In [232…

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm
import seaborn as sns
from matplotlib.pyplot import figure
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
from sklearn.utils import resample
from sklearn.metrics import accuracy_score
```

In [233…

```
df = pd.read_csv("walmart_data.csv")
```

Problem Statement

- **Primary Goal**
  - Recognizing **Purchase pattern** of Products wrt. Gender , Age , Occupation , Marital_Status, City_Category etc.
  - Indentifying **customer segments,profiling and formulating markerting strategy**
  - How to **drive sales of products and revenue** , across product categories
    - Data driven discounting / offers among customer segments
- **Statistical summary**
  - More **likelihood of purchase**
  - Range / Limitation of data
- **Long term benefits** : Sales growth , Customer acquisition and retention

In [234…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 550068 entries, 0 to 550067
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     550068 non-null  int64 
 1   Product_ID                  550068 non-null  object
 2   Gender                      550068 non-null  object
 3   Age                         550068 non-null  object
 4   Occupation                  550068 non-null  int64 
 5   City_Category               550068 non-null  object
 6   Stay_In_Current_City_Years  550068 non-null  object
 7   Marital_Status              550068 non-null  int64 
 8   Product_Category            550068 non-null  int64 
 9   Purchase                    550068 non-null  int64 
dtypes: int64(5), object(5)
memory usage: 42.0+ MB
```

Basic Analysis

- **Analysing metrics** - Basic metrics
  - Observations on **shape** of data
  - **Data types** of all the attributes

- **Conversion** of categorical attributes to 'category' (If required)
- **Structure & characteristics** of the dataset
- Statistical summary

In [235…

```
df.shape
```

Out[235]:

```
(550068, 10)
```

In [236…

```
df.describe()
```

Out[236]:

	User_ID	Occupation	Marital_Status	Product_Category	Purchase
count	5.500680e+05	550068.000000	550068.000000	550068.000000	550068.000000
mean	1.003029e+06	8.076707	0.409653	5.404270	9263.968713
std	1.727592e+03	6.522660	0.491770	3.936211	5023.065394
min	1.000001e+06	0.000000	0.000000	1.000000	12.000000
25%	1.001516e+06	2.000000	0.000000	1.000000	5823.000000
50%	1.003077e+06	7.000000	0.000000	5.000000	8047.000000
75%	1.004478e+06	14.000000	1.000000	8.000000	12054.000000
max	1.006040e+06	20.000000	1.000000	20.000000	23961.000000

In [237…

```
df.describe(include=object)
```

Out[237]:

	Product_ID	Gender	Age	City_Category	Stay_In_Current_City_Years
count	550068	550068	550068	550068	550068
unique	3631	2	7	3	5
top	P00265242	M	26-35	B	1
freq	1880	414259	219587	231173	193821

Non-Graphical Analysis

In [238…

```
df.head()
```

Out[238]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Proc
0	1000001	P00069042	F	0-17	10	A	2	0	
1	1000001	P00248942	F	0-17	10	A	2	0	
2	1000001	P00087842	F	0-17	10	A	2	0	
3	1000001	P00085442	F	0-17	10	A	2	0	
4	1000002	P00285442	M	55+	16	C	4+	0	

In [239…
```
df["Product_ID"].value_counts()
```

Out[239]:
```
P00265242    1880
P00025442    1615
P00110742    1612
P00112142    1562
P00057642    1470
             ...
P00314842       1
P00298842       1
P00231642       1
P00204442       1
P00066342       1
Name: Product_ID, Length: 3631, dtype: int64
```

In [240…
```
df["User_ID"].value_counts()
```

Out[240]:
```
1001680    1026
1004277     979
1001941     898
1001181     862
1000889     823
           ...
1002690       7
1002111       7
1005810       7
1004991       7
1000708       6
Name: User_ID, Length: 5891, dtype: int64
```

In [241…
```
df["Product_Category"].value_counts()
```

Out[241]:
```
5     150933
1     140378
8     113925
11     24287
2      23864
6      20466
3      20213
4      11753
16      9828
15      6290
13      5549
10      5125
12      3947
7       3721
18      3125
20      2550
19      1603
14      1523
17       578
9        410
Name: Product_Category, dtype: int64
```

In [242…
```
df["Gender"].value_counts()
```

Out[242]:
```
M    414259
F    135809
Name: Gender, dtype: int64
```

In [243…
```
df["Marital_Status"].value_counts()
```

Out[243]:
```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

In [244…
```
df["Age"].value_counts() # different generation categories
```

Out[244]:
```
26-35    219587
36-45    110013
18-25     99660
46-50     45701
51-55     38501
55+       21504
0-17      15102
Name: Age, dtype: int64
```

- **Data type conversion**

In [245…
```
df.User_ID = df.User_ID.astype(object)
df.Occupation = df.Occupation.astype(object)
df.Product_Category = df.Product_Category.astype(object)
```

In [246…
```
def marital_status_to_category(marital_status):
    if marital_status == 0:
        return "Un-Married"
    else:
        return "Married"
```

In [247…
```
df["Marital_Status"] = df["Marital_Status"].apply(marital_status_to_category)
```

In [248…
```
df["Marital_Status"].value_counts()
```

Out[248]:
```
Un-Married    324731
Married       225337
Name: Marital_Status, dtype: int64
```

In [249…
```
df["Stay_In_Current_City_Years"].value_counts()
```

Out[249]:
```
1     193821
2     101838
3      95285
4+     84726
0      74398
Name: Stay_In_Current_City_Years, dtype: int64
```

- **Feature types by data type**

In [250…
```
continious_features = df.select_dtypes(include=['int64','float64']).columns
continious_features
```

Out[250]:
```
Index(['Purchase'], dtype='object')
```

In [251…
```
categorical_features = df.select_dtypes(exclude=['int64','float64']).columns
categorical_features
```

Out[251]:
```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category'],
      dtype='object')
```

In [252…
```
def percentage_wise_features(feature_list):
    for feature_name in feature_list:
        features_stats = df[feature_name].value_counts(normalize=True)*100
        print(" <<< ",feature_name," >>> ")
        print()
        print(features_stats)
        print()
```

In [253…
```
percentage_wise_features(categorical_features)
```

```
 <<<  User_ID  >>>

1001680    0.186522
1004277    0.177978
1001941    0.163253
1001181    0.156708
1000889    0.149618
             ...
1002690    0.001273
1002111    0.001273
1005810    0.001273
1004991    0.001273
1000708    0.001091
Name: User_ID, Length: 5891, dtype: float64

 <<<  Product_ID  >>>

P00265242    0.341776
P00025442    0.293600
P00110742    0.293055
P00112142    0.283965
P00057642    0.267240
               ...
P00314842    0.000182
P00298842    0.000182
P00231642    0.000182
P00204442    0.000182
P00066342    0.000182
Name: Product_ID, Length: 3631, dtype: float64

 <<<  Gender  >>>

M    75.310507
F    24.689493
Name: Gender, dtype: float64

 <<<  Age  >>>

26-35    39.919974
36-45    19.999891
18-25    18.117760
46-50     8.308246
51-55     6.999316
55+       3.909335
0-17      2.745479
Name: Age, dtype: float64

 <<<  Occupation  >>>

4     13.145284
0     12.659889
7     10.750125
1      8.621843
17     7.279645
20     6.101427
12     5.668208
14     4.964659
2      4.833584
16     4.612339
6      3.700452
3      3.208694
10     2.350618
5      2.213726
15     2.211545
11     2.106285
19     1.538173
```

```
13     1.404917
18     1.203851
9      1.143677
8      0.281056
Name: Occupation, dtype: float64

 <<<  City_Category  >>>

B    42.026259
C    31.118880
A    26.854862
Name: City_Category, dtype: float64

 <<<  Stay_In_Current_City_Years  >>>

1     35.235825
2     18.513711
3     17.322404
4+    15.402823
0     13.525237
Name: Stay_In_Current_City_Years, dtype: float64

 <<<  Marital_Status  >>>

Un-Married    59.034701
Married       40.965299
Name: Marital_Status, dtype: float64

 <<<  Product_Category  >>>

5     27.438971
1     25.520118
8     20.711076
11     4.415272
2      4.338373
6      3.720631
3      3.674637
4      2.136645
16     1.786688
15     1.143495
13     1.008784
10     0.931703
12     0.717548
7      0.676462
18     0.568112
20     0.463579
19     0.291419
14     0.276875
17     0.105078
9      0.074536
Name: Product_Category, dtype: float64
```

In [254…

```
categorical_features
```

Out[254]:

```
Index(['User_ID', 'Product_ID', 'Gender', 'Age', 'Occupation', 'City_Category',
       'Stay_In_Current_City_Years', 'Marital_Status', 'Product_Category'],
      dtype='object')
```

Multi-feature Analysis (Marginal Probability)

- **Gender based**

In [255…

```python
pd.crosstab(df['Product_Category'], df['Gender'], margins=True,normalize=True)*100
```

Out[255]:

Gender Product_Category	F	M	All
1	4.514169	21.005948	25.520118
2	1.028600	3.309773	4.338373
3	1.091865	2.582772	3.674637
4	0.661555	1.475090	2.136645
5	7.628330	19.810642	27.438971
6	0.828807	2.891824	3.720631
7	0.171433	0.505028	0.676462
8	6.100700	14.610375	20.711076
9	0.012726	0.061811	0.074536
10	0.211247	0.720456	0.931703
11	0.861530	3.553742	4.415272
12	0.278511	0.439037	0.717548
13	0.265785	0.742999	1.008784
14	0.113259	0.163616	0.276875
15	0.190158	0.953337	1.143495
16	0.436673	1.350015	1.786688
17	0.011271	0.093807	0.105078
18	0.069446	0.498666	0.568112
19	0.081990	0.209429	0.291419
20	0.131438	0.332141	0.463579
All	24.689493	75.310507	100.000000

- **Age based**

In [256…

```python
pd.crosstab(df['Product_Category'], df['Age'], margins=True,normalize=True)*100
```

Out[256]:

Age	0-17	18-25	26-35	36-45	46-50	51-55	55+	All
Product_Category								
1	0.651738	4.901576	10.589418	5.026288	1.904128	1.645069	0.801901	25.520118
2	0.146346	0.804991	1.623072	0.892981	0.382680	0.323778	0.164525	4.338373
3	0.218155	0.856258	1.392919	0.700641	0.250151	0.167979	0.088535	3.674637
4	0.137801	0.447763	0.762088	0.427947	0.179978	0.123257	0.057811	2.136645
5	0.787175	5.185177	11.175527	5.340612	2.176276	1.798505	0.975698	27.438971
6	0.072536	0.681552	1.542537	0.708821	0.294873	0.263604	0.156708	3.720631
7	0.009635	0.087444	0.300145	0.147073	0.059447	0.048358	0.024361	0.676462
8	0.410495	3.256143	8.045551	4.235113	1.937215	1.697972	1.128588	20.711076
9	0.002909	0.011453	0.027997	0.019452	0.005999	0.005272	0.001454	0.074536
10	0.020179	0.109623	0.324869	0.224518	0.094534	0.094352	0.063628	0.931703
11	0.134529	0.835715	1.795051	0.900434	0.382498	0.265058	0.101987	4.415272
12	0.022724	0.079808	0.199248	0.180705	0.094534	0.078718	0.061811	0.717548
13	0.020361	0.137438	0.381044	0.227245	0.100169	0.087807	0.054721	1.008784
14	0.007090	0.041813	0.102533	0.056720	0.027088	0.027997	0.013635	0.276875
15	0.029087	0.186159	0.431219	0.253605	0.109441	0.092352	0.041631	1.143495
16	0.041631	0.290510	0.748635	0.355411	0.159798	0.122167	0.068537	1.786688
17	0.001091	0.007454	0.023088	0.024542	0.017271	0.019452	0.012180	0.105078
18	0.004908	0.061629	0.189431	0.127621	0.063810	0.076900	0.043813	0.568112
19	0.010726	0.049994	0.102351	0.058175	0.027088	0.024361	0.018725	0.291419
20	0.016362	0.085262	0.163253	0.091989	0.041268	0.036359	0.029087	0.463579
All	2.745479	18.117760	39.919974	19.999891	8.308246	6.999316	3.909335	100.000000

- **Occupation based**

In [257…

```
pd.crosstab(df['Product_Category'], df['Occupation'], margins=True,normalize=True)*100
```

Out[257]:

Occupation	0	1	2	3	4	5	6	7	8	
Product_Category										
1	3.207422	1.875223	1.043871	0.717002	3.497386	0.662827	0.836442	2.915458	0.093079	0.2
2	0.540115	0.349411	0.201793	0.121076	0.552477	0.107078	0.149254	0.456489	0.017998	0.0
3	0.479032	0.283420	0.174160	0.111623	0.601017	0.081263	0.136347	0.291600	0.012180	0.0
4	0.269239	0.176887	0.093079	0.072355	0.309598	0.045813	0.080899	0.199793	0.007090	0.0
5	3.451391	2.394068	1.384011	0.960972	3.719358	0.595563	1.034599	2.839467	0.067446	0.3
6	0.466851	0.321233	0.188886	0.118349	0.483031	0.068901	0.133620	0.418676	0.008181	0.0
7	0.101624	0.067992	0.041631	0.024906	0.081626	0.006726	0.026542	0.059447	0.000909	0.0
8	2.560956	2.181185	1.139132	0.698641	2.514235	0.350684	0.869892	2.294262	0.047812	0.2
9	0.008908	0.004727	0.002182	0.002363	0.009272	0.001636	0.002909	0.007090	0.000182	0.0
10	0.115622	0.093625	0.051448	0.035450	0.086171	0.015453	0.034359	0.095443	0.002363	0.0
11	0.682097	0.305599	0.209247	0.125984	0.580292	0.143800	0.155435	0.438491	0.011453	0.0
12	0.080899	0.078899	0.044176	0.032541	0.067628	0.014180	0.029269	0.084535	0.000545	0.0
13	0.130893	0.100715	0.044904	0.039995	0.107441	0.017452	0.037268	0.110532	0.001454	0.0
14	0.035632	0.030723	0.014725	0.012362	0.032723	0.004181	0.013089	0.029087	0.000000	0.0
15	0.130529	0.096170	0.055266	0.031087	0.135256	0.026179	0.035632	0.114895	0.004363	0.0
16	0.222700	0.142528	0.081444	0.062174	0.216700	0.038722	0.075627	0.212337	0.002909	0.0
17	0.011453	0.011090	0.002545	0.002727	0.008363	0.001636	0.003272	0.015089	0.000182	0.0
18	0.077445	0.043267	0.027451	0.018361	0.048721	0.017271	0.016543	0.083808	0.001273	0.0
19	0.034178	0.025270	0.011271	0.008908	0.032360	0.004363	0.011453	0.034905	0.000909	0.0
20	0.052903	0.039813	0.022361	0.011817	0.061629	0.009999	0.017998	0.048721	0.000727	0.0
All	12.659889	8.621843	4.833584	3.208694	13.145284	2.213726	3.700452	10.750125	0.281056	1.1

21 rows × 22 columns

- **Marital_Status based**

In [258…

