

# HIVE CASE STUDY STEPS

- **Key-pair creation and generation**

1. Click on Create Key pair

The screenshot shows the AWS EC2 Key Pairs page. On the left, there's a sidebar with various EC2-related options like Instances, Images, and Network & Security. The main area displays a table titled "Key pairs (2)" with columns for Name, Fingerprint, and ID. The table contains three rows: one unnamed key pair and two named ones ("Arpit\_629804944139\_2021-03-10 13..." and "Test"). A "Create key pair" button is located at the top right of the table.

2. Mention the name of a Key Pair with ppk file format and click create.

The screenshot shows the "Create key pair" wizard. Step 1: Key pair. It asks for a name (entered as "hive\_case\_study"), specifies the file format as "ppk" (selected from "pem" and "ppk"), and provides an optional tag field ("Add tag"). At the bottom, there are "Cancel" and "Create key pair" buttons.

### 3. Successfully Key Pair created.

The screenshot shows the AWS EC2 Key Pairs page. The left sidebar includes options like New EC2 Experience, Instances, Images, and Elastic Block Store. The main area displays a table titled 'Key pairs (1/3)' with columns for Name, Fingerprint, and ID. A message at the top says 'Successfully created key pair'. The table contains three rows: 'Arpit\_629804944139\_2021-03-10 13...', 'hive\_case\_study' (which is selected), and 'Test'.

Name	Fingerprint	ID
Arpit_629804944139_2021-03-10 13...	47:78:b2:c7:59:b3:cd:c9:6d:c4:60:26:2...	key-00049fe08bb55e30d
<b>hive_case_study</b>	52:86:a8:b3:f5:40:54:7b:89:bd:cfa:ef:...	key-0cf828841f0c1a0f2
Test	1b:41:8d:ac:73:88:1a:b2:a3:88:37:33:8...	key-067b18146a6a0ad1d

- **S3 Bucket to store data files**

#### 1. First we will create a new bucket by clicking on Create Bucket

The screenshot shows the AWS S3 Buckets page. The left sidebar includes options like Buckets, Storage Lens, and Feature spotlight. The main area displays a table titled 'Buckets (2)' with columns for Name, AWS Region, Access, and Creation date. The table lists 'arpit-hive' and 'aws-logs-629804944139-us-east-1'.

Name	AWS Region	Access	Creation date
arpit-hive	US East (N. Virginia) us-east-1	Objects can be public	April 1, 2021, 16:54:12 (UTC+05:30)
aws-logs-629804944139-us-east-1	US East (N. Virginia) us-east-1	Objects can be public	March 31, 2021, 02:05:57 (UTC+05:30)

2. Provide the unique name to the Bucket and uncheck the block all public access box for this case study.

**Create bucket**

Buckets are containers for data stored in S3. [Learn more](#)

**General configuration**

Bucket name  Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

Copy settings from existing bucket - optional  
Only the bucket settings in the following configuration are copied.  
[Choose bucket](#)

**Block Public Access settings for this bucket**

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

**Block all public access**  
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

**Block public access to buckets and objects granted through new access control lists (ACLs)**  
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access

Feedback English (US) © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

3. New Bucket created successfully with name “hive-casestudy”.

**Amazon S3**

**Buckets**

- Access Points
- Object Lambda Access Points
- Batch Operations
- Access analyzer for S3

Block Public Access settings for this account

**Storage Lens**

- Dashboards
- AWS Organizations settings

Feature spotlight (2)

▶ AWS Marketplace for S3

**Successfully created bucket "hive-casestudy"**  
To upload files and folders, or to configure additional bucket settings choose [View details](#).

**Buckets (3)**

Buckets are containers for data stored in S3. [Learn more](#)

Name	AWS Region	Access	Creation date
arpit-hive	US East (N. Virginia) us-east-1	Objects can be public	April 1, 2021, 16:54:12 (UTC+05:30)
aws-logs-629804944139-us-east-1	US East (N. Virginia) us-east-1	Objects can be public	March 31, 2021, 02:05:57 (UTC+05:30)
<b>hive-casestudy</b>	US East (N. Virginia) us-east-1	Objects can be public	April 3, 2021, 21:17:17 (UTC+05:30)

https://s3.console.aws.amazon.com/s3/buckets/hive-casestudy?region=us-east-1 © 2008 - 2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

#### 4. Upload the csv files to the new bucket created.

The screenshot shows the AWS S3 'Upload' interface. At the top, there's a search bar and navigation links. Below it, a table lists 'Files and folders (2 Total, 980.7 MB)' with columns for Name, Folder, Type, and Size. Two files are listed: '2019-Nov.csv' (application/vnd.ms-excel, 520.6 MB) and '2019-Oct.csv' (application/vnd.ms-excel, 460.2 MB). Below the table is a 'Destination' section where the target bucket is set to 's3://hive-casestudy'. Under 'Bucket Versioning', 'Enabled' is selected. Under 'Default encryption', 'Disabled' is selected. Under 'Object Lock', 'Disabled' is selected. At the bottom right are 'Cancel' and 'Upload' buttons.

