CAPSTONE PROJECT

MARKETING AND RETAIL ANALYTICS

By: Gurwinder Parhar

What is the problem ?

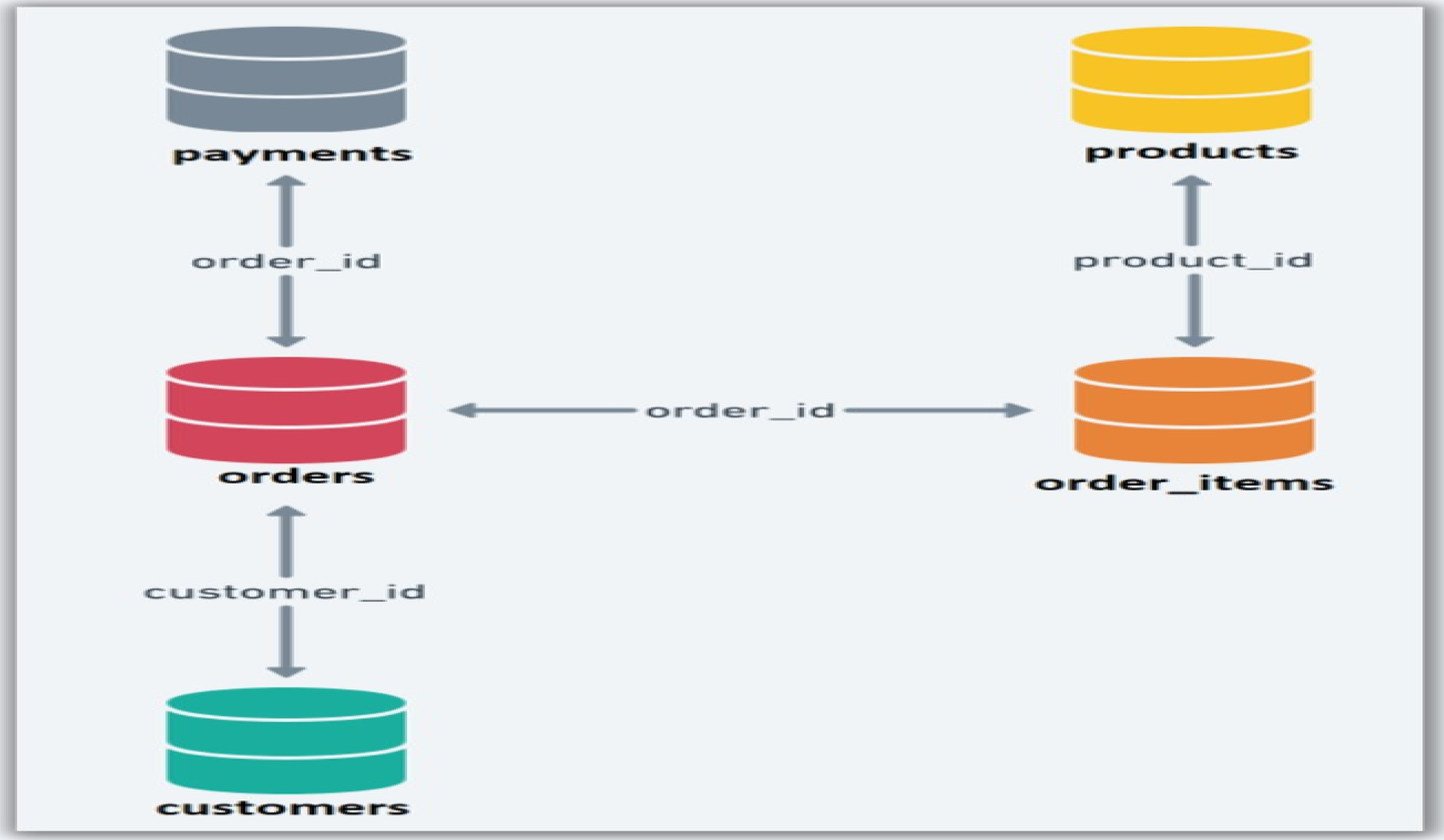
OList is an e-commerce company that has faced some losses recently and they want to manage their inventory very well so as to reduce any unnecessary costs that they might be bearing. In this assignment, you have to manage the inventory cost of this e-commerce company OList.

You need to identify top products that contribute to the revenue and also use market basket analysis to analyze the purchase behavior of individual customers to estimate with relative certainty, what items are more likely to be purchased individually or in combination with some other products. You need to help OList to identify the product categories which they can get rid of without significantly impacting business.

Dataset Overview

Dataset name	Column Name	Description
orders	order_id	Unique identifier for an order, acts as the primary key of this table
orders	customer_id	Unique identifier for a customer, however, this table wont be unique at this
orders	order_status	Indicates the status of an order, for example: delivered, cancelled, processing
orders	order_purchase_timestamp	Timestamp when the order was made from the customer
orders	order_approved_at	Timestamp when the order was approved from the sellers' side
orders	order_delivered_timestamp	Timestamp when the order was delivered at customer's location
orders	order_estimated_delivery_date	Estimated date of delivery shared with the customer while placing the order
order_items	order_id	Unique identifier for an order
order_items	order_item_id	Item number in each order. Order_id along with this column acts as the primary key of this table
order_items	product_id	Unique identifier for a product
order_items	seller_id	Unique identifier for the seller
order_items	price	selling price of the product
order_items	shipping_charges	charges associated with the shipping of the product
customers	customer_id	Unique identifier for a customer, acts as the primary key of this table
customers	customer_zip_code_prefix	Customer's Zip code
customers	customer_city	Customer's Zip city
customers	customer_state	Customer's Zip state
payments	order_id	Unique identifier for an order, this table can have duplicates in this column
payments	payment_sequential	Povides the info of the sequence of payments for the given order
payments	payment_type	Type of payment like credit_card, debit_card etc.
payments	payment_installments	Payment installement number in case of credit cards
payments	payment_value	Trasaction value
products	product_id	Unique identifier for each product, acts as the primary key of this table
products	product_category_name	Name of the category the product belongs to
products	product_weight_g	Product weight in grams
products	product_length_cm	Product length in centimeters
products	product_height_cm	Product height in centimeters
products	product_width_cm	Product width in centimeters

Relationship Diagram of Entities -



Steps performed in this project.

1. Data exploration and cleaning: Imported the dataset and identified missing and duplicate values in each column and treated them accordingly. Also, treated data quality issues associated with the dataset.

2. EDA: Created appropriate visualizations to explore Dataset.

3. Market basket analysis: Using Apriori, identified the combinations of product categories that are ordered frequently

Cleaning and EDA -

IMPORTING THE LIBRARIES

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

READING ALL THE EXCEL SHEETS

A. Orders

```
# Reading the Orders sheet
orders = pd.read_excel("Retail_dataset (1).xlsx", sheet_name="orders")
orders.shape
```

```
(99441, 7)
```

```
orders.head()
```

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at
0	e481f51cbdc54678b7cc49136f2d6af7	7c396fd4830fd04220f754e42b4e5bff	delivered	2017-10-02 10:56:33	2017-10-02 11:07:15
1	53cdb2fc8bc7dce0b6741e2150273451	af07308b275d755c9edb36a90c618231	delivered	2018-07-24 20:41:37	2018-07-26 03:24:27
2	47770eb9100c2d0c44946d9cf07ec65d	3a653a41f6f9fc3d2a113cf8398680e8	delivered	2018-08-08 08:38:49	2018-08-08 08:55:23
3	949d5b44dbf5de918fe9c16f97b45f8a	7c142cf63193a1473d2e66489a9ae977	delivered	2017-11-18 19:28:06	2017-11-18 19:45:59
4	ad21c59c0840e6cb83a9ceb5573f8159	72632f0f9dd73dfee390c9b22eb56dd6	delivered	2018-02-13 21:18:39	2018-02-13 22:20:29

B. Order_items

```
# Reading the Order_items sheet
order_items = pd.read_excel("Retail_dataset (1).xlsx", sheet_name="order_items")
order_items.shape
```

```
(112650, 6)
```

```
order_items.head()
```

	order_id	order_item_id	product_id	seller_id	price	shipping_charges
0	00010242fe8c5a6d1ba2dd792cb16214	1	4244733e06e7ecb4970a6e2683c13e61	48436dade18ac8b2bce089ec2a041202	58.90	13.29
1	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a8f	dd7ddc04e1b6c2c614352b383efe2d36	239.90	19.93
2	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f23d	199.00	17.87
3	00024acbcdf0a6daa1e931b038114c75	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c2b4	12.99	12.79
4	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e10089	df560393f3a51e74553ab94004ba5c87	199.90	18.14

C. Customers

```
# Reading the Customers sheet
customers = pd.read_excel("Retail_dataset (1).xlsx", sheet_name="customers")
customers.shape
```

```
(99441, 4)
```

```
customers.head()
```

	customer_id	customer_zip_code_prefix	customer_city	customer_state
0	861eff4711a542e4b93843c6dd7febb0	14409	franca	SP
1	290c77bc529b7ac935b93aa66c333dc3	9790	sao bernardo do campo	SP
2	060e732b5b29e8181a18229c7b0b2b5e	1151	sao paulo	SP
3	259dac757896d24d7702b9acbbff3f3c	8775	mogi das cruzes	SP
4	345ecd01c38d18a9036ed96c73b8d066	13056	campinas	SP

D. Payments

```
# Reading the Payments sheet
payments = pd.read_excel("Retail_dataset (1).xlsx", sheet_name="payments")
payments.shape
```

```
(103886, 5)
```

```
payments.head()
```

	order_id	payment_sequential	payment_type	payment_installments	payment_value
0	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99.33
1	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	24.39
2	25e8ea4e93396b6fa0d3dd708e76c1bd	1	credit_card	1	65.71
3	ba78997921bbcdc1373bb41e913ab953	1	credit_card	8	107.78
4	42fdf880ba16b47b59251dd489d4441a	1	credit_card	2	128.45

E. Products

```
# Reading the Products sheet
products = pd.read_excel("Retail_dataset (1).xlsx", sheet_name="products")
products.shape
```

```
(32951, 6)
```

```
products.head()
```

	product_id	product_category_name	product_weight_g	product_length_cm	product_height_cm	product_width_cm
0	1e9e8ef04dbcff4541ed26657ea517e5	perfumery	225.0	16.0	10.0	14.0
1	3aa071139cb16b67ca9e5dea641aaa2f	art	1000.0	30.0	18.0	20.0
2	96bd76ec8810374ed1b65e291975717f	sports_leisure	154.0	18.0	9.0	15.0
3	cef67bcfe19066a932b7673e239eb23d	baby	371.0	26.0	4.0	26.0
4	9dc1a7de274444849c219cff195d0b71	housewares	625.0	20.0	17.0	13.0

Treating Null values and Duplicates -

A

```
# Checking if order_id is duplicate.

orders["order_id"].duplicated().sum()
```

```
0
```

```
#Checking missing values.

orders.isnull().sum().sort_values(ascending=False)
```

```
order_approved_at              14
order_delivered_timestamp       8
order_id                        0
customer_id                     0
order_status                    0
order_purchase_timestamp        0
order_estimated_delivery_date   0
dtype: int64
```

```
#Imputing values of order_approved_at with order_purchase_timestamp

orders.order_approved_at.fillna(orders.order_purchase_timestamp,inplace=True)

#Imputing values of order_delivered_timestamp with order_estimated_delivery_date

orders.order_delivered_timestamp.fillna(orders.order_estimated_delivery_date, inplace=True)
```

```
#Checking again if any missing values are left.

orders.isnull().sum().sort_values(ascending=False)
```

```
order_id                       0
customer_id                    0
order_status                   0
order_purchase_timestamp       0
order_approved_at              0
order_delivered_timestamp      0
order_estimated_delivery_date  0
dtype: int64
```

B

```
#Checking missing values.

order_items.isnull().sum().sort_values(ascending=False)
```

```
order_id            0
order_item_id       0
product_id          0
seller_id           0
price               0
shipping_charges    0
dtype: int64
```

```
# Checking if order_id is duplicate.

order_items["order_id"].duplicated()
```

```
0         False
1         False
2         False
3         False
4         False
          ...
112645    False
112646    False
112647    False
112648    False
112649    False
Name: order_id, Length: 112650, dtype: bool
```

C

```
# Checking Missing Values

customers.isnull().sum()
```

```
customer_id                 0
customer_zip_code_prefix    0
customer_city               0
customer_state              0
dtype: int64
```

```
# Checking for duplicate values

customers.customer_id.duplicated().sum()
```

```
3345
```

```
# Dropping the duplicate values

customers.drop_duplicates(subset="customer_id", keep="first", inplace= True)
```

```
customers.shape
```

```
(96096, 4)
```

```
# Checking if any duplicates left

customers.customer_id.duplicated().sum()
```

```
0
```

D

```
#Checking the payments 'not defined' affected rows
payments[payments['payment_type']=='not_defined']
```

	order_id	payment_sequential	payment_type	payment_installments	payment_value
51280	4637ca194b6387e2d538dc89b124b0ee	1	not_defined	1	0.0
57411	00b1cb0320190ca0daa2c88b35206009	1	not_defined	1	0.0
94427	c8c528189310eaa44a745b8d9d26908b	1	not_defined	1	0.0

```
# Since there are only 3 recors affected, we can drop these records.

i=payments[payments['payment_type']=='not_defined'].index
payments.drop(i, axis=0, inplace=True)
```

```
# Checking Missing values

payments.isnull().sum()
```

```
order_id                0
payment_sequential      0
payment_type            0
payment_installments    0
payment_value           0
dtype: int64
```

E

```
# Checking the Missing Values

products.isna().sum()
```

```
product_id                 0
product_category_name    170
product_weight_g           2
product_length_cm          2
product_height_cm          2
product_width_cm           2
dtype: int64
```

```
#Checking the mode of "product_category_name" for imputing the categorical variable - 'product_category_name'

products["product_category_name"].mode()[0]
```

```
'toys'
```

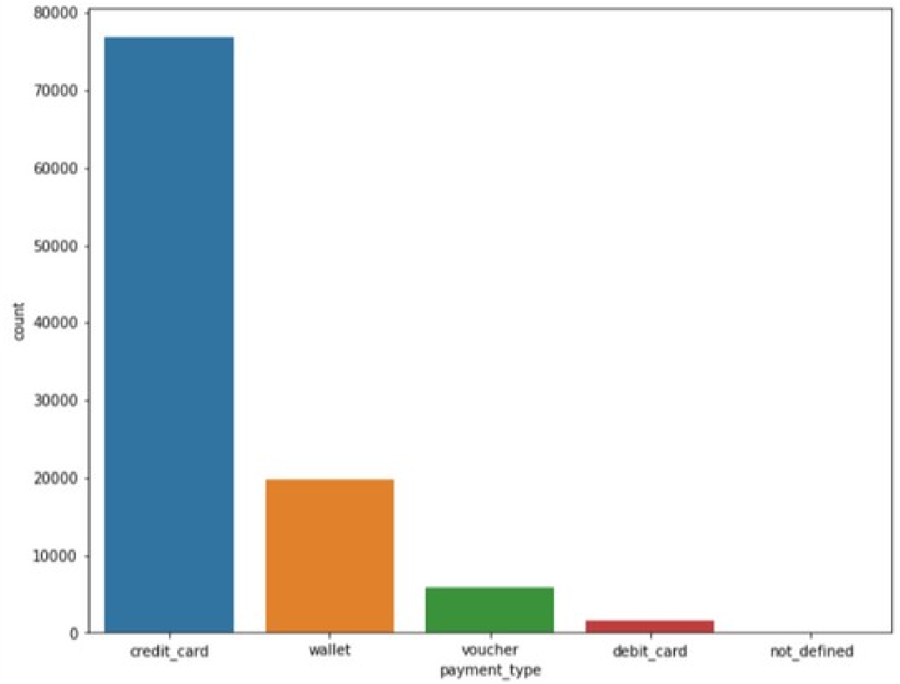
```
#Imputing the product_category_name NULL values

products["product_category_name"].fillna(products["product_category_name"].mode()[0], inplace=True)
```

```
#Checking missing values again for remaing columns.

products.isna().sum().sort_values(ascending=False)
```

```
product_weight_g         2
product_length_cm        2
product_height_cm        2
product_width_cm         2
product_id               0
product_category_name    0
dtype: int64
```

