Supply Chain Analysis Report

1. Introduction

This report provides an analysis of the supply chain data using Python for data cleaning and transformation, followed by visualization in Power BI. The dataset comprises three key sources: Orders and Shipments, Fulfillment, and Inventory. The objective is to derive actionable insights regarding sales, profit trends, shipment delays, and customer distribution.

2. Data Cleaning and Transformation

2.1 Data Preprocessing

- Three datasets were loaded using Pandas and checked for missing values.
- Duplicate records were removed to ensure data integrity.
- Column names were standardized by stripping whitespace.
- Non-alphanumeric characters in the 'Customer Country' column were cleaned using regex.

2.2 Data Transformations

- The 'Discount %' column was corrected by replacing '-' with 0 and converting it to float.
- Merging of datasets based on relevant keys to create a consolidated dataset.
- Derived columns were added to support further analysis, including profit calculations.

3. Key Insights from Power BI Dashboard

3.1 Sales and Profit Trends

- **Total Gross Sales:** $6M
- **Total Profit:** $4M
- **Average Profit Margin:** 84.43%
- **Storage Cost:** $86.43K
- The gross sales trend shows a steady increase from 2015 to 2017, followed by a slight decline in late 2017.

3.2 Product Performance

- The highest profit-generating products include **Perfect Fitness Rip Deck**, **Field & Stream Sports 16 Gun**, and **Nike Men's Free 5.0 Running Shoes**.

- The most profitable product category is **Cleats**, followed by **Fishing Gear** and **Cardio Equipment**.

3.3 Shipment and Delays Analysis

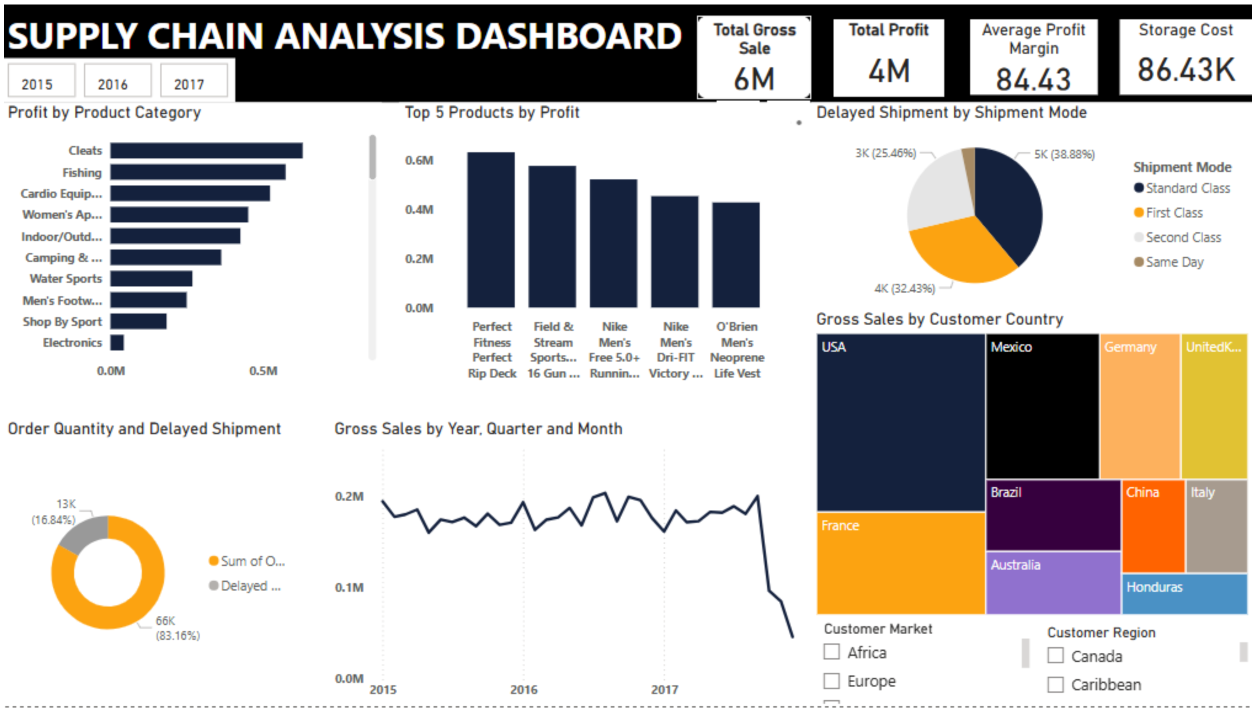
- **Delayed Shipments:** 16.84% of total shipments were delayed.
- **Shipment Mode Impact:**
  - Standard Class had the highest percentage of delays (38.88%).
  - Second Class shipments accounted for 32.43% of delays.
  - First Class had the lowest delays (25.46%).

3.4 Customer Demographics and Sales Distribution

- **Top Customer Markets:** USA, Mexico, Germany, and France.
- **Customer Region Breakdown:**
  - North America and Europe dominate the customer base.
  - Minimal sales contributions from Africa and the Caribbean.

