How safe are roads in DC?

Karen Alkoby | Carol Fogel | Monica Linsangan | Bic Vu

hypothesis

DC roads are not safe for drivers, bicyclists and pedestrians and safety improvements need to be made.

When do most accidents happen, during the day or after dark?

What age group are involved in the most accidents?

Are certain intersections more hazardous than others?

At which hour of the day, day of week, and month of the year do most accidents occur?

DATA SOURCES

- Open Data DC
- Naval Observatory

449,792

Rows of data

DATA CLEANING

CHECK

Invalid report dates were removed when analyzing data by time/date.

IDENTIFY

Ages less than 0 and greater than 100 were determined to be invalid.

DROP

Data containing values that are missing, null or out of range.

MOST DANGEROUS INTERSECTIONS

Can we pinpoint danger?

Available data

Coordinates

Modes of Transportation

Levels of Injury

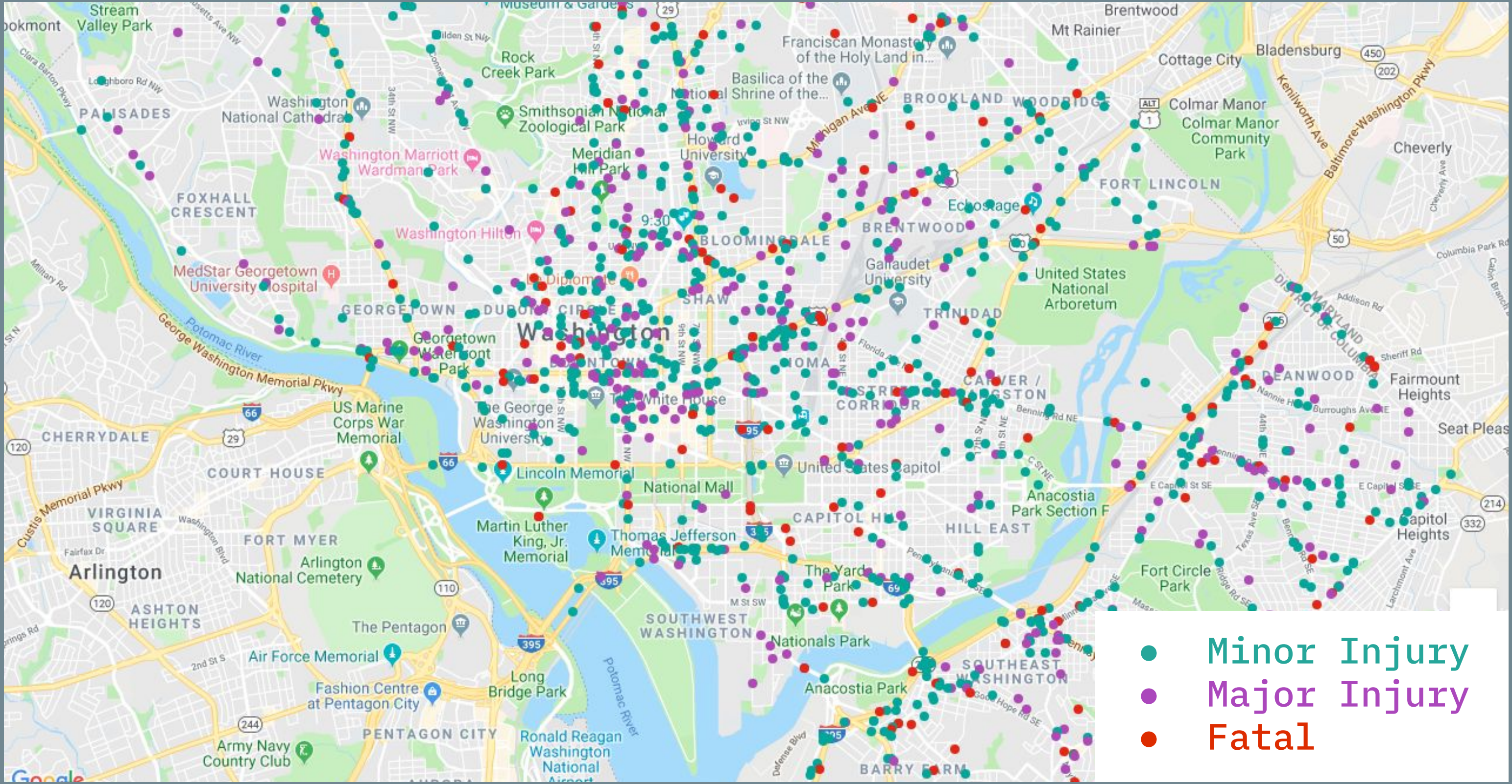
Too much data
to plot.

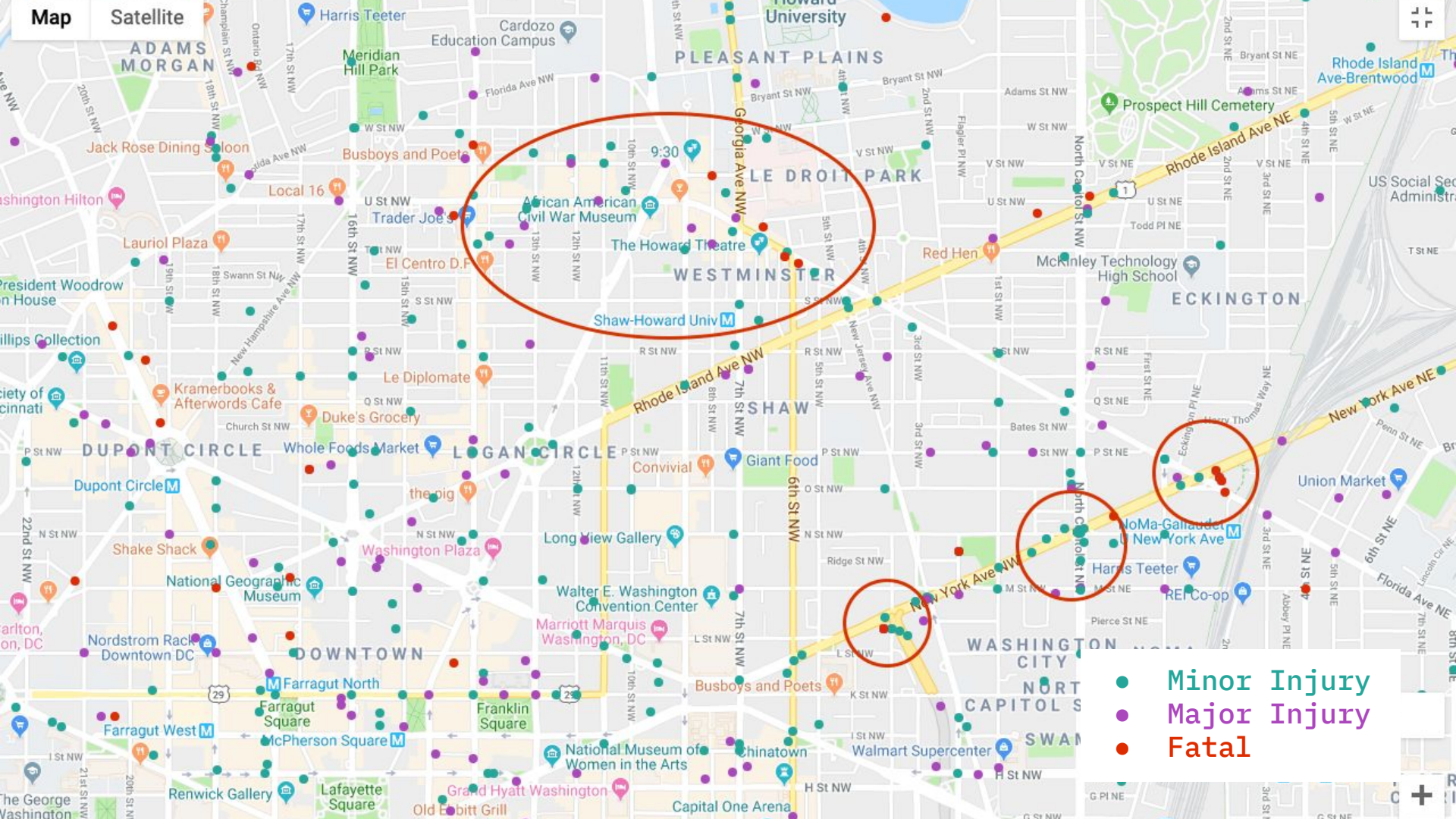
I got it
to work!

