

Create Crawler and Catalog Table:

Let us define and run the crawler to create a catalog table for flights data set.

- Crawler Name: **Flights Data Crawler**
- Database Name: **flights-db**
- Table Name: **flightscsv**

The screenshot shows the AWS Glue Console interface. On the left, the 'Data catalog' sidebar is expanded, showing 'Databases', 'Tables', 'Connections', 'Crawlers', 'Classifiers', 'Schema registries', 'Schemas', and 'Settings'. The 'Crawlers' section is selected. The main content area shows a notification: 'Crawler "Flights Data Crawler" completed and made the following changes: 1 tables created, 0 tables updated. See the tables created in database flights_db.' Below this, there are buttons for 'Add crawler', 'Run crawler', and 'Action'. A search bar is present with the text 'Filter by tags and attributes'. A table lists the crawler details:

	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
<input type="checkbox"/>	Flights Data Crawler		Ready	Logs	55 secs	55 secs	0	1

At the bottom of the console, there is a footer with 'Feedback', 'English (US)', '© 2022, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'. The Windows taskbar is visible at the very bottom, showing the time as 3:40 PM on 2/26/2022.

Created the Crawler

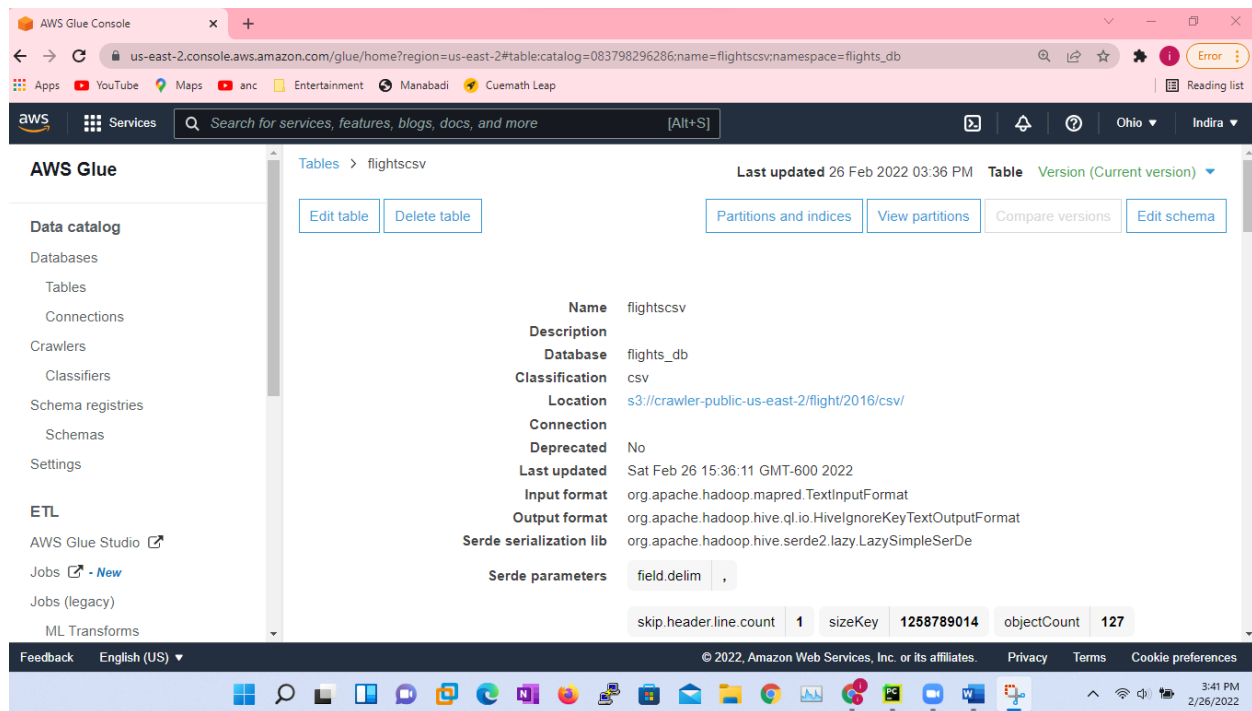


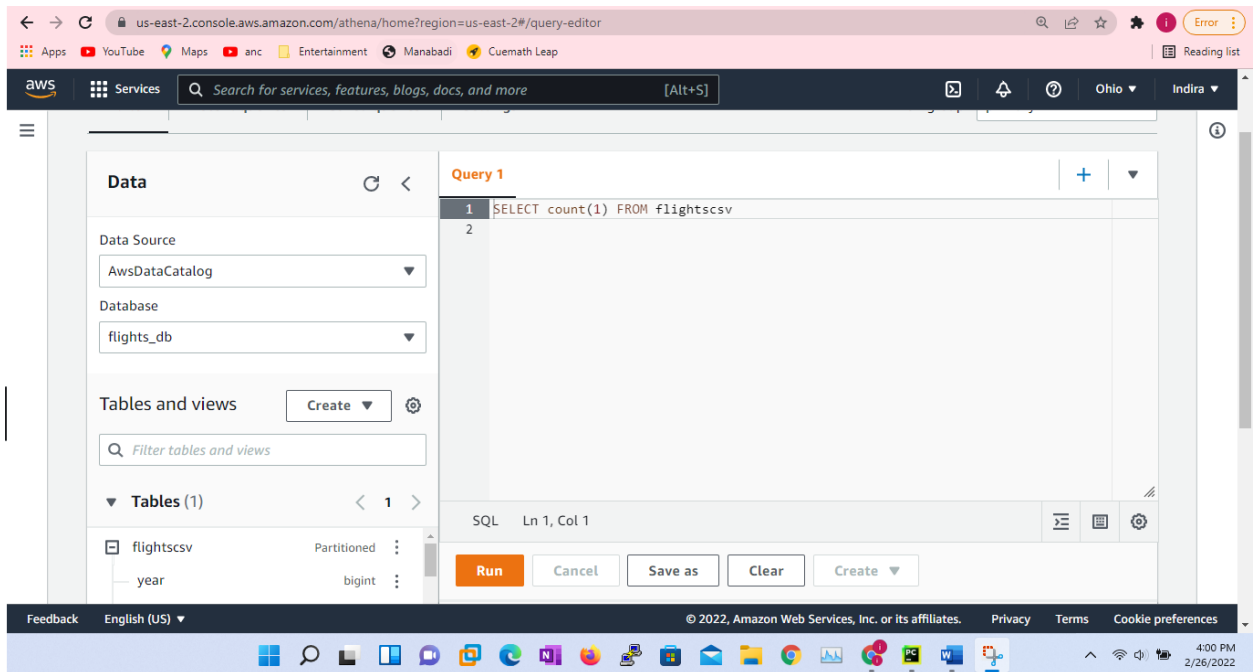
Figure 1 Created the catalog table

Analyze Data using Athena:

we can query the data using a server less query engine called Athena.

- You might have to configure s3 bucket to get started with Athena to store query results.
- Choose appropriate database or use database prefix while running queries.
- Run below queries and ensure that data is copied successfully.

```
SELECT count(1) FROM "flights-db".flightscsv;
```



Creating S3 Bucket and Role:

creating the s3 bucket and role that are required for our Glue Job to convert file format.

Bucket Name: itv-flights

Policy Name: ITVFlightsS3FullPolicy

Role Name: ITVFlightsGlueRole

Here are the steps we have followed to create s3 Bucket as well as Role for the Glue Job to convert file format will work.

Create s3 Bucket

Create Policy with following Permissions.

