3. sprawozdanie z laboratorium Hurtownie Danych

Mikołaj Kubś, 272662

25 marca 2025

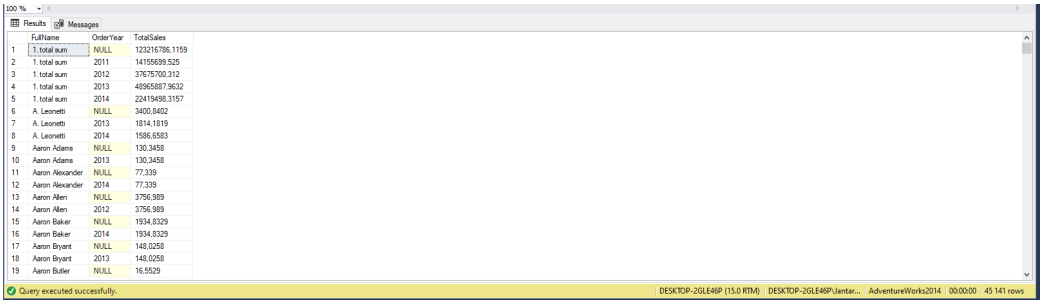
1 Zadanie 1 - funkcje grupujące

1.1

Przygotować zestawienie przedstawiające, ile pieniędzy wydali klienci na zamówienia na przestrzeni poszczególnych lat. Wykonaj zestawienie przy użyciu poleceń rollup, cube, grouping sets.

CUBE:

```
SELECT
    CASE
        WHEN GROUPING(Person.BusinessEntityID) = 1 THEN '1. total sum'
        ELSE CONCAT(MIN(Person.FirstName), ' ', MIN(Person.LastName))
    END AS FullName,
    YEAR(OrderDate) AS OrderYear,
    SUM(TotalDue) AS TotalSales
FROM Sales.SalesOrderHeader
JOIN Sales.Customer ON Customer.CustomerID = SalesOrderHeader.CustomerID
JOIN Person.Person ON Person.BusinessEntityID = Customer.PersonID
GROUP BY
    CUBE(Person.BusinessEntityID, YEAR(OrderDate))
ORDER BY
    FullName, OrderYear
```

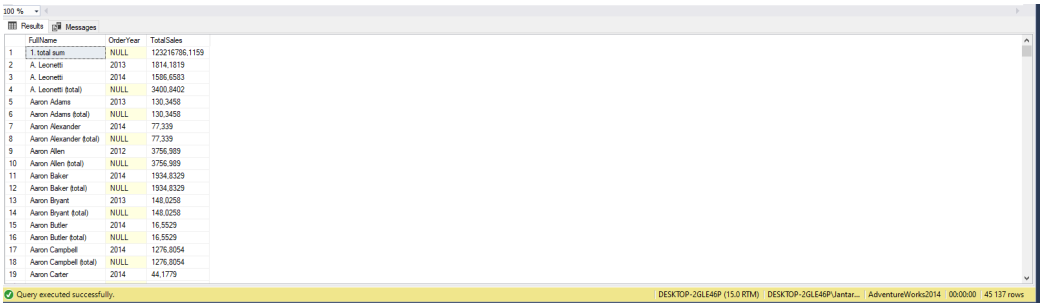


	FullName	OrderYear	TotalSales
1	1. total sum	NULL	123216786,1159
2	1. total sum	2011	14155669,525
3	1. total sum	2012	37675700,312
4	1. total sum	2013	48965887,9632
5	1. total sum	2014	22419498,3157
6	A. Leonetti	NULL	3400,8402
7	A. Leonetti	2013	1814,1819
8	A. Leonetti	2014	1586,6583
9	Aaron Adams	NULL	130,3458
10	Aaron Adams	2013	130,3458
11	Aaron Alexander	NULL	77,339
12	Aaron Alexander	2014	77,339
13	Aaron Allen	NULL	3756,989
14	Aaron Allen	2012	3756,989
15	Aaron Baker	NULL	1934,8329
16	Aaron Baker	2014	1934,8329
17	Aaron Bryant	NULL	148,0258
18	Aaron Bryant	2013	148,0258
19	Aaron Butler	NULL	16,5529

Rysunek 1: Wynik kwerendy 1.1: CUBE

ROLLUP:

```
SELECT
    CASE
        WHEN GROUPING(YEAR(OrderDate)) = 1 AND
        GROUPING(Person.BusinessEntityID) = 1
        THEN '1. total sum'
        WHEN GROUPING(YEAR(OrderDate)) = 1 THEN
        CONCAT(MIN(Person.FirstName), ' ', MIN(Person.LastName), ' (total)')
        ELSE MIN(CONCAT(Person.FirstName, ' ', Person.LastName))
    END AS FullName,
    YEAR(OrderDate) AS OrderYear,
    SUM(TotalDue) AS TotalSales
FROM Sales.SalesOrderHeader
JOIN Sales.Customer ON Customer.CustomerID = SalesOrderHeader.CustomerID
JOIN Person.Person ON Person.BusinessEntityID = Customer.PersonID
GROUP BY
    ROLLUP(Person.BusinessEntityID, YEAR(OrderDate))
ORDER BY
    FullName, OrderYear
```



	FullName	OrderYear	TotalSales
1	1. total sum	NULL	123216786,1159
2	A. Leonetti	2013	1814,1819
3	A. Leonetti	2014	1586,6583
4	A. Leonetti (total)	NULL	3400,8402
5	Aaron Adams	2013	130,3458
6	Aaron Adams (total)	NULL	130,3458
7	Aaron Alexander	2014	77,339
8	Aaron Alexander (total)	NULL	77,339
9	Aaron Allen	2012	3756,989
10	Aaron Allen (total)	NULL	3756,989
11	Aaron Baker	2014	1934,8329
12	Aaron Baker (total)	NULL	1934,8329
13	Aaron Bryant	2013	148,0258
14	Aaron Bryant (total)	NULL	148,0258
15	Aaron Butler	2014	16,5529
16	Aaron Butler (total)	NULL	16,5529
17	Aaron Campbell	2014	1276,8054
18	Aaron Campbell (total)	NULL	1276,8054
19	Aaron Carter	2014	44,1779

Rysunek 2: Wynik kwerendy 1.1: ROLLUP

GROUPING SETS:

```
SELECT
    MIN(CONCAT(Person.FirstName, ' ', Person.LastName)) AS FullName,
    YEAR(OrderDate) AS OrderYear,
    SUM(TotalDue) AS TotalDue
FROM Sales.SalesOrderHeader
JOIN Sales.Customer ON Customer.CustomerID = SalesOrderHeader.CustomerID
JOIN Person.Person ON Person.BusinessEntityID = Customer.PersonID
GROUP BY GROUPING SETS
    (
    (Person.BusinessEntityID),
        (YEAR(OrderDate), Person.BusinessEntityID)
    )
ORDER BY FullName, OrderYear
```

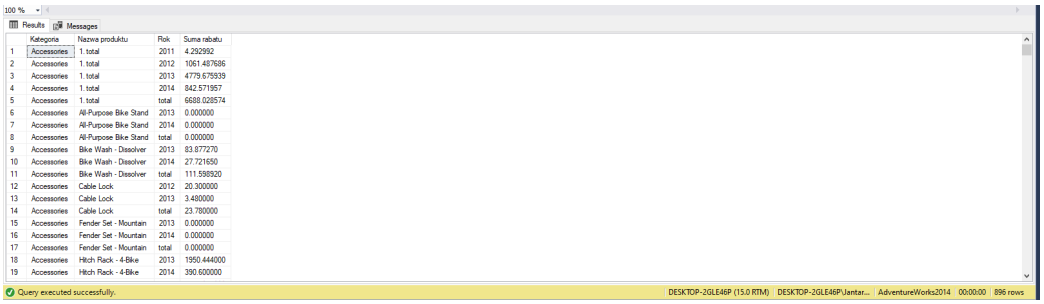
	FullName	OrderYear	TotalDue
1	A. Leonetti	NULL	3400,8402
2	A. Leonetti	2013	1814,1819
3	A. Leonetti	2014	1586,6583
4	Aaron Adams	NULL	130,3458
5	Aaron Adams	2013	130,3458
6	Aaron Alexander	NULL	77,339
7	Aaron Alexander	2014	77,339
8	Aaron Allen	NULL	3756,989
9	Aaron Allen	2012	3756,989
10	Aaron Baker	NULL	1934,8329
11	Aaron Baker	2014	1934,8329
12	Aaron Bryant	NULL	148,0258
13	Aaron Bryant	2013	148,0258
14	Aaron Butler	NULL	16,5529
15	Aaron Butler	2014	16,5529
16	Aaron Campbell	NULL	1276,8054
17	Aaron Campbell	2014	1276,8054
18	Aaron Carter	NULL	44,1779
19	Aaron Carter	2014	44,1779

Rysunek 3: Wynik kwerendy 1.1: GROUPING SETS

1.2

Przygotować zestawienie przedstawiające łączną kwotę zniżek z podziałem na katego- rie, produkty oraz lata.

```
SELECT
    ProductCategory.Name AS Kategoria,
    ISNULL(Product.Name, '1. total') AS "Nazwa produktu",
    ISNULL(CAST(YEAR(OrderDate) AS VARCHAR), 'total') AS Rok,
    SUM(UnitPrice * OrderQty - LineTotal) AS "Suma rabatu"
FROM Sales.SalesOrderDetail
JOIN Sales.SalesOrderHeader ON SalesOrderHeader.SalesOrderID =
    SalesOrderDetail.SalesOrderID
JOIN Production.Product ON SalesOrderDetail.ProductID = Product.ProductID
LEFT JOIN Production.ProductSubcategory ON
    Product.ProductSubcategoryID = ProductSubcategory.ProductSubcategoryID
LEFT JOIN Production.ProductCategory ON
    ProductSubcategory.ProductCategoryID = ProductCategory.ProductCategoryID
GROUP BY
    CUBE(Product.Name, YEAR(OrderDate)),
  ProductCategory.Name
ORDER BY
    Kategoria, "Nazwa produktu", Rok
```



	Kategoria	Nazwa produktu	Rok	Suma rabatu
1	Accessories	1. total	2011	4.292992
2	Accessories	1. total	2012	1061.487686
3	Accessories	1. total	2013	4779.675939
4	Accessories	1. total	2014	842.571957
5	Accessories	1. total	total	6688.028574
6	Accessories	All-Purpose Bike Stand	2013	0.000000
7	Accessories	All-Purpose Bike Stand	2014	0.000000
8	Accessories	All-Purpose Bike Stand	total	0.000000
9	Accessories	Bike Wash - Dissolver	2013	83.877270
10	Accessories	Bike Wash - Dissolver	2014	27.721650
11	Accessories	Bike Wash - Dissolver	total	111.598920
12	Accessories	Cable Lock	2012	20.300000
13	Accessories	Cable Lock	2013	3.480000
14	Accessories	Cable Lock	total	23.780000
15	Accessories	Fender Set - Mountain	2013	0.000000
16	Accessories	Fender Set - Mountain	2014	0.000000
17	Accessories	Fender Set - Mountain	total	0.000000
18	Accessories	Hitch Rack - 4-Bike	2013	1950.444000
19	Accessories	Hitch Rack - 4-Bike	2014	390.600000

Rysunek 4: Wynik kwerendy 1.2

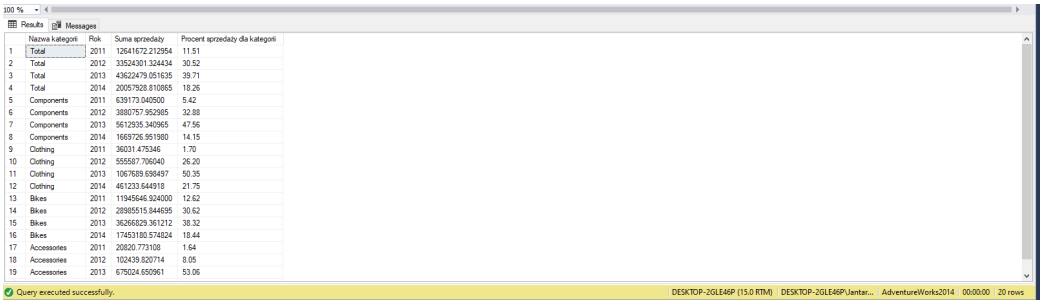
2 Zadanie 2 - funkcje okienkowe

2.1

Dla kategorii ‘Bikes’ przygotuj zestawienie prezentujące procentowy udział kwot sprzedaży produktów tej kategorii w poszczególnych latach w stosunku

do łącznej kwoty sprzedaży dla tej kategorii. W zadaniu wykorzystaj funkcje okna. Wykonaj podobne zestawienia dla pozostałych kategorii.

```
SELECT
    ISNULL(ProductCategory.Name, 'Total') AS "Nazwa kategorii",
    YEAR(SalesOrderHeader.OrderDate) AS Rok,
    SUM(SalesOrderDetail.LineTotal) AS "Suma sprzedazy",
    CAST(ROUND(SUM(SalesOrderDetail.LineTotal) / SUM(SUM(SalesOrderDetail.LineTotal))
    OVER (PARTITION BY ProductCategory.Name) * 100, 2) AS DECIMAL(10,2))
    AS "Procent sprzedazy dla kategorii"
FROM
    Sales.SalesOrderDetail
    JOIN Sales.SalesOrderHeader ON
      SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID
    JOIN Production.Product ON
      SalesOrderDetail.ProductID = Product.ProductID
    JOIN Production.ProductSubcategory ON
      Product.ProductSubcategoryID = ProductSubcategory.ProductSubcategoryID
    JOIN Production.ProductCategory ON
      ProductSubcategory.ProductCategoryID = ProductCategory.ProductCategoryID
GROUP BY
    YEAR(SalesOrderHeader.OrderDate),
    CUBE(ProductCategory.Name)
ORDER BY
    COALESCE(ProductCategory.Name, 'Total') DESC,
    YEAR(SalesOrderHeader.OrderDate);
```



	Nazwa kategorii	Rok	Suma sprzedazy	Procent sprzedazy dla kategorii
1	Total	2011	12641672.212954	11.51
2	Total	2012	33524301.324434	30.52
3	Total	2013	43622479.051635	39.71
4	Total	2014	20057928.810865	18.26
5	Components	2011	639173.040500	5.42
6	Components	2012	3880757.952985	32.88
7	Components	2013	5612935.340965	47.56
8	Components	2014	1669726.951980	14.15
9	Clothing	2011	36031.475346	1.70
10	Clothing	2012	555587.706040	26.20
11	Clothing	2013	1067689.698497	50.35
12	Clothing	2014	461233.644918	21.75
13	Bikes	2011	11945646.924000	12.62
14	Bikes	2012	28985515.844695	30.62
15	Bikes	2013	36266829.361212	38.32
16	Bikes	2014	17453180.574824	18.44
17	Accessories	2011	20820.773108	1.64
18	Accessories	2012	102439.820714	8.05
19	Accessories	2013	675024.650961	53.06

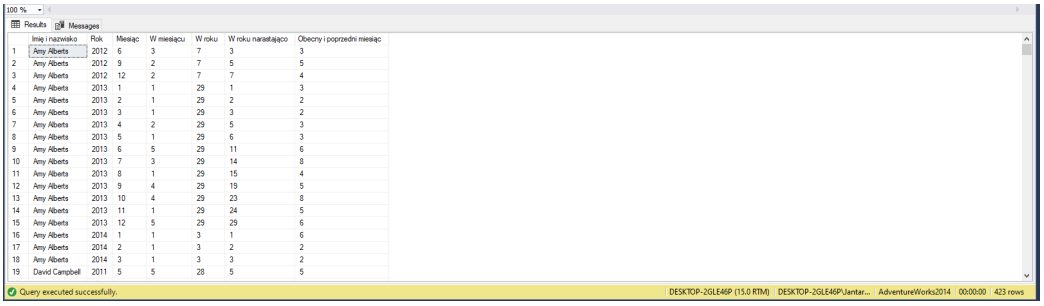
Rysunek 5: Wynik kwerendy 2.1

2.2

Przygotuj zestawienie dla sprzedawców z podziałem na lata i miesiące prezentujące liczbę obsłużonych przez nich zamówień w ciągu roku, w ciągu roku

narastająco oraz sumarycznie w obecnym i poprzednim miesiącu. W zadaniu wykorzystaj funkcje okna.

```
SELECT
    Person.FirstName + ' ' + Person.LastName AS [Imie i nazwisko],
    YEAR(SalesOrderHeader.OrderDate) AS [Rok],
    MONTH(SalesOrderHeader.OrderDate) AS [Miesiac],
    COUNT(*) AS [W miesiacu],
    SUM(COUNT(*)) OVER
      (PARTITION BY
        Person.BusinessEntityID,
        YEAR(SalesOrderHeader.OrderDate))
    AS [W roku],
    SUM(COUNT(*)) OVER
      (PARTITION BY
        Person.BusinessEntityID,
        YEAR(SalesOrderHeader.OrderDate)
      ORDER BY MONTH(SalesOrderHeader.OrderDate))
    AS [W roku narastajaco],
    COUNT(*) + LAG(COUNT(*), 1, 0) OVER
      (PARTITION BY Person.BusinessEntityID
      ORDER BY YEAR(SalesOrderHeader.OrderDate), MONTH(SalesOrderHeader.OrderDate))
    AS [Obecny i poprzedni miesiac]
FROM
    Sales.SalesOrderHeader
    JOIN Sales.SalesPerson ON
      SalesOrderHeader.SalesPersonID = SalesPerson.BusinessEntityID
    JOIN Person.Person ON
      SalesPerson.BusinessEntityID = Person.BusinessEntityID
GROUP BY
    Person.BusinessEntityID,
    Person.FirstName,
    Person.LastName,
    YEAR(SalesOrderHeader.OrderDate),
    MONTH(SalesOrderHeader.OrderDate)
ORDER BY
    [Imie i nazwisko],
    [Rok],
    [Miesiac];
```



	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
14	Amy Alberts	2013	11	1	29	24	5
15	Amy Alberts	2013	12	5	29	29	6
16	Amy Alberts	2014	1	1	3	1	6
17	Amy Alberts	2014	2	1	3	2	2
18	Amy Alberts	2014	3	1	3	3	2
19	David Campbell	2011	5	5	28	5	5

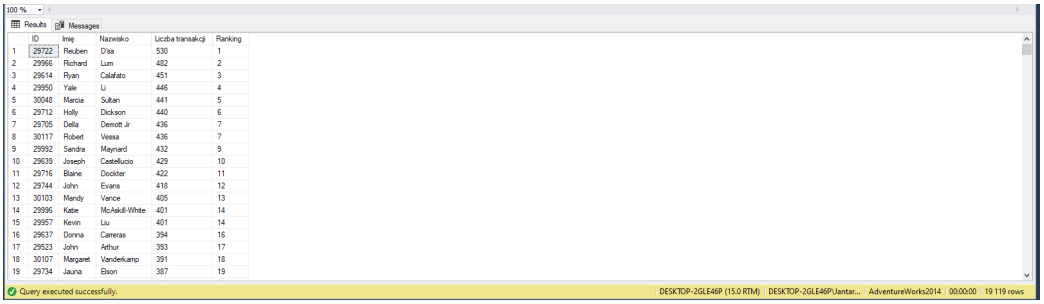
Rysunek 6: Wynik kwerendy 2.2

2.3

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

RANK:

```
SELECT
  Customer.CustomerID AS "ID",
  Person.FirstName AS "Imie",
  Person.LastName AS "Nazwisko",
  COUNT(*) AS "Liczba transakcji",
  RANK() OVER(ORDER BY COUNT(*) DESC) AS "Ranking"
FROM Sales.Customer
RIGHT JOIN Sales.SalesOrderHeader ON
  SalesOrderHeader.CustomerID = Customer.CustomerID
RIGHT JOIN Sales.SalesOrderDetail ON
  SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID
JOIN Person.Person ON Person.BusinessEntityID = Customer.PersonID
GROUP BY Customer.CustomerID, Person.FirstName, Person.LastName
ORDER BY COUNT(*) DESC
```

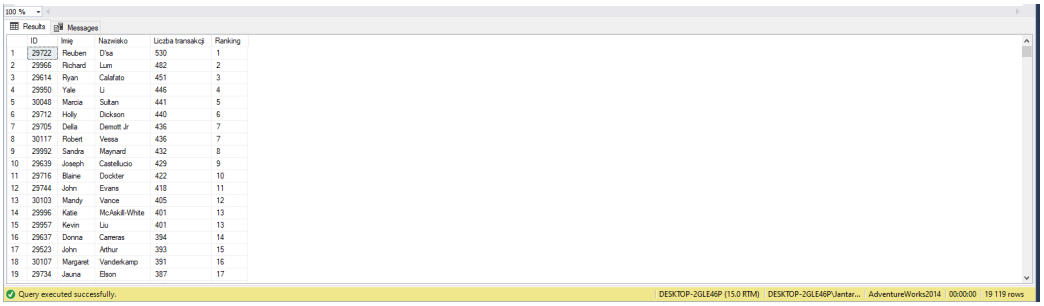


	ID	Imię	Nazwisko	Liczba transakcji	Ranking
1	29722	Reuben	D'sa	530	1
2	29966	Richard	Lum	482	2
3	29614	Ryan	Calafato	451	3
4	29950	Yale	Li	446	4
5	30048	Marcia	Sultan	441	5
6	29712	Holly	Dickson	440	6
7	29705	Della	Demott Jr	436	7
8	30117	Robert	Vessa	436	7
9	29992	Sandra	Maynard	432	9
10	29639	Joseph	Castellucio	429	10
11	29716	Blaine	Dockter	422	11
12	29744	John	Evans	418	12
13	30103	Mandy	Vance	405	13
14	29996	Katie	McAskill-White	401	14
15	29957	Kevin	Liu	401	14
16	29637	Donna	Carreras	394	16
17	29523	John	Arthur	393	17
18	30107	Margaret	Vandenkamp	391	18
19	29734	Jauna	Elson	387	19

Rysunek 7: Wynik kwerendy 2.3: RANK

DENSE_RANK:

```
SELECT
  Customer.CustomerID AS "ID",
  Person.FirstName AS "Imie",
  Person.LastName AS "Nazwisko",
  COUNT(*) AS "Liczba transakcji",
  DENSE_RANK() OVER(ORDER BY COUNT(*) DESC) AS "Ranking"
FROM Sales.Customer
RIGHT JOIN Sales.SalesOrderHeader ON
  SalesOrderHeader.CustomerID = Customer.CustomerID
RIGHT JOIN Sales.SalesOrderDetail ON
  SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID
JOIN Person.Person ON Person.BusinessEntityID = Customer.PersonID
GROUP BY Customer.CustomerID, Person.FirstName, Person.LastName
ORDER BY COUNT(*) DESC
```



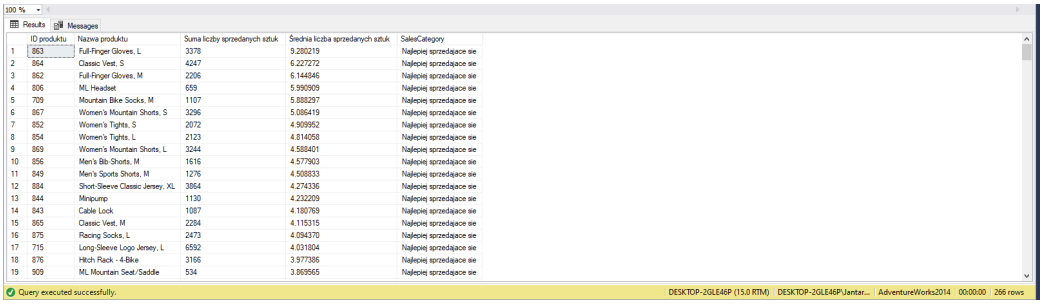
	ID	Imię	Nazwisko	Liczba transakcji	Ranking
1	29722	Reuben	D'sa	530	1
2	29966	Richard	Lum	482	2
3	29614	Ryan	Calafato	451	3
4	29950	Yale	Li	446	4
5	30048	Marcia	Sultan	441	5
6	29712	Holly	Dickson	440	6
7	29705	Della	Demott Jr	436	7
8	30117	Robert	Vessa	436	7
9	29992	Sandra	Maynard	432	8
10	29639	Joseph	Castellucio	429	9
11	29716	Blaine	Dockter	422	10
12	29744	John	Evans	418	11
13	30103	Mandy	Vance	405	12
14	29996	Katie	McAskill-White	401	13
15	29957	Kevin	Liu	401	13
16	29637	Donna	Carreras	394	14
17	29523	John	Arthur	393	15
18	30107	Margaret	Vandenkamp	391	16
19	29734	Jauna	Elson	387	17

Rysunek 8: Wynik kwerendy 2.3: DENSE_RANK

2.4

Przygotuj ranking produktów w zależności od średniej liczby sprzedanych sztuk. Wyróżnij 3 (prawie równoliczne) grupy produktów: sprzedających się najlepiej, średnio i najsłabiej.

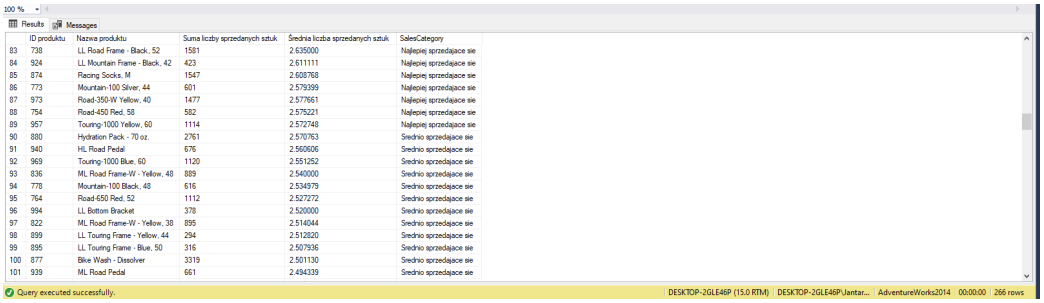
```
WITH ProductsRanked AS
(
  SELECT
    Product.ProductID AS "ID produktu",
    Product.Name AS "Nazwa produktu",
    SUM(SalesOrderDetail.OrderQty) AS "Suma liczby sprzedanych sztuk",
    AVG(CAST(SalesOrderDetail.OrderQty AS DECIMAL(10,2)))
    AS "Srednia liczba sprzedanych sztuk",
    NTILE(3) OVER (ORDER BY AVG(CAST(SalesOrderDetail.OrderQty AS FLOAT)) DESC)
    AS "Numer grupy"
  FROM Production.Product
  JOIN Sales.SalesOrderDetail
    ON SalesOrderDetail.ProductID = Product.ProductID
  GROUP BY Product.ProductID, Product.Name
)
SELECT
  ProductsRanked.[ID produktu],
  ProductsRanked.[Nazwa produktu],
  ProductsRanked.[Suma liczby sprzedanych sztuk],
  ProductsRanked.[Srednia liczba sprzedanych sztuk],
  CASE
    WHEN "Numer grupy" = 1 THEN 'Najlepiej sprzedajace sie'
    WHEN "Numer grupy" = 2 THEN 'Srednio sprzedajace sie'
    WHEN "Numer grupy" = 3 THEN 'Najslabiej sprzedajace sie'
  END AS SalesCategory
FROM ProductsRanked
ORDER BY ProductsRanked.[Srednia liczba sprzedanych sztuk] DESC
```