CHALLENGES

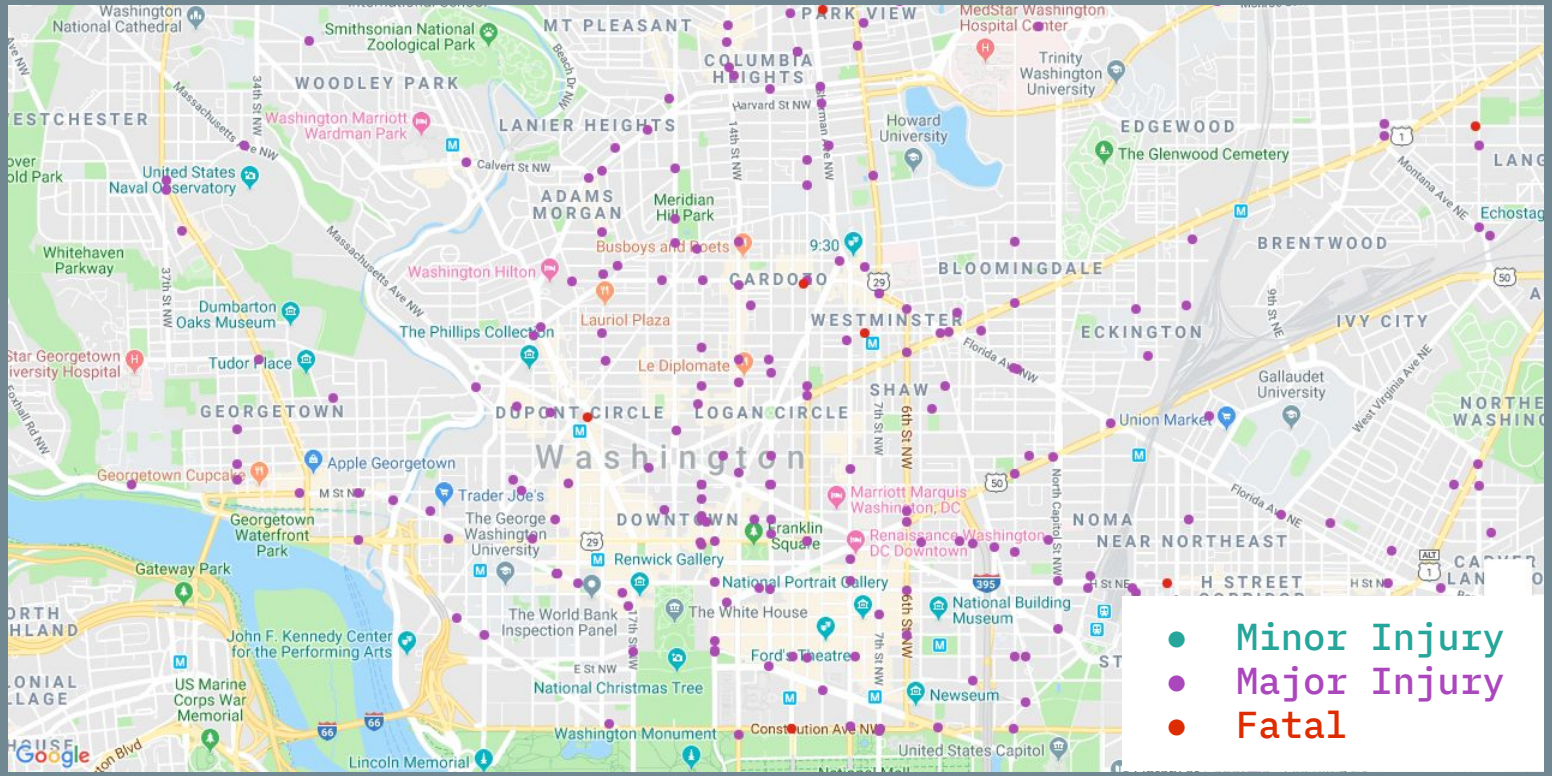
API MIA WTF

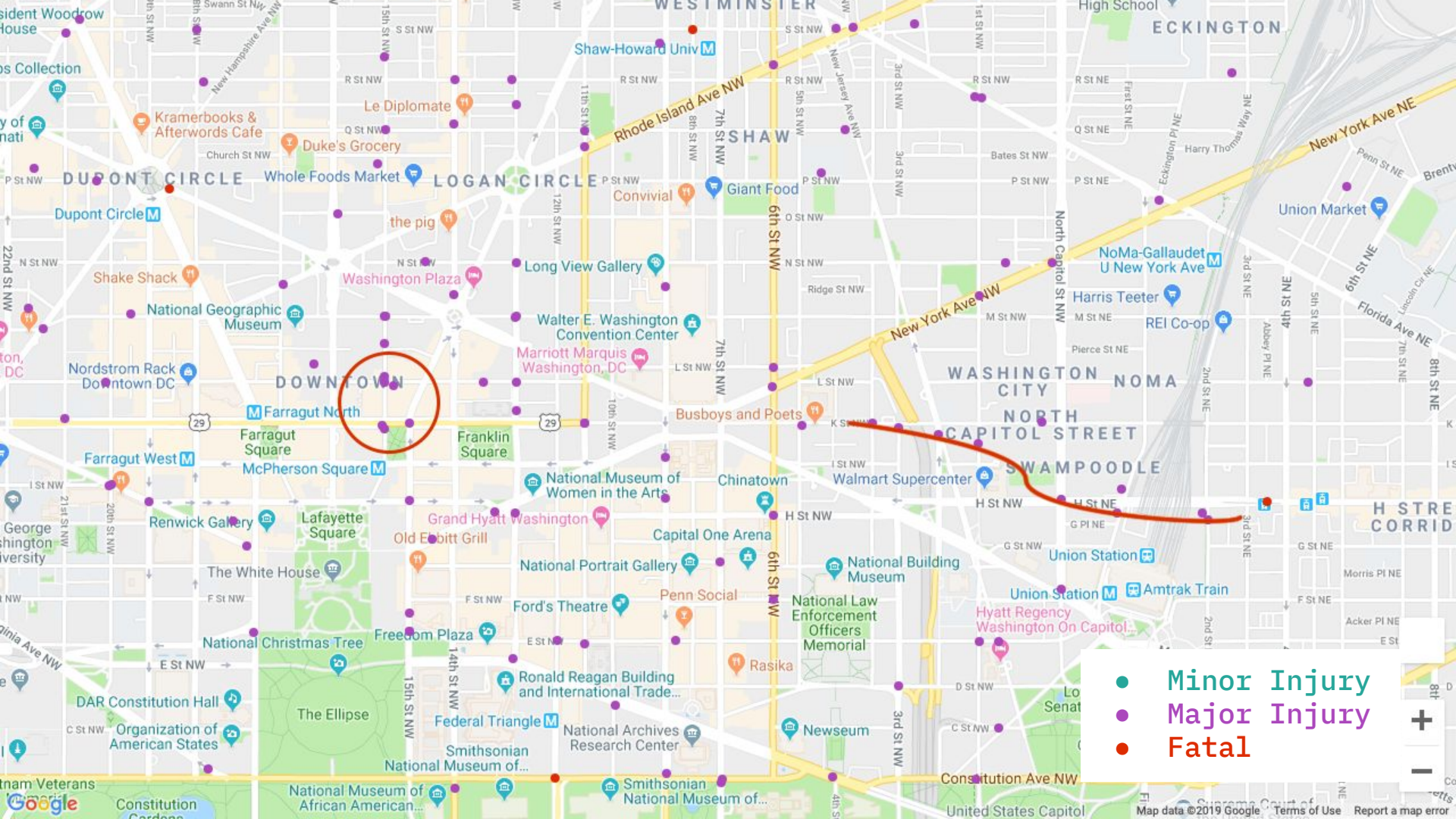
Drivers



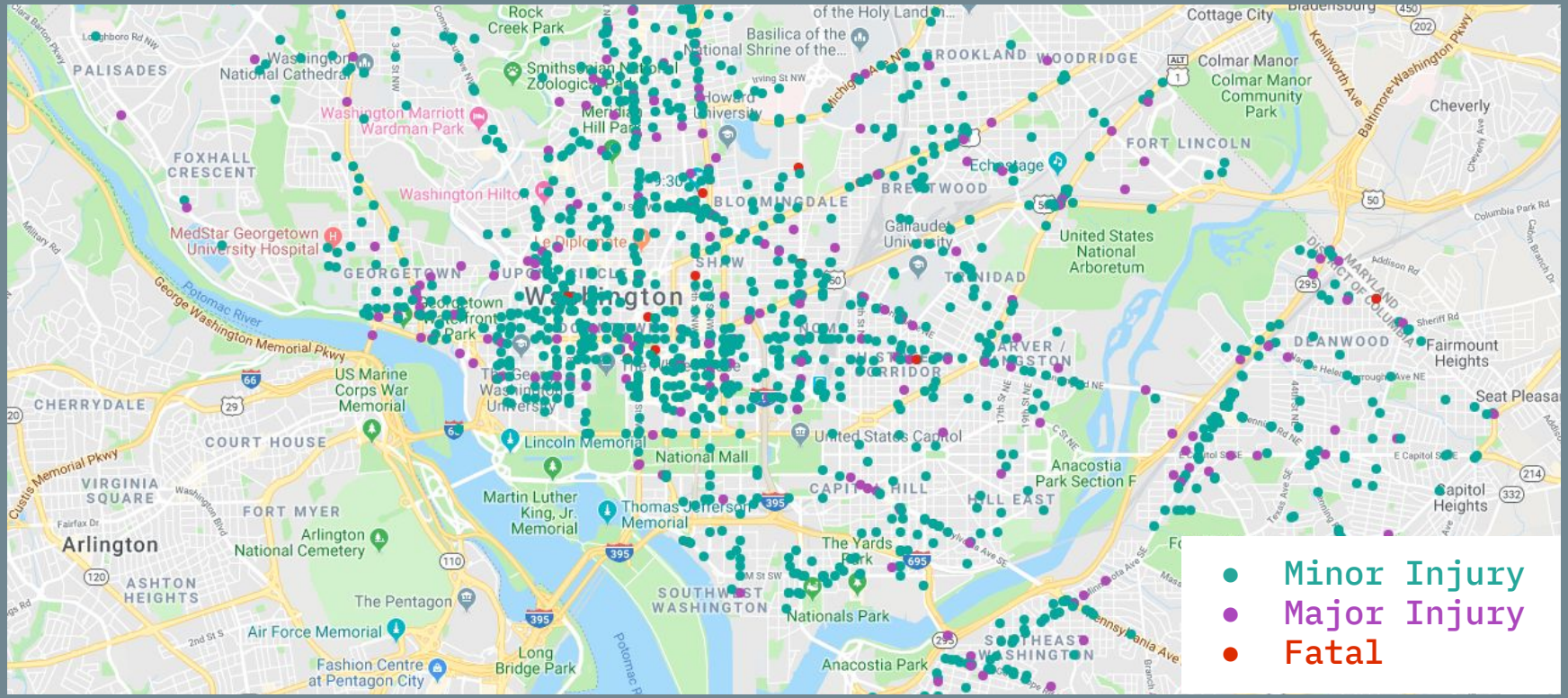

Map
Satellite
ADAMS MORGAN
PLEASANT PLAINS
LE DROIT PARK
WESTMINSTER
ECKINGTON
SHAW
DUPONT CIRCLE
LOGAN CIRCLE
DOWNTOWN
Howard University
Cardozo Education Campus
Meridian Hill Park
Harris Teeter
Jack Rose Dining Saloon
Busboys and Poets
9:30
Local 16
Trader Joe's
African American Civil War Memorial
The Howard Theatre
Lauriol Plaza
El Centro D.F.
Shaw-Howard Univ
Rhode Island Ave NW
Rhode Island Ave NE
Georgia Ave NW
Red Hen
McKinley Technology High School
Prospect Hill Cemetery
Rhode Island Ave-Brentwood
Le Diplomate
Kramerbooks & Afterwords Cafe
Duke's Grocery
Whole Foods Market
Convivial
Giant Food
Dupont Circle
the pig
Long View Gallery
Shake Shack
Washington Plaza
National Geographic Museum
Walter E. Washington Convention Center
Marriott Marquis Washington, DC
Nordstrom Rack Downtown DC
Farragut North
Farragut Square
Farragut West
McPherson Square
Franklin Square
National Museum of Women in the Arts
Chinatown
Walmart Supercenter
Renwick Gallery
Lafayette Square
Grand Hyatt Washington
Old Ebbitt Grill
Capital One Arena
New York Ave NW
New York Ave NE
NoMa-Gallaudet U
New York Ave
Harris Teeter
REI Co-op
Union Market
Busboys and Poets
Minor Injury
Major Injury
Fatal

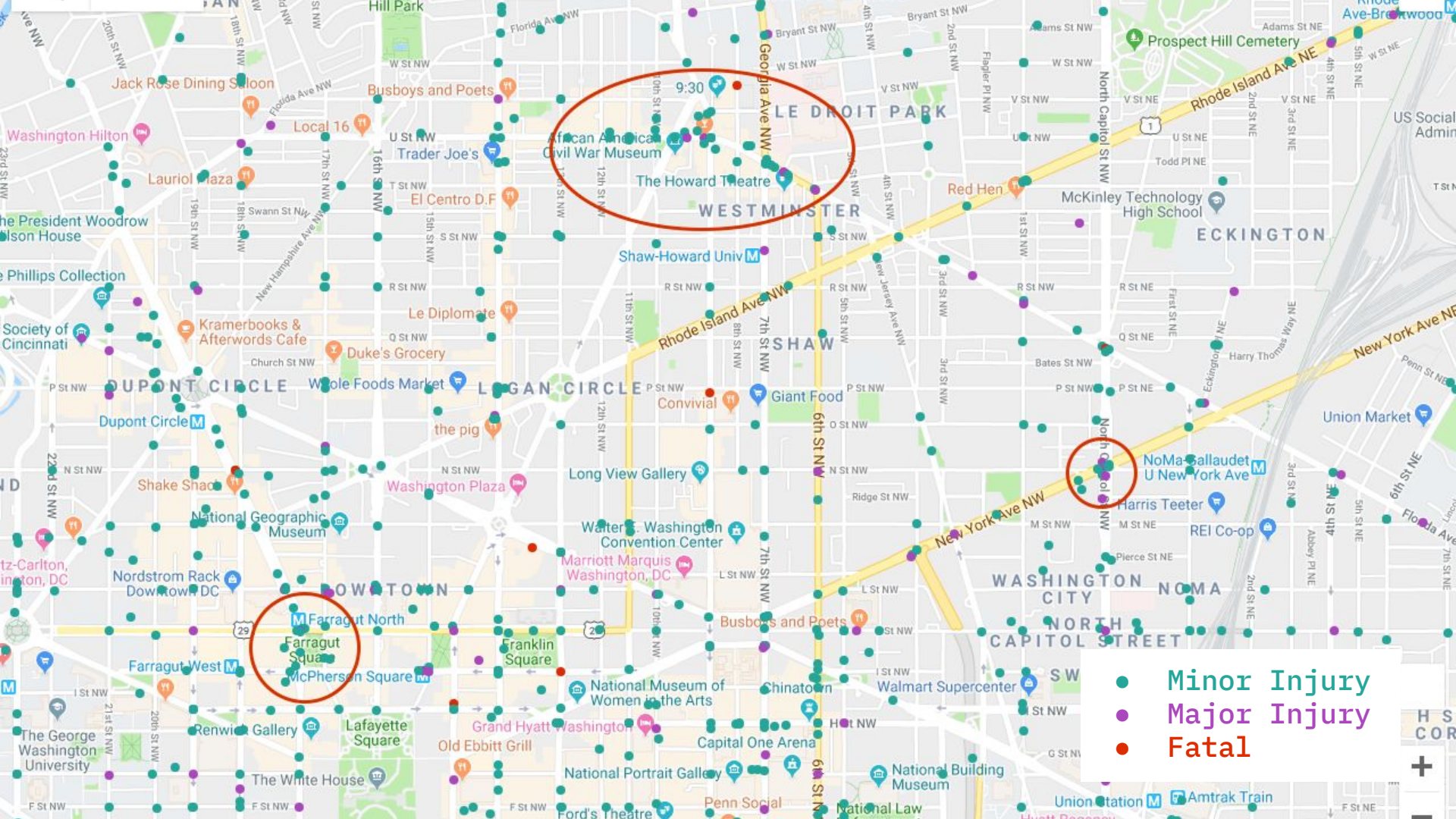
Bicyclists


Washington
National Cathedral
Smithsonian National Zoological Park
MT PLEASANT
PARK VIEW
MedStar Washington Hospital Center
COLUMBIA HEIGHTS
Trinity Washington University
WOODLEY PARK
Washington Marriott Wardman Park
LANIER HEIGHTS
Howard University
EDGEWOOD
The Glenwood Cemetery
United States Naval Observatory
ADAMS MORGAN
Meridian Hill Park
Busboys and Poets
9:30
Washington Hilton
CARDOZO
BLOOMINGDALE
BRENTWOOD
Whitehaven Parkway
Dumbarton Oaks Museum
Lauriol Plaza
WESTMINSTER
ECKINGTON
IVY CITY
The Phillips Collection
Tudor Place
Le Diplomate
SHAW
Gallaudet University
GEORGETOWN
DUPONT CIRCLE
LOGAN CIRCLE
Union Market
Washington
Apple Georgetown
Georgetown Cupcake
Trader Joe's
Marriott Marquis Washington, DC
Georgetown Waterfront Park
The George Washington University
DOWNTOWN
Franklin Square
Renaissance Washington DC Downtown
NOMA
NEAR NORTHEAST
Gateway Park
Renwick Gallery
National Portrait Gallery
H STREET
The World Bank Inspection Panel
The White House
National Building Museum
John F. Kennedy Center for the Performing Arts
Ford's Theatre
US Marine Corps War Memorial
National Christmas Tree
Newseum
Washington Monument
Constitution Ave NW
United States Capitol
Lincoln Memorial
Google
Minor Injury
Major Injury
Fatal


Minor Injury
Major Injury
Fatal
DUPONT CIRCLE
LOGAN CIRCLE
SHAW
DOWNTOWN
WASHINGTON CITY
NOMA
NORTH CAPITOL STREET
SWAMPOODLE
ECKINGTON
WESTMINSTER
Rhode Island Ave NW
New York Ave NW
New York Ave NE
6th St NW
7th St NW
Shaw-Howard Univ
Le Diplomate
Kramerbooks & Afterwords Cafe
Duke's Grocery
Whole Foods Market
Dupont Circle
Convivial
Giant Food
the pig
Shake Shack
Washington Plaza
Long View Gallery
National Geographic Museum
Walter E. Washington Convention Center
Marriott Marquis Washington, DC
Nordstrom Rack Downtown DC
Farragut North
Farragut Square
Franklin Square
Farragut West
McPherson Square
Busboys and Poets
Union Market
NoMa-Gallaudet U New York Ave
Harris Teeter
REI Co-op
Chinatown
Walmart Supercenter
National Museum of Women in the Arts
Renwick Gallery
Lafayette Square
Grand Hyatt Washington
Old Ebbitt Grill
Capital One Arena
National Portrait Gallery
National Building Museum
Union Station
Union Station Amtrak Train
The White House
Penn Social
Ford's Theatre
National Law Enforcement Officers Memorial
Hyatt Regency Washington On Capitol...
National Christmas Tree
Freedom Plaza
Rasika
Ronald Reagan Building and International Trade...
DAR Constitution Hall
The Ellipse
Federal Triangle
Organization of American States
National Archives Research Center
Newseum
Smithsonian National Museum of...
Constitution Ave NW
National Museum of African American...
Smithsonian National Museum of...
United States Capitol
Google
Map data ©2019 Google
Terms of Use
Report a map error

Pedestrians




Minor Injury
Major Injury
Fatal

FATALITIES & INJURIES

Do most occur after sunset?

Data wrangling

Sunrise/sunset data for Washington DC in 2018

```
['01  0727 1657  0714 1729  0641 1801  0553 1832  0511 1900  0445 1927  0446 1937  0509 1919  0537 1839  0604 1751
0635 1708  0707 1647\n', '02  0727 1658  0714 1730  0639 1802  0552 1832  0509 1901  0444 1928  0447 1937  0510 191
8  0538 1837  0605 1750  0636 1707  0708 1646\n', '03  0727 1658  0713 1732  0638 1803  0550 1833  0508 1902  0444
1929  0447 1937  0511 1917  0539 1836  0606 1748  0637 1706  0709 1646\n', '04  0727 1659  0712 1733  0637 1804  05
49 1834  0507 1903  0444 1929  0448 1937  0512 1916  0540 1834  0607 1746  0638 1705  0710 1646\n', '05  0727 1700
0711 1734  0635 1805  0547 1835  0506 1904  0444 1930  0449 1937  0513 1915  0540 1833  0608 1745  0639 1704  0711
1646\n', '06  0727 1701  0710 1735  0634 1806  0545 1836  0505 1905  0443 1931  0449 1936  0513 1914  0541 1831  06
09 1743  0641 1702  0712 1646\n', '07  0727 1702  0709 1736  0632 1807  0544 1837  0504 1906  0443 1931  0450 1936
0514 1913  0542 1829  0609 1742  0642 1701  0713 1646\n', '08  0727 1703  0708 1737  0631 1808  0542 1838  0503 190
7  0443 1932  0450 1936  0515 1912  0543 1828  0610 1740  0643 1701  0714 1646\n', '09  0727 1704  0707 1739  0629
1809  0541 1839  0502 1908  0443 1932  0451 1936  0516 1911  0544 1826  0611 1739  0644 1700  0715 1646\n', '10  07
27 1705  0705 1740  0628 1810  0539 1840  0501 1909  0443 1933  0452 1935  0517 1909  0545 1825  0612 1737  0645 16
59  0716 1646\n', '11  0726 1706  0704 1741  0626 1811  0538 1841  0500 1910  0442 1933  0452 1935  0518 1908  0546
1823  0613 1736  0646 1658  0716 1646\n', '12  0726 1707  0703 1742  0624 1812  0536 1842  0459 1911  0442 1934  04
53 1934  0519 1907  0547 1822  0614 1734  0647 1657  0717 1646\n', '13  0726 1708  0702 1743  0623 1813  0535 1843
0458 1912  0442 1934  0454 1934  0520 1906  0548 1820  0615 1733  0648 1656  0718 1647\n', '14  0726 1709  0701 174
4  0621 1814  0534 1844  0457 1913  0442 1935  0454 1933  0521 1904  0548 1818  0616 1731  0649 1655  0719 1647\n',
'15  0725 1710  0700 1745  0620 1815  0532 1845  0456 1914  0442 1935  0455 1933  0522 1903  0549 1817  0617 1730
0651 1654  0719 1647\n', '16  0725 1711  0658 1747  0618 1816  0531 1846  0455 1915  0442 1935  0456 1932  0522 190
```

Converted rows of data to a dictionary.

```
Out[43]: {1: ['01',
              '0727 1657',
              '0714 1729',
              '0641 1801',
              '0553 1832',
              '0511 1900',
              '0445 1927',
              '0446 1937',
              '0509 1919',
              '0537 1839',
              '0604 1751',
              '0635 1708',
              '0707 1647\n'],
             2: ['02',
              '0727 1658',
              '0714 1730',
              '0639 1802',
              '0552 1832',
              '0509 1901',
```

… then, converted to dictionary of tuples containing sunrise / sunset values.


```
{0: [('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A')
, ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), (
'N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/
A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A'), ('N/A',
'N/A'), ('N/A', 'N/A'), ('N/A', 'N/A')], 1: [('0727', '1657'), ('0727', '1658'), ('0727', '1658'), ('0727', '1659')
, ('0727', '1700'), ('0727', '1701'), ('0727', '1702'), ('0727', '1703'), ('0727', '1704'), ('0727', '1705'), ('072
6', '1706'), ('0726', '1707'), ('0726', '1708'), ('0726', '1709'), ('0725', '1710'), ('0725', '1711'), ('0724', '17
12'), ('0724', '1713'), ('0724', '1714'), ('0723', '1716'), ('0722', '1717'), ('0722', '1718'), ('0721', '1719'), (
'0721', '1720'), ('0720', '1721'), ('0719', '1722'), ('0719', '1724'), ('0718', '1725'), ('0717', '1726'), ('0716',
'1727'), ('0715', '1728')], 2: [('0714', '1729'), ('0714', '1730'), ('0713', '1732'), ('0712', '1733'), ('0711', '1
734'), ('0710', '1735'), ('0709', '1736'), ('0708', '1737'), ('0707', '1739'), ('0705', '1740'), ('0704', '1741'),
('0703', '1742'), ('0702', '1743'), ('0701', '1744'), ('0700', '1745'), ('0658', '1747'), ('0657', '1748'), ('0656'
, '1749'), ('0655', '1750'), ('0653', '1751'), ('0652', '1752'), ('0651', '1753'), ('0649', '1754'), ('0648', '1755
'), ('0647', '1756'), ('0645', '1757'), ('0644', '1759'), ('0642', '1800'), ('0641', '1801'), ('N/A', 'N/A'), ('N/A
', 'N/A')], 3: [('0641', '1801'), ('0639', '1802'), ('0638', '1803'), ('0637', '1804'), ('0635', '1805'), ('0634',
'1806'), ('0632', '1807'), ('0631', '1808'), ('0629', '1809'), ('0628', '1810'), ('0626', '1811'), ('0624', '1812')
, ('0623', '1813'), ('0621', '1814'), ('0620', '1815'), ('0618', '1816'), ('0617', '1817'), ('0615', '1818'), ('061
4', '1819'), ('0612', '1820'), ('0610', '1821'), ('0609', '1822'), ('0607', '1823'), ('0606', '1824'), ('0604', '18
25'), ('0603', '1826'), ('0601', '1827'), ('0559', '1828'), ('0558', '1829'), ('0556', '1830'), ('0555', '1831')],
```

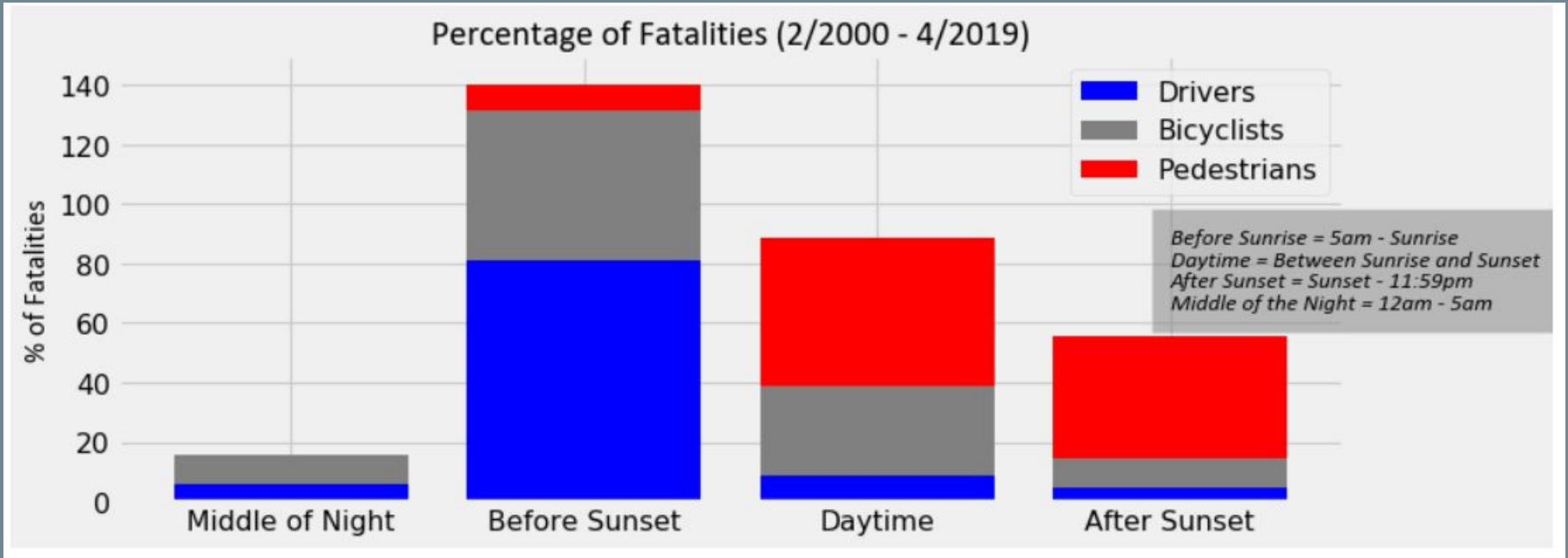
… which was then converted to a Data Frame containing sunrise / sunset data for each day of the year.

	X	January	February	March	April	May	June	July	August	September	October	November	December
0	(N/A, N/A)	(0727, 1657)	(0714, 1729)	(0641, 1801)	(653, 1932)	(611, 2000)	(545, 2027)	(546, 2037)	(609, 2019)	(637, 1939)	(704, 1851)	(735, 1808)	(0707, 1647)
1	(N/A, N/A)	(0727, 1658)	(0714, 1730)	(0639, 1802)	(652, 1932)	(609, 2001)	(544, 2028)	(547, 2037)	(610, 2018)	(638, 1937)	(705, 1850)	(736, 1807)	(0708, 1646)
2	(N/A, N/A)	(0727, 1658)	(0713, 1732)	(0638, 1803)	(650, 1933)	(608, 2002)	(544, 2029)	(547, 2037)	(611, 2017)	(639, 1936)	(706, 1848)	(737, 1806)	(0709, 1646)
3	(N/A, N/A)	(0727, 1659)	(0712, 1733)	(0637, 1804)	(649, 1934)	(607, 2003)	(544, 2029)	(548, 2037)	(612, 2016)	(640, 1934)	(707, 1846)	(0638, 1705)	(0710, 1646)
4	(N/A, N/A)	(0727, 1700)	(0711, 1734)	(0635, 1805)	(647, 1935)	(606, 2004)	(544, 2030)	(549, 2037)	(613, 2015)	(640, 1933)	(708, 1845)	(0639, 1704)	(0711, 1646)

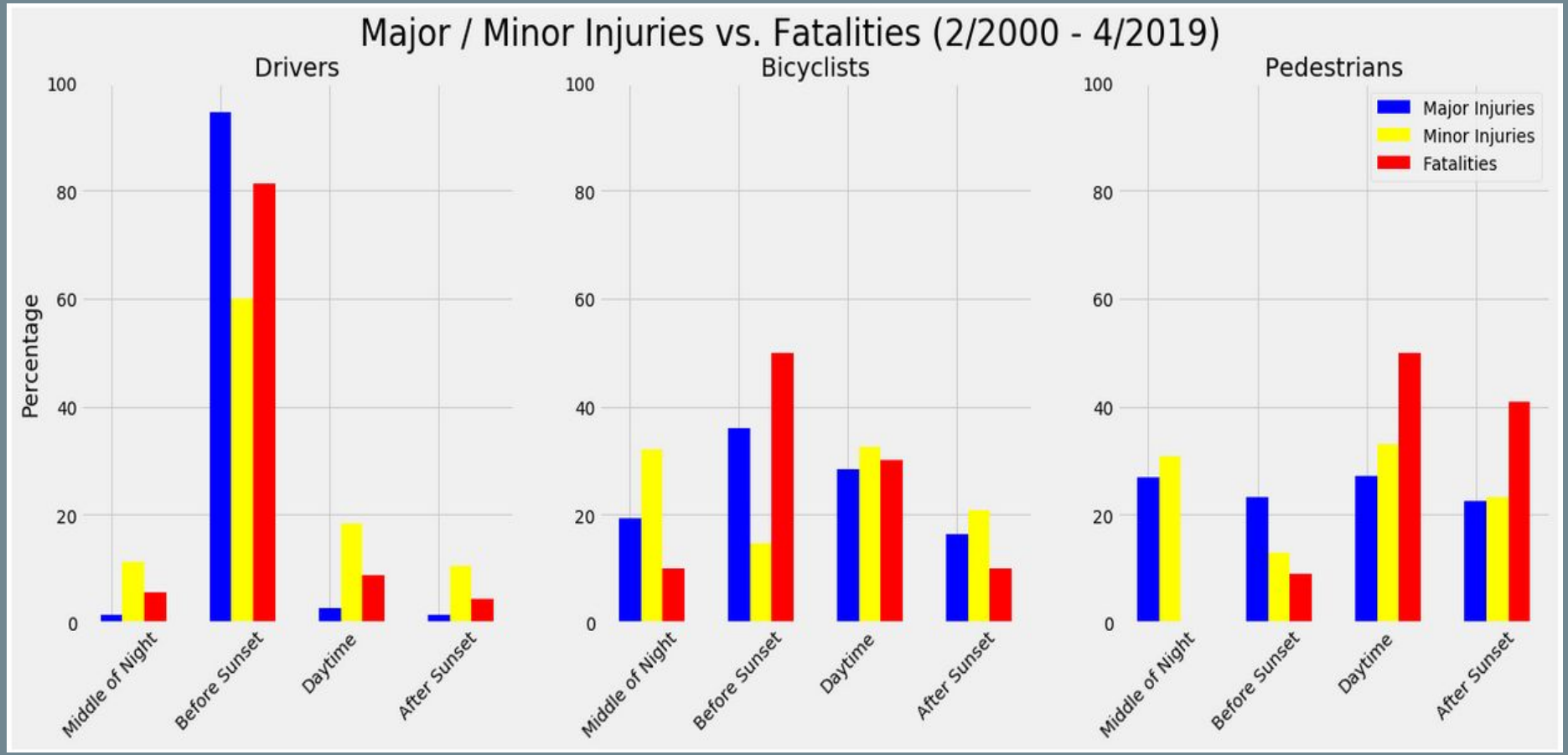
Updated Data Frame to account for Daylight Savings Time

Website instructs to add 1 hour for daylight savings time. 2018 data. Daylight savings time in 2018: March 11th - November 3rd.

```
# Incorporate daylight savings time change for April through October
for day, month in sunrises_sunsets.iterrows():
    for col in range(4,11):
        if month[col][0] == "N/A":
            continue

        rise = int(month[col][0]) + 100
        set = int(month[col][1]) + 100

        sunrises_sunsets.iloc[day, col] = (str(rise), str(set))

# Incorporate time change for March 11th - March 31st
for i in range(10,31):
    rise = sunrises_sunsets.iloc[i, 3][0]
    updated_rise = int(rise) + 100

    set = sunrises_sunsets.iloc[i, 3][1]
    updated_set = int(set) + 100

    sunrises_sunsets.iloc[i, 3] = (str(updated_rise), str(updated_set))

# Incorporate time change for November 1st - November 3rd
for i in range(0,3):
    rise = sunrises_sunsets.iloc[i, 11][0]
    updated_rise = int(rise) + 100

    set = sunrises_sunsets.iloc[i, 11][1]
    updated_set = int(set) + 100

    sunrises_sunsets.iloc[i, 11] = (str(updated_rise), str(updated_set))
```



Time Frame	Drivers	Bicyclists	Pedestrians
1) Middle of Night	6.0	10.0	0.0
2) Before Sunrise	81.0	50.0	9.0
3) Daytime	9.0	30.0	50.0
4) After Sunset	4.0	10.0	41.0


Major / Minor Injuries vs. Fatalities (2/2000 - 4/2019)
Drivers
Bicyclists
Pedestrians
Percentage
100
80
60
40
20
0
Middle of Night
Before Sunset
Daytime
After Sunset
Major Injuries
Minor Injuries
Fatalities

Data Analysis

- Used report date/time with sunrise/sunset times to determine when accident occurred
- Analyzed different types of data (types of cars involved in fatalities, major/minor injuries, etc.) to determine what told the best story
- Experimented with different types of graphs (stacked bar, bar, line, pie)

Data Exploration Insights

Indicators of Fatalities / Injuries were not correct.

```
num = len(merged_data[(merged_data['FATAL']=='N') & ((merged_data['FATAL_DRIVER']!=0) | (merged_data['FATAL_BICYCLIST']
                                                    (merged_data['FATAL_PEDESTRIAN']!=0))])
print(f"Number of rows having FATAL = 'N' and actual fatalities noted: {num}")
```

```
Number of rows having FATAL = 'N' and actual fatalities noted: 249
```

```
num = (len(merged_data[(merged_data['MAJORINJURY']=='N') & ((merged_data['MAJORINJURIES_DRIVER']!=0) |
                                                            (merged_data['MAJORINJURIES_BICYCLIST']!=0) &
                                                            (merged_data['MAJORINJURIES_PEDESTRIAN']!=0))]))
print(f"Number of rows having MAJORINJURY = 'N' and actual major injuries noted: {num}")
```

```
Number of rows having MAJORINJURY = 'N' and actual major injuries noted: 23257
```

Problems encountered

- Missing report dates caused erroneous float error
- Majority of data had Report Date of 5am
- Incorrect Fatality / Injury indicators limited results

Resolution

- Removed missing report dates
- Accepted data as is
- Checked columns containing number of fatalities and injuries

Conclusions

- Most driver and bicycle fatalities were between 5am and sunrise (in the dark)
- Most pedestrian fatalities were during the daytime hours
- Percentage of bicycle and pedestrian injuries were consistent throughout the day
- Data may have been bulk uploaded and may not contain actual time of accident

When Are Accidents More Likely to Happen in DC?

- Which months?
- Which hours?
- Which days?

1st Step: Converted datetime and then extract data for each year.

```
s2014 = df['2014']
s2015 = df['2015']
s2016 = df['2016']
s2017 = df['2017']
s2018 = df['2018']
```

2nd Step: Perform groupby [Month, Hour, & Day] and CRIME _ID.

```
numAccidentMonthly = s2014.groupby([s2014.index.month,s2014['CRIMEID']]).agg({'count'})
monthYearly = (numAccidentMonthly.iloc[:,0].groupby('datetime').count())
yearlyCount['2014'] = monthYearly
```

	CRIMEID	REPORTDATE	VEHICLEID	INVEHICLETYPE	LICENSEPLATESTATE
0	27674721	2018-05-13T21:15:36.000Z	3884008	Passenger Car/automobile	VA
1	27112008	2016-11-01T02:26:03.000Z	2757090	Passenger Car/automobile	DC
2	27112008	2016-11-01T02:26:03.000Z	2757091	Large/heavy Truck	IL
3	24490573	2013-03-17T05:00:00.000Z	927946	Passenger Car/automobile	Un
4	27112014	2016-11-01T02:31:25.000Z	2753149	Passenger Car/automobile	MD
5	27112014	2016-11-01T02:31:25.000Z	2753148	Passenger Car/automobile	DC
6	24490575	2013-03-19T05:00:00.000Z	929019	Suv (sport Utility Vehicle)	DC
7	24490575	2013-03-19T05:00:00.000Z	929018	Suv (sport Utility Vehicle)	MD
8	24490575	2013-03-19T05:00:00.000Z	929018	Suv (sport Utility Vehicle)	MD
9	24490575	2013-03-19T05:00:00.000Z	929019	Suv (sport Utility Vehicle)	DC
10	27237816	2017-03-06T01:17:05.000Z	3009111	Passenger Car/automobile	MD
11	27237816	2017-03-06T01:17:05.000Z	3009110	Passenger Car/automobile	VA
12	27112023	2016-11-01T02:40:28.000Z	2755013	Passenger Car/automobile	MD
13	27112023	2016-11-01T02:40:28.000Z	2753141	Passenger Car/automobile	DC

4th step: Pivot tables

datetime	2014	2015	2016	2017	2018
1	1378	1538	1889	2113	2024
2	1305	1458	1806	1836	1866
3	1548	1723	2266	2355	2193
4	1649	1865	2289	2341	2232
5	1788	1832	2383	2463	2402
6	1712	1647	2433	2295	2378
7	1660	1876	2287	2248	2278
8	1641	1787	2184	2097	2218
9	1753	2089	2342	2269	2278
10	1839	2435	2430	2269	2393
11	1592	2088	2200	2126	2127
12	1525	1974	2068	2074	2151

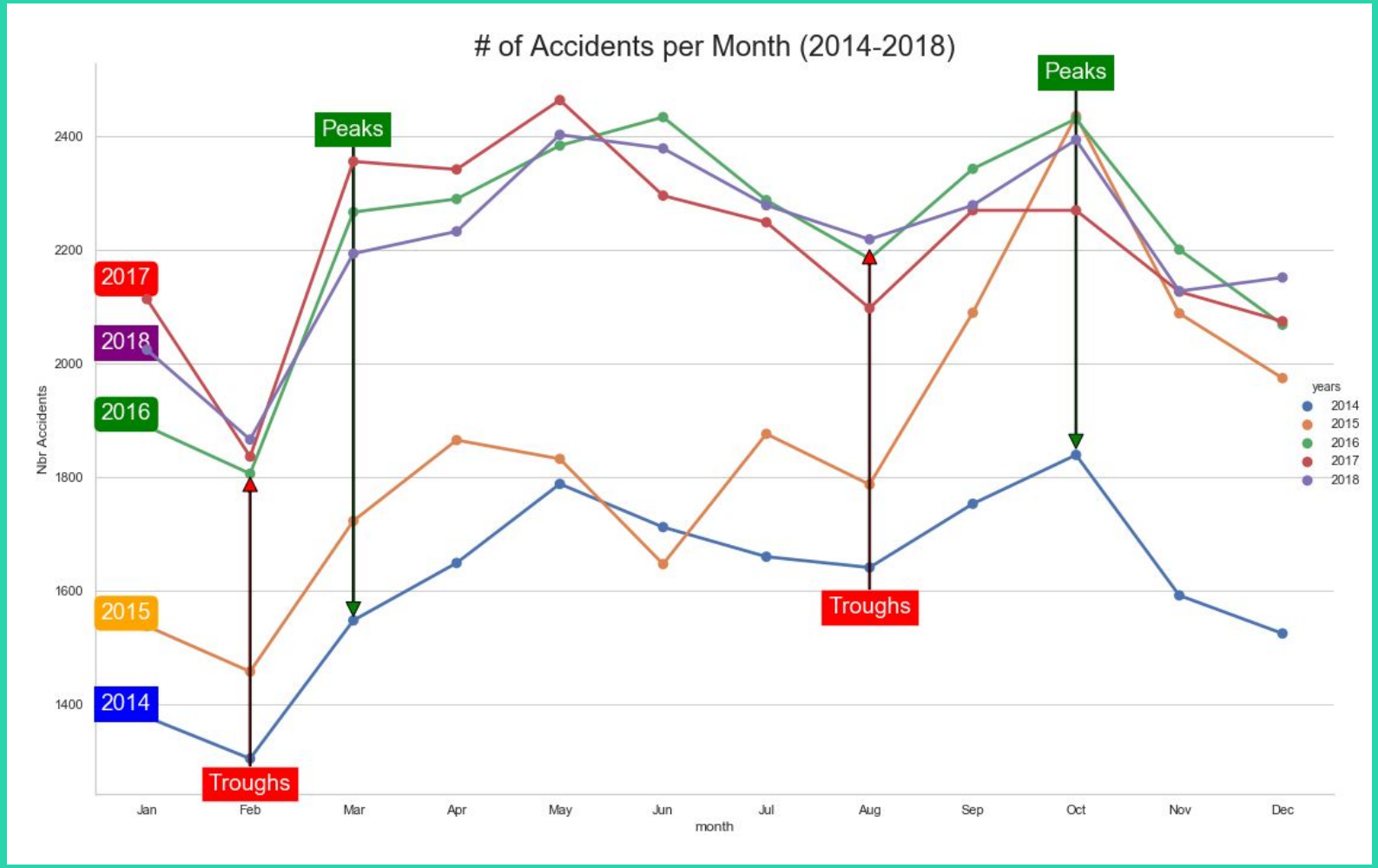
	month	2014	2015	2016	2017	2018
0	Jan	1378	1538	1889	2113	2024
1	Feb	1305	1458	1806	1836	1866
2	Mar	1548	1723	2266	2355	2193
3	Apr	1649	1865	2289	2341	2232
4	May	1788	1832	2383	2463	2402

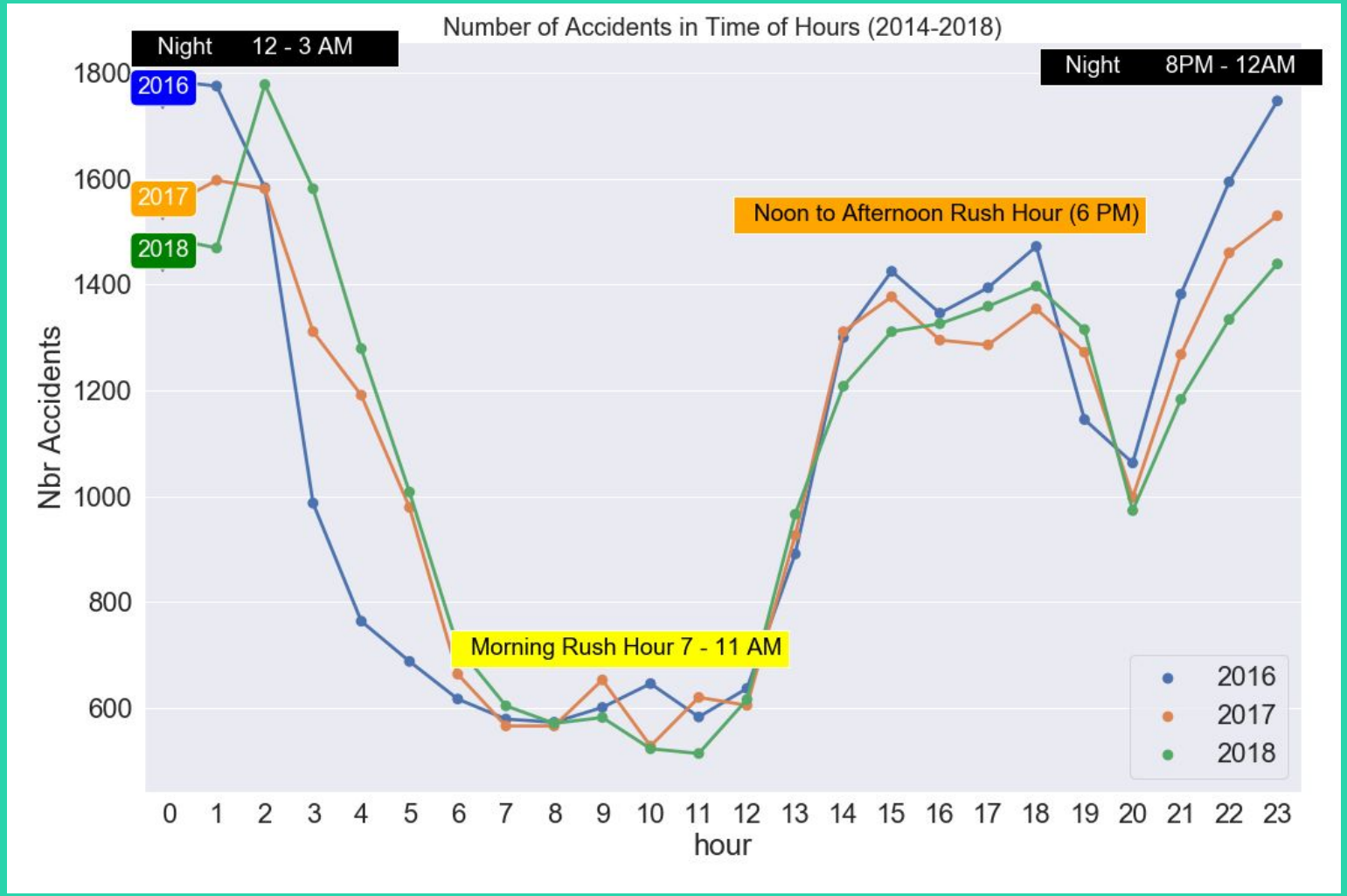
5th step:
UnPivot table

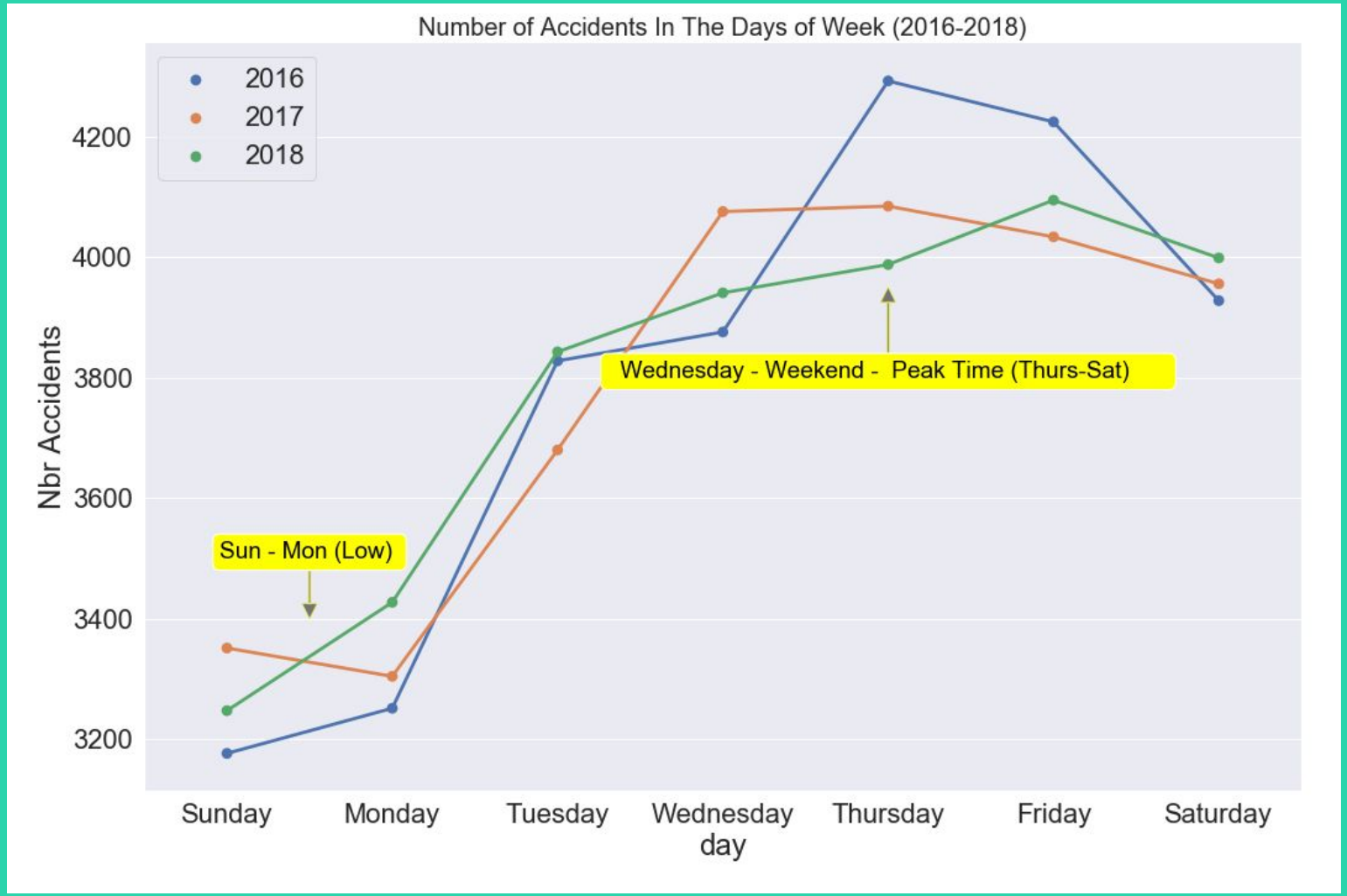
pd.melt()

```
YearlyCount_new (data)
id_vars='month'
var_name="years"
value_name="Nbr Accidents"
```

	month	years	Nbr Accidents
0	Jan	2014	1378
1	Feb	2014	1305
2	Mar	2014	1548
3	Apr	2014	1649
4	May	2014	1788
5	Jun	2014	1712
6	Jul	2014	1660
7	Aug	2014	1641
8	Sep	2014	1753
9	Oct	2014	1839
10	Nov	2014	1592
11	Dec	2014	1525
12	Jan	2015	1538
13	Feb	2015	1458
14	Mar	2015	1723
15	Apr	2015	1865
16	May	2015	1832


of Accidents per Month (2014-2018)
Peaks
Peaks
2017
2018
2016
2015
2014
Troughs
Troughs
Nbr Accidents
2400
2200
2000
1800
1600
1400
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
month
years
2014
2015
2016
2017
2018


Number of Accidents in Time of Hours (2014-2018)
Night 12 - 3 AM
Night 8PM - 12AM
2016
2017
2018
Noon to Afternoon Rush Hour (6 PM)
Morning Rush Hour 7 - 11 AM
Nbr Accidents
hour
1800
1600
1400
1200
1000
800
600
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
2016
2017
2018


Number of Accidents In The Days of Week (2016-2018)
2016
2017
2018
Nbr Accidents
4200
4000
3800
3600
3400
3200
Sun - Mon (Low)
Wednesday - Weekend - Peak Time (Thurs-Sat)
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
day

Data Analysis

- Used GROUPBY on time/date and crimeid to count number of accidents as much accurate as it can.
- Experimented with grouped bars first then settled with line charts because there are obvious patterns.
- Research on weather accident-related online.

Problems encountered

- Had no problem with missing REPORTID ????
- Prior to year 2014, approx. 14 - 20 K recorded at 5 AM
- Much less data before year 2014
- Need to have background information on how they collected data
- Need more investigating on how CRIMEID is implemented.
- Use 12hr format is a problematic. No official function is found as of now just like days/month.
- Using Jupyter notebook is not reliability and consistency but it’s a powerful tool.
- My brand new laptop crashed. **Remember to backup regularly!**

Conclusion

- **Accidents more likely to occur in DC**
  - *In the months of March and October.*
  - *During the night and afternoon.*
  - *Thursday to Saturday*
- **Surprising - fewer accidents**
  - *During the morning rush hours*
  - *In the month of February*
- **Obviously accidents are not related to weathers.**

Crash Analysis by Operators Related to Age Group

What age group causes the most accidents?

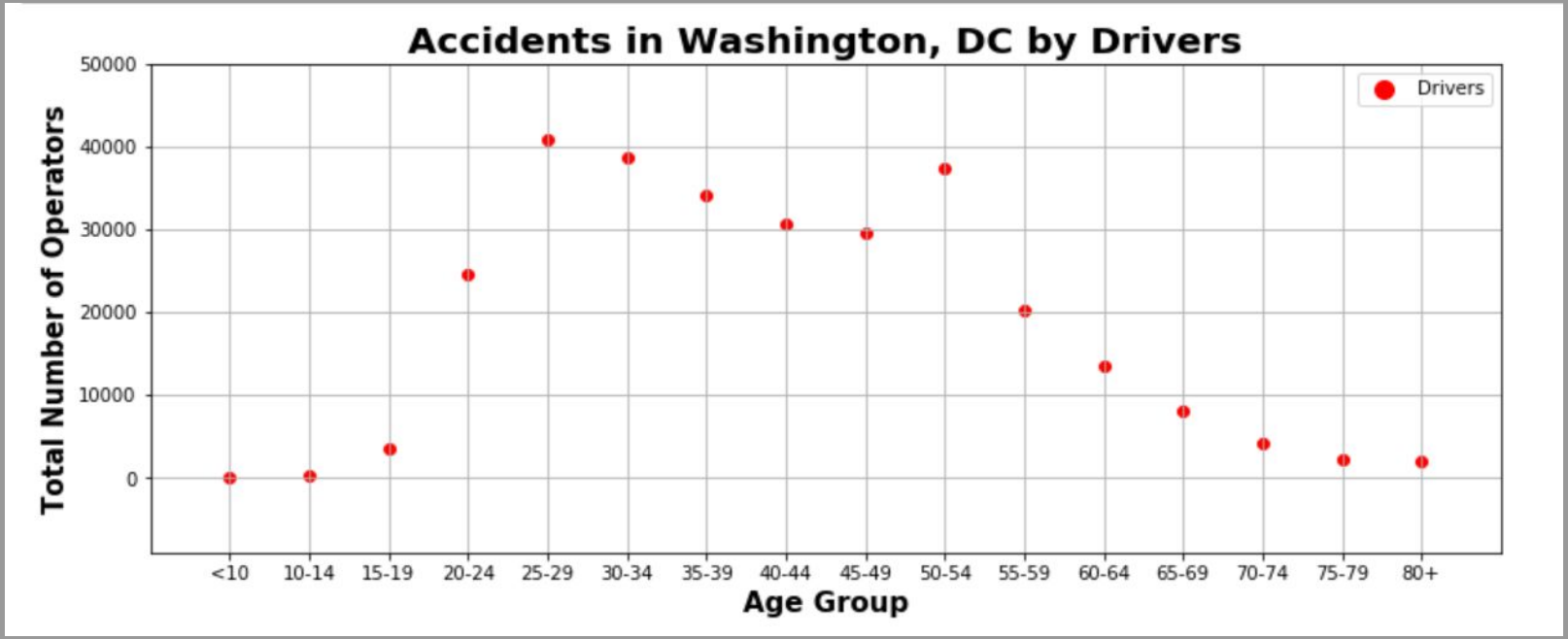
Data Analysis

- Summarized average percent accidents for drivers, bicyclists and pedestrians by age group.
- Highest percent of accidents in age group is between 25- 29.
- Next highest percent of accidents in age group 30 - 34.
- Among the operators, drivers holds the highest number of accidents, pedestrians second, and bicylists the third.

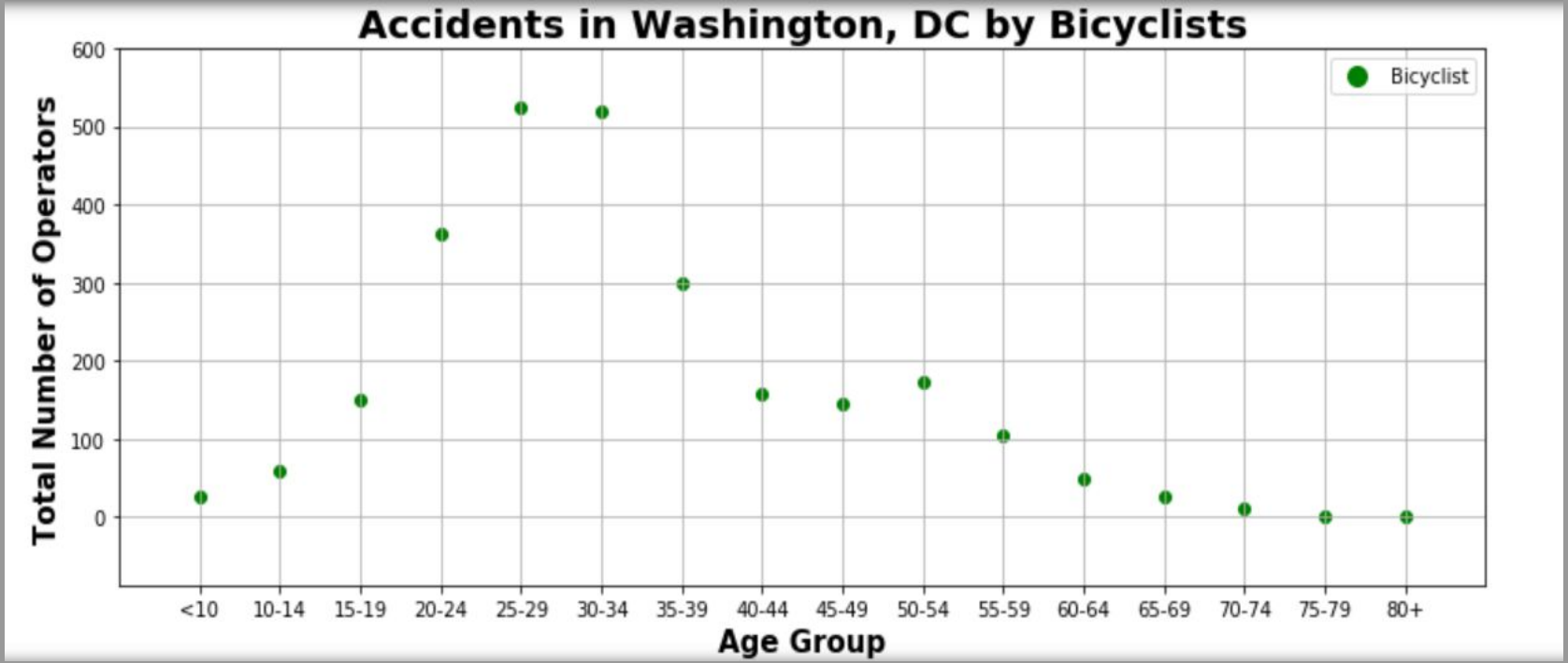
	NUMBER OF CRASHES			PERCENTAGE		
Age Group	Drivers	Bicyclist	Pedestrian	Drivers	Bicyclist	Pedestrian
< less than 10	138	25	199	5%	96%	511%
10-14	214	60	193	7%	230%	496%
15-19	3585	149	214	124%	571%	550%
20-24	24575	363	414	847%	1392%	1063%
25-29	40817	525	544	1407%	2013%	1397%
30-34	38770	518	454	1336%	1986%	1166%
35-39	34176	298	310	1178%	1143%	796%
40-44	30644	157	233	1056%	602%	599%
45-49	29626	146	233	1021%	560%	599%
50-54	37342	174	370	1287%	667%	950%
55-59	20180	104	234	696%	399%	601%
60-64	13573	48	215	468%	184%	552%
65-69	8005	27	138	276%	104%	354%
70-74	4139	12	69	143%	46%	177%
75-79	2252	1	42	78%	4%	108%
80+	2079	1	31	72%	4%	80%
TOTAL	290115	2608	3893	10000%	10000%	10000%

*Note: The average is the sum of the total value divided by the count.

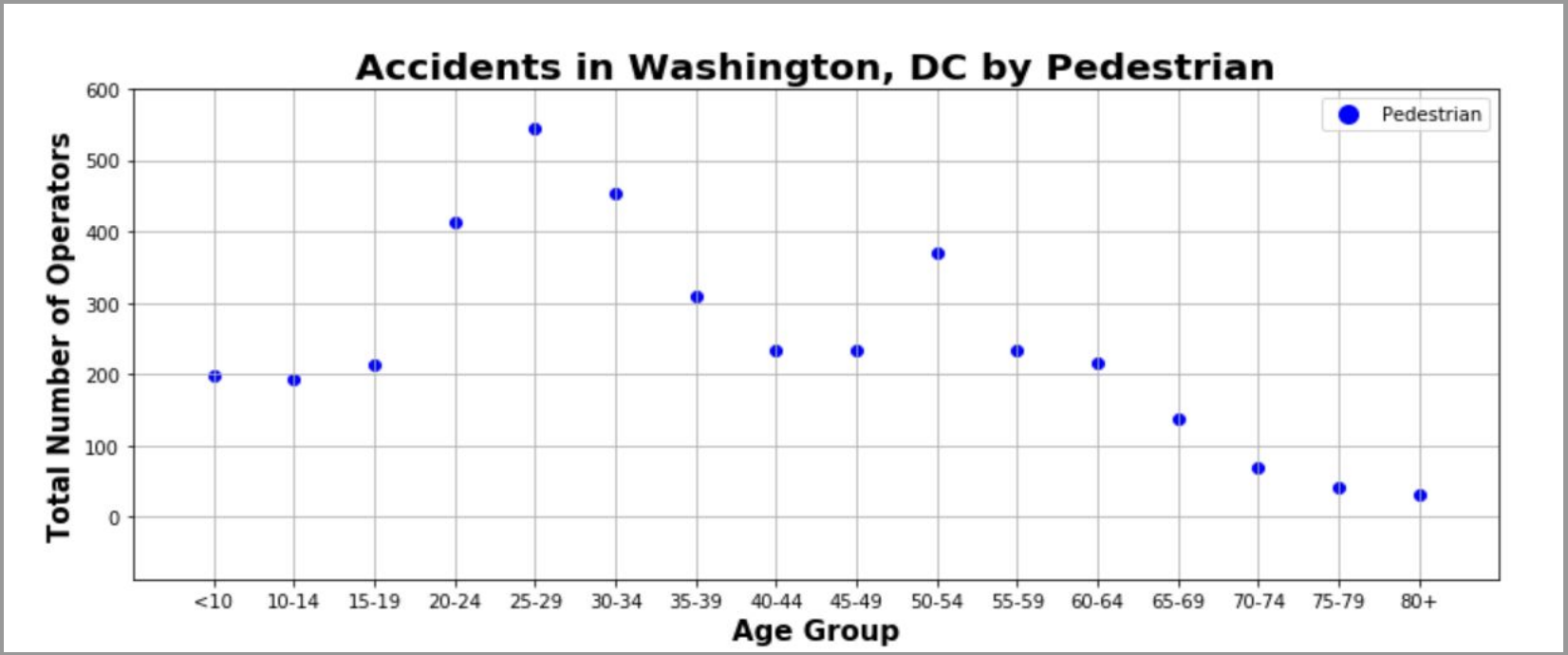
Referencing the table (page 42), the figure below shows the trends in total number of accidents by drivers/motorists related to age.



Referencing the table (page 42), the figure below shows the trends in total number of accidents by bicyclists related to age.



Referencing the table (page 42), the figure below shows the trends in total number of accidents by pedestrians related to age.



Data Exploration Insights

Jupyter Notebook was exhausted to generate the output of the data analysis and visualization.

Data 1:

```
In [8]: new_df.AGE.unique()

Out[8]: array([   0,   28,   23,   54,   60,   64,   32,   46,   39,
                 41,   57,   38,   70,   24,   35,   45,   52,   53,
                 33,   27,   63,   31,   49,   43,   62,   61,   40,
                 65,   47,   36,   48,   26,   59,   21,   74,   25,
                 56,   18,   50,   58,   42,   34,   19,   30,   20,
                 79,   44,   66,   22,   67,   55,   37,   68,   29,
                 87,   51,   76,   69,   71,   84,   78,   82,   72,
                 11,   10,   13,   75,   80,   98,   81,   73,   90,
                  4,   17,    2,    3,   85,   77,   94,   16,   86,
                118,   15,  100,   83,   89,    5,   88,   92,    9,
                 14,   -2,    8,   12,   91,    6,    7,    1,   95,
                117,   96,   97,   93,  119,   -1,  105,  104,  -80,
                 99,  156,   -3,  -38,  -19,  103, -7991,  -26,  112,
                109,  -35,  102,  115,  -49,  137,  116,  114,  -96,
                -67,  -10,  -28,  220,  -24,  207,  -34, -964,  110,
               -182,   -6,  -36,  139,  157,  237, -184,   -5,  108,
                 -4,  167,  -70,  142,  215,  113,  125,  138,  -37,
                -13,  146,  133,   -9,  126,  111,  -47,  218,  -25,
                -23,  -56,  -51,  -58,  101,   -8], dtype=int64)
```

Data 2:

```
In [8]: new_df.AGE.unique()
```

```
Out[8]: array([    0,    28,    23,    54,    60,    64,    32,    46,    39,
                  41,    57,    38,    70,    24,    35,    45,    52,    53,
                  33,    27,    63,    31,    49,    43,    62,    61,    40,
                  65,    47,    36,    48,    26,    59,    21,    74,    25,
                  56,    18,    50,    58,    42,    34,    19,    30,    20,
                  79,    44,    66,    22,    67,    55,    37,    68,    29,
                  87,    51,    76,    69,    71,    84,    78,    82,    72,
                  11,    10,    13,    75,    80,    98,    81,    73,    90,
                   4,    17,     2,     3,    85,    77,    94,    16,    86,
                 118,    15,   100,    83,    89,     5,    88,    92,     9,
                  14,    -2,     8,    12,    91,     6,     7,     1,    95,
                 117,    96,    97,    93,   119,    -1,   105,   104,   -80,
                  99,   156,    -3,   -38,   -19,   103, -7991,   -26,   112,
                 109,   -35,   102,   115,   -49,   137,   116,   114,   -96,
                 -67,   -10,   -28,   220,   -24,   207,   -34,  -964,   110,
                -182,    -6,   -36,   139,   157,   237,  -184,    -5,   108,
                  -4,   167,   -70,   142,   215,   113,   125,   138,   -37,
                 -13,   146,   133,    -9,   126,   111,   -47,   218,   -25,
                 -23,   -56,   -51,   -58,   101,    -8], dtype=int64)
```

Data 3:

```
In [9]: # Bins for ages
        age_bins = [0, 9.90, 14.90, 19.90, 24.90, 29.90, 34.90, 39.90, 44.90, 49.90, 56.90, 61.90, 66.90, 71.90, 76.99, 81.90, 99999]
        group_names = ["<10", "10-14", "15-19", "20-24", "25-29", "30-34", "35-39", "40-44", "45-49", "50-54", "55-59", "60-64", "65-

        # Group and sort age values into bins noted above
        crash_details["Age Group"] = pd.cut(crash_details["AGE"],age_bins, labels=group_names)
        crash_details

        # Data frame for added "Age Group" and groupby
        age_grouped = crash_details.groupby("PERSONTYPE")
```

Data 4:

```
In [13]: Ptype_drivers=drivers_data.groupby("Age Group")["PERSONTYPE"].count().reset_index()
         Ptype_drivers
```

Out[13]:

	Age Group	PERSONTYPE
0	<10	138
1	10-14	214
2	15-19	3585
3	20-24	24575
4	25-29	40817
5	30-34	38770
6	35-39	34176
7	40-44	30644
8	45-49	29626
9	50-54	37342
10	55-59	20180
11	60-64	13573
12	65-69	8005
13	70-74	4139
14	75-79	2252
15	80+	2079

Data 5:

```
In [14]: Ptype_bicyclist=bicyclist_data.groupby("Age Group")["PERSONTYPE"].count().reset_index()
         Ptype_bicyclist
```

Out[14]:

	Age Group	PERSONTYPE
0	<10	25
1	10-14	60
2	15-19	149
3	20-24	363
4	25-29	525
5	30-34	518
6	35-39	298
7	40-44	157
8	45-49	146
9	50-54	174
10	55-59	104
11	60-64	48
12	65-69	27
13	70-74	12
14	75-79	1
15	80+	1

Data 6:

In [15]:
```
Ptype_pedestrian=pedestrian_data.groupby("Age Group")["PERSONTYPE"].count().reset_index()
Ptype_pedestrian
```

Out[15]:

	Age Group	PERSONTYPE
0	<10	199
1	10-14	193
2	15-19	214
3	20-24	414
4	25-29	544
5	30-34	454
6	35-39	310
7	40-44	233
8	45-49	233
9	50-54	370
10	55-59	234
11	60-64	215
12	65-69	138
13	70-74	69
14	75-79	42
15	80+	31

Data 7:

In [16]:

```
#Build a scatter plot for Drivers
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.pyplot

#fig = plt.figure(figsize=(15,9))
fig = plt.figure(figsize=(13,5))
#plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)
plt.grid(True)
sct_drivers=plt.scatter(x=Ptype_drivers["Age Group"],
            y=Ptype_drivers["PERSONTYPE"], label="Drivers", color=['red'])


# Set the legends then set equal sizes for legend pts
lgnd = plt.legend(handles=[sct_drivers],loc="best")
lgnd.legendHandles[0]._sizes = [100]
lgnd.legendHandles[0].set_color('red')


# Set texts and positions on the chart
#plt.text(3,5," ",fontsize=15)
# Set texts and positions on the chart
plt.text(9,30000, " ", fontsize=35)

# Incorporate the other graph properties
plt.title("Accidents in Washington, DC by Drivers", fontsize=20, fontweight='bold')
plt.xlabel ("Age Group", fontsize=15, fontweight='bold')
plt.ylabel ("Total Number of Operators", fontsize=15, fontweight='bold')
plt.grid(True)
plt.xlim(-1,16)
plt.ylim(-9000,50000)

# Show plot
plt.show()

# Save the figure
plt.savefig("drivers.png")
```

Data 8:

In [17]:

```
#Build a scatter plot for Bicyclists (%)
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.pyplot

#fig = plt.figure(figsize=(15,9))
fig = plt.figure(figsize=(13,5))
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)
plt.grid(True)
sct_bicyclist=plt.scatter(x=Ptype_bicyclist["Age Group"],
            y=Ptype_bicyclist["PERSONTYPE"], label="Bicyclist", color=['green'])


# Set the legends then set equal sizes for legend pts
lgnd = plt.legend(handles=[sct_bicyclist],loc="best")
lgnd.legendHandles[0]._sizes = [100]
lgnd.legendHandles[0].set_color('green')


# Set texts and positions on the chart
#plt.text(3,5," ",fontsize=15)
# Set texts and positions on the chart
plt.text(12,100000, " ", fontsize=35)

# Incorporate the other graph properties
plt.title("Accidents in Washington, DC by Bicyclists", fontsize=20, fontweight='bold')
plt.xlabel ("Age Group", fontsize=15, fontweight='bold')
plt.ylabel ("Total Number of Operators", fontsize=15, fontweight='bold')
plt.grid(True)
plt.xlim(-1,16)
plt.ylim(-87,600)

# Show plot
plt.show()

# Save the figure
plt.savefig("bicyclist.png")
```

Data 9:

In [18]:

```
#Build a scatter plot for Pedestrians
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import matplotlib.pyplot

#fig = plt.figure(figsize=(15,9))
fig = plt.figure(figsize=(13,5))
#plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)
plt.grid(True)
sct_pedestrian=plt.scatter(x=Ptype_pedestrian["Age Group"],
            y=Ptype_pedestrian["PERSONTYPE"], label="Pedestrian", color=['blue'])


# Set the legends then set equal sizes for legend pts
lgnd = plt.legend(handles=[sct_pedestrian],loc="best")
lgnd.legendHandles[0]._sizes = [100]
lgnd.legendHandles[0].set_color('blue')


# Set texts and positions on the chart
#plt.text(3,5," ",fontsize=15)
# Set texts and positions on the chart
plt.text(9,30000, " ", fontsize=35)

# Incorporate the other graph properties
plt.title("Accidents in Washington, DC by Pedestrian", fontsize=20, fontweight='bold')
plt.xlabel ("Age Group", fontsize=15, fontweight='bold')
plt.ylabel ("Total Number of Operators", fontsize=15, fontweight='bold')
plt.grid(True)
plt.xlim(-1,16)
plt.ylim(-87,600)

# Show plot
plt.show()

# Save the figure
plt.savefig("pedestrian.png")
```

Problems encountered

- Data contains null information.
- Data contains “AGE” values with either negative or above 100.
- Data contains blank/missing “REPORTDATE” values.
- Rethinked on whether to accept the data as is.

Resolution

- Removed null data.
- Removed ages less than 0 (zero) and greater than 100.
- Removed missing report dates.
- Due to time constraints, accepted the data as is.

Conclusion

In conclusion, this analysis by age group finds that most number of accidents are within ages of 25-29 and are by far possess the greatest driving risk. In general, accidents increases quickly with age from the teens and 20’s, and continues to increase through the age of 30s, 40s, 50s, and 60s. Then, slowly decreasing from the 70s.

Recommendations

DC has these issues and would benefit from the following improvements:

- Reduce constant road construction.
- Simplify the complex structure and mixture of local, commercial, and long-distance commuter traffic.
- Enforce stronger restrictions for commercial truck drivers to follow state and federal regulations.

TWO MORE WEEKS

DATAVIZ

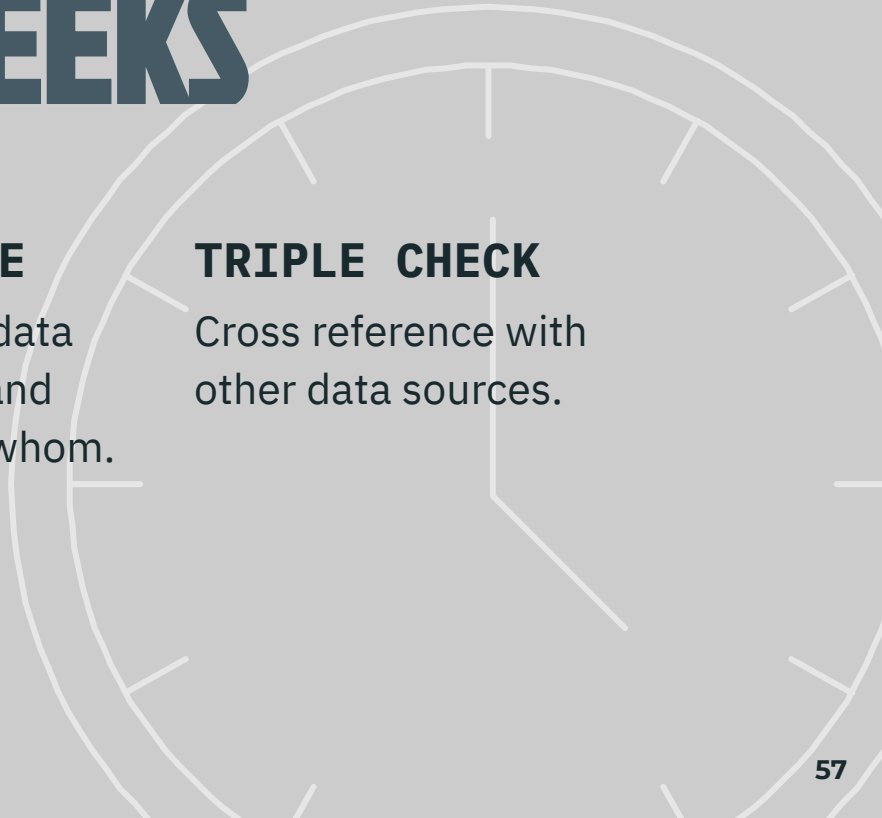
Brush up on Seaborn & improve readability of chart with design.

INVESTIGATE

Research how data was collected and logged and by whom.

TRIPLE CHECK

Cross reference with other data sources.




Safety first.
Questions?