ENCE 201 Engineering Information Processing, Spring Semester, 2025

Solutions to Homework 4

Question 1: 20 points.

Problem Statement: On April 12, 1912, the RMS Titanic (a mail and passenger vessel) commenced her maiden voyage across the Atlantic, departing from Ireland and headed to New York. Three days later (April 15) the Titanic struck an iceberg and sank. Of the 2,224 passengers and crew aboard, approximately 1,500 died, making it the deadliest sinking of a ship at that time. Wikipedia has a nice writeup on the Titanic and the numerous deficiencies that lead to this disaster.

In more recent times, the sinking of the Titanic has inspired numerous artistic works, including the 1997 romantic disaster film Titanic (directed by James Cameron). Like many movies, Titanic places an emphasis on romance and drama – a viewer might wonder, how much of this story is true? Who survived, and why? What does the data say?

Data Source. The data file `python-code.d/data/disaster/titanic.csv` contains information on 887 of the passengers and their attributes, including:

```
--- Survived: 1 means passenger survived; 0 for victims.
--- Pclass:   1, 2 and 3 for first, second and third class.
--- Name:     Master/miss first name, family name.
--- Sex:      male or female.
--- Age:      covers the range 0 to 80.
--- Siblings/Spouses Aboard
--- Parents/Children Aboard
--- Fare:     First class (1) tickets are the most expensive.
```

Things to do:

1. Write a Python program that will read `titanic.csv` into a Pandas dataframe.
2. Separate the data into two categories: passengers that survived, passengers that drowned. For each category compute the relevant statistics (e.g., how many people, ratio of males and females, number of passengers in each passenger class).
3. Generate histograms for the distribution of age among the survivors and victims.

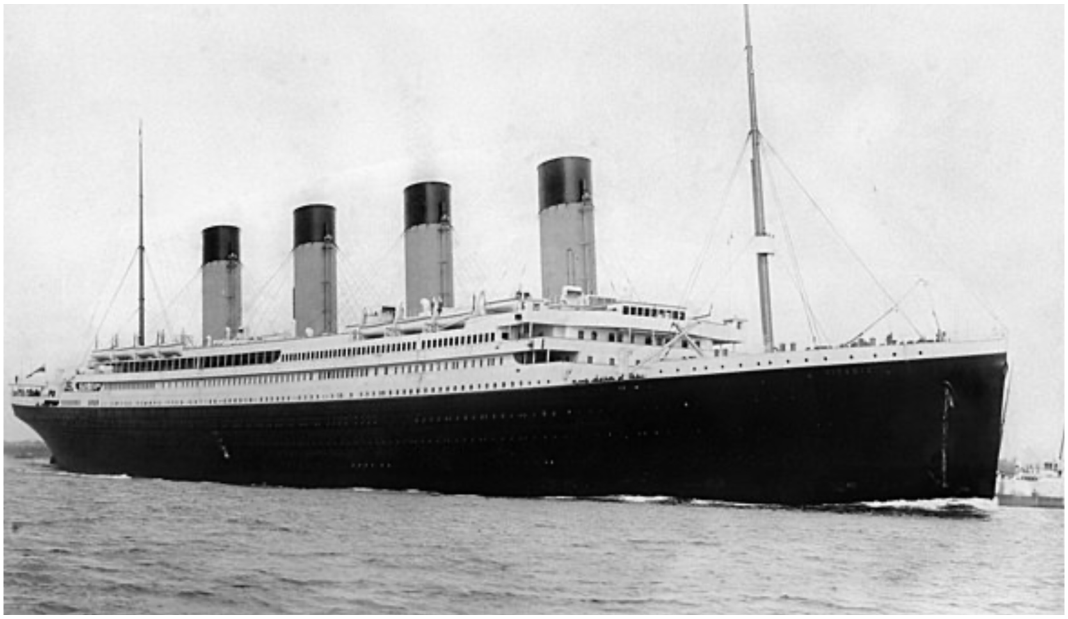

Figure 1: Titanic departing Southampton on April 10, 1912.

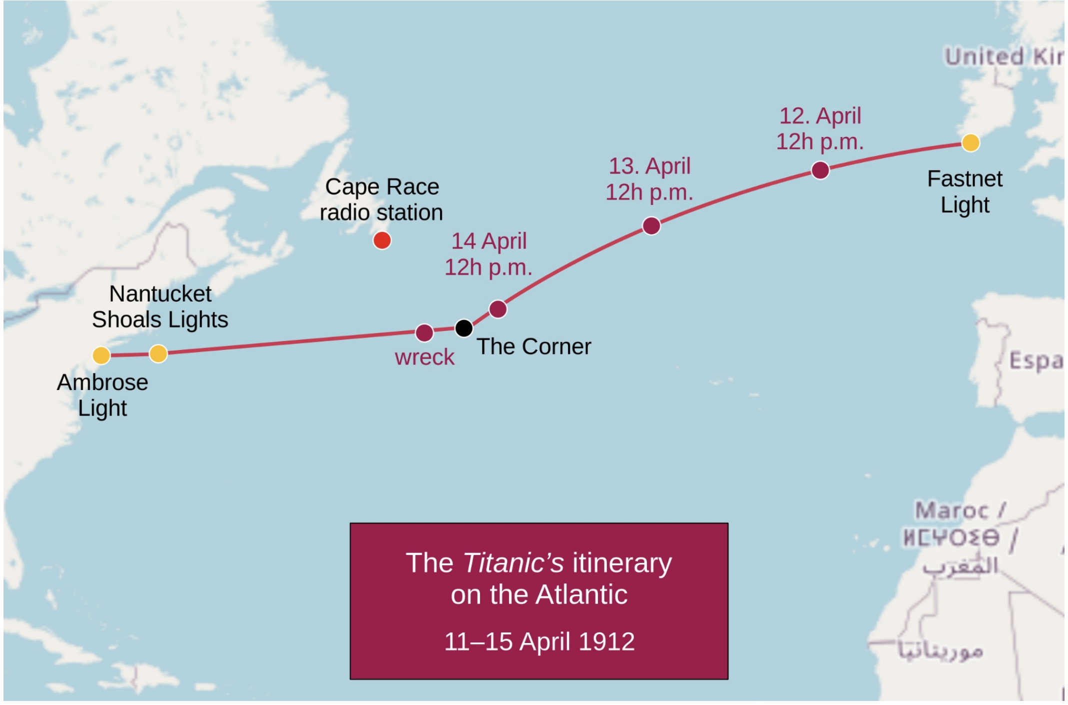

Figure 2: Trans-Atlantic route for the Titanic, from Ireland to New York.

In Cameron’s movie, women and children were given priority to board a lifeboat, and hence survived.

4. Is this part of the story supported by the `titanic.csv` data, or not?

5. Is there any evidence in the data that first class passengers (class 1) were given priority in boarding a lifeboat?

Note: In the early 1900s a child would be someone younger than say 10 or 12 (not 18). And you’d expect children and their mothers would be given access to the lifeboats, regarless of their gender. So, a reasonable strategy is: isolate lists for each of these categories and compute appropriate percentages.

So, a reasonable strategy is: isolate lists for each of these categories and compute appropriate percentages.