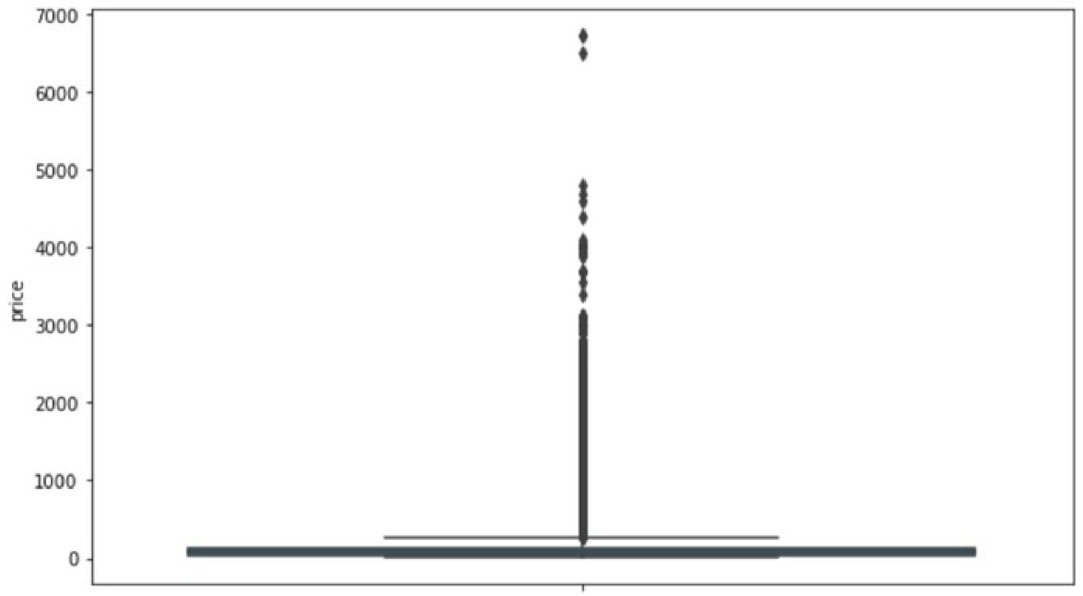
EDA -

```python
#Checking the most used 'payment_type' and their counts.

plt.figure(figsize=(10,8))
sns.countplot(data=payments, x=payments['payment_type'])
plt.show()
```



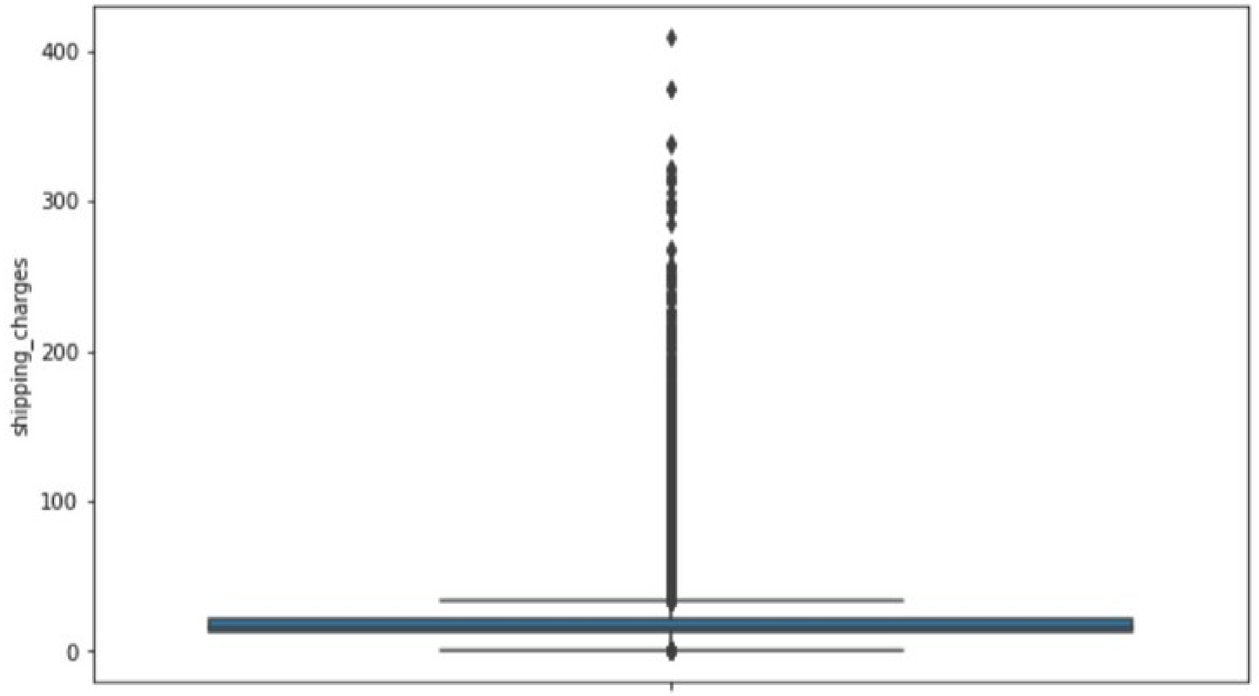
As per the following graph we can see that credit Card is the most used payment type for purchase.

```python
#Checking for 'price' outliers

plt.figure(figsize=(10,6))
sns.boxplot(data=orders_delivered, y=orders_delivered['price'])
plt.show()
```

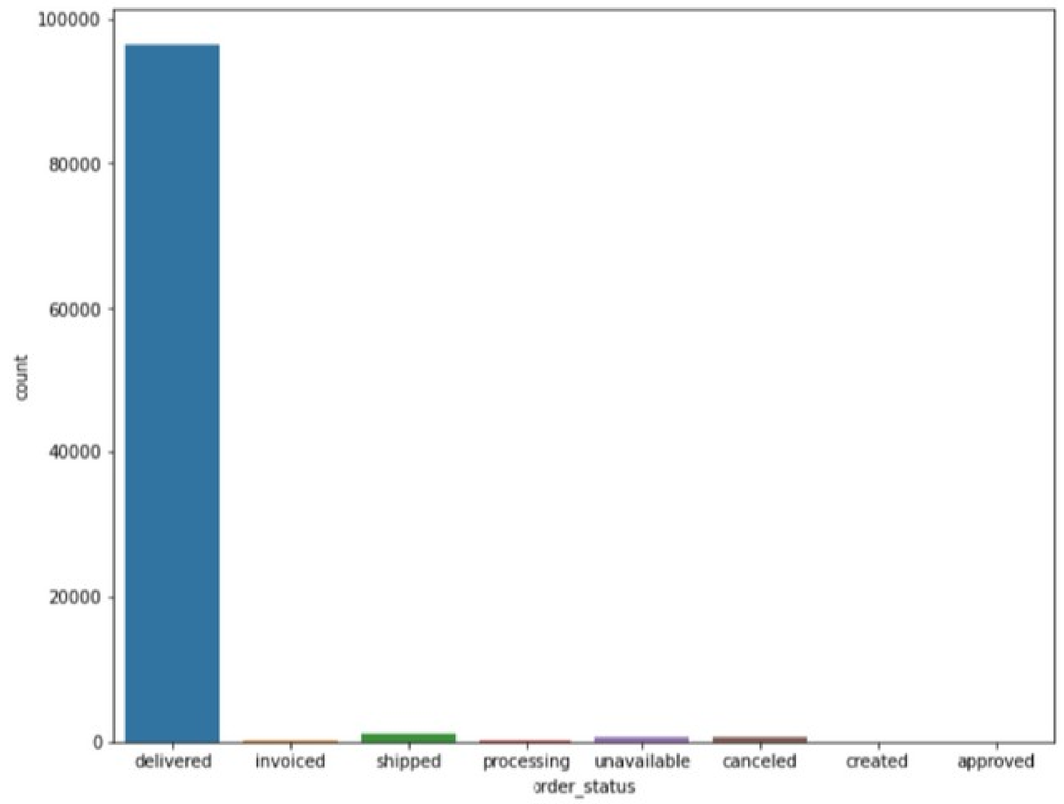



```python
#Assuming that anything above the price of 3000 is too expensive and shall be removed.
#Therefore checking the records for further investigation

orders_delivered[orders_delivered['price']>3000].head()
```

