Problem Statement:

The Management team at Walmart Inc. wants to analyze the customer purchase behavior (specifically, purchase amount) against the customer's gender and the various other factors to help the business make better decisions. They want to understand if the spending habits differ between male and female customers: Do women spend more on Black Friday than men?

INSIGHTS:

- The median or Q2 of average spending of males and females is almost same
- The median or Q2 of average spending across all age groups is almost same
- Persons having occupation 0-4 are a lot in this
- The Average spending people living in C is more comapred to other 2
- The Average spending people living in A and B are almost same
- The median of spending is same for all the categories of stay_in_current_city_years
- The average spending is independent of marital_status
- this is the reason why the confidence intervals of marital status are overlapping
- The confidence interval of Gender are not overlapping

RECOMMENDATIONS

- Company should focus on increasing the spending of females.
- Company should provide more discounts on the products related to women
- Company should concentrate more on married because both married and singles spending same
- Company should focus on the other two cities, they should either bring more discounts in that areas
- Company shouls collect more data on marital status as the confidence intervals are overlapping
- Company doen't need to worry about the howmany years people staying in the current city,this means they have established good reputaion
- Company should focus on the occupation of the people having 6 to 8 years

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
import statsmodels.api as sm
```

In [2]:

```
data=pd.read_csv("C:/Users/Ajith/Desktop/scaler case studiess/walmart_data.csv")
```

Checking Characteristics of data

In [3]:

```
data.head()
```

Out[3]:

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

In [4]:

```
data.shape
```

Out[4]:

```
(550068, 10)
```

In [5]:

```
data.dtypes
```

Out[5]:

```
User_ID                        int64
Product_ID                    object
Gender                        object
Age                           object
Occupation                     int64
City_Category                 object
Stay_In_Current_City_Years    object
Marital_Status                 int64
Product_Category               int64
Purchase                       int64
dtype: object
```

In [6]:

```
data.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
```

Observations:

```
* the dataset contains 550068 rows and 10 columns
* There are 5 object and 5 int datatype
```

In [ ]:

2)Detecting null values and outliers

In [ ]:

In [7]:

```
data.isna().sum()
```

Out[7]:

```
User_ID                       0
Product_ID                    0
Gender                        0
Age                           0
Occupation                    0
City_Category                 0
Stay_In_Current_City_Years    0
Marital_Status                0
Product_Category              0
Purchase                      0
dtype: int64
```

There are no null values in the dataset

In [8]:

```
data.duplicated().sum()
```

Out[8]:

```
0
```

No data is duplicated

In [9]:

```
data.describe()
```

Out[9]:

	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 [10]:

```
data.describe(include='all')
```

Out[10]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Curr
count	5.500680e+05	550068	550068	550068	550068.000000	550068	
unique	NaN	3631	2	7	NaN	3	
top	NaN	P00265242	M	26-35	NaN	B	
freq	NaN	1880	414259	219587	NaN	231173	
mean	1.003029e+06	NaN	NaN	NaN	8.076707	NaN	
std	1.727592e+03	NaN	NaN	NaN	6.522660	NaN	
min	1.000001e+06	NaN	NaN	NaN	0.000000	NaN	
25%	1.001516e+06	NaN	NaN	NaN	2.000000	NaN	
50%	1.003077e+06	NaN	NaN	NaN	7.000000	NaN	
75%	1.004478e+06	NaN	NaN	NaN	14.000000	NaN	
max	1.006040e+06	NaN	NaN	NaN	20.000000	NaN	

In [11]:

```
#finding no of unique values
for i in data.columns:
    print(i,':',data[i].nunique())
```

```
User_ID : 5891
Product_ID : 3631
Gender : 2
Age : 7
Occupation : 21
City_Category : 3
Stay_In_Current_City_Years : 5
Marital_Status : 2
Product_Category : 20
Purchase : 18105
```

from above we can see that marital status can be converted into int datatype

checking valuecounts for each categorical columns

In [12]:

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

Out[12]:

```
M    414259
F    135809
Name: Gender, dtype: int64
```

In [13]:

```
data['Age'].value_counts(ascending=True)
```

Out[13]:

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

In [14]:

```
data['City_Category'].value_counts()
```

Out[14]:

```
B    231173
C    171175
A    147720
Name: City_Category, dtype: int64
```

In [15]:

```
data['Marital_Status'].value_counts()
```

Out[15]:

```
0    324731
1    225337
Name: Marital_Status, dtype: int64
```

In [16]:

```
data['Stay_In_Current_City_Years'].value_counts()
```

Out[16]:

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

In [83]:

```
data['Occupation'].value_counts(ascending=True)
```

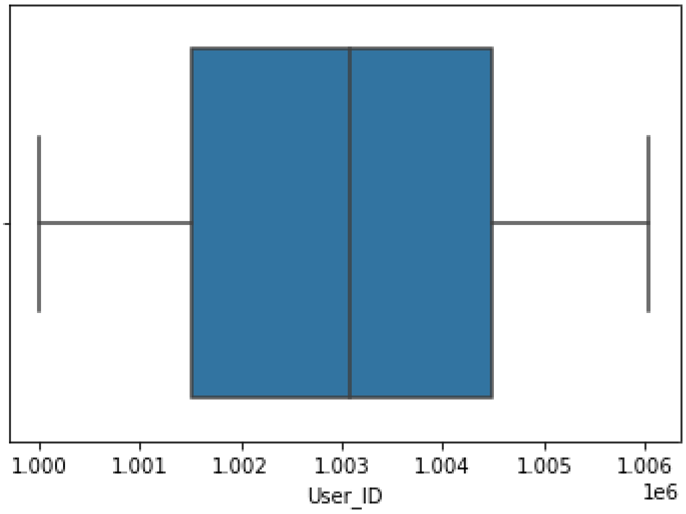
Out[83]:

```
8      1546
9      6291
18     6622
13     7728
19     8461
11    11586
15    12165
5     12177
10    12930
3     17650
6     20355
16    25371
2     26588
14    27309
12    31179
20    33562
17    40043
1     47426
7     59133
0     69638
4     72308
Name: Occupation, dtype: int64
```

Detecting outliers

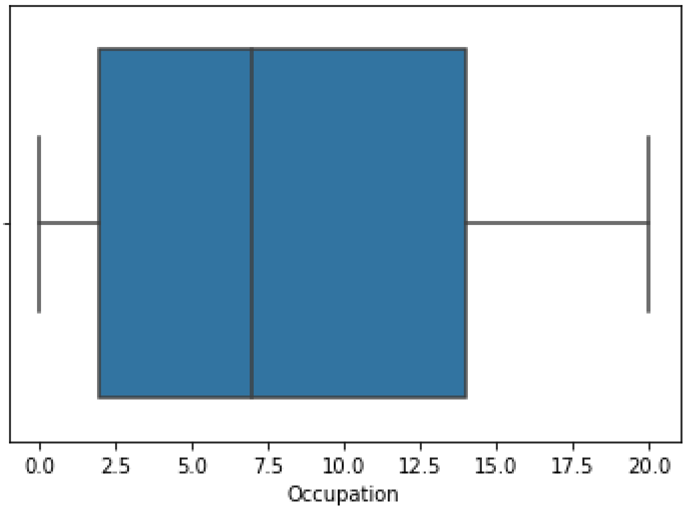
In [17]:

