3. sprawozdanie z laboratorium Hurtownie Danych

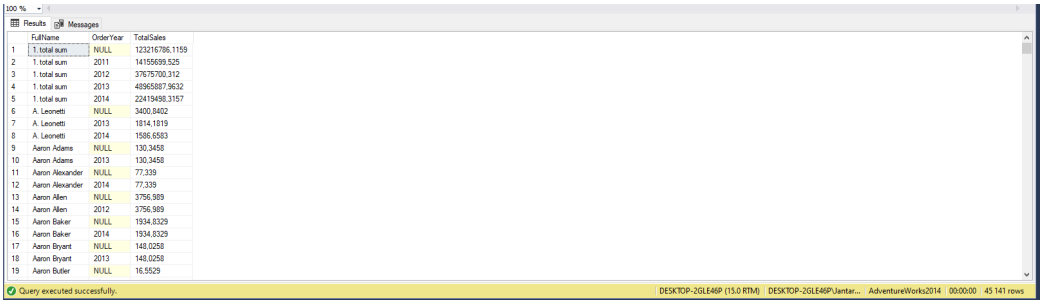
Mikołaj Kubś, 272662

25 marca 2025

1 Zadanie 1 - funkcje grupujące

1.1

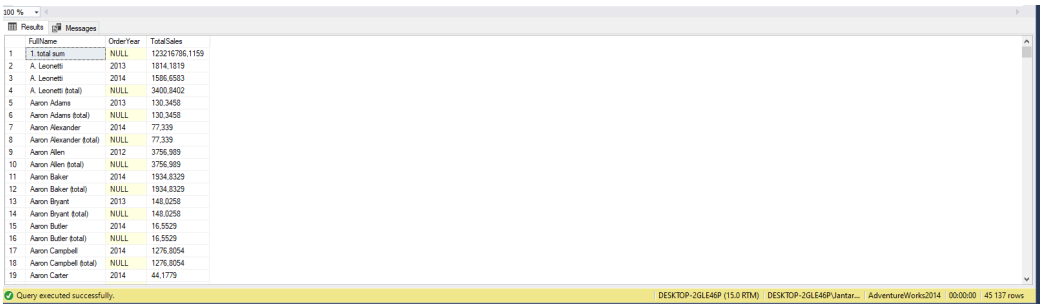
```
SELECT
    CASE
        WHEN GROUPING(Person.BusinessEntityID) = 1 THEN '1. total sum'
        ELSE CONCAT(MIN(Person.FirstName), ' ', MIN(Person.LastName))
    END AS FullName,
    YEAR(OrderDate) AS OrderYear,
    SUM(TotalDue) AS TotalSales
FROM Sales.SalesOrderHeader
JOIN Sales.Customer ON Customer.CustomerID = 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	14155689.525
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

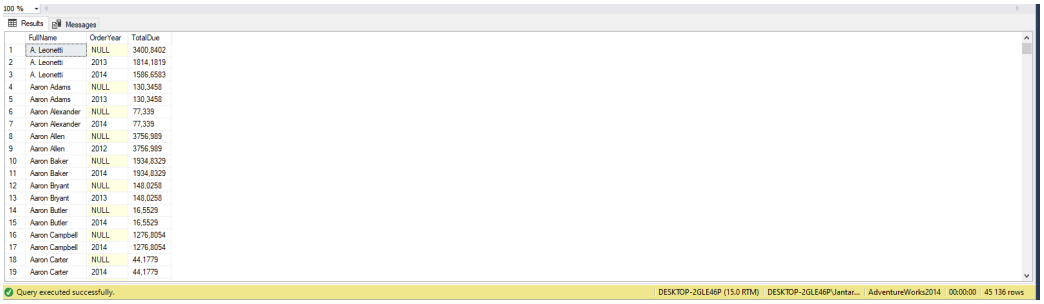
```sql
SELECT
    CASE
        WHEN GROUPING(YEAR(OrderDate)) = 1 AND
        GROUPING(Person.BusinessEntityID) = 1
        THEN '1. total sum'
        WHEN GROUPING(YEAR(OrderDate)) = 1 THEN
        CONCAT(MIN(Person.FirstName), ' ', MIN(Person.LastName), ' (total)')
        ELSE MIN(CONCAT(Person.FirstName, ' ', 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

```sql
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
```



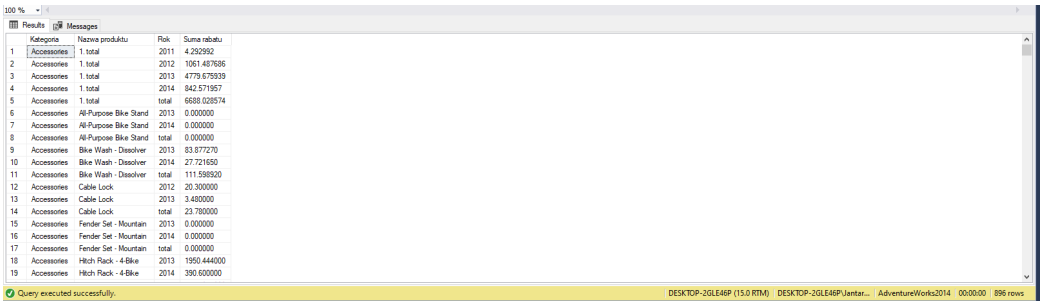
	FullName	OrderYear	TotalDue
1	A. Leonetti	NULL	3400,8402
2	A. Leonetti	2013	1814,1819
3	A. Leonetti	2014	1586,6583
4	Aaron Adams	NULL	130,3458
5	Aaron Adams	2013	130,3458
6	Aaron Alexander	NULL	77,339
7	Aaron Alexander	2014	77,339
8	Aaron Allen	NULL	3756,989
9	Aaron Allen	2012	3756,989
10	Aaron Baker	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

1.2

```sql
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
```

```
15      CUBE(Product.Name, YEAR(OrderDate)),
16    ProductCategory.Name
17  ORDER BY
18      Kategoria, "Nazwa produktu", Rok
```



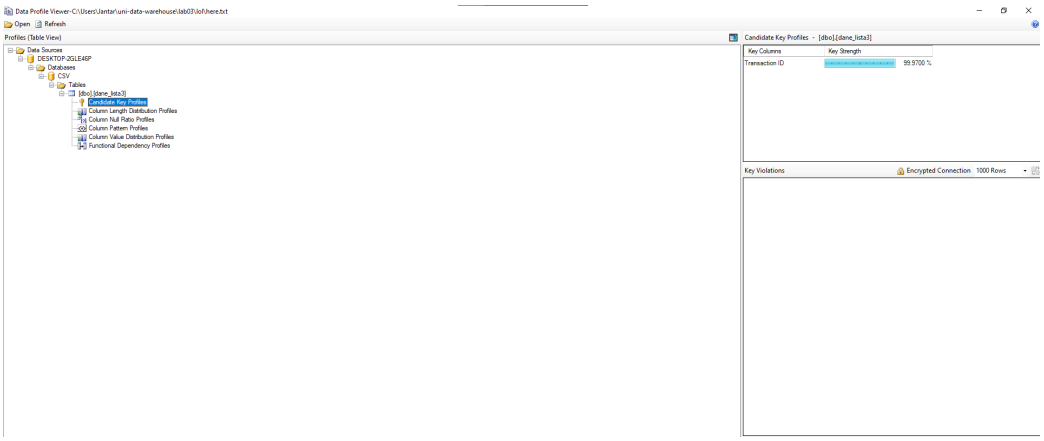
	Kategoria	Nazwa produktu	Rok	Suma rabatu
1	Accessories	1. total	2011	4.292992
2	Accessories	1. total	2012	1061.487686
3	Accessories	1. total	2013	4779.675939
4	Accessories	1. total	2014	842.571957
5	Accessories	1. total	total	6688.028574
6	Accessories	All-Purpose Bike Stand	2013	0.000000
7	Accessories	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

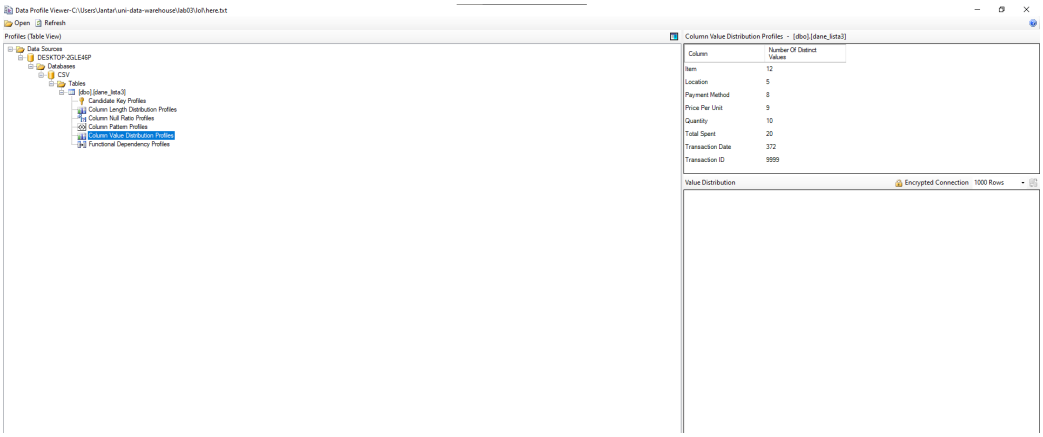
2 Zadanie 2 - funkcje okienkowe

3 Zadanie 3 - profilowanie danych

Po próbach analizy w SSIS uzyskano tylko część rezultatów.



Rysunek 5: Profil kolumny kandydującej

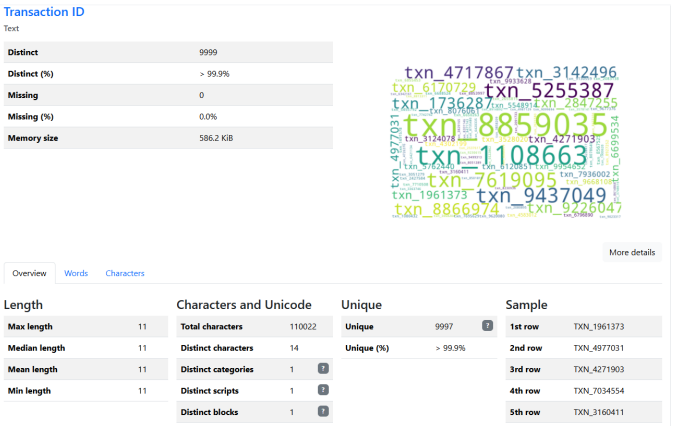


Rysunek 6: Liczba unikatowych wartości w kolumnach

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"))
```

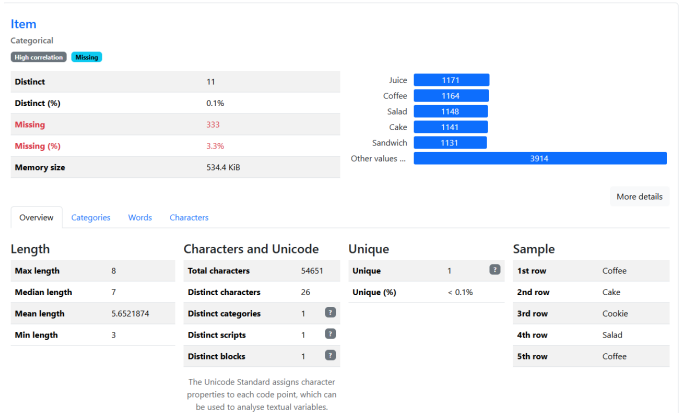
3.1 Transaction ID



Rysunek 7: 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.

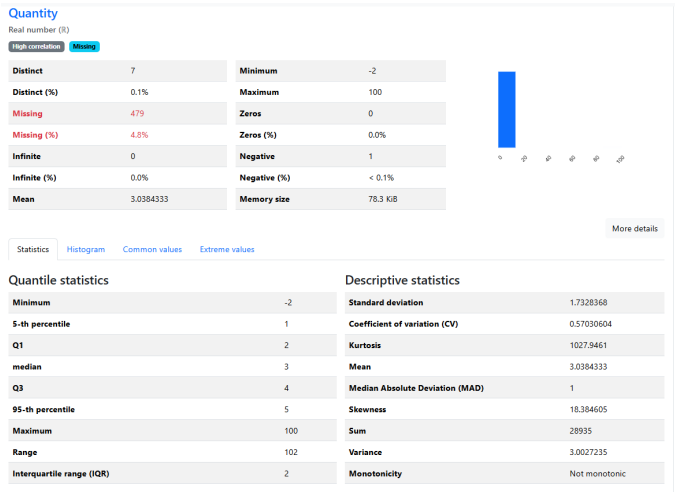
3.2 Item



Rysunek 8: Profil kolumny *Item*

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

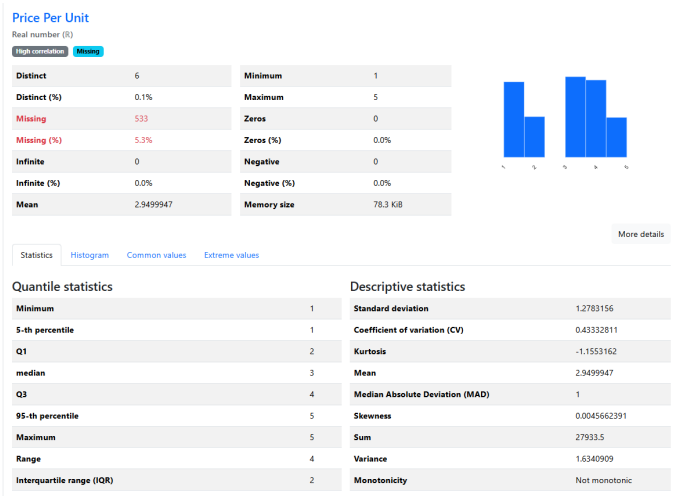
3.3 Quantity



Rysunek 9: 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.

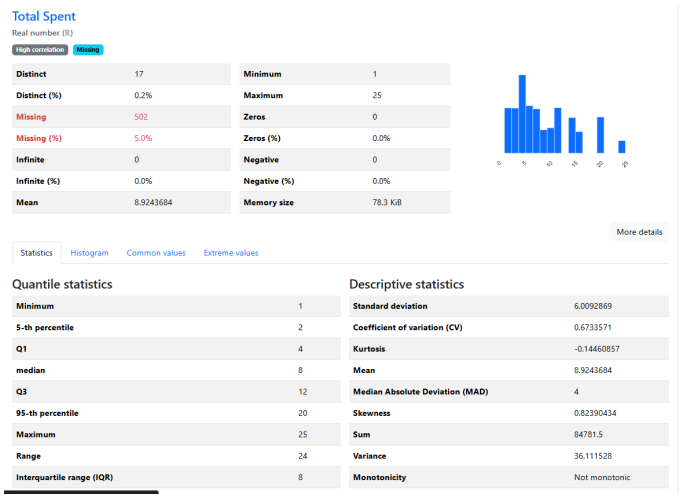
3.4 Price Per Unit



Rysunek 10: 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.

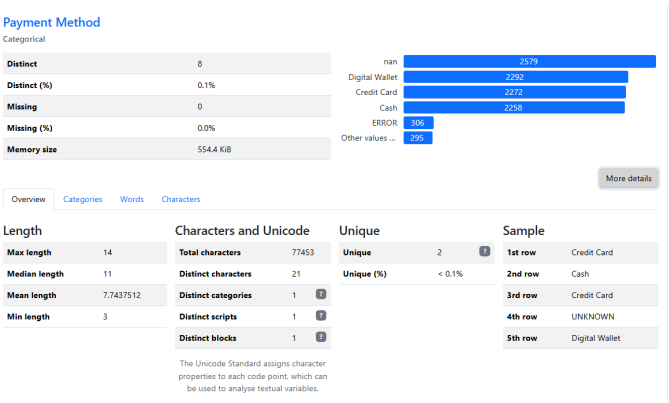
3.5 Total Spent



Rysunek 11: 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.

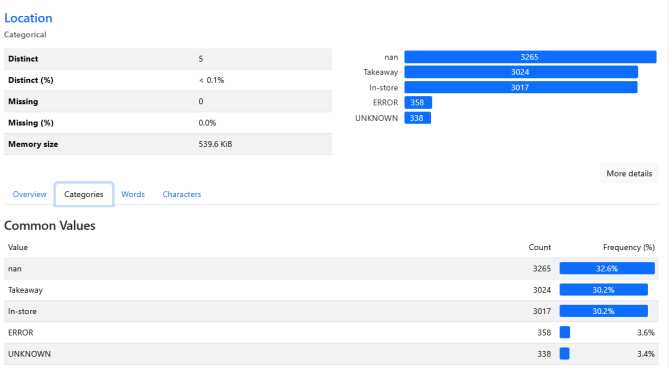
3.6 Payment Method



Rysunek 12: 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.

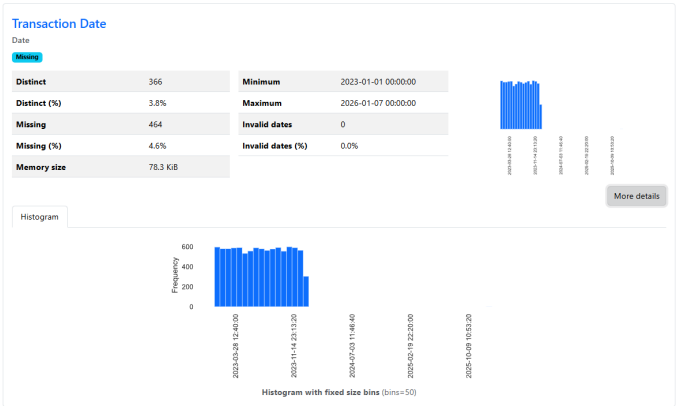
3.7 Location



Rysunek 13: 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.

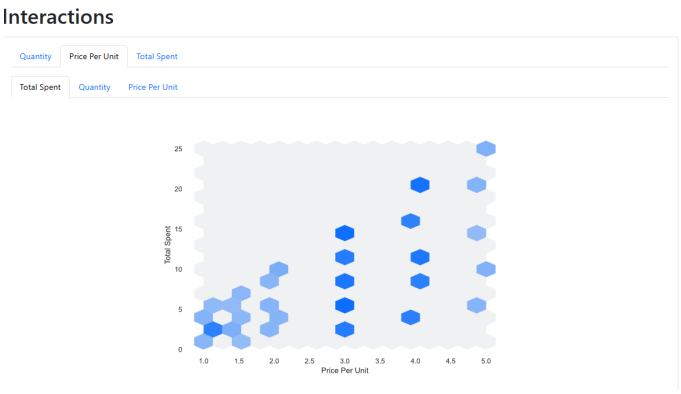
3.8 Transaction Date



Rysunek 14: 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 15: Przykład interakcji między Price Per Unit a Total Spent

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