4.3 Avance de proyecto 1: Sistema de Recomendación

Análisis de grandes volúmenes de datos

TC4034 grupo 10 Equipo 26

Luis Arturo Dan Fong | A01650672

Eduardo Rodríguez Ramírez| A01794892

Felipe Enrique Vázquez Ruiz | A01638116

17 de Mayo de 2024

1. Justificación y Preprocesamiento de Datos

- **Justificación del Conjunto de Datos:**

  1. Las listas de reproducción de Spotify reflejan las preferencias musicales reales de los usuarios, proporcionando una visión valiosa de sus gustos y hábitos de escucha.
  2. Este conjunto de datos es rico en metadatos (artistas, géneros, popularidad, etc.), lo que permite un análisis profundo y diversas estrategias de recomendación.
  3. Spotify es una plataforma líder en streaming, por lo que las recomendaciones basadas en sus datos tienen alta relevancia para un público amplio.

- **Pasos de Preprocesamiento:**

  1. **Limpieza:** Eliminar duplicados, valores faltantes y datos inconsistentes.
  2. **Normalización:** Estandarizar nombres de artistas y géneros, ajustar valores numéricos (por ejemplo, popularidad) a una escala común.
  3. **Transformación:** Indexar y separar las columnas en conjuntos independientes, relacionándolos con la matriz de reproducciones por medio de los índices, lo que facilitará las operaciones sobre la matriz.
  4. **Filtrado:** Seleccionar un subconjunto de datos relevante (por ejemplo, listas de reproducción de un género específico, período de tiempo) si es necesario.

Instalando Dependencias

[ ] ↳ *4 celdas ocultas*

Preprocesamiento

```python
import opendatasets as od
import pandas as pd
import pyspark
from pyspark.sql import SparkSession
import pyspark.sql.functions as fn
import os
from pyspark.conf import SparkConf
from pyspark.context import SparkContext
from pyspark.sql.window import Window
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator
```

```python
od.download(
  "https://www.kaggle.com/datasets/andrewmvd/spotify-playlists")
```

```
Please provide your Kaggle credentials to download this dataset. Learn more: http://bit.ly/kaggle-creds
Your Kaggle username: a01794892tecmx
Your Kaggle Key: ··········
Dataset URL: https://www.kaggle.com/datasets/andrewmvd/spotify-playlists
Downloading spotify-playlists.zip to ./spotify-playlists
100%|██████████| 183M/183M [00:01<00:00, 109MB/s]
```

```python
# Set the Spark master URL and other Spark settings
#os.environ['PYSPARK_SUBMIT_ARGS'] = '--master local[*] --executor-memory 4G --num-executors 4 pyspark-shell'
conf = SparkConf(loadDefaults=True)
conf.setMaster("local").setAppName("sptifyApp")
sc = SparkContext(conf=conf)
```

```python
spark = SparkSession.builder.getOrCreate()
```

```python
spark.conf.set("spark.sql.pivotMaxValues", 2200000)
```

```python
spark
```

SparkSession - in-memory

SparkContext

Spark UI

```
Version
      v3.5.1
Master
      local
AppName
      sptifyApp
```

Limpieza de datos

```python
df = spark.read.option("header", "true").csv("spotify-playlists//spotify_dataset.csv")

df = df.dropna()

df = df.drop_duplicates()

df = df.toDF(*[col.replace(' ', '').replace('"', '') for col in df.columns])


df.head()

```

```
Row(user_id='9cc0cfd4d7d7885102480dd99e7a90d6', artistname='Elvis Costello', trackname='(The Angels Wanna Wear My) Red Shoes', playlistname='HARD ROCK 2010')
```

Creando dimensiones separadas para playlists, artistas, canciones y usuarios, reconstruyendo la matriz de reproducciones con los indices de dichas dimensiones.

```python
dims={}

def df_dim(df, input_col):
    windowSpec = Window.orderBy(input_col)
    dims[input_col]=df.select(input_col).distinct().withColumn(f"{input_col}_index", fn.row_number().over(windowSpec))

for col_name in df.columns:
    df_dim(df,col_name)

for dim in dims.values():
    print(dim.head(10))

newdf=df

for i in range(0, len(df.columns)):
    col_name = df.columns[i]
    newdf=newdf.join(dims[col_name].withColumnRenamed(col_name, col_name+'_base'), fn.col(col_name)==fn.col(col_name+'_base')).drop(col_name).drop(col_name+'_base')

print(newdf.head(10))
```

```
[Row(user_id='00055176fea33f6e027cd3302289378b', user_id_index=1), Row(user_id='0007f3dd09c91198371454c608d47f22', user_id_index=2), Row(user_id='000b0f32b5739f052b9d40fcc5c41079', user_id_index=3), Row(user_id='000c11a16c89aa
[Row(artistname=' Dolce', artistname_index=1), Row(artistname=' OneVoice', artistname_index=2), Row(artistname='!!!', artistname_index=3), Row(artistname='!!! (Chk Chk Chk)', artistname_index=4), Row(artistname='!!! Chk Chik C
[Row(trackname=' "Cachaito" López Y "Guajiro" Mirabal De Buena Vista Social Club Y Manuel "Galbán" Torralba"', trackname_index=1), Row(trackname=' 15 Years of Tummy Touch Records in Dub', trackname_index=2), Row(trackname=' Al
[Row(playlistname=' ', playlistname_index=1), Row(playlistname='        waves', playlistname_index=2), Row(playlistname='  11', playlistname_index=3), Row(playlistname='  Frida', playlistname_index=4), Row(playlistname='  New
[Row(user_id_index=6511, artistname_index=15, trackname_index=1, playlistname_index=5707), Row(user_id_index=4852, artistname_index=49030, trackname_index=2, playlistname_index=70282), Row(user_id_index=4213, artistname_index=
```

Matriz de reporducciones según el artista y normalización de los datos.

```python
counts_df = newdf.groupBy("user_id_index", "artistname_index").agg(fn.count("*").alias("reproductions"))
```

```python
max_reproduction = counts_df.agg({"reproductions": "max"}).collect()[0][0]
min_reproduction = counts_df.agg({"reproductions": "min"}).collect()[0][0]

normalized_pl_counts_df = counts_df.withColumn("normalized_reproduction", (fn.col("reproductions") - min_reproduction) / (max_reproduction - min_reproduction))
normalized_pl_counts_df.head(10)
```

```
[Row(user_id_index=11738, artistname_index=212990, reproductions=24, normalized_reproduction=0.0068759342301943195),
 Row(user_id_index=1036, artistname_index=152785, reproductions=4, normalized_reproduction=0.0008968609865470852),
 Row(user_id_index=2695, artistname_index=118440, reproductions=20, normalized_reproduction=0.005680119581464873),
 Row(user_id_index=7942, artistname_index=45951, reproductions=236, normalized_reproduction=0.07025411061285501),
 Row(user_id_index=4281, artistname_index=258892, reproductions=3, normalized_reproduction=0.0005979073243647235),
 Row(user_id_index=6765, artistname_index=81653, reproductions=38, normalized_reproduction=0.011061285500747383),
 Row(user_id_index=5278, artistname_index=125652, reproductions=42, normalized_reproduction=0.012257100149476832),
 Row(user_id_index=15667, artistname_index=198257, reproductions=44, normalized_reproduction=0.012855007473841554),
 Row(user_id_index=5381, artistname_index=91263, reproductions=61, normalized_reproduction=0.017937219730941704),
 Row(user_id_index=13572, artistname_index=280007, reproductions=52, normalized_reproduction=0.015246636771300448)]
```

Sistema de recomendación

[ ] ↳ *6 celdas ocultas*

2. Exploración Inicial y Análisis

- **Análisis de Distribución:**
    - Visualizar la distribución de géneros, artistas y canciones más populares.
    - Identificar patrones de escucha según la hora del día, día de la semana, etc.

- **Top 10:**
    - Analizar los usuarios, artistas y playlists con más repeticiones
    - Visualizar los resultados gráficamente

ANALISIS EXPLORATORIO

```
pip install opendatasets
```

```
Requirement already satisfied: opendatasets in /usr/local/lib/python3.10/dist-packages (0.1.22)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from opendatasets) (4.66.4)
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (from opendatasets) (1.6.12)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from opendatasets) (8.1.7)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2024.2.2)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.31.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.0.7)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle->opendatasets) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (3.7)
```

```
import pandas as pd
import matplotlib.pyplot as plt
import opendatasets as od
```

```
od.download(
    "https://www.kaggle.com/datasets/andrewmvd/spotify-playlists")
```

```
Skipping, found downloaded files in "./spotify-playlists" (use force=True to force download)
```

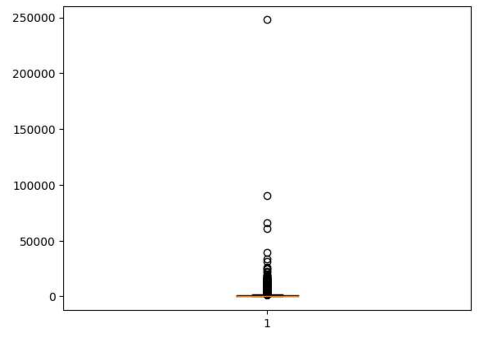
```
users = {}
artists = {}
playlists = {}
for chunk in pd.read_csv("spotify-playlists//spotify_dataset.csv", chunksize=10000,on_bad_lines='skip'):
    chunk.columns = ['user', 'artist', 'song', 'playlist']
    chunk.dropna(inplace=True)
    chunk.drop_duplicates(inplace=True)
    chunk.drop(chunk[chunk['playlist']=='Starred'].index,inplace=True)
    chunk.drop(chunk[chunk['playlist']=='Liked from Radio'].index,inplace=True)
    chunk.drop(chunk[chunk['playlist']=='Favoritas de la radio'].index,inplace=True)
    for user in chunk['user']:
        if user in users:
            users[user] += 1
        else:
            users[user] = 1
    for artist in chunk['artist']:
        if artist in artists:
            artists[artist] += 1
        else:
            artists[artist] = 1
    for playlist in chunk['playlist']:
        if playlist in playlists:
            playlists[playlist] += 1
        else:
            playlists[playlist] = 1
```

```
df = pd.DataFrame.from_dict(users,orient='index')
df.columns = ['Reproducciones']
df.reset_index(inplace=True)
df.rename(columns={"index": "Usuario"},inplace=True)
df.sort_values('Reproducciones',inplace=True,ascending=False)
print('El top 10 usuarios con más reproducciones son:' )
df.head(10)
```

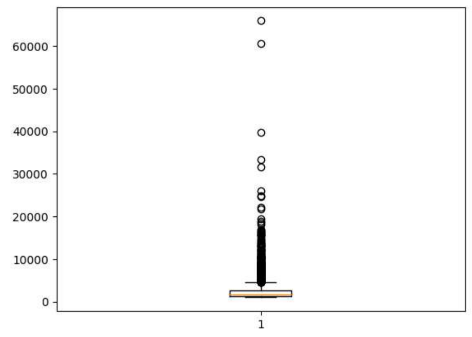
```
El top 10 usuarios con más reproducciones son:
```

	Usuario	Reproducciones
7651	4398de6902abde3351347b048fcdc287	247927
3462	7ee2b92c5bcf6133b8132363e5bda960	90628
1766	99deafd9b792af8e6a535483088faef2	65909
5646	fa849dabeb14a2800ad5130907fc5018	60614
8850	6b85a8076b4f3b1036ef15f09e2feeeb	39638
11547	ed140fce438f59e6e07e5ee7bd726692	33258
14521	2fa1f93e57cfe2f6c4456e98da54061c	31581
12226	c0efe4e704a37894150489b25eac9042	25951
7821	c2d2fed26e858f82fdd8ac2e791cab1f	24830
14772	d49c0fdadc701f66d10aec118604f4b7	24630

```
plt.boxplot(df['Reproducciones'])
plt.show()
```

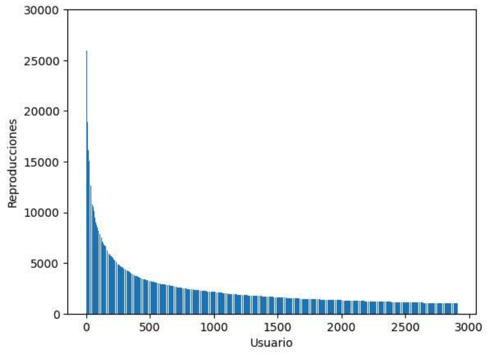


```
filtro = (df['Reproducciones'] < 1000) | (df['Reproducciones'] > 70000)
df.drop(df[filtro].index,inplace=True)
df.reset_index(inplace=True)
plt.boxplot(df['Reproducciones'])
plt.show()
```



```
plt.bar(df.index,df['Reproducciones'])
plt.ylim(0,30000)
plt.ylabel('Reproducciones')
plt.xlabel('Usuario')

```

```
Text(0.5, 0, 'Usuario')
```

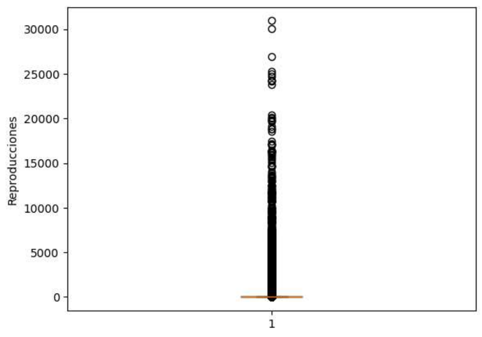


```
df = pd.DataFrame.from_dict(artists,orient='index')
df.columns = ['Reproducciones']
df.reset_index(inplace=True)
df.rename(columns={"index": "Artist"},inplace=True)
df.sort_values('Reproducciones',inplace=True,ascending=False)
print('El top 10 artistas con más reproducciones son:' )
df.head(10)
```

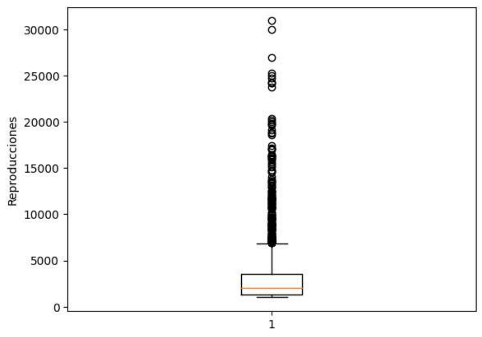
```
El top 10 artistas con más reproducciones son:
```

	Artist	Reproducciones
103	Daft Punk	30926
393	Coldplay	30030
198	The Rolling Stones	26966
248	Radiohead	25306
912	Eminem	24991
1074	Kanye West	24690
1356	David Bowie	24253
969	JAY Z	24220
228	Queen	24181
144	Michael Jackson	23746

```
plt.boxplot(df['Reproducciones'])
plt.ylabel('Reproducciones')
plt.show()
```

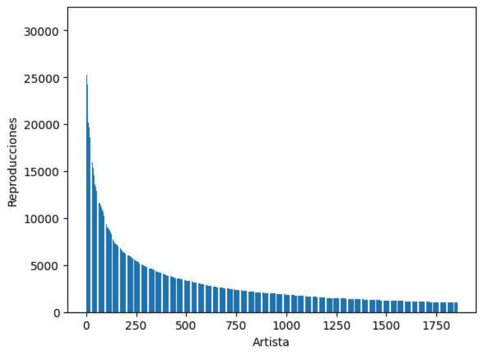


```
filtro = (df['Reproducciones'] < 1000) | (df['Reproducciones'] > 100000)
df.drop(df[filtro].index,inplace=True)
df.reset_index(inplace=True)
plt.boxplot(df['Reproducciones'])
plt.ylabel('Reproducciones')
plt.show()
```



```
plt.bar(df.index,df['Reproducciones'])
plt.ylabel('Reproducciones')
plt.xlabel('Artista')
```

```
Text(0.5, 0, 'Artista')
```

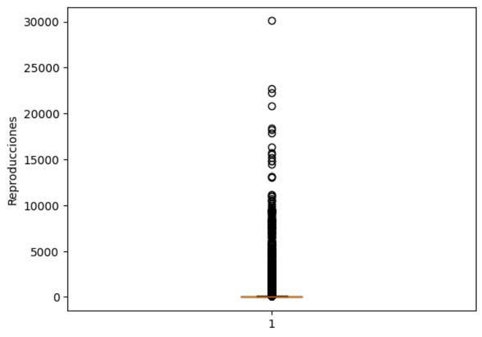


```
df = pd.DataFrame.from_dict(playlists,orient='index')
df.columns = ['Reproducciones']
df.reset_index(inplace=True)
df.rename(columns={"index": "Playlist"},inplace=True)
df.sort_values('Reproducciones',inplace=True,ascending=False)
print('El top 10 playlist con más reproducciones son:' )
df.head(10)
```

```
El top 10 playlist con más reproducciones son:
```

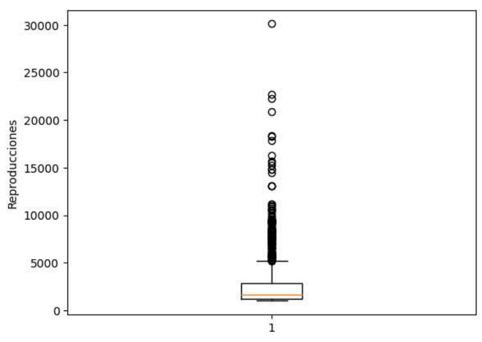
	Playlist	Reproducciones
2811	Rock	30107
859	2014	22674
1382	Christmas	22236
1404	2013	20870
1997	Work	18408
207	Jazz	18266
1464	Indie	17858
465	Classical	16328
63474	everything	15705
2677	Country	15503

```
plt.boxplot(df['Reproducciones'])
plt.ylabel('Reproducciones')
plt.show()
```



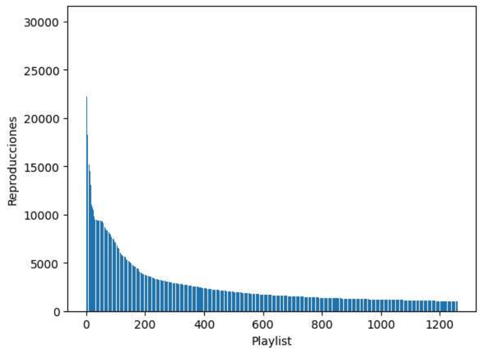
```
df.drop(df[df['Reproducciones']<1000].index,inplace=True)
df.reset_index(inplace=True)

```

```
plt.boxplot(df['Reproducciones'])
plt.ylabel('Reproducciones')
plt.show()
```



```
plt.bar(df.index,df['Reproducciones'])
plt.ylabel('Reproducciones')
plt.xlabel('Playlist')
```

```
Text(0.5, 0, 'Playlist')
```



3. Implementación de Algoritmos de Recomendación

- **Algoritmo de Filtrado Colaborativo Basado en Usuarios:**
    - Calcular la similitud entre usuarios en función de sus listas de reproducción.
    - Recomendar artistas que usuarios similares hayan escuchado y que el usuario objetivo aún no haya escuchado.

- **Algoritmo de Filtrado Colaborativo Basado en Ítems (Futura etapa del proyecto):**
    - Calcular la similitud entre listas de reproducción.
    - Recomendar playlists o canciones similares a las que el usuario ya ha escuchado.

Instalando Dependencias

[ ] ↳ *4 celdas ocultas*

Preprocesamiento

[ ] ↳ *13 celdas ocultas*

Sistema de recomendación de artistas

```python
(training, test) = normalized_pl_counts_df.randomSplit([0.8, 0.2])

als = ALS(maxIter=5, regParam=0.01, userCol="user_id_index", itemCol="artistname_index", ratingCol="normalized_reproduction", coldStartStra
model = als.fit(training)

predictions = model.transform(test)
evaluator = RegressionEvaluator(metricName="rmse", labelCol="normalized_reproduction", predictionCol="prediction")
rmse = evaluator.evaluate(predictions)

print("Root Mean Squared Error (RMSE) = " + str(rmse))

#Top 10 recomendaciones por usuario
userRecs = model.recommendForAllUsers(10)
userRecs.head(10)
```

```
Root Mean Squared Error (RMSE) = 0.12709003382739936
[Row(user_id_index=1, recommendations=[Row(artistname_index=163412, rating=0.13188980519771576), Row(artistname_index=76809,
rating=0.10977599024772644), Row(artistname_index=112985, rating=0.10864003002643585), Row(artistname_index=37366,
rating=0.10733484476804733), Row(artistname_index=51082, rating=0.09390018880367279), Row(artistname_index=135095,
rating=0.0905061587691307), Row(artistname_index=243033, rating=0.08670885115861893), Row(artistname_index=190699,
rating=0.08421992510557175), Row(artistname_index=188137, rating=0.08420371264219284), Row(artistname_index=85883,
rating=0.08340750634670258)]),
 Row(user_id_index=2, recommendations=[Row(artistname_index=51082, rating=0.05577825382351875), Row(artistname_index=76809,
rating=0.05316533148288727), Row(artistname_index=149394, rating=0.05300728231668472), Row(artistname_index=126554,
rating=0.04858596622943878), Row(artistname_index=101031, rating=0.047018200159072876), Row(artistname_index=89915,
rating=0.04597727581858635), Row(artistname_index=85883, rating=0.044620268046855927), Row(artistname_index=33920,
rating=0.0416952446103096), Row(artistname_index=250287, rating=0.041003886610269547), Row(artistname_index=119829,
rating=0.040725190192461014)]),
 Row(user_id_index=3, recommendations=[Row(artistname_index=210588, rating=0.030846279114484787), Row(artistname_index=28600,
rating=0.028309248387813568), Row(artistname_index=51082, rating=0.02540583349764347), Row(artistname_index=6178,
rating=0.024118760600686073), Row(artistname_index=33914, rating=0.02337726205587387), Row(artistname_index=126554,
rating=0.02053743600845337), Row(artistname_index=161852, rating=0.019279845058918), Row(artistname_index=135095,
rating=0.018824901431798935), Row(artistname_index=157985, rating=0.017690397799015045), Row(artistname_index=10727,
rating=0.017658183351159096)]),
 Row(user_id_index=4, recommendations=[Row(artistname_index=51082, rating=0.4566318988800049), Row(artistname_index=28600,
rating=0.4060654640197754), Row(artistname_index=135095, rating=0.39582130312919617), Row(artistname_index=177187,
rating=0.3942975699901581), Row(artistname_index=89362, rating=0.39415091276168823), Row(artistname_index=163412,
rating=0.39120715856552124), Row(artistname_index=126554, rating=0.36608120799064636), Row(artistname_index=210588,
rating=0.36477309465408325), Row(artistname_index=131896, rating=0.33749303221702576), Row(artistname_index=76809,
rating=0.33517804741859436)]),
 Row(user_id_index=5, recommendations=[Row(artistname_index=37256, rating=0.3592250645160675), Row(artistname_index=32856,
rating=0.357729971408844), Row(artistname_index=63174, rating=0.3001914918422699), Row(artistname_index=89025,
rating=0.2927144169807434), Row(artistname_index=257149, rating=0.2921302318572998), Row(artistname_index=127166,
rating=0.29011568427085876), Row(artistname_index=80387, rating=0.28227561712265015), Row(artistname_index=269792,
rating=0.26432308554649353), Row(artistname_index=180982, rating=0.2621675431728363), Row(artistname_index=226782,
rating=0.26152005791664124)]),
 Row(user_id_index=6, recommendations=[Row(artistname_index=59560, rating=0.10600991547107697), Row(artistname_index=51082,
rating=0.10047482699155807), Row(artistname_index=203157, rating=0.0977829098701477), Row(artistname_index=169467,
rating=0.09565585106611252), Row(artistname_index=252681, rating=0.0955275222659111), Row(artistname_index=177444,
rating=0.09449592977762222), Row(artistname_index=32621, rating=0.09361398220062256), Row(artistname_index=63174,
rating=0.08741925656795502), Row(artistname_index=269792, rating=0.08592982590198517), Row(artistname_index=67152,
rating=0.0856465995311737)]),
 Row(user_id_index=8, recommendations=[Row(artistname_index=41327, rating=0.02837226539850235), Row(artistname_index=118249,
rating=0.02811780944466591), Row(artistname_index=69899, rating=0.027686085551977158), Row(artistname_index=217000,
rating=0.026888255029916763), Row(artistname_index=101546, rating=0.024137025699019432), Row(artistname_index=266207,
rating=0.02337983436882496), Row(artistname_index=150028, rating=0.023266948759555817), Row(artistname_index=109823,
rating=0.022870903834700584), Row(artistname_index=34153, rating=0.022860487923026085), Row(artistname_index=276726,
rating=0.022809412330389023)]),
 Row(user_id_index=9, recommendations=[Row(artistname_index=89915, rating=0.008723069913685322), Row(artistname_index=101031,
rating=0.007975384593009949), Row(artistname_index=85883, rating=0.007645925506949425), Row(artistname_index=17682,
rating=0.007015491835772991), Row(artistname_index=285318, rating=0.006918910425156355), Row(artistname_index=149394,
rating=0.00685532670468092), Row(artistname_index=76809, rating=0.006643352098762989), Row(artistname_index=112985,
rating=0.006585936993360519), Row(artistname_index=192596, rating=0.006423609796911478), Row(artistname_index=252681,
rating=0.006138001102954149)]),
 Row(user_id_index=10, recommendations=[Row(artistname_index=63521, rating=0.39124444127082825), Row(artistname_index=40206,
rating=0.35592275857925415), Row(artistname_index=21058, rating=0.33048632740974426), Row(artistname_index=198927,
rating=0.32979658246040344), Row(artistname_index=135095, rating=0.31560784578323364), Row(artistname_index=210588,
rating=0.3098471164703369), Row(artistname_index=163412, rating=0.2912026345729828), Row(artistname_index=37366,
rating=0.27047058939933777), Row(artistname_index=143141, rating=0.2605879306793213), Row(artistname_index=112985,
rating=0.2553003132343292)]),
 Row(user_id_index=11, recommendations=[Row(artistname_index=210588, rating=0.5407277941703796), Row(artistname_index=63521,
rating=0.5070255994796753), Row(artistname_index=135095, rating=0.4872817099094391), Row(artistname_index=198927,
rating=0.47255939245224), Row(artistname_index=163412, rating=0.46511709690093994), Row(artistname_index=37366,
```

10 recomendaiones para un usuario

```
user_id_index = 5914  # Usuario de ejemplo

playlist_df = userRecs.filter(fn.col("user_id_index") == user_id_index).select(fn.explode("recommendations").alias("recommendation"))
playlist_df = dims['artistname'].join(playlist_df.select(fn.col("recommendation.artistname_index").alias("artistname_index"), fn.col("rec
              .orderBy("recommendation_score", ascending=False)
playlist_df.show()
```

```
+----------------+---------------+--------------------+
|artistname_index|     artistname|recommendation_score|
+----------------+---------------+--------------------+
|          157985|        Madonna|          0.02455879|
|           63174|    David Bowie|         0.023608776|
|          210588|        Rihanna|         0.022057204|
|          169467|Michael Jackson|          0.02189978|
|           28600|        Beyoncé|         0.021125346|
|          164255|    Marvin Gaye|         0.020584796|
|          236202|  Stevie Wonder|         0.020487295|
|          143141|      Lady Gaga|         0.018550886|
|           36605| Britney Spears|         0.017046666|
|           17755|Aretha Franklin|         0.016739469|
+----------------+---------------+--------------------+
```

Los artistas que ya ha reproducido el usuario.

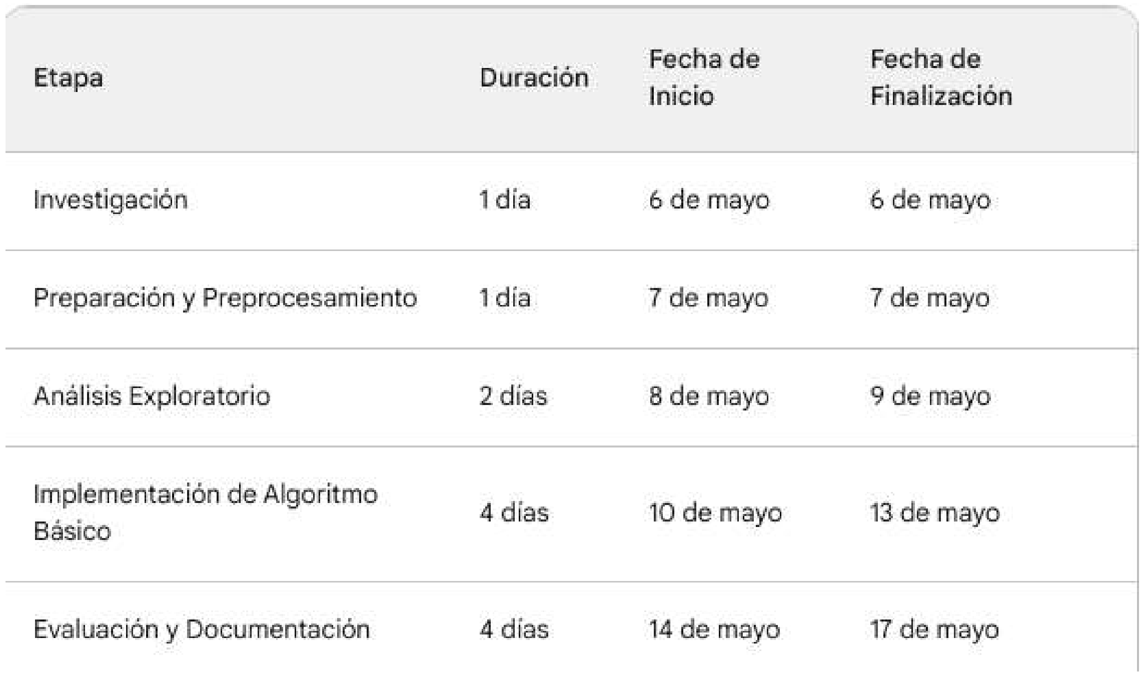
```
usr_reps = counts_df.filter(fn.col("user_id_index") == user_id_index).distinct()\
    .join(dims['artistname'],'artistname_index').orderBy('reproductions', ascending=False).select('artistname','reproductions')

usr_reps.show()
```

```
+-------------+-------------+
|   artistname|reproductions|
+-------------+-------------+
|      Madonna|          290|
|Talking Heads|          132|
| Lana Del Rey|           12|
|   Theme Park|            5|
|  Hybrid Funk|            2|
|     Smokeman|            1|
+-------------+-------------+
```

```
#Eliminando los artistas que el usuario ya ha reproducido

print('Recomendaciones de artistas para el usuario:\n')
playlist_df.join(usr_reps, playlist_df["artistname"] == usr_reps["artistname"], "left_anti").show()
```

```
Recomendaciones de artistas para el usuario:

+----------------+---------------+--------------------+
|artistname_index|     artistname|recommendation_score|
+----------------+---------------+--------------------+
|           63174|    David Bowie|         0.023608776|
|          210588|        Rihanna|         0.022057204|
|          169467|Michael Jackson|          0.02189978|
|           28600|        Beyoncé|         0.021125346|
|          164255|    Marvin Gaye|         0.020584796|
|          236202|  Stevie Wonder|         0.020487295|
|          143141|      Lady Gaga|         0.018550886|
|           36605| Britney Spears|         0.017046666|
|           17755|Aretha Franklin|         0.016739469|
+----------------+---------------+--------------------+
```

Cronograma del Proyecto (Estimado)



Etapa	Duración	Fecha de Inicio	Fecha de Finalización
Investigación	1 día	6 de mayo	6 de mayo
Preparación y Preprocesamiento	1 día	7 de mayo	7 de mayo
Análisis Exploratorio	2 días	8 de mayo	9 de mayo
Implementación de Algoritmo Básico	4 días	10 de mayo	13 de mayo
Evaluación y Documentación	4 días	14 de mayo	17 de mayo