	Confections	31511,60	87227,77	58359,73
5	Produce	15134,20	57718,55	32415,85
6	Dairy Products	44615,80	123910,80	82803,90
7	Beverages	53879,20	110424,00	122223,75
8	Grains/Cereals	9817,60	60486,95	30422,25

UNPIVOT

GFT

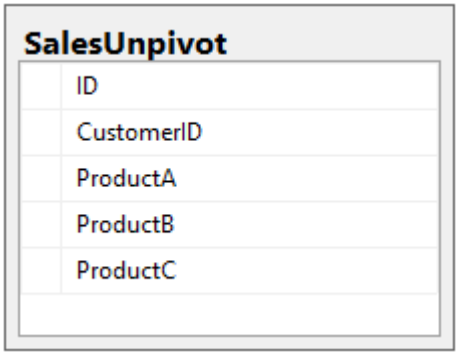
- Polecenie **UNPIVOT** jest operacją odwrotną do operacji **PIVOT** w sensie układu (przekształca z wierszowego układu tabelarycznego na kolumnowy), ale korzystając z **UNPIVOT** nie można odtworzyć oryginalnej tabeli, która została przekształcona za pomocą polecenia **PIVOT**

- Wszystkie kolumny na liście **UNPIVOT** muszą być dokładnie tego samego typu oraz tej samej długości

UNPIVOT

GFT

- Mamy tabelę:



	ID	CustomerID	ProductA	ProductB	ProductC
1	1	100	32	44	55
2	2	101	22	13	0
3	2	102	123	2	12
4	2	103	66	72	14
5	2	104	22	32	63

- Chcemy przekształcić i zaprezentować wynik w postaci:

	ID	CustomerID	ProductCode	Quantity
1	1	100	ProductA	32
2	1	100	ProductB	44
3	1	100	ProductC	55
4	2	101	ProductA	22
5	2	101	ProductB	13
6	2	101	ProductC	0
7	2	102	ProductA	123
8	2	102	ProductB	2
9	2	102	ProductC	12
10	2	103	ProductA	66
11	2	103	ProductB	72
12	2	103	ProductC	14
13	2	104	ProductA	22
14	2	104	ProductB	32
15	2	104	ProductC	63

UNPIVOT

GFT

```
SELECT ID, CustomerID, ProductCode, Quantity
FROM
(
  SELECT ID, CustomerID, ProductA, ProductB,
         ProductC
  FROM   SalesUnpivot) p
UNPIVOT
  (Quantity FOR ProductCode IN (ProductA,
        ProductB, ProductC )) AS unpvt
```

	ID	CustomerID	ProductA	ProductB	ProductC
1	1	100	32	44	55
2	2	101	22	13	0
3	2	102	123	2	12
4	2	103	66	72	14
5	2	104	22	32	63

	ID	CustomerID	ProductCode	Quantity
1	1	100	ProductA	32
2	1	100	ProductB	44
3	1	100	ProductC	55
4	2	101	ProductA	22
5	2	101	ProductB	13
6	2	101	ProductC	0
7	2	102	ProductA	123
8	2	102	ProductB	2
9	2	102	ProductC	12
10	2	103	ProductA	66
11	2	103	ProductB	72
12	2	103	ProductC	14
13	2	104	ProductA	22
14	2	104	ProductB	32
15	2	104	ProductC	63

Podsumowanie

GFT

- **NULL** w funkcjach agregujących!
- Polecenia **ROLLUP** and **CUBE** umożliwiają tworzenie podsumowań na różnych poziomach agregacji
- Polecenie **GROUPING SETS** umożliwia określenie własnych zbiorów grupowań
- Polecenia **PIVOT/UNPIVOT** pozywają transformować wynik z jednego układu tabelarycznego w inny (np. z rekordowego na kolumnowy i odwrotnie)
- **GROUPING** i **GROUPING_ID** – pomocne funkcje do określenia poziomu grupowania

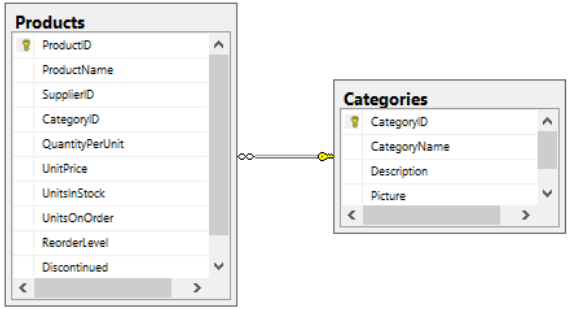
GFT

Common Table Expression

CTE - przykład

GFT

```
WITH
ProdAvgUnitPrice (AvgUnitPrice)
AS
(
          SELECT AVG(UnitPrice)
          FROM   Products
),
GreaterThanAvg (ProductName, CategoryID, UnitPrice)
AS
(
          SELECT ProductName, CategoryID, UnitPrice
          FROM   Products P
          WHERE  UnitPrice > (SELECT AvgUnitPrice FROM ProdAvgUnitPrice)
)
SELECT     G.ProductName, C.CategoryName, G.UnitPrice
FROM       GreaterThanAvg G JOIN Categories C
ON         (C.CategoryID = G.CategoryID);
```



	ProductName	CategoryName	UnitPrice
1	Uncle Bob's Organic Dried Pears	Produce	30,00
2	Northwoods Cranberry Sauce	Condiments	40,00
3	Mishi Kobe Niku	Meat/Poultry	97,00
4	Ikura	Seafood	31,00
5	Queso Manchego La Pastora	Dairy Products	38,00
6	Alice Mutton	Meat/Poultry	39,00
7	Carnarvon Tigers	Seafood	62,50
8	Sir Rodney's Marmalade	Confections	81,00
9	Gumbär Gummibärchen	Confections	31,23
10	Schoggi Schokolade	Confections	43,90
11	Rössle Sauerkraut	Produce	45,60
12	Thüringer Rostbratwurst	Meat/Poultry	123,79
13	Mascarpone Fabioli	Dairy Products	32,00
14	Côte de Blaye	Beverages	263,50
15	Ipoh Coffee	Beverages	46,00

Rekurencja – definicja i przykłady

GFT

- **Rekurencja**, zwana także rekursją jest odwołaniem się np. funkcji lub definicji do samej siebie (Wikipedia)

- Obliczanie silni jako przykład rekurencji:

$$n! = \begin{cases} 1 & \text{dla } n = 0 \\ n \cdot (n-1)! & \text{dla } n \geqslant 1 \end{cases}$$

- Przykłady:

  - $4! = 4 * 3! = 4 * 3 * 2! = 4 * 3 * 2 * 1! = 4 * 3 * 2 * 1 = 24$
  - $5! = 5 * 4! = 5 * 4 * 3! = \ldots = 5 * 4! = 5 * 24 = 120$

Rekurencyjne CTE

GFT

- Rekurencyjna definicja **CTE** musi zawierać co najmniej dwa zapytania: zapytanie zakotwiczające (*anchor member*) oraz zapytanie rekurencyjne (*recursive member*).
- Można użyć wielu zapytań zakotwiczający oraz rekurencyjnych, natomiast wszystkie zapytania zakotwiczający muszą być zdefiniowane przed zapytaniami rekurencyjnymi.
- Wszystkie zapytania zakotwiczające muszą być połączone za pomocą operatorów UNION ALL, UNION, INTERSECT lub EXCEPT.
- Pomiędzy zapytaniem zakotwiczający a rekurencyjnym może być użyty jedynie UNION ALL
- Warunek stopu występuje, gdy zapytanie rekurencyjne nie zwróci żadnego rekordu

Rekurencyjne CTE

GFT

- Liczba kolumn oraz typ zapytania zakotwiczającego oraz rekurencyjnego musi być taka sama

- Następujące elementy nie są dozwolone w rekurencyjnym zapytaniu:
  - SELECT DISTINCT
  - GROUP BY
  - PIVOT (w zależności od poziomu kompatybilności bazy danych)
  - HAVING
  - Skalarna agregacja
  - TOP
  - OUTER JOIN
  - Podzapytania
  - Podpowiedź (hint) stosowana do zapytania rekurencyjnego wewnątrz definicji CTE

Rekurencyjne CTE - silnia

GFT

```
WITH Factorial (N, FactorialValue) AS
(
        SELECT 1, 1
        UNION ALL
        SELECT N+1, (N+1) * FactorialValue
        FROM  Factorial
        WHERE N < 10
)
SELECT N, FactorialValue
FROM Factorial
```

Zapytanie zakotwiczające

Tu się dzieje rekurencja

Odwołanie do CTE

Warunek stopu

Zapytanie rekurencyjne

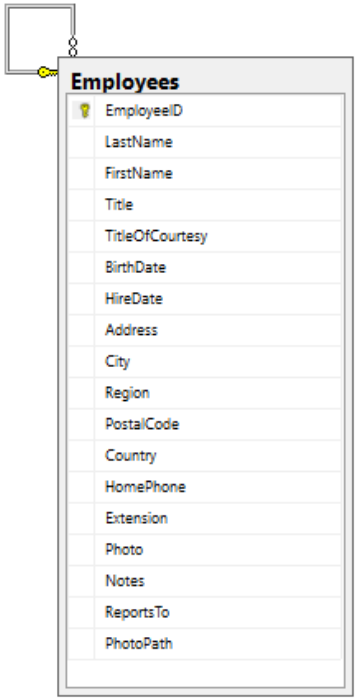
	N	FactorialValue
1	1	1
2	2	2
3	3	6
4	4	24
5	5	120
6	6	720
7	7	5040
8	8	40320
9	9	362880
10	10	3628800

Zapytania hierarchiczne

GFT

Zadanie

Korzystając z tabeli ***Employees*** wyświetl identyfikator (***EmployeeID*** *[int]*), imię (***FirstName*** *[nvarchar])* oraz nazwisko (***LastName*** *[nvarchar]*) pracownika oraz identyfikator, imię i nazwisko jego przełożonego. Do znalezienia przełożonego danego pracowania użyj pola ***ReportsTo*** (*FK, [int]*).



	EmployeeID	FirstName	LastName	ReportsTo	ManagerFirstName	ManagerLastName
1	2	Andrew	Fuller	NULL	NULL	NULL
2	1	Nancy	Davolio	2	Andrew	Fuller
3	3	Janet	Leverling	2	Andrew	Fuller
4	4	Margaret	Peacock	2	Andrew	Fuller
5	5	Steven	Buchanan	2	Andrew	Fuller
6	8	Laura	Callahan	2	Andrew	Fuller
7	6	Michael	Suyama	5	Steven	Buchanan
8	7	Robert	King	5	Steven	Buchanan
9	9	Anne	Dodsworth	5	Steven	Buchanan

CTE

GFT

```
WITH EmployeesRecCTE
(
  EmployeeID, FirstName, LastName, ReportsTo, ManagerFirstName, ManagerLastName
)
AS
(
  SELECT      EmployeeID, FirstName, LastName, ReportsTo,
              CAST(NULL AS NVARCHAR(10)) AS ManagerFirstName,
              CAST(NULL AS NVARCHAR(20))  AS ManagerLastName
  FROM        Employees
  WHERE       ReportsTo IS NULL
  UNION ALL
  SELECT      E.EmployeeID, E.FirstName, E.LastName, R.EmployeeID, R.FirstName,
              R.LastName
  FROM Employees E JOIN EmployeesRecCTE R ON E.ReportsTo = R.EmployeeID
)
SELECT        EmployeeID, FirstName, LastName, ReportsTo, ManagerFirstName,
              ManagerLastName
FROM          EmployeesRecCTE;
```

	EmployeeID	FirstName	LastName	ReportsTo	ManagerFirstName	ManagerLastName
1	2	Andrew	Fuller	NULL	NULL	NULL
2	1	Nancy	Davolio	2	Andrew	Fuller
3	3	Janet	Leverling	2	Andrew	Fuller
4	4	Margaret	Peacock	2	Andrew	Fuller
5	5	Steven	Buchanan	2	Andrew	Fuller
6	8	Laura	Callahan	2	Andrew	Fuller
7	6	Michael	Suyama	5	Steven	Buchanan
8	7	Robert	King	5	Steven	Buchanan
9	9	Anne	Dodsworth	5	Steven	Buchanan

CTE – Poziom rekurencji

GFT

```
WITH EmployeesRecCTE
(
 EmployeeID, FirstName, LastName, ReportsTo, ManagerFirstName, ManagerLastName,
Level
)
AS
(
  SELECT       EmployeeID, FirstName, LastName, ReportsTo,
               CAST(NULL AS NVARCHAR(10)) AS ManagerFirstName,
               CAST(NULL AS NVARCHAR(20))  AS ManagerLastName,
               0 AS Level
  FROM         Employees
  WHERE        ReportsTo IS NULL
  UNION ALL
  SELECT       E.EmployeeID, E.FirstName, E.LastName, R.EmployeeID, R.FirstName,
               R.LastName, Level + 1
  FROM Employees E JOIN EmployeesRecCTE R ON E.ReportsTo = R.EmployeeID
)
SELECT         EmployeeID, FirstName, LastName, ReportsTo, ManagerFirstName,
               ManagerLastName, Level
FROM           EmployeesRecCTE;
```

	EmployeeID	FirstName	LastName	ReportsTo	ManagerFirstName	ManagerLastName	Level
1	2	Andrew	Fuller	NULL	NULL	NULL	0
2	1	Nancy	Davolio	2	Andrew	Fuller	1
3	3	Janet	Leverling	2	Andrew	Fuller	1
4	4	Margaret	Peacock	2	Andrew	Fuller	1
5	5	Steven	Buchanan	2	Andrew	Fuller	1
6	8	Laura	Callahan	2	Andrew	Fuller	1
7	6	Michael	Suyama	5	Steven	Buchanan	2
8	7	Robert	King	5	Steven	Buchanan	2
9	9	Anne	Dodsworth	5	Steven	Buchanan	2

CTE – MAXRECURSION – ograniczenie poziomu rekurencji

GFT

```
WITH EmployeesRecCTE
(
 EmployeeID, FirstName, LastName, ReportsTo, ManagerFirstName, ManagerLastName, Level
)
AS
(
  SELECT        EmployeeID, FirstName, LastName, ReportsTo,
                CAST(NULL AS NVARCHAR(10)) AS ManagerFirstName,
                CAST(NULL AS NVARCHAR(20))  AS ManagerLastName,
                0 AS Level
  FROM          Employees
  WHERE         ReportsTo IS NULL
  UNION ALL
  SELECT        E.EmployeeID, E.FirstName, E.LastName, R.EmployeeID, R.FirstName,
                R.LastName, Level + 1
  FROM Employees E JOIN EmployeesRecCTE R ON E.ReportsTo = R.EmployeeID
)
SELECT          EmployeeID, FirstName, LastName, ReportsTo, ManagerFirstName,
                ManagerLastName, Level
FROM            EmployeesRecCTE
OPTION          (MAXRECURSION 1);
```

