Generative AI :Building the prompt: Standardization and Normalization

import libraries

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

Read the CSV file

```
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())

file_path = "https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/
laptop_pricing_dataset_mod1.csv"

await download(file_path, "laptops.csv")
file_name="laptops.csv"

df = pd.read_csv(file_name, header=0)

df.head()
```

```
   Unnamed: 0 Manufacturer  Category     Screen  GPU  OS  CPU_core  \
0           0         Acer         4  IPS Panel    2   1         5
1           1         Dell         3    Full HD    1   1         3
2           2         Dell         3    Full HD    1   1         7
3           3         Dell         4  IPS Panel    2   1         5
4           4           HP         4    Full HD    2   1         7

   Screen_Size_cm  CPU_frequency  RAM_GB  Storage_GB_SSD  Weight_kg
Price
0          35.560            1.6       8             256       1.60
978
1          39.624            2.0       4             256       2.20
634
2          39.624            2.7       8             256       2.20
946
3          33.782            1.6       8             128       1.22
```

```
1244
4          39.624             1.8       8              256        1.91
837
```

Identify missing values

```python
# Create a sample DataFrame with missing values
data = {
    'Name': ['John', 'Mary', 'David', 'Jane', 'Bob'],
    'Age': [25, 30, 35, None, 40],
    'Country': ['USA', 'Canada', 'Mexico', None, 'UK'],
    'Score': [90, 85, 95, 80, 92]
}
df = pd.DataFrame(data)

# Print the original DataFrame
print("Original DataFrame:")
print(df)

# Identify columns with missing values
missing_columns = df.isnull().sum()

# Print the columns with missing values
print("\nColumns with missing values:")
print(missing_columns)

# Identify columns with missing values by value
missing_by_value = df.isnull().sum().idxmax()

# Print the columns with missing values by value
print("\nColumns with missing values by value:")
print(missing_by_value)
```

```
Original DataFrame:
    Name   Age Country  Score
0   John  25.0     USA     90
1   Mary  30.0  Canada     85
2  David  35.0  Mexico     95
3   Jane   NaN    None     80
4    Bob  40.0      UK     92

Columns with missing values:
Name       0
Age        1
Country    1
Score      0
dtype: int64

Columns with missing values by value:
Age
```

Replace the missing values

```
import numpy as np

# Create a sample DataFrame
data = {
    "Screen_Size_cm": [100, 120, 110, 130, 105, 125, 115, 135, 130,
140],
    "Weight_kg": [70, 80, 75, 90, 65, 85, 70, 95, 80, 90]
}
df = pd.DataFrame(data)

# Replace missing values in Screen_Size_cm with the most frequent
value
df["Screen_Size_cm"] =
df["Screen_Size_cm"].fillna(df["Screen_Size_cm"].mode().iloc[0])

# Replace missing values in Weight_kg with the mean value
df["Weight_kg"] = df["Weight_kg"].fillna(df["Weight_kg"].mean())

print(df)
```

```
   Screen_Size_cm  Weight_kg
0             100         70
1             120         80
2             110         75
3             130         90
4             105         65
5             125         85
6             115         70
7             135         95
8             130         80
9             140         90
```

Change data type to float

```
# Create a sample DataFrame
data = {
    "Screen_Size_cm": [120, 150, 180, 200],
    "Weight_kg": [50, 60, 70, 80]
}
df = pd.DataFrame(data)

# Print the original DataFrame
print("Original DataFrame:")
print(df)

# Change the data type of "Screen_Size_cm" and "Weight_kg" to float
df["Screen_Size_cm"] = df["Screen_Size_cm"].astype(float)
df["Weight_kg"] = df["Weight_kg"].astype(float)
```

```
# Print the updated DataFrame
print("\nUpdated DataFrame:")
print(df)
```

```
Original DataFrame:
   Screen_Size_cm  Weight_kg
0             120         50
1             150         60
2             180         70
3             200         80

Updated DataFrame:
   Screen_Size_cm  Weight_kg
0           120.0       50.0
1           150.0       60.0
2           180.0       70.0
3           200.0       80.0
```

Modify contents

```
# Create a sample DataFrame
data = {
    'Screen_Size_cm': [120, 150, 180, 200, 220],
    'Weight_kg': [10, 15, 20, 25, 30]
}
df = pd.DataFrame(data)

# Convert 'Screen_Size_cm' to 'Screen_Size_inch'
df['Screen_Size_inch'] = df['Screen_Size_cm'] * 2.54

# Convert 'Weight_kg' to 'Weight_pounds'
df['Weight_pounds'] = df['Weight_kg'] * 2.20462

print(df)
```

```
   Screen_Size_cm  Weight_kg  Screen_Size_inch  Weight_pounds
0             120         10             304.8        22.0462
1             150         15             381.0        33.0693
2             180         20             457.2        44.0924
3             200         25             508.0        55.1155
4             220         30             558.8        66.1386
```

Normalize the contents

```
data = {
    'CPU_frequency': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
}
df = pd.DataFrame(data)

# Print the original DataFrame
```

```
print("Original DataFrame:")
print(df)

# Normalize the 'CPU_frequency' column
df['CPU_frequency'] = df['CPU_frequency'] / df['CPU_frequency'].max()

# Print the normalized DataFrame
print("\nNormalized DataFrame:")
print(df)
```

```
Original DataFrame:
   CPU_frequency
0             10
1             20
2             30
3             40
4             50
5             60
6             70
7             80
8             90
9            100

Normalized DataFrame:
   CPU_frequency
0            0.1
1            0.2
2            0.3
3            0.4
4            0.5
5            0.6
6            0.7
7            0.8
8            0.9
9            1.0
```

Convert dataframe

```
data = {
    "Screen": ["Screen1", "Screen2", "Screen1", "Screen3", "Screen2",
"Screen1", "Screen3", "Screen2", "Screen1"]
}
df = pd.DataFrame(data)

# Convert 'Screen' column to indicator variables
df1 = df.assign(Screen=df["Screen"].map({v: 1 if v in ["Screen1",
"Screen2"] else 0 for v in df["Screen"]}).reset_index(drop=True))

# Append df1 to the original DataFrame
df = pd.concat([df, df1])
```

```
# Drop the original 'Screen' column
df = df.drop("Screen", axis=1)

df1.head()
```

```
   Screen
0       1
1       1
2       1
3       0
4       1
```

Modify unit of 'price' from USD to Euros

```
# Create a sample DataFrame
data = {
    'Price_USD': [100, 120, 150, 180, 200],
    'Price_EUR': [90, 110, 140, 160, 180]
}
df = pd.DataFrame(data)

# Convert 'Price_USD' to 'Price_EUR'
df['Price_EUR'] = df['Price_USD'] * 0.88

print(df)
```

```
   Price_USD  Price_EUR
0        100       88.0
1        120      105.6
2        150      132.0
3        180      158.4
4        200      176.0
```

Normalisation

```
data = {
    "CPU_frequency": [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
}
df = pd.DataFrame(data)

# Print the original DataFrame
print("Original DataFrame:")
print(df)

# Normalize the CPU_frequency column
df['CPU_frequency_normalized'] = (df['CPU_frequency'] -
df['CPU_frequency'].min()) / (df['CPU_frequency'].max() -
df['CPU_frequency'].min())
```

```python
# Print the normalized DataFrame
print("\nNormalized DataFrame:")
print(df)
```

```
Original DataFrame:
   CPU_frequency
0             10
1             20
2             30
3             40
4             50
5             60
6             70
7             80
8             90
9            100

Normalized DataFrame:
   CPU_frequency  CPU_frequency_normalized
0             10                  0.000000
1             20                  0.111111
2             30                  0.222222
3             40                  0.333333
4             50                  0.444444
5             60                  0.555556
6             70                  0.666667
7             80                  0.777778
8             90                  0.888889
9            100                  1.000000
```

```python
df['CPU_frequency'].min()
```

```
10
```

```python
df['CPU_frequency'].max()
```

```
100
```