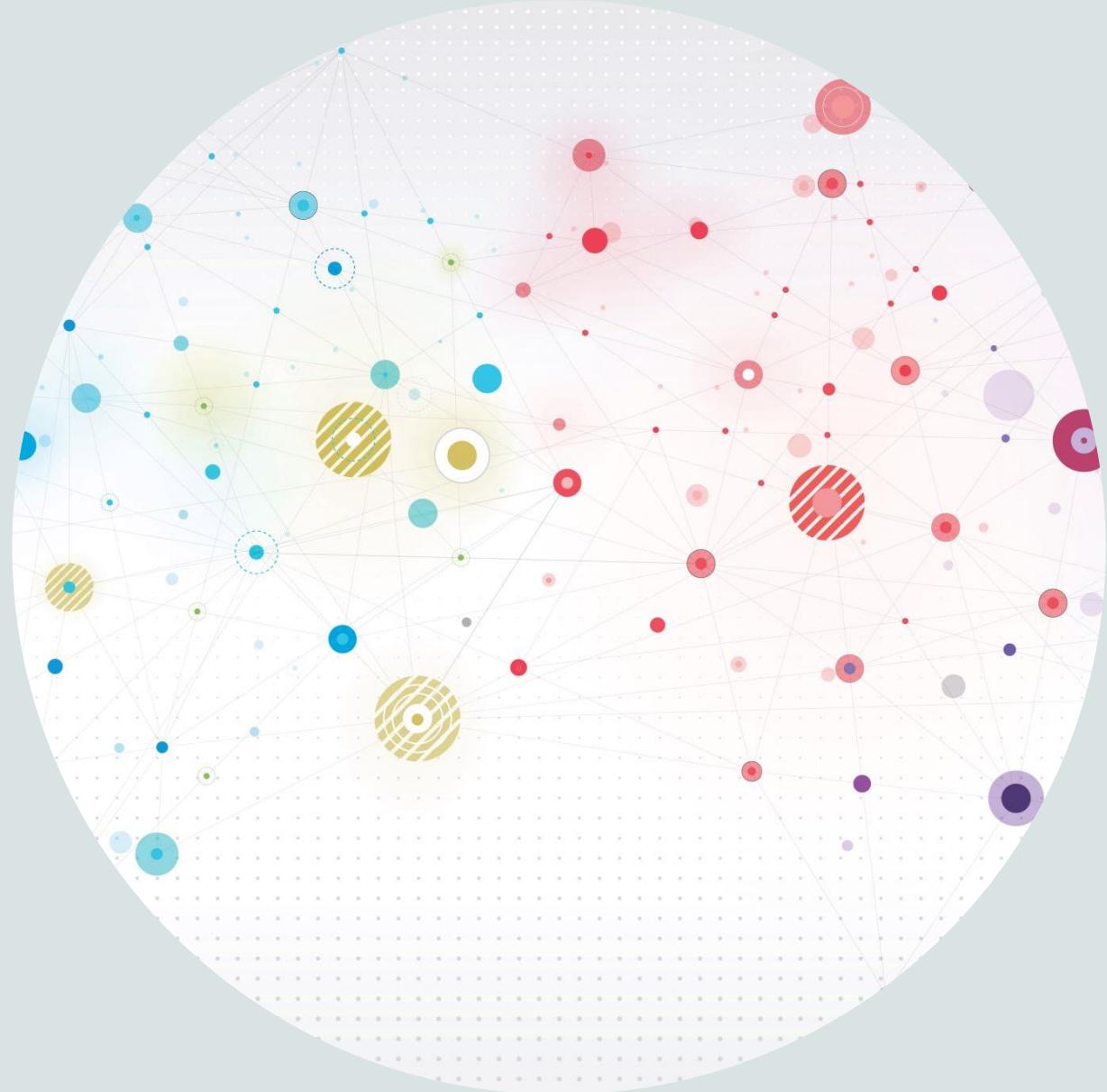


Walmart Sales Project



Agenda

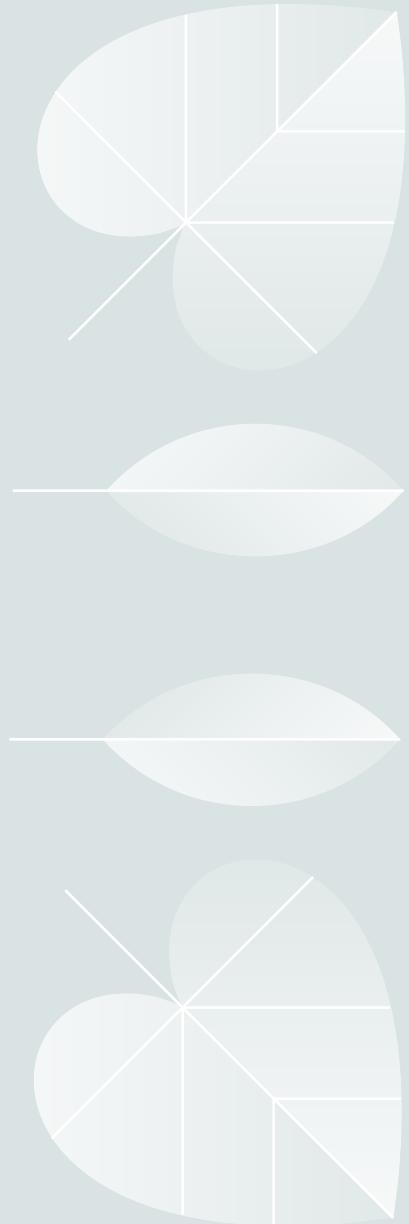
- 1- define objective
- 2- Apply data exploration method
- 3- Apply data cleaning(preprocessing)
- 4- visualization for the data
- 5- Develop and evaluate machine model
- 6- Time Series for sales of walmart

Introduction

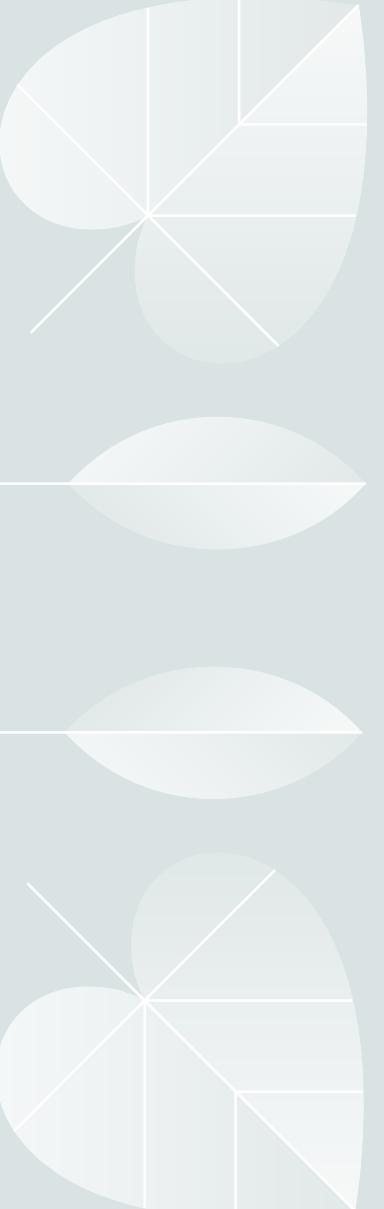
- we will delve into a comprehensive analysis of Walmart's sales dataset. Through rigorous examination and advanced analytics, we aim to provide actionable insights to Walmart's stakeholders, enabling informed decision-making and strategic planning

Objective

- The objective of this project is to analyze the Walmart sales dataset and provide actionable insights to the owners to optimize business strategies.
- Determine which product categories generate the highest sales revenue, allowing Walmart to focus resources and marketing efforts on high-performing categories.
- Understanding customer segments' preferences and behaviors enables targeted marketing and personalized strategies to maximize sales.
- Explore how various factors such as product category, customer segment, region, and discounts influence each other and impact sales and profitability. Understanding these relationships enables Walmart to make informed decisions and optimize business operations.



First : loading data



A screenshot of a Google Colab notebook titled "BI_Project.ipynb". The notebook interface includes a sidebar with file management tools like "Files", "Search", and "Sample Data". The main workspace shows Python code for importing pandas and other libraries, followed by a command to read a CSV file named "Walmart_Dataset.csv". The resulting DataFrame is displayed as a table with columns including Row ID, Order ID, Order Date, Ship Date, Ship Mode, Customer ID, Customer Name, Segment, Country, City, Postal Code, Region, Product ID, Category, Sub-Category, Product Name, Sales, Quantity, and Dis. The first three rows of the dataset are shown in detail.

```
[3]: import pandas as pd
      from sklearn.preprocessing import MinMaxScaler
      import seaborn as sns
      import matplotlib.pyplot as plt
      import numpy as np
      from sklearn.preprocessing import LabelEncoder

[4]: data = pd.read_csv('/content/Walmart_Dataset.csv', encoding='latin-1')
      data.head()
```

Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Dis
0	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2	
1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.9400	3	
2	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.6200	2	

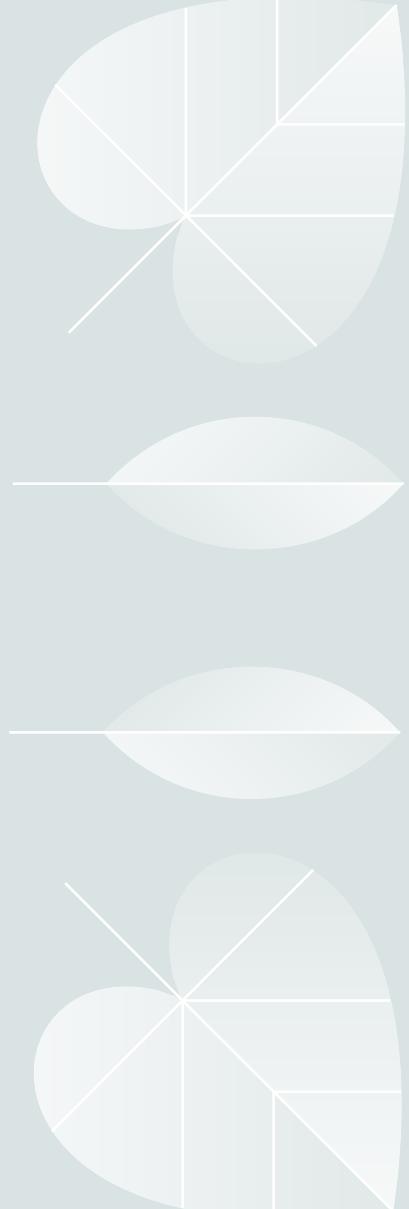
Disk 81.40 GB available

3 4 US-2015-10/11/2015 10/18/2015 Standard SO- Sean Consumer United Fort 33311 South FUR-TA- Furniture Tables Series Slim 957.5775 5

0s completed at 10:59PM

Search ENG US 10:59 PM 4/19/2024

Second :Data Exploration



Google Translate | Basic JavaScript: Understanding... | (10266) amr salih | YouTube | BI_Project.ipynb - Colab

colab.research.google.com/drive/1n-fCWl4A7DHudLvzSS5Dx_3oIgoM0RpC#scrollTo=50KAwK_x869K

FCIS 2024 - Google... Cyber Security codeforces me50/jihadkamili2 Dashboard Login | ASU Smart... ASU - Materials - G... watches ChatGPT Bard Log in to Cisco Additional Resource... FCIS 2025 - Google... IS - Google Drive All Bookmarks

BI_Project.ipynb File Edit View Insert Runtime Tools Help All changes saved

Data Exploration

```
[0] summary_statistics = data.describe()
print(summary_statistics)
```

	Row ID	Postal Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9945.000000	9994.000000
mean	4997.500000	55190.379428	229.858001	3.789574	0.156972	28.656896
std	2885.163629	32063.693358	623.245101	2.225110	0.206668	234.260188
min	1.000000	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	2499.250000	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	4997.500000	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	7495.750000	90088.000000	209.940000	5.000000	0.200000	29.364000
max	9994.000000	99301.000000	22638.480000	14.000000	0.800000	8399.976000

Usable Functions

```
[6] def print_missing_values(df):
    missing_values = df.isnull().sum()
    print(missing_values)

def visualize_box_plot(data):
    numerical_columns = data.select_dtypes(include=['int64', 'float64']).columns
    # Creating box plots for each numerical column
```

0s completed at 10:59PM

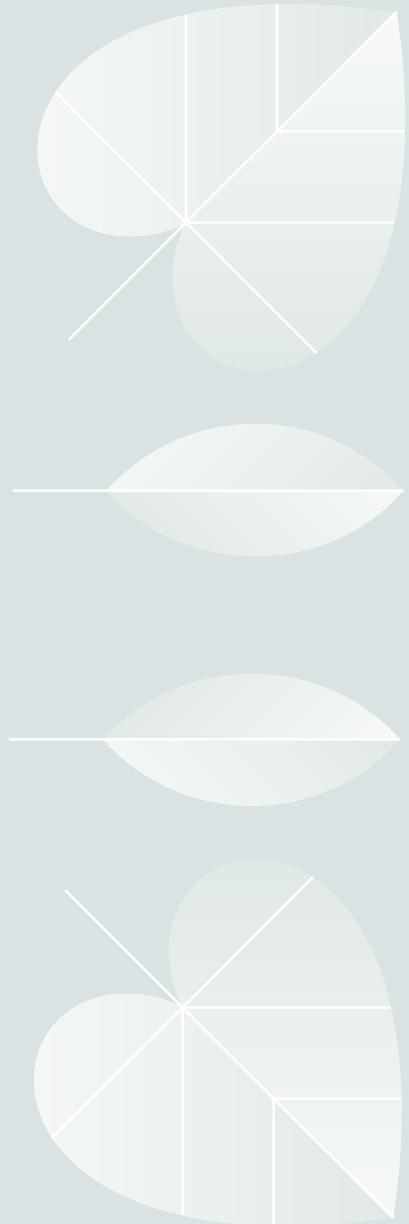
Disk 81.40 GB available

Search, File Explorer, Task View, Start, Control Panel, Mail, Photos, OneDrive, Edge, Google Chrome, ZM, CL, PC, File, Chat, Bard, Log in to Cisco, Additional Resource, FCIS 2025 - Google, IS - Google Drive, All Bookmarks

Data preprocessing

- Steps:

- [1] drop unnecessary column
- [2] Handle missing values (Null)
- [3] Handle duplicate
- [4] outlier handling
- [5] Normalization
- [6] Handling inconsistency



[1] drop unnecessary column:

- Code:

output:



BI_Project.ipynb

Pre-Processing Steps

(1) drop unnecessary columns

```
data.drop(data.columns[0], axis=1, inplace= True)
data.head()
```

	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Dis.
0	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2	
1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.9400	3	
2	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.6200	2	
3	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.5775	5	
4	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	Florida	33311	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold 'N Roll Cart System	22.3680	2	

0s completed at 10:59PM

Google Translate

Basic JavaScript: Understanding

(10286) amr salih

YouTube

Project.ipynb - Colab

FCIS 2024 - Google...

Cyber Security

codeforces

me50/jihadkandil2

Dashboard

Login | ASU Smart...

ASU - Materials - G...

watches

ChatGPT

Bard

Log In to Cisco

Additional Resources

FCIS 2025 - Google...

IS - Google Drive

All Bookmarks

Comment

Share

RAM Disk

Colab AI

Disk 81.40 GB available

Search

11:03 PM 4/19/2024

[2] Handle missing values (Null):

- Code: before handling output

```
print_missing_values(data)

Row ID      0
Order ID     0
Order Date   0
Ship Date    0
Ship Mode    23
Customer ID  0
Customer Name 0
Segment      0
Country      0
City          0
State         0
Postal Code   0
Region        0
Product ID    0
Category      0
Sub-Category   0
Product Name  0
Sales         0
Quantity      0
Discount      49
Profit         0
dtype: int64

[9] mode = data['Ship Mode'].mode().iloc[0]
data['Ship Mode'].fillna(mode, inplace=True)

mean_discount = data['Discount'].mean()
data['Discount'].fillna(mean_discount, inplace=True)

[10] print_missing_values(data)

Order ID      0
Order Date    0
```

- After handling output:

```
Product Name 0
Sales         0
Quantity      0
Discount      49
Profit         0
dtype: int64

[9] mode = data['Ship Mode'].mode().iloc[0]
data['Ship Mode'].fillna(mode, inplace=True)

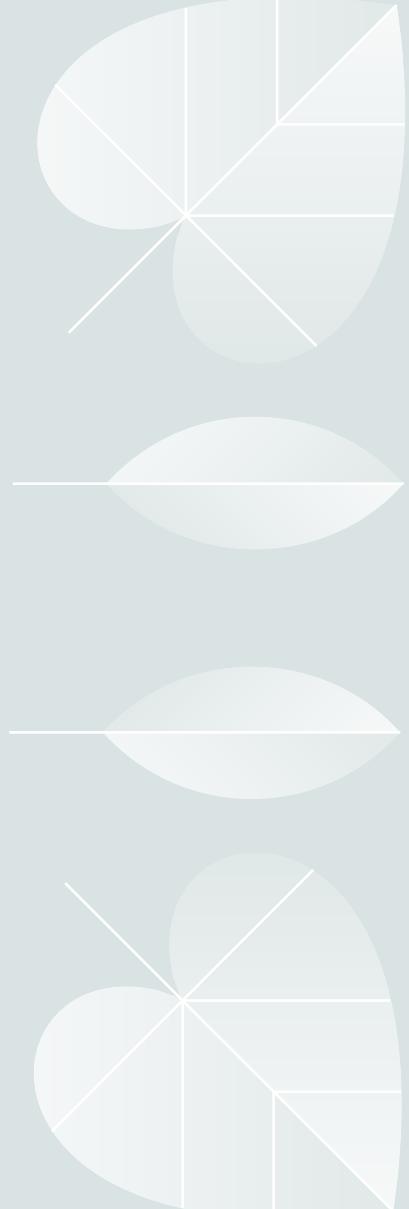
mean_discount = data['Discount'].mean()
data['Discount'].fillna(mean_discount, inplace=True)

[10] print_missing_values(data)

Order ID      0
Order Date    0
Ship Date    0
Ship Mode    0
Customer ID  0
Customer Name 0
Segment      0
Country      0
City          0
State         0
Postal Code   0
Region        0
Product ID    0
Category      0
Sub-Category   0
Product Name  0
Sales         0
Quantity      0
Discount      0
Profit         0
dtype: int64
```

[3] Handle duplicates:

- Code: in this case we found it by checking for the count of rows it decreased :



A screenshot of a Google Colab notebook titled "BI_Project.ipynb". The code cell contains the following Python code:

```
(3) removing duplicate rows
print(data.count())
data.duplicated()
data = data.drop_duplicates()
print(data.count())
```

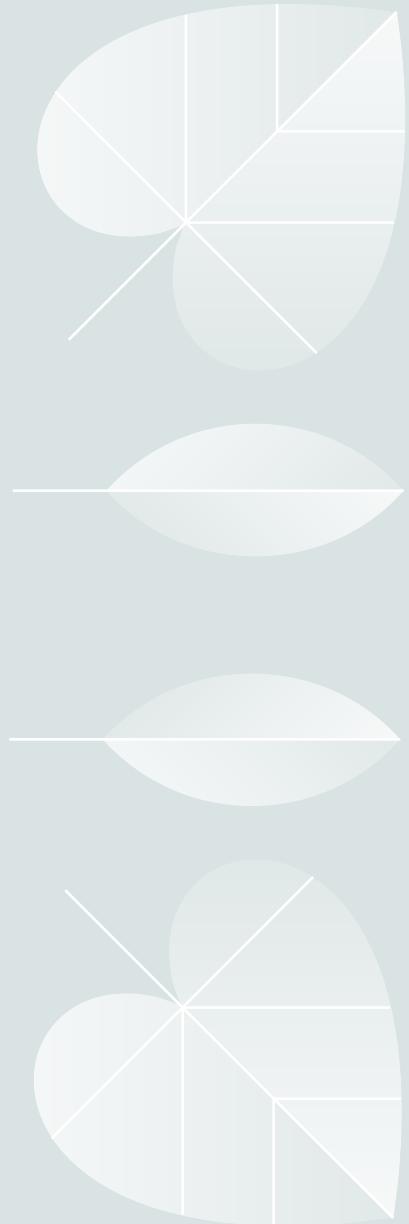
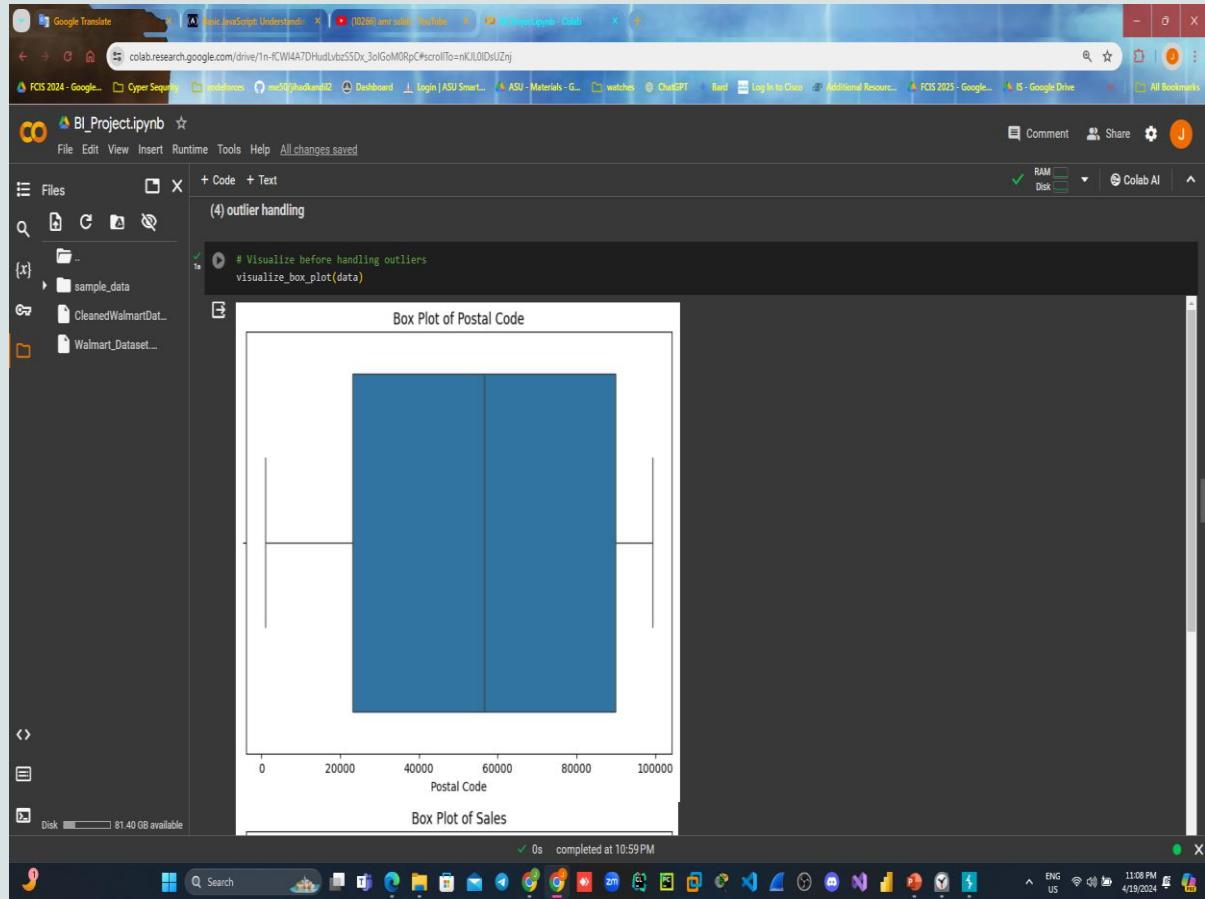
The output of the code shows the count of rows before and after dropping duplicates:

Column	Count Before	Count After
Order ID	9994	9993
Order Date	9994	9993
Ship Date	9994	9993
Ship Mode	9994	9993
Customer ID	9994	9993
Customer Name	9994	9993
Segment	9994	9993
Country	9994	9993
City	9994	9993
State	9994	9993
Postal Code	9994	9993
Region	9994	9993
Product ID	9994	9993
Category	9994	9993
Sub-Category	9994	9993
Product Name	9994	9993
Sales	9994	9993
Quantity	9994	9993
Discount	9994	9993
Profit	9994	9993
dtype: int64		

The notebook interface shows a sidebar with files like "sample_data", "CleanedWalmartDat...", and "Walmart_Dataset....". The status bar at the bottom indicates "0s completed at 10:59 PM".

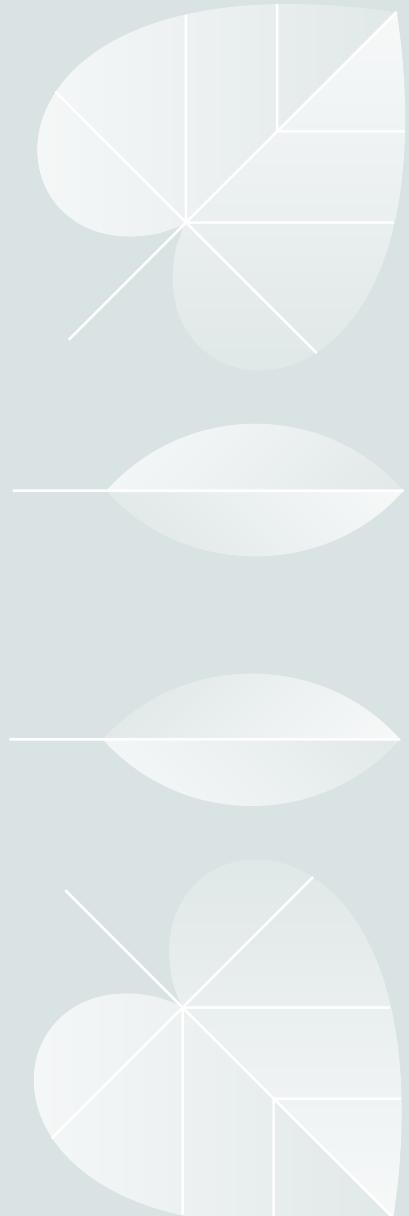
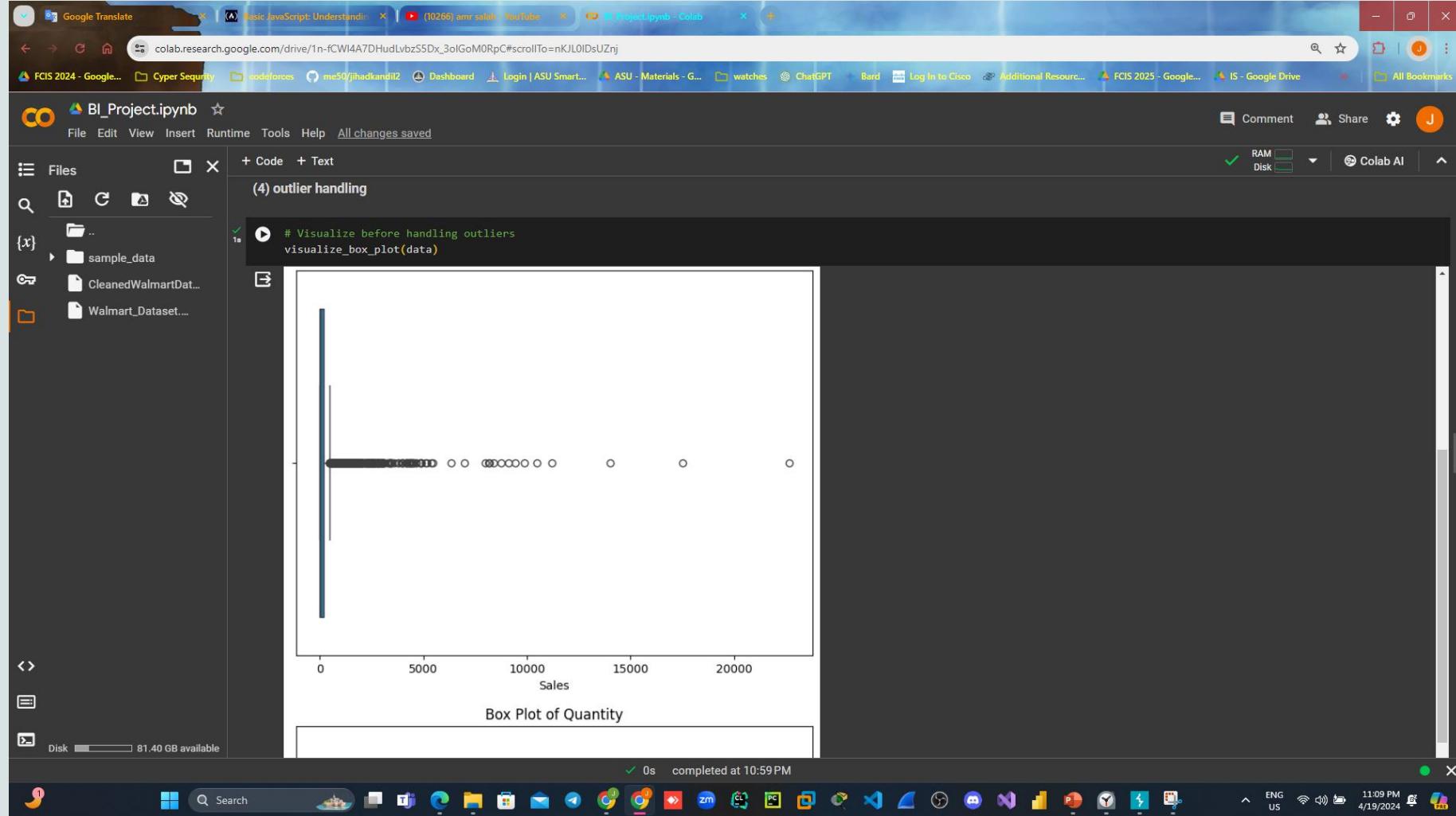
[4] outlier handling :

- Code: by visualizing box plot of each column to see the outlier



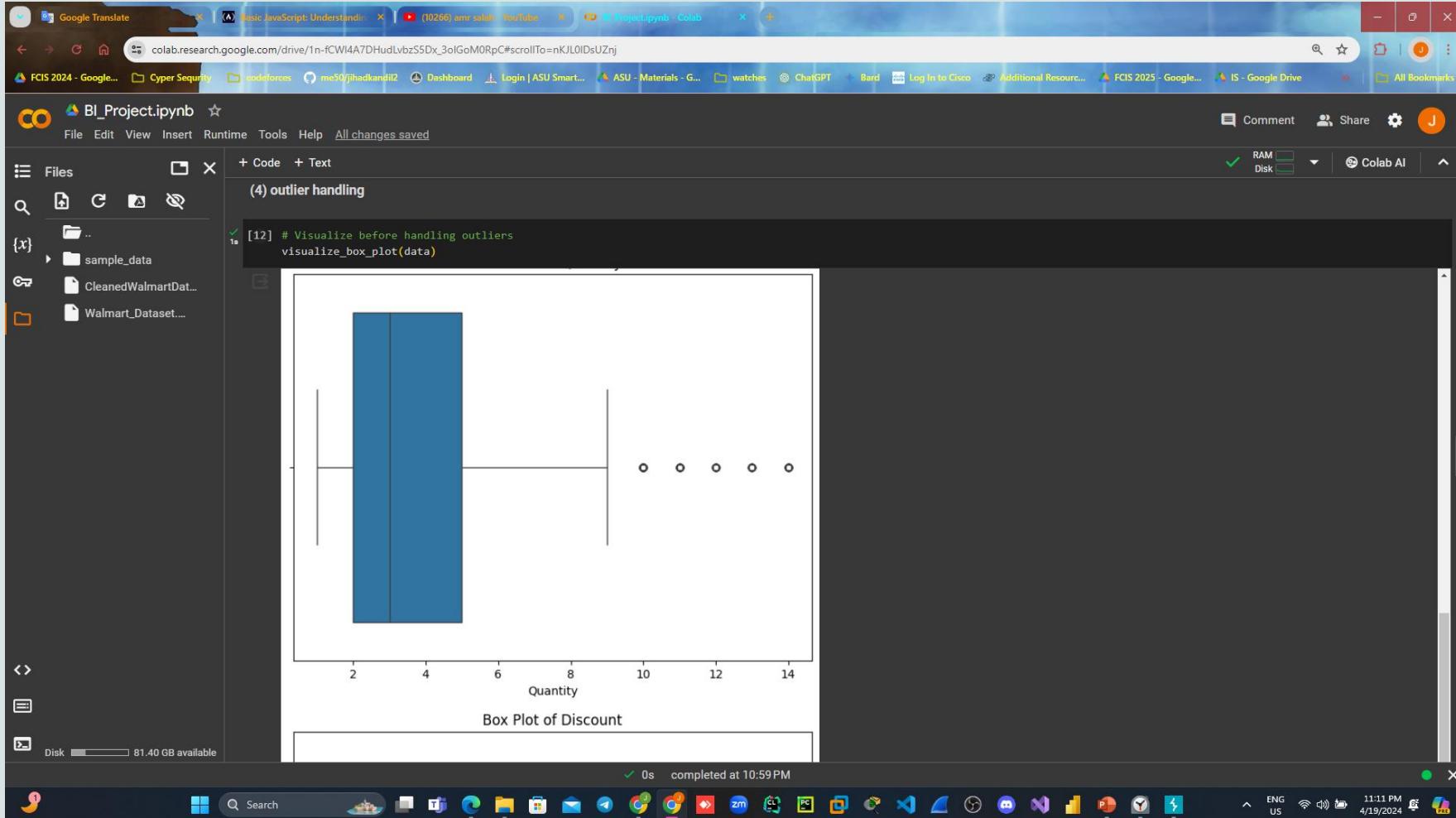
[4] outlier handling continue.....

- Box plot of sales



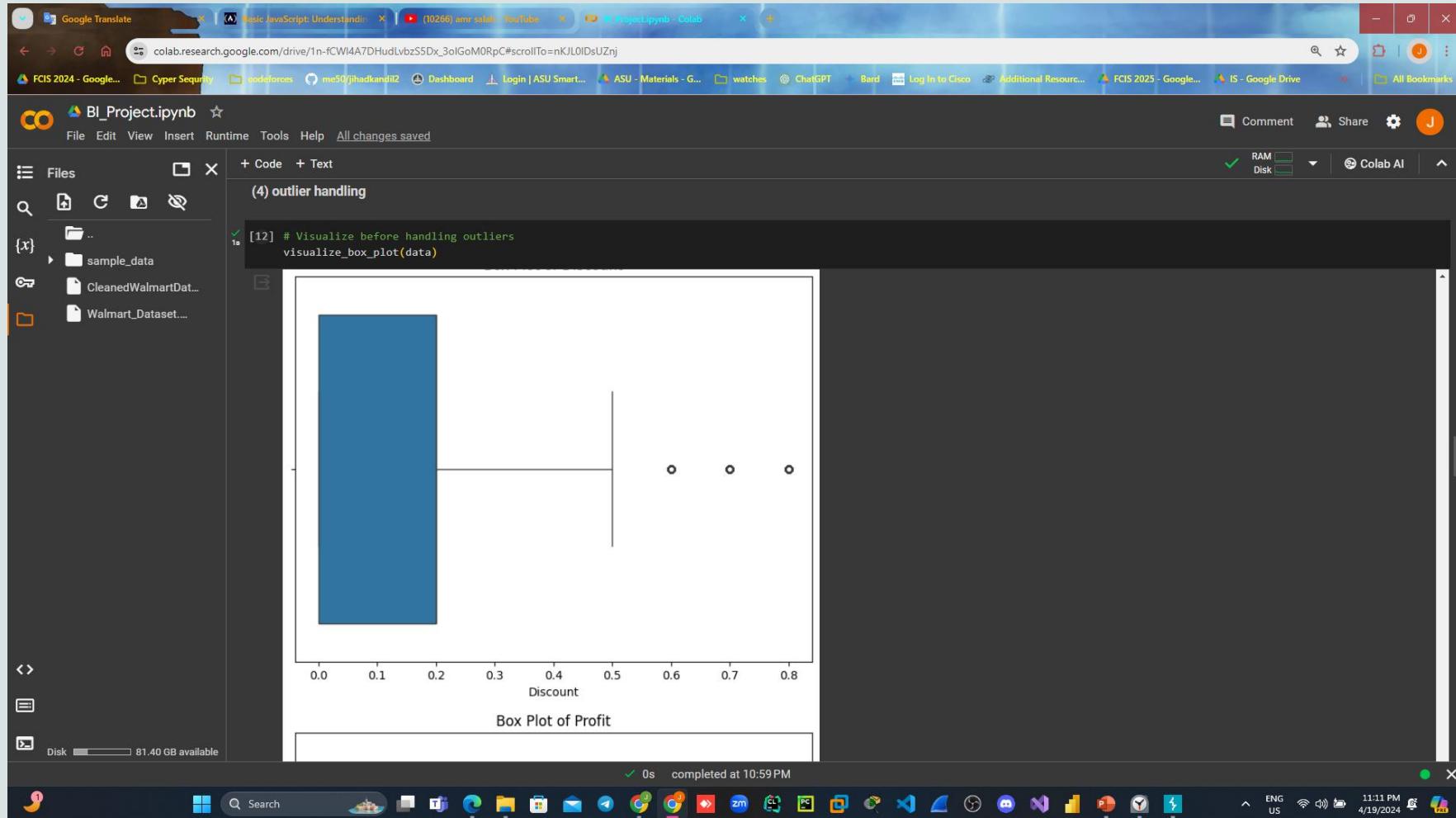
[4] outlier handling continue....

- Box plot of Quantity



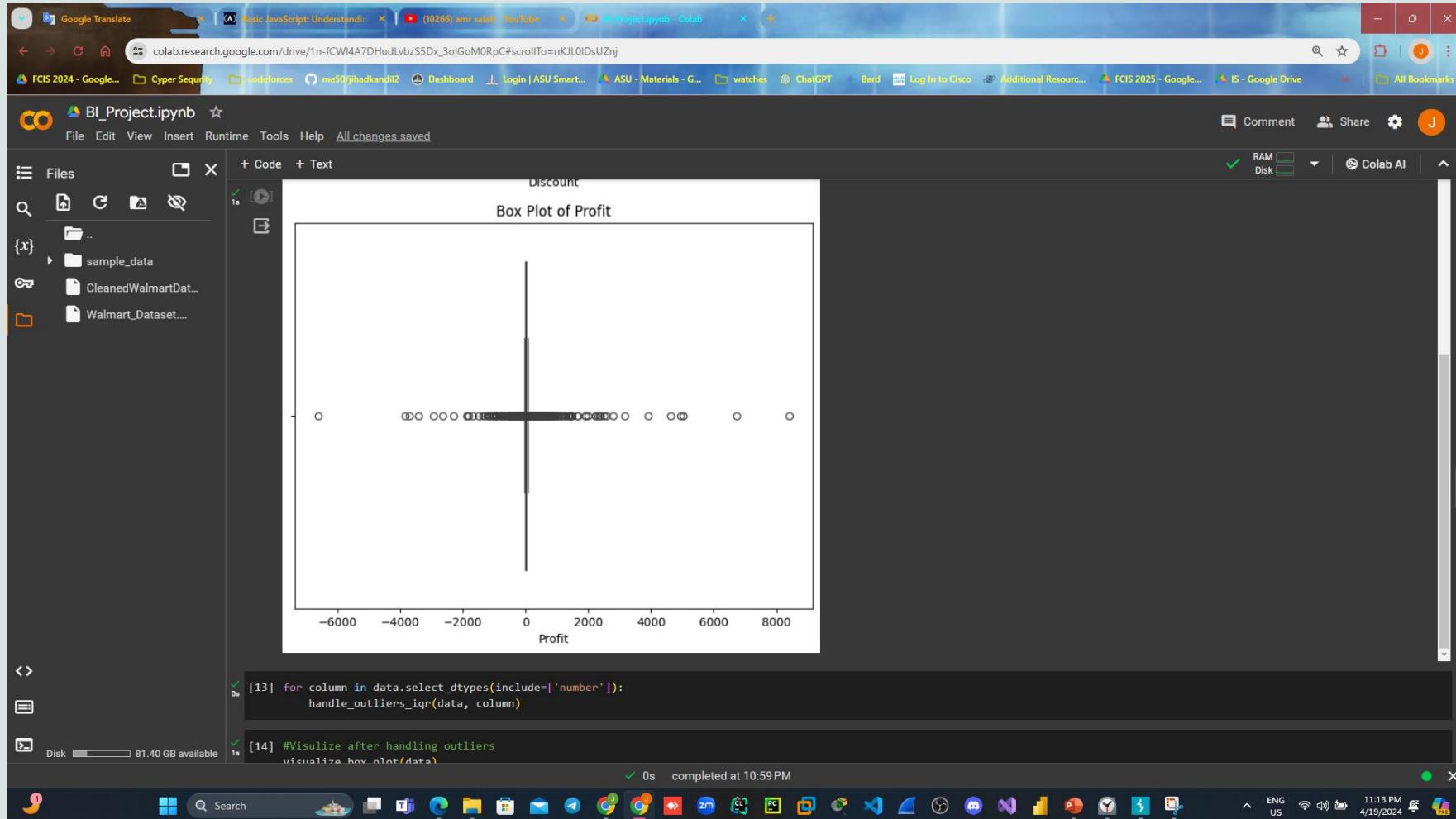
[4] outlier handling continue.....

- Box plot of Discount



[4] outlier handling continue....

- Box plot of Profit



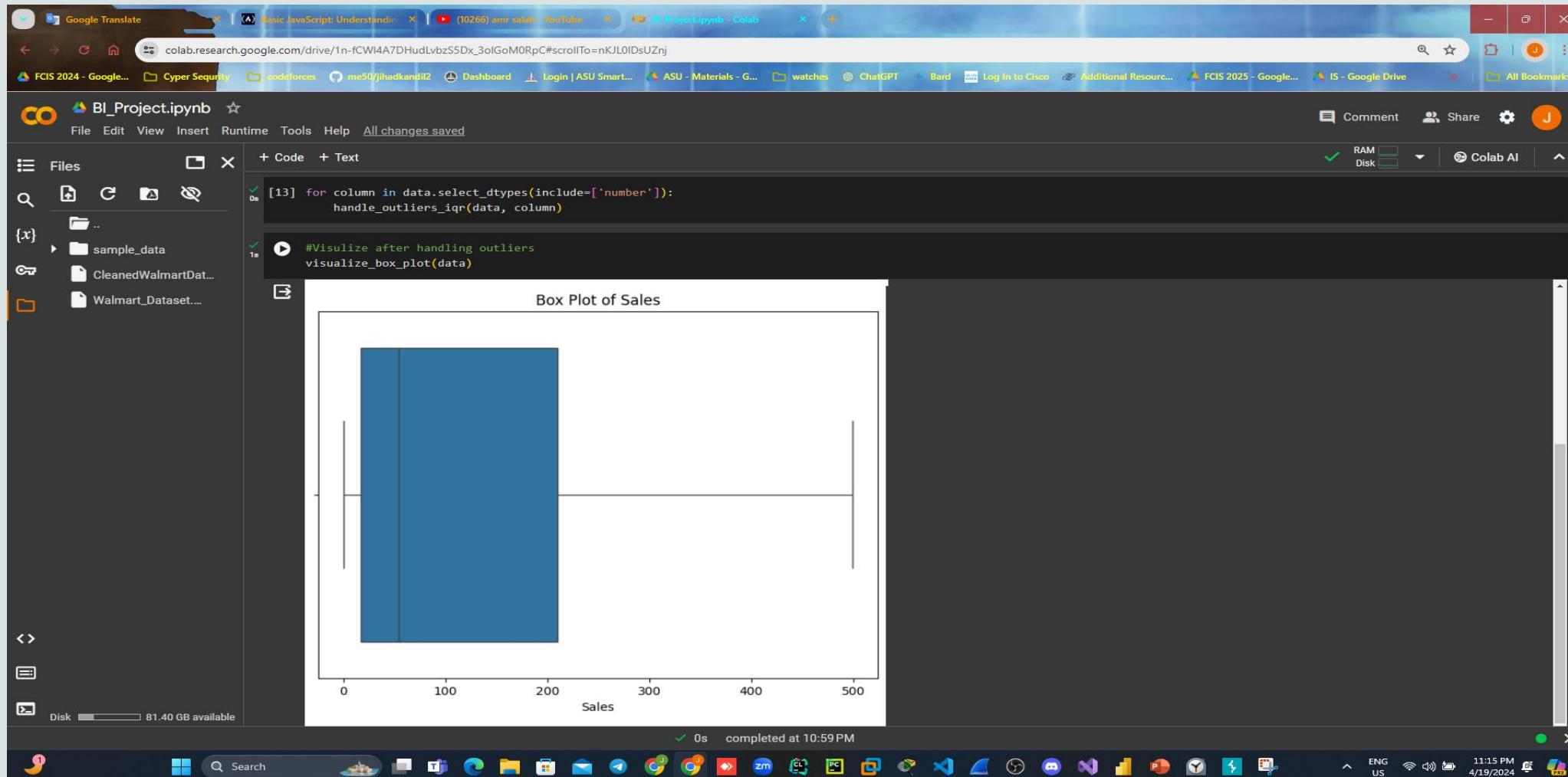
[4] outlier handling continue.....

- After code of handling : Box plot of postal code:



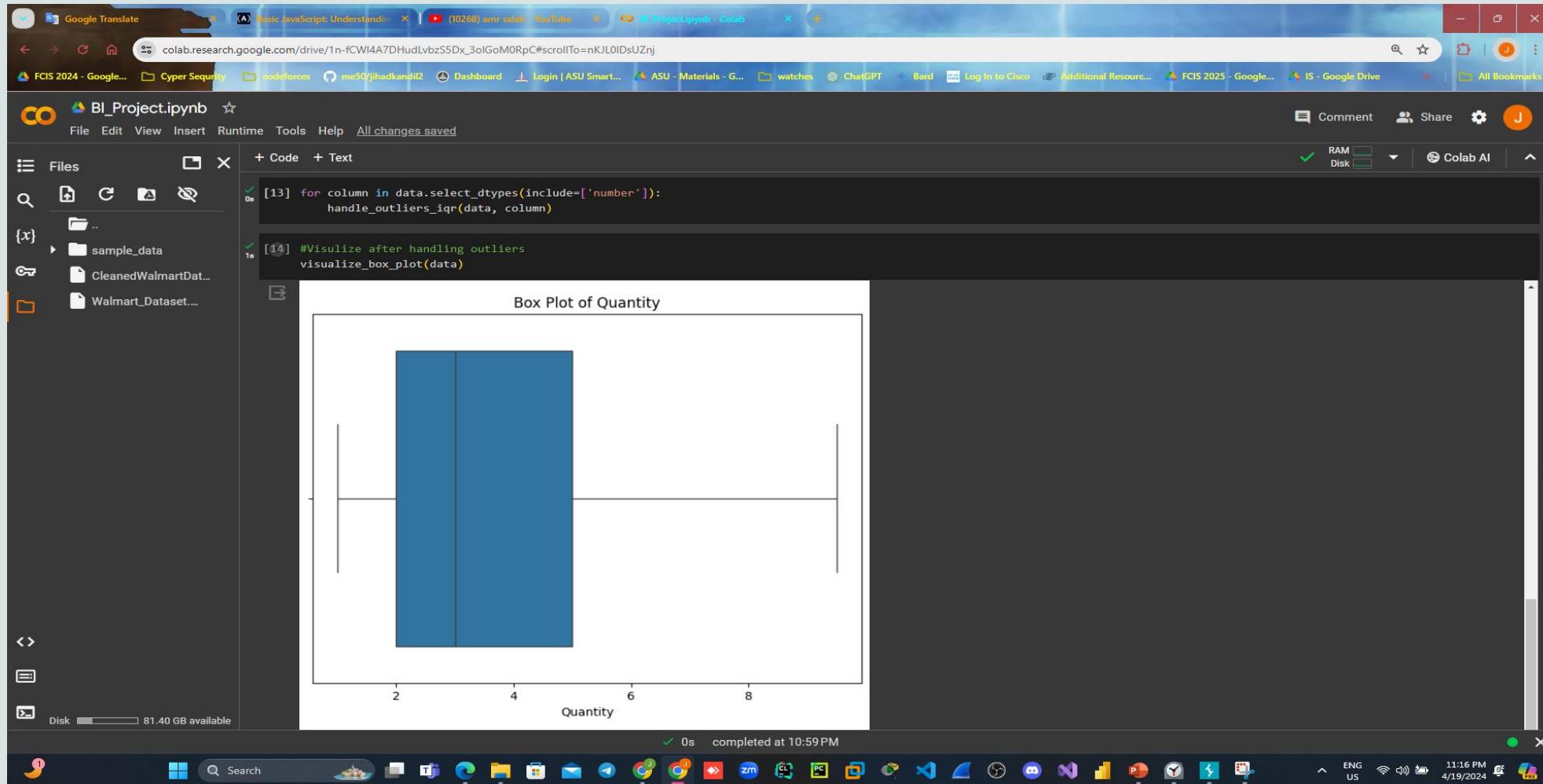
[4] outlier handling continue.....

- Box plot of Sales:



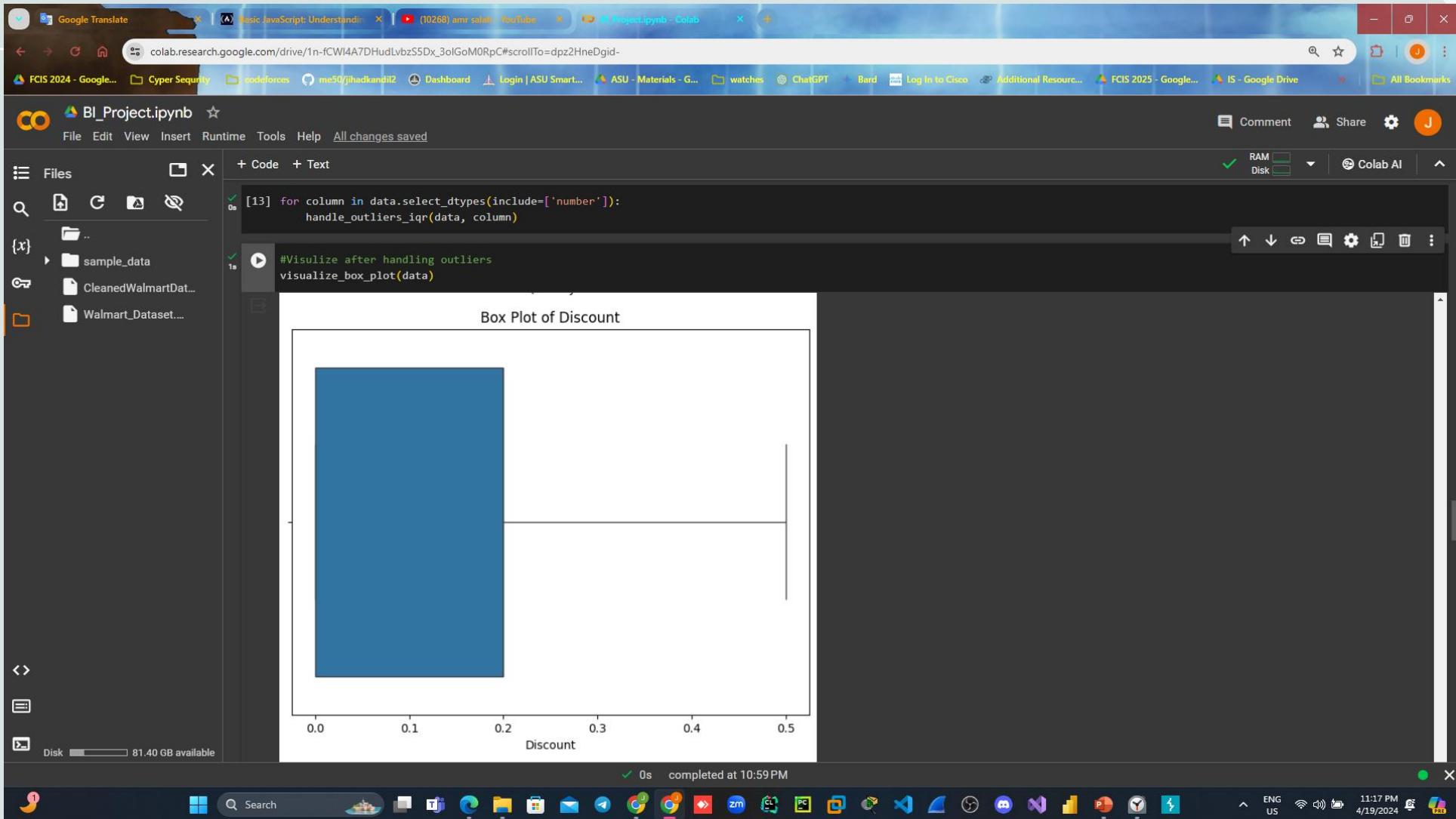
[4] outlier handling continue.....

- Box plot of Quantity:



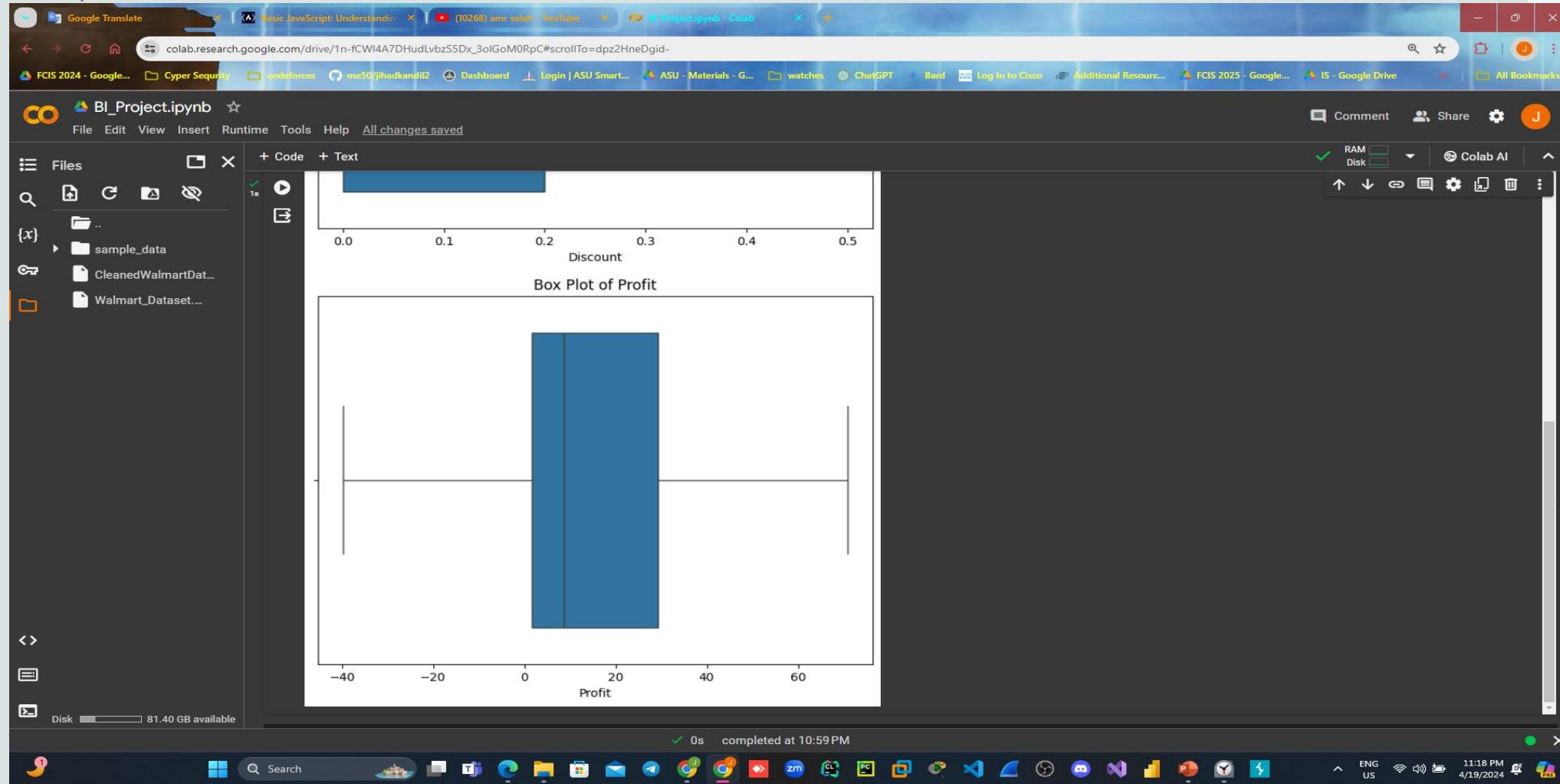
[4] outlier handling continue.....

- Box plot of Discount:



[4] outlier handling continue.....

- Box plot of Profit



[5] Normalization:

- Code:

we normalize data to handle negative value of the data

so as we view box plot of the profit column to make sure :



A screenshot of a Google Colab notebook titled "BI_Project.ipynb". The code cell [15] contains the function "normalize_data(data)" which prints three dataframes. The first dataframe shows order details with columns: Order ID, Order Date, Ship Date, Ship Mode, Customer ID, Customer Name, Segment, Country, City, State, Postal Code, Region, Product ID, Category, Sub-Category, Product Name, Sales, and Quantity. The second and third dataframes show product details with columns: Discount and Profit. The code cell [16] contains the command "#Visualize after handling outliers and normalization". The status bar at the bottom indicates "Os completed at 10:59PM".

```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[15] normalize_data(data)
    Order ID Order Date Ship Date Ship Mode Customer ID \
0 CA-2016-152156 11/8/2016 11/11/2016 Second Class CG-12520
1 CA-2016-152156 11/8/2016 11/11/2016 Second Class CG-12520
2 CA-2016-138688 6/12/2016 6/16/2016 Second Class DV-13045
3 US-2015-108966 10/11/2015 10/18/2015 Standard Class SO-20335
4 US-2015-108966 10/11/2015 10/18/2015 Standard Class SO-20335

    Customer Name Segment Country City State \
0 Claire Gute Consumer United States Henderson Kentucky
1 Claire Gute Consumer United States Henderson Kentucky
2 Darrin Van Huff Corporate United States Los Angeles California
3 Sean O'Donnell Consumer United States Fort Lauderdale Florida
4 Sean O'Donnell Consumer United States Fort Lauderdale Florida

Postal Code Region Product ID Category Sub-Category \
0 42420.0 South FUR-B0-10001798 Furniture Bookcases
1 42420.0 South FUR-CH-10000454 Furniture Chairs
2 90036.0 West OFF-LA-10000240 Office Supplies Labels
3 33311.0 South FUR-TA-10000577 Furniture Tables
4 33311.0 South OFF-ST-10000760 Office Supplies Storage

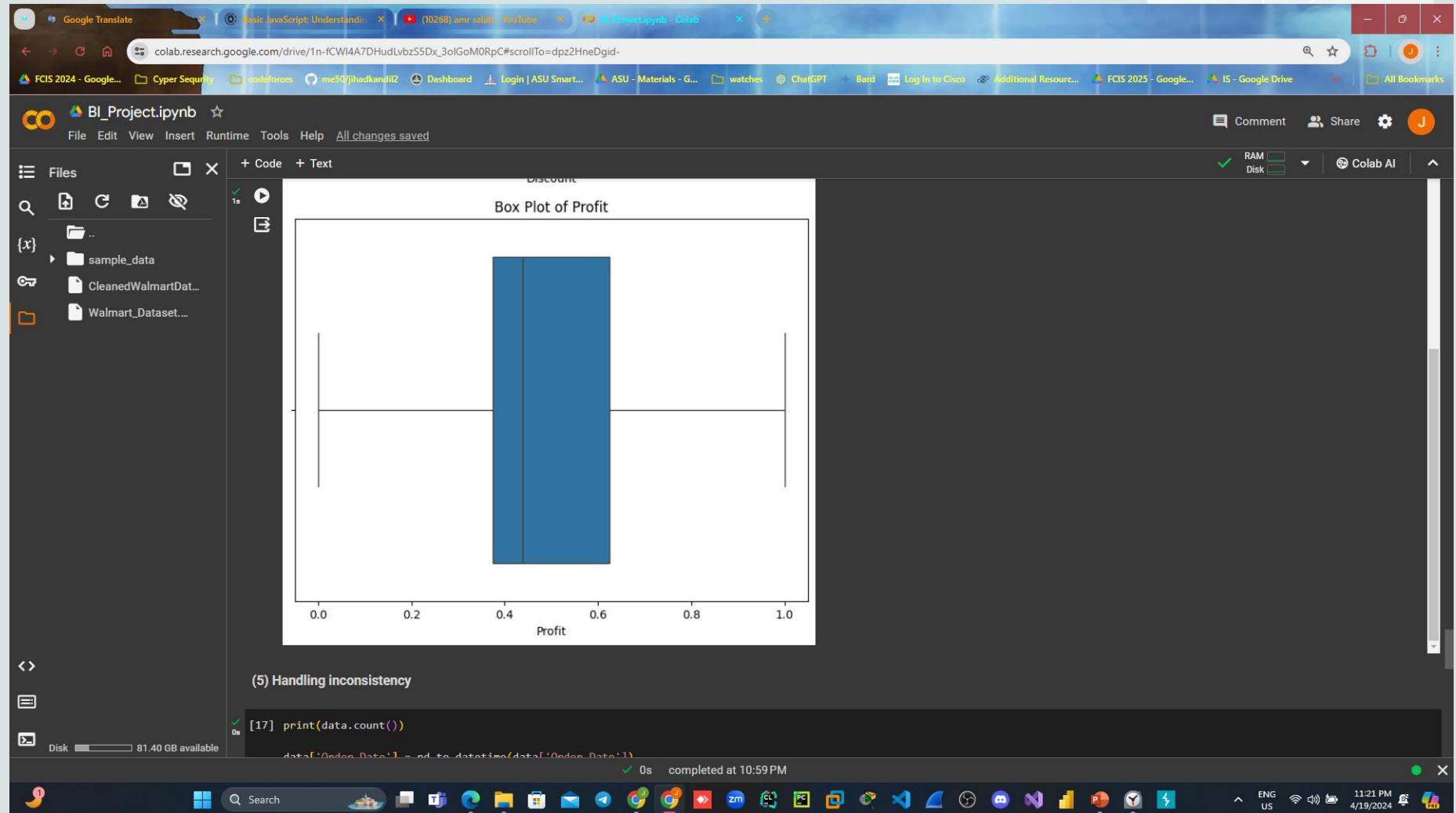
Product Name Sales Quantity \
0 Bush Somerset Collection Bookcase 0.524621 2.0
1 Hon Deluxe Fabric Upholstered Stacking Chairs,... 1.000000 3.0
2 Self-Adhesive Address Labels for Typewriters b... 0.028438 2.0
3 Bretford CR4500 Series Slim Rectangular Table 1.000000 5.0
4 Eldon Fold 'N Roll Cart System 0.043981 2.0

Discount Profit
0 0.00 0.738538
1 0.00 1.000000
2 0.00 0.421506
3 0.45 0.000000
4 0.20 0.382106

[16] #Visualize after handling outliers and normalization
Disk 81.40 GB available Os completed at 10:59PM
To switch input methods, press Windows key + space.
```

[5] Normalization continue.....:

so as we view box plot
of the profit column to
make sure :



[6] Handling inconsistency:

- For the logic of our data

the consistency could found if

The shipping date is before order date

If this found in the data we should drop:

So we check by count we find no conflict

Consistencies.

The screenshot shows a Google Colab notebook titled "BI_Project.ipynb". The code cell contains the following Python script:

```
print(data.count())

data['Order Date'] = pd.to_datetime(data['Order Date'])
data['Ship Date'] = pd.to_datetime(data['Ship Date'])

# Filter out rows where Ship Date is smaller than Order Date
data = data[data['Ship Date'] >= data['Order Date']]

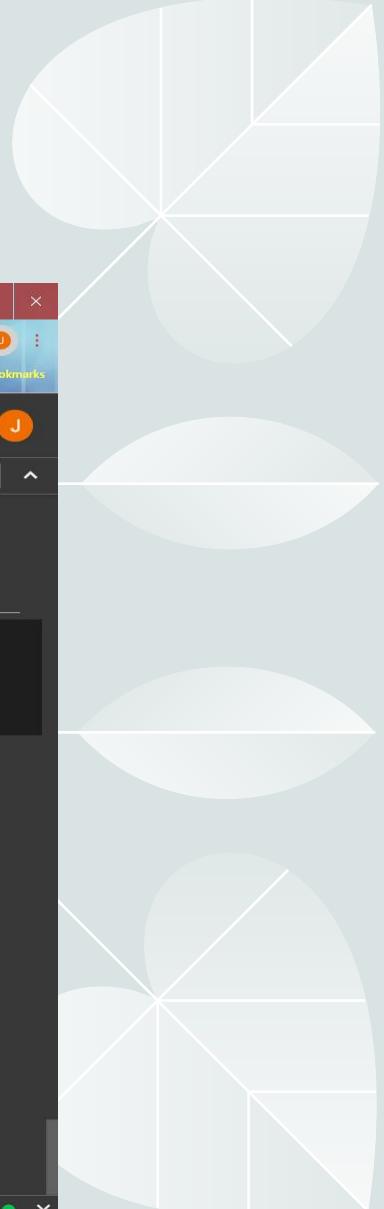
print(data.count())
```

The output of the code is a table showing the count of rows for various columns:

	Count
Order ID	9993
Order Date	9993
Ship Date	9993
Ship Mode	9993
Customer ID	9993
Customer Name	9993
Segment	9993
Country	9993
City	9993
State	9993
Postal Code	9993
Region	9993
Product ID	9993
Category	9993
Sub-Category	9993
Product Name	9993
Sales	9993
Quantity	9993
Discount	9993
Profit	9993
dtype: int64	
Order ID	9993
Order Date	9993
Ship Date	9993
Order Item ID	9993

The status bar at the bottom indicates "0s completed at 10:59 PM".

Exporting data set after cleaning it:



A screenshot of a Google Colab notebook titled "BI_Project.ipynb". The notebook interface shows a sidebar with files like "sample_data", "CleanedWalmartDataset", and "Walmart_Dataset....". The main workspace displays code for data cleaning and export. At the top, a snippet of code shows two rows of data: "Discount" and "Profit", both with the value 9993 and a "dtype: int64". Below this, a section titled "Exporting the dataset after cleaning" contains the following code:

```
[17] Discount      9993
[17] Profit        9993
[17] dtype: int64

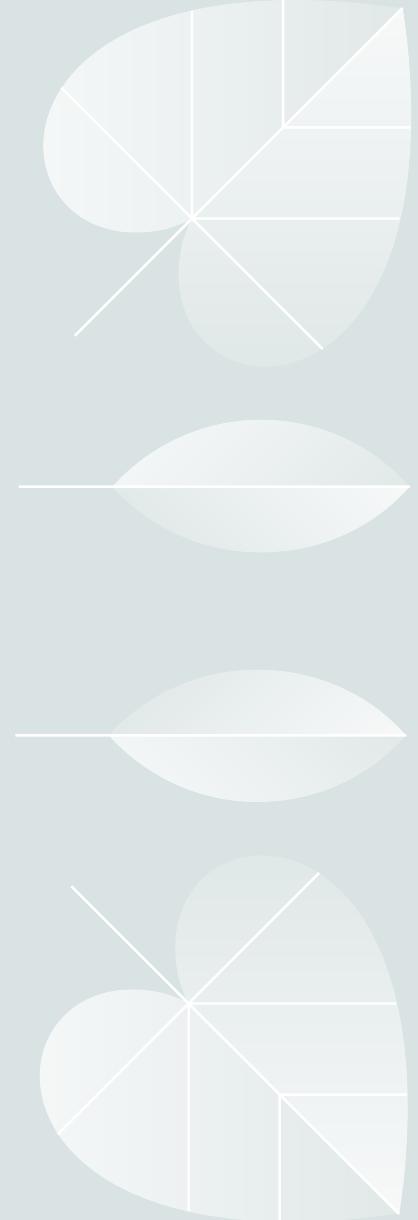
[18] # Define the file path for the CSV file
[18] output_file_path = '/content/CleanedWalmartDataset.csv'

[18] # Export the DataFrame to a CSV file
[18] data.to_csv(output_file_path, index=False)
```

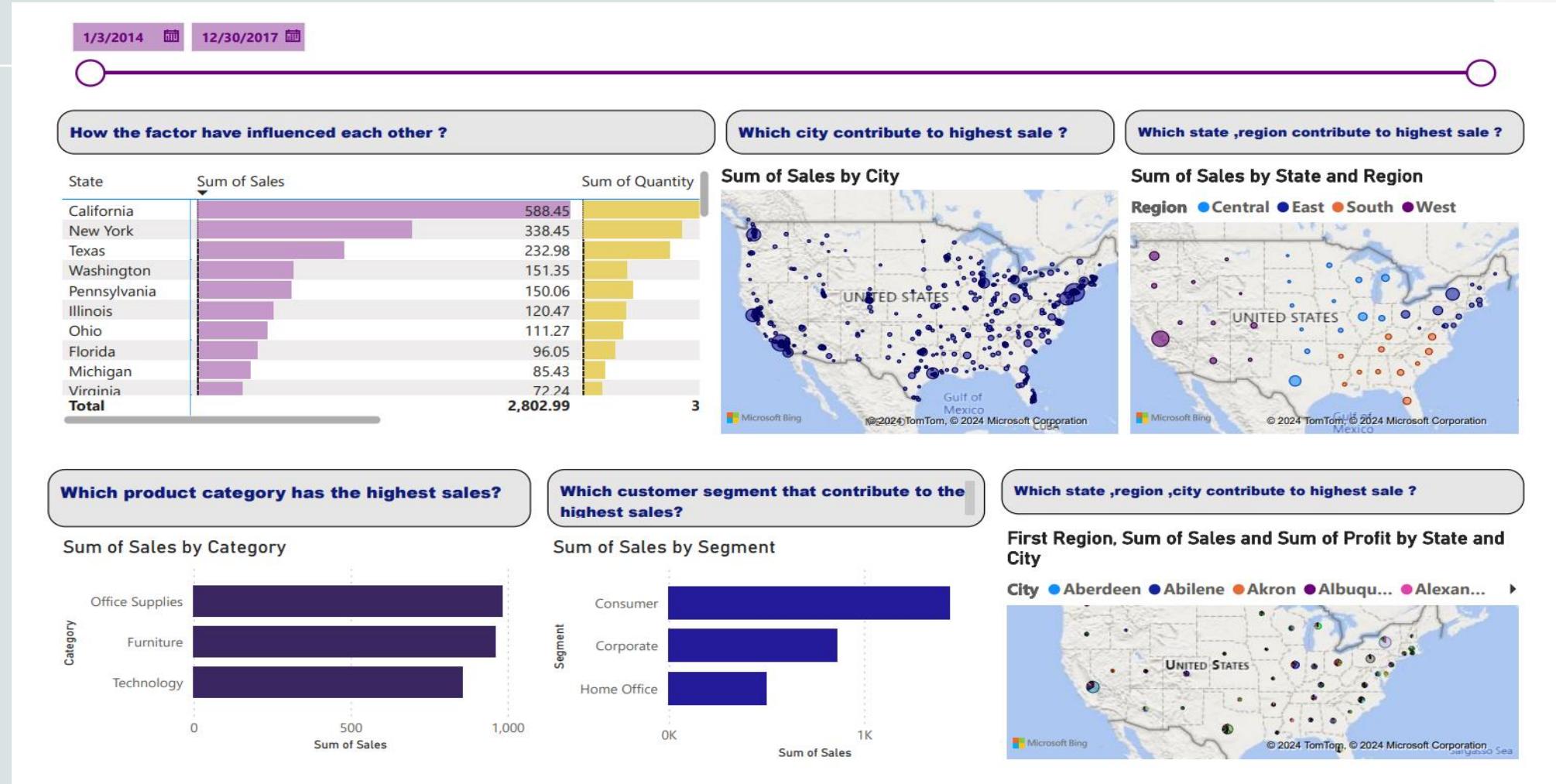
The bottom status bar indicates "Colab paid products - Cancel contracts here" and shows system information including disk usage (81.40 GB available), completion time (0s completed at 10:59 PM), and system details (ENG US, 11:26 PM, 4/19/2024).

Data visualization

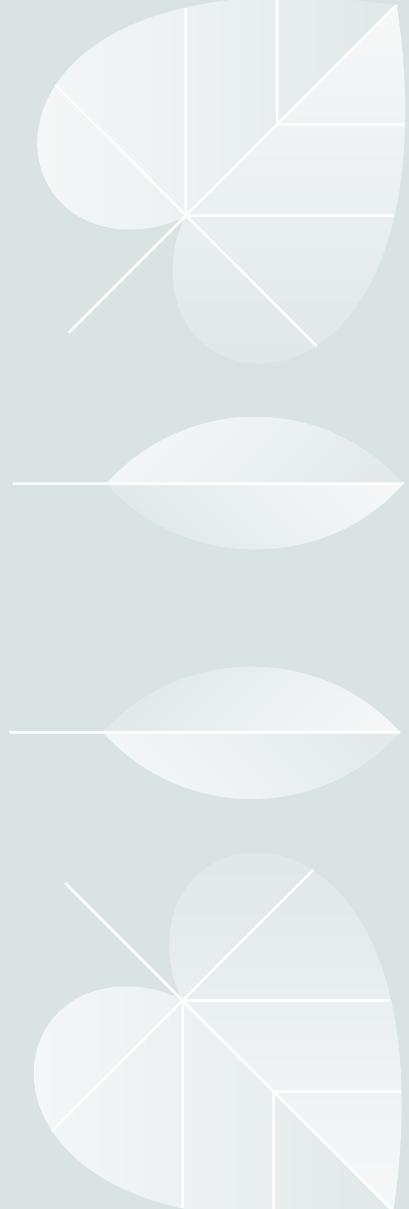
- Lets visualize data to answer:
- ◦ Which product category has the highest sales?
- ◦ Which customer segment that contribute to the highest sales?
- ◦ Which region, state and city contribute to the highest sales and profit?
- ◦ How the factors have influenced on each other?



Data visualization continue...



Model



Lectures - Google Drive Stream Migrated Videos - OneDrive BI_Project/bi version 3.ipynb colab.google bi version 3.ipynb - Colab - +

colab.research.google.com/drive/1UjZ0YNErFsWF0kostSxb5Dc48-wT77s#scrollTo=C4L7uabZKriA

FCIS 2024 - Google... Cyber Security codeforces me50/jhakandil2 Dashboard Login | ASU Smart C... ASU - Materials - G... watches ChatGPT Bard Log In to Cisco Additional Resource... FCIS 2025 - Google... IS - Google Drive All Bookmarks

bi version 3.ipynb ★

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

{x}

```
import pandas as pd

df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])

# Calculate the difference between ship date and order date in days
df['Days_to_Ship'] = (df['Ship Date'] - df['Order Date']).dt.days
```

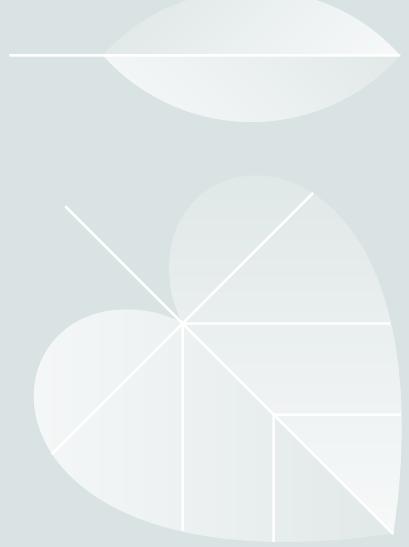
[18] import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

[19] # Select features and target variable
features = df[['Ship Mode', 'Segment', 'Country', 'Region', 'Category', 'Quantity', 'Sub-Category', 'Discount', 'Profit', 'Days_to_Ship']]
target = df['Sales']

[20] # Convert categorical variables into dummy/indicator variables
features = pd.get_dummies(features)

[21] # Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

Model cont..



Lectures - Google Drive | Dream Migrated Videos - OneDrive | BI Project/bi version 3.ipynb | bi version 3.ipynb - Colab | WhatsApp

colab.research.google.com/drive/1UJzOYNExRFsWF0kostSxb5Dc48-wT77s#scrollTo=C4L7uabZKrlA

FCIS 2024 - Google... Cyber Security codewithdave me50/hadkandil2 Dashboard Login | ASU Smart C... ASU - Materials - G... watches ChatGPT Bard Log In to Cisco Additional Resource... FCIS 2025 - Google... IS - Google Drive All Bookmarks

File Edit View Insert Runtime Tools Help All changes saved

Comment Share Colab AI RAM Disk

+ Code + Text

Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

* LinearRegression
LinearRegression()

[23] # Make predictions on the testing set
y_pred = model.predict(X_test)

[24] from sklearn.metrics import r2_score

[25]
Calculate R^2 score
r2 = r2_score(y_test, y_pred)
print("R^2 Score:", r2)

R^2 Score: 0.6376385990127516

Decision Tree Regressor

[26] from sklearn.tree import DecisionTreeRegressor

[27] DTR= DecisionTreeRegressor(max_depth=5)

Model cont..



```
Lectures - Google Drive Stream Migrated Videos One BI_Project/bi version 3.ipynb Colab bi version 3.ipynb WhatsApp FCIS 2024 - Google... Cyber Security codeforces me50/jhadkandil2 Dashboard Login | ASU Smart C... ASU - Materials - G... watches ChatGPT Bard Log In to Cisco Additional Resource... FCIS 2025 - Google... IS - Google Drive All Bookmarks
```

colab.research.google.com/drive/1UJzOYNExRFsWF0kost5xb5Dc48-wT77s#scrollTo=C4L7uabZKrlA

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[28] DTR.fit(X_train, y_train)
* DecisionTreeRegressor
DecisionTreeRegressor(max_depth=5)

[29] y_pred=DTR.predict(X_test)

[30] r2_score(y_test,y_pred)
0.76575504302593

Random Forest Regressor

[31] from sklearn.ensemble import RandomForestRegressor

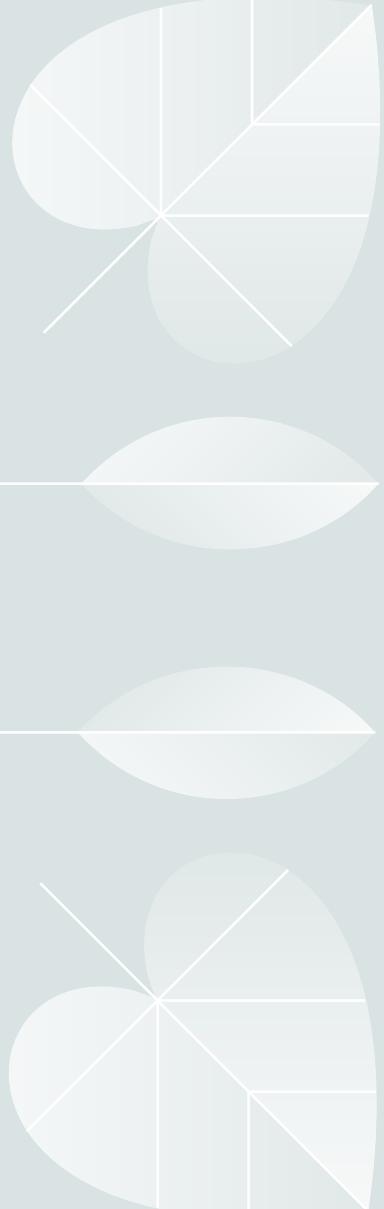
[32] RFR=RandomForestRegressor()

[33] RFR.fit(X_train, y_train)
* RandomForestRegressor
RandomForestRegressor()

[34] y_pred=RFR.predict(X_test)

[35] r2_score(y_test,y_pred)
0.8516910195342183

Time series visualization cont..

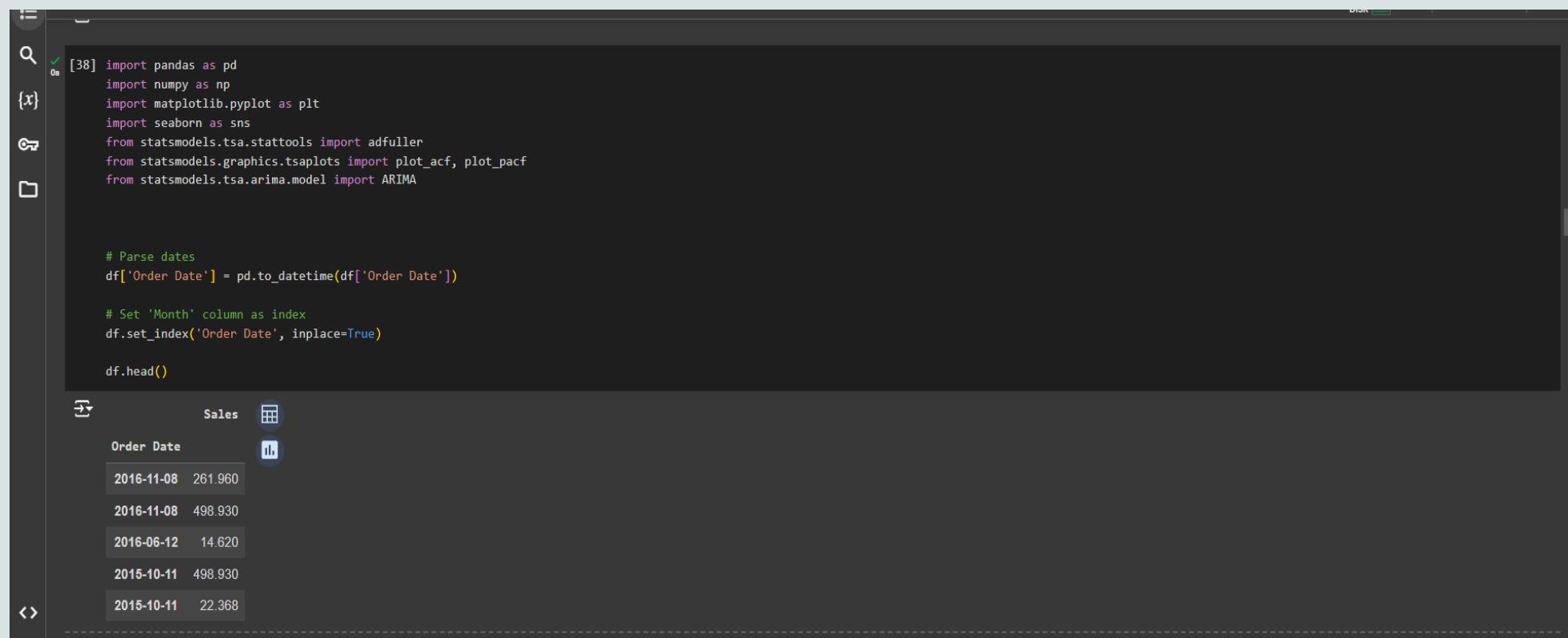


A screenshot of a Google Colab notebook titled "bi version 3.ipynb". The notebook contains Python code for time series analysis. The code includes imports for pandas, numpy, matplotlib.pyplot, seaborn, and various modules from statsmodels. It also includes a section for setting 'order date as index' and selecting specific columns ('Order Date' and 'Sales'). A preview of the resulting DataFrame is shown, displaying the first few rows with columns 'Order Date' and 'Sales'. The code cell [37] shows the output of `df.head()`, which looks like this:

	Order Date	Sales
0	2016-11-08	261.968
1	2016-11-08	498.938
2	2016-06-12	14.620
3	2015-10-11	498.938
4	2015-10-11	22.368

The notebook interface shows other tabs open in the background, including "Lectures - Google Drive", "Stream Migrated Videos - OneDrive", "BL_Project/bi version 3.ipynb", "BI version 3.ipynb - Colab", and "WhatsApp". The status bar at the bottom indicates "All changes saved".

Time series visualization cont..



A screenshot of a Jupyter Notebook interface. The code cell [38] contains Python imports for pandas, numpy, matplotlib.pyplot, seaborn, and various time series analysis modules from statsmodels. It also includes code to parse dates, set the 'Order Date' column as the index, and display the first few rows of the dataset.

```
[38] import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     from statsmodels.tsa.stattools import adfuller
     from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
     from statsmodels.tsa.arima.model import ARIMA

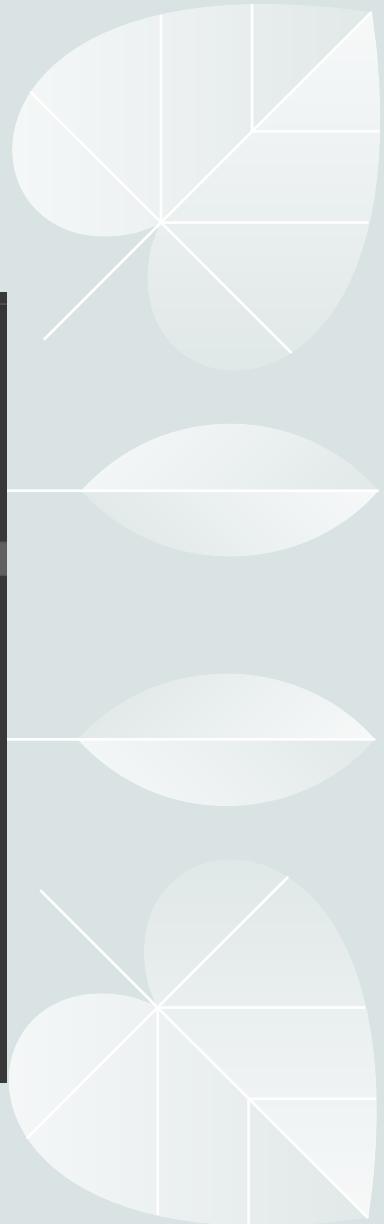
     # Parse dates
     df['Order Date'] = pd.to_datetime(df['Order Date'])

     # Set 'Month' column as index
     df.set_index('Order Date', inplace=True)

     df.head()
```

The data preview shows a table with two columns: 'Order Date' and 'Sales'. The visible rows are:

Order Date	Sales
2016-11-08	261.960
2016-11-08	498.930
2016-06-12	14.620
2015-10-11	498.930
2015-10-11	22.368



Time series visualization cont..

biversion3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

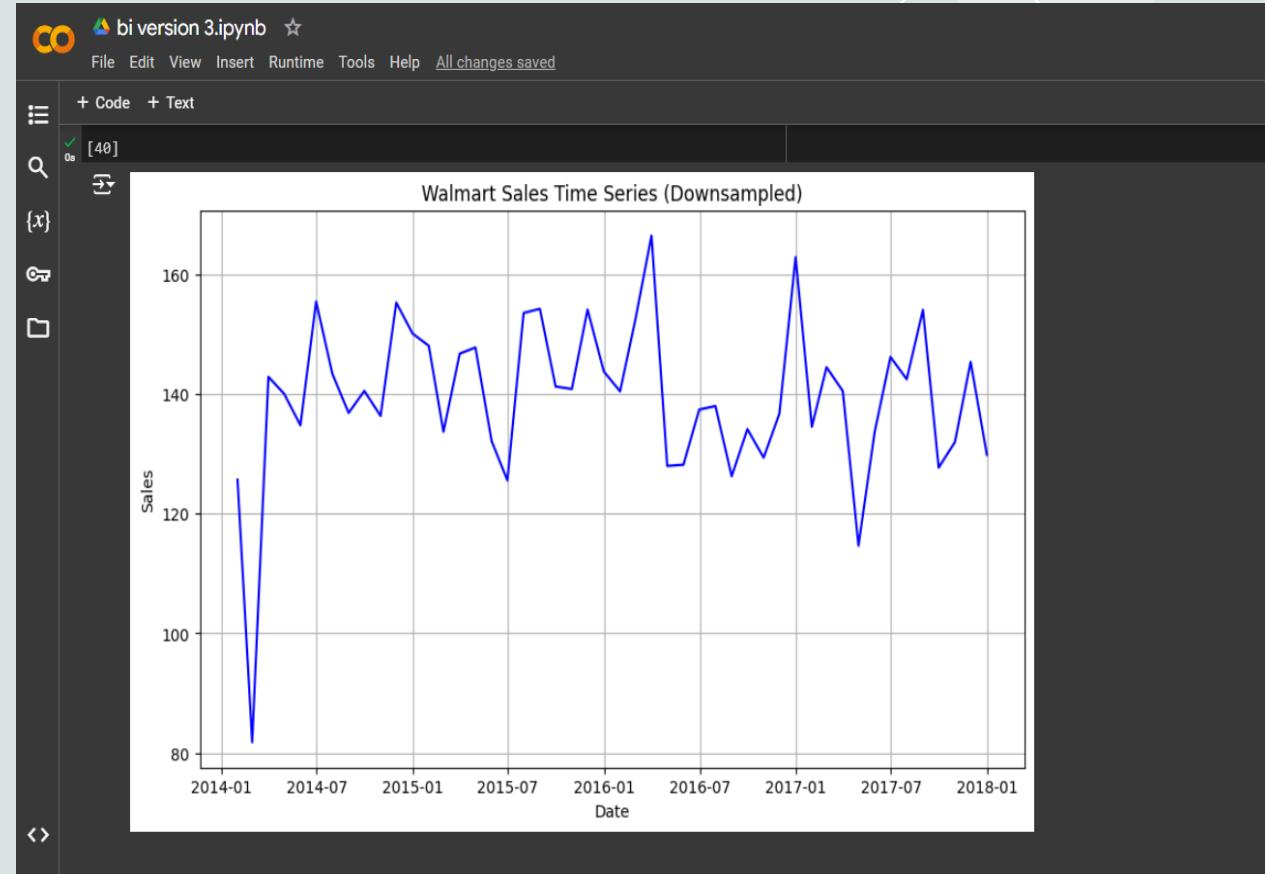
+ Code + Text

[x] [40] # step 1

```
# Create a time series for sales
sales_time_series = df['Sales']

# Downsample the time series to reduce the number of data points
sales_time_series_downsampled = sales_time_series.resample('W').mean().to_frame()
# sales_time_series_downsampled

## Visualize the downsampled time series
plt.figure(figsize=(10, 6))
# index is the x axis , y is the values of the data frame which is the sales
plt.plot(sales_time_series_downsampled.index,sales_time_series_downsampled['Sales'], color='blue')
plt.title('Walmart Sales Time Series (Downsampled)')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.grid(True)
plt.show()
```



Time series visualization cont..

+ Code + Text

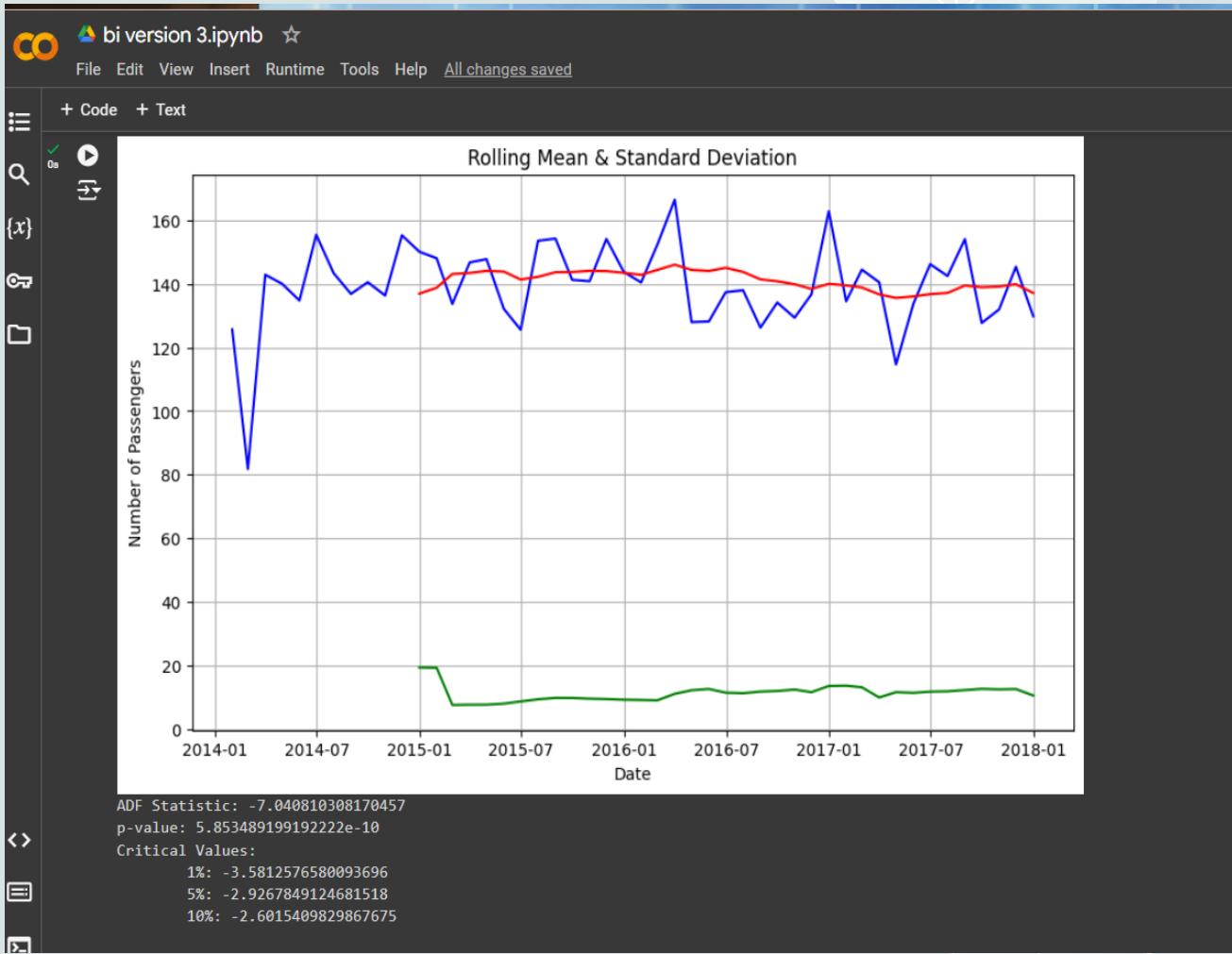
step 2: Stationarize the series

```
[x] 0s [41] # Step 2: Stationarize the series
def stationarize_series(series):
    # Calculate rolling statistics
    rolling_mean = series.rolling(window=12).mean()
    rolling_std = series.rolling(window=12).std()

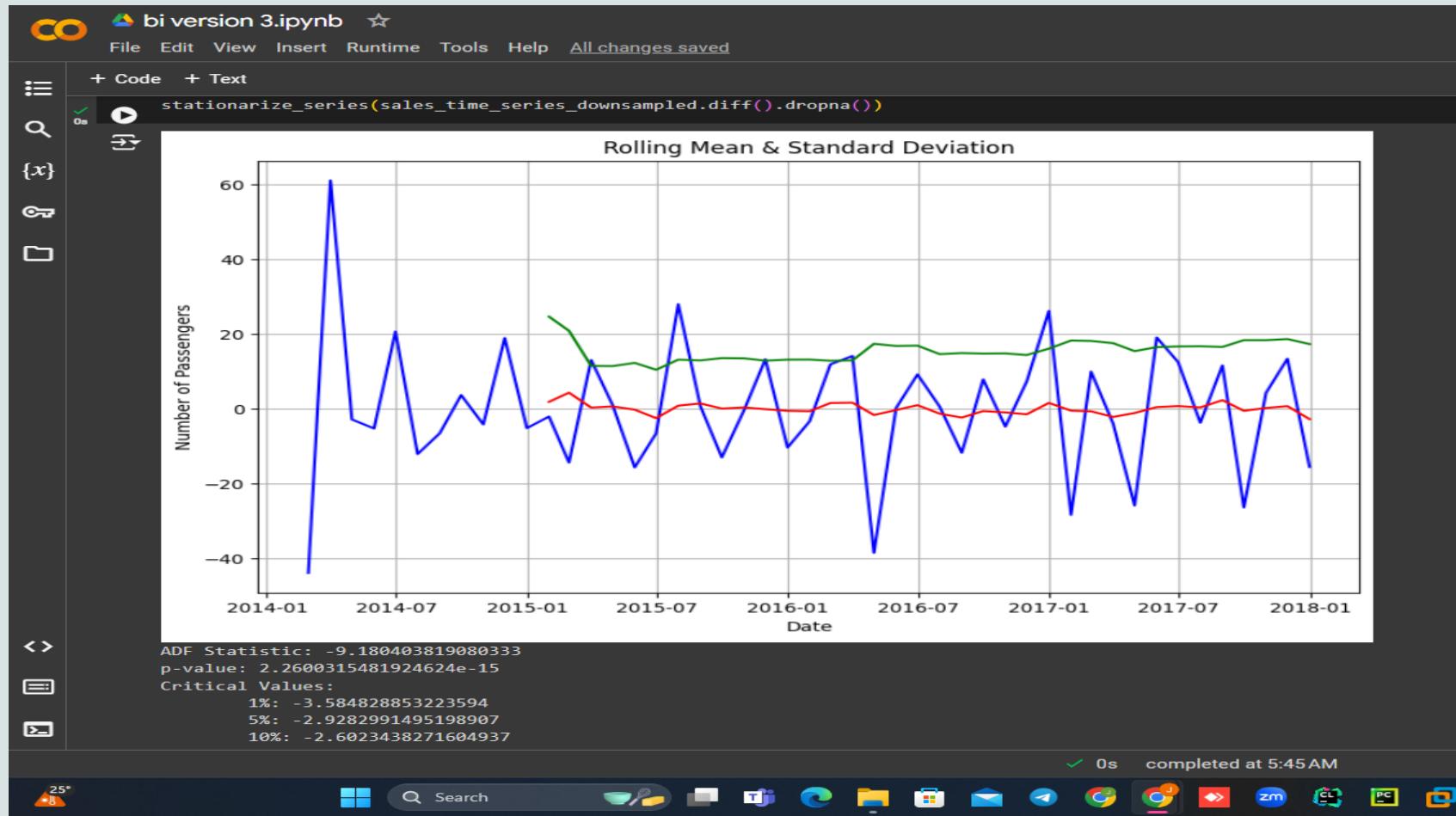
    # Plot rolling statistics
    plt.figure(figsize=(10, 6))
    plt.plot(series, label='Original', color='blue')
    plt.plot(rolling_mean, label='Rolling Mean', color='red')
    plt.plot(rolling_std, label='Rolling Std', color='green')
    plt.title('Rolling Mean & Standard Deviation')
    plt.xlabel('Date')
    plt.ylabel('Number of Passengers')
    # plt.legend(loc='best')
    plt.grid(True)
    plt.show()

    # Perform Dickey-Fuller test
    result = adfuller(series)
    print('ADF Statistic:', result[0])
    print('p-value:', result[1])
    print('Critical Values:')
    for key, value in result[4].items():
        print('\t{}: {}'.format(key, value))

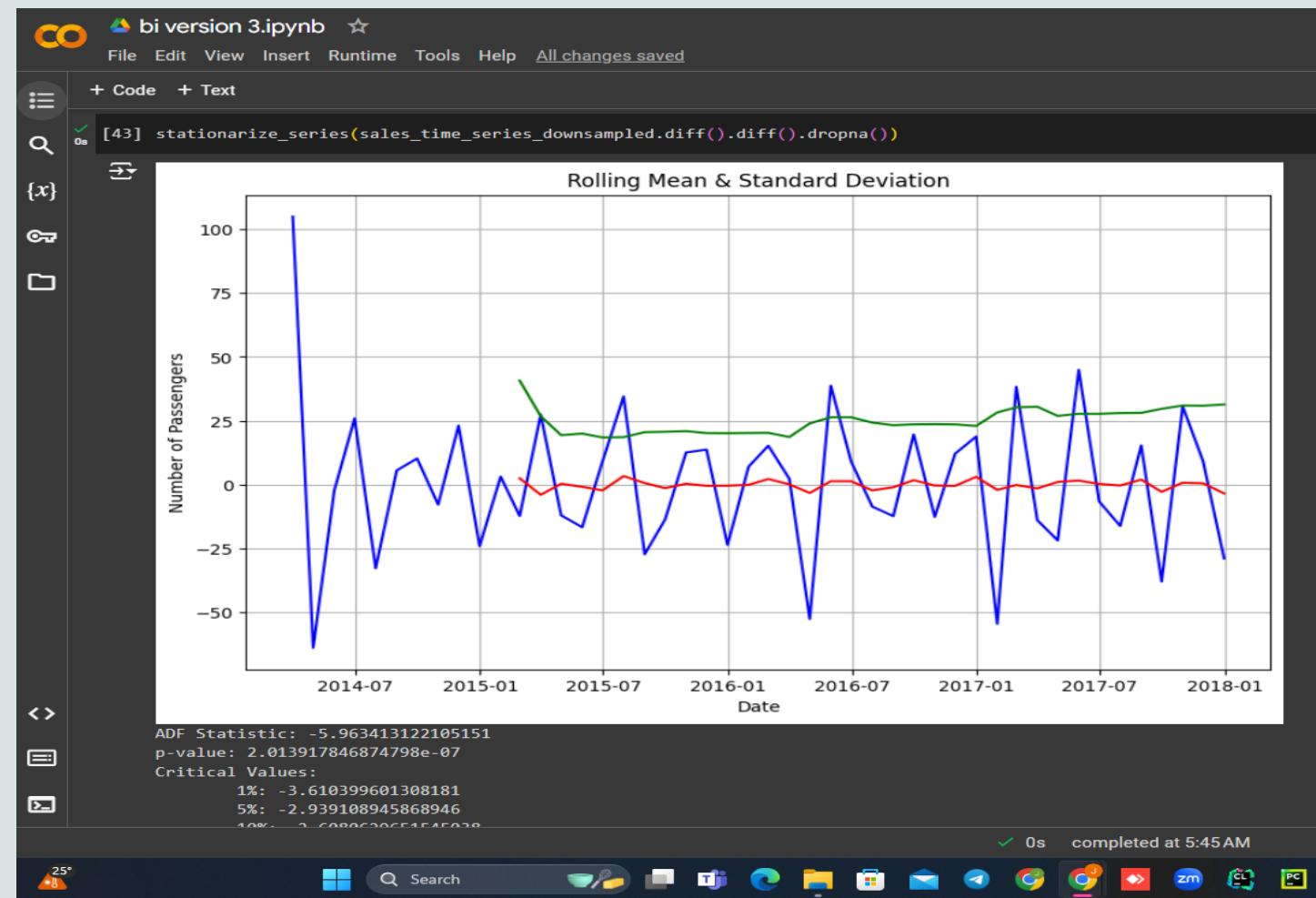
# Apply stationarize_series function
stationarize_series(sales_time_series_downsampled)
```



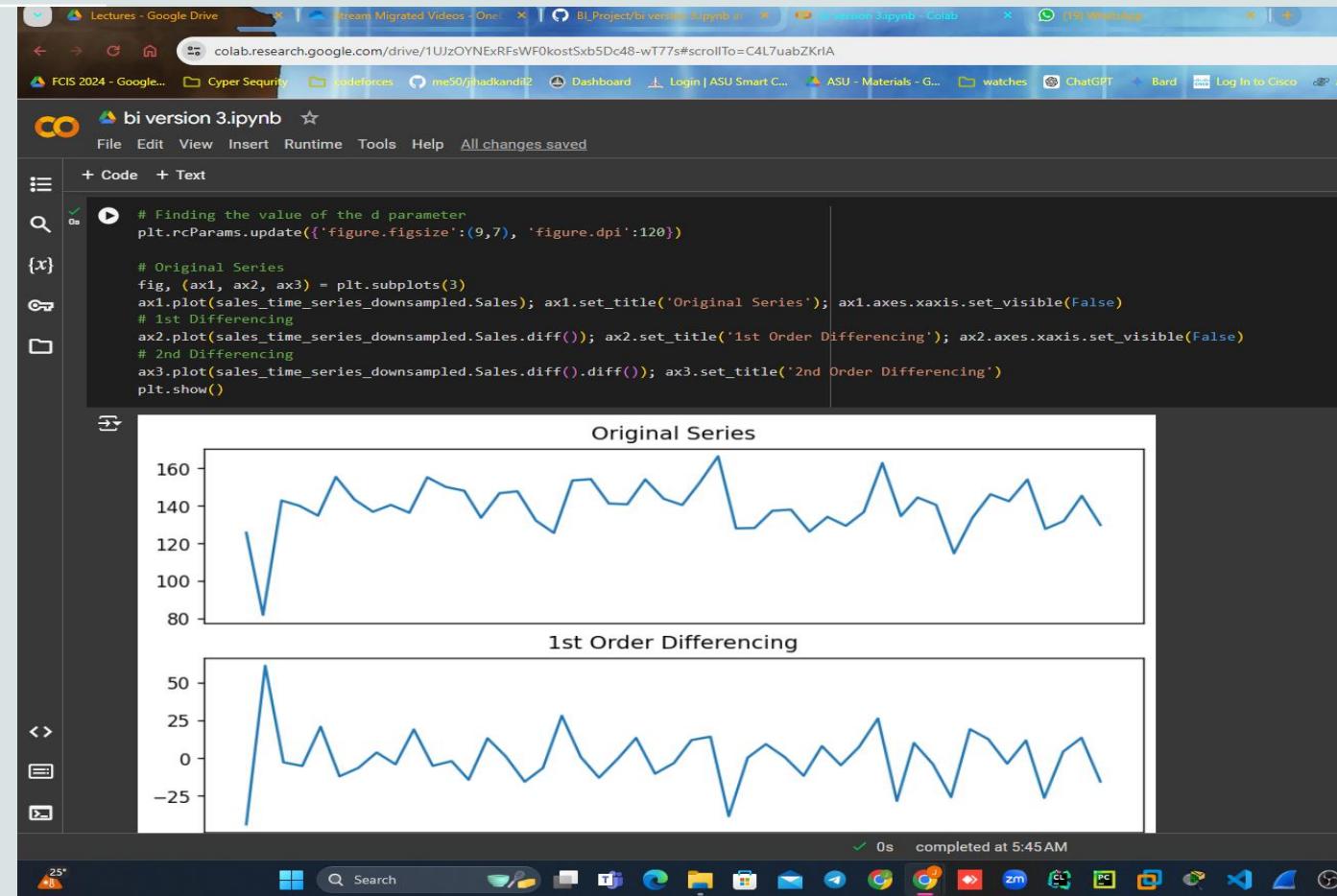
Time series visualization cont..



Time series visualization cont..

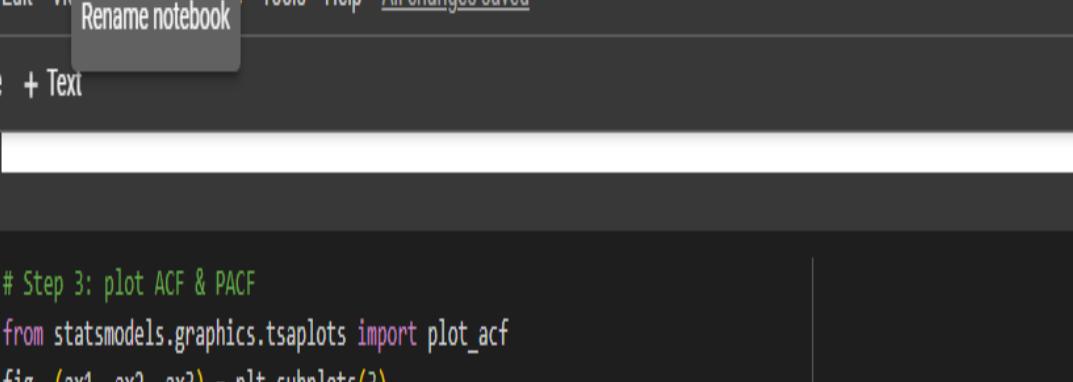


Time series visualization cont..



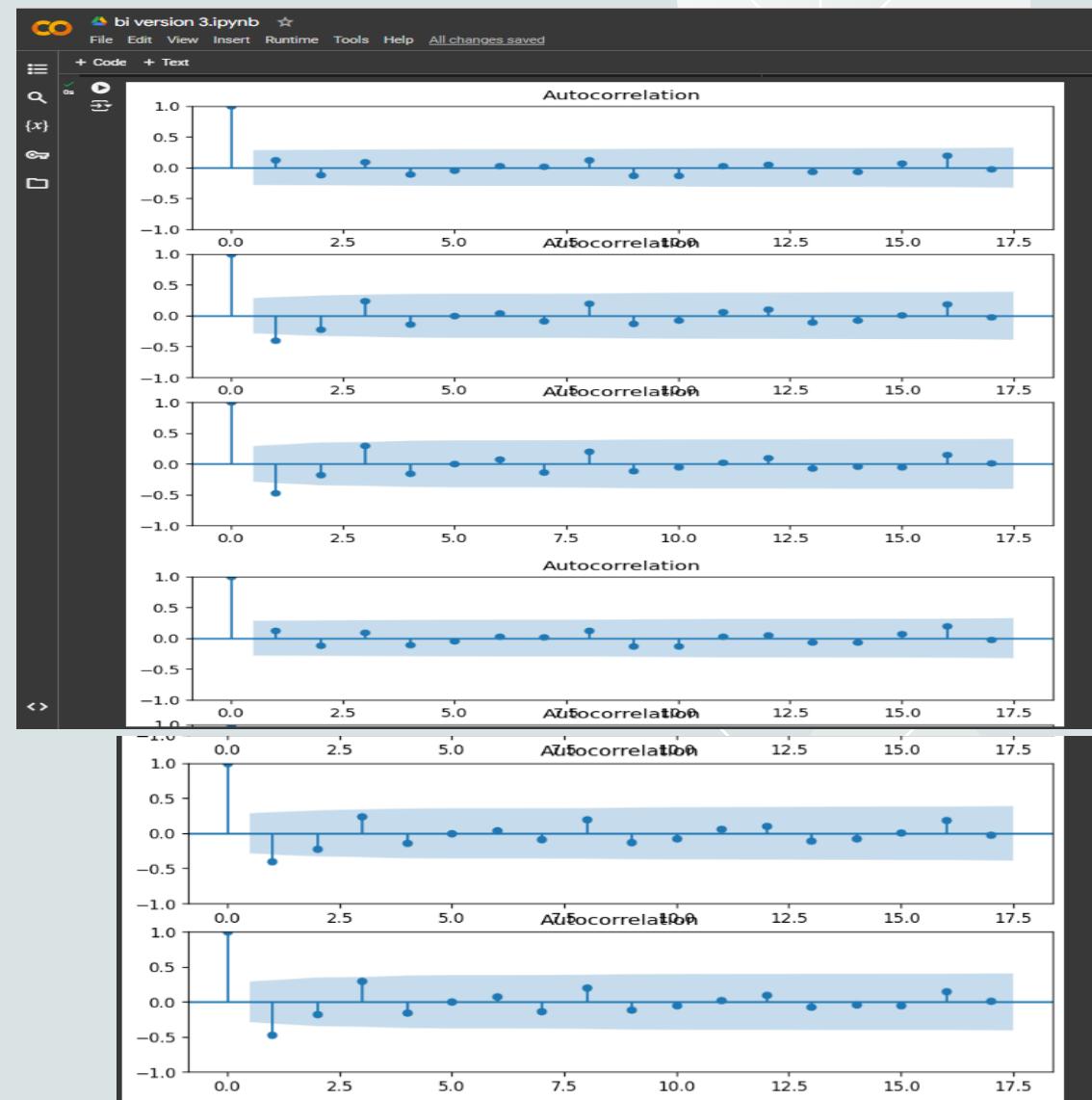
Time series visualization cont..



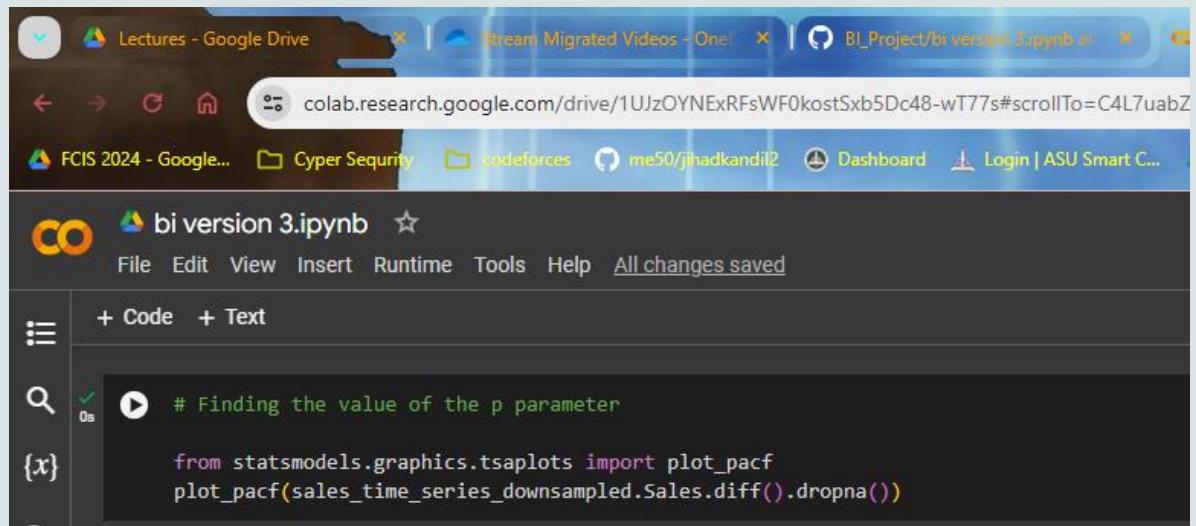


The screenshot shows a Jupyter Notebook interface. The title bar displays "bi version 3.ipynb". The menu bar includes File, Edit, View, Tools, Help, and "All changes saved". Below the menu is a toolbar with "+ Code" and "+ Text" buttons. The main area contains a code cell with the following content:

```
[45] # Step 3: plot ACF & PACF
from statsmodels.graphics.tsaplots import plot_acf
fig, (ax1, ax2, ax3) = plt.subplots(3)
plot_acf(sales_time_series_downsampled.Sales, ax=ax1)
plot_acf(sales_time_series_downsampled.Sales.diff().dropna(), ax=ax2)
plot_acf(sales_time_series_downsampled.Sales.diff().diff().dropna(), ax=ax3)
```



Time series visualization cont..



Lectures - Google Drive | Stream Migrated Videos - OneDrive | BI_Project/bi version 3.ipynb | colab.research.google.com/drive/1UJzOYNExRFsWF0kostSxb5Dc48-wT77s#scrollTo=C4L7uabZKRIA

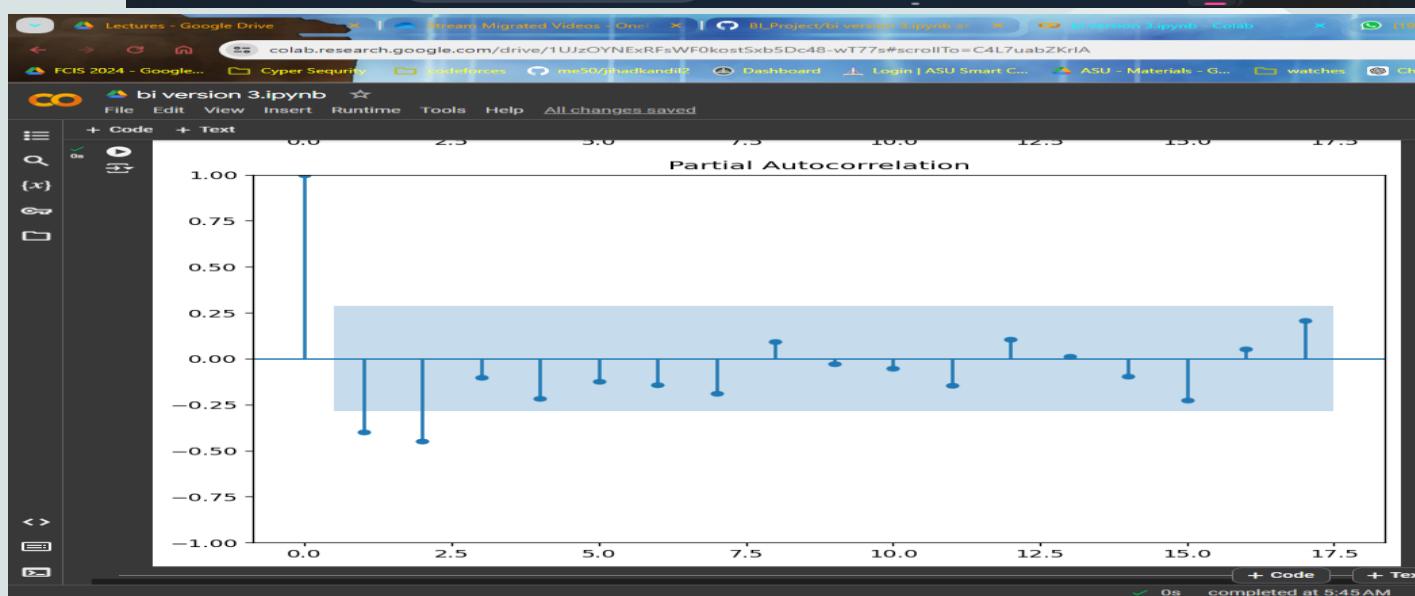
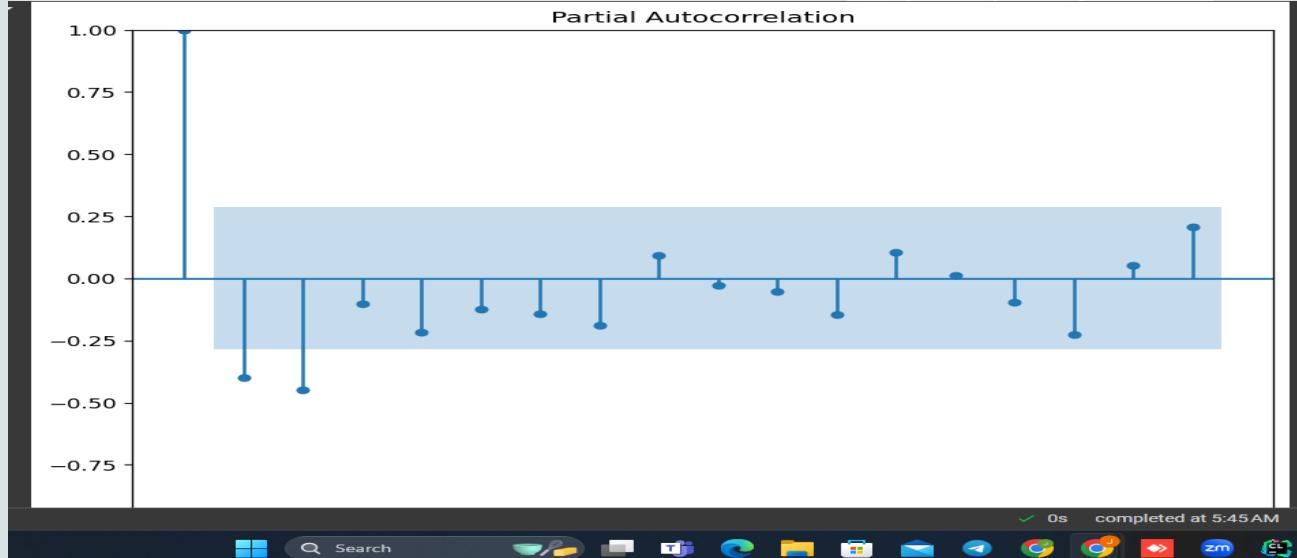
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

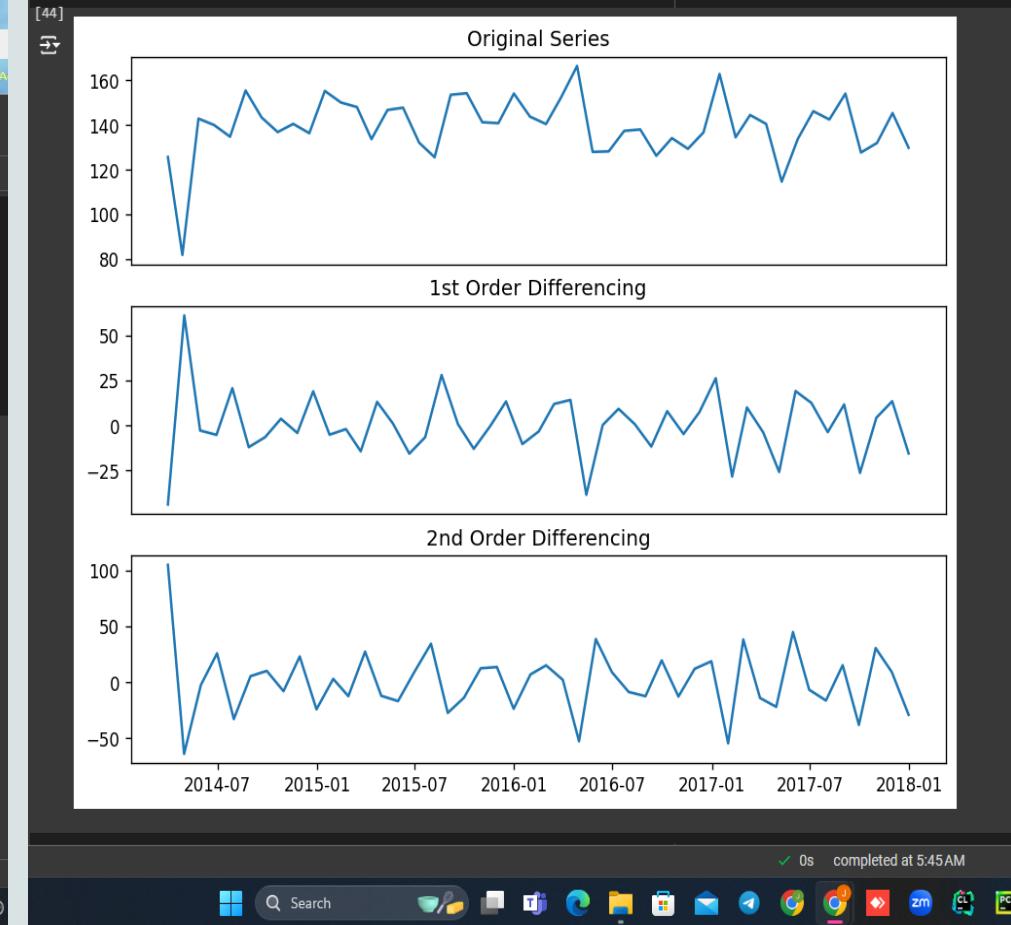
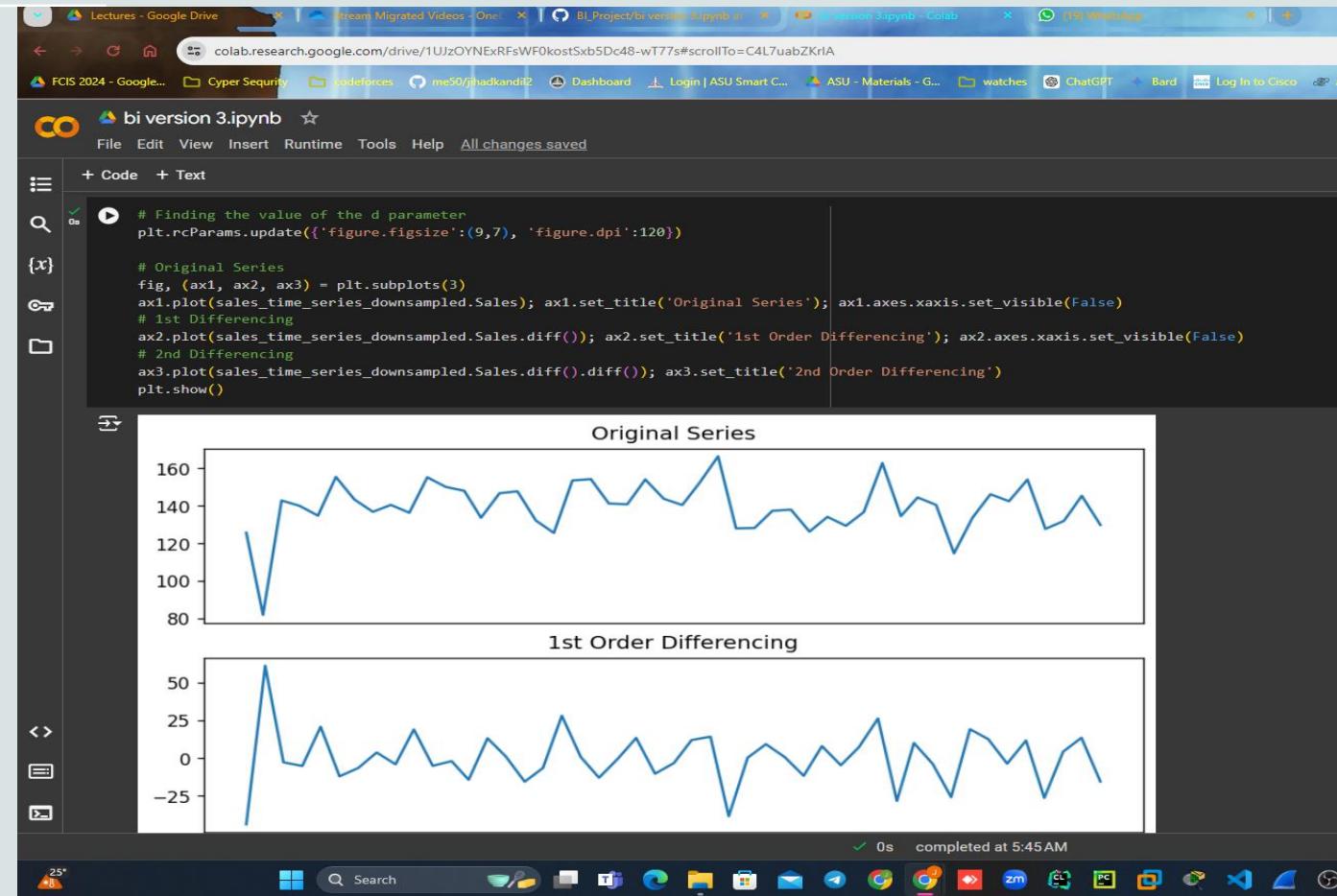
0s {x}

```
# Finding the value of the p parameter

from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(sales_time_series_downsampled.Sales.diff().dropna())
```



Time series visualization cont..



Time series visualization cont..

```
[49] from pmдарима import auto_arima
     import warnings
     warnings.filterwarnings('ignore')

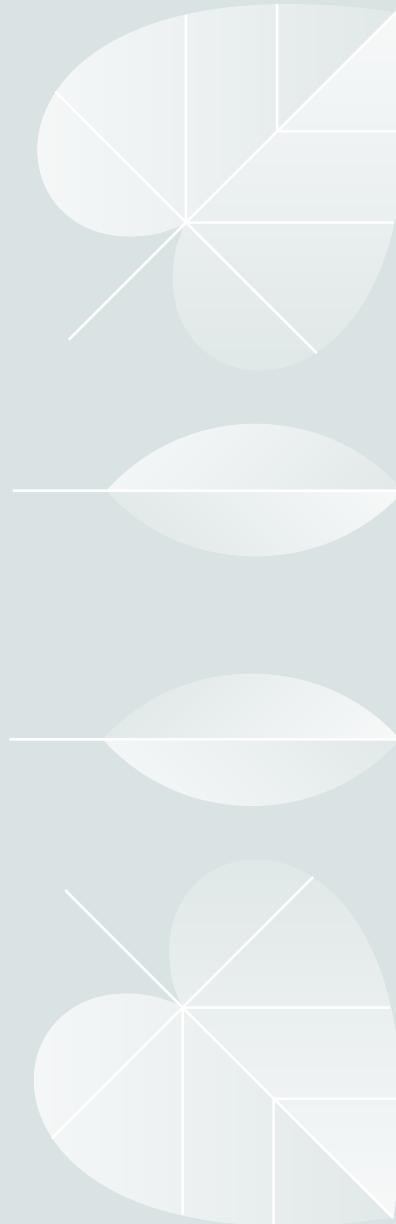
[50] stepwise_fit = auto_arima(sales_time_series_downsampled.Sales, trace=True, suppress_warnings=True)
     stepwise_fit.summary()

Performing stepwise search to minimize aic
ARIMA(2,0,2)(0,0,0)[0] intercept : AIC=393.555, Time=0.69 sec
ARIMA(0,0,0)(0,0,0)[0] intercept : AIC=389.520, Time=0.02 sec
ARIMA(1,0,0)(0,0,0)[0] intercept : AIC=390.789, Time=0.05 sec
ARIMA(0,0,1)(0,0,0)[0] intercept : AIC=390.484, Time=0.06 sec
ARIMA(0,0,0)(0,0,0)[0]          : AIC=612.639, Time=0.01 sec
ARIMA(1,0,1)(0,0,0)[0] intercept : AIC=390.388, Time=0.17 sec

Best model: ARIMA(0,0,0)(0,0,0)[0] intercept
Total fit time: 1.027 seconds
SARIMAX Results
Dep. Variable: y           No. Observations: 48
Model: SARIMAX            Log Likelihood   -192.760
Date: Sun, 19 May 2024      AIC             389.520
Time: 02:45:48              BIC             393.262
Sample: 01-31-2014          HQIC            390.934
                           - 12-31-2017

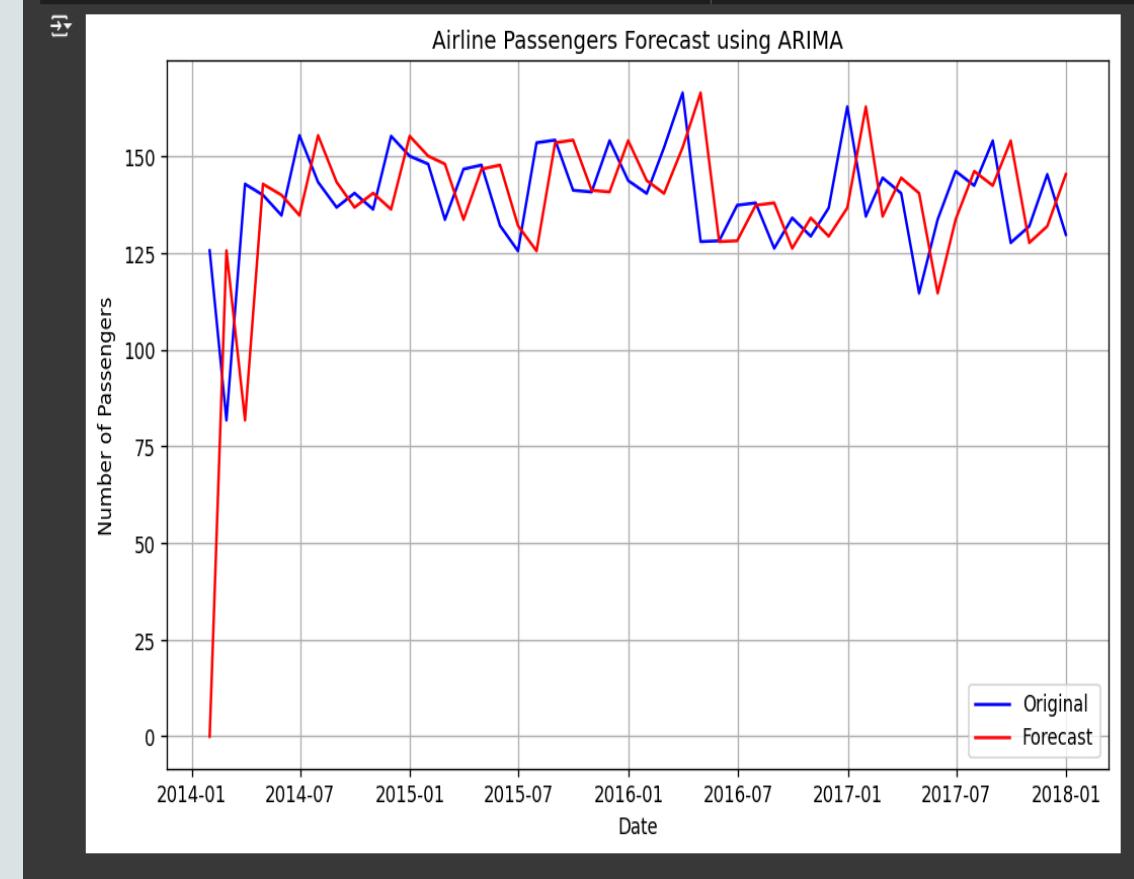
Covariance Type: opg
      coef  std err      z  P>|z|  [0.025  0.975]
intercept 139.3887 2.269  61.431 0.000 134.941 143.836
sigma2    180.1487 22.699  7.936 0.000 135.660 224.638
Ljung-Box (L1) (Q): 0.76 Jarque-Bera (JB): 69.70
Prob(Q): 0.38 Prob(JB): 0.00
Heteroskedasticity (H): 0.45 Skew: -1.40
Prob(H) (two-sided): 0.12 Kurtosis: 8.20

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```



Time series visualization cont..

```
[51] from statsmodels.tsa.arima_model import ARIMA  
  
arima model  
  
[52] import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
from statsmodels.tsa.stattools import adfuller  
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf  
from statsmodels.tsa.arima.model import ARIMA  
  
# Step 4: Build the ARIMA model  
  
model = ARIMA(sales_time_series_downsampled['Sales'], order=(0,1,0))  
arima_model = model.fit()  
  
# Step 5: Predict  
forecast = arima_model.predict()  
  
[53] # Plot the forecast  
plt.figure(figsize=(10, 6))  
plt.plot(sales_time_series_downsampled.index,sales_time_series_downsampled['Sales'], label='Original', color='blue')  
plt.plot(sales_time_series_downsampled.index , forecast, label='Forecast', color='red')  
plt.title('Airline Passengers Forecast using ARIMA')  
plt.xlabel('Date')  
plt.ylabel('Number of Passengers')  
plt.legend(loc='best')  
plt.grid(True)  
plt.show()
```



Time series visualization cont..

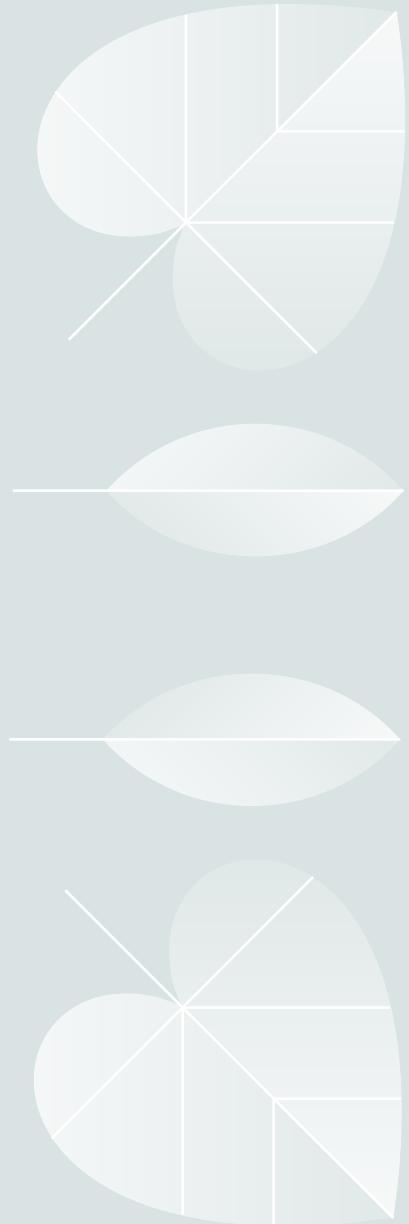
```
0s [54] sales_time_series_downsampled.head()

      Sales
Order Date
2014-01-31  125.677532
2014-02-28  81.795696
2014-03-31  142.873478
2014-04-30  140.038081
2014-05-31  134.785861

Next steps: Generate code with sales\_time\_series\_downsampled View recommended plots

0s [55] sales_time_series_downsampled.tail()

      Sales
Order Date
2017-08-31  154.082486
2017-09-30  127.707412
2017-10-31  131.965806
2017-11-30  145.376375
2017-12-31  129.836054
```



Time series visualization cont..



Conclusion

- By synthesizing our findings, we aim to equip Walmart's stakeholders with actionable recommendations to enhance sales performance, optimize resource allocation, and drive sustainable growth.