```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#get the url
URL = "https://cmsc320.github.io/files/top-50-solar-flares.html"
r = requests.get(url = URL)
#extract the text
text = BeautifulSoup(r.text, 'html.parser')
#read the data
table = text.find('table')
data = pd.concat(pd.read_html(str(table), flavor="bs4"))
#set the name
data.columns = ['rank', 'x_classification', 'date', 'region',
'start_time', 'maximum_time', 'end_time', 'movie']
data
```

```
    rank x_classification        date  ...  maximum_time end_time 
movie
0      1             X28+  2003/11/04  ...         19:53    20:06 
MovieView archive
1      2             X20+  2001/04/02  ...         21:51    22:03 
MovieView archive
2      3           X17.2+  2003/10/28  ...         11:10    11:24 
MovieView archive
3      4             X17+  2005/09/07  ...         17:40    18:03 
MovieView archive
4      5            X14.4  2001/04/15  ...         13:50    13:55 
MovieView archive
5      6              X10  2003/10/29  ...         20:49    21:01 
MovieView archive
6      7             X9.4  1997/11/06  ...         11:55    12:01 
MovieView archive
7      8             X9.3  2017/09/06  ...         12:02    12:10 
MovieView archive
8      9               X9  2006/12/05  ...         10:35    10:45 
MovieView archive
9     10             X8.3  2003/11/02  ...         17:25    17:39 
MovieView archive
10    11             X8.2  2017/09/10  ...         16:06    16:31 
MovieView archive
11    12             X7.1  2005/01/20  ...         07:01    07:26 
MovieView archive
12    13             X6.9  2011/08/09  ...         08:05    08:08 
MovieView archive
13    14             X6.5  2006/12/06  ...         18:47    19:00 
MovieView archive
14    15             X6.2  2005/09/09  ...         20:04    20:36 
MovieView archive
```

```
15    16             X6.2  2001/12/13  ...         14:30    14:35
MovieView archive
16    17             X5.7  2000/07/14  ...         10:24    10:43
MovieView archive
17    18             X5.6  2001/04/06  ...         19:21    19:31
MovieView archive
18    19             X5.4  2012/03/07  ...         00:24    00:40
MovieView archive
19    20             X5.4  2005/09/08  ...         21:06    21:17
MovieView archive
20    21             X5.4  2003/10/23  ...         08:35    08:49
MovieView archive
21    22             X5.3  2001/08/25  ...         16:45    17:04
MovieView archive
22    23             X4.9  2014/02/25  ...         00:49    01:03
MovieView archive
23    24             X4.9  1998/08/18  ...         22:19    22:28
View archive
24    25             X4.8  2002/07/23  ...         00:35    00:47
MovieView archive
25    26               X4  2000/11/26  ...         16:48    16:56
MovieView archive
26    27             X3.9  2003/11/03  ...         09:55    10:19
MovieView archive
27    28             X3.9  1998/08/19  ...         21:45    21:50
View archive
28    29             X3.8  2005/01/17  ...         09:52    10:07
MovieView archive
29    30             X3.7  1998/11/22  ...         06:42    06:49
MovieView archive
30    31             X3.6  2005/09/09  ...         09:59    10:08
MovieView archive
31    32             X3.6  2004/07/16  ...         13:55    14:01
MovieView archive
32    33             X3.6  2003/05/28  ...         00:27    00:39
MovieView archive
33    34             X3.4  2006/12/13  ...         02:40    02:57
MovieView archive
34    35             X3.4  2001/12/28  ...         20:45    21:32
MovieView archive
35    36             X3.3  2013/11/05  ...         22:12    22:15
MovieView archive
36    37             X3.3  2002/07/20  ...         21:30    21:54
MovieView archive
37    38             X3.3  1998/11/28  ...         05:52    06:13
MovieView archive
38    39             X3.2  2013/05/14  ...         01:11    01:20
MovieView archive
39    40             X3.1  2014/10/24  ...         21:41    22:13
MovieView archive
```

```
40    41              X3.1  2002/08/24  ...         01:12    01:31
MovieView archive
41    42                X3  2002/07/15  ...         20:08    20:14
MovieView archive
42    43              X2.8  2013/05/13  ...         16:05    16:16
MovieView archive
43    44              X2.8  2001/12/11  ...         08:08    08:14
MovieView archive
44    45              X2.8  1998/08/18  ...         08:24    08:32
View archive
45    46              X2.7  2015/05/05  ...         22:11    22:15
MovieView archive
46    47              X2.7  2003/11/03  ...         01:30    01:45
MovieView archive
47    48              X2.7  1998/05/06  ...         08:09    08:20
MovieView archive
48    49              X2.6  2005/01/15  ...         23:02    23:31
MovieView archive
49    50              X2.6  2001/09/24  ...         10:38    11:09
MovieView archive

[50 rows x 8 columns]
```

```
#drop the last column of the table
data.pop(data.columns[-1])
#combine the date
data['start_datetime'] = pd.to_datetime(data['date'] + ' ' +
data['start_time'])
data['max_datetime'] = pd.to_datetime(data['date'] + ' ' +
data['maximum_time'])
data['end_datetime'] = pd.to_datetime(data['date'] + ' ' +
data['end_time'])
#drop the unnessary columns
dara = data.drop(['date','start_time', 'maximum_time', 'end_time'],
axis=1)
#rearrange the order
data = data[['rank', 'x_classification', 'start_datetime',
'max_datetime', 'end_datetime', 'region']]
#set regions coded as - as NaN
data.replace('-', 'NaN')
data
```

```
    rank x_classification  ...        end_datetime region
0      1             X28+  ... 2003-11-04 20:06:00    486
1      2             X20+  ... 2001-04-02 22:03:00   9393
2      3           X17.2+  ... 2003-10-28 11:24:00    486
3      4             X17+  ... 2005-09-07 18:03:00    808
4      5            X14.4  ... 2001-04-15 13:55:00   9415
5      6              X10  ... 2003-10-29 21:01:00    486
6      7             X9.4  ... 1997-11-06 12:01:00   8100
```

```
7      8             X9.3  ... 2017-09-06 12:10:00   2673
8      9               X9  ... 2006-12-05 10:45:00    930
9     10             X8.3  ... 2003-11-02 17:39:00    486
10    11             X8.2  ... 2017-09-10 16:31:00   2673
11    12             X7.1  ... 2005-01-20 07:26:00    720
12    13             X6.9  ... 2011-08-09 08:08:00   1263
13    14             X6.5  ... 2006-12-06 19:00:00    930
14    15             X6.2  ... 2005-09-09 20:36:00    808
15    16             X6.2  ... 2001-12-13 14:35:00   9733
16    17             X5.7  ... 2000-07-14 10:43:00   9077
17    18             X5.6  ... 2001-04-06 19:31:00   9415
18    19             X5.4  ... 2012-03-07 00:40:00   1429
19    20             X5.4  ... 2005-09-08 21:17:00    808
20    21             X5.4  ... 2003-10-23 08:49:00    486
21    22             X5.3  ... 2001-08-25 17:04:00   9591
22    23             X4.9  ... 2014-02-25 01:03:00   1990
23    24             X4.9  ... 1998-08-18 22:28:00   8307
24    25             X4.8  ... 2002-07-23 00:47:00     39
25    26               X4  ... 2000-11-26 16:56:00   9236
26    27             X3.9  ... 2003-11-03 10:19:00    488
27    28             X3.9  ... 1998-08-19 21:50:00   8307
28    29             X3.8  ... 2005-01-17 10:07:00    720
29    30             X3.7  ... 1998-11-22 06:49:00   8384
30    31             X3.6  ... 2005-09-09 10:08:00    808
31    32             X3.6  ... 2004-07-16 14:01:00    649
32    33             X3.6  ... 2003-05-28 00:39:00    365
33    34             X3.4  ... 2006-12-13 02:57:00    930
34    35             X3.4  ... 2001-12-28 21:32:00   9767
35    36             X3.3  ... 2013-11-05 22:15:00   1890
36    37             X3.3  ... 2002-07-20 21:54:00     39
37    38             X3.3  ... 1998-11-28 06:13:00   8395
38    39             X3.2  ... 2013-05-14 01:20:00   1748
39    40             X3.1  ... 2014-10-24 22:13:00   2192
40    41             X3.1  ... 2002-08-24 01:31:00     69
41    42               X3  ... 2002-07-15 20:14:00     30
42    43             X2.8  ... 2013-05-13 16:16:00   1748
43    44             X2.8  ... 2001-12-11 08:14:00   9733
44    45             X2.8  ... 1998-08-18 08:32:00   8307
45    46             X2.7  ... 2015-05-05 22:15:00   2339
46    47             X2.7  ... 2003-11-03 01:45:00    488
47    48             X2.7  ... 1998-05-06 08:20:00   8210
48    49             X2.6  ... 2005-01-15 23:31:00    720
49    50             X2.6  ... 2001-09-24 11:09:00   9632

[50 rows x 6 columns]
```

```python
#get the url
URL = "https://cmsc320.github.io/files/waves_type2.html"
r = requests.get(url = URL)
#extract the text
```

```
text = BeautifulSoup(r.text, 'html.parser')
#read the data
new_data = text.find('pre').get_text().splitlines()
del new_data[0:12]
del new_data[-1:]
#set the name
new_data = pd.DataFrame(new_data)
new_data[['start_date', 'start_time', 'end_date', 'end_time',
'start_frequency',
'end_frequency', 'flare_location', 'flare_region',
'flare_classification',
'cme_date', 'cme_time', 'cme_angle', 'cme_width', 'cme_speed', 'del1',

'del2', 'del3','del4','del5','del6','del7','del8','del9','del10']] =
new_data[0].str.split(expand=True)
#drop useless columns
new_data = new_data.iloc[: ,1:]
new_data.drop(new_data.loc[:, 'del1':'del10'].columns, axis = 1,
inplace = True)
new_data
```

```
     start_date start_time end_date  ... cme_angle cme_width cme_speed
0    1997/04/01      14:00    04/01  ...        74        79       312
1    1997/04/07      14:30    04/07  ...      Halo       360       878
2    1997/05/12      05:15    05/14  ...      Halo       360       464
3    1997/05/21      20:20    05/21  ...       263       165       296
4    1997/09/23      21:53    09/23  ...       133       155       712
..          ...        ...      ...  ...       ...       ...       ...
513  2017/09/04      20:27    09/05  ...      Halo       360      1418
514  2017/09/06      12:05    09/07  ...      Halo       360      1571
515  2017/09/10      16:02    09/11  ...      Halo       360      3163
516  2017/09/12      07:38    09/12  ...       124        96       252
517  2017/09/17      11:45    09/17  ...      Halo       360      1385

[518 rows x 14 columns]
```

```
#set any missing entries as NaN
new_data.replace(['----', '-----', '------', '--/--', '--:--',
'????'], 'NaN', inplace = True)
#create a new column that indicates if a row corresponds to a halo
flare or not
is_halo = []
for row, col in new_data.iterrows():
    col = col.tolist()
    if (col[11] == 'Halo'):
        is_halo.append("true")
    else:
        is_halo.append("false")
new_data['is_halo'] = is_halo
#replace Halo entries in the cme_angle column as NA
```

```
new_data['cme_angle'].replace('Halo', 'NA')
#create a new column that indicates if width is given as a lower bound
with_lower_bound = []
for row, col in new_data.iterrows():
    col = col.tolist()
    if (('>') in col[12]):
        with_lower_bound.append("true")
    else:
        with_lower_bound.append("false")
new_data['with_lower_bound'] = with_lower_bound
#remove any non-numeric part of the width column
width = []
for row, col in new_data.iterrows():
    col = col.tolist()
    if (col[12].isnumeric()):
        width.append(col[12])
    else:
        width.append("")
new_data['cme_width'] = width
#combine date and time columns for start, end and cme
new_data['end_time'].replace('24:00', '00:00', inplace = True)
start_datetime = []
end_datetime = []
cme_datetime = []
for row, col in new_data.iterrows():
    year = col['start_date'][:5]
    # start_datetime
    col['start_date'] = col['start_date'] + ' ' + col['start_time']
    col['start_date'] = pd.to_datetime(col['start_date'],
errors='coerce', format='%Y-%m-%d %H:%M:%S')
    # end_datetime
    col['end_date'] = year + col['end_date'] + ' ' + col['end_time']
    col['end_date'] = pd.to_datetime(col['end_date'], errors='coerce',
format='%Y-%m-%d %H:%M:%S')
    # cme_datetime
    col['cme_date'] = year + col['cme_date'] + ' ' + col['cme_time']
    col['cme_date'] = pd.to_datetime(col['cme_date'], errors='coerce',
format='%Y-%m-%d %H:%M:%S')
#drop the unnessary columns
new_data.drop(['start_time', 'end_time', 'cme_time'], axis = 1,
inplace = True)
new_data = new_data.rename(columns={'start_date' : 'start_datetime',
'end_date' : 'end_datetime', 'cme_date' : 'cme_datetime'})
new_data
```

```
          start_datetime         end_datetime  ... is_halo
with_lower_bound
0    1997-04-01 14:00:00  1997-04-01 14:15:00  ...   false
false
1    1997-04-07 14:30:00  1997-04-07 17:30:00  ...    true
```

```
false
2    1997-05-12 05:15:00  1997-05-14 16:00:00  ...    true
false
3    1997-05-21 20:20:00  1997-05-21 22:00:00  ...   false
false
4    1997-09-23 21:53:00  1997-09-23 22:16:00  ...   false
false
..                   ...                  ...  ...      ...
...
513  2017-09-04 20:27:00  2017-09-05 04:54:00  ...    true
false
514  2017-09-06 12:05:00  2017-09-07 08:00:00  ...    true
false
515  2017-09-10 16:02:00  2017-09-11 06:50:00  ...    true
false
516  2017-09-12 07:38:00  2017-09-12 07:43:00  ...   false
false
517  2017-09-17 11:45:00  2017-09-17 12:35:00  ...    true
false

[518 rows x 13 columns]
```

```python
#for question 1, I get all data start with 'X' at first
#then, I extract the data with 'X' from the table
#sort the value and get the the top 50
#while the flare class from the first dataset ranges from 2.6 to 28.0
#the NASA dataset has flare class ranging from 1.9 to 28.0
top50 = []
for row, col in new_data.iterrows():
    col = col.tolist()
    if (('X') in col[6]):
        top50.append("true")
    else:
        top50.append("false")
new_data['top50'] = top50
nasa_top50 = new_data[new_data['top50'] == 'true']
nasa_top50['classification'] =
(nasa_top50['flare_classification'].str)[1:]
nasa_top50['classification'] =
nasa_top50['classification'].astype(float)
# sort the value in decending order
nasa_top50 = nasa_top50.sort_values(['classification'], ascending =
False)
nasa_top50.drop(['top50', 'classification'], axis = 1, inplace = True)
nasa_top50 = nasa_top50.head(50)
nasa_top50
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  from ipykernel import kernelapp as app
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:16:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  app.launch_new_instance()

          start_datetime         end_datetime  ... is_halo
with_lower_bound
240  2003-11-04 20:00:00  2003-11-04 00:00:00  ...    true
false
117  2001-04-02 22:05:00  2001-04-03 02:30:00  ...   false
false
233  2003-10-28 11:10:00  2003-10-29 00:00:00  ...    true
false
126  2001-04-15 14:05:00  2001-04-16 13:00:00  ...   false
false
234  2003-10-29 20:55:00  2003-10-29 00:00:00  ...    true
false
8    1997-11-06 12:20:00  1997-11-07 08:30:00  ...    true
false
514  2017-09-06 12:05:00  2017-09-07 08:00:00  ...    true
false
328  2006-12-05 10:50:00  2006-12-05 20:00:00  ...   false
false
237  2003-11-02 17:30:00  2003-11-03 01:00:00  ...    true
false
515  2017-09-10 16:02:00  2017-09-11 06:50:00  ...    true
false
288  2005-01-20 07:15:00  2005-01-20 16:30:00  ...    true
false
359  2011-08-09 08:20:00  2011-08-09 08:35:00  ...    true
false
331  2006-12-06 19:00:00  2006-12-08 00:00:00  ...   false
false
317  2005-09-09 19:45:00  2005-09-09 22:00:00  ...    true
false
82   2000-07-14 10:30:00  2000-07-15 14:30:00  ...    true
false
121  2001-04-06 19:35:00  2001-04-07 01:50:00  ...    true
false
```

```
375  2012-03-07 01:00:00  2012-03-08 19:00:00  ...    true
false
135  2001-08-25 16:50:00  2001-08-25 23:00:00  ...    true
false
443  2014-02-25 00:56:00  2014-02-25 11:28:00  ...    true
false
193  2002-07-23 00:50:00  2002-07-23 04:00:00  ...    true
false
104  2000-11-26 17:00:00  2000-11-26 17:15:00  ...    true
false
239  2003-11-03 10:00:00  2003-11-03 12:30:00  ...   false
false
286  2005-01-17 10:00:00  2005-01-17 10:35:00  ...    true
false
222  2003-05-28 01:00:00  2003-05-29 00:30:00  ...    true
false
332  2006-12-13 02:45:00  2006-12-13 10:40:00  ...    true
false
160  2001-12-28 20:35:00  2001-12-29 03:00:00  ...    true
false
192  2002-07-20 21:30:00  2002-07-20 22:20:00  ...    true
false
404  2013-05-14 01:16:00  2013-05-14 08:20:00  ...    true
false
201  2002-08-24 01:45:00  2002-08-24 03:25:00  ...    true
false
403  2013-05-13 16:15:00  2013-05-13 19:10:00  ...    true
false
487  2015-05-05 22:24:00  2015-05-05 23:14:00  ...    true
false
19   1998-05-06 08:25:00  1998-05-06 08:35:00  ...   false
false
238  2003-11-03 01:15:00  2003-11-03 01:25:00  ...   false
false
284  2005-01-15 23:00:00  2005-01-17 00:00:00  ...    true
false
142  2001-09-24 10:45:00  2001-09-25 20:00:00  ...    true
false
9    1997-11-27 13:30:00  1997-11-27 14:00:00  ...   false
false
276  2004-11-10 02:25:00  2004-11-10 03:40:00  ...    true
false
123  2001-04-10 05:24:00  2001-04-10 00:00:00  ...    true
false
99   2000-11-24 15:25:00  2000-11-24 22:00:00  ...    true
false
73   2000-06-06 15:20:00  2000-06-08 09:00:00  ...    true
false
345  2011-02-15 02:10:00  2011-02-15 07:00:00  ...    true
false
```

```
318  2005-09-10 21:45:00  2005-09-11 01:00:00  ...    true
false
361  2011-09-06 22:30:00  2011-09-07 15:40:00  ...    true
false
420  2013-10-25 15:08:00  2013-10-25 22:32:00  ...    true
false
7    1997-11-04 06:00:00  1997-11-05 04:30:00  ...    true
false
98   2000-11-24 05:10:00  2000-11-24 15:00:00  ...    true
false
125  2001-04-12 10:20:00  2001-04-12 10:40:00  ...    true
false
274  2004-11-07 16:25:00  2004-11-08 20:00:00  ...    true
false
285  2005-01-17 09:25:00  2005-01-17 16:00:00  ...    true
false
102  2000-11-25 19:00:00  2000-11-25 19:35:00  ...    true
false

[50 rows x 13 columns]
```

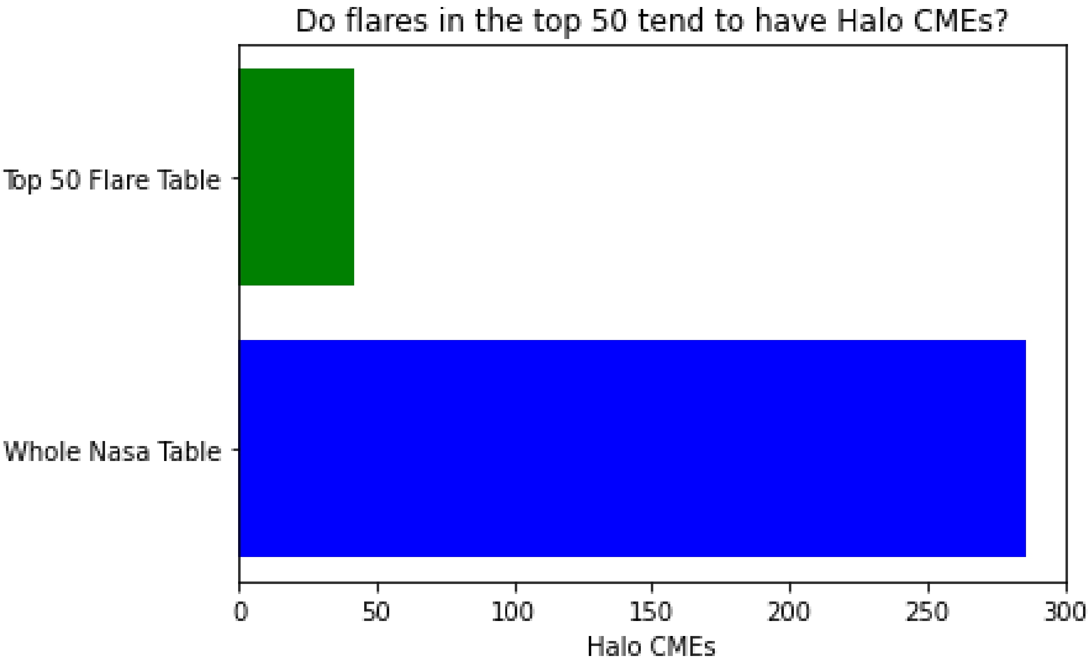
```
#for question 2, I use pd.merge to create a new table with the same
classification from the NASA and SpaceWeatherLive table
#if more than one SpaceWeatherLive entry "best matches", I will choose
the one with same start year
#I defined the best matching rows across the two datasets to be the
ones that have the same classification
#I found 51 such matches in the NASA and SpaceWeatherLive table
nasa_top50['x_classification'] = nasa_top50['flare_classification']
mergeData = pd.merge(data, nasa_top50, how = 'inner', on =
'x_classification')
mergeData
```

```
    rank x_classification  ... is_halo with_lower_bound
0      7             X9.4  ...    true            false
1      8             X9.3  ...    true            false
2     10             X8.3  ...    true            false
3     10             X8.3  ...    true            false
4     12             X7.1  ...    true            false
5     13             X6.9  ...    true            false
6     14             X6.5  ...   false            false
7     15             X6.2  ...    true            false
8     16             X6.2  ...    true            false
9     17             X5.7  ...    true            false
10    18             X5.6  ...    true            false
11    19             X5.4  ...    true            false
12    20             X5.4  ...    true            false
13    21             X5.4  ...    true            false
14    22             X5.3  ...    true            false
15    23             X4.9  ...    true            false
```

```
16    24             X4.9  ...    true            false
17    25             X4.8  ...    true            false
18    27             X3.9  ...   false            false
19    28             X3.9  ...   false            false
20    29             X3.8  ...    true            false
21    31             X3.6  ...    true            false
22    32             X3.6  ...    true            false
23    33             X3.6  ...    true            false
24    34             X3.4  ...    true            false
25    34             X3.4  ...    true            false
26    35             X3.4  ...    true            false
27    35             X3.4  ...    true            false
28    36             X3.3  ...    true            false
29    37             X3.3  ...    true            false
30    38             X3.3  ...    true            false
31    39             X3.2  ...    true            false
32    40             X3.1  ...    true            false
33    41             X3.1  ...    true            false
34    43             X2.8  ...    true            false
35    44             X2.8  ...    true            false
36    45             X2.8  ...    true            false
37    46             X2.7  ...    true            false
38    46             X2.7  ...   false            false
39    46             X2.7  ...   false            false
40    47             X2.7  ...    true            false
41    47             X2.7  ...   false            false
42    47             X2.7  ...   false            false
43    48             X2.7  ...    true            false
44    48             X2.7  ...   false            false
45    48             X2.7  ...   false            false
46    49             X2.6  ...    true            false
47    49             X2.6  ...    true            false
48    49             X2.6  ...   false            false
49    50             X2.6  ...    true            false
50    50             X2.6  ...    true            false
51    50             X2.6  ...   false            false

[52 rows x 19 columns]
```

```
#for question 3, I make a barplot that compares the number (or
proportion) of Halo CMEs in the top 50 flares vs. the dataset as a
whole
#the dataset as a whole has 286, and the top 50 flares has 42
nasa_count = 0
top50_count = 0
for i, j in new_data.iterrows():
  if (j['is_halo']) == 'true':
    nasa_count += 1
for i, j in nasa_top50.iterrows():
  if (j['is_halo']) == 'true':
    top50_count += 1
y_labels = [nasa_count, top50_count]
fig, ax = plt.subplots()
ax.barh([0, 1], y_labels, 0.8, color= ['blue', 'green'])
ax.set_title('Do flares in the top 50 tend to have Halo CMEs?')
ax.set_xlabel('Halo CMEs')
ax.set_yticks([0, 1])
ax.set_yticklabels(['Whole Nasa Table', 'Top 50 Flare Table'])
plt.show()
# print(nasa_count)
# print(top50_count)
```


Do flares in the top 50 tend to have Halo CMEs?
Top 50 Flare Table
Whole Nasa Table
Halo CMEs