In [1]:
```python
print("test")
```

test

In [101]:
```python
# First adding all necessary libraries:
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 [102]:
```python
# Loading the data of “laptopPrice” dataset
df = pd.read_csv('D:laptopPrice.csv')
```

In [103]:
```python
# display the first five records
df.head(5)
```

Out[103]:

	brand	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	os	os_bit	graphic_card_gb
0	ASUS	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GE
1	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GE
2	Lenovo	Intel	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	64-bit	0 GE
3	ASUS	Intel	Core i5	10th	8 GB	DDR4	512 GB	0 GB	Windows	32-bit	2 GE
4	ASUS	Intel	Celeron Dual	Not Available	4 GB	DDR4	0 GB	512 GB	Windows	64-bit	0 GE

In [104]:
```python
#The shape function is used to display the total number of rows and columns of the laptopPrice dataset.
print(df.shape)
```

(823, 19)

In [105]:
```
#Checking for null values in each column and displaying the sum of all null values in each column
missing_values = df.isnull().sum()
print("Missing Values:")
print(missing_values)
```

```
Missing Values:
brand                0
processor_brand      0
processor_name       0
processor_gnrtn      0
ram_gb               0
ram_type             0
ssd                  0
hdd                  0
os                   0
os_bit               0
graphic_card_gb      0
weight               0
warranty             0
Touchscreen          0
msoffice             0
Price                0
rating               0
Number of Ratings    0
Number of Reviews    0
dtype: int64
```

In [106]:
```
#Removing the rows with empty values
print(df.dropna())
```

```
      brand processor_brand processor_name processor_gnrtn ram_gb ram_type  \
0      ASUS           Intel        Core i3            10th   4 GB     DDR4   
1    Lenovo           Intel        Core i3            10th   4 GB     DDR4   
2    Lenovo           Intel        Core i3            10th   4 GB     DDR4   
3      ASUS           Intel        Core i5            10th   8 GB     DDR4   
4      ASUS           Intel   Celeron Dual   Not Available   4 GB     DDR4   
..      ...             ...            ...             ...    ...      ...   
818    ASUS             AMD        Ryzen 9   Not Available   4 GB     DDR4   
819    ASUS             AMD        Ryzen 9   Not Available   4 GB     DDR4   
820    ASUS             AMD        Ryzen 9   Not Available   4 GB     DDR4   
821    ASUS             AMD        Ryzen 9   Not Available   4 GB     DDR4   
822  Lenovo             AMD        Ryzen 5            10th   8 GB     DDR4   

          ssd      hdd       os  os_bit graphic_card_gb      weight  \
0        0 GB  1024 GB  Windows  64-bit            0 GB      Casual   
1        0 GB  1024 GB  Windows  64-bit            0 GB      Casual   
2        0 GB  1024 GB  Windows  64-bit            0 GB      Casual   
3      512 GB     0 GB  Windows  32-bit            2 GB      Casual   
4        0 GB   512 GB  Windows  64-bit            0 GB      Casual   
..        ...      ...      ...     ...             ...         ...   
818  1024 GB     0 GB  Windows  64-bit            0 GB      Casual   
819  1024 GB     0 GB  Windows  64-bit            0 GB      Casual   
820  1024 GB     0 GB  Windows  64-bit            4 GB      Casual   
821  1024 GB     0 GB  Windows  64-bit            4 GB      Casual   
822   512 GB     0 GB      DOS  64-bit            0 GB  ThinNlight   

        warranty Touchscreen msoffice   Price   rating  Number of Ratings  \
0    No warranty          No       No   34649  2 stars                  3   
1    No warranty          No       No   38999  3 stars                 65   
2    No warranty          No       No   39999  3 stars                  8   
3    No warranty          No       No   69990  3 stars                  0   
4    No warranty          No       No   26990  3 stars                  0   
..           ...         ...      ...     ...      ...                ...   
818       1 year          No       No  135990  3 stars                  0   
819       1 year          No       No  144990  3 stars                  0   
820       1 year          No       No  149990  3 stars                  0   
821       1 year          No       No  142990  3 stars                  0   
822  No warranty          No       No   57490  4 stars                 18   

     Number of Reviews  
0                    0  
1                    5  
2                    1  
3                    0  
4                    0  
..                 ...  
818                  0  
819                  0  
820                  0  
821                  0  
822                  4  

[823 rows x 19 columns]
```

In [107]:
```
# Checking if there is any duplicates value
df.duplicated().sum()
```

Out[107]: 21

In [108]:
```
df = df.drop_duplicates()
```

In [109]:
```
# Display basic information about the dataset
# info() is a method used to provide a summary of dataFrame
# and understand the dataset .Also getting the structure of dataframe that am going to work on it.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 802 entries, 0 to 822
Data columns (total 19 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   brand              802 non-null    object
 1   processor_brand    802 non-null    object
 2   processor_name     802 non-null    object
 3   processor_gnrtn    802 non-null    object
 4   ram_gb             802 non-null    object
 5   ram_type           802 non-null    object
 6   ssd                802 non-null    object
 7   hdd                802 non-null    object
 8   os                 802 non-null    object
 9   os_bit             802 non-null    object
 10  graphic_card_gb    802 non-null    object
 11  weight             802 non-null    object
 12  warranty           802 non-null    object
 13  Touchscreen        802 non-null    object
 14  msoffice           802 non-null    object
 15  Price              802 non-null    int64 
 16  rating             802 non-null    object
 17  Number of Ratings  802 non-null    int64 
 18  Number of Reviews  802 non-null    int64 
dtypes: int64(3), object(16)
memory usage: 125.3+ KB
```

In [110]:
```
#Checking the data types to see if all the data is in correct format.
# dtypes used to check and understanding the types of data presented in each column of the dataFrame
df.dtypes
```

Out[110]:
```
brand                object
processor_brand      object
processor_name       object
processor_gnrtn      object
ram_gb               object
ram_type             object
ssd                  object
hdd                  object
os                   object
os_bit               object
graphic_card_gb      object
weight               object
warranty             object
Touchscreen          object
msoffice             object
Price                 int64
rating               object
Number of Ratings     int64
Number of Reviews     int64
dtype: object
```

In [111]:
```
# count the number of unique values in each column of a DataFrame.
df.nunique()
```

Out[111]:
```
brand                  8
processor_brand        3
processor_name        11
processor_gnrtn        8
ram_gb                 4
ram_type               6
ssd                    7
hdd                    4
os                     3
os_bit                 2
graphic_card_gb        5
weight                 3
warranty               4
Touchscreen            2
msoffice               2
Price                405
rating                 5
Number of Ratings    282
Number of Reviews    135
dtype: int64
```

In [112]:
```
#For numerical columns only
df.describe()
```

Out[112]:

	Price	Number of Ratings	Number of Reviews
count	802.000000	802.00000	802.000000
mean	76625.543641	299.84414	36.089776
std	45232.984422	1001.78442	118.313553
min	16990.000000	0.00000	0.000000
25%	45990.000000	0.00000	0.000000
50%	63990.000000	17.00000	2.000000
75%	89525.000000	140.25000	18.000000
max	441990.000000	15279.00000	1947.000000

In [113]:
```
df.duplicated().sum()
```

Out[113]: 0

In [114]:
```
# Checking the number of numeric features and cat_features (Categorical) from dataFram:
numeric_features = [feature for feature in df.columns if df[feature].dtype != 'object']
cat_features = [feature for feature in df.columns if df[feature].dtype == 'object']
# Display Numerical and Categorical variables
print(" Numerical features: ", numeric_features)
print("Categorical featues:", cat_features)
```

