**use the above code to find the count calls by priority, emergency type**

# Save the modified script as count\_calls.py

from mrjob.job import MRJob

import csv

class EmergencyCounts(MRJob):

# Mapper function which includes splitting and yielding the priority and emergency type counts

def mapper(self, \_, line):

# Splitting the data separated by comma

data = line.split(',')

priority = data[3] # Assuming priority is in the fourth column

emergency\_type = data[-1].strip() # Assuming emergency type is in the last column

# Yielding the priority and emergency type

yield (priority, emergency\_type), 1

# Reducer function to find the count value for each priority and emergency type

def reducer(self, key, values):

# Summing up the counts for each priority and emergency type

count = sum(values)

# Yield the priority and emergency type along with their counts

yield key, count

if \_\_name\_\_ == '\_\_main\_\_':

EmergencyCounts.run()

**REDUCER**

import count\_calls

mr\_job = count\_calls.EmergencyCounts(args=['path/to/your/911/data.csv'])

with mr\_job.make\_runner() as runner:

runner.run()

for key, value in mr\_job.parse\_output(runner.cat\_output()):

print(key, value)

**use the above code to find the count calls by zip code,district,police district,council district**

# Modify the mapper to extract the relevant columns and yield counts for each category

def mapper(self, \_, line):

# Splitting the data separated by comma

data = line.split(',')

zip\_code = data[-3] # Assuming ZIP code is the third-to-last column

district = data[4] # Assuming district is the fifth column

police\_district = data[10] # Assuming police district is the eleventh column

council\_district = data[11] # Assuming council district is the twelfth column

# Yield counts for each category

yield ('ZIP Code', zip\_code), 1

yield ('District', district), 1

yield ('Police District', police\_district), 1

yield ('Council District', council\_district), 1

# Modify the reducer to sum the counts for each category

def reducer(self, category, counts):

total\_count = sum(counts)

# Yield the category and its total count

yield category, total\_count

**use the above code to find the count calls by time which has time format in the form 2021/01/04 16:33:00+00 by month**

**Mapper**

# Import required libraries

from mrjob.job import MRJob

from datetime import datetime

import csv

class Monthly911CallCounts(MRJob):

# Mapper function to extract month and yield the count for each month

def mapper(self, \_, line):

data = line.split(',')

# Assuming the timestamp is in the second column (adjust if needed)

timestamp\_str = data[1]

timestamp = datetime.strptime(timestamp\_str, "%Y/%m/%d %H:%M:%S+00")

month = timestamp.strftime("%Y-%m") # Extract year and month in the format YYYY-MM

# Yield the month and count as 1 for each record

yield month, 1

# Reducer function to sum the counts for each month

def reducer(self, month, counts):

total\_calls = sum(counts)

yield month, total\_calls

if \_\_name\_\_ == '\_\_main\_\_':

Monthly911CallCounts.run()

**REDUCER**

import count\_values

mr\_job = count\_values.Monthly911CallCounts(args=['path/to/your/911/data.csv'])

with mr\_job.make\_runner() as runner:

runner.run()

for key, value in mr\_job.parse\_output(runner.cat\_output()):

print(key, value)