



Amazon Web Services Data Engineering Immersion Day

Extract, Transform and Load Data Lake with Glue

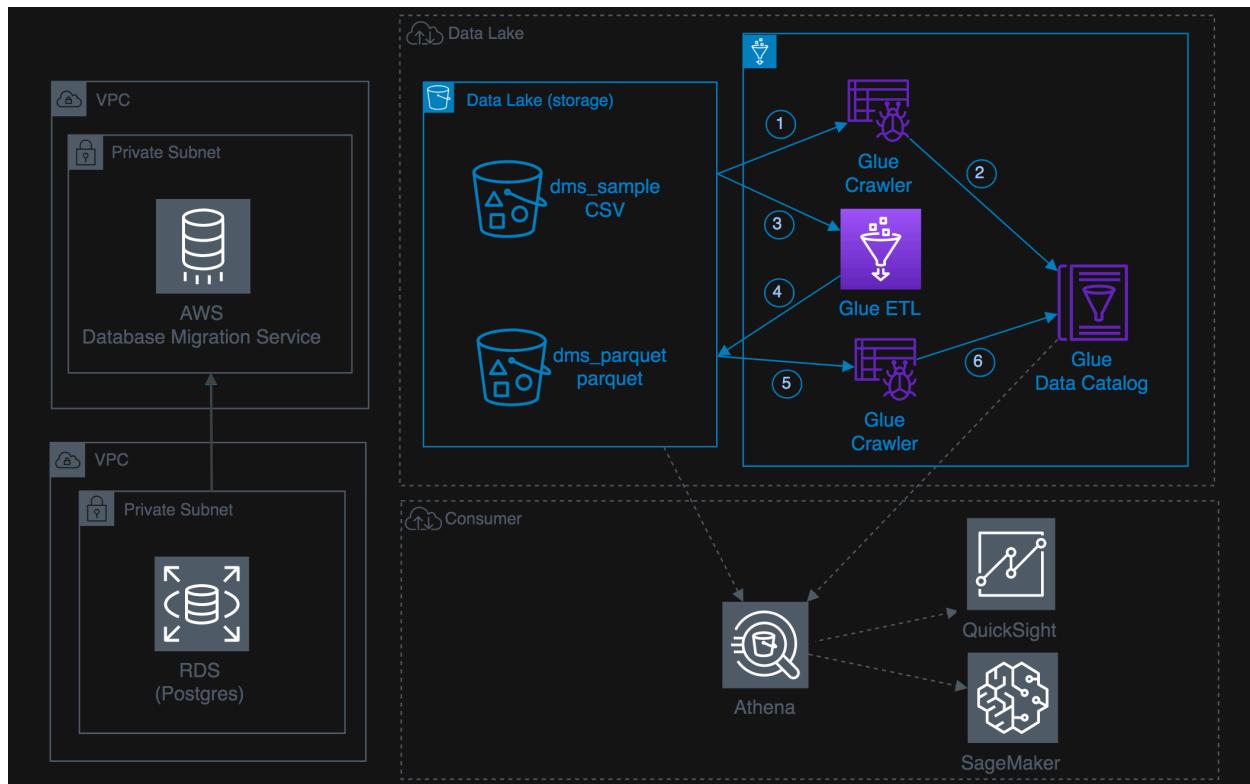
March 2020

Table of Contents

<i>Introduction</i>	2
Prerequisites:	2
Tasks Completed in this Lab:.....	2
Getting Started.....	3
<i>PART-(A): Data Validation and ETL.....</i>	3
Create Glue Crawler for initial full load data	3
Data ETL Exercise	9
Create Glue Crawler for Parquet Files	15
<i>PART-(B): Glue Job Bookmark (Optional):.....</i>	18
Step 1: Create Glue Crawler for ongoing replication (CDC Data)	18
Step 2: Create a Glue Job with Bookmark Enabled	22
Step 3: Create Glue crawler for Parquet data in S3	24
Step 4: Generate CDC data and to observe bookmark functionality	27
<i>PART- (C): Glue Workflows (Optional)</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:

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 GitHub - <https://github.com/aws-samples/data-engineering-for-aws-immersion-day>

Getting Started

Navigate to the AWS Glue service.

The screenshot shows the AWS Services Catalog interface. At the top, there is a search bar with the text "glue" entered. Below the search bar, a list of services is displayed, with "AWS Glue" being the first item. The "AWS Glue" entry includes a brief description: "AWS Glue is a fully managed ETL (extract, transform, and load) service". There are also links for "AWS Lake Formation" and "All services".

PART-(A): Data Validation and ETL

Create Glue Crawler for initial full load data

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

The screenshot shows the AWS Glue Crawlers page. On the left, there is a navigation sidebar with options like "AWS Glue", "Data catalog", "Databases", "Tables", "Connections", "Crawlers" (which is selected and highlighted in orange), "Classifiers", and "Settings". The main content area is titled "Crawlers" and contains a sub-instruction: "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." Below this, there are buttons for "Add crawler", "Run crawler", and "Action". A search bar says "Filter by tags and attributes" and shows "Showing: 0 - 0". To the right, there is a table header with columns: Name, Schedule, Status, Logs, Last runtime, Median runtime, Tables updated, and Tables added. A message at the bottom states "You don't have any crawlers yet." with a "Add crawler" button.

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

5. Choose Crawler Source Type as Data Source 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**

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 for your DMS initial export, e.g., "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**

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**

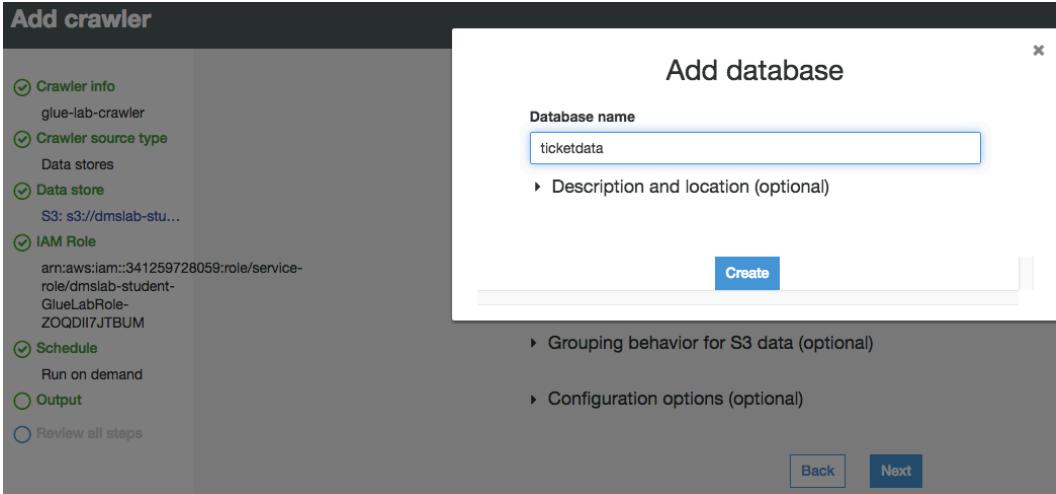
9. On the **Choose an IAM role** page, make the following selections:
- Select **Choose an existing IAM role**.
 - For IAM role, select **<stackname>-GlueLabRole-<RandomString>** created from the AWS CloudFormation template during the student lab.
For example "dmslab-student-GlueLabRole-ZOQDII7JTBUM"

10. Click **Next**.

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

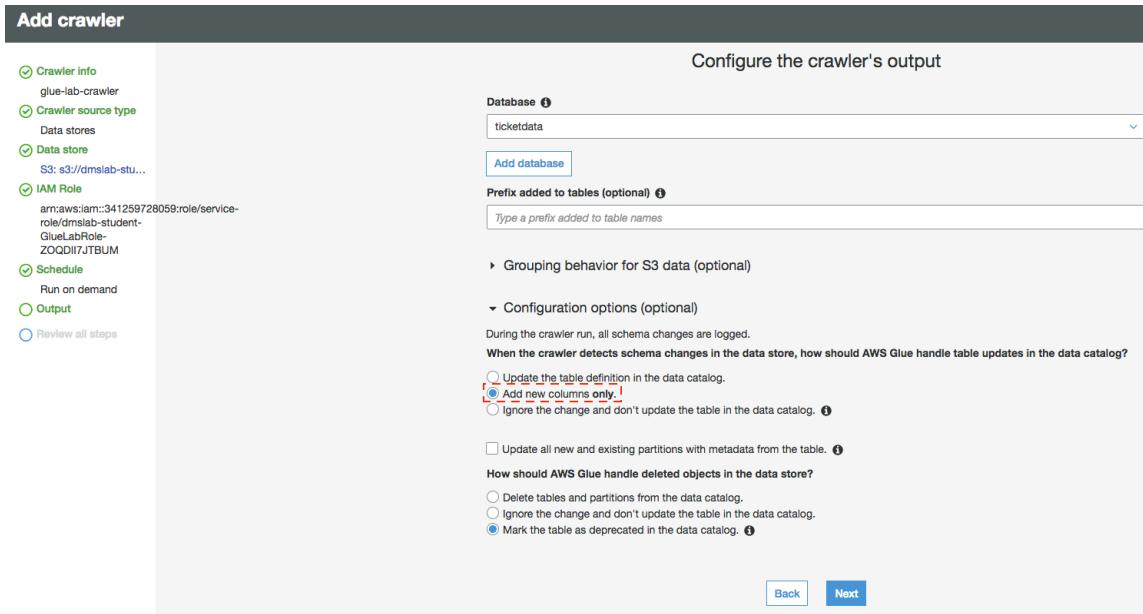
12. On the Configure the crawler's output page, click **Add database** to create a new database for our Glue Catalogue.

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

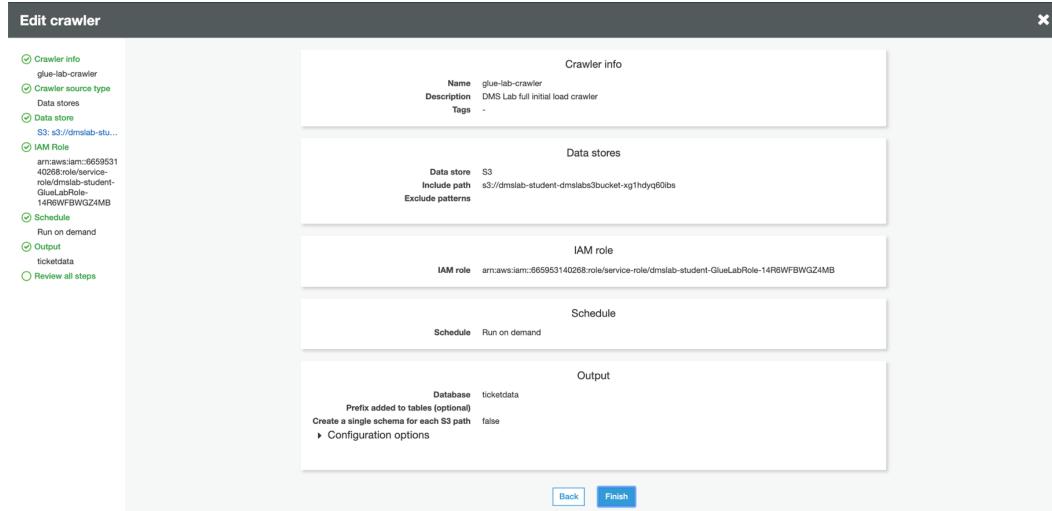


14. For Prefix added to tables (optional), leave the field empty.

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' section is currently selected. In the main area, there's a table titled 'Tables' with columns: Name, Database, Location, Classification, Last updated, and Deprecated. The 'person' table is highlighted with a red box. Other tables listed include 'mib_data', 'name_data', 'nfl_data', 'nfl_stadium_data', 'player', 'seat', 'seat_type', 'sport_division', and 'sport_league'. Each table entry includes its location (e.g., s3://dmslab-student-dmslabs3bucket...), classification (e.g., csv), and last update time (e.g., 10 January 2020 1:37 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 'Edit table' view 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 displays detailed information about the table, including its name ('person'), database ('ticketdata'), classification ('csv'), and location ('s3://dmslab-student-dmslabs3bucket-ug1hdyqf0bs/tickets/dmt_sample/person'). It also shows the table's creation date ('2020-01-10T13:37:23Z'), last update date ('2020-01-10T13:37:23Z'), and various serde parameters like 'field.delim' and 'stozKey'. Below this, the 'Schema' section shows a table with columns: Column name, Data type, Partition key, and Comment. The columns listed are col0, col1, col2, and col3, all of which are of type 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.

AWS Glue

Tables > person

Add column

Edit schema

Last updated 10 Jan 2020 Table Version (Current version) ▾

Cancel Save

Showing: 1 - 4 of 4

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”

dmslab-student-dmslabs3bucket-z3fwyjc9thf9

Overview

Type a prefix and press Enter to search. Press ESC to clear.

Upload + Create folder Download Actions

Name

- person
- sport_location
- sport_team
- sporting_event
- sporting_event_ticket

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

AWS Glue

Jobs 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.

User preferences

Showing: 0 - 0

Add job Action Filter by attributes

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

Add job

Data catalog

Databases

Tables

Connections

Crawlers

Classifiers

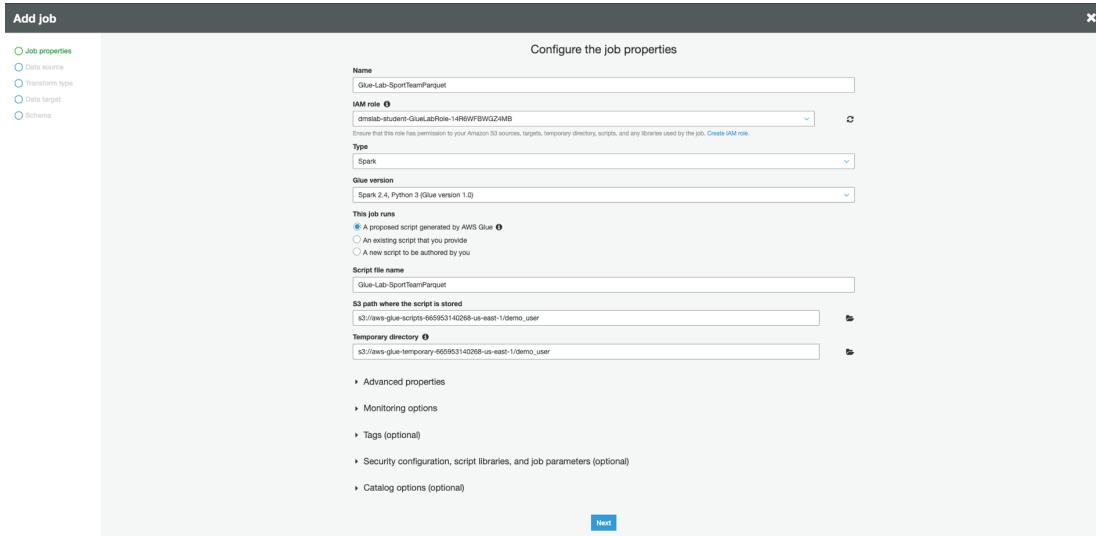
ETL

Jobs

Triggers

Dev endpoints

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. "dmslab-student-GlueLabRole-ZOQDII7JTBUM"
 - 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.)



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

Add job

Job properties
Glue-Lab-SportTeamParquet

Data source
sport_team

Transform type

Data target

Schema

Choose a data source

search : sport_team Filter or search for tables...

Name	Database	Location	Classification
<input checked="" type="radio"/> sport_team	ticketdata	s3://dmslab-student-dmslabs3bucket-wotl4bf73cw3/tick...	csv
<input type="radio"/> sportstickets.dms_sample_sport_team	onprem-db	sportstickets.dms_sample.sport_team	postgresql

Showing: 1 - 2

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

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

Add job

Job properties
Glue-Lab-SportTeamParquet

Data source
sport_team

Transform type
Change schema

Data target

Schema

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**, choose a folder location which you create at the beginning of this section to store the results e.g., "s3://dmslab-student-dmslabs3bucket-wotl4bf73cw3/tickets/dms_parquet/sport_team"

10. Click **Next**.

Add job

Job properties
Glue-Lab-SportTeamParquet

Data source
sport_team

Transform type
Change schema

Data target

Schema

Choose a data target

Create tables in your data target

Use tables in the data catalog and update your data target

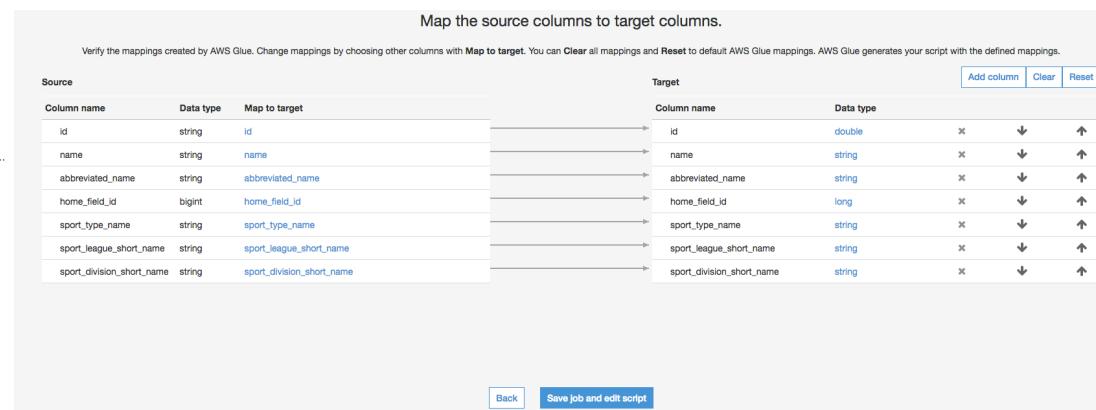
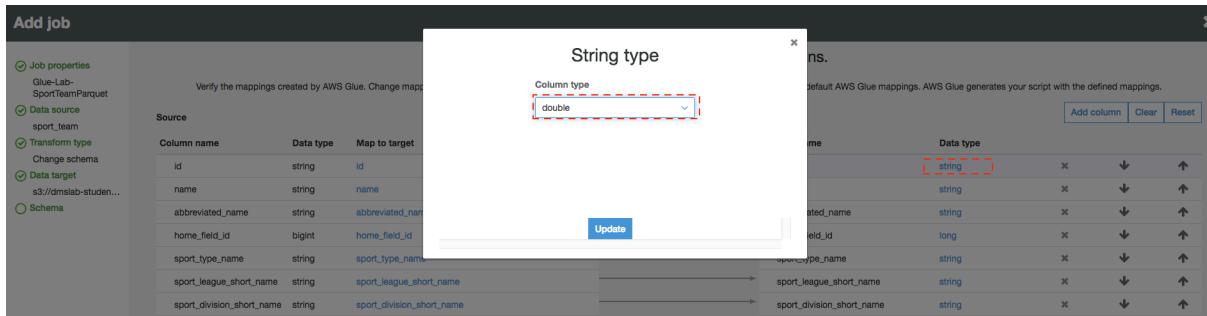
Data store
Amazon S3

Format
Parquet

Target path
s3://dmslab-student-dmslabs3bucket-xg1hdq80bs/ticke

[Back](#) [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**.

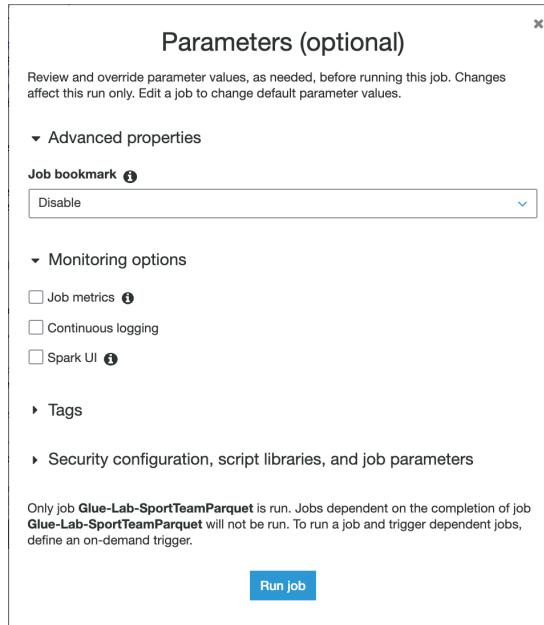
13. View the job. (This screen provides you with the ability to customize this script as required.) Click **Save** and then **Run Job**.

```

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.init(args['JOB_NAME'], args)
16 ## @type: DataSource
17 ## @return: datasource0
18 ## @type: DataSource
19 ## @return: datasource0
20 datasource0 = glueContext.create_dynamic_frame("ticketeddata", table_name = "sport_team", transformation_ctx = "datasource0")
21 ## @type: ApplyMapping
22 ## @return: applymapping1
23 ## @type: ApplyMapping
24 ## @return: applymapping1
25 ## @type: ApplyMapping
26 ## @return: applymapping1
27 ## @type: ResolveChoice
28 ## @return: resolvechoice1
29 ## @type: ResolveChoice
30 ## @return: resolvechoice1
31 ## @type: DropNullFields
32 ## @return: dropnullfields3
33 ## @return: dropnullfields3
34
    
```

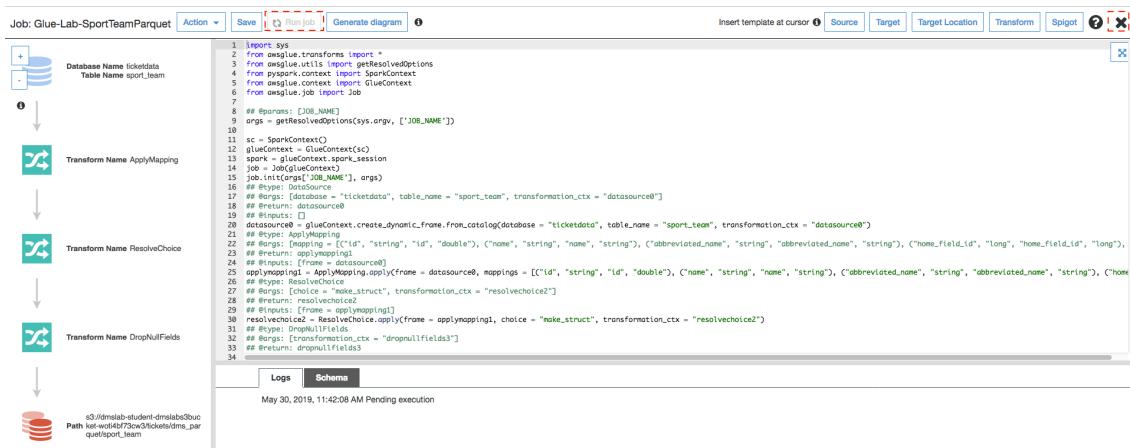
14. In **Parameters** option,

- 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.

The screenshot shows the AWS Glue Job Overview page. At the top, there are buttons for 'Add job' and 'Action' (with a dropdown menu), and a search bar labeled 'Filter by tags and attributes'. Below this, a table lists the job details: Name (Glue-Lab-SportTeamParquet), Type (Spark), ETL language (python), Script location (s3://aws-glue-scri...), Last modified (11 March 2020 3:17 PM UTC-7), and Job bookmark (Disable). Below the table are tabs for History, Details, Script, and Metrics. Under the Metrics tab, there are buttons for 'View run metrics' and 'Rewind job bookmark'. A second table below shows the current run details: Run ID (jr_e37b56aa03cd3...), Retry attempt (1), Run status (Running), Error (Logs), Logs (Error logs), Glue version (1.0), Maximum capacity (5), Triggered by (None), Start time (11 Mar...), End time (0 secs), Execution time (2880 mins), Timeout (N/A), Delay (N/A), and Job run input (s3://aws-glue-tem...).

You need to repeat the preceding steps to create new ETL Jobs to transform the additional tables. You will use transform 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?
Glue-Lab-SportLocationParquet	sport_location	dms_parquet/sport_location	No
Glue-Lab-SportingEventParquet	sporting_event	dms_parquet/sporting_event	Yes
Glue-Lab-SportingEventTicketParquet	sporting_event_ticket	dms_parquet/sporting_event_ticket	Yes
Glue-Lab-PersonParquet	person	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**.

The screenshot shows the AWS Glue interface with the 'Crawlers' section selected. A table lists one crawler: 'glue-lab-crawler'. The crawler is in a 'Ready' status, with 'Logs' and '1 min' for last runtime. It has processed 0 tables and added 15 tables.

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

The screenshot shows the 'Add crawler' configuration screen. Under 'Crawler info', the 'Name' field is filled with 'glue-lab-parquet-crawler'. The 'Crawler source type' section is expanded, showing 'Data stores' selected. The 'Data store' dropdown is set to 'S3'. The 'Next' button is visible at the bottom right.

3. In next screen **Specify crawler source type**, select **Data Source** as choice for **Crawler resource type** and click **Next**.

4. In Add a data store screen

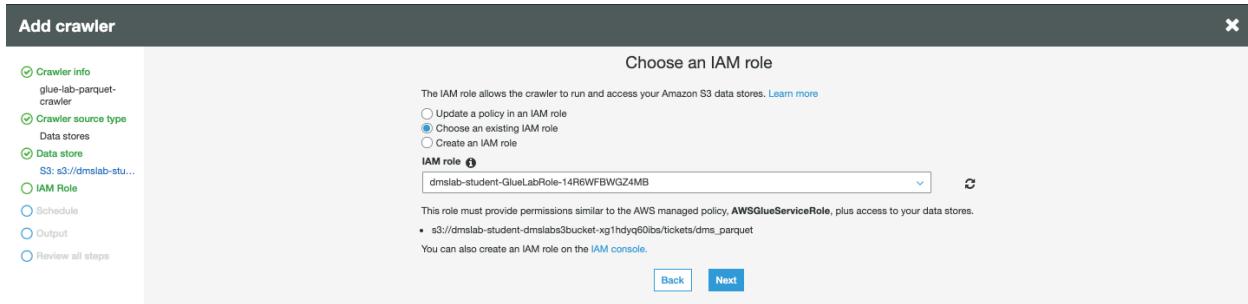
- a. For **Choose a data store**, select "S3".
- b. For **Crawl data in**, select "Specified path in account".
- c. For **Include path**, specify the S3 Path (Parent Parquet folder) that contains the nested parquet files e.g., s3://dmslab-student-dmslabs3bucket-woti4bf73cw3/tickets/dms_parquet
- d. Click **Next**.

The screenshot shows the 'Add a data store' configuration screen. Under 'Choose a data store', 'S3' is selected. Under 'Crawl data in', 'Specified path in my account' is selected. The 'Include path' field contains 's3://dmslab-student-dmslabs3bucket-woti4bf73cw3/tickets/dms_parquet'. The 'Exclude patterns (optional)' section is collapsed. At the bottom are 'Back' and 'Next' buttons.

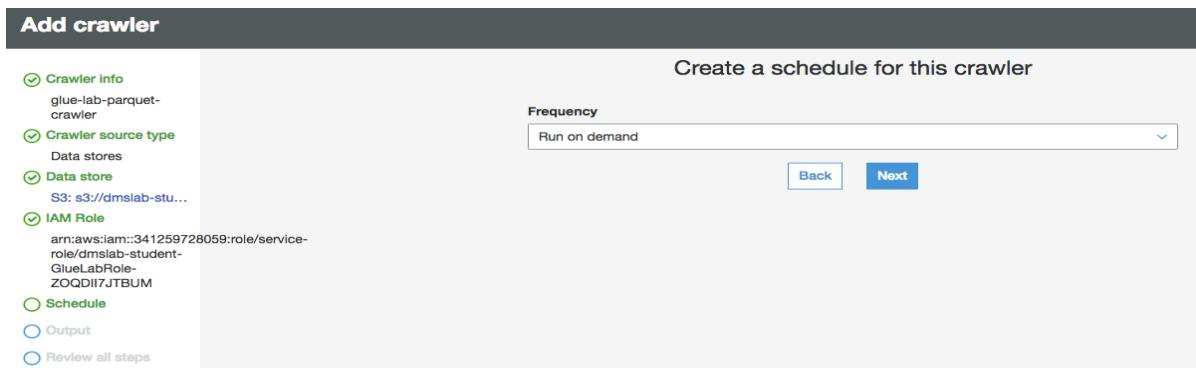
5. For Add another data store, select **No** and Click **Next**.

The screenshot shows the 'Add another data store' configuration screen. The 'Yes' radio button is unselected, and the 'No' radio button is selected. The 'Chosen data stores' section shows 'S3: s3://dmslab-stud...'. At the bottom are 'Back' and 'Next' buttons.

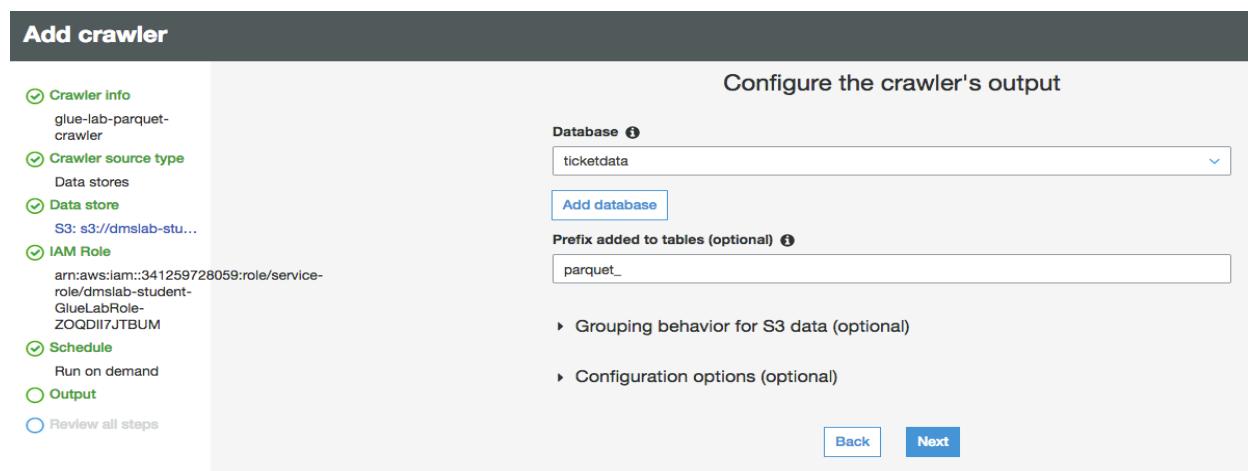
6. On the Choose an IAM role page, select **Choose an existing IAM role**.
For IAM role, select the existing role “dmslab-student-GlueLabRole-ZOQDII7JTBUM” and Click **Next**.



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



8. For the crawler’s output database, choose your existing database which you created earlier e.g. “ticket-data”
9. For the **Prefix added to tables** (optional), type “parquet_”



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

11. On the notification bar, click **Run it now.**

Once your crawler has finished running, you should report that 5 tables were added.

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

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.

Name	Database	Location	Classification	Last updated	Deprecated
mib_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: 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.

The screenshot shows the AWS Glue service interface. On the left, there is a sidebar with navigation links: AWS Glue, Data catalog, Databases, Tables, Connections, **Crawlers**, Classifiers, and Settings. The main area is titled 'Crawlers' and contains the following text: '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.' Below this is a search bar with 'Add crawler', 'Run crawler', 'Action', and a filter 'Filter by tags and attributes'. A message says 'Showing: 0 - 0' and includes 'User preferences' and 'Edit' buttons. A table header row is shown with columns: Name, Schedule, Status, Logs, Last runtime, Median runtime, Tables updated, and Tables added. A message at the bottom center says 'You don't have any crawlers yet.' with a 'Add crawler' button.

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. The title bar says 'Add crawler'. The left sidebar has steps: Crawler info (selected), Crawler source type, Data store, IAM Role, Schedule, Output, and Review all steps. The main area is titled 'Add information about your crawler'. It has a 'Crawler name' field containing 'glue-lab-cdc-crawler' and a note 'Tags, description, security configuration, and classifiers (optional)'. At the bottom right is a 'Next' button.

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

The screenshot shows the 'Add crawler' wizard Step 2. The title bar says 'Add crawler'. The left sidebar has steps: Crawler info, Crawler source type (selected), Data store, IAM Role, Schedule, Output, and Review all steps. The main area is titled 'Specify crawler source type'. It has 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.' and a 'Crawler source type' section with 'Data stores' (selected) and 'Existing catalog tables' options. At the bottom are 'Back' and 'Next' buttons.

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://dmslab-student-dmslabs3bucket-wot14bf73cw3/cdc/dms_sample"

7. Click **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**.

The screenshot shows the 'Add crawler' wizard at step 4: 'Create a schedule for this crawler'. The left sidebar lists configuration steps: Crawler info, Crawler source type, Data stores, S3, IAM Role, Schedule, Output, and Review all steps. The 'Schedule' step is selected. The main area shows a dropdown menu for 'Frequency' with the option 'Run on demand' chosen. At the bottom are 'Back' and 'Next' buttons.

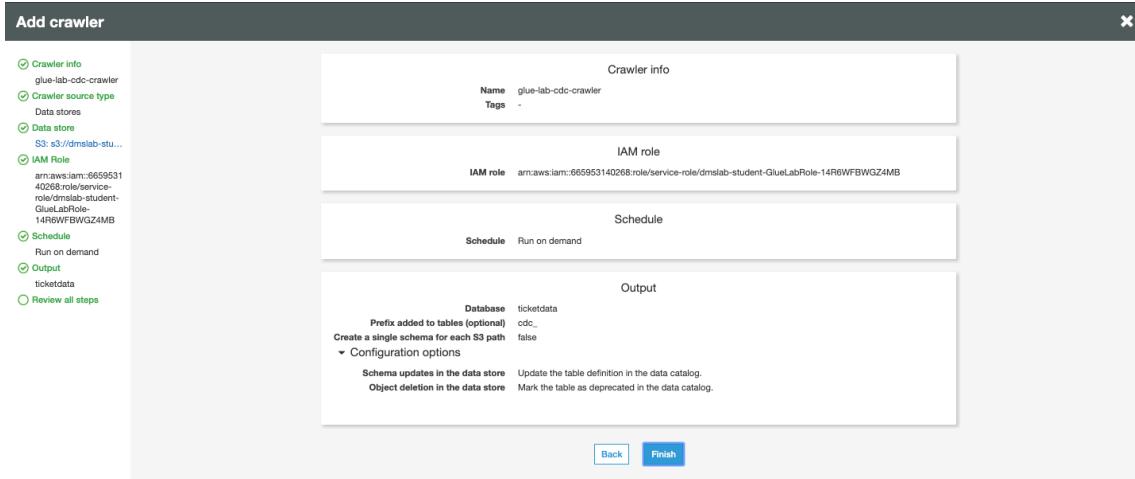
12. On the Configure the crawler's output page, select the existing **Database** for crawler output (e.g., "ticketdata").

13. For **Prefix added to tables (optional)**, specify "cdc_"

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

The screenshot shows the 'Add crawler' wizard at step 5: 'Configure the crawler's output'. The left sidebar lists configuration steps: Crawler info, Crawler source type, Data stores, S3, IAM Role, Schedule, Run on demand, Output, and Review all steps. The 'Output' step is selected. The main area shows a 'Database' dropdown set to 'ticketdata'. Below it is a 'Prefix added to tables (optional)' input field containing 'cdc_'. Under 'Configuration options (optional)', there are three radio buttons: 'Update the table definition in the data catalog.' (selected), 'Add new columns only.', and 'Ignore the change and don't update the table in the data catalog.'. Another section, 'How should AWS Glue handle deleted objects in the data store?', contains three radio buttons: 'Delete tables and partitions from the data catalog.' (selected), 'Ignore the change and don't update the table in the data catalog.', and 'Mark the table as deprecated in the data catalog.'. At the bottom are 'Back' and 'Next' buttons.

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



16. Click Run it now.

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

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	

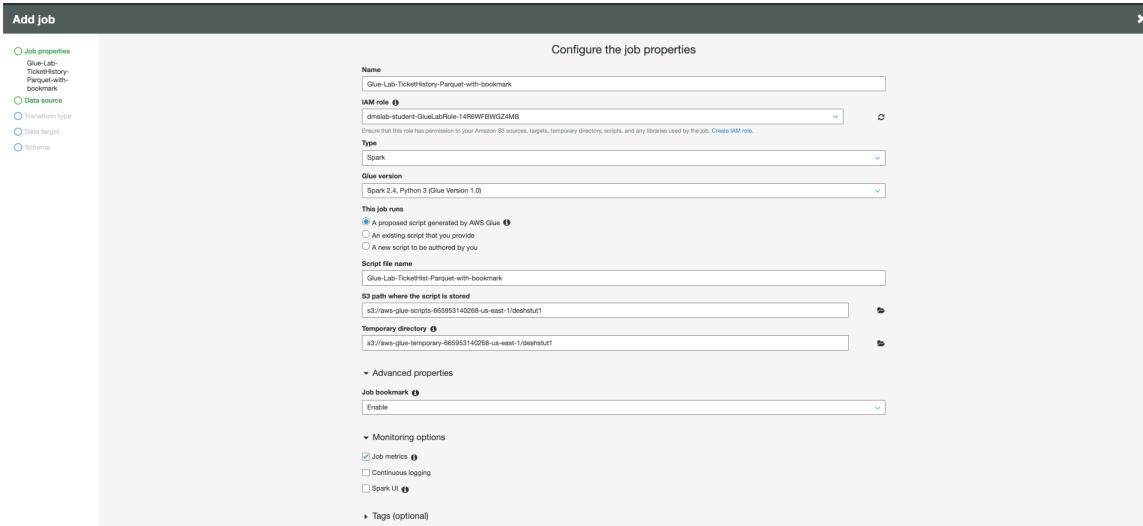
Step 2: Create a Glue Job with Bookmark Enabled

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



The screenshot shows the AWS Glue Jobs console. On the left, there's a sidebar with 'AWS Glue' at the top, followed by 'Data catalog', 'Databases', and 'Tables'. 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' dropdown, and a 'Filter by tags and attributes' search bar. A table header row includes columns for 'Name', 'Type', 'ETL language', 'Script location', 'Last modified', and 'Job bookmark'. At the bottom right of the table area, it says 'Showing: 1 - 9'.

2. On the Job properties page, make the following selections:
 - a. For **Name**, type **Glue-Lab-TicketHistory-Parquet-with-bookmark**.
 - b. For **IAM role**, choose existing role "dmslab-student-GlueLabRole-ZOQDI17JTBUM"
 - 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-TicketHistory-Parquet-with-bookmark**.
 - 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.)
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**



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

- Name:** Glue-Lab-TicketHistory-Parquet-with-bookmark
- IAM role:** dmslab-student-GlueLabRole-1496WFBWGDZAMB
- 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-TicketHistory-Parquet-with-bookmark
- S3 path where the script is stored:** s3://aws-glue-scripts-66595314268-us-east-1/deashut1
- Temporary directory:** s3://aws-glue-temporary-66595314268-us-east-1/deashut1
- Advanced properties:** Job bookmark (dropdown set to 'Enable')
- Monitoring options:** Job metrics (checkbox checked), Continuous logging (checkbox unchecked), Spark UI (checkbox unchecked)
- Tags (optional):** (empty field)

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:

- For Data store: select amazon S3
- Format: parquet
- Target path: s3://<dms-lab-bucket-path>/cdc_bookmark/ticket_purchase_history_data/
- Click Next

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

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

```

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 # Initialize job
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10 sc = SparkContext()
11 glueContext = GlueContext(sc)
12 spark = glueContext.sparkSession
13 job = Job(glueContext)
14 job.init(args['JOB_NAME'], args)
15
16 # Load ticketdata
17 # Merge: [database = "ticketdata", table_name = "cdc_ticket_purchase_hist", transformation_ctx = "datasource"]
18 # Inputs: []
19 # Outputs: [GlueContext.create_dynamic_frame.from_catalog(database = "ticketdata", table_name = "cdc_ticket_purchase_hist", transformation_ctx = "datasource")]
20
21 ## Phases: ApplyMapping
22 ## Merge: [mapping = ["op", "op", "string"], ("sporting.event.ticket_id", "string", "purchased_by_id", "string"), ("transaction_date_time", "string", "transaction_date_time", "string")]
23 ## Inputs: [frame = datasource]
24 ## Outputs: [frame = datasource]
25 ## Phases: ResolveChoice
26 ## Inputs: [make_struct, transformation_ctx = "resolvechoice"]
27 ## Outputs: resolvechoice2
28 ## Phases: DropNullFields
29 ## Inputs: [frame = applymapping]
30 ## Outputs: [DropNullFields3]
31 ## Phases: ResolveChoice
32 ## Inputs: [frame = applymapping, choice = "make_struct", transformation_ctx = "resolvechoice"]
33 ## Outputs: dropnullfields3
34 ## Phases: DataSink
35 ## Inputs: [frame = resolvechoice2]
36 ## Outputs: dropnullfields3
37 ## Phases: DataSink
38 ## Inputs: [connection_type = "s3", connection_options = {"path": "s3://dmslab-student-dmslabs3bucket-xg1hdq60ibs/cdc_bookmark/ticket_purchase_history/data"}, format = "parquet", transformation_ctx = "datasink4"]
39 ## Outputs: datasink4
40 ## Phases: DataSink
41 datasink4 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields3, connection_type = "s3", connection_options = {"path": "s3://dmslab-student-dmslabs3bucket-xg1hdq60ibs/cdc_bookmark/ticket_purchase_history/data"}, format = "parquet", transformation_ctx = "datasink4")
42 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-c000.snappy.parquet	Jan 24, 2020 7:03:16 PM GMT-0500	1.1 MB	Standard
part-00001-498ea7fc-2ac1-4787-b431-9e16f5e24a3f-c000.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**

- 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**

Specify crawler source type

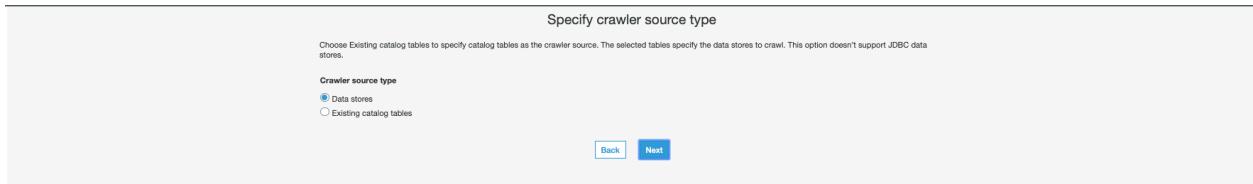
Choose Existing catalog tables to specify catalog tables as the crawler source. The selected tables specify the data stores to crawl. This option doesn't support JDBC data stores.

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

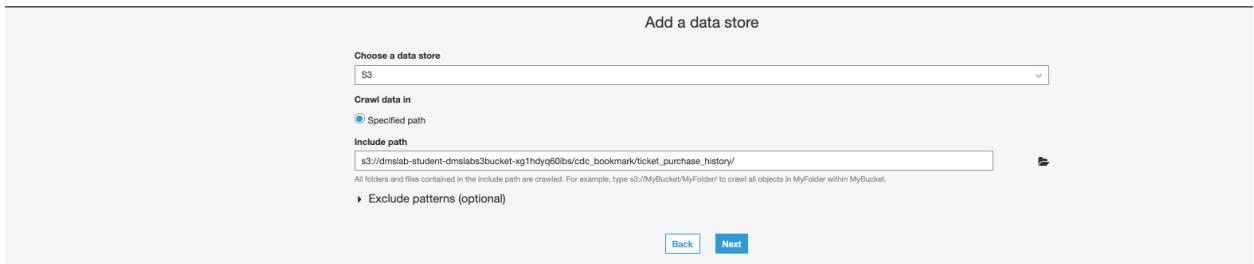
Include path

`s3://dmslab-student-dmslabs3bucket-xg1hydq60bs/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

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](#)

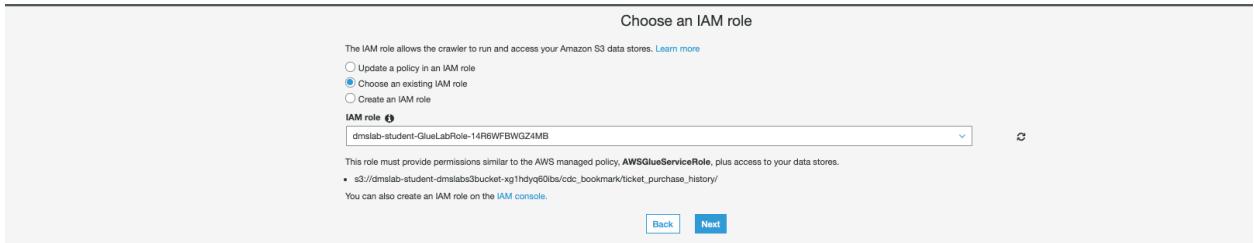
dmslab-student-GlueLabRole-14R6WFBWGZ4MB

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

s3://dmslab-student-dmslabs3bucket-xg1hydq60bs/cdc_bookmark/ticket_purchase_history/

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

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



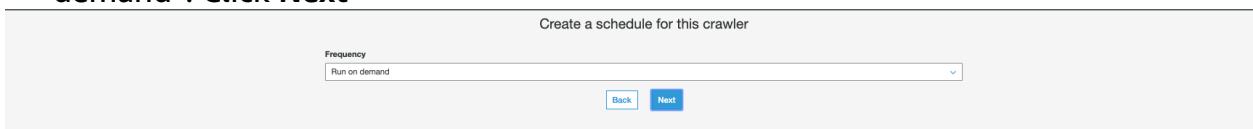
9. For setting the **frequency** in create a schedule for this crawler, select "Run on demand". Click **Next**

Create a schedule for this crawler

Frequency

Run on demand

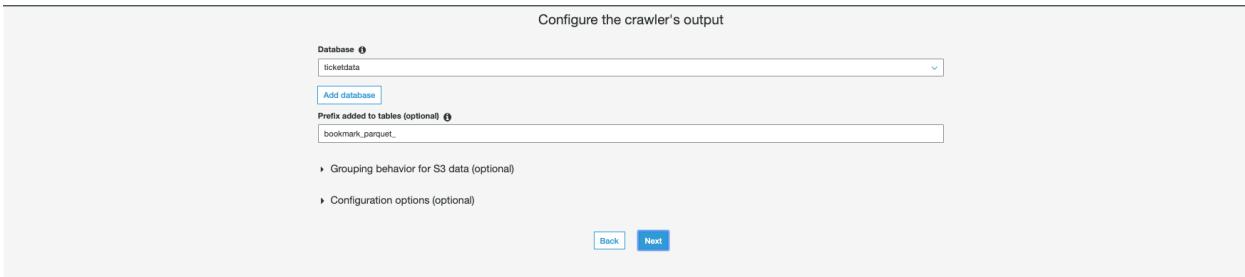
[Back](#) [Next](#)



10. For the crawler's output:

- a. For Database, select "ticket" database.
- b. Optionally, add prefix to the newly created tables for easy identification. Provide the prefix as "**bookmark_parquet_**"

c. Click Next



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

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.

The screenshot shows the AWS Athena Query Editor interface. On the left, the Data source dropdown is set to 'awsdatasatalog' and the Database dropdown is set to 'processed-data'. Under 'Tables (4)', the 'ticketdata' table is selected. The 'Views (0)' section indicates no views have been created. In the main area, a new query window titled 'New query 1' contains the SQL command: 'SELECT * FROM `ticketdata`.`bookmark_parquet_ticket_purchase_history` limit 10;'. Below it, another window titled 'New query 2' also contains the same SQL command. The results pane shows 10 rows of data from the table, with columns including op, sporting_event_ticket_id, purchased_by_id, transaction_date_time, transferred_from_id, purchase_price, and partition_0. The data is timestamped between 2020-01-23 20:27:49 and 2020-01-23 20:27:49, and purchase prices range from 37.72 to 37.72.

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

The screenshot shows the AWS Glue Table view for the 'bookmark_parquet_ticket_purchase_history' table. The table properties include: Name: bookmark_parquet_ticket_purchase_history, Description: ticketdata, Database: ticketdata, Classification: parquet, Location: s3://mramab-student-dmsdb.s3bucket.ql1dygb0s.vdr, Last updated: Sat Jan 25 2020 04:54:00, Input format: org.apache.hadoop.mapred.TextInputFormat, Output format: org.apache.hadoop.mapred.TextOutputFormat, Serde serialization lib: org.apache.hadoop.hive.oio.parquet.MapredParquetHiveSerDe, and Serde parameters: serialization.format: 1. The recordCount is highlighted with a yellow box and shows the value 1521922. Other properties listed include sizeKey, objectCount, updated_by_crawler, crawlerSchemaDeserializerVersion, and compressionType.

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.

Amazon S3 > dmslab-student-dmslabs3bucket-kg1hdyyq60bs > cdc_bookmark > ticket_purchase_history > data

Overview

Upload + Create folder Download Actions

Name * Size * Last modified * Storage class *

part-00000-002020b4-2fac-47c2-8249-d0100b7ddad-c000.snappy.parquet	9.3 KB	Jan 24, 2020 9:20:13 PM GMT-0500	Standard
part-00000-49bea7fc-2ac1-4787-b431-9e1f5e2fa2f-c000.snappy.parquet	1.1 MB	Jan 24, 2020 7:03:16 PM GMT-0500	Standard
part-00000-d1666723-315b-45fb-b8fe-a66f2384b234-c000.snappy.parquet	1.7 MB	Jan 25, 2020 11:24:20 PM GMT-0500	Standard
part-00000-e8cc00f1-d440-43c0-9230-c3cd2151ba-c000.snappy.parquet	7.2 KB	Jan 25, 2020 10:24:27 PM GMT-0500	Standard
part-00000-002020b4-2fac-47c2-8249-d0100b7ddad-c000.snappy.parquet	66.5 KB	Jan 24, 2020 9:20:13 PM GMT-0500	Standard
part-00000-49bea7fc-2ac1-4787-b431-9e1f5e2fa2f-c000.snappy.parquet	1.2 MB	Jan 20, 2020 7:03:16 PM GMT-0500	Standard
part-00001-d1666723-315b-45fb-b8fe-a66f2384b234-c000.snappy.parquet	1.7 MB	Jan 25, 2020 11:24:20 PM GMT-0500	Standard
part-00000-002020b4-2fac-47c2-8249-d0100b7ddad-c000.snappy.parquet	1.7 MB	Jan 24, 2020 9:20:15 PM GMT-0500	Standard
part-00000-d1666723-315b-45fb-b8fe-a66f2384b234-c000.snappy.parquet	1.5 MB	Jan 25, 2020 11:24:19 PM GMT-0500	Standard

- 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**:

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

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

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

AWS Glue

Data catalog

Tables

bookmarks_parquet_ticket_purchase_history

Name: bookmarks_parquet_ticket_purchase_history

Description: ticketdata

Database: parquet

Location: s3://dmslab-student-dmslabs3bucket-kg1hdyyq60bs/ticketdata/bookmarks/ticket_purchase_history

Connection:

Deprecated: No

Last updated: Sun Jan 26 00:00:56 GMT-0500 2020

Input format: org.apache.hadoop.hive.serde2.parquetserde.ParquetInputFormat

Output format: org.apache.hadoop.hive.serde2.parquetserde.ParquetOutputFormat

Serde parameters: serdeVersion=1, inputFormatClass=org.apache.hadoop.hive.serde2.parquetserde.ParquetHiveSerDe, outputFormatClass=org.apache.hadoop.hive.serde2.parquetserde.ParquetHiveSerDe

Serde parameters: serialization.format: 1

Table properties: sizeKey: 11217289 objectCount: 11 UPDATED_BY_CRAWLER: glue-lab-cdc_bookmark_crawler CrawlerSchemaDeserializerVersion: 1.0 recordCount: 1813493 averageRecordSize: 27 CrawlerSchemaDeserializerVersion: 1.0 compressionType: none typeOfData: file

Schema

Column name	Data type	Partition key	Comment
cp	string		
spotted_event_id	string		
purchase_id	string		
transaction_date_time	string		
transformed_from_id	string		
purchase_price	double		
partition_0	string	Partition 0	

- 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.

AWS Glue

Data catalog

Tables

bookmarks_parquet_ticket_purchase_history

Action

View data

- On Athena Console, run the following query:

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

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

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.

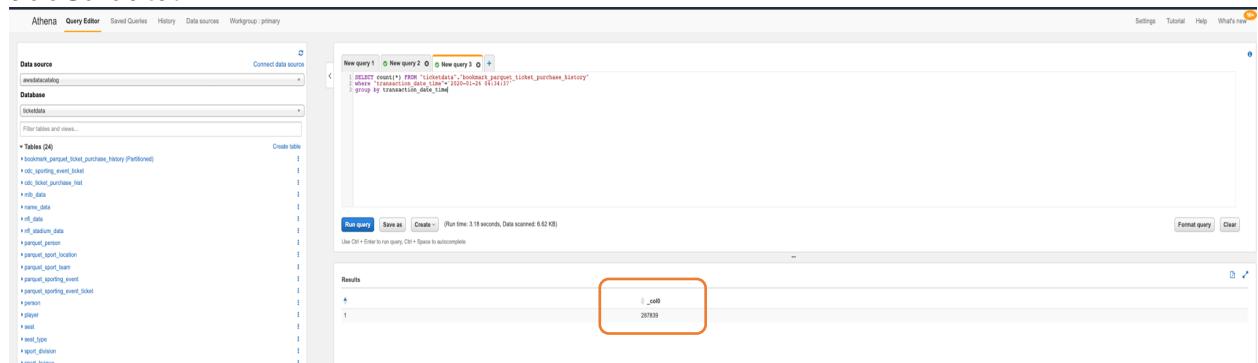
Now let's make sure that only the new data was scanned and added to the destination table by our **Glue Job with Bookmark enabled**. To validate the functioning of bookmark, timestamp values are used, in order to make sure only newly added data is scanned and added. In this example,

"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.



The screenshot shows the AWS Athena Query Editor interface. On the left, the Data source dropdown is set to 'awsdatasql'. The main area contains a query editor with the following SQL code:

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

Below the query editor, the results section displays a single row of data:

count(*)
207839

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**.

2. Give the workflow name as “**Workflow_MLB_Data**”. 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.

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.

Add trigger

Clone existing Add new

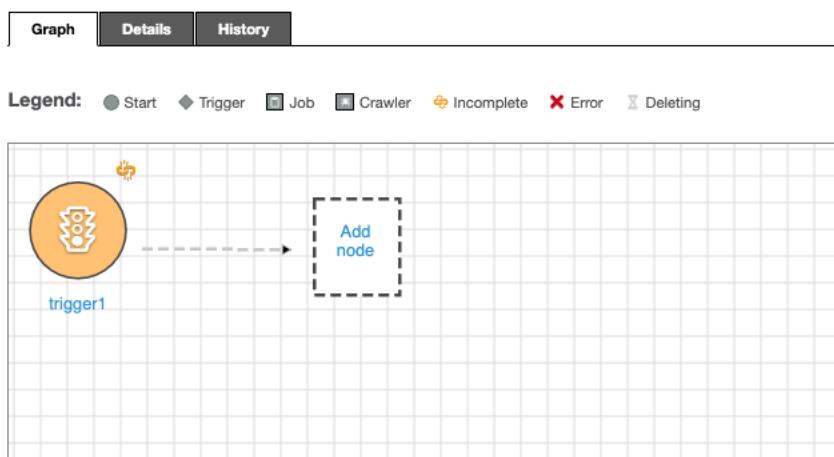
Name
trigger1

Description (optional)
Trigger to start the workflow

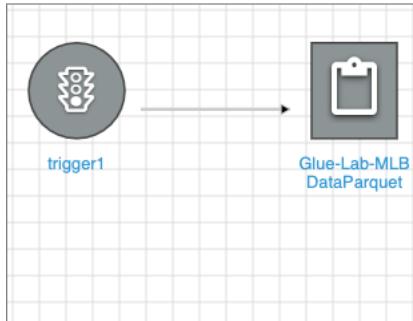
Trigger type
 Schedule Event On demand

Cancel Add

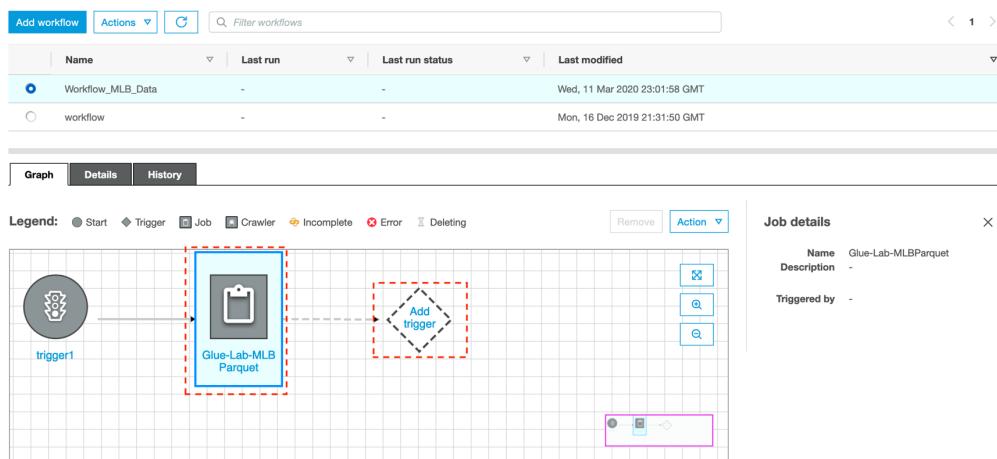
5. Click on **trigger1** to add a **new node**. New Node can be a crawler or job, depending upon the workflow you want to build. Make sure you add node only after clicking on trigger so that whenever the "trigger1" is activated, it triggers job/crawler specified in the new node.



6. 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**

Add trigger

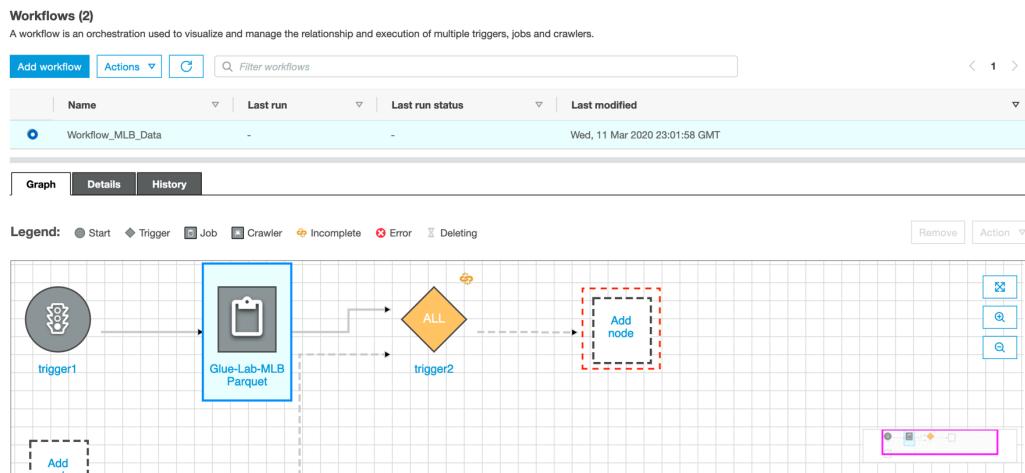
Name
trigger2

Description (optional)
Trigger to execute crawler

Trigger type
 Schedule Event On demand

Trigger logic
 Start after ANY watched event Start after ALL watched event

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



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

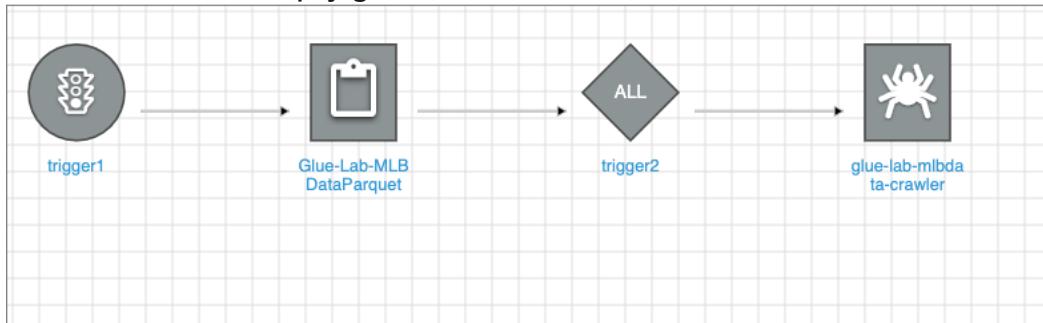
Add job(s) and crawler(s) to trigger

Crawlers

	Name	Description
<input type="checkbox"/>	glue-lab-cdc-crawler	
<input type="checkbox"/>	glue-lab-crawler	DMS Lab full initial load crawler
<input checked="" type="checkbox"/>	glue-lab-mlbdata-crawler	
<input type="checkbox"/>	glue-lab-parquet-crawler	
<input type="checkbox"/>	glue_lab_cdc_bookmark_crawler	
<input type="checkbox"/>	kdg-crawler	
<input type="checkbox"/>	processed-crawler	
<input type="checkbox"/>	rawdataset-crawler	

Cancel 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"

Name	Last run	Last run status	Last modified
Workflow1	-	-	Sun, 26 Jan 2020 05:12:03 GMT

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://ave-gue-scripts-669953140288-us-e...	25 January 2020 10:21 PM UTC-5	Disable
Spark	python	s3://ave-gue-scripts-669953140288-us-e...	10 January 2020 3:09 PM UTC-5	Disable
Spark	python	s3://ave-gue-scripts-669953140288-us-e...	10 January 2020 3:01 PM UTC-5	Disable
Spark	python	s3://ave-gue-scripts-669953140288-us-e...	10 January 2020 2:24 PM UTC-5	Disable
Spark	python	s3://ave-gue-scripts-669953140288-us-e...	10 January 2020 2:04 PM UTC-5	Disable
Spark	python	s3://ave-gue-scripts-669953140288-us-e...	10 January 2020 2:07 PM UTC-5	Disable
Spark	python	s3://ave-gue-scripts-669953140288-us-e...	24 January 2020 5:40 PM UTC-5	Disable
Spark	python	s3://ave-gue-scripts-669953140288-us-e...	24 January 2020 6:1 PM UTC-5	Enable
Spark	python	s3://ave-gue-scripts-669953140288-us-e...	24 January 2020 6:1 PM UTC-5	Enable
Spark	python	s3://ave-gue-scripts-669953140288-us-e...	15 December 2019 11:16 PM UTC-5	Disable
Spark	python	s3://ave-gue-scripts-669953140288-us-e...	15 December 2019 11:24 PM UTC-5	Disable

Glue Crawler

Schedule	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
	Ready	Logs	1 min	1 min	2	0
	Ready	Logs	1 min	1 min	0	15
	Ready	Logs	1 min	1 min	0	1
	Ready	Logs	57 secs	57 secs	0	5
	Ready	Logs	1 min	1 min	1	0
	Ready	Logs	1 min	1 min	0	1

AWS Glue

Name	Database	Location	Classification	Last updated	Deprecated
bookman_parquet_ticket_purchase_history	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/cdc_bookmark..._parquet	parquet	26 January 2020 12:09 PM UTC-5	
cdc_sporting_event_ticket	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/cdc_sportin...	csv	25 January 2020 11:42 PM UTC-5	
cdc_ticket_purchaser_hist	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/cdc_ticket_p...	csv	25 January 2020 11:42 PM UTC-5	
intf_data	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
name_data	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
rf_data	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
rf_statusn_data	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
parquet_person	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	parquet	23 January 2020 1:49 PM UTC-5	
parquet_sport_location	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	parquet	23 January 2020 1:49 PM UTC-5	
parquet_sport_team	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	parquet	23 January 2020 1:49 PM UTC-5	
parquet_sporting_event	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	parquet	23 January 2020 1:49 PM UTC-5	
parquet_sporting_event	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	parquet	23 January 2020 1:49 PM UTC-5	
person	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:48 PM UTC-5	
seat	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
seat_type	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
sport_division	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
sport_league	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
sport_location	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
sport_team	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:44 PM UTC-5	
sporting_event	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
sporting_event_ticket	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
ticket_purchase_hist	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	csv	10 January 2020 1:37 PM UTC-5	
workflow_mlb_data	S3nData	s3://mlsde-student-dmslab2bucket-nghyq0b1s0/intf/intf_sampl...	parquet	25 January 2020 10:38 PM UTC-5	

Table Created

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

New query 1 | New query 2 | New query 3 | (Run time: 2.92 seconds, Data scanned: 336.53 KB)

1. SELECT * FROM "ticketdata"."workflow_mlb_data" limit 10;

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	rbc_id	rbc_n	
1	+5.056000000000e+05	Alexi Amarista	3B	SD	San Diego Padres	L	R	1989	-5.588000000000e+04	amarista01	Alexi Amarista	1730553	Alexi Amarista	2B	+3.062000000000e+04	Alexi Amarista	2B	9063	Alexi Amarista	amarista01	+8.914000000000e+03	Alexi Amarista
2	+4.582100000000e+05	Alex Casilla	2B	TB	Tampa Bay Rays	S	R	1984	+4.579900000000e+04	casilla01	Alex Casilla	1103724	Alex Casilla	3B	+2.857500000000e+04	Alex Casilla	3B	5248	Alex Casilla	casilla01	+7.855000000000e+03	Alex Casilla
3	+4.853600000000e+05	Alex Ogando	P	ATL	Atlanta Braves	R	R	1983	+4.591000000000e+04	ogando01	Alex Ogando	1174286	Alex Ogando	RP	+3.054900000000e+04	Alex Ogando	RP	10261	Alex Ogando	ogando01	+8.743000000000e+03	Alex Ogando
4	+4.696800000000e+05	Alfredo Aceves	P	NYY	New York Yankees	R	R	1982	+4.692800000000e+04	aceves01	Alfredo Aceves	163980	Alfredo Aceves	RP	+2.922300000000e+04	Alfredo Aceves	RP	5164	Alfredo Aceves	aceves01	+8.396000000000e+03	Alfredo Aceves
5	+4.516200000000e+05	Alfredo Figaro	P	TEX	Texas Rangers	R	R	1984	+5.245300000000e+04	figaro01	Alfredo Figaro	165434	Alfredo Figaro	RP	+3.067000000000e+04	Alfredo Figaro	RP	8404	Alfredo Figaro	figaro01	+8.513000000000e+03	Alfredo Figaro
6	+5.540540000000e+05	Alfredo Gonzalez	C	CWS	Chicago White Sox	R	R	1992	+6.920100000000e+04	gonzalez01	Alfredo Gonzalez	2210083	Alfredo Gonzalez	C	+3.484400000000e+04	Alfredo Gonzalez	C	sa505301	Alfredo Gonzalez	gonzalez01	+8.513000000000e+03	Alfredo Gonzalez
7	+5.012450000000e+05	Alfredo Marte	LF	BAL	Baltimore Orioles	R	R	1989	+5.175500000000e+04	marteal01	Alfredo Marte	1956711	Alfredo Marte	LF	+3.225100000000e+04	Alfredo Marte	LF	5212	Alfredo Marte	marteal01	+9.354000000000e+03	Alfredo Marte
8	+4.305600000000e+05	Alfredo Simon	P	CIN	Cincinnati Reds	R	R	1981	+4.558100000000e+04	simon01	Alfredo Simon	448968	Alfredo Simon	SP	+2.869900000000e+04	Alfredo Simon	SP	2155	Alfredo Simon	simon01	+7.975000000000e+03	Alfredo Simon
9	+4.553790000000e+05	Ali Sols	C	LAD	Los Angeles Dodgers	R	R	1987	+5.236200000000e+04	solsal01	Ali Sols	1946524	Ali Sols	C	+3.211400000000e+04	Ali Sols	C	8848	Ali Sols	solsal01	+9.340000000000e+03	Ali Sols
10	+4.888520000000e+05	Allan Dykstra	1B	TB	Tampa Bay Rays	L	R	1987	+5.781300000000e+04	dykstra01	Allan Dykstra	1960245	Allan Dykstra	1B	+3.236100000000e+04	Allan Dykstra	1B	9113	Allan Dykstra	dykstra01	+9.822000000000e+03	Allan Dykstra

Congratulations!! You have successfully completed this lab