



# Amazon Web Services Data Engineering Immersion Day

---

Lab 2. ETL with AWS Glue

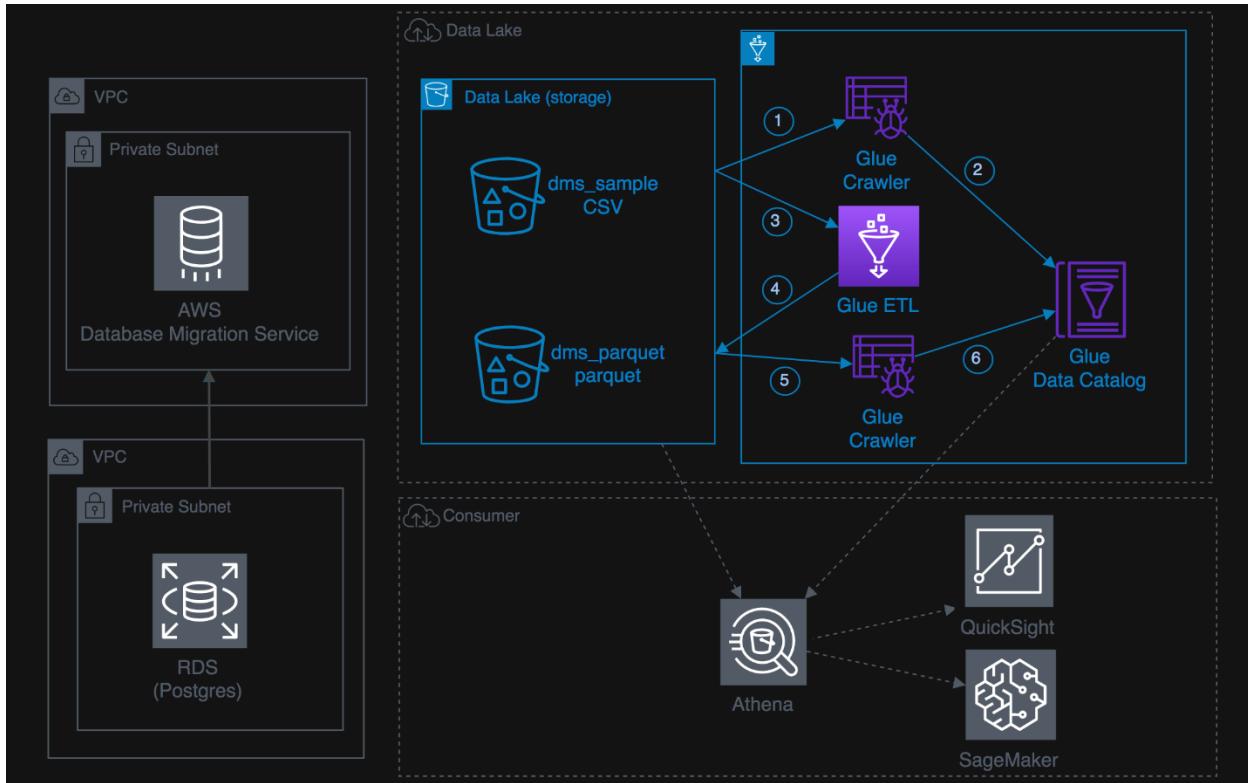
*August 2020*

## Table of Contents

<i>Introduction</i> .....	2
<i>Get Started Using the Lab Environment</i> .....	3
<i>PART A: Data Validation and ETL</i> .....	6
Create Glue Crawler for initial full load data .....	6
Data Validation Exercise .....	10
Data ETL Exercise .....	11
Create Glue Crawler for Parquet Files.....	16
<i>PART B: Glue Job Bookmark (Optional):</i> .....	20
Step 1: Create Glue Crawler for ongoing replication (CDC Data).....	20
Step 2: Create a Glue Job with Bookmark Enabled .....	24
Step 3: Create Glue crawler for Parquet data in S3.....	27
Step 4: Generate CDC data and to observe bookmark functionality.....	30
<i>PART C: Glue Workflows (Optional, self-paced)</i> .....	31
Overview: .....	31
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

1. Completed Lab 1. Hydrating the Data Lake with DMS

### 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](#)
2. [PART- \(B\): Glue Job Bookmark Functionality\(Optional\)](#)
3. [PART- \( C \): Glue Workflows\(Optional\)](#)

The Lab is also available - <https://aws-dataengineering-day.workshop.aws/>

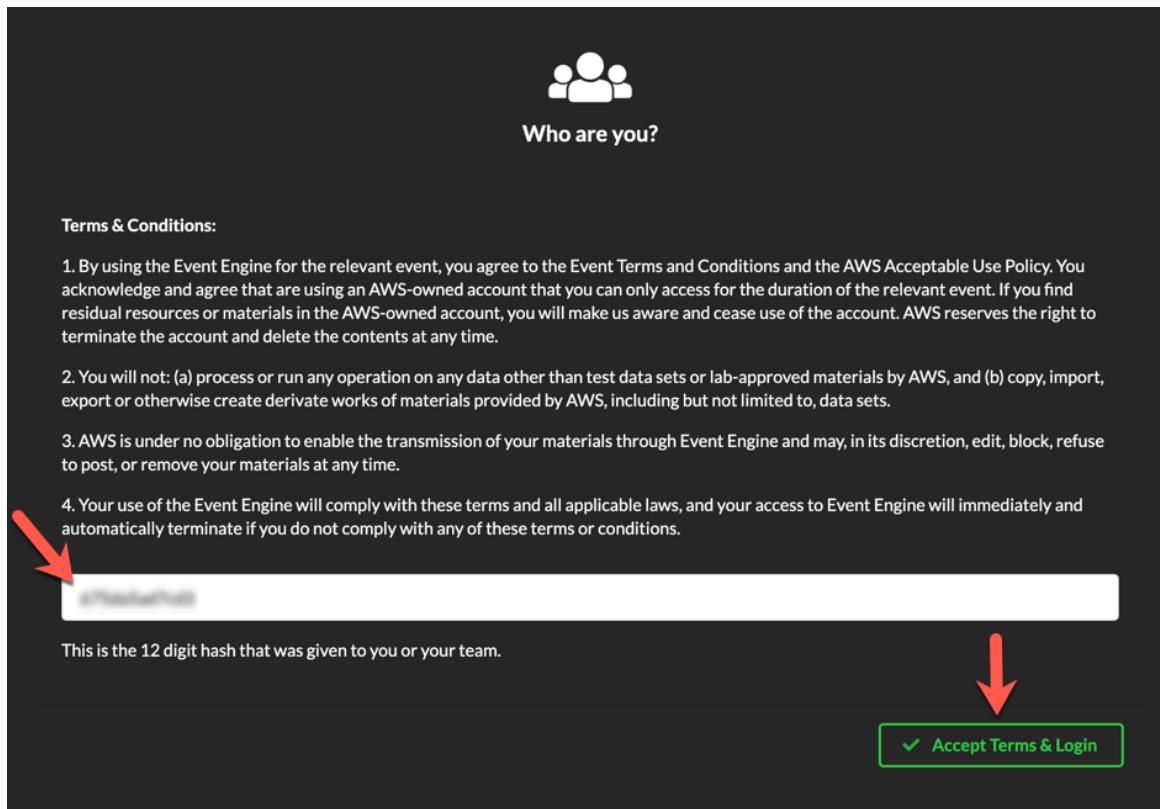
## Get Started Using the Lab Environment

Please skip this section if you are running the lab on your own AWS account.

Today, you are attending a formal event and you will have been sent your access details beforehand. If in the future you might want to perform these labs in your own AWS environment by yourself, you can follow instructions on GitHub - <https://github.com/aws-samples/data-engineering-for-aws-immersion-day>.

A 12-character access code (or 'hash') is the access code that grants you permission to use a dedicated AWS account for the purposes of this workshop.

1. Go to <https://dashboard.eventengine.run/>, enter the access code and click Proceed:



2. On the Team Dashboard web page you will see a set of parameters that you will need during the labs. Best to save them to a text file locally, alternatively you can always go to this page to review them. Replace the parameters with the corresponding values from here where indicated in subsequent labs:

## Lab 2. ETL with AWS Glue

Because you're at a formal event, some AWS resources have been pre-deployed for your convenience, for example:

- The source database connection in RDS DB Info module

RDS DB Info

Outputs:  
No outputs defined

Readme

- S3 Bucket, IAM role for the Glue lab etc

Environment Setup

Outputs:  
**S3 Bucket name**  
mod-3fccddd609114925-dmslabs3bucket-1ngcgzznd15u  
**BusinessAnalystUser**  
mod-3fccddd609114925-BusinessAnalystUser-MB0XFZLQLOXX  
**DMSLabRoleS3 ARN**  
arn:aws:iam::377243295828:role/mod-3fccddd609114925-DMSLabRoleS3-O2VT1RSN43SG  
**Glue Lab Role**  
mod-3fccddd609114925-GlueLabRole-YLTJA13WW6WT  
**S3BucketWorkgroupA**  
mod-3fccddd609114925-s3bucketnetworkgroupa-tbon3m1mkunh  
**S3BucketWorkgroupB**  
mod-3fccddd609114925-s3bucketnetworkgroupb-18ygl8nfp8ead  
**WorkgroupManagerUser**  
mod-3fccddd609114925-WorkgroupManagerUser-5IVE0UQNIBG4

Readme

3. On the Team Dashboard, please click AWS Console to log into the AWS Management Console:

Team Dashboard

Event

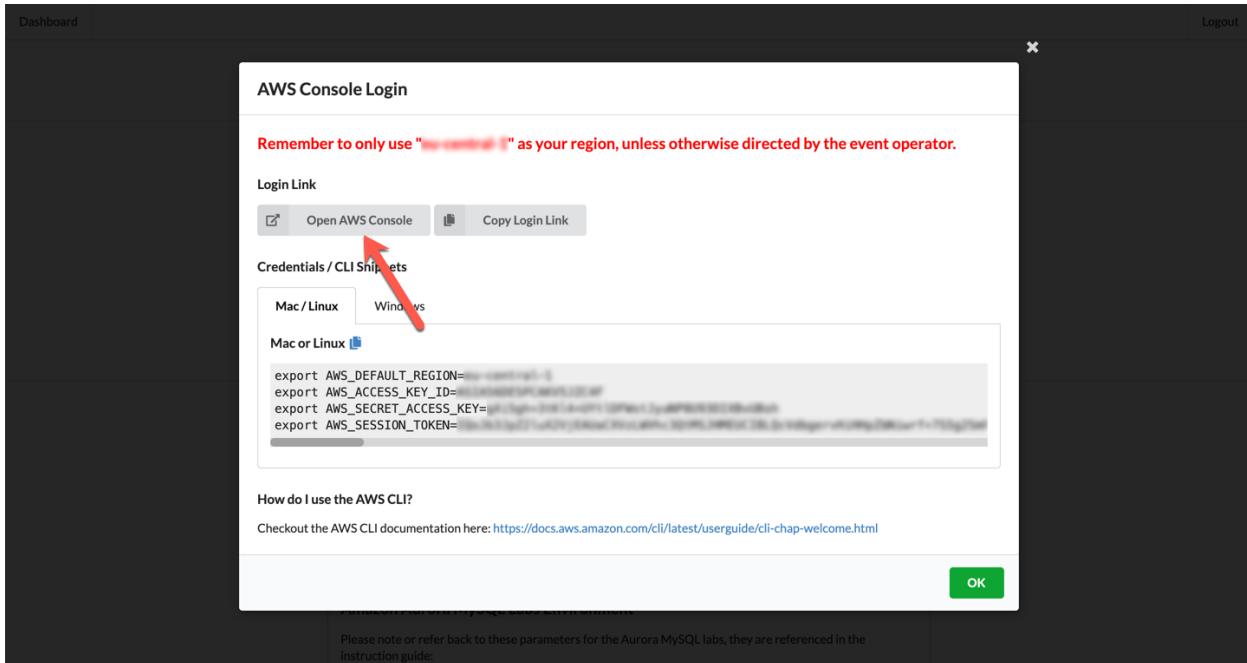
AWS Console    SSH Key

Event: Data Engineering Immersion Day - Test  
Team Name:

Event ID: d2302d4ae9ff4ea2857846b74f7de7e2  
Team ID: 1c2f7ad7ec044b0b8276f917c5983133

## Lab 2. ETL with AWS Glue

4. Click Open Console. For the purposes of this workshop, you will not need to use command line and API access credentials:



Once you have completed these steps, you can continue with the rest of this lab.

### PART A: Data Validation and ETL

Create Glue Crawler for initial full load data

1. Navigate to the [AWS Glue service](#)

The screenshot shows the AWS Services search interface. The search bar at the top contains the text "glue". Below the search bar, the results section displays two items: "AWS Glue" and "AWS Lake Formation". The "AWS Glue" item is described as "AWS Glue is a fully managed ETL (extract, transform, and load) service". There are also links for "S3" and "EC2" below the search bar.

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

The screenshot shows the "Crawlers" page under the AWS Glue service. The left sidebar has a navigation menu with "Crawlers" selected. The main area displays a table with columns: Name, Schedule, Status, Logs, Last runtime, Median runtime, Tables updated, and Tables added. A message states "You don't have any crawlers yet." and there is a prominent blue "Add crawler" button.

3. Click **Add crawler**.
4. Enter **glue-lab-crawler** as the crawler name for initial data load.
5. Optionally, enter the description. This should also be descriptive and easily recognized and Click **Next**.

The screenshot shows the "Add crawler" step 1: Crawler info. On the left, a sidebar lists options: Crawler info (selected), Crawler source type, Data store, IAM Role, Schedule, Output, and Review all steps. The main area has a title "Add information about your crawler" and a "Crawler name" field containing "glue-lab-crawler". Below the field is a note: "Tags, description, security configuration, and classifiers (optional)". At the bottom right is a blue "Next" button.

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

The screenshot shows the "Add crawler" step 2: Crawler source type. The left sidebar shows "Crawler info" is selected. The main area has a title "Specify crawler source type" and a note: "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." It shows a "Crawler source type" section with "Data stores" selected (radio button is blue). At the bottom are "Back" and "Next" buttons.

7. On the **Add a data store** page, make the following selections:

## Lab 2. ETL with AWS Glue

- 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 for your DMS initial export from Lab 1, e.g., **s3://dmslab-student-dmslabs3bucket-wotl4bf73cw3/tickets**
8. Click **Next**.

Add crawler

Choose a data store

Crawl data in

Include path

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

Exclude patterns (optional)

Back Next

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

Add crawler

Add another data store

Chosen data stores

Yes No

Back Next

10. 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>** pre-created for you. For example "dmslab-student-GlueLabRole-ZOQDII7JTBUM"
11. 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-ZOQDII7JTBUM

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

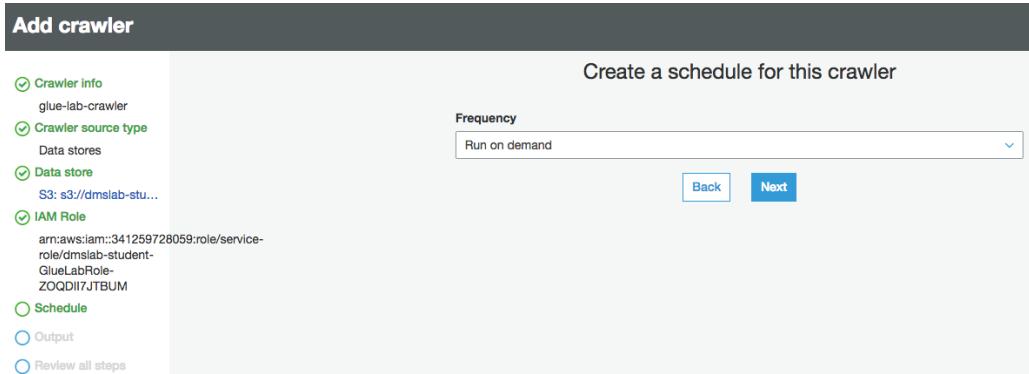
s3://dmslab-student-dmslabs3bucket-wotl4bf73cw3

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

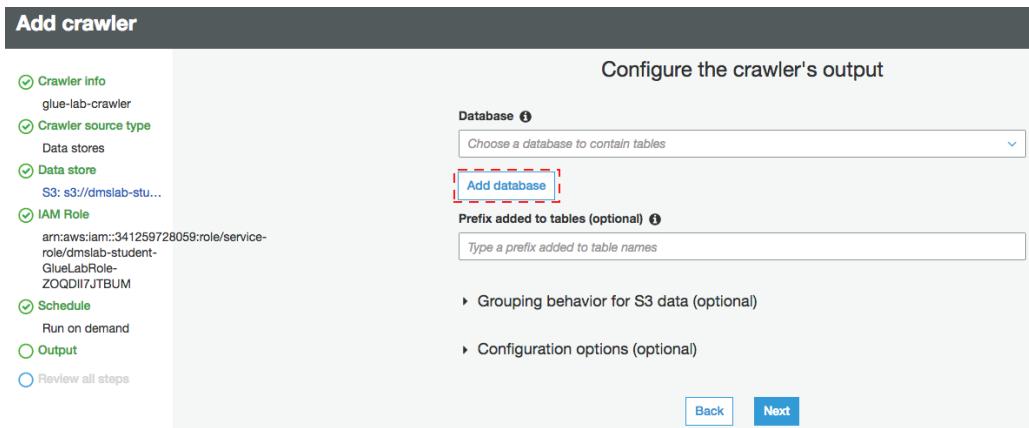
Back Next

## Lab 2. ETL with AWS Glue

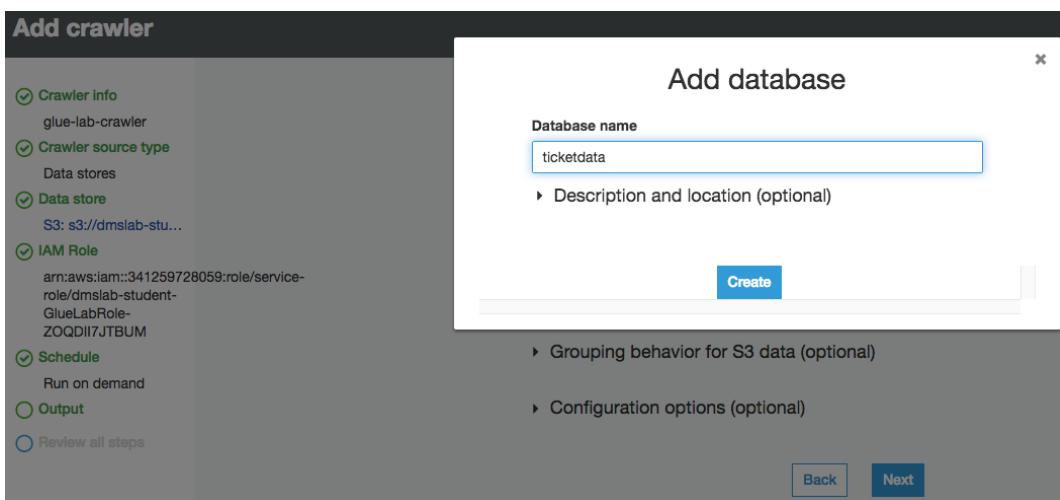
12. On the Create a schedule for this crawler page, for Frequency, select **Run on demand** and Click **Next**.



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



14. Enter **ticketdata** as your database name and click **create**



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

## Lab 2. ETL with AWS Glue

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

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

18. Click **Run it now**.

## Lab 2. ETL with AWS Glue

Crawler will change status from starting to stopping, wait until crawler comes back to ready state (the process will take a few minutes), you can see that it has created 15 tables.

The screenshot shows the AWS Glue Crawler list page. A message at the top states: "Crawler 'glue-lab-crawler' completed and made the following changes: 15 tables created, 0 tables updated. See the tables created in database ticketdata." Below this, there is a table with columns: Name, Schedule, Catalog type, Status, Logs, Last runtime, Median runtime, Tables updated, and Tables added. One row is visible for the crawler, showing it is in a Ready state with 1 min runtime, 0 updated tables, and 15 added tables.

19. In the AWS Glue navigation pane, click **Databases > Tables**. You can also click the **ticketdata** database to browse the tables.

The screenshot shows the AWS Glue Tables list page for the 'ticketdata' database. A message at the top says: "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." Below this, there is a table with columns: Name, Database, Location, Classification, Last updated, and Deprecated. 15 tables are listed, including 'mlb\_data', 'name\_data', 'nfl\_data', 'nfl\_stadium\_data', 'person', 'player', 'seat', 'seat\_type', 'sport\_division', 'sport\_league', 'sport\_location', 'sport\_team', 'sporting\_event', 'sporting\_event\_ticket', and 'ticket\_purchase\_hist'.

## Data Validation Exercise

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

The screenshot shows the AWS Glue Tables list page for the 'ticketdata' database. A message at the top says: "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." Below this, there is a table with columns: Name, Database, Location, Classification, Last updated, and Deprecated. The 'person' table is highlighted with a red box. Other tables listed include 'mlb\_data', 'name\_data', 'nfl\_data', 'nfl\_stadium\_data', 'player', 'seat', 'seat\_type', 'sport\_division', 'sport\_league', and 'sporting\_event\_ticket'.

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.

## Lab 2. ETL with AWS Glue

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

The screenshot shows the AWS Glue Table Editor interface for a table named 'person'. The 'Edit schema' button is highlighted with a red box. The table properties are listed, including the location: `s3://dmstutorial-dmlab-tutorial-xg1hdyof0bs/tickets/dms_sample/person/`. The schema section shows four columns: col0, col1, col2, and col3, all of type string. A yellow circle highlights the 'Column name' column header.

Column name	Data type	Partition key	Comment
1 col0	string		
2 col1	string		
3 col2	string		
4 col3	string		

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

The screenshot shows the 'Edit schema' section with the column names changed. The first column, which was originally 'col0', is now circled in yellow and has been renamed to 'id'. The other three columns have also been renamed to 'full\_name', 'last\_name', and 'first\_name' respectively. The 'Save' button is highlighted with a blue box.

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

- Click **Save**.

### Data ETL Exercise

**Pre-requisite:** To store processed data in parquet format, we need a new folder location for each table, eg. the full path for sport\_team table look like this –

`"s3://<s3_bucket_name>/tickets/dms_parquet/sport_team"`

Glue will create the new folder automatically, based on your input of the full file path, such as the example above. Please refer to the [user guide](#) in terms of how to manually create a folder in S3 bucket.

## Lab 2. ETL with AWS Glue

1. Go to the Glue console, in the left navigation pane, under **ETL** click **Jobs**, and then click **Add job**.

The screenshot shows the AWS Glue Jobs page. On the left, there's a navigation sidebar with sections for Data catalog, Databases, Tables, Connections, Crawlers, Classifiers, ETL, Jobs (which is selected and highlighted in orange), Triggers, and Dev endpoints. The main area has a title "Jobs" with a sub-instruction: "A job is your business logic required to perform extract, transform and load (ETL) work. Job runs are initiated by triggers which can be scheduled or driven by events." Below this is a search bar with "Add job" and "Action" buttons, and a "Filter by attributes" search field. To the right, it says "Showing: 0 - 0" and has a "User preferences" link. A table header row includes columns for Name, ETL language, Script location, Last modified, and Job bookmark. A message below the table states "You don't have any jobs defined yet." and features a blue "Add job" button.

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. "xxxx-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**, type **Glue-Lab-SportTeamParquet**.
  - 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.)

The screenshot shows the "Add job" configuration dialog box. On the left, a sidebar lists "Job properties", "Data source", "Transform type", "Data target", and "Libraries". The main area is titled "Configure the job properties". It contains the following fields:

- Name:** Glue-Lab-SportTeamParquet
- IAM role:** dmlab-student-GlueJobRole-14RWFBIWGZAMB
- Type:** Spark
- Glue version:** Spark 2.4, Python 3 (Glue version 1.0)
- This job runs:** A proposed script generated by AWS Glue (radio button selected)
- Script file name:** Glue-Lab-SportTeamParquet
- S3 path where the script is stored:** s3://aws-glue-scripts-66953140268-us-east-1/demo\_user
- Temporary directory:** s3://aws-glue-temporary-66953140268-us-east-1/demo\_user

At the bottom, there are several expandable sections: "Advanced properties", "Monitoring options", "Tags (optional)", "Security configuration, script libraries, and job parameters (optional)", and "Catalog options (optional)". A "Next" button is located at the very bottom right.

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

## Lab 2. ETL with AWS Glue

**Add job**

Choose a data source

Name	Database	Location	Classification
sport_team	ticketdata	s3://dmslab-student-dmslabs3bucket-wol4bf73ow3/tick...	csv
sportickets_dms_sample_sport_team	onprem-db	sportickets.dms_sample.sport_team	postgresql

Showing: 1 - 2

Back Next

- 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

- On the Choose your data targets page, select **Create tables in your data target**.
- For Data store, select **Amazon S3**.
- For Format, select **Parquet**.
- For Target path, click the folder icon and choose the s3 bucket, then append **/tickets/dms\_parquet/sport\_team** to it, making the target path look like **s3://xxx-dmslabs3bucket-xxx/tickets/dms\_parquet/sport\_team** – Glue will create necessary folders
- Click **Next**.

**Add job**

Choose a data target

Create tables in your data target  
 Use tables in the data catalog and update your data target

Data store: Amazon S3

Format: Parquet

Target path: s3://dmslab-student-dmslabs3bucket-xg1hdq60bs/ticks

Back Next

## Lab 2. ETL with AWS Glue

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

The screenshot shows the AWS Glue 'Add job' interface. On the left, under 'Job properties', there is a tree view with nodes like 'Glue-Lab-SportTeamParquet', 'Data source', 'sport\_team', 'Transform type', 'Change schema', 'Data target', 's3://dmslab-studen...', and 'Schema'. The 'Schema' node is expanded. In the center, there is a table titled 'Source' with columns 'Column name', 'Data type', and 'Map to target'. One row shows 'id' with 'string' as the data type and 'id' as the map to target. A modal window titled 'String type' is open over this table, focusing on the 'Column type' dropdown which has 'double' selected. At the bottom of the modal is a blue 'Update' button. To the right of the table is a preview of the target schema with columns 'name', 'Data type', and 'Map to target'. The 'id' column is mapped to 'double'. At the bottom right of the main interface are buttons for 'Add column', 'Clear', and 'Reset'.

**Map the source columns to target columns.**

The 'Map the source columns to target columns' section shows a grid where source columns are mapped to target columns. The source columns are 'id', 'name', 'abbreviated\_name', 'home\_field\_id', 'sport\_type\_name', 'sport\_league\_short\_name', and 'sport\_division\_short\_name'. The target columns are 'id', 'name', 'abbreviated\_name', 'home\_field\_id', 'sport\_type\_name', 'sport\_league\_short\_name', and 'sport\_division\_short\_name'. Arrows indicate the mapping from source to target. Below this grid are buttons for 'Back' and 'Save job and edit script'.

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. At the top, it says 'Job: Glue-Lab-SportTeamParquet' with buttons for 'Action' (dropdown), 'Save' (highlighted in red), 'Run job', 'Generate diagram', and a help icon. To the right are buttons for 'Insert template at cursor' (dropdown), 'Source', 'Target', 'Target Location', 'Transform', 'Spigot', and a help icon. The main area shows a job flow with four steps:

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

Below the steps is a large code editor containing the generated Python script:

```

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 ## Options: [JOB_NAME]
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 spark = glueContext.spark_session
14 job = Job(glueContext)
15 job.init(args['JOB_NAME'], args)
16 ## Args: DataSource
17 ## Bings: [database = "ticketdata", table_name = "sport_team", transformation_ctx = "datasource0"]
18 ## Inputs: []
19 ## Outputs: []
20 datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "ticketdata", table_name = "sport_team", transformation_ctx = "datasource0")
21 ## Bings: [ApplyMapping, choice = "make_struct"]
22 ## Inputs: [choice = "make_struct", transformation_ctx = "applymapping0"]
23 ## Outputs: [choice = "make_struct"]
24 ## Bings: [ApplyMapping]
25 ## Inputs: [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"}], transformation_ctx = "applymapping0"]
26 ## Bings: [ResolveChoice, transformation_ctx = "resolvechoice0"]
27 ## Inputs: [choice = "make_struct", transformation_ctx = "resolvechoice0"]
28 ## Outputs: [choice = "make_struct"]
29 ## Bings: [DropNullFields]
30 ## Inputs: [frame = applymapping0, choice = "make_struct", transformation_ctx = "resolvechoice0"]
31 ## Outputs: [DropNullFields]
32 ## Bings: [DropNullFields]
33 ## Inputs: [dropnullfields3]
34 ## Outputs: [dropnullfields3]

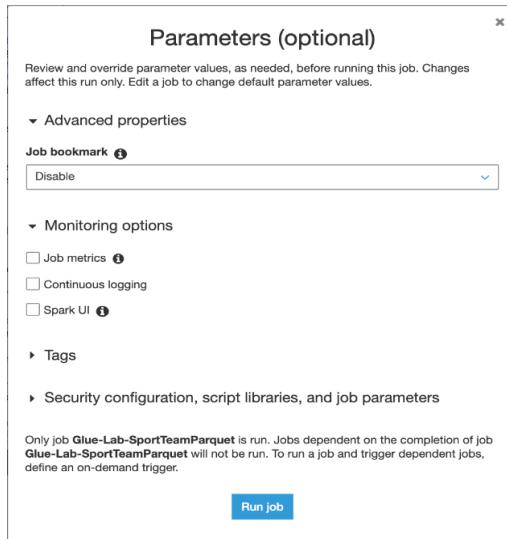
```

At the bottom of the code editor are tabs for 'Logs' and 'Schema'.

14. In **Parameters** option,

- a. 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.

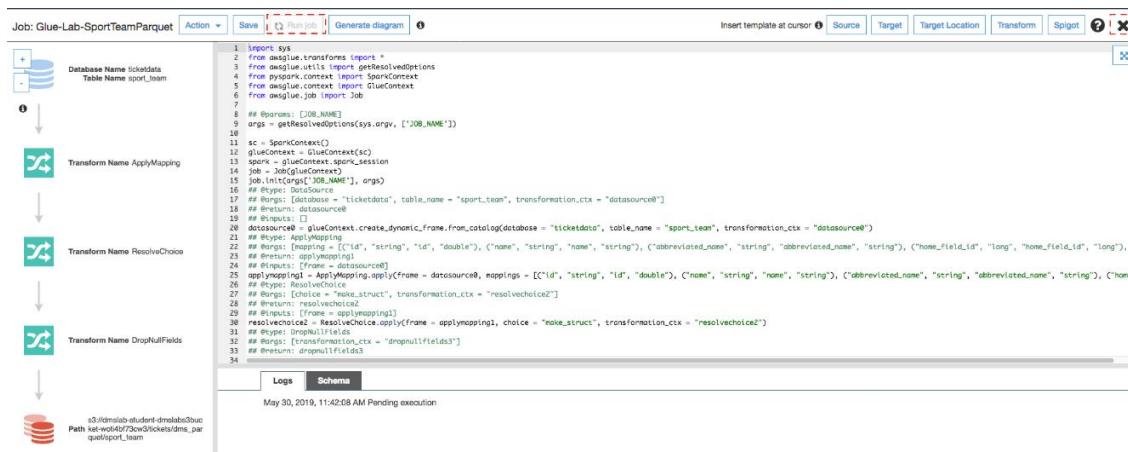
## Lab 2. ETL with AWS Glue



- b. 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 became greyed out. 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.

User preferences														
Add job	Action	Filter by tags and attributes	Showing: 1 - 11											
Name	Type	ETL language	Script location	Last modified	Job bookmark									
Glue-Lab-SportTeamParquet	Spark	python	s3://aws-glue-scri...	11 March 2020 3:17 PM UTC-7	Disable									
<a href="#">History</a> <a href="#">Details</a> <a href="#">Script</a> <a href="#">Metrics</a>														
View run metrics <a href="#">Rewind job bookmark</a>														
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
jr_e37b56aa03cd3...	-	Running	Logs	Error logs		1.0	5		11 Mar...	0 secs	2880 mins			s3://aws-glue-tem...

## Lab 2. ETL with AWS Glue

If you plan to continue with the [Lab 3. Consuming data with Athena and Quicksight](#), you'll have to complete the rest of this section to create more ETL Jobs to transform additional tables to parquet, changing their schema as per instructions below. Otherwise you can proceed to the section "[Create Glue Crawler for Parquet Files](#)"

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

**Table 1:**

Job Name & Script Filename	Source Table	S3 Target Path	Need Schema Change?
Glue-Lab-SportLocationParquet	sport_location	tickets/dms_parquet/sport_location	No
Glue-Lab-SportingEventParquet	sporting_event	tickets/dms_parquet/sporting_event	Yes
Glue-Lab-SportingEventTicketParquet	sporting_event_ticket	tickets/dms_parquet/sporting_event_ticket	Yes
Glue-Lab-PersonParquet	person	tickets/dms_parquet/person	Yes

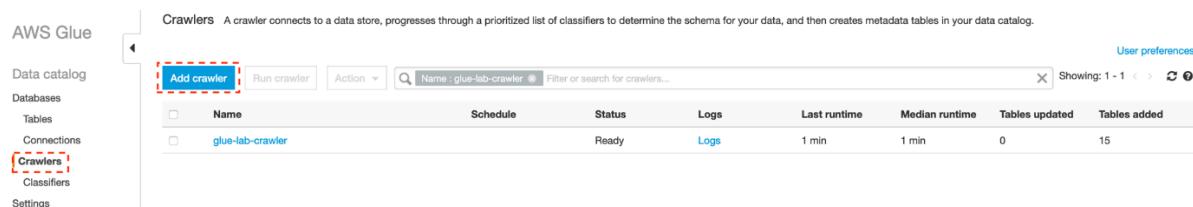
**Table 2:**

Job Name	Table	Column	Source Data Type	Target Data Type
Glue-Lab-SportingEventParquet	sporting_event	start_date_time	STRING	TIMESTAMP
Glue-Lab-SportingEventParquet	sporting_event	start_date	STRING	DATE
Glue-Lab-SportingEventTicketParquet	sporting_event_ticket	id	STRING	DOUBLE
Glue-Lab-SportingEventTicketParquet	sporting_event_ticket	sporting_event_id	STRING	DOUBLE
Glue-Lab-SportingEventTicketParquet	sporting_event_ticket	ticketholder_id	STRING	DOUBLE
Glue-Lab-PersonParquet	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**.



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

## Lab 2. ETL with AWS Glue

Add crawler

Add information about your crawler

Crawler name: glue-lab-parquet-crawler

Crawler source type: Data stores

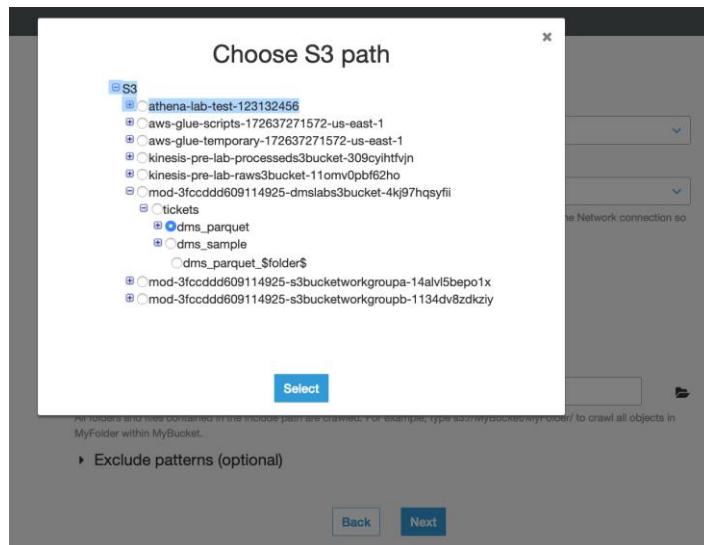
S3: s3://dmslab-stu...

IAM Role

Schedule

Next

3. In next screen **Specify crawler source type**, select **Data Stores** as choice for **Crawler source 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 nested parquet files e.g., `s3://xxx-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**.

## Lab 2. ETL with AWS Glue

**Add crawler**

**Add another data store**

Yes  
 No

**Back** **Next**

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

- On the Choose an IAM role page, select **Choose an existing IAM role**.  
For IAM role, select the existing role “xxx-GlueLabRole-xxx” 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-dmslab3bucket-xg1hdqg0bs/tickets/dms\_parquet

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

**Back** **Next**

- For **Frequency**, select “Run On Demand” and Click **Next**.

**Add crawler**

**Create a schedule for this crawler**

**Frequency** [Run on demand](#)

**Back** **Next**

- For the crawler’s output database, choose your existing database which you created earlier e.g. **“ticketdata”**
- For the **Prefix added to tables** (optional), type **“parquet\_”**

## Lab 2. ETL with AWS Glue

**Add crawler**

**Configure the crawler's output**

**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**

**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-14R6WFWGZ4MB

**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 tables were added from 1 to 5, depending on how many parquet ETL conversions you set up in the previous section

**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.

Name	Schedule	Catalog type	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
glue-lab-cdc-cra...		Glue	Ready	Logs	1 min	1 min	0	2
glue-lab-crawler		Glue	Ready	Logs	1 min	1 min	0	15
glue-lab-parquet...		Glue	Ready	Logs	1 min	1 min	0	5

**User preferences**  
Showing: 1 - 3

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.

## Lab 2. ETL with AWS Glue

The screenshot shows the AWS Glue Tables interface. On the left, there's a navigation sidebar with options like AWS Glue, Data catalog, Databases, Tables, Connections, Crawlers, Classifiers, Settings, ETL, Workflows, Jobs, ML Transforms, Triggers, Dev endpoints, and Notebooks. The main area displays a table of tables. The columns are Name, Database, Location, Classification, Last updated, and Deprecated. One row, 'parquet\_person', is highlighted with a red box.

Name	Database	Location	Classification	Last updated	Deprecated
mb_data	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	10 January 2020 1:37 PM UTC-5	
name_data	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	10 January 2020 1:37 PM UTC-5	
nfl_data	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	10 January 2020 1:37 PM UTC-5	
nfl_stadium_data	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	10 January 2020 1:37 PM UTC-5	
parquet_person	ticketdata	s3://dmslab-student-dmslabs3buck...	parquet	23 January 2020 1:49 PM UTC-5	
parquet_person_annotation	ticketdata	s3://dmslab-student-dmslabs3buck...	parquet	23 January 2020 1:49 PM UTC-5	
parquet_sport_team	ticketdata	s3://dmslab-student-dmslabs3buck...	parquet	23 January 2020 1:49 PM UTC-5	
parquet_sporting_event	ticketdata	s3://dmslab-student-dmslabs3buck...	parquet	23 January 2020 1:49 PM UTC-5	
parquet_sporting_event_ticket	ticketdata	s3://dmslab-student-dmslabs3buck...	parquet	23 January 2020 1:49 PM UTC-5	
person	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	10 January 2020 1:48 PM UTC-5	

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

**\*\*Pre-requisite: Completion of CDC 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.

The screenshot shows the AWS Glue Crawlers interface. On the left, there's a navigation sidebar with options like AWS Glue, Data catalog, Databases, Tables, Connections, Crawlers, Classifiers, and Settings. The main area displays a table of crawlers. The columns are Name, Schedule, Status, Logs, Last runtime, Median runtime, Tables updated, and Tables added. A message says 'You don't have any crawlers yet.' with a 'Add crawler' button.

Name	Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
You don't have any crawlers yet.							
<a href="#">Add crawler</a>							

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**.

The screenshot shows the 'Add crawler' wizard. Step 1: Crawler Info. It has a sidebar with tabs: Crawler info (selected), Crawler source type, Data store, IAM Role, Schedule, Output, and Review all steps. The main area has a title 'Add information about your crawler'. It shows a 'Crawler name' field with 'glue-lab-cdc-crawler' and a note 'Tags, description, security configuration, and classifiers (optional)'. At the bottom is a 'Next' button.

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

## Lab 2. ETL with AWS Glue

**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**

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-xxx/cdc/dms\_sample**"
7. Click **Next**.

**Add crawler**

**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**.

**Add crawler**

**Add another data store**

Chosen data stores

S3: s3://dmslab-stud...

Yes  No

**Back** **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 **xxx-GlueLabRole-xxx**. E.g. "dmslab-student-GlueLabRole-ZOQDII7JTBUM"
10. Click **Next**.

## Lab 2. ETL with AWS Glue

**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-dmslab3bucket-xg1hdqg60ts/cdc/dms\_sample

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

**Back** **Next**

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

**Add crawler**

**Create a schedule for this crawler**

**Frequency**

Run on demand

**Back** **Next**

- On the Configure the crawler's output page, select the existing **Database** for crawler output (e.g., "ticketdata").
- For **Prefix added to tables (optional)**, specify "cdc\_"
- 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)**

Type a prefix added to table names

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. [i](#)

Update all new and existing partitions with metadata from the table. [i](#)

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. [i](#)

**Back** **Next**

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

## Lab 2. ETL with AWS Glue

### 16. Click Run it now.

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.

Name	Schedule	Catalog type	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
glue-lab-cdc-cra...		Glue	Ready	Logs	1 min	1 min	0	2
glue-lab-crawler		Glue	Ready	Logs	1 min	1 min	0	15

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

## Lab 2. ETL with AWS Glue

Tables						
Name	Database	Location	Classification	Last updated	Deprecated	
cdc_sporting_event_ticket	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	23 January 2020 4:38 PM UTC-5		
cdc_ticket_purchase_hist	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	23 January 2020 4:38 PM UTC-5		
mlb_data	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	10 January 2020 1:37 PM UTC-5		
name_data	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	10 January 2020 1:37 PM UTC-5		
nfl_data	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	10 January 2020 1:37 PM UTC-5		
nfl_stadium_data	ticketdata	s3://dmslab-student-dmslabs3buck...	csv	10 January 2020 1:37 PM UTC-5		
parquet_person	ticketdata	s3://dmslab-student-dmslabs3buck...	parquet	23 January 2020 1:49 PM UTC-5		
parquet_sport_location	ticketdata	s3://dmslab-student-dmslabs3buck...	parquet	23 January 2020 1:49 PM UTC-5		
parquet_sport_team	ticketdata	s3://dmslab-student-dmslabs3buck...	parquet	23 January 2020 1:49 PM UTC-5		

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

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

Name	Type	ETL language	Script location	Last modified	Job bookmark

- On the Job properties page, make the following selections:
  - For **Name**, type **Glue-Lab-TicketHistory-Parquet-with-bookmark**.
  - For **IAM role**, choose existing role “xxx-GlueLabRole-xxx”
  - 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.)
- Expand the **Advanced properties** section. For Job bookmark, select **Enable** from the drop-down option.
- Expand on the **Monitoring** options, enable **Job metrics**.
- Click **Next**

## Lab 2. ETL with AWS Glue

**Add job**

Configure the job properties

<b>Name</b>	Glue-Lab-TicketHistory-Parquet-with-bookmark
<b>IAM role</b>	dmslab-student-GlueJobRole-14R6WFBWGZ4MB
<b>Type</b>	Spark
<b>Glue version</b>	Spark 2.4, Python 3 (Glue Version 1.0)
<b>This job runs</b>	<input checked="" type="radio"/> A proposed script generated by AWS Glue <small>(1)</small>
	<input type="radio"/> An existing script that you provide
	<input type="radio"/> A new script to be authored by you
<b>Script file name</b>	Glue-Lab-TicketHistory-Parquet-with-bookmark
<b>S3 path where the script is stored</b>	s3://aws-glue-scripts-665993140268-us-east-1/deshtut1
<b>Temporary directory</b>	s3://aws-glue-temporary-666993140268-us-east-1/deshtut1
<b>Advanced properties</b>	
<b>Job bookmark</b>	<input type="checkbox"/> Enable
<b>Monitoring options</b>	
<input checked="" type="checkbox"/> Job metrics <small>(1)</small>	
<input type="checkbox"/> Continuous logging	
<input type="checkbox"/> Spark UI <small>(1)</small>	
<b>Tags (optional)</b>	

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**

**Add job**

Choose a data source

Name	Database	Location	Classification
bookmark_parquet_ticket_purchase_history	ticketdata	s3://dmslab-student-dmslabs3bucket-xg1hdyg0ls/cdc_bookmark/ticket.../parquet	
cdc_sporting_event_ticket	ticketdata	s3://dmslab-student-dmslabs3bucket-xg1hdyg0ls/cdc_dms_sample/.../csv	
<b>cdc_ticket_purchase_hist</b>	ticketdata	s3://dmslab-student-dmslabs3bucket-xg1hdyg0ls/cdc_dms_sample/.../csv	
clickstream_data	processed-data	s3://rawdataset-deshtut1/Clickstream_data/	json
csv_clickstream_data	processed-data	s3://processed-deshtut1/Clickstream_data/	csv

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

**Add job**

Choose a transform type

<input checked="" type="radio"/> <b>Change schema</b>	Change schema of your source data and create a new target dataset
<input type="radio"/> <b>Find matching records</b>	Use machine learning to find matching records within your source data

**Back** **Next**

8. In **Choose a data target**:

- For **Data store**: select **amazon S3**
- Format: **parquet**
- Target path**: s3://xxx-dmslabs3bucket-xxx/cdc\_bookmark/ticket\_purchase\_history/data/
- Click **Next**

## Lab 2. ETL with AWS Glue

Add job

- Job properties
  - Create tables in your data target
  - Use tables in the data catalog and update your data target
  
- Data source
  - cdc\_ticket\_purchases...
  
- Transform type
  - Change schema
  
- Data target
  - cdc\_ticket\_purchase...
  
- Schema

Choose a data target

**Data store**  
 Amazon S3

**Format**  
 Parquet

**Target path**  
 11/12/2018/cdc\_bookmark/ticket\_purchase\_history/data/

Back
Next

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

Add job

Job properties  
Glue Job  
Ticket History  
Parquet with  
binaries

Data source  
cdc\_ticket\_purchas...

Transform type  
Change schema

Data target  
s3://mlab-studen...

Schema

Map the source columns to target columns.

Verify the mappings created by AWS Glue. Change mappings by choosing other columns with Map to target. You can Clear all mappings and Reset to default AWS Glue mappings. AWS Glue generates your script with the defined mappings.

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

[Add column](#) [Clear](#) [Reset](#)

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.

Job: Glue-Lab-TicketHistory-Parquet-with-bookmark

Action | Save | Run job | Generate diagram

Insert template at cursor | Source | Target | Target Location | Transform | Script

Database Name (optional): Table Name (optional): job\_ticket.purchase\_hist

Transform Name App/Mapping

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from pyspark.context import GlueContext
6 from awsglue.context import GlueContext
7 from awsglue.job import Job
8
9 # --- [JOE_NAME]
10 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
11
12 sc = SparkContext()
13 glueContext = GlueContext(sc)
14 spark = glueContext.sparkSession
15 job = Job(glueContext)
16 job.addArgs(['JOB_NAME', args])
17
18 # --- [JOE_NAME]
19 # --- [JOE_NAME]
20 # --- [JOE_NAME]
21 # --- [JOE_NAME]
22 # --- [JOE_NAME]
23 # --- [JOE_NAME]
24 # --- [JOE_NAME]
25 # --- [JOE_NAME]
26 # --- [JOE_NAME]
27 # --- [JOE_NAME]
28 # --- [JOE_NAME]
29 # --- [JOE_NAME]
30 # --- [JOE_NAME]
31 # --- [JOE_NAME]
32 # --- [JOE_NAME]
33 # --- [JOE_NAME]
34 # --- [JOE_NAME]
35 # --- [JOE_NAME]
36 # --- [JOE_NAME]
37 # --- [JOE_NAME]
38 # --- [JOE_NAME]
39 # --- [JOE_NAME]
40 # --- [JOE_NAME]
41 # --- [JOE_NAME]
```

Transform Name ResolveChoice

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.dynamicframe import DynamicFrame
4 from awsglue.dynamicframe import DynamicFrameMapping
5 from awsglue.dynamicframe import DynamicFrameTransformation
6 from awsglue.dynamicframe import DynamicFrameTransformationContext
7 from awsglue.dynamicframe import DynamicFrameTransformationOptions
8
9 # --- [JOE_NAME]
10 # --- [JOE_NAME]
11 # --- [JOE_NAME]
12 # --- [JOE_NAME]
13 # --- [JOE_NAME]
14 # --- [JOE_NAME]
15 # --- [JOE_NAME]
16 # --- [JOE_NAME]
17 # --- [JOE_NAME]
18 # --- [JOE_NAME]
19 # --- [JOE_NAME]
20 # --- [JOE_NAME]
21 # --- [JOE_NAME]
22 # --- [JOE_NAME]
23 # --- [JOE_NAME]
24 # --- [JOE_NAME]
25 # --- [JOE_NAME]
26 # --- [JOE_NAME]
27 # --- [JOE_NAME]
28 # --- [JOE_NAME]
29 # --- [JOE_NAME]
30 # --- [JOE_NAME]
31 # --- [JOE_NAME]
32 # --- [JOE_NAME]
33 # --- [JOE_NAME]
34 # --- [JOE_NAME]
35 # --- [JOE_NAME]
36 # --- [JOE_NAME]
37 # --- [JOE_NAME]
38 # --- [JOE_NAME]
39 # --- [JOE_NAME]
40 # --- [JOE_NAME]
41 # --- [JOE_NAME]
```

Transform Name DropNullFields

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.dynamicframe import DynamicFrame
4 from awsglue.dynamicframe import DynamicFrameMapping
5 from awsglue.dynamicframe import DynamicFrameTransformation
6 from awsglue.dynamicframe import DynamicFrameTransformationContext
7 from awsglue.dynamicframe import DynamicFrameTransformationOptions
8
9 # --- [JOE_NAME]
10 # --- [JOE_NAME]
11 # --- [JOE_NAME]
12 # --- [JOE_NAME]
13 # --- [JOE_NAME]
14 # --- [JOE_NAME]
15 # --- [JOE_NAME]
16 # --- [JOE_NAME]
17 # --- [JOE_NAME]
18 # --- [JOE_NAME]
19 # --- [JOE_NAME]
20 # --- [JOE_NAME]
21 # --- [JOE_NAME]
22 # --- [JOE_NAME]
23 # --- [JOE_NAME]
24 # --- [JOE_NAME]
25 # --- [JOE_NAME]
26 # --- [JOE_NAME]
27 # --- [JOE_NAME]
28 # --- [JOE_NAME]
29 # --- [JOE_NAME]
30 # --- [JOE_NAME]
31 # --- [JOE_NAME]
32 # --- [JOE_NAME]
33 # --- [JOE_NAME]
34 # --- [JOE_NAME]
35 # --- [JOE_NAME]
36 # --- [JOE_NAME]
37 # --- [JOE_NAME]
38 # --- [JOE_NAME]
39 # --- [JOE_NAME]
40 # --- [JOE_NAME]
41 # --- [JOE_NAME]
```

Path: s3://mlab-student-dsml-labs/JobLab/JobLab/JobLab/bookmark\_and\_permanent

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

Overview			
<input type="text"/> Type a prefix and press Enter to search. Press ESC to clear.			
<a href="#">Upload</a> <a href="#">+ Create folder</a> <a href="#">Download</a> <a href="#">Actions</a>			
US East (N. Virginia) <a href="#">Viewing 1 to 2</a>			
<input type="checkbox"/> Name	Last modified	Size	Storage class
<input type="checkbox"/> <a href="#">part-00000-498ea7fc-2act-4787-b431-9e16f5e24a3f-c0000.snappy.parquet</a>	Jan 24, 2020 7:03:16 PM GMT-0500	1.1 MB	Standard
<input type="checkbox"/> <a href="#">part-00001-498ea7fc-2act-4787-b431-9e16f5e24a3f-c0000.snappy.parquet</a>	Jan 24, 2020 7:03:16 PM GMT-0500	1.2 MB	Standard
Viewing 1 to 2			

## Lab 2. ETL with AWS Glue

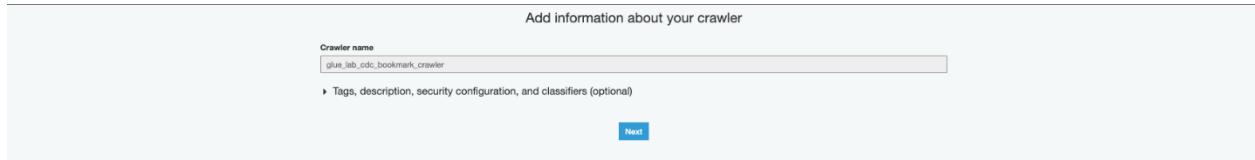
### Step 3: Create Glue crawler for Parquet data in S3

- 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**



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



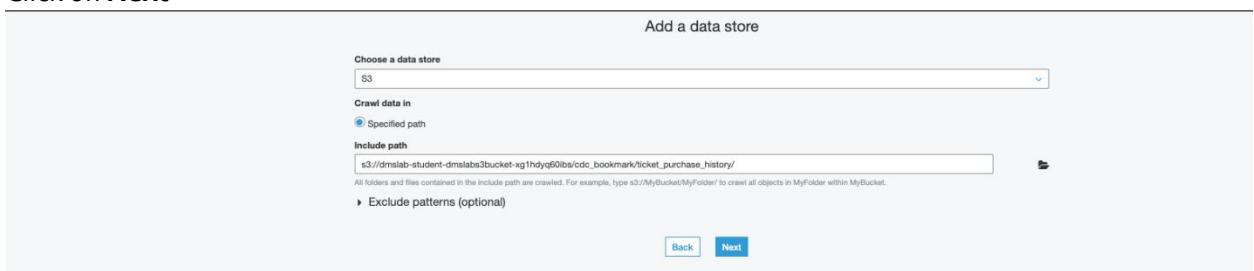
- In specify **crawler source type**, for crawler source type, select **Data stores**. Click **Next**



- In **Add a data store**:

- For **Choose a data store**, select **S3**
- For the **Include path**, click the folder icon and choose your target S3 bucket, then append **/cdc\_bookmark/ticket\_purchase\_history**, e.g., "s3://xxx-dmslabs3bucket-xxx/cdc\_bookmark/ticket\_purchase\_history"

- Click on **Next**



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



## Lab 2. ETL with AWS Glue

8. In **Choose an IAM role**, select an existing IAM role contains **GlueLabRole** text. Something looks like this: xxx-GlueLabRole-xxx

The screenshot shows the 'Choose an IAM role' step. It includes a note about the IAM role allowing the crawler to run and access Amazon S3 data stores. There are three radio button options: 'Update a policy in an IAM role', 'Choose an existing IAM role' (which is selected), and 'Create an IAM role'. A dropdown menu shows the selected IAM role: 'dmslab-student-GlueLabRole-14R8WFBWGZ4MB'. Below the dropdown, a note states: 'This role must provide permissions similar to the AWS managed policy, **AWSGlueServiceRole**, plus access to your data stores.' It also lists the ARN: 'arn:aws:iam::065953140268:role/service-role/dmslab-student-GlueLabRole-14R8WFBWGZ4MB'. A link to the IAM console is provided. At the bottom are 'Back' and 'Next' buttons.

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

The screenshot shows the 'Create a schedule for this crawler' step. It has a 'Frequency' dropdown set to 'Run on demand'. At the bottom are 'Back' and 'Next' buttons.

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**

The screenshot shows the 'Configure the crawler's output' step. It has a 'Database' dropdown set to 'ticketdata' and an 'Add database' button. A 'Prefix added to tables (optional)' input field contains 'bookmark\_parquet\_'. Below the fields are two expandable sections: 'Grouping behavior for S3 data (optional)' and 'Configuration options (optional)'. At the bottom are 'Back' and 'Next' buttons.

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

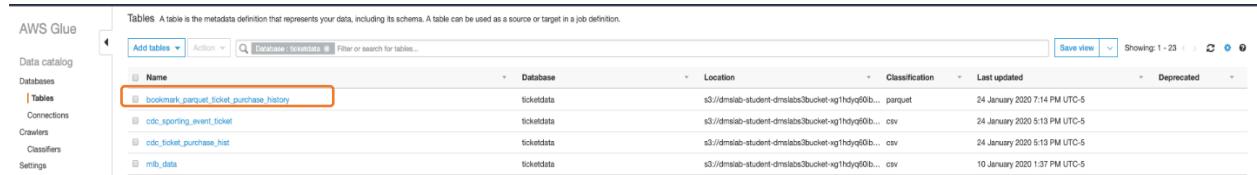
The screenshot shows the review step of the AWS Glue Crawler setup wizard. On the left is a sidebar with a tree view of configuration settings, including 'Crawler info', 'Crawler source type', 'Data stores', 'IAM Role', 'Schedule', 'Output', and 'Review all steps'. The main area displays five panels with the following details:

- Crawler info:** Name: 'glue\_lab\_cdc\_bookmark\_crawler', Tags: '-'
- Data stores:** Data store: S3, Include path: 's3://dmslab-student-dmslab3bucket-xg1hdqg0bs/cdc\_bookmark/ticket\_purchase\_history/'
- IAM role:** IAM role: 'arn:aws:iam::065953140268:role/service-role/dmslab-student-GlueLabRole-14R8WFBWGZ4MB'
- Schedule:** Schedule: Run on demand
- Output:** Database: 'ticketdata', Prefix added to tables (optional): 'bookmark\_parquet\_'. Note: 'Create a single schema for each S3 path' is checked. Below it are 'Configuration options' and 'Output' sections.

At the bottom are 'Back' and 'Finish' buttons.

## Lab 2. ETL with AWS Glue

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”



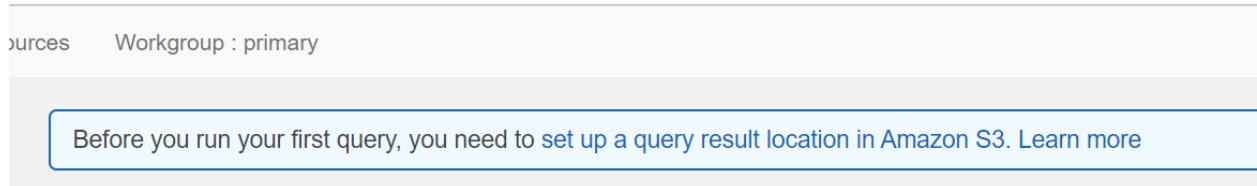
Name	Database	Location	Classification	Last updated	Deprecated
bookmark_parquet_ticket_purchase_history	ticketdata	s3://dmstlab-student-dmstlats3bucket:xg1hdyg90b... parquet		24 January 2020 7:14 PM UTC-5	
cdc_sporting_event_ticket	ticketdata	s3://dmstlab-student-dmstlats3bucket:xg1hdyg90b... csv		24 January 2020 5:13 PM UTC-5	
cdc_ticket_purchase_hist	ticketdata	s3://dmstlab-student-dmstlats3bucket:xg1hdyg90b... csv		24 January 2020 5:13 PM UTC-5	
mib_data	ticketdata	s3://dmstlab-student-dmstlats3bucket:xg1hdyg90b... csv		10 January 2020 1:37 PM UTC-5	

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

If it's the first time you are using Athena in your AWS Account, click **Get Started**



Then click **set up a query result location in Amazon S3** at the top



## Lab 2. ETL with AWS Glue

In the pop-up window in the **Query result location** field, enter your s3 bucket location followed by /, so that it looks like `s3://xxx-dmslabs3bucket-xxx/` and click **Save**

### Settings

Settings apply by default to all new queries. [Learn more](#)

**Workgroup:** `primary`

Query result location

Example: `s3://query-results-bucket/folder/`

Encrypt query results  [i](#)

Autocomplete  [i](#)

[Cancel](#)

**Save**

To select some rows from the table, try running:

```
SELECT * FROM "ticketdata"."bookmark_parquet_ticket_purchase_history"  
limit 10;
```

To get a row count, run:

```
SELECT count(*) as recordcount FROM  
"ticketdata"."bookmark_parquet_ticket_purchase_history";
```

Before moving on to next step, note the rowcount.

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

Ask your instructor generate more CDC data at source database, if you ran the instructor setup on your own, then make sure to follow "**Generate the CDC Data**" section from instructor prelab.

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.

## Lab 2. ETL with AWS Glue

The screenshot shows the Amazon S3 console with the path: Amazon S3 > dmslab-student-dmslabs3bucket-xg1hdyyq60bs > cdc\_bookmark > ticket\_purchase\_history > data. The 'Overview' tab is selected. The table lists 9 objects:

Name	Last modified	Size	Storage class
part-00000-05202004-2fbc-47c2-0249-d9100b7edad-c000.snappy.parquet	Jan 24, 2020 9:20:13 PM GMT+0500	9.3 kB	Standard
part-00000-49bea7fc-2ac1-4787-b421-9e105a142f-c000.snappy.parquet	Jan 24, 2020 7:03:16 PM GMT+0500	1.1 MB	Standard
part-00000-d1666723-315b-45fa-b230-c3d2e1519a-c000.snappy.parquet	Jan 25, 2020 11:24:29 PM GMT+0500	1.7 MB	Standard
part-00000-efcc00ff-dc40-43bc-b230-c3d2e1519a-c000.snappy.parquet	Jan 25, 2020 10:24:29 PM GMT+0500	7.2 kB	Standard
part-00000-05202004-2fbc-47c2-0249-d9100b7edad-c000.snappy.parquet	Jan 24, 2020 9:20:13 PM GMT+0500	66.5 kB	Standard
part-00001-49bea7fc-2ac1-4787-b421-9e105a142f-c000.snappy.parquet	Jan 24, 2020 7:03:16 PM GMT+0500	1.2 MB	Standard
part-00001-d1666723-315b-45fa-b230-c3d2e1519a-c000.snappy.parquet	Jan 25, 2020 11:24:29 PM GMT+0500	1.7 MB	Standard
part-00002-05202004-2fbc-47c2-0249-d9100b7edad-c000.snappy.parquet	Jan 24, 2020 9:20:15 PM GMT+0500	1.7 MB	Standard
part-00002-d1666723-315b-45fa-b230-c3d2e1519a-c000.snappy.parquet	Jan 25, 2020 11:24:19 PM GMT+0500	1.9 MB	Standard

2. Rerun the Glue job **Glue-Lab-TicketHistory-Parquet-with-bookmark** you created in Step 2
3. Go to the Athena Console, and rerun the following query to notice the increase in row count:

```
SELECT count(*) as recordcount FROM  
"ticketdata"."bookmark_parquet_ticket_purchase_history";
```

To review the latest transactions, run:

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

## PART C: Glue Workflows (Optional, self-paced)

**\*\*Pre-requisite before creating workflow\*\* - completed Part B**

### 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:

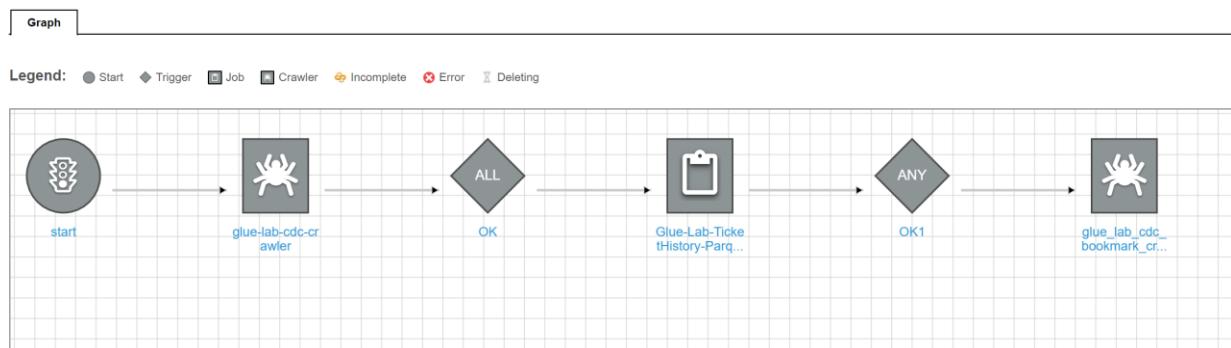
Above mentioned Part A (ETL with Glue) and Part B (Glue Job Bookmarks) can be created and executed using workflows. Complex ETL jobs involving multiple crawlers and jobs can also be created and

## Lab 2. ETL with AWS Glue

executed using workflows in an automated fashion. Below is a simple example to demonstrate how to create and run workflows.

Try creating a new Glue Workflow to string together the two Crawlers and one Job from part B as follows:

On-demand trigger -> glue-lab-cdc-crawler -> Glue-Lab-TicketHistory-Parquet-with-bookmark -> glue\_lab\_cdc\_bookmark\_crawler



### 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 page. On the left, there is a sidebar with navigation links for Data catalog, Databases, Tables, Connections, Crawlers, Classifiers, Settings, ETL (which is selected), Workflows (which is also selected), Jobs, and ML Transforms. The main content area has a title "Workflows (0)". A sub-instruction below the title states: "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 the placeholder "Filter workflows". A button labeled "Add workflow" is visible. A table header with columns "Name", "Last run", "Last run status", and "Last modified" is shown, but the table body is empty, indicating "No workflows". A blue button labeled "Add a new ETL workflow" is located at the bottom right of the table area.

2. Give the workflow name as "**Workflow\_tickethistory**". Provide a description (optional) and click on **Add Workflow** to create it.
3. Click on the **workflow** and scroll to the bottom of the page. You will see an option **Add Trigger**. Click on that button.

The screenshot shows the AWS Glue Workflows page after creating a workflow. The sidebar and top navigation are identical to the previous screenshot. The main content area now displays a table with one row, showing the newly created workflow. The table columns are "Name", "Last run", "Last run status", and "Last modified". The row contains the value "Workflow\_MLB\_Data" for the Name column, and "-" for the Last run, Last run status, and Last modified columns. The date "Sun, 26 Jan 2020 05:12:03 GMT" is shown in the Last modified column.

## Lab 2. ETL with AWS Glue

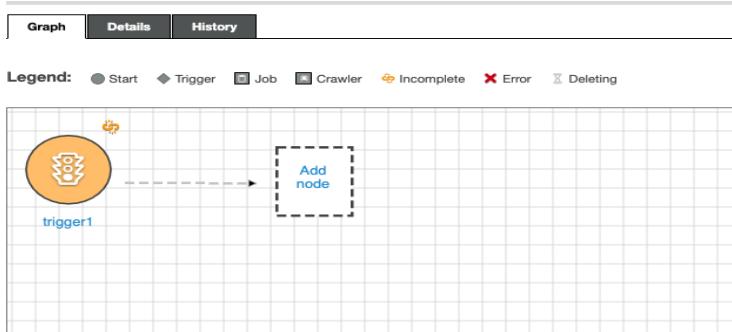


4. In **Add Trigger** window, From Clone Existing and Add New options, click on **Add New**.
  - a. Provide **Name** as "**trigger1**"
  - b. Provide a **description**: Trigger to start workflow
  - c. **Trigger type**: On-demand.
  - d. Click on **Add**

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

The dialog box has a title bar "Add trigger" and a close button "X". It contains two tabs: "Clone existing" (selected) and "Add new".  
Name: trigger1  
Description (optional): Trigger to start the workflow  
Trigger type:  Schedule  Event  On demand  
Buttons: Cancel, Add

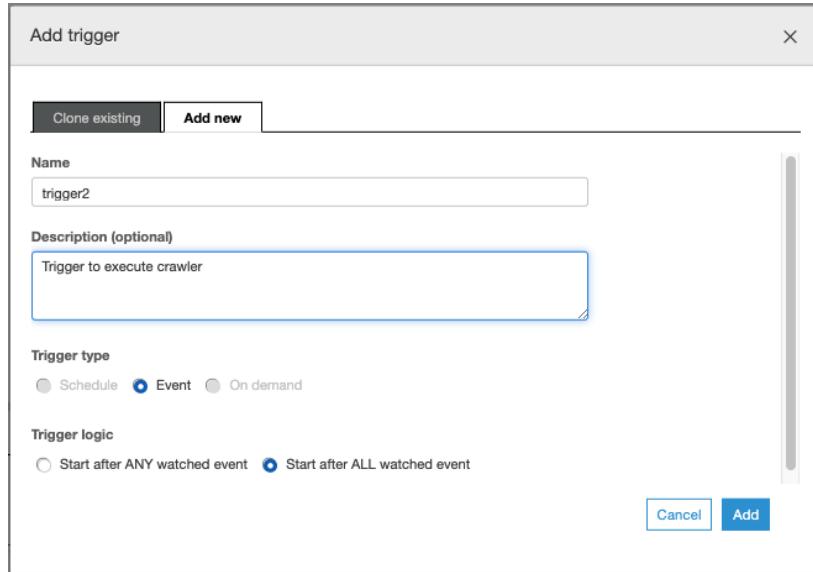
5. Click on **trigger1** to add a **new node**. New Node can be a crawler or job, depending upon the workflow you want to build.



6. Click on **Add node**, a new window to add jobs or crawlers will open. Select the Crawler **glue-lab-cdc-crawler**, then **Add**.
7. Click on the crawler and **Add Trigger** provide the following:
  - a. **Name: trigger2**
  - b. **Description: Trigger to execute job**
  - c. **Trigger type: Event**
  - d. **Trigger logic: Start after ALL watched event**. This will make sure that job starts once Glue Crawler finishes.

## Lab 2. ETL with AWS Glue

- e. Click Add



8. After **trigger2** is added to workflow, Click on **Add node**, select job **Glue-Lab-TicketHistory-Parquet-with-bookmark**, click **Add**.
9. Click on the job and **Add Trigger** provide the following:
  - a. **Name:** trigger3
  - b. **Description:** Trigger to execute crawler
  - c. **Trigger type:** Event
  - d. **Trigger logic:** Start after ANY watched event. This will make sure that crawler starts once Glue job finishes processing of ALL data.
  - e. Click **Add**
10. Click on **Add node**, Select the Crawler **glue\_lab\_cdc\_bookmark\_crawler**, then **Add**.
11. Select your workflow, click on **Actions->Run** and this will start the first trigger "trigger1"
12. Once the workflow is completed, you will observe that glue job and crawlers have been successfully executed.

Congratulations!! You have successfully completed this lab