Project Report

For

Data Engineering

For the Academic Year 2025 - April

Submitted by

Hanadi Ghlaib 443011994

Rama Sabbagh 443011958

Nada Al-qurashi 443005586

Supervisor:

Dr. Maram Almaghrabi

جامعة أم القرى
UMM AL-QURA UNIVERSITY

Department of Software Engineering

UQU

Table of Content

Database overview 3

ER- Diagram (UML) 3

Database Normalization Process 4

Schema Mapping Using Normalized Structures 4

Phase 1: Relational Database 5

1.1 Design a normalized relational schema 5

1.2 Create tables and insert sample data 6

1.3 Perform CRUD operations using SQL and Python 7

1.4 Apply indexing and query optimization 8

Phase 2: NoSQL Database 9

2.1 Transform part of the data into a document or key-value format 9

2.2 Insert data using a Python-based NoSQL library (Firebase) 9

2.3 Perform basic queries 10

Phase 3: Stream Processing 11

3.1 Simulate a data stream (e.g., JSON or CSV) 12

3.2 Use PySpark to filter and process the data 12

3.3 Save or display the processed output 13

Phase 4: Integration 14

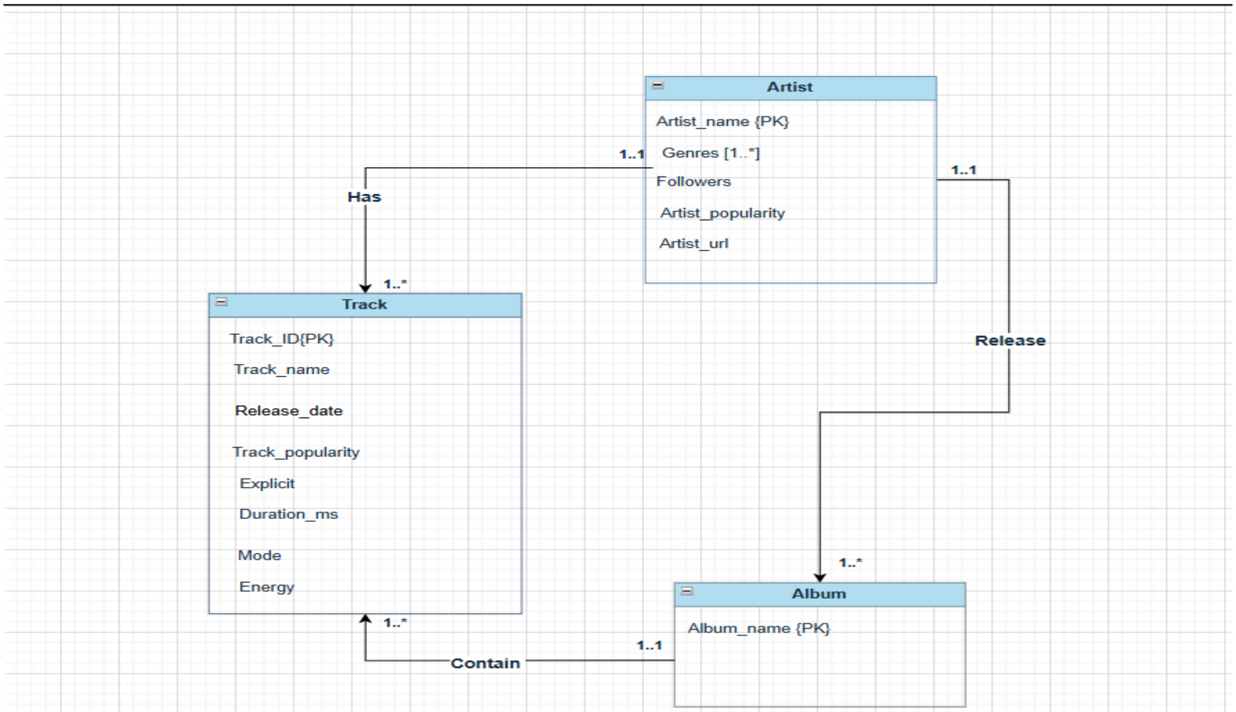
4.1 Merge all components into a unified pipeline dashboard 14

4.2 Learning Outcomes 16

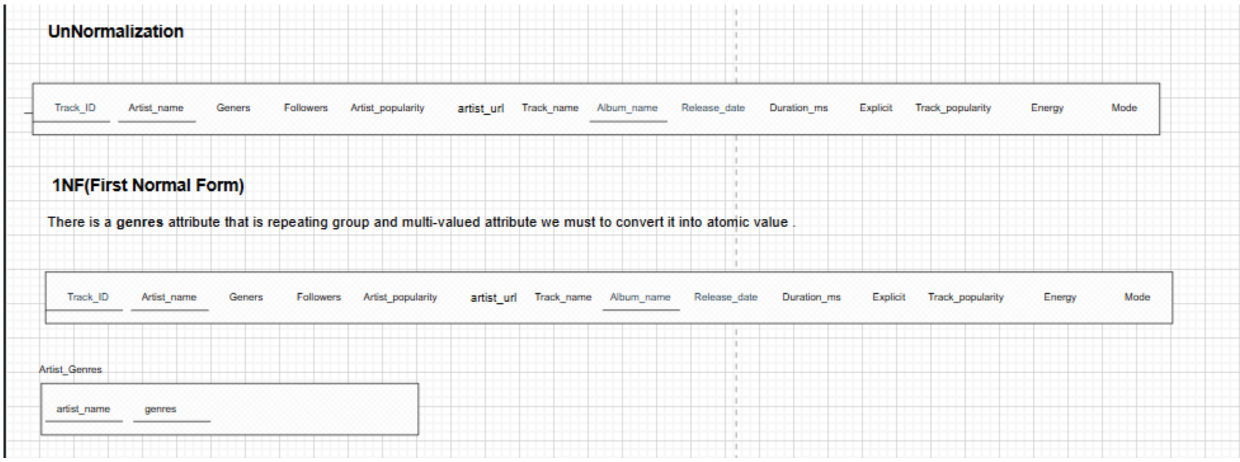
Database overview

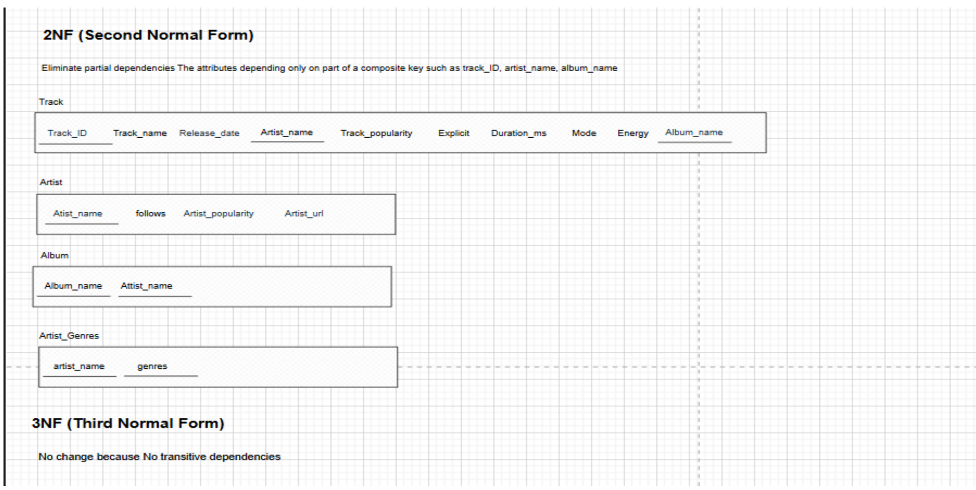
The **Spotify Songs and Artists Dataset | Audio Features** provides detailed information about songs and artists on the Spotify platform. It includes both artist-related metadata and audio features for each track. This dataset is useful for music analysis, building recommendation systems, or machine learning projects involving audio data.

ER- Diagram (UML)

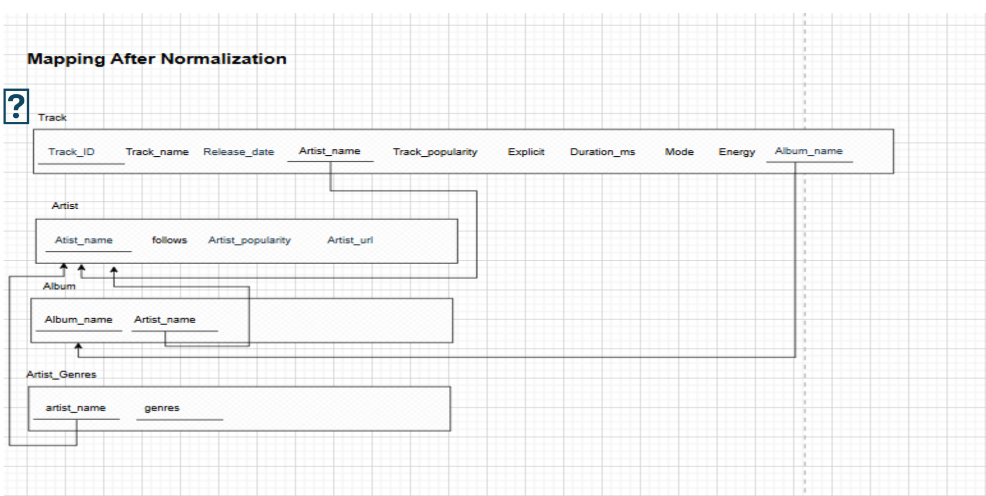


Database Normalization Process





Schema Mapping Using Normalized Structures



Phase 1: Relational Database

1.1 Design a normalized relational schema

```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files  No file chosen        Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving track database (1).csv to track database (1) (1).csv
```

Database before Normalization

** Database Before Normalization**

```
import pandas as pd

# Read the uploaded CSV file using the filename from 'uploaded' dictionary
df = pd.read_csv("track database (1).csv")  # Use the filename directly

# Show the first few rows
df.head()
```

	track_ID	artist_name	genres	followers	artist_popularity	artist_url	track_name	album_name	release_
	0	Ariana Grande	pop	98934105	89	https://open.spotify.com/artist/66CXWjxzNUsdJx...	we can't be friends wait for your love	eternal sunshine	3/8/
	1	Ariana Grande	pop	98934105	85	https://open.spotify.com/artist/66CXWjxzNUsdJx...	the boy is mine	eternal sunshine	3/8/
	2	Ariana	pop	98934105	83	https://open.spotify.com/artist/66CXWjxzNUsdJx...	intro	eternal	3/8/

Create tables after Normalization

```
cursor.execute("""
CREATE TABLE IF NOT EXISTS Artist (
    artist_name TEXT PRIMARY KEY,
    followers INTEGER,
    artist_popularity INTEGER,
    artist_url TEXT
)
""")

cursor.execute("""
CREATE TABLE IF NOT EXISTS Artist_Genres (
    artist_name TEXT,
    genres TEXT,
    PRIMARY KEY (artist_name, genres),
    FOREIGN KEY (artist_name) REFERENCES Artist(artist_name)
)
""")
```

```
CREATE TABLE Album (
    album_name TEXT PRIMARY KEY,
    artist_name TEXT,
    FOREIGN KEY (artist_name) REFERENCES Artist(artist_name)
);
""")

cursor.execute("""
CREATE TABLE Track (
    track_ID TEXT PRIMARY KEY,
    track_name TEXT,
    artist_name TEXT,
    album_name TEXT,
    release_date TEXT,
    duration_ms INTEGER,
    explicit BOOLEAN,
    track_popularity INTEGER,
    energy REAL,
    mode INTEGER,
    FOREIGN KEY (artist_name) REFERENCES Artist(artist_name),
    FOREIGN KEY (album_name) REFERENCES Album(album_name)
```

Split multi-value in column genres into atomic value in each row

```
# CSV
songs = pd.read_csv("track database (1).csv")

# spilt the gener multi-value into specific one value in each row
songs['genres'] = songs['genres'].astype(str).apply(lambda x: re.sub(r"[\[\]' ]", "", x))
songs['genres'] = songs['genres'].str.split(',')

# add each value in one row
artist_genres_df = songs.explode('genres')[['artist_name', 'genres']].drop_duplicates()
artist_genres_df.rename(columns={'genres': 'genre'}, inplace=True)
```

1.2 Create tables and insert sample data

```
# إدخال البيانات في SQLite
artist_df.to_sql('Artist', conn, if_exists='replace', index=False)
artist_genres_df.to_sql('Artist_Genres', conn, if_exists='replace', index=False)
album_df.to_sql('Album', conn, if_exists='replace', index=False)
track_df.to_sql('Track', conn, if_exists='replace', index=False)
```

View tables after normalization

```
#  View tables After Normalization
print(" Artist_Genres After Normalization :")
for row in cursor.execute("SELECT * FROM Artist_Genres "):
    print(row)
#    df = pd.read_sql_query(query, conn)
# df
```

```
 Artist_Genres After Normalization :
('Ariana Grande', 'pop')
('Adele', 'britishsoul')
('A$AP Rocky', 'rap')
('A$AP Rocky', 'hiphop')
```

```
[ ] #  View tables After Normalization
    print(" Track  After Normalization:")
    for row in cursor.execute("SELECT * FROM Track "):
        print(row)
```

```
 Track  After Normalization:
(0, "we can't be friends wait for your love", 'Ariana Grande', 'eternal sunshine', '3/8/2024', 228639, 0, 89, 0.646, 1)
(1, 'the boy is mine', 'Ariana Grande', 'eternal sunshine', '3/8/2024', 173639, 1, 85, 0.63, 0)
```

```
[ ] # 📋 View tables After Normalization
    print("🎙 Artist After Normalization:")
    for row in cursor.execute("SELECT * FROM Artist "):
        print(row)
```

```
🎙 Artist After Normalization:
('Taylor Swift', 119286617, 92, 'https://open.spotify.com/artist/06HL4z0CvFAxyc27GXpf02')
('Arctic Monkeys', 26055385, 91, 'https://open.spotify.com/artist/7Ln80lUS6He07XvHI8qqHH')
('Ariana Grande', 98934105, 89, 'https://open.spotify.com/artist/66CXWjxzNUsdJxJ2JdwvnR')
('Gracie Abrams', 2232980, 87, 'https://open.spotify.com/artist/4tuJ0bMpJh08umKkEXKUI5')
```

```
[ ]  #  View tables After Normalization
     print(" Album  After Normalization")
     for row in cursor.execute("SELECT * FROM Album "):
         print(row)
```

```
 Album  After Normalization
('eternal sunshine', 'Ariana Grande')
('After Hours (Deluxe)', 'Ariana Grande')
```

1.3 Perform CRUD operations using SQL and Python

Insert Artiest to the table

```
[ ] # Insert into Artist table
    cursor.execute("""
    INSERT INTO Artist (artist_name, followers, artist_popularity, artist_url)
    VALUES (?, ?, ?, ?)
    """, (
        "The Weeknd", 82000000, 96, "https://open.spotify.com/artist/1Xyo4u8uXC1ZmMpatF05PJ"
    ))
```

Read the Artiest Data

Read The Data

```
cursor.execute("SELECT * FROM Artist WHERE artist_name = ?", ("Ariana Grande",))
result = cursor.fetchone()
print(result)
```

```
('Ariana Grande', 98934105, 89, 'https://open.spotify.com/artist/66CXWjxzNUsdJxJ2JdwvnR')
```

Update Artiest Data

UPDATE THE FOLLOWING'S ARTIST

```
cursor.execute("""
UPDATE Artist
SET followers = ?
WHERE artist_name = ?
""", (992000000, "Ariana Grande"))
conn.commit()
```

```
[ ] cursor.execute("SELECT * FROM Artist WHERE artist_name = ?", ("Ariana Grande",))
    result = cursor.fetchone()  #print the
    print(result)
```

```
('Ariana Grande', 992000000, 89, 'https://open.spotify.com/artist/66CXWjxzNUsdJxJ2JdwvnR')
```

Delete One Artist

```python
conn = sqlite3.connect("university.db")
cursor = conn.cursor()

# Delete the track
cursor.execute("DELETE FROM Track WHERE track_name = ?", ("the boy is mine",))
conn.commit()

# Check if it still exists
cursor.execute("SELECT * FROM Track WHERE track_name = ?", ("the boy is mine",))
result = cursor.fetchall()

if result:
    print("Track still exists:", result)
else:
    print("Track successfully deleted.")

conn.close()
```

```
Track successfully deleted.
```

1.4 Apply indexing and query optimization

```python
# Time measurement without indexing
start = time.time()
cursor.execute("SELECT * FROM Track WHERE artist_name = 'Andrew Underberg'")
cursor.fetchall()
print(" Time without index:", time.time() - start)

# create insex to artist_name
cursor.execute("CREATE INDEX IF NOT EXISTS idx_track_artist ON Track(artist_name)")
conn.commit()

# Time Measurement after indexing
start = time.time()
cursor.execute("SELECT * FROM Track WHERE artist_name = 'Andrew Underberg'")
cursor.fetchall()
print(" Time with index:", time.time() - start)

conn.close()
```

```
Time without index: 0.0010018348693847656
Time with index: 0.0003085136413574219
```

Phase 2: NoSQL Database

2.1 Transform part of the data into a document or key-value format

```
[5] cred = credentials.Certificate("/content/drive/My Drive/firebase.json")
    firebase_admin.initialize_app(cred)

    db = firestore.client()
```

```
df = pd.read_csv("/content/drive/My Drive/song1 database.csv")
df.head() #fun print 5 rows by defult
```

	track_ID	artist_name	genres	followers	artist_popularity	artist_url	track_name	album_name	release_date	duration_ms	explicit	track_popularity
0	0	Ariana Grande	pop	98934105	89	https://open.spotify.com/artist/66CXWjxzNUsdJx...	we can't be friends (wait for your love)	eternal sunshine	3/8/2024	228639	False	89
1	1	Ariana Grande	pop	98934105	85	https://open.spotify.com/artist/66CXWjxzNUsdJx...	the boy is mine	eternal sunshine	3/8/2024	173639	True	85
2	2	Ariana Grande	pop	98934105	83	https://open.spotify.com/artist/66CXWjxzNUsdJx...	intro (end of the world)	eternal sunshine	3/8/2024	92400	True	83
3	3	Ariana Grande	pop	98934105	80	https://open.spotify.com/artist/66CXWjxzNUsdJx...	Save Your Tears (Remix) (with Ariana Grande) -...	After Hours (Deluxe)	3/20/2020	191013	False	80
4	4	Ariana Grande	pop	98934105	79	https://open.spotify.com/artist/66CXWjxzNUsdJx...	yes, and?	eternal sunshine	3/8/2024	214994	True	79

Next steps: Generate code with df | View recommended plots | New interactive sheet

2.2 Insert data using a Python-based NoSQL library (Firebase)

- I used the firebase_admin Python library.

- Read a CSV file using pandas.

- Extract and structure the data row by row.

