Introduction

This document provides an overview of the files and processes involved in data cleaning, analysis, and visualization for the given dataset. The workflow consists of three key steps:

- **Data Cleaning**: A Python Jupyter Notebook is used to preprocess the raw Excel file, handling missing values, removing unnecessary columns, and ensuring data consistency.
- **Data Analysis and Data Visualization**: Another Python Jupyter Notebook is dedicated to exploring and visualizing the cleaned dataset using various charts and statistical summaries.
- **Power BI Dashboard**: The final step involves creating an interactive Power BI dashboard to present insights and trends effectively.

These components work together to transform raw data into meaningful visual insights, facilitating better decision-making. The following sections describe each part in more detail and provide instructions for running the notebooks and dashboard.

1. DATA CLEANING AND DATA TRANSFORMATION

The dataset consists of three sheets:

- **Orders**: Contains details about customer orders, products, pricing, and shipping.
- **Returns**: Contains information on returned orders.
- **Users**: Contains regional managers for different regions.

The goal was to clean and structure the data to prepare it for further analysis.

The first step I took is importing these sheets into pandas DataFrames.

Orders Sheet

To understand the structure of the Orders dataset, I inspect the first few rows:

```
print("Orders Data:")
orders.head()
```

Additionally, I checked the dataset's metadata, including column names, data types, and missing values:

```
orders.info()
```

Observations:

- The dataset contains **1952 rows** and **25 columns**.
- `Product Base Margin` column has missing values, which will be addressed in later steps
- Some numerical columns, such as `Customer ID`, `Order ID`, and `Postal Code`, are better stored as strings to avoid unintended mathematical operations. I converted them using:

```
orders['Customer ID'] = orders['Customer ID'].astype(str)
orders['Order ID'] = orders['Order ID'].astype(str)
orders['Postal Code'] = orders['Postal Code'].astype(str)
```

- To improve clarity, the column **`Quantity ordered new`** is renamed to **`Quantity ordered`:**

```
orders.rename(columns={'Quantity ordered new': 'Quantity ordered'}, inplace=True)
```

Returns Sheet

The Returns sheet was examined:

```
returns.info()
```

Observations:

- The `Returns dataset` contains 1,634 rows and 2 columns
- The `Order ID` column is currently stored as an integer (int64), but since it represents a categorical identifier rather than a numerical value, it should be converted to a string
- The `Status` column is already in the correct format (object)
- No missing values are present in this dataset

Users Sheet

The Users sheet was inspected:

```
users.info()
```

Observations:

- Contains **only 4 rows** and **2 columns**.
- No missing or duplicate values were found

Handling Missing Values

The only missing values were found in the `Product Base Margin` column in the **Orders sheet**

I noticed that the `Product Base Margin` column contains 16 missing values.

I confirmed this by using:

```
orders.isna().sum()
```

To properly handle these missing values, we will need to decide on a strategy:

- Remove rows with missing values (if they are insignificant to analysis).
- Impute missing values using the mean, median, or, for example, replacing them with zeros

After identifying 16 missing values in the `Product Base Margin` column, I chose the second approach to handle them: replacing missing values with the **average margin for each Product Sub-Category** rather than removing the rows

Approach:

To inspect the rows with missing values in `Product Base Margin`, I used:

```
missing_margin_rows = orders[orders['Product Base Margin'].isna()]
missing_margin_rows
```

To analyze the missing data further, I extracted key columns:

```
missing_margin_rows[['Product Category', 'Product Sub-Category', 'Product Name', 'Sales', 'Profit']]
```

Additionally, I checked which `Product Sub-Categories` contain missing values:

```
unique_values = orders[orders['Product Base Margin'].isna()]['Product Sub-Category'].unique()
unique_values
```

And output for this code is: **'Chairs & Chairmats'**, **'Tables'**, **'Storage & Organization'**

All missing values in `Product Base Margin` belong to products from the `brand SAFCO`.
To verify this, I filtered for rows where the product name contains "SAFCO":

```
safco_rows = orders[orders['Product Name'].str.contains('SAFCO', case=False, na=False)]
len(safco_rows)
```

The output confirmed that all SAFCO products lack Product Base Margin values, reinforcing the need to replace them with category-based averages rather than randomly chosen values.

