WALMART EXPLORATORY DATA ANALYSIS (EDA)

Prepared for Walmart

Presented To

INVENTORY MANAGERS

Presented By

AMALAI RHAM MACHUNGA

WALMART DATA ANALYSIS

Case Study

- I am a Data Analyst recently recruited by Walmart which has multiple retail stores nationwide. My task is to help the inventory managers with insights that could help them manage inventories – to match demand and supply.
- I have been assigned to conduct data analysis and develop data-driven recommendations for the inventory manager.
- I have one (1) week to present my analysis and recommendations to the managers.

About Walmart

Walmart is one of the world's leading multinational retail corporations with a huge turnover. It was founded in the year **1962** and was incorporated in the year **1969** with its headquarters in Bentonville, Arkansas.

Besides being a huge firm, Walmart is still controlled by the Walton family. It operates globally with more than **11,700** retail stores in **28**+ countries. Also, it serves its e-retailers in eleven countries.

Dataset Description

Dataset: https://www.kaggle.com/datasets/asahu40/walmart-data-analysis-and-forcasting

Initially, the dataset had 8 columns and 6,435 rows, I feature-engineered the columns to 11 by splitting the 'Date' column into 'Day', 'Month', and 'Year'.

Interestingly, I found a strong correlation between the 'Weekly_Sales' and the 'Month' columns.

Description of the Columns

Store: *Store number.*
Date: *Week of sales.*
Weekly_Sales: *Weekly sales for the given store.*
Holiday_Flag: *Whether the week is a special holiday week or not (1 = holiday week; 0 = non-holiday week).*
Temperature: *Average temperature in the region for the given week.*
Fuel_Price: *Cost of fuel in the region for the given week.*
CPI: *Consumer price index for the given week.*
Unemployment: *Unemployment rate for the region for the given week.*

Tools Used

Python, Visual Studio Code, GitHub, Microsoft Word

Issues / Difficulties Encountered

1. The ‘Date’ column was in a String (Object) datatype format; I needed to convert it to a datetime datatype. I used the .to_datetime function of the pandas dataframe to convert the date column into a datetime format.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   object
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

Fig: Object (String) datatype

```
Store                    int64
Date            datetime64[ns]
Weekly_Sales           float64
Holiday_Flag             int64
Temperature            float64
Fuel_Price             float64
CPI                    float64
Unemployment           float64
dtype: object
```

Fig: Datetime format

2. Converting the exponential outputs of the ‘Weekly_Sales’ column from exponential to integer values by using the .astype(int) function to make it easier to read.

```
Total_weekly_sales = df.groupby('Store')['Weekly_Sales'].sum()
print("Total Weekly Sales For each store: ", Total_weekly_sales)
```

[109] ✓ 0.0s

```
Total Weekly Sales For each store:  Store
1     2.224028e+08
2     2.753824e+08
3     5.758674e+07
4     2.995440e+08
5     4.547569e+07
6     2.237561e+08
7     8.159828e+07
8     1.299512e+08
9     7.778922e+07
10    2.716177e+08
11    1.939628e+08
12    1.442872e+08
13    2.865177e+08
14    2.889999e+08
15    8.913368e+07
16    7.425243e+07
17    1.277821e+08
18    1.551147e+08
19    2.066349e+08
20    3.013978e+08
21    1.081179e+08
22    1.470756e+08
23    1.987506e+08
24    1.940160e+08
...
43    9.056544e+07
44    4.329309e+07
45    1.123953e+08
Name: Weekly_Sales, dtype: float64
```

Fig: Exponential outputs

```
Average_weekly_sales = df.groupby('Store')['Weekly_Sales'].mean().astype(int)
print("Average Weekly Sales For each store: ", Average_weekly_sales)
```

[102] ✓ 0.0s

```
Average Weekly Sales For each store:  Store
1     1555264
2     1925751
3      402704
4     2094712
5      318011
6     1564728
7      570617
8      908749
9      543980
10    1899424
11    1356383
12    1009001
13    2003620
14    2020978
15     623312
16     519247
17     893581
18    1084718
19    1444999
20    2107676
21     756069
22    1028501
23    1389864
24    1356755
...
43     633324
44     302748
45     785981
Name: Weekly_Sales, dtype: int32
```

Fig: Using the .astype(int) function

3. I encountered ‘depreciation’ warnings, which I ignored by adding the Python script below.

```
import warnings
warnings.filterwarnings("ignore")
```

Fig: Python script to ignore depreciation warnings

Findings

1. Store Number '20' has the highest Weekly Sales across all 45 Walmart Stores.
2. Store Number '33' has the lowest Weekly Sales across all 45 Walmart Stores.
3. The Stores experience more Weekly Sales when it is a non-holiday week (0).
4. Q4 (October - December) Weekly Sales increases are experienced across all stores.

Recommendations

1. Increase inventories during the Q4 - fourth quarter to match the increasing sales.
2. Increase Inventories on days that are marked as non-holiday weeks as compared to holiday weeks.