



# **Amazon Web Services Data Engineering Immersion Day**

---

Extract, Transform and Load Data Lake with Glue

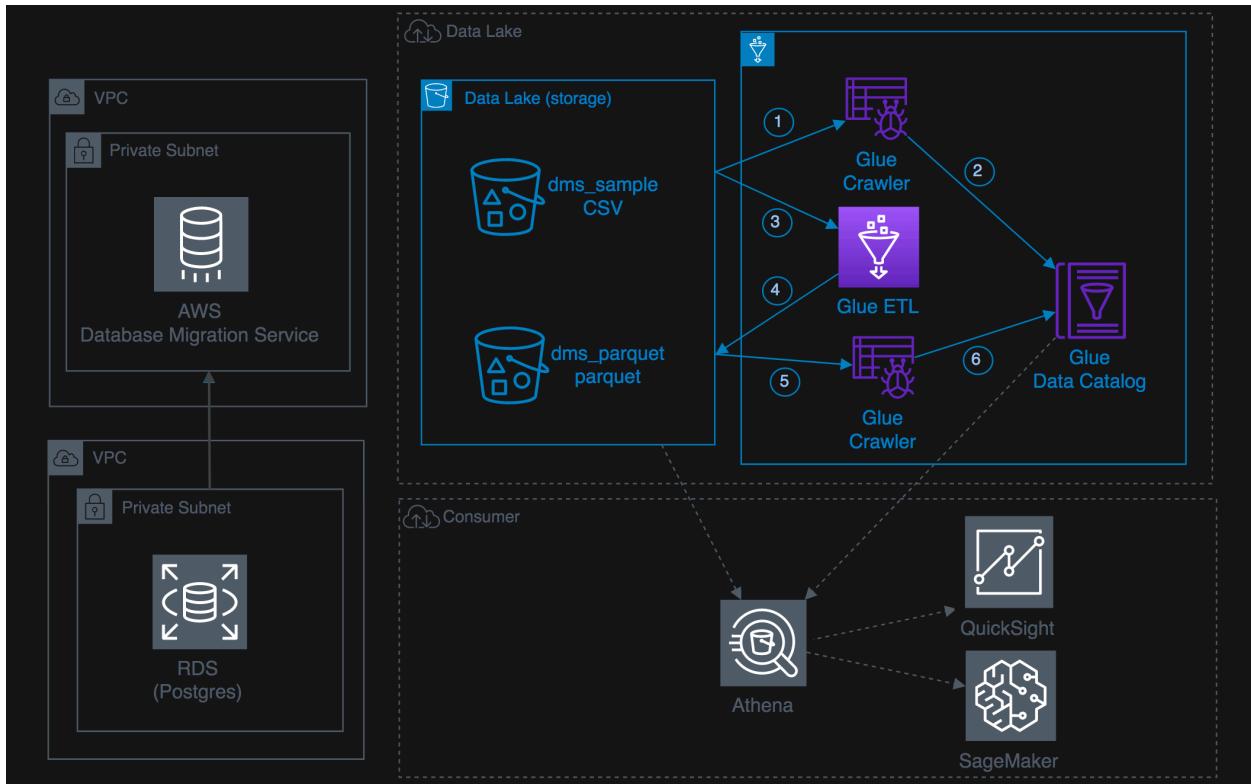
***March 2020***

## Table of Contents

<b><i>Introduction .....</i></b>	<b>2</b>
<b>Prerequisites: .....</b>	<b>2</b>
<b>Tasks Completed in this Lab:.....</b>	<b>2</b>
<b>Getting Started.....</b>	<b>3</b>
<b><i>PART-(A): Data Validation and ETL.....</i></b>	<b>3</b>
Create Glue Crawler for initial full load data .....	3
Data ETL Exercise .....	9
Create Glue Crawler for Parquet Files .....	14
<b><i>PART-(B): Glue Job Bookmark (Optional):.....</i></b>	<b>17</b>
Step 1: Create Glue Crawler for ongoing replication (CDC Data) .....	17
Step 2: Create a Glue Job with Bookmark Enabled .....	21
Step 3: Create Glue crawler for Parquet data in S3 .....	23
Step 4: Generate CDC data and to observe bookmark functionality .....	26
<b><i>PART- ( C ): Glue Workflows (Optional) .....</i></b>	<b>30</b>
Overview: .....	30
Creating and Running Workflows:.....	31

# Introduction

This lab will give you an understanding of the AWS Glue – a fully managed data catalog and ETL service



## Prerequisites:

The DMS Lab is a prerequisite for this lab.

## Tasks Completed in this Lab:

In this lab you will be completing the following tasks. You can choose to complete only Part-(A) to move to next lab where tables can be queried using Amazon Athena and Visualize with Amazon Quicksight

1. [PART-\(A\): Data Validation and ETL](#)
  - a. [Create Glue crawler for initial full load data](#)
  - b. [Create Glue ETL to transform CSV data to Parquet format](#)
  - c. [Create Glue crawler to crawl Parquet Data to build Data Catalog](#)
2. [PART- \(B\): Glue Job Bookmark Functionality\(Optional\)](#)
  - a. [Create Glue crawler for ongoing replication](#)
  - b. [Create Glue Job with Bookmark enabled](#)

- c. [Create Glue crawler to create initial CDC Data catalog](#)
  - d. [Generate CDC and observe bookmark functionality](#)

3. **PART- ( C ): Glue Workflows(Optional)**

  - a. [Overview](#)
  - b. [Creating and Running workflows](#)

Labs are also available in the self-paced web portal - <https://aws-dataengineering-day.workshop.aws>

# Getting Started

**Navigate to the AWS Glue service.**



## PART-(A): Data Validation and ETL

## Create Glue Crawler for initial full load data

1. On the AWS Glue menu, select **Crawlers**.

AWS Glue

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Add crawler Run crawler Action Filter by tags and attributes Showing: 0 - 0 < > 

<input type="checkbox"/>	Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
You don't have any crawlers yet.								

 Add crawler

2. Click **Add crawler**.
  3. Enter the crawler name for initial data load. This name should be descriptive and easily recognized (e.g., "glue-lab-crawler").
  4. Optionally, enter the description. This should also be descriptive and easily recognized and Click **Next**.

**Add crawler**

Add information about your crawler

Crawler name  
glue-lab-crawler

Tags, description, security configuration, and classifiers (optional)

**Next**

Crawler info  
 Crawler source type  
 Data store  
 IAM Role  
 Schedule  
 Output  
 Review all steps

## 5. Choose Data Stores as Crawler Source Type and Click Next

**Add crawler**

Specify crawler source type

Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

Crawler source type  
 Data stores  
 Existing catalog tables

**Back** **Next**

Crawler info  
glue-lab-crawler  
 Crawler source type  
 Data store  
 IAM Role  
 Schedule  
 Output  
 Review all steps

6. On the **Add a data store** page, make the following selections:
  - a. For Choose a data store, click the drop-down box and select **S3**.
  - b. For Crawl data in, select **Specified path in my account**.
  - c. For Include path, browse to the target folder that contains 'dmslab', for example: "s3://dmslab-student-dmslabs3bucket-woti4bf73cw3/tickets
7. Click **Next**.

**Add crawler**

Add a data store

Choose a data store  
S3

Crawl data in  
 Specified path

Include path  
s3://dmslab-student-dmslabs3bucket-1xby1wp8fe8iq/tickets

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

Exclude patterns (optional)

**Back** **Next**

Crawler info  
glue-lab-crawler  
 Crawler source type  
Data stores  
 Data store  
S3: s3://dmslab-stu...  
 IAM Role  
 Schedule  
 Output  
 Review all steps

8. On the **Add another data store** page, select **No**. and Click **Next**.

**Add crawler**

Add another data store

Chosen data stores  
S3: s3://dmslab-stu...

Yes  
 No

**Back** **Next**

Crawler info  
glue-lab-crawler  
 Crawler source type  
Data stores  
 Data store  
S3: s3://dmslab-stu...  
 IAM Role  
 Schedule  
 Output  
 Review all steps

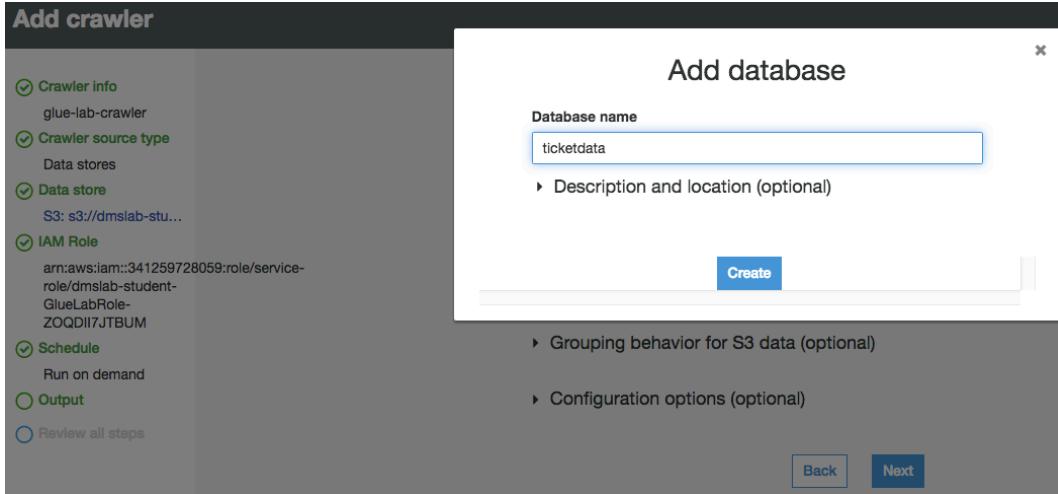
9. On the **Choose an IAM role** page, make the following selections:
- Select **Choose an existing IAM role**.
  - For IAM role, select <stackname>-GlueLabRole-<RandomString> created from the AWS CloudFormation template initially. For example, "dmslab-student-GlueLabRole-ZOQDII7JTBUM"

10. Click **Next**.

11. On the **Create a schedule for this crawler** page, for Frequency, select **Run on demand** and Click **Next**.

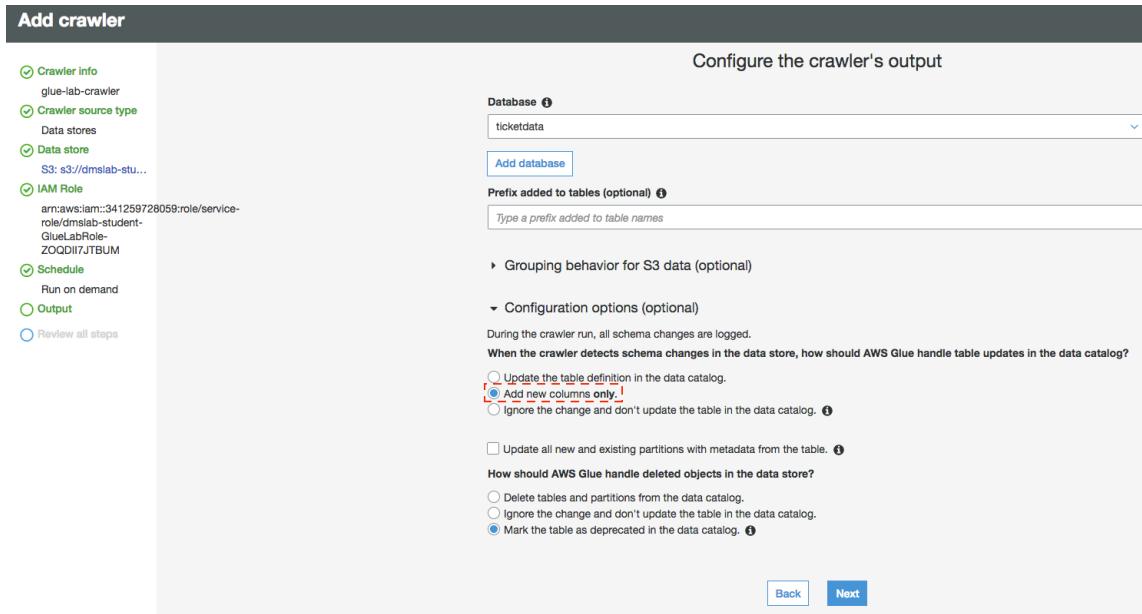
12. On the **Configure the crawler's output** page, click **Add database** to create a new database for our Glue Catalogue.

