**Data Cleaning Modifications**

Since the data cleaning process was originally designed with machine learning in mind, I had to adjust certain aspects to better suit the creation of tables for reporting. Below are the key modifications:

* **Commented Out One-Hot Encoding for Categorical Columns:**  
  The original code included one-hot encoding for categorical columns (Company, TypeName, Gpu\_Brand, and OS), which would have split these columns into multiple binary columns. To preserve the original structure of these categorical variables, I commented out this section of the code:

# df = df.join(pd.get\_dummies(df['Company'], dtype=int))

# df = df.drop("Company", axis=1)

# df = df.join(pd.get\_dummies(df['TypeName'], dtype=int))

# df = df.drop("TypeName", axis=1)

# df = df.join(pd.get\_dummies(df['Gpu\_Brand'], dtype=int))

# df = df.join(pd.get\_dummies(df['OS'], dtype=int))

* **Formatted the ppi Column:**  
  To ensure consistent formatting, I rounded the ppi values to two decimal places:

df['ppi'] = df['ppi'].round(2)

* **Converted and Formatted the Price Column:**  
  The Price column, originally in Indian Rupees, was converted to USD by multiplying by a conversion factor of 0.01176. I also rounded the resulting values to two decimal places for consistency:

df['Price'] = (df['Price'] \* 0.01176).round(2)

* **Removed an Invalid Weight Entry:**  
  A row in the Weight column had an unrealistic value of 0.0002. I identified and removed this row to maintain data integrity:

if (df['Weight'] == 0.0002).any():

weight\_row\_index = df[df['Weight'] == 0.0002].index[0]

df = df.drop(index=weight\_row\_index)

* **Saved the Cleaned Data to a CSV File:**  
  After making the necessary changes, I saved the cleaned dataset into a CSV file. This allowed for easier handling and processing when creating the tables:

new\_file\_path = '/content/drive/MyDrive/Colab Notebooks/WIA1007\_Group\_Assignment/laptopData\_table\_cleaned.csv'

df.to\_csv(new\_file\_path, index=False)

These adjustments ensured that the dataset was cleaned and formatted in a way that made it suitable for generating the tables in the next sections.

**Table 1: Data Properties**

Using Python's default libraries, I extracted data type, minimum value, maximum value (top value), unique values, and null values for each column. The types of data and measurement levels were determined based on the data types. During the data cleaning process, all units for the columns were removed. To provide a complete summary, I manually added the units back for each column based on their context and relevance.

The range was calculated as the difference between the minimum and maximum values, which is displayed for numerical columns. Outliers were detected using the Interquartile Range (IQR) method, where values below Q1−(1.5×IQR)Q1 - (1.5 \times \text{IQR})Q1−(1.5×IQR) or above Q3+(1.5×IQR)Q3 + (1.5 \times \text{IQR})Q3+(1.5×IQR) are considered outliers. Outlier ranges are only shown if outliers exist in the data.

I consolidated all variables into a table according to their column names and saved it as a CSV file. Additionally, I used the IPython.display library to display the CSV file as an HTML-like table, allowing me to apply HTML commands for formatting and improved readability.

**Table 2: Statistics**

This table provides a detailed summary of statistical metrics, including frequency, data completeness, percentiles (25th, 50th, 75th), mean, median, mode, standard deviation, variance, skewness, and kurtosis. The table accounts for two types of data: categorical and numerical.

* Categorical Data: For categorical variables, only frequency, data completeness, and mode are displayed. Frequency values are calculated by normalizing the value counts (i.e., dividing by the total number of observations) and multiplying by 100 to express them as percentages.
* Numerical Data: For numerical variables, all indicated statistics are calculated, excluding frequency. Most of these metrics are computed directly using Python's default libraries.

To format the frequency values, I stored newline characters (\n) in the CSV file to separate the frequencies of unique values. Later, I replaced “\n” with “<br>” so that, when displayed in Google Colab, the table correctly renders each frequency value on a new line. This enhances the readability of the frequency distribution for categorical variables.