INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT AKURDI, PUNE

Documentation On

"Subscriber Churn Analysis for Telecom Services"

PG-DBDA September 2023

Submitted by-

Group No: 23

Roll No.	Name:
239552	Yash Tulaskar
239518	Harshal Meshram

Dr. Shantanu Pathak
Project Guide

Mr. Rohit Puranik
Centre Coordinator

Abstract

The Subscriber Churn Analysis project for Telecom Services aims to address the critical issue of customer retention within the telecommunications industry. Leveraging a comprehensive suite of AWS services, the project orchestrates a seamless flow of data from storage to analysis, enabling telecom providers to gain valuable insights into customer behavior and churn patterns.

The project begins by ingesting and storing data in Amazon S3, a scalable and secure cloud storage solution. AWS Glue, a fully managed extract, transform, and load (ETL) service, is then utilized to process the data. This ETL process is orchestrated using Apache Airflow, a platform for programmatically authoring, scheduling, and monitoring workflows.

Once processed, the data is loaded into Amazon Redshift, a fully managed data warehouse service, where it is transformed into actionable insights through visualization tools. These insights enable telecom providers to identify trends, patterns, and potential churn indicators, empowering them to implement targeted retention strategies.

To automate and streamline the workflow, the project utilizes Apache Airflow to schedule and execute the ETL jobs. By establishing connections between AWS services and configuring job definitions, the project ensures a smooth and efficient data pipeline.

Furthermore, the project addresses technical challenges such as configuring security settings, establishing connections between services, and troubleshooting errors. Through careful planning and implementation, these challenges are overcome to ensure the successful execution of the project.

Overall, the Subscriber Churn Analysis project for Telecom Services demonstrates the power of cloud computing and data analytics in addressing real-world business challenges. By leveraging AWS services and best practices in data engineering, the project delivers actionable insights that drive informed decision-making and improve customer retention strategies within the telecommunications industry.

ACKNOWLEDGEMENT

I take this occasion to thank God, almighty for blessing us with his grace and taking our endeavour to a successful culmination. I extend my sincere and heartfelt thanks to our esteemed guide, **Dr. Shantanu Pathak** for providing me with the right guidance and advice at the crucial juncture sand for showing me the right way. I extend my sincere thanks to our respected **Centre Co- Ordinator Mr. Rohit Puranik**, for allowing us to use the facilities available. I would like to thank the other faculty members also, at this occasion. Last but not the least, I would like to thank my friends and family for the support and encouragement they have given me during the course of our work.

Harshal R. Meshram
(230941225018)

Yash P. Tulaskar
(230941225051)

Table of Contents

Table of Figures

INTRODUCTION

In today's highly competitive telecommunications industry, customer retention is paramount. Telecom service providers face the ongoing challenge of reducing churn rates and retaining valuable subscribers. Understanding customer behavior and identifying churn indicators are essential for implementing effective retention strategies.

The Subscriber Churn Analysis project for Telecom Services aims to address this challenge by leveraging advanced data analytics techniques and AWS cloud services. By analyzing customer data, including usage patterns, service preferences, and interaction history, the project seeks to uncover insights that can inform targeted retention efforts.

This introduction provides an overview of the project's objectives, methodologies, and significance within the telecommunications landscape. It highlights the importance of data-driven decision-making and the role of advanced analytics in addressing customer churn.

Through the utilization of AWS services such as Amazon S3, AWS Glue, and Amazon Redshift, the project demonstrates the power of cloud computing in managing and analyzing vast datasets. Apache Airflow is employed to orchestrate data workflows, ensuring efficient processing and analysis.

The introduction sets the stage for the subsequent sections, which delve into the project's implementation details, technical considerations, and outcomes. By leveraging cutting-edge technologies and best practices in data engineering, the Subscriber Churn Analysis project aims to provide telecom service providers with actionable insights to enhance customer retention strategies and drive business success.

Purpose:
The purpose of our project is to analyze subscriber churn within the telecom industry using advanced data analytics techniques and AWS cloud services. By identifying churn indicators and customer behavior patterns, we aim to provide insights that enable telecom service providers to implement targeted retention strategies.

Scope:
Our project encompasses analyzing subscriber churn within the telecom industry using AWS services. This includes ingesting and processing data, orchestrating workflows, analyzing trends in customer behavior, and providing actionable insights to improve retention strategies.

Objective of Project:
1. Identify churn indicators and patterns in subscriber behavior within the telecom industry.
2. Utilize AWS services to ingest, process, and analyze large datasets efficiently.
3. Orchestrate data workflows using Apache Airflow for seamless ETL processes.
4. Analyze data in Amazon Redshift to uncover actionable insights for improving customer retention strategies.
5. Implement automated processes to streamline data pipelines and enhance efficiency.
6. Provide telecom service providers with valuable insights to optimize retention efforts and minimize churn rates.

Functionalities provided as follows:
The "Subscriber Churn Analysis for Telecom Services" project provides a range of functionalities. Some of the key functionalities offered by the project include:

1. Data Ingestion: Ability to ingest and store customer data from various sources into Amazon S3.
2. Data Processing: Utilize AWS Glue for ETL (Extract, Transform, Load) processes to clean, transform, and prepare data for analysis.
3. Workflow Orchestration: Employ Apache Airflow to schedule and monitor data workflows, ensuring seamless execution of tasks.
4. Data Analysis: Utilize Amazon Redshift for analyzing large datasets and uncovering insights related to subscriber churn and behavior patterns.
5. Automation: Implement automated processes for data ingestion, processing, and analysis to minimize manual intervention and improve efficiency.
6. Error Handling: Incorporate error handling mechanisms to identify and resolve issues during data processing and analysis.
7. Scalability: Design the project to be scalable, allowing for the handling of large volumes of data and accommodating future growth.
8. Security: Implement security measures to ensure the confidentiality, integrity, and availability of data throughout the project lifecycle.

AWS Architecture

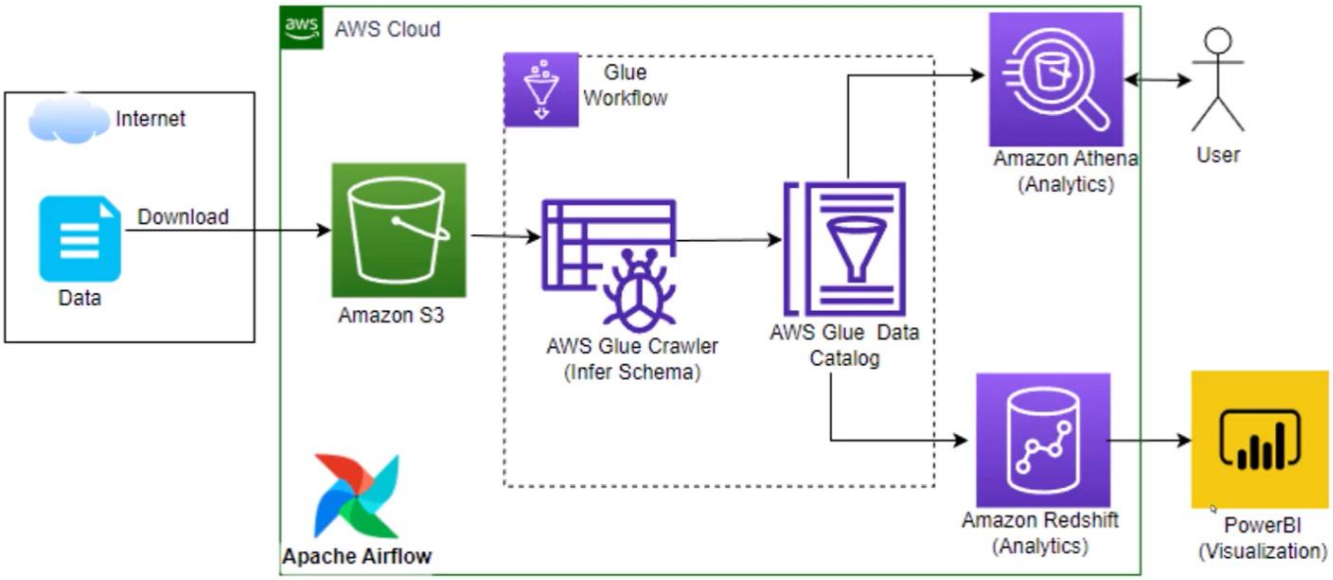

Fig.1 Architecture

Services Used:

1. Amazon S3: Amazon S3 is an object storage service that provides manufacturing scalability, data availability, security,and performance.

2. AWS IAM: This is nothing but identity and access management which enables us to manage access to AWSservices and resources securely.

3. AWS Glue: A serverless data integration service that makesit easy to discover, prepare, and combine data for analytics, machine learning, and application development.

4. AWS Athena: Athena is an interactive query service for S3 inwhich there is no need to load data it stays in S3.

5. AWS Redshift: Amazon Redshift is a data warehouse product which forms part of the larger cloud-computing platform Amazon Web Services. It is built on top of technology from the massive parallel processing data warehouse company ParAccel, to handle large scale data sets and database migrations.

6. AWS Elastic Compute Cloud(EC2): Amazon Elastic Compute Cloud is a part of Amazon.com's cloud-computing platform, Amazon Web Services, that allows users to rent virtual computers on which to run their own computer applications.

Dataset

- Context: A fictional telco company serving 7043 customers in California during Q3.
- Data Description: 7043 observations with 33 variables.
- Customer Information: Demographic details, tenure, and geographical information.
- Service Subscriptions: Details of phone, internet, and additional services.
- Financial Metrics: Monthly and total charges, churn-related metrics, and CLTV prediction.
- Churn Reason: Specific reasons for customer churn.
- Comprehensive dataset for analyzing customer demographics, service subscriptions, financial metrics, and churn-related factors.

Identity Access Management

A web service that helps to securely control access to AWS resources. Shared access to your AWS account. Granular permissions. Roles are created by AWS Management Console.Then Policies are attached to manage granular access. An IAM account is created with custom password and admin access in AWS console.

The access key and secret key are generated and the accesskey credentials are downloaded.

Then the AWS CLI is set up using the access key credentials and configured with the access key, secret key, and region. Theconfiguration is verified by running a command in AWS CLI.

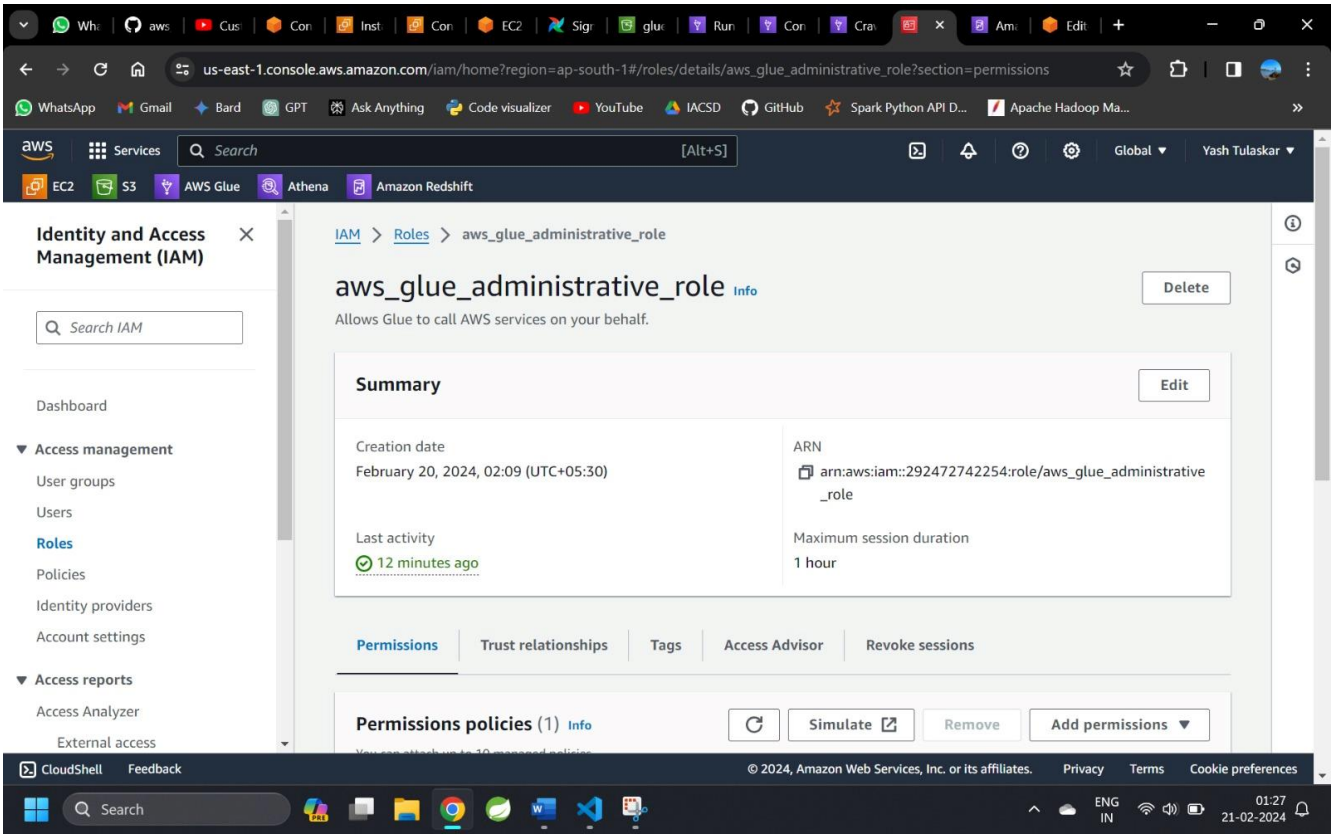

Fig.2- IAM role

Simple Storage Service S3:

The object storage service to store and protect any amount of data. It includes use cases, such as data lakes, websites, big data analytics etc. S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements. Seamlessly integrate and move data.

Upload data to S3 bucket using AWS CLI command.

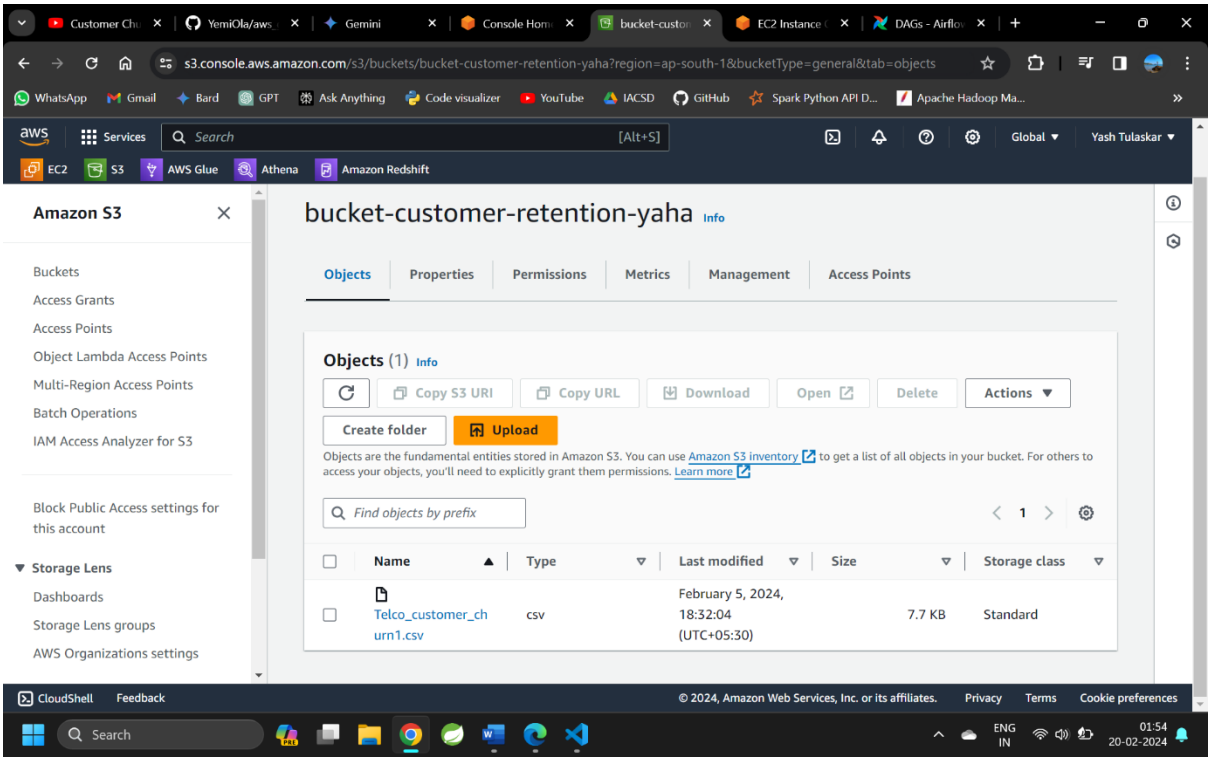

Fig.3- S3 Bucket With CSV file

Amazon EC2: Your Cloud-Based Virtual Server Powerhouse

An Amazon Elastic Compute Cloud (Amazon EC2) instance is a virtual server running in the Amazon Web Services (AWS) cloud. It provides on-demand computing capacity, allowing you to easily scale your infrastructure up or down as needed. Think of it as renting a computer instead of buying and maintaining your own hardware.

Benefits:

- **Scalability:** Scale your computing resources on demand, without having to manage physical servers.
- **Cost-Effectiveness:** Pay only for the resources you use, eliminating upfront hardware costs.
- **Security:** Leverage AWS's robust security infrastructure and tools to protect your data.
- **Reliability:** Enjoy Amazon's high uptime and performance guarantees.
- **Variety:** Choose from a wide range of instance types optimized for different workloads, including CPU, memory, storage, and network needs.
- **Flexibility:** Select various operating systems, software configurations, and network settings.

Use Cases:

- **Web applications:** Host your web applications with scaling capabilities and high availability.
- **Development and testing:** Set up secure and isolated environments for development, testing, and staging.
- **High-performance computing:** Use powerful instances for complex simulations and data analysis.
- **Machine learning:** Leverage GPU-enabled instances for training and deploying machine learning models.
- **Databases:** Run various database systems in the cloud with managed services or self-managed instances.

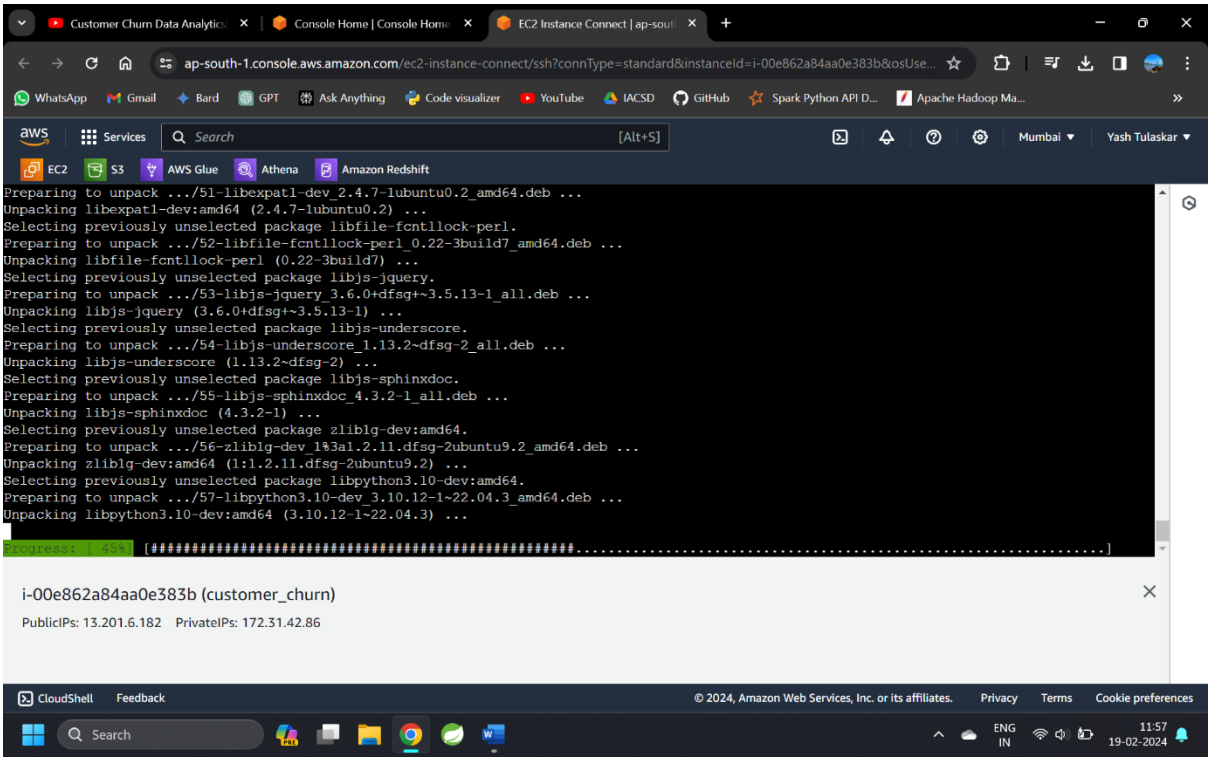

Fig.4- AWS EC2 created

Make virtual environment for a project in which we will install the other dependencies such Airflow and AWS Airflow service connectivity providers

Fig.5 – Virtual environment for project created

AWS Glue - Crawler

A crawler is used to populate the AWS Glue Data Catalog with tables. Crawlers can crawl multiple data stores in a single run. ETL jobs that you define in AWS Glue use these Data Catalog tables as sources and targets. AWS Glue Data Catalog is a managed metadata repository that stores and organizes metadata. It used to define the structure and schema of your data during glue etl job .

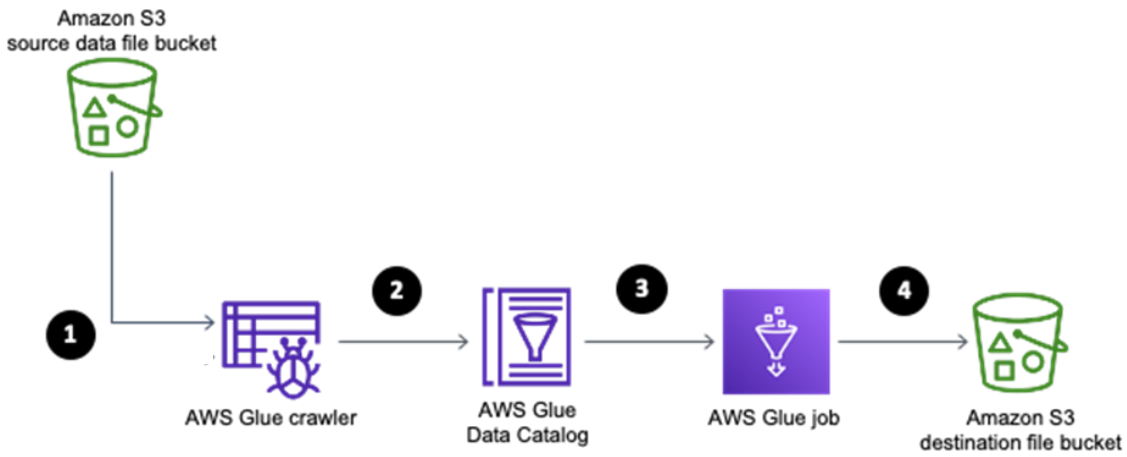

Fig.6- Crawler DataCatalog job structure

Crawler run on S3 on csv.

