

---

# Hive Case Study

By

- Amber Kaushal ([amber.kaushal10@gmail.com](mailto:amber.kaushal10@gmail.com))

---

## Problem Statement:-

With online sales gaining popularity, tech companies are exploring ways to improve their sales by analysing customer behaviour and gaining insights about product trends. Furthermore, the websites make it easier for customers to find the products they require without much scavenging. The clickstream data contains all the logs as to how you navigated through the website. From this, machine learning engineers or business analysts use this data to derive valuable insights.

## Objective:-

For this Case Study, We have to work with a public clickstream dataset of a cosmetics store. Using this dataset, our job is to extract valuable insights which generally data engineers come up within an e-retail company. And required to provide answers to the questions given below:

- Find the total revenue generated due to purchases made in October.
- Write a query to yield the total sum of purchases per month in a single output.
- Write a query to find the change in revenue generated due to purchases from October to November.
- Find distinct categories of products. Categories with null category code can be ignored.
- Find the total number of products available under each category.
- Which brand had the maximum sales in October and November combined?
- Which brands increased their sales from October to November?
- 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.

## Index

Content	Page No.
1. EMR Cluster Creation	(4)
a. Creating Key-Pair	(4)
b. Cluster Creation	(5)
c. Connect The Master Node Through PUTTY	(6)
2. Create New Directory	(7)
3. Load the Datasets into HDFS from S3	(8)
4. Launch Hive	(9)
5. Create The Database - clickstream	(10)
6. Creating The External Table - retaildata	(10)
7. Partitioning & Bucketing	(12)
a. Creating Partition Table 1 - part_1_retaildata	(12)
b. Creating Partition Table 2 - part_2_retaildata	(14)
c. Creating Partition Table 3 - part_3_retaildata	(15)
8. Answering The Questions	(17)
9. Cleaning up	(22)

# 1. EMR Cluster Creation

## a. Creating Key-Pair

First, we have to create Key-Pair with ppk format.

**Create 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**

Hive\_Case\_Study\_Key\_Pair

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

**Private key file format**

☐ .pem  
For use with OpenSSH

☒ .ppk  
For use with PuTTY

**Tags (Optional)**

No tags associated with the resource.

[Add tag](#)

You can add 50 more tags.

[Cancel](#) [Create key pair](#)

We have successfully created and downloaded the key-pair **"Hive\_Case\_Study\_Key\_Pair"**. Please refer to below screenshot.

**Key pairs (1)** Info

[Filter key pairs](#)

<input type="checkbox"/>	Name	Fingerprint	ID
<input type="checkbox"/>	Hive_Case_Study_Key_Pair	61:60:02:b5:f2:8e:5a:bb:f5:b3:f4:fd:80:...	key-0ac5fe14cf08f438d

## b. Cluster Creation

We have created the cluster “Case\_Study\_Hive” with **emr-5.29.0** release and used a **2-node** EMR cluster with both the master and core nodes as **M4.large**.

The screenshot shows the AWS Management Console for the EMR cluster 'Case\_Study\_Hive'. The cluster is in a 'Waiting' state. The 'Summary' tab is selected, showing details such as ID, creation date, and configuration. The 'Configuration details' section highlights the 'Release label: emr-5.29.0'. The 'Network and hardware' section shows the 'Subnet ID: subnet-10492971' and the 'Master: Running 1 m4.large' and 'Core: Running 1 m4.large' nodes. The 'Security and access' section shows the 'Key name: Hive\_Case\_Study\_Key\_Pair'.

Now set the SSH rule to inbound rule for the Master node, if not created yet.

The screenshot shows the 'Add rule' dialog in the AWS Management Console. The 'Rule type' is set to 'SSH'. The 'Protocol' is set to 'TCP'. The 'Port' is set to '22'. The 'Source' is set to 'Anywh...'. The 'Destination' is set to '0.0.0.0/0'. The 'Action' is set to 'Allow'. The 'Rule name' is set to '-'. The 'Add rule' button is highlighted.

## c. Connect The Master Node Through PUTTY

After the cluster reddy, we have to click on Master public DNS and copy the hostname **hadoop@ec2-18-209-179-95.compute-1.amazonaws.com**.

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](#).

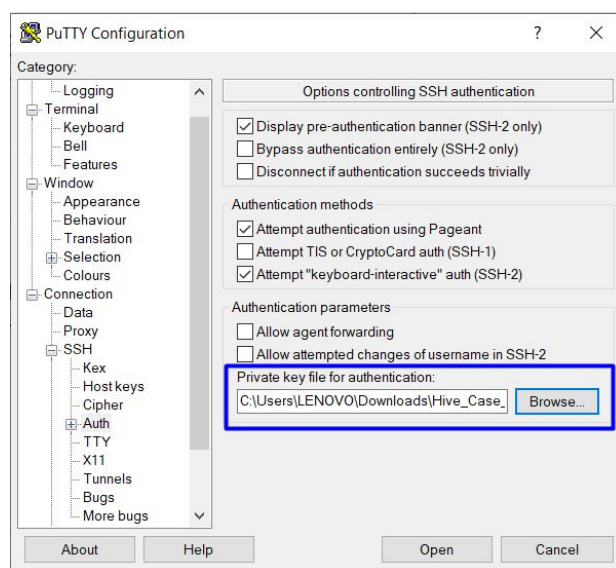
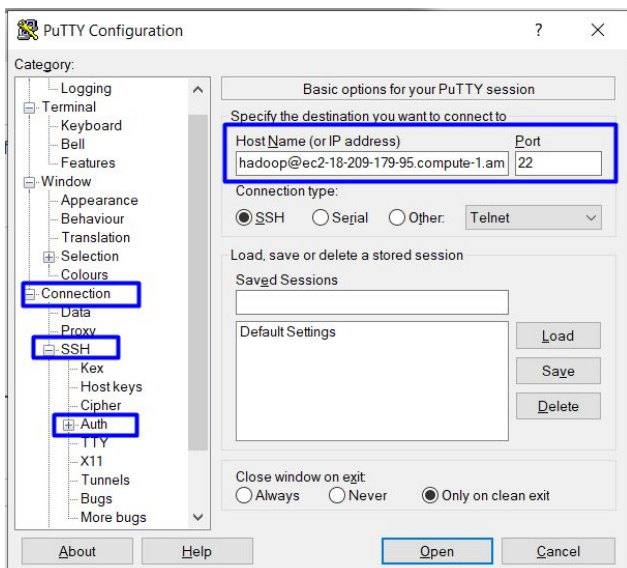
Windows

Mac / Linux

1. Download PuTTY.exe to your computer from: <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>
2. Start PuTTY.
3. In the Category list, click Session.
4. In the Host Name field, type **hadoop@ec2-18-209-179-95.compute-1.amazonaws.com**
5. In the Category list, expand Connection > SSH, and then click Auth.
6. For Private key file for authentication, click Browse and select the private key file (**Hive\_Case\_Study\_Key\_Pair.ppk**) used to launch the cluster.
7. Click Open.
8. Click Yes to dismiss the security alert.

Close

After the cluster reddy, we have to click on Master public DNS and copy the hostname **hadoop@ec2-18-209-179-95.compute-1.amazonaws.com**.



After the cluster reddy, we have to click on Master public DNS and copy the hostname **hadoop@ec2-18-209-179-95.compute-1.amazonaws.com**.

EMR CLI launched Successfully. Please Refer the Screenshot below:

```
hadoop@ip-172-31-52-97:~$
Using username "hadoop".
Authenticating with public key "Hive_Case_Study_Key_Pair"
Last login: Sun Aug  8 19:17:07 2021

 _ _ _ _ _
| | | | |
|_|_|_|_|_|

Amazon Linux AMI

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
68 package(s) needed for security, out of 107 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEE MMMMMMM MMMMMMM RRRRRRRRRRRRRR
E::::::::::::::::::E M::::::::M M::::::::M R::::::::::::R
EE::::::::EEEEEEEEE E M::::::::M M::::::::M R::RRRRR::R
E:::E EEEEE M::::::::M M::::::::M RR::R R:::R
E:::E M::::::::M M::::::::M R::R R:::R
E:::EEEEEEEEEE M:::M M:::M M:::M R::RRRRR::R
E::::::::::::::::::E M:::M M:::M M:::M R::::::::::::R
E:::EEEEEEEEEE M:::M M:::M M:::M R::RRRRR::R
E:::E M:::M M:::M M:::M R::R R:::R
E:::E EEEEE M:::M M:::M M:::M R::R R:::R
EE::::::::EEEEEE E M:::M M:::M M:::M R::R R:::R
E::::::::::::::::::E M:::M M:::M M:::M R::R R:::R
EEEEEEEEEEEEEEEE MMMMMMM MMMMMMM RRRRRRRRRRRRRR

[hadoop@ip-172-31-52-97 ~]$
```

## 2. Create New Directory

- We have to create a new directory "**hivecasestudy**" to load the data into the HDFS from s3.
- **Command:** `hadoop fs -mkdir /user/hive/hivecasestudy`
- And verifying whether new directory created or not.
- **Command:** `hadoop fs -ls /user/hive/`

```
[hadoop@ip-172-31-52-97 ~]$ hadoop fs -mkdir /user/hive/hivecasestudy
[hadoop@ip-172-31-52-97 ~]$ hadoop fs -ls /user/hive/
Found 2 items
drwxr-xr-x - hadoop hadoop 0 2021-08-08 20:32 /user/hive/hivecasestudy
drwxrwxrwt - hdfs hadoop 0 2021-08-08 18:52 /user/hive/warehouse
[hadoop@ip-172-31-52-97 ~]$
```

The directory created successfully

To load the datasets into HDFS from S3, for that purpose we are going to use distributed copy command.



### 3. Load the Datasets into HDFS from S3

To load the datasets into HDFS from S3, for that purpose we are going to use distributed copy command.

**Command:**

```
hadoop distcp 's3://e-commerce-events-ml/*' '/user/hive/hivecasestudy/'
```

```
[hadoop@ip-172-31-52-97 ~]$ hadoop distcp 's3://e-commerce-events-ml/*' '/user/hive/hivecasestudy/'
21/08/08 21:15:29 INFO tools.DistCp: Input Counters: DistCpOptions(atomicCommit=false, syncFolder=false, deleteMissing=false, ignoreFailures=false, overwrite=false, skipCRC=false, blocking=true, numListStatusThreads=0, maxMaps=20, mapBandwidth=100, sslConfigurationFile='null', copyStrategy='uniformsize', preserveStatus=[], preserveRawXattrs=false, atomicWorkPath=null, logPath=null, sourceFileListing=null, sourcePaths=[s3://e-commerce-events-ml/*], targetPath=/user/hive/hivecasestudy, targetPathExists=true, filtersFile='null')
21/08/08 21:15:29 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-52-97.ec2.internal/172.31.52.97:8032
21/08/08 21:15:34 INFO tools.SimpleCopyListing: Paths (files+dirs) cnt = 2; dirCnt = 0
21/08/08 21:15:34 INFO tools.SimpleCopyListing: Build file listing completed.
21/08/08 21:15:34 INFO Configuration.deprecation: io.sort.mb is deprecated. Instead, use mapreduce.task.io.sort.mb
21/08/08 21:15:34 INFO Configuration.deprecation: io.sort.factor is deprecated. Instead, use mapreduce.task.io.sort.factor
21/08/08 21:15:34 INFO tools.DistCp: Number of paths in the copy list: 2
21/08/08 21:15:34 INFO tools.DistCp: Number of paths in the copy list: 2
21/08/08 21:15:34 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-52-97.ec2.internal/172.31.52.97:8032
21/08/08 21:15:34 INFO mapreduce.JobSubmitter: number of splits:2
21/08/08 21:15:35 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1628448789720_0001
21/08/08 21:15:35 INFO impl.YarnClientImpl: Submitted application application_1628448789720_0001
21/08/08 21:15:36 INFO mapreduce.Job: The url to track the job: http://ip-172-31-52-97.ec2.internal:20888/proxy/application_1628448789720_0001/
21/08/08 21:15:36 INFO tools.DistCp: DistCp job-id: job_1628448789720_0001
21/08/08 21:15:36 INFO mapreduce.Job: Running job: job_1628448789720_0001
21/08/08 21:15:45 INFO mapreduce.Job: Job job_1628448789720_0001 running in uber mode : false
21/08/08 21:15:45 INFO mapreduce.Job: map 0% reduce 0%
21/08/08 21:16:05 INFO mapreduce.Job: map 100% reduce 0%
21/08/08 21:16:20 INFO mapreduce.Job: Job job_1628448789720_0001 completed successfully
21/08/08 21:16:20 INFO mapreduce.Job: Counters: 36
File System Counters
  FILE: Number of bytes read=0
  FILE: Number of bytes written=344922
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=898
  HDFS: Number of bytes written=1028381690
  HDFS: Number of read operations=26
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=8
  S3: Number of bytes read=1028381690
  S3: Number of bytes written=0
  S3: Number of read operations=0
  S3: Number of large read operations=0
  S3: Number of write operations=0
Job Counters
  Launched map tasks=2
  Other local map tasks=2
  Total time spent by all maps in occupied slots (ms)=2033728
  Total time spent by all reduces in occupied slots (ms)=0
  Total time spent by all map tasks (ms)=63554
  Total vcore-milliseconds taken by all map tasks=63554
  Total megabyte-milliseconds taken by all map tasks=65079296
```

#### Map-Reduce Framework

```
Map input records=2
Map output records=0
Input split bytes=272
Spilled Records=0
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=1285
CPU time spent (ms)=42640
Physical memory (bytes) snapshot=1144143872
Virtual memory (bytes) snapshot=6595477504
Total committed heap usage (bytes)=956301312
```

#### File Input Format Counters

```
Bytes Read=626
```

#### File Output Format Counters

```
Bytes Written=0
```

#### DistCp Counters

```
Bytes Copied=1028381690
```

```
Bytes Expected=1028381690
```

```
Files Copied=2
```

```
[hadoop@ip-172-31-52-97 ~]$
```



To Check the data sets in directory

**Command:** `hadoop fs -ls /user/hive/hivecasestudy/`

```
[hadoop@ip-172-31-52-97 ~]$ hadoop fs -ls /user/hive/hivecasestudy/
Found 2 items
-rw-r--r-- 1 hadoop hadoop 545839412 2021-08-08 21:16 /user/hive/hivecasestudy/2019-Nov.csv
-rw-r--r-- 1 hadoop hadoop 482542278 2021-08-08 21:16 /user/hive/hivecasestudy/2019-Oct.csv
[hadoop@ip-172-31-52-97 ~]$
```

Files copied successfully into HDFS from S3 buckets

Showing the data in HDFS

**Command:** `hadoop fs -cat /user/hive/hivecasestudy/2019-Nov.csv | head`  
`hadoop fs -cat /user/hive/hivecasestudy/2019-Oct.csv | head`

```
[hadoop@ip-172-31-52-97 ~]$ hadoop fs -cat /user/hive/hivecasestudy/2019-Nov.csv | head
event time,event_type,product id,category id,category code,brand,price,user_id,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,22.22,556138645,57ed222e-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
2019-11-01 00:00:24 UTC,remove_from_cart,5826182,1487580007483048900,,3.33,553329724,2067216c-31b5-455d-a1cc-af0575a34ffb
2019-11-01 00:00:25 UTC,view,5856189,1487580009026551821,,runail,15.71,562076640,09fafd6c-6c99-46b1-834f-33527f4de241
2019-11-01 00:00:32 UTC,view,5837835,1933472286753424063,,3.49,514649199,432a4e95-375c-4b40-bd36-0fc039e77580
2019-11-01 00:00:34 UTC,remove_from_cart,5870838,1487580007675986893,,milv,0.79,429913900,2f0bfff3c-252f-4fe6-afcd-5d8a6a92839a
cat: Unable to write to output stream.
[hadoop@ip-172-31-52-97 ~]$ hadoop fs -cat /user/hive/hivecasestudy/2019-Oct.csv | head
event time,event_type,product id,category id,category code,brand,price,user_id,user_session
2019-10-01 00:00:00 UTC,cart,5773203,1487580005134238553,,runail,2.62,463240011,26dd6e6e-4dac-4778-8d2c-92e149dab885
2019-10-01 00:00:03 UTC,cart,5773353,1487580005134238553,,runail,2.62,463240011,26dd6e6e-4dac-4778-8d2c-92e149dab885
2019-10-01 00:00:07 UTC,cart,5881589,2151191071051219817,,lovely,13.48,429681830,49e8d843-adf3-428b-a2c3-fe8bc6a307c9
2019-10-01 00:00:07 UTC,cart,5723490,1487580005134238553,,runail,2.62,463240011,26dd6e6e-4dac-4778-8d2c-92e149dab885
2019-10-01 00:00:15 UTC,cart,5881449,1487580013522845895,,lovely,0.56,429681830,49e8d843-adf3-428b-a2c3-fe8bc6a307c9
2019-10-01 00:00:16 UTC,cart,5857269,1487580005134238553,,runail,2.62,430174032,73dea1e7-664e-43f4-8b30-d32b9d5af04f
2019-10-01 00:00:19 UTC,cart,5739055,1487580008246412266,,kapous,4.75,377667011,81326ac6-daa4-4f0a-b488-fd0956a78733
2019-10-01 00:00:24 UTC,cart,5825598,1487580009445982239,,0.56,467916806,2f5b5546-b8cb-9ee7-7ecd-84276f8ef486
2019-10-01 00:00:25 UTC,cart,5698989,1487580006317032337,,1.27,385985999,d30965e8-1101-44ab-b45d-cc1bb9fae694
cat: Unable to write to output stream.
[hadoop@ip-172-31-52-97 ~]$
```

## 4. Launch Hive

We have added the data into the directory. Now we are going to launch hive

**Command:** `hive`

```
[hadoop@ip-172-31-51-218 ~]$ hive
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive>
```

Let's view the databases

**Command:** `show databases ;`

```
hive> show databases ;
OK
default
Time taken: 0.655 seconds, Fetched: 1 row(s)
hive>
```

We can see that only the default database is present. We have to create the database first.

## 5. Create The Database - clickstream

**Command:** CREATE DATABASE IF NOT EXISTS clickstream ;  
SHOW DATABASES ;  
DESCRIBE DATABASE clickstream ;  
USE clickstream ;

```
hive> CREATE DATABASE IF NOT EXISTS clickstream ;
OK
Time taken: 0.762 seconds
```

```
hive> SHOW DATABASES ;
OK
clickstream
default
Time taken: 0.687 seconds, Fetched: 2 row(s)
hive> DESCRIBE DATABASE clickstream ;
OK
clickstream      hdfs://ip-172-31-51-218.ec2.internal:8020/user/hive/warehouse/clickstream.db  hadoop  USER
Time taken: 0.073 seconds, Fetched: 1 row(s)
hive> USE clickstream ;
OK
Time taken: 0.047 seconds
hive>
```

## 6. Creating The External Table - retaildata

**Command:** CREATE EXTERNAL TABLE IF NOT EXISTS retaildata (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 '/user/hive/hivecasestudy' TBLPROPERTIES ("skip.header.line.count"="1");

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS retaildata (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 '/user/hive/hivecasestudy' TBLPROPERTIES ("skip.header.line.count"="1");
OK
Time taken: 0.801 seconds
hive>
```

Describe the table

**Command:** DESCRIBE retaildata ;

```
hive> DESCRIBE retaildata ;
OK
col_name      data_type      comment
event_time    string         from deserializer
event_type    string         from deserializer
product_id    string         from deserializer
category_id   string         from deserializer
category_code string         from deserializer
brand         string         from deserializer
price         string         from deserializer
user_id       string         from deserializer
user_session  string         from deserializer
Time taken: 0.065 seconds, Fetched: 9 row(s)
hive>
```

To display the header of the table

**Command:** SET hive.cli.print.header = true;

Viewing the first five rows of the table

**Command:** SELECT \* FROM retaildata LIMIT 5 ;

```
hive> SELECT * FROM retaildata LIMIT 5 ;
OK
retaildata.event_time  retaildata.event_type  retaildata.product_id  retailda
ta.category_id  retaildata.category_code  retaildata.brand  retailda
ta.price  retaildata.user_id  retaildata.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-alcc-af0575a34ffb
2019-11-01 00:00:10 UTC view 5837166 1783999064103190764 pnb 2
2.22 556138645 57ed222e-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-alcc-af0575a34ffb
Time taken: 2.178 seconds, Fetched: 5 row(s)
hive>
```

Our base table is ready (retaildata), now we can run a query, we are taking Question one for testing purposes i.e. Find the total revenue generated due to purchases made in October.

**Command:** SELECT SUM(price) AS total\_revenue FROM retaildata WHERE MONTH(event\_time)='10' AND event\_type='purchase';

```
hive> SELECT SUM(price) AS total_revenue FROM retaildata WHERE MONTH(event_time)='10' AND event_type='purchase';
Query ID = hadoop_20210809212550_1957088b-ac77-44b8-94a2-38a341cc5bc5
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1628530574915_0004)

-----
VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container  SUCCEEDED    2         2         0         0         0         0
Reducer 2 ..... container  SUCCEEDED    1         1         0         0         0         0
-----
VERTICES: 02/02 [=====>>>] 100%  ELAPSED TIME: 64.83 s
-----
OK
total_revenue
1211538.4299997438
Time taken: 65.517 seconds, Fetched: 1 row(s)
hive>
```

Time Taken to execute the above query is 65.51 sec.



## 7. Partitioning & Bucketing

Time taken to compute that query on base table is too high it's time to optimize the table by partitioning and bucketing, we will try different combinations of partitioning & bucketing and execute the same query until we get an optimized table.

To enable the dynamic partitioning and bucketing we are going to use below commands

**Command:** SET hive.exec.dynamic.partition.mode = nonstrict;  
SET hive.exec.dynamic.partition = true;  
SET hive.enforce.bucketing = true ;

```
hive> SET hive.exec.dynamic.partition.mode = nonstrict;  
hive> SET hive.exec.dynamic.partition = true;  
hive> SET hive.enforce.bucketing = true ;  
hive>
```

### a. Creating Partition Table 1 - part\_1\_retaildata

We are going to partition on "event\_type" and clustered by "price".

**Command:** CREATE TABLE IF NOT EXISTS part\_1\_retaildata (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 10 BUCKETS ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE ;

```
hive> CREATE TABLE IF NOT EXISTS part_1_retaildata (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 10 BUCKETS ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE ;  
OK  
Time taken: 0.069 seconds
```

Describe the newly created table

**Command:** DESCRIBE part\_1\_retaildata ;

```
hive> DESCRIBE part_1_retaildata ;  
OK  
col_name      data_type      comment  
event_time    string          from deserializer  
product_id    string          from deserializer  
category_id    string          from deserializer  
category_code string          from deserializer  
brand         string          from deserializer  
price         string          from deserializer  
user_id       string          from deserializer  
user_session  string          from deserializer  
event_type    string  
  
# Partition Information  
# col_name      data_type      comment  
event_type      string  
Time taken: 0.13 seconds, Fetched: 14 row(s)  
hive>
```

Now Inserting the data into the new optimized table (part\_1\_retaildata) from base table (retaildata)

**Command:** INSERT INTO TABLE part\_1\_retaildata PARTITION (event\_type) SELECT event\_time, product\_id, category\_id, category\_code, brand, price, user\_id, user\_session, event\_type FROM retaildata ;

```
hive> INSERT INTO TABLE part_1_retaildata PARTITION (event_type) SELEC
T event_time, product_id, category_id, category_code, brand, price, us
er_id, user_session, event_type FROM retaildata ;
Query ID = hadoop_20210810212944_c9c7f369-328a-49d6-b690-e3325cf3daef
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_162
8624052120_0012)

Map 1: 0/2      Reducer 2: 0/5
Map 1: 0/2      Reducer 2: 0/5
Map 1: 0(+1)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
```

```
Loading data to table clickstream.part_1_retaildata partition (event_t
ype=null)

Time taken to load dynamic partitions: 0.344 seconds
Time taken for adding to write entity : 0.004 seconds
OK
event_time      product_id      category_id      category_code      brandp
rice      user_id user_session      event_type
Time taken: 180.394 seconds
hive>
```

Now running the same query on the optimized table and note the time.

**Command:** SELECT SUM(price) AS total\_revenue FROM part\_1\_retaildata WHERE MONTH(event\_time)='10' AND event\_type='purchase';

```
hive> SELECT SUM(price) AS total_revenue FROM part_1_retaildata WHERE
MONTH(event_time)='10' AND event_type='purchase';
Query ID = hadoop_20210810213731_7a7aaf6b-6fce-4cda-a883-3461afde6404
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_162
8624052120_0012)

Map 1: 0/3      Reducer 2: 0/1
Map 1: 0/3      Reducer 2: 0/1
Map 1: 0/3      Reducer 2: 0/1
Map 1: 0(+1)/3  Reducer 2: 0/1
Map 1: 0(+2)/3  Reducer 2: 0/1
Map 1: 0(+3)/3  Reducer 2: 0/1
Map 1: 0(+3)/3  Reducer 2: 0/1
Map 1: 0(+3)/3  Reducer 2: 0/1
Map 1: 0(+3)/3  Reducer 2: 0/1
Map 1: 0(+3)/3  Reducer 2: 0/1
Map 1: 0(+3)/3  Reducer 2: 0/1
Map 1: 1(+2)/3  Reducer 2: 0(+1)/1
Map 1: 1(+2)/3  Reducer 2: 0(+1)/1
Map 1: 2(+1)/3  Reducer 2: 0(+1)/1
Map 1: 3/3      Reducer 2: 0(+1)/1
Map 1: 3/3      Reducer 2: 1/1
OK
total_revenue
1211538.430000279
Time taken: 25.211 seconds, Fetched: 1 row(s)
hive>
```

Time Taken to execute the above query is 25.211 sec.

## b. Creating Partition Table 2 - part\_2\_retaildata

We are going to partition on "month" and clustered by "brand".

**Command:** CREATE TABLE IF NOT EXISTS part\_2\_retaildata (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) PARTITIONED BY (month int) CLUSTERED BY (brand) INTO 10 BUCKETS ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE ;

```
hive> CREATE TABLE IF NOT EXISTS part_2_retaildata (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) PARTITIONED BY (month int) CLUSTERED BY (brand) INTO 10 BUCKETS ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE ;
OK
Time taken: 0.104 seconds
hive>
```

Now Inserting the data into the new optimized table (part\_2\_retaildata) from base table (retaildata)

**Command:** INSERT INTO TABLE part\_2\_retaildata PARTITION (month) SELECT event\_time, event\_type, product\_id, category\_id, category\_code, brand, price, user\_id, user\_session, MONTH(CAST(REPLACE(event\_time,'UTC','') AS timestamp)) FROM retaildata ;

```
hive> INSERT INTO TABLE part_2_retaildata PARTITION (month) SELECT event_time, event_type, product_id, category_id, category_code, brand, price, user_id, user_session, MONTH(CAST(REPLACE(event_time,'UTC','') AS timestamp)) FROM retaildata ;
```

```
Query ID = hadoop_20210811174356_36cc4fc7-21f9-45fb-a6be-33ca36ac89e2
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1628701152532_0003)
```

```
Map 1: 0/2      Reducer 2: 0/5
Map 1: 0/2      Reducer 2: 0/5
Map 1: 0(+1)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
Map 1: 0(+2)/2  Reducer 2: 0/5
```

```
Map 1: 2/2      Reducer 2: 3(+2)/5
Map 1: 2/2      Reducer 2: 4(+1)/5
Map 1: 2/2      Reducer 2: 4(+1)/5
Map 1: 2/2      Reducer 2: 5/5
```

```
Loading data to table clickstream.part_2_retaildata partition (month=null)
```

```
Time taken to load dynamic partitions: 0.152 seconds
Time taken for adding to write entity : 0.0 seconds
```

```
OK
event_time      event_type      product_id      category_id      category_code    b
rand_price      user_id user_session  _c9
Time taken: 177.59 seconds
hive>
```



Now running the same query on the optimized table 2 and note the time.

**Command:** SELECT SUM(price) AS total\_revenue FROM part\_2\_retaildata WHERE MONTH(event\_time)='10' AND event\_type='purchase';

```
hive> SELECT SUM(price) AS total_revenue FROM part_2_retaildata WHERE MONTH(event_time)='10' AND event_type='purchase';
Query ID = hadoop_20210811180052_97aef317-c96a-487d-94b0-d54950fd9d8f
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1628701152532_0004)

-----
VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container  SUCCEEDED   7         7         0         0         0         0
Reducer 2 ..... container  SUCCEEDED   1         1         0         0         0         0
-----
VERTICES: 02/02  [=====>>] 100%  ELAPSED TIME: 82.44 s
-----
OK
total_revenue
1211538.4300000907
Time taken: 83.094 seconds, Fetched: 1 row(s)
```

Time Taken to execute the above query is 83.094 sec.

### c. Creating Partition Table 3 - part\_3\_retaildata

We are going to partition on "event\_type" and clustered by "user\_id".

**Command:** CREATE TABLE IF NOT EXISTS part\_3\_retaildata (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 (user\_id) INTO 10 BUCKETS ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE ;

```
hive> CREATE TABLE IF NOT EXISTS part_3_retaildata (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 (user_id) INTO 10 BUCKETS ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE ;
OK
Time taken: 0.057 seconds
hive>
```

Now Inserting the data into the new optimized table (part\_3\_retaildata) from base table (retaildata)

**Command:** INSERT INTO TABLE part\_3\_retaildata PARTITION (event\_type) SELECT event\_time, product\_id, category\_id, category\_code, brand, price, user\_id, user\_session, event\_type FROM retaildata ;

```
hive> INSERT INTO TABLE part_3_retaildata PARTITION (event_type) SELECT event_time, product_id, category_id, category_code, brand, price, user_id, user_session, event_type FROM retaildata ;
Query ID = hadoop_20210811181338_da75cf87-5891-4cfc-be3a-4dee3a88ba86
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_1628701152532_0005)

-----
VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container    SUCCEEDED    2         2         0         0         0         0
Reducer 2 ..... container    SUCCEEDED    5         5         0         0         0         0
-----
VERTICES: 02/02 [=====>>>] 100% ELAPSED TIME: 165.03 s
-----
Loading data to table clickstream.part_3_retaildata partition (event_type=null)
Loaded : 4/4 partitions.
Time taken to load dynamic partitions: 0.402 seconds
Time taken for adding to write entity : 0.001 seconds
OK
event_time  product_id  category_id  category_code  brand  price  user_id  user_session  event_type
Time taken: 174.822 seconds
hive>
```

Now running the same query on the optimized table 3 and note the time.  
**Command:** SELECT SUM(price) AS total\_revenue FROM part\_3\_retaildata WHERE MONTH(event\_time)='10' AND event\_type='purchase';

```
hive> SELECT SUM(price) AS total_revenue FROM part_3_retaildata WHERE MONTH(event_time)='10' AND event_type='purchase';
Query ID = hadoop_20210811182051_10cfe7d8-58ec-4290-b899-558e770ed636
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1628701152532_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
-----
VERTICES: 02/02 [=====>>>] 100% ELAPSED TIME: 22.87 s
-----
OK
total_revenue
1211538.4299998833
Time taken: 23.666 seconds, Fetched: 1 row(s)
hive>
```

Time Taken to execute the above query is 23.666 sec.

Optimized Table	Excution Time
part_1_retaildata	25.211 sec
part_2_retaildata	83.094 sec
part_3_retaildata	23.666 sec

From the above table, we can see that the execution time (23.666 sec) of the optimized table "**part\_3\_retaildata**" is less as compared to others.  
"part\_3\_retaildata" table is Partitioning on "**event\_type**" and clustering by "**user\_id**" so we are going to use that table for all the following analysis.

## 8. Answering The Questions

**Q1.** Find the total revenue generated due to purchases made in October.

**Query:** SELECT SUM(price) AS total\_revenue FROM part\_3\_retaildata WHERE MONTH(event\_time)='10' AND event\_type='purchase';

```
hive> SELECT SUM(price) AS total_revenue FROM part_3_retaildata WHERE MONTH(event_time)='10' AND event_type='purchase';
Query ID = hadoop_20210811182051_10cfe7d8-58ec-4290-b899-558e770ed636
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1628701152532_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

```
VERTICES: 02/02 [=====>>] 100% ELAPSED TIME: 22.87 s
OK
total_revenue
1211538.4299998833
Time taken: 23.666 seconds, Fetched: 1 row(s)
```

### Insights:

- Time Taken to execute the above query is 23.66 sec.
- The total revenue generated due to purchases made in October is 1,211,538.43 /-

**Q2.** Write a query to yield the total sum of purchases per month in a single output.

**Query:** SELECT MONTH(event\_time) AS month, COUNT(event\_type) AS sum\_purchases FROM part\_3\_retaildata WHERE event\_type = 'purchase' GROUP BY MONTH(event\_time);

```
hive> SELECT MONTH(event_time) AS month, COUNT(event_type) AS sum_purchases FROM part_3_retaildata WHERE event_type = 'purchase' GROUP BY MONTH(event_time);
Query ID = hadoop_20210811193120_b62b5ccc-7fa9-4fe5-88aa-3e57c4178cad
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1628701152532_0007)
```

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: 23.11 s
OK
month    sum_purchases
10       245624
11       322417
Time taken: 23.826 seconds, Fetched: 2 row(s)
```

### Insights:

- Time Taken to execute the above query is 23.826 sec.
- The Sum of purchases in November month is 322417 which is higher than October month purchases 245624. We can conclude that the number of purchases increased in November month.

**Q3.** Write a query to find the change in revenue generated due to purchases from October to November.

**Query:** SELECT (SUM(CASE WHEN MONTH(event\_time)=11 THEN price ELSE 0 END) - SUM(CASE WHEN MONTH(event\_time)=10 THEN PRICE ELSE 0 END)) AS change\_in\_rev FROM part\_3\_retaildata WHERE event\_type = 'purchase' AND MONTH(event\_time) IN ('10','11');

```
hive> SELECT (SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END) - SUM(CASE WHEN MONTH(event_time)=10 THEN PRICE ELSE 0 END)) AS change_in_rev FROM part_3_retaildata WHERE event_type
= 'purchase' AND MONTH(event_time) in ('10','11');
Query ID = hadoop_20210811200151_94bc7f70-3a6b-418a-aea6-c9313631bb32
Total jobs = 1
Launching Job 1 out of 1
Ter session was closed. Reopening...
Session re-established.
Status: Running (Executing on YARN cluster with App id application_1628701152532_0008)
```

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: 26.33 s
OK
change in rev
319478.4699999499
Time taken: 35.113 seconds, Fetched: 1 row(s)
```

### Insights:

- Time Taken to execute the above query is 35.113 sec.
- The change in revenue generated from October to November is 319478.47/-

**Q4.** Find distinct categories of products. Categories with null category code can be ignored.

**Query:** SELECT DISTINCT SPLIT(category\_code,'\\\.')[0] AS category FROM part\_3\_retaildata WHERE category\_code != '';

```
hive> SELECT DISTINCT SPLIT(category_code,'\\\.')[0] AS category FROM part_3_retaildata WHERE category_code != '';
Query ID = hadoop_20210811204126_aedcbdef-2b4f-4fbc-877b-eedcfa455789
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1628701152532_0011)
```

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: 63.19 s
OK
category
furniture
appliances
accessories
apparel
sport
stationery
Time taken: 63.79 seconds, Fetched: 6 row(s)
```

### Insights:

- Time Taken to execute the above query is 63.79 sec.
- We have found a total 6 number of distinct product categories: furniture, appliances, accessories, apparel, sport, stationary



**Q5.** Find the total number of products available under each category.

**Query:** SELECT SPLIT(category\_code,'\\.[0] AS Category, COUNT(product\_id) AS no\_of\_products FROM part\_3\_retaildata WHERE category\_code != " GROUP BY SPLIT(category\_code,'\\.[0] ORDER BY no\_of\_products DESC;

```
hive> SELECT SPLIT(category_code,'\\.[0] AS Category, COUNT(product_id) AS no_of_products FROM part_3_retaildata WHERE category_code != '' GROUP BY SPLIT(category_code,'\\.[0] ORDER BY no_of_products DESC;
Query ID = hadoop_20210811205929_6268201a-cf93-4318-82ea-488dcd77f284
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_1628701152532_0012)
```

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: 64.07 s
OK
category      no_of_products
appliances     61736
stationery     26722
furniture      23604
apparel 18232
accessories    12929
sport          2
Time taken: 71.831 seconds, Fetched: 6 row(s)
```

### Insights:

- Time Taken to execute the above query is 71.831 sec.
- “appliances” has the highest number of products,
- “Sport” category has the least number of products,
- The number of products in the range of 23000 to 27000 is available in the "furniture" & "stationery" category

**Q6.** Which brand had the maximum sales in October and November combined?

**Query:** SELECT brand, SUM(price) AS max\_sales FROM part\_3\_retaildata WHERE brand <>" AND event\_type='purchase' GROUP BY brand ORDER BY max\_sales DESC LIMIT 1 ;

```
hive> SELECT brand, SUM(price) AS max_sales FROM part_3_retaildata WHERE brand <>' AND event_type='purchase' GROUP BY brand ORDER BY max_sales DESC LIMIT 1 ;
Query ID = hadoop_20210811211939_d3ffbe63-e42f-4d12-9829-2f71f6570a42
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1628701152532_0013)
```

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: 21.72 s
OK
brand      max sales
runail 148297.93999999968
Time taken: 22.446 seconds, Fetched: 1 row(s)
hive>
```

### Insights:

- Time Taken to execute the above query is 22.446 sec.
- Brand “runail” has the maximum sales i.e. 148297.94 for both months combined
- Brand "runail" is quite popular among users.

Q7. Which brands increased their sales from October to November?

**Query:** WITH monthly\_sales AS ( SELECT brand, SUM(CASE WHEN date\_format(event\_time, 'MM')=10 THEN price ELSE 0 END) AS Oct\_sales, SUM(CASE WHEN date\_format(event\_time, 'MM')=11 THEN price ELSE 0 END) AS Nov\_sales FROM part\_3\_retaildata WHERE event\_type='purchase' AND date\_format(event\_time, 'MM') IN ('10', '11') GROUP BY brand ) SELECT brand, Oct\_sales, Nov\_sales, Nov\_sales-Oct\_sales AS Sales\_diff FROM monthly\_sales WHERE (Nov\_sales - Oct\_sales)>0 ORDER BY Sales\_diff;

```
hive> WITH monthly_sales AS ( SELECT brand, SUM(CASE WHEN date_format(event_time, 'MM')=10 THEN price ELSE 0 END) AS Oct_sales, SUM(CASE WHEN date_format(event_time, 'MM')=11 THEN price ELSE 0 END) AS Nov_sales FROM part_3_retaildata WHERE event_type='purchase' AND date_format(event_time, 'MM') IN ('10', '11') GROUP BY brand ) SELECT brand, Oct_sales, Nov_sales, Nov_sales-Oct_sales AS Sales_diff FROM monthly_sales WHERE (Nov_sales - Oct_sales)>0 ORDER BY Sales_diff;
Query ID = hadoop_20210812175117_af6c3bec-9cf9-4b13-927c-6e44f943861b
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1628783204537_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: 39.35 s
-----
OK
brand  oct_sales  nov_sales  sales_diff
owale  2.54      3.1       0.56
cosima 20.229999999999997 20.929999999999996 0.6999999999999993
grace  100.91999999999997 102.60999999999999 1.6900000000000012
helloganic 0.0      3.1       3.1
skinity 8.88     12.440000000000001 3.5600000000000005
bodyton 1376.34000000000004 1380.64000000000003 4.2999999999999545
moyou  5.71     10.280000000000001 4.570000000000001
neoleor 43.41    51.7     8.290000000000006
soleo  204.19999999999995 212.52999999999998 8.3300000000000155
jaguar 1102.11000000000004 1110.6500000000003 8.539999999999964
tartio 236.16000000000003 245.79999999999998 9.639999999999958
fly 17.14   27.17   10.030000000000001
rasyan 18.799999999999997 28.939999999999998 10.14
deoproce 316.84000000000003 329.16999999999996 12.329999999999927
barbie 0.0     12.39   12.39
supertan 50.37   66.51   16.140000000000008
treaclemoon 163.37 181.49 18.120000000000005
kamill 63.009999999999999 81.490000000000001 18.480000000000018
juno 0.0     21.08   21.08
veraelara 50.11   71.210000000000001 21.100000000000001
glysolid 69.72999999999999 91.58999999999999 21.86
godefroy 401.22 425.12 23.899999999999977
binacil 0.0   24.259999999999998 24.259999999999998
blixz 38.95   63.39999999999999 24.449999999999999
profepil 93.36   118.02000000000001 24.660000000000001
estelare 444.81000000000003 471.87000000000006 27.059999999999975
orly 902.38 931.09 28.710000000000036
biore 60.650000000000006 90.31 29.659999999999997
beautyblender 78.740000000000001 109.41 30.669999999999987
```

```
bpw.style 11572.1500000000194 14837.4400000000333 3265.2900000000139
staleks 8519.7300000000018 11875.6100000000019 3355.8800000000001
freedecor 3421.77999999999975 7671.7999999999962 4250.0199999999964
runail 71539.279999999985 76758.659999999993 5219.3800000000077
polarus 6013.7200000000001 11371.9300000000006 5358.2100000000005
cosmoprofi 8322.8099999999989 14536.990000000007 6214.1800000000082
jessnail 26287.8400000000018 33345.2300000000156 7057.3899999999978
strong 29196.6300000000012 38671.270000000001 9474.64
ingarden 23161.390000000004 33566.210000000022 10404.8200000000178
lianail 5892.8399999999985 16394.239999999998 10501.4000000000012
uno 35302.03000000000086 51039.750000000007 15737.7199999999987
grattol 35445.5400000000045 71472.710000000033 36027.170000000325
474679 060000001146 619509 2400000115 144830.18000000005
Time taken: 44.135 seconds, Fetched: 161 row(s)
hive>
```

Insights:

- Time Taken to execute the above query is 44.135 sec.
- Total 161 brands increased their sales from October to November



**Q8.** 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:** SELECT user\_id, SUM(price) AS total\_spend FROM part\_3\_retaildata WHERE event\_type = 'purchase' GROUP BY user\_id ORDER BY total\_spend DESC LIMIT 10;

```
hive> SELECT user_id, SUM(price) AS total_spend FROM part_3_retaildata WHERE event_type = 'purchase' GROUP BY user_id ORDER BY total_spend DESC LIMIT 10;
Query ID = hadoop_20210812182241_2b56e1ba-fa9e-4075-ad56-24d229f8df02
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_1628783284537_0007)
```

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: 25.35 s
OK
user_id total_spend
557790271 2715.8699999999976
150318419 1645.9700000000003
562167663 1352.8500000000001
531900924 1329.4499999999999
557850743 1295.4800000000002
522130011 1185.39
561592095 1109.7000000000007
431950134 1097.5899999999999
566576008 1056.3600000000004
521347209 1040.91
Time taken: 33.589 seconds, Fetched: 10 row(s)
hive>
```

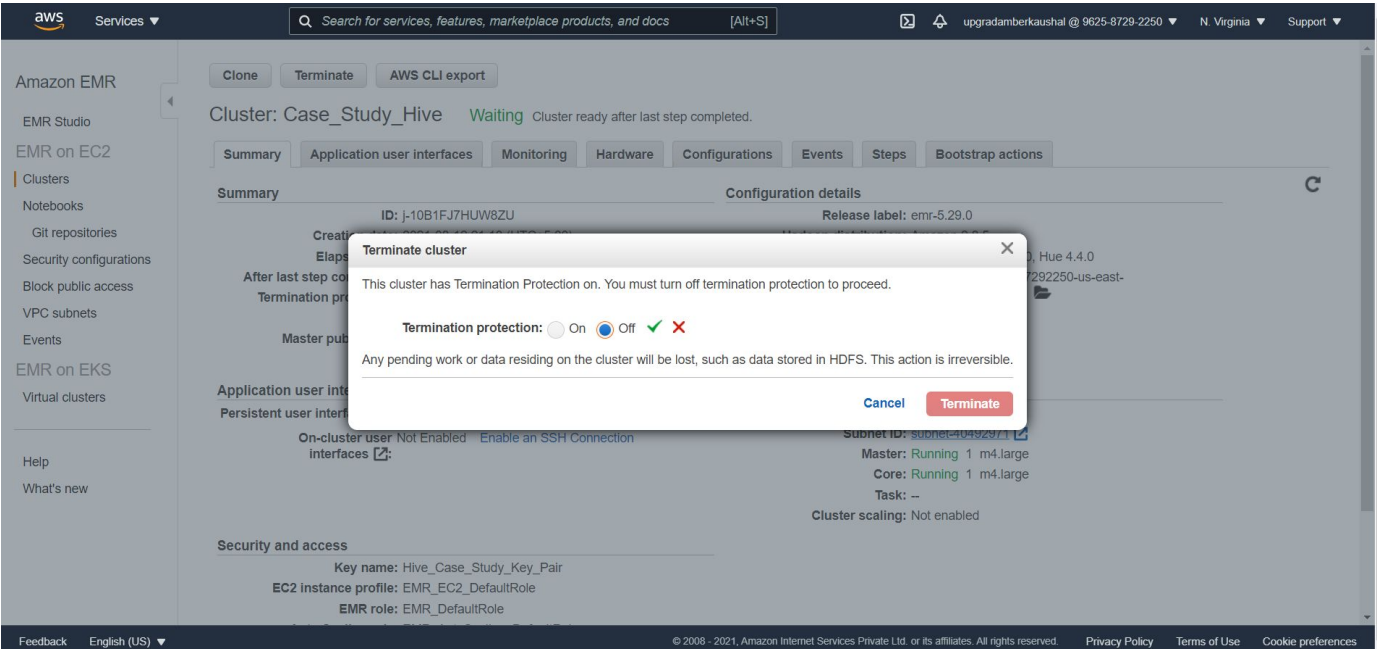
### Insights:

- Time Taken to execute the above query is 33.589 sec.
- We have got the list of the top ten users who spends the most. We can reward them with Golden Customer plan. It could attract new users.

# 9. Cleaning up

- We have done the analysis, Now deleting the database & terminating the cluster.

```
hive> drop database clickstream;
OK
Time taken: 0.155 seconds
hive> show databases;
OK
database_name
default
Time taken: 0.008 seconds, Fetched: 1 row(s)
```



12 clusters (all loaded)						
	Name	ID	Status	Creation time (UTC+5:30)	Elapsed time	Normalized instance hours
<input type="checkbox"/>	Case_Study_Hive	j-10B1FJ7HUW8ZU	Terminated User request	2021-08-12 21:10 (UTC+5:30)	3 hours, 8 minutes	24

Cluster Terminated