	EmployeeID	FirstName	LastName	ReportsTo	ManagerFirstName	ManagerLastName	Level
1	2	Andrew	Fuller	NULL	NULL	NULL	0
2	1	Nancy	Davolio	2	Andrew	Fuller	1
3	3	Janet	Leverling	2	Andrew	Fuller	1
4	4	Margaret	Peacock	2	Andrew	Fuller	1
5	5	Steven	Buchanan	2	Andrew	Fuller	1
6	8	Laura	Callahan	2	Andrew	Fuller	1

```
Msg 530, Level 16, State 1, Line 119
The statement terminated. The maximum recursion 1 has been exhausted before statement completion.
```

CTE - Podsumowanie

GFT

- Poprawia czytelność i przejrzystość kodu
- Umożliwia tworzenia zapytań rekurencyjnych
- Umożliwia tworzenie zapytań hierarchicznych (parent – child)
- …

GFT

Funkcje analityczne

Funkcje analityczne

GFT

- Funkcje analityczne **umożliwiają realizowanie obliczeń na zbiorach wierszy w elastyczny, czytelny i efektywny sposób.** (Itzik Ben-Gan, „Microsoft SQL Server 2012 Optymalizacja kwerend T-SQL przy użyciu funkcji okna)

- Funkcje analityczne vs. funkcje okna

- Podział funkcji analitycznych:
  - Funkcje agregujące
  - Funkcje rankingowe
  - Funkcje rozkładu
  - Funkcje przesunięcia

Agregacja na całym zbiorze vs. OVER()

GFT

```
SELECT    SUM(UnitPrice) AS SUM
FROM      Products
```

	SUM
1	2222,71

```
SELECT    P.ProductName,
          C.CategoryName,
          SUM(P.UnitPrice) OVER () AS SUM
FROM      Products P JOIN Categories C
ON        P.CategoryID = C.CategoryID
```

	ProductName	CategoryName	SUM
1	Chai	Beverages	2222,71
2	Chang	Beverages	2222,71
3	Aniseed Syrup	Condiments	2222,71
4	Chef Anton's Cajun Seasoning	Condiments	2222,71
5	Chef Anton's Gumbo Mix	Condiments	2222,71
6	Grandma's Boysenberry Spread	Condiments	2222,71
7	Uncle Bob's Organic Dried Pears	Produce	2222,71
8	Northwoods Cranberry Sauce	Condiments	2222,71
9	Mishi Kobe Niku	Meat/Poultry	2222,71
10	Ikura	Seafood	2222,71
11	Queso Cabrales	Dairy Products	2222,71
12	Queso Manchego La Pastora	Dairy Products	2222,71
13	Konbu	Seafood	2222,71
14	Tofu	Produce	2222,71
15	Genen Shouyu	Condiments	2222,71

...

	ProductName	CategoryName	SUM
63	Vegie-spread	Condiments	2222,71
64	Wimmers gute Semmelknödel	Grains/Cereals	2222,71
65	Louisiana Fiery Hot Pepper Sa...	Condiments	2222,71
66	Louisiana Hot Spiced Okra	Condiments	2222,71
67	Laughing Lumberjack Lager	Beverages	2222,71
68	Scottish Longbreads	Confections	2222,71
69	Gudbrandsdalsost	Dairy Products	2222,71
70	Outback Lager	Beverages	2222,71
71	Flotemysost	Dairy Products	2222,71
72	Mozzarella di Giovanni	Dairy Products	2222,71
73	Röd Kaviar	Seafood	2222,71
74	Longlife Tofu	Produce	2222,71
75	Rhönbräu Klosterbier	Beverages	2222,71
76	Lakkalikööri	Beverages	2222,71
77	Original Frankfurter grüne Soße	Condiments	2222,71

GROUP BY vs OVER (PARTITION BY…)

GFT

```
SELECT    C.CategoryName,
          SUM(P.UnitPrice) AS SUM
FROM      Products P JOIN Categories C
ON        P.CategoryID = C.CategoryID
GROUP BY  C.CategoryName
```

	CategoryName	SUM
1	Beverages	455,75
2	Condiments	276,75
3	Confections	327,08
4	Dairy Products	287,30
5	Grains/Cereals	141,75
6	Meat/Poultry	324,04
7	Produce	161,85
8	Seafood	248,19

```
SELECT    P.ProductName,
          C.CategoryName,
          SUM(P.UnitPrice) OVER (PARTITION BY C.CategoryName)
                    AS SUM
FROM      Products P JOIN Categories C
ON        P.CategoryID = C.CategoryID
```

	ProductName	CategoryName	SUM
1	Chai	Beverages	455,75
2	Chang	Beverages	455,75
3	Guaraná Fantástica	Beverages	455,75
4	Sasquatch Ale	Beverages	455,75
5	Steeleye Stout	Beverages	455,75
6	Côte de Blaye	Beverages	455,75
7	Chartreuse verte	Beverages	455,75
8	Ipoh Coffee	Beverages	455,75
9	Laughing Lumberjack Lager	Beverages	455,75
10	Outback Lager	Beverages	455,75
11	Rhönbräu Klosterbier	Beverages	455,75
12	Lakkalikööri	Beverages	455,75
13	Aniseed Syrup	Condiments	276,75
14	Chef Anton's Cajun Seasoning	Condiments	276,75
15	Chef Anton's Gumbo Mix	Condiments	276,75

...

63	Rössle Sauerkraut	Produce	161,85
64	Manjimup Dried Apples	Produce	161,85
65	Longlife Tofu	Produce	161,85
66	Ikura	Seafood	248,19
67	Konbu	Seafood	248,19
68	Carnarvon Tigers	Seafood	248,19
69	Nord-Ost Matjeshering	Seafood	248,19
70	Inlagd Sill	Seafood	248,19
71	Gravad lax	Seafood	248,19
72	Boston Crab Meat	Seafood	248,19
73	Jack's New England Clam Ch...	Seafood	248,19
74	Rogede sild	Seafood	248,19
75	Spegesild	Seafood	248,19
76	Escargots de Bourgogne	Seafood	248,19
77	Röd Kaviar	Seafood	248,19

OVER () – składnia

GFT

```
nazwa_funkcji (<argumenty>) OVER (
                              [ <PARTITION BY ...> ]
                              [ <ORDER BY ...> ]
                              [ <ROW or RANGE ...> ]
                             )
```

Klauzula partycji

Klauzula porządku w ramach partycji lub całego zbioru

Klauzula okna (ramki)

OVER () – definicja pojęć

GFT

- **Partycja** – jest to podzbiór danych wyznaczony poprzez unikalne wartości podanych argumentów, na którym jest wykonywana funkcja
- **Klauzula porządku** – określa kolejność wykonywania operacji na podzbiorze danych wyspecyfikowanych poprzez partycję (o ile została zdefiniowana) lub na pełnym zbiorze
- **Okno (ramki)** – występuje tylko w przypadku funkcji okna i pozwala zdefiniować ruchomy zakres wierszy w partycji, w ramach których funkcja będzie wyznaczała wartość
- **Bieżący wiersz** – wiersz, dla którego w danym momencie wyznaczany jest wynik funkcji analitycznej

OVER (PARTITION BY …)

GFT

```
SELECT    ProductID, CategoryID, UnitPrice,
          MAX(UnitPrice) OVER () AS MaxAll,
          MAX(UnitPrice) OVER (PARTITION BY CategoryID)
                    AS MaxByCategory
FROM      Products
```

Funkcja MAX wykonana na całym zbiorze

Funkcja MAX wykonana na partycjach wyznaczonych poprzez identyfikator kategorii CategoryID

	ProductID	CategoryID	UnitPrice	MaxAll	MaxByCategory
1	1	1	18,00	263,50	263,50
2	2	1	19,00	263,50	263,50
3	24	1	4,50	263,50	263,50
4	34	1	14,00	263,50	263,50
5	35	1	18,00	263,50	263,50
6	38	1	263,50	263,50	263,50
7	39	1	18,00	263,50	263,50
8	43	1	46,00	263,50	263,50
9	70	1	15,00	263,50	263,50
10	67	1	14,00	263,50	263,50
11	75	1	7,75	263,50	263,50
12	76	1	18,00	263,50	263,50
13	77	2	13,00	263,50	43,90
14	61	2	28,50	263,50	43,90
15	63	2	43,90	263,50	43,90
16	65	2	21,05	263,50	43,90
17	66	2	17,00	263,50	43,90
18	44	2	19,45	263,50	43,90
19	3	2	10,00	263,50	43,90
20	4	2	22,00	263,50	43,90
21	5	2	21,35	263,50	43,90
22	6	2	25,00	263,50	43,90
23	8	2	40,00	263,50	43,90
24	15	2	15,50	263,50	43,90

OVER (PARTITION BY … ORDER BY … + WINDOW)

GFT

```
SELECT    ProductID, CategoryID, UnitPrice, UnitsOnOrder,
          MAX(UnitPrice) OVER () AS MaxAll,
          MAX(UnitPrice) OVER (PARTITION BY CategoryID)
           AS MaxByCategory,
          MAX(UnitPrice) OVER (PARTITION BY CategoryID
                    ORDER BY UnitsOnOrder, ProductID DESC
                    ROWS BETWEEN 3 PRECEDING
                    AND 3 FOLLOWING)
                              AS MaxByCategoryWithWindow
FROM      Products
```

Okno/ramka – bieżący rekord + 3 poprzedzające oraz 3 następujące po bieżącym

Bieżący rekord

	ProductID	CategoryID	UnitPrice	UnitsOnOrder	MaxAll	MaxByCategory	MaxByCategoryWithWindow
1	76	1	18,00	0	263,50	263,50	18,00
2	75	1	7,75	0	263,50	263,50	263,50
3	67	1	14,00	0	263,50	263,50	263,50
4	39	1	18,00	0	263,50	263,50	263,50
5	38	1	263,50	0	263,50	263,50	263,50
6	35	1	18,00	0	263,50	263,50	263,50
7	34	1	14,00	0	263,50	263,50	263,50
8	24	1	4,50	0	263,50	263,50	263,50
9	1	1	18,00	0	263,50	263,50	46,00
10	70	1	15,00	10	263,50	263,50	46,00
11	43	1	46,00	10	263,50	263,50	46,00
12	2	1	19,00	40	263,50	263,50	46,00
13	77	2	13,00	0	263,50	43,90	43,90
14	65	2	21,05	0	263,50	43,90	43,90
15	63	2	43,90	0	263,50	43,90	43,90
16	61	2	28,50	0	263,50	43,90	43,90
17	44	2	19,45	0	263,50	43,90	43,90
18	15	2	15,50	0	263,50	43,90	43,90
19	8	2	40,00	0	263,50	43,90	40,00
20	6	2	25,00	0	263,50	43,90	40,00
21	5	2	21,35	0	263,50	43,90	40,00
22	4	2	22,00	0	263,50	43,90	40,00
23	3	2	10,00	70	263,50	43,90	25,00
24	66	2	17,00	100	263,50	43,90	22,00

OVER (PARTITION BY … ORDER BY … + WINDOW)

GFT

```
SELECT    ProductID, CategoryID, UnitPrice, UnitsOnOrder,
          MAX(UnitPrice) OVER () AS MaxAll,
          MAX(UnitPrice) OVER (PARTITION BY CategoryID)
           AS MaxByCategory,
          MAX(UnitPrice) OVER (PARTITION BY CategoryID
                    ORDER BY UnitsOnOrder, ProductID DESC
                    ROWS BETWEEN 3 PRECEDING
                    AND 3 FOLLOWING)
                              AS MaxByCategoryWithWindow
FROM      Products
```

Okno/ramka – bieżący rekord + 3 poprzedzające oraz 3 następujące po bieżącym

Bieżący rekord

	ProductID	CategoryID	UnitPrice	UnitsOnOrder	MaxAll	MaxByCategory	MaxByCategoryWithWindow
1	76	1	18,00	0	263,50	263,50	18,00
2	75	1	7,75	0	263,50	263,50	263,50
3	67	1	14,00	0	263,50	263,50	263,50
4	39	1	18,00	0	263,50	263,50	263,50
5	38	1	263,50	0	263,50	263,50	263,50
6	35	1	18,00	0	263,50	263,50	263,50
7	34	1	14,00	0	263,50	263,50	263,50
8	24	1	4,50	0	263,50	263,50	263,50
9	1	1	18,00	0	263,50	263,50	46,00
10	70	1	15,00	10	263,50	263,50	46,00
11	43	1	46,00	10	263,50	263,50	46,00
12	2	1	19,00	40	263,50	263,50	46,00
13	77	2	13,00	0	263,50	43,90	43,90
14	65	2	21,05	0	263,50	43,90	43,90
15	63	2	43,90	0	263,50	43,90	43,90
16	61	2	28,50	0	263,50	43,90	43,90
17	44	2	19,45	0	263,50	43,90	43,90
18	15	2	15,50	0	263,50	43,90	43,90
19	8	2	40,00	0	263,50	43,90	40,00
20	6	2	25,00	0	263,50	43,90	40,00
21	5	2	21,35	0	263,50	43,90	40,00
22	4	2	22,00	0	263,50	43,90	40,00
23	3	2	10,00	70	263,50	43,90	25,00
24	66	2	17,00	100	263,50	43,90	22,00

Partycje - PARTITION BY

GFT

- Do podziału na partycje używa się słów kluczowych **PARTITION BY**
- Podział może być realizowany za pomocą jednego lub wielu wyrażeń
- Przykłady

```
PARTITION BY CategoryID

PARTITION BY CategoryID, SupplierID

PARTITION BY DATEPART(yyyy, OrderDate)
```

```
OVER (
       [ <PARTITION BY clause> ]
       [ <ORDER BY clause> ]
       [ <ROW or RANGE clause> ]
      )

<PARTITION BY clause> ::=
PARTITION BY value_expression , ... [ n ]
```

Sortowanie – ORDER BY

GFT

- Do ustalenia porządku wykorzystuje się polecenie **ORDER BY**
- W niektórych funkcjach analitycznych ustalenie porządku w ramach zbioru jest obligatoryjne
- Sortowanie może być wykonane w oparciu o jedno lub wiele wyrażeń
- Przykłady:

```
ORDER BY ProductID ASC
ORDER BY ProductID ASC, UnitsOnOrder DESC
ORDER BY DATEPART(yyyy, OrderDate), ShipCity DESC
ORDER BY (SELECT NULL)
```

```
OVER (
       [ <PARTITION BY clause> ]
       [ <ORDER BY clause> ]
       [ <ROW or RANGE clause> ]
      )

<PARTITION BY clause> ::=
PARTITION BY value_expression , ... [ n ]

<ORDER BY clause> ::=
ORDER BY order_by_expression
    [ COLLATE collation_name ]
    [ ASC | DESC ]
    [ ,...n ]
```

Definiowanie okna - składnia

GFT

- Okno (ramka) pozwala na wyznaczenie wartości funkcji ruchomych, przyrostowych.

- Składnia:

```
ROWS | RANGE BETWEEN  <początek przedziału okna> AND <koniec przedziału okna>
```

- `UNBOUNDED PRECEDING` – wszystkie rekordy od początku partycji (poprzedzające bieżący wiersz)
- `<n> PRECEDING` – **n** rekordów poprzedzający bieżący wiersz
- `CURRENT ROW` – bieżący wiersz
- `<n> FOLLOWING` – **n** rekordów następujący po bieżącym wierszu
- `UNBOUNDED FOLLOWING` – wszystkie rekordy następujące po danym wierszu (do końca partycji)

Definiowanie okna – ROWS vs. RANGE

GFT

- **ROWS** – określa rozmiar okna (ramki) z dokładnością do jednego wiersza

- **RANGE** – oznacza zakres bazując na wartości danego wiersza. Dla RANGE **CURRENT ROW** oznacza bieżący rekord oraz wszystkie poprzedzające/następujące rekordy, które mają taką samą wartość jak bieżący. W **RANGE** można wykorzystać kombinacje:

```
UNBOUNDED PRECEDING AND CURRENT ROW
CURRENT ROW
CURRENT ROW AND UNBOUNDED FOLLOWING
```

- Dla funkcji okna (z ramką), gdy podamy **ORDER BY** a nie wyspecyfikujemy okna to domyślna wartość okna/ramki przyjmuje postać:

```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```

Definiowanie okna - przykłady

GFT

- Okno obejmujące 7 wierszy: bieżący, 4 poprzedzający i 2 następujące po bieżącym:

```
ROWS BETWEEN 4 PRECEDING AND 2 FOLLOWING
```

- Okno obejmujące bieżący wiersz i 3 następne:

```
ROWS BETWEEN CURRENT ROW AND 3 FOLLOWING
```

- Okno obejmujące wszystkie poprzednie do bieżącego wiersza włącznie:

```
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```

```
ROWS UNBOUNDED PRECEDING
```

- Okno obejmujące wszystkie poprzedzające rekordy oraz wszystkie wiersze z taką samą wartością jak wiersz bieżący:

```
RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
```

```
RANGE UNBOUNDED PRECEDING
```

Funkcje agregujące

GFT

- Funkcje agregujące mogę być wykorzystywane zarówno z klauzulą **OVER** jak i w połączeniu z **GROUP BY**
- Funkcje agregacji służą do wykonywania operacji na zbiorze wierszy
- Funkcje agregujące:
  - SUM
  - COUNT
  - COUNT_BIG
  - AVG
  - MIN
  - MAX
  - STDEV
  - STDEVP
  - VAR
  - VARP
  - CHECKSUM_AGG

Funkcje agregujące – AVG & SUM

GFT

```
SELECT    OrderID,
          ProductID,
          UnitPrice,
          Quantity,
          UnitPrice*Quantity AS Value,
          SUM(UnitPrice*Quantity) OVER (
                    PARTITION BY OrderID
                    ORDER BY ProductID
                    ROWS BETWEEN UNBOUNDED PRECEDING AND
                              CURRENT ROW) AS RunSum,
          AVG(UnitPrice*Quantity) OVER (
                    PARTITION BY OrderID
                    ORDER BY ProductID
                    ROWS BETWEEN 1 PRECEDING AND
                              CURRENT ROW) AS MovAvg
FROM      [Order Details]
```

	OrderID	ProductID	UnitPrice	Quantity	Value	RunSum	MovAvg
1	10248	11	14,00	12	168,00	168,00	168,00
2	10248	42	9,80	10	98,00	266,00	133,00
3	10248	72	34,80	5	174,00	440,00	136,00
4	10249	14	18,60	9	167,40	167,40	167,40
5	10249	51	42,40	40	1696,00	1863,40	931,70
6	10250	41	7,70	10	77,00	77,00	77,00
7	10250	51	42,40	35	1484,00	1561,00	780,50
8	10250	65	16,80	15	252,00	1813,00	868,00
9	10251	22	16,80	6	100,80	100,80	100,80
10	10251	57	15,60	15	234,00	334,80	167,40
11	10251	65	16,80	20	336,00	670,80	285,00
12	10252	20	64,80	40	2592,00	2592,00	2592,00
13	10252	33	2,00	25	50,00	2642,00	1321,00
14	10252	60	27,20	40	1088,00	3730,00	569,00
15	10253	31	10,00	20	200,00	200,00	200,00
16	10253	39	14,40	42	604,80	804,80	402,40
17	10253	49	16,00	40	640,00	1444,80	622,40
18	10254	24	3,60	15	54,00	54,00	54,00
19	10254	55	19,20	21	403,20	457,20	228,60
20	10254	74	8,00	21	168,00	625,20	285,60
21	10255	2	15,20	20	304,00	304,00	304,00
22	10255	16	13,90	35	486,50	790,50	395,25
23	10255	36	15,20	25	380,00	1170,50	433,25
24	10255	59	44,00	30	1320,00	2490,50	850,00

Funkcje agregujące – AVG & SUM

GFT

```
SELECT    OrderID,
          ProductID,
          UnitPrice,
          Quantity,
          UnitPrice*Quantity AS Value,
          SUM(UnitPrice*Quantity) OVER (
                    PARTITION BY OrderID
                    ORDER BY ProductID
                    ROWS BETWEEN UNBOUNDED PRECEDING AND
                              CURRENT ROW) AS RunSum,
          AVG(UnitPrice*Quantity) OVER (
                    PARTITION BY OrderID
                    ORDER BY ProductID
                    ROWS BETWEEN 1 PRECEDING AND
                              CURRENT ROW) AS MovAvg
FROM      [Order Details]
```

	OrderID	ProductID	UnitPrice	Quantity	Value	RunSum	MovAvg
1	10248	11	14,00	12	168,00	168,00	168,00
2	10248	42	9,80	10	98,00	266,00	133,00
3	10248	72	34,80	5	174,00	440,00	136,00
4	10249	14	18,60	9	167,40	167,40	167,40
5	10249	51	42,40	40	1696,00	1863,40	931,70
6	10250	41	7,70	10	77,00	77,00	77,00
7	10250	51	42,40	35	1484,00	1561,00	780,50

Funkcje agregujące – AVG & SUM – różne partycje?

GFT

```
SELECT    OrderID,
          ProductID,
          UnitPrice,
          Quantity,
          UnitPrice*Quantity AS Value,
          SUM(UnitPrice*Quantity) OVER (
                    PARTITION BY OrderID
                    ORDER BY ProductID
                    ROWS BETWEEN UNBOUNDED PRECEDING AND
                              CURRENT ROW) AS RunSum,
          AVG(UnitPrice*Quantity) OVER (
                    PARTITION BY ProductID
                    ORDER BY ProductID ROWS BETWEEN 3
                    PRECEDING AND CURRENT ROW) AS MovAvg
FROM      [Order Details]
```

Nie ma problemu! Trzeba tylko odpowiednio zaprezentować wyniki.

	OrderID	ProductID	UnitPrice	Quantity	Value	RunSum	MovAvg
1	10285	1	14,40	45	648,00	648,00	648,00
2	10294	1	14,40	18	259,20	259,20	453,60
3	10317	1	14,40	20	288,00	288,00	398,40
4	10348	1	14,40	15	216,00	216,00	352,80
5	10354	1	14,40	12	172,80	172,80	234,00
6	10370	1	14,40	15	216,00	216,00	223,20
7	10406	1	14,40	10	144,00	144,00	187,20
8	10413	1	14,40	24	345,60	345,60	219,60
9	10477	1	14,40	15	216,00	216,00	230,40
10	10522	1	18,00	40	720,00	720,00	356,40
11	10526	1	18,00	8	144,00	144,00	356,40
12	10576	1	18,00	10	180,00	180,00	315,00
13	10590	1	18,00	20	360,00	360,00	351,00
14	10609	1	18,00	3	54,00	54,00	184,50
15	10611	1	18,00	6	108,00	108,00	175,50
16	10628	1	18,00	25	450,00	450,00	243,00
17	10646	1	18,00	15	270,00	270,00	220,50
18	10691	1	18,00	30	540,00	540,00	342,00
19	10689	1	18,00	35	630,00	630,00	472,50
20	10700	1	18,00	5	90,00	90,00	382,50
21	10729	1	18,00	50	900,00	900,00	540,00
22	10752	1	18,00	8	144,00	144,00	441,00
23	10838	1	18,00	4	72,00	72,00	301,50
24	10847	1	18,00	80	1440...	1440,00	639,00

Funkcje agregujące – AVG & SUM – różne partycje?

GFT

```
SELECT    OrderID,
          ProductID,
          UnitPrice,
          Quantity,
          UnitPrice*Quantity AS Value,
          SUM(UnitPrice*Quantity) OVER (
                    PARTITION BY OrderID
                    ORDER BY ProductID
                    ROWS BETWEEN UNBOUNDED PRECEDING AND
                              CURRENT ROW) AS RunSum,
          AVG(UnitPrice*Quantity) OVER (
                    PARTITION BY ProductID
                    ORDER BY ProductID ROWS BETWEEN 3
                    PRECEDING AND CURRENT ROW) AS MovAvg
FROM      [Order Details]
ORDER BY  OrderID
```

	OrderID	ProductID	UnitPrice	Quantity	Value	RunSum	MovAvg
1	10248	11	14,00	12	168,00	168,00	469,80
2	10248	42	9,80	10	98,00	266,00	294,00
3	10248	72	34,80	5	174,00	440,00	377,10
4	10249	51	42,40	40	1696,00	1863,40	1335,60
5	10249	14	18,60	9	167,40	167,40	292,95
6	10250	41	7,70	10	77,00	77,00	125,125
7	10250	51	42,40	35	1484,00	1561,00	1515,80
8	10250	65	16,80	15	252,00	1813,00	383,5125
9	10251	65	16,80	20	336,00	670,80	431,025
10	10251	57	15,60	15	234,00	334,80	358,80
11	10251	22	16,80	6	100,80	100,80	697,20
12	10252	20	64,80	40	2592,00	2592,00	1494,45
13	10252	33	2,00	25	50,00	2642,00	39,625
14	10252	60	27,20	40	1088,00	3730,00	788,80
15	10253	49	16,00	40	640,00	1444,80	480,00
16	10253	31	10,00	20	200,00	200,00	362,50
17	10253	39	14,40	42	604,80	804,80	547,20
18	10254	24	3,60	15	54,00	54,00	54,00
19	10254	55	19,20	21	403,20	457,20	446,40
20	10254	74	8,00	21	168,00	625,20	119,00
21	10255	59	44,00	30	1320,00	2490,50	968,00
22	10255	16	13,90	35	486,50	790,50	559,475
23	10255	2	15,20	20	304,00	304,00	349,60
24	10255	36	15,20	25	380,00	1170,50	279,30

Funkcje agregujące – domyślne okno/ramka

GFT

```
SELECT    OD.ProductID,
          O.OrderDate,
          OD.UnitPrice*OD.Quantity AS Amount,
          SUM(UnitPrice*Quantity) OVER (
                    PARTITION BY OD.ProductID,
                    DATEPART(yyyy, O.OrderDate)) AS Total,
          SUM(UnitPrice*Quantity) OVER (
                    PARTITION BY OD.ProductID,
                    DATEPART(yyyy, O.OrderDate)
                    ORDER BY O.OrderDate) AS RunTotal
FROM      [Orders] O
JOIN      [Order Details] OD
ON        (O.OrderID = OD.OrderID)
```

	ProductID	OrderDate	Amount	Total	RunTotal
1	1	1996-08-20 00:00:00.000	648,00	1800,00	648,00
2	1	1996-08-30 00:00:00.000	259,20	1800,00	907,20
3	1	1996-09-30 00:00:00.000	288,00	1800,00	1195,20
4	1	1996-11-07 00:00:00.000	216,00	1800,00	1411,20
5	1	1996-11-14 00:00:00.000	172,80	1800,00	1584,00
6	1	1996-12-03 00:00:00.000	216,00	1800,00	1800,00
7	1	1997-01-07 00:00:00.000	144,00	5295,60	144,00
8	1	1997-01-14 00:00:00.000	345,60	5295,60	489,60
9	1	1997-03-17 00:00:00.000	216,00	5295,60	705,60
10	1	1997-04-30 00:00:00.000	720,00	5295,60	1425,60
11	1	1997-05-05 00:00:00.000	144,00	5295,60	1569,60
12	1	1997-06-23 00:00:00.000	180,00	5295,60	1749,60
13	1	1997-07-07 00:00:00.000	360,00	5295,60	2109,60
14	1	1997-07-24 00:00:00.000	54,00	5295,60	2163,60
15	1	1997-07-25 00:00:00.000	108,00	5295,60	2271,60
16	1	1997-08-12 00:00:00.000	450,00	5295,60	2721,60
17	1	1997-08-27 00:00:00.000	270,00	5295,60	2991,60
18	1	1997-10-01 00:00:00.000	630,00	5295,60	3621,60
19	1	1997-10-03 00:00:00.000	540,00	5295,60	4161,60
20	1	1997-10-10 00:00:00.000	90,00	5295,60	4251,60
21	1	1997-11-04 00:00:00.000	900,00	5295,60	5151,60
22	1	1997-11-24 00:00:00.000	144,00	5295,60	5295,60

