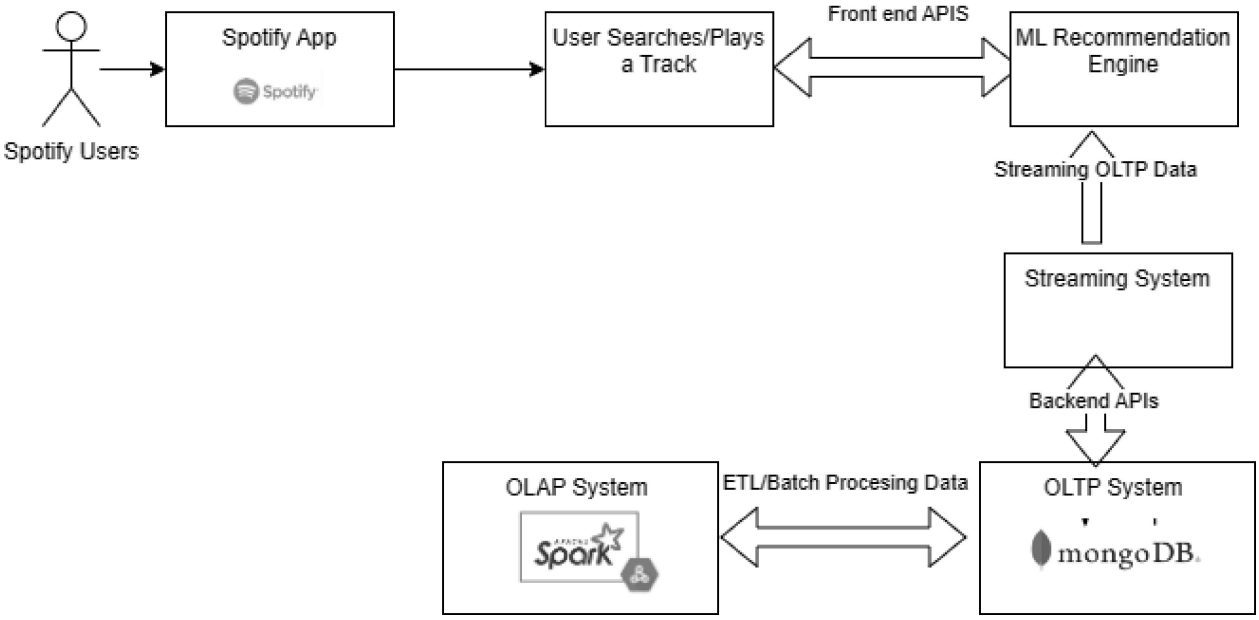
BITS_MTech_BDSAssignment_SpotifyRecommendation

Our project focuses on developing a Music Recommendation System using the Spotify Playlist Dataset. Users can search for songs by track name and artist name, select their preferred songs from the list, and receive relevant recommendations from the system.

Team Members

1. Bhaskar Kurada - 2022OG04023 - **Contribution:** Installing Spark on local, Connecting Sparkshell/Pyspark on local command line troubleshooting and the architecture diagram creation on draw.io, Github Creation
2. Mahesh Kammari - 2022OG04033 - **Contribution:** Creating MongoDB Cloud Instance, MongoDB and Pyspark connection troubleshooting, MongoDB Data Model and queries, Data Synthesization and loom video setup

Part1 - Architecture diagram of our system - BigData Recommendation Pipeline



Brief Overview of the System components and Rationale

Components FrontendAPIs: APIs to connect the mobile App to the ML Recommendation Engine and the underlying OLTP System **ML Recommendation Engine:** NLP based ML Algorithm to find similar documents for recommendation basis the attributes in the databset **Streaming Component:** Apache Kafka could be used here to stream the data from OLTP databse related services **Backend APIs:** APIs to connect the MongoDB to the streaming component **OLTP Sytem:** MongoDB to store the user details, playlists, search history and recommendations provided by ML library **OLAP System:** SparkSQL to load the master data as well as to run the analytics queries **Pyspark:** Utilized for preprocessing the Spotify Dataset. **ETL/Batch processing jobs:** To collate the user search and playlist history to dervie analytical queries for better sales and growth

Rationale for the design choices OLTP System: We have chosen the MongoDB which is a document based NOSQL DB to store the search cum playlist history and also the associated recommendations MongoDB natively provides Consistency and Performance tradeoffs with scale and durability with various read and write choices

Considering this problem of recommendation only being a part of the overall spotify app, consistency in the recommendations basis user playlist history and the search criteria takes a notch high priority than the availability.

Availability could take precedence once the user accepts the recommendation and starts playing the track which is not the scope of the project

OLTP Data Model:

Album MasterCollection Document
Track_id
Artists
album_name
track_name
popularity
duration
explicit
danceability
energy
key
loudness
mode
speechiness
acousticness
instrumentalness
liveness
valence
tempo
time_signature
track_genre

User Document
id
name

UserPlaylist Document
id
Userid
Trackid

UserPlayHistory Document
id
Userid
Trackid

UserSearchHistory Document
id
Userid
SearchItem

UserPlayRecommendations Document
id
playhistoryid
RecommendationStatus

UserSearchRecommendations Document
id
searchhistoryid
RecommendationStatus

OLAP System: Spark provides a powerful and flexible framework that can integrate well with various NoSQL databases. like Mongo, Cassandra, Dynamo and also provides RDD Dataframes which can handle huge load of data from both batch processing and also running ML based analytics on them.

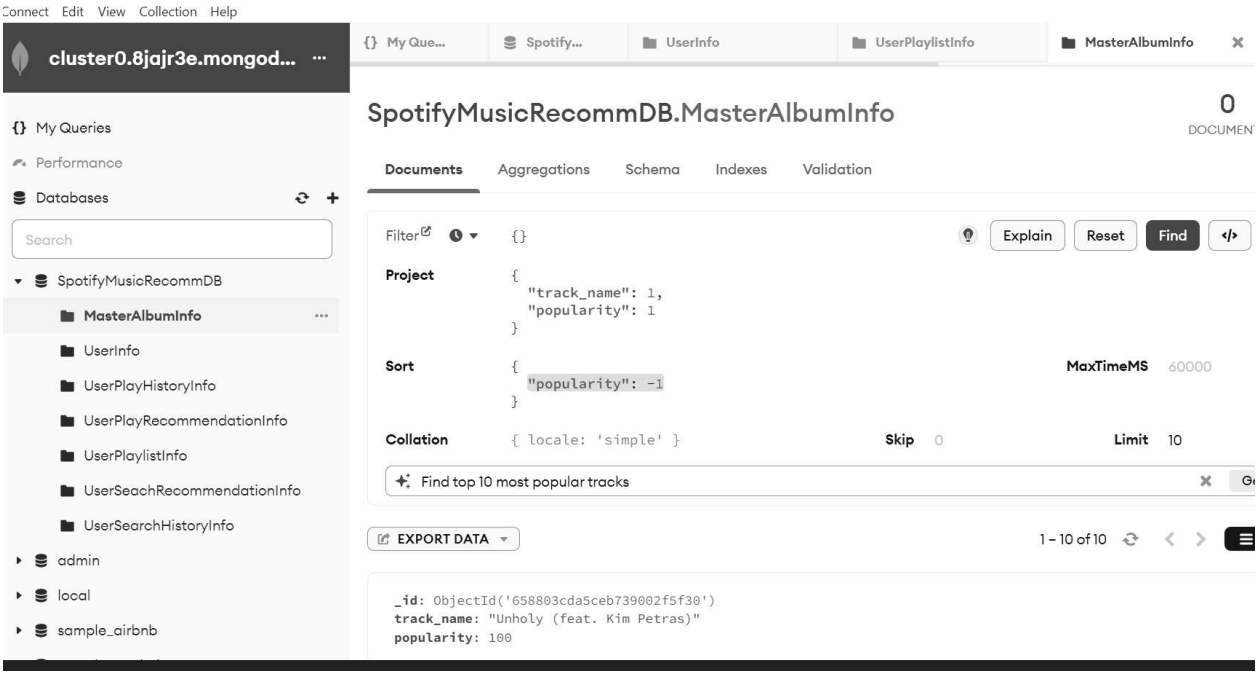
End of Part-1

Part-2 - OLTP Read/Write Queries on Mongo DB

Loom Video Link for Pymongo - Read/Write Queries

https://www.loom.com/share/1398b11cc5c84cd6afd3c2fdc29be7df?sid=9abbf41c-17a6-4f98-9673-26840758af39

Also Find Mongo Compass IDE & Pymongo Screenshots



End of Part-2

Part-3 - Pyspark & MongoDB Connection Setup & OLAP Queries

Loom Video Link for Spark - OLAP Queries

https://www.loom.com/share/d26e9a49c8ca497788a5fc1d053a6fb6

Commands to connect to Pyspark and connect to Mongo DB and then finally run the OLAP aggregate queries

pyspark --packages org.mongodb.spark:mongo-spark-connector_2.12:3.0.1

df = spark.read.format("mongo").option("uri", "mongodb+srv://kuradabhaskar:fiYR58qHxeMqlE1P@cluster0.8jajr3e.mongodb.net/SpotifyMusicRecommDB.MasterAlbumInfo").load()

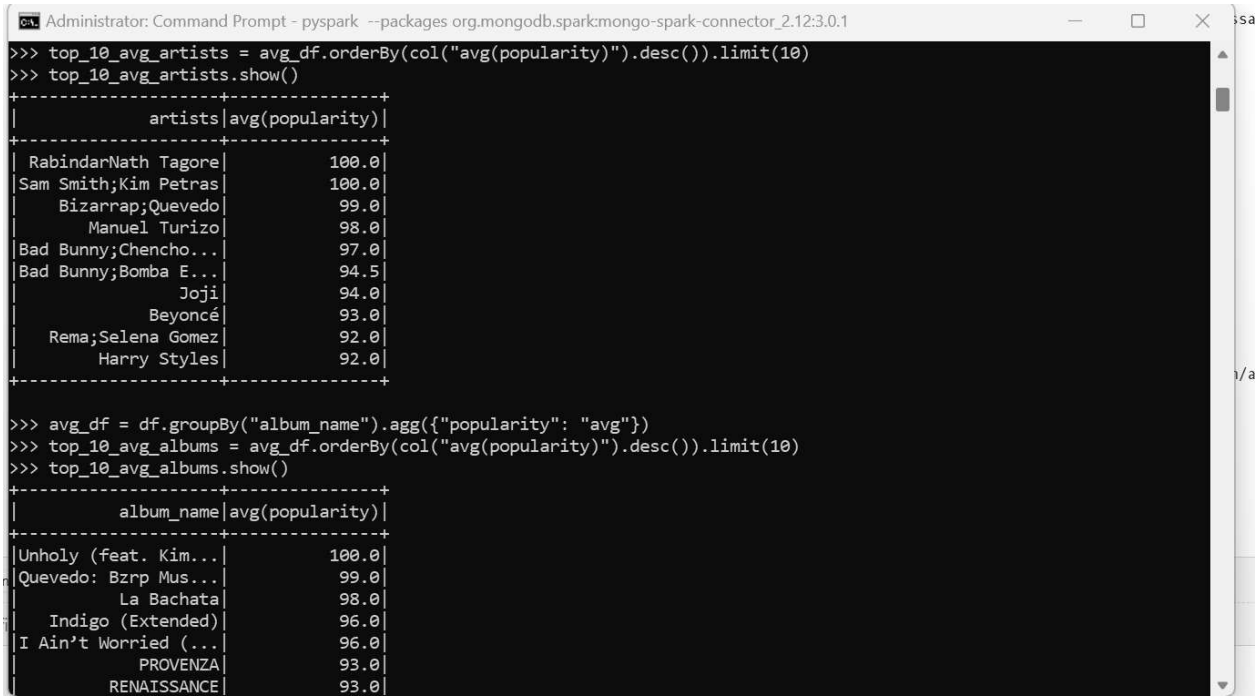
df.show()

from pyspark.sql.functions import col

avg_df = df.groupBy("album_name").agg({"popularity": "avg"}) top_10_avg_albums = avg_df.orderBy(col("avg(popularity)").desc()).limit(10) top_10_avg_albums.show()

avg_df = df.groupBy("artists").agg({"popularity": "avg"}) top_10_avg_artists = avg_df.orderBy(col("avg(popularity)").desc()).limit(10) top_10_avg_artists.show()

avg_df = df.groupBy("track_genre").agg({"popularity": "avg"}) top_10_avg_genres = avg_df.orderBy(col("avg(popularity)").desc()).limit(10) top_10_avg_genres.show()



Github Link

https://github.com/Bhaskarkurada/BITS_MTech_BDSAssignment_SpotifyRecommendation.git