

# 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)
```

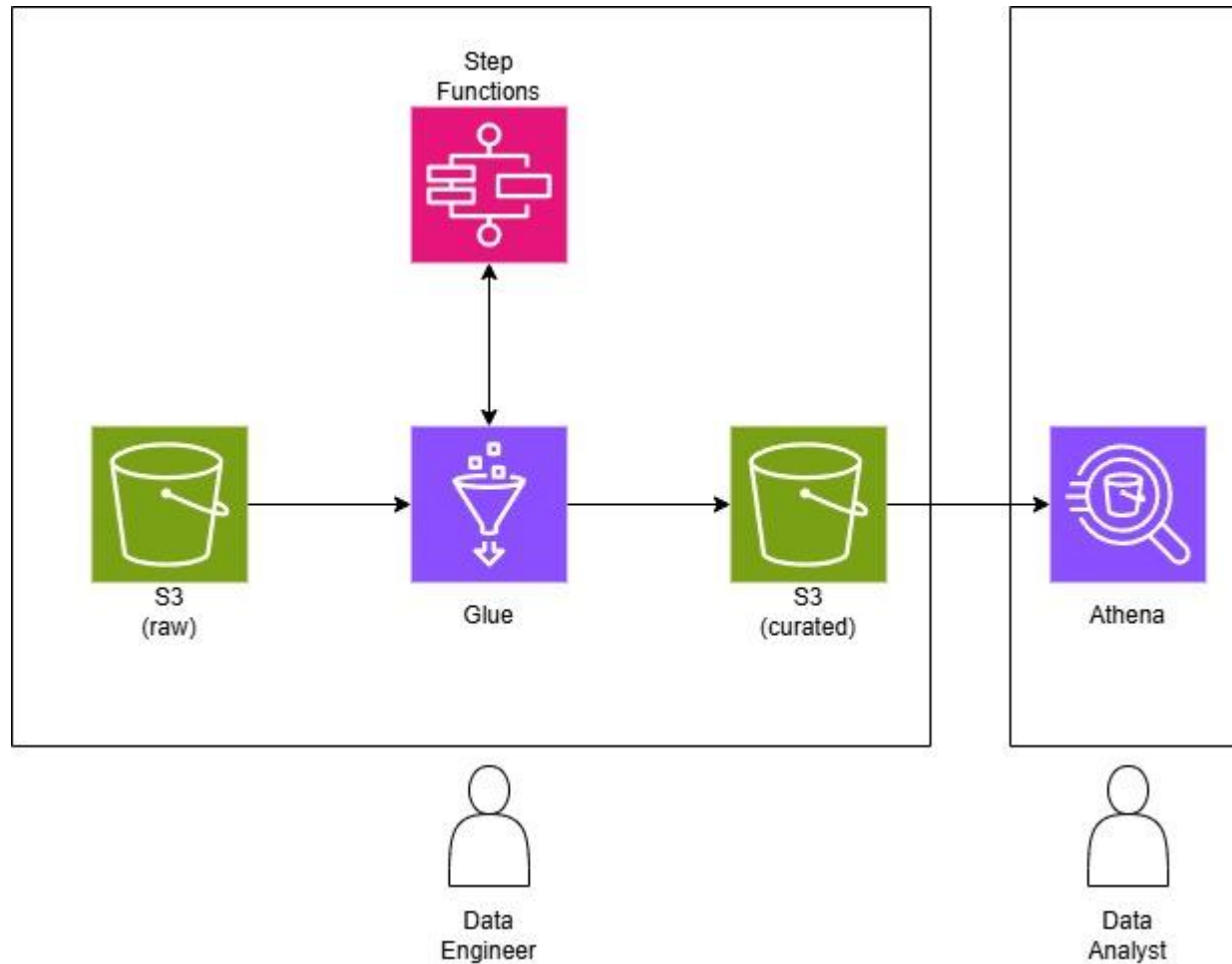
# Input Data Overview

## Streams Schema

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

# High-Level Architecture







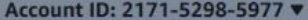
- AWS services used: S3, Glue, Athena, Step Functions








# Snapshot

Loading data into S3


   [Alt+S]     United States (N. Virginia)  ebsubbangloreaws08

 [Amazon S3](#) > [Buckets](#) > [mui-input-dataset](#) > [user/](#)  


## user/


[Copy S3 URI](#)

[Objects](#) [Properties](#)

**Objects (1)**  [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open ↗](#) [Delete](#) [Actions ▼](#) [Create folder](#) [Upload](#)







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](#)



 < 1 > 

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 <a href="#">users.csv</a>	csv	November 1, 2025, 09:46:52 (UTC+05:30)	2.3 MB	Standard

# Snapshot


Loading data into S3

   [Alt+S]     United States (N. Virginia) Account ID: 2171-5298-5977  
ebsubbangaloreaws08


[Amazon S3](#) > [Buckets](#) > [mui-input-dataset](#) > streams/  



streams/ [Copy S3 URI](#)

[Objects](#) [Properties](#)

**Objects (2)**  [Copy S3 URI](#) [Copy URL](#) [Download](#) [Open](#) [Delete](#) [Actions](#) [Create folder](#) [Upload](#)







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](#)

 < 1 > 

<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

 [Alt+S] United States (N. Virginia) ▼ Account ID: 2171-5298-5977 ▼ ebsubbangaloreaws08

[Amazon S3](#) > [Buckets](#) > [mui-input-dataset](#) > songs/


## songs/

[Copy S3 URI](#)


Objects

Properties

Objects (1)

[Copy S3 URI](#)[Copy URL](#)[Download](#)[Open](#)[Delete](#)[Actions](#)[Create folder](#)[Upload](#)

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](#)

<input type="checkbox"/>	Name	Type	Last modified	Size	Storage class
<input type="checkbox"/>	 <a href="#">songs.csv</a>	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-input/streams/', 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

The screenshot displays the AWS Step Functions console interface. At the top, a green notification bar states "State machine successfully created". Below this, the "StepFunction" configuration is shown in the "Code" view. The JSON configuration defines a state machine named "mui-demo" with a single task state "RunETLJob" that uses the "Glue: StartJobRun" resource. The right-hand side of the console shows a visual state machine diagram with a "Start" node, a task node labeled "Glue: StartJobRun RunETLJob", and an "End" node. The left sidebar contains navigation links for "Step Functions", "Developer resources", and "Join our feedback panel". The top navigation bar includes the AWS logo, search bar, account ID, and region information.

**Step Functions**

Dashboard  
State machines  
Activities

▼ **Developer resources**

- Execution inspector
- Online learning workshop
- Local Development
- Data flow simulator
- Feature spotlight
- Documentation

Join our feedback panel

**StepFunction** Standard Design Code Config Exit Actions Execute Save

Undo Redo Format Copy Commands Zoom in Zoom out Center Feedback

```
1 {  
2   "Comment": "mui-demo",  
3   "StartAt": "RunETLJob",  
4   "States": {  
5     "RunETLJob": {  
6       "Type": "Task",  
7       "Resource": "arn:aws:states:::glue:startJobRun.sync",  
8       "Parameters": {  
9         "JobName": "mui-demo"  
10      },  
11     "End": true  
12   }  
13 }  
14 }
```

Start

Glue: StartJobRun  
RunETLJob

End

CloudShell Feedback

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



# Snapshot

## Track metrics

✓ Track\_Metrics : ✕ | ✓ Top 5 Genres Overall : ✕ | ✓ Top 3 songs : ✕ | ( + ) ▼

```
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

Q Search rows

< 1 > ⚙

#	track_id	report_date	total_listens	unique_users	total_listening_duration	avg_listening_time_per_user
1	0BtD9k8XjGliQJMS2hxxNx	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	2u1EtHkbpRjKxCywhXEYp	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 : ✕

✓ Top 5 Genres Overall : ✕

✓ Top 3 songs : ✕

( + ) ▼

```
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)		<div>CopyDownload results CSV</div>				
<input type="text" value="Search rows"/>		<div>&lt; 1 &gt; ⚙</div>				
#	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	5Hnfk1EuhqQuh7QhcnikR1	Masaharu Fukuyama	2	2
3	acoustic	2024-06-25	5MYPzdIWgx3pMLRGlq2fVq	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	3RE5nHIVkC4KcA7snFaTKd	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

Top 5 Genres Overall

Top 3 songs

```
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)		<a href="#">Copy</a>		<a href="#">Download results CSV</a>	
<input type="text" value="Search rows"/>		< 1 >		⚙	
#	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!**