week_1_extended

April 16, 2020

1 How can we control the increasing number of accidents in New York?

[36]:
```
import json
import requests
from bs4 import BeautifulSoup
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

1.1 Introduction

Business Context. The city of New York has seen a rise in the number of accidents on the roads in the city. They would like to know if the number of accidents have increased in the last few weeks. For all the reported accidents, they have collected details for each accident and have been maintaining records for the past year and a half (from January 2018 to August 2019).

The city has contracted you to build visualizations that would help them identify patterns in accidents, which would help them take preventive actions to reduce the number of accidents in the future. They have certain parameters like borough, time of day, reason for accident, etc. Which they care about and which they would like to get specific information on.

Business Problem. Your task is to format the given data and provide visualizations that would answer the specific questions the client has, which are mentioned below.

Analytical Context. You are given a CSV file containing details about each accident like date, time, location of the accident, reason for the accident, types of vehicles involved, injury and death count, etc. The delimiter in the given CSV file is `;` instead of the default `,`. You will be performing the following tasks on the data:

1. Extract data from Wikipedia
2. Read, transform, and prepare data for visualization
3. Perform analytics and construct visualizations of the data to identify patterns in the dataset

The client has a specific set of questions they would like to get answers to. You will need to provide visualizations to accompany these:

1. How have the number of accidents fluctuated over the past year and a half? Have they increased over the time?

2. For any particular day, during which hours are accidents most likely to occur?
3. Are there more accidents on weekdays than weekends?
4. What are the accidents count-to-area ratio per borough? Which boroughs have disproportionately large numbers of accidents for their size?
5. For each borough, during which hours are accidents most likely to occur?
6. What are the top 5 causes of accidents in the city?
7. What types of vehicles are most involved in accidents per borough?
8. What types of vehicles are most involved in deaths?

1.2 Fetch borough data from Wikipedia

The client has requested analysis of the accidents-to-area ratio for boroughs. You will need to fetch the area of each borough from the Wikipedia page: https://en.wikipedia.org/wiki/Boroughs_of_New_York_City.

Since we are fetching this resource from an external page, you should instead fetch the HTML document and store the results locally in a JSON file, so that you can parse it later when you need it. Create a folder named `data` and store the file inside it.

Answer. One possible solution is given below:

[37]:
```
response = requests.get('https://en.wikipedia.org/wiki/
↪Boroughs_of_New_York_City')
soup = BeautifulSoup(response.text)

borough_data = {}
table_body = soup.find('table', {'class': ['wikitable', 'sortable',␣
↪'jquery-tablesorter']}).find('tbody')

for row in table_body.find_all('tr'):
    cells = row.find_all('td')
    if cells and len(cells) == 9:
        name = cells[0].text.strip().lower()
        population = float(cells[2].text.replace(',', '').strip())
        area = float(cells[5].text.strip())
        borough_data[name] = {
            'name': name,
            'population': population,
            'area': area
        }

borough_data
```

[37]:
```
{'the bronx': {'name': 'the bronx', 'population': 1471160.0, 'area': 42.1},
 'brooklyn': {'name': 'brooklyn', 'population': 2648771.0, 'area': 70.82},
 'manhattan': {'name': 'manhattan', 'population': 1664727.0, 'area': 22.83},
 'queens': {'name': 'queens', 'population': 2358582.0, 'area': 108.53},
 'staten island': {'name': 'staten island',
```

```
 'population': 479458.0,
 'area': 58.37}}
```

For later usage, let's store the borough data into a JSON file in the already created `data` folder:

[38]:
```python
with open('data/borough_data.json', 'w+') as f:
    json.dump(borough_data, f)
```

1.3 Overview of the data

Now that we've stored the borough data in a JSON file, we can re-open it and use it whenever we wish. We can use the `read_json()` function in `pandas` to do that:

[39]:
```python
with open('data/data.json') as f:
    df = pd.read_json(f, orient='records')
```

Let's go through the columns present in the dataframe:

[24]:
```python
df.columns
```

[24]:
```
Index(['BOROUGH', 'COLLISION_ID', 'CONTRIBUTING FACTOR VEHICLE 1',
       'CONTRIBUTING FACTOR VEHICLE 2', 'CONTRIBUTING FACTOR VEHICLE 3',
       'CONTRIBUTING FACTOR VEHICLE 4', 'CONTRIBUTING FACTOR VEHICLE 5',
       'DATE', 'DATETIME', 'LATITUDE', 'LONGITUDE',
       'NUMBER OF CYCLIST INJURED', 'NUMBER OF CYCLIST KILLED',
       'NUMBER OF MOTORIST INJURED', 'NUMBER OF MOTORIST KILLED',
       'NUMBER OF PEDESTRIANS INJURED', 'NUMBER OF PEDESTRIANS KILLED',
       'ON STREET NAME', 'TIME', 'TOTAL INJURED', 'TOTAL KILLED',
       'VEHICLE TYPE CODE 1', 'VEHICLE TYPE CODE 2', 'VEHICLE TYPE CODE 3',
       'VEHICLE TYPE CODE 4', 'VEHICLE TYPE CODE 5', 'ZIP CODE'],
      dtype='object')
```

We have the following columns

1. **Borough**: The borough in which the accident occured
2. **COLLISION_ID**: A unique identifier for this collision
3. **CONTRIBUTING FACTOR VEHICLE (1, 2, 3, 4, 5)**: Reasons for the accident
4. **CROSS STREET NAME**: Nearest cross street to the place of accidents
5. **DATE**: Date of the accident
6. **TIME**: Time of accident
7. **DATETIME**: The column we previously created with the combination of date and time
8. **LATITUDE**: Latitude of the accident
9. **LONGITUDE**: Longitude of the accident
10. **NUMBER OF (CYCLIST, MOTORIST, PEDESTRIANS) INJURED**: Category wise injury
11. **NUMBER OF (CYCLIST, MOTORIST, PEDESTRIANS) KILLED**: Category wise death
12. **ON STREET NAME**: Street where the accident occured

13. **TOTAL INJURED**: Total injury from the accident
14. **TOTAL KILLED**: Total casualties in the accident
15. **VEHICLE TYPE CODE (1, 2, 3, 4, 5)**: Types of vehicles involved in the accident
16. **ZIP CODE**: zip code of the accident location

Let's go ahead and answer each of the client's questions.

1.4 Answering the client's questions

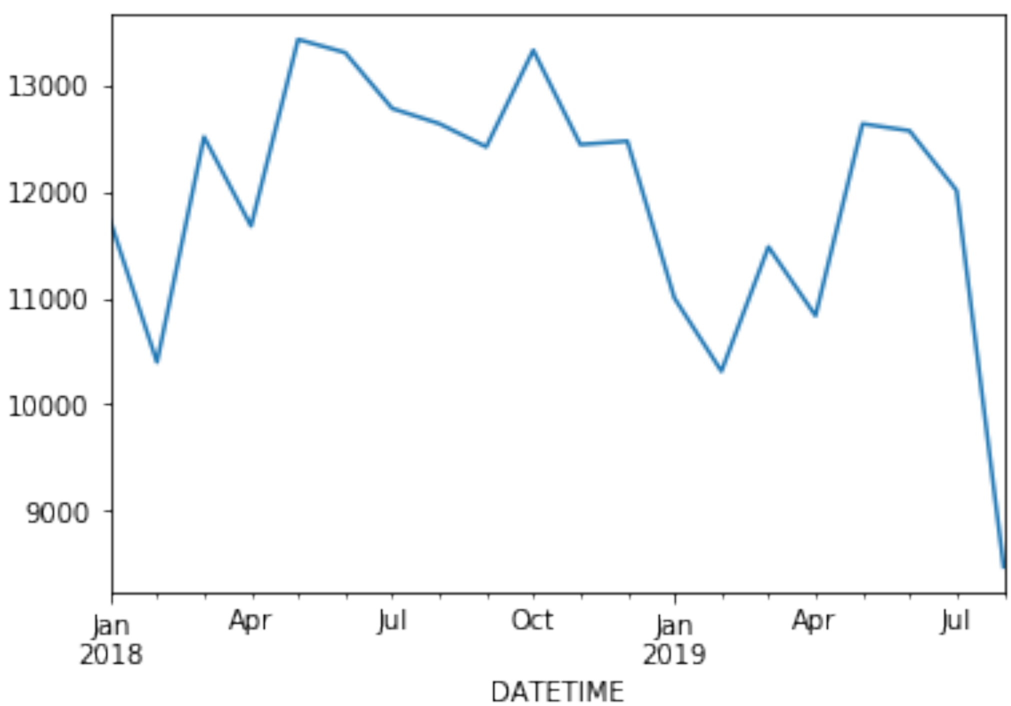
1.4.1 Part 1: Accidents over time

Group the available data on a monthly basis and generate a line plot of accidents over time. Has the number of accidents increased over the past year and a half?

Answer. One possible solution is given below:

[25]:
```
# Check whether the number of accidents has increased over time
monthly_accidents = df.groupby(df['DATETIME'].dt.to_period('M')).size()
monthly_accidents.plot.line()
```

[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1a65a67cc0>



The line graph we plotted clearly shows that there is no obvious uptrend in accidents over time.

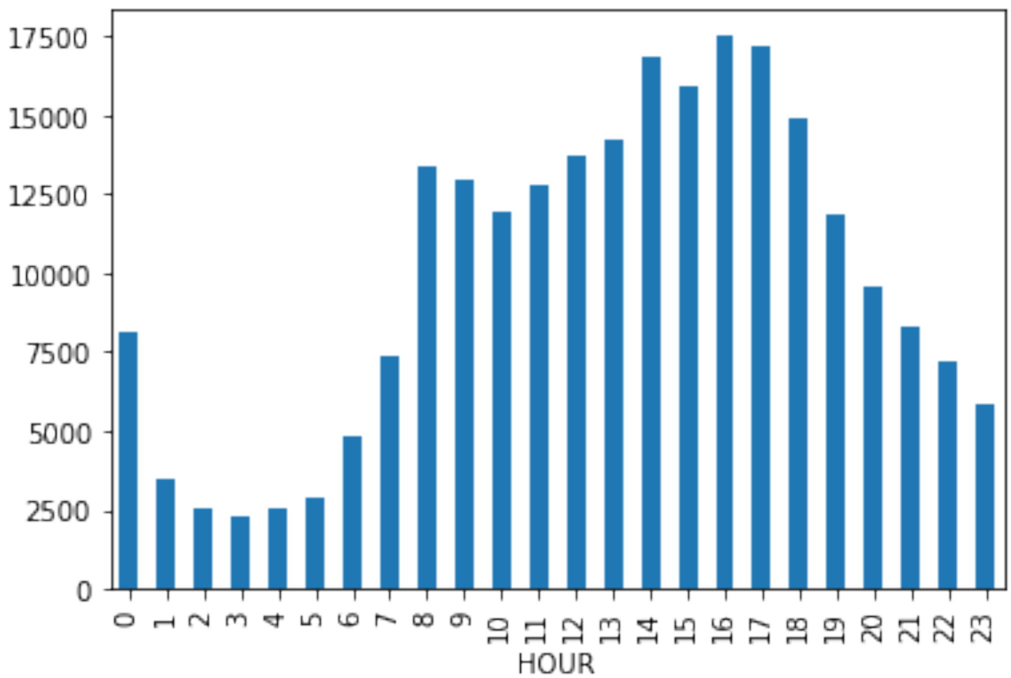
1.4.2 Part 2: Accident hotspots in a day

How does the number of accidents vary throughout a single day? Create a new column `HOUR` based on the data from the `DATETIME` column, then plot a bar graph of the distribution per hour throughout the day.

Answer. One possible solution is given below:

[26]:
```
# Find out how the number of accidents varies across hours.
# Are there more accidents in the night? Or during peak hours?
# Create a new hour column and group by
df['HOUR'] = df['DATETIME'].dt.hour
hourly_accidents = df.groupby('HOUR').size()
hourly_accidents.plot.bar()
```

[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1a64ed4eb8>



From this, it is clear that more accidents occur in the afternoon (2 - 6 PM) than at other times of day.

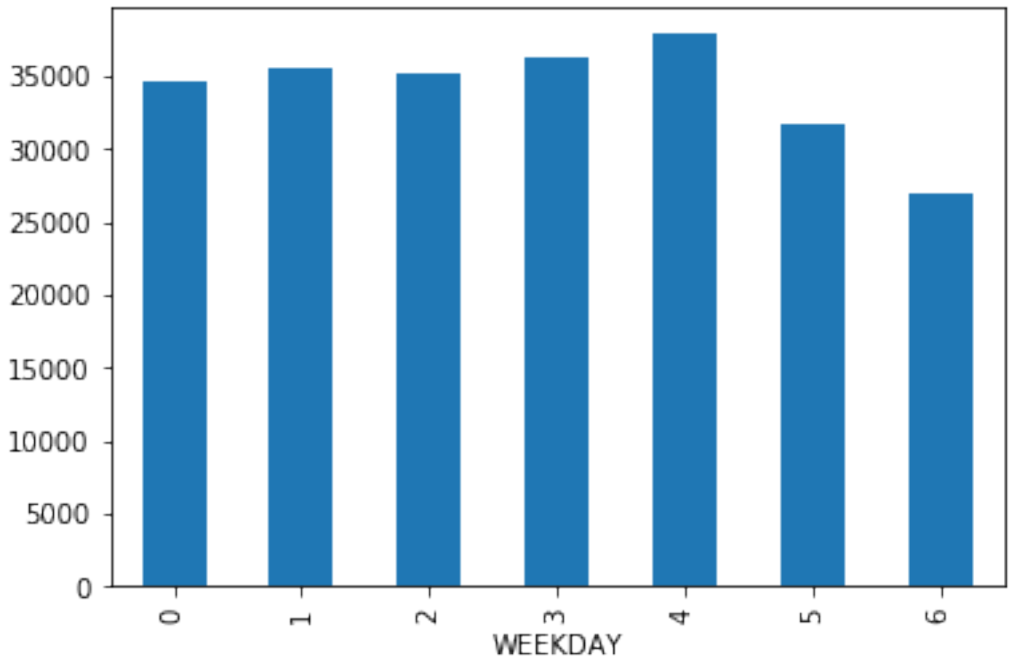
1.4.3 Part 3: Accidents by weekday

How does the number of accidents vary throughout a single week? Plot a bar graph based on the accidents count by day of the week.

Answer. One possible solution is given below:

[27]:
```
df['WEEKDAY'] = df['DATETIME'].dt.weekday
weekday_accidents = df.groupby('WEEKDAY').size()
weekday_accidents.plot.bar()
```

[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1a65949160>



There are relatively fewer accidents on weekends than weekdays.

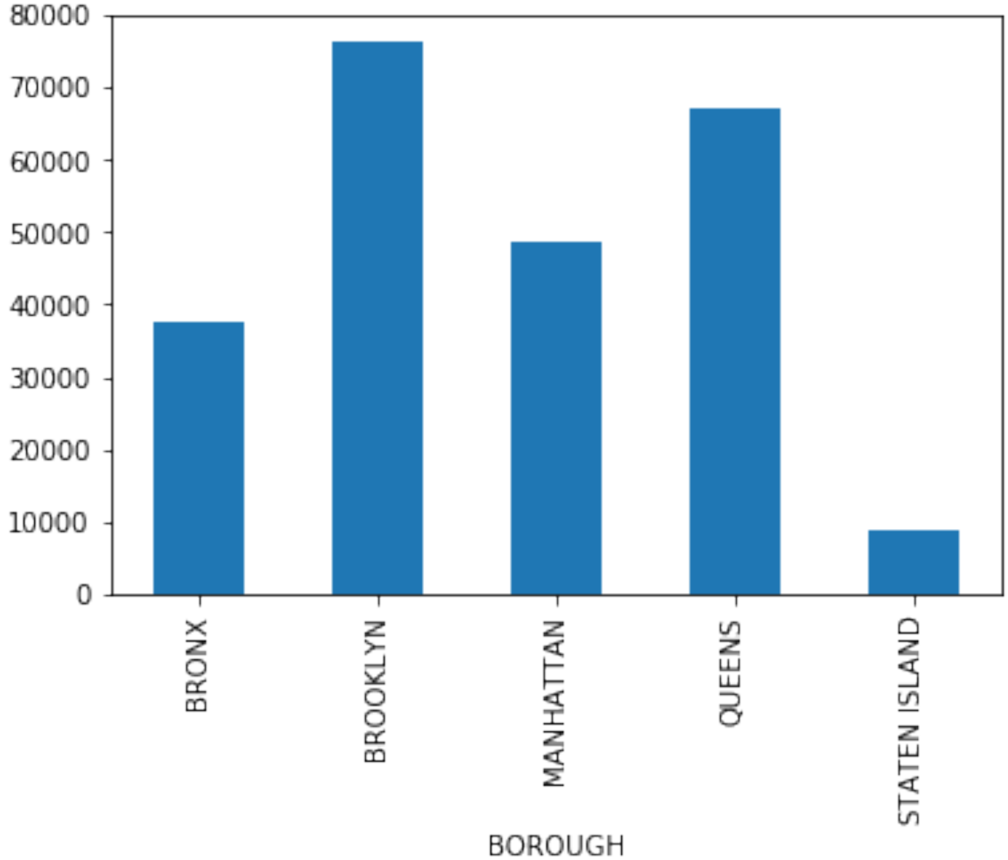
1.4.4 Part 4: Borough analysis

Plot a bar graph of the total number of accidents in each borough, as well as one of the accidents per square kilometer per borough. What can you conclude?

Answer. One possible solution is given below:

[40]:
```
boroughs = df.groupby('BOROUGH').size()
boroughs.plot.bar()
```

[40]: <matplotlib.axes._subplots.AxesSubplot at 0x1a65b8c5f8>



We can see that Brooklyn and Queens have a very high number of accidents relative to the other three boroughs. But how about per square kilometer?

[29]:
```
# Update keys in borough data
print(borough_data.keys())
print(df['BOROUGH'].unique())

# Since there are differences in the text used in the data and Wikipedia data,␣
↪let's update it
borough_data['bronx'] = borough_data.pop('the bronx')
```

```
dict_keys(['the bronx', 'brooklyn', 'manhattan', 'queens', 'staten island'])
['BRONX' 'BROOKLYN' 'QUEENS' 'MANHATTAN' 'STATEN ISLAND']
```

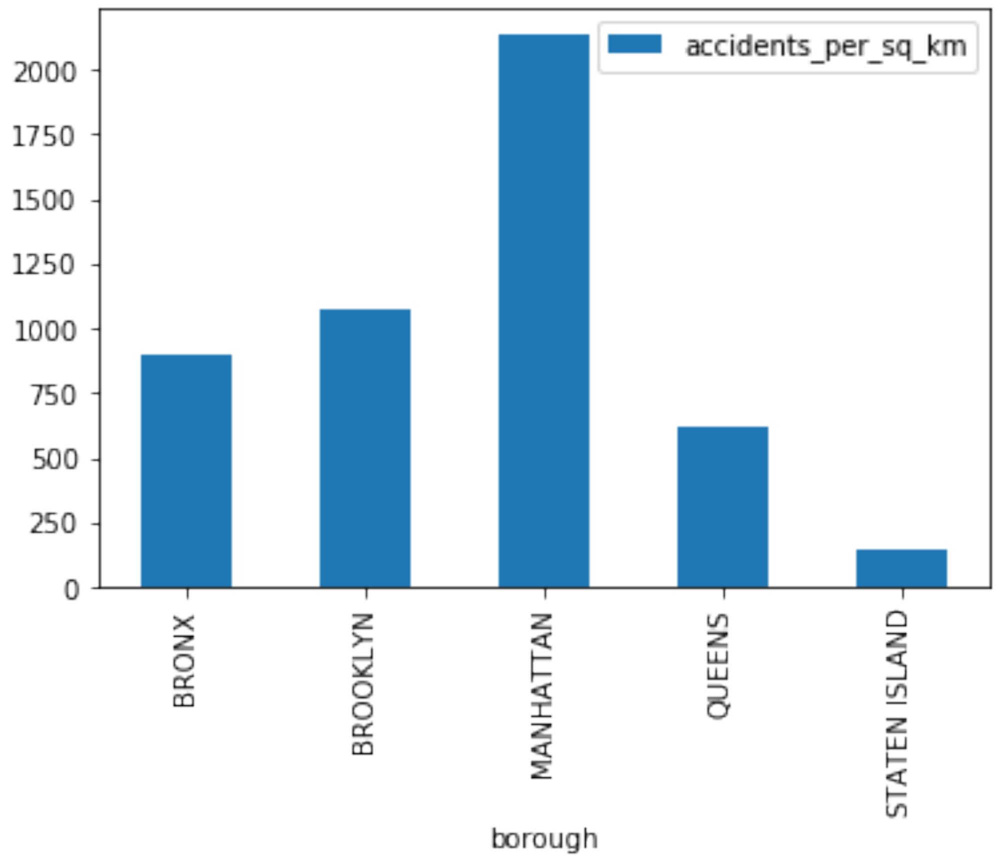
We have now got the keys to match in the dictionary and the dataframe. The difference in case can be handled by making the mapping action case-insensitive. This can be done by either converting the dictionary keys to uppercase, or the dataframe data to lowercase.

Let's do that and plot `accidents_per_sq_km`, which is the accidents-to-area ratio:

[30]:
```
borough_frame = pd.DataFrame(boroughs)
borough_frame.columns = ['count']
borough_frame['borough'] = borough_frame.index

borough_frame['accidents_per_sq_km'] = borough_frame.apply(lambda x: x['count']␣
↪/ borough_data[x['borough'].lower()]['area'], axis=1)
borough_frame.plot.bar(x='borough', y='accidents_per_sq_km')
```

[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1a658c1b00>



When looking at the `accidents_per_sq_km` parameter, Manhattan tops the list by a wide margin. This clearly shows that even though Brooklyn and Queens have more total accidents, Manhattan has a much higher concentration of accidents.

1.4.5 Part 5: Borough hourly analysis

Which hours have the most accidents for each borough? Plot a bar graph for each borough showing the number of accidents for each hour of the day.

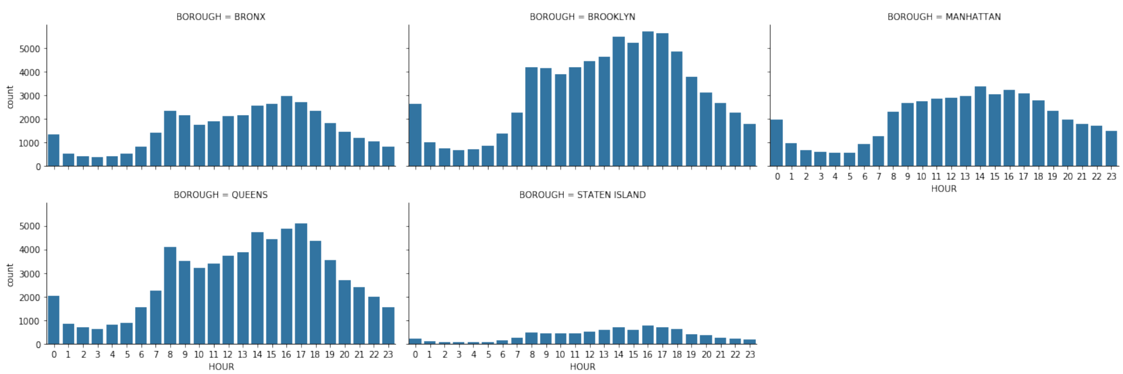
Answer. One possible solution is given below:

```
[31]: df1 = pd.DataFrame({'count': df.groupby(['BOROUGH', 'HOUR']).size()})
df1 = df1.reset_index()

chart = sns.FacetGrid(df1, col='BOROUGH', margin_titles=True, col_wrap=3,␣
 ↪aspect=2)
chart.map(sns.barplot, 'HOUR', 'count',)
```

```
/Users/haris.jaliawala/anaconda3/lib/python3.7/site-
packages/seaborn/axisgrid.py:715: UserWarning: Using the barplot function
without specifying `order` is likely to produce an incorrect plot.
  warnings.warn(warning)
```

```
[31]: <seaborn.axisgrid.FacetGrid at 0x1a65756438>
```



Is the number of accidents higher at different times in different boroughs? Should we concentrate at different times for each borough?

We can see that in all the boroughs the accident count is highest from approximately 2 - 6PM. But in Manhattan and the Bronx, you can see that there is not as much of a relative increase during these hours as in Brooklyn or Queens. Additionally, Staten Island has the lowest overall number of accidents.

1.4.6 Part 6: Cause of accidents

What factors cause the most accidents?

Answer. Since the data is present in multiple columns (`CONTRIBUTING FACTOR VEHICLE 1 - 5`), we need to add up all the values to get the right picture:

```
[32]: combined_df = pd.DataFrame(columns=['factor', 'count'])
columns = ['CONTRIBUTING FACTOR VEHICLE 1', 'CONTRIBUTING FACTOR VEHICLE 2',
           'CONTRIBUTING FACTOR VEHICLE 3', 'CONTRIBUTING FACTOR VEHICLE 4',
          'CONTRIBUTING FACTOR VEHICLE 5']
for column in columns:
    column_df = pd.DataFrame({'count': df.groupby(column).size()})
```

```python
    column_df = column_df.reset_index()
    column_df.columns = ['factor', 'count']
    combined_df = pd.concat([combined_df, column_df], sort=True)

# Now that we have combined the sum of each column into a single dataframe, a␣
↪groupby on that dataframe
# will give you the overall sum of each contributing factor
combined_df.groupby('factor')['count'].sum().sort_values(ascending=False)
```

[32]:
```
factor
                                                        739738
Unspecified                                             240164
Driver Inattention/Distraction                           72203
Failure to Yield Right-of-Way                            20413
Following Too Closely                                    17909
Backing Unsafely                                         15068
Passing Too Closely                                      13378
Passing or Lane Usage Improper                           12953
Other Vehicular                                          11093
Unsafe Lane Changing                                      6986
Turning Improperly                                        6798
Traffic Control Disregarded                               5146
Driver Inexperience                                       4431
Unsafe Speed                                              3530
Reaction to Uninvolved Vehicle                            3190
View Obstructed/Limited                                   2537
Alcohol Involvement                                       2514
Pavement Slippery                                         2146
Oversized Vehicle                                         2023
Pedestrian/Bicyclist/Other Pedestrian Error/Confusion     1895
Aggressive Driving/Road Rage                              1153
Passenger Distraction                                     1053
Brakes Defective                                           842
Fell Asleep                                                600
Outside Car Distraction                                    572
Obstruction/Debris                                         457
Glare                                                      447
Failure to Keep Right                                      378
Steering Failure                                           320
Pavement Defective                                         315
Illnes                                                     256
Driverless/Runaway Vehicle                                 247
Tire Failure/Inadequate                                    220
Fatigued/Drowsy                                            218
Lost Consciousness                                         206
Lane Marking Improper/Inadequate                           194
Animals Action                                             189
```

```
Accelerator Defective                                      143
Traffic Control Device Improper/Non-Working                141
Drugs (illegal)                                            118
Cell Phone (hand-Held)                                      94
Physical Disability                                         61
Other Lighting Defects                                      40
Other Electronic Device                                     31
Tow Hitch Defective                                         28
Tinted Windows                                              27
Vehicle Vandalism                                           24
Prescription Medication                                     20
Using On Board Navigation Device                            19
Headlights Defective                                        17
Eating or Drinking                                          16
Texting                                                     13
Cell Phone (hands-free)                                     12
Shoulders Defective/Improper                                 9
Windshield Inadequate                                        8
Listening/Using Headphones                                   7
Name: count, dtype: int64
```

The top 5 causes are:

1. Driver Inattention/Distraction
2. Failure to Yield Right-of-Way
3. Following Too Closely
4. Backing Unsafely
5. Passing Too Close

1.4.7 Part 7: Boroughs and vehicle types

Which vehicle types are most involved in accidents per borough?

Answer. One possible solution is shown below:

[33]:
```python
combined_df = pd.DataFrame(columns=['vehicle', 'borough', 'count'])
columns = ['VEHICLE TYPE CODE 1', 'VEHICLE TYPE CODE 2', 'VEHICLE TYPE CODE 3',
           'VEHICLE TYPE CODE 4', 'VEHICLE TYPE CODE 5',]
for column in columns:
    column_df = pd.DataFrame({'count': df.groupby([column, 'BOROUGH']).size()})
    column_df = column_df.reset_index()
    column_df.columns = ['vehicle', 'borough', 'count']
    combined_df = pd.concat([combined_df, column_df], sort=True)

# We can see that vehicle type is empty for all rows, unless all 5 vehicle type␣
↪columns are filled.
# Since that is not useful for us to analyze, let's drop those values before␣
↪continuing.
```

```python
combined_df = combined_df[combined_df['vehicle'] != '']

# Now that we have combined the sum of each column into a single dataframe, a␣
↪groupby on that dataframe
# will give you the overall sum of each contributing factor

combined_df = combined_df.groupby(['vehicle', 'borough'])['count'].sum()
combined_df = combined_df.reset_index()
combined_df.columns = ['vehicle', 'borough', 'count']
combined_df = combined_df.sort_values(['count'], ascending=False)
combined_df
```

[33]:

	vehicle	borough	count
860	Sedan	BROOKLYN	51983
862	Sedan	QUEENS	45987
884	Station Wagon/Sport Utility Vehicle	BROOKLYN	40895
886	Station Wagon/Sport Utility Vehicle	QUEENS	40500
859	Sedan	BRONX	25714
861	Sedan	MANHATTAN	25614
885	Station Wagon/Sport Utility Vehicle	MANHATTAN	19730
883	Station Wagon/Sport Utility Vehicle	BRONX	19185
687	PASSENGER VEHICLE	BROOKLYN	13480
689	PASSENGER VEHICLE	QUEENS	11608
973	Taxi	MANHATTAN	10509
822	SPORT UTILITY / STATION WAGON	BROOKLYN	10124
824	SPORT UTILITY / STATION WAGON	QUEENS	10049
688	PASSENGER VEHICLE	MANHATTAN	6899
863	Sedan	STATEN ISLAND	6584
686	PASSENGER VEHICLE	BRONX	6421
823	SPORT UTILITY / STATION WAGON	MANHATTAN	5256
821	SPORT UTILITY / STATION WAGON	BRONX	4643
887	Station Wagon/Sport Utility Vehicle	STATEN ISLAND	4353
148	Box Truck	MANHATTAN	3669
899	TAXI	MANHATTAN	3536
737	Pick-up Truck	BROOKLYN	3310
739	Pick-up Truck	QUEENS	3222
738	Pick-up Truck	MANHATTAN	2473
147	Box Truck	BROOKLYN	2446
139	Bike	BROOKLYN	2378
972	Taxi	BROOKLYN	2311
690	PASSENGER VEHICLE	STATEN ISLAND	2215
140	Bike	MANHATTAN	2104
161	Bus	MANHATTAN	1897
…	…	…	…
681	P/SH	QUEENS	1
682	PAS	BRONX	1

```
683                              PAS      MANHATTAN      1
648                               ND          BRONX      1
647                            NAVIG      MANHATTAN      1
643           Multi-Wheeled Vehicle          BRONX      1
601                            MTA b      MANHATTAN      1
573                             MINI       BROOKLYN      1
574                             MINI  STATEN ISLAND      1
575                            MINIV      MANHATTAN      1
577                             MOPD       BROOKLYN      1
580                            MOPED      MANHATTAN      1
582                            MOPED  STATEN ISLAND      1
583                            MOPET       BROOKLYN      1
586                            MOTOR      MANHATTAN      1
597                            MTA B          BRONX      1
599                            MTA B      MANHATTAN      1
602                           MTRIZ          BRONX      1
642                              Mta          BRONX      1
603                             Mail      MANHATTAN      1
604                            Marke      MANHATTAN      1
605                            Mecha       BROOKLYN      1
606                             Mini          BRONX      1
607                             Mini         QUEENS      1
614                        Minicycle      MANHATTAN      1
620                            Mopen       BROOKLYN      1
623                            Motor         QUEENS      1
624                       Motor Home       BROOKLYN      1
636                  Motorized Home      MANHATTAN      1
1369                          ï¿½MBU       BROOKLYN      1

[1370 rows x 3 columns]
```

We can see that `Sedan` and `Station Wagon/Sport Utility Vehicle` are clear winners for causing the highest number of accidents, and that this does not differ across boroughs.

1.4.8 Part 8: Death counts by vehicle type

Calculate the number of deaths by vehicle and plot a bar chart for the top 5 vehicles. Which vehicles are most often involved in deaths, and by how much more than the others?

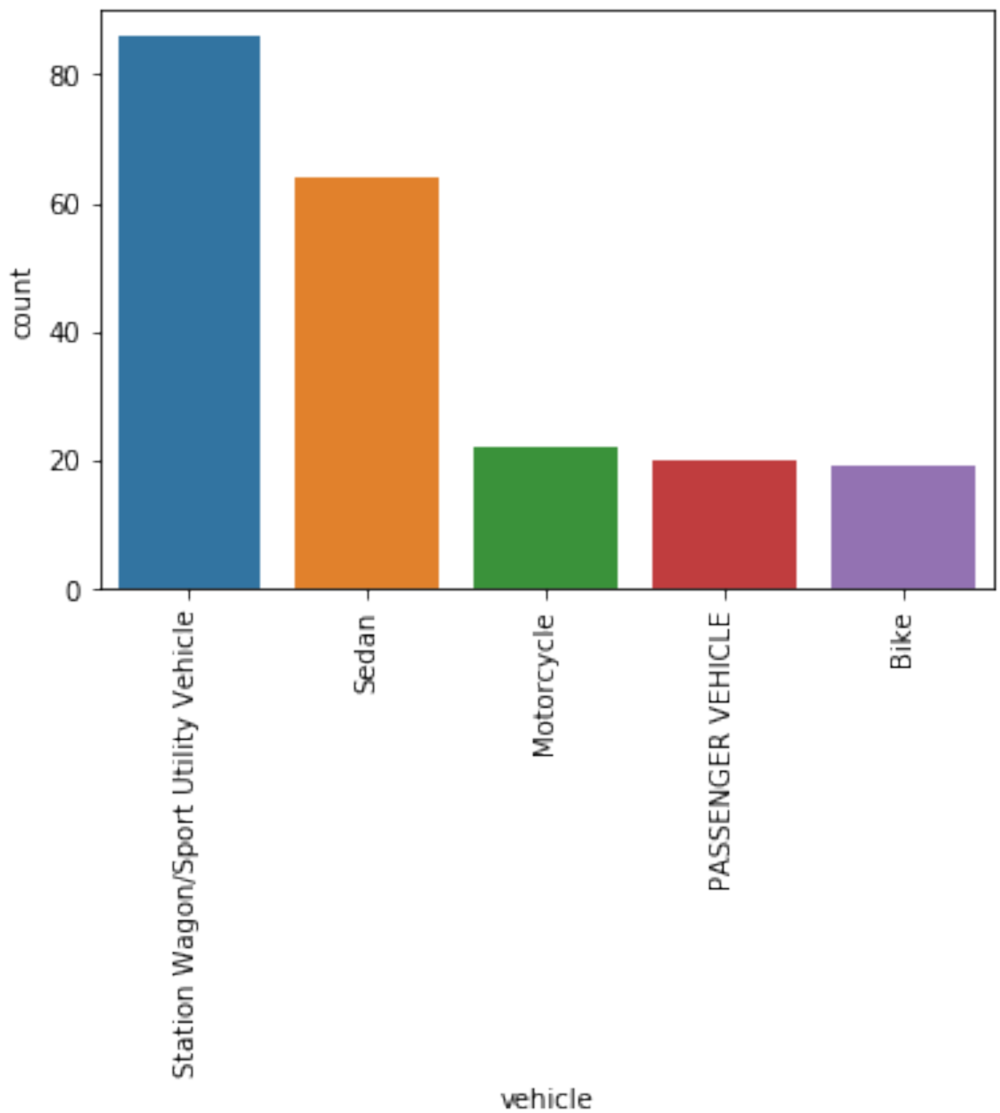
Answer. One possible solution is given below:

[34]:
```python
deaths_df = pd.DataFrame(df[df['TOTAL KILLED'] > 0])
columns = ['VEHICLE TYPE CODE 1', 'VEHICLE TYPE CODE 2', 'VEHICLE TYPE CODE 3',
           'VEHICLE TYPE CODE 4', 'VEHICLE TYPE CODE 5',]
deaths_df['involved_vehicles'] = deaths_df.apply(lambda x: set([x[column] for␣
 ↪column in columns if x[column]]), axis=1)
deaths_count = {}
```

```
for index, row in deaths_df.iterrows():
    for vehicle_type in row['involved_vehicles']:
        count = deaths_count.get(vehicle_type, 0)
        count += 1
        deaths_count[vehicle_type] = count
deaths_count = sorted(deaths_count.items(), key=lambda x: x[1], reverse=True)
```

[35]:
```
deaths_count_df = pd.DataFrame(deaths_count)
deaths_count_df.columns = ['vehicle', 'count']
barplot = sns.barplot(data=deaths_count_df[:5], x='vehicle', y='count')
barplot.set_xticklabels(barplot.get_xticklabels(), rotation=90)
```

[35]:
```
[Text(0, 0, 'Station Wagon/Sport Utility Vehicle'),
 Text(0, 0, 'Sedan'),
 Text(0, 0, 'Motorcycle'),
 Text(0, 0, 'PASSENGER VEHICLE'),
 Text(0, 0, 'Bike')]
```



It appears that `Station Wagon/Sport Utility Vehicle` and `Sedan` cause the most deaths. The former causes 4 times as many deaths as the other vehicles, and the latter causes 3 times as many deaths as the other vehicles.