NYC crash feather format

AUTHOR
Wilson Tang

1 Read data in single precision

▼ Code

```
import numpy as np
import pandas as pd

file_path = 'data/nyc_crashes_lbdwk_2025.csv'
df = pd.read_csv(file_path,
                 dtype={'LATITUDE': np.float32,
                        'LONGITUDE': np.float32,
                        'ZIP CODE': str})
```

2 change variables to be lower case and replace space with underscore

▼ Code

```
df.columns = df.columns.str.lower().str.replace(' ', '_')

df.head()
```

	crash_date	crash_time	borough	zip_code	latitude	longitude	location	on_street_name	cross_street_r
0	08/31/2025	12:49	QUEENS	11101	40.753113	-73.933701	(40.753113, -73.9337)	30 ST	39 AVE
1	08/31/2025	15:30	MANHATTAN	10022	40.760601	-73.964317	(40.7606, -73.96432)	E 59 ST	2 AVE
2	08/31/2025	19:00	NaN	NaN	40.734234	-73.722748	(40.734234, -73.72275)	CROSS ISLAND PARKWAY	HILLSIDE AVEN
3	08/31/2025	1:19	BROOKLYN	11220	40.648075	-74.007034	(40.648075, -74.007034)	NaN	NaN
4	08/31/2025	2:41	MANHATTAN	10036	40.756561	-73.986107	(40.75656, -73.98611)	W 43 ST	BROADWAY

5 rows × 29 columns

3 remove September 7th from the data

▼ Code

```
df = df[df["crash_date"] != "09/07/2025"].copy()
df["crash_date"].unique()
```

```
array(['08/31/2025', '09/01/2025', '09/02/2025', '09/03/2025',
       '09/04/2025', '09/05/2025', '09/06/2025'], dtype=object)
```

4 set missing and 0 values of longitude and latitude to NA

▼ Code

```
df.loc[(df['latitude'] == 0) | (df['latitude'].isna()), 'latitude'] = pd.NA
df.loc[(df['longitude'] == 0) | (df['longitude'].isna()), 'longitude'] = pd.NA

print("Latitude with NA:", df['latitude'].isna().sum())
print("Longitude with NA:", df['longitude'].isna().sum())
```

```
Latitude with NA: 36
Longitude with NA: 36
```

5 set anything that isn't a valid zip code to 0 with NA

▼ Code

```
valid_zip_codes = {
    10463, 10471, 10466, 10469, 10470, 10475, 10458, 10467, 10468,
    10461, 10462, 10464, 10465, 10472, 10473, 10453, 10457, 10460,
    10451, 10452, 10456, 10454, 10455, 10459, 10474, 11211, 11222,
    11201, 11205, 11215, 11217, 11231, 11213, 11212, 11216, 11233,
    11238, 11207, 11208, 11220, 11232, 11204, 11218, 11219, 11230,
    11203, 11210, 11225, 11226, 11234, 11236, 11239, 11209, 11214,
    11228, 11223, 11224, 11229, 11235, 11206, 11221, 11237, 10031,
    10032, 10033, 10034, 10040, 10026, 10027, 10030, 10037, 10039,
    10029, 10035, 10023, 10024, 10025, 10021, 10028, 10044, 10128,
    10001, 10011, 10018, 10019, 10020, 10036, 10010, 10016, 10017,
    10022, 10012, 10013, 10014, 10002, 10003, 10009, 10004, 10005,
    10006, 10007, 10038, 10280, 11101, 11102, 11103, 11104, 11105,
    11106, 11368, 11369, 11370, 11372, 11373, 11377, 11378, 11354,
    11355, 11356, 11357, 11358, 11359, 11360, 11361, 11362, 11363,
    11364, 11374, 11375, 11379, 11385, 11365, 11366, 11367, 11414,
    11415, 11416, 11417, 11418, 11419, 11420, 11421, 11412, 11423,
    11432, 11433, 11434, 11435, 11436, 11004, 11005, 11411, 11413,
```

```
    11422, 11426, 11427, 11428, 11429, 11691, 11692, 11693, 11694,
    11695, 11697, 10302, 10303, 10310, 10301, 10304, 10305, 10314,
    10306, 10307, 10308, 10309, 10312
}

df.loc[~df['zip_code'].isin([str(z) for z in valid_zip_codes]), 'zip_code'] = pd.NA

print("Number of NA zip codes:", df['zip_code'].isna().sum())
```

```
Number of NA zip codes: 272
```

6 fill any non valid borough with NA

▼ Code

```
valid_boroughs = {"BRONX", "BROOKLYN", "MANHATTAN", "QUEENS", "STATEN ISLAND"}

df['borough'] = df['borough'].where(df['borough'].isin(valid_boroughs), pd.NA)
print("Number of NA boroughs:", df['borough'].isna().sum())
```

```
Number of NA boroughs: 263
```

7 store a column where geocode isn't missing but zipcode or borough isn't

▼ Code

```
df["was_fillable"] = (
    (df["zip_code"].isna() | df["borough"].isna()) &
    df["latitude"].notna() &
    df["longitude"].notna()
)
```

8 fill missing zip and borough but has geocode with geocode

▼ Code

```
from geopy.geocoders import Nominatim
import time

geolocator = Nominatim(user_agent="nyc_zip_borough")

def get_zip_code(lat, lon):
    try:
        location = geolocator.reverse((lat, lon), timeout=10)
```

```
        if location:
            return location.raw["address"].get("postcode")
    except Exception as e:
        print(f"Error: {e} at {lat}, {lon}")
    finally:
        time.sleep(1)
    return None

fillable = df[
    df["zip_code"].isna() &
    df["borough"].isna() &
    df["latitude"].notna() &
    df["longitude"].notna()
]

cache = {}
for idx, row in fillable.iterrows():
    coords = (row["latitude"], row["longitude"])
    if coords in cache:
        zc = cache[coords]
    else:
        zc = get_zip_code(*coords)
        cache[coords] = zc
    df.loc[idx, "zip_code"] = zc

df["zip_code_numeric"] = pd.to_numeric(df["zip_code"], errors="coerce")

df.loc[df["zip_code_numeric"].between(10001, 10282), "borough"] = "MANHATTAN"
df.loc[df["zip_code_numeric"].between(10301, 10314), "borough"] = "STATEN ISLAND"
df.loc[df["zip_code_numeric"].between(10451, 10475), "borough"] = "BRONX"
df.loc[
    df["zip_code_numeric"].between(11004, 11009) |
    df["zip_code_numeric"].between(11351, 11697),
    "borough"
] = "QUEENS"
df.loc[df["zip_code_numeric"].between(11201, 11256), "borough"] = "BROOKLYN"

print("Remaining missing ZIP codes:", df["zip_code"].isna().sum())
print("Remaining missing Boroughs:", df["borough"].isna().sum())
```

```
Remaining missing ZIP codes: 20
Remaining missing Boroughs: 28
```

9 Check boroughs after filling with geocodes

▼ Code

```
print(df['borough'].value_counts(dropna=False))
```

```
borough
BROOKLYN         474
QUEENS           396
MANHATTAN        229
BRONX            190
STATEN ISLAND     64
<NA>              28
Name: count, dtype: int64
```

10 check zip codes after filling with geocodes

▼ Code

```
valid_zip_str = {str(z) for z in valid_zip_codes}

invalid_zip_rows = df[~df['zip_code'].isin(valid_zip_str) & df['zip_code'].notna()]

if invalid_zip_rows.empty:
    print("All non-missing ZIP codes are valid.")
else:
    print(f"Found {invalid_zip_rows.shape[0]} rows with invalid ZIP codes.")
    print("Unique invalid ZIP codes:", invalid_zip_rows['zip_code'].unique())
```

```
Found 21 rows with invalid ZIP codes.
Unique invalid ZIP codes: ['11040' '10129' '10311' '11251' '11252' '11430' '10162' '10158'
'11020'
 '10115' '11249']
```

11 clean the remaining zip codes

▼ Code

```
zip_overrides = {
    "10115": "MANHATTAN",
    "10129": "MANHATTAN",
    "10158": "MANHATTAN",
    "10162": "MANHATTAN",
    "10311": "STATEN ISLAND",
    "11249": "BROOKLYN",
    "11251": "BROOKLYN",
    "11252": "BROOKLYN",
    "11430": "QUEENS",
    "11101": "QUEENS",
    "11102": "QUEENS",
    "11104": "QUEENS",
    "11105": "QUEENS",
    # Nassau zips outside NYC → mark NA
```

```
    "11040": pd.NA,
    "11020": pd.NA,
}

df["borough"] = df.apply(
    lambda r: zip_overrides.get(r["zip_code"], r["borough"]),
    axis=1
)

print("Still missing boroughs:", df["borough"].isna().sum())
print("Unique zips left unmapped:", df.loc[df["borough"].isna(), "zip_code"].unique())
```

```
Still missing boroughs: 14
Unique zips left unmapped: ['11040' <NA> '11020']
```

