Problem Overview:  
This project is a data engineering pipeline implemented on Amazon Web Services (AWS) for processing Spotify data. The pipeline involves loading CSV files containing information about artists, tracks, and albums into an S3 bucket, performing ETL (Extract, Transform, Load) using AWS Glue, storing the processed data as Parquet files, and finally querying and visualizing the data using Amazon Athena and Power BI.

Dataset:  
  
We have used the data from Spotify which is available in online resource. We used three CSV files named “Tracks”, “Albums”, “Artists”.  
  
Services Used:  
  
**AWS S3 Bucket**: We used this service to store the data

**AWS Glue:** To perform ETL

1. **ETL Script**: Performs inner joins on artist, track, and album data.
2. **Target Data Format**: Parquet files.
3. **IAM Role**: Ensure that the Glue ETL job has appropriate permissions to access S3 and write data to S3.

**AWS Crawler:** It is used to convert the raw data into a meaningful format.

Here we have used the Glue crawler to infer the transformed parquet file which is stored in S3 bucket and create a metadata tables.

**AWS Athena:** To write SQL query the transformed and stored Parquet files.

**Work Process:**

**S3 Bucket:**

Step1: We have created a bucket in S3 where we have created folder named rawdata, so the folder structure was like

Spotifydb/rawdata/albums/albums.csv

Spotifydb/rawdata/artists/artists.csv

Spotifydb/rawdata/Tracks/tracks.csv

**IAM Role:**

Step1: Create a IAM role with access to the following.

Step2: These are those:  
 AmazonAthenaFullaccess

AmazonS3Fullaccess

AWSGlueServiceRole.

Step3: Name it as **Glue\_S3\_Athena\_Role\_FreeTier.**

**AWS GLUE:**

Step2: Creating glue data catalog

1: Here click on the data base option in the left pane of the to create new one.

2: We named it as spotifydb.

Step3: Creating a crawler.

1:Name it as Spotify crawler.

2: Chosen data source as S3 and path as “s3://spotifydatapi/raw-data/”.

3: For IAM role we have given the role as **Glue\_S3\_Athena\_Role\_FreeTier**

4: To output we have specified the location as **spotify\_db.**

**ETL JOB:**

Step1: Create a ETL JOB as spotifyetl

Step2: Then we have used the IAM role as Glue-ETL-Role.

Step3: In the scripts tab give the transformation code which will join all three dataset based on the condition .

Transformation code:

*import sys*

*from awsglue.transforms import \**

*from awsglue.utils import getResolvedOptions*

*from pyspark.context import SparkContext*

*from awsglue.context import GlueContext*

*from awsglue.job import Job*

*from pyspark.sql.functions import col*

*# Initialize Glue and Spark Contexts*

*args = getResolvedOptions(sys.argv, ['JOB\_NAME'])*

*sc = SparkContext()*

*glueContext = GlueContext(sc)*

*spark = glueContext.spark\_session*

*job = Job(glueContext)*

*job.init(args['JOB\_NAME'], args)*

*# Read Data from Glue Catalog*

*artists\_dynamic\_df = glueContext.create\_dynamic\_frame.from\_catalog(database="spotify\_db", table\_name="artists")*

*tracks\_dynamic\_df = glueContext.create\_dynamic\_frame.from\_catalog(database="spotify\_db", table\_name="tracks")*

*albums\_dynamic\_df = glueContext.create\_dynamic\_frame.from\_catalog(database="spotify\_db", table\_name="albums")*

*# Convert DynamicFrames to DataFrames*

*artists\_df = artists\_dynamic\_df.toDF()*

*tracks\_df = tracks\_dynamic\_df.toDF()*

*albums\_df = albums\_dynamic\_df.toDF()*

*# Perform JOINs*

*result\_df = (*

*tracks\_df.alias("t")*

*.join(albums\_df.alias("al"), col("t.id") == col("al.track\_id"), "inner") # Join Tracks with Albums*

*.join(artists\_df.alias("a"), col("al.artist\_id") == col("a.id"), "inner") # Join Albums with Artists*

*.select(*

*col("t.id").alias("Track\_ID"),*

*col("al.track\_name").alias("Track\_Name"),*

*col("a.name").alias("Artist\_Name"),*

*col("al.album\_name").alias("Album\_Name"),*

*col("t.track\_popularity").alias("Track\_Popularity"),*

*col("a.artist\_popularity").alias("Artist\_Popularity"),*

*col("al.album\_popularity").alias("Album\_Popularity")*

*)*

*)*

*# Save Transformed Data as Parquet in S3*

*output\_path = "s3://spotify-data-pipeline/processed-data/"*

*result\_df.write.mode("overwrite").parquet(output\_path)*

*# Complete the Job*

*job.commit()*

Step4: The transformed will be stored as parquet file in the S3 bucket.

AWS Athena:

Step1: Here to query the transformed data we have created 3 external tables

Named: Albums, Artists and Tracks.

Step2: While creating this external table we have given the location for the raw data pointing to the data source in S3 , so the location of each table will point to the albums , artists, tracks.

Step3: After that we have queried the transformed data to get insights from the transformed data using pipeline.

Step4: Then we have downloaded the result to as CSV file to visualize the data.