😊 Perform ETL in Glue

ETL jobs in AWS Glue refer to **Extract, Transform, and Load (ETL)** processes that help automate the movement and transformation of data between different data sources and destinations. AWS Glue is a fully managed serverless service that simplifies the creation and execution of ETL jobs.

Here's a breakdown of the ETL process in AWS Glue:

1. **Extract**: Data is extracted from various sources, which could include databases, data lakes (like Amazon S3), or streaming data sources. AWS Glue supports multiple data formats like JSON, CSV, Parquet, and more.
2. **Transform**: The extracted data is processed and transformed to fit the target system's requirements. Common transformations include cleaning the data (e.g., handling missing values), reformatting data, enriching it, or applying business logic.
3. **Load**: The transformed data is then loaded into a target destination, such as a data warehouse (e.g., Amazon Redshift), a different S3 bucket, or a database (e.g., Amazon RDS).

Features of AWS Glue ETL Jobs:

- **Automatic Schema Discovery**: Glue can automatically detect the schema of your data.
- **Serverless**: No need to manage infrastructure; AWS handles it.
- **Job Scheduler**: You can schedule ETL jobs to run at specific intervals.
- **Glue Data Catalog**: Stores metadata for your datasets, making them easy to manage and query.

AWS Glue ETL jobs are ideal for automating data pipelines in analytics, data warehousing, and data integration tasks.

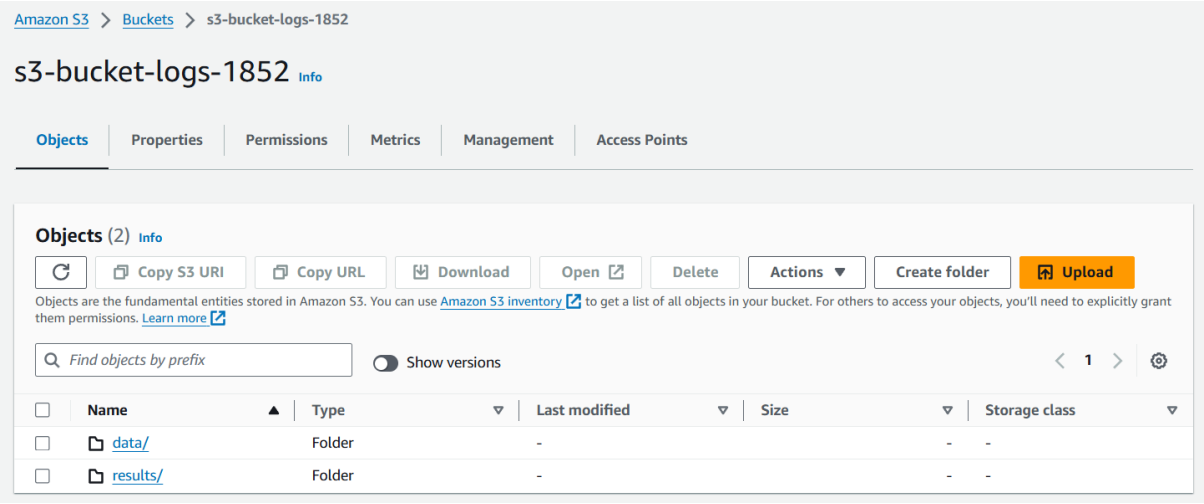
In this lab, we perform an ETL job using AWS Glue to process data from a CSV file stored in an S3 bucket. The steps involve creating a database and table using a crawler in AWS Glue to extract data from the CSV file. The crawler scans the file, creates a table, and stores the metadata in the database.

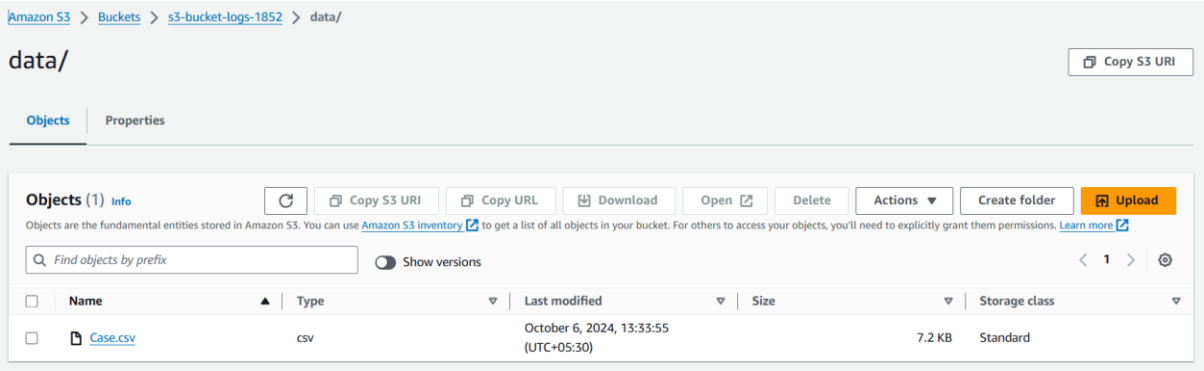
Next, we use the AWS Glue Script Editor to write and run an ETL job. The script reads data from the created table, performs transformations (such as grouping by province and city, summing confirmed cases, and identifying the maximum confirmed cases), and then writes the processed data back to another S3 bucket in CSV format.

The end goal is to automate the data extraction, transformation, and loading process, resulting in a cleaned, aggregated dataset that can be easily accessed for further analysis or reporting.

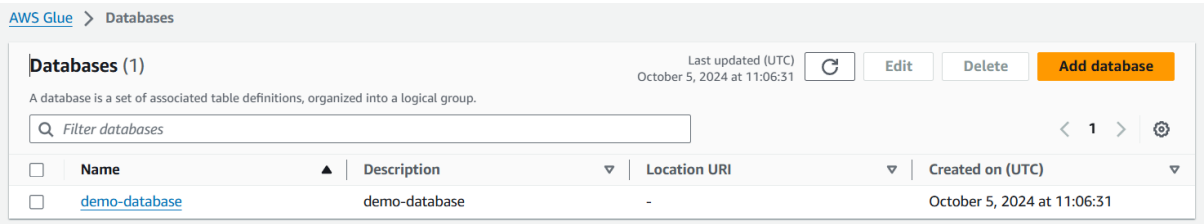
😄 To begin with the Lab:

1. In this lab we are going to see how we can perform ETL jobs in AWS Glue.
2. Here we also have a CSV file named Case which we are going to upload in S3 Bucket.
3. In your S3 bucket you need to create 2 folders with the same name and in the data folder, you need to upload the CSV file.

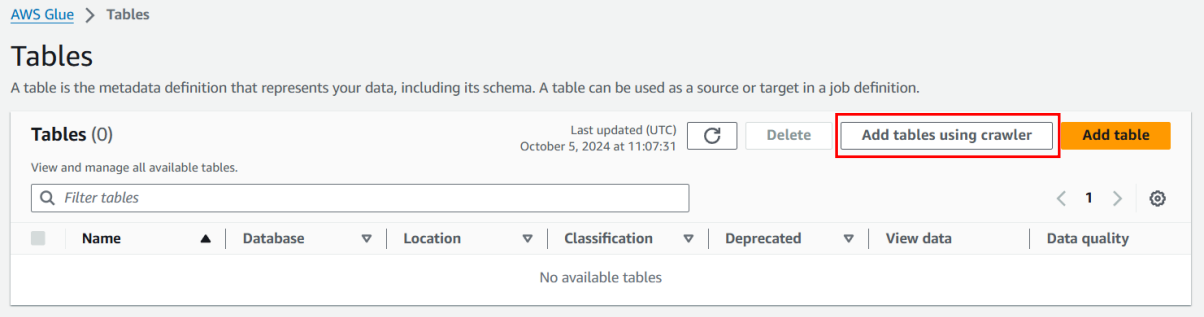




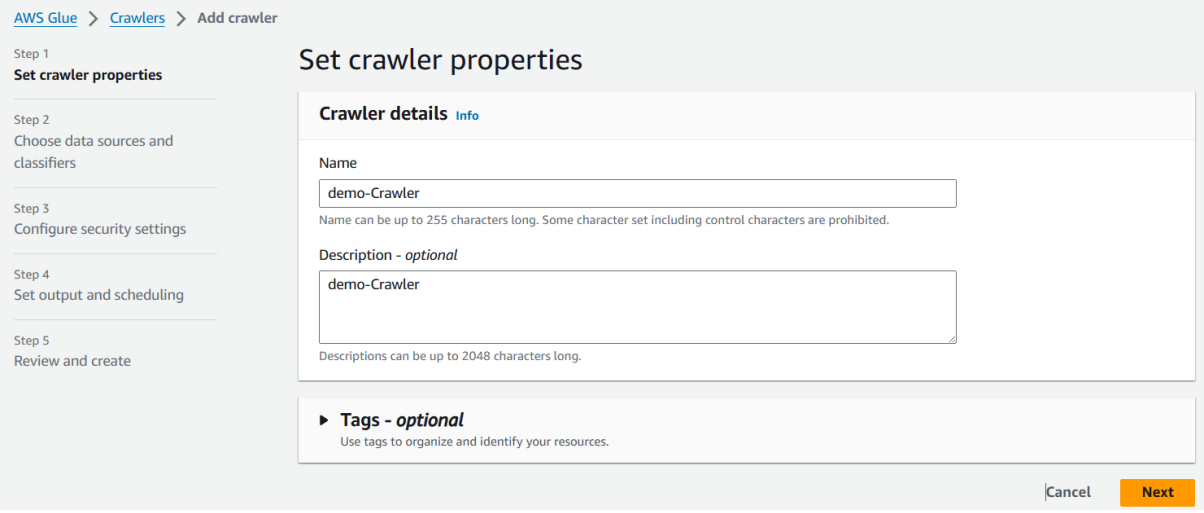
4. Now come to AWS Glue, here you need to go to Databases and create one. For that, you just need to give a name to your Database and create it.



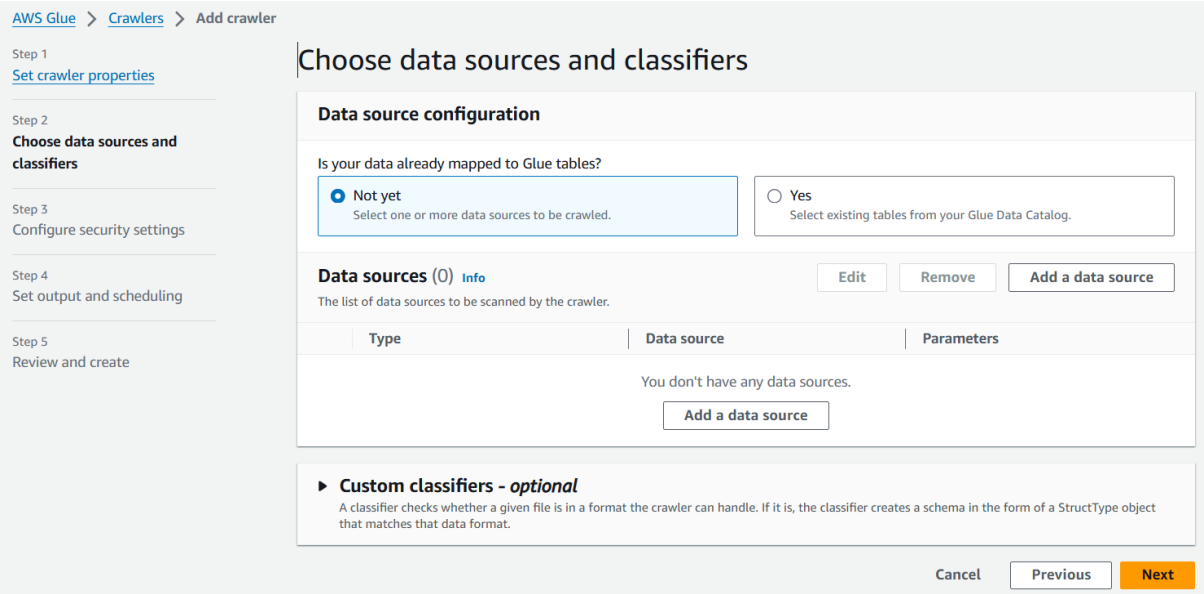
5. Then go to Tables and choose Add tables using crawlers.

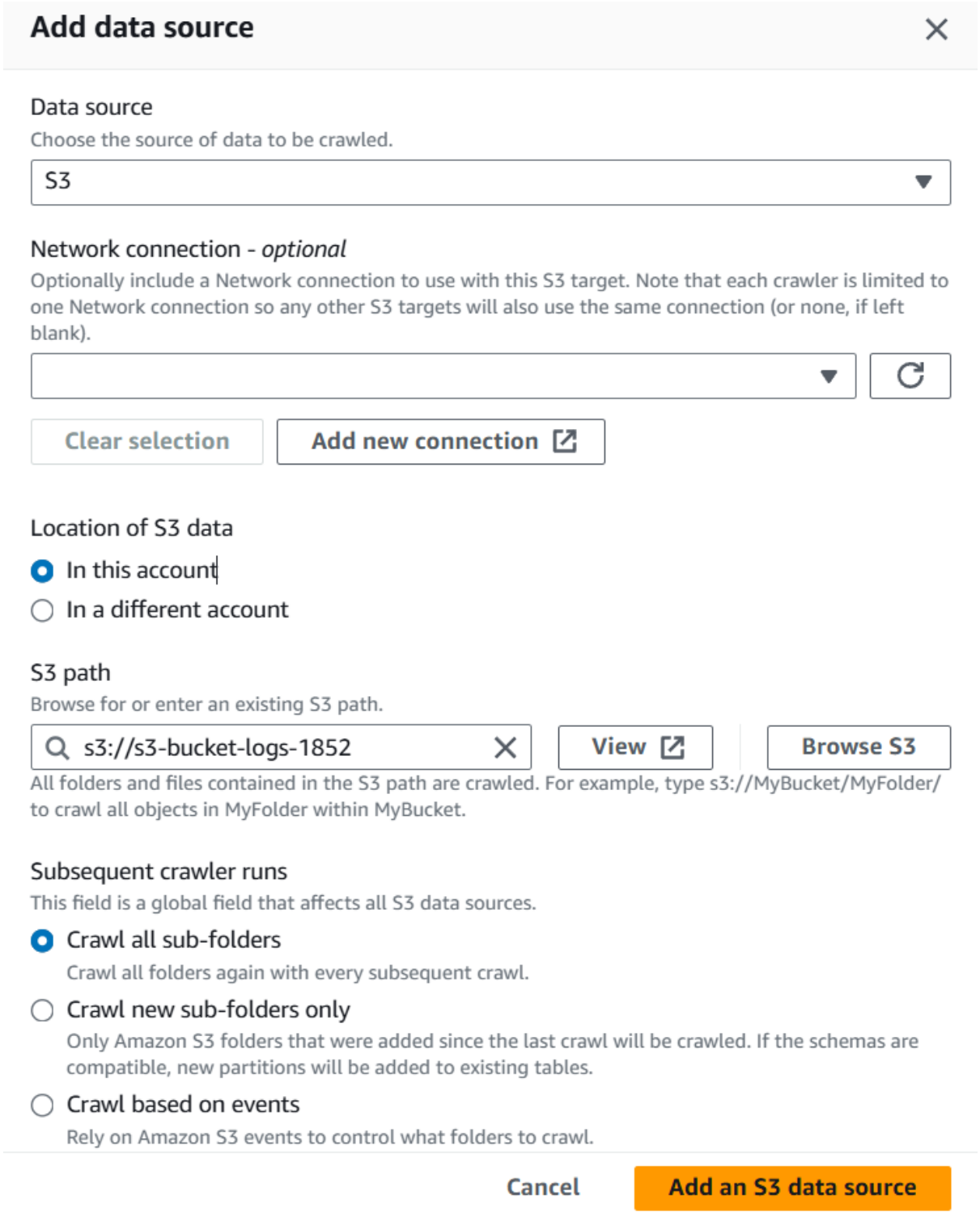


6. On step one, give a name and description to your crawler.

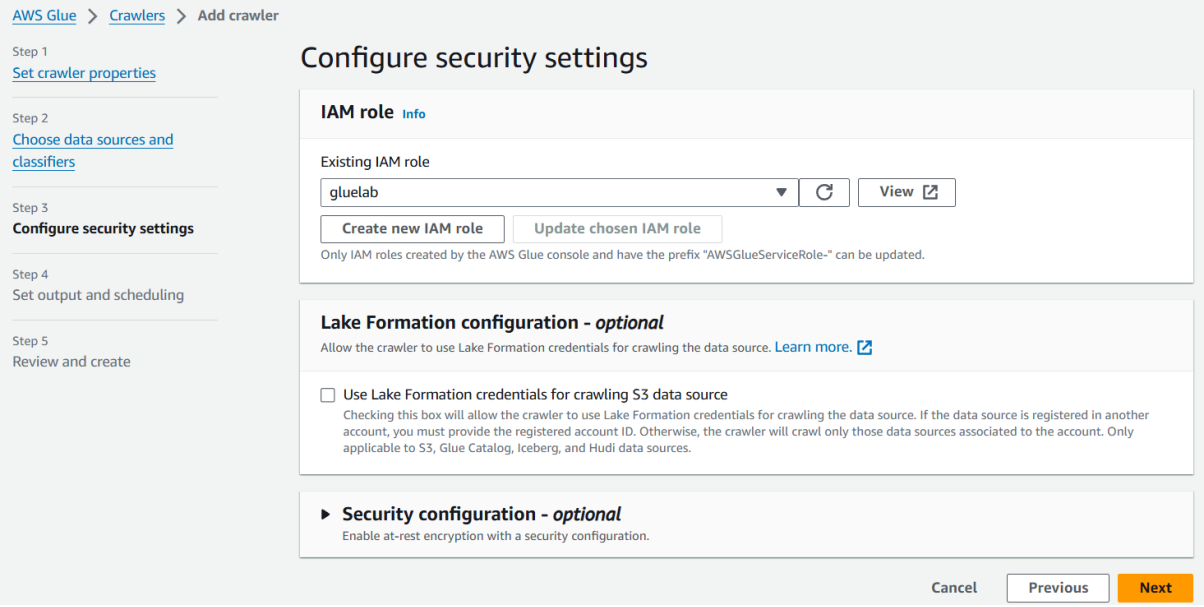


7. Then in step 2 you need to add a data source which is your S3 bucket. Choose the same options as shown in the snapshot.

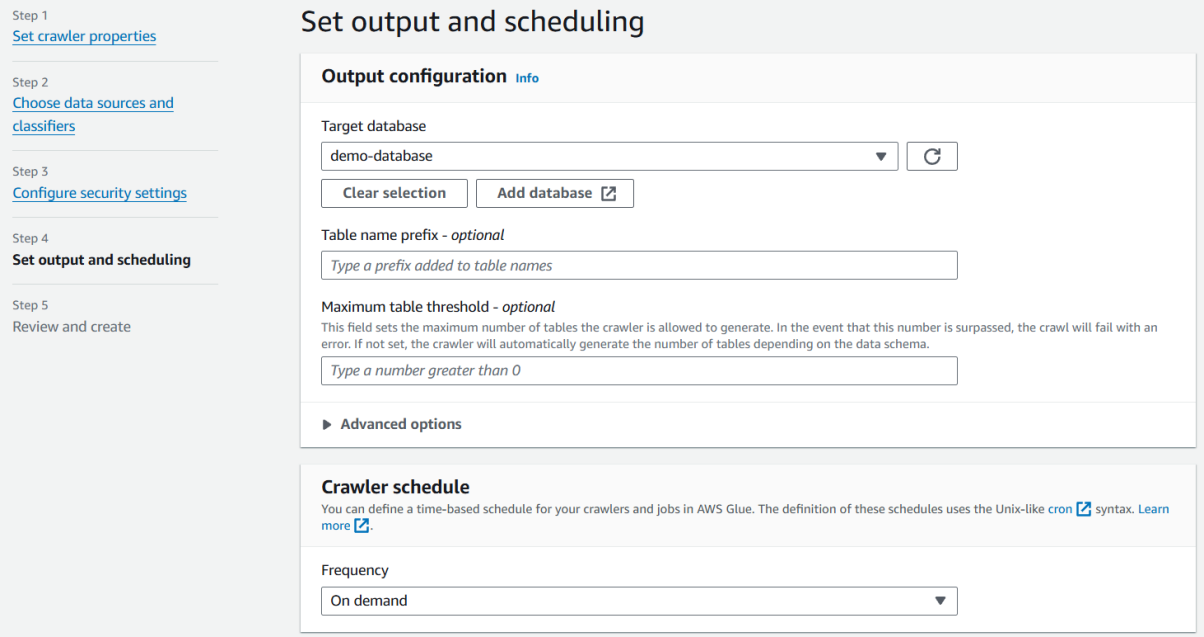




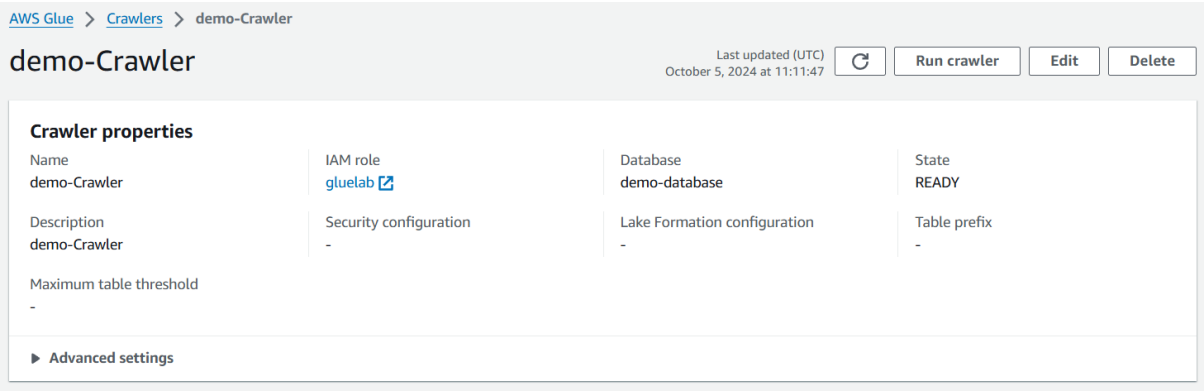
8. Now on step 3 you need to choose an IAM role if you have, or you can click on Create new IAM role and create a new one. For the IAM role permission for simplicity you can add Administrator Access.



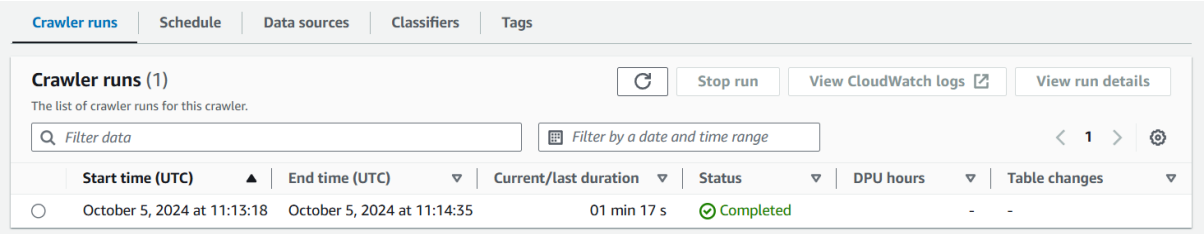
9. In step 4, choose your target database and for the crawler schedule choose on demand. Then just move ahead and create your crawler.



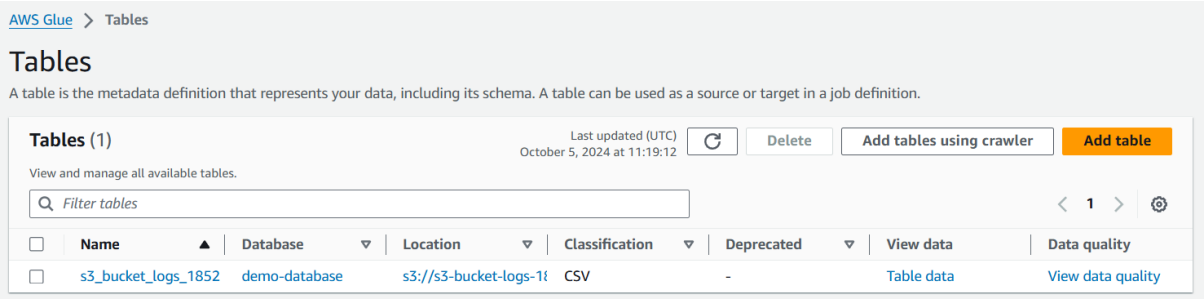
10. Below you can see that your crawler is ready, so click on run crawler and wait for it to get completed.



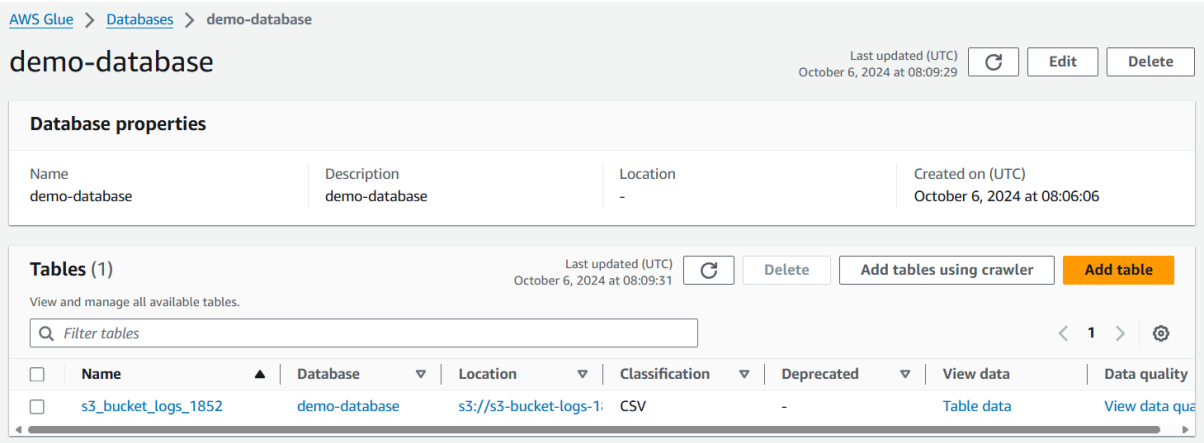
11. You can see that our crawler run has been completed. Now if you to tables.



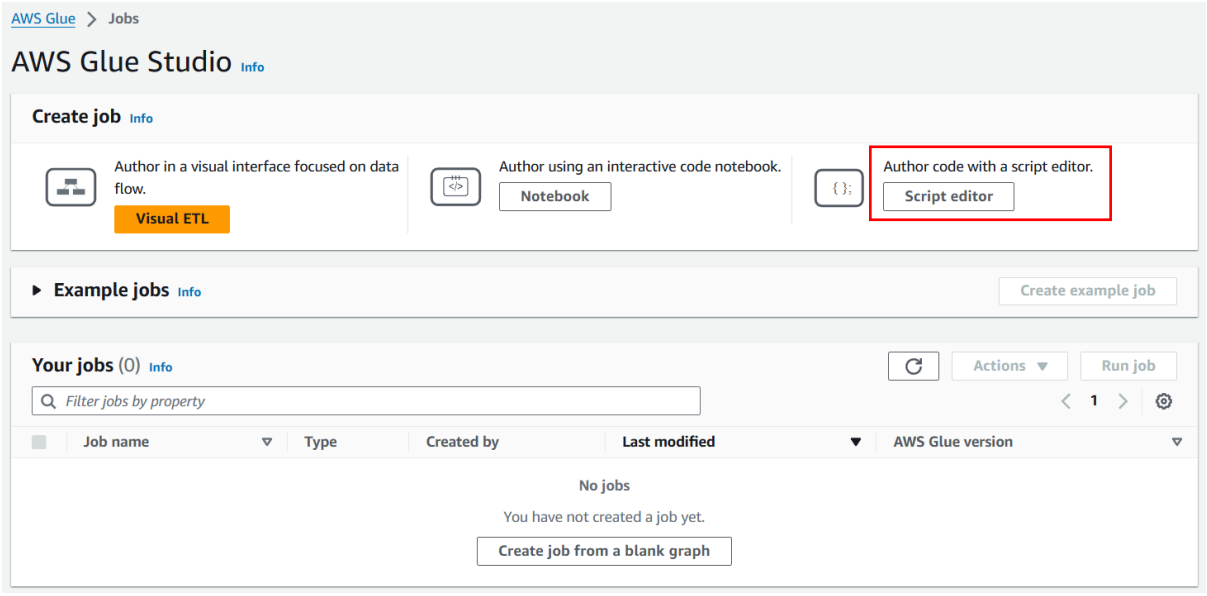
12. You will see a table has been created.



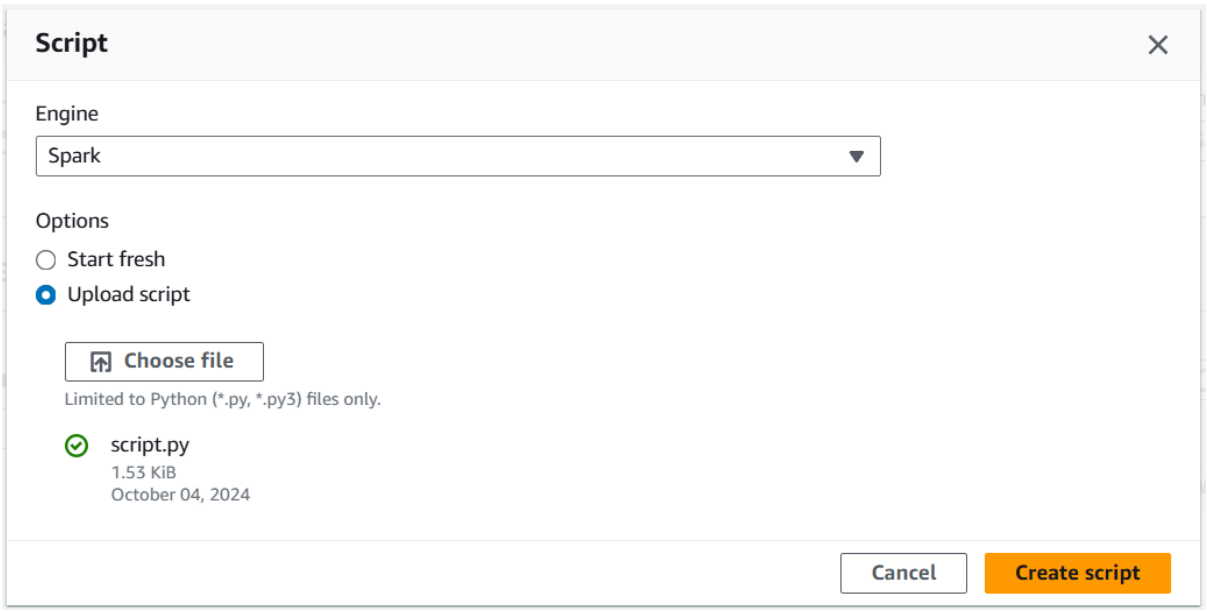
13. Also, if you go inside your database, you can see your table there too.



14. Now to create an ETL job, from the left pane choose ETL jobs. Then click on Script Editor.



15. Then choose Spark as your Engine and choose Upload script to upload your script which is given to you with the lab and click on create script.

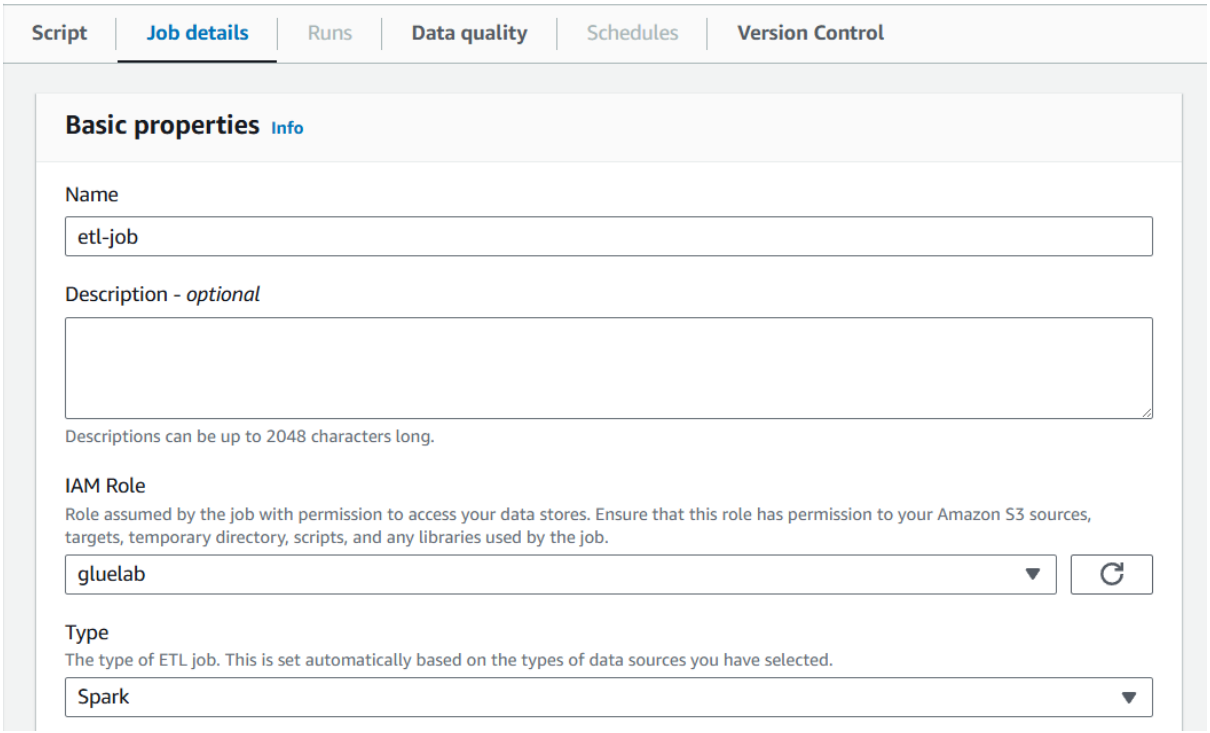


16. Once your script has been uploaded, in lines 16, 17, and 19 you need to mention the name of your glue database, glue table, and the s3 bucket name as you can see below.

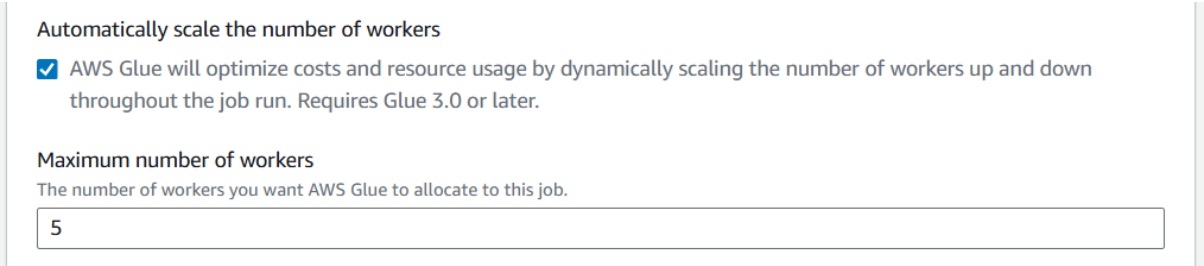
Script Info

```
9
10  spark_context = SparkSession.builder.getOrCreate()
11  #spark_context = SparkContext.getOrCreate()
12
13  glue_context = GlueContext(spark_context)
14  session = glue_context.spark_session
15
16  glue_db = "demo-database"
17  glue_tbl = "s3_bucket_logs_1852"
18  # s3://aws-ml-13123123/Case.csv
19  s3_write_path = "s3://s3-bucket-logs-1852"
20
21  dt_start = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
22  print("Start time:", dt_start)
```

17. Then go to job details and give a name to your job, choose the same IAM role that you have chosen for the crawlers.

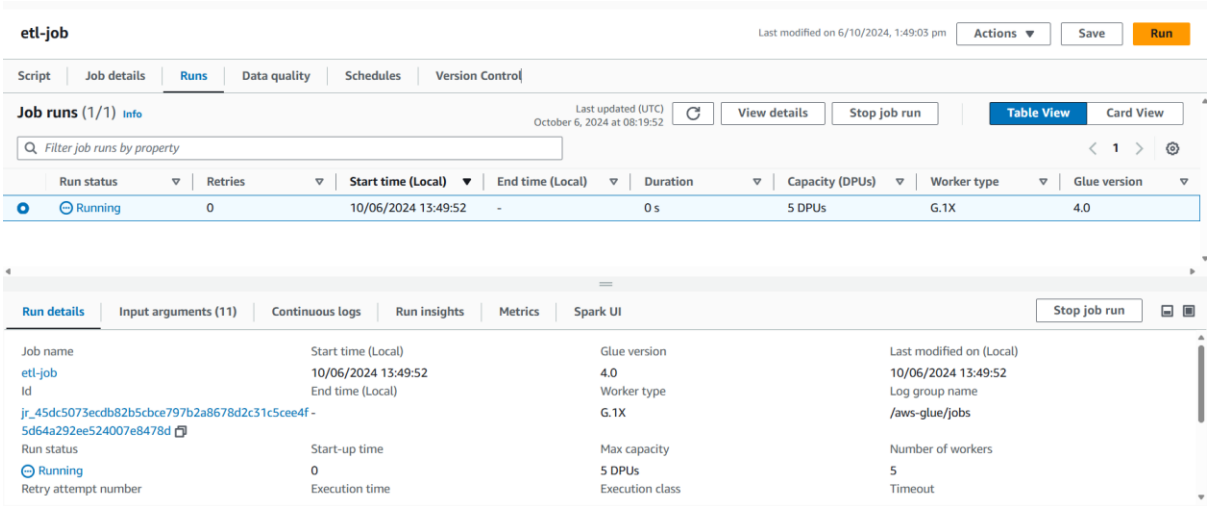


18. After that scroll down and enable Automatically scale the number of workers and in the maximum number of workers write 5.
19. Then just keep everything to default and save your job.

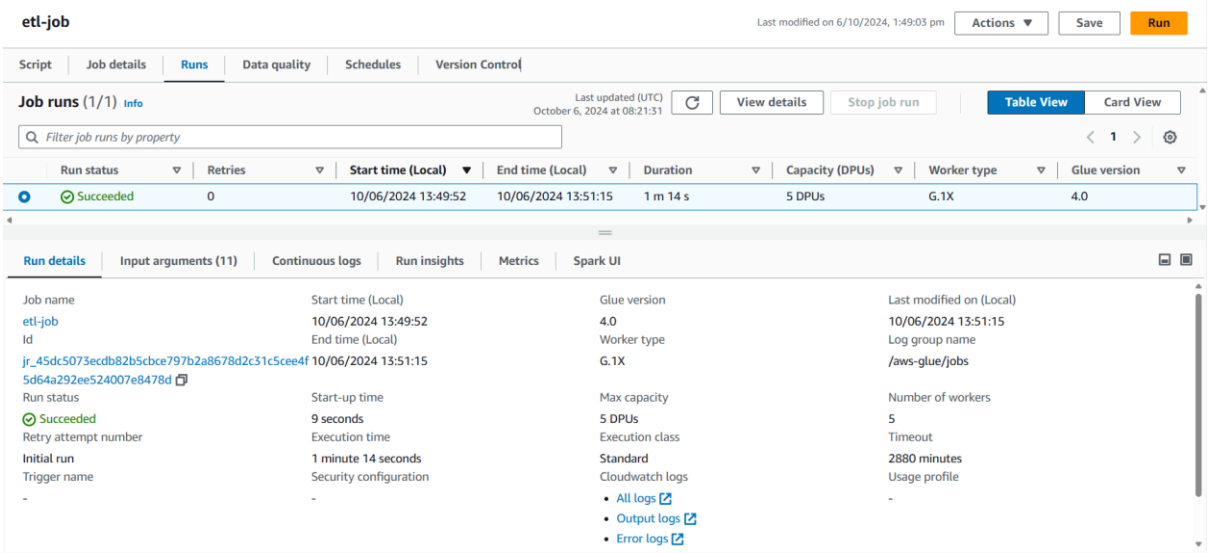


20. Below you can see that your job has been saved successfully but you need to run it manually.

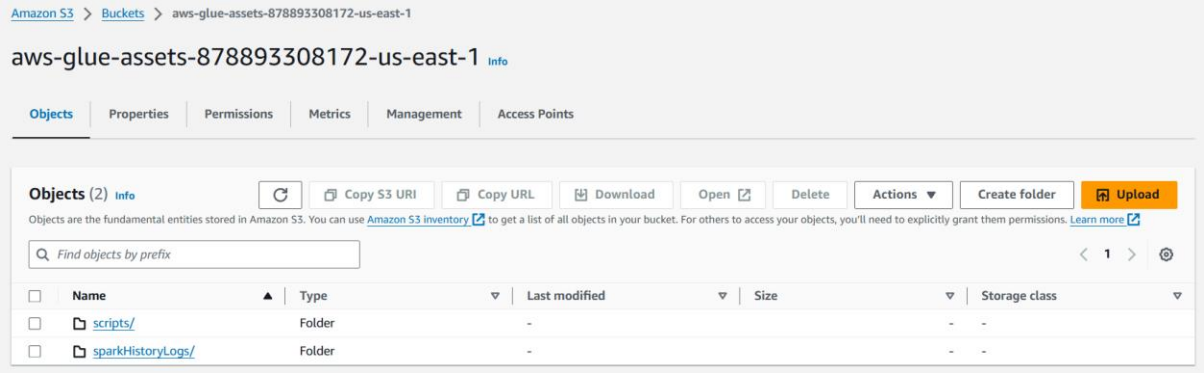
21. And if you go to runs you will see that your job has been running.



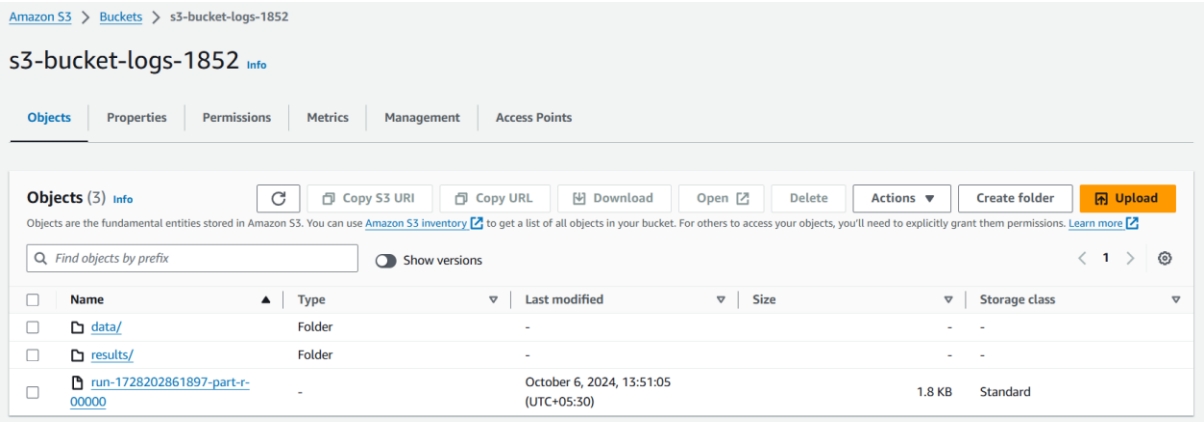
22. Below you can see that the run status has been changed to succeed.



23. Now if you go to S3 you will see that a new bucket has been created where your script was saved and the logs that the ETL job has created. You can go inside these folders and check out the files.



24. Navigate to the bucket that you have mentioned to store the data, here you will see that a new object has been added to your bucket. Now you should download this object.



25. Open this file in Excel, if you cannot open this file in Excel then you can rename the file and add a .csv extension.
26. You can also compare the file to see what has changed. Also, you must read the script because the changes were written in the script.

	A	B	C	D
1	province	city	TotalConfirmed	MaxFromOneConfirmedCase
2	Gyeongsangnam-do	-	39	18
3	Seoul	Seongdong-gu	13	13
4	Busan	Suyeong-gu	5	5
5	Daejeon	Seo-gu	3	3
6	Gyeonggi-do	Uijeongbu-si	50	50
7	Gangwon-do	Wonju-si	3	3
8	Chungcheongbuk-do	from other city	6	6
9	Chungcheongnam-do	-	25	14
10	Jeju-do	from other city	1	1
11	Chungcheongbuk-do	Goesan-gun	11	11
12	Jeollabuk-do	from other city	1	1
13	Gyeongsangbuk-do	-	336	192
14	Jeju-do	-	13	9
15	Seoul	Eunpyeong-gu	14	14
16	Seoul	Jung-gu	7	7
17	Busan	Dongnae-gu	39	39
18	Gyeongsangbuk-do	from other city	615	566
19	Seoul	from other city	8	8
20	Busan	from other city	13	12
21	Daegu	Dong-gu	37	37
22	Gyeonggi-do	Seongnam-si	94	72
23	Gyeonggi-do	Suwon-si	10	10

run-1728202861897-part-r-00000

27. Once you are done then delete all the resources. Start with database, crawlers, tables and ETL jobs.