In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
from matplotlib import pyplot as plt
import itertools

%matplotlib inline
matplotlib.style.use('ggplot')
# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list the files in the input di
rectory

# import os
# print(os.listdir("../input"))

# Any results you write to the current directory are saved as output.
```

In [3]:
```
# Reading the dataset
sales_data = pd.read_csv("SalesKaggle3.csv")
```

In [4]:
```
sales_data.head()
```

Out[4]:

	Order	File_Type	SKU_number	SoldFlag	SoldCount	MarketingType	ReleaseNumber	New_Release_Flag	StrengthFactor	PriceReg	ReleaseY
0	2	Historical	1737127	0.0	0.0	D	15	1	682743.0	44.99	20
1	3	Historical	3255963	0.0	0.0	D	7	1	1016014.0	24.81	20
2	4	Historical	612701	0.0	0.0	D	0	0	340464.0	46.00	20
3	6	Historical	115883	1.0	1.0	D	4	1	334011.0	100.00	20
4	7	Historical	863939	1.0	1.0	D	2	1	1287938.0	121.95	20

In [5]:
```
#Statistical description of the dataset
sales_data.describe()
```

Out[5]:

	Order	SKU_number	SoldFlag	SoldCount	ReleaseNumber	New_Release_Flag	StrengthFactor	PriceReg	ReleaseY
count	198917.000000	1.989170e+05	75996.000000	75996.000000	198917.000000	198917.000000	1.989170e+05	198917.000000	198917.000(
mean	106483.543242	8.613626e+05	0.171009	0.322306	3.412202	0.642248	1.117115e+06	90.895243	2006.016
std	60136.716784	8.699794e+05	0.376519	1.168615	3.864243	0.479340	1.522090e+06	86.736367	9.158
min	2.000000	5.000100e+04	0.000000	0.000000	0.000000	0.000000	6.275000e+00	0.000000	0.000(
25%	55665.000000	2.172520e+05	0.000000	0.000000	1.000000	0.000000	1.614188e+05	42.000000	2003.000(
50%	108569.000000	6.122080e+05	0.000000	0.000000	2.000000	1.000000	5.822240e+05	69.950000	2007.000(
75%	158298.000000	9.047510e+05	0.000000	0.000000	5.000000	1.000000	1.430083e+06	116.000000	2011.000(
max	208027.000000	3.960788e+06	1.000000	73.000000	99.000000	1.000000	1.738445e+07	12671.480000	2018.000(

In [6]:
```
# Includes categorical variable
sales_data.describe(include='all')
```

Out[6]:

	Order	File_Type	SKU_number	SoldFlag	SoldCount	MarketingType	ReleaseNumber	New_Release_Flag	StrengthFactor
count	198917.000000	198917	1.989170e+05	75996.000000	75996.000000	198917	198917.000000	198917.000000	1.989170e+05
unique	NaN	2	NaN	NaN	NaN	2	NaN	NaN	NaN
top	NaN	Active	NaN	NaN	NaN	S	NaN	NaN	NaN
freq	NaN	122921	NaN	NaN	NaN	100946	NaN	NaN	NaN
mean	106483.543242	NaN	8.613626e+05	0.171009	0.322306	NaN	3.412202	0.642248	1.117115e+06
std	60136.716784	NaN	8.699794e+05	0.376519	1.168615	NaN	3.864243	0.479340	1.522090e+06
min	2.000000	NaN	5.000100e+04	0.000000	0.000000	NaN	0.000000	0.000000	6.275000e+00
25%	55665.000000	NaN	2.172520e+05	0.000000	0.000000	NaN	1.000000	0.000000	1.614188e+05
50%	108569.000000	NaN	6.122080e+05	0.000000	0.000000	NaN	2.000000	1.000000	5.822240e+05
75%	158298.000000	NaN	9.047510e+05	0.000000	0.000000	NaN	5.000000	1.000000	1.430083e+06
max	208027.000000	NaN	3.960788e+06	1.000000	73.000000	NaN	99.000000	1.000000	1.738445e+07

In [7]:
```
sales_data.shape
```

Out[7]: (198917, 14)

In [8]:
```
sales_data.nunique()
```

Out[8]:
```
Order              198917
File_Type               2
SKU_number         133360
SoldFlag                2
SoldCount              37
MarketingType           2
ReleaseNumber          71
New_Release_Flag        2
StrengthFactor     197424
PriceReg            11627
ReleaseYear            85
ItemCount             501
LowUserPrice        12102
LowNetPrice         15403
dtype: int64
```

In [9]:
```
sales_data.isnull().values.any()
```

Out[9]:
```
True
```

In [10]:
```
sales_data.isnull().sum()
```

Out[10]:
```
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 [11]:
```
sales_data['SoldFlag'].fillna(0, inplace=True)
sales_data['SoldCount'].fillna(0, inplace=True)
```

In [12]:
```
sales_data.isnull().sum()
```

Out[12]:
```
Order              0
File_Type          0
SKU_number         0
SoldFlag           0
SoldCount          0
MarketingType      0
ReleaseNumber      0
New_Release_Flag   0
StrengthFactor     0
PriceReg           0
ReleaseYear        0
ItemCount          0
LowUserPrice       0
LowNetPrice        0
dtype: int64
```

In [13]:
```
sales_data[sales_data['File_Type'] == 'Historical']['SKU_number'].count()
```

Out[13]:
```
75996
```

In [14]:
```
sales_data[sales_data['File_Type'] == 'Active']['SKU_number'].count()
```

Out[14]:
```
122921
```

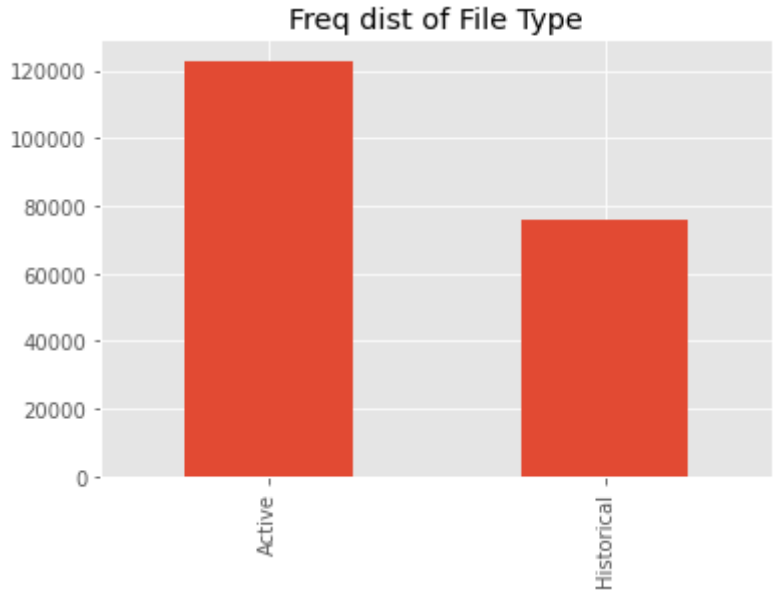
In [15]:
```
sales_data_hist = sales_data[sales_data['File_Type'] == 'Historical']
sales_data_act = sales_data[sales_data['File_Type'] == 'Active']
```

