Introduction

The Groceries dataset is a popular dataset used for market basket analysis, association rule learning, and frequent itemset mining. It consists of transactions from a grocery store, where each transaction is a list of items purchased together by a customer.

Key Characteristics:

Transactional Data: The dataset is a collection of transactions, where each transaction is a list of items.

Sparse Data: Typically, the dataset is sparse, meaning that most items do not appear in most transactions.

Categorical Data: Items are represented by categorical names or IDs.

Details of the dataset

The dataset has 38765 rows of the purchase orders of people from the grocery stores. These orders can be analysed and association rules can be generated using Market Basket Analysis by algorithms like Apriori Algorithm.

Association Rule Mining

Market Basket Analysis is one of the key techniques used by large retailers to uncover associations between items. It works by looking for combinations of items that occur together frequently in transactions. To put it another way, it allows retailers to identify relationships between the items that people buy.

Apriori Algorithm

Apriori is an algorithm for frequent itemset mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent itemsets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.

Source:

https://www.kaggle.com/datasets/heeraldedhia/groceries-dataset?

Importing Necessary Libraries

```
!pip install apyori #Installing apriori library
```

```
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5955 sha256=ade3
  Stored in directory: /root/.cache/pip/wheels/c4/1a/79/20f55c470a50bb3702a8cb7c94d8ada1
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
import numpy as np #NumPy is a powerful tool for numerical computations in Python.
import pandas as pd  #Pandas is a powerful library for data manipulation and analysis.
import seaborn as sns #Seaborn is a statistical data visualization library based on Matplotl
import matplotlib.pyplot as plt #Matplotlib is a comprehensive library for creating static,
```

Loading the Dataset.

```
df = pd.read_csv('Groceries_dataset.csv')
```

We will now read the data from a CSV file into a Pandas DataFrame Let us have a look at how our dataset looks like using df.head()

```
df.head() #Displays the first 5 rows of the dataset.
```

	Member_number	Date	itemDescription
0	1808	21-07-2015	tropical fruit
1	2552	05-01-2015	whole milk
2	2300	19-09-2015	pip fruit
3	1187	12-12-2015	other vegetables
4	3037	01-02-2015	whole milk

Next steps: Generate code with df View recommended plots

```
df.columns #Displays columns names of the dataset.
```

```
Index(['Member_number', 'Date', 'itemDescription'], dtype='object')
```

```
df.shape #Displays the total count of the Rows and Columns respectively.
```

```
(38765, 3)
```

```
df.isnull().sum() # Displays the total count of the null values in the particular columns.
```

```
Member_number      0
Date               0
itemDescription    0
dtype: int64
```

There is no null or missing value in the dataset.

```
df.info() # Displays the total count of values present in the particular column along with t
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Member_number    38765 non-null  int64 
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
dtypes: int64(1), object(2)
memory usage: 908.7+ KB
```

Checking for the Duplicate values

```
# Check for duplicate rows
duplicate_rows = df[df.duplicated()]

if duplicate_rows.empty:
    print("No duplicate values found.")
else:
    print("Duplicate values found:")
    print(duplicate_rows)
```

```
Duplicate values found:
       Member_number        Date   itemDescription
5015            2051  11-09-2015        frankfurter
5022            3055  18-08-2015  other vegetables
5044            1994  11-03-2015        whole milk
5055            1682  25-06-2015         pip fruit
5059            4324  05-01-2015           sausage
...              ...         ...               ...
38614           2027  26-02-2014     domestic eggs
38684           2936  07-03-2014        newspapers
38685           2311  13-03-2014        pot plants
38722           3834  18-05-2014       salty snack
38723           1146  23-05-2014            yogurt

[759 rows x 3 columns]
```

```
df.describe(include='all')
```

	Member_number	Date	itemDescription
count	38765.000000	38765	38765
unique	NaN	728	167
top	NaN	21-01-2015	whole milk
freq	NaN	96	2502
mean	3003.641868	NaN	NaN
std	1153.611031	NaN	NaN
min	1000.000000	NaN	NaN
25%	2002.000000	NaN	NaN
50%	3005.000000	NaN	NaN
75%	4007.000000	NaN	NaN
max	5000.000000	NaN	NaN

```
df['Date'] = pd.to_datetime(df['Date']) #Type-Conversion from Object to Dateime
```

```
<ipython-input-11-dabe41c7ec21>:1: UserWarning: Parsing dates in %d-%m-%Y format when da
  df['Date'] = pd.to_datetime(df['Date']) #Type-Conversion from Object to Dateime
```

This code converts the data in the 'Date' column of the DataFrame df to datetime objects. This conversion is crucial for performing any time-series analysis or date-related operations such as filtering, extracting specific parts of the date (like year, month, day), and calculating date differences.

```
df.info() #checking if the date columns are in proper format so we can use the data for furt
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Member_number    38765 non-null  int64
 1   Date             38765 non-null  datetime64[ns]
 2   itemDescription  38765 non-null  object
dtypes: datetime64[ns](1), int64(1), object(1)
memory usage: 908.7+ KB
```

```
df.head()
```

	Member_number	Date	itemDescription
0	1808	2015-07-21	tropical fruit
1	2552	2015-01-05	whole milk
2	2300	2015-09-19	pip fruit
3	1187	2015-12-12	other vegetables
4	3037	2015-02-01	whole milk

Next steps: Generate code with df | View recommended plots

```
df.Member_number.nunique() #The df.Member_number.nunique() function is a Pandas method used
```

```
3898
```

```
df.itemDescription.nunique() #The df.itemDescription.nunique() function is a Pandas method u
```

```
167
```

```
df.Date.nunique() #The df.Date.nunique() function is a Pandas method used to find the number
```

728

Creating Distribution of Item Sold

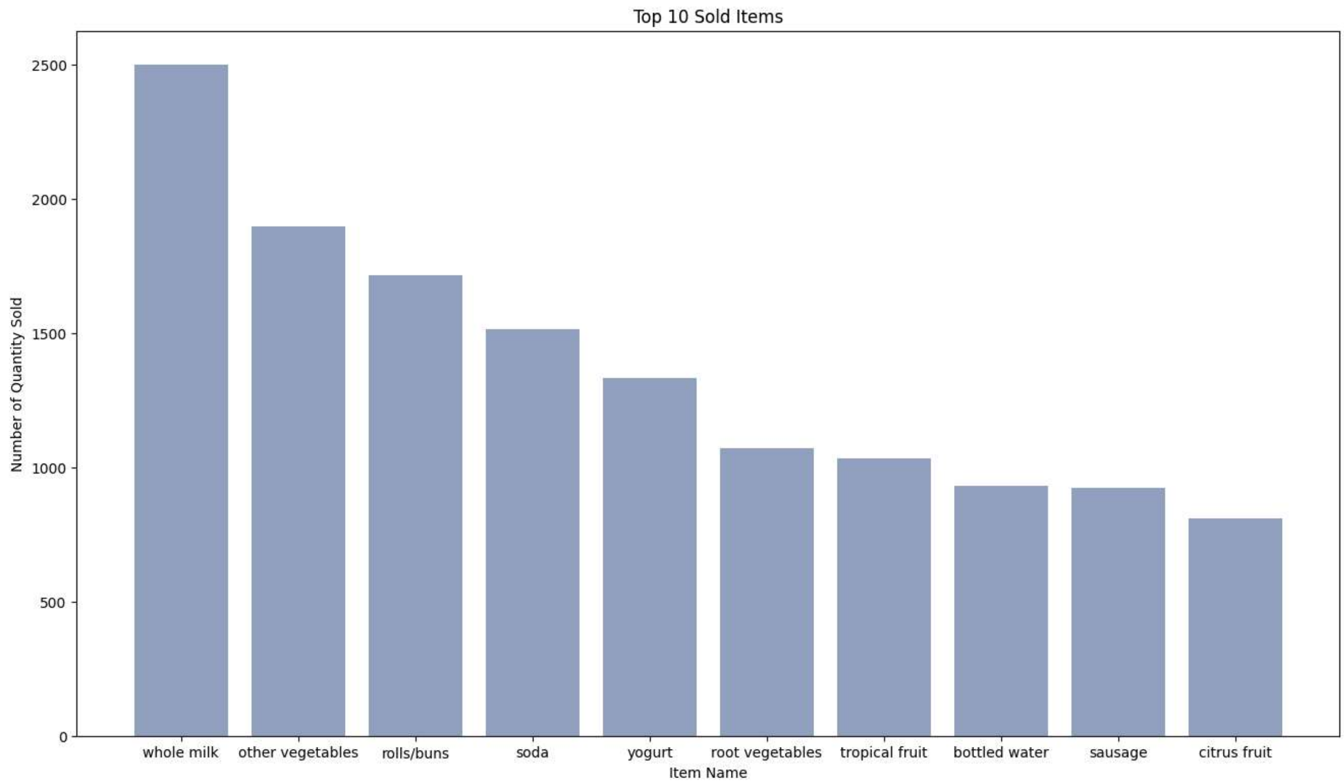
```
Item_distr = df.groupby(by = "itemDescription").size().reset_index(name='Frequency').sort_va
```

```
Item_distr #Displays the result.
```

