Store Sales Dataset Analysis Report

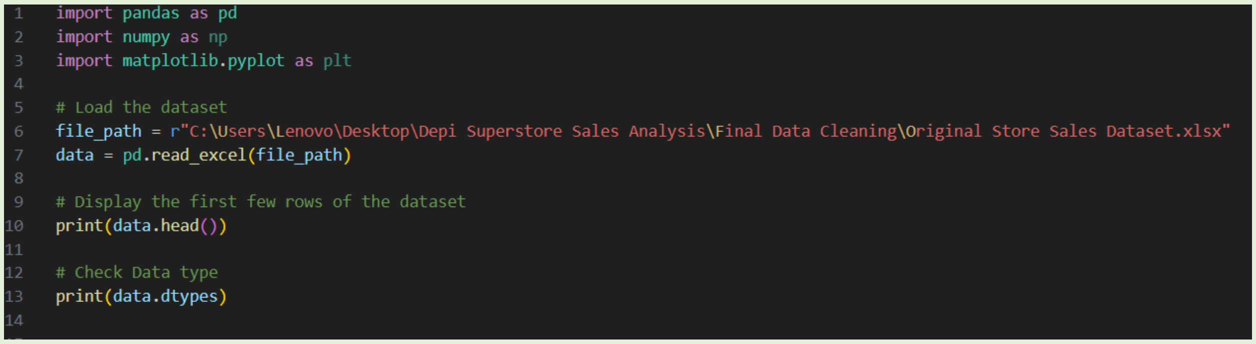
Phase: Cleaning, preprocessing Dataset

This report details the data analysis process for the " **Store Sales Dataset** " including data extraction, cleaning, preprocessing, and final output. Below is a step-by-step explanation of the code along with each phase and the corresponding results.

1. Importing Libraries and Extracting Data

Objective:

To import necessary Python libraries and load the dataset into a DataFrame for analysis.



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the dataset
file_path = r"C:\Users\Lenovo\Desktop\Depi Superstore Sales Analysis\Final Data Cleaning\Original Store Sales Dataset.xlsx"
data = pd.read_excel(file_path)

# Display the first few rows of the dataset
print(data.head())

# Check Data type
print(data.dtypes)

```

Explanation:

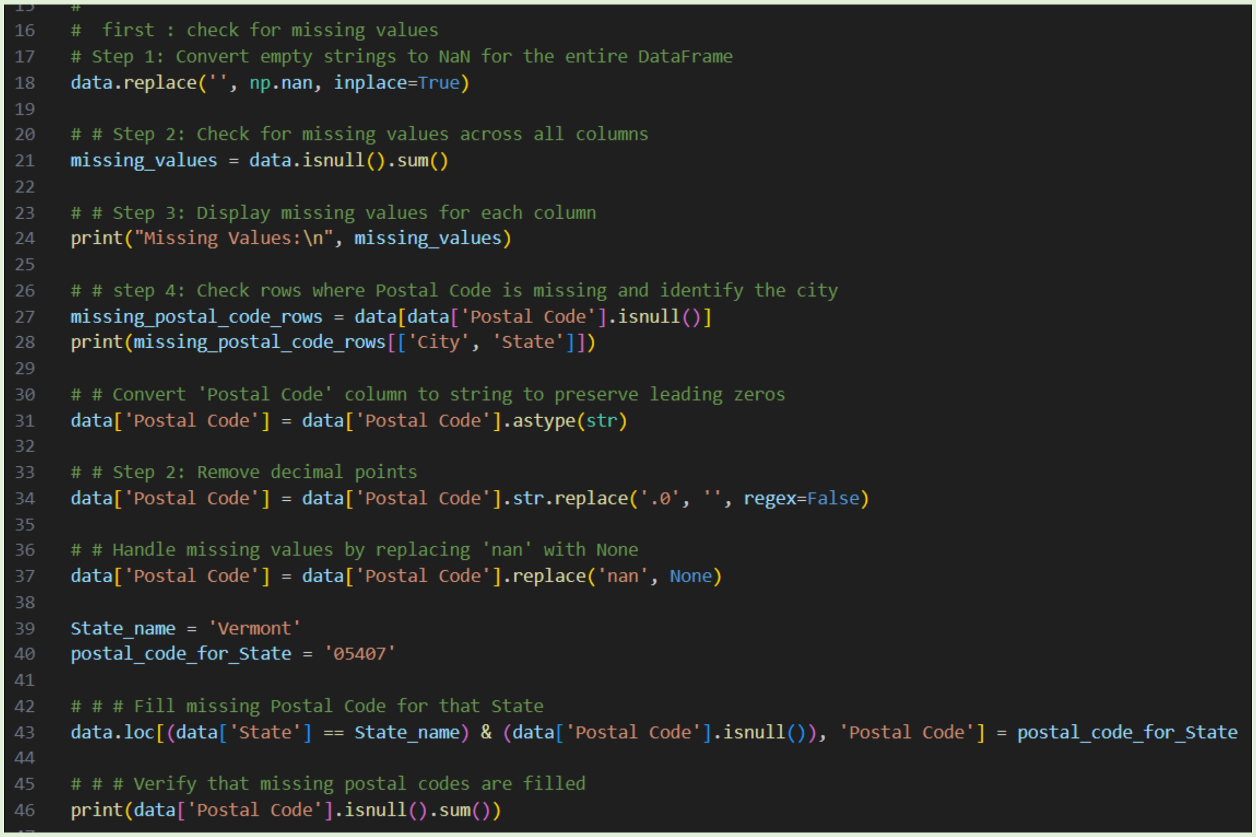
- The dataset was loaded from an Excel file into a Data Frame for analysis. This allowed for easier manipulation and inspection of the data.
- A thorough check was conducted on the data types of all columns to ensure they were appropriately formatted. This was particularly important for date and categorical fields, ensuring they were correctly recognized for further analysis.

2. Data Cleaning and Preprocessing

Objective:

To clean the data by handling missing values, correcting data types, and removing duplicates for better analysis.

Step 1: Handling Missing Values



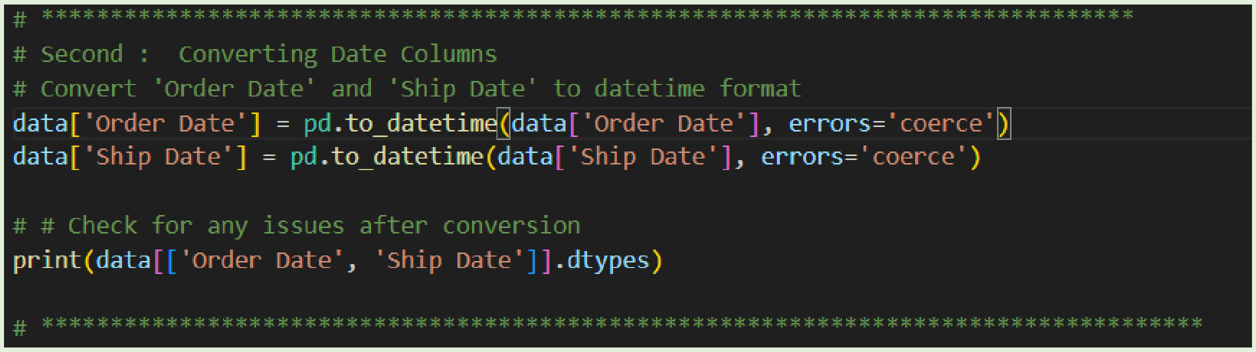
```
#
#  first : check for missing values
# Step 1: Convert empty strings to NaN for the entire DataFrame
data.replace('', np.nan, inplace=True)

# # Step 2: Check for missing values across all columns
missing_values = data.isnull().sum()

# # Step 3: Display missing values for each column
print("Missing Values:\n", missing_values)

# # step 4: Check rows where Postal Code is missing and identify the city
missing_postal_code_rows = data[data['Postal Code'].isnull()]
print(missing_postal_code_rows[['City', 'State']])

# # Convert 'Postal Code' column to string to preserve leading zeros
data['Postal Code'] = data['Postal Code'].astype(str)

# # Step 2: Remove decimal points
data['Postal Code'] = data['Postal Code'].str.replace('.0', '', regex=False)

# # Handle missing values by replacing 'nan' with None
data['Postal Code'] = data['Postal Code'].replace('nan', None)

State_name = 'Vermont'
postal_code_for_State = '05407'

# # # Fill missing Postal Code for that State
data.loc[(data['State'] == State_name) & (data['Postal Code'].isnull()), 'Postal Code'] = postal_code_for_State

# # # Verify that missing postal codes are filled
print(data['Postal Code'].isnull().sum())
```

Explanation:

- Standardizing Missing Data: Empty values were standardized by converting any blank entries into a recognized format for missing data, ensuring consistency across the dataset.

- Identifying Missing Data: A comprehensive check was performed to identify columns containing missing values, allowing us to focus on areas requiring further cleaning.
- Addressing Missing Postal Codes: Rows with missing postal codes were isolated. The corresponding city and state information was used to identify potential corrections. For example, missing postal codes for certain states, such as Vermont, were filled based on geographic knowledge.
- Postal Code Format Correction: Postal codes were converted to string format to preserve leading zeros, and any erroneous decimal points were removed. This ensured accurate representation of postal information across all records.

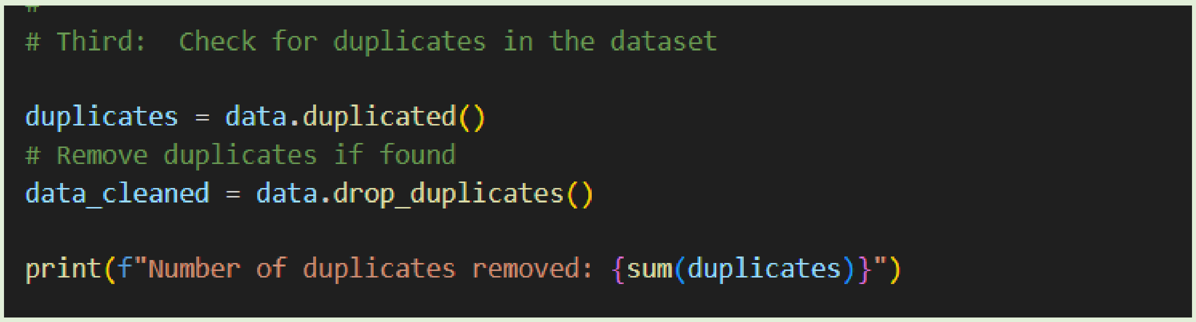
Step 2: Converting Date Columns



```
# ********************************************************************************
# Second :  Converting Date Columns
# Convert 'Order Date' and 'Ship Date' to datetime format
data['Order Date'] = pd.to_datetime(data['Order Date'], errors='coerce')
data['Ship Date'] = pd.to_datetime(data['Ship Date'], errors='coerce')

# # Check for any issues after conversion
print(data[['Order Date', 'Ship Date']].dtypes)

# *************************************************************************************
```

- The 'Order Date' and 'Ship Date' columns were converted to a consistent date format. This transformation facilitated future operations like filtering and time-based analysis. Additionally, any inconsistencies in the date formats were automatically corrected.

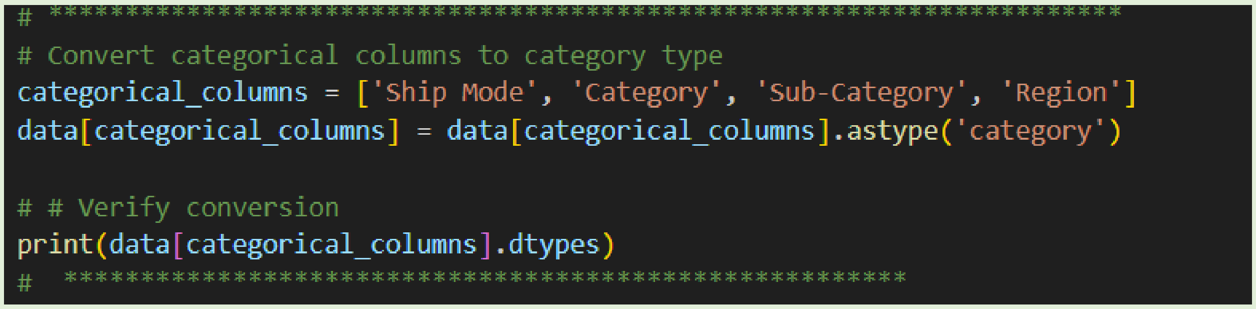
Step 3: Removing Duplicates



```python
# Third:  Check for duplicates in the dataset

duplicates = data.duplicated()
# Remove duplicates if found
data_cleaned = data.drop_duplicates()

print(f"Number of duplicates removed: {sum(duplicates)}")
```

- Duplicate records were identified and removed from the dataset. This step ensured data integrity and prevented potential distortions in subsequent analysis.

Step 4: Converting Categorical Columns



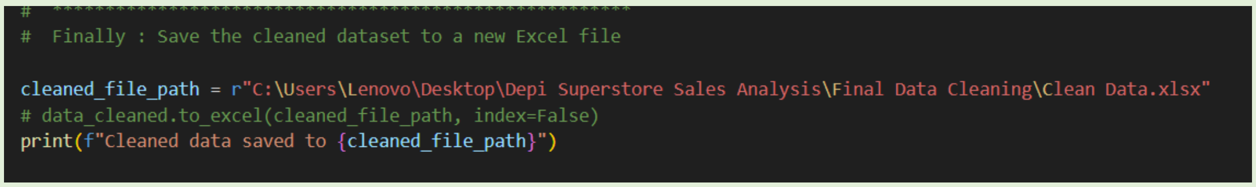
```python
# **********************************************************************
# Convert categorical columns to category type
categorical_columns = ['Ship Mode', 'Category', 'Sub-Category', 'Region']
data[categorical_columns] = data[categorical_columns].astype('category')

# # Verify conversion
print(data[categorical_columns].dtypes)
#  *******************************************************
```

- Key categorical columns such as 'Ship Mode', 'Category', 'Sub-Category', and 'Region' were converted into categorical data types. This conversion helped optimize memory usage and enhanced processing efficiency when performing analyses involving these variables.

3. Saving the Cleaned Data

Objective:

To save the cleaned dataset for future analysis or reporting.



```
#  *******************************************************
#  Finally : Save the cleaned dataset to a new Excel file

cleaned_file_path = r"C:\Users\Lenovo\Desktop\Depi Superstore Sales Analysis\Final Data Cleaning\Clean Data.xlsx"
# data_cleaned.to_excel(cleaned_file_path, index=False)
print(f"Cleaned data saved to {cleaned_file_path}")
```

Explanation:

After completing all data cleaning steps, the cleaned dataset was saved into a new Excel file. This ensured that all transformations were preserved, and the dataset was ready for future analysis and reporting.