- Insert each row as a document into Firebase Firestore, using: python db.collection("spotify_songs").add(doc)

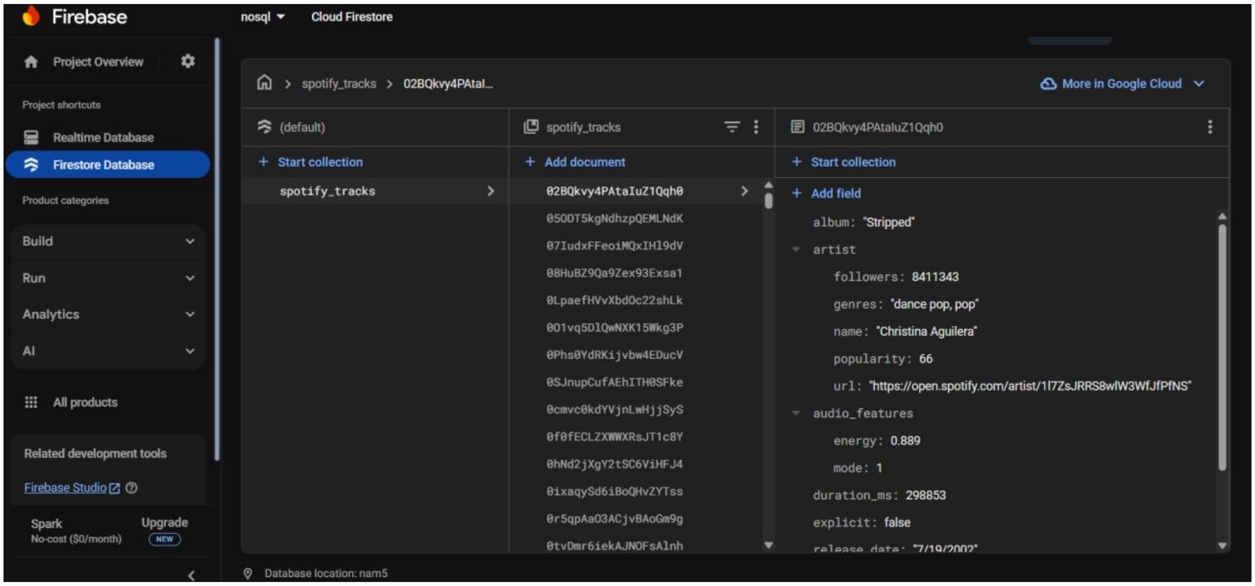
```
[ ] import firebase_admin
    from firebase_admin import credentials, firestore
    import pandas as pd
```

```
[36]
df = pd.read_csv("/content/drive/My Drive/song1 database.csv")

for idx, row in df.iterrows():
    doc = {
        "track_id": row.get("track_ID"),
        "track_name": row.get("track_name"),
        "artist": {
            "name": row.get("artist_name"),
            "popularity": row.get("artist_popularity"),
            "followers": row.get("followers"),
            "genres": row.get("genres"),
            "url": row.get("artist_url")
        },
        "album": row.get("album_name"),
        "release_date": row.get("release_date"),
        "duration_ms": row.get("duration_ms"),
        "explicit": row.get("explicit"),
        "track_popularity": row.get("track_popularity"),
        "audio_features": {
            "energy": row.get("energy"),
            "mode": row.get("mode")
        }
    }
    db.collection("spotify_tracks").add(doc)

print("Inserted data into Firebase Firestore successfully.")
```

```
Inserted data into Firebase Firestore successfully.
```



