Python Script Documentation: Superstore Sales Data Analysis

Prepared by:

Mariam Atef Gamal Eldin

Mostafa Ayman Reda

Mostafa Khaled Farouk

Sandra Hassem Sobhy

# Introduction

This document provides a detailed breakdown of a Python script used for exploratory data analysis (EDA) on the 'Superstore Sales Dataset'. The script performs several tasks including data loading, inspection, cleaning, visualization, and export. Each section explains the purpose and function of the code used.

# 1. Importing Required Libraries

import pandas as pd  
import matplotlib.pyplot as plt

- pandas: Used for data manipulation and analysis.  
- matplotlib.pyplot: Used for plotting graphs and charts.

# 2. Load and Preview the Dataset

df = pd.read\_csv("Superstore Sales Dataset.csv")  
df.head()

- Reads the CSV file and stores it in a DataFrame named 'df'.  
- df.head(): Displays the first five rows to give an overview of the data.

# 3. Check for Missing Values

null\_counts = df.isnull().sum()  
print("Null values per column:  
", null\_counts)  
print("Columns with nulls:  
", null\_counts[null\_counts > 0])

- df.isnull().sum(): Counts null/missing values in each column.  
- Filters and prints only the columns where null values exist.

# 4. Descriptive Statistics

df.describe()  
df.describe(include='all')

- df.describe(): Gives summary statistics of numeric columns (mean, std, min, max, etc).  
- df.describe(include='all'): Includes all columns, including categorical variables.

# 5. Dataset Info

df.info()

- Displays information such as column names, data types, and non-null counts.

# 6. Visualizing Missing Values

null\_counts[null\_counts > 0].plot(kind='bar', figsize=(8, 4), color='orange')  
plt.title("Columns with Null Values")  
plt.ylabel("Number of Missing Values")  
plt.xticks(rotation=45)  
plt.grid(axis='y', linestyle='--', alpha=0.7)  
plt.tight\_layout()  
plt.show()

- Plots a bar chart showing how many missing values each column has.  
- Enhances visual understanding of data quality.

# 7. Histogram of Sales

plt.figure(figsize=(10, 5))  
plt.hist(df['Sales'], bins=50, color='skyblue', edgecolor='black')  
plt.title('Sales Distribution')  
plt.xlabel('Sales')  
plt.ylabel('Frequency')  
plt.grid(axis='y', linestyle='--', alpha=0.7)  
plt.xlim(0, 6000)  
plt.tight\_layout()  
plt.show()

- Shows the frequency distribution of Sales values.  
- Uses 50 bins and limits the X-axis to 6000 for better clarity.

# 8. Top 10 States by Total Sales

state\_sales = df.groupby('State')['Sales'].sum().sort\_values(ascending=False).head(10)  
state\_sales.plot(kind='bar', figsize=(10, 5), color='green')  
plt.title("Top 10 States by Total Sales")  
plt.ylabel("Total Sales")  
plt.xticks(rotation=45)  
plt.tight\_layout()  
plt.show()

- Groups data by State and calculates total Sales.  
- Sorts them in descending order and plots the top 10.

# 9. Export Cleaned Dataset

df.to\_csv(r'D:\المبادرة\Superstore\_Sales\_Updated.csv', index=False, encoding='utf-8-sig')  
df.to\_excel(r'D:\المبادرة\Superstore\_Sales\_Updated.xlsx', index=False, engine='openpyxl')

- Saves the cleaned DataFrame to CSV and Excel formats.  
- Uses utf-8-sig encoding for proper handling of Arabic characters.  
- Requires the openpyxl package for Excel export.

# 10. Install openpyxl (if not installed)

pip install openpyxl

- Required for saving the DataFrame as an Excel file.  
- Should be run in your terminal or notebook before using to\_excel.

# Conclusion

This script provides a strong foundation for exploratory data analysis. It highlights important data insights, visualizes missing data and sales distribution, and exports the cleaned dataset. Each step plays a crucial role in ensuring the dataset is ready for deeper analysis or modeling.

