Hands-on Practice Lab: Data Wrangling

In this lab, we will address the issues of handling missing data, correct the data type of the dataframe attribute and execute the processes of data standardization and data normalization on specific attributes of the dataset.

Objectives

- Handle missing data in different ways
- Correct the data type of different data values as per requirement
- Standardize and normalize the appropriate data attributes
- Visualize the data as grouped bar graph using Binning
- Converting a categorical data into numerical indicator variables(or dummy_variables)

Setup

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

- `skillsnetwork` to download the dataset
- `pandas` (https://pandas.pydata.org/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_terr SkillsNetwork-Channel-SkillsNetworkCoursesIBMML0187ENSkillsNetwork31430127-2021-01-01) for managing the data.
- `numpy` (https://numpy.org/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_terr SkillsNetwork-Channel-SkillsNetworkCoursesIBMML0187ENSkillsNetwork31430127-2021-01-01) for mathematical operations.
- `matplotlib` (https://matplotlib.org/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_terr SkillsNetwork-Channel-SkillsNetworkCoursesIBMML0187ENSkillsNetwork31430127-2021-01-01) for additional plotting tools.

Importing Required Libraries

In [63]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

Download and save the dataset

In [64]:
```
file_path= "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clo
```

In [65]:
```
file_name="laptops.csv"
```

First we load data into a `pandas.DataFrame` :

In [66]:
```
df = pd.read_csv(file_name)
df.head()
```

Out[66]:

	Unnamed: 0	Manufacturer	Category	Screen	GPU	OS	CPU_core	Screen_Size_cm	CPU_f
0	0	Acer	4	IPS Panel	2	1	5	35.560	
1	1	Dell	3	Full HD	1	1	3	39.624	
2	2	Dell	3	Full HD	1	1	7	39.624	
3	3	Dell	4	IPS Panel	2	1	5	33.782	
4	4	HP	4	Full HD	2	1	7	39.624	

In [67]:
```
# Drop the first column (assuming it's the index)
df = df.drop(df.columns[0], axis=1)  # Drop by column position
```

In [68]:
```
df.head()
```

Out[68]:

	Manufacturer	Category	Screen	GPU	OS	CPU_core	Screen_Size_cm	CPU_frequency	R
0	Acer	4	IPS Panel	2	1	5	35.560	1.6	
1	Dell	3	Full HD	1	1	3	39.624	2.0	
2	Dell	3	Full HD	1	1	7	39.624	2.7	
3	Dell	4	IPS Panel	2	1	5	33.782	1.6	
4	HP	4	Full HD	2	1	7	39.624	1.8	

Verify loading by displaying the dataframe summary using `dataframe.info()`

In [69]:

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 238 entries, 0 to 237
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Manufacturer    238 non-null    object 
 1   Category        238 non-null    int64  
 2   Screen          238 non-null    object 
 3   GPU             238 non-null    int64  
 4   OS              238 non-null    int64  
 5   CPU_core        238 non-null    int64  
 6   Screen_Size_cm  234 non-null    float64
 7   CPU_frequency   238 non-null    float64
 8   RAM_GB          238 non-null    int64  
 9   Storage_GB_SSD  238 non-null    int64  
 10  Weight_kg       233 non-null    float64
 11  Price           238 non-null    int64  
dtypes: float64(3), int64(7), object(2)
memory usage: 22.4+ KB
None
```

Note that we can update the `Screen_Size_cm` column such that all values are rounded to nearest 2 decimal places by using `numpy.round()`

In [70]:

```
df[['Screen_Size_cm']] = np.round(df[['Screen_Size_cm']],2)
df.head()
```

Out[70]:

	Manufacturer	Category	Screen	GPU	OS	CPU_core	Screen_Size_cm	CPU_frequency	R
0	Acer	4	IPS Panel	2	1	5	35.56	1.6	
1	Dell	3	Full HD	1	1	3	39.62	2.0	
2	Dell	3	Full HD	1	1	7	39.62	2.7	
3	Dell	4	IPS Panel	2	1	5	33.78	1.6	
4	HP	4	Full HD	2	1	7	39.62	1.8	

Task - 1

Evaluate the dataset for missing data

Pandas uses NaN and Null values interchangeably. This means, you can just identify the entries having Null values. Write a code that identifies which columns have missing data.

```
In [71]: missing_data = df.isnull()
         print(missing_data.head())
         for column in missing_data.columns.values.tolist():
             print(column)
             print (missing_data[column].value_counts())
             print("")
```

```
   Manufacturer  Category  Screen    GPU     OS  CPU_core  Screen_Size_cm  \
0         False     False   False  False  False     False           False   
1         False     False   False  False  False     False           False   
2         False     False   False  False  False     False           False   
3         False     False   False  False  False     False           False   
4         False     False   False  False  False     False           False   

   CPU_frequency  RAM_GB  Storage_GB_SSD  Weight_kg  Price  
0          False   False           False      False  False  
1          False   False           False      False  False  
2          False   False           False      False  False  
3          False   False           False      False  False  
4          False   False           False      False  False  
Manufacturer
Manufacturer
False    238
Name: count, dtype: int64

Category
Category
False    238
Name: count, dtype: int64

Screen
Screen
False    238
Name: count, dtype: int64

GPU
GPU
False    238
Name: count, dtype: int64

OS
OS
False    238
Name: count, dtype: int64

CPU_core
CPU_core
False    238
Name: count, dtype: int64

Screen_Size_cm
Screen_Size_cm
False    234
True       4
Name: count, dtype: int64

CPU_frequency
CPU_frequency
False    238
Name: count, dtype: int64

RAM_GB
RAM_GB
False    238
Name: count, dtype: int64

Storage_GB_SSD
```

```
Storage_GB_SSD
False    238
Name: count, dtype: int64

Weight_kg
Weight_kg
False    233
True       5
Name: count, dtype: int64

Price
Price
False    238
Name: count, dtype: int64
```

Task - 2

Replace with mean

Missing values in attributes that have continuous data are best replaced using Mean value. We note that values in "Weight_kg" attribute are continuous in nature, and some values are missing. Therefore, we will write a code to replace the missing values of weight with the average value of the attribute.

In [72]:

```python
# replacing missing data with mean
avg_weight=df['Weight_kg'].astype('float').mean(axis=0)
df["Weight_kg"].replace(np.nan, avg_weight, inplace=True)

# astype() function converts the values to the desired data type
# axis=0 indicates that the mean value is to calculated across all column el
df.head()
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_42724\2523453082.py:3: FutureWa
rning: A value is trying to be set on a copy of a DataFrame or Series thro
ugh chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never wor
k because the intermediate object on which we are setting values always be
haves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df["Weight_kg"].replace(np.nan, avg_weight, inplace=True)
```

Out[72]:

	Manufacturer	Category	Screen	GPU	OS	CPU_core	Screen_Size_cm	CPU_frequency	R
0	Acer	4	IPS Panel	2	1	5	35.56	1.6	
1	Dell	3	Full HD	1	1	3	39.62	2.0	
2	Dell	3	Full HD	1	1	7	39.62	2.7	
3	Dell	4	IPS Panel	2	1	5	33.78	1.6	
4	HP	4	Full HD	2	1	7	39.62	1.8	

Replace with the most frequent value

Missing values in attributes that have categorical data are best replaced using the most frequent value. We note that values in "Screen_Size_cm" attribute are categorical in nature, and some values are missing. Therefore, write a code to replace the missing values of Screen Size with the most frequent value of the attribute.

In [73]:
```
# replacing missing data with mode
common_screen_size = df['Screen_Size_cm'].value_counts().idxmax()

df["Screen_Size_cm"].replace(np.nan, common_screen_size, inplace = True)
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_42724\667827415.py:4: FutureWar
ning: A value is trying to be set on a copy of a DataFrame or Series throu
gh chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never wor
k because the intermediate object on which we are setting values always be
haves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df["Screen_Size_cm"].replace(np.nan, common_screen_size, inplace = True)
```

In [74]:
```
df.head()
```

Out[74]:

	Manufacturer	Category	Screen	GPU	OS	CPU_core	Screen_Size_cm	CPU_frequency	R
0	Acer	4	IPS Panel	2	1	5	35.56	1.6	
1	Dell	3	Full HD	1	1	3	39.62	2.0	
2	Dell	3	Full HD	1	1	7	39.62	2.7	
3	Dell	4	IPS Panel	2	1	5	33.78	1.6	
4	HP	4	Full HD	2	1	7	39.62	1.8	

Task - 3

Fixing the data types

Both "Weight_kg" and "Screen_Size_cm" are seen to have the data type "int", while both of them should be having a data type of "float". we will write a code to fix the data type of these two columns.

In [75]:
```
# fix columns types isue
df[["Weight_kg","Screen_Size_cm"]] = df[["Weight_kg","Screen_Size_cm"]].asty
```

Task - 4

Data Standardization

The value of Screen_size usually has a standard unit of inches. Similarly, weight of the laptop is needed to be in pounds. We will use the below mentioned units of conversion and will write a code to modify the columns of the dataframe accordingly. Update their names as well.

```
1 inch = 2.54 cm
1 kg   = 2.205 pounds
```

In [76]:
```
# Data standardization: convert weight from kg to pounds
df["Weight_kg"] = df["Weight_kg"]*2.205
df.rename(columns={'Weight_kg':'Weight_pounds'}, inplace=True)

# Data standardization: convert screen size from cm to inch
df["Screen_Size_cm"] = df["Screen_Size_cm"]/2.54
df.rename(columns={'Screen_Size_cm':'Screen_Size_inch'}, inplace=True)
```

In [77]:
```
df.head()
```

Out[77]:

	Manufacturer	Category	Screen	GPU	OS	CPU_core	Screen_Size_inch	CPU_frequency
0	Acer	4	IPS Panel	2	1	5	14.000000	1.6
1	Dell	3	Full HD	1	1	3	15.598425	2.0
2	Dell	3	Full HD	1	1	7	15.598425	2.7
3	Dell	4	IPS Panel	2	1	5	13.299213	1.6
4	HP	4	Full HD	2	1	7	15.598425	1.8

Data Normalization

Often it is required to normalize a continuous data attribute. We will write a code to normalize the "CPU_frequency" attribute with respect to the maximum value available in the dataset.

In [78]:
```
# normalize "CPU_frequency"
df['CPU_frequency'] = df['CPU_frequency']/df['CPU_frequency'].max()
```

Task - 5

Binning

Binning is a process of creating a categorical attribute which splits the values of a continuous data into a specified number of groups. In this case, we will write a code to create 3 bins for the attribute "Price". These bins would be named "Low", "Medium" and "High". The new attribute will be named "Price-binned".

In [79]:

```
bins = np.linspace(min(df["Price"]), max(df["Price"]), 4)
group_names = ['Low', 'Medium', 'High']
df['Price-binned'] = pd.cut(df['Price'], bins, labels=group_names, include_
```

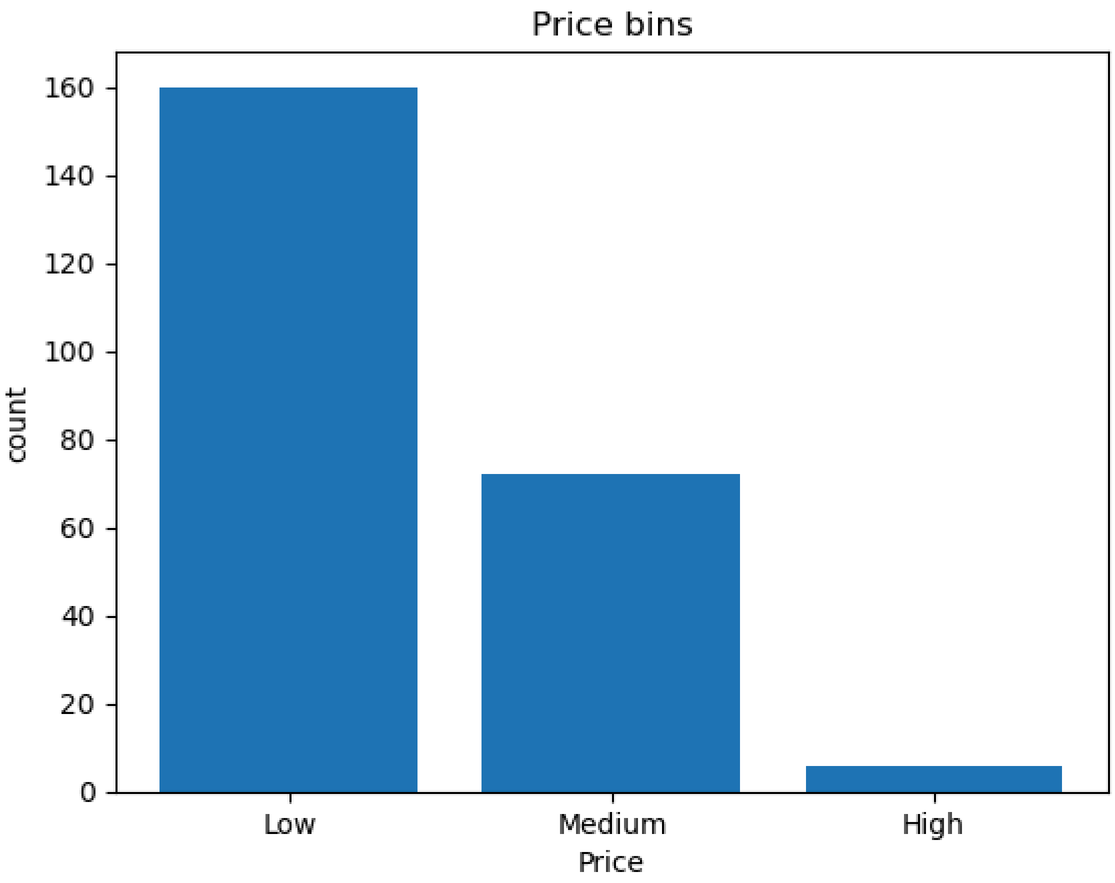
Also, plot the bar graph of these bins.

In [80]:

```
plt.bar(group_names, df["Price-binned"].value_counts())
plt.xlabel("Price")
plt.ylabel("count")
plt.title("Price bins")
```

Out[80]:

```
Text(0.5, 1.0, 'Price bins')
```



Task - 6

Indicator variables

Convert the "Screen" attribute of the dataset into 2 indicator variables, "Screen-IPS_panel" and "Screen-Full_HD". Then drop the "Screen" attribute from the dataset.

In [81]:
```
#Indicator Variable: Screen
dummy_variable_1 = pd.get_dummies(df["Screen"])
dummy_variable_1.rename(columns={'IPS Panel':'Screen-IPS_panel', 'Full HD':
dummy_variable_1 = dummy_variable_1.astype(int)
df = pd.concat([df, dummy_variable_1], axis=1)

# drop original column "Screen" from "df"
df.drop("Screen", axis = 1, inplace=True)
```

This version of the dataset, now finalized, is the one w'll be using in all subsequent modules.

Print the content of dataframe.head() to verify the changes that were made to the dataset.

In [82]:
```
print(df.head())
```

```
  Manufacturer  Category  GPU  OS  CPU_core  Screen_Size_inch  CPU_frequen
cy  \
0         Acer         4    2   1         5         14.000000       0.5517
24
1         Dell         3    1   1         3         15.598425       0.6896
55
2         Dell         3    1   1         7         15.598425       0.9310
34
3         Dell         4    2   1         5         13.299213       0.5517
24
4           HP         4    2   1         7         15.598425       0.6206
90

   RAM_GB  Storage_GB_SSD  Weight_pounds  Price Price-binned  Screen-Full_
HD  \
0       8             256        3.52800    978          Low
0
1       4             256        4.85100    634          Low
1
2       8             256        4.85100    946          Low
1
3       8             128        2.69010   1244          Low
0
4       8             256        4.21155    837          Low
1

   Screen-IPS_panel
0                 1
1                 0
2                 0
3                 1
4                 0
```

In [83]:
```
df.to_csv('clean_laptops_df.csv')
```