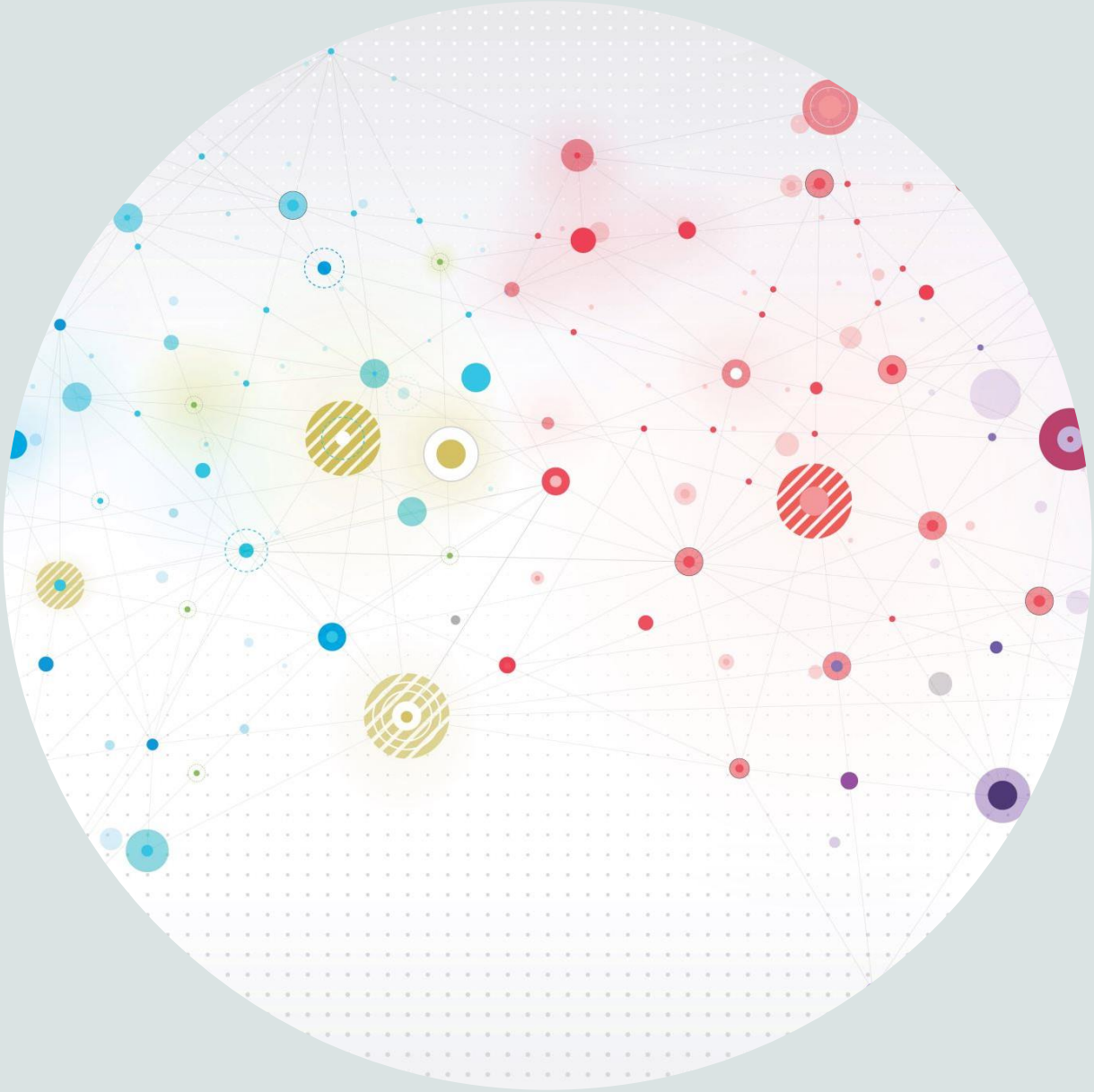
Walmart Sales Project



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

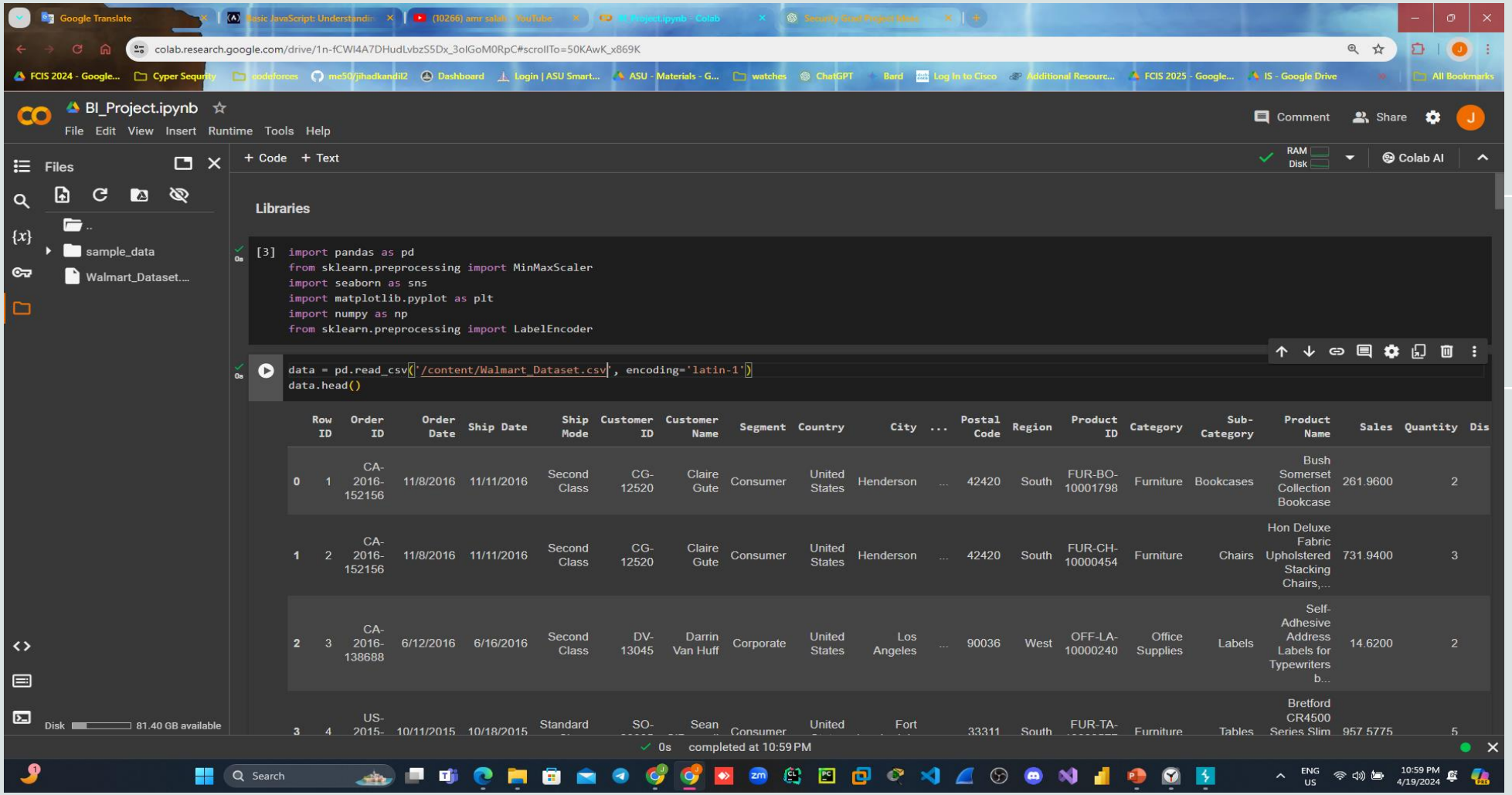
Introduction

- 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

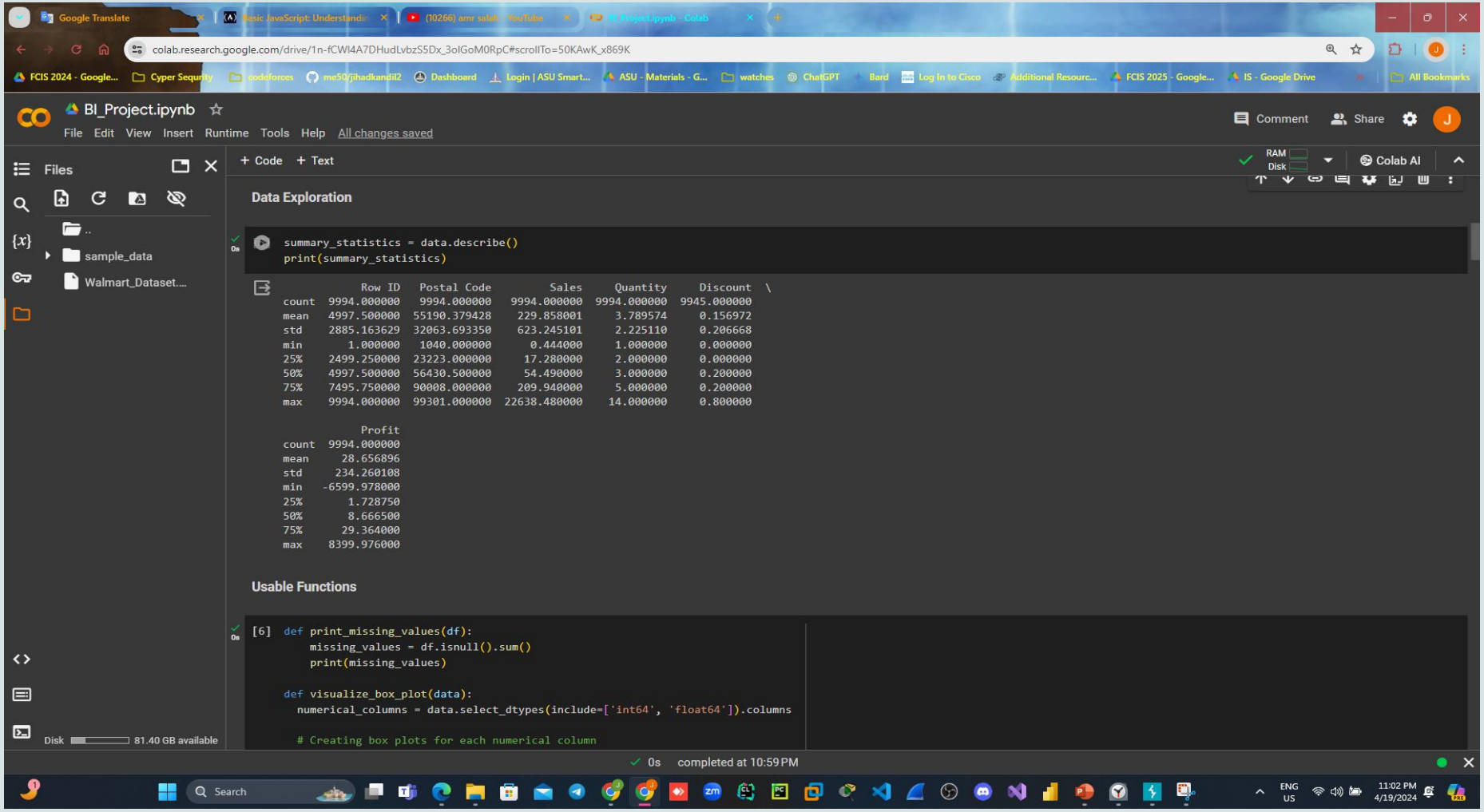
- The objective of this project is to analyze the Walmart sales dataset and provide actionable insights to the owners to optimize business strategies.
- Determine which product categories generate the highest sales revenue, allowing Walmart to focus resources and marketing efforts on high-performing 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



	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	1	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	2	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	3	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	
3	4	US-2015-	10/11/2015	10/18/2015	Standard	SO-	Sean	Consumer	United	Fort		33311	South	FUR-TA-	Furniture	Tables	Bretford CR4500 Series Slim	957.5775	5	

Second :Data Exploration

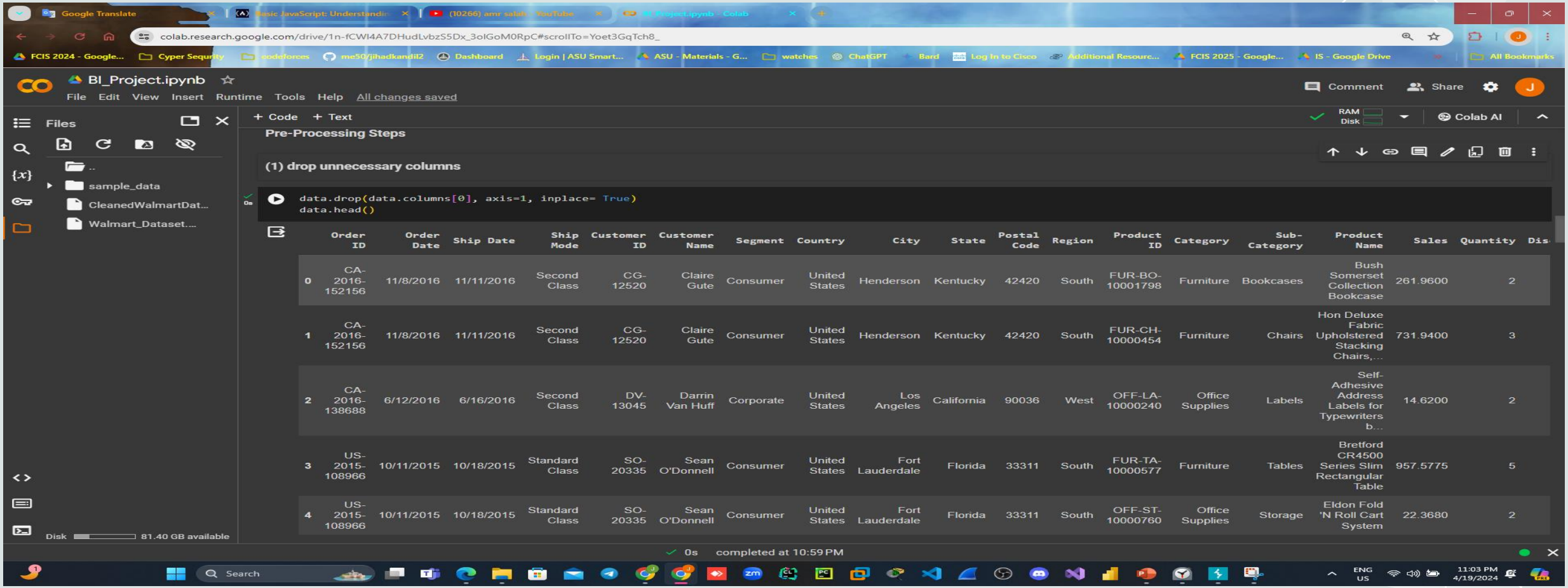


Data preprocessing

- 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:

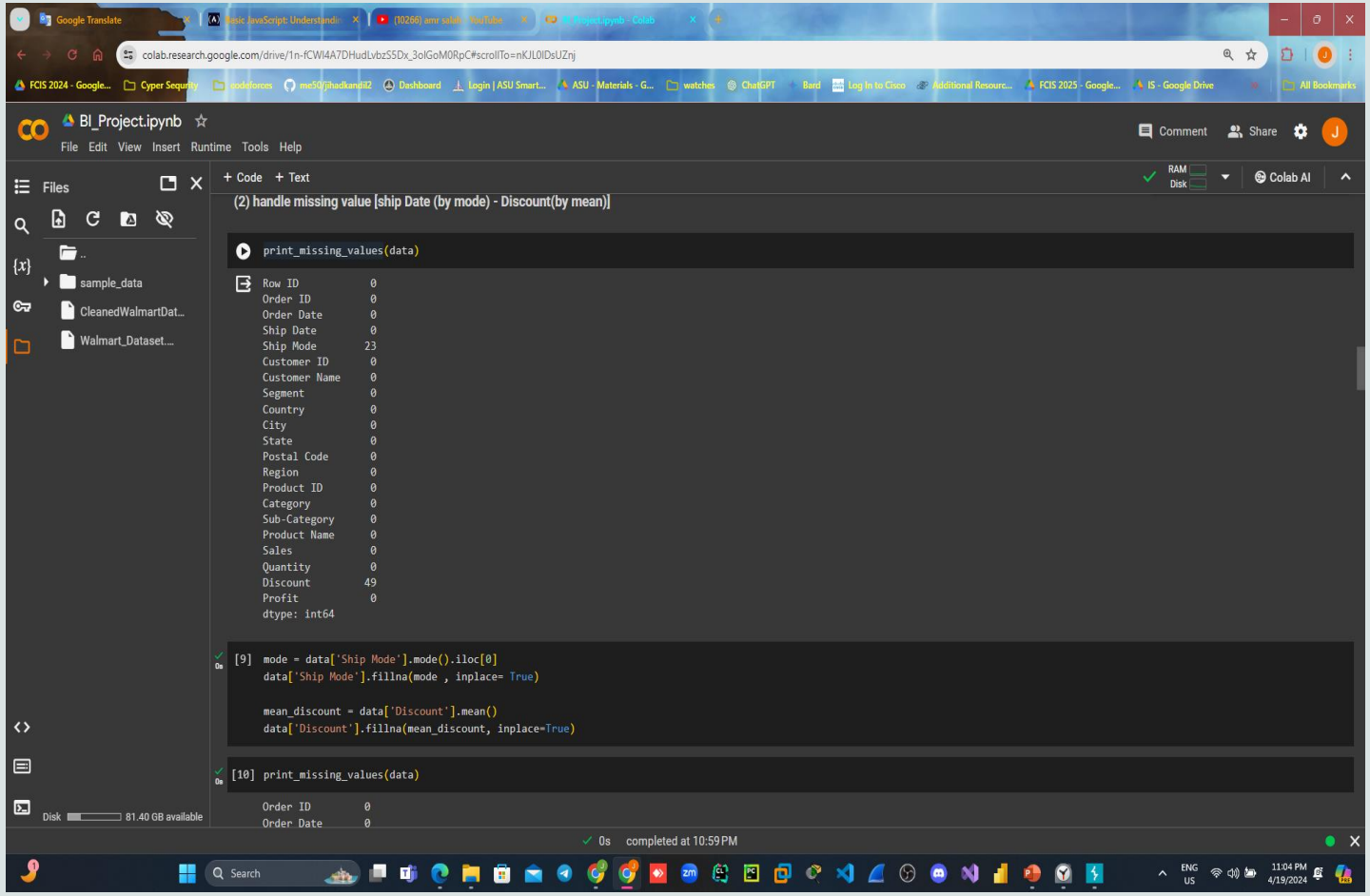
- Code: output:



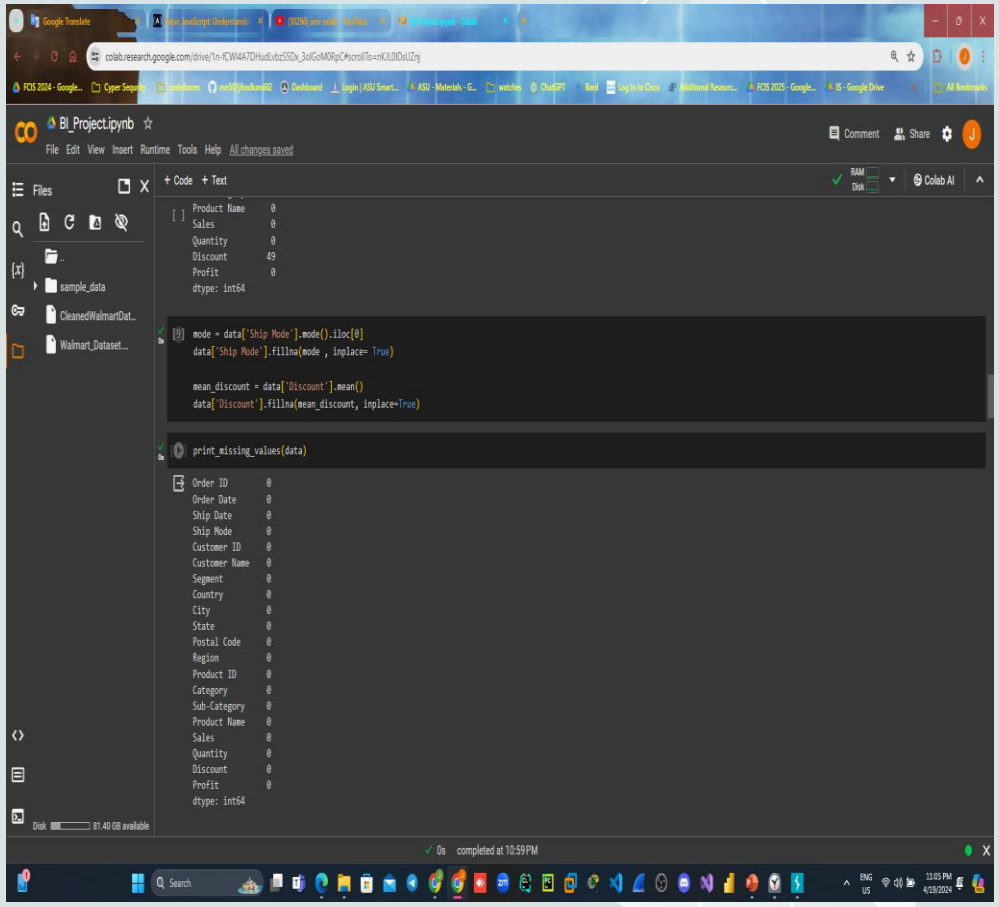
	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Dis
0	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	42420	South	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	

[2] Handle missing values (Null):

- Code: before handling output

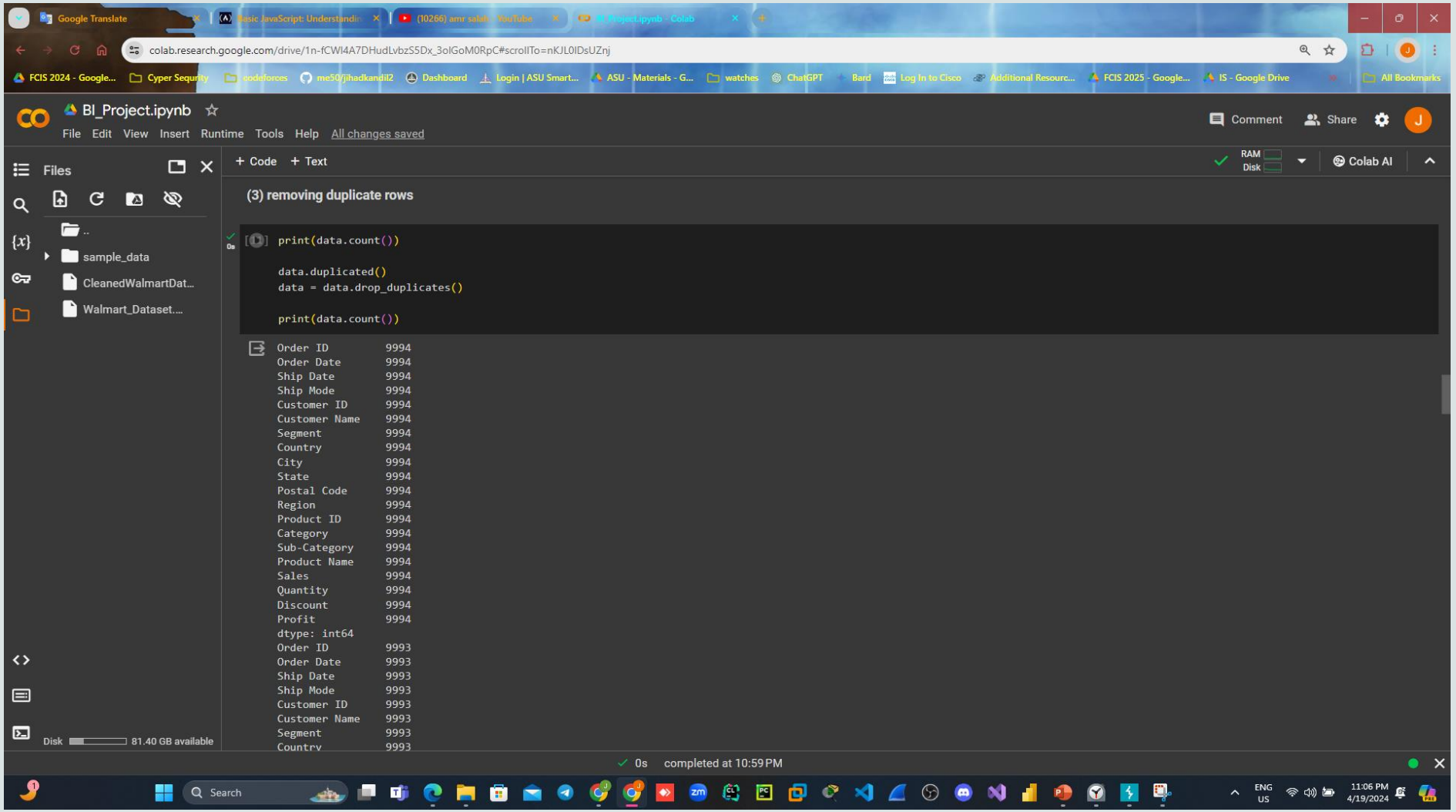


After handling output:



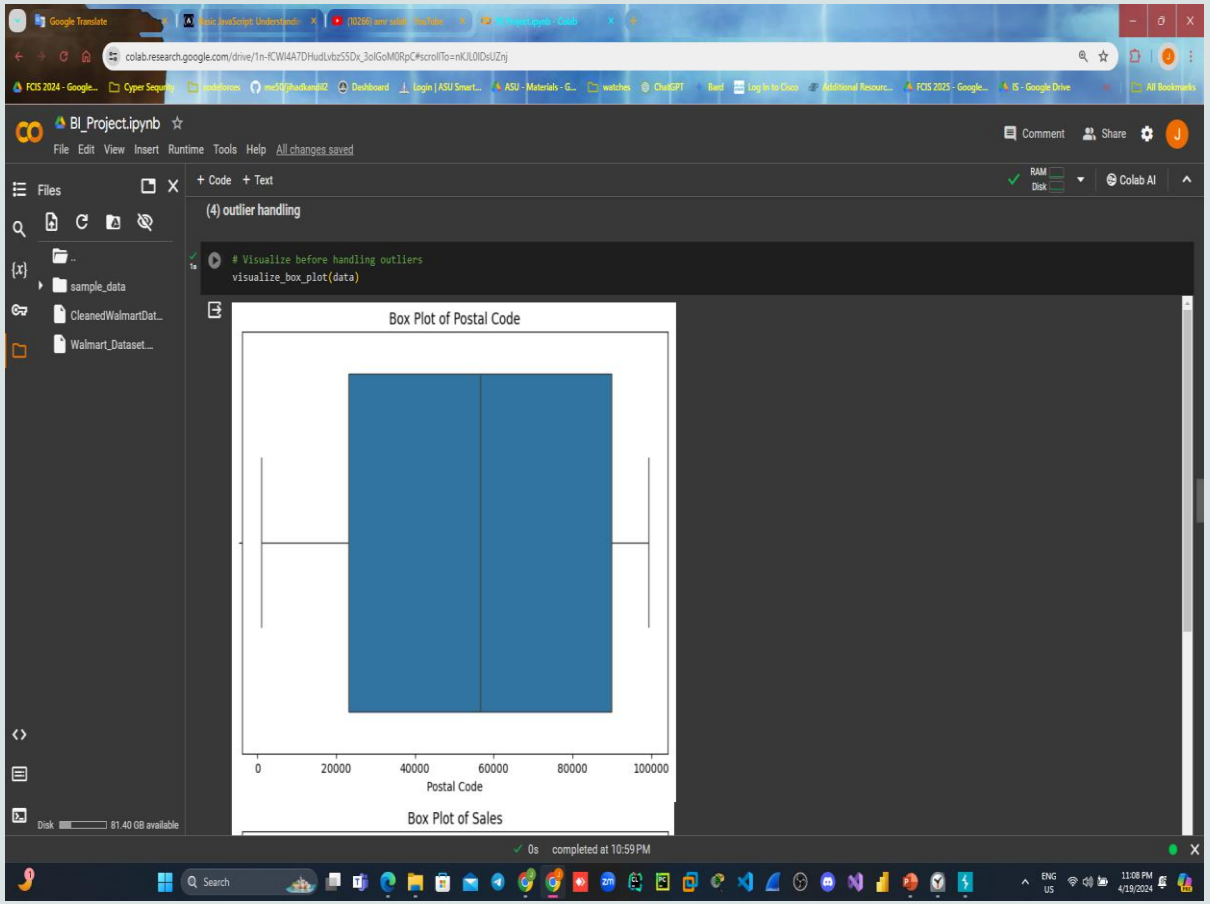
[3] Handle duplicates:

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



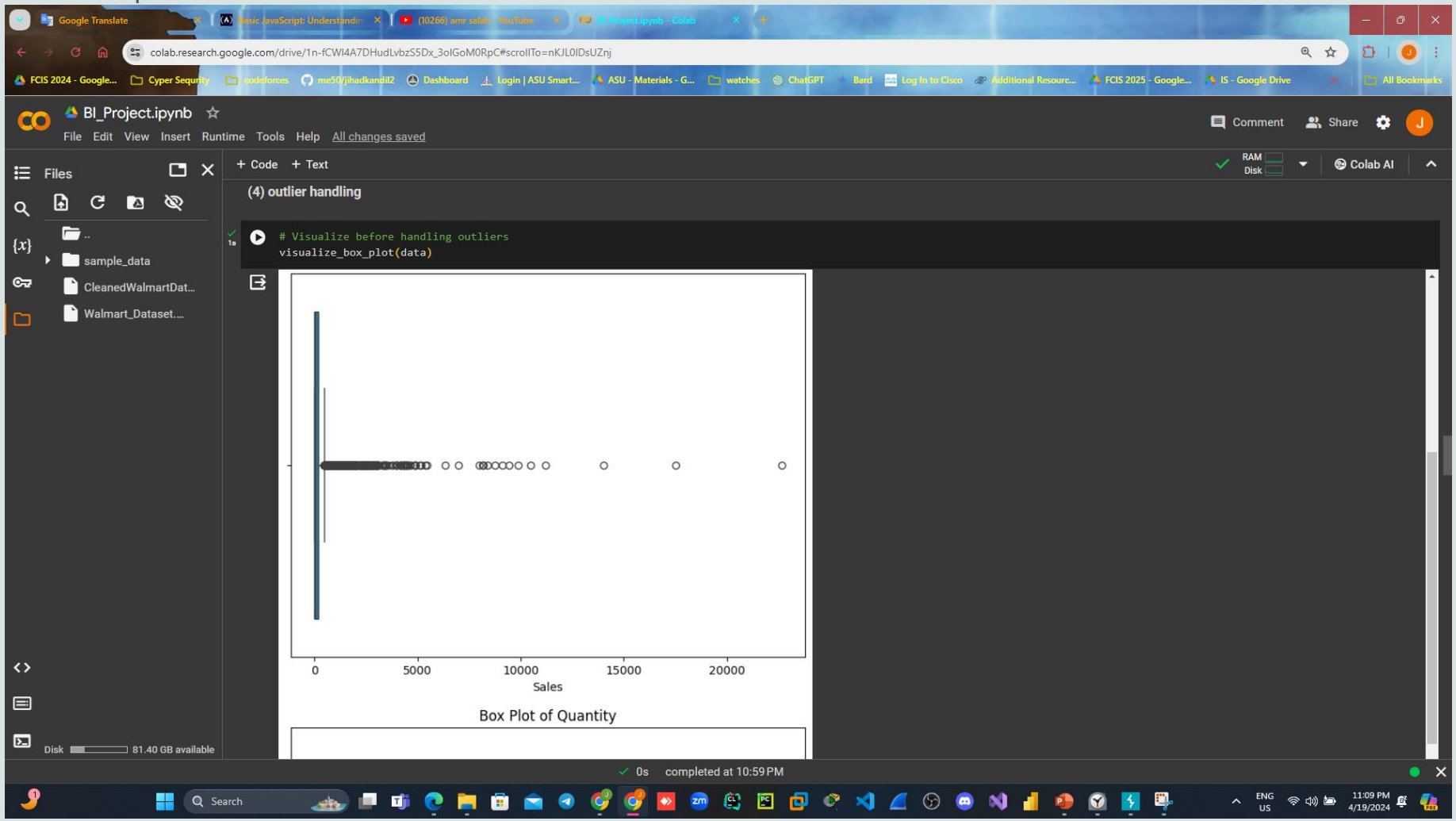
[4] outlier handling :

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



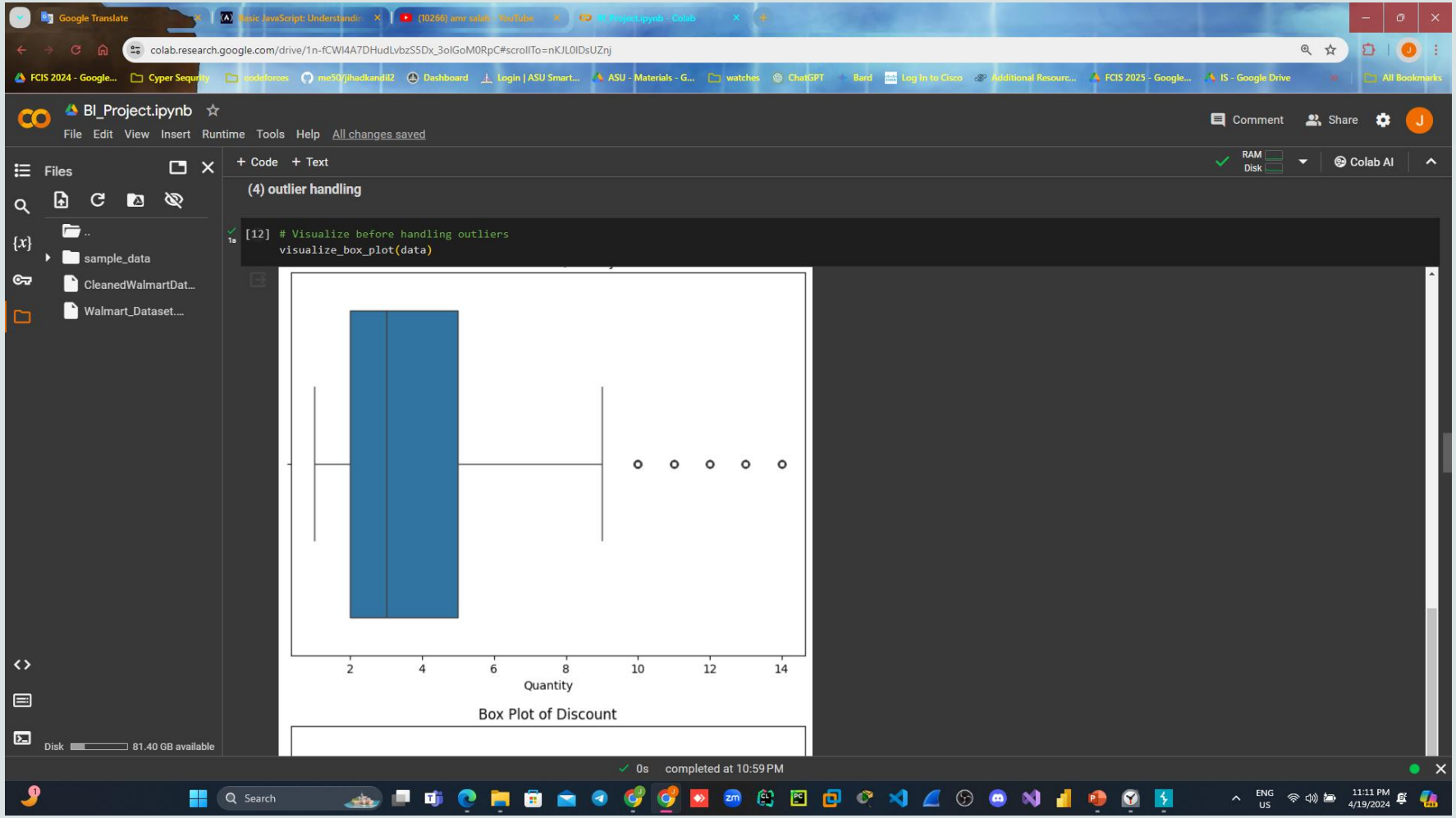
[4] outlier handling continue.....

- Box plot of sales



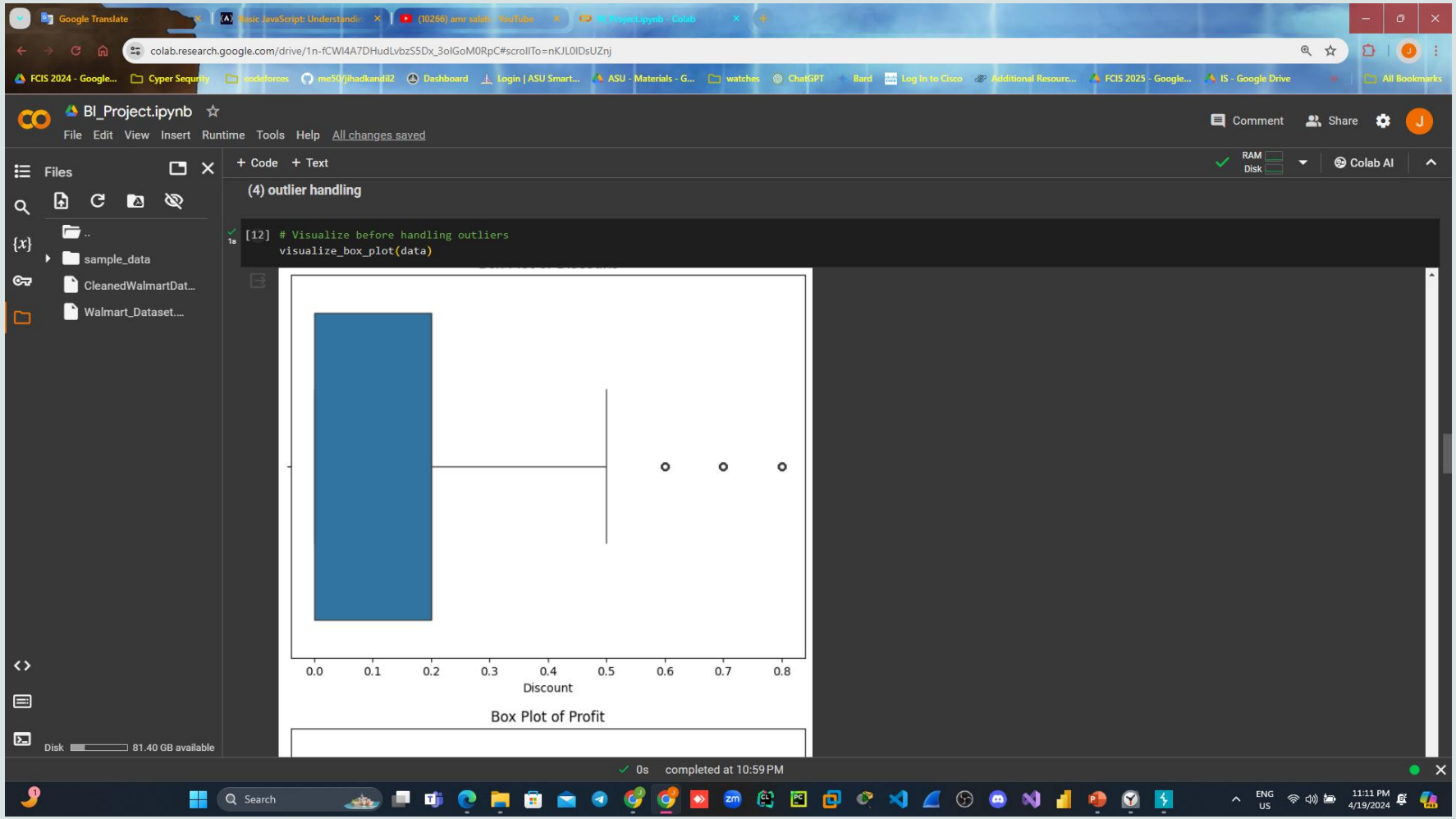
[4] outlier handling continue.....

- Box plot of Quantity



[4] outlier handling continue.....

- Box plot of Discount



[4] outlier handling continue.....

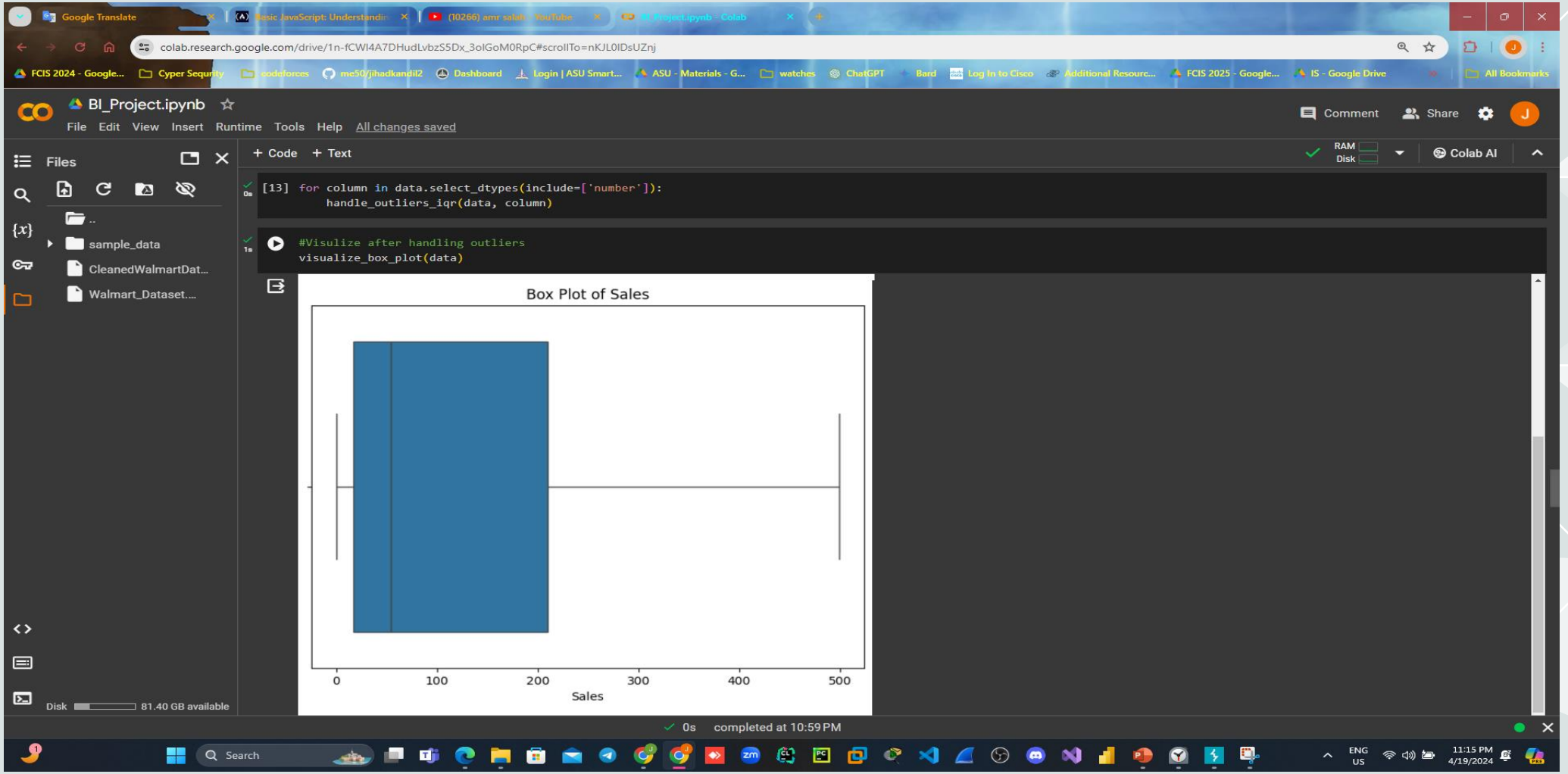
- Box plot of Profit

[4] outlier handling continue.....

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

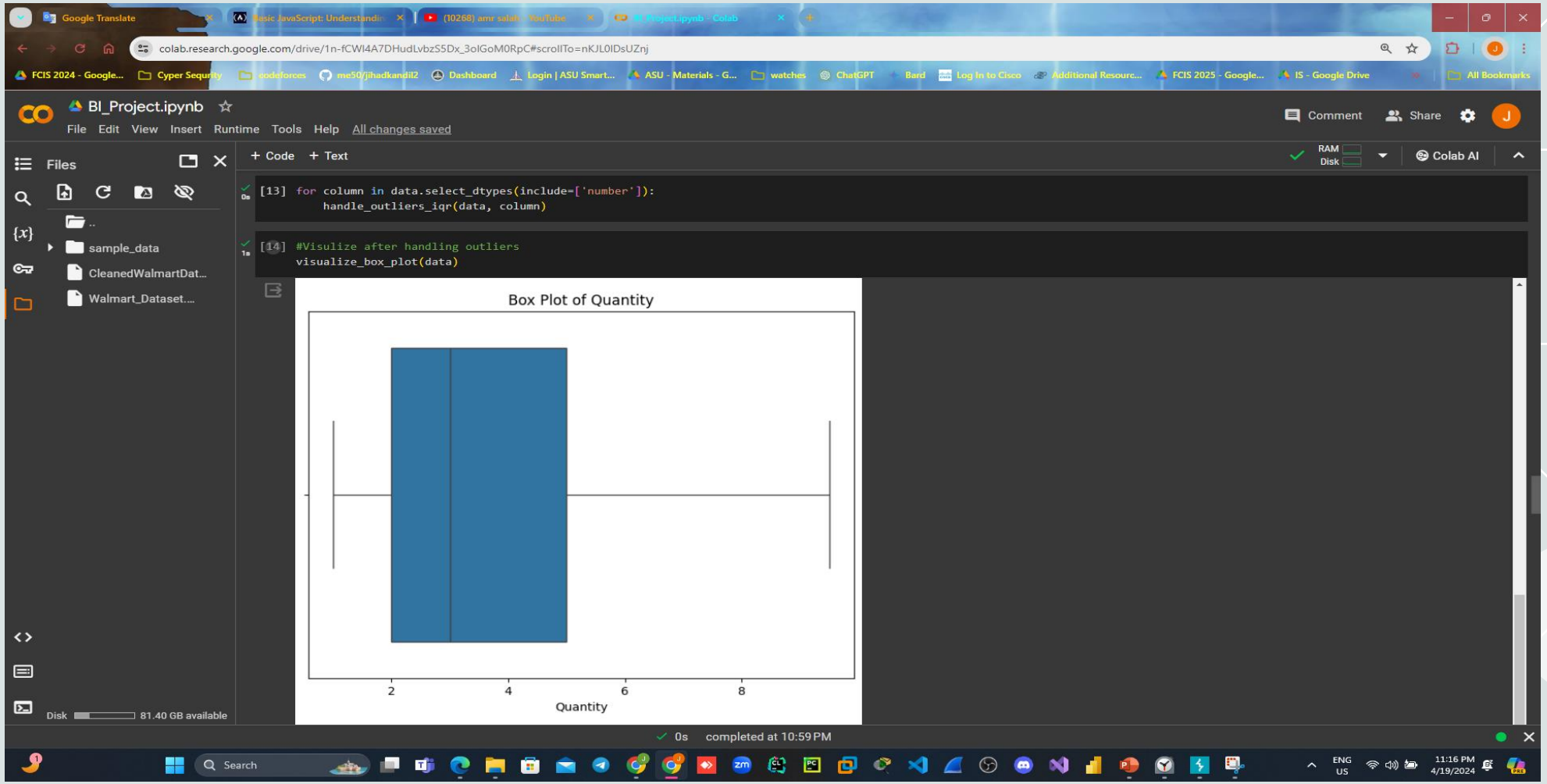
[4] outlier handling continue.....

- Box plot of Sales:



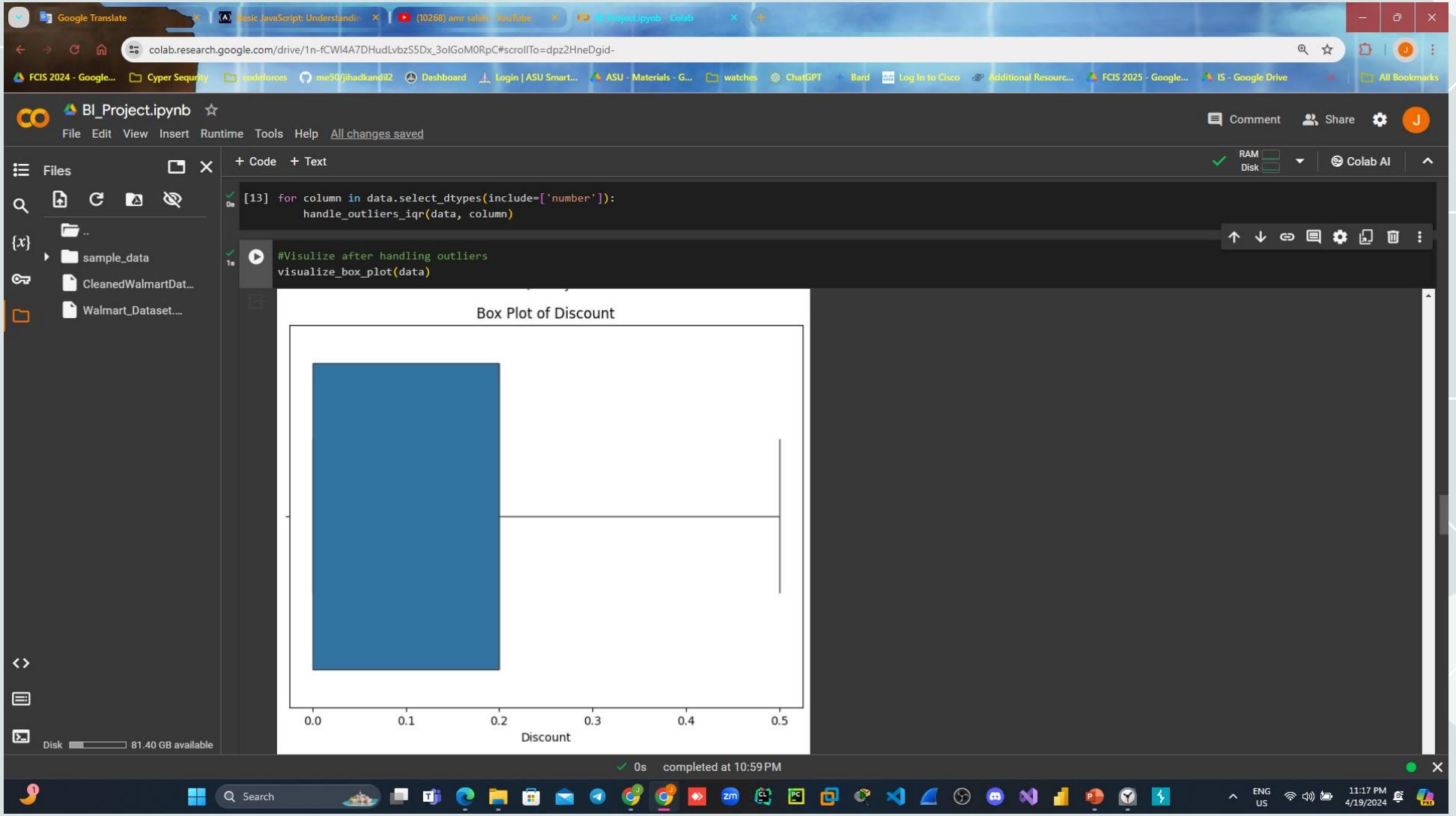
[4] outlier handling continue.....

- Box plot of Quantity:



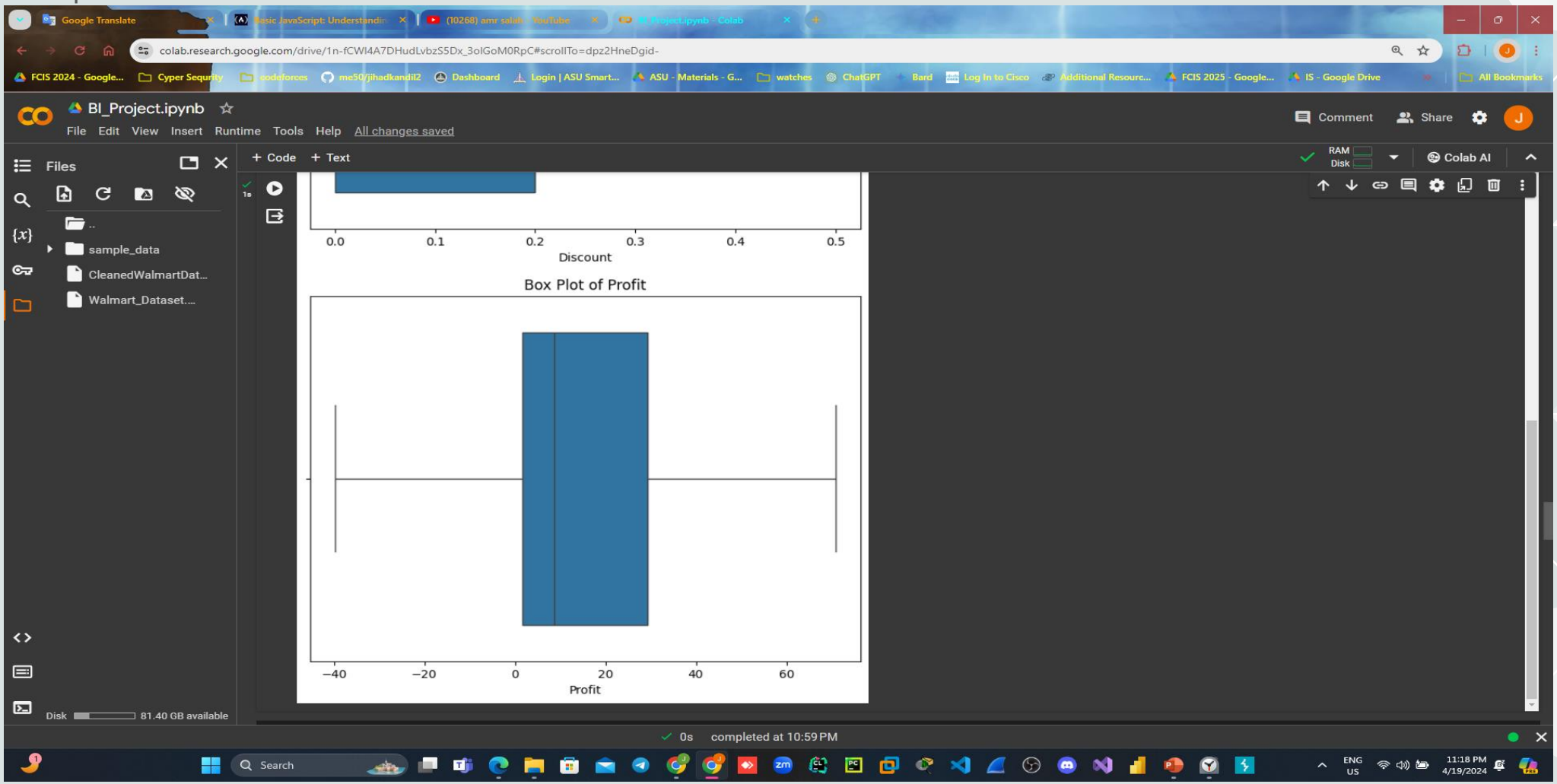
[4] outlier handling continue.....

- Box plot of Discount:



[4] outlier handling continue.....

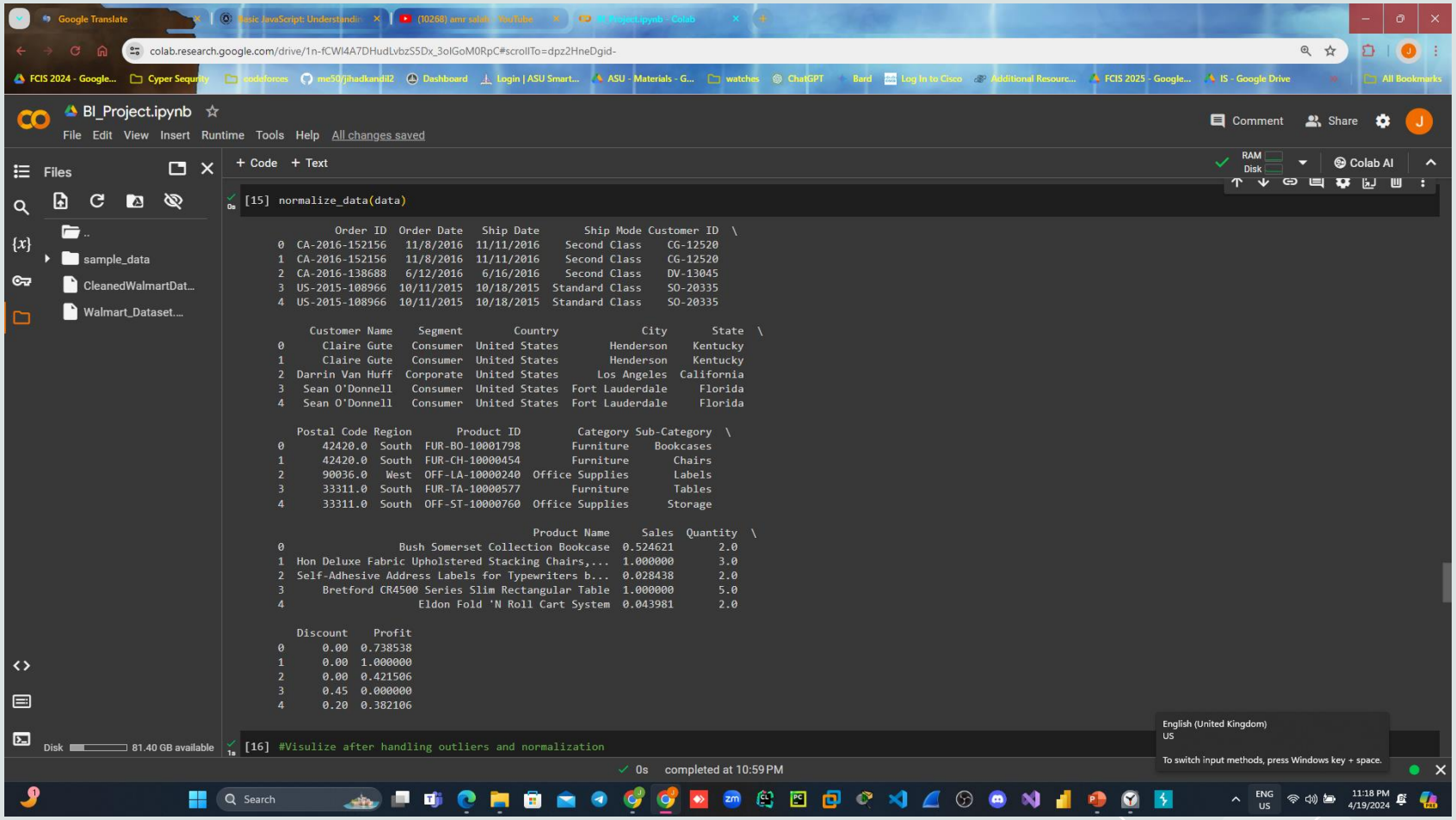
- Box plot of Profit



[5] Normalization:

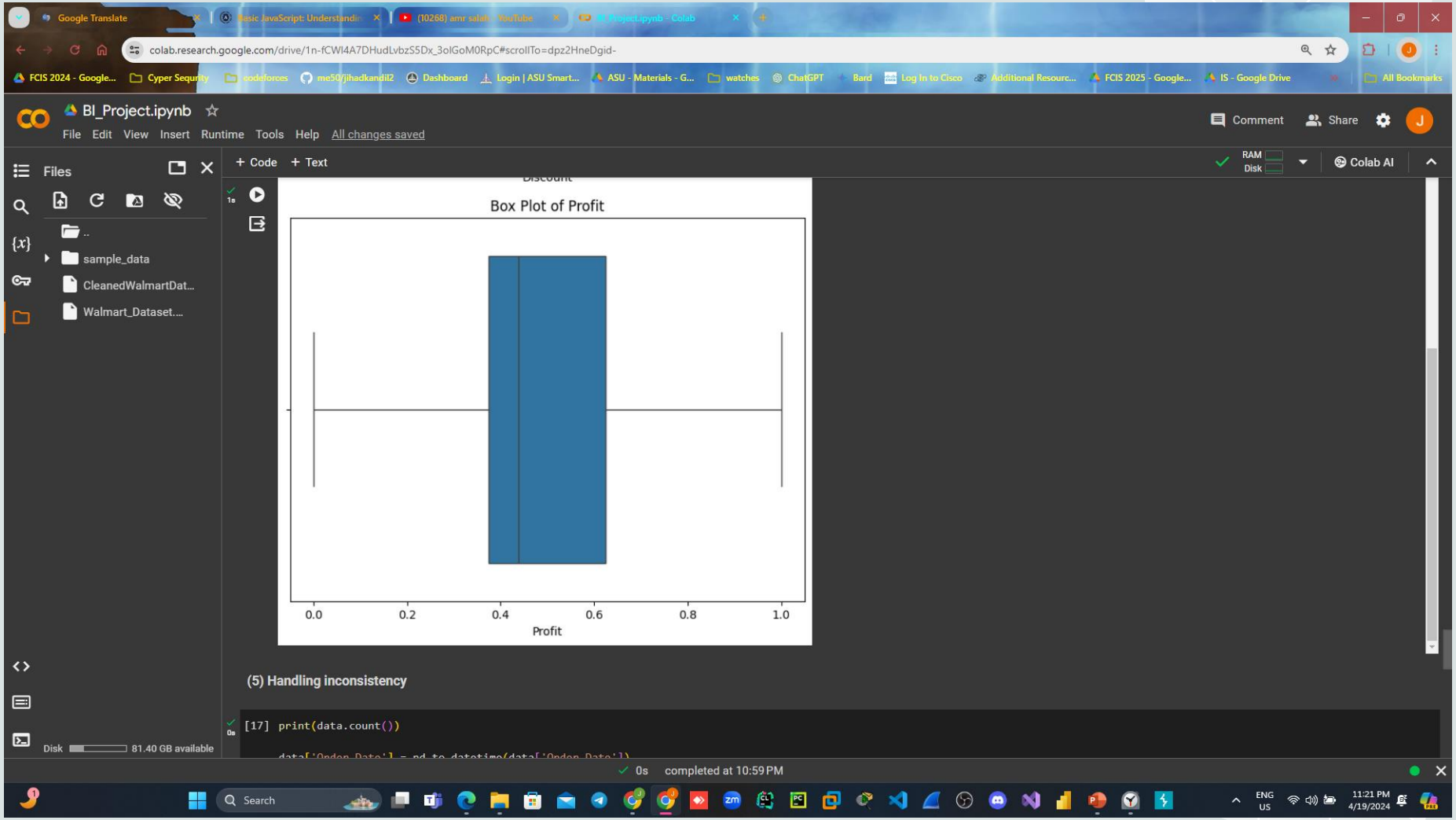
- Code:

we normalize data to handle negative value of the data
so as we view box plot of the profit column to make sure :



[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

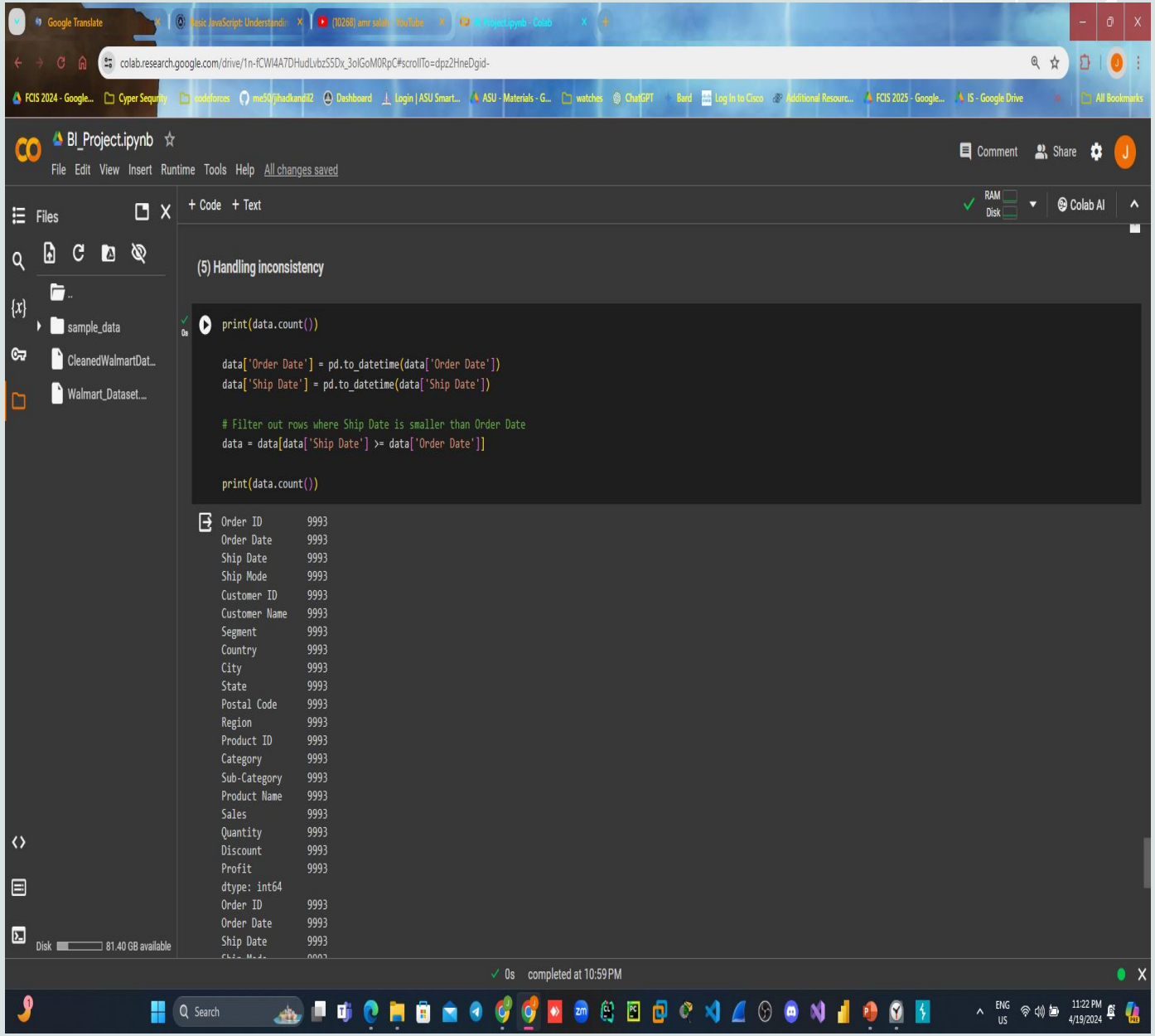
the consistency could found if

The shipping date is before order date

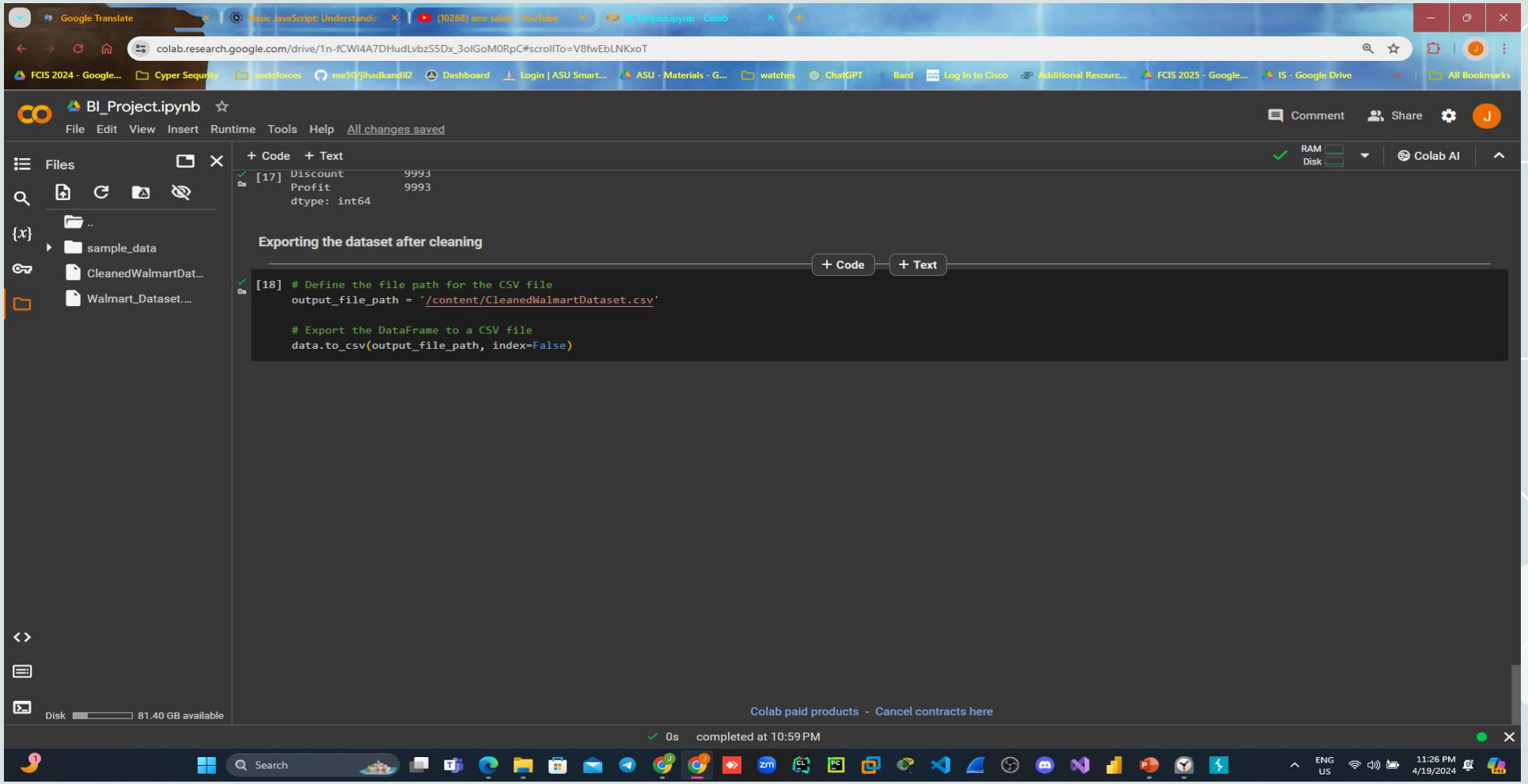
If this found in the data we should drop:

So we check by count we find no conflict

Consistencies.



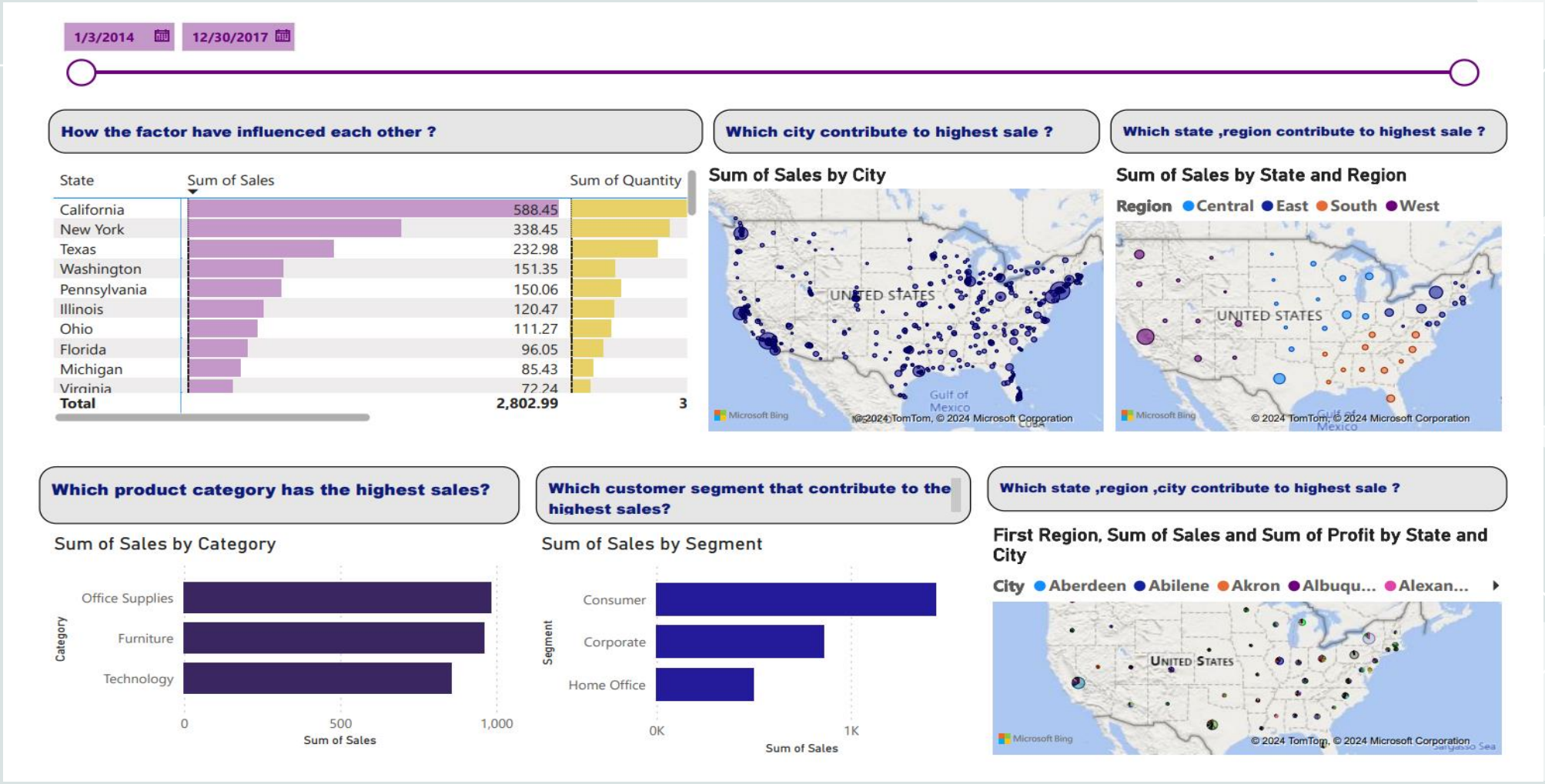
Exporting data set after cleaning it:



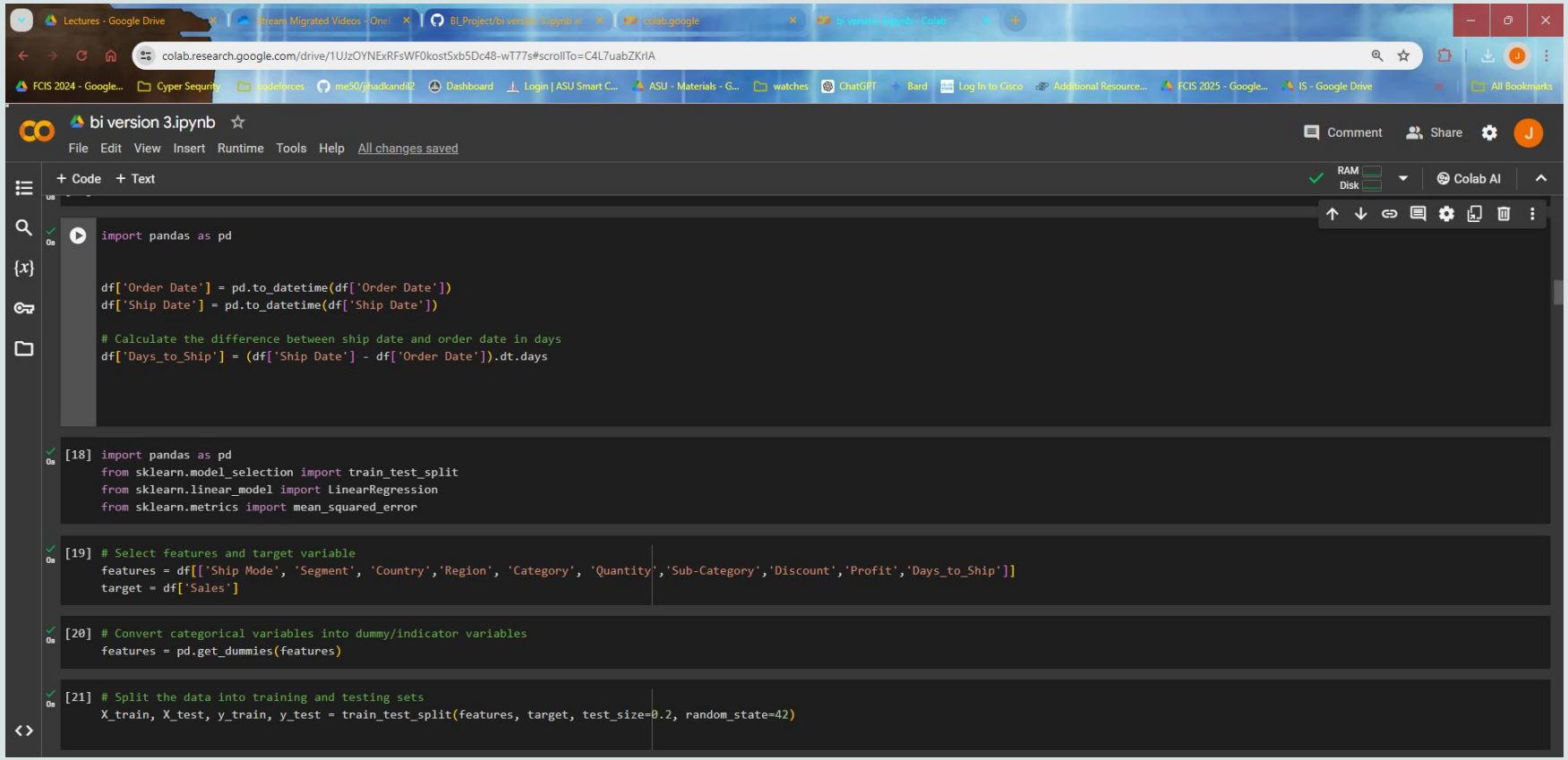
Data visualization

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

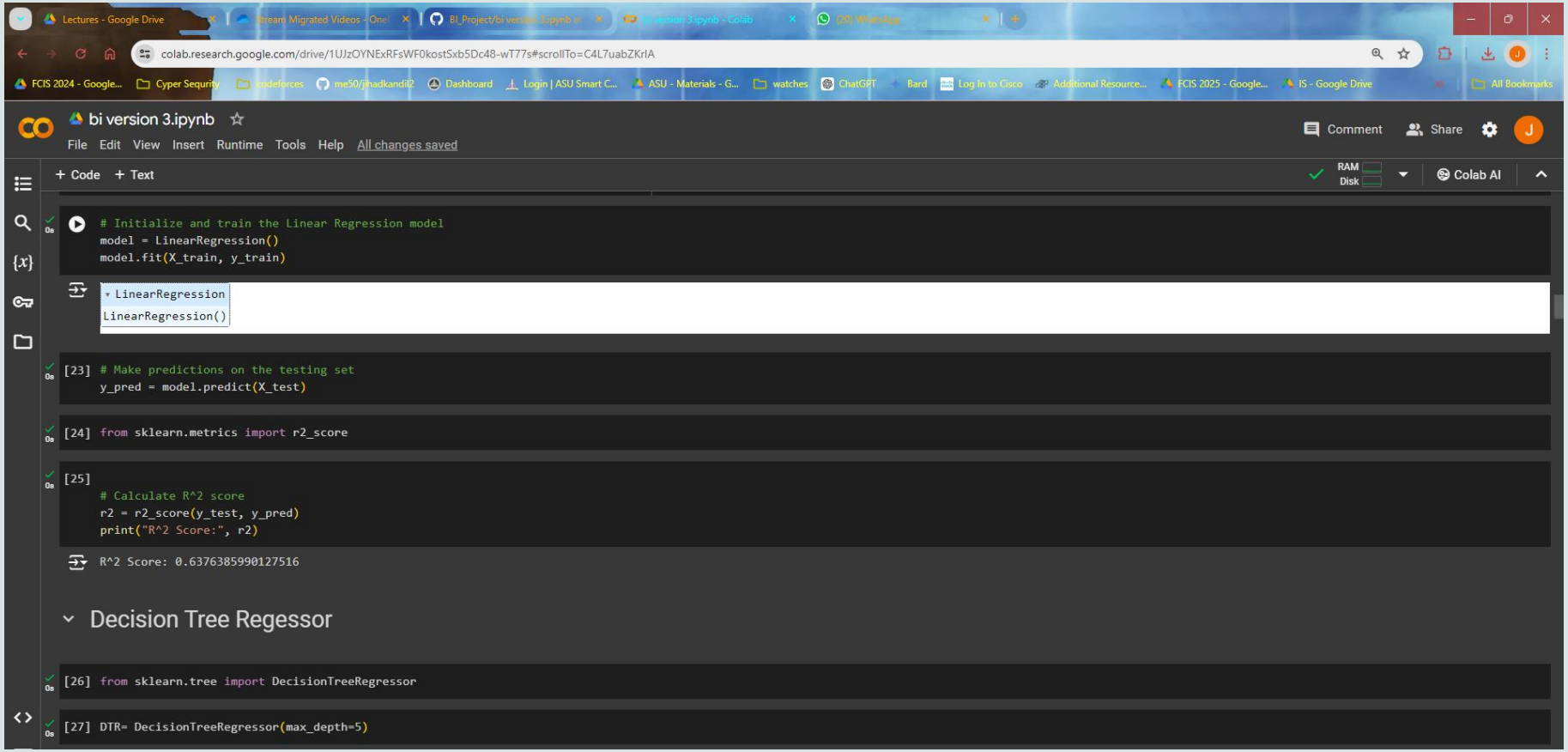
Data visualization continue...



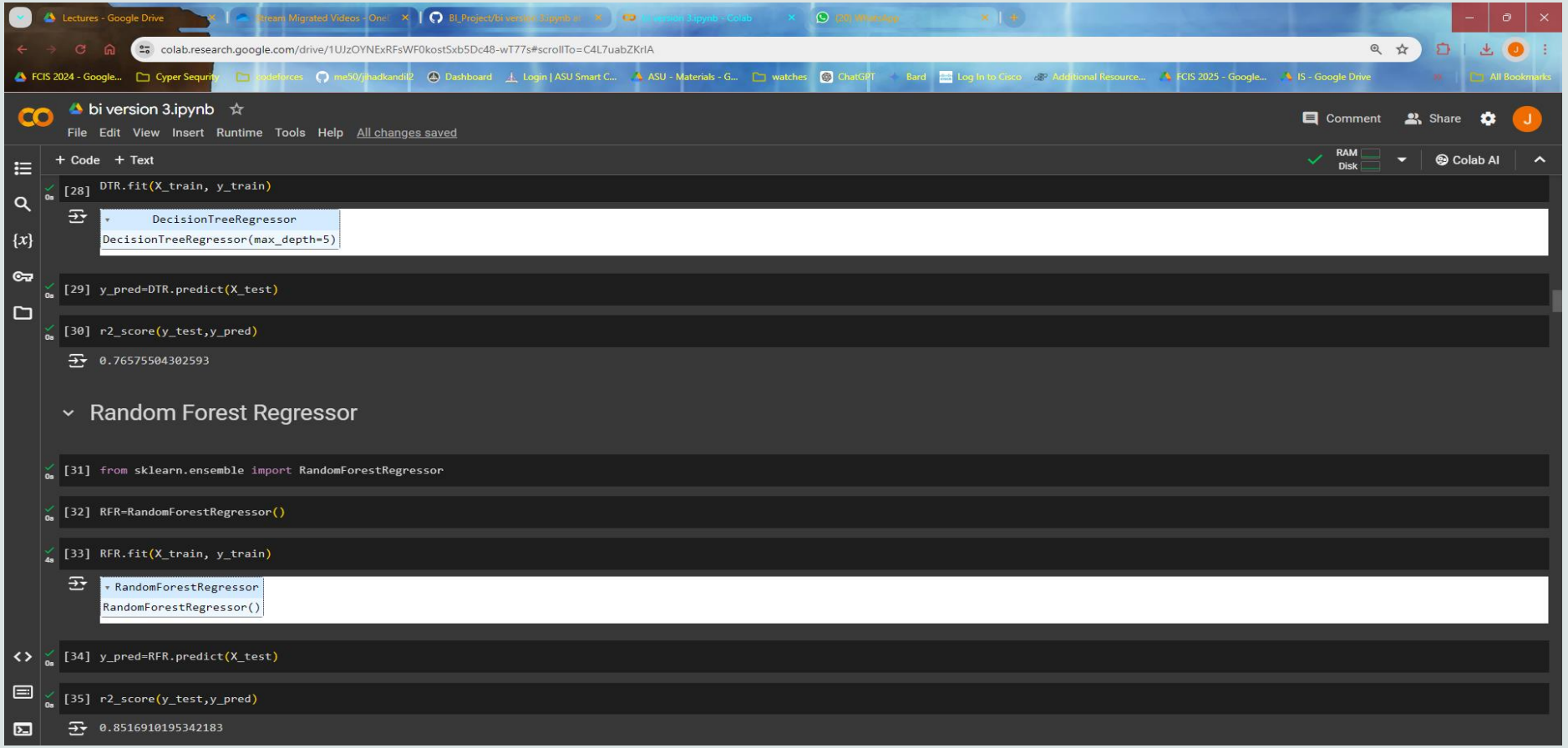
Model


bi version 3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Colab AI
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)

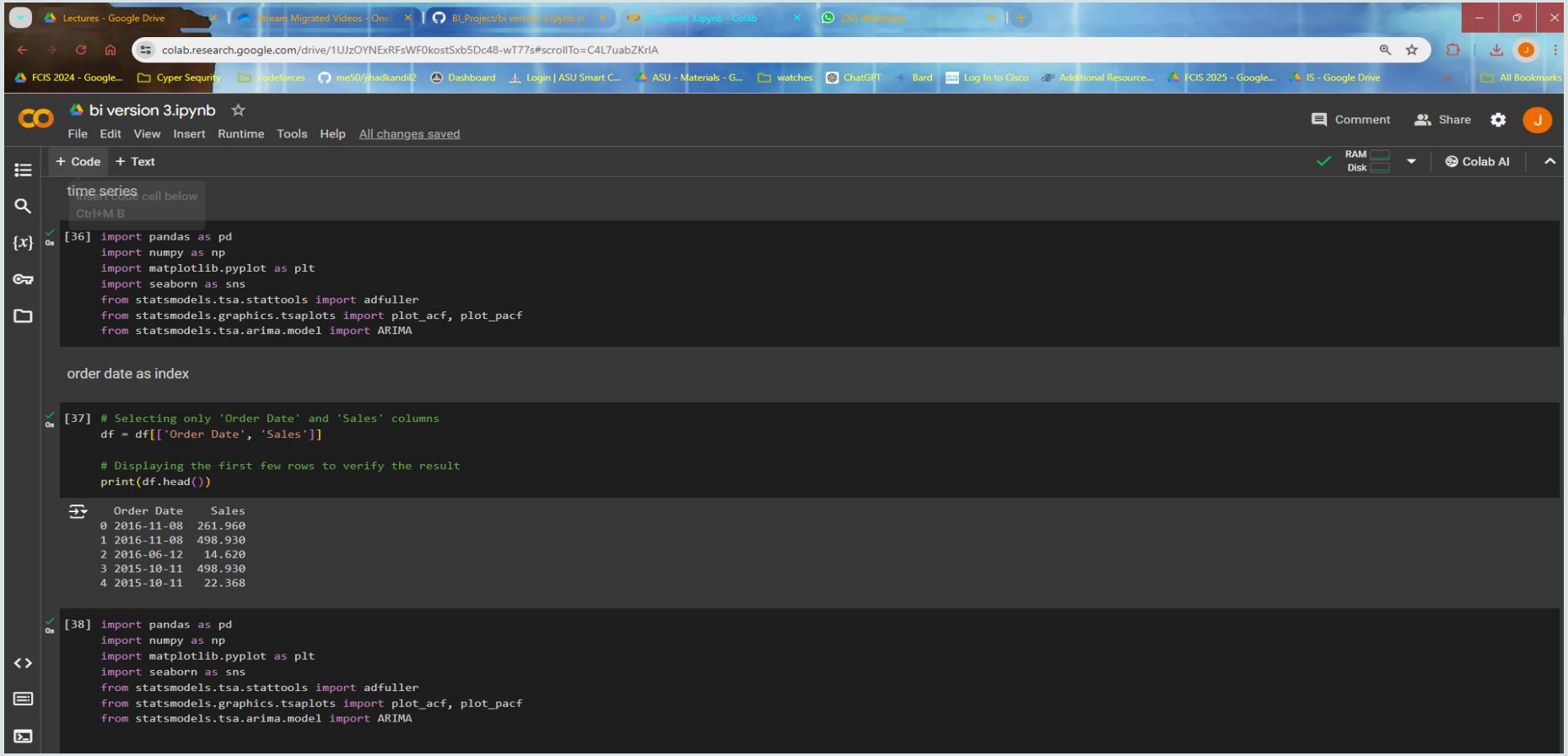
Model cont..


bi version 3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Colab AI
# Initialize and train the Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)
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 Regessor
[26] from sklearn.tree import DecisionTreeRegressor
[27] DTR= DecisionTreeRegressor(max_depth=5)

Model cont..


colab.research.google.com/drive/1UJzOYNExRFsWF0kostSxb5Dc48-wT77s#scrollTo=C4L7uabZKrIA
bi version 3.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Colab AI
[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..



Time series visualization cont..

```
[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()
```

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..

bi version 3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

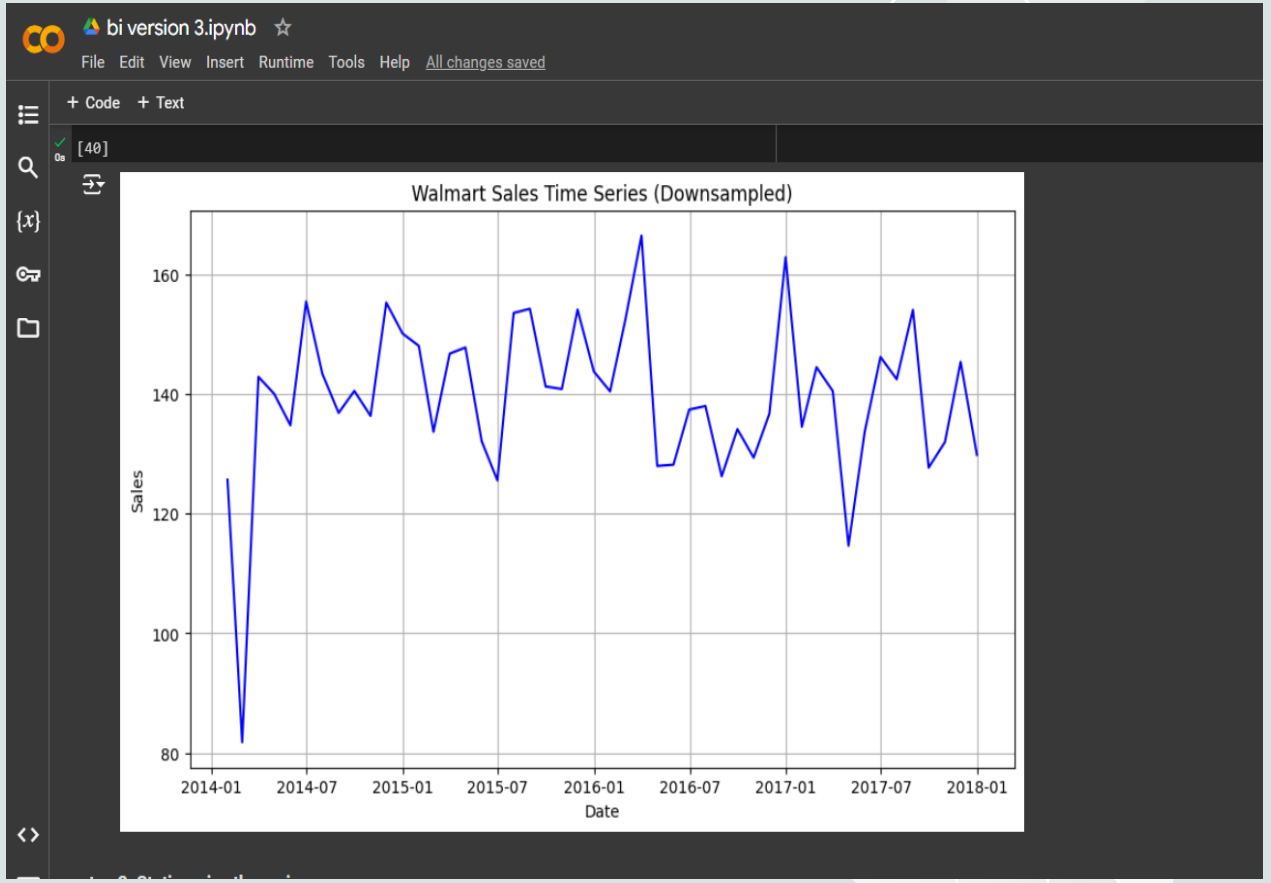
Comment Share

+ Code + Text

RAM Disk Colab AI

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

```
[40] # step 1

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

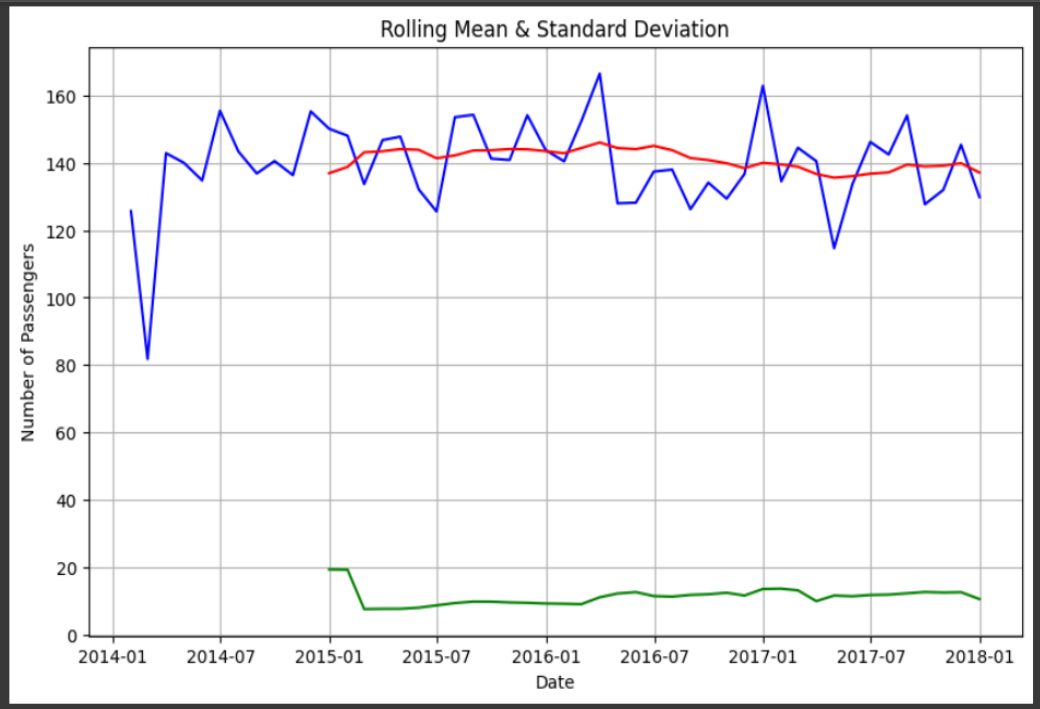
step 2: Stationarize the series

```
[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)
```

bi version 3.ipynb ☆

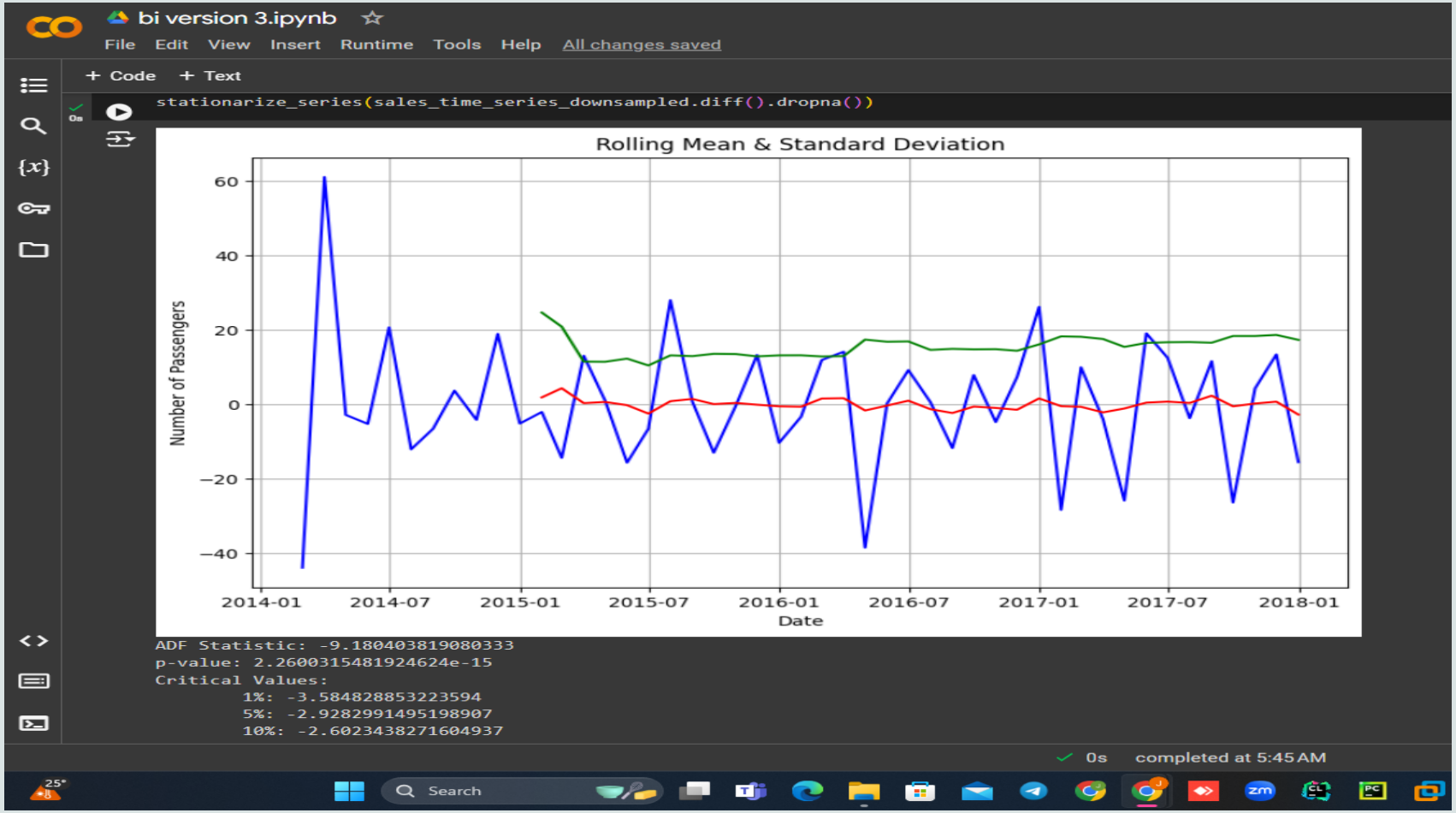
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

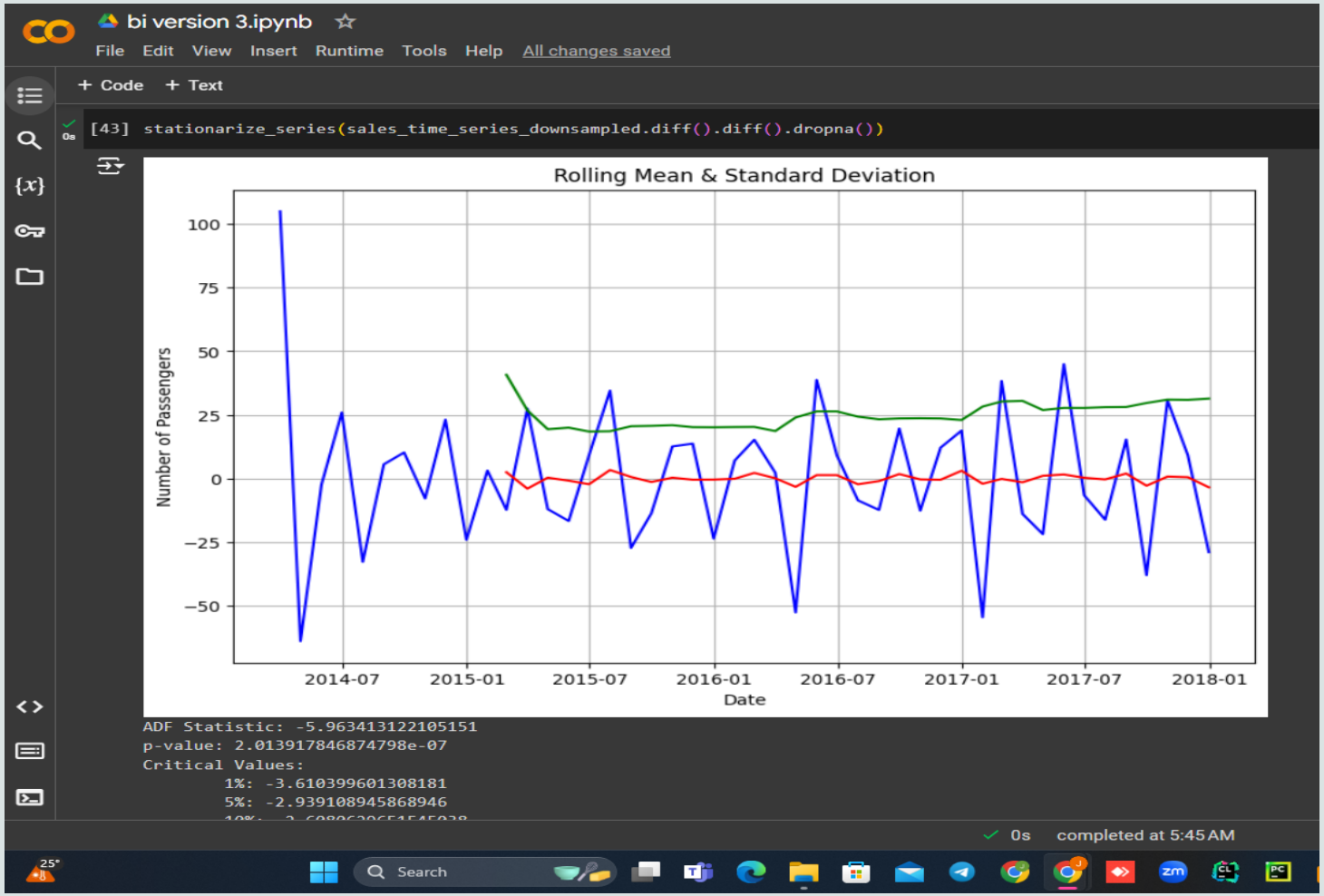



```
ADF Statistic: -7.040810308170457
p-value: 5.853489199192222e-10
Critical Values:
        1%: -3.5812576580093696
        5%: -2.9267849124681518
        10%: -2.6015409829867675
```

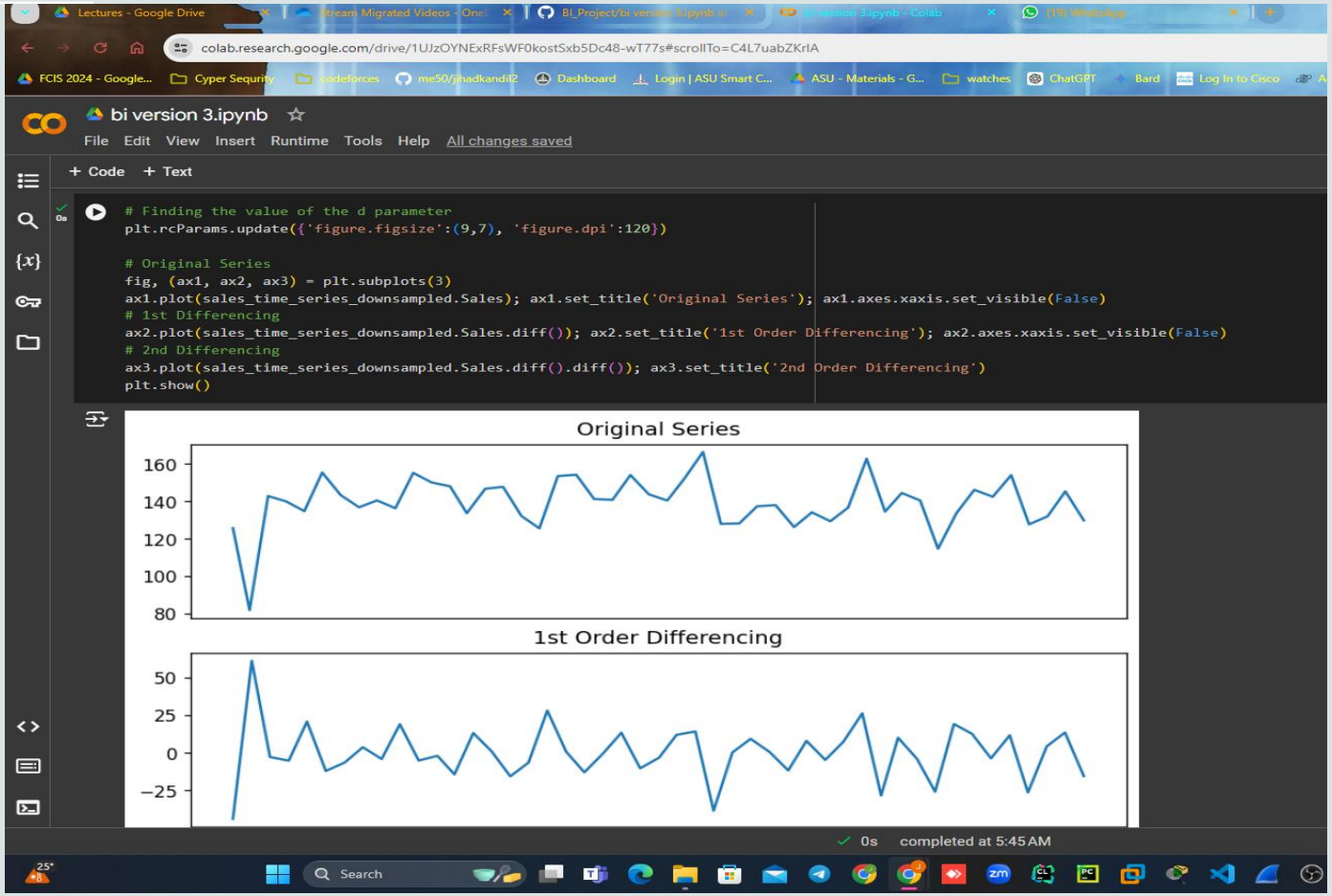
Time series visualization cont..

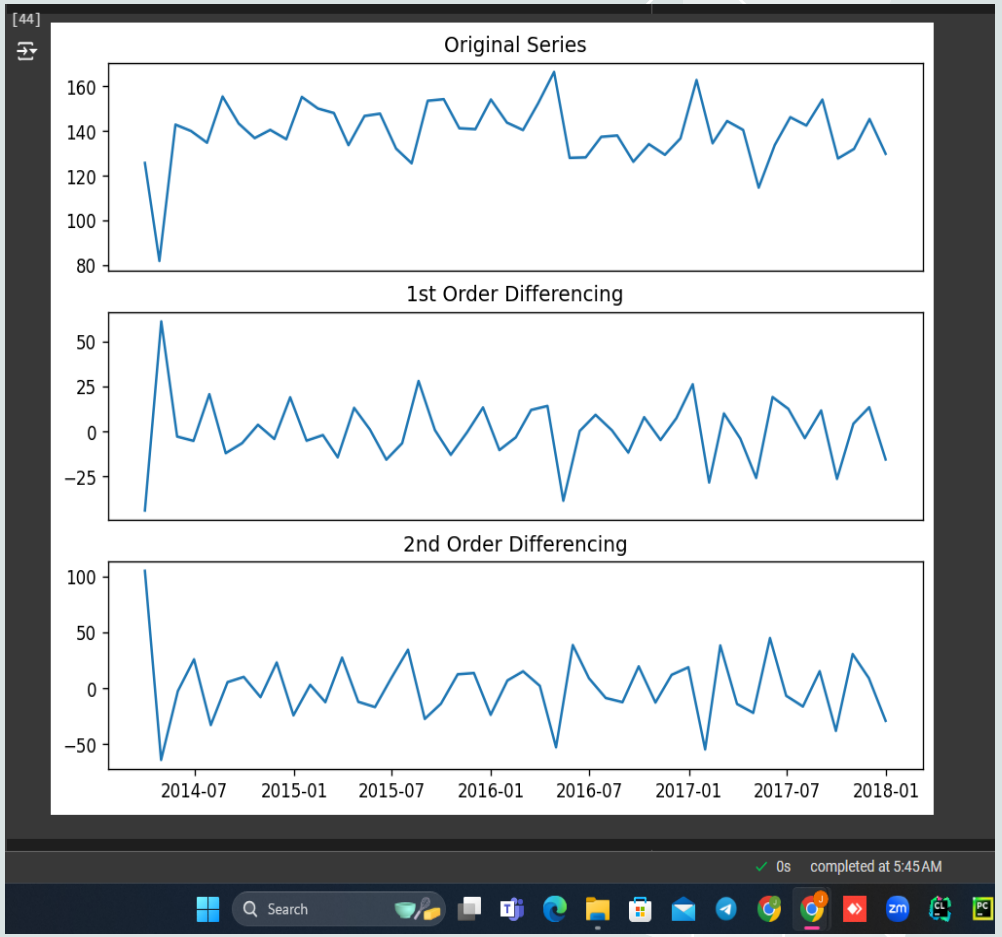


Time series visualization cont..

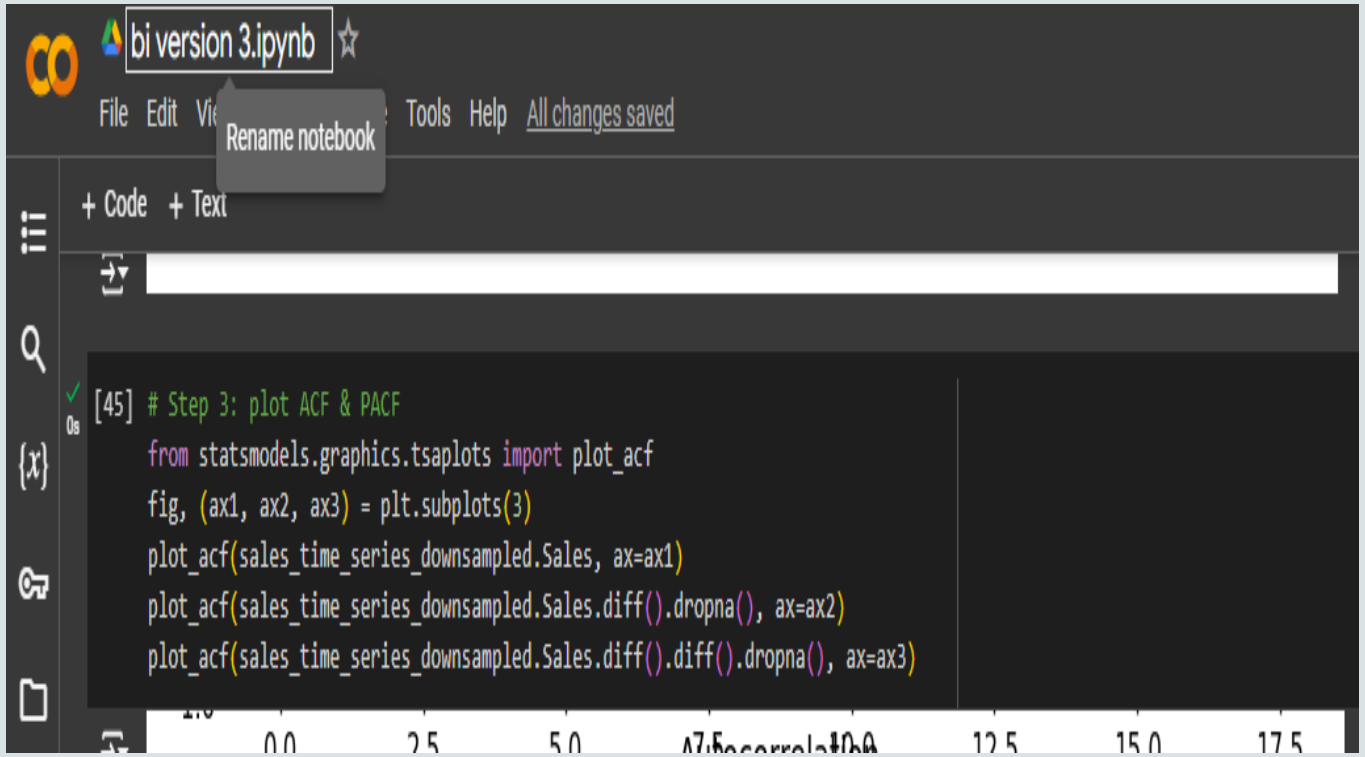


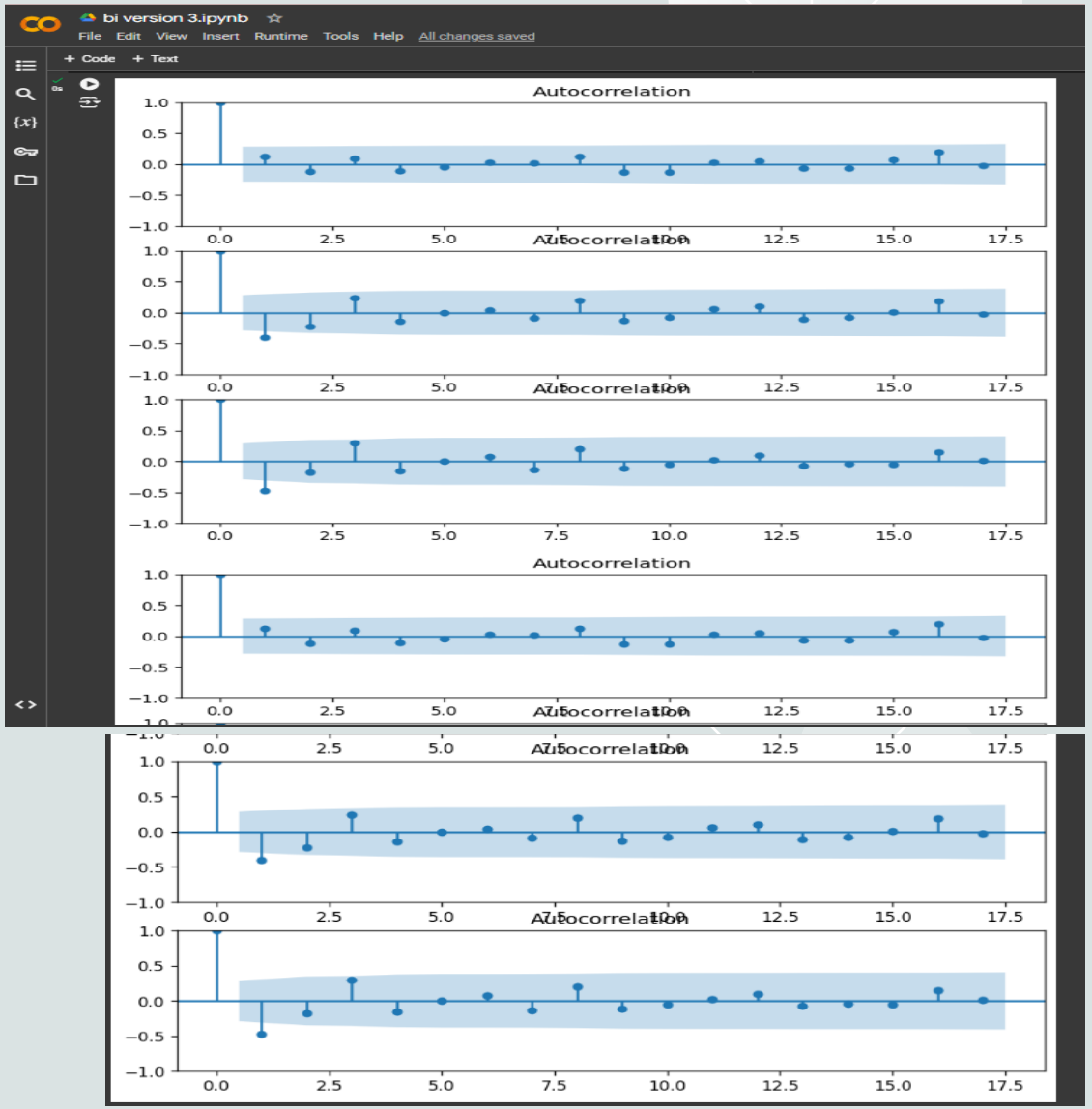
Time series visualization cont..



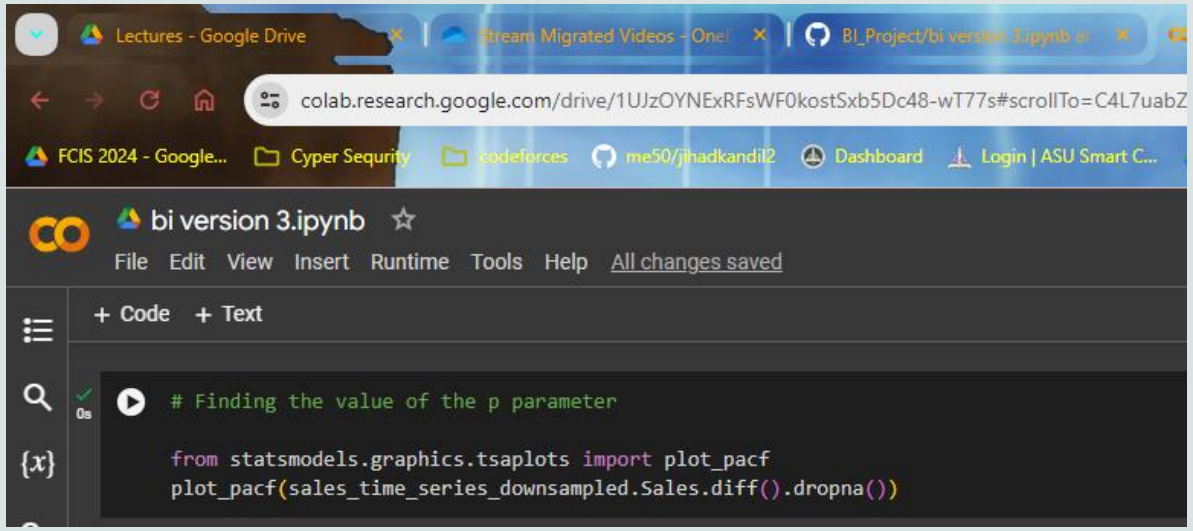


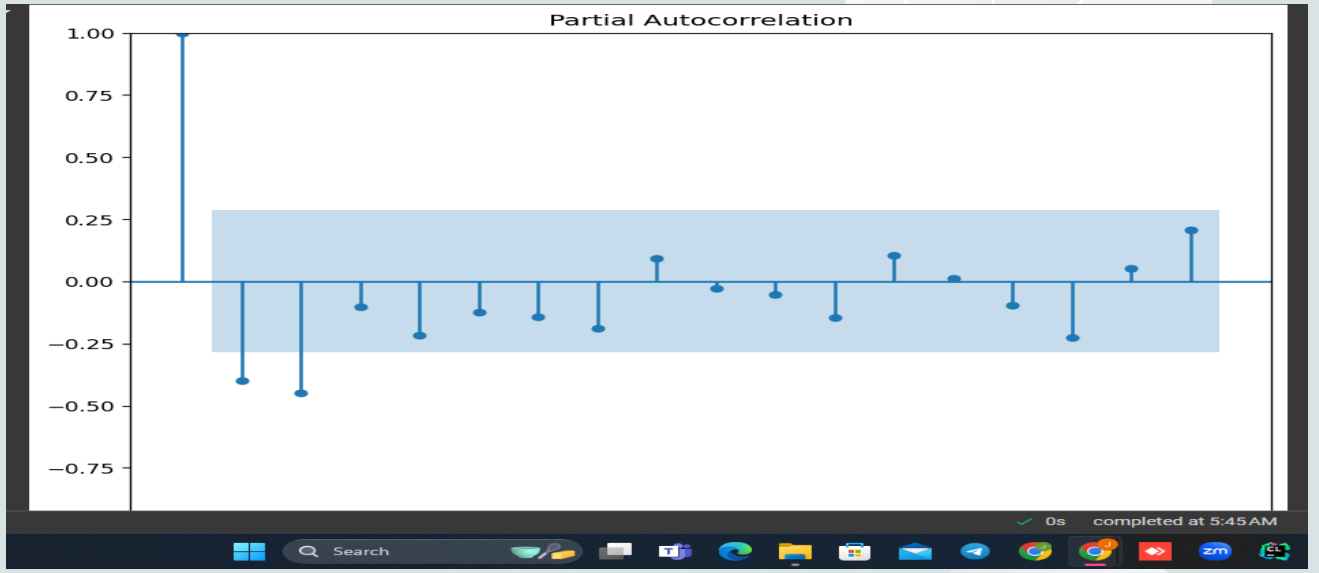
Time series visualization cont..

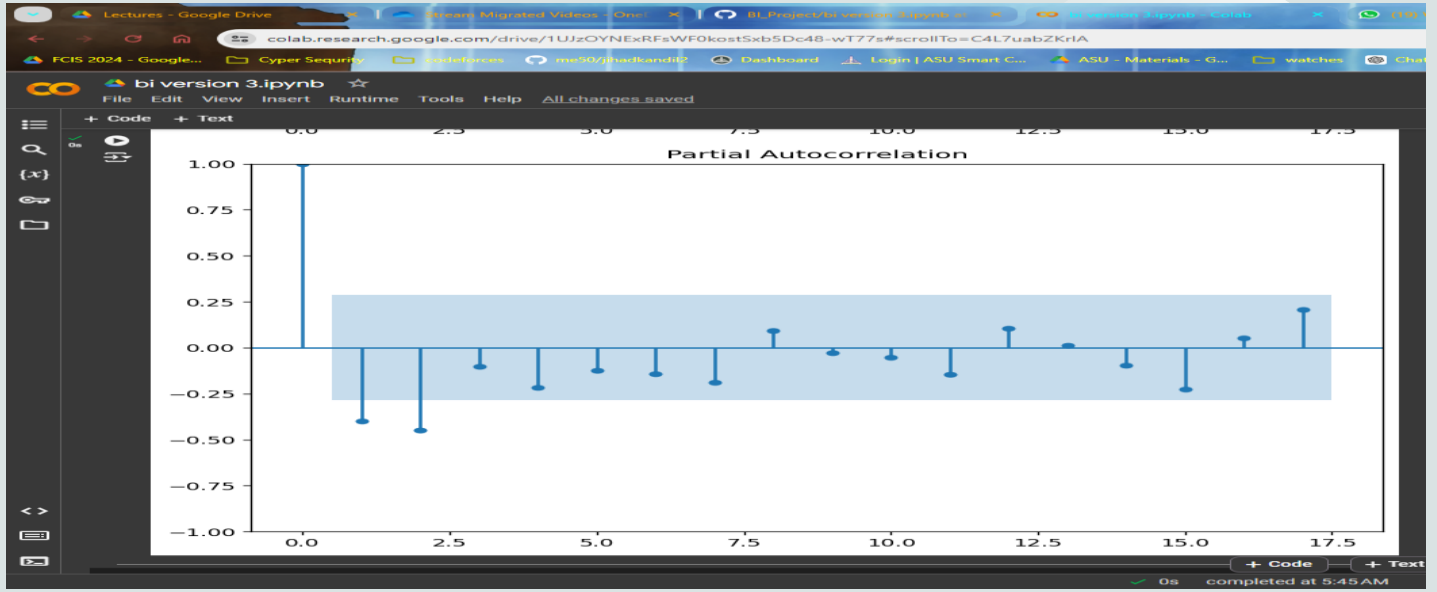




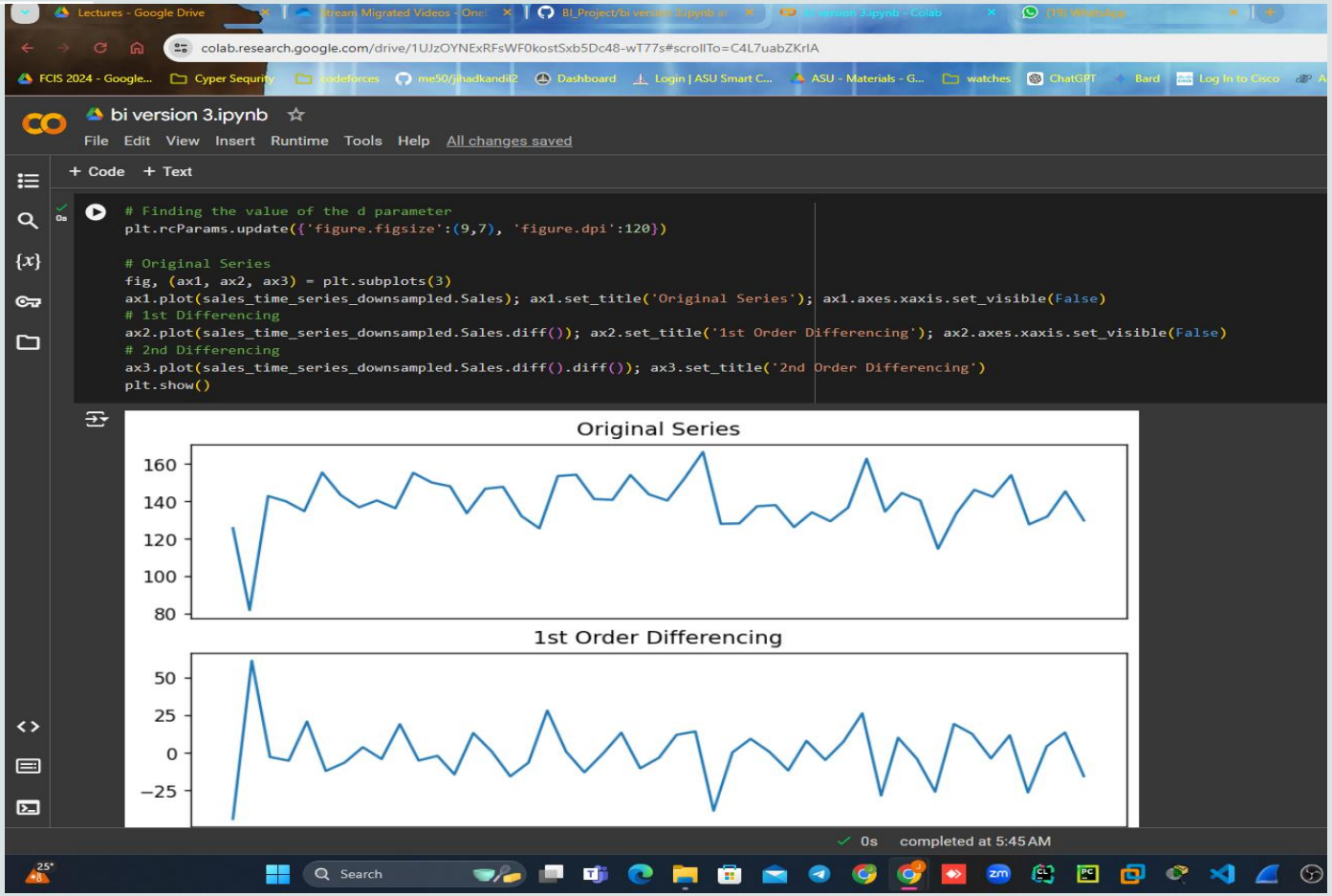
Time series visualization cont..

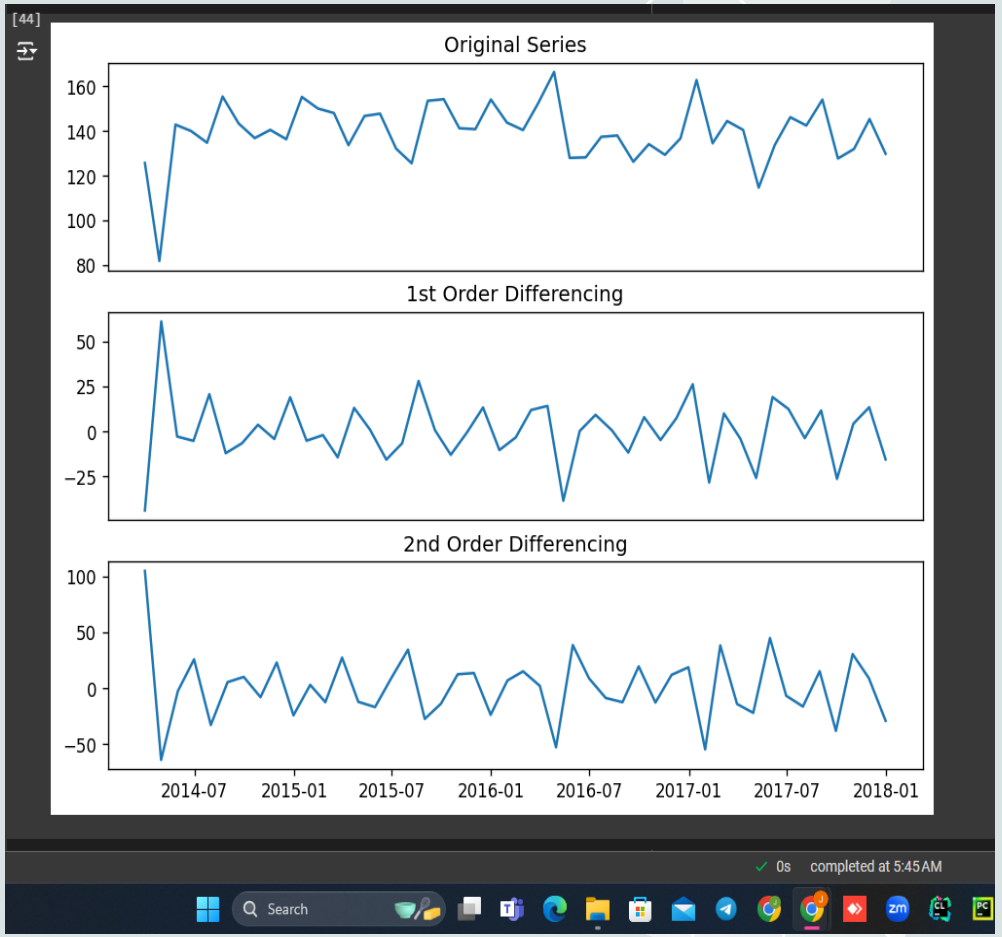






Time series visualization cont..





Time series visualization cont..

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

     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).
```

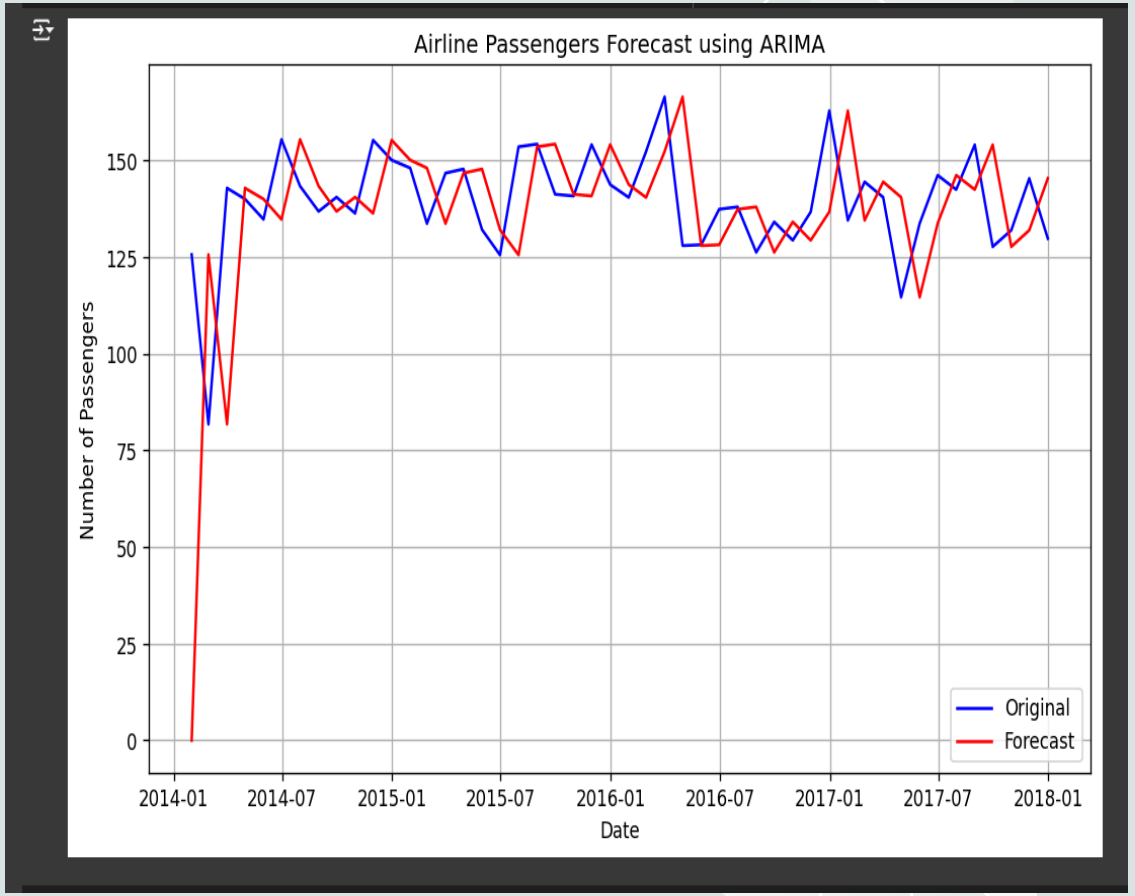
Time series visualization cont..

```
[51] from statsmodels.tsa.arima_model import ARIMA
```

arima model

```
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..

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

Order Date	Sales
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

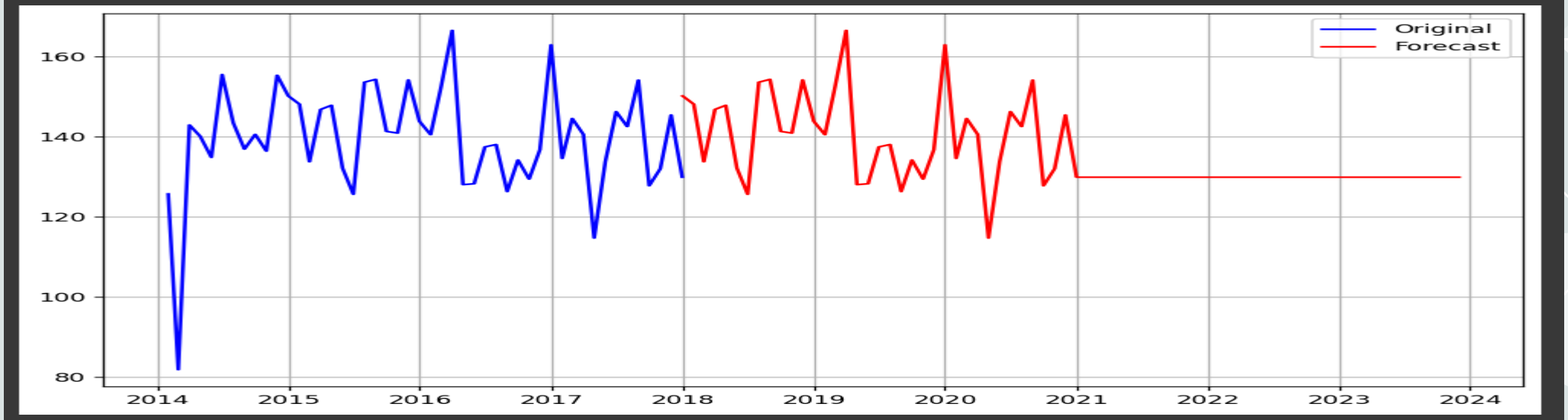
```
[55] sales_time_series_downsampled.tail()
```

Order Date	Sales
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..

```
[56] start_index = '2015-1-31'
     end_index = '2020-12-1'
     forecast = arima_model.predict(start=start_index, end=end_index , dynamic=False)
```

```
# 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(pd.date_range(start=sales_time_series_downsampled.index[-1], periods=len(forecast), freq='MS'), forecast, label='Forecast', color='red')
plt.legend(loc='best')
plt.grid(True)
plt.show()
```



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.