### 

Bella Chen

bella.chen@baruchmail.cuny.edu

CIS 4130 CMWA [15843]

April 19. 2024

## Project Proposal

The top 4 causes of death are Heart disease, Cancer, COVID, and Accidents. According to the CDC, there was an estimated 710,000 increase in the population who died by Unintentional injuries from 2020 to 20211, it includes injuries such as, falls, motor vehicle crashes, poisoning, etc.

By using the [NYPD Motor Vehicle Collisions](https://console.cloud.google.com/marketplace/product/city-of-new-york/nypd-mv-collisions?authuser=1&project=cis-4130-semester-project) dataset from BigQuery has a section where I will be focusing which is, “new\_york\_mv\_collisions”. This dataset includes information such as Contributing factors of vehicle collisions, location of incidents, timestamps, unique keys, vehicle types, and most importantly the number of people injured or killed.

With the use of this dataset, I want to predict how many incidents may occur on a daily or monthly basis. To add on by displaying what type of factors or most common in the incidents I can predict future changes that need to be made to make driving safer and controlled, to keep incident numbers low.

The dataset has the following column name; borough, Contributing\_factor\_vehicle\_1 to Contributing\_factor\_vehicle\_5, cross\_street\_name, timestamps, number\_of \_cyclist\_injured, number\_of \_cyclist\_killed, number\_of \_motorist\_injured, number\_of \_motorist\_killed, number\_of \_pedestrians\_injured, number\_of \_pedestrians\_killed, number\_of \_persons\_injured, number\_of \_persons\_killed, unqiue\_keys, as well as the vehicle\_type\_code1 to vehicle\_type\_code5.

## Data Acquisition

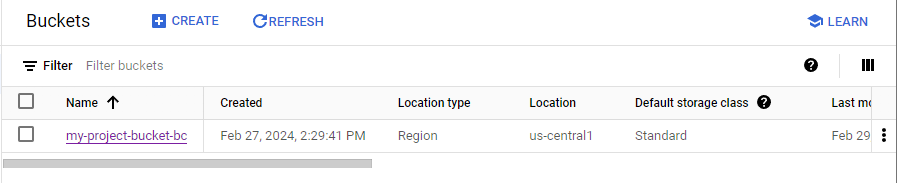
**Project ID:** cis-4130-semester-project

**Bucket Name:** my-project-bucket-bc

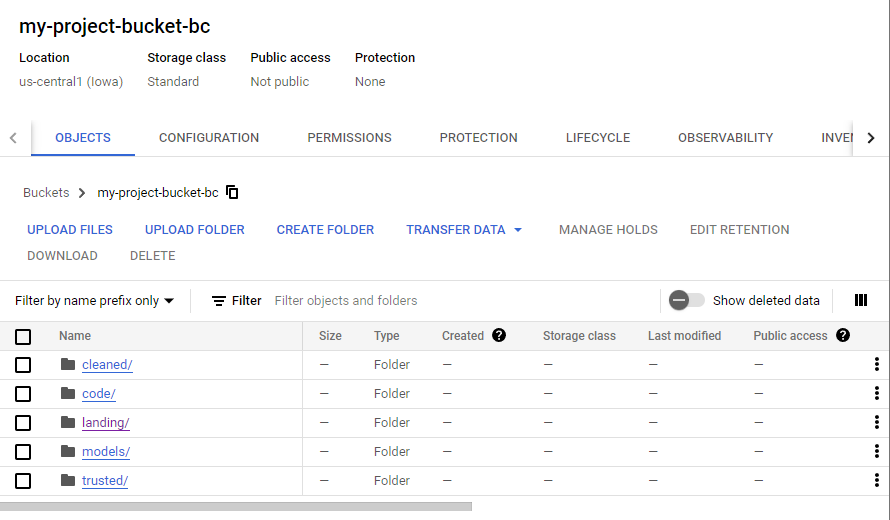
**File in Bucket:** my-project-bucket-bc/landing/nypd\_mv\_collisions

Step 1: Navigate to Cloud Storage >> Buckets

Step 2: Create a new bucket with the name: my-project-bucket-bc



Step 3: Create folders: cleaned, code, landing, models, trusted



Step 4: Go to the Dataset Link: [https://console.cloud.google.com/marketplace/product/city-of-new-york/nypd-mv-collisions](https://console.cloud.google.com/marketplace/product/city-of-new-york/nypd-mv-collisions?authuser=1&project=cis-4130-semester-project)

Step 5: Select View Dataset and It should open up BigQuery (after selected Project)

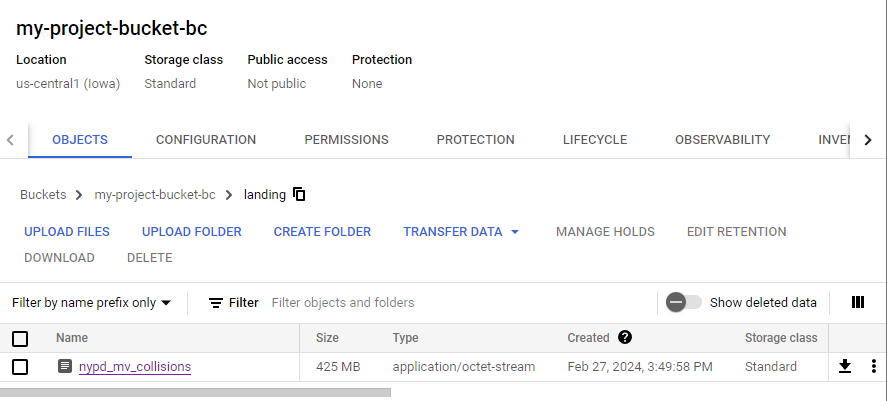
Step 6: Open the dataset with the dropdown error and select on the Table

Step 7: Select the Export icon in the top ribbon 

Step 8: Select Export to GCS

Step 9: Browse Buckets and select: my-project-bucket-bc

Step 10: Select landing/ folder save file name “nypd\_mv\_collisions”

Step 11: Make Sure the table is exported to Buckets; Navigate to Cloud Storage >> Buckets 

Step 12: Activate the Cloud Shell Editor on the top 

Step 13: Install Google Cloud Storage in Cloud Shell Editor

| bellachen1212@cloudshell:~ (cis-4130-semester-project)$ pip3 install google-cloud-storage |
| --- |

Step 14: Authenticate with Google Account

| bellachen1212@cloudshell:~ (cis-4130-semester-project)$ gcloud auth application-default login |
| --- |

| Ran into error:  ERROR: (gcloud.auth.application-default.login) PERMISSION\_DENIED: Service Usage API has not been used in project cis-4130-semester-project before or it is disabled. Enable it by visiting https://console.developers.google.com/apis/api/serviceusage.googleapis.com/overview?project=cis-4130-semester-project then retry. If you enabled this API recently, wait a few minutes for the action to propagate to our systems and retry.  - '@type': type.googleapis.com/google.rpc.Help  links:  - description: Google developers console API activation  url: https://console.developers.google.com/apis/api/serviceusage.googleapis.com/overview?project=cis-4130-semester-project  - '@type': type.googleapis.com/google.rpc.ErrorInfo  domain: googleapis.com  metadata:  consumer: projects/cis-4130-semester-project  service: serviceusage.googleapis.com  reason: SERVICE\_DISABLED   * Went to the website provided and enabled the Services with API * Re-run the code to authenticate |
| --- |

Step 15: Create and Avtivate Python Virtual Enviroment and Install Libraries

| **Create a Python Virtual Environment:** python3 -m venv pythondev  **Activate the Python VE:** cd pythondev; source bin/activate  **Install python libraries:** pip3 install jupyter pandas numpy |
| --- |

Step 16: Activate the Python Enviroment

| (pythondev) bellachen1212@cloudshell:~/pythondev (cis-4130-semester-project)$ python |
| --- |

Step 17: Import storage from google.cloud

| from google.cloud import storage |
| --- |

| Ran into error:  Traceback (most recent call last):  File "<stdin>", line 1, in <module>  ModuleNotFoundError: No module named 'google'   * Exited Python and Installed Google: * (pythondev) bellachen1212@cloudshell:~/pythondev (cis-4130-semester-project)$ pip install google |
| --- |

Step 18: List all Buckets

| # Create a client object that points to GCS  >>> storage\_client = storage.Client()  >>> buckets = storage\_client.list\_buckets()  >>> print(buckets)  <google.api\_core.page\_iterator.HTTPIterator object at 0x7b9114948bb0> |
| --- |

Step 19: Load file from GCS as DataFrame in Pandas (Import Pandas and io)

| >>> bucket = storage\_client.get\_bucket("my-project-bucket-bc")  >>> blob = bucket.blob("landing/"+"nypd\_mv\_collisions")  >>> import pandas as pd  >>> from io import StringIO  >>> df = pd.read\_csv(StringIO(blob.download\_as\_text())) |
| --- |

Step 20: Get info on DataFrame

| >>> df.info() |  |
| --- | --- |

## Exploratory Data Analysis and Data Cleaning

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.

## Feature Engineering and Modeling

| Column name | Column type | Variable Type | Indexer | Modeler | Scaler |
| --- | --- | --- | --- | --- | --- |
| borough | string | Categorical | Indexer |  |  |
| contributing\_factor\_vehicle\_1 | string | Categorical | Indexer |  |  |
| contributing\_factor\_vehicle\_2 | string | Categorical | Indexer |  |  |
| contributing\_factor\_vehicle\_3 | string | Categorical | Indexer |  |  |
| contributing\_factor\_vehicle\_4 | string | Categorical | Indexer |  |  |
| contributing\_factor\_vehicle\_5 | string | Categorical | Indexer |  |  |
| cross\_street\_name | string | Categorical | Indexer |  |  |
| timestamp | timestamp | Unknown |  |  | Scaler |
| number\_of\_cyclist\_injured | bigint | Unknown |  |  | Scaler |
| number\_of\_cyclist\_killed | bigint | Unknown |  |  | Scaler |
| number\_of\_motorist\_injured | bigint | Unknown |  |  | Scaler |
| number\_of\_motorist\_killed | bigint | Unknown |  |  | Scaler |
| number\_of\_pedestrians\_injured | bigint | Unknown |  |  | Scaler |
| number\_of\_pedestrians\_killed | bigint | Unknown |  |  | Scaler |
| number\_of\_persons\_injured | bigint | Unknown |  |  | Scaler |
| number\_of\_persons\_killed | bigint | Unknown |  |  | Scaler |
| unique\_key | bigint | Unknown |  |  | Scaler |
| vehicle\_type\_code1 | string | Categorical | Indexer |  |  |
| vehicle\_type\_code2 | string | Categorical | Indexer |  |  |
| vehicle\_type\_code\_3 | string | Categorical | Indexer |  |  |
| vehicle\_type\_code\_4 | string | Categorical | Indexer |  |  |
| vehicle\_type\_code\_5 | string | Categorical | Indexer |  |  |
| time\_year | int | Continuous |  | Modeler |  |
| time\_month | int | Continuous |  | Modeler |  |
| time\_yearmonth | string | Categorical | Indexer |  |  |
| time\_dayofweek | string | Categorical | Indexer |  |  |
| time\_weekend | double | Continuous |  |  | Scaler |

In my code, I used libraries such as Pandas, MatPlotLib, Google Cloud, PySpark, and to make some tables interactive I also used IPython. [Model Code](https://docs.google.com/document/d/1rdC2ukBvXXe6yT5FOtTZdJVj--TMmHwUHg3oSDd1lng/edit?usp=sharing)

One challenge I encountered was setting up a PySpark session in CoLab as I had to create it in Google Storage first. Then because most of my data are qualitative I had to consider how I wanted to graph by dataset to display relations.