```
 Numerical features:  ['Price', 'Number of Ratings', 'Number of Reviews']
Categorical featues: ['brand', 'processor_brand', 'processor_name', 'processor_gnrtn', 'ram_gb', 'ram_
type', 'ssd', 'hdd', 'os', 'os_bit', 'graphic_card_gb', 'weight', 'warranty', 'Touchscreen', 'msoffic
e', 'rating']
```

In [115]:
```
# describe () method used to describe numerical column only
df.describe(include = 'object')
```

Out[115]:

	brand	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	os	os_bit	graphic_carc
count	802	802	802	802	802	802	802	802	802	802	
unique	8	3	11	8	4	6	7	4	3	2	
top	ASUS	Intel	Core i5	11th	8 GB	DDR4	512 GB	0 GB	Windows	64-bit	(
freq	243	594	284	328	404	690	389	602	763	693	

In [117]:
```
df.loc[df['Price'] == 1, 'Price'] = 500
```

In [118]:
```
df.describe()
```

Out[118]:

	Price	Number of Ratings	Number of Reviews
count	802.000000	802.00000	802.000000
mean	76625.543641	299.84414	36.089776
std	45232.984422	1001.78442	118.313553
min	16990.000000	0.00000	0.000000
25%	45990.000000	0.00000	0.000000
50%	63990.000000	17.00000	2.000000
75%	89525.000000	140.25000	18.000000
max	441990.000000	15279.00000	1947.000000

In [119]:
```
df['Price'].describe()
```

Out[119]:
```
count       802.000000
mean      76625.543641
std       45232.984422
min       16990.000000
25%       45990.000000
50%       63990.000000
75%       89525.000000
max      441990.000000
Name: Price, dtype: float64
```

In [120]:
```
df.describe(include = 'object')#summary statistics for categorical values
```

Out[120]:

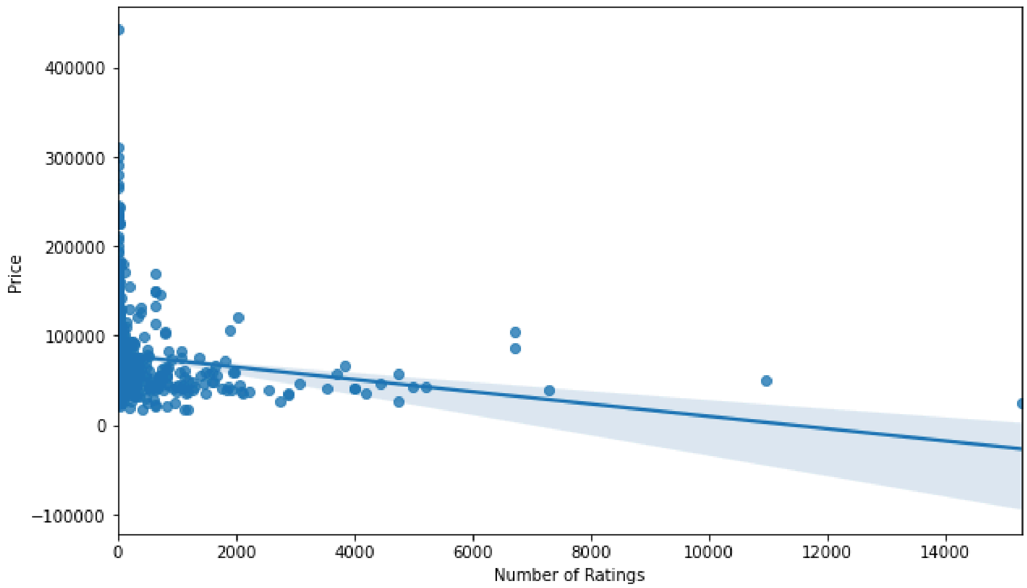
	brand	processor_brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	os	os_bit	graphic_carc
count	802	802	802	802	802	802	802	802	802	802	
unique	8	3	11	8	4	6	7	4	3	2	
top	ASUS	Intel	Core i5	11th	8 GB	DDR4	512 GB	0 GB	Windows	64-bit	(
freq	243	594	284	328	404	690	389	602	763	693	

In [121]:
```
numeric_features = [feature for feature in df.columns if df[feature].dtype != 'object']
cat_features = [feature for feature in df.columns if df[feature].dtype == 'object']
print("Numerical features: ", numeric_features)
print("Categorical featues:", cat_features)
```

```
Numerical features:  ['Price', 'Number of Ratings', 'Number of Reviews']
Categorical featues: ['brand', 'processor_brand', 'processor_name', 'processor_gnrtn', 'ram_gb', 'ram_
type', 'ssd', 'hdd', 'os', 'os_bit', 'graphic_card_gb', 'weight', 'warranty', 'Touchscreen', 'msoffic
e', 'rating']
```

In [122]:
```
import seaborn as sns
plt.figure(figsize=(10,6))
sns.regplot(x="Number of Ratings", y="Price", data=df)
```

Out[122]: <matplotlib.axes._subplots.AxesSubplot at 0x122d257adc0>

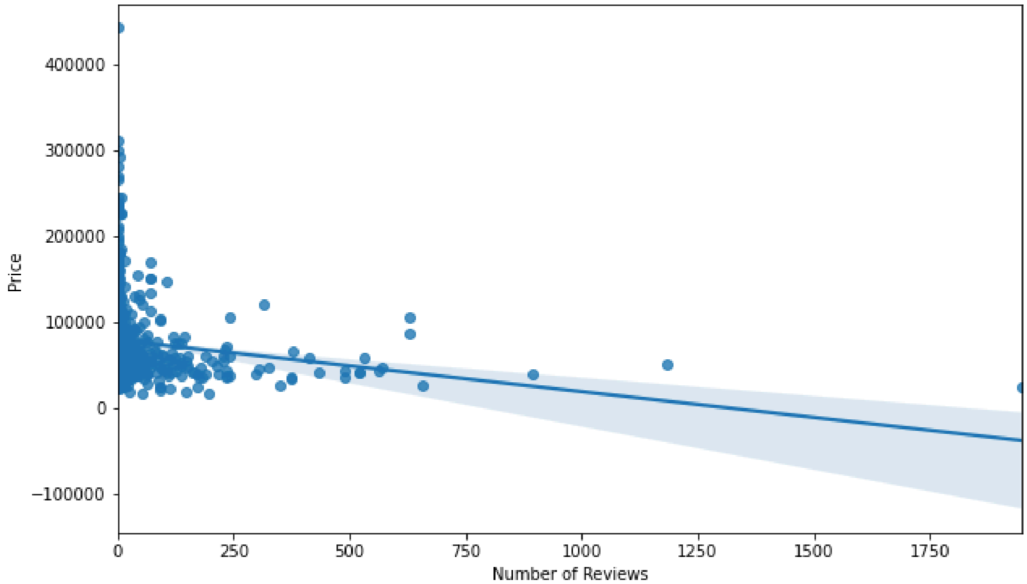


In [123]:
```
from scipy import stats
pearson_coef, p_value = stats.pearsonr(df['Number of Ratings'], df['Price'])
print("The Pearson Correlation Coefficient is", pearson_coef, " with a P-value of P =", p_value)
```

```
The Pearson Correlation Coefficient is -0.15255276430421938  with a P-value of P = 1.4318727176497412e
-05
```

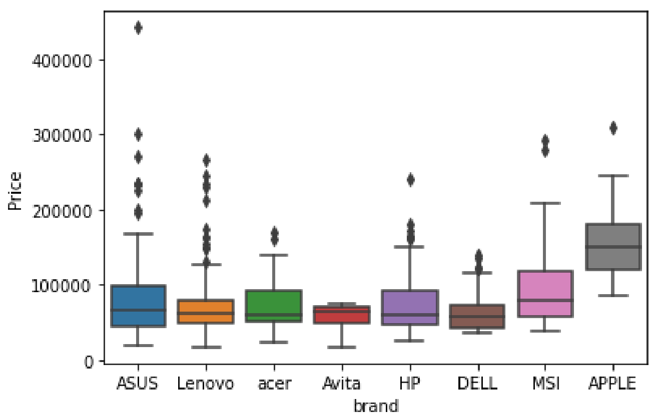
In [124]:
```
plt.figure(figsize=(10,6))
sns.regplot(x="Number of Reviews", y="Price", data=df)
```

Out[124]: <matplotlib.axes._subplots.AxesSubplot at 0x122d26d7250>



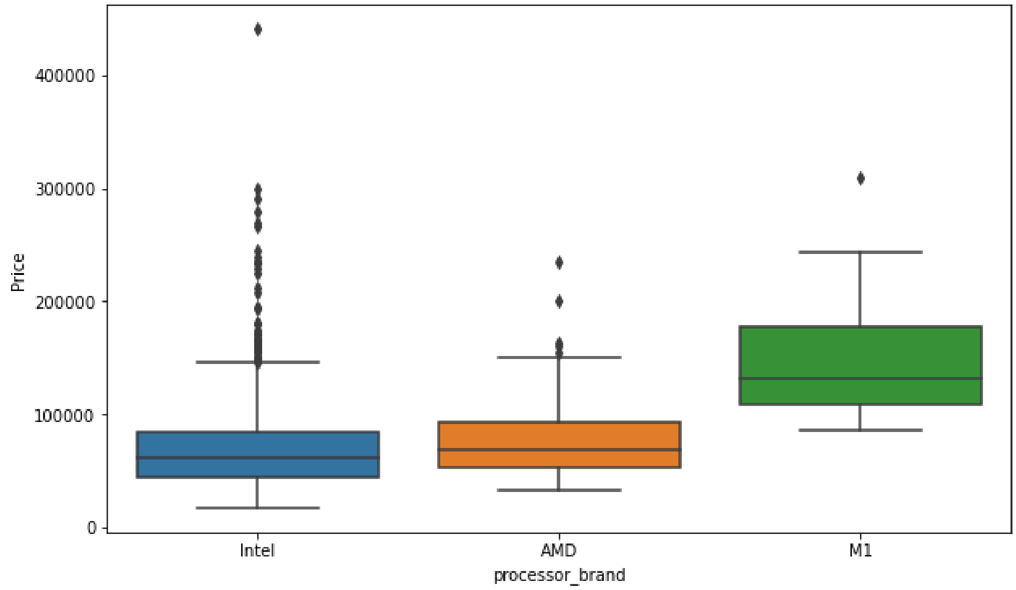
In [125]:
```
# In the given plot below, it is observed that the price range vary for ASUS and Apple Brand.
# This indicates the categories can vary with price hence features can be used for prediction.
sns.boxplot(x="brand", y="Price", data=df)
```

Out[125]: <matplotlib.axes._subplots.AxesSubplot at 0x122d27504f0>



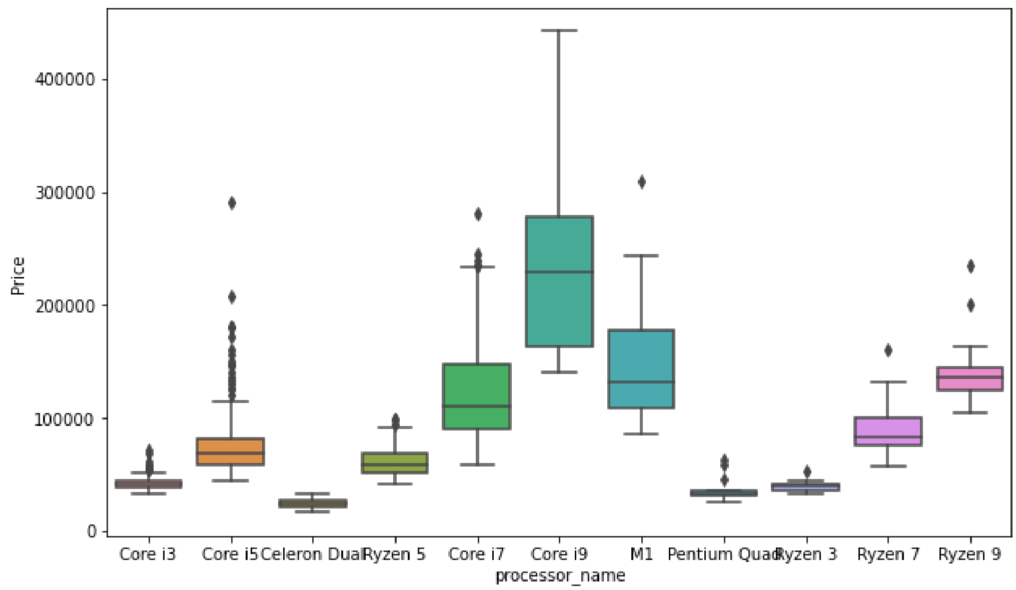
In [126]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="processor_brand", y="Price", data=df)
```

Out[126]: <matplotlib.axes._subplots.AxesSubplot at 0x122d29bfa90>



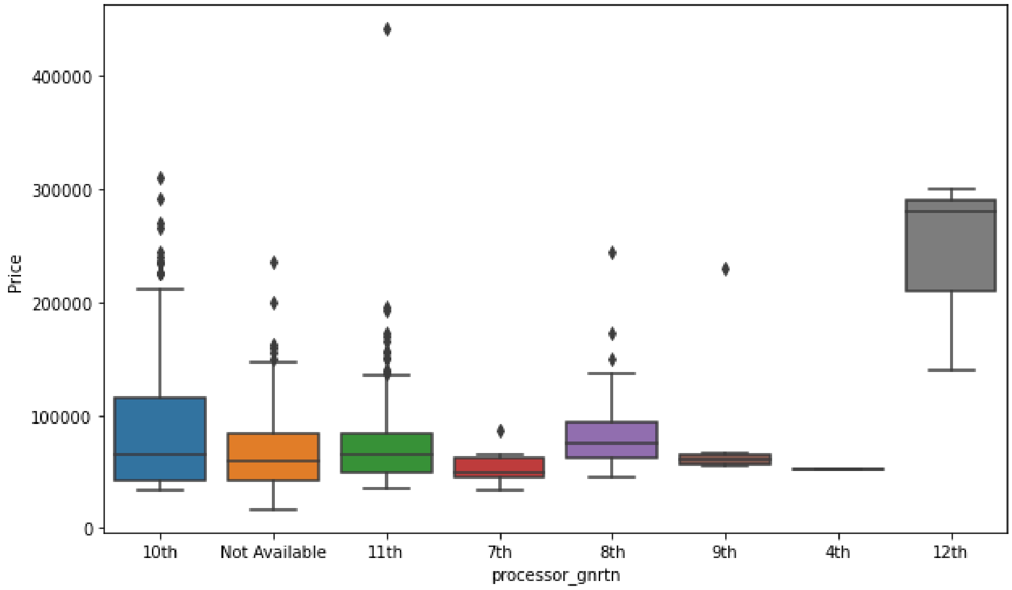
In [127]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="processor_name", y="Price", data=df)
```

Out[127]: <matplotlib.axes._subplots.AxesSubplot at 0x122d2a42250>



In [128]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="processor_gnrtn", y="Price", data=df)
```

