Importing Libraries

In [107…
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Loading the dataset

In [104…
```
df1 = pd.read_csv('Test.csv')
df2 = pd.read_csv('Train.csv')
```

Exploratort Data Analysis and Data Cleaning

In [106…
```
# Basic Data Exploration- Now in this step,we will perform the below operations to che
df1.head()
```

Out[106]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identi
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT

In [6]:
```
df2.head()
```

Out[6]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identi
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT

In [7]:
```
df1.tail()
```

Out[7]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Id
5676	FDB58	10.5	Regular	0.013496	Snack Foods	141.3154	
5677	FDD47	7.6	Regular	0.142991	Starchy Foods	169.1448	
5678	NCO17	10.0	Low Fat	0.073529	Health and Hygiene	118.7440	
5679	FDJ26	15.3	Regular	0.000000	Canned	214.6218	
5680	FDU37	9.5	Regular	0.104720	Canned	79.7960	

In [8]:
```
df2.tail()
```

Out[8]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Id
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	

In [9]:
```
#Let us see how many rows and columns are present in our data set.
df1.shape
```

Out[9]:
```
(5681, 11)
```

In [10]:
```
#Let us see how many rows and columns are present in our data set.
df2.shape
```

Out[10]:
```
(8523, 12)
```

Loading the data set,we will loading the EDA Item_Type using pandas

In [54]:
```
df2.boxplot(column=["Item_Outlet_Sales"])
plt.show
```

Out[54]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

In [55]:
```
df1.boxplot(column=["Item_MRP"])
plt.show
```

Out[55]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

In [11]:
```
# Let us see the columns in dataset.
df1.columns
```

Out[11]:
```
Index(['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'],
      dtype='object')
```

In [ ]:
```
# Let us see the columns in dataset.
```

In [12]:
```
df2.columns
```

Out[12]:
```
Index(['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'],
      dtype='object')
```

In [13]:
```
# Now we check the datatypes
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5681 entries, 0 to 5680
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Item_Identifier            5681 non-null   object 
 1   Item_Weight                4705 non-null   float64
 2   Item_Fat_Content           5681 non-null   object 
 3   Item_Visibility            5681 non-null   float64
 4   Item_Type                  5681 non-null   object 
 5   Item_MRP                   5681 non-null   float64
 6   Outlet_Identifier          5681 non-null   object 
 7   Outlet_Establishment_Year  5681 non-null   int64  
 8   Outlet_Size                4075 non-null   object 
 9   Outlet_Location_Type       5681 non-null   object 
 10  Outlet_Type                5681 non-null   object 
dtypes: float64(3), int64(1), object(7)
memory usage: 488.3+ KB
```

In [14]:
```
# Now we check the datatypes
df2.info()
```

```
<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
```

In [21]:
```
df1.nunique() # check unique values in each column
```

Out[21]:
```
Item_Identifier              1543
Item_Weight                   410
Item_Fat_Content                5
Item_Visibility              5277
Item_Type                      16
Item_MRP                     4402
Outlet_Identifier              10
Outlet_Establishment_Year       9
Outlet_Size                     3
Outlet_Location_Type            3
Outlet_Type                     4
dtype: int64
```

In [22]:
```
df2.nunique() # check unique values in each column
```

Out[22]:
```
Item_Identifier              1559
Item_Weight                   415
Item_Fat_Content                5
Item_Visibility              7880
Item_Type                      16
Item_MRP                     5938
Outlet_Identifier              10
Outlet_Establishment_Year       9
Outlet_Size                     3
Outlet_Location_Type            3
Outlet_Type                     4
Item_Outlet_Sales            3493
dtype: int64
```

In [25]:
```
# described method will help here to see how the data has been spread for numerical va
df1.describe()
```

Out[25]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year
count	4705.000000	5681.000000	5681.000000	5681.000000
mean	12.695633	0.065684	141.023273	1997.828903
std	4.664849	0.051252	61.809091	8.372256
min	4.555000	0.000000	31.990000	1985.000000
25%	8.645000	0.027047	94.412000	1987.000000
50%	12.500000	0.054154	141.415400	1999.000000
75%	16.700000	0.093463	186.026600	2004.000000
max	21.350000	0.323637	266.588400	2009.000000

In [26]:
```
# described method will help here to see how the data has been spread for numerical va
df2.describe()
```

Out[26]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	7060.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.643456	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	8.773750	0.026989	93.826500	1987.000000	834.247400
50%	12.600000	0.053931	143.012800	1999.000000	1794.331000
75%	16.850000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

Handling the missing values

In [23]:
```
df1.isnull().sum()
```

Out[23]:
```
Item_Identifier               0
Item_Weight                 976
Item_Fat_Content              0
Item_Visibility               0
Item_Type                     0
Item_MRP                      0
Outlet_Identifier             0
Outlet_Establishment_Year     0
Outlet_Size                1606
Outlet_Location_Type          0
Outlet_Type                   0
dtype: int64
```

In [24]:
```
df2.isnull().sum()
```

Out[24]:
```
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
```

In [63]:
```
# Here we remove the columns of missing values and if chages any dataframe then will u
df1.drop(['Item_Weight', 'Outlet_Size'], axis = 1, inplace = True)
df1.dropna(inplace = True)
```

In [64]:
```
df1.isnull().sum()
```

Out[64]:
```
Item_Identifier            0
Item_Fat_Content            0
Item_Visibility             0
Item_Type                   0
Item_MRP                    0
Outlet_Identifier           0
Outlet_Establishment_Year   0
Outlet_Location_Type        0
Outlet_Type                 0
dtype: int64
```

In [65]:
```
df2.drop(['Item_Weight', 'Outlet_Size'], axis = 1, inplace = True)
df2.dropna(inplace = True)
```

In [66]:
```
df2.isnull().sum()
```

Out[66]:
```
Item_Identifier              0
Item_Fat_Content             0
Item_Visibility              0
Item_Type                    0
Item_MRP                     0
Outlet_Identifier            0
Outlet_Establishment_Year    0
Outlet_Location_Type         0
Outlet_Type                  0
Item_Outlet_Sales            0
dtype: int64
```

In [67]:
```
df1.describe()
```

Out[67]:

	Item_Visibility	Item_MRP	Outlet_Establishment_Year
count	5681.000000	5681.000000	5681.000000
mean	0.065684	141.023273	1997.828903
std	0.051252	61.809091	8.372256
min	0.000000	31.990000	1985.000000
25%	0.027047	94.412000	1987.000000
50%	0.054154	141.415400	1999.000000
75%	0.093463	186.026600	2004.000000
max	0.323637	266.588400	2009.000000

In [27]:
```
df1.head()
```

Out[27]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identi
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT
1	FDW14	8.300	reg	0.038428	Dairy	87.3198	OUT
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT

In [34]:
```
df1.duplicated().sum() # check duplicate values
```

Out[34]:
```
0
```

In [35]:
```
df2.duplicated().sum() # check duplicate values
```

Out[35]:
```
0
```

Handling check the duplicates record of big mart items products.

In [59]:
```
duplicate = df1.duplicated()
print(duplicate.sum())
df1[duplicate]
```

```
0
```

Out[59]:

Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifi

In [36]:
```
# here some data type is objects data type that is categorical culumns so lets we chec
df1.describe(include = 'object')
```

Out[36]:

	Item_Identifier	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Size	Outlet_Location_Ty
count	5681	5681	5681	5681	4075	56
unique	1543	5	16	10	3	
top	DRF48	Low Fat	Snack Foods	OUT027	Medium	Tie
freq	8	3396	789	624	1862	22

In [37]:
```
# here some data type is objects data type that is categorical culumns so lets we chec
df2.describe(include = 'object')
```

Out[37]:

	Item_Identifier	Item_Fat_Content	Item_Type	Outlet_Identifier	Outlet_Size	Outlet_Location_Ty
count	8523	8523	8523	8523	6113	85
unique	1559	5	16	10	3	
top	FDW13	Low Fat	Fruits and Vegetables	OUT027	Medium	Tie
freq	10	5089	1232	935	2793	33

Data Analysis and visualizations

Here we predict the items outlet sales data as per given sheet. so we merge here both train and test data in EDA

In [77]:
```
df1['source'] = 'train'
df1['source'] = 'test'
```

```
df1['Item_Outlet_Sales'] = 0
data = pd.concat([df1, df2], sort = False)
print(df1.shape, df2.shape, data.shape)
```

```
(5681, 11) (8523, 10) (14204, 11)
```

In [78]:
```
data['Item_Outlet_Sales'].describe()
```

Out[78]:
```
count    14204.000000
mean      1308.865489
std       1699.791423
min          0.000000
25%          0.000000
50%        559.272000
75%       2163.184200
max      13086.964800
Name: Item_Outlet_Sales, dtype: float64
```

In [79]:
```
data['Item_MRP'].describe()
```

Out[79]:
```
count    14204.000000
mean       141.004977
std         62.086938
min         31.290000
25%         94.012000
50%        142.247000
75%        185.855600
max        266.888400
Name: Item_MRP, dtype: float64
```

In [81]:
```
import seaborn as sns
```

In [82]:
```
sns.distplot(data['Item_Outlet_Sales'])
```

Out[82]:
```
<Axes: xlabel='Item_Outlet_Sales', ylabel='Density'>
```

In [83]:
```
data.dtypes
```

Out[83]:
```
Item_Identifier               object
Item_Fat_Content              object
Item_Visibility              float64
Item_Type                     object
Item_MRP                     float64
Outlet_Identifier             object
Outlet_Establishment_Year      int64
Outlet_Location_Type          object
Outlet_Type                   object
source                        object
Item_Outlet_Sales            float64
dtype: object
```

In [89]:
```
data['Outlet_Establishment_Year'].value_counts()
```

Out[89]:
```
1985    2439
1987    1553
1999    1550
1997    1550
2004    1550
2002    1548
2009    1546
2007    1543
1998     925
Name: Outlet_Establishment_Year, dtype: int64
```

To check the missing values in the given dataset

In [93]:
```
data.apply(lambda x: sum(x.isnull()))
```

Out[93]:
```
Item_Identifier                 0
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Location_Type            0
Outlet_Type                     0
source                       8523
Item_Outlet_Sales               0
dtype: int64
```

In [94]:
```
data.apply(lambda x : len(x.unique()))
```

Out[94]:
```
Item_Identifier              1559
Item_Fat_Content                5
Item_Visibility             13006
Item_Type                      16
Item_MRP                     8052
Outlet_Identifier              10
Outlet_Establishment_Year       9
Outlet_Location_Type            3
Outlet_Type                     4
source                          2
Item_Outlet_Sales            3494
dtype: int64
```

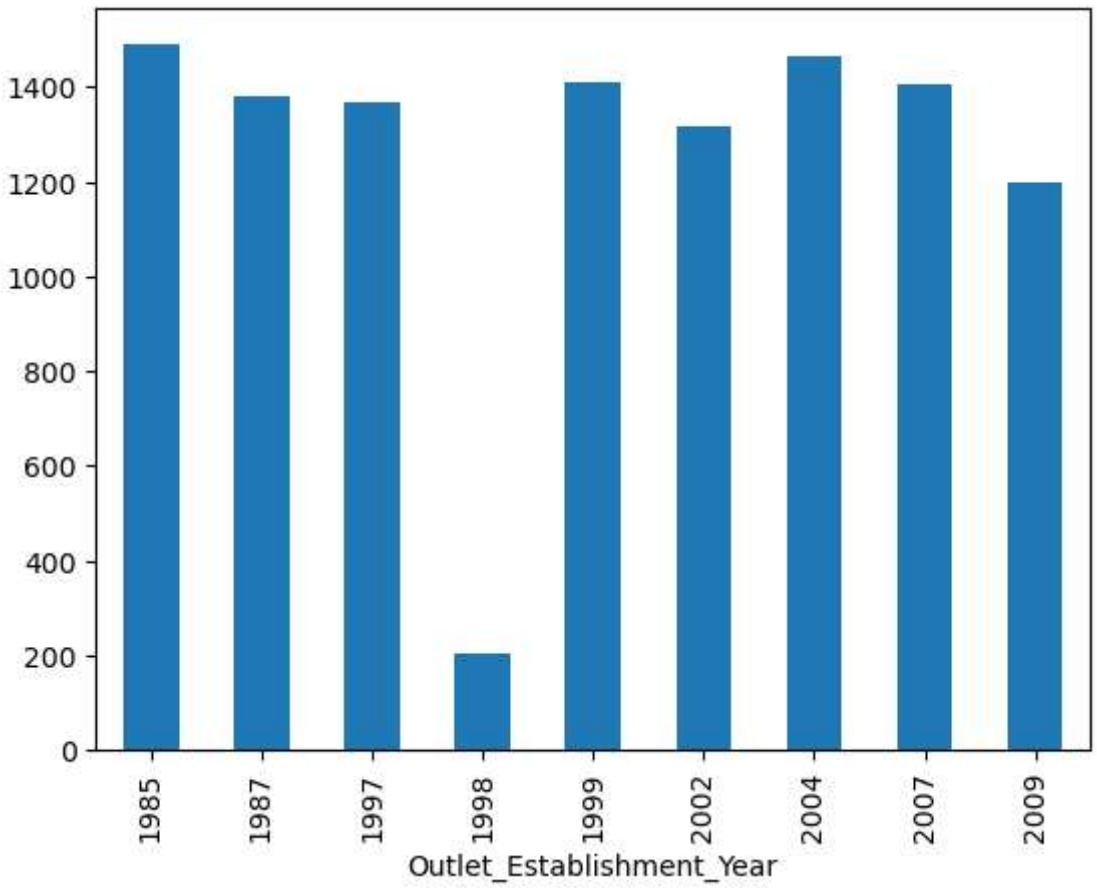
determine the years of opeation of sales in a store.

In [96]:
```
data['Outlet_Years'] = 2009 - data['Outlet_Establishment_Year']
data['Outlet_Years'].describe()
```

Out[96]:
```
count    14204.000000
mean        11.169319
std          8.371664
min          0.000000
25%          5.000000
50%         10.000000
75%         22.000000
max         24.000000
Name: Outlet_Years, dtype: float64
```

In [97]:
```
data.groupby('Outlet_Establishment_Year')['Item_Outlet_Sales'].mean().plot.bar()
```

Out[97]:
```
<Axes: xlabel='Outlet_Establishment_Year'>
```



In [98]:
```
df1.head()
```

Out[98]:

	Item_Identifier	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Es
0	FDW58	Low Fat	0.007565	Snack Foods	107.8622	OUT049	
1	FDW14	reg	0.038428	Dairy	87.3198	OUT017	
2	NCN55	Low Fat	0.099575	Others	241.7538	OUT010	
3	FDQ58	Low Fat	0.015388	Snack Foods	155.0340	OUT017	
4	FDY38	Regular	0.118599	Dairy	234.2300	OUT027	

In [99]:
```
temp_data = data.loc[data['Outlet_Establishment_Year'] == 1998]
```

In [100…
```
temp_data['Outlet_Type'].value_counts()
```

Out[100]:
```
Grocery Store    925
Name: Outlet_Type, dtype: int64
```

In [102…
```
test_temp_data = df1.loc[df1['Outlet_Establishment_Year'] == 1998]
test_temp_data['Outlet_Type'].value_counts()
```

Out[102]:
```
Grocery Store    370
Name: Outlet_Type, dtype: int64
```

Now the entire process has been done of bigmart sales data of different 10 stores in differenr city of 1559 product. This is just a trial asumption from my end with some help from other websites. In this summaery i have tried to look out all the assumtion of given data. The deeply aanlysis has been shown here and few steps are not covered.

In [ ]: