Source: The following hands-on tutorial is modified from AWS. Big Data on AWS 3.2 (EN): Student Guide. AWS/Gilmore. VitalBook file.

Lab 1 - Using Amazon Athena to Analyze Log Data

In this lab, you will analyze Elastic Load Balancer log data. The data is provided in an Amazon S3 bucket in various different formats:

- Raw data stored in plain text
- Compressed data in gzip format
- Partitioned data split into sub-directories
- Columnar data, stored in Parquet format

Table 1. Runtime and Data scanned of Different File Formats

	Runtime & Data scanned
Task 1: Raw data stored in plain text	
Task 2: Compressed data in gzip format	
Task 3: Compressed and Partitioned data split into sub-directories	
Task 4: Columnar data (Parquet with Snappy compression)	

You will create table definitions to access the data from Amazon Athena and run queries against the data. You will then compare the efficiency of each storage format.

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to setup or manage, and you pay only for the queries you run. You simply point to your data in Amazon S3, define the schema, and start querying using standard SQL.

Task 1: Query Data in Raw Format

1. Log into https://cs-min-dev.signin.aws.amazon.com/console
2. In the AWS Management Console, on the Services menu, click Athena.
3. Click Get Started.
4. Copy this command and click Run Query
   ```
   CREATE DATABASE lab
   ```
5. In DATABASE section, select lab.
6. Copy the query shown below:
   ```
   CREATE EXTERNAL TABLE IF NOT EXISTS lab.elb_logs_raw (request_timestamp string, elb_name string, request_ip string, request_port int, backend_ip string, backend_port int, request_processing_time double, backend_processing_time double, client_response_time double, elb_response_code
   ```

```
string, backend_response_code string, received_bytes bigint,
sent_bytes bigint, request_verb string, url string, protocol
string, user_agent string, ssl_cipher string, ssl_protocol
string )
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH     SERDEPROPERTIES     ('serialization.format'     =
'1','input.regex' = '([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^
]*)[:\-]([0-9]*) ([-.0-9]*) ([-.0-9]*) ([-.0-9]*) (|[-0-9]*)
(-|[-0-9]*) ([-0-9]*) ([-0-9]*) \\\"([^ ]*) ([^ ]*) (- |[^
]*)\\\" (\"[^\"]*\") ([A-Z0-9-]+) ([A-Za-z0-9.-]*)$' )
LOCATION   's3://aws-tc-largeobjects/AWS-200-BIG/v3.1/lab-2-
athena/raw/';
```

7. Paste the query into the Athena query editor (replacing any existing text). Run the query.
8. Click elb_logs_raw in the table listing. A list of columns will be displayed, which matches the table definition you created.
9. Click the ⋮ dots to the right of the table name, then select Preview table. The first 10 rows will be displayed. The Run time and Data scanned are also shown to the right of the New Query button.
10. The first 10 rows will be displayed. The Run time and Data scanned are also shown to the right of the New Query button.

```
SELECT elb_response_code, count(url) as Count
FROM elb_logs_raw WHERE request_timestamp LIKE '2015-01-01%'
GROUP BY elb_response_code ORDER BY elb_response_code;
```

11. Make a note of the Run time and the Data scanned for in Table 1.

Task 2: Query Compressed Data

1. Run the query below. The new elb_logs_compressed table definition will appear in the Tables listing in the left navigation panel.

```
CREATE EXTERNAL TABLE IF NOT EXISTS lab.elb_logs_compressed
( request_timestamp string, elb_name string, request_ip
string,
request_port int, backend_ip string, backend_port int,
request_processing_time   double,   backend_processing_time
double,   client_response_time   double,   elb_response_code
string, backend_response_code string, received_bytes bigint,
sent_bytes bigint,
request_verb string, url string, protocol string, user_agent
string, ssl_cipher string, ssl_protocol string )
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH     SERDEPROPERTIES     ('serialization.format'     =
'1','input.regex' = '([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^
]*)[:\-]([0-9]*) ([-.0-9]*) ([-.0-9]*) ([-.0-9]*) (|[-0-9]*)
```

```
(-|[-0-9]*) ([-0-9]*) ([-0-9]*) \\\"([^ ]*) ([^ ]*) (- |[^
]*)\\\" (\"[^\"]*\") ([A-Z0-9-]+) ([A-Za-z0-9.-]*)$' )
LOCATION   's3://aws-tc-largeobjects/AWS-200-BIG/v3.1/lab-2-
athena/compressed/';
```

2. Run the query below to query the compressed data. Make a note of the Run time and the Data scanned for later comparison.

```
SELECT elb_response_code, count(url)as Count
FROM elb_logs_compressed WHERE request_timestamp LIKE '2015-
01-01%'    GROUP    BY    elb_response_code    ORDER    BY
elb_response_code;
```

Task 3: Query Partitioned Data

1. Run the query below to create a table from partitioned data. Notice that this query has a PARTITIONED BY clause, which tells Athena that data is partitioned into directories by Year, Month and Day.

```
CREATE EXTERNAL TABLE IF NOT EXISTS lab.elb_logs_partitioned
( request_timestamp string, elb_name string, request_ip
string, request_port int, backend_ip string, backend_port
int, request_processing_time double, backend_processing_time
double,   client_response_time   double,   elb_response_code
string, backend_response_code string, received_bytes bigint,
sent_bytes bigint, request_verb string, url string, protocol
string, user_agent string, ssl_cipher string, ssl_protocol
string ) PARTITIONED BY (year int, month int, day int)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH     SERDEPROPERTIES     ('serialization.format'     =
'1','input.regex' = '([^ ]*) ([^ ]*) ([^ ]*):([0-9]*) ([^
]*)[:\-]([0-9]*) ([-.0-9]*) ([-.0-9]*) ([-.0-9]*) (|[-0-9]*)
(-|[-0-9]*) ([-0-9]*) ([-0-9]*) \\\"([^ ]*) ([^ ]*) (- |[^
]*)\\\" (\"[^\"]*\") ([A-Z0-9-]+) ([A-Za-z0-9.-]*)$' )
LOCATION   's3://aws-tc-largeobjects/AWS-200-BIG/v3.1/lab-2-
athena/compressed/';
```

2. Run this query to scan the disk, and load the partitions into the Athena metadata:

```
MSCK REPAIR TABLE lab.elb_logs_partitioned;
```

3. Run this query against the compressed data. Make a note of the Run time and the Data scanned for later comparison in Table 1.

```
SELECT elb_response_code, count(url)as Count
FROM elb_logs_partitioned WHERE year=2015 AND month=1
AND   day=1   GROUP   BY   elb_response_code   ORDER   BY
elb_response_code;
```

Task 4: Query Columnar Data

1. For this task, data has been stored in Apache Parquet format which stores data by column. Run the query below to create a table from data in Parquet format:

```
CREATE EXTERNAL TABLE IF NOT EXISTS lab.elb_logs_parquet ( request_timestamp string, elb_name string, request_ip string, request_port int, backend_ip string, backend_port int, request_processing_time double, backend_processing_time double, client_response_time double, elb_response_code string, backend_response_code string, received_bytes bigint, sent_bytes bigint,
request_verb string, url string, protocol string, user_agent string, ssl_cipher string, ssl_protocol string )
PARTITIONED BY(year int, month int, day int)
STORED AS PARQUET LOCATION 's3://aws-tc-largeobjects/AWS-200-BIG/v3.1/lab-2-athena/parquet/' tblproperties ("parquet.compress"="SNAPPY");
```

2. Run this query to scan the disk, and load the partitions into the Athena metadata:

```
MSCK REPAIR TABLE lab.elb_logs_parquet;
```

3. Run this query against the compressed data. Make a note of the Run time and the Data scanned for later comparison in Table 1.

```
SELECT elb_response_code, count(url)as Count
FROM elb_logs_parquet WHERE year=2015
AND month=1
AND day=1 GROUP BY elb_response_code ORDER BY elb_response_code;
```