Hurtownie danych
Laboratorium Czw 11:15

Lista 3

Kajetan Pynka 254495

Zad 1.1a

```
SELECT ISNULL(P.FirstName + ' ' + P.LastName, '') "Klient",
ISNULL(STR(YEAR(SOH.DueDate)), '') "Rok",
    SUM(SOH.TotalDue) "Kwota"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.Customer C ON C.CustomerID=SOH.CustomerID
    JOIN Person.Person P ON P.BusinessEntityID=C.PersonID
    GROUP BY GROUPING SETS (
        (YEAR(SOH.DueDate)),
        (P.FirstName + ' ' + P.LastName, YEAR(SOH.DueDate)),
        (P.FirstName + ' ' + P.LastName),
        ()
    )
    ORDER BY 1;
```

	Klient	Rok	Kwota
1			123216786,1159
2		2013	48181124,1984
3		2014	26072957,9986
4		2011	13903981,3182
5		2012	35058722,6007
6	A. Leonetti	2013	1814,1819
7	A. Leonetti	2014	1586,6583
8	A. Leonetti		3400,8402
9	Aaron Adams	2013	130,3458
10	Aaron Adams		130,3458
11	Aaron Alexander	2014	77,339
12	Aaron Alexander		77,339

Wniosek: GROUP BY GROUPING SETS pozwala w bardzo prosty i czytelny sposób zdefiniować jakie konkretne grupy nas interesują (szczególnie z uwzględnieniem wartości NULL). W ten sposób za pomocą jednej kwerendy otrzymujemy sumę kwot transakcji, sumę kwot w podziale na rok oraz sumę kwot w podziale na rok i na klienta.

Zad 1.1b

```
SELECT ISNULL(P.FirstName + ' ' + P.LastName, '') "Klient",
ISNULL(STR(YEAR(SOH.DueDate)), '') "Rok",
    SUM(SOH.TotalDue) "Kwota"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.Customer C ON C.CustomerID=SOH.CustomerID
    JOIN Person.Person P ON P.BusinessEntityID=C.PersonID
    GROUP BY CUBE (YEAR(SOH.DueDate), P.FirstName + ' ' + P.LastName)
    ORDER BY 1;
```

	Klient	Rok	Kwota
1			123216786,1159
2		2013	48181124,1984
3		2014	26072957,9986
4		2011	13903981,3182
5		2012	35058722,6007
6	A. Leonetti	2013	1814,1819
7	A. Leonetti	2014	1586,6583
8	A. Leonetti		3400,8402
9	Aaron Adams	2013	130,3458
10	Aaron Adams		130,3458
11	Aaron Alexander	2014	77,339
12	Aaron Alexander		77,339

Wniosek: GROUP BY CUBE sprowadza się do tego samego wyniku co w poprzednim podpunkcie, ponieważ następuje grupowanie „każdy z każdym” (z uwzględnieniem wartości NULL), czyli tak jak ręcznie podałem w poprzednim podpunkcie. Jeśli jesteśmy pewni, że chcemy mieć wszystkie kombinacje to możemy użyć CUBE. Ogólnie warto się zastanowić czy faktycznie potrzebne są nam wszystkie kombinacje, gdyż będzie to wiązać się z dużym obciążeniem wydajnościowym.

Zad 1.1c

```
SELECT ISNULL(P.FirstName + ' ' + P.LastName, '') "Klient",
ISNULL(STR(YEAR(SOH.DueDate)), '') "Rok",
    SUM(SOH.TotalDue) "Kwota"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.Customer C ON C.CustomerID=SOH.CustomerID
    JOIN Person.Person P ON P.BusinessEntityID=C.PersonID
    GROUP BY ROLLUP (YEAR(SOH.DueDate), P.FirstName + ' ' + P.LastName)
    ORDER BY 1;
```

	Klient	Rok	Kwota
1		2012	35058722,6007
2		2013	48181124,1984
3		2011	13903981,3182
4		2014	26072957,9986
5			123216786,1159
6	A. Leonetti	2014	1586,6583
7	A. Leonetti	2013	1814,1819
8	Aaron Adams	2013	130,3458
9	Aaron Alexander	2014	77,339
10	Aaron Allen	2012	3756,989
11	Aaron Baker	2014	1934,8329
12	Aaron Bryant	2014	82,8529

Wniosek: W przypadku GROUP BY ROLLUP otrzymujemy troszkę inny wynik, ponieważ pominięte zostały rekordy gdzie klient posiada wartość a rok jest NULL'em (wynika to wprost z działania polecenia ROLLUP). W związku z tym należy dobrze przemyśleć jakie grupowania chcemy otrzymać i w jakiej kolejności należy podawać kolumny do polecenia ROLLUP. Z tego względu prościej będzie wykorzystywać GROUPING SETS.

