

Hurtownie Danych – laboratorium Lista 3

Spis Treści

Zadanie 1. Wykorzystanie funkcji grupujących.....	2
Zadanie 2. Wykorzystanie funkcji okienkowych.....	7
Zadanie 3. Ocena jakości danych.	15
Kompletność Danych.....	16
Spójność:.....	21
Poprawność Danych:	24
Unikalność Danych:.....	26
Integralność i dokładność danych:.....	27
Ocena jakości danych.....	29
Możliwe sposoby poprawienia jakości danych:	30
Kompletność:	30
Spójność:	31
Poprawność:.....	31
Unikalność:	31
Integralność:.....	32
Wnioski	33
Zadanie 1.	33
Zadanie 2.	34
Zadanie 3.	35

Zadanie 1. Wykorzystanie funkcji grupujących

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

```
SELECT CONCAT(p.LastName, ' ', p.FirstName) AS Klient, Year(soh.OrderDate) AS Rok, CAST(SUM(TotalDue) AS DECIMAL(10,2)) AS SumaTransakcji
FROM Sales.SalesOrderHeader soh
JOIN Sales.Customer c ON soh.CustomerID = c.CustomerID
JOIN Person.Person p ON c.CustomerID = p.BusinessEntityID
GROUP BY ROLLUP(YEAR(soh.OrderDate), CONCAT(p.LastName, ' ', p.FirstName))
ORDER BY 1,2;
```

```
SELECT CONCAT(p.LastName, ' ', p.FirstName) AS Klient, Year(soh.OrderDate) AS Rok, CAST(SUM(TotalDue) AS DECIMAL(10,2)) AS SumaTransakcji
FROM Sales.SalesOrderHeader soh
JOIN Sales.Customer c ON soh.CustomerID = c.CustomerID
JOIN Person.Person p ON c.CustomerID = p.BusinessEntityID
GROUP BY CUBE(YEAR(soh.OrderDate), CONCAT(p.LastName, ' ', p.FirstName))
ORDER BY 1,2;
```

```
SELECT CONCAT(p.LastName, ' ', p.FirstName) AS Klient, Year(soh.OrderDate) AS Rok, CAST(SUM(TotalDue) AS DECIMAL(10,2)) AS SumaTransakcji
FROM Sales.SalesOrderHeader soh
JOIN Sales.Customer c ON soh.CustomerID = c.CustomerID
JOIN Person.Person p ON c.CustomerID = p.BusinessEntityID
GROUP BY GROUPING SETS(YEAR(soh.OrderDate), CONCAT(p.LastName, ' ', p.FirstName), (YEAR(soh.OrderDate), CONCAT(p.LastName, ' ', p.FirstName)))
ORDER BY 1,2;
```

	Klient	Rok	SumaTransakcji
1	NULL	NULL	21764628.64
2	NULL	2011	2549441.31
3	NULL	2012	4860450.59
4	NULL	2013	8278860.21
5	NULL	2014	6075876.53
6	Adams Aaron	2014	2632.04
7	Adams Adam	2014	44.18
8	Adams Alex	2014	44.18
9	Adams Angel	2013	36.02
10	Adams Angel	2014	38.65
11	Adams Carlos	2011	3953.99
12	Adams Carlos	2013	987.28
13	Adams Con...	2014	2606.96
14	Adams Elijah	2014	5.51

Klienci którzy wydali najwięcej w ciągu 1 roku:

```
SELECT CONCAT(p.LastName, ' ', p.FirstName) AS Klient, Year(soh.OrderDate) AS Rok, CAST(SUM(TotalDue) AS DECIMAL(10,2)) AS SumaTransakcji
FROM Sales.SalesOrderHeader soh
JOIN Sales.Customer c ON soh.CustomerID = c.CustomerID
JOIN Person.Person p ON c.CustomerID = p.BusinessEntityID
GROUP BY GROUPING SETS((YEAR(soh.OrderDate), CONCAT(p.LastName, ' ', p.FirstName)))
ORDER BY 3 DESC;
```

	Klient	Rok	SumaTransakcji
1	Chen Jenny	2013	11630.79
2	Yang Jenny	2013	11127.47
3	Kelly Chase	2013	8112.29
4	Bradley Elizabeth	2013	8090.09
5	Jones Taylor	2013	8086.33
6	Clark Hannah	2013	8064.35
7	Gonzales Carson	2013	8059.64
8	Lee Hannah	2013	8025.56
9	Sharma Colleen	2013	8020.79
10	Andersen Colleen	2013	8012.30
11	Miller Madison	2013	7999.46
12	Diaz Sarah	2013	7995.07
13	Hughes Sarah	2013	7979.48
14	Smith Taylor	2013	7977.35
15	Chander Kelli	2013	7976.63
16	Tang Clayton	2013	7972.58
17	She Colleen	2013	7968.59
18	White Madison	2013	7968.44
19	Tang Colleen	2013	7967.99
20	Perry Xavier	2013	7956.38

Wnioski:

- Najwięcej w trakcie jednego roku wydały Jenny Chen, Jenny Yang oraz Kelly Chase, wszystkie w trakcie 2013 roku.
- 2013 rok dominuje wśród lat z największymi zakupami dokonywanymi przez klientów. Jest to spowodowane gwałtownym wzrostem firmy w 2013 roku oraz niekompletnością danych dla 2014 roku (dane kończą się w czerwcu 2014).

- Największe sumy transakcji dla klienta według roku to:

```

SELECT CONCAT(p.LastName, ' ', p.FirstName) AS Klient, Year(soh.OrderDate) AS Rok, CAST(SUM(TotalDue) AS DECIMAL(10,2)) AS SumaTransakcji,
ROW_NUMBER() OVER(ORDER BY SUM(TotalDue) DESC) AS NajwiekszaSumaOgolnie,
ROW_NUMBER() OVER(PARTITION BY YEAR(soh.OrderDate) ORDER BY SUM(TotalDue) DESC) AS NajwiekszaSumaRok
FROM Sales.SalesOrderHeader soh
JOIN Sales.Customer c ON soh.CustomerID = c.CustomerID
JOIN Person.Person p ON c.CustomerID = p.BusinessEntityID
GROUP BY CONCAT(p.LastName, ' ', p.FirstName), Year(soh.OrderDate)
ORDER BY NajwiekszaSumaRok, Rok;

```

	Klient	Rok	SumaTransakcji	NajwiekszaSumaOgolnie	NajwiekszaSumaRok
1	Adams Carlos	2011	3953.99	619	1
2	Xu Willie	2012	4821.25	451	1
3	Chen Jenny	2013	11630.79	1	1
4	Rai Christy	2014	6864.70	46	1