12 check if all rows that have geocode has zip/borough

▼ Code

```
missing_rows = df[
    df["latitude"].notna() &
    df["longitude"].notna() &
    (df["zip_code"].isna() | df["borough"].isna())
]

print("Total rows:", df.shape[0])
print("Rows with lat/lon but missing zip or borough:", missing_rows.shape[0])
print("Sample of missing rows:")
print(missing_rows.head(10))
```

```
Total rows: 1381
Rows with lat/lon but missing zip or borough: 7
Sample of missing rows:
      crash_date crash_time    borough zip_code   latitude  longitude  \
2     08/31/2025      19:00       <NA>    11040  40.734234 -73.722748
233   09/01/2025      16:00  MANHATTAN     <NA>  40.761856 -73.963425
657   09/03/2025      12:00  MANHATTAN     <NA>  40.768646 -73.969826
893   09/04/2025       7:39       <NA>    11020  40.764267 -73.722946
989   09/04/2025      19:50  MANHATTAN     <NA>  40.772072 -73.949936
1235  09/06/2025       9:20       <NA>    11020  40.764267 -73.722946
1303  09/06/2025       3:17  MANHATTAN     <NA>  40.766609 -73.964989

                     location          on_street_name cross_street_name  \
2      (40.734234, -73.72275)    CROSS ISLAND PARKWAY   HILLSIDE AVENUE
233   (40.761856, -73.963425)                   2 AVE           E 61 ST
657   (40.768646, -73.969826)                   5 AVE           E 66 ST
893   (40.764267, -73.722946)  LONG ISLAND EXPRESSWAY              NaN
989    (40.77207, -73.949936)                YORK AVE           E 80 ST
```

```
1235  (40.764267, -73.722946)  LONG ISLAND EXPRESSWAY              NaN
1303    (40.76661, -73.96499)                E 66 ST     LEXINGTON AVE

     off_street_name  ...  contributing_factor_vehicle_4  \
2                NaN  ...                           NaN
233              NaN  ...                           NaN
657              NaN  ...                           NaN
893              NaN  ...                           NaN
989              NaN  ...                           NaN
1235             NaN  ...                           NaN
1303             NaN  ...                           NaN

      contributing_factor_vehicle_5  collision_id  \
2                               NaN       4838966
233                             NaN       4839111
657                             NaN       4839553
893                             NaN       4839669
989                             NaN       4840164
1235                            NaN       4840048
1303                            NaN       4840166

                    vehicle_type_code_1  \
2                                 Sedan
233                               Sedan
657                           Flat Rack
893                        Pick-up Truck
989                                Taxi
1235                 Tractor Truck Diesel
1303  Station Wagon/Sport Utility Vehicle

                    vehicle_type_code_2  vehicle_type_code_3  \
2                                 Sedan                  NaN
233                                 NaN                  NaN
657   Station Wagon/Sport Utility Vehicle                  NaN
893                 Tractor Truck Diesel                  NaN
989                                Bike                  NaN
1235                              Sedan                  NaN
1303                               Taxi                  NaN

      vehicle_type_code_4  vehicle_type_code_5 was_fillable zip_code_numeric
2                     NaN                  NaN         True          11040.0
233                   NaN                  NaN         True              NaN
657                   NaN                  NaN         True              NaN
893                   NaN                  NaN         True          11020.0
989                   NaN                  NaN         True              NaN
1235                  NaN                  NaN         True          11020.0
1303                  NaN                  NaN         True              NaN

[7 rows x 31 columns]
```

11040 and 11020 are Nassau county zip codes

13 manually fill the remaining missing zips

From Chatgpt, I learned that Nominatim sometimes struggle with Manhattan zip codes

▼ Code

```
manual_zip_fill = {
    233: "10065",  # 2nd Ave & E 61st
    657: "10065",  # 5th Ave & E 66th
    989: "10021",  # York Ave & E 80th
    1303: "10065"  # E 66th & Lexington
}

for idx, zipcode in manual_zip_fill.items():
    df.loc[idx, "zip_code"] = zipcode
    df.loc[idx, "zip_code_numeric"] = float(zipcode)

missing_rows2 = df[
    df["latitude"].notna() &
    df["longitude"].notna() &
    (df["zip_code"].isna() | df["borough"].isna())
]

print("Total rows:", df.shape[0])
print("Rows with lat/lon but missing zip or borough:", missing_rows.shape[0])
print("Sample of missing rows:")
print(missing_rows2.head(10))
```

```
Total rows: 1381
Rows with lat/lon but missing zip or borough: 7
Sample of missing rows:
      crash_date crash_time borough zip_code   latitude  longitude  \
2     08/31/2025      19:00    <NA>    11040  40.734234 -73.722748
893   09/04/2025       7:39    <NA>    11020  40.764267 -73.722946
1235  09/06/2025       9:20    <NA>    11020  40.764267 -73.722946

                     location          on_street_name cross_street_name  \
2      (40.734234, -73.72275)    CROSS ISLAND PARKWAY   HILLSIDE AVENUE
893   (40.764267, -73.722946)  LONG ISLAND EXPRESSWAY               NaN
1235  (40.764267, -73.722946)  LONG ISLAND EXPRESSWAY               NaN

     off_street_name  ...  contributing_factor_vehicle_4  \
2                NaN  ...                            NaN
893              NaN  ...                            NaN
1235             NaN  ...                            NaN

      contributing_factor_vehicle_5  collision_id   vehicle_type_code_1  \
2                               NaN       4838966                 Sedan
893                             NaN       4839669         Pick-up Truck
```

```
1235                          NaN       4840048  Tractor Truck Diesel

       vehicle_type_code_2  vehicle_type_code_3  vehicle_type_code_4  \
2                    Sedan                  NaN                  NaN
893   Tractor Truck Diesel                  NaN                  NaN
1235                 Sedan                  NaN                  NaN

      vehicle_type_code_5 was_fillable zip_code_numeric
2                     NaN         True          11040.0
893                   NaN         True          11020.0
1235                  NaN         True          11020.0

[3 rows x 31 columns]
```

14 store the variables that had zip_code, borough or both missing that has now

▼ Code

```
df["zip_filled"] = df["was_fillable"] & (
    df["zip_code"].notna() | df["borough"].notna()
)

print(df["was_fillable"].head(20))
```

```
0     False
1     False
2      True
3     False
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11     True
12    False
13     True
14    False
15     True
16    False
17    False
18    False
19    False
Name: was_fillable, dtype: bool
```

15 drop location only

▼ Code

```
df = df.drop(columns=['location'])

df.head()
```

	crash_date	crash_time	borough	zip_code	latitude	longitude	on_street_name	cross_street_name	off_str
0	08/31/2025	12:49	QUEENS	11101	40.753113	-73.933701	30 ST	39 AVE	NaN
1	08/31/2025	15:30	MANHATTAN	10022	40.760601	-73.964317	E 59 ST	2 AVE	NaN
2	08/31/2025	19:00	<NA>	11040	40.734234	-73.722748	CROSS ISLAND PARKWAY	HILLSIDE AVENUE	NaN
3	08/31/2025	1:19	BROOKLYN	11220	40.648075	-74.007034	NaN	NaN	4415 5
4	08/31/2025	2:41	MANHATTAN	10036	40.756561	-73.986107	W 43 ST	BROADWAY	NaN

5 rows × 31 columns

16 store crash date and crash time as a date time variable

▼ Code

```
df['crash_datetime'] = pd.to_datetime(
    df['crash_date'] + " " + df['crash_time'],
    format="%m/%d/%Y %H:%M",
    errors="coerce"
)

print("DataFrame with crash_datetime added:")
print(df[['crash_date', 'crash_time', 'crash_datetime']].head())
```

```
DataFrame with crash_datetime added:
   crash_date crash_time      crash_datetime
0  08/31/2025      12:49 2025-08-31 12:49:00
1  08/31/2025      15:30 2025-08-31 15:30:00
2  08/31/2025      19:00 2025-08-31 19:00:00
3  08/31/2025       1:19 2025-08-31 01:19:00
4  08/31/2025       2:41 2025-08-31 02:41:00
```

17 drop crash_time and crash_date

▼ Code

```
df = df.drop(columns=['crash_date', 'crash_time'])

df.head()
```

	borough	zip_code	latitude	longitude	on_street_name	cross_street_name	off_street_name	number_of_p
0	QUEENS	11101	40.753113	-73.933701	30 ST	39 AVE	NaN	0
1	MANHATTAN	10022	40.760601	-73.964317	E 59 ST	2 AVE	NaN	0
2	<NA>	11040	40.734234	-73.722748	CROSS ISLAND PARKWAY	HILLSIDE AVENUE	NaN	0
3	BROOKLYN	11220	40.648075	-74.007034	NaN	NaN	4415 5 AVE	2
4	MANHATTAN	10036	40.756561	-73.986107	W 43 ST	BROADWAY	NaN	1

5 rows × 30 columns

18 export the cleaned csv file as in feather format

▼ Code

```
df.to_feather("nyc_crashes_cleaned.feather")
```