HURTOWNIE DANYCH

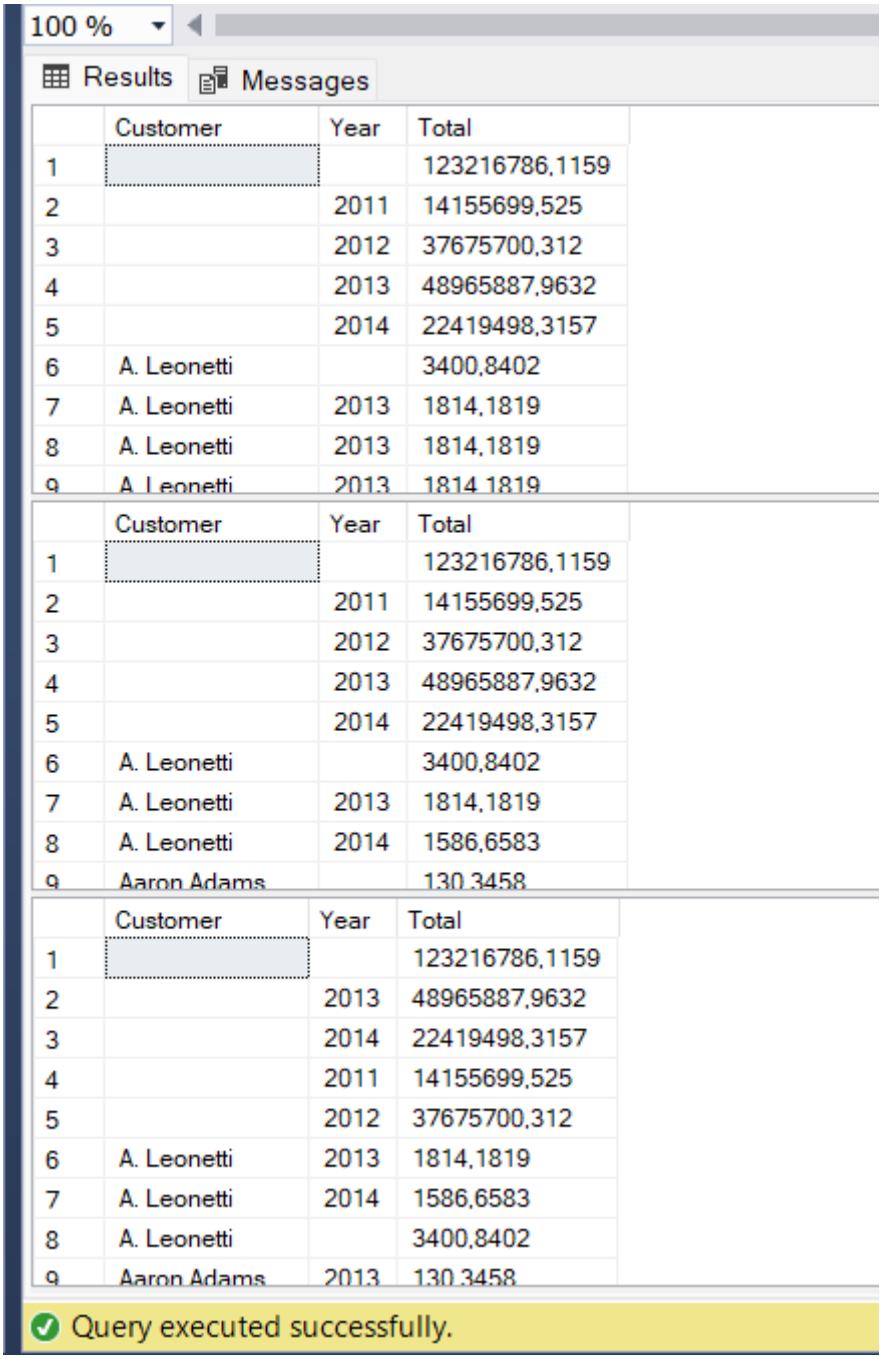
Lista 3 – Podstawy SQL: funkcje grupujące i okienkowe

ALEKSANDER STEPANIUK
NR. INDEKSU: 272644
Politechnika Wrocławska, Informatyka Stosowana

Rozwiązania:

Zadanie 1.

1. Kolejno: Rollup, Cube, Grouping sets



100 %

Results | Messages

	Customer	Year	Total
1			123216786,1159
2		2011	14155699,525
3		2012	37675700,312
4		2013	48965887,9632
5		2014	22419498,3157
6	A. Leonetti		3400,8402
7	A. Leonetti	2013	1814,1819
8	A. Leonetti	2013	1814,1819
9	A. Leonetti	2013	1814,1819

	Customer	Year	Total
1			123216786,1159
2		2011	14155699,525
3		2012	37675700,312
4		2013	48965887,9632
5		2014	22419498,3157
6	A. Leonetti		3400,8402
7	A. Leonetti	2013	1814,1819
8	A. Leonetti	2014	1586,6583
9	Aaron Adams		130,3458

	Customer	Year	Total
1			123216786,1159
2		2013	48965887,9632
3		2014	22419498,3157
4		2011	14155699,525
5		2012	37675700,312
6	A. Leonetti	2013	1814,1819
7	A. Leonetti	2014	1586,6583
8	A. Leonetti		3400,8402
9	Aaron Adams	2013	130,3458