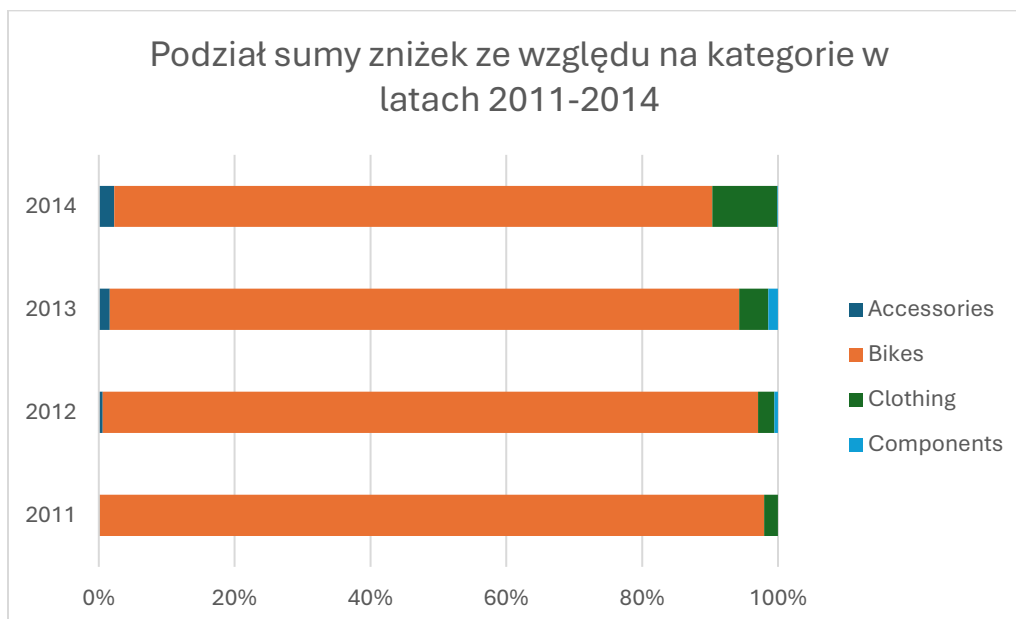
- 2011 – Carlos Adams – 3953.99\$ - 619 miejsce ogólnie
- 2012 – Willie Xu – 4821.25\$ - 451 miejsce ogólnie
- 2013 – Jenny Chen – 11630.79\$ - 1 miejsce ogólnie
- 2014 – Christy Rai – 6864.70\$ - 46 miejsce ogólnie

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

```
SELECT pc.Name AS Kategoria, p.Name AS Produkt, YEAR(soh.OrderDate) AS Rok, SUM(UnitPriceDiscount * OrderQty * UnitPrice) AS SumaZnizek
FROM Sales.SalesOrderDetail sod
JOIN Sales.SalesOrderHeader soh ON sod.SalesOrderID = soh.SalesOrderID
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
GROUP BY GROUPING SETS((pc.Name, p.Name, YEAR(soh.OrderDate)))
ORDER BY 1, 2, 3;
```

	Kategoria	Produkt	Rok	SumaZnizek
1	Accessories	All-Purpose Bike Stand	2013	0.00
2	Accessories	All-Purpose Bike Stand	2014	0.00
3	Accessories	Bike Wash - Dissolver	2013	83.8772
4	Accessories	Bike Wash - Dissolver	2014	27.7213
5	Accessories	Cable Lock	2012	20.30
6	Accessories	Cable Lock	2013	3.48
7	Accessories	Fender Set - Mountain	2013	0.00
8	Accessories	Fender Set - Mountain	2014	0.00
9	Accessories	Hitch Rack - 4-Bike	2013	1950.444

Sum of SumaZnizek	Column Labels				
Row Labels	Accessories	Bikes	Clothing	Components	Grand Total
2011	4.293	4344.6957	90.9593	0	4439.948
2012	1061.4872	180073.3397	4498.9691	961.8196	186595.6156
2013	4779.6779	277731.7099	12847.2903	4212.9922	299571.6703
2014	842.5712	32490.9079	3527.2866	39.9266	36900.6923
Grand Total	6688.0293	494640.6532	20964.5053	5214.7384	527507.9262



Row Labels	Sum of SumaZnizek
▣ Touring-1000 Yellow, 60	60526.5306
Bikes	60526.5306
▣ Touring-1000 Yellow, 46	50653.8584
Bikes	50653.8584
▣ Touring-1000 Yellow, 50	27273.7608
Bikes	27273.7608

Wnioski:

- Największe sumy zniżek odnotowano na rowerach. Dominują one w każdym roku.
- Największą sumę zniżek według produktu osiągnięto na rowerze Touring-1000 Yellow, 60 (60526\$)
- W 2011 roku nie było żadnej zniżki na komponenty (lub nikt nie kupił przecenionego towaru).
- Łączna suma zniżek wyniosła 527507.93\$.

Zadanie 2. Wykorzystanie funkcji okienkowych

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.

Osobna kwerenda dla każdej kategorii:

```
WITH SprzedazeKategorii AS (
SELECT pc.Name AS Kategoria, YEAR(soh.OrderDate) AS Rok, SUM(sod.LineTotal) AS kwotaSprzedazy
FROM Sales.SalesOrderDetail sod
JOIN Sales.SalesOrderHeader soh ON sod.SalesOrderID = soh.SalesOrderID
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
WHERE pc.Name = 'Bikes'
GROUP BY pc.Name, YEAR(soh.OrderDate)
)
SELECT Kategoria, Rok,
CAST((100 * kwotaSprzedazy / SUM(kwotaSprzedazy) OVER ())) AS DECIMAL(10,2)) AS ProcentSprzedazy
FROM SprzedazeKategorii
GROUP BY Kategoria, Rok, kwotaSprzedazy;
```

Kategoria	Rok	ProcentSprzedazy
Bikes	2013	38.32
Bikes	2014	18.44
Bikes	2011	12.62
Bikes	2012	30.62

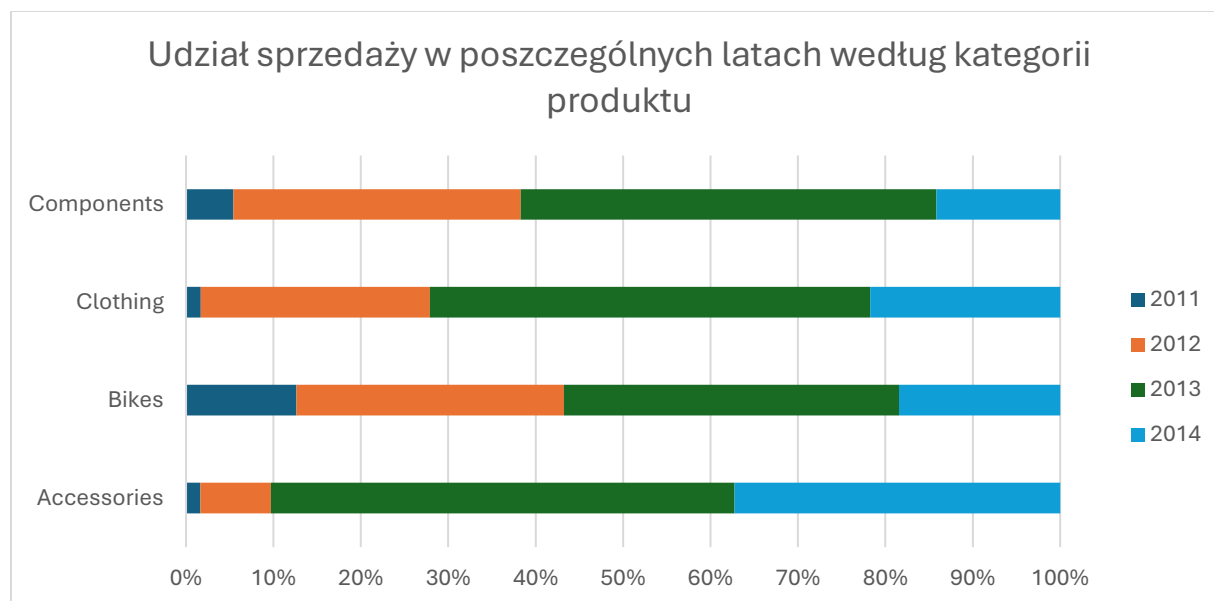
Kategoria	Rok	ProcentSprzedazy
Accessories	2013	53.06
Accessories	2014	37.25
Accessories	2011	1.64
Accessories	2012	8.05

Kategoria	Rok	ProcentSprzedazy
Clothing	2013	50.35
Clothing	2014	21.75
Clothing	2011	1.7
Clothing	2012	26.2

Kategoria	Rok	ProcentSprzedazy
Components	2013	47.56
Components	2014	14.15
Components	2011	5.42
Components	2012	32.88

Lub w jednej kwerendzie:

```
WITH SprzedazeKategorii AS (
SELECT pc.Name AS Kategoria, YEAR(soh.OrderDate) AS Rok, SUM(sod.LineTotal) AS kwotaSprzedazy
FROM Sales.SalesOrderDetail sod
JOIN Sales.SalesOrderHeader soh ON sod.SalesOrderID = soh.SalesOrderID
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
GROUP BY pc.Name, YEAR(soh.OrderDate)
)
SELECT Kategoria, Rok,
CAST((100 * kwotaSprzedazy / SUM(kwotaSprzedazy) OVER (PARTITION BY Kategoria))) AS DECIMAL(4,2)) AS ProcentSprzedazy
FROM SprzedazeKategorii
GROUP BY Kategoria, Rok, kwotaSprzedazy;
```



Wnioski:

- Największą sprzedaż dla każdej kategorii produktu osiągnięto w 2013 roku.
- Największy wzrost wartości sprzedaży w stosunku do poprzedniego roku osiągnięto dla ubrań pomiędzy 2011 i 2012 rokiem. Wynosił on 1541%.
- Sprzedaż rowerów nie miała aż tak drastycznych zmian w żadnym z lat. Jest to kategoria która sprzedawała się najrówniej na przestrzeni lat.

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.

```

WITH OrdersByMonth AS (
    SELECT
        CONCAT(p.FirstName, ' ', p.LastName) AS Sprzedawca,
        YEAR(soh.OrderDate) AS Rok,
        MONTH(soh.OrderDate) AS Miesiac,
        COUNT(soh.SalesOrderID) AS LiczbaZamowien
    FROM Sales.SalesOrderHeader soh
    JOIN Person.Person p ON soh.SalesPersonID = p.BusinessEntityID
    GROUP BY
        CONCAT(p.FirstName, ' ', p.LastName),
        YEAR(soh.OrderDate),
        MONTH(soh.OrderDate)
)
SELECT
    Sprzedawca,
    Rok,
    Miesiac,
    LiczbaZamowien AS LiczbaZamowienMiesiac,
    SUM(LiczbaZamowien) OVER (PARTITION BY Sprzedawca, Rok) AS LiczbaZamowienRok,
    SUM(LiczbaZamowien) OVER (PARTITION BY Sprzedawca, Rok ORDER BY Miesiac) AS LiczbaZamowienRokNarastajaco,
    -- Liczba zamówień w bieżącym i poprzednim miesiącu
    LiczbaZamowien + COALESCE(LAG(LiczbaZamowien) OVER(PARTITION BY Sprzedawca, Rok ORDER BY Miesiac), 0) AS LiczbaZamowienMiesiacIPopzedni
FROM OrdersByMonth obm
ORDER BY
    Sprzedawca,
    Rok,
    Miesiac;

```

	Sprzedawca	Rok	Miesiac	LiczbaZamowienMiesiac	LiczbaZamowienRok	LiczbaZamowienRokNarastajaco	LiczbaZamowienMiesiacIPopzedni
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



Wnioski:

- Oboje sprzedawcy z największą liczbą sprzedaży wykazali podobne wzorce sprzedażowe w każdym roku na przestrzeni lat 2012-2014.
- Największe sprzedaże Jillian Carson i Michael Blythe odnotowali w marcu 2014 roku oraz październiku 2011 roku.
- Michael Blythe dokonał większej ilości transakcji w 2011 roku, Jillian Carson w 2012 i 2013, a w 2014 roku dokonali bardzo podobnej liczby sprzedaży.

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

```
SELECT CONCAT(p.FirstName, ' ', p.LastName) AS Klient, SUM(sod.OrderQty) AS LiczbaProduktow,
       RANK() OVER(ORDER BY SUM(sod.OrderQty) DESC) AS Rank,
       DENSE_RANK() OVER(ORDER BY SUM(sod.OrderQty) DESC) AS DenseRank
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 CONCAT(p.FirstName, ' ', p.LastName);
```

	Klient	LiczbaProduktow	Rank	DenseRank
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

Różnica pojawia się w 29 wierszu tabeli ponieważ Johny Caprio oraz Richard Bready zakupili taką samą liczbę produktów (1578). Korzystając z Rank, Roger Harui zajął 29 miejsce (28 nie było), a korzystając z Dense Rank zajął 28 miejsce.

26	John Arthur	1579	26	26
27	Johnny Caprio	1578	27	27
28	Richard Bready	1578	27	27
29	Roger Harui	1561	29	28
30	Mary Gimmi	1545	30	29
31	Scott Culp	1532	31	30

Od tamtego czasu rangi różnią się, aż końcowo osiągają wartości 16534 (Rank) i 366 (Dense Rank).

16530	Ashley Griffin	2	11463	365
16531	Regina Ray	2	11463	365
16532	Grace Morris	2	11463	365
16533	Lucas Adams	2	11463	365
16534	James Yang	1	16534	366
16535	Miranda Butler	1	16534	366
16536	Ian Cooper	1	16534	366
16537	Anna Powell	1	16534	366
16538	Timothy Kelly	1	16534	366
16539	Thomas Hernandez	1	16534	366

Wnioski:

- 16533 osoby kupiły ponad jeden produkt. 2484 osoby (liczba wierszy – najwyższy Rank) kupiło dokładnie jeden produkt.
- Liczba unikalnych liczb zakupionych produktów wynosi 366.
- Najwięcej produktów zakupił Reuben D'sa (2737).

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

```
--d
SELECT p.Name, CAST(CAST(SUM(sod.OrderQty) AS FLOAT)/COUNT(*) AS DECIMAL(10,2)) AS SredniaLiczbaWZamowieniu,
NTILE(3) OVER(ORDER BY CAST(SUM(sod.OrderQty) AS FLOAT)/COUNT(*) DESC) AS GrupaSprzedazy
FROM Sales.SalesOrderDetail sod
JOIN Production.Product p ON sod.ProductID = p.ProductID
GROUP BY p.Name
ORDER BY 2 DESC;
```

	Name	SredniaLiczbaWZamowieniu	GrupaSprzedazy
1	Full-Finger Gloves, L	9.28	1
2	Classic Vest, S	6.23	1
3	Full-Finger Gloves, M	6.14	1
4	ML Headset	5.99	1
5	Mountain Bike Socks, M	5.89	1
6	Women's Mountain Shorts, S	5.09	1
7	Women's Tights, S	4.91	1
8	Women's Tights, L	4.81	1
9	Women's Mountain Shorts, L	4.59	1
10	Men's Bib-Shorts, M	4.58	1
11	Men's Sports Shorts, M	4.51	1
12	Short-Sleeve Classic Jersey, XL	4.27	1
13	Minipump	4.23	1
14	Cable Lock	4.18	1
15	Classic Vest, M	4.12	1

Pierwsza grupa produktów pod względem średniej ilości kupowanych razem w zamówieniu obejmuje produkty których sprzedaje się średnio więcej niż 2.573 i obejmuje 89 produktów.

	Name	SredniaLiczbaWZamowieniu	GrupaSprzedazy
85	Racing Socks, M	2.609	1
86	Mountain-100 Silver, 44	2.579	1
87	Road-350-W Yellow, 40	2.578	1
88	Road-450 Red, 58	2.575	1
89	Touring-1000 Yellow, 60	2.573	1
90	Hydration Pack - 70 oz.	2.571	2
91	HL Road Pedal	2.561	2

Druga grupa zaczyna się od 2.751 i kończy na 2.075 produktu na zamówienie obejmując 89 produktów.

	Name	SredniaLiczbaWZamowieniu	GrupaSprzedazy
176	Full-Finger Gloves, S	2.083	2
177	HL Mountain Frame - Silver, 42	2.078	2
178	Road-350-W Yellow, 42	2.075	2
179	LL Touring Frame - Yellow, 50	2.073	3
180	Touring Pedal	2.070	3
181	LL Headset	2.065	3
182	ML Road Rear Wheel	2.065	3
183	ML Mountain Frame - Black, 40	2.061	3

Wnioski:

- Produkt który średnio kupuje się w największej liczbie to Full-Finger Gloves L.
- N-Tile zaokrągliła wielkość początkowych grup w górę (mają po 89 wyników, a ostatnia ma 88).

Zadanie 3. Ocena jakości danych.

1. Przeanalizować, scharakteryzować i ocenić dane znajdujące się w pliku dane_lista3.csv wykorzystując wybrane oprogramowanie, np. Python, R, Matlab, Integration Services Project w Visual Studio, itp.

Dane w pliku dane_lista3.csv prezentują się w taki sposób.

1	Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
2	TXN_1961373	Coffee	2	2	4	Credit Card	Takeaway	9/8/2023
3	TXN_4977031	Cake	4	3	12	Cash	In-store	5/16/2023
4	TXN_4271903	Cookie	4	1	ERROR	Credit Card	In-store	7/19/2023
5	TXN_7034554	Salad	2	5	10	UNKNOWN	UNKNOWN	4/27/2023
6	TXN_3160411	Coffee	2	2	4	Digital Wallet	In-store	6/11/2023
7	TXN_2602893	Smoothie	5	4	20	Credit Card		3/31/2023
8	TXN_4433211	UNKNOWN	3	3	9	ERROR	Takeaway	10/6/2023
9	TXN_6699534	Sandwich	4	4	16	Cash	UNKNOWN	10/28/2023
10	TXN_4717867		5	3	15		Takeaway	7/28/2023
11	TXN_2064365	Sandwich	5	4	20		In-store	12/31/2023
12	TXN_2548360	Salad	5	5	25	Cash	Takeaway	11/7/2023
13	TXN_3051279	Sandwich	2	4	8	Credit Card	Takeaway	ERROR
14	TXN_7619095	Sandwich	2	4	8	Cash	In-store	5/3/2023
15	TXN_9437049	Cookie	5	1	5		Takeaway	6/1/2023
16	TXN_8915701	ERROR	2	1.5	3		In-store	3/21/2023
17	TXN_2847255	Salad	3	5	15	Credit Card	In-store	11/15/2023
18	TXN_3765707	Sandwich	1	4	4			6/10/2023
19	TXN_6769710	Juice	2	3	6	Cash	In-store	2/24/2023






















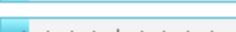
Dane są rozmieszczone w następujących kolumnach:

- TransactionID
- Item
- Quantity
- Price Per Unit
- Total Spent
- Payment Method
- Location
- Transaction Date

Kompletność Danych











Już na pierwszy rzut oka możemy zobaczyć, że dane są **niekompletne**, występują w nich puste komórki lub komórki w których występują wartości UNKNOWN i ERROR.

ITEM:

Frequent Value Distribution (0.1000 %) - Item				🔒 Encrypted Co
Value	Count	Percentage		
UNKNOWN		344		3.4393 %
Cookie		1092		10.9178 %
Salad		1148		11.4777 %
		333		3.3293 %
ERROR		292		2.9194 %
Tea		1089		10.8878 %
Smoothie		1096		10.9578 %
Juice		1171		11.7077 %
Cake		1141		11.4077 %
Sandwich		1131		11.3077 %
Coffee		1164		11.6377 %

Aż 969 rekordów (9.69%) stanowią dane o wartościach **UNKNOWN, ERROR** lub **brak wartości**.













Location:

Value	Count	Percentage	
UNKNOWN		338	 3.3793 %
Takeaway		3024	 30.2340 %
		3265	 32.6435 %
ERROR		358	 3.5793 %
In-store		3017	 30.1640 %

W kolumnie **Location** niekompletnych mamy **3961 (39.61%)** rekordów.

Payment Method:



















Frequent Value Distribution (0.1000 %) - Payment Method

Value	Count	Percentage
	 2579	 25.7848 %
Digital Wallet	 2292	 22.9154 %
Credit Card	 2272	 22.7155 %
Cash	 2258	 22.5755 %
ERROR	 306	 3.0594 %
UNKNOWN	 293	 2.9294 %

Liczba niekompletnych rekordów – **3178 (31.78%)**.


Price Per Unit:

Frequent Value Distribution (0.1000 %) - Price Per Unit

Value	Count	Percentage
UNKNOWN	 164	 1.6397 %
5.0	 1204	 12.0376 %
2.0	 1227	 12.2675 %
3.0	 2431	 24.3051 %
1.5	 1133	 11.3277 %
	 179	 1.7896 %
4.0	 2331	 23.3053 %
ERROR	 190	 1.8996 %
1.0	 1143	 11.4277 %

Liczba niekompletnych rekordów – **533 (5.33%)**.


Quantity:

Frequent Value Distribution (0.1000 %) - Quantity  Encrypted Connection 1000 I

Value	Count	Percentage
UNKNOWN	171	1.7097 %
3	1851	18.5063 %
2	1973	19.7261 %
	138	1.3797 %
1	1821	18.2064 %
ERROR	170	1.6997 %
5	2013	20.1260 %
4	1863	18.6263 %

Liczba niekompletnych rekordów – **479 (4.79%)**.

Total Spent:

Frequent Value Distribution (0.1000 %) - Total Spent  Encrypted Connection 1000 Rows

Value	Count	Percentage
ERROR	164	1.6397 %
UNKNOWN	165	1.6497 %
	173	1.7297 %
1.5	205	2.0496 %
4.5	225	2.2496 %
1.0	232	2.3195 %
7.5	237	2.3695 %
25.0	259	2.5895 %
16.0	444	4.4391 %
5.0	468	4.6791 %
9.0	481	4.8090 %
2.0	497	4.9690 %

Liczba niekompletnych rekordów – **502 (5.02%)**.

Transaction Date:

Frequent Value Distribution (0.1000 %) - Transaction Date				1000 Rows
Value	Count	Percentage		
	160	1.5997 %		
UNKNOWN	159	1.5897 %		
ERROR	142	1.4197 %		
2023-02-06	40	0.3999 %		
2023-06-16	40	0.3999 %		
2023-09-21	39	0.3899 %		

Liczba niekompletnych rekordów – **461 (4.61%)**.

Transaction ID:

Transaction ID jest jedyną kolumną **kompletną** – w każdym rekordzie przypisaną mamy jakąś wartość.

Problem z Column Null Ratio Profiles:

Column Null Ratio Profiles - [dbo].[dane_lista3]				Encrypted Connection	1000 Rows
Column	Null Count	Null Percentage			
Item	0	0.0000 %			
Location	0	0.0000 %			
Payment Method	0	0.0000 %			
Price Per Unit	0	0.0000 %			
Quantity	0	0.0000 %			
Total Spent	0	0.0000 %			
Transaction Date	0	0.0000 %			
Transaction ID	0	0.0000 %			

Korzystając z narzędzia Column Null Ratio Profiles możemy wpaść w błędne przekonanie, że dane są kompletne – nie ma żadnych rekordów **NULL**. Narzędzie to jednak nie wykrywa rekordów pustych, UNKNOWN oraz ERROR, których w całych danych było aż **9622 (12.03%)**.

Wnioski:

Dane przedstawione w pliku dane_lista3.csv charakteryzują się **dużą niekompletnością**. Tylko 88% komórek było wypełnione poprawnymi danymi. Do niektórych danych możemy jednak dojść patrząc na pozostałe kolumny tabeli.

Przykład uzupełnienia brakujących danych:

Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
TXN_1961373	Coffee	2	2	4	Credit Card	Takeaway	9/8/2023
TXN_4977031	Cake	4	3	12	Cash	In-store	5/16/2023
TXN_4271903	Cookie	4	1 ERROR		Credit Card	In-store	7/19/2023
TXN_7034554	Salad	2	5	10	UNKNOWN	UNKNOWN	4/27/2023

Korzystając z zależności **Total Spent = Quantity * Price Per Unit** możemy obliczyć **Total Spent** dla Transakcji **TXN_4977031**. $\text{Total Spent} = 4 * 3 = 12$. Korzystając z podobnej logiki możemy uzupełnić wszystkie **niekompletne** rekordy w kolumnach Quantity, Price Per Unit oraz Total Spent dla których pozostałe dwie kolumny są wypełnione poprawnie.

TXN_3229409	Juice	UNKNOWN	3	UNKNOWN	Cash	Takeaway	4/15/2023
TXN_9076216	Cake	1	3	3	Cash		12/17/2023
TXN_6371987	Tea	5	1.5 ERROR		UNKNOWN	Takeaway	3/13/2023

Nie wszystkie dane w tych kolumnach są jednak możliwe do uzupełnienia. W zamówieniu TXN_3229409 kolumny **Quantity** oraz **Total Spent** są niewiadome. Z tymi brakującymi danymi nie jesteśmy w stanie nic zrobić (nie zależą bezpośrednio od żadnej znanej nam kolumny).

Spójność:

Największym problemem ze spójnością danych jest różny zapis wartości niepoprawnych/niekompletnych. Zapisywane są na 3 różne sposoby:

- „” – pusta komórka
- „UNKNOWN”
- „ERROR”

Przeanalizujemy wszystkie wartości dla różnych kolumn pliku dane_lista3.csv

```
def sprawdz_spojnosc(file):
    usage
    reader = csv.DictReader(file)
    item_dict = {"Item" : [], "Quantity" : [], "Price Per Unit" : [], "Total Spent" : [],
                "Payment Method" : [], "Location" : []}
    for row in reader:
        for column in row:
            if column != "Transaction Date" and column != "Transaction ID":
                if row[column] not in item_dict[column]:
                    item_dict[column].append(row[column])
    for column in item_dict:
        print(column, item_dict[column])
```

```
Item ['Coffee', 'Cake', 'Cookie', 'Salad', 'Smoothie', 'UNKNOWN', 'Sandwich', '', 'ERROR', 'Juice', 'Tea', 'Coffe']
Quantity ['2', '4', '5', '3', '1', 'ERROR', 'UNKNOWN', '', '-2', '100']
Price Per Unit ['2.0', '3.0', '1.0', '5.0', '4.0', '1.5', '', 'ERROR', 'UNKNOWN']
Total Spent ['4.0', '12.0', 'ERROR', '10.0', '20.0', '9.0', '16.0', '15.0', '25.0', '8.0', '5.0', '3.0', '6.0', '', 'UNKNOWN', '2.0', '1.0', '7.5', '4.5', '1.5']
Payment Method ['Credit Card', 'Cash', 'UNKNOWN', 'Digital Wallet', 'ERROR', '', 'Digital Walle', 'CreditCard']
Location ['Takeaway', 'In-store', 'UNKNOWN', '', 'ERROR']
```

Problemy:

- W kolumnie Item pojawiają się wartości „Coffee” oraz „Coffe” – literówka
- W kolumnie Quantity pojawia się wartość „-2” – błąd wartości
- W kolumnie Payment Method pojawiają się wartości „Credit Card” i „CreditCard” oraz „Digital Wallet” oraz „Digital Walle” – literówki.

Kolumny Transaction Date i Transaction ID:

Z racji dużej ilości różnych wartości kolumn Transaction Date i Transaction ID przeanalizujemy je osobno.

Transaction Date:

```
def sprawdz_spojnosc_dat(file): 1 usage
    reader = csv.DictReader(file)
    niepoprawne_daty = []
    for row in reader:
        if (row['Transaction Date'] == "" or row['Transaction Date'] == "UNKNOWN" or row['Transaction Date'] == "ERROR"):
            continue
        else:
            try:
                datetime.strptime(row['Transaction Date'], format: '%Y-%m-%d')
            except ValueError:
                niepoprawne_daty.append(row['Transaction Date'])
    if niepoprawne_daty:
        print("Niepoprawne daty:")
        for data in niepoprawne_daty:
            print(data)
    else:
        print("Wszystkie daty są poprawne.")
```

```
Niepoprawne daty:
2023/07/24
2023-16-04
```

Poza kolumnami „”, „UNKNOWN”, „ERROR” – 2 daty zostały zapisane w niepoprawnym formacie.

Transaction ID:

```
def sprawdz_spojnosc_transaction_id(file): 1 usage
    reader = csv.DictReader(file)
    niepoprawne_id = []
    pattern = re.compile(r'^TXN_\d{7}$') #regular expression dla TXN_0000000
    for row in reader:
        if not pattern.match(row['Transaction ID']):
            niepoprawne_id.append(row['Transaction ID'])
    if niepoprawne_id:
        print("Niepoprawne Transaction ID:")
        for txn_id in niepoprawne_id:
            print(txn_id)
    else:
        print("Wszystkie Transaction ID są poprawne.")

with open('dane_lista3.csv', newline='') as csvfile:
    sprawdz_spojnosc_transaction_id(csvfile)
```

Wszystkie dane są zapisane w spójnym formacie:

Wszystkie Transaction ID są poprawne.

TXN_0000000

Wnioski:

Jedynymi kolumnami które nie wykazywały problemów ze spójnością danych (poza różnymi zapisami wartości NULL) były Transaction ID, Price Per Unit, Total Spent oraz Location.

W pozostałych kolumnach:

- Item - literówka „Coffe”
- Quantity – błędna wartość - -2.
- Payment Method – literówki/błędny format “CreditCard”, “Digital Walle”
- Transaction Date – błędny format dat dla “2023/07/24”, “2023-16-04”

Poprawność Danych:

Sprawdzając spójność danych natrafiłszy już na niepoprawne dane:

```
Item ['Coffee', 'Cake', 'Cookie', 'Salad', 'Smoothie', 'UNKNOWN', 'Sandwich', '', 'ERROR', 'Juice', 'Tea', 'Coffe']
Quantity ['2', '4', '5', '3', '1', 'ERROR', 'UNKNOWN', '', '-2', '100']
Price Per Unit ['2.0', '3.0', '1.0', '5.0', '4.0', '1.5', '', 'ERROR', 'UNKNOWN']
Total Spent ['4.0', '12.0', 'ERROR', '10.0', '20.0', '9.0', '16.0', '15.0', '25.0', '8.0', '5.0', '3.0', '6.0', '', 'UNKNOWN', '2.0', '1.0', '7.5', '4.5', '1.5']
Payment Method ['Credit Card', 'Cash', 'UNKNOWN', 'Digital Wallet', 'ERROR', '', 'Digital Walle', 'CreditCard']
Location ['Takeaway', 'In-store', 'UNKNOWN', '', 'ERROR']
```

W jednym z rekordów trafiliśmy na ujemne quantity – jak wiadomo.

Reszta rekordów z kolumn Item, Quantity, Price Per Unit, Total Spent, Payment Method oraz Location, Transaction ID nie wydaje się wykazywać problemów z poprawnością.

Sprawdźmy jednak Transaction Date.

Transaction Date:

```
def sprawdz_poprawnosc_transaction_date(file): 1 usage
    reader = csv.DictReader(file)
    niepoprawne_daty = []
    current_date = datetime.now()
    for row in reader:
        if row['Transaction Date'] == "" or row['Transaction Date'] == "UNKNOWN" or row['Transaction ID'] == "":
            continue
        else:
            try:
                transaction_date = datetime.strptime(row['Transaction Date'], format='%Y-%m-%d')
                if transaction_date >= current_date or transaction_date.year < 2020:
                    niepoprawne_daty.append(row['Transaction Date'])
            except ValueError:
                niepoprawne_daty.append(row['Transaction Date'])
    if niepoprawne_daty:
        print("Niepoprawne daty:")
        for data in niepoprawne_daty:
            print(data)
    else:
        print("Wszystkie daty są poprawne.")
```

Niepoprawne daty:

2026-01-07

2023/07/24

2023-16-04

1023-03-24

Zauważamy 2 daty, które nie są poprawne:

- 2026-01-07 – data z przyszłości, zakup zostanie dokonany za prawie rok (w stosunku do reszty zakupów 3 lata)
- 1023-03-24 – prawdopodobnie literówka – chodziło o 2023 a nie 1013.

Unikalność Danych:

Unikalność danych odnosi się tylko do Transaction ID – w każdej innej kolumnie powtórzenie się takiej samej wartości jest czymś naturalnym. Sprawdźmy, czy dane w Transaction ID są unikalne.

Candidate Key Profiles - [dbo].[dane_lista3]		
Key Columns	Key Strength	
Transaction ID	<div><div></div></div>	99.9700 %

Key Violations			🔒 Encrypted Connection	All Rows
Transaction ID	Count	Percentage		
TXN_8859035	3	<div><div></div></div>	0.0300 %	
TXN_1108663	2	<div><div></div></div>	0.0200 %	

Dwa klucze powtarzają się więcej niż 1 raz:

- TXN_8859035 – 3 razy
- TXN_1108663 – 2 razy

Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
TXN_8859035	Cake	3	3	9	Digital Wallet	Takeaway	1/19/2023
TXN_8859035	Cake	3	3	9	Digital Wallet	Takeaway	1/19/2023
TXN_8859035	Cake	3	3	9	Digital Wallet	Takeaway	1/19/2023

Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
TXN_1108663	Juice	3	3	9	Cash	Takeaway	8/16/2023
TXN_1108663	Cookie	2	1	2	Digital Wallet	UNKNOWN	1023-03-24

Możliwe rozwiązanie:

- Usunięcie powtarzających się wierszy transakcji TXN_8859035 – zostawienie tylko jednego.
- Jeśli dane z pliku nie są powiązane z żadnym innym zbiorem danych za pomocą Transaction ID możemy przypisać jednej z transakcji o ID TXN_1108663 inny klucz, np. UNI_000000. Jeśli jednak Transaction ID jest powiązane z innym zbiorem danych, musimy rozwiązać ten problem w inny sposób.

