**E-commerce Data Warehouse in HIVE using AWS**

This big data project will look at Hive's capabilities to run analytical queries on massive datasets. Using only sales and Customer demographics data from the Adventure works dataset to perform analysis and answer the following questions:

* To find the upper and lower discount limits offered for any product
* Sales contributions by customer
* To Understand customer persona purchasing pattern based on gender, education and yearly income
* To find the sales contribution by customers on the overall year to date sales belong to categorized by same gender, yearly income.
* To identify the top performing territory based on sales
* To find the territory-wise sales and their adherence to the defined sales quota.

**Aim**

To perform Hive analytics on Sales and Customer Demographics data using big data tools such as Sqoop, Spark, and HDFS.

**Data Description**

Adventure Works is a free sample database of retail sales data. In this project, we will be only using Customer test, Individual test, Credit card, Sales order details, Store, Sales territory, Salesperson, Sales order header, Special offer tables from this database.

**Tech Stack**

➔ Language: SQL, Scala

➔ Services: AWS EC2, Docker, MySQL, Sqoop, Hive, HDFS, Spark

**Approach**

* Create an AWS EC2 instance and launch it.
* Create docker images using docker-compose file on EC2 machine via SSH.
* Create tables in MySQL.
* Load data from MySQL into HDFS storage using Sqoop commands.
* Move data from HDFS to Hive.
* Using Scala programming language, extract Customer demographics information from data and store it as parquet files.
* Move parquet files from Spark to Hive.
* Create tables in Hive and load data from Parquet files into tables.
* Perform Hive analytics on Sales and Customer demographics data.

**Team**

Student Name: Srividya Katam

Number of Students in Team: One

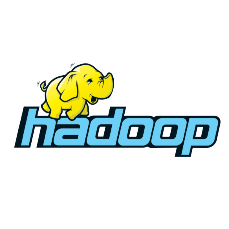
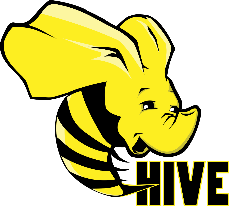
**Architecture**

HIVE data warehouse created to host data and perform analytics.

Data transformed to HDFS files in AWS docker containers

Relational data uploaded to MySQL database in AWS

Relational data in SQL Server



Send Customer Demographics flattened data

Send Customer Demographics XML data

**Implementation Details**

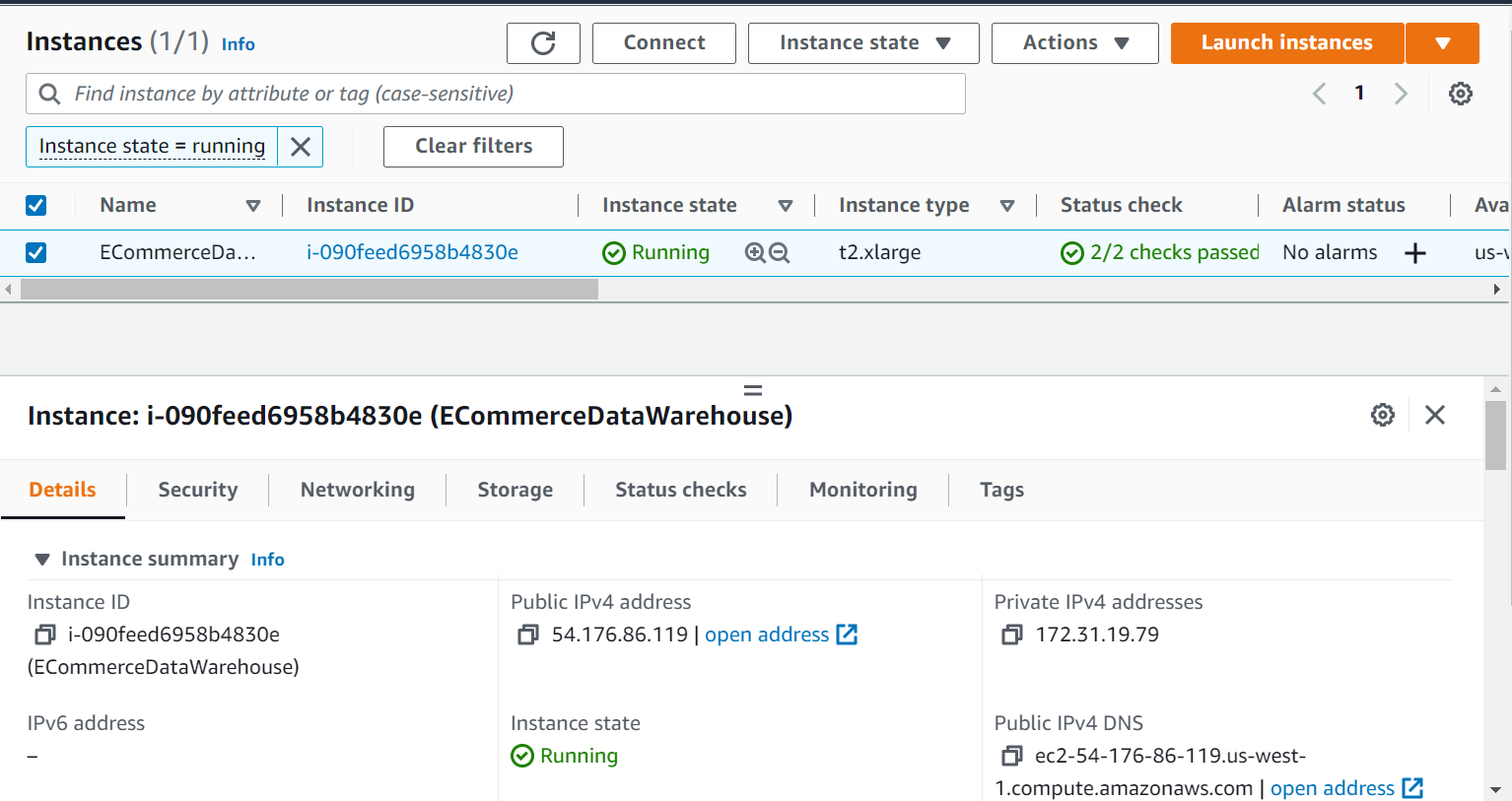
* Create an AWS EC2 instance and launch it.
* Create docker images using docker-compose file on EC2 machine via SSH.
* Create tables in MySQL.
* Load data from MySQL into HDFS storage using Sqoop commands.
* Move data from HDFS to Hive.
* Using Scala programming language, extract Customer demographics information from data and store it as parquet files.
* Move parquet files from Spark to Hive.
* Create tables in Hive and load data from Parquet files into tables.
* Perform Hive analytics on Sales and Customer demographics data.

**AWS Instance:**

Create t2.2xlarge ec2 instance with Amazon Linux 2 AMI (HVM) and 96GB storage.

Created pem file for security keys.

Added security group to allow all inbound traffic for easy evaluation of project.



**Docker containers:**

Using following commands, install docker in the EC2 instance.   
Updated docker-compose.yml with container names, user name and passwords for each container.

Attached docker set up files in “Installation” folder.

sudo yum update -y

sudo yum install docker

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

sudo gpasswd -a $USER docker

newgrp docker

**Data copy from SQL to MySQL:**

1. Created data copy files to insert relational data into MySQL database in AWS. Attached SQL files in “Code” folder.

**Creating HDFS files from MySQL:**

1. Created Sqoop jobs to read data from MySQL and create HDFS files. Attached Sqoop jobs. Refer to “Sqoop-import.txt” in “Code” folder

**Installing Hive and Spark Dependencies:**

1. Copy hive-site.xml file copied to Spark container conf folder.
2. Download Postgresql JAR from official website and copy to EC2 cluster and then to spark container jars folder.

**Create Hive tables:**

1. Using the HDFS files created, create and load tables in Hive. Refer to “04\_Hive\_tables\_creation(cust,sales,stores).hql” in “Code” folder.

**Using Spark to extract Customer Demographics XML data:**

1. Extract customer demographics XML data and flatten using Scala program in Spark. Refer to “05\_customer\_demographic.scala” file in “code” folder.

**Copy data from Spark container to Hive container:**

1. Using file copy commands, copy data from Spark container to Hive container. Refer to “06\_File\_copy\_commands (from spark container to hive container).txt” in “code” folder.

**Creating Customer Demographics table from parquet file:**

1. From the file copied in above step, create new table customer demographics with flattened data. Refer to “07\_customer\_demograhics\_creation.hql.txt” in “code” folder.

**Hive analytics queries**

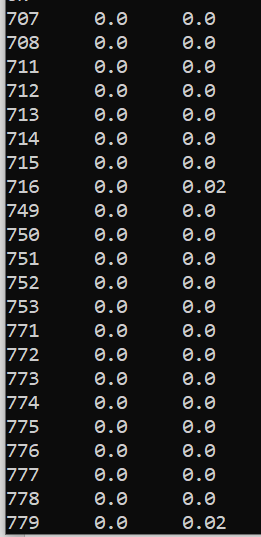
Performed Hive analytics to answer questions discussed above.

* To find the upper and lower discount limits offered for any product  
  **Query:**

select productid, min(discountpct) as min\_discount, max(discountpct) as max\_discount

from sales\_order\_details

group by productid

**Output:** Most products have minimum discount of zero and maximum discount of 2%.  


* Sales contributions by customer  
  **Query:**

select soh.CustomerID , sum(soh.SubTotal) subtotal,

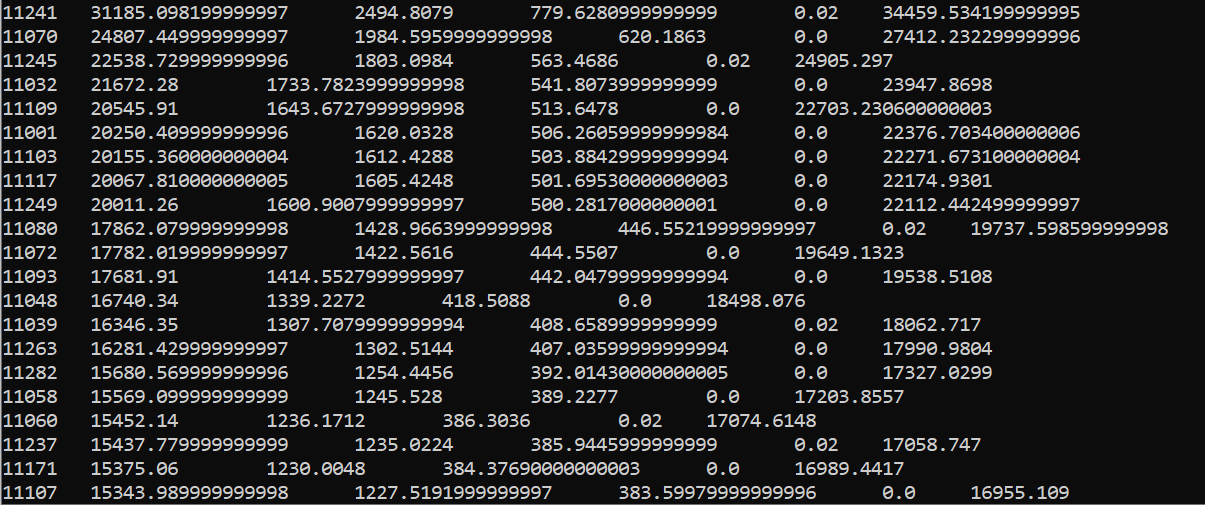
sum(soh.TaxAmt) Taxamt, sum(soh.Freight) Freight,

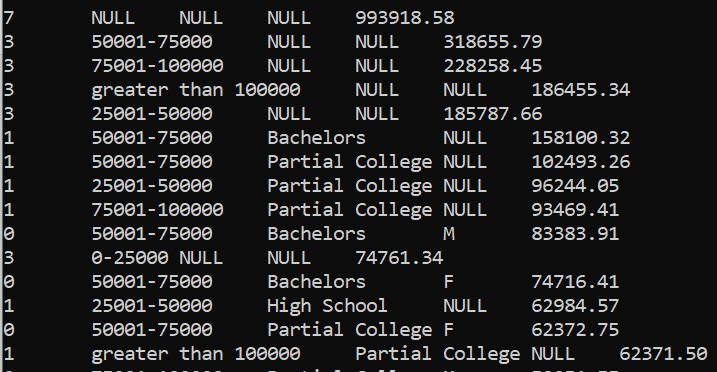
sum(sod.DiscountPct) discountpercent, sum(soh.TotalDue) Totaldue

from sales\_order\_details sod join sales\_order\_header soh on sod.salesorderid = soh.salesorderid

group by soh.CustomerID

order by Totaldue desc;

**Output:** The customer 11241 has contributed to most sales.  


* To Understand customer persona purchasing pattern based on gender, education and yearly income  
  Query:  
  **Output**: The customers with yearly income of 50001 – 75000 are making most purchases.  
  

select grouping\_\_id, yearlyincome, education, gender, sum(totalpurchaseytd) sales\_value from customer\_demo

group by yearlyincome, education,gender

with rollup

order by sales\_value desc;

* To find the sales contribution by customers on the overall year to date sales belong to categorized by same gender, yearly income.

Query:

select gender, yearlyincome,

sum(v.percentage) as percentage\_of\_purchase

from customer\_demo cd

join

(select customerid,

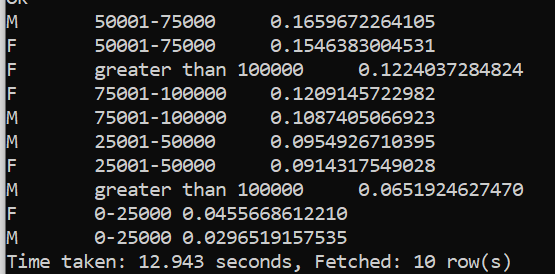
(sum(totalpurchaseytd) over (partition by customerid) / sum(totalpurchaseytd) over ()) percentage

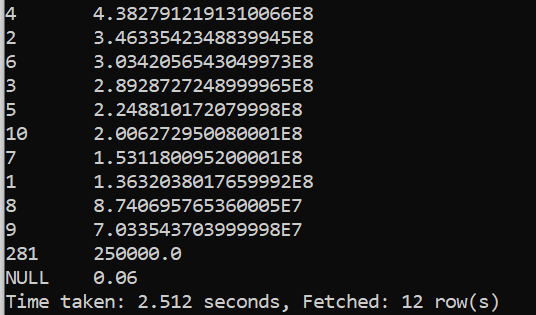
from customer\_demo cd) as v

on v.customerid = cd.customerid

group by gender, yearlyincome

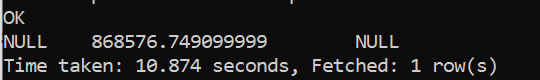
order by percentage\_of\_purchase desc;

**Output:** Males with yearly income of 50001-75000 have contributed most in the year to date sales. ****

* To identify the top performing territory based on sales  
  Query:  
  **Output:** Territory id 4 has the highest sales recorded.  
  

select TerritoryID,sum(SalesYTD) t\_sales from store\_details

group by TerritoryID order by t\_sales desc;

* To find the territory-wise sales and their adherence to the defined sales quota.  
  Query:  
  **Output:**  
  

select soh.TerritoryID,sum(soh.TotalDue), sum(stores.targets\_acheived) as target\_completed

from sales\_order\_header soh

left join

(select TerritoryID, (sum(SalesYTD) over (partition by TerritoryID) / sum(SalesQuota) over ()) targets\_acheived from store\_details) as stores

on stores.TerritoryID = soh.TerritoryID

group by soh.TerritoryID;