

Hive Case Study- DA Track

Submitted by

- 1. Sukeerthi G**
- 2. Shubham Aggarwal**

E-Commerce Hive Case Study

Problem Statement:

With online sales gaining popularity, tech companies are exploring ways to improve their sales by analyzing customer behavior and gaining insights about product trends. Furthermore, the websites make it easier for customers to find the products they require without much scavenging. Needless to say, the role of big data analysts is among the most sought-after job profiles of this decade. Therefore, as part of this assignment, it is expected to extract data and gather insights from a real-life data set of an e-commerce company.

Dataset:

The dataset used for this case study is available in the following links,

<https://e-commerce-events-ml.s3.amazonaws.com/2019-Oct.csv>

<https://e-commerce-events-ml.s3.amazonaws.com/2019-Nov.csv>

Attribute Description:

| Attribute_Name | Data_type | Description |
|----------------|-----------|--|
| event_time | timestamp | Time at which the event took place |
| event_type | string | Event type may be 'view', 'cart', 'remove_from_cart', 'purchase' |
| product_id | string | Unique identification of the product |
| category_id | string | Unique identification of the product category. Each product category contains several products |
| category_code | string | Name (if present) of the product category |
| brand | string | Name of the brand |
| price | float | Price of the product |
| user_id | bigint | Permanent user id |
| user_session | string | Identification for the user's session. Remains same for each user's session. It changes every time the user returns back to the website after a long pause |

Table 1 Attribute Description

Importing the data to HDFS:

Step 1 : Creating S3 bucket and loading the dataset:

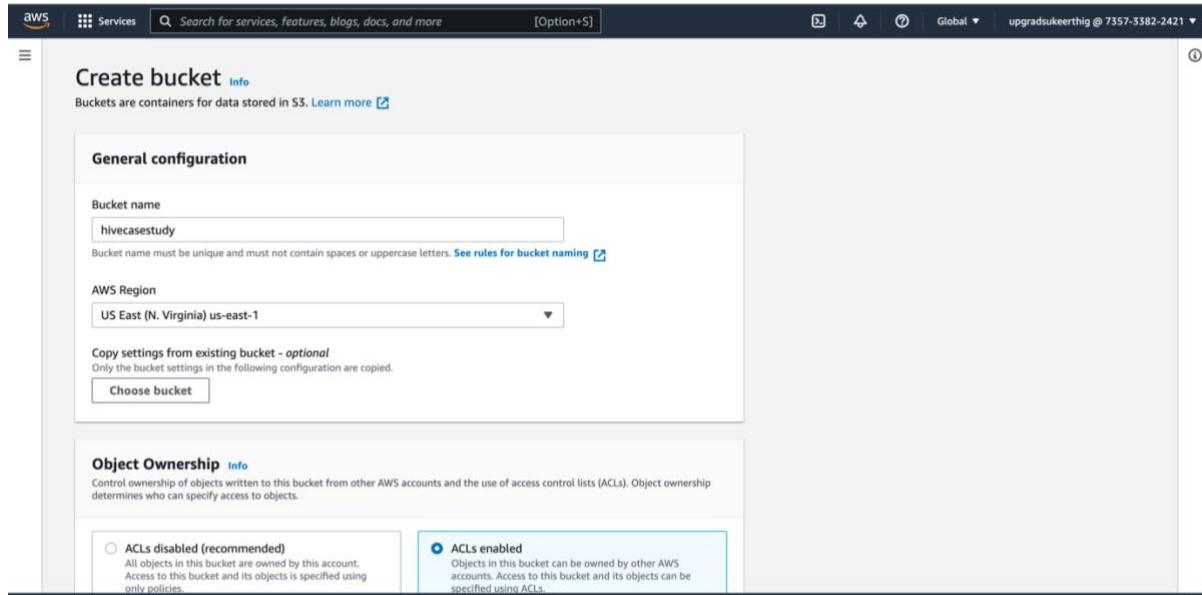


Figure 1 Bucket Creation

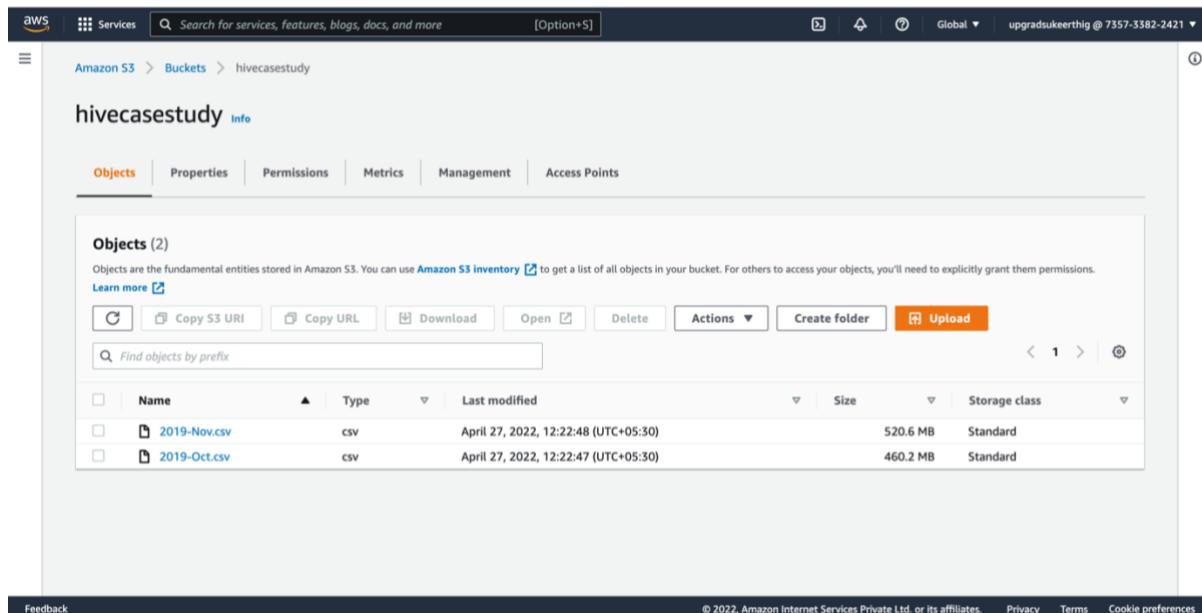


Figure 2 Uploading data into the S3 bucket

Step 2: Creation of Key-Pair:

Create key pair Info

Key pair
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type Info
 RSA
 ED25519

Private key file format
 .pem
For use with OpenSSH
 .ppk
For use with PuTTY

Tags (Optional)
No tags associated with the resource.

You can add up to 50 more tags.

Figure 3 Key-pair configurations

Key pairs (1) Info

Successfully created key pair

| <input type="checkbox"/> | Name | Type | Fingerprint | ID |
|--------------------------|---------------------|------|--|-----------------------|
| <input type="checkbox"/> | case_study_key_pair | rsa | 06:6e:0a:99:b3:c0:f9:94:a3:42:d0:7c:b... | key-0a8815b84d310315d |

Figure 4 Key-Pair creation

Step 3: Creation of EMR Cluster:

The screenshot shows the 'Create Cluster - Advanced Options' page in the AWS Management Console. The 'Software Configuration' section is displayed, with the 'Release' dropdown set to 'emr-5.29.0'. Under this release, several software components are listed with checkboxes for selection. The selected components are Hadoop 2.8.5, Hive 2.3.6, Hue 4.4.0, Zeppelin 0.8.2, Tez 0.9.2, HBase 1.4.10, Presto 0.227, Sqoop 1.4.7, Phoenix 4.14.3, HCatalog 2.3.6, Pig 0.17.0, ZooKeeper 3.4.14, and TensorFlow 1.14.0. There are also sections for 'Multiple master nodes (optional)' and 'AWS Glue Data Catalog settings (optional)'. A text input field for 'Edit software settings' contains the configuration string: `classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]`. At the bottom, there is a note about steps and a feedback link.

Figure 5 Software Configuration

This screenshot shows the same 'Create Cluster - Advanced Options' page as Figure 5, but with different configuration settings. The 'Software Configuration' section remains largely the same, with the 'Release' dropdown still set to 'emr-5.29.0'. However, the checked software components are different: Hadoop 2.8.5, Hive 2.3.6, Hue 4.4.0, Zeppelin 0.8.2, Tez 0.9.2, HBase 1.4.10, Presto 0.227, Sqoop 1.4.7, Phoenix 4.14.3, HCatalog 2.3.6, Flink 1.9.1, ZooKeeper 3.4.14, and TensorFlow 1.14.0. The 'Edit software settings' field now contains the configuration string: `classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]`. The rest of the page, including the note about steps and the feedback link, is identical to Figure 5.

Figure 6 Advanced Options

Cluster Nodes and Instances

Choose the instance type, number of instances, and a purchasing option. [Learn more about instance purchasing options](#)

Console options for automatic scaling have changed. [Learn more](#)

| Node type | Instance type | Instance count | Purchasing option |
|---|--|----------------|--|
| Master Master - 1 | m4.large 2 vCore, 8 GiB memory, EBS only storage EBS Storage: 32 GiB | 1 Instances | <input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price |
| Core Core - 2 | m4.large 2 vCore, 8 GiB memory, EBS only storage EBS Storage: 32 GiB | 1 Instances | <input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price |
| + Add task instance group | | | |
| Total core and task units | | 1 Total units | |

Cluster scaling

Adjust the number of Amazon EC2 instances available to an EMR cluster via EMR-managed scaling or a custom automatic scaling policy. [Learn more](#)

Feedback © 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

Figure 7 Cluster node hardware settings

Step 1: Software and Steps

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

General Options

Cluster name:

Logging
 Debugging
 Termination protection

Tags

| Key | Value (optional) |
|--|----------------------|
| <input type="text" value="Add a key to create a tag"/> | <input type="text"/> |

Additional Options

EMRFS consistent view
 Custom AMI ID

Bootstrap Actions

Cancel Previous Next

Feedback © 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

Figure 8 General cluster settings

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia upgradsukeerthig @ 7357-3382-2421

Create Cluster - Advanced Options Go to quick options

Step 1: Software and Steps Step 2: Hardware Step 3: General Cluster Settings Step 4: Security

Security Options

EC2 key pair case_study_key_pair Cluster visible to all IAM users in account

Permissions Default Custom Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role EMR_DefaultRole Use EMR_DefaultRole_V2

EC2 instance profile EMR_EC2_DefaultRole Auto Scaling role EMR_AutoScaling_DefaultRole

► Security Configuration
► EC2 security groups

Create cluster

Feedback © 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

Figure 9 Security Options

AWS Services Search for services, features, blogs, docs, and more [Option+S] N. Virginia upgradsukeerthig @ 7357-3382-2421

Amazon EMR Cluster: hive_case_study Starting

Clone Terminate AWS CLI export Auto-termination is not available for this account when using this release of EMR.

SSH Connect to the Master Node Using SSH You can connect to the Amazon EMR master node using SSH to run interactive queries, examine log files, submit Linux commands, and so on. Learn more

Windows Mac / Linux

- Open a terminal window. On Mac OS X, choose Applications > Utilities > Terminal. On other Linux distributions, terminal is typically found at Applications > Accessories > Terminal.
- To establish a connection to the master node, type the following command. Replace ~/case_study_key_pair.pem with the location and filename of the private key file (.pem) used to launch the cluster.
ssh -i ~/case_study_key_pair.pem hadoop@ec2-44-202-179-149.compute-1.amazonaws.com
- Type yes to dismiss the security warning.

Close Cluster scaling: Not enabled

Security and access Key name: case_study_key_pair EC2 instance profile: EMR_EC2_DefaultRole EMR role: EMR_DefaultRole Auto Scaling role: EMR_AutoScaling_DefaultRole

Feedback © 2022, Amazon Internet Services Private Ltd. or its affiliates. Privacy Terms Cookie preferences

Figure 10 SSH command for connecting with the master node

The screenshot shows the AWS EMR Cluster details page. The cluster is named 'hive_case_study' and is currently 'Running'. Key details include:

- Summary:** ID: j-L5PGGTP4R9SH, Creation date: 2022-04-30 18:58 (UTC+5:30), Elapsed time: 12 minutes.
- Configuration details:** Release label: emr-5.29.0, Hadoop distribution: Amazon 2.8.5, Applications: Hive 2.3.6, Pig 0.17.0, Hue 4.4.0, Log URI: s3://aws-logs-735733822421-us-east-1/elasticsearch/reduce/, EMRFS consistent view: Disabled, Custom AMI ID: --.
- Network and hardware:** Availability zone: us-east-1b, Subnet ID: subnet-2b60e44d, Master: Bootstrapping 1 m4.large, Core: Provisioning 1 m4.large, Task: --, Cluster scaling: Not enabled.
- Security and access:** Key name: case_study_key_pair, EC2 instance profile: EMR_EC2_DefaultRole, EMR role: EMR_DefaultRole, Auto Scaling role: EMR_AutoScaling_DefaultRole, Visible to all users: All Change, Security groups for Master: sg-02725556efcb02585 (ElasticMapReduce-master), Security groups for Core & Task: sg-02cc257141c2bc031 (ElasticMapReduce-task).

Figure 11 Cluster running

The screenshot shows the AWS EC2 Inbound rules page. There are 19 rules listed, including:

| Name | Security group rule... | IP version | Type | Protocol | Port range |
|------------------------|-------------------------|------------|-----------------|----------|------------|
| sgr-06961441109a1cf86 | sg-r-06961441109a1cf86 | IPv4 | Custom TCP | TCP | 8443 |
| sgr-05d140d2a1aa125... | sg-r-05d140d2a1aa125... | IPv4 | Custom TCP | TCP | 8443 |
| sgr-08d10bb11fb5d63... | sg-r-08d10bb11fb5d63... | IPv4 | Custom TCP | TCP | 8443 |
| sgr-0fe6a0d641900be9 | sg-r-0fe6a0d641900be9 | - | All ICMP - IPv4 | ICMP | All |
| sgr-0870bdedeb4e8f7cd | sg-r-0870bdedeb4e8f7cd | IPv4 | Custom TCP | TCP | 8443 |
| sgr-0cb8fc0b5ecbc2a2c | sg-r-0cb8fc0b5ecbc2a2c | IPv4 | Custom TCP | TCP | 8443 |
| sgr-0435485ac322463c1 | sg-r-0435485ac322463c1 | IPv4 | Custom TCP | TCP | 8443 |
| sgr-038ea7b6c4a6c6008 | sg-r-038ea7b6c4a6c6008 | IPv4 | SSH | TCP | 22 |
| sgr-06e80694223043... | sg-r-06e80694223043... | - | All UDP | UDP | 0 - 65535 |
| sgr-045467adb41aba7... | sg-r-045467adb41aba7... | - | All UDP | UDP | 0 - 65535 |
| sgr-0b0fd98e7d3e7bba | sg-r-0b0fd98e7d3e7bba | - | All ICMP - IPv4 | ICMP | All |
| sgr-01b3fa0acd6aeb944 | sg-r-01b3fa0acd6aeb944 | - | All TCP | TCP | 0 - 65535 |
| sgr-03c82e017ed9dd6... | sg-r-03c82e017ed9dd6... | - | All TCP | TCP | 0 - 65535 |
| sgr-08e91233c99edb4... | sg-r-08e91233c99edb4... | IPv4 | Custom TCP | TCP | 8443 |
| sgr-063c0d25553bf51ff | sg-r-063c0d25553bf51ff | IPv4 | Custom TCP | TCP | 8443 |
| sgr-a8c70616e823b6... | sg-r-a8c70616e823b6... | IPv4 | Custom TCP | TCP | 8443 |
| sgr-059273d2404426... | sg-r-059273d2404426... | IPv4 | Custom TCP | TCP | 8443 |
| sgr-0345453afaaa5c375 | sg-r-0345453afaaa5c375 | IPv4 | Custom TCP | TCP | 8443 |

Figure 12 Inbound rules already updated to SSH with TCP at port 22 in the past

Step 4: Connecting to the master node through terminal window

Command:

cd /Users/sukeerthig/Downloads

chmod 400 case_study_key_pair.pem

```
ssh -i case_study_key_pair.pem hadoop@ec2-44-201-28-81.compute-1.amazonaws.com
```

```
Last login: Sat Apr 30 18:26:03 on console
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Sukeerthis-MacBook-Pro:~ sukeerthig$ cd /Users/sukeerthig/Downloads
(base) Sukeerthis-MacBook-Pro:Downloads sukeerthig$ chmod 400 case_study_key_pair.pem
(base) Sukeerthis-MacBook-Pro:Downloads sukeerthig$ ssh -i case_study_key_pair.pem hadoop@ec2-44-201-28-81.compute-1.amazonaws.com
The authenticity of host 'ec2-44-201-28-81.compute-1.amazonaws.com (44.201.28.81)' can't be established.
ED25519 key fingerprint is SHA256:UjNjMCfge+y4wARXVaRqmcNb/R2OpdZGx7LMc1HMzyw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-201-28-81.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

 _I _I_ )
 _I ( _ / Amazon Linux AMI
 __\_\_I_|

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
72 package(s) needed for security, out of 102 available
Run "sudo yum update" to apply all updates.
-bash: warning: setlocale: LC_CTYPE: cannot change locale (UTF-8): No such file or directory

EEEEEEEEEEEEEEEEEE MMMMM RRRRRRRRRRRRRR
E:::::::::::E M:::::M M:::::M R:::::R
EE:::::EEEEEEEEE:::E M:::::M M:::::M R:::::RRRRR:::::R
E:::::E EEEE M:::::M M:::::M RR:::R R:::R
E:::::E M:::::M:::M M:::M::::M R:::R R:::R
E:::::EEEEEEEEE E M::::M M:::M M::::M R:::::RRRRR:::::R
E:::::EEEEEEEEE M::::M M:::M M::::M R:::::RR
E:::::EEEEEEEEE M::::M M::::M M::::M R:::::RRRRR:::::R
E:::::E M::::M M:::M M::::M R:::R R:::R
E:::::E EEEE M::::M MMM M::::M R:::R R:::R
EE:::::EEEEEEEEE:::E M::::M M::::M R:::::R R:::R
E:::::::::::E M::::M M::::M RR:::R R:::::R
EEEEEEEEEEEEEEEEEE MMMMM RRRRRRR RRRRRR

[hadoop@ip-172-31-14-147 ~]$
```

Figure 13 Successfully connecting to the master node

Command:

```
hadoop fs -mkdir /hive_case_study
```

```
hadoop fs -ls /
```

```
[hadoop@ip-172-31-14-147 ~]$ hadoop fs -ls /
Found 4 items
drwxr-xr-x  - hdfs hadoop          0 2022-04-30 13:36 /apps
drwxrwxrwt  - hdfs hadoop          0 2022-04-30 13:39 /tmp
drwxr-xr-x  - hdfs hadoop          0 2022-04-30 13:36 /user
drwxr-xr-x  - hdfs hadoop          0 2022-04-30 13:36 /var
[hadoop@ip-172-31-14-147 ~]$ hadoop fs -mkdir /hive_case_study
[hadoop@ip-172-31-14-147 ~]$ hadoop fs -ls /
Found 5 items
drwxr-xr-x  - hdfs hadoop          0 2022-04-30 13:36 /apps
drwxr-xr-x  - hadoop hadoop         0 2022-04-30 13:43 /hive_case_study
drwxrwxrwt  - hdfs hadoop          0 2022-04-30 13:39 /tmp
drwxr-xr-x  - hdfs hadoop          0 2022-04-30 13:36 /user
drwxr-xr-x  - hdfs hadoop          0 2022-04-30 13:36 /var
[hadoop@ip-172-31-14-147 ~]$
```

Figure 14 Creating directory for the case study

Command:

```
hadoop distcp s3://hivecasestudy/2019-Nov.csv /hive_case_study
```

```
hadoop distcp s3://hivecasestudy/2019-Oct.csv /hive_case_study
```

```
hadoop fs -ls /hive_case_study
```

```
[hadoop@ip-172-31-14-147 ~]$ hadoop fs -ls /hive_case_study
Found 2 items
-rw-r--r-- 1 hadoop hadoop 545839412 2022-04-30 13:51 /hive_case_study/2019-Nov.csv
-rw-r--r-- 1 hadoop hadoop 482542278 2022-04-30 13:52 /hive_case_study/2019-Oct.csv
[hadoop@ip-172-31-14-147 ~]$
```

Figure 15 Moving the data from S3 bucket to HDFS

Creating Suitable Hive tables:

Step 1: Creation of database

Command:

```
CREATE DATABASE IF NOT EXISTS clickstream_data ;
```

```
SHOW DATABASES ;
```

```
[hadoop@ip-172-31-14-147 ~]$ hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> SHOW DATABASES ;
OK
default
Time taken: 1.087 seconds, Fetched: 1 row(s)
hive> CREATE DATABASE IF NOT EXISTS clickstream_data ;
OK
Time taken: 0.783 seconds
hive> SHOW DATABASES ;
OK
clickstream_data
default
Time taken: 0.023 seconds, Fetched: 2 row(s)
hive>
```

Figure 16 Database creation

Step 2: Creating and inserting the data into the table

Command:

```
USE clickstream_data ;
```

```
CREATE TABLE IF NOT EXISTS ecommerce (event_time timestamp, event_type string,
product_id string, category_id string, category_code string, brand string, price float, user_id
bigint, user_session string) ROW FORMAT SERDE
'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE LOCATION
'/hive_case_study' TBLPROPERTIES ('serialization.null.format'='', 'skip.header.line.count'=1');
```

```
SET hive.cli.print.header=true;
```

```
SELECT * FROM ecommerce LIMIT 5 ;
```

```
hive> USE clickstream_data ;
OK
Time taken: 0.063 seconds
hive> CREATE TABLE IF NOT EXISTS ecommerce (event_time timestamp, event_type string, product_id string, category_id string, category_code string, brand string, price float, user_id bigint, user_session string) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE LOCATION '/hive_case_study' TBLPROPERTIES ('serialization.null.format=''','skip.header.line.count='1') ;
OK
Time taken: 0.526 seconds
hive> SET hive.cli.print.header=true ;
hive> SELECT * FROM ecommerce LIMIT 5 ;
OK
ecommerce.event_time    ecommerce.event_type    ecommerce.product_id    ecommerce.category_id    ecommerce.category_code    ecommerce.brand    ecommerce.price    ecommerce.user_id    ecommerce.user_session
2019-11-01 00:00:02 UTC view      5802432 1487580009286598681          0.32     562076640    09fafd6c-6c99-46b1-834f-33527f4de241
2019-11-01 00:00:09 UTC cart     5844397 1487580006317032337          2.38     553329724    2067216c-31b5-455d-a1cc-af0575a34ffb
2019-11-01 00:00:10 UTC view      5837166 1783999064103190764          pnb      22.22     556138645    57ed22e-a54a-4907-9944-5a875c2d7f4f
2019-11-01 00:00:11 UTC cart     5876812 1487580010100293687          jessnail  3.16     564506666    186c1951-8052-4b37-adce-dd9644b1d5f7
2019-11-01 00:00:24 UTC remove_from_cart  5826182 1487580007483048900          3.33     553329724    2067216c-31b5-455d-a1cc-af0575a34ffb
Time taken: 2.32 seconds, Fetched: 5 row(s)
hive>
```

Figure 17 Creating and uploading data into the table

Optimization Techniques

Command:

```
SET hive.exec.dynamic.partition =true;
```

```
SET hive.exec.dynamic.partition.mode =nonstrict;
```

```
|hive> set hive.exec.dynamic.partition =true;
|hive> set hive.exec.dynamic.partition.mode =nonstrict;
```

Figure 18 Creation of partitioned table based on event_type

Step 1 : Creating partitioned and bucketed table and inserting data into the table:

For a lesser querying time this table was created with partition done based on 'event_type' and clustered/ bucketed based on price.

Command:

```
CREATE TABLE IF NOT EXISTS ecommerce_bucketed (event_time timestamp, product_id string, category_id string, category_code string, brand string, price float, user_id bigint, user_session string) PARTITIONED BY (event_type string) CLUSTERED BY (price) INTO 30 BUCKETS ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde';
```

```
INSERT INTO TABLE ecommerce_bucketed PARTITION(event_type) SELECT event_time, product_id, category_id, category_code, brand, price, user_id, user_session, event_type FROM ecommerce ;
```

```

hive> CREATE TABLE IF NOT EXISTS ecommerce_bucketed (event_time timestamp, product_id string, category_id string, category_code string, brand string, price float, user_id bigint, user_session string) PARTITIONED BY (event_type string) CLUSTERED BY (price) INTO 30 BUCKETS ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde';
OK
Time taken: 0.121 seconds
hive> INSERT INTO TABLE ecommerce_bucketed PARTITION(event_type) SELECT event_time, product_id, category_id, category_code, brand, price, user_id, user_session, event_type FROM ecommerce ;
Query ID = hadoop_20220430151100_ee4ec6a5-5cc3-4367-87e9-e1ea01c8c39c
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1651325890766_0006)

-----  

      VERTICES    MODE     STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

-----  

Map 1 ..... container SUCCEEDED   2      2      0      0      0      0      0  

Reducer 2 ..... container SUCCEEDED   5      5      0      0      0      0      0  

-----  

VERTICES: 02/02 [=====] 100% ELAPSED TIME: 172.33 s  

-----  

Loading data to table clickstream_data.ecommerce_bucketed partition (event_type=null)  

Loaded : 4/4 partitions.  

Time taken to load dynamic partitions: 0.429 seconds  

Time taken for adding to write entity : 0.002 seconds  

OK
event_time      product_id      category_id      category_code      brand      price      user_id user_session      event_type
Time taken: 184.578 seconds
hive>

```

Figure 19 Creation and insertion of data into bucketed table

Command:

```
SELECT * FROM ecommerce_bucketed LIMIT 5 ;
```

```

hive> SELECT * FROM ecommerce_bucketed LIMIT 5 ;
OK
ecommerce_bucketed.event_time  ecommerce_bucketed.product_id  ecommerce_bucketed.category_id  ecommerce_bucketed.category_code  ecommerce_bucketed.brand
ecommerce_bucketed.price  ecommerce_bucketed.user_id  ecommerce_bucketed.user_session ecommerce_bucketed.event_type
2019-10-08 07:00:58 UTC 5864603 1487580013086638258  italwax 7.94  557958553  5f4f15b4-ff56-4bfa-8a59-d0e6be5bd57c  cart
2019-10-10 08:13:48 UTC 5695629 1487580013841613016  estel  6.59  558716501  dc31944b-dd09-4f00-87a8-5244300fb7e  cart
2019-10-09 08:32:15 UTC 5858175 1487580011585077370  3.89  558362705  3f3a3061-5e9f-4a29-b29e-3e4d0e8000be  cart
2019-10-10 18:44:23 UTC 5869143 1783999063314661546  cosmoprofi 7.94  549195612  93620f57-af00-42b9-b3a8-40b8fdb286d1  cart
2019-10-11 05:49:56 UTC 5690991 1998040850688377703  cnd  8.57  558528420  66ad7c3b-025f-4cf1-b4b1-9fa9be0c6af0  cart
Time taken: 0.216 seconds, Fetched: 5 row(s)
hive>

```

Figure 20 Checking the data in the bucketed table

Step 2: Comparing query performance between the tables and utilizing the optimized table for analysis

Analysis: Your company wants to reward the top 10 users of its website with a Golden Customer plan. Write a query to generate a list of top 10 users who spend the most.

3.1 Using the unoptimized table (flat table)

Command:

```

WITH Customer_Rank AS( SELECT user_id AS Customer, ROUND(SUM(price),2) AS Expenditure, RANK() OVER(ORDER BY ROUND(SUM(price),2) DESC) AS Rank FROM ecommerce WHERE event_type = 'purchase' GROUP BY user_id) SELECT Customer, Expenditure, Rank FROM Customer_Rank WHERE Rank <=10;

```

```

hive> WITH Customer_Rank AS( SELECT user_id AS Customer, ROUND(SUM(price),2) AS Expenditure, RANK() OVER(ORDER BY ROUND(SUM(price),2) DESC) AS Rank FROM ecommerce WHERE event_type = 'purchase' GROUP BY user_id) SELECT Customer, Expenditure, Rank FROM Customer_Rank WHERE Rank <=10;
Query ID = hadoop_20220430173129_4fa2637e-4c9f-480f-b03c-fbee6e11f02e
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1651325890766_0011)

-----  

      VERTICES    MODE     STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

-----  

Map 1 ..... container  SUCCEEDED   2      2      0      0      0      0  

Reducer 2 ..... container  SUCCEEDED   3      3      0      0      0      0  

Reducer 3 ..... container  SUCCEEDED   2      2      0      0      0      0  

-----  

VERTICES: 03/03 [----->>] 100% ELAPSED TIME: 62.95 s  

-----  

OK  

customer expenditure rank  

557790271 2715.87 1  

150318419 1645.97 2  

562167663 1352.85 3  

531900924 1329.45 4  

557850743 1295.48 5  

522130011 1185.39 6  

561592095 1109.7 7  

431950134 1097.59 8  

566576008 1056.36 9  

521347209 1040.91 10  

Time taken: 75.029 seconds, Fetched: 10 row(s)

```



Figure 21 Analysis using the flat table

3.2 Using Bucketed table

Command:

```

WITH Customer_Rank AS( SELECT user_id AS Customer, ROUND(SUM(price),2) AS Expenditure, RANK() OVER(ORDER BY ROUND(SUM(price),2) DESC) AS Rank FROM ecommerce_bucketed WHERE event_type = 'purchase' GROUP BY user_id) SELECT Customer, Expenditure, Rank FROM Customer_Rank WHERE Rank <=10;

```

```

hive> WITH Customer_Rank AS( SELECT user_id AS Customer, ROUND(SUM(price),2) AS Expenditure, RANK() OVER(ORDER BY ROUND(SUM(price),2) DESC) AS Rank FROM ecommerce_bucketed WHERE event_type = 'purchase' GROUP BY user_id) SELECT Customer, Expenditure, Rank FROM Customer_Rank WHERE Rank <=10;
Query ID = hadoop_20220430173453_0bf100f7-97ed-4560-bb3c-8856a456ale4
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1651325890766_0011)

-----  

      VERTICES    MODE     STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

-----  

Map 1 ..... container  SUCCEEDED   3      3      0      0      0      0  

Reducer 2 ..... container  SUCCEEDED   1      1      0      0      0      0  

Reducer 3 ..... container  SUCCEEDED   1      1      0      0      0      0  

-----  

VERTICES: 03/03 [----->>] 100% ELAPSED TIME: 28.78 s  

-----  

OK  

customer expenditure rank  

557790271 2715.87 1  

150318419 1645.97 2  

562167663 1352.85 3  

531900924 1329.45 4  

557850743 1295.48 5  

522130011 1185.39 6  

561592095 1109.7 7  

431950134 1097.59 8  

566576008 1056.36 9  

521347209 1040.91 10  

Time taken: 29.776 seconds, Fetched: 10 row(s)

```



Figure 22 Analysis using bucketed table

| Table | Table Name | Time Taken (in Seconds) |
|----------------|--------------------|-------------------------|
| Flat table | ecommerce | 75.029 |
| Bucketed Table | ecommerce_bucketed | 29.776 |

Table 2 Time taken by the tables to fetch the columns

Inference:

We can see that the query time for bucketed table is much lower than the unoptimized table and hence the Bucketed table is the optimized table giving faster results.

Hence, the right choice of columns for partitioning and bucketing leads to a better query time. Here, we have chosen 'event_type' as partitioning parameter due to its low cardinality and better performance and 'price' as the clustering parameter.

Analysis using Hive queries

Question 1:

Find the total revenue generated due to purchases made in October.

Query:

```
SELECT ROUND(SUM(price),2) AS total_revenue_october FROM ecommerce_bucketed
WHERE event_type = 'purchase' GROUP BY month(event_time) HAVING month(event_time)
= 10;
```

```
hive> SELECT ROUND(SUM(price),2) AS total_revenue_october FROM ecommerce_bucketed WHERE event_type = 'purchase' GROUP BY month(event_time) HAVING month(event_time) = 10;
Query ID = hadoop_20220430175523_34b1a069-8628-443c-84dd-8a8450a5a0d1
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1651325890766_0012)

-----  

      VERTICES    MODE     STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

-----  

Map 1 ..... container    SUCCEEDED    3       3       0       0       0       0  

Reducer 2 ..... container    SUCCEEDED    1       1       0       0       0       0  

-----  

VERTICES: 02/02 [————>>] 100% ELAPSED TIME: 27.00 s  

-----  

OK  

total_revenue_october  

1211538.43  

Time taken: 38.036 seconds, Fetched: 1 row(s)
hive> ■
```

Figure 23 Query for finding revenue in October

Answer:

Total revenue generated due to purchases in October was 1211538.43.

Question 2:

Write a query to yield the total sum of purchases per month in a single output.

Query:

```
SELECT CASE WHEN (month(event_time) ==10) THEN 'Oct' ELSE 'Nov' END AS Month,
ROUND(SUM(price),2) AS total FROM ecommerce_bucketed WHERE event_type = 'purchase'
GROUP BY month(event_time);
```

```
hive> SELECT CASE WHEN (month(event_time) ==10) THEN 'Oct' ELSE 'Nov' END AS Month, ROUND(SUM(price),2) AS total FROM ecommerce_bucketed WHERE
event_type = 'purchase' GROUP BY month(event_time);
Query ID = hadoop_20220430175939_785962d2-3753-4268-98e4-3760af061702
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1651325890766_0012)

-----  

 VERTICES      MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

Map 1 ..... container SUCCEEDED    3       3       0       0       0       0  

Reducer 2 ..... container SUCCEEDED    1       1       0       0       0       0  

-----  

VERTICES: 02/02 [—————>>>] 100% ELAPSED TIME: 27.97 s  

-----  

OK  

month      total  

Oct        1211538.43  

Nov        1531016.9  

Time taken: 28.937 seconds, Fetched: 2 row(s)
hive>
```

Figure 24 Query to find sum purchases for October and November

Answer:

The sum of purchases for the month of October is 1211538.43 whereas for November is 1531016.9. Hence, the revenue due to 'purchases' in November is more than that of October.

Question 3:

Write a query to find the change in revenue generated due to purchases from October to November.

Query:

```
SELECT (Rev_Nov - Rev_Oct) AS difference FROM (SELECT ROUND(SUM(CASE WHEN
month(event_time)=10 THEN price ELSE 0 END),2) AS Rev_Oct, ROUND(SUM(CASE WHEN
month(event_time)=11 THEN price ELSE 0 END),2) AS Rev_Nov FROM ecommerce_bucketed
WHERE event_type='purchase') AS rev;
```

```

hive> SELECT (Rev_Nov - Rev_Oct) AS difference FROM (SELECT ROUND(SUM(CASE WHEN month(event_time)=10 THEN price ELSE 0 END),2) AS Rev_Oct, ROUND(SUM(CASE WHEN month(event_time)=11 THEN price ELSE 0 END),2) AS Rev_Nov FROM ecommerce_bucketed WHERE event_type='purchase') AS rev;
Query ID = hadoop_20220501091327_eaf82e01-c5f5-423a-aa6b-d61b3f0b370c
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1651395421072_0003)

-----  

      VERTICES    MODE     STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

-----  

Map 1 ..... container SUCCEEDED   3       3       0       0       0       0  

Reducer 2 ..... container SUCCEEDED   1       1       0       0       0       0  

-----  

VERTICES: 02/02 [—————>>] 100% ELAPSED TIME: 26.46 s  

-----  

OK  

difference  

319478.47  

Time taken: 27.922 seconds, Fetched: 1 row(s)
hive> ■

```

Figure 25 Query to find change in revenue from October to November

Answer:

The difference between sales between October and November is 319478.47.

Question 4:

Find distinct categories of products. Categories with null category code can be ignored.

Query:

```

SELECT DISTINCT SPLIT(category_code,'\\.')[0] AS Category_List FROM ecommerce_bucketed
WHERE category_code != '';

```

```

hive> SELECT DISTINCT SPLIT(category_code,'\\.')[0] AS Category_List FROM ecommerce_bucketed WHERE category_code != '';
Query ID = hadoop_20220504074820_162b859e-2502-4c66-af3d-8083c4540657
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1651647952134_0004)

-----  

      VERTICES    MODE     STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

-----  

Map 1 ..... container SUCCEEDED   6       6       0       0       0       0  

Reducer 2 ..... container SUCCEEDED   5       5       0       0       0       0  

-----  

VERTICES: 02/02 [—————>>] 100% ELAPSED TIME: 75.82 s  

-----  

OK  

category_list  

furniture  

appliances  

accessories  

apparel  

sport  

stationery  

Time taken: 76.763 seconds, Fetched: 6 row(s)
hive> ■

```

Figure 26 Query to find distinct categories of products

Answer:

category_list
furniture
appliances
accessories
apparel
sport
stationery

.....

Question 5:

Find the total number of products available under each category.

Query:

```
SELECT SPLIT(category_code,'\\.')[0] AS Category, COUNT(product_id) AS Product_count
FROM ecommerce_bucketed WHERE category_code !='' GROUP BY
SPLIT(category_code,'\\.')[0] ORDER BY Product_count DESC;
```

```
hive> SELECT SPLIT(category_code,'\\.')[0] AS Category, COUNT(product_id) AS Product_count FROM ecommerce_bucketed WHERE category_
code !='' GROUP BY SPLIT(category_code,'\\.')[0] ORDER BY Product_count DESC;
Query ID = hadoop_20220504074416_5a69c56d-b7d2-4f6b-894f-2c2aa1854398
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1651647952134_0004)

-----  

      VERTICES    MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

-----  

Map 1 ..... container SUCCEEDED   6      6      0      0      0      0  

Reducer 2 ..... container SUCCEEDED   5      5      0      0      0      0  

Reducer 3 ..... container SUCCEEDED   1      1      0      0      0      0  

-----  

VERTICES: 03/03 [————>] 100% ELAPSED TIME: 70.61 s  

-----  

OK
category      product_count
appliances    61736
stationery    26722
furniture     23604
apparel       18232
accessories   12929
sport         2
Time taken: 71.391 seconds, Fetched: 6 row(s)
hive> 
```

Figure 27 Query to find total number of products available under each category

Answer:

| category | product_count |
|-------------|---------------|
| appliances | 61736 |
| stationery | 26722 |
| furniture | 23604 |
| apparel | 18232 |
| accessories | 12929 |
| sport | 2 |

Question 6:

Which brand had the maximum sales in October and November combined?

Query:

```
SELECT brand, ROUND(SUM(price),2) AS total_sales FROM ecommerce_bucketed WHERE brand != '' AND event_type = 'purchase' GROUP BY brand ORDER BY total_sales DESC LIMIT 1;
```

```
hive> SELECT brand, ROUND(SUM(price),2) AS total_sales FROM ecommerce_bucketed WHERE brand != '' AND event_type = 'purchase' GROUP BY brand ORDER BY total_sales DESC LIMIT 1;
Query ID = hadoop_20220501100708_51d2f59d-0bf4-4793-a2d9-1022b5a81d95
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1651395421072_0005)

-----  
 VERTICES      MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  
-----  
 Map 1 ..... container SUCCEEDED    3        3        0        0        0        0  
 Reducer 2 ..... container SUCCEEDED    1        1        0        0        0        0  
 Reducer 3 ..... container SUCCEEDED    1        1        0        0        0        0  
-----  
 VERTICES: 03/03  [=====>>] 100% ELAPSED TIME: 25.43 s  
-----  
OK  
brand  total_sales  
runail 148297.94  
Time taken: 26.09 seconds, Fetched: 1 row(s)  
hive> █
```

Figure 28 Query to find brand with maximum sales in both the months

Answer:

```
brand  total_sales  
runail 148297.94
```

Question 7:

Which brands increased their sales from October to November?

Query:

```
SELECT oct.brand as Brand, ROUND((nov.Nov_sale - oct.Oct_sale),2) AS increase_in_sale
FROM (SELECT brand, SUM(price) as Oct_sale FROM ecommerce_bucketed WHERE event_type = 'purchase' AND brand != '' AND month(event_time) = 10 GROUP BY brand) oct
JOIN (SELECT brand, SUM(price) as Nov_sale FROM ecommerce_bucketed WHERE event_type = 'purchase' AND brand != '' AND month(event_time) = 11 GROUP BY brand) nov
ON oct.brand = nov.brand WHERE nov.Nov_sale > oct.Oct_sale ORDER BY increase_in_sale DESC;
```

```

hive> SELECT oct.brand as Brand, ROUND((nov.Nov_sale - oct.Oct_sale),2) AS increase_in_sale FROM (SELECT brand, SUM(price) as Oct_sale FROM ecommerce_bucketed WHERE event_type = 'purchase' AND brand != '' AND month(event_time) = 10 GROUP BY brand) oct JOIN (SELECT brand, SUM(price) as Nov_sale FROM ecommerce_bucketed WHERE event_type = 'purchase' AND brand != '' AND month(event_time) = 11 GROUP BY brand) nov ON oct.brand = nov.brand WHERE nov.Nov_sale > oct.Oct_sale ORDER BY increase_in_sale DESC;
Query ID = hadoop_20220501101345_06469e06-dbef-46b9-b681-c41c92a3d3ae
Total jobs = 1
Launching Job 1 out of 1
Tez session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1651395421072_0006)

-----  

  VERTICES   MODE    STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  

-----  

Map 1 ..... container  SUCCEEDED  3      3      0      0      0      0      0  

Map 5 ..... container  SUCCEEDED  3      3      0      0      0      0      0  

Reducer 2 ..... container  SUCCEEDED  1      1      0      0      0      0      0  

Reducer 3 ..... container  SUCCEEDED  1      1      0      0      0      0      0  

Reducer 4 ..... container  SUCCEEDED  1      1      0      0      0      0      0  

Reducer 6 ..... container  SUCCEEDED  1      1      0      0      0      0      0  

-----  

VERTICES: 06/06 [=====>>] 100% ELAPSED TIME: 35.82 s

```

Figure 29 Query to find the brands that increase sales

| | | | | | | |
|------------|------------------|-------------|-----------|--------------|-----------|--------|
| roubloff | 1422.41 | bioqua | 455.23 | elizavecca | 133.77 | |
| OK | levrana | 1420.54 | farmavita | 454.6 | nefertiti | 133.12 |
| brand | increase_in_sale | oniq | 1416.24 | sophin | 447.66 | |
| grattol | 36027.17 | irisk | 1354.08 | yu-r | 402.3 | |
| uno | 15737.72 | severina | 1344.6 | kiss | 395.78 | |
| lianail | 10501.4 | joico | 1309.58 | lador | 387.92 | |
| ingarden | 10404.82 | zeitun | 1300.97 | ellips | 360.19 | |
| strong | 9474.64 | beauty-free | 1228.69 | jas | 338.47 | |
| jessnail | 7057.39 | swarovski | 1155.23 | lowence | 324.91 | |
| cosmoprofi | 6214.18 | de.lux | 1115.81 | nitrile | 315.4 | |
| polarus | 5358.21 | metzger | 1083.71 | shary | 304.53 | |
| runail | 5219.38 | markell | 1065.68 | kims | 302.0 | |
| freedecor | 4250.02 | sanoto | 1052.54 | happyfons | 289.67 | |
| staleks | 3355.88 | nagaraku | 957.94 | kocostar | 284.08 | |
| bpw.style | 3265.29 | ecolab | 951.45 | insight | 278.26 | |
| lovely | 3234.68 | art-visage | 905.09 | candy | 264.42 | |
| marathon | 2992.35 | levissime | 857.81 | bluesky | 258.29 | |
| haruyama | 2962.22 | missha | 856.45 | beaugreen | 256.84 | |
| yoko | 2950.97 | solomeya | 786.1 | protokeratin | 255.54 | |
| italwax | 2859.13 | rosi | 764.52 | trind | 244.89 | |
| benovy | 2850.35 | refectocil | 759.4 | entity | 239.55 | |
| kaypro | 2387.36 | kaaral | 673.64 | skinlite | 238.51 | |
| estel | 2385.92 | kosmekka | 631.93 | provoc | 235.83 | |
| concept | 2348.26 | kinetics | 611.01 | fedula | 211.43 | |
| kapous | 2165.92 | browxenna | 585.36 | ecocraft | 200.79 | |
| f.o.x | 1953.05 | airnails | 572.62 | keen | 199.27 | |
| masura | 1792.39 | uskusi | 548.04 | mane | 193.47 | |
| milv | 1737.07 | coifin | 525.49 | freshbubble | 183.64 | |
| beautix | 1729.0 | s.care | 500.39 | chi | 179.67 | |
| artex | 1596.61 | limoni | 487.7 | cristalinas | 157.32 | |
| domix | 1537.12 | matrix | 483.49 | farmona | 150.97 | |
| shik | 1498.52 | gehwo | 468.61 | latinoil | 135.07 | |
| smart | 1444.88 | greymy | 460.28 | miskin | 135.03 | |

| | |
|-------------|-------|
| orly | 28.71 |
| estelare | 27.06 |
| profepil | 24.66 |
| blixz | 24.45 |
| godefroy | 23.9 |
| glysolid | 21.86 |
| veraclara | 21.1 |
| kamill | 18.48 |
| treaclemoon | 18.12 |
| supertan | 16.14 |
| deoproce | 12.33 |
| rasyan | 10.14 |
| fly | 10.03 |
| tertio | 9.64 |
| jaguar | 8.54 |
| soleo | 8.33 |
| neoleor | 8.29 |
| moyou | 4.57 |
| bodyton | 4.3 |
| skinity | 3.56 |
| grace | 1.69 |
| cosima | 0.7 |
| ovale | 0.56 |

Answer:

The brand with highest increase in sales from October to November is Grattol with 36027.17

.....

Question 8:

Your company wants to reward the top 10 users of its website with a Golden Customer plan. Write a query to generate a list of top 10 users who spend the most.

Query:

```
WITH Customer_Rank AS( SELECT user_id AS Customer, ROUND(SUM(price),2) AS Expenditure, RANK() OVER(ORDER BY ROUND(SUM(price),2) DESC) AS Rank FROM ecommerce_bucketed WHERE event_type = 'purchase' GROUP BY user_id) SELECT Customer, Expenditure, Rank FROM Customer_Rank WHERE Rank <=10;
```

```
hive> WITH Customer_Rank ASC SELECT user_id AS Customer, ROUND(SUM(price),2) AS Expenditure, RANK() OVER(ORDER BY ROUND(SUM(price),2) DESC) AS Rank FROM ecommerce_bucketed WHERE event_type = 'purchase' GROUP BY user_id) SELECT Customer, Expenditure, Rank FROM Customer_Rank WHERE Rank <=10;
Query ID = hadoop_20220430173453_0bf100f7-97ed-4560-bb3c-8856a456a1e4
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1651325890766_0011)

-----  

      VERTICES    MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

-----  

Map 1 ..... container    SUCCEEDED    3      3      0      0      0      0  

Reducer 2 ..... container    SUCCEEDED    1      1      0      0      0      0  

Reducer 3 ..... container    SUCCEEDED    1      1      0      0      0      0  

-----  

VERTICES: 03/03 [—————>>] 100% ELAPSED TIME: 28.78 s  

-----  

OK  

customer      expenditure      rank  

557790271    2715.87 1  

150318419    1645.97 2  

562167663    1352.85 3  

531900924    1329.45 4  

557850743    1295.48 5  

522130011    1185.39 6  

561592095    1109.7  7  

431950134    1097.59 8  

566576008    1056.36 9  

521347209    1040.91 10  

Time taken: 29.776 seconds, Fetched: 10 row(s)
```

Figure 30 Query to find top 10 users

Answer:

| customer | expenditure | rank |
|-----------|-------------|------|
| 557790271 | 2715.87 | 1 |
| 150318419 | 1645.97 | 2 |
| 562167663 | 1352.85 | 3 |
| 531900924 | 1329.45 | 4 |
| 557850743 | 1295.48 | 5 |
| 522130011 | 1185.39 | 6 |
| 561592095 | 1109.7 | 7 |
| 431950134 | 1097.59 | 8 |
| 566576008 | 1056.36 | 9 |
| 521347209 | 1040.91 | 10 |

521347209 1040.91 10

Dropping tables, database and terminating the cluster

Step 1: Dropping tables and database

Command:

```
SHOW TABLES ;
```

```
DROP TABLE ecommerce ;
```

```
DROP TABLE ecommerce_bucketed ;
```

```
SHOW DATABASES ;
```

```
DROP DATABASE clickstream_data ;
```

```
hive> SHOW TABLES ;
OK
ecommerce
ecommerce_bucketed
Time taken: 0.027 seconds, Fetched: 2 row(s)
hive> DROP TABLE ecommerce ;
OK
Time taken: 0.382 seconds
hive> DROP TABLE ecommerce_bucketed ;
OK
Time taken: 0.161 seconds
hive> SHOW TABLES ;
OK
Time taken: 0.023 seconds
hive> SHOW DATABASES ;
OK
clickstream_data
default
Time taken: 0.03 seconds, Fetched: 2 row(s)
hive> DROP DATABASE clickstream_data ;
OK
Time taken: 0.194 seconds
hive> SHOW DATABASES ;
OK
default
Time taken: 0.019 seconds, Fetched: 1 row(s)
hive> EXIT ;
```

Figure 31 Dropping the tables and database

We have successfully dropped all the tables created and the database and only left with the default database.

Step 2: Terminating the cluster

The screenshot shows the AWS EMR Cluster details page for a cluster named 'hive_case_study'. The cluster status is 'Waiting'. A modal dialog box titled 'Terminate cluster' is open, asking 'Are you sure you want to terminate this cluster? Any pending work or data residing on the cluster will be lost, such as data stored in HDFS. This action is irreversible.' The 'Terminate' button is highlighted in red.

Figure 32 Cluster termination

```

hive>
Broadcast message from root@ip-172-31-4-238
(unknown) at 10:43 ...

The system is going down for power off NOW!
Connection to ec2-34-201-32-127.compute-1.amazonaws.com closed by remote host.
Connection to ec2-34-201-32-127.compute-1.amazonaws.com closed.
(base) Sukeerthis-MacBook-Pro:Downloads sukeerthig$ 

```

Figure 33 Closing the connection

The screenshot shows the AWS EMR Cluster details page for the same cluster. The status is now 'Terminated by user request'. The 'Terminate' button is now greyed out. The 'Configuration details' section shows the cluster was terminated at 16:13 UTC on May 1, 2022. The 'Network and hardware' section shows the cluster was terminated in the us-east-1b availability zone, with a master instance of type m4.large and a core instance of type m4.large.

Figure 34 Cluster successfully terminated