Instructions

1. import NYC_Accidents_2020.csv as a DataFrame

- bring in the csv file by using read_csv from pandas; don't use any keyword arguments initially
- use a relative path as if your notebook were opened from the root of the repository if possible (../data/raw/NYC_Accidents_2020.csv)
- compare the resulting DataFrame against opening the spreadsheet in LibreOffice, Google Sheets, Excel, Numbers, etc.
- you should immediately see an issue with the import
- use a keyword argument from the docs to fix the issue
- in a markdown cell after the import, describe what fix had to be made to make the initial import usable

---

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

The first three rows of the file are general descriptions about the dataset. We hava to skip the first three rows when reading the csv to ensure that the dataset has the right column names.

In [ ]:
```
file_path = sys.path[0] +'/../data/raw/NYC_Accidents_2020.csv'
accidents = pd.read_csv(file_path, skiprows=3)
```

1. display columns and row samples

- show only the names of the columns
- show the first 5 rows
- show a random sampling of 5 rows
- show the last 5 rows

In [ ]:
```
print(f'The names of the columns:\n {accidents.columns}')
```

```
The names of the columns:
 Index(['CRASH DATE', 'CRASH TIME', 'BOROUGH', 'ZIP CODE', 'LATITUDE',
       'LONGITUDE', 'LOCATION', 'ON STREET NAME', 'CROSS STREET NAME',
       'OFF STREET NAME', 'NUMBER OF PERSONS INJURED',
       'NUMBER OF PERSONS KILLED', 'NUMBER OF PEDESTRIANS INJURED',
       'NUMBER OF PEDESTRIANS KILLED', 'NUMBER OF CYCLIST INJURED',
       'NUMBER OF CYCLIST KILLED', 'NUMBER OF MOTORIST INJURED',
       'NUMBER OF MOTORIST KILLED', 'CONTRIBUTING FACTOR VEHICLE 1',
       'CONTRIBUTING FACTOR VEHICLE 2', 'CONTRIBUTING FACTOR VEHICLE 3',
       'CONTRIBUTING FACTOR VEHICLE 4', 'CONTRIBUTING FACTOR VEHICLE 5',
       'COLLISION_ID', 'VEHICLE TYPE CODE 1', 'VEHICLE TYPE CODE 2',
       'VEHICLE TYPE CODE 3', 'VEHICLE TYPE CODE 4', 'VEHICLE TYPE CODE 5'],
      dtype='object')
```

In [ ]:
```
accidents.head(5)
```

Out[ ]:

	CRASH DATE	CRASH TIME	BOROUGH	ZIP CODE	LATITUDE	LONGITUDE	LOCATION	ON STREET NAME	CROSS STREET NAME	OFF STREET NAME	...	CONTRIBUTING FACTOR VEHICLE 2	CO
0	8/29/20	15:40:00	BRONX	10466.0	40.89210	-73.833760	POINT (-73.83376 40.8921)	PRATT AVENUE	STRANG AVENUE	NaN	...	Unspecified	
1	8/29/20	21:00:00	BROOKLYN	11221.0	40.69050	-73.919914	POINT (-73.919914 40.6905)	BUSHWICK AVENUE	PALMETTO STREET	NaN	...	Unspecified	
2	8/29/20	18:20:00	NaN	NaN	40.81650	-73.946556	POINT (-73.946556 40.8165)	8 AVENUE	NaN	NaN	...	NaN	
3	8/29/20	0:00:00	BRONX	10459.0	40.82472	-73.892960	POINT (-73.89296 40.82472)	NaN	NaN	1047 SIMPSON STREET	...	Unspecified	
4	8/29/20	17:10:00	BROOKLYN	11203.0	40.64989	-73.933890	POINT (-73.93389 40.64989)	NaN	NaN	4609 SNYDER AVENUE	...	Unspecified	

5 rows × 29 columns

In [ ]:
```
accidents.sample(5)
```

Out[ ]:

	CRASH DATE	CRASH TIME	BOROUGH	ZIP CODE	LATITUDE	LONGITUDE	LOCATION	ON STREET NAME	CROSS STREET NAME	OFF STREET NAME	...	CONTRIBUT FAC VEHICI
13646	7/16/20	11:02:00	QUEENS	11422.0	40.658768	-73.73768	POINT (-73.73768 40.658768)	249 STREET	145 AVENUE	NaN	...	Unspeci
69635	1/13/20	17:33:00	QUEENS	11413.0	40.682404	-73.75181	POINT (-73.75181 40.682404)	LUCAS STREET	SPRINGFIELD BOULEVARD	NaN	...	Unspeci
22319	6/14/20	19:50:00	NaN	NaN	40.697330	-73.78456	POINT (-73.78456 40.69733)	MERRICK BOULEVARD	NaN	NaN	...	Unspeci
46559	3/1/20	18:35:00	MANHATTAN	10036.0	40.758980	-73.99595	POINT (-73.99595 40.75898)	WEST 41 STREET	10 AVENUE	NaN	...	Unspeci
55975	2/10/20	16:00:00	QUEENS	11419.0	40.685400	-73.82947	POINT (-73.82947 40.6854)	LIBERTY AVENUE	113 STREET	NaN	...	Unspeci

5 rows × 29 columns

In [ ]:
```
accidents.tail(5)
```

Out[ ]:

	CRASH DATE	CRASH TIME	BOROUGH	ZIP CODE	LATITUDE	LONGITUDE	LOCATION	ON STREET NAME	CROSS STREET NAME	OFF STREET NAME	...	CONTRIBUT FACTOR VEHICL
74876	1/1/20	15:13:00	BRONX	10459.0	40.826810	-73.896510	POINT (-73.89651 40.82681)	NaN	NaN	1122 INTERVALE AVENUE	...	N
74877	1/1/20	8:00:00	BROOKLYN	11235.0	40.582935	-73.959210	POINT (-73.95921 40.582935)	NaN	NaN	3401 GUIDER AVENUE	...	Unspeci
74878	1/1/20	11:36:00	BRONX	10461.0	40.848553	-73.830055	POINT (-73.830055 40.848553)	NaN	NaN	1810 MAHAN AVENUE	...	Unspeci
74879	1/1/20	1:45:00	MANHATTAN	10017.0	40.753624	-73.969440	POINT (-73.96944 40.753624)	EAST 48 STREET	2 AVENUE	NaN	...	Dr Inattention/Distrac
74880	1/1/20	18:00:00	QUEENS	11367.0	40.726875	-73.830960	POINT (-73.83096 40.726875)	NaN	NaN	70-25 PARK DRIVE EAST	...	Dr Inattention/Distrac

5 rows × 29 columns

1. describe the rows and data types

- use any method to show:
  - each column
  - the type of each column
  - the number of non-missing values in each column
- in a markdown cell after displaying the column info:
  - list out the columns that look like they have the "wrong" (or too wide) type
  - and next to the column name, specify what type the column should probably be
  - lastly, preview the remainder of the instructions and write out any data transformations or cleaning that you think will be necessary to complete this part of the homework

---

In [ ]:
```
accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74881 entries, 0 to 74880
Data columns (total 29 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   CRASH DATE                      74881 non-null  object
 1   CRASH TIME                      74881 non-null  object
 2   BOROUGH                         49140 non-null  object
 3   ZIP CODE                        49134 non-null  float64
 4   LATITUDE                        68935 non-null  float64
 5   LONGITUDE                       68935 non-null  float64
 6   LOCATION                        68935 non-null  object
 7   ON STREET NAME                  55444 non-null  object
 8   CROSS STREET NAME               35681 non-null  object
 9   OFF STREET NAME                 19437 non-null  object
 10  NUMBER OF PERSONS INJURED       74881 non-null  int64
 11  NUMBER OF PERSONS KILLED        74881 non-null  int64
 12  NUMBER OF PEDESTRIANS INJURED   74881 non-null  int64
 13  NUMBER OF PEDESTRIANS KILLED    74881 non-null  int64
 14  NUMBER OF CYCLIST INJURED       74881 non-null  int64
 15  NUMBER OF CYCLIST KILLED        74881 non-null  int64
 16  NUMBER OF MOTORIST INJURED      74881 non-null  int64
 17  NUMBER OF MOTORIST KILLED       74881 non-null  int64
 18  CONTRIBUTING FACTOR VEHICLE 1   74577 non-null  object
 19  CONTRIBUTING FACTOR VEHICLE 2   59285 non-null  object
 20  CONTRIBUTING FACTOR VEHICLE 3   6765 non-null   object
 21  CONTRIBUTING FACTOR VEHICLE 4   1851 non-null   object
 22  CONTRIBUTING FACTOR VEHICLE 5   523 non-null    object
 23  COLLISION_ID                    74881 non-null  int64
 24  VEHICLE TYPE CODE 1             74246 non-null  object
 25  VEHICLE TYPE CODE 2             53638 non-null  object
 26  VEHICLE TYPE CODE 3             6424 non-null   object
 27  VEHICLE TYPE CODE 4             1771 non-null   object
 28  VEHICLE TYPE CODE 5             503 non-null    object
dtypes: float64(3), int64(9), object(17)
memory usage: 16.6+ MB
```

- columns that have too wide data type

  0 CRASH DATE date

  1 CRASH TIME date

  2 BOROUGH string

  3 ZIP CODE int

  6 LOCATION string

  7 ON STREET NAME string

  8 CROSS STREET NAME string

  9 OFF STREET NAME string

  18 CONTRIBUTING FACTOR VEHICLE 1 string

  19 CONTRIBUTING FACTOR VEHICLE 2 string

  20 CONTRIBUTING FACTOR VEHICLE 3 string

  21 CONTRIBUTING FACTOR VEHICLE 4 string

  22 CONTRIBUTING FACTOR VEHICLE 5 string

  24 VEHICLE TYPE CODE 1 string

  25 VEHICLE TYPE CODE 2 string

  26 VEHICLE TYPE CODE 3 string

  27 VEHICLE TYPE CODE 4 string

  28 VEHICLE TYPE CODE 5 string

- transformation/cleaning necessary for completing this homework: (TODO)

1. initial column (or row) clean-up

- remove at least two columns
  - in a markdown cell describe why the columns should be removed
  - show evidence (with code) of why each column should be removed
- rename or transform at least one column
  - in a markdown cell describe why the column(s) should be renamed
- (optional) do any other clean up you deem necessary to make the following work easier

---

- Remove column "LOCATION" since all information in this column is contained in column "LATITUDE" and "LONGITUDE".
- Remove the column "COLLISION_ID" since it's a identifier for each accident, so it's not helpful in answering questions in this homework.

In [ ]:
```
accidents.drop(['LOCATION', 'COLLISION_ID'], axis = 1, inplace = True)
```

rename all columns from upper case to lower case for convenience

In [ ]:
```
accidents.columns = accidents.columns.map(lambda x: x.lower())
```

make all values in the borough column lower case for convenience.

In [ ]:
```
accidents.borough = accidents.borough.str.lower()
```

1. determine the top three streets(Use the ON STREET NAME column) that had the most accidents

- it's ok to show more than 3 streets
- show the street name and the number of accidents occurred on each street
- document every step that you use to do this, including how the data was cleaned and/or transformed

In [ ]:
```
print('Top three streets that had the most accidents:')
accidents['on street name'].value_counts()[:3]
```

Out[ ]:
```
Top three streets that had the most accidents:
BELT PARKWAY                 1241
LONG ISLAND EXPRESSWAY        745
BROOKLYN QUEENS EXPRESSWAY    738
Name: on street name, dtype: int64
```

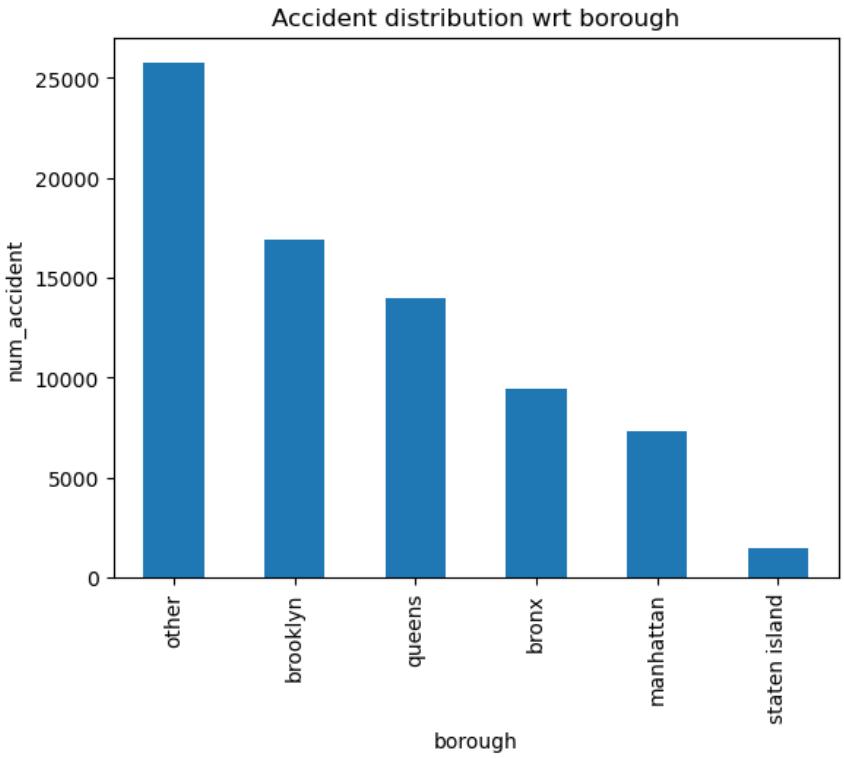
1. shows the number of accidents that occurred at each borough

- show a visualization that allows comparison of the number of the accidents.
  - BRONX
  - BROOKLYN
  - QUEENS
  - MANHATTAN
  - everything else can fall under "other" (including missing values)
- document every step that you use to do this, including how the data was cleaned and/or transformed
- hint, read the accompanying data dictionary / glossary

First I transformed all the values in borough column to lower case. Then I replace the nan value in this column with 'other'. After that I used the value_counts() function to obtain the number of accidents in each borough. Finally, I made a bar plot of the accident distribution with respect to different borough in NYC.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
accidents.borough.fillna('other').value_counts().plot.bar()
plt.title('Accident distribution wrt borough')
plt.xlabel('borough')
plt.ylabel('num_accident')
```

Out[ ]:
```
Text(0, 0.5, 'num_accident')
```



1. calculate summary statistics for the number of persons injured in all NYC and for a couple of selected boroughs (you can choose the two boroughs)

- use any method to calculate mean, median, percentiles (25 and 75), max, and min
- again, pick two boroughs
  - calculate summary statistics for each borough: use any method to calculate mean, median, percentiles (25 and 75), max, and min
  - in a markdown cell below the calculations, compare the results
- document every step that you use to do this, including how the data was cleaned and/or transformed

---

I first filter out accidents that occur in brooklyn, and then get their "number of persons injured" column valuesas a series. Then I compute the mean, median, 25 percentile, 75 percentile, min value and max value of the series.

Similarly, I first filter out accidents that occur in queens, and then get their "number of persons injured" column valuesas a series. Then I compute the mean, median, 25 percentile, 75 percentile, min value and max value of the series.

In [ ]:
```
num_injured_brooklyn = accidents[accidents['borough'] == 'brooklyn']['number of persons injured']
num_injured_queens = accidents[accidents['borough'] == 'queens']['number of persons injured']
print(f'Number of persons injured in an accident in Brooklyn \n \
    mean: {np.mean(num_injured_brooklyn)},\n \
    median: {np.median(num_injured_brooklyn)},\n \
    25 percentile: {np.percentile(num_injured_brooklyn, 25)},\n \
    75 percentile: {np.percentile(num_injured_brooklyn, 75)},\n \
    max: {np.max(num_injured_brooklyn)}, \n \
    min: {np.min(num_injured_brooklyn)}')

print(f'Number of persons injured in an accident in Queens \n \
    mean: {np.mean(num_injured_queens)},\n \
    median: {np.median(num_injured_queens)},\n \
    25 percentile: {np.percentile(num_injured_queens, 25)},\n \
    75 percentile: {np.percentile(num_injured_queens, 75)},\n \
    max: {np.max(num_injured_queens)}, \n \
    min: {np.min(num_injured_queens)}')
```

```
Number of persons injured in an accident in Brooklyn
     mean: 0.35630212338084816,
     median: 0.0,
     25 percentile: 0.0,
     75 percentile: 1.0,
     max: 15,
     min: 0
Number of persons injured in an accident in Queens
     mean: 0.32817293286723265,
     median: 0.0,
     25 percentile: 0.0,
     75 percentile: 1.0,
     max: 7,
     min: 0
```

Compare two results: The average number of persons injured in an accident in Brooklyn(0.35630) is slightly higher than that in Queens (0.32817). There is a maximum of 15 persons injured in an accident in Brroklyn, while the maximum number of persons injured in accidents occurred in Queens is 7. The 25 percentile, median, 75 percentile, and min value for the number of persons injured in an accident is the same for Brooklyns and Queens.

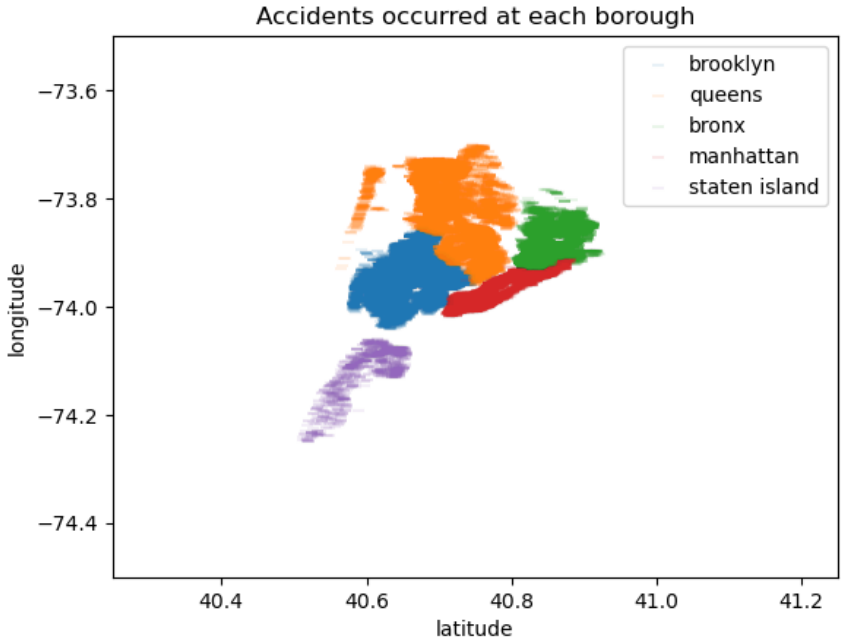
1. what are the distributions of accidents based on the geo location (latitude & longitude)?

- show a visualization that shows the accidents that occurred at each borough.
- that is plot the accidents based on the geo location, where x-axis is the latitude & y-axis is the longitude. And then differentiate the points by borough(by point color).
- document every step that you use to do this, including how the data was cleaned and/or transformed

---

For each borough, I filter out accidents that occurred in this borough and obtain its latitude and longitude series. I then plot a scatter plot using one color. Then I switch to another boroughs, follow the same procedure but plot with another color. The process is repeated five times until I finish looping over all boroughs. Note that when I plot the scatter plot, I restrict the x range from 40.25 to 41.25 and y range from -74.5 to -73.5 for clarity. zeros are dropped since it might have special meanings rather than representing the true data. I set alpha to 0.1 to reflect accident frequency in each location. Denser area has more number of accidents.

In [ ]:
```
colors = ['red', 'green', 'blue', 'purple', 'orange']
boroughs = list(accidents.borough.value_counts().index)
for i in range(len(boroughs)):
    latitude = accidents[accidents['borough'] == boroughs[i]].latitude
    longitude = accidents[accidents['borough'] == boroughs[i]].longitude
    plt.scatter(latitude, longitude, marker = 1, label = boroughs[i], alpha = 0.1)
    plt.xlim(40.25, 41.25)
    plt.ylim(-74.5, -73.5)
plt.legend()
plt.title("Accidents occurred at each borough")
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.show()
```



1. shows the covariance between each pair of the columns

- choose the columns that you think are necessary & explain your choices
- document every step that you use to do this, including how the data was cleaned and/or transformed

---

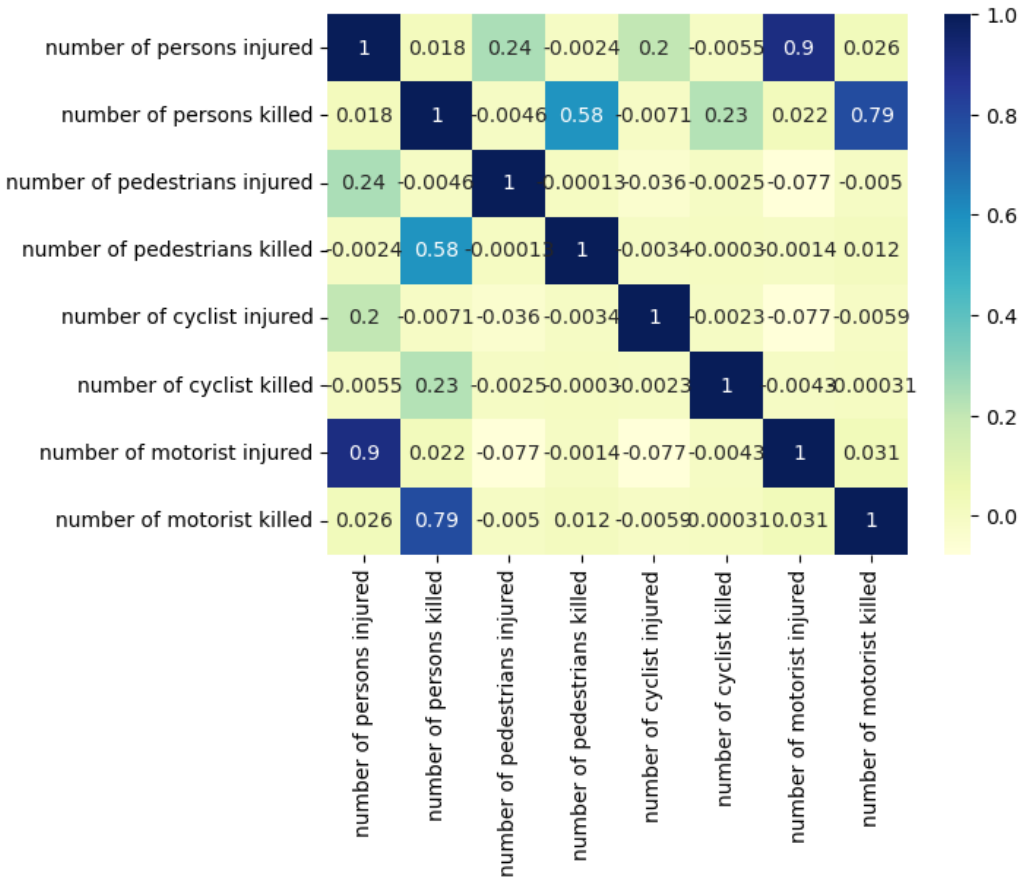
I chose columns of numeric type in this dataset. I think the number of persons/pedestrians/cyclist/motorist injured/ killed should be correlated, because those numbers are likely all to be high in a serious accident, while being all low in a minor accident. .

I first select the columns that I believe should be correlated into a dataframe. Then I call .corr to calculate the correlation between each features. Then I visualize the correlation by plotting the correlation heatmap.

In [ ]:
```
import seaborn as sns
corr = accidents[['number of persons injured', 'number of persons killed',
       'number of pedestrians injured', 'number of pedestrians killed',
       'number of cyclist injured', 'number of cyclist killed',
       'number of motorist injured', 'number of motorist killed']].corr()
sns.heatmap(corr, annot=True, cmap="YlGnBu")
```

```
/Users/yge/opt/anaconda3/envs/dma/lib/python3.8/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version
>=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.1
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

Out[ ]:
```
<AxesSubplot:>
```



1. which month did the most number of accidents occur?

- document every step that you use to do this, including how the data was cleaned and/or transformed
- the calendar module and month_abbr may be useful for labels
- it's ok to show more than one month
- optionally, visualize this data instead of simply listing the counts
- in a markdown cell, what can you conclude about when accidents reach a lull?

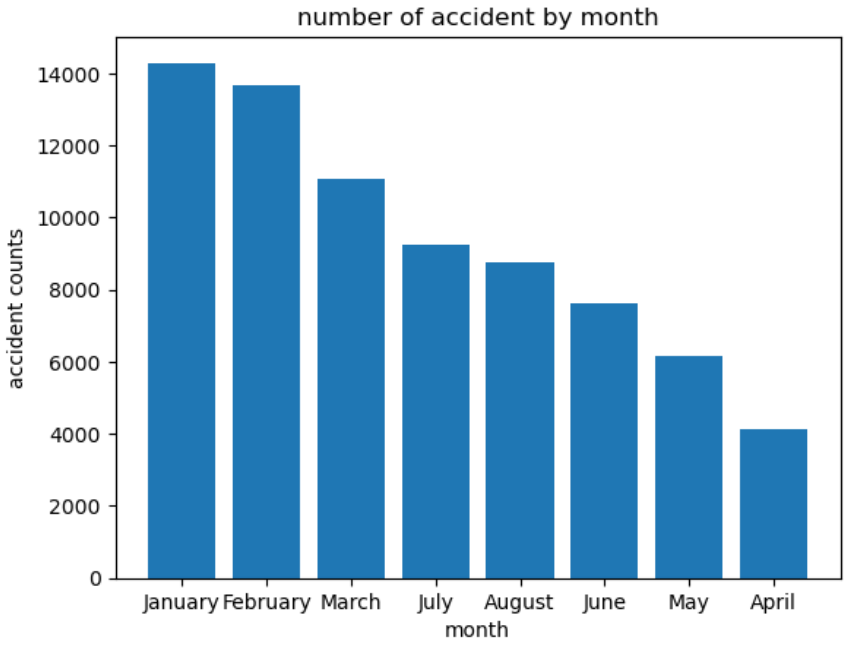
I transform the column "crash date" to datetime type to obtain the distribution of months and get its value counts. Then I create a month map that maps month number with month names from the calendar module. Then I visualize by plotting a bar plot of the number of accidents occuring each month.

In [ ]:
```
import calendar

month_accident = pd.to_datetime(accidents['crash date']).dt.month.value_counts()
month_map = {i: list(calendar.month_name)[i] for i in range(12)}
month_accident.index = [month_map[i] for i in month_accident.index]
```

In [ ]:
```
plt.bar(month_accident.index, month_accident.values)
plt.xlabel('month')
plt.ylabel('accident counts')
plt.title('number of accident by month')
```

Out[ ]:
```
Text(0.5, 1.0, 'number of accident by month')
```



The accidents reach a lull in Spring months (April, May, June)