Partition Data in S3 by Date from the Input File Name using AWS Glue

Tuesday, August 06, 2019 by Ujjwal Bhardwaj

Partitioning is an important technique for organizing datasets so they can be queried efficiently. It organizes data in a hierarchical directory structure based on the distinct values of one or more columns.

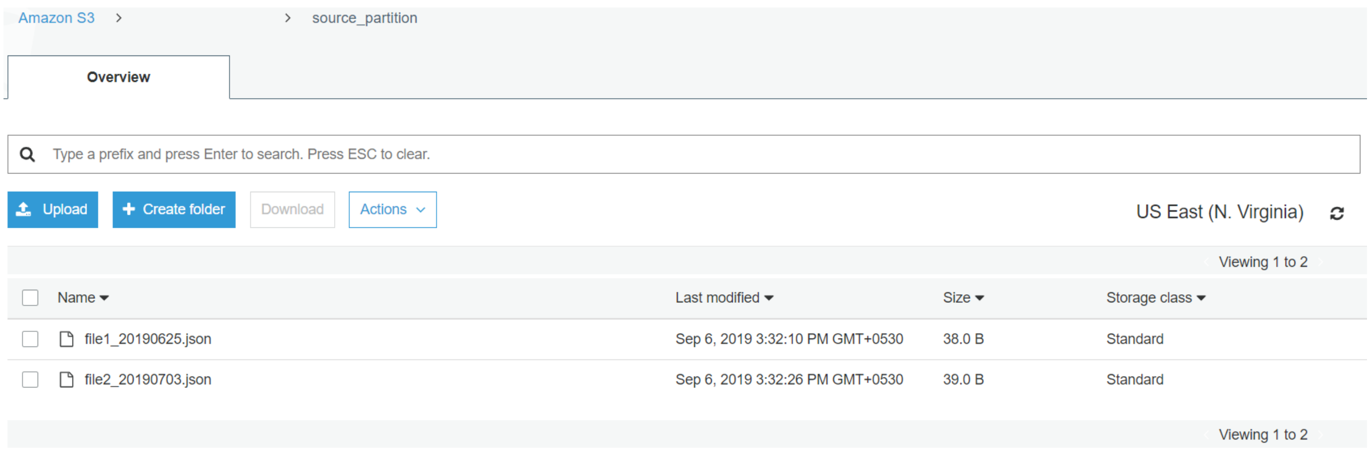
By default, a DynamicFrame is not partitioned when it is written and all the output files are written at the top level of the specified output path. However, DynamicFrames support native partitioning using a sequence of keys, using the partitionKeys option when you create a sink. From there, you can process these partitions using other systems, such as Amazon Athena.

There are data lakes where the data is stored in flat files with the file names containing the creation datetime of the data. These files are generally stored in a single level and thus have a lesser query performance as compared to a properly partitioned data. In a general consensus, the files are structured in a partition by the date of their creation. In S3, we find the files stored in the below format

```
s3://<bucket-name>/datastore/year=2019/month=01/day=01/
s3://<bucket-name>/datastore/year=2019/month=02/day=01/
s3://<bucket-name>/datastore/year=2019/month=03/day=01/
....
....
```

The above can be achieved with the help of Glue ETL job that can read the date from the input filename and then partition by the date after splitting it into year, month, and day.

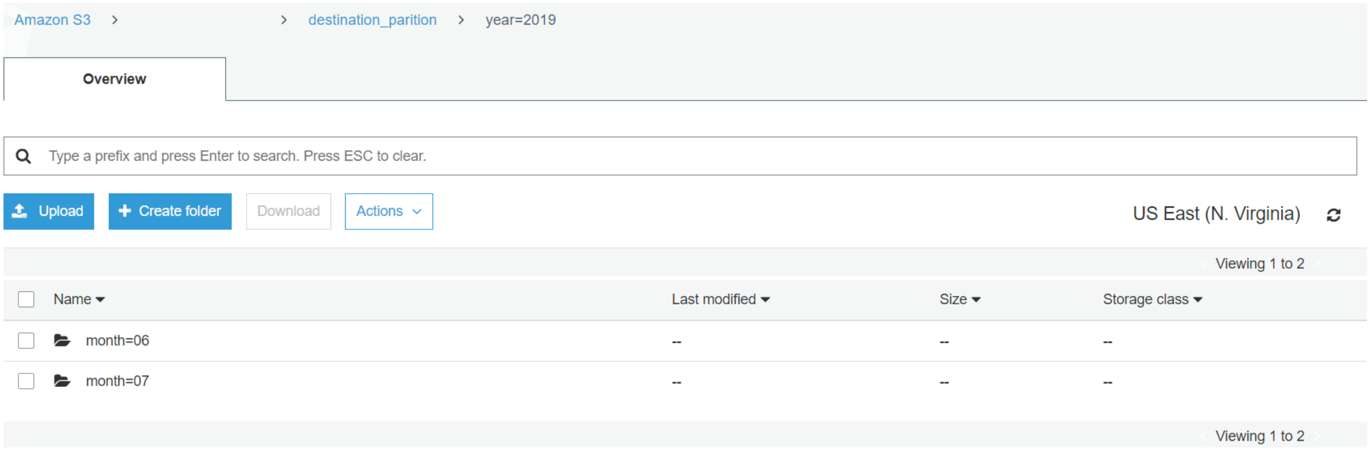
The below script paritions the dataset with the filename of the format `<filename>_YYYYMMDD.json` and then stores it in the Parquet format.


Amazon S3 > > source_partition
Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
Download
Actions
US East (N. Virginia)
Viewing 1 to 2

Name	Last modified	Size	Storage class
file1_20190625.json	Sep 6, 2019 3:32:10 PM GMT+0530	38.0 B	Standard
file2_20190703.json	Sep 6, 2019 3:32:26 PM GMT+0530	39.0 B	Standard

Viewing 1 to 2

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.dynamicframe import DynamicFrame
from pyspark.sql.functions import input_file_name

args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

datasource0 = glueContext.create_dynamic_frame.from_catalog(database =

# Create a DataFrame and add a new column in the containing the file na
# Refer `https://ujjwalbhardwaj.me/post/capture-the-input-file-name-in-
dataframe1 = datasource0.toDF().withColumn("filename", input_file_name(

# Convert the DataFrame back to DynamicFrame
dynamicframe2 = DynamicFrame.fromDF(dataframe1, glueContext, "dynamicfr

# Fetch the date of every DynamicRecord and create new column 'year', '
def map_function(dynamicRecord):
    date = dynamicRecord["filename"].split(".")[0][-8:]
    dynamicRecord["year"] = date[0:4]
    dynamicRecord["month"] = date[4:6]
    dynamicRecord["day"]= date[6:8]
    return dynamicRecord
# Apply the function to all the DynamicRecord
mapping3 = Map.apply(frame = dynamicframe2, f = map_function, transform

# Drop the input file name column
applymapping4 = ApplyMapping.apply(frame = mapping3, mappings = [("name

# Write the transformed dataset to S3 with Paritioning
datasink5 = glueContext.write_dynamic_frame.from_options(frame = applym
job.commit()
```


Amazon S3 > > destination_parition > year=2019

Overview

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

Upload | Create folder | Download | Actions

US East (N. Virginia)

Viewing 1 to 2

Name	Last modified	Size	Storage class
month=06	--	--	--
month=07	--	--	--

Viewing 1 to 2

0 Comments Sort by Oldest

Add a comment...

Facebook Comments Plugin