BLACK FRIDAY SALE

We have collected the purchase data of a superstore on the Black Friday Sale. We will analyse the data for better inventory management and increasing the sale for the future Black Friday Sale.

We will use Python libraries such as Numpy, Pandas, Scipy, Matplotlib, Seaborn, Plotly, Scikit-Learn, etc for achieving our goal.

IMPORT LIBRARIES

In [1]:

```
# We will import all packages before we import our data set

%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
from matplotlib import pyplot
import scipy.stats as stats
from scipy.stats import chisquare
sns.set(style='ticks', context='talk')
import plotly.plotly as py
import plotly
plotly.tools.set_credentials_file(username='saurabhone', api_key='D1gcJm0ztIvbBba8OMl2')
```

Out[1]:

	User_ID	Product_ID	Gender	Age	State	Marital_Status	Apparels	Electronics	Furniture	Purchase
0	1000001	P00069042	F	0-17	NYC	0	3	NaN	NaN	8370
1	1000001	P00248942	F	0-17	NYC	0	1	6.0	14.0	15200
2	1000001	P00087842	F	0-17	NYC	0	12	NaN	NaN	1422
3	1000001	P00085442	F	0-17	NYC	0	12	14.0	NaN	1057
4	1000002	P00285442	M	55+	PA	0	8	NaN	NaN	7969

IMPORT DATA

In [ ]:

```
# reading data set using pandas function

d = pd.read_csv('/Users/saurabhkarambalkar/Desktop/bb/data.csv')
d.head()
```

CLEANING DATA

In [2]:

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65499 entries, 0 to 65498
Data columns (total 10 columns):
User_ID           65499 non-null int64
Product_ID        65499 non-null object
Gender            65499 non-null object
Age               65499 non-null object
State             65499 non-null object
```

```
State              65499 non-null object
Marital_Status     65499 non-null int64
Apparels           65499 non-null int64
Electronics        44908 non-null float64
Furniture          19886 non-null float64
Purchase           65499 non-null int64
dtypes: float64(2), int64(4), object(4)
memory usage: 5.0+ MB
```

We see that there are missing values in Electronics and Furniture which means that people did not buy from these two departments and thus we need to fill them with the value zero.

In [3]:

```
d.fillna(value=0,inplace=True)
d.head()
```

Out[3]:

	User_ID	Product_ID	Gender	Age	State	Marital_Status	Apparels	Electronics	Furniture	Purchase
0	1000001	P00069042	F	0-17	NYC	0	3	0.0	0.0	8370
1	1000001	P00248942	F	0-17	NYC	0	1	6.0	14.0	15200
2	1000001	P00087842	F	0-17	NYC	0	12	0.0	0.0	1422
3	1000001	P00085442	F	0-17	NYC	0	12	14.0	0.0	1057
4	1000002	P00285442	M	55+	PA	0	8	0.0	0.0	7969

In [4]:

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65499 entries, 0 to 65498
Data columns (total 10 columns):
User_ID            65499 non-null int64
Product_ID         65499 non-null object
Gender             65499 non-null object
Age                65499 non-null object
State              65499 non-null object
Marital_Status     65499 non-null int64
Apparels           65499 non-null int64
Electronics        65499 non-null float64
Furniture          65499 non-null float64
Purchase           65499 non-null int64
dtypes: float64(2), int64(4), object(4)
memory usage: 5.0+ MB
```

Now that we see there are no missing values in our data, we can start with exploration part.

EXPLORATORY DATA ANALYSIS

In [5]:

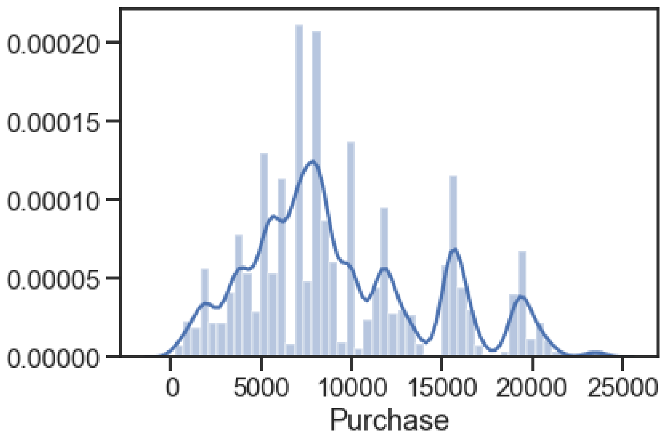
```
sns.distplot(d['Purchase'])
```

```
/usr/local/lib/python2.7/site-packages/scipy/stats/stats.py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` inst
ead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`,
which will result either in an error or a different result.
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x10bd8fe90>
```