Power BI Data Transformation Order Date Parsing Documentation

# Introduction

This document outlines the step by step approach used to clean and correctly parse the Order Date column in the Superstore Sales dataset imported into Power BI Desktop. The primary objective was to ensure that Power BI could correctly interpret the Order Date column for time-based analysis and reporting.

# Step 1: Importing the Excel Dataset

The source file was an Excel spreadsheet titled 'Superstore\_Sales\_mostafa Table.xlsx'. It was imported into Power BI using the 'Get Data > Excel' option. The dataset included 18 columns such as Customer Name, Segment, Country, Product ID, Category, Product Name, Sales, and importantly, Order Date.

# Step 2: Issue Encountered with Order Date

Upon importing the data, Power BI attempted to automatically detect the data types. When trying to convert the Order Date column to type 'Date', Power BI threw an error:  
DataFormat.Error: We couldn't parse the input provided as a Date value. Details: 15/04/2018  
  
This happened because Power BI's default locale interprets dates in the MM/DD/YYYY format, whereas the dataset uses the DD/MM/YYYY format.

# Step 3: Manual Parsing Solution in Power Query

To resolve the issue, the following steps were applied:

## 3.1 Convert Order Date to Text

The Order Date column was first converted to Text. This ensured that Power BI would not attempt to parse it prematurely, allowing for full manual control over the parsing process.

## 3.2 Add Custom Column to Parse Dates

A new custom column was added using Power Query (Add Column > Custom Column) with the following M code:

try   
 let   
 cleanText = Text.Trim([Order Date]),  
 parts = Text.Split(cleanText, "/")  
 in   
 if List.Count(parts) = 3 then  
 #date(Number.FromText(parts{2}), Number.FromText(parts{1}), Number.FromText(parts{0}))  
 else null  
otherwise null

This formula performs the following:  
- Trims leading and trailing spaces.  
- Splits the date string by '/' to isolate day, month, and year.  
- Validates that there are exactly three parts.  
- Rearranges them into YYYY-MM-DD format and creates a valid date.  
- Uses try/otherwise to handle errors and replace them with null.

## 3.3 Change the New Column Type to Date

After the custom column was added, its data type was explicitly set to 'Date' using the Power Query interface. The transformation now successfully parsed valid dates while leaving invalid or empty values as nulls.

# Step 4: Final Clean-up and Applying Changes

The original Order Date column can optionally be removed or renamed. Once satisfied, the changes were applied by clicking 'Close & Apply' in Power Query to load the cleaned data into the Power BI model

.

# Step 5: Create a Date Table

To support advanced time intelligence functions like Year-to-Date or Month-over-Month calculations, a Date Table can be created using the following DAX formula:

DateTable = CALENDAR(MIN('Sales'[Order Date]), MAX('Sales'[Order Date]))

You can relate this Date Table to the 'Order Date' field in your model and use it in slicers or time-based visuals.

**Step4: Adding Dax measures**

We added DAX measures to the Power BI report to calculate important numbers:

\*4 count measures\* to track totals (like orders or customers).

\*1 average measure\* to find the average value (like price).

\*1 total measure\* to sum up a key number (like revenue).

These measures update automatically when you filter or click on charts, making it easy to explore your data by:

AOV = DIVIDE([Total sales],([Orders]))

Growth Profit = DIVIDE([Total sales]-([Sales Previous Year]),([Sales Previous Year]))

Sales color = IF([Growth Profit]>=0,"green","red")

sales color2 = IF([Total sales]>=100,"green","red")

Sales Previous Year = CALCULATE([Total sales],PREVIOUSYEAR('Calendar'[Date]))

AVg. sales = AVERAGE(Sales\_Fact[Sales])

Total sales = sum(Sales\_Fact[Sales])

**Step 6: Data modeling**

In modeling our data we made several relations as the following

* 1 fact table (sales fact)
* 5 dimensions
* 1 calendar

All these relations were made with (many to 1)

1. Sales\_fact to Dim\_region (related with customer ID)
2. Sales\_fact to calendar (related with date)
3. Sales\_fact to Dim\_country (related with Postal code)
4. Sales\_fact to Dim\_Product (related with Product ID)
5. Sales\_fact to Dim\_Region (related with Region ID)
6. Sales\_fact to calendar (related with Date)
7. Sales\_fact to Dim\_ship (related with Date)

Step 6 Creating pages

So we created 6 pages each page has its specified goal as following

**Page 1: Executive Overview (KPI Dashboard)**

**Purpose:**

Provide a **summary of key performance indicators (KPIs)** such as sales, profit, growth, and customer segments.

