# Allows us to connect to the data source and pulls the information

from sodapy import Socrata

import requests

from requests.auth import HTTPBasicAuth

import json

import argparse

import sys

import os

import math

# Creates a parser. Parser is the thing where you add your arguments.

parser = argparse.ArgumentParser(description='Fire Incident Dispatch Data')

# In the parse, we have two arguments to add.

# The first one is a required argument for the program to run. If page\_size is not passed in, don’t let the program to run

parser.add\_argument('--page\_size', type=int, help='how many rows to get per page', required=True)

# The second one is an optional argument for the program to run. It means that with or without it your program should be able to work.

parser.add\_argument('--num\_pages', type=int, help='how many pages to get in total')

# Take the command line arguments passed in (sys.argv) and pass them through the parser.

# Then you will ned up with variables that contains page size and num pages.

args = parser.parse\_args(sys.argv[1:])

print(args)

DATASET\_ID=os.environ["DATASET\_ID"]

APP\_TOKEN=os.environ["APP\_TOKEN"]

ES\_HOST=os.environ["ES\_HOST"]

ES\_USERNAME=os.environ["ES\_USERNAME"]

ES\_PASSWORD=os.environ["ES\_PASSWORD"]

INDEX\_NAME=os.environ["INDEX\_NAME"]

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

try:

#Using requests.put(), we are creating an index (db) first.

resp = requests.put(f"{ES\_HOST}/{INDEX\_NAME}", auth=HTTPBasicAuth(ES\_USERNAME, ES\_PASSWORD),

json={

"settings": {

"number\_of\_shards": 1,

"number\_of\_replicas": 1

},

#We are specifying the columns and define what we want the data to be.

#However, it is not guaranteed that the data will come us clean.

#We will might need to clean it in the next steps.

#If the data you're pushing to the Elasticsearch is not compatible with these definitions,

#you'll either won't be able to push the data to Elasticsearch in the next steps

#and get en error due to that or the columns will not be usable in Elasticsearch

"mappings": {

"properties": {

"starfire\_incident\_id": {"type": "keyword"},

"incident\_datetime": {"type": "date"},

"alarm\_box\_borough": {"type": "keyword"},

"incident\_classification": {"type": "keyword"},

"alarm\_source\_description\_tx": {"type": "keyword"},

"dispatch\_response\_seconds\_qy": {"type": "float"},

"incident\_response\_seconds\_qy": {"type": "float"},

"engines\_assigned\_quantity": {"type": "float"},

"ladders\_assigned\_quantity": {"type": "float"},

"zipcode": {"type": "float"}, #This should normally be considered for keyword

#but I need a numeric field for the next steps.

}

},

}

)

resp.raise\_for\_status()

print(resp.json())

#If we send another put() request after creating an index (first put() request), the pogram will give an error and crash.

#In order to avoid it, we use try and except here.

#If the index is already created, it will raise an excepion and the program will not be crashed.

except Exception as e:

print("Index already exists! Skipping")

client = Socrata("data.cityofnewyork.us", APP\_TOKEN, timeout=10000)

total = client.get(DATASET\_ID, select="COUNT(\*)")

max\_pagesize = math.ceil(int(total[0]["COUNT"])/int(args.page\_size))

print(max\_pagesize)

#insert for loop here.

if args.num\_pages is not None:

for args.num\_pages in range(1,args.num\_pages):

rows = client.get(DATASET\_ID, limit=args.page\_size, where="starfire\_incident\_id" != None and "incident\_datetime" != None, offset=(args.num\_pages-1)\*args.page\_size)

es\_rows=[]

for row in rows:

try:

# Convert

es\_row = {}

es\_row["starfire\_incident\_id"] = row["starfire\_incident\_id"]

es\_row["incident\_datetime"] = row["incident\_datetime"]

es\_row["alarm\_box\_borough"] = row["alarm\_box\_borough"]

es\_row["incident\_classification"] = row["incident\_classification"]

es\_row["alarm\_source\_description\_tx"] = row["alarm\_source\_description\_tx"]

es\_row["dispatch\_response\_seconds\_qy"] = row["dispatch\_response\_seconds\_qy"]

es\_row["incident\_response\_seconds\_qy"] = row["incident\_response\_seconds\_qy"]

es\_row["engines\_assigned\_quantity"] = row["engines\_assigned\_quantity"]

es\_row["ladders\_assigned\_quantity"] = row["ladders\_assigned\_quantity"]

es\_row["zipcode"] = float(row["zipcode"])

#print(es\_row)

#There might be still some bad data coming from the source

#For instance, incident\_zip might have N/A instead of numerice values.

#In this case, the conversion will not work and the program will crash.

#We do not want that. That's why we raise an exception here.

except Exception as e:

print (f"Error!: {e}, skipping row: {row}")

continue

es\_rows.append(es\_row)

#print(es\_rows)

bulk\_upload\_data = ""

for line in es\_rows:

#print(f'Handling row {line["starfire\_incident\_id"]}')

action = '{"index": {"\_index": "' + INDEX\_NAME + '", "\_type": "\_doc", "\_id": "' + line["starfire\_incident\_id"] + '"}}'

data = json.dumps(line)

bulk\_upload\_data += f"{action}\n"

bulk\_upload\_data += f"{data}\n"

try:

# Upload to Elasticsearch by creating a document

resp = requests.post(f"{ES\_HOST}/\_bulk",

# We upload es\_row to Elasticsearch

data=bulk\_upload\_data,auth=HTTPBasicAuth(ES\_USERNAME, ES\_PASSWORD), headers = {"Content-Type": "application/x-ndjson"})

resp.raise\_for\_status()

print ('Done')

#print(max\_pagesize)

except Exception as e:

print(f"Failed to insert in ES: {e}")

else:

for max\_pagesize in range(1,max\_pagesize):

rows = client.get(DATASET\_ID, limit=args.page\_size, where="starfire\_incident\_id" != None and "incident\_datetime" != None, offset=(max\_pagesize-1)\*args.page\_size)

es\_rows=[]

for row in rows:

try:

# Convert

es\_row = {}

es\_row["starfire\_incident\_id"] = row["starfire\_incident\_id"]

es\_row["incident\_datetime"] = row["incident\_datetime"]

es\_row["alarm\_box\_borough"] = row["alarm\_box\_borough"]

es\_row["incident\_classification"] = row["incident\_classification"]

es\_row["alarm\_source\_description\_tx"] = row["alarm\_source\_description\_tx"]

es\_row["dispatch\_response\_seconds\_qy"] = row["dispatch\_response\_seconds\_qy"]

es\_row["incident\_response\_seconds\_qy"] = row["incident\_response\_seconds\_qy"]

es\_row["engines\_assigned\_quantity"] = row["engines\_assigned\_quantity"]

es\_row["ladders\_assigned\_quantity"] = row["ladders\_assigned\_quantity"]

es\_row["zipcode"] = float(row["zipcode"])

#print(es\_row)

#There might be still some bad data coming from the source

#For instance, incident\_zip might have N/A instead of numerice values.

#In this case, the conversion will not work and the program will crash.

#We do not want that. That's why we raise an exception here.

except Exception as e:

print (f"Error!: {e}, skipping row: {row}")

continue

es\_rows.append(es\_row)

#print(es\_rows)

bulk\_upload\_data = ""

for line in es\_rows:

#print(f'Handling row {line["starfire\_incident\_id"]}')

action = '{"index": {"\_index": "' + INDEX\_NAME + '", "\_type": "\_doc", "\_id": "' + line["starfire\_incident\_id"] + '"}}'

data = json.dumps(line)

bulk\_upload\_data += f"{action}\n"

bulk\_upload\_data += f"{data}\n"

try:

# Upload to Elasticsearch by creating a document

resp = requests.post(f"{ES\_HOST}/\_bulk",

# We upload es\_row to Elasticsearch

data=bulk\_upload\_data,auth=HTTPBasicAuth(ES\_USERNAME, ES\_PASSWORD), headers = {"Content-Type": "application/x-ndjson"})

resp.raise\_for\_status()

print ('Done')

print(total)

except Exception as e:

print(f"Failed to insert in ES: {e}")