```
# -*- coding: utf-8 -*-
"""Project-Walmart Analysis.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1vGfuyP2GSUc6Zh7HRm3K5CCotQ5mp0Z7
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

#Importing the Data
df = pd.read_csv("Walmart_Store_sales.csv")

df.head()

df.info() #no null value missing value treatment is not required



"""##Which store has max sales"""

total_sales_per_store = df.groupby('Store')['Weekly_Sales'].sum()
total_sales_per_store

store_max_sales = total_sales_per_store.idxmax()
max_sales = total_sales_per_store.max()

print("Store with Max Sales:", store_max_sales)
print("Maxium Sales Amount:", max_sales)



"""#Which store has maximum standard deviation i.e., the sales vary a lot. Also, find out the
coefficient of mean to standard deviation


"""

#Calculating std deviation
std_dev_per_store = df.groupby('Store')['Weekly_Sales'].std()

#Identifying store with the max std deviation in sales
store_max_std_dev = std_dev_per_store.idxmax()
max_std_dev = std_dev_per_store.max()


print("Store with Max Standard Deviation :", store_max_std_dev)
print("Maxium Standard Deviation:", max_std_dev)

#Calculating the mean sales for the store with max std deviation
mean_sales = df[df['Store'] == store_max_std_dev]['Weekly_Sales'].mean()
mean_sales

#Calculate the coefficent of mean to the std deviation
coefficient_of_variation = max_std_dev/mean_sales * 100
print("Coefficent of Mean :", coefficient_of_variation)
```

```
"""*Which store has good quarterly growth rate in Q3'2012"""

#Convert the date into datetime format
df['Date'] = pd.to_datetime(df['Date'], format = '%d-%m-%Y')
df['Date']

#Quarter 2 and 3 of 2012 data
q2_data_2012 = df[(df['Date'] >= '2012-04-01') & (df['Date'] <= '2012-06-30')]
q3_data_2012 = df[(df['Date'] >= '2012-07-01') & (df['Date'] <= '2012-09-30')]

#total sales of each store in Q2
total_sales_q2_2012 = q2_data_2012.groupby('Store')['Weekly_Sales'].sum()
total_sales_q3_2012 = q3_data_2012.groupby('Store')['Weekly_Sales'].sum()

#Calculate growth rate
growth_rate_q3_2012 = (total_sales_q3_2012 - total_sales_q2_2012 ) / total_sales_q2_2012 * 100
growth_rate_q3_2012

#Identifying stoes with good growth sales in Q3
 store_with_good_growth = growth_rate_q3_2012[growth_rate_q3_2012 > 0]
 store_with_good_growth
 print(store_with_good_growth.sort_values(ascending = False))

"""#Some holidays have a negative impact on sales. Find out holidays which have higher sales than the
mean sales in non-holiday season for all stores together



1.    *Mean sales for non-holiday sales for all stores
2.    *Mean sales for each holiday and compare them to the mean of non-holiday sales


"""

##Mean sales for non-holiday sales
mean_sales_non_holiday = df[df['Holiday_Flag'] == 0 ]['Weekly_Sales'].mean()
mean_sales_non_holiday



"""*Holiday Events

**Super Bowl**: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13

**Labour Day**: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13

**Thanksgiving**: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13

**Christmas**: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13
"""

holiday_dates = {
    'Super Bowl' : pd.to_datetime(['2010-02-12','2011-02-11','2012-02-10', '2013-02-08']),
    'Labour Day' : pd.to_datetime(['2010-09-10', '2011-09-11', '2012-09-07','2013-09-06']),
    'Thanksgiving': pd.to_datetime(['2010-11-26', '2011-11-25', '2012-11-23','2013-11-29']),
    'Christmas': pd.to_datetime(['2010-12-31', '2011-12-30', '2012-12-28','2013-12-27'])
}
holiday_dates

#Mean Sales for each holiday
holiday_sales = {holiday : df[df['Date'].isin(dates)]['Weekly_Sales'].mean()
                 for holiday, dates in holiday_dates.items()}
holiday_sales

print("Non Holiday mean sales:", mean_sales_non_holiday )
```

```
holiday_sales

#compare the holiday sales with non holiday sales
holidays_higher_than_non_holiday = {holiday : sales for holiday, sales in holiday_sales.items()
                                    if sales > mean_sales_non_holiday }
print("Holidays with Higher sales than the mean non-holiday sales:",
holidays_higher_than_non_holiday)

"""##Provide a monthly and semester view of sales in units and give insights


"""

#Extracting year and month for monthly aggregation
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month

df.head(3)

#Total Monthly sales
monthly_sales = df.groupby(['Year','Month'])['Weekly_Sales'].sum()
monthly_sales

#Creating Semester column
df['Semester'] = df['Month'].apply(lambda x : 'S1' if x <= 6 else 'S2')
df.head(3)

#Total Semester sales
semester_sales = df.groupby(['Year', 'Semester'])['Weekly_Sales'].sum()
semester_sales

#Print Monthly and Semester Sales
print(monthly_sales, semester_sales)

#Plotting Monthly Sales
plt.figure(figsize = (9, 4))
monthly_sales_plot = monthly_sales.unstack(level = 0)
sns.lineplot(data = monthly_sales.unstack(level = 0))
plt.title('Monthly Sales Over the Years')
plt.ylabel ('Total Sales')
plt.xlabel('Month')
plt.xticks(range (1, 13))
plt.legend(title = 'Year', labels = monthly_sales_plot.columns)
plt.show()

#Observation - from Nov to December there are high Sales as people are shopping for christmas

"""###Part 2 -  Statistical Model"""



"""**For Store 1 – Build  prediction models to forecast demand


1.   **Linear Regression** – Utilize variables like date and restructure dates as 1 for 5 Feb 2010
(starting from the earliest date in order). Hypothesize if CPI, unemployment, and fuel price have any
impact on sales.
2.   **Change** dates into days by creating new variable.




"""
```

```
#Importing Library
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

1# Filter data for Store 1
store_1_data = df[df["Store"] == 1]

#Sort by Dates Ascending
store_1_data = store_1_data.sort_values(by='Date')
store_1_data['Date_Num'] = range(1, len(store_1_data) + 1)
store_1_data

#Selecting features and target variable
X = store_1_data [['Date_Num', 'CPI', 'Unemployment', 'Fuel_Price']]
y = store_1_data ['Weekly_Sales']

#Splitting data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =0.2, random_state = 0)

#Building Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

#Prediction on test data
y_pred = model.predict(X_test)
y_pred

#Evaluating the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print('Mean Squared Error:', mse)
print("R^2 Score:", r2)
```