CSV Import and basic filtering using Python [0.5 point]:

In [587]:
```
#Step 1. Import the census.csv file into python using pandas.
```

In [158]:
```
import pandas as pd
```

In [159]:
```
df = pd.read_csv(r"C:\Users\16478\Desktop\census.csv",encoding='latin1')
```

In [160]:
```
print(df.STNAME[1832])
```

```
New Mexico
```

In [161]:
```
df
```

Out[161]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMAT
0	40	3	6	1	0	Alabama	Alabama	4779736	
1	50	3	6	1	1	Alabama	Autauga County	54571	
2	50	3	6	1	3	Alabama	Baldwin County	182265	
3	50	3	6	1	5	Alabama	Barbour County	27457	
4	50	3	6	1	7	Alabama	Bibb County	22915	
...	...	...	...	...	...	...	...	...	
3188	50	4	8	56	37	Wyoming	Sweetwater County	43806	
3189	50	4	8	56	39	Wyoming	Teton County	21294	

In [588]:
```
#Step 2. Import and Querying the census.csv file into python using pandas.
```

In [162]:
```
result=df[(df["STNAME"]=='New York') & (df["CENSUS2010POP"]>1000000)]
```

In [163]:
```
result
```

Out[163]:

	SUMLEV	REGION	DIVISION	STATE	COUNTY	STNAME	CTYNAME	CENSUS2010POP	ESTIMATESE
1860	40	1	2	36	0	New York	New York	19378102	
1863	50	1	2	36	5	New York	Bronx County	1385108	
1884	50	1	2	36	47	New York	Kings County	2504700	
1890	50	1	2	36	59	New York	Nassau County	1339532	
1891	50	1	2	36	61	New York	New York County	1585873	
1901	50	1	2	36	81	New York	Queens County	2230722	
1912	50	1	2	36	103	New York	Suffolk County	1493350	

7 rows × 100 columns

Step 3. Calculate sum and average population of counties in the state of New York containing more than one million people

In [ ]:
```
Sum_Value=result.CENSUS2010POP.sum()
```

In [165]:
```
Sum_Value
```

Out[165]: 29917387

In [166]:
```
Avg_Value=result.CENSUS2010POP.mean()
```

In [167]:
```
Avg_Value
```

Out[167]: 4273912.428571428

In [ ]:
```
#Step 4. Print all the datatypes with column names.
```

In [168]:
```
df.dtypes
```

Out[168]:
```
SUMLEV           int64
REGION           int64
DIVISION         int64
STATE            int64
COUNTY           int64
                 ...
RNETMIG2011    float64
RNETMIG2012    float64
RNETMIG2013    float64
RNETMIG2014    float64
RNETMIG2015    float64
Length: 100, dtype: object
```

In [169]:
```
df.dtypes.value_counts()
```

Out[169]:
```
int64      68
float64    30
object      2
dtype: int64
```

In [ ]:
```
#Step 5. Print the number of empty values in each column
```

In [170]:
```
df.isnull().sum().sort_values(ascending=False)
```

Out[170]:
```
SUMLEV                0
GQESTIMATESBASE2010    0
RBIRTH2014             0
RBIRTH2013             0
RBIRTH2012             0
                      ..
DEATHS2013             0
DEATHS2012             0
DEATHS2011             0
DEATHS2010             0
RNETMIG2015            0
Length: 100, dtype: int64
```

Exploring the relational model [1.0 point]:

```
Q1: Which state has Box Butte County
Q2: What is the population of Box Butte County the CENSUS2010POP
Q3: What County in the state of Montana has the smallest estimated population?
```

In [171]:
```
state=df[(df["CTYNAME"]=='Box Butte County')].STNAME
population=df[(df["CTYNAME"]=='Box Butte County')].CENSUS2010POP
z=df[(df["STNAME"]=='Montana')]
output= z['CENSUS2010POP'].min()
smallest_pop=z[z['CENSUS2010POP']==output].CTYNAME
```

In [172]:
```
print(f'\n State has Box Butte County :',state.to_string(index=False))
print(f'\n Population of Box Butte County :',population.to_string(index=False))
print(f'\n Smallest Estimated Population :',smallest_pop.to_string(index=False))
```

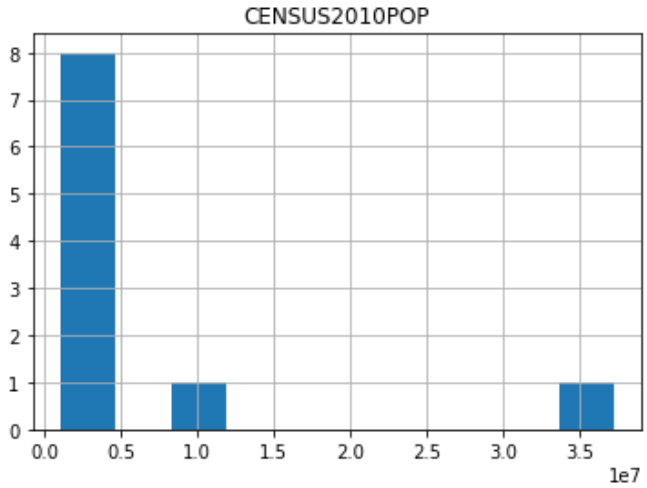
```
 State has Box Butte County : Nebraska

 Population of Box Butte County : 11308

 Smallest Estimated Population : Petroleum County
```

Q4: Plot a histogram of CENSUS2010POP for counties with population grater than one million in state of California.

In [173]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [174]:
```
df_graph = df[(df["CENSUS2010POP"]>1000000)&(df["STNAME"]=="California")]
#df_graph.plot.hist()

df_graph.hist(column = 'CENSUS2010POP')
```

Out[174]:
```
array([[<AxesSubplot:title={'center':'CENSUS2010POP'}>]], dtype=object)
```



Exploring the Semi-structured Data Model of JSON data

JSON Import and exploration printing [1 point]

In [591]:
```
#Step 1. Import twitter.json into Python
```

In [175]:
```
import json
import jsonschema
```

In [176]:
```
data=pd.read_json(r'C:\Users\16478\Desktop\BigData\Assignment2\data\twitter.json', lin
```

In [177]:
```
data
```

Out[177]:

	created_at	id	id_str	text	s
0	2015-08-13 21:07:54+00:00	631935230014681089	631935230014681088	RT @Warcraft: We've just posted a sneak previe...	<a href="http://twitte rel="nofollow
1	2015-08-13 21:07:56+00:00	631935238415978496	631935238415978496	RT @gamespot: Fallout Shelter available now on...	href="http://twitter.com/download/ar
2	2015-08-13 21:07:57+00:00	631935243373690882	631935243373690880	RT @NBA2K: Watch today's #NBA2K16 Presents Jam...	href="http://twitter.com/download/ip
3	2015-08-13 21:07:58+00:00	631935246624256000	631935246624256000	RT @gamespot: LIVE: The GameSpot crew tries ou...	href="http://twitter.com/download/ip
4	2015-08-13 21:08:15+00:00	631935320142032896	631935320142032896	RT @gamespot: You can now buy Prototype 1 and ...	href="http://twitter.com/download/ip
...	...	...	...	...	
495	2015-08-13 21:47:16+00:00	631945138563096576	631945138563096576	RT @gamespot: #GameSpot Poll: Which Blizzard g...	href="https://twitter.com/NextGenP
496	2015-08-13 21:47:19+00:00	631945149187260416	631945149187260416	@gamespot no, I don't care	href="http://twitter.com/download/ar
497	2015-08-13 21:47:22+00:00	631945162646687744	631945162646687744	@IGN This is actually REALLY sick!	href="http://twitter.com/download/ip
498	2015-08-13 21:47:24+00:00	631945170246893568	631945170246893568	RT @talign: Spartan party! aka Sparty! http://...	href="http://twitter.com/download/ar
499	2015-08-13 21:47:25+00:00	631945175984541696	631945175984541696	RT @IGN: #StarWars, @Marvel, Kingdom Hearts 3 ...	href="http://twitter.com/download/ar

500 rows × 27 columns

In [178]:
```
data.user[0]['friends_count']
```

Out[178]: 628

In [179]:
```
file=r'C:\Users\16478\Desktop\BigData\Assignment2\data\twitter.json'
twitter_data=[]
i=0
with open(file) as f:
    for line in f:
        i=i+1
        if i%2==1:
            twitter_data.append(json.loads(line))
for line in twitter_data:
    print(line['created_at'])
```

```
Thu Aug 13 21:21:35 +0000 2015
Thu Aug 13 21:21:40 +0000 2015
Thu Aug 13 21:21:47 +0000 2015
Thu Aug 13 21:21:51 +0000 2015
Thu Aug 13 21:21:58 +0000 2015
Thu Aug 13 21:22:01 +0000 2015
Thu Aug 13 21:22:11 +0000 2015
Thu Aug 13 21:22:34 +0000 2015
Thu Aug 13 21:22:36 +0000 2015
Thu Aug 13 21:22:40 +0000 2015
Thu Aug 13 21:22:42 +0000 2015
Thu Aug 13 21:22:47 +0000 2015
Thu Aug 13 21:22:53 +0000 2015
Thu Aug 13 21:23:03 +0000 2015
Thu Aug 13 21:23:09 +0000 2015
Thu Aug 13 21:23:13 +0000 2015
Thu Aug 13 21:23:26 +0000 2015
Thu Aug 13 21:23:45 +0000 2015
Thu Aug 13 21:23:52 +0000 2015
Thu Aug 13 21:23:56 +0000 2015
```

In [592]:
```
#Step 2: Print the schema of the twitter.json
```

In [180]:
```
with open(file) as f:
    for line in f:
        i=i+1
        if i%2==1:
            schema=json.loads(line)
            json_structure = json.dumps(schema, indent=4, sort_keys=True, separators=(
print(json_structure)
```


```
{
    "contributors": null,
    "coordinates": null,
    "created_at": "Thu Aug 13 21:47:25 +0000 2015",
    "entities": {
        "hashtags": [
            {
                "indices": [
                    9,
                    18
                ],
                "text": "StarWars"
            }
        ],
        "media": [
            {
                "display_url": "pic.twitter.com/bj6HoGcXyf",
                "expanded_url": "http://twitter.com/IGN/status/631917845501345792/
photo/1",
```


In [586]:
```
result_schema = data.to_json(orient="table")
parsed_schema = json.loads(result_schema)
parsed_schema['schema']
#json.dumps(parsed_schema , indent='')
```


```
{'name': 'text', 'type': 'string'},
{'name': 'source', 'type': 'string'},
{'name': 'truncated', 'type': 'boolean'},
{'name': 'in_reply_to_status_id', 'type': 'number'},
{'name': 'in_reply_to_status_id_str', 'type': 'number'},
{'name': 'in_reply_to_user_id', 'type': 'number'},
{'name': 'in_reply_to_user_id_str', 'type': 'number'},
{'name': 'in_reply_to_screen_name', 'type': 'string'},
{'name': 'user', 'type': 'string'},
{'name': 'geo', 'type': 'number'},
{'name': 'coordinates', 'type': 'number'},
{'name': 'place', 'type': 'string'},
{'name': 'contributors', 'type': 'number'},
{'name': 'retweeted_status', 'type': 'string'},
{'name': 'retweet_count', 'type': 'integer'},
{'name': 'favorite_count', 'type': 'integer'},
{'name': 'entities', 'type': 'string'},
{'name': 'extended_entities', 'type': 'string'},
{'name': 'favorited', 'type': 'boolean'},
{'name': 'retweeted', 'type': 'boolean'},
```


#Step 3: Given a tweet, what path would you enter to obtain a count of the number of friends for a user?

In [133]:
```
for line in twitter_data:
    if(line['user']['id']==397495839):
        print(line['user']['friends_count'])
```

```
628
```

In [680]:
```
#Step 4: Which of the following fields are nested within the 'extended_entities' field
```

In [679]:
```
for key in twitter_data:
    if 'extended_entities' in key:
        nested = key['extended_entities'].keys()

print('The nested value present within Extended Entity is - ',list(nested) )
print('\n')


for line in twitter_data:
    if 'extended_entities' in line:
        extended_entities_keys = line["extended_entities"]
print('Fields present within extended entities are - \n', extended_entities_keys)
```

```
The nested value present within Extended Entity is -  ['media']


Fields present within extended entities are -
 {'media': [{'id': 631917844788150272, 'id_str': '631917844788150272', 'indices': [12
4, 140], 'media_url': 'http://pbs.twimg.com/media/CMUF7onUAAA7tbI.jpg', 'media_url_ht
tps': 'https://pbs.twimg.com/media/CMUF7onUAAA7tbI.jpg', 'url': 'http://t.co/bj6HoGcX
yf', 'display_url': 'pic.twitter.com/bj6HoGcXyf', 'expanded_url': 'http://twitter.co
m/IGN/status/631917845501345792/photo/1', 'type': 'photo', 'sizes': {'large': {'w': 9
85, 'h': 554, 'resize': 'fit'}, 'thumb': {'w': 150, 'h': 150, 'resize': 'crop'}, 'sma
ll': {'w': 340, 'h': 191, 'resize': 'fit'}, 'medium': {'w': 600, 'h': 337, 'resize':
'fit'}}, 'source_status_id': 631917845501345792, 'source_status_id_str': '63191784550
1345792'}]}
```

List and count the distinct locations of the users in the provided JSON file? [1 point]

In [681]:
```
unique_list=[]
for line in twitter_data:
    unique=line['user']['location']
    unique_list.append(unique)
```

In [682]:
```
from collections import Counter
k=Counter(unique_list).keys()
v=Counter(unique_list).values()
dict(zip(k,v))
```

Out[682]:
```
{'Orange County, CA': 1,
 'Franca com c, Kattegat': 2,
 '': 192,
 'Irvine, CA': 1,
 'Krefeld.Germany': 3,
 'Queens, NY': 1,
 'UK': 4,
 'denmark': 1,
 'Silvermoon City': 2,
 'New Jersey': 4,
 'San Francisco, CA': 4,
 'XBOX GT: BeyondLimits01': 1,
 'In-game lobby': 1,
 'Boston QZ': 1,
 'United Kingdom': 6,
 'US & UK': 3,
 'New Jersey, USA': 1,
 'Lone star state ': 1,
 'California ': 3,
```

Exploring Sensory Data

In [406]:
```
import socket
from datetime import datetime
import re
```

In [683]:
```
#Connect to web socket and stream at least fifty samples. [0.5 points]
```

In [397]:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('rtd.hpwren.ucsd.edu', 12020))
for i in range (0,51):
    d=s.recv(1024)
    p=d.decode('utf-8').split('\t',2)
    print(p)
    print(('--' * 50))
    print(i, p[2])

s.close()
```

```
['198.202.124.3', 'HPWREN:LP-WXT536:0R1:4:0', '1676670848\t0R1,Dn=000#,Dm=000#,Dx=
000#,Sn=0.0#,Sm=99.9#,Sx=0.0#\n\r']
----------------------------------------------------------------------------------
------------------
0 1676670848    0R1,Dn=000#,Dm=000#,Dx=000#,Sn=0.0#,Sm=99.9#,Sx=0.0#

['198.202.124.3', 'HPWREN:LP-WXT536:0R1:4:0', '1676670849\t0R1,Dn=000#,Dm=000#,Dx=
000#,Sn=0.0#,Sm=99.9#,Sx=0.0#\n\r']
----------------------------------------------------------------------------------
------------------
1 1676670849    0R1,Dn=000#,Dm=000#,Dx=000#,Sn=0.0#,Sm=99.9#,Sx=0.0#

['198.202.124.3', 'HPWREN:LP-WXT536:0R2:4:0', '1676670849\t0R2,Ta=9.4C,Ua=20.4P,Pa
=882.5H\n\r']
----------------------------------------------------------------------------------
------------------
2 1676670849    0R2,Ta=9.4C,Ua=20.4P,Pa=882.5H

['198.202.124.3', 'HPWREN:LP-WXT536:0R1:4:0', '1676670850\t0R1,Dn=000#,Dm=000#,Dx=
```

Gather at least five hundred sensor readings from the stream for air pressure and air temperature [1 point]

In [562]:

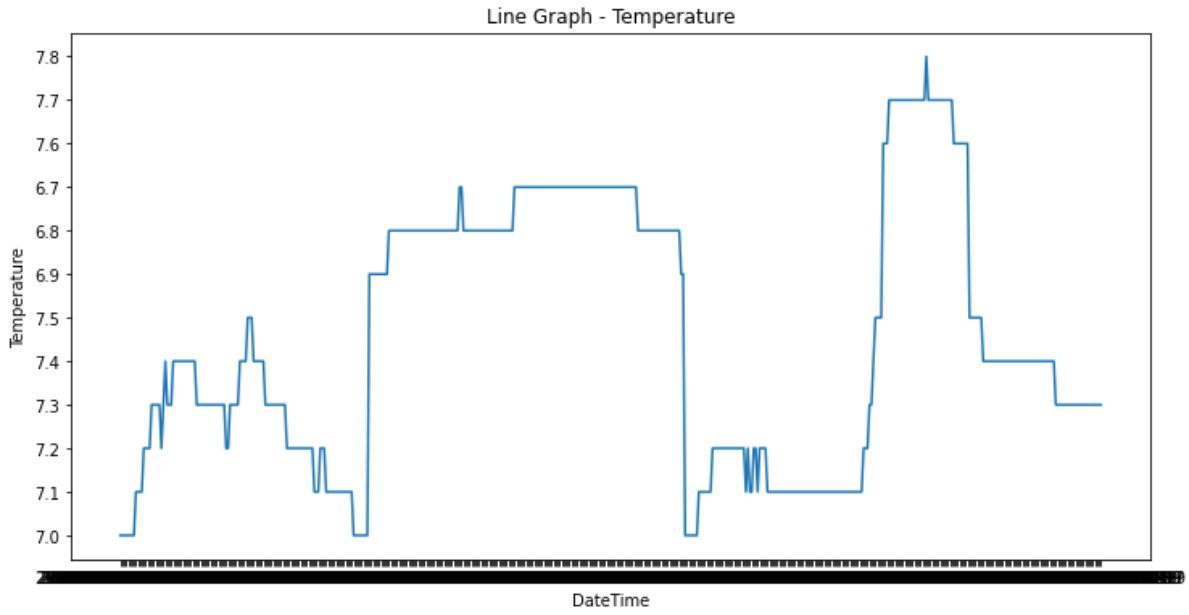
```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('rtd.hpwren.ucsd.edu', 12020))
temperature=[]
pressure=[]
date=[]
count=0

df_socket = pd.DataFrame(columns=['DateTime', 'Temperature' , 'Pressure'])
while(count<=500) :
    d=s.recv(1024)
    p=d.decode('utf-8').split('\t',2)
    #print(p)

    #parse variables

    temp=p[2]
    split=temp.split('\t')
    time=split[0]
    attr=split[1]


    if re.search('Ta', attr):
        count=count+1
        #print(count)
        patt1=r"Ta=(.*?)C"
        par1=re.search(patt1,attr)
        temperature.append(par1.group(1))
        patt2=r"Pa=(.*?)H"
        par2=re.search(patt2,attr)
        pressure.append(par2.group(1))

        #get date time
        date_time = str(datetime.fromtimestamp(int(time),tz=None))
        date.append(date_time)



s.close()
df_socket['Temperature']= temperature
df_socket['Pressure']= pressure
df_socket['DateTime']= date
df_socket
```

	DateTime	Temperature	Pressure
0	2023-02-17 21:38:59	7.0	882.4
1	2023-02-17 21:39:09	7.0	882.4
2	2023-02-17 21:39:17	7.0	882.4
3	2023-02-17 21:39:19	7.0	882.4
4	2023-02-17 21:39:29	7.0	882.4
...	...	...	...
496	2023-02-17 22:49:49	7.3	882.7
497	2023-02-17 22:49:59	7.3	882.7
498	2023-02-17 22:50:09	7.3	882.7
499	2023-02-17 22:50:17	7.3	882.8
500	2023-02-17 22:50:19	7.3	882.7

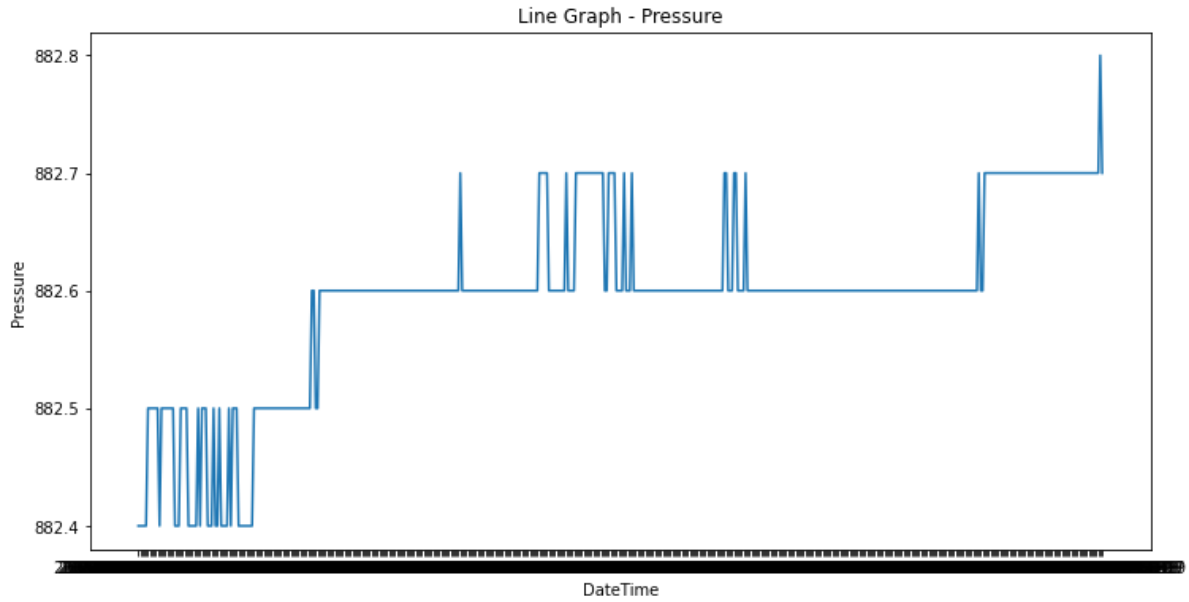
501 rows × 3 columns

In [684]:
```
#Plot the sensor readings as a line plot. [0.25 points]
```

In [568]:
```
import matplotlib.pyplot as plt


# Plot the data
plt.figure(figsize=(12, 6))
plt.plot(df_socket['DateTime'], df_socket['Temperature'])
plt.xlabel('DateTime')
plt.ylabel('Temperature')
plt.title('Line Graph - Temperature')
plt.show()
```



In [567]:
```
import matplotlib.pyplot as plt


# Plot the data
plt.figure(figsize=(12, 6))
plt.plot(df_socket['DateTime'], df_socket['Pressure'])
plt.xlabel('DateTime')
plt.ylabel('Pressure')
plt.title('Line Graph - Pressure')
plt.show()
```



In [687]:
```
#Print summary statistics for the sensor values [0.25 points]
```

In [686]:
```
df_socket['Pressure'] = pd.to_numeric(df_socket['Pressure'], errors='coerce')
df_socket['Temperature']=pd.to_numeric(df_socket['Temperature'], errors='coerce')
df_socket.describe()
```

Out[686]:

	Temperature	Pressure
count	501.000000	501.000000
mean	7.123353	882.594411
std	0.296772	0.075952
min	6.700000	882.400000
25%	6.800000	882.600000
50%	7.100000	882.600000
75%	7.300000	882.600000
max	7.800000	882.800000

In [476]:
```
print('Median -\n ',df_socket.median(numeric_only=True))
print('Mode - \n',df_socket.mode(numeric_only=True,dropna=True))
```

```
Median -
  Temperature      9.3
Pressure       882.5
dtype: float64
Mode -
    Temperature  Pressure
0          9.3     882.5
```

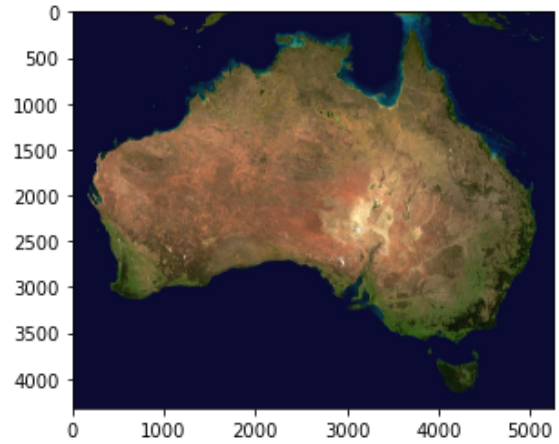
Exploring the Vector Model of an image

In [478]:
```
! pip install opencv-python
import cv2
import numpy as np
```

```
Collecting opencv-python
  Downloading opencv_python-4.7.0.68-cp37-abi3-win_amd64.whl (38.2 MB)
Requirement already satisfied: numpy>=1.17.3 in c:\users\16478\anaconda3\lib\site-pac
kages (from opencv-python) (1.21.5)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.7.0.68
```

In [688]:
```
#Import the image as 3-dimensinal array. [0.25 points]
```

In [490]:
```
array = cv2.imread(r"C:\Users\16478\Desktop\BigData\Assignment2\data\Australia.jpg")
# Converting BGR to RGB
array = cv2.cvtColor(array, cv2.COLOR_BGR2RGB)
plt.imshow(array)
plt.show()
```



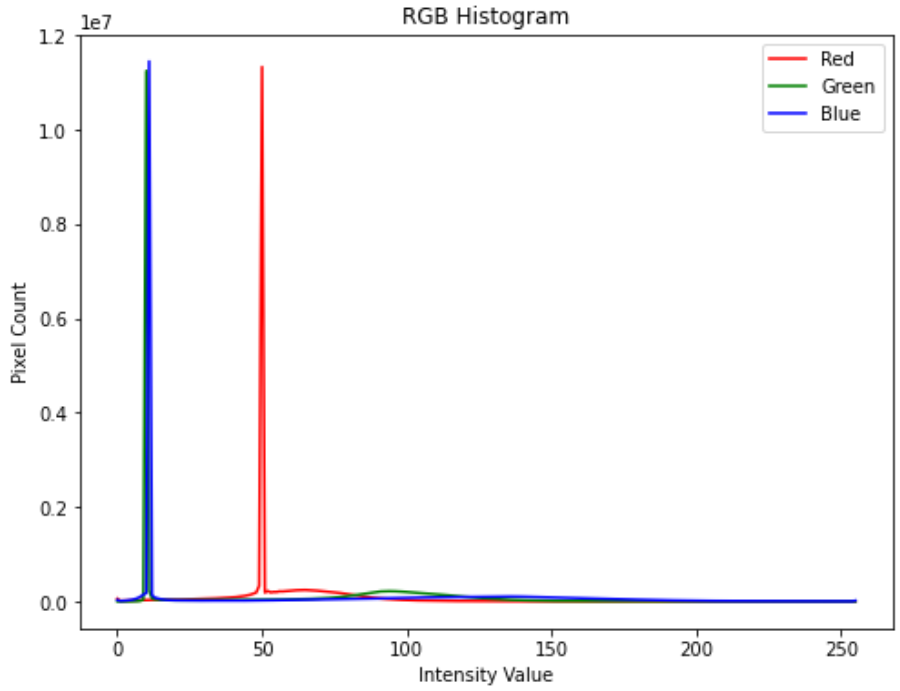
In [689]:
```
#What is the (Red, Green, Blue) pixel value for location 500, 2000? [0.25 points]
```

In [488]:
```
#What is the (Red, Green, Blue) pixel value for location 500, 2000?
pixel=array[500,2000]
print('RGB Vqalue of Location is ', pixel)
```

```
RGB Vqalue of Location is  [11 10 50]
```

In [690]:
```
#Plot the histogram of RGB channels? [0.5 point]
```

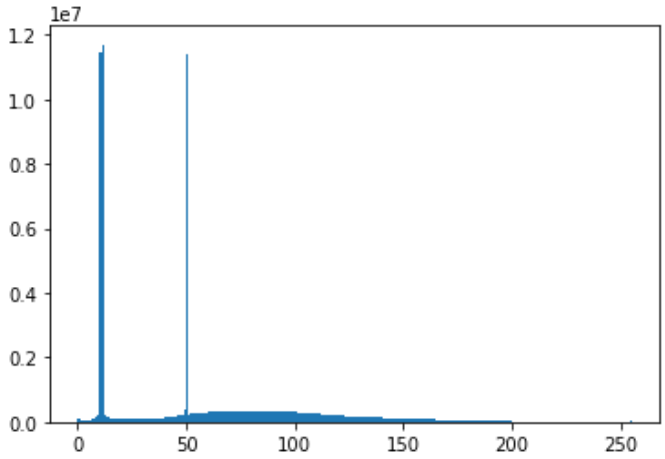
In [492]:
```
b,g,r=cv2.split(array)
hist_r = cv2.calcHist([r], [0], None, [256], [0, 256])
hist_g = cv2.calcHist([g], [0], None, [256], [0, 256])
hist_b = cv2.calcHist([b], [0], None, [256], [0, 256])

plt.figure(figsize=(8, 6))
plt.plot(hist_r, color='r', label='Red')
plt.plot(hist_g, color='g', label='Green')
plt.plot(hist_b, color='b', label='Blue')
plt.xlabel('Intensity Value')
plt.ylabel('Pixel Count')
plt.title('RGB Histogram')
plt.legend()
plt.show()
```



In [495]:
```
plt.hist(array.ravel(), bins=range(256))
```

Out[495]:
```
(array([8.7610000e+04, 2.2392000e+04, 2.8130000e+04, 3.5187000e+04,
        4.5229000e+04, 5.8330000e+04, 7.5510000e+04, 1.0224500e+05,
        1.4299200e+05, 2.1638800e+05, 1.1453201e+07, 1.1704168e+07,
        2.2701900e+05, 1.5653300e+05, 1.3274300e+05, 1.1236600e+05,
        1.0035200e+05, 9.3102000e+04, 8.8164000e+04, 8.5076000e+04,
        8.3656000e+04, 8.3158000e+04, 8.3816000e+04, 8.5008000e+04,
        8.5617000e+04, 8.7786000e+04, 8.9806000e+04, 9.2016000e+04,
        9.4649000e+04, 9.7039000e+04, 9.9131000e+04, 1.0176300e+05,
        1.0393300e+05, 1.0623000e+05, 1.0823200e+05, 1.1146200e+05,
        1.1393100e+05, 1.1822500e+05, 1.2167700e+05, 1.2668200e+05,
        1.3135000e+05, 1.3660300e+05, 1.4540300e+05, 1.5421200e+05,
        1.6240600e+05, 1.7668400e+05, 1.8944700e+05, 2.1462300e+05,
        2.3726800e+05, 3.8060600e+05, 1.1380412e+07, 2.4005800e+05,
        2.7665100e+05, 2.4722200e+05, 2.5968300e+05, 2.6032800e+05,
        2.7566600e+05, 2.7663400e+05, 2.8444300e+05, 2.9087300e+05,
        2.9961800e+05, 3.0670700e+05, 3.1355500e+05, 3.1862400e+05,
        3.2346400e+05, 3.2590200e+05, 3.2622300e+05, 3.2661100e+05,
        3.2450000e+05, 3.2362000e+05, 3.2037000e+05, 3.2028400e+05,
        3.1902200e+05, 3.1709700e+05, 3.1577800e+05, 3.1438100e+05,
        3.1425900e+05, 3.1245800e+05, 3.1159600e+05, 3.1032200e+05,
        3.1007000e+05, 3.0996300e+05, 3.1083400e+05, 3.1259900e+05,
        3.1423100e+05, 3.1729000e+05, 3.2041600e+05, 3.2442500e+05,
        3.2797700e+05, 3.3109300e+05, 3.3399800e+05, 3.3663000e+05,
        3.3492600e+05, 3.3590900e+05, 3.3320800e+05, 3.3153800e+05,
        3.2659000e+05, 3.2296500e+05, 3.1693600e+05, 3.1148200e+05,
        3.0571700e+05, 2.9812300e+05, 2.9368700e+05, 2.8699000e+05,
        2.8230700e+05, 2.7693100e+05, 2.7166800e+05, 2.6594900e+05,
        2.6182600e+05, 2.5717500e+05, 2.5044300e+05, 2.4710800e+05,
        2.4131500e+05, 2.3620800e+05, 2.3012200e+05, 2.2345900e+05,
        2.1872500e+05, 2.1184700e+05, 2.0668900e+05, 2.0175800e+05,
        1.9825500e+05, 1.9289600e+05, 1.8854600e+05, 1.8404900e+05,
        1.8067900e+05, 1.7641200e+05, 1.7399100e+05, 1.6902600e+05,
        1.6698400e+05, 1.6280400e+05, 1.5923800e+05, 1.5674300e+05,
        1.5167500e+05, 1.4866500e+05, 1.4536000e+05, 1.4276100e+05,
        1.3963600e+05, 1.3710300e+05, 1.3402900e+05, 1.3007200e+05,
        1.2800100e+05, 1.2478500e+05, 1.2179500e+05, 1.1876000e+05,
        1.1633900e+05, 1.1396000e+05, 1.1069600e+05, 1.0788900e+05,
        1.0491500e+05, 1.0254800e+05, 9.9373000e+04, 9.7704000e+04,
        9.6014000e+04, 9.4520000e+04, 9.1848000e+04, 8.9891000e+04,
        8.8863000e+04, 8.7057000e+04, 8.5813000e+04, 8.3818000e+04,
        8.2893000e+04, 8.1151000e+04, 7.9142000e+04, 7.7144000e+04,
        7.4718000e+04, 7.1679000e+04, 7.0019000e+04, 6.6898000e+04,
        6.3830000e+04, 6.0678000e+04, 5.8938000e+04, 5.5646000e+04,
        5.3459000e+04, 5.2187000e+04, 5.0332000e+04, 4.8776000e+04,
        4.7184000e+04, 4.5640000e+04, 4.3946000e+04, 4.2095000e+04,
        4.0776000e+04, 3.8925000e+04, 3.7757000e+04, 3.6080000e+04,
        3.4538000e+04, 3.2960000e+04, 3.1907000e+04, 3.0629000e+04,
        2.9372000e+04, 2.7886000e+04, 2.6799000e+04, 2.5802000e+04,
        2.4629000e+04, 2.3443000e+04, 2.2569000e+04, 2.1201000e+04,
        2.0073000e+04, 1.9029000e+04, 1.8383000e+04, 1.7142000e+04,
        1.6308000e+04, 1.5540000e+04, 1.4865000e+04, 1.4013000e+04,
        1.3423000e+04, 1.2919000e+04, 1.2464000e+04, 1.1875000e+04,
        1.1505000e+04, 1.0977000e+04, 1.0519000e+04, 1.0117000e+04,
        9.8220000e+03, 9.3280000e+03, 8.8970000e+03, 8.5020000e+03,
        8.2810000e+03, 7.9560000e+03, 7.7040000e+03, 7.4060000e+03,
        7.3660000e+03, 6.8700000e+03, 6.7490000e+03, 6.6410000e+03,
        6.2560000e+03, 6.1990000e+03, 5.8240000e+03, 5.7040000e+03,
        5.5030000e+03, 5.2870000e+03, 5.1600000e+03, 4.8320000e+03,
        4.8540000e+03, 4.5910000e+03, 4.4570000e+03, 4.2650000e+03,
        4.2530000e+03, 4.0500000e+03, 3.8520000e+03, 3.8130000e+03,
        3.5910000e+03, 3.6250000e+03, 3.4430000e+03, 3.3570000e+03,
        3.2860000e+03, 3.1310000e+03, 3.1900000e+03, 2.9820000e+03,
        3.0610000e+03, 2.9670000e+03, 3.0430000e+03, 2.9790000e+03,
```

```
       2.8760000e+03, 2.8640000e+03, 2.0246000e+04]),
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103,
       104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
       117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142,
       143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
       156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168,
       169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181,
       182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194,
       195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207,
       208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220,
       221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233,
       234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246,
       247, 248, 249, 250, 251, 252, 253, 254, 255]),
<BarContainer object of 255 artists>)
```



Exploring Vector Model of Text

Import texts and TFIDF vectorize [2 points]

What news file talks about voters more. [0.5 points]

What news file talks about delegates more. [0.5 points]

In [496]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

In [498]:
```
n1 = pd.read_csv(r"C:\Users\16478\Desktop\BigData\Assignment2\data\news1.csv",encoding=
n2 = pd.read_csv(r"C:\Users\16478\Desktop\BigData\Assignment2\data\news2.csv",encoding=
n3 = pd.read_csv(r"C:\Users\16478\Desktop\BigData\Assignment2\data\news3.csv",encoding=
```

In [500]:
```
n1
```

Out[500]:

	ID	NewsSource	Date	Title	NewsBody
0	1	ABC News	4/8/2016	Bernie Sanders Explains Why Hillary Clinton Is...	Democratic presidential candidate Bernie Sande...
1	2	Los Angeles Times	4/8/2016	Trump, Clinton lead the pack in California, ne...	With just two months to go before the Californ...
2	3	New York Times	4/8/2016	Bill Clinton Says He Regrets Showdown With Bla...	Former President Bill Clinton said Friday that...
3	4	CNN	4/8/2016	TSA chief: More visible security, random check...	Transportation Security Administration Adminis...
4	5	The Wall Street Journal	4/8/2016	Bernie Sanders Will Visit the Vatican	Democratic presidential candidate Bernie Sande...

In [501]:
```
n2
```

Out[501]:

	ID	NewsSource	Date	Title	NewsBody
0	6	The New York Times	4/8/2016	Hillary Clinton and Bernie Sanders Bring Their...	In Brooklyn Heights, on the 11th floor of an o...
1	7	The Wall Street Journal	4/8/2016	Bernie Sanders s Supporters Press Hillary Clin...	Below the surface of the Democratic campaign, ...
2	8	CNN	4/8/2016	Bernie Sanders' campaign manager: Hillary Clin...	Bernie Sanders' campaign manager Jeff Weaver s...
3	9	The New York Times	4/8/2016	Catholics Express Hope and Disappointment Over...	Wedding invitations. Empty nesters. In vitro f...
4	10	CNN	4/8/2016	Paris terror suspect Mohamed Abrini arrested i...	Is a terror suspect arrested Friday in Belgium...

In [502]:
```
n3
```

Out[502]:

	ID	NewsSource	Date	Title	NewsBody
0	12	Boston Globe	3/22/2016	Clinton, Trump offer contrasting responses to ...	Donald Trump called Tuesday for greater relian...
1	13	The Washington Post	4/6/2016	Donald Trump thinks more countries should have...	According to Donald Trump, the United States s...
2	14	CNN	3/31/2016	Japan and South Korea hit back at Trump's nucl...	Confused, shocked, bewildered. Just a few of t...
3	15	BBC	4/7/2016	US election: Could Trump really cut the US $19...	Donald Trump's policy proposals have generated...

In [557]:
```
q="voters"
w="delegates"

v1=TfidfVectorizer()
v2=TfidfVectorizer()
v3=TfidfVectorizer()

d1=v1.fit_transform(n1['NewsBody'])
d2=v2.fit_transform(n2['NewsBody'])
d3=v3.fit_transform(n3['NewsBody'])

q1=v1.transform([q])
q2=v2.transform([q])
q3=v3.transform([q])

w1=v1.transform([w])
w2=v2.transform([w])
w3=v3.transform([w])

cs1=cosine_similarity(d1,q1)
cs2=cosine_similarity(d2,q2)
cs3=cosine_similarity(d3,q3)

cs4=cosine_similarity(d1,w1)
cs5=cosine_similarity(d2,w2)
cs6=cosine_similarity(d3,w3)


a={'News1':sum(cs1),'News2':sum(cs2),'News3':sum(cs3)}
print('News file talks about Voters more is',max(a, key=a.get))

b={'News1':sum(cs3),'News2':sum(cs4),'News3':sum(cs4)}
print('News file talks about Delegates more is ',max(b, key=b.get))
```

```
News file talks about Voters more is News1
News file talks about Delegates more is  News2
```

In [505]:

In [519]:

In [521]:

In [ ]:

In [ ]: