BLACK FRIDAY SALE ANALYSIS

NAME: AKSHATA ABHAY BHENDE

RUID:197007541

INTRODUCTION

A retail company wants to **understand the customer purchase behavior (specifically, purchase amount) against various products of different categories.** They have shared purchase summary of various customers for a selected high-volume products from last month.

They want to build a model to predict the purchase amount of customer against various products which will help them to create personalized offer for customers against different products.

PROBLEM STATEMENT

- To perform Exploratory Data Analysis on the given dataset.
- To identify the most important variables and to define the best regression model for predicting out target variable.

DATA

Let's start by importing some libraries and our data.

```
%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')
```

```
d=pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
d.head()
```

Dataset was obtained from Kaggle.It was already divided in test.csv and train.csv.

	User_ID	Product_ID	Gender	Age	Occupation	State	Stay_In_Current_City_Years	Marital_Status	Furniture	Clothing	Electronics	Purchase
0	1000001	P00069042	M	0-17	10	NJ	2	Single	3	NaN	NaN	8370
1	1000001	P00248942	M	0-17	10	NJ	2	Single	1	6.0	14.0	15200
2	1000001	P00087842	M	0-17	10	NJ	2	Single	12	NaN	NaN	1422
3	1000001	P00085442	M	0-17	10	NJ	2	Single	12	14.0	NaN	1057
4	1000002	P00285442	F	55+	16	CA	4+	Single	8	NaN	NaN	7969

```
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 12 columns):
User_ID                       550068 non-null int64
Product_ID                    550068 non-null object
Gender                        550068 non-null object
Age                           550068 non-null object
Occupation                    550068 non-null int64
State                         550068 non-null object
Stay_In_Current_City_Years    550068 non-null object
Marital_Status                550068 non-null object
Furniture                     550068 non-null int64
Clothing                      376430 non-null float64
Electronics                   166821 non-null float64
Purchase                      550068 non-null int64
dtypes: float64(2), int64(4), object(6)
memory usage: 50.4+ MB
```

```
d.describe()
```

	User_ID	Occupation	Furniture	Clothing	Electronics	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	5.404270	6.735436	3.841941	9263.968713
std	1.727592e+03	6.522660	3.936211	6.215492	6.250712	5023.065394
min	1.000001e+06	0.000000	1.000000	0.000000	0.000000	12.000000
25%	1.001516e+06	2.000000	1.000000	0.000000	0.000000	5823.000000
50%	1.003077e+06	7.000000	5.000000	5.000000	0.000000	8047.000000
75%	1.004478e+06	14.000000	8.000000	14.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	20.000000	18.000000	18.000000	23961.000000

Considering Gender,Age,State,Furniture,Clothing,Electronics as the predictors that will influence more the amount spent by a customer on this day. While **Purchase** is target variable.

Our goal as is to identify the most important variables and to define the best regression model for predicting out target variable. Hence, this analysis will be divided into five stages:

1. Exploratory data analysis (EDA);
2. Data Pre-processing;

3. Feature engineering;

4. Modeling;

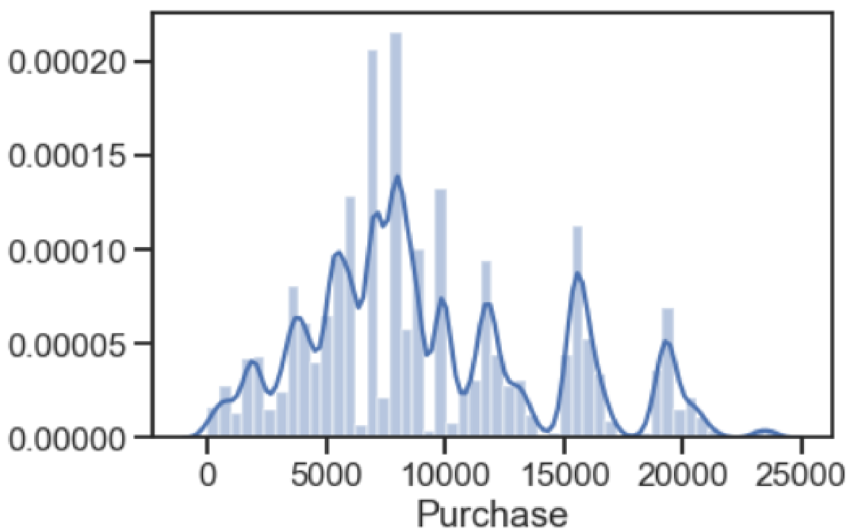
1.EXPLORATORY DATA ANALYSIS (EDA)

How about we play out some essential data exploration and come up with some inference. Thus, the objective for this segment is to take a look on the data as well as any irregularities so that we can address them on the next section, **Data Pre-Processing.**

1.1.1)Distribution of the target variable:**Purchase**

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x170ad788e48>
```

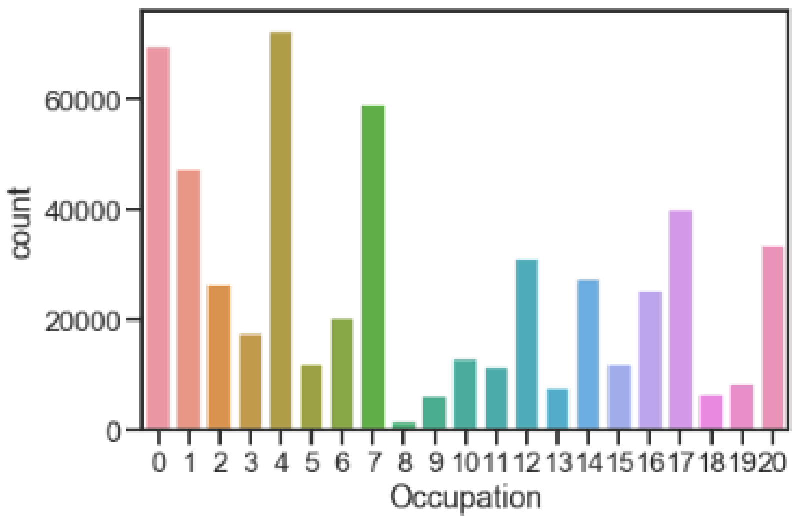


It seems like our target variable has an Gaussian distribution.

```
print ("Skew is:", d.Purchase.skew())
print("Kurtosis: %f" % d.Purchase.kurt())
```

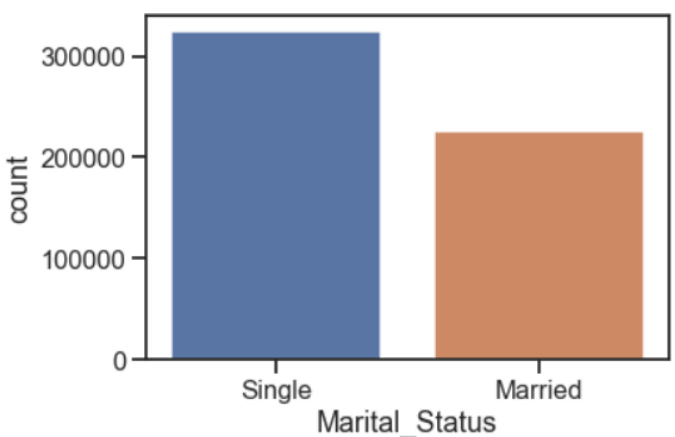
```
Skew is: 0.6001400037087128
Kurtosis: -0.338378
```

1.1.2)Distribution of the variable:**Occupation**



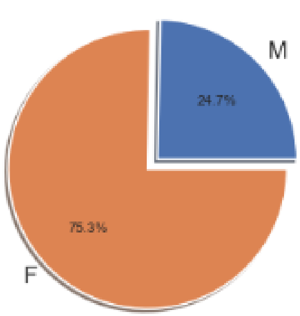
Occupation has at least 20 different values. Since we do not known to each occupation each number corresponds, is difficult to make any analysis.

1.1.3)Distribution of the variable: **Marital_Status**



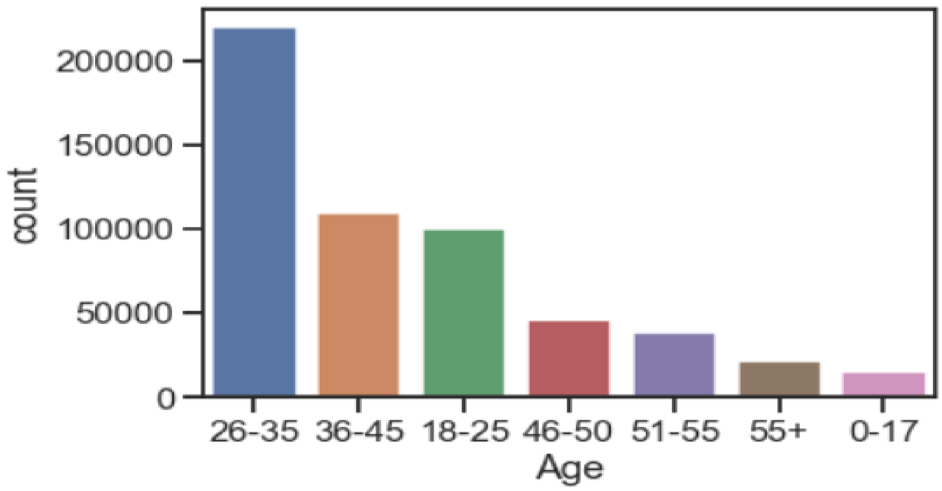
There are more single people buying products on Black Friday than married people, but do they spend more?

1.1.4)Distribution of the variable: **Gender**



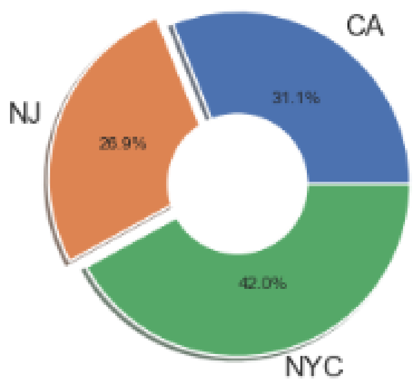
Most of the buyers are females, but who spends more on each purchase: man or woman?

1.1.5) Distribution of the variable: **Age**



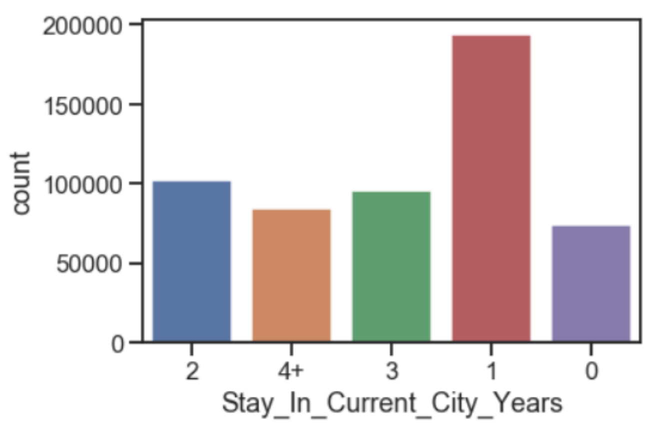
As expected, most purchases are made by people between 18 to 45 years old.

1.1.6)Distribution of the variable: **State**



New York City has higher sales than New Jersey and California.

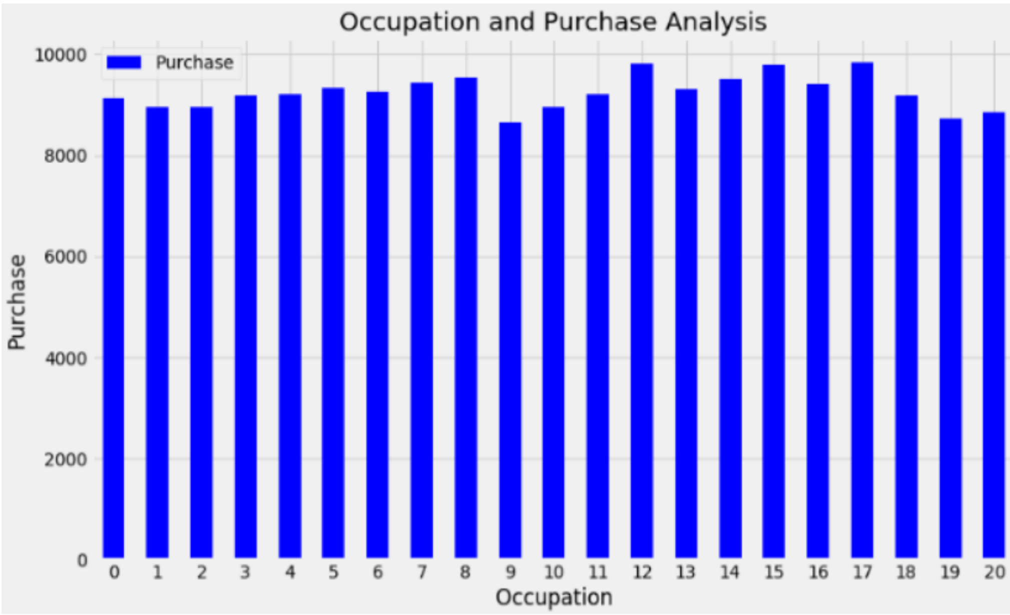
1.1.7)Distribution of the variable: **Stay_In_Current_City_Years**



The tendency looks like the longest somebody is living in that city the less inclined they are to purchase new things. Henceforth, in the event that somebody is new around the local area and requirements an extraordinary number of new things for their home that they'll exploit the low costs in Black Friday to buy every one of the things required.

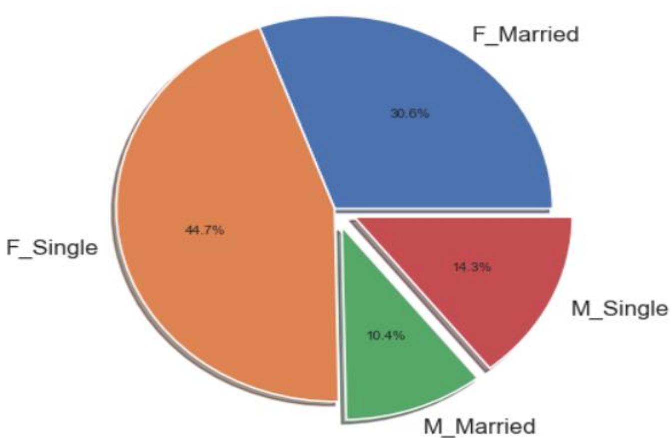
Firstly, we individually analysed some of the existent features, now it is time to understand the relationship between our target variable and predictors as well as the relationship among predictors.

1.2.1)**Occupation** and **Purchase** analysis



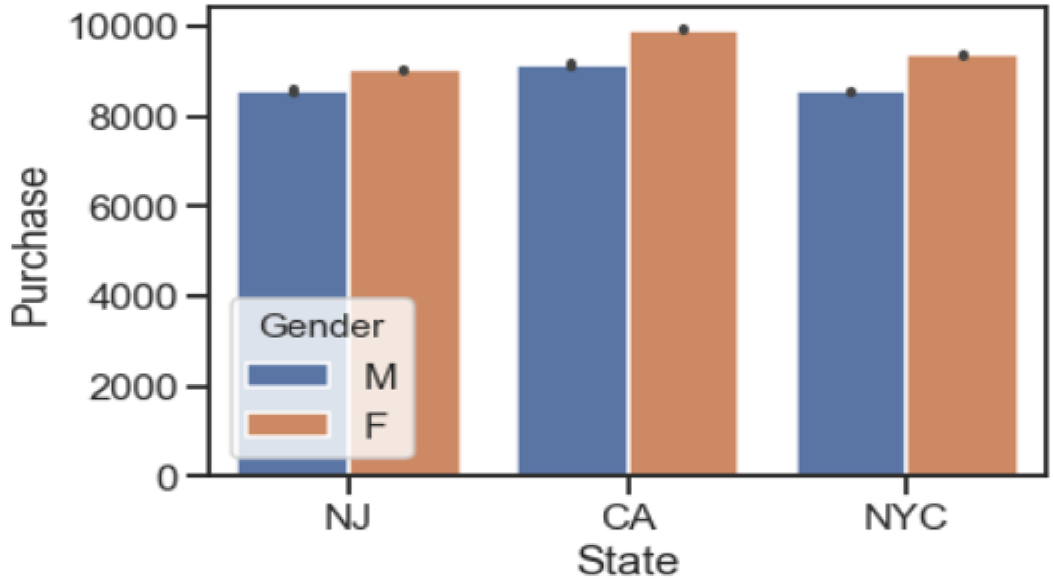
In spite of the fact that there are a few occupations which have higher representation, it appears that the amount every customer spends is pretty much the equivalent for all occupations.

1.2.2)**Gender,Marital Status** and **Purchase** analysis



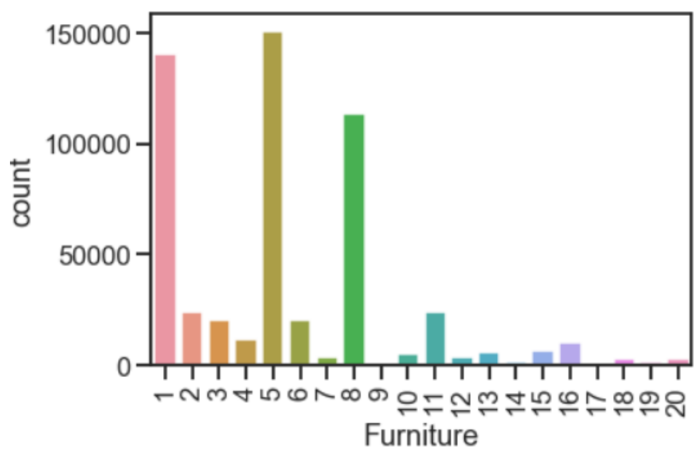
It seems that Single Females tends to purchase more than anyone else.

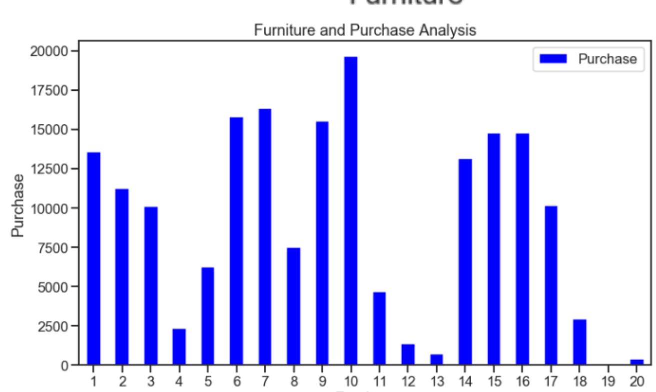
1.2.3)**State**,**Gender** and **Purchase** analysis



Even though there are more buyers from NYC ,CA buyers spend most amount during the sale.

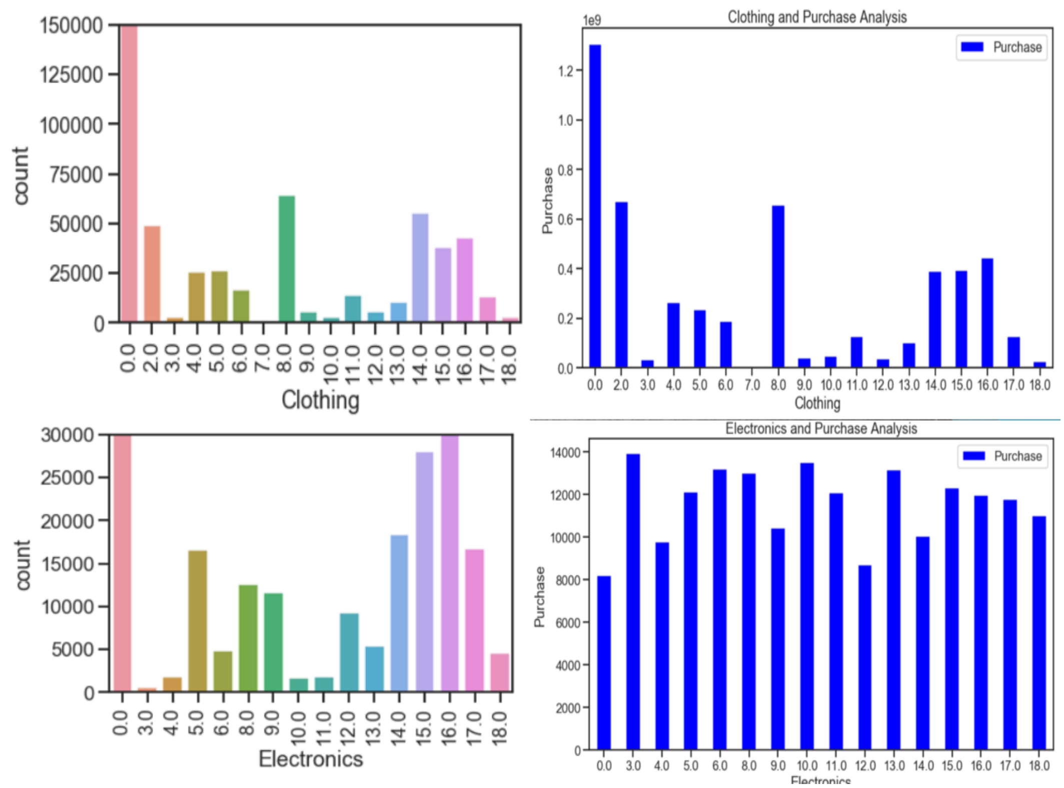
1.2.4)**Product** and **Purchase** analysis





Even though Furniture 1,5,8 have the highest sum of sales as shown above but the amount spent on those three is not the highest.

We can see the same behaviour for Clothing and Electronics as shown below.



1.3)Correlation between numeric predictors and target variable

```
sns.heatmap(corr, vmax=.9,annot_kws={'size': 20}, annot=True);
```

```
s = corr.unstack()
s
```

```
User_ID      User_ID        1.000000
             Occupation    -0.023971
             Furniture      0.003825
             Clothing       0.003896
             Electronics    0.003605
             Purchase       0.004716
Occupation   User_ID       -0.023971
             Occupation     1.000000
             Furniture     -0.007618
             Clothing       0.006712
             Electronics    0.012269
             Purchase       0.020833
Furniture    User_ID        0.003825
             Occupation    -0.007618
             Furniture      1.000000
             Clothing      -0.067877
             Electronics   -0.385534
             Purchase      -0.343703
Clothing     User_ID        0.003896
             Occupation     0.006712
             Furniture     -0.067877
             Clothing       1.000000
             Electronics    0.094750
             Purchase       0.052288

Electronics  User_ID        0.003605
             Occupation     0.012269
             Furniture     -0.385534
             Clothing       0.094750
             Electronics    1.000000
             Purchase       0.288501
Purchase     User_ID        0.004716
             Occupation     0.020833
             Furniture     -0.343703
             Clothing       0.052288
             Electronics    0.288501
             Purchase       1.000000
dtype: float64
```

From correlation graph we observe that Electronics is strongly correlated to purchase.

2.DATA PRE-PROCESSING

During our EDA we were able to take some conclusions regarding our first assumptions and the available data.We have to solve those problems:

i)Age: should be treated as numerical.

```
# Giving Age Numerical values
age_dict = {'0-17':0, '18-25':1, '26-35':2, '36-45':3, '46-50':4, '51-55':5, '55+':6}
data["Age"] = data["Age"].apply(lambda line: age_dict[line])

data["Age"].value_counts()
```

```
2    311554
3    155898
1    141209
4     64902
5     54450
6     30316
0     21185
Name: Age, dtype: int64
```

ii)State: We must convert this to numerical as well.

```
city_dict = {'NJ':0, 'NYC':1, 'CA':2}
data["State"] = data["State"].apply(lambda line: city_dict[line])

data["State"].value_counts()
```

```
1    328524
2    241487
0    209503
Name: State, dtype: int64
```

iii)Gender: We make to convert M and F to 1 and 0.

```
#Turn gender binary
gender_dict = {'F':0, 'M':1}
data["Gender"] = data["Gender"].apply(lambda line: gender_dict[line])

data["Gender"].value_counts()
```

```
0    587052
1    192462
Name: Gender, dtype: int64
```

iv)Furniture, Clothing, Electronics: Remove null values

```
#Check the percentage of null values per variable
data.isnull().sum()/data.shape[0]*100
```

```
User_ID                       0.000000
Product_ID                    0.000000
Gender                        0.000000
Age                           0.000000
Occupation                    0.000000
State                         0.000000
Stay_In_Current_City_Years    0.000000
Marital_Status                0.000000
Furniture                     0.000000
Clothing                      9.231472
Electronics                  20.743760
Purchase                     29.808452
Gender_MaritalStatus         29.808452
source                        0.000000
dtype: float64
```

```
data["Clothing"]= \
data["Clothing"].fillna(-2.0).astype("float")
```

```
data.Clothing.value_counts().sort_index()
```

```
-2.0      72344
 0.0     173638
 2.0      70498
 3.0       4123
 4.0      36705
 5.0      37165
 6.0      23575
 7.0        854
 8.0      91317
 9.0       8177
 10.0      4420
 11.0     20230
 12.0      7801
 13.0     15054
 14.0     78834
 15.0     54114
 16.0     61687
 17.0     19104
 18.0      4027
Name: Clothing, dtype: int64
```

```
data["Electronics"]= \
data["Electronics"].fillna(-2.0).astype("float")
```

```
data.Electronics.value_counts().sort_index()
```

```
-2.0     162562
 0.0     383247
 3.0        878
 4.0       2691
 5.0      23799
 6.0       6888
 8.0      17861
 9.0      16532
 10.0      2501
 11.0      2585
 12.0     13115
 13.0      7849
 14.0     26283
 15.0     39968
 16.0     46469
 17.0     23818
 18.0      6621
Name: Electronics, dtype: int64
```

3.FEATURE ENGINEERING

3.1)Function to create count features

```
# feature representing the count of each user
def getCountVar(compute_df, count_df, var_name):
    grouped_df = count_df.groupby(var_name)
    count_dict = {}
    for name, group in grouped_df:
        count_dict[name] = group.shape[0]

    count_list = []
    for index, row in compute_df.iterrows():
        name = row[var_name]
        count_list.append(count_dict.get(name, 0))
    return count_list
```

3.2)Exporting Data

```
#Divide into test and train:
train = data.loc[data['source']=="train"]
test = data.loc[data['source']=="test"]

#Drop unnecessary columns:
test.drop(['source'],axis=1,inplace=True)
train.drop(['source'],axis=1,inplace=True)

#Export files as modified versions:
train.to_csv("train_modified.csv",index=False)
test.to_csv("test_modified.csv",index=False)
```

4.MODELING

Select a Performance Measure

Usually for regression problems the typical performance measure is the Root Mean Square Error (RMSE). This function gives an idea of how much error the system makes in its predictions with higher weight for large errors.

```
train_df = pd.read_csv('train_modified.csv')
test_df = pd.read_csv('test_modified.csv')
```

Since, we'll be using many model, instead of defining function again and again, we well define a generic function which takes the algorithm and data as input and perform cross-validation and generate submission.

```
#Define target and ID columns:
target = 'Purchase'
IDcol = ['User_ID','Product_ID']
from sklearn.model_selection import cross_validate,cross_val_score
from sklearn import metrics


def modelfit(alg, dtrain, dtest, predictors, target, IDcol, filename):
    #Fit the algorithm on the data
    alg.fit(dtrain[predictors], dtrain[target])

    #Predict training set:
    dtrain_predictions = alg.predict(dtrain[predictors])

    #Perform cross-validation:
    cv_score = cross_val_score(alg, dtrain[predictors],(dtrain[target]) , cv=20, scoring='neg_mean_squared_error')
    cv_score = np.sqrt(np.abs(cv_score))



    #Print model report:
    print("\nModel Report")
    print("RMSE : %.4g" % np.sqrt(metrics.mean_squared_error((dtrain[target]).values, dtrain_predictions)))
    print("CV Score : Mean - %.4g | Std - %.4g | Min - %.4g | Max - %.4g" % (np.mean(cv_score),np.std(cv_score),np.min(cv_sc

    #Predict on testing data:
    dtest[target] = alg.predict(dtest[predictors])

    #Export submission file:
    IDcol.append(target)
    submission = pd.DataFrame({ x: dtest[x] for x in IDcol})
```

4.1)Linear Regression Model

```
from sklearn.linear_model import LinearRegression

LR = LinearRegression(normalize=True)
predictors = train_df.columns.drop(['Purchase','Product_ID','User_ID'])
modelfit(LR, train_df, test_df, predictors, target, IDcol, 'LR.csv')
```

```
Model Report
RMSE : 4631
CV Score : Mean - 4631 | Std - 31.33 | Min - 4573 | Max - 4687
```

4.2)Ridge Regression Model

```
from sklearn.linear_model import Ridge
RR = Ridge(alpha=0.05,normalize=True)
modelfit(RR, train_df, test_df, predictors, target, IDcol, 'RR.csv')
```

```
Model Report
RMSE : 4631
CV Score : Mean - 4631 | Std - 31.01 | Min - 4572 | Max - 4686
```

4.3)Decision Tree Regression Model

```
from sklearn.tree import DecisionTreeRegressor
DT = DecisionTreeRegressor(max_depth=15, min_samples_leaf=100)
modelfit(DT, train_df, test_df, predictors, target, IDcol, 'DT.csv')
```

```
Model Report
RMSE : 2914
CV Score : Mean - 2941 | Std - 20.86 | Min - 2907 | Max - 2975
```

4.4)Random Forest Regression Model

```
RF = DecisionTreeRegressor(max_depth=8, min_samples_leaf=150)
modelfit(RF, train_df, test_df, predictors, target, IDcol, 'RF.csv')
```

```
Model Report
RMSE : 2978
CV Score : Mean - 2981 | Std - 20.89 | Min - 2945 | Max - 3021
```

5.CONCLUSION

The Regression model that perform the best was Decision Tree Model with RMSE value of 2914 .So this will be the best fit to predict future sales and purchase for the Black Friday Sale.

REFERENCES:

1) https://www.kaggle.com/sdolezel/black-friday
2) https://seaborn.pydata.org/
3) https://python-graph-gallery.com/
4) https://towardsdatascience.com/simple-and-multiple-linear-regression-in-python-c928425168f9
5) https://www.geeksforgeeks.org/python-decision-tree-regression-using-sklearn/