DATS_6450_15

TIME SERIES MODELING & ANALYSIS

Instructor Name: Reza Jafri

Lab#: 1

Submitted by: Dinesh Kumar Padmanabhan

Date: 16-sept-2020

ABSTRACT

In LAB #1, we practiced how to read a time series data which relates to the quarterly sales for a small company over period 1981-2005. from a .csv file using Panda library and plot the dataset Sales, AdBudget and GPD against time. Using Numpy we found the descriptive statistics and plotted against Time and performed the Augmented Dickey-Fuller test in python using statsmodel library a statistical test to inform the degree to which a null hypothesis can be rejected or fail to be rejected meaning non-stationary or stationary.

INTRODUCTION

When performing time series analysis, most statistical forecasting methods assume that the time series Is approximately stationary. How can you determine if a time series is stationary? The Augmented Dickey Fuller Test is a well-known statistical test that can help determine if your time series is stationary. In this lab , it will be demonstrated as how to perform the Augmented Dickey-Fuller Test (ADF) test in python and prove that the given dataset is stationary.

The Null and Alternate hypothesis of the Augmented Dickey-Fuller test is defined as follows:

- Null Hypothesis**(H0)** - states there is the presence of a unit root.
- Alternate Hypothesis**(H1)** - states there is no unit root. In other words, Stationarity exists.

In a stationary time, series, statistical properties such as mean and variance are constant over time. In a non-stationary series, these properties are dependent on time. Here is a data set (tute1) of a Stationary time series where there is no visible trend (see below fig1).

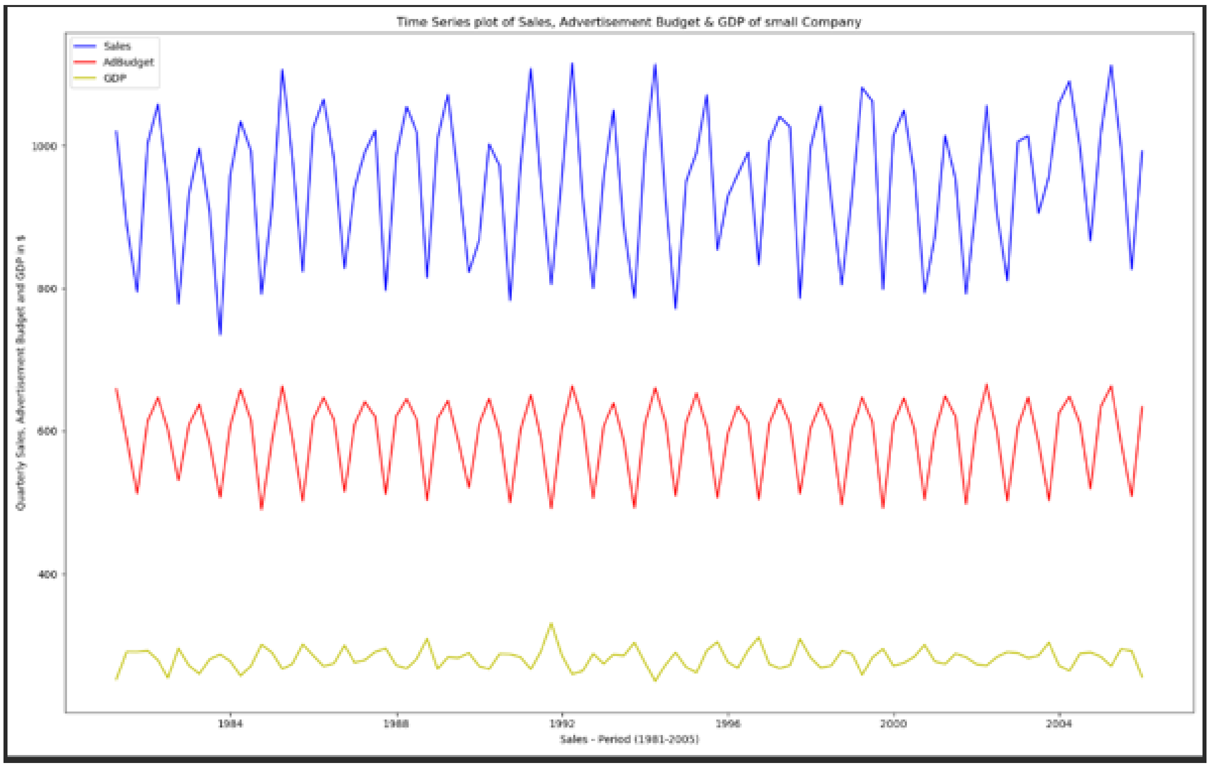


Fig1: Tute1 Dataset of sales, advertisement budget & GDP of a company are constant

METHOD, THEORY & PROCEDURES

Method:

1. Programming Language: Python

Libraries used: Some basic libraries used for analysis & model building are mentioned below

- *library(Numpy)* - large collection of high-level mathematical functions to operate on these arrays.
- *library (Pandas)* – For Data manipulation and analysis
- *library(Matplotlib)* – is a system for declaratively creating graphics

Theory:

To investigate the data set and Prove that the Sales, AdBudget and GDP in this time series dataset is Stationary.

Procedure:

I shall be looking at all variables through some plots and infer about it in my exploratory analysis. And through my exploration I shall try to identify the Variables that tend to have an impact in the sales and use it to test hypotheses and draw inferences.

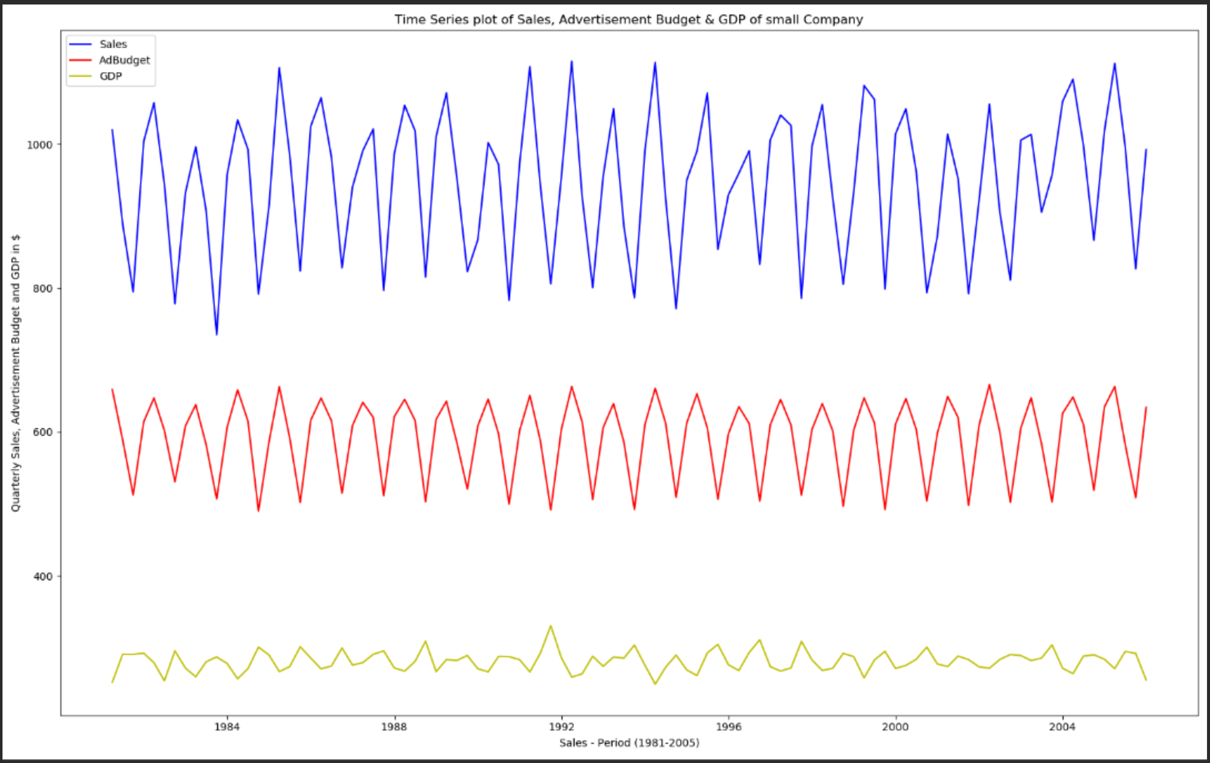
The Dataset will be explored in following stages:

1. **Hypothesis Generation** – understanding the problem statement better by working out possible factors that can have impact on the outcome
2. **Data Exploration (EDA)** – looking at continuous variables summaries and making inferences about the data.
3. **Data Visualization** – Plotting Time series graphs.
4. **Diagnostic Testing** – Running ADF test on data set.

ANSWERS TO ASKED QUESTIONS

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.stattools import adfuller

#%%================================================================================# 1:

Load the time series data called tute1.

# %%---------------------------------------------------------------------------


q_sales = pd.read_csv("tute1.csv")

print("Shape of Tute1:",q_sales.shape)

print('\n')



#%%================================================================================# 2:

This date relates to the quarterly sales for a small company over period 1981-2005.

# %%---------------------------------------------------------------------------


# print(q_sales.columns)

for i in q_sales.columns:

    print(i)

print('\n')

date_rng = pd.date_range(start='3/1/1981', end='3/1/2006', freq='Q')

print(date_rng)
```

```
#%%===========================================================================

# 3: Sales contains the quarterly sales, AdBudget is the advertisement budget and GPD is the gross

domestic product for a small company.

# %%-----------------------------------------------------------------------------


q_sales.rename( columns={'Unnamed: 0':'Q-SalesDate'}, inplace=True )

print(q_sales.head(5))

print('\n')

print(q_sales.tail(5))

print('\n')

print("DataType of Tute1:",q_sales.dtypes)



#%%===========================================================================

# 4: Plot Sales, AdBudget and GPD versus time step.

# %%-----------------------------------------------------------------------------


plt.figure(figsize=(16,10))

plt.plot(date_rng, q_sales['Sales'], 'b-', label = 'Sales')

plt.plot(date_rng, q_sales['AdBudget'], 'r-', label = 'AdBudget')

plt.plot(date_rng, q_sales['GDP'], 'y-', label = 'GDP')

plt.xlabel('Sales - Period (1981-2005)')

plt.ylabel('Quarterly Sales, Advertisement Budget and GDP in $')

plt.title('Time Series plot of Sales, Advertisement Budget & GDP of small Company')

plt.legend(loc='best')
plt.show()
```

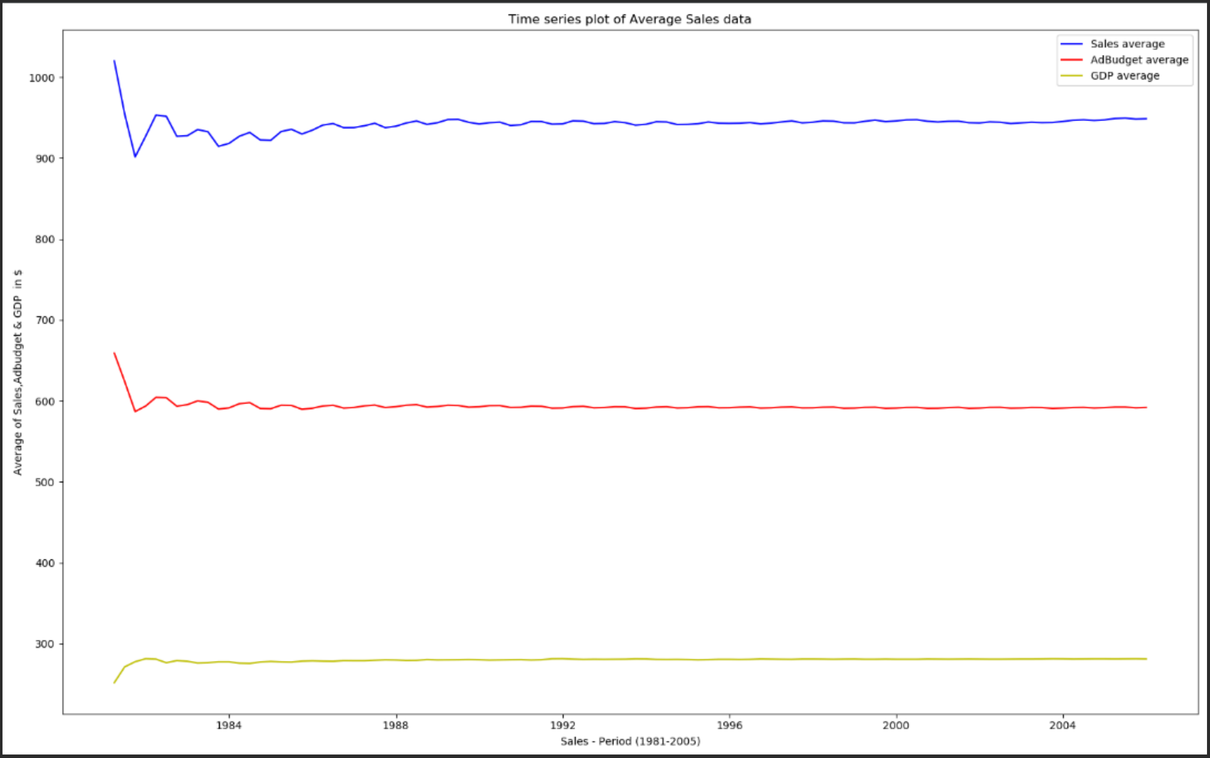


```
#%%===========================================================================
# 5: Find the time series statistics (average, variance and standard deviation) of Sales, AdBudget and GPD.
# %%-----------------------------------------------------------------------------

# print("MEAN")
m_sales = np.round(np.mean(q_sales['Sales']))
#print(m_sales)
m_AdBudget = np.round(np.mean(q_sales['AdBudget']))
#print(m_AdBudget)
m_GDP = np.round(np.mean(q_sales['GDP']))
#print(m_GDP)
# print("\n")


# print("VARIANCE")
```

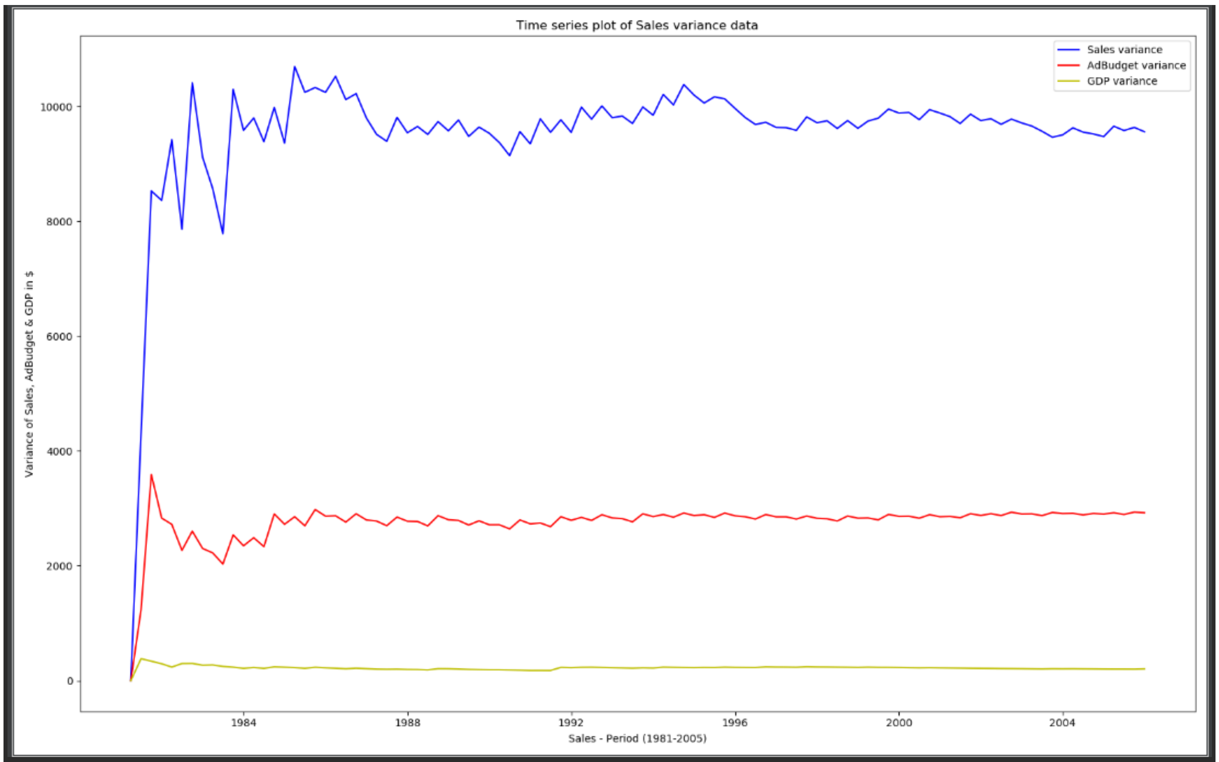
```
var_sales = np.round(np.var(q_sales['Sales']))

#print(var_sales)

var_adbudget = np.round(np.var(q_sales['AdBudget']))

#print(var_adbudget)

var_gdp = np.round(np.var(q_sales['GDP']))

#print(var_gdp)

# print("\n")



# print("STANDARD DEVIATION")

std_sales = np.round(np.std(q_sales['Sales']))

#print(std_sales)

std_adbudget = np.round(np.std(q_sales['AdBudget']))

#print(std_adbudget)

std_gdp = np.round(np.std(q_sales['GDP']))

#print(std_gdp)


#%%=================================================================================

# 6: Display the Average, variance, and standard deviation as follow:

# a. The Sales mean is : -------- and the variance is : -------- with standard deviation : -------

# b. The AdBudget mean is : -------- and the variance is : -------- with standard deviation : -----

# c. The GDP mean is : -------- and the variance is : -------- with standard deviation : -------

# %%------------------------------------------------------------------------------
```

```
print("The Sales mean is : {0} and the variance is : {1} with standard deviation :{2}"

     .format(m_sales,var_sales,std_sales))

print("The AdBudget mean is : {0} and the variance is : {1} with standard deviation :{2}"

     .format(m_AdBudget,var_adbudget,std_adbudget))

print("The GDP mean is : {0} and the variance is : {1} with standard deviation : {2}"

     .format(m_GDP,var_gdp,std_gdp))


#%%===========================================================================

# 7: Prove that the Sales, AdBudget and GDP in this time series dataset is stationary.

# Hint: To show a process is stationary, you need to show that data statistics is not changing by time.

# You need to create 100 sub-sequences from the original sequence and save the average and variance of

each sub-sequence. Plot all means and variances and show that the means and variances are almost

constant. To create sub-sequences, start with a sequence with the first sales data and find the mean.

# Then create another sub-sequence by adding the second sales date to the first sub-sequence,

# then find the corresponding mean. Repeat this process till you added the last sales date to the last sub-

sequence and  find the average. Repeat the same procedures for variances.

# Hint: Create a loop for the length of the dataset and use the following command to bring new data

sample at  each iteration: pd.read_csv('tute1.csv').head(i)# where is the number of samples

# %%---------------------------------------------------------------------------


Sales_avg, AdBudget_avg, GDP_avg, Sales_var, AdBudget_var, GDP_var = [], [], [], [], [], []

for i in range(1,101):

   k = pd.read_csv('tute1.csv').head(i)

   Sales_avg.append(np.round(np.mean(k['Sales']),1))

   AdBudget_avg.append(np.round(np.mean(k['AdBudget']),1))

   GDP_avg.append(np.round(np.mean(k['GDP']),1))

   Sales_var.append(np.round(np.var(k['Sales']), 1))
```

```
    AdBudget_var.append(np.round(np.var(k['AdBudget']), 1))

    GDP_var.append(np.round(np.var(k['GDP']), 1))


plt.figure(figsize=(16,10))

plt.plot(date_rng, Sales_avg,'b-',label = 'Sales average')

plt.plot(date_rng, AdBudget_avg,'r-',label = 'AdBudget average')

plt.plot(date_rng, GDP_avg, 'y-',label = 'GDP average')

plt.xlabel("Sales - Period (1981-2005)")

plt.ylabel("Average of Sales,Adbudget & GDP  in $")

plt.title('Time series plot of Average Sales data')

plt.legend(loc='best')

plt.show()
```



```
#%%=================================================================================

# 8:Plot all average and variance.

# Write down your observation about if this time series date is stationary or not? Why?

# %%---------------------------------------------------------------------------


plt.figure(figsize=(16,10))

plt.plot(date_rng, Sales_var,'b-',label = 'Sales variance')

plt.plot(date_rng, AdBudget_var,'r-',label = 'AdBudget variance')

plt.plot(date_rng, GDP_var, 'y-',label = 'GDP variance')

plt.xlabel("Sales - Period (1981-2005)")

plt.ylabel("Variance of Sales, AdBudget & GDP in $")

plt.title('Time series plot of Sales variance data')

plt.legend(loc='best')

plt.show()

Yes this is a stationary time series, statistical properties such as mean and variance are constant over

time
```



```
#%%================================================================================
# 9: Perform an ADF-test to check if the Sales, AdBudget and GDP stationary or not (confidence interval
95% or above). # Does your answer for this question reinforce your observations in the previous step?
#%%-----------------------------------------------------------------------

def ADFcal(x):
    result = adfuller(x)
    print("ADF Statistic: %f" %result[0])
    print("p-value: %f" %result[1])
    print("Critical values :")
    for key,value in result[4].items():
        print("\t%s: %.3f" % (key, value))
    print()
print()
print("#### ADF test on Sales ####")
```

```
ADFcal(q_sales['Sales'])


print("#### ADF test on AdBudget ####")

ADFcal(q_sales['AdBudget'])


print("#### ADF test on GDP ####")

ADFcal(q_sales['GDP'])
```

Yes it does. The results show that series is actually stationary. In this case, the p-value from ADF test is much smaller than our 5% significance level, therefore we can reject the null hypothesis and instead accept the alternate hypothesis that stationary exists.

```
#%%==========================================================================================#

10: Add an appropriate x-label, y-label, legend, and title to each graph.

# %%---------------------------------------------------------------------------------

For each of the above plots x-label, y-label, legend and title are added appropriately.
```

CONCLUSION

The results show that the series is actually stationary. In this case, the P-Value from our ADF test is much smaller than our 5% significance level, therefore we can reject the Null hypothesis and instead accept the alternate hypothesis that stationarity exists. Taking a look at the critical value yields the same conclusion. The tests critical value ends up being **-3.26** which is much smaller than the 5% critical value of **-3.5** for Sales so is for AdBudget and GDP as shown in below table. so, we have enough evidence to conclude that unit root does not exist. In other words, the series is stationary.

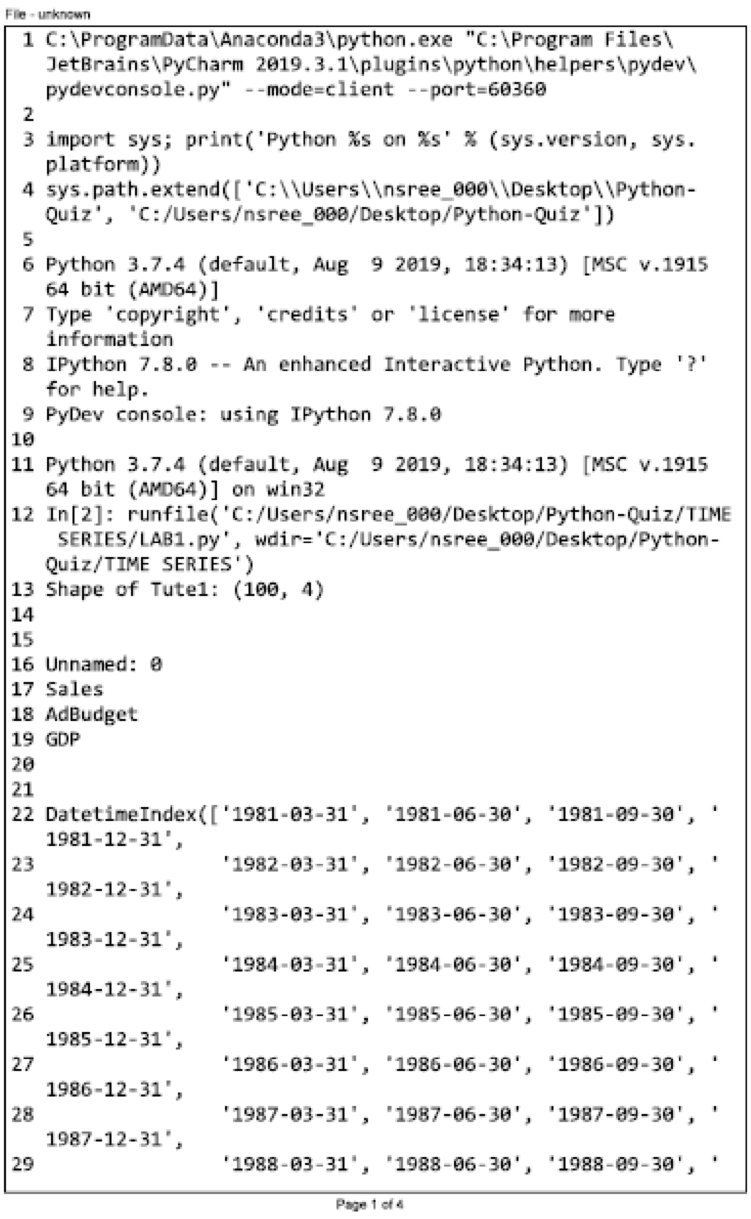
ADF TEST		
Sales	AdBudget	GDP
ADF Statistic: -3.262755	ADF Statistic: -2.758605	ADF Statistic: -3.227577
p-value: 0.016628	p-value: 0.064434	p-value: 0.018443
Critical values :	Critical values :	Critical values :
1%: -3.505	1%: -3.504	1%: -3.504
5%: -2.894	5%: -2.894	5%: -2.894
10%: -2.584	10%: -2.584	10%: -2.584

We have learned how to test for stationarity using the **Augmented Dickey-Fuller Test** (ADF) and are able to **interpret** the test using the **P- Value** or the **Critical Values** returned by the test. We created our own function which implements the ADF test from the statsmodels python package. Knowledge of this statistical test will greatly help when we are building time series forecasting models in which stationarity is many times, a strong underlying assumption for various models.

CHALLENGE

The dataset given to us is not formatted especially the dates. Hence the challenge faced is to format

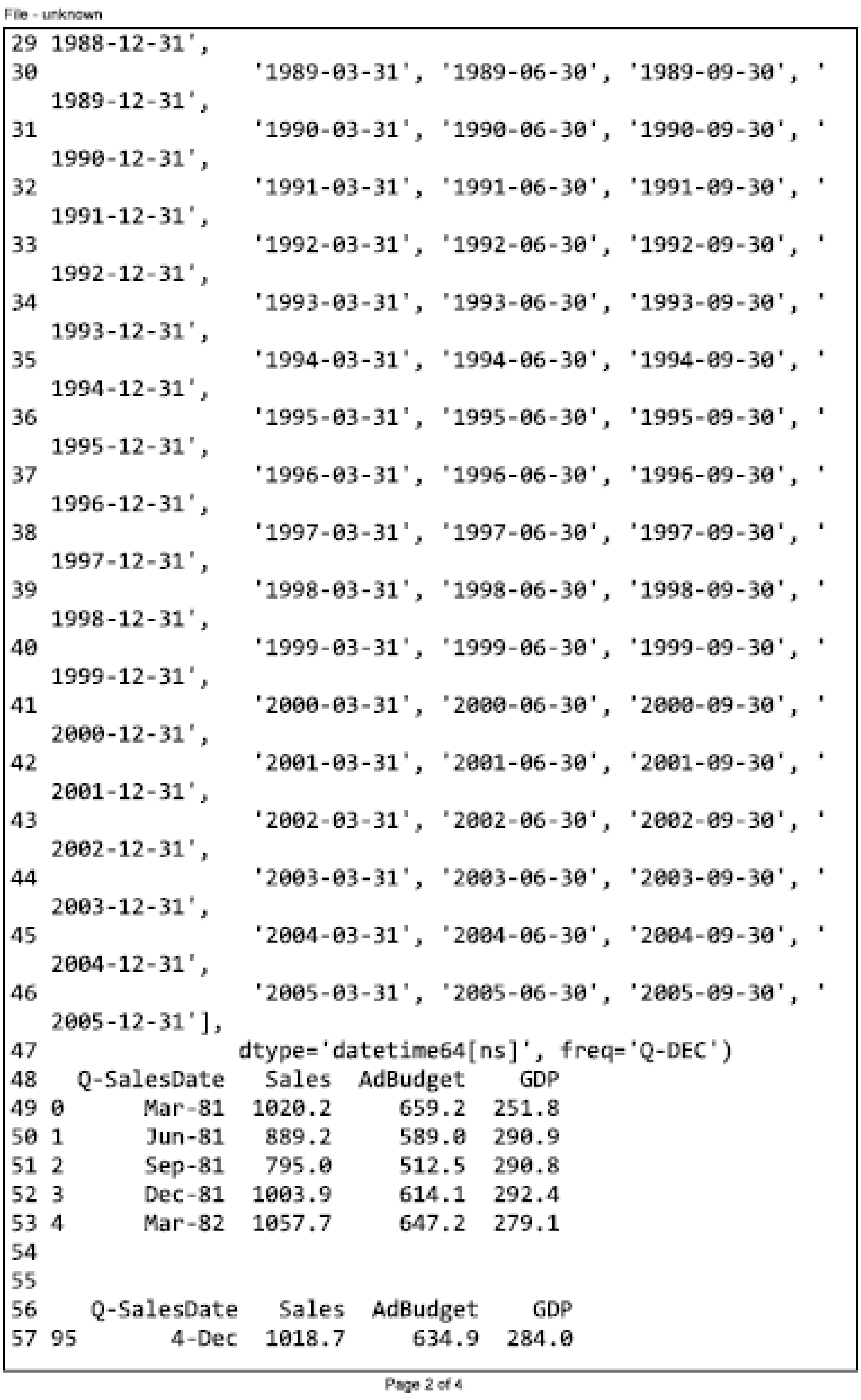
APPENDIX



File - unknown

```
C:\ProgramData\Anaconda3\python.exe "C:\Program Files\JetBrains\PyCharm 2019.3.1\plugins\python\helpers\pydev\pydevconsole.py" --mode=client --port=60360

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['C:\\Users\\nsree_000\\Desktop\\Python-Quiz', 'C:/Users/nsree_000/Desktop/Python-Quiz'])

Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.8.0 -- An enhanced Interactive Python. Type '?' for help.
PyDev console: using IPython 7.8.0

Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] on win32
In[2]: runfile('C:/Users/nsree_000/Desktop/Python-Quiz/TIME SERIES/LAB1.py', wdir='C:/Users/nsree_000/Desktop/Python-Quiz/TIME SERIES')
Shape of Tute1: (100, 4)


Unnamed: 0
Sales
AdBudget
GDP


DatetimeIndex(['1981-03-31', '1981-06-30', '1981-09-30', '1981-12-31',
               '1982-03-31', '1982-06-30', '1982-09-30', '1982-12-31',
               '1983-03-31', '1983-06-30', '1983-09-30', '1983-12-31',
               '1984-03-31', '1984-06-30', '1984-09-30', '1984-12-31',
               '1985-03-31', '1985-06-30', '1985-09-30', '1985-12-31',
               '1986-03-31', '1986-06-30', '1986-09-30', '1986-12-31',
               '1987-03-31', '1987-06-30', '1987-09-30', '1987-12-31',
               '1988-03-31', '1988-06-30', '1988-09-30', '
```

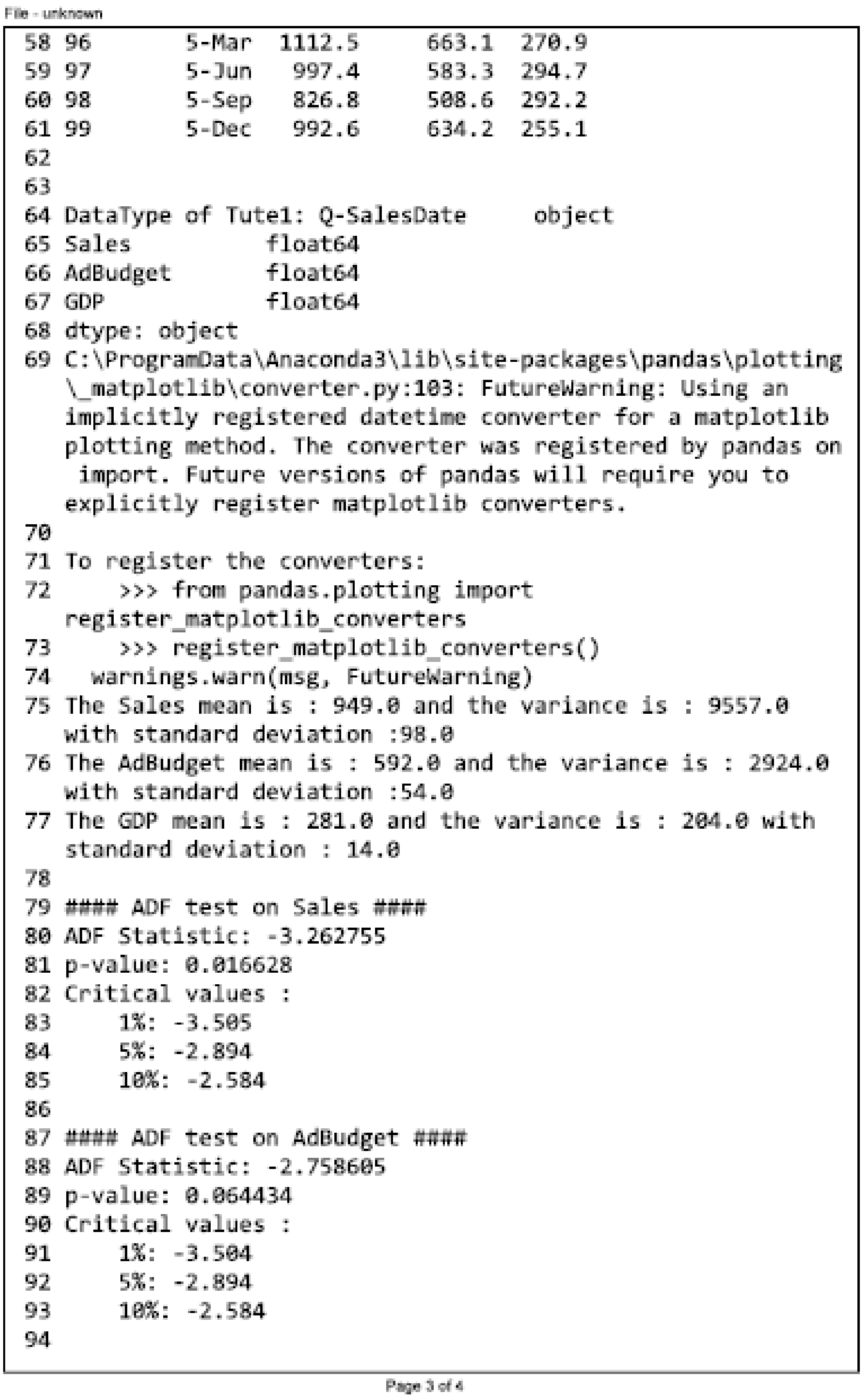
Page 1 of 4



File - unknown

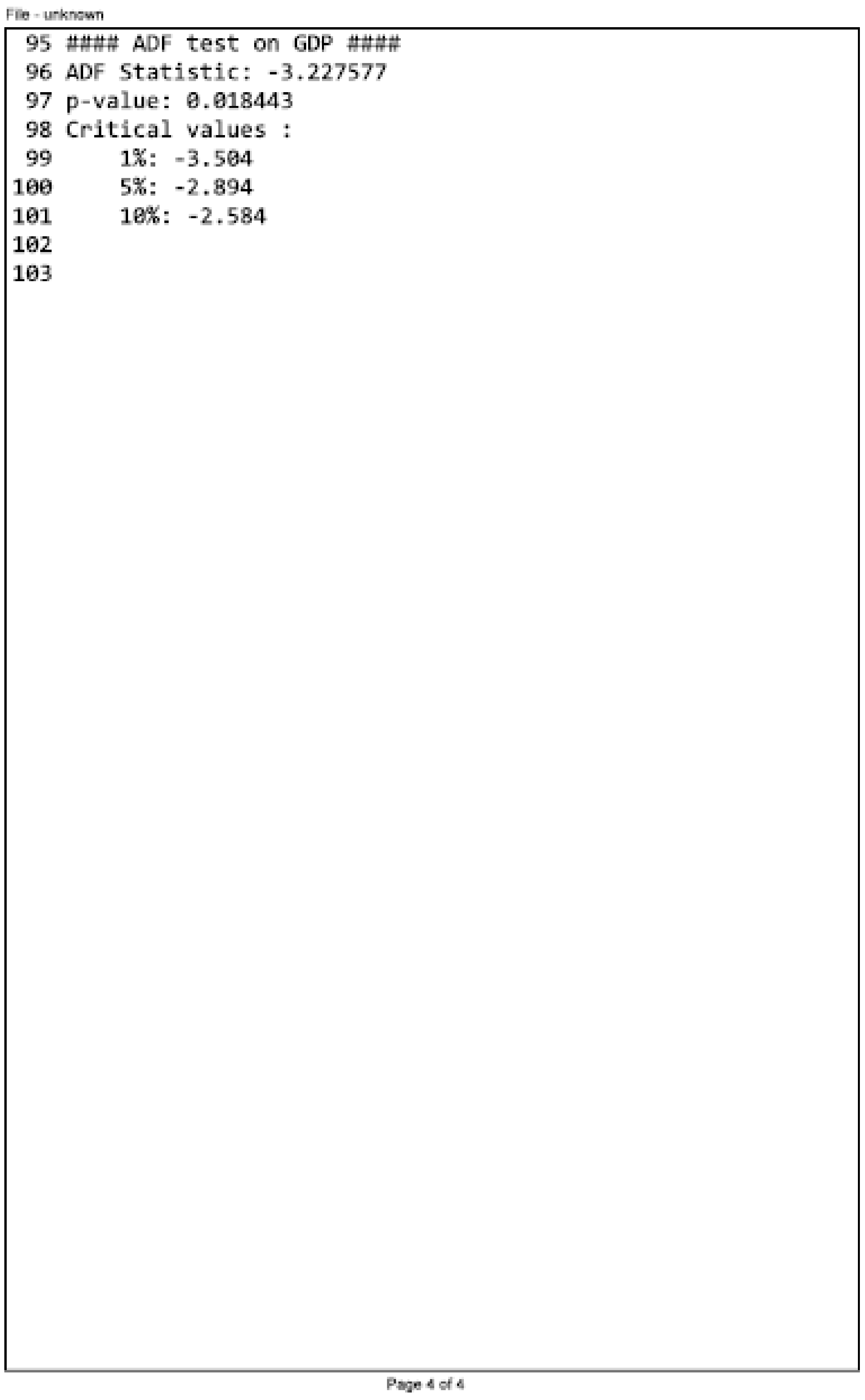
```
1988-12-31',
                  '1989-03-31', '1989-06-30', '1989-09-30', '1989-12-31',
                  '1990-03-31', '1990-06-30', '1990-09-30', '1990-12-31',
                  '1991-03-31', '1991-06-30', '1991-09-30', '1991-12-31',
                  '1992-03-31', '1992-06-30', '1992-09-30', '1992-12-31',
                  '1993-03-31', '1993-06-30', '1993-09-30', '1993-12-31',
                  '1994-03-31', '1994-06-30', '1994-09-30', '1994-12-31',
                  '1995-03-31', '1995-06-30', '1995-09-30', '1995-12-31',
                  '1996-03-31', '1996-06-30', '1996-09-30', '1996-12-31',
                  '1997-03-31', '1997-06-30', '1997-09-30', '1997-12-31',
                  '1998-03-31', '1998-06-30', '1998-09-30', '1998-12-31',
                  '1999-03-31', '1999-06-30', '1999-09-30', '1999-12-31',
                  '2000-03-31', '2000-06-30', '2000-09-30', '2000-12-31',
                  '2001-03-31', '2001-06-30', '2001-09-30', '2001-12-31',
                  '2002-03-31', '2002-06-30', '2002-09-30', '2002-12-31',
                  '2003-03-31', '2003-06-30', '2003-09-30', '2003-12-31',
                  '2004-03-31', '2004-06-30', '2004-09-30', '2004-12-31',
                  '2005-03-31', '2005-06-30', '2005-09-30', '2005-12-31'],
                 dtype='datetime64[ns]', freq='Q-DEC')
  Q-SalesDate   Sales  AdBudget    GDP
0      Mar-81  1020.2     659.2  251.8
1      Jun-81   889.2     589.0  290.9
2      Sep-81   795.0     512.5  290.8
3      Dec-81  1003.9     614.1  292.4
4      Mar-82  1057.7     647.2  279.1


   Q-SalesDate   Sales  AdBudget    GDP
95       4-Dec  1018.7     634.9  284.0
```

Page 2 of 4


File - unknown

```
96       5-Mar  1112.5     663.1  270.9
97       5-Jun   997.4     583.3  294.7
98       5-Sep   826.8     508.6  292.2
99       5-Dec   992.6     634.2  255.1


DataType of Tute1: Q-SalesDate     object
Sales          float64
AdBudget       float64
GDP            float64
dtype: object
C:\ProgramData\Anaconda3\lib\site-packages\pandas\plotting\_matplotlib\converter.py:103: FutureWarning: Using an implicitly registered datetime converter for a matplotlib plotting method. The converter was registered by pandas on import. Future versions of pandas will require you to explicitly register matplotlib converters.

To register the converters:
    >>> from pandas.plotting import register_matplotlib_converters
    >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
The Sales mean is : 949.0 and the variance is : 9557.0 with standard deviation :98.0
The AdBudget mean is : 592.0 and the variance is : 2924.0 with standard deviation :54.0
The GDP mean is : 281.0 and the variance is : 204.0 with standard deviation : 14.0

#### ADF test on Sales ####
ADF Statistic: -3.262755
p-value: 0.016628
Critical values :
    1%: -3.505
    5%: -2.894
    10%: -2.584

#### ADF test on AdBudget ####
ADF Statistic: -2.758605
p-value: 0.064434
Critical values :
    1%: -3.504
    5%: -2.894
    10%: -2.584

```

Page 3 of 4



File - unknown

```
#### ADF test on GDP ####
ADF Statistic: -3.227577
p-value: 0.018443
Critical values :
    1%: -3.504
    5%: -2.894
    10%: -2.584
```

Page 4 of 4

REFERENCES

https://otexts.com/fpp2/#

https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html