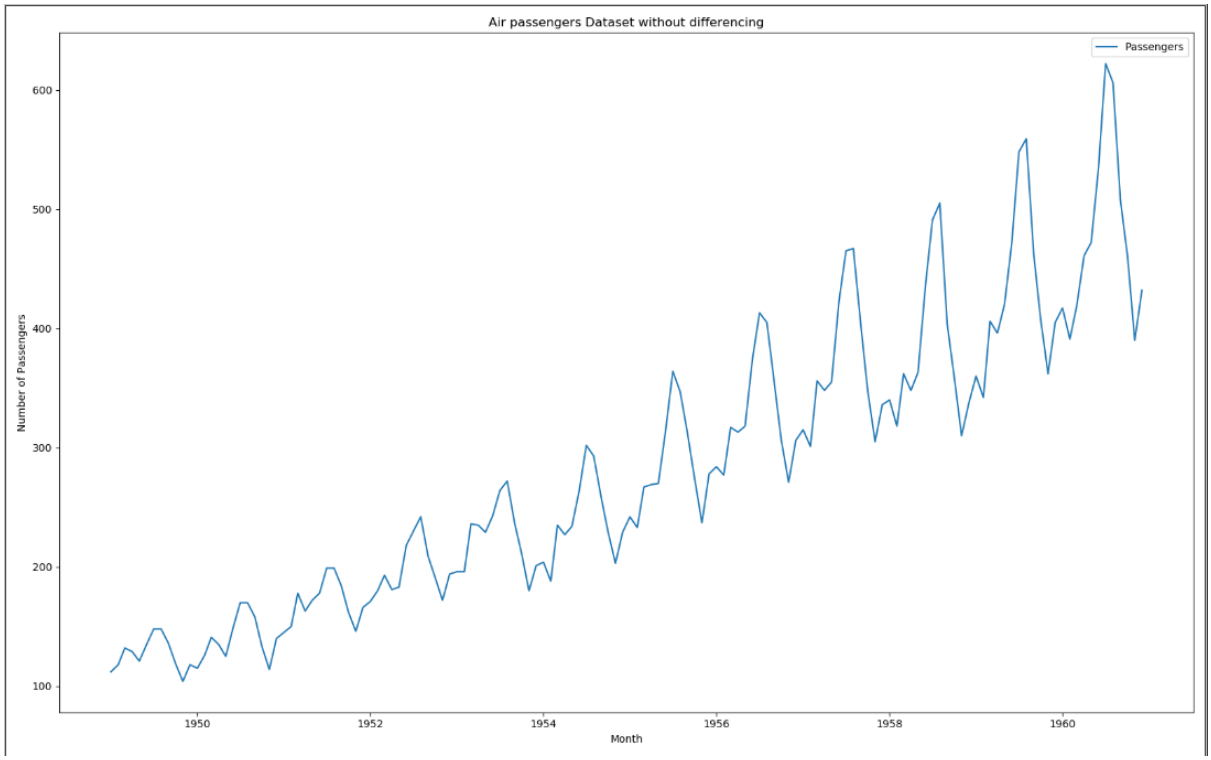
File - unknown

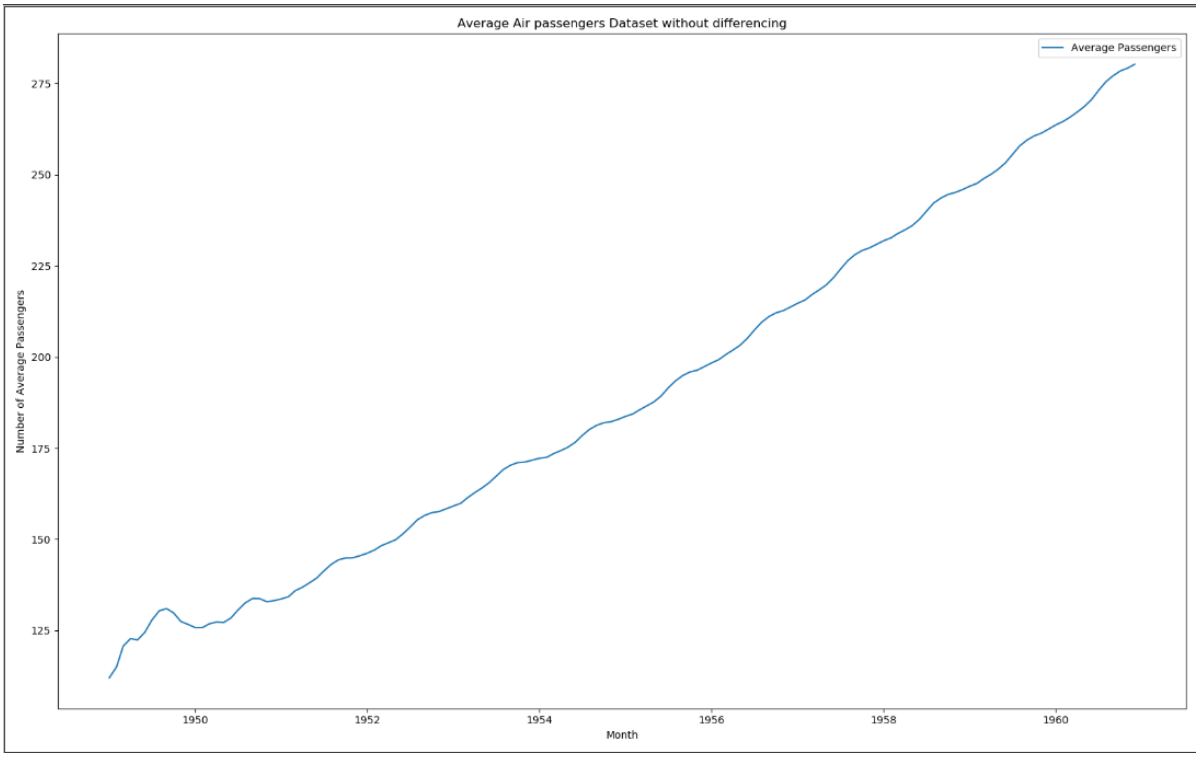
```
C:\ProgramData\Anaconda3\python.exe "C:\Program Files\
JetBrains\PyCharm 2019.3.1\plugins\python\helpers\pydev\
pydevconsole.py" --mode=client --port=54568

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/HW1.py', wdir='C:/Users/nsree_000/Desktop/Python-
Quiz/TIME SERIES')
Shape of AirPassengers: (144, 2)


DataType of AirPassengers: Month          object
#Passengers     int64
dtype: object


Month
#Passengers


        Month  #Passengers
0 1949-01-01          112
1 1949-02-01          118
2 1949-03-01          132
3 1949-04-01          129
4 1949-05-01          121


          Month  #Passengers
139 1960-08-01          606
140 1960-09-01          508
141 1960-10-01          461
```

```
37 142 1960-11-01          390
38 143 1960-12-01          432
39
40
41 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.
42
43 To register the converters:
44     >>> from pandas.plotting import
   register_matplotlib_converters
45     >>> register_matplotlib_converters()
46   warnings.warn(msg, FutureWarning)
47 ADF Statistic: 0.815369
48 p-value: 0.991880
49 Critical values :
50     1%: -3.482
51     5%: -2.884
52     10%: -2.579
53
54 ADF Statistic: -2.829267
55 p-value: 0.054213
56 Critical values :
57     1%: -3.482
58     5%: -2.884
59     10%: -2.579
60
61 ADF Statistic: -2.829267
62 p-value: 0.054213
63 Critical values :
64     1%: -3.482
65     5%: -2.884
66     10%: -2.579
67
68 ADF Statistic: -16.384232
69 p-value: 0.000000
70 Critical values :
71     1%: -3.482
72     5%: -2.884
73     10%: -2.579
74
75
```

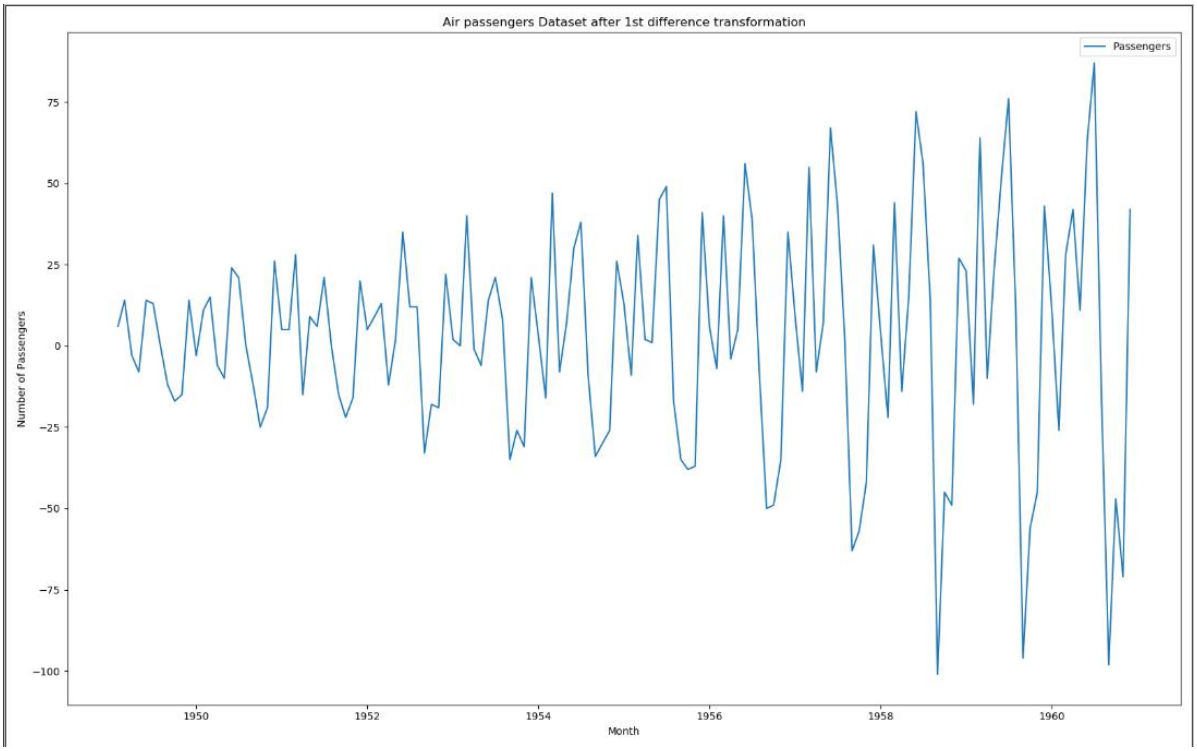
TIME SERIES PLOTS



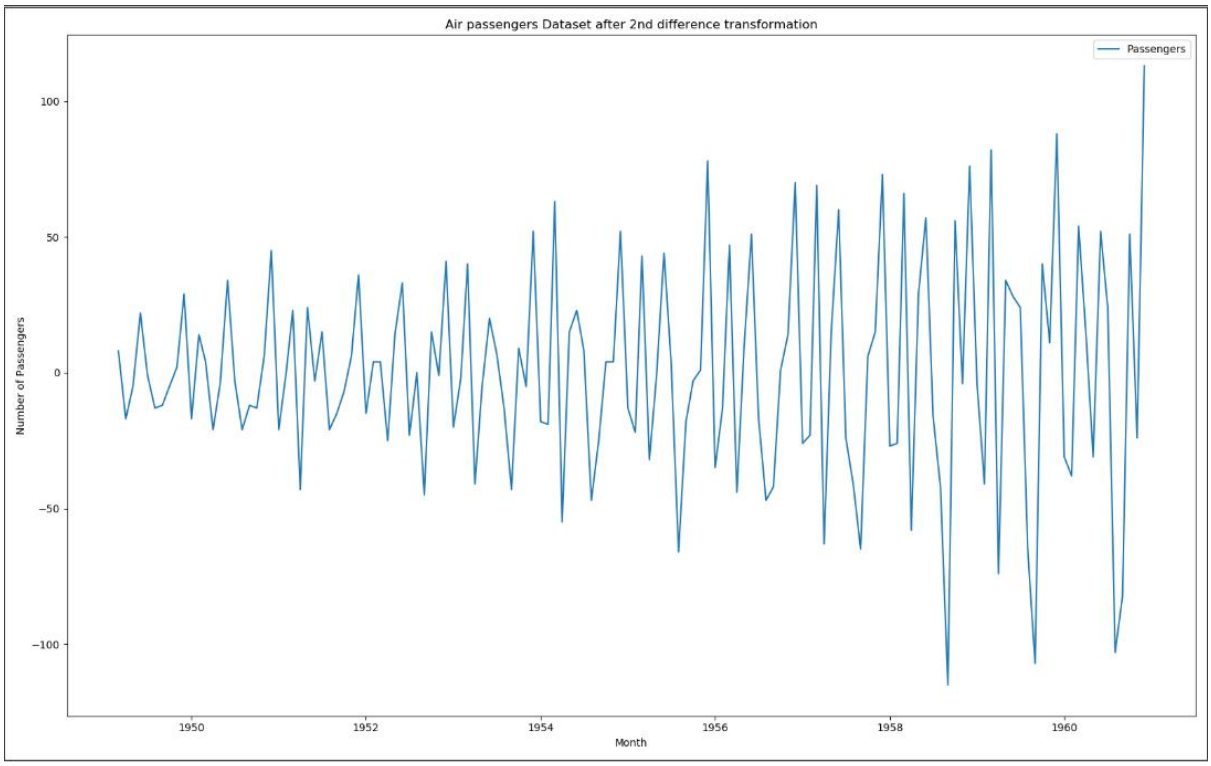
#Visualizing the Time Series plot for the number of Air Passengers



'Average Air passengers Dataset without differencing'



Air passengers Dataset after 1st difference transformation



Air passengers Dataset after 2nd difference transformation

PYTHON CODE

```
#Homework
# 1- Nonstationary process and removing trend from time series data (1st , 2nd
differencing and logarithmic transformation)
# Using the Python program and the required libraries perform the following tasks:

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 AirPassengers.csv.
# %%-----------------------------------------------------------------------------


data = pd.read_csv("AirPassengers.csv")
print("Shape of AirPassengers:",data.shape)
print('\n')

#%%=============================================================================
# 2: This date relates to the air passengers between 1949-1960.
# %%-----------------------------------------------------------------------------

print("DataType of AirPassengers:", data.dtypes)
print('\n')
for i in data.columns:
        print(i)
print('\n')


#%%=============================================================================
# 3: Write a Python code to display the first 5 rows of the data on the console.
# %%-----------------------------------------------------------------------------


#Creating the 'Date' as Index for data and viewing the dataset
data.Month = pd.to_datetime(data.Month)
print(data.head(5))
print('\n')
print(data.tail(5))
print('\n')


#%%=============================================================================
# 4: Explore the dataset by plotting the entire dataset, where you can learn more
about the data set pattern
# (trend, seasonality, cyclic, …). Add the label to the horizontal and vertical
axis as Month and Sales Number.
# Add the title as “Air passengers Dataset without differencing”.
# Add an appropriate legend to your plot.
# Do you see any trend, seasonality, or cyclical behavior in the plotted dataset?
# If yes, what is it?
# %%-----------------------------------------------------------------------------
```

```
#Visualizing the Time Series plot for the number of Air Passengers
plt.figure(figsize=(16,10))
plt.plot(data['Month'], data['#Passengers'], label = 'Passengers')
plt.xlabel("Month")
plt.ylabel("Number of Passengers")
plt.title('Air passengers Dataset without differencing')
plt.legend()
plt.show()
'''
It's clear from the plot that there is an overall increase in the trend,with some
seasonality in it.
'''
#%%=============================================================================
# 5: Run an ADF-test and check if the dataset is stationary or not. Is the dataset
non-stationary? Justify your answer.
# Calculate the average over the entire dataset and show the average plot.
# %%-----------------------------------------------------------------------------

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()

ADFcal(data['#Passengers'])
passavg = []
for i in range(1,145):
    k = pd.read_csv('AirPassengers.csv').head(i)
    passavg.append(np.mean(k['#Passengers']))

plt.figure(figsize=(16,10))
plt.plot(data['Month'], passavg, label = 'Average Passengers')
plt.xlabel("Month")
plt.ylabel("Number of Average Passengers")
plt.title('Average Air passengers Dataset without differencing')
plt.legend()
plt.show()
'''
From above ADF test, we fail to reject the null hypothesis, since p-value is
greater than 0.05
Below we took log transformation to make our Time series stationary and plotted
visual for it
We found graph upward trending over time with seasonality
'''
#%%=============================================================================
# 6: If the answer to the previous question is non-stationary,
# write a python code that detrend the dataset by 1st difference transformation.
Plot the detrended dataset.
# %%-----------------------------------------------------------------------------

# 1st difference transformation
diff1 = data['#Passengers'].diff()
plt.figure(figsize=(16,10))
plt.plot(data['Month'], diff1, label = 'Passengers')
plt.xlabel("Month")
plt.ylabel("Number of Passengers")
```

```
plt.title('Air passengers Dataset after 1st difference transformation')
plt.legend()
plt.show()

#%%=============================================================================
# 7: Is the detrended dataset stationary?
# Justify your answer by running an ADF-test. Plot the average and variance over
the entire dataset.
# %%-----------------------------------------------------------------------------

#Yes it is.
ADFcal(diff1[1:])


#%%=============================================================================
# 8:If the first differencing did not make the dataset to be stationary, then try
2nd differencing and repeat 7.
# %%-----------------------------------------------------------------------------

#Yes it did not make the dataset to stationary.
pass1avg, pass1var = [], []
for i in range(1,len(diff1)):
    pass1avg.append(np.mean(diff1[:i]))
    pass1var.append(np.var(diff1[:i]))


#%%=============================================================================
# 9: If the 2nd differencing did not make the dataset to be stationary,
# then perform the 1st differencing followed by logarithmic transformation. The
repeat step 7.
# %%-----------------------------------------------------------------------------

# 2nd difference transformation
diff2 = diff1.diff()
plt.figure(figsize=(16,10))
plt.plot(data['Month'], diff2, label = 'Passengers')
plt.xlabel("Month")
plt.ylabel("Number of Passengers")
plt.title('Air passengers Dataset after 2nd difference transformation')
plt.legend()
plt.show()

#%%=============================================================================
# 10: If the procedures in step 6, 8 & 9 did not make the dataset to be stationary
# (pass the ADF-test with 95% or more confidence interval) then stop.
# %%-----------------------------------------------------------------------------

ADFcal(diff2[2:])


#%%=============================================================================
==============================
# 11: Write a report and answer all the above questions. Include the required
graphs into your report.
# %%-----------------------------------------------------------------------------
------------------------------
#Created Solution Report with above code and added all relavant graphs
```