In [1]:
```
#! pip install pandas numpy seaborn matplotlib klib dtale scikit-learn joblib pandas-profiling
```

In [2]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df_train = pd.read_csv('Train.csv')
df_test = pd.read_csv('Test.csv')
```

In [4]:
```
df_train
```

Out[4]:

	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
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	FDX07	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052
...	...	...	...	...	...	...	...	...	...	...	...	...
8518	FDF22	6.865	Low Fat	0.056783	Snack Foods	214.5218	OUT013	1987	High	Tier 3	Supermarket Type1	2778.3834
8519	FDS36	8.380	Regular	0.046982	Baking Goods	108.1570	OUT045	2002	NaN	Tier 2	Supermarket Type1	549.2850
8520	NCJ29	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	OUT035	2004	Small	Tier 2	Supermarket Type1	1193.1136
8521	FDN46	7.210	Regular	0.145221	Snack Foods	103.1332	OUT018	2009	Medium	Tier 3	Supermarket Type2	1845.5976
8522	DRG01	14.800	Low Fat	0.044878	Soft Drinks	75.4670	OUT046	1997	Small	Tier 1	Supermarket Type1	765.6700

8523 rows × 12 columns

In [5]:
```
df_train.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 [6]:
```
df_train.isnull().sum()
```

Out[6]:
```
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 [7]:
```
df_train.shape
```

Out[7]:
```
(8523, 12)
```

In [8]:
```
df_test.isnull().sum()
```

Out[8]:
```
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 [9]:
```
df_train.describe()
```

Out[9]:

	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

In [10]:
```
df_train['Item_Weight'].describe()
```

Out[10]:
```
count    7060.000000
mean       12.857645
std         4.643456
min         4.555000
25%         8.773750
50%        12.600000
75%        16.850000
max        21.350000
Name: Item_Weight, dtype: float64
```

In [11]:
```
df_train['Item_Weight'].fillna(df_train['Item_Weight'].mean(), inplace=True)
df_test['Item_Weight'].fillna(df_test['Item_Weight'].mean(), inplace=True)
```

In [13]:
```
df_train.isnull().sum()
```

Out[13]:
```
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                  2410
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales               0
dtype: int64
```

In [14]:
```
df_test.isnull().sum()
```

Out[14]:
```
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                  1606
Outlet_Location_Type            0
Outlet_Type                     0
dtype: int64
```

In [15]:
```
df_train.describe()
```

Out[15]:

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Item_Outlet_Sales
count	8523.000000	8523.000000	8523.000000	8523.000000	8523.000000
mean	12.857645	0.066132	140.992782	1997.831867	2181.288914
std	4.226124	0.051598	62.275067	8.371760	1706.499616
min	4.555000	0.000000	31.290000	1985.000000	33.290000
25%	9.310000	0.026989	93.826500	1987.000000	834.247400
50%	12.857645	0.053931	143.012800	1999.000000	1794.331000
75%	16.000000	0.094585	185.643700	2004.000000	3101.296400
max	21.350000	0.328391	266.888400	2009.000000	13086.964800

We will use mode for imputation because outlet size is a category

In [16]:
```
df_train['Outlet_Size'].value_counts()
```

Out[16]:
```
Outlet_Size
Medium    2793
Small     2388
High       932
Name: count, dtype: int64
```

In [17]:
```
df_train['Outlet_Size'].mode()
```

Out[17]:
```
0    Medium
Name: Outlet_Size, dtype: object
```

In [18]:
```
df_train['Outlet_Size'].fillna(df_train['Outlet_Size'].mode()[0], inplace=True)
df_test['Outlet_Size'].fillna(df_test['Outlet_Size'].mode()[0], inplace=True)
```

In [19]:
```
df_train.isnull().sum()
```

Out[19]:
```
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
```

In [20]:
```
df_test.isnull().sum()
```

Out[20]:
```
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
dtype: int64
```

Selecting Features Based On General Requirements

In [21]:
```
df_train.drop(['Item_Identifier','Outlet_Identifier'],axis=1,inplace=True)
df_test.drop(['Item_Identifier','Outlet_Identifier'],axis=1,inplace=True)
```

