Project: Big Mart Sales Analysis using Python

CIS-5270

Professor: Dr. Shilpa Balan

California State University, Los Angeles

Submitted By:

Bhakti Parmar (401993186)

Shailja Pandit (401971060)

Contents

A. Introduction

Big Mart is a popular supermarket chain, with stores all around the USA. It is a large retail company that has been established around for over a century. Being one of the largest retailers in the world, they run over 3,000 stores and have an annual revenue of approximately $1.8 billion. In order to compete with other retailers, BigMart has been expanding its e-commerce business and increasing its online presence in recent years.

Sales analysis carries a lot of significance in the retail industry. Many Machine Learning Models are built which are used to predict future sales volume (R.P , S.M, 2021). With new challenges that keep growing with the increase in competition and due to ever-changing market dynamics, it is essential to analyze the data on hand, identify any associations, draw meaningful insights and interpretations from a given data and report and publish it for better decision-making. The reason why we chose this topic for our project was that we were keen on trying our hands-on to get a clear understanding of how to perform sales analysis in the retail industry. Our aim was to visualize the data using Python packages and libraries and interpret how the product Sales were impacted by Market Type (Grocery stores vs Supermarkets in our case), Outlet Type (Small, Medium, High), and if it had a correlation with various factors such as Item Weight or Item Visibility. The libraries in python come with lots of different features that enable users to make highly customized, elegant, and interactive plots. (AnalyticsVidhya.com, 2021)

Our effort was to come up with simple visualizations that would assist the retailers in making informed decisions while managing their stores or pertaining to their products w.r.t overall sales.

B. Data Description

To analyze the Big Mart sales data that belonged to the Fast-Moving Consumer Goods Sector, we used the public dataset available on Kaggle.

This dataset file format is CSV. This dataset provides data on sales in different outlets of BigMart. Data fields include Item Description attributes and Outlet Description attributes. Dataset consists of 12 columns and approximately 8000 rows

Dataset URL:

Big Mart Sales Data: https://www.kaggle.com/datasets/brijbhushannanda1979/bigmart-sales-data?select=Train.csv

Column Name	**Column Description**	**Sample Value**
Item_Identifier	Unique identifier of the Item / Product	FDA15
Item_Weight	Weight of an item in units	9.3
Item_Fat_Content	The fat content in the item / product. Product can be low fat or contain regular fat content	Low Fat
Item_Visibility	Visibility index of an item or a product on the website or in-store.	0.13

	When considering in-store, it indicates the % of total display area allocated to the particular product from all products in a store. Bigger items will have a higher percent of visibility. Minimum value is 0.003 and maximum value is 0.328	
Item_Type	It is the category to which the product belongs	Dairy
Item_MRP	Maximum retail price of the product	249.8
Outlet_Identifier	Unique identifier of the store	OUT049
Outlet_Establishment_Year	The year in which the outlet was established	1999
Outlet_Size	The size of the Size of the outlet in terms of ground area covered. This can be Small, Medium, or Large	Medium
Outlet_Location_Type	The type of city the outlet is located in. This location type can be Tier 1, Tier 2, or Tier 3 cities. Tier 1 cities would be the most developed cities (such as NYC and LA), Tier 2 cities are developing,	Tier 1

	and Tier 3 cities can be considered as underdeveloped cities	
Outlet_Type	Whether the outlet is a grocery store or some sort of supermarket. Supermarket values are Supermarket 1, Supermarket 2 and Supermarket 3 based upon the size of the supermarket. Grocery stores will be very small-sized, and Supermarket 1 will greater in size as compared Grocery store but smaller as compared to Supermarket 2. Supermarket 2 will be bigger in size than Supermarket 1 or Grocery stores but smaller than Supermarket 3.	Supermarket Type 1
Item_Outlet_Sales	Overall sales of the particular outlet	3735.13

An excerpt of our dataset is as follows:

A	B	C	D	E	F	G	H	I	J	K	L
Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
FDA15	9.3	Low Fat	0.016047301	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.138
DRC01	5.92	Regular	0.019278216	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
FDN15	17.5	Low Fat	0.016760075	Meat	141.618	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.27
FDO10	13.65	Regular	0.012741089	Snack Foods	57.6588	OUT013	1987	High	Tier 3	Supermarket Type1	343.5528
FDP10	12.85764518	Low Fat	0.127469857	Snack Foods	107.7622	OUT027	1985	Medium	Tier 3	Supermarket Type3	4022.7636
FDH17	16.2	Regular	0.016687114	Frozen Foods	96.9726	OUT045	2002	Medium	Tier 2	Supermarket Type1	1076.5986
FDU28	19.2	Regular	0.09444959	Frozen Foods	187.8214	OUT017	2007	Medium	Tier 2	Supermarket Type1	4710.535
FDA03	18.5	Regular	0.045463773	Dairy	144.1102	OUT046	1997	Small	Tier 1	Supermarket Type1	2187.153

Tools and Technologies Used -

Language Used: Python 3.9

IDE Used: Spyder 5.1.5 , Jupiter Notebook 6.4.5

Python Libraries used for Visualization: Matplot, Seaborn

C. Data Cleaning

Before beginning to clean the data, we first read the data from our CSV file into a data frame and printed it. The code used to do so in the Spyder IDE was:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings('ignore')


# Read the CSV file into Dataframe
df = pd.read_csv('BigMart.csv')

# View the top 5 values
print(df.head())
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings('ignore')


# Read the CSV file into Dataframe
df = pd.read_csv('BigMart.csv')

# View the top 5 values
print(df.head())
```

We printed the top 5 rows of our data set and the result looked as follows:

```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/CSULA/CSU LA/CSU LA/MSIS/Spring 2022/CIS 5270 - Business Intelligence/
Python Project/Big Mart Project/BigMartProject.py', wdir='C:/CSULA/CSU LA/CSU LA/MSIS/Spring
2022/CIS 5270 - Business Intelligence/Python Project/Big Mart Project')
  Item_Identifier  Item_Weight  ...        Outlet_Type  Item_Outlet_Sales
0           FDA15         9.30  ...  Supermarket Type1          3735.1380
1           DRC01         5.92  ...  Supermarket Type2           443.4228
2           FDN15        17.50  ...  Supermarket Type1          2097.2700
3           FDX07        19.20  ...      Grocery Store           732.3800
4           NCD19         8.93  ...  Supermarket Type1           994.7052

[5 rows x 12 columns]
```

To view the information on data frame, we used the pandas data frame info() method as follows:

```
# View the information of the dataframe, col name, data type, total entries etc.
print(df.info())
```

```
# View the information of the dataframe, col name, data type, total entries etc.

print(df.info())
```

Below screenshot displays the data frame information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            8523 non-null   object
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object
 7   Outlet_Establishment_Year  8523 non-null   int64
 8   Outlet_Size                6113 non-null   object
 9   Outlet_Location_Type       8523 non-null   object
 10  Outlet_Type                8523 non-null   object
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
None
```

For data cleaning purposes, we used 3 different techniques based on our scenario which are explained in-depth as follows:

1. Missing Values

To inspect the missing values in the dataset, we used the isnull().sum() function of python for our columns.

Before Cleaning:

```
# To see the null columns that will depict the missing values
print(df.isnull().sum())

print('\n Item_Weight no. of null rows:' + str(df['Item_Weight'].isnull().sum()))
print('\n Outlet_Size no. of null rows:' + str(df['Outlet_Size'].isnull().sum()))
print('\n')
```

```
# To see the null columns that will depict the missing values
print(df.isnull().sum())

print('\n Item_Weight no. of null rows:' + str(df['Item_Weight'].isnull().sum()))
print('\n Outlet_Size no. of null rows:' + str(df['Outlet_Size'].isnull().sum()))
print('\n')
```

```
Item_Identifier                 0
Item_Weight                  1463
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Size                  2410
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales               0
dtype: int64

 Item_Weight no. of null rows:1463

 Outlet_Size no. of null rows:2410
```

It is clear that the two columns – Item_Weight and Outlet_Size in our dataset had large number of missing values. We can also clearly see the null values in both the column Item_Weight and Out_Size when we print the top 15 rows:

```
# To View the rows that contain null values in Item_Weight column or Outlet_Size column
print(df.head(15))
```

```
# To view the rows that contain null values in Item_Weight column or
Outlet_size
print(df.head(15))
```

Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility
FDA15	9.300	Low Fat	0.016047
DRC01	5.920	Regular	0.019278
FDN15	17.500	LF	0.016760
FDX07	19.200	Regular	0.000000
NCD19	8.930	Low Fat	0.000000
FDP36	10.395	Regular	0.000000
FDO10	13.650	Regular	0.012741
FDP10	NaN	Low Fat	0.127470
FDH17	16.200	Regular	0.016687
FDU28	19.200	Regular	0.094450
FDY07	11.800	Low Fat	0.000000
FDA03	18.500	reg	0.045464
FDX32	15.100	Regular	0.100014
FDS46	17.600	Regular	0.047257
FDF32	16.350	Low Fat	0.068024

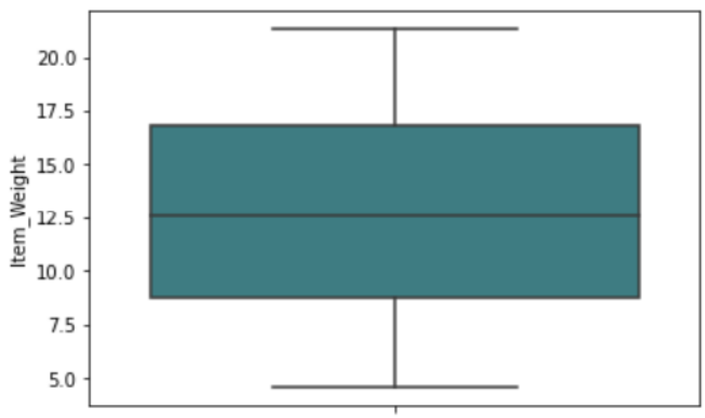
Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type
1999	Medium	Tier 1
2009	Medium	Tier 3
1999	Medium	Tier 1
1998	NaN	Tier 3
1987	High	Tier 3
2009	Medium	Tier 3
1987	High	Tier 3
1985	Medium	Tier 3
2002	NaN	Tier 2
2007	NaN	Tier 2
1999	Medium	Tier 1
1997	Small	Tier 1
1999	Medium	Tier 1
1997	Small	Tier 1
1987	High	Tier 3

To handle the missing values in python, we could use Mean or Median for numerical data and Mode for categorical data.

Because the Item_Weight column contains numerical values, we would either have to choose Mean or Median. To decide on which one would be used, we first checked the presence of outliers using a boxplot as follows:

```
# Box plot to check the outliers
sns.boxplot(x=None, y='Item_Weight', data=df, palette = 'crest')
plt.show()
```


```
# Box plot to check the outliers
sns.boxplot(x=None, y='Item_Weight', data=df, palette = 'crest')
plt.show()
```

Since we do not have any outliers, we decided to use the Mean to fill up our missing values.

```
# Calculate the mean of Item_Weight column that has null values
item_weight_mean = df.Item_Weight.mean()

# Replacing the missing values in the Item Weight column with the mean value
df['Item_Weight'] = df.Item_Weight.fillna(item_weight_mean)
```

```
# Calculate the mean of Item_Weight column that has null values
item_weight_mean = df.Item_Weight.mean()
```

```
# Replacing the missing values in the Item Weight column with the mean value
df['Item_Weight'] = df.Item_Weight.fillna(item_weight_mean)
```

Since the Outlet_Size column contains categorical data, we handled the missing values using Mode.

```
# Calculate the mode of Outlet_Size column that has null values
outlet_size_mode = df.Outlet_Size.mode()[0]

# Replacing the missing values in the Outlet Size column with the mode value
df['Outlet_Size'] = df.Outlet_Size.fillna(outlet_size_mode)
```

```
# Calculate the mode of Outlet_Size column that has null values
outlet_size_mode = df.Outlet_Size.mode()[0]

# Replacing the missing values in the Outlet Size column with the mode value
df['Outlet_Size'] = df.Outlet_Size.fillna(outlet_size_mode)
```

After Cleaning:

To verify if the Item_Weight and Outlet_Size missing values were handled,

```
# Check if the Item_Weight & Outlet_Size missing values have been filled
print(df.isnull().sum())

# Alternatively,
print('\n Item_Weight no. of null rows:' + str(df['Item_Weight'].isnull().sum()))
print('\n Outlet_Size no. of null rows:' + str(df['Outlet_Size'].isnull().sum()))
print('\n')
```

```
# Check if the Item_Weight & Outlet_Size missing values have been filled
print(df.isnull().sum())

# Alternatively,
print('\n Item_Weight no. of null rows:' + str(df['Item_Weight'].isnull().sum()))
print('\n Outlet_Size no. of null rows:' + str(df['Outlet_Size'].isnull().sum()))
print('\n')
```

```
Item_Identifier              0
Item_Weight                  0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Size                  0
Outlet_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
dtype: int64

 Item_Weight no. of null rows:0

 Outlet_Size no. of null rows:0
```

To verify if the null values in the Item_Weight and Out_Size column, we again print the top 15 rows:

```
# To Verify if the null value are handled in Item_Weight column or Outlet_Size column
print(df.head(15))
```

```
# To view if the null values are handled in Item_Weight column or Outlet_size
print(df.head(15))
```

Item_Weight replaced by the mean value:

Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility
FDA15	9.300000	Low Fat	0.016047
DRC01	5.920000	Regular	0.019278
FDN15	17.500000	LF	0.016760
FDX07	19.200000	Regular	0.000000
NCD19	8.930000	Low Fat	0.000000
FDP36	10.395000	Regular	0.000000
FDO10	13.650000	Regular	0.012741
FDP10	12.857645	Low Fat	0.127470
FDH17	16.200000	Regular	0.016687
FDU28	19.200000	Regular	0.094450
FDY07	11.800000	Low Fat	0.000000
FDA03	18.500000	reg	0.045464
FDX32	15.100000	Regular	0.100014
FDS46	17.600000	Regular	0.047257
FDF32	16.350000	Low Fat	0.068024

Outlet_Size replaced by mode value:

Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type
1999	Medium	Tier 1
2009	Medium	Tier 3
1999	Medium	Tier 1
1998	Medium	Tier 3
1987	High	Tier 3
2009	Medium	Tier 3
1987	High	Tier 3
1985	Medium	Tier 3
2002	Medium	Tier 2
2007	Medium	Tier 2
1999	Medium	Tier 1
1997	Small	Tier 1
1999	Medium	Tier 1
1997	Small	Tier 1
1987	High	Tier 3

2. **Duplicate Values**

We can see from the screenshot below that Rows 0, 1 and 3,4 are duplicate rows with exactly same values in all the columns. Hence, we would be removing duplicated values using the code below. For this purpose we used drop_duplicates() function.

Before Cleaning:

	Item_Identifier	Item_Weight	...	Outlet_Type	Item_Outlet_Sales
0	FDA15	9.300	...	Supermarket Type1	3735.1380
1	FDA15	9.300	...	Supermarket Type1	3735.1380
2	DRC01	5.920	...	Supermarket Type2	443.4228
3	FDN15	17.500	...	Supermarket Type1	2097.2700
4	FDN15	17.500	...	Supermarket Type1	2097.2700
5	FDX07	19.200	...	Grocery Store	732.3800
6	NCD19	8.930	...	Supermarket Type1	994.7052
7	FDP36	10.395	...	Supermarket Type2	556.6088
8	FDO10	13.650	...	Supermarket Type1	343.5528
9	FDP10	NaN	...	Supermarket Type3	4022.7636

Code:

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 22 19:01:40 2022

@author: spandit3
"""
import pandas as pd

#read in the file: df
df = pd.read_csv('BigMart.csv')
print(df.head(10))

#function to Drop/Remove Duplicates
df.drop_duplicates()

print(df.drop_duplicates().head(10))
```

```
import pandas as pd

# Read the CSV file into Dataframe
df = pd.read_csv('BigMart.csv')

# DROPPING OR REMOVING DUPLICATES
df.drop_duplicates()

print(df.drop_duplicates().head(10))
```

After Cleaning:

```
   Item_Identifier  Item_Weight  ...        Outlet_Type  Item_Outlet_Sales
0            FDA15        9.300  ...  Supermarket Type1          3735.1380
2            DRC01        5.920  ...  Supermarket Type2           443.4228
3            FDN15       17.500  ...  Supermarket Type1          2097.2700
5            FDX07       19.200  ...      Grocery Store           732.3800
6            NCD19        8.930  ...  Supermarket Type1           994.7052
7            FDP36       10.395  ...  Supermarket Type2           556.6088
8            FDO10       13.650  ...  Supermarket Type1           343.5528
9            FDP10          NaN  ...  Supermarket Type3          4022.7636
10           FDH17       16.200  ...  Supermarket Type1          1076.5986
11           FDU28       19.200  ...  Supermarket Type1          4710.5350

[10 rows x 12 columns]
```

After executing the code to drop the duplicates, we could observe that the rows 1 and 4 were dropped which were duplicate rows.

3. **Cleaning Data Inconsistency / Data Consistency -**

While inspecting the correctness of the column values, we encountered some discrepancies in Item_Fat_Content column. This column had 4 different types of values – Low Fat, Regular, LF, and reg. We wanted to have consistent data values and decided to have only 2 values throughout the column which was either Low Fat or Regular. The data values LF and reg looked like a data entry inconsistency and created confusion. Hence, we first simply displayed the top 15 rows from our data frame after setting the column width to view the discrepancies.

```
# Setting display , here max_rows, max_cols and col_width set to None
pd.set_option('display.max_rows', None, 'display.max_columns', None, 'display.max_colwidth', None)

# View the top 5 values
print(df.head(15))
```

```
pd.set_option('display.max_rows', None, 'display.max_columns', None,
'display.max_colwidth', None)

# To view the rows that contain null values in Item_Weight column or
Outlet_size
print(df.head(15))
```

Before Cleaning:

```
Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility  \
           FDA15        9.300          Low Fat         0.016047
           DRC01        5.920          Regular         0.019278
           FDN15       17.500               LF         0.016760
           FDX07       19.200          Regular         0.000000
           NCD19        8.930          Low Fat         0.000000
           FDP36       10.395          Regular         0.000000
           FDO10       13.650          Regular         0.012741
           FDP10          NaN          Low Fat         0.127470
           FDH17       16.200          Regular         0.016687
           FDU28       19.200          Regular         0.094450
           FDY07       11.800          Low Fat         0.000000
           FDA03       18.500              reg         0.045464
           FDX32       15.100          Regular         0.100014
           FDS46       17.600          Regular         0.047257
           FDF32       16.350          Low Fat         0.068024
```

To fix these discrepancies, we used 2 different methods:

a. String Replace function – to replace an incorrect value with a correct value to make the data consistent

b. Dataframe.loc[] function in pandas – to access a column and change its values with a condition

Code:

```
# HANDLING INCORRECT VALUES in Item_Fat_Content column
# Values present in this column were Low Fat, Regular, ref and LF
# Updated the values to only 2 categories - Low Fat and Regular

df = df.replace(to_replace ='LF', value='Low Fat', regex = True)

# Alternatively used .loc[] function of pandas
df.loc[df['Item_Fat_Content'] == "reg", 'Item_Fat_Content'] = 'Regular'

pd.set_option('display.max_rows', None, 'display.max_columns', None, 'display.max_colwidth', None)

print(df.head(15))
```

```
# HANDLING INCORRECT VALUES in Item_Fat_Content column
# Values present in this column were Low Fat, Regular, ref and LF
# Updated the values to only 2 categories - Low Fat and Regular

df = df.replace(to_replace ='LF', value='Low Fat', regex = True)
```

```
# Alternatively used .loc[] function of pandas
df.loc[df['Item_Fat_Content'] == "reg", 'Item_Fat_Content'] = 'Regular'

pd.set_option('display.max_rows', None, 'display.max_columns', None,
'display.max_colwidth', None)

# To view the rows that contain null values in Item_Weight column or Outlet_size
print(df.head(15))
```

After Cleaning:

```
Item_Identifier  Item_Weight Item_Fat_Content  Item_Visibility  \
          FDA15     9.300000          Low Fat         0.016047
          DRC01     5.920000          Regular         0.019278
          FDN15    17.500000          Low Fat         0.016760
          FDX07    19.200000          Regular         0.000000
          NCD19     8.930000          Low Fat         0.000000
          FDP36    10.395000          Regular         0.000000
          FDO10    13.650000          Regular         0.012741
          FDP10    12.857645          Low Fat         0.127470
          FDH17    16.200000          Regular         0.016687
          FDU28    19.200000          Regular         0.094450
          FDY07    11.800000          Low Fat         0.000000
          FDA03    18.500000          Regular         0.045464
          FDX32    15.100000          Regular         0.100014
          FDS46    17.600000          Regular         0.047257
          FDF32    16.350000          Low Fat         0.068024
```

After performing the data cleaning, we wanted to create a new clean data file for creating our visualizations. For this purpose, we used the dataframe.to_csv() function of pandas to write the data to a new csv file as follows:

```
# Write the clean data to a new csv file
df.to_csv('BigMartClean.csv', index = False)
```


```
# Write the clean data to a new csv file
df.to_csv('BigMartClean.csv', index = False)
```

Index=False was used in the to_csv() function to avoid creating an Index column which python would have done by default. The new file got created in the same location as the .py file and the old .csv file

PC > Windows (C:) > CSULA > CSU LA > CSU LA > MSIS > Spring 2022 > CIS 5270 - Business Intelligence > Python Project > Big Mart Project

Name	Date modified	Type	Size
BigMart	4/22/2022 8:13 PM	Microsoft Excel Co...	850 KB
BigMartClean	4/22/2022 8:35 PM	Microsoft Excel Co...	894 KB
BigMartProject	4/22/2022 8:35 PM	PY File	3 KB

4. **Removing unnecessary data**

Additionally, in our data file, we found that one of the columns named Item_Visibility had values as ‘0’. We did not find this relevant and decided to remove all the rows that had Item_Visibility values as 0. To do so, we used the dataframe.drop() function along with dataframe.loc[] of pandas to specify the rows that specifically need to be removed.

To verify the rows with a value 0, we displayed the top 10 rows of our data frame specifically for the Item_Visbility column as follows:

```
# HANDLING REMOVING OF 0'S in the Item_Visibility column

# To view the 0's in the Item_Visibility column
print('Displaying top 10 values of Item_Visibility column \n' + str(df['Item_Visibility'].head(10)))
```

Before Cleaning:

```
Displaying top 10 values of Item_Visibility column
0    0.016047
1    0.019278
2    0.016760
3    0.000000
4    0.000000
5    0.000000
6    0.012741
7    0.127470
8    0.016687
9    0.094450
Name: Item_Visibility, dtype: float64
```

Code:

To remove these unnecessary rows containing 0, we wrote the following code,

```
# Code to drop 0's in the Item_Visiblity column
df.drop(df.loc[df['Item_Visibility']==0].index, inplace=True)

# Verify if the rows containing 0's in the Item_Visibility column are dropped
print('Verifying if values in Item_Visibility column containg 0 are removed \n' + str(df['Item_Visibility'].head(10)))
```

```
# Code to drop 0's in the Item_Visiblity column
df.drop(df.loc[df['Item_Visibility']==0].index, inplace=True)

# Verify if the rows containing 0's in the Item_Visibility column are dropped
print('Verifying if values in Item_Visibility column containg 0 are removed \n' +
str(df['Item_Visibility'].head(10)))
```

After Cleaning:

```
Verifying if values in Item_Visibility column containg 0 are removed
0     0.016047
1     0.019278
2     0.016760
6     0.012741
7     0.127470
8     0.016687
9     0.094450
11    0.045464
12    0.100014
13    0.047257
Name: Item_Visibility, dtype: float64
```

It is clear that the rows with the index values 3, ,4 and 5 that contained a value 0 in the Item_Visiblity column have been dropped.

D. <u>Summary Statistics</u>

i. Summary Statistics for the Item Outlet Sales column

The below code is used to view the summary statistics for the Item_Outlet_Sales column of our dataset. The summary statistics can be shown using the describe() or by using the individual functions as shown below in the code snippet:

```
# SUMMARY STATISTICS

# Statistical Summary for the Column Item_Outlet_Sales

print('Summary of the Item Outlet Sales column:\n' + str(df['Item_Outlet_Sales'].describe()))

print('\nPrinting individual statistics for the Item Outlet Sales:')

print('\nMean of the Sales:', "{0:.2f}".format(df.Item_Outlet_Sales.mean()))
print('\nStandard deviation of the Sales:', "{0:.2f}".format(df.Item_Outlet_Sales.std()))
print('\nMinimum Sales:', "{0:.2f}".format(df.Item_Outlet_Sales.min()))
print('\n25th Percentile of the Sales:', "{0:.2f}".format(df.Item_Outlet_Sales.quantile(0.25)))
print('\nMedian of the Sales:', "{0:.2f}".format(df.Item_Outlet_Sales.quantile(0.5)))
print('\n75th Percentile of the Sales:', "{0:.2f}".format(df.Item_Outlet_Sales.quantile(0.75)))
print('\nMaximum Sales:', "{0:.2f}".format(df.Item_Outlet_Sales.max()))
print('\nMode of the Sales:', "{0:.2f}".format(statistics.mode(df.Item_Outlet_Sales)))
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
```

```
import statistics

warnings.filterwarnings('ignore')
# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')
# SUMMARY STATISTICS
# Statistical Summary for the Column Item_Outlet_Sales
print('Summary of the Item Outlet Sales column:\n' +
str(df['Item_Outlet_Sales'].describe()))

print('\nPrinting individual statistics for the Item Outlet Sales:')

print('\nMean of the Sales:', "{0:.2f}".format(df.Item_Outlet_Sales.mean()))
print('\nStandard deviation of the Sales:',
"{0:.2f}".format(df.Item_Outlet_Sales.std()))
print('\nMinimum Sales:', "{0:.2f}".format(df.Item_Outlet_Sales.min()))
print('\n25th Percentile of the Sales:',
"{0:.2f}".format(df.Item_Outlet_Sales.quantile(0.25)))
print('\nMedian of the Sales:',
"{0:.2f}".format(df.Item_Outlet_Sales.quantile(0.5)))
print('\n75th Percentile of the Sales:',
"{0:.2f}".format(df.Item_Outlet_Sales.quantile(0.75)))
print('\nMaximum Sales:', "{0:.2f}".format(df.Item_Outlet_Sales.max()))
print('\nMode of the Sales:',
"{0:.2f}".format(statistics.mode(df.Item_Outlet_Sales)))
```

The output of the above code is as follows:

```
In [1]: runfile('C:/CSULA/CSU LA/CSU LA/MSIS/Spring 2022/CIS 5270 - Business Intelligence/Python
Project/Big Mart Project/ForScreenshot.py', wdir='C:/CSULA/CSU LA/CSU LA/MSIS/Spring 2022/CIS
5270 - Business Intelligence/Python Project/Big Mart Project')
Summary of the Item Outlet Sales column:
count     7997.000000
mean      2178.575445
std       1704.227930
min         33.290000
25%        829.586800
50%       1794.331000
75%       3098.633200
max      13086.964800
Name: Item_Outlet_Sales, dtype: float64

Printing individual statistics for the Item Outlet Sales:

Mean of the Sales: 2178.58

Standard deviation of the Sales: 1704.23

Minimum Sales: 33.29

25th Percentile of the Sales: 829.59

Median of the Sales: 1794.33

75th Percentile of the Sales: 3098.63

Maximum Sales: 13086.96

Mode of the Sales: 958.75
```

The summary statistics depict that the minimum Item Outlet Sales value of the item/product is 33.29 and maximum Sales value is 13086.96. The mean value of Item Outlet Sales is 2178.58 and the 75th percentile of Item Outlet Sales Value is 3098.63 which indicates that most of the products in BigMart store are closer to the minimum Item Outlet Sales priced products. The median value which is the middle value for the Item Outlet Sales column is 1794.33 and the mode for the same column is 958.75 which indicates that this is the sales amount that appears most frequently in the dataset.

ii. Summary Statistics for the Item MRP column

The below code is used to view the summary statistics for the Item_MRP column of our dataset. The summary statistics can be shown using the describe() function or by using the individual functions as shown below in the code snippet:

```python
# Statistical Summary for the Column Item_MRP

print('\nSummary of the Item MRP column:\n' + str(df['Item_MRP'].describe()))

print('\nPrinting individual statistics for the Item MRP:')

print('\nMean of the MRP:', "{0:.2f}".format(df.Item_MRP.mean()))
print('\nStandard deviation of the MRP:', "{0:.2f}".format(df.Item_MRP.std()))
print('\nMinimum MRP:', "{0:.2f}".format(df.Item_MRP.min()))
print('\n25th Percentile of the MRP:', "{0:.2f}".format(df.Item_MRP.quantile(0.25)))
print('\nMedian of the MRP:', "{0:.2f}".format(df.Item_MRP.quantile(0.5)))
print('\n75th Percentile of the MRP:', "{0:.2f}".format(df.Item_MRP.quantile(0.75)))
print('\nMaximum MRP:', "{0:.2f}".format(df.Item_MRP.max()))
print('\nMode of the MRP:', "{0:.2f}".format(statistics.mode(df.Item_MRP)))
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import statistics

warnings.filterwarnings('ignore')

# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')

# SUMMARY STATISTICS
# Statistical Summary for the Column Item_MRP

print('\nSummary of the Item MRP column:\n' +
str(df['Item_MRP'].describe()))

print('\nPrinting individual statistics for the Item MRP:')

print('\nMean of the MRP:', "{0:.2f}".format(df.Item_MRP.mean()))
print('\nStandard deviation of the MRP:',
"{0:.2f}".format(df.Item_MRP.std()))
print('\nMinimum MRP:', "{0:.2f}".format(df.Item_MRP.min()))
print('\n25th Percentile of the MRP:',
"{0:.2f}".format(df.Item_MRP.quantile(0.25)))
print('\nMedian of the MRP:', "{0:.2f}".format(df.Item_MRP.quantile(0.5)))
print('\n75th Percentile of the MRP:',
"{0:.2f}".format(df.Item_MRP.quantile(0.75)))
print('\nMaximum MRP:', "{0:.2f}".format(df.Item_MRP.max()))
print('\nMode of the MRP:', "{0:.2f}".format(statistics.mode(df.Item_MRP)))
```

The output of the above code is as follows:

```
Summary of the Item MRP column:
count    7997.000000
mean      141.181925
std        62.201545
min        31.290000
25%        94.109400
50%       143.215400
75%       185.758200
max       266.888400
Name: Item_MRP, dtype: float64

Printing individual statistics for the Item MRP:

Mean of the MRP: 141.18

Standard deviation of the MRP: 62.20

Minimum MRP: 31.29

25th Percentile of the MRP: 94.11

Median of the MRP: 143.22

75th Percentile of the MRP: 185.76

Maximum MRP: 266.89

Mode of the MRP: 142.02
```

The summary statistics depict that the minimum MRP value of the item/product is 31.29 and maximum MRP value is 266.88. The mean value of MRP is 141.18 and the 75th percentile of MRP is 185.76 which indicates that most of the products in BigMart stores are closer to the maximum MRP priced products. The median value which is the middle value for the MRP column is 143.22 and the mode for the same column is 142.02 which indicates that this is the MRP amount that appears most frequently in the dataset.

iii. Summary Statistics for the Item Weight column

The below code is used to view the summary statistics for the Item_Weight column of our dataset. The summary statistics can be shown using the .describe() or by using the individual functions as shown below in the code snippet:

```
# Statistical Summary for the Column Item_Weight

print('\nSummary of the Item Weight column:\n' + str(df['Item_Weight'].describe()))

print('\nPrinting individual statistics for the Item Weight:')

print('\nMean of the Item Weight:', "{0:.2f}".format(df.Item_Weight.mean()))
print('\nStandard deviation of the Item Weight:', "{0:.2f}".format(df.Item_Weight.std()))
print('\nMinimum Item Weight:', "{0:.2f}".format(df.Item_Weight.min()))
print('\n25th Percentile of the Item Weight:', "{0:.2f}".format(df.Item_Weight.quantile(0.25)))
print('\nMedian of the Item Weight:', "{0:.2f}".format(df.Item_Weight.quantile(0.5)))
print('\n75th Percentile of the Item Weight:', "{0:.2f}".format(df.Item_Weight.quantile(0.75)))
print('\nMaximum Item Weight:', "{0:.2f}".format(df.Item_Weight.max()))
print('\nMode of the Item Weight:', "{0:.2f}".format(statistics.mode(df.Item_Weight)))
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import statistics

warnings.filterwarnings('ignore')

# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')

# SUMMARY STATISTICS

# Statistical Summary for the Column Item_Weight

print('\nSummary of the Item Weight column:\n' +
str(df['Item_Weight'].describe()))

print('\nPrinting individual statistics for the Item Weight:')

print('\nMean of the Item Weight:', "{0:.2f}".format(df.Item_Weight.mean()))
```

```
print('\nStandard deviation of the Item Weight:', "{0:.2f}".format(df.Item_Weight.std()))
print('\nMinimum Item Weight:', "{0:.2f}".format(df.Item_Weight.min()))
print('\n25th Percentile of the Item Weight:', "{0:.2f}".format(df.Item_Weight.quantile(0.25)))
print('\nMedian of the Item Weight:', "{0:.2f}".format(df.Item_Weight.quantile(0.5)))
print('\n75th Percentile of the Item Weight:', "{0:.2f}".format(df.Item_Weight.quantile(0.75)))
print('\nMaximum Item Weight:', "{0:.2f}".format(df.Item_Weight.max()))
print('\nMode of the Item Weight:', "{0:.2f}".format(statistics.mode(df.Item_Weight)))
```

The output of the above code is as follows:

```
Summary of the Item Weight column:
count    7997.000000
mean       12.873231
std         4.226817
min         4.555000
25%         9.310000
50%        12.857645
75%        16.100000
max        21.350000
Name: Item_Weight, dtype: float64

Printing individual statistics for the Item Weight:

Mean of the Item Weight: 12.87

Standard deviation of the Item Weight: 4.23

Minimum Item Weight: 4.55

25th Percentile of the Item Weight: 9.31

Median of the Item Weight: 12.86

75th Percentile of the Item Weight: 16.10

Maximum Item Weight: 21.35

Mode of the Item Weight: 12.86
```

The summary statistics depict that the minimum Weight of the item/product is 4.55 and maximum Weight of the item/product is 21.33. The mean value of Item

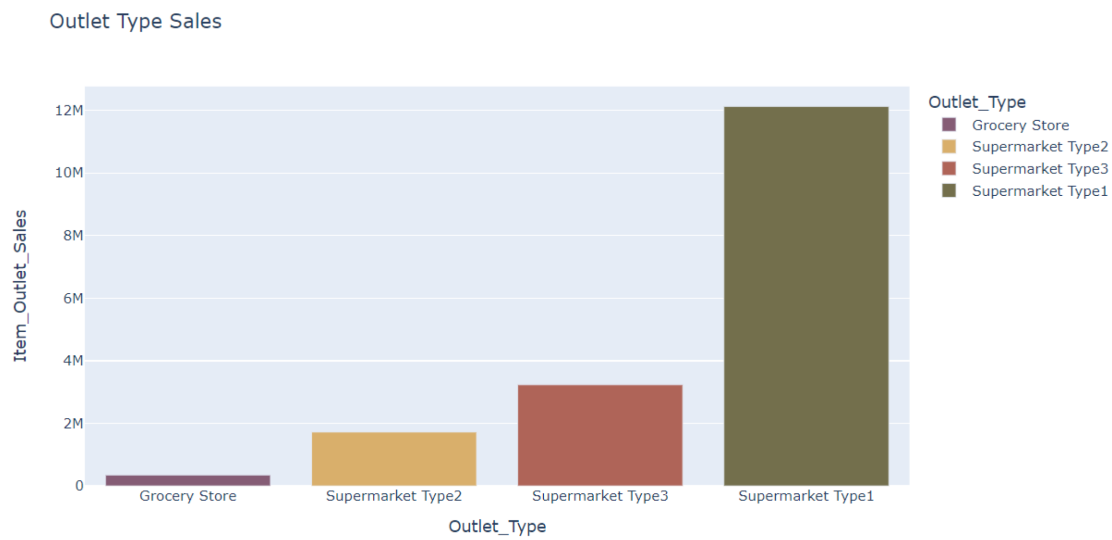
Weight is 12.87 and the 75th percentile of Item Weight is 16.10 which indicates that most of the products in BigMart stores are closer to the Maximum Weight of the products. The median value which is the middle value for the Weight column is 12.86 units and the mode for the same column is also 12.86 which indicates that this is the weight in units that appears most frequently in the dataset.

Code:

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

# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')

# SUMMARY STATISTICS

# Statistical Summary for the Column Item_Weight

print('\nSummary of the Item Weight column:\n' +
str(df['Item_Weight'].describe()))

print('\nPrinting individual statistics for the Item Weight:')

print('\nMean of the Item Weight:', "{0:.2f}".format(df.Item_Weight.mean()))
print('\nStandard deviation of the Item Weight:',
"{0:.2f}".format(df.Item_Weight.std()))
print('\nMinimum Item Weight:', "{0:.2f}".format(df.Item_Weight.min()))
print('\n25th Percentile of the Item Weight:',
"{0:.2f}".format(df.Item_Weight.quantile(0.25)))
print('\nMedian of the Item Weight:',
"{0:.2f}".format(df.Item_Weight.quantile(0.5)))
print('\n75th Percentile of the Item Weight:',
"{0:.2f}".format(df.Item_Weight.quantile(0.75)))
print('\nMaximum Item Weight:', "{0:.2f}".format(df.Item_Weight.max()))
```

E. <u>Data Visualization</u>

1. Does the Outlet Type (grocery store/supermarket type) have any impact on the overall sales?



Plot Type: Bar Chart

Libraries: matplotlib.pyplot, plotly, pandas

Methods: read_csv(), groupby(), sum(), show(), bar()

Insights:

This graph was obtained in combination of Jupyter Notebook IDE and plotly.express library. The bar graph above shows the relation between the Item Outlet sales and the Outlet Type. The Outlet type refers to the type of outlet - if it is a Supermarket or a Small Grocery Store. The Supermarkets are further sub-divided into 1,2 and 3 depending upon the size of the Supermarket, where 1 is the smallest and 3 is the largest. This analysis was achieved using the matplot library and seaborn package. As it is clear from the graph above that the Supermarket of Type 1 has the highest Sales whereas the Grocery Store has the lowest Sales. Hence, we can conclude that people tend to buy more from a mid-sized supermarket than going to a local Grocery store. However, other factors like the location of the Supermarket, and the higher availability of goods in the supermarket than in a Grocery Shop can also influence the overall sales.

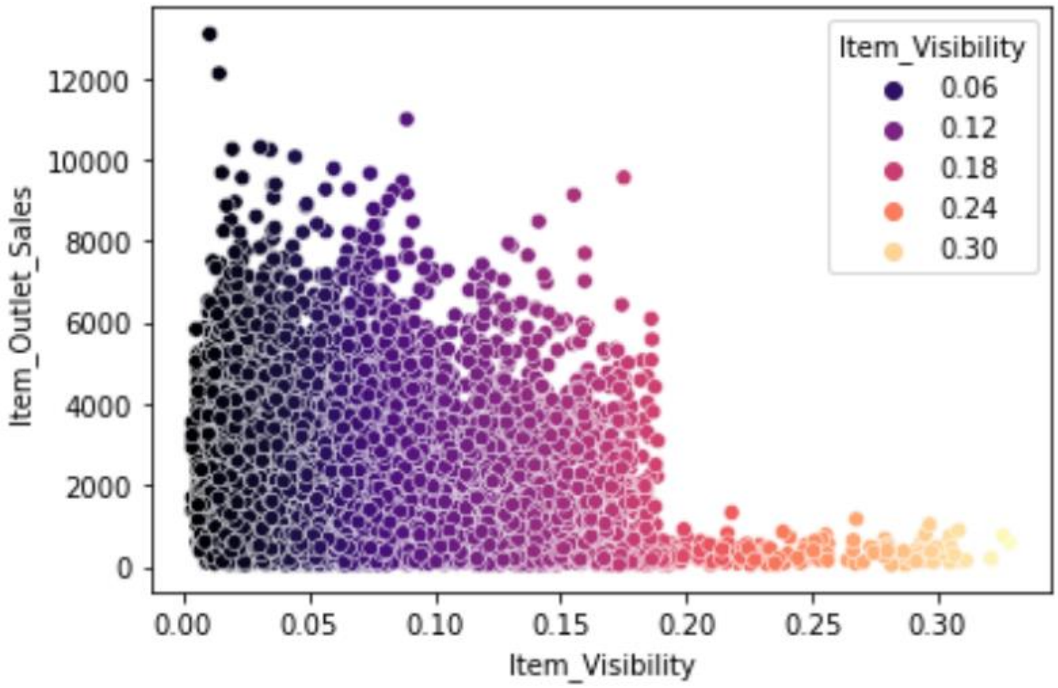
Code Screenshot:

```
In [3]: pip install plotly==5.7.0

        Requirement already satisfied: plotly==5.7.0 in c:\programdata\anaconda3\lib\site-packages (5.7.0)Note: you may need to restart
        the kernel to use updated packages.

        Requirement already satisfied: tenacity>=6.2.0 in c:\programdata\anaconda3\lib\site-packages (from plotly==5.7.0) (8.0.1)
        Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packages (from plotly==5.7.0) (1.16.0)

In [13]: import plotly.express as px
         import pandas as pd
         import matplotlib.pyplot as plt

         df = pd.read_csv('BigMartClean.csv')
         data = df.groupby("Outlet_Type")[["Item_Outlet_Sales"]].sum().sort_values(by=['Item_Outlet_Sales'],
                                                                                    ascending=[True]).reset_index()
         fig = px.bar(data, x = 'Outlet_Type', y = 'Item_Outlet_Sales', title='Outlet Type Sales', color='Outlet_Type',
                      color_discrete_sequence=px.colors.qualitative.Antique)
         fig.show()
```

Code:

```
pip install plotly==5.7.0

import plotly.express as px
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('BigMartClean.csv')
data =
df.groupby("Outlet_Type")[["Item_Outlet_Sales"]].sum().sort_values(by=['Ite
m_Outlet_Sales'],
                                        ascending=[True]).reset_index()
fig = px.bar(data, x = 'Outlet_Type', y = 'Item_Outlet_Sales', title='Outlet Type
Sales', color='Outlet_Type',
color_discrete_sequence=px.colors.qualitative.Antique)
fig.show()
```

2. **Does Item Visibility have any impact on the overall sales?**



Plot Type: Scatter Plot

Libraries: matplotlib.pyplot, seaborn, pandas

Methods: read_csv(), unique(), groupby(), sort_values(), reset_index(), show()

Insights:

The visibility index is an indicator for the visibility of an item or a product on the website or in-store. When talking about in-store, the visibility index will indicate the percentage of the overall viewing area assigned to a particular item from all the items in the store. The minimum value of the visibility index is 0.003 (or 0.03%) and the maximum value is 0.328 (or 32.8%). While analyzing if the item visibility index had a co-relation to the sales of the item, we observe that the items with visibility in the range of 0.06 to 0.18 have greater sales, and those with a visibility index of more than 0.24 end up getting less sold. This can be because bulkier items or large items will have greater visibility but will not have a daily sale as much as those in smaller sizes. When we walk into a Supermarket or a Grocery store such as Vons or Ralph's, we can see that large items such as garden furniture are easily visible, but because they are highly-priced or difficult to be carried around, they won't be bought frequently by customers; whereas other items such as Breakfast, Dairy, Baking goods, Health and Hygiene, etc. will have a greater sales on a day-to-day basis even though their visibility is lesser in the first go.

Code Screenshot:

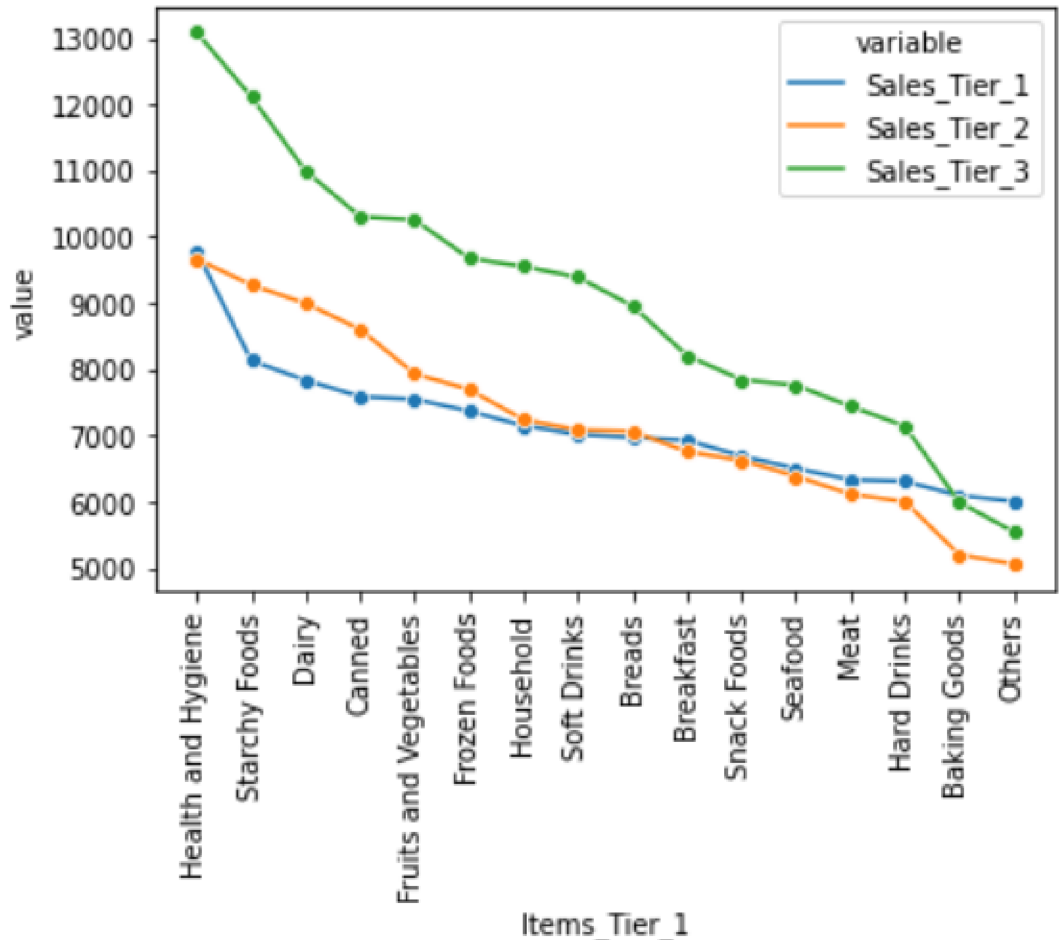
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings


warnings.filterwarnings('ignore')

# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')

# View the top 5 values
print(df.head())

# First get the unique values of the Item_Visibility column
df['Item_Visibility'].unique()

item_visibility_sales = df.groupby("Item_Visibility")[["Item_Outlet_Sales"]].sum().sort_values(by=['Item_Outlet_Sales'],
                                                                                                ascending=[False]).reset_index()

item_visibility_sales.sort_values(by=['Item_Outlet_Sales'],ascending=[False])

sns.scatterplot(data = df, x = 'Item_Visibility', y = 'Item_Outlet_Sales', hue='Item_Visibility', palette='magma')


plt.show()
```

Code:

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


# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')
```

```
# View the top 5 values
print(df.head())

# First get the unique values of the Item_Visibility column
df['Item_Visibility'].unique()

item_visibility_sales =
df.groupby("Item_Visibility")[["Item_Outlet_Sales"]].sum().sort_values(by=['It
em_Outlet_Sales'], ascending=[False]).reset_index()
item_visibility_sales.sort_values(by=['Item_Outlet_Sales'],ascending=[False])

sns.scatterplot(data = df, x = 'Item_Visibility', y = 'Item_Outlet_Sales',
hue='Item_Visibility', palette='magma')
plt.show()
```

3. **What are the items that have better sales in Tier 1 cities as compared to the sales of items in Tier 2 and Tier 3 cities?**



Plot Type: Line Chart with markers

Libraries: matplotlib.pyplot, seaborn, pandas

Methods: read_csv(), max(), groupby(), sort_values(), reset_index(), rename(), concat(), xticks(), show()

Insights:

Growth in population necessitates an increase in food production and the sales trends in the food and grocery market keep increasing with time. Tier 1 cities (such as NYC or LA) will have bigger-sized stores than Tier 2 and Tier 3 cities and will also have a wide variety of products that will lead to higher sales. However, to analyze if this assumption is true or not, we wanted to inspect if there were any specific food items that received better sales in Tier 1 cities than in Tier 2 or Tier 3 cities. We could observe from the above visualization that not many items' sales in Tier 1 city stores outperform the items' sales in Tier 2 or Tier 3 city stores. We could interpret that the baking goods, hard drinks, and meat had a little greater sales in the Tier 1 cities as compared to Tier 2 cities. However, while comparing the item sales of Tier 1 cities to Tier 3 cities, only the items that fall under the item type 'Others' had marginal sales differences. Also, looking at the above graph makes it clear that the items in Tier 2 and Tier 3 city stores clearly outperform the sales of those in Tier 1 city stores. Hence assuming that Tier 1 city stores that are bigger in size will have higher sales does not remain valid.

Code Screenshot:

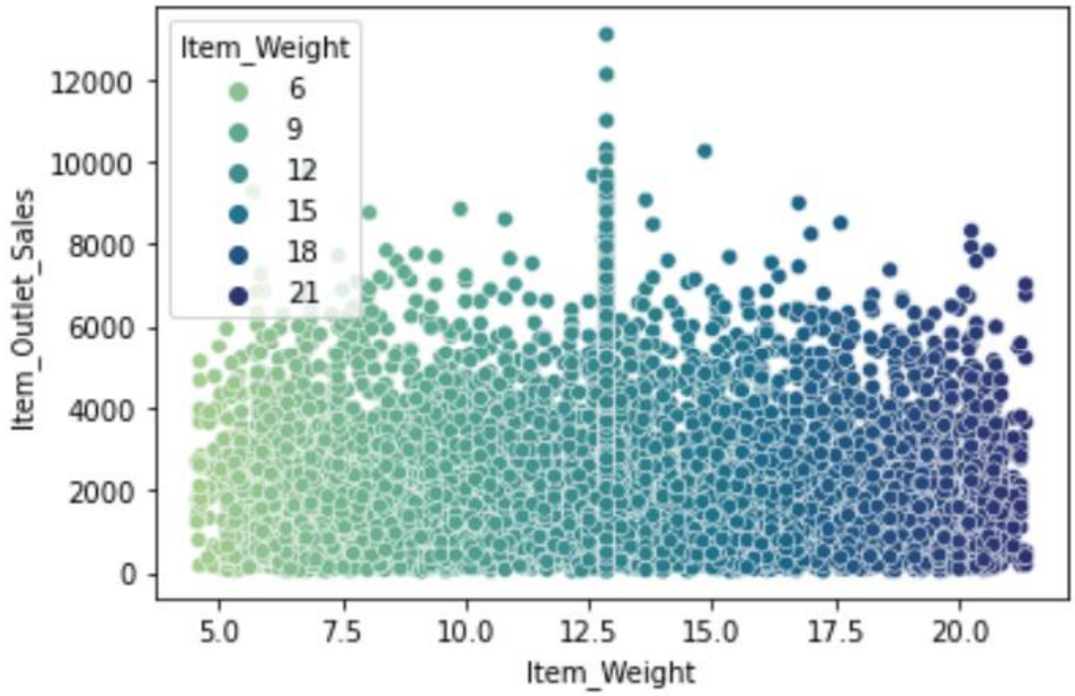
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

warnings.filterwarnings('ignore')

# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')

# View the top 5 values
print(df.head())

# Splitting the data
# Data frame for Tier 1
df_tier1 = pd.DataFrame(df.loc[df['Outlet_Location_Type'] =='Tier 1'])

# Data frame for Tier 2
df_tier2 = pd.DataFrame(df.loc[df['Outlet_Location_Type'] =='Tier 2'])

# Data frame for Tier 2
df_tier3 = pd.DataFrame(df.loc[df['Outlet_Location_Type'] =='Tier 3'])


# Calculating Sales for Items in Tier 1
df_tier1_list = df_tier1.groupby("Item_Type")[["Item_Outlet_Sales"]].max().sort_values(by=['Item_Outlet_Sales'],
                                                                                    ascending=[False]).reset_index()
df_tier1_list.rename(columns = {'Item_Type':'Items_Tier_1','Item_Outlet_Sales':'Sales_Tier_1'}, inplace = True)

# Calculating Sales for Items in Tier 2
df_tier2_list = df_tier2.groupby("Item_Type")[["Item_Outlet_Sales"]].max().sort_values(by=['Item_Outlet_Sales'],
                                                                                      ascending=[False]).reset_index()

df_tier2_list.rename(columns = {'Item_Type':'Items_Tier2','Item_Outlet_Sales':'Sales_Tier_2'}, inplace = True)

# Calculating Sales for Items in Tier 3
df_tier3_list = df_tier3.groupby("Item_Type")[["Item_Outlet_Sales"]].max().sort_values(by=['Item_Outlet_Sales'],
                                                                                      ascending=[False]).reset_index()

df_tier3_list.rename(columns = {'Item_Type':'Items_Tier3','Item_Outlet_Sales':'Sales_Tier_3'}, inplace = True)

# Concatenate the data frames for Tier 1 and Tier 2 and Tier 3
df_tot_sales = pd.concat([df_tier1_list, df_tier2_list, df_tier3_list], axis=1)

print(df_tot_sales)

del df_tot_sales['Items_Tier2']
del df_tot_sales['Items_Tier3']

sns.lineplot('Items_Tier_1', 'value', hue='variable', marker='o', data = pd.melt( df_tot_sales, 'Items_Tier_1'))
plt.xticks(rotation='vertical')

plt.show()
```

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')
# View the top 5 values
print(df.head())
# Splitting the data
# Data frame for Tier 1
df_tier1 = pd.DataFrame(df.loc[df['Outlet_Location_Type'] =='Tier 1'])
# Data frame for Tier 2
df_tier2 = pd.DataFrame(df.loc[df['Outlet_Location_Type'] =='Tier 2'])
# Data frame for Tier 2
df_tier3 = pd.DataFrame(df.loc[df['Outlet_Location_Type'] =='Tier 3'])
# Calculating Sales for Items in Tier 1
df_tier1_list =
df_tier1.groupby("Item_Type")[["Item_Outlet_Sales"]].max().sort_values(by=['
Item_Outlet_Sales'], ascending=[False]).reset_index()
df_tier1_list.rename(columns =
{'Item_Type':'Items_Tier_1','Item_Outlet_Sales':'Sales_Tier_1'}, inplace =
True)

#print(df_tier1_list)

# Calculating Sales for Items in Tier 2
```

```
df_tier2_list =
df_tier2.groupby("Item_Type")[["Item_Outlet_Sales"]].max().sort_values(by=['
Item_Outlet_Sales'], ascending=[False]).reset_index()

df_tier2_list.rename(columns =
{'Item_Type':'Items_Tier2','Item_Outlet_Sales':'Sales_Tier_2'}, inplace = True)
# Calculating Sales for Items in Tier 3
df_tier3_list =
df_tier3.groupby("Item_Type")[["Item_Outlet_Sales"]].max().sort_values(by=['
Item_Outlet_Sales'], ascending=[False]).reset_index()
df_tier3_list.rename(columns =
{'Item_Type':'Items_Tier3','Item_Outlet_Sales':'Sales_Tier_3'}, inplace = True)
#print(df_tier23_list)
# Concatenate the data frames for Tier 1 and Tier 2 and Tier 3
df_tot_sales = pd.concat([df_tier1_list, df_tier2_list, df_tier3_list], axis=1)
print(df_tot_sales)
del df_tot_sales['Items_Tier2']
del df_tot_sales['Items_Tier3']
#print(pd.melt( df_tot_sales, 'Items_Tier_1'))
sns.lineplot('Items_Tier_1', 'value', hue='variable', marker='o', data = pd.melt(
df_tot_sales, 'Items_Tier_1'))
plt.xticks(rotation='vertical')
plt.show()
```

4. **Does item weight have any impact on the overall sales? (Additional Analysis)**



Plot Type: Scatter Plot

Libraries: matplotlib.pyplot, seaborn, pandas

Methods: read_csv(), min(), groupby(), sort_values(), reset_index(), show()

Insights:

While analyzing if the item weight correlates with the sales of that item, we first wanted to check the minimum weight of the item that is being sold. Checking the minimum weight will also assure that no item is having a weight of 0 units. The minimum weight in our case was 4.555 units.

```
Minimum weight of the item is:4.555
```

Looking at the scatter plot, we could analyze that the items having a weight a little higher than 12.5 units have made the greatest sales. Even while we performed statistical analysis on this column (as shown in the section), we did observe that the mode and the median for the column weight were 12.6 units. Having mode as 12.6 units for the column does indicate that the frequency of having this data value is higher. This means that majority of the products are having a weight of 12.6 units. In the above visualization, we can clearly depict that the items having a weight of around 12.5 have the greatest sale. The items such as Meat, baking goods, Household, etc, weigh around 12.5 units and the above visualization clearly depicts that these goods are having higher sales as compared to other items.

Code Screenshot:

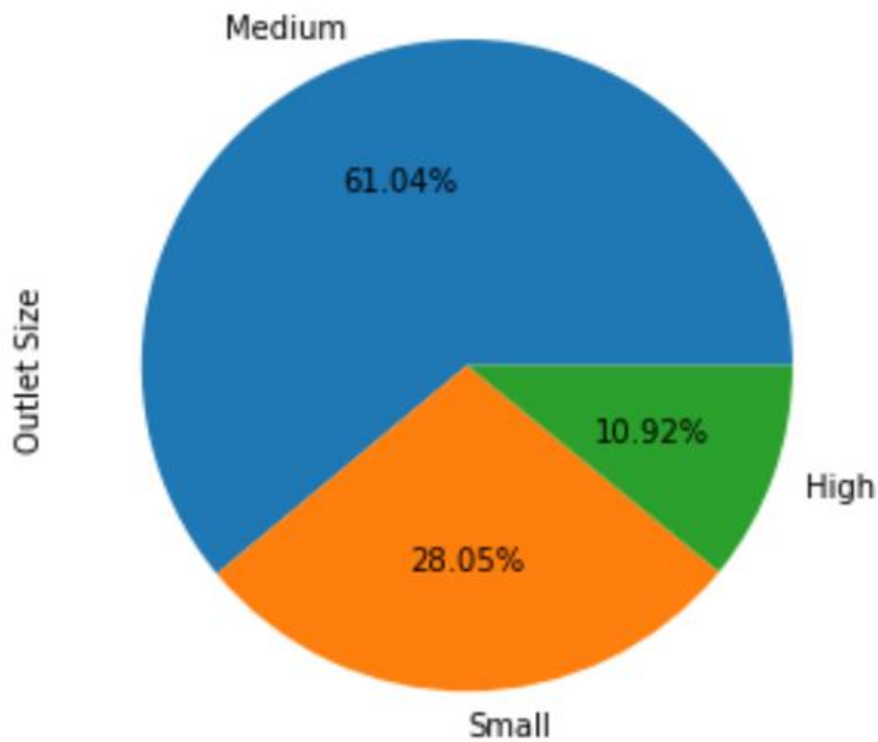
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings


warnings.filterwarnings('ignore')

# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')

# View the top 5 values
print(df.head())

# Check to see if the Item_Weight is greater than or equal to 0
df['Item_Weight'].min()

print('\n Minimum weight of the item is:' +  str(df['Item_Weight'].min()))

pd.set_option('display.float_format', lambda x: '%.3f' % x)
item_visibility_sales = df.groupby("Item_Weight")[["Item_Outlet_Sales"]].sum().sort_values(by=['Item_Outlet_Sales'],
                                                                                             ascending=[False]).reset_index()
item_visibility_sales.sort_values(by=['Item_Outlet_Sales'],ascending=[False])

sns.scatterplot(data = df, x = 'Item_Weight', y = 'Item_Outlet_Sales', hue='Item_Weight', palette='crest')

plt.show()
```

Code:

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

```
# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')
# View the top 5 values
print(df.head())
# Check to see if the Item_Weight is greater than or equal to 0
df['Item_Weight'].min()
print('\n Minimum weight of the item is:' +  str(df['Item_Weight'].min()))
pd.set_option('display.float_format', lambda x: '%.3f' % x)
item_visibility_sales =
df.groupby("Item_Weight")[["Item_Outlet_Sales"]].sum().sort_values(by=['Ite
m_Outlet_Sales'], ascending=[False]).reset_index()
item_visibility_sales.sort_values(by=['Item_Outlet_Sales'],ascending=[False])
sns.scatterplot(data = df, x = 'Item_Weight', y = 'Item_Outlet_Sales',
hue='Item_Weight', palette='crest')

plt.show()
```

5. **What is the percentage of Outlets of respective outlet size (Small, Medium, High)? (Additional Analysis)**



Plot Type: Pie Chart

Libraries: matplotlib.pyplot, pandas

Methods: read_csv(), pie(), value_counts()

Insights:

The Outlet Size refers to how big or small an outlet is in terms of ground area. We have used Matplot library for this analysis. It is evident from the Pie Chart above, that most of the Outlets (around 61% of the total outlets) are Medium-Sized outlets. These medium-sized outlets could be equivalent to other stores such as Vons or Ralphs or Sprouts which are there in the market. The second highest outlet type is the outlets that are Small-Sized (around 28.02%) and very few outlets are large in size, the percentage of High or Large

Outlet size being 10.94% (for e.g. consider on the lines of the alternative competition Costco which are very big outlets but less in number)

Code Screenshot:

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 22 19:01:40 2022

@author: spandit3
"""
import pandas as pd
import matplotlib.pyplot as plt


#read in the file: df
df = pd.read_csv('BigMartClean.csv')

counts = df['Outlet_Size'].value_counts()
print(counts)
counts.plot.pie(autopct='%.2f%%',label= "Outlet Size")
plt.tight_layout()
plt.show()
```

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
# Read the CSV file into Dataframe
df = pd.read_csv('BigMartClean.csv')
# View the top 5 values
print(df.head())
# Check to see if the Item_Weight is greater than or equal to 0
df['Item_Weight'].min()
print('\n Minimum weight of the item is:' +  str(df['Item_Weight'].min()))
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```

```
item_visibility_sales =
df.groupby("Item_Weight")[["Item_Outlet_Sales"]].sum().sort_values(by=['I
tem_Outlet_Sales'], ascending=[False]).reset_index()
item_visibility_sales.sort_values(by=['Item_Outlet_Sales'],ascending=[False
])
sns.scatterplot(data = df, x = 'Item_Weight', y = 'Item_Outlet_Sales',
hue='Item_Weight', palette='crest')
plt.show()
```

F. **References**

R. P and S. M, "Predictive Analysis for Big Mart Sales Using Machine Learning Algorithms," 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), 2021, pp. 1416-1421, doi: 10.1109/ICICCS51141.2021.9432109.

Analytics Vidhya. *An Intuitive Guide to Data Visualization in Python, Aishwarya Ajaykumar*, *February 2021*. Retrieved on May 14, 2022, from https://www.analyticsvidhya.com/blog/2021/02/an-intuitive-guide-to-visualization-in- python/