

# AWS Batch Data Pipeline

Objective: To create a Batch Data pipeline in AWS by using the following services

- Lambda
- S3
- Glue
- Redshift Serverless
- Event Bridge

The Data is pulled from one of the Rapid API

## Creation of Lambda Function

I had used a Movie Database API from RapidAPI for the free version it can only provide 2 records.

First, I had created a lambda function in which it calls the API and generates data and using pandas and JSON libraries had generated required data.

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with 'Lambda' and 'API Gateway' tabs. Below that, the 'lambda\_function' is selected. The 'Code' tab is active, displaying the source code of the function. The code is a Python script that uses the boto3 library to interact with the AWS S3 API. It defines a function 'lambda\_handler' that takes an event and context as input. The function checks if the event contains a 'key' parameter. If it does, it uses the 's3\_client' to get the object's metadata and returns it. If not, it returns an empty dictionary. The code is as follows:

```

1 import boto3
2 import os
3 import json
4
5 # Initialize the S3 client
6 s3_client = boto3.client('s3')
7
8 # Define the lambda handler function
9 def lambda_handler(event, context):
10     # Check if the event contains a 'key' parameter
11     if 'key' in event:
12         # Get the object's metadata from S3
13         response = s3_client.get_object_attributes(
14             Bucket=event['bucket'],
15             Key=event['key'],
16             RequestPayer='requester'
17         )
18         # Return the metadata
19         return response['ObjectAttributes']
20     else:
21         # Return an empty dictionary if no key is provided
22         return {}
23
24 # The following code is for testing purposes only
25 if __name__ == '__main__':
26     # Parse the command line arguments
27     import sys
28     event = {}
29     if len(sys.argv) > 1:
30         event['key'] = sys.argv[1]
31         if len(sys.argv) > 2:
32             event['bucket'] = sys.argv[2]
33     # Call the lambda handler function
34     result = lambda_handler(event, None)
35     print(json.dumps(result))

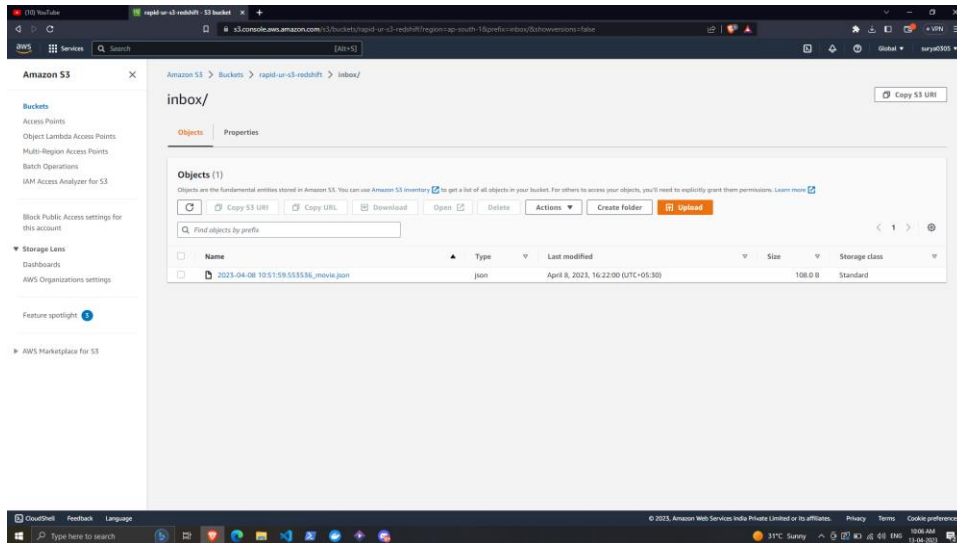
```

The console also shows the 'Function URL' tab, which is currently empty. The 'Environment' tab is also visible, showing the 'lambda\_function' environment variables.

In S3 created a bucket in which the lambda output is stored in the JSON format.

Data:

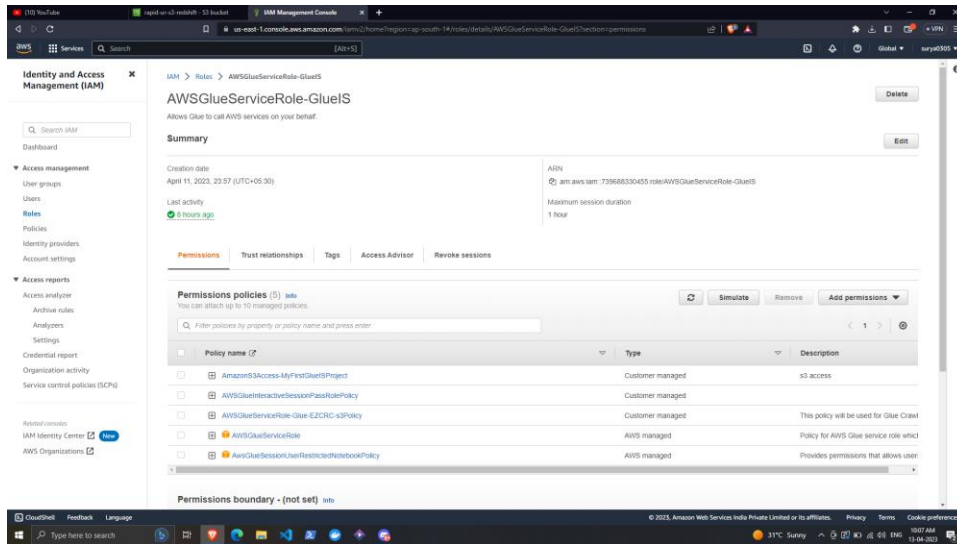
```
{"id": "tt0001702", "is_series": false, "Title": "The Indian Maiden's Lesson",  
"Release_Date": "1911-04-22"}
```



For the Glue job I had created a role **AWSGlueServiceRole-GlueIS**

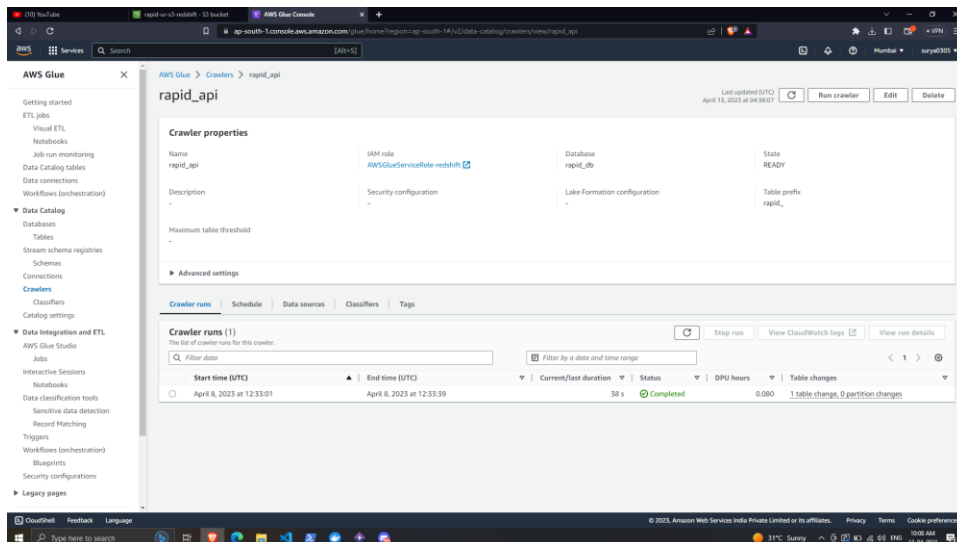
The following policies attached are

- AmazonS3Access-MyFirstGlueISProject
- AWSGlueInteractiveSessionPassRolePolicy
- AWSGlueServiceRole-Glue-EZCRC-s3Policy
- AWSGlueServiceRole
- AwsGlueSessionUserRestrictedNotebookPolicy

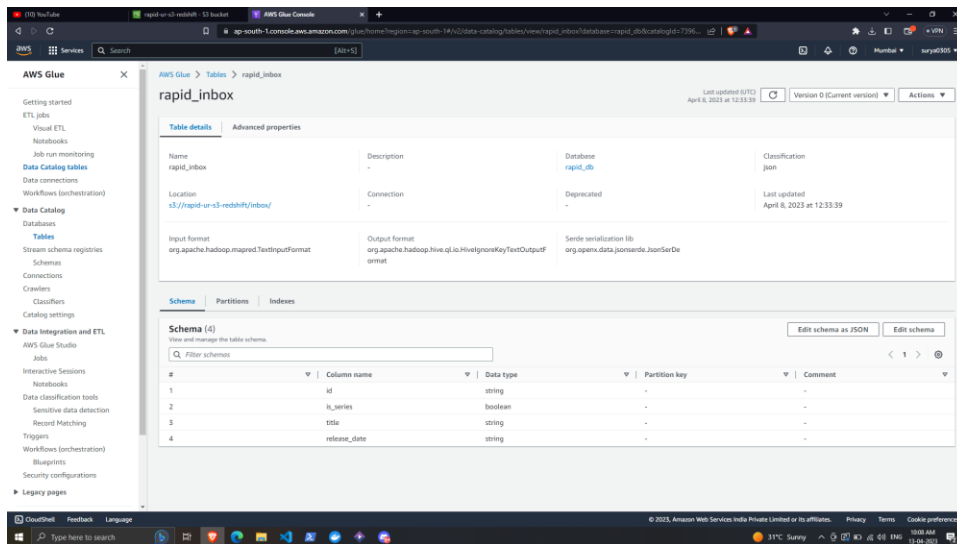


## Creating an AWS Glue crawler

I had created a crawler in Glue to store the metadata info in Glue Catalog as shown below

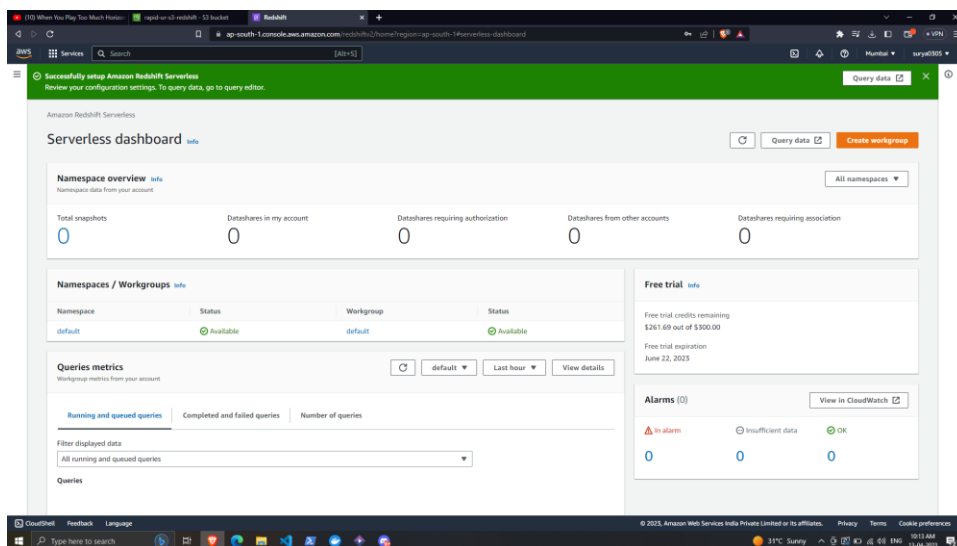


After successful crawler run the metadata is stored in **rapid\_api** Database and in **rapid\_inbox** table

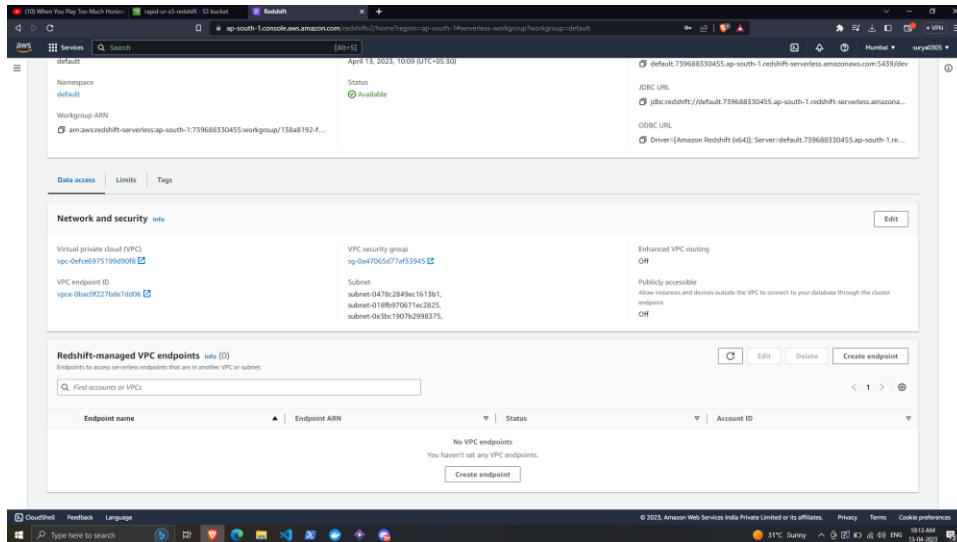


## Creating a Redshift Serverless Cluster

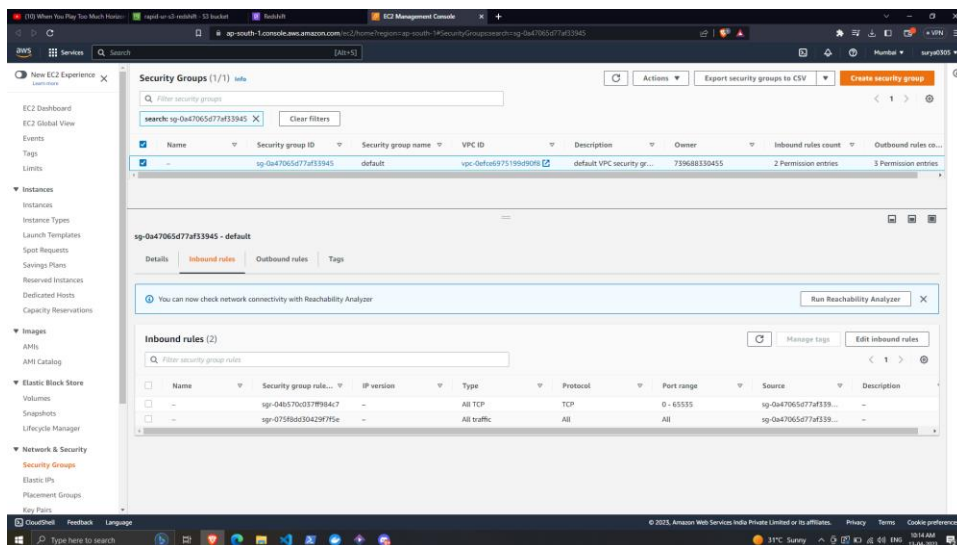
Now create a serverless redshift cluster in which mentioning username and password of our choice as they will be used in GLUE-REDSHIFT connectivity.



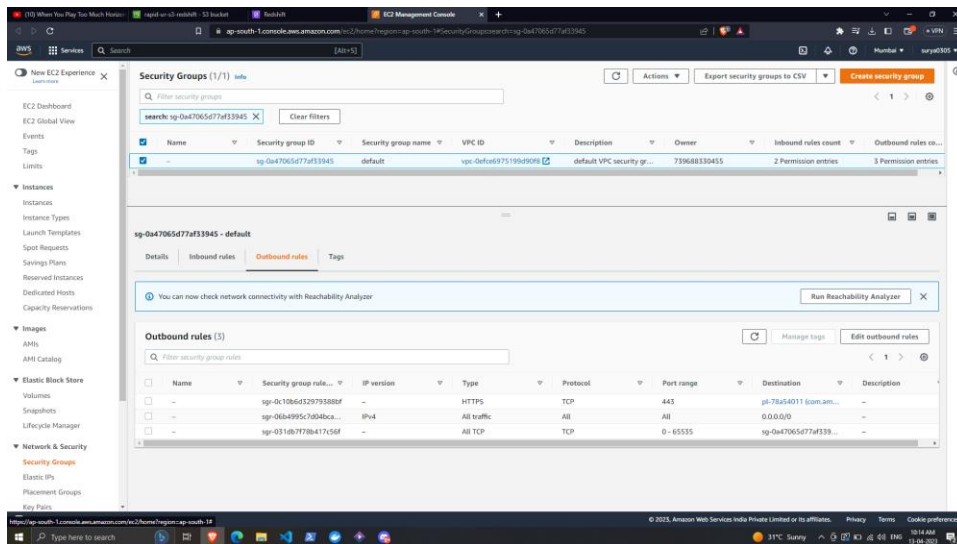
Open the workgroup of the serverless cluster where the security group is mentioned by clicking it, we will be redirected to EC2 instance window.



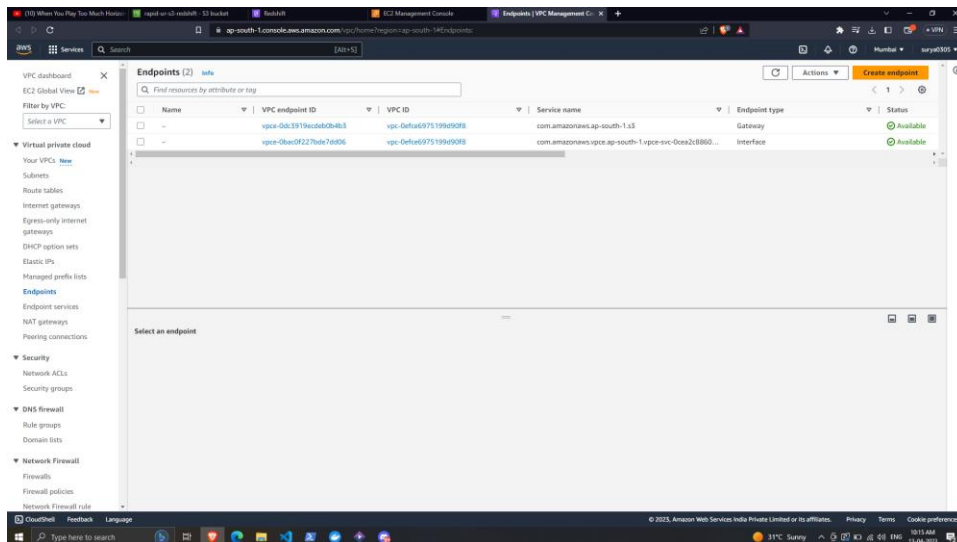
Now set up the inbound rule to allow all traffic from TCP and this rule should be a self-referring rule as shown below.



Again, setup the outbound rule will all TCP and HTTPS and both rules are self-referring rules as shown below.



Now to create a HTTPS outbound rule which would be connecting to S3, create an endpoint to point towards S3 Gateway



The Managed prefix for the endpoint will be used by the HTTPS outbound rule

EC2 Management Console

Managed Prefix Lists | VPC Man...

Services

Search

APIs

VPC dashboard

EC2 Global View

Filter by VPC

Select a VPC

Virtual private cloud

Your VPCs

Subnets

Route tables

Internet gateways

Egress-only Internet gateways

DHCP option sets

Elastic IPs

Managed prefix lists

Endpoints

Endpoint services

NAT gateways

Peering connections

Security

Network ACLs

Security groups

DNS firewall

Rule groups

Domain lists

Network Firewall

Firewalls

Firewall policies

Managed prefix lists (4)

Filter prefix lists

Prefix list ID

Prefix list name

Max entries

Address family

State

State message

Version

Prefix

pl-0419ec96614c525

com.amazonaws.global.groundstation

-

IPv4

Create-complete

-

-

2011.2017

pl-66a7420f

com.amazonaws.ap-south-1.dynamodb

-

IPv4

Create-complete

-

-

2011.2017

pl-78a54011

com.amazonaws.ap-south-1.s3

-

IPv4

Create-complete

-

-

2011.2017

pl-9a4247f5

com.amazonaws.global.cloudfront.origin-facing

-

IPv4

Create-complete

-

-

2011.2017

Select a prefix list above

© 2023 Amazon Web Services India Private Limited or its affiliates.

Privacy

Terms

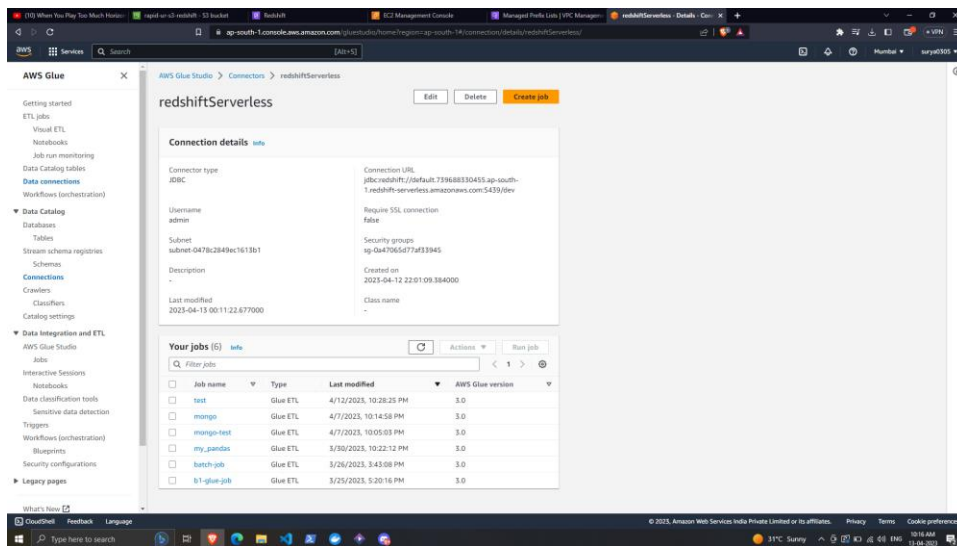
Cookie preferences

## Creation of Glue Job

First, before creating a job we need to have a connection to our redshift cluster.

In AWS Glue open connection and use JDBC option put the credentials of username and password of the cluster.

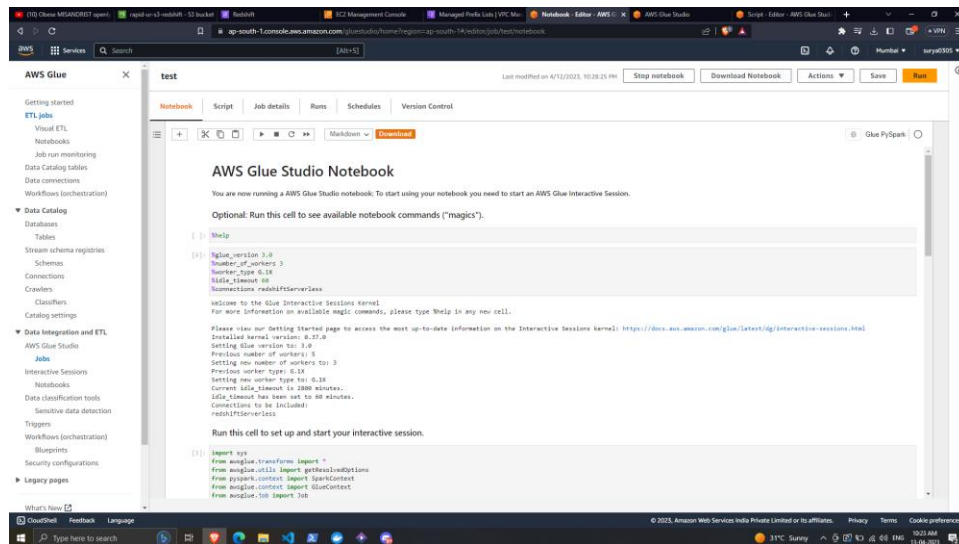
After giving the details final details would look like as shown below.





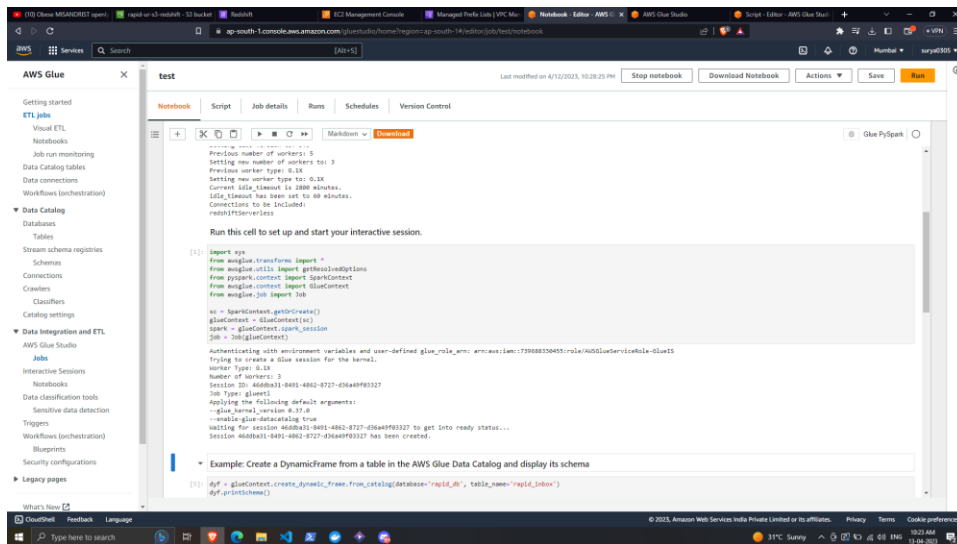
Now under the Job section, create a job using Notebook as shown below.

Use the following IAM Role **AWSGlueServiceRole-GlueIS**

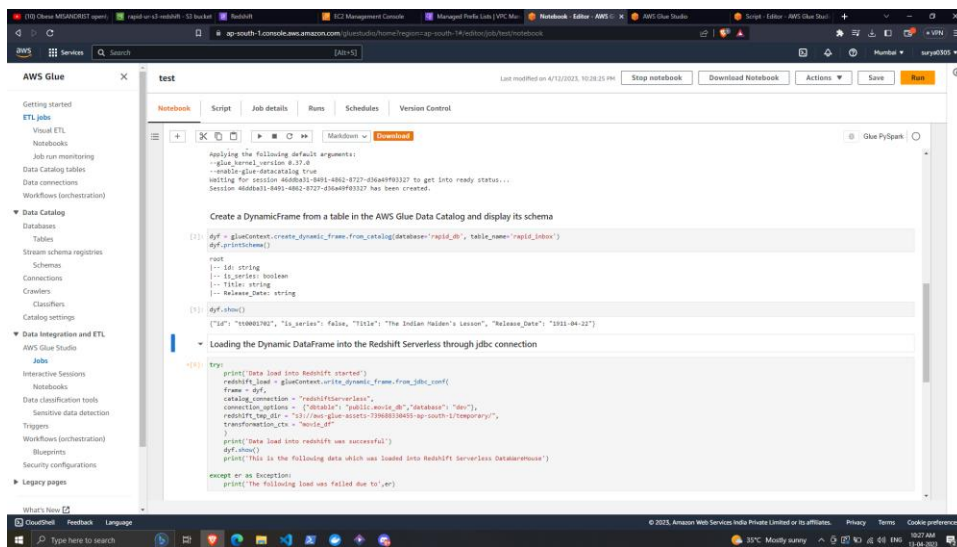


Execute the following blocks as shown below

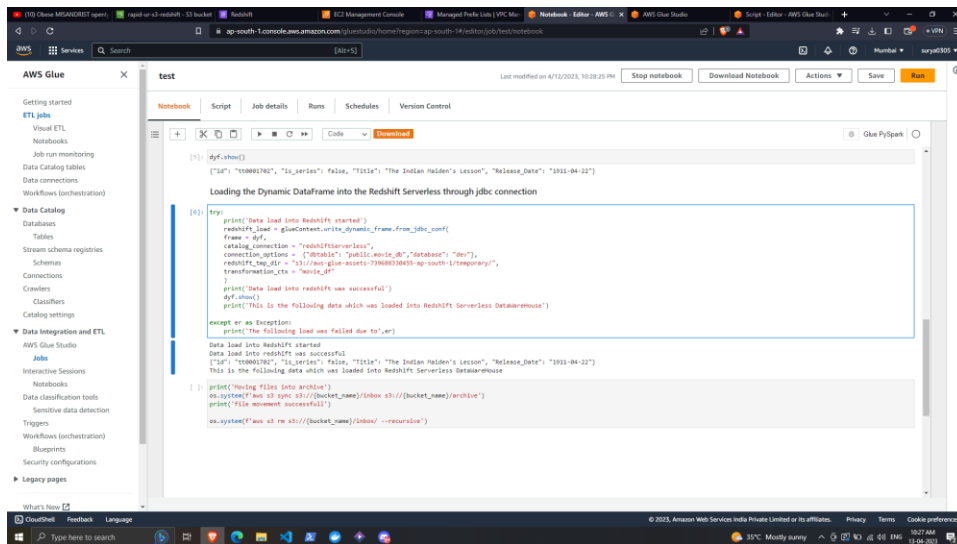
First, I am checking whether the redshift cluster is connecting or not.



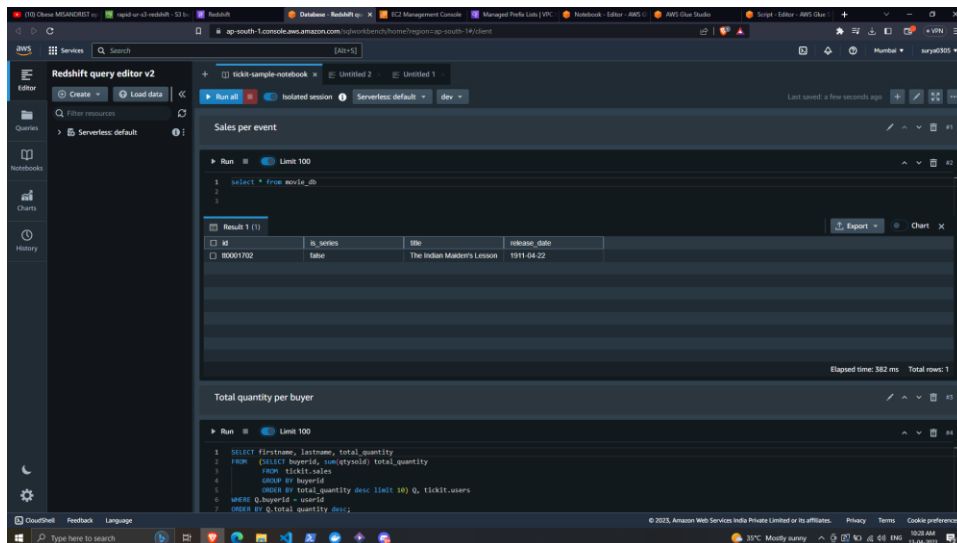
Once the connection is established, then I am reading data from catalog table



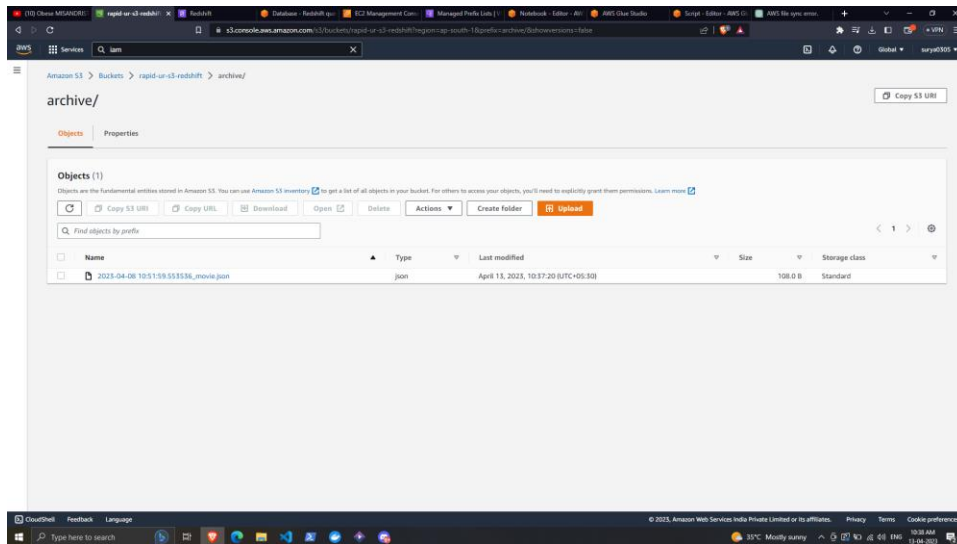
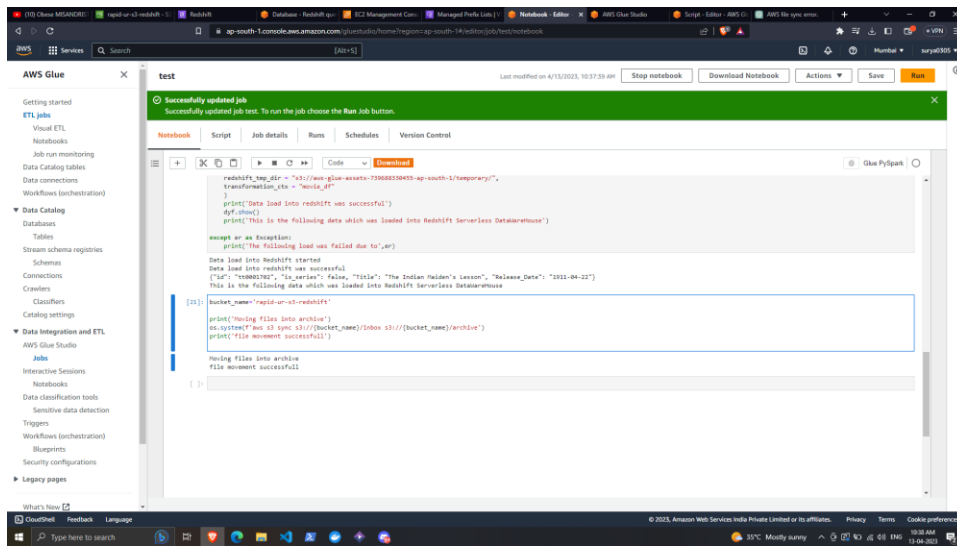
After successfully reading from the table and verifying the schema, now I am connecting to the cluster and loading the data into it. By this method we can directly load it without creating the table.



After the load is completed, the data is visible in Redshift.



After the DB load is completed the JSON files in /inbox folder is moved to /archive folder and the files in /inbox folder is removed.



## Creating a schedule for Lambda Function

Using Event-Bridge I had scheduled the lambda function to trigger daily around 10:30 AM IST

The screenshot displays the Amazon EventBridge console interface. On the left, a navigation pane lists various services and resources. The main content area shows the configuration for a schedule named 'rapid-url'. The 'Schedule detail' section includes fields for the schedule name, status (Enabled), description, schedule ARN, schedule group name, start time, end time, execution timezone, and creation/modification dates. Below this, the 'Schedule' tab is active, showing a cron expression field with the value '30 10 \* \* ? \*'. A 'Copy cron expression' button is present. To the right, a 'Cron expression' sidebar provides an explanation of cron expressions and a 'Learn more' link. The bottom of the screen shows the Windows taskbar with the date and time as 11:49:30.

Amazon EventBridge Scheduler

ap-south-1-console.amazonaws.com/scheduler/home?region=ap-south-1#/schedules/default/rapid-url

Amazon EventBridge Schedules > rapid-url

Disable Edit Delete

**Schedule detail**

Schedule name	rapid-url	Status	Enabled	Schedule start time	-	Flexible time window	-
Description	-	Schedule ARN	arn:aws:scheduler:ap-south-1:79968830455:schedule/default/rapid-url	Schedule end time	-	Created date	Apr 13, 2023, 10:42:49 (UTC+05:30)
Schedule group name	default	Execution timezone	Asia/Calcutta	Last modified date	Apr 13, 2023, 10:42:49 (UTC+05:30)		

**Schedule**

Cron expression [info](#)

30 10 \* \* ? \*

Minutes Hours Day of month Month Day of week Year

Copy cron expression

Next 10 trigger dates

Date and time are displayed in the selected time zone for which this schedule is set in UTC format.

6.9 "Week, Nov 9, 2022 09:00 (UTC -08:00)

Fri, 14 Apr 2023 10:30:00 (UTC +05:30)

Sat, 15 Apr 2023 10:30:00 (UTC +05:30)

**Cron expression**

A cron expression creates a fine-grained recurring schedule that runs at a specific time of your choosing. With cron-based schedules, you have more control over when and how often your schedule runs. Use cron expressions when you need a customized recurrence schedule that is not supported by one of EventBridge Scheduler's rate expressions. For example, you can create a cron-based schedule that runs at 8:00 a.m. PST on the first Monday of every month.

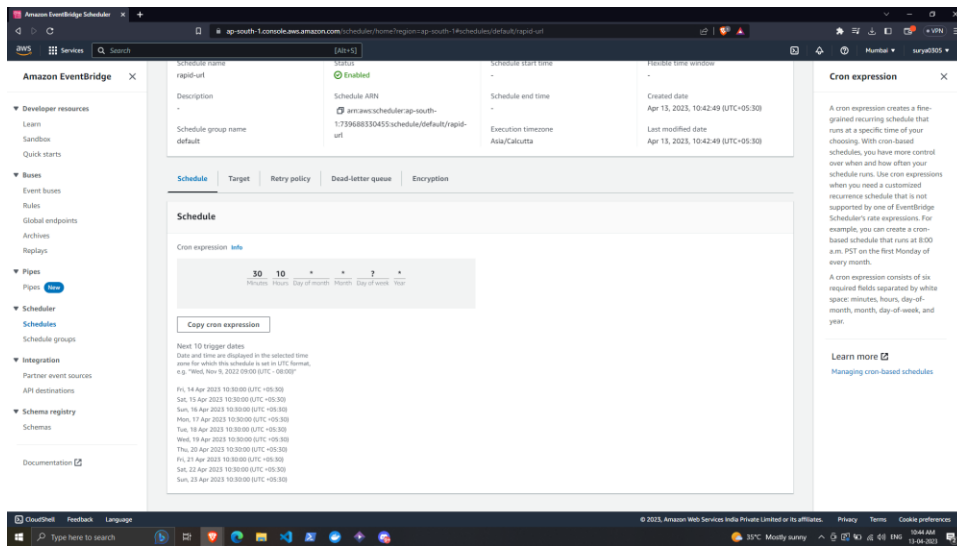
A cron expression consists of six required fields separated by white space: minutes, hours, day-of-month, month, day-of-week, and year.

[Learn more](#)

[Managing cron-based schedules](#)

© 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

33°C Mostly sunny 11:49:30



Similarly, created a rule to generate to trigger the Glue-Job

