Assignment

About The Dataset(Sales Dataset)

Here are the definitions for a few of the columns:

- **File_Type:** The value “Active” means that the particular product needs investigation
- **SoldFlag:** The value 1 = sale, 0 = no sale in past six months
- **SKU_number:** This is the unique identifier for each product.
- **Order:** Just a sequential counter. Can be ignored.
- **SoldFlag:** 1 = sold in past 6 mos. 0 = Not sold
- **MarketingType:** Two categories of how we market the product.
- **New_Release_Flag:** Any product that has had a future release (i.e., Release Number > 1)

Import the Required Libraries

In [1]:

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

Question - 1.Read the Sales dataset and display the top five rows

In [2]:

```
df=pd.read_csv('Sales_data.csv')
df.head()
```

Out[2]:

	Order	File_Type	SKU_number	SoldFlag	SoldCount	MarketingType	ReleaseNumber	New_F
0	2	Historical	1737127	0.0	0.0	D	15	
1	3	Historical	3255963	0.0	0.0	D	7	
2	4	Historical	612701	0.0	0.0	D	0	
3	6	Historical	115883	1.0	1.0	D	4	
4	7	Historical	863939	1.0	1.0	D	2	

Question - 2. Display the central tendency, Dispersion, and Shape of Dataset

In [3]:

```
# 'df.describe()' gives us a quick statistical summary of the dataset

df.describe()
```

Out[3]:

	Order	SKU_number	SoldFlag	SoldCount	ReleaseNumber	New_Release_
count	198917.000000	1.989170e+05	75996.000000	75996.000000	198917.000000	198917.00
mean	106483.543242	8.613626e+05	0.171009	0.322306	3.412202	0.64
std	60136.716784	8.699794e+05	0.376519	1.168615	3.864243	0.47
min	2.000000	5.000100e+04	0.000000	0.000000	0.000000	0.00
25%	55665.000000	2.172520e+05	0.000000	0.000000	1.000000	0.00
50%	108569.000000	6.122080e+05	0.000000	0.000000	2.000000	1.00
75%	158298.000000	9.047510e+05	0.000000	0.000000	5.000000	1.00
max	208027.000000	3.960788e+06	1.000000	73.000000	99.000000	1.00

In [4]:

```
# Measures of central tendency are -
# Mean
# Median
# Mode
```

In [5]:

```
# Mean of the Dataset
numcol=df[['PriceReg','ReleaseYear','ItemCount','LowUserPrice','LowNetPrice']]
numcol.mean()
```

Out[5]:

```
PriceReg          90.895243
ReleaseYear     2006.016414
ItemCount         41.426283
LowUserPrice      30.982487
LowNetPrice       46.832053
dtype: float64
```

In [6]:

```
# Median of the Dataset
numcol.median()
```

Out[6]:

```
PriceReg          69.95
ReleaseYear     2007.00
ItemCount         32.00
LowUserPrice      16.08
LowNetPrice       33.98
dtype: float64
```

In [7]:

```
# Mode of the Dataset
numcol.mode()
```

Out[7]:

	PriceReg	ReleaseYear	ItemCount	LowUserPrice	LowNetPrice
0	49.95	2010	21	4.0	0.0

In [8]:

```
# Dispersion measures
# Standard deviation
# Variance
```

In [9]:

```
# Standard deviation
numcol.std()
```

Out[9]:

```
PriceReg          86.736367
ReleaseYear        9.158331
ItemCount         37.541215
LowUserPrice      69.066155
LowNetPrice      128.513236
dtype: float64
```

In [10]:

```
# Variance
numcol.var()
```

Out[10]:

```
PriceReg         7523.197307
ReleaseYear        83.875030
ItemCount        1409.342820
LowUserPrice     4770.133814
LowNetPrice     16515.651882
dtype: float64
```

In [11]:

```
# Shape of the Dataset

df.shape
```

Out[11]:

```
(198917, 14)
```

1. Missing Value Analysis

Question - 3. Our dataset any missing values? If Yes then how many and which feature has a missing value

In [12]:

```
df.isnull().sum()
```

Out[12]:

```
Order                  0
File_Type              0
SKU_number             0
SoldFlag          122921
SoldCount         122921
MarketingType          0
ReleaseNumber          0
New_Release_Flag       0
StrengthFactor         0
PriceReg               0
ReleaseYear            0
ItemCount              0
LowUserPrice           0
LowNetPrice            0
dtype: int64
```

In [13]:

```
# we can see that "SoldFlag" and "SoldCount" columns both have 122921 missing or null value
```

Question - 4. Find a list of all the columns which have more than 50% of their values missing

In [14]:

```
df.isnull().sum()/len(df)*100
```

Out[14]:

```
Order                 0.000000
File_Type             0.000000
SKU_number            0.000000
SoldFlag             61.795121
SoldCount            61.795121
MarketingType         0.000000
ReleaseNumber         0.000000
New_Release_Flag      0.000000
StrengthFactor        0.000000
PriceReg              0.000000
ReleaseYear           0.000000
ItemCount             0.000000
LowUserPrice          0.000000
LowNetPrice           0.000000
dtype: float64
```

In [15]:

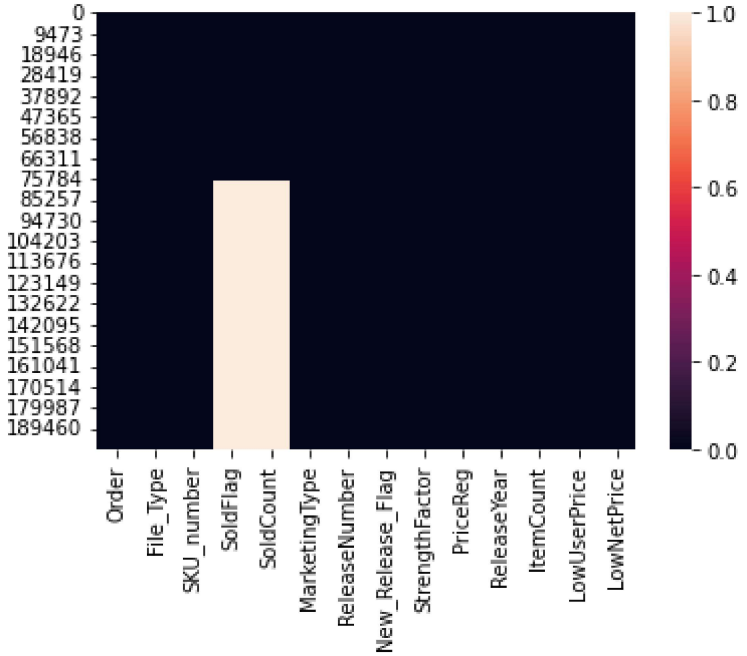
```
# we can see that "SoldFlag" and "SoldCount" columns both have 61.79% missing or null value
# which implies that these two columns have more than 50% values missing
```

Question - 5. View the missing values within the data.

Hint : Heatmap

In [16]:

```
sns.heatmap(df.isnull());
```



2. Dealing with missing values.

Question - 6. Remove all records if no more than 2 observations have been recorded.

In [17]:

```
df.dropna(thresh=2)
df.shape
```

Out[17]:

```
(198917, 14)
```

Question - 7. Discard unnecessary columns of data.

In [18]:

```
df.drop(['SoldCount'],axis=1)
```

Out[18]:

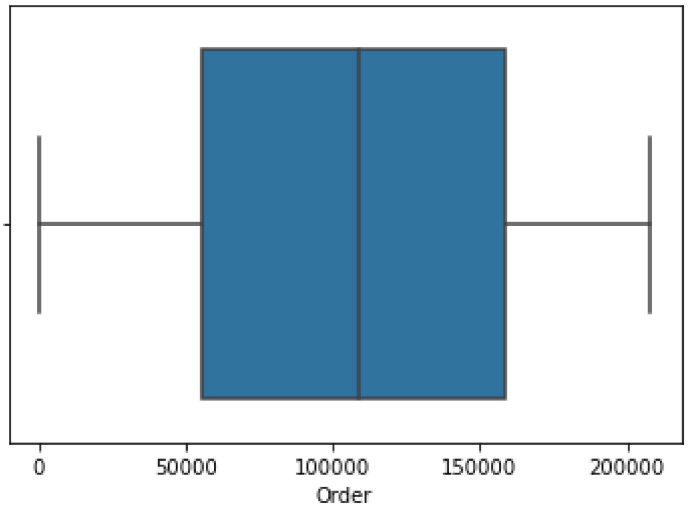
	Order	File_Type	SKU_number	SoldFlag	MarketingType	ReleaseNumber	New_Releas
0	2	Historical	1737127	0.0	D	15	
1	3	Historical	3255963	0.0	D	7	
2	4	Historical	612701	0.0	D	0	
3	6	Historical	115883	1.0	D	4	
4	7	Historical	863939	1.0	D	2	
...	...	...	...	...	...	...	
198912	208023	Active	109683	NaN	D	7	
198913	208024	Active	416462	NaN	D	8	
198914	208025	Active	658242	NaN	S	2	
198915	208026	Active	2538340	NaN	S	2	
198916	208027	Active	416662	NaN	D	15	

198917 rows × 13 columns

Question - 8.Is there any feature in the dataset that could be exempted from dealing with outliers?

In [19]:

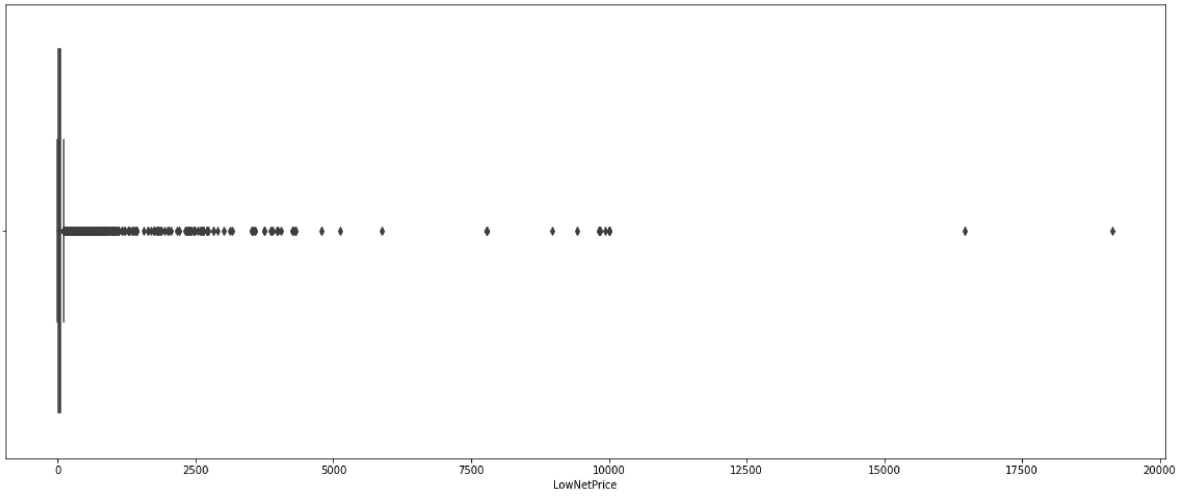
```
sns.boxplot(data=df,x='Order');
```



Question - 9.How do I check the outliers for the feature showing the low net price?

In [20]:

```
plt.figure(figsize=(20,8))
sns.boxplot(data=df, x='LowNetPrice');
```



Question - 10.Find out optimal value of K and minimize the outliers.?