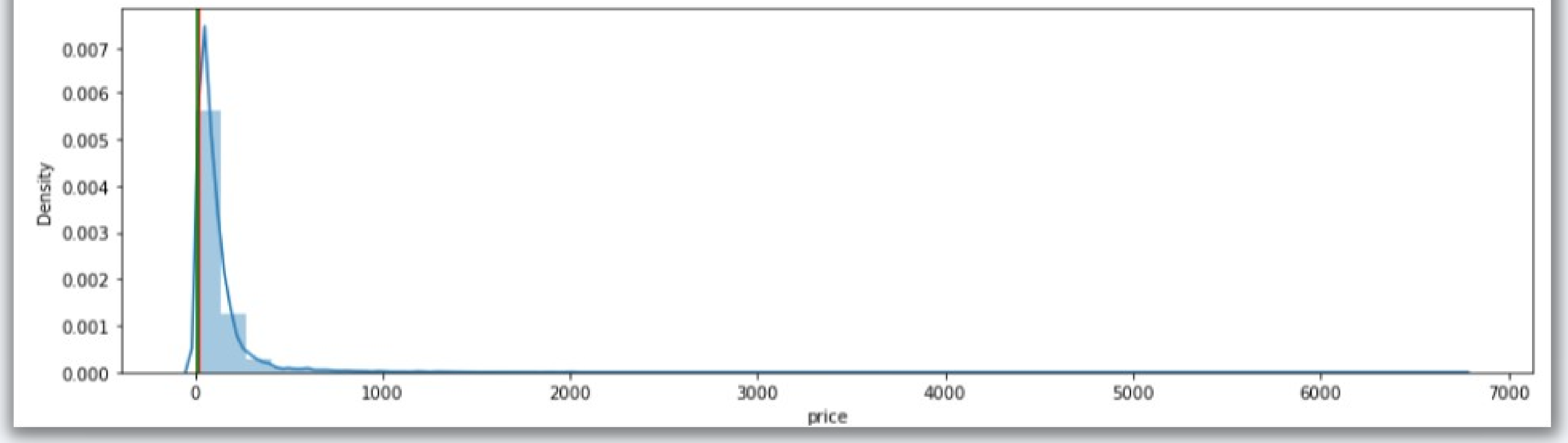
```python
#Checkinh for 'shipping_charges' outliers

plt.figure(figsize=(10,6))
sns.boxplot(data=orders_delivered, y=orders_delivered['shipping_charges'])
plt.show()
```




```python
#Checking the differet 'order_status' and their counts.

plt.figure(figsize=(10,8))
sns.countplot(data=orders, x=orders['order_status'])
plt.show()
```




```python
# Toys are usually not that costly, therefore the prices are outliers.
# Checking the correct measure for imputation using skewness

plt.figure(figsize=(15,4))
sns.distplot(orders_delivered.price)
plt.axvline(orders_delivered.shipping_charges.mean(), color="red")
plt.axvline(orders_delivered.shipping_charges.median(), color="green")
plt.show()
```



MARKET BASKET ANALYSIS

Association Rule Mining is primarily used when you want to identify an association between different items in a set, then find frequent patterns in a transactional database, relational databases (RDBMS).

Apriori Algorithm is a widely used and well-known Association Rule algorithm and is a popular algorithm used in market basket analysis. It is also considered accurate and overtop other algorithms. It helps to find frequent item sets in transactions and identifies association rules between these items.

```
#Installing the package Machine Learning Extension - mlxtend

!pip install mlxtend
```

```
Collecting mlxtend
  Downloading mlxtend-0.21.0-py2.py3-none-any.whl (1.3 MB)
Requirement already satisfied: pandas>=0.24.2 in c:\users\karti\anaconda3\lib\site-packages (from mlxtend) (1.4.2)
Requirement already satisfied: setuptools in c:\users\karti\anaconda3\lib\site-packages (from mlxtend) (61.2.0)
Requirement already satisfied: scipy>=1.2.1 in c:\users\karti\anaconda3\lib\site-packages (from mlxtend) (1.7.3)
Requirement already satisfied: numpy>=1.16.2 in c:\users\karti\anaconda3\lib\site-packages (from mlxtend) (1.21.5)
Requirement already satisfied: joblib>=0.13.2 in c:\users\karti\anaconda3\lib\site-packages (from mlxtend) (1.1.0)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\users\karti\anaconda3\lib\site-packages (from mlxtend) (1.0.2)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\karti\anaconda3\lib\site-packages (from mlxtend) (3.5.1)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\karti\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxt
end) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\karti\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(3.0.4)
Requirement already satisfied: pillow>=6.2.0 in c:\users\karti\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(9.0.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\karti\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxten
d) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\users\karti\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(21.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\karti\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxten
d) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\karti\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (0.
11.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\karti\anaconda3\lib\site-packages (from pandas>=0.24.2->mlxtend) (2021.
3)
Requirement already satisfied: six>=1.5 in c:\users\karti\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib>=
3.0.0->mlxtend) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\karti\anaconda3\lib\site-packages (from scikit-learn>=1.0.2->ml
xtend) (2.2.0)
Installing collected packages: mlxtend
Successfully installed mlxtend-0.21.0
```

```
#Load apriori and association modules from mlxtend.frequent_patterns

from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

```
#Updating orders_delivered dataframe with only the required columns for analysis

orders_delivered = orders_delivered[['order_id','product_category_name', 'order_item_id']]
```

```
#Checking the duplicates after updating

orders_delivered.duplicated().sum()
```

```
4841
```

```
#Dropping the duplicates keeping the first occurence

orders_delivered.drop_duplicates(keep='first', inplace=True)
```

```
#Creating prd_combo dataframe using pandas pivot, this is required for basket analysis

prd_combo = pd.pivot_table(data=orders_delivered,index='order_id',columns='product_category_name',
                          values='order_item_id',fill_value=0)
```


```
#Creating prd_combo dataframe using pandas pivot, this is required for basket analysis

prd_combo = pd.pivot_table(data=orders_delivered,index='order_id',columns='product_category_name',
                          values='order_item_id',fill_value=0)
```


```
prd_combo.head()
```

product_category_name order_id	agro_industry_and_commerce	air_conditioning	art	arts_and_craftmanship	audio	auto	baby	bed_bath_table
00010242fe8c5a6d1ba2dd792cb16214	0.0	0.0	0	0	0.0	0.0	0.0	0.0
00018f77f2f0320c557190d7a144bdd3	0.0	0.0	0	0	0.0	0.0	0.0	0.0
000229ec398224ef6ca0657da4fc703e	0.0	0.0	0	0	0.0	0.0	0.0	0.0
00024acbcdf0a6daa1e931b038114c75	0.0	0.0	0	0	0.0	0.0	0.0	0.0
00042b26cf59d7ce69dfabb4e55b4fd9	0.0	0.0	0	0	0.0	0.0	0.0	0.0

5 rows × 70 columns