**Visualizations:**

* **Cards**: Display total sales and average sales.
* **Bar Charts**:
  + Total & average sales by **Quarter** and **Category** (Furniture, Office Supplies, Technology).
  + Total & average sales by **Sub-Category** (e.g., Phones, Chairs, Tables).
  + Monthly trends (Jan–Dec) for total and average sales.
* **Pie/Donut Charts**:
  + Sales by **Segment** (Consumer, Corporate, Home Office).
* **Line or Column Chart**:
  + Sales and year-over-year performance from **2015 to 2018**.

**Data Used:**

* Sales figures by date, product category, and segment.
* Time intelligence (quarter, month, year).
* Customer segmentation.

**Page 2: Sales Analysis**

**Purpose:**

Drill deeper into sales trends across **time, geography, and customer segments**. **Visualizations:**

* **Bar Charts**:
  + Monthly sales for 2015–2018.
* **Map or Filled Map** (Disabled):
  + Intended to show **sales by U.S. state**.
* **Bar Chart**:
  + Top 5 states by sales (California, New York, Texas, Virginia, Ohio).
* **Summary Metrics**:
  + Total customers, total orders, total sales, and AOV (average order value).

**Data Used:**

* Sales and customer data by state and month.
* Geographical fields: State, Region.
* Customer and order counts.

**Page 3: Customer & Regional Insights**

**Purpose:**

Compare **regional performance** and **customer distribution**.

**Visualizations:**

* **Bar Charts**:
  + Total sales by **Region** (West, East, Central, South) and **Segment**.
  + Sales breakdown by **Region** and **Category**.
* **Donut or Pie Charts**:
  + Distribution of **customers by region**.
* **KPI Cards**:
  + Total sales and number of orders.
* **Map/Table hybrid**:
  + Top cities and states with highest sales (e.g., California, New York, Los Angeles).

**Data Used:**

* Region and customer-level data.
* Sales per region, per customer.
* Segment and category breakdowns.

**Page 4: Order and Shipping Details**

**Purpose:**

Analyze **shipping modes, order volume, and delivery patterns**.

**Visualizations:**

* **Bar Charts**:
  + Number of orders by **Ship Mode** (Standard, Second Class, First Class, Same Day).
  + Sales by Ship Mode.
  + Number of orders by **Month**.
* **Date Filter**:
  + Allows filtering orders between Oct 2015 and Dec 2018.

**Data Used:**

* Order date and shipping method.
* Sales by ship mode.
* Volume and timing of orders.

**Page 5: Interactive Tooltip / Customer Details**

**Purpose:**

View detailed **customer-level sales data**.

**Visualizations:**

* **Table**:
  + Lists customer names, IDs, segments, regions, and total sales.
* **Search Bar**:
  + Search by customer name.
* **Supporting Charts**:
  + Monthly sales.
  + Customers by ship mode.
  + Orders by region.
  + Top customers by sales.

**Data Used:**

* Customer master data: name, ID, region, segment.
* Order history and shipping mode.

**Templates and Design Notes**

* **Power BI Default Theme** is used — standard visuals (cards, bar charts, tables, maps).
* Color coding is likely by **category** or **region**.
* Clean layout with **consistent timeframes (2015–2018)** and **segmentation** across pages.
* Common visual types: **Bar chart, Line chart, Card, Table, Pie chart, Map (disabled)**.

**Summary**

This Power BI report is well-structured and covers:

* High-level KPIs,
* Detailed sales trends,
* Regional and customer insights,
* Order fulfillment and shipping data,
* Customer-level drill-downs.

It's a complete **dashboard for sales and operations performance**, ideal for executive and analyst review.

# Conclusion

By manually parsing the Order Date column and applying data cleaning in Power Query, the dataset is now ready for accurate time-based analysis. This ensures robust and error-free visualizations in Power BI.