Funkcje agregujące – domyślne okno/ramka

GFT

```
SELECT    OD.ProductID,
          O.OrderDate,
          OD.UnitPrice*OD.Quantity AS Amount,
          SUM(UnitPrice*Quantity) OVER (
                    PARTITION BY OD.ProductID,
                    DATEPART(yyyy, O.OrderDate)) AS Total,
          SUM(UnitPrice*Quantity) OVER (
                    PARTITION BY OD.ProductID,
                    DATEPART(yyyy, O.OrderDate)
                    ORDER BY O.OrderDate) AS RunTotal
FROM      [Orders] O
JOIN      [Order Details] OD
ON        (O.OrderID = OD.OrderID)
```

	ProductID	OrderDate	Amount	Total	RunTotal
1	1	1996-08-20 00:00:00.000	648,00	1800,00	648,00
2	1	1996-08-30 00:00:00.000	259,20	1800,00	907,20
3	1	1996-09-30 00:00:00.000	288,00	1800,00	1195,20
4	1	1996-11-07 00:00:00.000	216,00	1800,00	1411,20
5	1	1996-11-14 00:00:00.000	172,80	1800,00	1584,00
6	1	1996-12-03 00:00:00.000	216,00	1800,00	1800,00
7	1	1997-01-07 00:00:00.000	144,00	5295,60	144,00
8	1	1997-01-14 00:00:00.000	345,60	5295,60	489,60

Funkcje rankingowe

GFT

- Funkcje rankingowe:
  - ROW_NUMBER
  - NTILE
  - RANK
  - DENSE_RANK

Funkcje rankingowe – ROW_NUMBER - stronicowanie

GFT

```
DECLARE
          @pageNum  AS INT = 2,
          @pageSize AS INT = 25;
WITH OrdersWithRowNum AS
(
  SELECT  ROW_NUMBER() OVER (ORDER BY OrderID) as rowNum,
          OrderID, CustomerID, EmployeeID, OrderDate
  FROM    Orders
)
SELECT    OrderID, CustomerID, EmployeeID, OrderDate,
          @pageNum AS pageNum
FROM      OrdersWithRowNum
WHERE     rowNum BETWEEN (@pageNum - 1)*@pagesize + 1 AND
          @pageNum * @pageSize
```

	OrderID	CustomerID	EmployeeID	OrderDate	pageNum
1	10273	QUICK	3	1996-08-05 00:00:00.000	2
2	10274	VINET	6	1996-08-06 00:00:00.000	2
3	10275	MAGAA	1	1996-08-07 00:00:00.000	2
4	10276	TORTU	8	1996-08-08 00:00:00.000	2
5	10277	MORGK	2	1996-08-09 00:00:00.000	2
6	10278	BERGS	8	1996-08-12 00:00:00.000	2
7	10279	LEHMS	8	1996-08-13 00:00:00.000	2
8	10280	BERGS	2	1996-08-14 00:00:00.000	2
9	10281	ROMEY	4	1996-08-14 00:00:00.000	2
10	10282	ROMEY	4	1996-08-15 00:00:00.000	2
11	10283	LILAS	3	1996-08-16 00:00:00.000	2
12	10284	LEHMS	4	1996-08-19 00:00:00.000	2
13	10285	QUICK	1	1996-08-20 00:00:00.000	2
14	10286	QUICK	8	1996-08-21 00:00:00.000	2
15	10287	RICAR	8	1996-08-22 00:00:00.000	2
16	10288	REGGC	4	1996-08-23 00:00:00.000	2
17	10289	BSBEV	7	1996-08-26 00:00:00.000	2
18	10290	COMMI	8	1996-08-27 00:00:00.000	2
19	10291	QUEDE	6	1996-08-27 00:00:00.000	2
20	10292	TRADH	1	1996-08-28 00:00:00.000	2
21	10293	TORTU	1	1996-08-29 00:00:00.000	2
22	10294	RATTC	4	1996-08-30 00:00:00.000	2
23	10295	VINET	2	1996-09-02 00:00:00.000	2
24	10296	LILAS	6	1996-09-03 00:00:00.000	2
25	10297	BLONP	5	1996-09-04 00:00:00.000	2

Funkcje rankingowe – ROW_NUMBER - stronicowanie

GFT

```
DECLARE
          @pageNum  AS INT = 3,
          @pageSize AS INT = 25;
WITH OrdersWithRowNum AS
(
  SELECT  ROW_NUMBER() OVER (ORDER BY OrderID) as rowNum,
          OrderID, CustomerID, EmployeeID, OrderDate
  FROM    Orders
)
SELECT    OrderID, CustomerID, EmployeeID, OrderDate,
          @pageNum AS pageNum
FROM      OrdersWithRowNum
WHERE     rowNum BETWEEN (@pageNum - 1)*@pagesize + 1 AND
          @pageNum * @pageSize
```

	OrderID	CustomerID	EmployeeID	OrderDate	pageNum
1	10298	HUNGO	6	1996-09-05 00:00:00.000	3
2	10299	RICAR	4	1996-09-06 00:00:00.000	3
3	10300	MAGAA	2	1996-09-09 00:00:00.000	3
4	10301	WANDK	8	1996-09-09 00:00:00.000	3
5	10302	SUPRD	4	1996-09-10 00:00:00.000	3
6	10303	GODOS	7	1996-09-11 00:00:00.000	3
7	10304	TORTU	1	1996-09-12 00:00:00.000	3
8	10305	OLDWO	8	1996-09-13 00:00:00.000	3
9	10306	ROMEY	1	1996-09-16 00:00:00.000	3
10	10307	LONEP	2	1996-09-17 00:00:00.000	3
11	10308	ANATR	7	1996-09-18 00:00:00.000	3
12	10309	HUNGO	3	1996-09-19 00:00:00.000	3
13	10310	THEBI	8	1996-09-20 00:00:00.000	3
14	10311	DUMON	1	1996-09-20 00:00:00.000	3
15	10312	WANDK	2	1996-09-23 00:00:00.000	3
16	10313	QUICK	2	1996-09-24 00:00:00.000	3
17	10314	RATTC	1	1996-09-25 00:00:00.000	3
18	10315	ISLAT	4	1996-09-26 00:00:00.000	3
19	10316	RATTC	1	1996-09-27 00:00:00.000	3
20	10317	LONEP	6	1996-09-30 00:00:00.000	3
21	10318	ISLAT	8	1996-10-01 00:00:00.000	3
22	10319	TORTU	7	1996-10-02 00:00:00.000	3
23	10320	WARTH	5	1996-10-03 00:00:00.000	3
24	10321	ISLAT	3	1996-10-03 00:00:00.000	3
25	10322	PERIC	7	1996-10-04 00:00:00.000	3

Funkcje rankingowe – RANK vs. DENSE_RANK

GFT

```
SELECT    P.ProductName,
          C.CategoryName,
          P.UnitPrice,
          RANK() OVER (ORDER BY UnitPrice) AS rank,
          DENSE_RANK() OVER (ORDER BY UnitPrice)
                    AS denseRank
FROM                Products P
LEFT OUTER JOIN     Categories C
ON                  (P.CategoryID = C.CategoryID)
```

	ProductName	CategoryName	UnitPrice	rank	denseRank
1	Geitost	Dairy Products	2,50	1	1
2	Guaraná Fantástica	Beverages	4,50	2	2
3	Konbu	Seafood	6,00	3	3
4	Filo Mix	Grains/Cereals	7,00	4	4
5	Tourtière	Meat/Poultry	7,45	5	5
6	Rhönbräu Klosterbier	Beverages	7,75	6	6
7	Tunnbröd	Grains/Cereals	9,00	7	7
8	Teatime Chocolate Biscuits	Confections	9,20	8	8
9	Zaanse koeken	Confections	9,50	9	9
10	Rogede sild	Seafood	9,50	9	9
11	Jack's New England Clam Chowder	Seafood	9,65	11	10
12	Sir Rodney's Scones	Confections	10,00	12	11
13	Aniseed Syrup	Condiments	10,00	12	11
14	Longlife Tofu	Produce	10,00	12	11
15	Spegesild	Seafood	12,00	15	12
16	Scottish Longbreads	Confections	12,50	16	13
17	Gorgonzola Telino	Dairy Products	12,50	16	13
18	Chocolade	Confections	12,75	18	14
19	Original Frankfurter grüne Soße	Condiments	13,00	19	15
20	Escargots de Bourgogne	Seafood	13,25	20	16
21	Laughing Lumberjack Lager	Beverages	14,00	21	17
22	Singaporean Hokkien Fried Mee	Grains/Cereals	14,00	21	17
23	Sasquatch Ale	Beverages	14,00	21	17
24	NuNuCa Nuß-Nougat-Creme	Confections	14,00	21	17
25	Röd Kaviar	Seafood	15,00	25	18

Funkcje rankingowe – RANK vs. DENSE_RANK

GFT

```
SELECT    P.ProductName,
          C.CategoryName,
          P.UnitPrice,
          RANK() OVER (ORDER BY UnitPrice) AS rank,
          DENSE_RANK() OVER (ORDER BY UnitPrice)
                    AS denseRank
FROM                Products P
LEFT OUTER JOIN     Categories C
ON                  (P.CategoryID = C.CategoryID)
```

	ProductName	CategoryName	UnitPrice	rank	denseRank
1	Geitost	Dairy Products	2,50	1	1
...					
9	Zaanse koeken	Confections	9,50	9	9
10	Rogede sild	Seafood	9,50	9	9
11	Jack's New England Clam Chowder	Seafood	9,65	11	10
12	Sir Rodney's Scones	Confections	10,00	12	11
13	Aniseed Syrup	Condiments	10,00	12	11
14	Longlife Tofu	Produce	10,00	12	11
15	Spegesild	Seafood	12,00	15	12

Funkcje rankingowe – RANK vs. DENSE_RANK vs. ROW_NUM

GFT

```
SELECT    P.ProductName,
          C.CategoryName,
          P.UnitPrice,
          RANK() OVER (ORDER BY UnitPrice) AS rank,
          DENSE_RANK() OVER (ORDER BY UnitPrice)
                    AS denseRank,
          ROW_NUMBER() OVER (ORDER BY UnitPrice)
                    AS rowNum
FROM                Products P
LEFT OUTER JOIN     Categories C
ON                  (P.CategoryID = C.CategoryID)
```

	ProductName	CategoryName	UnitPrice	rank	denseRank	rowNum
1	Geitost	Dairy Products	2,50	1	1	1
2	Guaraná Fantástica	Beverages	4,50	2	2	2
3	Konbu	Seafood	6,00	3	3	3
4	Filo Mix	Grains/Cereals	7,00	4	4	4
5	Tourtière	Meat/Poultry	7,45	5	5	5
6	Rhönbräu Klosterbier	Beverages	7,75	6	6	6
7	Tunnbröd	Grains/Cereals	9,00	7	7	7
8	Teatime Chocolate Biscuits	Confections	9,20	8	8	8
9	Zaanse koeken	Confections	9,50	9	9	9
10	Rogede sild	Seafood	9,50	9	9	10
11	Jack's New England Clam Chowder	Seafood	9,65	11	10	11
12	Sir Rodney's Scones	Confections	10,00	12	11	12
13	Aniseed Syrup	Condiments	10,00	12	11	13
14	Longlife Tofu	Produce	10,00	12	11	14
15	Spegesild	Seafood	12,00	15	12	15
16	Scottish Longbreads	Confections	12,50	16	13	16
17	Gorgonzola Telino	Dairy Products	12,50	16	13	17
18	Chocolade	Confections	12,75	18	14	18
19	Original Frankfurter grüne Soße	Condiments	13,00	19	15	19
20	Escargots de Bourgogne	Seafood	13,25	20	16	20
21	Laughing Lumberjack Lager	Beverages	14,00	21	17	21
22	Singaporean Hokkien Fried Mee	Grains/Cereals	14,00	21	17	22
23	Sasquatch Ale	Beverages	14,00	21	17	23
24	NuNuCa Nuß-Nougat-Creme	Confections	14,00	21	17	24
25	Röd Kaviar	Seafood	15,00	25	18	25

Funkcje rankingowe – RANK vs. DENSE_RANK vs. ROW_NUM

GFT

```
SELECT    P.ProductName,
          C.CategoryName,
          P.UnitPrice,
          RANK() OVER (PARTITION BY P.CategoryID
                    ORDER BY UnitPrice) AS rank,
          DENSE_RANK() OVER (PARTITION BY P.CategoryID
                    ORDER BY UnitPrice) AS denseRank,
          ROW_NUMBER() OVER (PARTITION BY P.CategoryID
                    ORDER BY UnitPrice) AS rowNum
FROM                Products P
LEFT OUTER JOIN     Categories C
ON                  (P.CategoryID = C.CategoryID)
```

	ProductName	CategoryName	UnitPrice	rank	denseRank	rowNum
1	Guaraná Fantástica	Beverages	4,50	1	1	1
2	Rhönbräu Klosterbier	Beverages	7,75	2	2	2
3	Laughing Lumberjack Lager	Beverages	14,00	3	3	3
4	Sasquatch Ale	Beverages	14,00	3	3	4
5	Outback Lager	Beverages	15,00	5	4	5
6	Chartreuse verte	Beverages	18,00	6	5	6
7	Steeleye Stout	Beverages	18,00	6	5	7
8	Chai	Beverages	18,00	6	5	8
9	Lakkalikööri	Beverages	18,00	6	5	9
10	Chang	Beverages	19,00	10	6	10
11	Ipoh Coffee	Beverages	46,00	11	7	11
12	Côte de Blaye	Beverages	263,50	12	8	12
13	Aniseed Syrup	Condiments	10,00	1	1	1
14	Original Frankfurter grüne Soße	Condiments	13,00	2	2	2
15	Genen Shouyu	Condiments	15,50	3	3	3
16	Louisiana Hot Spiced Okra	Condiments	17,00	4	4	4
17	Gula Malacca	Condiments	19,45	5	5	5
18	Louisiana Fiery Hot Pepper S...	Condiments	21,05	6	6	6
19	Chef Anton's Gumbo Mix	Condiments	21,35	7	7	7
20	Chef Anton's Cajun Seasoning	Condiments	22,00	8	8	8
21	Grandma's Boysenberry Spread	Condiments	25,00	9	9	9
22	Sirop d'érable	Condiments	28,50	10	10	10
23	Northwoods Cranberry Sauce	Condiments	40,00	11	11	11
24	Vegie-spread	Condiments	43,90	12	12	12
25	Teatime Chocolate Biscuits	Confections	9,20	1	1	1

Funkcje rankingowe – RANK vs. DENSE_RANK vs. ROW_NUM

GFT

```
SELECT    P.ProductName,
          C.CategoryName,
          P.UnitPrice,
          RANK() OVER (PARTITION BY P.CategoryID
                    ORDER BY UnitPrice) AS rank,
          DENSE_RANK() OVER (PARTITION BY P.CategoryID
                    ORDER BY UnitPrice) AS denseRank,
          ROW_NUMBER() OVER (PARTITION BY P.CategoryID
                    ORDER BY UnitPrice) AS rowNum
FROM                Products P
LEFT OUTER JOIN     Categories C
ON                  (P.CategoryID = C.CategoryID)
```

	ProductName	CategoryName	UnitPrice	rank	denseRank	rowNum
1	Guaraná Fantástica	Beverages	4,50	1	1	1
2	Rhönbräu Klosterbier	Beverages	7,75	2	2	2
3	Laughing Lumberjack Lager	Beverages	14,00	3	3	3
4	Sasquatch Ale	Beverages	14,00	3	3	4
5	Outback Lager	Beverages	15,00	5	4	5
6	Chartreuse verte	Beverages	18,00	6	5	6
7	Steeleye Stout	Beverages	18,00	6	5	7
8	Chai	Beverages	18,00	6	5	8
9	Lakkalikööri	Beverages	18,00	6	5	9
10	Chang	Beverages	19,00	10	6	10
11	Ipoh Coffee	Beverages	46,00	11	7	11
12	Côte de Blaye	Beverages	263,50	12	8	12
13	Aniseed Syrup	Condiments	10,00	1	1	1

Funkcje rankingowe – NTILE

GFT

```
SELECT SupplierID,
       CompanyName,
       NTILE(4) OVER (
              ORDER BY CompanyName)
       AS Grp
FROM   Suppliers
```

	SupplierID	CompanyName	Grp
1	18	Aux joyeux ecclésiastiques	1
2	16	Bigfoot Breweries	1
3	5	Cooperativa de Quesos 'Las Cabras'	1
4	27	Escargots Nouveaux	1
5	1	Exotic Liquids	1
6	29	Forêts d'érables	1
7	14	Formaggi Fortini s.r.l.	1
8	28	Gai pâturage	1
9	24	G'day, Mate	2
10	3	Grandma Kelly's Homestead	2
11	11	Heli Süßwaren GmbH & Co. KG	2
12	23	Karkki Oy	2
13	20	Leka Trading	2
14	21	Lyngbysild	2
15	25	Ma Maison	2
16	6	Mayumi's	3
17	19	New England Seafood Cannery	3
18	2	New Orleans Cajun Delights	3
19	13	Nord-Ost-Fisch Handelsgesellschaft mbH	3
20	15	Norske Meierier	3
21	26	Pasta Buttini s.r.l.	3
22	7	Pavlova, Ltd.	3
23	9	PB Knäckebröd AB	4
24	12	Plutzer Lebensmittelgroßmärkte AG	4
25	10	Refrescos Americanas LTDA	4
26	8	Specialty Biscuits, Ltd.	4
27	17	Svensk Sjöföda AB	4
28	4	Tokyo Traders	4
29	22	Zaanse Snoepfabriek	4

Zawężanie wyników w funkcjach analitycznych

GFT

```
SELECT SupplierID,
       CompanyName,
       NTILE(4) OVER (
              ORDER BY CompanyName)
       AS Grp
FROM   Suppliers
WHERE  Grp = 2
WHERE  NTILE(4) OVER (ORDER BY
       CompanyName) = 2
```

	SupplierID	CompanyName	Grp
1	24	G'day, Mate	2
2	3	Grandma Kelly's Homestead	2
3	11	Heli Süßwaren GmbH & Co. KG	2
4	23	Karkki Oy	2
5	20	Leka Trading	2
6	21	Lyngbysild	2
7	25	Ma Maison	2

Zawężanie wyników w funkcjach analitycznych

GFT

```
SELECT  s.SupplierID,
        s.CompanyName,
        s.Grp
FROM (
        SELECT  SupplierID,
                CompanyName,
                NTILE(4) OVER (
                        ORDER BY CompanyName)
                        AS Grp
        FROM    Suppliers
) s
WHERE s.Grp = 2
```

```
WITH SuppliersGrp AS
(
        SELECT  SupplierID,
                CompanyName,
                NTILE(4) OVER (
                        ORDER BY CompanyName)
                        AS Grp
        FROM    Suppliers
)
SELECT  SupplierID,
        CompanyName,
        Grp
FROM    SuppliersGrp
WHERE   Grp = 2
```

Funkcje rozkładu

GFT

- **Funkcje rozkładu** dostarczają informacji o rozkładzie danych

- Funkcje rozkładu rankingu:
  - PERCENT_RANK
  - CUME_DIST
- Powyższe funkcje służą do wyznaczania tzw. percentyli, czyli określenia na jakim miejscu (wyrażonym procentowo) w uporządkowanym zbiorze znajduje się dana wartość.

- Funkcje rozkładu odwrotnego:
  - PERCENTILE_CONT
  - PERCENTILE_DISC
- Funkcje PERCENTIE_CONT oraz PERCENTILE_DISC służą do operacji odwrotnych niż 2 pierwsze funkcje – wyznaczają wartość znajdującą się na określonej pozycji w uporządkowanym zbiorze

Funkcje rozkładu - PERCENT_RANK & CUME_DIST

GFT

- Założenia:
  - **rk** – pozycja wiersza wyrażona w rankingu RANK
  - **nr** – liczba wierszy w partycji
  - **np** – liczba wierszy zajmujących niższą lub tę samą pozycję co wiersz bieżący

  - $\mathrm{PERCENT\_}RANK = \frac{rk-1}{nr-1}$
  - $CUME\_DIST = \frac{np}{nr}$

- Funkcje zwracają wartości z zakresu (0;1> (**PERCENT_RANK** przyjmuje 0 dla pierwszego wiersza)

- PERCENT_RANK – procent rekordów z rankingiem mniejszym niż bieżący
- CUME_DIST – procent rekordów z rankingiem mniejszym bądź równym niż bieżący

Funkcje rozkładu - PERCENT_RANK & CUME_DIST

GFT

```
WITH ProductsValue AS
(
  SELECT     C.CategoryName, S.CompanyName,
             SUM(P.UnitPrice * P.UnitsInStock) AS Value
  FROM       Products P
  JOIN       Suppliers S
  ON         P.SupplierID = S.SupplierID
  JOIN       Categories C
  ON         C.CategoryID = P.CategoryID
  GROUP BY   C.CategoryName, S.CompanyName
)
SELECT       CategoryName, CompanyName, Value,
             RANK() OVER (PARTITION BY CategoryName
                          ORDER BY Value) AS Rank,
             ROUND(PERCENT_RANK() OVER (PARTITION BY CategoryName
                                        ORDER BY Value),2) AS PercentRank,
             ROUND(CUME_DIST() OVER (PARTITION BY CategoryName
                                     ORDER BY Value),2) AS CumeDist
FROM         ProductsValue
```

	CategoryName	CompanyName	Value	Rank	PercentRank	CumeDist
1	Beverages	Refrescos Americanas LTDA	90,00	1	0	0,13
2	Beverages	Pavlova, Ltd.	225,00	2	0,14	0,25
3	Beverages	Leka Trading	782,00	3	0,29	0,38
4	Beverages	Plutzer Lebensmittelgroßmärkte AG	968,75	4	0,43	0,5
5	Beverages	Exotic Liquids	1025,00	5	0,57	0,63
6	Beverages	Karkki Oy	1026,00	6	0,71	0,75
7	Beverages	Bigfoot Breweries	2642,00	7	0,86	0,88
8	Beverages	Aux joyeux ecclésiastiques	5721,50	8	1	1
9	Condiments	Exotic Liquids	130,00	1	0	0,13
10	Condiments	Plutzer Lebensmittelgroßmärkte AG	416,00	2	0,14	0,25
11	Condiments	Leka Trading	525,15	3	0,29	0,38
12	Condiments	Mayumi's	604,50	4	0,43	0,5
13	Condiments	Pavlova, Ltd.	1053,60	5	0,57	0,63
14	Condiments	New Orleans Cajun Delights	2833,80	6	0,71	0,75
15	Condiments	Forêts d'érables	3220,50	7	0,86	0,88
16	Condiments	Grandma Kelly's Homestead	3240,00	8	1	1
17	Confections	Pavlova, Ltd.	506,05	1	0	0,17
18	Confections	Zaanse Snoepfabriek	533,25	2	0,2	0,33
19	Confections	Forêts d'érables	838,10	3	0,4	0,5
20	Confections	Karkki Oy	1256,25	4	0,6	0,67
21	Confections	Specialty Biscuits, Ltd.	3575,00	5	0,8	0,83
22	Confections	Heli Süßwaren GmbH & Co. KG	3683,55	6	1	1

Funkcje rozkładu - PERCENT_RANK & CUME_DIST

GFT

```
WITH ProductsValue AS
(
  SELECT     C.CategoryName, S.CompanyName,
             SUM(P.UnitPrice * P.UnitsInStock) AS Value
  FROM       Products P
  JOIN       Suppliers S
  ON         P.SupplierID = S.SupplierID
  JOIN       Categories C
  ON         C.CategoryID = P.CategoryID
  GROUP BY   C.CategoryName, S.CompanyName
)
SELECT       CategoryName, CompanyName, Value,
             RANK() OVER (PARTITION BY CategoryName
                          ORDER BY Value) AS Rank,
             ROUND(PERCENT_RANK() OVER (PARTITION BY CategoryName
                                        ORDER BY Value),2) AS PercentRank,
             ROUND(CUME_DIST() OVER (PARTITION BY CategoryName
                                     ORDER BY Value),2) AS CumeDist
FROM         ProductsValue
```

	CategoryName	CompanyName	Value	Rank	PercentRank	CumeDist
1	Beverages	Refrescos Americanas LTDA	90,00	1	0	0,13
2	Beverages	Pavlova, Ltd.	225,00	2	0,14	0,25
3	Beverages	Leka Trading	782,00	3	0,29	0,38
4	Beverages	Plutzer Lebensmittelgroßmärkte AG	968,75	4	0,43	0,5
5	Beverages	Exotic Liquids	1025,00	5	0,57	0,63
6	Beverages	Karkki Oy	1026,00	6	0,71	0,75
7	Beverages	Bigfoot Breweries	2642,00	7	0,86	0,88
8	Beverages	Aux joyeux ecclésiastiques	5721,50	8	1	1

Funkcje rozkładu odwrotnego - PERCENTILE_CONT & PERCENTILE_DISC

GFT

- Funkcja **PERCENTILE_DISC** (model dyskretny) zwraca pierwszą wartość w grupie (uporządkowanym zbiorze), dla której wartość funkcji CUME_DIST jest większa lub równa zadanej wartości wejściowej.

- Funkcja **PERCENTILE_CONT** działa podobnie do PERCENTILE_DISC z tym, że operuje na modelu ciągłym, przez co jest trochę bardziej skomplikowana. Wartość wyznaczana jest poprzez liniową interpolację wierszy otaczających wskazaną pozycję.
- Algorytm dla funkcji PERCENTILE_CONT (x):
  - $rn = (1 + x * (n - 1))$, gdzie n jest liczbą wierszy w grupie, a rn numerem wiersza
  - $value(rn)$ – wartość wiersza rn
  - Jeżeli $\lceil rn \rceil = \lfloor rn \rfloor$, to $PERCETILE\ CONT(x) = value(rn)$, w przeciwnym wypadku:

$$PERCETILE\ CONT(x) = (\lceil rn \rceil - rn) * value(\lceil rn \rceil) + (rn - \lfloor rn \rfloor) * value(\lfloor rn \rfloor)$$

Funkcje rozkładu odwrotnego - PERCENTILE_CONT & PERCENTILE_DISC

GFT

```
WITH ProductsValue AS
(
  SELECT      C.CategoryName, S.CompanyName,
              SUM(P.UnitPrice * P.UnitsInStock) AS Value
  FROM        Products P
  JOIN        Suppliers S
  ON          P.SupplierID = S.SupplierID
  JOIN        Categories C
  ON          C.CategoryID = P.CategoryID
  GROUP BY    C.CategoryName, S.CompanyName
)
SELECT        CategoryName, CompanyName, Value,
              ROUND(PERCENT_RANK() OVER (PARTITION BY CategoryName
                        ORDER BY Value),2) AS PercentRank,
              ROUND(CUME_DIST() OVER (PARTITION BY CategoryName
                        ORDER BY Value),2) AS CumeDist,
              PERCENTILE_DISC (0.5) WITHIN GROUP (ORDER BY Value) OVER (
                        PARTITION BY CategoryName) AS PercDisc,
              PERCENTILE_CONT (0.5) WITHIN GROUP (ORDER BY Value) OVER (
                        PARTITION BY CategoryName) AS PercCont
FROM ProductsValue
```

	CategoryName	CompanyName	Value	PercentRank	CumeDist	PercDisc	PercCont
23	Dairy Products	Formaggi Fortini s.r.l.	775,20	0	0,25	1775,00	2752,5
24	Dairy Products	Norske Meierier	1775,00	0,33	0,5	1775,00	2752,5
25	Dairy Products	Cooperativa de Quesos 'Las Cab...	3730,00	0,67	0,75	1775,00	2752,5
26	Dairy Products	Gai pâturage	4991,00	1	1	1775,00	2752,5
27	Grains/Cereals	G'day, Mate	266,00	0	0,2	731,50	731,5
28	Grains/Cereals	Leka Trading	364,00	0,25	0,4	731,50	731,5
29	Grains/Cereals	Plutzer Lebensmittelgroßmärkte AG	731,50	0,5	0,6	731,50	731,5
30	Grains/Cereals	Pasta Buttini s.r.l.	1500,00	0,75	0,8	731,50	731,5
31	Grains/Cereals	PB Knäckebröd AB	2733,00	1	1	731,50	731,5
32	Meat/Poultry	Plutzer Lebensmittelgroßmärkte AG	0,00	0	0,6	0,00	0
33	Meat/Poultry	Pavlova, Ltd.	0,00	0	0,6	0,00	0
34	Meat/Poultry	G'day, Mate	0,00	0	0,6	0,00	0
35	Meat/Poultry	Tokyo Traders	2813,00	0,75	0,8	0,00	0
36	Meat/Poultry	Ma Maison	2916,45	1	1	0,00	0
37	Produce	Tokyo Traders	40,00	0	0,2	813,75	813,75
38	Produce	Grandma Kelly's Homestead	450,00	0,25	0,4	813,75	813,75
39	Produce	Mayumi's	813,75	0,5	0,6	813,75	813,75
40	Produce	G'day, Mate	1060,00	0,75	0,8	813,75	813,75
41	Produce	Plutzer Lebensmittelgroßmärkte AG	1185,60	1	1	813,75	813,75

