Amazon Athena Demo

- Author: *Mahmoud Parsian*
- Date Updated: *Nonember 28, 2023*

1. Input Prep: Create Data File(s)

Record format:

```
        <customer_id><,><year><,><transaction_id><,><price>
```

Each record has fours fields:

- `<customer_id>`
- `<year>`
- `<transaction_id>`
- `<price>`

```
$ cat customers.txt
c1,2019,t00012,12.5677
c1,2019,t00010,14.56
c1,2019,t00011,14.56
c1,2018,t000126,12.5677
c1,2018,t000107,140.56
c1,2018,t000119,164.56
c1,2017,t100126,120.5677
c1,2017,t100107,1400.56
c1,2017,t100119,1640.56
c2,2019,t90012,12.5677
c2,2019,t90010,147.56
c2,2019,t90011,147.56
c2,2018,t0800126,127.5677
c2,2018,t0080107,1470.56
c2,2018,t0008119,164.56
c2,2017,t1001526,1720.5677
c2,2017,t1001057,14700.56
c2,2017,t1001195,16740.56
```

2. Upload the Input to Amazon S3

```
        $ aws s3 cp customers.txt s3://mybucket/data/customers.txt
```

3. Read Data from S3, create a DataFrame and create data partitions:

`$ cat extract_and_load.py`

```
# 1. import required libraries

from __future__ import print_function
import sys
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType
from pyspark.sql.types import StructField
from pyspark.sql.types import StringType
from pyspark.sql.types import DoubleType

# 2. create a custom schema
customer_schema = StructType([
   StructField("customer_id", StringType(), True),
   StructField("date", StringType(), True),
   StructField("transaction_id", StringType(), True),
   StructField("price", DoubleType(), True)])

# 3. define input path
input_path = "s3://mybucket/data/customers.txt"

# 4. Create an instance of SparkSession
spark = SparkSession.builder.getOrCreate()

# 5. read data and create a DataFrame
df = spark.read.csv(input_path, schema=customer_schema)

# 6. some debugging
df.show(5, truncate=False)
df.printSchema()

# 7. partition data by 2 fields and save in output path
output_path = "s3://mybucket/output/customers/"
df.repartition("customer_id", "date")\
    .write.partitionBy("customer_id", "date")\
    .parquet(output_path)

# 8. done!
spark.stop()
```

The PySpark will create the following output directores and files:

```
s3://mybucket/output/customers/customer_id=c1/date=2017/<file1>.parquet
s3://mybucket/output/customers/customer_id=c1/date=2018/<file2>.parquet
s3://mybucket/output/customers/customer_id=c1/date=2019/<file3>.parquet
s3://mybucket/output/customers/customer_id=c2/date=2017/<file4>.parquet
s3://mybucket/output/customers/customer_id=c2/date=2018/<file5>.parquet
s3://mybucket/output/customers/customer_id=c2/date=2019/<file6>.parquet
```

You can clearly see that data is partitioned by

- `customer_id`
- `date`

4. Create a sample database (catalog) called sampledb

A database in Athena is a logical grouping for tables you create in it. Open the Athena console . Enter

```
CREATE DATABASE sampledb;
```

and choose **Run Query**.

For details on creating a database and schema, see How to create a database in Amazon Athena

5. Create scahema and point to the output created by

PySpark program (in Athena Web Console)

```
CREATE EXTERNAL TABLE `sampledb.customers`(
  `transaction_id` string,
  `price` double
)
PARTITIONED BY (
  `customer_id` string,
  `date` string
)
STORED AS PARQUET
LOCATION 's3://mybucket/output/customers/'
tblproperties ("parquet.compress"="SNAPPY");
```

6. Load partitions (in Athena Web Console)

```
MSCK REPAIR TABLE customers;
```

7. Ready to query customers table: (in Athena Web Console):

This will be a full table scan.

```
 SELECT *
   FROM "sampledb"."customers";

 Results
        transaction_id  price   customer_id     date
1       t0800126        127.5677        c2      2018
2       t0080107        1470.56         c2      2018
3       t0008119        164.56          c2      2018
4       t1001526        1720.5677       c2      2017
5       t1001057        14700.56        c2      2017
6       t1001195        16740.56        c2      2017
7       t100126         120.5677        c1      2017
8       t100107         1400.56         c1      2017
9       t100119         1640.56         c1      2017
10      t90012          12.5677         c2      2019
11      t90010          147.56          c2      2019
12      t90011          147.56          c2      2019
13      t000126         12.5677         c1      2018
14      t000107         140.56          c1      2018
15      t000119         164.56          c1      2018
16      t00012          12.5677         c1      2019
17      t00010          14.56           c1      2019
18      t00011          14.56           c1      2019
```

8. Query slice of a data by using partitioned columns:

```
 SELECT *
   FROM "sampledb"."customers"
    where customer_id = 'c1' and
          date = '2017'

 Results
        transaction_id  price           customer_id     date
1       t100126         120.5677        c1              2017
2       t100107         1400.56         c1              2017
3       t100119         1640.56         c1              2017
```

For this query, only the following directory will be scanned (**analyze slice of a data rather than the whole data**):

```
s3://mybucket/output/customers/customer_id=c1/date=2017/
```