In [21]:

```
df['lnp_value']=np.log(df['LowNetPrice'])
Q1=df['lnp_value'].quantile(0.25)
Q3=df['lnp_value'].quantile(0.75)
IQR=Q3-Q1
LL=Q1-3*IQR
UL=Q3+3*IQR
df=df[(df['lnp_value']<UL)&(df['lnp_value']>LL)]
df.head()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\arraylike.py:397: Run
timeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

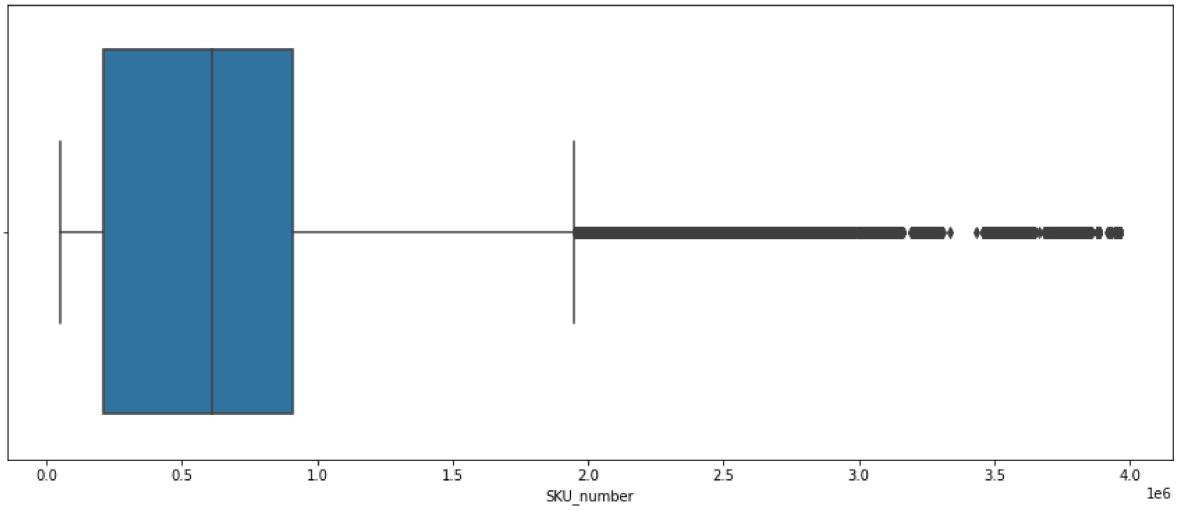
Out[21]:

	Order	File_Type	SKU_number	SoldFlag	SoldCount	MarketingType	ReleaseNumber	New_F
0	2	Historical	1737127	0.0	0.0	D	15	
1	3	Historical	3255963	0.0	0.0	D	7	
2	4	Historical	612701	0.0	0.0	D	0	
3	6	Historical	115883	1.0	1.0	D	4	
4	7	Historical	863939	1.0	1.0	D	2	

Question - 11.Suggest a method for finding outliers for the feature representing the number of sku number.?

In [22]:

```
plt.figure(figsize=(15,6))
sns.boxplot(data=df, x='SKU_number');
```



Question - 12.Which method allows you to retrieve the corresponding record within the data.?

In [23]:

```
# We can use loc[] or iloc[] method to retrieve the corresponding record within the data.
```

Question - 13.Calculate the standard score of Release Number?

In [24]:

```
from scipy.stats import zscore
df1=df
df1['zscore']= zscore(df1['ReleaseNumber'])
df1.sample(20)
```

Out[24]:

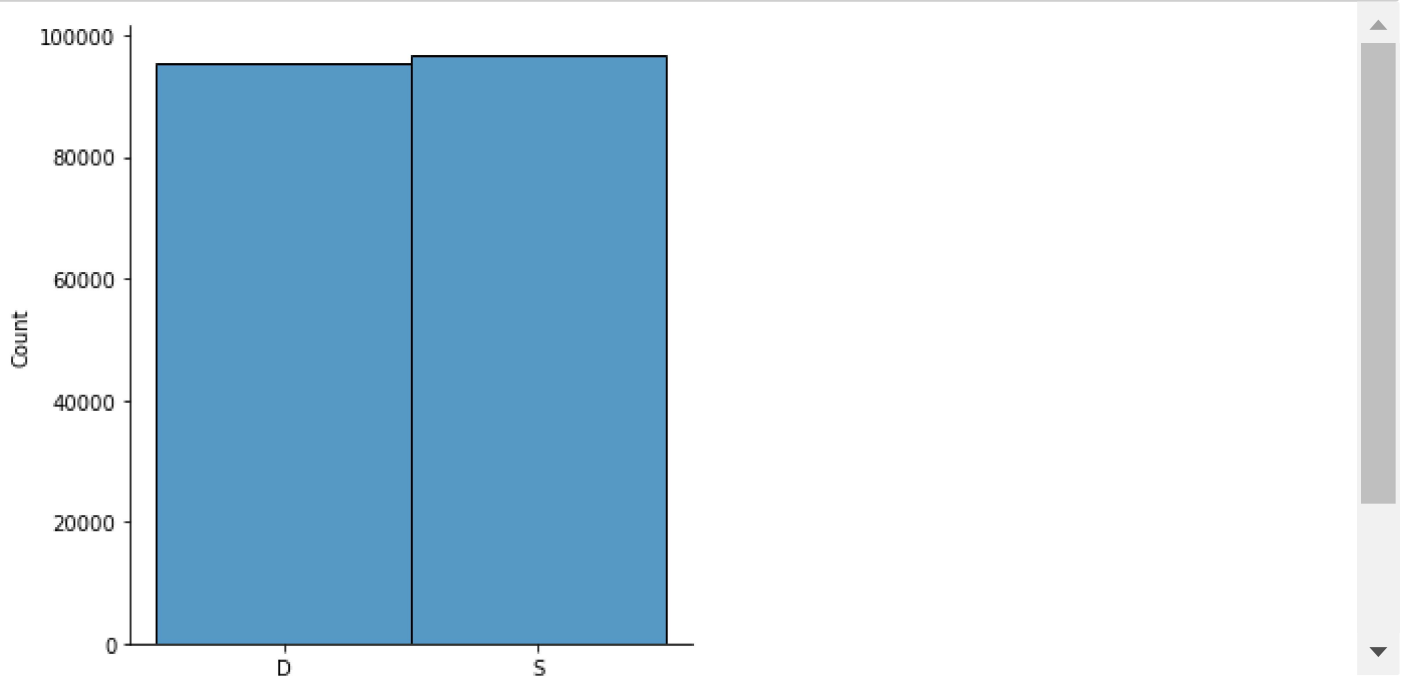
	Order	File_Type	SKU_number	SoldFlag	SoldCount	MarketingType	ReleaseNumber
118864	127975	Active	1483793	NaN	NaN	S	0
51191	57159	Historical	738458	0.0	0.0	S	2
32035	32745	Historical	211742	1.0	1.0	D	1
126748	135859	Active	630705	NaN	NaN	S	0
185731	194842	Active	661767	NaN	NaN	S	2
874	882	Historical	55123	0.0	0.0	D	5
88133	97244	Active	203347	NaN	NaN	D	1
197816	206927	Active	106656	NaN	NaN	S	5
20291	20762	Historical	165830	0.0	0.0	D	2
74716	83667	Historical	861806	0.0	0.0	S	2
155142	164253	Active	533364	NaN	NaN	D	3
196446	205557	Active	176723	NaN	NaN	S	1
146226	155337	Active	3065288	NaN	NaN	D	0
11477	11802	Historical	405431	1.0	1.0	D	4
73187	81885	Historical	641577	0.0	0.0	S	2
197586	206697	Active	714268	NaN	NaN	D	1
44726	50583	Historical	539863	0.0	0.0	S	3
6534	6769	Historical	57125	1.0	1.0	D	4
173180	182291	Active	287583	NaN	NaN	D	8
116271	125382	Active	3598385	NaN	NaN	S	0

Question - 14.In marketing type which items are more sellable.

NOTE:- Show the output in visualization

In [25]:

```
sns.displot(x='MarketingType', data=df1);
```



In [26]:

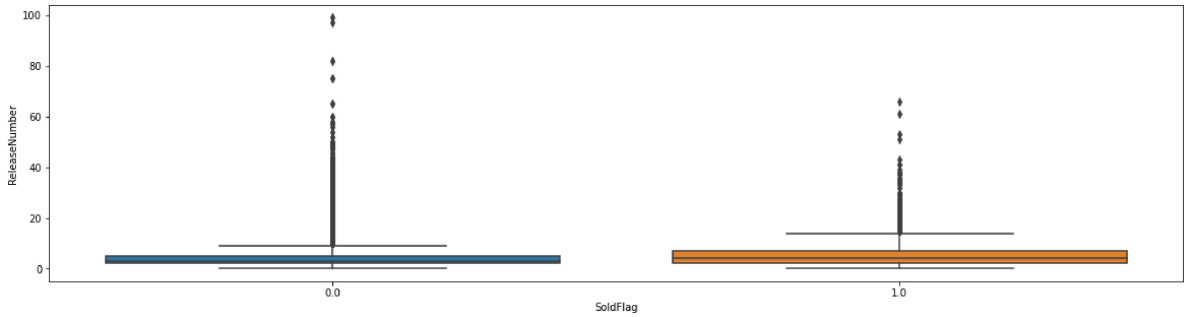
```
# Items with marketing type S are more sellable
```

Question - 15.Find out bias of soldflag based on release number?

NOTE:- Box Plot.

In [27]:

```
plt.figure(figsize=(20,5))
sns.boxplot(data=df1, x='SoldFlag',y='ReleaseNumber');
```



Question - 16.Find out missing values and fill with zero?

In [28]:

```
df1.fillna(0)
```

Out[28]:

	Order	File_Type	SKU_number	SoldFlag	SoldCount	MarketingType	ReleaseNumber
0	2	Historical	1737127	0.0	0.0	D	15
1	3	Historical	3255963	0.0	0.0	D	7
2	4	Historical	612701	0.0	0.0	D	0
3	6	Historical	115883	1.0	1.0	D	4
4	7	Historical	863939	1.0	1.0	D	2
...	...	...	...	...	...	...	...
198912	208023	Active	109683	0.0	0.0	D	7
198913	208024	Active	416462	0.0	0.0	D	8
198914	208025	Active	658242	0.0	0.0	S	2
198915	208026	Active	2538340	0.0	0.0	S	2
198916	208027	Active	416662	0.0	0.0	D	15

192075 rows × 16 columns

Question - 17.How many value are active and historical in File_Type column ?

NOTE:- Explain in brief from graph.

In [29]:

```
df1['File_Type'].value_counts()
```

Out[29]:

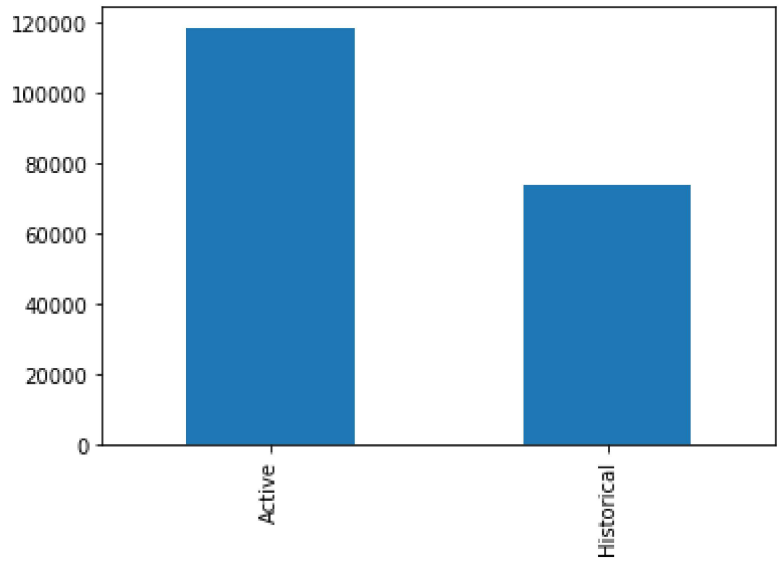
```
Active        118369
Historical     73706
Name: File_Type, dtype: int64
```

In [30]:

```
df1['File_Type'].value_counts().plot(kind='bar')
```

Out[30]:

<AxesSubplot:>

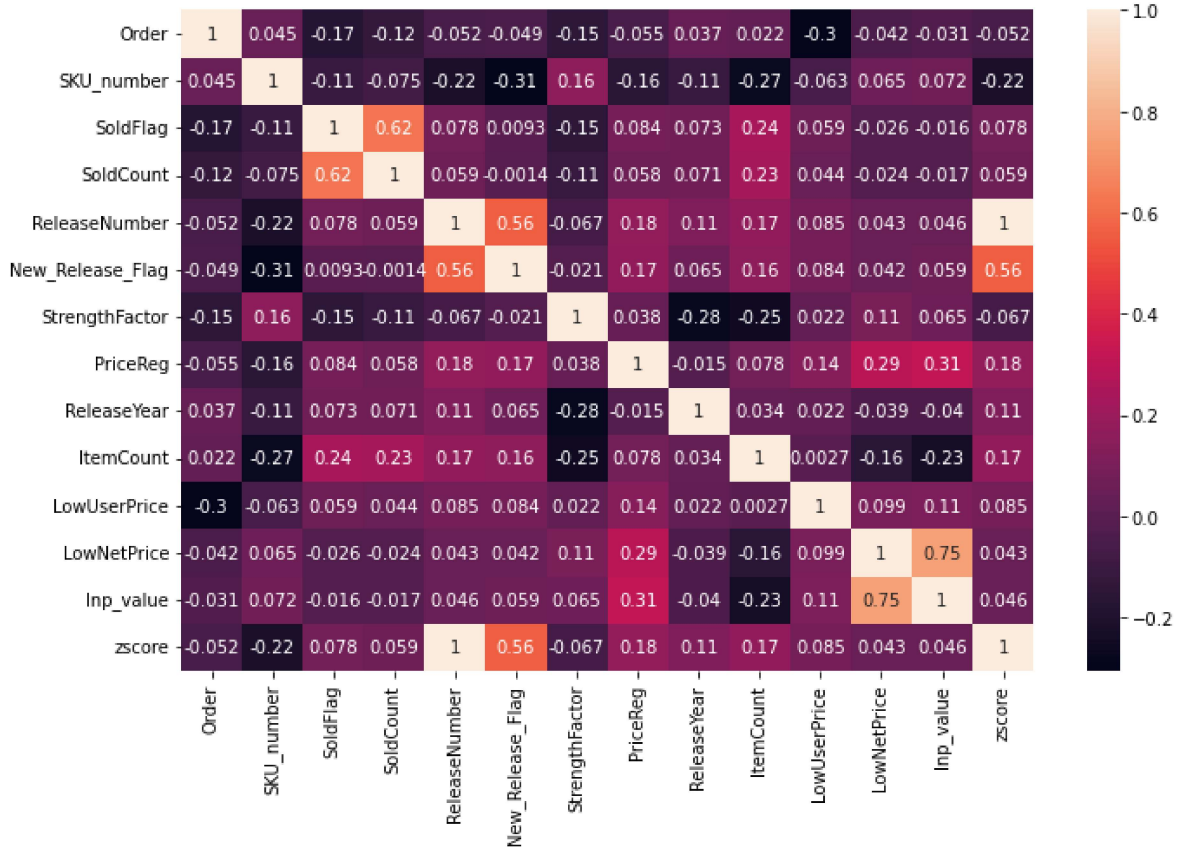


Question - 18. How many features are positive and negatively correlated?

NOTE:- Explain with values and plot the corrleation values.

In [31]:

```
plt.figure(figsize=(11,7))
sns.heatmap(df1.corr(),annot=True);
```



Question - 19. Show the corrleation between 0 and 1.

In [32]:

```
df1.corr()
```

Out[32]:

	Order	SKU_number	SoldFlag	SoldCount	ReleaseNumber	New_Releas
Order	1.000000	0.045165	-0.174267	-0.122051	-0.051544	-0.(
SKU_number	0.045165	1.000000	-0.109641	-0.075020	-0.216194	-0.:
SoldFlag	-0.174267	-0.109641	1.000000	0.618850	0.077915	0.(
SoldCount	-0.122051	-0.075020	0.618850	1.000000	0.059219	-0.(
ReleaseNumber	-0.051544	-0.216194	0.077915	0.059219	1.000000	0.!
New_Release_Flag	-0.049335	-0.305282	0.009287	-0.001353	0.559952	1.(
StrengthFactor	-0.150731	0.161495	-0.147906	-0.113258	-0.067220	-0.(
PriceReg	-0.055001	-0.160160	0.084345	0.057843	0.176652	0.'
ReleaseYear	0.037367	-0.110654	0.073253	0.071450	0.114793	0.(
ItemCount	0.022130	-0.265231	0.239829	0.231955	0.173502	0.'
LowUserPrice	-0.298476	-0.062532	0.058557	0.044041	0.085305	0.(
LowNetPrice	-0.042142	0.064750	-0.025548	-0.024083	0.042935	0.(
Inp_value	-0.030624	0.071828	-0.016091	-0.016556	0.046496	0.(
zscore	-0.051544	-0.216194	0.077915	0.059219	1.000000	0.!

Question - 20. Summaries the data with categorical variable.

In [33]:

```
# df1.describe(include='all')
df1.describe(include='object')
```

Out[33]:

	File_Type	MarketingType
count	192075	192075
unique	2	2
top	Active	S
freq	118369	96709

Question - 21. Calculate the mean and median of SKU_number.

In [34]:

```
# Mean
print('The mean of SKU_number is',df1['SKU_number'].mean())
```

```
The mean of SKU_number is 866877.1238474554
```

In [35]:

```
# Median
print('The median of SKU_number is',df1['SKU_number'].median())
```

```
The median of SKU_number is 612199.0
```

Question - 22. Calculate the mean and median of PriceCount.

In [36]:

```
# Mean
print('The mean of PriceReg is',df1['PriceReg'].mean())
```

```
The mean of PriceReg is 91.59467676677717
```

In [37]:

```
# Median
print('The median of PriceReg is',df1['PriceReg'].median())
```

```
The median of PriceReg is 69.95
```

Question - 23.Convert the File_Type columns to numerical.

In [38]:

```
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
```

In [39]:

```
df1['FileType_LE']=LE.fit_transform(df1['File_Type'])
df1.head()
```

Out[39]:

	Order	File_Type	SKU_number	SoldFlag	SoldCount	MarketingType	ReleaseNumber	New_F
0	2	Historical	1737127	0.0	0.0	D	15	
1	3	Historical	3255963	0.0	0.0	D	7	
2	4	Historical	612701	0.0	0.0	D	0	
3	6	Historical	115883	1.0	1.0	D	4	
4	7	Historical	863939	1.0	1.0	D	2	

In [40]:

```
df1['File_Type'].unique()
```

Out[40]:

```
array(['Historical', 'Active'], dtype=object)
```

In [41]:

```
df1['FileType_LE'].unique()
```

Out[41]:

```
array([1, 0])
```

Question - 24.Handle the remaining categorical data.

In [42]:

```
df1['MarketingType_LE']=LE.fit_transform(df1['MarketingType'])
df1.head()
```

Out[42]:

	Order	File_Type	SKU_number	SoldFlag	SoldCount	MarketingType	ReleaseNumber	New_F
0	2	Historical	1737127	0.0	0.0	D	15	
1	3	Historical	3255963	0.0	0.0	D	7	
2	4	Historical	612701	0.0	0.0	D	0	
3	6	Historical	115883	1.0	1.0	D	4	
4	7	Historical	863939	1.0	1.0	D	2	

Question - 25.Normalize the data.

In [43]:

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
df1['StrengthFactor']=ss.fit_transform(df1['StrengthFactor'].array.reshape(-1,1))
df1['SKU_number']=ss.fit_transform(df1['SKU_number'].array.reshape(-1,1))
```

In [44]:

```
df1.head()
```

Out[44]:

	Order	File_Type	SKU_number	SoldFlag	SoldCount	MarketingType	ReleaseNumber	New_F
0	2	Historical	0.994743	0.0	0.0	D	15	
1	3	Historical	2.730855	0.0	0.0	D	7	
2	4	Historical	-0.290537	0.0	0.0	D	0	
3	6	Historical	-0.858427	1.0	1.0	D	4	
4	7	Historical	-0.003358	1.0	1.0	D	2	

In [ ]:

```

```