# 

DATA DYNAMOS

Exploring Real-World Data Using Python

Table of Contents

[0](#_Toc181614214)

[OBJECTIVE 2](#_Toc181614215)

[Why do we need to clean data? 3](#_Toc181614216)

[STEP 1 5](#_Toc181614217)

[STEP 2 5](#_Toc181614218)

[IMPORTING PANDAS' LIBRARY 5](#_Toc181614219)

[STEP 3 6](#_Toc181614220)

[DATA ANALYSIS 6](#_Toc181614221)

[STEP 4 7](#_Toc181614222)

[DATA CLEANING (REMOVING / HANDLING MISSING VALUES) 7](#_Toc181614223)

[STEP 5 7](#_Toc181614224)

[ELIMINATE ANY IRRELEVANT COLUMNS 7](#_Toc181614225)

[STEP 6 8](#_Toc181614226)

[BAR GRAPH 8](#_Toc181614227)

[PIE CHART 9](#_Toc181614228)

[CONCLUSION 10](#_Toc181614229)

# OBJECTIVE

The purpose of this project is to analyze the \*\*StudentPerformanceFactors\*\* dataset, implementing data handling, cleaning, basic analysis, and visualization techniques in Python. By following these steps, we demonstrate the use of Python for data-driven insights, preparing us for more advanced data tasks.

# Why do we need to clean data?

Data cleaning is a crucial step in the data analysis process, especially when working with Python.

1. **Data Quality**: Raw data often contains errors, inconsistencies, and inaccuracies. Cleaning ensures high-quality data for analysis.
2. **Missing Values**: Datasets frequently have missing values. Cleaning helps identify and handle these gaps appropriately.
3. **Inconsistent Formatting**: Data can have inconsistent formats (e.g., date formats, categorical values). Standardizing these formats is essential for accurate analysis.
4. **Outliers**: Outliers can skew results and lead to incorrect conclusions. Cleaning helps detect and manage these anomalies.
5. **Noise Reduction**: Real-world data often contains irrelevant information. Cleaning filters out noise, enhancing signal clarity.
6. **Improved Accuracy**: Clean data leads to more accurate models and predictions, which is vital for decision-making.
7. **Data Integration**: When combining datasets, inconsistencies can arise. Cleaning ensures compatibility and coherence across sources.
8. **Efficiency**: Clean data enables faster processing and analysis, reducing computational costs and time.
9. **Easier Visualization**: Clean data is easier to visualize, leading to clearer insights and better communication of findings.
10. **Regulatory Compliance**: Many industries require adherence to data standards and regulations. Cleaning ensures compliance and data integrity.
11. **User Trust**: Clean data builds trust among stakeholders, as it reflects careful handling and attention to detail.
12. **Facilitates Automation**: Clean datasets allow for more effective automation in data pipelines and analysis workflows.
13. **Improved Collaboration**: When data is standardized and cleaned, it fosters collaboration among teams, as everyone works from the same reliable source.
14. **Prevention of Errors**: Cleaning reduces the risk of errors during analysis, ensuring more reliable outputs.
15. **Ethical Considerations**: Cleaning data helps eliminate biases that may arise from poor-quality data, promoting fair analysis.
16. **Scalability**: Clean data prepares datasets for scaling up analysis and handling larger volumes of information.
17. **Foundation for Insights**: Ultimately, data cleaning lays the groundwork for deriving meaningful insights and driving strategic decisions.

In summary, cleaning data in Python is essential for ensuring the reliability, accuracy, and usability of data in analysis and decision-making processes.

# STEP 1

INSTALLATION

pip install pandas- used to install the Pandas library in python

* **Data Structures:** Pandas provide two main data structures: Series (1-dimensional) and DataFrame (2-dimensional), which make data manipulation intuitive and efficient.
* **Data Manipulation**: It offers functions for filtering, aggregating, merging, and reshaping data easily.
* **Handling Missing Data**: Pandas has built-in functions to handle missing data effectively.
* **Data Analysis**: It supports various operations for data analysis and supports time series data.

# STEP 2

### IMPORTING PANDAS' LIBRARY

import pandas as pd

df=pd.read\_csv('C:\\Users\\Siphiwe.Mkhwanazi\\Downloads\\StudentPerformanceFactors.csv')

(df.head()

* The code imports the Pandas library, reads a CSV file containing student performance factors into a DataFrame, and prints the first five rows of that DataFrame.
* This allows you to quickly see the structure and some sample data from the CSV file, which is helpful for understanding what kind of data you are working with.

# STEP 3

### DATA ANALYSIS

print(df.info())

* **info()**: Gives a summary of the DataFrame structure, including data types and null counts.

print(df.describe())

* **describe()**: Provides descriptive statistics for numerical columns, offering insights into their distributions.

print(df.isnull().sum())

* **isnull().sum()**: Checks for missing values in the DataFrame, helping to identify potential issues with the data.

df.shape

* **Df.shape**: It helps you quickly assess the size of your dataset, which can be useful when planning data manipulation or analysis. You can use it to ensure that your DataFrame has the expected number of rows and columns after performing operations like filtering, dropping duplicates, or merging.

print("\nNumber of duplicate rows:", data.duplicated().sum())

* This command helps you identify how many duplicate rows exist in the DataFrame df.
* Duplicate rows can skew your analysis, lead to incorrect conclusions, or introduce bias, so it's essential to detect and handle them appropriately.

These commands are essential for understanding the quality and characteristics of your data, which is crucial for effective analysis and modeling.

# STEP 4

### DATA CLEANING (REMOVING / HANDLING MISSING VALUES)

df = df.drop\_duplicates()

* df.drop\_duplicates() creates a new DataFrame by removing any duplicate rows from df.
* The result is assigned back to df, so now df contains only unique rows.
* This is useful when you want to ensure that your data does not have repeated entries, which could skew analysis or results.

df.dropna()

* df.dropna() removes rows that contain any missing values (NaNs).
* The inplace=True parameter modifies the original DataFrame df directly, so it doesn't return a new DataFrame but updates df by dropping any rows with missing data.
* This is useful for cleaning the dataset before performing analysis, as missing values can lead to errors or inaccurate results.

# STEP 5

### ELIMINATE ANY IRRELEVANT COLUMNS

del df["Family\_Income"]

* This line removes the column named "Family\_Income" from the DataFrame df.
* After this operation, the DataFrame will no longer contain this column, which can be useful if you determine that the data in this column is not needed for your analysis

del df["Parental\_Education\_Level"]

* Similarly, this line removes the "Parental\_Education\_Level" column from df.
* Again, this is helpful if the column is deemed unnecessary or irrelevant for your current analysis.

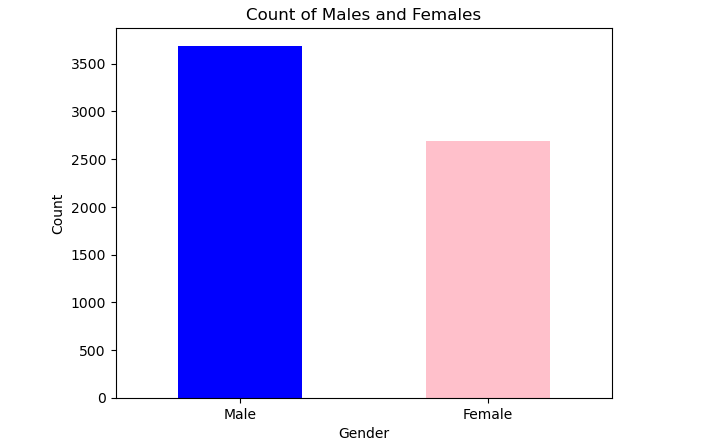
# STEP 6

## BAR GRAPH

The graph below represents the count of males and females in the student performance factors data.

Male= 3776

Female= 2753

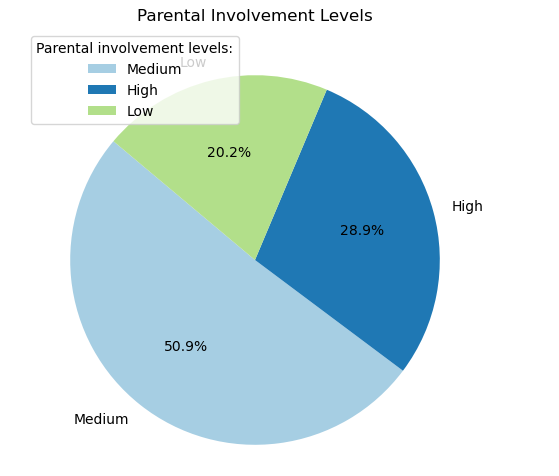


- This graph allows a quick visual comparison between the two groups, making it easy to see which group has a higher count.

- Bar graphs are effective for presenting categorical data in a way that is easy to understand for audiences, whether in reports, presentations, or data visualizations.

## PIE CHART

The graph bellow represents Parental involvement levels in the student performance factors data.



-Each segment corresponds to a level of parental involvement. Larger slices indicate a higher percentage of students whose parental involvement falls into that category.

- The percentage of students in each category is displayed within the pie slices, providing a quick reference to understand the distribution quantitatively.

-Overall, this chart gives a snapshot of how parental involvement varies across students, which could be useful for understanding its influence on other factors, such as academic performance

# CONCLUSION

Through data handling, cleaning, and basic analysis, we gained valuable insights into student performance factors. Key findings include the frequency of study times and score distributions, helping identify student study habits. These basic techniques in Python lay a foundation for deeper, more complex analyses in the future.