Zad 1.2

```
SELECT PC.Name "Kategoria", P.Name "Produkt", ISNULL(STR(YEAR(SOH.DueDate)),
'') "Rok",
    SUM(SOD.UnitPrice * SOD.UnitPriceDiscount * SOD.OrderQty) "Kwota"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID=SOD.SalesOrderID
    JOIN Production.Product P ON P.ProductID=SOD.ProductID
    JOIN Production.ProductSubcategory PSC ON
PSC.ProductSubcategoryID=P.ProductSubcategoryID
    JOIN Production.ProductCategory PC ON
PC.ProductCategoryID=PSC.ProductCategoryID
    GROUP BY GROUPING SETS (
        (PC.Name, P.Name, YEAR(SOH.DueDate)),
        (PC.Name, P.Name)
    ) ORDER BY 1, 2;
```

	Kategoria	Produkt	Rok	Kwota
1	Accessories	All-Purpose Bike Stand	2013	0,00
2	Accessories	All-Purpose Bike Stand	2014	0,00
3	Accessories	All-Purpose Bike Stand		0,00
4	Accessories	Bike Wash - Dissolver	2013	83,8663
5	Accessories	Bike Wash - Dissolver	2014	27,7164
6	Accessories	Bike Wash - Dissolver		111,5827
7	Accessories	Cable Lock	2012	20,30
8	Accessories	Cable Lock	2013	3,48
9	Accessories	Cable Lock		23,78
10	Accessories	Fender Set - Mountain	2013	0,00
11	Accessories	Fender Set - Mountain	2014	0,00
12	Accessories	Fender Set - Mountain		0,00

Wniosek: Tylko wykorzystanie GROUPING SETS pozwoli nam pogrupować po 3 kolumnach oraz 2 kolumnach (z pominięciem pojedynczych) co potwierdza elastyczność tego polecenia i prostotę jego wykorzystania.

Zad 2.1a

```
SELECT ISNULL("Name", '') "Kategoria", ISNULL("Rok", '') "Rok",
ISNULL("Procent", 100) "Procent" FROM (
SELECT PC.Name, STR(YEAR(SOH.OrderDate)) "Rok",
        SUM(SOD.UnitPrice * SOD.OrderQty * (1 - SOD.UnitPriceDiscount)) OVER
(PARTITION BY YEAR(SOH.OrderDate))
            / SUM(SOD.UnitPrice * SOD.OrderQty * (1 - SOD.UnitPriceDiscount))
OVER()*100 "Procent"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID=SOD.SalesOrderID
    JOIN Production.Product P ON P.ProductID=SOD.ProductID
    JOIN Production.ProductSubcategory PSC ON
P.ProductSubcategoryID=PSC.ProductSubcategoryID
    JOIN Production.ProductCategory PC ON
PC.ProductCategoryID=PSC.ProductCategoryID
    WHERE PC.Name='Bikes') INSIDE
    GROUP BY GROUPING SETS (
        ("Name", "Rok", "Procent"),
        ()
    );
```

	Name	Rok	Procent
1	Bikes	2013	38,31
2	Bikes	2011	12,62
3	Bikes	2012	30,62
4	Bikes	2014	18,43
5			100,00

Wniosek: Wykorzystując funkcje okienkowe możemy w bardzo łatwy sposób wyznaczyć sumy czy procentowy udział w całości. W tym wypadku sumę sprzedaży ze względu na rok dzielimy na ogólną sumę sprzedaży. Następnie grupujemy „na zewnątrz” rekordy by utworzyć proste i czytelne zestawienie.

Zad 2.1b

```
SELECT ISNULL("Name", '') "Name", ISNULL("Rok", '') "Rok", ISNULL("Procent",
100) "Procent" FROM (
SELECT PC.Name, STR(YEAR(SOH.OrderDate)) "Rok",
        SUM(SOD.UnitPrice * SOD.OrderQty * (1 - SOD.UnitPriceDiscount)) OVER
(PARTITION BY YEAR(SOH.OrderDate))
            / SUM(SOD.UnitPrice * SOD.OrderQty * (1 - SOD.UnitPriceDiscount))
OVER()*100 "Procent"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID=SOD.SalesOrderID
    JOIN Production.Product P ON P.ProductID=SOD.ProductID
    JOIN Production.ProductSubcategory PSC ON
P.ProductSubcategoryID=PSC.ProductSubcategoryID
    JOIN Production.ProductCategory PC ON
PC.ProductCategoryID=PSC.ProductCategoryID
    WHERE PC.Name='Accessories') INSIDE
GROUP BY GROUPING SETS (
    ("Name", "Rok", "Procent"),
    ()
);
```

	Name	Rok	Procent
1	Accessories	2013	53,06
2	Accessories	2012	8,05
3	Accessories	2011	1,63
4	Accessories	2014	37,24
5			100,00

Wniosek: Analogicznie do podpunktu 2.1a

Zad 2.1c

```
SELECT ISNULL("Name", '') "Name", ISNULL("Rok", '') "Rok", ISNULL("Procent",
100) "Procent" FROM (
SELECT PC.Name, STR(YEAR(SOH.OrderDate)) "Rok",
        SUM(SOD.UnitPrice * SOD.OrderQty * (1 - SOD.UnitPriceDiscount)) OVER
(PARTITION BY YEAR(SOH.OrderDate))
            / SUM(SOD.UnitPrice * SOD.OrderQty * (1 - SOD.UnitPriceDiscount))
OVER()*100 "Procent"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID=SOD.SalesOrderID
    JOIN Production.Product P ON P.ProductID=SOD.ProductID
    JOIN Production.ProductSubcategory PSC ON
P.ProductSubcategoryID=PSC.ProductSubcategoryID
    JOIN Production.ProductCategory PC ON
PC.ProductCategoryID=PSC.ProductCategoryID
    WHERE PC.Name='Clothing') INSIDE
GROUP BY GROUPING SETS (
    ("Name", "Rok", "Procent"),
    ()
);
```

	Name	Rok	Procent
1	Clothing	2013	50,34
2	Clothing	2012	26,20
3	Clothing	2014	21,75
4	Clothing	2011	1,69
5			100,00

Wniosek: Analogicznie do podpunktu 2.1a

Zad 2.1d

```
SELECT ISNULL("Name", '') "Name", ISNULL("Rok", '') "Rok", ISNULL("Procent",
100) "Procent" FROM (
SELECT PC.Name, STR(YEAR(SOH.OrderDate)) "Rok",
        SUM(SOD.UnitPrice * SOD.OrderQty * (1 - SOD.UnitPriceDiscount)) OVER
(PARTITION BY YEAR(SOH.OrderDate))
            / SUM(SOD.UnitPrice * SOD.OrderQty * (1 - SOD.UnitPriceDiscount))
OVER()*100 "Procent"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID=SOD.SalesOrderID
    JOIN Production.Product P ON P.ProductID=SOD.ProductID
    JOIN Production.ProductSubcategory PSC ON
P.ProductSubcategoryID=PSC.ProductSubcategoryID
    JOIN Production.ProductCategory PC ON
PC.ProductCategoryID=PSC.ProductCategoryID
    WHERE PC.Name='Components') INSIDE
GROUP BY GROUPING SETS (
    ("Name", "Rok", "Procent"),
    ()
);
```

	Name	Rok	Procent
1	Components	2014	14,14
2	Components	2012	32,88
3	Components	2013	47,55
4	Components	2011	5,41
5			100,00

Wniosek: Analogicznie do podpunktu 2.1a

Zad 2.2

```
SELECT * FROM (
SELECT "Klient", "Rok", MAX("Suma transakcji") "Suma transakcji", RANK()
OVER(ORDER BY "Suma klienta" DESC) "Ranga"
FROM (
    SELECT P.FirstName + ' ' + P.LastName "Klient", YEAR(SOH.DueDate) "Rok",
        COUNT(SOH.SalesOrderID) OVER(PARTITION BY SOH.CustomerID,
YEAR(SOH.DueDate)
            ORDER BY SOH.CustomerID, YEAR(SOH.DueDate) ROWS UNBOUNDED
PRECEDING) "Suma transakcji",
        COUNT(SOH.SalesOrderID) OVER(PARTITION BY SOH.CustomerID) "Suma
klienta"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.Customer C ON C.CustomerID=SOH.CustomerID
    JOIN Person.Person P ON P.BusinessEntityID=C.PersonID) INSIDE
    GROUP BY "Klient", "Rok", "Suma klienta") OUTSIDE
    WHERE "Ranga" <= 10
    ORDER BY 4,1,2;
```

