nicio**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**FACULTAD DE ELECTRICIDAD Y COMPUTACIÓN**

**Materia:** Sistemas de Base de Datos Avanzados **Paralelo:** 2

**Docente:** Ing. Vanessa Ivonne Echeverria Barzola

**Grupo:** 4

**Integrantes:**

* Dayse Joselyne Maroto Lema

**Proyecto del 1 Parcial**

**Enlace de Github:** <https://github.com/dmaroto98/MapReduce_MongoDB_ProyectoBDA.git>

**DATASET:** <https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks>

# Descripción del Dataset

El archivo "data.csv" contiene más de **170.000** canciones recopiladas de Spotify Web API. Contiene 19 columnas detalladas a continuación:

## Primary

* id (Id of track generated by Spotify) **column** **9**

## Numerical

Los datos mostrados a continuación son de tipo numérico.

* valence (Ranges from 0 to 1) **column** **1**
* year (Ranges from 1921 to 2020) **column** **2**
* acousticness (Ranges from 0 to 1) **column** **3**
* danceability (Ranges from 0 to 1) **column**  **5**
* duration\_ms (Integer typically ranging from 200k to 300k) **column**  **6**
* energy (Ranges from 0 to 1) **column** **7**
* instrumentalness (Ranges from 0 to 1) **column** **10**
* liveness (Ranges from 0 to 1) **column** **12**
* loudness (Float typically ranging from -60 to 0) **column** **13**
* popularity (Ranges from 0 to 100) **column** **16**
* speechiness (Ranges from 0 to 1) **column 18**
* tempo (Float typically ranging from 50 to 150) **column 19**

## ****Dummy****

Los datos mostrados a continuación son de tipo boleano.

* explicit (0 = No explicit content, 1 = Explicit content) **column 8**
* mode (0 = Minor, 1 = Major) **column** **14**

## ****Categorical****

Los datos mostrados a continuación son de tipo cadena.

* artists (List of artists mentioned) **column** **4**
* key (All keys on octave encoded as values ranging from 0 to 11, starting on C as 0, C# as 1 and so on…) **column** **11**
* name (Name of the song) **column** **15**
* release\_date (Date of release mostly in yyyy-mm-dd format, however precision of date may vary) **column 17**

Los campos subrayados son los que se utilizarán para realizar la tarea del MapReduce.

# Descripción del Código para operaciones CRUD

Para las operaciones CRUD se utilizó la librería de pymongo que nos proveé la herramienta de python 3.8 [3].

Para ejecutar los comandos debe posicionarse en la ruta de **.\proyectoBDA>**

## Create

### Comando para ejecutar por consola

python .\insercionDatos.py

### Código

from pymongo import MongoClient

#representa la direccion ip del localhost

ip='127.0.0.1'

# 27017 is the default port number for mongodb

port=27017

#nombre de la base de datos

dbName="db\_spotify"

#nombre de la coleccion que vamos a manipular dentro de nuestra base de datos

collectionName= 'Spotify'

#conexion con mongoDB

client= MongoClient(ip,port)

db=client[dbName]

#insercion de datos a la base de datos

#se usa el comando insert\_many para insertar varios datos a la vez

db[collectionName].insert\_many([

    {"valence":0.0594,"year":2020,"acousticness":0.232,"artists":"['Camila Cabello', 'Shawn Mendez']","danceability":0.99,"duration\_ms":74512,"energy":0.21100000000000002,"explicit":0,"id":"4BJqT0PrAfrxzMOxytFOIz","instrumentalness":0.878,"key":10,"liveness":0.665,"loudness":-20.096,"mode":1,"name":"Esta es una prueba","popularity":4,"release\_date":"1921","speechiness":0.0366,"tempo":80.954},

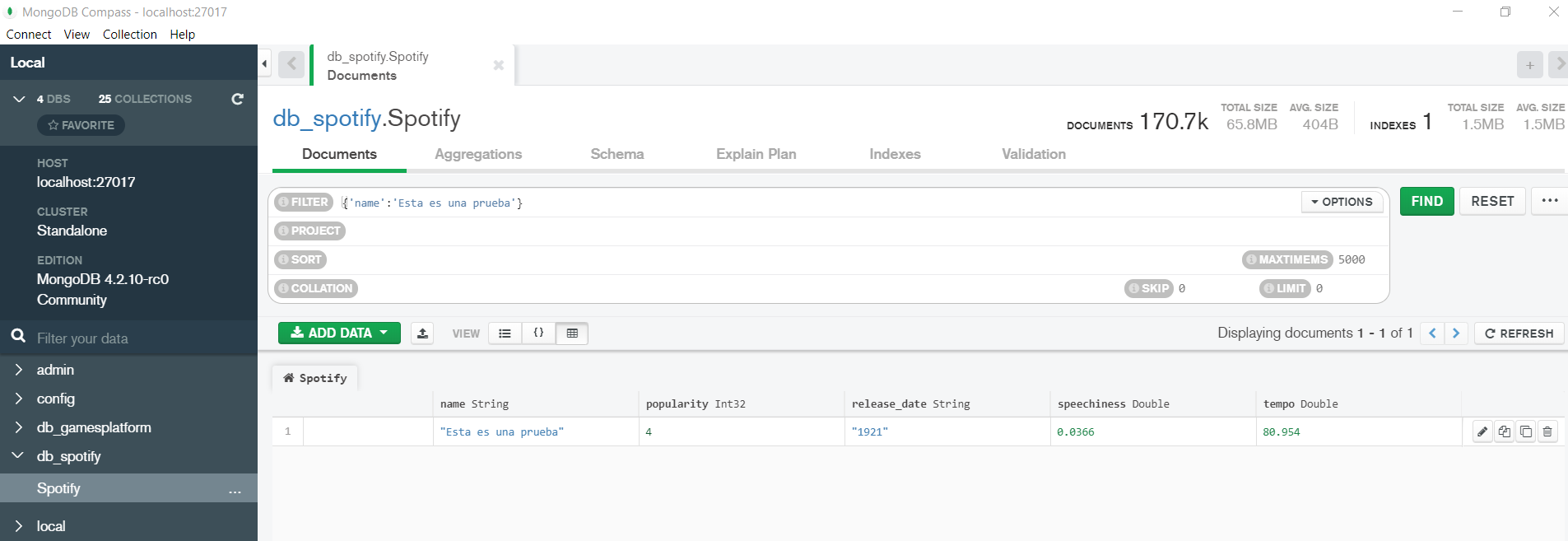
    {"valence":0.0594,"year":2020,"acousticness":0.562,"artists":"['Lauren Jauregui', 'Steven Aoki']","danceability":0.75,"duration\_ms":831667,"energy":0.21100000000000002,"explicit":0,"id":"4BJqT0PrAfrxzMOxytFOIz","instrumentalness":0.878,"key":10,"liveness":0.665,"loudness":-20.096,"mode":1,"name":"Esta es prueba 2","popularity":4,"release\_date":"1921","speechiness":0.0366,"tempo":80.954}

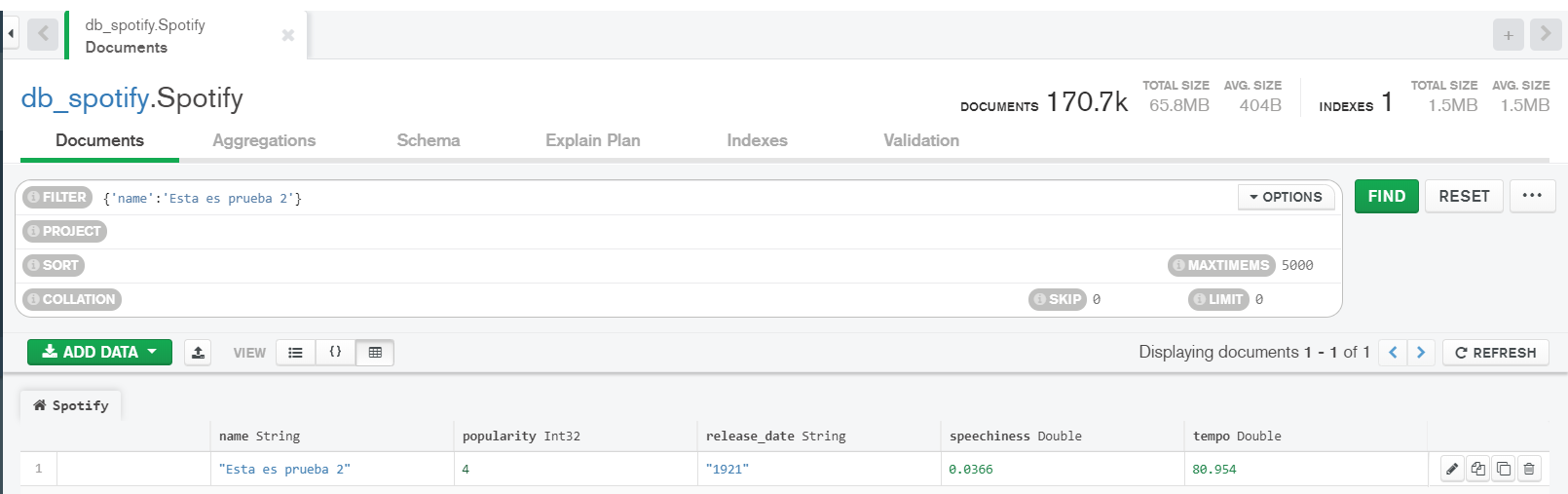
])

print("Inserción culminada con éxito")

### Resultado

A continuación, se puede observar la evidencia de la inserción de datos en la base de datos a través de la herramienta de MongoDB Compass [2].





## Read

Para poder realizar consultas en la base de datos MongoDB se debe de utilizar el comando find, al cual se le puede ingresar un query para una búsqueda en específico.

### Comando para ejecutar por consola

python .\consultaDatos.py

### Código

from pymongo import MongoClient

#representa la direccion ip del localhost

ip='127.0.0.1'

# 27017 is the default port number for mongodb

port=27017

#nombre de la base de datos

dbName="db\_spotify"

#nombre de la coleccion que vamos a manipular dentro de nuestra base de datos

collectionName= 'Spotify'

#conexion con mongoDB

client= MongoClient(ip,port)

db=client[dbName]

#lectura de datos a la base de datos

#almacenamos los datos en una variabla para luego proceder a imprimirlo por consola.

#se obtiene los datos cuyo campo "name" es igual a "Esta es una prueba"

result= db[collectionName].find({"name": "Esta es una prueba"})

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Primera consulta\*\*\*\*\*\*\*\*\*\*\*\*")

for i in result:

    print(i)

#se obtiene los datos cuyo campo "valence" es mayor a 0.99

result2= db[collectionName].find({"valence": { "$gt": 0.99 }})

print("")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Segunda consulta\*\*\*\*\*\*\*\*\*\*\*\*")

for j in result2:

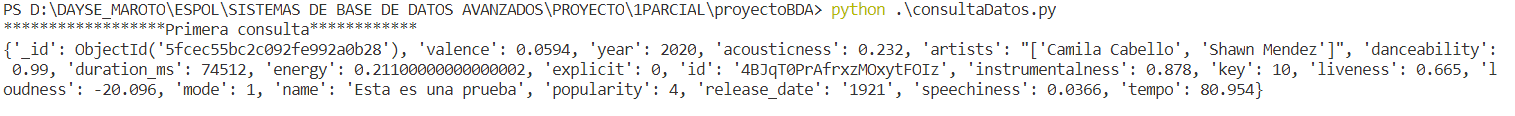
    print(j)

print("Lecturas culminadas con éxito")

### Resultado

Para la demostración de la consulta de datos se utilizaron 2 datos de prueba.

Aquí se extrajo la data cuyo nombre es “Esta es una prueba” y se lo imprime por consola.



Aquí se extrajo la data cuyos valores de “valence” sean mayores a 0.99 y se los imprime por consola [4].



## Update

### Comando para ejecutar por consola

python .\modificacionDatos.py

### Código

from pymongo import MongoClient

#representa la direccion ip del localhost

ip='127.0.0.1'

# 27017 is the default port number for mongodb

port=27017

#nombre de la base de datos

dbName="db\_spotify"

#nombre de la coleccion que vamos a manipular dentro de nuestra base de datos

collectionName= 'Spotify'

#conexion con mongoDB

client= MongoClient(ip,port)

db=client[dbName]

#modificacion de datos a la base de datos

#update\_one indica que solamente se actualizará un dato

db[collectionName].update\_one(

    {"name": "Esta es una prueba"}, #condicion para hacer la modificacion

    {

        "$set":{

            "valence": 0.1,  #valores que se modificarán

            "year": 2021

        }

    }

)

db[collectionName].update\_one(

    {"name": "Esta es prueba 2"},

    {

        "$set":{

            "valence": 0.7,

            "year": 2021

        }

    }

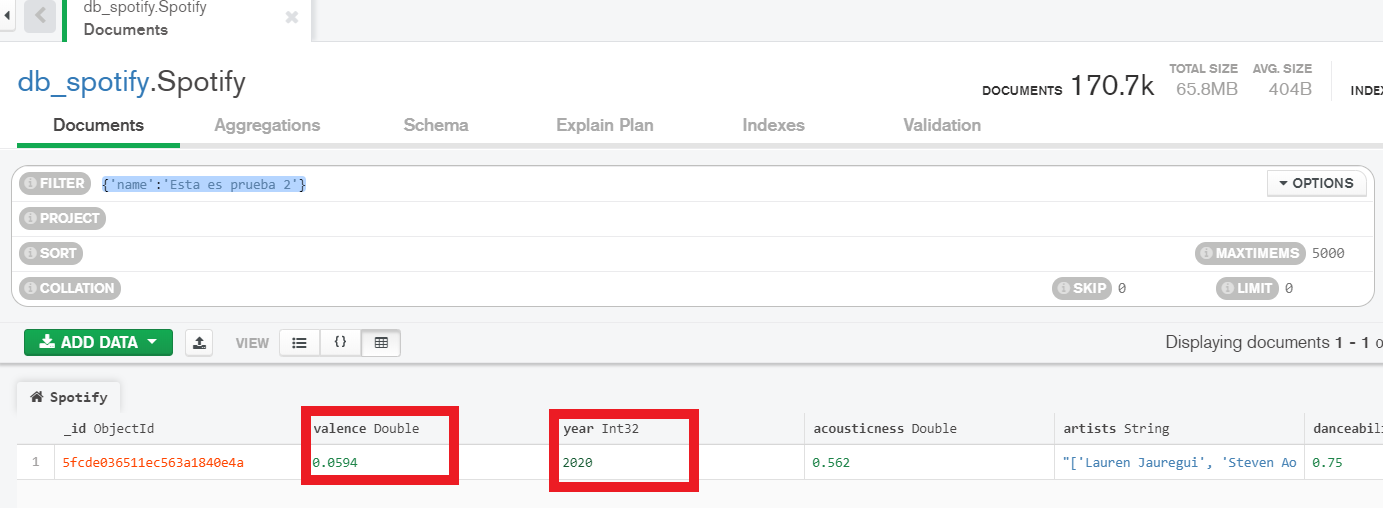
)

print("Modificación culminada con éxito")

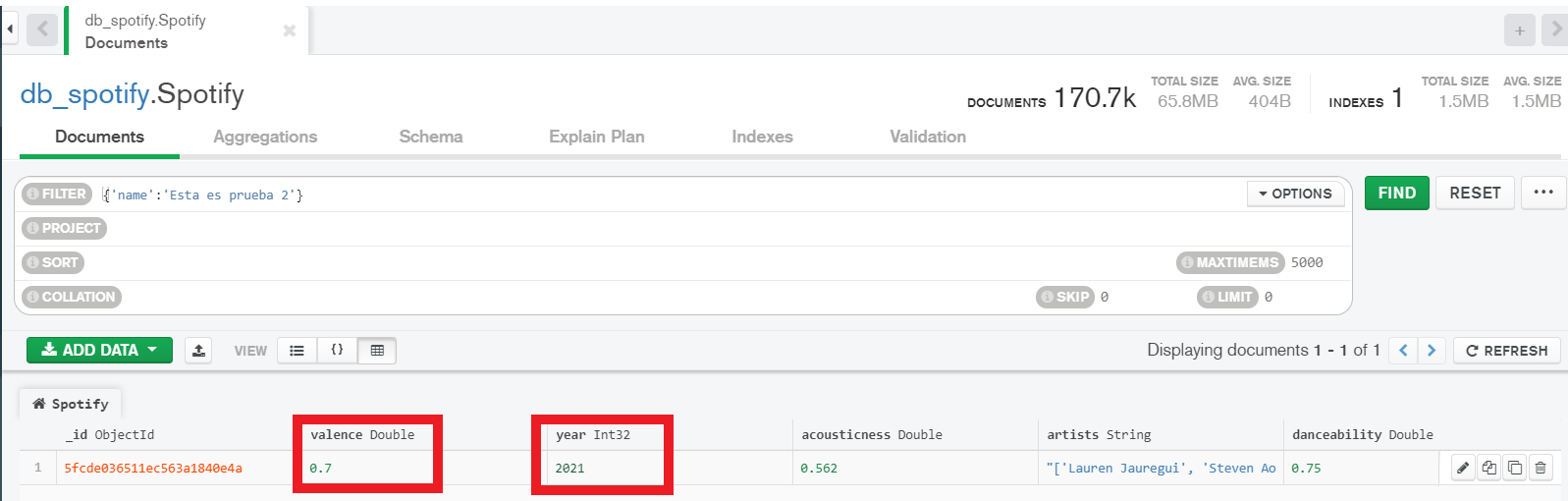
### Resultado

A continuación, se puede observar la evidencia de la modificación de datos en la base de datos a través de la herramienta de MongoDB Compass.

**Antes**



**Después**

****

## Delete

### Comando para ejecutar por consola

python .\eliminacionDatos.py

### Código

from pymongo import MongoClient

#representa la direccion ip del localhost

ip='127.0.0.1'

# 27017 is the default port number for mongodb

port=27017

#nombre de la base de datos

dbName="db\_spotify"

#nombre de la coleccion que vamos a manipular dentro de nuestra base de datos

collectionName= 'Spotify'

#conexion con mongoDB

client= MongoClient(ip,port)

db=client[dbName]

#eliminación de datos a la base de datos

db[collectionName].delete\_many(

    {"name":"Esta es una prueba"}

)

db[collectionName].delete\_many(

    {"year": 2021}

)

print("Eliminación culminada con éxito")

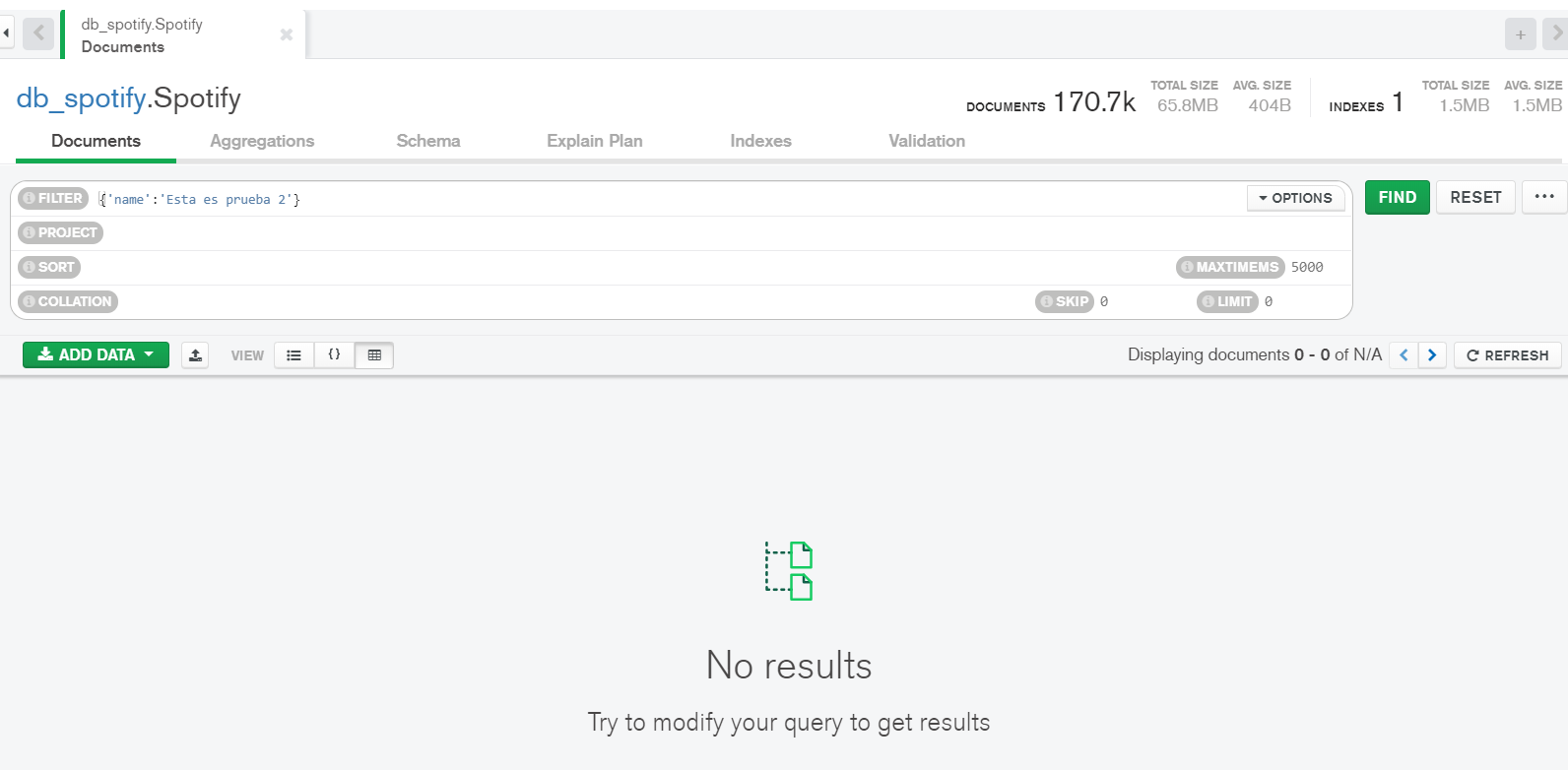
### Resultado

Para la demostración de la eliminación de datos se utilizaron los 2 datos creados anteriormente.

En la primera eliminación observamos que se eliminan los datos que en el campo “name” es igual a “Esta es una prueba”

En la segunda eliminación observamos que se eliminarán todos los datos que sean del año 2021.

Como evidencia observamos que ya no nos arroja datos con la consulta anteriormente utilizada.



# Descripción de la tarea a realizar con map-reduce.

Utilizando la base de datos de Spotify, se pretende conocer la cantidad de músicas que fueron lanzadas en el 2020, con una popularidad mayor a 30 y además que tenga el porcentaje de bailable mayor a 0.8; la función MapReduce se encargará de generar información del artista con las respectivas cantidad y nombre de canciones que ha creado.

Con la información generada se puede filtrar y conocer al artista que más canciones ha creado y mostrar la lista de sus canciones creadas.

# Descripción de la implementación realizada

Este proyecto utilizó el lenguaje de programación Python en conjunto con la base de datos no relacional MongoDB.

A continuación, se detalla los pasos para la implementación:

## Exportación de los datos en formato .csv a la base de datos MongoDB

### Comando para ejecutar por consola

python .\csv\_export\_to\_mongoDB.py

### Código

try:

    import pymongo

    from pymongo import MongoClient

    import pandas as pd

    import json

except Exception as e:

    print("some Modules are Missing")

class MongoDB(object):

    def \_\_init\_\_(self, dBName=None, collectionName=None):

        self.dBName = dBName

        self.collectionName = collectionName

        self.client = MongoClient("localhost", 27017, maxPoolSize=50)

        self.DB = self.client[self.dBName]

        self.collection = self.DB[self.collectionName]

    def InsertData(self, path=None):

        # :param path: Path del archivo csv

        # :return: None

        df = pd.read\_csv(path)

        data = df.to\_dict('records')

        self.collection.insert\_many(data, ordered=False)

        print("Toda la data ha sido exportada a Mongo DB Server")

if \_\_name\_\_ == "\_\_main\_\_":

    # se coloca un nombre para la base de datos de mongo, en este caso se llamará db\_spotify

    mongodb = MongoDB(dBName="db\_spotify", collectionName='Spotify')

    mongodb.InsertData(path="./data/data.csv")

## MapReduce

Se utilizó el archivo MapReduce.py provisto en el taller realizado en clases.

### Código

import json

class MapReduce:

    def \_\_init\_\_(self):

        self.intermediate = {}

        self.result = []

    def emit\_intermediate(self, key, value):

        self.intermediate.setdefault(key, [])

        self.intermediate[key].append(value)

    def emit(self, value):

        self.result.append(value)

    def execute(self, data, mapper, reducer):

        for line in data:

            record = json.loads(line)

            mapper(record)

        for key in self.intermediate:

            reducer(key, self.intermediate[key])

        #jenc = json.JSONEncoder(encoding='latin-1')

        jenc = json.JSONEncoder()

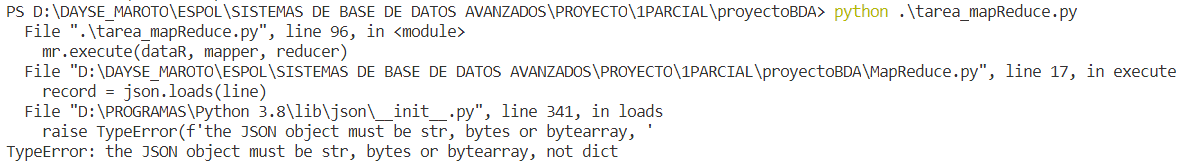
        for item in self.result:

            print (jenc.encode(item))

## Tarea del MapReduce

**Inconvenientes**

Al realizar la consulta a la base de datos de MongoDB, la información provista de la consulta realizada no se podía pasar directamente a la función MapReduce por lo que primero se generó el archivo **datagenerada.json**, el cual contiene la información del artista con la canción que ha creado para su respectivo análisis.



### Comando para ejecutar por consola

python .\tarea\_mapReduce.py

### Código

import MapReduce

import sys

from pymongo import MongoClient

from time import time

#representa la direccion ip del localhost

ip='127.0.0.1'

# 27017 is the default port number for mongodb

port=27017

#nombre de la base de datos

dbName="db\_spotify"

#nombre de la coleccion que vamos a manipular dentro de nuestra base de datos

collectionName= 'Spotify'

#conexion con mongoDB

client= MongoClient(ip,port)

db=client[dbName]

# instancia del objeto MapReduce

mr = MapReduce.MapReduce()

#se almacenarán todos los artistas que arroje la funcion de mapReduce

artistas=[]

#se almacenarán la cantidad de canciones que ha creado cada artista de acuerdo al resultado que arroje la funcion de mapReduce

cantidadCanciones=[]

canciones=[]

def mapper(record):

    # key: document identifier - artista

    # value: document contents - cancion que pertenece al artista

    key = record[0]

    value = record[1]

    mr.emit\_intermediate(key,value)

def reducer(key, list\_of\_values):

    # --- TU CODIGO AQUI ---

    #se quita las canciones que salen repetidas para obtener datos unicos

    value2=[]

    for cancion in list\_of\_values:

        if(cancion not in value2):

            value2.append(cancion)

    #se imprime el nombre del cantante con la lista de canciones que ha creado

    mr.emit((key,value2))

    artistas.append(key)

    cantidadCanciones.append(len(value2))

    canciones.append(value2)

    #se imprime el nombre del cantante con la cantidad de canciones que ha creado

    mr.emit((key,len(value2)))

if \_\_name\_\_ == '\_\_main\_\_':

    #cargamos nuestra colleccion Spotify de la base de datos de MongoDB al programa

    #filtramos la información, de tal forma que solo seleccionaremos las músicas que fueron

    # lanzadas en el año 2020;

    # una popularidad mayor a 30 y

    # porcentaje bailable mayor a 0.8

    result= db[collectionName].find({

        "year":{ "$gte": 2020 },

        "popularity":{ "$gt": 30 },

        "danceability":{ "$gt": 0.8 }

    })

    # En la variable result obtenmos un objeto tipo mongo, por lo que se debe recorrer ese objeto para ir agregando

    # su informacion a una lista

    # print (result)

    #Generaremos un archivo tipo .json que sera enviado a la funcion de mapReduce

    f= open("datagenerada.json","w+")

    cantidad\_contador=0

    for i in result:

        cantidad\_contador=cantidad\_contador+1

        columna\_artistas=i['artists']

        e=columna\_artistas[1:-1] #quitamos el simbolo "[" al inicio y "]" al final

        array\_artistas=e.split(",") #dividimos el string por "," para obtener cada artista

        # por cada artista se limpiara las comillas simples

        for artista in array\_artistas:

            nombre\_sin\_comilla=artista[1:-1]

            if (nombre\_sin\_comilla.find("'") !=-1):

                nombre\_sin\_comilla=artista[2:-1]

            # generamos el archivo de la data

            f.write("[\""+nombre\_sin\_comilla+"\",\""+i['name']+"\"]\n")

    #cerramos el archivo utilizado

    print("El resultado de la búsqueda realizada fue de: " +str(cantidad\_contador))

    f.close

    # Mandamos a ejecutar el Map Reduce

    inputdata = open("datagenerada.json")

    # Se toma el tiempo antes de ejecutar la funcion MapReduce

    start\_time = time()

    # Se ejecuta la función MapReduce

    mr.execute(inputdata, mapper, reducer)

    #Se toma el tiempo al finalizar la funcion MapReduce

    elapsed\_time = time() - start\_time

    # Se imprime el tiempo transcurrido

    print("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TIEMPO TRANSCURRIDO FUNCION MAPREDUCE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

    print("Elapsed time: %.10f seconds." % elapsed\_time)

    print("\n\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n")

# Se obtiene el valor maximo de la lista de cantidad de Canciones

maximo=max(cantidadCanciones)

# Se obtiene el índice del valor mayor de la lista de cantidad de canciones

indice= cantidadCanciones.index(maximo)

#se filtra las canciones que pertenecen al artista con mayor cantidad de canciones creadas

canciones\_artista\_ganador=canciones[indice]

print("El artista que más canciones ha creado es: "+ artistas[indice]+" con un total de "+str(maximo)+" canciones")

print("A continuación se muestra todas las canciones que ha creado: ")

contador=1

# Se muestran las canciones del artista ganador

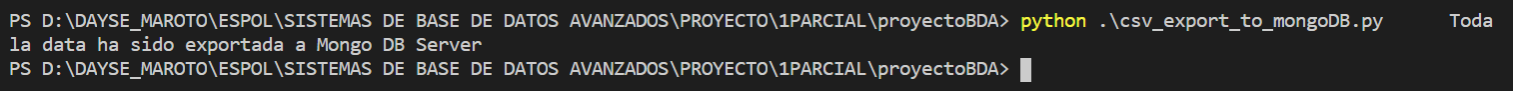
for cancion in canciones\_artista\_ganador:

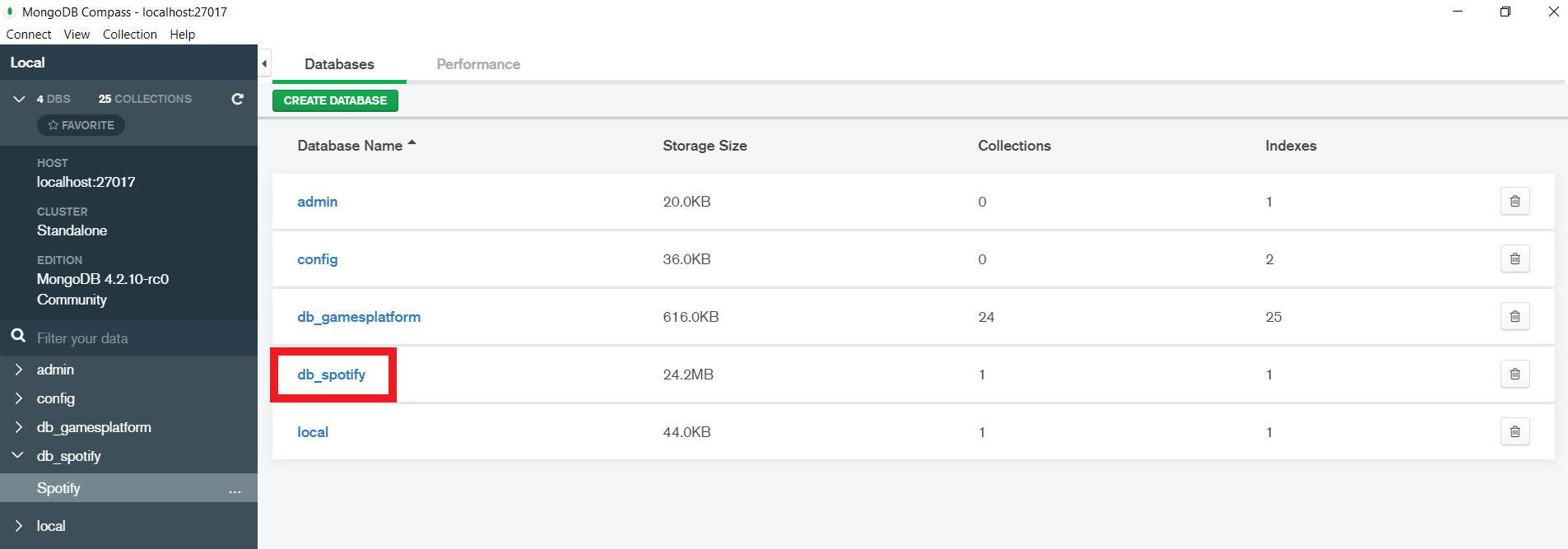
    print(str(contador) +".  "+ cancion)

    contador= contador+1

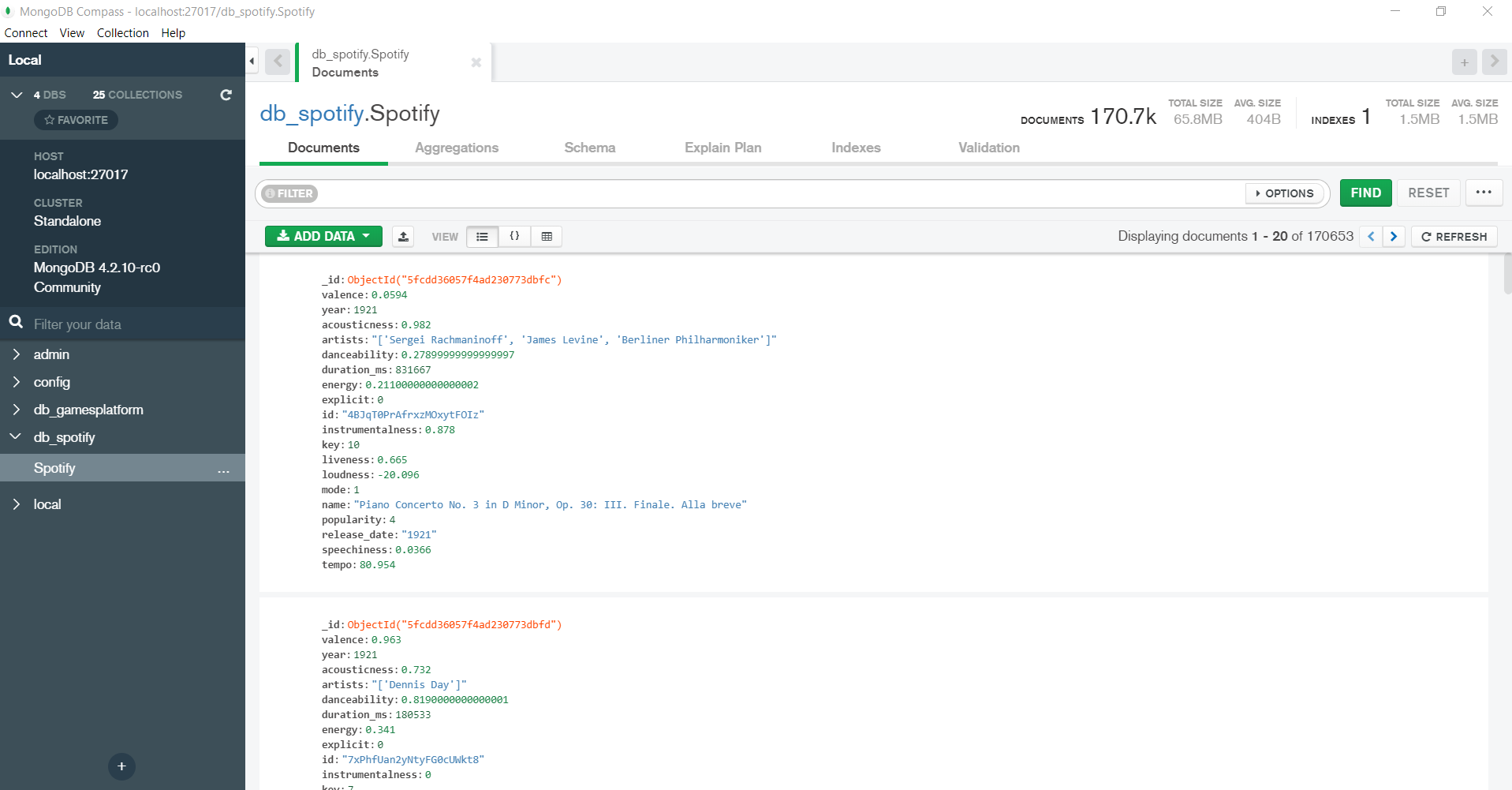
# Descripción de los resultados obtenidos.

Exportación de los datos del archivo data.csv a una base de datos de mongoDB [1].

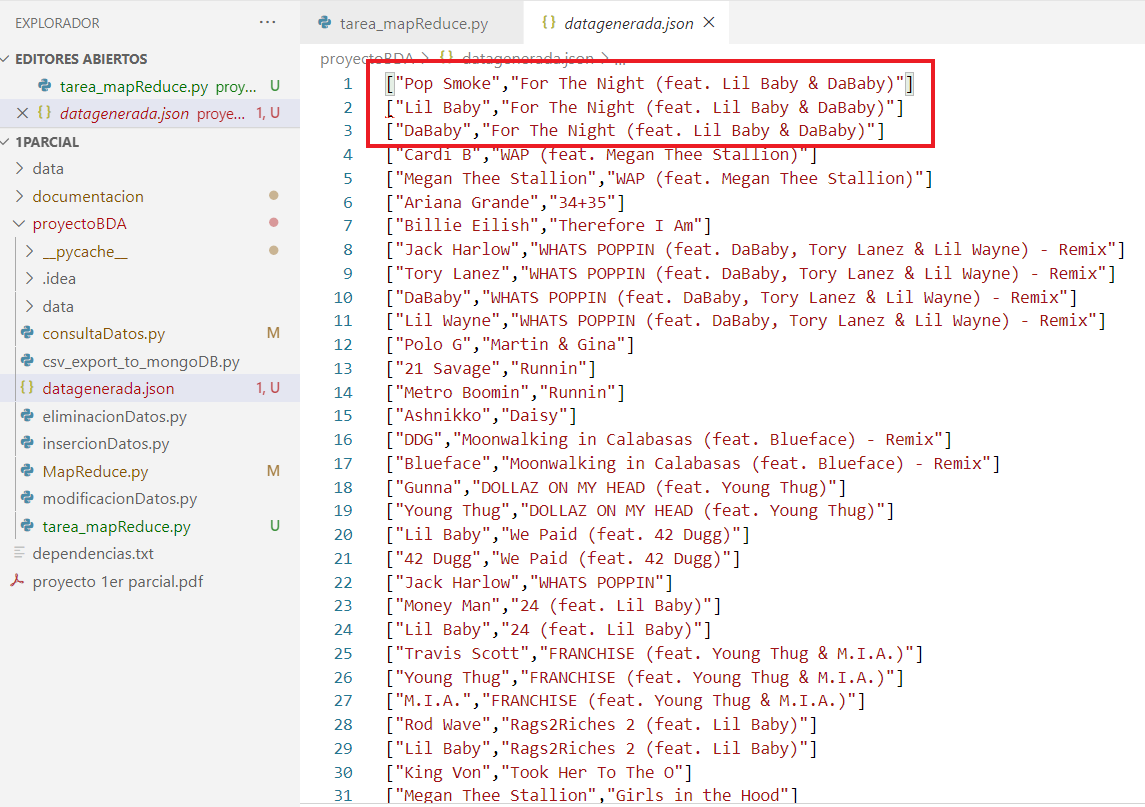




Se creó el documento Spotify, el cual representa cada fila del archivo data.csv

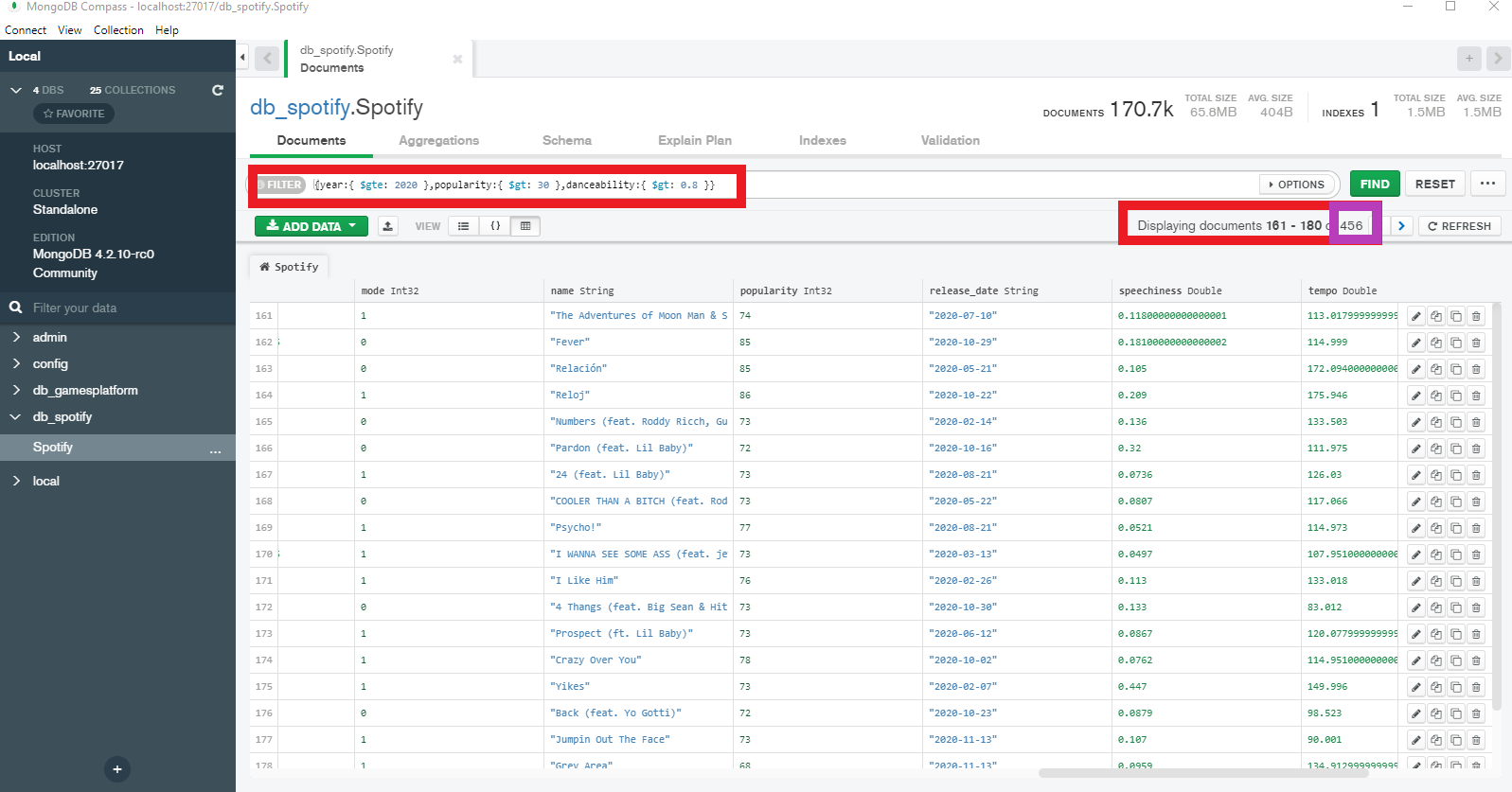


El archivo **datagenerada.json** generado. Como se puede observar, en el archivo existen canciones repetidas, y eso se debe a que esa canción fue creada por muchos artistas, a través del código implementado se logró esta separación de la data para poder realizar el análisis respectivo.

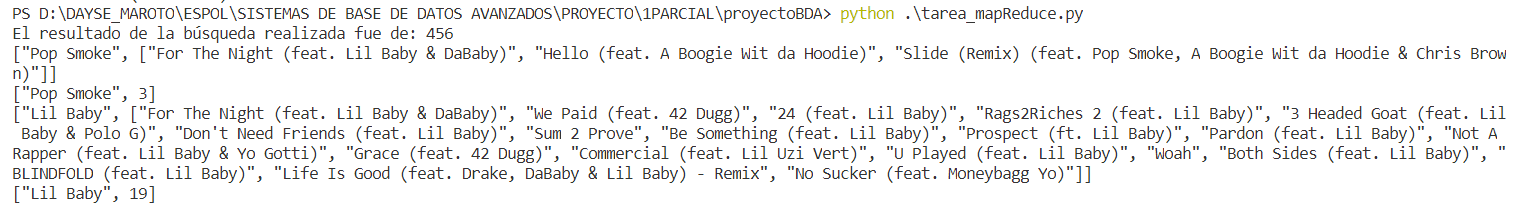


El total de la cantidad de canciones que cumple con las características mencionadas en la tarea concuerda con el resultado mostrado por MongoDB Compass.

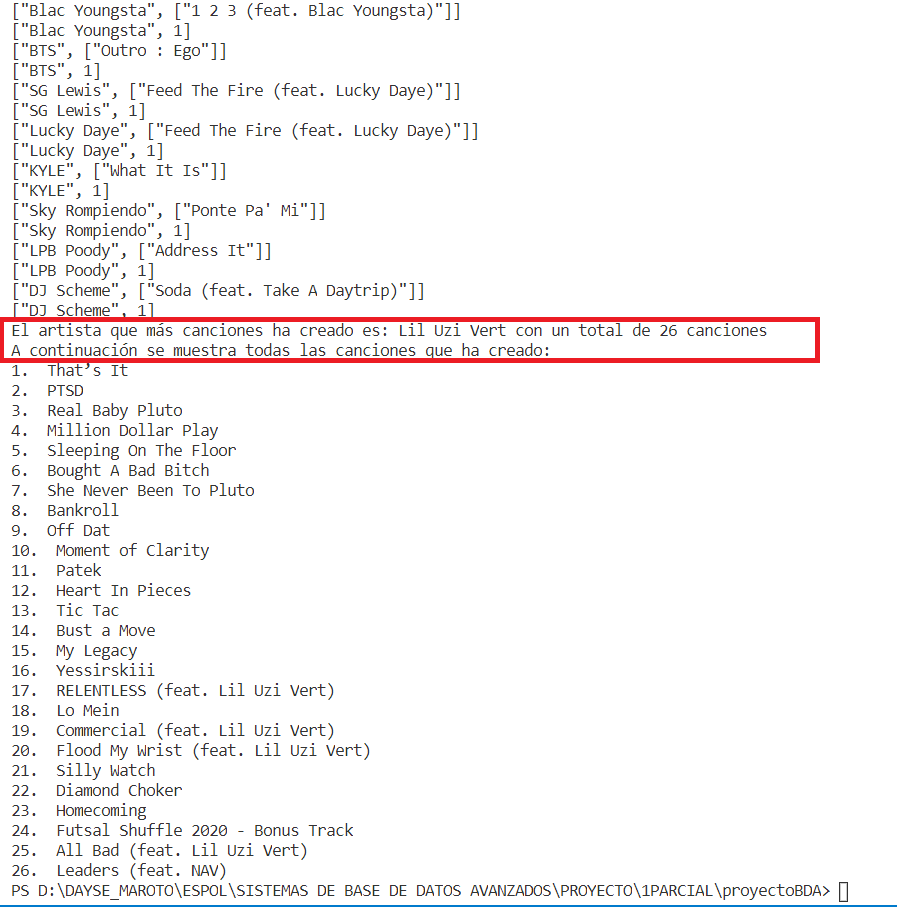




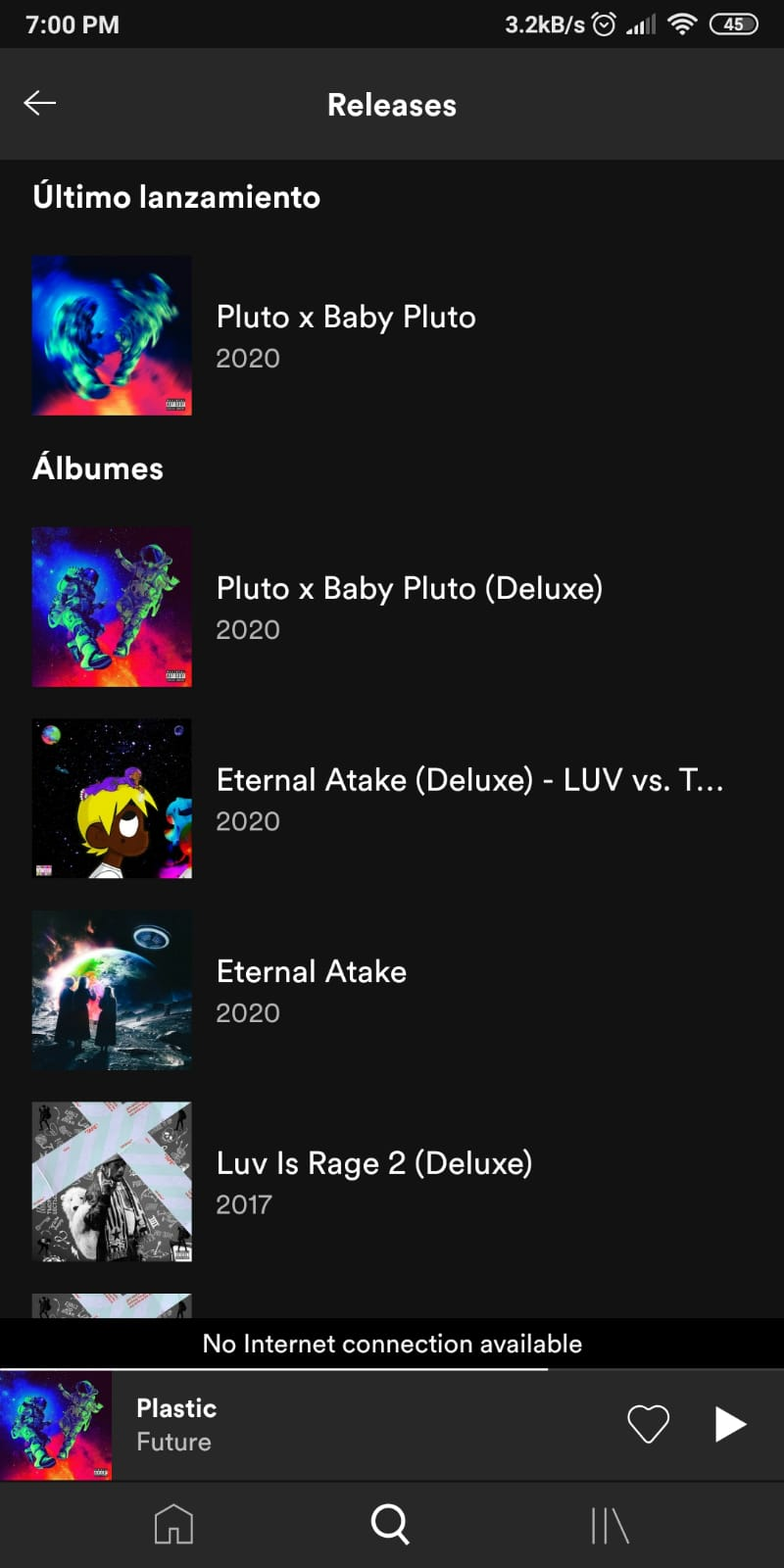
A continuación, se muestra el resultado del MapReduce, en donde se solicitó que presentara el artista con la lista de canciones que ha creado, así como la cantidad de canciones que ha creado.



Al final se muestra al artista que más canciones ha creado de acuerdo con las características anteriormente mencionadas para la tarea del MapReduce.



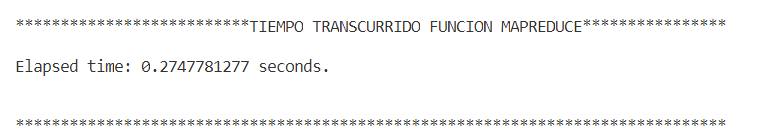
Se puede verificar la información obtenida en la plataforma de Spotify tal como se muestra a continuación:

Claramente se puede observar que este artista en el año 2020 ha lanzado 3 álbum y que cada álbum cuenta con una gran cantidad de canciones.

A continuación, se puede observar el tiempo de cómputo que se llevó al ejecutar la función de MapReduce.



# Bibliografía

[1] soumilshah1995, “Export your Csv File Data to MongoDb using Python,” *YouTube*. Dec. 31, 2019, Accessed: Dec. 07, 2020. [Online]. Available: <https://www.youtube.com/watch?v=CkckKPNbics&list=LL&index=1>.

[2] Redes Plus, “MongoDB04Como realizar Consultas en Compass de forma gráfica,” *YouTube*. Apr. 27, 2020, Accessed: Dec. 07, 2020. [Online]. Available: <https://www.youtube.com/watch?v=fl_3Jt5MIqo>.

[3] Craftech, “Mongodb Crud operations using pymongo,” *YouTube*. Apr. 06, 2019, Accessed: Dec. 07, 2020. [Online]. Available: <https://www.youtube.com/watch?v=orWal92Y-dU>.

[4] “$gt — MongoDB Manual,” *Mongodb.com*, 2020. https://docs.mongodb.com/manual/reference/operator/query/gt/index.html (accessed Dec. 07, 2020).

‌