

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>

1) 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 booleano.

- 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.

2) Descripción del Código para operaciones CRUD

Para las operaciones CRUD se utilizó la librería de pymongo que nos provee 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"

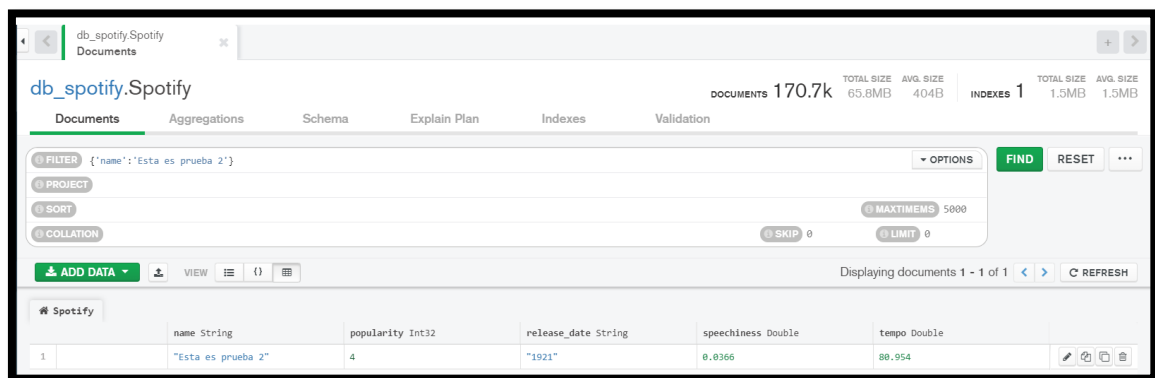
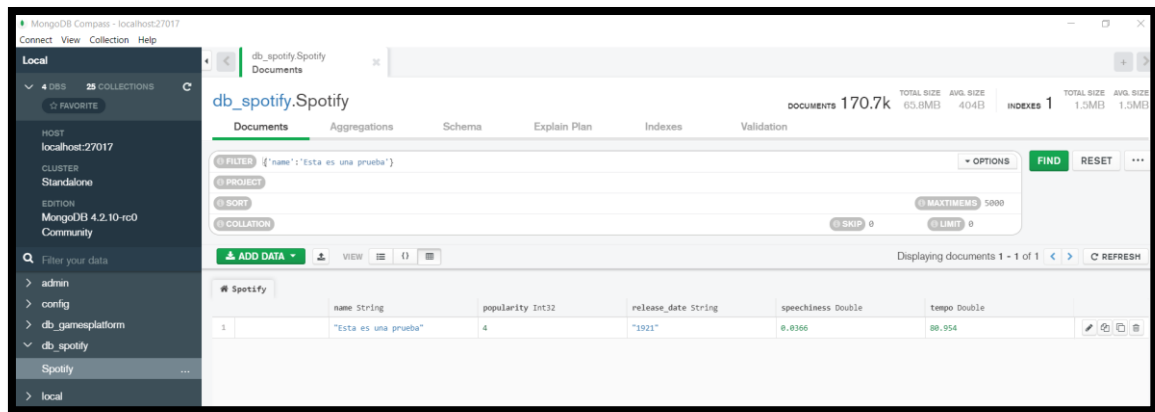
#nombre de la coleccion que vamos a manipular dentro de nuestra base de datos
collectionName= 'Spotify'

#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":"4BJqT
0PrAfrxzMOxytFOIz","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":"4BJqT
0PrAfrxzMOxytFOIz","instrumentalness":0.878,"key":10,"liveness":0.665,"loudness":-
20.096,"mode":1,"name":"Esta es prueba 2","popularity":4,"release_date":"1921","speechiness":0.0
366,"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 variable 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.

```
PS D:\DAYSE_MAROTO\ESPOL\SISTEMAS DE BASE DE DATOS AVANZADOS\PROYECTO\1PARCIAL\proyectoBDA> python .\consultaDatos.py
*****Primera consulta*****
{'_id': ObjectId('5fcec55bc2c092fe992a0b28'), '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': '4BjgT0PPrfrxzMoxytFOIz', '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}
```

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

```
*****Segunda consulta*****
{'_id': ObjectId('5fcd36057f4ad23077416eb'), 'valence': 0.991, 'year': 1997, 'acousticness': 0.0477, 'artists': '['Los Tucanes De Tijuana']', 'danceability': 0.8320000000000001, 'duration_ms': 174160, 'energy': 0.843, 'explicit': 0, 'id': '07Ag8vm1pW409NrhpPokFg', 'instrumentalness': 0.00883, 'key': 4, 'liveness': 0.0908, 'loudness': -7.595, 'mode': 1, 'name': 'El Tucanazo', 'popularity': 63, 'release_date': '1997-04-22', 'speechiness': 0.0449, 'tempo': 146.141}
{'_id': ObjectId('5fcd36057f4ad230774511c'), 'valence': 1.0, 'year': 1977, 'acousticness': 0.33899999999999997, 'artists': '['Raffi']', 'danceability': 0.878, 'duration_ms': 51547, 'energy': 0.255, 'explicit': 0, 'id': '0CgeqVC6ipphjyxgrs1mmZ', 'instrumentalness': 9.57e-06, 'key': 2, 'liveness': 0.0876, 'loudness': -14.14, 'mode': 1, 'name': 'Les Petites Marionnettes', 'popularity': 42, 'release_date': '1977', 'speechiness': 0.0753, 'tempo': 106.649}
{'_id': ObjectId('5fcd36057f4ad2307747ac5'), 'valence': 1.0, 'year': 1937, 'acousticness': 0.94, 'artists': '['Raymond Scott']', 'danceability': 0.738, 'duration_ms': 18855, 'energy': 0.314, 'explicit': 0, 'id': '53nhbx7yp4TJEpmMBCLWpQ', 'instrumentalness': 0.21100000000000002, 'key': 1, 'liveness': 0.564, 'loudness': -11.818, 'mode': 1, 'name': 'Chatter', 'popularity': 0, 'release_date': '1937', 'speechiness': 0.858, 'tempo': 119.255}
{'_id': ObjectId('5fcd36057f4ad2307747d1'), 'valence': 0.996, 'year': 1994, 'acousticness': 0.0731, 'artists': '['Barney']', 'danceability': 0.976, 'duration_ms': 56827, 'energy': 0.66, 'explicit': 0, 'id': '1JhF08L6Q2xQchBa5qJrxS', 'instrumentalness': 0.0, 'key': 7, 'liveness': 0.0486, 'loudness': -7.952000000000001, 'mode': 1, 'name': 'Barney Theme Song', 'popularity': 43, 'release_date': '1994-01-01', 'speechiness': 0.0351, 'tempo': 113.95700000000001}
{'_id': ObjectId('5fcd36057f4ad230774ac76'), 'valence': 0.993, 'year': 2000, 'acousticness': 0.0935, 'artists': '['Four Tops']', 'danceability': 0.705, 'duration_ms': 166533, 'energy': 0.8490000000000001, 'explicit': 0, 'id': '1h4rkw5vlg8JnvEueIBoXl', 'instrumentalness': 0.0, 'key': 2, 'liveness': 0.0384, 'loudness': -3.597, 'mode': 0, 'name': 'It's The Same Old Song - Single Version / Mono', 'popularity': 49, 'release_date': '2000-01-01', 'speechiness': 0.0314, 'tempo': 126.275}
{'_id': ObjectId('5fcd36057f4ad230774ae00'), 'valence': 1.0, 'year': 2002, 'acousticness': 0.0776, 'artists': '['Montez de Durango']', 'danceability': 0.912, 'duration_ms': 200213, 'energy': 0.8240000000000001, 'explicit': 0, 'id': '3rHsX3TLPyxr7eGzRLzqI', 'instrumentalness': 0.541, 'key': 9, 'liveness': 0.026000000000000002, 'loudness': -5.162999999999999, 'mode': 1, 'name': 'Pasito Duranguense', 'popularity': 50, 'release_date': '2002-01-04', 'speechiness': 0.0831, 'tempo': 144.925}
{'_id': ObjectId('5fcd36057f4ad230774f32b'), 'valence': 0.9940000000000001, 'year': 2002, 'acousticness': 0.0739, 'artists': '['Montez de Durango']', 'danceability': 0.8590000000000001, 'duration_ms': 176293, 'energy': 0.7170000000000001, 'explicit': 0, 'id': '1cmSizxgLDaGGFrWgZUEM', 'instrumentalness': 1.67e-05, 'key': 2, 'liveness': 0.0193, 'loudness': -6.367000000000001, 'mode': 1, 'name': 'El Sube Y Baja', 'popularity': 48, 'release_date': '2002-01-04', 'speechiness': 0.0882, 'tempo': 14
```

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
        }
    }
)

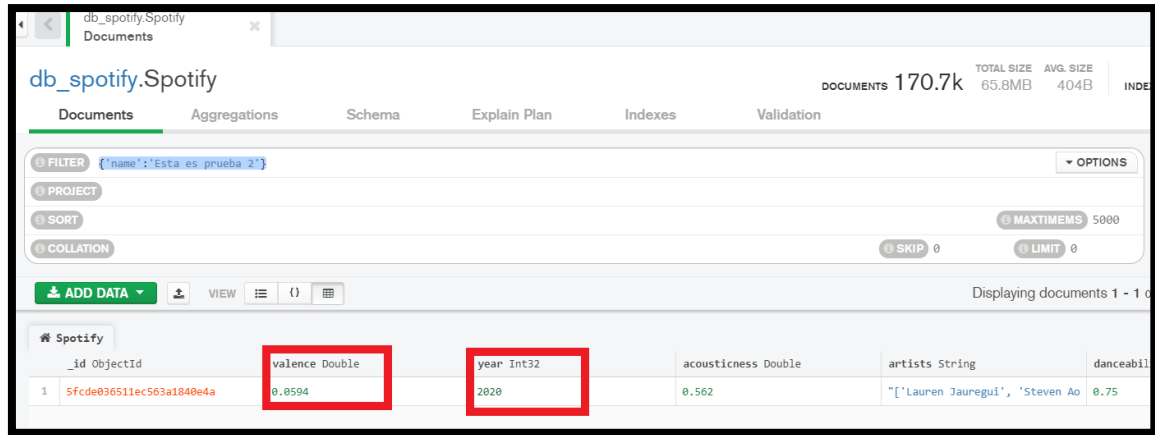
db[collectionName].update_one(
    {"name": "Esta es prueba 2"},
    {
        "$set":{
            "valence": 0.7,
            "year": 2021
        }
    }
)

print("Modificación culminada con éxito")
```

Resultado

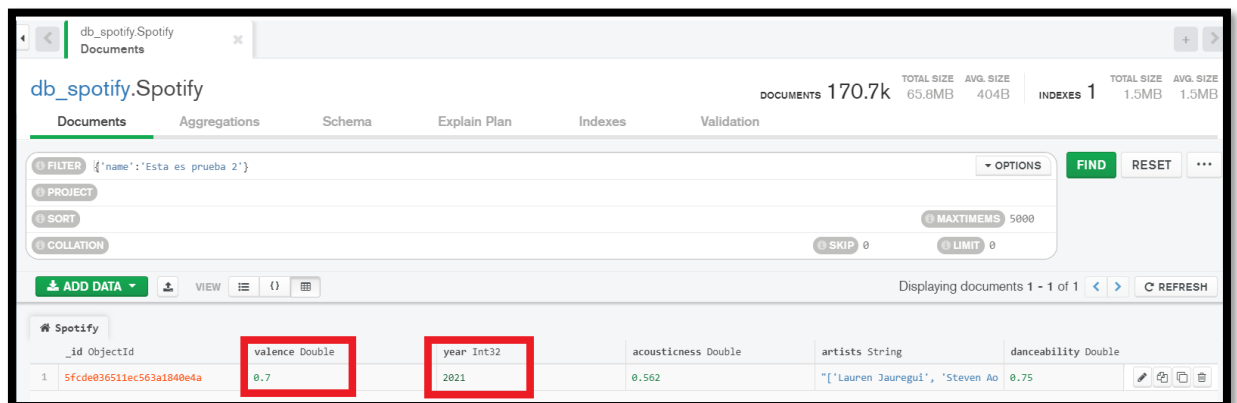
A continuación, se puede observar la evidencia de la modificación de datos en la base de datos a través de la herramienta de MongoDB Compass.

Antes



	_id ObjectId	valence Double	year Int32	acousticness Double	artists String	danceability Double
1	5fcd0e36511ec563a1840e4a	0.0594	2020	0.562	["Lauren Jauregui", "Steven Ao"]	0.75

Después



	_id ObjectId	valence Double	year Int32	acousticness Double	artists String	danceability Double
1	5fcd0e36511ec563a1840e4a	0.7	2021	0.562	["Lauren Jauregui", "Steven Ao"]	0.75

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")
```

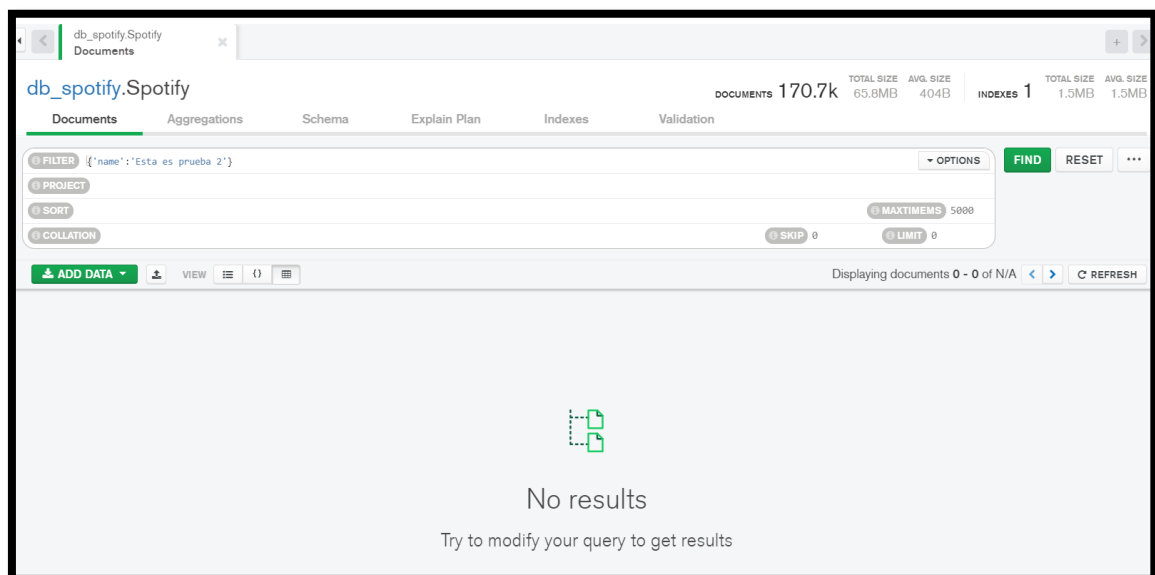
Resultado

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.



3) 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.

4) 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.

```
PS D:\DAYSE_MAROTO\ESPOL\SISTEMAS DE BASE DE DATOS AVANZADOS\PROYECTO\1PARCIAL\proyectoBDA> python .\tarea_mapReduce.py
File ".\tarea_mapReduce.py", line 96, in <module>
    mr.execute(dataR, mapper, reducer)
File "D:\DAYSE_MAROTO\ESPOL\SISTEMAS DE BASE DE DATOS AVANZADOS\PROYECTO\1PARCIAL\proyectoBDA\MapReduce.py", line 17, in execute
    record = json.loads(line)
File "D:\PROGRAMAS\Python 3.8\lib\json\__init__.py", line 341, in loads
    raise TypeError(f'the JSON object must be str, bytes or bytearray, '
TypeError: the JSON object must be str, bytes or bytearray, not dict
```

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 d
atos
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 ac
uerdo 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 c
    reado
    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 h
    a creado
    mr.emit((key,len(value2)))
if __name__ == '__main__':
    #cargamos nuestra colleccion Spotify de la base de datos de MongoDB a
    l 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 m
    apReduce
    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 obt
ener 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 MAPR
EDUCE*****\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 cancion
es
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]+" c
on un total de "+str(maximo)+" canciones")
print("A continuación se muestra todas las canciones que ha creado: ")
contador=1

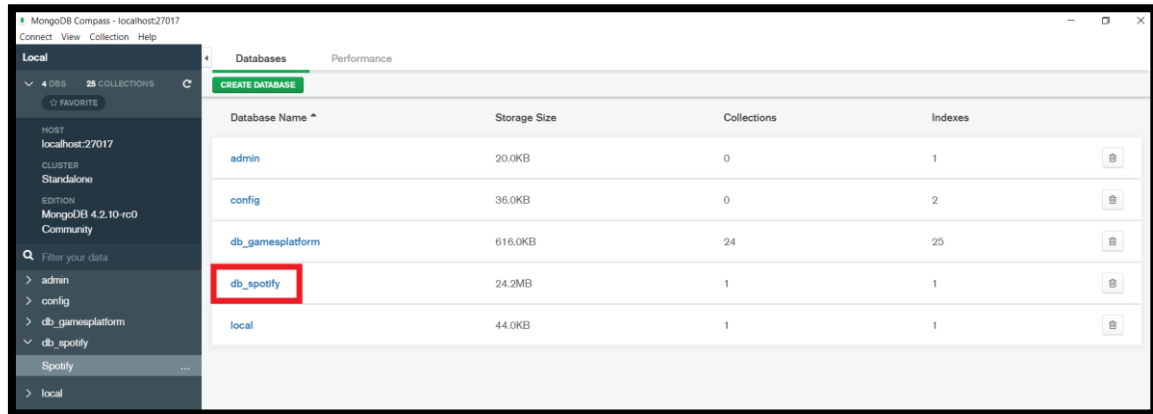
# Se muestran las canciones del artista ganador
for cancion in canciones_artista_ganador:
    print(str(contador) +".  "+ cancion)
    contador= contador+1

```

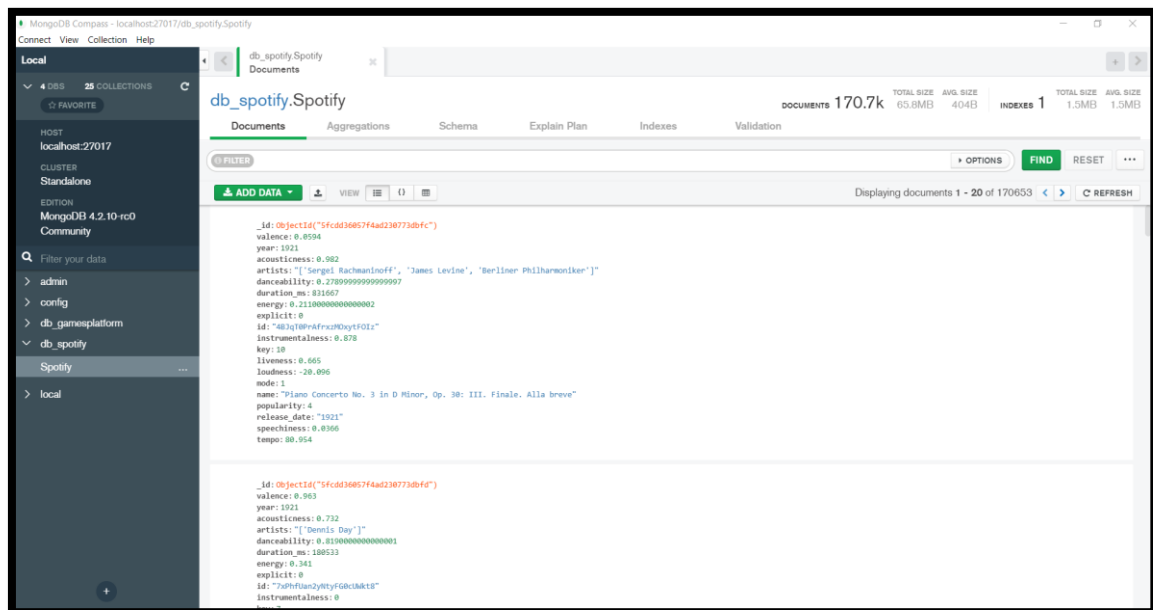
5) Descripción de los resultados obtenidos.

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

```
PS D:\DAYSE_MAROTO\ESPOL\SISTEMAS DE BASE DE DATOS AVANZADOS\PROYECTO\1PARCIAL\proyectoBDA> python .\csv_export_to_mongoDB.py Toda
1a data ha sido exportada a Mongo DB Server
PS D:\DAYSE_MAROTO\ESPOL\SISTEMAS DE BASE DE DATOS AVANZADOS\PROYECTO\1PARCIAL\proyectoBDA> |
```



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.

```
EXPLORADOR
...
tarea_mapReduce.py
datagenerada.json

EDITORES ABIERTOS
tarea_mapReduce.py proy... U
datagenerada.json proye... 1, U

1PARCIAL
> data
> documentacion
> proyectoBDA
> __pycache__
> .idea
> data
+ consultaDatos.py M
+ csv_export_to_mongoDB.py
+ datagenerada.json 1, U
+ eliminacionDatos.py
+ insercionDatos.py
+ MapReduce.py M
+ modificacionDatos.py
+ tarea_mapReduce.py U
+ dependencias.txt
+ proyecto 1er parcial.pdf

proyectoBDA > datagenerada.json
1 ["Pop Smoke", "For The Night (feat. Lil Baby & DaBaby)"]
2 ["Lil Baby", "For The Night (feat. Lil Baby & DaBaby)"]
3 ["DaBaby", "For The Night (feat. Lil Baby & DaBaby)"]
4 ["Cardi B", "WAP (feat. Megan Thee Stallion)"]
5 ["Megan Thee Stallion", "WAP (feat. Megan Thee Stallion)"]
6 ["Ariana Grande", "34+35"]
7 ["Billie Eilish", "Therefore I Am"]
8 ["Jack Harlow", "WHATS POPPIN (feat. DaBaby, Tory Lanez & Lil Wayne) - Remix"]
9 ["Tory Lanez", "WHATS POPPIN (feat. DaBaby, Tory Lanez & Lil Wayne) - Remix"]
10 ["DaBaby", "WHATS POPPIN (feat. DaBaby, Tory Lanez & Lil Wayne) - Remix"]
11 ["Lil Wayne", "WHATS POPPIN (feat. DaBaby, Tory Lanez & Lil Wayne) - Remix"]
12 ["Polo G", "Martin & Gina"]
13 ["21 Savage", "Runnin"]
14 ["Metro Boomin", "Runnin"]
15 ["Ashnikko", "Daisy"]
16 ["DDG", "Moonwalking in Calabasas (feat. Blueface) - Remix"]
17 ["Blueface", "Moonwalking in Calabasas (feat. Blueface) - Remix"]
18 ["Gunna", "DOLLAZ ON MY HEAD (feat. Young Thug)"]
19 ["Young Thug", "DOLLAZ ON MY HEAD (feat. Young Thug)"]
20 ["Lil Baby", "We Paid (feat. 42 Dugg)"]
21 ["42 Dugg", "We Paid (feat. 42 Dugg)"]
22 ["Jack Harlow", "WHATS POPPIN"]
23 ["Money Man", "24 (feat. Lil Baby)"]
24 ["Lil Baby", "24 (feat. Lil Baby)"]
25 ["Travis Scott", "FRANCHISE (feat. Young Thug & M.I.A.)"]
26 ["Young Thug", "FRANCHISE (feat. Young Thug & M.I.A.)"]
27 ["M.I.A.", "FRANCHISE (feat. Young Thug & M.I.A.)"]
28 ["Rod Wave", "Rags2Riches 2 (feat. Lil Baby)"]
29 ["Lil Baby", "Rags2Riches 2 (feat. Lil Baby)"]
30 ["King Von", "Took Her To The O"]
31 ["Megan Thee Stallion", "Girls in the Hood"]
```

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