	ID produktu	Nazwa produktu	Suma liczby sprzedanych sztuk	Średnia liczba sprzedanych sztuk	SalesCategory
1	863	Full-Finger Gloves, L	3378	9.280219	Najlepiej sprzedajace sie
2	864	Classic Vest, S	4247	6.227272	Najlepiej sprzedajace sie
3	862	Full-Finger Gloves, M	2206	6.144846	Najlepiej sprzedajace sie
4	806	ML Headset	659	5.990909	Najlepiej sprzedajace sie
5	709	Mountain Bike Socks, M	1107	5.888297	Najlepiej sprzedajace sie
6	867	Women's Mountain Shorts, S	3296	5.086419	Najlepiej sprzedajace sie
7	852	Women's Tights, S	2072	4.909952	Najlepiej sprzedajace sie
8	854	Women's Tights, L	2123	4.814058	Najlepiej sprzedajace sie
9	869	Women's Mountain Shorts, L	3244	4.588401	Najlepiej sprzedajace sie
10	856	Men's Bib-Shorts, M	1616	4.577903	Najlepiej sprzedajace sie
11	849	Men's Sports Shorts, M	1276	4.508833	Najlepiej sprzedajace sie
12	884	Short-Sleeve Classic Jersey, XL	3864	4.274336	Najlepiej sprzedajace sie
13	844	Minipump	1130	4.232209	Najlepiej sprzedajace sie
14	843	Cable Lock	1087	4.180769	Najlepiej sprzedajace sie
15	865	Classic Vest, M	2284	4.115315	Najlepiej sprzedajace sie
16	875	Racing Socks, L	2473	4.094370	Najlepiej sprzedajace sie
17	715	Long-Sleeve Logo Jersey, L	6592	4.031804	Najlepiej sprzedajace sie
18	876	Hitch Rack - 4-Bike	3166	3.977386	Najlepiej sprzedajace sie
19	909	ML Mountain Seat/Saddle	534	3.869565	Najlepiej sprzedajace sie

Query executed successfully. DESKTOP-2GLE46P (15.0 RTM) DESKTOP-2GLE46P\Jantar... AdventureWorks2014 00:00:00 266 rows

Rysunek 9: Wynik kwerendy 2.4



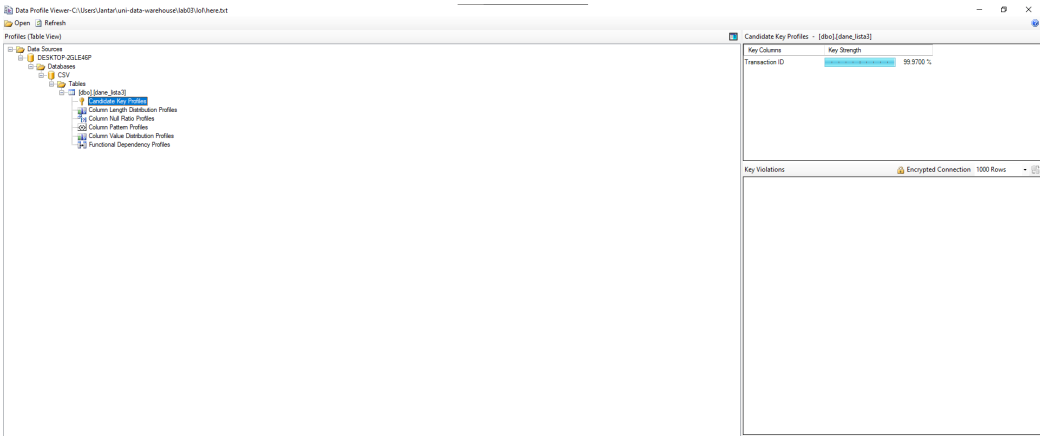
	ID produktu	Nazwa produktu	Suma liczby sprzedanych sztuk	Średnia liczba sprzedanych sztuk	SalesCategory
83	738	LL Road Frame - Black, 52	1581	2.635000	Najlepiej sprzedajace sie
84	924	LL Mountain Frame - Black, 42	423	2.611111	Najlepiej sprzedajace sie
85	874	Racing Socks, M	1547	2.608768	Najlepiej sprzedajace sie
86	773	Mountain-100 Silver, 44	601	2.579399	Najlepiej sprzedajace sie
87	973	Road-350-W Yellow, 40	1477	2.577661	Najlepiej sprzedajace sie
88	754	Road-450 Red, 58	582	2.575221	Najlepiej sprzedajace sie
89	957	Touring-1000 Yellow, 60	1114	2.572748	Najlepiej sprzedajace sie
90	880	Hydration Pack - 70 oz.	2761	2.570763	Srednio sprzedajace sie
91	940	HL Road Pedal	676	2.560606	Srednio sprzedajace sie
92	969	Touring-1000 Blue, 60	1120	2.551252	Srednio sprzedajace sie
93	836	ML Road Frame-W - Yellow, 48	889	2.540000	Srednio sprzedajace sie
94	778	Mountain-100 Black, 48	616	2.534979	Srednio sprzedajace sie
95	764	Road-650 Red, 52	1112	2.527272	Srednio sprzedajace sie
96	994	LL Bottom Bracket	378	2.520000	Srednio sprzedajace sie
97	822	ML Road Frame-W - Yellow, 38	895	2.514044	Srednio sprzedajace sie
98	899	LL Touring Frame - Yellow, 44	294	2.512820	Srednio sprzedajace sie
99	895	LL Touring Frame - Blue, 50	316	2.507936	Srednio sprzedajace sie
100	877	Bike Wash - Dissolver	3319	2.501130	Srednio sprzedajace sie
101	939	ML Road Pedal	661	2.494339	Srednio sprzedajace sie

Query executed successfully. DESKTOP-2GLE46P (15.0 RTM) DESKTOP-2GLE46P\Jantar... AdventureWorks2014 00:00:00 266 rows

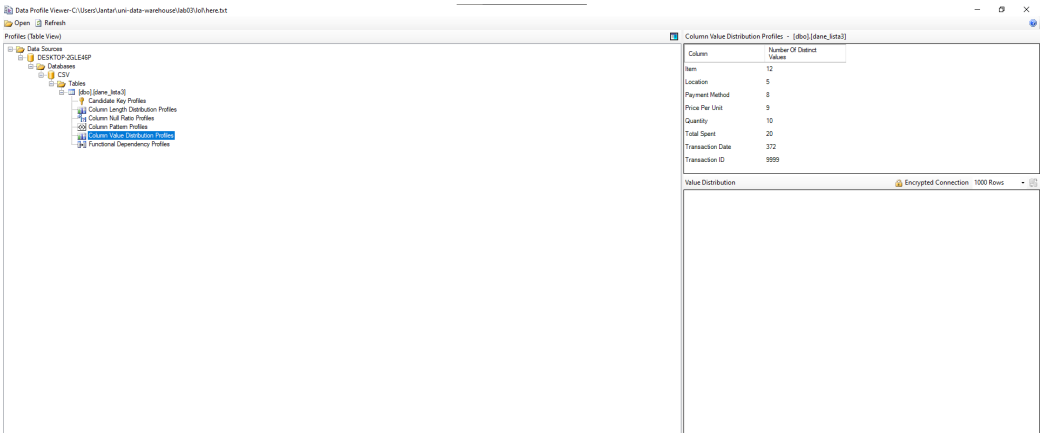
Rysunek 10: Widok przejścia do części średnio sprzedającej się

3 Zadanie 3 - profilowanie danych

Po próbach analizy w SSIS uzyskano tylko część rezultatów.



Rysunek 11: Profil kolumny kandydującej



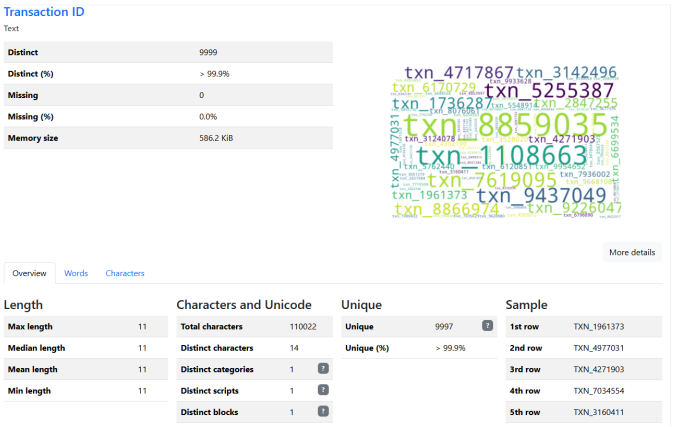
Rysunek 12: Liczba unikatowych wartości w kolumnach

Z powodu problemów technicznych resztę analizy przeprowadzono używając *Pythona* oraz bibliotek *Pandas* i *ydata_profiling*.

```
import pandas as pd
from ydata_profiling import ProfileReport
from os import path

dir_path: str = path.dirname(path.realpath(__file__))

df: pd.DataFrame = pd.read_csv(path.join(dir_path, "dane_lista3.csv"))

```

```
profile = ProfileReport(df, explorative=True)

profile.to_file(path.join(dir_path, "python_results.html"))

df['Transaction ID'] = df['Transaction ID'].astype('str')
df['Item'] = df['Item'].astype('str')
df['Quantity'] = pd.to_numeric(df['Quantity'], errors='coerce')
df['Price Per Unit'] = pd.to_numeric(df['Price Per Unit'], errors='coerce')
df['Total Spent'] = pd.to_numeric(df['Total Spent'], errors='coerce')
df['Payment Method'] = df['Payment Method'].astype('str')
df['Location'] = df['Location'].astype('str')
df['Transaction Date'] = pd.to_datetime(
    df['Transaction Date'], errors='coerce')

profile = ProfileReport(df, explorative=True)

profile.to_file(path.join(dir_path, "python_results_good_data_types.html"))
```

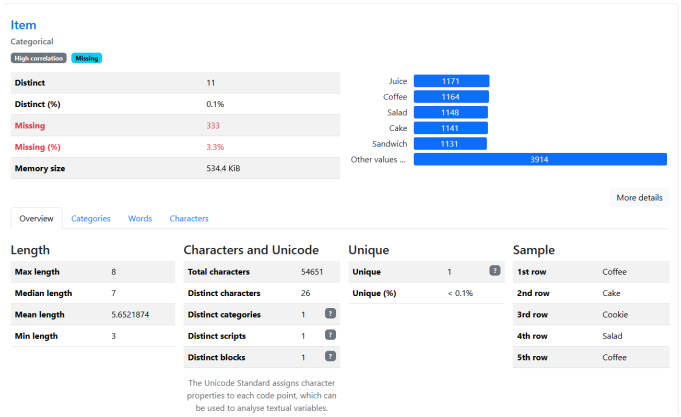
3.1 Transaction ID



Rysunek 13: Profil kolumny *Transaction ID*

Tak jak też wywnioskowano wcześniej, jest to jedyna kolumna nadająca się na klucz kandydujący. Są w niej tylko 3 duplikaty. Zawsze ma też tą samą długość - 11 znaków.

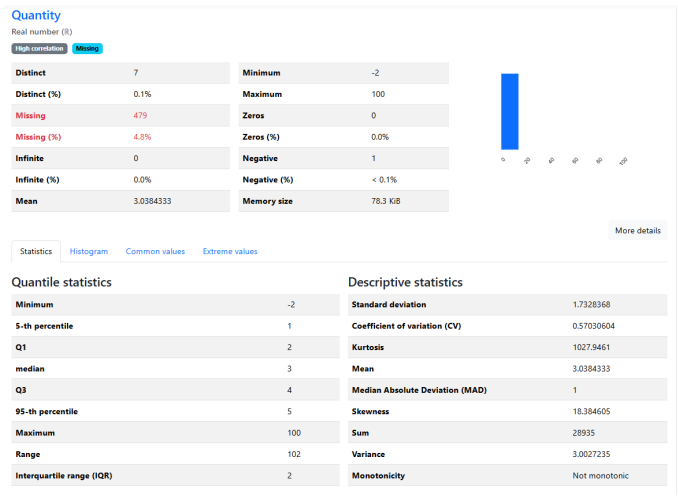
3.2 Item



Rysunek 14: Profil kolumny *Item*

Kolumna jest kategoryczna i przypisuje transakcjom kategorię kupionego towaru. W 3.3% wierszy brakuje wartości. Dodatkowo w 3.4% to *UNKNOWN*.

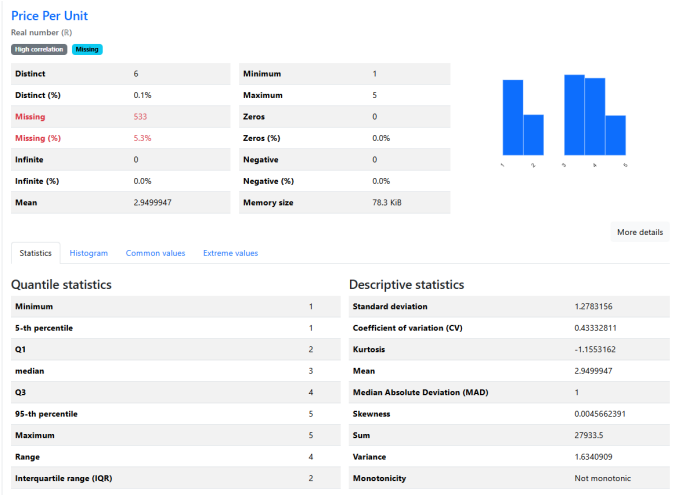
3.3 Quantity



Rysunek 15: Profil kolumny *Quantity*

W kolumnie występują anomalnie. Zdarzyła się raz wartość ujemna: -2. Zdarzyła się raz wartość 100, znacznie przekraczająca poza zakres innych (1-5), ale być może dozwolona. Brakuje danych w 4.8% wierszy. Wartości standardowe, czyli od 1 do 5 włącznie, występują z podobną częstością.

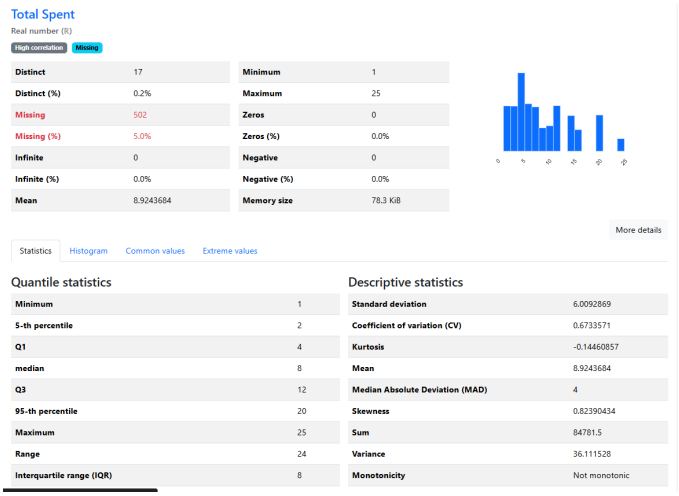
3.4 Price Per Unit



Rysunek 16: Profil kolumny *Price Per Unit*

Kolumna w większości składa się z liczb całkowitych, ale 11.3% wartości to jedyne z wartością po przecinku. Wszystkie wynoszą dokładnie 1,5. Brakuje wartości dla 5.3%. Standardowe i jedyne poza brakującymi wartości to liczby całkowite od 1-5 oraz 1,5. Najczęściej występuje wartość 3. Wystepowanie wartości 1,5 jest raczej jednak jak najbardziej dozwolone.

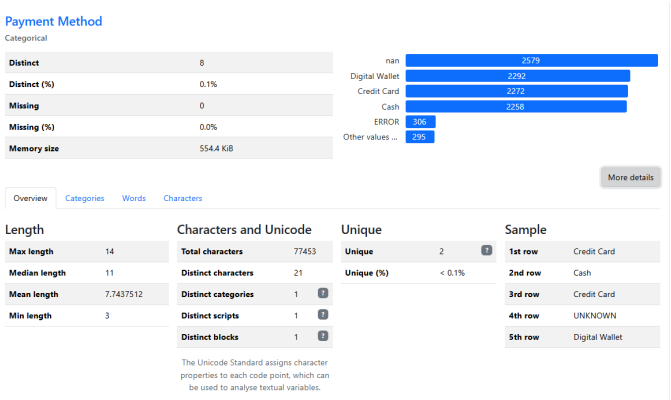
3.5 Total Spent



Rysunek 17: Profil kolumny *Total Spent*

Brakuje 5.02% wartości. Przedział to liczby od 1 do 25 (ale tylko 17 z nich jest w danych). Większość wartości to liczby całkowite, ale występują też wielokrotności 1,5, sugerując, że cena 1,5 w Price Per Unit nie jest żadną anomalią. Najczęściej występuje wartość 6. Większość wartości jest bliżej początku rozkładu niż końca.

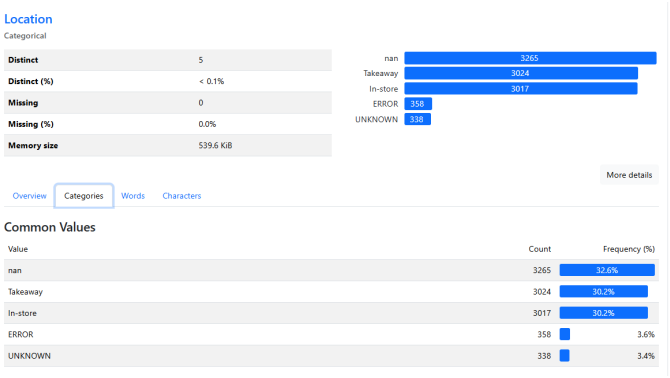
3.6 Payment Method



Rysunek 18: Profil kolumny *Payment Method*

Niepoprawne to 31,78% całości - połaczenie *nan*, *ERROR* i *UNKNOWN*. Oczywiście przy odczytywaniu karty mogą zdarzać się błędy, ale wtedy transakcje raczej powinny być odrzucane i nie znajdować się w bazie danych. Dodatkowo 2 razy wystąpiły literówki - Digital Walle i CreditCard zamiast Digital Wallet i Credit Card (występujących znacznie częściej). Poprawne kategorie to Digital Wallet, Credit Card i Cash.

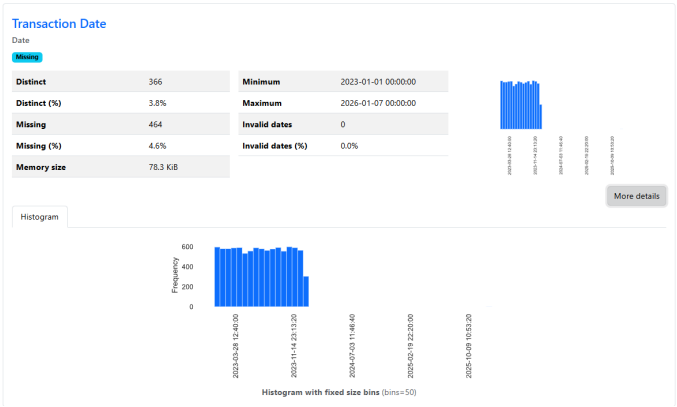
3.7 Location



Rysunek 19: Profil kolumny *Location*

Podobnie jak poprzednio, wielu danych brakuje. Nan, ERROR i UNKNOWN występują z częstością 39,61%. Poza tym, 2 poprawne kategorie to Takeaway lub In-store.

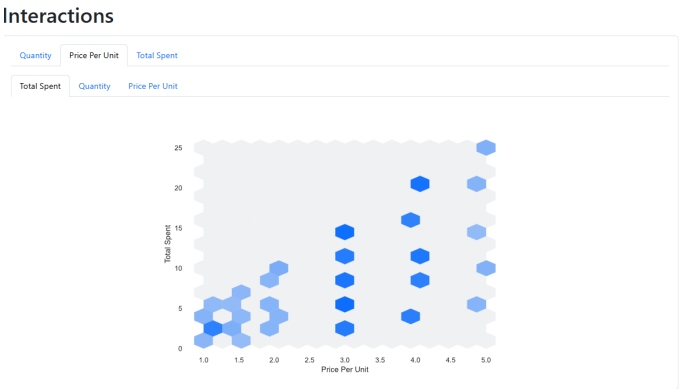
3.8 Transaction Date



Rysunek 20: Profil kolumny *Transaction Date*

Przez jedną literówkę w danych zakres wydaje się być od 2023-01-01 do 2026-01-07. Rzeczywiście jednak dane są od 2023-01-01 do 2023-12-31, a wystąpiła literówka w 2026 i rok powinien zostać zmieniony na 2023. Poza tym rozkład jest dość równomierny, jedynie w grudniu jest wyraźny spadek. Brakuje 4,6% wartości.

3.9 Przykład interakcji



Rysunek 21: Przykład interakcji między Price Per Unit a Total Spent

Jak widać, występuje pozytywna korelacja miedzy Price Per Unit a Total Spent.

4 Wnioski

4.1 Wnioski z zadania 1

Klienci bardzo różnią się miedzy sobą. Niektórzy wydają tylko w jednym roku niskie sumy, inni wydają na przestrzeni lat duże pieniądze. Ostatecznie sprowadza się to do wysokich przychodów firmy, rosnących między 2011-2013. Jest szansa na utrzymanie tego trendu w 2014 roku.

Wiele produktów nigdy nie miało rabatu. Najwięcej rabatów udzielono w 2013 roku, ale ta wartość i tak jest niska, biorąc pod uwagę przychody firmy. Zdecydowanie najwięcej rabatów udzielono na rowery - reszta kategorii nie osiągnęła sumarycznie nawet 10% sumy rabatów dla rowerów.

4.2 Wnioski z zadania 2

Podobnie jak wcześniej widać trendy firmy - przychodów rosnących między 2011-2013. W 2014 roku najbardziej spadła sprzedaż komponentów, a najmniej akcesoriów.

Tak jak w poprzednim raporcie zauważono, nie każdy sprzedawca obsługuje tyle samo transakcji co inny w danym roku. Na przykład, Syed Abbas w 2013 roku obsłużył 12 transakcji, a w 2014 tylko 4. Dla porównania Michael

Blythe obsłużył w 2011 65 transakcji, w 2012 - 148, 2013 - 175 a w 2014 - 62.

W tym przypadku DENSE_RANK wygląda lepiej niż RANK. Oba ostatecznie funkcjonują tak samo, ale wielu klientów ma tą samą liczbę transakcji (na przykład 2). Powoduje to okazjonalne duże skoki w miejscach rankingowych. W DENSE_RANK to zjawisko nie występuje. Zdecydowana większość klientów wykonała mało transakcji, ale 746 (na 19119) klientów wykonało ich co najmniej 10. 214 klientów wykonało co najmniej 100 transakcji, a rekordzistą jest Reuben D'sa z 530 transakcjami.

Do zadania 2.4 należało oczywiście użyć funkcji NTILE. Warto byłoby zobaczyć porównaniu między rankingiem opartym na średniej liczbie sprzedanych sztuk a sumą liczby sprzedanych sztuk/sumą przychodów. Ranking w aktualnej postaci dotyczy tego, ile w tej samej transakcji zakupiono średnio dany produkt. Nie oznacza to, że ten produkt jest faktycznie najbardziej opłacalny dla sklepu.

4.3 Wnioski z zadania 3

Jakość danych jest niska i wynika to prawdopodobnie z braku odpowiedniej walidacji w systemie zbierającym dane. Raczej niepoprawne jest przechowywanie w bazie danych transakcji, gdzie wystąpił błąd przy odczycie metody płatności. Możliwe, że dane nie były wpisywane automatycznie, tylko wymagały za każdym razem manualnego wpisu. Występują poważne literówki i błędy, takie jak literówki w kategorii karty, ujemna wartość Quantity czy rok 2026 w Transaction Date.

Największym problemem jest jednak liczba danych, jakiej brakuje - w zależności od kolumny waha się do nawet 40%. W tylko jednej kolumnie jakość danych jest prawie idealna - Transaction ID ma tylko 3 duplikaty i wszędzie ma wartości. Wykluczając Transaction ID, przedział brakujących wartości wynosi od 3,4% do 39,61%.

Najwięcej problemów mają kolumny Location i Payment Method. W Location brakuje 39,61% danych, a w Payment Method, wliczając ERROR i UNKNOWN 31,78%. Poza tym, w Payment Method wystąpiła 2 razy literówka.

Te problemy mocno utrudniłyby przeniesienie podanych danych do porządnej bazy danych z zapisem wszystkich wierszy. Dane te jednak można jak najbardziej przeanalizować, a nawet naprawić oczywiste błędy. Literówki w kategoriach karty można łatwo poprawić, usunąć wiersz z ujemną Quantity i poprawić rok 2026 na 2023. Można nawet próbować domyślić się niektórych brakujących wartości - jeśli brakuje tylko 1 wierzchołka w trójkącie Quantity, Price Per Unit i Total Spent, da się go wywnioskować. Bez dodatkowej

wiedzy nie będzie można jednak przywrócić innych brakujących danych.

Nowoczesne narzędzia do analizy danych potrafią bardzo mocno i szybko wspomóc proces analityka i inżyniera danych.