ETL for CIS 9440 Data Warehousing and Analytics

- Project title: NYC Motor Vehicle Collision Transparency Data Warehouse Project
- Final Project Milestone 3
- Group Number: 5
- Version A: collision as a dimension
- Student(s): Gabriel Fernandez, Jason Jiang

ETL - Extract data

In [3]:
```
#Install the required python packages
#!pip install --upgrade sodapy
```

In [ ]:
```
# pip install --upgrade sodapy
# pip install --upgrade db-dtypes
# pip install --upgrade pyarrow
# pip install --upgrade google-cloud-bigquery
#for progress bar
# pip install tqdm
```

In [4]:
```
# import libraries
import pandas as pd
import numpy as np
from sodapy import Socrata
from google.cloud import bigquery
from google.oauth2 import service_account
from tqdm.notebook import tqdm_notebook # to show progress bar
from IPython.display import Image # to attach images
pd.options.mode.chained_assignment = None  # default='warn'
```

Data sets

Dataset 1: Motor Vehicle Collisions - Crashes

he Motor Vehicle Collisions crash table contains details on the crash event. Each row represents a crash event. The Motor Vehicle Collisions data tables contain information from all police reported motor vehicle collisions in NYC.

https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Crashes/h9gi-nx95

Dataset 2: Motor Vehicle Collisions - Person

The Motor Vehicle Collisions person table contains details for people involved in the crash. Each row represents a person (driver, occupant, pedestrian, bicyclist,..) involved in a crash. The data in this table goes back to April 2016 when crash reporting switched to an electronic system. https://data.cityofnewyork.us/Public-Safety/Motor-Vehicle-Collisions-Person/f55k-p6yu

---

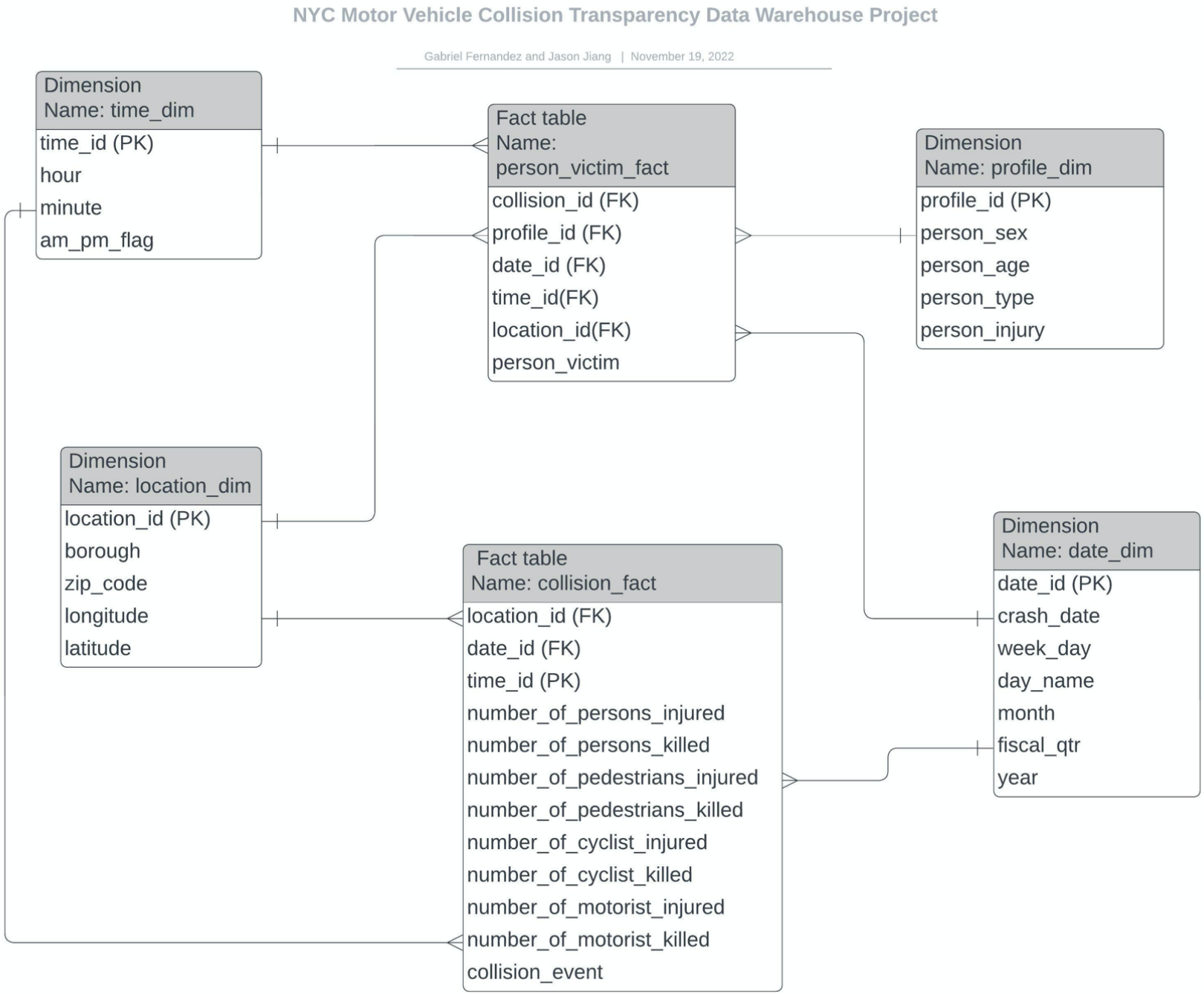
- Get your app-token from: https://data.cityofnewyork.us/profile/edit/developer_settings

Dimensional model (updated)

https://lucid.app/lucidchart/d42f0e3b-891b-49d3-9486-6ffabdc2f6d8/edit?page=0_0&invitationId=inv_b85589a9-5172-40f5-8ca8-450d9098461b#

In [7]:
```
Image("/content/drive/MyDrive/project_DW/version_2_collision_is_a_fact_table.jpeg")
```

Out[7]:


NYC Motor Vehicle Collision Transparency Data Warehouse Project
Gabriel Fernandez and Jason Jiang | November 19, 2022
Dimension
Name: time_dim
time_id (PK)
hour
minute
am_pm_flag
Fact table
Name:
person_victim_fact
collision_id (FK)
profile_id (FK)
date_id (FK)
time_id(FK)
location_id(FK)
person_victim
Dimension
Name: profile_dim
profile_id (PK)
person_sex
person_age
person_type
person_injury
Dimension
Name: location_dim
location_id (PK)
borough
zip_code
longitude
latitude
Fact table
Name: collision_fact
location_id (FK)
date_id (FK)
time_id (PK)
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
collision_event
Dimension
Name: date_dim
date_id (PK)
crash_date
week_day
day_name
month
fiscal_qtr
year

In [ ]:
```
# setup the host name for the API endpoint for NYC Open Data
data_url = 'data.cityofnewyork.us'
```

In [ ]:
```
# setup the data sets at the API endpoint

# end point https://data.cityofnewyork.us/resource/t5n6-gx8c.json
data_set1 = 'h9gi-nx95'
data_set2 = 'f55k-p6yu'
```

In [ ]:
```
# Setup your App Token
# You can find your app token by logging into: https://data.cityofnewyork.us/profile/edit/developer_settin
app_token = "your_token"
```

In [ ]:
```
# run this cell to setup your Socrata client that connects python to NYC Open Data

# create the client that points to the API endpoint
nyc_open_data_client = Socrata(data_url, app_token, timeout = 200)
print(f"nyc open data client name is: {nyc_open_data_client}")
print(f"nyc open data client data type is: {type(nyc_open_data_client)}")
```

```
nyc open data client name is: <sodapy.socrata.Socrata object at 0x7efdb4b62910>
nyc open data client data type is: <class 'sodapy.socrata.Socrata'>
```

In [ ]:
```
# Define a key_path variable to connect BigQuery
key_path = "your_key"
```

In [ ]:
```
# run this cell without changing anything to setup your credentials
credentials = service_account.Credentials.from_service_account_file(key_path,
                                                                    scopes=["https://www.googleapis.com/au
bigquery_client = bigquery.Client(credentials = credentials,
                                  project = credentials.project_id)

print(f"bigquery client name is: {bigquery_client}")
print(f"bigquery client data type is: {type(bigquery_client)}")
```

```
bigquery client name is: <google.cloud.bigquery.client.Client object at 0x7efdb4b6ea50>
bigquery client data type is: <class 'google.cloud.bigquery.client.Client'>
```

In [ ]:

```python
# Create a new dataset in BigQuery and copy its id

dataset_id = "your_database"

dataset_id = dataset_id.replace(':', '.')
print(f"your dataset_id is: {dataset_id}")
```

```
your dataset_id is: deft-stratum-361822.ETL_milestone3_v2
```

Extract data

1. connect to NYC Open Data with API Key
2. pull specific dataset as a pandas dataframe
3. Look at shape of extracted data

sodapy client.get parameters (https://dev.socrata.com/docs/:queries/)

1. select
2. where
3. order
4. limit
5. group '

In [ ]:

```python
# Get the total number of records in our the entire data sets

#data_set1
total_record_count = nyc_open_data_client.get(data_set1, select = "COUNT(*)")
print(f"total records in {data_set1}: {total_record_count[0]['COUNT']}")
```

```
total records in h9gi-nx95: 1945844
```

In [ ]:

```
#data_set2
total_record_count2 = nyc_open_data_client.get(data_set2, select = "COUNT(*)")
print(f"total records in {data_set2}: {total_record_count2[0]['COUNT']}")
```

```
total records in f55k-p6yu: 4865615
```

In [ ]:

```
# Now, loop through target data set to pull all rows in chunks (we cannot pull all rows at once)
# Maximum chunk size for limit parameter is 50,000 https://dev.socrata.com/docs/queries/limit.html

def extract_socrata_data(data_set,
                         chunk_size = 2500,
                         where = None):

    # measure time this function takes
    import time
    start_time1 = time.time()

    # get total number or records
    if where == None:
        total_records = int(nyc_open_data_client.get(data_set,
                                                     select= "COUNT(*)")[0]["COUNT"])
    else:
        total_records = int(nyc_open_data_client.get(data_set,
                                                     where = where,
                                                     select= "COUNT(*)")[0]["COUNT"])

    # start at 0, empty list for results
    start = 0
    results = []

    # for progress bar
    pbar = tqdm_notebook(desc = 'while loop', total = total_records)

    while True:

        if where == None:
            # fetch the set of records starting at 'start'
            results.extend(nyc_open_data_client.get(data_set,
                                                    offset = start,
                                                    limit = chunk_size))
```

```
        elif where != None:
            results.extend(nyc_open_data_client.get(data_set,
                                                    where = where,
                                                    offset = start,
                                                    limit = chunk_size))
        # update the starting record number
        start = start + chunk_size

        # Update progress bar
        print(start, end ='\r')
        pbar.update(chunk_size)

        # if we have fetched all of the records (we have reached total_records), exit loop
        if (start > total_records):
            print("Loop completed")
            #close progress bar
            pbar.close()
            break

    # convert the list into a pandas data frame
    end_time1 = time.time()
    print(f"Loop took {round(end_time1 - start_time1, 1)} seconds")

    start_time2 = time.time()

    data = pd.DataFrame.from_records(results)

    end_time2 = time.time()
    print(f"Transforming to pandas.DataFrame took {round(end_time2 - start_time2, 1)} seconds")

    print(f"The shape of your dataframe is: {data.shape}")
    return data



# progress bar: https://medium.com/@harshit4084/track-your-loop-using-tqdm-7-ways-progress-bars-in-python-
```

In [ ]:
```
# Fetch all rows for data set1

#data_set1 = 'h9gi-nx95'
data1 = extract_socrata_data(chunk_size = 10000,
                             data_set = data_set1)
```

```
Loop completed
Loop took 311.3 seconds
Transforming to pandas.DataFrame took 7.2 seconds
The shape of your dataframe is: (1945844, 29)
```

In [ ]:
```
data1.head()
```

Out[ ]:

	crash_date	crash_time	on_street_name	off_street_name	number_of_persons_injured	number_of_persons_killed	number_of
0	2021-09-11T00:00:00.000	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2	0	
1	2022-03-26T00:00:00.000	11:45	QUEENSBORO BRIDGE UPPER	NaN	1	0	
2	2022-06-29T00:00:00.000	6:55	THROGS NECK BRIDGE	NaN	0	0	
3	2021-09-11T00:00:00.000	9:35	NaN	NaN	0	0	
4	2021-12-14T00:00:00.000	8:13	SARATOGA AVENUE	DECATUR STREET	0	0	

5 rows × 29 columns

In [ ]:
```
# Fetch all rows for data set2
#data_set2 = 'f55k-p6yu'
data2 = extract_socrata_data(data_set = data_set2,
                             chunk_size = 50000)
```

```
Loop completed
Loop took 240.6 seconds
Transforming to pandas.DataFrame took 12.8 seconds
The shape of your dataframe is: (4865615, 21)
```

In [ ]:
```
data1.columns
```

Out[ ]:
```
Index(['crash_date', 'crash_time', 'on_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',
       'collision_id', 'vehicle_type_code1', 'vehicle_type_code2', 'borough',
       'zip_code', 'latitude', 'longitude', 'location', 'cross_street_name',
       'contributing_factor_vehicle_3', 'vehicle_type_code_3',
       'contributing_factor_vehicle_4', 'vehicle_type_code_4',
       'contributing_factor_vehicle_5', 'vehicle_type_code_5'],
      dtype='object')
```

In [ ]:
```
data2.columns
```

Out[ ]:
```
Index(['unique_id', 'collision_id', 'crash_date', 'crash_time', 'person_id',
       'person_type', 'person_injury', 'vehicle_id', 'ped_role', 'person_sex',
       'person_age', 'ejection', 'emotional_status', 'bodily_injury',
       'position_in_vehicle', 'safety_equipment', 'complaint', 'ped_location',
       'ped_action', 'contributing_factor_1', 'contributing_factor_2'],
      dtype='object')
```

Merge data

In [ ]:
```
data = data1.merge(data2,
                           how = 'inner',
                           left_on = "collision_id",
                           right_on = "collision_id")
```

In [ ]:
```
data.shape
```

Out[ ]:
```
(4865589, 49)
```

In [ ]:
```
data.head()
```

Out[ ]:

	crash_date_x	crash_time_x	on_street_name	off_street_name	number_of_persons_injured	number_of_persons_killed	number_
0	2021-09-11T00:00:00.000	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2	0	
1	2021-09-11T00:00:00.000	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2	0	
2	2021-09-11T00:00:00.000	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2	0	
3	2021-09-11T00:00:00.000	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2	0	
4	2022-03-26T00:00:00.000	11:45	QUEENSBORO BRIDGE UPPER	NaN	1	0	

5 rows × 49 columns

ETL - Transform data

Data profiling

1. Distinct values per column
2. Null values per column
3. Summary statistics per numeric column

In [ ]:
```
# what are the columns in our dataframe?
data.columns
```

Out[ ]:
```
Index(['crash_date_x', 'crash_time_x', 'on_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',
       'collision_id', 'vehicle_type_code1', 'vehicle_type_code2', 'borough',
       'zip_code', 'latitude', 'longitude', 'location', 'cross_street_name',
       'contributing_factor_vehicle_3', 'vehicle_type_code_3',
       'contributing_factor_vehicle_4', 'vehicle_type_code_4',
       'contributing_factor_vehicle_5', 'vehicle_type_code_5', 'unique_id',
       'crash_date_y', 'crash_time_y', 'person_id', 'person_type',
       'person_injury', 'vehicle_id', 'ped_role', 'person_sex', 'person_age',
       'ejection', 'emotional_status', 'bodily_injury', 'position_in_vehicle',
       'safety_equipment', 'complaint', 'ped_location', 'ped_action',
       'contributing_factor_1', 'contributing_factor_2'],
      dtype='object')
```

In [ ]:
```
data.rename(columns = {'person_id':'person_victim','crash_date_x':'crash_date','crash_time_x':'crash_time'
data.columns
```

Out[ ]:
```
Index(['crash_date', 'crash_time', 'on_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',
       'collision_id', 'vehicle_type_code1', 'vehicle_type_code2', 'borough',
       'zip_code', 'latitude', 'longitude', 'location', 'cross_street_name',
       'contributing_factor_vehicle_3', 'vehicle_type_code_3',
       'contributing_factor_vehicle_4', 'vehicle_type_code_4',
       'contributing_factor_vehicle_5', 'vehicle_type_code_5', 'unique_id',
       'crash_date_y', 'crash_time_y', 'person_victim', 'person_type',
       'person_injury', 'vehicle_id', 'ped_role', 'person_sex', 'person_age',
       'ejection', 'emotional_status', 'bodily_injury', 'position_in_vehicle',
       'safety_equipment', 'complaint', 'ped_location', 'ped_action',
       'contributing_factor_1', 'contributing_factor_2'],
      dtype='object')
```

In [ ]:

```
# create and run a function to ceate data profiling dataframe

def create_data_profiling_df(data):

    # create an empty dataframe to gather information about each column
    data_profiling_df = pd.DataFrame(columns = ["column_name",
                                                "column_type",
                                                "unique_values",
                                                "duplicate_values",
                                                "null_values",
                                                "non_null_values"])

    # loop through each column to add rows to the data_profiling_df dataframe
    for column in tqdm_notebook(data.columns):

        info_dict = {}

        try:
            info_dict["column_name"] = column
            info_dict["column_type"] = data[column].dtypes
            info_dict["unique_values"] = len(data[column].unique())
            info_dict["duplicate_values"] = data[column].count() - len(data[column].dropna().unique())
            info_dict["null_values"] = data[column].isna().sum()
            info_dict["non_null_values"] = data[column].count()

        except:
            print(f"unable to read column: {column}, you may want to drop this column")

        data_profiling_df = data_profiling_df.append(info_dict, ignore_index=True)

    data_profiling_df.sort_values(by = ['unique_values', "non_null_values"],
                                  ascending = [False, False],
                                  inplace=True)

    return data_profiling_df
```

In [ ]:

```
# view your data profiling dataframe
#RUN DATA PROFILING FUNCTION HERE
data_profiling_df = create_data_profiling_df(data = data)
data_profiling_df
```

```
unable to read column: location, you may want to drop this column
```

Out[ ]:

	column_name	column_type	unique_values	duplicate_values	null_values	non_null_values
29	unique_id	object	4861608	3981	0	4865589
32	person_victim	object	4666846	198725	19	4865570
35	vehicle_id	object	2241458	2430471	193661	4671928
14	collision_id	object	1316385	3549204	0	4865589
22	cross_street_name	object	184670	785020	3895900	969689
19	latitude	object	146020	4247622	471948	4393641
20	longitude	object	108562	4285080	471948	4393641
3	off_street_name	object	17477	2475153	2372960	2492629
2	on_street_name	object	16262	3841510	1007818	3857771
0	crash_date	object	3790	4861799	0	4865589
30	crash_date_y	object	3790	4861799	0	4865589
16	vehicle_type_code2	object	1619	4215792	648179	4217410
15	vehicle_type_code1	object	1449	4850115	14026	4851563
1	crash_time	object	1440	4864149	0	4865589
31	crash_time_y	object	1440	4864149	0	4865589
38	person_age	object	868	4379182	485540	4380049
18	zip_code	object	234	3012259	1853097	3012492
24	vehicle_type_code_3	object	232	552359	4312999	552590
26	vehicle_type_code_4	object	91	154610	4710889	154700
28	vehicle_type_code_5	object	63	49908	4815619	49970

12	contributing_factor_vehicle_1	object	62	4860201	5327	4860262
13	contributing_factor_vehicle_2	object	62	4356615	508913	4356676
47	contributing_factor_1	object	54	70286	4795250	70339
23	contributing_factor_vehicle_3	object	52	568742	4296796	568793
48	contributing_factor_2	object	51	70195	4795344	70245
25	contributing_factor_vehicle_4	object	39	158960	4706591	158998
4	number_of_persons_injured	object	29	4865519	42	4865547
10	number_of_motorist_injured	object	28	4865561	0	4865589
27	contributing_factor_vehicle_5	object	28	51167	4814395	51194
44	complaint	object	22	2558535	2307033	2558556
43	safety_equipment	object	19	2491423	2374148	2491441
46	ped_action	object	17	71498	4794075	71514
41	bodily_injury	object	15	2558535	2307040	2558549
6	number_of_pedestrians_injured	object	13	4865576	0	4865589
42	position_in_vehicle	object	12	2491502	2374076	2491513
36	ped_role	object	11	4670670	194909	4670680
40	emotional_status	object	9	2558498	2307083	2558506
5	number_of_persons_killed	object	8	4865490	92	4865497
39	ejection	object	7	2491158	2374425	2491164
11	number_of_motorist_killed	object	6	4865583	0	4865589
17	borough	object	6	3013306	1852278	3013311
8	number_of_cyclist_injured	object	5	4865584	0	4865589
45	ped_location	object	5	71611	4793974	71615
7	number_of_pedestrians_killed	object	4	4865585	0	4865589
33	person_type	object	4	4865585	0	4865589

37	person_sex	object	4	4310439	555147	4310442
9	number_of_cyclist_killed	object	3	4865586	0	4865589
34	person_injury	object	3	4865586	0	4865589
21	location	object	NaN	NaN	NaN	NaN

In [ ]:
```
#drop location because it is a dictionary and will raise an error in the next sections. Also, we already h
data.drop(['location'], axis=1, inplace=True)
```

Data cleaning

1. drop unneeded columns
2. drop duplicate rows
3. check for outliers

In [ ]:
```
# Run this to look at a list of your columns
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4865589 entries, 0 to 4865588
Data columns (total 48 columns):
 #   Column                          Dtype 
---  ------                          ----- 
 0   crash_date                      object
 1   crash_time                      object
 2   on_street_name                  object
 3   off_street_name                 object
 4   number_of_persons_injured       object
 5   number_of_persons_killed        object
 6   number_of_pedestrians_injured   object
 7   number_of_pedestrians_killed    object
 8   number_of_cyclist_injured       object
 9   number_of_cyclist_killed        object
 10  number_of_motorist_injured      object
 11  number_of_motorist_killed       object
 12  contributing_factor_vehicle_1   object
```

```
 13  contributing_factor_vehicle_2  object
 14  collision_id                   object
 15  vehicle_type_code1             object
 16  vehicle_type_code2             object
 17  borough                        object
 18  zip_code                       object
 19  latitude                       object
 20  longitude                      object
 21  cross_street_name              object
 22  contributing_factor_vehicle_3  object
 23  vehicle_type_code_3            object
 24  contributing_factor_vehicle_4  object
 25  vehicle_type_code_4            object
 26  contributing_factor_vehicle_5  object
 27  vehicle_type_code_5            object
 28  unique_id                      object
 29  crash_date_y                   object
 30  crash_time_y                   object
 31  person_victim                  object
 32  person_type                    object
 33  person_injury                  object
 34  vehicle_id                     object
 35  ped_role                       object
 36  person_sex                     object
 37  person_age                     object
 38  ejection                       object
 39  emotional_status               object
 40  bodily_injury                  object
 41  position_in_vehicle            object
 42  safety_equipment               object
 43  complaint                      object
 44  ped_location                   object
 45  ped_action                     object
 46  contributing_factor_1          object
 47  contributing_factor_2          object
dtypes: object(48)
memory usage: 1.8+ GB
```

Drop duplicates

In [ ]:

```
def drop_dupli(data):

  #check number of rows
  print(f"number of rows before dropping duplicates: {len(data)}")
  #check for duplciates
  print(f"number of duplicate rows: {len(data[data.duplicated()])}")
  #drop duplicate rows based on entire row
  data = data.drop_duplicates(keep = 'first')
  print(f"number of rows after duplicates dropped: {len(data)}")

  return data
```

In [ ]:

```
# drop duplicates
data_sin_du = drop_dupli(data)
```

```
number of rows before dropping duplicates: 4865589
number of duplicate rows: 3981
number of rows after duplicates dropped: 4861608
```

In [ ]:

```
data = data_sin_du.copy()
data.shape
```

Out[ ]:

```
(4861608, 48)
```

In [ ]:

```
# update numeric columns types for data

select_cols = ['latitude',
               'longitude',
               'person_age',
               'collision_id',
               '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']

for column in tqdm_notebook(select_cols):
    try:
        data[column] = data[column].astype(int)

    except:
        data[column] = data[column].astype(float)

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4861608 entries, 0 to 4865588
Data columns (total 48 columns):
 #   Column                          Dtype
---  ------                          -----
 0   crash_date                      object
 1   crash_time                      object
 2   on_street_name                  object
 3   off_street_name                 object
 4   number_of_persons_injured       float64
 5   number_of_persons_killed        float64
 6   number_of_pedestrians_injured   int64
 7   number_of_pedestrians_killed    int64
 8   number_of_cyclist_injured       int64
 9   number_of_cyclist_killed        int64
 10  number_of_motorist_injured      int64
 11  number_of_motorist_killed       int64
```

```
 12  contributing_factor_vehicle_1  object
 13  contributing_factor_vehicle_2  object
 14  collision_id                   int64
 15  vehicle_type_code1             object
 16  vehicle_type_code2             object
 17  borough                        object
 18  zip_code                       object
 19  latitude                       float64
 20  longitude                      float64
 21  cross_street_name              object
 22  contributing_factor_vehicle_3  object
 23  vehicle_type_code_3            object
 24  contributing_factor_vehicle_4  object
 25  vehicle_type_code_4            object
 26  contributing_factor_vehicle_5  object
 27  vehicle_type_code_5            object
 28  unique_id                      object
 29  crash_date_y                   object
 30  crash_time_y                   object
 31  person_victim                  object
 32  person_type                    object
 33  person_injury                  object
 34  vehicle_id                     object
 35  ped_role                       object
 36  person_sex                     object
 37  person_age                     float64
 38  ejection                       object
 39  emotional_status               object
 40  bodily_injury                  object
 41  position_in_vehicle            object
 42  safety_equipment               object
 43  complaint                      object
 44  ped_location                   object
 45  ped_action                     object
 46  contributing_factor_1          object
 47  contributing_factor_2          object
dtypes: float64(5), int64(7), object(36)
memory usage: 1.8+ GB
```

Check for outliers

In [ ]:
```
# select numerics col for data
numerics = ['int16', 'int32', 'int64', 'float16', 'float32', 'float64']

data_numerics= data.select_dtypes(include=numerics)
```

In [ ]:
```
# Descriptive statistics for data
data_numerics.drop(columns=['latitude','longitude', 'collision_id']).describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
number_of_persons_injured	4861566.0	0.475207	1.017065	0.0	0.0	0.0	1.0	43.0
number_of_persons_killed	4861516.0	0.002256	0.054918	0.0	0.0	0.0	0.0	8.0
number_of_pedestrians_injured	4861608.0	0.051150	0.253469	0.0	0.0	0.0	0.0	27.0
number_of_pedestrians_killed	4861608.0	0.000974	0.034329	0.0	0.0	0.0	0.0	6.0
number_of_cyclist_injured	4861608.0	0.024445	0.157043	0.0	0.0	0.0	0.0	4.0
number_of_cyclist_killed	4861608.0	0.000161	0.013120	0.0	0.0	0.0	0.0	2.0
number_of_motorist_injured	4861608.0	0.397206	0.998299	0.0	0.0	0.0	0.0	43.0
number_of_motorist_killed	4861608.0	0.001103	0.038202	0.0	0.0	0.0	0.0	5.0
person_age	4376707.0	36.979794	118.344302	-999.0	23.0	35.0	50.0	9999.0

In [ ]:
```
data.shape
```

Out[ ]:
```
(4861608, 48)
```

In [ ]:
```
 # Filter out person_age < 0 and > 120
 data= data[(0 < data["person_age"]) & (data["person_age"] < 120)].copy()
```

In [ ]:
```
data["person_age"].describe().T
```

Out[ ]:
```
count    3.826809e+06
mean     4.018758e+01
std      1.668685e+01
min      1.000000e+00
25%      2.800000e+01
50%      3.800000e+01
75%      5.200000e+01
max      1.190000e+02
Name: person_age, dtype: float64
```

In [ ]:
```
data.shape
```

Out[ ]:
```
(3826809, 48)
```

Create location dimension

In [ ]:
```
# first, copy the entire table
location_dim = data.copy()
```

In [ ]:
```
location_dim.columns
```

Out[ ]:
```
Index(['crash_date', 'crash_time', 'on_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',
       'collision_id', 'vehicle_type_code1', 'vehicle_type_code2', 'borough',
       'zip_code', 'latitude', 'longitude', 'cross_street_name',
       'contributing_factor_vehicle_3', 'vehicle_type_code_3',
       'contributing_factor_vehicle_4', 'vehicle_type_code_4',
       'contributing_factor_vehicle_5', 'vehicle_type_code_5', 'unique_id',
       'crash_date_y', 'crash_time_y', 'person_victim', 'person_type',
       'person_injury', 'vehicle_id', 'ped_role', 'person_sex', 'person_age',
       'ejection', 'emotional_status', 'bodily_injury', 'position_in_vehicle',
       'safety_equipment', 'complaint', 'ped_location', 'ped_action',
       'contributing_factor_1', 'contributing_factor_2'],
      dtype='object')
```

In [ ]:

```
# second, subset for only the wanted columns in the dimension
location_dim = location_dim[['borough', 'zip_code','latitude', 'longitude']]
```

In [ ]:

```
# third, drop duplicate rows in dimension
# create unique row for dimension
unique_row = ['borough', 'zip_code','latitude', 'longitude']
#drop duplicates
location_dim = location_dim.drop_duplicates(subset = unique_row, keep = 'first')
#drop nulls
location_dim.dropna(inplace = True)
#reset index
location_dim = location_dim.reset_index(drop = True)
location_dim
```

Out[ ]:

	borough	zip_code	latitude	longitude
0	BROOKLYN	11208	40.667202	-73.866500
1	BROOKLYN	11233	40.683304	-73.917274
2	BRONX	10475	40.868160	-73.831480
3	MANHATTAN	10017	40.751440	-73.973970
4	QUEENS	11413	40.675884	-73.755770
...	...	...	...	...
205613	QUEENS	11103	40.760605	-73.908010
205614	BROOKLYN	11226	40.646510	-73.948150
205615	QUEENS	11432	40.707283	-73.794655
205616	BRONX	10463	40.885624	-73.907200
205617	BRONX	10475	40.869587	-73.827090

205618 rows × 4 columns

In [ ]:

```
# fourth, add location_id as a surrogate key
location_dim.insert(0, 'location_id', range(1, 1 + len(location_dim)))
location_dim
```

Out[ ]:

	location_id	borough	zip_code	latitude	longitude
0	1	BROOKLYN	11208	40.667202	-73.866500
1	2	BROOKLYN	11233	40.683304	-73.917274
2	3	BRONX	10475	40.868160	-73.831480
3	4	MANHATTAN	10017	40.751440	-73.973970
4	5	QUEENS	11413	40.675884	-73.755770
...	...	...	...	...	...
205613	205614	QUEENS	11103	40.760605	-73.908010
205614	205615	BROOKLYN	11226	40.646510	-73.948150
205615	205616	QUEENS	11432	40.707283	-73.794655
205616	205617	BRONX	10463	40.885624	-73.907200
205617	205618	BRONX	10475	40.869587	-73.827090

205618 rows × 5 columns

In [ ]:

```
# fifth, add the location_id to the data table
data = data.merge(location_dim,
                  left_on = unique_row,
                  right_on = unique_row,
                  how = 'left')

data.head(100)
```

Out[ ]:

	crash_date	crash_time	on_street_name	off_street_name	number_of_persons_injured	number_of_persons_killed	number_c
0	2021-09-11T00:00:00.000	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	
1	2021-09-11T00:00:00.000	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	
2	2021-09-11T00:00:00.000	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	
3	2022-03-26T00:00:00.000	11:45	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	
4	2022-03-26T00:00:00.000	11:45	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	
...	...	...	...	...	...	...	
95	2021-07-09T00:00:00.000	0:43	ELIOT AVENUE	NaN	0.0	1.0	
96	2021-07-09T00:00:00.000	0:43	ELIOT AVENUE	NaN	0.0	1.0	
97	2022-04-24T00:00:00.000	16:45	STATEN ISLAND EXPRESSWAY	NaN	1.0	0.0	
98	2022-04-24T00:00:00.000	16:45	STATEN ISLAND EXPRESSWAY	NaN	1.0	0.0	
99	2022-04-24T00:00:00.000	16:45	STATEN ISLAND EXPRESSWAY	NaN	1.0	0.0	

100 rows × 49 columns

In [ ]:

```
data.shape
```

Out[ ]:
```
(3826809, 49)
```

Create date dimension

In [ ]:
```
# change to data data type
data['crash_date'] = pd.to_datetime(data['crash_date'])
#get the date portion
data['crash_date']= data['crash_date'].dt.floor('D')
data['crash_date']
```

Out[ ]:
```
0          2021-09-11
1          2021-09-11
2          2021-09-11
3          2022-03-26
4          2022-03-26
              ...
3826804    2022-11-11
3826805    2022-11-14
3826806    2022-11-14
3826807    2022-11-14
3826808    2022-11-14
Name: crash_date, Length: 3826809, dtype: datetime64[ns]
```

In [ ]:

```
## ACTION REQUIRED: update the start and end date at the bottom of the sql_query variable to fit your need

sql_query = """
            SELECT
              CONCAT (FORMAT_DATE("%Y",d),FORMAT_DATE("%m",d),FORMAT_DATE("%d",d)) as date_id,
              d AS crash_date,
              FORMAT_DATE('%w', d) AS week_day,
              FORMAT_DATE('%A', d) AS day_name,
              FORMAT_DATE('%B', d) as month,
              FORMAT_DATE('%Q', d) as fiscal_qtr,
              FORMAT_DATE('%Y', d) AS year,
            FROM (
              SELECT
                *
              FROM
                UNNEST(GENERATE_DATE_ARRAY('2012-07-01', '2024-01-01', INTERVAL 1 DAY)) AS d )
            """

# store extracted data in new dataframe
date_dim = bigquery_client.query(sql_query).to_dataframe()

# validate that > 0 rows have been extracted and return dataframe
if len(date_dim) > 0:
    print(f"date dimension created successfully, shape of dimension: {date_dim.shape}")
else:
    print("date dimension FAILED")
```

```
date dimension created successfully, shape of dimension: (4202, 7)
```

In [ ]:

```
#check date_dim
date_dim.head()
```

Out[ ]:

	date_id	crash_date	week_day	day_name	month	fiscal_qtr	year
0	20120701	2012-07-01	0	Sunday	July	3	2012
1	20120702	2012-07-02	1	Monday	July	3	2012
2	20120703	2012-07-03	2	Tuesday	July	3	2012
3	20120704	2012-07-04	3	Wednesday	July	3	2012
4	20120705	2012-07-05	4	Thursday	July	3	2012

In [ ]:

```
date_dim.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4202 entries, 0 to 4201
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   date_id     4202 non-null   object
 1   crash_date  4202 non-null   dbdate
 2   week_day    4202 non-null   object
 3   day_name    4202 non-null   object
 4   month       4202 non-null   object
 5   fiscal_qtr  4202 non-null   object
 6   year        4202 non-null   object
dtypes: dbdate(1), object(6)
memory usage: 229.9+ KB
```

In [ ]:

```
# create date_id column in the Fact Table
data['date_id'] = data['crash_date'].apply(lambda x: pd.to_datetime(x).strftime("%Y%m%d"))
```

In [ ]:

```
#check date_dim
data.head()
```

Out[ ]:

	crash_date	crash_time	on_street_name	off_street_name	number_of_persons_injured	number_of_persons_killed	number_of_pede
0	2021-09-11	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	
1	2021-09-11	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	
2	2021-09-11	2:39	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	
3	2022-03-26	11:45	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	
4	2022-03-26	11:45	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	

5 rows × 50 columns

In [ ]:
```
data.shape
```

Out[ ]:
```
(3826809, 50)
```

In [ ]:
```
# Drop date from complete dataset.
data.drop("crash_date", axis = 1, inplace = True)
```

In [ ]:
```
data.shape
```

Out[ ]:
```
(3826809, 49)
```

Create time dimension

In [ ]:

```
time_ids = []
hours = []
minutes = []

for hour in range(0,24):
    for minute in range(0,60):
        time_ids.append((str(hour).zfill(2) + str(minute).zfill(2)))
        hours.append(hour)
        minutes.append(minute)

time_dim_dict = {"time_id" : time_ids,
                 "hour" : hours,
                 "minute" : minutes}

time_dim = pd.DataFrame(data = time_dim_dict)
```

In [ ]:

```
# add a column for AM and PM
time_dim['am_pm_flag'] = np.where(time_dim['hour'] >= 12,'PM','AM')
```

In [ ]:

```
time_dim
```

Out[ ]:

	time_id	hour	minute	am_pm_flag
0	0000	0	0	AM
1	0001	0	1	AM
2	0002	0	2	AM
3	0003	0	3	AM
4	0004	0	4	AM
...	...	...	...	...
1435	2355	23	55	PM
1436	2356	23	56	PM
1437	2357	23	57	PM
1438	2358	23	58	PM
1439	2359	23	59	PM

1440 rows × 4 columns

In [ ]:

```
time_dim.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1440 entries, 0 to 1439
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   time_id     1440 non-null   object
 1   hour        1440 non-null   int64 
 2   minute      1440 non-null   int64 
 3   am_pm_flag  1440 non-null   object
dtypes: int64(2), object(2)
memory usage: 45.1+ KB
```

In [ ]:

```
# Remove ":"" from crash time
data["crash_time"]
```

Out[ ]:
```
0           2:39
1           2:39
2           2:39
3          11:45
4          11:45
           ...
3826804    20:21
3826805    17:10
3826806    17:10
3826807    17:10
3826808    17:10
Name: crash_time, Length: 3826809, dtype: object
```

In [ ]:
```
data["crash_time"] = data["crash_time"].replace({':': ''}, regex=True).str.strip()
```

In [ ]:
```
data["crash_time"]
```

Out[ ]:
```
0           239
1           239
2           239
3          1145
4          1145
           ...
3826804    2021
3826805    1710
3826806    1710
3826807    1710
3826808    1710
Name: crash_time, Length: 3826809, dtype: object
```

In [ ]:
```
# USE PAD here to add a zero to time_id on the left
data["crash_time"] = data["crash_time"].str.pad(4, side = "left", fillchar = "0")
```

In [ ]:
```
# check fact table progress
data.head()
```

Out[ ]:

	crash_time	on_street_name	off_street_name	number_of_persons_injured	number_of_persons_killed	number_of_pedestrians_injur
0	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	
1	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	
2	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	
3	1145	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	
4	1145	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	

5 rows × 49 columns

In [ ]:

```
data.rename(columns = {'crash_time':'time_id'}, inplace = True)
data["time_id"]
```

Out[ ]:

```
0          0239
1          0239
2          0239
3          1145
4          1145
           ...
3826804    2021
3826805    1710
3826806    1710
3826807    1710
3826808    1710
Name: time_id, Length: 3826809, dtype: object
```

Create profile dimension

In [ ]:
```
# first, copy the entire table
profile_dim = data.copy()
```

In [ ]:
```
profile_dim.columns
```

Out[ ]:
```
Index(['time_id', 'on_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',
       'collision_id', 'vehicle_type_code1', 'vehicle_type_code2', 'borough',
       'zip_code', 'latitude', 'longitude', 'cross_street_name',
       'contributing_factor_vehicle_3', 'vehicle_type_code_3',
       'contributing_factor_vehicle_4', 'vehicle_type_code_4',
       'contributing_factor_vehicle_5', 'vehicle_type_code_5', 'unique_id',
       'crash_date_y', 'crash_time_y', 'person_victim', 'person_type',
       'person_injury', 'vehicle_id', 'ped_role', 'person_sex', 'person_age',
       'ejection', 'emotional_status', 'bodily_injury', 'position_in_vehicle',
       'safety_equipment', 'complaint', 'ped_location', 'ped_action',
       'contributing_factor_1', 'contributing_factor_2', 'location_id',
       'date_id'],
      dtype='object')
```

In [ ]:
```
# second, subset for only the wanted columns in the dimension
profile_dim = profile_dim[['person_sex', 'person_age', 'person_type','person_injury']].copy()
```

In [ ]:

```
# third, drop duplicate rows in dimension
# create unique row for dimension
unique_row = ['person_sex', 'person_age', 'person_type','person_injury']
#drop duplicates
profile_dim = profile_dim.drop_duplicates(subset = unique_row, keep = 'first')
#drop nulls
profile_dim.dropna(inplace = True)
#reset index
profile_dim = profile_dim.reset_index(drop = True)
profile_dim
```

Out[ ]:

	person_sex	person_age	person_type	person_injury
0	M	29.0	Occupant	Injured
1	M	25.0	Occupant	Unspecified
2	M	33.0	Occupant	Injured
3	F	28.0	Occupant	Injured
4	M	29.0	Occupant	Unspecified
...	...	...	...	...
2126	F	59.0	Other Motorized	Unspecified
2127	M	87.0	Other Motorized	Unspecified
2128	M	23.0	Other Motorized	Killed
2129	F	62.0	Occupant	Killed
2130	M	44.0	Other Motorized	Killed

2131 rows × 4 columns

In [ ]:

```
# fourth, add profile_id as a surrogate key
profile_dim.insert(0, 'profile_id', range(1, 1 + len(profile_dim)))
profile_dim
```

Out[ ]:

	profile_id	person_sex	person_age	person_type	person_injury
0	1	M	29.0	Occupant	Injured
1	2	M	25.0	Occupant	Unspecified
2	3	M	33.0	Occupant	Injured
3	4	F	28.0	Occupant	Injured
4	5	M	29.0	Occupant	Unspecified
...	...	...	...	...	...
2126	2127	F	59.0	Other Motorized	Unspecified
2127	2128	M	87.0	Other Motorized	Unspecified
2128	2129	M	23.0	Other Motorized	Killed
2129	2130	F	62.0	Occupant	Killed
2130	2131	M	44.0	Other Motorized	Killed

2131 rows × 5 columns

In [ ]:
```
#convert age to int
profile_dim['person_age']= profile_dim['person_age'].astype(int)
```

In [ ]:
```
profile_dim['person_age']
```

Out[ ]:
```
0       29
1       25
2       33
3       28
4       29
        ..
2126    59
2127    87
2128    23
2129    62
2130    44
Name: person_age, Length: 2131, dtype: int64
```

In [ ]:
```
# fifth, add the profile_id to the data table
data = data.merge(profile_dim,
                  left_on = unique_row,
                  right_on = unique_row,
                  how = 'left')

data.head(100)
```

Out[ ]:

	time_id	on_street_name	off_street_name	number_of_persons_injured	number_of_persons_killed	number_of_pedestrians_injured
0	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	0
1	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	0
2	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	0
3	1145	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	0
4	1145	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	0
...	...	...	...	...	...	...
95	0043	ELIOT AVENUE	NaN	0.0	1.0	0
96	0043	ELIOT AVENUE	NaN	0.0	1.0	0
97	1645	STATEN ISLAND EXPRESSWAY	NaN	1.0	0.0	0
98	1645	STATEN ISLAND EXPRESSWAY	NaN	1.0	0.0	0
99	1645	STATEN ISLAND EXPRESSWAY	NaN	1.0	0.0	0

100 rows × 50 columns

Create collision fact table

In [ ]:
```
# Create person dimension from data2
data.head()
```

Out[ ]:

	time_id	on_street_name	off_street_name	number_of_persons_injured	number_of_persons_killed	number_of_pedestrians_injured
0	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	0
1	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	0
2	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	0
3	1145	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	0
4	1145	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	0

5 rows × 50 columns

In [ ]:
```
data.shape
```

Out[ ]:
```
(3826809, 50)
```

In [ ]:
```
# first, copy the entire table
collision_fact= data.copy()
```

In [ ]:
```
collision_fact.columns
```

Out[ ]:
```
Index(['time_id', 'on_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',
       'collision_id', 'vehicle_type_code1', 'vehicle_type_code2', 'borough',
       'zip_code', 'latitude', 'longitude', 'cross_street_name',
       'contributing_factor_vehicle_3', 'vehicle_type_code_3',
       'contributing_factor_vehicle_4', 'vehicle_type_code_4',
       'contributing_factor_vehicle_5', 'vehicle_type_code_5', 'unique_id',
       'crash_date_y', 'crash_time_y', 'person_victim', 'person_type',
       'person_injury', 'vehicle_id', 'ped_role', 'person_sex', 'person_age',
       'ejection', 'emotional_status', 'bodily_injury', 'position_in_vehicle',
       'safety_equipment', 'complaint', 'ped_location', 'ped_action',
       'contributing_factor_1', 'contributing_factor_2', 'location_id',
       'date_id', 'profile_id'],
      dtype='object')
```

In [ ]:
```
#second, subset for only the wanted columns
collision_fact= collision_fact[[
                               'location_id',
                               'date_id',
                               'time_id',
                               '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',
                               'collision_id']].copy()
```

In [ ]:
```
collision_fact.shape
```

Out[ ]:
```
(3826809, 12)
```

In [ ]:
```
# drop duplicates
collision_fact = drop_dupli(collision_fact)
```

```
number of rows before dropping duplicates: 3826809
number of duplicate rows: 2560473
number of rows after duplicates dropped: 1266336
```

In [ ]:
```
# remove nulls
collision_fact.dropna(inplace = True)
collision_fact = collision_fact.reset_index(drop = True)
collision_fact
```

Out[ ]:

	location_id	date_id	time_id	number_of_persons_injured	number_of_persons_killed	number_of_pedestrians_injured	nu
0	1.0	20210911	0935	0.0	0.0	0	
1	2.0	20211214	0813	0.0	0.0	0	
2	3.0	20211214	0817	2.0	0.0	0	
3	4.0	20211214	1458	0.0	0.0	0	
4	5.0	20211214	1650	0.0	0.0	0	
...	...	...	...	...	...	...	
790800	71741.0	20221102	2109	1.0	0.0	0	
790801	14783.0	20221115	1520	0.0	0.0	0	
790802	7562.0	20221115	2047	1.0	0.0	1	
790803	205618.0	20221114	1945	0.0	0.0	0	
790804	4919.0	20221115	1540	0.0	0.0	0	

790805 rows × 12 columns

In [ ]:
```
collision_fact.rename(columns = {'collision_id':'collision_event'}, inplace = True)
```

In [ ]:
```
collision_fact.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 790805 entries, 0 to 790804
Data columns (total 12 columns):
 #   Column                         Non-Null Count   Dtype  
---  ------                         --------------   -----  
 0   location_id                    790805 non-null  float64
 1   date_id                        790805 non-null  object 
 2   time_id                        790805 non-null  object 
 3   number_of_persons_injured      790805 non-null  float64
 4   number_of_persons_killed       790805 non-null  float64
 5   number_of_pedestrians_injured  790805 non-null  int64  
 6   number_of_pedestrians_killed   790805 non-null  int64  
 7   number_of_cyclist_injured      790805 non-null  int64  
 8   number_of_cyclist_killed       790805 non-null  int64  
 9   number_of_motorist_injured     790805 non-null  int64  
 10  number_of_motorist_killed      790805 non-null  int64  
 11  collision_event                790805 non-null  int64  
dtypes: float64(3), int64(7), object(2)
memory usage: 72.4+ MB
```

In [ ]:
```
#convert to correct type
collision_fact['number_of_persons_injured'] = collision_fact['number_of_persons_injured'].astype(int)
collision_fact['number_of_persons_killed']= collision_fact['number_of_persons_killed'].astype(int)
collision_fact['location_id']= collision_fact['location_id'].astype(int)
```

In [ ]:
```
collision_fact.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 790805 entries, 0 to 790804
Data columns (total 12 columns):
 #   Column                          Non-Null Count   Dtype 
---  ------                          --------------   ----- 
 0   location_id                     790805 non-null  int64 
 1   date_id                         790805 non-null  object
 2   time_id                         790805 non-null  object
 3   number_of_persons_injured       790805 non-null  int64 
 4   number_of_persons_killed        790805 non-null  int64 
 5   number_of_pedestrians_injured   790805 non-null  int64 
 6   number_of_pedestrians_killed    790805 non-null  int64 
 7   number_of_cyclist_injured       790805 non-null  int64 
 8   number_of_cyclist_killed        790805 non-null  int64 
 9   number_of_motorist_injured      790805 non-null  int64 
 10  number_of_motorist_killed       790805 non-null  int64 
 11  collision_event                 790805 non-null  int64 
dtypes: int64(10), object(2)
memory usage: 72.4+ MB
```

Create person_victim fact table

In [ ]:

```
data.head()
```

Out[ ]:

	time_id	on_street_name	off_street_name	number_of_persons_injured	number_of_persons_killed	number_of_pedestrians_injured
0	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	0
1	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	0
2	0239	WHITESTONE EXPRESSWAY	20 AVENUE	2.0	0.0	0
3	1145	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	0
4	1145	QUEENSBORO BRIDGE UPPER	NaN	1.0	0.0	0

5 rows × 50 columns

In [ ]:
```
data.shape
```

Out[ ]:
```
(3826809, 50)
```

In [ ]:
```
data.columns
```

Out[ ]:
```
Index(['time_id', 'on_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',
       'collision_id', 'vehicle_type_code1', 'vehicle_type_code2', 'borough',
       'zip_code', 'latitude', 'longitude', 'cross_street_name',
       'contributing_factor_vehicle_3', 'vehicle_type_code_3',
       'contributing_factor_vehicle_4', 'vehicle_type_code_4',
       'contributing_factor_vehicle_5', 'vehicle_type_code_5', 'unique_id',
       'crash_date_y', 'crash_time_y', 'person_victim', 'person_type',
       'person_injury', 'vehicle_id', 'ped_role', 'person_sex', 'person_age',
       'ejection', 'emotional_status', 'bodily_injury', 'position_in_vehicle',
       'safety_equipment', 'complaint', 'ped_location', 'ped_action',
       'contributing_factor_1', 'contributing_factor_2', 'location_id',
       'date_id', 'profile_id'],
      dtype='object')
```

In [ ]:
```
# take a subset of fact_table for only the needed columns: which are keys and measures
person_victim_fact = data[['profile_id', 'date_id','time_id','location_id','person_victim']].copy()
```

In [ ]:
```
person_victim_fact.shape
```

Out[ ]:
```
(3826809, 5)
```

In [ ]:
```
person_victim_fact.dropna(inplace = True)
```

In [ ]:
```
person_victim_fact.shape
```

Out[ ]:
```
(2103793, 5)
```

In [ ]:
```
person_victim_fact = drop_dupli(person_victim_fact)
```

```
number of rows before dropping duplicates: 2103793
number of duplicate rows: 0
number of rows after duplicates dropped: 2103793
```

In [ ]:

```
person_victim_fact.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2103793 entries, 7 to 3826802
Data columns (total 5 columns):
 #   Column         Dtype  
---  ------         -----  
 0   profile_id     float64
 1   date_id        object 
 2   time_id        object 
 3   location_id    float64
 4   person_victim  object 
dtypes: float64(2), object(3)
memory usage: 96.3+ MB
```

In [ ]:

```
# change to proper type


for column in ['profile_id',
               'location_id',
               'person_victim']:
    try:
        person_victim_fact[column] = person_victim_fact[column].astype(int)

    except:
        person_victim_fact[column] = person_victim_fact[column].astype(object)
```

In [ ]:

```
person_victim_fact.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2103793 entries, 7 to 3826802
Data columns (total 5 columns):
 #   Column         Dtype 
---  ------         ----- 
 0   profile_id     int64 
 1   date_id        object
 2   time_id        object
 3   location_id    int64 
 4   person_victim  object
dtypes: int64(2), object(3)
memory usage: 96.3+ MB
```

In [ ]:

```
# Resert index
person_victim_fact.reset_index(drop = True)
```

Out[ ]:

	profile_id	date_id	time_id	location_id	person_victim
0	8	20210911	0935	1	3cb21800-426f-47c8-a79e-fb65f2d2115e
1	8	20210911	0935	1	d7bbe88a-d44d-4155-8076-923b24b371be
2	9	20211214	0813	2	49b837fa-d00c-40a2-8af1-31ea2ce302f2
3	12	20211214	0817	3	a1699487-c586-4e1b-bf8c-a08cf53e331a
4	13	20211214	0817	3	1519c5d0-94a1-4dde-9f62-5311797ec6a2
...	...	...	...	...	...
2103788	353	20221114	1945	205618	535a8dff-f5e1-48e7-a411-a28ec4ac864b
2103789	250	20221114	1945	205618	299d9493-c2a5-4fd0-bf9f-85a0d274f774
2103790	353	20221114	1945	205618	dde4876b-1e50-4e98-8116-5122ae300825
2103791	258	20221115	1540	4919	a81936cc-c68b-468e-ac68-37265d808db6
2103792	258	20221115	1540	4919	da03992b-c91b-4973-81e2-f9757328913e

2103793 rows × 5 columns

ETL - Load data

Deliver Facts and Dimensions to Data Warehouse (BigQuery)

In [ ]:

```
# create a function to load dataframes to BigQuery

def load_table_to_bigquery(df,
                           table_name,
                           dataset_id):

    dataset_id = dataset_id #change dataset id to match your project id

    dataset_ref = bigquery_client.dataset(dataset_id)
    job_config = bigquery.LoadJobConfig()
    job_config.autodetect = True
    job_config.write_disposition = "WRITE_TRUNCATE"

    upload_table_name = f"{dataset_id}.{table_name}"

    load_job = bigquery_client.load_table_from_dataframe(df,
                                                         upload_table_name,
                                                         job_config = job_config)

    print(f"completed job {load_job} for table {table_name}")
```

In [ ]:

```
#Load each table to BigQuery: "collision_fact", "person_victim_fact", date_dim", "location_dim", "time_dim

#create an object with each table and their names
tables_objects = [[collision_fact, "collision_fact"],[person_victim_fact, "person_victim_fact"],
                  [date_dim,"date_dim" ],[location_dim,"location_dim"], [time_dim,"time_dim"],
                  [profile_dim,"profile_dim"]
                  ]
```

In [ ]:

```
#use a loop to load all the tables with the function "load_table_to_bigquery"
for table in tables_objects:
    load_table_to_bigquery(df = table[0],
                           table_name = table[1],
                           dataset_id = dataset_id)
```

```
completed job LoadJob<project=deft-stratum-361822, location=US, id=decaad7b-192a-41ca-a7b3-64794594b732> f
or table collision_fact
completed job LoadJob<project=deft-stratum-361822, location=US, id=80f7d62c-df53-4bff-85ec-d52d4b833adf> f
or table person_victim_fact
completed job LoadJob<project=deft-stratum-361822, location=US, id=1301e08d-1cb2-4142-85cc-ff1489851c99> f
or table date_dim
completed job LoadJob<project=deft-stratum-361822, location=US, id=cc2e780b-2524-49c4-9f6b-ba7fcc40d9e4> f
or table location_dim
completed job LoadJob<project=deft-stratum-361822, location=US, id=fa5976c9-5374-4bd7-92d6-e08f930ec05a> f
or table time_dim
completed job LoadJob<project=deft-stratum-361822, location=US, id=538bb37b-f0bf-47ae-8fca-9c17cf8f9037> f
or table profile_dim
```

Screenshots of database tables

Date dimension

In [ ]:

```
Image("/content/drive/MyDrive/project_DW/date_dim.png")
```

Out[ ]:

date_dim

date_dim QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW

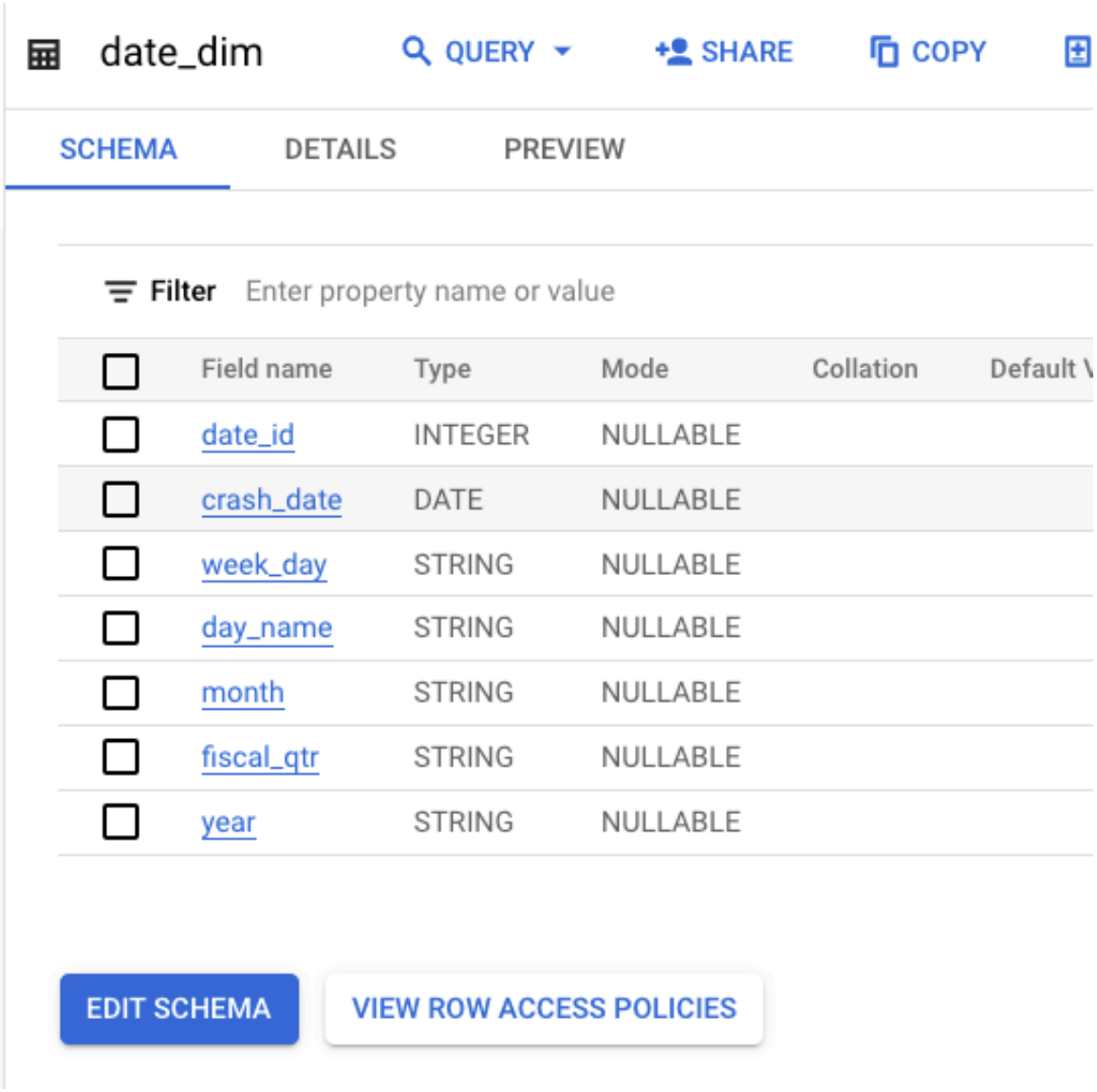
Row	date_id	crash_date	week_day	day_name	month	fiscal_qtr	year
1	20130505	2013-05-05	0	Sunday	May	2	2013
2	20130512	2013-05-12	0	Sunday	May	2	2013
3	20130519	2013-05-19	0	Sunday	May	2	2013
4	20130526	2013-05-26	0	Sunday	May	2	2013
5	20140504	2014-05-04	0	Sunday	May	2	2014
6	20140511	2014-05-11	0	Sunday	May	2	2014
7	20140518	2014-05-18	0	Sunday	May	2	2014
8	20140525	2014-05-25	0	Sunday	May	2	2014
9	20150503	2015-05-03	0	Sunday	May	2	2015
10	20150510	2015-05-10	0	Sunday	May	2	2015
11	20150517	2015-05-17	0	Sunday	May	2	2015
12	20150524	2015-05-24	0	Sunday	May	2	2015
13	20150531	2015-05-31	0	Sunday	May	2	2015
14	20160501	2016-05-01	0	Sunday	May	2	2016
15	20160508	2016-05-08	0	Sunday	May	2	2016
16	20160515	2016-05-15	0	Sunday	May	2	2016
17	20160522	2016-05-22	0	Sunday	May	2	2016
18	20160529	2016-05-29	0	Sunday	May	2	2016

Results per page: 50 1 – 50 of 4202

In [ ]:

```
Image("/content/drive/MyDrive/project_DW/date_dim_types.png")
```

Out[ ]:



Time dimension

In [ ]:

```
Image("/content/drive/MyDrive/project_DW/time_dim.png")
```

Out[ ]:

time_dim

time_dim QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW

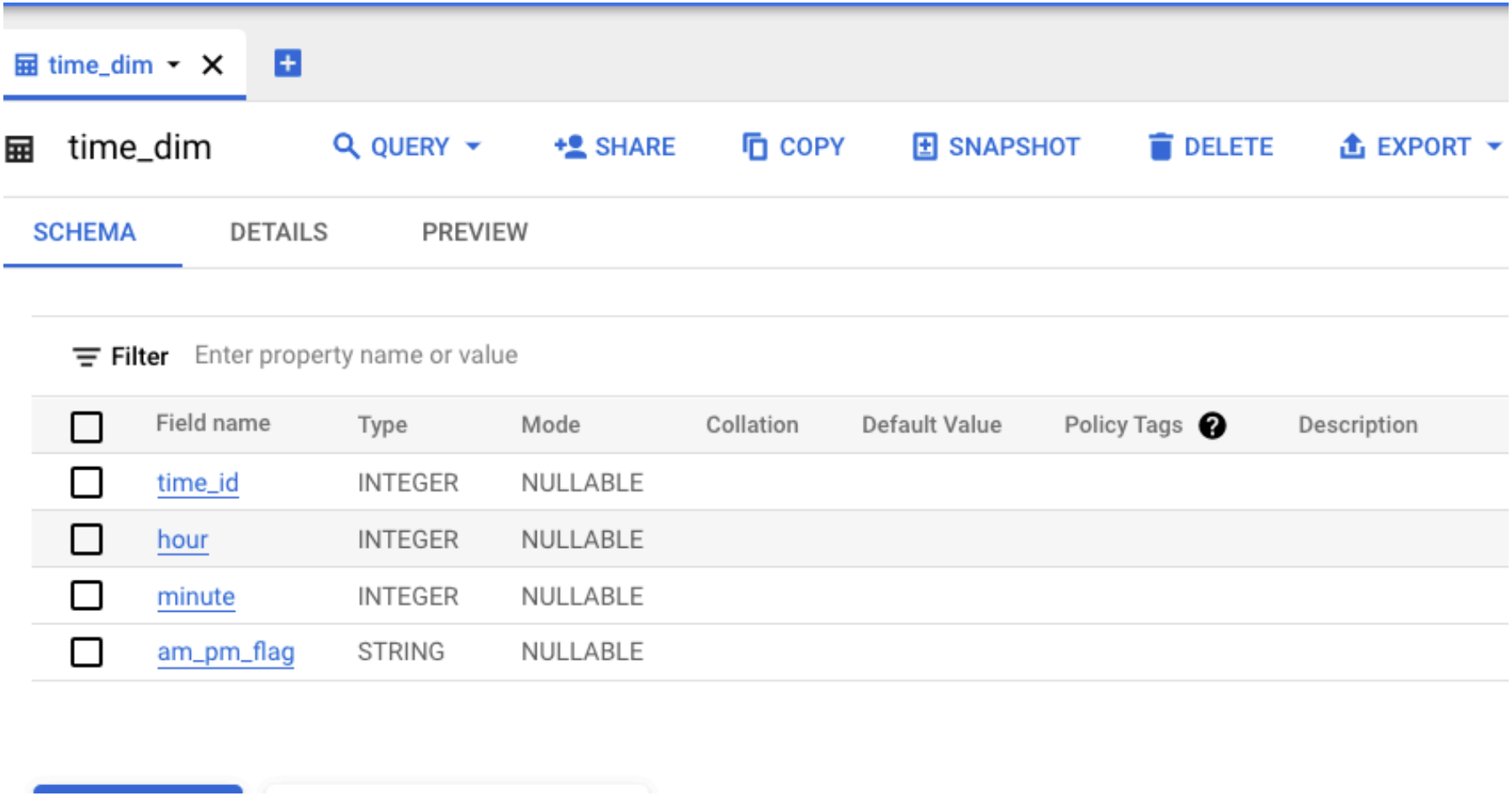
Row	time_id	hour	minute	am_pm_flag
1	0	0	0	AM
2	1	0	1	AM
3	2	0	2	AM
4	3	0	3	AM
5	4	0	4	AM
6	5	0	5	AM
7	6	0	6	AM
8	7	0	7	AM
9	8	0	8	AM
10	9	0	9	AM
11	10	0	10	AM
12	11	0	11	AM
13	12	0	12	AM
14	13	0	13	AM
15	14	0	14	AM
16	15	0	15	AM
17	16	0	16	AM
18	17	0	17	AM

Results per page: 50 1 – 50 of 1440

In [ ]:

```
Image("/content/drive/MyDrive/project_DW/time_dim_types.png")
```

Out[ ]:



Location dimension

In [ ]:

```
Image("/content/drive/MyDrive/project_DW/location_dim.png")
```

Out[ ]:

location_dim

location_dim QUERY SHARE COPY SNAPSHOT DELETE EXPORT

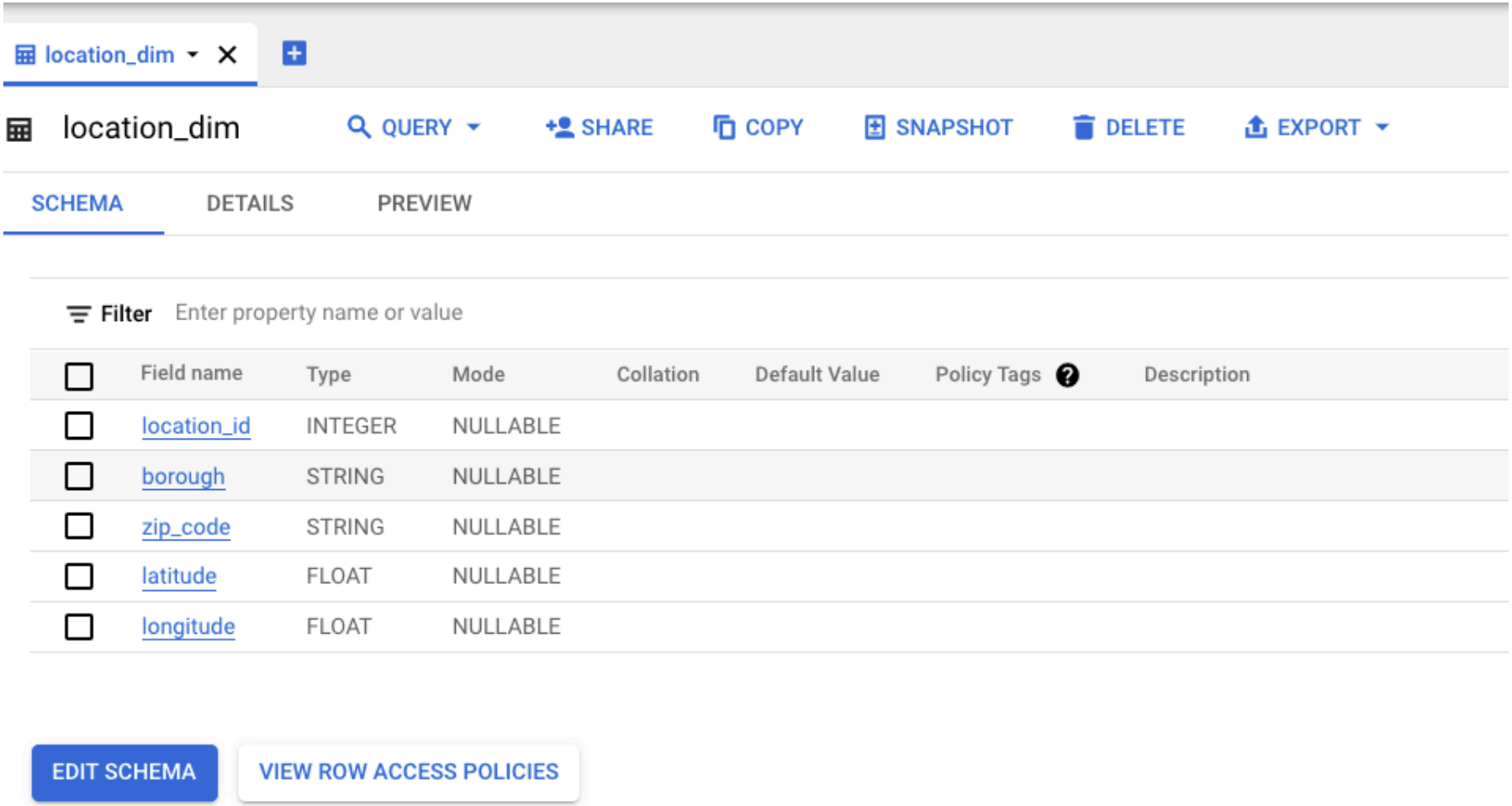
SCHEMA DETAILS PREVIEW

Row	location_id	borough	zip_code	latitude	longitude
1	284	BRONX	10451	40.817696	-73.922615
2	332	BRONX	10451	40.817627	-73.92366
3	521	BRONX	10451	40.827824	-73.91934
4	716	BRONX	10451	40.825455	-73.91317
5	809	BRONX	10451	40.813663	-73.931244
6	826	BRONX	10451	40.81934	-73.93012
7	882	BRONX	10451	40.81655	-73.91955
8	898	BRONX	10451	40.821667	-73.915184
9	955	BRONX	10451	40.818012	-73.92519
10	976	BRONX	10451	40.817387	-73.92277
11	1256	BRONX	10451	40.820198	-73.921486
12	1472	BRONX	10451	40.81693	-73.921036
13	1525	BRONX	10451	40.822304	-73.91485
14	1538	BRONX	10451	40.823658	-73.91206
15	1598	BRONX	10451	40.824806	-73.91352
16	1645	BRONX	10451	40.81738	-73.9257
17	1900	BRONX	10451	40.823383	-73.91834
18	2009	BRONX	10451	40.819202	-73.92597

In [ ]:

```
Image("/content/drive/MyDrive/project_DW/location_dim_types.png")
```

Out[ ]:



Profile dimension

In [ ]:
```
Image("/content/drive/MyDrive/project_DW/profile_dim.png")
```

Out[ ]:

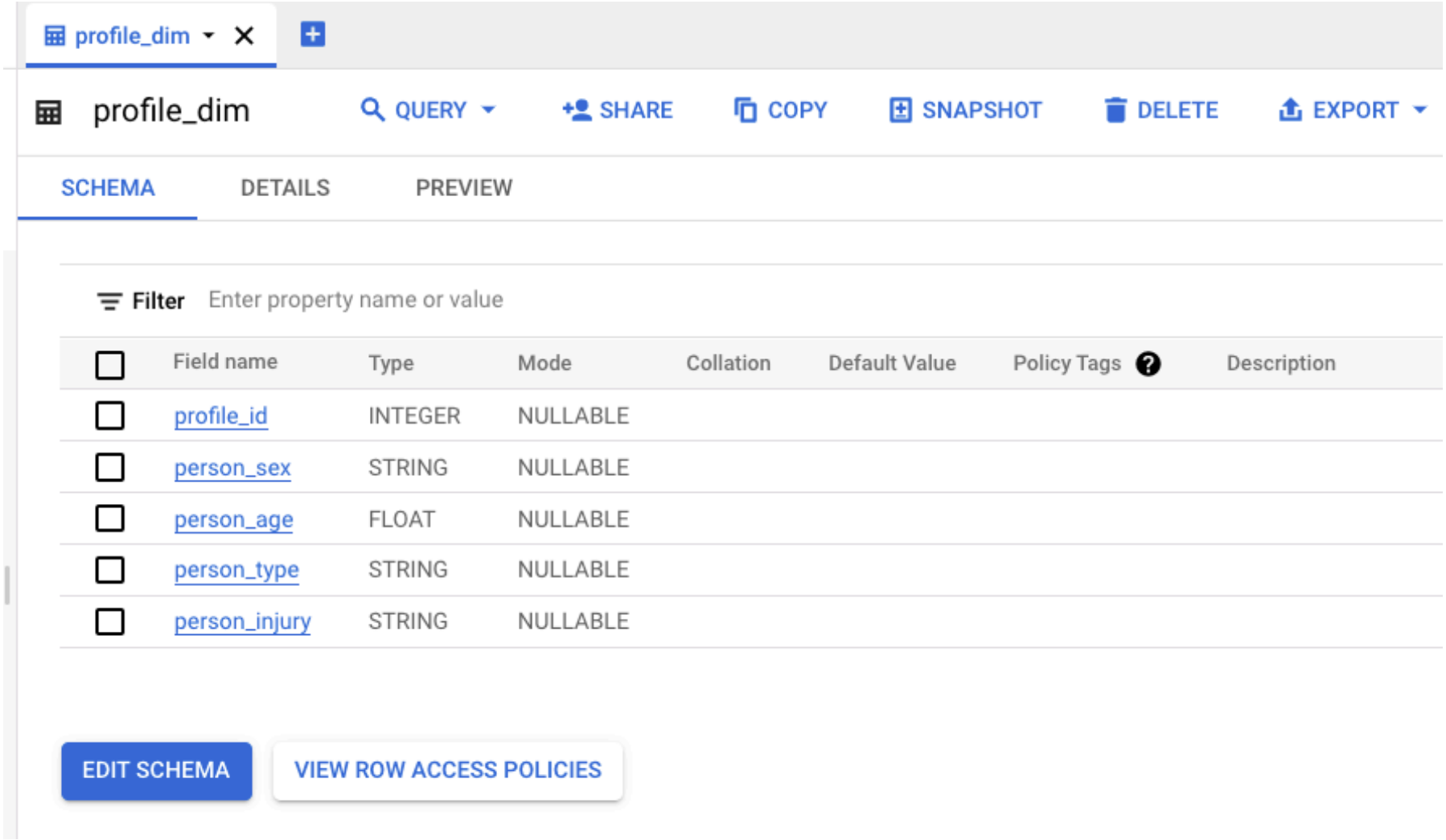
SCHEMA DETAILS PREVIEW

Row	profile_id	person_sex	person_age	person_type	person_injury
1	200	F	1	Occupant	Unspecified
2	238	F	1	Occupant	Injured
3	846	F	1	Pedestrian	Injured
4	1077	F	1	Bicyclist	Injured
5	1167	F	1	Pedestrian	Unspecified
6	1632	F	1	Pedestrian	Killed
7	214	F	2	Occupant	Unspecified
8	466	F	2	Pedestrian	Injured
9	621	F	2	Occupant	Injured
10	869	F	2	Other Motorized	Injured
11	940	F	2	Pedestrian	Unspecified
12	1042	F	2	Bicyclist	Injured
13	1397	F	2	Pedestrian	Killed
14	1610	F	2	Bicyclist	Unspecified
15	85	F	3	Occupant	Unspecified
16	145	F	3	Occupant	Injured
17	448	F	3	Pedestrian	Injured
18	1172	F	3	Pedestrian	Unspecified
19	1734	F	3	Bicyclist	Injured
20	2051	F	3	Occupant	Killed
21	2108	F	3	Pedestrian	Killed
22	361	F	4	Occupant	Unspecified
23	457	F	4	Other Motorized	Injured
24	725	F	4	Pedestrian	Injured

Results per page: 50 1 – 50 of 2131

In [ ]:
```
Image("/content/drive/MyDrive/project_DW/profile_dim_types.png")
```

Out[ ]:



Collision_fact table

In [ ]:

```
Image("/content/drive/MyDrive/project_DW/collision_fact.png")
```

Out[ ]:

collision_fact

QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW

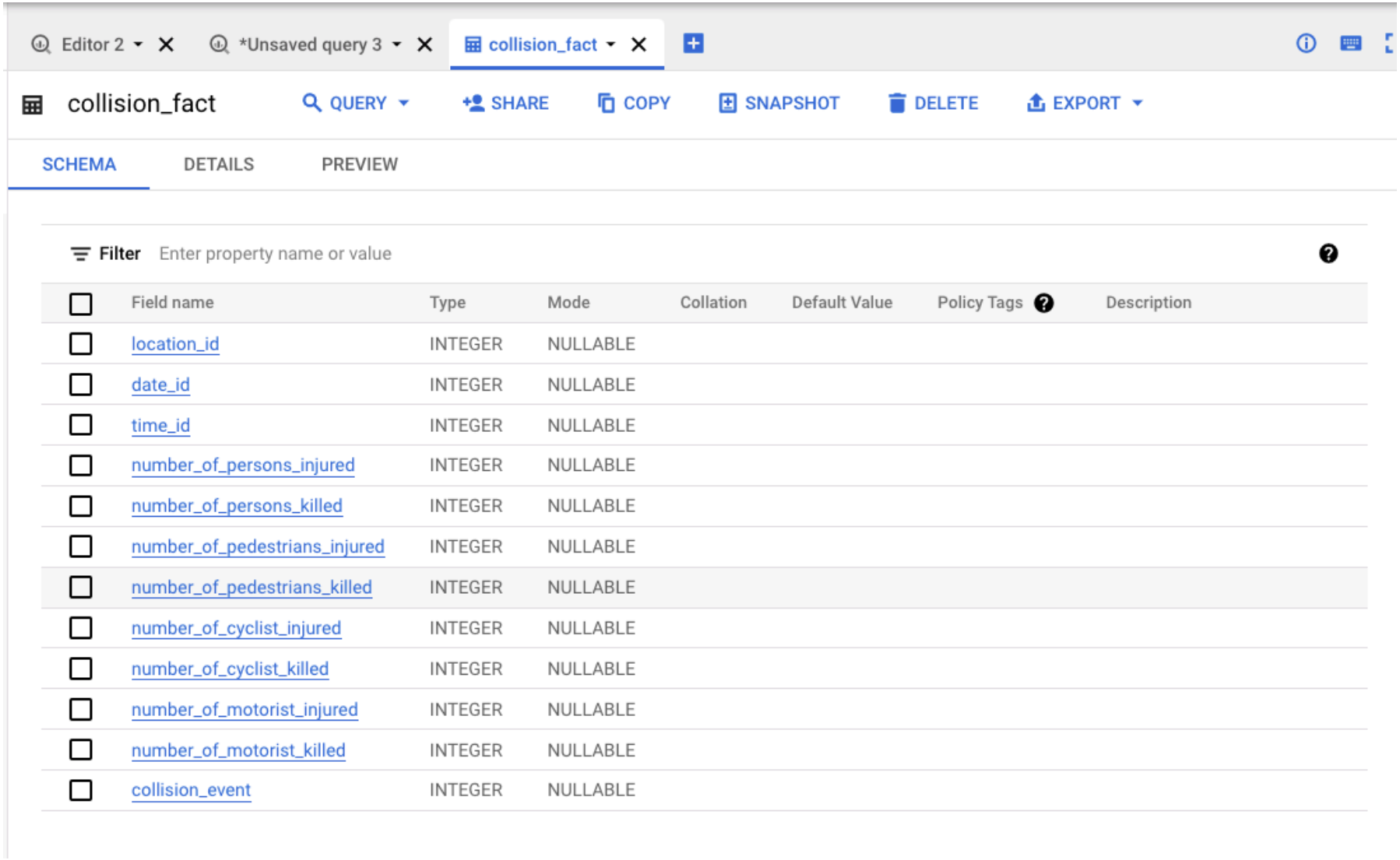
Row	location_id	date_id	time_id	number_of_pers	number_of_pers	number_of_pede	number_of_pede	number_of_cycli	number_of_cycli
1	21	20220424	0	0	0	0	0	0	0
2	100	20220325	0	0	0	0	0	0	0
3	157	20210911	0	0	0	0	0	0	0
4	262	20210709	0	0	0	0	0	0	0
5	322	20210414	0	0	0	0	0	0	0
6	345	20210414	0	0	0	0	0	0	0
7	353	20210415	0	0	0	0	0	0	0
8	433	20210416	0	0	0	0	0	0	0
9	448	20210415	0	0	0	0	0	0	0
10	489	20210414	0	0	0	0	0	0	0
11	537	20210414	0	0	0	0	0	0	0
12	687	20210416	0	0	0	0	0	0	0
13	706	20210414	0	0	0	0	0	0	0
14	757	20210911	0	0	0	0	0	0	0
15	777	20210706	0	0	0	0	0	0	0
16	820	20210725	0	0	0	0	0	0	0
17	885	20210707	0	0	0	0	0	0	0
18	907	20210417	0	0	0	0	0	0	0
19	919	20210417	0	0	0	0	0	0	0

Results per page: 50 1 – 50 of 790354

In [ ]:

```
Image("/content/drive/MyDrive/project_DW/collision_fact_types.png")
```

Out[ ]:



Field name	Type	Mode	Collation	Default Value	Policy Tags	Description
location_id	INTEGER	NULLABLE				
date_id	INTEGER	NULLABLE				
time_id	INTEGER	NULLABLE				
number_of_persons_injured	INTEGER	NULLABLE				
number_of_persons_killed	INTEGER	NULLABLE				
number_of_pedestrians_injured	INTEGER	NULLABLE				
number_of_pedestrians_killed	INTEGER	NULLABLE				
number_of_cyclist_injured	INTEGER	NULLABLE				
number_of_cyclist_killed	INTEGER	NULLABLE				
number_of_motorist_injured	INTEGER	NULLABLE				
number_of_motorist_killed	INTEGER	NULLABLE				
collision_event	INTEGER	NULLABLE				

Person_victim_fact table

In [ ]:

```
Image("/content/drive/MyDrive/project_DW/person_victim_fact.png")
```

Out[ ]:

person_victim_fact

person_victim_fact QUERY SHARE COPY SNAPSHOT DELETE EXPORT

SCHEMA DETAILS PREVIEW

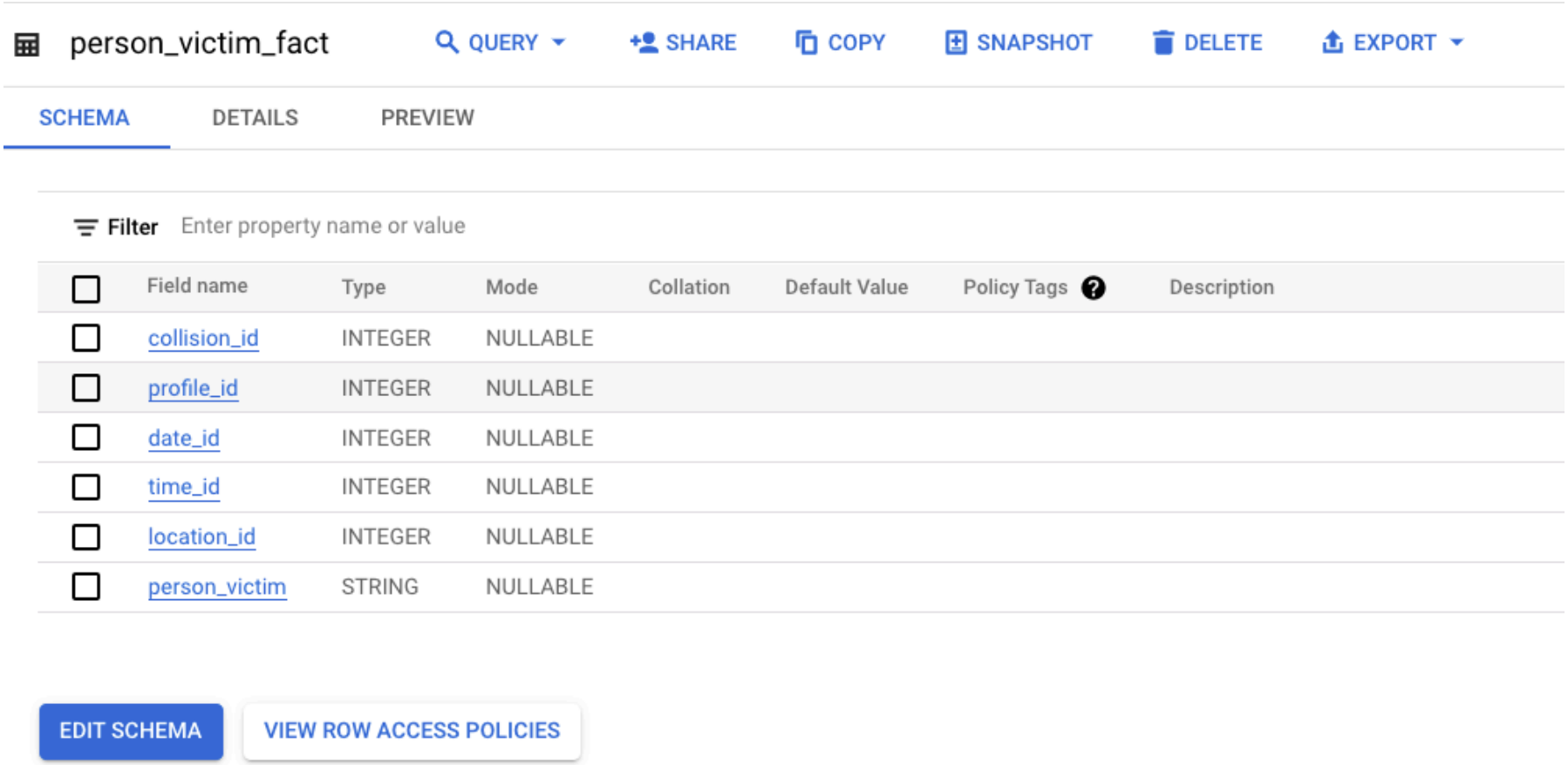
Row	collision_id	profile_id	date_id	time_id	location_id	person_victim
1	5	9	20211214	8130	2	49b837fa-d00c-40a2-8af1-31e...
2	8	12	20211214	8170	3	a1699487-c586-4e1b-bf8c-a08...
3	8	14	20211214	8170	3	72b05818-8e92-481a-ae72-9c0...
4	11	20	20211214	1650	5	501ef4bb-7180-4943-9d47-345...
5	11	20	20211214	1650	5	66475e18-b46d-4240-9480-89...
6	14	24	20211214	2310	6	d3bfea67-0a48-4510-9370-1c4...
7	14	25	20211214	2310	6	543a8ff2-4efd-48f1-8fb9-780d...
8	14	26	20211214	2310	6	cca678b5-8845-4ab2-b752-695...
9	16	28	20211214	2003	8	b1ad3506-709e-4ec8-99ce-afb...
10	16	29	20211214	2003	8	6aa9cdbb-6d28-40b8-9e05-88...
11	16	30	20211214	2003	8	1026933c-44fd-4065-832c-cb8...
12	16	31	20211214	2003	8	0833cb42-b0c8-4884-b3c2-218...
13	18	34	20211211	1943	9	d51ec997-f864-4f1c-85c1-7ab...
14	18	35	20211211	1943	9	a1cc5400-72f7-4b78-85b8-af7...
15	18	36	20211211	1943	9	4e195903-5981-4751-b3e7-e1e...
16	18	34	20211211	1943	9	e4abfc6b-e00d-4870-8cac-a6d...
17	22	42	20211213	1740	10	c23fbe63-963e-4dc4-a5b6-c42...
18	22	43	20211213	1740	10	56a57cf9-70ec-4697-a347-352...

Results per page: 50 1 – 50 of 2234864

In [ ]:

```
Image("/content/drive/MyDrive/project_DW/person_victim_fact_types.png")
```

Out[ ]:



References

- ETL Pipeline tutorial by Michael O'Donnell (CIS 9440 Data Warehousing and Analytics).
- Track your loop using tqdm: 7 ways progress bars in Python make things easier: https://medium.com/@harshit4084/track-your-loop-using-tqdm-7-ways-progress-bars-in-python-make-things-easier-fcbbb9233f24