

HOW TO JOIN TABLES IN AMAZON GLUE



Here we show how to join two tables in Amazon Glue. We make a crawler and then write Python code to create a Glue Dynamic Dataframe to join the two tables.

First, we'll share some information on how joins work in Glue, then we'll move onto the tutorial. You can start with the [basics on Amazon Glue Crawlers](#), but we are going to modify the procedure described there to fit the data we have prepared below.

Brief intro to Amazon Glue

Glue is not a database. It basically contains nothing but metadata. You point it at a data source and it vacuums up the schema. Or you create the schema manually. The data exists in

- S3
- [A SQL database](#)
- [DynamoDB](#)

Glue processes data sets using [Apache Spark](#), which is an in-memory database. Then you can write the resulting data out to S3 or mysql, PostgreSQL, Amazon Redshift, SQL Server, or Oracle.

Glue can crawl these data types:

- JSON
- CSV

- Parquet
- Avro
- XML

What is a join?

First, **join** means to take two tables and join them by a common element. Joining two tables is an important step in lots of [ETL operations](#).

A join is a SQL operation that you could not perform on most noSQL databases, like DynamoDB or [MongoDB](#). noSQL databases don't usually allow joins because it is an expensive operation that takes a lot of time, disk space, and memory.

Amazon Glue joins

Glue does the joins using Apache Spark, which runs in memory.

In this example, it pulls JSON data from S3 and uses the metadata schema created by the crawler to identify the attributes in the files so that it can work with those.

Set up Amazon Glue Crawler in S3 to get sample data

We will use a small subset of the IMDB database (just seven records). We have converted the data to JSON format and put in on S3. First check that you can open these files by opening one of each of these:

- [Ratings](#)
- [Titles](#)

The movie **titles** look like this:

```
{"tconst": "tt0276132", "titleType": "movie", "primaryTitle": "The  
Fetishist", "originalTitle": "The Fetishist", "isAdult": "0", "startYear":  
"2019", "endYear": "\\N", "runtimeMinutes": "\\N", "genres": "Animation"}
```

The **ratings** look like this:

```
{"tconst": "tt0305295", "averageRating": "6.1", "numVotes": "16"}
```

The goal is to rate movies and TV shows. We have to do that with a join operation since the rating and the title are in separate datasets. The common element is the unique element **tconst**.

Set up a crawler in Amazon Glue and crawl these two folders:

- s3://walkerimdb/ratings
- s3://movieswalker/

Make sure you select **Create Single Schema** so that it makes just one table for each S3 folder and not one for each file.

Start Amazon Glue Virtual Machine

Glue is nothing more than a virtual machine running Spark and Glue. We are using it here using the Glue PySpark CLI. PySpark is the Spark Python shell. You can also attach a [Zeppelin notebook](#) to it or perform limited operations on the web site, like creating the database. And you can use Scala.

Glue supports two languages: Scala and [Python](#). That's because it rides a top Apache Spark, which supports those two languages as well—and, for the most part, only those two. Glue is basically an Apache Spark instance with Glue libraries attached.

Set up The Development Endpoint

Next, set a billing alarm in your Amazon AWS account. When you start an [endpoint](#), you will incur charges from Amazon, since it's a [virtual machine](#). (You can download Glue and use it on a local machine if you don't want to incur charges. But then you can't use the GUI.)

Fill out these screens from the Glue console as follows. You will have to create a new public key in order to access the Glue VM from ssh. You cannot use the root Amazon credentials.

Set up your development endpoint

Use a development endpoint to set up an AWS Glue environment that you can use to develop your scripts with other software, such as notebooks, integrated development environments (IDEs), and a REPL shell.

Development endpoint name

IAM role ⓘ





Ensure that this role has permission to your Amazon S3 sources, targets, and any libraries used by the development endpoint. [Create IAM role.](#)

- ▶ Security configuration, script libraries, and job parameters (optional)
- ▶ Tags (optional)
- ▶ Monitoring options (optional)
- ▶ Catalog options (optional)

Networking

How do you want to provide the network information used to create the development endpoint?

When required, this information determines which VPC, subnet, and security groups are shared with your data stores and the development endpoint. The development endpoint uses this networking information to connect to your data stores in an AWS Glue environment.

☒ Skip networking information

Networking information is optional if you only connect to S3 data stores.

☐ Choose a connection

AWS Glue obtains the VPC, subnet, and security groups from the connection.

☐ Choose a VPC, subnet, and security groups

AWS Glue presents you with VPCs, subnets, and security groups that can be used to create the development endpoint.

Back

Next

Billing for a development endpoint is based on the Data Processing Unit (DPU) hours used during the **entire** time it remains in the **READY** state. To stop charges, delete the endpoint. To delete, choose the endpoint in the list, and then choose **Action, Delete**. [Learn more](#)

Development endpoint **BMCTutorial** is provisioning. When ready, create a notebook server at this endpoint to test AWS Glue scripts. To create a notebook server, choose the endpoint, then choose **Action, Create notebook server**. [Learn more](#)

Add endpoint

Action ▾

Filter by tags

Showing: 1 - 1

☐ Endpoint name	Provisioning status	Running for
☐ BMCTutorial	PROVISIONING	

Once the endpoint is created you change the path to point to your public key and open the shell using the URL Amazon gave you using ssh:

```
ssh -i /home/ubuntu/.ssh/glue glue@ec2-15-236-145-246.eu-west-3.compute.amazonaws.com -t gluepyspark3
```

That will open PySpark, which will be familiar to those who have used Apache Spark.

```
Python 3.6.11 (default, Jul 20 2020, 22:15:17)
```

```
on linux
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
SLF4J: Class path contains multiple SLF4J bindings.
```

```
SLF4J: Found binding in
```

```
SLF4J: Found binding in
```

```
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
```

```
SLF4J: Actual binding is of type
```

```
Setting default log level to "WARN".
```

```
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
```

```
2020-08-06 10:03:08,828 WARN yarn.Client (Logging.scala:logWarning(66)) - Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading libraries under SPARK_HOME.
```

Use Python code to join the tables

The code below is [here](#). Basically, the code creates two Glue Dynamic Frames. Then it creates a Spark Dataframe. Then we use the Join function to connect the two on the common element **tconst**.

The first step in an Apache Spark program is to get a **SparkContext**, meaning connect to an instance of Spark:

```
glueContext = GlueContext(SparkContext.getOrCreate())
```

Next we create Dynamic Dataframes. Those are Glue objects that don't exist in Spark.

The database you created manually in the GUI. The crawler created the table names with the same name as the S3 buckets.

```
titles =  
glueContext.create_dynamic_frame.from_catalog(database="moviesandratings",  
table_name="movieswalker")
```

```
ratings =  
glueContext.create_dynamic_frame.from_catalog(database="moviesandratings",  
table_name="walkerimdbratings")
```

Now we create a new Dynamic Dataframe using the Join object. You put the names of the two Dataframes to join and their common attributes, i.e., primary key field.

```
ratingsTitles = Join.apply(titles, ratings, 'tconst','tconst')
```

Then we convert that to a Spark Dataframe with toDF() so that we can use the select() method to pick the title and rating from the joined data.

```
ratingsTitles.toDF().select().show()
```

The result is:

```
+-----+-----+  
|      originalTitle|averageRating|  
+-----+-----+  
|Motherless Brooklyn|          6.8|  
|      Carnival Row|          7.9|  
|      Cine Manifest|          7.2|  
|      Pet Sematary|          5.7|  
|      The Dirt|          7.0|  
|      Dirt Music|          5.3|  
|      Bich bozhiy|          6.1|  
+-----+-----+
```

The complete code

Here is the complete code:

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import *

from awsglue.transforms import Join

glueContext = GlueContext(SparkContext.getOrCreate())

titles =
glueContext.create_dynamic_frame.from_catalog(database="moviesandratings",
table_name="movieswalker")

ratings =
glueContext.create_dynamic_frame.from_catalog(database="moviesandratings",
table_name="walkerimdbratings")

ratingsTitles = Join.apply(titles, ratings, 'tconst','tconst')
ratingsTitles.toDF().select().show()
```

Additional resources

Explore these resources:

- [BMC Machine Learning & Big Data Blog](#)
- [Apache Spark Guide](#)
- [AWS Guide](#)