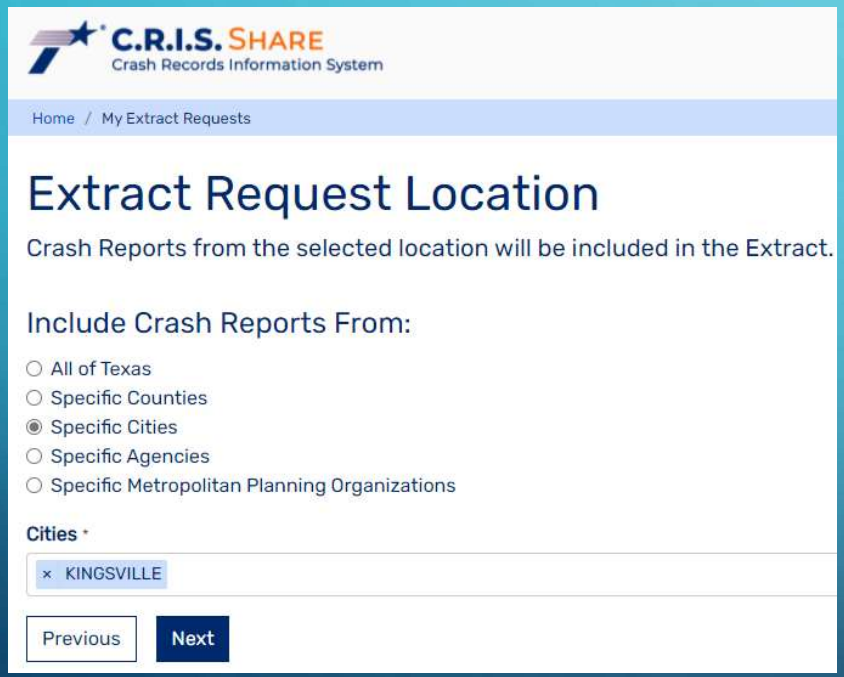
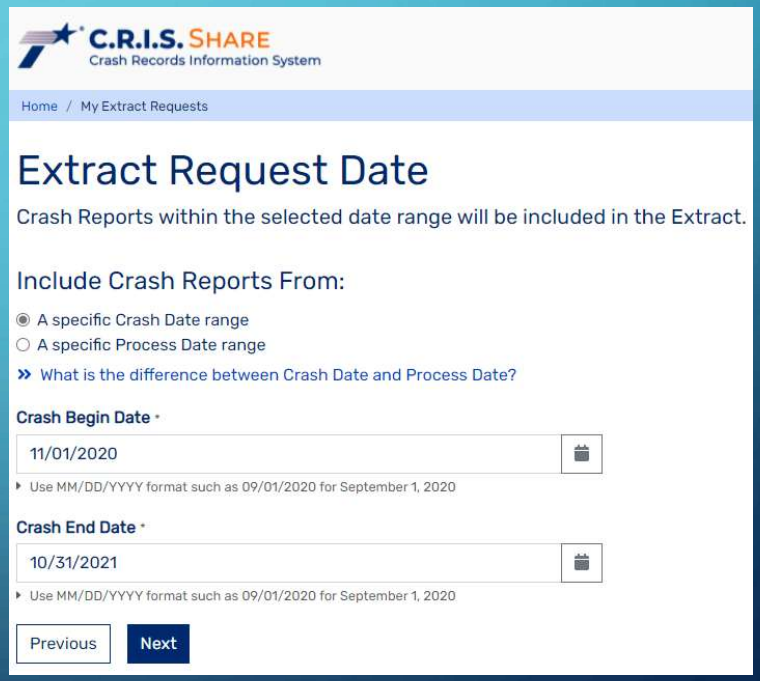
DATASET ANALYSIS OF ACCIDENTS

MIGUEL LOPEZ

HOW TO GET THE DATA?

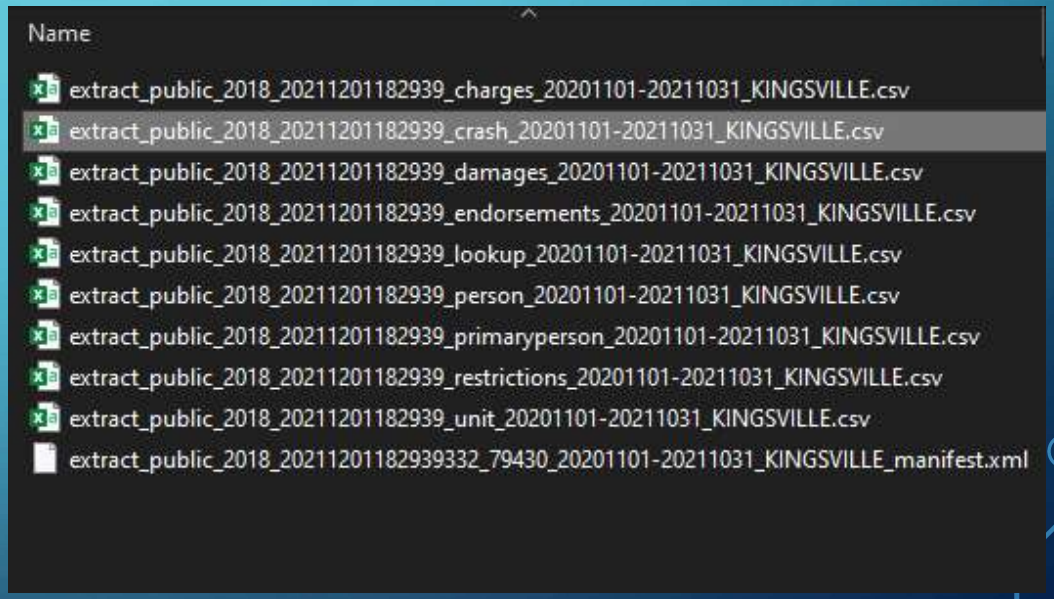
HOW TO GET THE DATA?





WE HAVE SOME DATA, NOW WHAT?

- Only used ‘crash’ and ‘primaryperson’ files
- ‘damages’, ‘endorsements’, ‘restrictions’, ‘unit’ low priority.
- ‘charges’, ‘person’ could be useful
- ‘lookup’ used to decipher codes



CLEAN THE DATA!

200	S	14TH	ST	1000	E	KING	AVE
2600	S	NOT REPORTED		2600	E	NOT REPORTED	
1900	E	KING	AVE	200	S	US 77	HWY
200	E	NOT REPC	ST	2600	S	6TH	ST
2600	S	BRAHMA	BLVD	1000	E	NOT REPC	RD
600	S	19TH	ST	1500	E	FORDYCE	AVE
1331	W	SANTA GE	AVE	1300	W	KING	AVE
1800	S	6TH	ST	100	E	AILSIE	AVE
208	S	14TH	ST	1000	E	KING	AVE
1700	E	GENERAL I	BLVD	2600	S	77	HWY
2900		BRAHMA	BLVD	900	E	NOT REPORTED	
1000	S	14TH	ST	900	E	RAGLAND	ST
1503		US 77	HWY	1900	E	CORRAL	AVE
200	E	FAIRVIEW	DR	200		FAIRVIEW	DR
400	E	KLEBERG	AVE	100	N	9TH	ST
1500	E	NOT REPC	RD	1400	N	17TH	ST
500	N	ARMSTRO	ST	700	W	RICHARD	AVE
1300	W	CIRCLE	DR	1000	E	MILLER	AVE
200		14TH	ST	900		KING	AVE
2600	S	BRAHMA	BLVD	1000	E	NOT REPORTED	
1500	N	77	HWY	2500	E	CORRAL	AVE
1000	N	ARMSTRO	ST	800	W	B	AVE
200	N	NOT REPORTED		2300	E	NOT REPORTED	
900	E	KING	AVE	200	S	14TH	ST
1600	S	HWY 77	HWY	2200		SEN CARL	BLVD
1100	E	GENERAL I	BLVD	2600	S	BRAHMA	BLVD

27.51212	-97.8478
27.49896	-97.8597
27.4915	-97.8467
27.49	-97.8558
27.50775	-97.856
27.52381	-97.8469

- Missing values!
- Coordinate retrieval function (broken)
- Program left to drop rows without coordinates
- Saved used data to a new csv

WHAT DOES OUR REFORMATTED CSV LOOK LIKE?

A	B	C	D	E	F	G	H
Date	Time	Day	Age	Latitude	Longitude	Street1	Street2
11/1/2020	7:22 PM	SUN	69	27.50248909	-97.85592651	1600.0 BRAHMA BLVD	900.0 LOOP 428
11/1/2020	3:58 PM	SUN	22	27.51570606	-97.85672817	1000.0 E KING AVE	200.0 S 14TH ST
11/1/2020	2:13 AM	SUN	57	27.491382	-97.855829	2200.0 S BRAHMA BLVD	1000.0 E GENERAL CAVAZOS BLVD BLVD
11/1/2020	10:29 AM	SUN	19	27.50679948	-97.85390537	1100.0 E CAESAR AVE	1000.0 S 15TH ST
11/2/2020	2:49 PM	MON	26	27.50657142	-97.87369616	2700.0 FRANKLIN ADAMS ST	400.0 W BIRCHWOOD DR
11/2/2020	8:42 AM	MON	33	27.49629438	-97.85587815	2200.0 S BRAHMA BLVD	1000.0 E AILSIE AVE
11/3/2020	8:44 PM	TUE	18	27.501774	-97.847497	1601.0 S NOT REPORTED	1700.0 SENATOR CARLOS TRUAN BLVD
11/6/2020	4:00 AM	FRI	54	27.52495937	-97.86933084	900.0 N 5TH ST	130.0 W NETTIE AVE
11/5/2020	11:07 PM	THU	23	27.52977551	-97.85616324	1300.0 N 14TH ST	900.0 E MESQUITE AVE
11/6/2020	10:42 AM	FRI	66	27.52259534	-97.87643254	600.0 W SANTA GERTRUDIS ST	700.0 N WELLS ST
11/3/2020	1:33 PM	TUE	58	27.530548	-97.86348571	500.0 E CORRAL ST	1400.0 N 6TH RD
11/8/2020	3:33 PM	SUN	35	27.491355	-97.854716	1133.0 E GENERAL CAVAZOS BLVD	1100.0 E GENERAL CAVAZOS BLVD
11/9/2020	10:43 PM	MON	18	27.50419567	-97.84062781	1300.0 S US 77 HWY	2200.0 E CAESER AVE
11/7/2020	8:28 PM	SAT	21	27.53037244	-97.87793829	700.0 W CORRAL AVE	1400.0 N ARMSTRONG AVE
11/12/2020	1:26 PM	THU	30	27.506677	-97.86591893	300.0 E CAESAR ST	1000.0 S 7TH ST
11/13/2020	10:08 PM	FRI	64	27.506797	-97.855995	1013.0 S 14TH ST	900.0 E CAESAR AVE
11/15/2020	1:19 PM	SUN	37	27.49137688	-97.8557663	1000.0 E GENERAL CAVAZOS BLVD	2700.0 S BRAHMA BLVD
11/18/2020	9:49 AM	WED	85	27.49137688	-97.8557663	2700.0 S BRAHMA BLVD	900.0 E GENERAL CAVAZOS BLVD
11/3/2020	5:05 PM	TUE	27	27.550249	-97.878438	3421.0 N 1355 ST	100.0 E 1355 RD
11/18/2020	3:24 PM	WED	78	27.5152303	-97.85610086	200.0 S 14TH ST	1000.0 E KING AVE

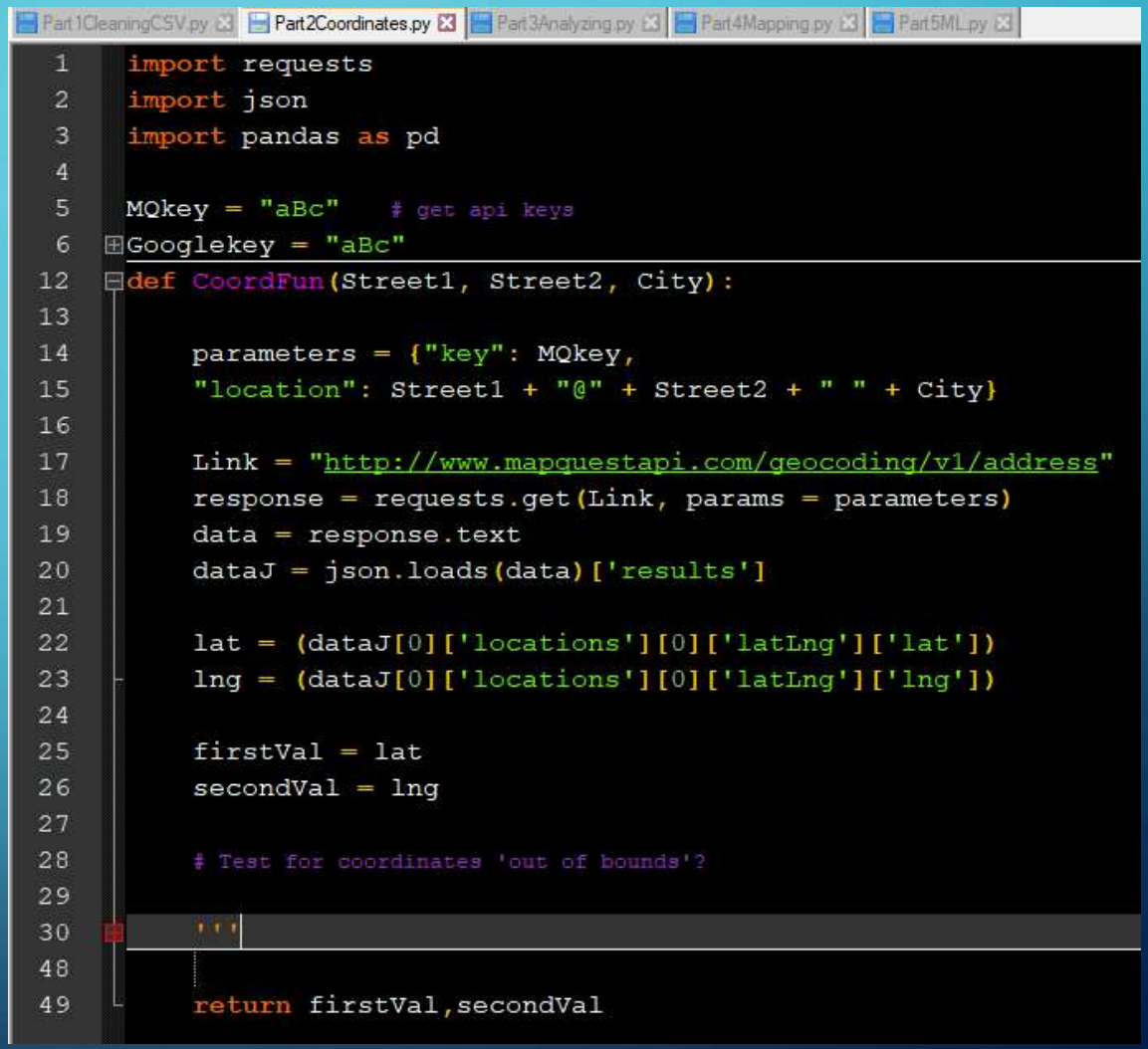
Light on attributes maybe?

CODE

```
import pandas as pd
from Part2Coordinates import CoordFun


# Change 1/6 vvv
City = "Kingsville, Texas"
# Change 1/6 ^^^
# Change 2/6 vvv
FileCrash = "KINGSVILLE/extract_public_2018_20211201182939332_79430_20201101-20211031_KINGSVILLE/extract_pub
# Change 2/6 ^^^
dfCrash = pd.read_csv(FileCrash)

dfCrash.Crash_ID.drop_duplicates(inplace=True)  # should drop duplicate CrashIDs, has not happened in testing.
# Change 3/6 vvv
FileAge = "KINGSVILLE/extract_public_2018_20211201182939332_79430_20201101-20211031_KINGSVILLE/extract_publi
# Change 3/6 ^^^
field = ['Prsn_Age']
dfAge = pd.read_csv(FileAge, usecols=field)
dfAge.fillna(dfAge.mean().round(0), inplace=True) # inplace here means no "dfAge ="; round instead of default truncate.
dfAge = dfAge.astype(int)

num = dfCrash['Latitude'].isnull().sum()
DataLength = len(dfCrash.index)
print('The percent of data with missing lat/long values is: ', (num/DataLength*100).round(2),'%')
# deleting NaN lat/long values.
# Change 4/6 vvv
dfCrash = dfCrash[dfCrash['Latitude'].notna()] # comment out depending on using part2 function (broken)
# Change 4/6 ^^^
num = dfCrash['Latitude'].isnull().sum()
print('After removing rows with missing lat/long - percent is: ', (num/DataLength*100).round(2),'%')
dfCrash = dfCrash.fillna('')
dfCrash['Street1'] = dfCrash['Rpt_Block_Num'].astype(str)+' '+dfCrash['Rpt_Street_Pfx']+' '+dfCrash['Rpt_Str
dfCrash['Street2'] = dfCrash['Rpt_Sec_Block_Num'].astype(str)+' '+dfCrash['Rpt_Sec_Street_Pfx']+' '+dfCrash[
# reference elements: dfCrash.Street1[0]

for i in range(0,DataLength):
    if (dfCrash.Latitude[i] == ''):
        dfCrash.iloc[i, dfCrash.columns.get_loc('Latitude')],dfCrash.iloc[i, dfCrash.columns.get_loc('Longitu
    # could probably fix but comment out if 'change 4/6' also commented out (ie. not deleting NaN lat/long val)
    # Change 5/6 vvv
    else:
        break
```

CODE

- Usually works..
- Main error was same coordinate was given no matter input
- Sometimes gave coordinates far away



WORKAROUND FOR MISSING COORDINATES

- Drop rows with missing coordinates.
- Estimate 15-20% of data could be missing the 'official' coordinates

Kingsville Data

```
/usr/bin/python3 "Part1CleaningCSV.py" (in directory: /home/ava/
The percent of data with missing lat/long values is:  15.77 %
After removing rows with missing lat/long - percent is:  0.0 %
Compilation finished successfully.
```

Laredo Data

```
/usr/bin/python3 "Part1CleaningCSV.py" (in directory: /home/ava/
The percent of data with missing lat/long values is:  18.33 %
After removing rows with missing lat/long - percent is:  0.0 %
Compilation finished successfully.
```

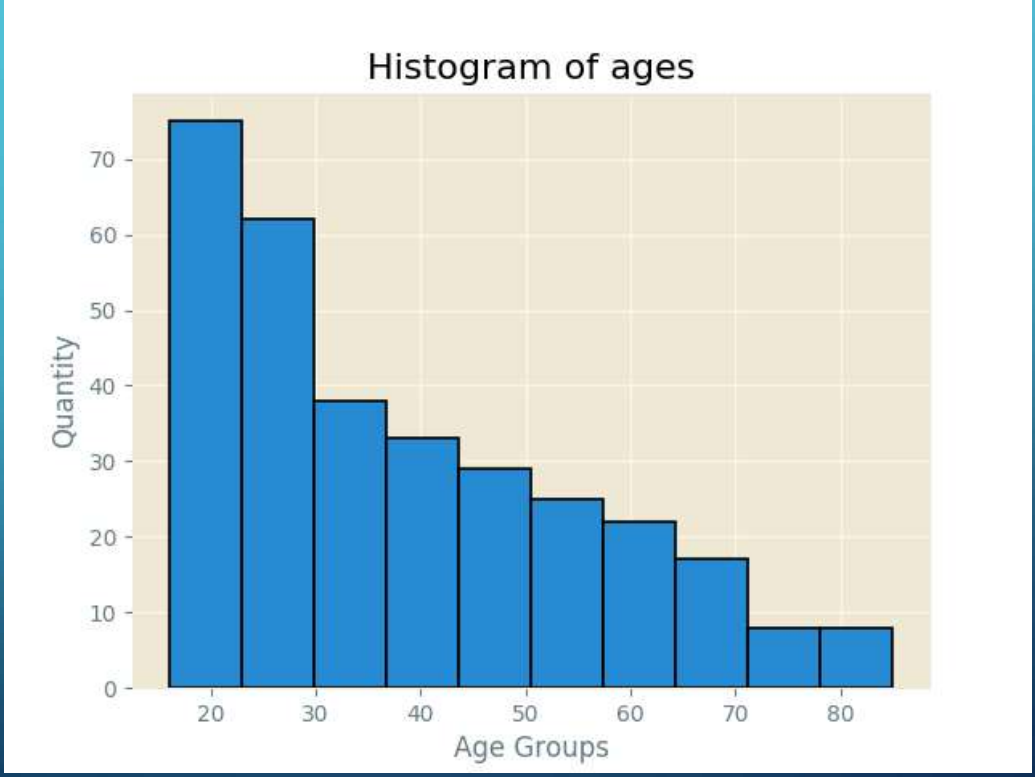
PART 3 – ANALYZING!

Part1CleaningCSV.py | Part2Coordinates.py | Part3Analyzing.py | Part4Mapping.py | Part5ML.py

```python
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
import calendar
plt.style.use('Solarize_Light2')
myFile = "output.csv"
df = pd.read_csv(myFile)
# MUST INCLUDE RANGE OF MONTHS BY DIGIT. (ie. 5 = May)
# Change 6/6 vvv
monthRange = [11,12,1,2,3,4,5,6,7,8,9,10]
# Change 6/6 ^^^
# Age analysis
avgAge = df.Age.mean().round(1) # slightly differs from the dfAge.mean() in part 1?
maxAge = df.Age.max()
minAge = df.Age.min()
print('The min age of the driver is: ',minAge,'years old')
print('The max age of the driver is: ',maxAge,'years old')
print('The average age of the driver is: ',avgAge,'years old')
plt.figure()
df.Age.hist(edgecolor='black', linewidth=1.2)
plt.title('Histogram of ages')
plt.xlabel('Age Groups')
plt.ylabel('Quantity')
plt.draw()
```

Status	/usr/bin/python3 "Part3Analyzing.py" (in directory: /home
Compiler	The min age of the driver is: 16 years old
Messages	The max age of the driver is: 85 years old
Scribble	The average age of the driver is: 38.2 years old

PART 3 – ANALYZING!

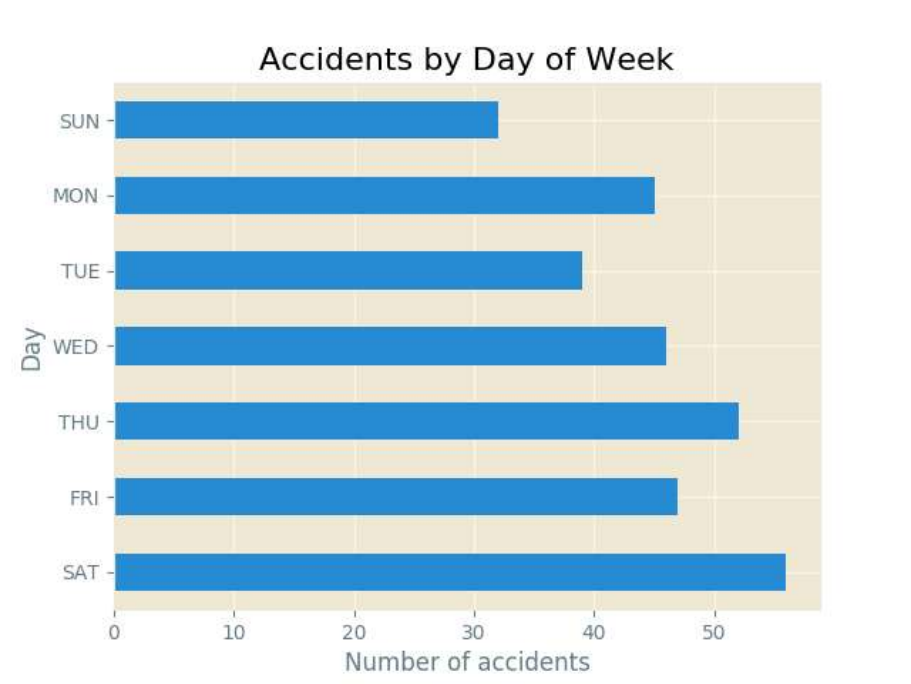


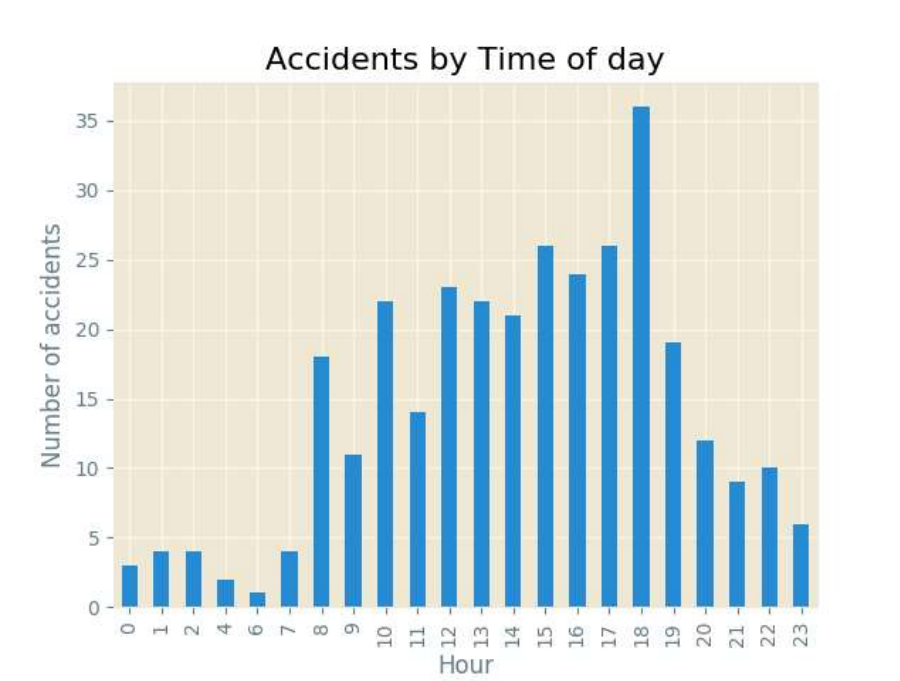
PART 3 – ANALYZING!

```
27  # Average time
28  x = pd.to_datetime(df.Time).values.astype(np.int64)
29  x = x.mean()
30  x = pd.to_datetime(x)
31  x = str(x.time())
32  x = datetime.strptime(x, "%H:%M:%S.%f")
33  avgTime = x.strftime("%I:%M %p")
34  print('The average time of an accident is:',avgTime)
35
36  # Bar graph of accidents by day of week
37  plt.figure()
38  df.Day = pd.Categorical(df.Day, categories=
39      ['SUN','MON','TUE','WED','THU','FRI','SAT'],
40      ordered=True)
41  accidentsbyday = df.Day.groupby(df['Day'],sort=False).count().plot(kind='barh')
42  plt.gca().invert_yaxis()     # going from bar to barh, inverting axis makes sense.
43  plt.title('Accidents by Day of Week')
44  plt.xlabel('Number of accidents')
45  plt.ylabel('Day')
46  plt.draw()
47
48  DateTimeArr = pd.to_datetime(df['Date'] + ' ' + df['Time']) # confusing method, but it works..
49
50  # Bar graph of accidents by hour
51  plt.figure()
52  accidentsbyhour = DateTimeArr.groupby(DateTimeArr.dt.hour).count().plot(kind='bar')
53  plt.title('Accidents by Time of day')
54  plt.xlabel('Hour')
55  plt.ylabel('Number of accidents')
56  plt.draw()
57
58  # Number of accident per month
59  accidentsbymonth = DateTimeArr.groupby(DateTimeArr.dt.month).count()
60  accidentsbymonth.index=[calendar.month_name[x] for x in monthRange] # converts month digit to calendar month
61  print(accidentsbymonth.to_string()) # .to_string() removes 'dtype' on console output.
```

```
The average time of an accident is: 02:48 PM
November     20
December     32
January      22
February     28
March        20
April        24
May          29
June         32
July         31
August       26
September    31
October      22
Compilation finished successfully.
```

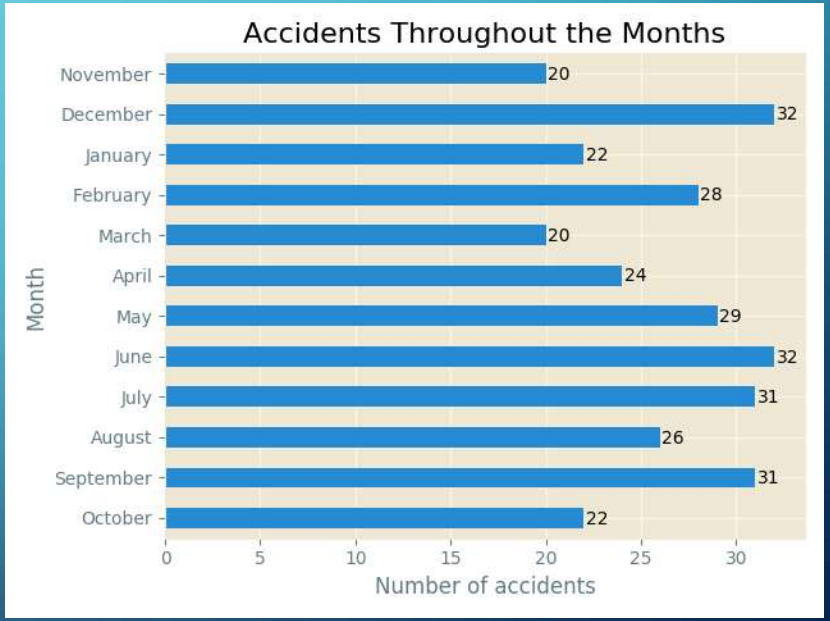
PART 3 – ANALYZING!



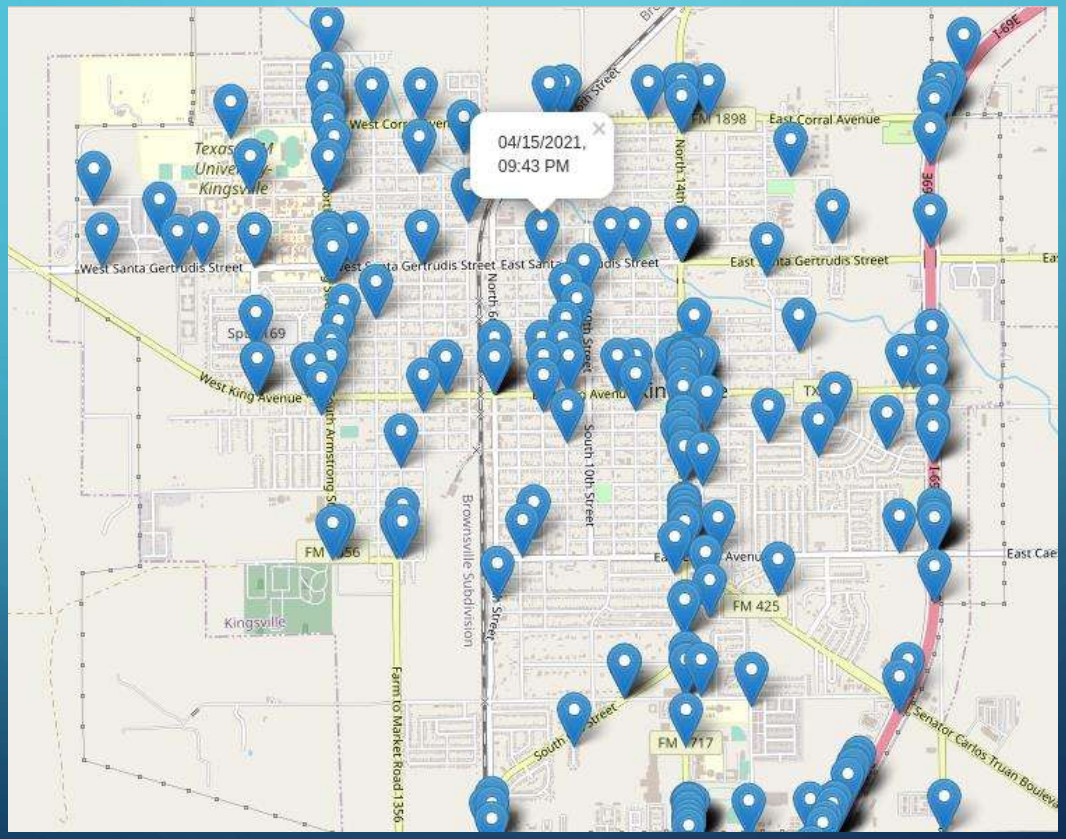


PART 3 – ANALYZING!

```
63  # plot accidents per month
64  plt.figure()
65  accidentsbymonth.plot(kind='barh')
66  plt.gca().invert_yaxis()     # going from bar to barh
67  plt.title('Accidents Throughout the Months')
68  plt.xlabel('Number of accidents')
69  plt.ylabel('Month')
70  plt.tight_layout()
71  for i, v in enumerate(accidentsbymonth):
72      plt.text(v+.1, i+.15, str(v))
73
74  plt.show()
```



PART 4 – MAPPING!

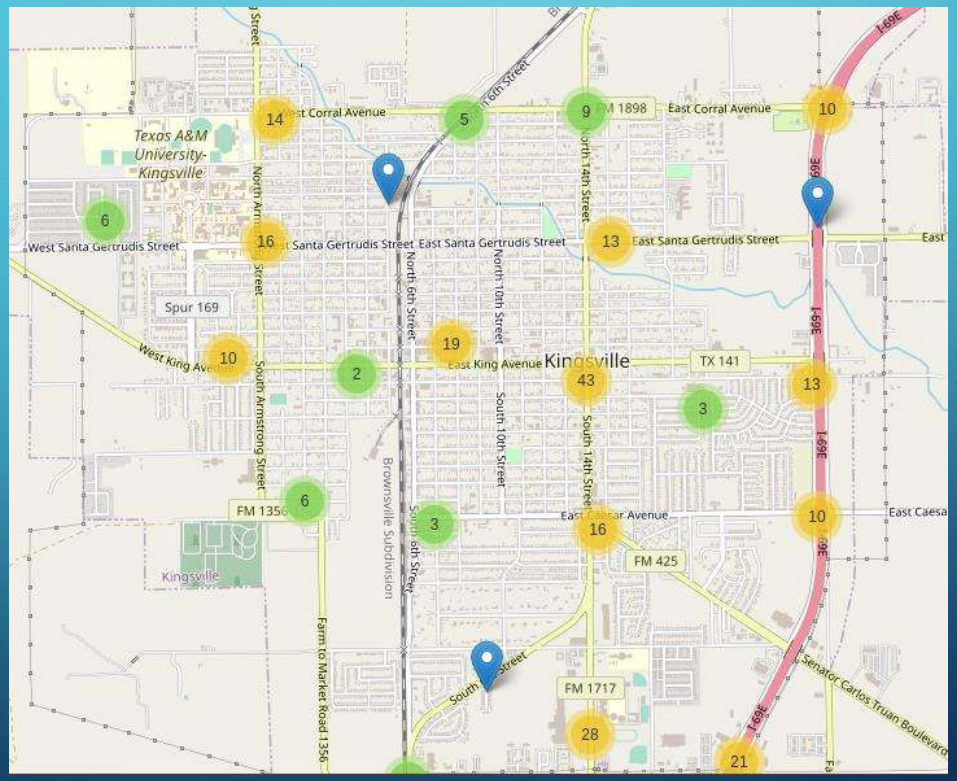


PART 4 – MAPPING!

Part1CleaningCSV.py | Part2Coordinates.py | Part3Analyzing.py | Part4Mapping.py | Part5ML.py

```
1  import folium
2  from folium import plugins
3  from folium.plugins import HeatMap,HeatMapWithTime
4  import webbrowser    # functionality depends on python environment, tested on Ubuntu+Geany.
5  import pandas as pd
6  myFile = "output.csv"
7  df = pd.read_csv(myFile)
8  myLocation = [df.Latitude.mean(), df.Longitude.mean()]
9
10 # Simple map creation with markers.
11 myMap = folium.Map(myLocation, zoom_start=13) # adjust zoom to specific application.
12 dftemp = df['Date'].astype(str)+',\n'+df['Time'].astype(str)
13 for index, row in df.iterrows():
14     folium.Marker([row['Latitude'], row['Longitude']], popup=dftemp[index]).add_to(myMap)
15 myMap.save("myMap.html")
16
```

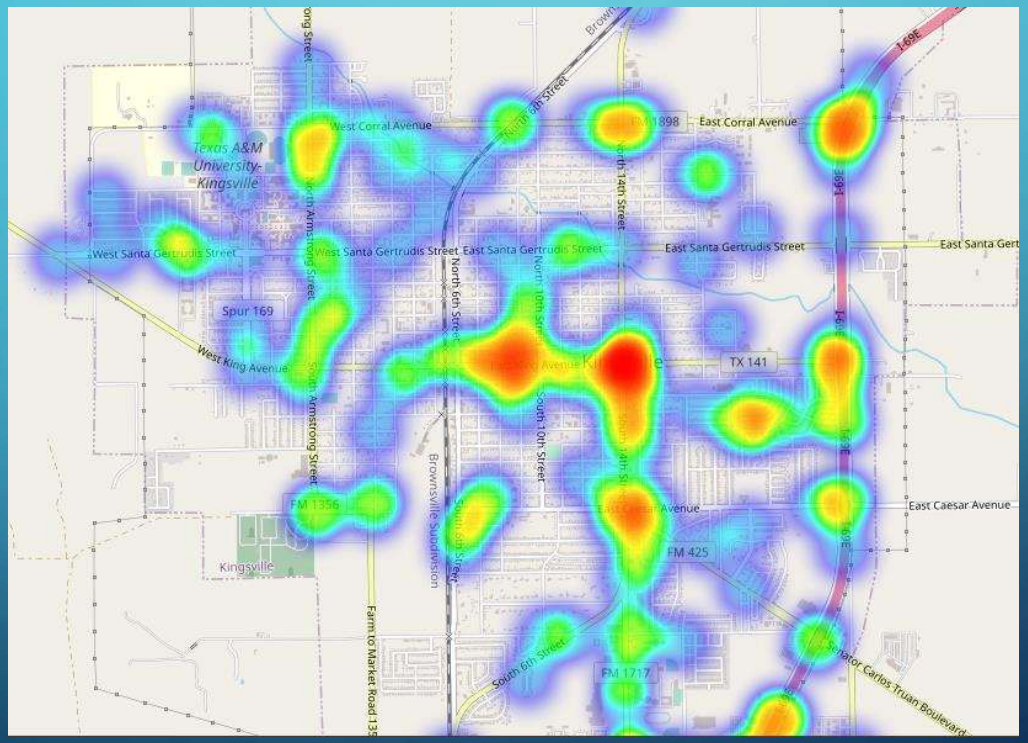
PART 4 – MAPPING!



PART 4 – MAPPING!

```
17  # map creation with automatically clustered groups.
18  myMap = folium.Map(myLocation, zoom_start=13) # clean map.
19  myMap = plugins.MarkerCluster().add_to(myMap)
20  for index, row in df.iterrows():
21      folium.Marker([row['Latitude'], row['Longitude']], popup=dftemp[index]).add_to(myMap)
22
23  myMap.save("myMap2.html")
24
```

PART 4 – MAPPING!



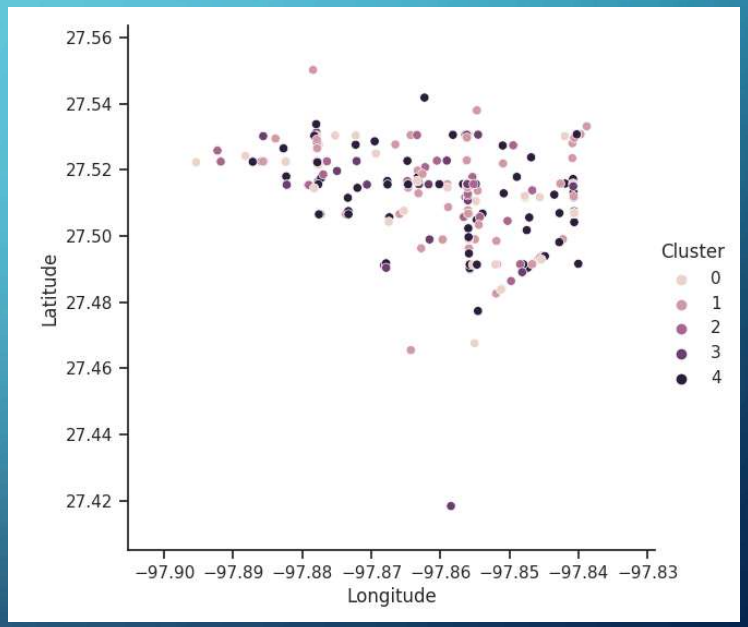
PART 4 – MAPPING!

```
24
25  # heat map
26
27  myMap = folium.Map(myLocation, zoom_start=13) # adjust zoom to specific application.
28  dftemp = df[['Latitude','Longitude']]
29  dftemp = [[row['Latitude'],row['Longitude']] for index, row in dftemp.iterrows()]
30  HeatMap(dftemp).add_to(myMap)
31  myMap.save("myMap3.html")
32
33
34  webbrowser.open("myMap.html")
35  webbrowser.open("myMap2.html")
36  webbrowser.open("myMap3.html")
37
38  # HeatMapWithTime sort of sparse for the data being used for Kingsville.
39  '''
```

PART 5 – MACHINE LEARNING ATTEMPT

Part1CleaningCSV.py | Part2Coordinates.py | Part3Analyzing.py | Part4Mapping.py | Part5ML.py

```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import MiniBatchKMeans,KMeans
import seaborn as sns    # pip3 install seaborn
sns.set_theme(style='ticks')
myFile = "output.csv"
df = pd.read_csv(myFile)

# Mini Batch K Means
df = df.loc[:,['Latitude','Longitude','Age']]
kmeans = MiniBatchKMeans(n_clusters=5, random_state=0, batch_size=6)
df['Cluster'] = kmeans.fit_predict(df)
df['Cluster'] = df['Cluster'].astype('int')
print(df.head())

plt.rc('figure', autolayout=True)
plt.rc('axes')
sns.relplot(x='Longitude',y='Latitude', hue = 'Cluster', data = df, height = 6)
plt.draw()

```



PART 5 – MACHINE LEARNING APPLICATION?

```
21
22  # Traditional K Means
23  # Pseudo Code...
24  # use of this ML could be tested against better data, like combining Dallas and Ft Worth
25  # enough to have thousands of entries, where weather conditions vary more often.
26  # test against weather condition versus area? or time of day/day of week?
27  '''
28  df = df.loc[:,['City','Day']]
29  grouped = pd.get_dummies(df['Day'])
30  grouped['City'] = df['City']
31  grouped = grouped.groupby('City').sum().reset_index()
32  # after grouping, reassign/drop identifier
33  newIdent = grouped[['City']]
34  grouped = grouped.drop('City', axis=1)
35  newIdent['cluster'] = kmeans.labels_
36  kmeans2 = KMeans(n_clusters=4, random_state=0).fit(grouped)
37  '''
38  plt.show()
39
```

FUTURE WORK? - CONCLUSIONS

- Could be redone/rehashed with good methods
- Coordinate retrieval solution could still be worked on
  - Work on reliability of output; alternative API?
- Would benefit from bigger dataset
  - Variations in weather to analyze influences on accidents
  - ML algorithms can handle the big data
  - Confusing studying data based on counts versus x with y data
- Satisfied with useful functionality

THANK YOU!