In [16]:
```
sales_data['MarketingType'].value_counts().plot.bar(title="Freq dist of Marketing Type")
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9b991995d0>
```

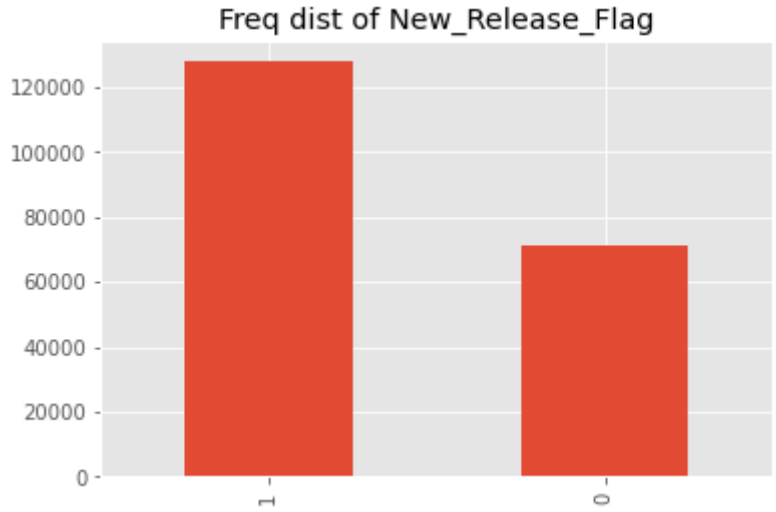
In [17]:
```
sales_data['File_Type'].value_counts().plot.bar(title="Freq dist of File Type")
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9b9ed9b450>
```



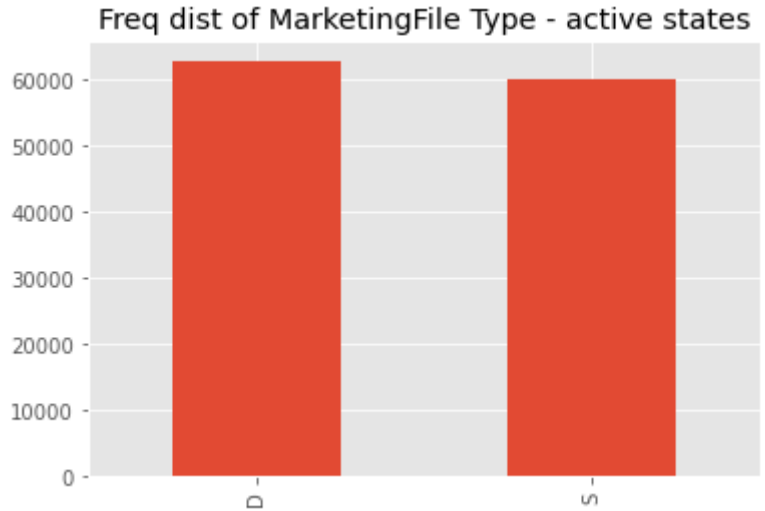
In [18]:
```
sales_data['New_Release_Flag'].value_counts().plot.bar(title="Freq dist of New_Release_Flag")
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9b9eb91250>
```



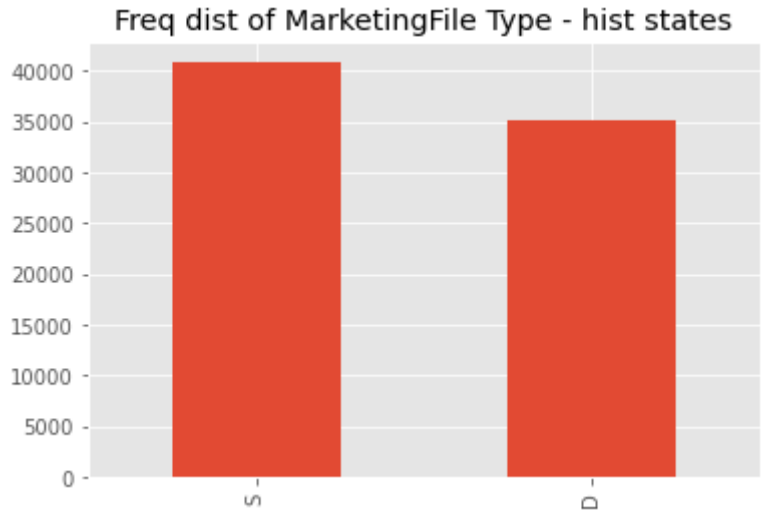
In [19]:
```
sales_data_act['MarketingType'].value_counts().plot.bar(title="Freq dist of MarketingFile Type - active s
tates")
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9b9eb733d0>
```

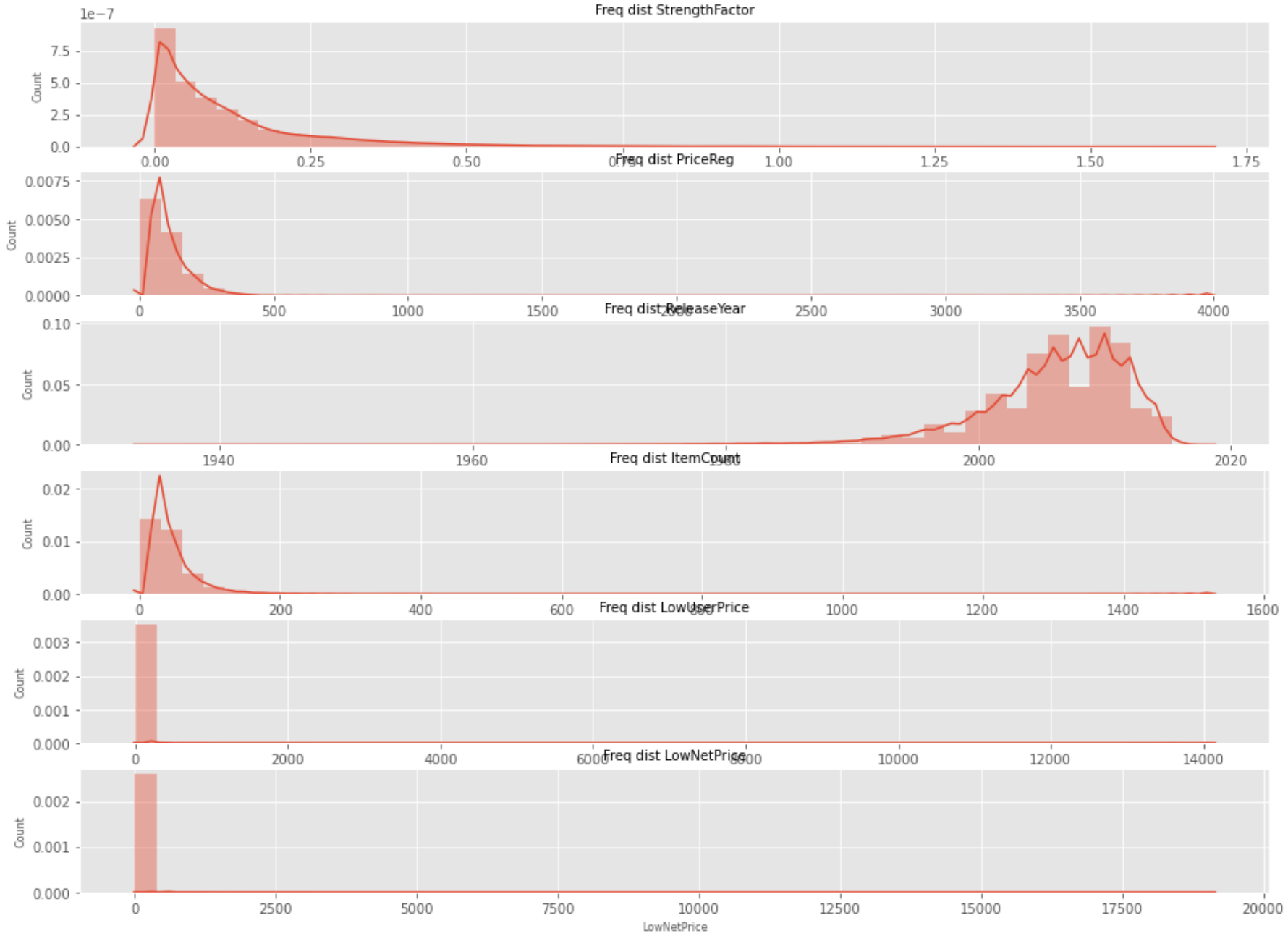


In [20]:
```
sales_data_hist['MarketingType'].value_counts().plot.bar(title="Freq dist of MarketingFile Type - hist st
ates")
```

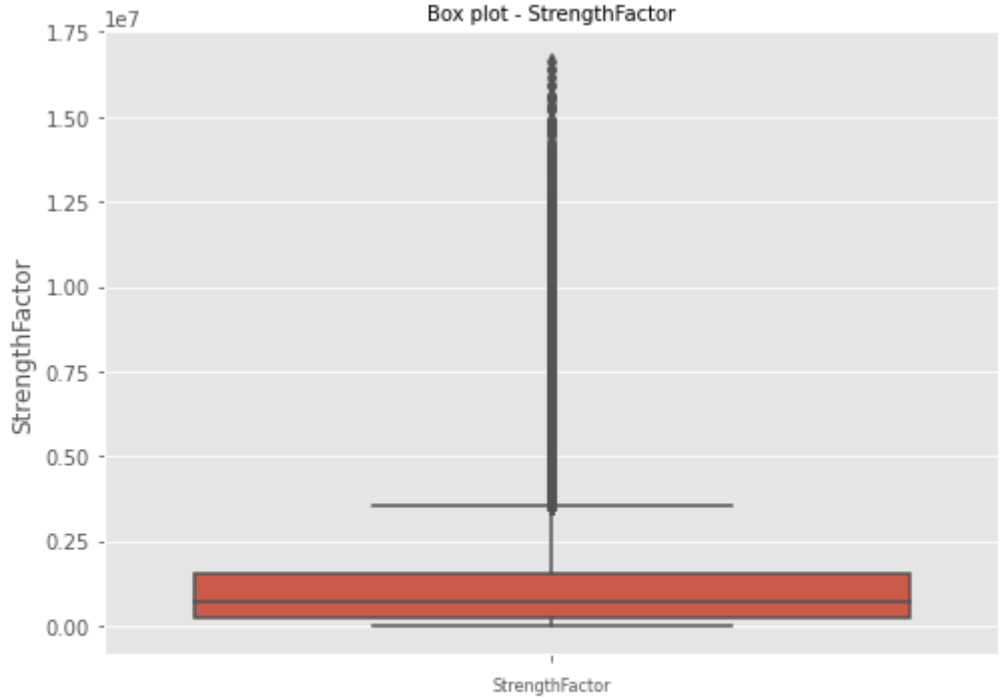
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9b9eb21190>

