Technical Report:

Data Cleaning Using Python

03/23/2025

Sales Store

Group
Data Whales

Names

Asmaa Hussien
Esraa Ismail
Mona Taher
Mostafa Mahmoud

Affiliation
DEPI

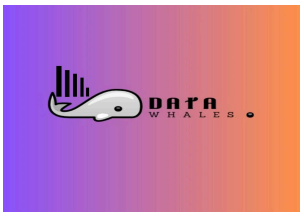

Table of Contents

1.Executive Summary

This report documents the process of data cleaning, analysis, and visualization using Python and Power BI. The dataset includes three Excel files: Orders, People, and Returns, which are used to analyze business transactions, customer information, and returned orders.

2.Introduction

Data quality plays a crucial role in deriving meaningful insights from business datasets. This report focuses on cleaning and analyzing a sales dataset using Python and SQL for data cleaning and transformation, Excel for modeling, Tableau for visualization, and Power BI for dashboard creation. The results will help answer critical business questions regarding sales trends, customer behavior, product performance, and regional profitability.

Tools Used

- Data Cleaning and Transformation: Python, SQL
- Data Modeling and Analysis: Excel
- Data Visualization: Tableau
- Dashboard Creation: Power BI, Tableau

Dataset Description

- Orders Sheet (9994 entries, 21 columns):
  Contains sales transaction data, including order details, customer information, product details, and financials (Sales, Quantity, Discount, Profit).
  Key fields: Order ID, Customer ID, Product ID, Sales, Profit, Order Date, Ship Date.
- Returns Sheet (296 entries, 2 columns):
  Tracks returned orders with columns Order ID and Returned.
- People Sheet (4 entries, 2 columns):
  Contains names of salespeople and their assigned regions.

3.Data Overview

The dataset includes three primary tables:

- **Orders**:
  Contains details of sales transactions, including order dates, customer details, product information, sales, discount levels, and profit.
- **People**:
  Contains information on regional managers responsible for sales performance in different regions.
- **Returns**:
  Tracks orders that were returned by customers.

4.Data Cleaning Process

Handling Missing Values

- Identified missing values in postal codes and replaced them with 'Unknown'.
- For missing profit or sales values, recalculated based on available data where possible.

Removing Duplicates

- Checked for duplicate rows and removed exact duplicates based on Order ID and Product ID.

Data Type Conversions

- Converted date columns to proper datetime format.
- Standardized numerical values for consistency.

Handling Outliers

- Used boxplots to detect outliers in sales and profit data.
- Removed extreme values that significantly deviated from the dataset’s distribution.

Standardization and Normalization

- Standardized region names and product categories to maintain consistency.
- Normalized discount values to fall within an expected range.

.

---

5. Results and Findings

- Sales and profit trends indicate seasonal fluctuations, with peak periods in Q4.
- The Technology category generates the highest revenue, while Office Supplies has lower profit margins.
- Corporate customer segments contribute the most to profitability.
- High discounts negatively impact profit margins but increase sales volume.
- The East and West regions have the highest sales, while some rural areas underperform.
- Fast shipping modes positively impact order value but add cost.
- A few products contribute significantly to revenue, while others show poor sales performance.

---

6. Conclusion and Recommendations

- Reduce discounts on high-selling products to maintain profitability.
- Focus marketing efforts on high-revenue product categories.
- Improve logistics to enhance shipping efficiency and customer satisfaction.
- Consider discontinuing underperforming products with low sales volume.
- Leverage predictive analytics to optimize inventory management and regional sales strategies.

7. Appendix (Python Script)

```
"""
Appendix A: Python Script for Data Cleaning and Preprocessing

This script performs data cleaning, handling missing values, removing
duplicates,
standardizing categorical variables, identifying and handling outliers,
and converting data types for analysis.
"""

# Import necessary libraries
import pandas as pd
import numpy as np

# Load datasets (Assumed to be pre-loaded as orders_df, returns_df,
people_df)

# Inspect missing values
print("Missing Values in Orders Dataset:")
print(orders_df.isnull().sum())

# Handle missing values in numerical columns
for col in ['Sales', 'Quantity', 'Discount', 'Profit']:
    if orders_df[col].isnull().any():
        orders_df[col] = orders_df[col].fillna(orders_df[col].mean())

# Handle missing values in the 'People' dataset
if people_df.isnull().values.any():
    print("There are missing values in the People dataset.")
else:
    print("No missing values in the People dataset.")

# Identify duplicate order IDs in the Returns dataset
duplicate_order_ids = returns_df[returns_df['Order
ID'].duplicated(keep=False)]
if not duplicate_order_ids.empty:
    print("Duplicate Order IDs found:")
    display(duplicate_order_ids)
```

```
# Standardize categorical variables (capitalization fix)
categorical_cols = ['Ship Mode', 'Segment', 'Country', 'City', 'State',
'Region', 'Category', 'Sub-Category']
for col in categorical_cols:
    if orders_df[col].dtype == 'object':
        orders_df[col] = orders_df[col].str.title()

# Standardize column names
people_df.columns = [col.lower().replace(' ', '_') for col in
people_df.columns]
returns_df.columns = [col.lower().replace(' ', '_') for col in
returns_df.columns]

# Identify and handle outliers using IQR method
for col in ['Sales', 'Quantity', 'Discount', 'Profit']:
    q1 = orders_df[col].quantile(0.25)
    q3 = orders_df[col].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    orders_df[col] = orders_df[col].clip(lower=lower_bound,
upper=upper_bound)

# Convert 'Discount' column to numeric if necessary
if orders_df['Discount'].dtype == 'object':
    orders_df['Discount'] =
pd.to_numeric(orders_df['Discount'].str.rstrip('%'), errors='coerce') /
100

# Convert 'order_id' column in Returns dataset to string if necessary
if not pd.api.types.is_string_dtype(returns_df['order_id']):
    returns_df['order_id'] = returns_df['order_id'].astype(str)
    print("'order_id' column in returns_df converted to string.")

# Remove duplicate rows from Orders dataset
num_duplicates = orders_df.duplicated().sum()
orders_df_cleaned = orders_df.drop_duplicates()
print(f"Number of duplicate rows removed: {num_duplicates}")

# Ensure discount values are valid
for index, row in orders_df.iterrows():
    discount = row['Discount']
    sales = row['Sales']
    if not (0 <= discount <= 1):
        orders_df.loc[index, 'Discount'] = discount / sales if sales !=
0 else 0
```

```
# Convert 'Discount' to percentage format
orders_df['Discount'] = (orders_df['Discount'] * 100).astype(int)

# Add new columns for cost estimation
orders_df['Cost'] = orders_df['Sales'] - orders_df['Profit']
orders_df['Original Price'] = orders_df['Sales'] / (1 -
(orders_df['Discount'] / 100))

# Display cleaned data samples
display(orders_df_cleaned.head())
display(people_df.head())
display(returns_df.head())
```