

# Music Insights

TEAM CLOUD CODERS



# Team Members

- Megha Singha Roy
- Pavan Shetti
- Sanketh Shinde
- Thrishala NP
- Sudeep S

# Project Overview

- Builds a distributed ETL pipeline to process users music listening data.
- Uses AWS Glue to clean and transform raw data.
- Final metrics are queried through Athena for insights.

# Use Cases

## **1. Measure User Engagement:**

- Measure total listens, users, and duration.

## **2. Top Songs**

- Find top 3 songs in each genre daily.

## **3. Top Genres**

- Identify 5 most popular genres overall.

# Input Data Overview

- Dataset source - S3
- Data format - csv
- Sample schemas

## Users schema

```
root
|-- user_id: integer (nullable = true)
|-- user_name: string (nullable = true)
|-- user_age: integer (nullable = true)
|-- user_country: string (nullable = true)
|-- created_at: timestamp (nullable = true)
```

# Input Data Overview

## Songs Schema

```
root
|-- id: integer (nullable = true)
|-- track_id: string (nullable = true)
|-- artists: string (nullable = true)
|-- album_name: string (nullable = true)
|-- track_name: string (nullable = true)
|-- popularity: integer (nullable = true)
|-- duration_ms: integer (nullable = true)
|-- explicit: boolean (nullable = true)
|-- danceability: double (nullable = true)
|-- energy: double (nullable = true)
|-- key: integer (nullable = true)
|-- loudness: double (nullable = true)
|-- mode: integer (nullable = true)
|-- speechiness: double (nullable = true)
|-- acousticness: double (nullable = true)
|-- instrumentalness: double (nullable = true)
|-- liveness: double (nullable = true)
|-- valence: double (nullable = true)
|-- tempo: double (nullable = true)
|-- time_signature: integer (nullable = true)
|-- track_genre: string (nullable = true)
```

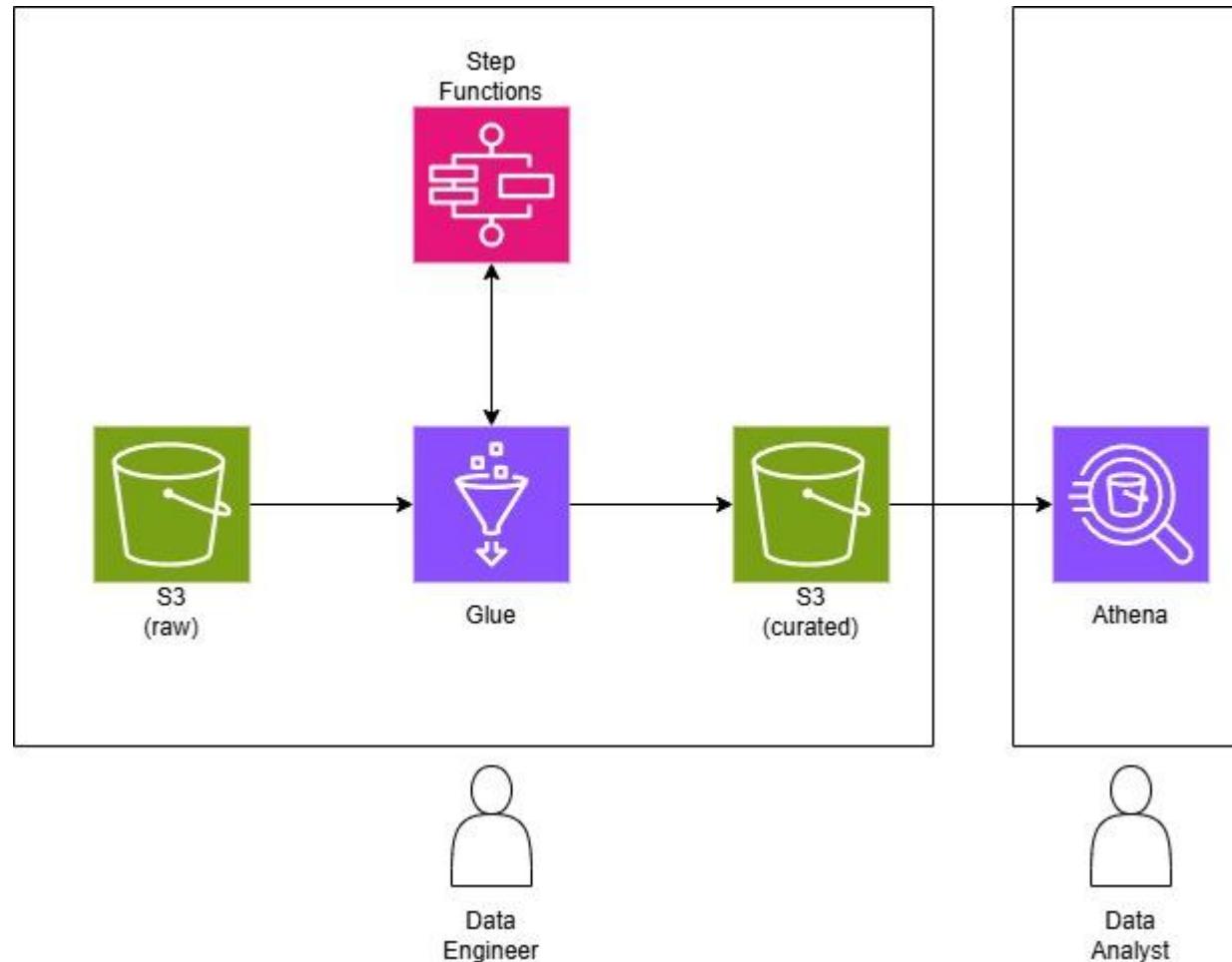
# Input Data Overview

## Streams Schema

```
root
|-- user_id: integer (nullable = true)
|-- track_id: string (nullable = true)
|-- listen_time: timestamp (nullable = true)
```

# High-Level Architecture

- AWS services used: S3, Glue, Athena, Step Functions



# Snapshot

Loading data into S3

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, a search bar containing 'Search', a keyboard shortcut '[Alt+S]', and various account and region settings including 'Account ID: 2171-5298-5977', 'United States (N. Virginia)', and 'esubbangloreaws08'. Below the navigation bar, the breadcrumb path indicates the user is in the 'user/' folder under the 'mui-input-dataset' bucket. On the left, there are tabs for 'Objects' (which is selected) and 'Properties'. A prominent orange 'Upload' button is located at the top right of the main content area. The main content area displays a table of objects. The table has columns for 'Name', 'Type', 'Last modified', 'Size', and 'Storage class'. One object, 'users.csv', is listed: it is a CSV file last modified on November 1, 2025, at 09:46:52 (UTC+05:30), is 2.3 MB in size, and is stored in the 'Standard' storage class. There are also buttons for 'Actions' (with a dropdown arrow), 'Create folder', and 'Copy S3 URI'.

Name	Type	Last modified	Size	Storage class
users.csv	CSV	November 1, 2025, 09:46:52 (UTC+05:30)	2.3 MB	Standard

# Snapshot

## Loading data into S3

The screenshot shows the AWS S3 console interface. At the top, the AWS logo, a search bar with placeholder text 'Search', and a keyboard shortcut '[Alt+S]' are visible. To the right, there are icons for account information ('Account ID: 2171-5298-5977'), notifications, help, and settings. The region is set to 'United States (N. Virginia)'.

The navigation path is 'Amazon S3 > Buckets > mui-input-dataset > streams/'. On the far right of the header, there are refresh and refresh-with-history icons.

The main content area shows a folder named 'streams/'. A blue button labeled 'Copy S3 URI' is located on the right side of the folder name. Below it, there are tabs for 'Objects' (which is selected) and 'Properties'.

The 'Objects' section has a heading 'Objects (2)'. It includes a refresh icon, buttons for 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', 'Create folder', and an orange 'Upload' button. A note below the heading states: 'Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)'.

Below this, there is a search bar with placeholder text 'Find objects by prefix' and a page navigation area with arrows and a gear icon. A table lists the two objects:

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	<a href="#">streams1.csv</a>	csv	November 1, 2025, 09:47:14 (UTC+05:30)	551.5 KB	Standard
<input type="checkbox"/>	<a href="#">streams2.csv</a>	csv	November 1, 2025, 09:47:13 (UTC+05:30)	551.6 KB	Standard

# Snapshot

Loading data into S3

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and account information (Account ID: 2171-5298-5977, United States (N. Virginia)). Below the navigation bar, the breadcrumb path indicates the location: Amazon S3 > Buckets > mui-input-dataset > songs/. The main area displays a folder named "songs/" with a "Copy S3 URI" button. There are two tabs: "Objects" (selected) and "Properties". Under the "Objects" tab, there's a summary section for "Objects (1)". It includes a "Copy S3 URI" button, a "Copy URL" button, a "Download" button, an "Open" button, a "Delete" button, an "Actions" dropdown, a "Create folder" button, and an "Upload" button. A note below states: "Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)". Below this, there's a search bar labeled "Find objects by prefix" and a table listing the object "songs.csv". The table columns are: Name, Type, Last modified, Size, and Storage class. The object details are: Name is "songs.csv", Type is "csv", Last modified is "November 1, 2025, 09:49:25 (UTC+05:30)", Size is "15.2 MB", and Storage class is "Standard".

Name	Type	Last modified	Size	Storage class
songs.csv	csv	November 1, 2025, 09:49:25 (UTC+05:30)	15.2 MB	Standard

# Snapshot

AWS Glue

## Script Info

```
1 from pyspark.sql import SparkSession
2 spark = SparkSession.builder.getOrCreate()
3
4 usersDf = spark.read.csv('s3://mui-input/songs/', header=True, inferSchema = True)
5 songsDf = spark.read.csv('s3://mui-input/songs/', header=True, inferSchema = True)
6 streamsDf = spark.read.csv('s3://mui-inputstreams/', header=True, inferSchema = True)
7
```

# Snapshot

AWS Glue

## Script Info

```
12▼ def transform_data(spark, usersDf, songsDf, streamsDf):
13    """drop/fill nulls, typecasting, remove duplicates"""
14
15    #Convert listen_time to date.
16    streamsDf = streamsDf.withColumn("report_date", to_date(streamsDf.listen_time))
17
18
19    #filter invalid keys
20    streamsDf = streamsDf.dropna(subset=['user_id', 'track_id', 'listen_time'])
21
22    #type casting of duration to long
23    songsDf = songsDf.withColumn("duration_ms", col("duration_ms").cast("long"))
24
25    return usersDf, songsDf, streamsDf
26
27 usersDf, songsDf, streamsDf = transform_data(spark, usersDf, songsDf, streamsDf)
28
```

# Snapshot

## AWS Glue

Script Info

```
29
30▼ def load_data(spark, usersDf, songsDf, streamsDf):
31    db_name = "music_insights"
32    s3_output_path = "s3://mui-output/"
33
34    #Creating database
35    spark.sql(f"CREATE DATABASE IF NOT EXISTS {db_name} LOCATION '{s3_output_path}';")
36
37    #Save dataframe as glue table
38▼ def save_as_table(df, table_name):
39    table_path = f"{s3_output_path}{table_name}/"
40    full_name = f"{db_name}.{table_name}"
41
42    df.write.format("parquet").mode("overwrite") \
43        .option("path", table_path) \
44        .saveAsTable(full_name)
45    print(f"Created table: {full_name}")
46
47    save_as_table(songsDf, "songs")
48    save_as_table(usersDf, "users")
49    save_as_table(streamsDf, "streams")
50
51 load_data(spark, usersDf, songsDf, streamsDf)
52
```

Python Ln 42, Col 35 ✘ Errors: 0 | ⚠ Warnings: 0

# Snapshot

## Step Functions

Screenshot of the AWS Step Functions console showing a successfully created state machine.

**Header:** AWS logo, Search bar, [Alt+S] keyboard shortcut, Notifications (0), United States (N. Virginia) region, Account ID: 8076-5071-7571, ebsubbangaloreaws06.

- Dashboard
- State machines
- Activities
- Developer resources
  - Execution inspector
  - Online learning workshop
  - Local Development
  - Data flow simulator
  - Feature spotlight
  - Documentation
- Join our feedback panel

State machine successfully created (green bar).

Standard

```
1 {
  "Comment": "mui-demo",
  "StartAt": "RunETLJob",
  "States": [
    "RunETLJob": {
      "Type": "Task",
      "Resource": "arn:aws:states:::glue:startJobRun.sync",
      "Parameters": {
        "JobName": "mui-demo"
      },
      "End": true
    }
  ]
}
```

```
graph TD
    Start((Start)) --> RunETLJob[Glue: StartJobRun  
RunETLJob]
    RunETLJob --> End((End))
```

Exit, Actions (dropdown), Execute (button), Save (button).

CloudShell, Feedback, © 2025, Amazon Web Services, Inc. or its affiliates., Privacy, Terms, Cookie preferences.

# Snapshot

## Track metrics

```
Track Metrics : X | Top 5 Genres Overall : X | Top 3 songs : X ( + ) ▾  
1 -- Track Metrics - Total Listens, Unique Users, Duration, Average Time per User  
2 SELECT  
3     s.track_id,  
4     st.report_date,  
5     COUNT(*) AS total_listens,  
6     COUNT(DISTINCT st.user_id) AS unique_users,  
7     SUM(s.duration_ms) AS total_listening_duration,  
8     ROUND(SUM(s.duration_ms) / COUNT(DISTINCT st.user_id), 2) AS avg_listening_time_per_user  
9 FROM streams AS st  
10 LEFT JOIN songs AS s  
11     ON st.track_id = s.track_id  
12 GROUP BY s.track_id, st.report_date  
13 ORDER BY st.report_date, total_listens DESC  
14 LIMIT 10  
15 ;  
SQL Ln 9, Col 19
```

# Snapshot

## Result of Track Metrics

Query results    Query stats

🕒 Completed    Time in queue: 102 ms    Run time: 976 ms    Data scanned: 3.12 MB

Results (10)    [Copy](#)    [Download results CSV](#)

🔍 Search rows

< 1 > | ⚙️

#	track_id	report_date	total_listens	unique_users	total_listening_duration	avg_listening_time_per_user
1	OBtD9k8XjGliQJMS2hxwNx	2024-06-25	5	5	1377705	275541
2	1kMObCQiYe5opqybH7ZNPD	2024-06-25	4	4	956332	239083
3	4686EWn7tO5A7ABm8DgY0w	2024-06-25	4	4	582544	145636
4	19coiw9dDWBNHtQhdKL1VC	2024-06-25	4	4	640152	160038
5	2u1EtHkbpeRjkxCywhXEYp	2024-06-25	4	4	1329396	332349
6	3a2jFwnts4Cf0OwJbK61SL	2024-06-25	4	4	840108	210027
7	2vdNa2nf6ptRqSiUgGjxLS	2024-06-25	4	4	1799252	449813
8	0k5uPly8GLdwF16eilwzml	2024-06-25	4	4	930024	232506
9	7HcM82GiytrUB4YrkQwY6Y	2024-06-25	4	4	489012	122253
10	3cg38isdTrBH63B4BMywsw	2024-06-25	4	4	1099292	274823

# Snapshot

## Top 3 songs

Track\_Metrics : X | Top 5 Genres Overall : X | **Top 3 songs : X** ( + ) ▾

```
1 SELECT * FROM (
2     SELECT
3         s.track_genre, st.report_date, s.track_id, s.artists,
4         COUNT(*) AS total_listens,
5         DENSE_RANK() OVER (
6             PARTITION BY s.track_genre, st.report_date
7             ORDER BY COUNT(*) DESC
8         ) AS song_rank
9     FROM streams AS st
10    LEFT JOIN songs AS s
11        ON st.track_id = s.track_id
12    GROUP BY s.track_genre, st.report_date, s.track_id, s.artists
13 )
14 WHERE song_rank <= 3
15 ORDER BY report_date, track_genre, song_rank LIMIT 10;
```

SQL Ln 9, Col 23

Run again Explain ↗ Cancel Clear Create ▾ Reuse query results up to 60 minutes ago 🖊

# Snapshot

## Result of Top 3 Songs

Query results      Query stats

Completed      Time in queue: 134 ms      Run time: 970 ms      Data scanned: 3.22 MB

Results (10)      [Copy](#)      [Download results CSV](#)

Search rows

#	track_genre	report_date	track_id	artists	total_listens	song_rank
1	acoustic	2024-06-25	3CqV5uSdPvj5Ux3E8Idusc	Eddie van der Meer	3	1
2	acoustic	2024-06-25	5Hnfk1EuhqQuh7QhcniR1	Masaharu Fukuyama	2	2
3	acoustic	2024-06-25	5MYPzdlWgx3pMLRGlq2fVq	Zack Tabudlo	2	2
4	acoustic	2024-06-25	685WYfzmOeDOF4laU8UuOv	Lee DeWyze	2	2
5	acoustic	2024-06-25	74ooA3upXyi3AUg9Ndz2Ck	Frank Turner	2	2
6	acoustic	2024-06-25	3RE5nHVkC4KcA7snFaTKd	Brandi Carlile;Lucius	2	2
7	acoustic	2024-06-25	3lvo0l8c2qBJDLZWqExDgC	Brandi Carlile	2	2
8	acoustic	2024-06-25	0U32q8CZRRo7xCzyiaZw5f	Motohiro Hata	2	2
9	acoustic	2024-06-25	7bhHLZxkRekrNPPkEdDTbn	Ben Woodward	2	2
10	acoustic	2024-06-25	2AHZHeLTPuGILKyr4l8uTU	Joe Brooks	2	2

# Snapshot

## Top 5 Genres

Track\_Metrics : X   Top 5 Genres Overall : X   Top 3 songs : X   ( + ) ▾

```
1 SELECT * FROM (
2     SELECT
3         s.track_genre, COUNT(*) AS total_listens_overall,
4         SUM(s.duration_ms) AS total_listening_duration_overall,
5         COUNT(DISTINCT s.track_id) AS unique_tracks,
6         DENSE_RANK() OVER (ORDER BY COUNT(*) DESC) AS rank
7     FROM streams AS st
8     LEFT JOIN songs AS s
9         ON st.track_id = s.track_id
10    WHERE s.track_genre IS NOT NULL
11    GROUP BY s.track_genre
12 ) AS ranked_genres
13 WHERE rank <= 5
14 ORDER BY rank;
15
```

SQL Ln 11, Col 26

Run Explain ↗ Cancel Clear Create ▾

Reuse query results  
up to 60 minutes ago ⌚

# Snapshot

## Result of Top 5 Genres

Query results    Query stats

Completed    Time in queue: 151 ms    Run time: 1.112 sec    Data scanned: 3.03 MB

Results (9)    [Copy](#)    [Download results CSV](#)

Search rows

#	track_genre	total_listens_overall	total_listening_duration_overall	unique_tracks	rank
1	anime	299	62298754	251	1
2	dance	280	54517297	239	2
3	children	272	37504339	235	3
4	disney	272	43107643	242	3
5	country	268	55357250	224	4
6	trip-hop	268	73139853	226	4
7	cantopop	267	60203416	235	5
8	chicago-house	267	100574211	228	5
9	happy	267	65391574	237	5

# Analytical Insights

- Identified Top Songs per Genre (daily) to track music popularity.
- Highlighted Top Genres overall, showing user listening trends.
- Calculated user engagement metrics - total listens, unique users, and listening duration.

# Future Scope

1. Real-Time Data Processing
2. Automated Data Ingestion and Crawling
3. Integrate Boto3 SDK



**THANK YOU!**