```
#For basket analysis encoding the data to 1s and 0s
def encdata(x):
    if x<=0:
        return 0
    if x>=1:
        return 1
prd_combo_encode = prd_combo.applymap(encdata)
```

```
prd_combo_encode.shape
```

```
(96477, 70)
```

```
#As reuired by the assignment, dropping the Product_cataegories (columns) whose sum value (total_sale)
#is less than equal to 5

for column in prd_combo_encode.columns:
    if (prd_combo_encode[column].sum(axis=0, skipna=True)<=5):
        prd_combo_encode.drop(column, inplace=True, axis=1)
```

```
prd_combo_encode.shape
```

```
(96477, 61)
```

```
#Selecting only those order_ids where at least two items were purchased to find product combinations.
#This is reuired else the 'Toys' product_category will affect the whole analysis.
#Because the Support value for 'Toys' is biased due to its too much presence as single item orders

prd_combo_encode = prd_combo_encode[(prd_combo_encode>0).sum(axis=1)>=2]
prd_combo_encode.head()
```

product_category_name	agro_industry_and_commerce	air_conditioning	art	audio	auto	baby	bed_bath_table	books_general_interest
order_id								
00337fe25a3780b3424d9ad7c5a4b35e	0	0	0	0	0	0	1	0
00946f674d880be1f188abc10ad7cf46	0	0	0	0	0	0	0	0
00bcee890eba57a9767c7b5ca12d3a1b	0	0	0	0	0	0	0	0
01144cadcf64b6427f0a6580a3033220	0	0	0	0	0	0	0	0
013a98b3a668bcef05b98898177f6923	0	0	0	0	0	0	1	0

5 rows × 61 columns

Generating frequent itemsets from a list of items

First step in generation of association rules is to get all the frequent itemsets.Frequent itemsets are the ones which occur at least a minimum number of times in the transactions.

```
'''Call apriori function and passing minimum support here we are passing 3%, which means at least 3% in total number
   of transaction the item should be present.'''
#Support - This measure gives an idea of how frequent `ItemSet` is in all the transactions.

frequent_items = apriori(prd_combo_encode, min_support=0.03, use_colnames=True)
frequent_items
```

```
C:\Users\karti\anaconda3\lib\site-packages\mlxtend\frequent_patterns\fpcommon.py:111: DeprecationWarning: DataFrames with non-b
ool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame
with bool type
  warnings.warn(
```

	support	itemsets
0	0.031201	(auto)
1	0.230889	(bed_bath_table)
2	0.084243	(computers_accessories)
3	0.032761	(fashion_bags_accessories)
4	0.127145	(furniture_decor)
5	0.042122	(garden_tools)
6	0.066303	(health_beauty)
7	0.053822	(housewares)
8	0.055382	(sports_leisure)
9	0.971139	(toys)
10	0.058502	(watches_gifts)
11	0.030421	(auto, toys)
12	0.226989	(toys, bed_bath_table)
13	0.080343	(computers_accessories, toys)
14	0.031981	(fashion_bags_accessories, toys)
15	0.119345	(furniture_decor, toys)
16	0.035101	(garden_tools, toys)
17	0.063183	(health_beauty, toys)
18	0.049142	(housewares, toys)
19	0.048362	(sports_leisure, toys)
20	0.056942	(toys, watches_gifts)

Generating all possible rules from the frequent item set.

After the frequent item set are generated, identifying rules such as Confidence and Lift.

```
#We would apply association rules on frequent itemset to find product combinations.
#Confidence - This measure defines the likeliness of occurrence of consequent on the cart given that the cart
#already has the antecedents.

rules_conf = association_rules(frequent_items, metric="confidence", min_threshold=0.1)
rules_conf
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(auto)	(toys)	0.031201	0.971139	0.030421	0.975000	1.003976	0.000120	1.154446
1	(toys)	(bed_bath_table)	0.971139	0.230889	0.226989	0.233735	1.012325	0.002764	1.003714
2	(bed_bath_table)	(toys)	0.230889	0.971139	0.226989	0.983108	1.012325	0.002764	1.708580
3	(computers_accessories)	(toys)	0.084243	0.971139	0.080343	0.953704	0.982047	-0.001469	0.623401
4	(fashion_bags_accessories)	(toys)	0.032761	0.971139	0.031981	0.976190	1.005202	0.000165	1.212168
5	(furniture_decor)	(toys)	0.127145	0.971139	0.119345	0.938650	0.966546	-0.004131	0.470437
6	(toys)	(furniture_decor)	0.971139	0.127145	0.119345	0.122892	0.966546	-0.004131	0.995151
7	(garden_tools)	(toys)	0.042122	0.971139	0.035101	0.833333	0.858099	-0.005805	0.173167
8	(health_beauty)	(toys)	0.066303	0.971139	0.063183	0.952941	0.981262	-0.001207	0.613300
9	(housewares)	(toys)	0.053822	0.971139	0.049142	0.913043	0.940178	-0.003127	0.331903
10	(sports_leisure)	(toys)	0.055382	0.971139	0.048362	0.873239	0.899191	-0.005422	0.227682
11	(watches_gifts)	(toys)	0.058502	0.971139	0.056942	0.973333	1.002260	0.000128	1.082293

```
#Lift - This measure defines the likeliness of occurrence of consequent on the cart given that the cart already
#has the antecedent, but controlling the popularity of consequent.
#Here we are setting based on lift and keeping minimum lift as >1.

rules_lift=rules_conf[(rules_conf['lift'] > 1)]
rules_lift
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(auto)	(toys)	0.031201	0.971139	0.030421	0.975000	1.003976	0.000120	1.154446
1	(toys)	(bed_bath_table)	0.971139	0.230889	0.226989	0.233735	1.012325	0.002764	1.003714
2	(bed_bath_table)	(toys)	0.230889	0.971139	0.226989	0.983108	1.012325	0.002764	1.708580
4	(fashion_bags_accessories)	(toys)	0.032761	0.971139	0.031981	0.976190	1.005202	0.000165	1.212168
11	(watches_gifts)	(toys)	0.058502	0.971139	0.056942	0.973333	1.002260	0.000128	1.082293

As required by the assignment, cleaned datasets are needed, therefore Exporting cleaned dataset to an excel and other Market Basket metrics data to create dashboard using Tableau.

```
#Extracting the clean datasheets to be uploaded
with pd.ExcelWriter(r"C:\Users\karti\OneDrive\Desktop\Clean_Retail_dataset.xlsx") as excel_sheets:
    #Extracting the clean datasheets
    orders.to_excel(excel_sheets, sheet_name="orders", index=False)
    order_items.to_excel(excel_sheets, sheet_name="order_items", index=False)
    products.to_excel(excel_sheets, sheet_name="products", index=False)
    customers.to_excel(excel_sheets, sheet_name="customers", index=False)
    payments.to_excel(excel_sheets, sheet_name="payments", index=False)

#Extracting the additional markest basket metrics data to be visualized
#Taking care of the frozenset before exporting
frequent_items["itemsets"] = frequent_items["itemsets"].apply(lambda x: ', '.join(list(x))).astype("unicode")
rules_conf["antecedents"] = rules_conf["antecedents"].apply(lambda x: ', '.join(list(x))).astype("unicode")
rules_conf["consequents"] = rules_conf["consequents"].apply(lambda x: ', '.join(list(x))).astype("unicode")
rules_lift["antecedents"] = rules_lift["antecedents"].apply(lambda x: ', '.join(list(x))).astype("unicode")
rules_lift["consequents"] = rules_lift["consequents"].apply(lambda x: ', '.join(list(x))).astype("unicode")

with pd.ExcelWriter(r"C:\Users\karti\OneDrive\Desktop\Apriori_Market_basket.xlsx") as excel_sheets:
    frequent_items.to_excel(excel_sheets, sheet_name="support", index=False)
    rules_conf.to_excel(excel_sheets, sheet_name="confidence", index=False)
    rules_lift.to_excel(excel_sheets, sheet_name="lift", index=False)
```

Main highlights from Market Basket Analysis

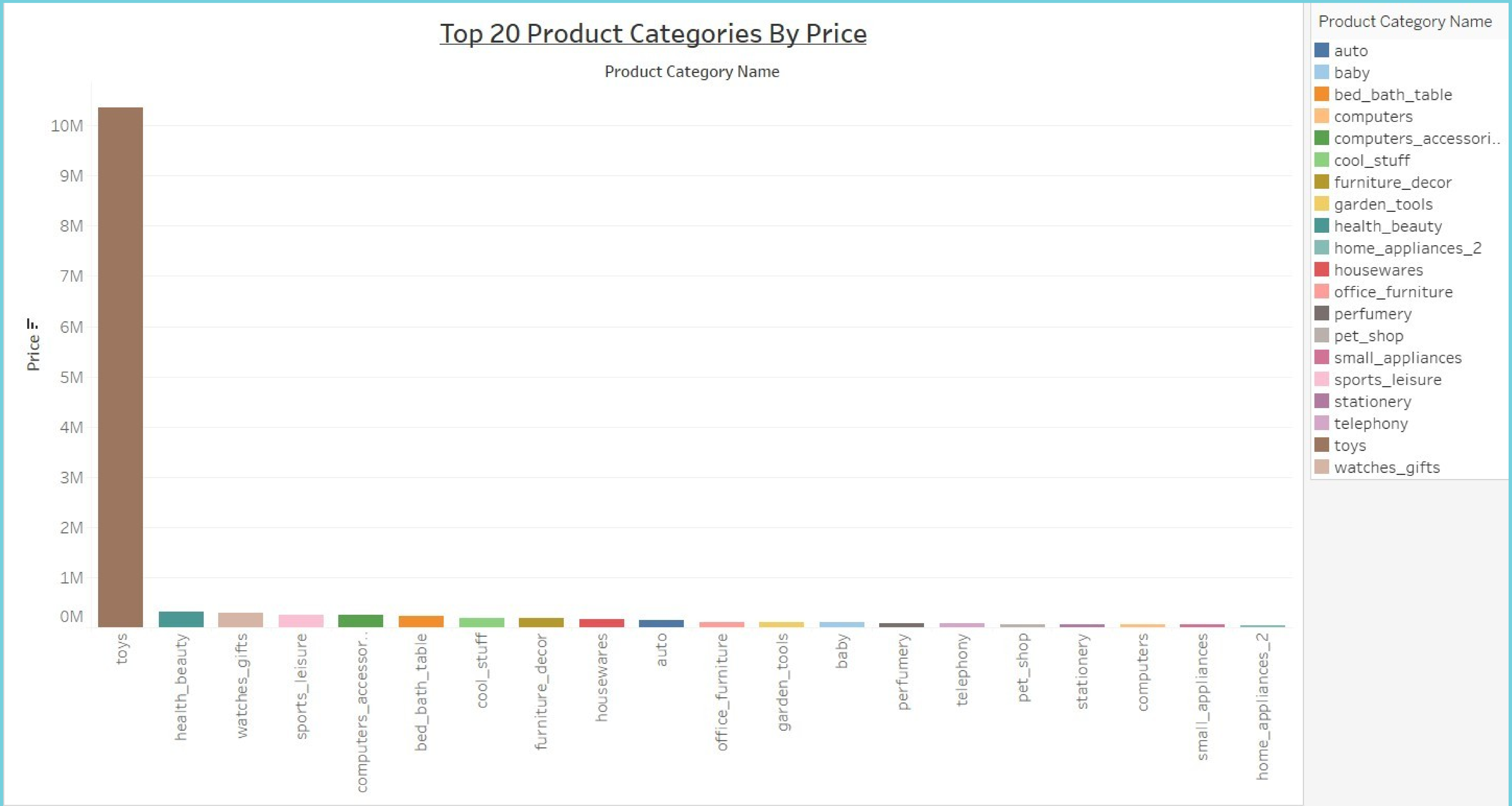
Top five products categories in groups of twos are:

1. Toys and Bed Bath Table
2. Toys and Fashion Bags Accessories
3. Toys and Auto
4. Toys and Watches Gift
5. Toys and Health & Beauty

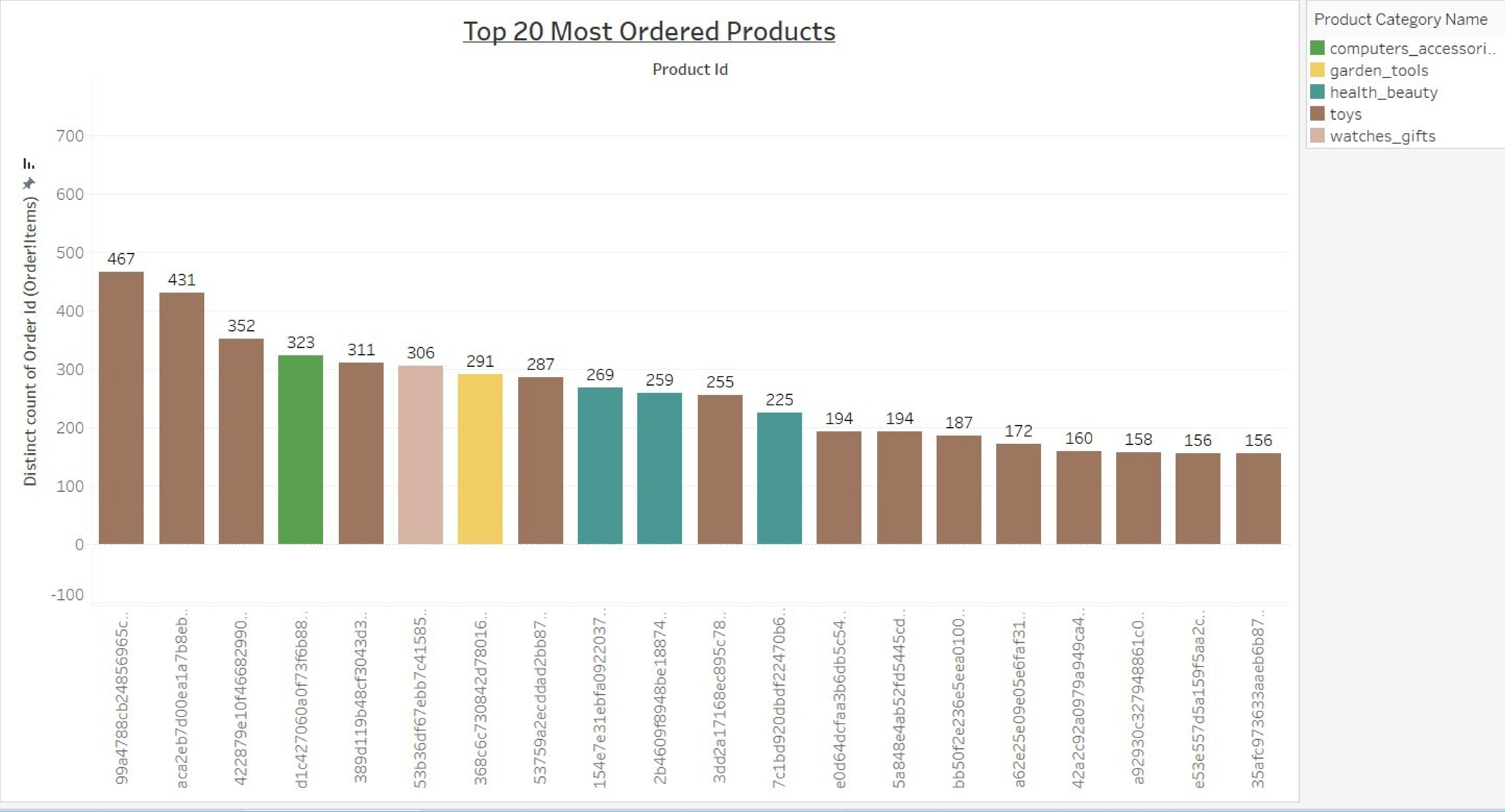
Top five products categories in groups of three are:

6.Toys, Cine photos and Telephony
7.Toys, Home Construction and Computer Accessories 3.Toys, Garden Tools and Computer Accessories 4.Toys Furniture Decor and Electronics
5.Toys, Furniture Decor and Health and Beauty