Imputing missing values based on Product Sub-Category averages

To determine appropriate replacement values, I computed the average `Product Base Margin` for each `Product Sub-Category`

```
# Average 'Product Base Margin' for each Product Sub-Category
subcategory_avg_margin = orders.groupby('Product Sub-Category')['Product Base Margin'].mean().round(2)

subcategory_avg_margin
```

I replaced missing values in `Product Base Margin` column with averages for each Sub-Category:

```
# Imputing missing 'Product Base Margin' values based on the Product Sub-Category averages
orders.loc[orders['Product Base Margin'].isna(), 'Product Base Margin'] = orders['Product Sub-Category'].map(subcategory_avg_margin)
```

To confirm that all missing values were successfully replaced, I ran the following check:

```
missing_after_imputation = orders['Product Base Margin'].isna().sum()
print(f"Number of missing values in 'Product Base Margin' after imputation: {missing_after_imputation}")
```

Now, we do not have missing values in `Product Base Margin` column

Data Validation and Cleanup

- **Order Date vs. Ship Date**

To ensure consistency in shipping and ordering data, I checked whether any orders were placed after they were shipped:

```
order_after_ship = orders[orders['Order Date'] > orders['Ship Date']]
len(order_after_ship)
```

Output confirms that all orders have valid dates, since `Order Date` always occurring before or on the same day as Ship Date.

- **Profit vs Sales**

I checked for cases where **Profit** exceeded **Sales**, which could indicate data inconsistencies:

```
profit_greater_than_sales = orders[orders['Profit'] > orders['Sales']]
len(profit_greater_than_sales)
```

The output I've got is: 70
Since Profit should not exceed Sales, I filtered out these 70 inconsistent rows:

```
orders = orders[orders['Profit'] <= orders['Sales']]
```

- **Checking for duplicates**

To ensure uniqueness in the dataset, I checked for duplicate rows:

```
orders.duplicated().sum()
```

No duplicate entries were found.

- **Column 'Country'**

Since the dataset includes a `Country` column, I verified if multiple countries were present:

```
orders['Country'].unique()
```

Since all records belong to the United States, the `Country` column does not add any analytical value.
I dropped this column, along with `Row ID`, which has no practical use:

```
orders.drop(['Row ID', 'Country'], axis=1, inplace=True)
```

Finally, I rearranged the columns to improve data structure and analysis:

```
new_order = ['Order ID', 'Order Date', 'Ship Date','Ship Mode','Order Priority', 'Customer ID',
'Customer Name',  'Customer Segment','Region', 'State or Province', 'City', 'Postal Code', 'Product
Category', 'Product Sub-Category', 'Product Container','Product Name', 'Product Base Margin',
'Sales', 'Quantity ordered', 'Discount', 'Unit Price', 'Shipping Cost','Profit']

orders=orders[new_order]
```

- ## Profit and Product Base Margin (PBM) Verification

To validate profit and PBM calculations, the expected formulas were used:

Expected Formulas:

- **Product Base Margin (PBM):**

  PBM = (Sales - COGS) / Sales

- **COGS Calculation:**

  COGS = Unit Price * Quantity

- **Profit Calculation:**

  Profit = Sales - (Unit Price * Quantity ordered) - Shipping Cost

By adding the Shipping Cost to the Profit, we actually obtain the numerator in the formula for calculating the Product Base Margin.

Profit + Shipping Cost = Sales – (Unit Price * Quantity)

Therefore, **Profit + Shipping Cost** and **Product Base Margin (PBM)** should have the same sign. We will now verify this:

```
orders['Profit + Shipping'] = orders['Profit'] + orders['Shipping Cost']

orders.describe()
```

It's interesting that all the values in the **PBM** column are positive, while in the **calculated column 'Profit + Shipping**' we have both positive and negative values, which shouldn't be the case (this can be seen from the output of the previous code, where the minimum value for **PBM** is positive and amounts to **0.35**, while the minimum value for **Profit + Shipping Cost** is negative, and amounts to **-16,471.85**).

However, it is possible that the calculation is done in a different way.

To further investigate, I recalculated **Total Costs** and **Profit** using another approach, and then, I reviewed the key financial metrics:

```
orders['COGS'] = orders['Unit Price'] * orders['Quantity ordered']

orders['Total Costs'] = orders['Unit Price'] * orders['Quantity ordered'] + orders['Shipping Cost']

orders['Calculated Profit'] = orders['Sales'] - (orders['Unit Price'] * orders['Quantity ordered'] +
orders['Shipping Cost'])

orders['Profit Difference'] = orders['Profit'] - orders['Calculated Profit']
```

```
orders['Calculated Product Base Margin'] = (orders['Sales'] - orders['Unit Price'] * orders['Quantity ordered']) / orders['Sales']

orders['PBM Difference'] = orders['Product Base Margin'] - orders['Calculated Product Base Margin']


orders[['Sales','Total Costs', 'COGS', 'Product Base Margin', 'Calculated Product Base Margin', 'PBM Difference','Profit', 'Calculated Profit', 'Profit Difference']].head()
```

This is the output:

	Sales	Total Costs	COGS	Product Base Margin	Calculated Product Base Margin	PBM Difference	Profit	Calculated Profit	Profit Difference
0	13.01	12.29	11.36	0.54	0.126826	0.413174	4.5600	0.72	3.8400
1	6362.85	6037.76	6011.76	0.60	0.055178	0.544822	4390.3665	325.09	4065.2765
2	211.15	215.85	208.56	0.45	0.012266	0.437734	-53.8096	-4.70	-49.1096
3	1164.45	1279.03	1259.04	0.43	-0.081231	0.511231	803.4705	-114.58	918.0505
4	22.23	25.27	22.96	0.56	-0.032839	0.592839	-24.0300	-3.04	-20.9900

Significant differences exist between the original and calculated values of **Profit** and **PBM**.
The recalculated Profit and PBM values are much lower than the originals.

Since the recalculated PBM is lower than expected, it suggests that **Unit Price** may not represent the true cost price (**COGS**).
Therfore, I did one more check:

```
orders['Cost Price'] = (orders['Sales'] - orders['Sales'] * orders['Product Base Margin']) / orders['Quantity ordered']
orders[['Cost Price', 'Unit Price']]
```

	Cost Price	Unit Price
0	1.496150	2.84
1	212.095000	500.98
2	5.278750	9.48
3	41.483531	78.69
4	1.397314	3.28

From this, we can see that the product price we used to verify the PBM calculation (**Unit Price**) is higher than the actual price that was used (**Cost Price**).
I assume that the `Unit Price` is not actually the base cost of the product (direct production costs) used in the PBM calculation, but rather includes some indirect costs, such as administrative costs, marketing and promotion expenses, etc.
Accounting adjustments may also influence the reported Profit and PBM values.
Since we do not have detailed information on the exact cost structure or accounting adjustments used in the dataset, I will continue the analysis using the original values for Profit and PBM.

To clean up unnecessary columns, I dropped the temporary calculations:

```
orders.drop(['COGS','Total Costs','Calculated Product Base Margin', 'PBM Difference', 'Calculated Profit', 'Profit Difference', 'Profit + Shipping', 'Cost Price'], axis=1, inplace=True)
orders.head()
```

I took similar steps to check the **'Returns'** and '**Users'** datasets. These datasets have neither missing values nor duplicates.

The next thing I did was check what values are present in the '**Status'** column of the **'Returns'** dataset. I did this using the following code:

```
returns['Status'].unique()
```

I found that there is only one value in this column, and that value is **'Returned'**.

Since this column does not provide additional insight, it can be dropped:

```
returns.drop(['Status'], axis=1, inplace=True)
returns.head()
```

Finally, I saved the cleaned datasets into new Excel file: `Cleaned_SuperStoreUS.xlsx`
This file will be used for further analysis and creating interactive dashboard.

2. Data Analysis and Data Visualization

First, I imported the previously cleaned dataset: 'Cleaned_SuperStoreUS.xlsx'

Next, I calculated the **"Profit Margin (%)"** for the Orders dataset:

```
orders['Profit Margin (%)'] = (orders['Profit'] / orders['Sales'] * 100).round(2)
orders.head()
```

I then explored the dataset with the following:

```
orders.describe(include = "all").T
```

From the output, I derived the following insights:

- There are 4 customer segments, with the majority being Corporate customers.
- There are 3 product categories, consisting of 17 sub-categories.
- The highest number of orders is for Office Supplies (1021 out of 1882, which is almost 55%), and, paper purchases dominate the most (around 15% of all orders).
- 75% of orders yield at least a 44% profit margin.
- The largest profit margin is 99.54%.

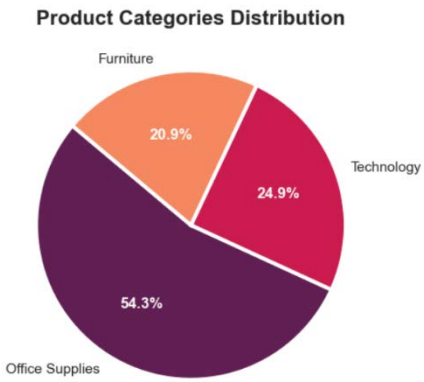
I examined the distribution of product categories using the following:

```
category_counts = orders['Product Category'].value_counts()
category_counts
```

Analysis shows that approximately 55% of products are from category Office Supplies.

Finally, I visualized the product category distribution using a pie chart.

The resulting pie chart showed the following product category distribution.



Then, I started by grouping the sales, profit, and quantity ordered by product category and calculated the sum for each:

```
order1 = (pd.DataFrame(orders.groupby(['Product Category'])[['Sales', 'Profit', 'Quantity ordered']].sum())).round(2)
```

Key Insights are:

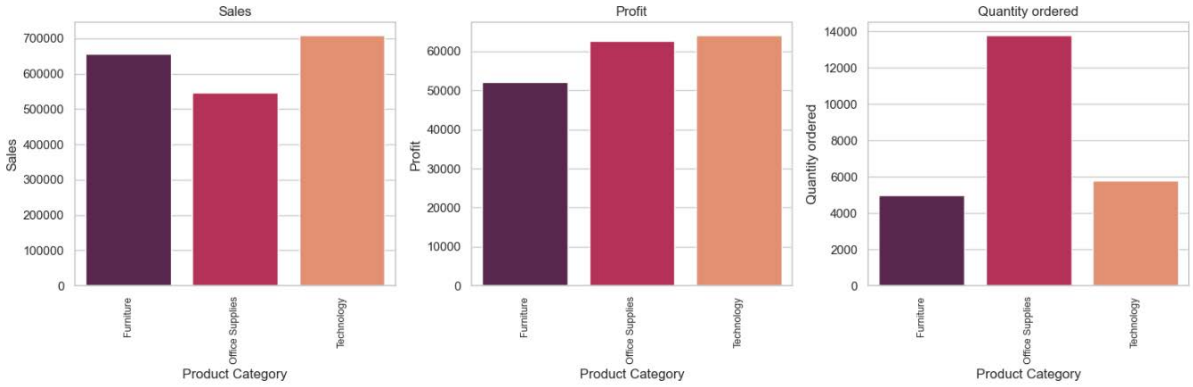
Technology generated the highest sales ($709,246.20), followed by Furniture and Office Supplies.

However, **Office Supplies** generated a higher profit ($62,695.24) compared to Furniture ($52,150.71), even though it had the lowest sales.

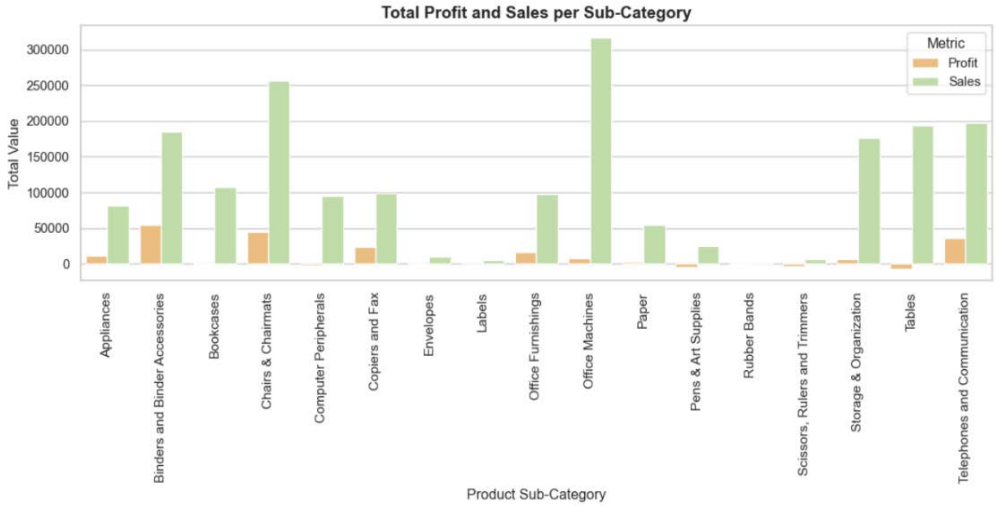
Office Supplies also had the highest quantity ordered (13,789 units), which is significantly higher than both Furniture and Technology.

In summary, **Technology** is the most profitable category. But, **Office Supplies** is the most profitable category relative to its sales, despite having the lowest sales. Furniture has the lowest profit despite having the second-highest sales.

To visualize this, I set the theme for the plots and created a figure with 1 row and 3 columns to display bar charts for sales, profit, and quantity ordered.



Next, I grouped the data by sub-category and calculated the total sales and profit for each. Then, I visualized the total profit and sales per sub-category using a bar plot:



Based on the obtained bar chart, we can conclude that **Office Machines** have the highest sales compared to other sub-categories.

Binders and Binder Accessories have the highest profit, despite having nearly half the sales of Office Machines.

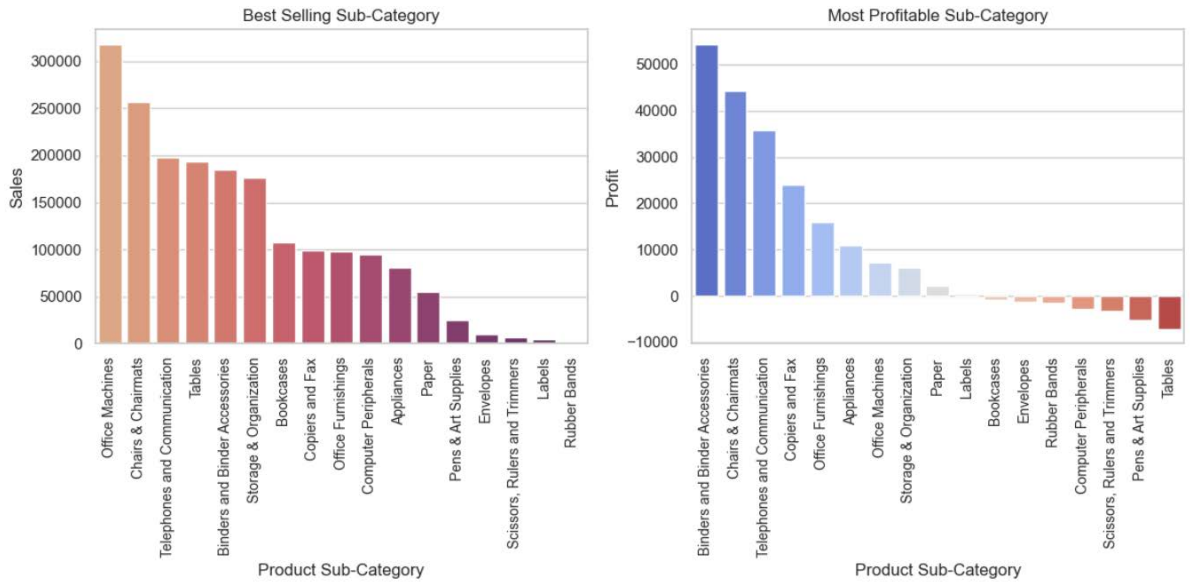
This indicates that even though Binders and Binder Accessories generate less revenue, they are more profitable, suggesting that the profit margin for this sub-category is significantly higher than that of Office Machines.

I then grouped the data by sub-category, sorted by descending order according to sales and profit:

```
orders2 = pd.DataFrame(orders.groupby(['Product Sub-Category'])[['Sales', 'Profit']].sum()).round(2)

sub_cat_profit = pd.DataFrame(orders2.sort_values('Profit', ascending = False))
```

Finally, I visualized the best-selling and most profitable sub-categories with two bar plots.



I visualized the distribution of product sub-categories across different regions using a count plot.

Based on the analysis, we can conclude that **paper products** are the most sold items across all regions of the United States, as seen from the distribution and sales data. The Central region stands out with the highest sales, dominating the other regions.

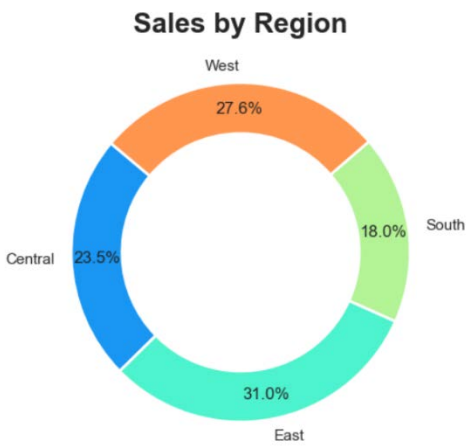
On the other hand, the **Copier and Fax** sub-category shows the lowest sales across all four regions, consistently underperforming compared to other product sub-categories.

I grouped the sales by region and created a **donut chart** to visualize the proportion of sales across regions.

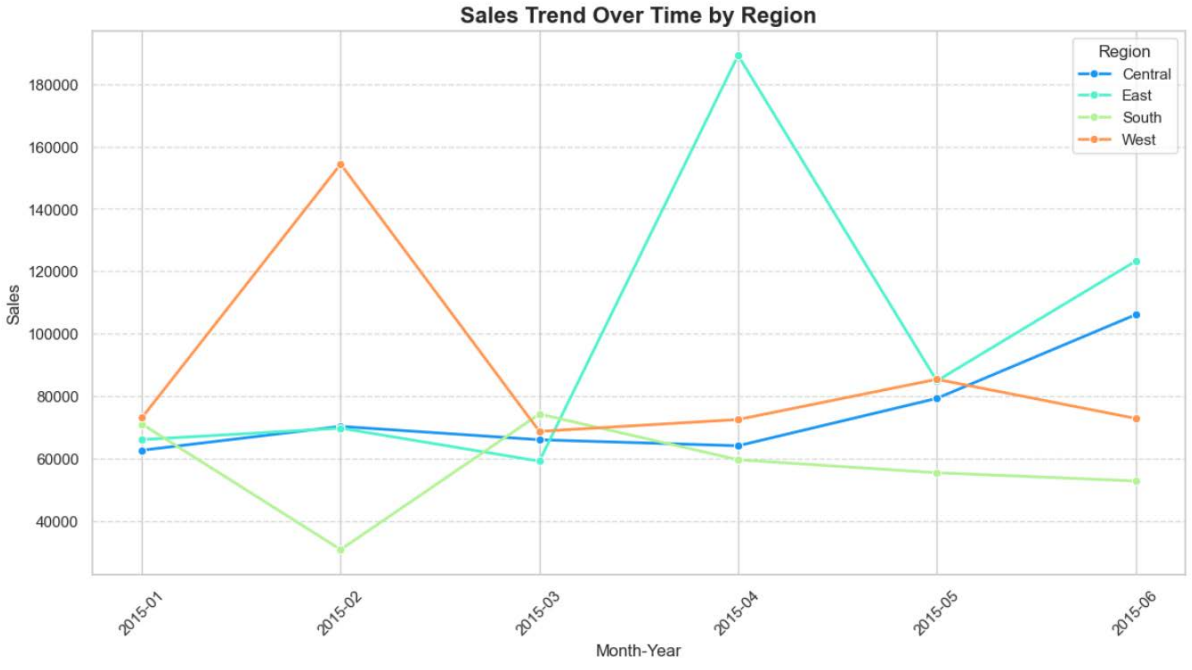
Based on the donut chart, we can observe that no single region has a dominant share in the total sales. The proportions of sales by region are as follows:

- **South**: 18%
- **West**: 28%
- **East**: 31%
- **Central**: 23%

These proportions indicate a fairly balanced distribution of sales across the regions.



I created a **line chart** to display the sales trend over time for each region, showing sales changes over the months.



In April 2015, the **East region** recorded the highest sales, which also represents the peak sales for the entire period in comparison to the other regions. On the other hand, the **South region** had the lowest sales (February 2015) in this period of 6 months, marking the least sales, when compared to the other regions as well.

I also calculated the shipping time for each product and analyzed the **average shipping time** by product sub-category.

The shortest shipping time, on average, is **1.5 days** for **Tables**, which is the fastest shipping time among all sub-categories. Following closely are the **Scissors, Rulers and Trimmers** and **Labels** sub-categories, both with an average shipping time of **1.7 days**.

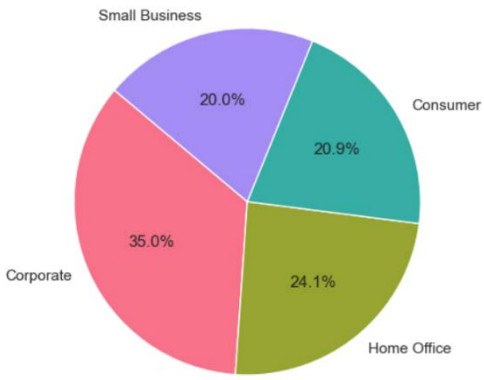
On the other hand, the **Copiers and Fax** category has the longest shipping time, with an average of **2.4 days**.

	Product Sub-Category	Shipping Time
5	Copiers and Fax	2.44
12	Rubber Bands	2.41
6	Envelopes	2.31
2	Bookcases	2.19
0	Appliances	2.18
14	Storage & Organization	2.16
9	Office Machines	2.01
3	Chairs & Chairmats	1.99
8	Office Furnishings	1.96
1	Binders and Binder Accessories	1.94
10	Paper	1.93
16	Telephones and Communication	1.92
4	Computer Peripherals	1.91
11	Pens & Art Supplies	1.87
7	Labels	1.68
13	Scissors, Rulers and Trimmers	1.67
15	Tables	1.51

A **pie chart** was used to display the distribution of customers by segment.

While **Corporate segment** makes up the largest portion at **35%**, it's still a bit too close in proportion to the other segments to say any one of them has a significant dominance. The shares are relatively balanced across Home Office (24.1%), Consumer (20.9%), and Small Business (20%), meaning no single segment overwhelmingly leads the others.

So, while **Corporate** has the highest share, it's not a massive difference, and we can't definitively say that any segment holds a truly dominant position in the overall distribution. All segments are somewhat comparable in size.



<u>Return Rate Calculation:</u>

To calculate the **Return Rate**, it was necessary to merge the **Orders** and **Returns** datasets. This was done using a left join on the **Order ID** column, ensuring that all orders from the Orders dataset remained intact while only matching records from the Returns dataset were included.

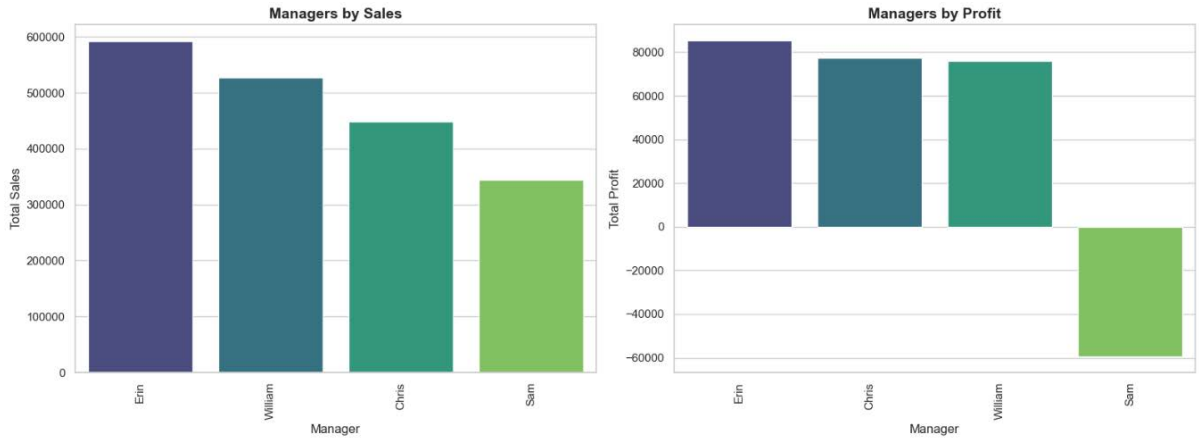
```
merged_data = pd.merge(orders, returns, on='Order ID', how='left')
```

A new column `**Returned**` was created to indicate whether an order was returned. The return rate was calculated as the **percentage of returned orders** for each **Product Category**.

```
return_rate['Return Rate (%)'] = return_rate['Returned'] * 100
```

To represent Managers by profit and sales, I opted for bar charts.

Since Managers are in the '**Users'** table and Sales and Profit are in the '**Orders'**, we need to merge these two datasets before creating the visualizations.



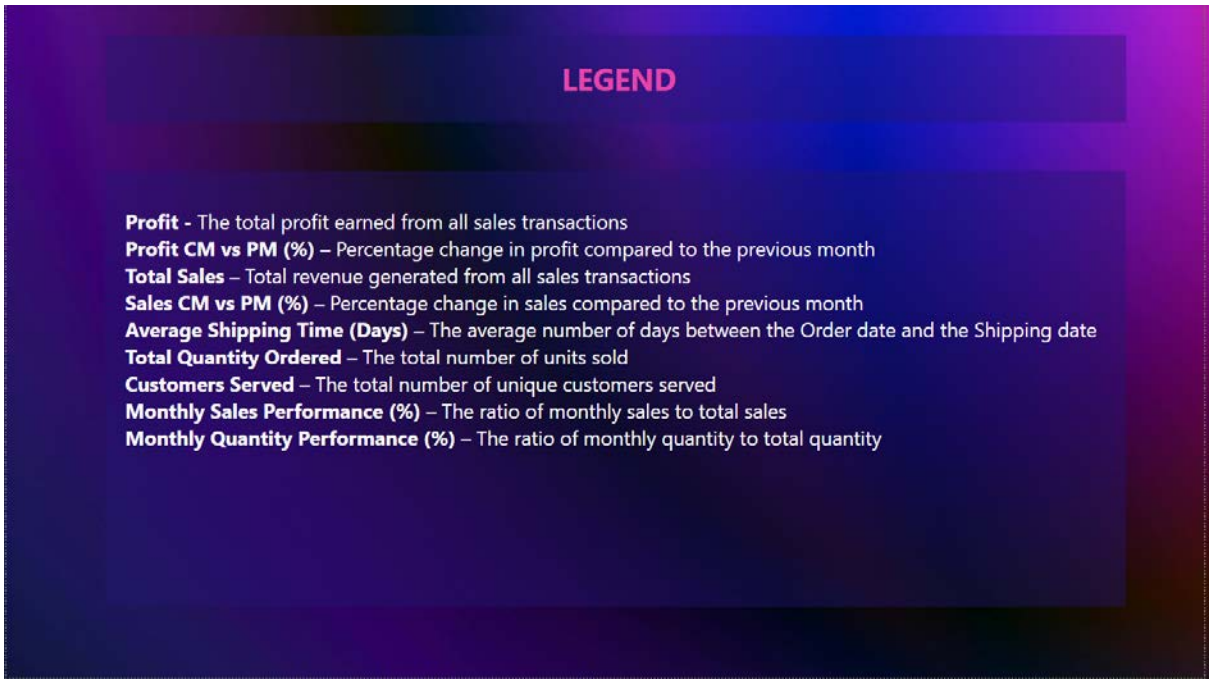
Erin is the best manager when looking at both profit and sales. When analyzing profit, Chris follows Erin, and then comes William, although the difference in profits between Chris and William is very small. The only manager incurring losses is Sam, with a loss of around $60,000, even though his sales aren't significantly different from the other three regional managers.

3. POWER BI DASHBOARD

In my Power BI dashboard, I have four sheets:

- Legend
- Overview
- Product Analysis
- Manager Performance

Sheet '**Legend'** contains explanations for each KPI used in the analysis:



Data Preparation Process:

- I first imported the cleaned Excel file into Power BI

- Then, I performed a final check to ensure all tables were structured correctly

- I added a new calculated column, **'Shipping Days'**, by computing the difference between '**Ship Date'** and **'Order Date'**. This column was formatted as a whole number

- In Model View, I established relationships between tables

- I created a **'Date Table'** and linked it to the 'Orders' table for better time-based analysis

- I added a **'Key Measures'** table where I stored all calculated measures essential for building the dashboard

Sheet ‘Overview’

In the "**Overview**" section of the Power BI dashboard, I have a summarized view of the most important KPIs, both by month and for the entire period.

In the top-right corner, there's a slicer that allows you to select the desired month. If you want to view the KPIs for the entire period, you can uncheck the months and analyze the data for the full 6-month period.

Additionally, by clicking on a specific state on the provided map, the KPIs for that state will be displayed, providing users with a more detailed insight into regional performance.

Sheet ‘Product Analysis’

In the "**Product Analysis**" section of the Power BI dashboard, I presented a detailed analysis of products by categories and sub-categories.

In addition to the slicer for selecting months, we have two additional slicers to refine the data:

1. **Product Category Slicer:** Here, you can choose one or multiple product categories of interest, or select all categories for a comprehensive view
2. **Region Slicer:** This slicer allows you to select one or more regions, or view data from all regions

When you select a specific region using the region slicer, that region is highlighted on the map, giving a geographical context to the selected data.

Sheet ‘Manager Performance’

In the last sheet, we have an overview of the performance of regional managers, where KPIs for each manager can be viewed.

Here, I also displayed the best manager by profit, both for each month (if a specific month is selected) and for the entire 6-month period, if months are unchecked.

In addition to the slicer for selecting months, there are two more slicers: one for selecting the manager whose performance we want to track, and another for selecting the product category.

Also, when selecting a specific manager, the region they oversee is highlighted on the map.