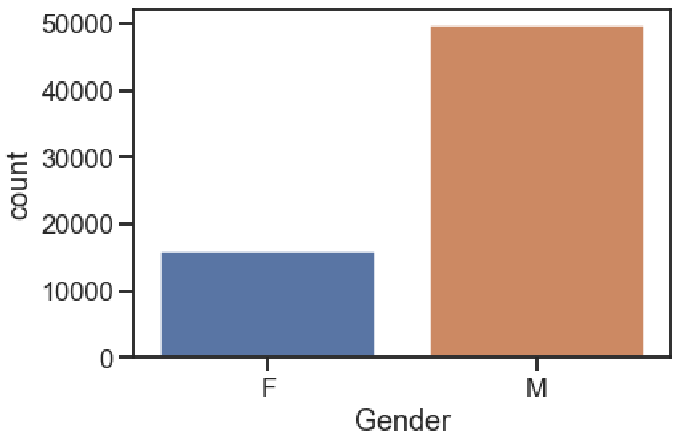
The purchase trend graph gives us an insight that most of the people purchased within the amount 5000 and 10000

In [6]:

```
# Countplot of Male and female(to compare the total count and the purchase done by the higher and
the lower gender)
sns.countplot(x ='Gender', data = d)
```

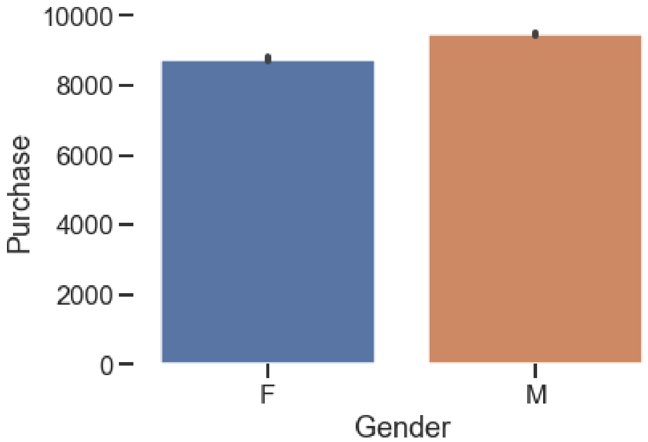
Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1048155d0>
```



In [8]:

```
#barplot for categorical vs numerical
sns.set_style('ticks')
sns.barplot(x='Gender',y='Purchase',data = d)
sns.despine(left=True,bottom=True)
```



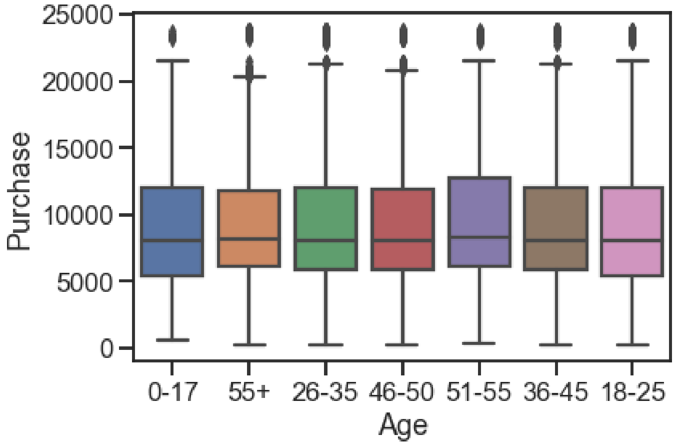
From the above two graphs we observe that there were more number of Males than Females who did the shopping. Also, even though the count of Females were low, the total amount spent by the Females is much closer to Males.

In [9]:

```
#Boxplot is used to check the purchase range by different age groups
sns.boxplot(x='Age',y='Purchase',data =d)
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x10ac3f650>
```



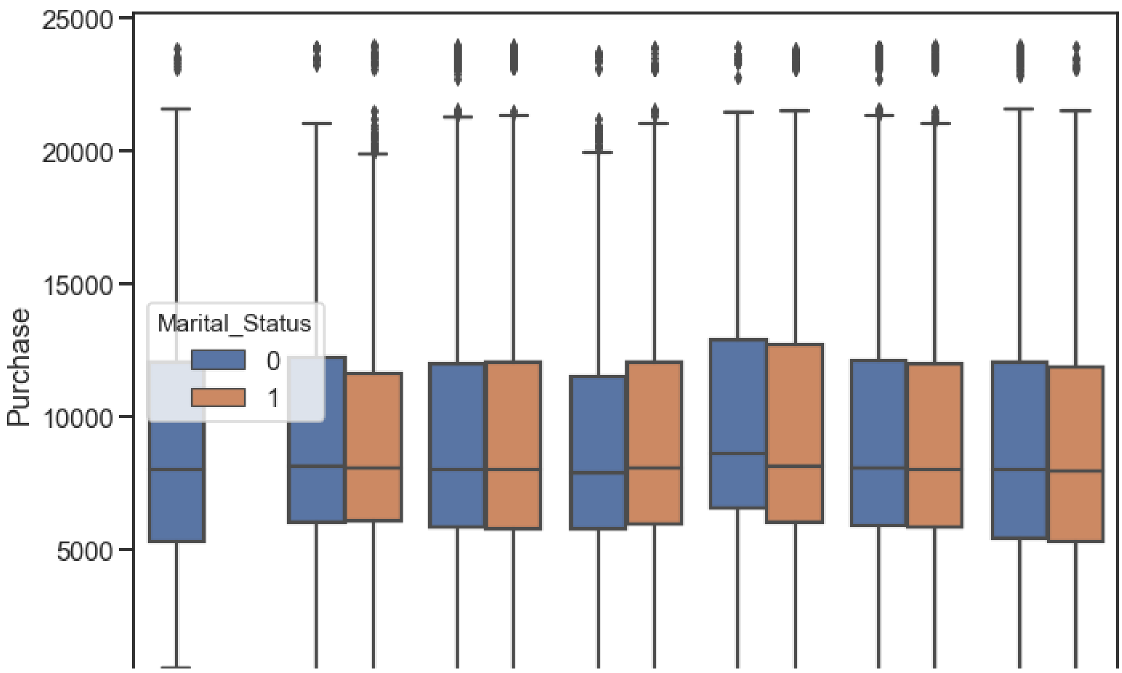
The age group 51-55 spent the most amount in shopping.

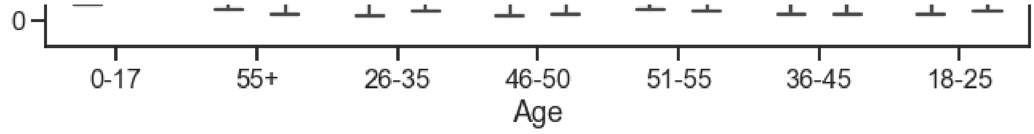
In [10]:

```
#Boxplot is used to check the purchase range by different age groups (which are further segregated by Marital status)
fig, ax = plt.subplots()
# the size of A4 paper
fig.set_size_inches(11, 8)
sns.boxplot(x='Age',y='Purchase',data =d,hue='Marital_Status', ax=ax)
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x10abd9f90>
```





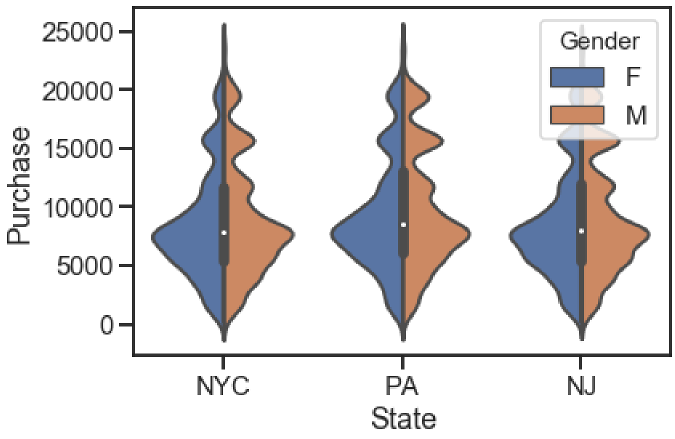
From the above graph it is clearly understood that the unmarried customers spent the most amount than the married customers.

In [11]:

```
sns.violinplot(x = 'State', y = 'Purchase', data = d, hue = 'Gender', split = True)
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x10ba0af10>
```



In [12]:

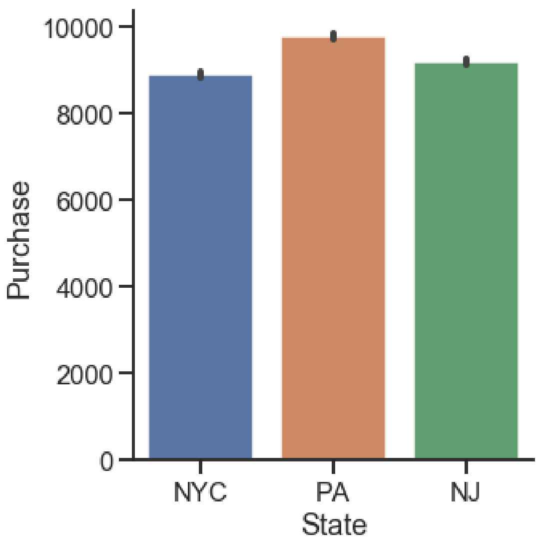
```
sns.factorplot(x = 'State', y = 'Purchase', data = d, kind = 'bar')
```

```
/usr/local/lib/python2.7/site-packages/seaborn/categorical.py:3666: UserWarning:

The `factorplot` function has been renamed to `catplot`. The original name will be removed in a future release. Please update your code. Note that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
```

Out[12]:

```
<seaborn.axisgrid.FacetGrid at 0x10ba0a690>
```



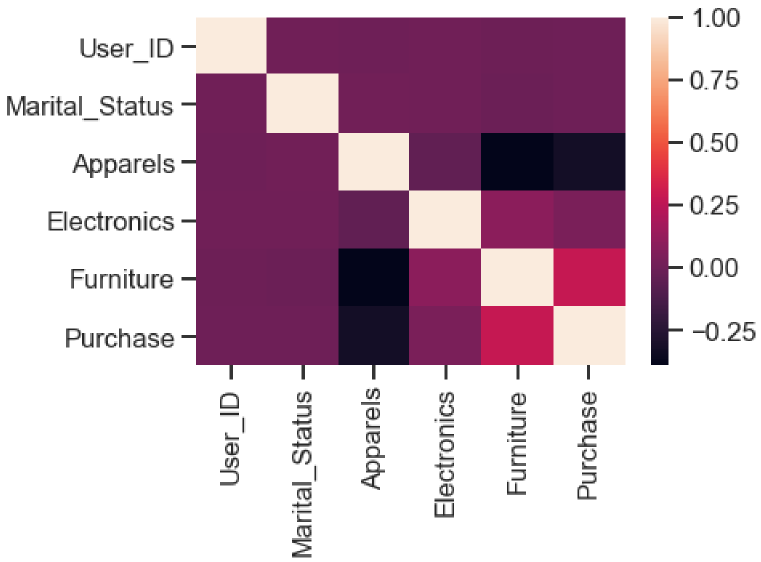
The bar graph shows that the State which spent the most is PA and the violin plot shows that the most purchases were made by PA Males.

In [13]:

```
tc = d.corr()
sns.heatmap(tc)
```

Out[13]:

<matplotlib.axes._subplots.AxesSubplot at 0x10ba0a2d0>



From the correlation graph we observe that the Furniture is more correlated to the Purchase.

We will plot some interesting interactive plotly graphs for insights

In [14]:

```
from plotly import __version__
import plotly.plotly as py
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs,init_notebook_mode,plot,iplot
plotly.offline.init_notebook_mode(connected=True)
import cufflinks as cf
cf.go_offline()
```

In [15]:

```
df = pd.DataFrame(data = d, columns = ['Apparels','Purchase'])
df.iplot()
```

In [18]:

```
d.iplot(kind = 'surface', colorscale='rdylbu')
```

In [19]:

```
df1 = pd.DataFrame(data = d, columns = ['State','Gender','Purchase'])
df1.iplot()
```

In [20]:

```
df2 = pd.DataFrame(data = d, columns = ['State','Gender','Apparels','Purchase'])
```

In [21]:

```
df2.iplot()
```

In [22]:

```
#We use pd.to_numeric function to change the data type to integer

d.Electronics = pd.to_numeric(d.Electronics, errors='coerce')
d = d.dropna(subset=['Electronics'])
d.Electronics = d.Electronics.astype(int)

d.Furniture = pd.to_numeric(d.Furniture, errors='coerce')
d = d.dropna(subset=['Furniture'])
d.Furniture = d.Furniture.astype(int)
```

In [23]:

```
final= d[['State', 'Apparels', 'Electronics','Furniture','Purchase']].copy()
final[:5]
```

Out[23]:

	State	Apparels	Electronics	Furniture	Purchase

	State	Apparels	Electronics	Furniture	Purchase
0	NYC	3	0	0	8370
1	NYC	1	6	14	15200
2	NYC	12	0	0	1422
3	NYC	12	14	0	1057
4	PA	8	0	0	7969

In [24]:

```
NYC= final[(final['State'] == 'NYC')]
print('Apparels:',sum(NYC['Apparels']))
print('Electronics:',sum(NYC['Electronics']))
print('Furniture:',sum(NYC['Furniture']))
NYC_apparels=sum(NYC['Apparels'])
NYC_electronics=sum(NYC['Electronics'])
NYC_furniture=sum(NYC['Furniture'])
NYC_Purchase=NYC_apparels+NYC_electronics+NYC_furniture
```

```
('Apparels:', 96877)
('Electronics:', 117498)
('Furniture:', 63256)
```

In [25]:

```
NJ= final[(final['State'] == 'NJ')]
print('Apparels:',sum(NJ['Apparels']))
print('Electronics:',sum(NJ['Electronics']))
print('Furniture:',sum(NJ['Furniture']))
NJ_apparels=sum(NJ['Apparels'])
NJ_electronics=sum(NJ['Electronics'])
NJ_furniture=sum(NJ['Furniture'])
NJ_Purchase=NJ_apparels+NJ_electronics+NJ_furniture
```

```
('Apparels:', 147206)
('Electronics:', 187289)
('Furniture:', 105928)
```

In [26]:

```
PA= final[(final['State'] == 'PA')]
print('Apparels:',sum(PA['Apparels']))
print('Electronics:',sum(PA['Electronics']))
print('Furniture:',sum(PA['Furniture']))
PA_apparels=sum(PA['Apparels'])
PA_electronics=sum(PA['Electronics'])
PA_furniture=sum(PA['Furniture'])
PA_Purchase=PA_apparels+PA_electronics+PA_furniture
```

```
('Apparels:', 103400)
('Electronics:', 138673)
('Furniture:', 83528)
```

In [27]:

```
f = {'State':['NY','NJ','PA'],
     'Apparels': [NYC_apparels,NJ_apparels,PA_apparels],
     'Electronics': [NYC_electronics, NJ_electronics,PA_electronics],
     'Furniture': [NYC_furniture,NJ_furniture,PA_furniture],
     'Purchase':[NYC_Purchase,NJ_Purchase,PA_Purchase]
    }
f1 = pd.DataFrame(data=f)
print(f1)
```

```
   Apparels  Electronics  Furniture  Purchase State
0     96877       117498      63256    277631    NY
1    147206       187289     105928    440423    NJ
2    103400       138673      83528    325601    PA
```

In [28]:

```
for col in f1.columns:
    f1[col] = f1[col].astype(str)

scl = [[0.0, 'rgb(242,240,247)'],[0.2, 'rgb(218,218,235)'],[0.4, 'rgb(188,189,220)'],\
            [0.6, 'rgb(158,154,200)'],[0.8, 'rgb(117,107,177)'],[1.0, 'rgb(84,39,143)']]

f1['text'] = f1['State'] + '<br>' +\
    'Apparels '+f1['Apparels']+' Electronics '+f1['Electronics']+'<br>'+\
    'Furniture '+f1['Furniture']



data = [ dict(
        type='choropleth',
        colorscale = scl,
        autocolorscale = False,
        locations = f['State'],
        z = f1['Purchase'].astype(float),
        locationmode = 'USA-states',
        text = f1['text'],
        marker = dict(
            line = dict (
                color = 'rgb(255,255,255)',
                width = 2
            ) ),
        colorbar = dict(
            title = "USD")
        ) ]

layout = dict(
        title = '2017 Black Friday Sale in Tri State Region<br>(Hover for breakdown)',
        geo = dict(
            scope='usa',
            projection=dict( type='albers usa' ),
            showlakes = True,
            lakecolor = 'rgb(255, 255, 255)'),
             )

fig = dict( data=data, layout=layout )
py.iplot( fig, filename='d3-cloropleth-map')
```

High five! You successfully sent some data to your account on plotly. View your plot in your browser at https://plot.ly/~saurabhone/0 or inside your plot.ly account where it is named 'd3-cloropleth-map'

Out[28]:

PREDICTION MODELING

Now using different features we will predict the Purchase which can help us in the future Black Friday Sales

In [54]:

```
#Selecting the features for prediction

data = d[['Gender','Apparels', 'Furniture', 'Electronics','Purchase']]
X = data.iloc[:,:-1].values
y = data.iloc[:,4].values
```

In [55]:

```
#Encoding the categorical variable

from sklearn.preprocessing import LabelEncoder, OneHotEncoder
labelencoder_X = LabelEncoder()
X[:,0] = labelencoder_X.fit_transform(X[:,0])
onehotencoder = OneHotEncoder(categorical_features=[0])
X = onehotencoder.fit_transform(X).toarray()
```

In [56]:

```
#Adding the dummy variable trap

X = X[:,1:]
```

In [57]:

```
#Spliting the data into train and test sets

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=101)
```

In [58]:

```
#Fitting Multiple Linear Regression Model on train set

from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train,y_train)
```

```
/usr/local/lib/python2.7/site-packages/sklearn/linear_model/base.py:509: RuntimeWarning:

internal gelsd driver lwork query error, required iwork dimension not returned. This is likely the
result of LAPACK bug 0038, fixed in LAPACK 3.2.2 (released July 21, 2010). Falling back to 'gelss'
driver.
```

Out[58]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [60]:

```
#Predicting the test set results

y_pred = reg.predict(X_test)
y_pred
```

Out[60]:

```
array([11346.8380932 ,  9730.51230332,  7545.69786319, ...,
        9740.72228087,  7412.96815504,  9607.77776017])
```

In [92]:

```
#Building the optimum model using Backward Elimination Technique

import statsmodels.formula.api as sm
X=np.append(arr = np.ones((65499,1)).astype(int),values=X,axis=1)
X_opt = X[:,[0,1,2,3,4]]
reg_OLS = sm.OLS(endog = y, exog = X_opt).fit()
reg_OLS.summary()
```

Out[92]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.133
Model:	OLS	**Adj. R-squared:**	0.133
Method:	Least Squares	**F-statistic:**	2515.
Date:	Sat, 06 Oct 2018	**Prob (F-statistic):**	0.00
Time:	21:56:55	**Log-Likelihood:**	-6.4559e+05
No. Observations:	65499	**AIC:**	1.291e+06
Df Residuals:	65494	**BIC:**	1.291e+06
Df Model:	4		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.657e+13	1.02e+14	0.163	0.871	-1.83e+14	2.16e+14
x1	-1.657e+13	1.02e+14	-0.163	0.871	-2.16e+14	1.83e+14
x2	499.8824	42.214	11.842	0.000	417.143	582.622
x3	-330.1878	5.224	-63.206	0.000	-340.427	-319.949
x4	182.9683	3.889	47.043	0.000	175.345	190.591

Omnibus:	6533.671	**Durbin-Watson:**	1.721
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	8912.282
Skew:	0.817	**Prob(JB):**	0.00
Kurtosis:	3.771	**Cond. No.**	5.09e+13

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.03e-21. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [94]:

```
#The p value of Gender is more than our significance value of 0.05 (5%) so we need eliminate it fr
om our model

X_opt = X[:,[0,2,3,4]]
reg_OLS = sm.OLS(endog = y, exog = X_opt).fit()
reg_OLS.summary()
```

Out[94]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.133
Model:	OLS	**Adj. R-squared:**	0.133

Model:	OLS	**Adj. R-squared:**	0.133
Method:	Least Squares	**F-statistic:**	3354.
Date:	Sat, 06 Oct 2018	**Prob (F-statistic):**	0.00
Time:	22:10:24	**Log-Likelihood:**	-6.4559e+05
No. Observations:	65499	**AIC:**	1.291e+06
Df Residuals:	65495	**BIC:**	1.291e+06
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	9783.1102	47.100	207.708	0.000	9690.793	9875.427
x1	499.9178	42.213	11.843	0.000	417.180	582.656
x2	-330.1852	5.224	-63.206	0.000	-340.424	-319.946
x3	182.9722	3.889	47.045	0.000	175.349	190.595

Omnibus:	6533.635	**Durbin-Watson:**	1.721
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	8912.219
Skew:	0.817	**Prob(JB):**	0.00
Kurtosis:	3.771	**Cond. No.**	20.5

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

The model we have obtained is best suitable for prediction of Purchase for future Black Friday Sales