Project - Dallas Shootings

Goal of Project

- A newspaper want a visualization of the shootings in Dallas with focus on subjects
- We will read data from from database and join into broader datasets
- We will explore ideas to visualize it and create a map with the shootings

Step 1: Acquire

- Explore problem
- Identify data
- Import data

Step 1.a: Import libraries

- Execute the cell below (SHIFT + ENTER)

In [16]:
```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

Step 1.b: Connect to SQLite database

- Use `sqlite3.connect(<filename>)` to connect to the database `files/dallas-ois.sqlite`

In [17]:
```
conn = sqlite3.connect('files/dallas-ois.sqlite')
```

Step 1. c: Read data into DataFrames

- The database consists of 3 tables: `incident`, `officers`, `subjects`
- Read the data from each table into a DataFrame.
- HINT: Use `pandas` `read_sql(sql_stmt, conn)`, which takes the SQL statement (`sql_stmt`) and the data base connection `conn` from previous step.
- HINT: The structure of the SQL statement is as follows.

  ```
  SELECT * FROM table
  ```

In [18]:
```
incidents = pd.read_sql('SELECT * FROM incidents', conn)
officers = pd.read_sql('SELECT * FROM officers', conn)
subjects = pd.read_sql('SELECT * FROM subjects', conn)
```

Step 1.d: Explore the length of the DataFrames

- What is the length of the DataFrames
  - HINT: Apply `len(...)` on the DataFrames
- We want to explore data based on officers and data based on subjects, both with incident data.
- Notice: It is difficult to create one dataset for both problems
- Explore data further to understand why

In [19]:
```
len(incidents), len(officers), len(subjects)
```

Out[19]:
```
(219, 370, 223)
```

In [20]:
```
incidents.head()
```

Out[20]:

	case_number	date	location	subject_statuses	subject_weapon	subjects	subject_count	o
0	44523A	2013-02-23	3000 Chihuahua Street	Injured	Handgun	Curry, James L/M	1	M Fil Briar
1	121982X	2010-05-03	1300 N. Munger Boulevard	Injured	Handgun	Chavez, Gabriel L/M	1	
2	605484T	2007-08-12	200 S. Stemmons Freeway	Other	Shotgun	Salinas, Nick L/M	1	Jerr
3	384832T	2007-05-26	7900 S. Loop 12	Shoot and Miss	Unarmed	Smith, James B/M; Dews, Antonio B/M; Spearman,...	3	M
4	244659R	2006-04-03	6512 South Loop 12	Injured	Hands	Watkins, Caleb B/M	1	Arms M

Step 1.e: Read data into dataset

- Create first dataset `subject_incidents` as officers joined with incidents.
  - What does this dataset give us?
  - HINT:

```
SELECT * FROM table_1 JOIN table_2 ON
table_1.column_name_1=table_2.column_name_2
```

  - HINT: You can join on columns `case_number`
- Is all data represented?

In [21]:
```
subjects_incidents = pd.read_sql('SELECT * FROM subjects s JOIN incidents i ON s.case_
```

In [22]:
```
len(subjects_incidents)
```

Out[22]:
```
223
```

In [23]:
```
subjects_incidents.head()
```

Out[23]:

	case_number	race	gender	last_name	first_name	full_name	case_number	date	location	sul
0	44523A	L	M	Curry	James	Curry, James	44523A	2013-02-23	3000 Chihuahua Street	
1	121982X	L	M	Chavez	Gabriel	Chavez, Gabriel	121982X	2010-05-03	1300 N. Munger Boulevard	
2	605484T	L	M	Salinas	Nick	Salinas, Nick	605484T	2007-08-12	200 S. Stemmons Freeway	
3	384832T	B	M	Smith	James	Smith, James	384832T	2007-05-26	7900 S. Loop 12	S
4	384832T	B	M	Dews	Antonio	Dews, Antonio	384832T	2007-05-26	7900 S. Loop 12	S

5 rows × 21 columns

Step 2: Prepare

- Explore data
- Visualize ideas
- Cleaning data

Step 2.a: Check the data types

- A step to get to understand the data better is to explore the data types
- Get the data types by `.dtypes`

In [24]:
```
subjects_incidents.dtypes
```

Out[24]:
```
case_number                 object
race                        object
gender                      object
last_name                   object
first_name                  object
full_name                   object
case_number                 object
date                        object
location                    object
subject_statuses            object
subject_weapon              object
subjects                    object
subject_count                int64
officers                    object
officer_count                int64
grand_jury_disposition      object
attorney_general_forms_url  object
summary_url                 object
summary_text                object
latitude                   float64
longitude                  float64
dtype: object
```

Step 2.b: Check for null (missing) values

- Data often is missing entries - there can be many reasons for this
- We need to deal with that (will do later in course)
- Use `.isna().sum()`

In [25]:
```
subjects_incidents.isnull().sum()
```

Out[25]:
```
case_number                  0
race                         0
gender                       0
last_name                    0
first_name                  18
full_name                    0
case_number                  0
date                         0
location                     0
subject_statuses             0
subject_weapon               0
subjects                     0
subject_count                0
officers                     0
officer_count                0
grand_jury_disposition      88
attorney_general_forms_url  221
summary_url                  3
summary_text                 3
latitude                     9
longitude                    9
dtype: int64
```

Step 2.c: Explore subject_statuses column

- As part of finding useful features let's explore `subject_statuses`
- We know that `subject_statuses` is categorical.
- Therefore we can use `groupby(...)` and `count()`

In [26]:
```
subjects_incidents.groupby('subject_statuses').count()
```

Out[26]:

subject_statuses	case_number	race	gender	last_name	first_name	full_name	case_number	date	l
1 Deceased 1 Injured	2	2	2	2	2	2	2	2	
2 Injured	1	1	1	1	1	1	1	1	
Deceased	69	69	69	69	67	69	69	69	
Deceased Injured	2	2	2	2	2	2	2	2	
Injured	60	60	60	60	60	60	60	60	
Other	2	2	2	2	2	2	2	2	
Shoot and Miss	87	87	87	87	71	87	87	87	

Step 2.d: Explore race column

- Repeat of previous step on column `race`

In [27]:
```
subjects_incidents.groupby('race').count()
```

Out[27]:

race	case_number	gender	last_name	first_name	full_name	case_number	date	location	subject_s
A	2	2	2	2	2	2	2	2	
B	111	111	111	104	111	111	111	111	
L	72	72	72	61	72	72	72	72	
W	38	38	38	38	38	38	38	38	

Step 2.e: Explore more columns

- Feel free to explore more columns (also called features)

In [46]:
```
subjects_incidents.columns
```

Out[46]:
```
Index(['case_number', 'race', 'gender', 'last_name', 'first_name', 'full_name',
       'case_number', 'date', 'location', 'subject_statuses', 'subject_weapon',
       'subjects', 'subject_count', 'officers', 'officer_count',
       'grand_jury_disposition', 'attorney_general_forms_url', 'summary_url',
       'summary_text', 'latitude', 'longitude'],
      dtype='object')
```

Step 2.f: Visualize ideas

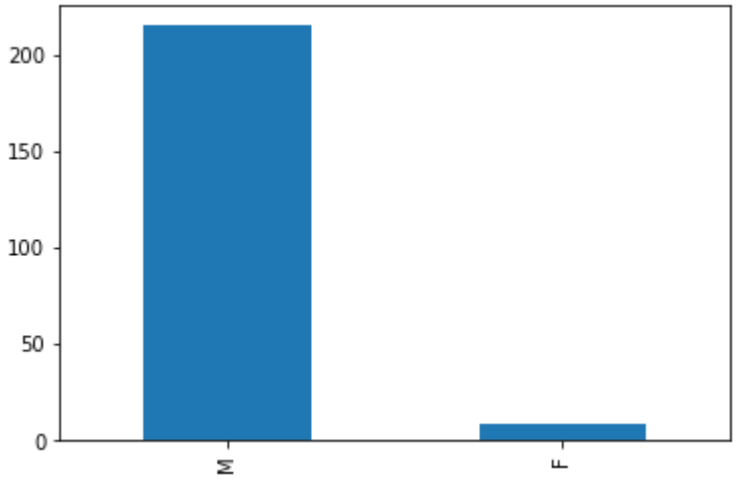
- We want to make a visual plot of the shooting incident
- Let's explore if we can make a plot based on longitude and latitude.
  - HINT: use `plot.scatter(x='longitude', y='latitude')`
- Then the analysis phase will focus on how to make useful insights with the feature selections.

In [55]:
```
subjects_incidents['gender'].value_counts()
```

Out[55]:
```
M    215
F      8
Name: gender, dtype: int64
```

In [49]:
```
subjects_incidents['gender'].value_counts().plot.bar()
```

Out[49]:
```
<AxesSubplot:>
```

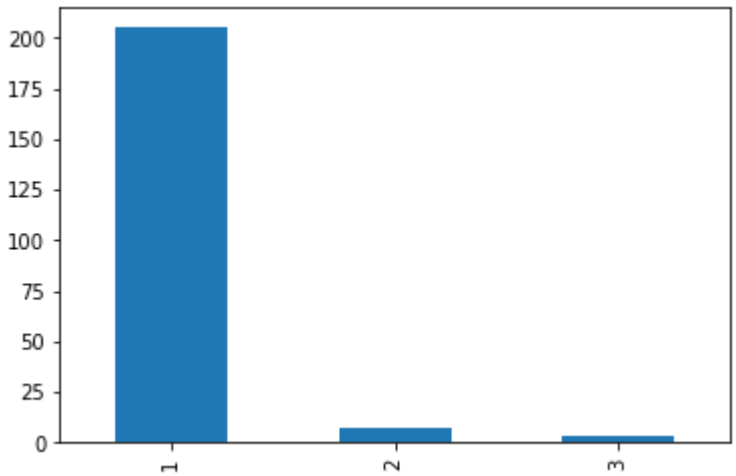


In [59]:
```
subjects_incidents['subject_count'][subjects_incidents['gender']=='M'].value_counts()
```

Out[59]:
```
1    205
2      7
3      3
Name: subject_count, dtype: int64
```

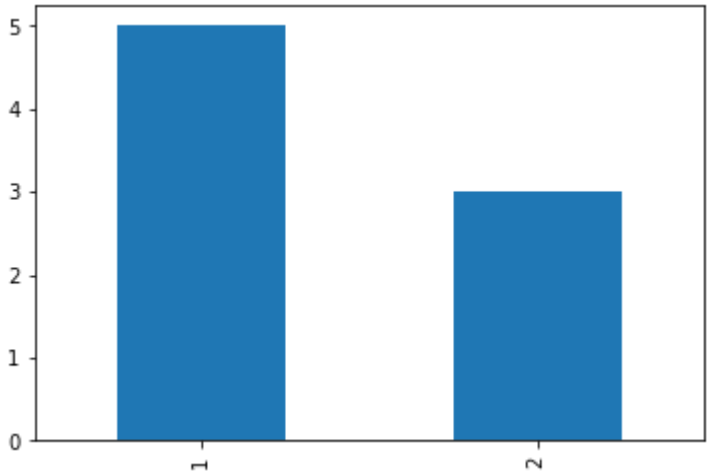
In [53]:
```
subjects_incidents['subject_count'][subjects_incidents['gender']=='M'].value_counts().
```

Out[53]:
```
<AxesSubplot:>
```



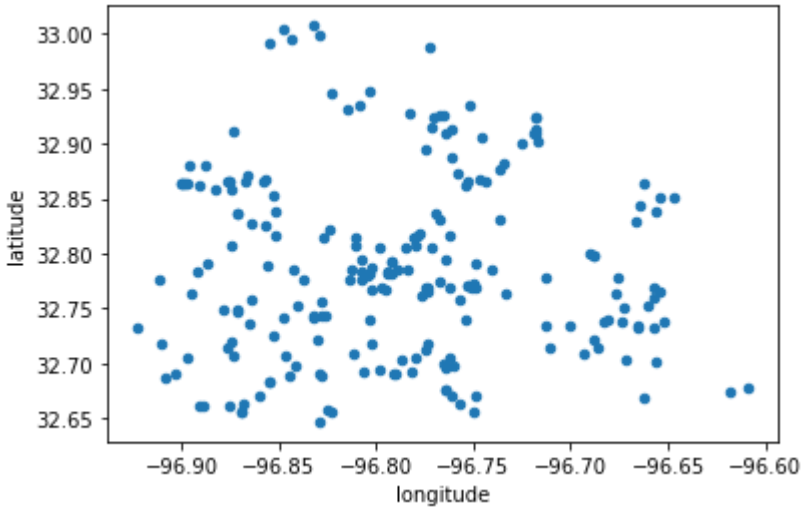
In [54]:
```
subjects_incidents['subject_count'][subjects_incidents['gender']=='F'].value_counts().
```

Out[54]:
```
<AxesSubplot:>
```



In [28]:
```
subjects_incidents.plot.scatter(x='longitude', y='latitude')
```

Out[28]:
```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```



Step 3: Analyze

- Feature selection
- Model selection
- Analyze data

Step 3.a: Feature selection

- Here we will continue with the features selected feel free to explore other features.
- Create a `dataset` with the features: `race`, `subject_statuses`, `latitude`, `longitude`
  - HINT: Select the features of the DataFrame `subject_incidents` by filtering with a list of the columns.
- To make further processing easier apply `dropna()` to remove missing data.

In [29]:
```
dataset = subjects_incidents[['race', 'subject_statuses', 'latitude', 'longitude']].dr
```

In [30]:
```
len(dataset)
```

Out[30]:
```
214
```

Step 3.b: How to visualize features

- We want to visualize the two features: `race` and `subject_statuses`
- A way to visualize data is by color and size.
- Idea:
  - Map the race features to colors
  - Map the subject_statuses to a size

Step 3.b.1: Convert column

- The `race` column has the following categories: B, W, A, L
- We can map that to color values (docs)
- A simple way to map columns is by using `apply` on a lambda-function.
  - Create a dict with the mapping:

    ```
    mapping = {'B': 'blue', 'W': 'yellow', 'A': 'red', 'L': 'cyan'}
    ```

  - Do the mapping with `apply` and `lambda` as follows

    ```
    dataset['race'] = dataset['race'].apply(lambda x: mapping[x])
    ```

In [31]:
```
mapping = {'B': 'blue', 'W': 'yellow', 'A': 'red', 'L': 'cyan'}
dataset['race'] = dataset['race'].apply(lambda x: mapping[x])
```

In [32]:
```
dataset.head()
```

Out[32]:

	race	subject_statuses	latitude	longitude
9	cyan	Deceased	32.68642	-96.908674
10	blue	Deceased	32.86400	-96.898998
11	yellow	Shoot and Miss	32.81482	-96.826787
12	blue	Injured	32.77540	-96.767489
13	blue	Shoot and Miss	32.74417	-96.828470

Step 3.b.2: Convert column

- The `subject_statuses` has the following categories: `'1 Deceased 1 Injured', '2 Injured', 'Deceased', 'Deceased Injured', 'Injured', 'Other', 'Shoot and Miss'`
- The main categories are: `'Deceased', 'Injured', 'Shoot and Miss'`
- A simple way is quite similar to last step.
  - Create a mapping of the main categories:

    ```
    mapping = {'Deceased': 1000, 'Injured': 500, 'Shoot and Miss': 250}
    ```

- Do the mapping with `apply` and `lambda` as follows

```
dataset['subject_statuses'] =
dataset['subject_statuses'].apply(lambda x: mapping.get(x, 100))
```

In [33]:
```
mapping = {'Deceased': 1000, 'Injured': 500, 'Shoot and Miss': 250}
dataset['subject_statuses'] = dataset['subject_statuses'].apply(lambda x: mapping.get(
```

Step 3.c: Visualize the data

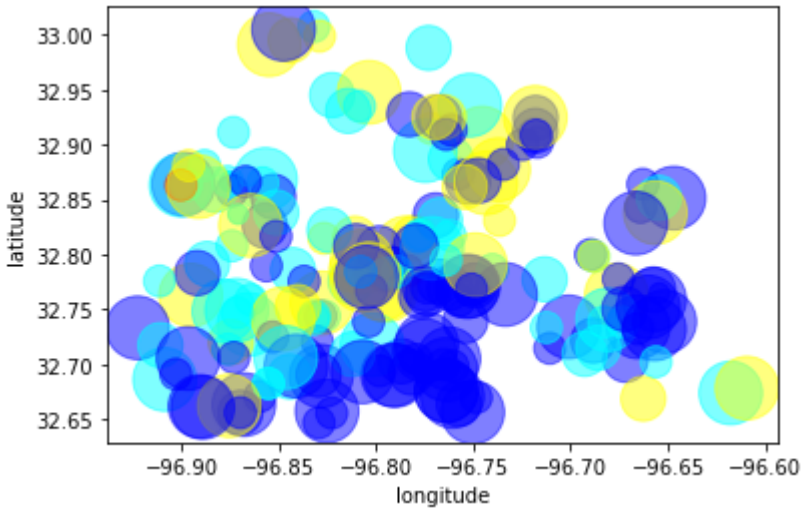
- Now we can visualize the data
- This can be done with a scatter plot

```
dataset.plot.scatter(x='longitude', y='latitude', s='subject_statuses',
c='race', alpha=.5)
```

- Where `s=` is the size feature and `c=` is the color feature, `figsize=` sets the size of the figure, `alpha=` sets the transparency of the dots.

In [34]:
```
dataset.plot.scatter(x='longitude', y='latitude', s='subject_statuses', c='race', alph
```

Out[34]:
```
<AxesSubplot:xlabel='longitude', ylabel='latitude'>
```



Step 4.a: Present a chart

- The goal here is to present your message
- Visualize one chart
- Add a headline (title) to give the audience a message

In [9]:
```
pip install folium
```

```
Requirement already satisfied: folium in c:\users\kingofkings\anaconda3\lib\site-packages (0.12.1.post1)
Requirement already satisfied: numpy in c:\users\kingofkings\anaconda3\lib\site-packages (from folium) (1.21.5)
Requirement already satisfied: requests in c:\users\kingofkings\anaconda3\lib\site-packages (from folium) (2.27.1)
Requirement already satisfied: jinja2>=2.9 in c:\users\kingofkings\anaconda3\lib\site-packages (from folium) (2.11.3)
Requirement already satisfied: branca>=0.3.0 in c:\users\kingofkings\anaconda3\lib\site-packages (from folium) (0.5.0)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\kingofkings\anaconda3\lib\site-packages (from jinja2>=2.9->folium) (2.0.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\kingofkings\anaconda3\lib\site-packages (from requests->folium) (1.26.9)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\kingofkings\anaconda3\lib\site-packages (from requests->folium) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\kingofkings\anaconda3\lib\site-packages (from requests->folium) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\kingofkings\anaconda3\lib\site-packages (from requests->folium) (3.3)
Note: you may need to restart the kernel to use updated packages.
```

In [40]:
```
import folium
```

In [41]:
```
m = folium.Map(location=[32.8, -96.8])
```

In [42]:
```
m
```

Out[42]:

In [43]:
```
for _, row in dataset.iterrows():

    folium.CircleMarker(
        location=[row['latitude'], row['longitude']],
        radius=row['subject_statuses']//100,
        color=row['race'],
        fill=True,
```

```
        fill_color=row['race']
    ).add_to(m)
```

In [44]:
```
m
```

Out[44]: