

# Automate CSV to JSON Transformation Between S3 Buckets Using AWS Glue and Lambda.

## Problem Statement :-

Automate the process of converting a CSV file uploaded to an S3 bucket into a JSON format and storing it in another S3 bucket. The process should trigger automatically upon CSV file .

## Objective :-

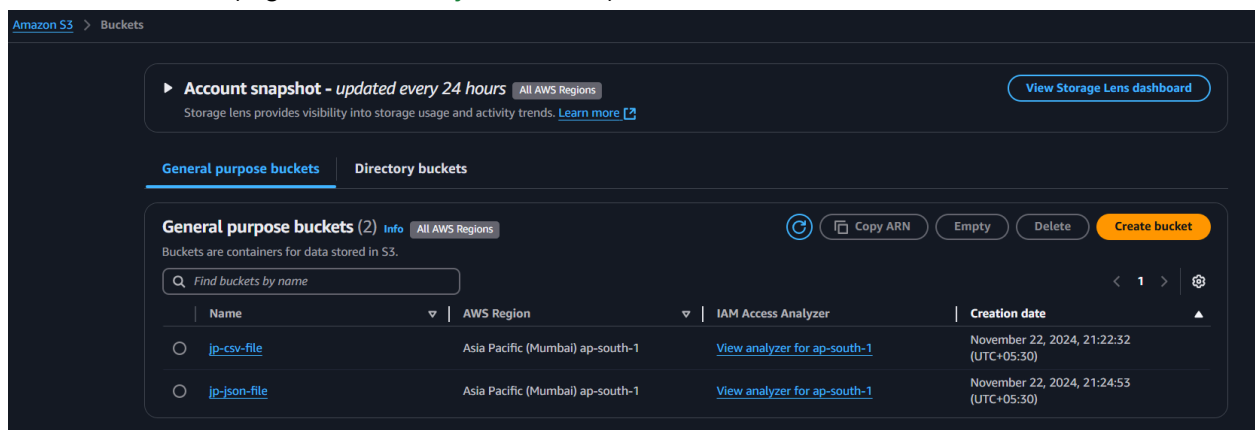
- Set up an AWS Glue job to convert CSV files to JSON format.
- Use AWS Lambda to trigger the process automatically whenever a CSV file is uploaded to the source S3 bucket.
- Ensure all services have the necessary permissions by configuring role

## Implementation Steps

### Step 1: Create Source and Destination S3 Buckets

#### 1. Create Source S3 Bucket :-

- Go to the **AWS S3 Console**.
- Click **Create Bucket**.
- Name the bucket (e.g., **source-csv-bucket**), select the region, and configure other settings (e.g., versioning if required)
- Click **Create**.
- Repeat the same steps as above to create another S3 bucket where the JSON files will be stored (e.g., **destination-json-bucket**).



## Step 2: Create IAM Role

### 1. Create IAM Role with Full Access to Lambda, Glue, and S3 :-

- Navigate to the **IAM Console**.
- Click on **Roles** and then **Create Role**.
- Select **AWS Service** as the trusted entity, and then choose **Lambda** as the use case
- Attach the following policies to the role:
  - **AmazonS3FullAccess**
  - **AWSGlueServiceRole**
  - **AWSGlueConsoleFullAccess**
  - **AWSLambda\_FullAccess**
- Give the role a name (e.g., **Data-pipeline-role**) and create it

The screenshot shows the 'Create role' page in the AWS IAM console. The breadcrumb navigation is 'IAM > Roles > Create role'. The left sidebar shows the progress: Step 1 'Select trusted entity' is active, Step 2 'Add permissions' is next, and Step 3 'Name, review, and create' is the current step. The main content area is titled 'Name, review, and create'. Under 'Role details', the 'Role name' field contains 'Data-pipeline-jp' with a note 'Maximum 64 characters. Use alphanumeric and '\*+,@-.' characters.' The 'Description' field contains 'Allows Glue to call AWS services on your behalf.' with a note 'Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: \_+=, @-/\[\]]\*%\*&~!<>'. An 'Edit' button is in the top right corner.

Step 1: Select trusted entities

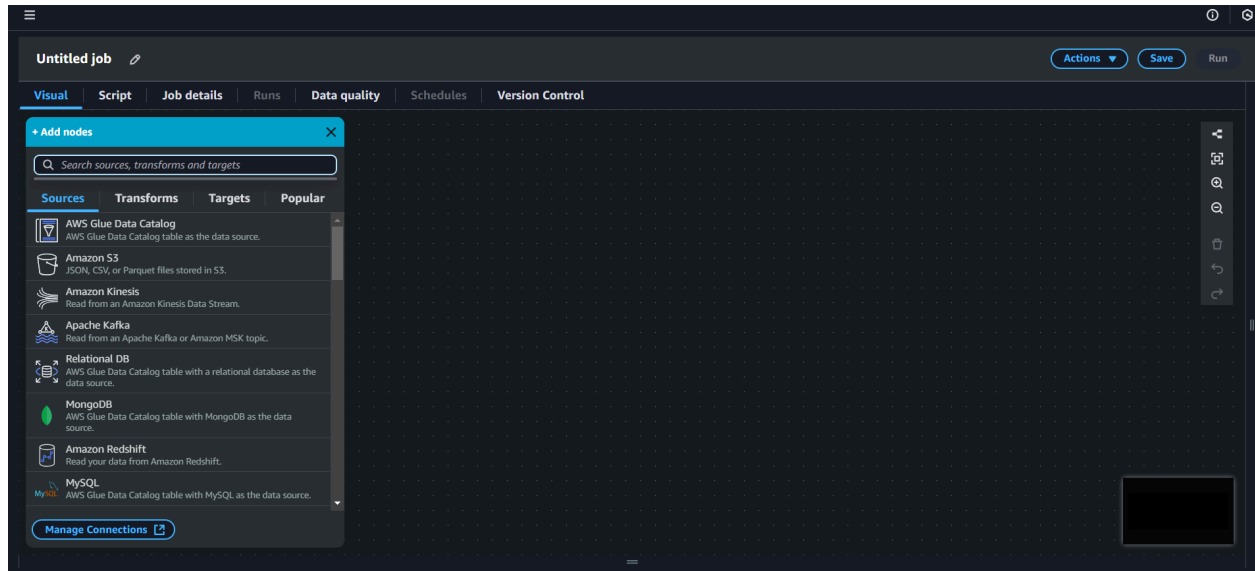
The screenshot shows the 'Add permissions' step of the 'Create role' process. The left sidebar shows Step 1 'Select trusted entity' as completed and Step 2 'Add permissions' as the current step. The main content area is titled 'Step 2: Add permissions'. It includes a 'Permissions policy summary' table with four rows: 'AmazonS3FullAccess', 'AWSGlueConsoleFullAccess', 'AWSGlueServiceRole', and 'AWSLambda\_FullAccess', all of type 'AWS managed' and attached as 'Permissions policy'. Below the table is 'Step 3: Add tags' section, which states 'Add tags - optional' and 'Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.' It shows 'No tags associated with the resource.' and an 'Add new tag' button with a note 'You can add up to 50 more tags.' At the bottom right are 'Cancel', 'Previous', and 'Create role' buttons.

Policy name	Type	Attached as
AmazonS3FullAccess	AWS managed	Permissions policy
AWSGlueConsoleFullAccess	AWS managed	Permissions policy
AWSGlueServiceRole	AWS managed	Permissions policy
AWSLambda_FullAccess	AWS managed	Permissions policy

## Step 3: Set Up AWS Glue for ETL Job Using Visual Interface

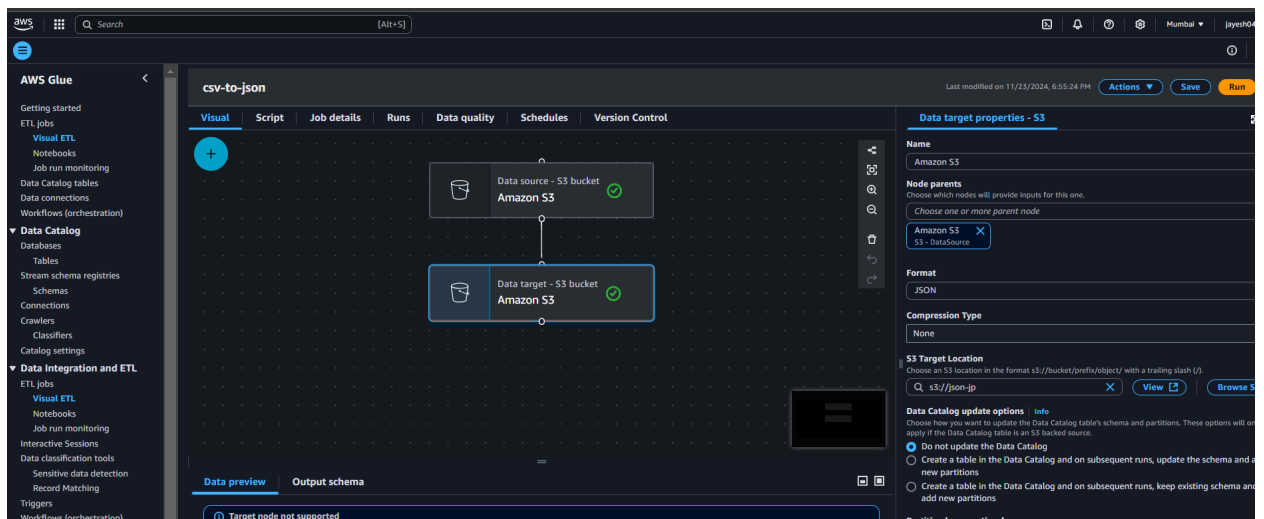
### 1. Navigate to AWS Glue and Start ETL Job Setup:

- Go to the AWS Glue console from the AWS Management Console.
- On the Glue dashboard, click on ETL Jobs to start setting up a new ETL job.



### 2. Launch the Visual Interface:

- When the ETL job setup screen appears, select the Visual with source and target option to use the visual editor.



### 3. Configure the Source (S3 Bucket with CSV Files) :-

- In the visual interface, click Add Source.
- Choose S3 as the data store.
- Specify the S3 bucket where the CSV files are stored (e.g., **source-csv-bucket**).

- In the File Format section, select CSV as the format for the source files

**Data source properties - S3**

**Name**

Amazon S3

**S3 source type** | Info

☒ S3 location  
Choose a file or folder in an S3 bucket.

☐ Data Catalog table

**S3 URL**

Q s3://csv-jp X View Browse S3

☒ Recursive  
Read files in all subdirectories.

**Data format**

CSV

**Delimiter**

Comma (,)

**Escape character - optional**  
Enter a character to use for escaping

The character which immediately follows is used as-is, except for a small set of well-known escapes (\n, \r, \t, and \0)

**Quote character**

Double quote (")

#### 4. Configure the Destination (S3 Bucket for JSON Files):

- Click Add Target to specify the destination for the transformed data.
- Choose S3 as the destination data store.
- Specify the target S3 bucket where the JSON files will be stored (e.g., `destination-json-bucket`).
- Set the file format as JSON.

### Data target properties - S3

**Name**

**Node parents**  
Choose which nodes will provide inputs for this one.

**Format**

**Compression Type**

**S3 Target Location**  
Choose an S3 location in the format s3://bucket/prefix/object/ with a trailing slash (/).

**Data Catalog update options** | [Info](#)  
Choose how you want to update the Data Catalog table's schema and partitions. These options will only apply if the Data Catalog table is an S3 backed source.  
☒ Do not update the Data Catalog  
☐ Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions  
☐ Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions

**Partition keys - optional**  
Add partition keys.

## 5. Job Details:

- After configuring the source and destination, go to the Job Details section.
- Enter a name for the job (e.g., `csv-to-json-etl-job`).
- Add the necessary role to allow Glue to access the resources (S3, Lambda).

### Basic properties [Info](#)

**Name**

**Description - optional**  
  
Descriptions can be up to 2048 characters long.

**IAM Role**  
Role assumed by the job with permission to access your data stores. Ensure that this role has permission to your Amazon S3 sources, targets, temporary directory, scripts, and any libraries used by the job.

**Type**  
The type of ETL job. This is set automatically based on the types of data sources you have selected.

**Glue version** | [Info](#)

**Language**

**Worker type**  
Set the type of predefined worker that is allowed when a job runs.

**Automatically scale the number of workers**  
☐ AWS Glue will optimize costs and resource usage by dynamically scaling the number of workers up and down throughout the job run. Requires Glue 3.0 or later.

**Requested number of workers**

## 6. Save the Job:

- Click Create to save the job.

## Step 4: Set Up AWS Lambda for Automatic Trigger

### 1. Create Lambda Function:

- Go to the **Lambda Console**
- Click on **Create Function** and choose Author from Scratch.
  - **Name:** Give the function a name (e.g., **csv-upload-trigger**).
  - **Runtime:** Select Python 3.x (or any preferred runtime).
  - **Permissions:** Create a new role with lambda as trusted entity and same permissions as before or you can create role automatically and add permissions later

The screenshot shows the 'Create function' page in the AWS Lambda console. The breadcrumb navigation is 'Lambda > Functions > Create function'. The page title is 'Create function' with an 'Info' link. Below the title, it says 'Choose one of the following options to create your function.' There are three radio button options: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. The 'Author from scratch' option has a subtext 'Start with a simple Hello World example.' Below the options is the 'Basic information' section. It contains three fields: 'Function name' with the value 'Data-pipeline', 'Runtime' with a dropdown set to 'Python 3.13', and 'Architecture' with a radio button set to 'x86\_64'. Each field has an 'Info' link. The 'Function name' field has a subtext 'Enter a name that describes the purpose of your function.' and a validation message: 'Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).' The 'Runtime' field has a subtext 'Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.' The 'Architecture' field has a subtext 'Choose the instruction set architecture you want for your function code.'

The screenshot shows the 'Add trigger' page in the AWS Lambda console. The breadcrumb navigation is 'Lambda > Add triggers'. The page title is 'Add trigger'. Below the title is the 'Trigger configuration' section with an 'Info' link. It contains a dropdown menu for the trigger type, currently set to 'S3'. Below this is the 'Bucket' section with a text input field containing 's3/jp-csv-file' and a 'Bucket region' dropdown set to 'ap-south-1'. Below the bucket information is the 'Event types' section with a dropdown menu and a 'PUT' button. Below the event types is the 'Prefix - optional' section with a text input field containing 'e.g. images/'. Below the prefix is the 'Suffix - optional' section with a text input field containing '.csv'. At the bottom is the 'Recursive invocation' section with a checkbox that is checked, indicating acknowledgment of the risks of recursive invocations.

## 2. Write Lambda Code:

- The Lambda function will trigger the Glue job when a CSV file is uploaded. Use the following Python code as a starting point:

```
import json
import boto3
# Initialize Glue client
glue_client = boto3.client('glue')
def lambda_handler(event, context):
# Start the Glue job
response = glue_client.start_job_run(
JobName='csv-to-json-job') #Replace with your Glue job name
return {
'statusCode': 200,
'body': json.dumps('CSV to JSON Glue job triggered
successfully!')}
}
```


## 3. Set up S3 Trigger for Lambda:



- In the Lambda function console, click on **Add Trigger**.
- Select **S3** as the trigger source.
- Choose the source S3 bucket (**source-csv-bucket**).
- Set the event type to **PUT** to trigger the Lambda function whenever a CSV file is uploaded.

Lambda > Add triggers


### Add trigger


**Trigger configuration** Info


 **S3**  
aws asynchronous storage


**Bucket**  
Choose or enter the ARN of an S3 bucket that serves as the event source. The bucket must be in the same region as the function.  
    
Bucket region: ap-south-1


**Event types**  
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.  



**PUT** 

**Prefix - optional**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters. Any **special characters**  must be URL encoded.

**Suffix - optional**  
Enter a single optional suffix to limit the notifications to objects with keys that end with matching characters. Any **special characters**  must be URL encoded.

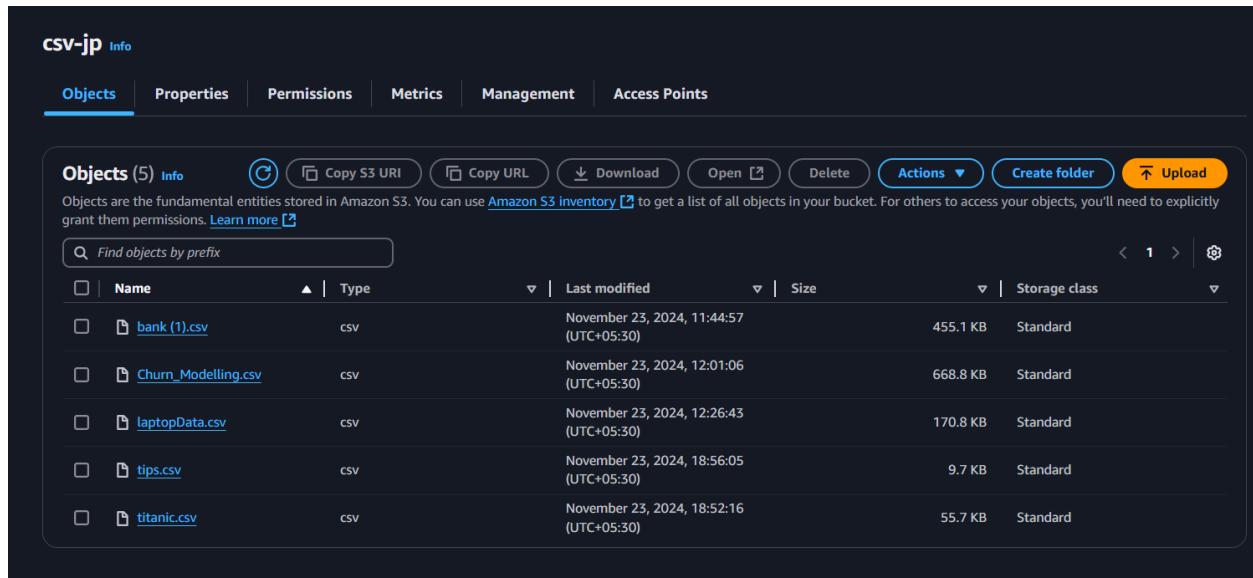
**Recursive invocation**  
If your function writes objects to an S3 bucket, ensure that you are using different S3 buckets for input and output. Writing to the same bucket increases the risk of creating a recursive invocation, which can result in increased Lambda usage and increased costs. [Learn more](#)   
☒ I acknowledge that using the same S3 bucket for both input and output is not recommended and that this configuration can cause recursive invocations, increased Lambda usage, and increased costs.

#### 4. Deploy :

- Deploy the Lambda function.
- Upload a CSV file to the **source-csv-bucket** and check if the Lambda function triggers the Glue job, converting the CSV to JSON and storing it in the **destination-json-bucket**.

#### 5.Test :

Input :-



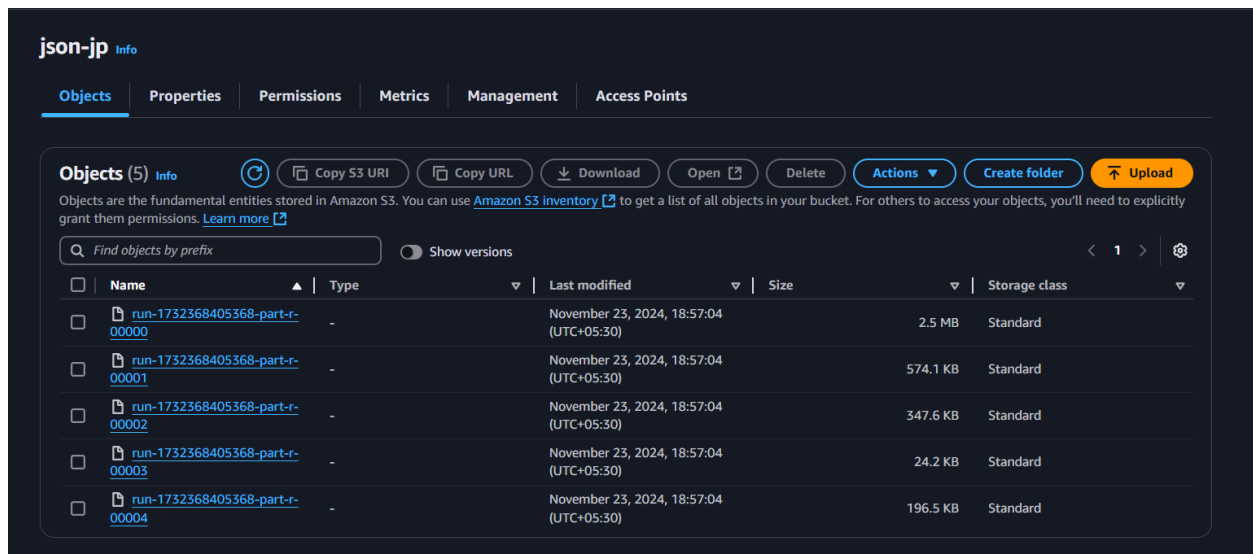
**csv-jp** Info

Objects (5) Info

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
<a href="#">bank (1).csv</a>	csv	November 23, 2024, 11:44:57 (UTC+05:30)	455.1 KB	Standard
<a href="#">Churn_Modelling.csv</a>	csv	November 23, 2024, 12:01:06 (UTC+05:30)	668.8 KB	Standard
<a href="#">laptopData.csv</a>	csv	November 23, 2024, 12:26:43 (UTC+05:30)	170.8 KB	Standard
<a href="#">tips.csv</a>	csv	November 23, 2024, 18:56:05 (UTC+05:30)	9.7 KB	Standard
<a href="#">titanic.csv</a>	csv	November 23, 2024, 18:52:16 (UTC+05:30)	55.7 KB	Standard

Output :-



**json-jp** Info

Objects (5) Info

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
<a href="#">run-1732368405368-part-r-00000</a>	-	November 23, 2024, 18:57:04 (UTC+05:30)	2.5 MB	Standard
<a href="#">run-1732368405368-part-r-00001</a>	-	November 23, 2024, 18:57:04 (UTC+05:30)	574.1 KB	Standard
<a href="#">run-1732368405368-part-r-00002</a>	-	November 23, 2024, 18:57:04 (UTC+05:30)	347.6 KB	Standard
<a href="#">run-1732368405368-part-r-00003</a>	-	November 23, 2024, 18:57:04 (UTC+05:30)	24.2 KB	Standard
<a href="#">run-1732368405368-part-r-00004</a>	-	November 23, 2024, 18:57:04 (UTC+05:30)	196.5 KB	Standard

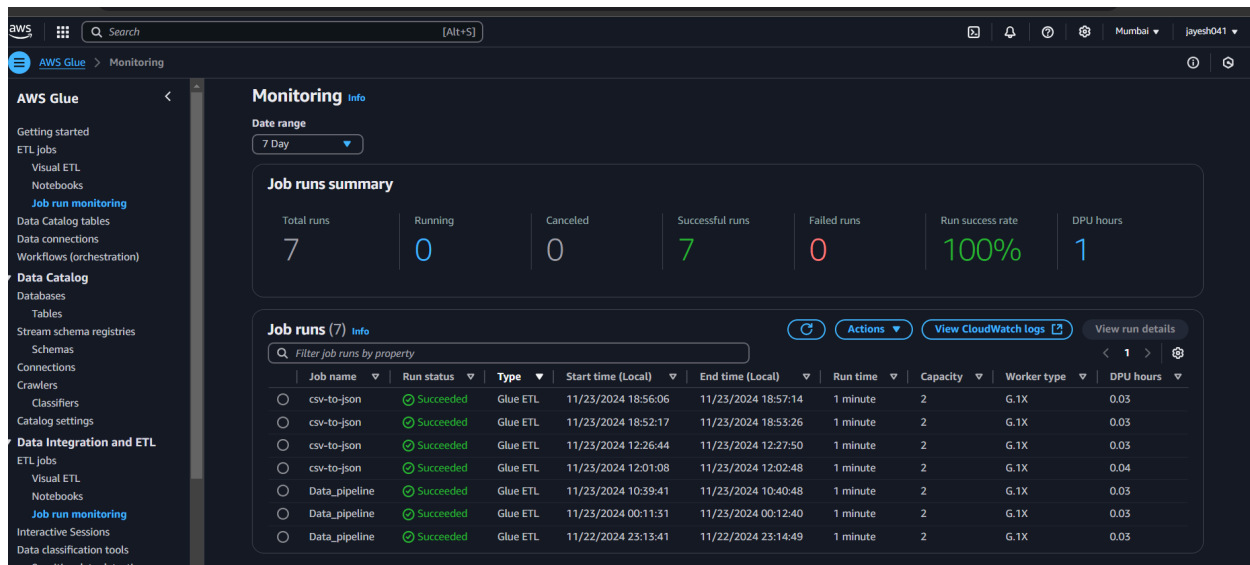
#### Step 5: Monitor and Debug

##### 1. Monitor with CloudWatch Logs:

- In the **CloudWatch Console**, check the logs for both the Lambda function and Glue job.



- Verify if the Glue job was successfully triggered and completed without errors



## Conclusion :-

The CSV to JSON transformation and automation setup was successfully completed using AWS Glue for ETL and AWS Lambda for automatic triggers. This setup efficiently converts files between two S3 buckets without manual intervention, saving time and improving data processing workflows.