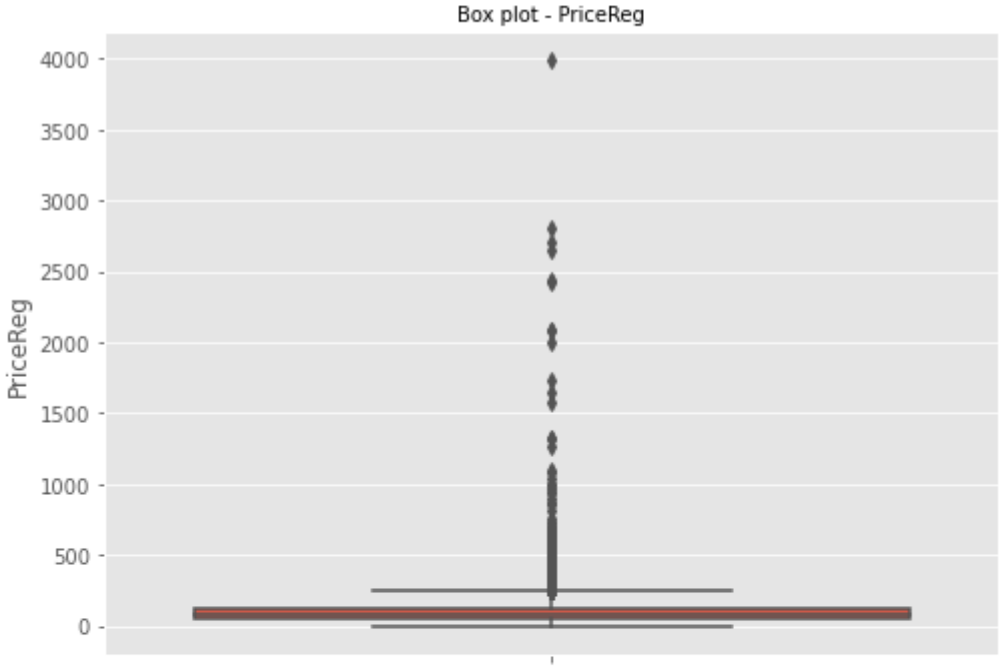


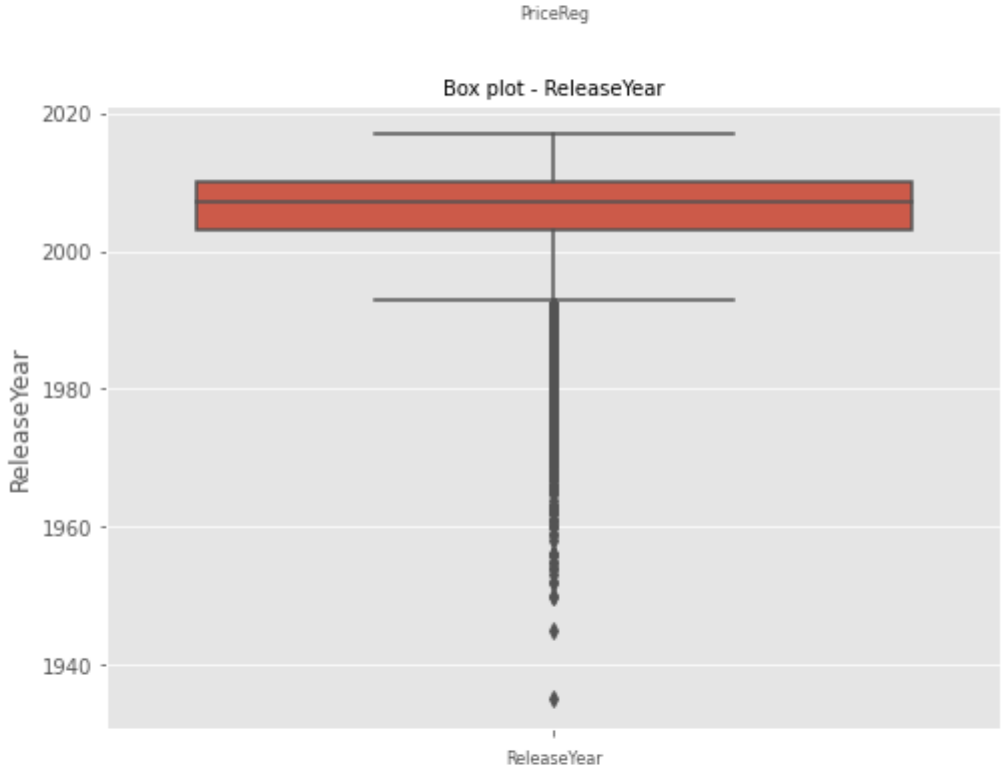
In [21]:
```
col_names = ['StrengthFactor','PriceReg', 'ReleaseYear', 'ItemCount', 'LowUserPrice', 'LowNetPrice']

fig, ax = plt.subplots(len(col_names), figsize=(16,12))

for i, col_val in enumerate(col_names):

    sns.distplot(sales_data_hist[col_val], hist=True, ax=ax[i])
    ax[i].set_title('Freq dist '+col_val, fontsize=10)
    ax[i].set_xlabel(col_val, fontsize=8)
    ax[i].set_ylabel('Count', fontsize=8)

plt.show()
```

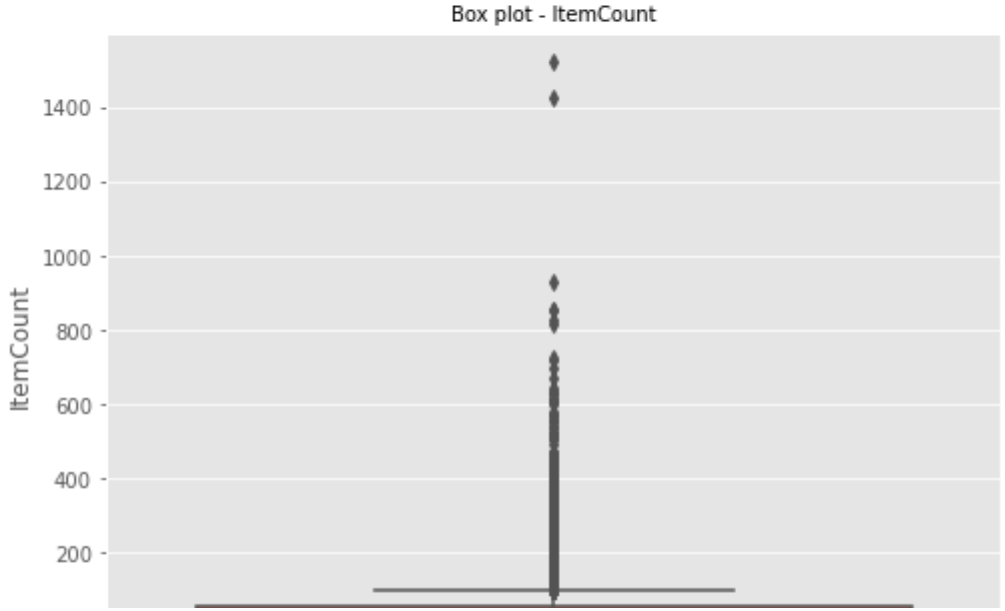

Freq dist StrengthFactor
1e−7
Count
Freq dist PriceReg
Freq dist ReleaseYear
Freq dist ItemCount
Freq dist LowUserPrice
Freq dist LowNetPrice
LowNetPrice

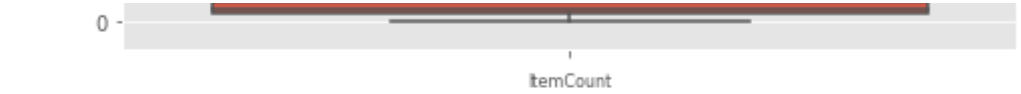
In [22]:
```
col_names = ['StrengthFactor','PriceReg', 'ReleaseYear', 'ItemCount', 'LowUserPrice', 'LowNetPrice']

fig, ax = plt.subplots(len(col_names), figsize=(8,40))

for i, col_val in enumerate(col_names):

    sns.boxplot(y=sales_data_hist[col_val], ax=ax[i])
    ax[i].set_title('Box plot - '+col_val, fontsize=10)
    ax[i].set_xlabel(col_val, fontsize=8)

plt.show()
```

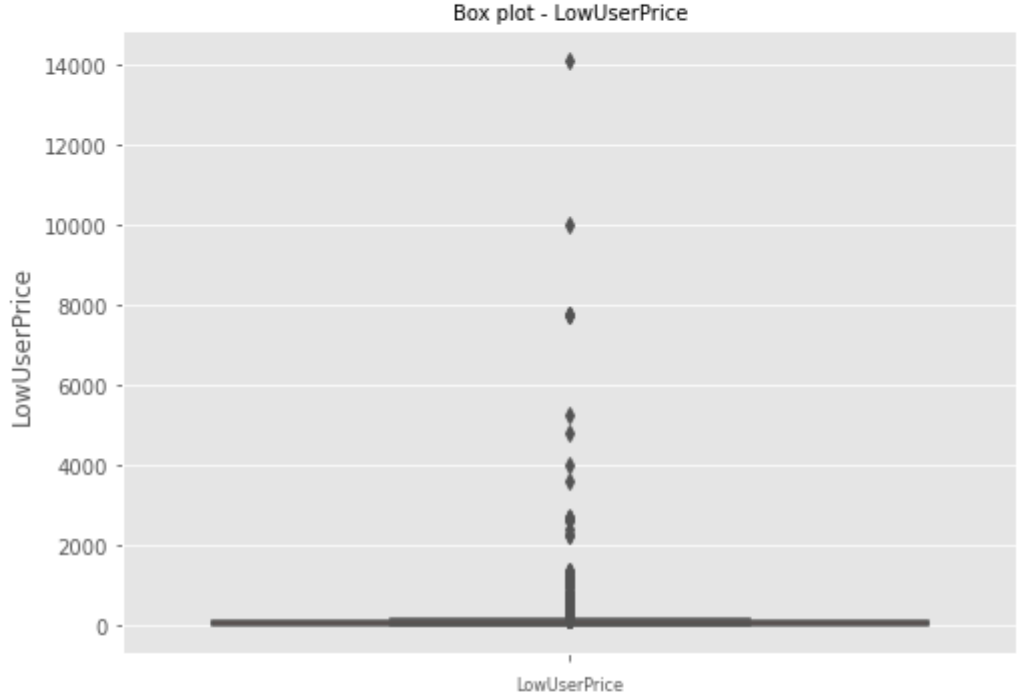

Box plot - StrengthFactor
1e7
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
StrengthFactor
StrengthFactor

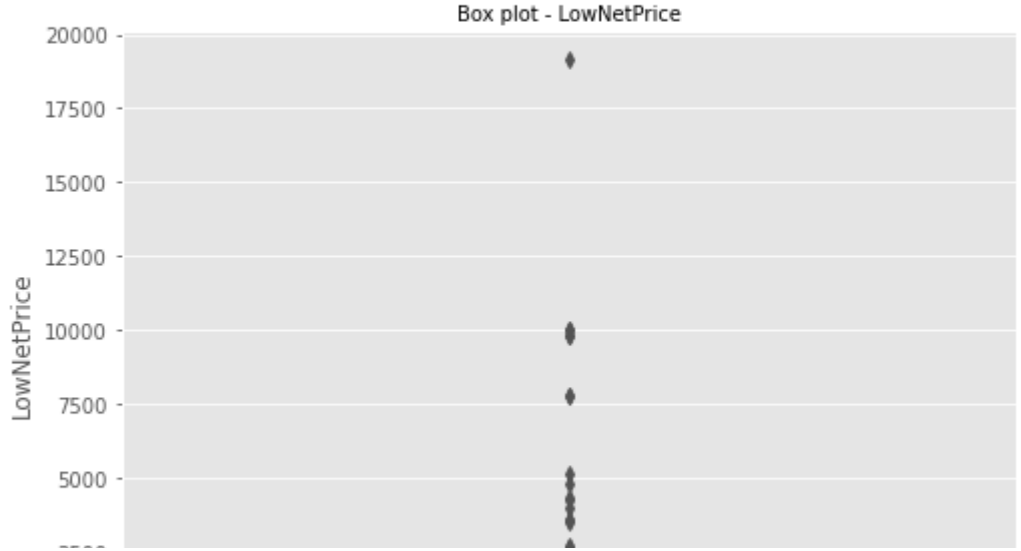

Box plot - PriceReg
4000
3500
3000
2500
2000
1500
1000
500
0
PriceReg

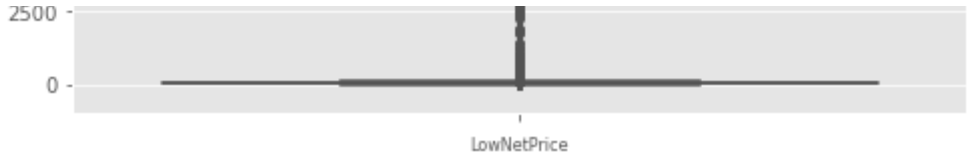

PriceReg
Box plot - ReleaseYear
2020
2000
1980
1960
1940
ReleaseYear
ReleaseYear


Box plot - ItemCount
1400
1200
1000
800
600
400
200
ItemCount


0
ItemCount


Box plot - LowUserPrice
14000
12000
10000
8000
6000
4000
2000
0
LowUserPrice
LowUserPrice