Visualizing Insights using Tableau



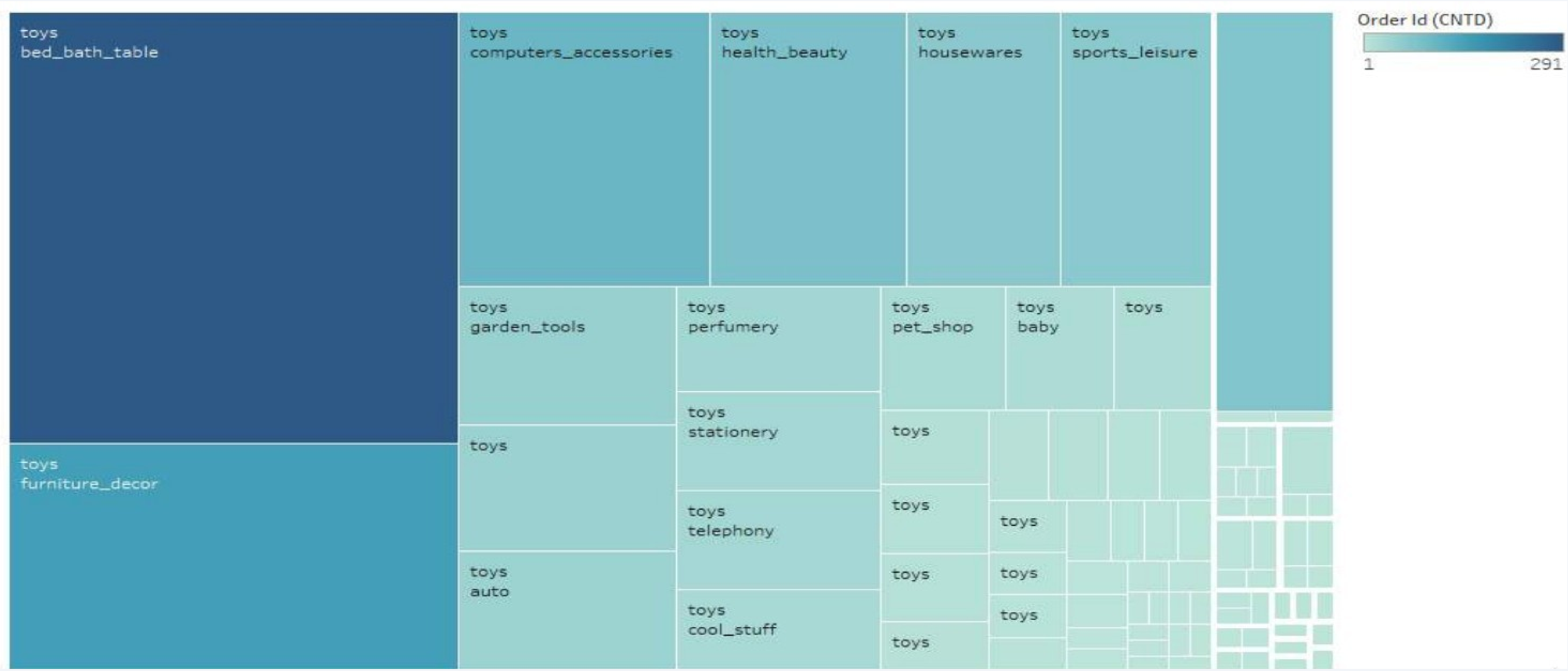
Top 20 Product Categories By Price
Product Category Name
Price
10M
9M
8M
7M
6M
5M
4M
3M
2M
1M
0M
toys
health_beauty
watches_gifts
sports_leisure
computers_accessor..
bed_bath_table
cool_stuff
furniture_decor
housewares
auto
office_furniture
garden_tools
baby
perfumery
telephony
pet_shop
stationery
computers
small_appliances
home_appliances_2
Product Category Name
auto
baby
bed_bath_table
computers
computers_accessori..
cool_stuff
furniture_decor
garden_tools
health_beauty
home_appliances_2
housewares
office_furniture
perfumery
pet_shop
small_appliances
sports_leisure
stationery
telephony
toys
watches_gifts

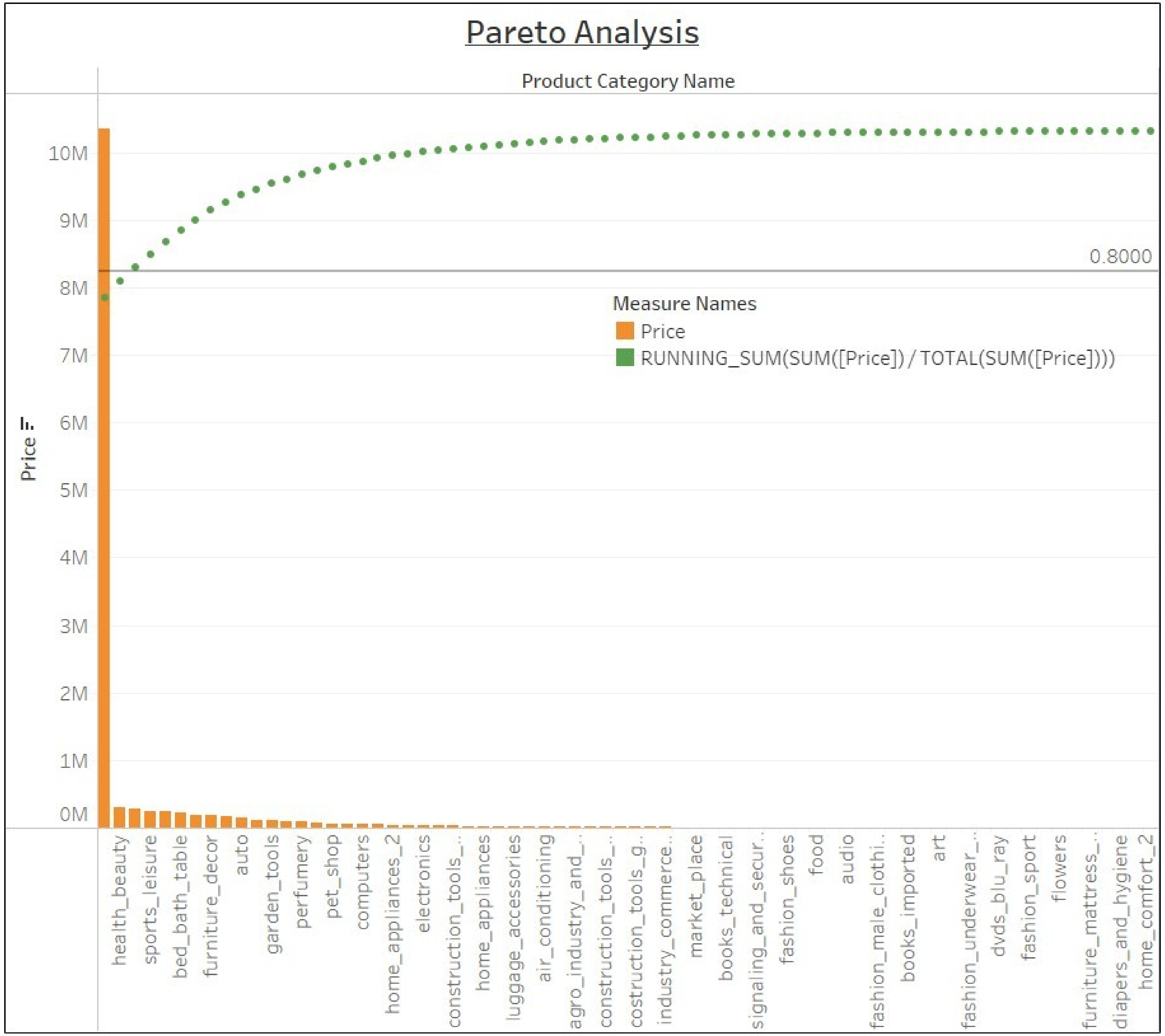


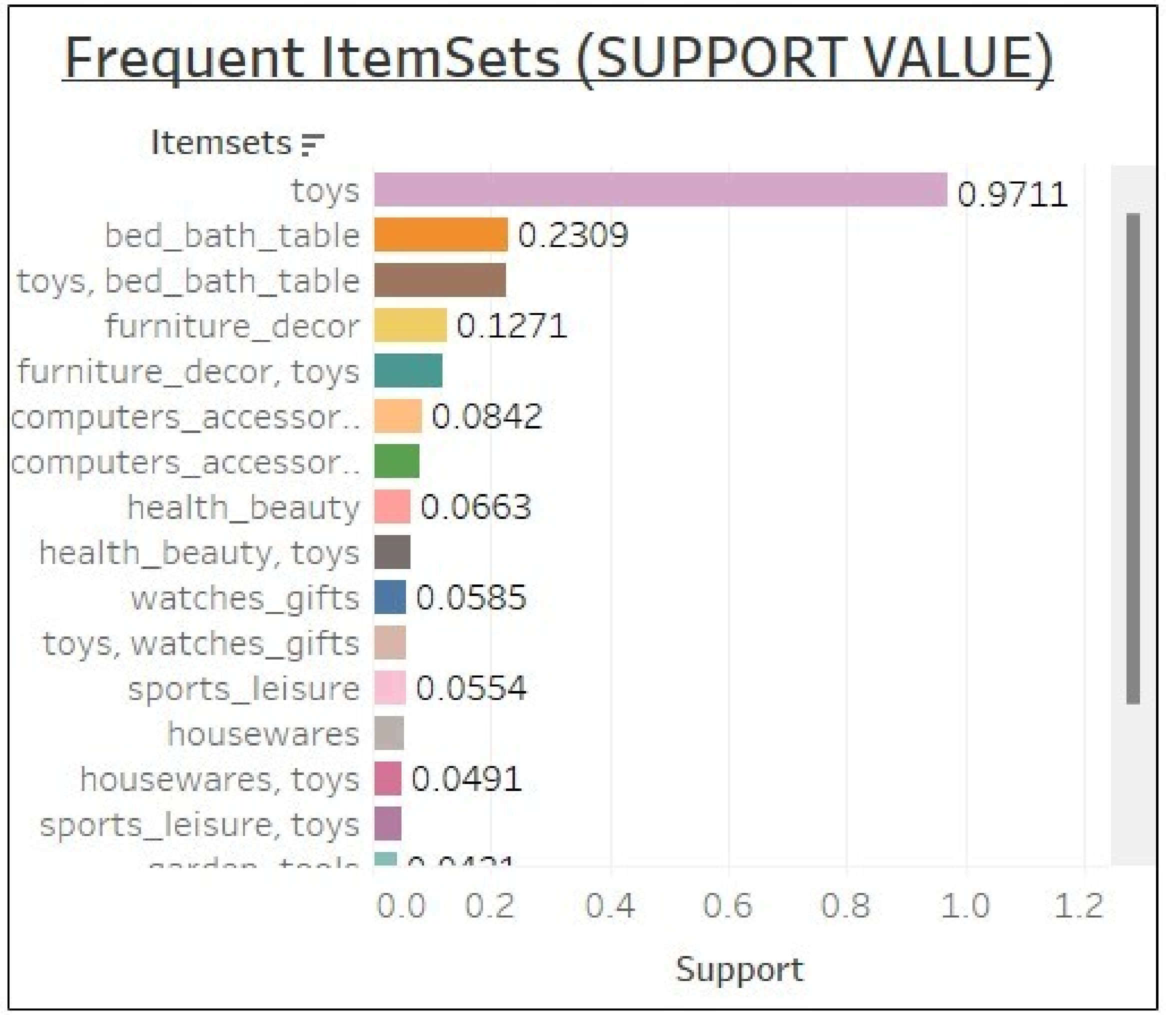
Top 20 Most Ordered Products
Product Id
Distinct count of Order Id (Order!Items)
700
600
500
400
300
200
100
0
-100
467
431
352
323
311
306
291
287
269
259
255
225
194
194
187
172
160
158
156
156
99a4788cb24856965c..
aca2eb7d00ea1a7b8eb..
422879e10f46682990..
d1c427060a0f73f6b88..
389d119b48cf3043d3..
53b36df67ebb7c41585..
368c6c730842d78016..
53759a2ecddad2bb87..
154e7e31ebfa0922037..
2b4609f8948be18874..
3dd2a17168ec895c78..
7c1bd920dbdf22470b6..
e0d64dcfaa3b6db5c54..
5a848e4ab52fd5445cd..
bb50f2e236e5eea0100..
a62e25e09e05e6faf31..
42a2c92a0979a949ca4..
a92930c327948861c0..
e53e557d5a159f5aa2c..
35afc973633aaeb6b87..
Product Category Name
computers_accessori..
garden_tools
health_beauty
toys
watches_gifts


Category-Wise Orders
Product Category Name
70K
60K
50K
40K
30K
20K
10K
0K
toys
health_beauty
bed_bath_table
sports_leisure
computers_ac..
furniture_decor
watches_gifts
housewares
telephony
auto
Product Category Name
auto
bed_bath_table
computers_accessori..
furniture_decor
health_beauty
housewares
sports_leisure
telephony
toys
watches_gifts

Market Basket Analysis for Two items at a time

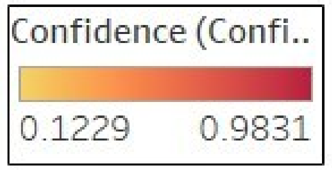


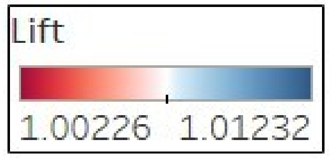




Product Combinations (Confidence Value)

Antecedents (Con..	Conseque..	
auto	toys	0.9750
bed_bath_table	toys	0.9831
computers_acces..	toys	0.9537
fashion_bags_ac..	toys	0.9762
furniture_decor	toys	0.9387
garden_tools	toys	0.8333
health_beauty	toys	0.9529
housewares	toys	0.9130
sports_leisure	toys	0.8732
toys	bed_bath_table	0.2337
	furniture_decor	0.1229
watches_gifts	toys	0.9733





Product Combinations (Lift > 1)

Antecedents	Consequents	
bed_bath_table	toys	1.01232
toys	bed_bath_table	1.01232
fashion_bags_ac..	toys	1.00520
auto	toys	1.00398
watches_gifts	toys	1.00226

Main Observations -

- It is observed that around 20 product categories account for 80% of Overall Sale count.
- Toys and bed_bath_table account for the highest number of sales
- Followed by the basket of Toys and furniture_decor.
- Close to 40-50 % of the sales of furniture decor, bed bath table, sports_leisure, fashion bags and accessories and auto have low frequency or hardly any sale.
- Flowers, home comfort, fashion children's clothing, furniture mattress and upholstery, security and services and many more have hardly any sale