# Data Cleaning and Processing Report

Submitted by: Amogh Javali

ID: NN/22/2355

Role: Data Analytics

To: Novanectar

## Project Overview

This report details the data cleaning and processing steps performed for AtliQ Hardware's dataset. The goal is to ensure data quality, consistency, and readiness for analysis. The cleaned data will be used for predictive modeling and business insights.

## Technologies Used

• Power BI  
• MySQL Workbench  
• Jupyter Notebook

## Languages Used

• DAX  
• Python  
• MySQL

## Libraries Used

• Pandas  
• NumPy  
• Matplotlib  
• Seaborn  
• Scikit-learn

## Data Cleaning and Processing Steps

### 1. Data Cleaning and Preprocessing

Handle Missing Values:  
- Use Pandas to check for missing values.  
- Fill missing values using mean, median, mode, or imputation techniques.  
- Drop rows/columns if missing data is excessive.  
  
Code:  
df.isnull().sum() # Check missing values  
df.fillna(df.median(), inplace=True) # Fill missing numerical values  
df.dropna(subset=['important\_column'], inplace=True) # Drop rows with missing values in a key column  
  
Identify and Handle Outliers:  
- Use box plots (Seaborn, Matplotlib) to detect outliers.

Code:  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
plt.figure(figsize=(10,6))  
sns.boxplot(data=df.select\_dtypes(include=['number'])) # Box plot to visualize outliers  
plt.show()  
  
Q1 = df.quantile(0.25)  
Q3 = df.quantile(0.75)  
IQR = Q3 - Q1  
df = df[~((df < (Q1 - 1.5 \* IQR)) | (df > (Q3 + 1.5 \* IQR))).any(axis=1)]  
  
Fix Inconsistencies and Duplicates:  
- Standardize categorical values (e.g., ensuring 'USA' and 'United States' are consistent).  
- Remove duplicate records.  
  
Code:  
df['country'] = df['country'].replace({'India': 'IN'}) # Standardize categorical values  
df.drop\_duplicates(inplace=True) # Remove duplicate rows

### 2. Data Standardization & Transformation

Convert data types where necessary:  
- Convert date strings to DateTime format.  
- Normalize and Scale Data for better model performance.  
  
Code:  
df['date\_column'] = pd.to\_datetime(df['date\_column']) # Convert to DateTime  
  
Normalization and Scaling:  
- Min-Max Scaling (range 0-1) using MinMaxScaler.  
- Standardization (zero mean, unit variance) using StandardScaler.  
  
Code:  
from sklearn.preprocessing import MinMaxScaler, StandardScaler  
  
scaler = MinMaxScaler()  
df[['numeric\_column1', 'numeric\_column2']] = scaler.fit\_transform(df[['numeric\_column1', 'numeric\_column2']])  
  
std\_scaler = StandardScaler()  
df[['numeric\_column1', 'numeric\_column2']] = std\_scaler.fit\_transform(df[['numeric\_column1', 'numeric\_column2']])

### 3. Feature Engineering

Create new variables based on existing data:  
- Extract day, month, year from a date column.  
- Apply encoding techniques for categorical data.  
  
Code:  
df['year'] = df['date\_column'].dt.year  
df['month'] = df['date\_column'].dt.month  
  
One-Hot Encoding for categorical variables:  
  
Code:  
from sklearn.preprocessing import OneHotEncoder  
  
encoder = OneHotEncoder(sparse=False, drop='first')  
encoded\_features = pd.DataFrame(encoder.fit\_transform(df[['category\_column']]), columns=encoder.get\_feature\_names\_out())  
df = pd.concat([df.drop('category\_column', axis=1), encoded\_features], axis=1)

### 4. Exploratory Data Analysis (EDA)

Use visualization tools to explore data distributions and relationships:  
- Create a correlation matrix using Seaborn.  
- Analyze statistical patterns like mean, variance, and correlation.  
  
Code:  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
plt.figure(figsize=(10,6))  
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')  
plt.show()  
  
Descriptive statistics:  
  
Code:  
df.describe()  
df.corr()