```
PS D:\DAYSE_MAROTO\ESPOL\SISTEMAS DE BASE DE DATOS_AVANZADOS\PROYECTO\1PARCIAL\proyectoBDA> python .\tarea_mapReduce.py
El resultado de la búsqueda realizada fue de 456
```

	mode Int32	name String	popularity Int32	release_date String	speechiness Double	tempo Double
161	1	"The Adventures of Moon Man & S	74	"2020-07-10"	0.11800000000000001	113.01799999999999
162	0	"Fever"	85	"2020-10-29"	0.18100000000000002	114.999
163	0	"Relación"	85	"2020-05-21"	0.105	172.09400000000000
164	1	"Reloj"	86	"2020-10-22"	0.209	175.946
165	0	"Numbers (feat. Roddy Ricch, Gu	73	"2020-02-14"	0.136	133.503
166	0	"Pardon (feat. Lil Baby)"	72	"2020-10-16"	0.32	111.975
167	1	"24 (feat. Lil Baby)"	73	"2020-08-21"	0.0736	126.03
168	0	"COOLER THAN A BITCH (feat. Rod	73	"2020-05-22"	0.0807	117.066
169	1	"Psychol"	77	"2020-08-21"	0.0521	114.973
170	1	"I WANNA SEE SOME ASS (feat. je	73	"2020-03-13"	0.0497	107.95100000000000
171	1	"I Like Him"	76	"2020-02-26"	0.113	133.018
172	0	"4 Thangs (feat. Big Sean & Hit	73	"2020-10-30"	0.133	83.012
173	1	"Prospect (ft. Lil Baby)"	73	"2020-06-12"	0.0867	120.07799999999999
174	1	"Crazy Over You"	78	"2020-10-02"	0.0762	114.95100000000000
175	1	"Yikes"	73	"2020-02-07"	0.447	149.996
176	0	"Back (feat. Yo Gotti)"	72	"2020-10-23"	0.0879	98.523
177	1	"Jumpin Out The Face"	73	"2020-11-13"	0.107	90.001
178	1	"Grev Area"	68	"2020-11-13"	0.0959	134.91799999999999

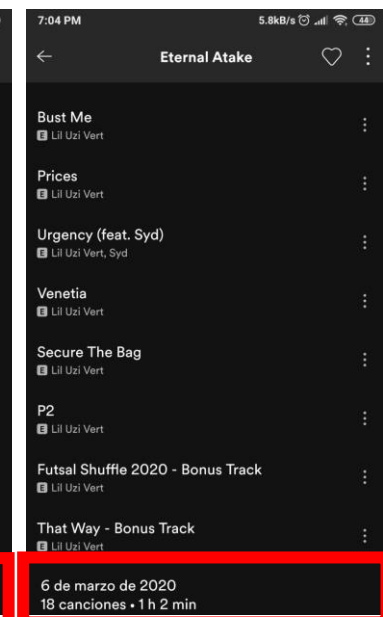
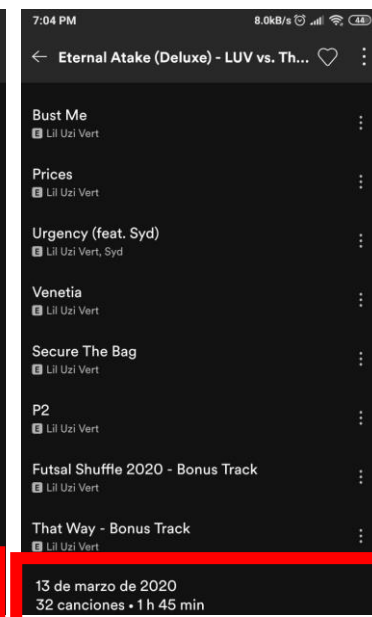
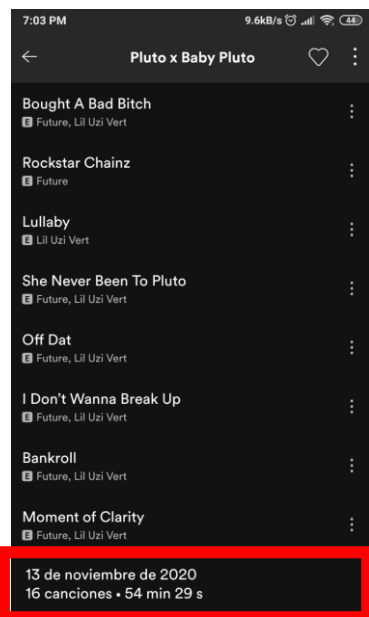
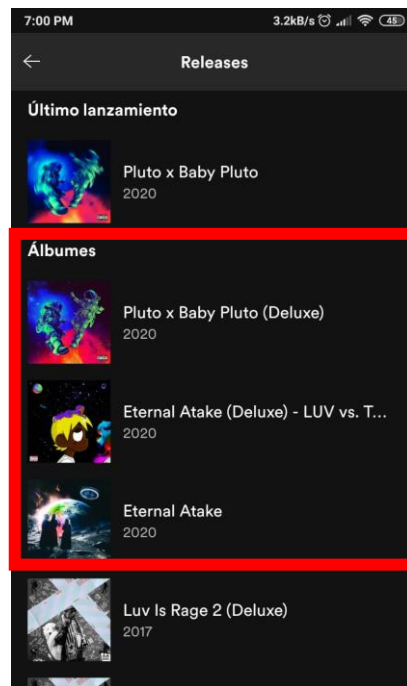
A continuación, se muestra el resultado del MapReduce, en donde se solicitó que presentara el artista con la lista de canciones que ha creado, así como la cantidad de canciones que ha creado.

```
PS D:\DAYSE_MAROTO\ESPOL\SISTEMAS DE BASE DE DATOS AVANZADOS\PROYECTO\1PARCIAL\proyectoBDA> python .\tarea_mapReduce.py
El resultado de la búsqueda realizada fue de: 456
["Pop Smoke", ["For The Night (feat. Lil Baby & DaBaby)", "Hello (feat. A Boogie Wit da Hoodie)", "Slide (Remix) (feat. Pop Smoke, A Boogie Wit da Hoodie & Chris Brown)"]]
["Pop Smoke", 3]
["Lil Baby", ["For The Night (feat. Lil Baby & DaBaby)", "We Paid (feat. 42 Dugg)", "24 (feat. Lil Baby)", "Rags2Riches 2 (feat. Lil Baby)", "3 Headed Goat (feat. Lil Baby & Polo G)", "Don't Need Friends (feat. Lil Baby)", "Sum 2 Prove", "Be Something (feat. Lil Baby)", "Prospect (ft. Lil Baby)", "Pardon (feat. Lil Baby)", "Not A Rapper (feat. Lil Baby & Yo Gotti)", "Grace (feat. 42 Dugg)", "Commercial (feat. Lil Uzi Vert)", "U Played (feat. Lil Baby)", "Woah", "Both Sides (feat. Lil Baby)", "BLINDFOLD (feat. Lil Baby)", "Life Is Good (feat. Drake, DaBaby & Lil Baby) - Remix", "No Sucker (feat. Moneybagg Yo)"]]
["Lil Baby", 19]
```

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

```
["Blac Youngsta", ["1 2 3 (feat. Blac Youngsta)"]]
["Blac Youngsta", 1]
["BTS", ["Outro : Ego"]]
["BTS", 1]
["SG Lewis", ["Feed The Fire (feat. Lucky Daye)"]]
["SG Lewis", 1]
["Lucky Daye", ["Feed The Fire (feat. Lucky Daye)"]]
["Lucky Daye", 1]
["KYLE", ["What It Is"]]
["KYLE", 1]
["Sky Rompiendo", ["Ponte Pa' Mi"]]
["Sky Rompiendo", 1]
["LPB Poody", ["Address It"]]
["LPB Poody", 1]
["DJ Scheme", ["Soda (feat. Take A Daytrip)"]]
["DJ Scheme", 1]
El artista que más canciones ha creado es: Lil Uzi Vert con un total de 26 canciones
A continuación se muestra todas las canciones que ha creado:
1. That's It
2. PTSD
3. Real Baby Pluto
4. Million Dollar Play
5. Sleeping On The Floor
6. Bought A Bad Bitch
7. She Never Been To Pluto
8. Bankroll
9. Off Dat
10. Moment of Clarity
11. Patek
12. Heart In Pieces
13. Tic Tac
14. Bust a Move
15. My Legacy
16. Yessirskiii
17. RELENTLESS (feat. Lil Uzi Vert)
18. Lo Mein
19. Commercial (feat. Lil Uzi Vert)
20. Flood My Wrist (feat. Lil Uzi Vert)
21. Silly Watch
22. Diamond Choker
23. Homecoming
24. Futsal Shuffle 2020 - Bonus Track
25. All Bad (feat. Lil Uzi Vert)
26. Leaders (feat. NAV)
PS D:\DAYSE_MAROTO\ESPOL\SISTEMAS DE BASE DE DATOS AVANZADOS\PROYECTO\1PARCIAL\proyectoBDA> []
```

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.

```
*****TIEMPO TRANSCURRIDO FUNCION MAPREDUCE*****

Elapsed time: 0.2747781277 seconds.

*****
```

6) 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).