Python for Data Science (LAB Session-8)

Working with Dataset using pandas

Q-1)Load the data from ufo_sighting_data_complete_NUFORC.csv into pandas dataframe and print the shape and columns of the data.

In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
df = pd.read_csv('ufo_sighting_data_complete_NUFORC.csv')
df
```

```
c:\users\arjun vankani\appdata\local\programs\python\python37\lib\site-packag
es\IPython\core\interactiveshell.py:3146: DtypeWarning: Columns (4) have mixe
d types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[2]:

	datetime	state	country	shape	duration (seconds)	latitude	longitude	Unnamed: 7
0	10-10-1949 20:30	tx	us	cylinder	2700	29.8830556	-97.941111	NaN
1	10-10-1949 21:00	tx	NaN	light	7200	29.38421	-98.581082	NaN
2	10-10-1955 17:00	NaN	gb	circle	20	53.2	-2.916667	NaN
3	10-10-1956 21:00	tx	us	circle	20	28.9783333	-96.645833	NaN
4	10-10-1960 20:00	hi	us	light	900	21.4180556	-157.803611	NaN
...	...	...	...	...	...	...	...	...
88870	09-09-2013 22:00	ca	us	other	1200	38.2972222	-122.284444	NaN
88871	09-09-2013 22:20	va	us	circle	5	38.9011111	-77.265556	NaN
88872	09-09-2013 23:00	ok	us	cigar	1020	35.6527778	-97.477778	NaN
88873	09-09-2013 23:00	sc	us	diamond	0	34.3769444	-82.695833	NaN
88874	09-09-2013 23:30	fl	us	oval	0	26.1219444	-80.143611	NaN

88875 rows × 8 columns

In [3]:

```
print(df)  # print data frame
```

```
                datetime state country     shape duration (seconds)  \
0       10-10-1949 20:30    tx      us  cylinder               2700   
1       10-10-1949 21:00    tx     NaN     light               7200   
2       10-10-1955 17:00   NaN      gb    circle                 20   
3       10-10-1956 21:00    tx      us    circle                 20   
4       10-10-1960 20:00    hi      us     light                900   
...                  ...   ...     ...       ...                ...   
88870   09-09-2013 22:00    ca      us     other               1200   
88871   09-09-2013 22:20    va      us    circle                  5   
88872   09-09-2013 23:00    ok      us     cigar               1020   
88873   09-09-2013 23:00    sc      us   diamond                  0   
88874   09-09-2013 23:30    fl      us      oval                  0   

          latitude   longitude  Unnamed: 7  
0       29.8830556  -97.941111         NaN  
1         29.38421  -98.581082         NaN  
2             53.2   -2.916667         NaN  
3       28.9783333  -96.645833         NaN  
4       21.4180556 -157.803611         NaN  
...            ...         ...         ...  
88870   38.2972222 -122.284444         NaN  
88871   38.9011111  -77.265556         NaN  
88872   35.6527778  -97.477778         NaN  
88873   34.3769444  -82.695833         NaN  
88874   26.1219444  -80.143611         NaN  

[88875 rows x 8 columns]
```

In [4]:

```
df.shape  #shape of data frame  (row,col)
```

Out[4]:

```
(88875, 8)
```

In [ ]:

```

```

Q-2)Calculate the percentage of null value entries present into each column and display the data by sorting the data.

In [5]:

```
df.describe()  # describe all the value
```

Out[5]:

	longitude	Unnamed: 7
count	88875.000000	196.0
mean	-84.834334	0.0
std	41.567822	0.0
min	-176.658056	0.0
25%	-112.046944	0.0
50%	-87.650000	0.0
75%	-77.615833	0.0
max	178.441900	0.0

In [23]:

```
per = df.isnull().sum() *100 / len(df)     # find percant
dataframe = pd.DataFrame({'Percentage' : per})
dataframe
```

Out[23]:

	Percentage
datetime	0.000000
state	8.460197
country	14.133333
shape	3.508298
duration (seconds)	0.002250
latitude	0.000000
longitude	0.000000
Unnamed: 7	99.779466

In [24]:

```
dataframe.sort_values(by=['Percentage'], inplace=True, ascending=False)
dataframe  # arrange in ascending
```

Out[24]:

	Percentage
Unnamed: 7	99.779466
country	14.133333
state	8.460197
shape	3.508298
duration (seconds)	0.002250
datetime	0.000000
latitude	0.000000
longitude	0.000000

Q-3) Remove the null values from the dataset. Our intention here is to keep the row if at least 6 values in the row and not-null , else we should remove the row . (Hint: check dropna() documentation). print the shape of the data now.(it should

In [17]:
```
data = pd.DataFrame(df.dropna(thresh=6))   # used dropna() function
data
```

Out[17]:

	datetime	state	country	shape	duration (seconds)	latitude	longitude	Unnamed: 7
0	10-10-1949 20:30	tx	us	cylinder	2700	29.8830556	-97.941111	NaN
1	10-10-1949 21:00	tx	NaN	light	7200	29.38421	-98.581082	NaN
2	10-10-1955 17:00	NaN	gb	circle	20	53.2	-2.916667	NaN
3	10-10-1956 21:00	tx	us	circle	20	28.9783333	-96.645833	NaN
4	10-10-1960 20:00	hi	us	light	900	21.4180556	-157.803611	NaN
...	...	...	...	...	...	...	...	...
88870	09-09-2013 22:00	ca	us	other	1200	38.2972222	-122.284444	NaN
88871	09-09-2013 22:20	va	us	circle	5	38.9011111	-77.265556	NaN
88872	09-09-2013 23:00	ok	us	cigar	1020	35.6527778	-97.477778	NaN
88873	09-09-2013 23:00	sc	us	diamond	0	34.3769444	-82.695833	NaN
88874	09-09-2013 23:30	fl	us	oval	0	26.1219444	-80.143611	NaN

83651 rows × 8 columns

In [19]:
```
data.shape   # shape of data
```

Out[19]:
```
(83651, 8)
```

Q-4) Find the countries where the most UFO sighting occurs.

```
In [21]: data1 = pd.DataFrame(df.groupby("country").count().iloc[:,0:1])
         data1     # group data into one frame
```

```
Out[21]:
                 datetime
         country
              au      593
              ca     3266
              de      112
              gb     2050
              us    70293
```

```
In [22]: data1.sort_values(by=['datetime'], inplace=True, ascending=False)
         data1  #sortind in ascending
```

```
Out[22]:
                 datetime
         country
              us    70293
              ca     3266
              gb     2050
              au      593
              de      112
```

Q-5) Create a sample dataframe the contains 5% random records(of original records) from the original dataset. Remove the rows with null values from the newly created dataframe and perform below operations. (consider the data in your dataframe as the dataset for below programs

In [26]:

```python
data = df.sample(frac=0.05)
data     # random 5% data
```

Out[26]:

	datetime	state	country	shape	duration (seconds)	latitude	longitude	Unnamed: 7
68527	07-04-1999 15:30	ca	us	other	480	38.5816667	-121.493333	NaN
48584	5/22/1996 24:00	oh	us	NaN	0	40.0333333	-83.158333	NaN
14004	11/23/2013 16:45	ca	us	NaN	10	34.3541667	-119.058333	NaN
78707	08-03-1998 17:53	tx	us	fireball	2	31.7586111	-106.486389	NaN
79825	08-07-2005 03:45	wy	us	light	2040	41.8955556	-106.204167	NaN
...	...	...	...	...	...	...	...	...
55233	6/17/2001 22:20	tn	us	fireball	120	36.595	-82.188889	NaN
2554	10/16/2002 20:45	tx	us	light	900	31.5491667	-97.146389	NaN
87981	09-06-2001 06:00	md	us	cigar	1800	39.4141667	-77.410833	NaN
47475	5/16/2011 00:30	mn	us	light	240	45.6091667	-94.451389	NaN
85589	9/24/1998 22:35	ak	us	sphere	600	64.7511111	-147.349444	NaN

4444 rows × 8 columns

In [27]:
```
data1 = pd.DataFrame(data.dropna(thresh=6))
data1
```

Out[27]:

	datetime	state	country	shape	duration (seconds)	latitude	longitude	Unnamed: 7
68527	07-04-1999 15:30	ca	us	other	480	38.5816667	-121.493333	NaN
48584	5/22/1996 24:00	oh	us	NaN	0	40.0333333	-83.158333	NaN
14004	11/23/2013 16:45	ca	us	NaN	10	34.3541667	-119.058333	NaN
78707	08-03-1998 17:53	tx	us	fireball	2	31.7586111	-106.486389	NaN
79825	08-07-2005 03:45	wy	us	light	2040	41.8955556	-106.204167	NaN
...	...	...	...	...	...	...	...	...
55233	6/17/2001 22:20	tn	us	fireball	120	36.595	-82.188889	NaN
2554	10/16/2002 20:45	tx	us	light	900	31.5491667	-97.146389	NaN
87981	09-06-2001 06:00	md	us	cigar	1800	39.4141667	-77.410833	NaN
47475	5/16/2011 00:30	mn	us	light	240	45.6091667	-94.451389	NaN
85589	9/24/1998 22:35	ak	us	sphere	600	64.7511111	-147.349444	NaN

4167 rows × 8 columns

Q-5(A) Write a program to count year wise frequency of reporting dates of unidentified flying object(UFO)

In [33]:
```
data1['datetime'] = pd.to_datetime(data1['datetime'] , errors = 'coerce')
data1['datetime']  # fetch data of datetime
```

Out[33]:
```
68527   1999-07-04 15:30:00
48584                   NaT
14004   2013-11-23 16:45:00
78707   1998-08-03 17:53:00
79825   2005-08-07 03:45:00
                ...
55233   2001-06-17 22:20:00
2554    2002-10-16 20:45:00
87981   2001-09-06 06:00:00
47475   2011-05-16 00:30:00
85589   1998-09-24 22:35:00
Name: datetime, Length: 4167, dtype: datetime64[ns]
```

In [37]:

```
data1['year'] = data1["datetime"].dt.year
data1['year']   # only fetch value of year
```

Out[37]:

```
68527    1999.0
48584       NaN
14004    2013.0
78707    1998.0
79825    2005.0
          ...
55233    2001.0
2554     2002.0
87981    2001.0
47475    2011.0
85589    1998.0
Name: year, Length: 4167, dtype: float64
```

In [36]:

```
print(data1['year'].value_counts().sort_values(ascending = True))
# sorting value of year by UFO values
```

```
1953.0      1
1944.0      1
1961.0      1
1954.0      1
1950.0      1
         ...
2005.0    227
2008.0    246
2011.0    271
2013.0    374
2012.0    395
Name: year, Length: 68, dtype: int64
```

Q-5(B) Write a program to get the current date , oldest date and number of days between current date and oldest date of sighting from UFO dataset.

In [48]:

```
print("Current Date : ",data1.datetime.max())  # current date for dataset
current = data1.datetime.max()
current
```

```
Current Date :  2014-05-06 21:00:00
```

Out[48]:

```
Timestamp('2014-05-06 21:00:00')
```

In [47]:

```
print("\nOldest Date: ", data1.datetime.min())   # oldest date for dataset
old = data1.datetime.min()
old
```

```
Oldest Date:  1936-07-15 00:00:00
```

Out[47]:

```
Timestamp('1936-07-15 00:00:00')
```

In [49]:
```
print("Diiffrence of days between oldest date and current date of dataset : ",
(current - old).days)
```

```
Diiffrence of days between oldest date and current date of dataset :  28419
```

Q-5(C) Write a program to get all the sighting dates of the uidentified flying object(UFO) from last 20 years (last 365*20 days)

In [53]:
```
import datetime

current1 = pd.to_datetime('today')
current1                      # today's time
```

Out[53]:
```
Timestamp('2020-08-31 16:04:32.630486')
```

In [58]:
```
timedur = datetime.timedelta(days = 365*20)
timedur     # 365*20 days
```

Out[58]:
```
datetime.timedelta(days=7300)
```

In [57]:
```
print("Result : ")
print(data1[current1-data1['datetime'] <= timedur])
```

```
Result :
                 datetime state country     shape duration (seconds)  \
14004 2013-11-23 16:45:00    ca      us       NaN                 10
79825 2005-08-07 03:45:00    wy      us     light               2040
15938 2003-11-04 19:30:00    mo      us     light                 60
31652 2002-02-25 12:00:00    ca      us    sphere                180
77098 2010-08-25 05:30:00    co      us    sphere               3600
...                   ...   ...     ...       ...                ...
16722 2011-01-15 22:40:00    vt      us  fireball                600
55233 2001-06-17 22:20:00    tn      us  fireball                120
2554  2002-10-16 20:45:00    tx      us     light                900
87981 2001-09-06 06:00:00    md      us     cigar               1800
47475 2011-05-16 00:30:00    mn      us     light                240

         latitude   longitude  Unnamed: 7    year
14004  34.3541667 -119.058333         NaN  2013.0
79825  41.8955556 -106.204167         NaN  2005.0
15938  37.1272222  -90.450000         NaN  2003.0
31652  39.2191667 -121.060000         NaN  2002.0
77098  38.4783333 -107.875556         NaN  2010.0
...           ...         ...         ...     ...
16722  44.1719444  -72.651389         NaN  2011.0
55233      36.595  -82.188889         NaN  2001.0
2554   31.5491667  -97.146389         NaN  2002.0
87981  39.4141667  -77.410833         NaN  2001.0
47475  45.6091667  -94.451389         NaN  2011.0

[3205 rows x 9 columns]
```