Python Source Code:

```
# ======================================================================================
# TestDataProcessingTitanic.py: Read, process and visualize data from data/titanic.csv.
#
# Written by: Mark Austin                                                    April, 2025
# ======================================================================================

import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d

from pandas import DataFrame
from pandas import read_csv

# ===============================================================
# Main function ...
# ===============================================================

def main():
    print("--- Enter TestDataProcessingTitanic.main()         ... ");
    print("--- ============================================== ... ");
    print("");

    # Load and print dataset

    print("--- ");
    print("--- Part 01: Load titanic data file ... ");
    print("--- ");

    df = pd.read_csv(’data/titanic.csv’)

    print("--- ");
    print("--- Part 02: Titanic dataframe description ... ");
    print("--- ");

    print( df.describe() )

    print("--- ");
    print("--- Part 03: Iterate over dataframe columns ... ");
    print("--- ");

    for col in df.columns:
       print(col)

    # Filter dataframe to separate survivors and victims ...

    print("--- ");
    print("--- Part 04: Filter dataframe to separate survivors and victims ... ");
    print("--- ");

    dfSurvived = df [ df[’Survived’] > 0 ].sort_values ( by=[’Age’] )
    print(dfSurvived)

    print( dfSurvived.info() )
    print( dfSurvived.shape )

    dfVictim = df [ df[’Survived’] == 0 ].sort_values ( by=[’Age’] )
    print(dfVictim)

    print( dfVictim.info() )
    print( dfVictim.shape )

    print("--- ");
    print("--- ====================================== ... ");
    print("--- Part 05: Gather statistics of survivors ... ");
    print("--- ====================================== ... ");
    print("--- ");

    print("--- ");
    print("--- Passenger,   Age, Passenger Class, Sex,   Fare ...");
    print("--- ======================================================================= ... ");

    # Traverse rows of survivors dataframe ...

    i = 1
    noFemaleSurvivors = 0
    noMaleSurvivors   = 0
    noChildren05      = 0
    noChildren10      = 0
    noChildren15      = 0
    noPClass01        = 0
    noPClass02        = 0
    noPClass03        = 0
    for index, row in dfSurvived.iterrows():
        age    = row["Age"];
        pclass = row["Pclass"];
        sex    = str( row["Sex"] );
        fare   = row["Fare"];

        # Print details of survivors ...

        print("--- {:9d}:  {:5.1f}, {:15d}, {:10s}, {:5.1f} ... ".format(i, age, pclass, sex.rjust(10

        # Gather statistics ...

        if(sex == "male"):
           noMaleSurvivors = noMaleSurvivors + 1;
        if(sex == "female"):
           noFemaleSurvivors = noFemaleSurvivors + 1;

        if(age <= 5):
           noChildren05 = noChildren05 + 1;
        if(age <= 10):
           noChildren10 = noChildren10 + 1;
        if(age <= 15):
           noChildren15 = noChildren15 + 1;

        # Gather count for no passengers in each pclass ...

        match pclass:
             case 1:
                  noPClass01 = noPClass01 + 1
             case 2:
                  noPClass02 = noPClass02 + 1
             case 3:
                  noPClass03 = noPClass03 + 1
             case _:
                  print("--- pclass not defined ..." );

        i = i + 1;

    print("--- ======================================================================== ... ");
    print("--- ");
    print("--- Summary of Statistics for Survivors:");
    print("--- ");
    print("--- No male survivors       = {:d} ...".format( noMaleSurvivors ));
    print("--- No female survivors     = {:d} ...".format( noFemaleSurvivors ));
    print("--- ");
    print("--- No children (age <=  5) = {:d} ...".format( noChildren05 ));
    print("--- No children (age <= 10) = {:d} ...".format( noChildren10 ));
    print("--- No children (age <= 15) = {:d} ...".format( noChildren15 ));
    print("--- ");
    print("--- No passengers (pclass 1) = {:d} ...".format( noPClass01 ));
    print("--- No passengers (pclass 2) = {:d} ...".format( noPClass02 ));
    print("--- No passengers (pclass 3) = {:d} ...".format( noPClass03 ));

    print("--- ");
    print("--- ============================================ ... ");
    print("--- Part 06: Gather details of passenger victims ... ");
    print("--- ============================================ ... ");
    print("--- ");
    print("--- ");
    print("--- Passenger,    Age, Passenger Class, Sex,   Fare ...");
    print("--- ======================================================================== ... ");

    # Traverse rows of victims dataframe ...

    i = 1
    noFemaleVictims = 0
    noMaleVictims   = 0
    noChildren05    = 0
    noChildren10    = 0
    noChildren15    = 0
    noPClass01      = 0
    noPClass02      = 0
    noPClass03      = 0
    for index, row in dfVictim.iterrows():
        age    = row["Age"];
        pclass = row["Pclass"];
        sex    = str( row["Sex"] );
        fare   = row["Fare"];

        # Print details of victims ...

        print("--- {:9d}:  {:5.1f}, {:15d}, {:10s}, {:5.1f} ... ".format(i, age, pclass, sex.rjust(10

        # Gather statistics ...

        if(sex == "male"):
           noMaleVictims = noMaleVictims + 1;
        if(sex == "female"):
           noFemaleVictims = noFemaleVictims + 1;

        if(age <= 5):
           noChildren05 = noChildren05 + 1;
        if(age <= 10):
           noChildren10 = noChildren10 + 1;
        if(age <= 15):
           noChildren15 = noChildren15 + 1;

        # Gather count for no passengers in each pclass ...

        match pclass:
             case 1:
                  noPClass01 = noPClass01 + 1
             case 2:
                  noPClass02 = noPClass02 + 1
             case 3:
                  noPClass03 = noPClass03 + 1
             case _:
                  print("--- pclass not defined ..." );

        i = i + 1;

    print("--- ======================================================================== ... ");
    print("--- ");
```

```
    print("--- Summary of Statistics for Victims:");
    print("--- ");
    print("--- No male victims         = {:d} ...".format( noMaleVictims ));
    print("--- No female victims       = {:d} ...".format( noFemaleVictims ));
    print("--- ");
    print("--- No children (age <=  5) = {:d} ...".format( noChildren05 ));
    print("--- No children (age <= 10) = {:d} ...".format( noChildren10 ));
    print("--- No children (age <= 15) = {:d} ...".format( noChildren15 ));
    print("--- ");
    print("--- No passengers (pclass 1) = {:d} ...".format( noPClass01 ));
    print("--- No passengers (pclass 2) = {:d} ...".format( noPClass02 ));
    print("--- No passengers (pclass 3) = {:d} ...".format( noPClass03 ));

    print("--- ");
    print("--- Part 04: Create histogram of age of Titanic Survivors/Victims ... ");
    print("--- ");

    # Extract array from dataframe ...

    survivors = np.array ( dfSurvived[’Age’].values )
    victims   = np.array ( dfVictim[’Age’].values )

    # Create histograms ...

    fig, ((ax0,ax1)) = plt.subplots(nrows=1, ncols=2)

    nbins = 20;
    colors = [ ’red’ ]
    ax0.hist( survivors, nbins, density=True, histtype=’bar’, color=colors, label=colors)
    ax0.set_title(’Age of 233 Survivors’);
    ax0.set( xlabel="Age", ylabel="Probability Density")
    ax0.set_xlim( [0,80] )
    ax0.set_ylim( [0,0.05] )
    ax0.grid()

    colors = [ ’blue’ ]
    ax1.hist( victims, nbins, density=True, histtype=’bar’, color=colors, label=colors)
    ax1.set_title(’Age of 464 Victims’);
    ax1.set( xlabel="Age", ylabel="Probability Density")
    ax1.set_xlim( [0,80] )
    ax1.set_ylim( [0,0.05] )
    ax1.grid()

    plt.show()

    print("--- ============================================== ... ");
    print("--- Leave TestDataProcessingAirport.main()         ... ");

# call the main method ...

main()
```

Program Output: The textual output is:

```
--- Enter TestDataProcessingTitanic.main()         ...
--- ============================================== ...

--- Part 01: Load titanic data file ...

--- Part 02: Titanic dataframe description ...

         Survived      Pclass  ...  Parents/Children Aboard       Fare
count  887.000000  887.000000  ...               887.000000  887.00000
mean     0.385569    2.305524  ...                 0.383315   32.30542
std      0.487004    0.836662  ...                 0.807466   49.78204
min      0.000000    1.000000  ...                 0.000000    0.00000
25%      0.000000    2.000000  ...                 0.000000    7.92500
50%      0.000000    3.000000  ...                 0.000000   14.45420
75%      1.000000    3.000000  ...                 0.000000   31.13750
max      1.000000    3.000000  ...                 6.000000  512.32920

[8 rows x 6 columns]

--- Part 03: Iterate over dataframe columns ...

Survived
Pclass
Name
Sex
Age
Siblings/Spouses Aboard
Parents/Children Aboard
Fare

--- Part 04: Filter dataframe to separate survivors and victims ...

     Survived  Pclass  ... Parents/Children Aboard     Fare
799         1       3  ...                       1   8.5167
751         1       2  ...                       1  14.5000
641         1       3  ...                       1  19.2583
466         1       3  ...                       1  19.2583
77          1       2  ...                       2  29.0000
..        ...     ...  ...                     ...      ...
567         1       2  ...                       0  10.5000
825         1       1  ...                       0  80.0000
273         1       1  ...                       0  77.9583
480         1       3  ...                       0   9.5875
627         1       1  ...                       0  30.0000

[342 rows x 8 columns]
<class 'pandas.core.frame.DataFrame'>
Index: 342 entries, 799 to 627
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Survived                 342 non-null    int64
 1   Pclass                   342 non-null    int64
 2   Name                     342 non-null    object
 3   Sex                      342 non-null    object
 4   Age                      342 non-null    float64
 5   Siblings/Spouses Aboard  342 non-null    int64
 6   Parents/Children Aboard  342 non-null    int64
 7   Fare                     342 non-null    float64
dtypes: float64(2), int64(4), object(2)
memory usage: 24.0+ KB
None
(342, 8)
     Survived  Pclass  ... Parents/Children Aboard     Fare
384         0       3  ...                        2  46.9000
163         0       3  ...                        1  39.6875
16          0       3  ...                        1  29.1250
204         0       3  ...                        1  10.4625
820         0       3  ...                        1  39.6875
..        ...     ...  ...                      ...      ...
669         0       2  ...                        0  10.5000
115         0       3  ...                        0   7.7500
490         0       1  ...                        0  49.5042
95          0       1  ...                        0  34.6542
847         0       3  ...                        0   7.7750

[545 rows x 8 columns]
<class 'pandas.core.frame.DataFrame'>
Index: 545 entries, 384 to 847
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Survived                 545 non-null    int64
 1   Pclass                   545 non-null    int64
 2   Name                     545 non-null    object
 3   Sex                      545 non-null    object
 4   Age                      545 non-null    float64
 5   Siblings/Spouses Aboard  545 non-null    int64
 6   Parents/Children Aboard  545 non-null    int64
 7   Fare                     545 non-null    float64
dtypes: float64(2), int64(4), object(2)
memory usage: 38.3+ KB
None
(545, 8)

--- ======================================= ...
--- Part 05: Gather statistics of survivors ...
--- ======================================= ...

---
--- Passenger,    Age, Passenger Class, Sex,   Fare ...
--- ========================================================================== ...
---         1:    0.4,               3,       male,   8.5 ...
---         2:    0.7,               2,       male,  14.5 ...
---         3:    0.8,               3,     female,  19.3 ...

... lines of output removed ...

---       340:   63.0,               1,     female,  78.0 ...
---       341:   63.0,               3,     female,   9.6 ...
---       342:   80.0,             1,       male,  30.0 ...
--- ========================================================================= ...
---
--- Summary of Statistics for Survivors:
---
--- No male survivors       = 109 ...
--- No female survivors     = 233 ...
---
--- No children (age <=  5) = 33 ...
--- No children (age <= 10) = 41 ...
--- No children (age <= 15) = 52 ...
---
--- No passengers (pclass 1) = 136 ...
--- No passengers (pclass 2) = 87 ...
--- No passengers (pclass 3) = 119 ...

--- ============================================ ...
--- Part 06: Gather details of passenger victims ...
--- ============================================ ...

--- Passenger,    Age, Passenger Class, Sex,   Fare ...
--- ========================================================================= ...
---         1:    1.0,             3,       male,  46.9 ...
---         2:    1.0,             3,       male,  39.7 ...
---         3:    2.0,             3,       male,  29.1 ...

... lines of output removed ...

---       543:   71.0,             1,       male,  49.5 ...
---       544:   71.0,             1,       male,  34.7 ...
---       545:   74.0,             3,       male,   7.8 ...
--- ========================================================================= ...

--- Summary of Statistics for Victims:
---
--- No male victims         = 464 ...
--- No female victims       = 81 ...
---
--- No children (age <=  5) = 16 ...
--- No children (age <= 10) = 32 ...
--- No children (age <= 15) = 42 ...
---
--- No passengers (pclass 1) = 80 ...
--- No passengers (pclass 2) = 97 ...
--- No passengers (pclass 3) = 368 ...

--- Part 04: Create histogram of age of Titanic Survivors/Victims ...

--- ============================================== ...
--- Leave TestDataProcessingAirport.main()         ...
```

Interpretation of Statistics. A side-by-side comparison of the statistics for survivors and victims reveals:

```
-----------------------------------------------------------
```

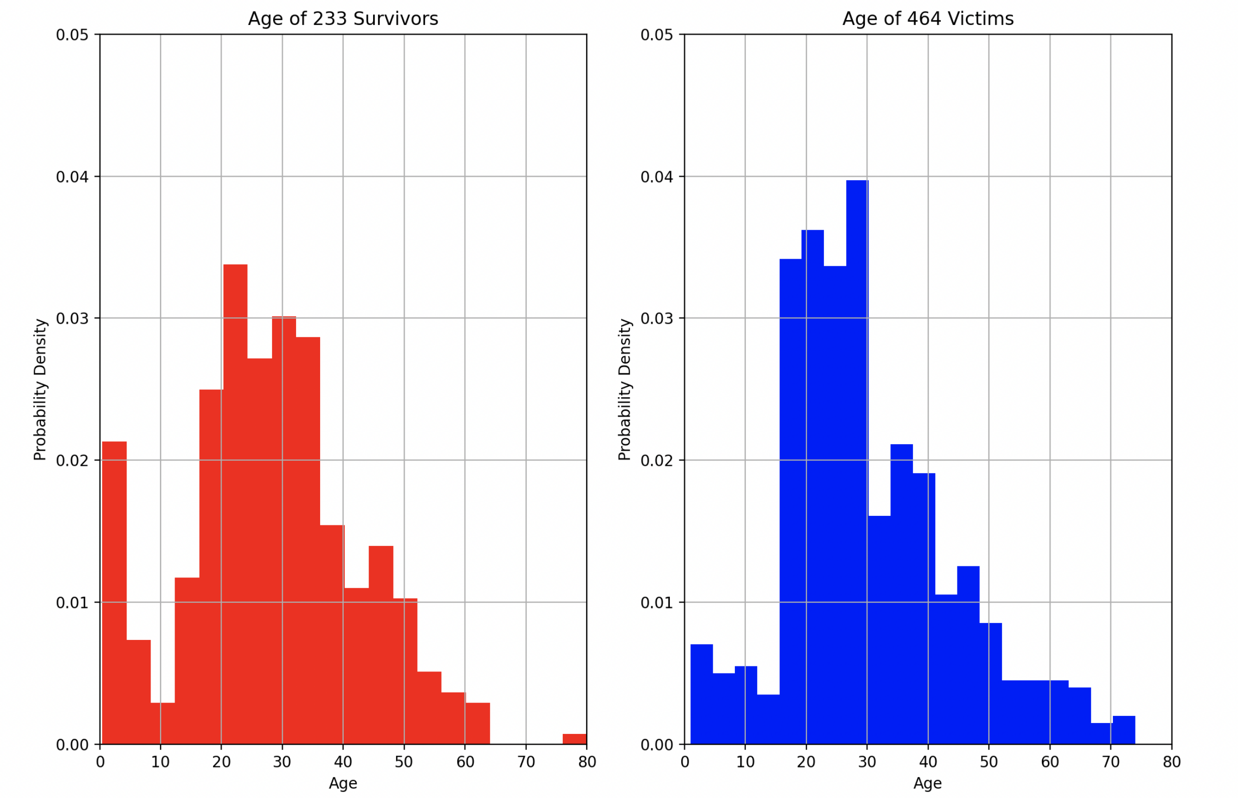

Figure 3: Histogram of probability density vs age for survivors and victims on the Titanic (blue + red areas sum to 1.0).

```
                     Factor         Survivors        Victims
===========================================================
                     Males:               109            464
                   Females:               233             81
                   ----------------------------------------
                     Total:               342            545
                   ----------------------------------------

Detailed Analysis:
-----------------------------------------------------------

Young children (age <=  5):                33             16
Young children (age <= 10):                41             32
Young children (age <= 15):                52             42

   Passengers (pClass = 1):               136             80
   Passengers (pClass = 2):                87             97
   Passengers (pClass = 3):               119            368
===========================================================
```

A few point to note:

1. The passenger list is comprised of 573 males and 314 females, so not a 50-50 split. The data indicates that on a percentage basis, females were much more likely to survive than males.

2. Two-thirds of young children (age ≤ 5) survived.

3. High paying passengers, passenger class 1, were much more likely to survive than those in passenger classes 2 and 3.

Question 2: 20 points.

Problem Statement: Motor vehicle accidents in New York City are the leading cause of death for the city residents. Statistics indicate that close to one in four accidents results in someone being injured or losing their life. The problem is particularly acute for accidents involving high-speed, and/or when accidents involve pedestrians, bicyclists, or motor cycles. Root causes for this situation can be traced back to the city being flat and very walkable, as well as a significant biking culture.

Preliminary Work: During the Spring Semester, 2024, we took a first step toward formally analyzing data on motor vehicle accidents and, specifically, understand **where** and **when** vehicle accidents occur? Figure 4 shows the spatial distribution of 312,000 motor vehicle accidents in Manhattan. The graphic suggests that with the exception of Central Park, accidents occur everywhere. Figure 5 is heatmap of the temporal distribution of accidents. As expected, during the work week, accidents peak during the morning and afternoon rush hours. During the weekend, accidents peak early in the morning, presumably after people have been out socializing and are headed home.

Problem Statement: This question seeks to understand the number and types of vehicle (e.g., taxis, bicycle/bike, e-bike, e-scooter, sport utility vehicle, station wagon, sedan, van, fire truck, pick-up truck, tractor truck, ambulance, bus, tanker), involved in accidents, where and when such accidents occur, and whether or not accidents result in injuries and/or fatalities.

From a medical standpoint it would be interesting to understand whether or not the data contains information on the types of accident (i.e., vehicles involved; time of the day) that lead to fatalities.

Our solution will gather data from multiple sources: (1) geospatial (to draw the city map, major streets, and coastline), (2) accident data.

Data Source 1 – Geospatial: Within the folder `data/cities/nyc` the data files `dcm-nyc-major-street.csv` and `nyc-shoreline.csv` contain geospatial data on the main streets and shoreline in the NYC area.

To run, see: python-code.d/applications/cities/nyc/TestMajorStreetsNYC.py.

Data Source 2 – Accident: The folder `python-code.d/data/cities/nyc/` contains data files that can be used in the analysis of motor vehicle accidents in NYC. The main file, Motor-Vehicle-Collisions.csv, comprises 2.06 million motor vehicle accidents recorded across the five boroughs of NYC and for about a decade.

The data is organized into 29 columns:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2062758 entries, 0 to 2062757
Data columns (total 29 columns):
```

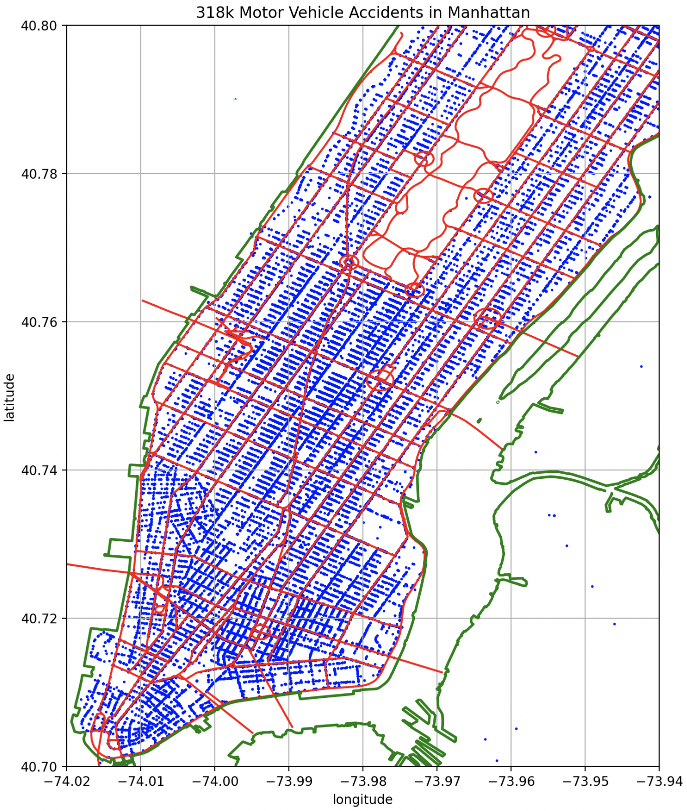

Figure 4: Spatial distribution of motor vehicle accidents in Manhattan.

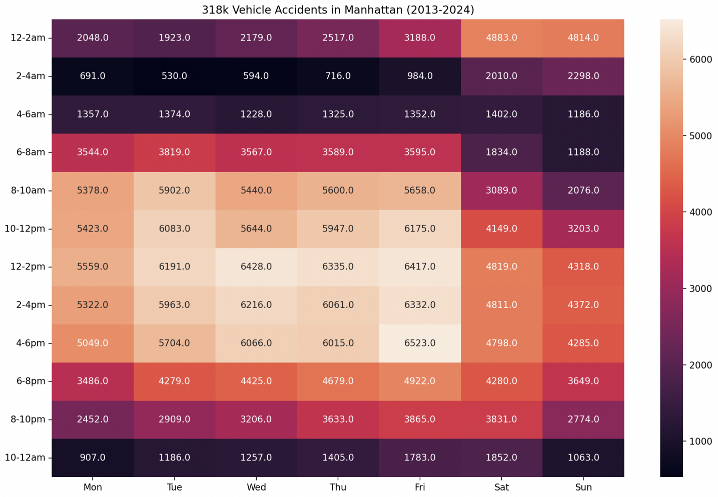

Figure 5: Temporal view of 318k motor vehicle accidents in Manhattan (2013-2024).

```
 #   Column                         Dtype
---  ------                         -----
 0   CRASH DATE                     object
 1   CRASH TIME                     object
 2   BOROUGH                        object
 3   ZIP CODE                       object
 4   LATITUDE                       float64
 5   LONGITUDE                      float64
 6   LOCATION                       object
 7   ON STREET NAME                 object
 8   CROSS STREET NAME              object
 9   OFF STREET NAME                object
 10  NUMBER OF PERSONS INJURED      float64
 11  NUMBER OF PERSONS KILLED       float64
 12  NUMBER OF PEDESTRIANS INJURED  int64
 13  NUMBER OF PEDESTRIANS KILLED   int64
 14  NUMBER OF CYCLIST INJURED      int64
 15  NUMBER OF CYCLIST KILLED       int64
 16  NUMBER OF MOTORIST INJURED     int64
 17  NUMBER OF MOTORIST KILLED      int64
 18  CONTRIBUTING FACTOR VEHICLE 1  object
 19  CONTRIBUTING FACTOR VEHICLE 2  object
 20  CONTRIBUTING FACTOR VEHICLE 3  object
 21  CONTRIBUTING FACTOR VEHICLE 4  object
 22  CONTRIBUTING FACTOR VEHICLE 5  object
 23  COLLISION_ID                   int64
 24  VEHICLE TYPE CODE 1            object
 25  VEHICLE TYPE CODE 2            object
 26  VEHICLE TYPE CODE 3            object
 27  VEHICLE TYPE CODE 4            object
 28  VEHICLE TYPE CODE 5            object
dtypes: float64(4), int64(7), object(18)
memory usage: 456.4+ MB
None
(2062758, 29)
```

Columns 10 through 17 store data on injuries/deaths to pedestrians, cyclists and motorists. Factors contributing to an accident are located in columns 18 through 22 (it's NYC, road rage is common). Columns 24 through 28 store the associated vehicle types (could be more than two) involved in the accident.

Things to do:

1. Download the latest version of python-code.zip, unpack, and run the python code that generates Figures 4 and 5. To generate the spatial view, run:

see: python-code.d/applications/cities/nyc/TestMotorVehicleAccidentsNYC01.py.

And to generate the temporal view (heatmap):

see: python-code.d/applications/cities/nyc/TestMotorVehicleAccidentsNYC02.py.

Both programs filter the accident data to only keep accidents occuring in Manhattan – this operation will reduce the number of accidents from 2 million to approximately 318,000. Then, they remove from further consideration accidents that do not have (lat,long) coordinates (c.f., there are about 10,000 of them).

2. The data indicates that approximately 17% of accidents result in injuries; slightly less than 0.1% of accidents result in fatalities. Write a Python program that will gather and print print the total number of accidents and injuries/fatalities involving cyclists and pedestrians.

A snippet of the accident statistics might look like:

```
--- Total No Accidents     = xxx ...

--- No Persons Injured     = xxx ...
--- No Cyclists Injured    = xxx ...
--- No Pedestrians Injured = xxx ...
--- No Motorists Injured   = xxx ...

--- No Persons Killed      = xxx ...
--- No Cyclists Killed     = xxx ...
--- No Pedestrians Killed  = xxx ...
--- No Motorists Killed    = xxx ...
```

3. Extend your program to systematally assemble a dictionary (i.e., key-value pairs) of vehicle types and counts of accidents in which they are involved.

A snippet of dictionary content might look like:

```
Vehicle Type                Accident Count
==========================================
e-scooter                            1,131
e-bike                               1,261
dump truck                           1,287
ambulance                            2,114
motorcycle                           4,054
bicycle                              6,069

  ... many lines of output removed ...

bike                                11,403

  ... more lines of output removed ...

bus                                 17,535
van                                 21,346
sport utility/station wagon         52,151
station wagon/sport utility vehicle 79,262
taxi                                79,867
sedan                              105,686
passenger vehicle                  110,392
==========================================
```

4. From the previous section we see that over the 2013-2024 period, bikes (i.e., bike, bicycle, e-scooter, e-bike) have been involved in more than 17000 accidents. And more than 50% of these accidents result in injury and/or death.

Create heatmap views of pedestrian and cyclist injuries/fatalities in Manhattan, 2013-2024.

5. Briefly summarize and discuss your findings.

Python Source Code:

```
# ======================================================================================
# TestMotorVehicleAccidentsNYC03.py: Multiple purposes:
#
# 1. Read, process and visualize data from data/cities/nyc/Motor-Vehicle-Collisions.csv
#    2,062,760 motor vehicle collisions in NYC.
#
# 2. Create tables of vehicles involved in accidents ...
#
# 3. Create plot of temporal dimensions of motor vehicle accidents ...
#
# Written by: Mark Austin                                                    April, 2025
# ======================================================================================

import math

import numpy as np
import pandas as pd
import seaborn as sns

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d

from pandas import DataFrame
from pandas import Series
from pandas import read_csv

from datetime import datetime

from collections import OrderedDict

# ===============================================================
# Print dictionary ...
# ===============================================================

def printDictionary( title, dictionary, sort = False ):
    print("Dictionary: {:s}:".format(title))
    print("-------------------------------------------------- ")

    if sort == True:
       sortedDictionary = OrderedDict( sorted( dictionary.items() ) )
       for key, value in sortedDictionary.items():
          print("key: {:s} --> value: {:s} ...".format( key, str(value) ))
    else:
       for key, value in dictionary.items():
          print("key: {:s} --> value: {:s} ...".format( key, str(value) ))

    print("--------------------------------------------------- ")

# ===============================================================
# Main function ...
# ===============================================================

def main():
    print("--- Enter TestMotorVehicleAccidentsNYC03.main()         ... ");
    print("--- =================================================== ... ");
    print("");

    # Load and print dataset

    print("--- ");
    print("--- Part 01: Load motor vehicle collisions data file           ...");
    print("---        : Filter data to only keep accidents in Manhattan   ...");
    print("--- ---------------------------------------------------------- ...");

    df = pd.read_csv('data/cities/nyc/Motor-Vehicle-Collisions.csv')

    options = ['MANHATTAN']
    dfManhattan = df [ df['BOROUGH'].isin(options) ].copy()

    print( dfManhattan.describe() )
    print( dfManhattan.info() )
    print( dfManhattan.shape )

    print("--- ");
    print("--- Part 02: Convert dfManhattan dataframe to list ...");
    print("--- ---------------------------------------------- ...");
    print("--- ");

    manhattanlist = dfManhattan.values.tolist();

    print("--- Part 03: Create dictionary of vehicles involved in accidents ... ")
    print("--- ------------------------------------------------------------ ...");
    print("--- ");

    # Create dictionary for storing vehicle types involved in accidents ...

    statsVehiclesAccidents = {}

    # Traverse list of accidents, populate dictionary ...

    i = 1;
    for row in manhattanlist:

        # Types of vehicles involved in accident ...

        vehicletype01 = row[24];
        vehicletype02 = row[25];
        vehicletype03 = row[26];
        vehicletype04 = row[27];
        vehicletype05 = row[28];

        # Details on types of vehicle involved ...

        if isinstance(vehicletype01, str ):
           if vehicletype01.lower() in statsVehiclesAccidents:
              statsVehiclesAccidents[ vehicletype01.lower() ] += 1
           else:
              statsVehiclesAccidents[ vehicletype01.lower() ] = 1

        if isinstance(vehicletype02, str ):
           if vehicletype02.lower() in statsVehiclesAccidents:
              statsVehiclesAccidents[ vehicletype02.lower() ] += 1
           else:
              statsVehiclesAccidents[ vehicletype02.lower() ] = 1

        if isinstance(vehicletype03, str ):
           if vehicletype03.lower() in statsVehiclesAccidents:
              statsVehiclesAccidents[ vehicletype03.lower() ] += 1
           else:
              statsVehiclesAccidents[ vehicletype03.lower() ] = 1

        if isinstance(vehicletype04, str ):
           if vehicletype04.lower() in statsVehiclesAccidents:
              statsVehiclesAccidents[ vehicletype04.lower() ] += 1
           else:
              statsVehiclesAccidents[ vehicletype04.lower() ] = 1

        if isinstance(vehicletype05, str ):
           if vehicletype05.lower() in statsVehiclesAccidents:
              statsVehiclesAccidents[ vehicletype05.lower() ] += 1
           else:
              statsVehiclesAccidents[ vehicletype05.lower() ] = 1

        i = i + 1;

    # Print dictionary of motor vehicle types involved in accidents ...

    printDictionary( "Motor Vehicle Types", statsVehiclesAccidents, sort = True );

    # Sort motor vehicle accident items, then print ...

    print("--- ");
    print("--- Sorted Motor Vehicle Accidents ... ");
    print("--- ------------------------------ ...");
    print("--- ");

    for i in sorted( statsVehiclesAccidents, key = statsVehiclesAccidents.get ):
        print("--- Vehicle Type: {:30s} --> Accident Count: {:2d} ...".format(i, statsVehiclesAccider

    print("--- ");
    print("--- Part 04: Gather statistics for injuries/fatalities ... ")
    print("--- ------------------------------------------------------------ ...");
    print("--- ");

    # Variables to keep track of accident statistics ...

    noPersonsInjured     = 0;
    noCyclistsInjured    = 0;
    noPedestriansInjured = 0;
    noMotoristsInjured   = 0;

    noPersonsKilled     = 0;
    noCyclistsKilled    = 0;
    noPedestriansKilled = 0;
    noMotoristsKilled   = 0;

    # Traverse list, print details of individual motor vehicle accidens ...

    i = 1;
    for row in manhattanlist:
        crashdate = str( row[0] );
        crashtime = str( row[1] );
        zipcode = str( row[3] );
        lat     = row[4];
        long    = row[5];

        # Location and street address ...

        location        = row[6];
        onstreetname    = row[7];
        crossstreetname = row[8];
        offstreetname   = row[9];

        personinjured     = row[10];
        personkilled      = row[11];

        pedestrianinjured = row[12];
        pedestriankilled  = row[13];

        cyclistinjured    = row[14];
        cyclistkilled     = row[15];

        motoristinjured   = row[16];
        motoristkilled    = row[17];

        # Accident cause ...

        vehicle01 = row[18];
        vehicle02 = row[19];
        vehicle03 = row[20];
        vehicle04 = row[21];
        vehicle05 = row[22];

        collisionID = row[23];

        # Types of vehicles involved in accident ...
```

```
vehicletype01 = row[24];
vehicletype02 = row[25];
vehicletype03 = row[26];
vehicletype04 = row[27];
vehicletype05 = row[27];

# Details on accident location and time ...

days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

if  math.isnan(lat) == False and math.isnan(long) == False:
    print("--- {:8d}: (date, time) --> ({:5s}, {:5s}); (lat,long) = ({:f}, {:f})".format(i, c

# Location and Street address ...

if isinstance(location, str ):
   print("---   GeoLocation: {:s} ... ".format( location ) );
if isinstance(onstreetname, str ):
   print("---   On Street Name: {:s} ... ".format( onstreetname ) );
if isinstance(crossstreetname, str ):
   print("---   Cross Street Name: {:s} ... ".format( crossstreetname ) );
if isinstance(offstreetname, str ):
   print("---   Off Street Name: {:s} ... ".format( offstreetname ) );

# Details on accident cause ...

if isinstance(vehicle01, str ):
   print("---   Accident Cause (Vehicle01): {:s} ... ".format( str( vehicle01 ) ) );
if isinstance(vehicle02, str ):
   print("---   Accident Cause (Vehicle02): {:s} ... ".format( str( vehicle02 ) ) );
if isinstance(vehicle03, str ):
   print("---   Accident Cause (Vehicle03): {:s} ... ".format( str( vehicle03 ) ) );
if isinstance(vehicle04, str ):
   print("---   Accident Cause (Vehicle04): {:s} ... ".format( str( vehicle04 ) ) );
if isinstance(vehicle05, str ):
   print("---   Accident Cause (Vehicle05): {:s} ... ".format( str( vehicle05 ) ) );

# Details on types of vehicle involved ...

if isinstance(vehicletype01, str ):
   print("---   VehicleType01: {:s} ... ".format( str( vehicletype01 ) ) );
if isinstance(vehicletype02, str ):
   print("---   VehicleType02: {:s} ... ".format( str( vehicletype02 ) ) );
if isinstance(vehicletype03, str ):
   print("---   VehicleType03: {:s} ... ".format( str( vehicletype03 ) ) );
if isinstance(vehicletype04, str ):
   print("---   VehicleType04: {:s} ... ".format( str( vehicletype04 ) ) );
if isinstance(vehicletype05, str ):
   print("---   VehicleType05: {:s} ... ".format( str( vehicletype05 ) ) );

# Persons Injured/Killed ...

if isinstance(personinjured, float ) and personinjured > 0.0:
   print("---   Person Injured: {:.2f} ... ".format( personinjured ) );
```

```
       noPersonsInjured = noPersonsInjured + 1;

    if isinstance(personkilled, float ) and personkilled > 0.0:
       print("---   Person Killed: {:.2f} ... ".format( personkilled ) );
       noPersonsKilled = noPersonsKilled + 1;

    # Pedestrians Injured/Killed ...

    if isinstance(pedestrianinjured, int ) and pedestrianinjured > 0:
       noPedestriansInjured = noPedestriansInjured + 1;
       print("---   Pedestrian Injured: {:d} ... ".format( pedestrianinjured ) );

    if isinstance(pedestriankilled, int ) and pedestriankilled > 0:
       print("---   Pedestrian Killed: {:d} ... ".format( pedestriankilled ) );
       noPedestriansKilled = noPedestriansKilled + 1;

    # Cyclists Injured/Killed ...

    if isinstance(cyclistinjured, int ) and cyclistinjured > 0:
       noCyclistsInjured = noCyclistsInjured + 1;
       print("---   Cyclist Injured: {:d} ... ".format( cyclistinjured ) );

    if isinstance(cyclistkilled, int ) and cyclistkilled > 0:
       print("---   Cyclist Killed: {:d} ... ".format( cyclistkilled ) );
       noCyclistsKilled = noCyclistsKilled + 1;

    # Motorists Injured/Killed ...

    if isinstance(motoristinjured, int ) and motoristinjured > 0:
       noMotoristsInjured = noMotoristsInjured + 1;
       print("---   Motorist Injured: {:d} ... ".format( motoristinjured ) );

    if isinstance(motoristkilled, int ) and motoristkilled > 0:
       print("---   Motorist Killed: {:d} ... ".format( motoristkilled ) );
       noMotoristsKilled = noMotoristsKilled + 1;

    i = i + 1;

print("--- ");
print("--- Part 04: Summary of Accidents Statistics ...");
print("--- ---------------------------------------- ...");
print("--- ");

print("--- Total No Accidents     = {:d} ...\n".format( i ) );

print("--- No Persons Injured     = {:.1f} ...".format( noPersonsInjured ) );
print("--- No Cyclists Injured    = {:.1f} ...".format( noCyclistsInjured ) );
print("--- No Pedestrians Injured = {:.1f} ...".format( noPedestriansInjured ) );
print("--- No Motorists Injured   = {:.1f} ...\n".format( noMotoristsInjured ) );

print("--- No Persons Killed      = {:d} ...".format( noPersonsKilled ) );
print("--- No Cyclists Killed     = {:d} ...".format( noCyclistsKilled ) );
print("--- No Pedestrians Killed  = {:d} ...".format( noPedestriansKilled ) );
print("--- No Motorists Killed    = {:d} ...".format( noMotoristsKilled ) );
print("--- ");
print("--- Part 05: Assemble heatmap for Pedestrians Injured/Killed ...");
print("--- ---------------------------------------------------- ...");
print("--- ");

heatmapdata = np.zeros( [ 12, 7] )

i = 1;
for row in manhattanlist:
    crashdate    = str( row[0] );
    crashtime    = str( row[1] );
    zipcode      = str( row[3] );
    lat          = row[4];
    long         = row[5];

    pedestrianinjured = row[12];
    pedestriankilled  = row[13];

    cyclistinjured = row[14];
    cyclistkilled  = row[15];

    # Traverse list, print details of individual motor vehicle accidens ...
    # if cyclistinjured > 0.0 or cyclistkilled > 0.0:

    if pedestrianinjured > 0.0 or pedestriankilled > 0.0:

        # Example with the standard date and time format

        date_format = '%m/%d/%Y %H:%M'
        date_obj = datetime.strptime( crashdate + " " + crashtime, date_format)

        days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

        # print("--- components of time ... ")
        # print("---    year: {:d} ... ".format( date_obj.year ) )
        # print("---    month: {:d} ... ".format( date_obj.month ) )
        # print("---    day: {:d} ... ".format( date_obj.day ) )
        # print("---    date_obj.weekday() --> {:d} ... ".format( date_obj.weekday()) )
        # print("---    day of week: {:s} ... ".format( days[ date_obj.weekday() ] ) )
        # print("---    hour: {:d} ... ".format( date_obj.hour ) )
        # print("---    minute: {:d} ... ".format( date_obj.minute ) )

        # Increment data in heatmap cells ...

        colno = date_obj.weekday();
        hour  = date_obj.hour;

        if hour <= 2:
           rowno = 0;
        elif 2 < hour <=  4:
           rowno = 1;
        elif 4 < hour <=  6:
           rowno = 2;
        elif 6 < hour <=  8:
           rowno = 3;
```

```
          elif 8 < hour <= 10:
             rowno = 4;
          elif 10 < hour <= 12:
             rowno = 5;
          elif 12 < hour <= 14:
             rowno = 6;
          elif 14 < hour <= 16:
             rowno = 7;
          elif 16 < hour <= 18:
             rowno = 8;
          elif 18 < hour <= 20:
             rowno = 9;
          elif 20 < hour <= 22:
             rowno = 10;
          else:
             rowno = 11;

          heatmapdata[rowno][colno] += 1;

      i = i + 1;

  print("--- ======================================================================================== ...
  print("--- ");

  print("--- Days count ... ")

  print( heatmapdata )

  print("--- Total no accidents = {:d} ... ".format( i ) );

  df = pd.DataFrame( heatmapdata )

  # Change the column names and row indexes ...

  df.columns = [   "Mon",  "Tue",  "Wed", "Thu", "Fri", "Sat", "Sun"]
  df.index   = ["12-2am","2-4am","4-6am","6-8am","8-10am","10-12pm",
                "12-2pm","2-4pm","4-6pm","6-8pm","8-10pm","10-12am" ]

  # Default heatmap with active annotations ...

  p1 = sns.heatmap(df, annot=True, fmt=".1f")

  # This sets the yticks "upright" with 0, as opposed to sideways with 90.

  plt.yticks(rotation=0)
  plt.title("Pedestrian Injuries/Fatalities in Manhattan (2013-2024)")

# plt.title("Cyclist Injuries/Fatalities in Manhattan (2013-2024)")

  plt.show()

  print("--- ");
  print("--- Part 05: Create dictionary of vehicles involved in accidents ... ")
  print("--- ----------------------------------------------------------------- ...");
  print("--- ");
```

```
    print("--- =============================================== ... ");
    print("--- Leave TestMotorVehicleAccidentsNYC03.main()     ... ");

# call the main method ...

if __name__ == "__main__":
    main()
```

Abbreviated Output:

```
--- Enter TestMotorVehicleAccidentsNYC03.main()         ...
--- =================================================== ...
---
--- Part 01: Load motor vehicle collisions data file          ...
---        : Filter data to only keep accidents in Manhattan   ...
--- ---------------------------------------------------------- ...
            LATITUDE      LONGITUDE  ...  NUMBER OF MOTORIST KILLED  COLLISION_ID
count  308929.000000  308929.000000  ...              318800.000000  3.188000e+05
mean       40.688430     -73.835943  ...                   0.000166  2.734547e+06
std         1.765067       3.202378  ...                   0.013134  1.706742e+06
min         0.000000     -74.123400  ...                   0.000000  2.200000e+01
25%        40.739243     -73.992908  ...                   0.000000  3.491368e+05
50%        40.758945     -73.979760  ...                   0.000000  3.439796e+06
75%        40.786485     -73.956024  ...                   0.000000  4.010985e+06
max        40.904600       0.000000  ...                   2.000000  4.699543e+06

[8 rows x 11 columns]
<class 'pandas.core.frame.DataFrame'>
Index: 318800 entries, 9 to 2062756
Data columns (total 29 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   CRASH DATE                     318800 non-null  object
 1   CRASH TIME                     318800 non-null  object
 2   BOROUGH                        318800 non-null  object
 3   ZIP CODE                       318777 non-null  object
 4   LATITUDE                       308929 non-null  float64
 5   LONGITUDE                      308929 non-null  float64
 6   LOCATION                       308929 non-null  object
 7   ON STREET NAME                 258540 non-null  object
 8   CROSS STREET NAME              258472 non-null  object
 9   OFF STREET NAME                60245 non-null   object
 10  NUMBER OF PERSONS INJURED      318797 non-null  float64
 11  NUMBER OF PERSONS KILLED       318794 non-null  float64
 12  NUMBER OF PEDESTRIANS INJURED  318800 non-null  int64
 13  NUMBER OF PEDESTRIANS KILLED   318800 non-null  int64
 14  NUMBER OF CYCLIST INJURED      318800 non-null  int64
 15  NUMBER OF CYCLIST KILLED       318800 non-null  int64
 16  NUMBER OF MOTORIST INJURED     318800 non-null  int64
 17  NUMBER OF MOTORIST KILLED      318800 non-null  int64
 18  CONTRIBUTING FACTOR VEHICLE 1  317532 non-null  object
 19  CONTRIBUTING FACTOR VEHICLE 2  270646 non-null  object
 20  CONTRIBUTING FACTOR VEHICLE 3  11019 non-null   object
 21  CONTRIBUTING FACTOR VEHICLE 4  2017 non-null    object
 22  CONTRIBUTING FACTOR VEHICLE 5  548 non-null     object
 23  COLLISION_ID                   318800 non-null  int64
 24  VEHICLE TYPE CODE 1            316650 non-null  object
 25  VEHICLE TYPE CODE 2            267161 non-null  object
 26  VEHICLE TYPE CODE 3            10770 non-null   object
 27  VEHICLE TYPE CODE 4            1963 non-null    object
 28  VEHICLE TYPE CODE 5            538 non-null     object
dtypes: float64(4), int64(7), object(18)
memory usage: 73.0+ MB
None
(318800, 29)
---
--- Part 02: Convert dfManhattan dataframe to list ...
--- ---------------------------------------------- ...
---
--- Part 03: Create dictionary of vehicles involved in accidents ...
--- ------------------------------------------------------------ ...

Dictionary: Motor Vehicle Types:
---------------------------------------------------
key: 0 --> value: 2 ...
key: 013 --> value: 1 ...
key: 1 --> value: 1 ...
key: 11 pa --> value: 1 ...
key: 12 pa --> value: 1 ...
key: 2 doo --> value: 3 ...
key: 2 dr sedan --> value: 539 ...
key: 2 hor --> value: 1 ...
key: 2000 --> value: 1 ...
key: 26 ft --> value: 1 ...
key: 2dr --> value: 1 ...
key: 3 whe --> value: 2 ...
key: 3 wheel sc --> value: 1 ...
key: 3-door --> value: 97 ...

... many lines of output removed ...

key: access a r --> value: 1 ...
key: winne --> value: 2 ...
key: work --> value: 2 ...
key: work truck --> value: 2 ...
key: work van --> value: 1 ...
key: workcart --> value: 1 ...
key: workh util --> value: 1 ...
key: x amb --> value: 1 ...
key: yello --> value: 2 ...
key: yellow tax --> value: 1 ...
key: yll p --> value: 1 ...
key: omm --> value: 1 ...
---------------------------------------------------

---
```

```
--- Sorted Motor Vehicle Accidents ...
--- ------------------------------ ...
---
--- Vehicle Type: uspcs                          --> Accident Count:  1 ...
--- Vehicle Type: util truck                     --> Accident Count:  1 ...
--- Vehicle Type: wheelchair                     --> Accident Count:  1 ...
--- Vehicle Type: wh                             --> Accident Count:  1 ...
--- Vehicle Type: fdny rig                       --> Accident Count:  1 ...
--- Vehicle Type: ambulace                       --> Accident Count:  1 ...
--- Vehicle Type: skatboard                      --> Accident Count:  1 ...
--- Vehicle Type: uhal                           --> Accident Count:  1 ...

... many lines of output removed ...

--- Vehicle Type: rv                             --> Accident Count: 12 ...
--- Vehicle Type: horse                          --> Accident Count: 12 ...
--- Vehicle Type: garba                          --> Accident Count: 12 ...
--- Vehicle Type: self                           --> Accident Count: 13 ...
--- Vehicle Type: fdny truck                     --> Accident Count: 13 ...
--- Vehicle Type: pallet                         --> Accident Count: 13 ...
--- Vehicle Type: snow plow                      --> Accident Count: 13 ...
--- Vehicle Type: wagon                          --> Accident Count: 13 ...
--- Vehicle Type: schoo                          --> Accident Count: 13 ...
--- Vehicle Type: refg                           --> Accident Count: 14 ...
--- Vehicle Type: motorized home                 --> Accident Count: 14 ...
--- Vehicle Type: firetruck                      --> Accident Count: 14 ...
--- Vehicle Type: minibike                       --> Accident Count: 14 ...
--- Vehicle Type: utili                          --> Accident Count: 14 ...
--- Vehicle Type: van camper                     --> Accident Count: 15 ...
--- Vehicle Type: tow truck                      --> Accident Count: 15 ...
--- Vehicle Type: usps truck                     --> Accident Count: 16 ...
--- Vehicle Type: us po                          --> Accident Count: 16 ...
--- Vehicle Type: food                           --> Accident Count: 16 ...
--- Vehicle Type: fdny fire                      --> Accident Count: 17 ...
--- Vehicle Type: sanit                          --> Accident Count: 17 ...
--- Vehicle Type: posta                          --> Accident Count: 17 ...
--- Vehicle Type: tow t                          --> Accident Count: 19 ...
--- Vehicle Type: firet                          --> Accident Count: 19 ...
--- Vehicle Type: open body                      --> Accident Count: 20 ...
--- Vehicle Type: util                           --> Accident Count: 21 ...
--- Vehicle Type: pickup with mounted camper     --> Accident Count: 23 ...
--- Vehicle Type: tract                          --> Accident Count: 24 ...
--- Vehicle Type: box t                          --> Accident Count: 24 ...
--- Vehicle Type: e-sco                          --> Accident Count: 26 ...
--- Vehicle Type: trk                            --> Accident Count: 27 ...
--- Vehicle Type: trailer                        --> Accident Count: 27 ...
--- Vehicle Type: ambu                           --> Accident Count: 27 ...
--- Vehicle Type: lunch wagon                    --> Accident Count: 28 ...
--- Vehicle Type: bulk agriculture               --> Accident Count: 33 ...
--- Vehicle Type: comme                          --> Accident Count: 37 ...
--- Vehicle Type: delv                           --> Accident Count: 38 ...
--- Vehicle Type: multi-wheeled vehicle          --> Accident Count: 42 ...
--- Vehicle Type: deliv                          --> Accident Count: 43 ...
--- Vehicle Type: e-bik                          --> Accident Count: 46 ...
--- Vehicle Type: usps                           --> Accident Count: 47 ...
--- Vehicle Type: com                            --> Accident Count: 48 ...
--- Vehicle Type: lift boom                      --> Accident Count: 51 ...
--- Vehicle Type: fdny                           --> Accident Count: 52 ...
--- Vehicle Type: trail                          --> Accident Count: 52 ...
--- Vehicle Type: stake or rack                  --> Accident Count: 56 ...
--- Vehicle Type: unkno                          --> Accident Count: 71 ...
--- Vehicle Type: school bus                     --> Accident Count: 77 ...
--- Vehicle Type: fire                           --> Accident Count: 80 ...
--- Vehicle Type: unk                            --> Accident Count: 95 ...
--- Vehicle Type: 3-door                         --> Accident Count: 97 ...
--- Vehicle Type: limo                           --> Accident Count: 107 ...
--- Vehicle Type: truck                          --> Accident Count: 116 ...
--- Vehicle Type: motorbike                      --> Accident Count: 121 ...
--- Vehicle Type: beverage truck                 --> Accident Count: 128 ...
--- Vehicle Type: flat rack                      --> Accident Count: 144 ...
--- Vehicle Type: concrete mixer                 --> Accident Count: 152 ...
--- Vehicle Type: ambul                          --> Accident Count: 169 ...
--- Vehicle Type: chassis cab                    --> Accident Count: 181 ...
--- Vehicle Type: scooter                        --> Accident Count: 188 ...
--- Vehicle Type: tanker                         --> Accident Count: 190 ...
--- Vehicle Type: armored truck                  --> Accident Count: 197 ...
--- Vehicle Type: pedicab                        --> Accident Count: 241 ...
--- Vehicle Type: refrigerated van               --> Accident Count: 249 ...
--- Vehicle Type: motorscooter                   --> Accident Count: 253 ...
--- Vehicle Type: tractor truck gasoline         --> Accident Count: 332 ...
--- Vehicle Type: tow truck / wrecker            --> Accident Count: 380 ...
--- Vehicle Type: fire truck                     --> Accident Count: 420 ...
--- Vehicle Type: 2 dr sedan                     --> Accident Count: 539 ...
--- Vehicle Type: carry all                      --> Accident Count: 607 ...
--- Vehicle Type: moped                          --> Accident Count: 742 ...
--- Vehicle Type: flat bed                       --> Accident Count: 846 ...
--- Vehicle Type: convertible                    --> Accident Count: 863 ...
--- Vehicle Type: garbage or refuse              --> Accident Count: 1003 ...
--- Vehicle Type: pk                             --> Accident Count: 1018 ...
--- Vehicle Type: e-scooter                      --> Accident Count: 1131 ...
--- Vehicle Type: e-bike                         --> Accident Count: 1261 ...
--- Vehicle Type: dump                           --> Accident Count: 1287 ...
--- Vehicle Type: tractor truck diesel           --> Accident Count: 1562 ...
--- Vehicle Type: ambulance                      --> Accident Count: 2114 ...
--- Vehicle Type: motorcycle                     --> Accident Count: 4054 ...
--- Vehicle Type: bicycle                        --> Accident Count: 6069 ...
--- Vehicle Type: large com veh(6 or more tires) --> Accident Count: 7218 ...
--- Vehicle Type: livery vehicle                 --> Accident Count: 8076 ...
--- Vehicle Type: 4 dr sedan                     --> Accident Count: 9508 ...
--- Vehicle Type: small com veh(4 tires)         --> Accident Count: 9543 ...
--- Vehicle Type: other                          --> Accident Count: 11159 ...
--- Vehicle Type: bike                           --> Accident Count: 11403 ...
--- Vehicle Type: box truck                      --> Accident Count: 13408 ...
--- Vehicle Type: unknown                        --> Accident Count: 15422 ...
--- Vehicle Type: pick-up truck                  --> Accident Count: 15779 ...
--- Vehicle Type: bus                            --> Accident Count: 17535 ...
--- Vehicle Type: van                            --> Accident Count: 21346 ...
--- Vehicle Type: sport utility / station wagon  --> Accident Count: 52151 ...
--- Vehicle Type: station wagon/sport utility vehicle --> Accident Count: 79262 ...
--- Vehicle Type: taxi                           --> Accident Count: 79867 ...
--- Vehicle Type: sedan                          --> Accident Count: 105686 ...
--- Vehicle Type: passenger vehicle              --> Accident Count: 110392 ...
---
--- Part 04: Gather statistics for injuries/fatalities ...
--- ------------------------------------------------------------ ...
---
---        1: (date, time) --> (12/14/2021, 14:58); (lat,long) = (40.751440, -73.973970)
---   GeoLocation: (40.75144, -73.97397) ...
---   On Street Name: 3 AVENUE ...
---   Cross Street Name: EAST 43 STREET ...
---   Accident Cause (Vehicle01): Passing Too Closely ...
---   Accident Cause (Vehicle02): Unspecified ...
---   VehicleType01: Sedan ...
---   VehicleType02: Station Wagon/Sport Utility Vehicle ...
---        2: (date, time) --> (12/11/2021, 4:45 ); (lat,long) = (40.748917, -73.993546)
---   GeoLocation: (40.748917, -73.993546) ...
---   Off Street Name: 232       WEST 30 STREET ...
---   Accident Cause (Vehicle01): Following Too Closely ...
---   Accident Cause (Vehicle02): Unspecified ...
---   VehicleType01: Station Wagon/Sport Utility Vehicle ...

... lines of output removed ...

---   318799: (date, time) --> (01/30/2024, 19:43); (lat,long) = (40.784355, -73.981170)
---   GeoLocation: (40.784355, -73.98117) ...
---   On Street Name: WEST 79 STREET ...
---   Cross Street Name: WEST END AVENUE ...
---   Accident Cause (Vehicle01): Driver Inattention/Distraction ...
---   Accident Cause (Vehicle02): Unspecified ...
---   VehicleType01: Sedan ...
---   VehicleType02: Station Wagon/Sport Utility Vehicle ...
---   318800: (date, time) --> (01/27/2024, 15:36); (lat,long) = (40.872314, -73.912740)
---   GeoLocation: (40.872314, -73.91274) ...
---   On Street Name: BROADWAY ...
---   Cross Street Name: WEST 220 STREET ...
---   Accident Cause (Vehicle01): Aggressive Driving/Road Rage ...
---   Accident Cause (Vehicle02): Unspecified ...
---   VehicleType01: Sedan ...
---   VehicleType02: Sedan ...
---
--- Part 04: Summary of Accidents Statistics ...
--- ---------------------------------------- ...
---
--- Total No Accidents     = 318801 ...

--- No Persons Injured     = 55701.0 ...
--- No Cyclists Injured    = 12506.0 ...
--- No Pedestrians Injured = 20408.0 ...
--- No Motorists Injured   = 22486.0 ...

--- No Persons Killed      = 317 ...
--- No Cyclists Killed     = 37 ...
--- No Pedestrians Killed  = 226 ...
--- No Motorists Killed    = 52 ...
---
```

```
--- Part 05: Assemble heatmap for Pedestrians Injured/Killed ...
--- -------------------------------------------------------- ...
---
--- ================================================================================= ...
---
--- Days count ...
[[103. 114. 156. 143. 195. 293. 298.]
 [ 37.  33.  48.  42.  65. 151. 163.]
 [113. 127. 110. 111. 132.  89.  56.]
 [208. 251. 264. 277. 271. 105.  48.]
 [286. 332. 312. 311. 326. 193. 115.]
 [299. 312. 373. 369. 371. 245. 165.]
 [350. 378. 370. 393. 411. 253. 210.]
 [406. 429. 394. 398. 410. 300. 235.]
 [397. 493. 471. 463. 461. 314. 277.]
 [314. 381. 329. 399. 358. 364. 257.]
 [217. 251. 262. 271. 288. 238. 207.]
 [ 67.  99.  94. 109. 109. 104.  71.]]
--- Total no accidents = 318801 ...
---
--- Part 05: Create dictionary of vehicles involved in accidents ...
--- ------------------------------------------------------------ ...
---
--- =============================================== ...
--- Leave TestMotorVehicleAccidentsNYC03.main()     ...
```

A few points:

1. Too many bicycle/pedestrian accidents occur early in the morning on weekends.
2. Perhaps the main takeaway from this exercise is: the motor vehicle accident data is very messy. Not a lot of care has been given to consistently describing the details of accidents, removing spelling errors, and so forth. We need some sort of AI routine to simplify the data by bundling similar items into groups.

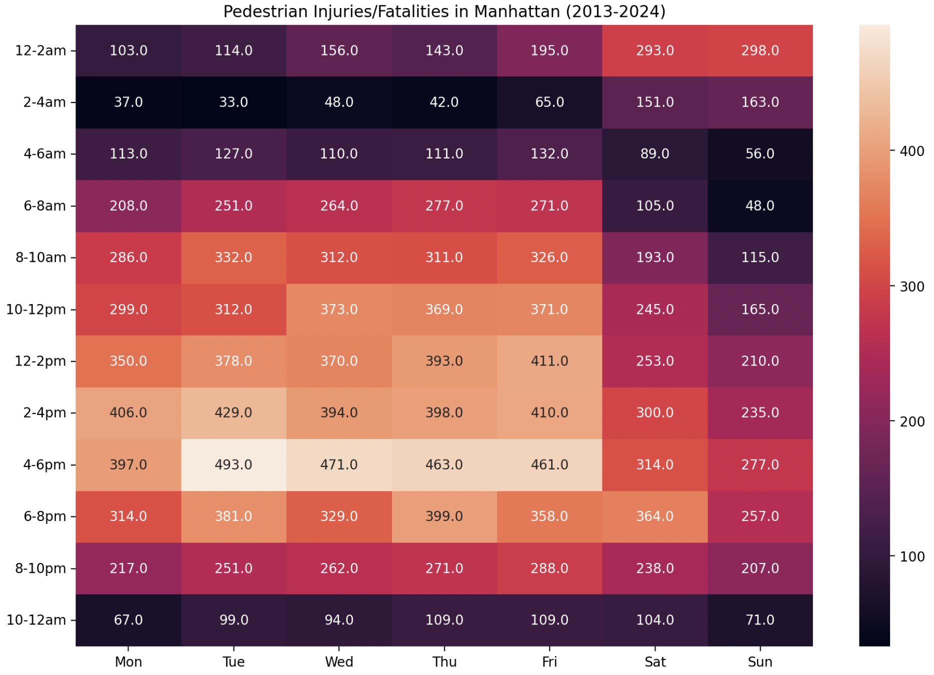

Figure 6: Temporal view of pedestrian injuries/fatalities in Manhattan (2013-2024).

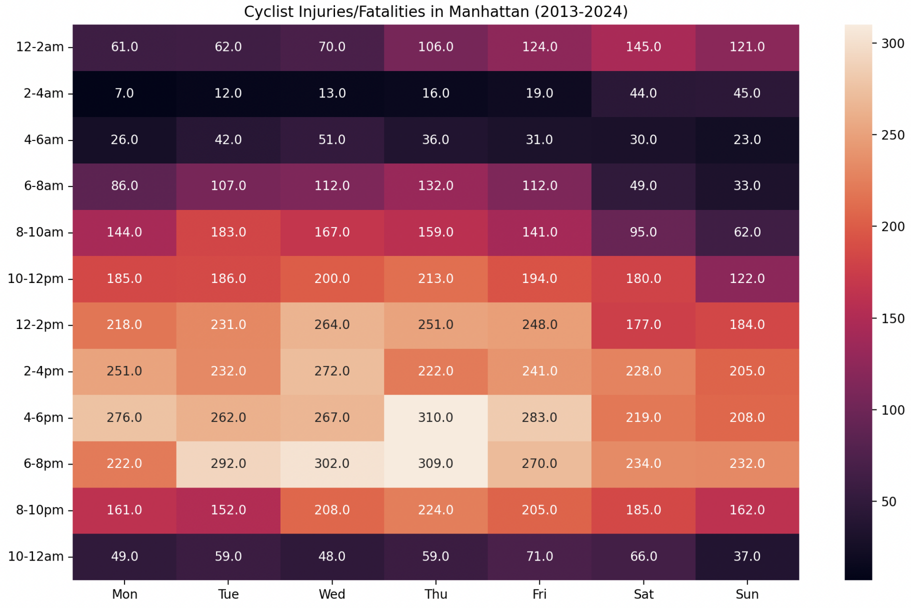

Figure 7: Temporal view of cyclist injuries/fatalities in Manhattan (2013-2024).