Data Wrangling Project: NYC Motor Vehicle Collisions Dataset

Dataset Description & Source: For this project, use the **NYC Motor Vehicle Collisions** dataset from NYC Open Data [1]. This is a large, real-world CSV file (hundreds of MB) containing every police-reported traffic crash in New York City over many years (millions of rows). Each record includes date/time, location (borough, zip code, latitude/longitude), contributing factors, and counts of people injured or killed. The dataset has **29 columns** and (as of mid-2018) about **1.25 million rows** [2]. It is intentionally messy: many fields have missing entries (for example, some records lack ZIP code or contributing-factor data) [3], data types vary (dates, floats, ints, text), and text fields are inconsistent (varying case, extra spaces, "Unspecified" vs blank, etc.).

Download Instructions: Download the CSV from NYC's Open Data portal (Data.gov listing "Motor Vehicle Collisions – Crashes" [1]). For example, use the *Data.gov* preview/download link or the Socrata API (`https://data.cityofnewyork.us/api/views/h9gi-nx95/rows.csv?accessType=DOWNLOAD`). (You may need to sign up/log in to NYC Open Data or use the Kaggle mirror if preferred.) Ensure your environment can handle a few hundred MB; you may optionally work on a subset (e.g. a single year) for testing, but full analysis should run on a standard PC.

Project Tasks: Perform the following data wrangling steps using Pandas (and NumPy as needed). Structure your code into clear sections or functions. Make sure to **comment your code** and **report findings**.

- **1. Inspect the Data:** Load the CSV into a Pandas DataFrame (`pd.read_csv`). Display its shape, columns, and a few sample rows (`df.head()`), and use `df.info()` / `df.describe()` to summarize types and stats. Explain each column's meaning (e.g. "`CRASH_DATE` is the date of the collision"). Identify which columns are numerical, categorical (text), or dates.

- **2. Handle Missing Values:** Check for nulls (`df.isnull().sum()`) and review how many values are missing in each column. Note especially columns like `ZIP CODE` or `CONTRIBUTING FACTOR` that may have many blanks [3]. Decide on a strategy: for example, you might drop rows missing critical info (e.g. no borough or no coordinates), or fill missing numeric counts with 0, or treat missing categories explicitly (e.g. replace empty strings with "Unknown"). Document your reasoning and apply it with Pandas methods (`dropna`, `fillna`, or conditional filtering).

- **3. Correct Data Types:** Ensure each column has an appropriate dtype. Parse date/time columns into `datetime64` (e.g. `pd.to_datetime` on `CRASH_DATE` and/or `CRASH_TIME`). Numeric columns (like `NUMBER OF PERSONS INJURED`) may have been read as floats if NaNs were present; convert them to integers if appropriate (e.g. `df[col] = df[col].astype('Int64')`). Convert true/false flags or small categorical columns to `category` dtype if useful. Verify with `df.dtypes` that types make sense.

- **4. Clean Text/Categorical Data:** Clean up string fields to standardize values. For example, strip leading/trailing spaces and convert borough names and factor descriptions to consistent case (`str.strip().str.title()` or `.str.upper()`). Fix obvious anomalies (e.g. change "0" to missing, "Unspecified" to NaN, or correct known typos). If needed, combine similar categories (e.g. multiple ways of naming the same factor). Show examples of before/after for a couple of fields.

- **5. Filter and Group Data:** Demonstrate selection and grouping operations. For instance, filter the data by date range or borough (e.g. select only collisions in Manhattan), and explain the subset size. Use `groupby` to compute aggregates: for example, group by `BOROUGH` and compute the total number of collisions or total injured. Group by time (year or month) to show trends (e.g. collisions per year or per month). Include a pivot table or group results (e.g. counts of collisions by borough and year). You might also compute a correlation matrix of numeric fields (e.g. correlate `NUMBER OF PERSONS INJURED` with `NUMBER OF PEDESTRIANS INJURED`), using Pandas or NumPy (`np.corrcoef`).

- **6. Statistical Summaries:** Provide basic statistics of the (cleaned) dataset. For numerical columns compute mean, median, min/max, etc. For categorical columns, show the top frequencies (`value_counts()`). For example, report the average injuries per collision, or which borough has the most crashes. Compute one or two simple derived statistics (e.g. percentage of crashes with any injuries). (No plotting is needed—just textual or tabular summaries.)

- **7. Export Raw and Cleaned Data:** Finally, serialize the data. First, save the **raw (initially loaded)** DataFrame to JSON and Parquet:

```
df.to_json('collisions_raw.json', orient='records', lines=True)
df.to_parquet('collisions_raw.parquet')
```

  Then, save your **cleaned/filtered** DataFrame similarly (e.g. `collisions_clean.json`, `collisions_clean.parquet`). Pandas makes this easy: `DataFrame.to_json()` and `DataFrame.to_parquet()` will write the DataFrame to those formats [4] [5]. (Choose an orientation/option as appropriate; using `orient='records'` and line-delimited JSON or default Parquet is fine.) Ensure these files include all records and can be read back by Pandas.

Throughout your report, **cite any assumptions or external references**. For example, note that the NYC collisions dataset comes from the NYC Open Data portal [1]. Include brief commentary with each code section explaining what you did (e.g. "Cleaned the `BOROUGH` field by capitalizing all entries"). The final write-up should read like a clear instruction/answer key for each bullet above.

Dataset Source: NYC Open Data portal – *"Motor Vehicle Collisions – Crashes"* [1] (CSV download via Data.gov or Socrata API). Instructions: Go to the Data.gov catalog link, select **Comma Separated Values**, and download the file. Cite example: the dataset description notes it contains **all NYC traffic collision records** [1].

References: Official dataset description [1]; example analysis confirming size and messiness of this data [2] [3]; Pandas documentation for JSON/Parquet export [4] [5]. These will help you understand the data and available tools. Good luck!

---

[1] Motor Vehicle Collisions - Crashes - Catalog
https://catalog.data.gov/dataset/motor-vehicle-collisions-crashes

[2] [3] NYC Motor Vehicle Collisions
https://thomasnilsson.github.io/02806/PROJECT/explainer.html

[4] pandas.DataFrame.to_json — pandas 2.3.1 documentation
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_json.html

[5] pandas.DataFrame.to_parquet — pandas 2.3.1 documentation
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.to_parquet.html