2.3 Perform basic queries

```python
# get all songs query
songs = db.collection("spotify_tracks").order_by('track_id').stream()
for song in songs:
    print(song.to_dict())
```

```
{'track_id': 0, 'track_popularity': 89, 'release_date': '3/8/2024', 'duration_ms': 228639, 'audio_features': {'energy': 0.646, 'mode': 1}, 'track_name': "we can't be friends (wait fo
{'track_id': 1, 'track_popularity': 85, 'release_date': '3/8/2024', 'duration_ms': 173639, 'audio_features': {'energy': 0.63, 'mode': 0}, 'track_name': 'the boy is mine', 'album': 'e
{'track_id': 2, 'track_popularity': 83, 'release_date': '3/8/2024', 'duration_ms': 92400, 'audio_features': {'energy': 0.362, 'mode': 1}, 'track_name': 'intro (end of the world)', 'a
{'track_id': 3, 'track_popularity': 80, 'release_date': '3/20/2020', 'duration_ms': 191013, 'audio_features': {'energy': 0.825, 'mode': 1}, 'track_name': 'Save Your Tears (Remix) (wi
{'track_id': 4, 'track_popularity': 79, 'release_date': '3/8/2024', 'duration_ms': 214994, 'audio_features': {'energy': 0.775, 'mode': 1}, 'track_name': 'yes, and?', 'album': 'eterna
{'track_id': 5, 'track_popularity': 82, 'release_date': '8/22/2014', 'duration_ms': 197266, 'audio_features': {'energy': 0.593, 'mode': 1}, 'track_name': 'One Last Time', 'album': 'M
{'track_id': 6, 'track_popularity': 78, 'release_date': '3/14/2023', 'duration_ms': 232857, 'audio_features': {'energy': 0.485, 'mode': 0}, 'track_name': 'Die For You (with Ariana Gr
{'track_id': 7, 'track_popularity': 82, 'release_date': '2/8/2019', 'duration_ms': 178626, 'audio_features': {'energy': 0.317, 'mode': 0}, 'track_name': '7 rings', 'album': 'thank u,
{'track_id': 8, 'track_popularity': 70, 'release_date': '5/20/2016', 'duration_ms': 235946, 'audio_features': {'energy': 0.602, 'mode': 0}, 'track_name': 'Dangerous Woman', 'album':
{'track_id': 9, 'track_popularity': 70, 'release_date': '5/20/2016', 'duration_ms': 244453, 'audio_features': {'energy': 0.734, 'mode': 1}, 'track_name': 'Into You', 'album': 'Danger
{'track_id': 10, 'track_popularity': 84, 'release_date': '11/19/2021', 'duration_ms': 224694, 'audio_features': {'energy': 0.366, 'mode': 1}, 'track_name': 'Easy On Me', 'album': '30
{'track_id': 11, 'track_popularity': 79, 'release_date': '1/24/2011', 'duration_ms': 242973, 'audio_features': {'energy': 0.67, 'mode': 0}, 'track_name': 'Set Fire to the Rain', 'alb
{'track_id': 12, 'track_popularity': 78, 'release_date': '1/24/2011', 'duration_ms': 285240, 'audio_features': {'energy': 0.319, 'mode': 1}, 'track_name': 'Someone Like You', 'album'
{'track_id': 13, 'track_popularity': 79, 'release_date': '1/24/2011', 'duration_ms': 228093, 'audio_features': {'energy': 0.769, 'mode': 1}, 'track_name': 'Rolling in the Deep', 'alb
{'track_id': 14, 'track_popularity': 84, 'release_date': '10/4/2012', 'duration_ms': 286480, 'audio_features': {'energy': 0.552, 'mode': 0}, 'track_name': 'Skyfall', 'album': 'Skyfal
{'track_id': 15, 'track_popularity': 74, 'release_date': '11/20/2015', 'duration_ms': 285935, 'audio_features': {'energy': 0.341, 'mode': 0}, 'track_name': 'Love In The Dark', 'album
{'track_id': 16, 'track_popularity': 73, 'release_date': '11/20/2015', 'duration_ms': 290900, 'audio_features': {'energy': 0.595, 'mode': 1}, 'track_name': 'When We Were Young', 'alb
{'track_id': 17, 'track_popularity': 72, 'release_date': '1/28/2008', 'duration_ms': 212040, 'audio_features': {'energy': 0.172, 'mode': 1}, 'track_name': 'Make You Feel My Love', 'a
{'track_id': 18, 'track_popularity': 70, 'release_date': '1/28/2008', 'duration_ms': 210506, 'audio_features': {'energy': 0.47, 'mode': 0}, 'track_name': 'Chasing Pavements', 'album'
{'track_id': 19, 'track_popularity': 72, 'release_date': '11/20/2015', 'duration_ms': 295502, 'audio_features': {'energy': 0.43, 'mode': 0}, 'track_name': 'Hello', 'album': '25', 'ar
{'track_id': 20, 'track_popularity': 83, 'release_date': '11/20/2018', 'duration_ms': 158205, 'audio_features': {'energy': 0.707, 'mode': 1}, 'track_name': 'Sundress', 'album': 'Sund
{'track_id': 21, 'track_popularity': 83, 'release_date': '8/18/2023', 'duration_ms': 190285, 'audio_features': {'energy': 0.652, 'mode': 1}, 'track_name': 'I Smoked Away My Brain', '
{'track_id': 22, 'track_popularity': 81, 'release_date': '5/25/2018', 'duration_ms': 205040, 'audio_features': {'energy': 0.569, 'mode': 0}, 'track_name': 'Praise The Lord (Da Shine)
{'track_id': 23, 'track_popularity': 79, 'release_date': '5/26/2015', 'duration_ms': 260986, 'audio_features': {'energy': 0.661, 'mode': 1}, 'track_name': 'Everyday (feat. Rod Stewar
{'track_id': 24, 'track_popularity': 77, 'release_date': '8/2/2024', 'duration_ms': 190074, 'audio_features': {'energy': 0.682, 'mode': 1}, 'track_name': 'HIGHJACK', 'album': 'HIGHJA
{'track_id': 25, 'track_popularity': 77, 'release_date': '2013', 'duration_ms': 236280, 'audio_features': {'energy': 0.819, 'mode': 1}, 'track_name': 'Fashion Killa', 'album': 'LONG.
{'track_id': 26, 'track_popularity': 76, 'release_date': '6/2/2023', 'duration_ms': 256026, 'audio_features': {'energy': 0.537, 'mode': 0}, 'track_name': 'Am I Dreaming (Metro Boomin
{'track_id': 27, 'track_popularity': 76, 'release_date': '2013', 'duration_ms': 233786, 'audio_features': {'energy': 0.693, 'mode': 1}, 'track_name': "F**kin' Problems (feat. Drake,
{'track_id': 28, 'track_popularity': 76, 'release_date': '2013', 'duration_ms': 220133, 'audio_features': {'energy': 0.427, 'mode': 0}, 'track_name': 'LVL', 'album': 'LONG.LIVE.A$AP
{'track_id': 29, 'track_popularity': 72, 'release_date': '5/26/2015', 'duration_ms': 323960, 'audio_features': {'energy': 0.803, 'mode': 0}, 'track_name': 'Jukebox Joints (feat. Joe
{'track_id': 30, 'track_popularity': 92, 'release_date': '8/23/2019', 'duration_ms': 178426, 'audio_features': {'energy': 0.702, 'mode': 1}, 'track_name': 'Cruel Summer', 'album': 'L
{'track_id': 31, 'track_popularity': 89, 'release_date': '4/18/2024', 'duration_ms': 228965, 'audio_features': {'energy': 0.386, 'mode': 1}, 'track_name': 'Fortnight (feat. Post Malo
{'track_id': 32, 'track_popularity': 87, 'release_date': '4/18/2024', 'duration_ms': 218004, 'audio_features': {'energy': 0.751, 'mode': 1}, 'track_name': 'I Can Do It With a Broken
{'track_id': 33, 'track_popularity': 87, 'release_date': '7/24/2020', 'duration_ms': 261922, 'audio_features': {'energy': 0.623, 'mode': 1}, 'track_name': 'august', 'album': 'folklor
```

```python
# Find songs by artist query
query = db.collection("spotify_tracks").where("artist.name", "==", "Ariana Grande").stream()

for doc in query:
    data = doc.to_dict()
    print(f"{data['track_name']} by {data['artist']['name']} (popularity: {data['artist']['popularity']})")
```

```
7 rings by Ariana Grande (popularity: 82)
yes, and? by Ariana Grande (popularity: 79)
Die For You (with Ariana Grande) - Remix by Ariana Grande (popularity: 78)
Dangerous Woman by Ariana Grande (popularity: 70)
intro (end of the world) by Ariana Grande (popularity: 83)
One Last Time by Ariana Grande (popularity: 82)
Save Your Tears (Remix) (with Ariana Grande) - Bonus Track by Ariana Grande (popularity: 80)
the boy is mine by Ariana Grande (popularity: 85)
Into You by Ariana Grande (popularity: 70)
we can't be friends (wait for your love) by Ariana Grande (popularity: 89)
/usr/local/lib/python3.11/dist-packages/google/cloud/firestore_v1/base_collection.py:303: UserWarning: Detected filter using positional arguments. Prefer using the 'filter' keyword a
  return query.where(field_path, op_string, value)
```

Phase 3: Stream Processing

You can find the work regarding this phase in the file “Tracks_Stream_Processing(phase_3).ipynb”

The process began with the installation of PySpark, followed by the initialization of a SparkSession named "SongsStreamLab". A basic test confirmed that the Spark environment was functioning as expected.

3.1 Simulate a data stream (e.g., JSON or CSV)

Simulate a data stream (e.g., JSON or CSV)

```
[3]
# Simulate tracks/songs records (like CSV rows)
data = [
    ("101", "Ed Sheeran", ["pop", "uk pop"], 93639154, 96, "https://open.spotify.com/artist/6eUKZXaKkcviH0Ku9w2n3V", "Shivers"
    ("102", "Nirvana", ["grunge", "rock"], 16802723, 87, "https://open.spotify.com/artist/6olE6TJLqED3rqDCT0FyPh", "Smells Lik
    ("103", "The Weeknd", ["canadian pop", "r&b"], 51478288, 98, "https://open.spotify.com/artist/1Xyo4u8uXC1ZmMpatF05PJ", "Bl
    ("104", "Eminem", ["hip hop", "rap"], 50367072, 97, "https://open.spotify.com/artist/7dGJo4pcD2V6oG8kP0tJRR", "Lose Yourse
]

columns = [
    "track_ID", "artist_name", "genres", "followers", "artist_popularity", "artist_url",
    "track_name", "album_name", "release_date", "duration_ms", "explicit",
    "track_popularity", "energy", "mode"
]

# Create DataFrame
df = spark.createDataFrame(data, columns)
df.show()
```

```
+--------+-----------+-------------------+---------+-----------------+--------------------+----------------+-----------+---
|track_ID|artist_name|             genres|followers|artist_popularity|          artist_url|      track_name| album_name|rel
+--------+-----------+-------------------+---------+-----------------+--------------------+----------------+-----------+---
|     101| Ed Sheeran|      [pop, uk pop]| 93639154|               96|https://open.spot...|         Shivers|          =|  2
|     102|    Nirvana|    [grunge, rock]| 16802723|               87|https://open.spot...|Smells Like Teen ...|   Nevermind|  1
|     103| The Weeknd|[canadian pop, r&b]| 51478288|               98|https://open.spot...|  Blinding Lights|After Hours|  2
|     104|     Eminem|     [hip hop, rap]| 50367072|               97|https://open.spot...|   Lose Yourself|     8 Mile|  2
+--------+-----------+-------------------+---------+-----------------+--------------------+----------------+-----------+---
```

A small sample dataset was created to simulate music track records. Each record contained attributes such as artist name, genres, popularity, explicit content flag, energy, and other relevant metadata.

3.2 Use PySpark to filter and process the data

Use PySpark to filter and process the data

```
[4]
# Filter high popularity tracks/songs
popular_tracks = df.filter(df["track_popularity"] >= 85)
popular_tracks.show()
```

```
+--------+-----------+-------------------+---------+-----------------+--------------------+---------------+-----------+--------
|track_ID|artist_name|             genres|followers|artist_popularity|          artist_url|     track_name| album_name|release_
+--------+-----------+-------------------+---------+-----------------+--------------------+---------------+-----------+--------
|     103| The Weeknd|[canadian pop, r&b]| 51478288|               98|https://open.spot...|Blinding Lights|After Hours|  2019-1
|     104|     Eminem|     [hip hop, rap]| 50367072|               97|https://open.spot...|  Lose Yourself|     8 Mile|  2002-1
+--------+-----------+-------------------+---------+-----------------+--------------------+---------------+-----------+--------
```

```
[5] # Filter tracks/songs that are explicit
explicit_tracks = df.filter(df["explicit"] == True)
explicit_tracks.show()
```

```
+--------+-----------+--------------+---------+-----------------+--------------------+--------------------+----------+--------
|track_ID|artist_name|        genres|followers|artist_popularity|          artist_url|          track_name|album_name|release_c
+--------+-----------+--------------+---------+-----------------+--------------------+--------------------+----------+--------
|     102|    Nirvana|[grunge, rock]| 16802723|               87|https://open.spot...|Smells Like Teen ...| Nevermind|  1991-09
|     104|     Eminem|[hip hop, rap]| 50367072|               97|https://open.spot...|       Lose Yourself|    8 Mile|  2002-10
+--------+-----------+--------------+---------+-----------------+--------------------+--------------------+----------+--------
```

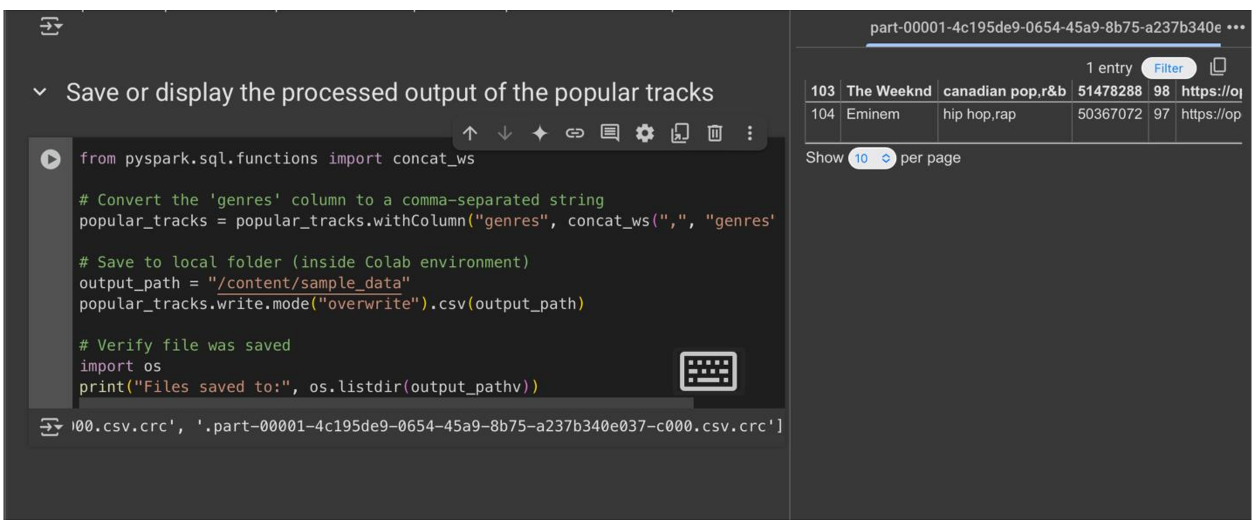
Several filters were applied to extract meaningful subsets from the dataset:

- **Popular Tracks:** Filtered to include tracks with a popularity score of 85 or higher.
- **Explicit Tracks:** Extracted based on the explicit content flag being set to true.
- **High-Energy Tracks:** Selected tracks with an energy value greater than 0.8.

```
[6] # Filter tracks/songs with energy greater than 0.8
high_energy_tracks = df.filter(df["energy"] > 0.8)
high_energy_tracks.show()

+--------+-----------+--------------+---------+-----------------+--------------------+-------------+----------+------------+---
|track_ID|artist_name|        genres|followers|artist_popularity|         artist_url|   track_name|album_name|release_date|dur
+--------+-----------+--------------+---------+-----------------+--------------------+-------------+----------+------------+---
|     101| Ed Sheeran| [pop, uk pop]| 93639154|               96|https://open.spot...|      Shivers|         =|  2021-09-10|
|     104|     Eminem|[hip hop, rap]| 50367072|               97|https://open.spot...|Lose Yourself|    8 Mile|  2002-10-28|
+--------+-----------+--------------+---------+-----------------+--------------------+-------------+----------+------------+---
```

3.3 Save or display the processed output



Save or display the processed output of the popular tracks

```
from pyspark.sql.functions import concat_ws

# Convert the 'genres' column to a comma-separated string
popular_tracks = popular_tracks.withColumn("genres", concat_ws(",", "genres'

# Save to local folder (inside Colab environment)
output_path = "/content/sample_data"
popular_tracks.write.mode("overwrite").csv(output_path)

# Verify file was saved
import os
print("Files saved to:", os.listdir(output_pathv))

)00.csv.crc', '.part-00001-4c195de9-0654-45a9-8b75-a237b340e037-c000.csv.crc']
```

part-00001-4c195de9-0654-45a9-8b75-a237b340e

1 entry Filter

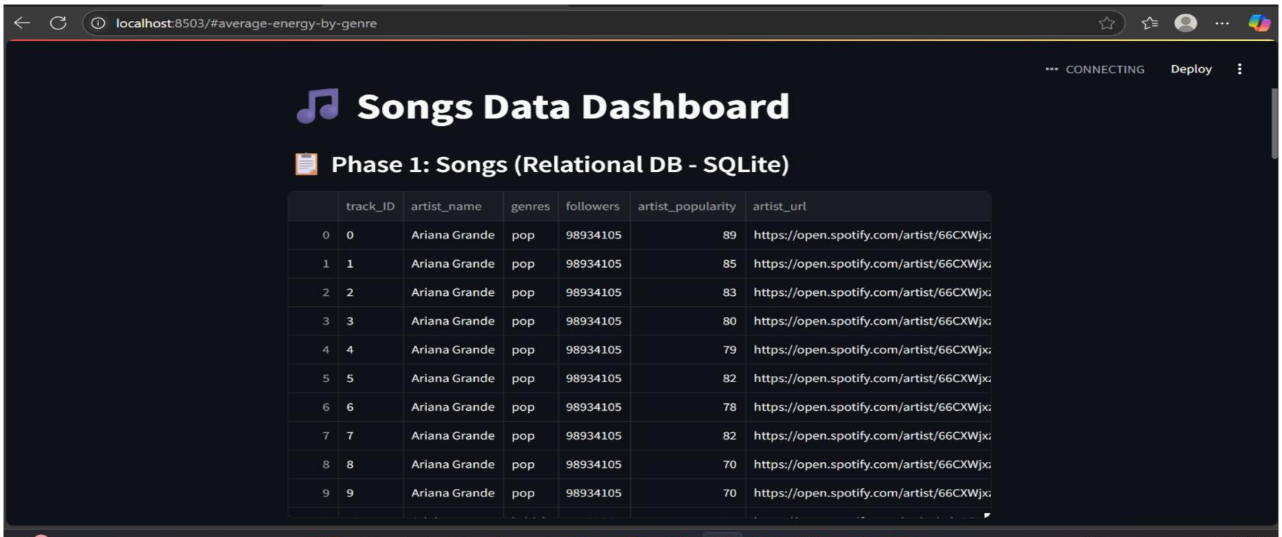
103	The Weeknd	canadian pop,r&b	51478288	98	https://oı
104	Eminem	hip hop,rap	50367072	97	https://op

Show 10 per page

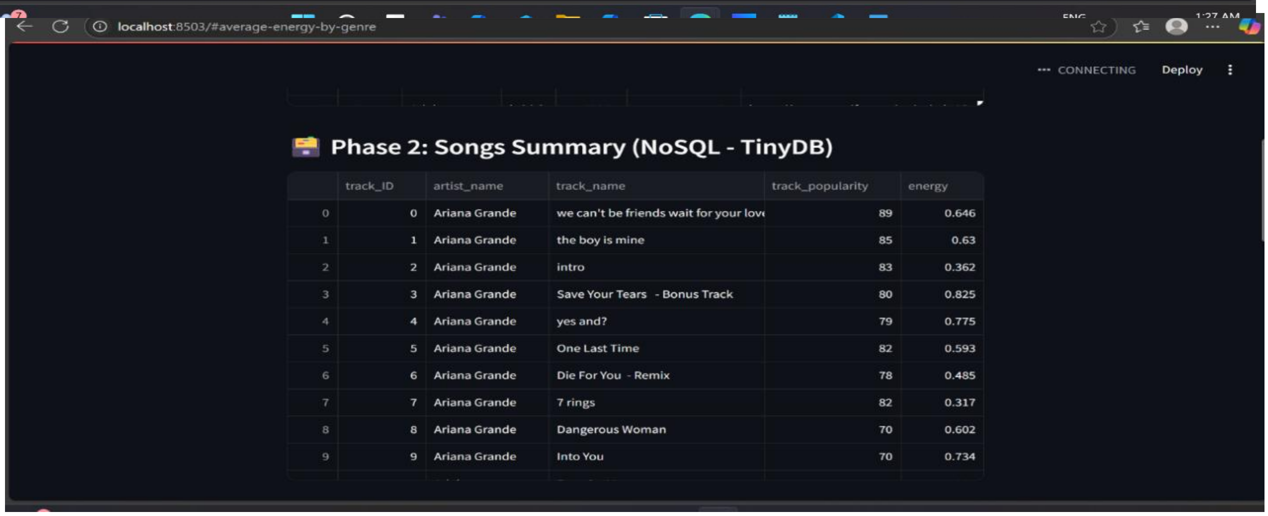
The genres column, originally stored as an array, was converted into a comma-separated string for better readability and compatibility. The resulting set of popular tracks was then saved to a local directory(sample_data) in overwrite mode.

Phase 4: Integration

4.1 Merge all components into a unified pipeline dashboard

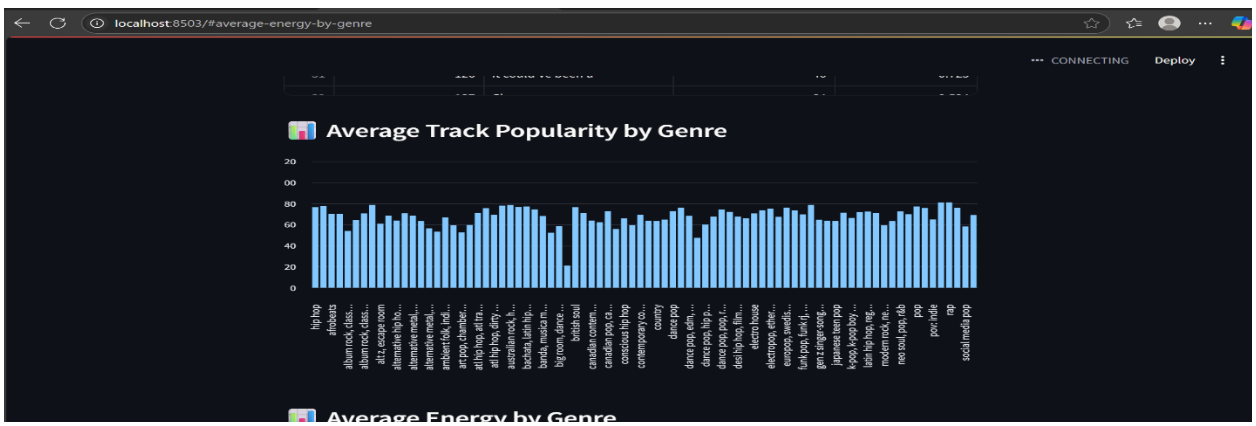


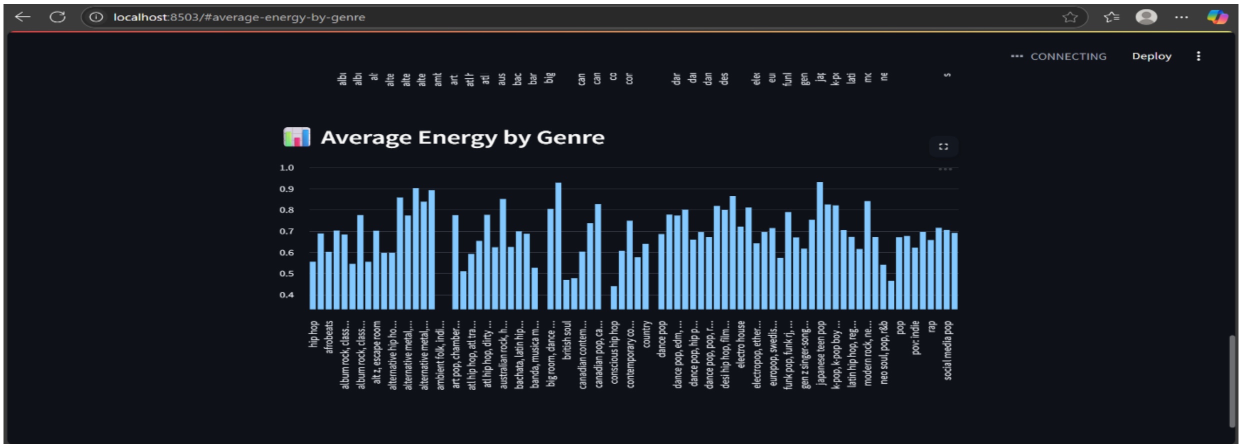
	track_ID	artist_name	genres	followers	artist_popularity	artist_url
0	0	Ariana Grande	pop	98934105	89	https://open.spotify.com/artist/66CXWjx:
1	1	Ariana Grande	pop	98934105	85	https://open.spotify.com/artist/66CXWjx:
2	2	Ariana Grande	pop	98934105	83	https://open.spotify.com/artist/66CXWjx:
3	3	Ariana Grande	pop	98934105	80	https://open.spotify.com/artist/66CXWjx:
4	4	Ariana Grande	pop	98934105	79	https://open.spotify.com/artist/66CXWjx:
5	5	Ariana Grande	pop	98934105	82	https://open.spotify.com/artist/66CXWjx:
6	6	Ariana Grande	pop	98934105	78	https://open.spotify.com/artist/66CXWjx:
7	7	Ariana Grande	pop	98934105	82	https://open.spotify.com/artist/66CXWjx:
8	8	Ariana Grande	pop	98934105	70	https://open.spotify.com/artist/66CXWjx:
9	9	Ariana Grande	pop	98934105	70	https://open.spotify.com/artist/66CXWjx:



	track_ID	artist_name	track_name	track_popularity	energy
0	0	Ariana Grande	we can't be friends wait for your love	89	0.646
1	1	Ariana Grande	the boy is mine	85	0.63
2	2	Ariana Grande	intro	83	0.362
3	3	Ariana Grande	Save Your Tears - Bonus Track	80	0.825
4	4	Ariana Grande	yes and?	79	0.775
5	5	Ariana Grande	One Last Time	82	0.593
6	6	Ariana Grande	Die For You - Remix	78	0.485
7	7	Ariana Grande	7 rings	82	0.317
8	8	Ariana Grande	Dangerous Woman	70	0.602
9	9	Ariana Grande	Into You	70	0.734

	track_ID	track_name	popularity	energy
0	0	we can't be friends wait for you	89	0.646
1	1	the boy is mine	85	0.63
2	10	Easy On Me	84	0.366
3	100	Treehouse	80	0.196
4	101	Mary	72	0.467
5	102	Sarah	73	0.686
6	103	16 Mirrors	70	0.3
7	104	Things to Do	73	0.525
8	105	Advice	67	0.611
9	106	Race	69	0.641





4.2 Learning Outcomes

Through this project, we gained hands-on experience with both relational and NoSQL databases, learning how to design normalized schemas, implement efficient data storage, and perform CRUD operations using SQL and Python. We explored performance optimization through indexing and query tuning, and transitioned to NoSQL by transforming structured data into document format using Firebase. Additionally, we learned the fundamentals of stream processing using PySpark, including real-time filtering and transformation of streaming data. In the final phase, we merged all components into a unified pipeline dashboard, bringing together the various technologies into one cohesive system.This multi-phase project deepened our understanding of data modeling, integration across technologies, and the practical application of data processing in modern systems.