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)),

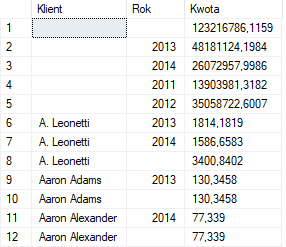
        (P.FirstName + ' ' + P.LastName, YEAR(SOH.DueDate)),

        (P.FirstName + ' ' + P.LastName),

        ()

    )

    ORDER BY 1;



**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"

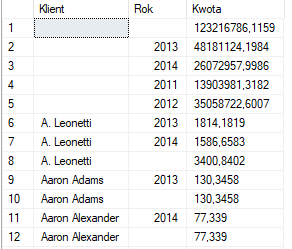
    FROM Sales.SalesOrderHeader SOH

    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;

****

**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"

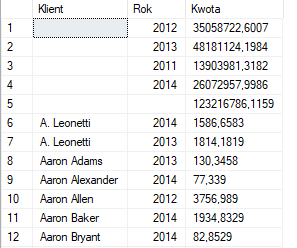
    FROM Sales.SalesOrderHeader SOH

    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;

****

**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;

****

**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

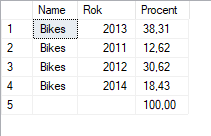
    WHERE PC.Name='Bikes') INSIDE

    GROUP BY GROUPING SETS (

        ("Name", "Rok", "Procent"),

        ()

    );

****

**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"),

    ()

);

****

**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"),

    ()

);

****

**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

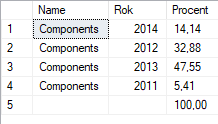
    WHERE PC.Name='Components') INSIDE

GROUP BY GROUPING SETS (

    ("Name", "Rok", "Procent"),

    ()

);

****

**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;

****

**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)

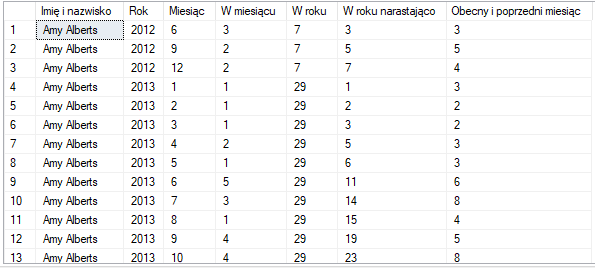
    OVER (PARTITION BY SOH.SalesPersonID, YEAR(SOH.OrderDate) ORDER BY SOH.SalesPersonID, YEAR(SOH.OrderDate) ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) "W roku narastająco"

    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;

****

**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 (

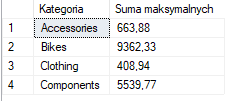
SELECT DISTINCT PC.Name "Kategoria", MAX(P.ListPrice) OVER(PARTITION BY PSC.ProductSubcategoryID) "Kwota"

    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";

****

**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"

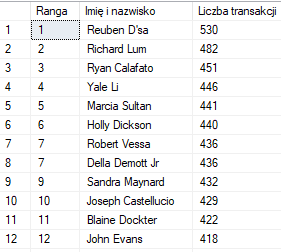
    FROM Sales.SalesOrderHeader SOH

    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;

****

**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"

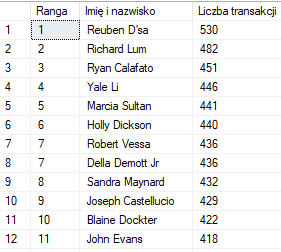
    FROM Sales.SalesOrderHeader SOH

    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;

****

**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"

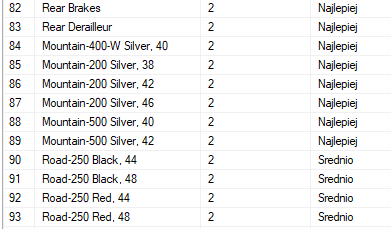
    FROM Sales.SalesOrderHeader SOH

    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;

****

**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;

****

**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.