School of Information Studies
Syracuse University

IST 615 - Cloud Management

Final Project Report

Professor in-charge: Carlos E. Caicedo Bastidas

<u>Group Members</u>

Names:

- Rahul Jadhav SUID: 705889151
- Ramazan Yener SUID: 4604870161
- Francia Ortiz Leyva SUID: 2566771622
- Himamshu Chandrashekara SUID: 254732602

A) <u>Project Goal:</u>

The project focuses on creating an Analytics platform in which we would ingest, store, transform and consume data using AWS analytics cloud services such as:

1. *Amazon S3*
2. *Amazon Glue*
3. *Amazon Athena*
4. *Amazon QuickSight*
5. *Amazon Redshift*

1. **AWS Simple Storage Service (AWS S3):**

Organizations need to collect, store, and analyze data that they must manage. Since these data are on a huge scale, it has been a problem for organizations in terms of storage capacity. Establishing and maintaining your own storage costs a lot, besides it takes extra time to build and manage, as well as organizations might even need to hire more staff to do this. Amazon Simple Storage Service, also known as Amazon S3, provides organizations solutions to deal with storage issues. Amazon S3 is an object storage service that helps companies for scalability, availability, security, as well as performance. Therefore, companies of all sizes can utilize Amazon Simple Storage Services. Simplicity is one of the key features of Amazon S3 that provides users with user-friendly interfaces so that they can store and retrieve their data at any time and anywhere they want. In this service, customers only pay for the storage they use with no minimum fee or set up costs. Since there is no standard payment for this service, customers can start with a small amount of data, after that they can scale the usage and decide if they want to extend their data storage without lowering performance or reliability, so flexibility is another advantageous feature of the service. The service is used for many different purposes such as archives, storage for applications, backup, recovery for disaster, data lakes etc. Amazon S3 offers different types of storage classes based on customers' expectations like durability, availability, and performance requirements. These are S3 Intelligent-Tiering, S3 Standard, S3 Standard-Infrequent Access (S3 Standard-IA), S3 One Zone-Infrequent Access (S3 One Zone-IA), S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive.

2. **<u>AWS Glue:</u>**

Amazon Glue

Managing data is a complicated process that needs to be spent time to transferring data from a repository to another, to extract and load these data to analysis, so Amazon Glue is a cloud optimized extract, transform, and load service (ETL), this makes it simpler and less cost to classify data, cleaning and moving data. It helps users to organize, locate, move, and transform all their data across business so that customers can put data to utilize. Amazon Glue is a serverless ETL service that provides developers with freedom of use without considering server-based, infrastructure issues like configure, spin up or manage their Lifecycle. AWS Glue provides all features for data integration, so customers can start analyzing data they have, and they can spend less time on this process.
AWS Glue helps to transform data from one format to another. AWS Glue will help to understand data, suggest transformations and create ETL codes and Python, so the users spend less time coding. If users want to modify this code, they can modify codes using an IDE or notebook. AWS Glue has both visual and code-based interfaces so that users can make data integration easier. Users can find and access

easily by using AWS Glue Data Catalog. AWS Glue has many benefits, for instance it provides faster data integration when people in a company from separate groups work together for data integration, extraction, cleaning, normalization, combining, loading, and running scalable ETL workflows. AWS Glue helps these groups work together to analyze data and reduces time while they are working together. Besides AWS Glue, automates data integration at scale which helps automates effort required for data integration, it crawls in data sources, defines data formats and offers schemas to store data.

3. <u>Amazon Athena:</u>

Athena quickly analyzes data stored in S3 buckets, using SQL language. It is serverless which facilitates the process of querying the data. There is no need for setting up or managing data warehouses. Another advantage is that organizations only pay for the queries they run, no extra expenses. This saves organizations a lot of money and grants them control to see what they are investing their revenue into. If they need more queries, they can simply run them and pay the cost. Something else that can be controlled, is the access to the data being stored. This can be done through the Amazon Identity and Access Management (IAM) policies, access control lists and the Amazon S3 bucket policies. Restricting other users to run queries, provides a sense of security within the organization.

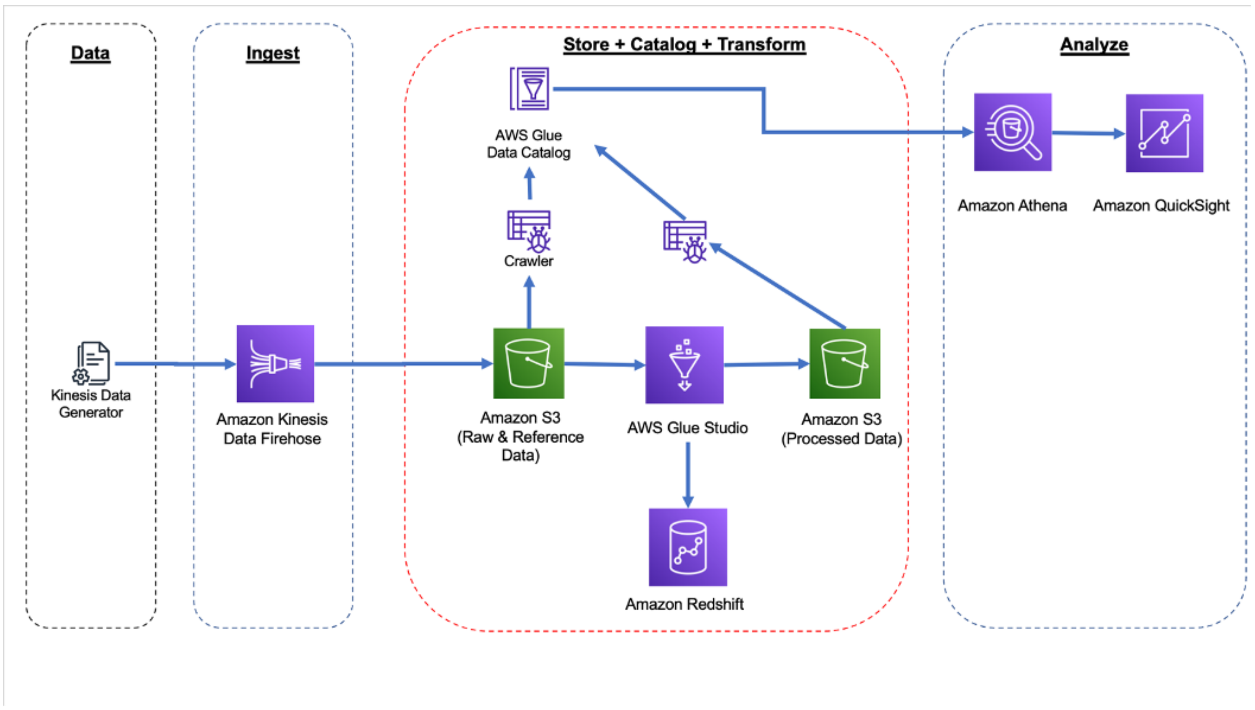
4. **<u>Amazon QuickSight:</u>**

In order to work together and make decisions as a group, communication is an absolute necessity. AWS Quicksight offers just this, a business centered intelligence service that helps analyze data as whole. The data is presented in a series of dashboards that can then be embedded with other applications, allowing a flexible and efficient way to share business data. This eliminates the need to manually write code to extract, transform and load operations. QuickSight does it all for you and you can choose what to do with the presented information. Another benefit to this service is its scalability, granting more space when needed. This is important because as an organization grows, they will be dealing with more data from its stakeholders. QuickSight would then be able to store and analyze the new data, fulfilling the businesses needs.

5. **<u>Amazon Redshift:</u>**

Redshift uses SQL language to analyze and store data from operational databases, data lakes hardware, and machine learning. Using the most advanced technology, Massively Parallel Processing (MPP) performs operations on large data sets at a rapid speed. Other than speed, some of the other benefits this brings is the ability to have historical and current data. If someone in the organization needs to obtain a certain data they can access it in little to no time. This information is then used to make decisions that require quick solutions. The data in RedShift is also encrypted, providing a sense of security within the organization.

B) Cloud Architecture



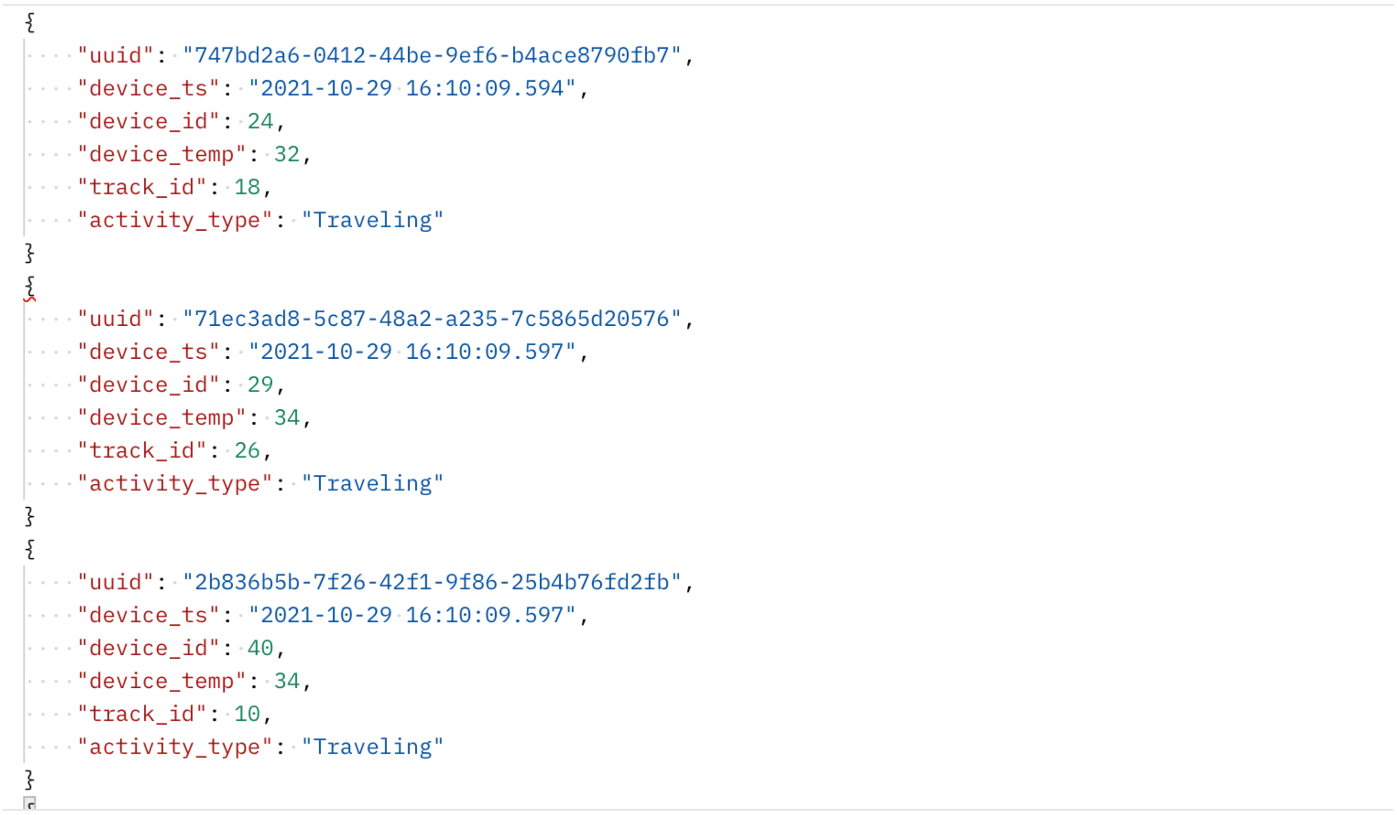
C) Execution Process:

1) Ingest & store

1.1 Data Generation

Data is being generated using Kinesis Data tool Generator. Here we would be leveraging Cloud formation Stack ready to use template provided by AWS to generate data which can be used to ingest in the next process.

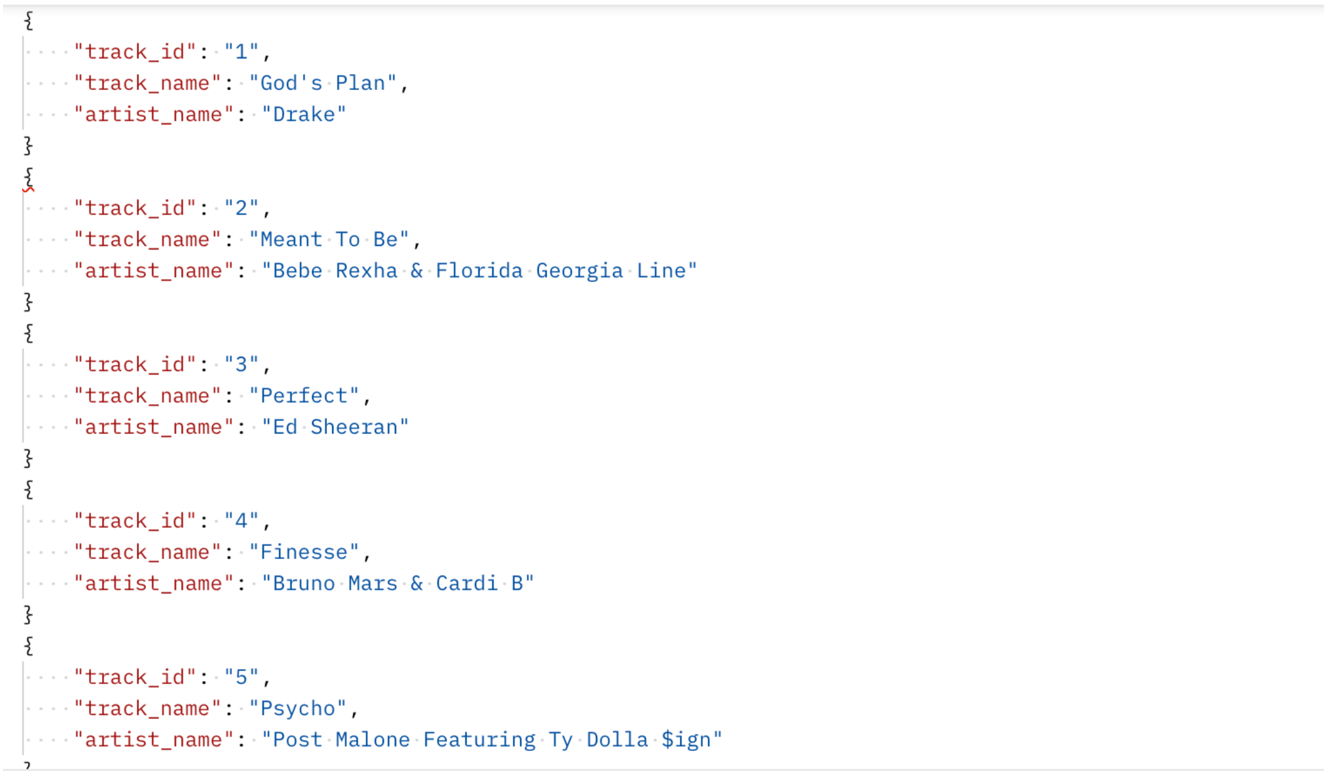
Screenshot of Raw json Data



```
{
    "uuid": "747bd2a6-0412-44be-9ef6-b4ace8790fb7",
    "device_ts": "2021-10-29 16:10:09.594",
    "device_id": 24,
    "device_temp": 32,
    "track_id": 18,
    "activity_type": "Traveling"
}
{
    "uuid": "71ec3ad8-5c87-48a2-a235-7c5865d20576",
    "device_ts": "2021-10-29 16:10:09.597",
    "device_id": 29,
    "device_temp": 34,
    "track_id": 26,
    "activity_type": "Traveling"
}
{
    "uuid": "2b836b5b-7f26-42f1-9f86-25b4b76fd2fb",
    "device_ts": "2021-10-29 16:10:09.597",
    "device_id": 40,
    "device_temp": 34,
    "track_id": 10,
    "activity_type": "Traveling"
}
```

Description: The above screenshot shows the raw JSON structure of songs with it’s activity_type

Screenshot of Reference json Data



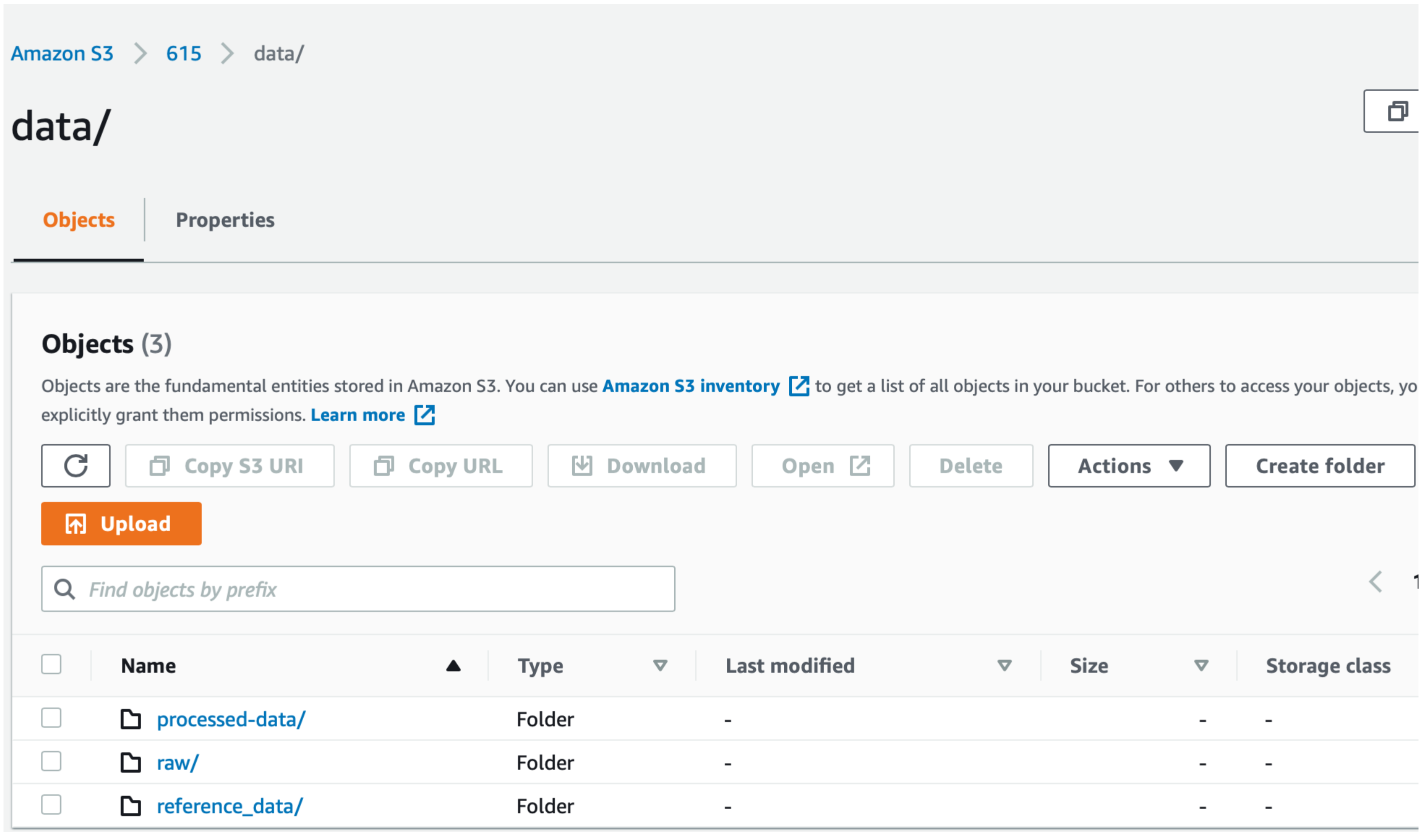
```
{
    "track_id": "1",
    "track_name": "God's Plan",
    "artist_name": "Drake"
}
{
    "track_id": "2",
    "track_name": "Meant To Be",
    "artist_name": "Bebe Rexha & Florida Georgia Line"
}
{
    "track_id": "3",
    "track_name": "Perfect",
    "artist_name": "Ed Sheeran"
}
{
    "track_id": "4",
    "track_name": "Finesse",
    "artist_name": "Bruno Mars & Cardi B"
}
{
    "track_id": "5",
    "track_name": "Psycho",
    "artist_name": "Post Malone Featuring Ty Dolla $ign"
```

Description: The above screenshot shows the reference JSON structure of songs details.

1.2. Ingesting & Store

For Ingesting data, we are utilizing Kinesis Data Firehose Cloud Service to stream the incoming raw data from the Kinesis Data tool Generator to the target S3 bucket named 615 into respective data folders named "*raw*" and "*reference_data*" for further storing and data transformation process.

Screenshot of file structure S3 Bucket named 615



Amazon S3 > 615 > data/

data/

Objects | Properties

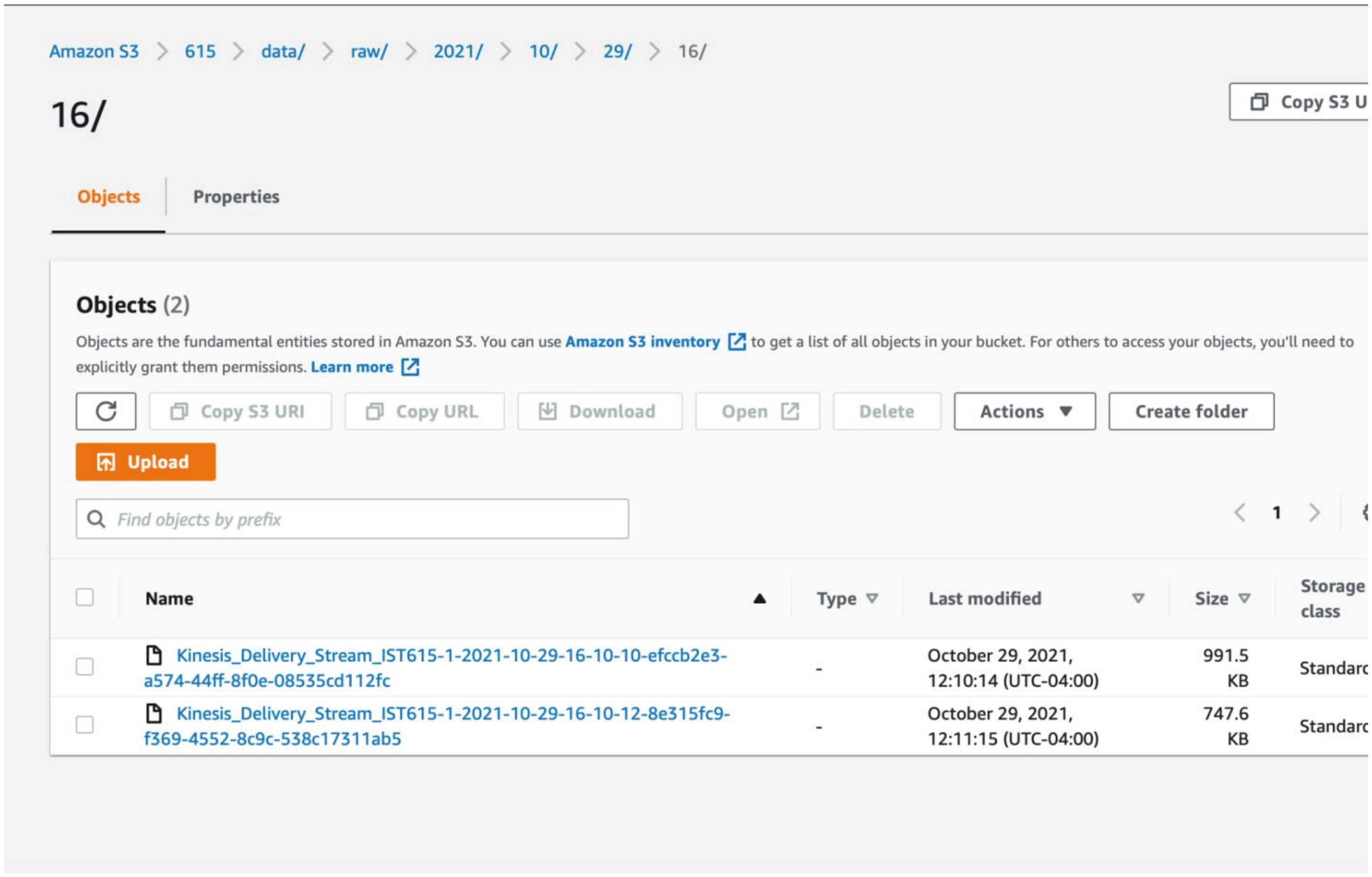
Objects (3)

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, yo explicitly grant them permissions. **Learn more**

	Name	Type	Last modified	Size	Storage class
	processed-data/	Folder	-	-	-
	raw/	Folder	-	-	-
	reference_data/	Folder	-	-	-

Description: The above screenshot shows all the folders we created to store *raw, reference , processed data* in the same S3 Bucket.

Screenshot of raw data stored in S3 bucket under /raw folder

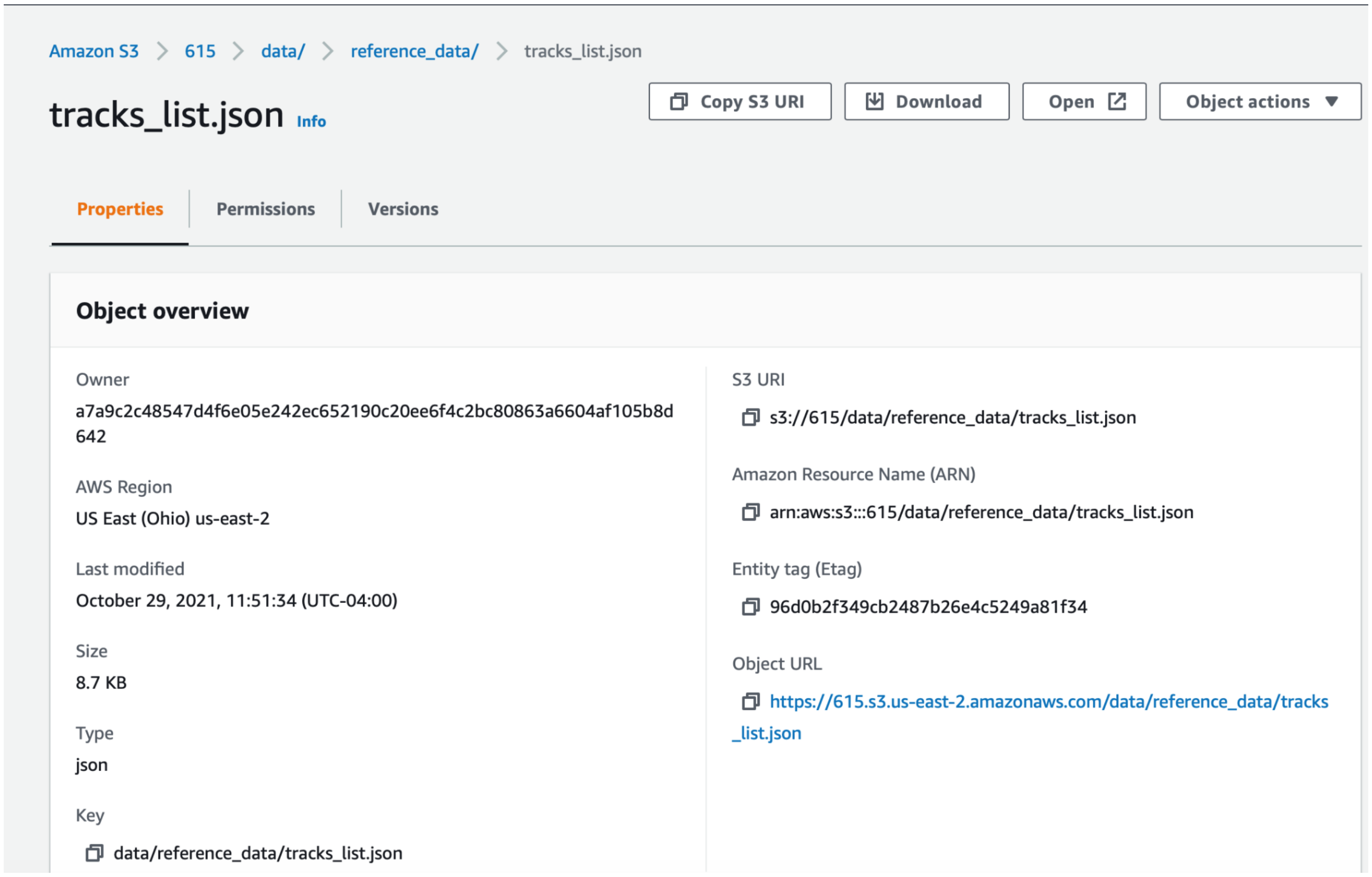


Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
Kinesis_Delivery_Stream_IST615-1-2021-10-29-16-10-10-efccb2e3-a574-44ff-8f0e-08535cd112fc	-	October 29, 2021, 12:10:14 (UTC-04:00)	991.5 KB	Standarc
Kinesis_Delivery_Stream_IST615-1-2021-10-29-16-10-12-8e315fc9-f369-4552-8c9c-538c17311ab5	-	October 29, 2021, 12:11:15 (UTC-04:00)	747.6 KB	Standarc

Description: The above screenshot shows the raw data files generated inside raw folder using kinesis data generator tool.

Screenshot of reference data stored in S3 Bucket under /reference folder



tracks_list.json Info

Object overview

Owner
a7a9c2c48547d4f6e05e242ec652190c20ee6f4c2bc80863a6604af105b8d642

AWS Region
US East (Ohio) us-east-2

Last modified
October 29, 2021, 11:51:34 (UTC-04:00)

Size
8.7 KB

Type
json

Key
data/reference_data/tracks_list.json

S3 URI
s3://615/data/reference_data/tracks_list.json

Amazon Resource Name (ARN)
arn:aws:s3:::615/data/reference_data/tracks_list.json

Entity tag (Etag)
96d0b2f349cb2487b26e4c5249a81f34

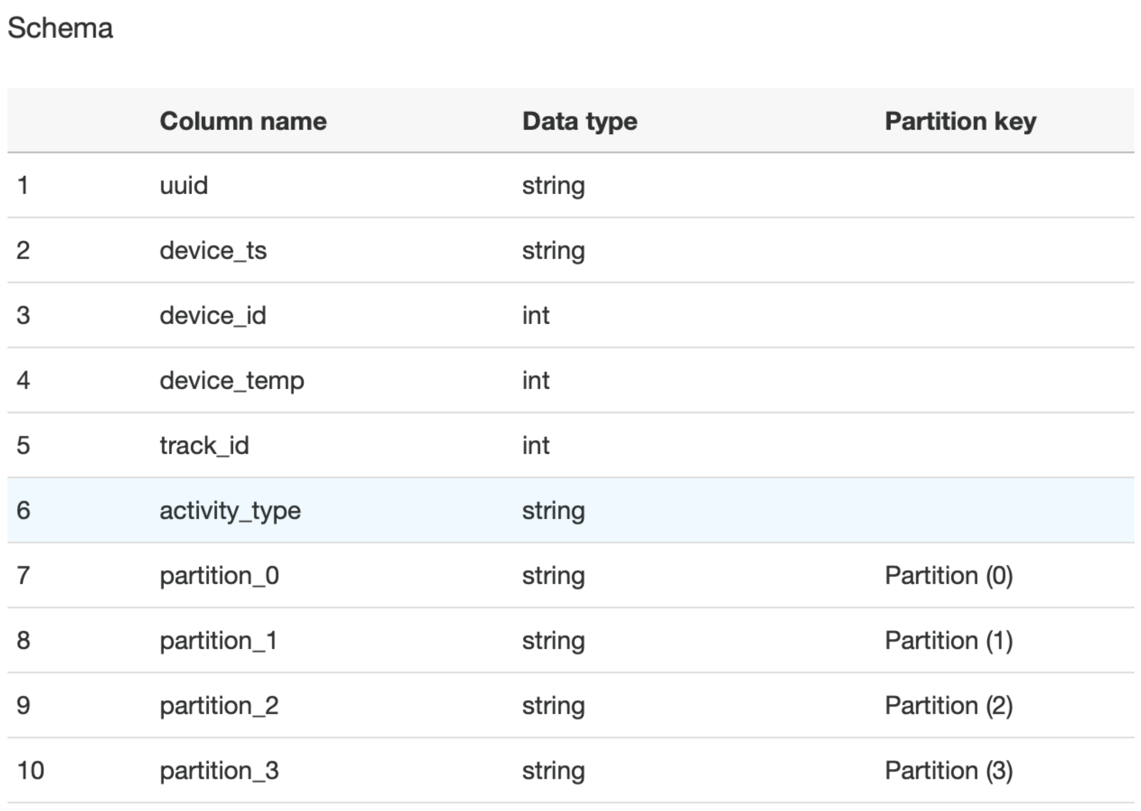
Object URL
https://615.s3.us-east-2.amazonaws.com/data/reference_data/tracks_list.json

Description: The above Screenshot shows the track_list json file being stored inside the reference_data folder in S3 Bucket named 615.

2) Catalog Data

We used AWS Glue Crawlers to crawl the raw & reference data from S3 bucket in JSON format and convert that into a tabular schema format and populate them in the AWS Glue Data Catalog under Tables.

Screenshot of Raw data with Schema

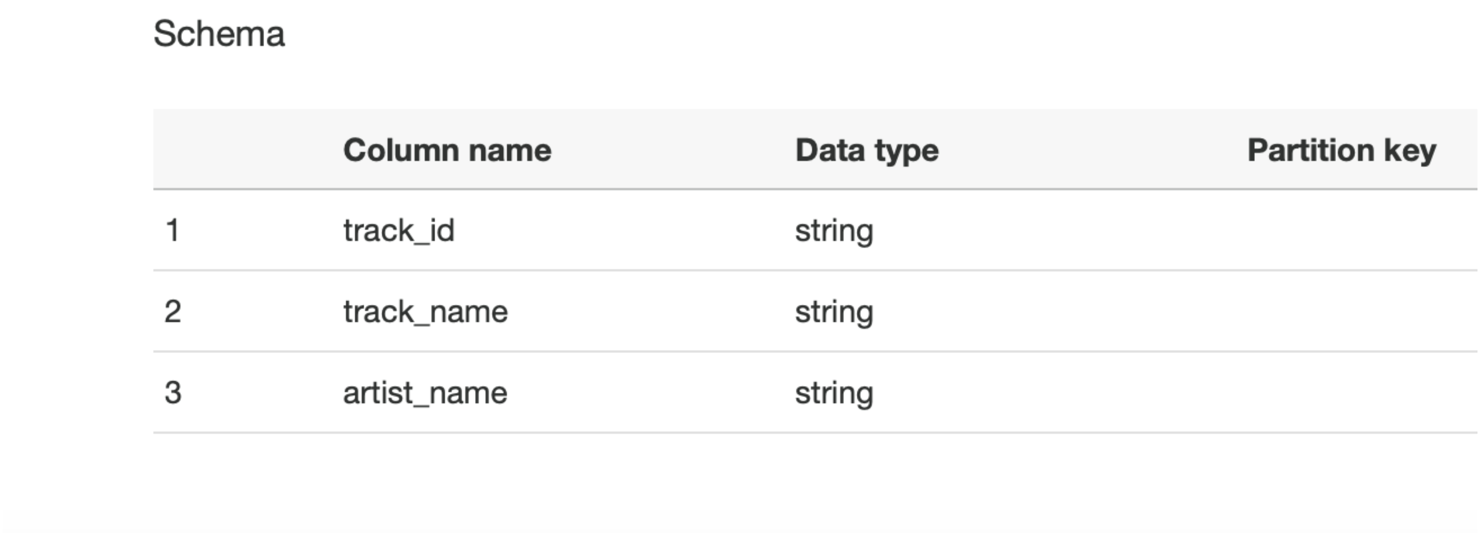


Schema

	Column name	Data type	Partition key
1	uuid	string	
2	device_ts	string	
3	device_id	int	
4	device_temp	int	
5	track_id	int	
6	activity_type	string	
7	partition_0	string	Partition (0)
8	partition_1	string	Partition (1)
9	partition_2	string	Partition (2)
10	partition_3	string	Partition (3)

Description: The above screenshot shows how all the key value pair of raw Song data was converted into Relational Database with some defined schema of Data types

Screenshot of Reference data schema

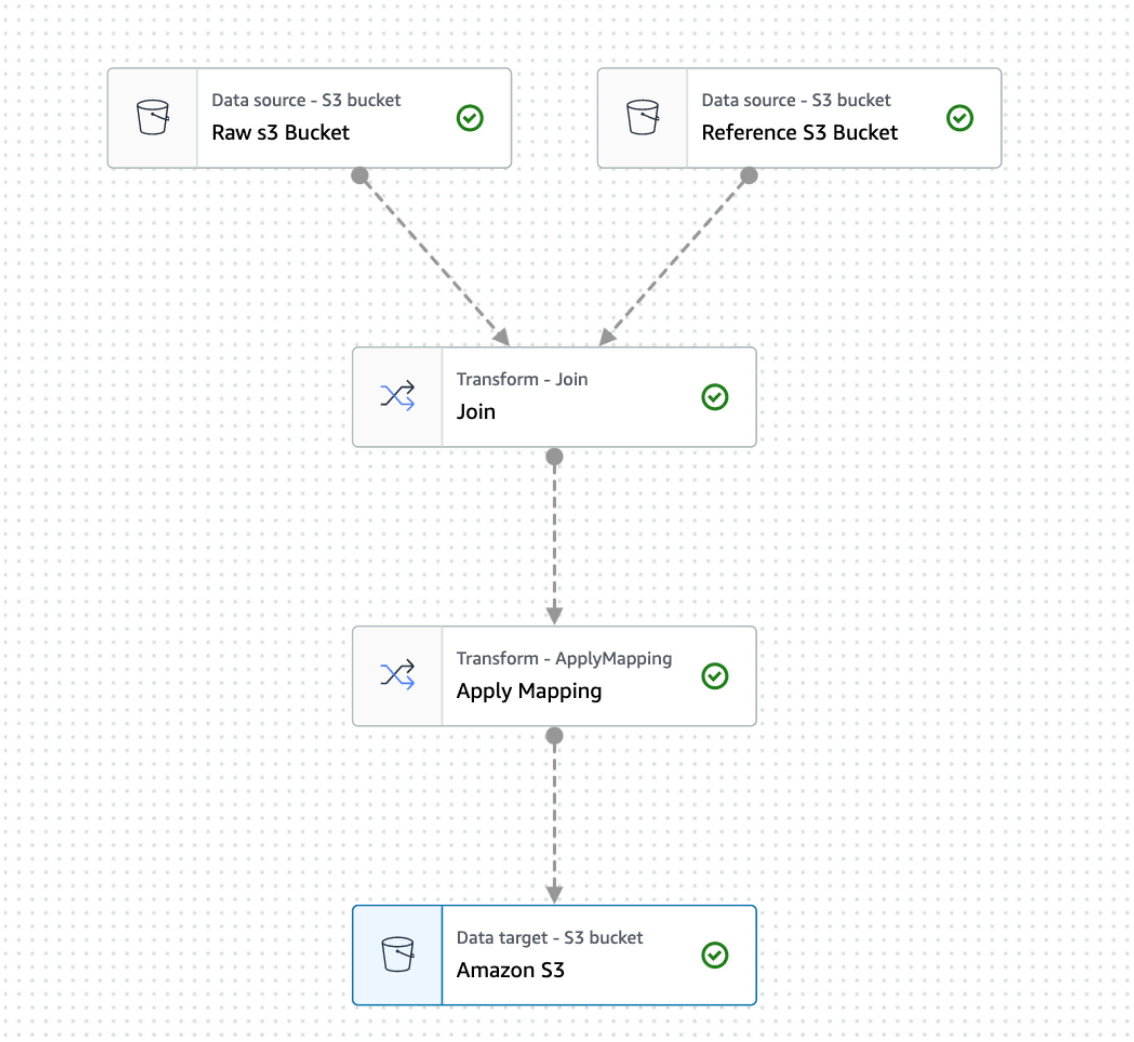


Schema

	Column name	Data type	Partition key
1	track_id	string	
2	track_name	string	
3	artist_name	string	

Description: The above screenshot shows how all the key value pair of reference Song data was converted into Relational Database with some defined schema of Data types

3) Transform Data

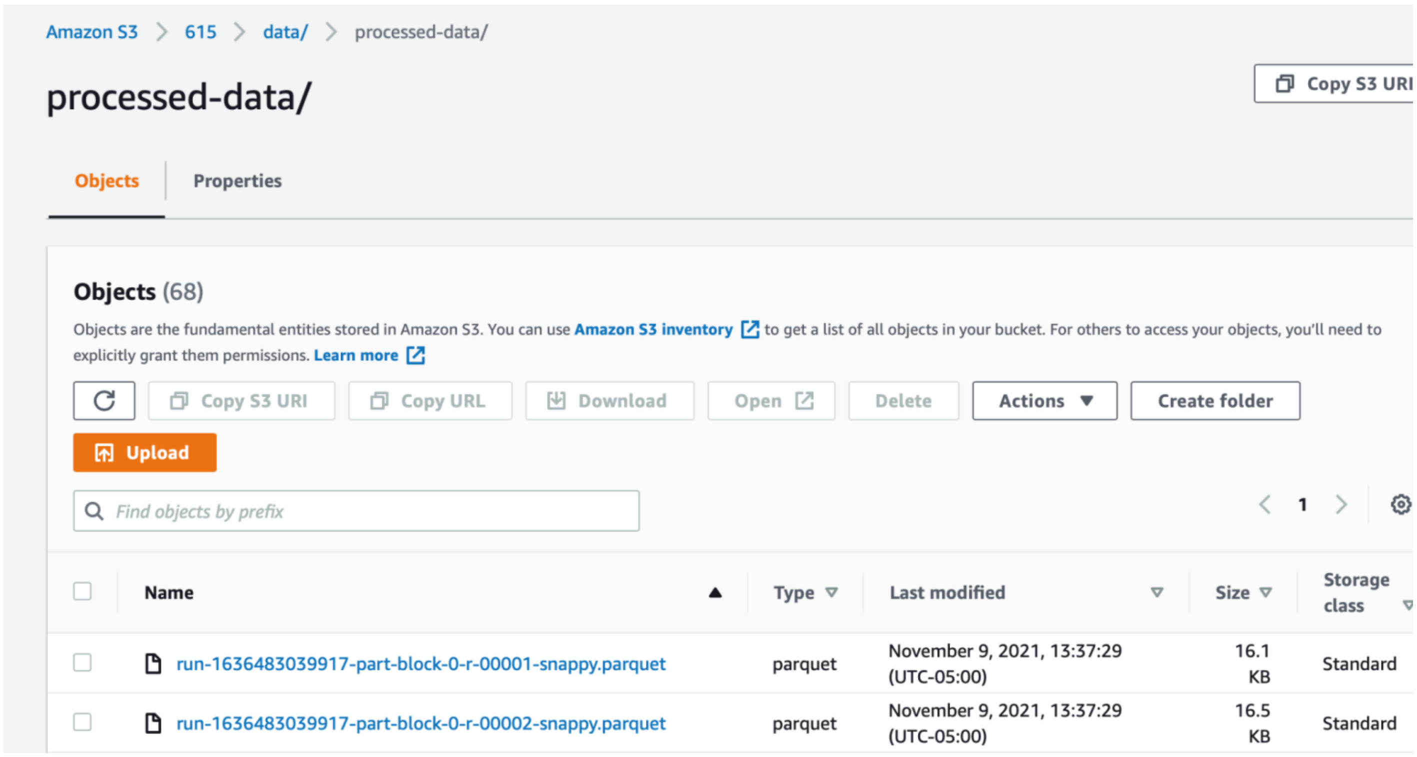


We have leveraged AWS Glue Studio which is an easy tool for generating scripts for ETL transformation of desired language such as python.(Check Appendix for Python Script)

The transformation of data is done by -

- Joining Raw Songs data with the Reference Data using common column "track_id"
- Changing the Data type of track_id from *string* to *int*
- Dropping unwanted columns such as partition-1,2,3 (system generated)

Screenshot of processed data stored in S3 Bucket under /processed folder



processed-data/

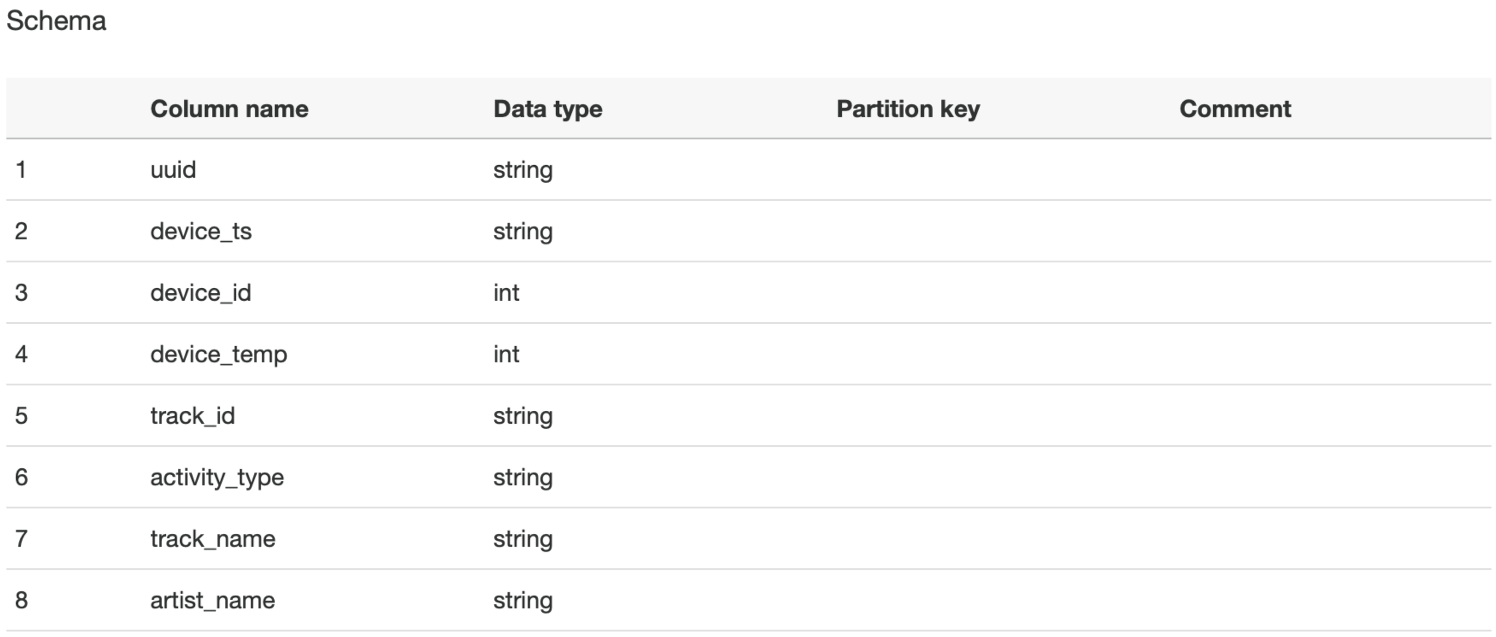
Objects (68)

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

Name	Type	Last modified	Size	Storage class
run-1636483039917-part-block-0-r-00001-snappy.parquet	parquet	November 9, 2021, 13:37:29 (UTC-05:00)	16.1 KB	Standard
run-1636483039917-part-block-0-r-00002-snappy.parquet	parquet	November 9, 2021, 13:37:29 (UTC-05:00)	16.5 KB	Standard

Description: The above screenshot shows The transformed data is then stored in the same above-mentioned S3 bucket under another file structure named as "*processed data*".

Screenshot of Processed data schema


Schema

	Column name	Data type	Partition key	Comment
1	uuid	string		
2	device_ts	string		
3	device_id	int		
4	device_temp	int		
5	track_id	string		
6	activity_type	string		
7	track_name	string		
8	artist_name	string		

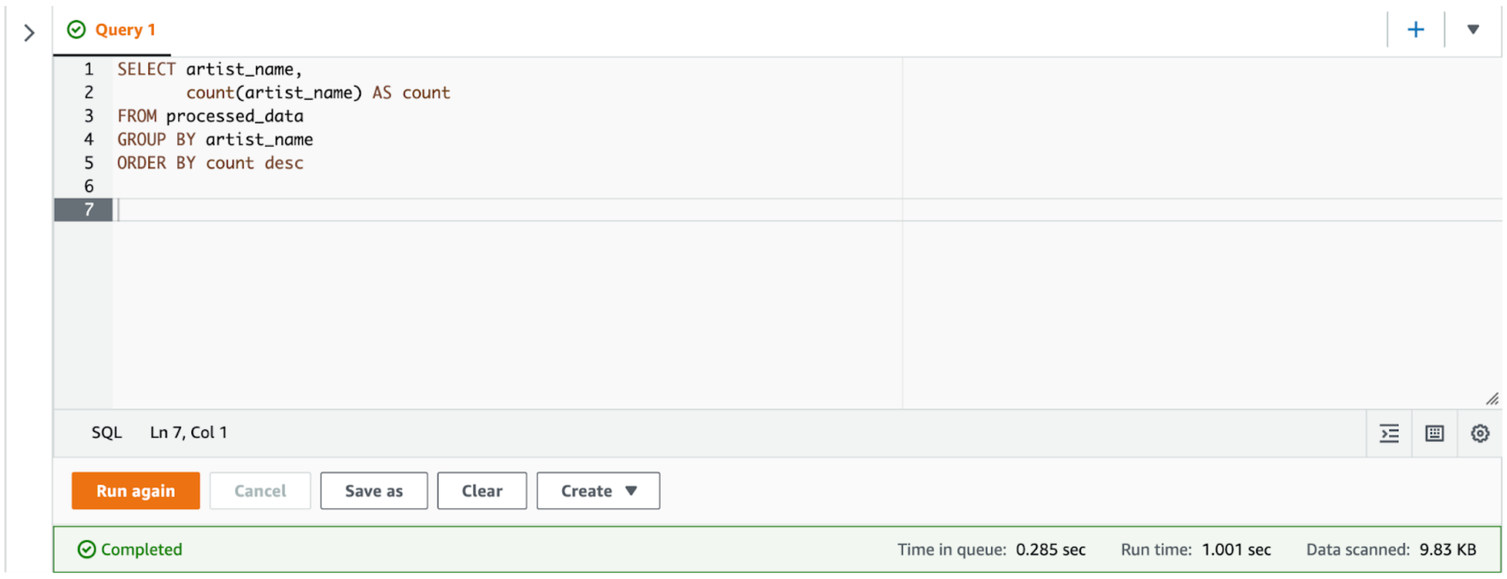
Description: The *processed data* is further crawled using AWS crawler to create a schema of processed data table in the AWS Glue data catalog in the same fashion similar to raw and referenced data schema tables defined earlier.

Lastly the Transformed *processed data* table is being pushed to the Amazon Redshift Data warehouse.

4) Analyze with Athena

With Amazon Athena we will be able to explore the transformed *processed data* using Standard SQL queries to make meaningful insights out of it.

Screenshot of the Athena Query Editor

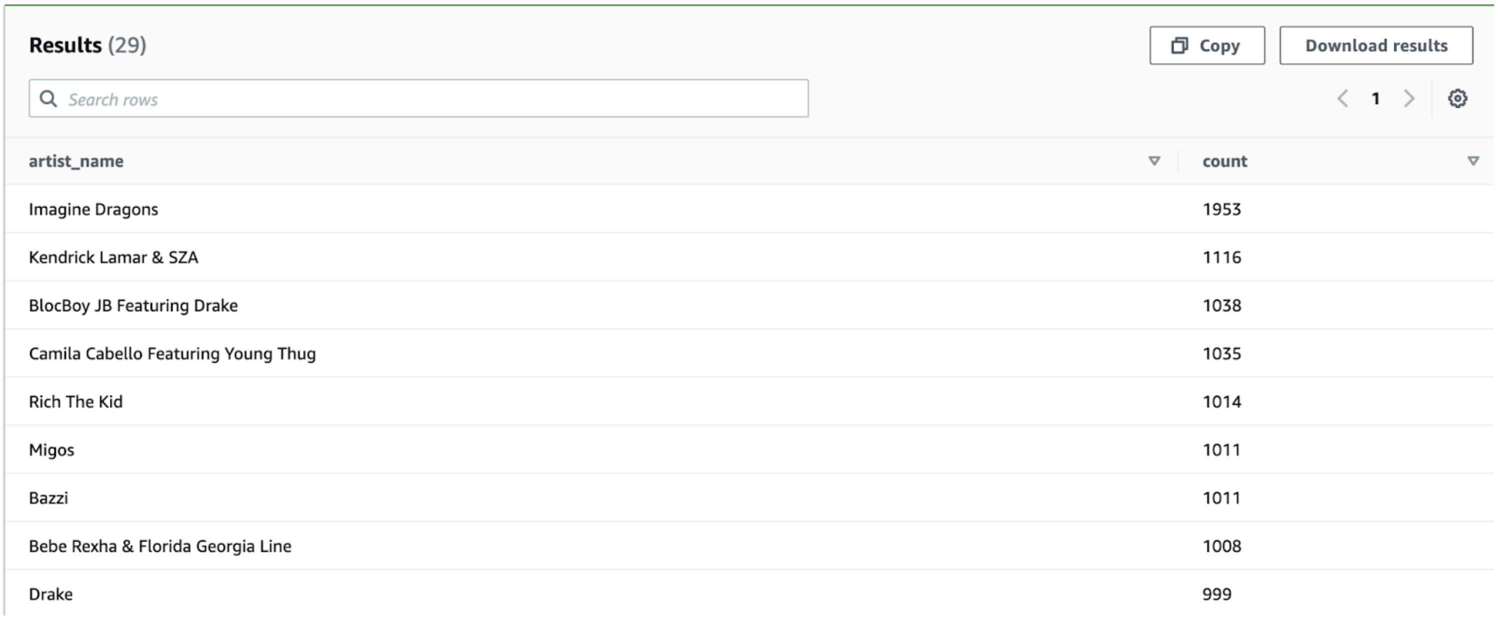


```
SELECT artist_name,
       count(artist_name) AS count
FROM processed_data
GROUP BY artist_name
ORDER BY count desc

```

Completed   Time in queue: 0.285 sec   Run time: 1.001 sec   Data scanned: 9.83 KB

Query Description - Arrange the artists names from the processed data Table in descending order to find the count from most popular trending artists to least popular.

Screenshot of the Results of the above query



Results (29)

Copy | Download results

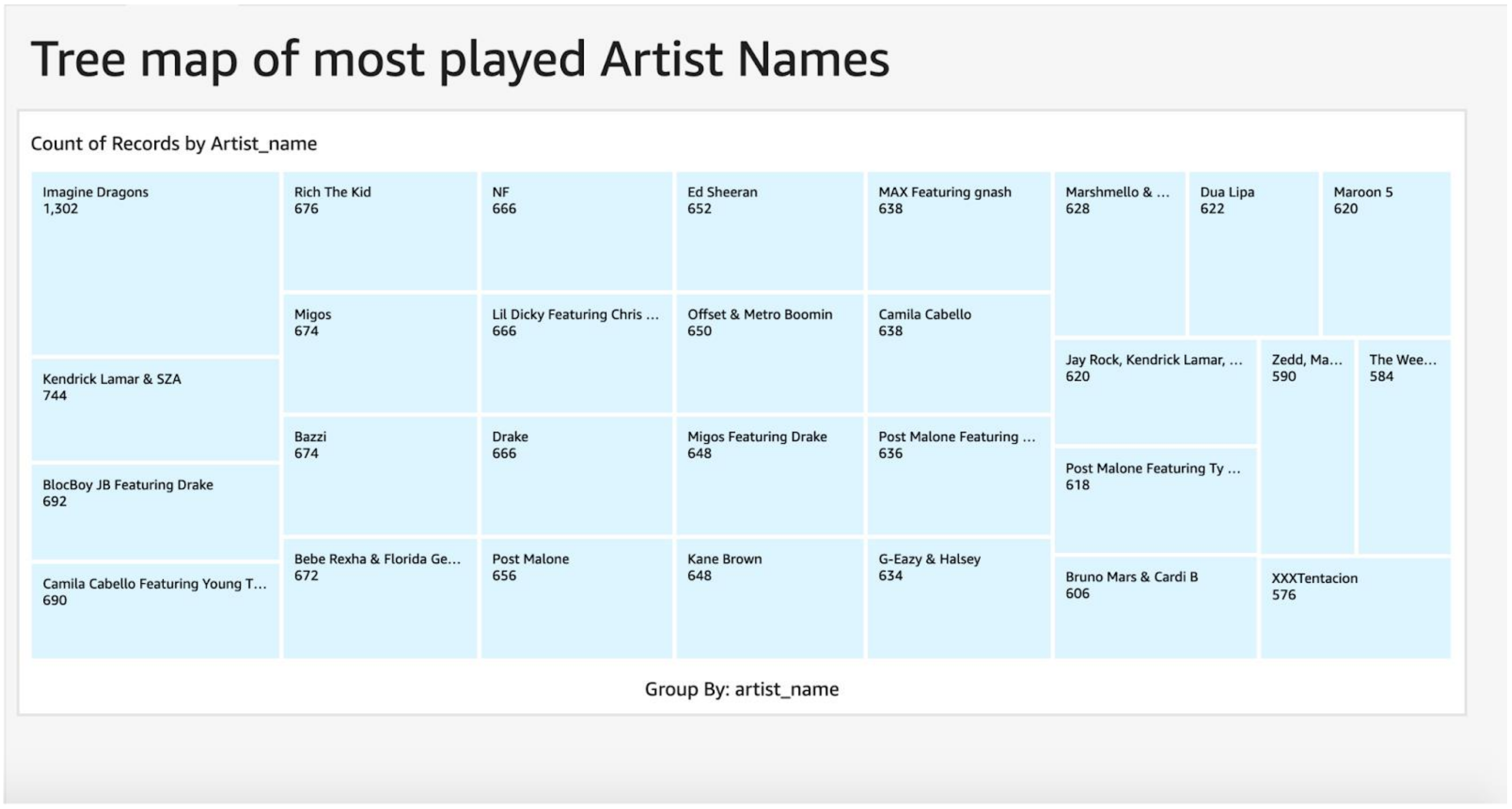
artist_name	count
Imagine Dragons	1953
Kendrick Lamar & SZA	1116
BlocBoy JB Featuring Drake	1038
Camila Cabello Featuring Young Thug	1035
Rich The Kid	1014
Migos	1011
Bazzi	1011
Bebe Rexha & Florida Georgia Line	1008
Drake	999

Description: The above screenshot shows the results displayed from executing the query which we would be using to display on the Dashboard in Graphical format.

5) Visualize on Quicksight

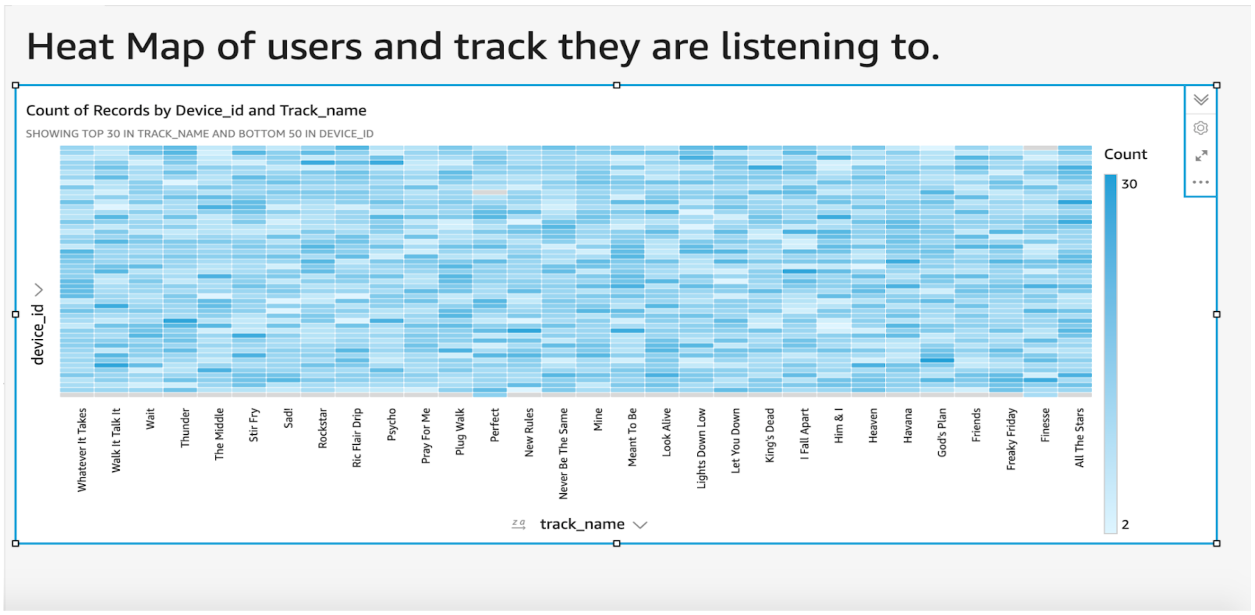
After leveraging AWS Athena and finally querying the output of AWS Athena we have used AWS QuickSight BI tool to produce BI insights in the form of dashboards.

Screenshot of Tree Map Graph



Description: The above screenshot display a tree map which we created on QuickSight to display which Artist is most played by the users and what is the count of the users playing those artist.

Description: The below screenshot displays a Heat map which we created on QuickSight to display which Track is trending within users and what is the count of the users playing those tracks is displayed using color intensity.



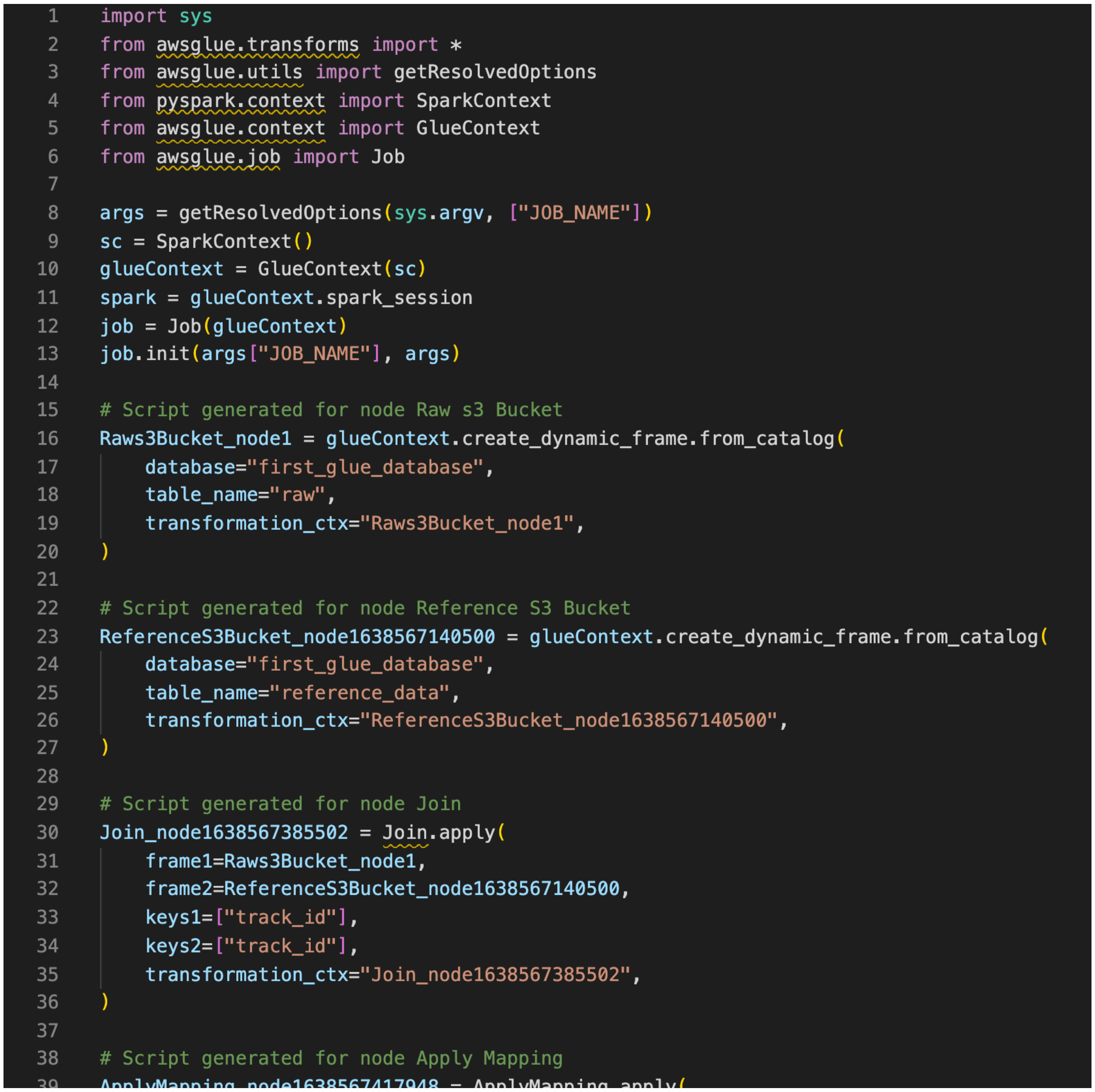
Screenshot of Heat Map Graph

D) <u>Things Learned:</u>

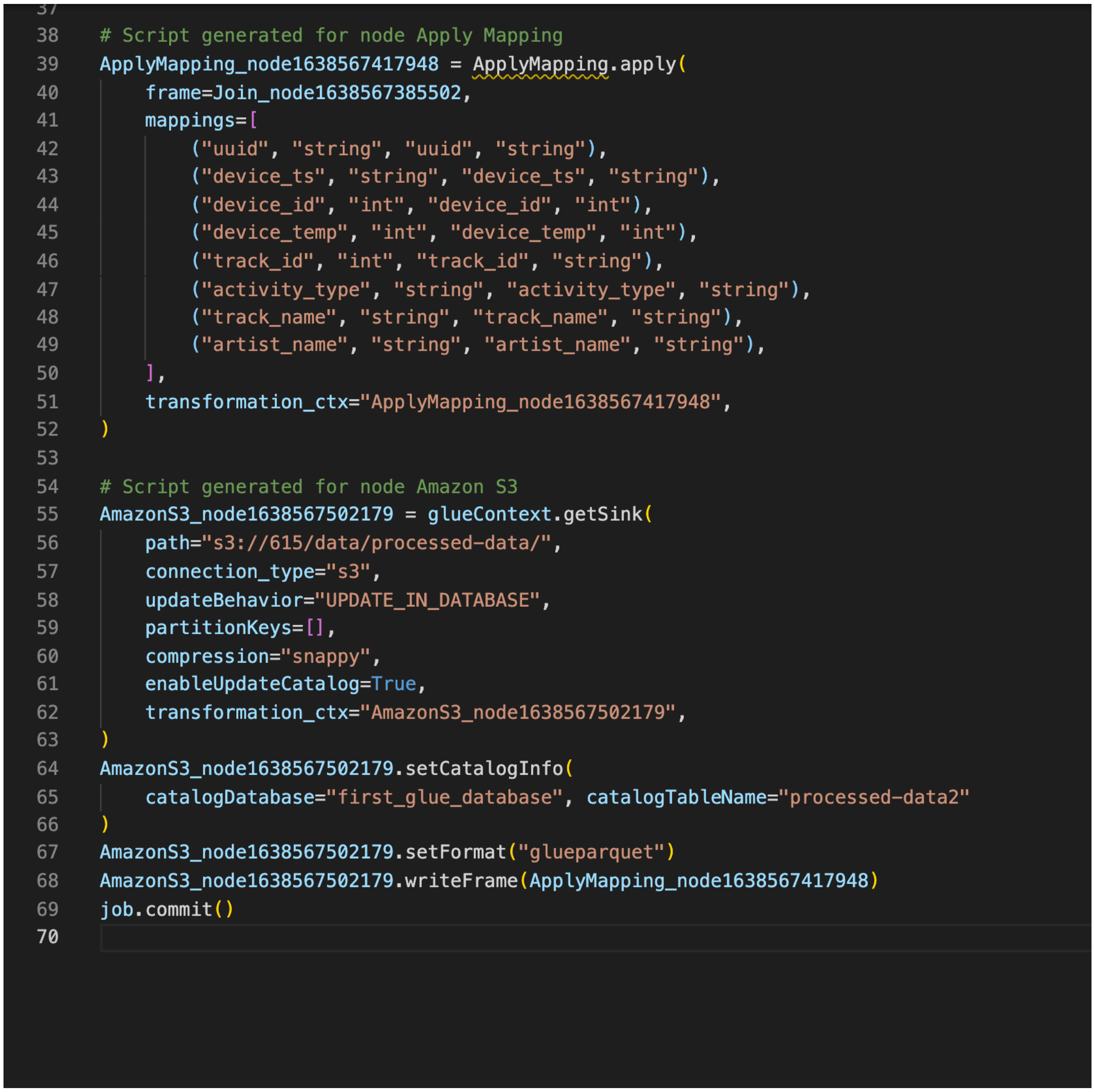
- Design serverless data lake architecture
- Build a data processing pipeline and Data Lake using Amazon S3 for storing data
- Use Amazon Kinesis for real-time streaming data
- Use AWS Glue to automatically catalog datasets
- Data Transformation
    - Use Glue Studio to run, and monitor ETL jobs in AWS Glue.
- Query data using Amazon Athena
- Visualize it using Amazon QuickSight

APPENDIX

Python Script Generated by AWS Glue Studio



```python
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

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node Raw s3 Bucket
Raws3Bucket_node1 = glueContext.create_dynamic_frame.from_catalog(
    database="first_glue_database",
    table_name="raw",
    transformation_ctx="Raws3Bucket_node1",
)

# Script generated for node Reference S3 Bucket
ReferenceS3Bucket_node1638567140500 = glueContext.create_dynamic_frame.from_catalog(
    database="first_glue_database",
    table_name="reference_data",
    transformation_ctx="ReferenceS3Bucket_node1638567140500",
)

# Script generated for node Join
Join_node1638567385502 = Join.apply(
    frame1=Raws3Bucket_node1,
    frame2=ReferenceS3Bucket_node1638567140500,
    keys1=["track_id"],
    keys2=["track_id"],
    transformation_ctx="Join_node1638567385502",
)

# Script generated for node Apply Mapping
ApplyMapping_node1638567417948 = ApplyMapping.apply(
```



```
# Script generated for node Apply Mapping
ApplyMapping_node1638567417948 = ApplyMapping.apply(
    frame=Join_node1638567385502,
    mappings=[
        ("uuid", "string", "uuid", "string"),
        ("device_ts", "string", "device_ts", "string"),
        ("device_id", "int", "device_id", "int"),
        ("device_temp", "int", "device_temp", "int"),
        ("track_id", "int", "track_id", "string"),
        ("activity_type", "string", "activity_type", "string"),
        ("track_name", "string", "track_name", "string"),
        ("artist_name", "string", "artist_name", "string"),
    ],
    transformation_ctx="ApplyMapping_node1638567417948",
)

# Script generated for node Amazon S3
AmazonS3_node1638567502179 = glueContext.getSink(
    path="s3://615/data/processed-data/",
    connection_type="s3",
    updateBehavior="UPDATE_IN_DATABASE",
    partitionKeys=[],
    compression="snappy",
    enableUpdateCatalog=True,
    transformation_ctx="AmazonS3_node1638567502179",
)
AmazonS3_node1638567502179.setCatalogInfo(
    catalogDatabase="first_glue_database", catalogTableName="processed-data2"
)
AmazonS3_node1638567502179.setFormat("glueparquet")
AmazonS3_node1638567502179.writeFrame(ApplyMapping_node1638567417948)
job.commit()
```