MINI PROJECT 1 – WALMART DATASET

Data and Source:

The Walmart Dataset which has the Walmart stores sales data can be found on the Kaggle dictionary of datasets by going to the below link:
https://www.kaggle.com/datasets/rutuspatel/walmart-dataset-retail

The dataset consists of data of sales from 2010-02-05 to 2012-11-01, in the file Walmart Store sales. Within this file you will find the following fields as mentioned on the Kaggle:

Store - the store number

Date - the week of sales

Weekly_Sales - sales for the given store.

Holiday_Flag - whether the week is a special holiday week 1 – Holiday week 0 – Non-holiday week.

Temperature - Temperature on the day of sale

Fuel_Price - Cost of fuel in the region

CPI – Prevailing consumer price index

Unemployment – The prevailing unemployment rate

Description of Data Exploration and Data Cleaning:

In order to proceed with the analysis, it is very important to identify and understand the data. The structure and size of data, role that each column plays in the dataset, the datatype of the data.

In [181]:
```
df.info()
#The info() method prints information about the DataFrame.
#The information contains the number of columns, column labels, column data types, memory usage,
#range index, and the number of cells in each column (non-null values).
```

```
<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
```

The above screenshot shows the info() method that gives an information about all the Columns and datatypes, from which we can infer which column datatypes should be changed according to our requirements. Also, it shows us about how many non-null values are present in each column.

Dtypes: Another way of data exploration that shows the datatype of each column as below.

```
In [184]: df.dtypes
          #Return the dtypes in the DataFrame. This returns a Series with the data type of each column.
          #Just checking again if the Date column has been converted to datetime data type or not.
```

```
Out[184]: Store                    int64
          Date           datetime64[ns]
          Weekly_Sales          float64
          Holiday_Flag            int64
          Temperature           float64
          Fuel_Price            float64
          CPI                   float64
          Unemployment          float64
          dtype: object
```

The describe() method returns description of the data in the DataFrame. If the DataFrame contains numerical data, the description contains these information for each column: count - The number of not-empty values.

mean - The average (mean) value.

std - The standard deviation.

min-The minimum value in that column

25%: what is 25th percent of the data in that column.

50%: what is 50th percent of the data in that column.

75%: what is 75th percent of the data in that column.

max: The maximum value of that column.

This helps us to find required outliers.

```
In [185]: df.describe()
```

Out[185]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.999151
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.875885
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.879000
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.891000
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.874000
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.622000
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.313000

Data Cleaning and Formatting

Now, since the column “Date” is of object/string type, hence we need to convert it to required datetime64(ns) format so that we perform required operations on Date column through pandas. Changing the date column datatype from object to Datetime using below code. Using to_datetime function of the datetime package, we are converting its data type and specifying the original format that already exist in our imported dataset which is dd-mm-yyyy. Important step in data cleaning and formatting.

In [183]:
```
from datetime import datetime #import datetime package
df['Date'] = pd.to_datetime((df['Date']), format="%d-%m-%Y")

print(df)
```

```
      Store       Date  Weekly_Sales  Holiday_Flag  Temperature  Fuel_Price  \
0         1 2010-02-05    1643690.90             0        42.31       2.572
1         1 2010-02-12    1641957.44             1        38.51       2.548
2         1 2010-02-19    1611968.17             0        39.93       2.514
3         1 2010-02-26    1409727.59             0        46.63       2.561
4         1 2010-03-05    1554806.68             0        46.50       2.625
...     ...        ...           ...           ...          ...         ...
6430     45 2012-09-28     713173.95             0        64.88       3.997
6431     45 2012-10-05     733455.07             0        64.89       3.985
6432     45 2012-10-12     734464.36             0        54.47       4.000
6433     45 2012-10-19     718125.53             0        56.47       3.969
6434     45 2012-10-26     760281.43             0        58.85       3.882

            CPI  Unemployment
0    211.096358         8.106
1    211.242170         8.106
2    211.289143         8.106
3    211.319643         8.106
4    211.350143         8.106
...         ...           ...
6430 192.013558         8.684
6431 192.170412         8.667
6432 192.327265         8.667
6433 192.330854         8.667
6434 192.308899         8.667

[6435 rows x 8 columns]
```

Separating the month, date and year column from Date Column and creating separate column for each individual information using below code.

In [191]:
```
df['Month'] = pd.DatetimeIndex(df['Date']).month
df['Day'] = pd.DatetimeIndex(df['Date']).day
df

#Just like year, separating the Month and Day from Date column and creating a separate
#column for those so that we can easily access the month as integer and use it in our code.
```

Out[191]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment	Year	Month	Day
0	1	02-05-2010	1643690.90	0	42.31	2.572	211.096358	8.106	2010	2	5
1	1	02-12-2010	1641957.44	1	38.51	2.548	211.242170	8.106	2010	2	12
2	1	02-19-2010	1611968.17	0	39.93	2.514	211.289143	8.106	2010	2	19
3	1	02-26-2010	1409727.59	0	46.63	2.561	211.319643	8.106	2010	2	26
4	1	03-05-2010	1554806.68	0	46.50	2.625	211.350143	8.106	2010	3	5
...	...	...	...	...	...	...	...	...	...	...	...
6430	45	09-28-2012	713173.95	0	64.88	3.997	192.013558	8.684	2012	9	28
6431	45	10-05-2012	733455.07	0	64.89	3.985	192.170412	8.667	2012	10	5
6432	45	10-12-2012	734464.36	0	54.47	4.000	192.327265	8.667	2012	10	12
6433	45	10-19-2012	718125.53	0	56.47	3.969	192.330854	8.667	2012	10	19
6434	45	10-26-2012	760281.43	0	58.85	3.882	192.308899	8.667	2012	10	26

6435 rows × 11 columns

Now lets begin with the analysis wherein I have solved two comparison questions

Q1. Find out the annual increment/decrement percentage of sales of each store every consecutive year for a period of 3 years from 2010 to 2012 and determine which financial year the stores performed worst/best.

Here, we are finding out for each store that what was the increment or decrement for each store in two consecutive years i.e from 2010 to 2011 and other from 2011 to 2012. This will help us find which year was best for the stores overall, and how much stores actually progressed over time. If not progressed, what was their decrement percentage?

Here the unit of analysis is total sales for each year and the comparison factor is all the stores from 1 to 45. We can compute the comparison between two stores can comparing their increment/decrement over two years and decide which store is better over other.

Program & Description:

```
In [189]:
First_yr_Sales = df[(df['Year'] == 2010)].groupby('Store')['Weekly_Sales'].sum().round()
#For year 2010, grouped Store column and performed Sum() on 'Weekly Sales' for each store which will give us the combined
#sales for that particular store in the entire year of 2010.

Second_yr_Sales = df[(df['Year'] == 2011)].groupby('Store')['Weekly_Sales'].sum().round()
#For year 2011, again grouped Store column because we want sum for each store and performed Sum() on 'Weekly Sales' for
#each store which will give us the combined
#sales for that particular store in the entire year of 2011.

Third_yr_Sales = df[(df['Year'] == 2012)].groupby('Store')['Weekly_Sales'].sum().round()
#For year 2012, again grouped Store column because we want sum for each store and performed Sum() on 'Weekly Sales' for
#each store which will give us the combined
#sales for that particular store in the entire year of 2012.

#Calculating the Increment for two consecutive years using appropriate mathematical formula and creating a datafram out of it.
pd.DataFrame({ '2010 Sales': First_yr_Sales, '2011 Sales': Second_yr_Sales, '2012 Sales': Third_yr_Sales,
               'Increment % 2010-2011': ((Second_yr_Sales-First_yr_Sales)/First_yr_Sales)*100,
               'Increment % 2011-2012': ((Third_yr_Sales-Second_yr_Sales)/Second_yr_Sales)*100
}).sort_values(by=['Store'])
#Sorting the Stores in the ascending order
```

Output:

Store	2010 Sales	2011 Sales	2012 Sales	Increment % 2010-2011	Increment % 2011-2012
1	73278832.0	80921919.0	68202058.0	10.430143	-15.718684
2	95277864.0	98607881.0	81496695.0	3.495058	-17.352757
3	18745419.0	20816877.0	18024440.0	11.050476	-13.414294
4	95680471.0	111092293.0	92771189.0	16.107594	-16.491787
5	14836031.0	16470820.0	14168838.0	11.019045	-13.976123
6	76912321.0	80528763.0	66315047.0	4.702032	-17.650483
7	25568078.0	30662641.0	25367556.0	19.925483	-17.268848
8	43204475.0	47512786.0	39233920.0	9.971909	-17.424501
9	25129220.0	28685970.0	23974030.0	14.153842	-16.425939
10	94472202.0	98916895.0	78228617.0	4.704763	-20.914807
11	65255138.0	70523583.0	58184066.0	8.073609	-17.497008
12	48370384.0	52582001.0	43334846.0	8.707016	-17.586160
13	95272735.0	104537513.0	86707455.0	9.724480	-17.056134
14	105462242.0	106096271.0	77441398.0	0.601191	-27.008370
15	32023528.0	32282625.0	24827531.0	0.809083	-23.093209
16	24728633.0	27421367.0	22102425.0	10.889134	-19.397071
17	41104920.0	46391840.0	40285379.0	12.862013	-13.162791
18	55978417.0	54217740.0	44918577.0	-3.145278	-17.151514
19	72580529.0	74841900.0	59212433.0	3.115672	-20.883311
20	101733081.0	109837002.0	89827709.0	7.965866	-18.217261
21	37631108.0	40234884.0	30251887.0	6.919212	-24.811795

From the ouput, we can infer that almost all of the stores performed positively from year 2010-2011, meaning, their growth is positive and their sales increased over time. However, from 2011-2012, the sales declines for almost all of the stores which shows that 2012 was the worst year for the stores and their Sales was highest in 2011, showing that it was the best year for all the stores.

Q2. Find out top 5 stores and which particular month for that store has the highest sales in year 2010 and 2011.

Here, we are finding out which store performed the best in 2010 and 2011 along with which month was the most successful month for that particular store. In this way, we are finding the top 5 stores that performed the best for a particular month according to their Sales amount.

Here, the unit of analysis is Store and Month as we are finding out which store and what particular month that store gave their highest output and Comparison factor should be Sum of Monthly Sales as we are displaying the store with highest sales at the top.

In order to proceed with this, first, we had to find out the sum of all the sales in the entire month, for all months, for all the stores. Once we find that, we can find out which month had the highest sales and display that month for that store along with its sales for that month.

Program & Explaination:

For year 2010

```
#Now that we have to find which Store and which month for that particular store we have highest sales,
#I have performed group by on both Store an Month and since we are finding highest sales, I performed sum operation on
#Weekly Sales of all weeks of that particular month of year 2010.
new_df = df[(df['Year'] == 2010)].groupby(["Store", "Month"])['Weekly_Sales'].sum().round()
#Grouped the store and month columns and took the sum on all the sales for that month by setting year as df['Year'] == 2010.

df2 = pd.DataFrame(new_df.sort_values(ascending = False)) #Creating a dataframe and sorting the values in descending order
#as we want the highest sales on top.
df2.head()
#Using head function because we are displaying just the Top 5 stores and the highest sales month.
```

Output:

		Weekly_Sales
Store	**Month**	
20	**12**	13553792.0
14	**12**	13064273.0
10	**12**	12931000.0
13	**12**	12587690.0
4	**12**	12466674.0

For year 2011:

```
new_df = df[(df['Year'] == 2011)].groupby(["Store", "Month"])['Weekly_Sales'].sum().round()
#Setting the year as 2011 and grouping the Store and Month columns and performing sum function on Weekly Sales.
#At last, rounding out the output to avoid the output in unreadable format.

df3 = pd.DataFrame(new_df.sort_values(ascending = False))
df3.head()
#Using head function because we are displaying just the Top 5 stores and the highest sales month.
```

Output:

Store	Month	Weekly_Sales
20	12	13206333.0
4	12	13144847.0
13	12	12800264.0
14	12	12491243.0
10	12	12471117.0

From the above screenshots, we can infer that, for all top 5 stores, the 12th Month, December, had the highest sales with Store 20 bagging the topmost position. Also, we can see that for both year, there is some similarity in the store numbers, meaning that store 20,4,13,14,10 are only appearing in top 5 list for both years. Hence, these are some best performing stores with Store 20 being at Top.