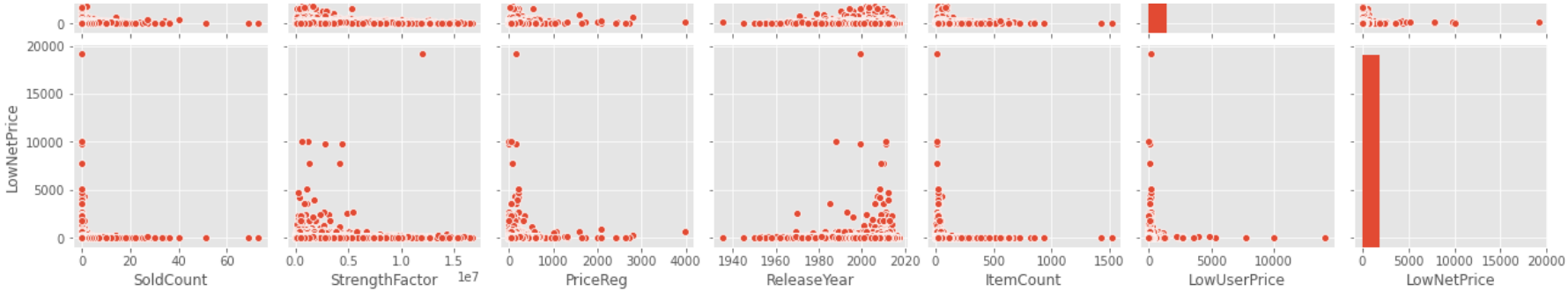

Box plot - LowNetPrice
20000
17500
15000
12500
10000
7500
5000
LowNetPrice


2500
0
LowNetPrice

In [23]:
```
sales_data_hist = sales_data_hist.drop(['Order', 'File_Type','SKU_number','SoldFlag','MarketingType','Rel
easeNumber','New_Release_Flag'], axis=1)
sns.pairplot(sales_data_hist)
```

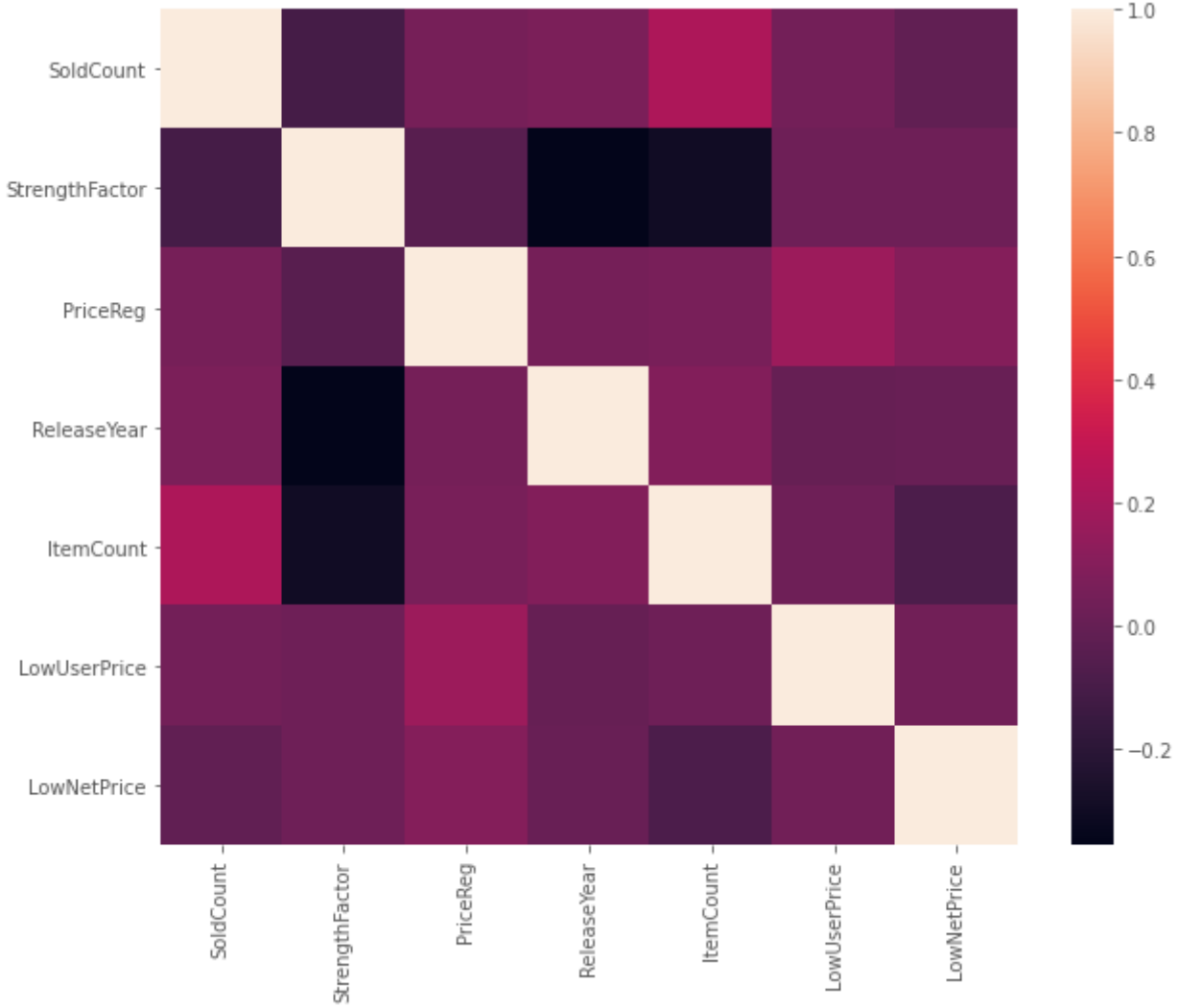
Out[23]: <seaborn.axisgrid.PairGrid at 0x7f9b99199850>


SoldCount
StrengthFactor
1e7
PriceReg
ReleaseYear
ItemCount
LowUserPrice

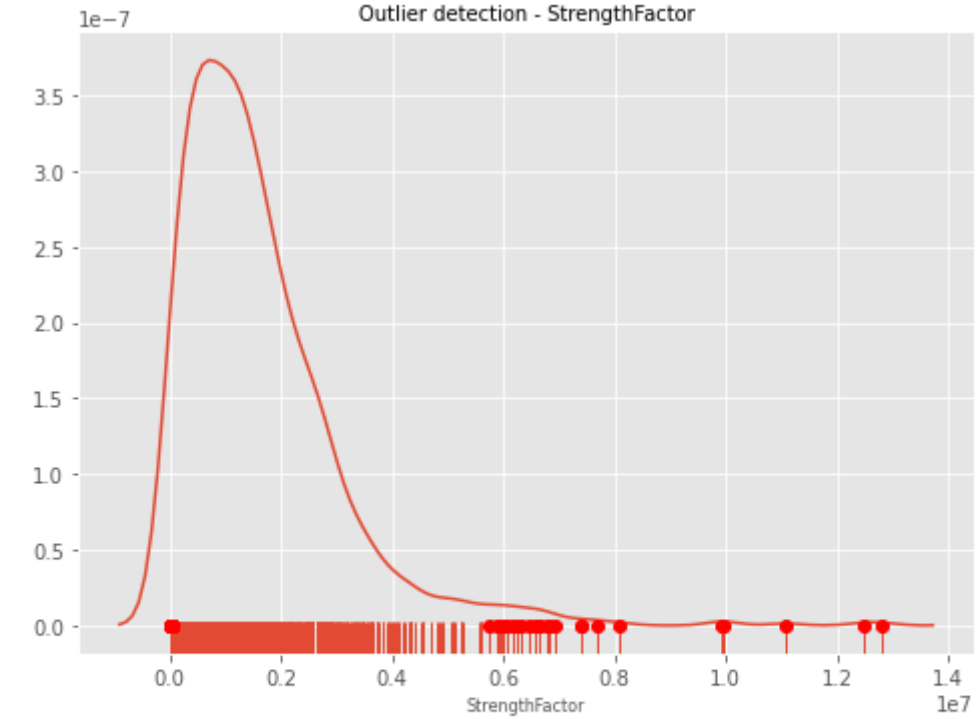

LowNetPrice
SoldCount
StrengthFactor
1e7
PriceReg
ReleaseYear
ItemCount
LowUserPrice
LowNetPrice

In [24]:
```
f, ax = plt.subplots(figsize=(10, 8))
corr = sales_data_hist.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
```

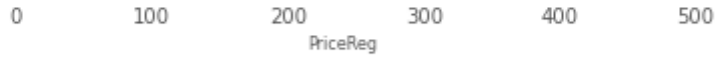
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7f9b9a65e310>

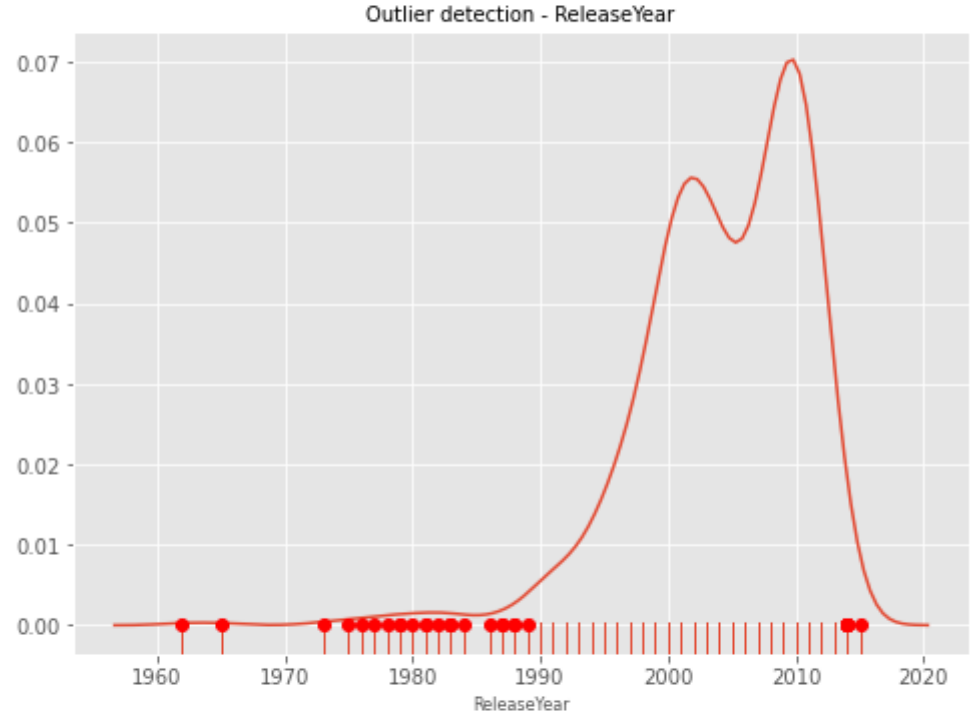


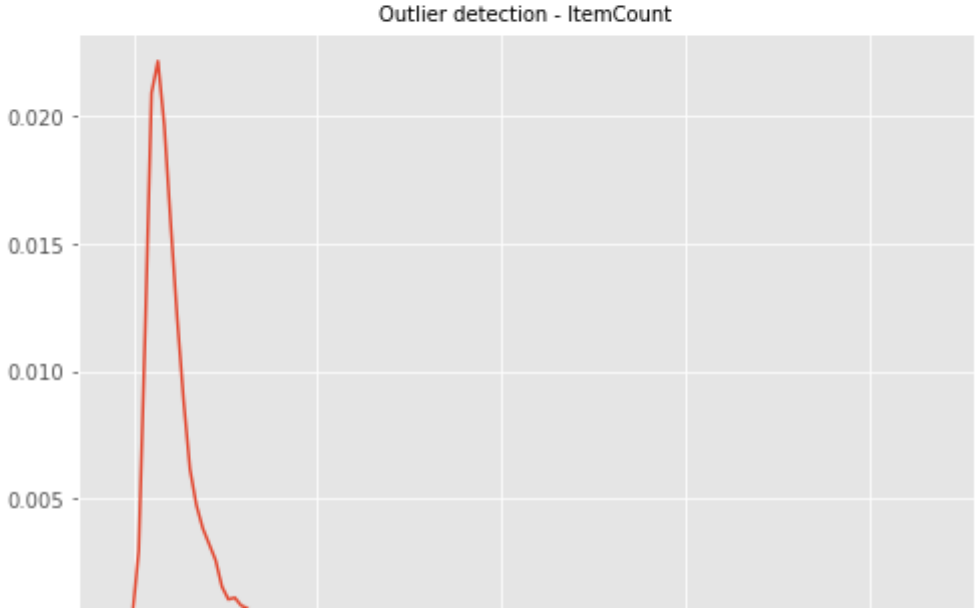
In [25]:
```
# Percentile based outlier removal
def percentile_based_outlier(data, threshold=95):
    diff = (100 - threshold) / 2.0
    minval, maxval = np.percentile(data, [diff, 100 - diff])
    return (data < minval) | (data > maxval)

col_names = ['StrengthFactor','PriceReg', 'ReleaseYear', 'ItemCount', 'LowUserPrice', 'LowNetPrice']

fig, ax = plt.subplots(len(col_names), figsize=(8,40))

for i, col_val in enumerate(col_names):
    x = sales_data_hist[col_val][:1000]
    sns.distplot(x, ax=ax[i], rug=True, hist=False)
    outliers = x[percentile_based_outlier(x)]
    ax[i].plot(outliers, np.zeros_like(outliers), 'ro', clip_on=False)

    ax[i].set_title('Outlier detection - '+col_val, fontsize=10)
    ax[i].set_xlabel(col_val, fontsize=8)

plt.show()
```

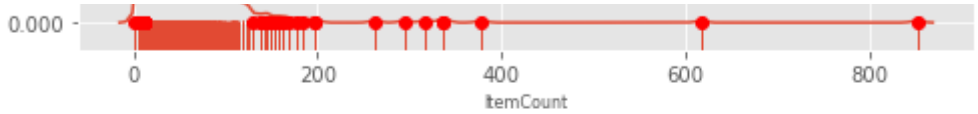

Outlier detection - StrengthFactor
1e−7
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
StrengthFactor
1e7

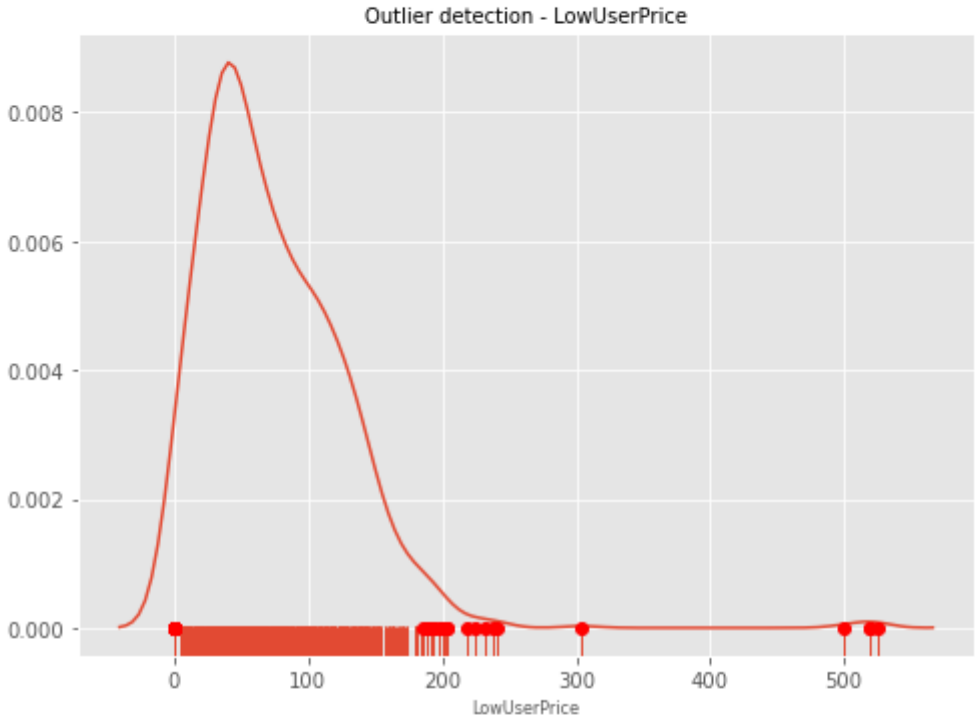

Outlier detection - PriceReg
0.006
0.005
0.004
0.003
0.002
0.001
0.000

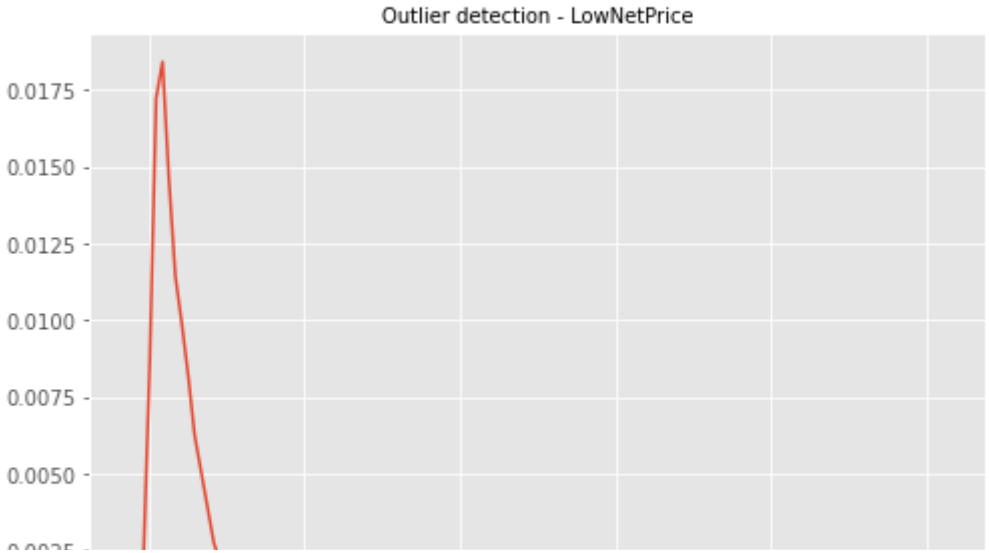

0
100
200
300
400
500
PriceReg


Outlier detection - ReleaseYear
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
1960
1970
1980
1990
2000
2010
2020
ReleaseYear


Outlier detection - ItemCount
0.020
0.015
0.010
0.005


0.000
0
200
400
600
800
ItemCount


Outlier detection - LowUserPrice
0.008
0.006
0.004
0.002
0.000
0
100
200
300
400
500
LowUserPrice


Outlier detection - LowNetPrice
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050

In [ ]: