Step 1:

| #Import necessary libraries  import pandas as pd  import matplotlib.pyplot as plt  from google.cloud import storage  #Authenticate User  from google.colab import auth  auth.authenticate\_user()  print('Authenticated') |
| --- |

Step 2:

| # Defined my Google Cloud Storage bucket, folder, and file name  bucket\_name = 'my-project-bucket-bc'  folder\_name = 'landing'  file\_name = 'nypd\_mv\_collisions'  # Initialize a client  storage\_client = storage.Client()  # Get the bucket and the blob  bucket = storage\_client.get\_bucket(bucket\_name)  blob = bucket.blob(folder\_name + '/' + file\_name) # Include folder path  # Download the file to a local directory  local\_file\_path = '/tmp/' + file\_name  #The file downloaded from Google Cloud Storage will be stored in the /tmp/ directory  blob.download\_to\_filename(local\_file\_path)  # Loaded the data into a DataFrame  df = pd.read\_csv(local\_file\_path)  # Print the descriptive statistics  print(df.describe()) |
| --- |

Step 3:

| # Display DF information/list of columns  df.info() |
| --- |

Step 4:

| # Number of NaN fields in the observations  missing\_counts = df.isna().sum()  missing\_counts |
| --- |

Step 5-18: (made minor changes in code for the column name in df[‘’]

| # Distribution of Data for Borough  plt.figure(figsize=(8, 6))  df['borough'].value\_counts().plot(kind='bar', color='skyblue')  plt.title('Distribution of Borough')  plt.xlabel('Borough')  plt.ylabel('Frequency')  plt.grid(True)  plt.show() |
| --- |

Step 19:

| # Display the Min/Max Date  df['timestamp'] = pd.to\_datetime(df['timestamp'])  min\_date = df['timestamp'].min().strftime('%Y-%m-%d')  max\_date = df['timestamp'].max().strftime('%Y-%m-%d')  print("The min date is", min\_date)  print("The max date is", max\_date) |
| --- |

From the Distribution of the Borough Diagram, it can be inferred that most of the incidents occur in Brooklyn while there is a drastically lower frequency of incidents in Staten Island. Based on the multiple Distribution of Contributing Factor Vehicle Diagrams, as the number increases from 1 to 5 there is a visible decrease in factors. By being able to see the oldest and newest dates, we can note the accuracy of the data as it includes data from 2012 to the current year 2024.

Step 20: Cleaning Begin

| #Import necessary libraries  import pandas as pd  import matplotlib.pyplot as plt  from google.cloud import storage  import pyarrow.parquet as pq  #Authenticate User  from google.colab import auth  auth.authenticate\_user()  print('Authenticated') |
| --- |

Step 21:

| # Initialize a GCS client  storage\_client = storage.Client()  # Define bucket and file paths  landing\_folder\_name = 'landing'  input\_file\_name = 'nypd\_mv\_collisions'  output\_folder\_name = 'cleaned'  output\_file\_name = 'parquet nypd\_mv\_collisions'  # Read data from the landing folder file  landing\_bucket = storage\_client.bucket('my-project-bucket-bc')  landing\_blob = landing\_bucket.blob(f'{landing\_folder\_name}/{input\_file\_name}')  local\_file\_path = '/tmp/input\_file.csv'  landing\_blob.download\_to\_filename(local\_file\_path) |
| --- |

Step 22:

| # Load data into a DataFrame  df = pd.read\_csv(local\_file\_path) |
| --- |

Step 23:

| # Removed Columns that isn't going to be used in the Project  columns\_to\_remove= ['latitude','longitude','location','off\_street\_name','on\_street\_name','zip\_code']  df = df.drop(labels=columns\_to\_remove, axis=1) |
| --- |

Step 24:

| #Created a filter to display rows with boro filled in  df\_boro\_filtered = df[df['borough'].notna()]  # Display the DataFrame with rows where 'borough' is not NaN  df\_boro\_filtered.head() |
| --- |

Step 25:

| # Further filter the DataFrame to remove rows where 'contributing\_factor\_vehicle\_1' is NaN  df\_CFV1\_filtered = df\_boro\_filtered[df\_boro\_filtered['contributing\_factor\_vehicle\_1'].notna()]  # Display the final filtered DataFrame  df\_CFV1\_filtered.head() |
| --- |

Step 26:

| #Display incidents where 5 vehicles were involved with no NaN data  # Create filter columns  columns\_to\_filter = df.columns  # Filtered DataFrame with the condition  df\_filtered = df[df[columns\_to\_filter[0]].notna()]  # Iterate over the remaining columns and apply the filtering condition  for column in columns\_to\_filter[1:]:  df\_filtered = df\_filtered[df\_filtered[column].notna()]  # Display final filtered DataFrame  df\_filtered |
| --- |

Step 27:

| missing\_counts = df\_filtered.isna().sum()  missing\_counts |
| --- |

Step 28:

| # Save the outputs to a new file in the output folder  output\_bucket = storage\_client.bucket('my-project-bucket-bc')  output\_blob = output\_bucket.blob(f'{output\_folder\_name}/{output\_file\_name}')  # Save description DataFrame to Parquet format  output\_file\_path = '/tmp/output\_file.parquet'  description.to\_parquet(output\_file\_path, engine='pyarrow')  # Upload Parquet file to GCS  output\_blob.upload\_from\_filename(output\_file\_path)  print("Outputs saved successfully as Parquet.") |
| --- |

The data started with multiple empty values in each column but after filtering through the dataset I was able to get an NaN count of 0. To do this I mainly utilized the .notna() function to edit the data frame. To note I left rows with the value of “Unspecified” as this doesn’t indicate an error in the data as an incident has still occurred. In most cases, the term “Unspecified” appears as a contributing factor of vehicles 2-5. This is mainly because, in a car accident, the 1st vehicle is more likely to be the cause while the following vehicles are just caught up in the incident.

One challenge I believe I will have in the following milestones will be working with my data in PySparks, as it is an API that I haven't touched on too often, even with multiple practices and videos to help. I think that concepts and knowledge of such topics will be hard to grasp when put into a practical situation.