**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
2. most\_frequent\_value = df['Screen\_Size\_cm'].mode()[0]
3. df['Screen\_Size\_cm'].fillna(most\_frequent\_value, inplace=True)
4. # Replace missing values in the 'Weight\_kg' column with the mean value
5. mean\_value = df['Weight\_kg'].mean()
6. 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

1. df['Screen\_Size\_cm'] = df['Screen\_Size\_cm'].astype(float)
2. 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)