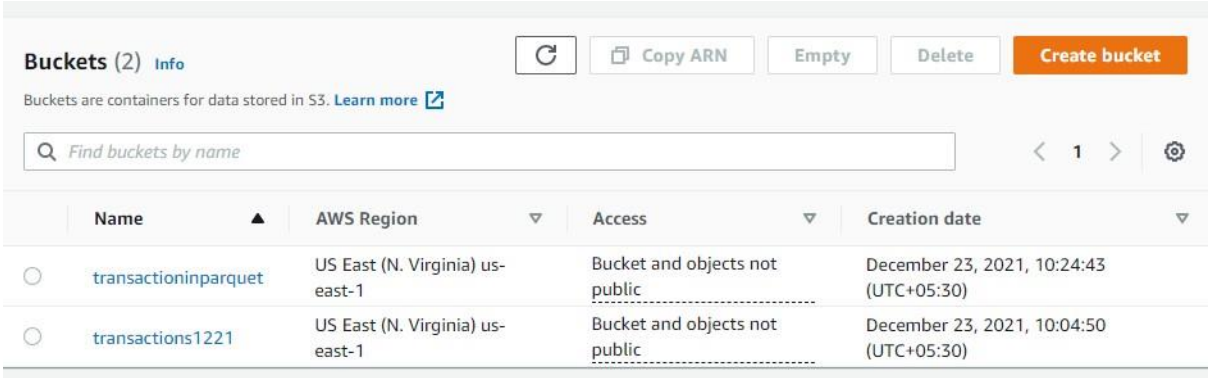
Prerequisites:

- Create two buckets as below



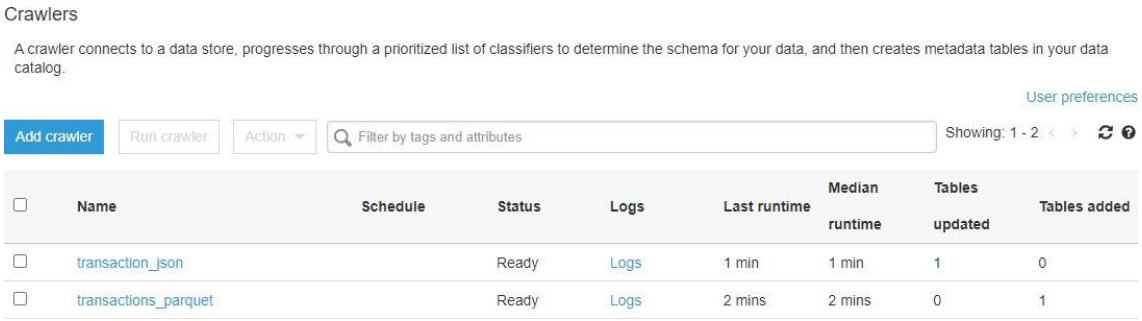
Buckets (2) Info

Buckets are containers for data stored in S3. Learn more

Find buckets by name

Name	AWS Region	Access	Creation date
transactioninparquet	US East (N. Virginia) us-east-1	Bucket and objects not public	December 23, 2021, 10:24:43 (UTC+05:30)
transactions1221	US East (N. Virginia) us-east-1	Bucket and objects not public	December 23, 2021, 10:04:50 (UTC+05:30)

- Create two glue crawlers specifying respective source folders of s3



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	Status	Logs	Last runtime	Median runtime	Tables updated	Tables added
transaction_json		Ready	Logs	1 min	1 min	1	0
transactions_parquet		Ready	Logs	2 mins	2 mins	0	1

Setting up Data Server that will emit realtime logs

- Create a new EC2 instance and login the same via putty.

- Install AWS kinesis firehose agent in the data source machine

```
sudo yum install -y aws-kinesis-agent
```

- Create necessary folders

```
sudo mkdir /var/log/Transactions
```

- Copy the txns file using scp command(connect using pem file)

```
scp -i ./pairnew.pem ./txns.txt ec2-user@18.212.166.35:/home/ec2-user
```

- Write a Python program that will generate logs

```
sudo vi LogGenerator.py
```

- write the below code and save

```
import csv
import time
import sys

sourceData = "txns.txt"
placeholder = "LastLine.txt"

def GetLineCount():
    with open(sourceData, encoding='latin-1') as f:
        for i, l in enumerate(f):
            pass
    return i

def MakeLog(startLine, numLines):
    destData = time.strftime("/var/log/Transactions/%Y%m%d-%H%M%S.log")
    with open(sourceData, 'r' , encoding='latin-1') as csvfile:
        with open(destData, 'w') as dstfile:
            reader = csv.reader(csvfile)
            writer = csv.writer(dstfile)
            next (reader) #skip header
            inputRow = 0
            linesWritten = 0
            for row in reader:
                inputRow += 1
                if (inputRow > startLine):
                    writer.writerow(row)
                    linesWritten += 1
                    if (linesWritten >= numLines):
                        break
            return linesWritten

numLines = 100
startLine = 0
if (len(sys.argv) > 1):
    numLines = int(sys.argv[1])

try:
    with open(placeholder, 'r') as f:
        for line in f:
            startLine = int(line)
except IOError:
    startLine = 0

print("Writing " + str(numLines) + " lines starting at line " + str(startLine) + "\n")

totalLinesWritten = 0
linesInFile = GetLineCount()
```

```
while (totalLinesWritten < numLines):
    linesWritten = MakeLog(startLine, numLines - totalLinesWritten)
    totalLinesWritten += linesWritten
    startLine +=  linesWritten
    if (startLine >= linesInFile):
        startLine = 0

print("Wrote " + str(totalLinesWritten) + " lines.\n")

with open(placeholder, 'w') as f:
    f.write(str(startLine))
```

- Give access to the *LogGenerator.py*

```
chmod a+x LogGenerator.py
```

- Setup Kinesis Delivery Stream named *TransactionsWithPreProcessing* in Kinesis Firehose. This stream will get the data from Agent and store the same in S3

- Create Kinesis Data Firehose Delivery Stream

  a. On the search bar type "kinesis" and click on Kinesis

  b. Click on/Select Kinesis Data Firehose > Create Delivery Stream

  c. Select Source as Direct PUT and Destination as Amazon S3

  d. Set the delivery stream name as " Transactions "

  e. Go to Destination Settings > Click on Create (to create a bucket). Once bucket is created go back to Kinesis tab and click on Browse > Select relevant bucket and click on Choose.

  f. Once done scroll down and click on "Create delivery Stream"

- Configure the Kinesis Firehose Agent to get the realtime streams from */var/log/Transactions* and store the same in my AWS Kinesis delivery stream *TransactionsWithPreProcessing* object. Go to Putty and perform the following

```
cd /etc/aws-kinesis/

sudo vi agent.json
```

Insert the below json file

```
{
  "cloudwatch.emitMetrics": true,
  "kinesis.endpoint": "",
  "firehose.endpoint": "firehose.us-east-1.amazonaws.com",

  "flows": [
        {
```

```
    "filePattern": "/var/log/Transactions/*.log",
    "kinesisStream": "TransactionsWithPreProcessing",
    "partitionKeyOption": "RANDOM",
    "dataProcessingOptions": [
      {
        "optionName": "CSVTOJSON",
        "customFieldNames": ["txnsid", "txndate", "custid", "amount", "category",
"subcategory", "city", "state", "txntype"]
      }
    ]
  }
 ]
}
```

- Start the agent

  sudo service aws-kinesis-agent start

- To check the status of the agent, create a duplicate session in Putty(Open a new session in PuTTY for EC2 instance) and type the following command

  tail -f /var/log/aws-kinesis-agent/aws-kinesis-agent.log

- Emit some logs in the original session

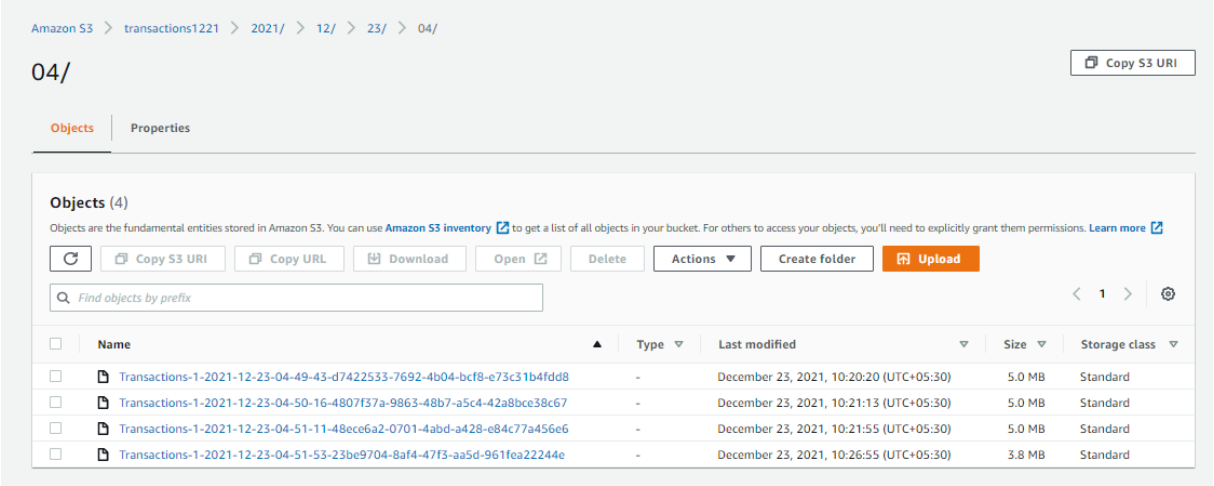
  cd\

  sudo ./LogGenerator.py 100000

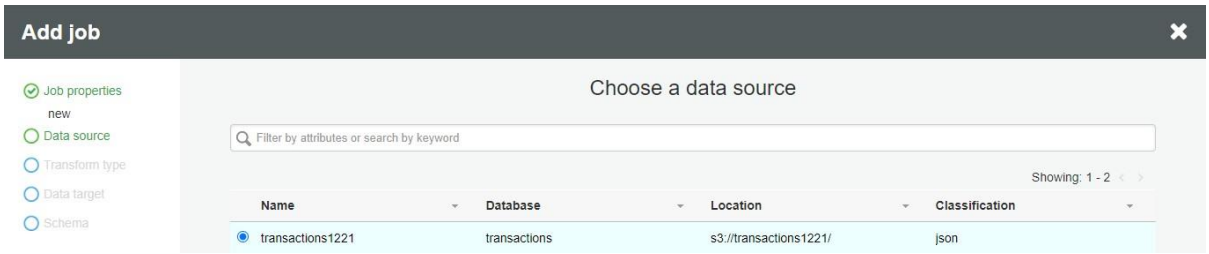
- In Duplicate session, you can see the records delivered successfully as below

```
2021-12-23 04:55:59.883+0000  (Agent.MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.Agent [INFO] Agent: Progress: 100000 records parsed (8934289 bytes), and
 100000 records sent successfully to destinations. Uptime: 420052ms
2021-12-23 04:56:29.881+0000  (FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log].MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.
tailing.FileTailer [INFO] FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log]: Tailer Progress: Tailer has parsed 100000 records (8934289 byte
s), transformed 100000 records, skipped 0 records, and has successfully sent 100000 records to destination.
2021-12-23 04:56:29.883+0000  (Agent.MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.Agent [INFO] Agent: Progress: 100000 records parsed (8934289 bytes), and
 100000 records sent successfully to destinations. Uptime: 450052ms
2021-12-23 04:56:59.881+0000  (FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log].MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.
tailing.FileTailer [INFO] FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log]: Tailer Progress: Tailer has parsed 100000 records (8934289 byte
s), transformed 100000 records, skipped 0 records, and has successfully sent 100000 records to destination.
2021-12-23 04:56:59.884+0000  (Agent.MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.Agent [INFO] Agent: Progress: 100000 records parsed (8934289 bytes), and
 100000 records sent successfully to destinations. Uptime: 480054ms
2021-12-23 04:57:29.881+0000  (FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log].MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.
tailing.FileTailer [INFO] FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log]: Tailer Progress: Tailer has parsed 100000 records (8934289 byte
s), transformed 100000 records, skipped 0 records, and has successfully sent 100000 records to destination.
2021-12-23 04:57:29.883+0000  (Agent.MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.Agent [INFO] Agent: Progress: 100000 records parsed (8934289 bytes), and
 100000 records sent successfully to destinations. Uptime: 510052ms
2021-12-23 04:57:59.881+0000  (FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log].MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.
tailing.FileTailer [INFO] FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log]: Tailer Progress: Tailer has parsed 100000 records (8934289 byte
s), transformed 100000 records, skipped 0 records, and has successfully sent 100000 records to destination.
2021-12-23 04:57:59.883+0000  (Agent.MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.Agent [INFO] Agent: Progress: 100000 records parsed (8934289 bytes), and
 100000 records sent successfully to destinations. Uptime: 540052ms
2021-12-23 04:58:29.881+0000  (FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log].MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.
tailing.FileTailer [INFO] FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log]: Tailer Progress: Tailer has parsed 100000 records (8934289 byte
s), transformed 100000 records, skipped 0 records, and has successfully sent 100000 records to destination.
2021-12-23 04:58:29.884+0000  (Agent.MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.Agent [INFO] Agent: Progress: 100000 records parsed (8934289 bytes), and
 100000 records sent successfully to destinations. Uptime: 570054ms
2021-12-23 04:58:59.881+0000  (FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log].MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.
tailing.FileTailer [INFO] FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log]: Tailer Progress: Tailer has parsed 100000 records (8934289 byte
s), transformed 100000 records, skipped 0 records, and has successfully sent 100000 records to destination.
2021-12-23 04:58:59.883+0000  (Agent.MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.Agent [INFO] Agent: Progress: 100000 records parsed (8934289 bytes), and
 100000 records sent successfully to destinations. Uptime: 600052ms
2021-12-23 04:59:29.881+0000  (FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log].MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.
tailing.FileTailer [INFO] FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log]: Tailer Progress: Tailer has parsed 100000 records (8934289 byte
s), transformed 100000 records, skipped 0 records, and has successfully sent 100000 records to destination.
2021-12-23 04:59:29.883+0000  (Agent.MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.Agent [INFO] Agent: Progress: 100000 records parsed (8934289 bytes), and
 100000 records sent successfully to destinations. Uptime: 630052ms
2021-12-23 04:59:59.881+0000  (FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log].MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.
tailing.FileTailer [INFO] FileTailer[kinesis:TransactionsWithPreProcessing:/var/log/Transactions/*.log]: Tailer Progress: Tailer has parsed 100000 records (8934289 byte
s), transformed 100000 records, skipped 0 records, and has successfully sent 100000 records to destination.
2021-12-23 04:59:59.883+0000  (Agent.MetricsEmitter RUNNING) com.amazon.kinesis.streaming.agent.Agent [INFO] Agent: Progress: 100000 records parsed (8934289 bytes), and
 100000 records sent successfully to destinations. Uptime: 660052ms
```

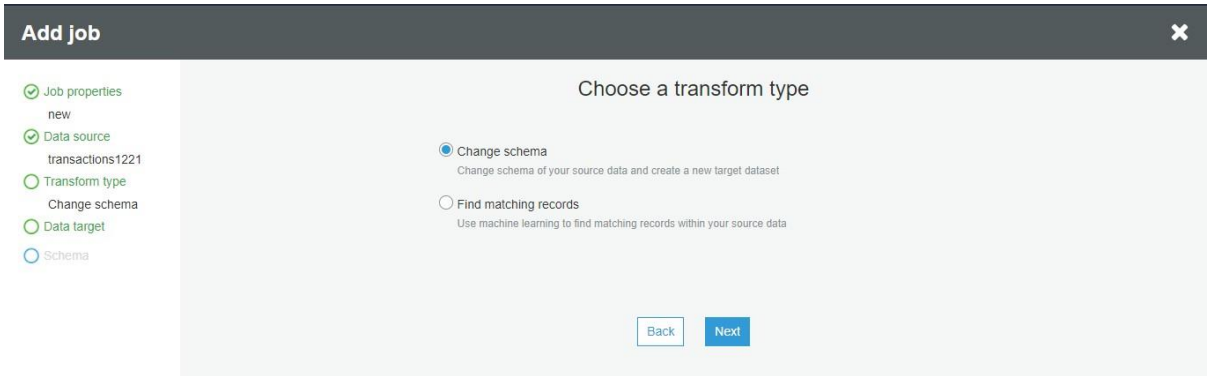
- Check the destination s3 bucket for JSON file generated



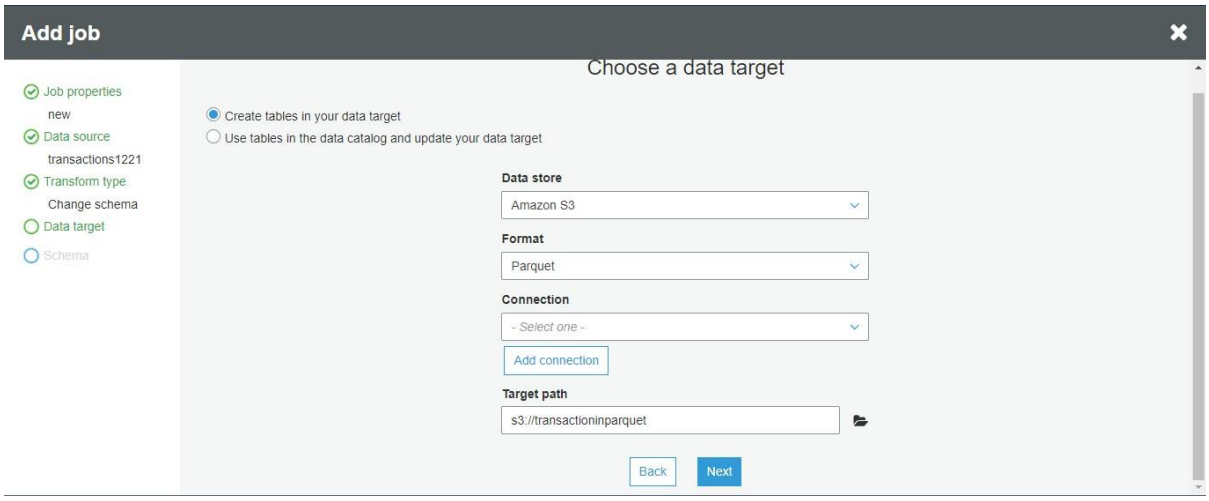
- Now run the crawler which will create the table for CSV to JSON converted data
- Once table is added, Go to jobs and create a job with an IAM role which will give access to the below two policies
    1. AmazonS3FullAccess
    2. AmazonGlueConsoleAccess
- Select the source dataset



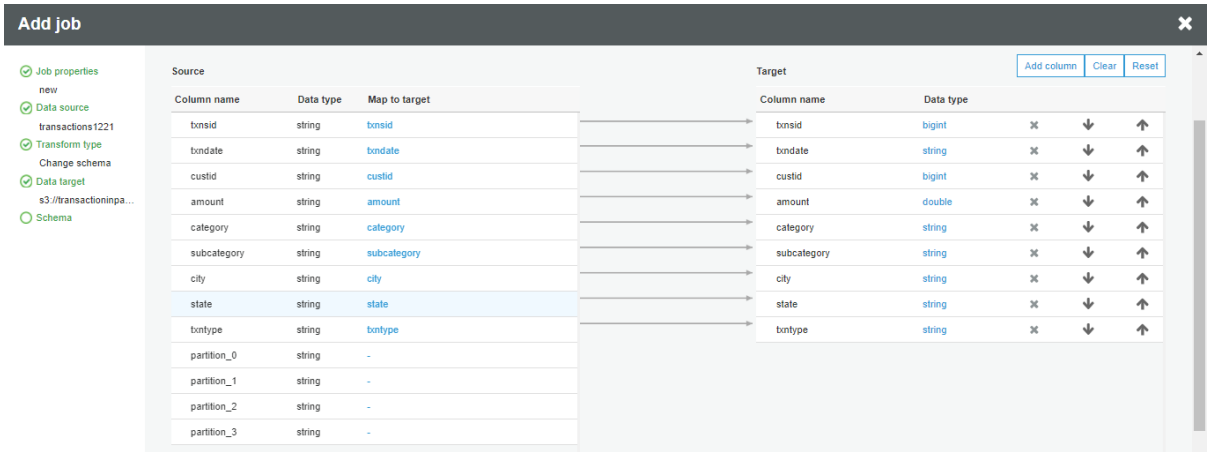
- Select the transformation type as change schema



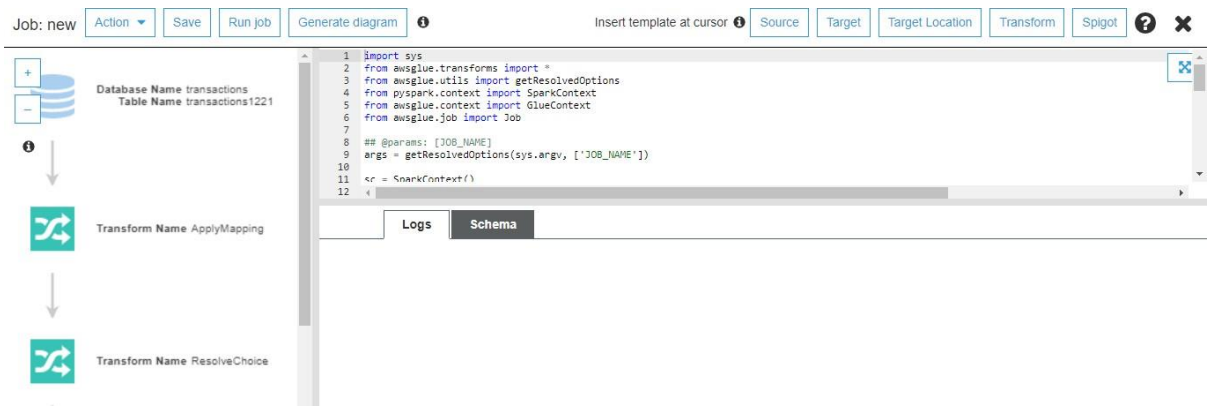
- In data target, choose create table as below and select the destination s3 bucket



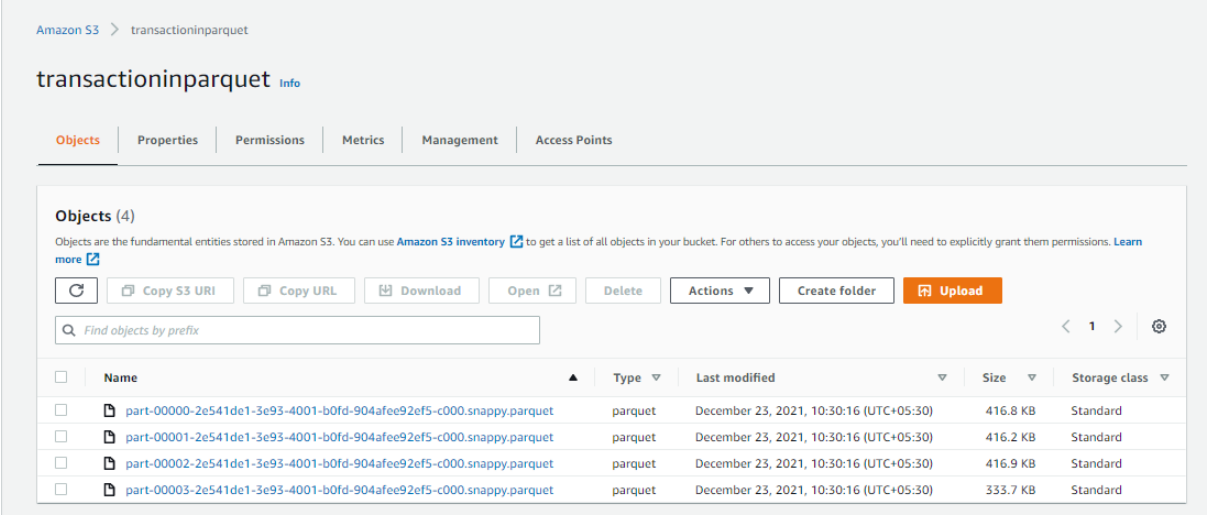
- Transform the dataset with relevant schema type as required also you can remove unnecessary columns as below



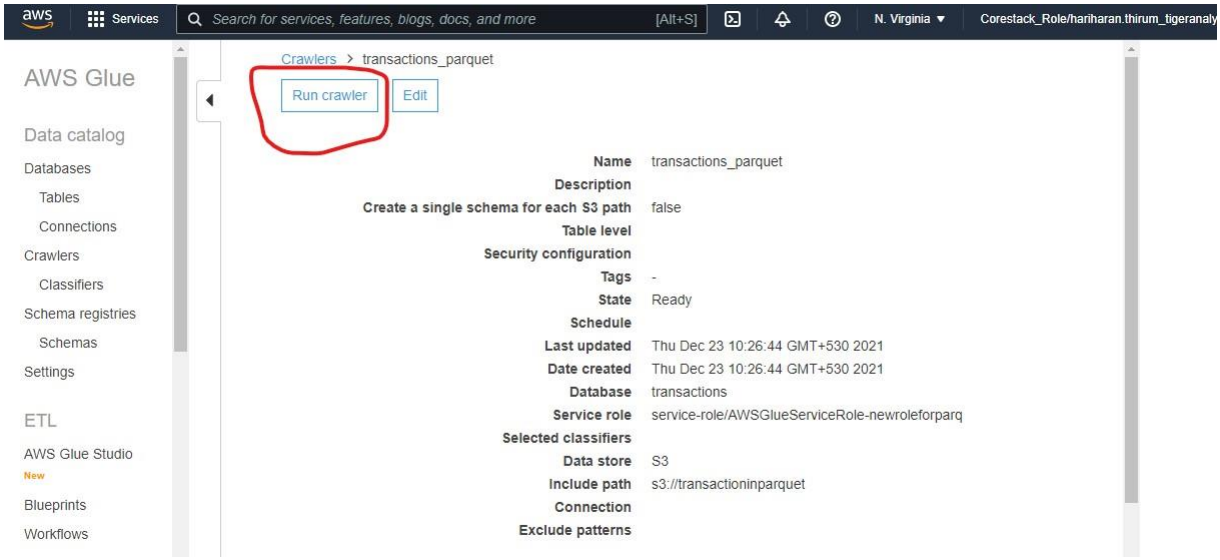
- Once saving the job, you will be having an option to Run crawler as below



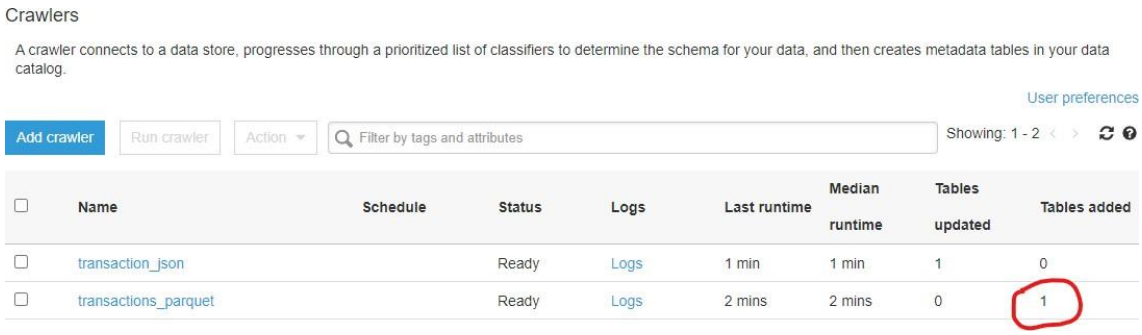
- You can see the parquet data in the destination bucket



- Now go to transactions_parquet crawler and run it



- Once crawler run completed, you'll see tables being added as below



- Now we are all set to run the query in Athena

Queries:

1. Find the total count of transactions done by Credit Card

Query 1

```
select txntype, count(txntype) as count_of_transactions from transactioninparquet where txntype like
    'credit%' group by txntype;
```

SQL Ln 1, Col 129

Run again | Cancel | Save as | Clear | Create

Completed Time in queue: 0.253 sec Run time: 1.301 sec Data scanned: 13.11 KB

Results (1) Copy Download results

Search rows 1

txntype	count_of_transactions
credit	86329

2. Find the total revenue generated based on category

Query 1

```
select category, sum(amount) as revenue from transactioninparquet group by category order by revenue desc;
```

SQL Ln 1, Col 106

Run again | Cancel | Save as | Clear | Create

Completed

Results (15)

Search rows

category	revenue
Outdoor Recreation	1717261.250000001
Exercise & Fitness	1522469.1199999996
Team Sports	1240278.3199999984
Water Sports	1070712.1800000006
Games	742629.9699999997
Gymnastics	662534.64
Winter Sports	647571.44
Outdoor Play Equipment	581474.3600000001
Indoor Games	572380.1599999995
Jumping	399577.14
Combat Sports	334560.24999999994
Racquet Sports	331485.51999999984
Air Sports	208062.14000000004
Puzzles	122995.13999999998
Dancing	86071.36999999998

3. Find the top selling subcategory in 'Exercise & Fitness'

```
select subcategory, count(txnsid) as sell_count from transactioninparquet where category = 'Exercise
    & Fitness' group by subcategory order by sell_count desc limit 1;
```

SQL Ln 1, Col 112

Run again | Cancel | Save as | Clear | Create

Completed Time in queue: 0.265 sec Run time: 0.661 sec Data scanned: 584.54 KB

Results (1) Copy Download results

Search rows 1

subcategory	sell_count
Exercise Balls	882

4. Find the least selling category.

```
select category, count(txntype) as sell_count from transactioninparquet group by category order by
    sell_count limit 1;
```

SQL Ln 1, Col 27

Run again | Cancel | Save as | Clear | Create

Completed Time in queue: 0.191 sec Run time: 0.576 sec Data scanned: 63.26 KB

Results (1) Copy Download results

Search rows 1

category	sell_count
Dancing	829

5. Find the top selling subcategory in least selling category

Query 1

```
select subcategory, sum(amount) as revenue,count(subcategory) as sell_count from transactioninparquet
    where category in (select a.category from
(select category, sum(amount) as revenue,count(category) as sell_count from transactioninparquet
    group by category order by sell_count limit 1) a)
group by subcategory order by sell_count desc limit 1;
```

SQL Ln 3, Col 55

Run again | Cancel | Save as | Clear | Create

Completed Time in queue: 0.184 sec Run time: 1.238 sec Data scanned: 673.61 KB

Results (1) Copy Download results

Search rows 1

subcategory	revenue	sell_count
Ballet Bars	86071.36999999998	829

6. Find total revenue generated in Air Sports category

Query 1

```
select category, sum(amount) as revenue from transactioninparquet where category = 'Air Sports' group
    by category;
```

SQL Ln 1, Col 115

Run again | Cancel | Save as | Clear | Create

Completed Time in queue: 0.163 sec Run time: 0.554 sec Data scanned: 531.46 KB

Results (1) Copy Download results

Search rows 1

category	revenue
Air Sports	208062.14000000004