Prompts to Read CSV

Write a Python code that can perform the following tasks.

Read the CSV file, located on a given file path, into a Pandas data frame, assuming that the first rows of the file are the headers for the data.

```
import pandas as pd


# Specify the file path

file_path = "path/to/your/file.csv"


# Read the CSV file into a Pandas data frame

df = pd.read_csv(file_path)


# Assuming the first rows of the file are the headers, you don't need to specify any additional parameters


# Additional details:

# - The `pd.read_csv()` function is used to read a CSV file into a Pandas data frame.

# - By default, it assumes that the first row of the file contains the headers for the data.

# - If your file doesn't have headers, you can specify `header=None` as an additional parameter.

# - You can also specify other parameters, such as `sep` to specify the delimiter used in the file.

# - Make sure you have the Pandas library installed in your Python environment before running this code.
```

Prompt for Missing Data

Write a Python code that identifies the columns with missing values in a pandas data frame.

```
import pandas as pd


# Assuming you already have a Pandas data frame named 'df'


# Identify columns with missing values
columns_with_missing_values = df.columns[df.isnull().any()]


# Additional details:
# - The `df.isnull()` function returns a Boolean data frame where each cell is True if it contains a missing value (NaN), and False otherwise.
# - The `df.columns` attribute returns the column labels of the data frame.
# - The `.any()` method returns a Boolean Series indicating whether any value in the given axis (in this case, columns) is True.
# - Finally, the `.columns` attribute is used to retrieve the column labels where the condition is True.


# You can now use the 'columns_with_missing_values' variable to further analyze or handle the columns with missing values.
```

Part 2

Write a Python code to replace the missing values in a pandas data frame, per the following guidelines.

1. For a categorical attribute "Screen_Size_cm", replace the missing values with the most frequent value in the column.

1. 2. For a continuous value attribute "Weight_kg", replace the missing values with the mean value of the entries in the column.

```
# Replace missing values in the 'Screen_Size_cm' column with the most frequent value
most_frequent_value = df['Screen_Size_cm'].mode()[0]
df['Screen_Size_cm'].fillna(most_frequent_value, inplace=True)

# Replace missing values in the 'Weight_kg' column with the mean value
mean_value = df['Weight_kg'].mean()
df['Weight_kg'].fillna(mean_value, inplace=True)
```

Building the prompt: Modify data type

Write a Python code snippet to change the data type of the attributes "Screen_Size_cm" and "Weight_kg" of a data frame to float.

```
# 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)
```

Building the prompt: Standardization and Normalization

Write a Python code to modify the contents under the following attributes of the data frame as required.

1. Data under 'Screen_Size_cm' is assumed to be in centimeters. Convert this data into inches. Modify the name of the attribute to 'Screen_Size_inch'.

2. Data under 'Weight_kg' is assumed to be in kilograms. Convert this data into pounds. Modify the name of the attribute to 'Weight_pounds'.

```
# Convert 'Screen_Size_cm' from centimeters to inches and modify the attribute name
```

```
df['Screen_Size_inch'] = df['Screen_Size_cm'] * 0.393701
df.drop('Screen_Size_cm', axis=1, inplace=True)

# Convert 'Weight_kg' from kilograms to pounds and modify the attribute name
df['Weight_pounds'] = df['Weight_kg'] * 2.20462
df.drop('Weight_kg', axis=1, inplace=True)
```

Write a Python code to normalize the content under the attribute "CPU_frequency" in a data frame df concerning its maximum value. Make changes to the original data, and do not create a new attribute.

```
# Normalize the content under 'CPU_frequency' with respect to its maximum value
max_value = df['CPU_frequency'].max()
df['CPU_frequency'] = df['CPU_frequency'] / max_value
```

Building the prompt: Categorical to numerical

Write a Python code to perform the following tasks.

1. Convert a data frame df attribute "Screen", into indicator variables, saved as df1, with the naming convention "Screen_<unique value of the attribute>".

2. Append df1 into the original data frame df.

3. Drop the original attribute from the data frame df.

```
# Convert the 'Screen' attribute into indicator variables
df1 = pd.get_dummies(df['Screen'], prefix='Screen')
```

```
# Append df1 into the original data frame df
df = pd.concat([df, df1], axis=1)


# Drop the original 'Screen' attribute from the data frame
df.drop('Screen', axis=1, inplace=True)
```