We can use DeltaLake file format with AWS Glue by using the Open Source DeltaLake library JAR file and configuring the Glue Job options. The JAR file can be downloaded from the Maven repository and stored in the S3 to use in the Glue job. In addition to configuring both Python lib path and Jar lib path to this S3 location, the Job parameters option in the Glue job needs to be configured as shown below.

Python lib path	s3://aws-glue-pr/jar/delta-core_2.11-0.6.1.jar
Jar lib path	s3://aws-glue-pr/jar/delta-core_2.11-0.6.1.jar
Other lib path	-
Job parameters	**--conf** **spark.delta.logStore.class=org.apache.spark.sql.delta.storage.S3SingleDriverLogStore** **--conf spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension**

We can use below sample code snippet to write data files in DeltaLake format

```
from awsglue.context import GlueContext
from pyspark.context import SparkContext

# create/get spark + glue context
spark = GlueContext(SparkContext.getOrCreate()).sparkSession

# folder within S3 for the delta table
s3_path = f"s3://aws-glue-pr/users"

# initially prepopulate the table with some data
users_initial = [
    { 'user_id': 1, 'name': 'Gina Burch', 'gender': 'f' },
    { 'user_id': 2, 'name': 'Francesco Coates', 'gender': 'm' },
    { 'user_id': 3, 'name': 'Saeed Wicks', 'gender': 'm' },
    { 'user_id': 4, 'name': 'Raisa Oconnell', 'gender': 'f' },
    { 'user_id': 5, 'name': 'Josh Copeland', 'gender': 'm' },
    { 'user_id': 6, 'name': 'Kaiden Williamson', 'gender': 'm' }
]

spark.createDataFrame(users_initial) \
  .write.format("delta").mode("overwrite").save(s3_path)

# load and print results via Spark API
print("DF reading after initial load:")
spark.read.format("delta").load(s3_path).orderBy("user_id").show()
```

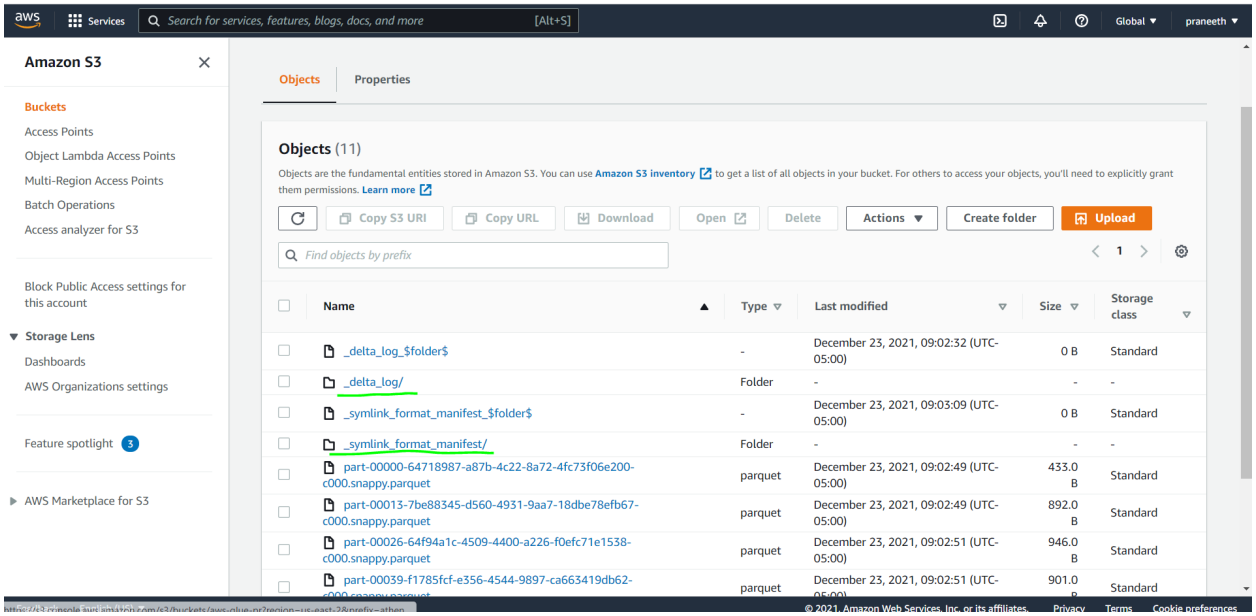
This would generate standard parquet files with transaction logs in JSON format. Once written the data can be queried through Athena by generating the manifest file from the written DelatLake data.

The manifest files can be generated using the below API calls

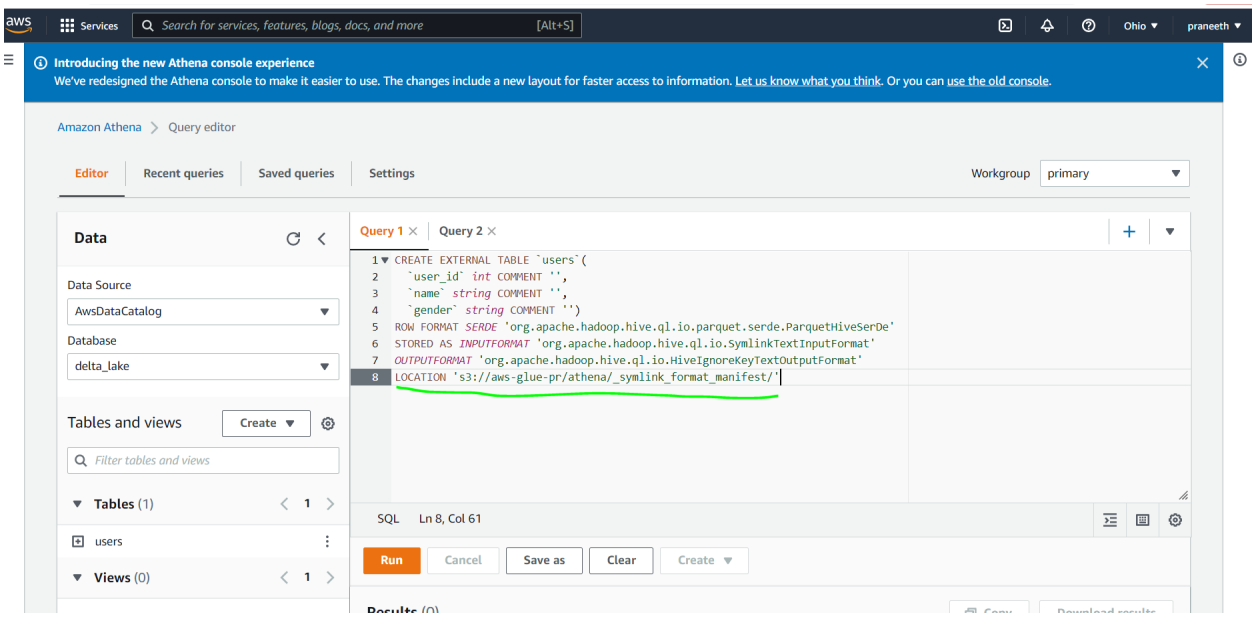
```
27  deltaTable = DeltaTable.forPath(spark, s3_path)
28  deltaTable.generate("symlink_format_manifest")
```

Logs | Schema

After this step the S3 data folder would look like the below picture



Here the **_delta_log** folder contains the DettaLake transaction log files in JSON format and **_symlink_format_manifest** contains the manifest files. After this we can query the deltalake data in the Athena by creating the external tables pointing to the manifest files location as in the below image



The below image shows the query result of a DeltaLake folder