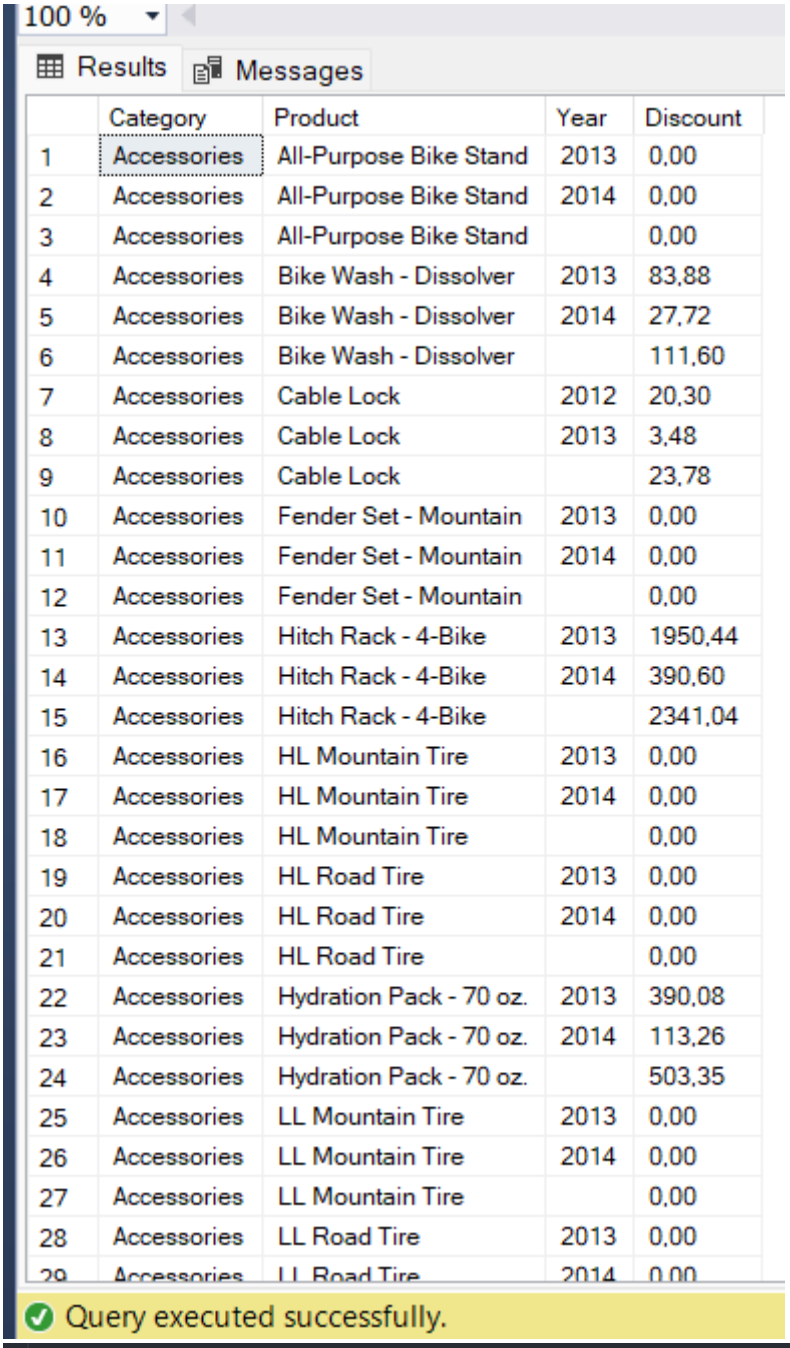
Query executed successfully.

```sql
--rollup
SELECT
    COALESCE(CONCAT(FirstName, ' ', LastName), '') as "Customer",
    COALESCE(CAST(DATEPART(YEAR, OrderDate) AS VARCHAR), '') as "Year",
    SUM(TotalDue) as "Total"
FROM Sales.SalesOrderHeader SOH
    JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID
    JOIN Person.Person P ON C.PersonID = P.BusinessEntityID
GROUP BY ROLLUP((FirstName, LastName), YEAR(OrderDate)),
         ROLLUP((YEAR(OrderDate)))
ORDER BY "Customer", "Year";

--cube
SELECT
    COALESCE(CONCAT(FirstName, ' ', LastName), '') as "Customer",
    COALESCE(CAST(DATEPART(YEAR, OrderDate) AS VARCHAR), '') as "Year",
    SUM(TotalDue) as "Total"
FROM Sales.SalesOrderHeader SOH
    JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID
    JOIN Person.Person P ON C.PersonID = P.BusinessEntityID
GROUP BY CUBE((FirstName, LastName), YEAR(OrderDate))
ORDER BY "Customer", "Year";

--grouping sets
SELECT
    COALESCE(CONCAT(FirstName, ' ', LastName), '') as "Customer",
    COALESCE(CAST(DATEPART(YEAR, OrderDate) AS VARCHAR), '') as "Year",
    SUM(TotalDue) as "Total"
FROM Sales.SalesOrderHeader SOH
         JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID
         JOIN Person.Person P ON C.PersonID = P.BusinessEntityID
GROUP BY GROUPING SETS ((Year(OrderDate), CONCAT(FirstName, ' ', LastName)),
                        (Year(OrderDate)), (CONCAT(FirstName, ' ', LastName)), ())
ORDER BY 1, 2;
```

2. *Przygotować zestawienie przedstawiające łączną kwotę zniżek z podziałem na kategorie…*



100 %

Results | Messages

	Category	Product	Year	Discount
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	83,88
5	Accessories	Bike Wash - Dissolver	2014	27,72
6	Accessories	Bike Wash - Dissolver		111,60
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
13	Accessories	Hitch Rack - 4-Bike	2013	1950,44
14	Accessories	Hitch Rack - 4-Bike	2014	390,60
15	Accessories	Hitch Rack - 4-Bike		2341,04
16	Accessories	HL Mountain Tire	2013	0,00
17	Accessories	HL Mountain Tire	2014	0,00
18	Accessories	HL Mountain Tire		0,00
19	Accessories	HL Road Tire	2013	0,00
20	Accessories	HL Road Tire	2014	0,00
21	Accessories	HL Road Tire		0,00
22	Accessories	Hydration Pack - 70 oz.	2013	390,08
23	Accessories	Hydration Pack - 70 oz.	2014	113,26
24	Accessories	Hydration Pack - 70 oz.		503,35
25	Accessories	LL Mountain Tire	2013	0,00
26	Accessories	LL Mountain Tire	2014	0,00
27	Accessories	LL Mountain Tire		0,00
28	Accessories	LL Road Tire	2013	0,00
29	Accessories	LL Road Tire	2014	0,00

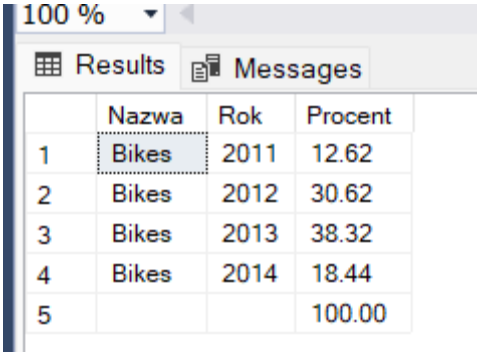
Query executed successfully.

```sql
SELECT
    COALESCE(PC.Name, '') as "Category",
    COALESCE(P.Name, '') as "Product",
    COALESCE(CAST(DATEPART(YEAR, OrderDate) AS VARCHAR), '') as "Year",
    ROUND(SUM(SOD.OrderQty * SOD.UnitPrice * SOD.UnitPriceDiscount), 2) as "Discount"
FROM Sales.SalesOrderDetail SOD
        JOIN Production.Product P ON SOD.ProductID = P.ProductID
        JOIN Production.ProductSubcategory PSC ON P.ProductSubcategoryID = PSC.ProductSubcategoryID
        JOIN Production.ProductCategory PC ON PSC.ProductCategoryID = PC.ProductCategoryID
        JOIN Sales.SalesOrderHeader SOH ON SOD.SalesOrderID = SOH.SalesOrderID
GROUP BY GROUPING SETS ((PC.Name, P.Name, YEAR(OrderDate)), (PC.Name, P.Name), (PC.Name), ())
```

Zad2.

1. *Dla kategorii ‘Bikes’ przygotuj zestawienie prezentujące procentowy udział kwot…*

a) Bikes

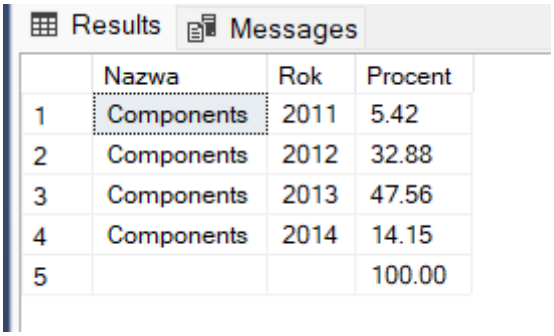


100 %

Results | Messages

	Nazwa	Rok	Procent
1	Bikes	2011	12.62
2	Bikes	2012	30.62
3	Bikes	2013	38.32
4	Bikes	2014	18.44
5			100.00

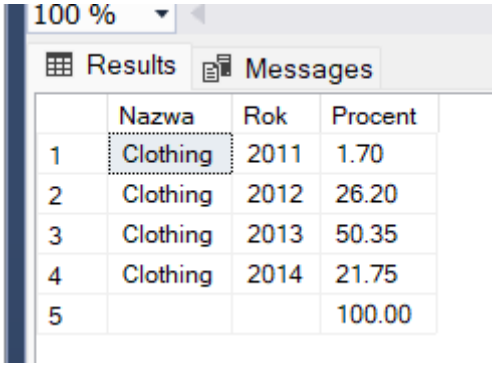
b) Components



Results | Messages

	Nazwa	Rok	Procent
1	Components	2011	5.42
2	Components	2012	32.88
3	Components	2013	47.56
4	Components	2014	14.15
5			100.00

c) Clothing

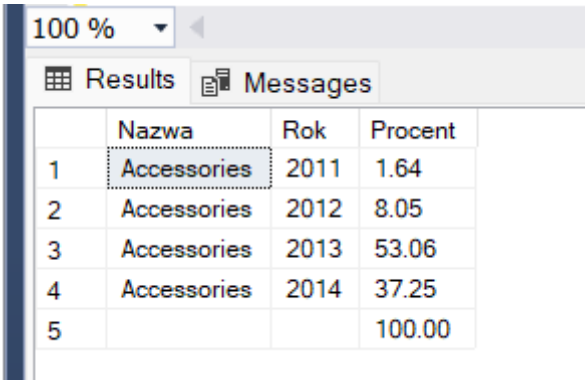


100 %

Results | Messages

	Nazwa	Rok	Procent
1	Clothing	2011	1.70
2	Clothing	2012	26.20
3	Clothing	2013	50.35
4	Clothing	2014	21.75
5			100.00

d) Accessories



100 %

Results | Messages

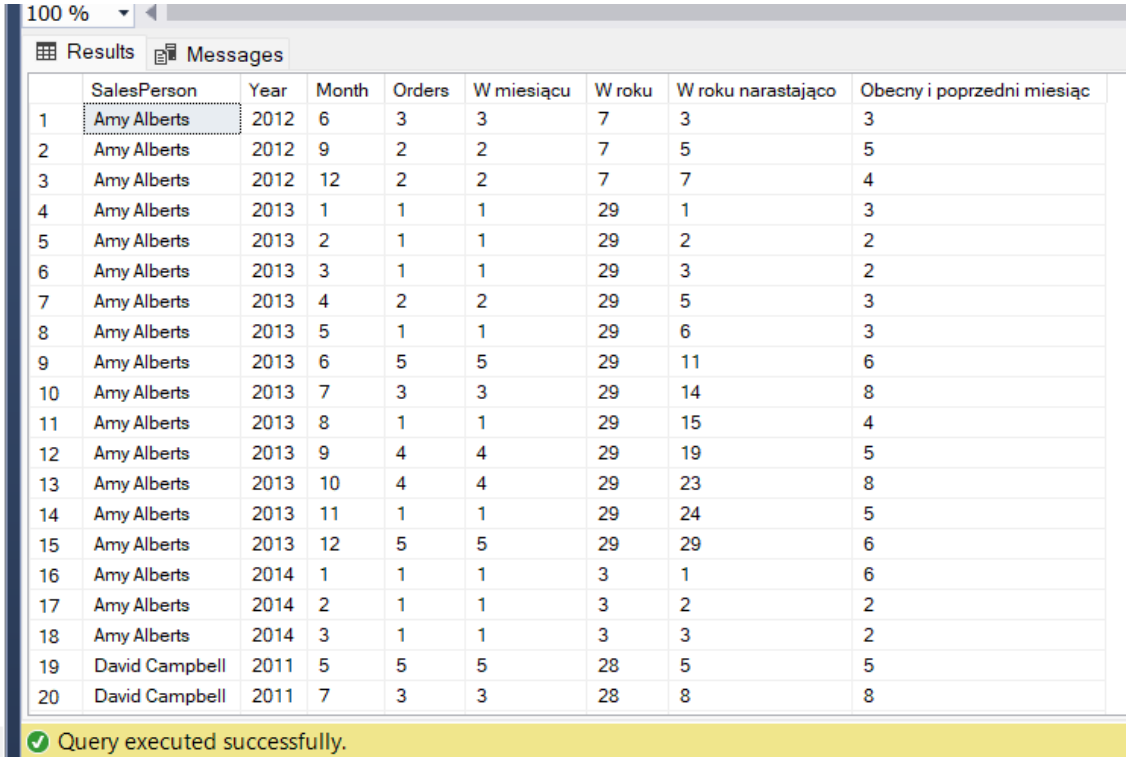
	Nazwa	Rok	Procent
1	Accessories	2011	1.64
2	Accessories	2012	8.05
3	Accessories	2013	53.06
4	Accessories	2014	37.25
5			100.00

```sql
WITH BikesSales AS (
    SELECT
        YEAR(SOH.OrderDate) AS Rok,
        SUM(SOD.LineTotal) AS SalesAmount
    FROM Sales.SalesOrderHeader SOH
        JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID = SOD.SalesOrderID
        JOIN Production.Product P ON SOD.ProductID = P.ProductID
        JOIN Production.ProductSubcategory PS ON P.ProductSubcategoryID = PS.ProductSubcategoryID
        JOIN Production.ProductCategory PC ON PS.ProductCategoryID = PC.ProductCategoryID
    WHERE PC.Name = 'Bikes'
    GROUP BY YEAR(SOH.OrderDate)
)
SELECT Nazwa, Rok, Procent
FROM (
     SELECT
         'Bikes' AS Nazwa,
         CAST(Rok AS VARCHAR(4)) AS Rok,
         CAST(ROUND(100.0 * SalesAmount / SUM(SalesAmount) OVER (), 2) AS DECIMAL(5,2)) AS Procent,
         Rok AS SortOrder
     FROM BikesSales

     UNION ALL

     SELECT
         '' AS Nazwa,
         '' AS Rok,
         CAST(100.00 AS DECIMAL(5,2)) AS Procent,
         9999 AS SortOrder
) AS T
ORDER BY SortOrder;
```

2. *Przygotuj zestawienie dla sprzedawców z podziałem na lata i miesiące prezentujące liczbę obsłużonych przez nich zamówień w ciągu roku, narastająco, sumarycznie w poprzecznim i obecnym miesiącu. (Wykorzystaj funkcje okna)*



100 %

Results | Messages

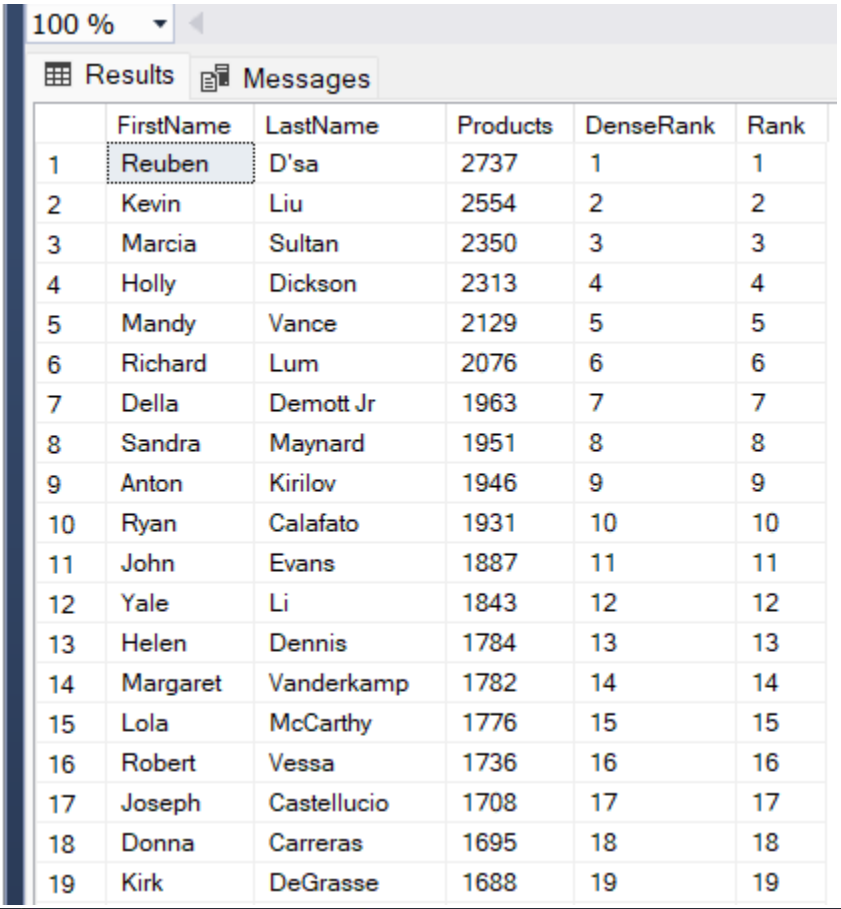
	SalesPerson	Year	Month	Orders	W miesiącu	W roku	W roku narastająco	Obecny i poprzedni miesiąc
1	Amy Alberts	2012	6	3	3	7	3	3
2	Amy Alberts	2012	9	2	2	7	5	5
3	Amy Alberts	2012	12	2	2	7	7	4
4	Amy Alberts	2013	1	1	1	29	1	3
5	Amy Alberts	2013	2	1	1	29	2	2
6	Amy Alberts	2013	3	1	1	29	3	2
7	Amy Alberts	2013	4	2	2	29	5	3
8	Amy Alberts	2013	5	1	1	29	6	3
9	Amy Alberts	2013	6	5	5	29	11	6
10	Amy Alberts	2013	7	3	3	29	14	8
11	Amy Alberts	2013	8	1	1	29	15	4
12	Amy Alberts	2013	9	4	4	29	19	5
13	Amy Alberts	2013	10	4	4	29	23	8
14	Amy Alberts	2013	11	1	1	29	24	5
15	Amy Alberts	2013	12	5	5	29	29	6
16	Amy Alberts	2014	1	1	1	3	1	6
17	Amy Alberts	2014	2	1	1	3	2	2
18	Amy Alberts	2014	3	1	1	3	3	2
19	David Campbell	2011	5	5	5	28	5	5
20	David Campbell	2011	7	3	3	28	8	8

Query executed successfully.

```sql
WITH SalesPerSalesPerson AS (
    SELECT
        P.FirstName + ' ' + P.LastName AS SalesPerson,
        YEAR(SOH.OrderDate) AS Year,
        MONTH(SOH.OrderDate) AS Month,
        COUNT(SOH.SalesOrderID) AS Orders
    FROM Sales.SalesOrderHeader SOH
        JOIN Sales.SalesPerson SP ON SOH.SalesPersonID = SP.BusinessEntityID
        JOIN Person.Person P ON SP.BusinessEntityID = P.BusinessEntityID
    GROUP BY P.FirstName, P.LastName, YEAR(SOH.OrderDate), MONTH(SOH.OrderDate)
)

SELECT
    SalesPerson,
    Year,
    Month,
    Orders,
    SUM(Orders) OVER (PARTITION BY SalesPerson, Year, Month) AS "W miesiącu",
    SUM(Orders) OVER (PARTITION BY SalesPerson, Year) AS "W roku",
    SUM(Orders) OVER (PARTITION BY SalesPerson, Year ORDER BY Month ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS "W roku narastająco",
    SUM(Orders) OVER (PARTITION BY SalesPerson ORDER BY Year, Month ROWS BETWEEN 1 PRECEDING AND CURRENT ROW) AS "Obecny i poprzedni miesiąc"
FROM SalesPerSalesPerson
ORDER BY SalesPerson, Year, Month;
```

3. *Przygotuj ranking klientów w zależności od liczby zakupionych produktów. Porównaj rozwiązania uzyskane przez funkcje rank i dense_rank…*



100 %

Results | Messages

	FirstName	LastName	Products	DenseRank	Rank
1	Reuben	D'sa	2737	1	1
2	Kevin	Liu	2554	2	2
3	Marcia	Sultan	2350	3	3
4	Holly	Dickson	2313	4	4
5	Mandy	Vance	2129	5	5
6	Richard	Lum	2076	6	6
7	Della	Demott Jr	1963	7	7
8	Sandra	Maynard	1951	8	8
9	Anton	Kirilov	1946	9	9
10	Ryan	Calafato	1931	10	10
11	John	Evans	1887	11	11
12	Yale	Li	1843	12	12
13	Helen	Dennis	1784	13	13
14	Margaret	Vanderkamp	1782	14	14
15	Lola	McCarthy	1776	15	15
16	Robert	Vessa	1736	16	16
17	Joseph	Castellucio	1708	17	17
18	Donna	Carreras	1695	18	18
19	Kirk	DeGrasse	1688	19	19

```sql
WITH CustomersProducts AS (
    SELECT
        P.FirstName,
        P.LastName,
        SUM(SOD.OrderQty) AS Products
    FROM Sales.SalesOrderDetail SOD
        JOIN Sales.SalesOrderHeader SOH ON SOD.SalesOrderID = SOH.SalesOrderID
        JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID
        JOIN Person.Person P ON C.PersonID = P.BusinessEntityID
    GROUP BY P.FirstName, P.LastName
)

SELECT
    FirstName,
    LastName,
    Products,
    DENSE_RANK() OVER (ORDER BY Products DESC) AS DenseRank,
    RANK() OVER (ORDER BY Products DESC) AS Rank
FROM CustomersProducts
ORDER BY Products DESC;
```

4. *Przygotuj ranking produktów w zależności od średniej liczby sprzedanych sztuk. Wyróżnij 3 (prawie równoliczne)* grupy produktów…

Results | Messages

	Product	AvgSold	ranking
1	Full-Finger Gloves, L	9	najlepiej
2	Full-Finger Gloves, M	6	najlepiej
3	Classic Vest, S	6	najlepiej
4	ML Headset	5	najlepiej
5	Mountain Bike Socks, M	5	najlepiej
6	Women's Mountain Shorts, S	5	najlepiej
7	Women's Tights, L	4	najlepiej
8	Women's Mountain Shorts, L	4	najlepiej
9	Women's Tights, S	4	najlepiej
10	Racing Socks, L	4	najlepiej
11	Short-Sleeve Classic Jersey, XL	4	najlepiej
12	Men's Sports Shorts, M	4	najlepiej
13	Minipump	4	najlepiej
14	Long-Sleeve Logo Jersey, L	4	najlepiej
15	Men's Bib-Shorts, M	4	najlepiej
16	Classic Vest, M	4	najlepiej

```sql
WITH ProductsSales AS (
    SELECT
        P.Name AS Product,
        AVG(SOD.OrderQty) AS AvgSold
    FROM Sales.SalesOrderDetail SOD
        JOIN Production.Product P ON SOD.ProductID = P.ProductID
    GROUP BY P.Name
)

SELECT
    Product,
    AvgSold,
    CASE NTILE(3) OVER (ORDER BY AvgSold DESC)
    WHEN 1 THEN 'najlepiej'
        WHEN 2 THEN 'srednio'
        ELSE 'najslabiej'
END ranking
FROM ProductsSales
ORDER BY AvgSold DESC;
```

Zad3.

Ocena jakości danych (profilowanie danych) zostało wykonanie w języku Python za pomocą bibliotek pandas i ydata_profiling.

```python
import pandas as pd
from ydata_profiling import ProfileReport

data = pd.read_csv('dane_lista3.csv')

# data frame info
print("Info o DataFrame:")
print(data.info())
# statystyki opisowe
print("\nStatystyki opisowe (wszystkie kolumny):")
print(data.describe(include='all'))

# liczba wystąpień error i unknown
print("\nLiczba wystąpień 'ERROR' i 'UNKNOWN' w każdej kolumnie:")
print(data.isin(['ERROR', 'UNKNOWN']).sum())

# liczba brakujących wartości
print("\nLiczba brakujących wartości:")
print(data.isnull().sum())

profile = ProfileReport(data, title="Raport profilu danych - dane_lista3.csv", explorative=True)
profile.to_file("raport_danych.html")

print("\nraport został zapisany do pliku 'raport_danych.html'")
```

Wszystkie rekordy mają id transakcji, ale tylko blisko 2/3 posiada dane w kolumnie Location.

```
RangeIndex: 10002 entries, 0 to 10001
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Transaction ID    10002 non-null  object
 1   Item              9669 non-null   object
 2   Quantity          9864 non-null   object
 3   Price Per Unit    9823 non-null   object
 4   Total Spent       9829 non-null   object
 5   Payment Method    7423 non-null   object
 6   Location          6737 non-null   object
 7   Transaction Date  9842 non-null   object
dtypes: object(8)
memory usage: 625.3+ KB
```

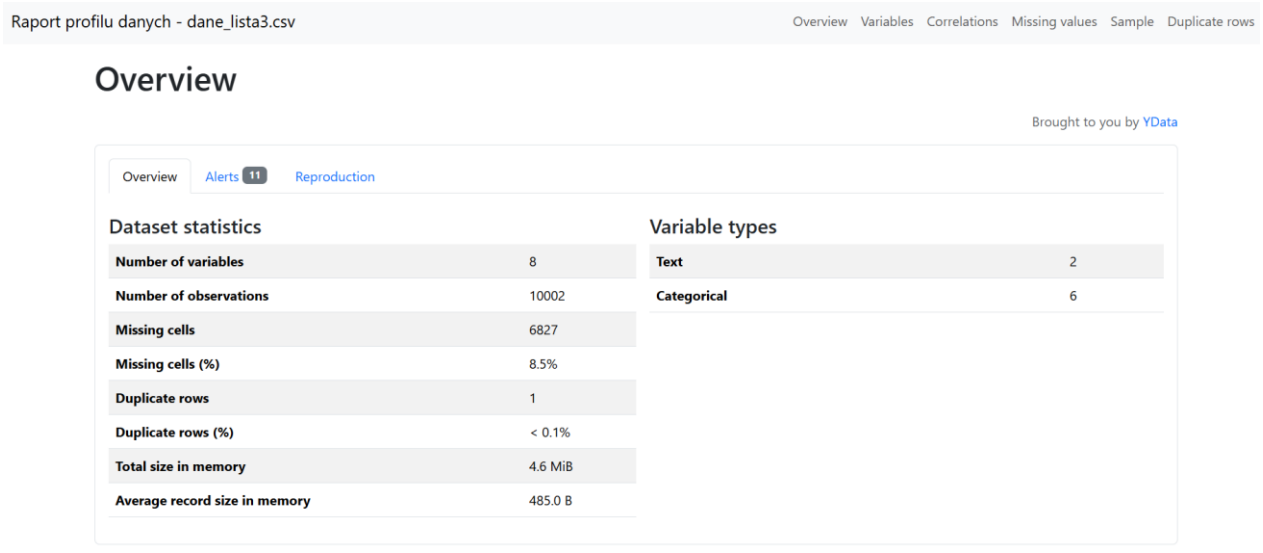
Ilość wystąpień słów ‘Error’ i ‘Unknown’ w różnych kolumnach wydaje się być zbliżona (około 300-600 rekordów na 10002 czyli jakieś 3%-6%) , tylko kolumna ID nie posiada takich rekordów.

```
Liczba wystąpień 'ERROR' i 'UNKNOWN' w każdej kolumnie:
Transaction ID        0
Item                636
Quantity            341
Price Per Unit      354
Total Spent         329
Payment Method      599
Location            696
Transaction Date    301
dtype: int64
```

Najwięcej brakujących wartości (pustych pól) widać w kolumnach metody płatności i lokalizacji. Jest ich tam o cały rząd wielkości więcej niż w pozostałych kolumnach.

```
Liczba brakujących wartości:
Transaction ID        0
Item                333
Quantity            138
Price Per Unit      179
Total Spent         173
Payment Method     2579
Location           3265
Transaction Date    160
dtype: int64
```

Wygenerowany raport za pomocą ydata_profiling:



Możemy na tej stronie html interaktywnie przeglądać nasze dane, tu kilka najważniejszych wykresów:

ID:

Overview | Words | Characters

Value	Count	Frequency (%)
txn_8859035	3	< 0.1%
txn_1108663	2	< 0.1%
txn_5255387	1	< 0.1%
txn_7619095	1	< 0.1%
txn_9437049	1	< 0.1%
txn_4717867	1	< 0.1%
txn_8866974	1	< 0.1%
txn_1736287	1	< 0.1%
txn_3142496	1	< 0.1%
txn_9226047	1	< 0.1%
Other values (9989)	9989	99.9%

Widać że ID nie jest unikalne, i niektóre pojedyncze rekordy występują więcej niż jeden raz.

Kolumna Item:

Common Values

Value	Count	Frequency (%)
Juice	1171	11.7%
Coffee	1164	11.6%
Salad	1148	11.5%
Cake	1141	11.4%
Sandwich	1131	11.3%
Smoothie	1096	11.0%
Cookie	1092	10.9%
Tea	1089	10.9%
UNKNOWN	344	3.4%
ERROR	292	2.9%
(Missing)	333	3.3%

Widać że ilość produktów jest podobna w poszczególnych kategoriach, z wyjątkiem brakujących danych, które to kategorią nie są, ale widać że jest to łącznie około 1000 rekordów (~10%), które nie mają podanej wartości w ‘Item’.

Kolumna Quantity:

Common Values

Value	Count	Frequency (%)
5	2013	20.1%
2	1973	19.7%
4	1863	18.6%
3	1851	18.5%
1	1821	18.2%
UNKNOWN	171	1.7%
ERROR	170	1.7%
-2	1	< 0.1%
100	1	< 0.1%
(Missing)	138	1.4%

Widać, że ilość oscyluje w granicach 1-5, pojedyncze wartości -2 i 100 które prawdopodobnie zostały wprowadzone do danych ręcznie lub są wynikiem błędu, więc można je kategoryzować razem z resztą błędów.

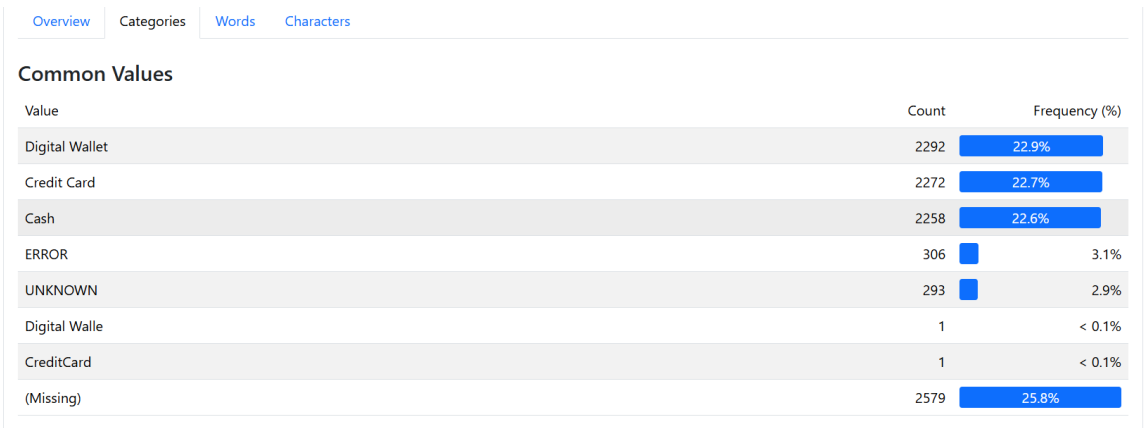
Kolumna Price per Unit:

Common Values

Value	Count	Frequency (%)
3.0	2431	24.3%
4.0	2331	23.3%
2.0	1227	12.3%
5.0	1204	12.0%
1.0	1143	11.4%
1.5	1133	11.3%
ERROR	190	1.9%
UNKNOWN	164	1.6%
(Missing)	179	1.8%

Widać że jest prawie dwa razy więcej wartości 3.0 i 4.0 niż pozostałych wartości takich jak 2.0, 5.0, 1.0, 1.5. Pozostałe wartości typu ERROR, UNKNOWN i (Missing) czyli puste pole stanowią zdecydowaną mniejszość.

Kolumna Payment method:



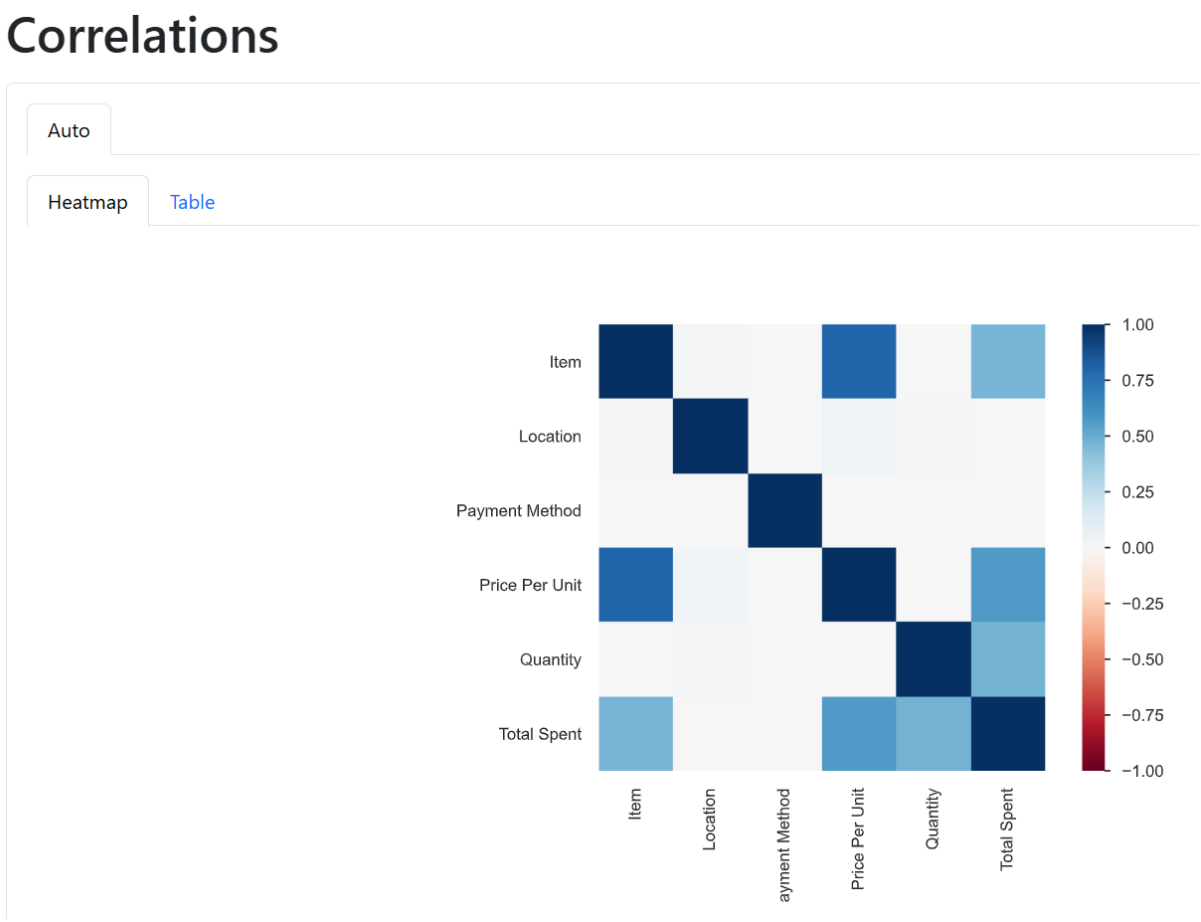
Overview | Categories | Words | Characters

Common Values

Value	Count	Frequency (%)
Digital Wallet	2292	22.9%
Credit Card	2272	22.7%
Cash	2258	22.6%
ERROR	306	3.1%
UNKNOWN	293	2.9%
Digital Walle	1	< 0.1%
CreditCard	1	< 0.1%
(Missing)	2579	25.8%

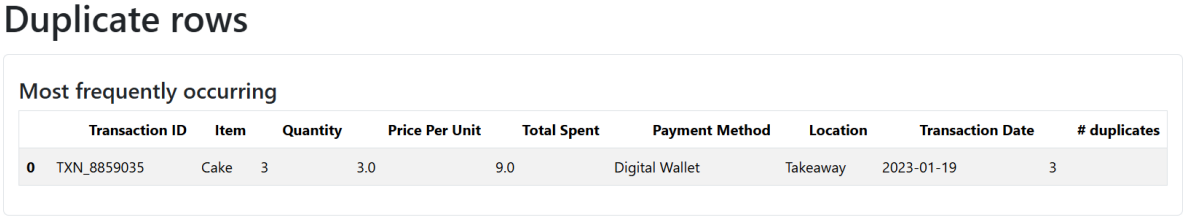
Widać że większość danych to Digital Wallet, Credit Card lub Cash, jest też standardowo trochę pustych rekordów z wartościami ERROR lub UNKNOWN ale również pojedyncze literówki w Digital Walle (brakujące ‘t’ na końcu) oraz CreditCard pisane razem. Widać też że ponad 25% rekordów jest brakujących (puste pola)

Mapa korelacji:



Widać największą korelację pomiędzy Price per unit, a Item oraz Total Spent i polami Item, Price Per Unit, Quantity. Reszta pól jest bez żadnej korelacji.

Duplikaty:



	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date	# duplicates
0	TXN_8859035	Cake	3	3.0	9.0	Digital Wallet	Takeaway	2023-01-19	3

Widać także że jeden rekord występuje aż 3 razy w dokładnie tej samej formie.

Wnioski:

Wykorzystanie funkcji grupujących (rollup, cube, grouping sets) okazało się przydatne do uzyskania dodatkowych informacji o danych, takich jak sumy całkowite, sumy częściowe czy rankingu i podsumowania na różnych poziomach szczegółów w relatywnie prosty sposób. Wyniki takiego zapytania są właściwie gotowe do użycia w raportach i analizach.

Największe zniżki dotyczą produktów z kategorii 'Bikes'. Łączna kwota zniżek dla tej kategorii wynosi ponad 0,5 miliona. Widać, że niektóre produkty nigdy nie potrzebowały zniżki, co może sugerować ich wysoką jakość lub popularność wśród klientów. Inne produkty były przeceniane tylko w niektórych latach lub miesiącach, co może być związane z sezonowością sprzedaży lub innymi czynnikami rynkowymi.

Najwięcej zniżek zostało udzielonych w 2012 i 2013 roku (około 200tys w 2012 i 300tys w 2013) - może to sugerować, że w tych latach sklep dynamicznie się rozwijał i wprowadzał nowe promocje, aby przyciągnąć klientów (pik w sprzedaży 2013 rok). We wszystkich kategoriach widać, że najwięcej promocji zostało nałożone w 2013 roku, jednak dla rowerów i komponentów 2012 rok jest podobnie obfity w zniżki, co sugeruje, że sklep najpierw nakładał zniżki na rowery i komponenty a z czasem, rozszerzył to również o pozostałe kategorie.

Najlepszym pracownikiem był Jillian Carson, który w ciągu 2013 roku obsłużył 185 transakcji (w każdym miesiącu regularnie radził sobie dobrze, najgorszym był Amy Alberts, który obsłużył zaledwie 3 transakcje.

Najlepszym klientem był Reuben D'Sa który zakupił aż 2737 produktów. Najgorszych klientów było ponad 3500, czyli tych, którzy kupili tylko jeden produkt.

Najlepszym produktem był Full-Finger Gloves L (9 sztuk średnio sprzedanych). Na drugim miejscu ten sam produkt w rozmiarze M.

Pandas Profiling jest doskonałym narzędziem, aby automatycznie wygenerować ciekawe statystyki dotyczące zbioru danych. Udało się z łatwością zobaczyć ile jest unikalnych wartości, brakujących rekordów i inne ciekawe statystyki.