# Amazon Customer Reviews Analysis Using AWS

BY:

Jashwanth Kumar Enabothula (015920489)

Siddharth Magidewar (015918500)

**Abstract:**

Extract, Transform, and Load, or ETL, is a data integration process that integrates data from various sources into a single, consistent data store put into a data warehouse or other destination system. ETL was developed to integrate and load data for calculation and analysis as databases become more popular in the 1970s. It eventually became the dominant way of processing data for data warehousing operations. Data Analytics and Machine Learning work-streams rely on ETL for their basis. ETL cleanses and organizes data using a set of business rules to meet particular business intelligence requirements, such as monthly reporting. Still, it may also handle complex analytics to enhance back-end operations or end-user experiences.

An organization's ETL is frequently used to Retrieve data from legacy systems, to improve data accuracy and reliability, clean the data. Update a target database with data.

It is prevalent to use Redshift as a data-warehousing tool in the AWS cloud. However, there are quite some ways to orchestrate the loading, unloading and querying Redshift. In this project, we use in-house AWS tools to orchestrate end-to-end loading and deriving business insights. Since it uses in-house tools, the availability and durability of the solution are guaranteed by AWS

**Objective:**

Our project aims to analyze the Amazon Customer Reviews data from two different data sources, one with a data length of 2 million reviews and a second data source of 1 million reviews with various graph representations to interpret these datasets.

**Data Description:**

One of Amazon's most recognizable products is Amazon Customer Reviews. Millions of Amazon consumers have posted over a hundred million reviews to express their thoughts and explain their experiences with items on the Amazon.com website during the past two decades. This makes Amazon Customer Reviews a valuable resource for academics working in disciplines like Natural Language Processing, Information Retrieval, and Machine Learning. This dataset was created specifically to reflect a sample of customer evaluations and opinions, variance in product perception across geographic locations, and promotional purpose or bias in reviews.

The Parquet dataset, which is one of two accessible formats, including TSV, is partitioned (split into subfolders) on S3 by "product category" to increase further query speed. This enables queries that include a WHERE clause on "product category" only to read data from that category.

**AWS Services Used**:

**Amazon Redshift:**

Fully managed, petabyte-scale cloud data warehouse service. It also includes Redshift Spectrum that runs SQL queries directly against structured or unstructured data in Amazon S3 without loading them into Redshift cluster.

Redshift lets us run complex, analytic queries against structured data and semi-structured data, using sophisticated query optimization, columnar storage on high-performance storage like SSD, and massively parallel query execution.

Implementation:

AWS cloud based OLAP solution to store petabytes of information without owning infrastructure(Paas)

**Amazon VPC:**

Amazon Virtual Private Cloud (Amazon VPC) is a service that lets us launch AWS resources in a logically isolated virtual network we define. We have complete control over virtual networking environment, including selection of our own IP address range, creation of subnets, and configuration of route tables and network gateways.

There are no additional charges for creating and using the VPC itself. Usage charges for other Amazon Web Services, including

Amazon EC2, Elastic IP address still apply at published rates for those resources, including data transfer charges

Implementation:

All environments in AWS cloud. If VPC is not created by user then he/she is bound to use the default VPC for most of the products.

**Amazon Step Function:**

A workflow(or State Machine) of steps(or tasks) where the output of one step acts as an input to the next. Each step in an application executes in order, as defined by business logic.

With its built-in operational controls, Step Functions manages sequencing, error handling, retry logic, state management, parameter passing removing a significant operational burden from developers.

Implementation:

Mobile Apps management. Ex) Uber, Zomato, FoodPanda apps where right from order placement till order dispatch happens in sequence of steps. User’s new request-> check User’s details(for pending payment, authenticity)->check for

nearby service riders->check with restaurant for food availability->Place an order and assign a rider->Close the order->Feedback

**Amazon Glue:**

A serverless data integration service that makes it easy to discover, prepare, and combine data for analytics, machine learning, and application development.

Run Spark/Python code without managing Infrastructure at nominal cost. You pay only during run time of the job. Also pay storage cost for Data Catalog objects

Glue can automatically discover both structured and semi-structured data stored in your data lake on Amazon S3,data warehouse in Amazon Redshift, and various databases running on AWS/on-premises using JDBC. It provides a

unified view of your data via the Glue Data Catalog that is available for ETL, querying and reporting using services like Amazon Athena, Amazon EMR, and Amazon Redshift Spectrum.

**Amazon SNS(Simple Notification Service):**

Amazon Simple Notification Service is a notification service provided as part of Amazon Web Services since 2010. It provides a low-cost infrastructure for the mass delivery of messages, predominantly to mobile users.

Out of the box solution with low operational overhead to deliver notifications via following endpoints

● HTTP/HTTPS

● Email/Email-JSON

● Amazon Kinesis Data Firehose

● Amazon SQS

● AWS Lambda

● Platform application endpoint

● SMS

Implementation:

In Alarming applications

**AWS QuickSight:**

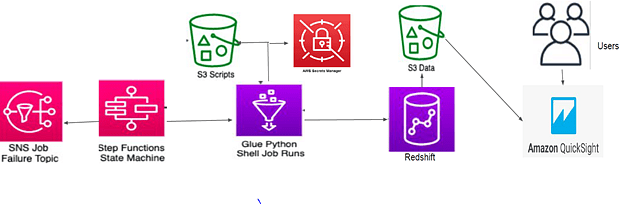
Amazon QuickSight is a scalable, serverless, embeddable, machine learning-powered business intelligence (BI) service built for the cloud.

The first BI service to offer pay-per-session pricing, where you only pay when your users access their dashboards or reports, making it cost-effective for large scale deployments. It can connect to wide variety of sources like Redshift, S3, Dynamo, RDS, files like JSON, text,csv, tsv, etc, jira, salesforce and on-premise oracle, sqlserver.

Implementation:

A visualization Paas tool available in AWS. It supports wide variety of charts like bar, pie, donut, scatterplot, heatmap, treemap, etc. It also has Autograph where Quicksight will pick a chart category based on columns chosen and theirdata format.

**Architecture & Workflow:**

****

**Redshift ETL workflow using Glue and Step Functions:**

Its very common to use Redshift as data-warehousing tool in AWS cloud. However, there are quite some ways to orchestrate the loading/unloading/querying Redshift. In this solution, we will use in-house AWS tools to orchestrate end-to-end loading and deriving business insights. Since it uses in-house tools, availability and durability of the solution is guaranteed by AWS.

1. The state machine launches a series of runs of an AWS Glue Python Shell job with parameters for retrieving database connection information from AWS Secrets Manager and an .sql file from S3.

2. Each run of the AWS Glue Python Shell job uses the database connection information to connect to the Amazon Redshift cluster and submit the queries contained in the .sql file.

For Task 1: The cluster utilizes Amazon Redshift Spectrum to read data from S3 and load it into an Amazon Redshift table. For Task 2: The cluster executes an aggregation query and exports the results to another Amazon S3 location via UNLOAD.

3. The state machine may send a notification to an Amazon Simple Notification Service (SNS) topic in the case of pipeline failure.

4. Users can query the data from the cluster and/or retrieve report output files directly from S3/Redshift using Quicksight.

**VPC**

1. Create a VPC in us-east-1 region with IPV4 CIDR range: 10.0.0.0/24

2. Create the following subnets:

a. Public Subnet A - 10.0.0.0/24

b. Private Subnet A - 10.0.16.0/24

c. Public Subnet B - 10.0.32.0/24

d. Private Subnet B - 10.0.48.0/24

3. Create Internet Gateway-> “myprojectgateway” and assign it to Public subnets A & B

4. Allocate a NAT Gateway-> “myprojectNAT” and allocate an Elastic IP address

5. Create 4 route tables, one for each subnet.

a. 2 Public subnets(A&B) will have Internet Gateway referred if traffic is routed for 0.0.0.0/0

b. 2 Private subnets(A&B) will have NAT Gateway referred if traffic is routed for 0.0.0.0/0

**IAM**

1. Create “myproject\_gluerole” with AWSGlueServiceRole and a new policy using glue\_demo\_policy.json(in project resources)

2. Create “myproject\_redshiftrole” with a new policy using redshift\_demo\_policy.json(in project resources).

3. Create “myproject\_stepfunctionsrole” with a new policy using step\_functions\_demo\_policy.json(in project resources)

**Redshift**

1. Create “myprojectredshiftcluster” as redshift cluster name after creating “Subnet Group” from Config tab. Subnet group should hold 2 private subnets(in 2 different regions) from VPC created. Choose default database name

as “reviews” instead of “dev”

2. Make a note of admin/master user name and password

3. Click on Editor (on left below Clusters)->Connect to Database->Create New Connection->Store the secret in

Secrets Manager->key in your cluster details->Give a name for the secret

4. Check the secrets manager for redshift secret. Use this secret as “db\_creds” parameter in glue job

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

**S3**

1. Create a new S3 bucket “redshift-databucketpro<junk numbers> to hold the extract from redshift(this is the

source for quicksight)

2. Create a new S3 bucket “redshift-databucketpro<junk numbers> and create 2 folders: python, sql

3. Upload following files from project resources section in the same hierarchy:

1) python

a) rs\_query.py

b) redshift\_module-0.1-py3.6.egg

2) sql

a) reviewsschema.sql

b) topreviews.sql

c) etl.sql

Graphical user interface, text, application

Description automatically generated

**Glue**

1. A connection in Glue Console pointing to “myprojectredshiftcluster” and test the connection using

“myproject\_gluerole” role. Check the policies if the connection fails(step 1 in IAM)

2. Create a new job using “Python Shell” option and choosing rs\_query.py(as source file), \*.egg file (as pythondependency file) and use the following job parameters:

--file sql/reviewsschema.sql (from step 2 in S3)

--db\_creds redshiftqueryeditor-awsuser-myredshiftsecret (from step 4 in Redshift)

--db reviews (from step 1 in Redshift)

--bucket redshift-bucketpro<junk numbers> (from step 2 in S3)

3. Select the connection created in step 1 for connecting to Redshift cluster

4. Run the job manually

5. Check the logs for any issues. Ensure to re-check the parameters for any naming issue

A screenshot of a computer

Description automatically generated

**SNS**

1. Create a Standard Simple Notification Service(SNS) named “alarm-topic” and copy the arn from topic dashboard page

2. Create a subscription to SNS topic by adding your email-id and “EMAIL” as option

3. Accept subscription request by logging into your email account. Only then you will start receiving messages

Graphical user interface, text, application, email

Description automatically generated

**Step Functions**

4. Create a state machine and copy paste state\_machine.json(in project resources) and save the definition

using “Myproject\_StepFunction” name

5. Update the definition with the parameter values similar to your succeeded Glue job except for “file”

parameter

6. Update the SNS arn to your SNS topic arn (Step 1 from SNS)

7. Attach the “myproject\_stepfunctionsrole” role to state machine (step 3 from IAM)

8. Click “Start Execution” button and check out the job status

9. Ensure Glue manual run succeeded before coming to this step. Then ensure policy and parameters in the state machine definition is correct as per your environment (Choose “Edit Definition” to alter the JSON file any time you want.

Graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generatedA picture containing graphical user interface

Description automatically generated

**Quicksight**

1. Activate Quicksight subscription under “Standard” edition

2. Choose the same region as your S3 account by clicking on right top corner Account name dropdown

3. Choose Manage Quicksight(right top corner Account name dropdown)-> Choose S3 bucket that you want to

connect to

4. Click on “Create Dataset” and add “S3” and upload the manifest.json(in project resources by altering the

name of the file according to your extract in redshift\_databucket<junk numbers>

5. Rename columns to appropriate ones to avoid confusion as follows:

Old Column: New Column

Column 1: marketplace

Column 2: product\_category

Column 3: product\_title

Column 4: review\_id

Column 5: helpful\_votes

Column 6: average\_stars

6. Click on Analysis-> create a new one-> add “Add Visuals” on top left to add more charts and populate it by dragging and dropping corresponding column names

**Code:**

**Python code(Used in Glue):**

from redshift\_module import pygresql\_redshift\_common as rs\_common

import sys

from awsglue.utils import getResolvedOptions

import boto3

#get job args

args = getResolvedOptions(sys.argv,['db','db\_creds','bucket','file'])

db = args['db']

db\_creds = args['db\_creds']

bucket = args['bucket']

file = args['file']

#get sql statements

s3 = boto3.client('s3')

sqls = s3.get\_object(Bucket=bucket, Key=file)['Body'].read().decode('utf-8')

sqls = sqls.split(';')

#get database connection

print('connecting...')

con = rs\_common.get\_connection(db,db\_creds)

#run each sql statement

print("connected...running query...")

results = []

for sql in sqls[:-1]:

sql = sql + ';'

result = rs\_common.query(con, sql)

print(result)

results.append(result)

print(results)

**Sql scripts:**

Script1: (To create external schema and table in redshift cluster)

CREATE EXTERNAL SCHEMA amzreviews

from data catalog

database 'amzreviews'

iam\_role 'arn:aws:iam::647616946879:role/myredshiftrole'

CREATE EXTERNAL database IF NOT EXISTS;

CREATE EXTERNAL TABLE amzreviews.reviews(

marketplace varchar(10),

customer\_id varchar(15),

review\_id varchar(15),

product\_id varchar(25),

product\_parent varchar(15),

product\_title varchar(50),

star\_rating int,

helpful\_votes int,

total\_votes int,

vine varchar(5),

verified\_purchase varchar(5),

review\_headline varchar(25),

review\_body varchar(1024),

review\_date date,

year int)

PARTITIONED BY (

product\_category varchar(25))

ROW FORMAT SERDE

'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'

STORED AS INPUTFORMAT

'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'

OUTPUTFORMAT

'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'

LOCATION

's3://amazon-reviews-pds/parquet/';

ALTER TABLE amzreviews.reviews ADD

partition(product\_category='Apparel')

location 's3://amazon-reviews-pds/parquet/product\_category=Apparel/'

partition(product\_category='Automotive')

location 's3://amazon-reviews-pds/parquet/product\_category=Automotive'

partition(product\_category='Baby')

location 's3://amazon-reviews-pds/parquet/product\_category=Baby'

partition(product\_category='Beauty')

location 's3://amazon-reviews-pds/parquet/product\_category=Beauty'

partition(product\_category='Books')

location 's3://amazon-reviews-pds/parquet/product\_category=Books'

partition(product\_category='Camera')

location 's3://amazon-reviews-pds/parquet/product\_category=Camera'

partition(product\_category='Grocery')

location 's3://amazon-reviews-pds/parquet/product\_category=Grocery'

partition(product\_category='Furniture')

location 's3://amazon-reviews-pds/parquet/product\_category=Furniture'

partition(product\_category='Watches')

location 's3://amazon-reviews-pds/parquet/product\_category=Watches'

partition(product\_category='Lawn\_and\_Garden')

location 's3://amazon-reviews-pds/parquet/product\_category=Lawn\_and\_Garden';

CREATE TABLE reviews(

marketplace varchar(10),

customer\_id varchar(15),

review\_id varchar(15),

product\_id varchar(25) DISTKEY,

product\_parent varchar(15),

product\_title varchar(50),

star\_rating int,

helpful\_votes int,

total\_votes int,

vine varchar(5),

verified\_purchase varchar(5),

review\_date date,

year int,

product\_category varchar(25))

SORTKEY (

review\_date

);

Script 2: (Extract, Transform and Load)

INSERT INTO public.reviews

SELECT marketplace, customer\_id, review\_id, product\_id, product\_parent, product\_title, star\_rating, helpful\_votes, total\_votes, vine, verified\_purchase, review\_date, year, product\_category

FROM amzreviews.reviews where year>=2015

Script 3: (loading to S3)

UNLOAD ('SELECT marketplace, product\_category, product\_title, review\_id, helpful\_votes, AVG(star\_rating) as average\_stars FROM public.reviews GROUP BY marketplace, product\_category, product\_title, review\_id, helpful\_votes ORDER BY helpful\_votes DESC, average\_stars DESC')

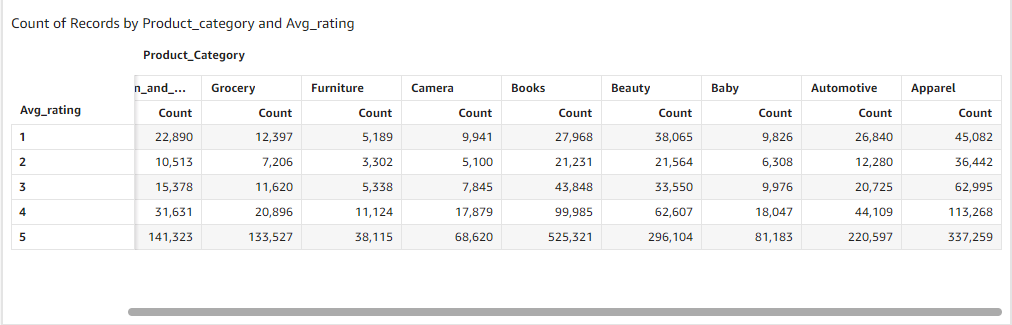
TO 's3://redshift-databucketpro-result/testunload/'

iam\_role 'arn:aws:iam::647616946879:role/myredshiftrole';

**Visualization:**

**Chart

Description automatically generated**

****

**Chart, bar chart

Description automatically generated**

**Chart, pie chart

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Dashboard Link:** [**https://us-east-1.quicksight.aws.amazon.com/sn/dashboards/a64c588c-0086-4912-bb14-d536236bbb0f**](https://us-east-1.quicksight.aws.amazon.com/sn/dashboards/a64c588c-0086-4912-bb14-d536236bbb0f)

**Github Link:** [**https://github.com/siddharth414/sjsu-data228**](https://github.com/siddharth414/sjsu-data228)

**Conclusion:**

Customer reviews play a vital role in the business. Reviews not only have the power to influence consumer decisions but can strengthen a company's credibility. Reviews have the power to gain customer trust. Increase in customer interaction ultimately leads to improved profits for businesses.

So, we can conclude that not only sales data even customer reviews data plays a prominent role in driving the business.

**References:**

* [1.https://s3.amazonaws.com/amazon-reviews-pds/tsv/sample\_us.tsv](file:///C:\Users\Checkout\Downloads\1.%09https:\s3.amazonaws.com\amazon-reviews-pds\tsv\sample_us.tsv)
* <https://aws.amazon.com/blogs/big-data/orchestrate-amazon-redshift-based-etl-workflows-with-aws-step-functions-and-aws-glue/>
* <https://docs.aws.amazon.com/step-functions/latest/dg/sample-etl-orchestration.html>
* <https://docs.aws.amazon.com/prescriptive-guidance/latest/patterns/orchestrate-an-etl-pipeline-with-validation-transformation-and-partitioning-using-aws-step-functions.html>