Funkcje rozkładu odwrotnego - PERCENTILE_CONT & PERCENTILE_DISC

GFT

```
WITH ProductsValue AS
(
  SELECT      C.CategoryName, S.CompanyName,
              SUM(P.UnitPrice * P.UnitsInStock) AS Value
  FROM        Products P
  JOIN        Suppliers S
  ON          P.SupplierID = S.SupplierID
  JOIN        Categories C
  ON          C.CategoryID = P.CategoryID
  GROUP BY    C.CategoryName, S.CompanyName
)
SELECT        CategoryName, CompanyName, Value,
              ROUND(PERCENT_RANK() OVER (PARTITION BY CategoryName
                          ORDER BY Value),2) AS PercentRank,
              ROUND(CUME_DIST() OVER (PARTITION BY CategoryName
                          ORDER BY Value),2) AS CumeDist,
              PERCENTILE_DISC (0.5) WITHIN GROUP (ORDER BY Value) OVER (
                          PARTITION BY CategoryName) AS PercDisc,
              PERCENTILE_CONT (0.5) WITHIN GROUP (ORDER BY Value) OVER (
                          PARTITION BY CategoryName) AS PercCont
FROM ProductsValue
```

	CategoryName	CompanyName	Value	PercentRank	CumeDist	PercDisc	PercCont
23	Dairy Products	Formaggi Fortini s.r.l.	775,20	0	0,25	1775,00	2752,5
24	Dairy Products	Norske Meierier	1775,00	0,33	0,5	1775,00	2752,5
25	Dairy Products	Cooperativa de Quesos 'Las Cab...	3730,00	0,67	0,75	1775,00	2752,5
26	Dairy Products	Gai pâturage	4991,00	1	1	1775,00	2752,5

Funkcje przesunięcia

GFT

- **Funkcje przesunięcia** w stosunku do bieżącego wiersza:
  - LAG
  - LEAD
- **Funkcje przesunięcia** względem początku i końca ramy okna:
  - FIRST_VALUE
  - LAST_VALUE

Funkcje przesunięcia – LAG & LEAD

GFT

```
SELECT   OD.OrderID,
         OD.ProductID,
         O.OrderDate,
         OD.UnitPrice*OD.Quantity AS Value,
         LAG(OD.UnitPrice*OD.Quantity) OVER (
                  PARTITION BY OD.ProductID
                  ORDER BY O.OrderDate)
                           AS PrevValue,
         LEAD(OD.UnitPrice*OD.Quantity) OVER (
                  PARTITION BY OD.ProductID
                  ORDER BY O.OrderDate)
                           AS NextValue
FROM     [Orders] O JOIN [Order Details] OD
ON       (O.OrderID = OD.OrderID)
```

	OrderID	ProductID	OrderDate	Value	PrevValue	NextValue
1	10285	1	1996-08-20 00:00:00.000	648,00	NULL	259,20
2	10294	1	1996-08-30 00:00:00.000	259,20	648,00	288,00
3	10317	1	1996-09-30 00:00:00.000	288,00	259,20	216,00
4	10348	1	1996-11-07 00:00:00.000	216,00	288,00	172,80
5	10354	1	1996-11-14 00:00:00.000	172,80	216,00	216,00
			...			
37	11047	1	1998-04-24 00:00:00.000	450,00	180,00	720,00
38	11070	1	1998-05-05 00:00:00.000	720,00	450,00	NULL
39	10255	2	1996-07-12 00:00:00.000	304,00	NULL	760,00
40	10258	2	1996-07-17 00:00:00.000	760,00	304,00	532,00
41	10264	2	1996-07-24 00:00:00.000	532,00	760,00	608,00

Funkcje przesunięcia – LAG & LEAD

GFT

```
SELECT   OD.OrderID,
         OD.ProductID,
         O.OrderDate,
         OD.UnitPrice*OD.Quantity AS Value,
         LAG(OD.UnitPrice*OD.Quantity, 2) OVER(
                  PARTITION BY OD.ProductID
                  ORDER BY O.OrderDate)
                           AS PrevValue,
         LEAD(OD.UnitPrice*OD.Quantity, 2) OVER(
                  PARTITION BY OD.ProductID
                  ORDER BY O.OrderDate)
                           AS NextValue
FROM     [Orders] O JOIN [Order Details] OD
ON       (O.OrderID = OD.OrderID)
```

	OrderID	ProductID	OrderDate	Value	PrevValue	NextValue
1	10285	1	1996-08-20 00:00:00.000	648,00	NULL	288,00
2	10294	1	1996-08-30 00:00:00.000	259,20	NULL	216,00
3	10317	1	1996-09-30 00:00:00.000	288,00	648,00	172,80
4	10348	1	1996-11-07 00:00:00.000	216,00	259,20	216,00
5	10354	1	1996-11-14 00:00:00.000	172,80	288,00	144,00
			...			
36	11035	1	1998-04-20 00:00:00.000	180,00	180,00	720,00
37	11047	1	1998-04-24 00:00:00.000	450,00	810,00	NULL
38	11070	1	1998-05-05 00:00:00.000	720,00	180,00	NULL
39	10255	2	1996-07-12 00:00:00.000	304,00	NULL	532,00
40	10258	2	1996-07-17 00:00:00.000	760,00	NULL	608,00
41	10264	2	1996-07-24 00:00:00.000	532,00	304,00	380,00

Funkcje przesunięcia – LAG & LEAD

GFT

```
SELECT    OD.OrderID,
          OD.ProductID,
          O.OrderDate,
          OD.UnitPrice*OD.Quantity AS Value,
          LAG(OD.UnitPrice*OD.Quantity, 2, 0) OVER(
                    PARTITION BY OD.ProductID
                    ORDER BY O.OrderDate)
                              AS PrevValue,
          LEAD(OD.UnitPrice*OD.Quantity, 2, 0) OVER(
                    PARTITION BY OD.ProductID
                    ORDER BY O.OrderDate)
                              AS NextValue
FROM      [Orders] O JOIN [Order Details] OD
ON        (O.OrderID = OD.OrderID)
```

	OrderID	ProductID	OrderDate	Value	PrevValue	NextValue
1	10285	1	1996-08-20 00:00:00.000	648,00	0,00	288,00
2	10294	1	1996-08-30 00:00:00.000	259,20	0,00	216,00
3	10317	1	1996-09-30 00:00:00.000	288,00	648,00	172,80
4	10348	1	1996-11-07 00:00:00.000	216,00	259,20	216,00
5	10354	1	1996-11-14 00:00:00.000	172,80	288,00	144,00
			...			
36	11035	1	1998-04-20 00:00:00.000	180,00	180,00	720,00
37	11047	1	1998-04-24 00:00:00.000	450,00	810,00	0,00
38	11070	1	1998-05-05 00:00:00.000	720,00	180,00	0,00
39	10255	2	1996-07-12 00:00:00.000	304,00	0,00	532,00
40	10258	2	1996-07-17 00:00:00.000	760,00	0,00	608,00
41	10264	2	1996-07-24 00:00:00.000	532,00	304,00	380,00

Funkcje przesunięcia – LAG & LEAD – obliczanie różnic

GFT

```
SELECT   OD.ProductID,
         DATEPART(yyyy, O.OrderDate) AS
                  OrderYear,
         SUM(UnitPrice*Quantity) AS Amount,
         SUM(UnitPrice*Quantity) -
         LAG(SUM(UnitPrice*Quantity), 1)
                  OVER (PARTITION BY OD.ProductID
                           ORDER BY DATEPART(yyyy,
                           O.OrderDate)) AS Diff
FROM     [Orders] O JOIN [Order Details] OD
ON       (O.OrderID = OD.OrderID)
GROUP BY OD.ProductID,
          DATEPART(yyyy, O.OrderDate)
```

	ProductID	OrderYear	Amount	Diff
1	1	1996	1800,00	NULL
2	1	1997	5295,60	3495,60
3	1	1998	7182,00	1886,40
4	2	1996	3435,20	NULL
5	2	1997	7600,00	4164,80
6	2	1998	7524,00	-76,00
7	3	1996	240,00	NULL
8	3	1997	1760,00	1520,00
9	3	1998	1080,00	-680,00
10	4	1996	1883,20	NULL
11	4	1997	5737,60	3854,40
12	4	1998	1804,00	-3933,60
13	5	1996	2193,00	NULL
14	5	1997	405,65	-1787,35
15	5	1998	3202,50	2796,85
16	6	1996	720,00	NULL
17	6	1997	2500,00	1780,00
18	6	1998	4125,00	1625,00
19	7	1996	600,00	NULL
20	7	1997	9444,00	8844,00
21	7	1998	12420...	2976,00
22	8	1996	4480,00	NULL
23	8	1997	4560,00	80,00
24	8	1998	4720,00	160,00
25	9	1997	8536,00	NULL

Funkcje przesunięcia – LAG & LEAD – obliczanie różnic

GFT

```
SELECT  OD.ProductID,
        DATEPART(yyyy, O.OrderDate) AS
                OrderYear,
        SUM(UnitPrice*Quantity) AS Amount,
        SUM(UnitPrice*Quantity) -
        LAG(SUM(UnitPrice*Quantity), 1)
                OVER (PARTITION BY OD.ProductID
                        ORDER BY DATEPART(yyyy,
                        O.OrderDate)) AS Diff
FROM    [Orders] O JOIN [Order Details] OD
ON      (O.OrderID = OD.OrderID)
GROUP BY OD.ProductID,
         DATEPART(yyyy, O.OrderDate)
```

	ProductID	OrderYear	Amount	Diff
1	1	1996	1800,00	NULL
2	1	1997	5295,60	3495,60
3	1	1998	7182,00	1886,40
4	2	1996	3435,20	NULL
5	2	1997	7600,00	4164,80
6	2	1998	7524,00	-76,00

Funkcje przesunięcia – FIRST_VALUE, LAST_VALUE

GFT

```
SELECT    P.ProductName,
          C.CategoryName,
          P.UnitPrice,
          FIRST_VALUE(P.UnitPrice) OVER (
                    PARTITION BY      P.CategoryID
                    ORDER BY          P.UnitPrice
                    ROWS BETWEEN 2 PRECEDING
                    AND 2 FOLLOWING) AS FirstValue,
          LAST_VALUE(P.UnitPrice) OVER (
                    PARTITION BY      P.CategoryID
                    ORDER BY          P.UnitPrice
                    ROWS BETWEEN 2 PRECEDING
                    AND 2 FOLLOWING) as LastValue
FROM      Products P LEFT OUTER JOIN Categories C
ON        (P.CategoryID = C.CategoryID)
```

	ProductName	CategoryName	UnitPrice	FirstValue	LastValue
1	Guaraná Fantástica	Beverages	4,50	4,50	14,00
2	Rhönbräu Klosterbier	Beverages	7,75	4,50	14,00
3	Laughing Lumberjack Lager	Beverages	14,00	4,50	15,00
4	Sasquatch Ale	Beverages	14,00	7,75	18,00
5	Outback Lager	Beverages	15,00	14,00	18,00
6	Chartreuse verte	Beverages	18,00	14,00	18,00
7	Steeleye Stout	Beverages	18,00	15,00	18,00
8	Chai	Beverages	18,00	18,00	19,00
9	Lakkalikööri	Beverages	18,00	18,00	46,00
10	Chang	Beverages	19,00	18,00	263,50
11	Ipoh Coffee	Beverages	46,00	18,00	263,50
12	Côte de Blaye	Beverages	263,50	19,00	263,50
13	Aniseed Syrup	Condiments	10,00	10,00	15,50
14	Original Frankfurter grüne Soße	Condiments	13,00	10,00	17,00
15	Genen Shouyu	Condiments	15,50	10,00	19,45
16	Louisiana Hot Spiced Okra	Condiments	17,00	13,00	21,05
17	Gula Malacca	Condiments	19,45	15,50	21,35
18	Louisiana Fiery Hot Pepper S...	Condiments	21,05	17,00	22,00
19	Chef Anton's Gumbo Mix	Condiments	21,35	19,45	25,00
20	Chef Anton's Cajun Seasoning	Condiments	22,00	21,05	28,50
21	Grandma's Boysenberry Spread	Condiments	25,00	21,35	40,00
22	Sirop d'érable	Condiments	28,50	22,00	43,90
23	Northwoods Cranberry Sauce	Condiments	40,00	25,00	43,90
24	Vegie-spread	Condiments	43,90	28,50	43,90
25	Teatime Chocolate Biscuits	Confections	9,20	9,20	10,00

Funkcje przesunięcia – FIRST_VALUE, LAST_VALUE

GFT

```
SELECT    P.ProductName,
          C.CategoryName,
          P.UnitPrice,
          FIRST_VALUE(P.UnitPrice) OVER (
                    PARTITION BY      P.CategoryID
                    ORDER BY          P.UnitPrice
                    ROWS BETWEEN 2 PRECEDING
                    AND 2 FOLLOWING) AS FirstValue,
          LAST_VALUE(P.UnitPrice) OVER (
                    PARTITION BY      P.CategoryID
                    ORDER BY          P.UnitPrice
                    ROWS BETWEEN 2 PRECEDING
                    AND 2 FOLLOWING) as LastValue
FROM      Products P LEFT OUTER JOIN Categories C
ON        (P.CategoryID = C.CategoryID)
```

	ProductName	CategoryName	UnitPrice	FirstValue	LastValue
1	Guaraná Fantástica	Beverages	4,50	4,50	14,00
2	Rhönbräu Klosterbier	Beverages	7,75	4,50	14,00
3	Laughing Lumberjack Lager	Beverages	14,00	4,50	15,00
4	Sasquatch Ale	Beverages	14,00	7,75	18,00
5	Outback Lager	Beverages	15,00	14,00	18,00
6	Chartreuse verte	Beverages	18,00	14,00	18,00
7	Steeleye Stout	Beverages	18,00	15,00	18,00
8	Chai	Beverages	18,00	18,00	19,00
9	Lakkalikööri	Beverages	18,00	18,00	46,00
10	Chang	Beverages	19,00	18,00	263,50
11	Ipoh Coffee	Beverages	46,00	18,00	263,50
12	Côte de Blaye	Beverages	263,50	19,00	263,50

Funkcje przesunięcia – FIRST_VALUE, LAST_VALUE

GFT

```
SELECT    O.OrderID,
          C.CompanyName,
          CAST(O.OrderDate AS date) AS OrderDate,
          FIRST_VALUE(O.OrderID) OVER (
                    PARTITION BY      C.CustomerID
                    ORDER BY          O.OrderDate
                    ROWS BETWEEN 2 PRECEDING
                    AND 2 FOLLOWING) AS FirstValue,
          LAST_VALUE(O.OrderID) OVER (
                    PARTITION BY      C.CustomerID
                    ORDER BY          O.OrderDate
                    ROWS BETWEEN 2 PRECEDING
                    AND 2 FOLLOWING) AS LastValue
FROM      Orders O JOIN Customers C
ON        O.CustomerID = C.CustomerID
```

	OrderID	CompanyName	OrderDate	FirstValue	LastValue
1	10643	Alfreds Futterkiste	1997-08-25	10643	10702
2	10692	Alfreds Futterkiste	1997-10-03	10643	10835
3	10702	Alfreds Futterkiste	1997-10-13	10643	10952
4	10835	Alfreds Futterkiste	1998-01-15	10692	11011
5	10952	Alfreds Futterkiste	1998-03-16	10702	11011
6	11011	Alfreds Futterkiste	1998-04-09	10835	11011
7	10308	Ana Trujillo Emparedados y helados	1996-09-18	10308	10759
8	10625	Ana Trujillo Emparedados y helados	1997-08-08	10308	10926
9	10759	Ana Trujillo Emparedados y helados	1997-11-28	10308	10926
10	10926	Ana Trujillo Emparedados y helados	1998-03-04	10625	10926
11	10365	Antonio Moreno Taquería	1996-11-27	10365	10535
12	10507	Antonio Moreno Taquería	1997-04-15	10365	10573
13	10535	Antonio Moreno Taquería	1997-05-13	10365	10677
14	10573	Antonio Moreno Taquería	1997-06-19	10507	10682
15	10677	Antonio Moreno Taquería	1997-09-22	10535	10856
16	10682	Antonio Moreno Taquería	1997-09-25	10573	10856
17	10856	Antonio Moreno Taquería	1998-01-28	10677	10856
18	10355	Around the Horn	1996-11-15	10355	10453
19	10383	Around the Horn	1996-12-16	10355	10558
20	10453	Around the Horn	1997-02-21	10355	10707
21	10558	Around the Horn	1997-06-04	10383	10741
22	10707	Around the Horn	1997-10-16	10453	10743
23	10741	Around the Horn	1997-11-14	10558	10768
24	10743	Around the Horn	1997-11-17	10707	10793
25	10768	Around the Horn	1997-12-08	10741	10864

Funkcje przesunięcia – FIRST_VALUE, LAST_VALUE

GFT

```
SELECT    O.OrderID,
          C.CompanyName,
          CAST(O.OrderDate AS date) AS OrderDate,
          FIRST_VALUE(O.OrderID) OVER (
                    PARTITION BY      C.CustomerID
                    ORDER BY          O.OrderDate
                    ROWS BETWEEN 2 PRECEDING
                    AND 2 FOLLOWING) AS FirstValue,
          LAST_VALUE(O.OrderID) OVER (
                    PARTITION BY      C.CustomerID
                    ORDER BY          O.OrderDate
                    ROWS BETWEEN 2 PRECEDING
                    AND 2 FOLLOWING) AS LastValue
FROM      Orders O JOIN Customers C
ON        O.CustomerID = C.CustomerID
```

	OrderID	CompanyName	OrderDate	FirstValue	LastValue
1	10643	Alfreds Futterkiste	1997-08-25	10643	10702
2	10692	Alfreds Futterkiste	1997-10-03	10643	10835
3	10702	Alfreds Futterkiste	1997-10-13	10643	10952
4	10835	Alfreds Futterkiste	1998-01-15	10692	11011
5	10952	Alfreds Futterkiste	1998-03-16	10702	11011
6	11011	Alfreds Futterkiste	1998-04-09	10835	11011
7	10308	Ana Trujillo Emparedados y helados	1996-09-18	10308	10759
8	10625	Ana Trujillo Emparedados y helados	1997-08-08	10308	10926
9	10759	Ana Trujillo Emparedados y helados	1997-11-28	10308	10926
10	10926	Ana Trujillo Emparedados y helados	1998-03-04	10625	10926

Podsumowanie

GFT

- Funkcje analityczne umożliwiają przeprowadzanie mniej lub bardziej skomplikowanych analiz po stronie bazy danych

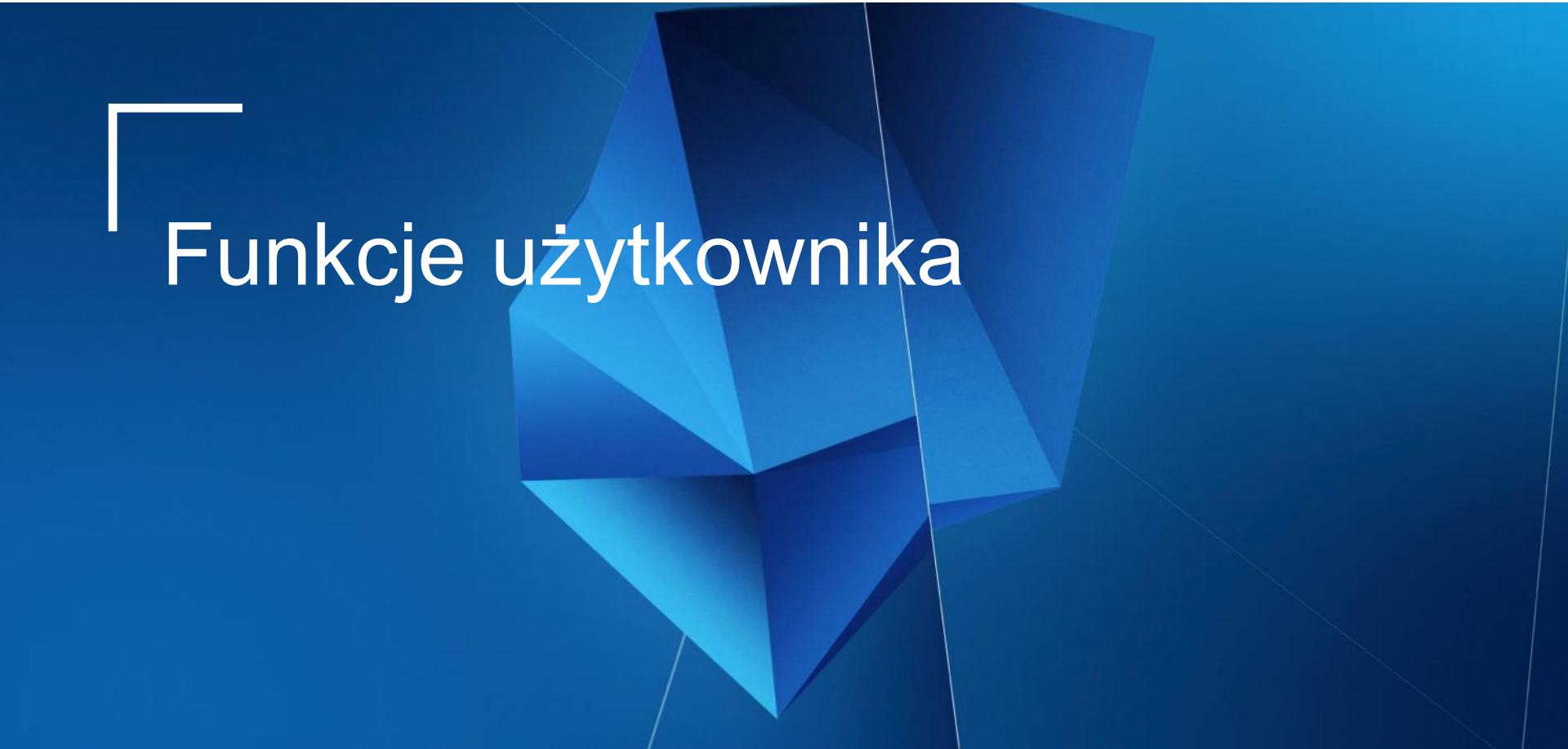
- Funkcje analityczne pomagają między innymi w rozwiązywanie takich problemów jak:
  - Stronicowanie
  - Usuwanie powtarzających się danych
  - Zwracanie **n** pierwszych wierszy dla każdej grupy
  - Obliczanie sum bieżących
  - Identyfikowanie luk i wysp
  - Obliczania centyli
  - Wyliczaniu wartości modalnej dla rozkładu
  - …

Podsumowanie

GFT

- Mogą być używane tylko w sekcjach: **SELECT** i **ORDER BY**
- Pamiętajmy o wartościach **NULL**
- Pamiętajmy o wydajności!
- Pamiętajmy o różnej nomenklaturze, która jest używana w odniesieniu do funkcji analitycznych/okna

GFT



Funkcje użytkownika

Funkcje użytkownika

GFT

- Funkcje użytkownika *(user-defined functions)* są to obiekty stworzone przez użytkownika, realizujące pewną logikę i zwracające dane

- Funkcje mogą zwracać pojedynczą wartość danego typu (funkcje skalarne) lub tabelę (funkcje tabelaryczne)

Funkcje użytkownika vs procedury składowane

GFT

Function (UDF – User Defined Function)	Stored Procedure (SP)
UDF zwraca jedynie jedną wartość – obowiązkowo!	SP może zwrócić zero, pojedynczą lub wiele wartości
Nie można używać transakcji	Można używać transakcji
Tylko parametry wejściowe	SP mogą posiadać parametry wejściowe oraz wyjściowe
Nie można zawołać SP z funkcji	Można zawołać funkcję z SP
Funkcje można używać w poleceniach SELECT / WHERE / HAVING	Nie można używać SP w poleceniach SELECT / WHERE / HAVING
Nie można używać TRY…CATCH	Można używać TRY…CATCH

Funkcje użytkownika - zalety

GFT

- Pozwalają raz stworzony kod przechować w bazie danych i używać wielokrotnie
- Zwiększają czytelność kodu
- W niektórych przypadkach pozwalają zwiększyć wydajność przechowywanie planów zapytania skompilowanych funkcji (podobnie jak w przypadku procedur składowanych)
- Pozwalają zrealizować zadania i obliczenia, które trudno lub niemożliwe jest do wykonania w czystym SQL (T-SQL).
- Posiadają możliwość wywołań rekurencyjnych
- Pozwalają zmniejszyć ruch sieciowy

Funkcje użytkownika – niektóre ograniczenia

GFT

- Nie mogą modyfikować danych
- Nie mogą zwracać kilka zbiorów danych (*result sets*), w przeciwieństwie do procedur (w MS SQL!)
- Ograniczona obsługa błędów (nie wspierane TRY…CATCH, @ERROR, RAISERROR)
- Nie wolno używać dynamicznego SQL i tabel tymczasowych
- … -> https://msdn.microsoft.com/en-us/library/ms191320.aspx

Funkcje skalarne

GFT

```
IF OBJECT_ID (N'dbo.averageUnitPrice', N'FN') IS NOT NULL
    DROP FUNCTION dbo.averageUnitPrice;
GO
CREATE FUNCTION dbo.averageUnitPrice(@CategoryID INT)
RETURNS MONEY
AS
BEGIN
    DECLARE @avgUnitPrice MONEY;

    SELECT @avgUnitPrice = AVG(p.UnitPrice)
    FROM Products P
    WHERE P.CategoryID = @CategoryID;

    IF (@avgUnitPrice IS NULL)
        SET @avgUnitPrice = 0;
    RETURN @avgUnitPrice;
END;
GO
```

Kod odpowiedzialny za usuwanie funkcji przed utworzeniem

Nagłówek funkcji, parametr, typ zwracany

Ciało funkcji

Funkcje skalarne – przykład użycia

GFT

```
SELECT ROUND(dbo.averageUnitPrice(1),2)
       AS averageUnitPriceInCategory
```

	averageUnitPriceInCategory
1	37,98

```
SELECT    CategoryID,
          ROUND (dbo.averageUnitPrice(CategoryID),2)
                    AS averageUnitPriceInCategory
FROM      Categories
```

	CategoryID	averageUnitPriceInCategory
1	1	37,98
2	2	23,06
3	3	25,16
4	4	28,73
5	5	20,25
6	6	54,01
7	7	32,37
8	8	20,68

Funkcje skalarne – rekurencja – liczymy silnię

GFT

```
CREATE FUNCTION dbo.factorial(@N INT)
RETURNS BIGINT
AS
BEGIN
    DECLARE @ret BIGINT;

    IF @N = 1
        SELECT @ret = 1
    ELSE
        SELECT @ret =
            @N * dbo.factorial(@N-1);

    RETURN @ret;
END;
```

```
SELECT dbo.factorial(5) AS Factorial
```

	Factorial
1	120

```
SELECT dbo.factorial(7) AS Factorial
```

	Factorial
1	5040

Funkcje tabelaryczne

GFT

- Tabelaryczne funkcje użytkownika możemy podzielić na:
  - Funkcje proste *(inline)*
  - Funkcje złożone
- Tabelaryczne funkcje użytkownika często używane są z operatorem APPLY (więcej w następnym rozdziale)

Funkcje tabelaryczne – proste (inline)

GFT

```
IF OBJECT_ID (N'dbo.averageUnitPriceTab', N'IF') IS NOT NULL
    DROP FUNCTION dbo.averageUnitPriceTab;
GO
CREATE FUNCTION dbo.averageUnitPriceTab(@CategoryID INT)
RETURNS TABLE
AS
RETURN (
  SELECT  ProductID,
          @CategoryID AS CategoryID,
          SupplierID,
          AVG(p.UnitPrice) OVER (PARTITION BY SupplierID)
                    AS AverageUnitPrice
  FROM  Products P
  WHERE CategoryID = @CategoryID
)
GO
```

Kod odpowiedzialny za usuwanie funkcji przed utworzeniem

Nagłówek funkcji, parametr, typ zwracany - TABLE

Ciało funkcji zawierające jedno zapytanie zawierające wynik funkcji

Funkcje tabelaryczne – proste (inline)

GFT

```
SELECT ProductID, CategoryID, SupplierID, AverageUnitPrice
FROM   dbo.averageUnitPriceTab(4)
```

	ProductID	CategoryID	SupplierID	AverageUnitPrice
1	11	4	5	29,50
2	12	4	5	29,50
3	31	4	14	26,4333
4	32	4	14	26,4333
5	72	4	14	26,4333
6	69	4	15	20,00
7	71	4	15	20,00
8	33	4	15	20,00
9	59	4	28	44,50
10	60	4	28	44,50

Funkcje tabelaryczne – proste (inline)

GFT

```
SELECT T.ProductID, T.CategoryID, T.SupplierID, T.AverageUnitPrice, C.CategoryName
FROM   dbo.averageUnitPriceTab(4) T JOIN Categories C
ON     (T.CategoryID = C.CategoryID)
WHERE  AverageUnitPrice > 25
```

	ProductID	CategoryID	SupplierID	AverageUnitPrice	CategoryName
1	11	4	5	29,50	Dairy Products
2	12	4	5	29,50	Dairy Products
3	31	4	14	26,4333	Dairy Products
4	32	4	14	26,4333	Dairy Products
5	72	4	14	26,4333	Dairy Products
6	59	4	28	44,50	Dairy Products
7	60	4	28	44,50	Dairy Products

Funkcje tabelaryczne - złożone

GFT

```
IF OBJECT_ID (N'dbo.distinctRegionsAndCities', N'TF') IS NOT NULL
    DROP FUNCTION dbo.distinctRegionsAndCities;
GO
CREATE FUNCTION dbo.distinctRegionsAndCities(@ProductID INT)
RETURNS @result TABLE
(
  ProductID   INT,
  Region      NVARCHAR(15),
  City        NVARCHAR(15)
)
AS
BEGIN
  INSERT INTO @result(ProductID, Region, City)
  SELECT      DISTINCT OD.ProductID, O.ShipRegion, O.ShipCity
  FROM        [Orders] O
  JOIN        [Order Details] OD ON (O.OrderID = OD.OrderID)
  WHERE       OD.ProductID = COALESCE(@ProductID, OD.ProductID);

RETURN
END
GO
```

Kod odpowiedzialny za usuwanie funkcji przed utworzeniem

Nagłówek funkcji, parametry, typ zwracany - tabela

Ciało funkcji

Funkcje tabelaryczne - złożone

GFT

```
SELECT ProductID, Region, City
FROM   dbo.distinctRegionsAndCities(4);
```

	ProductID	Region	City
1	4	NULL	Bergamo
2	4	NULL	Bräcke
3	4	NULL	Charleroi
4	4	NULL	Cunewalde
5	4	NULL	Lisboa
6	4	NULL	London
7	4	NULL	Luleå
8	4	NULL	Madrid
9	4	NULL	Marseille
10	4	NULL	Oulu
11	4	Co. Cork	Cork
12	4	NM	Albuquerque
13	4	Québec	Montréal
14	4	SP	Sao Paulo
15	4	WA	Seattle

Funkcje tabelaryczne - złożone

GFT

```
SELECT ProductID, Region, City
FROM   dbo.distinctRegionsAndCities(NULL)
```

	ProductID	Region	City
1	1	NULL	Bern
2	1	NULL	Charleroi
3	1	NULL	Cunewalde
4	1	NULL	Frankfurt a.M.
5	1	NULL	Helsinki
6	1	NULL	Lisboa
7	1	NULL	London
8	1	NULL	Luleå
9	1	NULL	México D.F.
10	1	NULL	Nantes
11	1	NULL	Oulu
12	1	NULL	Sevilla
13	1	NULL	Strasbourg
14	1	NULL	Stuttgart
15	1	NULL	Toulouse

Funkcje tabelaryczne - złożone

GFT

```
SELECT   ProductID, Region, City
FROM     dbo.distinctRegionsAndCities(NULL)
ORDER BY City
```

	ProductID	Region	City
1	11	NULL	Aachen
2	13	NULL	Aachen
3	26	NULL	Aachen
4	31	NULL	Aachen
5	41	NULL	Aachen
6	53	NULL	Aachen
7	59	NULL	Aachen
8	75	NULL	Aachen
9	76	NULL	Aachen
10	72	NM	Albuquerque
11	73	NM	Albuquerque
12	71	NM	Albuquerque
13	62	NM	Albuquerque
14	64	NM	Albuquerque
15	65	NM	Albuquerque

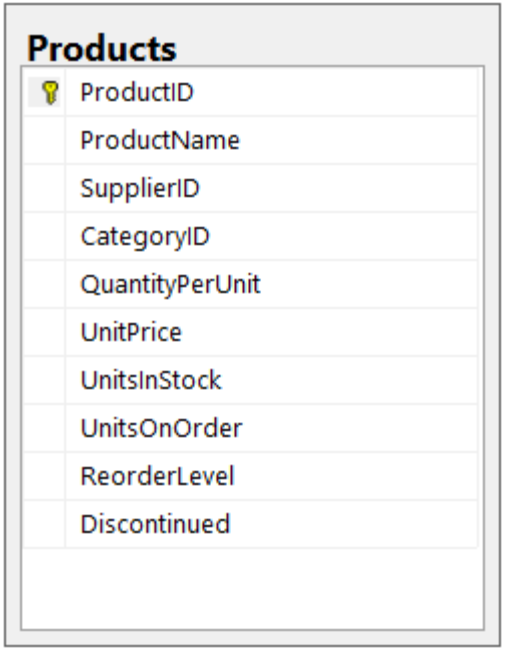
Funkcje tabelaryczne - złożone

GFT

Czy można wyznaczyć unikalny region i miasto dla każdego produktu z tabeli **Products** (wyświetlając również dane z tabeli Products: **ProductName**)?

```
SELECT P.ProductName, F.City, F.Region
FROM   Products P
JOIN   (SELECT ProductID, City, Region FROM
dbo.distinctRegionsAndCities(NULL)) F
ON     F.ProductID = P.ProductID
```

Można, ale są lepsze sposoby łączenia funkcji tabelarycznych z innymi tabelami – operator APPLY



Funkcje tabelaryczne - złożone

GFT

	ProductName	City	Region
1	Chai	Bern	NULL
2	Chai	Charleroi	NULL
3	Chai	Cunewalde	NULL
4	Chai	Frankfurt a.M.	NULL
5	Chai	Helsinki	NULL
6	Chai	Lisboa	NULL
7	Chai	London	NULL
8	Chai	Luleå	NULL
9	Chai	México D.F.	NULL
10	Chai	Nantes	NULL
11	Chai	Oulu	NULL
12	Chai	Sevilla	NULL
13	Chai	Strasbourg	NULL
14	Chai	Stuttgart	NULL
15	Chai	Toulouse	NULL

Podsumowanie

GFT

- Funkcje użytkownika dzielą się na: skalarne i tabelaryczne (proste i złożone)
- W przedstawionej formie dotyczą jedynie bazy danych MS SQL Server
- Pozwalają zrealizować logikę, którą trudno zrealizować w samym SQL
- Mogą wpływać pozytywnie na wydajność
- Posiadają ograniczenia (patrz: dokumentacja)

GFT

Operator APPLY

Operator APPLY

GFT

- Operator **APPLY** pozwala na połączenie dwóch wyrażeń tablicowych (*table expressions*).
- Wyrażenie tablicowe jest wywoływane dla każdego wiersza z lewej strony operatora.
- Posiada dwie formy: **CROSS APPLY** oraz **OUTER APPLY**
  - CROSS APPLY – zwraca tylko wiersze z zewnętrznej tabeli (lewa strona) jeżeli istnieje odpowiednik w tabeli wewnętrznej (prawa strona)
  - OUTER APPLY – zwraca wszystkie rekordy z tabeli zewnętrznej (lewa strona) niezależnie czy istnieje odpowiednik w tabeli wewnętrznej (prawa strona)
- Pozwala korelować zapytania
- Często używany w połączeniu z tabelarycznymi funkcjami użytkownika

Operator APPLY – połączenie tabeli i funkcji

GFT

```
SELECT  P.ProductName, F.City, F.Region
FROM    Products P
JOIN    (SELECT ProductID, City, Region FROM
        dbo.distinctRegionsAndCities(NULL)) F
ON      F.ProductID = P.ProductID
```

```
SELECT  P.ProductName, F.City, F.Region
FROM    Products P
CROSS APPLY
        dbo.distinctRegionsAndCities(
        P.ProductID) AS F
```

Jak można porównać czy jest taki sam rezultat?

Operator APPLY – porównanie rezultatów

GFT

```
SELECT P.ProductName, F.City, F.Region
FROM   Products P
JOIN   (SELECT ProductID, City, Region FROM dbo.distinctRegionsAndCities(NULL)) F
ON     F.ProductID = P.ProductID
EXCEPT
SELECT P.ProductName, F.City, F.Region
FROM   Products P
CROSS APPLY dbo.distinctRegionsAndCities(P.ProductID) AS F
```

Wystarczające sprawdzenie?

Operator APPLY – porównanie rezultatów

GFT

```
(SELECT   P.ProductName, F.City, F.Region
FROM      Products P
JOIN      (SELECT ProductID, City, Region FROM dbo.distinctRegionsAndCities(NULL)) F
ON        F.ProductID = P.ProductID
EXCEPT
SELECT    P.ProductName, F.City, F.Region
FROM      Products P
CROSS APPLY dbo.distinctRegionsAndCities(P.ProductID) AS F)
UNION ALL
(SELECT   P.ProductName, F.City, F.Region
FROM      Products P
CROSS APPLY dbo.distinctRegionsAndCities(P.ProductID) AS F
EXCEPT
SELECT    P.ProductName, F.City, F.Region
FROM      Products P
JOIN      (SELECT ProductID, City, Region FROM dbo.distinctRegionsAndCities(NULL)) F
ON        F.ProductID = P.ProductID)
```

Operator APPLY – połączenie dwóch tabel

GFT

JOIN:

```
SELECT P.ProductName, C.CategoryName
FROM Products P LEFT OUTER JOIN
Categories C
ON P.CategoryID = C.CategoryID
```

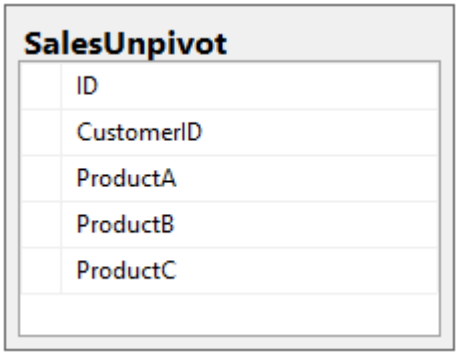
APPLY:

```
SELECT P.ProductName, C.CategoryName
FROM Products P
OUTER APPLY
(
        SELECT C2.CategoryName
        FROM   Categories C2
        WHERE  C2.CategoryID =
                        P.CategoryID
) C
```

Operator APPLY – UNPIVOT przy wykorzystaniu APPLY?

GFT

- Mamy tabelę:



	ID	CustomerID	ProductA	ProductB	ProductC
1	1	100	32	44	55
2	2	101	22	13	0
3	2	102	123	2	12
4	2	103	66	72	14
5	2	104	22	32	63

- Chcemy przekształcić i zaprezentować wynik w postaci:

	ID	CustomerID	ProductCode	Quantity
1	1	100	ProductA	32
2	1	100	ProductB	44
3	1	100	ProductC	55
4	2	101	ProductA	22
5	2	101	ProductB	13
6	2	101	ProductC	0
7	2	102	ProductA	123
8	2	102	ProductB	2
9	2	102	ProductC	12
10	2	103	ProductA	66
11	2	103	ProductB	72
12	2	103	ProductC	14
13	2	104	ProductA	22
14	2	104	ProductB	32
15	2	104	ProductC	63

Operator APPLY – UNPIVOT przy wykorzystaniu APPLY?

GFT

UNPIVOT:

```
SELECT  ID, CustomerID, ProductCode,
        Quantity
FROM
(
  SELECT ID, CustomerID, ProductA,
         ProductB,  ProductC
  FROM   SalesUnpivot) p
UNPIVOT
  (Quantity FOR ProductCode IN (
        ProductA,
        ProductB, ProductC )) AS unpvt
```

CROSS APPLY:

```
SELECT S.ID, S.CustomerID,
        C.ProductCode,C.Quantity
FROM    SalesUnpivot S
CROSS APPLY (VALUES
                ('ProductA',ProductA),
                ('ProductB', ProductB),
                ('ProductC', ProductC))
                C (ProductCode, Quantity)
```

Podsumowanie

GFT

- Operator **APPLY** pozwala na połączenie dwóch wyrażeń tablicowych (*table expressions*).
- W niektórych przypadkach jest wydajniejszy niż zwykły JOIN

GFT

Co warto jeszcze wiedzieć?

Co warto jeszcze wiedzieć?

GFT

- Widoki, widoki zmaterializowane (widoki z indeksem klastrowym w SQL Server)
- Tabele tymczasowe
- Wyzwalacze (*triggers*)
- Sekwencery (*sequences*)
- Zarządzanie transakcjami
- Optymalizacja

Co warto jeszcze wiedzieć?

GFT

- Klauzula MODEL – Oracle
- T-SQL/PLSQL – kolekcje, instrukcje sterujące
- Dobre praktyki
- Obsługa błędów
- …

Bibliografia i linki

- Itzik Ben-Gan, *Microsoft SQL Server 2012. Optymalizacja kwerend T-SQL przy użyciu funkcji okna*
- Materiały wykładowe dot. baz danych (funkcje analityczne) z Politechniki Poznańskiej
- Linki:
  - http://dba-presents.com/index.php/databases/sql-server/36-order-by-and-nulls-last-in-sql-server
  - http://www.sqlpedia.pl/
  - https://mndevnotes.wordpress.com/2012/10/03/grupowanie-danych-przy-uzyciu-polecen-rollup-cube-oraz-grouping-sets/
  - https://edu.pjwstk.edu.pl/
  - https://msdn.microsoft.com
  - https://oracle-base.com/articles/misc/rollup-cube-grouping-functions-and-grouping-sets
  - https://technet.microsoft.com
  - https://oracle-base.com/articles/11g/pivot-and-unpivot-operators-11gr1
  - https://explainextended.com/2009/07/16/inner-join-vs-cross-apply/
  - http://stackoverflow.com/questions/1179758/function-vs-stored-procedure-in-sql-server

GFT

Dziękuję za uwagę

Arkadiusz Kasprzak

akasprzak@wmi.amu.edu.pl