Hands-on lab: Exploratory Data Analysis - Laptops Pricing dataset

Estimated time needed: **45** minutes

In this lab, you will use the skills acquired throughout the module, to explore the effect of different features on the price of laptops.

Objectives

After completing this lab you will be able to:

- Visualize individual feature patterns
- Run descriptive statistical analysis on the dataset
- Use groups and pivot tables to find the effect of categorical variables on price
- Use Pearson Correlation to measure the interdependence between variables

Setup

For this lab, we will be using the following libraries:

- `skillsnetwork` for downloading the data
- `pandas` for managing the data.
- `numpy` for mathematical operations.
- `scipy` for statistical operations.
- `seaborn` for visualizing the data.
- `matplotlib` for additional plotting tools.

Install Required Libraries

You can install the required libraries by simply running the `pip install` command with a `%` sign before it. For this environment, `seaborn` library requires installation.

In [1]:
```
import piplite
await piplite.install('seaborn')
```

Importing Required Libraries

We recommend you import all required libraries in one place (here):

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
%matplotlib inline
```

Import the dataset

You should download the modified version of the data set from the last module. Run the following code block to download the CSV file to this environment.

The functions below will download the dataset into your browser:

In [3]:
```
from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())
```

In [4]:
```
filepath="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/laptop_pricing_dataset_mod2.csv"
```

In [5]:
```
await download(filepath, "laptops.csv")
file_name="laptops.csv"
```

Import the file to a pandas dataframe.

In [6]:
```
df = pd.read_csv(file_name, header=0)
```

> Note: This version of the lab is working on JupyterLite, which requires the dataset to be downloaded to the interface. While working on the downloaded version of this notebook on their local machines, the learners can simply **skip the steps above**, and simply use the URL directly in the `pandas.read_csv()` function. You can uncomment and run the statements in the cell below.

In [ ]:
```
#filepath="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/laptop_pricing_dataset_mod2.csv
#df = pd.read_csv(filepath, header=None)
```

Print the first 5 entries of the dataset to confirm loading.

In [7]:
```
df.head(5)
```

Out[7]:

	Unnamed: 0.1	Unnamed: 0	Manufacturer	Category	GPU	OS	CPU_core	Screen_Size_inch	CPU_frequency	RAM_GB	Storage_GB_SSD	Weight_pounds	Price	Price-binned	S Fu
0	0	0	Acer	4	2	1	5	14.0	0.551724	8	256	3.52800	978	Low	
1	1	1	Dell	3	1	1	3	15.6	0.689655	4	256	4.85100	634	Low	
2	2	2	Dell	3	1	1	7	15.6	0.931034	8	256	4.85100	946	Low	
3	3	3	Dell	4	2	1	5	13.3	0.551724	8	128	2.69010	1244	Low	
4	4	4	HP	4	2	1	7	15.6	0.620690	8	256	4.21155	837	Low	

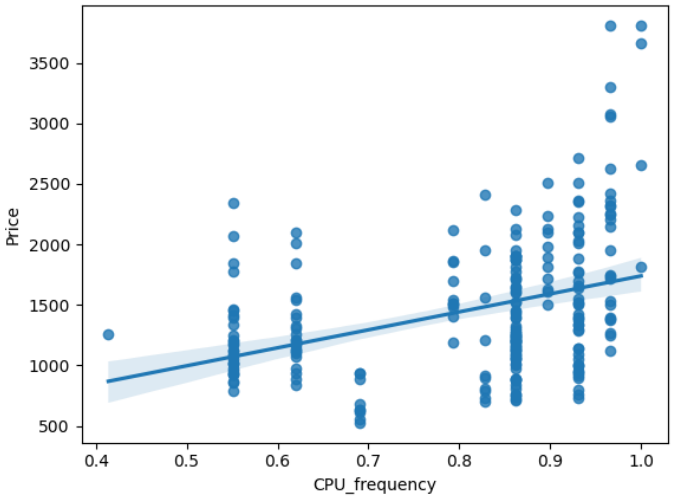
Task 1 - Visualize individual feature patterns

Continuous valued features

Generate regression plots for each of the parameters "CPU_frequency", "Screen_Size_inch" and "Weight_pounds" against "Price". Also, print the value of correlation of each feature with "Price".

In [8]:
```
# Write your code below and press Shift+Enter to execute
# CPU_frequency plot
sns.regplot(x="CPU_frequency",y="Price",data=df)
```

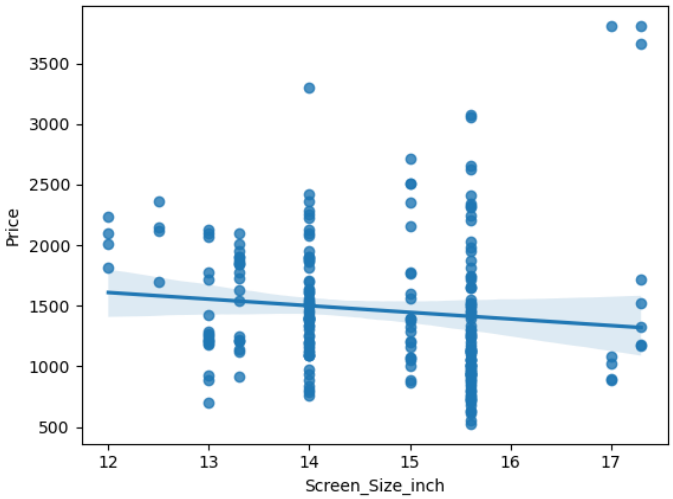
Out[8]:
```
<AxesSubplot:xlabel='CPU_frequency', ylabel='Price'>
```



▸ Click here for Solution

In [9]:
```
# Write your code below and press Shift+Enter to execute
# Screen_Size_inch plot
sns.regplot(x="Screen_Size_inch",y="Price",data=df)
```

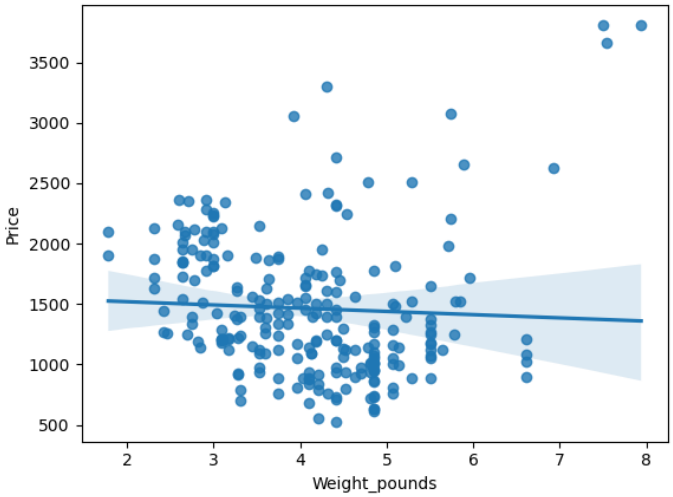
Out[9]:
```
<AxesSubplot:xlabel='Screen_Size_inch', ylabel='Price'>
```



▸ Click here for Solution

In [10]:
```
# Write your code below and press Shift+Enter to execute
# Weight_pounds plot
sns.regplot(x="Weight_pounds",y="Price",data=df)
```

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



▸ Click here for Solution

In [13]:
```
# Correlation values of the three attributes with Price
for item in ["CPU_frequency","Screen_Size_inch","Weight_pounds"]:
    print(f"Correlation {item} with price \n",df[[item,"Price"]].corr())
```

```
Correlation CPU_frequency with price
                CPU_frequency     Price
CPU_frequency       1.000000  0.366666
Price               0.366666  1.000000
Correlation Screen_Size_inch with price
                  Screen_Size_inch     Price
Screen_Size_inch          1.000000 -0.110644
Price                    -0.110644  1.000000
Correlation Weight_pounds with price
                Weight_pounds     Price
Weight_pounds       1.000000 -0.050312
Price              -0.050312  1.000000
```

▸ Click here for Solution

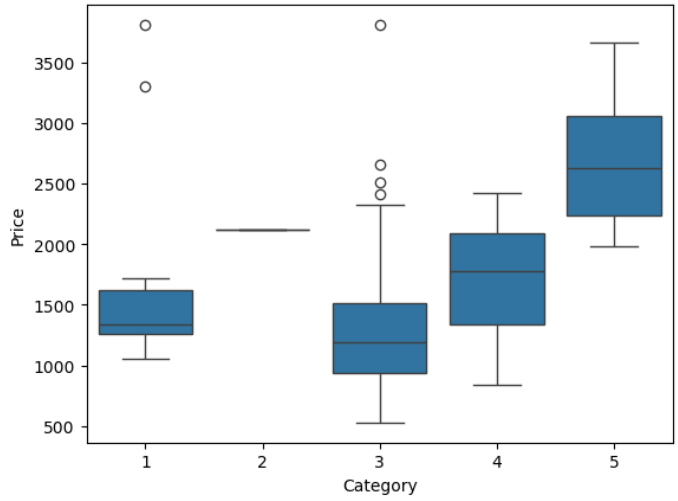
Interpretation: "CPU_frequency" has a 36% positive correlation with the price of the laptops. The other two parameters have weak correlation with price.

Categorical features

Generate Box plots for the different feature that hold categorical values. These features would be "Category", "GPU", "OS", "CPU_core", "RAM_GB", "Storage_GB_SSD"

In [14]:
```
# Write your code below and press Shift+Enter to execute
# Category Box plot
sns.boxplot(x="Category",y="Price",data=df)
```

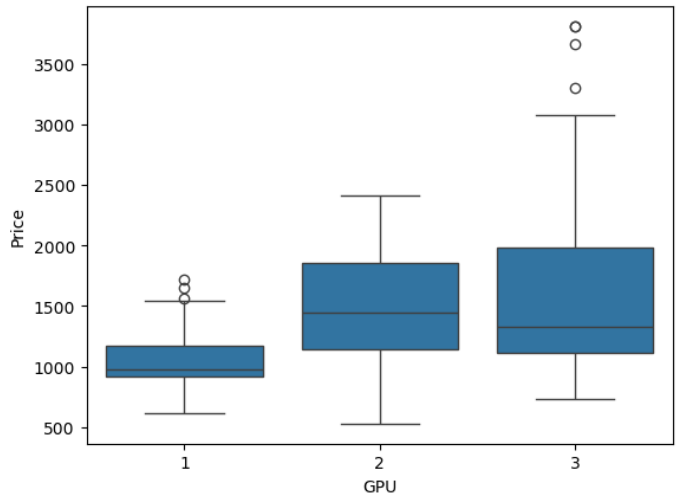
Out[14]: <AxesSubplot:xlabel='Category', ylabel='Price'>



▸ Click here for Solution

In [15]:
```
# Write your code below and press Shift+Enter to execute
# GPU Box plot
sns.boxplot(x="GPU",y="Price",data=df)
```

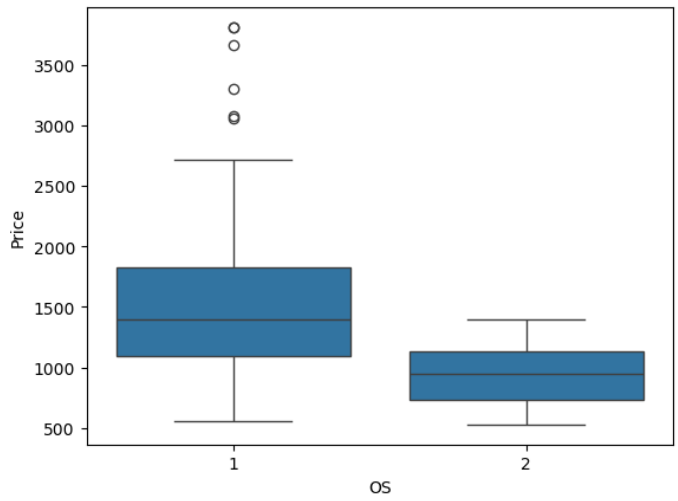
Out[15]: <AxesSubplot:xlabel='GPU', ylabel='Price'>



▸ Click here for Solution

In [16]:
```
# Write your code below and press Shift+Enter to execute
# OS Box plot
sns.boxplot(x="OS",y="Price",data=df)
```

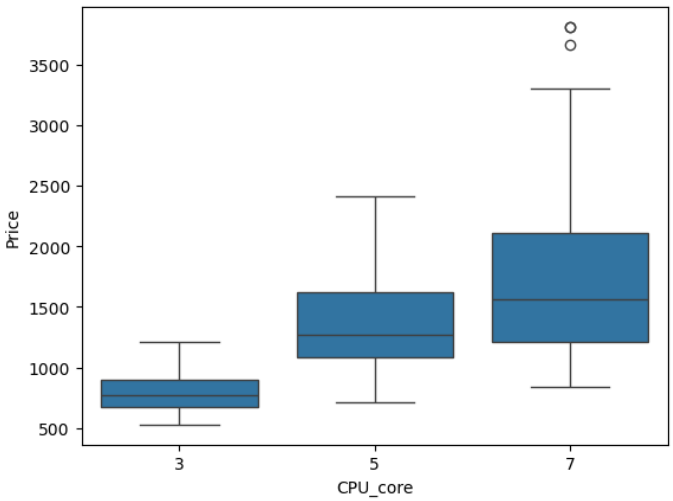
Out[16]: <AxesSubplot:xlabel='OS', ylabel='Price'>



▸ Click here for Solution

In [17]:
```
# Write your code below and press Shift+Enter to execute
# CPU_core Box plot
sns.boxplot(x="CPU_core",y="Price",data=df)
```

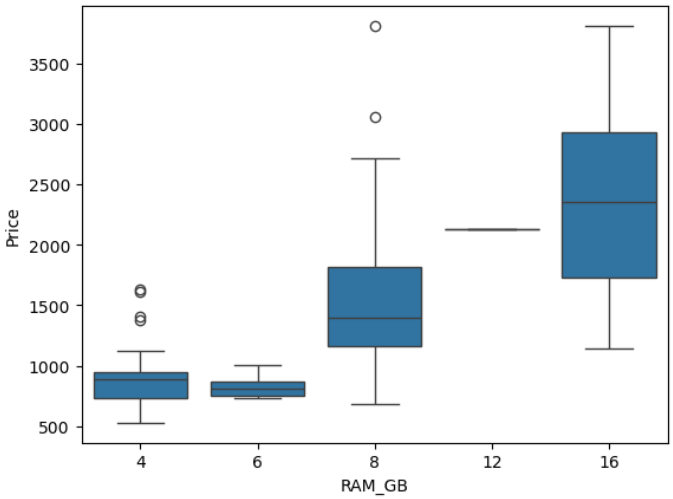
Out[17]: <AxesSubplot:xlabel='CPU_core', ylabel='Price'>



▸ Click here for Solution

In [18]:
```
# Write your code below and press Shift+Enter to execute
# RAM_GB Box plot
sns.boxplot(x="RAM_GB",y="Price",data=df)
```

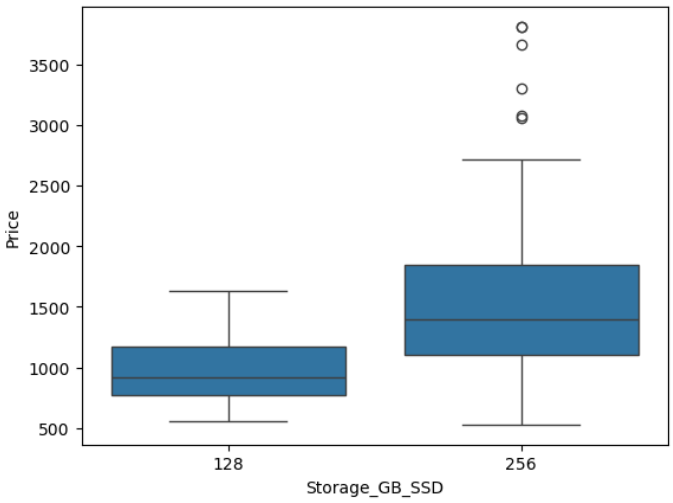
Out[18]: <AxesSubplot:xlabel='RAM_GB', ylabel='Price'>



▸ Click here for Solution

In [19]:
```
# Write your code below and press Shift+Enter to execute
# Storage_GB_SSD Box plot
sns.boxplot(x="Storage_GB_SSD",y="Price",data=df)
```

Out[19]: <AxesSubplot:xlabel='Storage_GB_SSD', ylabel='Price'>



▸ Click here for Solution

Task 2 - Descriptive Statistical Analysis

Generate the statistical description of all the features being used in the data set. Include "object" data types as well.

In [20]:
```
# Write your code below and press Shift+Enter to execute
df.describe(include="all")
```

Out[20]:

	Unnamed: 0.1	Unnamed: 0	Manufacturer	Category	GPU	OS	CPU_core	Screen_Size_inch	CPU_frequency	RAM_GB	Storage_GB_SSD	Weight_p
count	238.000000	238.000000	238	238.000000	238.000000	238.000000	238.000000	238.000000	238.000000	238.000000	238.000000	238.(
unique	NaN	NaN	11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
top	NaN	NaN	Dell	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
freq	NaN	NaN	71	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
mean	118.500000	118.500000	NaN	3.205882	2.151261	1.058824	5.630252	14.688655	0.813822	7.882353	245.781513	4.1
std	68.848868	68.848868	NaN	0.776533	0.638282	0.235790	1.241787	1.166045	0.141860	2.482603	34.765316	1.(
min	0.000000	0.000000	NaN	1.000000	1.000000	1.000000	3.000000	12.000000	0.413793	4.000000	128.000000	1.7
25%	59.250000	59.250000	NaN	3.000000	2.000000	1.000000	5.000000	14.000000	0.689655	8.000000	256.000000	3.2
50%	118.500000	118.500000	NaN	3.000000	2.000000	1.000000	5.000000	15.000000	0.862069	8.000000	256.000000	4.1
75%	177.750000	177.750000	NaN	4.000000	3.000000	1.000000	7.000000	15.600000	0.931034	8.000000	256.000000	4.8
max	237.000000	237.000000	NaN	5.000000	3.000000	2.000000	7.000000	17.300000	1.000000	16.000000	256.000000	7.9

▸ Click here for Solution

Task 3 - GroupBy and Pivot Tables

Group the parameters "GPU", "CPU_core" and "Price" to make a pivot table and visualize this connection using the pcolor plot.

In [27]:
```
# Write your code below and press Shift+Enter to execute
# Create the group
group_data = df[["GPU","CPU_core","Price"]].groupby(["GPU","CPU_core"],as_index=False).mean()
group_data
```

Out[27]:

	GPU	CPU_core	Price
0	1	3	769.250000
1	1	5	998.500000
2	1	7	1167.941176
3	2	3	785.076923
4	2	5	1462.197674
5	2	7	1744.621622
6	3	3	784.000000
7	3	5	1220.680000
8	3	7	1945.097561

▸ Click here for Solution

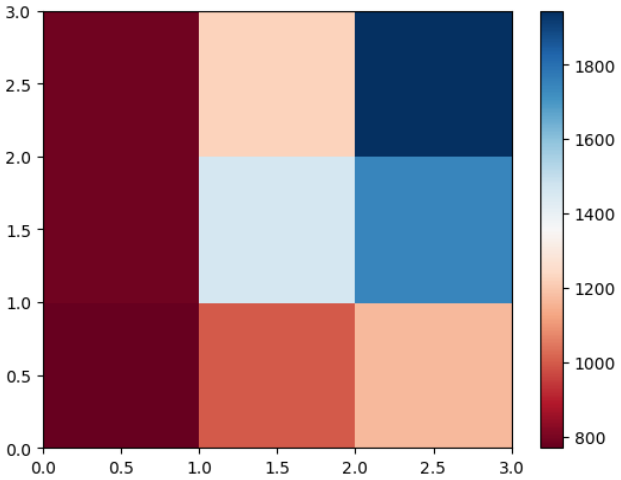
In [30]:
```
# Write your code below and press Shift+Enter to execute
# Create the Pivot table
df_pivot = group_data.pivot(index="GPU",columns="CPU_core")
df_pivot
```

Out[30]:

	Price		
CPU_core	**3**	**5**	**7**
GPU			
1	769.250000	998.500000	1167.941176
2	785.076923	1462.197674	1744.621622
3	784.000000	1220.680000	1945.097561

▸ Click here for Solution

In [34]:
```
# Write your code below and press Shift+Enter to execute
# Create the Plot
plt.pcolor(df_pivot,cmap='RdBu')
plt.colorbar()
plt.show()
```



▸ Click here for Solution

Task 4 - Pearson Correlation and p-values

Use the `scipy.stats.pearsonr()` function to evaluate the Pearson Coefficient and the p-values for each parameter tested above. This will help you determine the parameters most likely to have a strong effect on the price of the laptops.

In [35]:
```
# Write your code below and press Shift+Enter to execute
for item in df.select_dtypes('int','float').columns:
    p_coeff, p_val = stats.pearsonr(df[item],df["Price"])
    print(f"For {item} Coefficient = {p_coeff} P-Value = {p_val}")
```

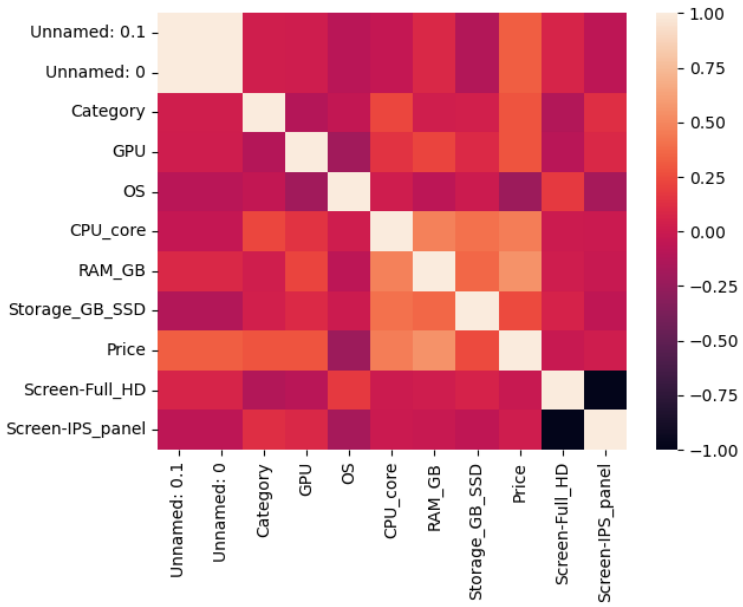
```
For Unnamed: 0.1 Coefficient = 0.32193291314756983 P-Value = 3.8510584639457855e-07
For Unnamed: 0 Coefficient = 0.32193291314756983 P-Value = 3.8510584639457855e-07
For Category Coefficient = 0.286242755812641 P-Value = 7.225696235806858e-06
For GPU Coefficient = 0.2882981988881427 P-Value = 6.166949698364507e-06
For OS Coefficient = -0.22172980114827356 P-Value = 0.0005696642559246817
For CPU_core Coefficient = 0.45939777733551174 P-Value = 7.912950127008979e-14
For RAM_GB Coefficient = 0.5492972971857849 P-Value = 3.6815606288424503e-20
For Storage_GB_SSD Coefficient = 0.24342075521810297 P-Value = 0.00014898923191724168
For Price Coefficient = 0.9999999999999999 P-Value = 0.0
For Screen-Full_HD Coefficient = -0.021074572471421686 P-Value = 0.746356830977638
For Screen-IPS_panel Coefficient = 0.02107457247142169 P-Value = 0.746356830977638
```

▸ Click here for Solution

Heatmap

In [37]:
```
sns.heatmap(df.select_dtypes('int','float').corr())
```

Out[37]: <AxesSubplot:>



Congratulations! You have completed the lab

Authors

Abhishek Gagneja

Vicky Kuo

Copyright © 2023 IBM Corporation. All rights reserved.