In [1]:
```python
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]:
```python
formattedData=pd.read_csv(r'C:\Users\21F22087\Desktop\formattedData.csv')
```

In [3]:
```python
formattedData.head(5)
```

Out[3]:

	Year	Make	Model	Kilometres	Body_Type	Engine	Transmission	Drivetrain	Exterior_Colour	Interior_Colour	Passengers	Doors	Fuel_Type	City	Highway	Price
0	2014	Acura	RDX	290000.0	SUV	4	Automatic	AWD	Black	Black	5.0	5	Gas	11.336434	8.668992	11600
1	2014	Acura	RDX	158868.0	SUV	6	6 Speed Automatic	AWD	Silver	Black	5.0	5	Gas	10.700000	7.300000	17998
2	2016	Acura	MDX	226214.0	SUV	6	Automatic	AWD	White	Black	7.0	5	Gas	12.700000	9.100000	17999
3	2019	Acura	MDX	42081.0	SUV	6	9 Speed Automatic	AWD	White Diamond Pearl	Black	5.0	5	NaN	12.200000	9.000000	40588
4	2021	Acura	RDX	NaN	SUV	4	10 Speed Automatic	AWD	Majestic Black Pearl	Black	5.0	5	Gas	11.000000	8.600000	41599

In [4]:
```python
formattedData.shape
```

Out[4]:
```
(18647, 16)
```

In [5]:
```python
formattedData.isnull().sum()
```

Out[5]:
```
Year                0
Make                2
Model               0
Kilometres         21
Body_Type           1
Engine              3
Transmission        3
Drivetrain          0
Exterior_Colour     0
Interior_Colour     0
Passengers          5
Doors               0
Fuel_Type           1
City                0
Highway             0
Price               0
dtype: int64
```

In [6]:
```python
formattedData=formattedData.dropna()
```

In [7]:
```python
formattedData.isnull().sum()
```

Out[7]:
```
Year               0
Make               0
Model              0
Kilometres         0
Body_Type          0
Engine             0
Transmission       0
Drivetrain         0
Exterior_Colour    0
Interior_Colour    0
Passengers         0
Doors              0
Fuel_Type          0
City               0
Highway            0
Price              0
dtype: int64
```

In [8]:
```python
formattedData.shape
```

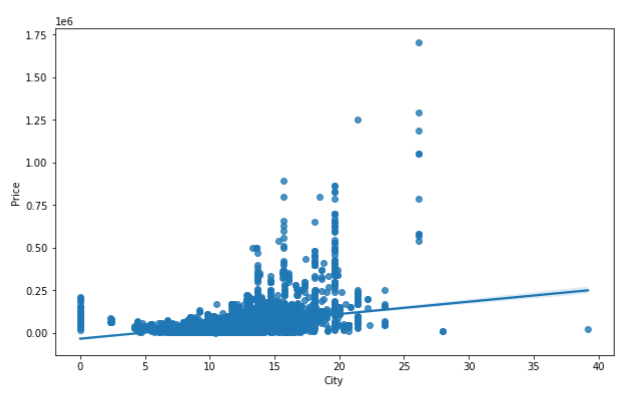
Out[8]:
```
(18612, 16)
```

In [9]:
```python
formattedData.dtypes
```

Out[9]:
```
Year                 int64
Make                object
Model               object
Kilometres         float64
Body_Type           object
Engine              object
Transmission        object
Drivetrain          object
Exterior_Colour     object
Interior_Colour     object
Passengers         float64
Doors                int64
Fuel_Type           object
City               float64
Highway            float64
Price                int64
dtype: object
```

In [10]:
```python
import seaborn as sns
plt.figure(figsize=(10,6))
sns.regplot(x="City", y="Price", data=formattedData)
```

Out[10]:
```
<AxesSubplot:xlabel='City', ylabel='Price'>
```

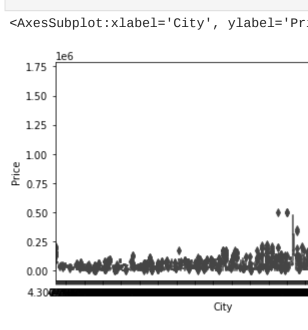


In [11]:
```python
from scipy import stats
pearson_coef,p_value=stats.pearsonr(formattedData['City'],formattedData['Price'])
print("The Pearson Correlation Coefficient is",pearson_coef,"with a P-value of P=",p_value)
```

```
The Pearson Correlation Coefficient is 0.39678696948232167 with a P-value of P= 0.0
```

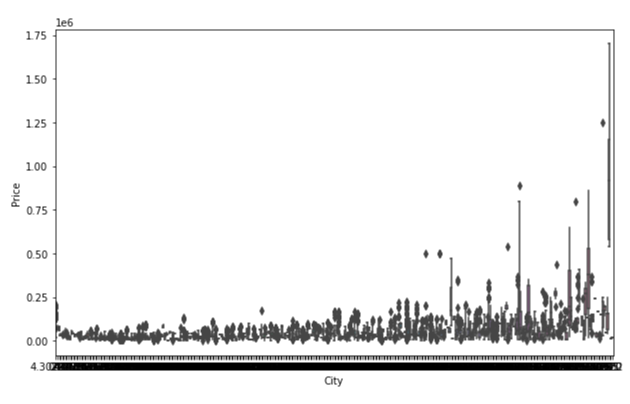
In [12]:
```python
sns.boxplot(x="City", y="Price", data=formattedData)
```

Out[12]:
```
<AxesSubplot:xlabel='City', ylabel='Price'>
```



In [13]:
```python
plt.figure(figsize=(10,6))
sns.boxplot(x="City", y="Price", data=formattedData)
```

Out[13]:
```
<AxesSubplot:xlabel='City', ylabel='Price'>
```



In [14]:
```python
formattedData.drop(['Year', 'Make', 'Model','Kilometres','Body_Type','Engine','Transmission'], axis = 1, inplace = True)
```

In [15]:
```python
formattedData.shape
```

Out[15]:
```
(18612, 9)
```

In [16]:
```python
formattedData.describe()
```

Out[16]:

	Passengers	Doors	City	Highway	Price
count	18612.000000	18612.000000	18612.000000	18612.000000	1.861200e+04
mean	5.131850	3.734956	11.208205	8.401187	4.742887e+04
std	0.947508	0.729917	2.932372	2.095519	5.341182e+04
min	2.000000	2.000000	0.000000	0.000000	2.000000e+03
25%	5.000000	4.000000	9.300000	7.200000	2.485000e+04
50%	5.000000	4.000000	11.200000	8.414118	3.699500e+04
75%	5.000000	4.000000	12.900000	9.600000	5.791550e+04
max	15.000000	5.000000	39.200000	42.800000	1.699998e+06

In [18]:
```python
formattedData.describe(include=['object'])
```

Out[18]:

	Drivetrain	Exterior_Colour	Interior_Colour	Fuel_Type
count	18612	18612	18612	18612
unique	6	1305	16	9
top	AWD	Black	Black	Gas
freq	9462	1891	14953	17056

In [26]:
```python
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
#formattedData.Body_Type = labelencoder.fit_transform(formattedData.Body_Type)
#formattedData.Make = labelencoder.fit_transform(formattedData.Make)
#formattedData.Model = labelencoder.fit_transform(formattedData.Model)
#formattedData.Transmission = labelencoder.fit_transform(formattedData.Transmission)
formattedData.Drivetrain = labelencoder.fit_transform(formattedData.Drivetrain)
formattedData.Exterior_Colour = labelencoder.fit_transform(formattedData.Exterior_Colour)
formattedData.Interior_Colour = labelencoder.fit_transform(formattedData.Interior_Colour)
formattedData.Fuel_Type = labelencoder.fit_transform(formattedData.Fuel_Type)
```

In [30]:
```python
formattedData.head(10)
```

Out[30]:

	Drivetrain	Exterior_Colour	Interior_Colour	Passengers	Doors	Fuel_Type	City	Highway	Price
0	3	116	1	5.0	5	3	11.336434	8.668992	11600
1	3	1056	1	5.0	5	3	10.700000	7.300000	17998
2	3	1254	1	7.0	5	3	12.700000	9.100000	17999
7	3	536	1	7.0	5	3	12.656604	9.261321	8995
8	3	116	1	5.0	5	3	10.400000	8.600000	28988
9	3	790	1	5.0	5	3	10.212500	7.128571	68917
10	3	498	1	5.0	5	3	13.700000	10.950000	96003
11	3	790	1	5.0	5	3	10.811111	8.533333	64921
12	3	708	1	5.0	5	3	10.811111	8.533333	54969
14	3	116	1	5.0	5	3	10.212500	7.128571	59819

In [35]:
```python
import scipy.stats as stats
formattedData = stats.zscore(formattedData)
formattedData = stats.zscore(formattedData)
```

In [32]:
```python
formattedData
```

Out[32]:

	Drivetrain	Exterior_Colour	Interior_Colour	Passengers	Doors	Fuel_Type	City	Highway	Price
0	-0.172438	-1.251028	-0.401947	-0.139159	1.733181	-0.017795	0.043730	0.127803	-0.670822
1	-0.172438	1.069340	-0.401947	-0.139159	1.733181	-0.017795	-0.173313	-0.525510	-0.551033
2	-0.172438	1.558099	-0.401947	1.971697	1.733181	-0.017795	0.508747	0.333489	-0.551014
7	-0.172438	-0.214268	-0.401947	1.971697	1.733181	-0.017795	0.493948	0.410475	-0.719596
8	-0.172438	-1.251028	-0.401947	-0.139159	1.733181	-0.017795	-0.275622	0.094878	-0.345268
...	...	...	...	...	...	...	...	...	...
18642	-0.172438	-1.251028	0.270355	1.971697	0.363125	-0.017795	0.099511	0.524377	-0.363803
18643	-0.172438	-1.199190	-0.401947	-0.139159	0.363125	-0.017795	0.611056	0.476655	-0.532366
18644	-0.172438	1.558099	-0.401947	1.971697	0.363125	-0.017795	0.099511	0.572100	-0.176724
18645	-0.172438	0.523807	-0.401947	-0.139159	0.363125	-0.017795	-0.207416	-0.382344	-0.176911
18646	0.930077	-1.043676	-0.401947	-0.139159	0.363125	-0.017795	-0.241519	-0.859565	-0.672788

18612 rows × 9 columns

In [46]:
```python
x_formattedData=formattedData.iloc[:,0:7]
y_formattedData=formattedData.iloc[:,8]
x_formattedData=formattedData.iloc[:,0:7]
y_formattedData=formattedData.iloc[:,8]
```

In [47]:
```python
x_formattedData
```

Out[47]:

	Drivetrain	Exterior_Colour	Interior_Colour	Passengers	Doors	Fuel_Type	City
0	-0.172438	-1.251028	-0.401947	-0.139159	1.733181	-0.017795	0.043730
1	-0.172438	1.069340	-0.401947	-0.139159	1.733181	-0.017795	-0.173313
2	-0.172438	1.558099	-0.401947	1.971697	1.733181	-0.017795	0.508747
7	-0.172438	-0.214268	-0.401947	1.971697	1.733181	-0.017795	0.493948
8	-0.172438	-1.251028	-0.401947	-0.139159	1.733181	-0.017795	-0.275622
...	...	...	...	...	...	...	...
18642	-0.172438	-1.251028	0.270355	1.971697	0.363125	-0.017795	0.099511
18643	-0.172438	-1.199190	-0.401947	-0.139159	0.363125	-0.017795	0.611056
18644	-0.172438	1.558099	-0.401947	1.971697	0.363125	-0.017795	0.099511
18645	-0.172438	0.523807	-0.401947	-0.139159	0.363125	-0.017795	-0.207416
18646	0.930077	-1.043676	-0.401947	-0.139159	0.363125	-0.017795	-0.241519

18612 rows × 7 columns

In [48]:
```python
rg = LinearRegression()
mdl=rg.fit(x_formattedData,y_formattedData)
```

In [49]:
```python
y_pred1 = rg.predict(x_formattedData)
```

In [50]:
```python
print('The R-square for Multiple Linear regression is: ', rg.score(x_formattedData,y_formattedData))
```

```
The R-square for Multiple Linear regression is:  0.23225798157693855
```

In [64]:
```python
mae1= mean_absolute_error(y_formattedData, y_pred1)
print('The mean absolute error for Multiple Linear Regression: ', mae1)
```

```
The mean absolute error for Multiple Linear Regression:  0.42210709604029417
```

In [52]:
```python
mse1 = mean_squared_error(y_formattedData, y_pred1)
print('The mean square error for Multiple Linear Regression: ', mse1)
```

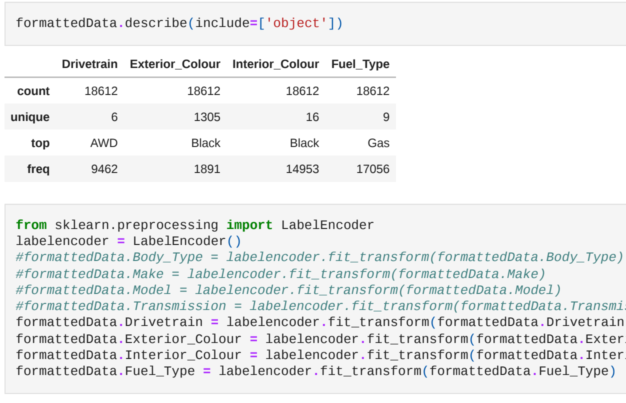
```
The mean square error for Multiple Linear Regression:  0.7677420184230614
```

In [53]:
```python
plt.figure(figsize=(10,6))

ax1 = sns.distplot(y_formattedData, 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 of Cars')

plt.show()
plt.close()
```



In [54]:
```python
rf = RandomForestRegressor()
model=rf.fit(x_formattedData,y_formattedData)
```

In [55]:
```python
y_pred2 = rf.predict(x_formattedData)
```

In [56]:
```python
print('The R-square for Random Forest is: ', rf.score(x_formattedData,y_formattedData))
```

```
The R-square for Random Forest is:  0.940419181805703
```

In [57]:
```python
mse2 = mean_squared_error(y_formattedData, y_pred2)
print('The mean square error of price and predicted value is: ', mse2)
```

```
The mean square error of price and predicted value is:  0.059580818194296996
```

In [66]:
```python
mae2= mean_absolute_error(y_formattedData, y_pred2)
print('The mean absolute error of price and predicted value is: ', mae2)
```

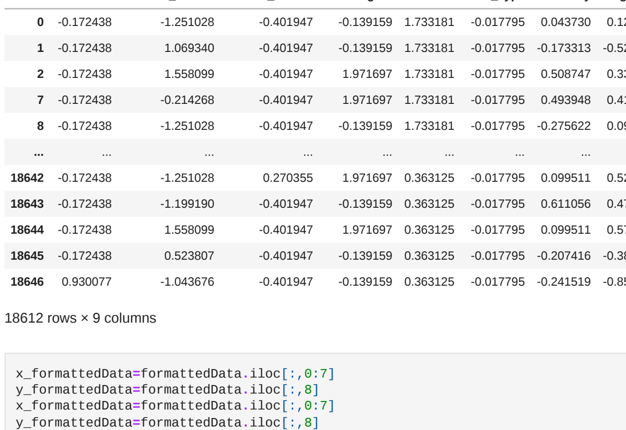
```
The mean absolute error of price and predicted value is:  0.0966348475559992
```

In [58]:
```python
plt.figure(figsize=(10,6))

ax1 = sns.distplot(y_formattedData, 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 of Cars')

plt.show()
plt.close()
```



In [59]:
```python
LassoModel=Lasso()
lm=LassoModel.fit(x_formattedData,y_formattedData)
```

In [60]:
```python
y_pred3 = lm.predict(x_formattedData)
```

In [61]:
```python
print('The R-square for LASSO is: ', lm.score(x_formattedData,y_formattedData))
```

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

In [62]:
```python
mae3= mean_absolute_error(y_formattedData, y_pred3)
print('The mean absolute error of price and predicted value is: ', mae3)
```

```
The mean absolute error of price and predicted value is:  0.48392371068875223
```

In [67]:
```python
scores = [('MLR', mae1),
          ('Random Forest', mae2),
          ('LASSO', mae3) ]
```

In [68]:
```python
mae = pd.DataFrame(data = scores, columns=['Model', 'MAE Score'])
mae
```

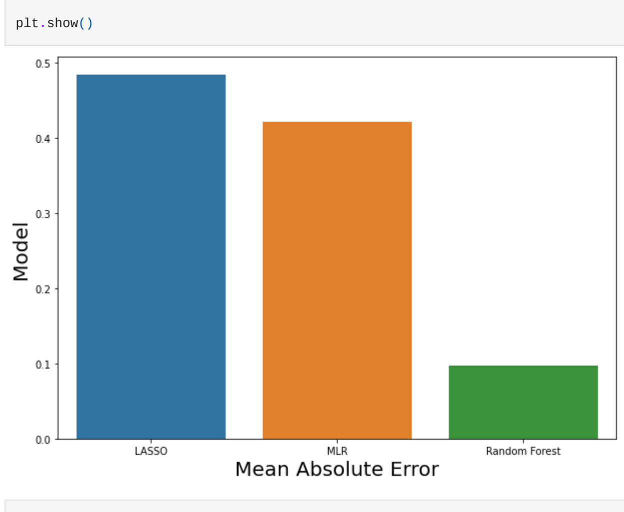
Out[68]:

	Model	MAE Score
0	MLR	0.422107
1	Random Forest	0.096635
2	LASSO	0.483924

In [69]:
```python
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 [ ]: