In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.linear_model import LinearRegression,Lasso
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
supermarket_sales=pd.read_csv(r'C:\Users\18F17897\Desktop\supermarket_sales.c
supermarket_sales.head()
```

Out[2]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	T
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3

In [3]:
```
supermarket_sales.shape
```

Out[3]: (1000, 17)

In [4]:
```
supermarket_sales.isnull().sum()
```

Out[4]:
```
Invoice ID                 1
Branch                     1
City                       0
Customer type              0
Gender                     1
Product line               0
Unit price                 0
Quantity                   0
Tax 5%                     0
Total                      1
Date                       0
Time                       0
Payment                    2
cogs                       0
gross margin percentage    0
gross income               0
Rating                     1
dtype: int64
```

In [5]:
```
supermarket_sales=supermarket_sales.dropna()
```

In [6]:
```
supermarket_sales.isnull().sum()
```

Out[6]:
```
Invoice ID                 0
Branch                     0
City                       0
Customer type              0
Gender                     0
Product line               0
Unit price                 0
Quantity                   0
Tax 5%                     0
Total                      0
Date                       0
Time                       0
Payment                    0
cogs                       0
gross margin percentage    0
gross income               0
Rating                     0
dtype: int64
```

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

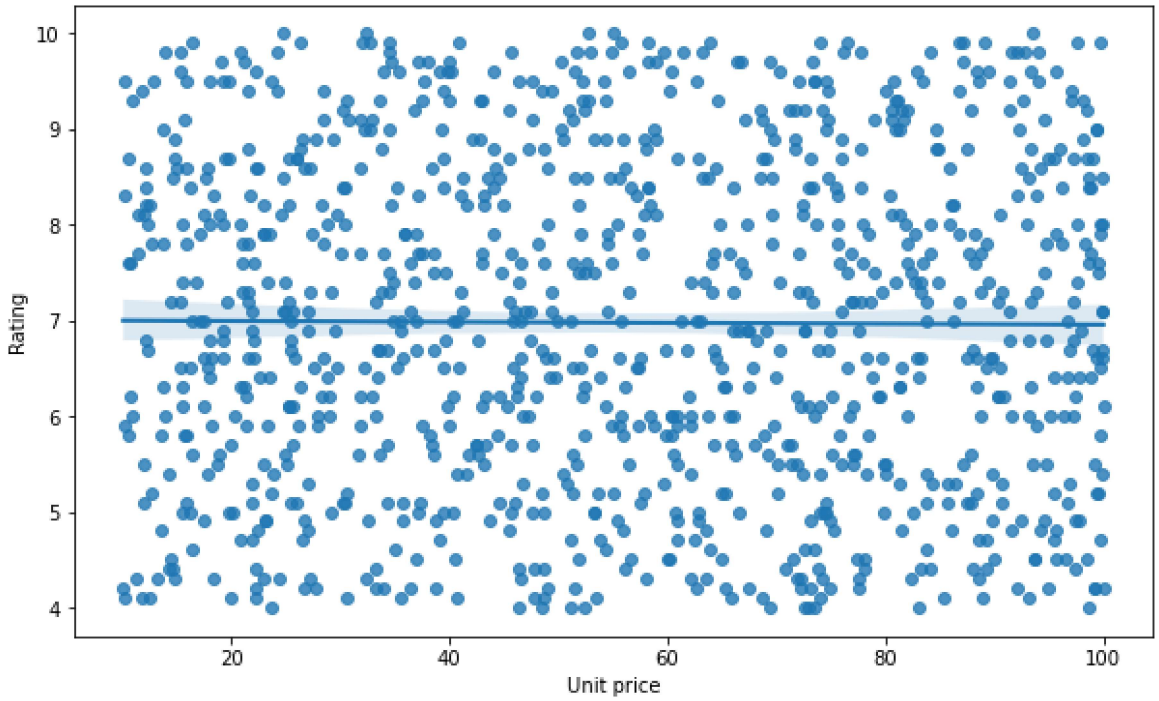
Out[7]:
```
(995, 17)
```

In [8]:
```
supermarket_sales.dtypes
```

Out[8]:
```
Invoice ID                 object
Branch                     object
City                       object
Customer type              object
Gender                     object
Product line               object
Unit price                float64
Quantity                    int64
Tax 5%                    float64
Total                     float64
Date                       object
Time                       object
Payment                    object
cogs                      float64
gross margin percentage   float64
gross income              float64
Rating                    float64
dtype: object
```

In [10]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="Unit price", y="Rating", data=supermarket_sales)
```

Out[10]:
```
<AxesSubplot:xlabel='Unit price', ylabel='Rating'>
```

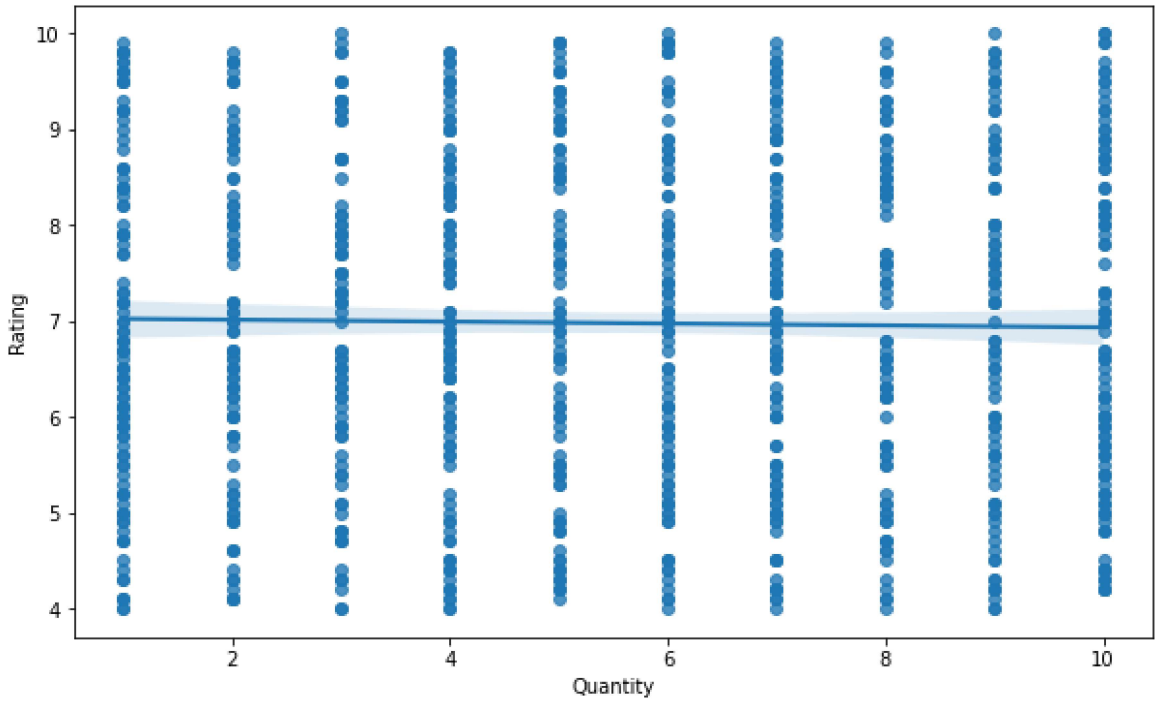


In [11]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(supermarket_sales['Unit price'],
supermarket_sales['Rating'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-valu
```

```
The Pearson Correlation Coefficient is -0.00752477193063086  with a P-value
of P = 0.8126074929543003
```

In [12]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="Quantity", y="Rating", data=supermarket_sales)
```

Out[12]:
```
<AxesSubplot:xlabel='Quantity', ylabel='Rating'>
```

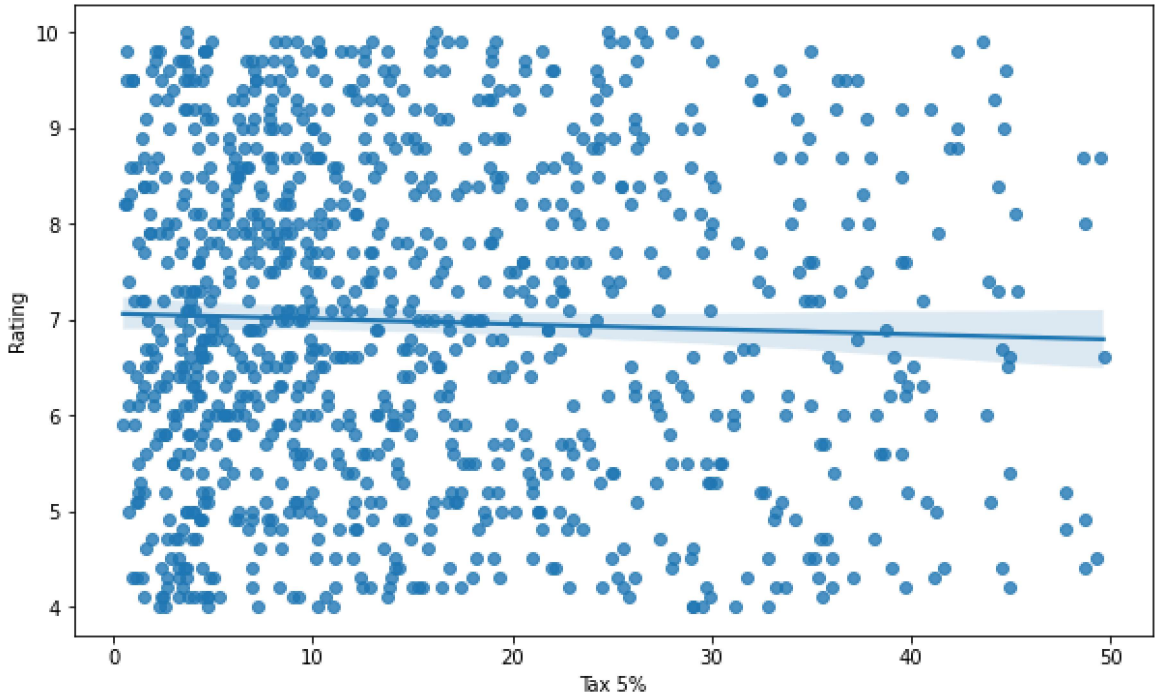


In [13]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(supermarket_sales['Quantity'],
supermarket_sales['Rating'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-valu
```

```
The Pearson Correlation Coefficient is -0.016976449330662065  with a P-valu
e of P = 0.5927438386810999
```

In [14]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="Tax 5%", y="Rating", data=supermarket_sales)
```

Out[14]: <AxesSubplot:xlabel='Tax 5%', ylabel='Rating'>

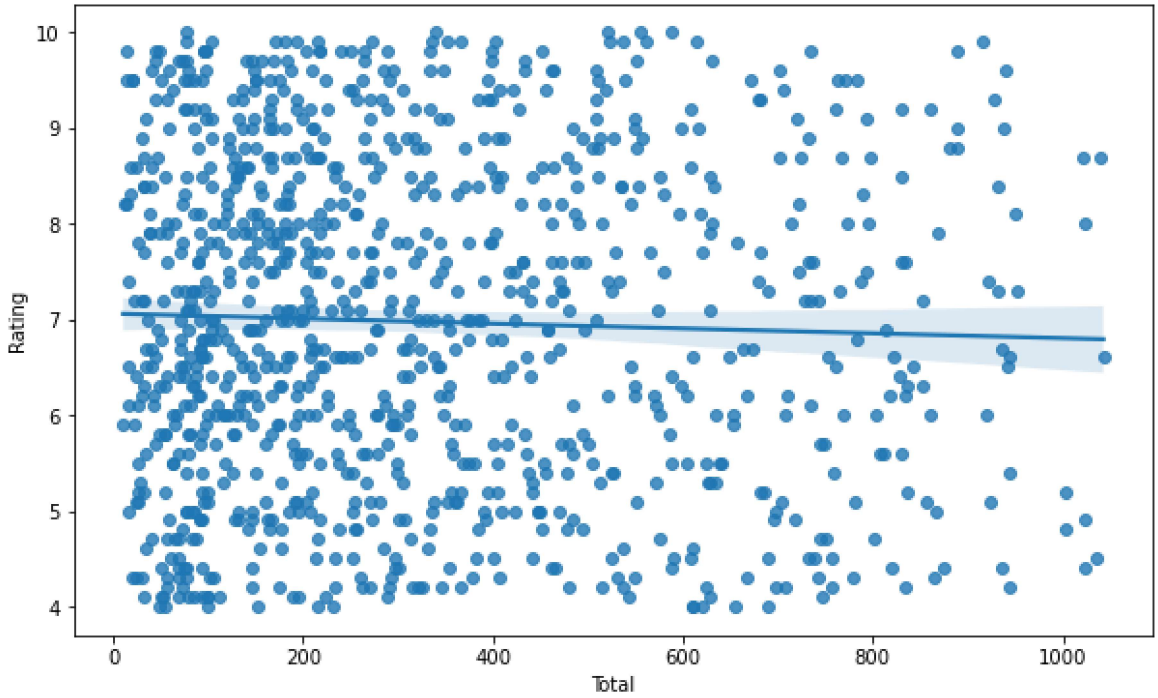


In [15]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(supermarket_sales['Tax 5%'],
supermarket_sales['Rating'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-valu
```

```
The Pearson Correlation Coefficient is -0.03652485423580504  with a P-value
of P = 0.24970673308810218
```

In [16]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="Total", y="Rating", data=supermarket_sales)
```

Out[16]: <AxesSubplot:xlabel='Total', ylabel='Rating'>

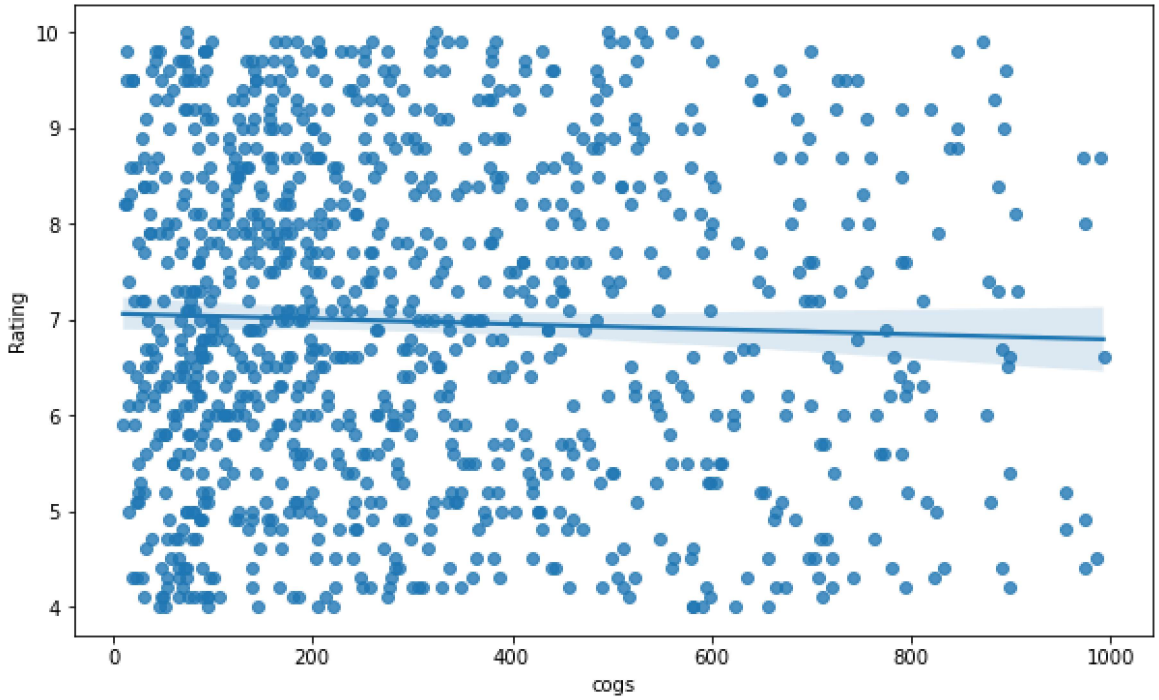


In [17]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(supermarket_sales['Total'],
supermarket_sales['Rating'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-valu
```

```
The Pearson Correlation Coefficient is -0.03652485423580504  with a P-value
of P = 0.24970673308810218
```

In [21]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="cogs", y="Rating", data=supermarket_sales)
```

Out[21]: <AxesSubplot:xlabel='cogs', ylabel='Rating'>

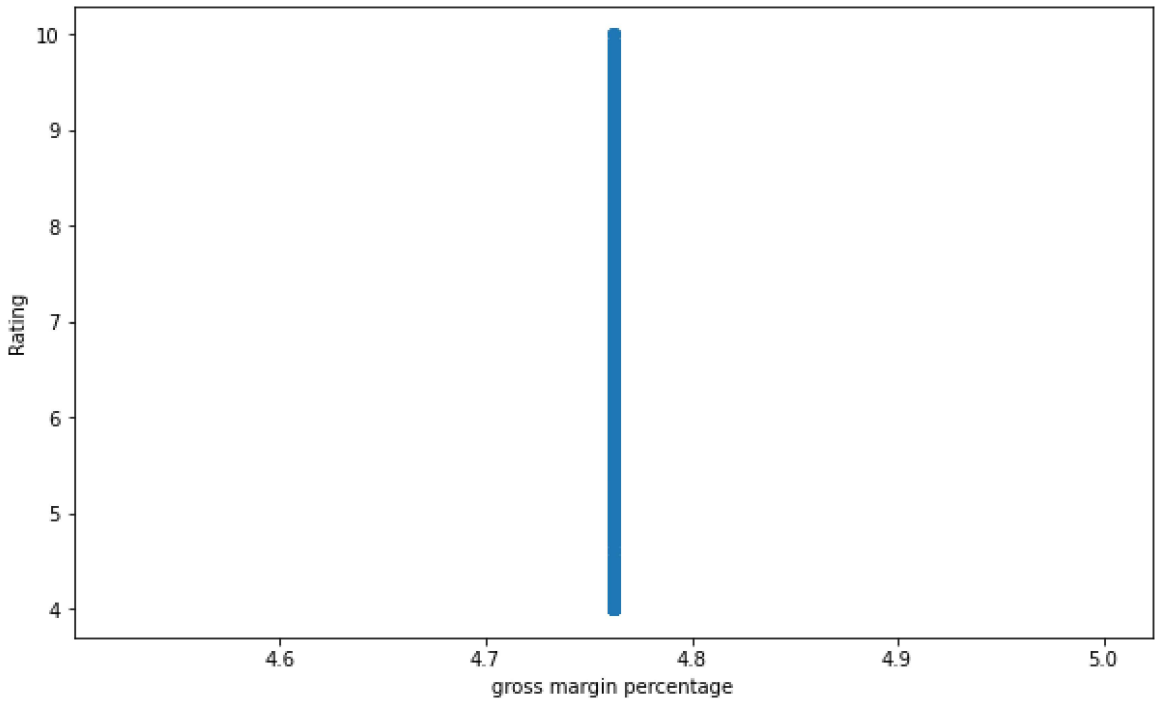


In [22]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(supermarket_sales['cogs'],
supermarket_sales['Rating'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-valu
```

```
The Pearson Correlation Coefficient is -0.03652485423580504  with a P-value
of P = 0.24970673308810218
```

In [23]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="gross margin percentage", y="Rating", data=supermarket_sales)
```

Out[23]: <AxesSubplot:xlabel='gross margin percentage', ylabel='Rating'>

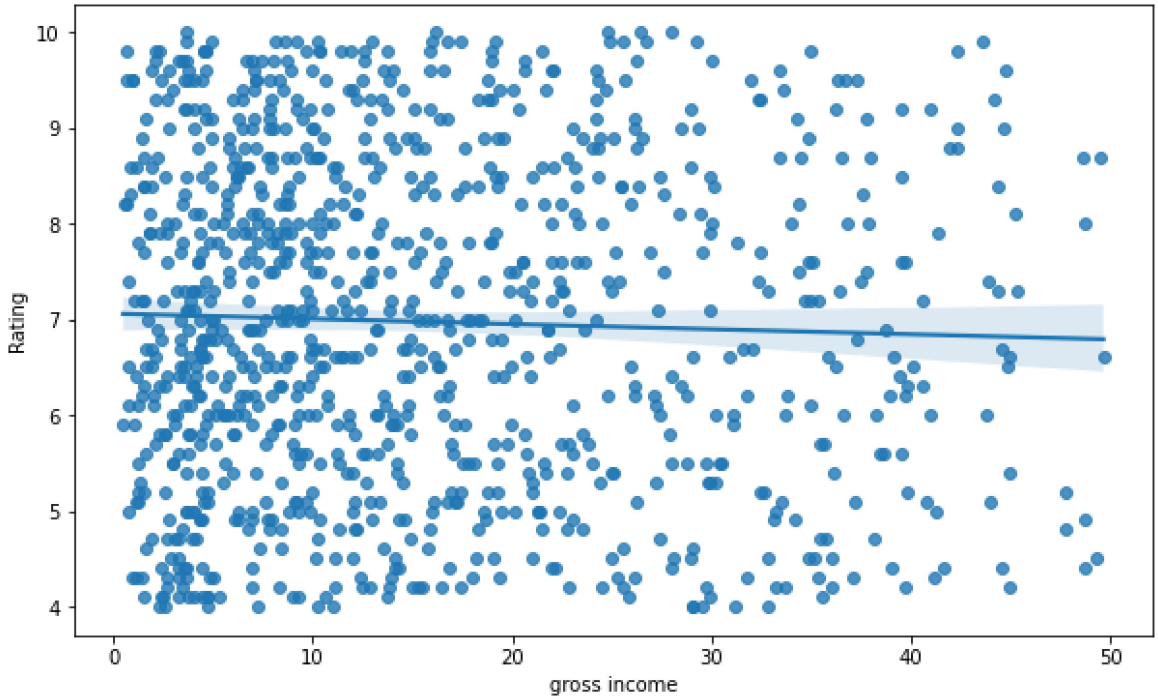


In [24]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(supermarket_sales['gross margin percen
supermarket_sales['Rating'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-valu
```

```
The Pearson Correlation Coefficient is nan  with a P-value of P = nan
```

In [25]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="gross income", y="Rating", data=supermarket_sales)
```

Out[25]: <AxesSubplot:xlabel='gross income', ylabel='Rating'>

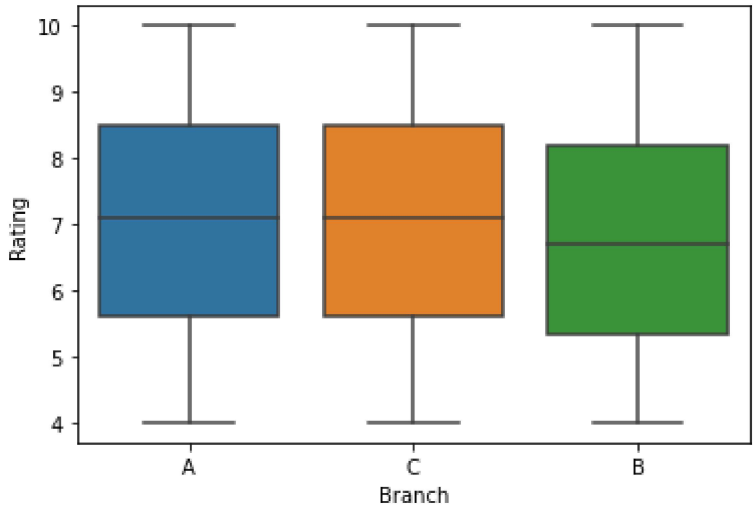


In [26]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(supermarket_sales['gross income'],
supermarket_sales['Rating'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-valu
```

```
The Pearson Correlation Coefficient is -0.03652485423580504  with a P-value
of P = 0.24970673308810218
```

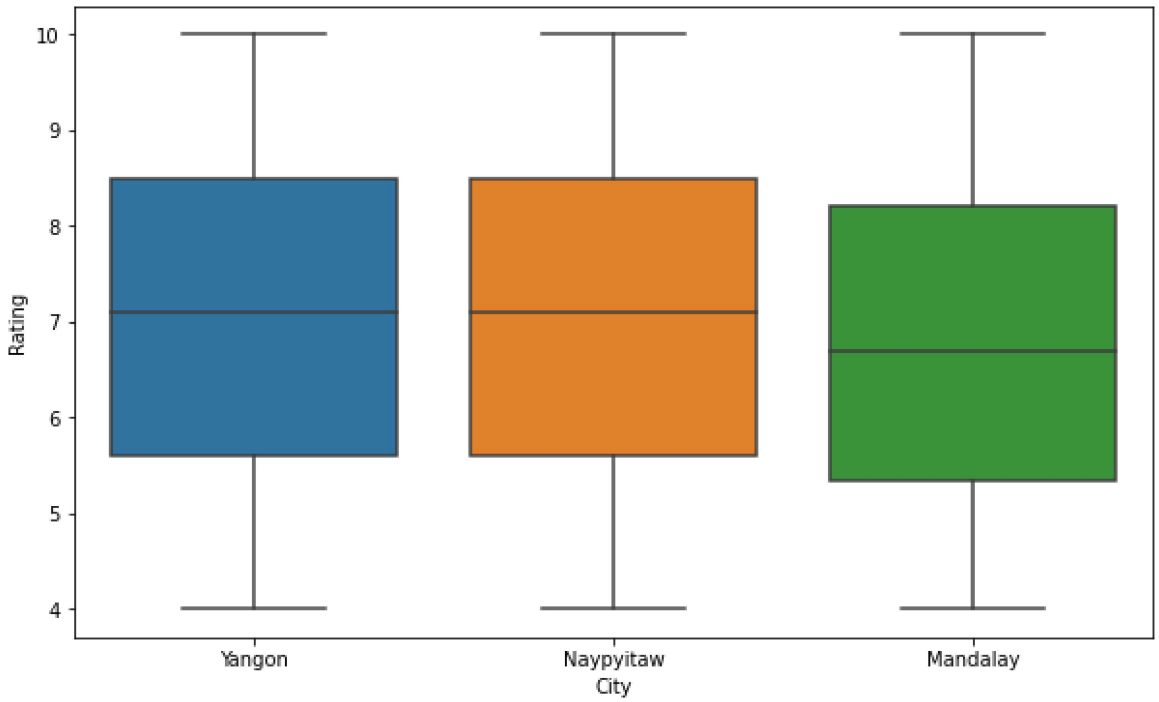
In [27]:
```
sns.boxplot(x="Branch", y="Rating", data=supermarket_sales)
```

Out[27]: <AxesSubplot:xlabel='Branch', ylabel='Rating'>



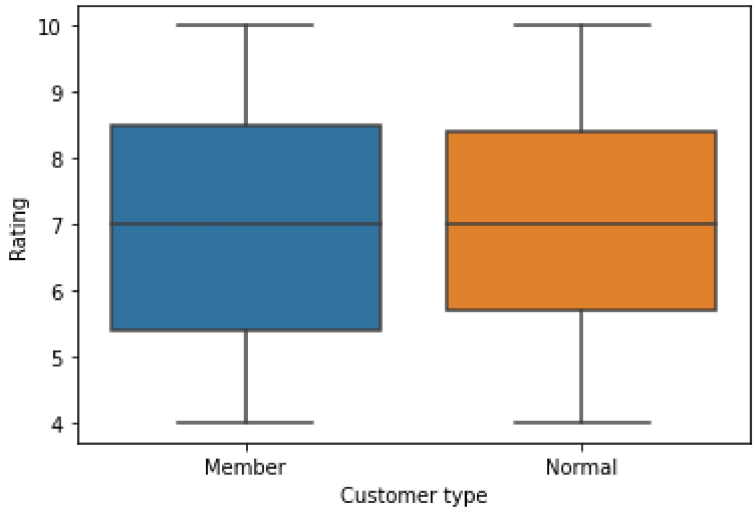
In [28]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="City", y="Rating", data=supermarket_sales)
```

Out[28]: <AxesSubplot:xlabel='City', ylabel='Rating'>



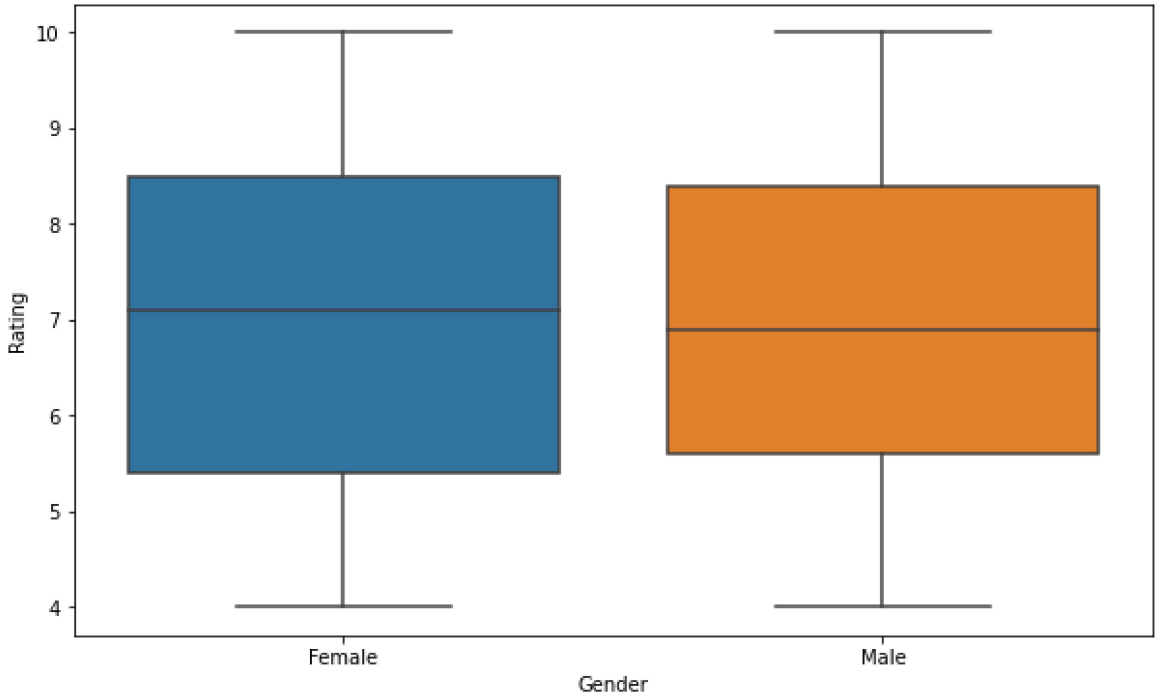
In [29]:
```
sns.boxplot(x="Customer type", y="Rating", data=supermarket_sales)
```

Out[29]: <AxesSubplot:xlabel='Customer type', ylabel='Rating'>



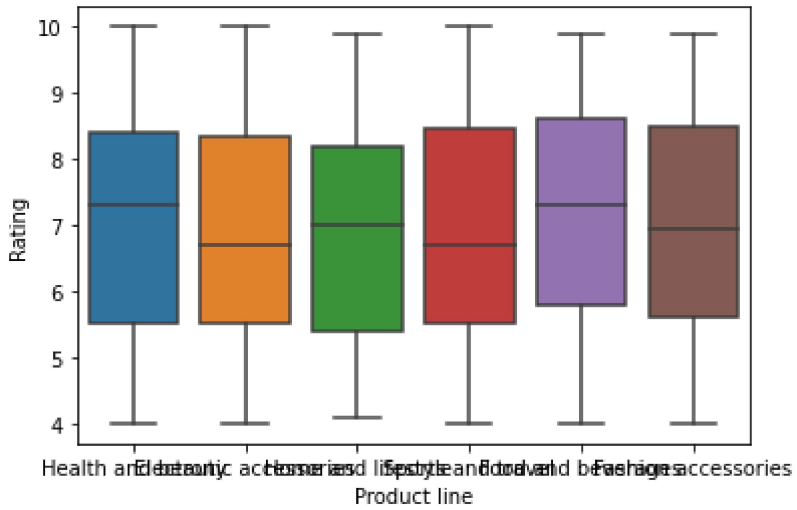
In [30]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="Gender", y="Rating", data=supermarket_sales)
```

Out[30]: <AxesSubplot:xlabel='Gender', ylabel='Rating'>



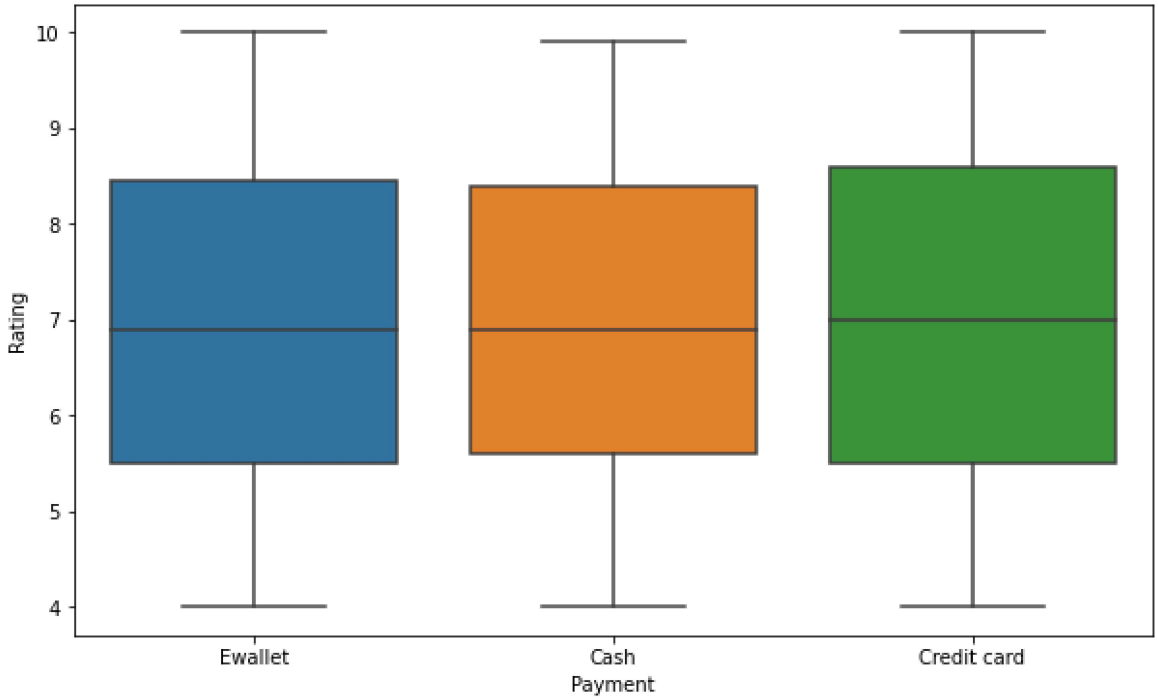
In [31]:
```
sns.boxplot(x="Product line", y="Rating", data=supermarket_sales)
```

Out[31]: <AxesSubplot:xlabel='Product line', ylabel='Rating'>



In [33]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="Payment", y="Rating", data=supermarket_sales)
```

Out[33]: <AxesSubplot:xlabel='Payment', ylabel='Rating'>



In [34]:
```
supermarket_sales.shape
```

Out[34]: (995, 17)

In [35]:
```
supermarket_sales.describe()
```

Out[35]:

	Unit price	Quantity	Tax 5%	Total	cogs	gross margin percentage	gross income
count	995.000000	995.000000	995.000000	995.000000	995.000000	9.950000e+02	995.000000
mean	55.662302	5.520603	15.403206	323.467327	308.064121	4.761905e+00	15.403206
std	26.526430	2.923974	11.721070	246.142477	234.421406	6.131513e-14	11.721070
min	10.080000	1.000000	0.508500	10.678500	10.170000	4.761905e+00	0.508500
25%	32.755000	3.000000	5.966000	125.286000	119.320000	4.761905e+00	5.966000
50%	55.390000	5.000000	12.096000	254.016000	241.920000	4.761905e+00	12.096000
75%	77.940000	8.000000	22.460500	471.670500	449.210000	4.761905e+00	22.460500
max	99.960000	10.000000	49.650000	1042.650000	993.000000	4.761905e+00	49.650000

In [36]:
```
supermarket_sales.describe(include=['object'])
```

Out[36]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Date	Time	Payment
count	995	995	995	995	995	995	995	995	995
unique	995	3	3	2	2	6	89	505	3
top	750-67-8428	A	Yangon	Member	Female	Fashion accessories	02-07-2019	14:42	Cash
freq	1	337	337	499	498	178	20	7	344

In [40]:
```
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
supermarket_sales.Branch = labelencoder.fit_transform(supermarket_sales.Branc
supermarket_sales.City= labelencoder.fit_transform(supermarket_sales.City)
supermarket_sales. Gender= labelencoder.fit_transform(supermarket_sales.Gende
supermarket_sales.Payment = labelencoder.fit_transform(supermarket_sales.Paym
```

In [41]:
```
supermarket_sales.head(10)
```

Out[41]:

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total
0	750-67-8428	0	2	Member	0	Health and beauty	74.69	7	26.1415	548.9715
1	226-31-3081	2	1	Normal	0	Electronic accessories	15.28	5	3.8200	80.2200
2	631-41-3108	0	2	Normal	1	Home and lifestyle	46.33	7	16.2155	340.5255
3	123-19-1176	0	2	Member	1	Health and beauty	58.22	8	23.2880	489.0480
4	373-73-7910	0	2	Normal	1	Sports and travel	86.31	7	30.2085	634.3785
5	699-14-3026	2	1	Normal	1	Electronic accessories	85.39	7	29.8865	627.6165
6	355-53-5943	0	2	Member	0	Electronic accessories	68.84	6	20.6520	433.6920
7	315-22-5665	2	1	Normal	0	Home and lifestyle	73.56	10	36.7800	772.3800
9	692-92-5582	1	0	Member	0	Food and beverages	54.84	3	8.2260	172.7460
10	351-62-0822	1	0	Member	0	Fashion accessories	14.48	4	2.8960	60.8160

In [33]:
```
import pandas as pd
```

In [34]:
```
supermarket_sales=pd.read_csv(r'C:\Users\18F17897\Desktop\supermarket_sales.c
```

In [35]:
```
supermarket_sales.head()
```

Out[35]:

	Invoice	Branch	City	Customer	Gender	Product	UnitPrice	Quantity	Tax 5%	
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	5
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	3
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	4
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	6

In [36]:
```
from sklearn.preprocessing import LabelEncoder
```

In [37]:
```
labelencoder = LabelEncoder()
```

In [38]:
```
supermarket_sales.Branch = labelencoder.fit_transform(supermarket_sales.Branc
```

In [39]:
```
supermarket_sales.City = labelencoder.fit_transform(supermarket_sales.City)
```

In [40]:
```
supermarket_sales.Customer = labelencoder.fit_transform(supermarket_sales.Cus
```

In [41]:
```
supermarket_sales.Gender = labelencoder.fit_transform(supermarket_sales.Gende
```

In [42]:
```
supermarket_sales.Product = labelencoder.fit_transform(supermarket_sales.Prod
```

In [43]:
```
supermarket_sales.Payment = labelencoder.fit_transform(supermarket_sales.Paym
```

In [44]:
```
supermarket_sales.Invoice = labelencoder.fit_transform(supermarket_sales.Invo
```

In [45]:
```
supermarket_sales.head()
```

Out[45]:

	Invoice	Branch	City	Customer	Gender	Product	UnitPrice	Quantity	Tax 5%	Total
0	813	0	2	0	0	3	74.69	7	26.1415	548.9715
1	142	2	1	1	0	0	15.28	5	3.8200	80.2200
2	653	0	2	1	1	4	46.33	7	16.2155	340.5255
3	18	0	2	0	1	3	58.22	8	23.2880	489.0480
4	339	0	2	1	1	5	86.31	7	30.2085	634.3785

In [46]:
```
import scipy.stats as stats
supermarket_sales = stats.zscore(supermarket_sales)
```

In [47]:
```
supermarket_sales
```

Out[47]:

	Invoice	Branch	City	Customer	Gender	Product	UnitPrice	Quantity	
0	1.085996	-1.207766	1.210174	-0.998002	-0.996024	0.319617	0.718160	0.509930	0.
1	-1.238417	1.232166	-0.009759	1.002002	-0.996024	-1.430109	-1.525303	-0.174540	-0.
2	0.531740	-1.207766	1.210174	1.002002	0.996024	0.902859	-0.352781	0.509930	0.
3	-1.667966	-1.207766	1.210174	-0.998002	0.996024	0.319617	0.096214	0.852165	0.
4	-0.555989	-1.207766	1.210174	1.002002	0.996024	1.486101	1.156959	0.509930	1.
...	...	...	...	...	...	...	...	...	
995	-1.200312	1.232166	-0.009759	1.002002	0.996024	0.319617	-0.578600	-1.543480	-1.
996	-0.864294	0.012200	-1.229693	1.002002	-0.996024	0.902859	1.574989	1.536635	2.
997	0.923184	-1.207766	1.210174	-0.998002	0.996024	-0.263625	-0.899958	-1.543480	-1.
998	-0.663376	-1.207766	1.210174	1.002002	0.996024	0.902859	0.383208	-1.543480	-1.
999	1.505153	-1.207766	1.210174	-0.998002	-0.996024	-0.846867	1.233617	0.509930	1.

1000 rows × 15 columns

In [48]:
```
x_train=supermarket_sales.iloc[:,0:5]
```

In [49]:
```
y_train=supermarket_sales.iloc[:,6]
```

In [50]:
```
x_test=supermarket_sales.iloc[:,0:5]
```

In [51]:
```
y_test=supermarket_sales.iloc[:,6]
```

In [52]:
```
x_train
```

Out[52]:

	Invoice	Branch	City	Customer	Gender
0	1.085996	-1.207766	1.210174	-0.998002	-0.996024
1	-1.238417	1.232166	-0.009759	1.002002	-0.996024
2	0.531740	-1.207766	1.210174	1.002002	0.996024
3	-1.667966	-1.207766	1.210174	-0.998002	0.996024
4	-0.555989	-1.207766	1.210174	1.002002	0.996024
...	...	...	...	...	...
995	-1.200312	1.232166	-0.009759	1.002002	0.996024
996	-0.864294	0.012200	-1.229693	1.002002	-0.996024
997	0.923184	-1.207766	1.210174	-0.998002	0.996024
998	-0.663376	-1.207766	1.210174	1.002002	0.996024
999	1.505153	-1.207766	1.210174	-0.998002	-0.996024

1000 rows × 5 columns

In [53]:
```
from sklearn.linear_model import LinearRegression
```

In [54]:
```
rg = LinearRegression()
```

In [55]:
```
mdl=rg.fit(x_train,y_train)
```

In [56]:
```
y_pred1 = rg.predict(x_test)
```

In [57]:
```
print('The R-square for Multiple Linear Regression is: '),
rg.score(x_train,y_train)
```

```
The R-square for Multiple Linear Regression is:
```

Out[57]:
```
0.0014621156386838141
```

In [59]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [61]:
```
mse1 =mean_squared_error(y_test, y_pred1)
print('The mean square error for Multiple Linear Regression: ', mse1)
```

```
The mean square error for Multiple Linear Regression:  0.9985378843613162
```

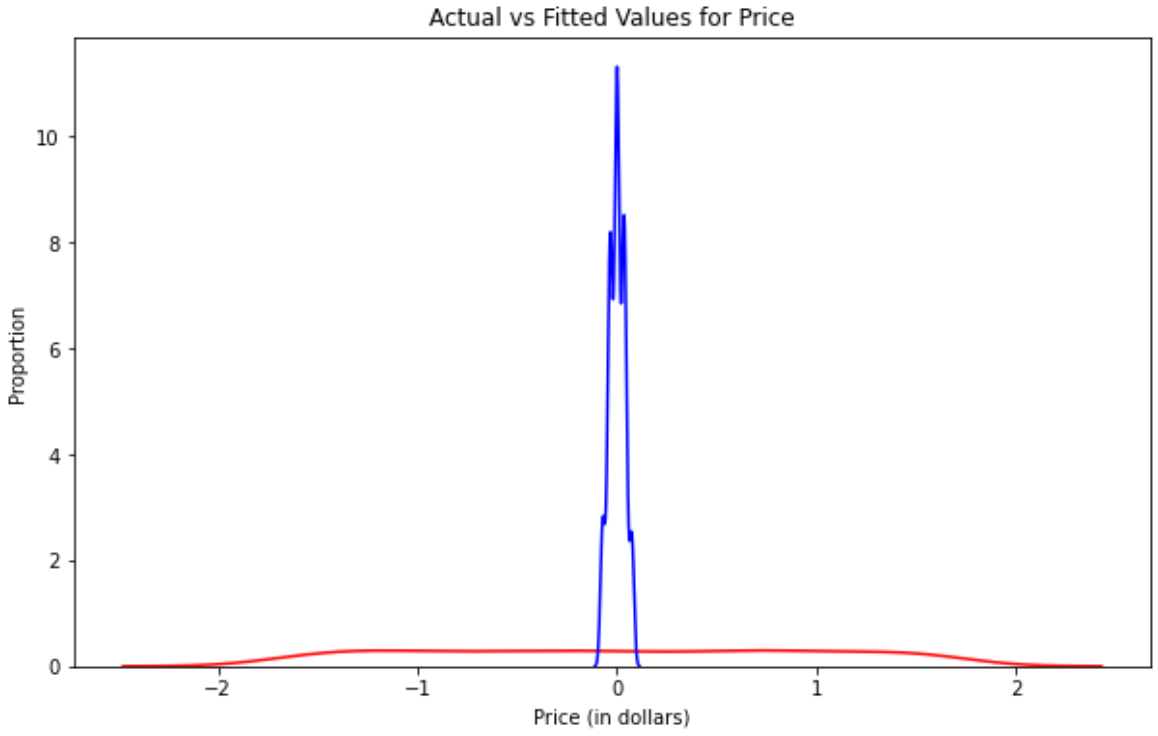
In [62]:
```
mae1= mean_absolute_error(y_test, y_pred1)
print('The mean absolute error for Multiple Linear Regression:', mae1)
```

```
The mean absolute error for Multiple Linear Regression: 0.8663001954009701
```

In [63]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,StandardScaler
from sklearn.linear_model import LinearRegression,Lasso
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.ensemble import RandomForestRegressor
import warnings
warnings.filterwarnings("ignore")
```

```
Matplotlib is building the font cache; this may take a moment.
```

In [65]:
```
plt.figure(figsize=(10,6))
ax1 = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
sns.distplot(y_pred1, hist=False, color="b", label="Fitted Values" , ax=ax1)
plt.title('Actual vs Fitted Values for Price')
plt.xlabel('Price (in dollars)')
plt.ylabel('Proportion')
plt.show()
plt.close()
```



In [66]:
```
rf = RandomForestRegressor()
model=rf.fit(x_train,y_train)
```

In [67]:
```
y_pred2 = rf.predict(x_test)
```

In [68]:
```
print('The R-square for Random Forest is: ', rf.score(x_train,y_train))
```

```
The R-square for Random Forest is:  0.8173197188973209
```

In [69]:
```
mse2 = mean_squared_error(y_test, y_pred2)
print('The mean square error of price and predicted value is: ', mse2)
```

```
The mean square error of price and predicted value is:  0.18268028110267906
```

In [70]:
```
mae2= mean_absolute_error(y_test, y_pred2)
print('The mean absolute error of price and predicted value is: ', mae2)
```

```
The mean absolute error of price and predicted value is:  0.357488390521206
5
```

In [71]:
```
plt.figure(figsize=(10,6))
ax1 = sns.distplot(y_test, hist=False, color="r", label="Actual Value")
sns.distplot(y_pred2, hist=False, color="b", label="Fitted Values" , ax=ax1)
plt.title('Actual vs Fitted Values for Price')
plt.xlabel('Price (in dollars)')
plt.ylabel('Proportion')
plt.show()
plt.close()
```

In [72]:
```
LassoModel=Lasso()
lm=LassoModel.fit(x_train,y_train)
```

In [73]:
```
y_pred3 = lm.predict(x_test)
```

In [74]:
```
print('The R-square for LASSO is: ', lm.score(x_train,y_train))
```

```
The R-square for LASSO is:  0.0
```

In [75]:
```
mae3= mean_absolute_error(y_test, y_pred3)
print('The mean absolute error of price and predicted value is: ', mae3)
```

```
The mean absolute error of price and predicted value is:  0.867743959651284
5
```

In [76]:
```
mse3 = mean_squared_error(y_test, y_pred3)
print('The mean square error of price and predicted value is: ', mse3)
```

```
The mean square error of price and predicted value is:  1.0
```

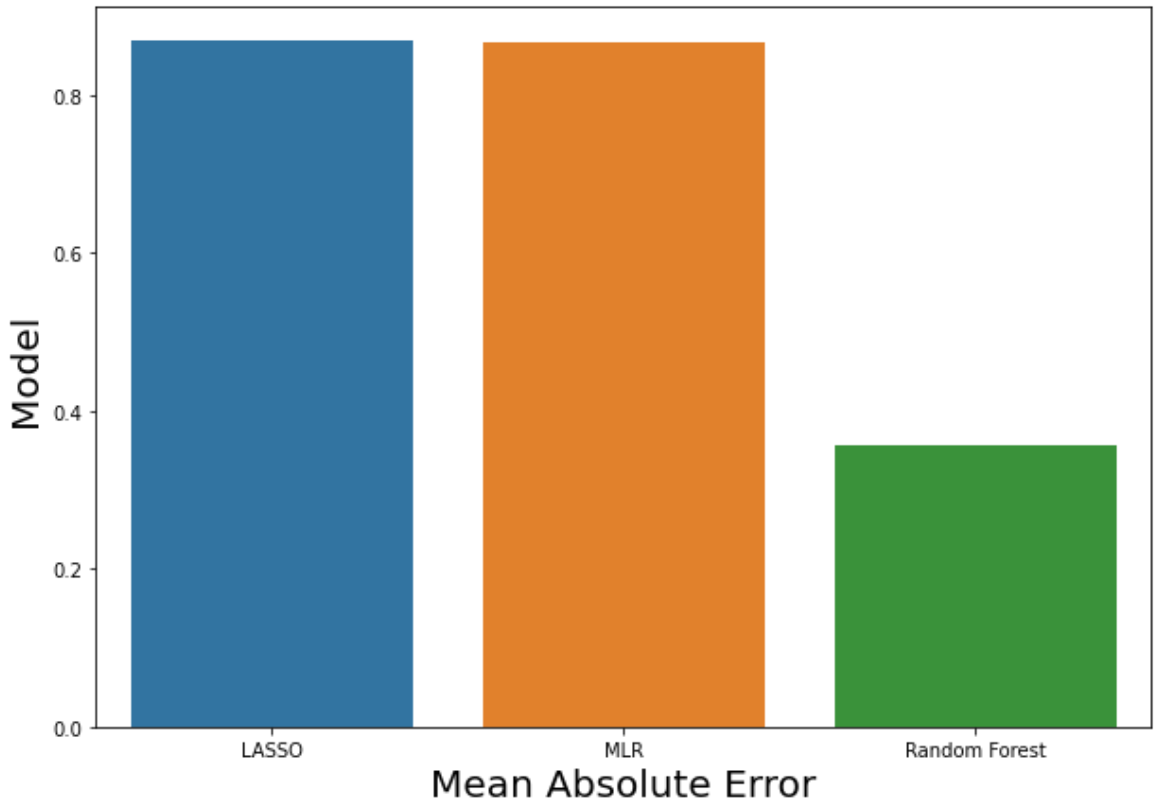
In [77]:
```
scores = [('MLR', mae1),
('Random Forest', mae2),
('LASSO', mae3)
]
```

In [78]:
```
mae = pd.DataFrame(data = scores, columns=['Model', 'MAE Score'])
mae
```

Out[78]:

	Model	MAE Score
0	MLR	0.866300
1	Random Forest	0.357488
2	LASSO	0.867744

In [79]:
```
mae.sort_values(by=(['MAE Score']), ascending=False, inplace=True)
f, axe = plt.subplots(1,1, figsize=(10,7))
sns.barplot(x = mae['Model'], y=mae['MAE Score'], ax = axe)
axe.set_xlabel('Mean Absolute Error', size=20)
axe.set_ylabel('Model', size=20)
plt.show()
```



In [ ]:
```

```