	Klient	Rok	Suma transakcji	Ranga
1	Dalton Perez	2013	14	1
2	Dalton Perez	2014	14	1
3	Mason Roberts	2013	14	1
4	Mason Roberts	2014	14	1
5	Ashley Henderson	2013	7	5
6	Ashley Henderson	2014	20	5
7	Charles Jackson	2013	14	5
8	Charles Jackson	2014	13	5
9	Daniel Davis	2013	11	5
10	Daniel Davis	2014	16	5
11	Fernando Barnes	2013	10	5
12	Fernando Barnes	2014	17	5
13	Hailey Patterson	2013	13	5

Wniosek: Z wykorzystaniem funkcji okienkowych możemy ustalić sumę transakcji każdego klienta w danym roku, oraz policzyć narastającą sumę transakcji (ze wszystkich lat). Następnie używamy funkcji szeregującej RANK() ze względu na tę narastającą sumę by ustalić top 10 klientów ze względu na ogólną suma przeprowadzonych transakcji. Można oczywiście się zastanawiać czy należy użyć RANK(), DENSE_RANK() czy też po prostu TOP 10 wierszy, to już zależne od interpretacji rankingu.

Zad 2.3

```
SELECT "Imię i nazwisko", "Rok", "Miesiąc", "W miesiącu", "W roku",
    MAX("W roku narastająco") "W roku narastająco" ,
    ISNULL(LAG("W miesiącu") OVER (ORDER BY "Imię i nazwisko", "Rok",
"Miesiąc"), 0) + "W miesiącu" "Obecny i poprzedni miesiąc"
    FROM (SELECT P.FirstName + ' ' + P.LastName "Imię i nazwisko",
        YEAR(SOH.OrderDate) "Rok", MONTH(SOH.OrderDate) "Miesiąc",
        COUNT(SOH.SalesOrderID) OVER (PARTITION BY SOH.SalesPersonID,
YEAR(SOH.OrderDate), MONTH(SOH.OrderDate)) "W miesiącu",
        COUNT(SOH.SalesOrderID) OVER (PARTITION BY SOH.SalesPersonID,
YEAR(SOH.OrderDate)) "W roku",
        COUNT(SOH.SalesOrderID)
    OVER (PARTITION BY SOH.SalesPersonID, YEAR(SOH.OrderDate) ORDER BY
SOH.SalesPersonID, YEAR(SOH.OrderDate) ROWS BETWEEN UNBOUNDED PRECEDING AND
CURRENT ROW) "W roku narastająco"
    FROM Sales.SalesOrderHeader SOH JOIN Sales.SalesPerson SP ON
SP.BusinessEntityID = SOH.SalesPersonID
    JOIN Person.Person P ON P.BusinessEntityID = SP.BusinessEntityID) INSIDE
GROUP BY "Imię i nazwisko", "Rok", "Miesiąc", "W miesiącu", "W roku"
ORDER BY 1,2,3;
```

	Imię i nazwisko	Rok	Miesiąc	W miesiącu	W roku	W roku narastająco	Obecny i poprzedni miesiąc
1	Amy Alberts	2012	6	3	7	3	3
2	Amy Alberts	2012	9	2	7	5	5
3	Amy Alberts	2012	12	2	7	7	4
4	Amy Alberts	2013	1	1	29	1	3
5	Amy Alberts	2013	2	1	29	2	2
6	Amy Alberts	2013	3	1	29	3	2
7	Amy Alberts	2013	4	2	29	5	3
8	Amy Alberts	2013	5	1	29	6	3
9	Amy Alberts	2013	6	5	29	11	6
10	Amy Alberts	2013	7	3	29	14	8
11	Amy Alberts	2013	8	1	29	15	4
12	Amy Alberts	2013	9	4	29	19	5
13	Amy Alberts	2013	10	4	29	23	8

Wniosek: Znowu funkcje okienkowe przydają się do wyznaczenia narastającej sumy w roku. Dodatkowo wykorzystana została funkcja LAG(), która pozwala w bardzo prosty sposób uzyskać sumę obsłużonych zamówień w obecnym i poprzednim miesiącu. Ogólnie funkcje okienkowe łatwiej jest rozgraniczyć od grupowania tj. przeprowadzać wszystkie operacje na funkcjach okienkowych wewnątrz podzapytania, a następnie „na zewnątrz” grupować wyniki.

Zad 2.4

```
SELECT "Kategoria", SUM("Kwota") "Suma maksymalnych" FROM (
SELECT DISTINCT PC.Name "Kategoria", MAX(P.ListPrice) OVER(PARTITION BY
PSC.ProductSubcategoryID) "Kwota"
    FROM Production.ProductCategory PC
    JOIN Production.ProductSubcategory PSC ON
PSC.ProductCategoryID=PC.ProductCategoryID
    JOIN Production.Product P ON
P.ProductSubcategoryID=PSC.ProductSubcategoryID) INSIDE
    GROUP BY "Kategoria";
```

	Kategoria	Suma maksymalnych
1	Accessories	663,88
2	Bikes	9362,33
3	Clothing	408,94
4	Components	5539,77

Wniosek: Najpierw w podzapytaniu zestawiamy kategorie z maksymalnymi cenami produktów ze względu na podkategorię, a dopiero na zewnątrz sumujemy wszystkie te maksymalne ceny i grupujemy po samej kategorii. DISTINCT w podzapytaniu jest wymagany, ponieważ dla każdej kategorii i powiązanej z nią podkategorią pojawi się wiele rekordów o tej samej wartości MAX(P.ListPrice) (ponieważ nie grupujemy po nazwie kategorii).

Zad 2.5a

```
SELECT RANK() OVER(ORDER BY COUNT(SOD.OrderQty) DESC) "Ranga",
    P.FirstName + ' ' + P.LastName "Imię i nazwisko", COUNT(SOD.OrderQty)
"Liczba transakcji"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID=SOD.SalesOrderID
    JOIN Sales.Customer C ON C.CustomerID=SOH.CustomerID
    JOIN Person.Person P ON P.BusinessEntityID=C.PersonID
    GROUP BY P.FirstName + ' ' + P.LastName;
```

	Ranga	Imię i nazwisko	Liczba transakcji
1	1	Reuben D'sa	530
2	2	Richard Lum	482
3	3	Ryan Calafato	451
4	4	Yale Li	446
5	5	Marcia Sultan	441
6	6	Holly Dickson	440
7	7	Robert Vessa	436
8	7	Della Demott Jr	436
9	9	Sandra Maynard	432
10	10	Joseph Castellucio	429
11	11	Blaine Dockter	422
12	12	John Evans	418

Wniosek: Błędne nazewnictwo, zamiast ‘liczba transakcji’ precyzyjnie powinno być ‘liczba zakupionych produktów. W tym wypadku widzimy przeskok z rangi 7 na 9, ponieważ 2 klientów posiada tę samą rangę 7.

Zad 2.5b

```
SELECT DENSE_RANK() OVER(ORDER BY COUNT(SOD.OrderQty) DESC) "Ranga",
    P.FirstName + ' ' + P.LastName "Imię i nazwisko", COUNT(SOD.OrderQty)
"Liczba transakcji"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID=SOD.SalesOrderID
    JOIN Sales.Customer C ON C.CustomerID=SOH.CustomerID
    JOIN Person.Person P ON P.BusinessEntityID=C.PersonID
    GROUP BY P.FirstName + ' ' + P.LastName;
```

	Ranga	Imię i nazwisko	Liczba transakcji
1	1	Reuben D'sa	530
2	2	Richard Lum	482
3	3	Ryan Calafato	451
4	4	Yale Li	446
5	5	Marcia Sultan	441
6	6	Holly Dickson	440
7	7	Robert Vessa	436
8	7	Della Demott Jr	436
9	8	Sandra Maynard	432
10	9	Joseph Castellucio	429
11	10	Blaine Dockter	422
12	11	John Evans	418

Wniosek: Wykorzystując DENSE_RANK() widzimy różnicę w stosunku do poprzedniego zadania. Mianowicie następuję kontynuacja przydzielania rang (po dwóch klientach z rangą 7 występuje klient z rangą 8). Po prostu w zależności od potrzeb należy wykorzystywać albo jedno albo drugie rozwiązanie.

Zad 2.6

```
SELECT P.Name "Nazwa produktu", AVG(SOD.OrderQty) "Średnia liczba sztuk",
    CASE NTILE(3) OVER(ORDER BY AVG(SOD.OrderQty) DESC)
        WHEN 1 THEN 'Najlepiej'
        WHEN 2 THEN 'Średnio'
        WHEN 3 THEN 'Najsłabiej'
    END "Ranga sprzedaży"
    FROM Sales.SalesOrderHeader SOH
    JOIN Sales.SalesOrderDetail SOD ON SOD.SalesOrderID=SOH.SalesOrderID
    JOIN Production.Product P ON P.ProductID=SOD.ProductID
    GROUP BY P.Name
    ORDER BY 2 DESC;
```

82	Rear Brakes	2	Najlepiej
83	Rear Derailleur	2	Najlepiej
84	Mountain-400-W Silver, 40	2	Najlepiej
85	Mountain-200 Silver, 38	2	Najlepiej
86	Mountain-200 Silver, 42	2	Najlepiej
87	Mountain-200 Silver, 46	2	Najlepiej
88	Mountain-500 Silver, 40	2	Najlepiej
89	Mountain-500 Silver, 42	2	Najlepiej
90	Road-250 Black, 44	2	Srednio
91	Road-250 Black, 48	2	Srednio
92	Road-250 Red, 44	2	Srednio
93	Road-250 Red, 48	2	Srednio

Wniosek: Dzięki NTILE() możemy podzielić nasze rekordy na niemal równoliczne grupy. W tym wypadku widać, że niezależnie od średniej zakupionych produktów zasada równoliczności jest trzymana i w ten sposób jedne produkty „łapią się” do kubełka najlepiej sprzedających, a inne do średnio sprzedających.

Zadanie dodatkowe:

```
SELECT "Kolor", "Miesiąc", "Dzień miesiąca", MAX("Liczba transakcji") "Liczba
transakcji",
    MAX("narast") "Liczba transakcji narastająco"
FROM (SELECT P.Color "Kolor", MONTH(SOH.OrderDate) "Miesiąc",
DAY(SOH.OrderDate) "Dzień miesiąca",
        COUNT(SOD.SalesOrderID) OVER(PARTITION BY P.Color,
MONTH(SOH.OrderDate), DAY(SOH.OrderDate)) "Liczba transakcji",
        COUNT(SOD.SalesOrderID) OVER(PARTITION BY P.Color,
MONTH(SOH.OrderDate) ORDER BY P.Color, MONTH(SOH.OrderDate),
DAY(SOH.OrderDate) ROWS UNBOUNDED PRECEDING) "narast"
        FROM Sales.SalesOrderHeader SOH
        JOIN Sales.SalesOrderDetail SOD ON SOD.SalesOrderID=SOH.SalesOrderID
        JOIN Production.Product P ON P.ProductID=SOD.ProductID
        WHERE P.Color IS NOT NULL) INSIDE
GROUP BY "Kolor", "Miesiąc", "Dzień miesiąca"
ORDER BY 1,2,3;
```

	Kolor	Miesiąc	Dzień miesiąca	Liczba transakcji	Liczba transakcji narastająco
1	Black	1	1	287	287
2	Black	1	2	34	321
3	Black	1	3	26	347
4	Black	1	4	27	374
5	Black	1	5	21	395
6	Black	1	6	31	426
7	Black	1	7	34	460
8	Black	1	8	25	485
9	Black	1	9	28	513
10	Black	1	10	22	535
11	Black	1	11	28	563
12	Black	1	12	27	590
13	Black	1	13	29	619

Wniosek: Znowu następuje rozgraniczenie między funkcjami okienkowymi a agregującymi. W podzapytaniu tworzona jest narastająca suma liczb transakcji ze względu na miesiąc, a następnie na zewnątrz dochodzi do grupowania po kolorze, miesiącu i dniu.

Wnioski:

- Funkcje grupujące pozwalają nam w bardzo precyzyjny sposób dobrać dane, po których chcemy pogrupować wynik. Korzystając z GROUPING SETS możemy np. określić by w wyniku znalazły się rekordy pogrupowane po wszystkich kolumnach, tylko po jednej czy dwóch wybranych kolumnach.
- Funkcje okienkowe pozwalają nam zaimplementować w bardzo prosty i czytelny sposób pewne narastające wartości wśród danych grup jak i pewne rankingi, które możemy przypisać rekordom. Należy jednak rozróżniać funkcje okienkowe od agregujących, ponieważ ich wpływ na strukturę ostatecznego wyniku jest zgoła odmienny.