```
pd.crosstab(df['Product_Category'], df['Marital_Status'], margins=True,normalize=True)*1
```

Out[258]:

Marital_Status	Married	Un-Married	All
Product_Category			
1	10.181105	15.339013	25.520118
2	1.768145	2.570228	4.338373
3	1.427823	2.246813	3.674637
4	0.831897	1.304748	2.136645
5	11.139895	16.299076	27.438971
6	1.513813	2.206818	3.720631
7	0.305599	0.370863	0.676462
8	8.819637	11.891439	20.711076
9	0.029633	0.044904	0.074536
10	0.426675	0.505028	0.931703
11	1.748693	2.666579	4.415272
12	0.347775	0.369772	0.717548
13	0.433946	0.574838	1.008784
14	0.123076	0.153799	0.276875
15	0.484849	0.658646	1.143495
16	0.748089	1.038599	1.786688
17	0.050903	0.054175	0.105078
18	0.269785	0.298327	0.568112
19	0.119440	0.171979	0.291419
20	0.194521	0.269058	0.463579
All	40.965299	59.034701	100.000000

- **City_Category based**

In [259…

```
pd.crosstab(df['Product_Category'], df['City_Category'], margins=True,normalize=True)*1(
```

Out[259]:

City_Category Product_Category	A	B	C	All
1	6.377575	10.590145	8.552397	25.520118
2	1.116407	1.898674	1.323291	4.338373
3	0.898616	1.561080	1.214941	3.674637
4	0.554477	0.950064	0.632104	2.136645
5	7.673779	11.660013	8.105180	27.438971
6	1.001149	1.549990	1.169492	3.720631
7	0.222882	0.290691	0.162889	0.676462
8	5.850004	8.644931	6.216141	20.711076
9	0.019998	0.031632	0.022906	0.074536
10	0.242334	0.375045	0.314325	0.931703
11	1.200033	1.906128	1.309111	4.415272
12	0.193249	0.304508	0.219791	0.717548
13	0.293418	0.412858	0.302508	1.008784
14	0.087444	0.114895	0.074536	0.276875
15	0.312143	0.479577	0.351775	1.143495
16	0.517754	0.734091	0.534843	1.786688
17	0.021997	0.048539	0.034541	0.105078
18	0.136892	0.252514	0.178705	0.568112
19	0.049630	0.083990	0.157799	0.291419
20	0.085080	0.136892	0.241606	0.463579
All	26.854862	42.026259	31.118880	100.000000

- **Stay_In_Current_City_Years based**

In [260…

```
pd.crosstab(df['Product_Category'], df['Stay_In_Current_City_Years'], margins=True,norma
```

Out[260]:

Stay_In_Current_City_Years	0	1	2	3	4+	All
Product_Category						
1	3.373219	8.875085	4.819950	4.541439	3.910426	25.520118
2	0.581746	1.523266	0.838078	0.768996	0.626286	4.338373
3	0.507028	1.274024	0.711185	0.651738	0.530662	3.674637
4	0.288510	0.745181	0.387952	0.382135	0.332868	2.136645
5	3.744992	9.653716	5.109550	4.794316	4.136398	27.438971
6	0.497575	1.320928	0.678643	0.625559	0.597926	3.720631
7	0.101442	0.234698	0.109259	0.125439	0.105623	0.676462
8	2.788928	7.481620	3.679181	3.467571	3.293775	20.711076
9	0.011271	0.023997	0.015816	0.013998	0.009453	0.074536
10	0.120894	0.336867	0.165071	0.165979	0.142891	0.931703
11	0.633558	1.480726	0.837351	0.727910	0.735727	4.415272
12	0.097806	0.255968	0.133256	0.123439	0.107078	0.717548
13	0.140892	0.364682	0.177978	0.171433	0.153799	1.008784
14	0.039268	0.104533	0.045267	0.042904	0.044904	0.276875
15	0.166525	0.398133	0.210156	0.191249	0.177433	1.143495
16	0.239970	0.641739	0.327959	0.295236	0.281783	1.786688
17	0.015634	0.030542	0.020907	0.020907	0.017089	0.105078
18	0.077445	0.219609	0.095625	0.088898	0.086535	0.568112
19	0.036177	0.110350	0.053812	0.046176	0.044904	0.291419
20	0.062356	0.160162	0.096715	0.077081	0.067264	0.463579
All	13.525237	35.235825	18.513711	17.322404	15.402823	100.000000

Multi-feature Analysis (Conditional Probability)

In [261…

```
pd.crosstab(index=df['Gender'], columns=df['Product_Category'], margins=True, normalize=
```

Out[261]:

Product_Category	1	2	3	4	5	6	7	8	9	
Gender										
F	0.182838	0.041661	0.044224	0.026795	0.308971	0.033569	0.006944	0.247097	0.000515	0.0085
M	0.278925	0.043948	0.034295	0.019587	0.263053	0.038399	0.006706	0.194002	0.000821	0.0095
All	0.255201	0.043384	0.036746	0.021366	0.274390	0.037206	0.006765	0.207111	0.000745	0.0093

In [262…

```
pd.crosstab(index=df['Age'], columns=df['Product_Category'], margins=True, normalize='ir
```

Out[262]:

Product_Category	1	2	3	4	5	6	7	8	9	
Age										
0-17	0.237386	0.053304	0.079460	0.050192	0.286717	0.026420	0.003509	0.149517	0.001059	0.0073
18-25	0.270540	0.044431	0.047261	0.024714	0.286193	0.037618	0.004826	0.179721	0.000632	0.0060
26-35	0.265266	0.040658	0.034893	0.019090	0.279948	0.038641	0.007519	0.201542	0.000701	0.0081
36-45	0.251316	0.044649	0.035032	0.021397	0.267032	0.035441	0.007354	0.211757	0.000973	0.0112
46-50	0.229185	0.046060	0.030109	0.021663	0.261942	0.035492	0.007155	0.233168	0.000722	0.0113
51-55	0.235033	0.046259	0.023999	0.017610	0.256954	0.037661	0.006909	0.242591	0.000753	0.0134
55+	0.205125	0.042085	0.022647	0.014788	0.249581	0.040086	0.006231	0.288690	0.000372	0.0162
All	0.255201	0.043384	0.036746	0.021366	0.274390	0.037206	0.006765	0.207111	0.000745	0.0093

In [263…

```
pd.crosstab(index=df['Marital_Status'], columns=df['Product_Category'], margins=True, no
```

Out[263]:

Product_Category	1	2	3	4	5	6	7	8	9	
Marital_Status										
Married	0.248530	0.043162	0.034854	0.020307	0.271935	0.036954	0.007460	0.215295	0.000723	0.0104
Un-Married	0.259830	0.043538	0.038059	0.022101	0.276093	0.037382	0.006282	0.201431	0.000761	0.0085
All	0.255201	0.043384	0.036746	0.021366	0.274390	0.037206	0.006765	0.207111	0.000745	0.0093

In [264…

```
pd.crosstab(index=df['City_Category'], columns=df['Product_Category'], margins=True, nor
```

Out[264]:

Product_Category	1	2	3	4	5	6	7	8	9	
City_Category										
A	0.237483	0.041572	0.033462	0.020647	0.285750	0.037280	0.008299	0.217838	0.000745	0.0090
B	0.251989	0.045178	0.037145	0.022606	0.277446	0.036881	0.006917	0.205703	0.000753	0.0089
C	0.274830	0.042524	0.039042	0.020313	0.260459	0.037581	0.005234	0.199755	0.000736	0.0101
All	0.255201	0.043384	0.036746	0.021366	0.274390	0.037206	0.006765	0.207111	0.000745	0.0093

In [265…

```
pd.crosstab(index=df['Stay_In_Current_City_Years'], columns=df['Product_Category'], marg
```

Out[265]:

Product_Category	1	2	3	4	5	6	7	8	9
Stay_In_Current_City_Years									
0	0.249402	0.043012	0.037488	0.021331	0.276889	0.036789	0.007500	0.206202	0.000833
1	0.251877	0.043231	0.036157	0.021148	0.273974	0.037488	0.006661	0.212330	0.000681
2	0.260345	0.045268	0.038414	0.020955	0.275987	0.036656	0.005902	0.198727	0.000854
3	0.262171	0.044393	0.037624	0.022060	0.276770	0.036113	0.007241	0.200178	0.000808
4+	0.253877	0.040660	0.034452	0.021611	0.268548	0.038819	0.006857	0.213842	0.000614
All	0.255201	0.043384	0.036746	0.021366	0.274390	0.037206	0.006765	0.207111	0.000745

- **Observation from Non-graphical Analysis**
  - **Gender** - **1, 5 , 8 product categories** are more likely to be purchased by **males than females**
  - **Age** - **1,5 product categories** are more **frequently** being **purchased** in **age group 26-35**
  - **Marital_Status** - **1, 5 , 8 product categories** are more popular purchase among **un-married** users
  - **City Category**
    - City **A** : **1, 5 , 8 product categories** are popular
    - City **B** : **1, 5 , 8 product categories** are popular
    - City **C** : **1, 5 , 8 product categories** are popular
  - **Stay_In_Current_City_Years** - **1, 5 , 8 product categories** are popular

Missing Value & Outlier Detection

- **Missing value detection**

In [266…

```
percent_missing = df.isnull().sum() * 100 / len(df)
missing_value_df = pd.DataFrame({'column_name': df.columns,
                                  'percent_missing': percent_missing})
missing_value_df.sort_values('percent_missing', ascending=False)
```

Out[266]:

	column_name	percent_missing
User_ID	User_ID	0.0
Product_ID	Product_ID	0.0
Gender	Gender	0.0
Age	Age	0.0
Occupation	Occupation	0.0
City_Category	City_Category	0.0
Stay_In_Current_City_Years	Stay_In_Current_City_Years	0.0
Marital_Status	Marital_Status	0.0
Product_Category	Product_Category	0.0
Purchase	Purchase	0.0

- **Outlier detection**

In [267…

```
def find_outliers_IQR(column_name):
    print("Outliers by feature name --> ",column_name)
    Q1=df[column_name].quantile(0.25)
    Q3=df[column_name].quantile(0.75)

    IQR=Q3-Q1
    lower = Q1 - 1.5*IQR
    upper = Q3 + 1.5*IQR

    outliers = df[((df[column_name]<lower) | (df[column_name]>upper))]

    return outliers
```

In [268…

```
outlier_df_by_purchase = find_outliers_IQR("Purchase")
outlier_df_by_purchase
```

```
Outliers by feature name -->  Purchase
```

Out[268]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
343	1000058	P00117642	M	26-35	2	B	3	Un-Married
375	1000062	P00119342	F	36-45	3	A	1	Un-Married
652	1000126	P00087042	M	18-25	9	B	1	Un-Married
736	1000139	P00159542	F	26-35	20	C	2	Un-Married
1041	1000175	P00052842	F	26-35	2	B	1	Un-Married
...	...	...	...	...	...	...	...	...
544488	1005815	P00116142	M	26-35	20	B	1	Un-Married
544704	1005847	P00085342	F	18-25	4	B	2	Un-Married
544743	1005852	P00202242	F	26-35	1	A	0	Married
545663	1006002	P00116142	M	51-55	0	C	1	Married
545787	1006018	P00052842	M	36-45	1	C	3	Un-Married

2677 rows × 10 columns

In [269…

```
outlier_df_by_occupation = find_outliers_IQR("Occupation")
outlier_df_by_occupation
```

```
Outliers by feature name -->  Occupation
```

Out[269]:

User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Produ

In [270…

```
outlier_df_by_Product_Category = find_outliers_IQR("Product_Category")
outlier_df_by_Product_Category
```

```
Outliers by feature name -->  Product_Category
```

Out[270]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
545915	1000001	P00375436	F	0-17	10	A	2	Un-Married
545916	1000002	P00372445	M	55+	16	C	4+	Un-Married
545917	1000004	P00375436	M	46-50	7	B	2	Married
545918	1000006	P00375436	F	51-55	9	A	1	Un-Married
545919	1000007	P00372445	M	36-45	1	B	1	Married
...	...	...	...	...	...	...	...	...
550063	1006033	P00372445	M	51-55	13	B	1	Married
550064	1006035	P00375436	F	26-35	1	C	3	Un-Married
550065	1006036	P00375436	F	26-35	15	B	4+	Married
550066	1006038	P00375436	F	55+	1	C	2	Un-Married
550067	1006039	P00371644	F	46-50	0	B	4+	Married

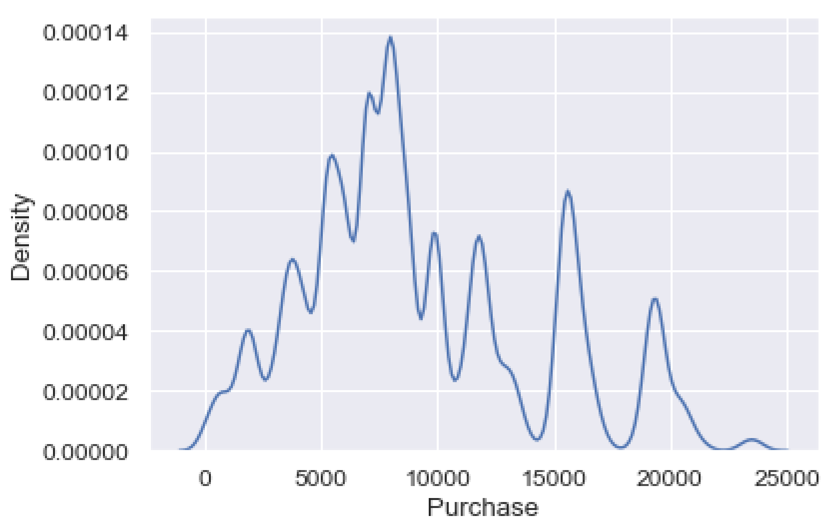
4153 rows × 10 columns

Visual Analysis

In [271…

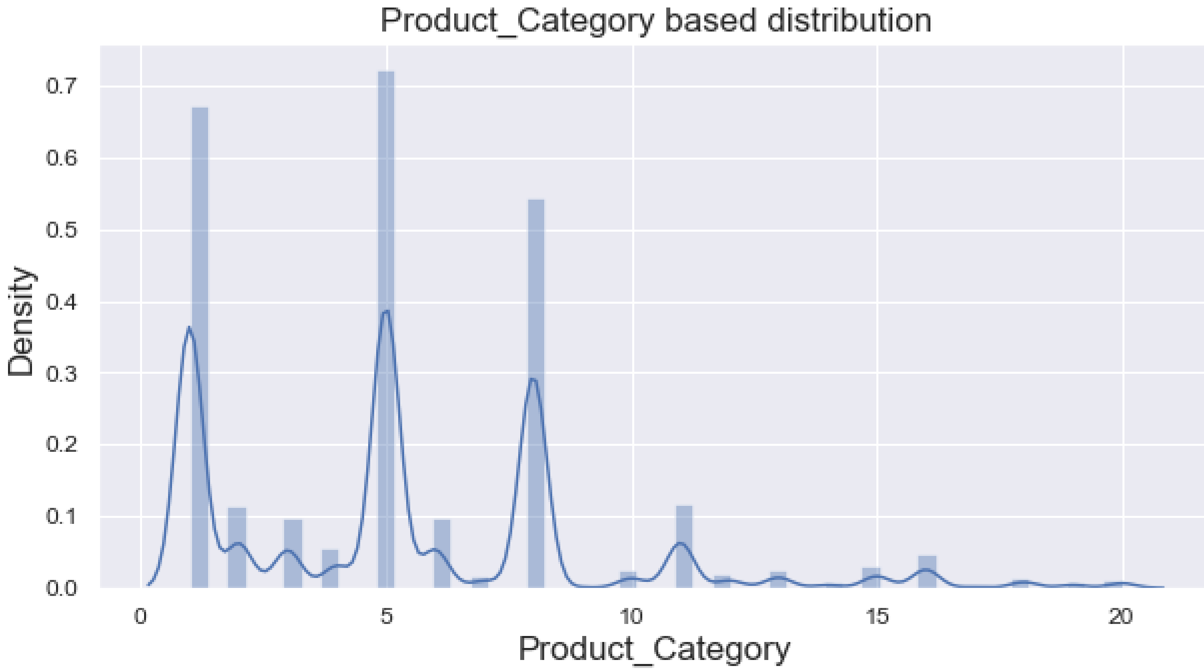
```
sns.kdeplot(data=df, x="Purchase")
```

Out[271]: <AxesSubplot:xlabel='Purchase', ylabel='Density'>



In [272…

```
sns.set(font_scale = 1.1)
plt.figure(figsize=(10,5))
plt.xlabel("Gender",fontsize=17)
plt.ylabel("Density",fontsize=17)
sns.distplot(df["Product_Category"])
plt.title("Product_Category based distribution",fontdict ={"fontsize": 17})
plt.show()
```



Puchase Analysis

In [273…

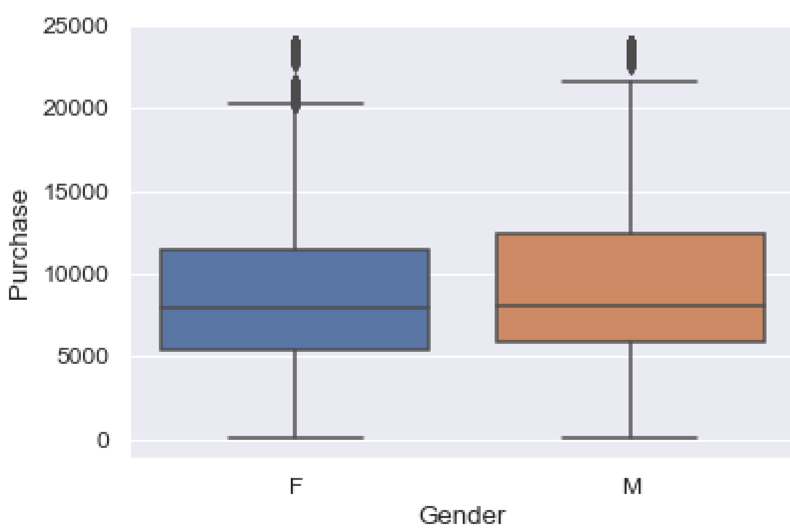
```
selected_features = ['Gender', 'Age', 'Occupation', 'City_Category','Stay_In_Current_Cit
```

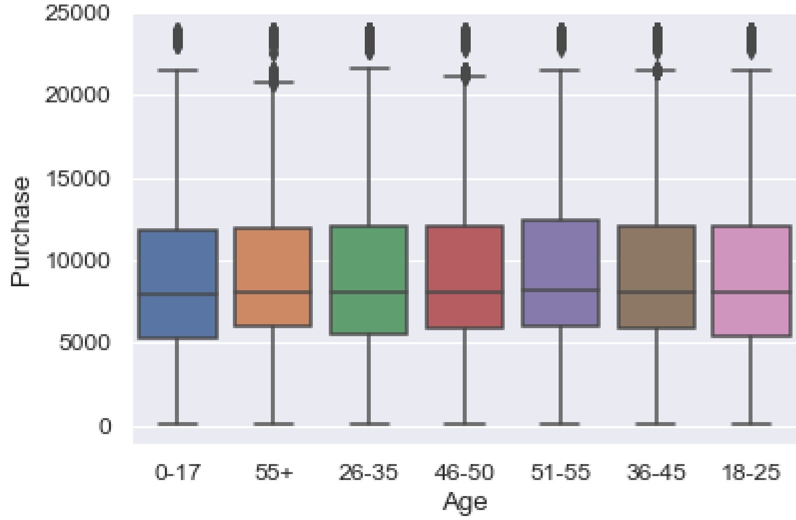
In [274…

```
def boxplot_all_catogorical_vs_all_continious_features(feature_y_catg_list,feature_y_con
    plt.figure()
    for categorical_feature in feature_y_catg_list:
        for continious_feature in feature_y_contn_list:
            sns.boxplot(x = categorical_feature, y=continious_feature, data=df)
            plt.show()
```

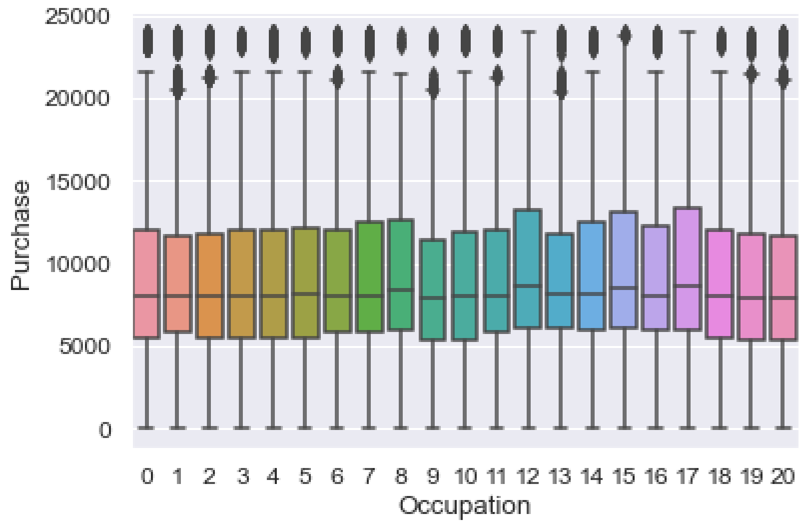
In [275…

```
boxplot_all_catogorical_vs_all_continious_features(selected_features,continious_features
```

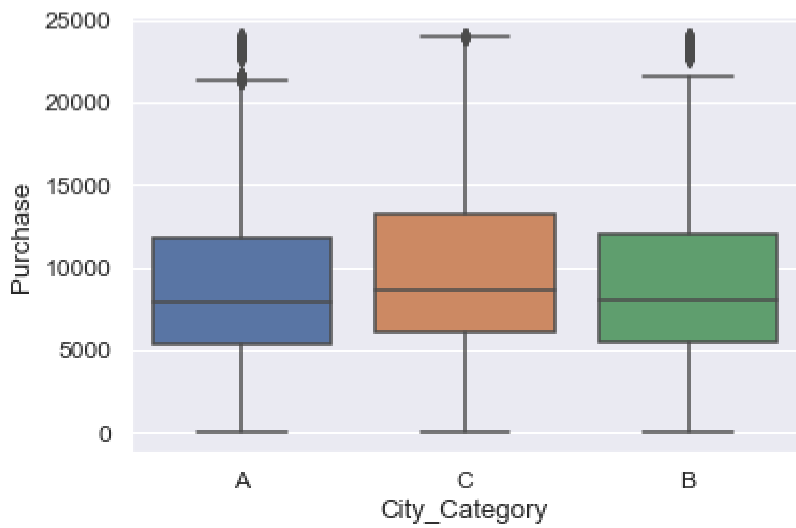




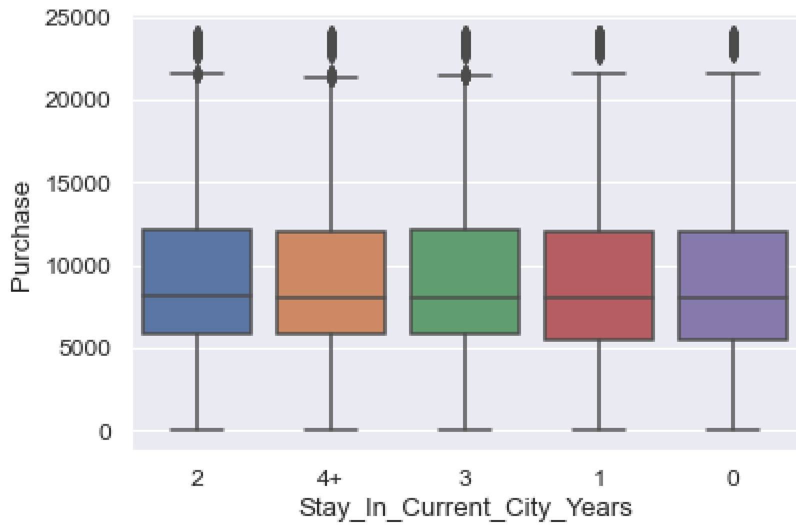
Purchase
25000
20000
15000
10000
5000
0
0-17
55+
26-35
46-50
51-55
36-45
18-25
Age

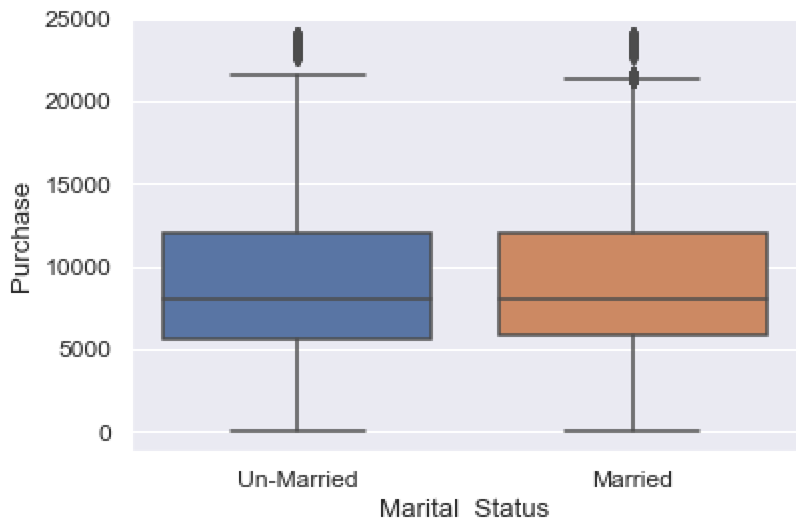


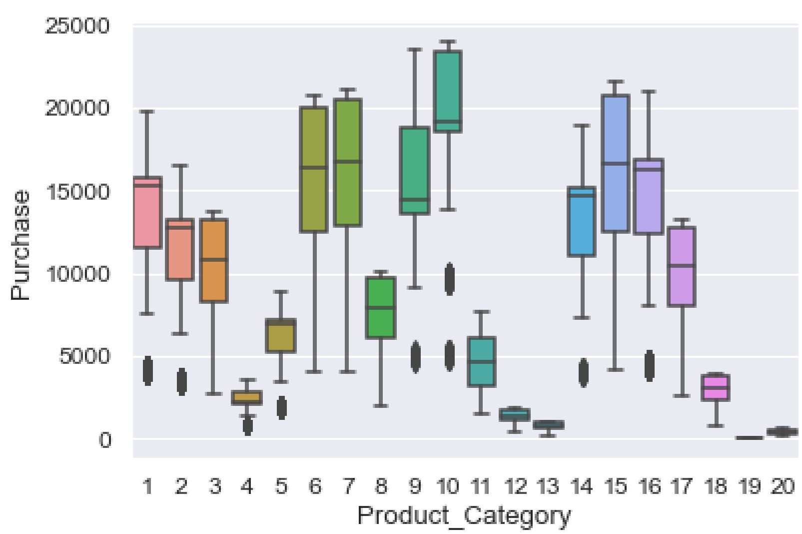
Purchase
25000
20000
15000
10000
5000
0
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Occupation



Purchase
25000
20000
15000
10000
5000
0
A
C
B
City_Category







- **Observation from Visual Analysis**
  - Except in "City C" , **overall purchase pattern almost similar across 'Gender', 'Age', 'Stay_In_Current_City_Years', 'Marital_Status'**
  - There is **varied purchase pattern** across **'Occupation','Product_Category'**
  - **Total purchase** for **product category 6,7,9,10,15,16 are higher** than other product categories

- **Pair plot**

In [276…

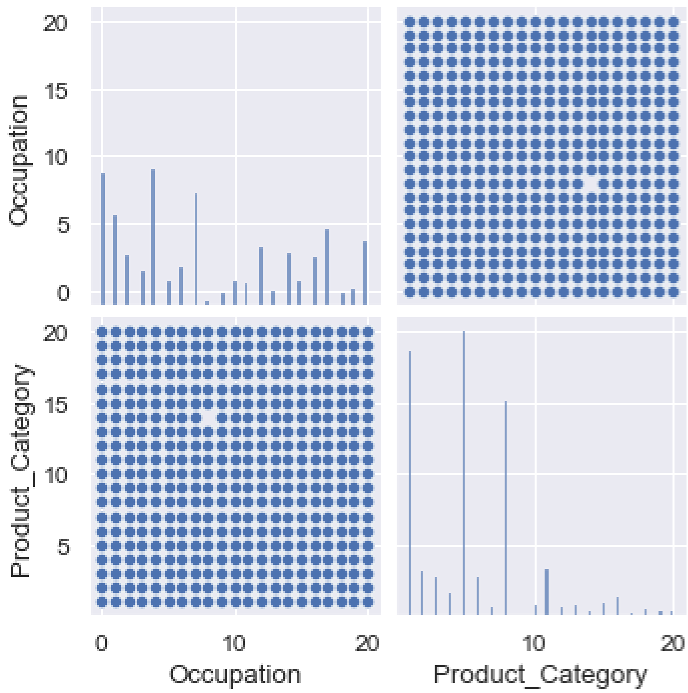
```
selected_features = ['Gender', 'Age', 'Occupation', 'City_Category','Stay_In_Current_Cit
```

In [277…

```
sns.pairplot(data=df[selected_features])
plt.plot()
```

Out[277]:

```
[]
```



In [278…

```
df_product_category_1_5_8 = df[(df["Product_Category"] == 1) | (df["Product_Category"] =
df_product_category_1 = df[df["Product_Category"] == 1]
df_product_category_5 = df[df["Product_Category"] == 5]
df_product_category_8 = df[df["Product_Category"] == 8]
```

In [279…

```
df_product_category_1_5_8["Product_Category"].value_counts()
```

Out[279]:

```
5    150933
1    140378
8    113925
Name: Product_Category, dtype: int64
```

In [280…

```
df_product_category_1 = df[df["Product_Category"] == 1]
df_product_category_5 = df[df["Product_Category"] == 5]
df_product_category_8 = df[df["Product_Category"] == 8]
```

Product_Category Analysis

In [281…

```
df_selected = df_product_category_1_5_8[selected_features]
```

In [282…

```
def show_values_on_bars(axs, h_v="v", space=1):
    total = float(len(df_selected))
    def _show_on_single_plot(ax):
        if h_v == "v":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() / 2
                _y = p.get_y() + p.get_height()
                value='{:.1f}%'.format(100 * p.get_height()/total)
                ax.text(_x, _y, value, ha="center")
```

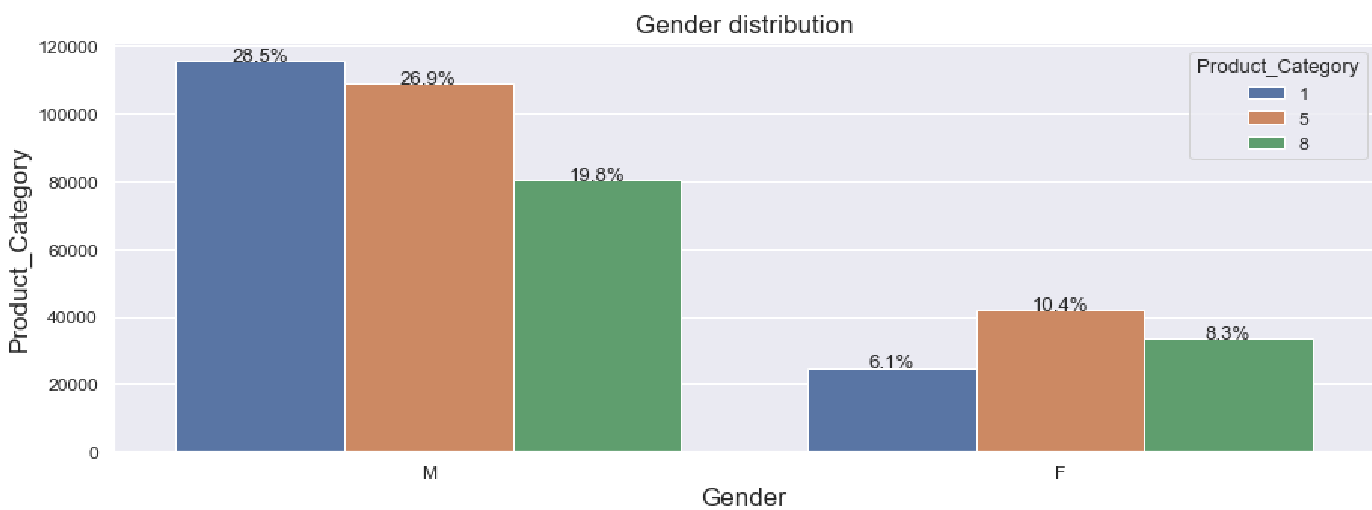
```
        elif h_v == "h":
            for p in ax.patches:
                _x = p.get_x() + p.get_width() + float(space)
                _y = p.get_y() + p.get_height()
                value='{:.1f}%'.format(100 * p.get_width()/total)
                ax.text(_x, _y, value, ha="left")
    if isinstance(axs, np.ndarray):
         for idx, ax in np.ndenumerate(axs):
            _show_on_single_plot(ax)
    else:
            _show_on_single_plot(axs)
```

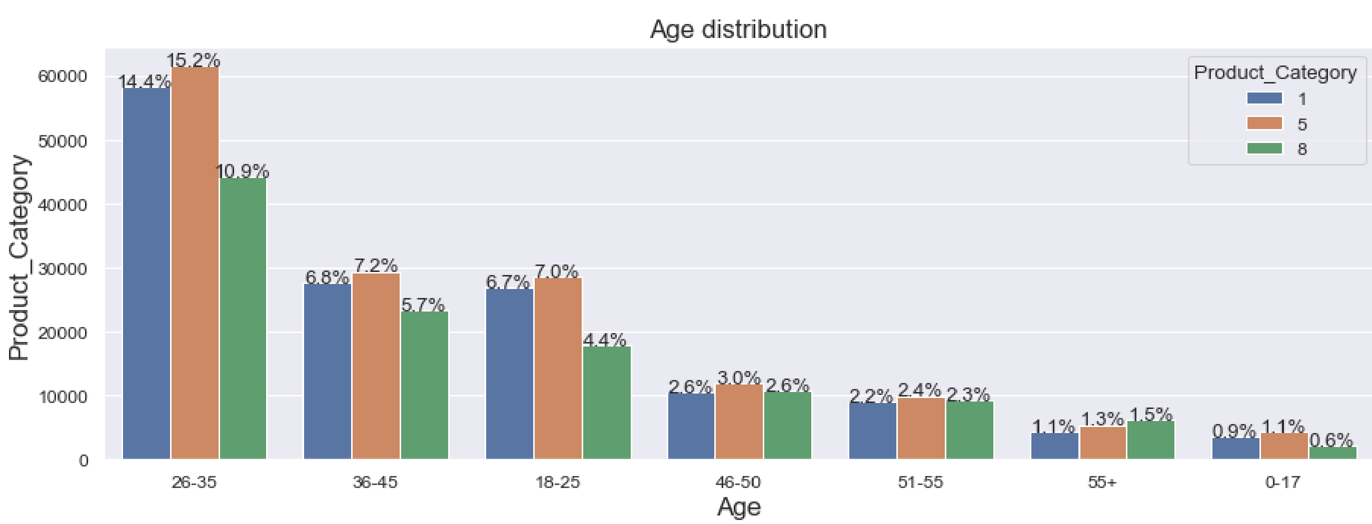
In [283…

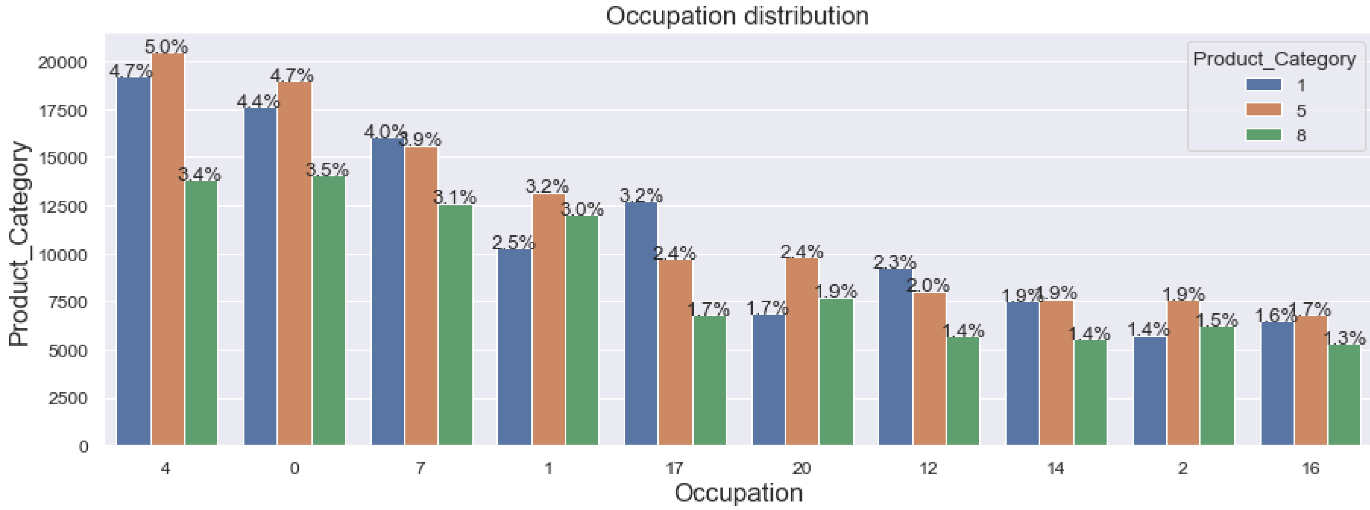
```
def plot_products_by_feature(feature_list):
    for feature_name in feature_list:
        sns.set(style="whitegrid")
        sns.set(font_scale = 1.1)
        plt.figure(figsize=(15,5))
        ax = sns.countplot(x=feature_name,data=df_selected,hue="Product_Category",order=
        plt.xlabel(feature_name, fontsize=17)
        plt.ylabel("Product_Category",fontsize=17)
        plt.title(feature_name +" distribution",fontdict ={"fontsize": 17})
        show_values_on_bars(ax,h_v="v",space=1)
```

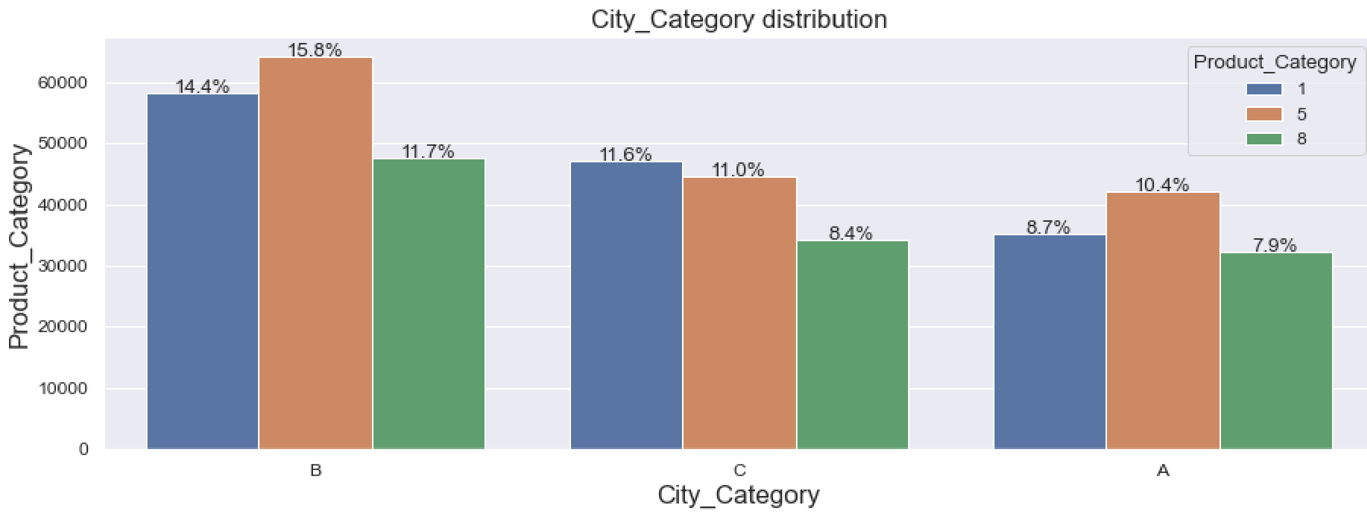
In [284…

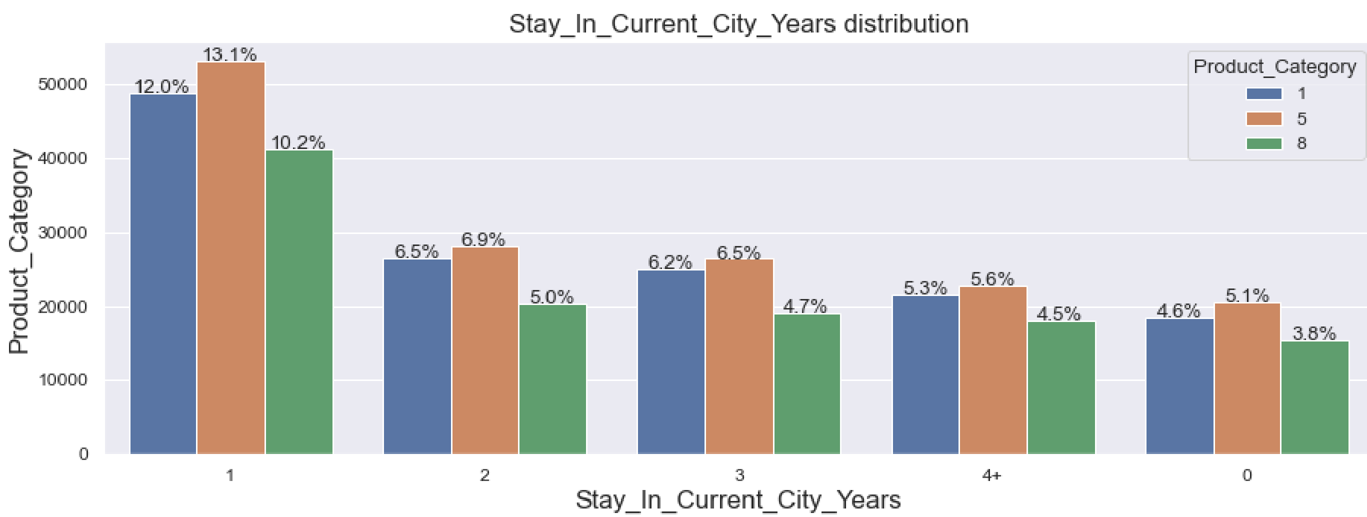
```
plot_products_by_feature(['Gender', 'Age', 'Occupation', 'City_Category','Stay_In_Currer
```

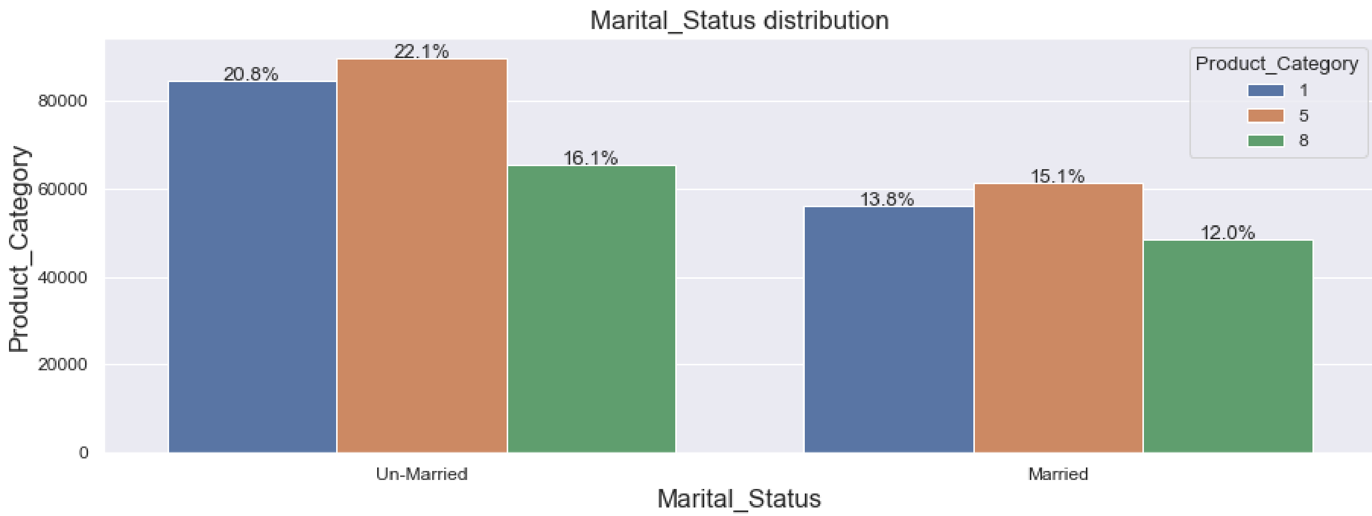





Occupation distribution
Product_Category
1
5
8
4.7%
5.0%
3.4%
4.4%
4.7%
3.5%
4.0%
3.9%
3.1%
2.5%
3.2%
3.0%
3.2%
2.4%
1.7%
1.7%
2.4%
1.9%
2.3%
2.0%
1.4%
1.9%
1.9%
1.4%
1.4%
1.9%
1.5%
1.6%
1.7%
1.3%
0
2500
5000
7500
10000
12500
15000
17500
20000
4
0
7
1
17
20
12
14
2
16
Occupation


City_Category distribution
Product_Category
1
5
8
14.4%
15.8%
11.7%
11.6%
11.0%
8.4%
8.7%
10.4%
7.9%
0
10000
20000
30000
40000
50000
60000
B
C
A
City_Category


Stay_In_Current_City_Years distribution
Product_Category
1
5
8
12.0%
13.1%
10.2%
6.5%
6.9%
5.0%
6.2%
6.5%
4.7%
5.3%
5.6%
4.5%
4.6%
5.1%
3.8%
0
10000
20000
30000
40000
50000
1
2
3
4+
0
Stay_In_Current_City_Years



- **Outlier treatment** [**Removing outliers** just to ensure sample mean distribution is not impacted by outlier while using CLT]

In [285…

```
df.drop(outlier_df_by_purchase.index,inplace=True)
df.drop(outlier_df_by_Product_Category.index,inplace=True)
```

- **Re-checking outliers post Outlier removal**

In [286…

```
outlier_df_by_purchase = find_outliers_IQR("Purchase")
outlier_df_by_purchase
```

```
Outliers by feature name -->  Purchase
```

Out[286]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status
5846	1000949	P00111042	M	51-55	17	B	4+	Married
10481	1001611	P00111742	M	26-35	0	B	1	Un-Married
11148	1001682	P00111742	M	26-35	2	A	3	Married
12497	1001884	P00111742	M	46-50	20	B	1	Married
14484	1002147	P00174242	F	18-25	0	B	3	Un-Married
...	...	...	...	...	...	...	...	...
534978	1004354	P00111742	M	36-45	14	B	2	Un-Married
541307	1005359	P00174242	M	18-25	16	B	3	Un-Married
542825	1005580	P00071442	M	46-50	7	B	4+	Un-Married
545664	1006002	P00071442	M	51-55	0	C	1	Married
545864	1006036	P00111042	F	26-35	15	B	4+	Married

165 rows × 10 columns

In [287…

```
outlier_df_by_occupation = find_outliers_IQR("Occupation")
outlier_df_by_occupation
```

```
Outliers by feature name -->  Occupation
```

Out[287]:

User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Produ

In [288…

```
outlier_df_by_Product_Category = find_outliers_IQR("Product_Category")
outlier_df_by_Product_Category
```

```
Outliers by feature name -->  Product_Category
```

Out[288]:

User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Years	Marital_Status	Produ

Confidence Interval (CI)

In [295…

```
def getConfidenceIntervalByFeatureValue(feature_x_name,feature_x_value,feature_y,bootstr
    if dataset_catagory == "Product_Category_All":
        data = df
    else:
        data = df_product_category_1_5_8

    # Configure bootstrap
    transactions = data[data[feature_x_name] == feature_x_value][feature_y]

    sample_size = 10000
    bootstrap_size = int(bootstrap_repetition_factor *len(transactions))
```

```
    bootstrapped_means = np.empty(sample_size)
    for i in range(sample_size):
        bootstrapped_sample = transactions.sample(bootstrap_size, replace=True)
        x_bar = np.mean(bootstrapped_sample) # Sample mean; Replace by median/percentile
        bootstrapped_means[i] = x_bar

    print(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

    # Mean & Standard deviation
    mean = np.mean(bootstrapped_means)
    standard_deviation = np.std(bootstrapped_means,ddof = 1)      # ddof = 1 for un-biase
    print("CI by "+feature_x_name+" == "+feature_x_value+" Mean ",mean," S.D - ",standar

    # Plot mean scores
    plt.figure()
    plt.hist(bootstrapped_means, bins=100)
    plt.grid()
    plt.show()

    # Compute Confidence Intervals

    # Strategy #1 - Bootstrap with CLT, which works on only mean
    # given that bootstrapped_means that follows Gaussian distribution: CLT
    confidence_interval_on_mean = [(mean-1.96*standard_deviation),(mean+1.96*standard_de
    print("C.I (on the mean)",confidence_interval_on_mean)

    # Strategy #2 - Pure Bootstrap , as CLT won't work for percentile
    confidence_interval_on_percentile = np.percentile(bootstrapped_means, [2.5, 97.5])
    print("C.I (on the percentile) by",confidence_interval_on_percentile)

    CI_length = confidence_interval_on_mean[1] - confidence_interval_on_mean[0]
    print("C.I length ",CI_length)
    print(">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

In [296…

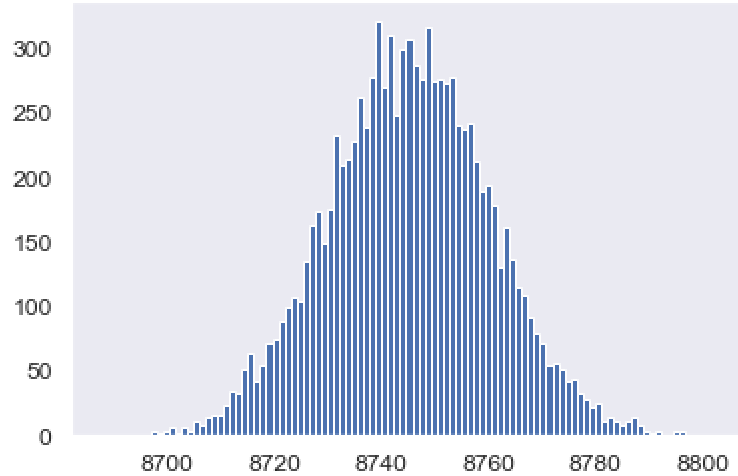
```
def getConfidenceIntervalByFeature(feature_x,feature_y,bootstrap_repetition_factor,datas
    if dataset_catagory == "Product_Category_All":
        data = df
    else:
        data = df_product_category_1_5_8

    levels =  data[feature_x].unique()
    for level in levels:
        getConfidenceIntervalByFeatureValue(feature_x,level,feature_y,bootstrap_repetit
```

Gender (Confidence Interval)

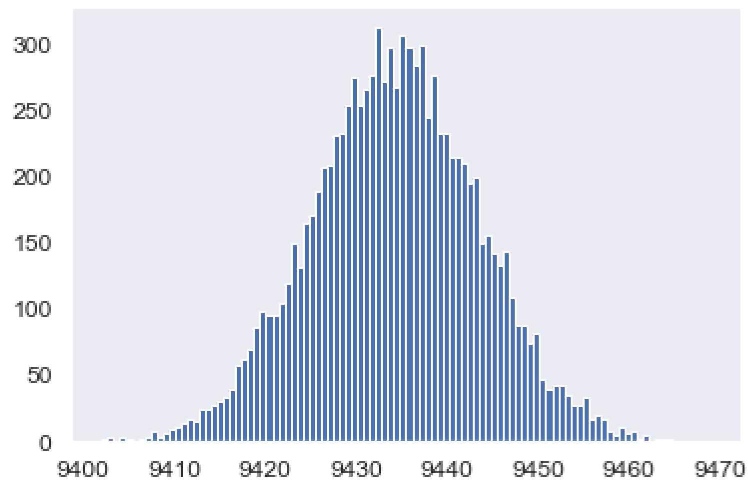
In [297…

```
getConfidenceIntervalByFeature("Gender","Purchase",0.7,"Product_Category_All")
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Gender == F Mean  8745.09616804928  S.D -  15.159546487015946
```




```
C.I (on the mean) [8715.38345693473, 8774.808879163831]
C.I (on the percentile) by [8715.31521336 8775.29461271]
C.I length  59.42542222910197
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Gender == M Mean  9434.220693842908  S.D -  9.248570271905049
```



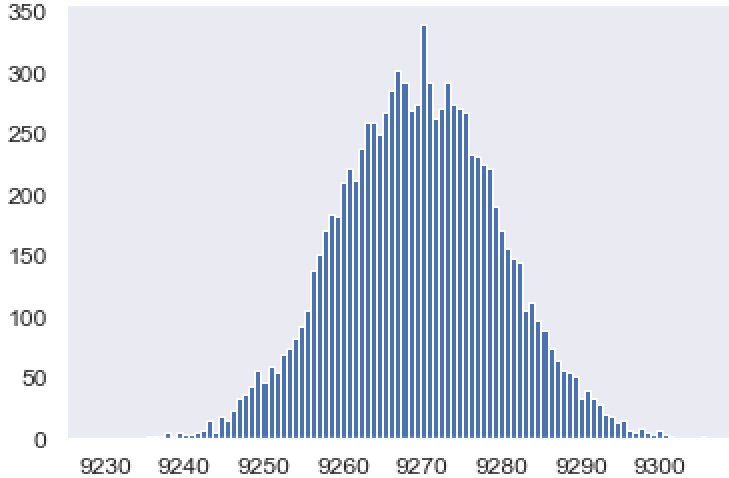

```
C.I (on the mean) [9416.093496109974, 9452.347891575842]
C.I (on the percentile) by [9416.29449969 9452.55852369]
C.I length  36.25439546586858
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Marital_Status (Confidence Interval)

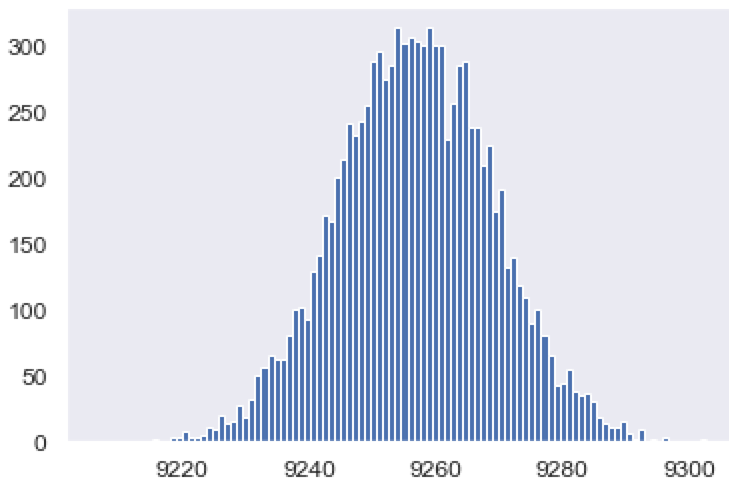
In [298…

```
getConfidenceIntervalByFeature("Marital_Status","Purchase",0.7,"Product_Category_All")
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Marital_Status == Un-Married Mean  9269.404120659836  S.D -  10.223493395464192
```




```
C.I (on the mean) [9249.366073604726, 9289.442167714946]
C.I (on the percentile) by [9249.09992107 9289.58412228]
C.I length  40.076094110219856
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Marital_Status == Married Mean  9256.705364256399  S.D -  12.292757916925515
```



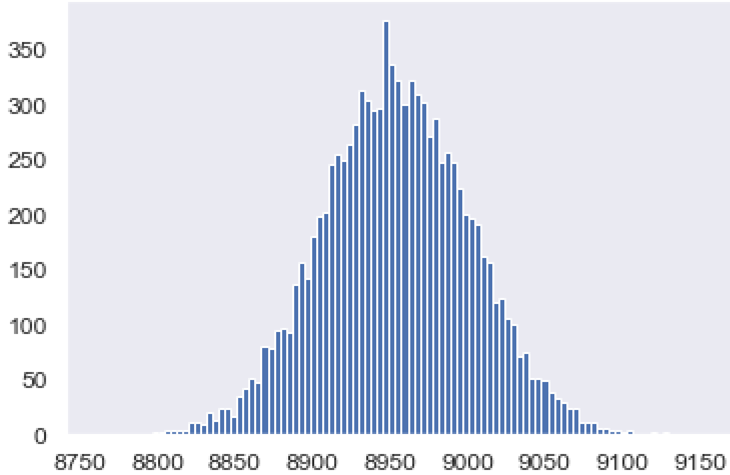

```
C.I (on the mean) [9232.611558739225, 9280.799169773572]
C.I (on the percentile) by [9232.78417226 9281.26008301]
C.I length  48.187611034347356
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

Age (Confidence Interval)

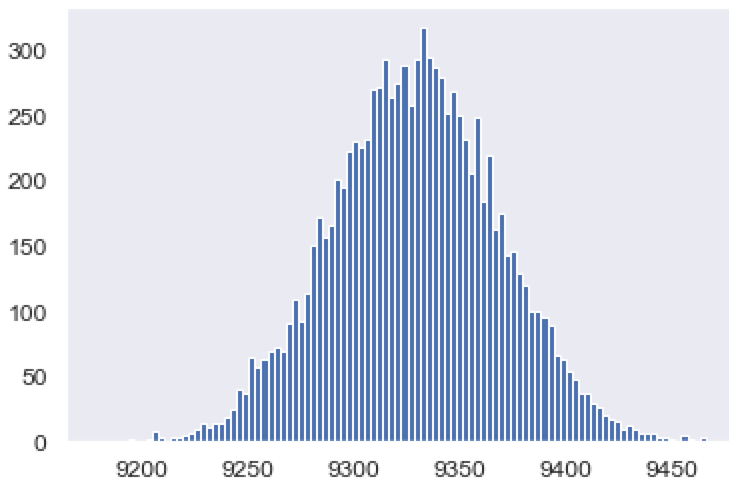
In [299…

```
getConfidenceIntervalByFeature("Age","Purchase",0.7,"Product_Category_All")
```

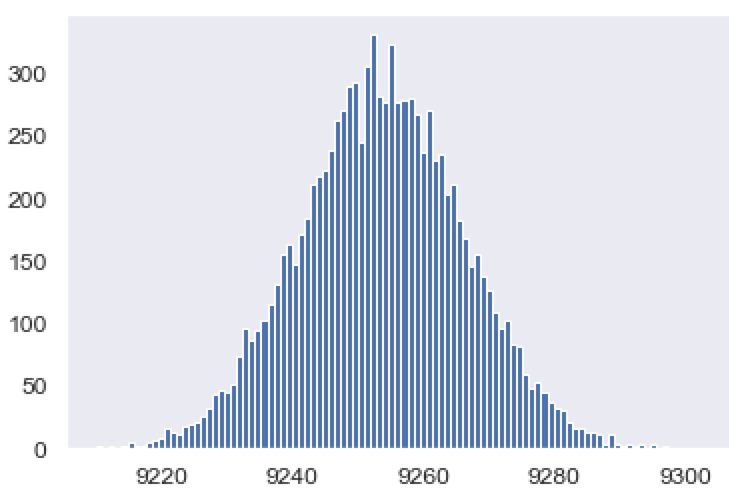
```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 0-17 Mean  8953.8233664139  S.D -  48.189136179751394
```



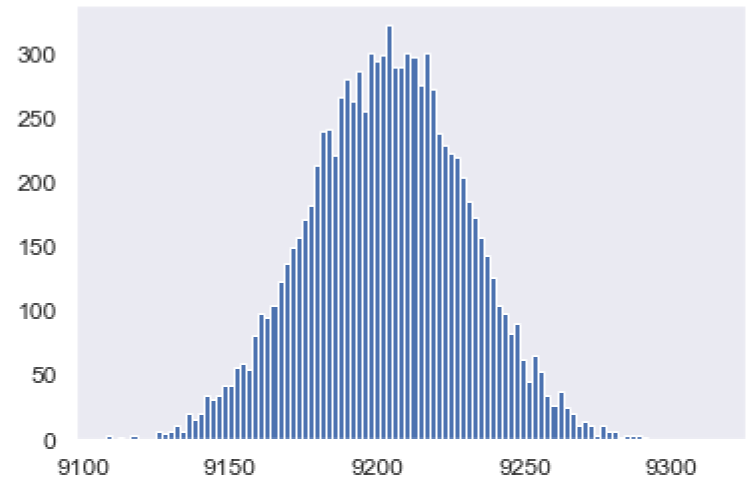

```
C.I (on the mean) [8859.372659501589, 9048.274073326213]
C.I (on the percentile) by [8859.72956182 9048.9992369 ]
C.I length  188.9014138246239
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 55+ Mean  9328.71241035208  S.D -  39.5551946706759
```



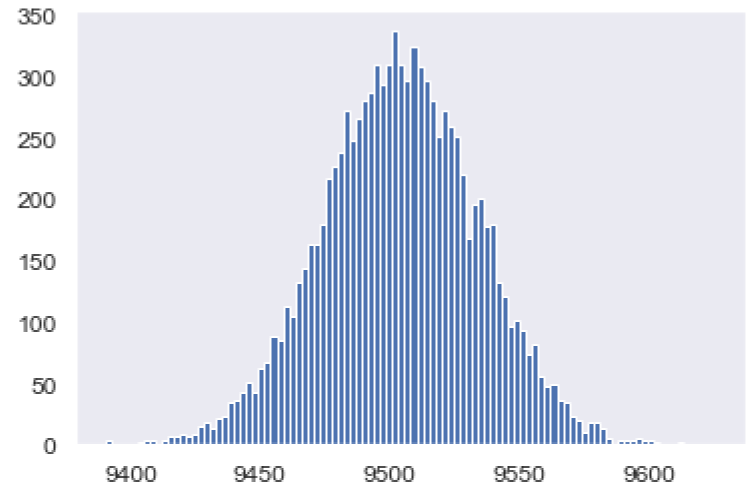

```
C.I (on the mean) [9251.184228797556, 9406.240591906604]
C.I (on the percentile) by [9251.75955329 9406.10029001]
C.I length  155.0563631090481
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 26-35 Mean  9253.596765900422  S.D -  12.614413644465603
```



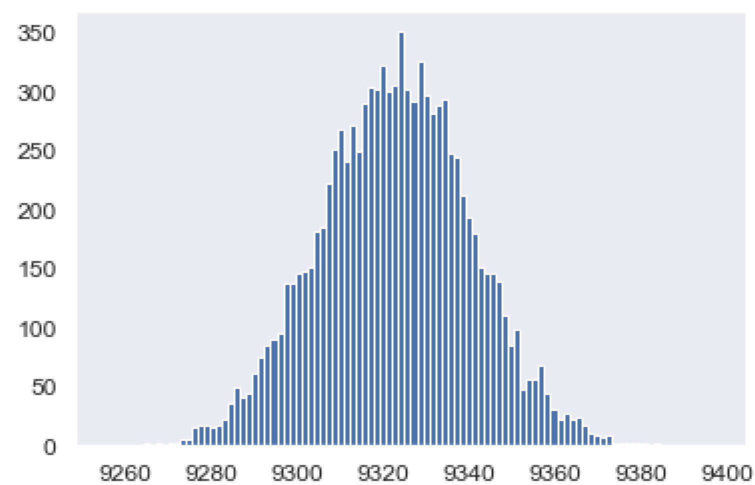

```
C.I (on the mean) [9228.872515157269, 9278.321016643575]
C.I (on the percentile) by [9228.95243735 9278.44814046]
C.I length  49.44850148630576
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 46-50 Mean  9203.139868563367  S.D -  26.9779525867384
```



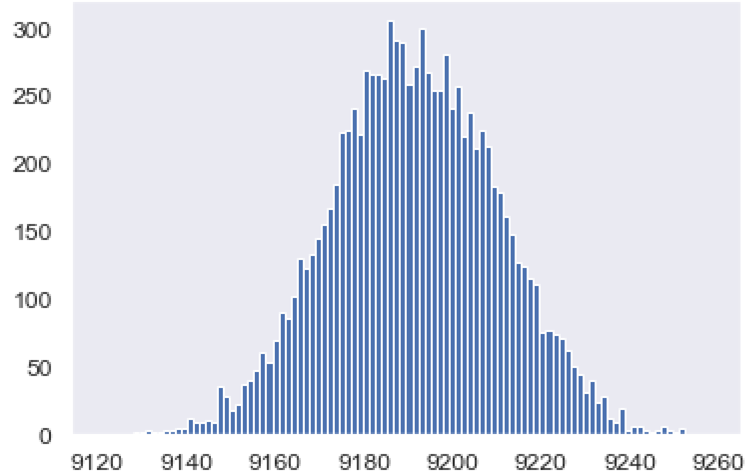

```
C.I (on the mean) [9150.26308149336, 9256.016655633373]
C.I (on the percentile) by [9149.59602327 9255.83352168]
C.I length  105.75357414001337
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 51-55 Mean  9503.819408909769  S.D -  30.127571201044287
```




```
C.I (on the mean) [9444.769369355721, 9562.869448463816]
C.I (on the percentile) by [9444.22479716 9562.58762878]
C.I length  118.10007910809509
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 36-45 Mean  9322.69674657536  S.D -  17.73075272523181
```




```
C.I (on the mean) [9287.944471233906, 9357.449021916815]
C.I (on the percentile) by [9287.74403747 9357.39365396]
C.I length  69.5045506829083
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 18-25 Mean  9191.266349393592  S.D -  18.851748929916305
```



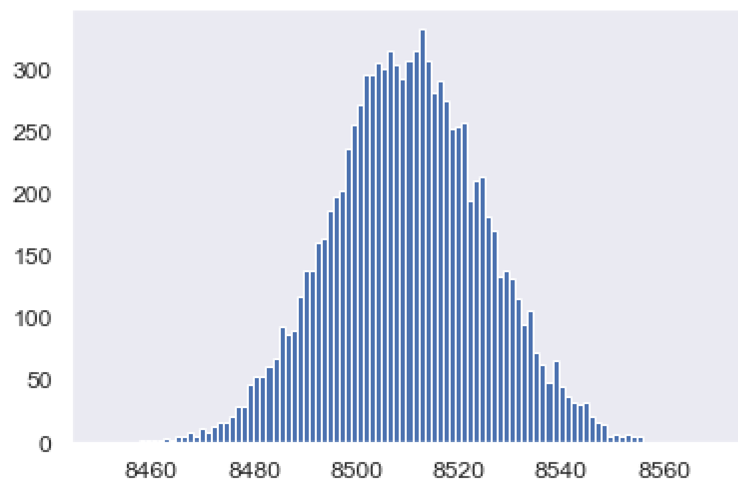

```
C.I (on the mean) [9154.316921490956, 9228.21577729623]
C.I (on the percentile) by [9154.67105908 9227.92701592]
C.I length  73.89885580527334
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

CI for selected Product Category

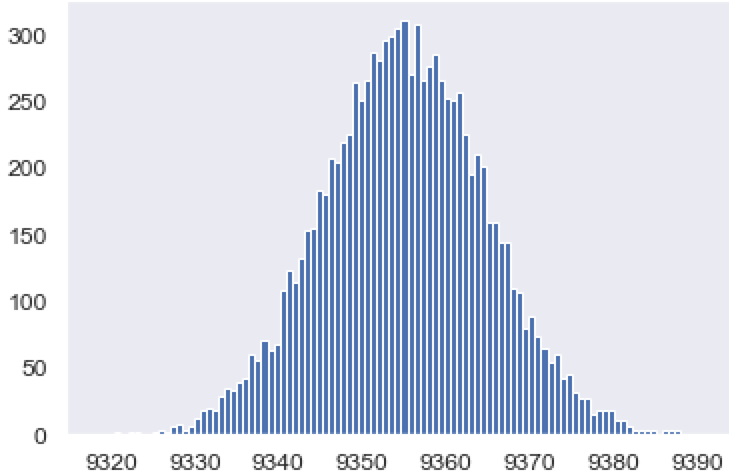
In [300…

```
getConfidenceIntervalByFeature("Gender","Purchase",0.7,"Product_category_1_5_8")
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Gender == F Mean  8509.97255433127  S.D -  15.166278523648097
```




```
C.I (on the mean) [8480.24664842492, 8539.698460237621]
C.I (on the percentile) by [8480.22798135 8539.90933269]
C.I length  59.451811812701635
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Gender == M Mean  9354.916858801425  S.D -  9.740679785440566
```

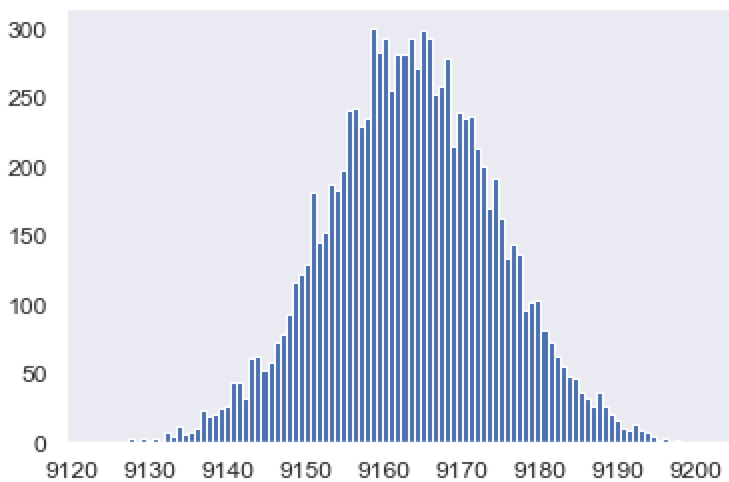



```
C.I (on the mean) [9335.825126421962, 9374.00859118089]
C.I (on the percentile) by [9335.73070073 9374.20957912]
C.I length  38.183464758927585
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

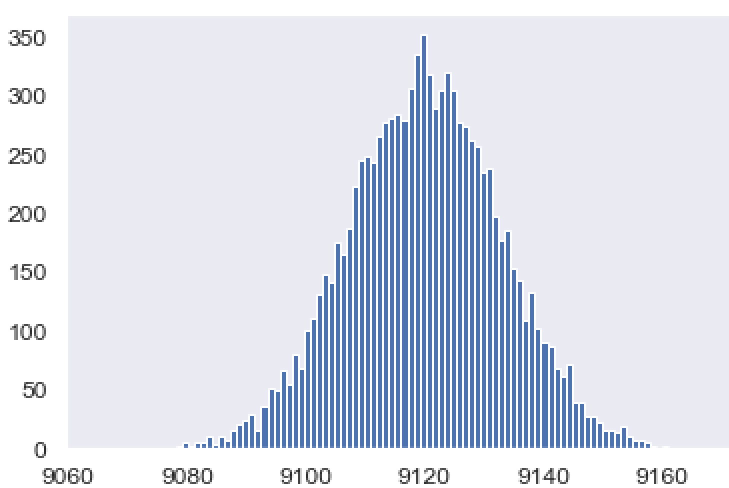
In [301…

```
getConfidenceIntervalByFeature("Marital_Status","Purchase",0.7,"Product_category_1_5_8")
```

```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Marital_Status == Un-Married Mean  9163.403381226546  S.D -  10.90388720319212
```




```
C.I (on the mean) [9142.03176230829, 9184.775000144802]
C.I (on the percentile) by [9141.49313879 9184.92732699]
C.I length  42.74323783651198
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Marital_Status == Married Mean  9119.884802769377  S.D -  12.90513721186992
```

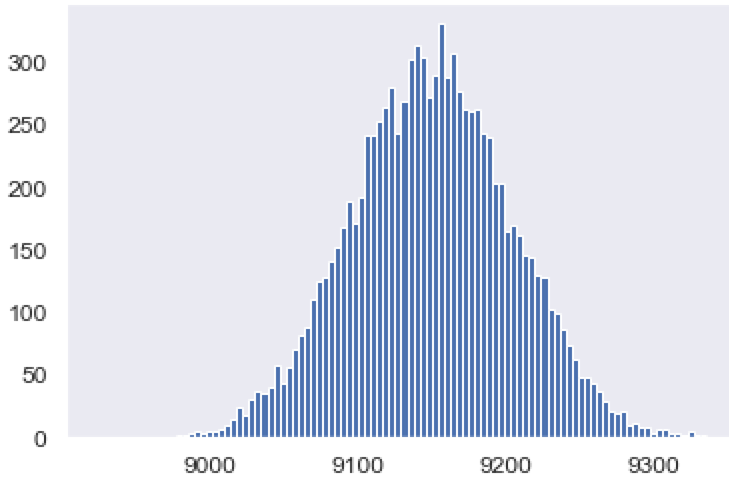


```
C.I (on the mean) [9094.590733834111, 9145.178871704642]
C.I (on the percentile) by [9094.50438908 9145.10506139]
C.I length  50.58813787053077
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

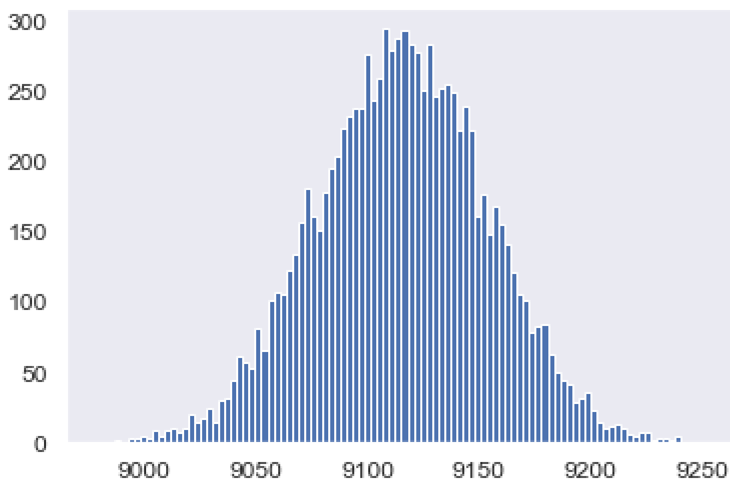
In [302…

```python
getConfidenceIntervalByFeature("Age","Purchase",0.7,"Product_category_1_5_8")
```

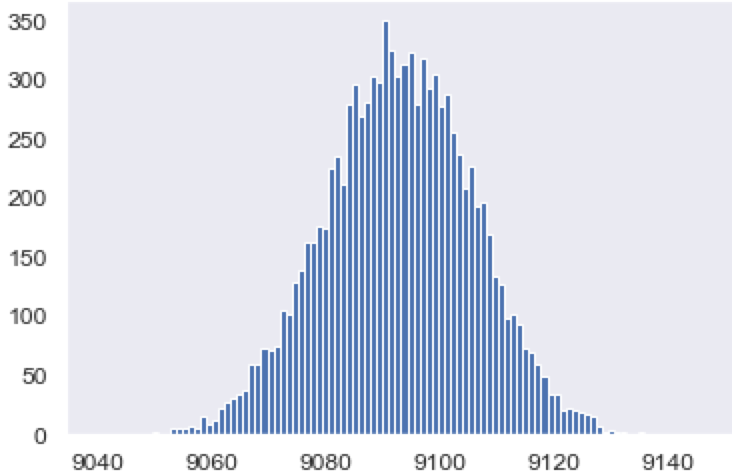
```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 0-17 Mean  9149.829537775595  S.D -  54.61355328117947
```



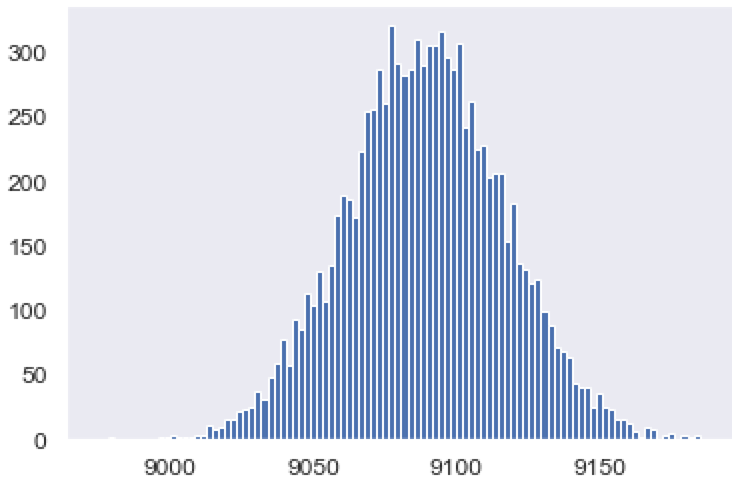

```
C.I (on the mean) [9042.786973344482, 9256.872102206707]
C.I (on the percentile) by [9043.12273206 9257.10039671]
C.I length  214.08512886222525
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 55+ Mean  9115.229604334227  S.D -  38.54036778575268
```



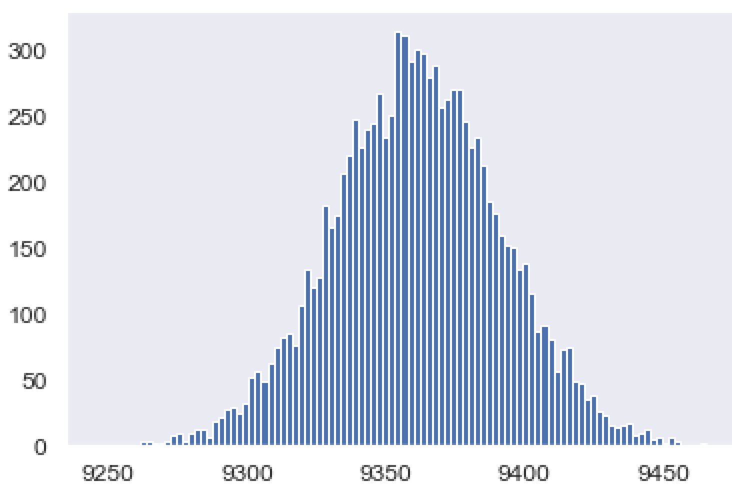

```
C.I (on the mean) [9039.69048347415, 9190.768725194303]
C.I (on the percentile) by [9040.43468052 9190.06072833]
C.I length  151.0782417201517
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 26-35 Mean  9092.811687781397  S.D -  13.106393296943022
```



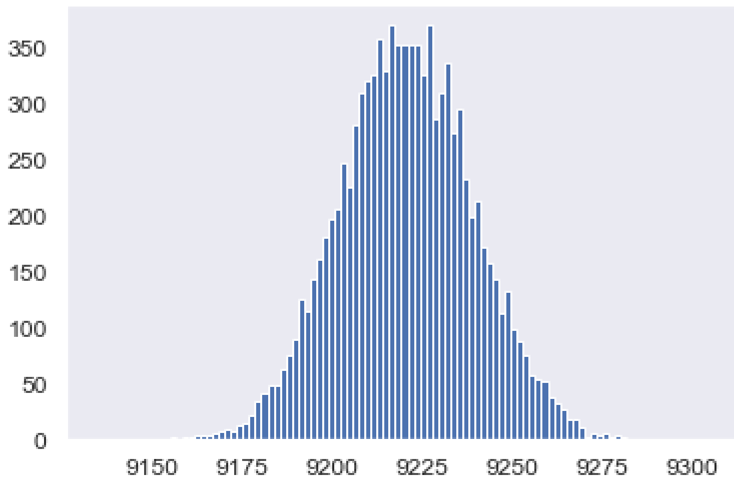

```
C.I (on the mean) [9067.12315691939, 9118.500218643405]
C.I (on the percentile) by [9066.91274938 9118.01737132]
C.I length  51.37706172401522
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 46-50 Mean  9088.950219615881  S.D -  27.837817788258885
```



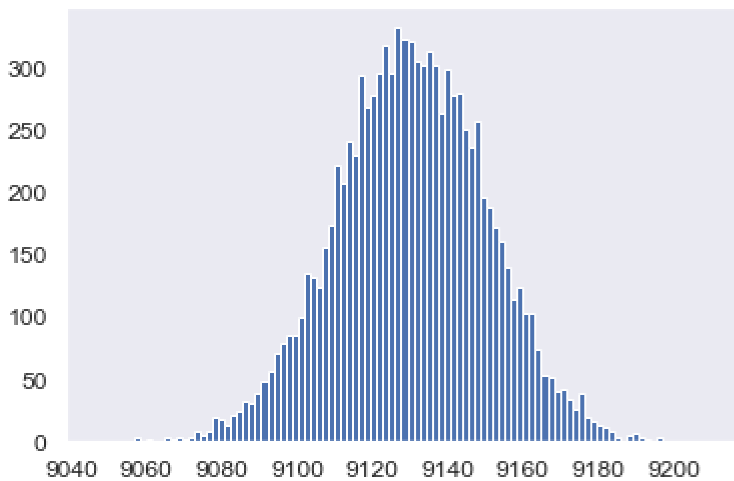

```
C.I (on the mean) [9034.388096750894, 9143.512342480868]
C.I (on the percentile) by [9034.62346893 9144.01300281]
C.I length  109.12424572997406
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 51-55 Mean  9361.178385906956  S.D -  30.4624798229407
```




```
C.I (on the mean) [9301.471925453992, 9420.88484635992]
C.I (on the percentile) by [9301.2778527  9420.72068748]
C.I length  119.41292090592833
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 36-45 Mean  9220.55444152675  S.D -  18.645607516769974
```




```
C.I (on the mean) [9184.009050793882, 9257.099832259619]
C.I (on the percentile) by [9184.48376583 9257.5301406 ]
C.I length  73.09078146573665
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by Age == 18-25 Mean  9130.180920408362  S.D -  20.075105222839404
```

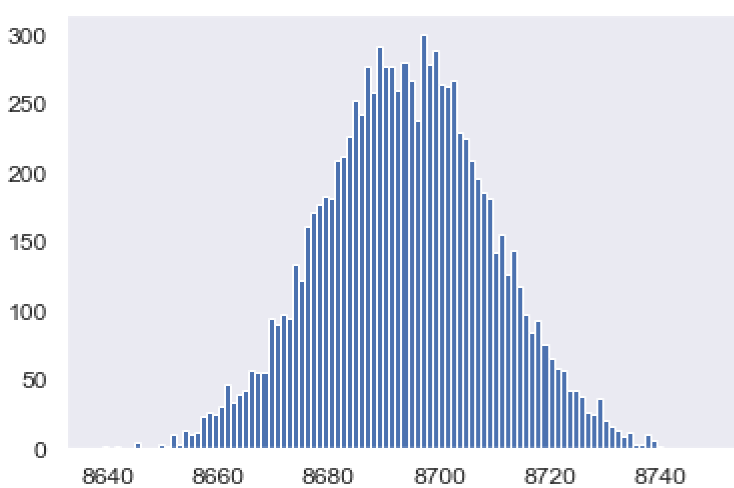



```
C.I (on the mean) [9090.833714171597, 9169.528126645127]
C.I (on the percentile) by [9090.47116942 9169.39782729]
C.I length  78.69441247353097
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

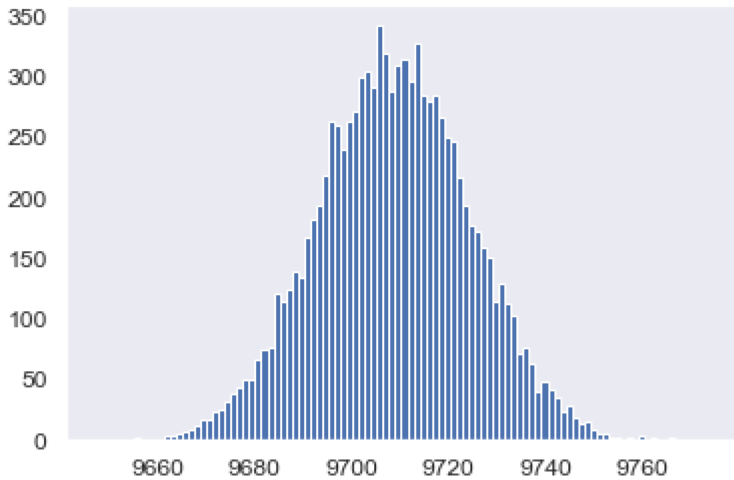
In [306…

```python
getConfidenceIntervalByFeature("City_Category","Purchase",0.7,"Product_category_1_5_8")
```

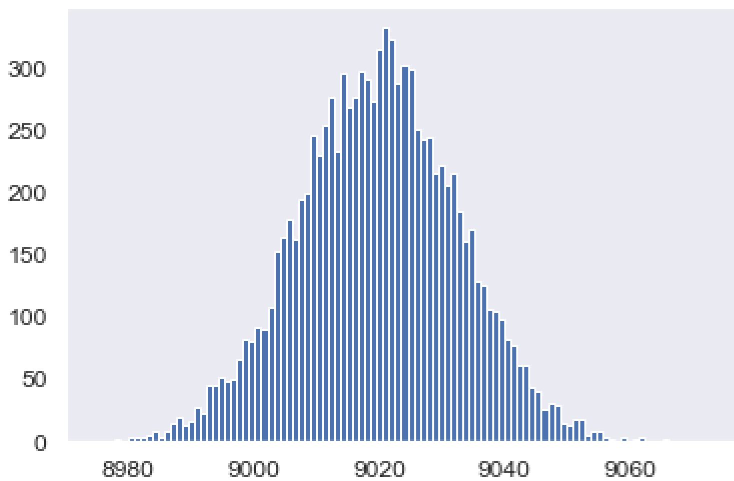
```
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by City_Category == A Mean  8693.8968781323  S.D -  15.394344949200736
```



```
C.I (on the mean) [8663.723962031867, 8724.069794232733]
C.I (on the percentile) by [8663.33165577 8724.42219264]
C.I length  60.345832200866425
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by City_Category == C Mean  9708.600337990782  S.D -  15.708773130652405
```




```
C.I (on the mean) [9677.811142654702, 9739.389533326861]
C.I (on the percentile) by [9677.71316024 9739.31187013]
C.I length  61.578390672159
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
CI by City_Category == B Mean  9019.733874512442  S.D -  12.737929636279409
```




```
C.I (on the mean) [8994.767532425334, 9044.70021659955]
C.I (on the percentile) by [8994.37904296 9044.32447714]
C.I length  49.932684174214955
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

In [308…

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 543238 entries, 0 to 545914
Data columns (total 10 columns):
 #   Column                      Non-Null Count   Dtype 
---  ------                      --------------   ----- 
 0   User_ID                     543238 non-null  object
 1   Product_ID                  543238 non-null  object
 2   Gender                      543238 non-null  object
 3   Age                         543238 non-null  object
 4   Occupation                  543238 non-null  object
 5   City_Category               543238 non-null  object
 6   Stay_In_Current_City_Years  543238 non-null  object
 7   Marital_Status              543238 non-null  object
 8   Product_Category            543238 non-null  object
 9   Purchase                    543238 non-null  int64 
dtypes: int64(1), object(9)
memory usage: 45.6+ MB
```

- Answering questions
  - Are women spending more money per transaction than men? Why or Why not?
    - For **product category 1,3,5 - Mens are spending more money than women**
  - Confidence intervals(CI) and distribution of the mean of the expenses by female and male customers
    - **Men** : CI ==> **[9416.093496109974, 9452.347891575842] length 36.25**
    - **Female** : CI ==> **[8715.38345693473, 8774.808879163831] length 59.42**
    - **Conclusion**: **More accurate estimation of expenses**(i.e. of population) **of Male** is possible **than that of Female** , as CI length is smaller(i.e 36.25) for men.
  - Are confidence intervals of average male and female spending overlapping? How can Walmart leverage this conclusion to make changes or improvements?
    - **Confidence intervals** of average expenses by male and female customers spending **not overlapping**
    - **Conclusion** - The **null hypothesis of zero difference** between **expenses by male and female customers** will be **rejected**. Hence **there is difference in expenses by male and female customers**
  - Results when the same activity is performed for Married vs Unmarried
    - **Married** : CI ==> **[9232.611558739225, 9280.799169773572] length 48.187611034347356**
    - **UnMarried**: CI ==> **[9249.366073604726, 9289.442167714946] length 40.076094110219856**
      - **Conclusion**: **Slitely better estimation of expenses** of **Un-Married people** than that of Married people as CI length is smaller for Un-Married people (i.e 40.07)
    - **Confidence intervals** of average expenses by Married and Unmarried customers spending **overlapping**
      - **Conclusion** - The **null hypothesis of zero difference** between **expenses by Married and Unmarried customers** will be **not be rejected**. Hence there is **no significant evidence** that there **ZERO difference in expenses by Married and Unmarried customers**
  - Results when the same activity is performed for Age
    - **0-17** CI ==> [8859.372659501589, 9048.274073326213] length 188.9014138246239
    - **18-25** CI ==> [9154.316921490956, 9228.21577729623] length 73.89885580527334
    - **26-35** CI ==> [9228.872515157269, 9278.321016643575] length 49.44850148630576
    - **36-45** CI ==> [9287.944471233906, 9357.449021916815] length 69.5045506829083
    - **46-50** CI ==> [9150.26308149336, 9256.016655633373] length 105.75357414001337
    - **51-55** CI ==> [9444.769369355721, 9562.869448463816] length 118.10007910809509
    - **55+** CI ==> [9251.184228797556, 9406.240591906604] length 155.0563631090481

    - **Conclusion** : Better estimation of expenses of population based on Age group **26-35** (length 49.45) , **36-45** (length 69.50), **18-25**(length 73.89)
  - **Confidence intervals** of average expenses by different age group customers spending **overlapping** , except group **0-17**
    - **Conclusion** - The **null hypothesis of zero difference** between **expenses by different age group customers** will be **not be rejected**. Hence there is **no significant evidence** that there **ZERO difference in expenses by diffrent age group customers**

- **Final Insights**

  - **Insights based on exploration**

    - **Gender** based - **Males spend more** on **Product categories - 1, 5 and 8 than females**
    - **Age** based - **Age group 26-35** spend more on **Product categories - 1, 5**
    - **Marital_Status** based - **1, 5 and 8 product categories** are more popular purchase among **un-married** customers
    - **City Category**
      - **Product categories - 1, 5 and 8** are **equally being purchased** in city category **A, B and C**
      - Except in "City C" , **overall purchase pattern almost similar across 'Gender', 'Age', 'Stay_In_Current_City_Years', 'Marital_Status'**
    - There is **varied purchase pattern** across **'Occupation','Product_Category'**
    - **Total purchase** for **product category 6,7,9,10,15,16 are higher** than other product categories
  - **Insights based on CLT** and **generalization on Population**

    - **More accurate estimation of expenses**(i.e. of population) **of Male** is possible **than that of Female** , as CI length is smaller(i.e 36.25) for men.
    - There is a **definite difference in expenses by male and female customers**
    - **Slitely better estimation of expenses** possible of **Un-Married customers** than that of Married customers as CI length is smaller for Un-Married people (i.e 40.07)
    - There is **no significant evidence** that there **ZERO difference in expenses by Married and Unmarried customers**
    - Better estimation of expenses of population possible based on Age group **26-35** (length 49.45) , **36-45** (length 69.50), **18-25**(length 73.89)
  - **Comments on the distribution of the variables and relationship between them**
    - Product category 1,5,8 have higest distribution of expenses
    - The distribution seems to follow normal disrtribution
  - **Comments for each univariate and bivariate plots**
    - **Analysis based on top product categories i.e. 1, 5 and 8**
      - **Gender**:
        - **Females spend more** on product category **5,8**
        - **Males spend more** on product category **1,5**
      - **Age group**:
        - **78% purchases (of product category 1,5,8)** are contributed by age group **26-35(40.5%) , 36-45 (19.7%), 18-25(18.1%)**
        - **26-35 spend more** on product category **5**
        - **55+ spend more** on product category **8**
      - **Occupation**:
        - **36.7% purchases (of product category 1,5,8)** are contributed across **occupation category 4, 0 and 7**

        - **45.4% purchases (of product category 1,5,8)** are contributed across **occupation category 4, 0 , 7 and 1**
    - **City Category**:
        - **41.9% purchases (of product category 1,5,8)** are contributed by **city B**
        - **Product category 1,5,8** are more popular in city **B and C** , i.e. **~ 73 % of purchases of product 1,5,8**
        - **Product category 5 is more popular in city B, A**
        - **Product category 8 is more popular in city B**
        - **Product category 1 is more popular in city B and C**
            - **Stay_In_Current_City_Years**:
            - **35.3% purchases for product categry 1,5,8 are contributed** by people **staying in the city for 1 year**
            - **Marital_Status**:
            - **Un-married** customers contributing to more purchase of **Product 1**

- **Recommendations**

- **Actionable items for business**
    - **Gender based targeted channel** would be effective to boost sales
        - **Males spend more** on product category **1,5**
            - More accurate positioning for Males
        - For **Females more focus on product category 5,8**
    - **Marital status** based targeted channels are very effective too
        - For **Un-married**, target **Product category 5**
        - Both **Un-married** and **married** clients, **target product category 1,8**
    - **Age group based targeted channels** can contribute to productive sales
        - For age group **26-35**, target product category **5**
            - More accurate positioning for 26-35 age group
        - For age group **36-45** and **18-25**, target for product category **1, 5 and 8**
    - **City category with Product category** based targeted channels will yield more results . **More focus on city category "B"** to multiple sales
        - Target **Product category 5 in city B, A**
        - Target **Product category 8 in city B**
        - Target **Product category 1 in city B and C**
        - Target **Product category 1,5,8** in city **B and C** [**~ 73 % of historical purchases of product 1,5,8**]