Twitter Data Analysis

in

AMAZON REDSHIFT

For IDS 521 Advanced Database Management Systems



**Submitted By:**

**(AMAZON REDSHIFT TEAM 2)**

DEVYANI SINGH-654766729

MEENAKSHI PARAMESWARAN-673380596

SUMEET KUKREJA- 674668076

TARUN KHURANA-667683595

**Executive Summary**

The following report summarizes the data warehousing techniques implemented using **Amazon Redshift**, which were further analyzed using some Business intelligence tools like **tableau** etc. For this project, the data we used was Twitter data provided to us for our homework assignments.

We divided our work into three main steps.

The first step was to perform an ETL (Extract, Transform and Load) process where we were supposed to extract, transform and load the Twitter data into the Amazon Redshift Dashboard to later analyze it.

Here is the summary of the steps we followed for the ETL process.

The twitter data that we had was in the form of excel files, because of which we created an s3 bucket and loaded this data into it. After successful loading data into the s3 bucket, we created a Cluster in Redshift Dashboard. To allow this cluster to access the s3 bucket, we created an IAM role (IAM roles attach a policy to itself which defines that role) with “fulls3access” policy as the “fulls3access” policy enables any cluster to have full access to the s3 bucket. Even after creating an IAM role the data stored in s3 cannot be directly loaded into the Amazon Redshift cluster because of which ETL tools like SQL workbench/J were used. In SQL workbench/J, for loading the data stored in s3 to the Amazon Redshift cluster, we further created a group specifying the JDBC/ODBC Redshift driver, cluster username and password, for establishing a connectivity between our Amazon Redshift cluster and SQL workbench/J. After successfully connection establishment, we wrote the COPY command in SQL workbench/J for copying the data from s3 bucket to our Redshift cluster using the IAM role with “fulls3access” policy. After we ran the COPY command, the uploaded Twitter data excel files in s3 bucket were Copied into the Redshift cluster, hence completing the ETL process.

In our next step, we tried to analyze the load performance on this cluster. From the metrics, we could easily see which load failed, or which Copy command had errors along with its error stated explicitly. We analyzed how these queries performed from the execution time and the performance time plots. We also saw that automated snapshots created periodic back-ups in local memory, S3 and compute nodes to save data in case of failures. Now we had the data stored in our cluster in an optimized manner.

Finally, we tried analyzing this Twitter data stored in our cluster using a Business Intelligence tool called Tableau. We connected our Redshift cluster with Tableau, using endpoint and cluster user credentials and loaded the data from cluster to Tableau. We then used Tableau to visually capture the analysis of various firms and finding the effect on twitter volume per twitter mentions at ten minute intervals.

In our execution of the whole process, we could fully understood the capabilities of Amazon Redshift, also identifying some of its limitations and challenges along the process.

**Introduction**

From early 2006 Amazon Web Services (AWS), a cloud computing platform by Amazon, started providing services to business in the form of web services with benefits like flexible, scalable, reliable, secure and cost effective which proved to be highly beneficial to businesses as they didn’t have to plan and reserve servers and other resources well in advance, helping them to deliver cost effective and faster solutions.

Amazon Redshift is a cloud based data warehousing service which forms a part of AWS. It is used as a traditional Enterprise Data Warehouse or by companies with Big Data. One of the biggest advantage of using this relational database is that you can use standard SQL and existing BI tools for analyzing data. It provides massively parallel columnar storage and has a distributed architecture which dramatically reduces I/O thereby, speeding up the query execution time.

There are several features which makes Amazon Redshift stand out from other data warehouses. Firstly, it is faster than traditional DBMS tools due to its columnar storage (stores data tables by column rather than by row) and uses a block size of 1MB which reduces the number of I/O requests, optimizing query performance.

Secondly, it’s cost effective in the sense that it has a “pay as you go” policy which says that you only pay for resources you use, with no minimum or setup fees. This is highly beneficial for businesses as it enables them to create and destroy resources on the go as required and pay only for the ones they use.

Thirdly, scaling up and down is easy. Scaling is a primary concern in production environments where you want the flexibility to resize the cluster without having to configure related items. As required, cluster can be easily resized with a few clicks or through a API call to petabytes of data for $1,000 per terabyte per year, less than a tenth the cost of traditional solutions.

Fourthly, it has automated and continuous data backups. It has mainly 3 backups – replicated data within warehouse cluster, backups to Amazon S3 bucket, replicate snapshots (point-in-time backups) of a cluster to another region for disaster recovery.

Lastly, it has built-in security with encryption. It supports SSL enabled connections through IAM roles between client application and your Redshift data cluster. Moreover, it also has an Amazon Virtual Private Cloud (Amazon VPC) which enables you to define a traditional network that you might operate in your own data center. This will give you complete access and control over your Redshift cluster.

Amazon Redshift continues to grow and prosper at a very fast pace as in the month of November (2017) alone, it has come up with new features like introduction of Result Caching for repeated queries and better performance, Cross Region backups, Quick Start, Reserved Instances Utilization report and alerts, Short Query Acceleration and many more.

**Architecture**

The snapshot of the architecture of Amazon Redshift data warehouse is shown below:



Fig: Amazon Redshift Architecture  
Source: <http://docs.aws.amazon.com/redshift/latest/dg/c_high_level_system_architecture.html>

* **Client Applications:** Client Applications like MySQL, SQL workbench are connected to the cluster using JDBC/ODBC drivers.
* **Data Warehouse Cluster:** The Cluster is the core component of the Amazon Redshift data warehouse. It is composed of one or more compute nodes which are further controlled by the leader node. The leader node is created only if the number of compute nodes increases, hence, is the one which interacts with the Client applications using JDBC/ODBC drivers.
* **JDBC/ODBC**: Amazon Redshift uses the standards JDBC/ODBC drivers for connections.
* **Leader Node:** The leader node is the main node that manages connections between the cluster and the client applications by developing the execution plans for the smooth execution of complete process. Based on the execution plan the leader node compiles, distributes and assigns the code to each of the compute nodes.
* **Compute Node:** After receiving the compiled code from the leader node, the commute node executes the code and sends the results back to the leader node for the final aggregation.
* **Node Slices:** The compute nodes are further partitioned into parallel slices called Node Slices which work in tandem to complete the assigned task.

**Workflow**

Our team followed the underlined workflow to analyze the twitter data using Amazon Redshift:

* Cluster Creation: A cluster in Redshift is a network of nodes (Lead and Compute nodes) that contain all the distributed data. The Lead node of the cluster communicates with the BI tools. This is the foremost point of communication between Redshift and the BI/ETL tools. After creating the cluster, access roles are assigned to the cluster so that we can access the data set from S3 bucket.

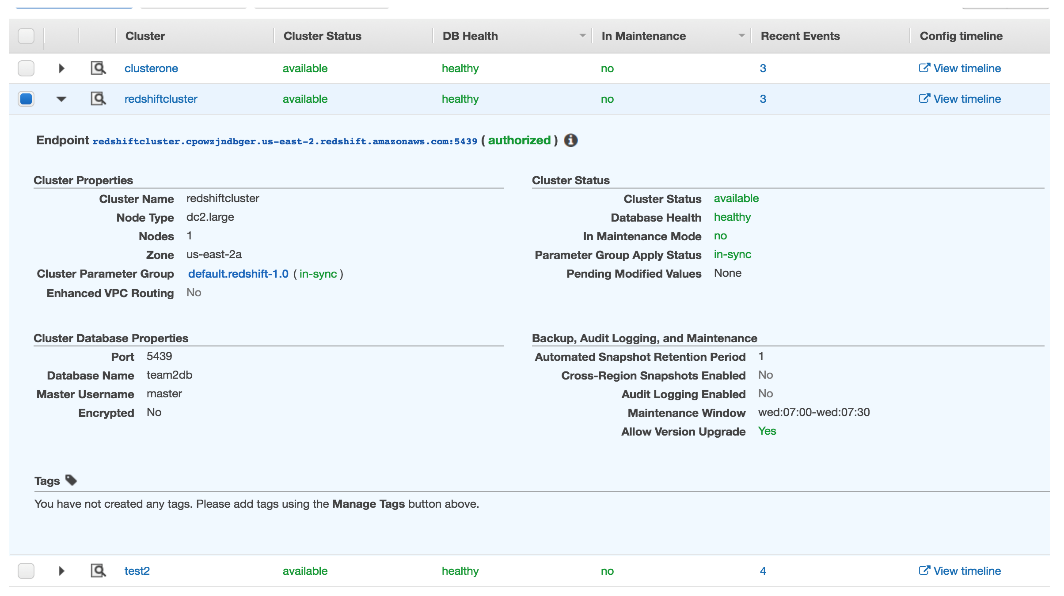


Fig. The cluster is connected to ETL tools via authorized endpoint

* We create a S3 bucket to store our data files. The size of the data files that can be stored does not depend on this size of the cluster nodes. The cluster nodes divide the entire data among it selves depending on the size of cluster.

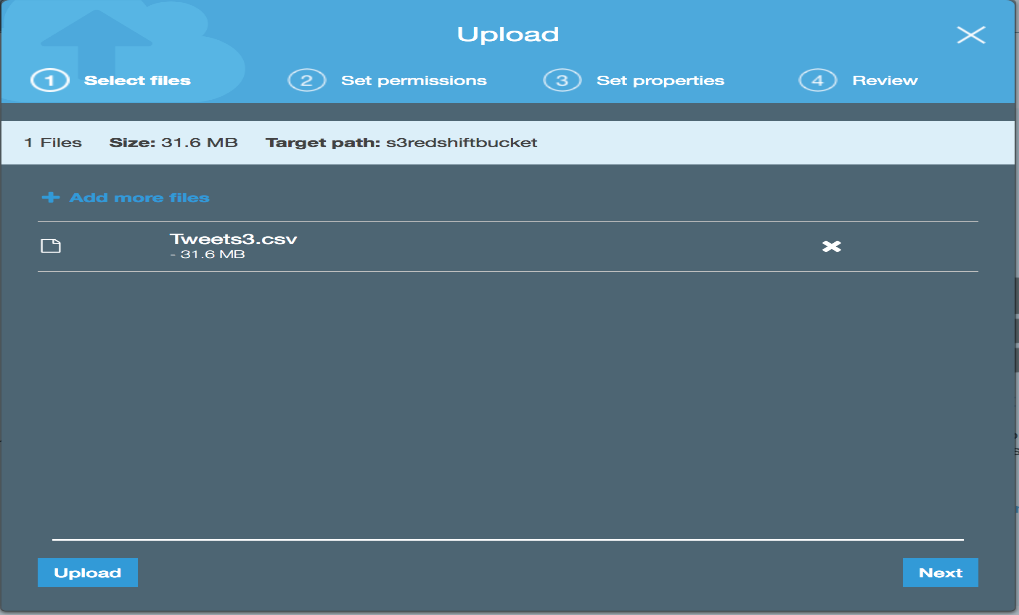


Fig. The S3 bucket “s3redhiftbucket” containing the file “Tweets3.csv

* Before connecting the ETL tool (SQL Workbench/J in our project), VPC security group should be given which depends if the cluster is publicly accessible or not. In case of private access, provide local IP address.



Fig: The local IP address is given in the inbound rules

* The SQL Workbench/J is connected using the JDBC driver address mentioned in the cluster’s dbinstance. Once the connection is established, we can load the data from S3 bucket for SQL query implementation. The data is copied using the COPY command.

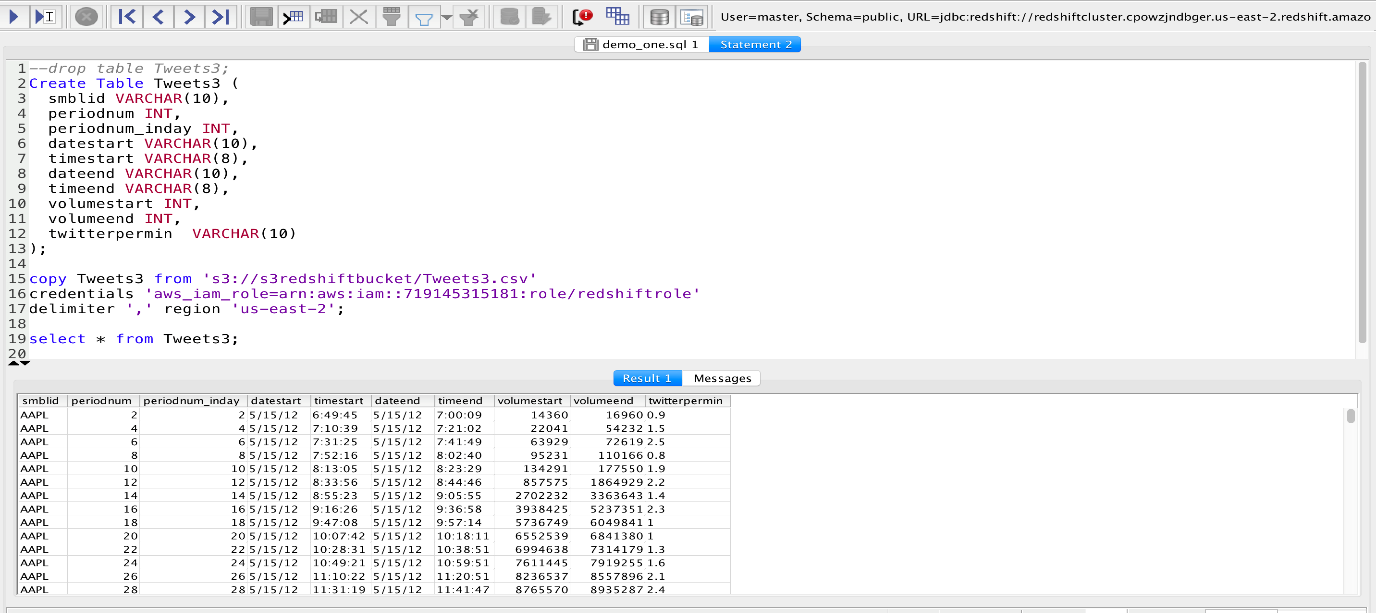


Fig: Loading the twitter data from S3 bucket to locally created table

* We can observe the query performance on Redshift to figure out how much time does an insert or update command takes. We can also check for any error(s) for unsuccessful query implementation

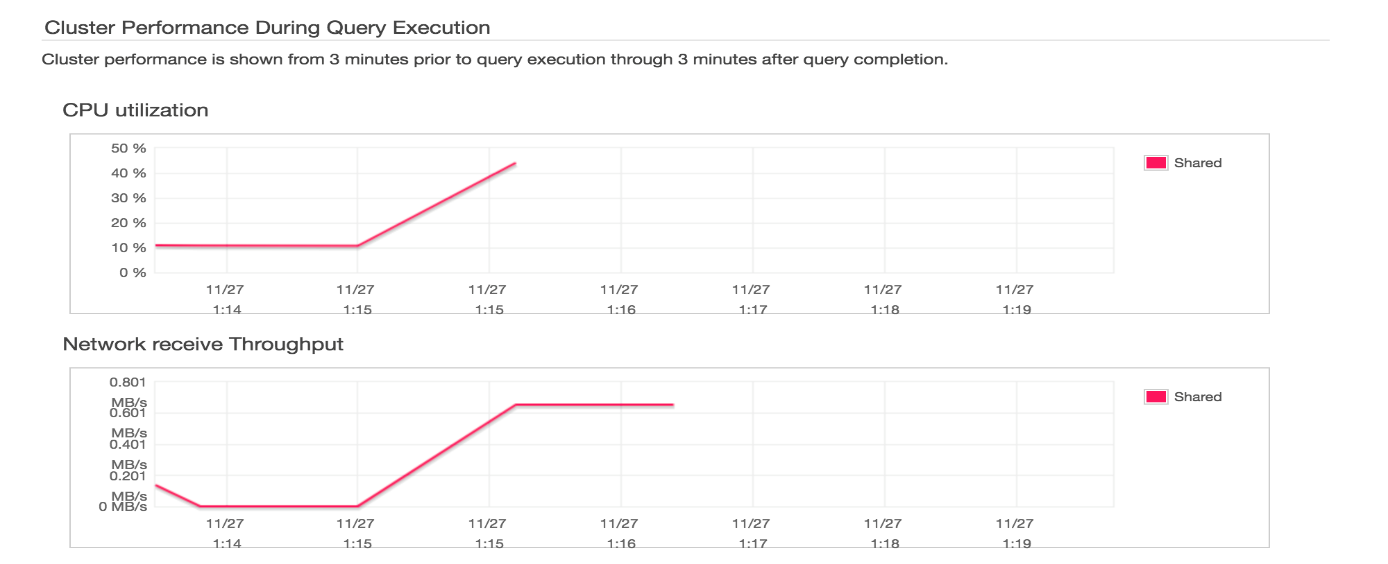


Fig: Cluster performance during query execution

**Analysis**

After establishing a connection from Redshift to Tableau, we conducted analysis by joining tables as well as taking them individually. Using Tweets3 table, visualizations were created taking the measure attributes Periodnum, Twitterpermin and Volumestart. A line graph was drawn in Tableau for Akamai Technologies (AKAM) and Adobe Systems(ADBE) with Periodnum on the X-axis, Twitterpermin and sum of Volumestart on the Y-axis.

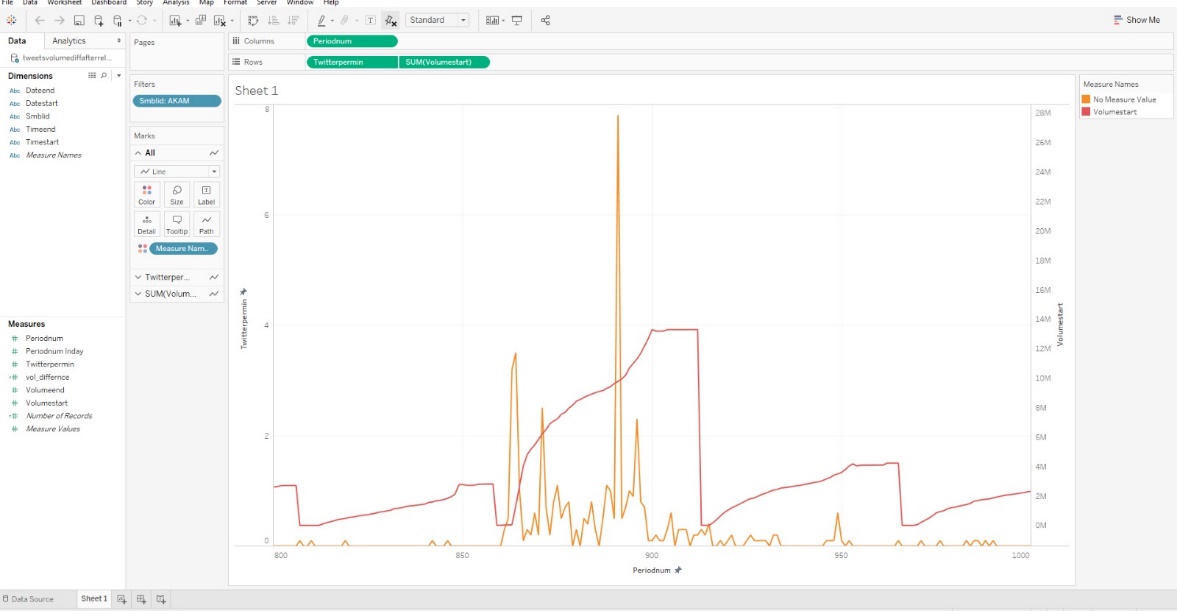


Fig: Titter peaks and trading volume for Akamai Technologies

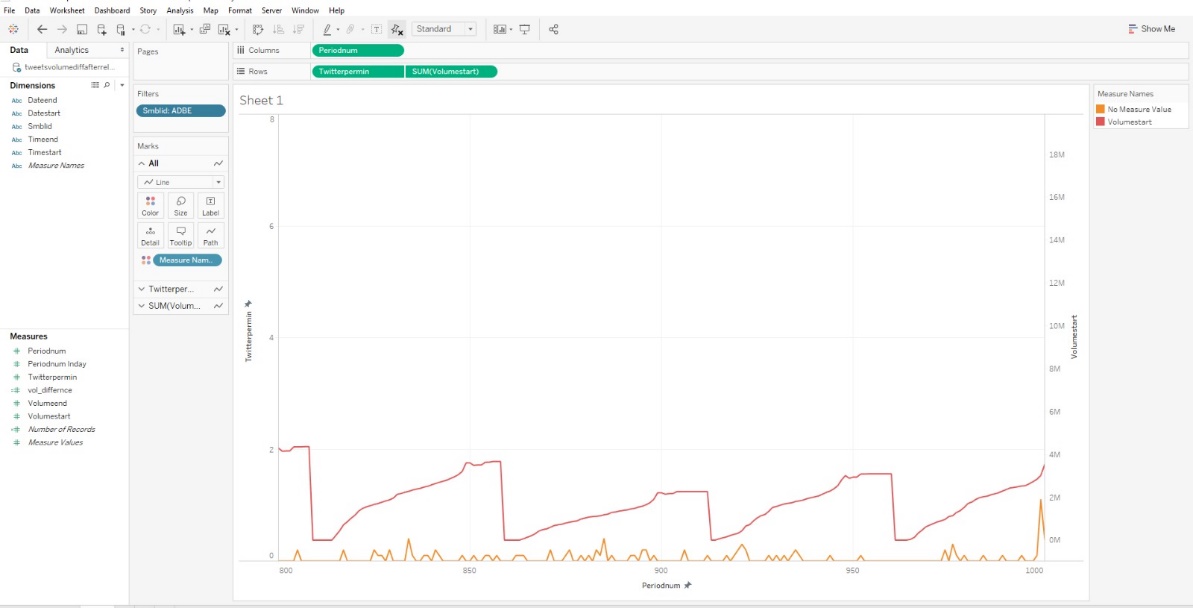


Fig: Titter peaks and trading volume for Adobe

The graph depicts Trading Volume with Twitter mentions at ten-minute intervals. A larger spike (in orange) in the graph depicts that a twitter activity has occurred due to recent announcements or updates from the company orabout the company. A higher value for Trading Volume (in red) is also seen with a larger spike.

**Limitations**

During the course of our project, we came across a few key limitations as mentioned below:

* User have to pay per hour for each of the servers that are not in use. A business data might be connected to multiple servers but operating only on one or few of the total bought servers. The cost incurred by the business would be for the total number of servers and not the current in use servers.
* Loading the data using the COPY command is tricky. Establishing a connection through an authorized endpoint is easy, but for loading the data from S3 bucket into the workbench, we need to provide the file path, and IAM roles associated with it. With increase in number of files, every time a data file has to be used for analysis, the credentials have to be looked up on Redshift.
* The user has to decide the size of each node before uploading the file. Based on the number of nodes (in case of multi-nodes cluster), we have to decide the size of each node so the data file can be divided in the cluster nodes without any loss of information.
* The keys do not play an important role in Redshift. Defining keys is for information only but are not enforced while table partitioning.

**Citations**

* The team took inspiration from the Twitter data provided by Professor Ali Tafti for class homework and his research paper, “Real-Time Diffusion of Information on Twitter and the Financial Markets”, Published: August 9, 2016 <https://doi.org/10.1371/journal.pone.0159226>
* To get a hold of Amazon Redshift’s Functionality, we referred to some tutorials. The list of all tutorials can be found [here](https://youtu.be/1eIaPmqWEOI).
* AWS documents were a great help in understanding the process of establishing a connection through JDBC/ODBC drivers and authorized endpoint of the cluster to the ETL/BI tools.
  + Cluster creation: <http://docs.aws.amazon.com/redshift/latest/gsg/rs-gsg-create-sample-db.html>
  + Connecting dbinstance: <http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ConnectToInstance.html>
  + Load data from S3 into Redshift: <http://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-template-s3redshift.html>