Homework Assignment 4 (NYC Crash Data Cleaning)

Ammar Alsadadi

Table of contents

1 Context

1.1 Overview

The Motor Vehicle Collisions dataset contains details on crashes in NYC. Each row represents a crash event reported by the NYPD. The data includes occurrences of injuries, fatalities, and location details. Reports are required when an injury, fatality, or at least $1000 in damage occurs.

1.2 Historical Context

The NYPD launched TrafficStat in 1998 to track fatal incidents. In 1999, the Traffic Accident Management System (TAMS) improved data collection. Vision Zero began in 2014 to reduce fatalities, and in 2016, FORMS replaced TAMS, allowing officers to record crash data electronically.

1.3 Data Dictionary

Column Name	Description	Data Type
CRASH DATE	Date of collision occurrence	Floating Timestamp
CRASH TIME	Time of collision occurrence	Text
BOROUGH	Borough where collision occurred	Text
ZIP CODE	Postal code of incident	Text
LATITUDE	Latitude coordinate	Number
LONGITUDE	Longitude coordinate	Number
LOCATION	Latitude, Longitude pair	Location
ON STREET NAME	Street where the collision occurred	Text
CROSS STREET NAME	Nearest cross street to the collision	Text
OFF STREET NAME	Street address if known	Text
NUMBER OF PERSONS INJURED	Number of persons injured	Number
NUMBER OF PERSONS KILLED	Number of persons killed	Number
NUMBER OF PEDESTRIANS INJURED	Number of pedestrians injured	Number
NUMBER OF PEDESTRIANS KILLED	Number of pedestrians killed	Number
NUMBER OF CYCLISTS INJURED	Number of cyclists injured	Number
NUMBER OF CYCLISTS KILLED	Number of cyclists killed	Number
NUMBER OF MOTORISTS INJURED	Number of motorists injured	Number

Column Name	Description	Data Type
NUMBER OF MOTORISTS KILLED	Number of motorists killed	Number
CONTRIBUTING FACTOR VEHICLE 1	Contributing factor for vehicle 1	Text
CONTRIBUTING FACTOR VEHICLE 2	Contributing factor for vehicle 2	Text
CONTRIBUTING FACTOR VEHICLE 3	Contributing factor for vehicle 3	Text
CONTRIBUTING FACTOR VEHICLE 4	Contributing factor for vehicle 4	Text
CONTRIBUTING FACTOR VEHICLE 5	Contributing factor for vehicle 5	Text
UNIQUE KEY	Unique identifier for each crash event	Text
VEHICLE TYPE CODE 1	Vehicle type code for vehicle 1	Text
VEHICLE TYPE CODE 2	Vehicle type code for vehicle 2	Text
VEHICLE TYPE CODE 3	Vehicle type code for vehicle 3	Text
VEHICLE TYPE CODE 4	Vehicle type code for vehicle 4	Text
VEHICLE TYPE CODE 5	Vehicle type code for vehicle 5	Text

2 Exlploring data

2.1 Exploring

```python
import pandas as pd

# Load the dataset
file_path1 = "C:/Users/Ammar/Downloads/Motor_Vehicle_Collisions_-_Crashes_20250214.csv"

df1 = pd.read_csv(file_path1)

# Display basic info about the dataset
df1_info = df1.info()

# Show first few rows
df1_head = df1.head()

# Check for missing values
missing_values1 = df1.isnull().sum()

df1_info, df1_head, missing_values1
```

```
C:\Users\Ammar\AppData\Local\Temp\ipykernel_38792\2397593892.py:6: DtypeWarning:

Columns (3) have mixed types. Specify dtype option on import or set low_memory=False.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2155718 entries, 0 to 2155717
Data columns (total 29 columns):
 #   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: 477.0+ MB
```

```
(None,
    CRASH DATE CRASH TIME   BOROUGH ZIP CODE  LATITUDE  LONGITUDE  \
 0  09/11/2021       2:39       NaN      NaN       NaN        NaN
```

```
1  03/26/2022      11:45       NaN      NaN       NaN        NaN
2  11/01/2023       1:29  BROOKLYN  11230.0  40.62179 -73.970024
3  06/29/2022       6:55       NaN      NaN       NaN        NaN
4  09/21/2022      13:21       NaN      NaN       NaN        NaN

              LOCATION           ON STREET NAME CROSS STREET NAME  \
0                  NaN    WHITESTONE EXPRESSWAY         20 AVENUE
1                  NaN  QUEENSBORO BRIDGE UPPER               NaN
2  (40.62179, -73.970024)          OCEAN PARKWAY          AVENUE K
3                  NaN       THROGS NECK BRIDGE               NaN
4                  NaN          BROOKLYN BRIDGE               NaN

  OFF STREET NAME  ...  CONTRIBUTING FACTOR VEHICLE 2  \
0             NaN  ...                    Unspecified
1             NaN  ...                            NaN
2             NaN  ...                    Unspecified
3             NaN  ...                    Unspecified
4             NaN  ...                    Unspecified

   CONTRIBUTING FACTOR VEHICLE 3  CONTRIBUTING FACTOR VEHICLE 4  \
0                            NaN                            NaN
1                            NaN                            NaN
2                    Unspecified                            NaN
3                            NaN                            NaN
4                            NaN                            NaN

   CONTRIBUTING FACTOR VEHICLE 5  COLLISION_ID  \
0                            NaN       4455765
1                            NaN       4513547
2                            NaN       4675373
3                            NaN       4541903
4                            NaN       4566131

                 VEHICLE TYPE CODE 1  VEHICLE TYPE CODE 2  \
0                              Sedan                Sedan
1                              Sedan                  NaN
2                              Moped                Sedan
3                              Sedan         Pick-up Truck
4  Station Wagon/Sport Utility Vehicle                  NaN

   VEHICLE TYPE CODE 3 VEHICLE TYPE CODE 4 VEHICLE TYPE CODE 5
0                  NaN                 NaN                 NaN
1                  NaN                 NaN                 NaN
2                Sedan                 NaN                 NaN
3                  NaN                 NaN                 NaN
```

```
4                 NaN                 NaN                 NaN

[5 rows x 29 columns],
CRASH DATE                             0
CRASH TIME                             0
BOROUGH                           667415
ZIP CODE                          667683
LATITUDE                          239663
LONGITUDE                         239663
LOCATION                          239663
ON STREET NAME                    463684
CROSS STREET NAME                 822176
OFF STREET NAME                  1784407
NUMBER OF PERSONS INJURED             18
NUMBER OF PERSONS KILLED              31
NUMBER OF PEDESTRIANS INJURED          0
NUMBER OF PEDESTRIANS KILLED           0
NUMBER OF CYCLIST INJURED              0
NUMBER OF CYCLIST KILLED               0
NUMBER OF MOTORIST INJURED             0
NUMBER OF MOTORIST KILLED              0
CONTRIBUTING FACTOR VEHICLE 1       7382
CONTRIBUTING FACTOR VEHICLE 2     341042
CONTRIBUTING FACTOR VEHICLE 3    2000331
CONTRIBUTING FACTOR VEHICLE 4    2120349
CONTRIBUTING FACTOR VEHICLE 5    2146057
COLLISION_ID                           0
VEHICLE TYPE CODE 1                15087
VEHICLE TYPE CODE 2               423849
VEHICLE TYPE CODE 3              2006218
VEHICLE TYPE CODE 4              2121619
VEHICLE TYPE CODE 5              2146357
dtype: int64)
```

2.1.1 Dataset Overview

- **Total Records**: 2,155,718
- **Total Columns**: 29
- **Data Type Warning**: Mixed types due to inconsistent entries
- **Missing Values:**
  - **BOROUGH**: Around 31% missing
  - **ZIP CODE**: Around 31% missing
  - **VEHICLE TYPE CODE 3-5**: Mostly empty

- **Key Issues Identified:**
  - **Geographical Data**: Significant missing values in boroughs, zip codes, and coordinates
  - **Vehicle & Contributing Factors**: Sparse data for secondary vehicle details
  - **Time Format**: **CRASH DATE** & **CRASH TIME** require conversion to datetime format

3 Solving Questions

3.1 Part A

1. Use the filter on the website to obtain crash data for the week of June 30, 2024, in CSV format.
2. Open a terminal or command prompt and run to create data directory:

   ```
   mkdir data
   ```

3. Navigate to your downloads folder and move the file to `data/`:

   ```
   mv "C:\Users\Ammar\Downloads\Motor_Vehicle_Collisions_-_Crashes_06302024.csv" data/
   ```

4. Rename the file to make it more informative:

   ```
   mv data/Motor_Vehicle_Collisions_-_Crashes_06302024.csv
      data/nyccrashes_2024w0630_by20240916.csv
   ```

5. Commit the data directory to repo:

   ```
   git add data/
   git commit -m "Added data directory"
   git push origin main
   ```

3.2 Part B

Clean up the variable names. Use lower cases and replace spaces with underscores.

Standardizing column names improves data consistency and simplifies manipulation. I convert names to lowercase and replace spaces with underscores for easier access and readability.

```python
import pandas as pd

# Load the dataset
file_path = "C:/Users/Ammar/ids-s25/4-nyc-crash-data-cleaning-CoderAmmar0/data/nyccrashes_2024w06

df = pd.read_csv(file_path)

# Convert column names to lowercase and replace spaces with underscores
df.columns = df.columns.str.lower().str.replace(" ", "_")

# Display cleaned column names
df.columns
```

```
Index(['crash_date', 'crash_time', 'borough', 'zip_code', 'latitude',
       'longitude', 'location', 'on_street_name', 'cross_street_name',
       'off_street_name', 'number_of_persons_injured',
       'number_of_persons_killed', 'number_of_pedestrians_injured',
       'number_of_pedestrians_killed', 'number_of_cyclist_injured',
       'number_of_cyclist_killed', 'number_of_motorist_injured',
       'number_of_motorist_killed', 'contributing_factor_vehicle_1',
       'contributing_factor_vehicle_2', 'contributing_factor_vehicle_3',
       'contributing_factor_vehicle_4', 'contributing_factor_vehicle_5',
       'collision_id', 'vehicle_type_code_1', 'vehicle_type_code_2',
       'vehicle_type_code_3', 'vehicle_type_code_4', 'vehicle_type_code_5'],
      dtype='object')
```

3.3 Part C

Check the crash date and time to see if they really match the filter we intented. Remove the extra rows if needed.

```python
# Checking the unique crash dates
unique_crash_dates = df["crash_date"].unique()

# Display unique crash dates
unique_crash_dates
```

```
array(['06/30/2024', '07/01/2024', '07/02/2024', '07/03/2024',
       '07/04/2024', '07/05/2024', '07/06/2024', '07/07/2024'],
      dtype=object)
```

Seems like the following Sunday is included in the dataset(7/07) so I will remove it.

```python
# Filter out crashes that occurred on 07/07/2024
df = df[df["crash_date"] != "07/07/2024"]

# Checking the unique crash dates again
unique_crash_dates_filtered = df["crash_date"].unique()

# Display the updated unique crash dates
unique_crash_dates_filtered
```

```
array(['06/30/2024', '07/01/2024', '07/02/2024', '07/03/2024',
       '07/04/2024', '07/05/2024', '07/06/2024'], dtype=object)
```

3.4 Part D

Get the basic summaries of each variables: missing percentage; descriptive statistics for continuous variables; frequency tables for discrete variables.

```python
import numpy as np

# Calculate missing percentage for each column
missing_percentage = df.isnull().sum() / len(df) * 100

# Get descriptive statistics for continuous variables
continuous_vars = df.select_dtypes(include=[np.number]).describe()

# Get frequency tables for discrete (categorical) variables
categorical_vars = df.select_dtypes(include=['object'])
frequency_tables = {col: categorical_vars[col].value_counts() for col in categorical_vars.columns

# Create a DataFrame for missing percentages
missing_df = pd.DataFrame(missing_percentage, columns=["missing_percentage"])

# Display the missing percentage summary
print("Missing Percentage Summary:")
print(missing_df)

# Display descriptive statistics for continuous variables
print("\nDescriptive Statistics for Continuous Variables:")
print(continuous_vars)

# Display frequency tables for categorical variables
print("\nFrequency Tables for Categorical Variables:")
for col, freq_table in frequency_tables.items():
    print(f"\n{col}:\n{freq_table}")
```

```
Missing Percentage Summary:
                               missing_percentage
crash_date                               0.000000
crash_time                               0.000000
borough                                 28.477612
zip_code                                28.477612
latitude                                 6.985075
longitude                                6.985075
location                                 6.985075
on_street_name                          28.835821
cross_street_name                       49.194030
off_street_name                         71.164179
number_of_persons_injured                0.000000
number_of_persons_killed                 0.000000
number_of_pedestrians_injured            0.000000
number_of_pedestrians_killed             0.000000
number_of_cyclist_injured                0.000000
number_of_cyclist_killed                 0.000000
number_of_motorist_injured               0.000000
number_of_motorist_killed                0.000000
contributing_factor_vehicle_1            0.477612
contributing_factor_vehicle_2           23.402985
contributing_factor_vehicle_3           90.865672
contributing_factor_vehicle_4           97.253731
contributing_factor_vehicle_5           99.164179
collision_id                             0.000000
vehicle_type_code_1                      1.611940
vehicle_type_code_2                     33.492537
vehicle_type_code_3                     91.522388
vehicle_type_code_4                     97.432836
vehicle_type_code_5                     99.164179

Descriptive Statistics for Continuous Variables:
           zip_code     latitude    longitude  number_of_persons_injured  \
count   1198.000000  1558.000000  1558.000000                1675.000000
mean   10895.911519    40.639880   -73.778445                   0.625075
std      530.386669     1.787455     3.242765                   0.928941
min    10001.000000     0.000000   -74.237366                   0.000000
25%    10456.000000    40.661310   -73.967592                   0.000000
50%    11208.000000    40.712446   -73.924315                   0.000000
75%    11237.000000    40.766748   -73.870863                   1.000000
max    11694.000000    40.907246     0.000000                  11.000000

       number_of_persons_killed  number_of_pedestrians_injured  \
count               1675.000000                    1675.000000
```

```
mean                  0.004776                        0.093134
std                   0.109200                        0.343458
min                   0.000000                        0.000000
25%                   0.000000                        0.000000
50%                   0.000000                        0.000000
75%                   0.000000                        0.000000
max                   4.000000                        7.000000

       number_of_pedestrians_killed  number_of_cyclist_injured  \
count                   1675.000000                1675.000000
mean                       0.002985                   0.067463
std                        0.100729                   0.250897
min                        0.000000                   0.000000
25%                        0.000000                   0.000000
50%                        0.000000                   0.000000
75%                        0.000000                   0.000000
max                        4.000000                   1.000000

       number_of_cyclist_killed  number_of_motorist_injured  \
count                    1675.0                 1675.000000
mean                        0.0                    0.438806
std                         0.0                    0.902829
min                         0.0                    0.000000
25%                         0.0                    0.000000
50%                         0.0                    0.000000
75%                         0.0                    1.000000
max                         0.0                   11.000000

       number_of_motorist_killed  collision_id
count                1675.000000  1.675000e+03
mean                    0.001791  4.738503e+06
std                     0.042295  1.760799e+03
min                     0.000000  4.736561e+06
25%                     0.000000  4.737588e+06
50%                     0.000000  4.738146e+06
75%                     0.000000  4.738778e+06
max                     1.000000  4.765601e+06

Frequency Tables for Categorical Variables:

crash_date:
crash_date
07/03/2024    258
07/05/2024    254
07/02/2024    252
```

```
06/30/2024    249
07/01/2024    247
07/06/2024    212
07/04/2024    203
Name: count, dtype: int64

crash_time:
crash_time
0:00     34
15:00    19
19:00    18
13:00    17
14:30    15
         ..
17:32     1
0:23      1
11:58     1
0:59      1
0:37      1
Name: count, Length: 736, dtype: int64

borough:
borough
BROOKLYN         416
QUEENS           336
MANHATTAN        212
BRONX            189
STATEN ISLAND     45
Name: count, dtype: int64

location:
location
(40.668507, -73.92561)     4
(40.848038, -73.93285)     3
(0.0, 0.0)                 3
(40.753326, -73.8718)      3
(40.669476, -73.919975)    3
                          ..
(40.610367, -73.95438)     1
(40.763042, -73.956535)    1
(40.881073, -73.878494)    1
(40.58038, -73.967606)     1
(40.574085, -73.97672)     1
Name: count, Length: 1493, dtype: int64
```

```
on_street_name:
on_street_name
BELT PARKWAY                  28
FDR DRIVE                     15
LONG ISLAND EXPRESSWAY        14
BROADWAY                      13
BROOKLYN QUEENS EXPRESSWAY    12
                              ..
G.C.P. / LAGUARDIA (CDR)       1
YORK AVENUE                    1
81 STREET                      1
71 PLACE                       1
WEIHER COURT                   1
Name: count, Length: 651, dtype: int64

cross_street_name:
cross_street_name
BROADWAY              16
BRUCKNER BOULEVARD    10
3 AVENUE               8
2 AVENUE               7
SHORE PARKWAY          6
                      ..
EAST 66 STREET         1
WEST 7 STREET          1
BRONXDALE AVENUE       1
ARLINGTON AVENUE       1
WEST 51 STREET         1
Name: count, Length: 604, dtype: int64

off_street_name:
off_street_name
369       HYLAN BOULEVARD      2
1825      EASTCHESTER ROAD     2
2724      UNIVERSITY AVENUE    2
66-50     73 PLACE             1
555       KAPPOCK STREET       1
                              ..
149       WEST 37 STREET       1
88-48     COOPER AVENUE        1
205       CHESTNUT STREET      1
30        KENMARE STREET       1
2951      WEST 8 STREET        1
Name: count, Length: 480, dtype: int64
```

```
contributing_factor_vehicle_1:
contributing_factor_vehicle_1
Unspecified                                                423
Driver Inattention/Distraction                             404
Failure to Yield Right-of-Way                              109
Following Too Closely                                       89
Unsafe Speed                                                74
Passing or Lane Usage Improper                              67
Other Vehicular                                             60
Traffic Control Disregarded                                 51
Alcohol Involvement                                         50
Passing Too Closely                                         50
Backing Unsafely                                            45
Driver Inexperience                                         45
Turning Improperly                                          40
Unsafe Lane Changing                                        22
Pedestrian/Bicyclist/Other Pedestrian Error/Confusion       22
Reaction to Uninvolved Vehicle                              14
Aggressive Driving/Road Rage                                13
View Obstructed/Limited                                     13
Pavement Slippery                                            9
Fell Asleep                                                  8
Oversized Vehicle                                            8
Brakes Defective                                             6
Tire Failure/Inadequate                                      5
Lost Consciousness                                           5
Outside Car Distraction                                      4
Illnes                                                       3
Failure to Keep Right                                        3
Glare                                                        3
Fatigued/Drowsy                                              2
Steering Failure                                             2
Pavement Defective                                           2
Obstruction/Debris                                           2
Accelerator Defective                                        2
Tow Hitch Defective                                          2
Tinted Windows                                               2
Cell Phone (hand-Held)                                       2
Driverless/Runaway Vehicle                                   2
Passenger Distraction                                        2
Cell Phone (hands-free)                                      1
Lane Marking Improper/Inadequate                             1
Name: count, dtype: int64

contributing_factor_vehicle_2:
```

```
contributing_factor_vehicle_2
Unspecified                                                1087
Driver Inattention/Distraction                               63
Other Vehicular                                              26
Unsafe Speed                                                 19
Following Too Closely                                        13
Failure to Yield Right-of-Way                                13
Passing or Lane Usage Improper                               10
Pedestrian/Bicyclist/Other Pedestrian Error/Confusion         9
Traffic Control Disregarded                                   9
Turning Improperly                                            4
Aggressive Driving/Road Rage                                  4
Driver Inexperience                                           4
Passing Too Closely                                           4
Unsafe Lane Changing                                          4
Alcohol Involvement                                           3
View Obstructed/Limited                                       3
Reaction to Uninvolved Vehicle                                2
Backing Unsafely                                              2
Passenger Distraction                                         1
Drugs (illegal)                                               1
Failure to Keep Right                                         1
Fatigued/Drowsy                                               1
Name: count, dtype: int64

contributing_factor_vehicle_3:
contributing_factor_vehicle_3
Unspecified                     147
Other Vehicular                   3
Unsafe Speed                      2
Aggressive Driving/Road Rage      1
Name: count, dtype: int64

contributing_factor_vehicle_4:
contributing_factor_vehicle_4
Unspecified                     45
Aggressive Driving/Road Rage     1
Name: count, dtype: int64

contributing_factor_vehicle_5:
contributing_factor_vehicle_5
Unspecified    14
Name: count, dtype: int64

vehicle_type_code_1:
```

```
vehicle_type_code_1
Sedan                                   769
Station Wagon/Sport Utility Vehicle     570
Taxi                                     49
Bike                                     37
Pick-up Truck                            35
Motorcycle                               24
Box Truck                                23
Bus                                      22
E-Bike                                   14
Moped                                    13
Tractor Truck Diesel                     11
E-Scooter                                11
Ambulance                                10
Van                                       6
Convertible                               5
Dump                                      5
Motorscooter                              4
subn                                      3
Tractor Truck Gasoline                    3
Garbage or Refuse                         3
Flat Bed                                  3
PK                                        3
Chassis Cab                               3
MOPED                                     2
Beverage Truck                            2
Motorbike                                 2
Armored Truck                             2
Flat Rack                                 1
Pedicab                                   1
Tow Truck / Wrecker                       1
MTA BUS                                   1
R/V                                       1
PICK UP                                   1
Multi-Wheeled Vehicle                     1
FDNY FIRE                                 1
U-Haul                                    1
moped                                     1
AMBULANCE                                 1
LIMO                                      1
TRUCK                                     1
UNK                                       1
Name: count, dtype: int64

vehicle_type_code_2:
```

```
vehicle_type_code_2
Sedan                                  447
Station Wagon/Sport Utility Vehicle    319
Bike                                    76
Box Truck                               34
Moped                                   32
Pick-up Truck                           27
E-Scooter                               24
Taxi                                    23
E-Bike                                  22
Bus                                     22
Motorcycle                              20
Tractor Truck Diesel                    13
Van                                      6
Garbage or Refuse                        5
PK                                       4
Chassis Cab                              4
Carry All                                4
Ambulance                                3
Dump                                     3
Motorbike                                3
Unknown                                  2
UNKNOWN                                  2
Convertible                              2
Flat Bed                                 2
PASSENGER                                1
TRUCK                                    1
COURIER VA                               1
3-Door                                   1
UHAUL VAN                                1
Sprinter V                               1
UK                                       1
SCOOTER                                  1
Scooter                                  1
MOPED                                    1
FORKLIFT                                 1
Tow Truck / Wrecker                      1
Power shov                               1
Van Camper                               1
Motorscooter                             1
Name: count, dtype: int64

vehicle_type_code_3:
vehicle_type_code_3
Sedan                                  73
```

```
Station Wagon/Sport Utility Vehicle    51
Pick-up Truck                           5
Bus                                     5
Taxi                                    2
Tractor Truck Diesel                    2
Moped                                   1
Bike                                    1
Van                                     1
Motorcycle                              1
Name: count, dtype: int64

vehicle_type_code_4:
vehicle_type_code_4
Station Wagon/Sport Utility Vehicle    21
Sedan                                  19
Convertible                             1
Pick-up Truck                           1
Motorcycle                              1
Name: count, dtype: int64

vehicle_type_code_5:
vehicle_type_code_5
Sedan                                  9
Station Wagon/Sport Utility Vehicle    3
Bike                                   1
Box Truck                              1
Name: count, dtype: int64
```

Looking at the dataset, I noticed that many columns have missing values, especially for borough (28%), cross street names (49%), and off-street names (71%). There are also a lot of missing entries for contributing factors and vehicle types beyond the second vehicle, which suggests that most crashes involved only one or two vehicles. This could make it harder to analyze multi-vehicle crashes accurately.

In terms of injuries, most crashes didn't result in fatalities, but injuries were fairly common, with motorists being the most affected, followed by pedestrians and cyclists. Brooklyn had the highest number of crashes, and driver inattention/distraction was the most frequently reported cause, though many records simply list "Unspecified." Sedans and SUVs were by far the most common vehicle types involved in crashes, which makes sense given their high presence on NYC roads.

It seems like the dataset has some gaps in reporting, which could affect deeper analysis and limit the accuracy of certain insights.

3.5 Part E

Are their invalid longitude and latitude in the data? If so, replace them with NA.

Lets check for blank and 0 values:

```
# Check for null values in latitude and longitude columns
null_lat_long = df[['latitude', 'longitude']].isnull().sum()

# Check for zero values in latitude and longitude columns
zero_lat_long = (df[['latitude', 'longitude']] == 0).sum()

# Display results
null_lat_long, zero_lat_long
```

```
(latitude     117
 longitude    117
 dtype: int64,
 latitude     3
 longitude    3
 dtype: int64)
```

There is 117 rows with blank values and 3 with values of 0 so lets replace them.

```
# Replace zero values and blank values with "NA"
df.loc[df['latitude'] == 0, 'latitude'] = "NA"
df.loc[df['longitude'] == 0, 'longitude'] = "NA"
df.loc[df['latitude'].isnull(),'latitude'] = "NA"
df.loc[df['longitude'].isnull(), 'longitude'] = "NA"

# Check for the number of "NA" values in latitude and longitude columns
na_lat_long_count = ((df['latitude'] == "NA") | (df['longitude'] == "NA")).sum()

# Display the count of "NA" values
na_lat_long_count
```

```
C:\Users\Ammar\AppData\Local\Temp\ipykernel_38792\241116059.py:2: FutureWarning:

Setting an item of incompatible dtype is deprecated and will raise an error in a future version o

C:\Users\Ammar\AppData\Local\Temp\ipykernel_38792\241116059.py:3: FutureWarning:

Setting an item of incompatible dtype is deprecated and will raise an error in a future version o

120
```

3.6 Part F

Are there zip_code values that are not legit NYC zip codes? If so, replace them with NA

Lets check for blank values:

```
# Check for null values in zip_code column
null_zip_code_count = df['zip_code'].isnull().sum()

# Display the number of null zip code values
null_zip_code_count
```

```
477
```

There is 477 blank values so lets replace them.

```
# Replace null values in zip_code column with "NA"
df.loc[df['zip_code'].isnull(), 'zip_code'] = "NA"

# Check if the null values were replaced by "NA"
na_zip_code_count = (df['zip_code'] == "NA").sum()

# Display the count of "NA"
na_zip_code_count
```

```
C:\Users\Ammar\AppData\Local\Temp\ipykernel_38792\533630329.py:2: FutureWarning:

Setting an item of incompatible dtype is deprecated and will raise an error in a future version o

477
```

3.7 PART G

Are there missing in zip_code and borough? Do they always co-occur?

```
# Check if zip_code is "NA" and borough is null (Since we didn't change it yet to NA)
missing_both_together = (df['zip_code'] == "NA") & df['borough'].isnull()

# Count rows where zip_code is "NA" and borough is null
missing_both_together_count = missing_both_together.sum()

missing_both_together_count
```

```
477
```

They seem to co-occur, lets replace null values with NA for borough:

```
# Replace null values in borough column with "NA"
df.loc[df['borough'].isnull(), 'borough'] = "NA"

# Check if the null values were replaced by "NA"
na_borough_count = (df['borough'] == "NA").sum()

# Display the count of "NA"
na_borough_count
```

```
477
```

3.8 Part H

Are there cases where zip_code and borough are missing but the geo codes are not missing? If so, fill in zip_code and borough using the geo codes.

```
missing_zip_borough_geo_non_missing = ((df['zip_code'] == "NA") &
                                       (df['borough'] == "NA") &
                                       (df['latitude'] != "NA") &
                                       (df['longitude'] != "NA"))

# Count rows where zip_code and borough are "NA" but geo codes are not missing
missing_zip_borough_geo_non_missing_count = missing_zip_borough_geo_non_missing.sum()

# Display the count of such cases
missing_zip_borough_geo_non_missing_count
```

```
370
```

Seems like 370 cases have the geo location values so lets update them.

```
from opencage.geocoder import OpenCageGeocode
import time
# OpenCage API Key
OPENCAGE_API_KEY = "e5faf2c7a6cb40bab9d3f01a57ff8670"
geocoder = OpenCageGeocode(OPENCAGE_API_KEY)

# Ensure latitude and longitude are numeric (convert and handle errors)
def convert_to_float(value):
    try:
        return float(value)
    except ValueError:
```

```python
        return None  # Return None for invalid values

df['latitude'] = df['latitude'].apply(convert_to_float)
df['longitude'] = df['longitude'].apply(convert_to_float)

# Function for reverse geocoding using only "county" and mapping it to borough names
def get_location_info(lat, lon, retries=3, delay=2):
    if lat is None or lon is None:  # Skip if invalid coordinates
        return "NA", "NA"

    for _ in range(retries):
        try:
            result = geocoder.reverse_geocode(lat, lon)
            if result:
                components = result[0]["components"]
                zip_code = components.get("postcode", "NA")

                # Extract county from OpenCage response
                county = components.get("county", "NA")

                # Map county names to boroughs
                borough_map = {
                    "New York County": "Manhattan",
                    "Kings County": "Brooklyn",
                    "Bronx County": "Bronx",
                    "Queens County": "Queens",
                    "Richmond County": "Staten Island"
                }
                borough = borough_map.get(county, county)  # Convert county to borough name if ap

                return borough, zip_code
        except Exception as e:
            print(f"Error: {e}, retrying...")
            time.sleep(delay)

    return "NA", "NA"

# Find rows where ZIP code and borough are missing but latitude/longitude are available
missing_geo = (df['zip_code'] == "NA") & (df['borough'] == "NA") & df['latitude'].notnull() & df[

# Apply reverse geocoding
for index, row in df.loc[missing_geo].iterrows():
    borough, zip_code = get_location_info(row['latitude'], row['longitude'])
    df.at[index, 'borough'] = borough  # Store mapped borough
    df.at[index, 'zip_code'] = zip_code
```

```python
#Check the data if changed
df.head(20)
```

	crash_date	crash_time	borough	zip_code	latitude	longitude	location	on_
0	06/30/2024	23:17	BRONX	10460.0	40.838844	-73.878170	(40.838844, -73.87817)	EA
1	06/30/2024	8:30	BRONX	10468.0	40.862732	-73.903330	(40.862732, -73.90333)	WE
2	06/30/2024	20:47	Manhattan	10019	40.763630	-73.953300	(40.76363, -73.9533)	FD
3	06/30/2024	13:10	BROOKLYN	11234.0	40.617030	-73.919890	(40.61703, -73.91989)	EA
4	06/30/2024	16:42	NA	NA	NaN	NaN	NaN	33
5	06/30/2024	13:40	QUEENS	11379.0	40.713800	-73.880165	(40.7138, -73.880165)	Na
6	06/30/2024	21:03	BRONX	10458.0	40.859875	-73.893230	(40.859875, -73.89323)	EA
7	06/30/2024	12:15	BROOKLYN	11239.0	40.644040	-73.877525	(40.64404, -73.877525)	SE
8	06/30/2024	0:51	Brooklyn	11252	40.604850	-74.025590	(40.60485, -74.02559)	BE
9	06/30/2024	1:36	QUEENS	11374.0	40.730930	-73.864586	(40.73093, -73.864586)	QU
10	06/30/2024	16:00	QUEENS	11362.0	40.760128	-73.731590	(40.760128, -73.73159)	Na
11	06/30/2024	0:35	BROOKLYN	11213.0	40.669270	-73.939995	(40.66927, -73.939995)	Na
12	06/30/2024	23:21	QUEENS	11420.0	40.680496	-73.821365	(40.680496, -73.821365)	LE
13	06/30/2024	8:08	MANHATTAN	10013.0	40.723747	-74.006120	(40.723747, -74.00612)	VA
14	06/30/2024	1:10	QUEENS	11354.0	40.775110	-73.844505	(40.77511, -73.844505)	Na
15	06/30/2024	10:00	BROOKLYN	11207.0	40.677470	-73.898766	(40.67747, -73.898766)	Na
16	06/30/2024	20:36	BROOKLYN	11208.0	40.674652	-73.877120	(40.674652, -73.87712)	MI
17	06/30/2024	7:17	QUEENS	11367.0	40.726048	-73.824005	(40.726048, -73.824005)	141
18	06/30/2024	17:30	NA	NA	NaN	NaN	(0.0, 0.0)	Na
19	06/30/2024	16:42	Staten Island	10314	40.581764	-74.156060	(40.581764, -74.15606)	Na

3.8.1 Filling Missing Boroughs and ZIP Codes

I initially tried filling missing boroughs and ZIP codes using latitude and longitude by matching existing values within the dataset or using a geocoding API. However, multiple challenges made this more complex than expected.

3.8.2 First Attempt: Dataset Mapping

I rounded latitude and longitude to two decimal places and used the most frequent ZIP code and borough for each coordinate pair. While this worked in some cases, issues arose:

- Some lat/lon points mapped to multiple ZIP codes, causing incorrect assignments.
- Certain locations were missing entirely in the dataset.
- The most frequent ZIP code wasn't always accurate.

3.8.3 Second Attempt: Nominatim API

I then used the Nominatim API (OpenStreetMap's geocoder), but it had issues:

- Slow response times and frequent timeouts.
- Rate limits blocked requests after a few queries.

3.8.4 Third Attempt: OpenCage Geocoder

OpenCage was more stable, but it labeled boroughs as counties, returning only "New York" for many locations. However, I found that boroughs were stored under "county," which led to the final solution.

3.8.5 Final Fix: Using County Data

I extracted the "county" field and mapped it to boroughs:

- New York County → Manhattan
- Kings County → Brooklyn
- Bronx County → Bronx
- Queens County → Queens
- Richmond County → Staten Island

This method successfully assigned missing boroughs and ZIP codes accurately.

3.8.6 Lessons Learned

- Internal dataset mapping can help but has limitations.
- API responses vary in format, requiring careful inspection.
- Rate limits and timeouts need handling when using geocoders.
- Sometimes, useful data exists in unexpected fields, like "county" instead of "borough."

Despite unexpected challenges, this approach provided a reliable way to fill in missing boroughs and ZIP codes.

3.9 Part I

Is it redundant to keep both location and the longitude/latitude at the NYC Open Data server?

Storing both location and latitude/longitude can be redundant since location is derived from the other two. However, keeping it improves usability for different users and tools. Removing location would save space, but retaining it adds convenience for analysis and accessibility.

3.10 Part J

Check the frequency of crash_time by hour. Is there a matter of bad luck at exactly midnight? How would you interpret this?

```
# Convert crash_time to datetime format to extract hour
df['crash_time'] = pd.to_datetime(df['crash_time'], format='%H:%M', errors='coerce')

# Extract hour from crash_time
df['crash_hour'] = df['crash_time'].dt.hour

# Count occurrences of crashes by hour
crash_hour_counts = df['crash_hour'].value_counts().sort_index()

crash_hour_counts
```

```
crash_hour
0     104
1      55
2      57
3      36
4      40
5      37
6      37
7      47
8      65
9      52
10     59
11     61
12     72
13     94
14     97
15     92
16    101
17     99
18     86
19     96
20     68
21     81
22     78
23     61
Name: count, dtype: int64
```

The spike in crashes at midnight (00:00) is likely due to fatigue, nightlife- related driving, shift changes, and potential data entry defaults. The increase around 16:00 (4 PM) may be linked to the afternoon rush hour and workers heading home. Both times indicate higher-risk driving conditions.

3.11 Part K

Are the number of persons killed/injured the summation of the numbers of pedestrians, cyclist, and motorists killed/injured? If so, is it redundant to keep these two columns at the NYC Open Data server?

```python
# Check if the total number of persons killed is the sum of pedestrians, cyclists, and motorists
df["calculated_persons_killed"] = (
    df["number_of_pedestrians_killed"].fillna(0) +
    df["number_of_cyclist_killed"].fillna(0) +
    df["number_of_motorist_killed"].fillna(0)
)

df["calculated_persons_injured"] = (
    df["number_of_pedestrians_injured"].fillna(0) +
    df["number_of_cyclist_injured"].fillna(0) +
    df["number_of_motorist_injured"].fillna(0)
)

# Check if the values match the provided total persons killed/injured columns
killed_match = (df["calculated_persons_killed"] == df["number_of_persons_killed"]).all()
injured_match = (df["calculated_persons_injured"] == df["number_of_persons_injured"]).all()

killed_match, injured_match
```

```
(True, False)
```

The "persons killed" column matches the sum of pedestrians, cyclists, and motorists killed, but the "persons injured" column does not fully align. This may indicate data inconsistencies or additional factors. NYC Open Data may retain these columns for validation, easier aggregation, or quick access to totals. The mismatch in injuries suggests both columns provide useful information.

3.12 Part L

Print the whole frequency table of contributing_factor_vehicle_1. Convert lower cases to uppercases and check the frequencies again.

```python
# Print the entire frequency table
print(df["contributing_factor_vehicle_1"].value_counts().to_string())
```

```
contributing_factor_vehicle_1
Unspecified                                              423
Driver Inattention/Distraction                           404
Failure to Yield Right-of-Way                            109
Following Too Closely                                     89
Unsafe Speed                                              74
Passing or Lane Usage Improper                            67
Other Vehicular                                           60
Traffic Control Disregarded                               51
Alcohol Involvement                                       50
Passing Too Closely                                       50
Backing Unsafely                                          45
Driver Inexperience                                       45
Turning Improperly                                        40
Unsafe Lane Changing                                      22
Pedestrian/Bicyclist/Other Pedestrian Error/Confusion     22
Reaction to Uninvolved Vehicle                            14
Aggressive Driving/Road Rage                              13
View Obstructed/Limited                                   13
Pavement Slippery                                          9
Fell Asleep                                                8
Oversized Vehicle                                          8
Brakes Defective                                           6
Tire Failure/Inadequate                                    5
Lost Consciousness                                         5
Outside Car Distraction                                    4
Illnes                                                     3
Failure to Keep Right                                      3
Glare                                                      3
Fatigued/Drowsy                                            2
Steering Failure                                           2
Pavement Defective                                         2
Obstruction/Debris                                         2
Accelerator Defective                                      2
Tow Hitch Defective                                        2
Tinted Windows                                             2
Cell Phone (hand-Held)                                     2
Driverless/Runaway Vehicle                                 2
Passenger Distraction                                      2
Cell Phone (hands-free)                                    1
Lane Marking Improper/Inadequate                           1
```

Now changing to Uppercase:

```
# Convert values in "contributing_factor_vehicle_1" to uppercase
```

```python
df["contributing_factor_vehicle_1"] = df["contributing_factor_vehicle_1"].astype(str).str.upper()

# Print the entire frequency table
print(df["contributing_factor_vehicle_1"].value_counts().to_string())
```

```
contributing_factor_vehicle_1
UNSPECIFIED                                             423
DRIVER INATTENTION/DISTRACTION                          404
FAILURE TO YIELD RIGHT-OF-WAY                           109
FOLLOWING TOO CLOSELY                                    89
UNSAFE SPEED                                             74
PASSING OR LANE USAGE IMPROPER                           67
OTHER VEHICULAR                                          60
TRAFFIC CONTROL DISREGARDED                              51
ALCOHOL INVOLVEMENT                                      50
PASSING TOO CLOSELY                                      50
BACKING UNSAFELY                                         45
DRIVER INEXPERIENCE                                      45
TURNING IMPROPERLY                                       40
UNSAFE LANE CHANGING                                     22
PEDESTRIAN/BICYCLIST/OTHER PEDESTRIAN ERROR/CONFUSION    22
REACTION TO UNINVOLVED VEHICLE                           14
AGGRESSIVE DRIVING/ROAD RAGE                             13
VIEW OBSTRUCTED/LIMITED                                  13
PAVEMENT SLIPPERY                                         9
FELL ASLEEP                                               8
OVERSIZED VEHICLE                                         8
NAN                                                       8
BRAKES DEFECTIVE                                          6
TIRE FAILURE/INADEQUATE                                   5
LOST CONSCIOUSNESS                                        5
OUTSIDE CAR DISTRACTION                                   4
ILLNES                                                    3
FAILURE TO KEEP RIGHT                                     3
GLARE                                                     3
FATIGUED/DROWSY                                           2
STEERING FAILURE                                          2
PAVEMENT DEFECTIVE                                        2
OBSTRUCTION/DEBRIS                                        2
ACCELERATOR DEFECTIVE                                     2
TOW HITCH DEFECTIVE                                       2
TINTED WINDOWS                                            2
CELL PHONE (HAND-HELD)                                    2
DRIVERLESS/RUNAWAY VEHICLE                                2
PASSENGER DISTRACTION                                     2
```

```
CELL PHONE (HANDS-FREE)                                             1
LANE MARKING IMPROPER/INADEQUATE                                    1
```

The most common contributing factor in crashes is "UNSPECIFIED" (423 cases), followed by "DRIVER INATTENTION/DISTRACTION" (404 cases) and "FAILURE TO YIELD RIGHT-OF-WAY" (109 cases). This suggests that inattentiveness and failure to follow right-of-way rules are significant contributors to accidents, while many cases lack specific attributions.

3.13 Part M

Provided an opportunity to meet the data provider, what suggestions would you make based on your data exploration experience? If I had the chance to meet the data provider, I would recommend:

- Reducing Missing or Unspecified Data: Encouraging detailed reporting and refining data collection methods could improve accuracy.
- Clarifying Injury Data: Investigating why total injuries don't fully match pedestrian, cyclist, and motorist counts could enhance reliability.
- Standardizing Formatting: Enforcing uniform capitalization and structured input methods would improve consistency in categorical data.
- Improving Location Accuracy: Addressing missing or incorrect latitude, longitude, and borough data would benefit spatial analysis.
- Providing Context on Data Collection: Clear documentation on how factors like injuries and contributing causes are determined would help analysts interpret data correctly.

These enhancements could improve data quality, making it more valuable for policy decisions and public safety initiatives.