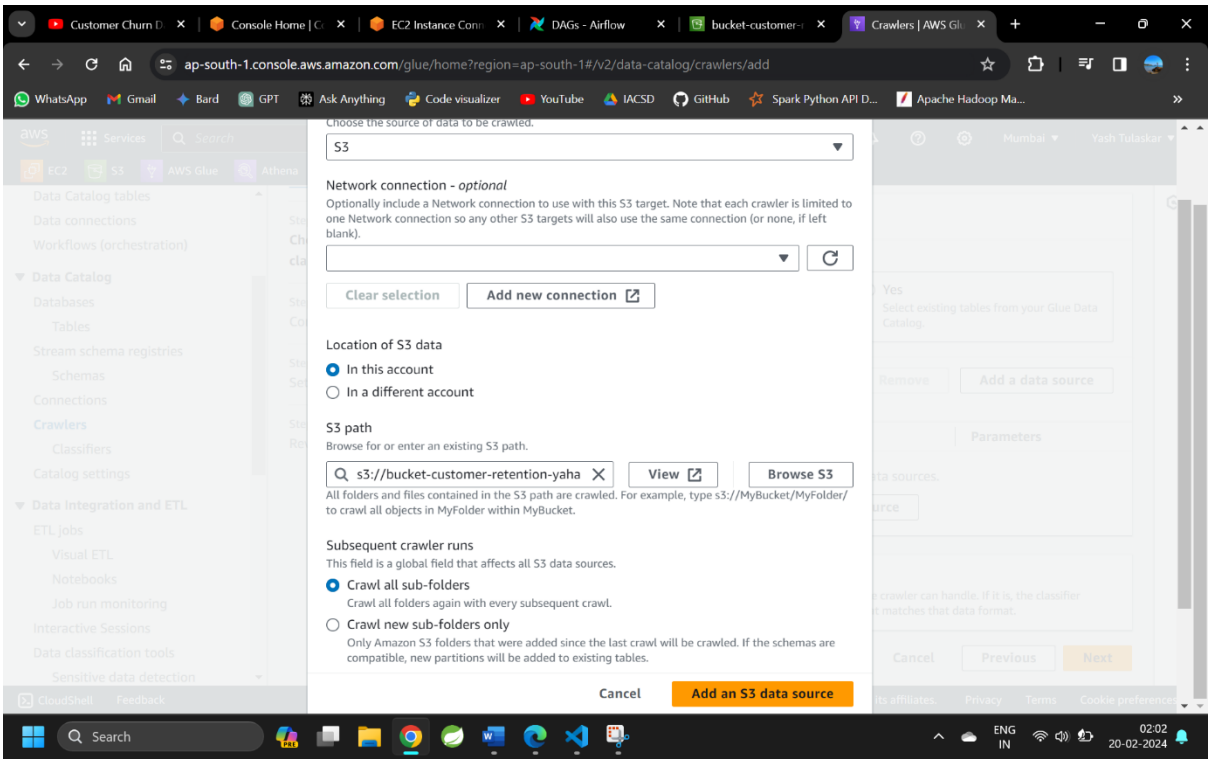

Fig. 7- Crawler run

Crawled Data Stored Tables

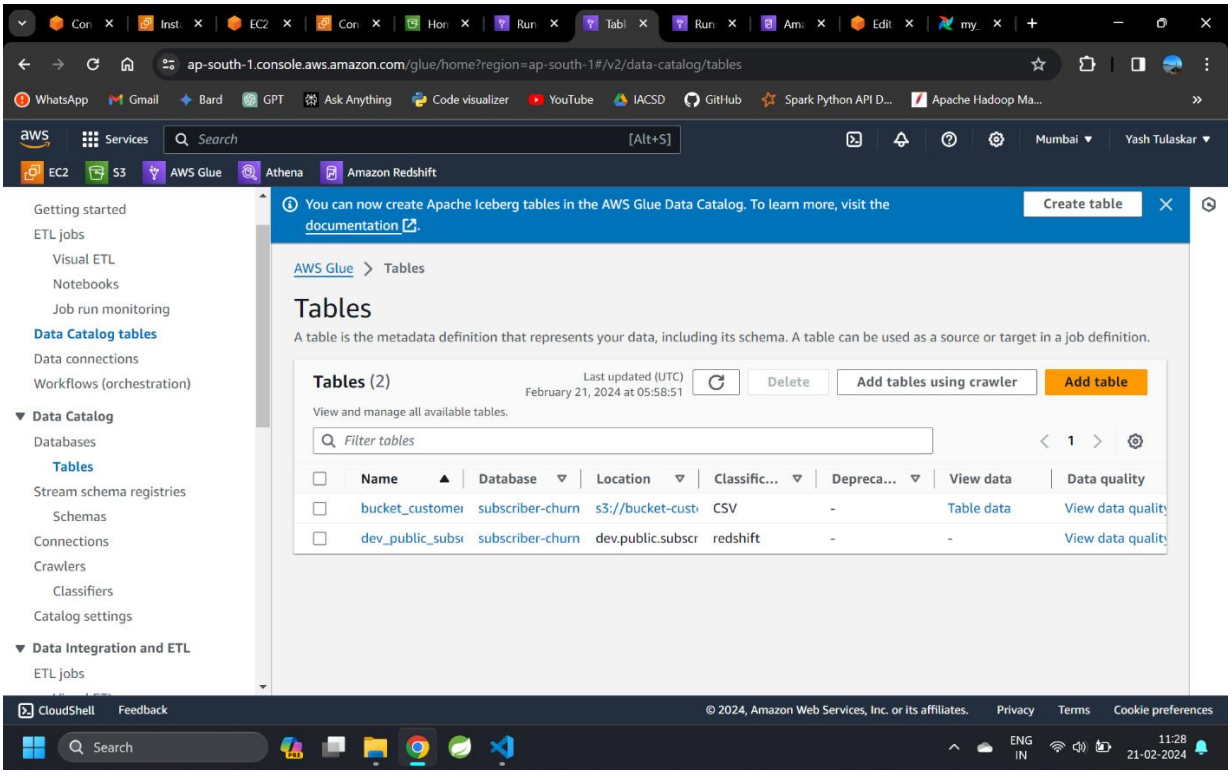

Fig. 8- Crawled Data Tables

AWS Glue

Glue is a serverless data integration service that makes it easierto discover, prepare, move, and integrate data from multiple sources. It simplifies ETL pipeline development. It supports various processing frameworks and workloads .

An ETL job was performed using AWS glue. The job extracted the raw .csv data from the s3 bucket through tables. Transformed the data by defining schema structure and predicate pushdown to select the region which filters the data before the source is extracted. It improves the ETL process andoptimizes resources used. Then we loaded the transformeddata to a new s3 bucket.

Following script was used to perform ETL process.

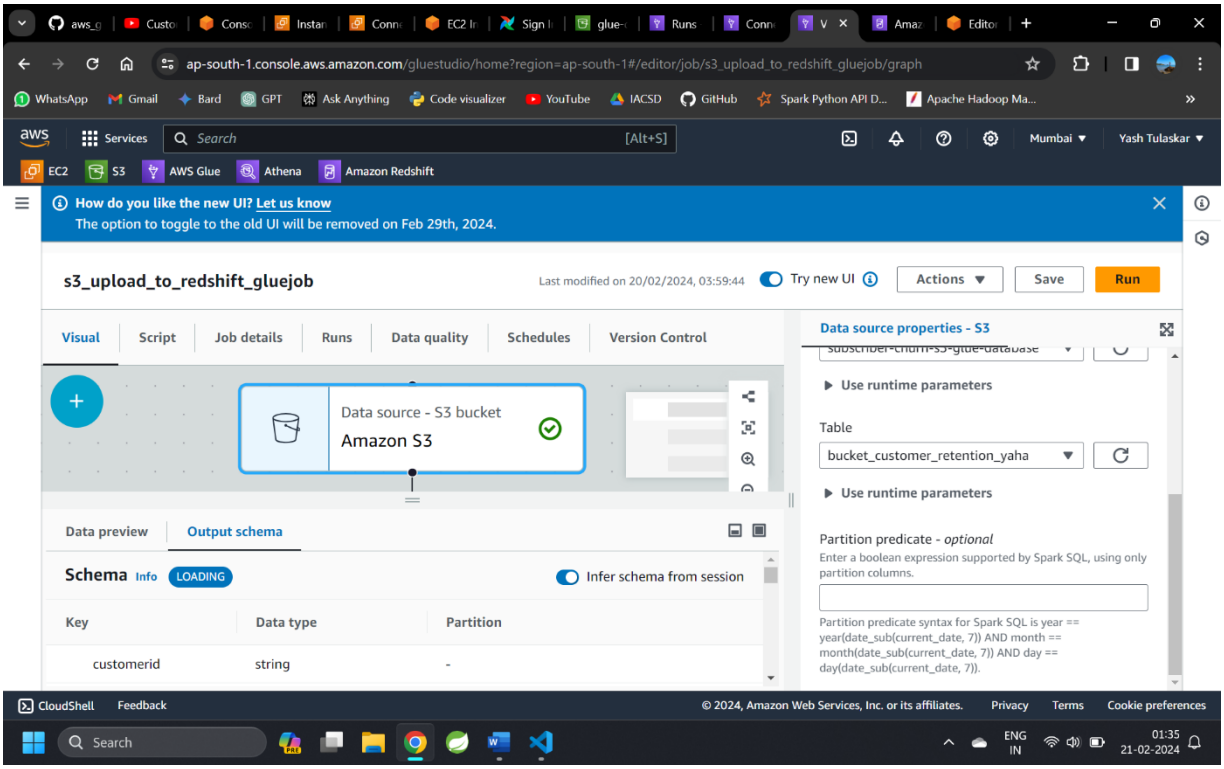

Fig. 9- ETL Extract glue job

Extract Code

```
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
from awsglue import DynamicFrame

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 Amazon S3
AmazonS3_node1708379590394 = glueContext.create_dynamic_frame.from_catalog(
    database="subscriber-churn-s3-glue-database",
    table_name="bucket_customer_retention_yaha",
    transformation_ctx="AmazonS3_node1708379590394",
)


# Script generated for node Change Schema
ChangeSchema_node1708379592351 = ApplyMapping.apply(
    frame=AmazonS3_node1708379590394,
    mappings=[
        ("customerid", "string", "customerid", "string"),
        ("city", "string", "city", "string"),
        ("zip code", "long", "zip_code", "int"),
        ("gender", "string", "gender", "string"),
        ("senior citizen", "string", "senior_citizen", "string"),
        ("partner", "string", "partner", "string"),
        ("dependents", "string", "dependents", "string"),
        ("tenure months", "long", "tenure_months", "int"),
        ("phone service", "string", "phone_service", "string"),
        ("multiple lines", "string", "multiple_lines", "string"),
        ("internet service", "string", "internet_service", "string"),
        ("online security", "string", "online_security", "string"),
        ("online backup", "string", "online_backup", "string"),
        ("device protection", "string", "device_protection", "string"),
        ("tech support", "string", "tech_support", "string"),
        ("streaming tv", "string", "streaming_tv", "string"),
```

```
        ("streaming movies", "string", "streaming_movies", "string"),
        ("contract", "string", "contract", "string"),
        ("paperless billing", "string", "paperless_billing", "string"),
        ("payment method", "string", "payment_method", "string"),
        ("monthly charges", "double", "monthly_charges", "double"),
        ("total charges", "double", "total_charges", "double"),
        ("churn label", "string", "churn_label", "string"),
        ("churn value", "long", "churn_value", "int"),
        ("churn score", "long", "churn_score", "int"),
        ("churn reason", "string", "churn_reason", "string"),
    ],
    transformation_ctx="ChangeSchema_node1708379592351",
)

# Script generated for node Amazon Redshift
AmazonRedshift_node1708379595732 =
glueContext.write_dynamic_frame.from_options(
    frame=ChangeSchema_node1708379592351,
    connection_type="redshift",
    connection_options={
        "redshiftTmpDir": "s3://aws-glue-assets-292472742254-ap-south-1/temporary/",
        "useConnectionProperties": "true",
        "dbtable": "public.subscriber_churn",
        "connectionName": "connect_redshift_ETL",
        "preactions": "CREATE TABLE IF NOT EXISTS public.subscriber_churn
(customerid VARCHAR, city VARCHAR, zip_code INTEGER, gender VARCHAR,
senior_citizen VARCHAR, partner VARCHAR, dependents VARCHAR, tenure_months
INTEGER, phone_service VARCHAR, multiple_lines VARCHAR, internet_service
VARCHAR, online_security VARCHAR, online_backup VARCHAR, device_protection
VARCHAR, tech_support VARCHAR, streaming_tv VARCHAR, streaming_movies
VARCHAR, contract VARCHAR, paperless_billing VARCHAR, payment_method
VARCHAR, monthly_charges DOUBLE PRECISION, total_charges DOUBLE
PRECISION, churn_label VARCHAR, churn_value INTEGER, churn_score INTEGER,
churn_reason VARCHAR); TRUNCATE TABLE public.subscriber_churn;",
    },
    transformation_ctx="AmazonRedshift_node1708379595732",
)

job.commit()
```

Transform Code

```
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
from awsglue import DynamicFrame

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 Amazon S3
AmazonS3_node1708379590394 = glueContext.create_dynamic_frame.from_catalog(
    database="subscriber-churn-s3-glue-database",
    table_name="bucket_customer_retention_yaha",
    transformation_ctx="AmazonS3_node1708379590394",
)

# Script generated for node Change Schema
ChangeSchema_node1708379592351 = ApplyMapping.apply(
    frame=AmazonS3_node1708379590394,
    mappings=[
        ("customerid", "string", "customerid", "string"),
        ("city", "string", "city", "string"),
        ("zip code", "long", "zip_code", "int"),
        ("gender", "string", "gender", "string"),
        ("senior citizen", "string", "senior_citizen", "string"),
        ("partner", "string", "partner", "string"),
        ("dependents", "string", "dependents", "string"),
        ("tenure months", "long", "tenure_months", "int"),
        ("phone service", "string", "phone_service", "string"),
        ("multiple lines", "string", "multiple_lines", "string"),
        ("internet service", "string", "internet_service", "string"),
        ("online security", "string", "online_security", "string"),
        ("online backup", "string", "online_backup", "string"),
        ("device protection", "string", "device_protection", "string"),
        ("tech support", "string", "tech_support", "string"),
        ("streaming tv", "string", "streaming_tv", "string"),
        ("streaming movies", "string", "streaming_movies", "string"),
        ("contract", "string", "contract", "string"),
```

```
        ("paperless billing", "string", "paperless_billing", "string"),
        ("payment method", "string", "payment_method", "string"),
        ("monthly charges", "double", "monthly_charges", "double"),
        ("total charges", "double", "total_charges", "double"),
        ("churn label", "string", "churn_label", "string"),
        ("churn value", "long", "churn_value", "int"),
        ("churn score", "long", "churn_score", "int"),
        ("churn reason", "string", "churn_reason", "string"),
    ],
    transformation_ctx="ChangeSchema_node1708379592351",
)

# Script generated for node Amazon Redshift
AmazonRedshift_node1708379595732 =
glueContext.write_dynamic_frame.from_options(
    frame=ChangeSchema_node1708379592351,
    connection_type="redshift",
    connection_options={
        "redshiftTmpDir": "s3://aws-glue-assets-292472742254-ap-south-1/temporary/",
        "useConnectionProperties": "true",
        "dbtable": "public.subscriber_churn",
        "connectionName": "connect_redshift_ETL",
        "preactions": "CREATE TABLE IF NOT EXISTS public.subscriber_churn
(customerid VARCHAR, city VARCHAR, zip_code INTEGER, gender VARCHAR,
senior_citizen VARCHAR, partner VARCHAR, dependents VARCHAR, tenure_months
INTEGER, phone_service VARCHAR, multiple_lines VARCHAR, internet_service
VARCHAR, online_security VARCHAR, online_backup VARCHAR, device_protection
VARCHAR, tech_support VARCHAR, streaming_tv VARCHAR, streaming_movies
VARCHAR, contract VARCHAR, paperless_billing VARCHAR, payment_method
VARCHAR, monthly_charges DOUBLE PRECISION, total_charges DOUBLE
PRECISION, churn_label VARCHAR, churn_value INTEGER, churn_score INTEGER,
churn_reason VARCHAR); TRUNCATE TABLE public.subscriber_churn;",
    },
    transformation_ctx="AmazonRedshift_node1708379595732",
)

job.commit()
```

Load Code

```
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
from awsglue import DynamicFrame

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 Amazon S3
AmazonS3_node1708379590394 = glueContext.create_dynamic_frame.from_catalog(
    database="subscriber-churn-s3-glue-database",
    table_name="bucket_customer_retention_yaha",
    transformation_ctx="AmazonS3_node1708379590394",
)

# Script generated for node Change Schema
ChangeSchema_node1708379592351 = ApplyMapping.apply(
    frame=AmazonS3_node1708379590394,
    mappings=[
        ("customerid", "string", "customerid", "string"),
        ("city", "string", "city", "string"),
        ("zip code", "long", "zip_code", "int"),
        ("gender", "string", "gender", "string"),
        ("senior citizen", "string", "senior_citizen", "string"),
        ("partner", "string", "partner", "string"),
        ("dependents", "string", "dependents", "string"),
        ("tenure months", "long", "tenure_months", "int"),
        ("phone service", "string", "phone_service", "string"),
        ("multiple lines", "string", "multiple_lines", "string"),
        ("internet service", "string", "internet_service", "string"),
        ("online security", "string", "online_security", "string"),
        ("online backup", "string", "online_backup", "string"),
        ("device protection", "string", "device_protection", "string"),
        ("tech support", "string", "tech_support", "string"),
        ("streaming tv", "string", "streaming_tv", "string"),
        ("streaming movies", "string", "streaming_movies", "string"),
```

```
        ("contract", "string", "contract", "string"),
        ("paperless billing", "string", "paperless_billing", "string"),
        ("payment method", "string", "payment_method", "string"),
        ("monthly charges", "double", "monthly_charges", "double"),
        ("total charges", "double", "total_charges", "double"),
        ("churn label", "string", "churn_label", "string"),
        ("churn value", "long", "churn_value", "int"),
        ("churn score", "long", "churn_score", "int"),
        ("churn reason", "string", "churn_reason", "string"),
    ],
    transformation_ctx="ChangeSchema_node1708379592351",
)

# Script generated for node Amazon Redshift
AmazonRedshift_node1708379595732 =
glueContext.write_dynamic_frame.from_options(
    frame=ChangeSchema_node1708379592351,
    connection_type="redshift",
    connection_options={
        "redshiftTmpDir": "s3://aws-glue-assets-292472742254-ap-south-1/temporary/",
        "useConnectionProperties": "true",
        "dbtable": "public.subscriber_churn",
        "connectionName": "connect_redshift_ETL",
        "preactions": "CREATE TABLE IF NOT EXISTS public.subscriber_churn
(customerid VARCHAR, city VARCHAR, zip_code INTEGER, gender VARCHAR,
senior_citizen VARCHAR, partner VARCHAR, dependents VARCHAR, tenure_months
INTEGER, phone_service VARCHAR, multiple_lines VARCHAR, internet_service
VARCHAR, online_security VARCHAR, online_backup VARCHAR, device_protection
VARCHAR, tech_support VARCHAR, streaming_tv VARCHAR, streaming_movies
VARCHAR, contract VARCHAR, paperless_billing VARCHAR, payment_method
VARCHAR, monthly_charges DOUBLE PRECISION, total_charges DOUBLE
PRECISION, churn_label VARCHAR, churn_value INTEGER, churn_score INTEGER,
churn_reason VARCHAR); TRUNCATE TABLE public.subscriber_churn;",
    },
    transformation_ctx="AmazonRedshift_node1708379595732",
)

job.commit()
```

Amazon Redshift:

Amazon Redshift is a fully managed, petabyte-scale data warehouse service offered by Amazon Web Services (AWS). It's designed to handle large datasets and enables you to analyze them efficiently and cost-effectively.

Key Features:

Scalability: Scale your data warehouse capacity seamlessly from gigabytes to petabytes to meet your growing needs.

Performance: Optimized for fast queries, allowing you to analyze your data in seconds or minutes, even for large datasets.

Cost-Effectiveness: Pay only for the resources you use with a pay-as-you-go pricing model.

Security: Benefit from AWS's robust security infrastructure and features to protect your data.

Ease of Use: Fully managed service with simple setup and minimal administration required.

Integrations: Works seamlessly with other AWS services like S3, EMR, and QuickSight for streamlined data processing and analysis.

Use Cases:

Business Intelligence and Analytics: Gain insights from your data for informed decision-making.

Data Warehousing: Store and analyze large datasets efficiently.

Machine Learning: Prepare data for machine learning training and analysis.

Financial Analysis: Analyze financial data for trends and forecasts.

Customer Insights: Understand your customers better through data analysis.

Compared to Traditional Data Warehouses:

Reduced complexity: No need to manage hardware, software, or infrastructure.

Faster time to insights: Get started quickly and access your data for analysis faster.

Lower costs: Pay only for the resources you use, eliminating upfront infrastructure costs.

Increased agility: Scale your data warehouse capacity on demand as your needs change.

Launch your Redshift cluster: Choose the instance type and configure your cluster based on your needs.

Load your data: Utilize various methods to load data into your Redshift cluster. Start querying your data: Use SQL to analyze your data and extract valuable insights. After connecting to the redshift cluster, we create it using username and password provisioning cluster, then we are connected with no tables, views, stored procedures, etc.

After connection create table in redshift with columns you wanted further for usage.

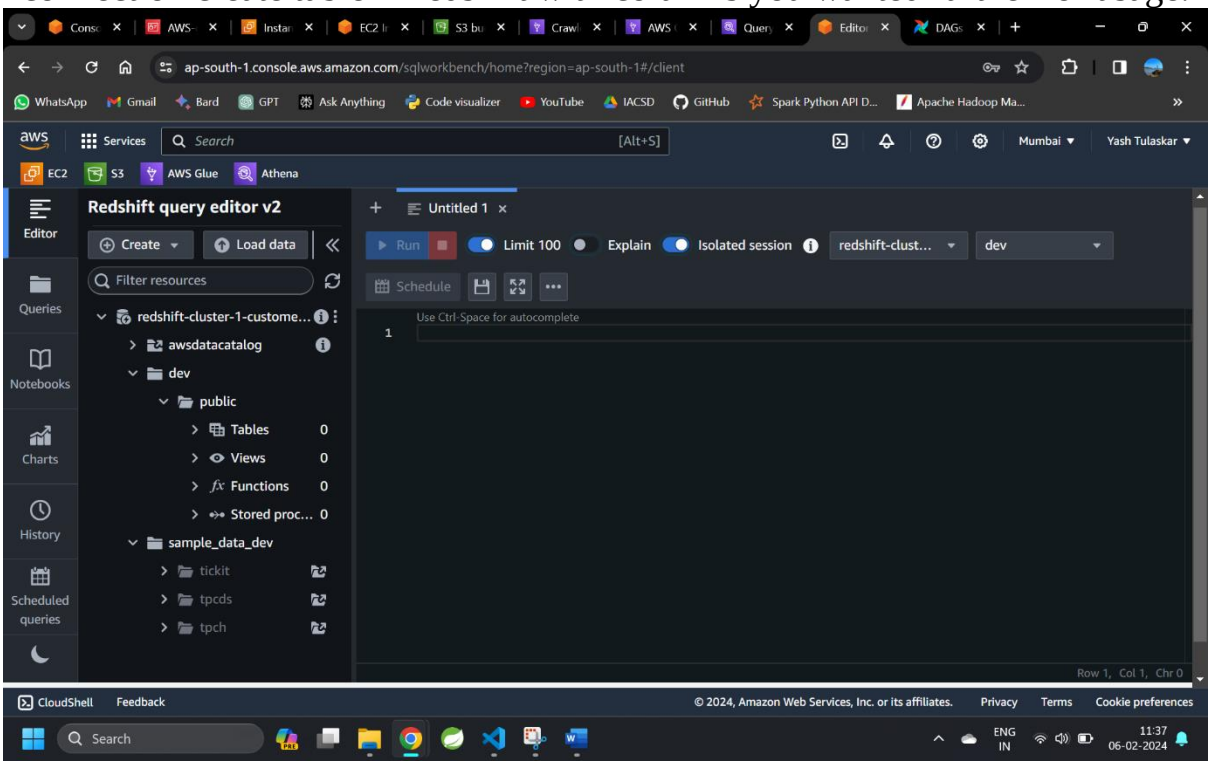

Fig. 10 Redshift cluster editor

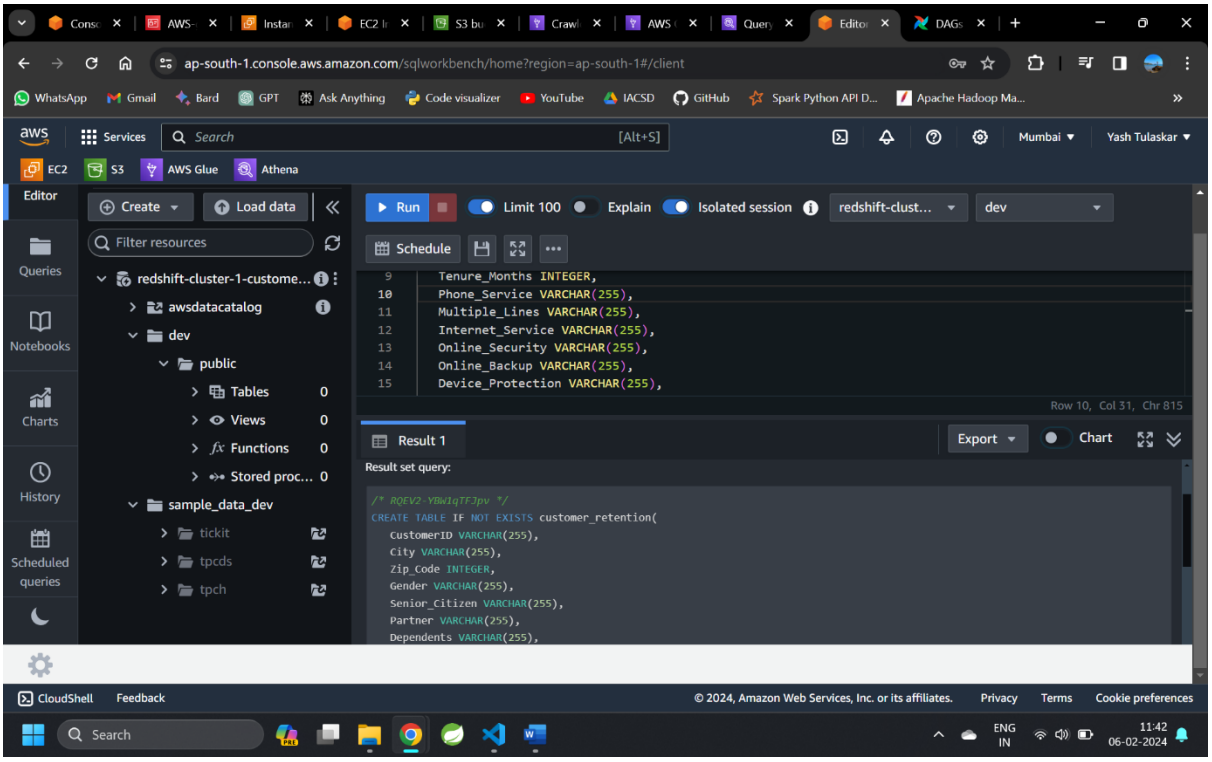

Fig. 11- Created schema in redshift

You can see some columns are not inserted over there as we don't need them further.

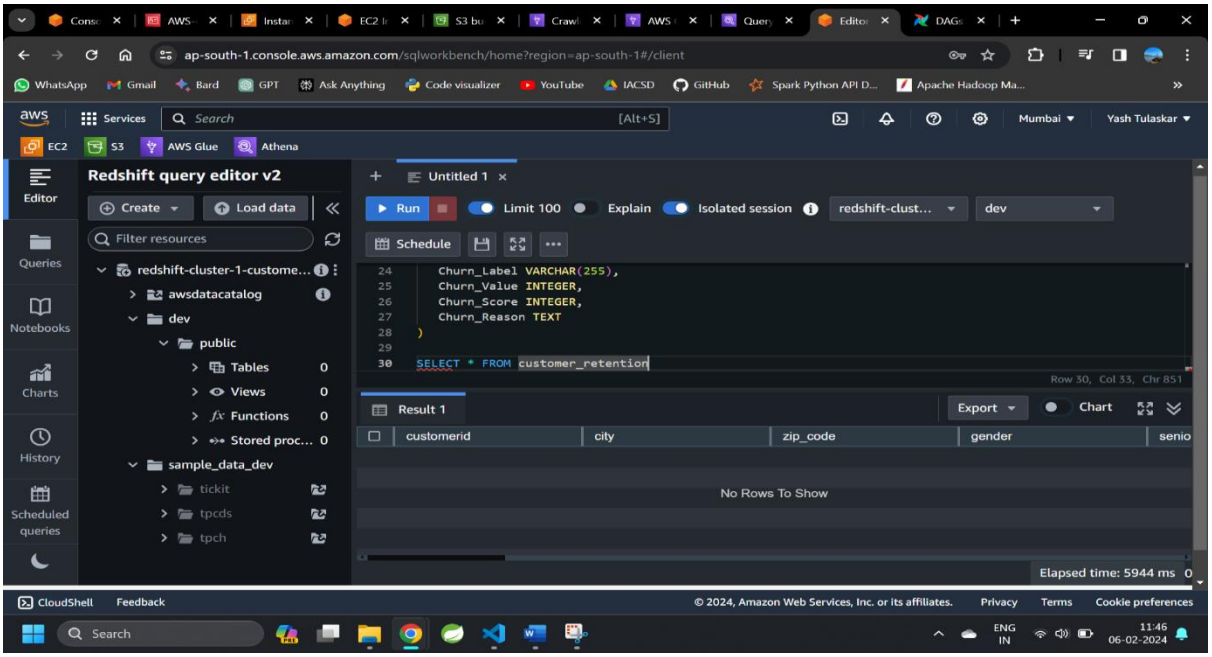

Fig. 12 schema created with empty rows

Now the table is created on the redshift cluster whereas it is empty, we need to be able to crawl this redshift cluster, so AWS Glue will now what schema is actually is on it.
Before we do that we need to establish connection to Redshift using Glue. So go to Glue in that Connections > Create Connections

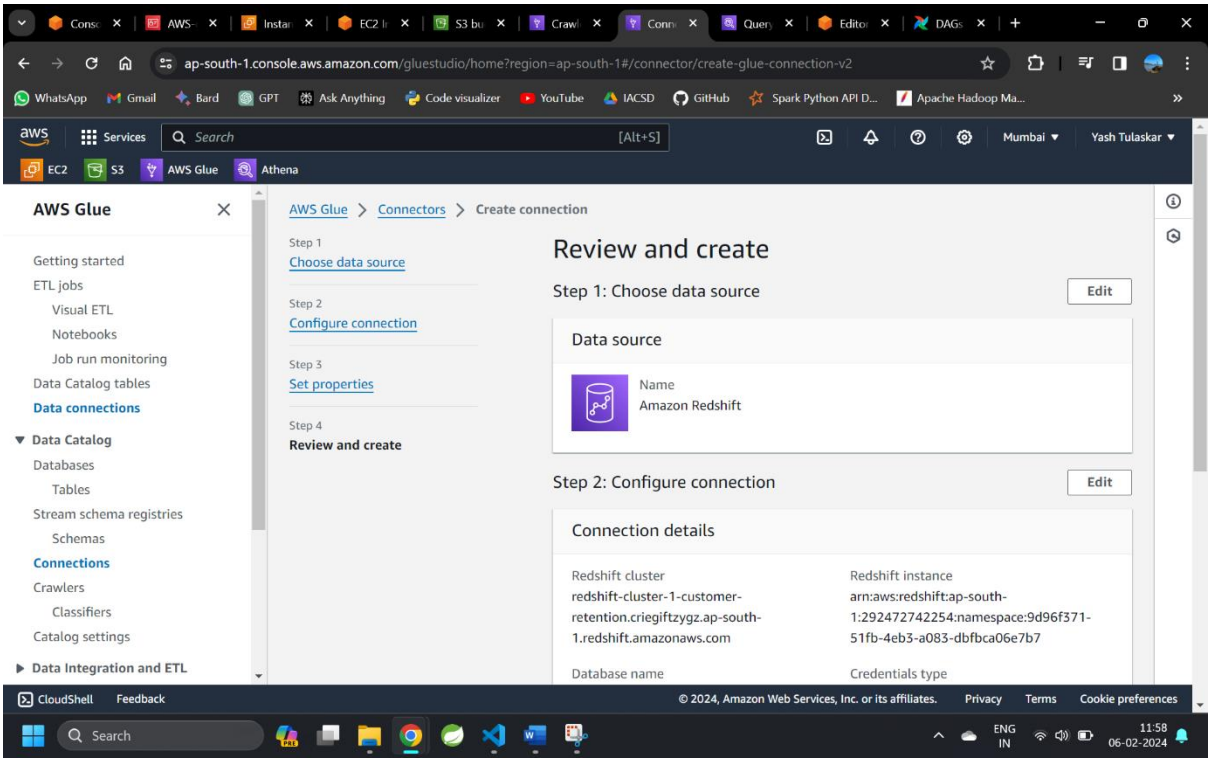

Fig.13- Created connection in AWS glue

"Connect_redshift_ETL" created the connection with this name.
Now use glue catalog to crawl it as well so we can glue data in the redshift.
Thus create glue to redshift crawler with name "glue-redshift-crawler".

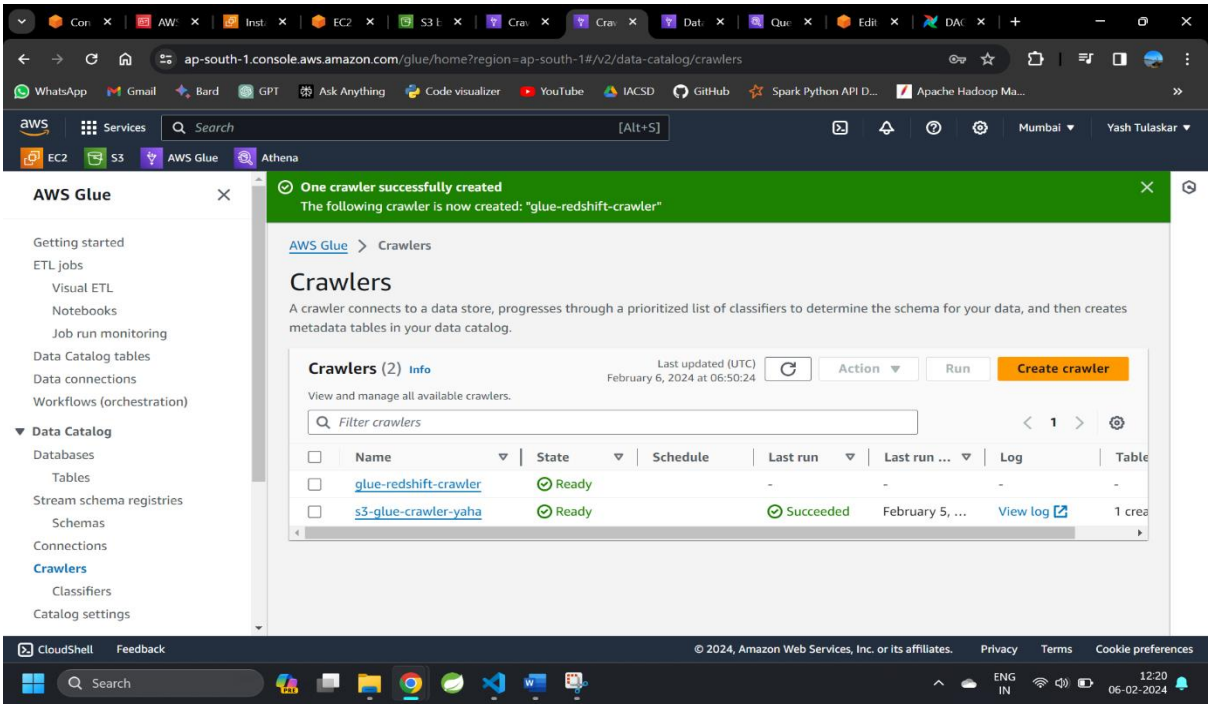

Fig.14- AWS Crawlers

After running the particular crawler state will be ready, but will fail to know why failed see logs, which will say VPC subnet failed because it could not find S3 end point.

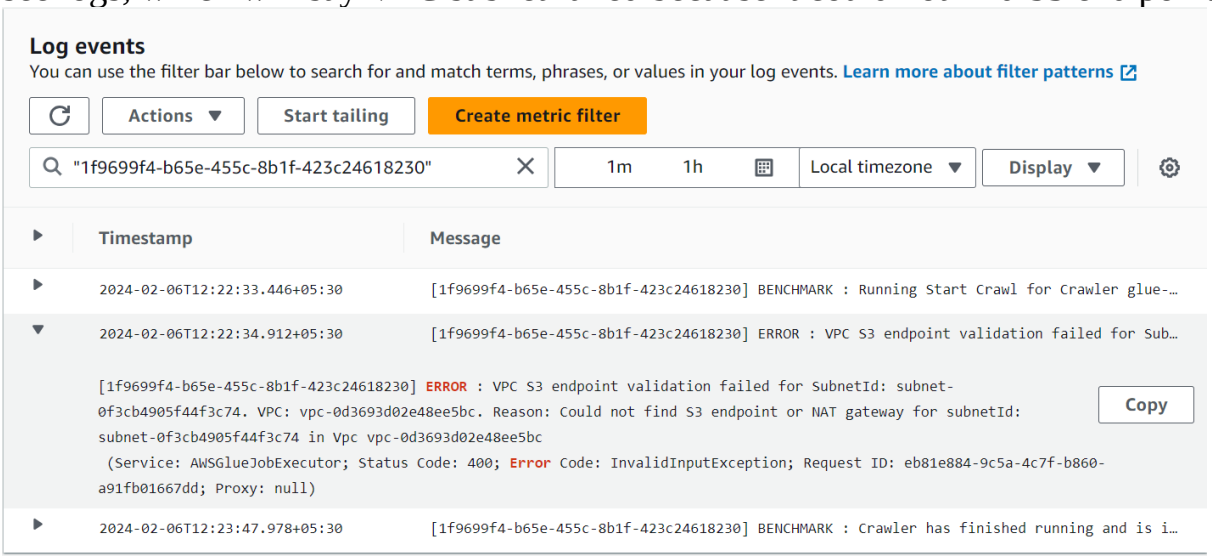

Fig.15- Second Crawler failed

To overcome that go to redshift and then go to properties and then VPC and create end point
and add the following gateway to solve that error.

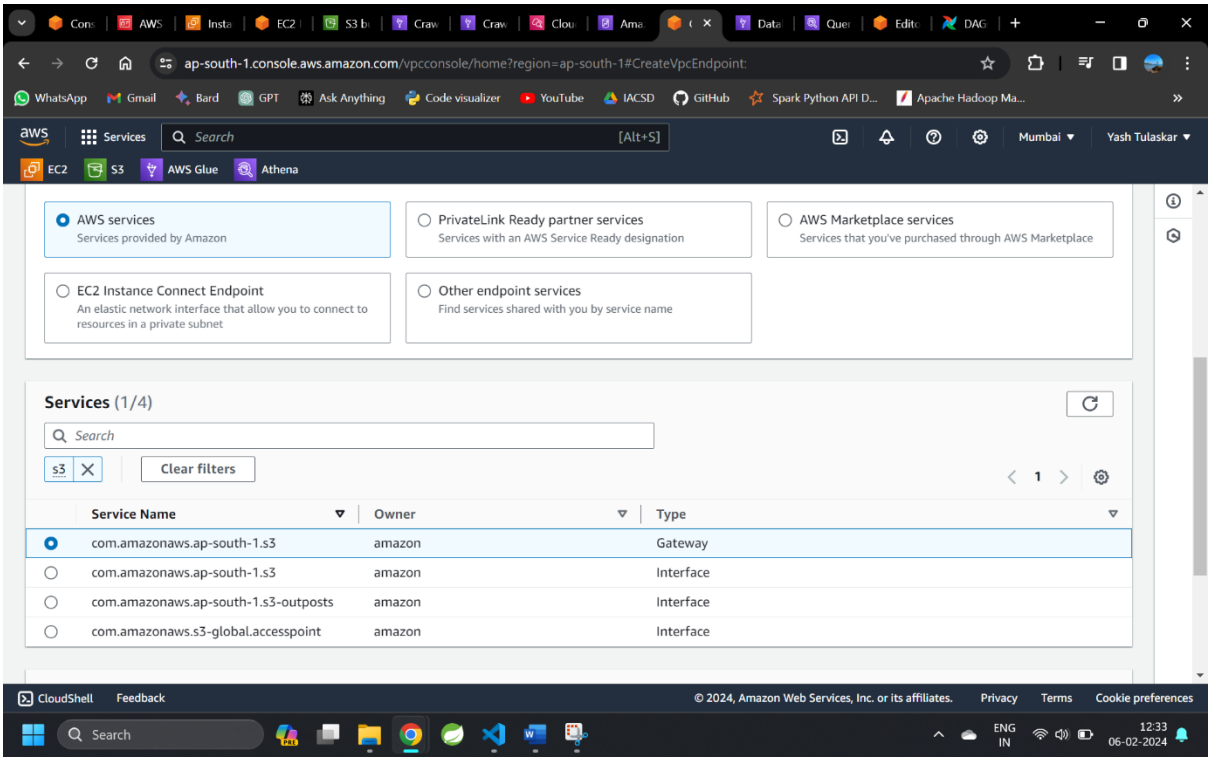

Fig.16- Added end points to VPC

Also select the route table in the create end point.

Then after running the two crawler one after other we can see that the tables which can be used as a source or target in a job definition.

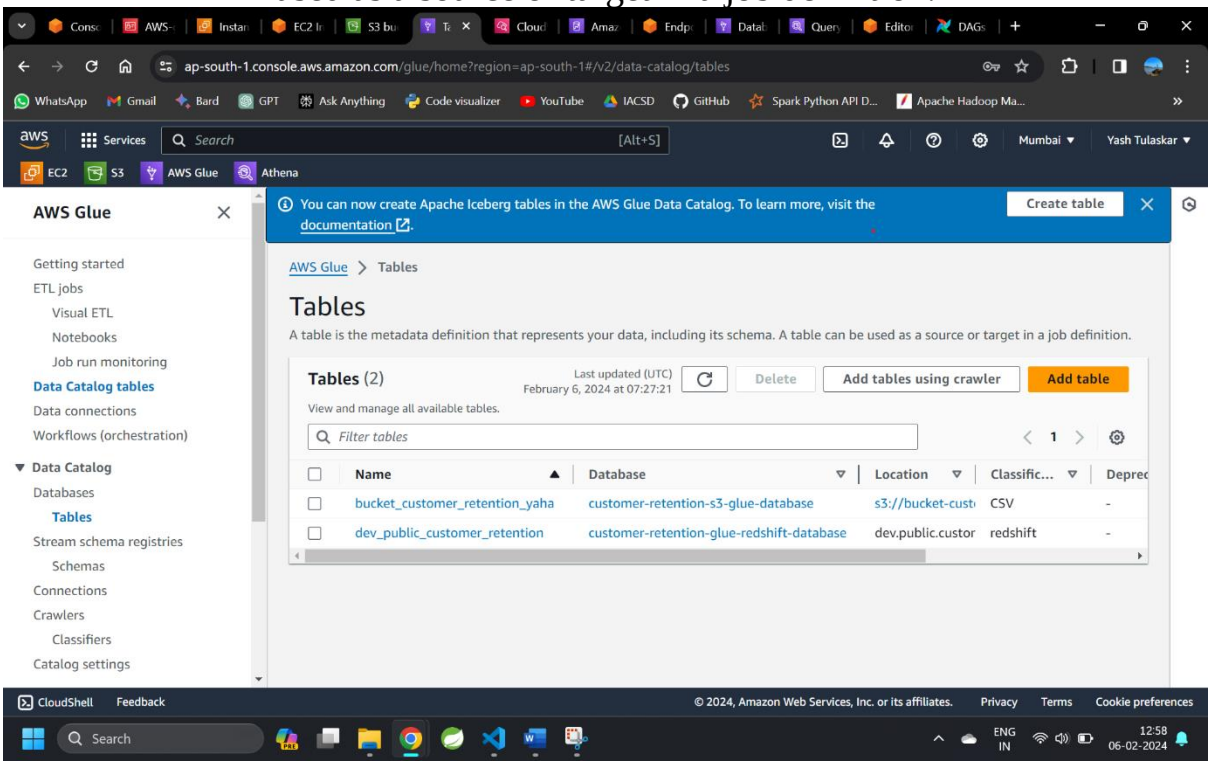

Fig.17- Table name

Now to bind those two crawlers together, we need to create a glue job, so we could automate that process, which will run the crawlers in sequence, we will be using this glue job in airflow

For that go to ETL job, then in extract add S3 bucket and necessary tables. Further,

transform where we remove unnecessary columns which are not assigned in the redshift.

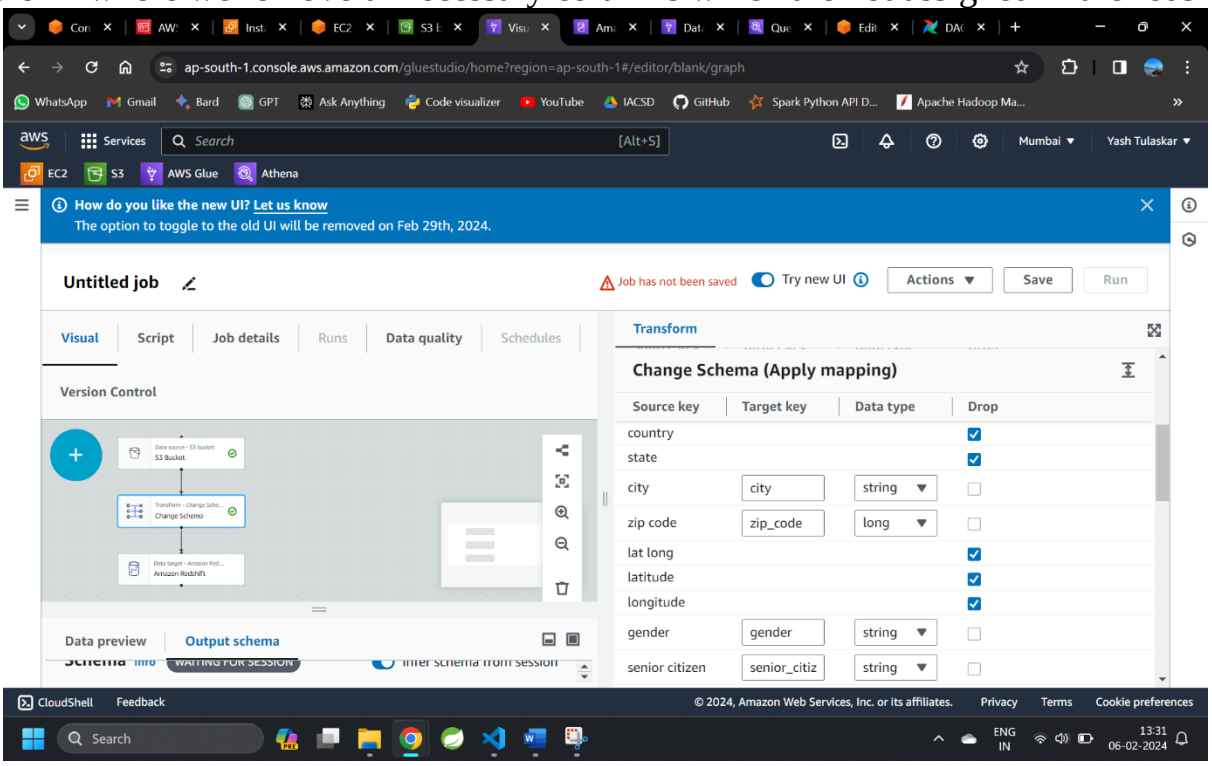

Fig.18 ETL job

Below we will configure the Load section of the ETL pipeline where we can decide how we have to handle the data and target table and also the operations performed on it.

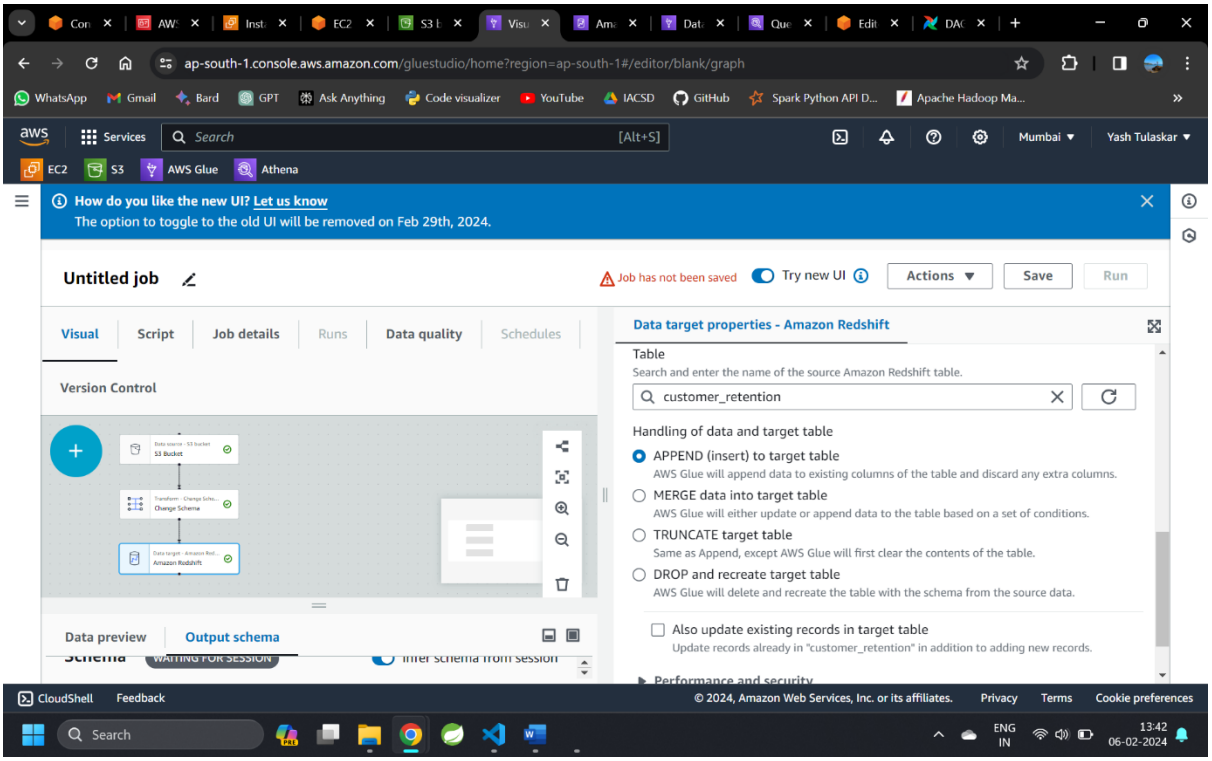

Fig.19- ETL job 2

In the Redshift before running ETL Job,
SELECT COUNT(*) FROM customer_ retention;
This query will return 0, as it has just schema at this point
After running the ETL Job, It will add data to the redshift cluster and the following query

will then give the counts as 29.

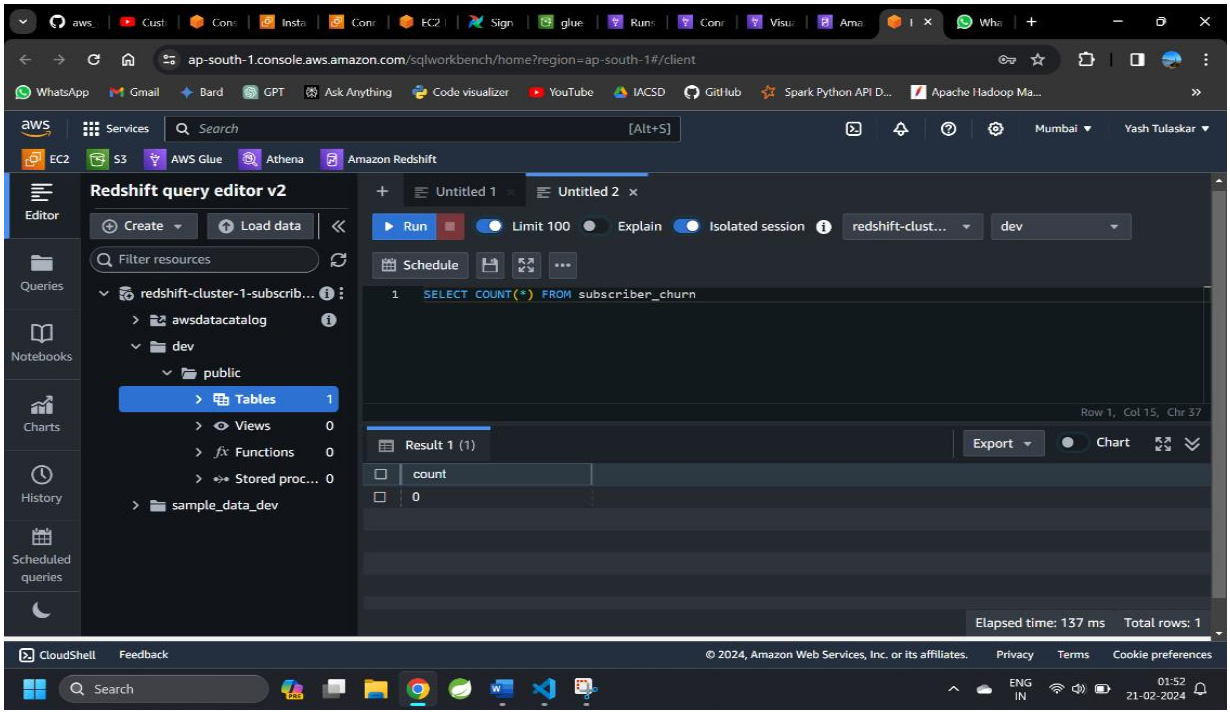

Fig.20- Below we see there is no data in the redshift cluster

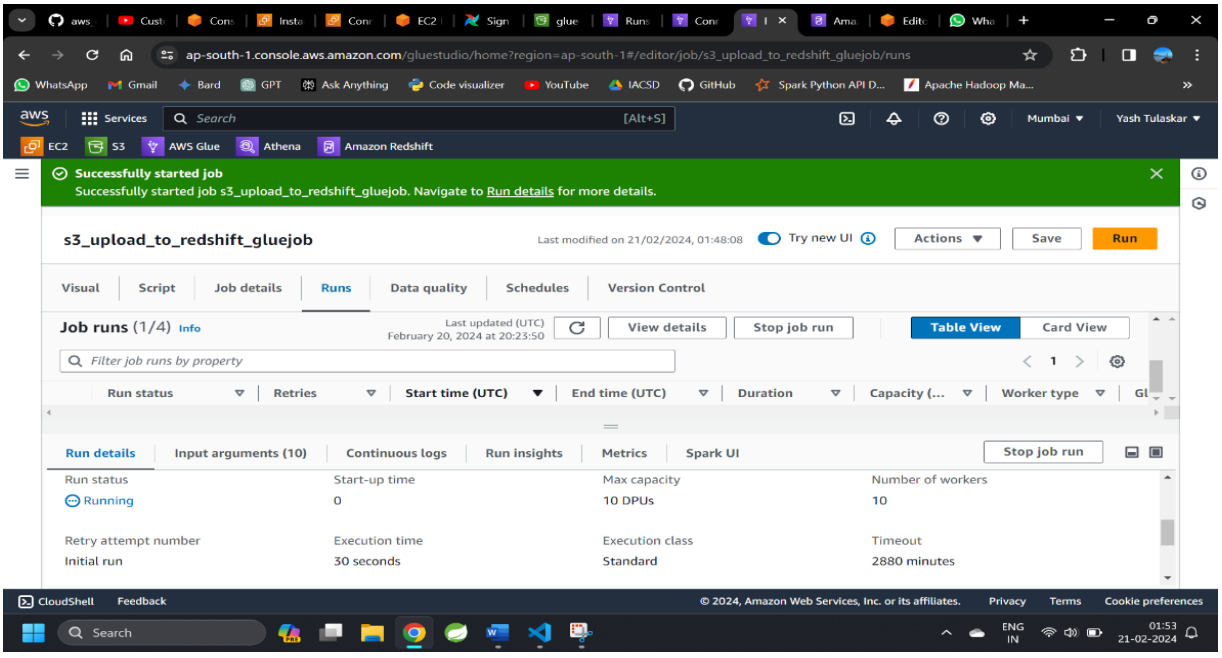

Fig.21- After running the glue job

We again go to the Redshift cluster and enter the query and count the rows, we conclude that the number of rows are added in to schema

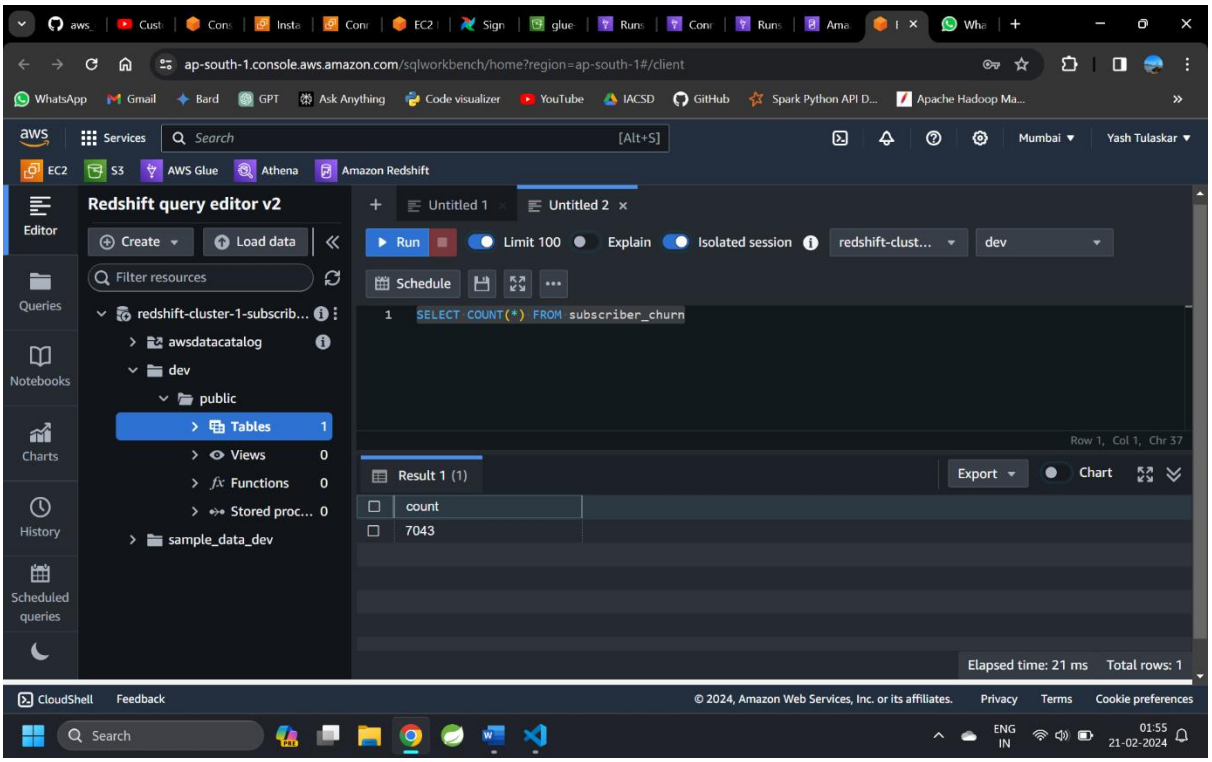

Fig.22- Data added

Airflow: Data Pipelines with Ease

Airflow, the open-source platform, empowers you to orchestrate and manage data pipelines efficiently. Let's explore its key features:

What it does:

- **Workflow Automation:** Airflow automates workflows by scheduling and executing tasks in order, ensuring smooth data processing.
- **Flexibility:** Define your workflows using Directed Acyclic Graphs (DAGs), specifying task dependencies and scheduling.
- **Task Diversity:** Execute tasks using various operators, including Python, Bash, SQL, and more, covering diverse needs.
- **Monitoring & Alerts:** Stay informed with Airflow's built-in monitoring and alerting, ensuring timely error detection.
- **Scalability:** Whether you have simple or complex pipelines, Airflow scales effortlessly to handle your workload.
- **Cloud Compatibility:** Deploy Airflow seamlessly on your preferred cloud platform, like AWS, Azure, or GCP.

Why use Airflow:

- **Ease of Use:** Python-based and built with intuitiveness in mind, Airflow is an approachable platform for data engineers and developers.

- **Community & Support:** Benefit from a vibrant community and extensive documentation, aiding you throughout your Airflow journey.
- **Extensibility:** Expand Airflow's functionalities with a rich ecosystem of plugins for various integrations and customizations.
- **Cost-Effectiveness:** Being open-source, Airflow minimizes infrastructure costs compared to proprietary solutions.

Beyond the basics:

- **Version control:** Manage DAGs with version control systems like Git, ensuring easy collaboration and rollbacks.
- **Scheduling options:** Schedule tasks manually, cron-like, or based on events, adapting to your workflow needs.
- **Parameterization:** Dynamically adjust tasks based on parameters, making workflows adaptable.

After execution of the code, we have our first DAG which will run the first job, we see the run state is running. How will the DAG know the job will be done. After we trigger the job, the ID will be associated, so for every job new ID is generated and can show new job goes to next task and whether job is successful or not.

Below all the tasks are seen, how the data flows

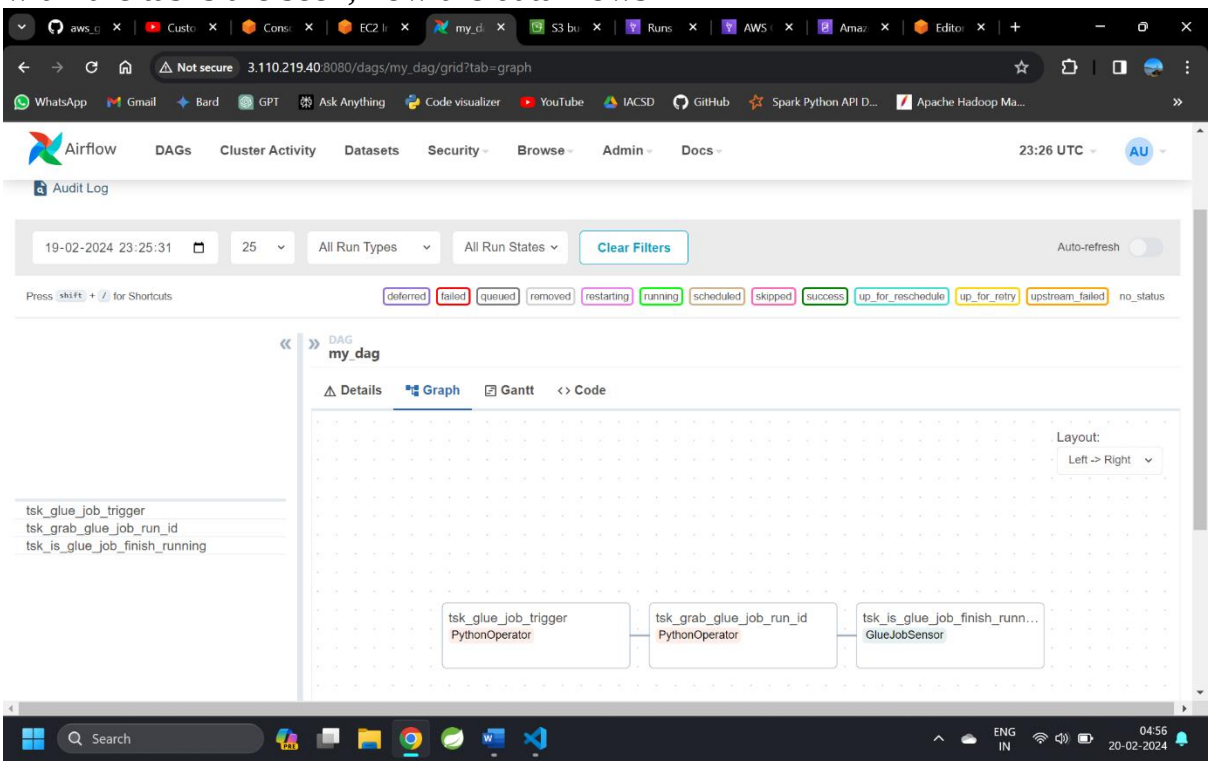

Fig.23- Airflow GUI

Establish connection using Admins > Create Connections > with connection id "aws_s3_conn"

Here, DAG2 generates the ID, the id generated is of the task completed.

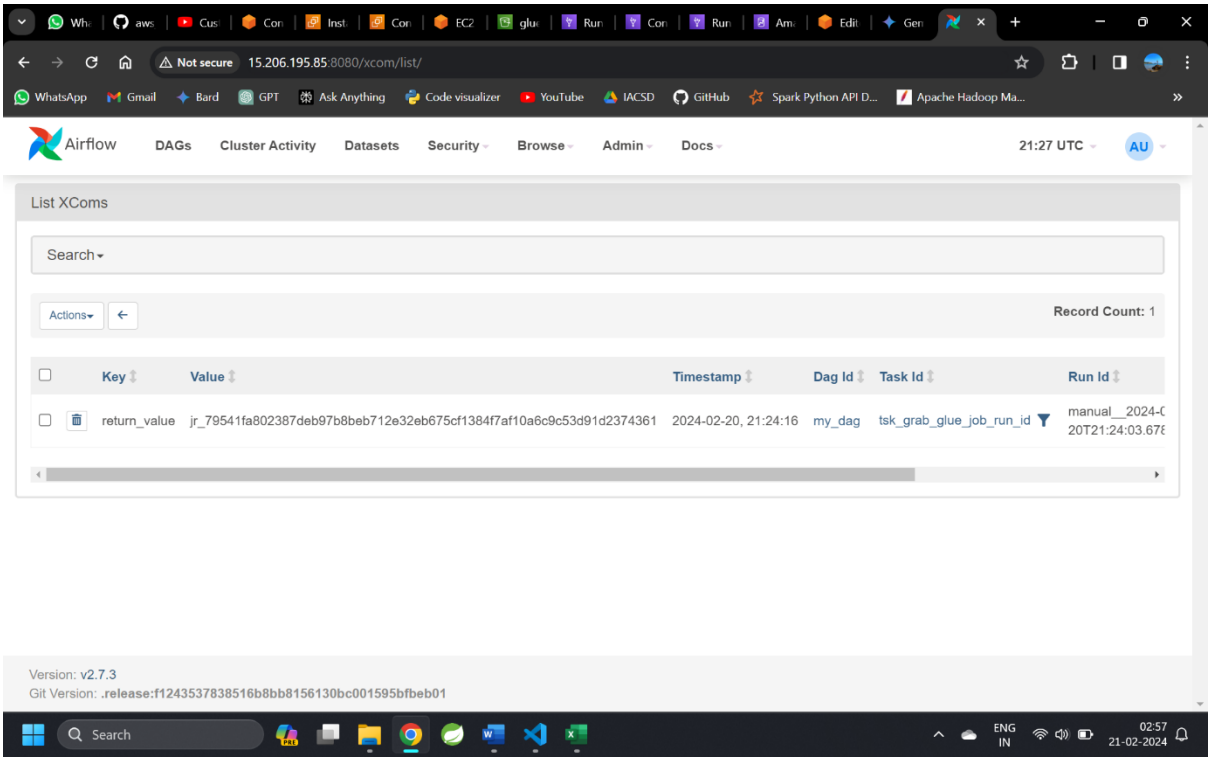

Fig.24- DAG No. 2 Generating ID

Error caused at 3rd task because we didn't give region by awscli,
Do it by downloading awscli configure aws and enter access key, id, and they region as "ap-south-1"

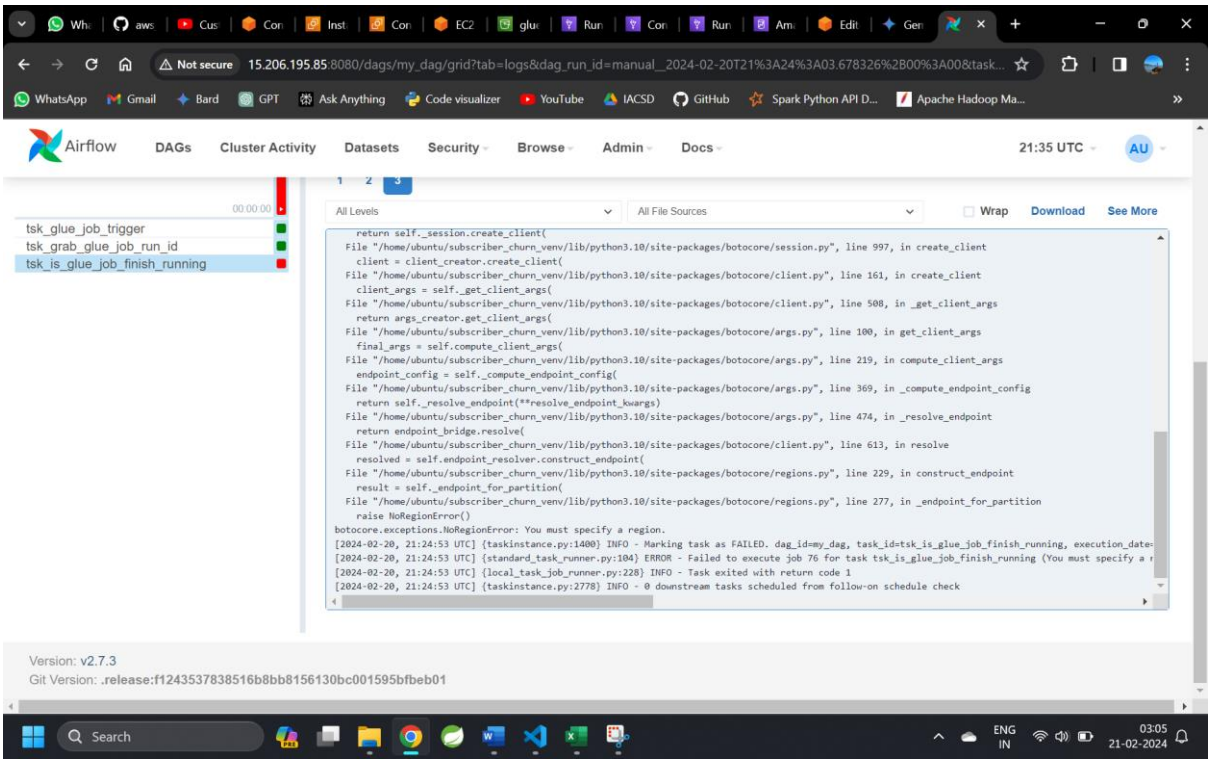

Fig.25- Task 3 not run

To overcome this download aws cli mention region

Task 3 returns success again after running and pokes success in the log every 60 seconds.

Thus, we conclude that the glue job is automated by the airflow as all the 3 dags runs successfully

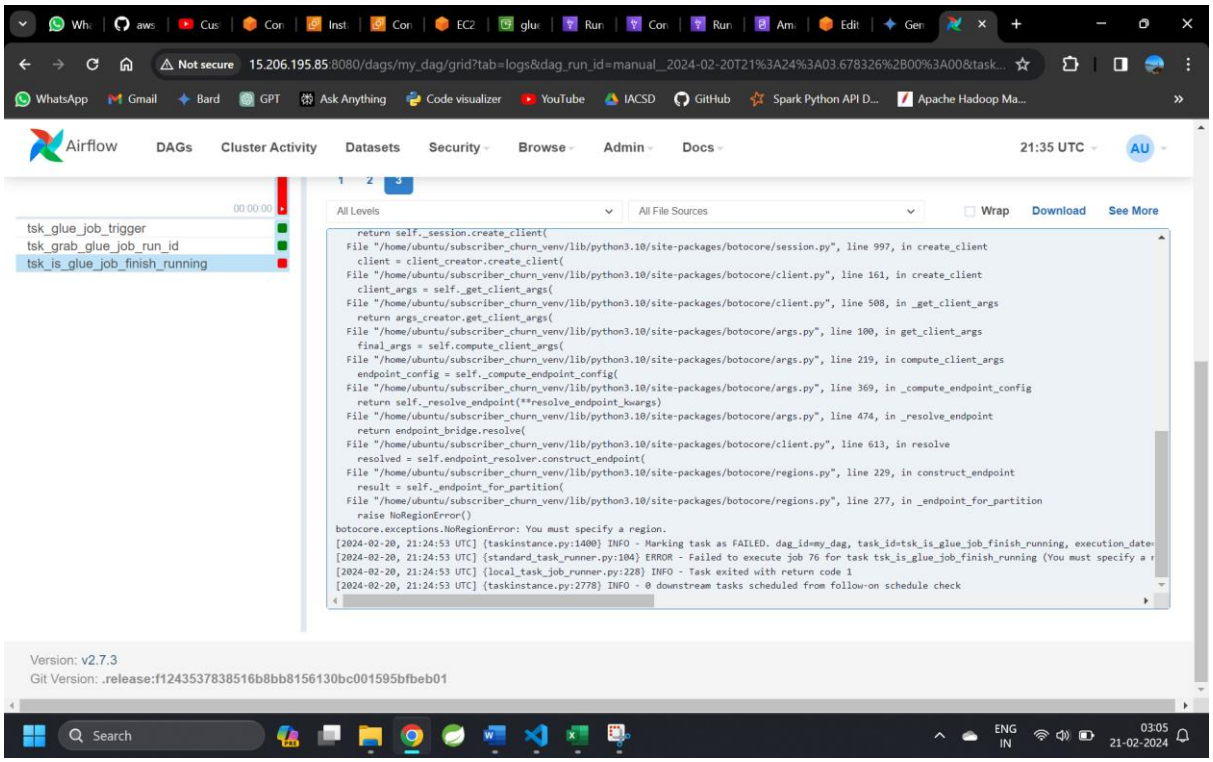

Fig.26- Sucessfully run

Future Scope

- Personalized Retention Strategies: Develop personalized retention strategies based on customer segmentation and behavior analysis to effectively mitigate churn and increase customer loyalty.
- Real-time Churn Monitoring: Implement real-time monitoring systems to track customer behavior and trigger proactive retention interventions in response to potential churn indicators.
- Integration with Customer Feedback: Integrate customer feedback and sentiment analysis data to identify underlying reasons for churn and tailor retention efforts accordingly.
- Expansion to New Markets: Extend the analysis to include data from additional regions and markets to gain insights into geographical variations in churn behavior and inform targeted retention strategies.
- Dynamic Pricing Models: Implement dynamic pricing models based on customer behavior and preferences to optimize revenue while reducing churn.
- Integration with Customer Support Systems: Integrate with customer support systems to streamline communication and address customer concerns promptly, improving overall customer satisfaction and reducing churn.
- Automated Decision Support Systems: Develop automated decision support systems to provide actionable insights and recommendations to frontline staff for proactive churn prevention efforts.
- Social Media Analysis: Incorporate social media analysis to monitor customer sentiment and identify potential churn risks or opportunities for engagement.

Conclusions

- Understanding Customer Behavior: Through thorough analysis of subscriber churn data, we have gained valuable insights into customer behavior, preferences, and patterns.

- Identification of Churn Factors: By examining churn-related metrics and factors such as service subscriptions, contract type, and payment methods, we have identified key drivers of churn within our subscriber base.

- Importance of Personalization: Our analysis highlights the significance of personalized retention strategies tailored to individual customer needs and preferences in reducing churn rates.

- Continuous Improvement: The project underscores the importance of continuous improvement in churn prediction models and retention strategies, ensuring adaptability to evolving customer trends and behaviors.

- Strategic Insights for Business Decision-Making: The findings from our analysis provide strategic insights for informed decision-making, guiding resource allocation and investment in customer retention initiatives.

- Business Value of Customer Retention: Our analysis reinforces the business value of customer retention efforts, emphasizing the potential for increased revenue and profitability through effective churn reduction strategies.

- Future Directions: Moving forward, there is a need to explore advanced analytics techniques, integrate additional data sources, and implement real-time monitoring systems to further enhance churn prediction and retention efforts.

References

- Aws Documentation- https://docs.aws.amazon.com/
- Wikipedia
- Airflow Documentation- https://airflow.apache.org/docs/
- Spark Documentation- https://spark.apache.org/documentation.html