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

```sql
SELECT
    COALESCE(CONCAT(Person.FirstName, ' ', Person.LastName), '')'FullName',
    COALESCE(CAST(YEAR(OrderDate) AS VARCHAR), '') 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(CONCAT(Person.FirstName, ' ', Person.LastName), YEAR(OrderDate))
ORDER BY
    1, 2 DESC;
```

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

```sql
SELECT
    COALESCE(CONCAT(Person.FirstName, ' ', Person.LastName), '')'FullName',
    COALESCE(CAST(YEAR(OrderDate) AS VARCHAR), '') 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 (CONCAT(Person.FirstName, ' ', Person.LastName), YEAR(OrderDate))
ORDER BY
    1, 2 DESC;
```

	FullName	OrderYear	TotalSales
1			123216786,1159
2	A. Leonetti	2014	1586,6583
3	A. Leonetti	2013	1814,1819
4	A. Leonetti		3400,8402
5	Aaron Adams	2013	130,3458
6	Aaron Adams		130,3458
7	Aaron Alexander	2014	77,339
8	Aaron Alexander		77,339
9	Aaron Allen	2012	3756,989
10	Aaron Allen		3756,989

```sql
SELECT
    COALESCE(CONCAT(Person.FirstName, ' ', Person.LastName), '')'FullName',
    COALESCE(CAST(YEAR(OrderDate) AS VARCHAR), '') 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 GROUPING SETS
    (
        (CONCAT(Person.FirstName, ' ', Person.LastName)),
        (YEAR(OrderDate), CONCAT(Person.FirstName, ' ', Person.LastName))
    )
ORDER BY
    1, 2 DESC;
```

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

Porównanie ROLLUP, CUBE i GROUPING SETS

- Rollup tworzy hierarchicznie sumowania, przechodząc od najniższego poziomu szczegółowości do sum całkowitych
- Cube tworzy wszystkie możliwe kombinacje agregacji, w przeciwieństwie do Rollupa wyświetli wszystkie możliwe podsumowania, nie tylko hierarchiczne. W przykładzie np. dodatkowo widoczne podsumowania dla konkretnych lat, których nie ma w Rollupie.
- Grouping Sets pozwala jawnie określić, które grupowania mają być uwzględnione. W przykładzie zdefiniowany szczegółowy poziom (Rok, FullName) i suma dla poszczególnych pracowników.

Przygotować zestawienie przedstawiające łączną kwotę zniżek z podziałem na kategorie, produkty oraz lata.

```
-- zadanie 1.2
SELECT
    PC.Name AS Kategoria,
    P.Name AS Produkt,
    COALESCE(CAST(YEAR(SOH.OrderDate) AS VARCHAR), '') AS Rok,
    SUM(SOD.LineTotal * SOD.UnitPriceDiscount) AS Kwota
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
GROUP BY GROUPING SETS (
    (PC.Name, P.Name, YEAR(SOH.OrderDate)),
    (PC.Name, P.Name)
)
ORDER BY PC.Name, P.Name, Rok DESC;
```

	Kategoria	Produkt	Rok	Kwota
1	Accessories	All-Purpose Bike Stand	2014	0.000000
2	Accessories	All-Purpose Bike Stand	2013	0.000000
3	Accessories	All-Purpose Bike Stand		0.000000
4	Accessories	Bike Wash - Dissolver	2014	26.944219
5	Accessories	Bike Wash - Dissolver	2013	79.922446
6	Accessories	Bike Wash - Dissolver		106.866665
7	Accessories	Cable Lock	2013	3.410400
8	Accessories	Cable Lock	2012	19.894000
9	Accessories	Cable Lock		23.304400
10	Accessories	Fender Set - Mountain	2014	0.000000
11	Accessories	Fender Set - Mountain	2013	0.000000
12	Accessories	Fender Set - Mountain		0.000000
13	Accessories	Hitch Rack - 4-Bike	2014	379.422000
14	Accessories	Hitch Rack - 4-Bike	2013	1846.554120
15	Accessories	Hitch Rack - 4-Bike		2225.976120
16	Accessories	HL Mountain Tire	2014	0.000000
17	Accessories	HL Mountain Tire	2013	0.000000
18	Accessories	HL Mountain Tire		0.000000

	Rok	Kategoria	Kwota_znizek
1	NULL		527507,9262
2	2011	Accessories	4,293
3	2011	Bikes	4344,6957
4	2011	Clothing	90,9593
5	2011	Components	0,00
6	2011		4439,948
7	2012	Accessories	1061,4872
8	2012	Bikes	180073,3397
9	2012	Clothing	4498,9691
10	2012	Components	961,8196
11	2012		186595,6156
12	2013	Accessories	4779,6779
13	2013	Bikes	277731,7099
14	2013	Clothing	12847,2903
15	2013	Components	4212,9922
16	2013		299571,6703
17	2014	Accessories	842,5712
18	2014	Bikes	32490,9079
19	2014	Clothing	3527,2866
20	2014	Components	39,9266
21	2014		36900,6923

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.

```sql
SELECT
    *
FROM
    (
        SELECT
            PC.Name 'Kategoria',
            YEAR(SOH.OrderDate) 'Rok',
            ROUND(
                SUM(SOH.TotalDue) OVER(PARTITION BY PC.Name, YEAR(SOH.OrderDate)) * 100 / SUM(SOH.TotalDue) OVER(Partition BY PC.Name),
                2
            ) 'Procent'
        FROM
            Production.Product P
            JOIN Sales.SalesOrderDetail SOD ON P.ProductID = SOD.ProductID
            JOIN Production.ProductSubcategory PS ON P.ProductSubcategoryID = PS.ProductSubcategoryID
            JOIN Production.ProductCategory PC ON PC.ProductCategoryID = PS.ProductCategoryID
            JOIN Sales.SalesOrderHeader SOH ON SOD.SalesOrderID = SOH.SalesOrderID
        WHERE
            PC.Name IN ('Accessories', 'Bikes', 'Components', 'Clothing')
    ) AS src
PIVOT
    (
        MAX(Procent)
        FOR Rok IN ([2011], [2012], [2013], [2014])

    ) AS pvot_table
ORDER BY 1;
```

	Kategoria	2011	2012	2013	2014
1	Accessories	5,62	26,20	48,52	19,66
2	Bikes	7,93	34,51	42,60	14,96
3	Clothing	4,62	36,11	45,82	13,45
4	Components	3,93	31,92	49,37	14,78

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.

```sql
SELECT
    *,
    SUM("w miesiacu") OVER (
        PARTITION BY "Imię i nazwisko",
        Rok
        ORDER BY Miesiac ROWS BETWEEN 1 PRECEDING AND CURRENT ROW
    ) 'obecny i poprzedni miesiac'
FROM
    (
        SELECT
            DISTINCT CONCAT(per.FirstName, ' ', per.LastName) "Imię i nazwisko",
            YEAR(soh.OrderDate) Rok,
            MONTH(soh.OrderDate) Miesiac,
            COUNT(soh.SalesOrderID) OVER (
                PARTITION BY CONCAT(per.FirstName, ' ', per.LastName),
                YEAR(soh.OrderDate),
                MONTH(soh.OrderDate)
            ) "w miesiacu",

            COUNT(soh.SalesOrderID) OVER (
                PARTITION BY CONCAT(per.FirstName, ' ', per.LastName),
                YEAR(soh.OrderDate)
            ) "w roku",

            COUNT(soh.SalesOrderID) OVER (
                PARTITION BY CONCAT(per.FirstName, ' ', per.LastName),
                YEAR(soh.OrderDate)
                ORDER BY
                    MONTH(soh.OrderDate) RANGE BETWEEN UNBOUNDED PRECEDING
                    AND CURRENT ROW
            ) "w roku narastająco"
        FROM
            Sales.SalesOrderHeader soh
            JOIN Sales.SalesPerson sp ON soh.SalesPersonID = sp.BusinessEntityID
            JOIN HumanResources.Employee E ON sp.BusinessEntityID = E.BusinessEntityID
            JOIN Person.Person per ON per.BusinessEntityID = E.BusinessEntityID
    ) AS Sprzedawcy
ORDER BY
    1, 2, 3;
```

	Imię i nazwisko	Rok	Miesiac	w miesiacu	w roku	w roku narastająco	obecny i poprzedni miesiac
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	1
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

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

```
SELECT
    CONCAT(P.FirstName, ' ', P.LastName) 'Full Name',
    RANK() OVER (ORDER BY SUM(SOD.OrderQty) DESC) 'rank',
    DENSE_RANK() OVER (ORDER BY SUM(SOD.OrderQty) DESC) 'dense_rank'
FROM
    Sales.SalesOrderHeader SOH
    JOIN SAles.SalesOrderDetail SOD ON SOH.SalesOrderID = SOD.SalesOrderID
    JOIN Sales.Customer C ON SOH.CustomerID = C.CustomerID
    JOIN Person.Person P ON C.PersonID = P.BusinessEntityID
GROUP BY    CONCAT(P.FirstName, ' ', P.LastName);
```

	Full Name	rank	dense_rar
1	Reuben D'sa	1	1
2	Kevin Liu	2	2
3	Marcia Sultan	3	3
4	Holly Dickson	4	4
5	Mandy Vance	5	5
6	Richard Lum	6	6
7	Della Demott Jr	7	7
8	Sandra Maynard	8	8
9	Anton Kirilov	9	9
10	Ryan Calafato	10	10
11	John Evans	11	11
12	Yale Li	12	12
13	Helen Dennis	13	13
14	Margaret Vanderkamp	14	14
15	Lola McCarthy	15	15

	Full Name	rank	dense_rank
19001	Timothy Cook	16534	366
19002	Savannah Murphy	16534	366
19003	Brianna Bryant	16534	366
19004	Mitchell Anand	16534	366

Rank – przypisuje pozycję w rankingu z przerwami w numeracji – jeżeli dwie wartości są równe otrzymują tę samą pozycję

DenseRank – bez przerw w numeracji, równe wartości również dostaną tę samą pozycję.

Czyli Dense_rank nie zostawia dziur w sekwencji po remisach, podczas gdy RANK je tworzy.

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

```
-- zadanie 2.4
SELECT
    produkt 'produkt',
    DENSE_RANK() OVER( ORDER BY srednia DESC) 'miejsce',
    CAST(srednia AS decimal(10,2)) 'Srednia',
    CASE NTILE(3) OVER (ORDER BY srednia DESC)
        WHEN 1 THEN 'najlepiej' WHEN 2 then 'srednio' WHEN 3 THEN 'najslabiej'
    END 'Grupa'
FROM
    (
        SELECt DISTINCT
            P.Name 'produkt',
            AVG(CAST(SOD.OrderQty AS FLOAT)) OVER (PARTITION BY SOD.ProductID) 'srednia'
        FROM
            Sales.SalesOrderDetail SOD
            JOIN Production.Product P ON SOD.ProductID = P.ProductID
    ) AS prd;
```

	produkt	miejsce	Srednia	Grupa
72	LL Touring Frame - Yellow, 62	72	2.68	najlepiej
73	Touring-3000 Yellow, 44	73	2.68	najlepiej
74	LL Mountain Frame - Silver, 52	74	2.67	najlepiej
75	Mountain-100 Silver, 38	75	2.66	najlepiej
76	LL Mountain Front Wheel	76	2.66	najlepiej
77	Road-450 Red, 52	77	2.66	najlepiej
78	Touring-2000 Blue, 54	78	2.66	najlepiej
79	LL Road Frame - Red, 60	79	2.66	najlepiej
80	HL Touring Frame - Blue, 60	80	2.65	najlepiej
81	LL Road Frame - Red, 44	81	2.65	najlepiej
82	Touring-3000 Yellow, 62	82	2.65	najlepiej
83	LL Road Frame - Black, 52	83	2.63	najlepiej
84	LL Mountain Frame - Black, 42	84	2.61	najlepiej
85	Racing Socks, M	85	2.61	najlepiej
86	Mountain-100 Silver, 44	86	2.58	najlepiej
87	Road-350-W Yellow, 40	87	2.58	najlepiej
88	Road-450 Red, 58	88	2.58	najlepiej
89	Touring-1000 Yellow, 60	89	2.57	najlepiej
90	Hydration Pack - 70 oz.	90	2.57	srednio
91	HL Road Pedal	91	2.56	srednio
92	Touring-1000 Blue, 60	92	2.55	srednio
93	ML Road Frame-W - Yellow, 48	93	2.54	srednio
94	Mountain-100 Black, 48	94	2.53	srednio
95	Road-650 Red, 52	95	2.53	srednio
96	LL Bottom Bracket	96	2.52	srednio

NTILE to funkcja funkcja, która dzieli zbiór wyników na określoną liczbę możliwie równych grup. Jeżeli liczba wierszy nie dzieli się równo przez n to grupy początkowo mogą mieć o jeden wiersz więcej. W naszym przypadku tworzymy 3 grupy produktów, w zależności od średniej liczby sprzedanych produktów. W naszym przykładzie do najlepszej grupy zostały przyporządkowane produkty ze średnią wynoszącą nawet 2.57. Oznacza to, że jest stosunkowo niewiele produktów, które sprzedają się w dużej liczbie sztuk, przeglądając tabele widzimy, że różnice są niewielkie. Liczba sztuk, przybiera jedynie większe wartości na samym początku przy produktach typu rękawiczki, kaski.

Zad. 3. Ocena jakości danych – profilowanie danych

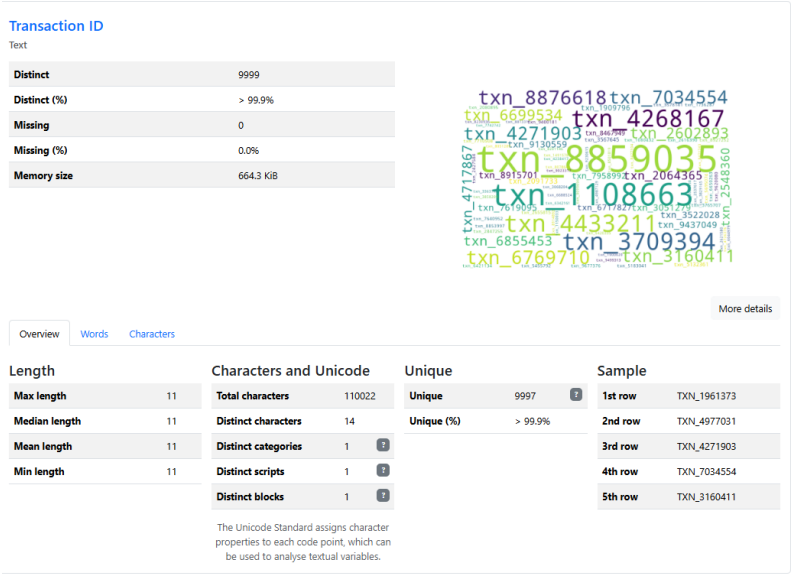
Dane

Plik dane_lista3.csv zawierają 10002 wierszy i mają 8 kolumn:

Transaction ID: String – identyfikator transakcji w sklepie
Item: String – nazwa sprzedanego produktu
Quantity: Int – Liczba sztuk sprzedanego produktu
Total Spent: Float – Całkowita kwota danej transakcji
Payment Method: String – metoda płatności (Karta kredytowa, gotówka)
Location: String – („In-Store", „Takeaway") – rodzaj danego zamówienia

Profilowanie kolumn

- Transaction ID



  - Każda transakcja ma przypisany identyfikator, występują jedynie 3 zduplikowane wartości co oznacza, że wskaźnik unikalności wynosi >99.9%
  - Identyfikatory mają jednolitą długość 11 znaków, co zważywszy na przybierane wartości sugeruje uporządkowany system generowania identyfikatorów
  - Jest to jedyna kolumna, w tabeli, która nadaje się na klucz kandydujący.
  - Identyfikator transakcji jest odpowiednio zorganizowany, TXN_xxxxxxx i żaden rekord nie odstaje od tej reguły

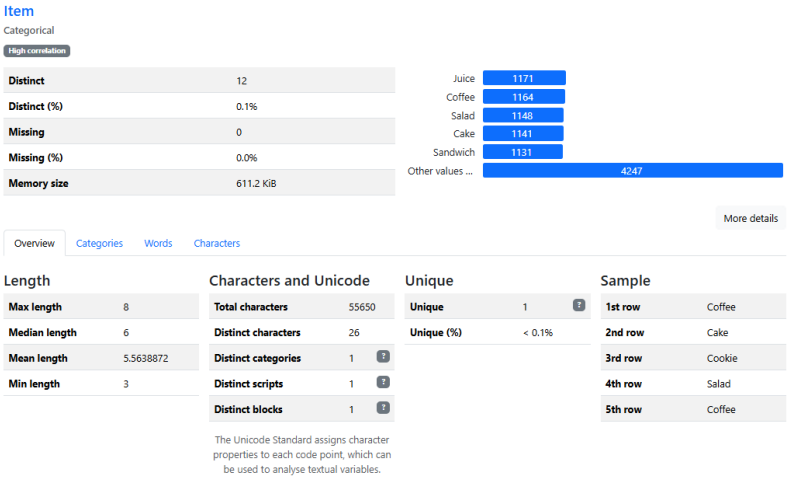
```
df[df['Transaction ID'].duplicated(keep=False)].sort_values('Transaction ID')
```

0.0s Python

	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
2216	TXN_1108663	Juice	3.0	3.0	9.0	Cash	Takeaway	2023-08-16
7033	TXN_1108663	Cookie	2.0	1.0	2.0	Digital Wallet	UNKNOWN	NaT
2205	TXN_8859035	Cake	3.0	3.0	9.0	Digital Wallet	Takeaway	2023-01-19
7013	TXN_8859035	Cake	3.0	3.0	9.0	Digital Wallet	Takeaway	2023-01-19
8004	TXN_8859035	Cake	3.0	3.0	9.0	Digital Wallet	Takeaway	2023-01-19

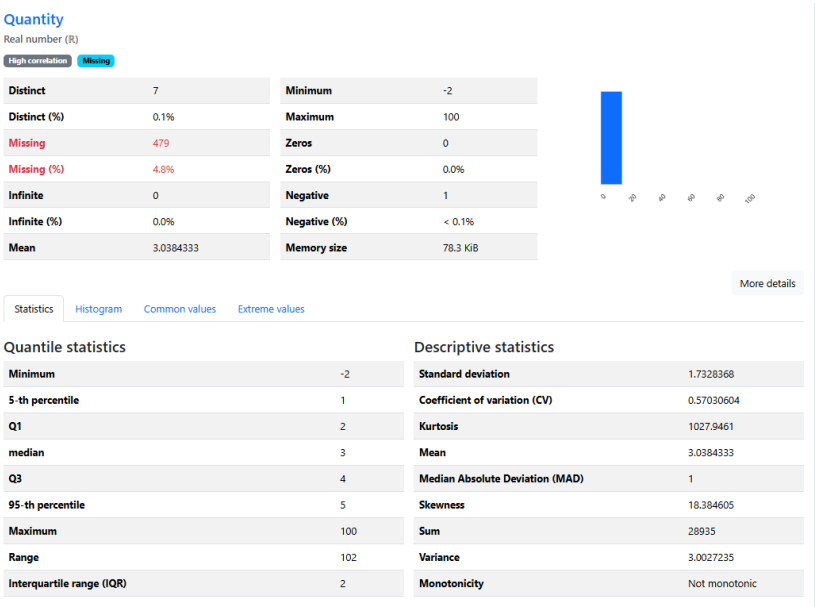
- 3krotne powtórzenie TransactionID, najprawdopodobniej miało dotyczyć tej samej transakcji zważywszy na te same produkty, liczbe sztuk ID i datę.

- **Item**



- Istnieje jedynie 12 unikalnych wartości w kolumnie Item. Należą do nich między innymi ‘ERROR’, ‘UNKNOWN’ oraz jeden rekord z wartością ‘Coffe’ (zamiast ‘Coffee’). Świadczy to o tym, że liczba produktów w sklepie jest ograniczona
- Wszystkie rekordy opisujące transakcje są skategoryzowane – 0% missing.
- Błędne wartości ERROR oraz UNKNOWN stanowią 6.4% całości, istnieje możliwość odtworzenia niektórych nazw produktów po cenie.

- **Quantity**



Quantity

Real number (R)

High correlation | Missing

Distinct	7	**Minimum**	-2
Distinct (%)	0.1%	**Maximum**	100
Missing	479	**Zeros**	0
Missing (%)	4.8%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	1
Infinite (%)	0.0%	**Negative (%)**	< 0.1%
Mean	3.0384333	**Memory size**	78.3 KiB

More details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

Minimum	-2
5-th percentile	1
Q1	2
median	3
Q3	4
95-th percentile	5
Maximum	100
Range	102
Interquartile range (IQR)	2

Descriptive statistics

Standard deviation	1.7328368
Coefficient of variation (CV)	0.57030604
Kurtosis	1027.9461
Mean	3.0384333
Median Absolute Deviation (MAD)	1
Skewness	18.384605
Sum	28935
Variance	3.0027235
Monotonicity	Not monotonic

- W kolumnie pojawiają się wybrakowane wartości dla 4.8% wierszy.
- W danych jest rekord z ujemną wartością: -2, jest też rekord z bardzo dużą wartością (100), która znacznie wykracza poza zakres wartości przyjmowany przez kolumnę (poprzednią największą wartością jest 5)
- Istnieje możliwość odtworzenia niektórych brakujących wartości na podstawie całkowitej kwoty zamówienia i ceny jednostkowej.

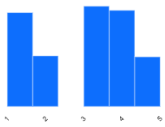
- **Price Per Unit**

Price Per Unit

Real number (ℝ)

High correlation Missing

Distinct	6	**Minimum**	1
Distinct (%)	0.1%	**Maximum**	5
Missing	533	**Zeros**	0
Missing (%)	5.3%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	2.9499947	**Memory size**	78.3 KiB



More details

Statistics | Histogram | Common values | Extreme values

Quantile statistics

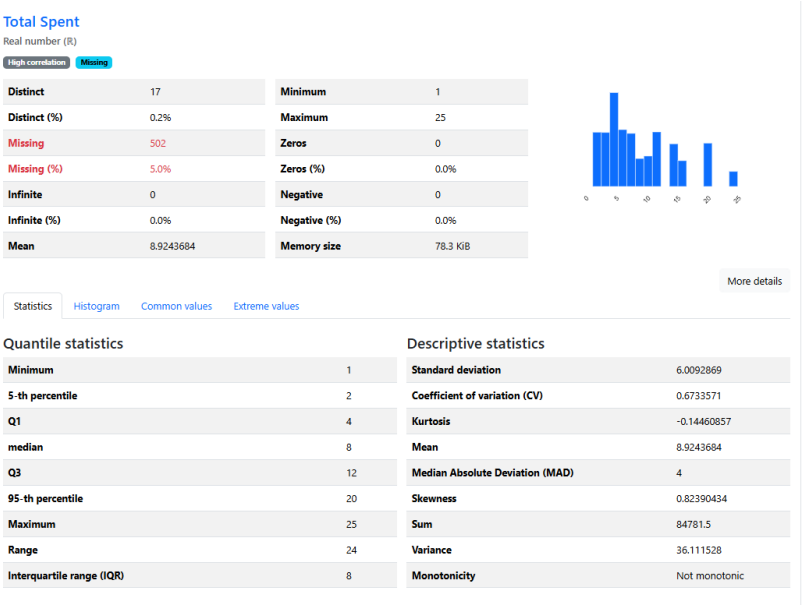
Minimum	1
5-th percentile	1
Q1	2
median	3
Q3	4
95-th percentile	5
Maximum	5
Range	4
Interquartile range (IQR)	2

Descriptive statistics

Standard deviation	1.2783156
Coefficient of variation (CV)	0.43332811
Kurtosis	-1.1553162
Mean	2.9499947
Median Absolute Deviation (MAD)	1
Skewness	0.0045662391
Sum	27933.5
Variance	1.6340909
Monotonicity	Not monotonic

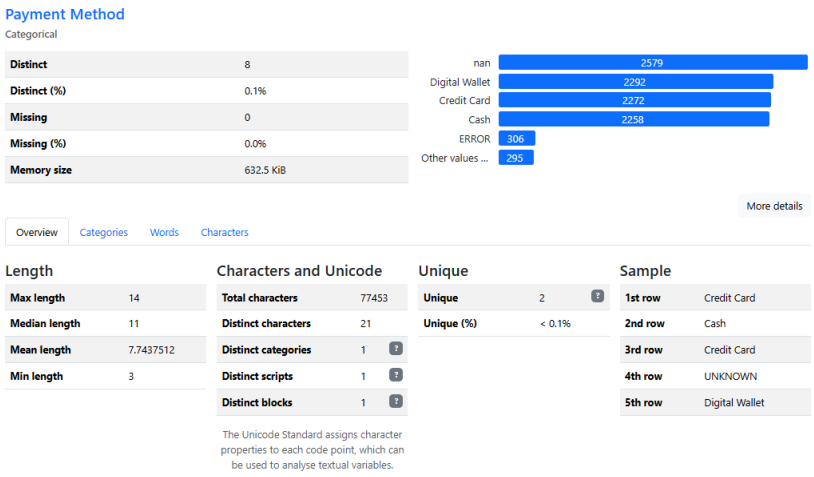
- Kolumna składa się z 6 unikalnych wartości, 1 z nich jest liczbą z przecinkiem (1.5). Wszystkie ceny jednostkowe mieszczą się w przedziale 1-5, co wydaje się być prawidłowym zakresem
- 533 (5.3%) brakujących rekordów. Dominują transakcje z produktami o wartości 3 – 4, reszta jest rozłożona równomiernie. W kolumnie ciężko doszukać się anomalii, dane są jedynie częściowo wybrakowane.

- **Total Spent**



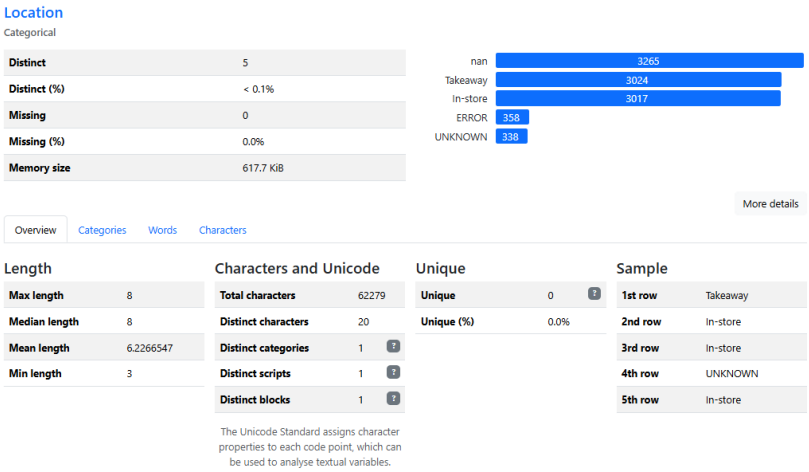
- W kolumnie jest 5% wybrakowanych wartości i 17 unikalnych.
- Ogólnie kolumna zdaje się być kolumną wtórną, zdecydowana większość wartości TotalSpent pochodzi z mnożenia Quantity * PricePerUnit (8550; do TotalPrice nie jest doliczana żadna marża, promocja itp.). Zdażają się jednak rekordy, w których brakuje Quantity lub PricePerUnit i posiadają wartość.
- Maksymalna wartość 25, przy maksymalnej liczbie kupionych produktów 5 i cenie 5 wydaje się nie być zaburzona, tak samo jak minimalna – 1.
- Dominują jednak wartości niższe pomiędzy 1 a 7

- **Payment Method**



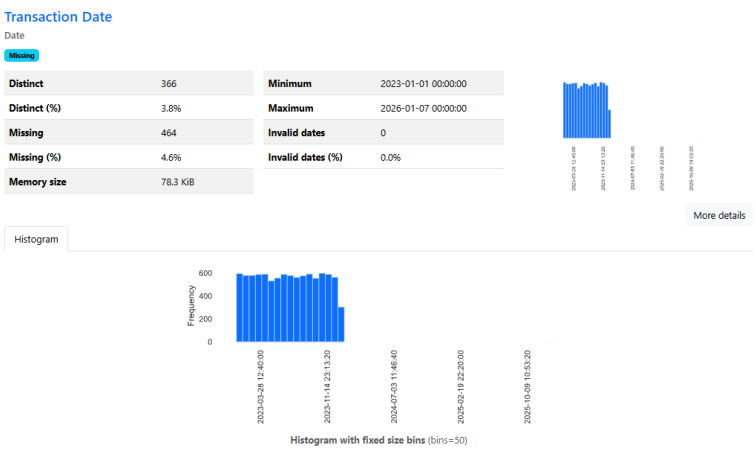
- Bardzo dużo niepoprawnych wartości, nan – 25.8% w połączeniu z UNKNOWN i ERROR 31.8%. Najprawdopodobniej odrzucone lub błędne płatności są również zapisywane.
- Poprawne sposoby płatności pojawiają się jednak na ogół tak samo często.
- W tabeli znajdują się również dane wprowadzone z błędem ‘Digital Walle’ i ‘CreditCard’

- **Location**



- Ponownie bardzo dużo niepoprawnych wartości. Liczba rekordów z ‘UNKNOWN’, ‘nan’ lub ‘ERROR’ 39.6%.
- Dwie poprawne kategorie to In-Store i Takeaway i pojawiają się tak samo często.

- **Transaction Date**



- Transakcji pochodzą z okresu od 2023-01 do końca roku.
- Maksymalny data jest błędna pochodzi z 2026
- Rozkład liczby transakcji na przestrzeni roku wydaje się być równomierny, bardzo niewielkie spadki w maju i grudniu.

Profilowanie międzykolumnowe

Transaction ID -> Wszystkie kolumny

- Jak wspomniano wyżej TransactionID jest unikalne (oprócz jedynie 3 zapewne błędnych wartości) i niewybrakowane, określa więc wartości we wszystkich innych kolumnach

Quantity i Price Per Unit -> Total Spent

- Oczywiście istnieje zależność funkcyjna we wszystkich niewybrakowanych rekordach. Suma transakcji jest obliczana na podstawie mnożenia. W rekordach wybrakowanych zależnosc funkcyjna jest zaburzona. Wygląda to następująco:

```
Niespójności w zależności (Quantity, Price Per Unit) -> Total Spent (dla jednej pary występuje więcej niż jedna wartość Total Spent):
Quantity  Price Per Unit
1.0       NaN               6
2.0       NaN               6
3.0       NaN               6
4.0       NaN               6
5.0       NaN               6
NaN       1.0               5
          1.5               5
          2.0               5
          3.0               5
          4.0               5
          5.0               5
          NaN              13
Name: Total Spent, dtype: int64
```

Name -> Price Per Unit

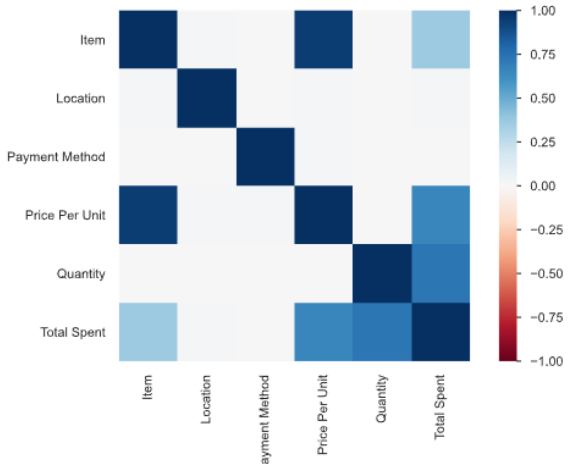
- Do każdego produktu przypisana jest cena jednostkowa, więc istnieje zależność funkcyjna w niewybrakowanych rekordach.

```
Niespójności w zależności Item -> Price Per Unit (występuje więcej niż jedna cena dla danego produktu):
Item
ERROR      6
UNKNOWN    6
nan        6
Name: Price Per Unit, dtype: int64
```

Payment Method, Location, Transaction Date

- Nie ma oczywistych zależności między tymi kolumnami

Macierz korelacji



Częściowo omówiona przy zależnościach funkcyjnych. Widoczna jest dodatkowo słaba korelacja pomiędzy Item i Total Spent, ta informacja nie jest jednak specjalnie użyteczna. Nie ma bowiem produktu, który jest zazwyczaj kupowany w jakiejś konkretnej liczbie sztuk.

Liczba zakupów poszczególnych produktów przy ilościach od 1 do 5:

Quantity	1.0	2.0	3.0	4.0	5.0
Item					
Cake	209	232	200	208	246
Coffe	0	0	1	0	0
Coffee	217	212	215	209	248
Cookie	209	231	179	218	202
ERROR	48	58	54	51	63
Juice	215	231	235	224	219
Salad	219	227	195	218	236
Sandwich	213	205	222	194	236
Smoothie	193	211	211	212	225
Tea	181	228	205	198	222
UNKNOWN	64	72	65	66	54
nan	53	66	69	65	62

Podsumowanie

Identyfikatory transakcji są generalnie spójne i uporządkowane, co umożliwia ich wykorzystanie jako klucza głównego. Głównym problemem są brakujące/błędne wartości, których procent wynosi nawet i 40%.

Część z nich da się jednak naprawić odtworzyć – nieliczne literówki można poprawić ręcznie, brakujący element z równania Quantity * Product Price = Total Spent wywnioskować (istnieje 1398 takich rekordów co oznacza, że da się odtworzyć 13.9% wartości).

```
cols_to_check = ['Quantity', 'Total Spent', 'Price Per Unit']
# Suma nulli w każdym wierszu
null_counts = df[cols_to_check].isnull().sum(axis=1)
# Wiersze z dokładnie jednym nullem
rows_with_exactly_one_null = (null_counts == 1)
num_rows_with_one_null = rows_with_exactly_one_null.sum()
total_rows = len(df)
percentage = (num_rows_with_one_null / total_rows) * 100
print(f"Procent rekordów, w których dokładnie jeden z atrybutów {cols_to_check} jest null: {percentage:.2f}%")
```

0.0s Python

```
Procent rekordów, w których dokładnie jeden z atrybutów ['Quantity', 'Total Spent', 'Price Per Unit'] jest null: 13.98%
```

Istnieje również możliwość odtworzenia, niektórych wartości z kolumny PricePerUnit, produkt charakteryzuje się ceną, więc możemy w brakujących miejscach ją wywnioskować.

```python
item_price_mapping = df.dropna(subset=['Price Per Unit', 'Item']).groupby('Item')['Price Per Unit'].unique()
consistent_price_items = {item: prices[0] for item, prices in item_price_mapping.items() if len(prices) == 1}
missing_price_records = df[df['Price Per Unit'].isnull() & df['Item'].notna()]
recoverable_records = missing_price_records[missing_price_records['Item'].isin(consistent_price_items.keys())]

print(f"Liczba produktów z unikalną, stałą ceną: {len(consistent_price_items)}")
print(f"Liczba rekordów z brakującą ceną jednostkową, które można uzupełnić: {len(recoverable_records)}")
print(f"Procent wszystkich rekordów z brakującą ceną, które można uzupełnić: {len(recoverable_records)/len(missing_price_records)*100:.2f}% (jeśli są takie rekordy)")
print(f"Procent wszystkich rekordów: {len(recoverable_records)/len(df)*100:.2f}%")
```

✓ 0.0s

```
Liczba produktów z unikalną, stałą ceną: 9
Liczba rekordów z brakującą ceną jednostkową, które można uzupełnić: 479
Procent wszystkich rekordów z brakującą ceną, które można uzupełnić: 89.87% (jeśli są takie rekordy)
Procent wszystkich rekordów: 4.79%
```

Istnieje również możliwość przeanalizowania tej zależności w drugą stronę, istnieją produkty, które jako jedyne mają daną cenę, tak jak Salad, jako jedyny produkt kosztuje 5. Możemy zatem uzupełnić część nazw produktów na podstawie takich unikalnych cen.

```python
df_clean = df.copy()
df_clean.loc[df_clean['Item'].isin(['UNKNOWN', 'ERROR', 'nan']), 'Item'] = None

#Unikalne ceny dla każdego produktu
price_item_mapping = df_clean.dropna(subset=['Price Per Unit', 'Item']).groupby('Price Per Unit')['Item'].unique()

display(price_item_mapping)

# Sprawdzenie, które ceny mają tylko jeden produkt przypisany
unique_price_to_item = {price: items[0] for price, items in price_item_mapping.items()
                        if len(items) == 1 and not any(item in ['nan', 'UNKNOWN', 'ERROR', None] for item in items)}

missing_item_records = df_clean[
    (df_clean['Item'].isnull() | df_clean['Item'].isin(['UNKNOWN', 'ERROR'])) &
    df_clean['Price Per Unit'].notna()
]

recoverable_records = missing_item_records[missing_item_records['Price Per Unit'].isin(unique_price_to_item.keys())]

print(f"Liczba rekordów z brakującą/niepoprawną nazwą produktu, które można uzupełnić: {len(recoverable_records)}")
print(f"Procent wszystkich rekordów: {len(recoverable_records)/len(df)*100:.2f}%")
```

✓ 0.0s

```
Price Per Unit
1.0               [Cookie]
1.5                  [Tea]
2.0        [Coffee, Coffe]
3.0          [Cake, Juice]
4.0    [Smoothie, Sandwich]
5.0                [Salad]
Name: Item, dtype: object

Liczba rekordów z brakującą/niepoprawną nazwą produktu, które można uzupełnić: 349
Procent wszystkich rekordów: 3.49%
```

Inne brakujące dane będzie jednak ciężko odtworzyć nie posiadając odpowiedniej dodatkowej wiedzy.

Pojawianie się literówek, wiele wybrakowanych wartości w kolumnie Location sugeruje ręczną konieczność wprowadzania danych, być może z pewnymi lukami, umożliwiającymi na wprowadzenie niezweryfikowanych danych.