**DataSpark: Illuminating Insights for Global Electronics**

### **1. Import Libraries**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

### **2. Load Data**

df1 = pd.read\_csv('C:/Users/Madhu/Downloads/Stores.csv')

### **3. Check for Missing Values**

(df1.isnull().sum()/len(df1))\*100

### **4. Define Null Value Imputer Function**

import pandas as pd

def null\_value\_imputer(df):

# Handling null values

for i in df.columns:

if (df[i].isnull().sum() / len(df)) \* 100 >= 80:

df.drop(i, axis=1, inplace=True)

elif df[i].dtypes in ["int64", "float64"]:

if df[i].skew() < 0.4 and df[i].skew() > -0.4:

df[i].fillna(df[i].mean(), inplace=True)

else:

df[i].fillna(df[i].median(), inplace=True)

else:

df[i].fillna(df[i].mode()[0], inplace=True)

print("Null values after imputation:\n", df.isnull().sum())

# Removing outliers using IQR method

for col in df.select\_dtypes(include=['int64', 'float64']).columns:

Q1 = df[col].quantile(0.25)

Q3 = df[col].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

# Filter the data to remove outliers

df = df[(df[col] >= lower\_bound) & (df[col] <= upper\_bound)]

print("Data after removing outliers:\n", df.describe())

return df

**Explanation:**

* **Handling Null Values:**
  + **Columns with ≥ 80% Missing Values:** These columns are dropped from the DataFrame.
  + **Numeric Columns (int64, float64):** If the skewness of the column is between -0.4 and 0.4 (close to normal distribution), missing values are filled with the mean of the column. Otherwise, missing values are filled with the median.
  + **Non-Numeric Columns:** Missing values are filled with the mode (most frequent value) of the column.
* **Removing Outliers:**
  + **Interquartile Range (IQR) Method:** For each numeric column, the IQR is calculated (Q3 - Q1). Outliers are defined as values outside the range [Q1 - 1.5 \* IQR, Q3 + 1.5 \* IQR]. These outliers are removed from the DataFrame.
* **Print Statements:**
  + After handling missing values, the number of remaining null values is printed.
  + After removing outliers, a summary of the DataFrame (using .describe()) is printed to show the statistical details of the cleaned data.

### **6. Apply Function and Save Data**

null\_value\_imputer(df1)

df1.to\_csv('C:/Users/Madhu/Downloads/storescleared.csv', index=False)

### **1. Import Library**

import pymysql

### **2. Establish Database Connection**

connection = pymysql.connect(host='127.0.0.1',

user='root',

password='Madhu@29',

database='data\_spark',

charset='utf8mb4',

cursorclass=pymysql.cursors.DictCursor)

### **3. Create a Cursor Object**

cursor = connection.cursor()

### **4. Define SQL Query to Create Table**

create\_table\_query = """

CREATE TABLE IF NOT EXISTS stores(

StoreKey INT,

Country TEXT,

State TEXT,

Square\_Meters FLOAT,

Open\_Date TEXT

)

"""

**5. Execute Table Creation Query**

try:

cursor.execute(create\_table\_query)

print("Table 'stores' created successfully.")

except pymysql.Error as e:

print("Error creating table:", e)

### **6. Define SQL Query to Insert Data**

insert\_query = """

INSERT INTO stores

(StoreKey, Country, State, Square\_Meters, Open\_Date)

VALUES (%s, %s, %s, %s, %s)

"""

### **7. Convert DataFrame to List of Lists**

stores = df1.values.tolist() # Assuming df1 is a pandas DataFrame

### **8. Insert Data into Table**

for row in stores:

try:

cursor.execute(insert\_query, (

row[0],

row[1],

row[2],

row[3],

row[4]

))

print("Inserted row successfully:", row[0])

except pymysql.Error as e:

print("Error inserting data:", e)

### **9. Commit the Transaction**

connection.commit()