	itemDescription	Frequency
164	whole milk	2502
102	other vegetables	1898
122	rolls/buns	1716
138	soda	1514
165	yogurt	1334
123	root vegetables	1071
156	tropical fruit	1032
12	bottled water	933
130	sausage	924
30	citrus fruit	812

Next steps: Generate code with `Item_distr` View recommended plots

```
## Declaring variables
bars = Item_distr["itemDescription"]
height = Item_distr["Frequency"]
x_pos = np.arange(len(bars))

## Defining Figure Size
plt.figure(figsize=(16,9))

# Create bars
plt.bar(x_pos, height, color=(0.3, 0.4, 0.6, 0.6))

# Add title and axis names
plt.title("Top 10 Sold Items")
plt.xlabel("Item Name")
plt.ylabel("Number of Quantity Sold")

# Create names on the x-axis
plt.xticks(x_pos, bars)

# Shows graph
plt.show()
```




```
df_date=df.set_index(['Date']) ## Setting date as index for plotting purpose
df_date
```

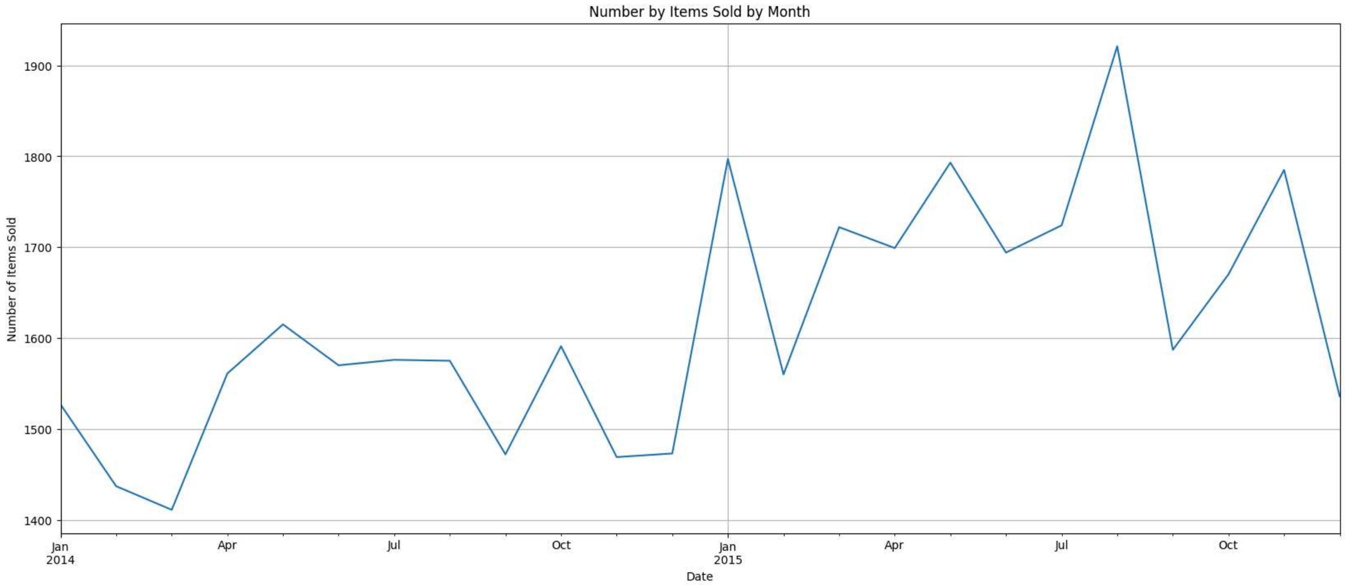
	Member_number	itemDescription
Date		
2015-07-21	1808	tropical fruit
2015-01-05	2552	whole milk
2015-09-19	2300	pip fruit
2015-12-12	1187	other vegetables
2015-02-01	3037	whole milk
...	...	...
2014-10-08	4471	sliced cheese
2014-02-23	2022	candy
2014-04-16	1097	cake bar
2014-12-03	1510	fruit/vegetable juice
2014-12-26	1521	cat food

38765 rows × 2 columns

Next steps: Generate code with `df_date` View recommended plots

```
df_date.resample("M")['itemDescription'].count().plot(figsize = (20,8), grid = True, title =
```

```
[Text(0.5, 0, 'Date'), Text(0, 0.5, 'Number of Items Sold')]
```



Apriori is an algorithm for frequent itemset mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent itemsets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis.

Data Preparation

```
cust_level = df[["Member_number", "itemDescription"]].sort_values(by = "Member_number", asce
cust_level['itemDescription'] = cust_level['itemDescription'].str.strip() #To remove any lea
cust_level
```

	Member_number	itemDescription
3578	5000	soda
34885	5000	semi-finished bread
11728	5000	fruit/vegetable juice
9340	5000	bottled beer
19727	5000	root vegetables
...	...	...
13331	1000	whole milk
17778	1000	pickled vegetables
6388	1000	sausage
20992	1000	semi-finished bread
8395	1000	whole milk

38765 rows × 2 columns

Next steps: Generate code with `cust_level` View recommended plots

Creating Transaction list

```
transactions = [a[1]['itemDescription'].tolist() for a in list(cust_level.groupby(['Member_r
```

Train Model

```
from apyori import apriori #Importing apriori package
```

```
rules = apriori(transactions = transactions, min_support = 0.002, min_confidence = 0.05, mir
```

```
results = list(rules) #Storing results in list format for better visualisation.
```

```
results
```

```
[RelationRecord(items=frozenset({'UHT-milk', 'kitchen towels'}),
support=0.002308876346844536, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'kitchen towels'}), items_add=frozenset({'UHT-
milk'}), confidence=0.30000000000000004, lift=3.821568627450981)]),
 RelationRecord(items=frozenset({'potato products', 'beef'}),
support=0.002565418163160595, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'potato products'}),
items_add=frozenset({'beef'}), confidence=0.4545454545454546,
lift=3.8021849395239955)]),
 RelationRecord(items=frozenset({'coffee', 'canned fruit'}),
support=0.002308876346844536, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'canned fruit'}),
items_add=frozenset({'coffee'}), confidence=0.4285714285714286,
lift=3.7289540816326534)]),
 RelationRecord(items=frozenset({'meat spreads', 'domestic eggs'}),
support=0.0035915854284248334, ordered_statistics=
[OrderedStatistic(items_base=frozenset({'meat spreads'}),
items_add=frozenset({'domestic eggs'}), confidence=0.4, lift=3.0042389210019267)]),
 RelationRecord(items=frozenset({'mayonnaise', 'flour'}), support=0.002308876346844536,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'flour'}),
items_add=frozenset({'mayonnaise'}), confidence=0.06338028169014086,
lift=3.3385991625428253), OrderedStatistic(items_base=frozenset({'mayonnaise'}),
items_add=frozenset({'flour'}), confidence=0.12162162162162163,
```