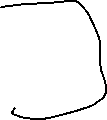
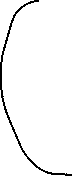
**Incremental Telecom data ingestion pipeline in AWS**



**Upstream Client**



**partitions**



s3\_Data\_lake GLUE RDS (Target)



**partitions read only**



**new s3 file**



**Filter prepaid Ingest**

**S3(Source) sales\_Db**

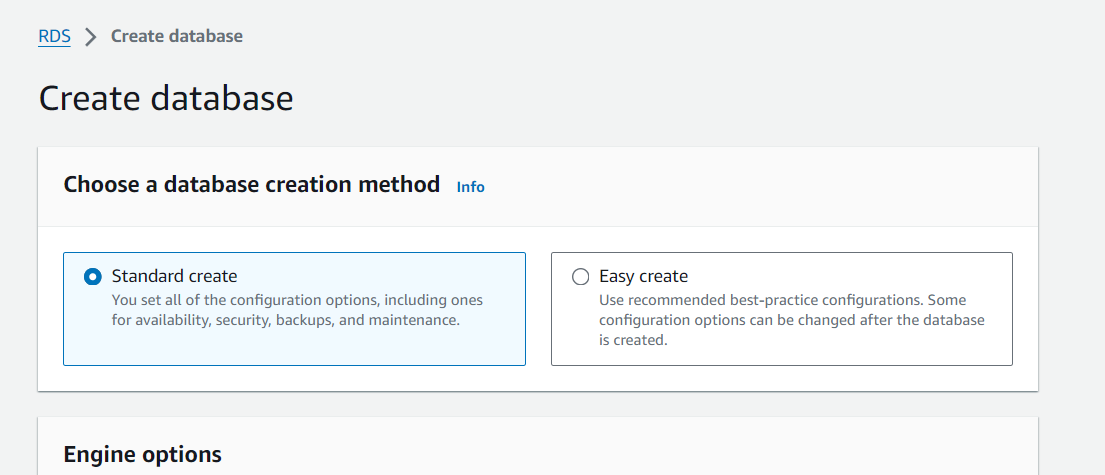


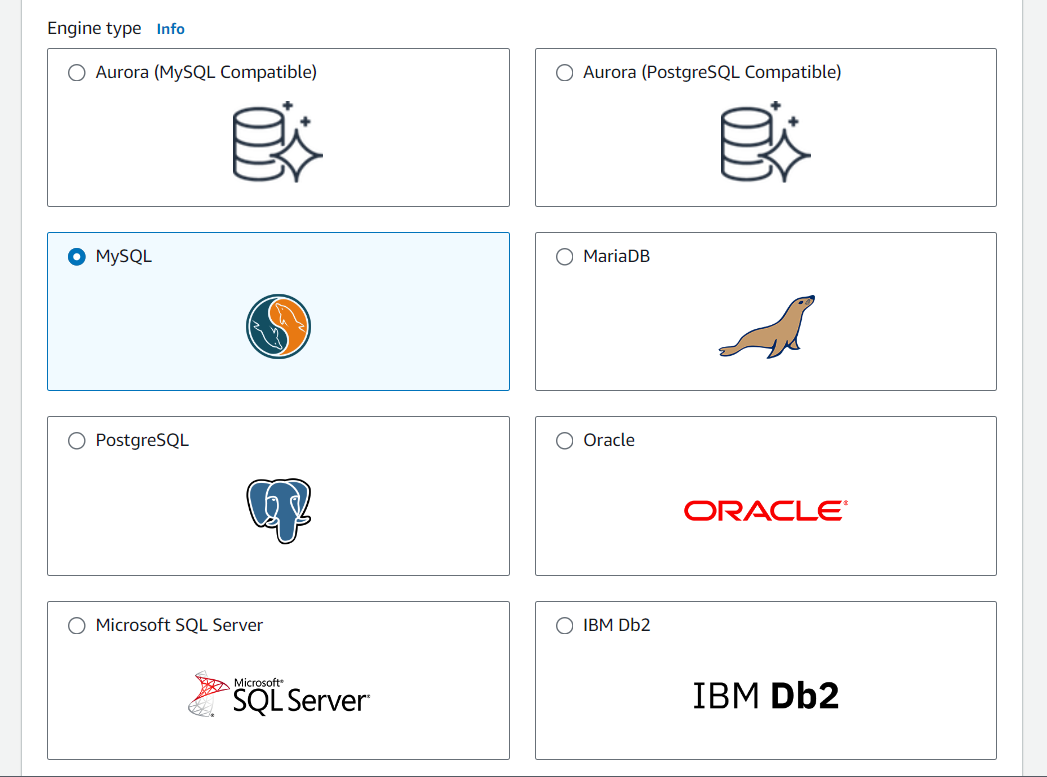
**Project Description:**

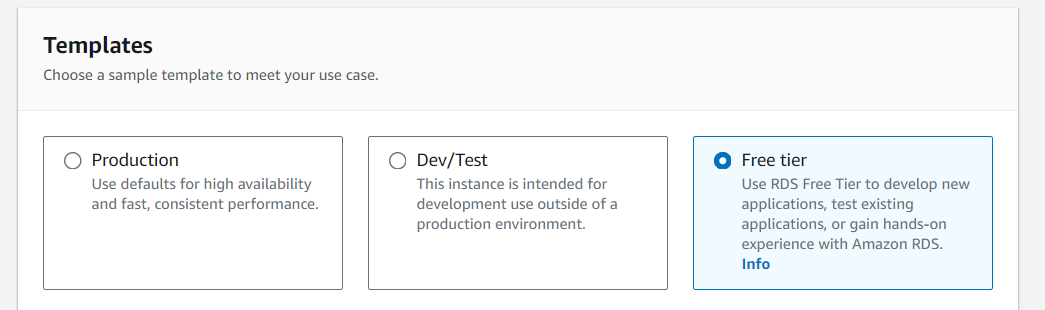
In this s3 datalake is source here and in s3 we created bucket assume that there some service or upstream client that is giving raw data to process , it is sending csv files, daily data to s3 bucket to process , you are getting daily data in form of partitions in s3 and for partitions columns they are using date partitions are named date=2023=08-21,date=2023-08-22,date=2023-08-23, we are creating hive style of parttion in that u are getting csv file , partition is like your folder struture and here our target is RDS and it is telecom data . inside RDS we are need to creating sales db and customer\_subscription table and Incremental pipeline means whatever daily file is receiving we are just processing that file and dump in target table in RDS Here glue works on middle between source and target or destination and In glue job we filter only prepaid customer (data) and ingest to RDS , and from datalake read only new s3 file .

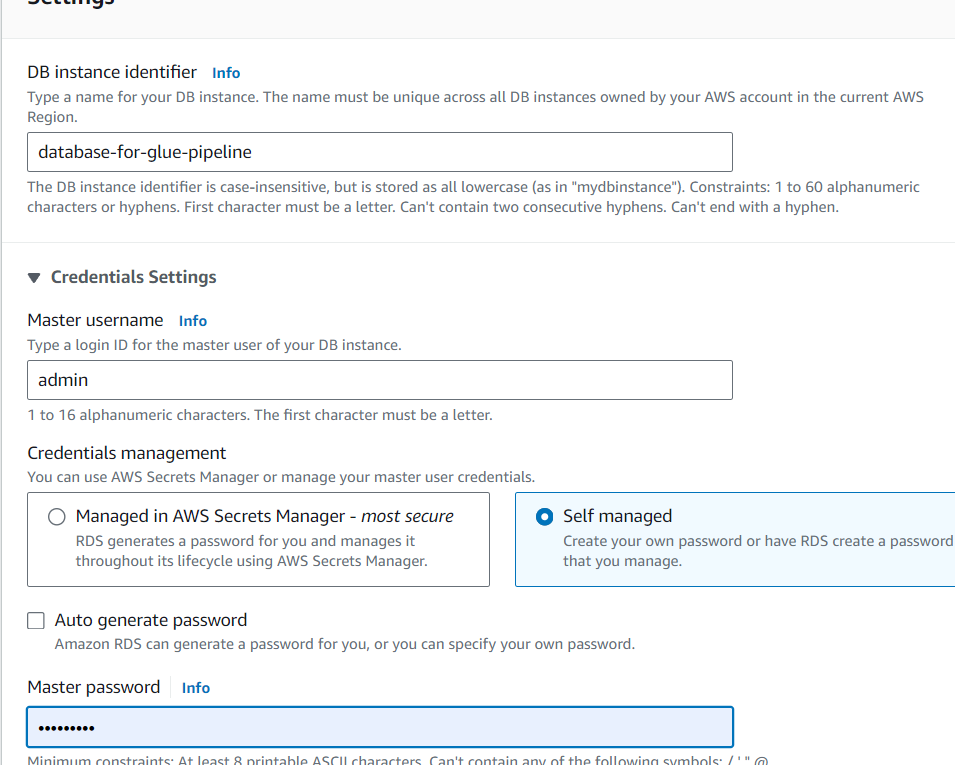
🡪In this the s3\_data\_lake(i.e s3 bucket) is the Source which receives daily a csv file from upstream client some client is sending daily csv files to s3\_data\_lake and in this Glue is used in the middle to store metadata of both S3 and RDS and RDS is the destination . and here we are filtering only customer subscription table in that Records 🡪 prepaid (plantype) ] through glue ETL and sending those filtered records to the RDS Target .

* First we will create Database In AWS RDS, we selected standard create and MYSQL database for our project

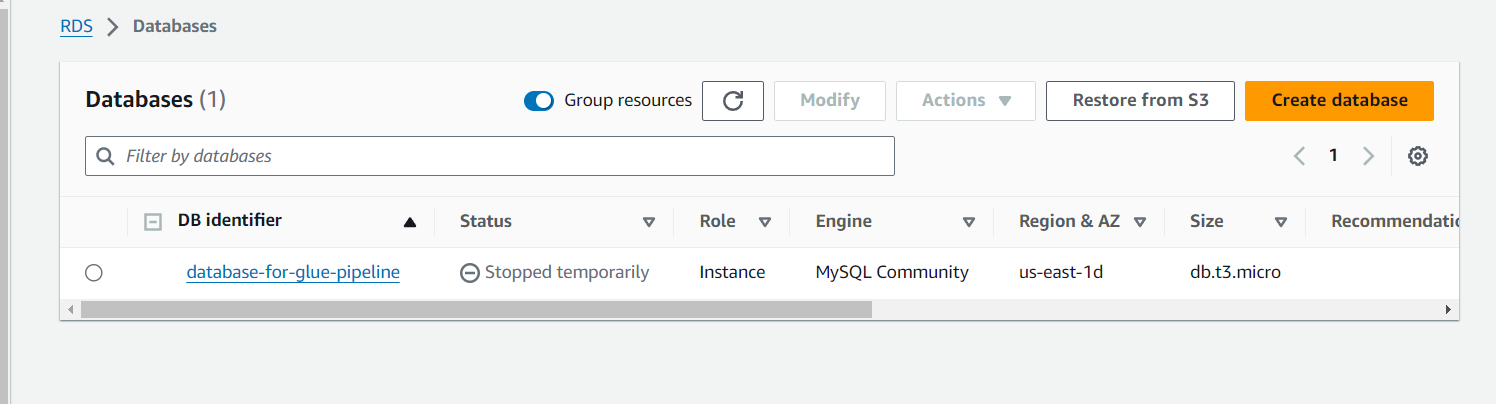




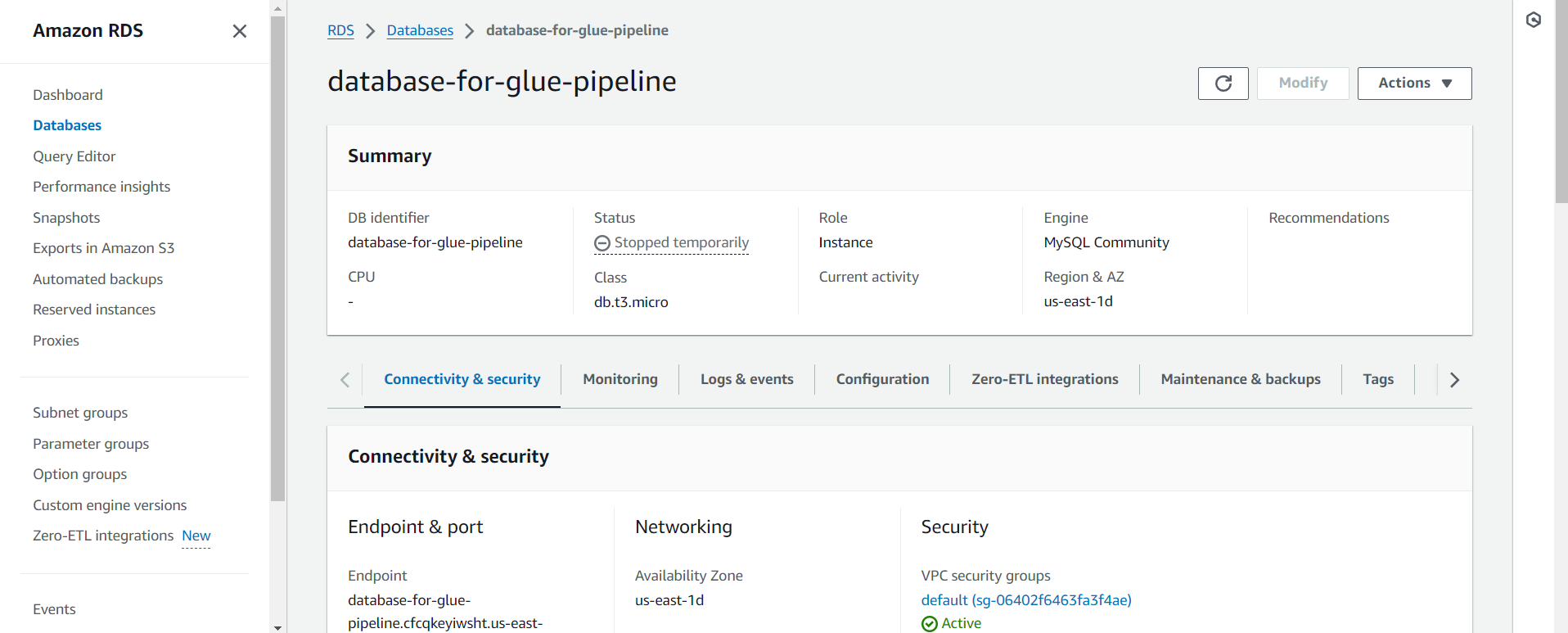




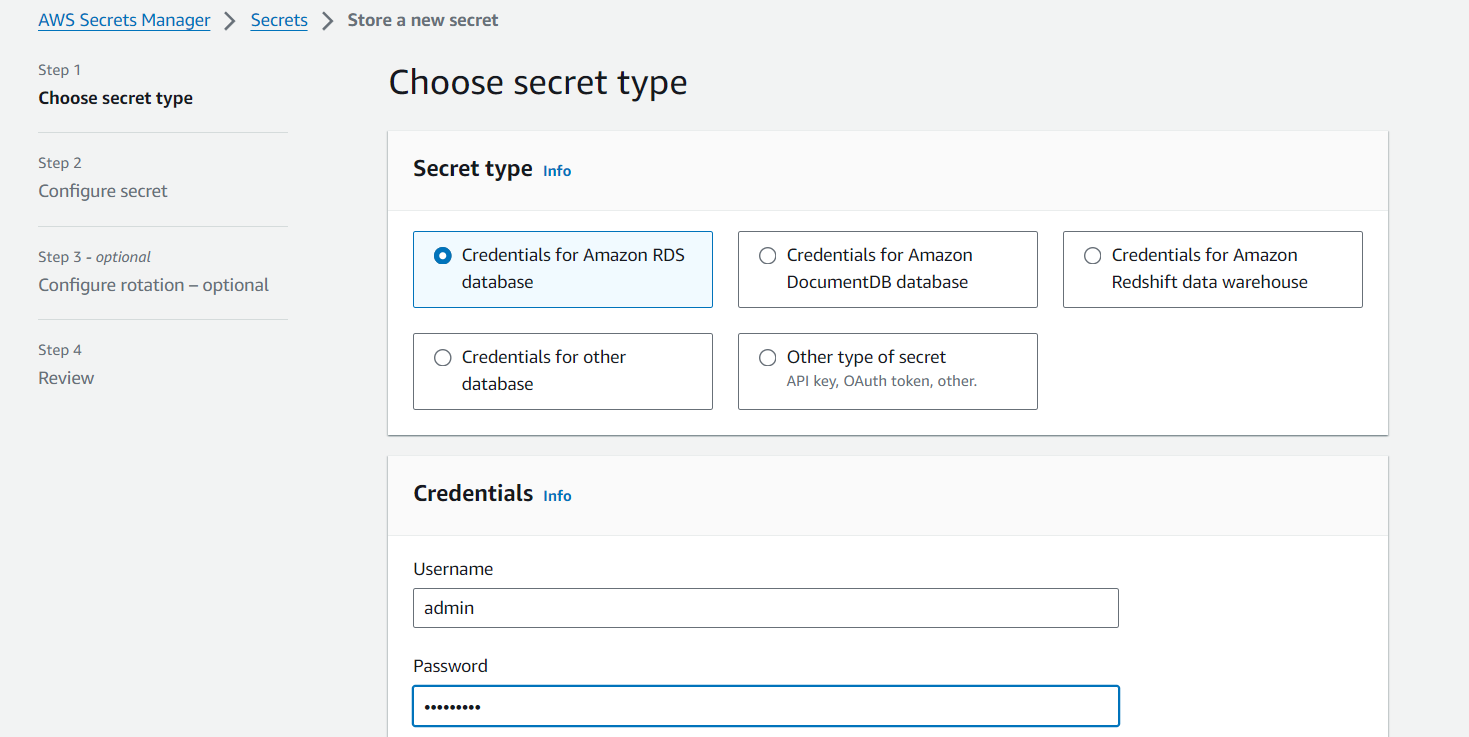
* And While creating need to give username and password for and we need give public access yes because we need to connect through Cmd and select required Templates(production/Dev/free-Tier) we used free-tier , and storage as per our use next create

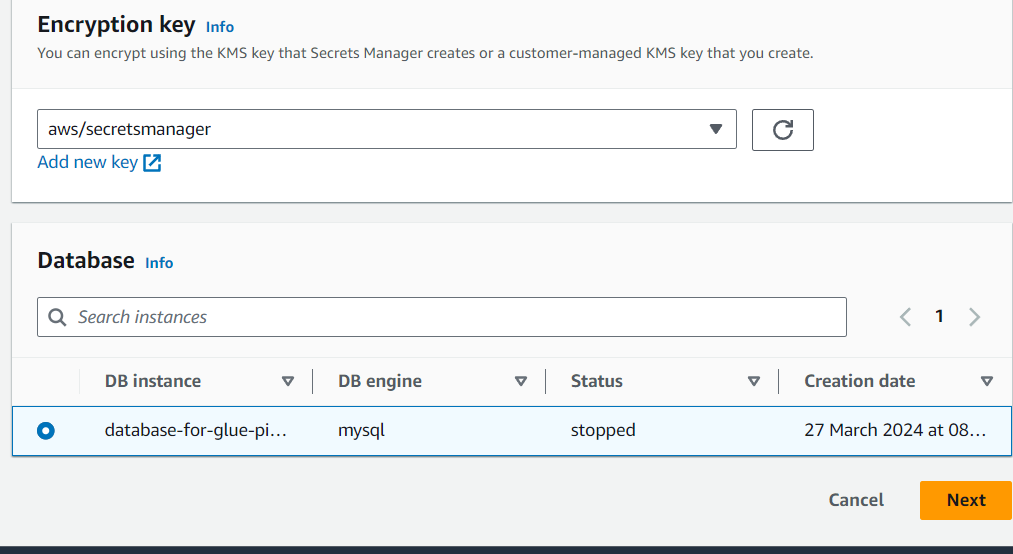


We can see details

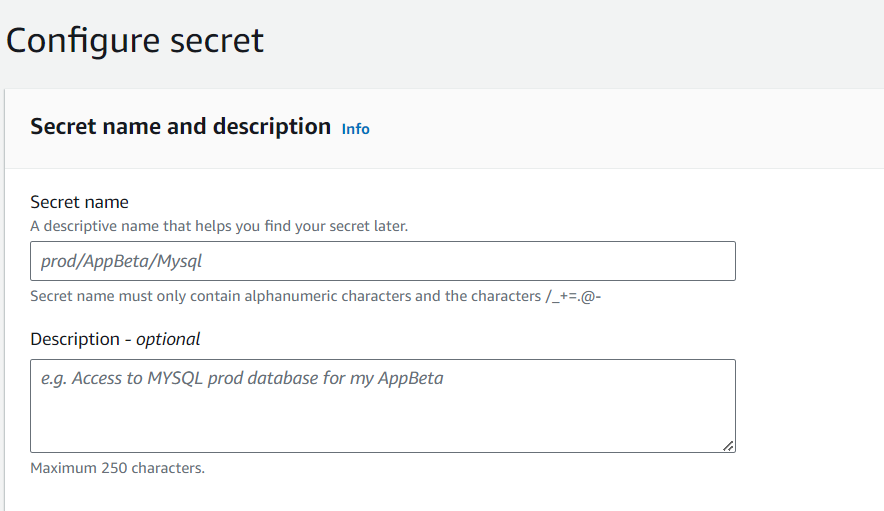


* Next when we connect to DB through programmatically then we need to mention host, port, username and password then they are visible(exposed) to public.
* So to avoid that we will store our credentials in some encrypted form using AWS Secret manager
* And then programmatically using boto3 we will fetch them during runtime , in this way credentials are not exposed to public
* Next Create secret in AWS Secret Manager
* This is known for managing secrets
* Click on store new secret select secret type

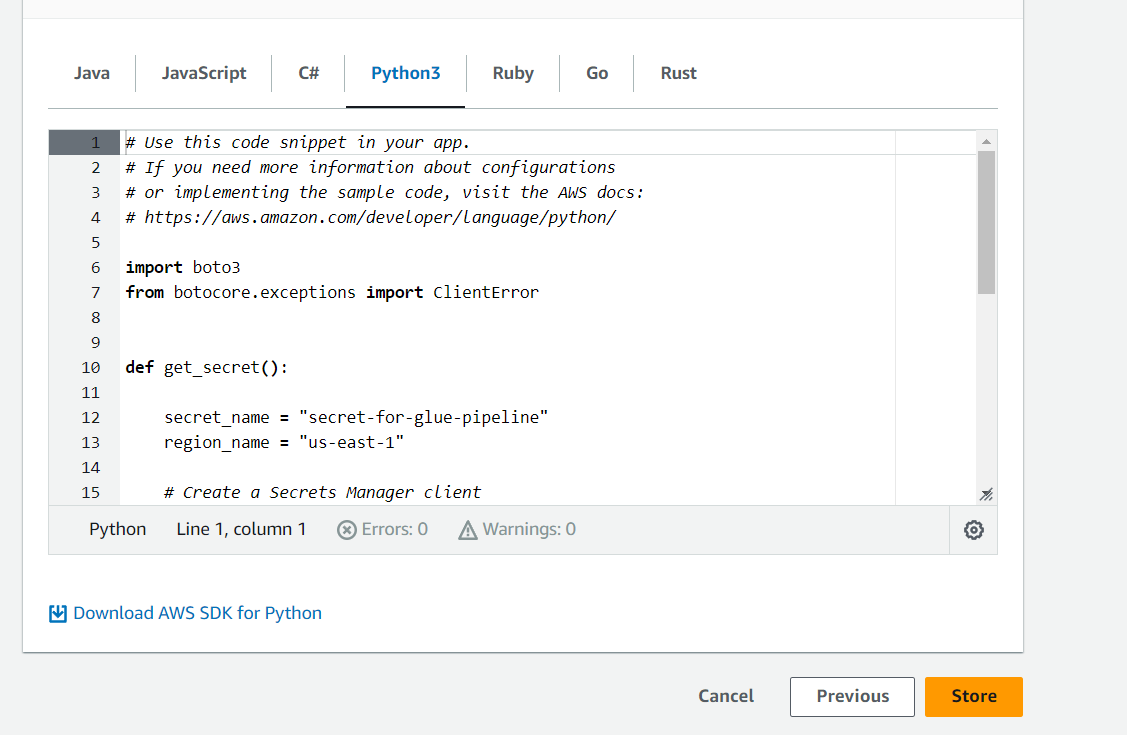


* Enter username and password of RDS DB we created select our DB and 
* name it

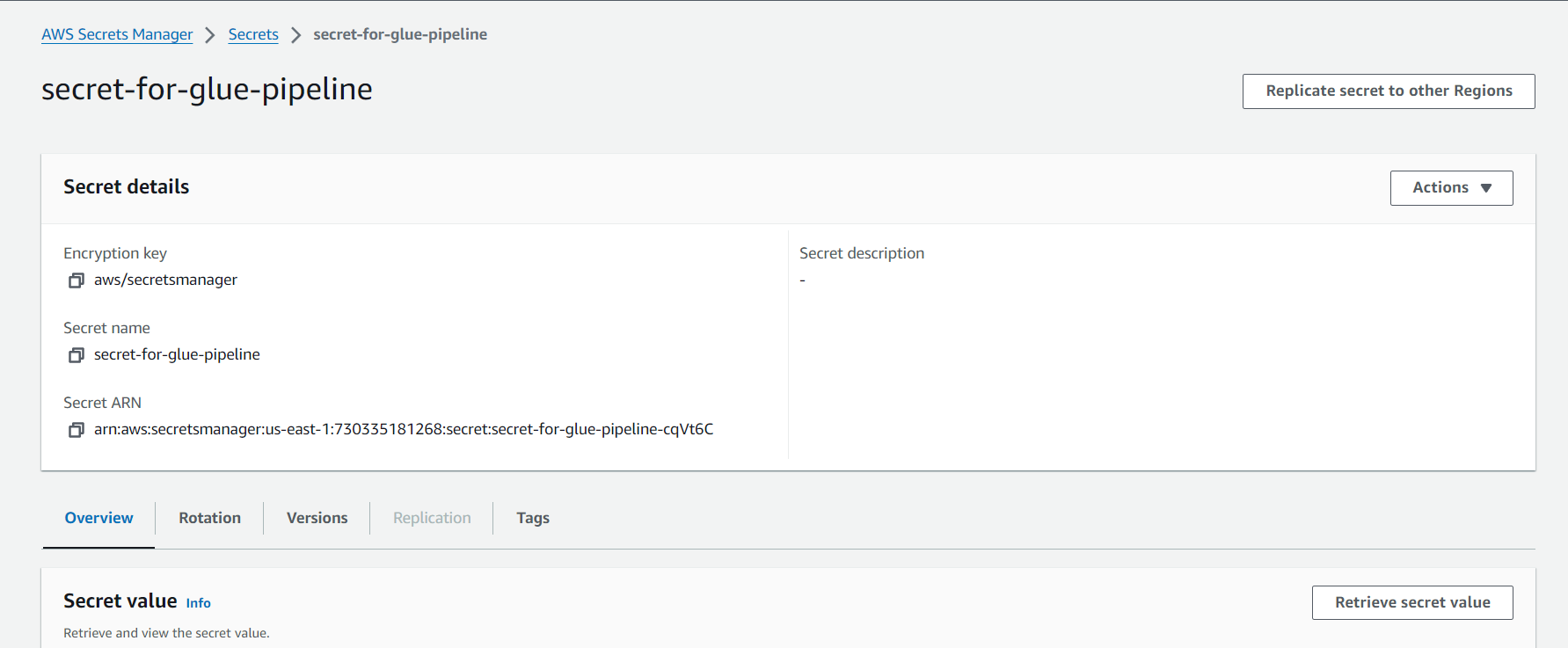
EX; (secret-for-glue-pipeline) 🡪 next🡪store



Next store here we can see it give code automatically which can be used in program to get secret

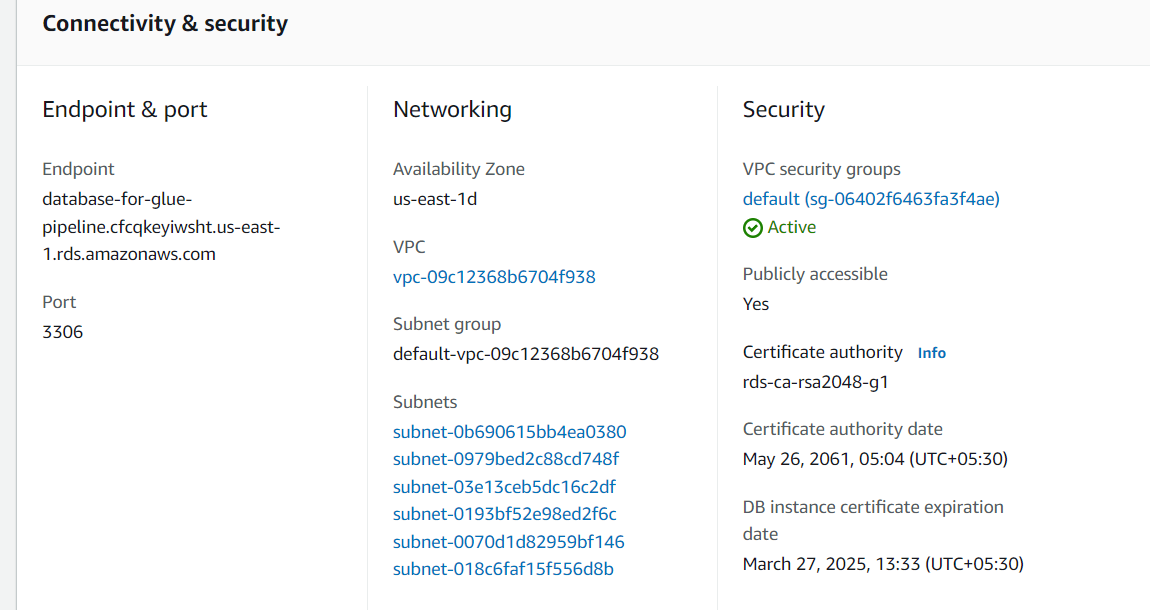


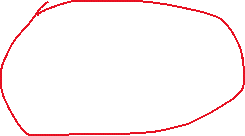
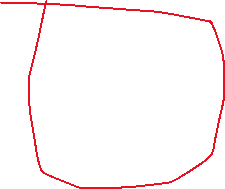
After creating details we can see



Next

* Setup Mysql on windows command if not there
* Next in through Cmd access the RDS
* See details below screenshot

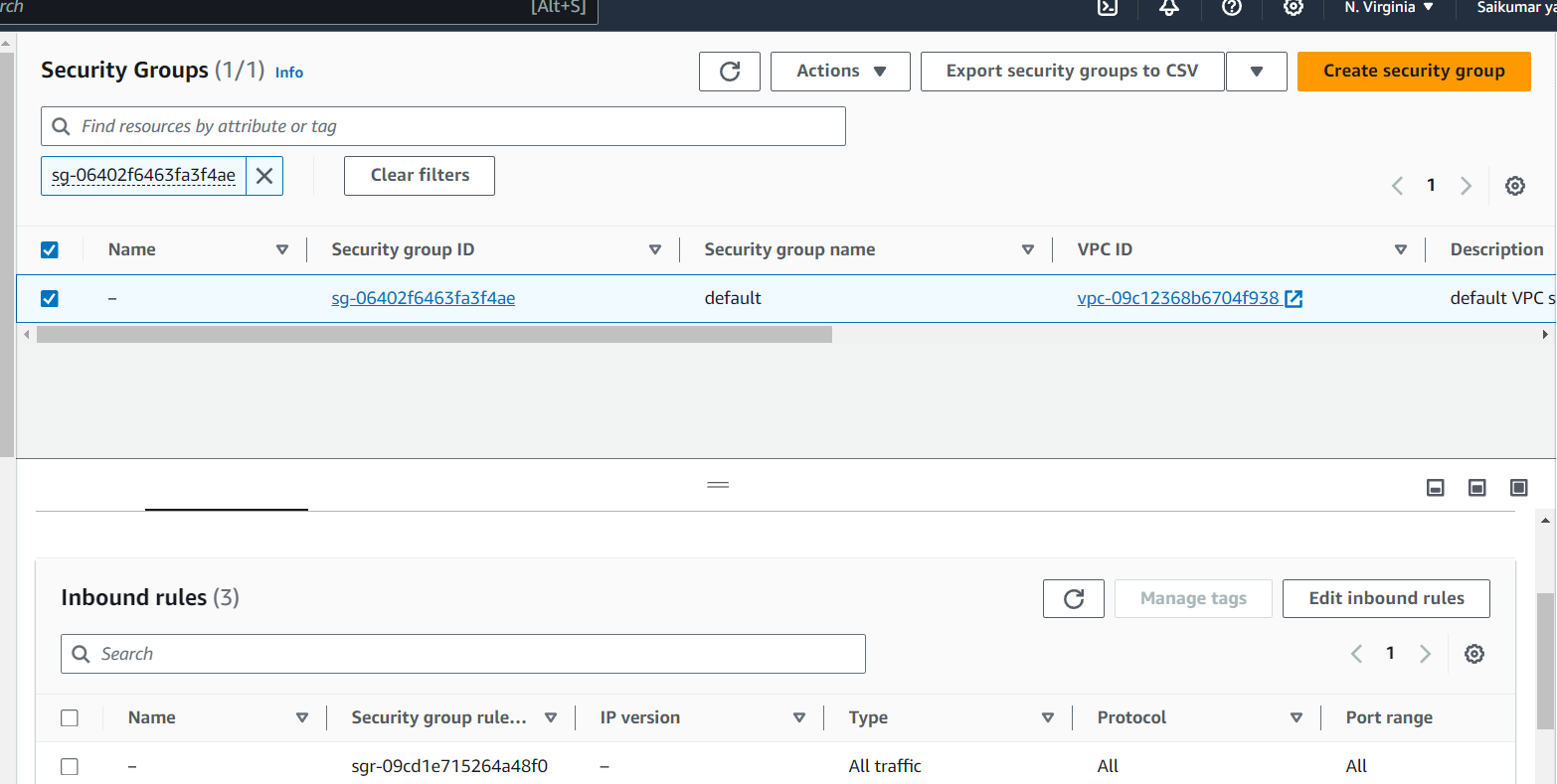




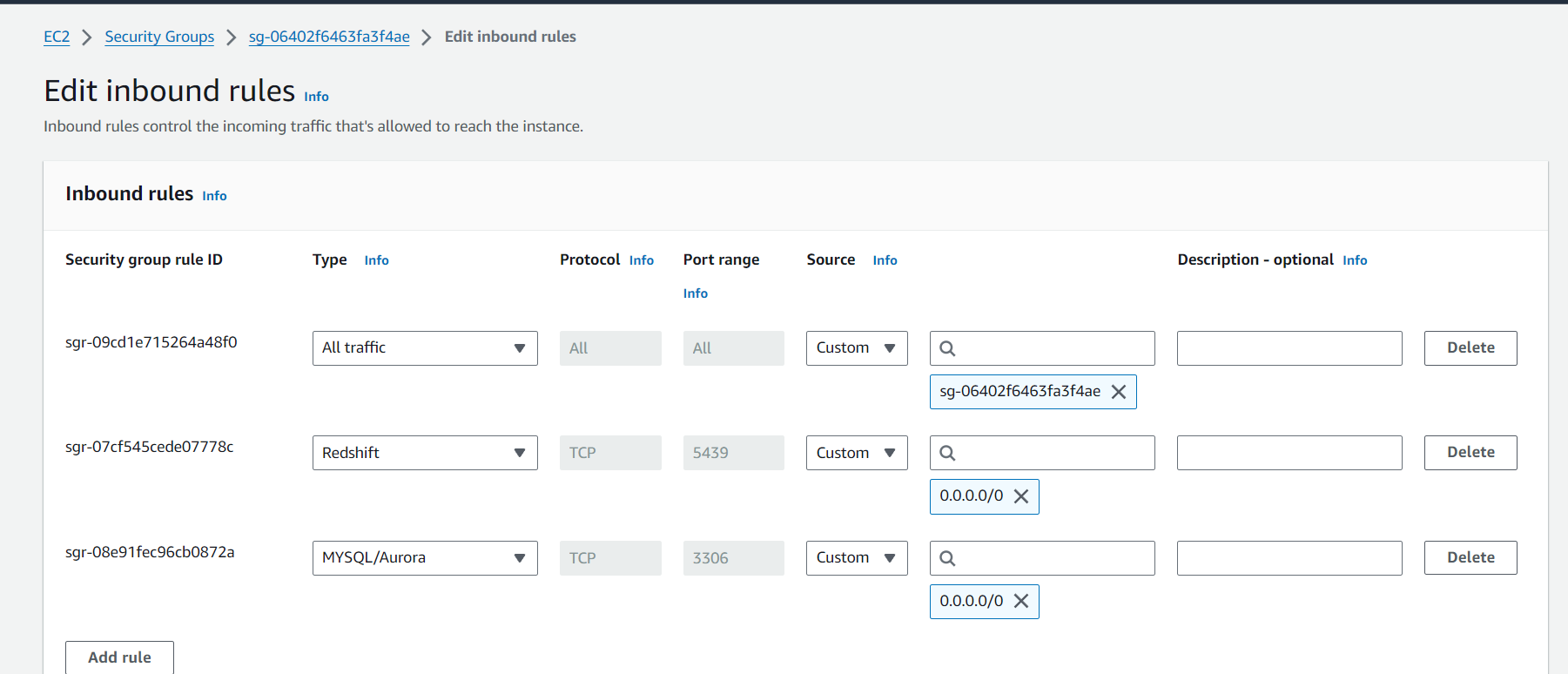
Before that we need to Open connectivity & Security select VPC security groups and we need to allow these to access mysql /aurora in security group .

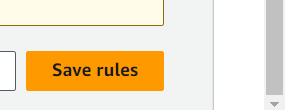


* So first Open Vpc security groups then select security group below Edit inbound rules



Add the redshift and mysql rules and save

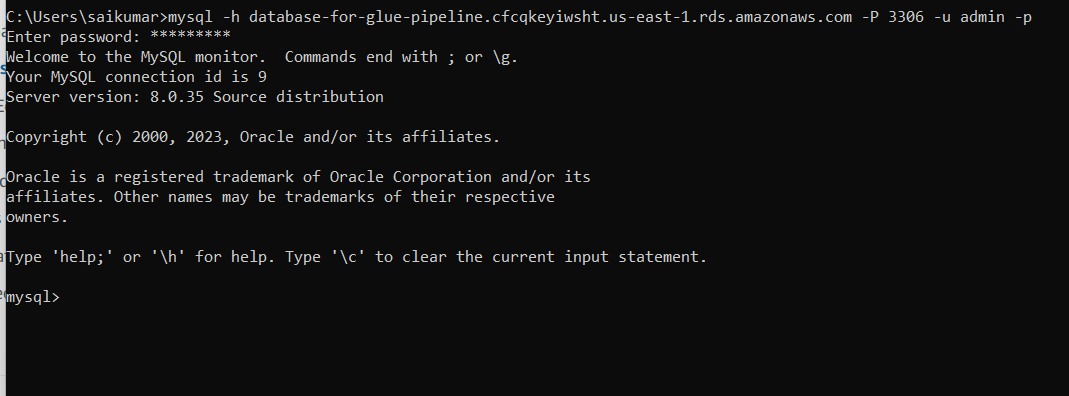




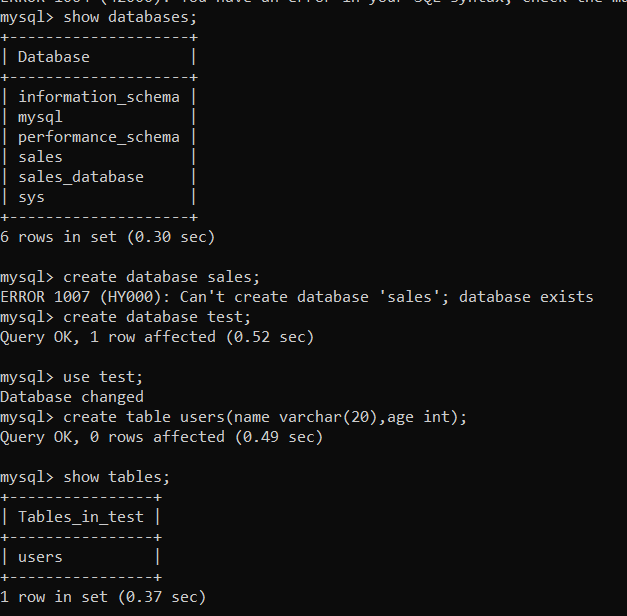
Then in CMD do following as per below screen shot to connect to RDS through CMD line

Here -h stands for hostname and -P stands for port and -u stands for username and -p for password

And we can see it’s connected after entering these



Next we can see databases, create db create table anything trough cmd in RDS



If we want trough programmatically we can use below code i.e connect\_rds.py

import mysql.connector

import boto3

import json

# Initialize a Secrets Manager client

client = boto3.client('secretsmanager', region\_name='us-east-1')

secret\_name = 'secret-for-rds-glue'

def get\_rds\_credentials(secret\_name):

    try:

        # Fetch the secret value

        get\_secret\_value\_response = client.get\_secret\_value(SecretId=secret\_name)

        # Check if the secret uses the Secrets Manager binary field

        if 'SecretString' in get\_secret\_value\_response:

            secret = get\_secret\_value\_response['SecretString']

            secret\_dict = json.loads(secret)

            return secret\_dict

        else:

            decoded\_binary\_secret = base64.b64decode(get\_secret\_value\_response['SecretBinary'])

            secret\_dict = json.loads(decoded\_binary\_secret)

            return secret\_dict

    except Exception as e:

        print(f"Error fetching secret: {e}")

        return None

def connect\_and\_create\_db():

    connection = None

    try:

        credentials = get\_rds\_credentials(secret\_name)

        if credentials:

            print("Fecthed RDS Credentials:")

            username = credentials['username']

            password = credentials['password']

            # Depending on how you've structured your secret, you might need

            # to adjust the keys (e.g., 'username' and 'password') accordingly.

        else:

            print("Failed to fetch credentials.")

        connection = mysql.connector.connect(

            host='db-for-glue-project.cfcqkeyiwsht.us-east-1.rds.amazonaws.com',

            port=3306,

            user=username,

            password=password

        )

        if connection.is\_connected():

            print("Successfully connected to the RDS instance.")

            cursor = connection.cursor()

            # Create a new database

            cursor.execute("CREATE DATABASE IF NOT EXISTS xavierDB;")

            print("Database created successfully.")

            cursor.execute("SHOW DATABASES;")

            print(cursor.fetchall())

        else:

            print("Failed to connect to the RDS instance.")

    except mysql.connector.Error as e:

        print(f"Error: {e}")

    except Exception as e:

        print(f"Error: {e}")

    finally:

        if connection is not None and connection.is\_connected():

            cursor.close()

            connection.close()

            print("Connection closed.")

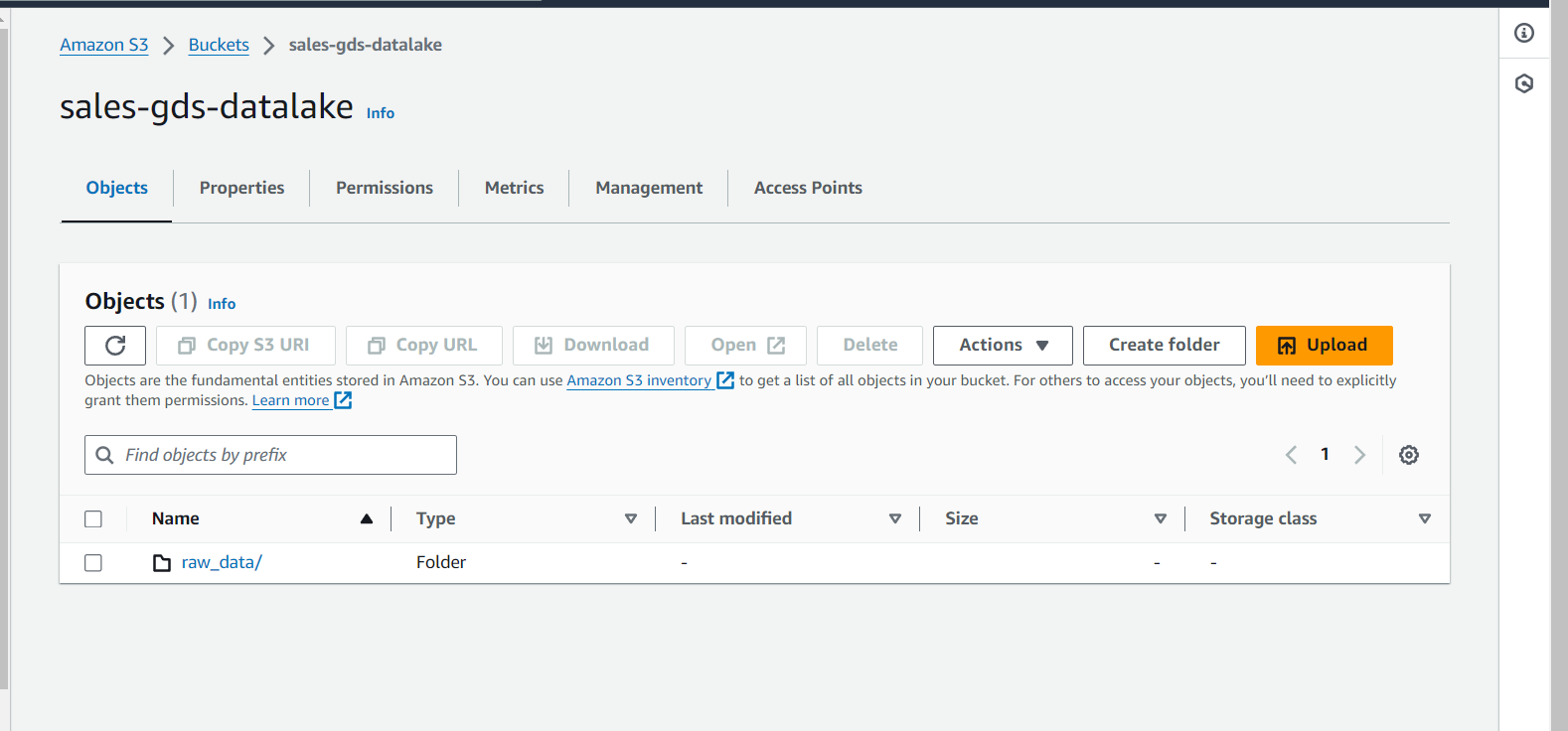
if \_\_name\_\_ == "\_\_main\_\_":

    connect\_and\_create\_db()

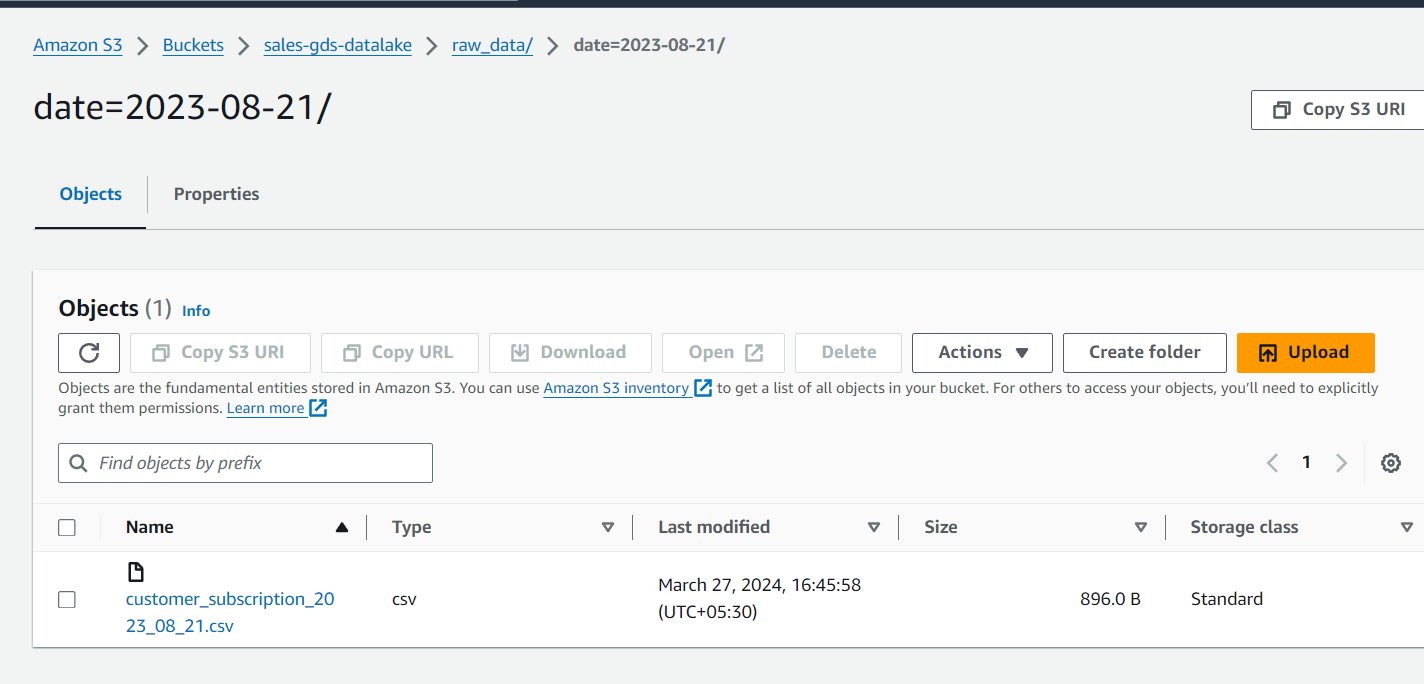
in the terminal run python3 connect\_rds.py and we can see it’s connected to rds and databases in it.

**S3 Bucket**

Next will create S3 bucket and in that we will create folder raw\_data and in that raw\_data again create folder date=2023-08-21 here we upload 1st csv file of same date



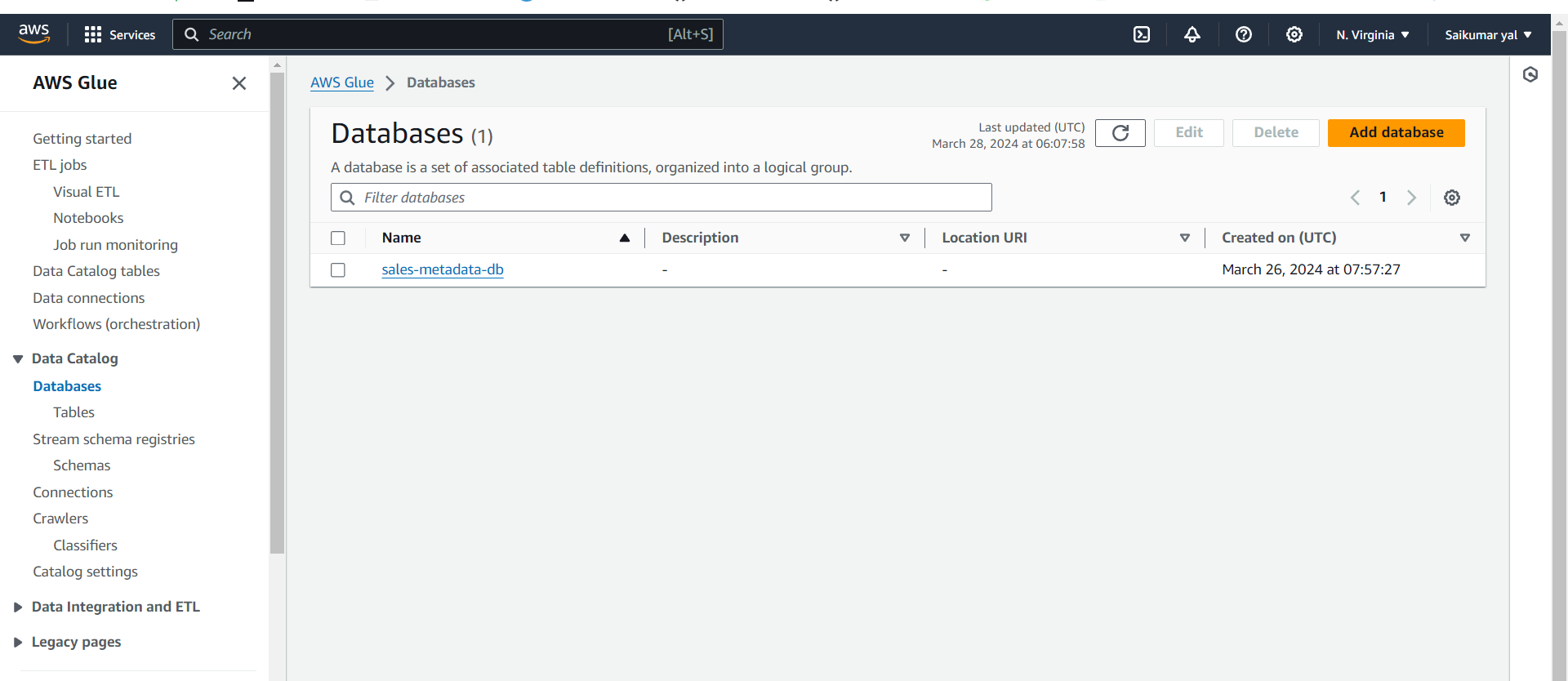
Here we upload csv in below path



**Next Glue part 🡪**

Next open AWS Glue and in that create database for storing metadata

Ex: sales\_metadata\_db create database by clicking on left pane

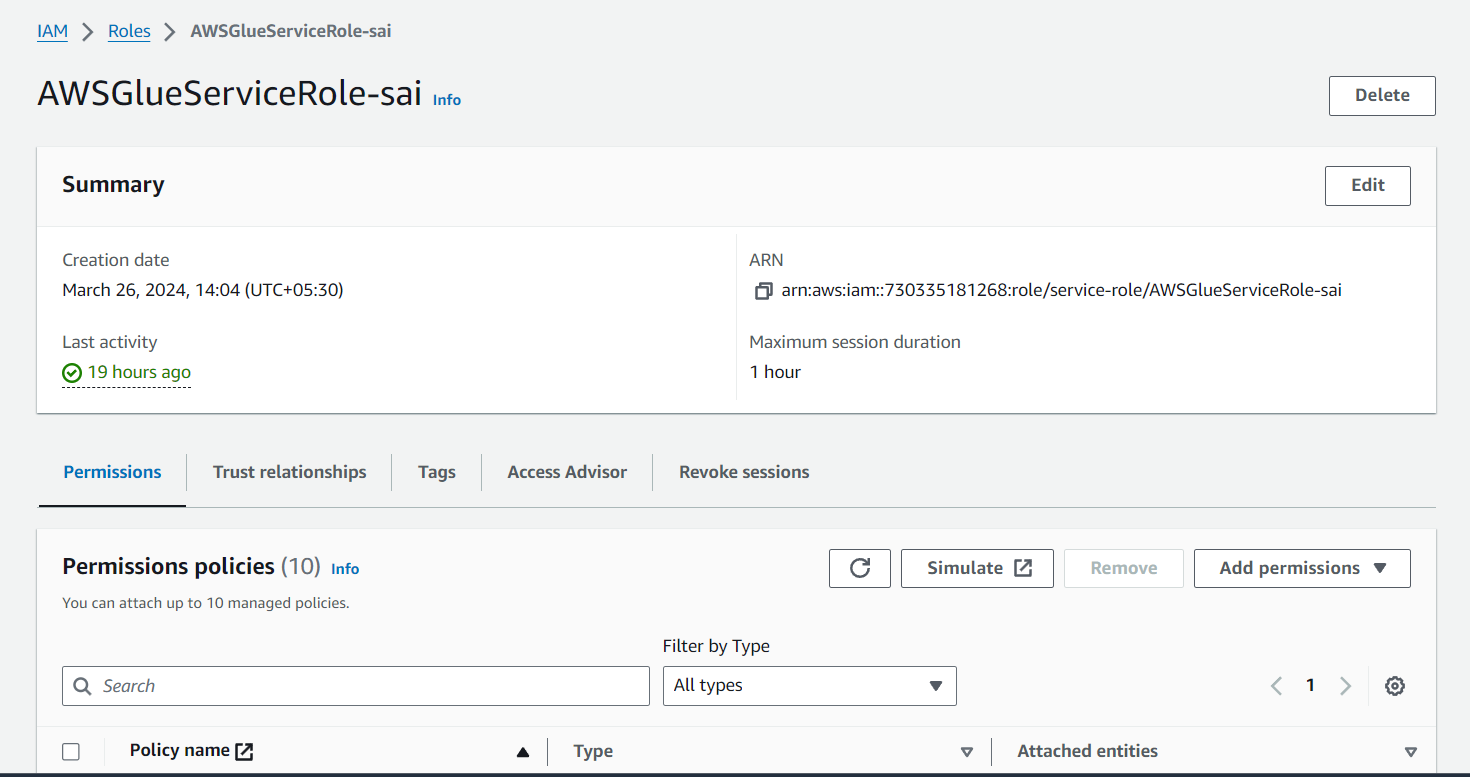


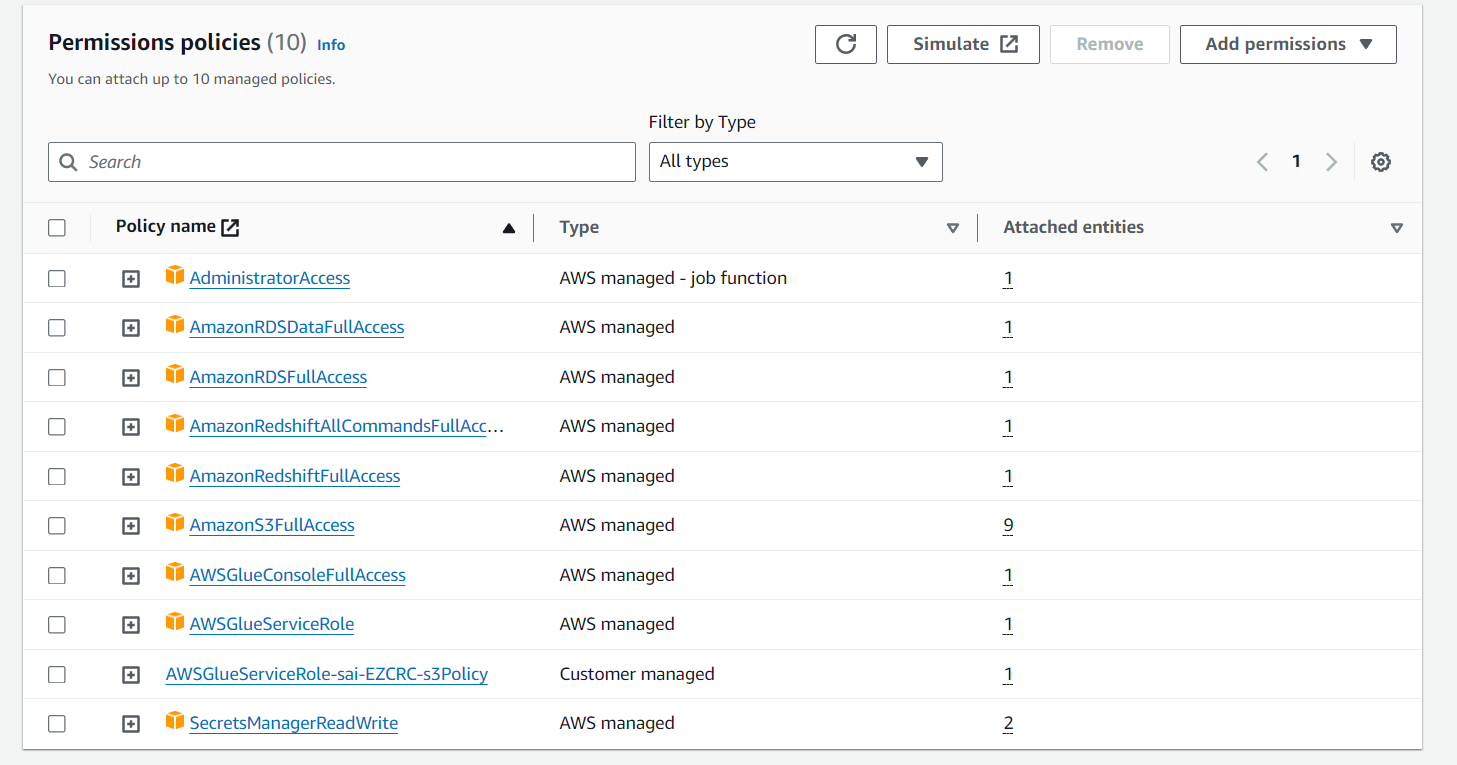
**Crawler**

Next will create crawler for crawling S3 source



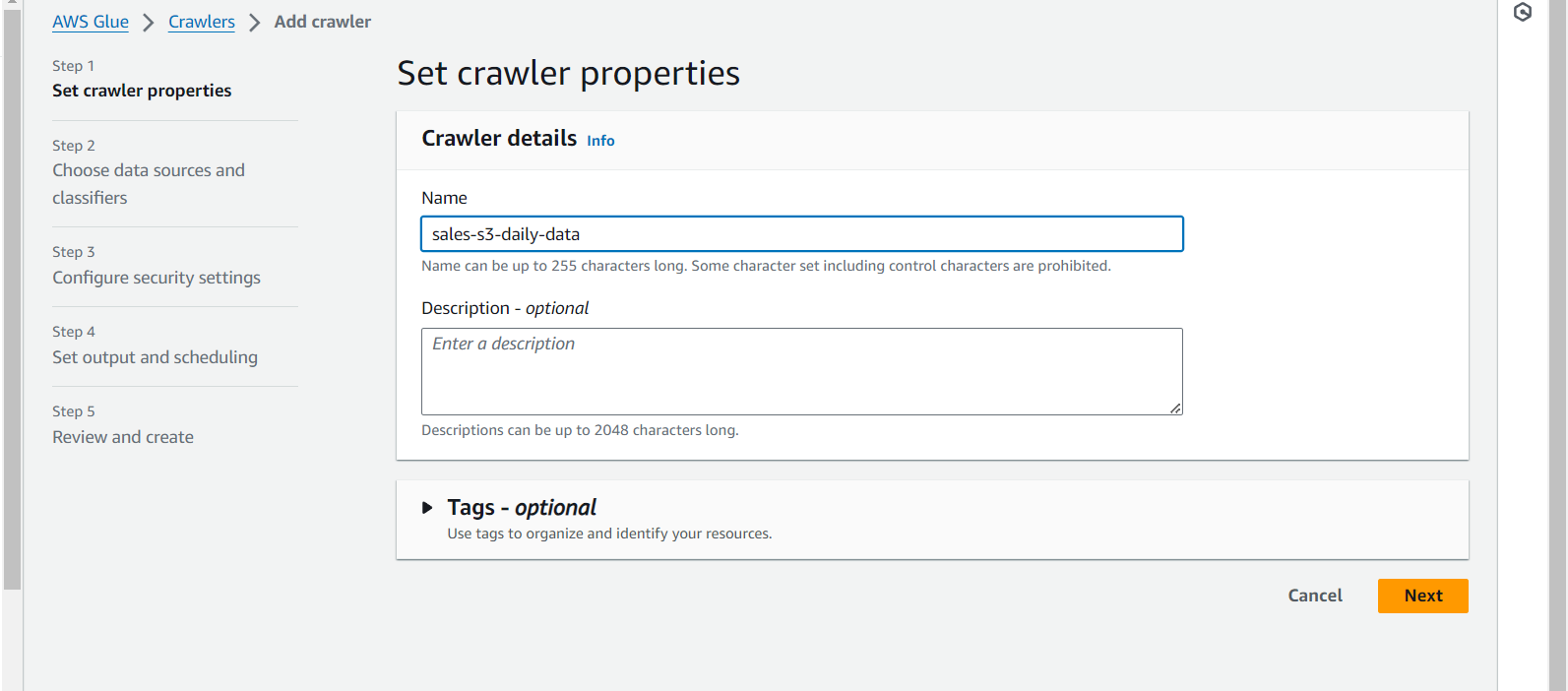
* Before that we will Create IAM Role 🡪(**AWSGlueServiceRole-sai**) with required permissions for this project . We have added required permissions u can see below
* We will use this IAM role for every services we are creating

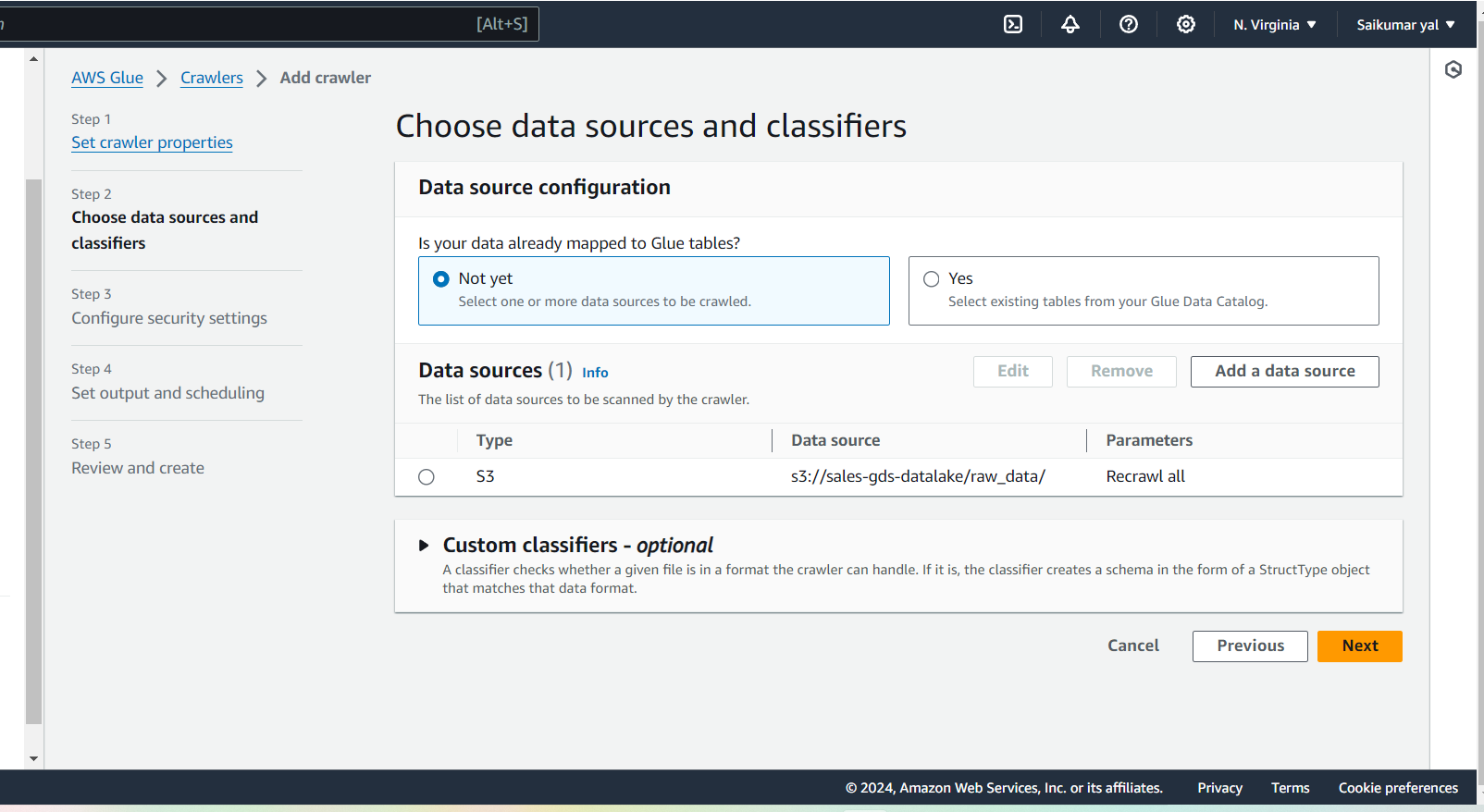




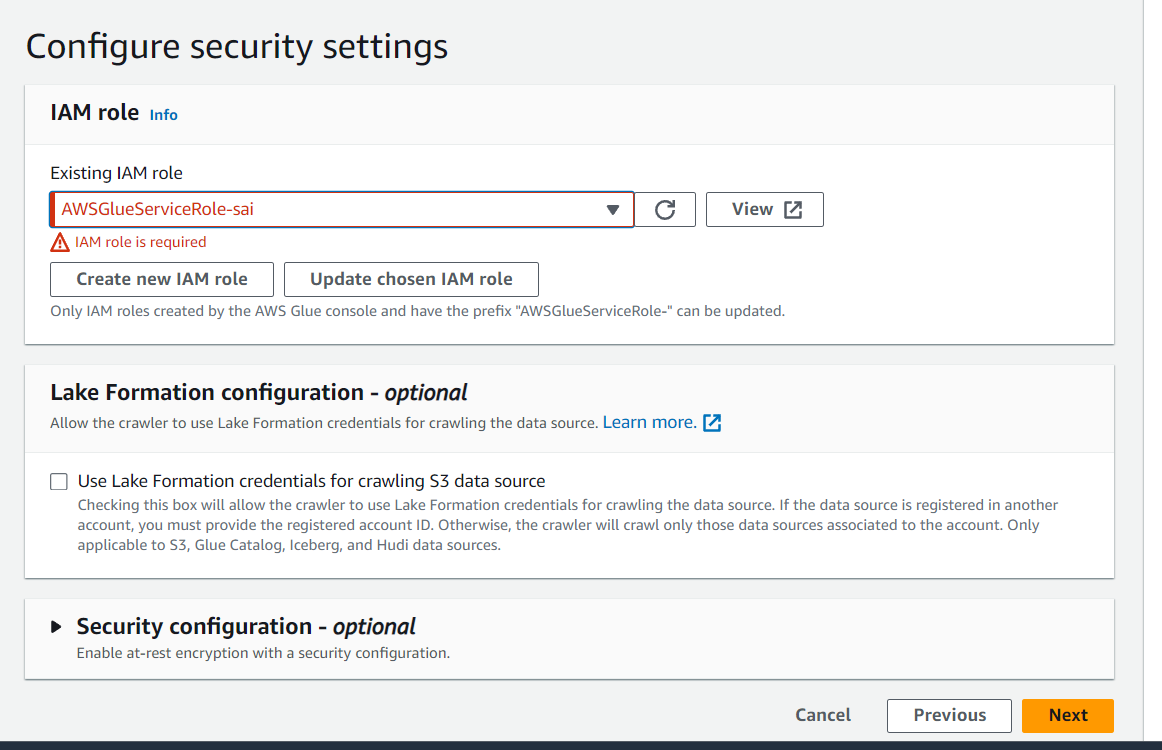
**Crawler**

Next In Glue Click create crawler then Ex: sales-s3-daily-data then select data source here it’s S3 and location of s3 data and locate path

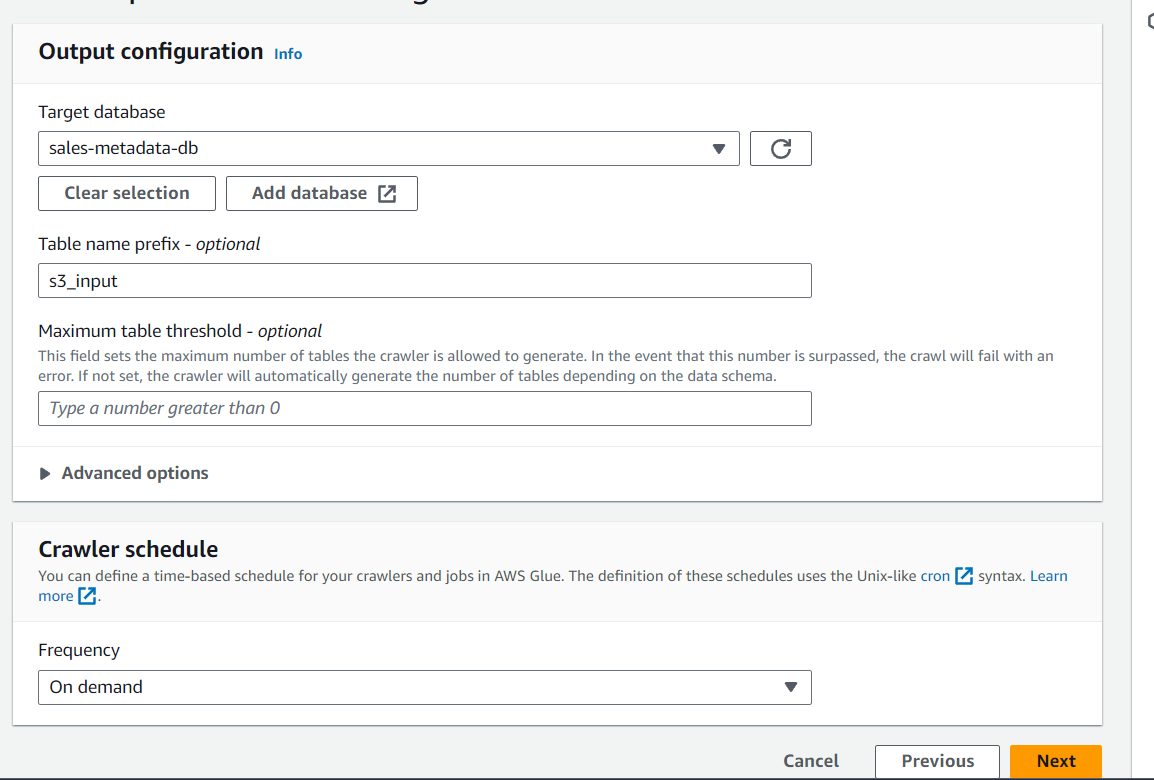




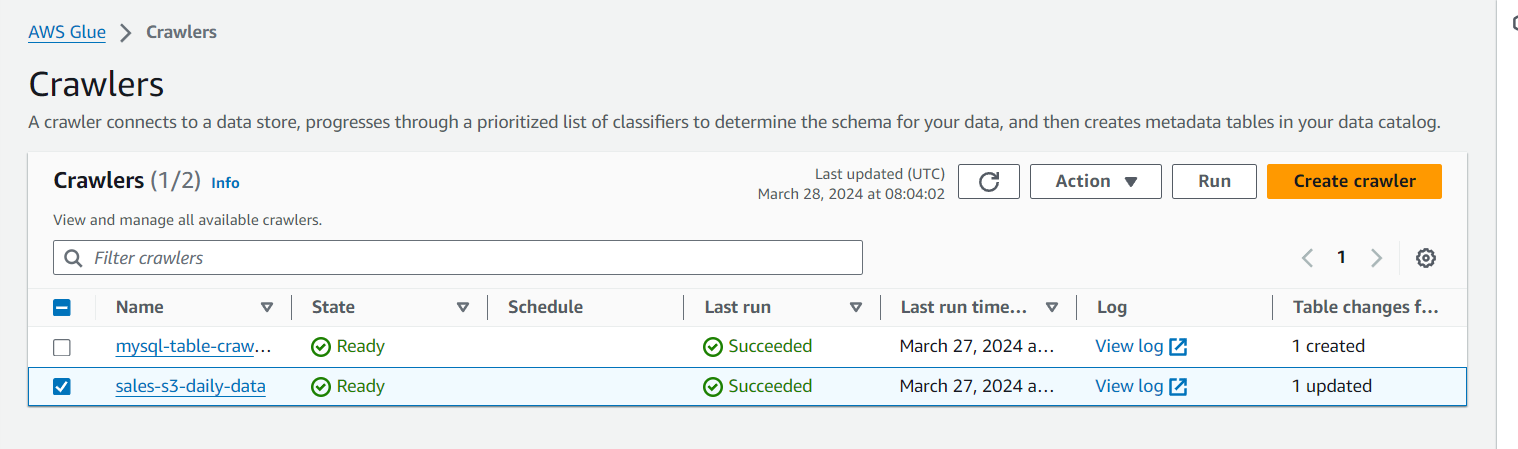
We have added IAM role before we created



Click Next 🡪 in output configuration select database and name table something and frequency on demand ( means manually we can run)

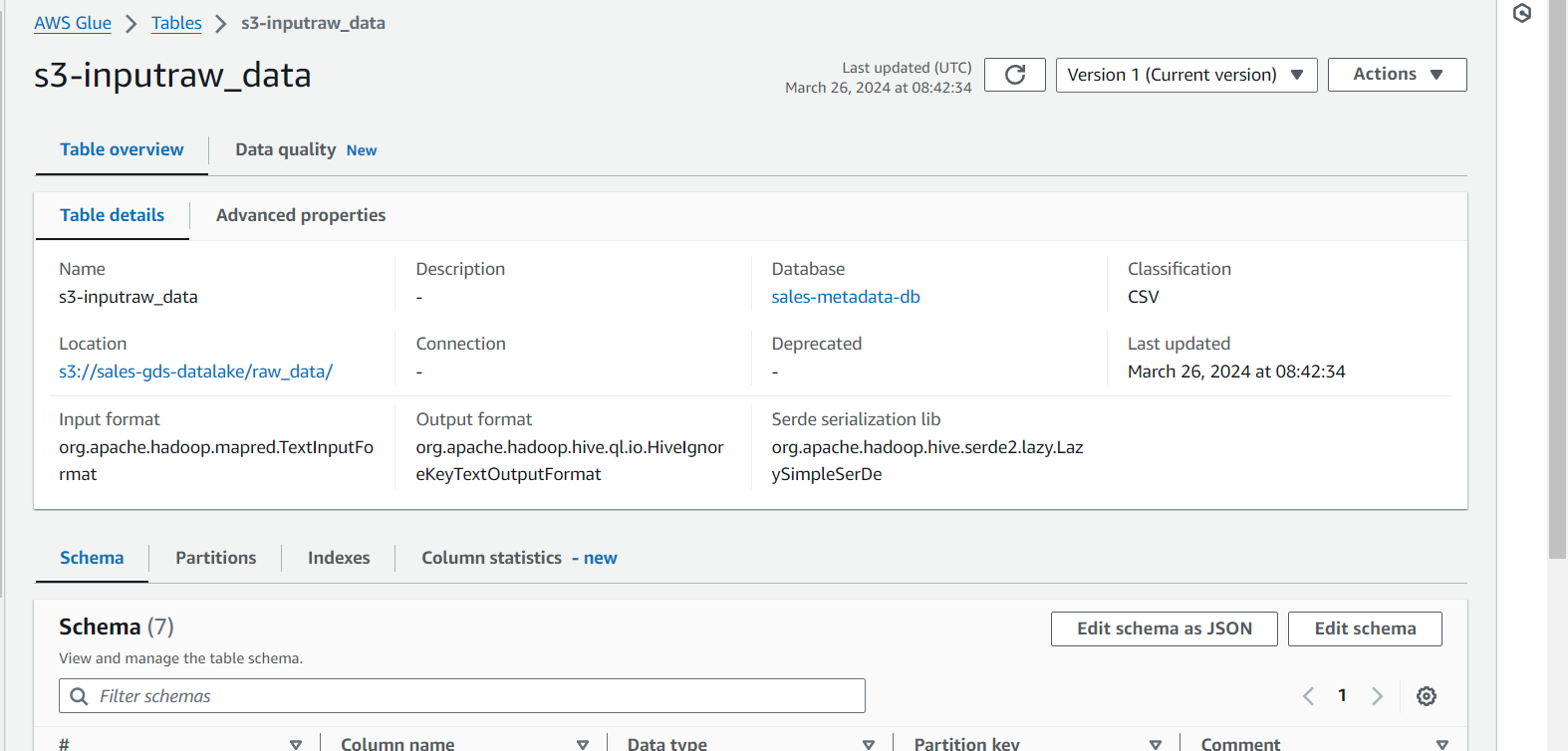


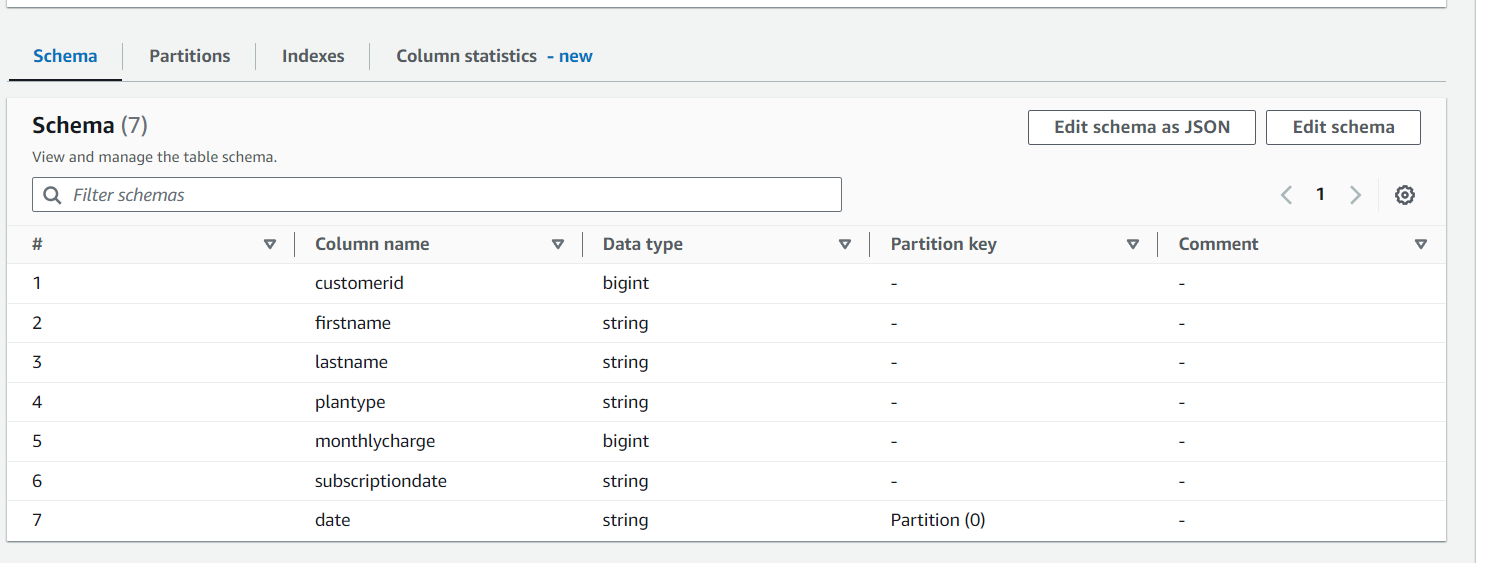
Next Create crawler then

* Next Run the crawler i.e (sales-s3-daily-data)ss we created just now
* 

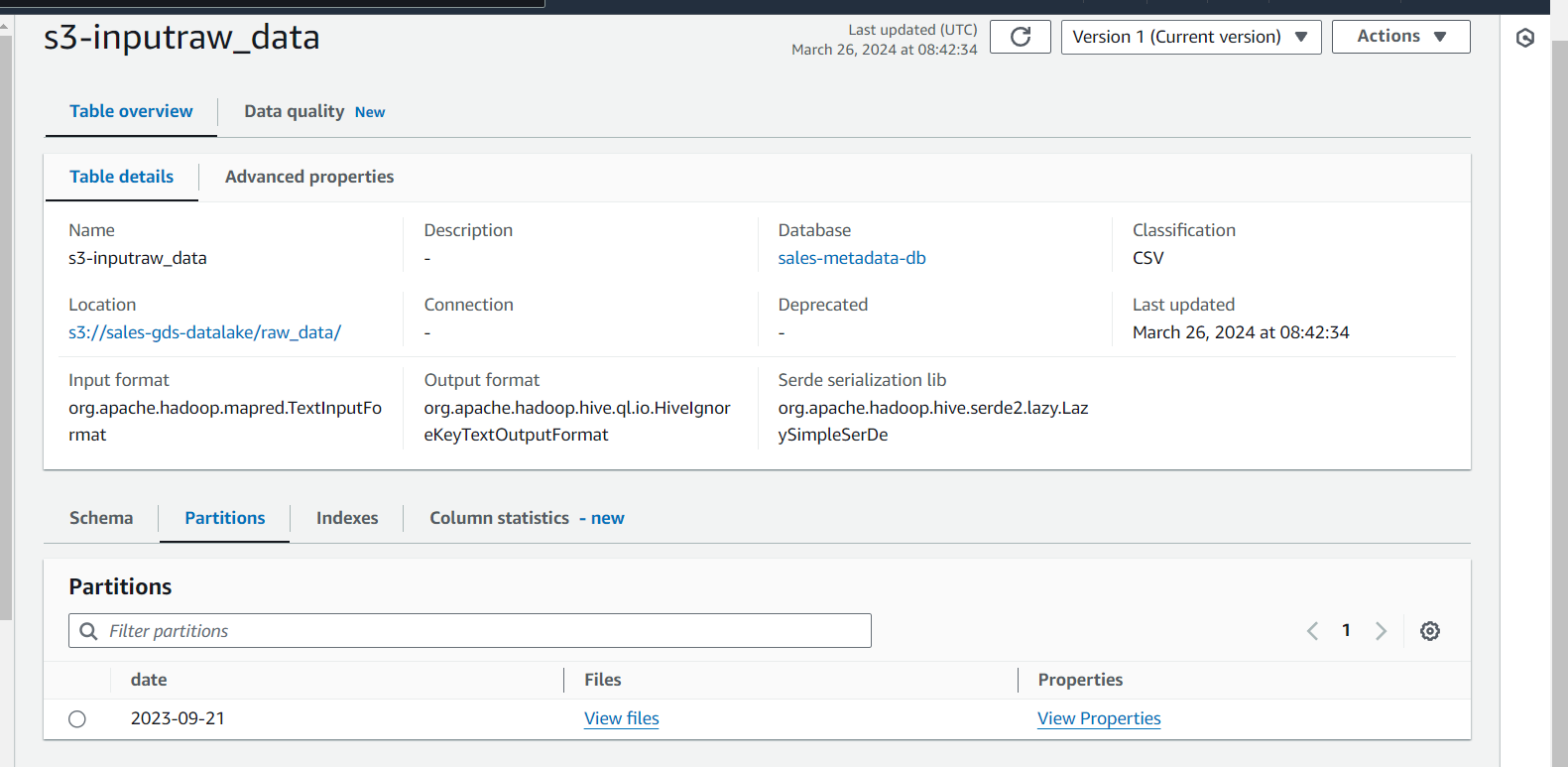
After Crawler run and stopped Next go to the databases🡪 open [ Sales-metadata-db ]we can see “s3-inputraw\_data” table has been created this is metadata table and we can see all information here

This crawler has crawled s3 source we upload a file there from that file it’s collected metadata and based on it it’s created metadata table “s3-inputraw\_data” in database ”Sales-metadata-db” in glue we can see all the information





In Partitions we can see it’s collected file from S3 which we uploaded



Upto here we have metadata table for source 🡪 source Catalog is ready i.e catalog with source data information is ready

* Next we should need 1 metadata table for our destination, so that seamless integration we can done in our Glue ETL pipeline

Next we will create table in Destination database i.e RDS trough CMD

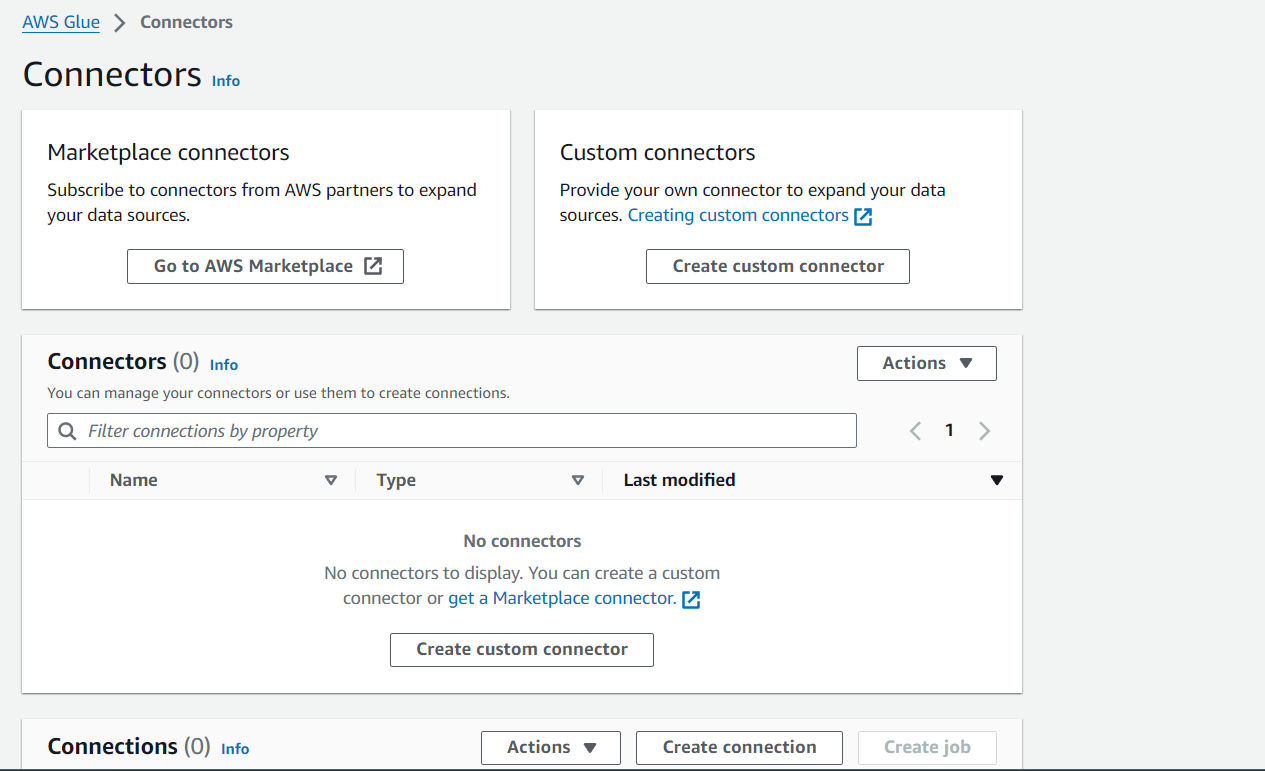
* First we will create Sales DB in our RDS if not Exists through CMD then next will create table customer\_subscription with following Parameters(attributes or fields)
* CREATE TABLE customer\_subscription(
* CustomerId INT,
* FirstName VARCHAR(255),
* LastName VARCHAR(255),
* PlanType ENUM('Postpaid','Prepaid'),
* MonthlyCharge INT,
* SubscriptionDate DATE,
* PRIMARY KEY (CustomerID)
* );

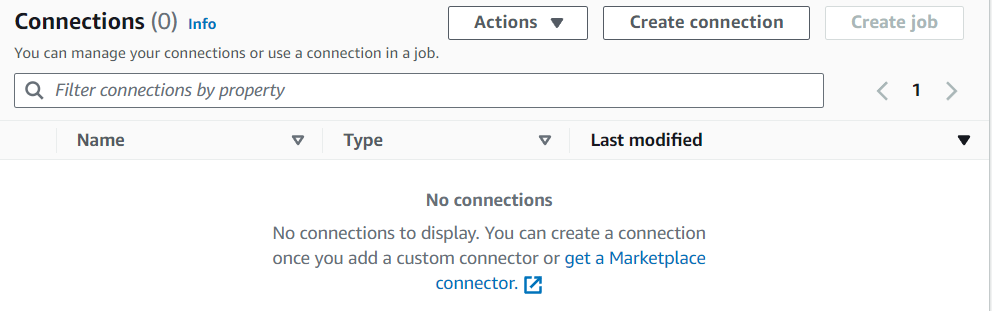
Next In order to Crawl RDS table 🡪 we need (connection) connectivity in glue

* In glue first we will create connection for that , once connection is ready then our crawler can crawl a particular table in RDS
* We will create a connection and using this connection the crawler will crawl to particular table or [place

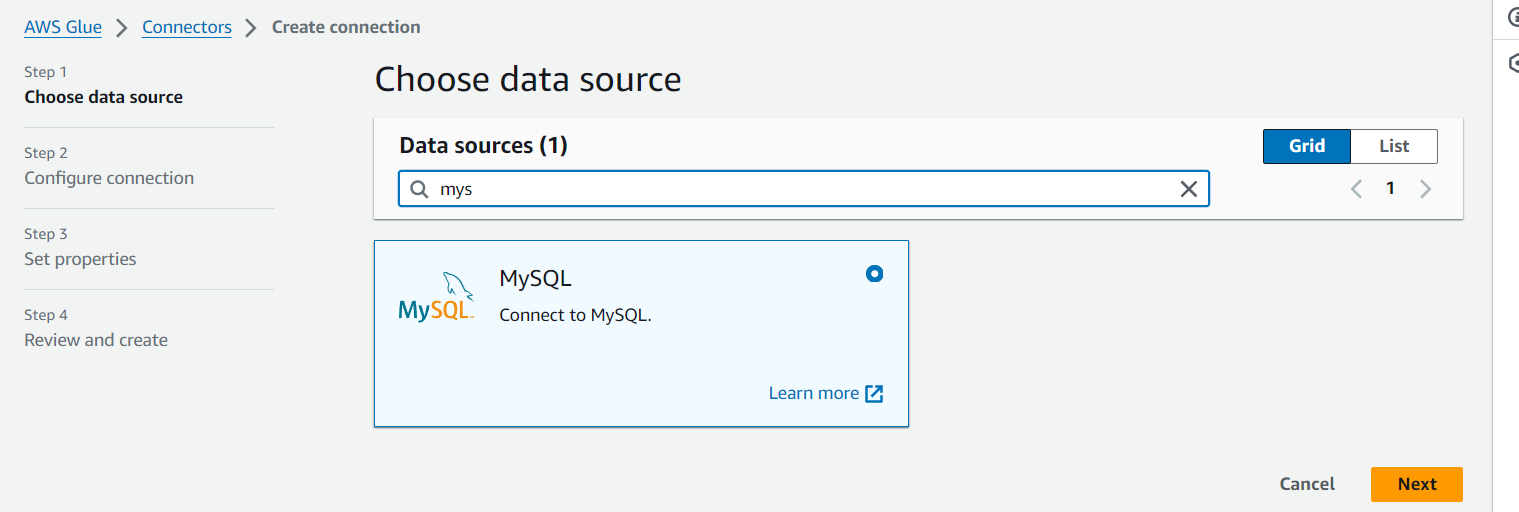
**Connections part**

Go to connector on left pane in Glue and click create connection then

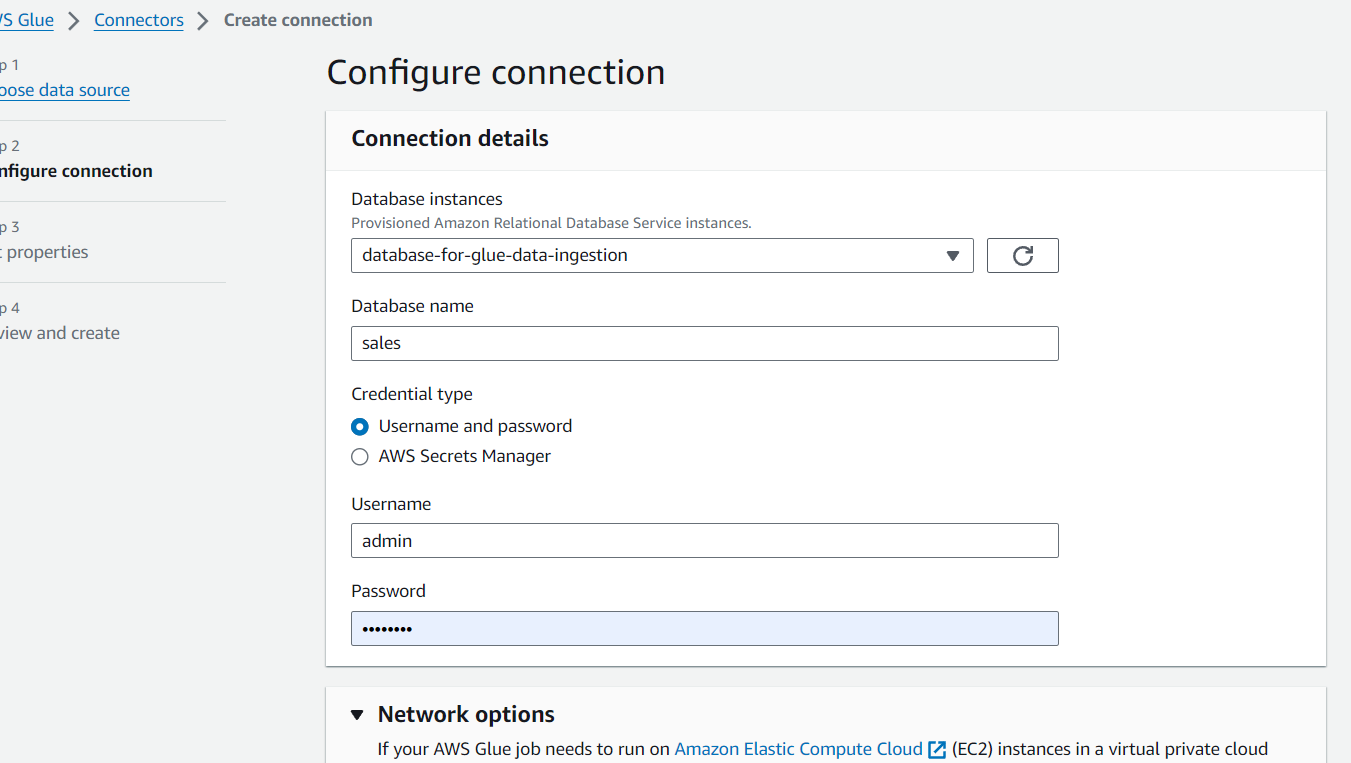




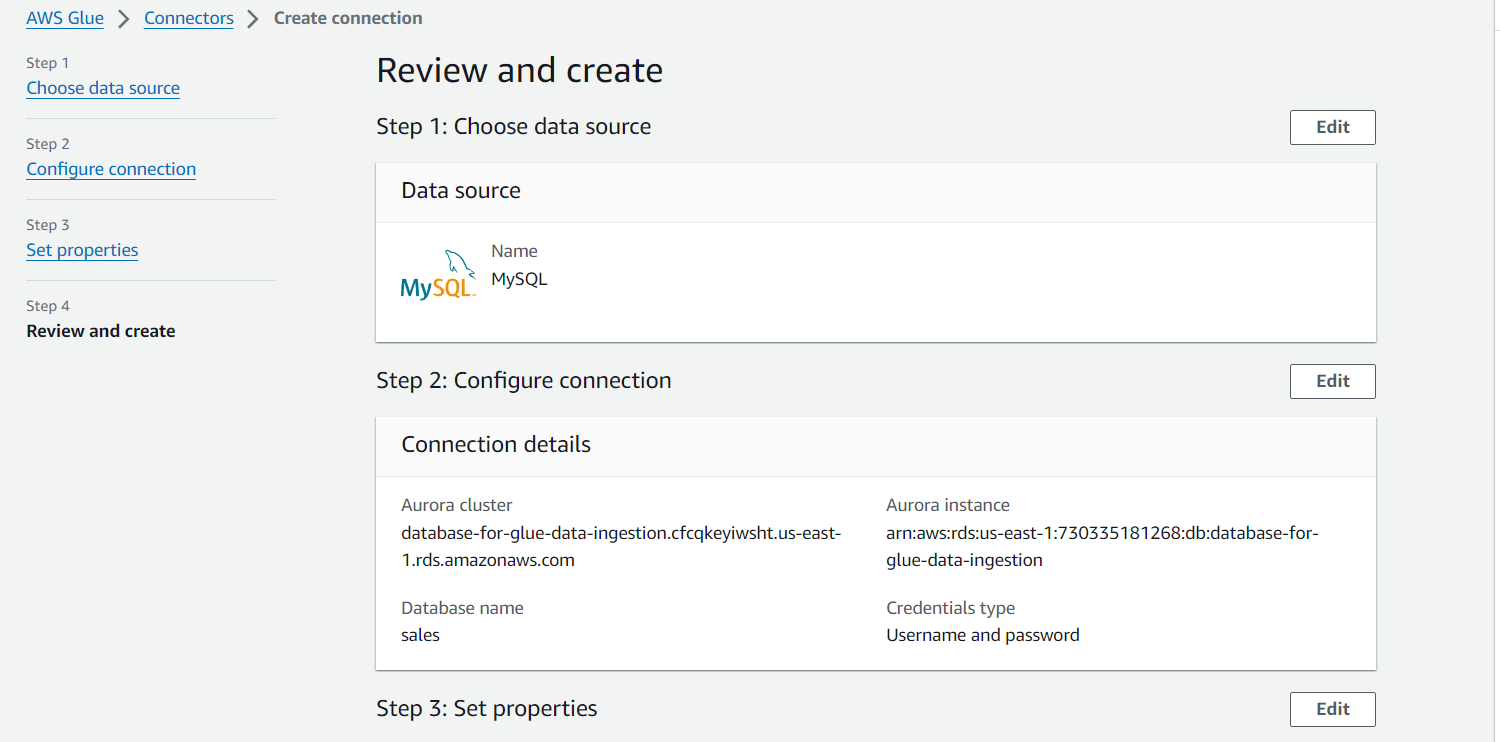
Create connection and choose required datasource from diff databses here we choosen MYSQL



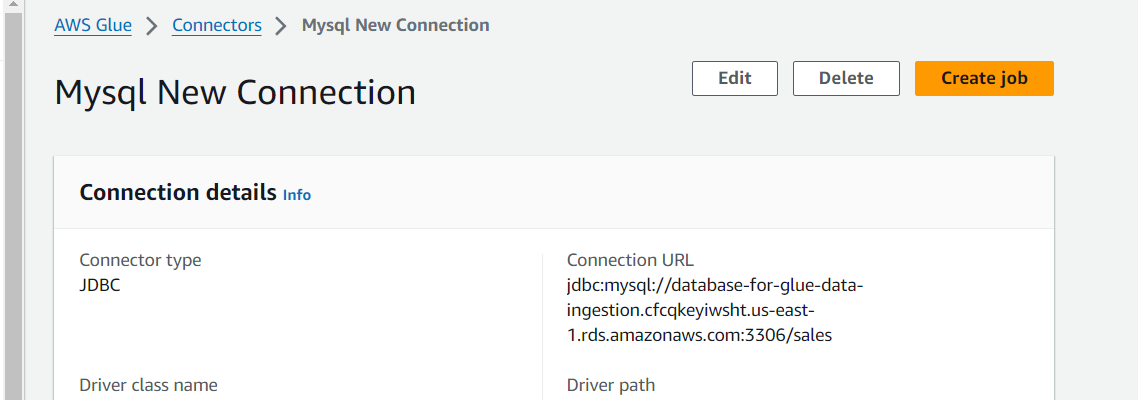
And next choose database instance we created in RDS and database (sales) and give username and password and then



Then name it Connector something ex: mysql new connection next review and create

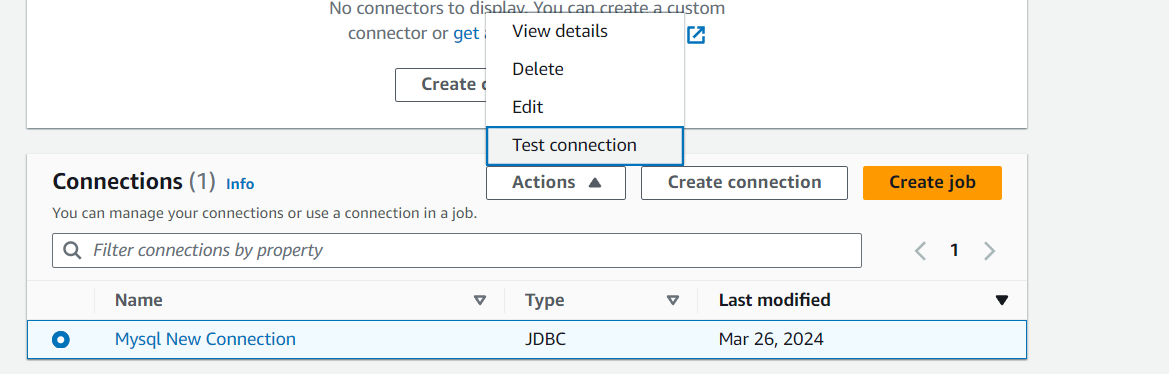


Here it is.

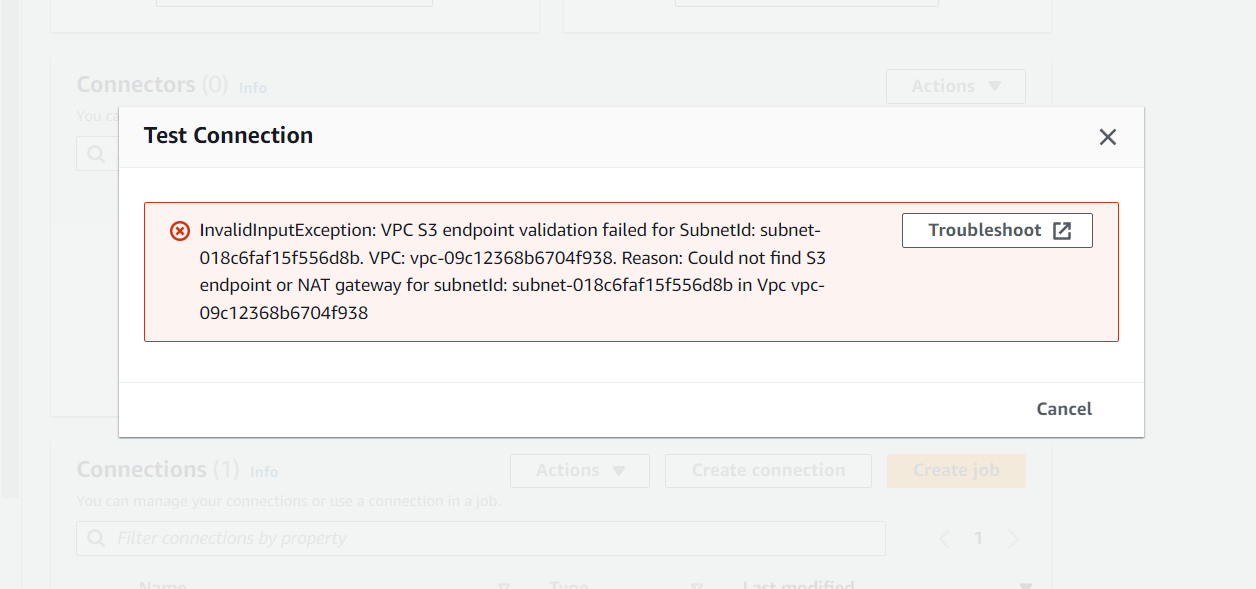


We need to test our connection whether is connecting or it has some errors

Click chose connection 🡪 Test connection

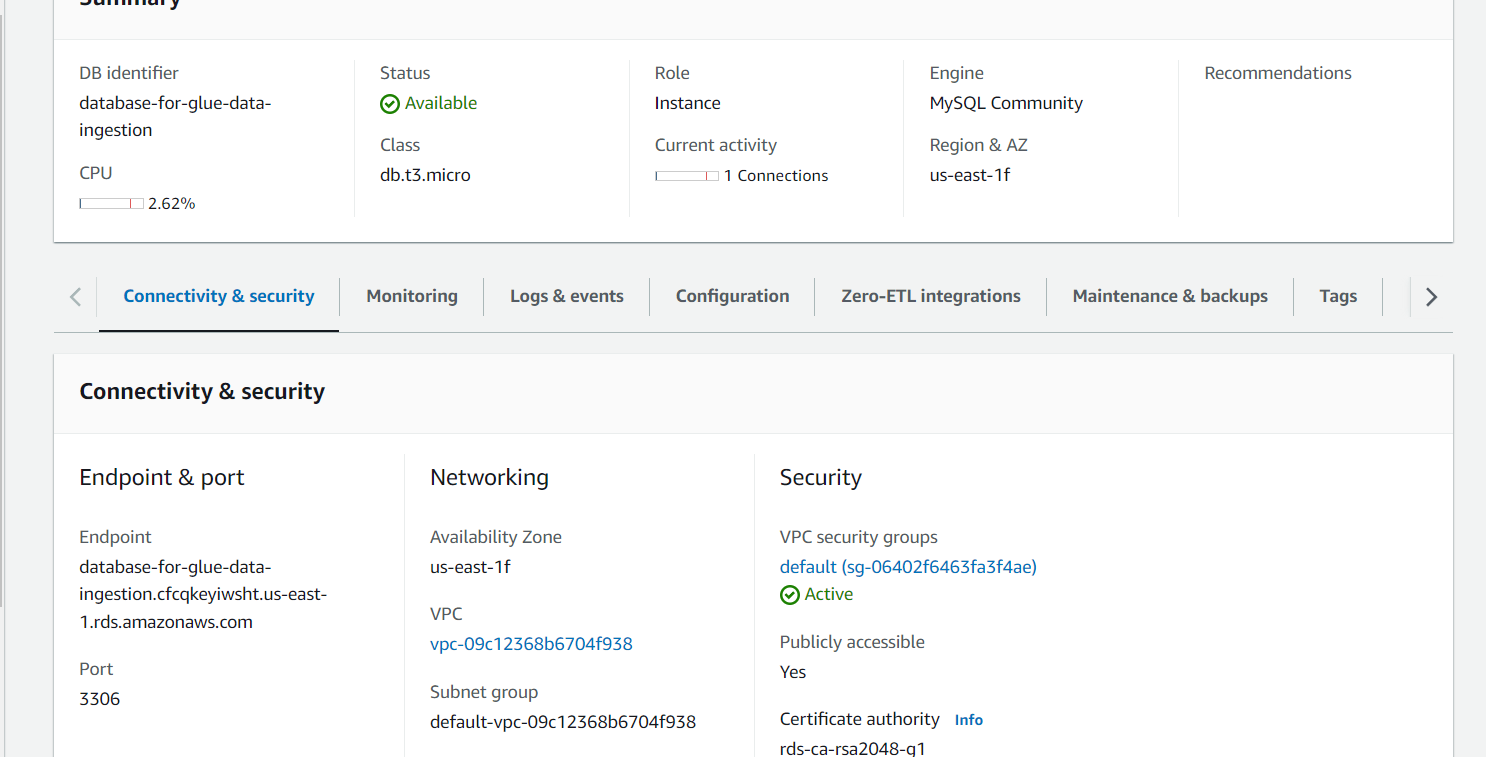


We got this error



Error Because the connector needs to connect with s3 endpoint securely

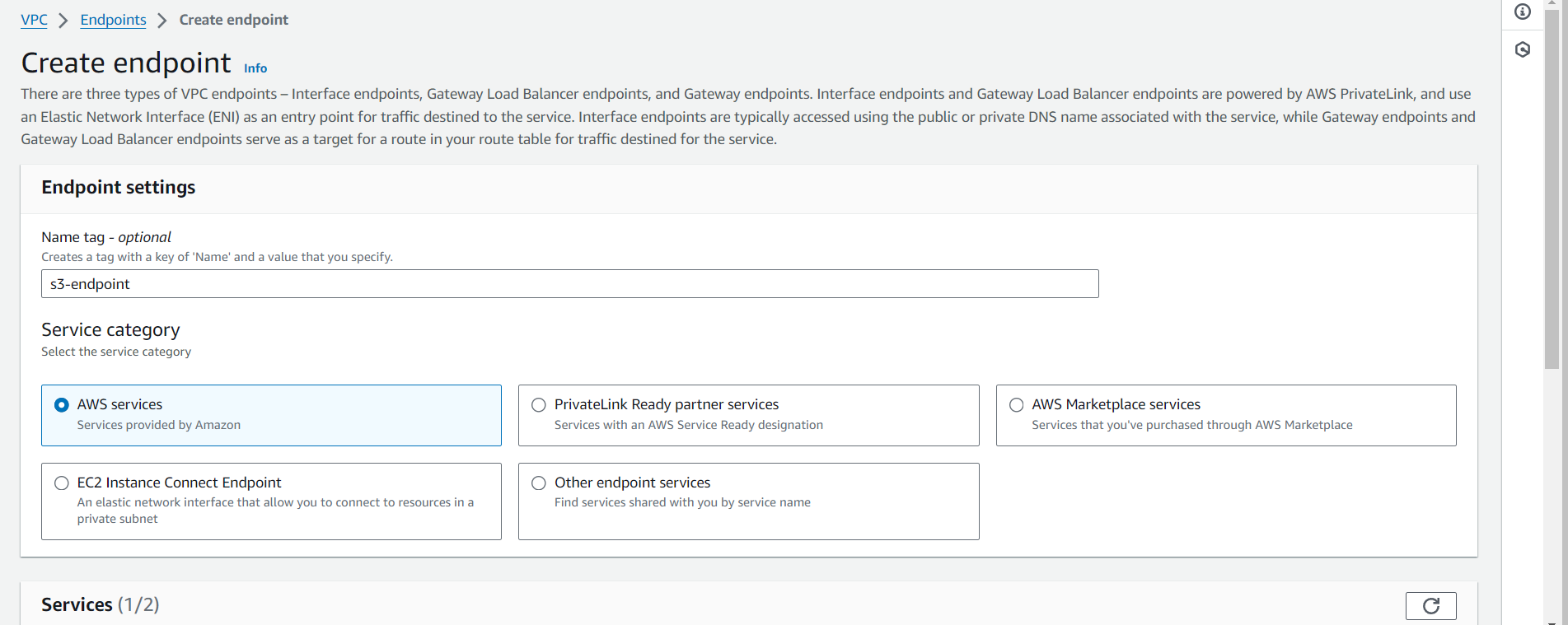
So go to RDS and in that select Vpc



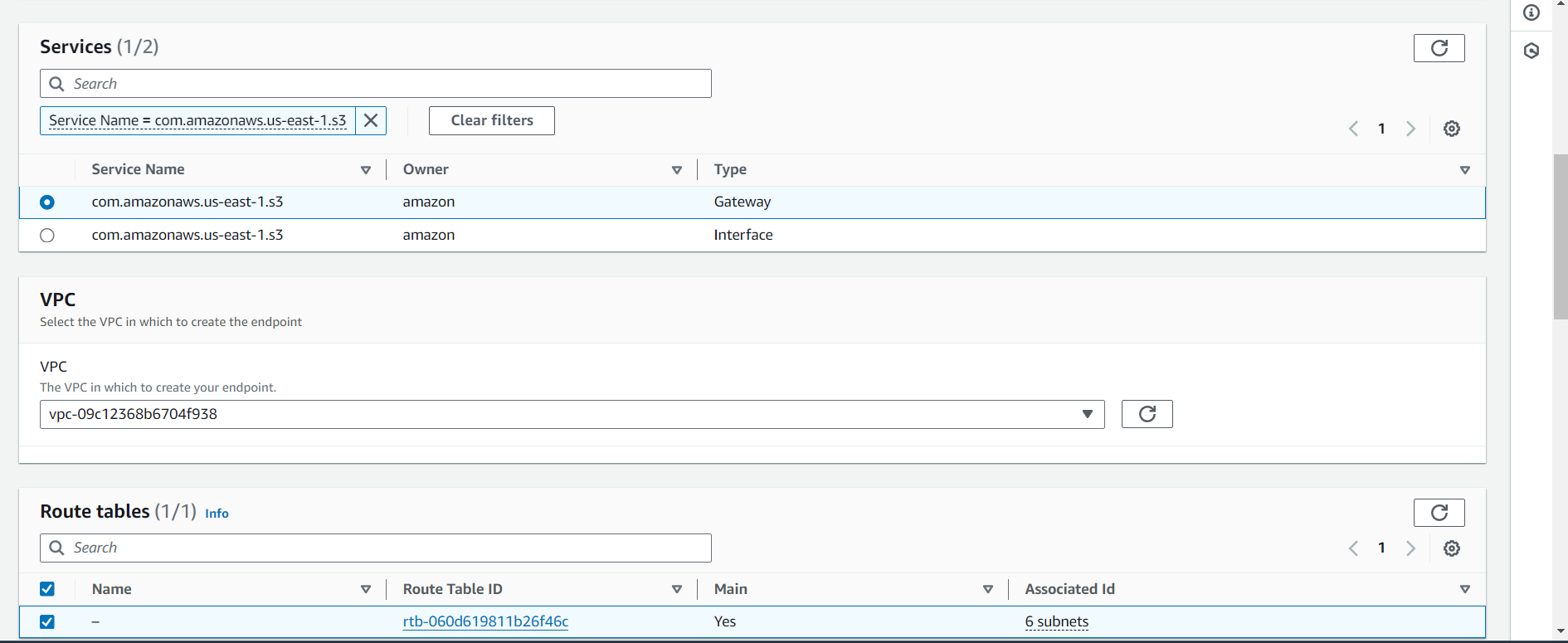


And select that VPC and go inside that 🡪 Endpoints on left pane

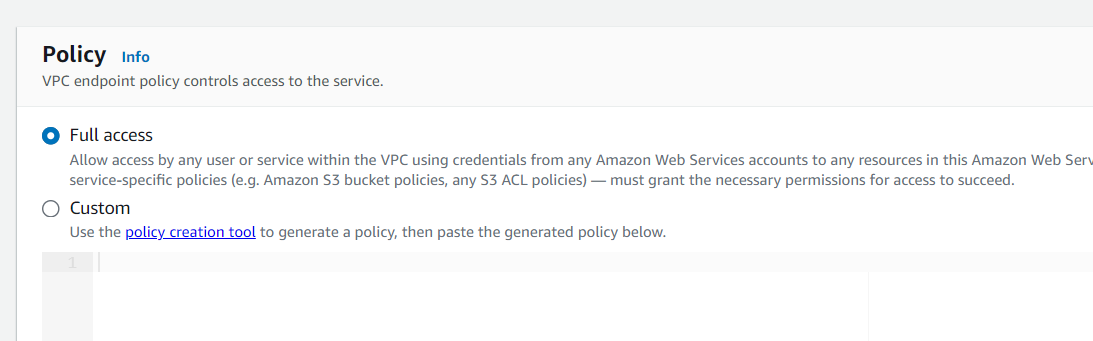
Next click create endpoint ex; s3-endpoint



Select s3 in service of our region and select below gateway and select same VPC and Routes

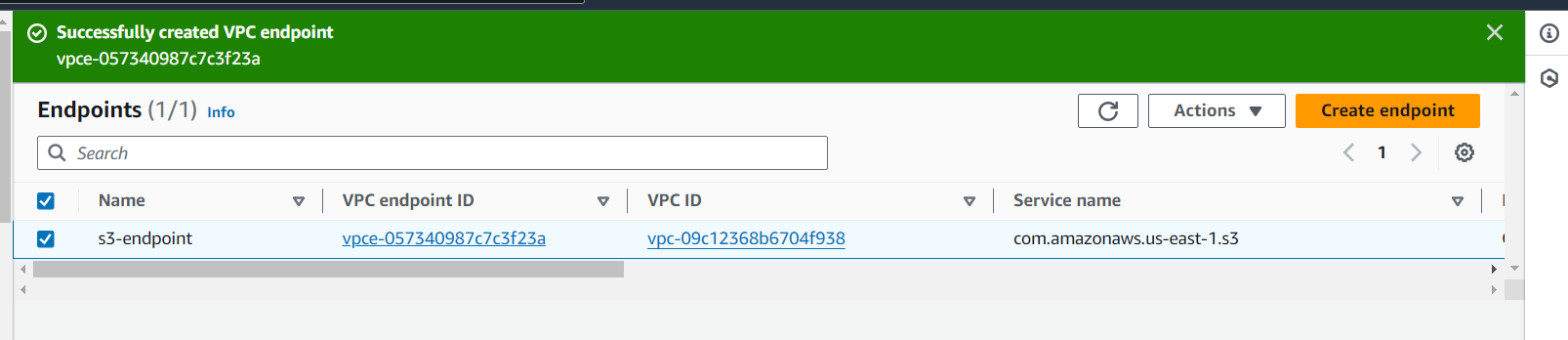


Then full access

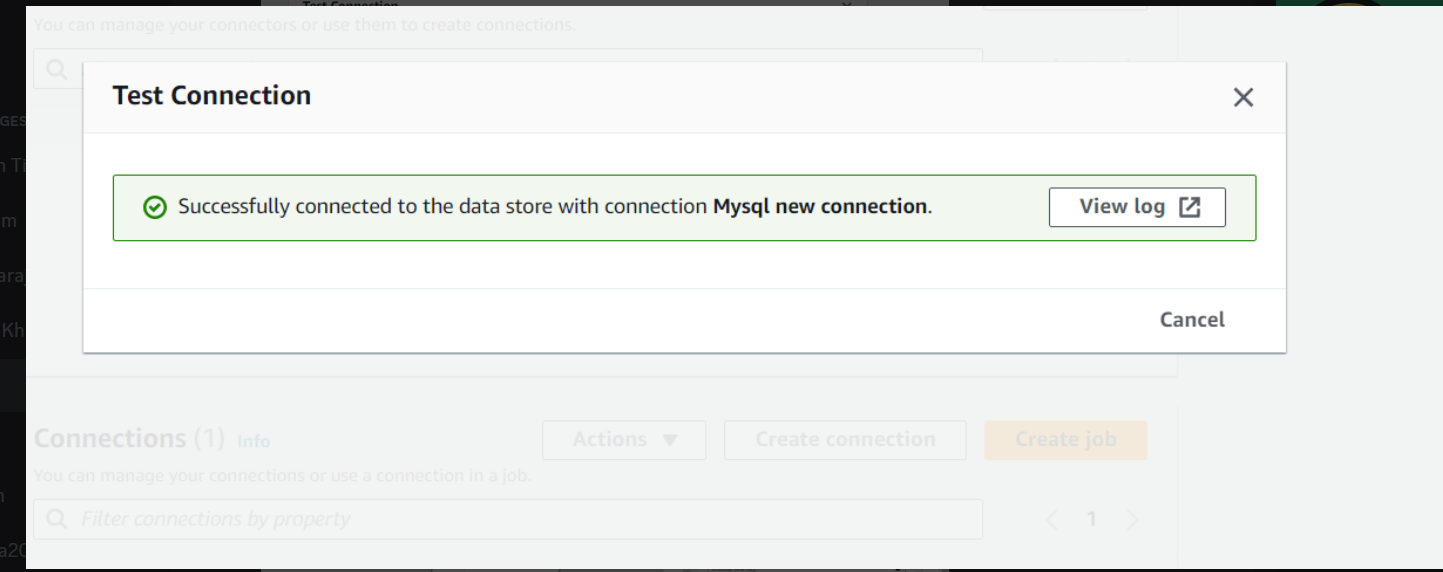


Finally click on create

It’s created endpoint below we can see below



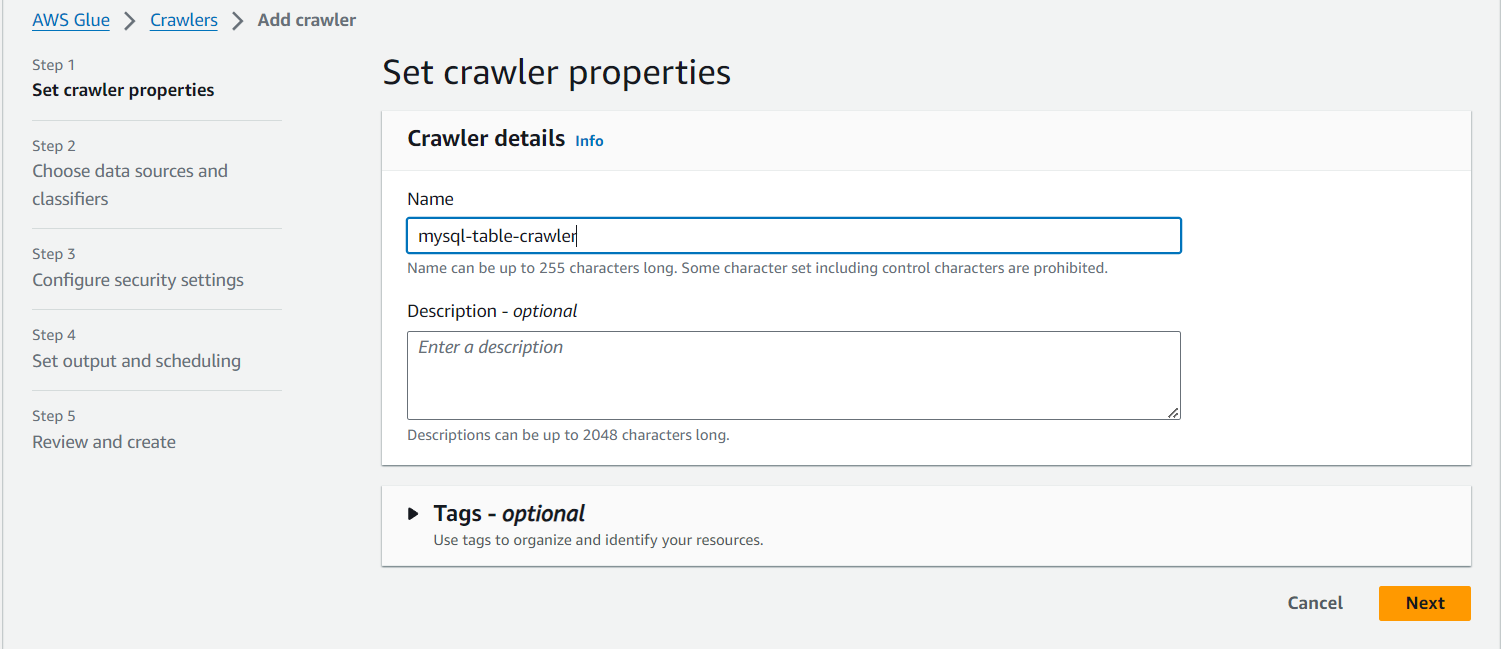
Now go back and select and test our connection it’s success



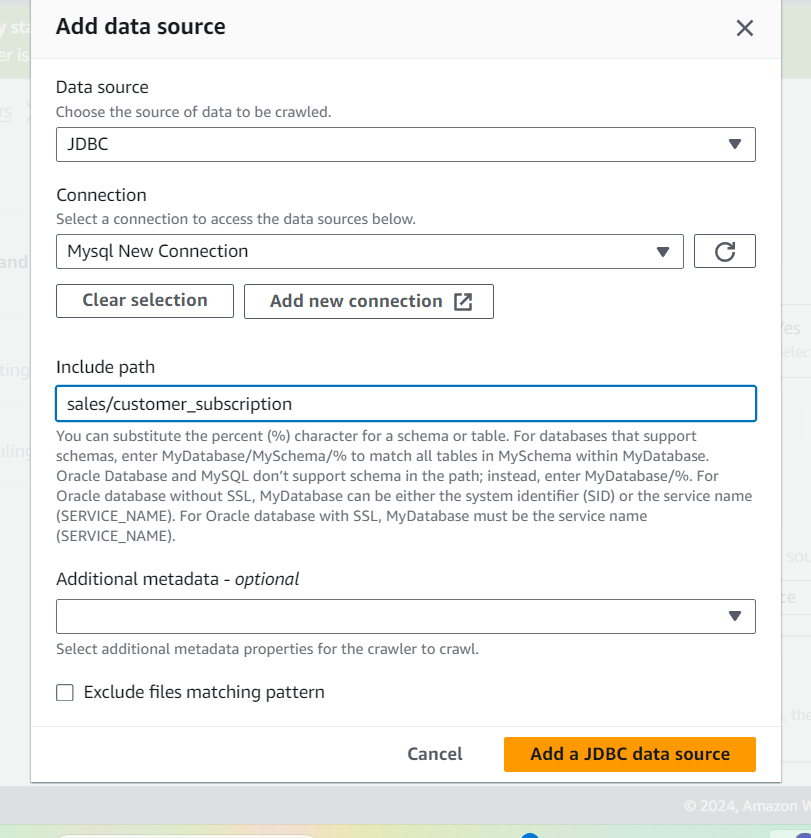
**Target Crawler/Another Crawler**

Next we will create a New crawler for target i.e RDS🡪 to get metadata of target table we are creating crawler

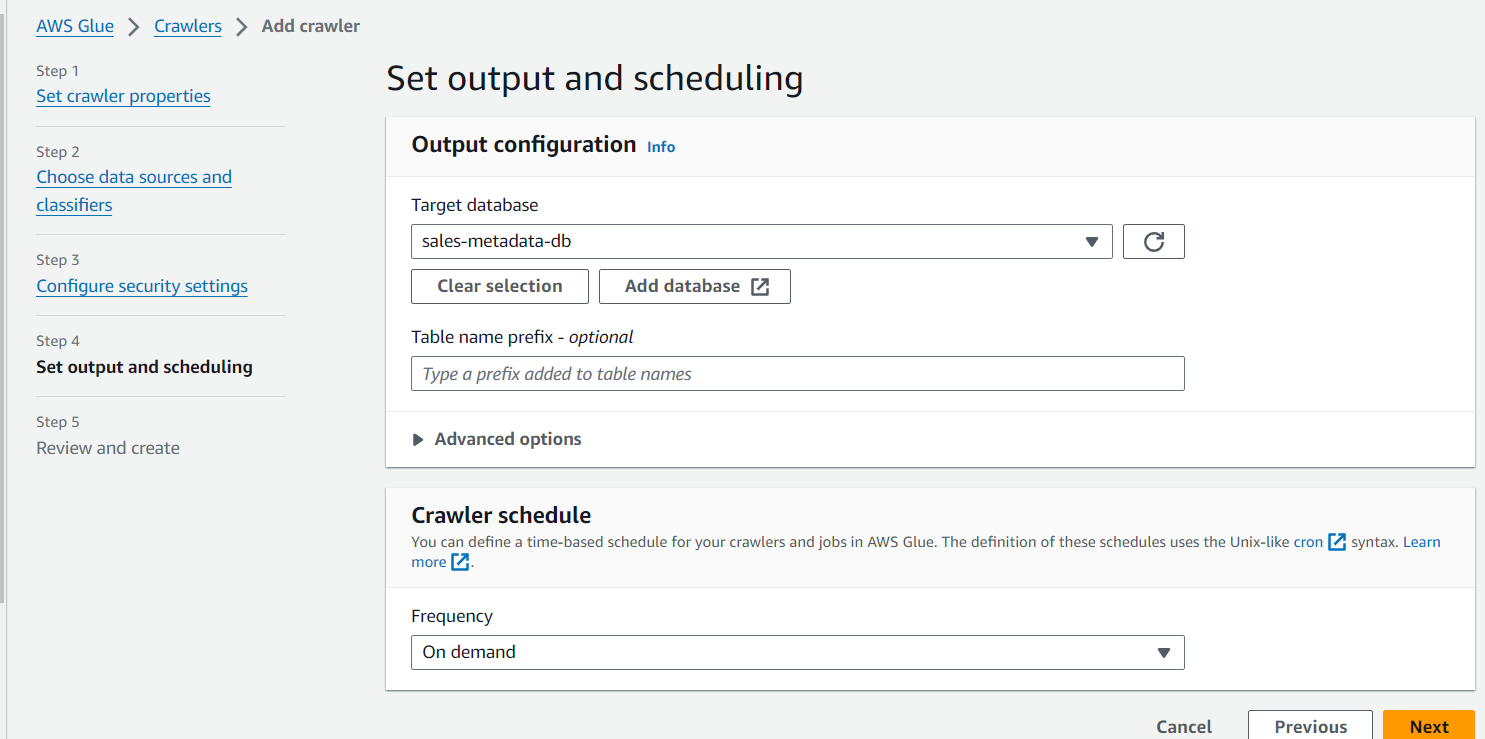
Next go to crawlers 🡪 click create



Include database/table path and select IAM role

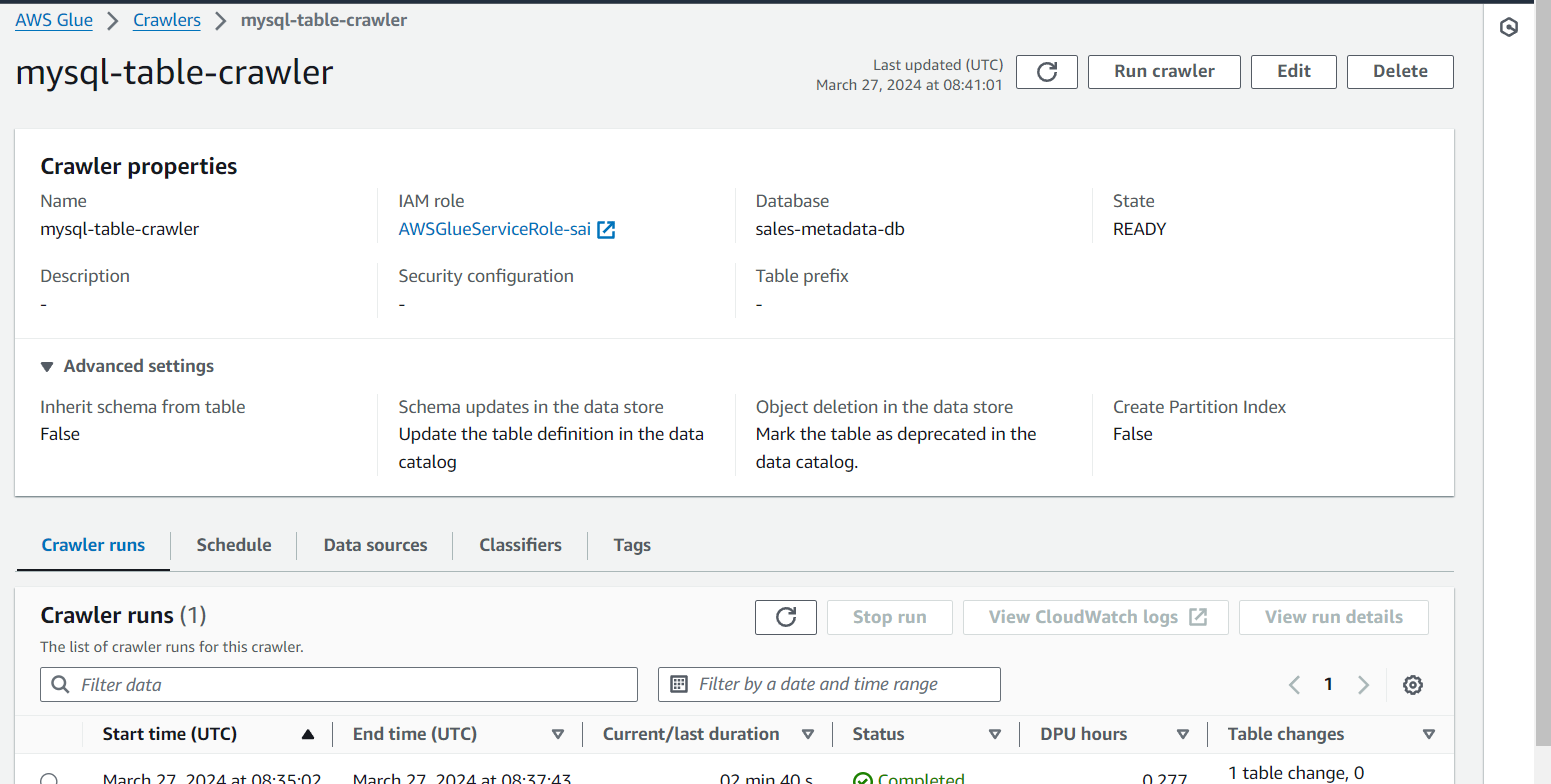


And select db and



click create

🡪Created and its’ below now run it . and we can see information



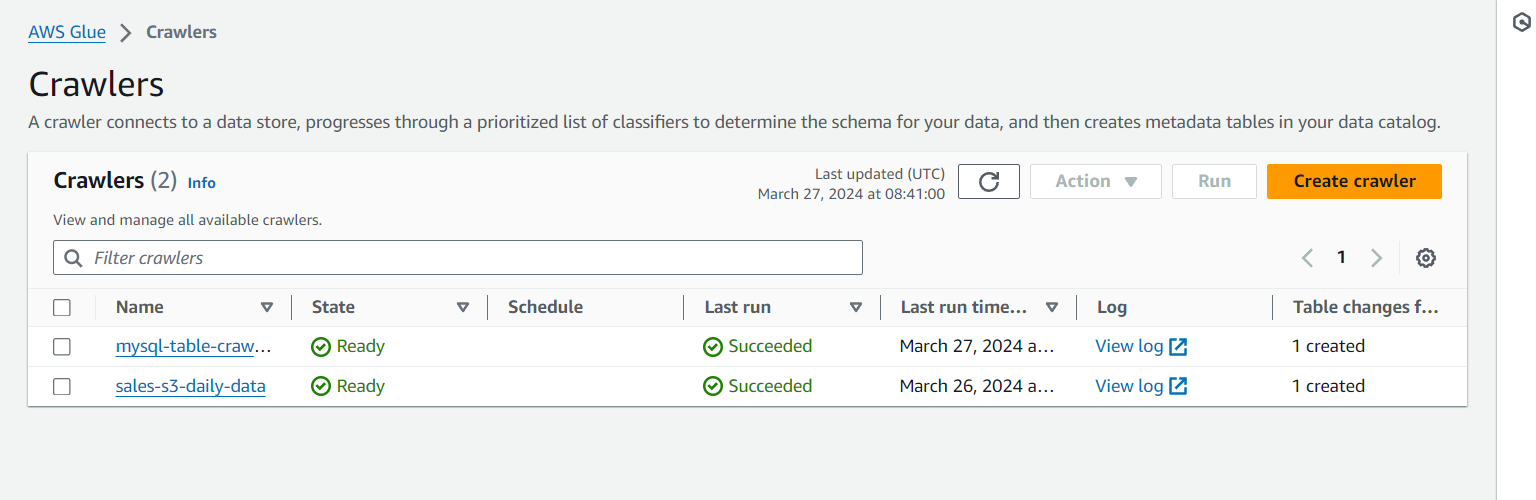
* 🡪 next

The 2 crawlers we created

For S3 🡪 we created crawler(sales-s3-daily-data) to get metadata from source (S3) and store it in glue

🡪and for RDS we created another crawler named mysql-table-crawler to crawl on RDS and to get metadata of it

For both of crawlers the sources are different



Next In Glue only go to database (sales-metadata-db) and we can see 2 tables created i.e

(1). S3-inputraw\_data and (2). sales\_customer\_subscription

* 1).the s3-inputraw\_data table has been created by 1st crawler which crawled on S3 (csv file uploaded in it )crawler crawled on that and created this metadata table)

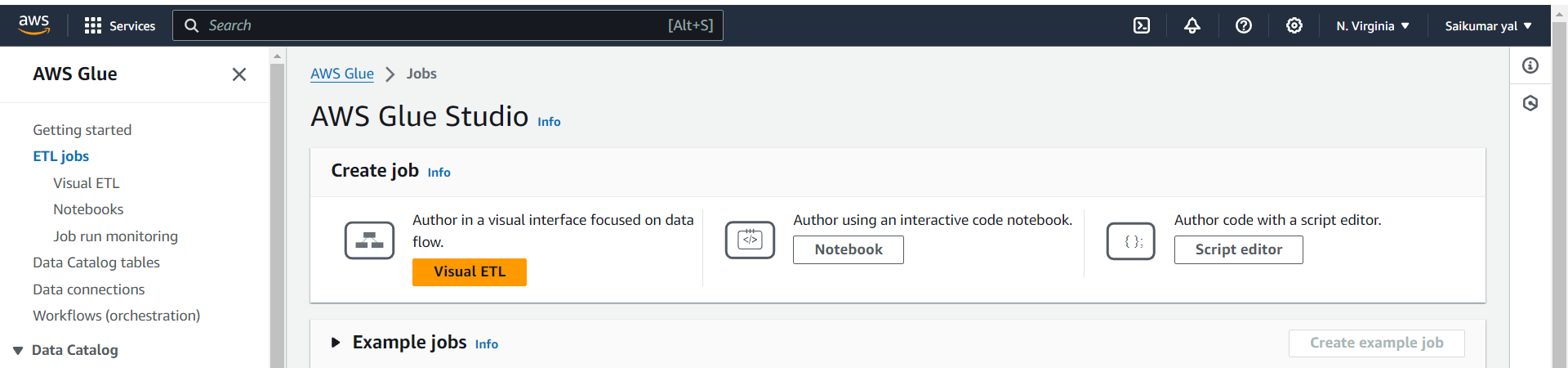
Next

* 2). Sales\_customer\_subscription table is created from RDS , 2nd crawler crawled RDS and table inside it and collected data and stored in this metadata tables

**Next ETL jobs**

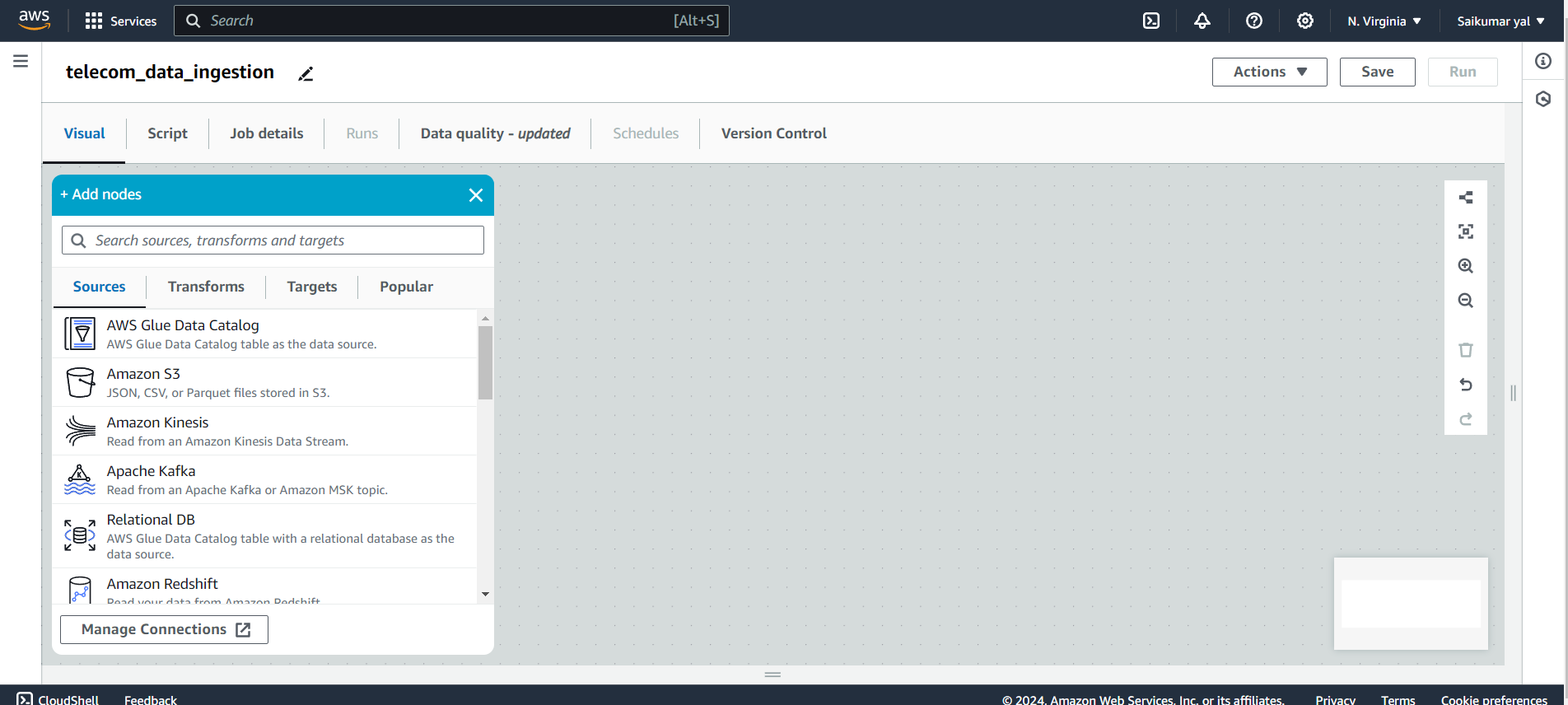
Glue provides out of box functionality to read data in incremental manner , we will do job book marking to avoid repetitive process ,

The out of box functionality of glue is job bookmarking , with help of job bookmarking it can keep track of all files it read so far and based on that one it will only look for newly created objects i.e newly updated files in source , will only read only newly created.



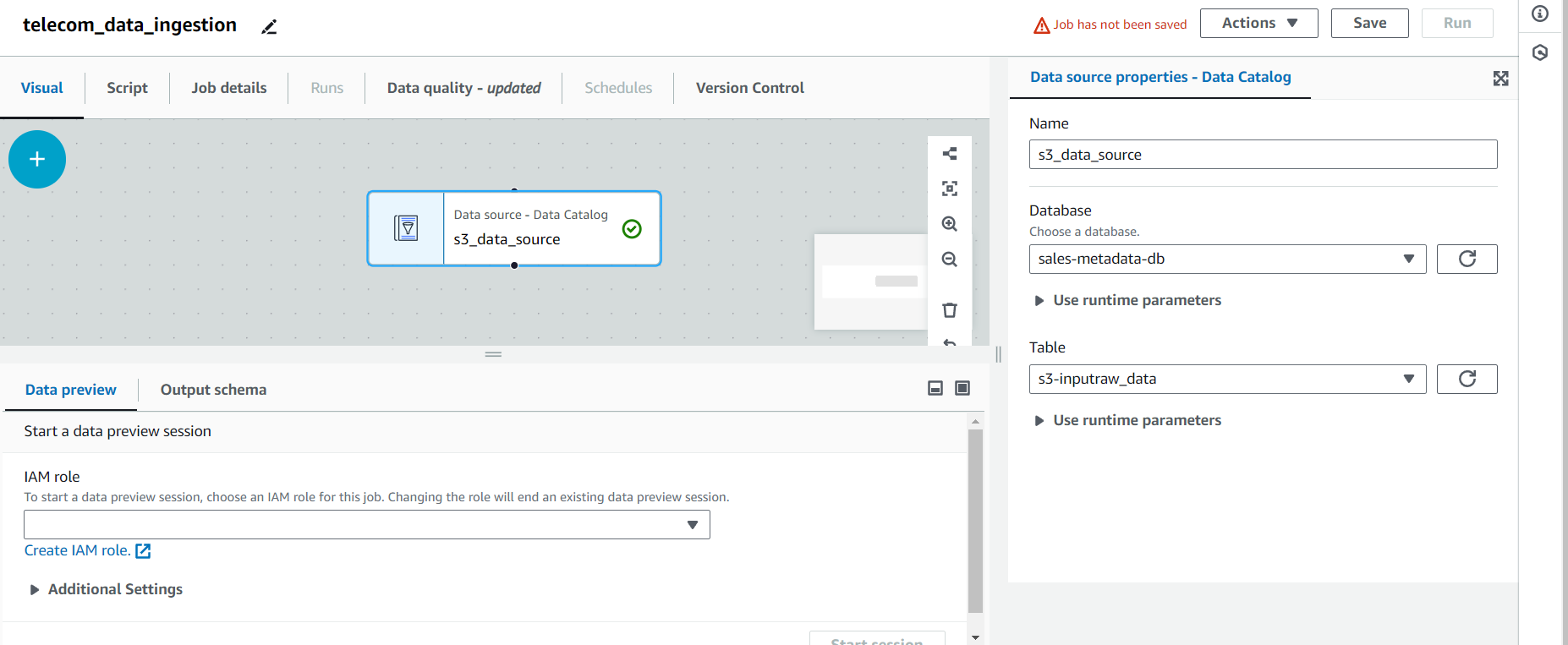
There are options for creating ETL jobs we are using Visual ETL here we can use others as well for creating ETL Jobs

Choose visual ETL then name it something ex telecom\_data\_ingestion

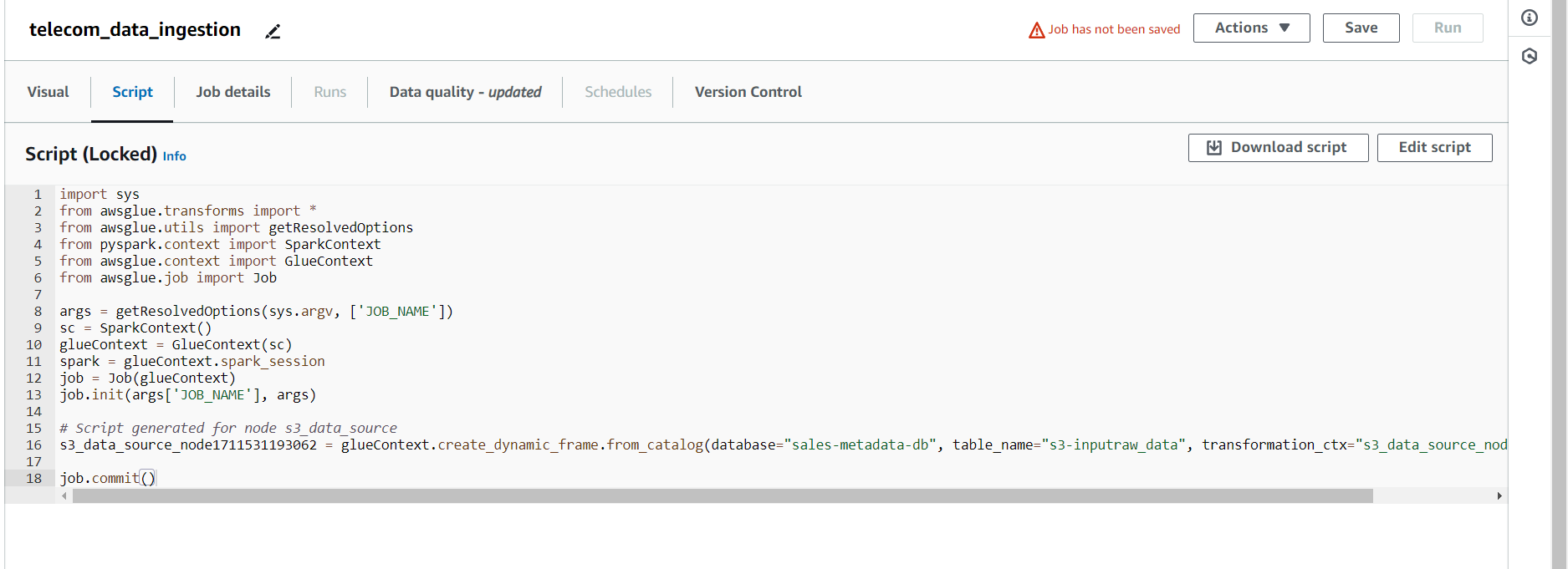


**🡪Source**

First we will choose Source, here choosen aws glue data catalog and rename it s3\_data\_source and we can choose database and table of source and attach IAM Role then



This is only read step and if we want to see backend code of above we created source we can see in script , glue is automatically generating it

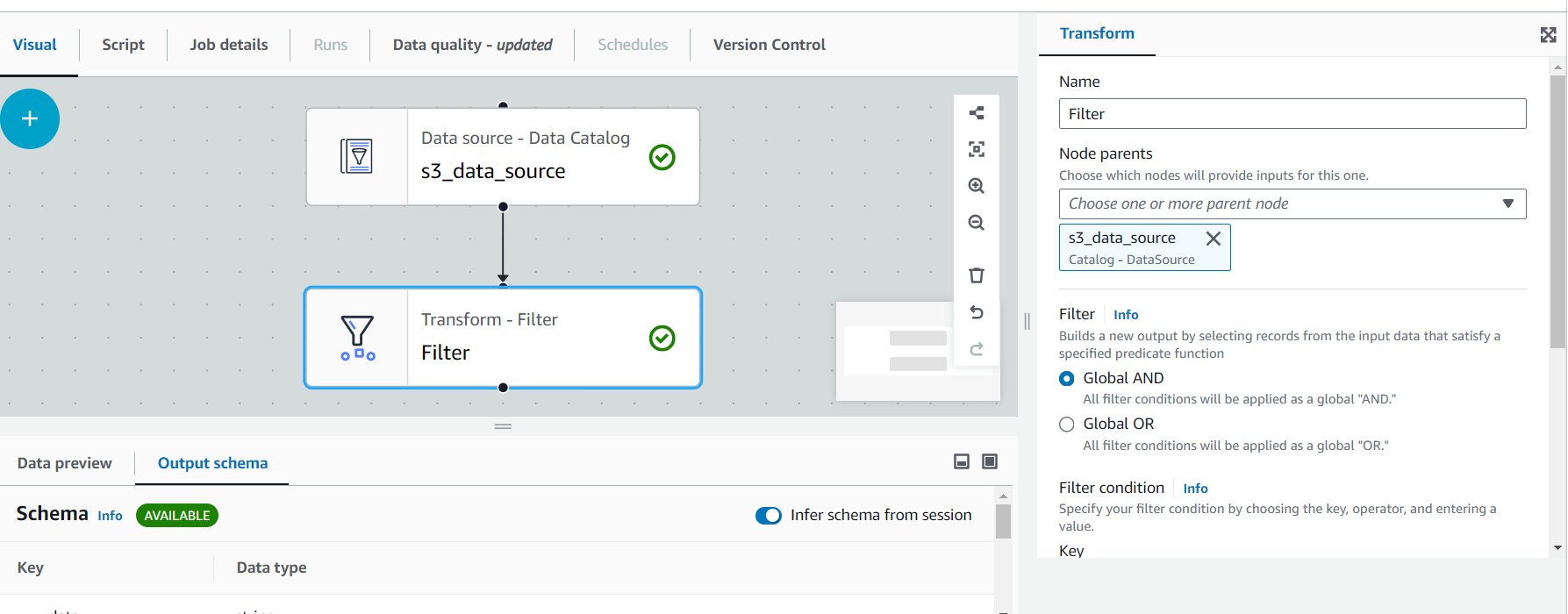


We can use glue dynamic frame for spark guide , here if we make any change in this backend above code manually then we will be not able to use visual representation of it ,then we need to take care of everything

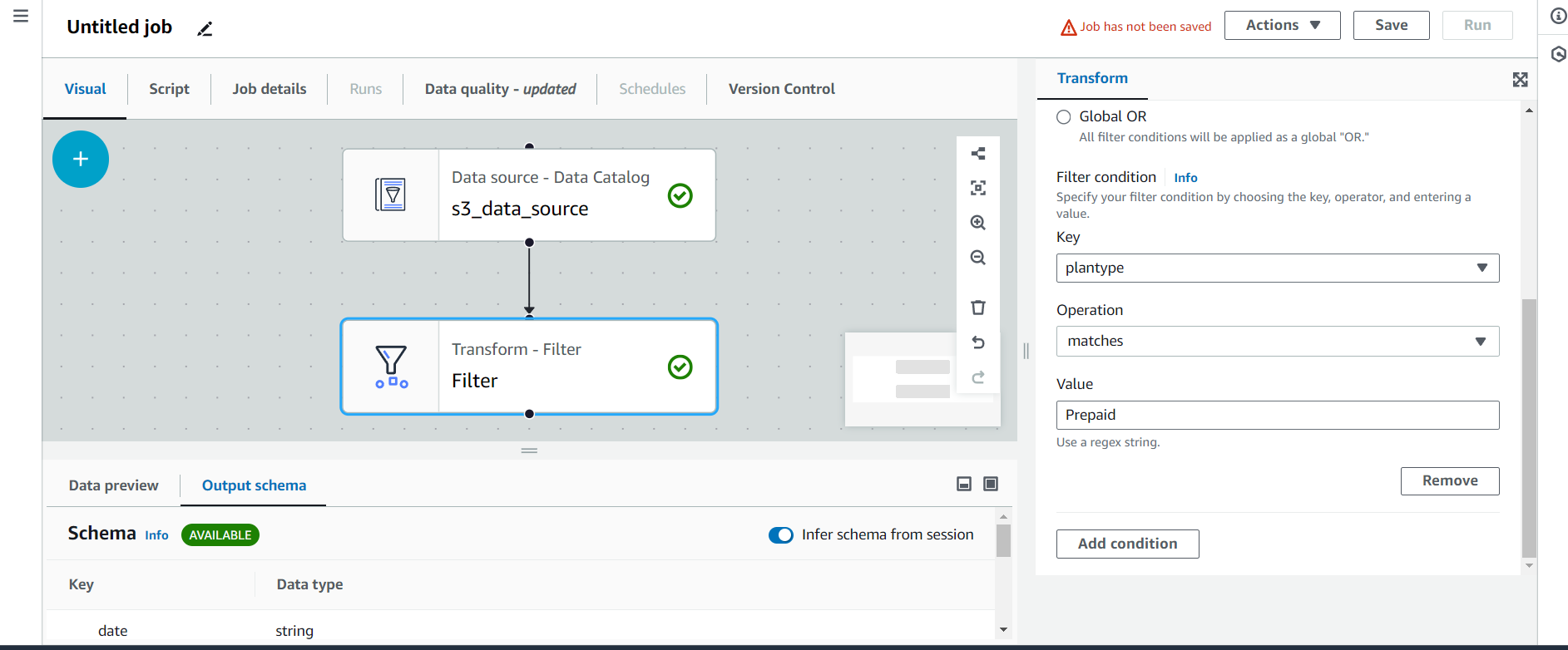
* We can leverage our ETL pipeline using Visual ETL without wasting much time
* **Transform**

Next we will add Filter in Transforms step, here we want to filter only prepaid records of customer table using key i.e plantype (column name) and

Select filter in transform then name it something and choose s3\_data\_source in node parents and In filter Global AND

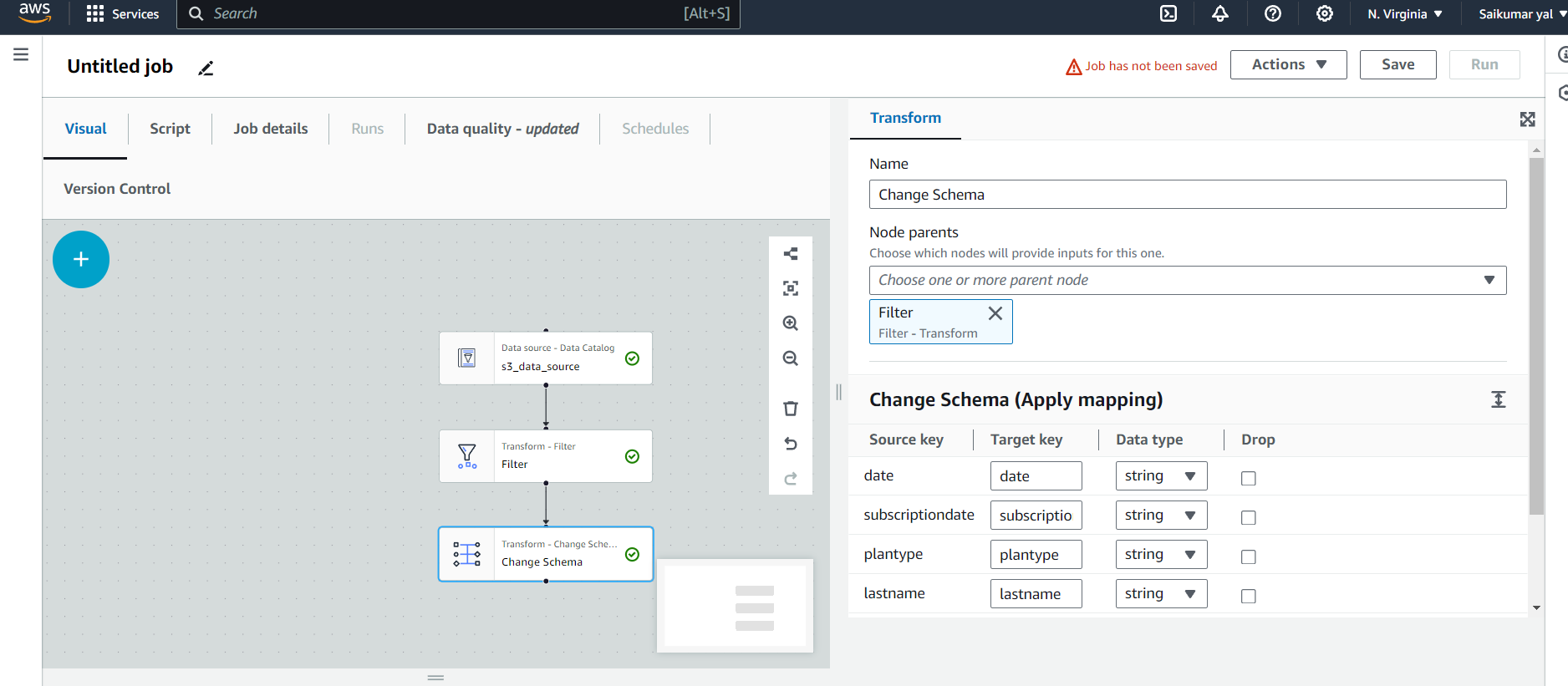


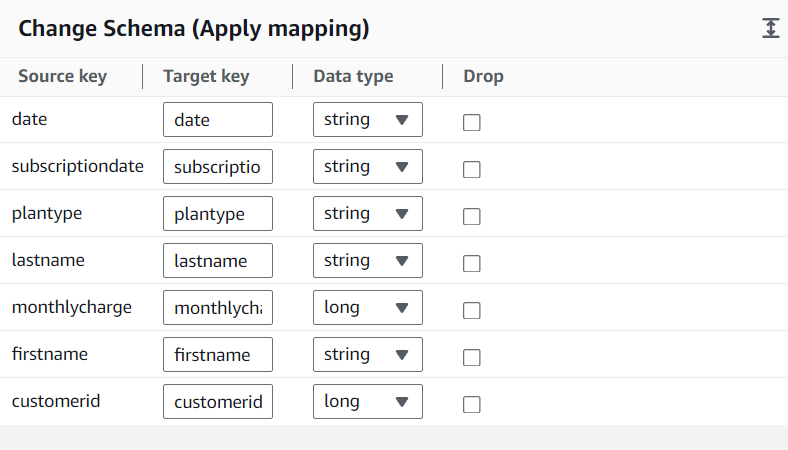
Then 🡪Add condition 🡪key (Plantype) and Value prepaid



**🡪Transform**

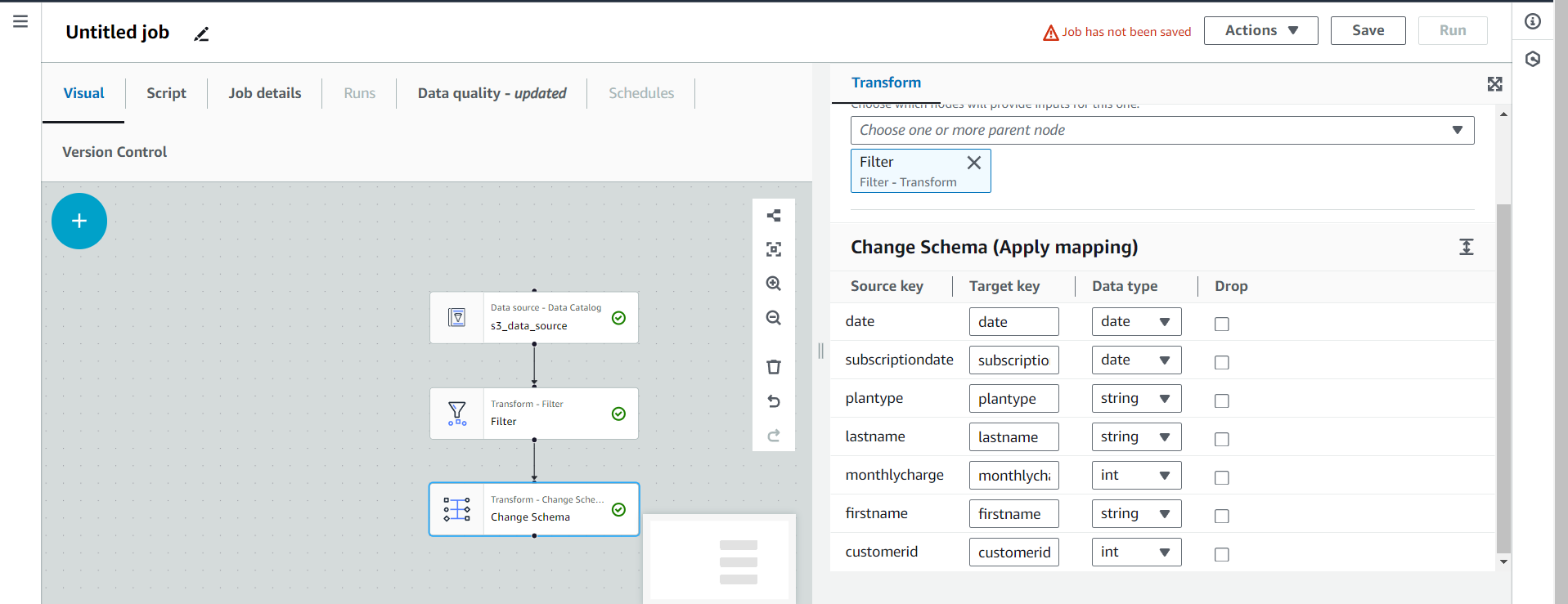
Next in Transform we are choosing change schema here changing schema of this metadata table according to schema of RDS table so do





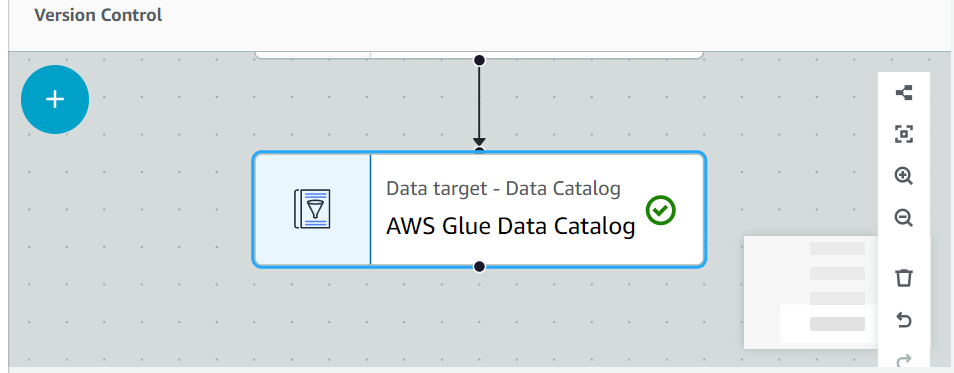
Here acc to RDS table schema we are changing this Metadata table schema We are changing following keys

* Subscription date – from (Data type) string to Date
* Date – (data type) from string to Date
* Monthlycharge, customerid – (data type) from long to int

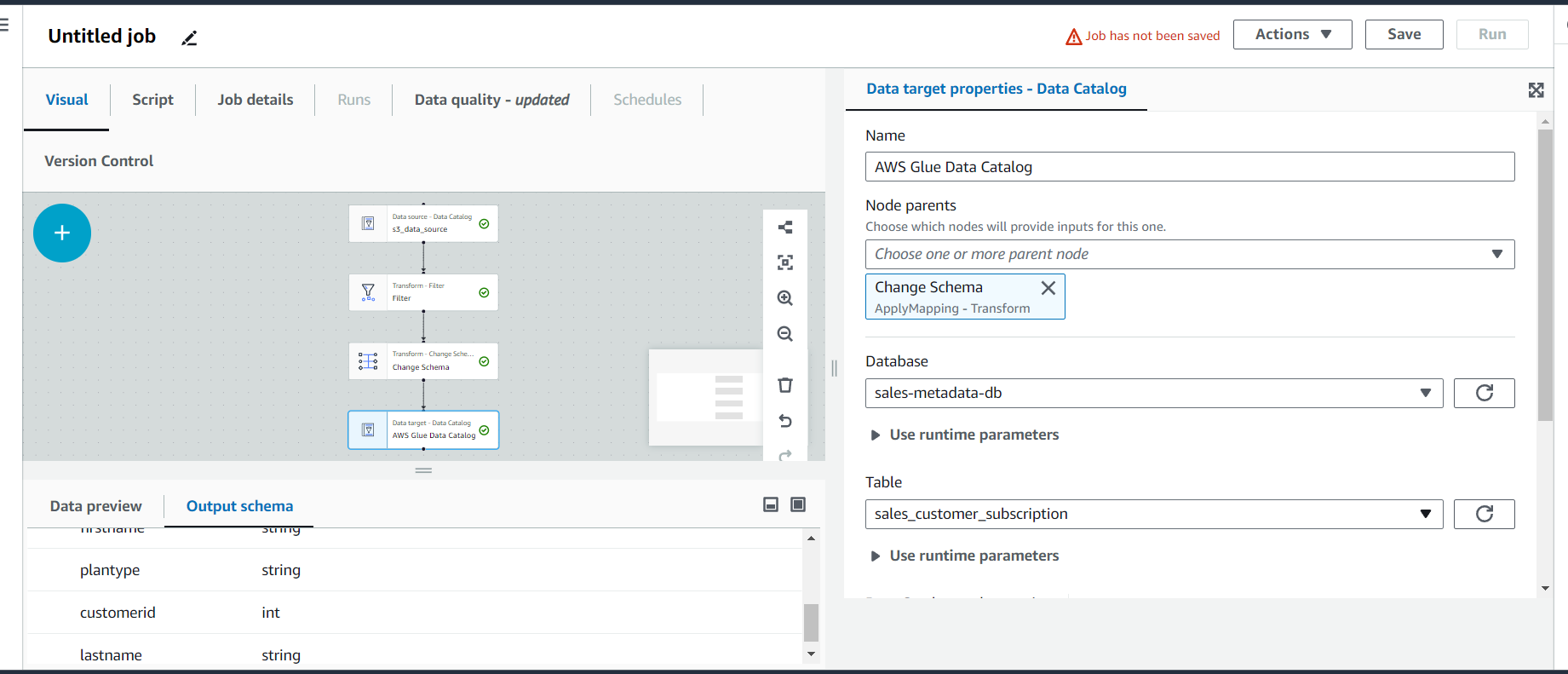


🡪**Target**

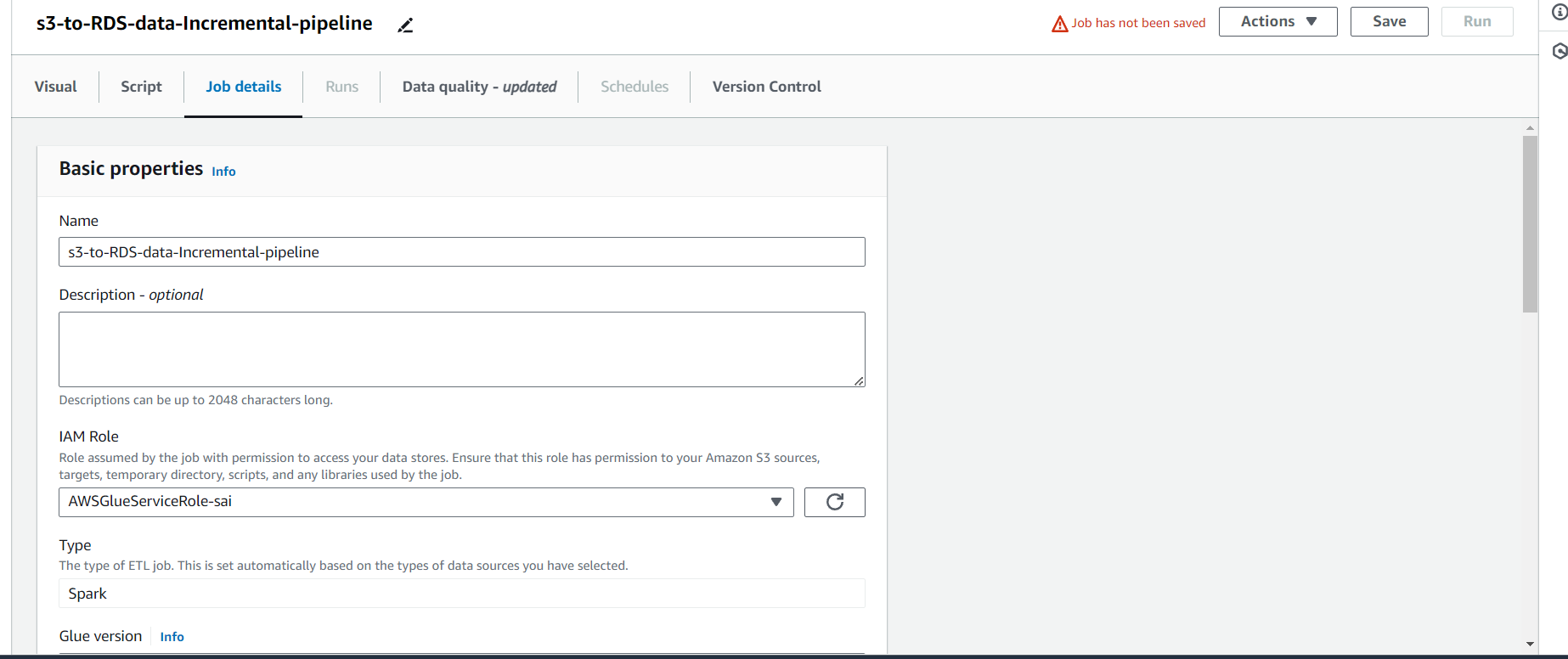
Add target and



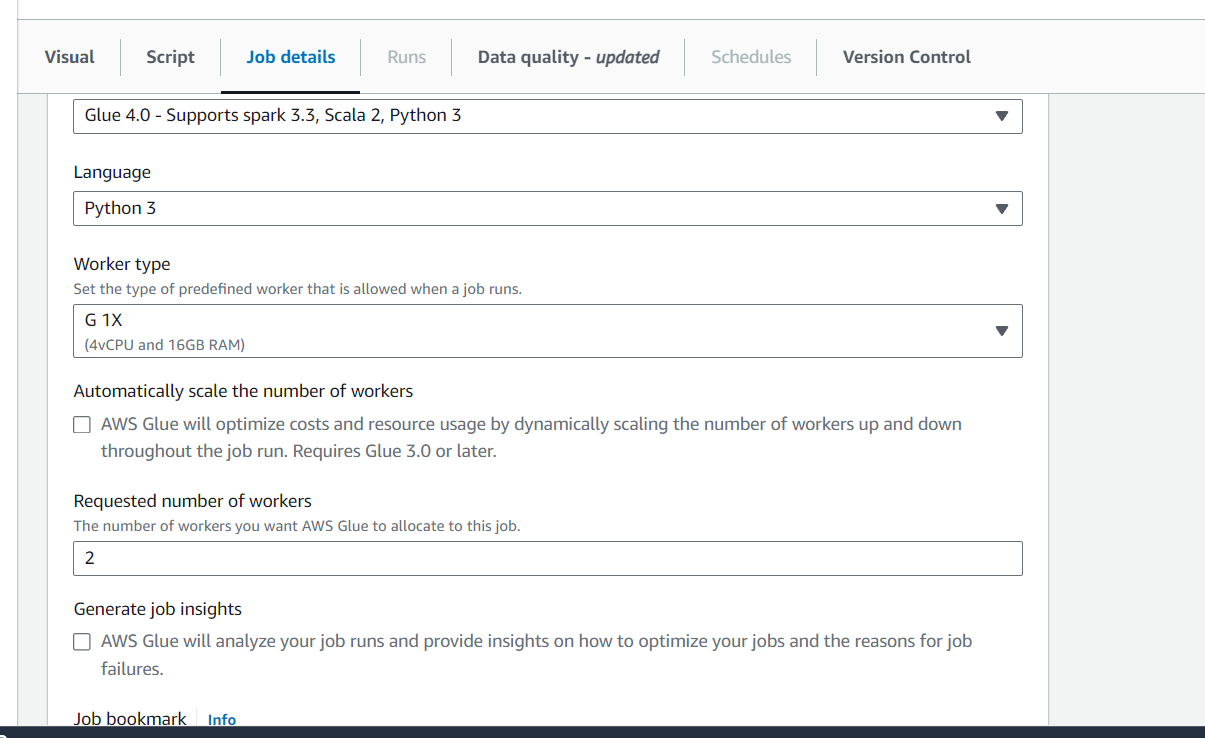
Here choose database and table.



And next go to Job details and add following name it something then choose IAM role



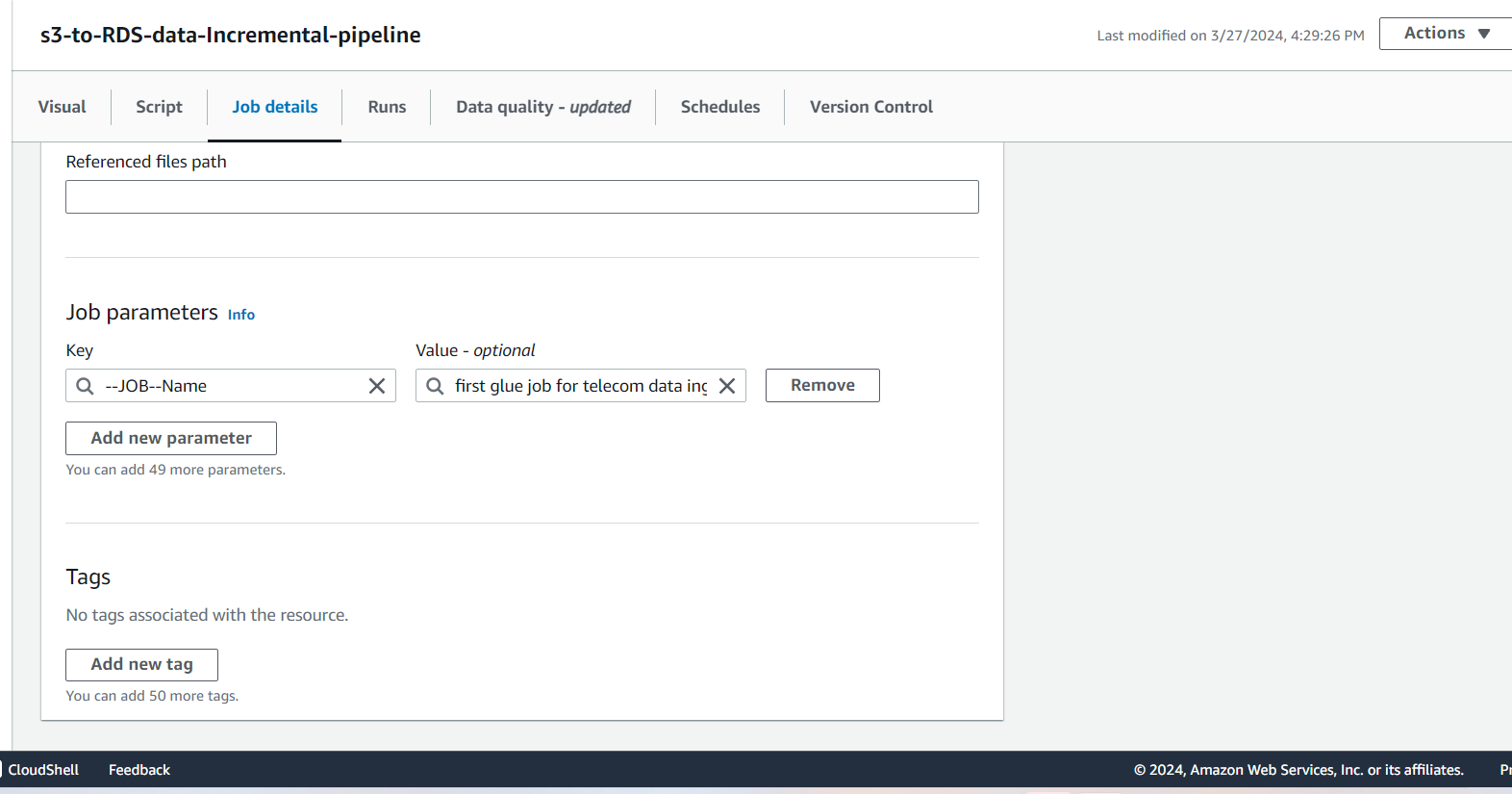
We can select workers as per our wish



Next Job book mark Enable it

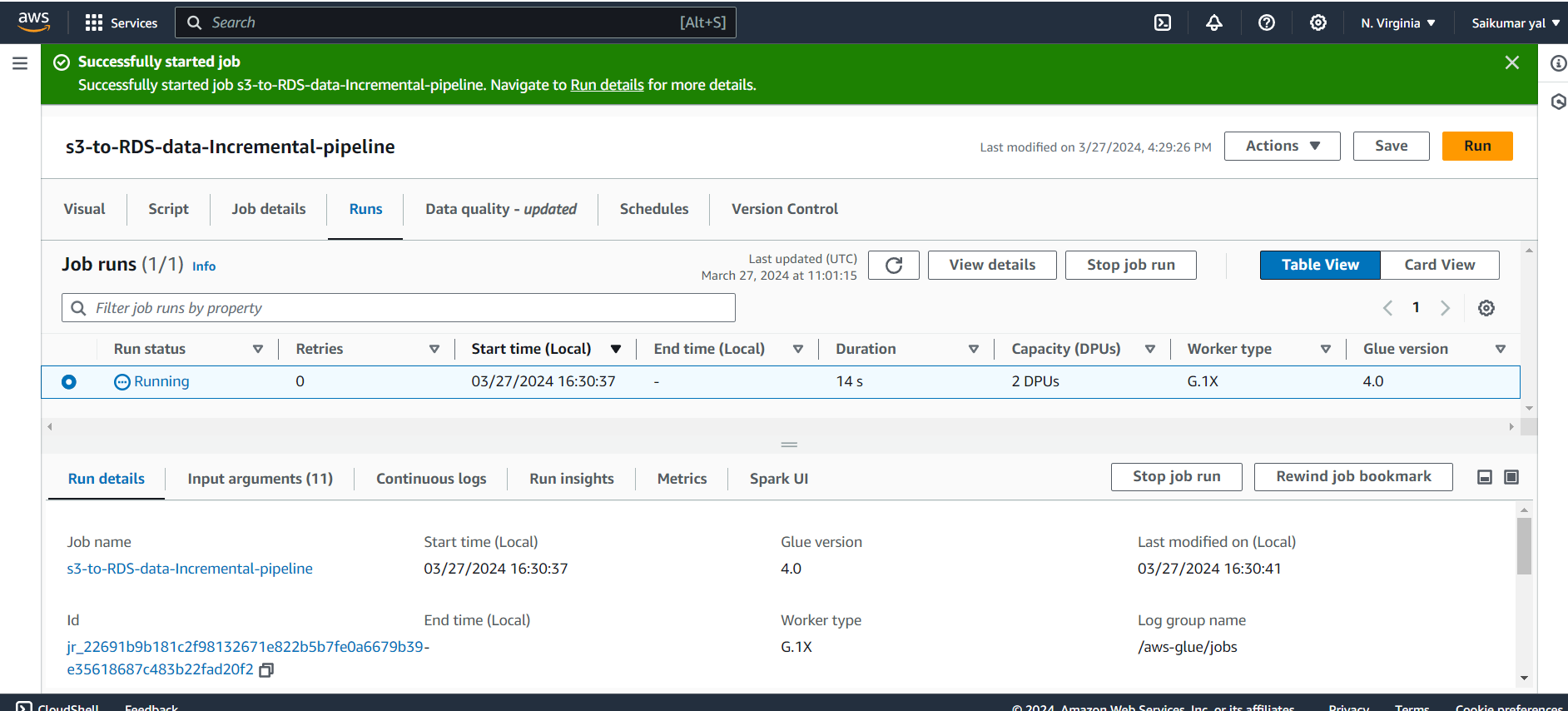


Job parameters optional

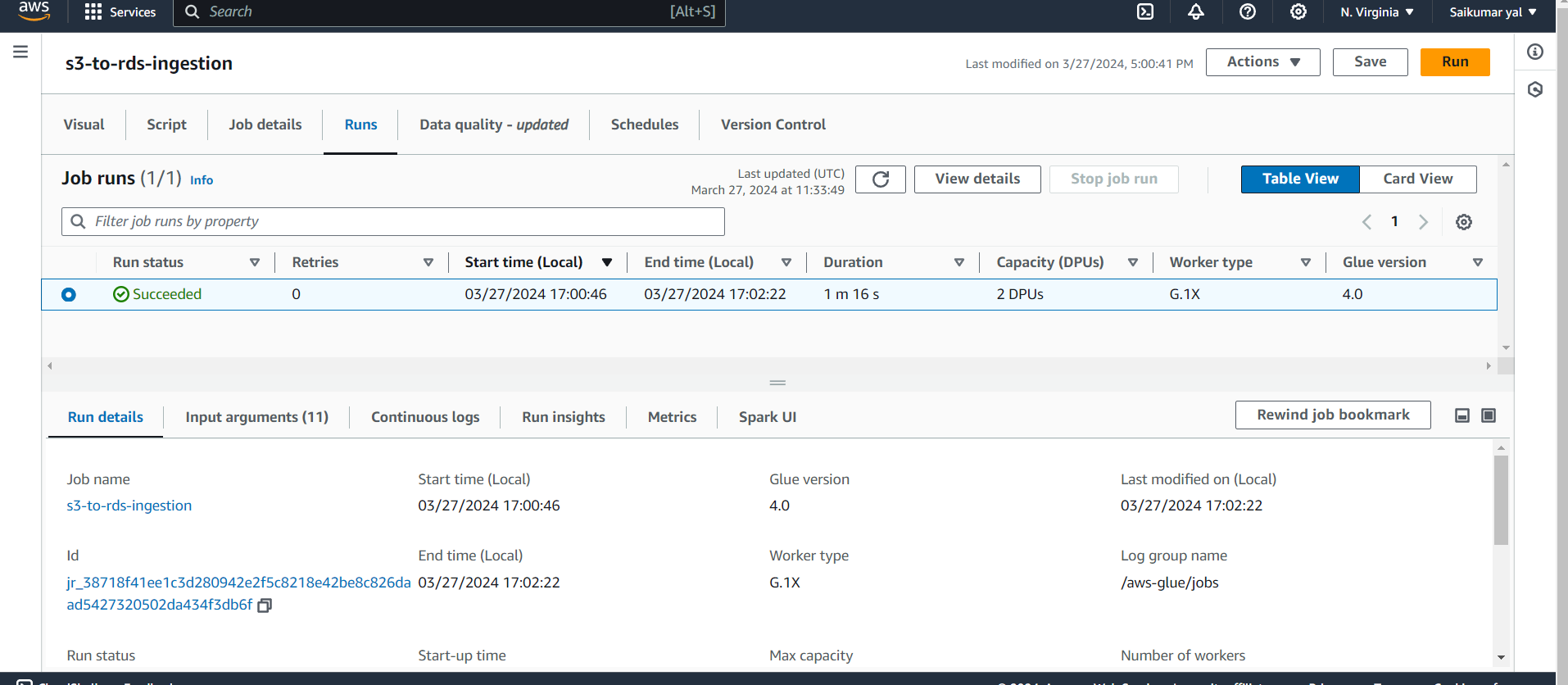


And save 🡪 next run it

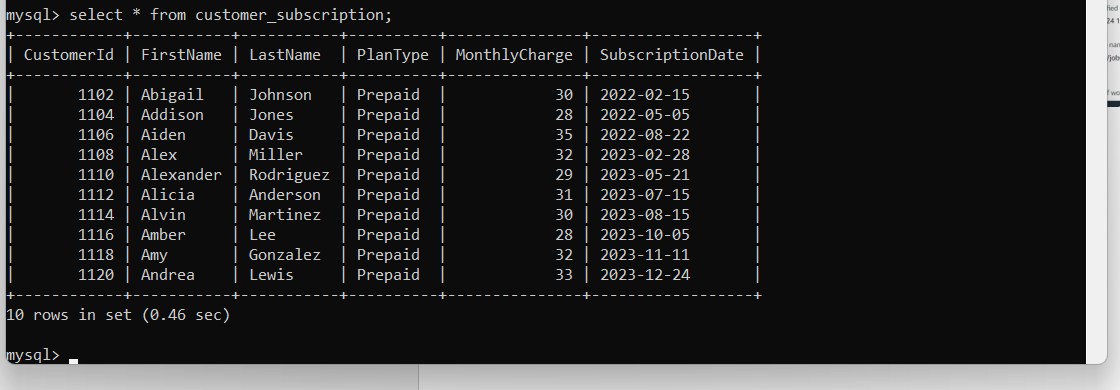
Go to Runs we can see execution details



It’s success mean Etl job has been Executed



Now go to cmd and type In🡪 sales\_db 🡪(Select\* from customer\_subscription) we can see only Prepaid records are received from file



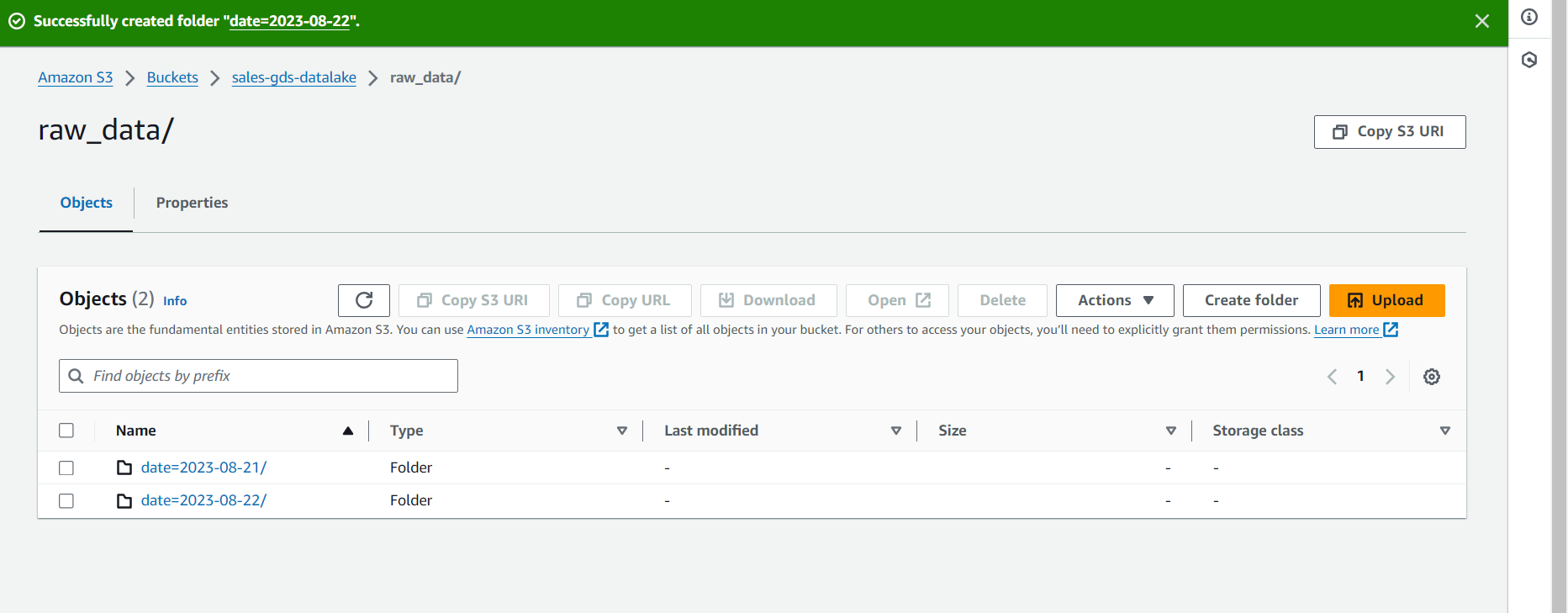
Upto this the data ingestion pipeline is designed and completed .

Next we are testing it by is it working by uploading next day files that is it works in incremental manner so we are testing it

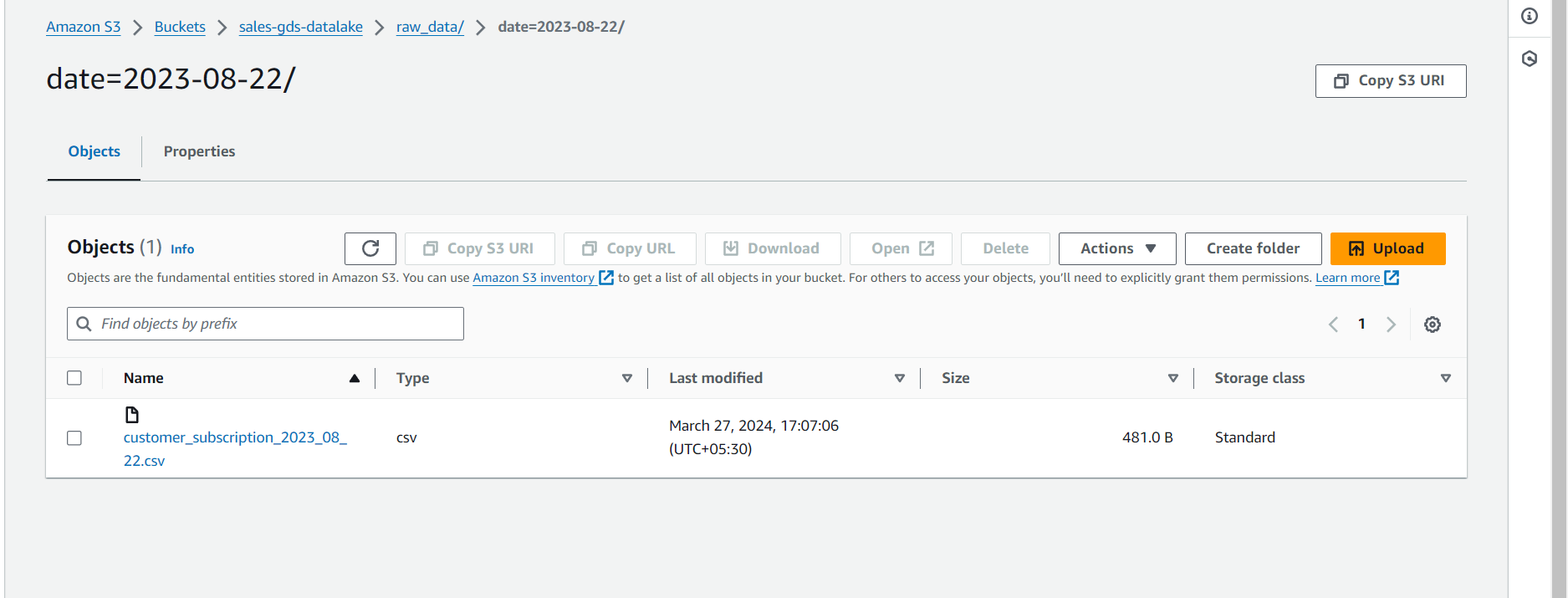
First create a folder in following path upload csv file in it.

S3>Buckets>sales\_data\_gds>raw\_data>create folder – folder name = date= 2023-08-22

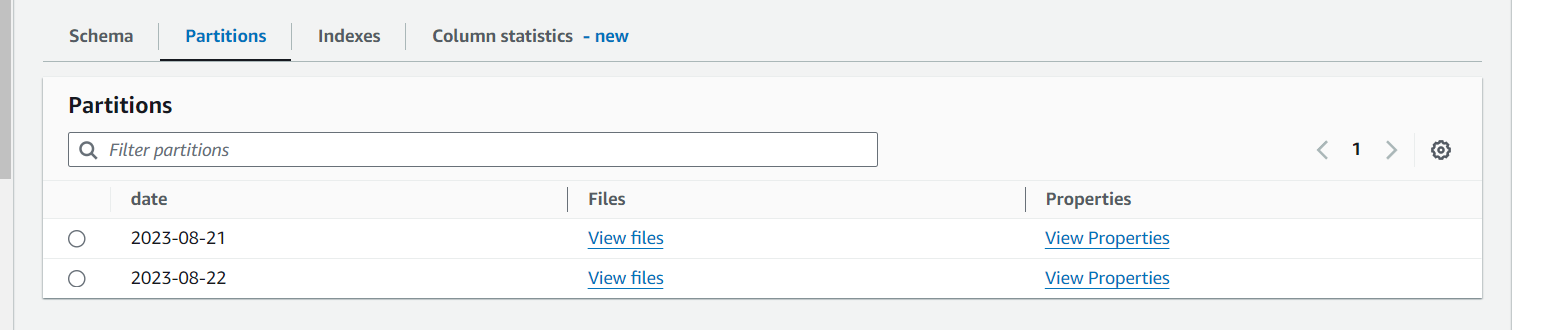
🡪



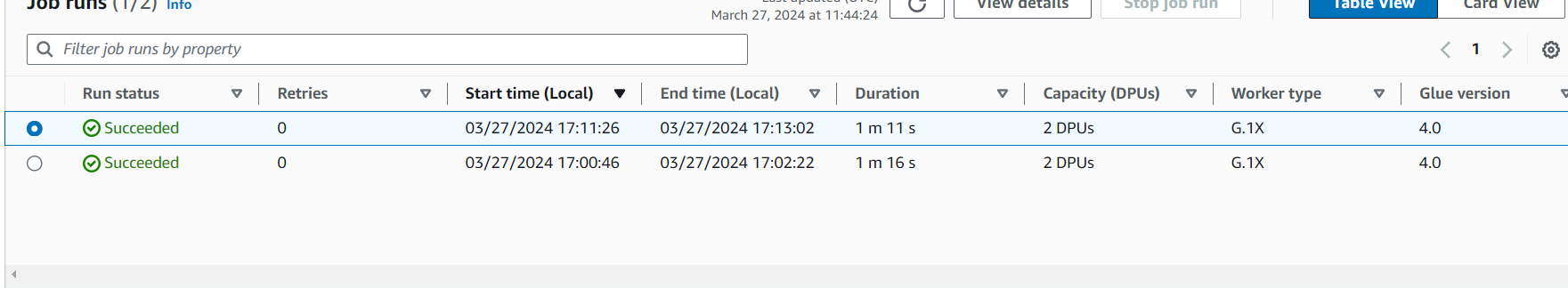
Next we can upload our 2nd csv file i.e 2023-08-22 in following location



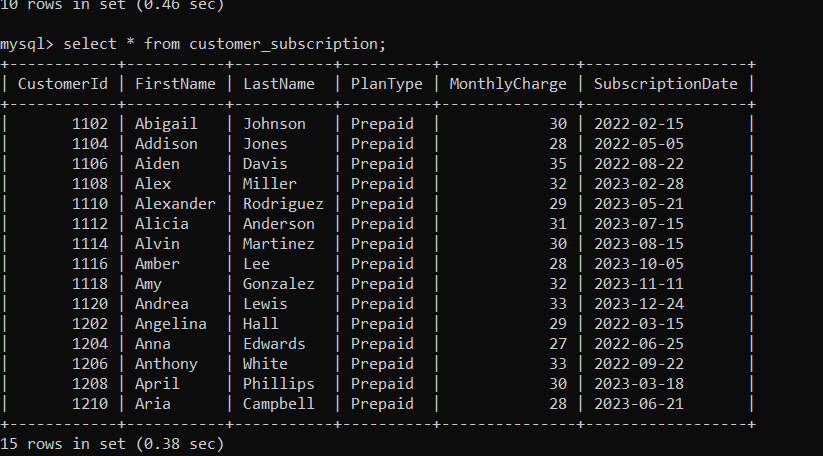
Then again Run the crawler (sales-s3-daily-data ) and go to table(s3\_inputraw\_data) we can see 1 updated in table changes and we see partitions are updated it became 2 .



Now again go to ETL Job and run it again



Again go to Cmd do select\*from customer\_subscription from sales\_db , we can see prepaid records from 2nd csv file we uploaded so here 5 records from it , total 15 rows 🡪10 rows from 1st csv + 5 rows from 2nd csv it’s filtered.



Next we will upload another csv file 3rd csv file in that path

First create a folder in following path upload csv file in it.

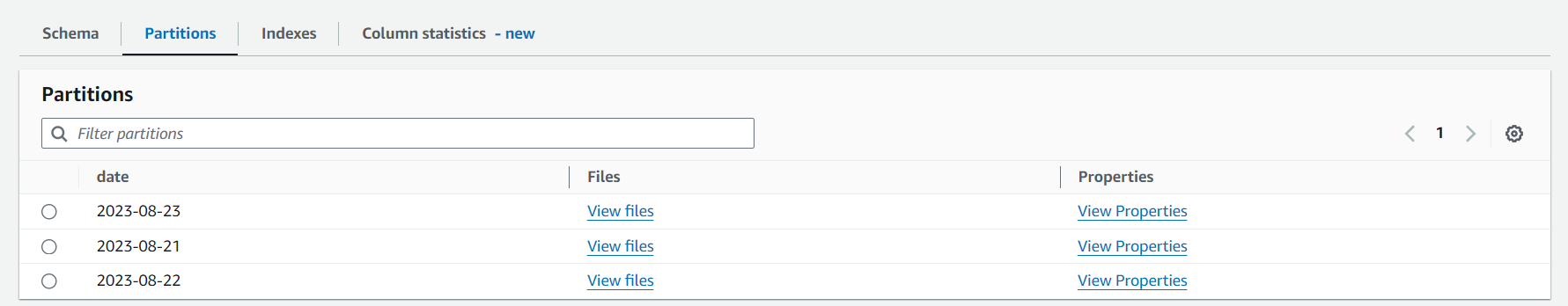
S3>Buckets>sales\_data\_gds>raw\_data>create folder – folder name = date= 2023-08-23

🡪

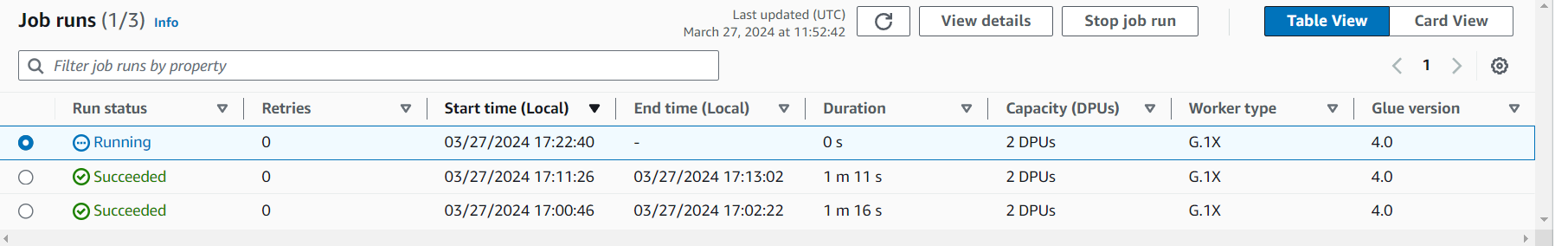


And repeat same

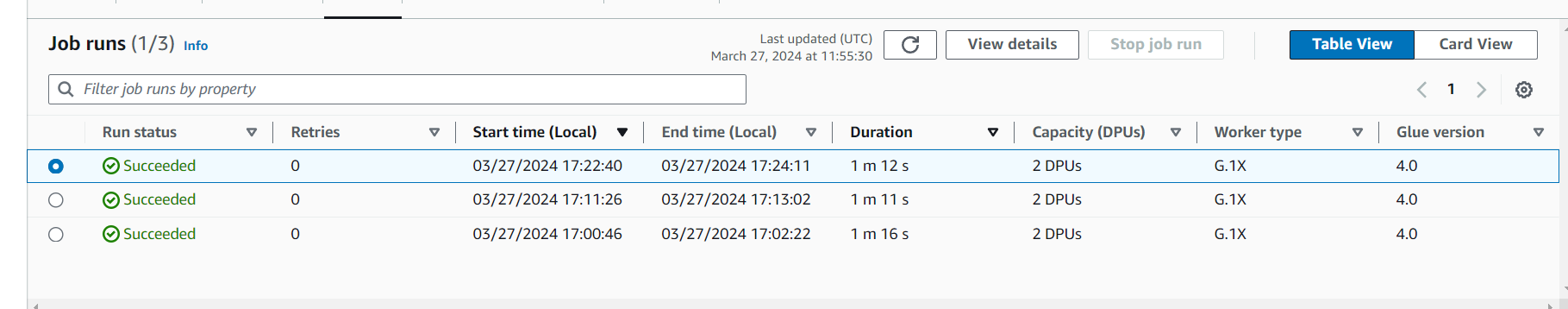
Then again Run the crawler (sales-s3-daily-data ) and go to table(s3\_inputraw\_data) we can see 2 updated in table changes and we see partitions are updated it became 3.



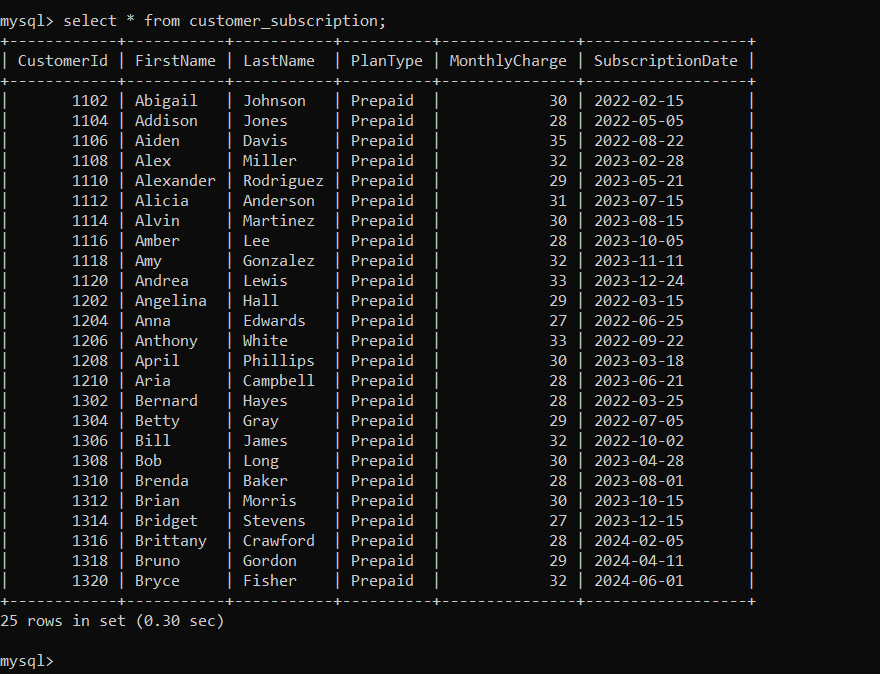
Next again go to ETL job 🡪 Run it



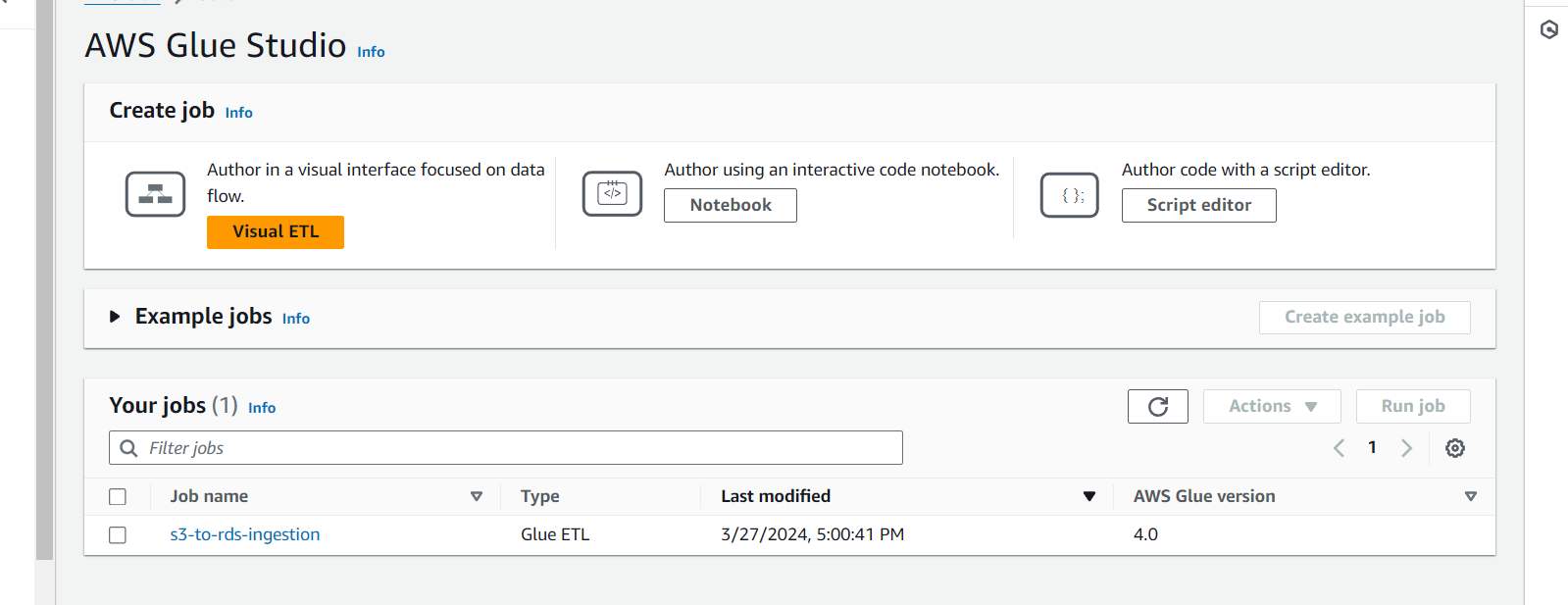
Succeded



Again go to Cmd do (select\*from customer\_subscription) from sales\_db , we can see prepaid records from 3rd csv file we uploaded so here 10 records from it , total 25 rows 🡪10 rows from 1st csv + 5 rows from2nd csv and 10 rows from 3rdcsv it’s filtered.



We can see Job we created in Jobs



---------- ------- -------------- ------------- ----- ---- --- END ---- ----------- ------- ------- ------- -------------