In [22]:
```
df_train
```

Out[22]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	9.300	Low Fat	0.016047	Dairy	249.8092	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	5.920	Regular	0.019278	Soft Drinks	48.2692	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	17.500	Low Fat	0.016760	Meat	141.6180	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	1998	Medium	Tier 3	Grocery Store	732.3800
4	8.930	Low Fat	0.000000	Household	53.8614	1987	High	Tier 3	Supermarket Type1	994.7052
...	...	...	...	...	...	...	...	...	...	...
8518	6.865	Low Fat	0.056783	Snack Foods	214.5218	1987	High	Tier 3	Supermarket Type1	2778.3834
8519	8.380	Regular	0.046982	Baking Goods	108.1570	2002	Medium	Tier 2	Supermarket Type1	549.2850
8520	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	2004	Small	Tier 2	Supermarket Type1	1193.1136
8521	7.210	Regular	0.145221	Snack Foods	103.1332	2009	Medium	Tier 3	Supermarket Type2	1845.5976
8522	14.800	Low Fat	0.044878	Soft Drinks	75.4670	1997	Small	Tier 1	Supermarket Type1	765.6700

8523 rows × 10 columns

EDA Using Dtale Library

In [ ]:
```
pip install dtale
```

In [25]:
```
import dtale
```

In [28]:
```
dtale.show(df_train)
```

ERROR

The requested URL could not be retrieved

The following error was encountered while trying to retrieve the URL: http://desktop-epium9q:40000/dtale/iframe/1

Unable to determine IP address from host name "desktop-epium9q"

The DNS server returned:

Name Error: The domain name does not exist.

This means that the cache was not able to resolve the hostname presented in the URL. Check if the address is correct.

Your cache administrator is webmaster.

Generated Wed, 13 Dec 2023 17:52:39 GMT by htmlpdf-proxy-us (squid/4.10)

Out[28]:

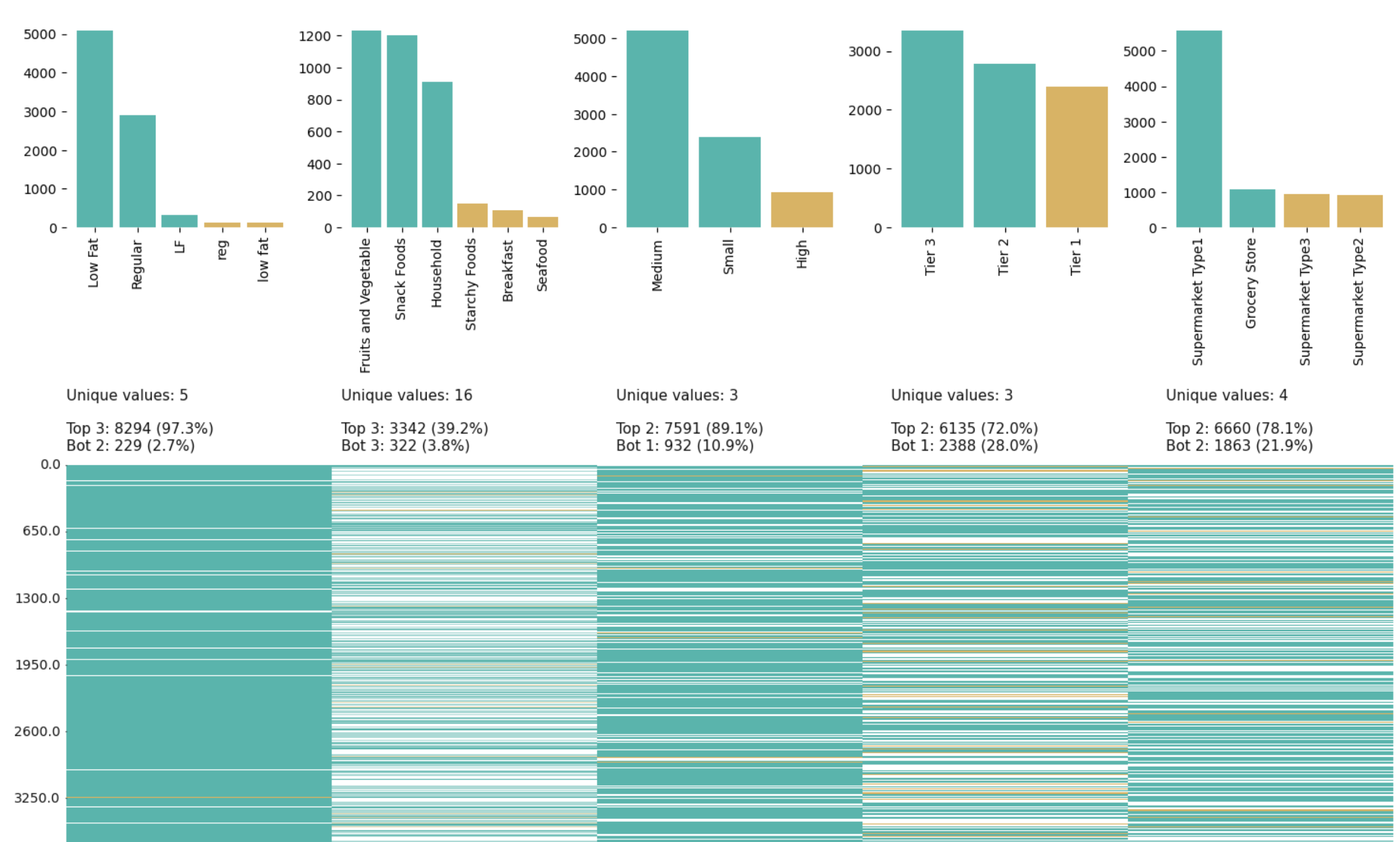
EDA Using klib library

In [ ]:
```
pip install klib
```

In [37]:
```
import klib
```

In [38]:
```
# klib.describe - functions for visualizing datasets
klib.cat_plot(df_train) # returns a visualization of the number and frequency of categorical features
```

Out[38]:
```
GridSpec(6, 5)
```



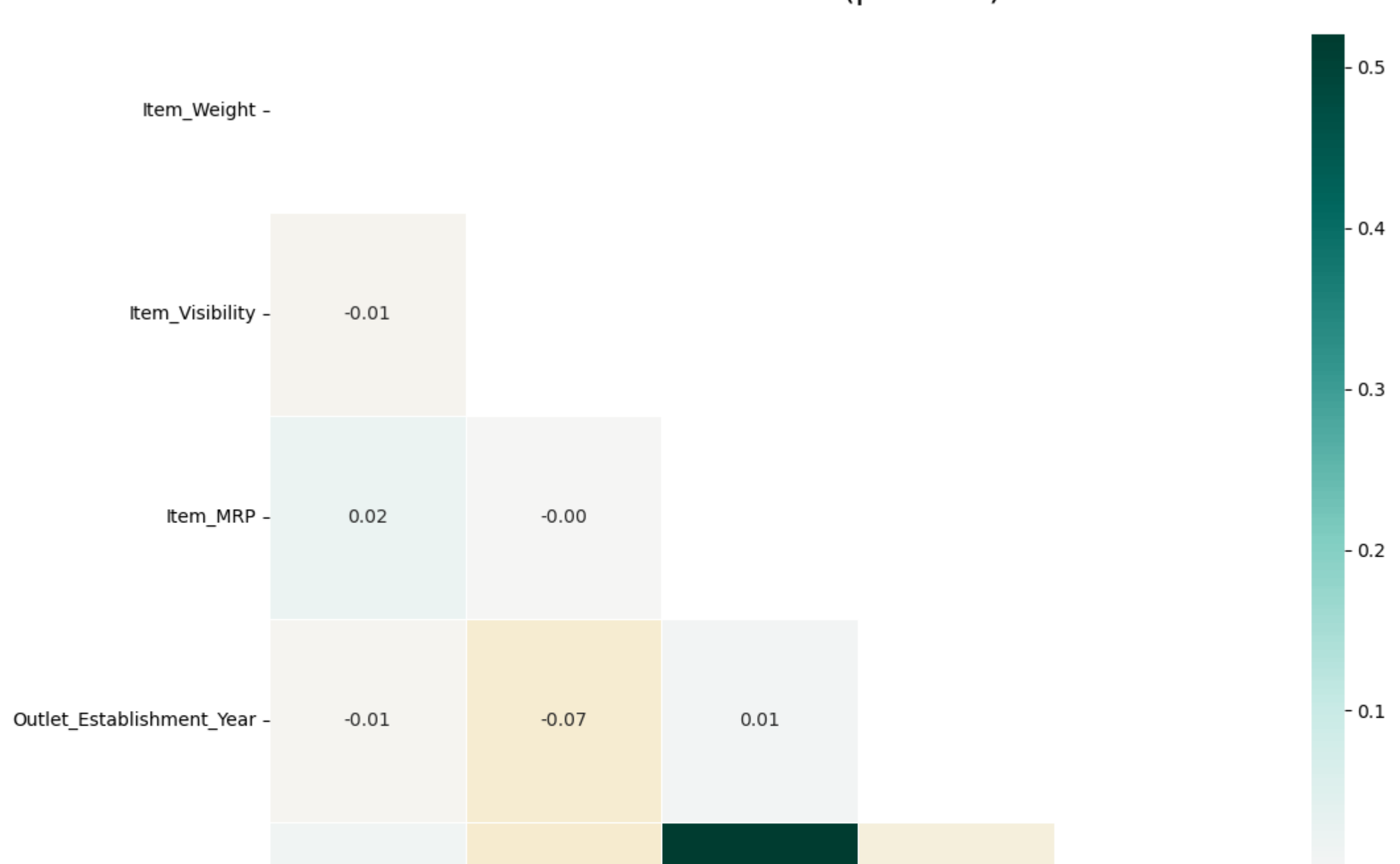
In [57]:
```
klib.corr_mat(df_train) # returns a color-encoded correlation matrix
```

Out[57]:

	item_weight	item_visibility	item_mrp	outlet_establishment_year	item_outlet_sales
item_weight	1.00	-0.01	0.02	-0.01	0.01
item_visibility	-0.01	1.00	-0.00	-0.07	-0.13
item_mrp	0.02	-0.00	1.00	0.01	0.57
outlet_establishment_year	-0.01	-0.07	0.01	1.00	-0.05
item_outlet_sales	0.01	-0.13	0.57	-0.05	1.00

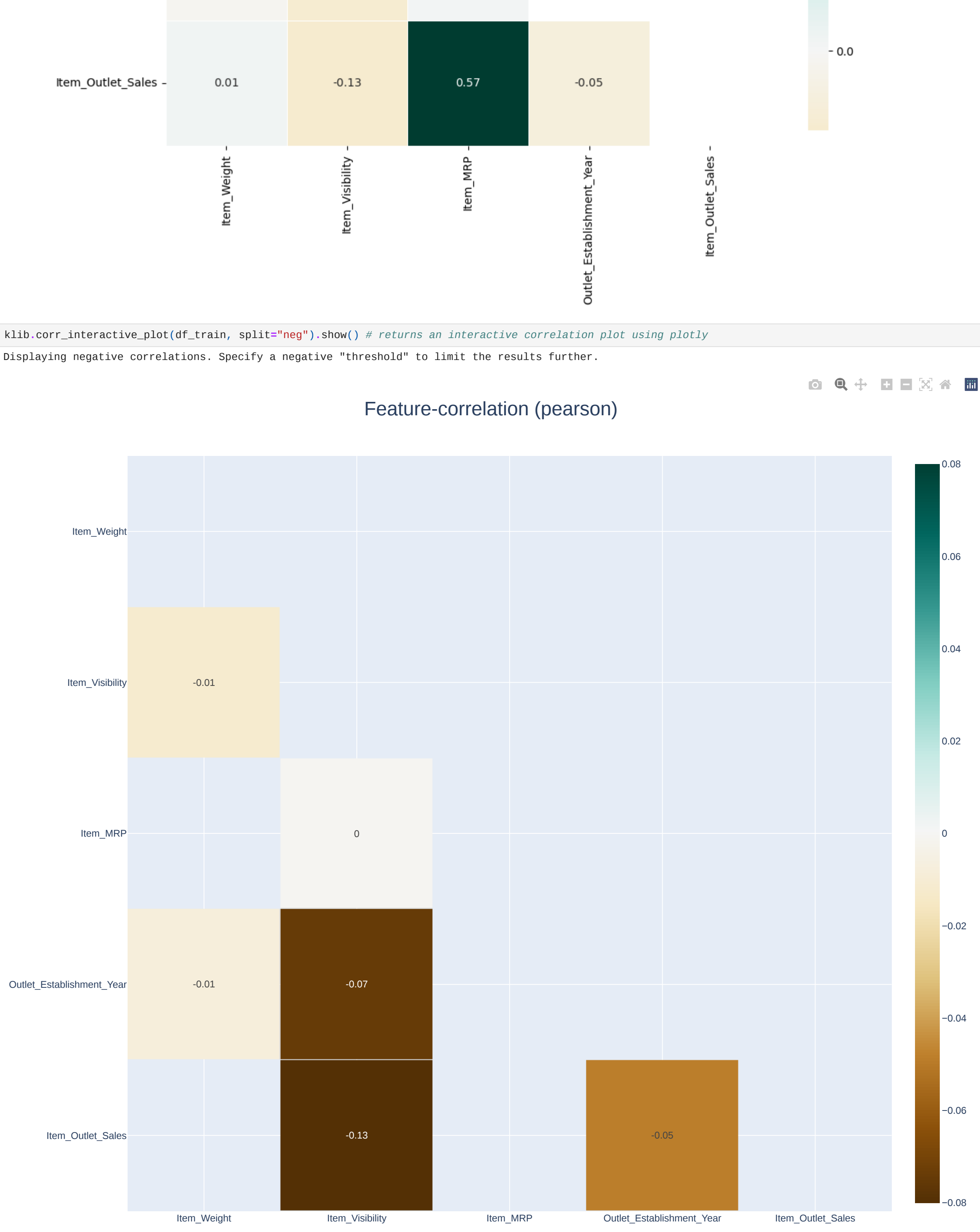
In [40]:
```
klib.corr_plot(df_train) # returns a color-encoded heatmap, ideal for correlations
```