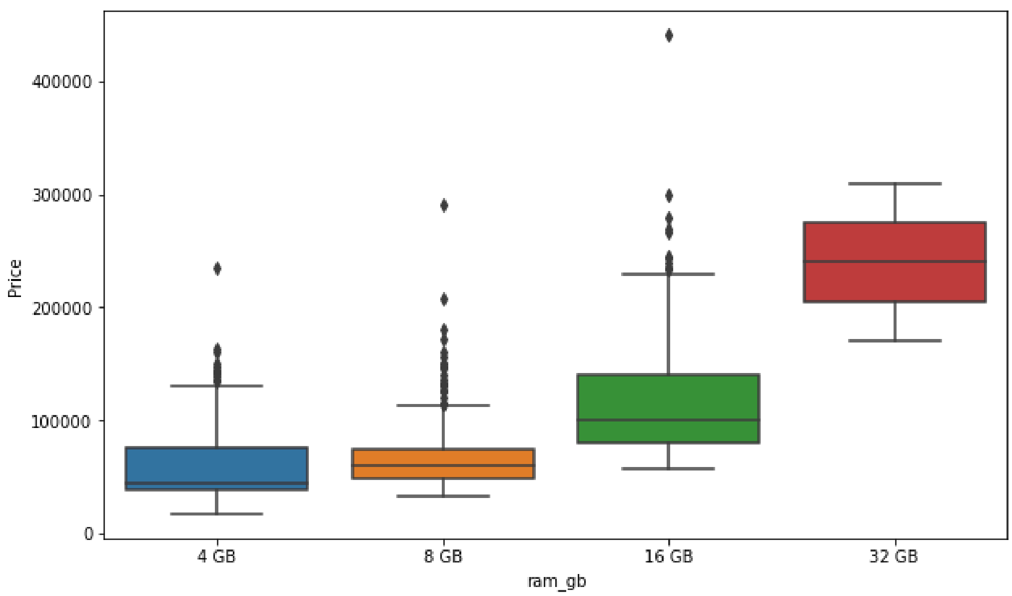
Out[128]: <matplotlib.axes._subplots.AxesSubplot at 0x122d2ca36d0>



In [ ]:
```
# processor_name feature shows a huge difference in price ranges between laptop processor name with Core
# This feature is very important for price prediction as the bigger the difference in range the better t
```

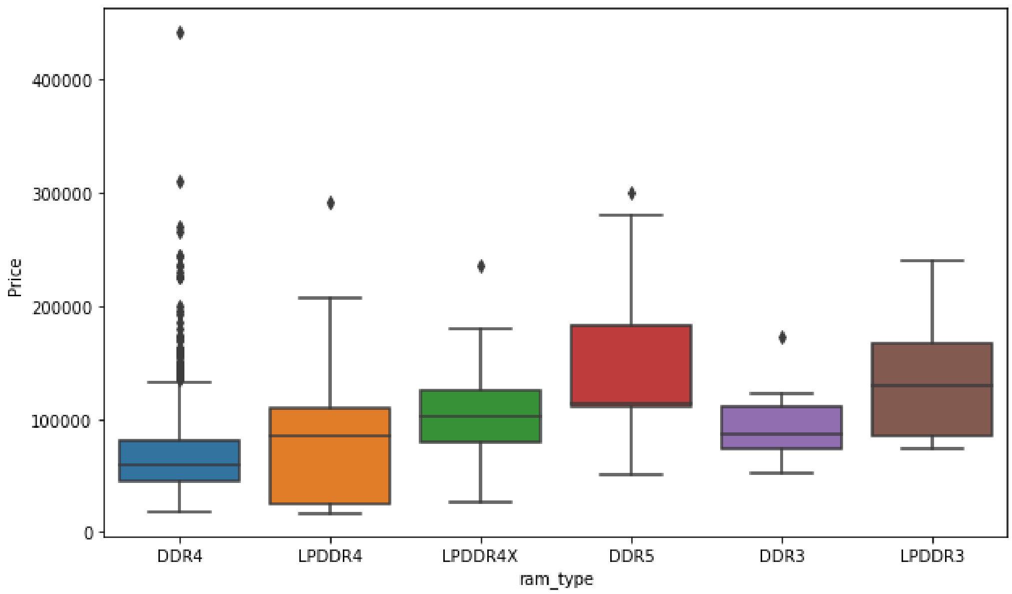
In [129]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="ram_gb", y="Price", data=df)
```

Out[129]: <matplotlib.axes._subplots.AxesSubplot at 0x122d2ca3610>



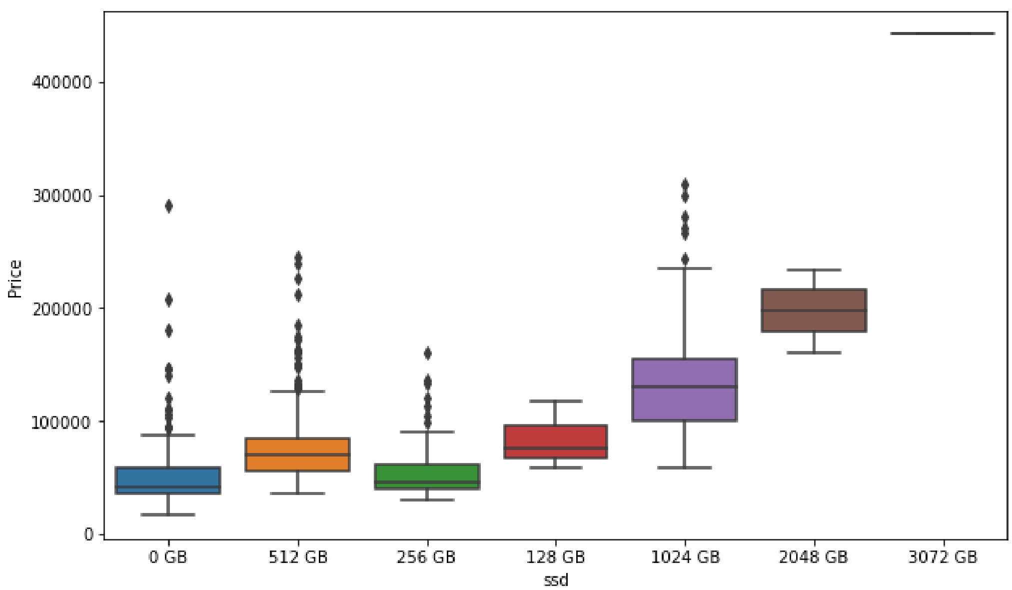
In [130]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="ram_type", y="Price", data=df)
```

Out[130]: <matplotlib.axes._subplots.AxesSubplot at 0x122d2b5c940>



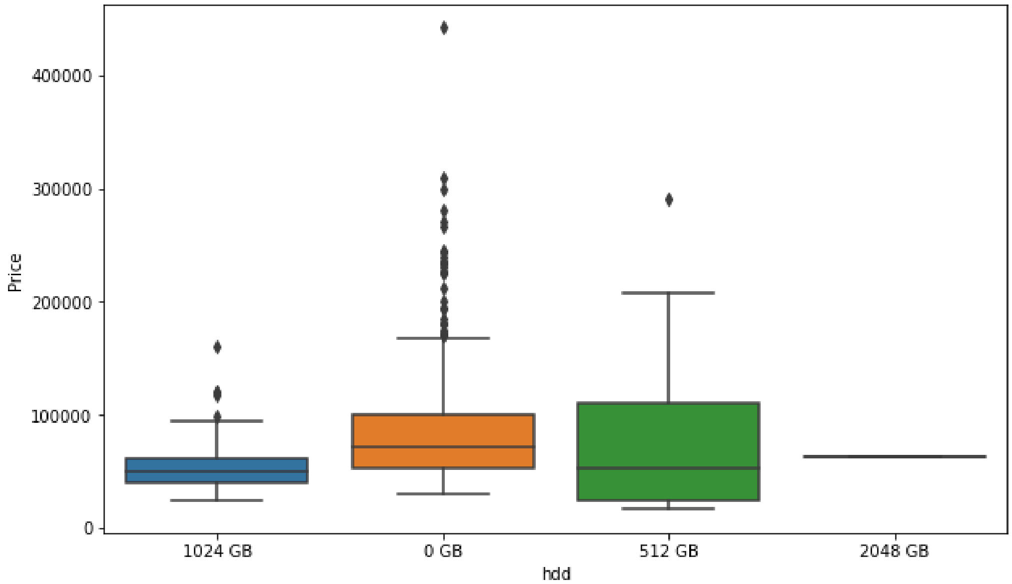
In [131]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="ssd", y="Price", data=df)
```

Out[131]: <matplotlib.axes._subplots.AxesSubplot at 0x122d2d6a6d0>



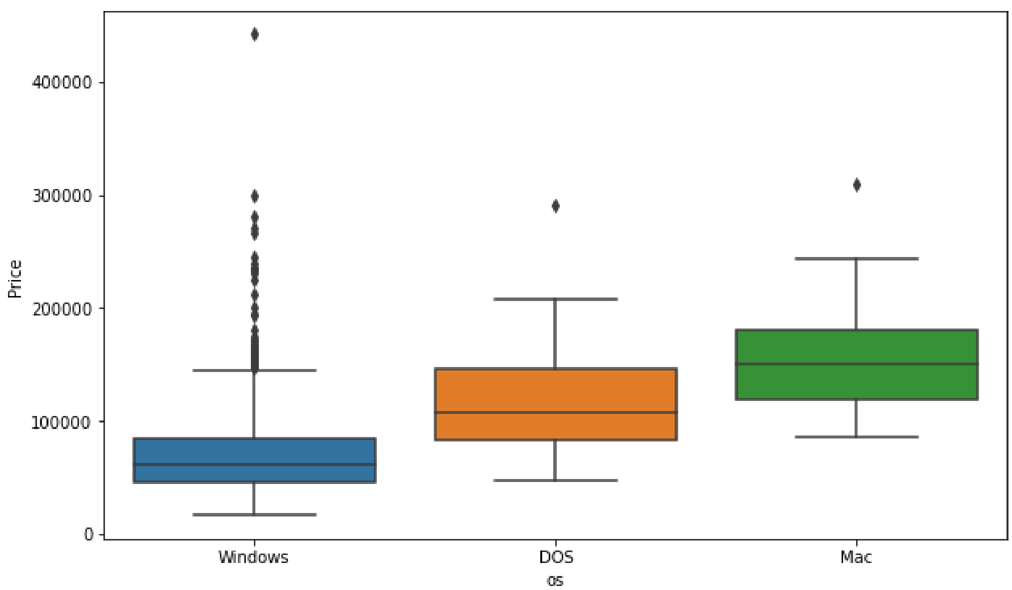
In [132]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="hdd", y="Price", data=df)
```

Out[132]: <matplotlib.axes._subplots.AxesSubplot at 0x122d2747220>



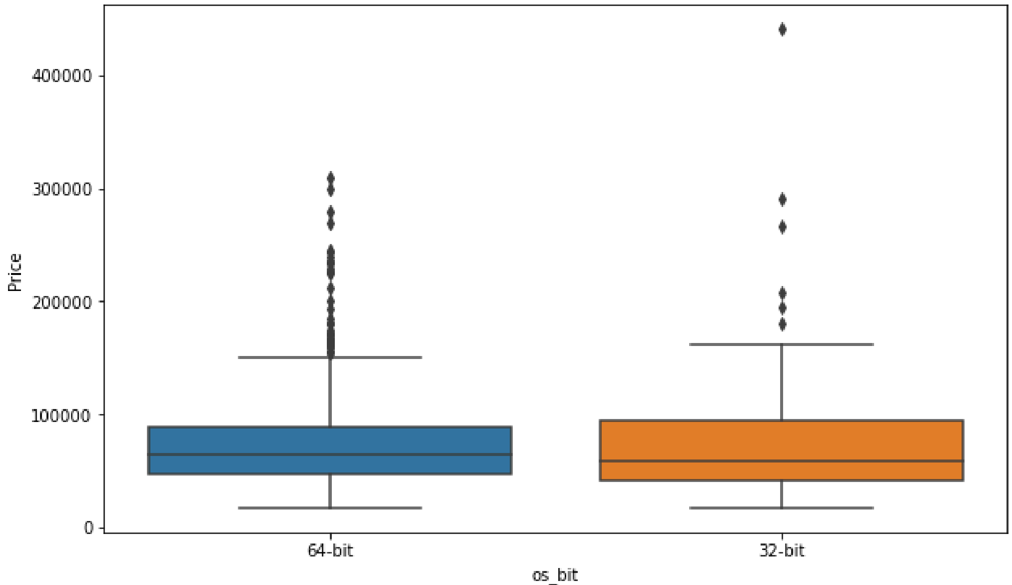
In [133]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="os", y="Price", data=df)
```

Out[133]: <matplotlib.axes._subplots.AxesSubplot at 0x122d2d91700>



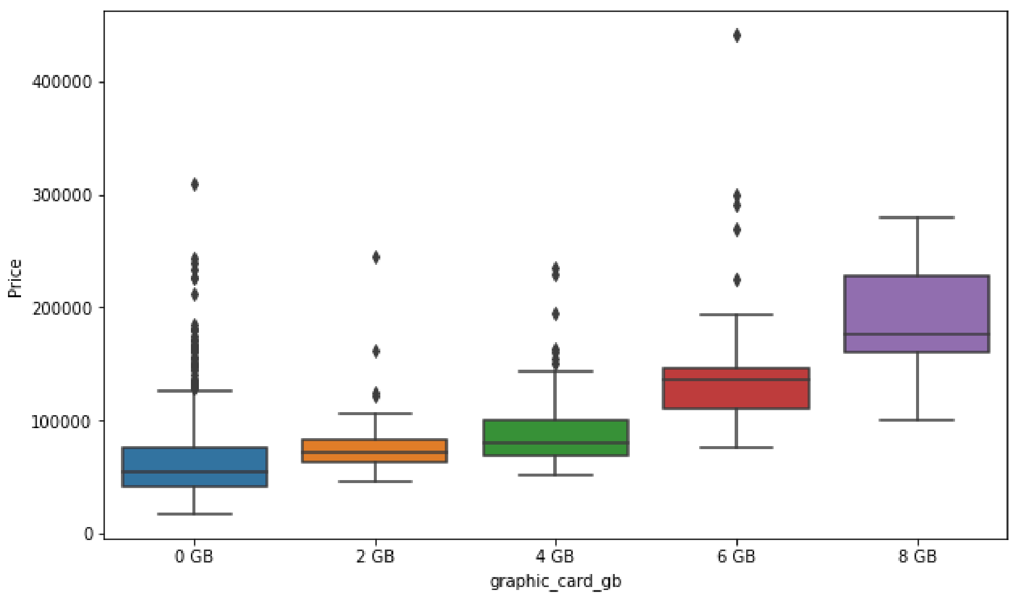
In [134]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="os_bit", y="Price", data=df)
```

Out[134]: <matplotlib.axes._subplots.AxesSubplot at 0x122d26b00d0>



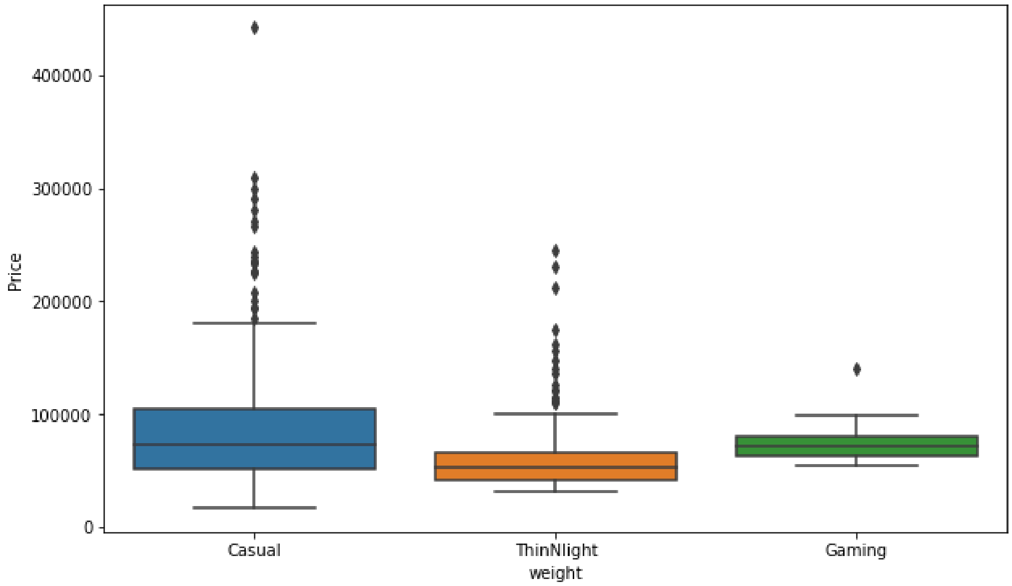
In [135]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="graphic_card_gb", y="Price", data=df)
```

Out[135]: <matplotlib.axes._subplots.AxesSubplot at 0x122d2db06d0>



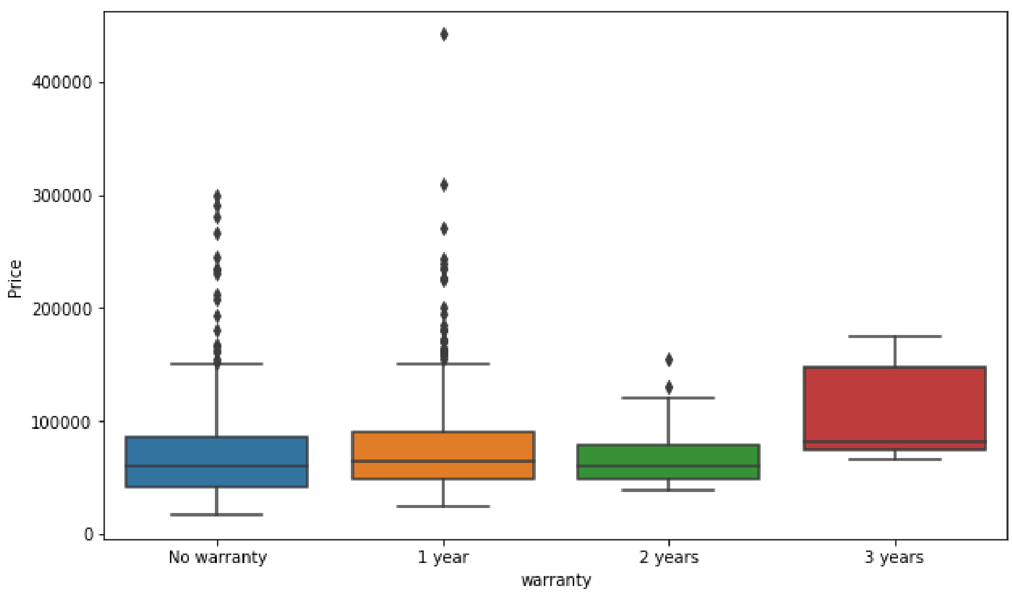
In [136]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="weight", y="Price", data=df)
```

Out[136]: <matplotlib.axes._subplots.AxesSubplot at 0x122d2e6b490>



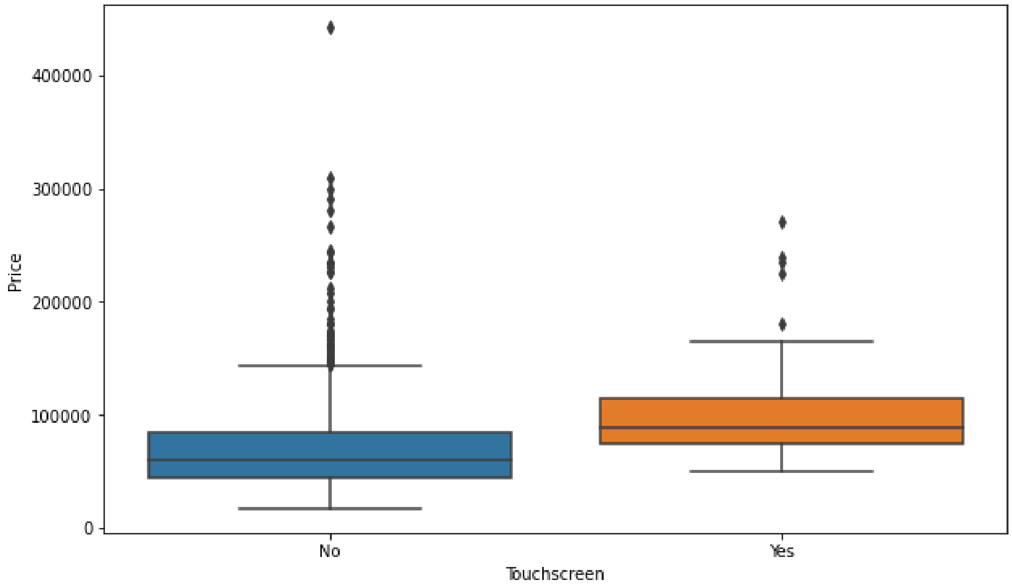
In [137]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="warranty", y="Price", data=df)
```

Out[137]: <matplotlib.axes._subplots.AxesSubplot at 0x122d2ed4b20>



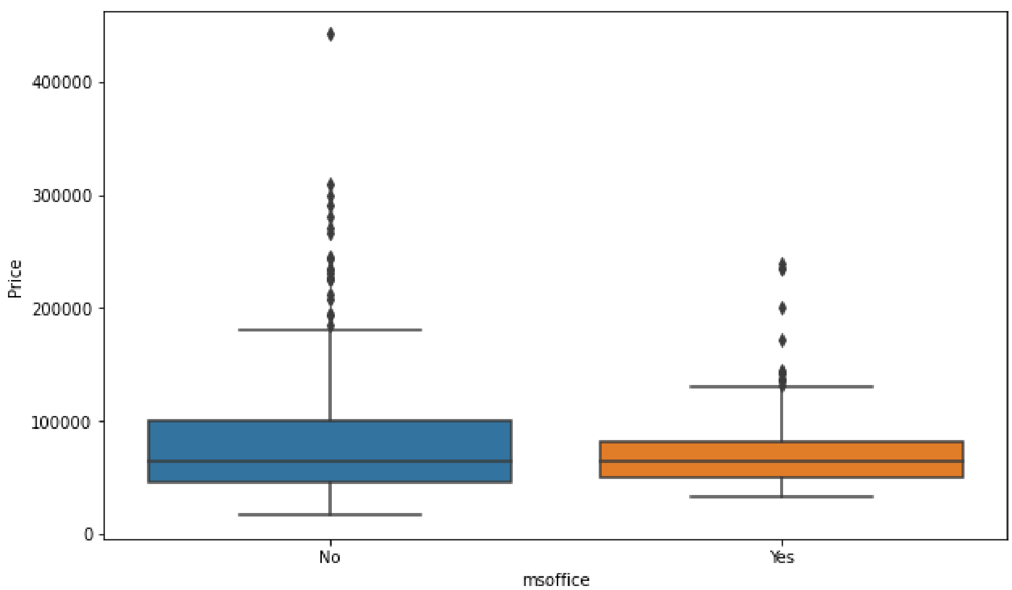
In [138]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="Touchscreen", y="Price", data=df)
```

Out[138]: <matplotlib.axes._subplots.AxesSubplot at 0x122d3f41f40>



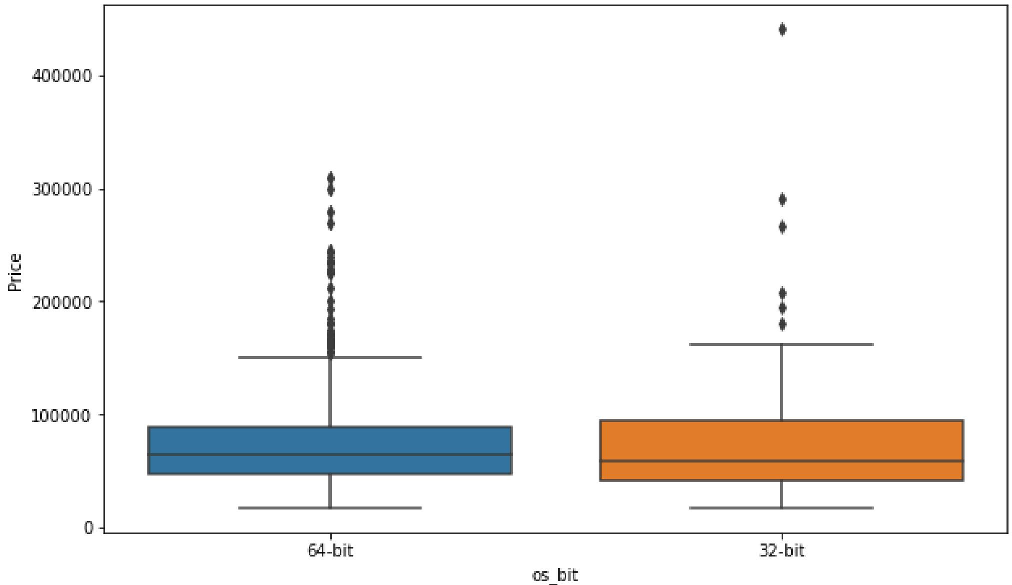
In [139]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="msoffice", y="Price", data=df)
```

Out[139]: <matplotlib.axes._subplots.AxesSubplot at 0x122d3fb1f10>



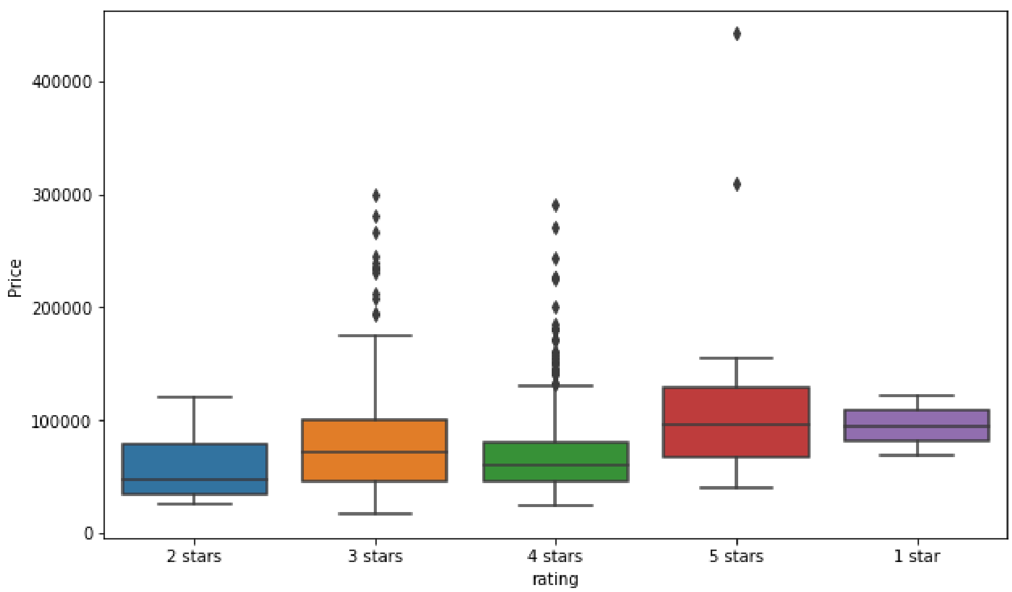
In [140]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="os_bit", y="Price", data=df)
```

Out[140]: <matplotlib.axes._subplots.AxesSubplot at 0x122d4013ca0>



In [141]:
```
plt.figure(figsize=(10,6))
sns.boxplot(x="rating", y="Price", data=df)
```

Out[141]: <matplotlib.axes._subplots.AxesSubplot at 0x122d40afeb0>



In [142]:
```
df.drop(['weight', 'warranty', 'Touchscreen','processor_brand','os_bit'], axis = 1, inplace = True)
```

In [143]:
```
df
```

Out[143]:

	brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	os	graphic_card_gb	msoffice	Price	ra
0	ASUS	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	0 GB	No	34649	
1	Lenovo	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	0 GB	No	38999	
2	Lenovo	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	0 GB	No	39999	
3	ASUS	Core i5	10th	8 GB	DDR4	512 GB	0 GB	Windows	2 GB	No	69990	
4	ASUS	Celeron Dual	Not Available	4 GB	DDR4	0 GB	512 GB	Windows	0 GB	No	26990	
...	...	...	...	...	...	...	...	...	...	...	...	
818	ASUS	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	0 GB	No	135990	
819	ASUS	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	0 GB	No	144990	
820	ASUS	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	4 GB	No	149990	
821	ASUS	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	4 GB	No	142990	
822	Lenovo	Ryzen 5	10th	8 GB	DDR4	512 GB	0 GB	DOS	0 GB	No	57490	

802 rows × 14 columns

In [144]:
```
df
```

Out[144]:

	brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	os	graphic_card_gb	msoffice	Price	ra
0	ASUS	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	0 GB	No	34649	
1	Lenovo	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	0 GB	No	38999	
2	Lenovo	Core i3	10th	4 GB	DDR4	0 GB	1024 GB	Windows	0 GB	No	39999	
3	ASUS	Core i5	10th	8 GB	DDR4	512 GB	0 GB	Windows	2 GB	No	69990	
4	ASUS	Celeron Dual	Not Available	4 GB	DDR4	0 GB	512 GB	Windows	0 GB	No	26990	
...	...	...	...	...	...	...	...	...	...	...	...	
818	ASUS	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	0 GB	No	135990	
819	ASUS	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	0 GB	No	144990	
820	ASUS	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	4 GB	No	149990	
821	ASUS	Ryzen 9	Not Available	4 GB	DDR4	1024 GB	0 GB	Windows	4 GB	No	142990	
822	Lenovo	Ryzen 5	10th	8 GB	DDR4	512 GB	0 GB	DOS	0 GB	No	57490	

802 rows × 14 columns

In [145]:
```
#brand', 'processor_name', 'processor_gnrtn', 'ram_gb', 'ram_type', 'ssd', 'hdd', 'os',
#'graphic_card_gb', 'msoffice', 'rating'
from sklearn.preprocessing import LabelEncoder

labelencoder = LabelEncoder()
df.brand = labelencoder.fit_transform(df.brand)
df.processor_name = labelencoder.fit_transform(df.processor_name)
df.processor_gnrtn = labelencoder.fit_transform(df.processor_gnrtn)
df.ram_gb = labelencoder.fit_transform(df.ram_gb)
df.ram_type = labelencoder.fit_transform(df.ram_type)
df.ssd = labelencoder.fit_transform(df.ssd)
df.hdd = labelencoder.fit_transform(df.hdd)
df.os = labelencoder.fit_transform(df.os)
df.graphic_card_gb = labelencoder.fit_transform(df.graphic_card_gb)
df.msoffice = labelencoder.fit_transform(df.msoffice)
df.rating = labelencoder.fit_transform(df.rating)
from sklearn.preprocessing import LabelEncoder
```

In [168]:
```
import scipy.stats as stats
df = stats.zscore(df)
```

In [170]:
```
df
```

Out[170]:
```
array([[-1.15409599, -0.87301913, -0.91832804, ..., -2.76166878,
        -0.2965003 , -0.30522536],
       [ 0.87903961, -0.87301913, -0.91832804, ..., -1.00105005,
        -0.23457211, -0.2629384 ],
       [ 0.87903961, -0.87301913, -0.91832804, ..., -1.00105005,
        -0.29150609, -0.29676797],
       ...,
       [-1.15409599,  2.20686947,  1.48628652, ..., -1.00105005,
        -0.29949682, -0.30522536],
       [-1.15409599,  2.20686947,  1.48628652, ..., -1.00105005,
        -0.29949682, -0.30522536],
       [ 0.87903961,  1.52244978, -0.91832804, ...,  0.75956868,
        -0.28151767, -0.27139579]])
```

In [171]:
```
x_train=df.iloc[:,0:13]
y_train=df.iloc[:,10]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-171-e365df6435b8> in <module>
----> 1 x_train=df.iloc[:,0:13]
      2 y_train=df.iloc[:,10]

AttributeError: 'numpy.ndarray' object has no attribute 'iloc'
```

In [172]:
```
x_train.head()
```

Out[172]:

	brand	processor_name	processor_gnrtn	ram_gb	ram_type	ssd	hdd	os	graphic_card_gb	msoffice	Price	rating	Numb Ratin
0	1	1	0	2	1	0	1	2	0	0	34649	1	
1	5	1	0	2	1	0	1	2	0	0	38999	2	6
2	5	1	0	2	1	0	1	2	0	0	39999	2	
3	1	2	0	3	1	6	0	2	1	0	69990	2	
4	1	0	7	2	1	0	3	2	0	0	26990	2	

In [173]:
```
y_train.head()
```

Out[173]:
```
0    34649
1    38999
2    39999
3    69990
4    26990
Name: Price, dtype: int64
```

In [174]:
```
# importing train_test_split from sklearn
from sklearn.model_selection import train_test_split
# splitting the data  # 30% for testing is used
X_train, X_test, Y_train, Y_test = train_test_split(x_train, y_train, test_size = 0.3, random_state = 0
```

In [175]:
```
#Multiple Linear Regression
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model_mlr = model.fit(X_train,Y_train)
```

In [176]:
```
#Making price prediction using the testing set (Fit to MLR)
Y_pred_MLR = model_mlr.predict(X_test)
```

In [177]:
```
#Calculating the Mean Square Error for MLR model
mse_MLR = mean_squared_error(Y_test, Y_pred_MLR)
print('The mean square error for Multiple Linear Regression: ', mse_MLR)
```

```
The mean square error for Multiple Linear Regression:  3.4493088314991315e-22
```

In [178]:
```
#The mean square error for Multiple Linear Regression:  0.3674647167443785
```

In [179]:
```
#Calculating the Mean Absolute Error for MLR model
mae_MLR= mean_absolute_error(Y_test, Y_pred_MLR)
print('The mean absolute error for Multiple Linear Regression: ', mae_MLR)
```

```
The mean absolute error for Multiple Linear Regression:  1.5533112831939305e-11
```

In [180]:
```
#Calling the random forest model and fitting the training data
rfModel = RandomForestRegressor()
model_rf = rfModel.fit(X_train,Y_train)
```

In [181]:
```
#Prediction of Laptop prices using the testing data
Y_pred_RF = model_rf.predict(X_test)
```

In [182]:
```
#Calculating the Mean Square Error for Random Forest Model
mse_RF = mean_squared_error(Y_test, Y_pred_RF)
print('The mean square error of price and predicted value is: ', mse_RF)
```

```
The mean square error of price and predicted value is:  2941640.472541086
```

In [183]:
```
#Calculating the Mean Absolute Error for Random Forest Model
mae_RF= mean_absolute_error(Y_test, Y_pred_RF)
print('The mean absolute error of price and predicted value is: ', mae_RF)
```

```
The mean absolute error of price and predicted value is:  317.3755186721992
```

In [184]:
```
#LASSO Model
#Calling the model and fitting the training data
LassoModel = Lasso()
model_lm = LassoModel.fit(X_train,Y_train)
```

In [185]:
```
#Price prediction uisng testing data
Y_pred_lasso = model_lm.predict(X_test)
```

In [186]:
```
#Mean Absolute Error for LASSO Model
mae_lasso= mean_absolute_error(Y_test, Y_pred_lasso)
print('The mean absolute error of price and predicted value is: ', mae_lasso)
```

```
The mean absolute error of price and predicted value is:  1.3399300380673692e-05
```

In [187]:
```
#Mean Squared Error for the LASSO Model
mse_lasso = mean_squared_error(Y_test, Y_pred_lasso)
print('The mean square error of price and predicted value is: ', mse_lasso)
```

```
The mean square error of price and predicted value is:  3.446638548304528e-10
```

In [188]:
```
scores = [('MLR', mae_MLR),
          ('Random Forest', mae_RF),
          ('LASSO', mae_lasso)
         ]
```

In [189]:
```
mae = pd.DataFrame(data = scores, columns=['Model', 'MAE Score'])
mae
```

Out[189]:

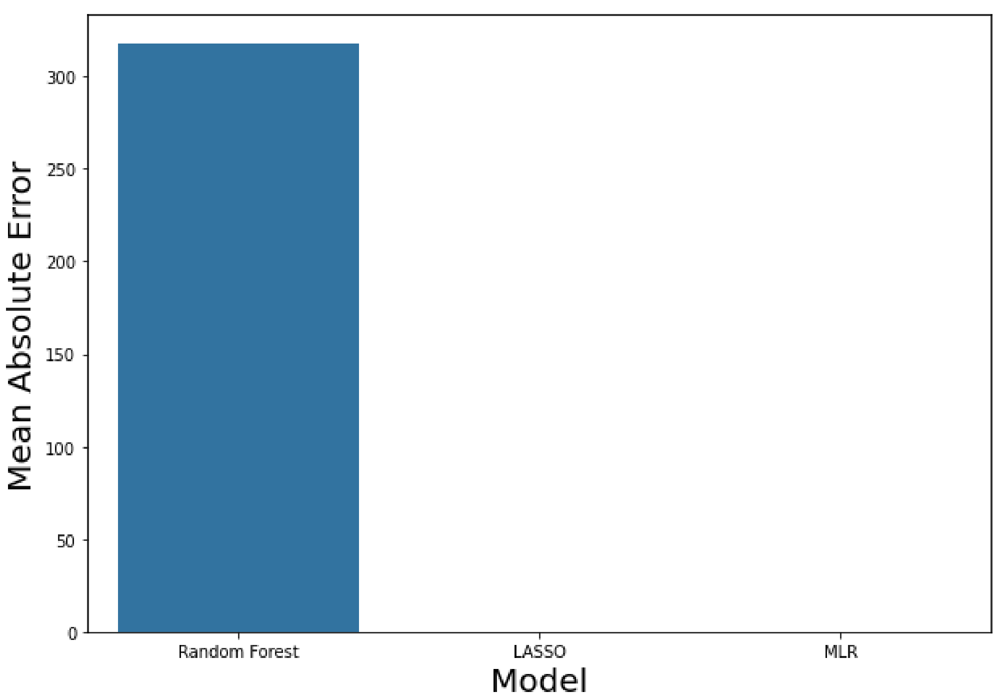
	Model	MAE Score
0	MLR	1.553311e-11
1	Random Forest	3.173755e+02
2	LASSO	1.339930e-05

In [ ]:
```
# By observing MAE score of MLR, Random Forest and LASSO I observe that random Forest algorith has the
# in general: the lower MAE score the better model.
# Hence , I conclude that Random Forest is the better model among the selected three models
```

In [190]:
```
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('Model', size=20)
axe.set_ylabel('Mean Absolute Error', size=20)

plt.show()
```



In [ ]:

In [ ]: