Laptop Price Predictor

Want to purchase Laptop?

Dont know about the price ?

Dont know about the Good Brand ?

Let’s explore

Shop Now

OVERVIEW

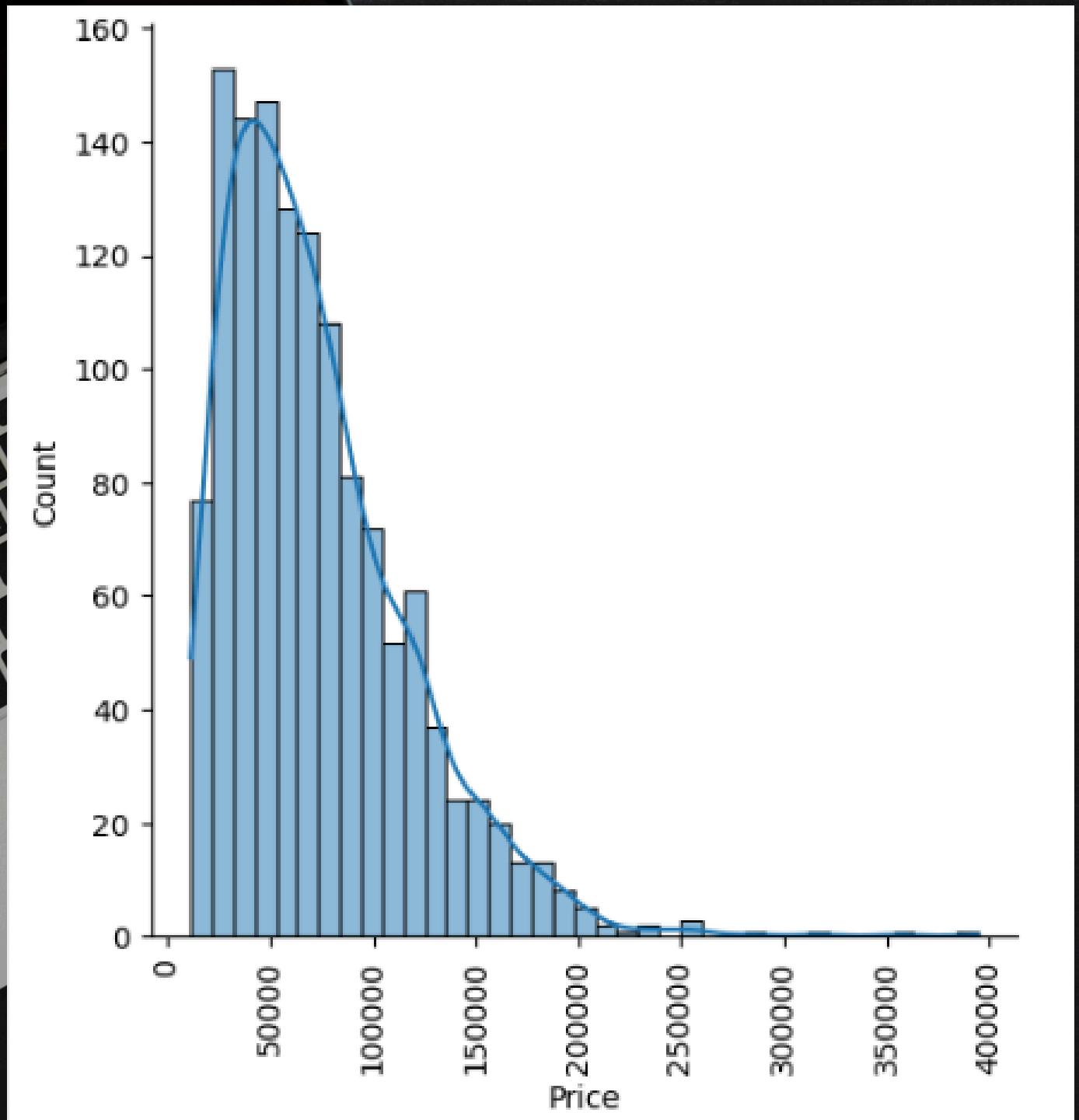
This project aims to forecast the current price of a laptop based on its specifications.

The dataset utilized is five years old, necessitating adjustments to the prices.

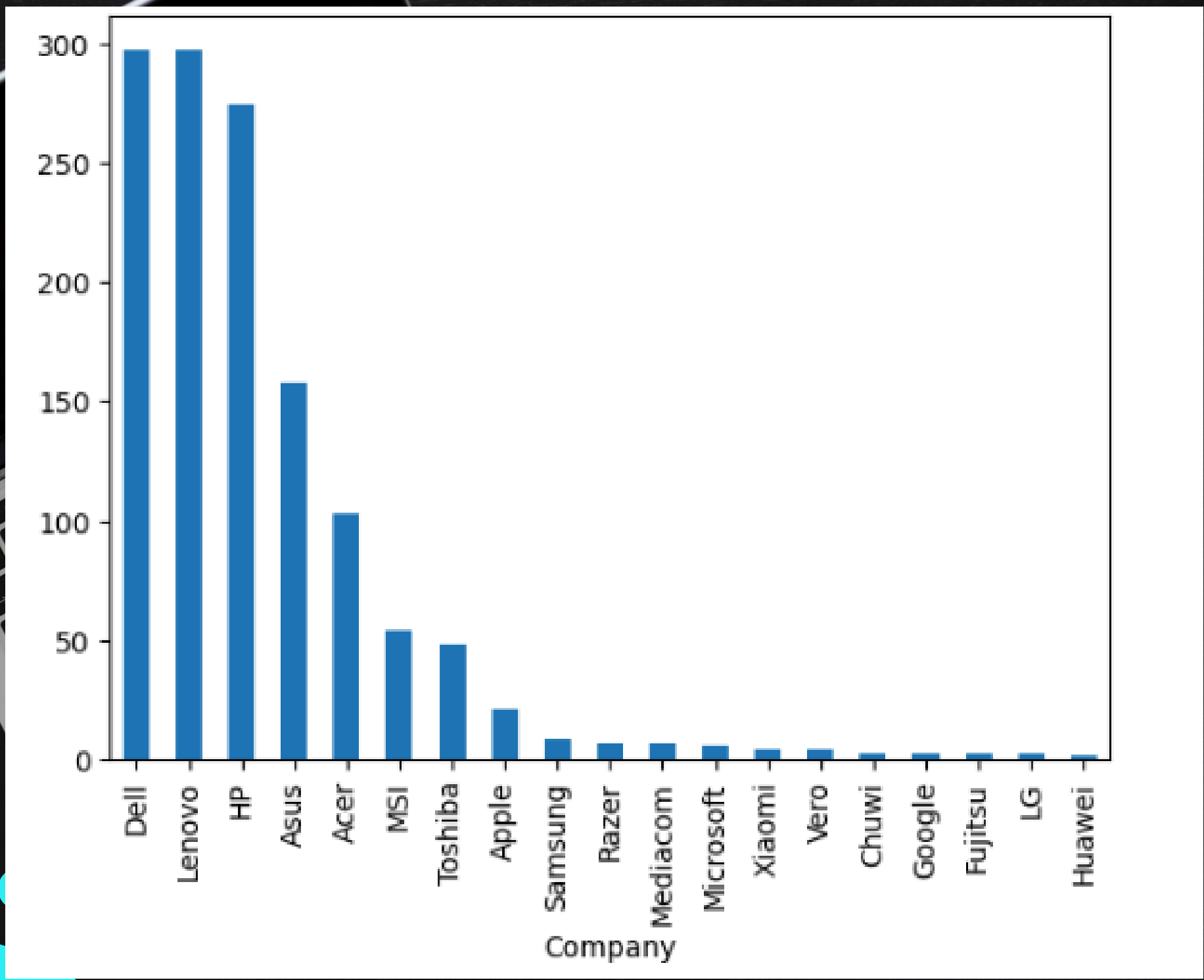
This was done by first converting them to USD using the exchange rate from 2019 and then recalibrating them with today's exchange rate.

EDA

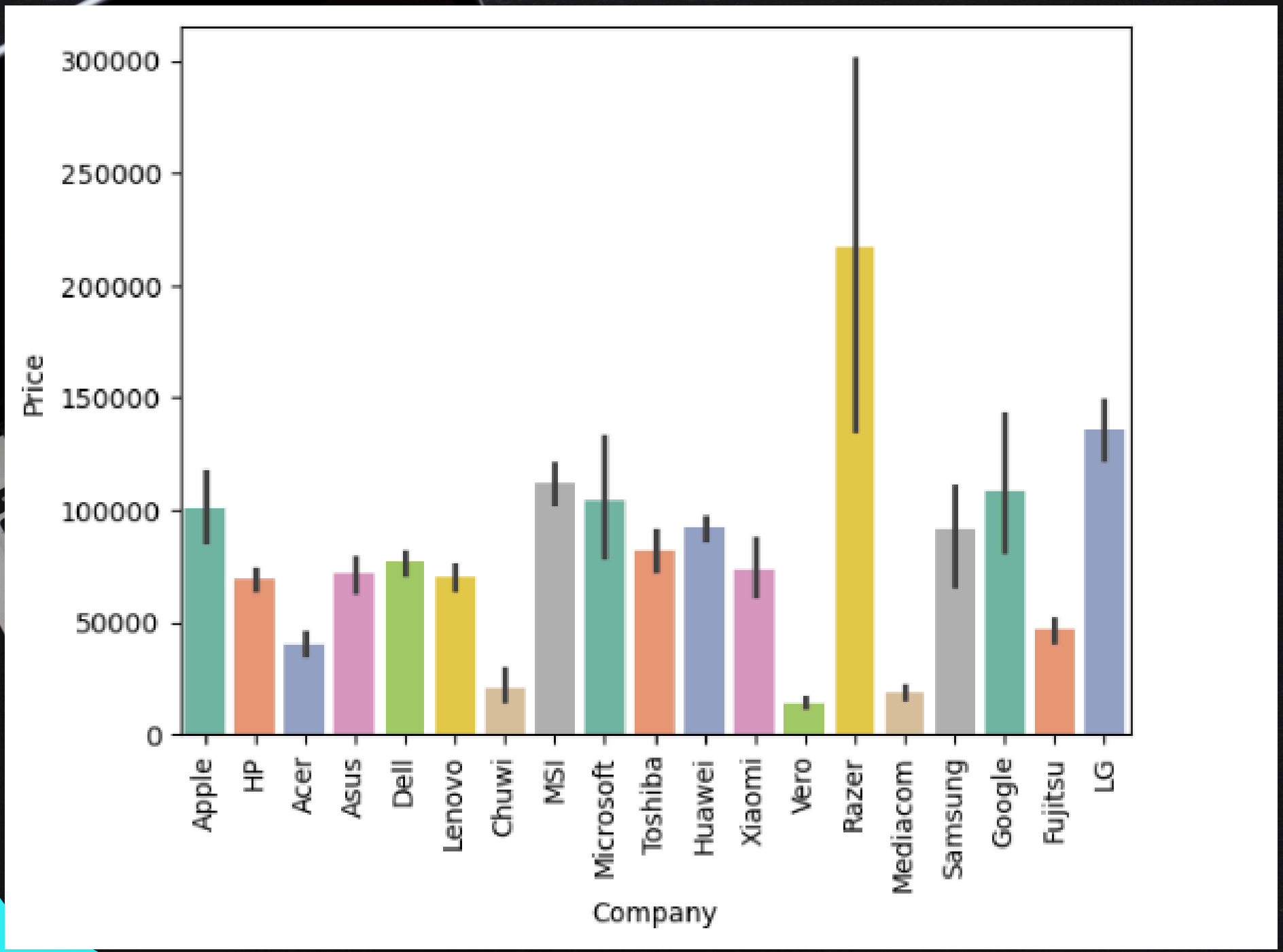
```
sns.displot(df['Price'],kde=True)
plt.xticks(rotation='vertical')
plt.show()
```



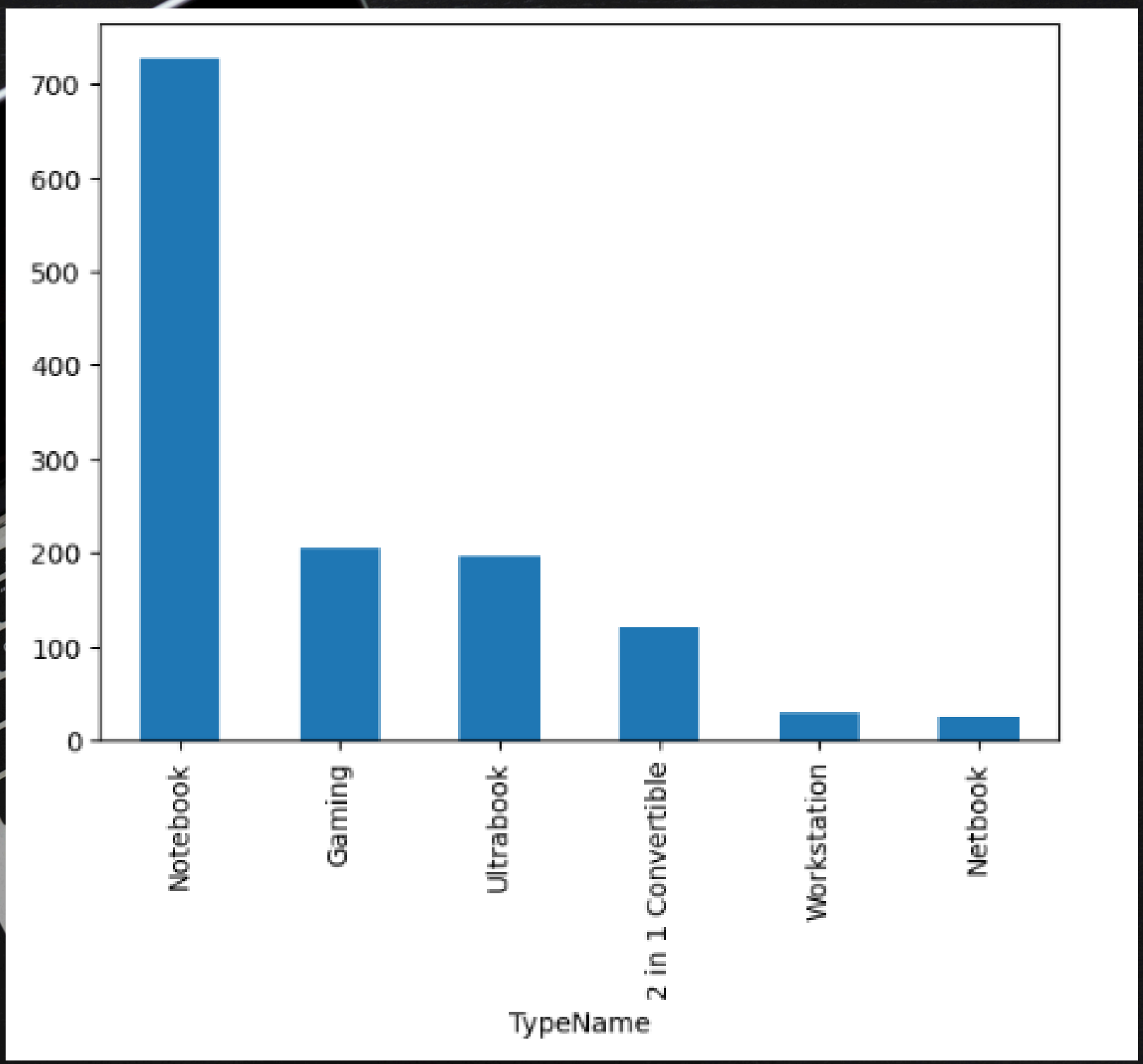
```python
# check for dominating company in market

df['Company'].value_counts().plot(kind='bar')
plt.show()
```



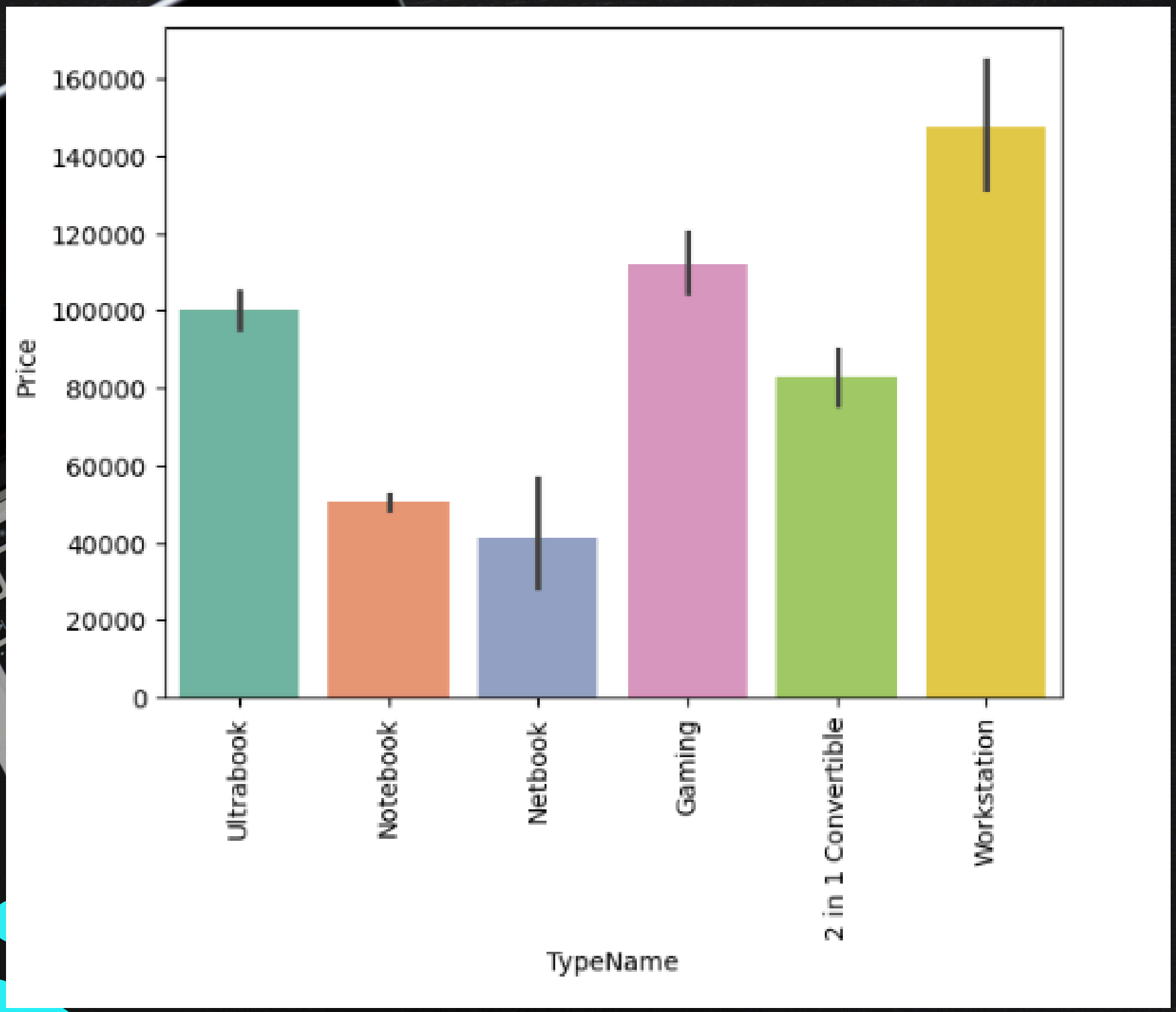
```python
# check price range for every company's laptop

sns.barplot(x=df['Company'],y=df['Price'],palette ='Set2')
plt.xticks(rotation='vertical')
plt.show()
```



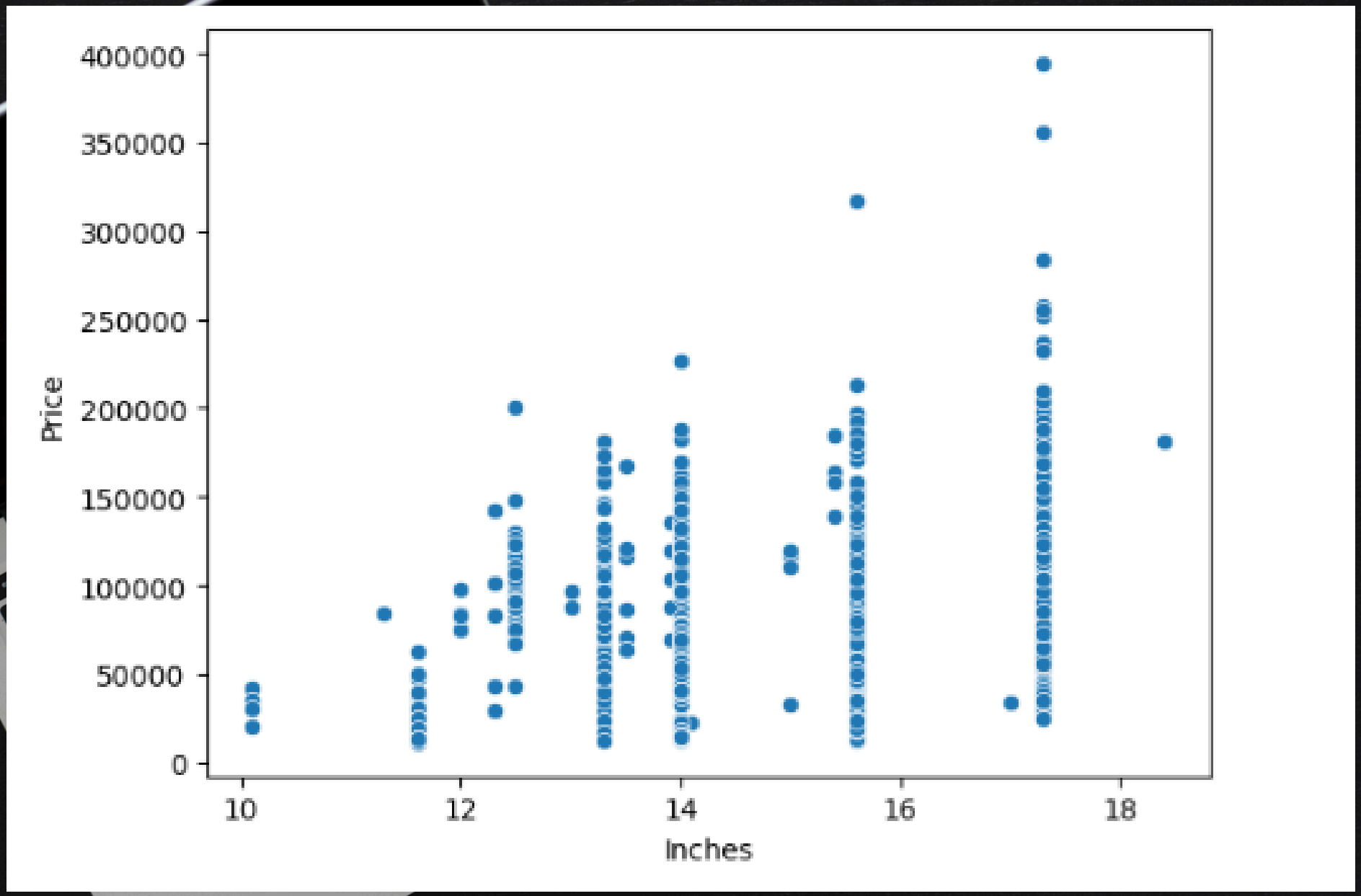
```python
# check for type of Laptops

df['TypeName'].value_counts().plot(kind='bar')
plt.show()
```



```python
# check price range for every type laptop

sns.barplot(x=df['TypeName'],y=df['Price'],palette='Set2')
plt.xticks(rotation='vertical')
plt.show()
```

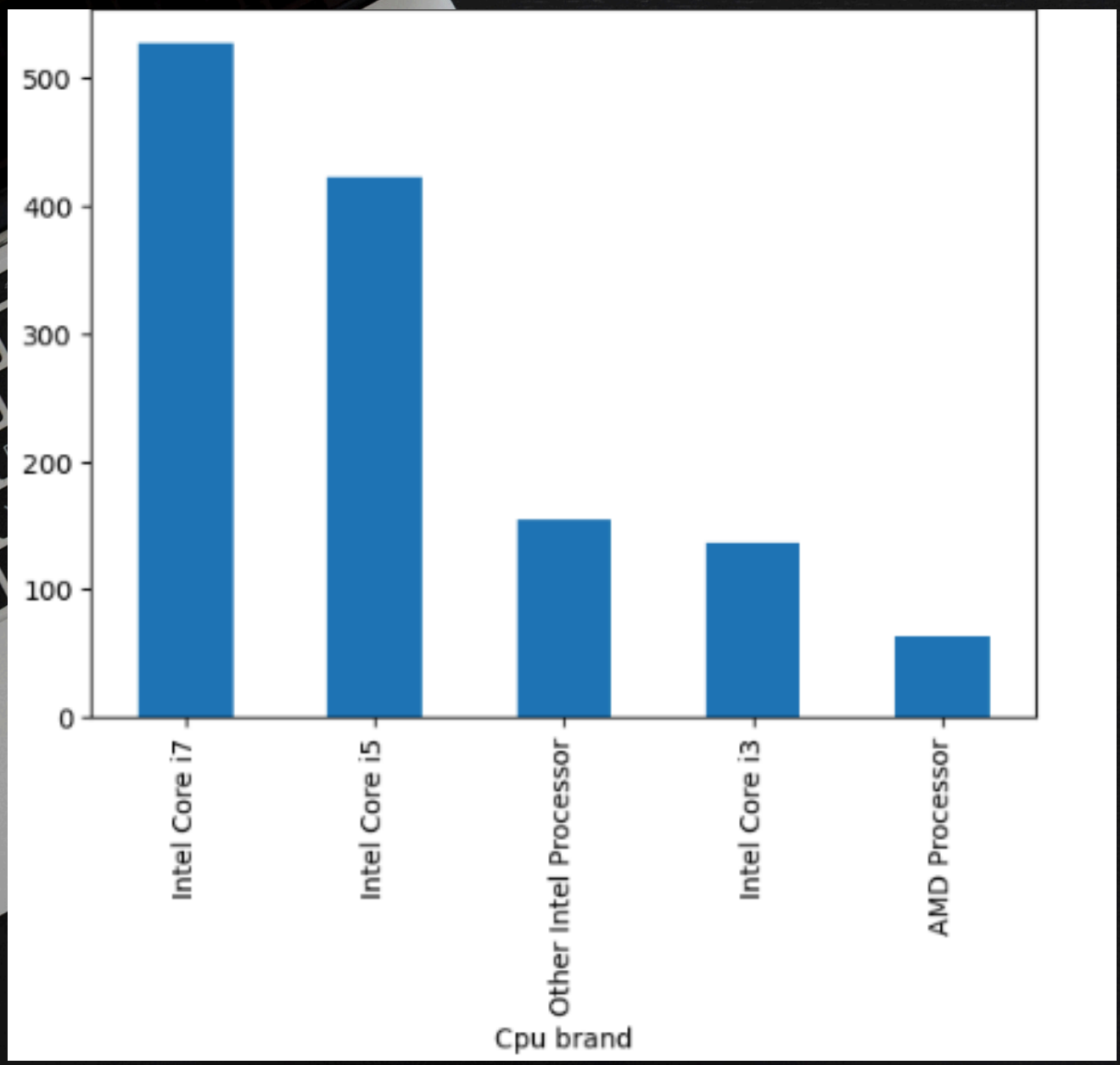


```python
# check size impact on price

sns.scatterplot(x= df['Inches'],y=df['Price'])
plt.show()
```

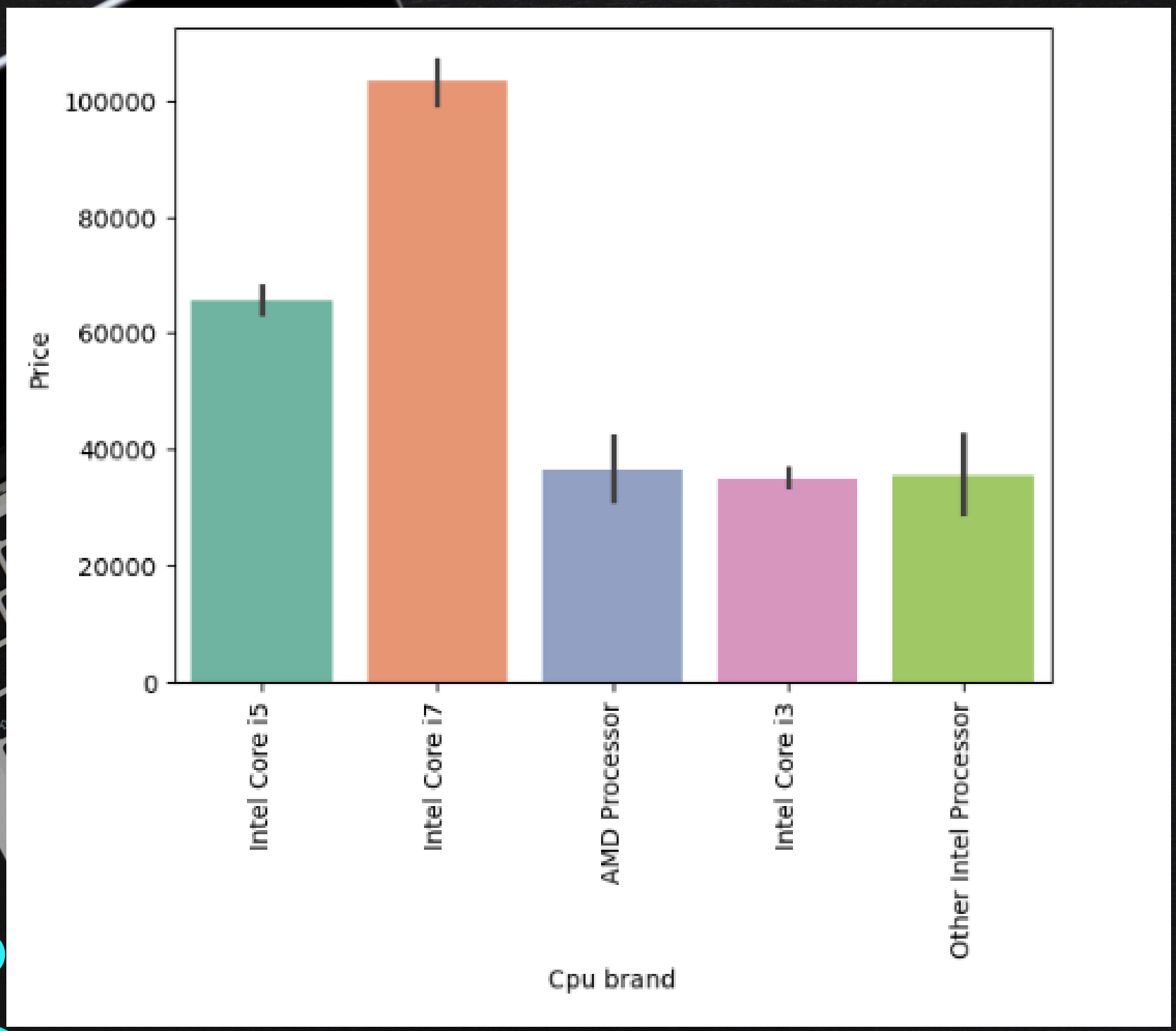


```python
# define a function to print out cpu name

def fetch_processor (text):
    if text== 'Intel Core i7' or text == 'Intel Core i5' or text == 'Intel Core i3':
        return text
    if text.split()[0]== 'Intel':
        return 'Other Intel Processor'
    else:
        return 'AMD Processor'
```

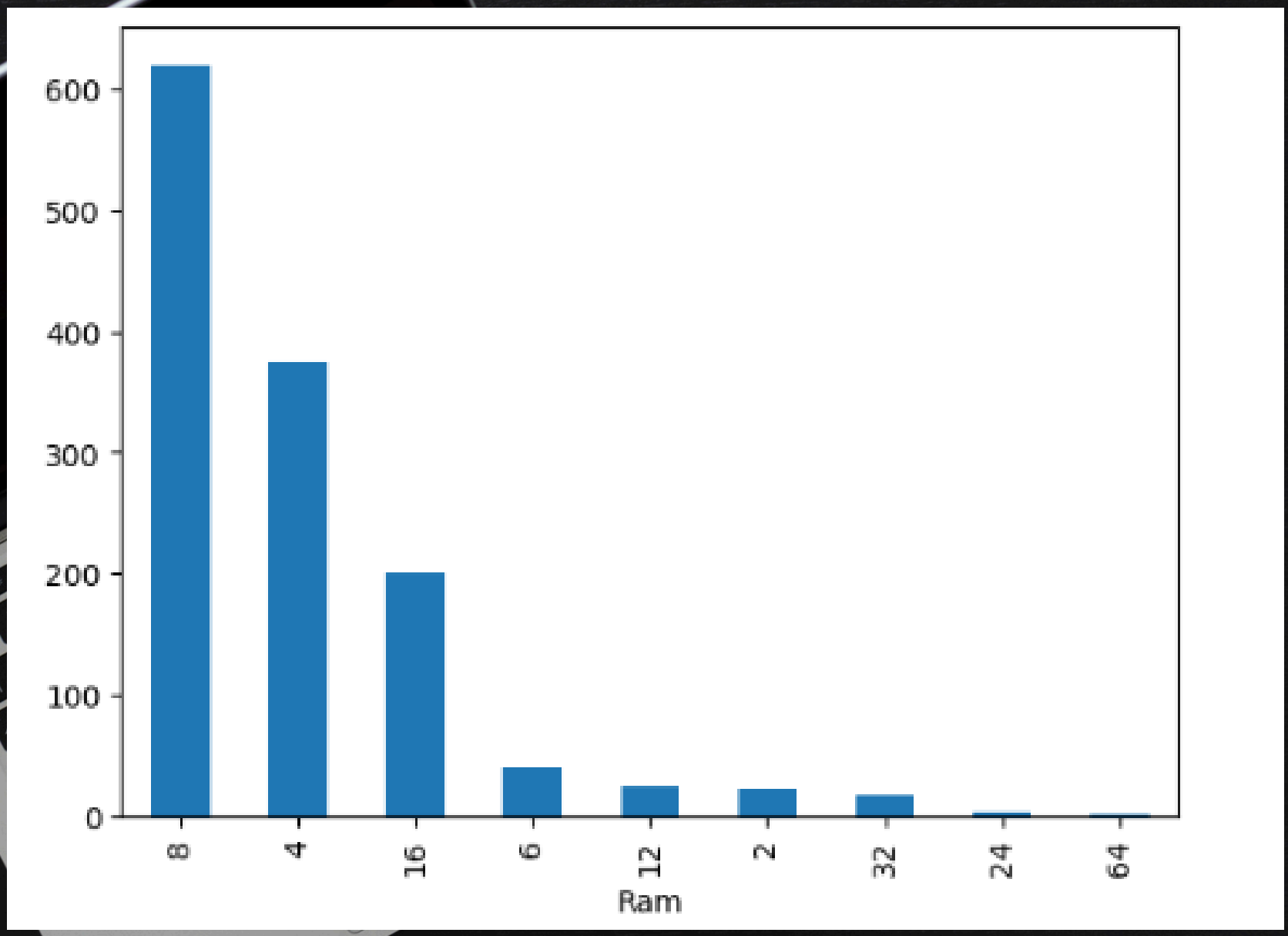
```python
df['Cpu brand']= df['Cpu Name'].apply(fetch_processor)
```



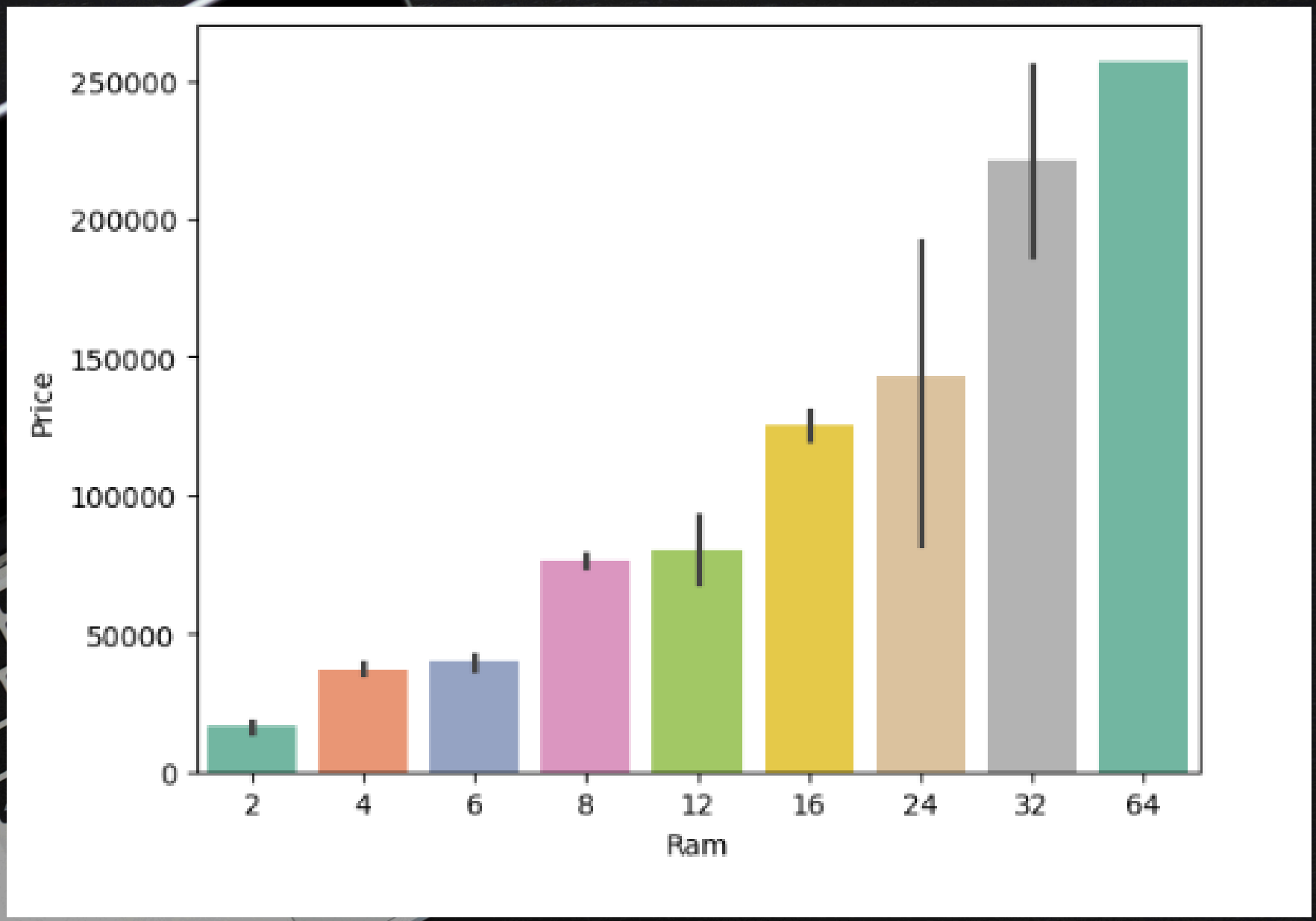
```python
# Check for average price for cpu brand

sns.barplot(x=df['Cpu brand'],y=df['Price'],palette='Set2')
plt.xticks(rotation='vertical')
plt.show()
```



```python
# analysis on Ram column

df['Ram'].value_counts().plot(kind='bar')
plt.show()
```



```
sns.barplot(x=df['Ram'],y=df['Price'],palette='Set2')
plt.show()
```



```python
# Clean the 'Memory' column
df['Memory'] = df['Memory'].astype(str).replace(r'\.0', '', regex=True)
df['Memory'] = df['Memory'].str.replace('GB', '', regex=False).str.replace('TB', '000', regex=False)
```

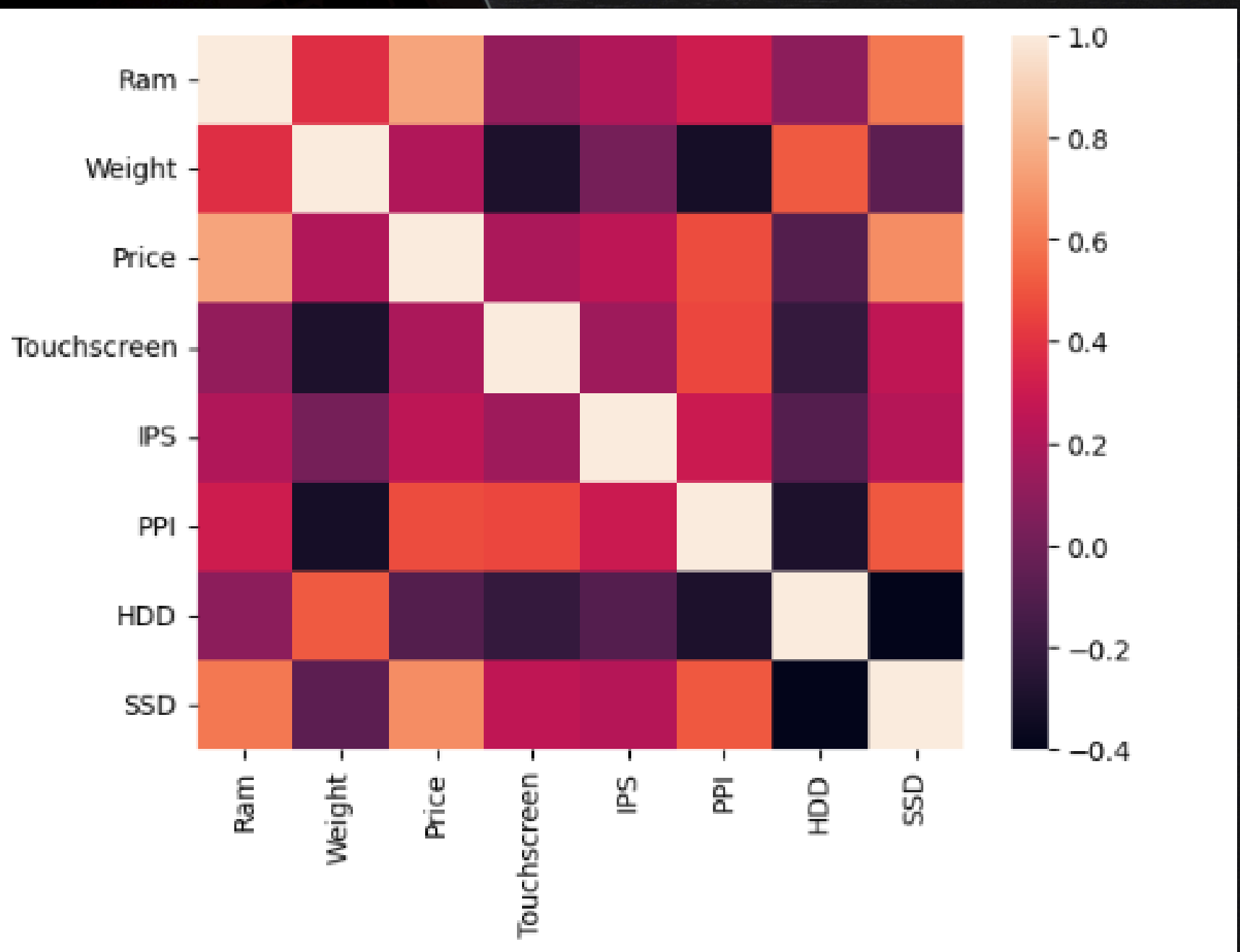
```python
# Split into two parts (if there's a "+" sign)
df[['first', 'second']] = df['Memory'].str.split('+', n=1, expand=True).fillna("0").applymap(str.strip)
```

```python
# Function to extract storage type and size
def extract_storage(storage_str, storage_type):
    return int(''.join(filter(str.isdigit, storage_str))) if storage_type in storage_str else 0
```

```python
# Apply function to extract values
df["HDD"] = df["first"].apply(lambda x: extract_storage(x, "HDD")) + df["second"].apply(lambda x: extract_storage(x, "HDD"))
df["SSD"] = df["first"].apply(lambda x: extract_storage(x, "SSD")) + df["second"].apply(lambda x: extract_storage(x, "SSD"))
df["Hybrid"] = df["first"].apply(lambda x: extract_storage(x, "Hybrid")) + df["second"].apply(lambda x: extract_storage(x, "Hybrid"))
df["Flash_Storage"] = df["first"].apply(lambda x: extract_storage(x, "Flash Storage")) + df["second"].apply(lambda x: extract_storage(x, "Flash Storage"
```

```python
# check for the correlation for all numeric features

df_numeric = df.select_dtypes(include=np.number)
df_corr = df_numeric.corr()
```

```python
sns.heatmap(df_corr)
plt.show()
```



Linear regression

```
step1=ColumnTransformer(transformers=[
    ('col_trans',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2= LinearRegression()

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2 score',r2_score(y_test,y_pred))
print('MSE',mean_absolute_error(y_test,y_pred))
```

```
R2 score 0.8073277448175185
MSE 0.21017827978646666
```

Decision Tree

```
step1 = ColumnTransformer(transformers=[
    ('col_trans',OneHotEncoder(sparse=False,drop='first'),[0,1,7,10,11])
],remainder='passthrough')

step2= DecisionTreeRegressor(max_depth=8)

pipe = Pipeline([
    ('step1',step1),
    ('step2',step2)
])

pipe.fit(X_train,y_train)

y_pred = pipe.predict(X_test)

print('R2_score',r2_score(y_test,y_pred))
print('MSE',mean_absolute_error(y_test,y_pred))
```

```
R2_score 0.8483547168609326
MSE 0.1782161621969353
```

THANK YOU

For more information
you can check out in my github account:-
https://github.com/ayushrawat0210

There are lots of calculations and
codes in this Project