Integralność i dokładność danych:

Przeanalizujemy integralność danych w kolumnach **Quantity**, **Price Per Unit**, **Total Spent**. Wiemy, że **Total Spent = Quantity * Price Per Unit**. Sprawdźmy zatem, czy w wierszach w których dane są kompletne, powyższa zależność jest spełniona.

```
def sprawdz_integralnosc(file): 1usage
    reader = csv.DictReader(file)
    print("Nieintegralne wartości:")
    for row in reader:
        try:
            if float(row['Quantity']) * float(row['Price Per Unit']) != float(row['Total Spent']):
                print(row)
                print(f"Błąd wartości: {float(row['Quantity']) * float(row['Price Per Unit'])} != {float(row['Total Spent'])}")
        except: # dla złych danych nic nie robimy
            pass
    return
with open('dane_lista3.csv', newline='') as csvfile:
    sprawdz_integralnosc(csvfile)
```

```
Nieintegralne wartości:
{'Transaction ID': 'TXN_7445257', 'Item': 'Coffee', 'Quantity': '100', 'Price Per Unit': '2.0',
Błąd wartości: 200.0 != 2.0
```

Problem pojawił się w transakcji o ID: TXN_7445257 – Quantity = 100, PPU = 2 spodziewamy się Total Spent = 200, a w danych zapisane jest 2. Reszta danych, o ile kompletna jest integralna.

Dokładność cen:

```
def sprawdz_ceny(file): 1usage
    reader = csv.DictReader(file)
    price_dict = {}
    for row in reader:
        if row['Item'] in price_dict:
            if row['Price Per Unit'] not in price_dict[row['Item']]:
                price_dict[row['Item']].append(row['Price Per Unit'])
        else:
            price_dict[row['Item']] = [row['Price Per Unit']]
    for item in price_dict:
        print(item, price_dict[item])
```

```

Coffee ['2.0', '', 'ERROR', 'UNKNOWN']
Cake ['3.0', '', 'UNKNOWN', 'ERROR']
Cookie ['1.0', 'UNKNOWN', '', 'ERROR']
Salad ['5.0', 'ERROR', 'UNKNOWN', '']
Smoothie ['4.0', '', 'UNKNOWN', 'ERROR']
UNKNOWN ['3.0', '1.0', '5.0', '4.0', '1.5', '2.0', '', 'UNKNOWN', 'ERROR']
Sandwich ['4.0', '', 'ERROR', 'UNKNOWN']
['3.0', '2.0', '1.0', '5.0', '', '4.0', '1.5', 'ERROR', 'UNKNOWN']
ERROR ['1.5', '3.0', '5.0', '', '4.0', '2.0', '1.0', 'UNKNOWN', 'ERROR']
Juice ['3.0', '', 'UNKNOWN', 'ERROR']
Tea ['1.5', '', 'ERROR', 'UNKNOWN']
Coffe ['2.0']

```

Widzimy, że żaden produkt nie ma więcej niż jednej ceny – ceny nie zmieniały się na przestrzeni lat.

- Coffee – 2.0
- Cake – 3.0
- Cookie – 1.0
- Salad – 5.0
- Smoothie – 4.0
- Sandwich – 4.0
- Juice – 3.0
- Tea – 1.5

Wszystkie produkty poza Cake i Juice – 3.0 oraz Smoothie i Sandwich – 4.0 mają unikalne ceny.

Wnioski:

Analizując integralność oraz dokładność danych, jesteśmy w stanie uzupełnić niektóre niekompletne dane.

Ocena jakości danych

Dane w pliku *dane_lista3.csv* posiadały problemy w każdej z poniższych kategorii:

- Kompletność – 12% danych stanowiły komórki typu „/„UNKNOWN”/”NULL
- Spójność – w danych pojawiały się literówki (np. „Coffe”), złe wartości (np. -2 w Quantity) i niepoprawne formaty daty („YYYY-DD-MM” oraz „YYYY/MM/DD”)
- Poprawność – w danych pojawiały się niepoprawne wartości
 - -2 Dla Quantity
 - Niepoprawne daty
 - 1023 – 03 – 24 – data z przeszłości
 - 2026 – 01 – 07 - data z przyszłości
- Unikalność – dwa klucze pojawiły się więcej niż jeden raz:
 - TXN_8859035 – 3 razy
 - TXN_1108663 – 2 razy
- Integralność - w transakcji o ID: TXN_7445257 – Quantity = 100, PPU = 2 spodziewamy się Total Spent = 200, a w danych zapisane jest 2

Możliwe sposoby poprawienia jakości danych:

Kompletność:

Analizując Integralność danych jesteśmy w stanie uzupełnić niektóre brakujące dane.

- Zależność pomiędzy produktem a ceną jednostkową – jeśli znamy nazwę produktu, wiemy jaką ma cenę jednostkową (ceny nie zmieniały się na przestrzeni trwania zapisów transakcji)
 - Ceny produktów
 - Coffee – 2.0
 - Cake – 3.0
 - Cookie – 1.0
 - Salad – 5.0
 - Smoothie – 4.0
 - Sandwich – 4.0
 - Juice – 3.0
 - Tea – 1.5
 - Dla większości cen jesteśmy też w stanie dowiedzieć się jakiego produktu dotyczą (poza 3.0 oraz 4.0 – które odpowiadają dwóm różnym produktom).
 - 1.0 – Cookie
 - 1.5 – Tea
 - 2.0 – Coffee
 - 3.0 – Cake/Juice
 - 4.0 – Smoothie/Sandwich
 - 5.0 – Salad
- Zależność pomiędzy Liczbą produktu, Ceną jednostkową i Ceną całościową
 - Jeśli znamy co najmniej 2 z 3 kolumn – jesteśmy w stanie wyliczyć 3 kolumnę za pomocą formuły
 - $\text{Quantity} * \text{Price Per Unit} = \text{Total Sum}$

Spójność:

Aby zwiększyć spójność danych powinniśmy zamienić wszystkie dane na taki sam format.

- Wartości typu NULL – powinniśmy ustandaryzować sposób zapisu wartości niepoprawnych/niekompletnych
 - „” -> NULL
 - „UNKNOWN” -> NULL
 - „ERROR” -> NULL
- Literówki – powinniśmy naprawić błędy w zapisie
 - „Coffe” -> „Coffee”
 - „CreditCard” -> „Credit Card”
 - „Digital Walle” -> „Digital Wallet”
- Daty
 - 1023-03-24 -> 2023-03-24
 - Wartości typu „YYYY-DD-MM” i „YYYY/MM/DD” -> „YYYY-MM-DD”

Poprawność:

Aby dane stały się poprawne należy zamienić następujące wartości.

- Quantity: -2 -> 2
- Transaction Date - 2026 – 01 – 07 -> NULL – nie znamy poprawnej daty

Unikalność:

Aby naprawić problemy związane z unikalnością danych możemy dokonać następujących poprawek:

- W danych transakcja TXN_8859035 powtarza się 3 razy – możemy usunąć powtarzające się wiersze i zostawić tylko jeden z nich.
- Jeśli dane z pliku nie są powiązane z żadnym innym zbiorem danych za pomocą Transaction ID możemy przypisać jednej z transakcji o ID TXN_1108663 inny klucz, np. UNI_000000. Jeśli jednak Transaction ID jest powiązane z innym zbiorem danych, musimy rozwiązać ten problem w inny sposób.

Integralność:

Powinniśmy sprawdzić, skąd wynikał błąd w transakcji TXN_7445257

```
Nieintegralne wartości:  
{'Transaction ID': 'TXN_7445257', 'Item': 'Coffee', 'Quantity': '100', 'Price Per Unit': '2.0',  
Bład wartosci: 200.0 != 2.0}
```

- Jeśli wynikał z błędu w Quantity:
 - 100 -> 1
- Jeśli wynikał z błędu w Total Sum
 - 2.0 -> 200.0

Wnioski

Zadanie 1.

Funkcje podsumowujące są przydatnym narzędziem do analizy danych na podstawie grupowania przez różne kryteria. Dzięki funkcjom ROLLUP, CUBE, GROUPING SETS dowiedzieliśmy się między innymi takich rzeczy:

- Łączna suma transakcji firmy AdventureWorks w latach 2011-2014 wynosiła 21764628\$ z czego:
 - 2011 Rok – 2549441\$
 - 2012 Rok – 4860450\$
 - 2013 Rok – 8278860\$
 - 2014 Rok – 6075876\$
- Klientami którzy osiągnęli największe sumy transakcji w poszczególnych latach byli:
 - 2011 – Carlos Adams – 3953.99\$ - 619 miejsce ogólnie
 - 2012 – Willie Xu – 4821.25\$ - 451 miejsce ogólnie
 - 2013 – Jenny Chen – 11630.79\$ - 1 miejsce ogólnie
 - 2014 – Christy Rai – 6864.70\$ - 46 miejsce ogólnie
- Największe sumy zniżek w każdym roku odnotowano na rowerach
- Największą sumę zniżek według produktu osiągnięto na rowerze Touring-1000 Yellow, 60 (60526\$)
- W 2011 roku nie było żadnej zniżki na komponenty (lub nikt nie kupił przecenionego towaru).

Zadanie 2.

Funkcje okienkowe to idealne narzędzie do analizowania jak poszczególny wiersz odnosi się do reszty zestawienia. Za pomocą funkcji okienkowych dowiedzieliśmy się poniższych rzeczy:

- Największą sprzedaż dla każdej kategorii produktu osiągnięto w 2013 roku.
- Największy wzrost wartości sprzedaży w stosunku do poprzedniego roku osiągnięto dla ubrań pomiędzy 2011 i 2012 rokiem. Wynosił on 1541%.
- Sprzedaż rowerów nie miała aż tak drastycznych zmian w żadnym z lat. Jest to kategoria która sprzedawała się najrówniej na przestrzeni lat.
- Przeanalizowaliśmy wzorce sprzedaży wśród 2 sprzedawców charakteryzujących się największą liczbą transakcji:
 - Oboje sprzedawcy z największą liczbą sprzedaży wykazali podobne wzorce sprzedażowe w każdym roku na przestrzeni lat 2012-2014.
 - Największe sprzedaże Jillian Carson i Michael Blythe odnotowali w marcu 2014 roku oraz październiku 2011 roku.
 - Michael Blythe dokonał większej ilości transakcji w 2011 roku, Jillian Carson w 2012 i 2013, a w 2014 roku dokonali bardzo podobnej liczby sprzedaży.



Zadanie 3.

Za pomocą Integration Services Project w Visual Studio oraz Pythona przeanalizowaliśmy, scharakteryzowaliśmy i oceniliśmy dane znajdujące się w pliku dane_lista3.csv

Dane_lista3.csv to plik zawierający informację o transakcjach baru lub kawiarni zawartych w 2013 roku. Firma ta sprzedaje takie produkty jak kawa, herbata, ciastka, smoothie, sałatki, kanapki i sok.

Dane charakteryzowały się bardzo dużymi problemami z kategoriami takimi jak:

- Kompletność – 12% komórek było niekompletnych
- Spójność – w pliku występowały niekonsekwentne sposoby zapisu wartości typu NULL, występowały literówki oraz daty zapisane w złym formacie
- Poprawność – w pliku występowały niepoprawne dane – liczba produktu w transakcji mniejsza od 0 oraz daty z 1013 i 2026 roku
- Unikalność – dwa klucze występowały więcej niż jeden raz
- Integralność – Liczba zamówionych kaw lub

Jednak za pomocą dogłębnej analizy korzystając z Visual Studio oraz Pythona odnaleźliśmy reguły pozwalające nam poprawić jakość danych:

- Kompletność:
 - Uzupelnienie danych o produkcie i cenie jednostkowej na podstawie zależności między tymi kolumnami
 - Uzupelnienie danych o Cenie jednostkowej, Cenie całkowitej oraz Liczbie sprzedanych sztuk za pomocą formuły
 - $\text{Quantity} * \text{Price Per Unit} = \text{Total Sum}$
 - Nie jesteśmy w stanie poprawić niekompletnych danych w kolumnach Location, Payment Method oraz Transaction Date ponieważ nie mamy wystarczającej wiedzy.
- Spójność:
 - Nadpisanie komórek z literówkami oraz złym formatem danych za pomocą dobrych formatów (np. „Coffe” -> „Coffee” lub “2023/07/24” -> “2023-07-24”)
- Poprawność:
 - Zamienienie niepoprawnych wartości na wartości NULL lub wynikające z integralności między komórkami
 - Quantity: -2 -> 2

- Transaction Date - 2026 – 01 – 07 -> NULL – nie znamy poprawnej daty
- Unikalność:
 - Usunięcie powtarzających się wierszy transakcji TXN_8859035
 - Nadpisanie klucza jednej z transakcji o ID TXN_1108663
- Integralność:

Transaction ID	Item	Quantity	Price Per Unit	Total Spent	Payment Method	Location	Transaction Date
TXN_7445257	Coffee	100	2	2			4/2/2023

- Jeśli problem z transakcją TXN_7445257 wynikał z błędu w Quantity:
 - 100 -> 1
- Jeśli problem z transakcją TXN_7445257 wynikał z błędu w Total Sum
 - 2.0 -> 200.0