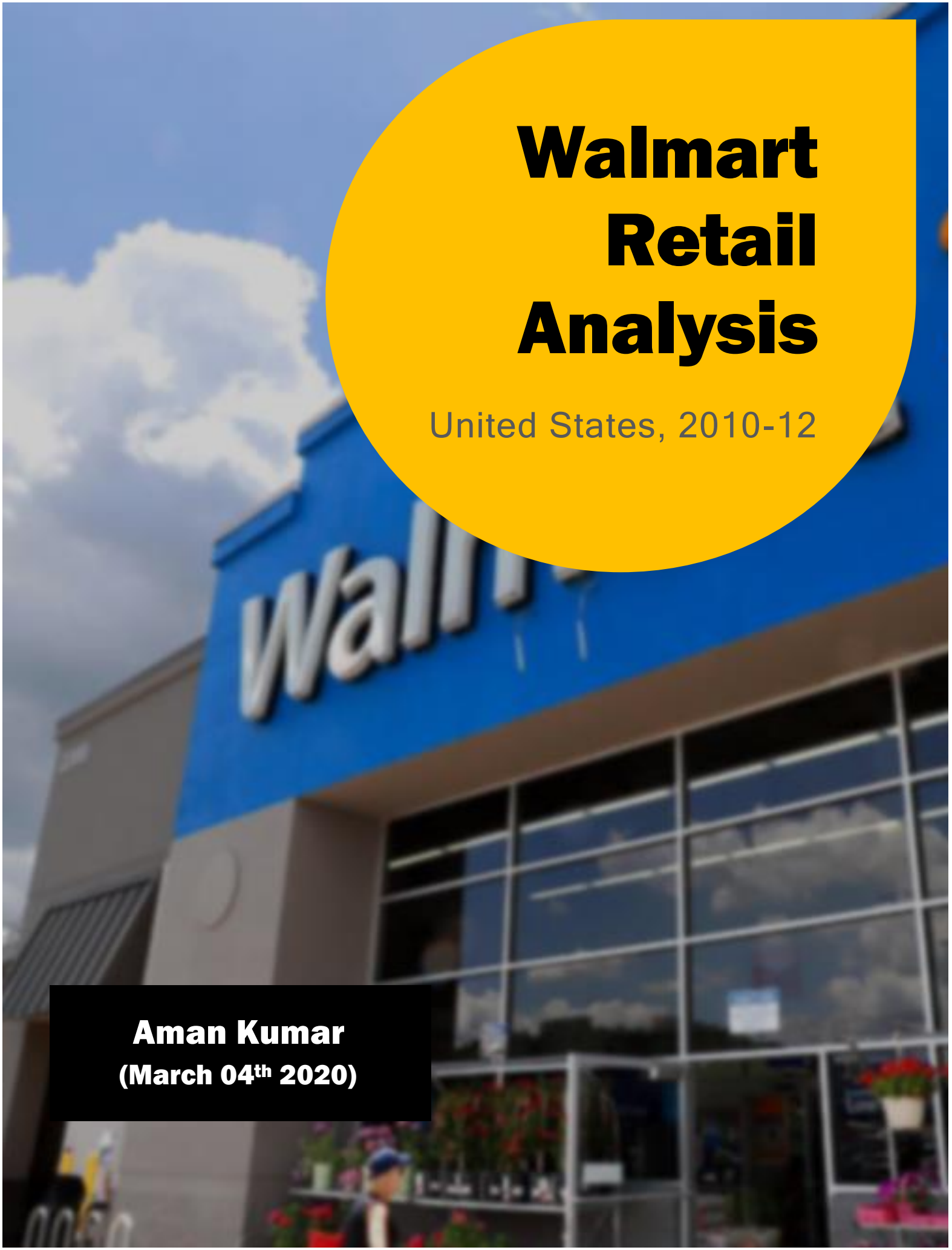


Walmart Retail Analysis

United States, 2010-12

Aman Kumar

(March 04th 2020)

The Whereabouts'

The Problem Statement:

One of the leading retail stores in the US, Walmart, would like to predict the sales and demand accurately. There are certain events and holidays which impact sales on each day. There are sales data available for 45 stores of Walmart. The business is facing a challenge due to unforeseen demands and runs out of stock sometimes, due to the inappropriate machine learning algorithm. An ideal ML algorithm will predict demand at different points of time covering seasonality and ingest factors like economic conditions including CPI, Unemployment Index, etc.

Walmart runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of all, which are the Super Bowl, Labor Day, Thanksgiving, and Christmas.

The weeks including these holidays are weighted five times higher in the evaluation than non-holiday weeks. Part of the challenge presented by this competition is modelling the effects of markdowns on these holiday weeks in the absence of complete/ideal historical data. Historical sales data for 45 Walmart stores located in different regions are available.

Data Sources and Tools Used:

This Project is made as a capstone course-end project for course – "*Understanding Data Science with SAS".* The data sources or the same has been collected through the Simplilearn LMS Portal. The procedures, tools and software used in project are a combination from multiple sources to utilize the full potential in solving the provided problem statements. The tools used are mentioned as under.

- SAS University Edition (Running on VMware Workstation 12 Player)
- VMware Workstation 12 Player
- Python (Running on IBM Watson Studio over Jupyter Notebooks)
- Microsoft Excel 2019 for basic data wrangling and visualizations.

Table of Contents

Understanding Data

Dataset Description:

The dataset used here contains historical data of Walmart and it covers sales from 2010-02-05 to 2012-11-01, in the file Walmart Store sales. Within this following fields are present.

- 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 - Prevailing unemployment rate

The provided dataset contains 8 columns and 6435 row values. Further the wrangling methods used to understand the data in the better context would be mentioned.

Dataset view and basic wrangling

By using the pandas data frame in Python to obtain a general overview of the dataset we get:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106

Project Statements

Basic Project Objectives:

The basic Project objectives is to identify and through the data calculate:

- Which store has the maximum sales
- Which store had maximum standard deviation
- Find the coefficient of mean to standard deviation
- Identify the store which has a positive quarterly growth rate (in Quarter 3, 2012)
- Holidays which have a higher sale than the mean sales in non-holiday season for all stores.
- Monthly and semester view of sales.

The above are the basic statistical tasks to be implemented on the mentioned dataset. Further done a Machine Learning model has to be implemented on the dataset for further analysis.

A Machine Learning algorithm is to be designed to predict the further sales on the basis of the given values. The basic topology and objectives for the Machine Learning Model includes:

1. Linear Regression Model

   - Find out if CPI, unemployment and fuel price impact the sales
   - Find the general insights from the Linear Regression Model.

2. Time Series Forecasting Model

   - Find out if data is fit for time series prediction.
   - Check for white noise
   - Make adjustments in data for holiday events.
   - Build ARIMA model to forecast for upcoming 6 months.

On the basis of the above the models the model with the highest possible accuracy is then selected to be implemented in the analysis of the model.

Configuring Tools

Installing SAS and VMware:

The VMware used here is - VMware Workstation Player 12 running on Windows 10 Pro. The software was downloaded from the website- https://www.vmware.com/in/products/workstation-player/workstation-player-evaluation.html.

The SAS Software used here is running on Windows 10 with 4GB RAM configuration. The same was downloaded from :http://www.sas.com/en_us/software/university-edition/download-software.html

We have used SAS University Edition for the project building process. Since SAS University Edition is a virtualization software, hence a virtualization tool VMware was installed. In SAS University Edition, we need to create a share folder from where we can access the data that we want to use in our SAS Program. In this folder we will place all our data files and then we will upload them in SAS University Edition.

The various procedures to perform this procedure were as under:

- Create folder named as “SASUniversityEdition” in any drive t
- In the folder add a new folder named “myfolders” in “SASUniversityEdition” folder.

Configuring VMware to locate shared folder:

The VMware needs to be configured for a shared folder so that the files running in SAS in VMware have a local shared folder to keep the resources. The myfolders named folder recently created was added as a shared folder in the VMware. The steps involved in doing the same are;

- Open VirtualBox > Select Machine > Settings.
- In navigation Panel Select **Shared Folders** and Click Plus button on right hand side.
- The Add Share dialog box appears.
- From the Folder Path drop-down list, select Other.
- In the Browse for Folder dialog box, open the SASUniversityEdition folder and select the myfolders folder.

Importing Data

Connecting to the VMware and SAS:

The VMware is launched and the downloaded AAS University Edition File is loaded to the VM and launched. The following are the preset configurations under which the VMware is initialized.

- Operating System – Red Hat Enterprise Linux 6 64-BIT
- Version – Workstation 6.5 Virtual Machine
- RAM – 2 Gb

The SAS University Edition was used with the IP Address provided by the VMware once it launches.

Loading the dataset in SAS Environment:

The VMware is launched and the downloaded AAS University Edition File is loaded to the VM and launched. The following are the basic syntax for loading the dataset in the SAS Environment.

```
1  FILENAME REFFILE '/folders/myfolders/Walmart_Store_sales.csv';
2
3  PROC IMPORT DATAFILE=REFFILE
4      DBMS=CSV
5      OUT=WORK.IMPORT;
6      GETNAMES=YES;
7  RUN;
8
9  PROC CONTENTS DATA=WORK.IMPORT; RUN;
10
```

The following syntax loads the Walmart Sales dataset which makes it ready for the analytical purposes. The underlying syntaxes mentioned underneath are used for solving the problem statements of the project.

Problem Solutions

Store having the maximum sales:

For the problem statement we need to calculated the store having the maximum sales value. The following are the procedures involved in doing the same.

- Create a table such that it contains the total sales grouped by the stores.
- View the newly created table, to cross check the values created.
- Select the store from the newly created table where Sales = Maximum.

1. Creating a new table to store the values:

Using the PROC SQL's create table syntax a new table named Sales_total was created. The newly created tables store the store name and counts the total sales provided in the table. The final result is grouped by store column so that one particular row denotes to each of the store. The syntax is:

```
11 PROC SQL;
12
13 CREATE TABLE Sales_total AS
14 SELECT STORE, SUM(Weekly_Sales) FROM IMPORT
15 GROUP BY STORE;
16
17 QUIT;
```

2. Viewing the Table and fetching the maximum sales values:

Using the PROC SQL's statement the store was selected and using the "WHERE" clause of the PROC SQL the sales column = max(sales) column was selected. The required condition was matched. The statement for the above code is as mentioned.

```
23 PROC SQL;
24
25 SELECT STORE FROM Sales_total WHERE (_TEMG001) = (SELECT MAX(_TEMG001) from Sales_total);
26 QUIT;
```

The Store having the maximum Sales as per above code – Store Number 20.

Problem Solutions

Store having the highest standard deviation:

The store having the highest standard deviation has the highest variability and seasonality trends in the sales. The following are the methods to calculate the highest standard deviation

- Creating PROC MEAN for the standard deviation and mean of Weekly_Sales classed by Store.
- View the contents of the PROC Mean dataset.
- Wrangle the dataset and find the highest standard deviation from the dataset.

1. Creating PROC MEANS for the dataset:

Using the PROC MEANS procedure, data is imported and the standard deviation of the data is created. The column selected for the same is Weekly_Sales and the data is classed by Store. The output data is obtained with the name – Statistical_data. The code for the same is:

```
PROC MEANS DATA = IMPORT MEAN STDDEV;
VAR Weekly_Sales;
Class Store;
OUTPUT OUT = Statistical_data;
Run;

PROC PRINT DATA = Statistical_data;
RUN;
```

The output for the dataset is provided as under:

The MEANS Procedure

Analysis Variable : Weekly_Sales			
Store	N Obs	Mean	Std Dev
1	143	1555264.40	155980.77
2	143	1925751.34	237683.69
3	143	402704.44	46319.63
4	143	2094712.96	266201.44
5	143	318011.81	37737.97
6	143	1564728.19	212525.86
7	143	570617.31	112585.47
8	143	908749.52	106280.83

- The Mean and Standard deviation store wise is provided in the resultant dataset.
- The dataset is further wrangled using PROC SQL procedures do that the store with the highest standard deviation is obtained.
- The row Mean is also kept here as we also need to calculate the coefficient of mean to standard deviation.

2. Wrangling the obtained PROC MEANS Dataset:

A new table is created in to the PROC SQL Statement from the original table named - Statistical_data obtained from the PROC MEANS Procedure. This was done because the table so created contained null values in the store, which is filtered w.r.t the PROC SQL Syntax.

STD is selected form the _STAT_ Column, and Value of store is filtered as >1. The obtained dataset is again via a SQL Syntax is filtered so that the store value for the Stastical_table1 is obtained. The query for the same is provided.

```
PROC SQL;
CREATE TABLE Stastical_table1 AS
SELECT Store, Weekly_Sales as Std_dev from Statistical_data where
STORE >=1 and _STAT_ = "STD";
QUIT;

PROC SQL;
SELECT STORE, Std_dev from Stastical_table1 where Std_dev =
(SELECT MAX(Std_dev) from Stastical_table1);

QUIT;
```

The Store having the maximum standard deviation for the Sales = Store 14.

Coefficient of Standard Deviation:

We need to calculate the coefficient of Standard Deviation for the store having the highest standard deviation i.e. Store 14. The Standard Deviation so calculated equals = standard-deviation/mean*100. The various steps associated with this process is as under:

- Similar to the method how Standard deviation table was created, a table for mean was created.
- The mean and standard deviation table was concatenated using an inner join.
- The resultant dataset contains Store, Mean and Standard Deviation
- Calculated column was created and data was filtered with Store = 14.

1. Creating a table for mean from earlier data:

```
PROC SQL;
CREATE TABLE Statistical_table2 AS
SELECT Store, Weekly_Sales as Mean from Statistical_data where
Store >= 1 and _STAT_ = "MEAN";
QUIT;
```

2. Wrangling the created table:

The above PROC SQL syntax creates a table that contains the column STORE and MEAN for the original data sorted store wise. Finally, both the tables, of mean and standard deviation is combined using an inner join such that the resultant table contains, store, mean, standard deviation.

The following is the code for the same in PROC SQL:

```
40 PROC SQL;
41
42 CREATE TABLE Final_stat as
43
44 SELECT Statistical_table1.Store, Statistical_table2.Mean, Statistical_table1.Std_dev
45 from Statistical_table1 Inner join
46 Statistical_table2 on Statistical_table1.Store = Statistical_table2.Store;
47
48 QUIT;
```

The resultant table obtained from this code was:

Store	Mean	Std_dev
1	1555264.3976	155980.76776
2	1925751.3355	237683.69468
3	402704.44105	46319.631557
4	2094712.9607	266201.4423
5	318011.81049	37737.965745
6	1564728.1863	212525.85586
7	570617.30867	112585.46922
8	908749.51839	106280.82988

- The Mean and Standard deviation store wise is provided in the resultant dataset.
- The dataset is further wrangled using PROC SQL procedures so that the calculated column for the coefficient of standard deviation is calculated from the dataset created.
- The following is the syntax for the same.

```
56 PROC SQL;
57
58 SELECT STORE, (Std_dev/Mean)*100 as Coeff from Final_Stat where Store = 14;
59
60 QUIT;
```

Using the following code above the calculated column for the store = 14 was created. The coefficient of standard deviation for this case was as per the formula above.

Problem Solutions

Store having Positive Q3 Growth Rate (2012):

For the problem statement we need to calculate the which stores that have attained a positive growth rate in Quarter 3 (2012) we deploy the maximum sales value. The following are the procedures involved in doing the same.

- Create a table such that it contains the total sales grouped by the stores.
- View the newly created table, to cross check the values created.
- Filter the data where values are sorted for Quarter 2 of 2012.
- Calculate the mean of Sales, grouped by Store for Quarter 2.
- Filter the data again to display values for Quarter 3 of 2012. Calculate mean of sales, store wise.
- Compare the two means obtained with formula mentioned underneath. The highest value obtained by any of the stores clearly says that is showed a growth w.r.t Quarter 2 of 2012.

Growth Rate in Percentage = ((Mean Sales in Q3 (2012) / Mean Sales in Q2 (2012) - 1) * 100

1. Wrangling the datasets:

The dataset was imported and using the WHERE Clause of the PROC SQL, a date filter was applied to display the data where date = Q2 of 2012. The results were stored into a new table named Q2_Sales. Similar to this, another table Q3_Sales were created where date = Q3 of 2012.

The PROC SQL Syntax for the same is provided underneath:

```
80 PROC SQL;
81
82 CREATE TABLE Q2_Sale AS
83 SELECT Store, SUM(Weekly_Sales) AS Mean_Sales_Q2 from IMPORT where Date >= 2012/04/01 and
84 Date <= 31/06/31 GROUP BY STORE;
85
86 QUIT;
87
88 PROC SQL;
89
90 CREATE TABLE Q3_Sale AS
91 SELECT Store, SUM(Weekly_Sales) AS Mean_Sales_Q3 from IMPORT where Date >= 2012/07/01 and
92 Date <= 30/09/30 GROUP BY STORE;
93
94 QUIT;
```

2. Combining the datasets:

The two datasets created were then combined using inner join procedure of PROC SQL. Once done a calculated column was created for growth from the newly created INNER JOINED dataset. The SQL Query for the same is mentioned underneath

```
PROC SQL;

CREATE TABLE Resultant_Sales AS

SELECT Q2_Sale.Store, Q2_Sale.Q2_Mean_Sales, Q3_Sale.Q3_Mean_Sales from Q2_Sale
INNER JOIN Q3_Sale ON Q2_Sale.Store = Q3_Sale.Store;

RUN;

PROC SQL;

CREATE TABLE Growth_Rate AS
SELECT STORE, ((Q3_Mean_Sales/Q2_Mean_Sales)-1)*100 AS GROWTH FROM Resultant_Sales;

RUN;
```

The calculated column “GROWTH” created was joined with original table of Quarter 1 and Quarter 2 sales to form a new table. A PROC SQL query was written on this new dataset to filter the data where growth is positive and then sort the data using the ORDER BY procedure in PROC SQL to arrange the data in ascending order. PROC SQL Syntax for the same is mentioned underneath.

```
PROC SQL;

CREATE TABLE Resultant AS

SELECT Growth_Rate.Store, Resultant_Sales.Q2_Mean_Sales, Resultant_Sales.Q3_Mean_Sales, Growth_Rate.GROWTH from Growth_Rate
INNER JOIN Resultant_Sales ON Growth_Rate.Store = Resultant_Sales.Store;

RUN;

PROC SQL;

SELECT STORE, GROWTH FROM Resultant
WHERE GROWTH > 0 ORDER BY GROWTH;

RUN;
```

The dataset obtained from the query was as under:

Store	GROWTH
23	0.825395
40	1.142841
24	1.652088
44	2.434638
41	2.45698

Store	GROWTH
39	2.478404
26	3.955478
35	4.466637
16	8.488378
7	13.33078

- The obtained table shows Store no 7 with the highest growth rate between Q2 and Q3 of 2012.

Problem Analysis

Comparing Holiday vs. Non-Holiday Sales:

The various holiday seasons as mentioned under the dataset are as below.

- Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13
- Labor Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13
- Thanksgiving: 26-Nov-10, 25-Nov-11, 24-Nov-12, 29-Nov-13
- Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

We first, divide the dataset into 2 parts such that the Table Non_Holiday contains dataset for the non-holiday days and the table Holiday contains Holiday data. The PROC SQL Syntax for doing the same is:

```
PROC SQL;

CREATE TABLE Non_Holiday AS
SELECT * FROM Import where Holiday_Flag = 0;

RUN;

PROC SQL;

CREATE TABLE Holiday AS
SELECT * FROM Import WHERE Holiday_Flag = 1;

RUN;
```

- The average for both the datasets are created which are grouped by store.
- The two created datasets along with the average column is then merged together using INNER JOIN Method of PROC SQL

The syntax to calculate the average column for holiday and non-holiday sales and combine the datasets using inner join in PROC SQL is provided below:

```
PROC SQL;

CREATE TABLE NonHoliday as
SELECT STORE, AVG(Weekly_Sales) as Non_holiday_sum from  Non_Holiday GROUP BY STORE;

QUIT;

PROC SQL;

CREATE TABLE HolidayDay as
SELECT STORE, AVG(Weekly_Sales) as Holiday_sum from  Holiday GROUP BY STORE;

QUIT;

PROC SQL;

CREATE TABLE Comparison AS
SELECT NonHoliday.Store, HolidayDay.Holiday_sum, NonHoliday.Non_holiday_sum from NonHoliday
inner join HolidayDay on HolidayDay.Store = NonHoliday.Store;

RUN;
```

The sample of dataset obtained from the following SQL Syntax is as under:

Obs	Store	Holiday_sum	Non_holiday_sum
1	1	1665747.66	1546957.39
2	2	2079266.90	1914208.81
3	3	437811.05	400064.85
4	4	2243102.62	2083555.84
5	5	359501.61	314892.28
6	6	1680907.93	1555992.87
7	7	672400.27	562964.45
8	8	975330.86	903743.40
9	9	588950.82	540599.33
10	10	2113755.95	1883309.43
11	11	1448394.49	1349464.98
12	12	1138140.42	999291.92

- The columns Holiday_sum contains average of sales over weekends that had a holiday while it is vice versa for the column Non_holiday_sum.
- The dataset mentioned here is grouped according to the Store. Further a calculated column for:
- Non_holiday_sum – Holiday_sum is created. The rows which show a positive value shows that for that store, the sales over holiday weekend were higher than normal weekends which is vice-versa for the left part.

The PROC SQL Query for creating the calculated column is as under:

```
54 PROC SQL;
55
56 CREATE TABLE Tabular as
57 SELECT STORE, (Holiday_sum - Non_holiday_sum) as FinalSales from Comparison;
58
59 RUN;
```

The following syntax mentioned above creates a calculated column for the average sales between weeks containing and not containing a holiday.

The negative values obtained from this data shows the stores where the performance of sales on holidays weren't better and normal weekends contributed more to their sales figures than holidays. While it is vice versa for those stores that showed a positive value.

The following are the list of Stores that depicted a negative value.

Store	FinalSales
30	-1849.66
36	-6312.82
37	-12230.5
38	-4539.2
44	-7218.02

- The Stores 30, 36, 37, 38 and 44 had a negative performance on the holiday. Their average sales figure on normal weekends were much better than the weekends which had a holiday.
- All the stores apart the mentioned stores above had a positive holiday weekend sale where the sales for the weeks that contained a holiday were higher than those that do not contained.

Sales Analysis

Comparing Holiday seasons for Sales Performance:

For comparison of which holiday was the best for the sales figures for the stores, we need to wrangle the dataset in Excel (2019) to get the most out of the data. The following new columns were added into the dataset and then the dataset was imported in the SAS Environment.

- Column for month using formula (= TEXT(Cell, “mmmm”))
- Column of Quarter (Where the months were divided into quarters)
- Column for holiday where type of holiday was written as per the data provided.

The dataset was then loaded to the SAS Environment. And using the select feature of the PROC SQL the data was viewed. The following are the results for the same:

Store	Date	Month	Quarter	Weekly_Sales	Holiday_Flag	Holiday_Type	Temperature	Fuel_Price	CPI	Unemployment
1	05/02/2010	February	1	1643690.9	0		42.31	2.572	211.0963582	8.106
1	12/02/2010	February	1	1641957.44	1	Super_bowl	38.51	2.548	211.2421698	8.106
1	19/02/2010	February	1	1611968.17	0		39.93	2.514	211.2891429	8.106
1	26/02/2010	February	1	1409727.59	0		46.63	2.561	211.3196429	8.106
1	05/03/2010	March	1	1554806.68	0		46.5	2.625	211.3501429	8.106
1	12/03/2010	March	1	1439541.59	0		57.79	2.667	211.3806429	8.106
1	19/03/2010	March	1	1472515.79	0		54.58	2.72	211.215635	8.106
1	26/03/2010	March	1	1404429.92	0		51.45	2.732	211.0180424	8.106
1	02/04/2010	April	1	1594968.28	0		62.27	2.719	210.8204499	7.808
1	09/04/2010	April	1	1545418.53	0		65.86	2.77	210.6228574	7.808
1	16/04/2010	April	1	1466058.28	0		66.32	2.808	210.4887	7.808

Once the dataset is loaded, we create 2 tables with the following features:

- A dataset containing holiday_flag = 0 i.e. data representing non-holiday sales. This dataset contains average of the weekly_sales grouped by Store and Month. Further using a WHERE clause of PROC SQL, data was filtered for months February, September, November and December.

- One more dataset containing holiday_flag = 1 i.e. data representing sales. With the similar variable as that of the earlier dataset.

The PROC SQL Syntax to create the above datasets are as under:

```
PROC SQL;

CREATE TABLE Non_Holiday AS
SELECT STORE, MONTH, AVG(Weekly_Sales) as Non_Holiday_average
FROM IMPORT Where Holiday_Flag = 0 and Month in('February','Septembe','November','December')
Group by STORE, MONTH;

RUN;
```

```
PROC SQL;

CREATE TABLE Holiday AS
SELECT STORE, MONTH, AVG(Weekly_Sales) as Holiday_average
FROM IMPORT Where Holiday_Flag = 1 and Month in('February','Septembe','November','December')
Group by STORE, MONTH;

RUN;
```

Both of these datasets were combined using the one-to-one reading method of SAS. The duplicated variables in the case was auto-removed. The syntax for the following code, as well as the output is mentioned as under.

```
DATA Sales_figure;
SET Non_holiday;
SET Holiday;

RUN;
```

Store	Month	Non_Holiday_average	Holiday_average
1	December	1880595.645	1432391.365
1	February	1601250.7533	1698016.6
1	November	1560262.52	1994472.385
1	Septembe	1462201.852	1569899.7533
2	December	2542786.2363	1812330.535
2	February	1996007.9444	2136391.2633
2	November	1932634.8117	2636463.795
2	Septembe	1763401.521	1828635.5167

The next procedure is to create a calculated column for Sales on holiday – Sales on non-holiday sales. The following column is created and it is viewed as a new table. Later using the PROC Means procedure the data in the mentioned table is analyzed. The following is the PROC SQL query to create a calculated column from this concatenated data above.

```
PROC SQL;

CREATE TABLE Final_Sales AS
SELECT *, (Holiday_average - Non_Holiday_average) as Final_Sale from Sales_figure;

RUN;
```

Later via the PROC MEANS the data is analyzed.

The following are the PROC MEANS measure that are calculated for the above created dataset.

- A holiday wise analysis of the sales.
- A holiday and store wise analysis of sales.
- Analysis of sales where sales on holiday > non-holiday grouped on Store.

The following are the PROC SQL Queries used for implementation for the following statements.

1. Holiday wise analysis of sales data:

```
78 PROC MEANS DATA = Final_Sales;
79
80 VAR Final_Sale;
81 Class Month;
82
83 QUIT;
84
```

Analysis Variable : Final_Sale						
Month	N Obs	N	Mean	Std Dev	Minimum	Maximum
December	45	45	-401288.15	280003.55	-983273.66	30273.03
February	45	45	34570.92	48628.91	-144623.54	140383.32
November	45	45	432010.03	284300.57	-31130.06	1023841.18
Septembe	45	45	69019.53	80172.78	-23444.07	304101.55

- We observe that the holidays in month February (Super Bowl), September (Labours Day) and November (Thanksgiving Day) showed a positive impact on sales, whereas sales figure of Walmart in Christmas, declined.

- The highest sales growth was seen during the Thanksgiving holiday season.

2. Holiday wise analysis of sales data, individual store wise:

```
70 PROC MEANS DATA = Final_Sales;
71
72 VAR Final_Sale;
73 Class Month;
74 Class Store;
75
76 QUIT;
```

Month	Store	N Obs	N	Mean	Std Dev	Minimum	Maximum
December	1	1	1	-448204.28	.	-448204.28	-448204.28
	2	1	1	-730455.70	.	-730455.70	-730455.70
	3	1	1	-109817.37	.	-109817.37	-109817.37
	4	1	1	-825205.98	.	-825205.98	-825205.98

- The holiday wise average view of sales is provided in the table below. The data here is grouped according to the store and if a pdf file the total results have been mentioned. The name for the pdf file is Holiday_comparison_results.

- The above datasets have been attached with the project repository.

Building Reports

Provide Monthly and Semester view of Sales:

We use PROC MEANS Method over the dataset to create the report for monthly and Semester views. The output sample for the same would be attached below and a pdf document named Sales_report would be attached with this repository that contains the entire dataset.

The PROC Means procedure table and syntax for the same is provided below:

Analysis Variable : Weekly_Sales							
Quarter	Month	N Obs	N	Mean	Std Dev	Minimum	Maximum
1	April	630	630	1026761.56	543864.62	232769.09	2565259.92
	February	540	540	1053199.80	564207.06	234218.03	2623469.95
	January	360	360	923884.55	472616.46	231155.90	2047766.07
	March	585	585	1013309.23	529805.74	238084.08	2237544.75
2	August	585	585	1048017.45	542653.06	224031.19	2283540.30
	July	630	630	1031747.58	531141.78	224806.96	2358055.30
	June	585	585	1064324.59	548683.95	238172.66	2363601.47
	May	540	540	1031714.02	536589.41	239206.26	2370116.52
3	December	450	450	1281863.63	774037.72	209986.25	3818686.45
	November	360	360	1147265.90	648832.35	224639.76	3004702.33
	October	585	585	999632.12	517186.65	213538.32	2246411.89
	Septembe	585	585	989335.35	510532.95	229731.98	2202742.90

```
17 MEAN STDDEV MIN MAX;
18
19 VAR Weekly_Sales;
20 CLASS Month;
21 CLASS Quarter;
22 CLASS STORE;
23
24 QUIT;
```

The statement mentioned in the R.H.S returns the following table above. The Weeky_Sales PROC Means table mentioned above contains various statistical measures for the Weekly_Sales column arranged by months and quarters 1,2,3.

The following is the knowledge for the various columns mentioned in the table above:

- Quarter – Contains info for quarter 1/2/3 for a year.
- Months – The months that the following quarters contain.
- N_Obs – Number of observations recorded
- Mean – Mean for the weekly sales figure for the given month
- Stddev – Standard deviation for the sales
- The columns mix and max give the maximum and minimum values for the same.

Note: A table named sales_report is added to the repo, that contains info for the various stores too.

Regression Analysis

Calculating Column Correlations:

For the sake of this project we need to calculate where the columns CPI, Fuel price and Unemployment impacts sales in anyway or not. To calculate such procedures, the correlation concept of mathematics is used. The following are the procedures to implement the task in SAS.

- Turn on the ODS Graphics Feature.
- Using PROC CORR add the data and define the columns
- Set Plot to Histogram matrix
- Run the query
- Turn off the ODS Graphics feature.

The following is the implementation method for the same in SAS:

```
15 ODS GRAPHICS ON;
16
17 PROC CORR DATA=IMPORT Plots = MATRIX(Histogram);
18 VAR Weekly_Sales CPI Unemployment Fuel_Price;
19 RUN;
20
21 ODS GRAPHICS OFF;
```

The values returned from the following SQL Query is as under:

Pearson Correlation Coefficients, N = 6435
Prob > |r| under H0: Rho=0

	Weekly_Sales	CPI	Unemployment	Fuel_Price
Weekly_Sales	1.00000	-0.07263 <.0001	-0.10618 <.0001	0.00946 0.4478
CPI	-0.07263 <.0001	1.00000	-0.30202 <.0001	-0.17064 <.0001
Unemployment	-0.10618 <.0001	-0.30202 <.0001	1.00000	-0.03468 0.0054
Fuel_Price	0.00946 0.4478	-0.17064 <.0001	-0.03468 0.0054	1.00000

- The columns show no a neutral correlation with each other as visible in this matrix chart.
- This indicates that there is almost null effect of the mentioned columns over the sales figure of the Walmart stores, as the change in value of one variable doesn't affect the sales variable

Implementing Linear Regression:

A basic Linear Regression model is created for the dataset using the PROC REG function over the dataset. Here Weekly Sales is the dependent variable and the CPI, Temperature, Fuel Price, Unemployment has been taken as the independent variable.

The results for the linear regression have been added under a LinearRegressionAnalysis-results.pdf. The general results returned from the Linear Regression is as under.

- Implement the Linear Regression Function using Proc Reg.
- Take weekly sales as the dependent variable.
- CPI, Unemployment, Fuel Price and Temperature is taken as dependent variable.
- R-squared error and column wise regression plots are plotted for each variable.

The following is the implementation code of Linear Regression in SAS:

```
* Performing basic linear regression;

PROC REG DATA = IMPORT1;
MODEL Temperature CPI Unemployment Fuel_Price = Weekly_Sales;
RUN;
```

The sample of results which are in output with the following regression model is as under:
(For column fuel price)

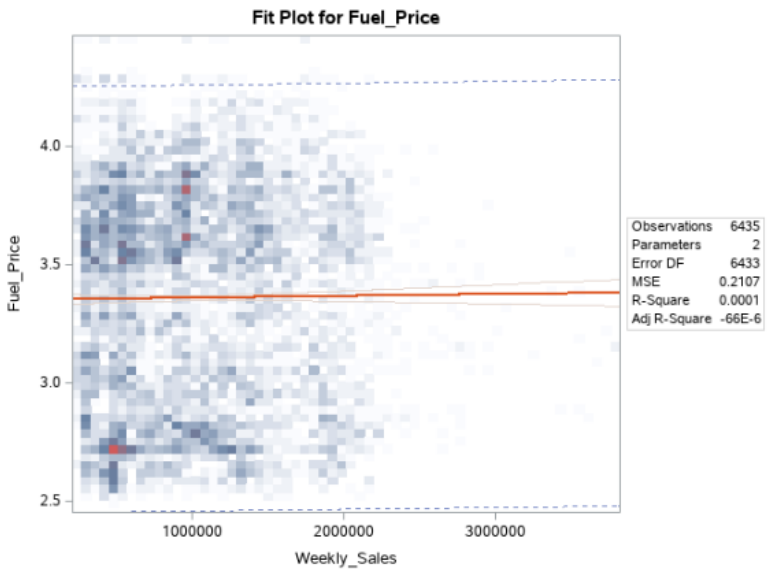
Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	1	0.12142	0.12142	0.58	0.4478
Error	6433	1355.51654	0.21071		
Corrected Total	6434	1355.63796			

Root MSE	0.45903	R-Square	0.0001
Dependent Mean	3.35861	Adj R-Sq	-0.0001
Coeff Var	13.66742		

Parameter Estimates

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	3.35055	0.01206	277.82	<.0001
Weekly_Sales	1	7.697238E-9	1.014014E-8	0.76	0.4478



Timeseries Analysis

Timeseries Analysis for the dataset:

We use timeseries analysis to predict the future sales for 6 months for the provided dataset. For the sake of simplicity, we import the model in SAS and using a PROC SQL Statement filter the data so that data for only Store 1 is displayed.

This was done because we would analyze forecast of sales for store 1 and accordingly the rest of the store values can be predicted.

The following are the methods adopted for doing the same in SAS:

- Check for White Noise in data
- Provide an estimation parameter for Autoregression function.
- Predict the sales for 6 months.

The following are the codes in SAS to implement this procedure.

```
PROC ARIMA DATA = Sales1_data;
Identify Var = Weekly_Sales NLAG=24;
RUN;

*Time series prediction;

PROC ARIMA DATA = Sales1_data;
Identify Var = Weekly_Sales(1) NLAG=24;

ESTIMATE P = 1;
forecast lead=6 interval= MONTH id=Date;

RUN;
```

The detailed report for the analysis in timeseries is mentioned in the attached pdf with the project repository with the name – Timseries_results.pdf

The value of predicted sales, and the associated graphs for the mentioned timeseries procedure is displayed in the sheet overleaf.

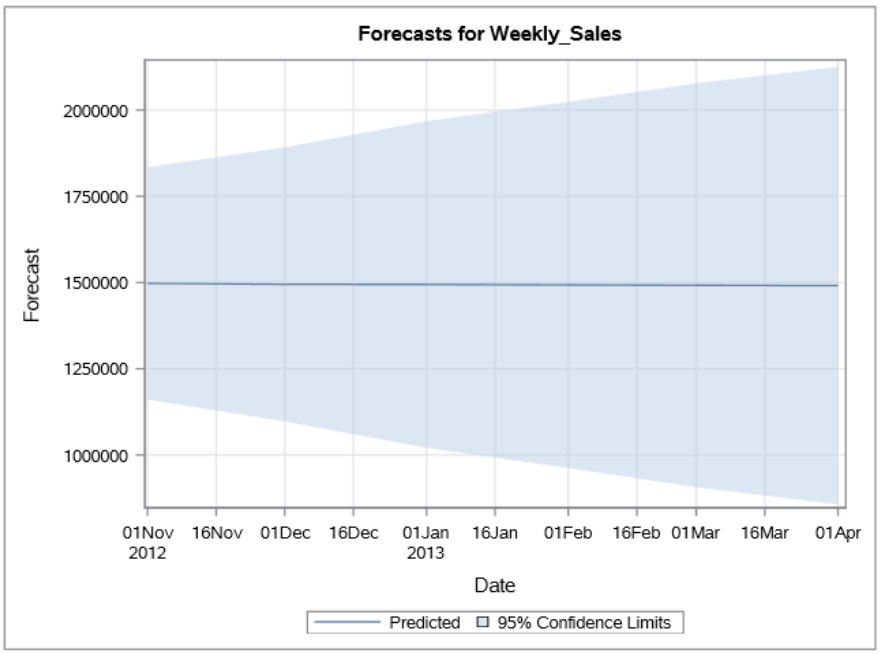
For a much higher accuracy, the Estimator P could be increased.

Timeseries Analysis report:

The report obtained from timeseries analysis is provided in the sheet presented here. Below is the prediction of sales for the next 6 months for store = 1.

The ARIMA Procedure

Forecasts for variable Weekly_Sales				
Obs	Forecast	Std Error	95% Confidence Limits	
144	1497656.4	171845	1160845.7	1834467.1
145	1494738.1	202585	1097678.8	1891797.4
146	1494417.7	241544	1021001.1	1967834.4
147	1493121.4	270798	962366.9	2023875.8
148	1492191.7	298613	906920.3	2077463.0
149	1491124.2	323552	856973.3	2125275.1



Conclusions and Bibliography:

This following mentioned PDF hence completes the objective for the Simplilearn Project. Through the medium we analyzed the sales for the stores, analyzed which holiday season contributed more into the growth of the sales figure.

We also analyzed the stores that displayed the highest and the least sales figures. The model also made used the PROC Means Procedure to analyze the quarterly growth rates for the stores. Further a linear regression and time series modelling procedure displayed the sales value.

All the required PDF's and results are mentioned in the documents attached.

The various sources that contributed towards the development of the project is:

- SAS University Edition and VMware.
- GitHub
- Simplilearn Online Classes, Simplilearn LMS
- SAS Documentations.