```
import pandas as pd
import numpy as np
import json
import matplotlib.pyplot as plt
import seaborn as sns
import re # For Regular expression


# Load datasets
customer_file_path = "/content/drive/MyDrive/PEI DataSets/Customer.xlsx"
order_file_path = "/content/drive/MyDrive/PEI DataSets/Order.csv"
shipping_file_path = "/content/drive/MyDrive/PEI DataSets/Shipping.json"


# customer_df = pd.read_excel(customer_file_path, engine="xlrd")
# Engine= xlrd as the file is in xls format which is old one
# customer_file_path = "/content/drive/MyDrive/PEI DataSets/Customer.xls"
#customer_df = pd.read_excel(customer_file_path, engine="xlrd")


# Load Customer Data
customer_df = pd.read_excel(customer_file_path)

# Load Order Data
order_df = pd.read_csv(order_file_path)

# Load Shipping Data
shipping_df = pd.read_json(shipping_file_path)


# Function to Perform EDA + Data Cleaning
# https://emojidb.org/stats-emojis emojis or icons are taken from this website for better look and feel

def perform_eda_and_clean(df, name):
    print(f"\n📊 EDA + Data Cleaning for {name} Dataset:")

    # 📝 1. Columns and Data Types
    print("\n📝 Columns and Data Types:")
    print(df.info())

    # 🔍 2. Printing First 5 Rows
    print("\n🔍 First 5 Rows:")
    print(df.head())

    # 📌 3. Check for Missing values
    print("\n Missing Values Count:")
    print(df.isnull().sum())

    # 📈 4. Summary Statistics for Numerical Data
    print("\n📈 Summary Statistics (Numerical Data):")
    print(df.describe())

    # ✅ 5. Unique Values Per Column
    print("\n✅ Unique Values Per Column:")
    print(df.nunique())

    # 🔍 6. Check for Special Characters in String Columns
    print("\n🔍 Special Character Check:")

    # Define regex pattern for special characters (excluding space, a-z, A-Z, 0-9, and basic punctuation)
    special_char_pattern = re.compile(r'[^A-Za-z0-9\s.,]')

    for col in df.select_dtypes(include=["object"]).columns:
        # Find all special characters in the column
        special_chars = df[col].astype(str).apply(lambda x: set(re.findall(special_char_pattern, x)))

        # Get unique special characters found in the column
        unique_special_chars = set().union(*special_chars)

        if unique_special_chars:
            print(f"⚠️ Column `{col}` contains {len(unique_special_chars)} unique special characters: {unique_special_chars}")
        else:
            print(f"✅ Column `{col}` has no special characters.")

        # 🔥 7. Data Cleaning - Remove Special Characters
        df[col] = df[col].apply(lambda x: re.sub(special_char_pattern, '', str(x)))
    # 🔥 8. Handle Missing Values
    for col in df.columns:
        if df[col].isnull().sum() > 0:  # If missing values exist
            if df[col].dtype == "object":
                df[col].fillna("Unknown", inplace=True)  # Fill text columns with "Unknown"
            else:
                df[col].fillna(df[col].median(), inplace=True)  # Fill numeric columns with median

    # 🔥 9. Remove Duplicate Rows
    before = len(df)
    df.drop_duplicates(inplace=True)
    after = len(df)
    print(f"\n✅ Removed {before - after} duplicate rows.")

    # 🔥 10. Ensure Correct Data Types
    if "Age" in df.columns:
        df["Age"] = df["Age"].astype(int)  # Convert Age to integer

    if "Amount" in df.columns:
        df["Amount"] = df["Amount"].astype(float)  # Convert Amount to float

    print("\n✅ Data Cleaning Completed! Dataset is Ready for Analysis 🚀")
    return df  # Return cleaned DataFrame



# Perform EDA on each dataset
customer_df = perform_eda_and_clean(customer_df, "Customer")
```

```
📊 EDA + Data Cleaning for Customer Dataset:

📝 Columns and Data Types:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Customer_ID  250 non-null    int64 
 1   First        250 non-null    object
 2   Last         250 non-null    object
 3   Age          250 non-null    int64 
 4   Country      250 non-null    object
dtypes: int64(2), object(3)
memory usage: 9.9+ KB
None

🔍 First 5 Rows:
   Customer_ID    First     Last  Age Country
0            1   Joseph     Rice   43     USA
1            2     Gary    Moore   71     USA
2            3     John   Walker   44      UK
3            4     Eric   Carter   38      UK
4            5  William  Jackson   58     UAE

 Missing Values Count:
Customer_ID    0
First          0
Last           0
Age            0
Country        0
dtype: int64

📈 Summary Statistics (Numerical Data):
       Customer_ID         Age
count   250.000000  250.000000
mean    125.500000   47.576000
std      72.312977   18.978011
min       1.000000   18.000000
25%      63.250000   29.000000
50%     125.500000   47.000000
75%     187.750000   63.000000
max     250.000000   80.000000

✅ Unique Values Per Column:
Customer_ID    250
First          171
Last           189
Age             62
Country          3
dtype: int64
```

```
🔍 Special Character Check:
✅ Column `First` has no special characters.
✅ Column `Last` has no special characters.
✅ Column `Country` has no special characters.
```

```
# Perform EDA on Order dataset
order_df = perform_eda_and_clean(order_df, "Order")
```

```
📊 EDA + Data Cleaning for Order Dataset:

📝 Columns and Data Types:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Order_ID     250 non-null    int64  
 1   Item         250 non-null    object 
 2   Amount       250 non-null    float64
 3   Customer_ID  250 non-null    int64  
dtypes: float64(1), int64(2), object(1)
memory usage: 7.9+ KB
None

🔍 First 5 Rows:
   Order_ID      Item   Amount  Customer_ID
0         1  Keyboard    400.0          139
1         2     Mouse    300.0          250
2         3   Monitor  12000.0          239
3         4  Keyboard    400.0          153
4         5  Mousepad    250.0          153

 Missing Values Count:
Order_ID       0
Item           0
Amount         0
Customer_ID    0
dtype: int64

📈 Summary Statistics (Numerical Data):
         Order_ID       Amount  Customer_ID
count  250.000000    250.00000   250.000000
mean   125.500000   2130.00000   130.404000
std     72.312977   3575.43493    69.192711
min      1.000000    200.00000     4.000000
25%     63.250000    300.00000    71.500000
50%    125.500000    400.00000   125.500000
75%    187.750000   1500.00000   190.750000
max    250.000000  12000.00000   250.000000

✅ Unique Values Per Column:
Order_ID       250
Item             8
Amount           9
Customer_ID    160
dtype: int64

🔍 Special Character Check:
✅ Column `Item` has no special characters.

✅ Removed 0 duplicate rows.

✅ Data Cleaning Completed! Dataset is Ready for Analysis 🚀
```

```
shipping_df = perform_eda_and_clean(shipping_df, "Shipping")
```

```
📊 EDA + Data Cleaning for Shipping Dataset:

📝 Columns and Data Types:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Shipping_ID  250 non-null    int64 
 1   Status       250 non-null    object
 2   Customer_ID  250 non-null    int64 
dtypes: int64(2), object(1)
memory usage: 6.0+ KB
None
```

```
🔍 First 5 Rows:
   Shipping_ID     Status  Customer_ID
0            1    Pending          173
1            2    Pending          155
2            3  Delivered          242
3            4    Pending          223
4            5  Delivered           72

 Missing Values Count:
Shipping_ID    0
Status         0
Customer_ID    0
dtype: int64

📈 Summary Statistics (Numerical Data):
       Shipping_ID  Customer_ID
count   250.000000   250.000000
mean    125.500000   120.620000
std      72.312977    73.893848
min       1.000000     1.000000
25%      63.250000    53.250000
50%     125.500000   118.000000
75%     187.750000   187.500000
max     250.000000   248.000000

✅ Unique Values Per Column:
Shipping_ID    250
Status           2
Customer_ID    154
dtype: int64

🔍 Special Character Check:
✅ Column `Status` has no special characters.

✅ Removed 0 duplicate rows.

✅ Data Cleaning Completed! Dataset is Ready for Analysis 🚀
```

```
'''
#Check for Duplicates in each dataset
df = order_df
duplicates = df[df.duplicated(keep=False)]  # Get all duplicate rows
total_duplicates = df.duplicated().sum()  # Count duplicate rows
print(f"\n📊 Checking Duplicates in {df} Dataset:")
print(f"🔄 Total Duplicate Rows: {total_duplicates}") '''
```

```
customer_df.head()
```

| | Customer_ID | First | Last | Age | Country |
|---|---|---|---|---|---|
| **0** | 1 | Joseph | Rice | 43 | USA |
| **1** | 2 | Gary | Moore | 71 | USA |
| **2** | 3 | John | Walker | 44 | UK |
| **3** | 4 | Eric | Carter | 38 | UK |
| **4** | 5 | William | Jackson | 58 | UAE |

Next steps: Generate code with customer_df  View recommended plots  New interactive sheet

```
order_df.head()
```

| | Order_ID | Item | Amount | Customer_ID |
|---|---|---|---|---|
| **0** | 1 | Keyboard | 400.0 | 139 |
| **1** | 2 | Mouse | 300.0 | 250 |
| **2** | 3 | Monitor | 12000.0 | 239 |
| **3** | 4 | Keyboard | 400.0 | 153 |
| **4** | 5 | Mousepad | 250.0 | 153 |

Next steps: Generate code with order_df  View recommended plots  New interactive sheet

```
shipping_df.head()
```

| | Shipping_ID | Status | Customer_ID |
|---|---|---|---|
| **0** | 1 | Pending | 173 |
| **1** | 2 | Pending | 155 |
| **2** | 3 | Delivered | 242 |
| **3** | 4 | Pending | 223 |
| **4** | 5 | Delivered | 72 |

Next steps: Generate code with shipping_df View recommended plots New interactive sheet

```
# Merge Order and Shipping Data
order_shipping_df = order_df.merge(shipping_df, on="Customer_ID", how="left")

# Merge with Customer Data
final_df = order_shipping_df.merge(customer_df, on="Customer_ID", how="left")
```

```
final_df.head()
```

| | Order_ID | Item | Amount | Customer_ID | Shipping_ID | Status | First | Last | Age | Country |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1 | Keyboard | 400.0 | 139 | NaN | NaN | Ryan | Martin | 61 | UK |
| **1** | 2 | Mouse | 300.0 | 250 | NaN | NaN | Stephen | Jones | 22 | USA |
| **2** | 3 | Monitor | 12000.0 | 239 | NaN | NaN | Janet | Holmes | 36 | UK |
| **3** | 4 | Keyboard | 400.0 | 153 | NaN | NaN | Janet | Valdez | 29 | UK |
| **4** | 5 | Mousepad | 250.0 | 153 | NaN | NaN | Janet | Valdez | 29 | UK |

Next steps: Generate code with final_df View recommended plots New interactive sheet

```
"""
✅ 1. Total amount spent for "Pending" delivery status per country
"""
pending_df = final_df[final_df["Status"] == "Pending"]
total_amount_pending = pending_df.groupby("Country")["Amount"].sum().reset_index()
print("\n🔥 Total Amount Spent for Pending Deliveries by Country:\n", total_amount_pending)
```

```
 🔥 Total Amount Spent for Pending Deliveries by Country:
    Country    Amount
0     UAE   53800.0
1      UK  136300.0
2     USA   65500.0
```

```
# 🔥 2. Total Transactions, Quantity Sold, and Amount Spent per Customer (with Product Details)
customer_summary = order_df.groupby(["Customer_ID", "Item"]).agg(
    Total_Transactions=("Order_ID", "count"),
    Total_Quantity_Sold=("Item", "count"),
    Total_Amount_Spent=("Amount", "sum")
).reset_index()

print("\n📌 Customer Transactions Summary:")
print(customer_summary.head())
```

```
 📌 Customer Transactions Summary:
   Customer_ID      Item  Total_Transactions  Total_Quantity_Sold  \
0            4  Mousepad                   1                    1   
1            5   DDR RAM                   1                    1   
2            8   DDR RAM                   1                    1   
3            8  Mousepad                   2                    2   
4            8    Webcam                   1                    1   

   Total_Amount_Spent  
0               200.0  
1              1500.0  
2              1500.0  
```

```
3              450.0
4              350.0
```

```
"""
✅ 3. Maximum product purchased per country
"""
max_product_per_country = final_df.groupby(["Country", "Item"]).size().reset_index(name="Total_Purchases")
max_product_per_country = max_product_per_country.loc[max_product_per_country.groupby("Country")["Total_Purchases"].idxmax()]
print("\n🔥 Maximum Product Purchased in Each Country:\n\n", max_product_per_country)
```

```
 🔥 Maximum Product Purchased in Each Country:

    Country      Item  Total_Purchases
3      UAE  Keyboard               19
11      UK  Keyboard               29
22     USA  Mousepad               23
```

```
"""
✅ 4. Most purchased product based on age category (<30 and ≥30)
"""
# Categorizing Age Groups
final_df["Age_Category"] = final_df["Age"].apply(lambda x: "Below 30" if x < 30 else "Above 30")

most_purchased_product_age = final_df.groupby(["Age_Category", "Item"]).size().reset_index(name="Total_Purchases")
most_purchased_product_age = most_purchased_product_age.loc[most_purchased_product_age.groupby("Age_Category")["Total_Purchases"].idxmax()]
print("\n🔥 Most Purchased Product by Age Category:\n", most_purchased_product_age)
```

```
 🔥 Most Purchased Product by Age Category:
    Age_Category      Item  Total_Purchases
3      Above 30  Keyboard               49
14     Below 30  Mousepad               23
```

```
"""
✅ 5. Country with Minimum Transactions and Sales Amount
"""
country_sales = final_df.groupby("Country").agg(
    Total_Transactions=("Order_ID", "count"),
    Total_Sales_Amount=("Amount", "sum")
).reset_index()
min_transaction_country = country_sales.loc[country_sales["Total_Transactions"].idxmin()]
min_sales_country = country_sales.loc[country_sales["Total_Sales_Amount"].idxmin()]

print("\n🔥 Country with Minimum Transactions:\n", min_transaction_country)
print("\n🔥 Country with Minimum Sales Amount:\n", min_sales_country)
```

```
 🔥 Country with Minimum Transactions:
 Country                   UAE
Total_Transactions         63
Total_Sales_Amount    81650.0
Name: 0, dtype: object

 🔥 Country with Minimum Sales Amount:
 Country                   UAE
Total_Transactions         63
Total_Sales_Amount    81650.0
Name: 0, dtype: object
```