13. Give Catalog database name as per your convenient choice for example "ticketdata" and click **Create**

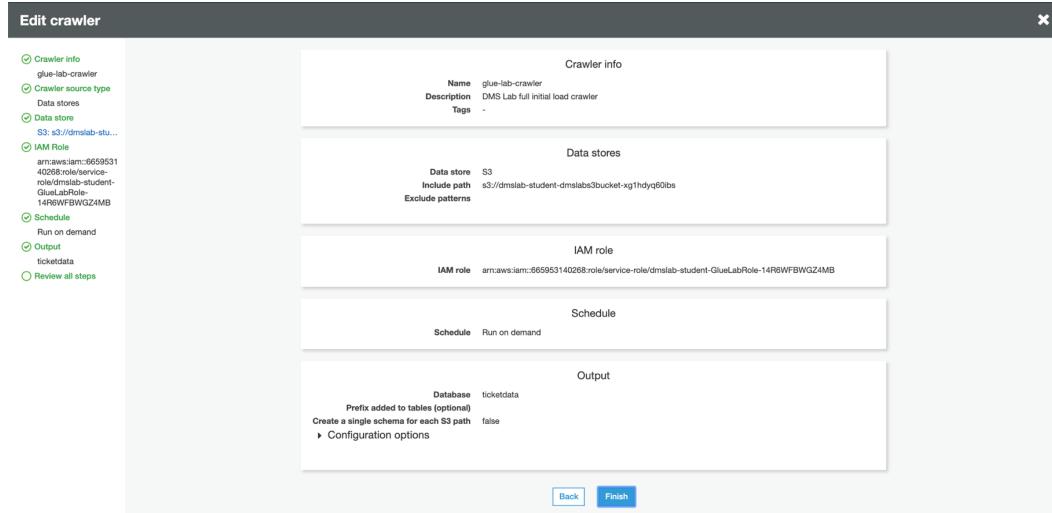


14. For Prefix added to tables (optional), leave the field empty.

15. For Configuration options (optional), select **Add new columns only** and keep the remaining default configuration options and Click **Next**.



16. Review the summary page noting the Include path and Database output and Click **Finish**. The crawler is now ready to run.



## 17. Click Run it now.

AWS Glue

- Data catalog
- Databases
- Tables
- Connections
- Crawlers**
- Classifiers
- Settings

Crawlers A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawler glue-lab-crawler was created to run on demand. [Run it now!](#)

Name	Schedule	Catalog type	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
glue-lab-crawler		Glue	Ready		0 secs	0 secs	0	0

Crawler will change status from starting to stopping, wait until crawler comes back to ready state, you can see that it has created 15 tables.

AWS Glue

- Data catalog
- Databases
- Tables
- Connections
- Crawlers**
- Classifiers
- Settings

Crawlers A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawler "glue-lab-crawler" completed and made the following changes: 15 tables created, 0 tables updated. See the tables created in database ticketdata.

Name	Schedule	Catalog type	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
glue-lab-crawler		Glue	Ready	Logs	1 min	1 min	0	15

## 18. In the AWS Glue navigation pane, click Databases > Tables. (You can also click the database name (e.g., "ticketdata" to browse the tables.).

AWS Glue

- Data catalog
- Databases**
- Tables
- Connections
- Crawlers
- Classifiers
- Settings
- ETL
- Workflows
- Jobs
- ML Transforms
- Triggers
- Dev endpoints
- Notebooks
- Security
- Security configurations
- Tutorials

Tables A table is the metadata definition that represents your data, including its schema. A table can be used as a source or target in a job definition.

Name	Database	Location	Classification	Last updated	Deprecated
mbo_data	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
name_data	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
nfl_data	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
nfl_stadium_data	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
person	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:48 PM ...	
player	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
seat	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
seat_type	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
sport_division	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
sport_league	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
sport_location	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
sport_team	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
sporting_event	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
sporting_event_ticket	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	
ticket_purchase_hist	ticketdata	s3://dmslab-student-dmsl...	csv	10 January 2020 1:37 PM ...	

## Data Validation Exercise

1. Within the Tables section of your ticketdata database, click the person table.

The screenshot shows the AWS Glue Data Catalog interface. On the left, there's a sidebar with various navigation options like Databases, Connections, Crawlers, Classifiers, ETL, Workflows, Jobs, ML Transforms, Triggers, Dev endpoints, and Notebooks. The 'Tables' option is selected. The main area displays a table of tables with columns: Name, Database, Location, Classification, Last updated, and Deprecated. One row for the 'person' table is highlighted with a red box around its name. The 'Database' column for the 'person' table is 'ticketdata'. The 'Location' column shows an S3 path. The 'Last updated' column indicates the table was last updated on January 10, 2020, at 1:48 PM UTC-5.

You may have noticed that some tables (such as person) have column headers such as col0,col1,col2,col3. In absence of headers or when the crawler cannot determine the header type, default column headers are specified.

This exercise uses the person table in an example of how to resolve this issue.

2. Click **Edit Schema** on the top right side.

The screenshot shows the AWS Glue Table Editor for the 'person' table. At the top, there are buttons for 'Edit table' and 'Delete table'. To the right, there are buttons for 'View properties', 'Compare versions', and 'Edit schema' (which is highlighted with a red box). The main area has tabs for 'Description', 'Classification', 'Location', 'Deprecation', 'Last updated', 'Input format', 'Output format', 'Serde serialization info', 'Serde parameters', and 'Table properties'. Under 'Table properties', there is a table with columns: sizeKey, objectCount, UPDATED\_BY\_CRAWLER, glue-lab-crawler, CrawlerSchemaSerializerVersion, recordCount, averageRecordSize, and CrawlerSchemaDeserializerVersion. The 'Schema' tab is selected, showing a table with columns: Column name, Data type, Partition key, and Comment. The data shows four columns: col0 (string), col1 (string), col2 (string), and col3 (string).

3. In the Edit Schema section, double-click **col0** (column name) to open edit mode. Type "id" as the column name.
4. Repeat the preceding step to change the remaining column names to match those shown in the following figure.

Column name	Data type	Key	Comment
1 id	string		x
2 full_name	string		x
3 last_name	string		x
4 first_name	string		x

## 5. Click **Save**.

## Data ETL Exercise

**Pre-requisite:** To store processed data you need new location. Please Follow this user guide to create folder following structure in your S3 bucket to store parquet file - <https://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-folder.html> .

Full path will look like e.g. – “s3://dmslab-student-dmslabs3bucket-woti4bf73cw3/tickets/**dms\_parquet/sport\_team**”

### 1. In the left navigation pane, under **ETL**, click **Jobs**, and then click **Add job**.

Name	ETL language	Script location	Last modified	Job bookmark
You don't have any jobs defined yet.				

### 2. On the Job properties page, make the following selections:

- a. For **Name**, type “Glue-Lab-SportTeamParquet”
- b. For **IAM role**, choose existing role e.g. “xxx-GlueLabRole-xxx”
- c. For **Type**, Select “Spark”
- d. For **Glue Version**, select “Spark 2.4, Python 3(Glue version 1.0) ” or whichever is the latest version.
- e. For **This job runs**, select “A proposed script generated by AWS Glue”.
- f. For **Script file name**, keep the default.
- g. For **S3 path where the script is stored**, provide a unique Amazon S3 path to store the scripts. (You can keep the default for this lab.)
- h. For **Temporary directory**, provide a unique Amazon S3 directory for a temporary directory. (You can keep the default for this lab.)

**Add job**

Configure the job properties

**Name:** Glue-Lab-SportTeamParquet

**IAM role:** dmslab-student-GlueJobRole-1486WF9WGZ4MB

**Type:** Spark

**Glue version:** Spark 2.x, Python 3 (Glue version 1.0)

**This job runs:** A proposed script generated by AWS Glue

**Script file name:** Glue-Lab-SportTeamParquet

**S3 path where the script is stored:** s3://aws-glue-scripts-665953140268-us-east-1/demo\_user

**Temporary directory:** s3://aws-glue-temporary-665953140268-us-east-1/demo\_user

**Next**

### 3. Click Next

4. On the Choose your data sources page, select **sport\_team** and Click **Next**.

**Add job**

Choose a data source

**Data source:** sport\_team

Name	Database	Location	Classification
sport_team	ticketdata	s3://dmslab-student-dmslabs3bucket-wot4bf73cw3/tick...	csv
sportstickets_dms_sample_sport_team	onprem-db	sportstickets.dms_sample.sport_team	postgresql

**Back** **Next**

5. On the Choose a transformation type page, select **change schema**

**Add job**

Choose a transform type

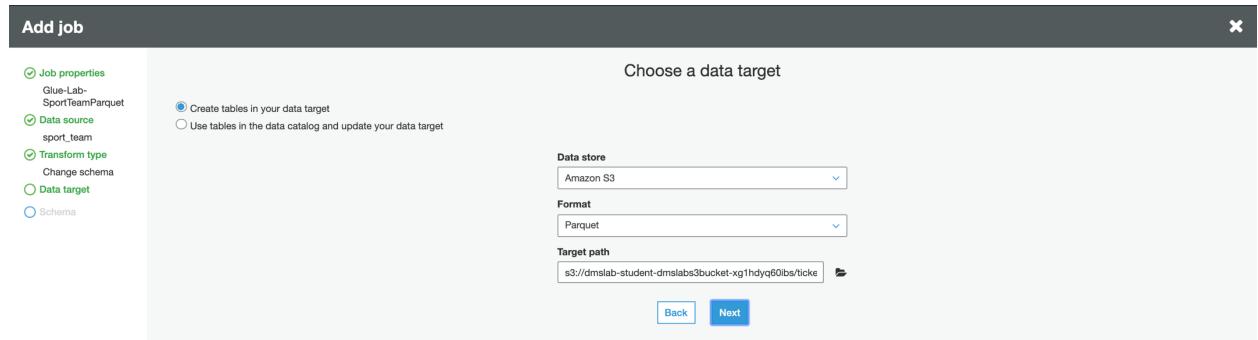
**Change schema:** Change schema of your source data and create a new target dataset

**Find matching records:** Use machine learning to find matching records within your source data

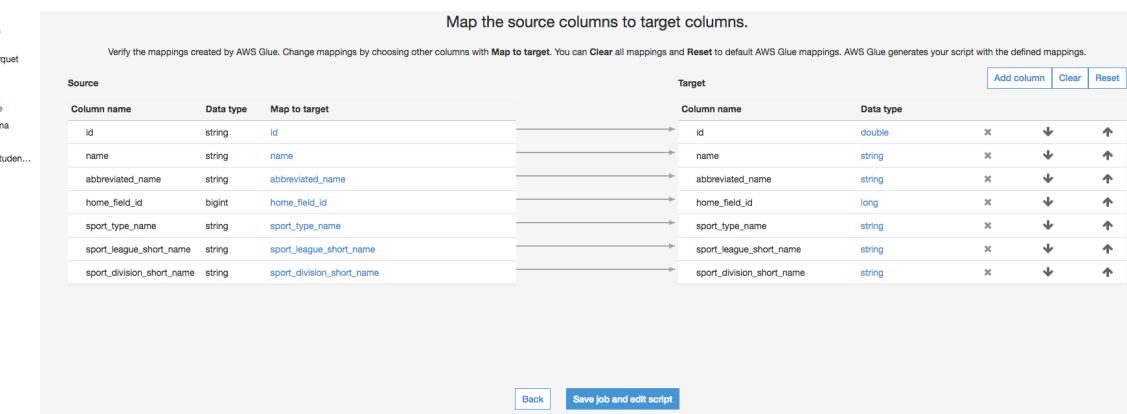
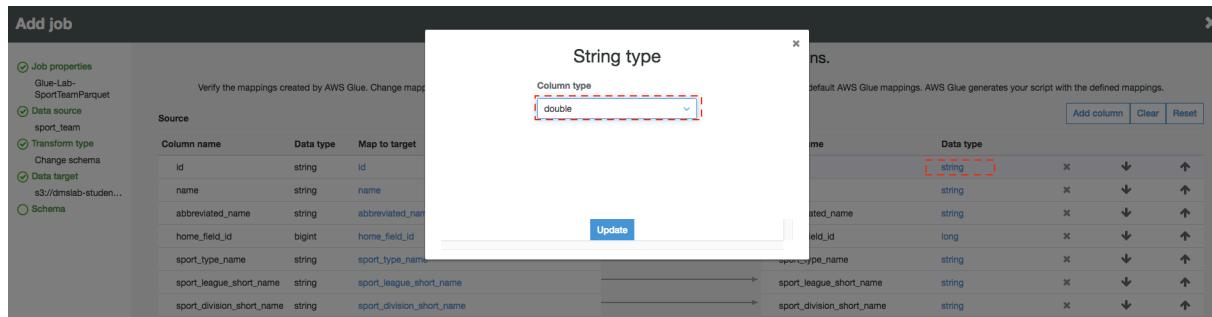
**Back** **Next**

6. On the Choose your data targets page, select **Create tables in your data target**.
7. For Data store, select **Amazon S3**.
8. For Format, select **Parquet**.
9. For **Target path**, click the **folder icon** and choose a location which you create at the beginning of this section to store the results. For example: "  
s3://dmslab-student-dmslabs3bucket-woti4bf73cw3/tickets/dms\_parquet/sport\_team"

**10. Click Next.**



11. Click the target **Data type** to edit the id schema mapping. In **String type** pop-up window Select **double** from **Column type** drop down and click **update**.



12. Click **Save job and edit script**.

**13. View the job. (This screen provides you with the ability to customize this script as required.) Click **Save** and then **Run Job**.**

The screenshot shows the AWS Glue Job Editor interface. At the top, there are buttons for Action (with Save and Run job highlighted), Generate diagram, and other options like Insert template at cursor, Source, Target, Target Location, Transform, Spigot, and Help.

The job flow consists of four steps:

- Database Name ticketdata** (Table Name sport\_team)
- Transform Name ApplyMapping**
- Transform Name ResolveChoice**
- Transform Name DropNullFields**

The Python script in the code editor is as follows:

```

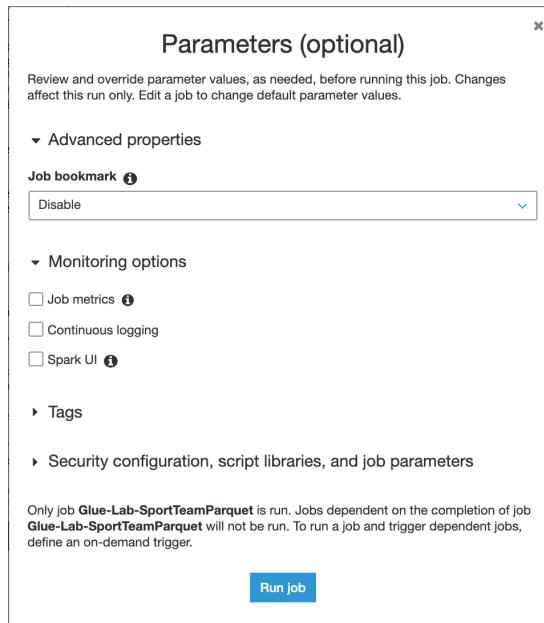
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 ## @params: [JOB_NAME]
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 spark = glueContext.sparkSession
14 job = Job(glueContext)
15 job.setArgs(args)
16 job.setJobName(args['JOB_NAME'])
17 ## @args: [datasource = "ticketdata", table_name = "sport_team", transformation_ctx = "datasource0"]
18 ## @outputs: [datasource0]
19 datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "ticketdata", table_name = "sport_team", transformation_ctx = "datasource0")
20
21 @type: ApplyMapping
22 ## @args: [mapping = [{"id": "string", "id": "double"}, {"name": "string", "name": "string"}, {"abbreviated_name": "string", "abbreviated_name": "string"}, {"home_field_id": "long", "home_field_id": "long"}], @inputs: [datasource0]
23 ## @outputs: [applymapping1]
24 applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [{"id": "string", "id": "double"}, {"name": "string", "name": "string"}, {"abbreviated_name": "string", "abbreviated_name": "string"}, {"home_field_id": "long", "home_field_id": "long"}])
25
26 @type: ResolveChoice
27 ## @args: [choice = "make_struct"]
28 ## @outputs: [applymapping1, resolvechoice2]
29 ## @inputs: [frame = applymapping1]
30 resolvechoice2 = ResolveChoice.apply(frame = applymapping1, choice = "make_struct", transformation_ctx = "resolvechoice2")
31
32 ## @args: [transformation_ctx = "dropnullfields3"]
33 ## @outputs: [dropnullfields3]
34

```

Below the code editor are two tabs: Logs and Schema.

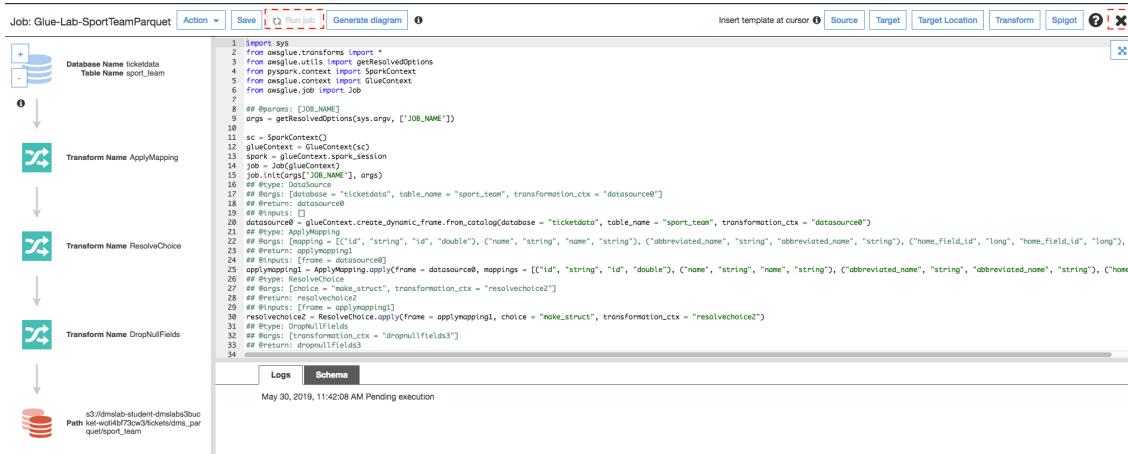
**14. In Parameters option,**

- you can leave Job bookmark as Disable.** AWS Glue tracks data that has already been processed during a previous run of an ETL job by persisting state information from the job run.
- You can leave the Job metrics option Unchecked.** You can collect metrics about AWS Glue jobs and visualize them on the AWS Glue with job metrics.



**15. Click Run Job**

16. You will see job in now running as **Run job** button got disable. Click the cross button located in top right corner to close the window to return to the ETL jobs.



17. Click your job to view history and verify that it ran successfully.

Add job		Action	Filter by tags and attributes		Showing: 1 - 11 < > 🔍						User preferences							
	Name	Type	ETL language	Script location	Last modified	Job bookmark												
<input checked="" type="checkbox"/>	Glue-Lab-SportTeamParquet	Spark	python	s3://aws-glue-scri...	11 March 2020 3:17 PM UTC-7	Disable												
History		Details	Script	Metrics														
<a href="#">View run metrics</a>		<a href="#">Rewind job bookmark</a>		Showing: 1 - 1 < > 🔍														
Run ID		Retry attempt	Run status	Error	Logs	Error logs	Glue version	Maximum capacity	Triggered by	Start time	End time	Execution time	Timeout	Delay	Job run input			
<input type="radio"/>	jr_e37b56aa03cd3...	-	Running	<a href="#">Logs</a>	<a href="#">Error logs</a>	1.0	5			11 Mar...	0 secs	2880 mins		<a href="#">s3://aws-glue-tem...</a>				

You need to repeat the preceding steps to create 4 more new ETL Jobs to transform the additional tables. You will use transformed data in next Athena lab. You can continue creating below ETL job without waiting for previous job to finish.

To enable us to join data, we will also update the target data types in the schema. If below **Table1** is indicating need Schema changes as "Yes". Refer **Table 2** find out column which need to changes with source and target data type during ETL job creation.

**Table 1:**

Job Name & Script Filename	Source Table	S3 Target Path	Need Schema Change?
<b>Glue-Lab- SportLocationParquet</b>	sport_location	tickets/dms_parquet/sport_location	No
<b>Glue-Lab- SportingEventParquet</b>	sporting_event	tickets/dms_parquet/sporting_event	Yes

<b>Glue-Lab-SportingEventTicketParquet</b>	sporting_event_ticket	tickets/dms_parquet/sporting_event_ticket	Yes
<b>Glue-Lab-PersonParquet</b>	person	tickets/dms_parquet/person	Yes

**Table 2:**

Job Name	Table	Column	Source Data Type	Target Data Type
<b>Glue-Lab-SportingEventTicketParquet</b>	sporting_event	start_date_time	STRING	TIMESTAMP
<b>Glue-Lab-SportingEventTicketParquet</b>	sporting_event	start_date	STRING	DATE
<b>Glue-Lab-SportingEventTicketParquet</b>	sporting_event_ticket	id	STRING	DOUBLE
<b>Glue-Lab-SportingEventTicketParquet</b>	sporting_event_ticket	sporting_event_id	STRING	DOUBLE
<b>Glue-Lab-SportingEventTicketParquet</b>	sporting_event_ticket	ticketholder_id	STRING	DOUBLE
<b>Glue-Lab-PersonParquet</b>	person	id	STRING	DOUBLE

Once these jobs have completed, we can create a crawler to index these parquet files.

## Create Glue Crawler for Parquet Files

1. In the AWS Glue navigation menu, click **Crawlers**, and then click **Add crawler**.

The screenshot shows the AWS Glue interface with the 'Crawlers' tab selected. A new crawler named 'glue-lab-crawler' is listed, showing it is Ready with 1 min runtime and 15 tables added. The 'Add crawler' button is highlighted.

2. For **Crawler name**, type "glue-lab-parquet-crawler" and Click **Next**.

The screenshot shows the 'Add crawler' wizard step 1: Crawler info. It shows the crawler name 'glue-lab-parquet-crawler' and a note to add information about the crawler.

3. In next screen **Specify crawler source type**, select **Data Stores** as choice for **Crawler resource type** and click **Next**.
4. In Add a data store screen
  - a. For **Choose a data store**, select "S3".
  - b. For **Crawl data in**, select "Specified path in account".

- c. For Include path, specify the S3 Path (Parent Parquet folder) that contains the transformed parquet files e.g., s3://xx-dmslabs3bucket-xxx/tickets/dms\_parquet
- d. Click **Next**.

Add a data store

Choose a data store

S3

Crawl data in

Specified path in my account  
 Specified path in another account

Include path

s3://dmslab-student-dmslabs3bucket-wotl4bf73cw3/tickets/dms\_parquet

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

Exclude patterns (optional)

Back Next

- 5. For Add another data store, select **No** and Click **Next**.

Add crawler

Add another data store

Yes  
 No

Chosen data stores

S3: s3://dmslab-stu...

Back Next

- 6. On the Choose an IAM role page, select **Choose an existing IAM role**. For IAM role, select the existing role, something similar to "dmslab-student-GlueLabRole-ZOQDII7JTBUM" and Click **Next**.

Add crawler

Choose an IAM role

The IAM role allows the crawler to run and access your Amazon S3 data stores. [Learn more](#)

Update a policy in an IAM role  
 Choose an existing IAM role  
 Create an IAM role

IAM role

dmslab-student-GlueLabRole-14R6WFBWGZ4MB

This role must provide permissions similar to the AWS managed policy, AWSGlueServiceRole, plus access to your data stores.

• s3://dmslab-student-dmslabs3bucket-xg1hdqg0iba/tickets/dms\_parquet

You can also create an IAM role on the [IAM console](#).

Back Next

- 7. For Frequency, select "Run On Demand" and Click **Next**.

**Add crawler**

Crawler info: glue-lab-parquet-crawler

Crawler source type: Data stores

Data store: S3: s3://dmslab-stu...

IAM Role: arn:aws:iam::341259728059:role/service-role/dmslab-student-GlueLabRole-ZOQDII7JTBUM

Schedule: Run on demand

Output: ticketdata

Review all steps

Create a schedule for this crawler

Frequency: Run on demand

Back Next

8. For the crawler's output database, choose your existing database which you created earlier e.g. "ticketdata"
9. For the **Prefix added to tables** (optional), type "**parquet\_**"

**Add crawler**

Crawler info: glue-lab-parquet-crawler

Crawler source type: Data stores

Data store: S3: s3://dmslab-stu...

IAM Role: arn:aws:iam::341259728059:role/service-role/dmslab-student-GlueLabRole-ZOQDII7JTBUM

Schedule: Run on demand

Output: ticketdata

Review all steps

Configure the crawler's output

Database: ticketdata

Add database

Prefix added to tables (optional): parquet\_

Grouping behavior for S3 data (optional)

Configuration options (optional)

Back Next

10. Review the summary page and click **Finish**.

**Add crawler**

Crawler info: Name: glue-lab-parquet-crawler Tags: -

IAM role: IAM role: arn:aws:iam::665953140268:role/service-role/dmslab-student-GlueLabRole-14R6WFBWGZ4MB

Schedule: Schedule: Run on demand

Output: Database: ticketdata Prefix added to tables (optional): parquet\_ Create a single schema for each S3 path: false Configuration options

Back Finish

11. On the notification bar, click **Run it now**. Once your crawler has finished running, you should report that 5 tables were added.

Confirm you can see the tables:

1. In the left navigation pane, click **Tables**.
2. Add the filter "parquet" to return the newly created tables.

## PART-(B): Glue Job Bookmark (Optional):

**\*\*Pre-requisite: Generate the CDC Data as part of DMS Lab\*\***

Step 1: Create Glue Crawler for ongoing replication (CDC Data)

Now, let's repeat this process to load the data from change data capture.

1. On the AWS Glue menu, select Crawlers.

2. Click **Add crawler**.
3. Enter the crawler name for ongoing replication. This name should be descriptive and easily recognized (e.g., "glue-lab-cdc-crawler").
4. Optionally, enter the description. This should also be descriptive and easily recognized and Click **Next**.

**Add crawler**

Crawler info  
 Crawler source type  
 Data store  
 IAM Role  
 Schedule  
 Output  
 Review all steps

Add information about your crawler

Crawler name  
glue-lab-cdc-crawler

Tags, description, security configuration, and classifiers (optional)

**Next**

5. Choose **Data Stores** as **Crawler Source Type** and Click **Next**

**Add crawler**

Crawler info  
 glue-lab-cdc-crawler

Crawler source type  
 Data stores  
 Data store  
 S3 null  
 IAM Role  
 Schedule  
 Output  
 Review all steps

Specify crawler source type

Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

Crawler source type  
 Data stores  
 Existing catalog tables

**Back** **Next**

6. On the Add a data store page, make the following selections:
  - a. For **Choose a data store**, click the drop-down box and select **S3**.
  - b. For **Crawl data in**, select **Specified path in my account**.
  - c. For **Include path**, enter the **target folder** for your DMS ongoing replication, e.g., "s3://xxx-dmslabs3bucket-xx/cdc/dms\_sample"
7. Click **Next**.

**Add crawler**

Crawler info  
 glue-lab-cdc-crawler

Crawler source type  
 Data stores  
 Data store  
 IAM Role  
 Schedule  
 Output  
 Review all steps

Add a data store

Choose a data store  
S3

Crawl data in  
 Specified path in my account  
 Specified path in another account

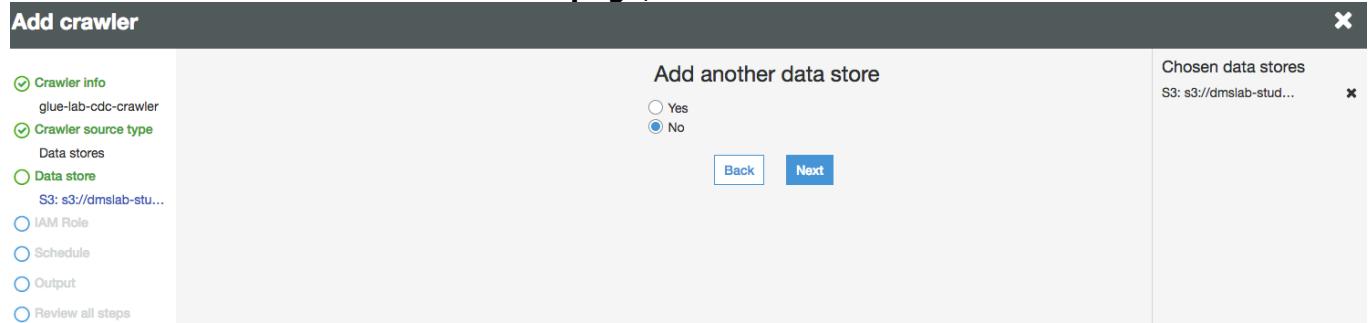
Include path  
s3://dmslab-student-dmslabs3bucket-wotl4bf73cw3/cdc/dms\_sample

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder/ to crawl all objects in MyFolder within MyBucket.

Exclude patterns (optional)

**Back** **Next**

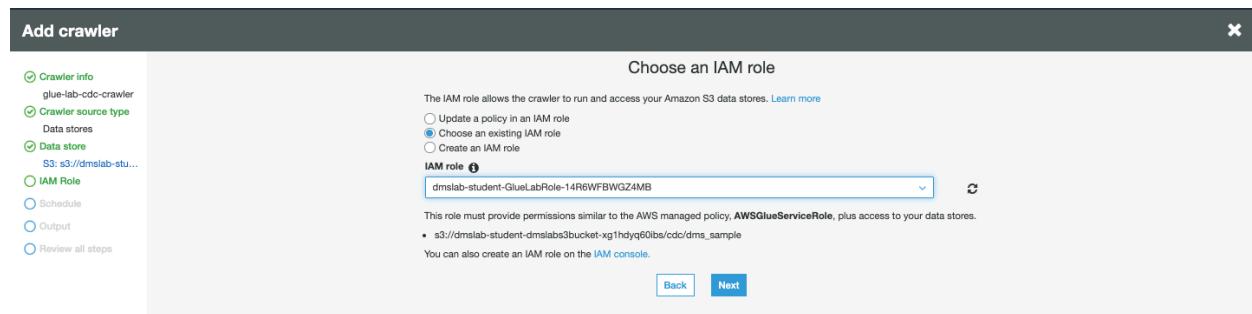
8. On the **Add another data store** page, select **No** and Click **Next**.



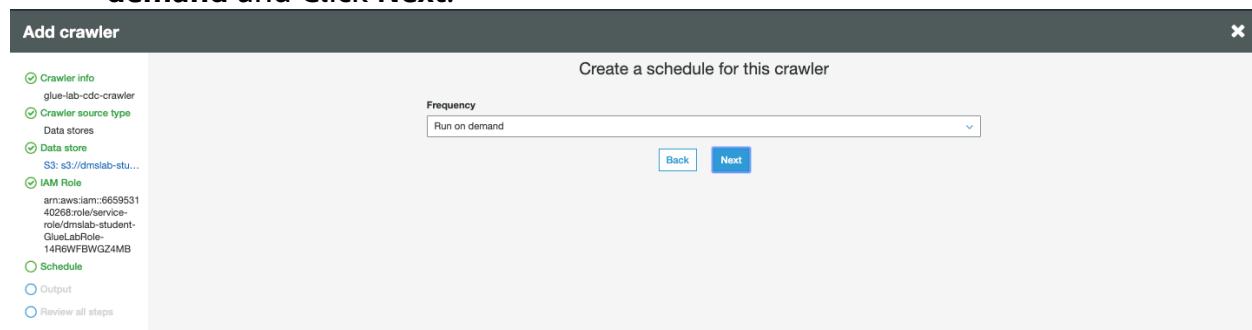
9. On the **Choose an IAM role** page, make the following selections:

- a. Select **Choose an existing IAM role**.
- b. For **IAM role**, select <stackname>-**GlueLabRole**-<RandomString>. E.g. "dmslab-student-GlueLabRole-ZOQDII7JTBUM"

10. Click **Next**.



11. On the **Create a schedule for this crawler** page, for Frequency, select **Run on demand** and Click **Next**.



12. On the **Configure the crawler's output** page, select the existing **Database** for crawler output (e.g., "ticketdata").

13. For **Prefix added to tables (optional)**, specify "cdc\_"

14. For Configuration options (optional), keep the default selections and click **Next**.

**Add crawler**

Configure the crawler's output

**Database** ticketdata

**Prefix added to tables (optional)** cdc\_

**Grouping behavior for S3 data (optional)**

**Configuration options (optional)**

During the crawler run, all schema changes are logged.

When the crawler detects schema changes in the data store, how should AWS Glue handle table updates in the data catalog?

- Update the table definition in the data catalog.
- Add new columns only.
- Ignore the change and don't update the table in the data catalog.

Update all new and existing partitions with metadata from the table.

How should AWS Glue handle deleted objects in the data store?

- Delete tables and partitions from the data catalog.
- Ignore the change and don't update the table in the data catalog.
- Mark the table as deprecated in the data catalog.

**Back** **Next**

15. Review the summary page noting the Include path and Database target and Click **Finish**. The crawler is now ready to run.

**Add crawler**

**Crawler info**  
Name: glue-lab-cdc-crawler  
Tags: -

**IAM role** arn:aws:iam::665953140268:role/service-role/dmslab-student-GlueLabRole-14R6WFBWGZ4MB

**Schedule** Run on demand

**Output** ticketdata

**Review all steps**

**Back** **Finish**

16. Click **Run it now**.

AWS Glue

Data catalog

Databases

Tables

Connections

**Crawlers**

Classifiers

Settings

ETL

Crawlers A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawler glue-lab-cdc-crawler was created to run on demand! **Run it now!**

User preferences

Name	Schedule	Catalog type	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
glue-lab-cdc-cra...	Glue	Ready	0 secs	0 secs	0	0		
glue-lab-crawler	Glue	Ready	Logs	1 min	1 min	0	15	

17. When the crawler is completed, you can see it has "Status" as **Ready**, Crawler will change status from starting to stopping, wait until crawler comes back to ready state, you can see that it has created 2 tables.

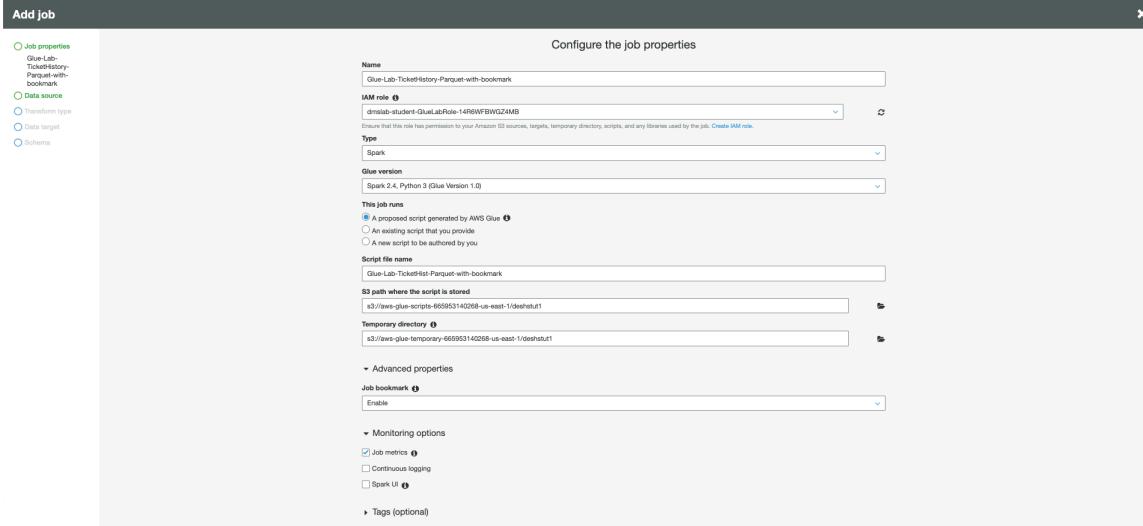
18. Click the database name (e.g., "ticketdata") to browse the tables. Specify "cdc" as the filter to list only newly imported tables.

## Step 2: Create a Glue Job with Bookmark Enabled

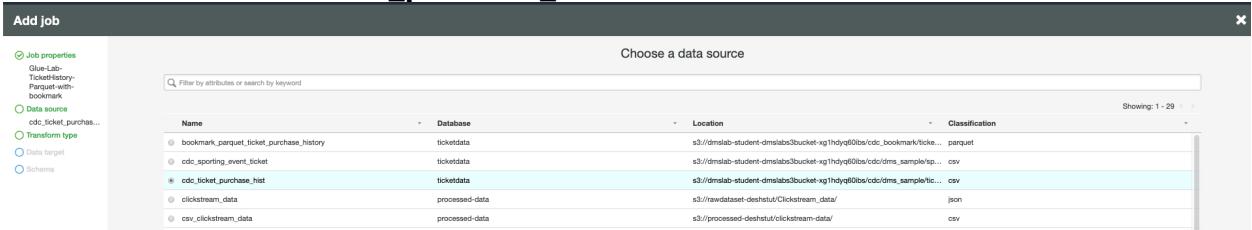
1. On the left-hand side of Glue Console, click on Jobs and then Click on Add Job

2. On the Job properties page, make the following selections:
- For Name, type **Glue-Lab-TicketHistory-Parquet-with-bookmark**.
  - For IAM role, choose existing **GlueLabRole**
  - For Type, Select **Spark**
  - For Glue Version, select **Spark 2.4, Python 3(Glue version 1.0)** or whichever is the latest version.
  - For This job runs, select **A proposed script generated by AWS Glue**.
  - For Script file name, type **Glue-Lab-TicketHistory-Parquet-with-bookmark**.
  - For S3 path where the script is stored, provide a unique Amazon S3 path to store the scripts. (You can keep the default for this lab.)
  - For Temporary directory, provide a unique Amazon S3 directory for a temporary directory. (You can keep the default for this lab.)
3. Expand the **Advanced properties** section. For Job bookmark, select **Enable** from the drop down option.

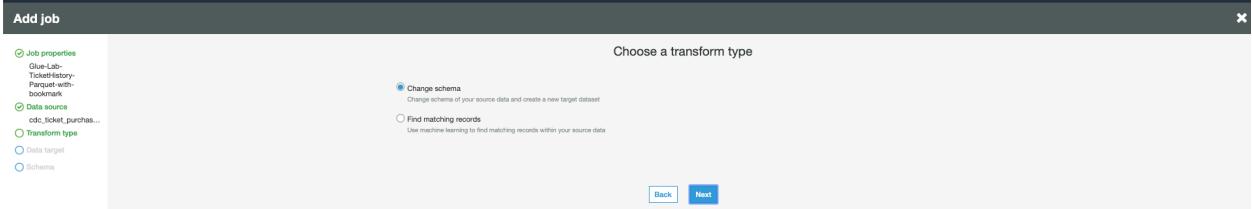
4. Expand on the **Monitoring** options, check for **Job metrics**.
5. Click **Next**



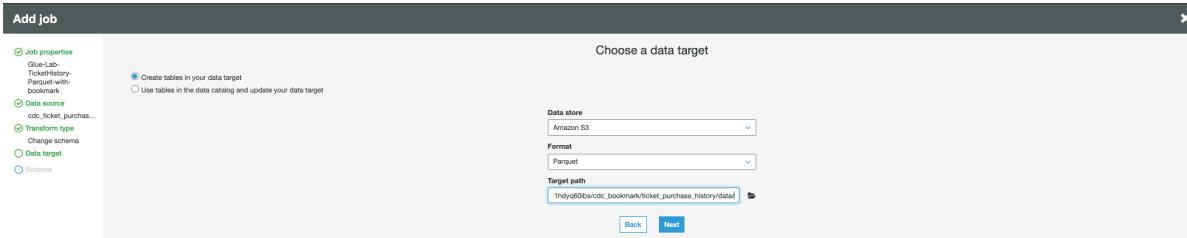
6. In **Choose a data source**, select **cdc\_ticket\_purchase\_hist** as we are generating new data entries for **ticket\_purchase\_hist** table. Click **Next**



7. In **Choose a transform type**, select **Change Schema** and Click **Next**



8. In **Choose a data target**:
  - a. For **Data store**: select **amazon S3**
  - b. **Format**: **parquet**
  - c. **Target path**: **s3://<dms-lab-bucket-path>/cdc\_bookmark/ticket\_purchase\_history\_data/**
  - d. Click **Next**



9. In map the source columns to target columns window, leave everything default and Click on **Save job and edit script**.

Source	Column name	Data type	Map to target	Target	Column name	Data type
	op	string	op		op	string
	sporting_event_ticket_id	string	sporting_event_ticket_id		sporting_event_ticket_id	string
	purchased_by_id	string	purchased_by_id		purchased_by_id	string
	transaction_date_time	string	transaction_date_time		transaction_date_time	string
	transferred_from_id	string	transferred_from_id		transferred_from_id	string
	purchase_price	double	purchase_price		purchase_price	double

10. In the next window, review the job script and click on **Run job**. Click on close mark on the top right of the window to close the screen.

```

1 #!/usr/bin/python
2
3 import sys
4
5 from awsglue.transforms import *
6 from awsglue.utils import getResolvedOptions
7 from pyspark.context import SparkContext
8 from awsglue.context import GlueContext
9
10 # Initialize spark context
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 spark = glueContext.sparkSession
14 job_name = getResolvedOptions(sys.argv, ['JOB_NAME'])
15 job.init(job_name, args)
16
17 ## Merges: [database = "ticketdata", table_name = "cdc_ticket_purchase_hist", transformation_ctx = "datasource0"]
18 ## Iterations: [datasource0]
19 ## ResolveChoice: [choice = "make_struct"]
20 ## DropNullFields: [choice = "dropnullfields"]
21
22 datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "ticketdata", table_name = "cdc_ticket_purchase_hist", transformation_ctx = "datasource0")
23 connection_type = "path"
24 connection_options = {"path": "s3://dmslab-student-dmslabs3bucket-xg1hdyq60ibs/cdc_bookmark/ticket_purchase_history/data/"}
25 applymapping0 = ApplyMapping.apply(frame = datasource0, mappings = [{"op": "string", "string": "op"}, {"sporting_event_ticket_id": "string", "string": "sporting_event_ticket_id"}, {"purchased_by_id": "string", "string": "purchased_by_id"}, {"transaction_date_time": "string", "string": "transaction_date_time"}, {"transferred_from_id": "string", "string": "transferred_from_id"}, {"purchase_price": "double", "double": "purchase_price"}], transformation_ctx = "applymapping0")
26 applymapping0 = ApplyMapping.apply(frame = applymapping0, mappings = [{"choice": "make_struct", "transformation_ctx": "resolvechoice0"}], transformation_ctx = "applymapping0")
27 resolvechoice0 = ResolveChoice.apply(frame = applymapping0, choice = "make_struct", transformation_ctx = "resolvechoice0")
28 dropnullfields0 = DropNullFields.apply(frame = resolvechoice0, transformation_ctx = "dropnullfields0")
29 dropnullfields0 = DropNullFields.apply(frame = dropnullfields0, transformation_ctx = "dropnullfields1")
30 dropnullfields1 = DropNullFields.apply(frame = dropnullfields0, transformation_ctx = "dropnullfields2")
31 dropnullfields2 = DropNullFields.apply(frame = dropnullfields1, transformation_ctx = "dropnullfields3")
32 dropnullfields3 = DropNullFields.apply(frame = dropnullfields2, transformation_ctx = "dropnullfields4")
33 dropnullfields4 = DropNullFields.apply(frame = dropnullfields3, transformation_ctx = "dropnullfields5")
34 dropnullfields5 = DropNullFields.apply(frame = dropnullfields4, transformation_ctx = "dropnullfields6")
35 dropnullfields6 = DropNullFields.apply(frame = dropnullfields5, transformation_ctx = "dropnullfields7")
36 dropnullfields7 = DropNullFields.apply(frame = dropnullfields6, transformation_ctx = "dropnullfields8")
37 dropnullfields8 = DropNullFields.apply(frame = dropnullfields7, transformation_ctx = "dropnullfields9")
38 dropnullfields9 = DropNullFields.apply(frame = dropnullfields8, transformation_ctx = "dropnullfields10")
39 dropnullfields10 = DropNullFields.apply(frame = dropnullfields9, transformation_ctx = "dropnullfields11")
40 datasource1 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields10, connection_type = "path", connection_options = {"path": "s3://dmslab-student-dmslabs3bucket-xg1hdyq60ibs/cdc_bookmark/ticket_purchase_history/data/"}, format = "parquet", transformation_ctx = "datasink0")
41 job.commit()

```

11. Once the job finishes its run, check the **S3 bucket** for the parquet partitioned data.

Name	Last modified	Size	Storage class
part-00000-498ea7fc-2ac1-4787-b431-9e16f5e24a3f-c0000.snappy.parquet	Jan 24, 2020 7:03:16 PM GMT-0500	1.1 MB	Standard
part-00001-498ea7fc-2ac1-4787-b431-9e16f5e24a3f-c0001.snappy.parquet	Jan 24, 2020 7:03:16 PM GMT-0500	1.2 MB	Standard

### Step 3: Create Glue crawler for Parquet data in S3

- Once you have the data in S3 bucket, navigate to **Glue Console** and now we will crawl the parquet data in S3 to create data catalog.
- Click on **Add crawler**

Crawler	Status	Last run
crawler1	Active	1 hour ago

3. In crawler configuration window, provide crawler name as **glue\_lab\_cdc\_bookmark\_crawler** and Click **Next**.

Add information about your crawler

Crawler name  
glue\_lab\_cdc\_bookmark\_crawler

Tags, description, security configuration, and classifiers (optional)

**Next**

4. In specify **crawler source type**, for crawler source type, select **Data stores**. Click **Next**

Specify crawler source type

Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

Crawler source type  
 Data stores  
 Existing catalog tables

**Back** **Next**

5. In **Add a data store**:

- a. For **Choose a data store**, select **S3**
- b. For the path, provide this: **s3://<dms-lab-bucket-path>** and add **/cdc\_bookmark/ticket\_purchase\_history/**.

6. Click on **Next**

Add a data store

Choose a data store  
S3

Crawl data in  
 Specified path

Include path  
s3://dmslab-student-dmslab3bucket-xg1hdyq60bs/cdc\_bookmark/ticket\_purchase\_history/

All folders and files contained in the include path are crawled. For example, type s3://MyBucket/MyFolder to crawl all objects in MyFolder within MyBucket.  
Exclude patterns (optional)

**Back** **Next**

7. For **Add another data store**, select **No** and click **Next**.

Add another data store

Yes  
 No

**Back** **Next**

8. In **Choose an IAM role**, select **Choose an existing IAM role** and select the role that you created as part of the DMS\_Student Lab. (for eg, this role name looks something like this: **dmslab-student-GlueLabRole-<random-alphanumeric-characters>**)

Choose an IAM role

The IAM role allows the crawler to run and access your Amazon S3 data stores. [Learn more](#)

Update a policy in an IAM role  
 Choose an existing IAM role  
 Create an IAM role

IAM role [Edit](#)

This role must provide permissions similar to the AWS managed policy, [AWSGlueServiceRole](#), plus access to your data stores.  

- s3://dmslab-student-dmslabs3bucket-xg1hdqg60bs/cdc\_bookmark/ticket\_purchase\_history/

You can also create an IAM role on the [IAM console](#).

[Back](#) [Next](#)

9. For setting the **frequency** in create a schedule for this crawler, select “Run on demand”. Click **Next**

Create a schedule for this crawler

Frequency

[Back](#) [Next](#)

10. For the crawler’s output:

- For Database, select “**ticketdata**” database.
- Optionally, add prefix to the newly created tables for easy identification. Provide the prefix as “**bookmark\_parquet\_**”
- Click **Next**

Configure the crawler’s output

Database [Edit](#)

Add database [Edit](#)

Prefix added to tables (optional) [Edit](#)

Grouping behavior for S3 data (optional)  
 Configuration options (optional)

[Back](#) [Next](#)

11. Review all the details and click on **Finish**. Next, run the crawler.

Crawler info

Name: glue\_lab\_cdc\_bookmark\_crawler

Tags: -

---

Data stores

Data store: S3  
 Include path: s3://dmslab-student-dmslabs3bucket-xg1hdqg60bs/cdc\_bookmark/ticket\_purchase\_history/  
 Exclude patterns: -

---

IAM role

IAM role: arn:aws:iam::665953140268:role/service-role/dmslab-student-GlueLabRole-14R6WFBWGZ4MB

---

Schedule

Schedule: Run on demand

---

Output

Database: ticketdata  
 Prefix added to tables (optional): bookmark\_parquet\_  
 Create a single schema for each S3 path: false  
 Configuration options: -

[Back](#) [Finish](#)

12. After the crawler finishes running, click on Databases, select “**ticketdata**” and view tables in this database. You will find the newly created table as “**bookmark\_parquet\_ticket\_purchase\_history**”

**13. Once the table is created, click on Action and from dropdown select View Data.**

**14. This will take you to Athena console and a prebuilt query will be run to display few entries of the table.**

**15. Before moving on to next step, note the “recordcount” for “bookmark\_parquet\_ticket\_purchase\_history”. This will be used at the later part. In this example, the current count is: 1521922**

**Step 4: Generate CDC data and to observe bookmark functionality**

Your instructor will generate CDC activity which above migration task will capture, if you ran the instructor setup on your own, then make sure to follow “**Generate the CDC Data**” section from instructor lab.

You may need to wait 5 to 10 minutes for CDC data to first reflect in your RDS postgres database and then picked up by DMS CDC migration task.

1. To make sure the new data has been successfully generated, check the S3 bucket for cdc data, you will see new files generated. Note the time when the files were generated.

Name	Last modified	Size	Storage class
part-00000-00202004-2abc-47c2-8249-d0100b75ddad-c000.snappy.parquet	Jan 24, 2020 9:20:13 PM GMT+0500	0.93 KB	Standard
part-00000-496ea70c-2ac1-47f7-0431-9e1f5ed4242b-c000.snappy.parquet	Jan 24, 2020 7:03:16 PM GMT+0500	1.1 MB	Standard
part-00000-d166723-2158-459a-b88e-a65238462346-c000.snappy.parquet	Jan 20, 2020 11:24:20 PM GMT+0500	1.7 MB	Standard
part-00000-0e00000-0e0000-0d40-43c0-8230-c3ca26119a-c000.snappy.parquet	Jan 25, 2020 10:24:27 PM GMT+0500	7.2 kB	Standard
part-00001-0e00000-0e0000-0d40-43c0-8230-c3ca26119a-c000.snappy.parquet	Jan 24, 2020 9:20:13 PM GMT+0500	66.5 kB	Standard
part-00001-496ea70c-2ac1-47f7-0431-9e1f5ed4242b-c000.snappy.parquet	Jan 24, 2020 7:03:16 PM GMT+0500	1.2 MB	Standard
part-00001-d166723-2158-459a-b88e-a65238462346-c000.snappy.parquet	Jan 25, 2020 11:24:20 PM GMT+0500	1.7 MB	Standard
part-00002-0e00000-0e0000-0d40-43c0-8230-c3ca26119a-c000.snappy.parquet	Jan 24, 2020 9:20:15 PM GMT+0500	1.7 MB	Standard
part-00002-d166723-2158-459a-b88e-a65238462346-c000.snappy.parquet	Jan 25, 2020 11:24:19 PM GMT+0500	1.5 MB	Standard

2. Repeat Step 1 – Step 3. When you run the “glue-lab-cdc-crawler” again, you will see that it will **only update the table with new data**:

Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
glue-lab-cdc-crawler		Stopping	Logs	1 min	1 min	2	0

3. Once the crawler finishes, note down the new record count after the table update. In this example, the new record count is: 1813493.

Column name	Data type	Partition key	Comment
cp	string		
spending_event_ticket_id	string		
purchase_tx_id	string		
transaction_date_time	string		
transferred_from_id	string		
purchase_price	double		
partition_0	string	Partition 0	

4. Once you run the crawler In Step 3, select the “**bookmark\_parquet\_ticket\_purchase\_history**” from the Table section and click on Action->View Data. This will take you to Athena console.

The screenshot shows the AWS Glue Data Catalog interface. On the left, there's a sidebar with navigation links like 'Data catalog', 'Tables', 'Connections', 'Crawlers', 'Classifiers', 'Settings', 'ETL', and 'Workflows'. The main area has tabs for 'Tables' (selected), 'Actions', and 'View details'. A search bar at the top says 'Filter or search for tables...'. Below it, a table lists tables with columns for 'Name', 'Database', 'Location', 'Classification', 'Last updated', and 'Deprecated'. One table, 'bookmark\_parquet\_ticket\_purchase\_history', is highlighted.

## 5. On Athena Console, run the following query:

```
SELECT * FROM "ticketdata"."bookmark_parquet_ticket_purchase_history"
order by transaction_date_time desc limit 100;
```

The screenshot shows the Amazon Athena Query Editor. The left sidebar lists databases and tables, including 'ticketdata' which contains 'bookmark\_parquet\_ticket\_purchase\_history'. The main area shows the query being run: 'New query 1' with the SQL command: 'SELECT \* FROM "ticketdata"."bookmark\_parquet\_ticket\_purchase\_history" ORDER BY transaction\_date\_time DESC LIMIT 100;'. Below the query, the results are displayed in a table format with columns: 'id', 'ap', 'sporting\_event\_id', 'bookmarked', 'purchase\_id', 'transaction\_date\_time', 'transaction\_date', 'transaction\_time', 'purchase\_price', and 'partition\_0'. The results show 10 rows of data.

Run the same query against RDS instance, by connecting to your RDS instance through SQL Workbench. (This has been covered in previous Lab, however, we are providing details for your convenience)

In SQL Workbench connection window, provide the following:

- Driver: PostgreSQL (org.postgresql.Driver)
- URL: jdbc:postgresql://<endpoint-rds-instance>:5432/sportstickets
- Username: master
- Password: master123

sporting_event.ticket_id	purchased_by_id	transaction_date_time	transferred_from_id	purchase_price
112156731	3754376	2020-01-26 04:34:37	1271126	67.30
111473601	3754376	2020-01-26 04:34:37	1271126	67.30
111680431	3754376	2020-01-26 04:34:37	1271126	67.30
110356571	3754376	2020-01-26 04:34:37	1271126	67.30
1113941	3754376	2020-01-26 04:34:37	1271126	67.30
112311011	3754376	2020-01-26 04:34:37	1271126	67.30
111766281	3754376	2020-01-26 04:34:37	1271126	67.30
113510301	3754376	2020-01-26 04:34:37	1271126	67.30
113750791	3754376	2020-01-26 04:34:37	1271126	67.30
113823441	3754376	2020-01-26 04:34:37	1271126	67.30
110805431	3754376	2020-01-26 04:34:37	1271126	67.30
110288721	3754376	2020-01-26 04:34:37	1271126	67.30

You will notice that the results for the column “transaction\_date\_time” and value of “purchase\_price” is similar for the last 100 entries in SQL Workbench and in Athena Console. The result matches which shows that the new and latest has been replicated and stored in our table.

Now let's make sure that only the new data was scanned and added to the destination table by our **Glue Job with Bookmark enabled**. To validate the functioning of bookmark, timestamp values are used, in order to make sure only newly added data is scanned and added. In this example, “transaction\_date\_time” is the timestamp value used by Job Bookmark and will be used to showcase the functionality.

Note the “transaction\_date\_time” value from the previous SQL query and then Run the following query:

```
SELECT count(*) FROM
"ticketdata"."bookmark_parquet_ticket_purchase_history"
where "transaction_date_time"='2020-01-26 04:34:37'
group by transaction_date_time
```

“transaction\_date\_time”='2020-01-26 04:34:37' is a subset of the newly added data.

```

Athena Query Editor Saved Queries History Data sources Workgroup: primary

Data source Connect data source
Database
Table/View
Filter tables and views...
Tables (26)
+ bookmark_parquet_ticket_purchase_history (partitioned)
+ vdc_spending_event_ticket
+ vdc_spending_purchase_hist
+ vdc_data
+ name_data
+ rft_data
+ rft_stadium_data
+ parquet_person
+ parquet_sport_location
+ rft_sport
+ parquet_spending_event
+ parquet_spending_event_ticket
+ person
+ player
+ user
+ user_type
+ height_dimension
+ sport_region

```

New query 1 New query 2 New query 3 +  
1. SELECT count(\*) FROM "livedata"."bookmark\_parquet\_ticket\_purchase\_history"  
2. WHERE transaction\_date\_time > '2020-01-01T00:00:00Z'  
3. AND transaction\_date\_time < '2020-01-01T23:59:59Z'  
4. GROUP BY transaction\_date\_time

Run query Save as Create... (Run time: 3.18 seconds, Data scanned: 0.62 KB)  
Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Results

_c00	287839

Since the data we are querying is a subset of newly added data, The result returned by this query should be less than or equal to the difference of record count for table: "bookmark\_parquet\_ticket\_purchase\_history" before and after running Glue Job with bookmark enabled.

In this example:  $(1813493 - 1521922) = 291571 > 287839$ .

This validates that only new data was scanned and added by Glue job with bookmark enabled rather than scanning the entire data from source table and only new data entries were added to the destination table.

*PS: Reference to timestamp is transaction\_date\_time value of the table which is used to show the bookmark functionality.*

When you are building an enterprise use cases, it's become important to automate entire pipeline and add notification. Please refer below blogs to try out end to end servlets datalike automation:

### Build and automate a serverless data lake using an AWS Glue trigger for the Data Catalog and ETL jobs:

<https://aws.amazon.com/blogs/big-data/build-and-automate-a-serverless-data-lake-using-an-aws-glue-trigger-for-the-data-catalog-and-etl-jobs/>

## PART- ( C ): Glue Workflows (Optional)

### Overview:

In AWS Glue, you can use workflows to create and visualize complex extract, transform, and load (ETL) activities involving multiple crawlers, jobs, and triggers. Each workflow manages the execution and monitoring of all its components. As a workflow runs each component, it records execution progress and status, providing you with an overview of the larger task and the details of each step. The AWS Glue console provides a visual representation of a workflow as a graph.

## Creating and Running Workflows:

Above mentioned Part A (ETL with Glue) and Part B (Glue Job Bookmarks) can be created and executed using workflows. Complex ETL jobs involving multiple crawlers and jobs can also be created and executed using workflows in an automated fashion. Below is a simple example to demonstrate how to create and run workflows.

### \*\*Pre-requisite before creating workflow\*\*

To store processed data you need new location. Please Follow this user guide to create folder following structure in your S3 bucket to store parquet file -

<https://docs.aws.amazon.com/AmazonS3/latest/user-guide/create-folder.html>.

Full path will look like e.g. – “s3://dmslab-student-dmslabs3bucket-woti4bf73cw3/tickets/dms\_parquet/mlb\_data/data/”

You need to have **Glue Jobs and Crawlers prebuilt in order to create workflow**. For this exercise, we will select the **table “mlb\_data”** from **“ticketdata” database** and create a glue job and crawler in a similar fashion as you created in Part A of this lab, with these details:

#### a) Glue Job:

- Name: **Glue-Lab-MLBParquet**
- Source: mlb\_data table under ticket database
- Transformation format: Parquet
- Destination: s3://<dms-lab-bucket>/dms\_parquet/mlb\_data/data/

#### b) Glue crawler:

- Name: **glue-lab-mlbdata-crawler**
- Source: <dms-lab-bucket>/dms\_parquet/mlb\_data/data/
- Destination Database: ticketdata
- Table prefix: workflow\_

## To create a workflow:

1. Navigate to **AWS Glue Console** and under **ETL**, click on **Workflows**. Then Click on **Add Workflow**.

The screenshot shows the AWS Glue Workflows console. On the left, there's a sidebar with a tree view: Data catalog, Databases, Tables, Connections, Crawlers, Classifiers, Settings, ETL (which is selected), Workflows (highlighted in orange), Jobs, and ML Transforms. The main content area has a title 'Workflows (0)' and a sub-instruction: 'A workflow is an orchestration used to visualize and manage the relationship and execution of multiple triggers, jobs and crawlers.' Below this is a search bar with 'Add workflow' and 'Actions' buttons, and a 'Filter workflows' input field. A table follows with columns: Name, Last run, Last run status, and Last modified. The table is currently empty, showing 'No workflows'. At the bottom right of the table area is a blue button labeled 'Add a new ETL workflow'.

2. Give the workflow name as **“Workflow\_MLB\_Data”**. Provide a description (optional) and click on **Add Workflow** to create it.

AWS Glue

Workflows > Add workflow

Add a new ETL workflow

Add a workflow in order to orchestrate ETL jobs, triggers, and crawlers

**Workflow name**  
Workflow\_MLB\_Data  
Names may only contain letters (A-Z), numbers (0-9), hyphens (-), or underscores (\_)

**Description (optional)**  
ETL pipeline for MLB Data  
250 characters max

**Default run properties (optional)**  
No default run properties

**Add property**

**Cancel** **Add workflow**

- Click on the **workflow** and scroll to the bottom of the page. You will see an option **Add Trigger**. Click on that button.

Crawlers  
Classifiers  
Settings

ETL

**Workflows**

A workflow is an orchestration used to visualize and manage the relationship and execution of multiple triggers, jobs and crawlers.

**Add workflow** **Actions** **Filter workflows**

Name	Last run	Last run status	Last modified
Workflow_MLB_Data	-	-	Sun, 26 Jan 2020 05:12:03 GMT

**Graph** **Details** **History**

Legend: Start Trigger Job Crawler Incomplete Error Deleting

The workflow is empty  
**Add trigger**

- In **Add Trigger** window, From Clone Existing and Add New options, click on **Add New**.

- Provide **Name** as "trigger1"
- Provide a **description**: Trigger to start workflow
- Trigger type: On-demand.**
- Click on **Add**

Triggers are used to initiate the workflow and there are multiple ways to invoke the trigger. Any scheduled operation or any event can activate the trigger which in turn starts the workflow.

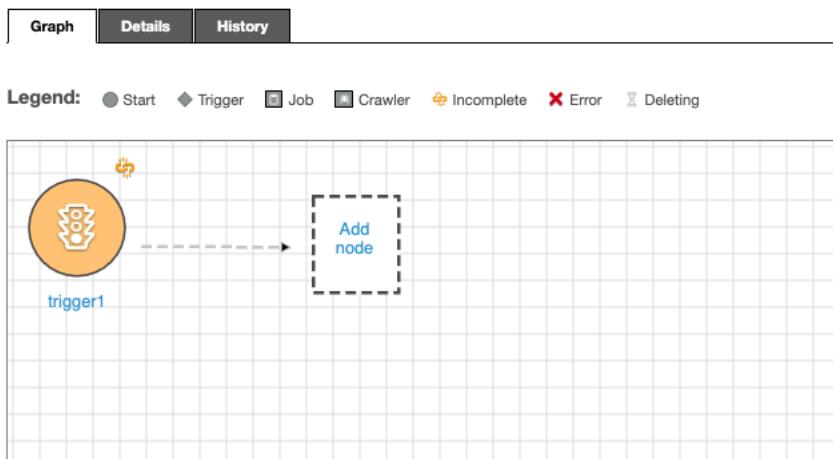
Add trigger

Name  
trigger1

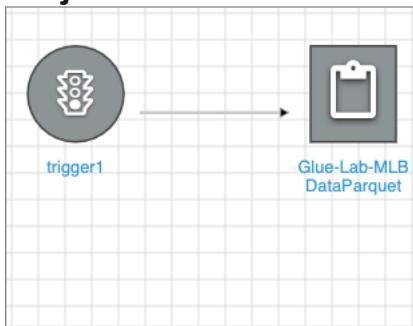
Description (optional)  
Trigger to start the workflow

Trigger type  
 Schedule  Event  On demand

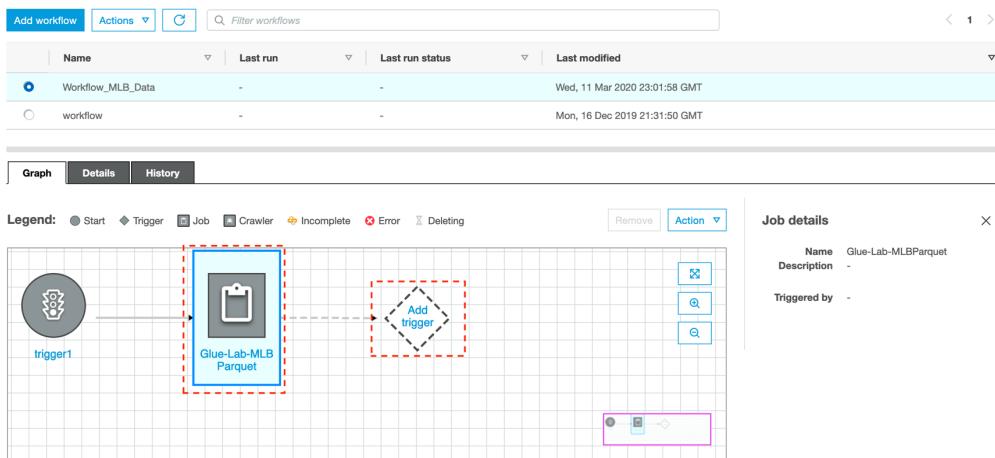
- Click on **trigger1** to add a **new node**. New Node can be a crawler or job, depending upon the workflow you want to build. Make sure you add node only after clicking on trigger so that whenever the “trigger1” is **activated**, it triggers job/crawler specified in the new node.



- Click on “Add Node”. A new window to add jobs or crawlers will open. Select the job **Glue-Lab-MLBDataParquet**



7. Click on **Glue-Lab-MLBDataParquet** and **Add Trigger** will appear as shown in below screen.

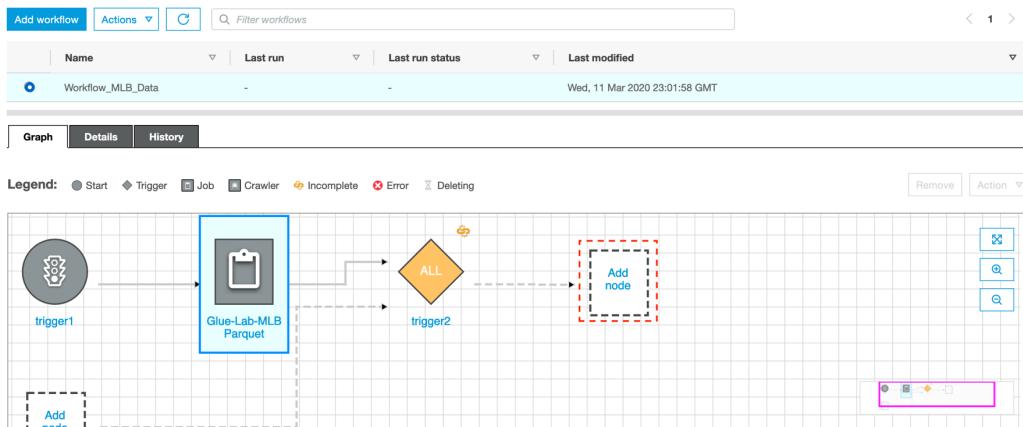


8. Click on **Add trigger**, provide the following:
- Name: trigger2**
  - Description: Trigger to execute crawler**
  - Trigger type: Event**
  - Trigger logic: Start after ALL watched event.** This will make sure that crawler starts once Glue job finishes processing of ALL data. Click **Add**

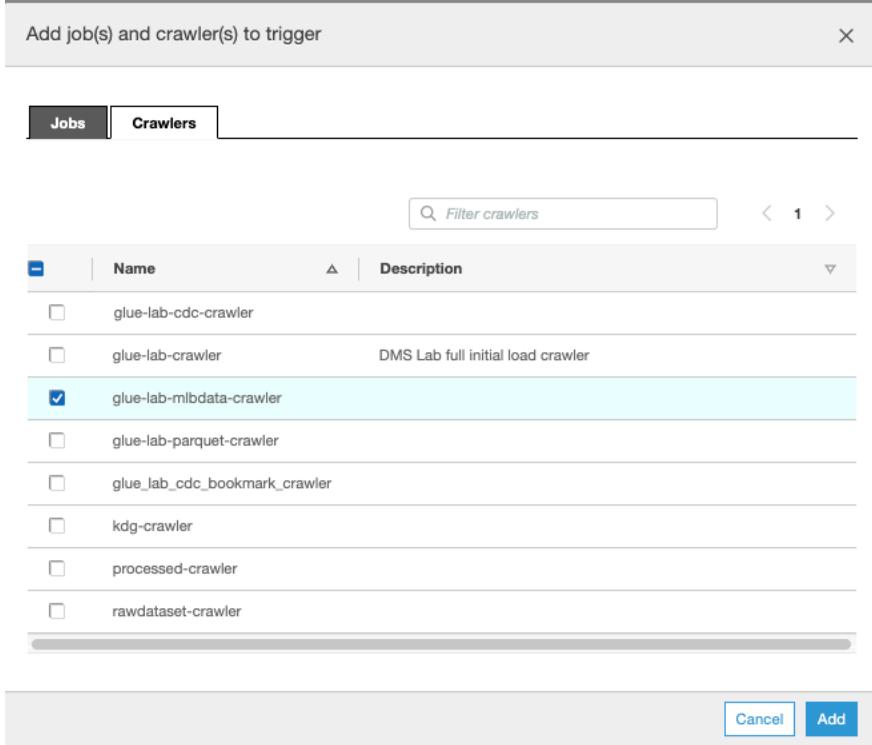
The screenshot shows the 'Add trigger' dialog box. At the top, there are two buttons: 'Clone existing' and 'Add new' (which is selected). Below that is a 'Name' field containing 'trigger2'. Underneath is a 'Description (optional)' field with the text 'Trigger to execute crawler'. The 'Trigger type' section has three radio buttons: 'Schedule' (unchecked), 'Event' (checked), and 'On demand' (unchecked). The 'Trigger logic' section has two radio buttons: 'Start after ANY watched event' (unchecked) and 'Start after ALL watched event' (checked). At the bottom right are 'Cancel' and 'Add' buttons.

9. After **trigger2** is added to workflow, Click on **Add node** which is connected to **trigger 2** as highlight below:

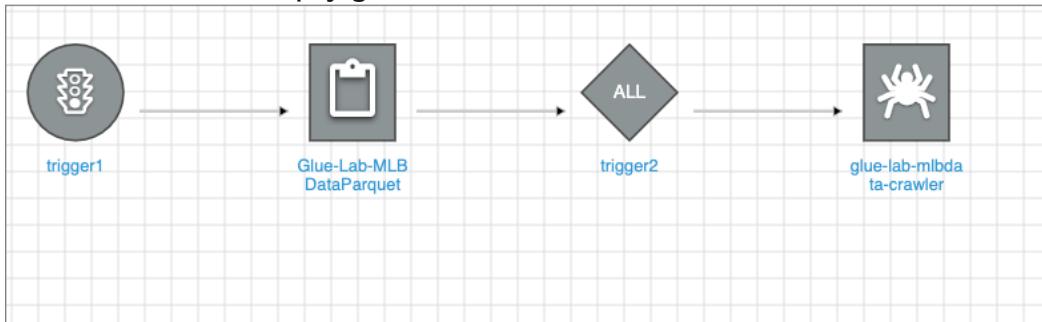
**Workflows (2)**  
A workflow is an orchestration used to visualize and manage the relationship and execution of multiple triggers, jobs and crawlers.



10. Select **crawler** option and then chose “**glue-lab-mlbdata-crawler**”. Click **Add**.



11. Now click on an empty grid and workflow will look like below:



12. Select your workflow, click on **Actions->Run** and this will start the first trigger "trigger1"

13. Now the first node will be executed, i.e. Glue Job and subsequently all other nodes will be executed.

14. Once finished, the workflow will be shown as **completed**.

15. You can click on any node at any time of processing of workflow, to get more details about that particular stage of processing. This gives you detailed level view for monitoring your pipeline.



16. Once the workflow is completed, you will observe that glue job and crawlers have been successfully executed and the table has been created.

**Glue Job**

Type	ETL language	Script location	Last modified	Job bookmark
Spark	python	s3://aws-glue-scripts-66953142058-ue...	26 January 2020 10:21 PM UTC-5	Disable
Spark	python	s3://aws-glue-scripts-66953142058-ue...	10 January 2020 3:19 PM UTC-5	Disable
Spark	python	s3://aws-glue-scripts-66953142058-ue...	10 January 2020 3:19 PM UTC-5	Disable
Spark	python	s3://aws-glue-scripts-66953142058-ue...	10 January 2020 2:24 PM UTC-5	Disable
Spark	python	s3://aws-glue-scripts-66953142058-ue...	10 January 2020 2:24 PM UTC-5	Disable
Spark	python	s3://aws-glue-scripts-66953142058-ue...	10 January 2020 5:40 PM UTC-5	Disable
Spark	python	s3://aws-glue-scripts-66953142058-ue...	24 January 2020 6:51 PM UTC-5	Enable
Spark	python	s3://aws-glue-scripts-66953142058-ue...	24 January 2020 6:51 PM UTC-5	Enable
clickstream-data				
sin-data				

**History**

User ID	Retry attempt	Run status	Error	Logs	Error logs	Glue version	Maximum capacity	Triggered by	Start time	End time	Execution time	Timeout	Delay	Job run input
0_26130610561017742946076100...		Succeeded		Logs		1.0	10	trigger1	25 January 2020 10:21 PM UTC-5	25 January 2020 10:21 PM UTC-5	40 secs	2860 mins		s3://aws-glue-temporary-66953142058-ue...

**Glue Crawler**

Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
glue-lab-cdc-crawler		Ready	Logs	1 min	1 min	2	0
gl-lab-crawler		Ready	Logs	1 min	1 min	0	15
glue-lab-mbdata-crawler		Ready	Logs	1 min	1 min	0	1
glue-lab-parquet-crawler		Ready	Logs	57 secs	57 secs	0	5
glue-lab_cdc_bookmark_crawler		Ready	Logs	1 min	1 min	1	0
kdp-crawler		Ready	Logs	1 min	1 min	0	1

AWS Glue

Data catalog

Databases

- Table
  - bookman\_parquet\_ticket\_purchase\_history
  - cic\_spending\_event\_ticket
  - cic\_ticket\_purchase\_hist
  - mlb\_data
  - name\_data
  - rf\_data
  - rf\_stadium\_data
  - parquet\_person
  - parquet\_sport\_location
  - parquet\_sport\_team
  - parquet\_sporting\_event
  - parquet\_sporting\_event\_ticket
  - parquet\_ticket\_purchase\_hist
  - seat
  - seat\_type
  - sport\_division
  - sport\_league
  - sport\_location
  - sport\_team
  - sporting\_event
  - sporting\_event\_ticket
  - solo\_purchase\_hist
  - workflow\_mlb\_data

Workflows

Jobs

ML Transforms

Triggers

Dev endpoints

Notebooks

Security

Security configurations

Tutorials

Add crawler

Deploy crawler

Add job

Resources

New

**Table Created**

17. Let's query the data in Athena. Select “**workflow\_mlb\_data**” table and click on Actions->View Data.

New query 1 (Run time: 2.92 seconds, Data scanned: 336.53 KB)

1. SELECT \* FROM "ticketeddata"."workflow\_mlb\_data" limit 10;

**Run query** **Save as** **Create** (Format query Clear)

Results

#	mlb_id	mlb_name	mlb_pos	mlb_team	mlb_team_long	bats	throws	birth_year	bg_id	brf_id	brf_name	cbs_id	cbs_name	cls_pos	espn_id	espn_name	espn_pos	fg_id	fg_name	lahman_id	mlbc_id	rfbc_nu
1	+5.0656e+000000000000e+05	Alexi Amarista	3B	SD	San Diego Padres	L	R	1989	+5.58860000000000e+04	amarista01	Alexi Amarista	1735053	Alexi Amarista	2B	+3.08620000000000e+04	Alexi Amarista	2B	9063	Alexi Amarista	amarista01	+8.91400000000000e+03	Alexi Amarista
2	+4.58210000000000e+05	Alexi Casilla	2B	TB	Tampa Bay Rays	S	R	1984	+4.57990000000000e+04	casillal01	Alexi Casilla	1103724	Alexi Casilla	3B	+2.85750000000000e+04	Alexi Casilla	3B	5248	Alexi Casilla	casillal01	+7.85500000000000e+03	Alexi Casilla
3	+4.68390000000000e+05	Alexi Ogando	P	ATL	Atlanta Braves	R	R	1983	+4.99100000000000e+04	ogandal01	Alexi Ogando	1174296	Alexi Ogando	RP	+3.05490000000000e+04	Alexi Ogando	RP	10261	Alexi Ogando	ogandal01	+8.74300000000000e+03	Alexi Ogando
4	+4.69860000000000e+05	Alfredo Aceves	P	NYY	New York Yankees	R	R	1982	+4.69260000000000e+04	acevalo1	Alfredo Aceves	1639890	Alfredo Aceves	RP	+2.62230000000000e+04	Alfredo Aceves	RP	5164	Alfredo Aceves	acevalo1	+8.36000000000000e+03	Alfredo Aceves
5	+5.16260000000000e+05	Alfredo Figaro	P	TEX	Texas Rangers	R	R	1984	+5.24530000000000e+04	figaral01	Alfredo Figaro	1654334	Alfredo Figaro	RP	+3.00670000000000e+04	Alfredo Figaro	RP	8404	Alfredo Figaro	figaral01	+8.51300000000000e+03	Alfredo Figaro
6	+5.54540000000000e+05	Alfredo Gonzalez	C	CWS	Chicago White Sox	R	R	1992	+6.82010000000000e+04	gonzale01	Alfredo Gonzalez	221083	Alfredo Gonzalez	C	+3.48440000000000e+04	Alfredo Gonzalez	C	sa605301	Alfredo Gonzalez	gonzale01	+8.51300000000000e+03	Alfredo Gonzalez
7	+5.01245000000000e+05	Alfredo Marte	LF	BAL	Baltimore Orioles	R	R	1989	+5.17550000000000e+04	marteal01	Alfredo Marte	1959711	Alfredo Marte	LF	+3.22500000000000e+04	Alfredo Marte	LF	5212	Alfredo Marte	marteal01	+9.35400000000000e+03	Alfredo Marte
8	+4.30580000000000e+05	Alfredo Simon	P	CIN	Cincinnati Reds	R	R	1981	+4.55810000000000e+04	simon01	Alfredo Simon	448668	Alfredo Simon	SP	+2.88900000000000e+04	Alfredo Simon	SP	2155	Alfredo Simon	simon01	+7.97500000000000e+03	Alfredo Simon
9	+4.55378000000000e+05	All Solis	C	LAD	Los Angeles Dodgers	R	R	1987	+5.23620000000000e+04	solisal01	All Solis	1946524	All Solis	C	+3.21140000000000e+04	All Solis	C	8848	All Solis	solisal01	+9.34000000000000e+03	All Solis
10	+4.88852000000000e+05	Allan Dykstra	1B	TB	Tampa Bay Rays	L	R	1987	+5.78130000000000e+04	dykstral01	Allan Dykstra	1960245	Allan Dykstra	1B	+3.28610000000000e+04	Allan Dykstra	1B	9113	Allan Dykstra	dykstral01	+9.82000000000000e+03	Allan Dykstra

Congratulations!! You have successfully completed this lab