```
ax=sns.boxplot(x=data['User_ID'])
plt.show()
```



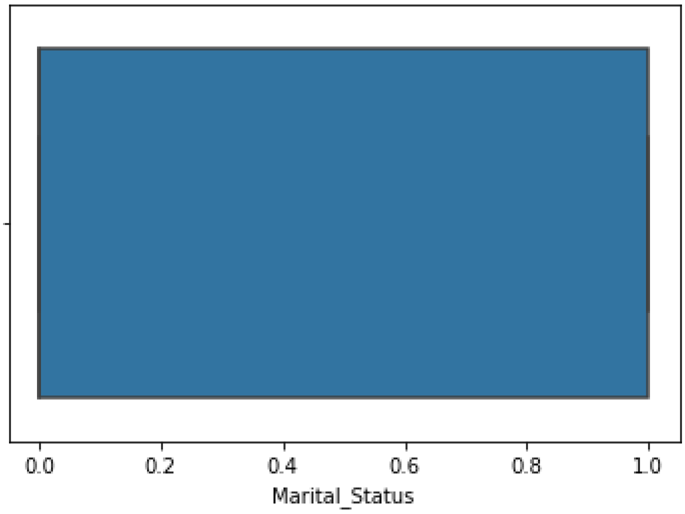
In [18]:

```
ax=sns.boxplot(x=data['Occupation'])
plt.show()
```



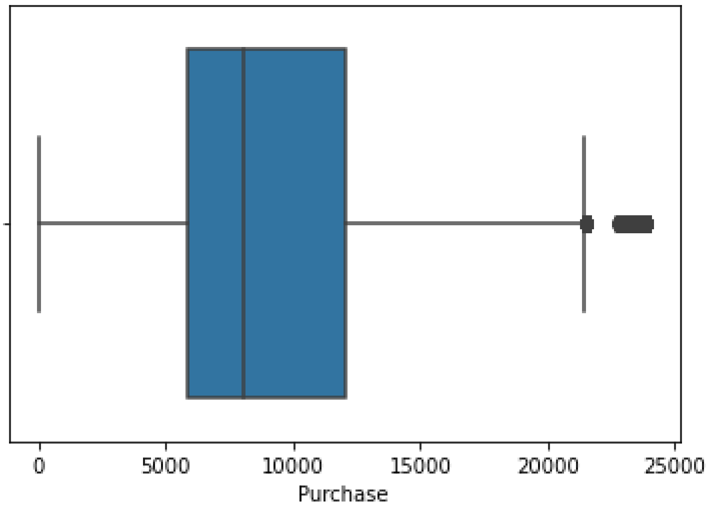
In [19]:

```
ax=sns.boxplot(x=data['Marital_Status'])
plt.show()
```



In [20]:

```
ax=sns.boxplot(x=data['Purchase'])
plt.show()
```



Clearly there are outliers in the purchase we can either

In [21]:

```
Q3=data['Purchase'].quantile(0.75)
Q1=data['Purchase'].quantile(0.25)
IQR=Q3-Q1
data1=data[(data['Purchase']>Q1-1.5*IQR)&(data['Purchase']<Q3+1.5*IQR)]
data1.shape[0]
print(data.shape[0]-data1.shape[0])# No of outliers
print(((data.shape[0]-data1.shape[0])/data.shape[0])*100)
```

```
2677
0.4866671029763593
```

Here the percentage of outliers is 0.48 so we can remove the outliers

In [22]:

```
data
```

Out[22]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Ye
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 10 columns

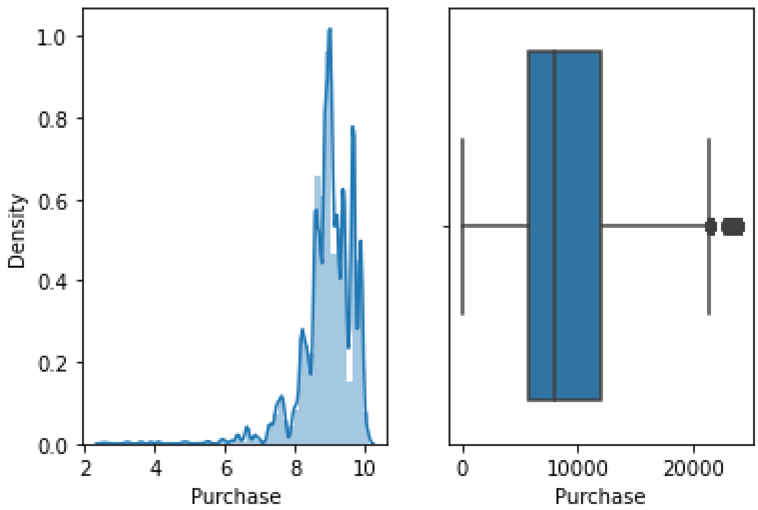
Visual Analysis

Univariate and Bivariate Analysis

In [23]:

```
# Continuous Variable
# Here purchase is Continuous Variable
# distplot
plt.subplot(121)
sns.distplot(np.log(data['Purchase']))
plt.subplot(122)
sns.boxplot(data['Purchase'])
plt.show()
```

```
C:\Users\Ajith\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
C:\Users\Ajith\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

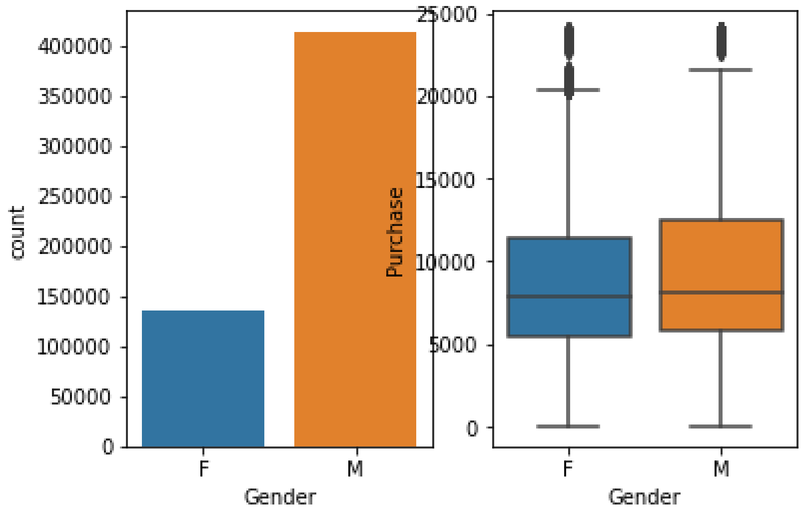


In [24]:

```
# categorical variables
plt.subplot(121)
sns.countplot(data=data,x='Gender')
plt.subplot(122)
sns.boxplot(data=data,x=data['Gender'],y=data['Purchase'])
```

Out[24]:

<AxesSubplot:xlabel='Gender', ylabel='Purchase'>

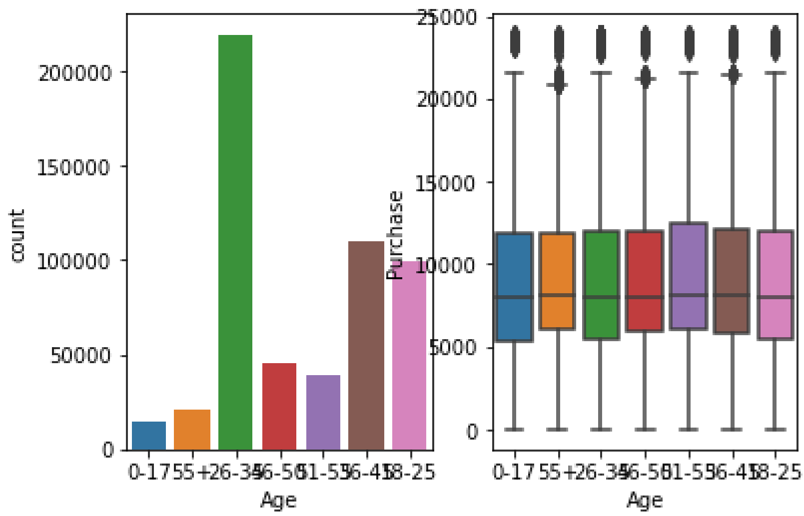


In [25]:

```
#Age
plt.subplot(121)
sns.countplot(data=data,x='Age')
plt.subplot(122)
sns.boxplot(data=data,x=data['Age'],y=data['Purchase'])
```

Out[25]:

<AxesSubplot:xlabel='Age', ylabel='Purchase'>

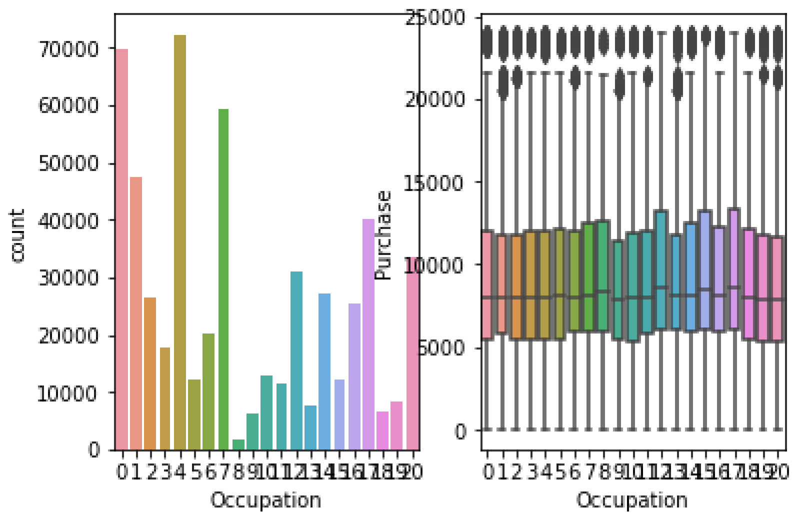


In [26]:

```
#Occupation
plt.subplot(121)
sns.countplot(data=data,x='Occupation')
plt.subplot(122)
sns.boxplot(data=data,x=data['Occupation'],y=data['Purchase'])
```

Out[26]:

```
<AxesSubplot:xlabel='Occupation', ylabel='Purchase'>
```

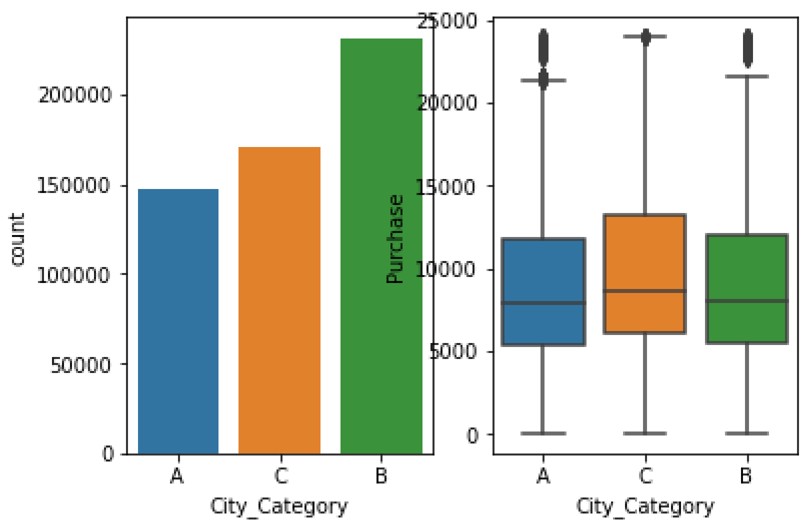


In [27]:

```
#City_Category
plt.subplot(121)
sns.countplot(data=data,x='City_Category')
plt.subplot(122)
sns.boxplot(data=data,x=data['City_Category'],y=data['Purchase'])
```

Out[27]:

```
<AxesSubplot:xlabel='City_Category', ylabel='Purchase'>
```

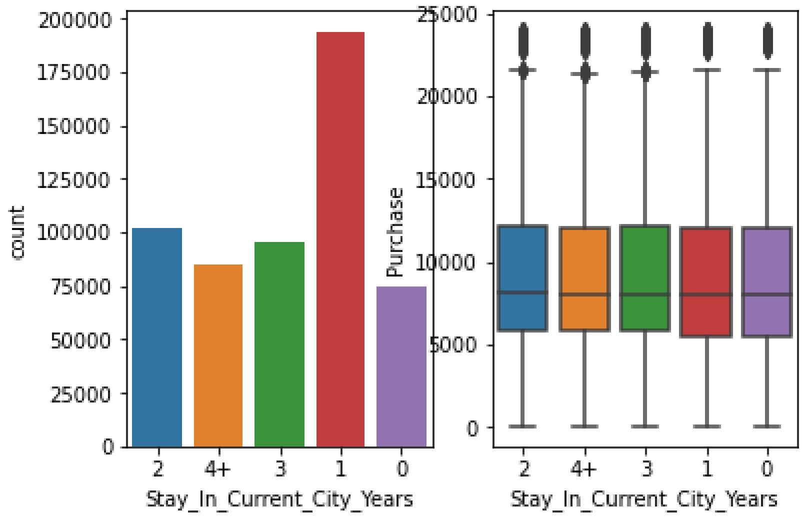


In [28]:

```
#Stay_in_
plt.subplot(121)
sns.countplot(data=data,x='Stay_In_Current_City_Years')
plt.subplot(122)
sns.boxplot(data=data,x=data['Stay_In_Current_City_Years'],y=data['Purchase'])
```

Out[28]:

<AxesSubplot:xlabel='Stay_In_Current_City_Years', ylabel='Purchase'>

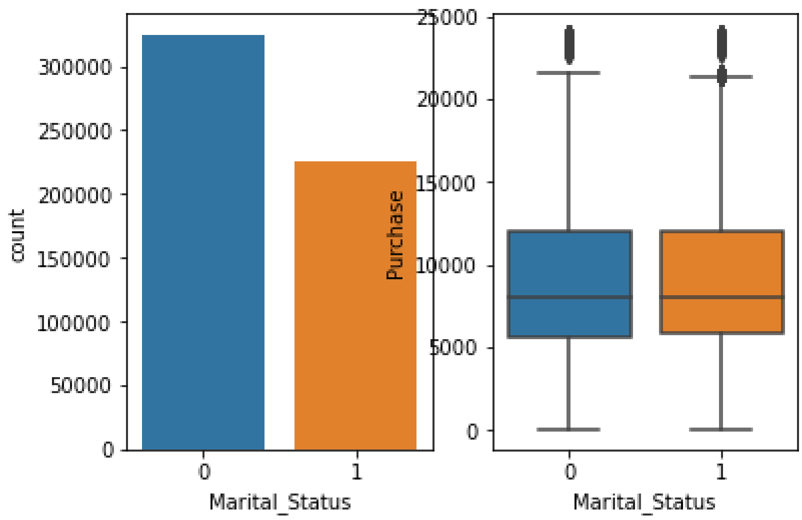


In [29]:

```
#Marital_Status
plt.subplot(121)
sns.countplot(data=data,x='Marital_Status')
plt.subplot(122)
sns.boxplot(data=data,x=data['Marital_Status'],y=data['Purchase'])
```

Out[29]:

<AxesSubplot:xlabel='Marital_Status', ylabel='Purchase'>

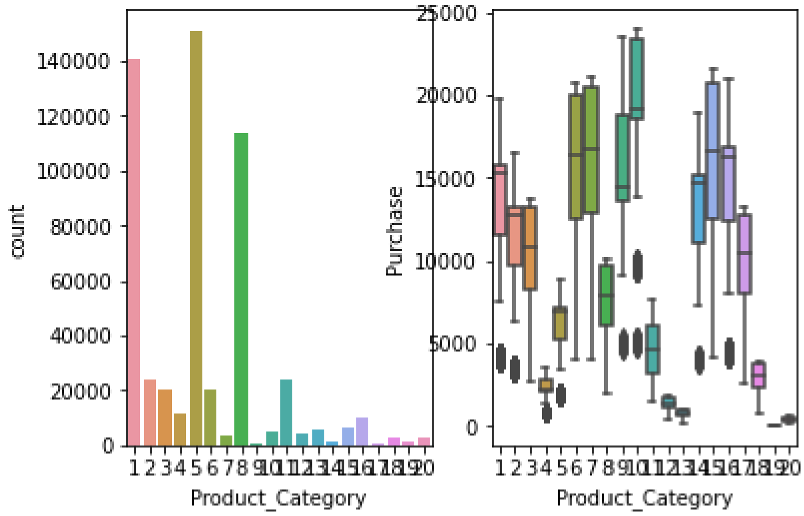


In [30]:

```
#Product category
plt.subplot(121)
sns.countplot(data=data,x='Product_Category')
plt.subplot(122)
sns.boxplot(data=data,x=data['Product_Category'],y=data['Purchase'])
```

Out[30]:

```
<AxesSubplot:xlabel='Product_Category', ylabel='Purchase'>
```



Correlation Analysis

In [38]:

```
data2=data.copy()
data
```

Out[38]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Ye
0	1000001	P00069042	F	0-17	10	A	
1	1000001	P00248942	F	0-17	10	A	
2	1000001	P00087842	F	0-17	10	A	
3	1000001	P00085442	F	0-17	10	A	
4	1000002	P00285442	M	55+	16	C	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	B	
550064	1006035	P00375436	F	26-35	1	C	
550065	1006036	P00375436	F	26-35	15	B	
550066	1006038	P00375436	F	55+	1	C	
550067	1006039	P00371644	F	46-50	0	B	

550068 rows × 10 columns

In [41]:

```
data2['City_Category'].replace(['A','B','C'],[0,1,2],inplace=True)
data2['Stay_In_Current_City_Years'].replace(['1','2','3','4+'],[1,2,3,4],inplace=True)
data2
```

Out[41]:

	User_ID	Product_ID	Gender	Age	Occupation	City_Category	Stay_In_Current_City_Ye
0	1000001	P00069042	F	0-17	10	0	
1	1000001	P00248942	F	0-17	10	0	
2	1000001	P00087842	F	0-17	10	0	
3	1000001	P00085442	F	0-17	10	0	
4	1000002	P00285442	M	55+	16	2	
...	...	...	...	...	...	...	
550063	1006033	P00372445	M	51-55	13	1	
550064	1006035	P00375436	F	26-35	1	2	
550065	1006036	P00375436	F	26-35	15	1	
550066	1006038	P00375436	F	55+	1	2	
550067	1006039	P00371644	F	46-50	0	1	

550068 rows × 10 columns

In [42]:

```
data2.corr()
```

Out[42]:

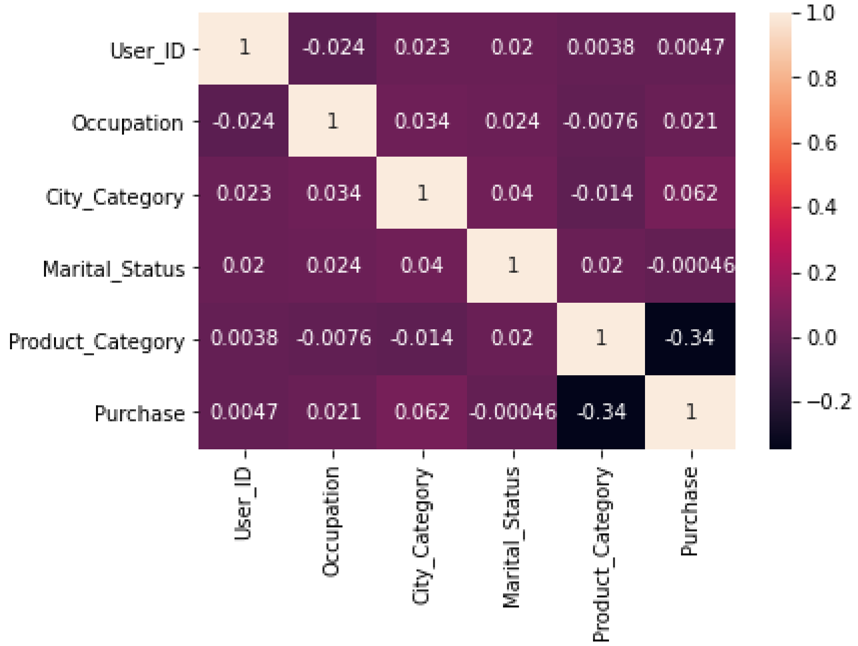
	User_ID	Occupation	City_Category	Marital_Status	Product_Category	Pur
User_ID	1.000000	-0.023971	0.022859	0.020443	0.003825	0.0
Occupation	-0.023971	1.000000	0.034479	0.024280	-0.007618	0.0
City_Category	0.022859	0.034479	1.000000	0.039790	-0.014364	0.0
Marital_Status	0.020443	0.024280	0.039790	1.000000	0.019888	-0.0
Product_Category	0.003825	-0.007618	-0.014364	0.019888	1.000000	-0.3
Purchase	0.004716	0.020833	0.061914	-0.000463	-0.343703	1.0

In [43]:

```
sns.heatmap(data2.corr(),annot=True)
```

Out[43]:

```
<AxesSubplot:>
```



Observations:

```
* Nearly every variable is independent of other
```

In [ ]:

```
data[data['Gender']=='F']['Purchase'].mean()
```

In [ ]:

```
data[data['Gender']=='M']['Purchase'].mean()
```

1)Are women spending more money per transaction than men? Why or Why not?

- Average amount spent per transaction by females is nearly 8671
- Average amount spent per transaction by males is nearly 9367
- Average amount spent per transaction by males is more than females

In [ ]:

```
x=data[data['Gender']=='F']
fe=x['Purchase']
r=1000
sample_means_females=np.empty(1000)
for i in range(r):
    z=np.random.choice(fe,size=10000)
    sample_means_females[i]=np.mean(z)
```

In [ ]:

```
plt.figure()
plt.hist(sample_means_females,bins=50)
plt.grid()
plt.show()
```

In [ ]:

```
y=data[data['Gender']=='M']
ma=y['Purchase']
r=1000
sample_means_males=np.empty(1000)
for i in range(r):
    z=np.random.choice(ma,size=10000)
    sample_means_males[i]=np.mean(z)
```

In [ ]:

```
plt.figure()
plt.hist(sample_means_males,bins=20)
plt.grid()
#sns.distplot(sample_means_females)
plt.show()
```

In [ ]:

```
sns.distplot(sample_means_males)
plt.show()
```

In [ ]:

```
f=sm.qqplot(sample_means_females,line='45',fit=True)
plt.grid()
```

In [ ]:

```
f=sm.qqplot(sample_means_males,line='45',fit=True)
plt.grid()
```

2)Confidence intervals and distribution of the mean of the expenses by female and male customers

Now we can apply CLT

In [ ]:

```
print(np.mean(sample_means_females))
print(sample_means_females.std())
```

In [ ]:

```
# Calculating 2.5th and 97.5th percentiles as per clt
print(np.percentile(sample_means_females,2.5))
print(np.percentile(sample_means_females,97.5))
```

In [ ]:

```
# Now Finding the 95% confidence interval
# As it follows clt the 95% confidene interval is (mu-1.96*sigma,mu+1.968sigma)
print(np.mean(sample_means_females)-1.96*np.std(sample_means_females))
print(np.mean(sample_means_females)+1.96*np.std(sample_means_females))
```

Calculating confidence interval for mens

In [ ]:

```
print(np.mean(sample_means_males))
print(sample_means_males.std())
```

In [ ]:

```
print(np.percentile(sample_means_males,2.5))
print(np.percentile(sample_means_males,97.5))
```

In [ ]:

```
print(np.mean(sample_means_males)-1.96*np.std(sample_means_males))
print(np.mean(sample_means_males)+1.96*np.std(sample_means_males))
```

- The 95% confidence intervals of average male and female spends are not overlapping i.e, (8643.244606007333,8826.614149392664) and (9292.506246571422,9584.215774628581)

Observations:

```
* Clearly the average spendings of females are less comapred to males
* Females are spending less compared to males
```

In [ ]:

```
```

In [ ]:

```

```

Now lets change the size of each sample

In [ ]:

```
size=5000
```

In [ ]:

```
x=data[data['Gender']=='F']
fe=x['Purchase']
r=1000
sample_means_females1=np.empty(1000)
for i in range(r):
    a=np.random.choice(fe,size=size)
    sample_means_females1[i]=np.mean(a)
```

In [ ]:

```
plt.figure()
plt.hist(sample_means_females1,bins=50)
plt.grid()
plt.show()
```

In [ ]:

```
y=data[data['Gender']=='M']
ma=y['Purchase']
r=1000
sample_means_males1=np.empty(1000)
for i in range(r):
    z=np.random.choice(ma,size=size)
    sample_means_males1[i]=np.mean(z)
```

In [ ]:

```
plt.figure()
plt.hist(sample_means_males1,bins=20)
plt.grid()
#sns.distplot(sample_means_females)
plt.show()
```

In [ ]:

```
f=sm.qqplot(sample_means_females1,line='45',fit=True)
plt.grid()
```

In [ ]:

```
f=sm.qqplot(sample_means_males1,line='45',fit=True)
plt.grid()
```

In [ ]:

```
print(np.mean(sample_means_females1))
print(sample_means_females1.std())
```

In [ ]:

```
print(np.percentile(sample_means_females1,2.5))
print(np.percentile(sample_means_females1,97.5))
```

In [ ]:

```
print(np.mean(sample_means_females1)-1.96*np.std(sample_means_females1))
print(np.mean(sample_means_females1)+1.96*np.std(sample_means_females1))
```

In [ ]:

```
print(np.mean(sample_means_males1))
print(sample_means_males1.std())
```

In [ ]:

```
print(np.percentile(sample_means_males1,2.5))
print(np.percentile(sample_means_males1,97.5))
```

In [ ]:

```
print(np.mean(sample_means_males1)-1.96*np.std(sample_means_males1))
print(np.mean(sample_means_males1)+1.96*np.std(sample_means_males1))
```

Calculating Confidence Interval for Marital_Status feature

In [ ]:

```
x=data[data['Marital_Status']==0]
fe=x['Purchase']
r=1000
sample_means_singles=np.empty(1000)
for i in range(r):
    z=np.random.choice(fe,size=10000)
    sample_means_singles[i]=np.mean(z)
```

In [ ]:

```
plt.figure()
plt.hist(sample_means_singles,bins=50)
plt.grid()
plt.show()
```

In [ ]:

```
x=data[data['Marital_Status']==1]
fe=x['Purchase']
r=1000
sample_means_married=np.empty(1000)
for i in range(r):
    z=np.random.choice(fe,size=10000)
    sample_means_married[i]=np.mean(z)
```

In [ ]:

```
plt.figure()
plt.hist(sample_means_married,bins=50)
plt.grid()
plt.show()
```

In [ ]:

```
f=sm.qqplot(sample_means_singles,line='45',fit=True)
plt.grid()
```

In [ ]:

```
f=sm.qqplot(sample_means_married,line='45',fit=True)
plt.grid()
```

In [ ]:

```
print(np.mean(sample_means_singles))
print(sample_means_singles.std())
```

In [ ]:

```
print(np.percentile(sample_means_singles,2.5))
print(np.percentile(sample_means_singles,97.5))
```

In [ ]:

```
ans_singles)-1.96*np.std(sample_means_singles),2),round(np.mean(sample_means_singles)+1.96*n
```

In [ ]:

```
print(np.mean(sample_means_married))
print(sample_means_married.std())
```

In [ ]:

```
print(np.percentile(sample_means_married,2.5))
print(np.percentile(sample_means_married,97.5))
```

In [ ]:

```
print(round(np.mean(sample_means_married)-1.96*np.std(sample_means_married),2),round(np.mea
```

Here we can clearly say that there is mostly overlapping of confidence intervals in case of marital status

Observations:

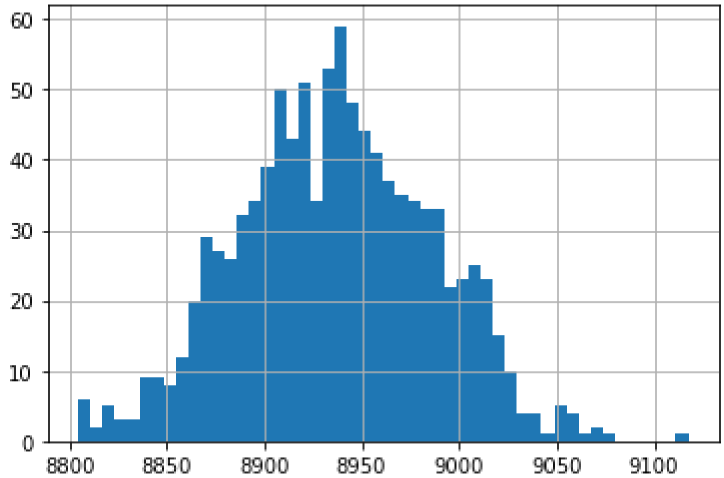
```
* The purchase is mostly independent of marital_status.
* Both married and singles have almost same confidence interval
```

In [50]:

```
x=data[data['Age']=='0-17']
fe=x['Purchase']
r=1000
sample_means_Age1=np.empty(1000)
for i in range(r):
    z=np.random.choice(fe,size=10000)
    sample_means_Age1[i]=np.mean(z)
```

In [51]:

```
plt.figure()
plt.hist(sample_means_Age1,bins=50)
plt.grid()
plt.show()
```

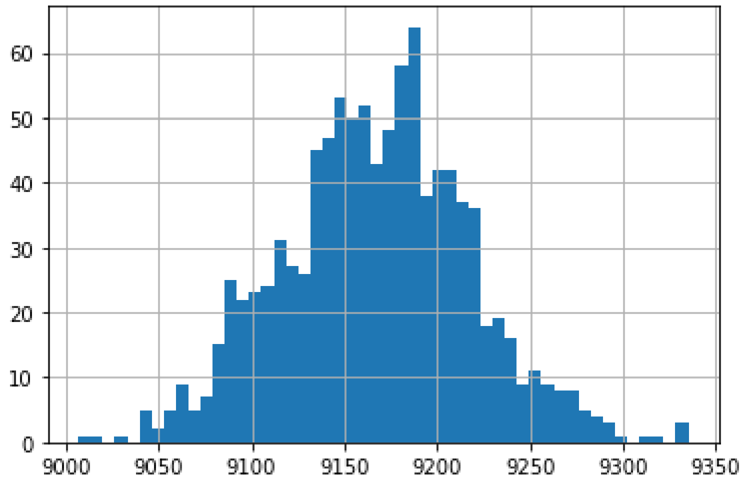


In [52]:

```
x=data[data['Age']=='18-25']
fe=x['Purchase']
r=1000
sample_means_Age2=np.empty(1000)
for i in range(r):
    z=np.random.choice(fe,size=10000)
    sample_means_Age2[i]=np.mean(z)
```

In [53]:

```
plt.figure()
plt.hist(sample_means_Age2,bins=50)
plt.grid()
plt.show()
```

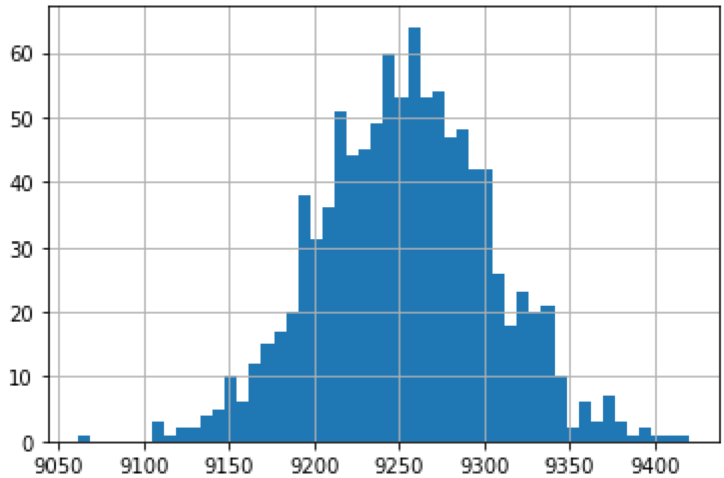


In [54]:

```
x=data[data['Age']=='26-35']
fe=x['Purchase']
r=1000
sample_means_Age3=np.empty(1000)
for i in range(r):
    z=np.random.choice(fe,size=10000)
    sample_means_Age3[i]=np.mean(z)
```

In [55]:

```
plt.figure()
plt.hist(sample_means_Age3,bins=50)
plt.grid()
plt.show()
```

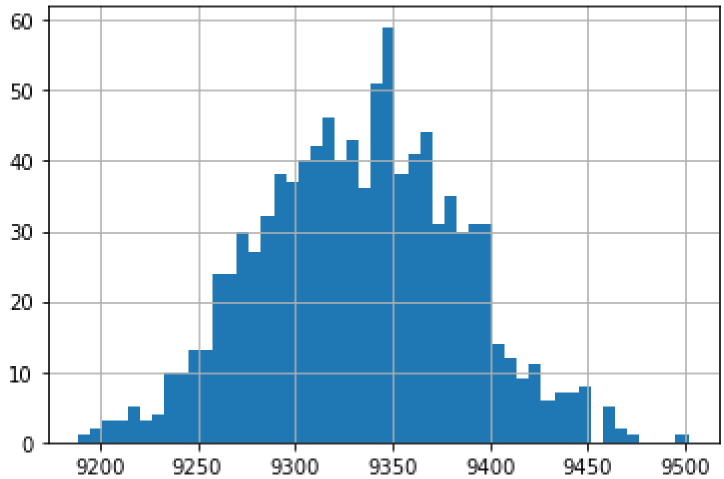


In [56]:

```
x=data[data['Age']=='36-45']
fe=x['Purchase']
r=1000
sample_means_Age4=np.empty(1000)
for i in range(r):
    z=np.random.choice(fe,size=10000)
    sample_means_Age4[i]=np.mean(z)
```

In [57]:

```
plt.figure()
plt.hist(sample_means_Age4,bins=50)
plt.grid()
plt.show()
```

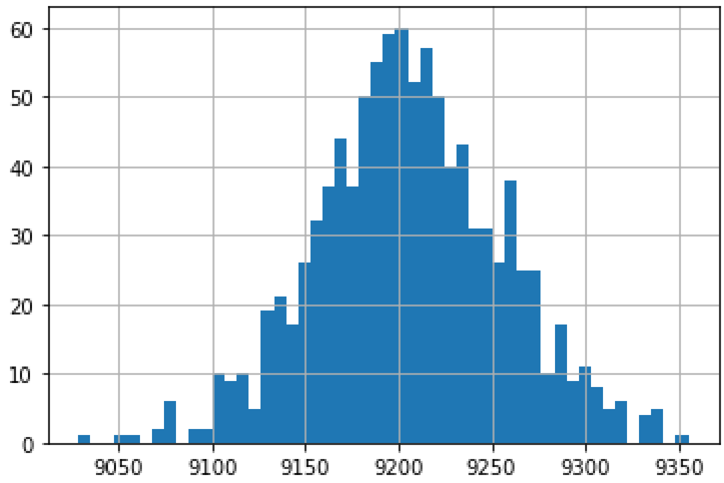


In [61]:

```
x=data[data['Age']=='46-50']
fe=x['Purchase']
r=1000
sample_means_Age5=np.empty(1000)
for i in range(r):
    z=np.random.choice(fe,size=10000)
    sample_means_Age5[i]=np.mean(z)
```

In [62]:

```
plt.figure()
plt.hist(sample_means_Age5,bins=50)
plt.grid()
plt.show()
```

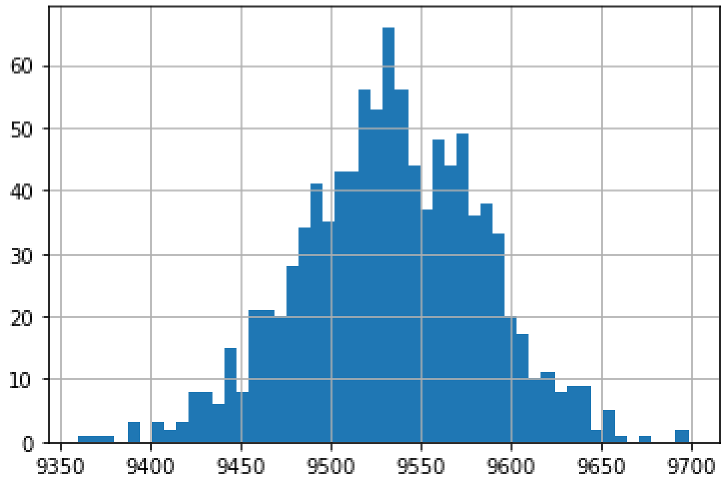


In [63]:

```
x=data[data['Age']=='51-55']
fe=x['Purchase']
r=1000
sample_means_Age6=np.empty(1000)
for i in range(r):
    z=np.random.choice(fe,size=10000)
    sample_means_Age6[i]=np.mean(z)
```

In [64]:

```
plt.figure()
plt.hist(sample_means_Age6,bins=50)
plt.grid()
plt.show()
```

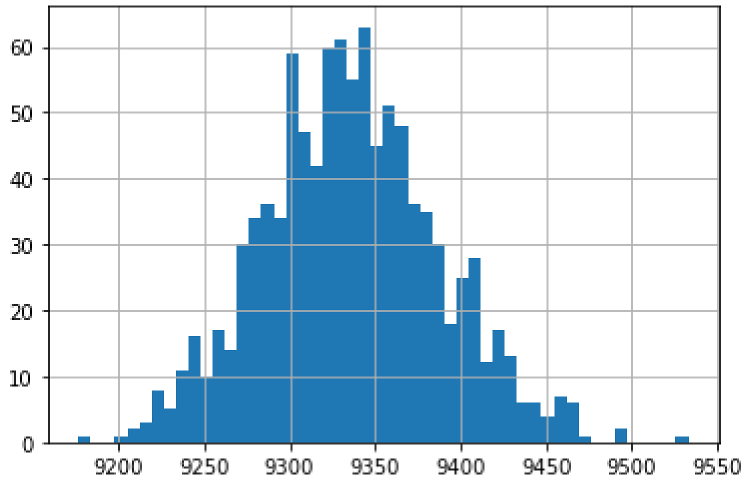


In [65]:

```
x=data[data['Age']=='55+']
fe=x['Purchase']
r=1000
sample_means_Age7=np.empty(1000)
for i in range(r):
    z=np.random.choice(fe,size=10000)
    sample_means_Age7[i]=np.mean(z)
```

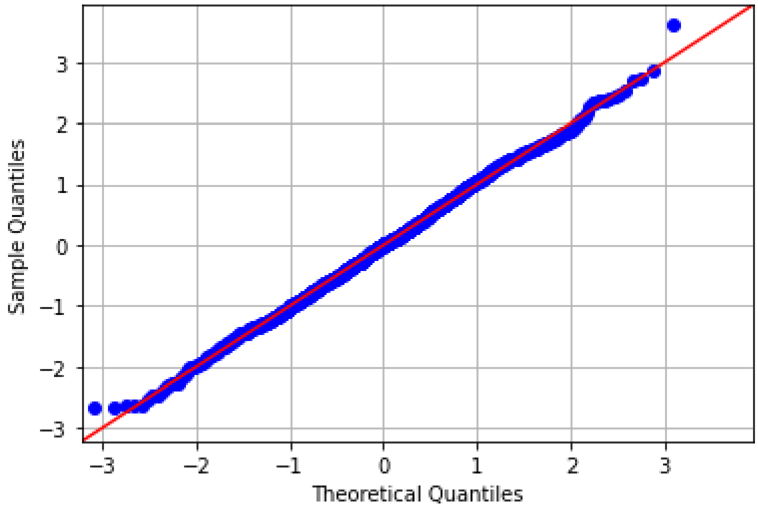
In [66]:

```
plt.figure()
plt.hist(sample_means_Age7,bins=50)
plt.grid()
plt.show()
```



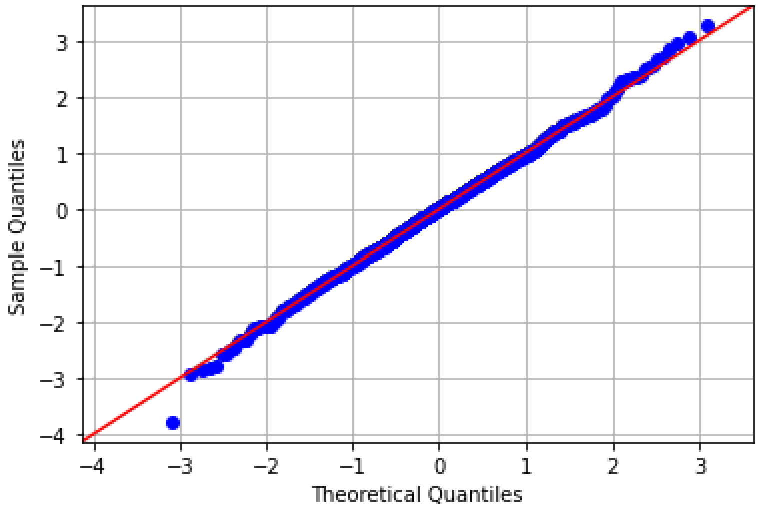
In [67]:

```
f=sm.qqplot(sample_means_Age1,line='45',fit=True)
plt.grid()
```



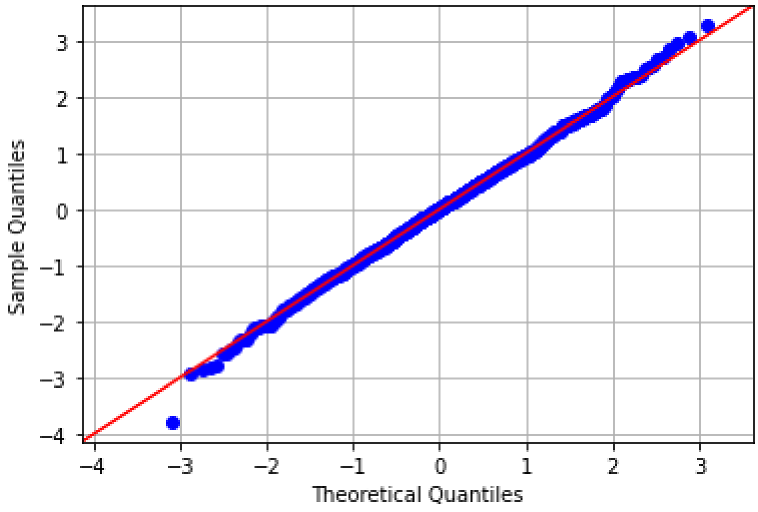
In [68]:

```
f=sm.qqplot(sample_means_Age3,line='45',fit=True)
plt.grid()
```



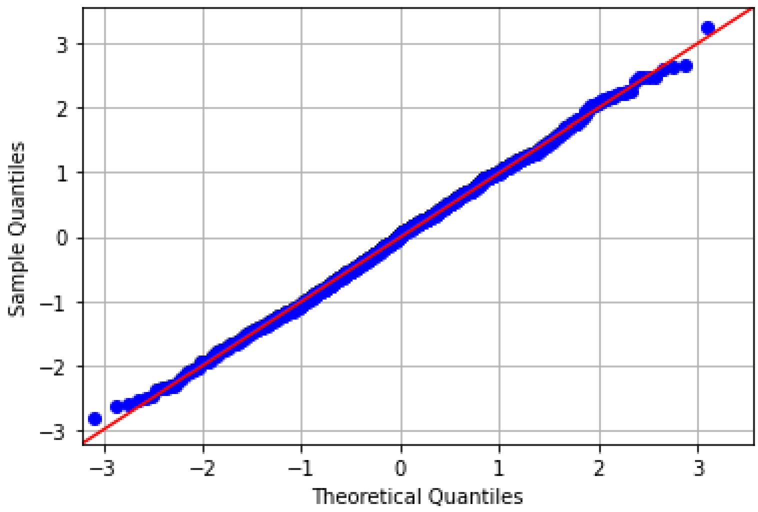
In [69]:

```
f=sm.qqplot(sample_means_Age3,line='45',fit=True)
plt.grid()
```



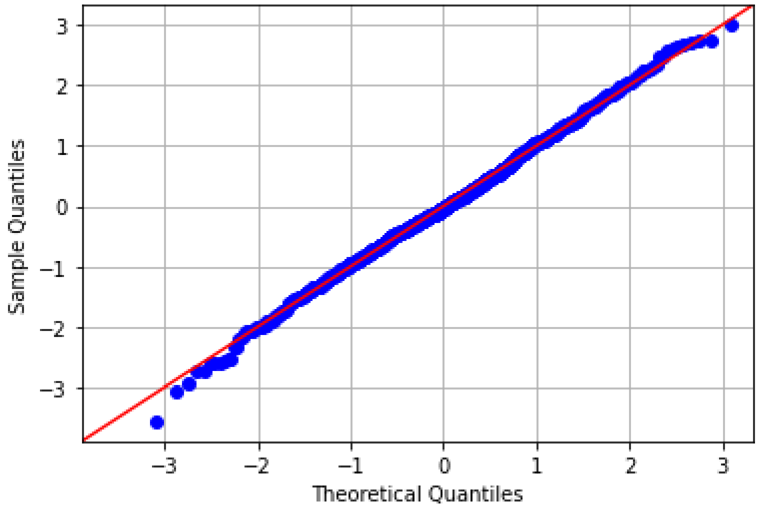
In [70]:

```
f=sm.qqplot(sample_means_Age4,line='45',fit=True)
plt.grid()
```



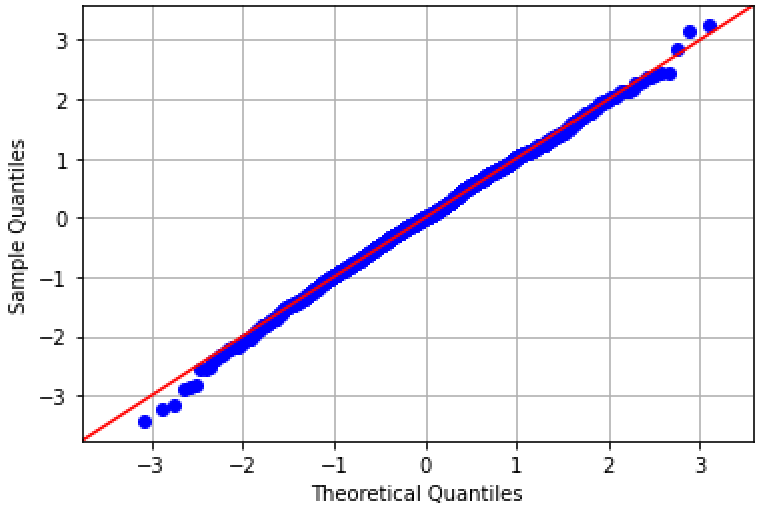
In [71]:

```
f=sm.qqplot(sample_means_Age5,line='45',fit=True)
plt.grid()
```



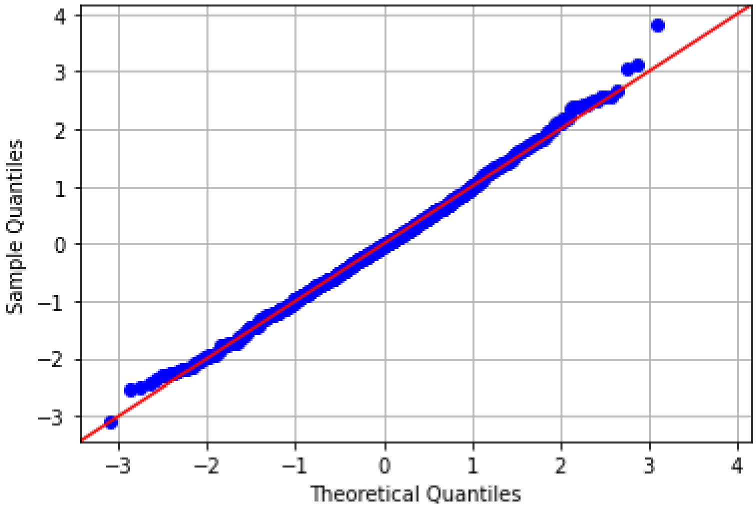
In [72]:

```
f=sm.qqplot(sample_means_Age6,line='45',fit=True)
plt.grid()
```



In [73]:

```
f=sm.qqplot(sample_means_Age7,line='45',fit=True)
plt.grid()
```



In [74]:

```
print(np.mean(sample_means_Age1))
print(sample_means_Age1.std())
print(np.percentile(sample_means_Age1,2.5))
print(np.percentile(sample_means_Age1,97.5))
print(np.mean(sample_means_Age1)-1.96*np.std(sample_means_Age1))
print(np.mean(sample_means_Age1)+1.96*np.std(sample_means_Age1))
```

```
8936.6996774
49.88596529662554
8840.51791
9029.0333075
8838.923185418615
9034.476169381385
```

In [75]:

```
print(np.mean(sample_means_Age2))
print(sample_means_Age2.std())
print(np.percentile(sample_means_Age2,2.5))
print(np.percentile(sample_means_Age2,97.5))
print(np.mean(sample_means_Age2)-1.96*np.std(sample_means_Age2))
print(np.mean(sample_means_Age2)+1.96*np.std(sample_means_Age2))
```

```
9167.080334199998
50.509479681330404
9066.803765
9269.3983125
9068.08175402459
9266.078914375406
```

In [76]:

```
print(np.mean(sample_means_Age3))
print(sample_means_Age3.std())
print(np.percentile(sample_means_Age3,2.5))
print(np.percentile(sample_means_Age3,97.5))
print(np.mean(sample_means_Age3)-1.96*np.std(sample_means_Age3))
print(np.mean(sample_means_Age3)+1.96*np.std(sample_means_Age3))
```

```
9253.6478414
50.80350762303648
9149.2712725
9352.95449
9154.072966458847
9353.222716341152
```

In [77]:

```
print(np.mean(sample_means_Age4))
print(sample_means_Age4.std())
print(np.percentile(sample_means_Age4,2.5))
print(np.percentile(sample_means_Age4,97.5))
print(np.mean(sample_means_Age4)-1.96*np.std(sample_means_Age4))
print(np.mean(sample_means_Age4)+1.96*np.std(sample_means_Age4))
```

```
9333.517395599998
51.59676251581295
9233.742065
9438.48226
9232.387741069004
9434.647050130992
```

In [78]:

```
print(np.mean(sample_means_Age5))
print(sample_means_Age5.std())
print(np.percentile(sample_means_Age5,2.5))
print(np.percentile(sample_means_Age5,97.5))
print(np.mean(sample_means_Age5)-1.96*np.std(sample_means_Age5))
print(np.mean(sample_means_Age5)+1.96*np.std(sample_means_Age5))
```

```
9205.8802727
49.832503661569746
9107.497585000001
9305.5260425
9108.208565523324
9303.551979876676
```

In [79]:

```
print(np.mean(sample_means_Age6))
print(sample_means_Age6.std())
print(np.percentile(sample_means_Age6,2.5))
print(np.percentile(sample_means_Age6,97.5))
print(np.mean(sample_means_Age6)-1.96*np.std(sample_means_Age6))
print(np.mean(sample_means_Age6)+1.96*np.std(sample_means_Age6))
```

```
9533.931625
50.74550781923633
9429.87811
9633.049737500001
9434.470429674297
9633.392820325702
```

In [80]:

```
print(np.mean(sample_means_Age7))
print(sample_means_Age7.std())
print(np.percentile(sample_means_Age7,2.5))
print(np.percentile(sample_means_Age7,97.5))
print(np.mean(sample_means_Age7)-1.96*np.std(sample_means_Age7))
print(np.mean(sample_means_Age7)+1.96*np.std(sample_means_Age7))
```

```
9336.238950500001
51.63261331481659
9235.895672499999
9443.458095
9235.03902840296
9437.438872597042
```

#Observations: * Here the age has particular confidence intervals

INSIGHTS:

```
* The median or Q2 of average spending of males and females is almost same
* The median or Q2 of average spending across all age groups is almost same
* Persons having occupation 0-4 are a lot in this
* The Average spending people living in C is more comapred to other 2
* The Average spending people living in A and B are almost same
* The median of spending is same for all the categories of stay_in_current_city_ye
ars
* The average spending is independent of marital_status
* this is the reason why the confidence intervals of marital status are overlappin
g
* The confidence interval of Gender are not overlapping
```

RECOMMENDATIONS

```
* Company should focus on increasing the spending of females.
* Company should provide more discounts on the products related to women
* Company should concentrate more on married because both married and singles spen
ding same
* Company should focus on the other two cities, they should either bring more disc
ounts in that areas
* Company shouls collect more data on marital status as the confidence intervals a
re overlapping
* Company doen't need to worry about the howmany years people staying in the curre
nt city,this means they have established
  good reputaion
* Company should focus on the occupation of the people having 6 to 8 years
```

In [ ]: