Before you turn in the homework, make sure everything runs as expected. To do so, select **Kernel→Restart & Run All** in the toolbar above. Remember to submit both on **DataHub** and **Gradescope**.

Please fill in your name and include a list of your collaborators below.

In [1]:
```
NAME = "Matthew Brennan"
COLLABORATORS = "Connor McCormick"
```

Project 2: NYC Taxi Rides

Part 3: NYC Accidents Data

In the real world, data isn't always nicely bundled in one file; data can be sourced from many places with many formats. Now we will use NYC accident data to try to improve our set of features.

In this part of the project, you'll do some EDA over the combined data set. We'll do a lot of the coding work for you, but there will be a few coding subtasks for you to complete on your own, as well as many results to interpret.

Note

If your kernel dies unexpectedly, make sure you have shutdown all other notebooks. Each notebook uses valuable memory which we will need for this part of the project.

Imports

Let us start by loading the Python libraries and custom tools we will use in this part.

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import zipfile
import os
from pathlib import Path

sns.set(style="whitegrid", palette="muted")

plt.rcParams['figure.figsize'] = (12, 9)
plt.rcParams['font.size'] = 12

%matplotlib inline
```

Downloading the Data

We will use the `fetch_and_cache` utility to download the dataset.

In [3]:
```
# Download and cache urls and get the file objects.
from utils import fetch_and_cache
data_url = 'https://github.com/DS-100/fa18/raw/gh-pages/assets/datasets/collisions.zip'
file_name = 'collisions.zip'
dest_path = fetch_and_cache(data_url=data_url, file=file_name)

print(f'Located at {dest_path}')
```

```
Using version already downloaded: Mon Dec  3 22:27:27 2018
MD5 hash of file: a445b925d24f319cb60bd3ace6e4172b
Located at data/collisions.zip
```

We will store the taxi data locally before loading it.

In [4]:
```
collisions_zip = zipfile.ZipFile(dest_path, 'r')

#Extract zip files
collisions_dir = Path('data/collisions')
collisions_zip.extractall(collisions_dir)
```

Loading and Formatting Data

The following code loads the collisions data into a Pandas DataFrame.

In [5]:
```
# Run this cell to load the collisions data.
skiprows = None
collisions = pd.read_csv(collisions_dir/'collisions_2016.csv', index_col='UNIQUE KEY',
                         parse_dates={'DATETIME':["DATE","TIME"]}, skiprows=skiprows)
collisions['TIME'] = pd.to_datetime(collisions['DATETIME']).dt.hour
collisions['DATE'] = pd.to_datetime(collisions['DATETIME']).dt.date
collisions = collisions.dropna(subset=['LATITUDE', 'LONGITUDE'])
collisions = collisions[collisions['LATITUDE'] <= 40.85]
collisions = collisions[collisions['LATITUDE'] >= 40.63]
collisions = collisions[collisions['LONGITUDE'] <= -73.65]
collisions = collisions[collisions['LONGITUDE'] >= -74.03]
collisions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 116691 entries, 3589202 to 3363795
Data columns (total 30 columns):
DATETIME                         116691 non-null datetime64[ns]
Unnamed: 0                       116691 non-null int64
BOROUGH                          100532 non-null object
ZIP CODE                         100513 non-null float64
LATITUDE                         116691 non-null float64
LONGITUDE                        116691 non-null float64
LOCATION                         116691 non-null object
ON STREET NAME                   95914 non-null object
CROSS STREET NAME                95757 non-null object
OFF STREET NAME                  61545 non-null object
NUMBER OF PERSONS INJURED        116691 non-null int64
NUMBER OF PERSONS KILLED         116691 non-null int64
NUMBER OF PEDESTRIANS INJURED    116691 non-null int64
NUMBER OF PEDESTRIANS KILLED     116691 non-null int64
NUMBER OF CYCLIST INJURED        116691 non-null int64
NUMBER OF CYCLIST KILLED         116691 non-null int64
NUMBER OF MOTORIST INJURED       116691 non-null int64
NUMBER OF MOTORIST KILLED        116691 non-null int64
CONTRIBUTING FACTOR VEHICLE 1    115162 non-null object
CONTRIBUTING FACTOR VEHICLE 2    101016 non-null object
CONTRIBUTING FACTOR VEHICLE 3    7772 non-null object
CONTRIBUTING FACTOR VEHICLE 4    1829 non-null object
CONTRIBUTING FACTOR VEHICLE 5    434 non-null object
VEHICLE TYPE CODE 1              115181 non-null object
VEHICLE TYPE CODE 2              92815 non-null object
VEHICLE TYPE CODE 3              7260 non-null object
VEHICLE TYPE CODE 4              1692 non-null object
```

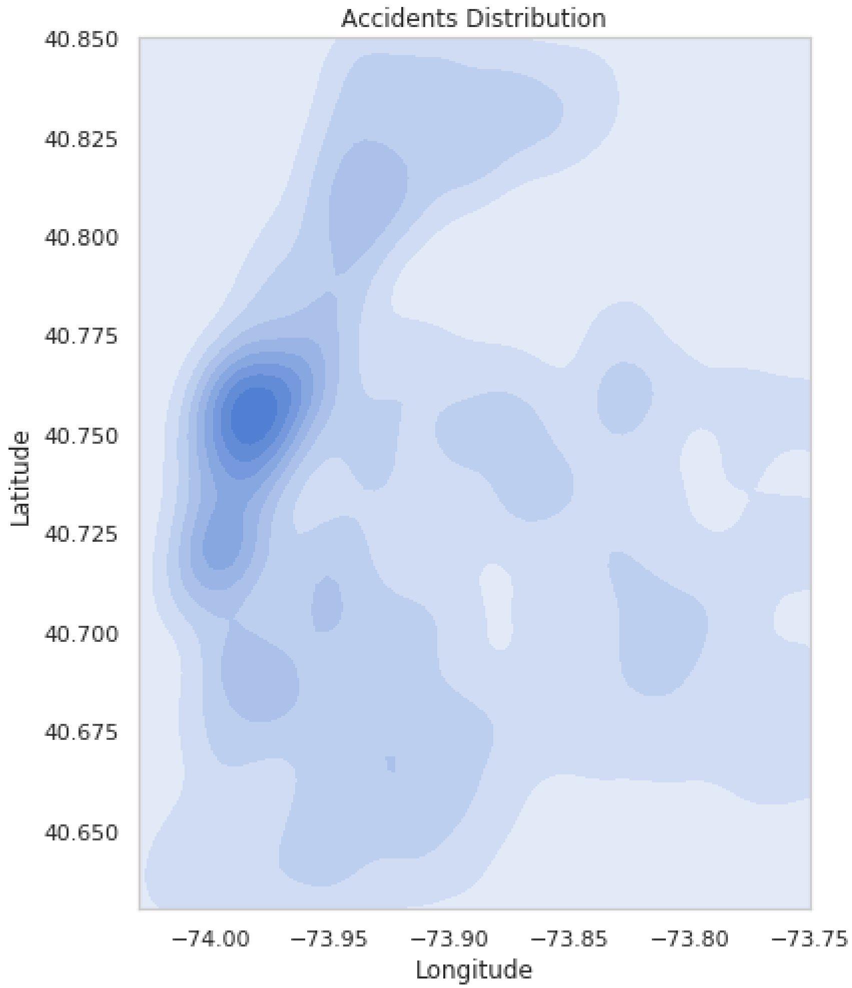
```
VEHICLE TYPE CODE 5                 403 non-null object
TIME                                116691 non-null int64
DATE                                116691 non-null object
dtypes: datetime64[ns](1), float64(3), int64(10), object(16)
memory usage: 27.6+ MB
```

1: EDA of Accidents

Let's start by plotting the latitude and longitude where accidents occur. This may give us some insight on taxi ride durations. We sample N times (given) from the collisions dataset and create a 2D KDE plot of the longitude and latitude. We make sure to set the x and y limits according to the boundaries of New York, given below.

Here is a map of Manhattan (https://www.google.com/maps/place/Manhattan,+New+York,+NY/@40.7590402,-74.0394431,12z/data=!3m1!4b1!4m5!3m4!1s0x89c2588f(73.9712488) for your convenience.

In [6]:
```
# Plot lat/lon of accidents, will take a few seconds
N = 20000
city_long_border = (-74.03, -73.75)
city_lat_border = (40.63, 40.85)

sample = collisions.sample(N)
plt.figure(figsize=(6,8))
sns.kdeplot(sample["LONGITUDE"], sample["LATITUDE"], shade=True)
plt.xlim(city_long_border)
plt.ylim(city_lat_border)
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.title("Accidents Distribution")
plt.show();
```



Question 1a

What can you say about the location density of NYC collisions based on the plot above?

Hint: Here is a [page (https://www.google.com/maps/place/Manhattan,+New+York,+NY/@40.7590402,-74.0394431,12z/data=!3m1!4b1!4m5!3m4!1s0x89c2 73.9712488)](https://www.google.com/maps/place/Manhattan,+New+York,+NY/@40.7590402,-74.0394431,12z/data=!3m1!4b1!4m5!3m4!1s0x89c2) that may be useful, and [another page (https://www.6sqft.com/what-nycs-population-looks-like-day-vs-night/)](https://www.6sqft.com/what-nycs-population-looks-like-day-vs-night/) that may be useful.

In [7]:
```
q1a_answer = r"""
Accident collisions are more likely to occur in more denseley populated areas. We were able to draw thi

"""

# YOUR CODE HERE
#raise NotImplementedError()

print(q1a_answer)
```

```
Accident collisions are more likely to occur in more denseley populated areas. We were able to draw th
is conclusion from the immense amount of accidents that occur in Manhattan at the 40.750 to 40.775 lat
itude markers with -74.00 longitude which is identifiable by the darker blue color of the density plo
t. From the 6sqft link that was given we can infer that the reason for the plethora of collisions in M
anhattan stems from the immense amount of daily travelers and workers that visit Manhattan during the
day but do not actually live there. Despite the fact that the large population, narrow streets, and ov
erall hecticness of Manhattan stem as a central problem, we also must take into account the vast incre
ase in daytime population that acuumulates from out of towners or suburban folk of Brooklyn and New Je
rsey.
```

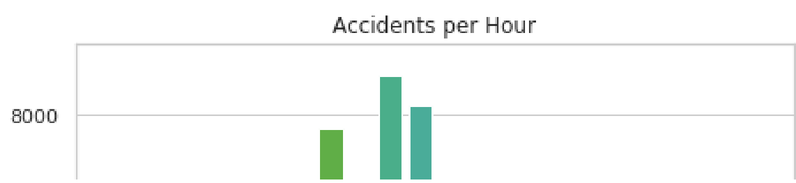
We see that an entry in accidents contains information on number of people injured/killed. Instead of using each of these columns separately, let's combine them into one column called `'SEVERITY'` . Let's also make columns `FATALITY` and `INJURY` , each aggregating the fatalities and injuries respectively.

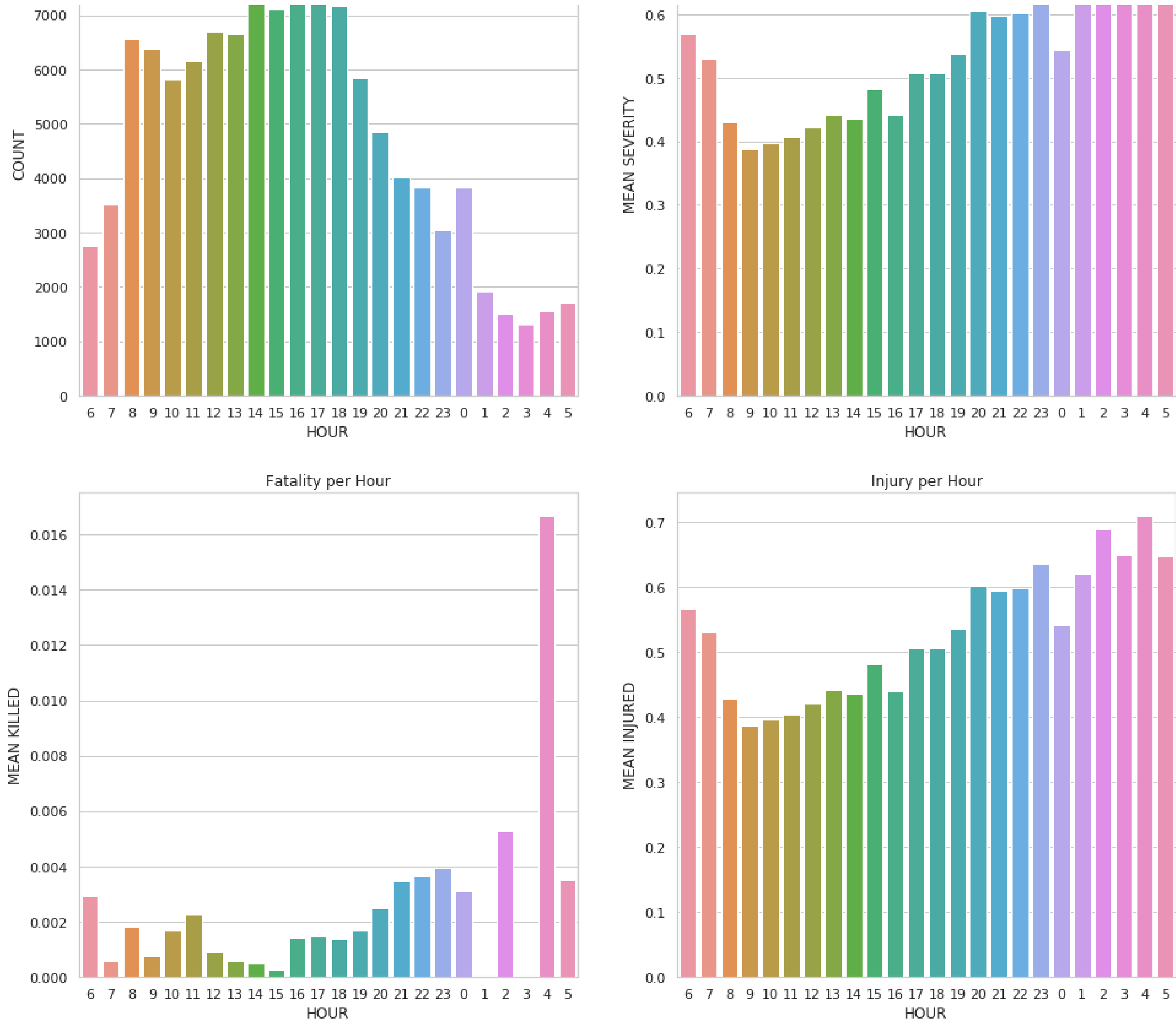
In [8]:
```
collisions['SEVERITY'] = collisions.filter(regex=r'NUMBER OF *').sum(axis=1)
collisions['FATALITY'] = collisions.filter(regex=r'KILLED').sum(axis=1)
collisions['INJURY'] = collisions.filter(regex=r'INJURED').sum(axis=1)
```

Now let's group by time and compare two aggregations: count vs mean. Below we plot the number of collisions and the mean severity of collisions by the hour, i.e. the `TIME` column. We visualize them side by side and set the start of our day to be 6 a.m.

Let's also take a look at the mean number of casualties per hour and the mean number of injuries per hour, plotted below.

In [9]:
```
fig, axes = plt.subplots(2, 2, figsize=(16,16))
order = np.roll(np.arange(24), -6)
ax1 = axes[0,0]
ax2 = axes[0,1]
ax3 = axes[1,0]
ax4 = axes[1,1]

collisions_count = collisions.groupby('TIME').count()
collisions_count = collisions_count.reset_index()
sns.barplot(x='TIME', y='SEVERITY', data=collisions_count, order=order, ax=ax1)
ax1.set_title("Accidents per Hour")
ax1.set_xlabel("HOUR")
ax1.set_ylabel('COUNT')


collisions_mean = collisions.groupby('TIME').mean()
collisions_mean = collisions_mean.reset_index()
sns.barplot(x='TIME', y='SEVERITY', data=collisions_mean, order=order, ax=ax2)
ax2.set_title("Severity of Accidents per Hour")
ax2.set_xlabel("HOUR")
ax2.set_ylabel('MEAN SEVERITY')

fatality_count = collisions.groupby('TIME').mean()
fatality_count = fatality_count.reset_index()
sns.barplot(x='TIME', y='FATALITY', data=fatality_count, order=order, ax=ax3)
ax3.set_title("Fatality per Hour")
ax3.set_xlabel("HOUR")
ax3.set_ylabel('MEAN KILLED')

injury_count = collisions.groupby('TIME').mean()
injury_count = injury_count.reset_index()
sns.barplot(x='TIME', y='INJURY', data=injury_count, order=order, ax=ax4)
ax4.set_title("Injury per Hour")
ax4.set_xlabel("HOUR")
ax4.set_ylabel('MEAN INJURED')

plt.show();
```




COUNT
0
1000
2000
3000
4000
5000
6000
7000
6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 0 1 2 3 4 5
HOUR
MEAN SEVERITY
0.0
0.1
0.2
0.3
0.4
0.5
0.6
6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 0 1 2 3 4 5
HOUR
Fatality per Hour
MEAN KILLED
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 0 1 2 3 4 5
HOUR
Injury per Hour
MEAN INJURED
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 0 1 2 3 4 5
HOUR

Question 1b

Based on the visualizations above, what can you say about each? Make a comparison between the accidents per hour vs the mean severity per hour. What about the number of fatalities per hour vs the number of injuries per hour? Why do we chose to have our hours start at 6 as opposed to 0?

In [10]:
```
q1b_answer = r"""
The above visualizations provide interesting dynamics about the likelihood of accidents at various hours

"""

# YOUR CODE HERE
#raise NotImplementedError()

print(q1b_answer)
```

```
The above visualizations provide interesting dynamics about the likelihood of accidents at various hou
rs and the corresponding severity of these accidents. Initially we see that the number of accidents pe
r hour and the severity of those accidents tend to be inverseley correlated meaning that at a time whe
re accidents are more common they also tend to be less severe and vice versa. One possible explanation
for this stem from the higher prevalance of cars during the day which would lead to higher amounts of
accidents, but more importantly slower speeds. Cars that are heavily occupied by traffic tend to produ
ce more mistakes due to the constant stopping and going, yet these accidents would produce a low mean
severity as accidents increase in intensity with heightened speed. Further, one could argue that durin
g the nighttime less accidents would occur due to a lower amount of cars on the road, but the severity
would increase significantly due to drunk drivers and an overall inability to see with poor lighting.
This argument's logic is profoundly amplified by comparing the fatalities per hour with injury per hou
r as we observe that both fatalities and injuries are in larger proportions late at night. The especia
lly staggering value stems from the hour 4 or hour 4 am in the fatalities chart which depicts that 16%
of fatalities in car collisions occurred at that hour. These charts very effictively highlight lightin
g as a factor by starting with hour 6 as this likely represents when the sun rises so that we can see
that this is a less violent time of the day than those that are just a little bit darker.
```

Let's also check the relationship between location and severity. We provide code to visualize a heat map of collisions, where the x and y coordinate are the location of the collision and the heat color is the severity of the collision. Again, we sample N points to speed up visualization.

In [11]:
```
N = 10000
sample = collisions.sample(N)

# Round / bin the latitude and longitudes
sample['lat_bin'] = np.round(sample['LATITUDE'], 3)
sample['lng_bin'] = np.round(sample['LONGITUDE'], 3)

# Average severity for regions
gby_cols = ['lat_bin', 'lng_bin']

coord_stats = (sample.groupby(gby_cols)
               .agg({'SEVERITY': 'mean'})
               .reset_index())

# Visualize the average severity per region
city_long_border = (-74.03, -73.75)
city_lat_border = (40.63, 40.85)
fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(14, 10))

scatter_trips = ax.scatter(sample['LONGITUDE'].values,
                           sample['LATITUDE'].values,
                           color='grey', s=1, alpha=0.5)

scatter_cmap = ax.scatter(coord_stats['lng_bin'].values,
                          coord_stats['lat_bin'].values,
                          c=coord_stats['SEVERITY'].values,
                          cmap='viridis', s=10, alpha=0.9)

cbar = fig.colorbar(scatter_cmap)
cbar.set_label("Manhattan average severity")
ax.set_xlim(city_long_border)
ax.set_ylim(city_lat_border)
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
plt.title('Heatmap of Manhattan average severity')
plt.axis('off');
```

Question 1c

Do you think the location of the accident has a significant impact on the severity based on the visualization above? Additionally, identify something that could be improved in the plot above and describe how we could improve it.

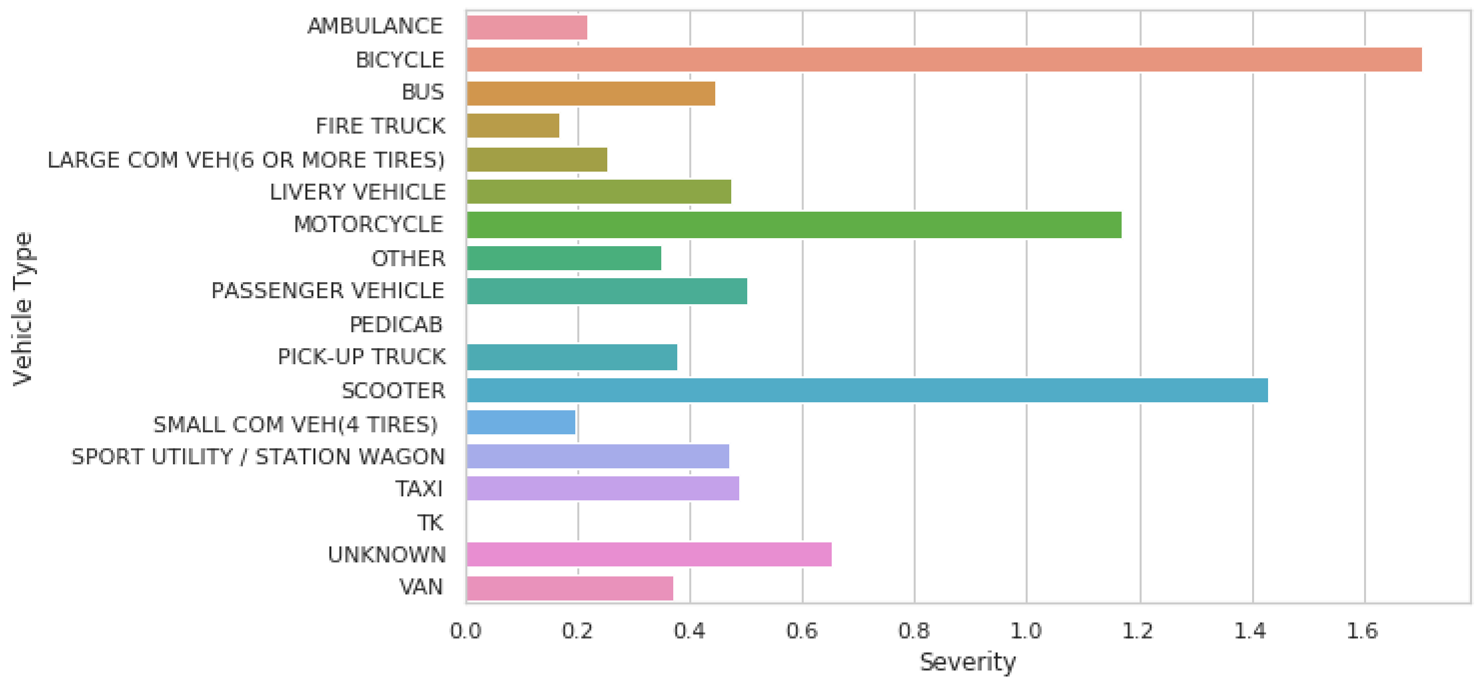
In [12]:
```
q1c_answer = r"""
I do think that the location of the accident has a significant impact on the severity but that this visu
"""

# YOUR CODE HERE
#raise NotImplementedError()

print(q1c_answer)
```

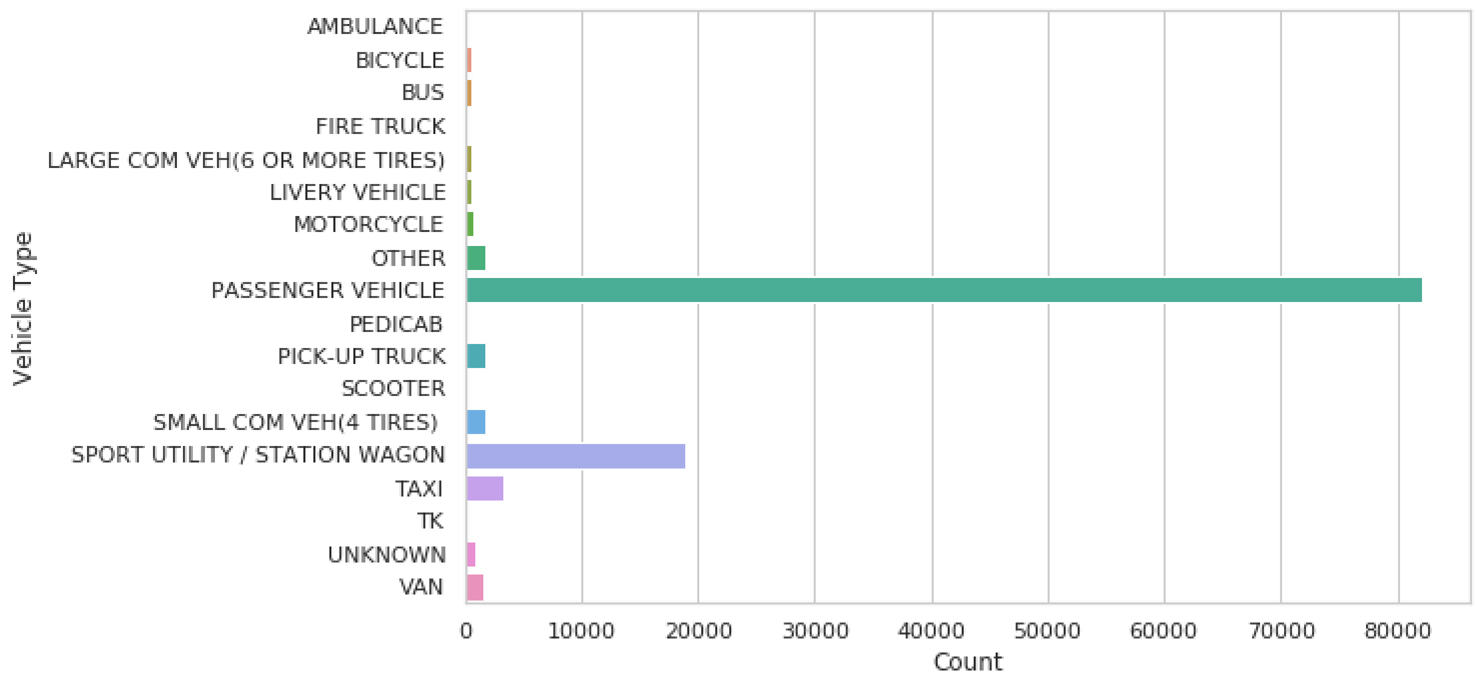
```
I do think that the location of the accident has a significant impact on the severity but that this vi
sualization does not adroitly reflect this factor. One reason for this stems from overplotting but mai
nly the fact that this scale should shrink by a factor of 4 (or more) in order to reflect more non-pur
ple colors. This would allow us to identify more easily the colors which right now are tough to spot b
ecause these dark colors tend to resemble each other, and additionally we see that there are so few no
n-purple colors reflected. Yet, what I can infer from this data is that we are supposed to see that ve
ry severe accidents occur in the surrounding areas of Manhattan which I assume is likely the result of
drunk driving. Further, the overplotting factor is especially a hindrance due to the plotting in Manha
ttan in the specific area we targeted earlier for having an immense amount of accidents.
```

Question 1d

Create a plot to visualize one or more features of the `collisions` table.

In [13]:
```
# YOUR CODE HERE
#creating the necessary table for comparison
types_of_vehicles = collisions['VEHICLE TYPE CODE 1'].unique()
collisionVehicles = collisions.groupby('VEHICLE TYPE CODE 1').agg({
                    'DATETIME':'count', 'SEVERITY': 'mean'}).rename(columns ={'DATETIME': 'COUNT'})
#initializing the plots for creation
f, ((ax1), (ax2)) = plt.subplots(ncols=1, nrows = 2, figsize=(9, 12))

sns.barplot(x = collisionVehicles.SEVERITY, y = collisionVehicles.index, ax = ax1)
sns.barplot(x = collisionVehicles.COUNT, y = collisionVehicles.index, ax = ax2)

ax1.set(ylabel="Vehicle Type", xlabel="Severity")
ax2.set(ylabel="Vehicle Type", xlabel="Count");
#raise NotImplementedError()
```


AMBULANCE
BICYCLE
BUS
FIRE TRUCK
LARGE COM VEH(6 OR MORE TIRES)
LIVERY VEHICLE
MOTORCYCLE
OTHER
PASSENGER VEHICLE
PEDICAB
PICK-UP TRUCK
SCOOTER
SMALL COM VEH(4 TIRES)
SPORT UTILITY / STATION WAGON
TAXI
TK
UNKNOWN
VAN
Vehicle Type
0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6
Severity


AMBULANCE
BICYCLE
BUS
FIRE TRUCK
LARGE COM VEH(6 OR MORE TIRES)
LIVERY VEHICLE
MOTORCYCLE
OTHER
PASSENGER VEHICLE
PEDICAB
PICK-UP TRUCK
SCOOTER
SMALL COM VEH(4 TIRES)
SPORT UTILITY / STATION WAGON
TAXI
TK
UNKNOWN
VAN
Vehicle Type
0 10000 20000 30000 40000 50000 60000 70000 80000
Count

Question 1e

Answer the following questions regarding your plot in 1d.

1. What feature you're visualization
2. Why you chose this feature
3. Why you chose this visualization method

In [14]:
```
q1e_answer = r"""

In question 1d I opted to plot two different features, the severity of the accident grouped by the the v

"""
# YOUR CODE HERE
#raise NotImplementedError()
print(q1e_answer)
```

```
In question 1d I opted to plot two different features, the severity of the accident grouped by the the
vehicle and the total counts of accidents grouped by the vehicle. The main feature of focus however is
the severity and the count plot is strictly there to help aid in overall understanding. Since the plot
s represent categorical data I utilized barplots as these allow me to incorporate easily the various t
ypes of vehicles and I felt as though this material was easiest to understand in a horizontal barplot
format. The severity feature depicts a fascinating means to analyze what variables contribute to the o
verall accident: the area, the hour of the day, and in this case the vehicle type. Therefore, from the
graphs above we can infer that bikes, scooters, and motorcycles present the worst accidents with mean
severities all over 1. This information parallels that of my initial assumption that non-cars were the
most likely to produce very severe accidents whereas larger automobiles such as firetrucks are the lea
st likely. Additionally, by plotting the second plot below with regards to the overall counts we see t
hat these non-automobile collisions are not actually very likely to occur but are definitely serious w
hen they do.
```

2: Combining External Datasets

It seems like accident timing and location may influence the duration of a taxi ride. Let's start to join our NYC Taxi data with our collisions data.

Let's assume that an accident will influence traffic in the surrounding area for around 1 hour. Below, we create two columns, `START` and `END` :

- `START` : contains the recorded time of the accident
- `END` : 1 hours after `START`

Note: We chose 1 hour somewhat arbitrarily, feel free to experiment with other time intervals outside this notebook.

In [15]:
```
collisions['START'] = collisions['DATETIME']
collisions['END'] = collisions['START'] + pd.Timedelta(hours=1)
```

Question 2a

Drop all of the columns besides the following: `DATETIME` , `TIME` , `START` , `END` , `DATE` , `LATITUDE` , `LONGITUDE` , `SEVERITY` . Feel free to experiment with other subsets outside of this notebook.

In [16]:
```
columns = ['DATETIME', 'TIME', 'START', 'END', 'DATE', 'LATITUDE', 'LONGITUDE', 'SEVERITY']
collisions_subset = collisions[columns]
# YOUR CODE HERE
#raise NotImplementedError()
collisions_subset.head(5)
```

Out[16]:

UNIQUE KEY	DATETIME	TIME	START	END	DATE	LATITUDE	LONGITUDE	SEVERITY
3589202	2016-12-29 00:00:00	0	2016-12-29 00:00:00	2016-12-29 01:00:00	2016-12-29	40.844107	-73.897997	0
3587413	2016-12-26 14:30:00	14	2016-12-26 14:30:00	2016-12-26 15:30:00	2016-12-26	40.692347	-73.881778	0
3578151	2016-11-30 22:50:00	22	2016-11-30 22:50:00	2016-11-30 23:50:00	2016-11-30	40.755480	-73.741730	2
3567096	2016-11-23 20:11:00	20	2016-11-23 20:11:00	2016-11-23 21:11:00	2016-11-23	40.771122	-73.869635	0
3565211	2016-11-21 14:11:00	14	2016-11-21 14:11:00	2016-11-21 15:11:00	2016-11-21	40.828918	-73.838403	0

In [17]:
```
assert collisions_subset.shape == (116691, 8)
```

Question 2b

Now, let's merge our `collisions_subset` table with `train_df` . Start by merging with only the date. We will filter by a time window in a later question.

We should be performing a left join, where our `train_df` is the left table. This is because we want to preserve all of the taxi rides in our end result. It happens that an inner join will also work, since both tables contain data on each date.

Note that the resulting `merged` table will have multiple rows for every taxi ride row in the original `train_df` table. For example, `merged` will have 483 rows with `index` equal to 16709, because there were 483 accidents that occurred on the same date as ride #16709.

Because of memory limitation, we will select the third week of 2016 to analyze. Feel free to change to it week 1 or 2 to see if the observation is general.

In [18]:
```
data_file = Path("./", "cleaned_data.hdf")
train_df = pd.read_hdf(data_file, "train")
train_df = train_df.reset_index()
train_df = train_df[['index', 'tpep_pickup_datetime', 'pickup_longitude', 'pickup_latitude', 'duration'
train_df['date'] = train_df['tpep_pickup_datetime'].dt.date
```

In [19]:
```
collisions_subset = collisions_subset[collisions_subset['DATETIME'].dt.weekofyear == 3]
train_df = train_df[train_df['tpep_pickup_datetime'].dt.weekofyear == 3]
```

In [20]:
```
# merge the dataframe here
merged = train_df.merge(collisions_subset, how = 'left', left_on = 'date', right_on = 'DATE')

# YOUR CODE HERE
#raise NotImplementedError()

merged.head()
```

Out[20]:

	index	tpep_pickup_datetime	pickup_longitude	pickup_latitude	duration	date	DATETIME	TIME	START	END	DATE	LATITUDE
0	16709	2016-01-21 22:28:17	-73.997986	40.741215	736.0	2016-01-21	2016-01-21 10:35:00	10	2016-01-21 10:35:00	2016-01-21 11:35:00	2016-01-21	40.701651
1	16709	2016-01-21 22:28:17	-73.997986	40.741215	736.0	2016-01-21	2016-01-21 13:20:00	13	2016-01-21 13:20:00	2016-01-21 14:20:00	2016-01-21	40.704760
2	16709	2016-01-21 22:28:17	-73.997986	40.741215	736.0	2016-01-21	2016-01-21 16:00:00	16	2016-01-21 16:00:00	2016-01-21 17:00:00	2016-01-21	40.732891
3	16709	2016-01-21 22:28:17	-73.997986	40.741215	736.0	2016-01-21	2016-01-21 18:30:00	18	2016-01-21 18:30:00	2016-01-21 19:30:00	2016-01-21	40.714122
4	16709	2016-01-21 22:28:17	-73.997986	40.741215	736.0	2016-01-21	2016-01-21 00:05:00	0	2016-01-21 00:05:00	2016-01-21 01:05:00	2016-01-21	40.700108

In [21]:
```
assert merged.shape == (1528162, 14)
```

Question 2c

Now that our tables are merged, let's use temporal and spatial proximity to condition on the duration of the average length of a taxi ride. Let's operate under the following assumptions.

Accidents only influence the duration of a taxi ride if the following are satisfied:

1) The haversine distance between the the pickup location of the taxi ride and location of the recorded accident is within 5 (km). This is roughly 3.1 miles.

2) The start time of a taxi ride is within a 1 hour interval between the start and end of an accident.

Complete the code below to create an `'accident_close'` column in the `merged` table that indicates if an accident was close or not according to the assumptions above.

In [22]:
```
def haversine(lat1, lng1, lat2, lng2):
    """
    Compute haversine distance
    """
    lat1, lng1, lat2, lng2 = map(np.radians, (lat1, lng1, lat2, lng2))
    average_earth_radius = 6371
    lat = lat2 - lat1
    lng = lng2 - lng1
    d = np.sin(lat * 0.5) ** 2 + np.cos(lat1) * np.cos(lat2) * np.sin(lng * 0.5) ** 2
    h = 2 * average_earth_radius * np.arcsin(np.sqrt(d))
    return h

def manhattan_distance(lat1, lng1, lat2, lng2):
    """
    Compute Manhattan distance
    """
    a = haversine(lat1, lng1, lat1, lng2)
    b = haversine(lat1, lng1, lat2, lng1)
    return a + b
```

In [23]:
```
start_to_accident = haversine(merged['pickup_latitude'].values,
                              merged['pickup_longitude'].values,
                              merged['LATITUDE'].values,
                              merged['LONGITUDE'].values)
merged['start_to_accident'] = start_to_accident

# initialze accident_close column to all 0 first
merged['accident_close'] = 0

# Boolean pd.Series to select the indices for which accident_close should equal 1:
# (1) record's start_to_accident <= 5
# (2) pick up time is between start and end
is_accident_close = [True if ((merged.loc[row, 'start_to_accident'] <= 5)
                     and (merged.loc[row,'tpep_pickup_datetime'] > merged.loc[row, 'START'])
                     and (merged.loc[row,'tpep_pickup_datetime'] < merged.loc[row, 'END']))
                     else False for row in range(len(merged))]
is_accident_close = pd.Series(is_accident_close)

# YOUR CODE HERE
#raise NotImplementedError()

merged.loc[is_accident_close, 'accident_close'] = 1
```

In [24]:
```
assert merged['accident_close'].sum() > 16000
```

In [25]:
```
merged['accident_close'].sum()
```

Out[25]:
```
17731
```

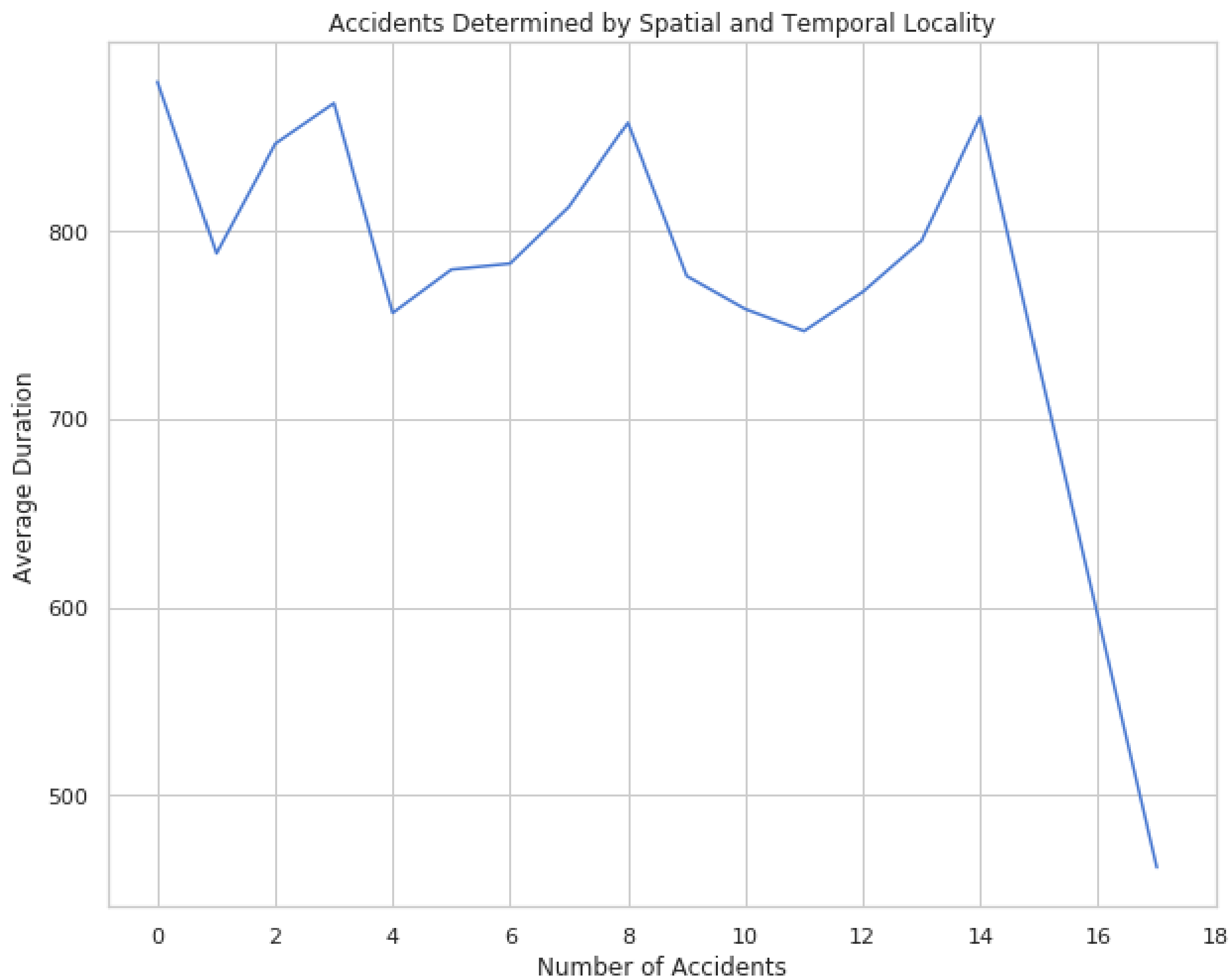
The last step is to aggregate the total number of proximal accidents. We want to count the total number of accidents that were close spatially and temporally and condition on that data.

The code below create a new data frame called `train_accidents` , which is a copy of `train_df` , but with a new column that counts the number of accidents that were close (spatially and temporally) to the pickup location/time.

In [26]:
```
train_df = train_df.set_index('index')
num_accidents = merged.groupby(['index'])['accident_close'].sum().to_frame()
train_accidents = train_df.copy()
train_accidents['num_accidents'] = num_accidents
```

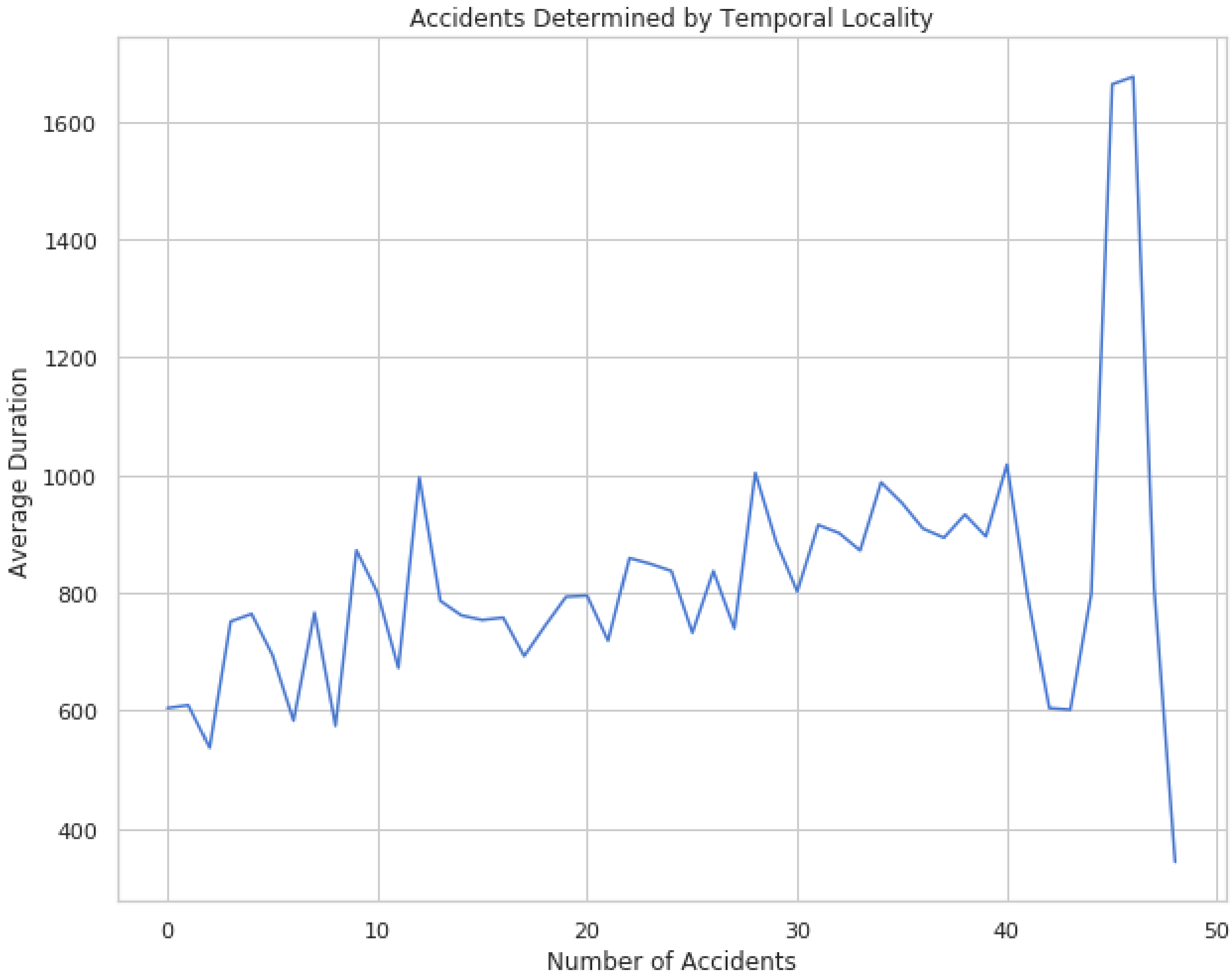
Next, for each value of `num_accidents` , we plot the average `duration` of rides with that number of accidents.

In [27]:
```
plt.figure(figsize=(10,8))
train_accidents.groupby('num_accidents')['duration'].mean().plot(xticks=np.arange(0, 20, 2))
plt.title("Accidents Determined by Spatial and Temporal Locality")
plt.xlabel("Number of Accidents")
plt.ylabel("Average Duration")
plt.show();
```

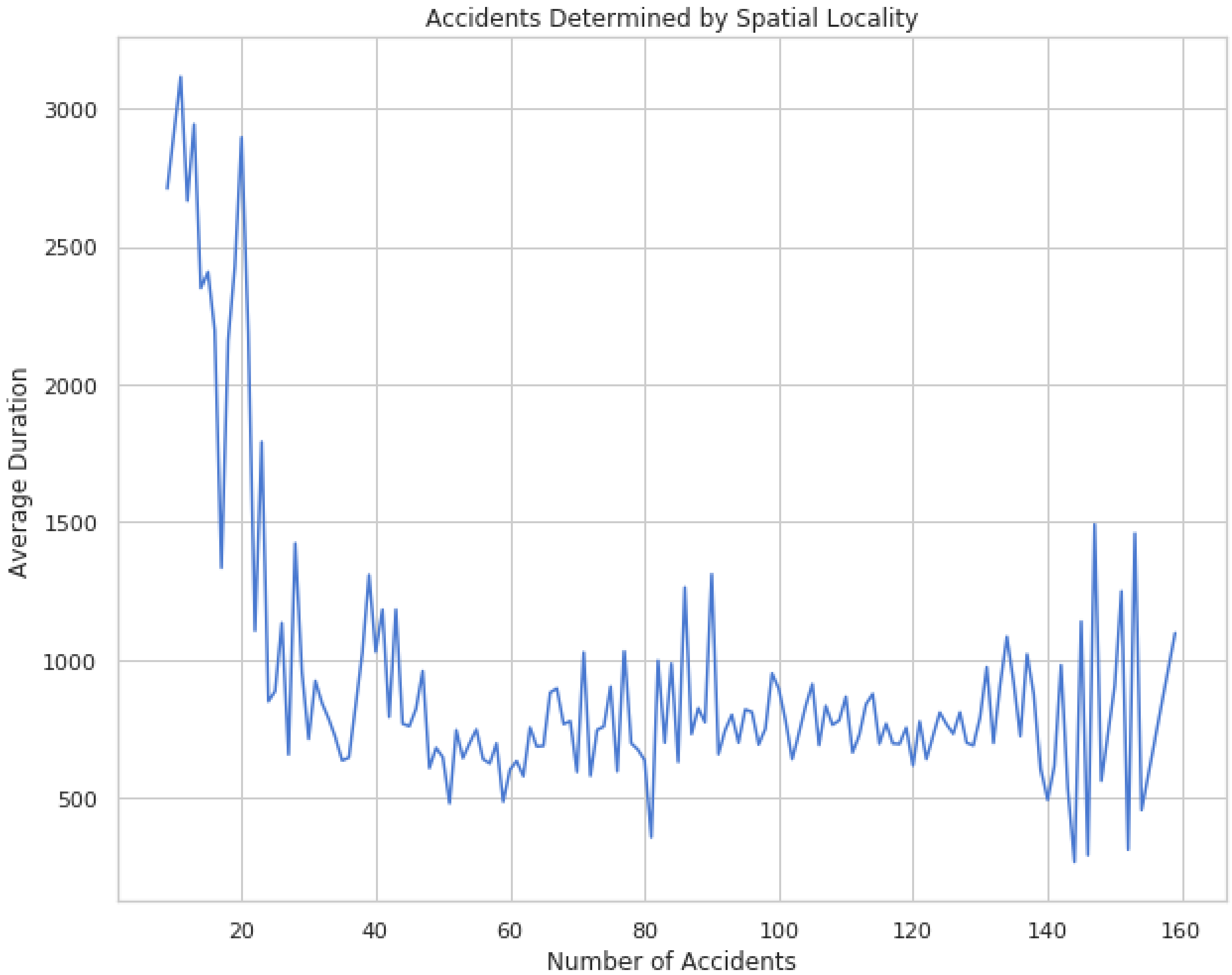


It seems that using both spatial and temporal proximity doesn't give us much insight on if collisions increase taxi ride durations. Let's try conditioning on spatial proximity and temporal proximity separately and see if there are more interesting results there.

In [28]:
```
# Temporal locality

# Condition on time
index = (((merged['tpep_pickup_datetime'] >= merged['START']) & \
         (merged['tpep_pickup_datetime'] <= merged['END'])))

# Count accidents
merged['accident_close'] = 0
merged.loc[index, 'accident_close'] = 1
num_accidents = merged.groupby(['index'])['accident_close'].sum().to_frame()
train_accidents_temporal = train_df.copy()
train_accidents_temporal['num_accidents'] = num_accidents

# Plot
plt.figure(figsize=(10,8))
train_accidents_temporal.groupby('num_accidents')['duration'].mean().plot()
plt.title("Accidents Determined by Temporal Locality")
plt.xlabel("Number of Accidents")
plt.ylabel("Average Duration")
plt.show();
```


Accidents Determined by Temporal Locality
Average Duration
1600
1400
1200
1000
800
600
400
0
10
20
30
40
50
Number of Accidents

In [29]:
```
# Spatial locality

# Condition on space
index = (merged['start_to_accident'] <= 5)

# Count accidents
merged['accident_close'] = 0
merged.loc[index, 'accident_close'] = 1
num_accidents = merged.groupby(['index'])['accident_close'].sum().to_frame()
train_accidents_spatial = train_df.copy()
train_accidents_spatial['num_accidents'] = num_accidents

# Plot
plt.figure(figsize=(10,8))
train_accidents_spatial.groupby('num_accidents')['duration'].mean().plot()
plt.title("Accidents Determined by Spatial Locality")
plt.xlabel("Number of Accidents")
plt.ylabel("Average Duration")
plt.show();
```



Question 2d

By conditioning on temporal and spatial proximity separately, we reveal different trends in average ride duration as a function of number of accidents nearby.

What can you say about the temporal and spatial proximity of accidents to taxi rides and the effect on ride duration? Think of a new hypothesis regarding accidents and taxi ride durations and explain how you would test it.

Additionally, comment on some of the assumptions being made when we condition on temporal and spatial proximity separately. What are the implications of only considering one and not the other?

In [30]:
```
q2d_answer = r"""

We see that the spatial and temporal proximities seperately show much more about our given data than th

"""

# YOUR CODE HERE
#raise NotImplementedError()

print(q2d_answer)
```

```
We see that the spatial and temporal proximities seperately show much more about our given data than t
he combined effect. We can insinuate that this very likely correlates to the various severities of acc
idents that can occur and also that collectively one or the other may impact the accident more given t
he focus on Manhattan accidents and taxis. From the first graph we see that the taxi rides that had nu
merous accidents around them tended to have high durations on their trips, yet that it does not matter
with regards to the actual number around them as in 1 accident appears to affect duration just as much
as 8. Further, for plot 2, the one that focusses upon temporal proximity, we can identify that when th
e number of accidents increases significantly then the taxi duration increases immenseley as well whic
h corresponds with what we originally beleived. Yet, there is an unusual small correlation up until th
e number of accidents increases above 40 where there is a then a massive spike in duration. For plot n
umber 3, which focusses upon spatial proximity, we see the sharpest time duration occurs when the numb
er of accidents fitting the criteria of the ride is less than 20. This appears as a very unusual resul
t to see as one would assume that more accidents should imply a higher time duration. This may be a re
sult of the fact that if there are a large amount of accidents in the proximity it is likely that they
are all lighter and thus the result of one large scale accident, yet that if the number of accidents i
s small then those could have been the result of a severe accident between cars that had the space to
go fast such as late at night. My new hypothesis regarding accidents and taxi ride durations would be
that the time of the day plays a heavy weight into whether or not the average duration will be affecte
d by an immense amount of accidents as there is a high likelihood that certain times of the day trigge
r more severe accidents which would require immediate attention that results in the longer durations o
f focus. I could test this through incorporating a time of day aspect into my model or by creating a s
catterplot of time of day with the duration to see if there is a trend.
```

Part 3 Exports

We are not requiring you to export anything from this notebook, but you may find it useful to do so. There is a space below for you to export anything you wish.

In [31]:
```
Path("data/part3").mkdir(parents=True, exist_ok=True)
data_file = Path("data/part3", "data_part3.hdf") # Path of hdf file
...
```

Out[31]:
```
Ellipsis
```

Part 3 Conclusions

We merged the NYC Accidents dataset with our NYC Taxi dataset, conditioning on temporal and spatial locality. We explored potential features by visualizing the relationship between number of accidents and the average duration of a ride.

Please proceed to part 4 where we will be engineering more features and building our models using a processing pipeline.

Submission

You're almost done!

Before submitting this assignment, ensure that you have:

1. Restarted the Kernel (in the menubar, select Kernel→Restart & Run All)
2. Validated the notebook by clicking the "Validate" button.

Then,

1. **Submit** the assignment via the Assignments tab in **Datahub**
2. **Upload and tag** the manually reviewed portions of the assignment on **Gradescope**