PROJECT TITLE:

Exploratory Data Analysis (EDA) on Sales, Transfers, and Warehouse Dataset

Name: Sadneya Sadanand Samant
Submitted To : Agratas EduTech
Date Of Submission: 22/01/25

ABSTRACT

This study presents an in-depth exploratory data analysis (EDA) of a comprehensive dataset encompassing transaction details related to retail sales, warehouse operations, and transfers. Leveraging powerful Python libraries such as Pandas, Matplotlib, and Seaborn, the analysis identifies key trends, anomalies, and relationships among critical variables. The dataset, consisting of 307,645 rows and 9 columns, focuses on sales and operational metrics associated with multiple suppliers. The findings reveal significant patterns and actionable insights that can enhance decision-making and operational efficiency. Recommendations are provided to optimize processes and address challenges highlighted by the observed trends.

Table of Contents

INTRODUCTION

Exploratory Data Analysis (EDA) is a fundamental step in the data analysis process that involves examining and summarizing datasets to uncover meaningful insights and patterns. It serves as a preliminary investigation phase where analysts use statistical methods and visualization techniques to understand the structure, relationships, and characteristics of the data. By identifying trends, outliers, and potential errors, EDA enables better decision-making and sets the foundation for building robust models or drawing significant conclusions.

This project focuses on performing EDA on a selected dataset, emphasizing a systematic approach to understanding the data's properties and deriving actionable insights. Through a combination of data cleaning, visualization, and statistical analysis, the project aims to uncover hidden patterns, provide valuable insights, and propose recommendations based on the findings.

The subsequent sections outline the dataset description, methodology, preprocessing steps, analysis, visualizations, and key insights derived from the EDA process.

1.1 Objective

The purpose of this analysis is to perform an in-depth Exploratory Data Analysis (EDA) on a dataset related to retail and warehouse transactions. EDA is a crucial first step in understanding the structure of the data and uncovering hidden patterns, trends, and relationships that can guide further analysis or decision-making. This analysis aims to identify missing or inconsistent data, detect outliers that may distort results, and visualize the distribution of sales and operational data. Additionally, it seeks to investigate correlations between different transaction types, such as retail sales, warehouse sales, and transfers, while also assessing supplier performance. Through these steps, the analysis provides valuable insights to support better decision-making and strategy formulation.

1.2 Dataset Overview

The chosen dataset contains 307,645 rows and 9 columns. The columns are:

1. **YEAR**: The year the transaction took place (e.g., 2022, 2023).
2. **MONTH**: The month of the transaction (1–12).
3. **SUPPLIER**: The supplier responsible for the item.
4. **ITEM CODE**: Unique code identifying the item.
5. **ITEM DESCRIPTION**: A brief description of the item.
6. **ITEM TYPE**: Category of the item (e.g., electronics, apparel).

7. **RETAIL SALES**: The sales made through retail channels.
8. **RETAIL TRANSFERS**: The amount transferred between retail outlets.
9. **WAREHOUSE SALES**: Sales made through warehouse operations.

METHODOLOGY

1. Import and load the dataset using Python libraries (Pandas, NumPy, etc.).
2. Conduct an initial overview to understand the structure and nature of the data.
3. Perform data cleaning by handling missing values and duplicates.
4. Use statistical and visual tools (Matplotlib, Seaborn) for analysis.
5. Making Conclusions based on findings.

DATA PREPROCESSING

3.1 Importing Libraries and Loading Dataset:

1. **Importing Libraries**
   import pandas as pd
   import matplotlib.pyplot as plt
   import seaborn as sns
2. **Loading Dataset**
   data = pd.read_csv("C://Users//Sadneya//Desktop//Warehouse_and_Retail_Sales.csv")

3.2 Display Basic Information

This step ensures that the data is loaded correctly and provides insights into the structure of the dataset.

1. **To get size of data** (ie no of rows and columns): data.shape
   (307645, 9)

2. **To print 5 rows of data:**
data.head()

	YEAR	MONTH	SUPPLIER	ITEM CODE	ITEM DESCRIPTION	ITEM TYPE	RETAIL SALES	RETAIL TRANSFERS	WAREHOUSE SALES
0	2020	1	REPUBLIC NATIONAL DISTRIBUTING CO	100009	BOOTLEG RED - 750ML	WINE	0.00	0.0	2.0
1	2020	1	PWSWN INC	100024	MOMENT DE PLAISIR - 750ML	WINE	0.00	1.0	4.0
2	2020	1	RELIABLE CHURCHILL LLLP	1001	S SMITH ORGANIC PEAR CIDER - 18.7OZ	BEER	0.00	0.0	1.0
3	2020	1	LANTERNA DISTRIBUTORS INC	100145	SCHLINK HAUS KABINETT - 750ML	WINE	0.00	0.0	1.0
4	2020	1	DIONYSOS IMPORTS INC	100293	SANTORINI GAVALA WHITE - 750ML	WINE	0.82	0.0	0.0

3. **Data information:**
print(data.info())

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307645 entries, 0 to 307644
Data columns (total 9 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   YEAR              307645 non-null  int64
 1   MONTH             307645 non-null  int64
 2   SUPPLIER          307478 non-null  object
 3   ITEM CODE         307645 non-null  object
 4   ITEM DESCRIPTION  307645 non-null  object
 5   ITEM TYPE         307644 non-null  object
 6   RETAIL SALES      307642 non-null  float64
 7   RETAIL TRANSFERS  307645 non-null  float64
 8   WAREHOUSE SALES   307645 non-null  float64
dtypes: float64(3), int64(2), object(4)
memory usage: 21.1+ MB
None
```

4. **Summary statistics for numerical columns:**
print(data.describe())

```
                YEAR          MONTH   RETAIL SALES  RETAIL TRANSFERS  \
count  307645.000000  307645.000000  307642.000000     307645.000000
mean     2018.438525       6.423862       7.024071          6.936465
std         1.083061       3.461812      30.986238         30.237195
min      2017.000000       1.000000      -6.490000        -38.490000
25%      2017.000000       3.000000       0.000000          0.000000
50%      2019.000000       7.000000       0.320000          0.000000
75%      2019.000000       9.000000       3.267500          3.000000
max      2020.000000      12.000000    2739.000000       1990.830000

       WAREHOUSE SALES
count    307645.000000
mean         25.294597
std         249.916798
min       -7800.000000
25%           0.000000
50%           1.000000
75%           5.000000
max       18317.000000
```

5. **Converting Columns to List Form**:
   data.columns.tolist()

```
['YEAR',
 'MONTH',
 'SUPPLIER',
 'ITEM CODE',
 'ITEM DESCRIPTION',
 'ITEM TYPE',
 'RETAIL SALES',
 'RETAIL TRANSFERS',
 'WAREHOUSE SALES']
```

3.3 Handling Missing Values

The dataset initially contained missing values in several columns. Missing values can affect results and prevent accurate analysis, so it is critical to handle them appropriately.

1. **Check for missing values**
   print(data.isnull().sum())

```
YEAR                  0
MONTH                 0
SUPPLIER            167
ITEM CODE             0
ITEM DESCRIPTION      0
ITEM TYPE             1
RETAIL SALES          3
RETAIL TRANSFERS      0
WAREHOUSE SALES       0
dtype: int64
```

2. **Fill Missing Values**
   Here we see that “SUPPLIER” column contains 167 missing values. Thus we replaced these with the placeholder "Unknown" to maintain consistency without discarding rows.Then “ITEM TYPE” contains 1 missing value. This was replaced with the most frequent category (mode). Also “RETAIL SALES” contains 3 missing values. We imputed the mean retail sales value for these records.
   Code:

```
# Fill SUPPLIER with 'Unknown'
data['SUPPLIER'].fillna('Unknown', inplace=True)

# Fill ITEM TYPE with mode
data['ITEM TYPE'].fillna(data['ITEM TYPE'].mode()[0], inplace=True)

# Fill RETAIL SALES with mean
data['RETAIL SALES'].fillna(data['RETAIL SALES'].mean(), inplace=True)
```

# Verify if missing values are handled
print(data.isnull().sum())
Result:

```
YEAR                0
MONTH               0
SUPPLIER            0
ITEM CODE           0
ITEM DESCRIPTION    0
ITEM TYPE           0
RETAIL SALES        0
RETAIL TRANSFERS    0
WAREHOUSE SALES     0
dtype: int64
```

Thus ,the imputation method was chosen based on the nature of the columns—categorical columns were imputed using the mode, while numerical columns were imputed with the mean to minimize bias.

3.4 Removing Duplicates

1. **Checking duplicate values**
   data.nunique()

```
YEAR                    4
MONTH                  12
SUPPLIER              397
ITEM CODE           34056
ITEM DESCRIPTION    34822
ITEM TYPE               8
RETAIL SALES        10675
RETAIL TRANSFERS     2504
WAREHOUSE SALES      4895
dtype: int64
```

2. **Remove duplicates:**
   Duplicate rows can distort analysis, particularly in large datasets. After inspecting the dataset, we used the following code to remove duplicates:
   data = data.drop_duplicates()
   This step ensured that each record was unique and contributed meaningful information to the analysis.

3. **Cleaning Column Names**
   Column names were cleaned to remove extra spaces and ensure consistency across the dataset:
   data.columns = [col.strip() for col in data.columns]
   This step ensures that no column names contain leading or trailing spaces that could lead to errors during analysis.

Analysis and Findings

4.1 Univariate Analysis

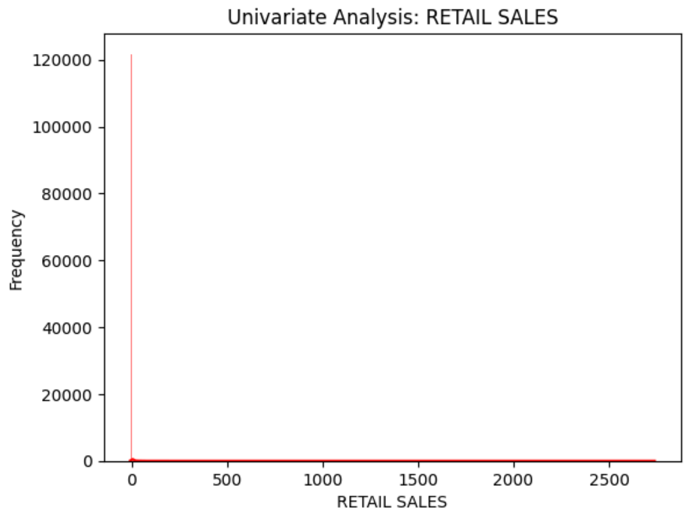
4.1.1 Distribution of Retail Sales, Retail Transfers and Warehouse Sales

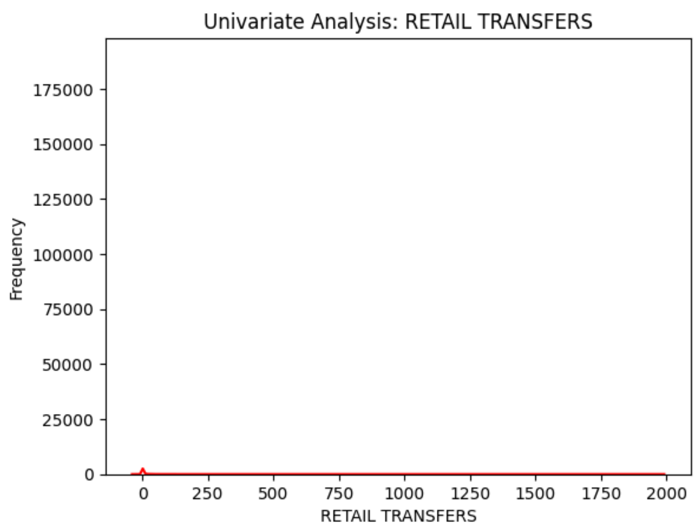
A histogram was used to visualize the distribution of retail sales, retail transfers and warehouse sales:

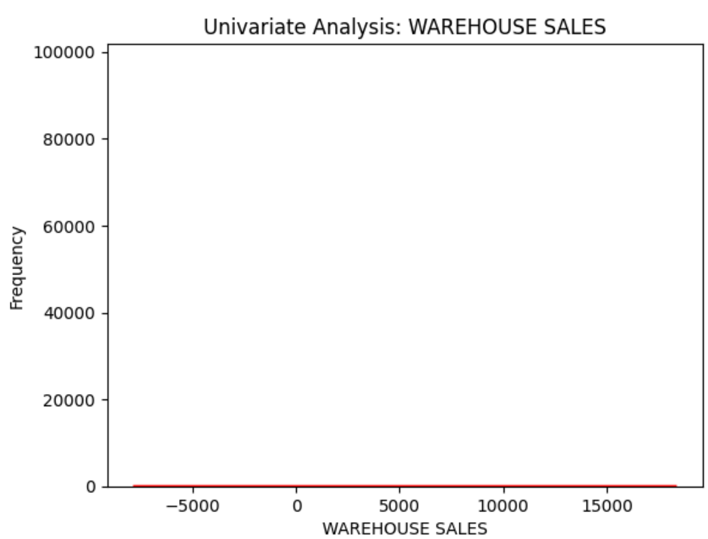
Code:

```
# Univariate analysis: Histograms for numerical columns

numerical_cols = ['RETAIL SALES', 'RETAIL TRANSFERS', 'WAREHOUSE SALES']

for col in numerical_cols:
   sns.histplot(data[col], kde=True, color='red')
   plt.title(f'Univariate Analysis: {col}')
   plt.xlabel(col)
   plt.ylabel('Frequency')
   plt.show()
```

Result:


Univariate Analysis: RETAIL SALES
Frequency
120000
100000
80000
60000
40000
20000
0
0
500
1000
1500
2000
2500
RETAIL SALES


Univariate Analysis: RETAIL TRANSFERS
Frequency
175000
150000
125000
100000
75000
50000
25000
0
0
250
500
750
1000
1250
1500
1750
2000
RETAIL TRANSFERS


Univariate Analysis: WAREHOUSE SALES
Frequency
100000
80000
60000
40000
20000
0
−5000
0
5000
10000
15000
WAREHOUSE SALES

The histogram indicates that retail sales are right-skewed, with a large number of transactions falling in the lower sales range. This suggests that the dataset contains a small number of high-value transactions, while the majority are smaller.

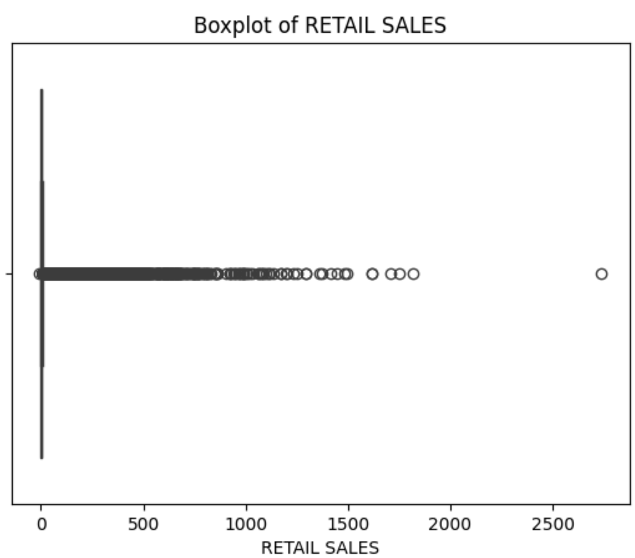
4.1.2 Boxplot for Detecting Outliers

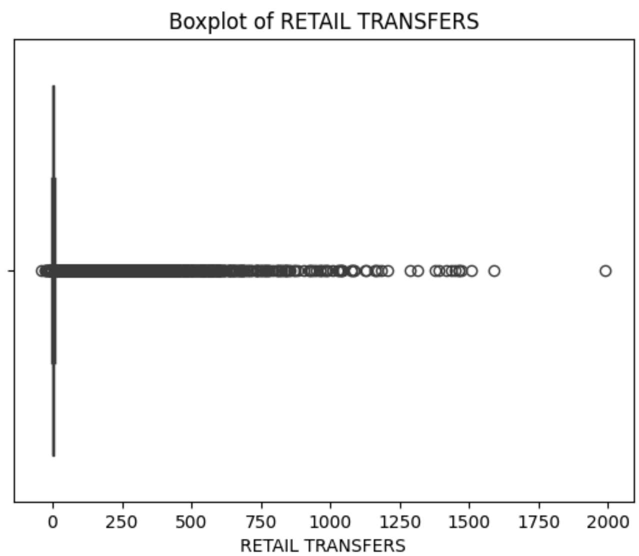
Boxplots for the numerical columns revealed the presence of outliers. These outliers may indicate either errors in data entry or legitimate extreme value.

Code:

```
# Boxplot for detecting outliers
numerical_cols = ['RETAIL SALES', 'RETAIL TRANSFERS', 'WAREHOUSE SALES']

for col in numerical_cols:
   sns.boxplot(x= data[col])
   plt.title(f'Boxplot of {col}')
   plt.show()
```

Result:



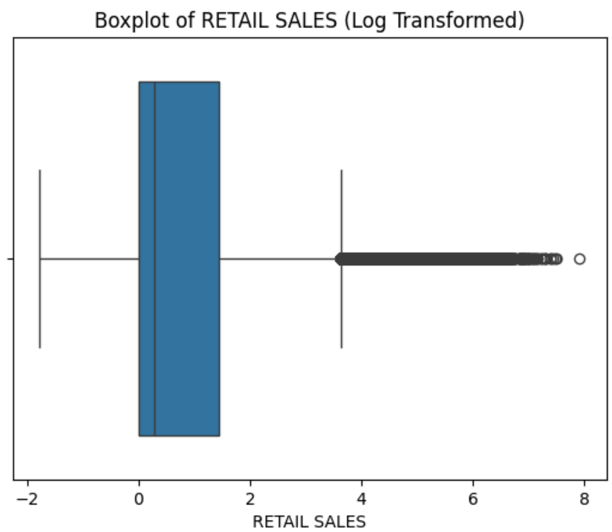


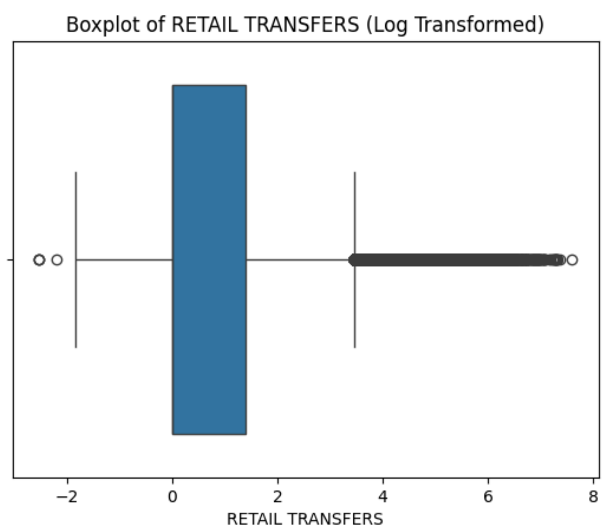
Here, boxplots are just showing like a single line, it likely indicates that the numerical columns (RETAIL SALES, RETAIL TRANSFERS, WAREHOUSE SALES) have little or no variation, or that the data in these columns contains extreme outliers or is not distributed properly.

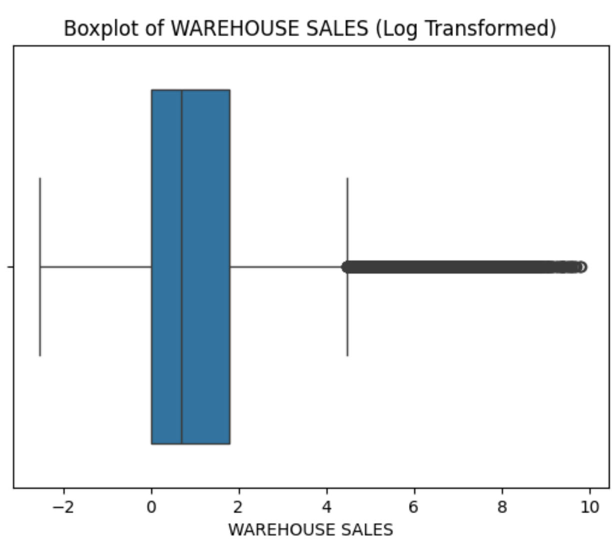
Code:

```
import numpy as np
for col in numerical_cols:
    sns.boxplot(x=np.log1p(data[col]))  # Log transform to handle skewness
    plt.title(f'Boxplot of {col} (Log Transformed)')
    plt.show()
```

Result:


Boxplot of RETAIL SALES (Log Transformed)
−2
0
2
4
6
8
RETAIL SALES


Boxplot of RETAIL TRANSFERS (Log Transformed)
−2
0
2
4
6
8
RETAIL TRANSFERS


Boxplot of WAREHOUSE SALES (Log Transformed)
−2
0
2
4
6
8
10
WAREHOUSE SALES

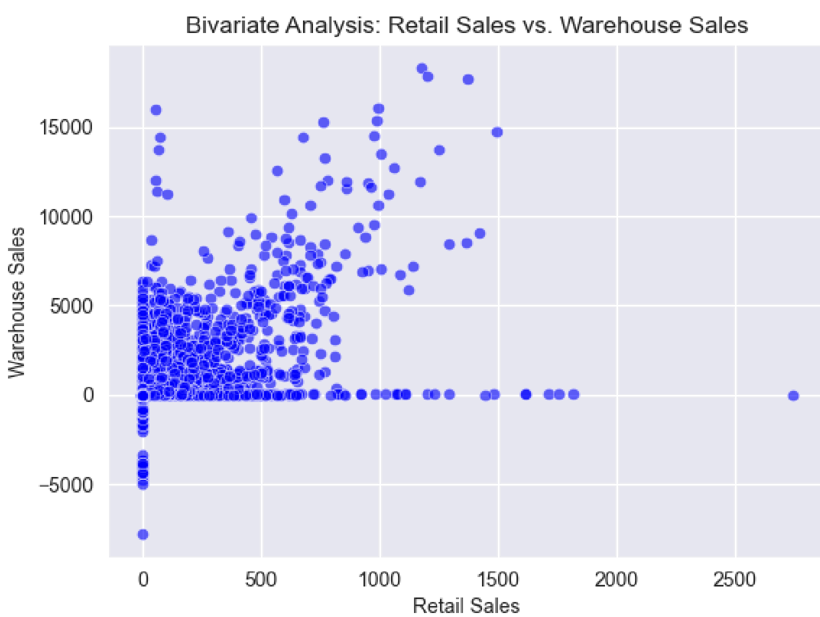
4.2 Bivariate Analysis

A scatter plot was used to explore the relationship between retail sales and warehouse sales:

Code:

```
# Bivariate analysis: Scatter plot for RETAIL SALES vs. WAREHOUSE SALES
sns.scatterplot(x='RETAIL SALES', y='WAREHOUSE SALES', data=data, color='blue',
alpha=0.6)
plt.title('Bivariate Analysis: Retail Sales vs. Warehouse Sales')
plt.xlabel('Retail Sales')
plt.ylabel('Warehouse Sales')
plt.show()
```

Result:



The scatter plot reveals a positive correlation between retail and warehouse sales, suggesting that higher retail sales often correlate with higher warehouse sales. This relationship may indicate that increased demand from retail channels leads to higher inventory turnover in warehouses.

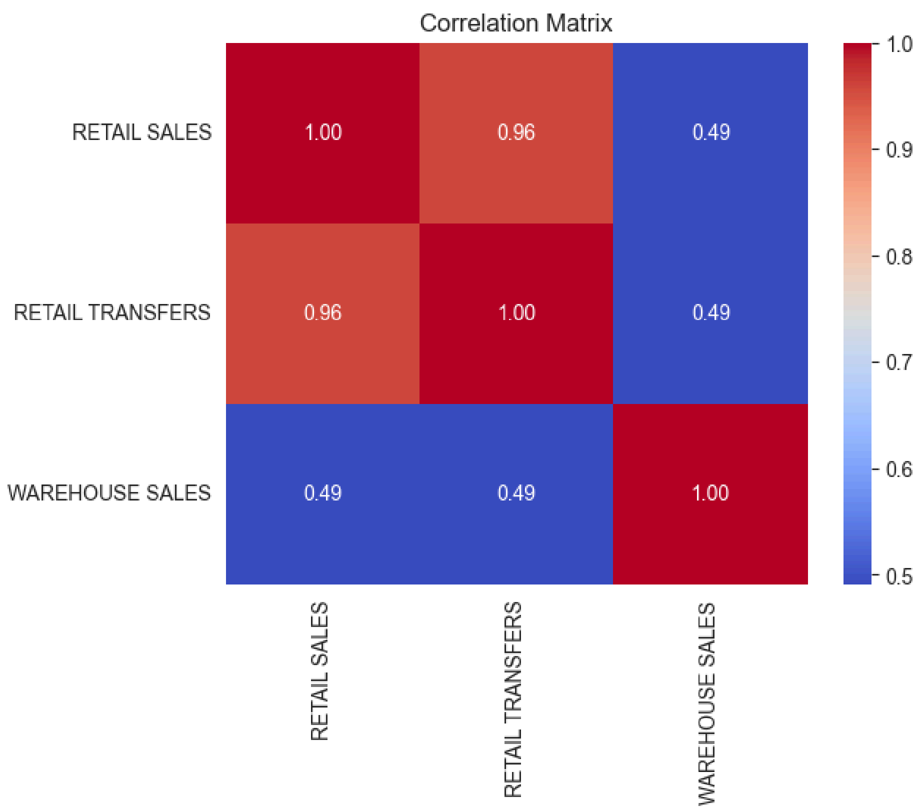
4.3 Multivariate Analysis: (using Correlation Matrix)

A heatmap was created to analyze the correlation between numerical columns

Code:

```
# Select numerical columns for the correlation matrix
numerical_cols = ['RETAIL SALES', 'RETAIL TRANSFERS', 'WAREHOUSE SALES']

# Compute the correlation matrix
correlation = data[numerical_cols].corr()

# Plot the heatmap
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```

Result:



The heatmap shows that RETAIL SALES and RETAIL TRANSFERS are strongly positively correlated, meaning that higher retail sales tend to coincide with higher retail transfers. The correlation between WAREHOUSE SALES and RETAIL SALES is also positive but weaker.

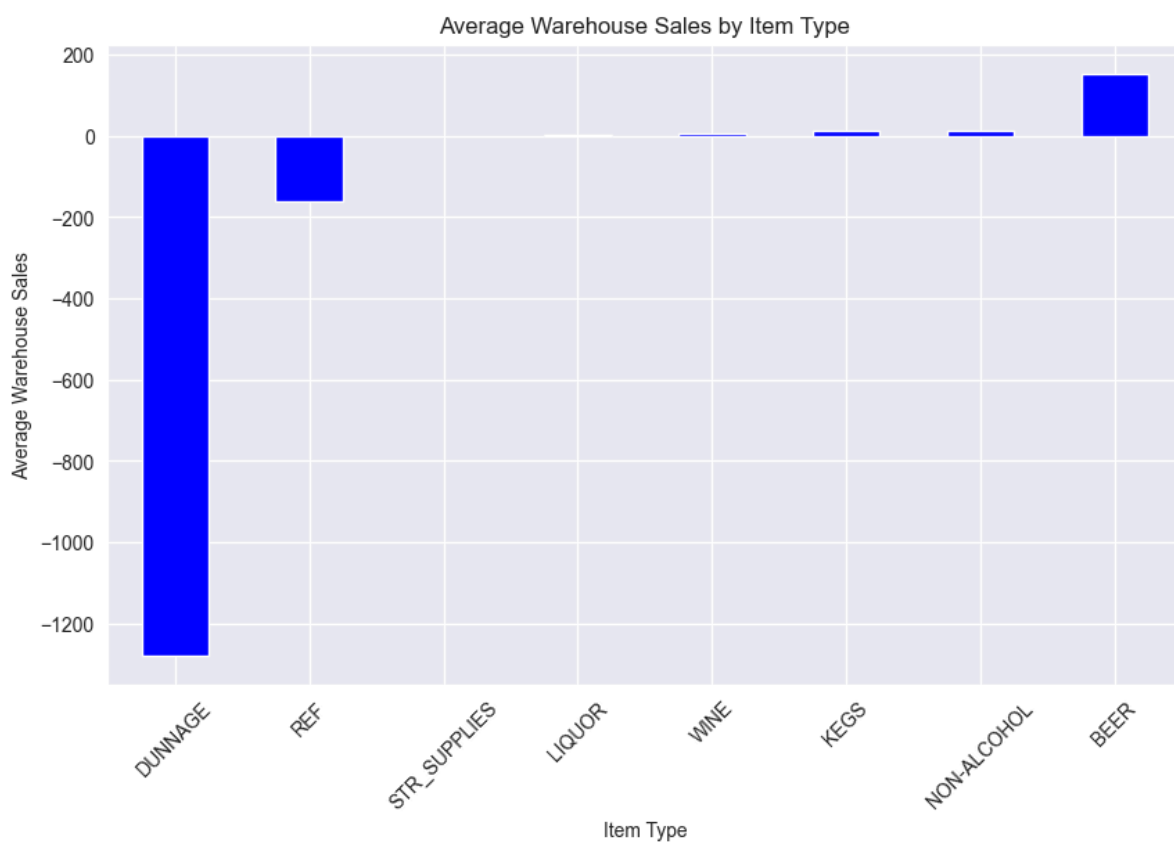
4.3 Grouped Analysis

Here, we calculated the average retail sales by supplier to understand supplier performance.The bar plot reveals significant variation in retail sales across different suppliers, indicating that some suppliers perform better in retail sales than others.

Code:

```
# Grouped analysis: Mean Warehouse Sales by Item Type

item_type_sales = data.groupby('ITEM TYPE')['WAREHOUSE SALES'].mean().sort_values()

# Plot the results
item_type_sales.plot(kind='bar', title='Average Warehouse Sales by Item Type', figsize=(10, 6),
color='blue')
plt.ylabel('Average Warehouse Sales')
plt.xlabel('Item Type')
plt.xticks(rotation=45)
plt.show()
```

Result:



Github link for complete code:
https://github.com/sadneya145/Exploratory-Data-Analysis-EDA-on-a-Dataset.git

CONCLUSION

The exploratory data analysis (EDA) conducted on the dataset related to retail and warehouse transactions provides a comprehensive understanding of the data's structure, trends, and relationships. Key findings include the identification of missing or inconsistent data, detection of outliers, and visualization of sales distributions. The analysis revealed significant correlations between various transaction types and offered insights into supplier performance, enabling actionable recommendations for improving operational efficiency and decision-making. By uncovering patterns and addressing data quality issues, this EDA lays a strong foundation for more advanced analytical techniques or predictive modeling, helping stakeholders make informed business decisions.

REFERENCES

1. Dataset from Warehouse and Retail Sales - Catalog
2. Python Software Foundation. (n.d.). Python documentation. Retrieved from https://docs.python.org
3. Pandas Documentation. (n.d.). Retrieved from https://pandas.pydata.org
4. Matplotlib: For creating visualizations. Documentation: https://matplotlib.org
5. Seaborn: For statistical plots like histograms, scatter plots, and heatmaps. Documentation: https://seaborn.pydata.org
6. Agratas EduTech Curriculum