Out[40]:
```
<Axes: title={'center': 'Feature-correlation (pearson)'}>
```



In [41]:
```
klib.corr_interactive_plot(df_train, split="neg").show() # returns an interactive correlation plot using plotly
```

```
Displaying negative correlations. Specify a negative "threshold" to limit the results further.
```



In [42]:
```
klib.dist_plot(df_train) # returns a distribution plot for every numeric feature
```

```
C:\Users\user\anaconda\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning:

The figure layout has changed to tight
```

Out[42]:
```
<Axes: xlabel='Item_Weight', ylabel='Density'>
```


In [43]:
```
klib.missingval_plot(df_train) # returns a figure containing information about missing values
```

```
No missing values found in the dataset.
```

Data Cleaning Using Klib

In [44]:
```
# klib.clean - functions for cleaning datasets
klib.data_cleaning(df_train) # performs datacleaning (drop duplicates & empty rows/cols, adjust dtypes,...)
```

```
Shape of cleaned data: (8523, 10) - Remaining NAs: 0


Dropped rows: 0
     of which 0 duplicates. (Rows (first 150 shown): [])

Dropped columns: 0
     of which 0 single valued.     Columns: []
Dropped missing values: 0
Reduced memory by at least: 0.46 MB (-70.77%)
```

Out[44]:

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type	item_outlet_sales
0	9.300000	Low Fat	0.016047	Dairy	249.809204	1999	Medium	Tier 1	Supermarket Type1	3735.137939
1	5.920000	Regular	0.019278	Soft Drinks	48.269199	2009	Medium	Tier 3	Supermarket Type2	443.422791
2	17.500000	Low Fat	0.016760	Meat	141.617996	1999	Medium	Tier 1	Supermarket Type1	2097.270020
3	19.200001	Regular	0.000000	Fruits and Vegetables	182.095001	1998	Medium	Tier 3	Grocery Store	732.380005
4	8.930000	Low Fat	0.000000	Household	53.861401	1987	High	Tier 3	Supermarket Type1	994.705200
...	...	...	...	...	...	...	...	...	...	...
8518	6.865000	Low Fat	0.056783	Snack Foods	214.521805	1987	High	Tier 3	Supermarket Type1	2778.383301
8519	8.380000	Regular	0.046982	Baking Goods	108.156998	2002	Medium	Tier 2	Supermarket Type1	549.284973
8520	10.600000	Low Fat	0.035186	Health and Hygiene	85.122398	2004	Small	Tier 2	Supermarket Type1	1193.113647
8521	7.210000	Regular	0.145221	Snack Foods	103.133202	2009	Medium	Tier 3	Supermarket Type2	1845.597656
8522	14.800000	Low Fat	0.044878	Soft Drinks	75.467003	1997	Small	Tier 1	Supermarket Type1	765.669983

8523 rows × 10 columns

In [45]:
```
klib.convert_datatypes(df_train) # converts existing to more efficient dtypes, also called inside data_cleaning()
```

Out[45]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	9.300000	Low Fat	0.016047	Dairy	249.809204	1999	Medium	Tier 1	Supermarket Type1	3735.137939
1	5.920000	Regular	0.019278	Soft Drinks	48.269199	2009	Medium	Tier 3	Supermarket Type2	443.422791
2	17.500000	Low Fat	0.016760	Meat	141.617996	1999	Medium	Tier 1	Supermarket Type1	2097.270020
3	19.200001	Regular	0.000000	Fruits and Vegetables	182.095001	1998	Medium	Tier 3	Grocery Store	732.380005
4	8.930000	Low Fat	0.000000	Household	53.861401	1987	High	Tier 3	Supermarket Type1	994.705200
...	...	...	...	...	...	...	...	...	...	...
8518	6.865000	Low Fat	0.056783	Snack Foods	214.521805	1987	High	Tier 3	Supermarket Type1	2778.383301
8519	8.380000	Regular	0.046982	Baking Goods	108.156998	2002	Medium	Tier 2	Supermarket Type1	549.284973
8520	10.600000	Low Fat	0.035186	Health and Hygiene	85.122398	2004	Small	Tier 2	Supermarket Type1	1193.113647
8521	7.210000	Regular	0.145221	Snack Foods	103.133202	2009	Medium	Tier 3	Supermarket Type2	1845.597656
8522	14.800000	Low Fat	0.044878	Soft Drinks	75.467003	1997	Small	Tier 1	Supermarket Type1	765.669983

8523 rows × 10 columns

In [46]:
```
klib.clean_column_names(df_train) # cleans and standardizes column names, also called inside data_cleaning()
```

Out[46]:

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type	item_outlet_sales
0	9.300	Low Fat	0.016047	Dairy	249.8092	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	5.920	Regular	0.019278	Soft Drinks	48.2692	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	17.500	Low Fat	0.016760	Meat	141.6180	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	1998	Medium	Tier 3	Grocery Store	732.3800
4	8.930	Low Fat	0.000000	Household	53.8614	1987	High	Tier 3	Supermarket Type1	994.7052
...	...	...	...	...	...	...	...	...	...	...
8518	6.865	Low Fat	0.056783	Snack Foods	214.5218	1987	High	Tier 3	Supermarket Type1	2778.3834
8519	8.380	Regular	0.046982	Baking Goods	108.1570	2002	Medium	Tier 2	Supermarket Type1	549.2850
8520	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	2004	Small	Tier 2	Supermarket Type1	1193.1136
8521	7.210	Regular	0.145221	Snack Foods	103.1332	2009	Medium	Tier 3	Supermarket Type2	1845.5976
8522	14.800	Low Fat	0.044878	Soft Drinks	75.4670	1997	Small	Tier 1	Supermarket Type1	765.6700

8523 rows × 10 columns

In [47]:
```
klib.drop_missing(df_train) # drops missing values, also called in data_cleaning()
```

Out[47]:

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type	item_outlet_sales
0	9.300	Low Fat	0.016047	Dairy	249.8092	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	5.920	Regular	0.019278	Soft Drinks	48.2692	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	17.500	Low Fat	0.016760	Meat	141.6180	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	1998	Medium	Tier 3	Grocery Store	732.3800
4	8.930	Low Fat	0.000000	Household	53.8614	1987	High	Tier 3	Supermarket Type1	994.7052
...	...	...	...	...	...	...	...	...	...	...
8518	6.865	Low Fat	0.056783	Snack Foods	214.5218	1987	High	Tier 3	Supermarket Type1	2778.3834
8519	8.380	Regular	0.046982	Baking Goods	108.1570	2002	Medium	Tier 2	Supermarket Type1	549.2850
8520	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	2004	Small	Tier 2	Supermarket Type1	1193.1136
8521	7.210	Regular	0.145221	Snack Foods	103.1332	2009	Medium	Tier 3	Supermarket Type2	1845.5976
8522	14.800	Low Fat	0.044878	Soft Drinks	75.4670	1997	Small	Tier 1	Supermarket Type1	765.6700

8523 rows × 10 columns

In [48]:
```
klib.mv_col_handling(df_train) # drops features with high ratio of missing vals based on informational content
```

Out[48]:

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type	item_outlet_sales
0	9.300	Low Fat	0.016047	Dairy	249.8092	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	5.920	Regular	0.019278	Soft Drinks	48.2692	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	17.500	Low Fat	0.016760	Meat	141.6180	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	1998	Medium	Tier 3	Grocery Store	732.3800
4	8.930	Low Fat	0.000000	Household	53.8614	1987	High	Tier 3	Supermarket Type1	994.7052
...	...	...	...	...	...	...	...	...	...	...
8518	6.865	Low Fat	0.056783	Snack Foods	214.5218	1987	High	Tier 3	Supermarket Type1	2778.3834
8519	8.380	Regular	0.046982	Baking Goods	108.1570	2002	Medium	Tier 2	Supermarket Type1	549.2850
8520	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	2004	Small	Tier 2	Supermarket Type1	1193.1136
8521	7.210	Regular	0.145221	Snack Foods	103.1332	2009	Medium	Tier 3	Supermarket Type2	1845.5976
8522	14.800	Low Fat	0.044878	Soft Drinks	75.4670	1997	Small	Tier 1	Supermarket Type1	765.6700

8523 rows × 10 columns

In [49]:
```
klib.pool_duplicate_subsets(df_train) # pools subset of cols based on duplicates with min. loss of information
```

Out[49]:

	item_visibility	item_mrp	item_outlet_sales	pooled_vars
0	0.016047	249.8092	3735.1380	0
1	0.019278	48.2692	443.4228	1
2	0.016760	141.6180	2097.2700	2
3	0.000000	182.0950	732.3800	3
4	0.000000	53.8614	994.7052	4
...	...	...	...	...
8518	0.056783	214.5218	2778.3834	8518
8519	0.046982	108.1570	549.2850	8519
8520	0.035186	85.1224	1193.1136	8520
8521	0.145221	103.1332	1845.5976	8521
8522	0.044878	75.4670	765.6700	8522

8523 rows × 4 columns

In [50]:
```
df_train
```

Out[50]:

	item_weight	item_fat_content	item_visibility	item_type	item_mrp	outlet_establishment_year	outlet_size	outlet_location_type	outlet_type	item_outlet_sales
0	9.300	Low Fat	0.016047	Dairy	249.8092	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	5.920	Regular	0.019278	Soft Drinks	48.2692	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	17.500	Low Fat	0.016760	Meat	141.6180	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	19.200	Regular	0.000000	Fruits and Vegetables	182.0950	1998	Medium	Tier 3	Grocery Store	732.3800
4	8.930	Low Fat	0.000000	Household	53.8614	1987	High	Tier 3	Supermarket Type1	994.7052
...	...	...	...	...	...	...	...	...	...	...
8518	6.865	Low Fat	0.056783	Snack Foods	214.5218	1987	High	Tier 3	Supermarket Type1	2778.3834
8519	8.380	Regular	0.046982	Baking Goods	108.1570	2002	Medium	Tier 2	Supermarket Type1	549.2850
8520	10.600	Low Fat	0.035186	Health and Hygiene	85.1224	2004	Small	Tier 2	Supermarket Type1	1193.1136
8521	7.210	Regular	0.145221	Snack Foods	103.1332	2009	Medium	Tier 3	Supermarket Type2	1845.5976
8522	14.800	Low Fat	0.044878	Soft Drinks	75.4670	1997	Small	Tier 1	Supermarket Type1	765.6700

8523 rows × 10 columns

Findings And Conclusion

In [58]:
```
# I used two exploratory libraries(tdale & klib libraries) on my EDA to confirm the findings, Which they did confirm the findings.
# Sales by Item Fat Content
# Low fat made the most sales followed by Regular.
# Sales by Item Type
# Fruits and Vegetable made the most sales followed by Snack foods and Household.
# Sales by Outlet Size
# Medium Outlet made the most sales followed by small outlets.
# Sales by Outlet Location Type
# Tier 3 made the most sale. However, Tier 2 & Tier 1 are also performing well.
# Sale by Type
# Supermarket Type 1 is making most sales.
```