3.5 Power BI Dashboard

Below is a screenshot of the Power BI dashboard showcasing these insights:



4. Recommendations

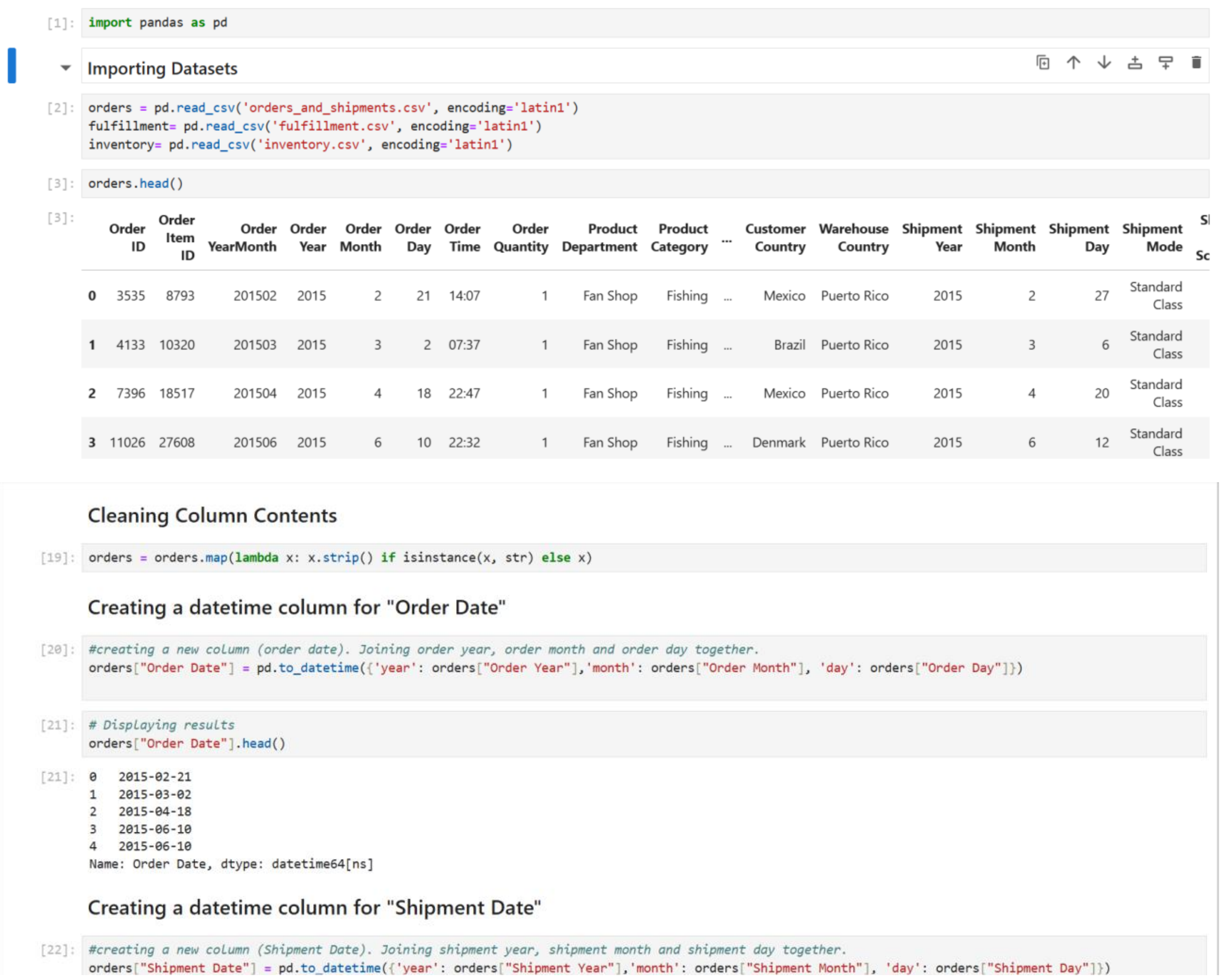
1. **Optimize High-Delay Shipment Modes**:
   - Improve logistics for Standard Class shipments to reduce the high delay rate.
   - Explore alternative fulfillment strategies to improve Second Class efficiency.
2. **Enhance Inventory Management**:
   - Given the high storage cost of $86.43K, optimize inventory levels to reduce excess stock.
3. **Expand Market Reach**:
   - Increase marketing efforts in underperforming regions like Africa and the Caribbean.
   - Leverage insights from top-performing countries to tailor new market strategies.
4. **Product Focus Strategy**:
   - Increase stock and promotions for top-performing products like Perfect Fitness Rip Deck.
   - Evaluate underperforming categories for potential discontinuation or improvement.

5. Conclusion

The supply chain analysis reveals strong profitability, but there are areas for improvement, especially in shipment delays and storage cost management. By optimizing logistics, enhancing inventory strategies, and expanding market reach, the company can further improve operational efficiency and profitability.

6. Technical Implementation

- Data analysis and cleaning were performed using Python (Pandas, Regex).
- Data visualization was conducted in Power BI, with DAX calculations used for profit margin and other key metrics.
- Below is an excerpt of the Python script used for data processing:



```
[1]: import pandas as pd
```

Importing Datasets

```
[2]: orders = pd.read_csv('orders_and_shipments.csv', encoding='latin1')
     fulfillment= pd.read_csv('fulfillment.csv', encoding='latin1')
     inventory= pd.read_csv('inventory.csv', encoding='latin1')
```

```
[3]: orders.head()
```

[3]:

	Order ID	Order Item ID	Order YearMonth	Order Year	Order Month	Order Day	Order Time	Order Quantity	Product Department	Product Category	...	Customer Country	Warehouse Country	Shipment Year	Shipment Month	Shipment Day	Shipment Mode
0	3535	8793	201502	2015	2	21	14:07	1	Fan Shop	Fishing	...	Mexico	Puerto Rico	2015	2	27	Standard Class
1	4133	10320	201503	2015	3	2	07:37	1	Fan Shop	Fishing	...	Brazil	Puerto Rico	2015	3	6	Standard Class
2	7396	18517	201504	2015	4	18	22:47	1	Fan Shop	Fishing	...	Mexico	Puerto Rico	2015	4	20	Standard Class
3	11026	27608	201506	2015	6	10	22:32	1	Fan Shop	Fishing	...	Denmark	Puerto Rico	2015	6	12	Standard Class

Cleaning Column Contents

```
[19]: orders = orders.map(lambda x: x.strip() if isinstance(x, str) else x)
```

Creating a datetime column for "Order Date"

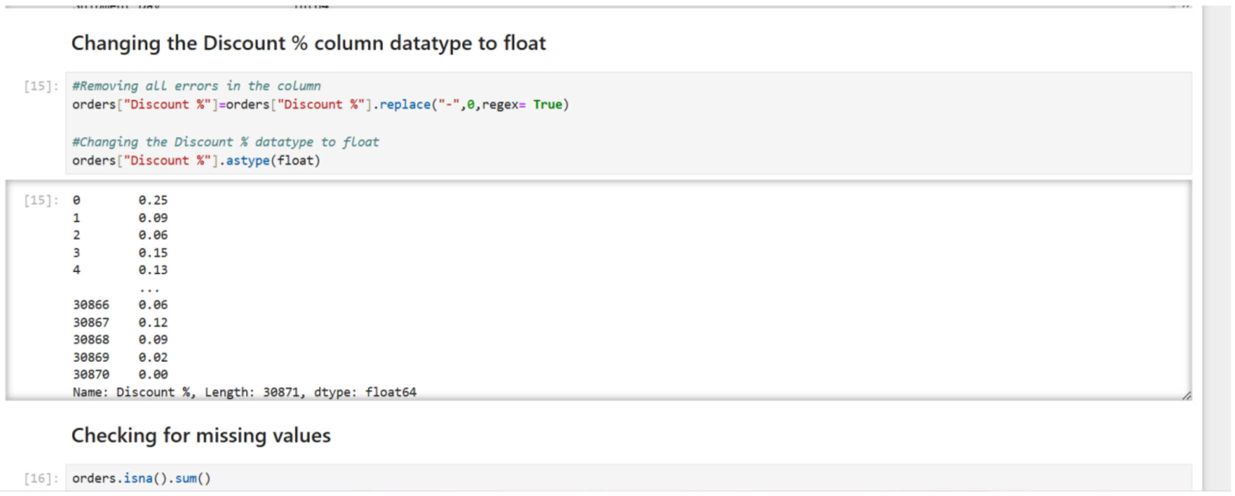
```
[20]: #creating a new column (order date). Joining order year, order month and order day together.
      orders["Order Date"] = pd.to_datetime({'year': orders["Order Year"],'month': orders["Order Month"], 'day': orders["Order Day"]})
```

```
[21]: # Displaying results
      orders["Order Date"].head()
```

```
[21]: 0    2015-02-21
      1    2015-03-02
      2    2015-04-18
      3    2015-06-10
      4    2015-06-10
      Name: Order Date, dtype: datetime64[ns]
```

Creating a datetime column for "Shipment Date"

```
[22]: #creating a new column (Shipment Date). Joining shipment year, shipment month and shipment day together.
      orders["Shipment Date"] = pd.to_datetime({'year': orders["Shipment Year"],'month': orders["Shipment Month"], 'day': orders["Shipment Day"]})
```



Changing the Discount % column datatype to float

```
[15]: #Removing all errors in the column
      orders["Discount %"]=orders["Discount %"].replace("-",0,regex= True)

      #Changing the Discount % datatype to float
      orders["Discount %"].astype(float)
```

```
[15]: 0        0.25
      1        0.09
      2        0.06
      3        0.15
      4        0.13
               ...
      30866    0.06
      30867    0.12
      30868    0.09
      30869    0.02
      30870    0.00
      Name: Discount %, Length: 30871, dtype: float64
```

Checking for missing values

```
[16]: orders.isna().sum()
```

For the complete script, refer to the attached Python notebook.

Supply Chain.ipynb

Download the Complete Python Script

For full access to the script, download it from the following link:
Download Python Script Here

This report serves as a foundation for further strategic decision-making in optimizing supply chain operations.