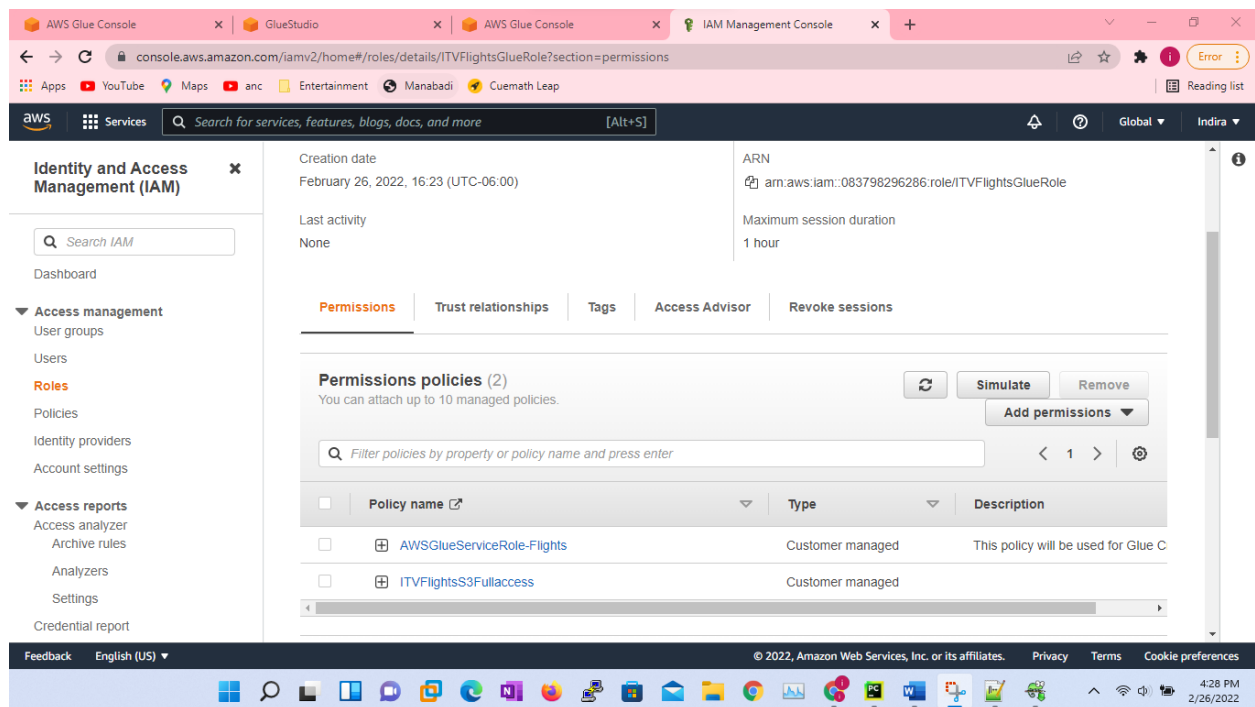
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListObjectsInBucket",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::itv-flights"
      ]
    }
  ],
}
```

```

    "Sid": "AllObjectActions",
    "Effect": "Allow",
    "Action": "s3:*Object",
    "Resource": [
        "arn:aws:s3:::itv-flights/*"
    ]
}
]
}

```

Create Role with a predefined policy AWSGlueServiceRole as well as the newly created custom policy.



Create and Run the Glue Job:

- We must have the role with appropriate permissions via policies before creating the Glue Job. In our case we have created it as **ITVFlightsGlueRole**
- We can go to Glue Console and click on Jobs. Enter the name as **flights_csv_to_parquet**.
- Source Table: **flights-db.flightscsv**
- Target Connection type: **s3**
- Target Folder with in bucket: **s3://itv-flights/flightsparquet**
- Make sure to choose all monitoring options and then create the job. It will redirect to Script Editor.
- You just need to click on **Run Job** and monitor the progress.
- Once the job is completely run, we can go to the target folder and see the created files - **ITVFlightsGlueRole**

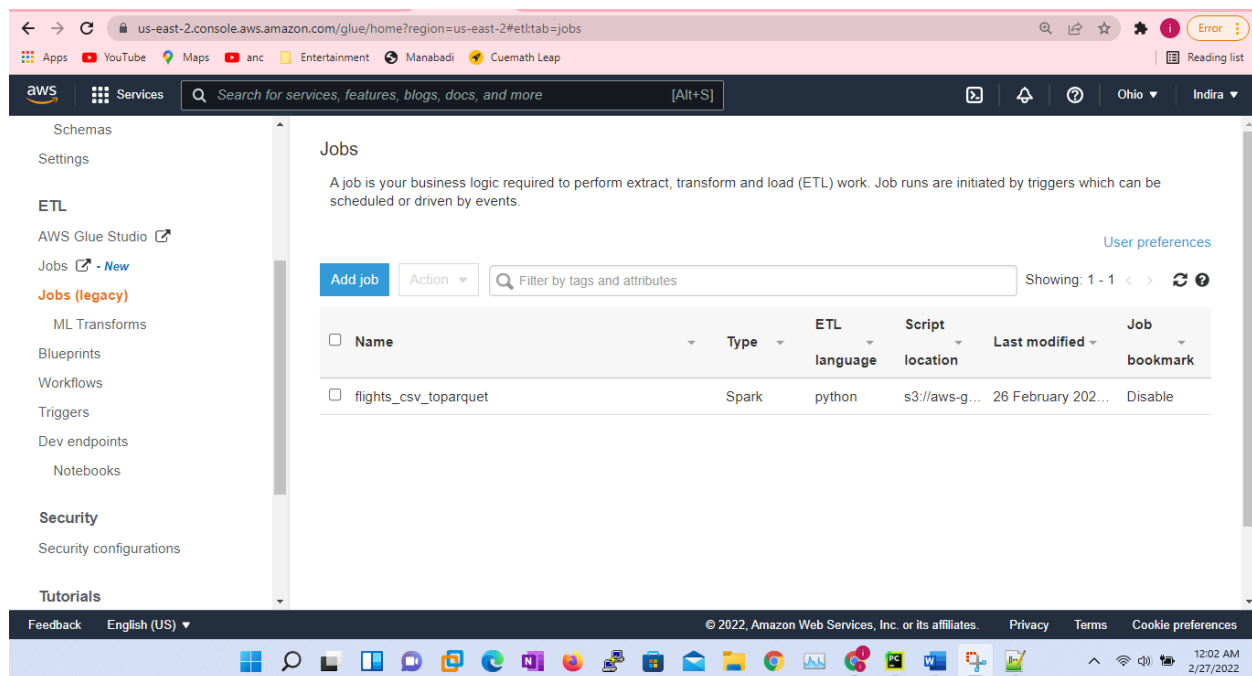


Figure 2 Glue job is created

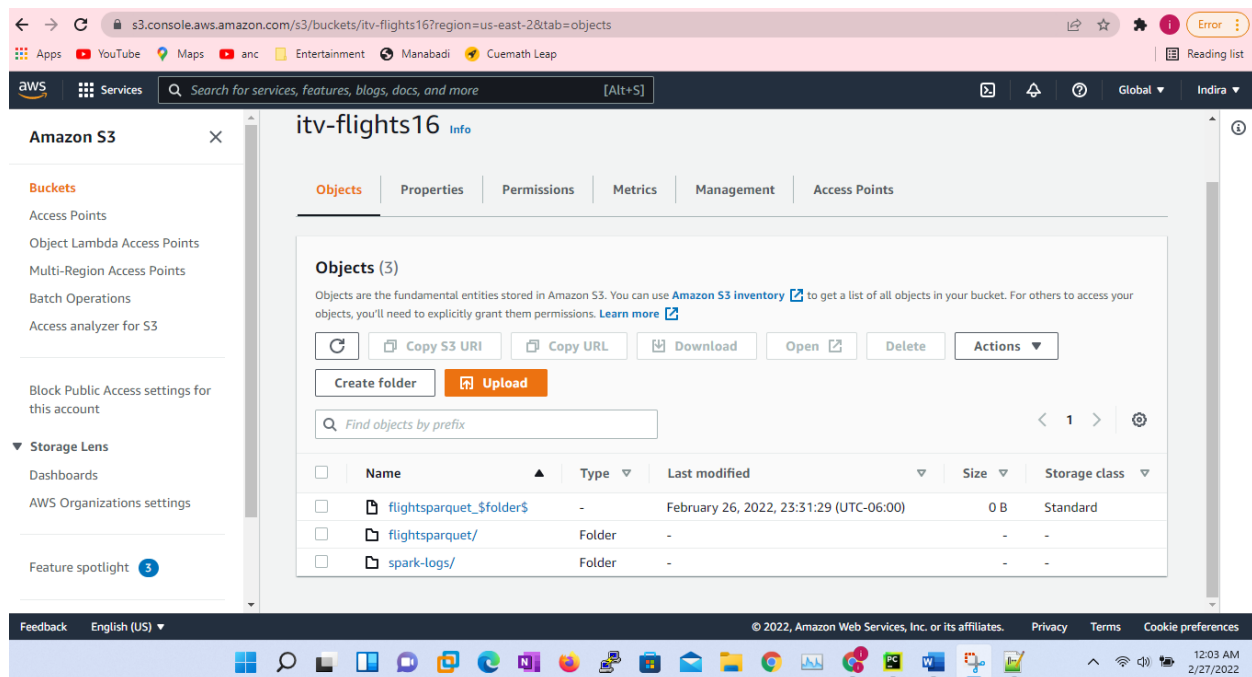


Figure 3 created the buckets in the target

As the job is successfully run, let us create Glue Catalog Table and run queries using Athena against the table.

- Create a Glue Crawler against the s3 URI - **s3://itv-flights/flightparquet/**
- Make sure to choose **Update a policy in an IAM role** and select **AWSGlueServiceRole-Flights** under IAM section. This will take care updating the role with required permissions to read from **s3://itv-flights/flightparquet/**

- Run Crawler to create the Glue Catalog Table. It will be created as **flightsparquet**.
 - Once the table is created, we can go to Athena Console and run below queries using Query Editor. Make sure to compare against the queries ran earlier using **flightscsv**.
- ```
SELECT count(1)
FROM "flights-db".flightsparquet;
```

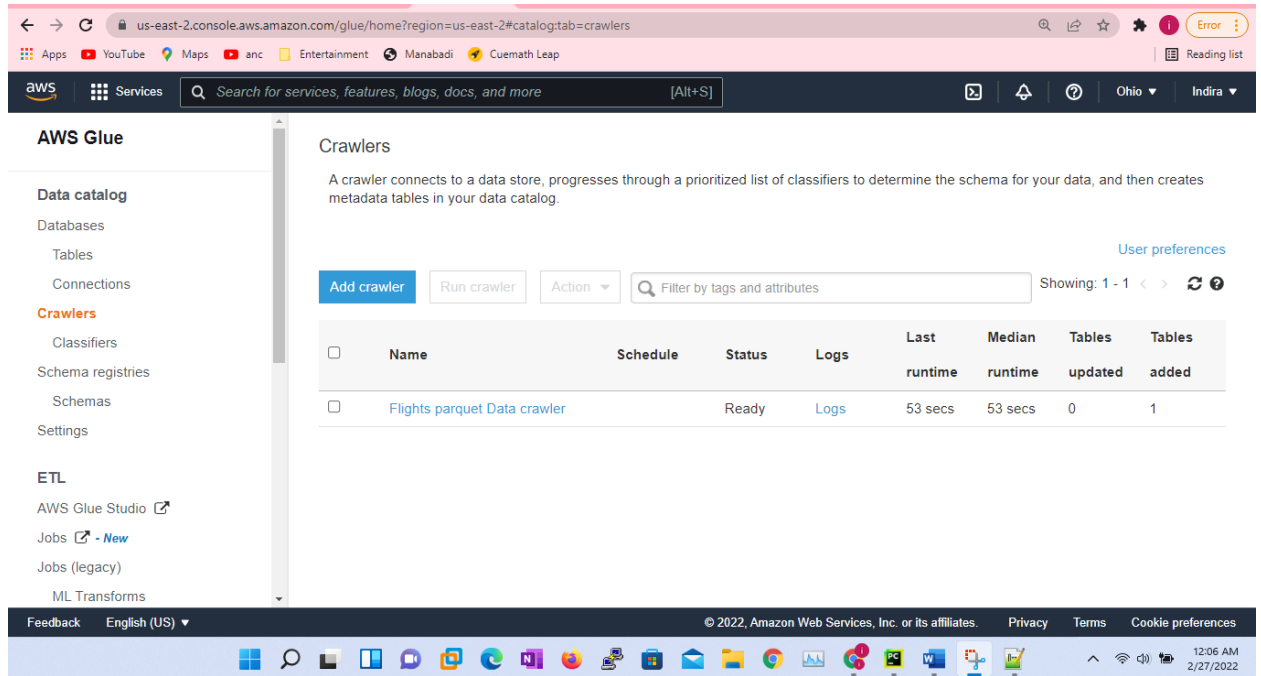
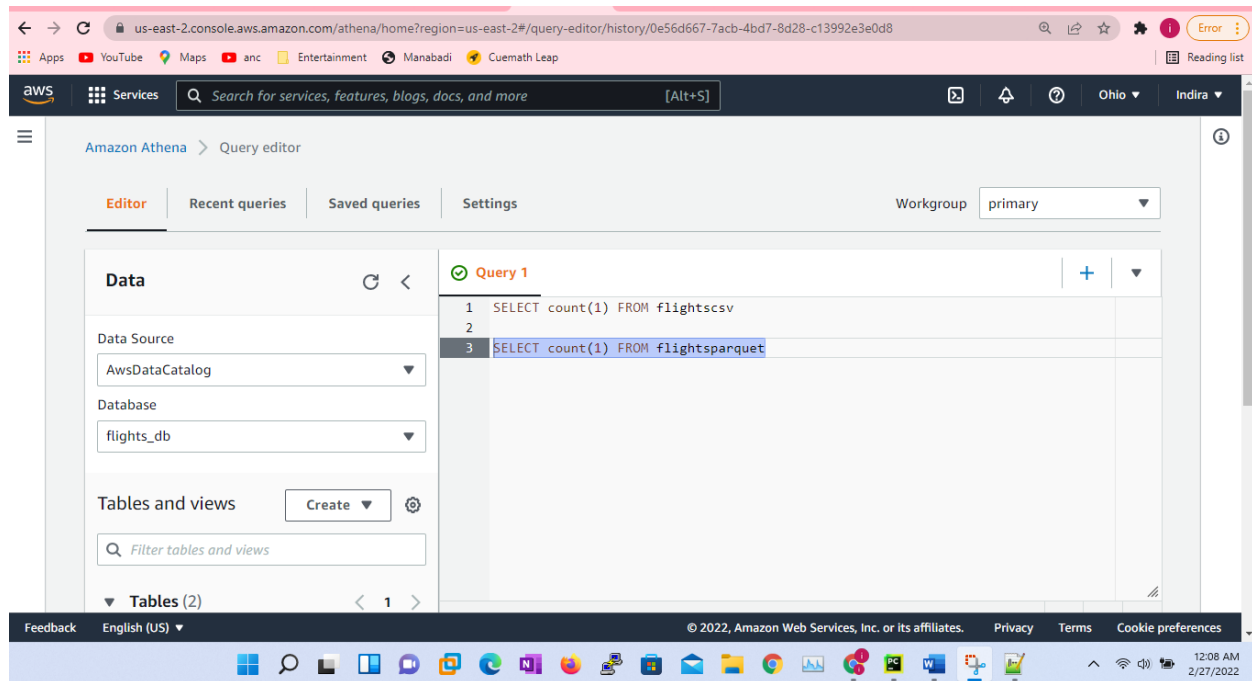


Figure 4 Created the crawler



The screenshot shows the AWS Athena console interface. On the left, a sidebar contains a search bar and a list of tables and views. The main content area displays the results of a query execution. The query is shown in the SQL editor, and the results are displayed in a table with one row containing the value 5248439. The status bar at the bottom indicates the query is completed.

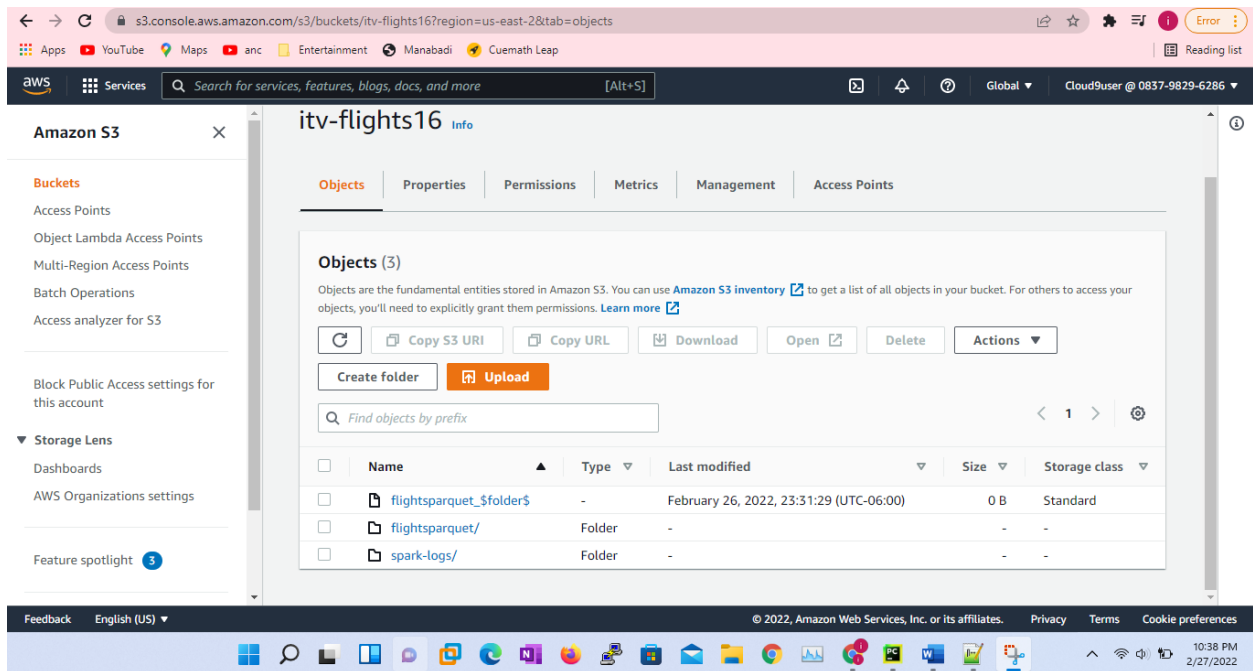
## Create and Run Glue Trigger:

Glue Triggers can be used to schedule Glue Jobs. Here are the steps we are going to follow.

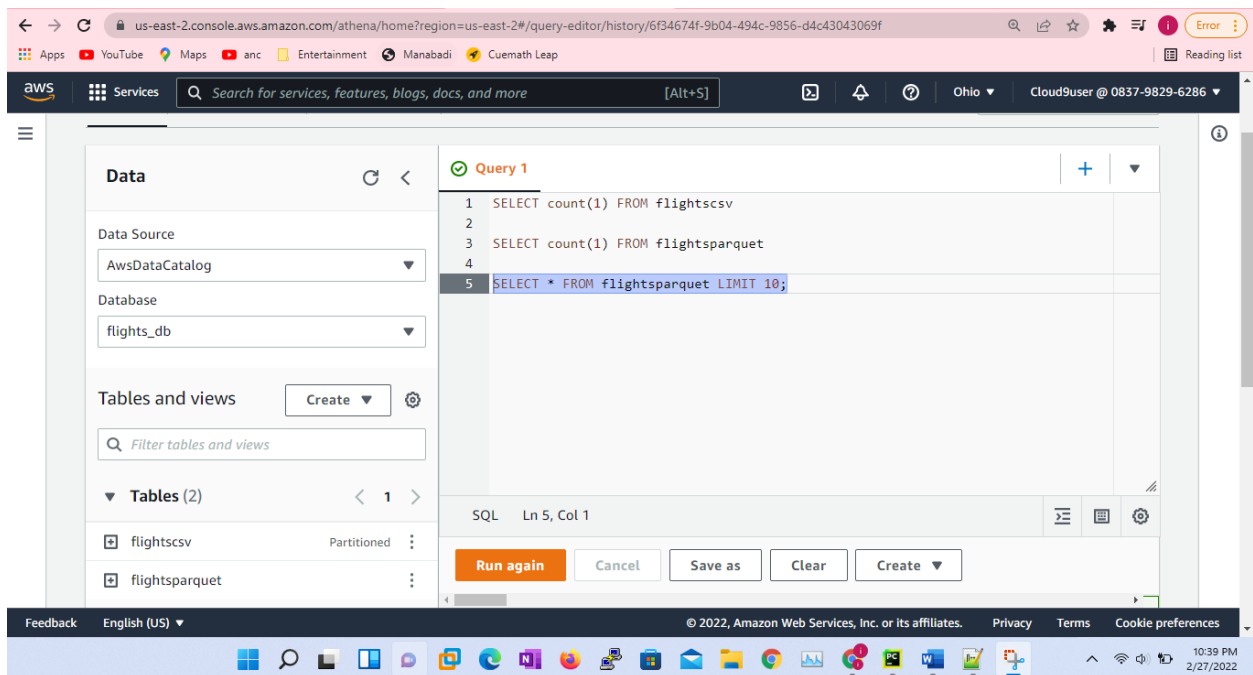
- Create and Start the trigger.

The screenshot shows the AWS Glue console interface. The left sidebar contains a list of services, with 'Triggers' highlighted. The main content area displays the 'Triggers' page, which includes a table listing the triggers. The table has columns for 'Trigger name', 'Trigger type', 'Trigger status', 'Trigger parameters', and 'Jobs to trigger'. One trigger is listed: 'flights\_csv\_to\_parquet' with type 'On-demand' and status 'CREATED'.

- Validate whether **flightparquet** is created or not under s3.



- Run queries against the Glue Catalog Table **flightparquet** using Athena.



You can monitor the status by going to the underlying Glue Job.



## Create Glue Workflow

create the Glue Workflow to create a catalog table for source data, run Glue Job and then create a catalog table for target data. Here are the steps to create glue workflow before running it.

- Create Workflow with Name - **Flights CSV to Parquet Workflow**
- Create Trigger to run crawler to create catalog table **flightscsv** - **Run Glue Crawler for Flights CSV Trigger**
- Attach Trigger with appropriate crawler - **Flights Data Crawler**
- Create Trigger to run Glue Job to convert file format from CSV to Parquet - **Run flights\_csv\_to\_parquet Job**
- Attach Trigger with appropriate Glue Job - **flights\_csv\_to\_parquet**
- Create Trigger to run crawler to create catalog table **flightsparquet** - **Run Glue Crawler for Flights Parquet**
- Attach Trigger with appropriate crawler - **Flights Parquet Data Crawler**
- Drop existing tables and delete the target folder so that we can validate the complete Workflow after running it.

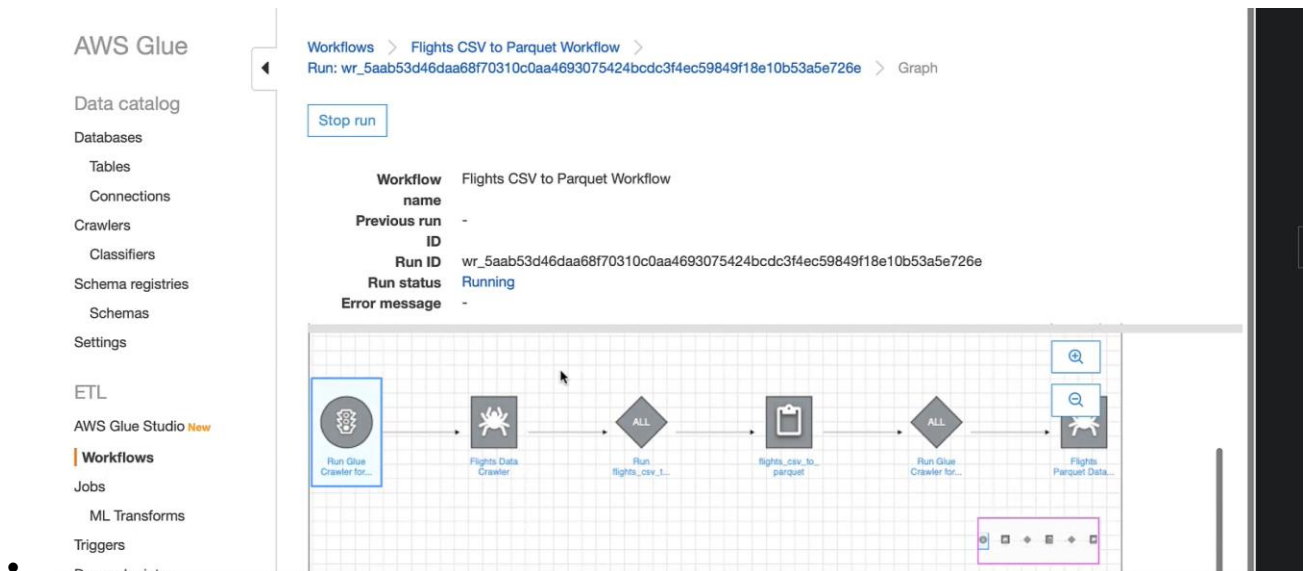


Figure 5 Workflow is created

AWS Glue

Workflows > Flights CSV to Parquet Workflow > Run: wr\_5aab53d46daa68f70310c0aa4693075424bcd3f4ec59849f18e10b53a5e726e > Graph

Stop run

| Workflow name   | Flights CSV to Parquet Workflow                                    |
|-----------------|--------------------------------------------------------------------|
| Previous run ID | -                                                                  |
| Run ID          | wr_5aab53d46daa68f70310c0aa4693075424bcd3f4ec59849f18e10b53a5e726e |
| Run status      | Completed                                                          |
| Error message   | -                                                                  |

Graph

Select the graph nodes to resume and then choose Resume run.

Resume run

Legend: ✔ Succeeded ▶ Running ✖ Stopped ✖ Failed ✖ Timeout ✖ Error ⚠ Warning ◻ Not started

Crawler details

Crawl

Sun, 24 Jan 2021 01:00:47 GMT - SUC...

| Name          | Flights Parquet Data Crawler  |
|---------------|-------------------------------|
| Description   | -                             |
| Status        | CANCELLING                    |
| Error message | -                             |
| Start time    | Sun, 24 Jan 2021 01:00:47 GMT |
| End time      | -                             |

Figure 6 Glue workflow is success

Once the Glue Workflow is completely run, make sure to validate.

Check for the tables in Glue Catalog in flights-db database.

Also, run queries against the tables using Athena.

```
SELECT count(1)
FROM "flights-db".flightscsv;
```

```
SELECT count(1)
FROM "flights-db".flightsparquet;
```

```
SELECT *
FROM "flights-db".flightsparquet
LIMIT 10;
```