The screenshot shows the 'Upload: status' page after the upload completed successfully. It displays a summary table with 'Destination' (s3://hive-casestudy), 'Succeeded' (2 files, 980.7 MB (100.00%)), and 'Failed' (0 files, 0 B (0%)). Below this is a 'Files and folders' table with the same two CSV files listed, each with a status column showing 'Succeeded'. Navigation links at the bottom include 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

## • CLUSTER CREATION

1. In Amazon EMR home page click on create cluster button.

The screenshot shows the Amazon EMR home page. On the left, there's a sidebar with navigation links for 'Clusters', 'Notebooks', 'Git repositories', 'Security configurations', 'Block public access', 'VPC subnets', 'Events', 'EMR on EKS', and 'Virtual clusters'. The main area has a search bar at the top with the placeholder 'Search for services, features, marketplace products, and docs' and a 'Create cluster' button. Below the search bar is a table titled '3 clusters (all loaded)'. The table columns are 'Name', 'ID', 'Status', 'Creation time (UTC+5:30)', 'Elapsed time', and 'Normalized instance hours'. The table contains three rows:

Name	ID	Status	Creation time (UTC+5:30)	Elapsed time	Normalized instance hours
Hive_case_study	j-9IC4SDT90UUA	Terminated User request	2021-04-03 20:21 (UTC+5:30)	47 minutes	8
upgrad-cluster	j-31GZ2PGS6JYNB	Terminated User request	2021-04-01 15:18 (UTC+5:30)	6 hours, 7 minutes	48
upgrad-cluster	j-2PSW6P1CXDQK	Terminated with errors Instance failure	2021-03-31 02:05 (UTC+5:30)	36 minutes	8

At the bottom of the page, there are links for 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

2. Then click on 'Go to advanced options'.

The screenshot shows the 'Create Cluster - Quick Options' page. It has several tabs at the top: 'General Configuration', 'Software configuration', 'Hardware configuration', and 'Security and access'. The 'General Configuration' tab is active. The fields include:

- Cluster name:** My cluster
- Logging:** S3 folder: s3://aws-logs-629804944139-us-east-1/elasticsearchmapred/
- Launch mode:** Cluster (selected)

The 'Software configuration' tab shows:

- Release:** emr-5.32.0
- Applications:** (empty field)

The 'Hardware configuration' tab shows:

- Instance type:** m5.xlarge
- Number of instances:** 3 (1 master and 2 core nodes)

The 'Security and access' tab shows:

- EC2 key pair:** Choose an option
- Permissions:** Default (selected)
- EMR role:** EMR\_DefaultRole
- EC2 instance profile:** EMR\_EC2\_DefaultRole

At the bottom of the page, there are links for 'Feedback', 'English (US)', 'Privacy Policy', 'Terms of Use', and 'Cookie preferences'.

3. In software and steps page we have to select all the required services needed to complete this analysis. We are selecting emr-5.29.0 release as per the instruction with required services.

The screenshot shows the 'Create Cluster - Advanced Options' page. Under 'Step 1: Software and Steps', the 'Release' dropdown is set to 'emr-5.29.0'. In the 'Software Configuration' section, several services are selected: Hadoop 2.8.5, Hive 2.3.6, Hue 4.4.0, Spark 2.4.4, 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, Livy 0.6.0, Flink 1.9.1, Pig 0.17.0, ZooKeeper 3.4.14, Mahout 0.13.0, Oozie 5.1.0, and TensorFlow 1.14.0. Below this, under 'Multiple master nodes (optional)', there is a checkbox for 'Use multiple master nodes to improve cluster availability'. Under 'AWS Glue Data Catalog settings (optional)', there are checkboxes for 'Use for Hive table metadata' and 'Use for Spark table metadata'. A 'Edit software settings' button is present. The 'Steps (optional)' section contains a note about steps being units of work and a 'Concurrency' checkbox for running multiple steps at once. The 'After last step completes' dropdown is set to 'Clusters enters waiting state'. At the bottom, there are links for Feedback, English (US), Privacy Policy, Terms of Use, and Cookie preferences.

4. Then we will proceed to next page, in this step we have to specify the required configuration needed for the cluster. Here we have to select the instance type and instance count for master and core node. We have selected instance type as m4.large and 1 instance count for both master and core node as we need a 2-node cluster for the analysis.

The screenshot shows the 'Cluster Nodes and Instances' configuration page. It lists two nodes: 'Master - 1' and 'Core - 2'. Both nodes are of type 'm4.large' with 1 instance each. Under 'Purchasing option', for both nodes, the 'On-demand' radio button is selected. A message at the top indicates that console options for automatic scaling have changed. The bottom of the page includes links for Feedback, English (US), Privacy Policy, Terms of Use, and Cookie preferences.

5. Here we have given the name of the cluster as “Hive-Case-Study”.

Create Cluster - Advanced Options [Go to quick options](#)

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

**General Options**

Cluster name: Hive-Case-Study

Logging [?](#)  
 S3 folder [s3://aws-logs-629804944139-us-east-1/elasticmapreduce/](#) [?](#)  
 Debugging [?](#)  
 Termination protection [?](#)

**Tags** [?](#)

Key	Value (optional)
Add a key to create a tag	

6. We have selected the created EC2 key pair and then click on the create cluster option.

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: **hive\_case\_study**  [?](#)  
 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](#) [?](#)  
EC2 Instance profile: [EMR\\_EC2\\_DefaultRole](#) [?](#)  
Auto Scaling role: [EMR\\_AutoScaling\\_DefaultRole](#) [?](#)

[► Security Configuration](#)  
[► EC2 security groups](#)

[Cancel](#) [Previous](#) **Create cluster**

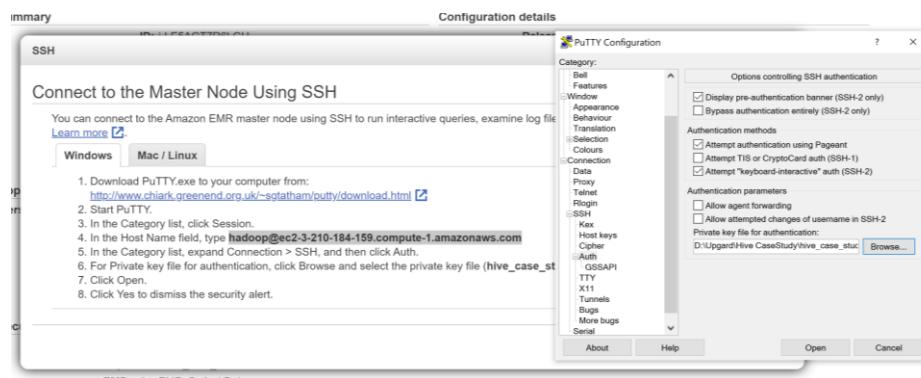
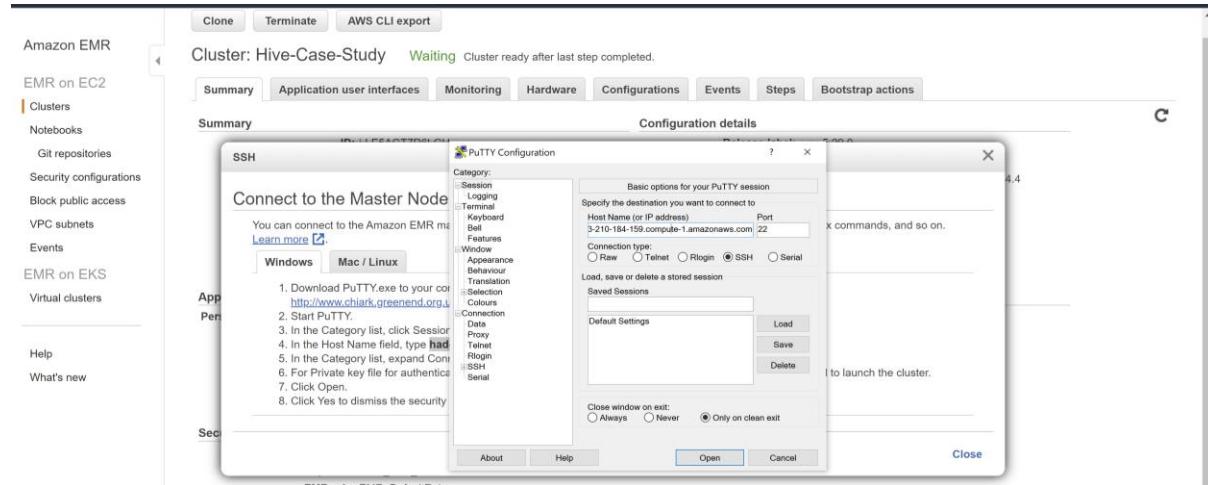
7. Then cluster has been created and will be in the “Starting” state and will change to “Waiting” state after sometime.

The screenshot shows the AWS EMR Cluster Overview page for a cluster named "Hive-Case-Study". The cluster status is "Starting". The left sidebar shows navigation options for Amazon EMR, EMR on EC2, Clusters, Notebooks, Git repositories, Security configurations, Block public access, VPC subnets, Events, EMR on EKS, Virtual clusters, Help, and What's new. The main content area includes tabs for Summary, Application user interfaces, Monitoring, Hardware, Configurations, Events, Steps, and Bootstrap actions. The Summary tab is selected. It displays cluster details such as ID: j-LE5AGT7R6LGU, Creation date: 2021-04-03 21:25 (UTC+5:30), Elapsed time: 0 seconds, and After last step completes: Cluster waits. Configuration details show 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, Spark 2.4.4, Log URI: s3://aws-logs-62980494139-us-east-1/elasticmapreduce/, EMRFS consistent view: Disabled, and Custom AMI ID: --. Network and hardware details include Availability zone: --, Subnet ID: subnet-0de9ce03, Master: Provisioning 1 m4.large, Core: Provisioning 1 m4.large, Task: --, and Cluster scaling: Not enabled. The Security and access section lists Key name: hive\_case\_study, EC2 instance profile: EMR\_EC2\_DefaultRole, EMR role: EMR\_DefaultRole, Auto Scaling role: EMR\_AutoScaling\_DefaultRole, and security groups for Master and Core. The bottom of the page includes links for Feedback, English (US), Privacy Policy, Terms of Use, and Cookie preferences.

8. Now cluster is started as Master and Core both are in “Running” State.

The screenshot shows the AWS EMR Cluster Overview page for the same cluster "Hive-Case-Study". The cluster status is now "Waiting". The left sidebar and tabs are identical to the previous screenshot. The main content area shows the cluster details and configuration settings. The "Waiting" status indicates that the cluster is ready after the last step completed. The bottom of the page includes links for Feedback, English (US), Privacy Policy, Terms of Use, and Cookie preferences.

9. Once cluster is running state we have to click on Master public DNS. We have to open the putty configuration and then give the host name (master node DNS) and then browse to the private key file location by clicking on Connection → SSH → Auth.



10. Then EMR command line interface will be open and login as Hadoop.

```

Using username "hadoop".
Authenticating with public key "hive_case_study"
Last login: Sun Apr  4 11:52:54 2021
[ -| ( _|_ ) /   Amazon Linux AMI
[ _\|_\|_|]

https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
55 package(s) needed for security, out of 97 available
Run "sudo yum update" to apply all updates.

EEEEEEEEEEEEEEEEEE MMMMMMM MBBBBBBB RRRRRRRRRRRRRRR
E:::::::::::E E M:::::::M M:::::::M R:::::R R:::::R
EE:::::EEEEE:::E M:::::::M M:::::::M R:::::R R:::::R
 E:::::E EEEEE M:::::::M M:::::::M RR:::::R R:::::R
 E:::::E M:::::::M:::M M:::::::M R:::::R R:::::R
 E:::::EEEEE EEE M:::::M M:::::M M:::::M R:::::R R:::::R
 E:::::::::::E E M:::::M M:::::M M:::::M R:::::R R:::::R
 E:::::EEEEE EEE M:::::M M:::::::M M:::::M R:::::R R:::::R
 E:::::E M:::::M M:::::M M:::::M R:::::R R:::::R
 E:::::E EEEE E M:::::M MMM M:::::M R:::::R R:::::R
EE:::::EEEEE E E M:::::M M:::::M R:::::R R:::::R
 E:::::::::::E E M:::::M M:::::M R:::::R R:::::R
 EEEEEEEEEEEEEEEEEE MMMMMMM RRRRRRRRRRRRRRR

```

- **Copying the data into HDFS**

1.To access the public s3 bucket.

**Command: aws s3 ls hive-casestudy**

```
[hadoop@ip-172-31-78-144 ~]$ aws s3 ls hive-casestudy
2021-04-03 15:48:16 545839412 2019-Nov.csv
2021-04-03 15:48:16 482542278 2019-Oct.csv
[hadoop@ip-172-31-78-144 ~]$
```

2.Creating a directory in HDFS.

**Command: hadoop fs -mkdir /user/hive/ecomcasestudy**

3.Checking the available directory.

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

```
[hadoop@ip-172-31-72-97 ~]$ hadoop fs -mkdir /user/hive/ecomcasestudy
[hadoop@ip-172-31-72-97 ~]$ hadoop fs -ls /user/hive
Found 2 items
drwxr-xr-x  - hadoop hadoop      0 2021-04-04 11:56 /user/hive/ecomcasestudy
drwxrwxrwt  - hdfs  hadoop      0 2021-04-04 11:48 /user/hive/warehouse
```

4>Loading the s3 public data set to created directory “ecomcasestudy” in hadoop .

**Command: hadoop distcp 's3://hive-casestudy/2019-Oct.csv' /user/hive/ecomcasestudy/oct.csv**

**hadoop distcp 's3://hive-casestudy/2019-Nov.csv' /user/hive/ecomcasestudy/nov.csv**

```
[hadoop@ip-172-31-72-97 ~]$ hadoop distcp 's3://hive-casestudy/2019-Nov.csv' /user/hive/ecomcasestudy/nov.csv
21/04/04 12:00:22 INFO tools.DistCp: Input Options: 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://hive-casestudy/2019-Nov.csv], targetPath=/user/hive/ecomcasestudy/nov.csv, targetPathExists=false, filtersFile='null'}
21/04/04 12:00:22 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-72-97.ec2.internal/172.31.72.97:8032
21/04/04 12:00:26 INFO tools.SimpleCopyListing: Paths (files+dirs) cnt = 1; dirCnt = 0
21/04/04 12:00:26 INFO tools.SimpleCopyListing: Build file listing completed.
21/04/04 12:00:26 INFO Configuration.deprecation: io.sort.mb is deprecated. Instead, use mapreduce.task.io.sort.mb
21/04/04 12:00:26 INFO Configuration.deprecation: io.sort.factor is deprecated. Instead, use mapreduce.task.io.sort.factor
21/04/04 12:00:26 INFO tools.DistCp: Number of paths in the copy list: 1
21/04/04 12:00:26 INFO tools.DistCp: Number of paths in the copy list: 1
21/04/04 12:00:26 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-72-97.ec2.internal/172.31.72.97:8032
21/04/04 12:00:26 INFO mapreduce.JobSubmitter: number of splits:1
21/04/04 12:00:27 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1617537007743_0002
21/04/04 12:00:27 INFO impl.YarnClientImpl: Submitted application application_1617537007743_0002
21/04/04 12:00:27 INFO mapreduce.Job: The url to track the job: http://ip-172-31-72-97.ec2.internal:20888/proxy/application_1617537007743_0002/
21/04/04 12:00:27 INFO tools.DistCp: DistCp job-id: job_1617537007743_0002
21/04/04 12:00:27 INFO mapreduce.Job: Running job: job_1617537007743_0002
21/04/04 12:00:35 INFO mapreduce.Job: Job job_1617537007743_0002 running in uber mode : false
21/04/04 12:00:35 INFO mapreduce.Job: map 0% reduce 0%
21/04/04 12:00:52 INFO mapreduce.Job: map 100% reduce 0%
21/04/04 12:00:54 INFO mapreduce.Job: Job job_1617537007743_0002 completed successfully
21/04/04 12:00:54 INFO mapreduce.Job: Counters: 38
File System Counters
  FILE: Number of bytes read=0
  FILE: Number of bytes written=172837
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=352
  HDFS: Number of bytes written=545839412
  HDFS: Number of read operations=12
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=4
  S3: Number of bytes read=545839412
  S3: Number of bytes written=0
```

```

FILE: Number of write operations=0
HDFS: Number of bytes read=352
HDFS: Number of bytes written=545839412
HDFS: Number of read operations=12
HDFS: Number of large read operations=0
HDFS: Number of write operations=4
S3: Number of bytes read=545839412
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=1
Other local map tasks=1
Total time spent by all maps in occupied slots (ms)=545376
Total time spent by all reduces in occupied slots (ms)=0
Total time spent by all map tasks (ms)=17043
Total vcore-milliseconds taken by all map tasks=17043
Total megabyte-milliseconds taken by all map tasks=17452032
Map-Reduce Framework
Map input records=1
Map output records=0
Input split bytes=135
Spilled Records=0
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=299
CPU time spent (ms)=21030
Physical memory (bytes) snapshot=617218048
Virtual memory (bytes) snapshot=3300544512
Total committed heap usage (bytes)=515375104
File Input Format Counters
Bytes Read=217
File Output Format Counters
Bytes Written=0
DistCp Counters
Bytes Copied=545839412
Bytes Expected=545839412
Files Copied=1
[hadoop@ip-172-31-72-97 ~]$ 

```

5.After loading the data set we have used below command to check the data set file and data set in the hadoop directory.

**Command: hadoop fs -ls /user/hive/ecomcasestudy/**

**Command: hadoop fs -cat /user/hive/ecomcasestudy/oct.csv | head**

```

[hadoop@ip-172-31-72-97 ~]$ hadoop fs -cat /user/hive/ecomcasestudy/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,2.62,430174032,73deale7-664e-43f4-8b30-d32b9d5af04f
2019-10-01 00:00:19 UTC,cart,5739055,148758000824612266,,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-172-31-72-97 ~]$ 

```

**Command: hadoop fs -cat /user/hive/ecomcasestudy/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-alcc-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-alcc-af0575a34f
2019-11-01 00:00:24 UTC,remove_from_cart,5826182,1487580007483048900,,,3.33,553329724,2067216c-31b5-455d-alcc-af0575a34f
2019-11-01 00:00:25 UTC,view,5856189,148758000926551821,,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,0.79,429913900,2f0bff3c-252f-4fe6-afcd-5d8a6a

```

- **Database & Table Creation**

Create the Database

```
hive> create database ECOMDB;
OK
Time taken: 0.225 seconds
hive>
```

1. After moving the data to the directory, we create the base table(ecom\_data) and check for the data in the table.

**Command:**

```
CREATE TABLE IF NOT EXISTS ecom_data (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/ecomcasestudy/' tblproperties('skip.header.line.count'=1');
```

```
hive> CREATE TABLE IF NOT EXISTS ecom_data (event_time timestamp, event_type string , product_id string , category_id st
ring , 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/ecomcasestudy/' tblproperties('skip.header.line.count'=1');
OK
Time taken: 1.04 seconds
hive>
```

```
hive> select * from ecom_data limit 5;
OK
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-6c99-46
b1-834f-33527f4de241
2019-11-01 00:00:09 UTC cart    5844397 1487580006317032337          2.38   553329724 2067216c-31b5-45
5d-alcc-af0575a34ffb
2019-11-01 00:00:10 UTC view    5837166 1783999064103190764          pnb    22.22   556138645 57ed222e-a54a-49
07-9944-5a875c2d7f4f
2019-11-01 00:00:11 UTC cart    5876812 1487580010100293687          jessnail   3.16   564506666 186c1951
-8052-4b37-adce-dd9644b1d5f7
Time taken: 2.096 seconds, Fetched: 5 row(s)
```

2. Once the base table is created, we need to optimize the table for quick query result through partitioning and bucketing. Our optimized table name is **ecom\_data\_part**.

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

**Command:**

```
CREATE TABLE IF NOT EXISTS ecom_data_part (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 (category_code) into 12 buckets
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
STORED AS TEXTFILE
LOCATION '/user/hive/ecomcasestudy/' tblproperties('skip.header.line.count'=1');
```

```

hive> CREATE TABLE IF NOT EXISTS ecom_data_part (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 (category_code) into 12 buckets
      > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
      > STORED AS TEXTFILE
      > LOCATION '/user/hive/ecomcasestudy/' tblproperties('skip.header.line.count'='1');
OK
Time taken: 0.11 seconds
hive>

```

### 3. Loading the data into optimize table from base table.

#### Command:

##### **Insert into table ecom\_data\_part partition (event\_type)**

```

select event_time , product_id , category_id , category_code , brand , price , user_id , user_session
, event_type from ecom_data ;

```

```

hive> Insert into table ecom_data_part partition (event_type)
      > select event_time , product_id , category_id , category_code , brand , price , user_id , user_session , event_type
      from ecom_data ;
Query ID = hadoop_20210404121016_accd5532-d54a-45f1-b77b1b0ac3e4
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_1617537007743_0004)

-----  

      VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  

-----  

Map 1 ..... container  SUCCEEDED    8       8       0       0       0       0       0  

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

-----  

VERTICES: 0/202  [=====>>>] 100%  ELAPSED TIME: 159.79 s  

-----  

Loading data to table default.ecom_data_part partition (event_type=null)  

Loaded : 5/5 partitions.  

      Time taken to load dynamic partitions: 1.199 seconds  

      Time taken for adding to write entity : 0.004 seconds  

OK
Time taken: 173.433 seconds
hive>

```

```

hive> set hive.cli.print.header=true ;
hive> select * from ecom_data_part limit 5;
OK
ecom_data_part.event_time      ecom_data_part.product_id      ecom_data_part.category_id      ecom_data_part.category_
code      ecom_data_part.brand      ecom_data_part.price      ecom_data_part.user_id      ecom_data_part.user_session      ecom_dat
a_part.event_type
2019-10-27 10:48:31 UTC 5812230 1487580005427839846          irisk    2.48      556028109      61ead6df-aac2-4167-8d4d-
484a3337aa1c      cart
2019-10-28 03:29:28 UTC 5787019 1487580005595612013          4.11      556453136      e1971dce-73f1-484b-a105-
10fad2538d78      cart
2019-10-28 03:29:29 UTC 47610   1487580007717929935          0.79      178186931      1ab7b37f-9d6e-4bc7-8df4-
287eaf05153f      cart
2019-10-26 22:25:01 UTC 5896615 1487580005427839846          irisk    2.48      468176825      14b007c6-468f-477d-82e0-
56359e34c1fd      cart
2019-10-29 13:30:53 UTC 5817692 1487580010872045658          0.79      72529230      02a293d0-75c4-428e-a05b-
8239bd9121d3      cart
Time taken: 0.19 seconds, Fetched: 5 row(s)
hive>

```

- **Query Analysis**

### 1. Find the revenue generated due to purchases made in October

**Code:**

```
select sum(price) as total_revenue from ecom_data where month(event_time)=10 and event_type = 'purchase' ;
```

```
hive> select sum(price) as total_revenue from ecom_data where month(event_time)=10 and event_type = 'purchase' ;
Query ID = hadoop_20210404121458_f6e7b660-eeb5-4946-843a-2769ad815ce7
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1617537007743_0004)

-----  
 VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  
-----  
Map 1 ..... container  SUCCEEDED    6        6        0        0        0        0  
Reducer 2 ..... container  SUCCEEDED    1        1        0        0        0        0  
-----  
VERTICES: 02/02  [=====>>] 100%  ELAPSED TIME: 122.70 s  
-----  
OK  
total_revenue  
1211538.4299997308  
Time taken: 123.919 seconds, Fetched: 1 row(s)
hive> █
```

**Note:** The below screenshot of the same query from both the base table and the bucketed table. When compared the bucketed table takes less time to query the result than the base table. This is the use of partitioning and bucketing the data.

**Code:**

```
select sum(price) as total_revenue from ecom_data_part where month(event_time)=10 and event_type = 'purchase' ;
```

```
hive> select sum(price) as total_revenue from ecom_data where month(event_time)=10 and event_type = 'purchase' ;
Query ID = hadoop_20210404121458_f6e7b660-eeb5-4946-843a-2769ad815ce7
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1617537007743_0004)

-----  
 VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  
-----  
Map 1 ..... container  SUCCEEDED    6        6        0        0        0        0  
Reducer 2 ..... container  SUCCEEDED    1        1        0        0        0        0  
-----  
VERTICES: 02/02  [=====>>] 100%  ELAPSED TIME: 122.70 s  
-----  
OK  
total_revenue  
1211538.4299997308  
Time taken: 123.919 seconds, Fetched: 1 row(s)
hive> select sum(price) as total_revenue from ecom_data_part where month(event_time)=10 and event_type = 'purchase' ;
Query ID = hadoop_20210404121801_5009aa18-c3fe-4269-8e21-0074ac3bf975
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1617537007743_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: 16.56 s  
-----  
OK  
total_revenue  
1211538.4299998144  
Time taken: 17.512 seconds, Fetched: 1 row(s)
```

### Insights:

- The Total revenue generated based on Purchase in the month of October 2019 was **1,211,538.43/-**. We tried getting the result using both base table[ecom\_data] and optimized table[ecom\_data\_part]. We figured out that Bucketed table reduces the query time when compared to the base table by approx. 107 seconds.

Hence moving forward, we will use the commands using optimized table for less query time.

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

#### Code:

```
SELECT date_format(event_time, 'MM') AS Months, COUNT(event_type) AS Sum_of_Purchases
FROM ecom_data_part WHERE event_type='purchase'
GROUP BY date_format(event_time, 'MM');
```

```
hive> SELECT date_format(event_time, 'MM') AS Months, COUNT(event_type) AS Sum_of_Purchases
    > FROM ecom_data_part WHERE event_type='purchase'
    > GROUP BY date_format(event_time, 'MM');
Query ID = hadoop_20210404122640_d7477ca6-a828-4618-a880-22db22242337
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_1617537007743_0005)

-----  
 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: 22.44 s  
-----  
OK  
months  sum_of_purchases  
10      245624  
11      322417  
Time taken: 31.212 seconds, Fetched: 2 row(s)
hive>
```

### Insights:

- It seems to be that there was more purchase made in the month of November (11) i.e., 322,417 than in the month of October (10) i.e., 245,624.
- Looking at these figures we could assume that the month of November must be more profitable than the month of October.

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

**Code:**

```
WITH rev_difference AS
(SELECT
SUM(case when MONTH(event_time) = '10' then price else 0 end) AS Oct_purchase,
SUM(case when MONTH(event_time) = '11' then price else 0 end) AS Nov_purchase
FROM ecom_data_part
WHERE event_type= 'purchase')
SELECT (Nov_purchase - Oct_purchase) as difference_revenue FROM rev_difference ;
```

```
hive> WITH rev_difference AS
> (SELECT
> SUM(case when MONTH(event_time) = '10' then price else 0 end) AS Oct_purchase,
> SUM(case when MONTH(event_time) = '11' then price else 0 end) AS Nov_purchase
> FROM ecom_data_part
> WHERE event_type= 'purchase'
> ) SELECT (Nov_purchase - Oct_purchase) as difference_revenue FROM rev_difference ;
Query ID = hadoop_20210404123258_dc937194-5442-44a4-ab4d-bbc1851d9fbe
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_1617537007743_0006)

-----  
 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: 18.59 s  
-----  
OK  
difference_revenue  
319478.4700001469  
Time taken: 28.115 seconds, Fetched: 1 row(s)
```

**Insights:**

- It is seen that the difference in revenue between October and November month is 319478.4700.
- As per the result, we could conclude that the revenue generated in November of 2019 was more than the revenue generated in the month of October. In other words, November was more profitable and had a better sale.

#### 4. Find the distinct categories of products. Categories with null value can be ignored.

**Code:**

```
SELECT DISTINCT SPLIT(category_code,'\\.')[0] AS Category
FROM ecom_data_part
WHERE SPLIT(category_code,'\\.')[0] <> '';
```

```
hive> SELECT DISTINCT SPLIT(category_code,'\\.')[0] AS Category
> FROM ecom_data_part
> WHERE SPLIT(category_code,'\\.')[0] <> '';
Query ID = hadoop_20210404193335_57d4d9b3-8568-45ee-9b9d-21abc4c71c4a
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1617562957470_0005)

-----  
 VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  
-----  
Map 1 ..... container    SUCCEEDED      4        4        0        0        0        0  
Reducer 2 ..... container    SUCCEEDED      5        5        0        0        0        0  
-----  
VERTICES: 02/02  [=====>] 100%  ELAPSED TIME: 66.26 s  
-----  
OK  
category  
furniture  
appliances  
accessories  
apparel  
sport  
stationery  
Time taken: 67.673 seconds, Fetched: 6 row(s)
hive> 
```

**Insights:**

- There are total 6 different categories under which company sells their different products. i.e Furniture, appliances, accessories, apparel, sport and stationery.

## 5. Find the total number of products available in each category.

**Code:**

```
SELECT SPLIT(category_code,'\\.')[0] AS Category, COUNT(product_id) AS No_of_products
FROM ecom_data_part
WHERE SPLIT(category_code,'\\.')[0] <> ''
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 ecom_data_part
> WHERE SPLIT(category_code,'\\.')[0] <> ''
> GROUP BY SPLIT(category_code,'\\.')[0]
> ORDER BY No_of_products DESC;
Query ID = hadoop_20210404124606_137fdfbb-5d99-47ec-b964-62c5f5170308
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1617537007743_0006)

-----  
 VERTICES      MODE      STATUS    TOTAL   COMPLETED   RUNNING   PENDING   FAILED   KILLED  
-----  
Map 1 ..... container  SUCCEEDED    7        7          0          0          0          0          0  
Reducer 2 ..... container  SUCCEEDED    5        5          0          0          0          0          0  
Reducer 3 ..... container  SUCCEEDED    1        1          0          0          0          0          0  
-----  
VERTICES: 03/03  [=====>>] 100%  ELAPSED TIME: 71.98 s  
-----  
OK  
category      no_of_products  
appliances    61736  
stationery    26722  
furniture     23604  
apparel       18232  
accessories   12929  
sport         2
```

### Insights:

- Appliance with i.e., 61,736 products is the leading category, followed by stationery as second furniture as third.
- Sports category has only 2 products registered. This must be due to low cosmetic products in the sports market.

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

Code:

```
SELECT brand,sum(price) as total_price from ecom_data_part where brand !=''  
and event_type ='purchase'  
group by brand order by total_price desc limit 1;
```

```
hive> SELECT brand,sum(price) as total_price from ecom_data_part where brand !=''  
    > and event_type ='purchase'  
    > group by brand order by total_price desc limit 1;  
Query ID = hadoop_20210404125208_739cc8ec-3af2-4162-bab1-1319aabf45b7  
Total jobs = 1  
Launching Job 1 out of 1  
Status: Running (Executing on YARN cluster with App id application_1617537007743_0006)  
  
-----  
 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  
Reducer 3 ..... container  SUCCEEDED   1       1       0       0       0       0  
-----  
VERTICES: 03/03  [=====>>] 100%  ELAPSED TIME: 15.83 s  
-----  
OK  
brand  total_price  
runail  148297.9400000478  
Time taken: 16.523 seconds, Fetched: 1 row(s)  
hive> █
```

### Insights:

- Runail is the brand that has highest / maximum sales in the month of October and November of 2019 combined.
- Reason being cosmetic lovers are loyal towards Runail brand and hence increasing the products related to Runail could further increase the profit.

## 7.Which brands increased their sales from October to November?

Code:

```
WITH Monthly_Revenue AS (
  SELECT brand,
    SUM(CASE WHEN date_format(event_time, 'MM')=10 THEN price ELSE 0 END) AS Oct_Revenue,
    SUM(CASE WHEN date_format(event_time, 'MM')=11 THEN price ELSE 0 END) AS Nov_Revenue
  FROM ecom_data_part
  WHERE event_type='purchase' AND date_format(event_time, 'MM') IN ('10', '11')
  GROUP BY brand)
  SELECT brand, Oct_Revenue, Nov_Revenue, Nov_Revenue-Oct_Revenue AS Sales_Difference
  FROM Monthly_Revenue
  WHERE (Nov_Revenue - Oct_Revenue)>0
  ORDER BY Sales_Difference;
```

```
hive> WITH Monthly_Revenue AS (
  >   SELECT brand,
  >     SUM(CASE WHEN date_format(event_time, 'MM')=10 THEN price ELSE 0 END) AS Oct_Revenue,
  >     SUM(CASE WHEN date_format(event_time, 'MM')=11 THEN price ELSE 0 END) AS Nov_Revenue
  >   FROM ecom_data_part
  >   WHERE event_type='purchase' AND date_format(event_time, 'MM') IN ('10', '11')
  >   GROUP BY brand)
  >   SELECT brand, Oct_Revenue, Nov_Revenue, Nov_Revenue-Oct_Revenue AS Sales_Difference
  >   FROM Monthly_Revenue
  >   WHERE (Nov_Revenue - Oct_Revenue)>0
  >   ORDER BY Sales_Difference;
Query ID = hadoop_20210404133419_16157ed6-e57c-4680-bdd4-ed08fdb4e6e
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_1617537007743_0008)

-----  

      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  

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

-----  

VERTICES: 03/03  [=====>>] 100%  ELAPSED TIME: 33.01 s  

-----  

OK  

brand  oct_revenue      nov_revenue      sales_difference  

ovale  2.54            3.1             0.56  

cosima 20.2299999999997 20.93           0.700000000000028  

grace  100.919999999996 102.6100000000003 1.690000000000688  

hellogenic 0.0          3.1             3.1  

skinny 8.88           12.440000000000001 3.560000000000005  

bodyton 1376.3400000000004 1380.639999999976 4.29999999997226  

moyou  5.71           10.280000000000001 4.570000000000001  

neoleor 43.41          51.7             8.290000000000006  

soleo  204.199999999953 212.5300000000034 8.33000000000808  

jaguar 1102.1100000000001 1110.6500000000005 8.54000000000418  

tertio  236.1600000000003 245.799999999999 9.63999999999873
```

deoproce	316.8399999999999	329.1700000000001	12.330000000000155
barbie	0.0	12.39	12.39
supertan	50.37	66.5100000000003	16.140000000000036
treaclemoon	163.37000000000003	181.49000000000015	18.12000000000012
kamill	63.01000000000005	81.4900000000001	18.48000000000004
juno	0.0	21.08	21.08
veraclara	50.11	71.2100000000001	21.10000000000001
glysolid	69.7299999999999	91.5899999999997	21.85999999999985
godefroy	401.22	425.1200000000023	23.900000000000205
binacil	0.0	24.25999999999998	24.25999999999998
blixz	38.94999999999996	63.3999999999994	24.44999999999946
profepil	93.3600000000006	118.0200000000001	24.65999999999954
estelare	444.8100000000003	471.8700000000001	27.05999999999983
orly	902.380000000006	931.09	28.70999999999468
biore	60.6500000000006	90.31	29.65999999999997
beautyblender	78.7400000000001	109.4099999999998	30.66999999999973
vilenta	197.6	231.2100000000015	33.610000000000156
mavala	409.039999999996	446.3199999999977	37.279999999998
likato	296.059999999995	340.969999999999	44.9099999999997
ladykin	125.65	170.5700000000002	44.92000000000016
foamie	35.04	80.4899999999998	45.4499999999998
elskin	251.0900000000012	307.649999999999	56.5599999999998
balbcare	155.3300000000004	212.3799999999977	57.04999999999973
koelcia	55.5	112.7500000000001	57.25000000000014
profhenna	679.229999999998	736.8500000000008	57.62000000000103
kares	0.0	59.45	59.45
marutaka-foot	49.22	109.3300000000001	60.11000000000014
dewal	0.0	61.2899999999999	61.2899999999999
imm	288.02	351.2100000000001	63.19000000000011
laboratorium	246.5	312.5200000000004	66.02000000000004
cutrin	299.369999999995	367.62	68.2500000000006
egomania	77.47	146.0400000000002	68.5700000000002
konad	739.829999999997	810.669999999994	70.83999999999969
nirvel	163.04	234.329999999998	71.289999999982
koelf	422.729999999999	507.29	84.5600000000012
plazan	101.369999999999	194.0099999999993	92.6399999999994
aura	83.95	177.5099999999996	93.5599999999996
kerasys	430.909999999997	525.2	94.2900000000008
enjoy	41.35	136.5700000000002	95.2200000000003
depilflax	2707.069999999965	2803.779999999975	96.71000000000095

zeitun	708.6600000000001	2009.6300000000003	1300.9700000000003
joico	705.5200000000001	2015.1000000000001	1309.58
severina	4775.8799999998	6120.48000000021	1344.600000000413
irisik	45591.95999999905	46946.0399999814	1354.079999982337
oniq	8425.40999999983	9841.65	1416.240000000162
levrana	2243.55999999995	3664.100000000076	1420.540000000081
roubloff	3491.3600000000015	4913.770000000003	1422.410000000017
smart	4457.25999999993	5902.1400000004	1444.880000000474
shik	3341.200000000016	4839.719999999975	1498.51999999996
domix	10472.0499999997	12009.16999999975	1537.120000000044
artex	2730.64000000001	4327.25000000001	1596.610000000001
beautix	10493.94999999984	12222.9499999995	1728.999999999654
milv	3904.93999999977	5642.01000000093	1737.070000001161
masura	31266.0799999918	33058.46999999813	1792.3899999998517
f.o.x	6624.22999999992	8577.27999999988	1953.049999999956
kapous	11927.16000000007	14093.07999999985	2165.919999999782
concept	11032.13999999921	13380.40000000005	2348.2600000001294
estel	21756.750000000015	24142.670000000373	2385.9200000003584
kaypro	881.339999999999	3268.69999999998	2387.35999999998
benovy	409.62	3259.9699999995	2850.34999999995
italwax	21940.23999999583	24799.3699999992	2859.1300000003357
yoko	8756.90999999994	11707.880000000023	2950.9700000000284
haruyama	9390.68999999935	12352.909999999805	2962.21999999987
marathon	7280.750000000001	10273.10000000001	2992.3500000000085
lovely	8704.37999999998	11939.06000000054	3234.680000000075
bpw.style	11572.150000000322	14837.440000001963	3265.2900000016416
staleks	8519.730000000018	11875.61000000059	3355.880000000041
freedecor	3421.780000000025	7671.80000000241	4250.02000000239
runail	71539.28000000017	76758.65999999538	5219.379999995205
polarus	6013.720000000075	11371.93000000001	5358.210000000002
cosmoprofi	8322.81000000007	14536.990000000136	6214.180000000129
jessnail	26287.8400000002	33345.22999999956	7057.389999999359
strong	29196.63000000005	38671.26999999975	9474.63999999997
ingarden	23161.39000000036	33566.20999999908	10404.819999999047
lianail	5892.83999999999	16394.240000000398	10501.400000000409
uno	35302.03000000007	51039.7499999992	15737.71999999915
grattol	35445.54000000023	71472.71000000041	36027.17000000019
	474679.06000001216	619509.2400000392	144830.18000002706

Time taken: 39.64 seconds, Fetched: 161 row(s)

## Insights:

- Total of 161 brands had an increase in the selling from October to November.
- 'Grattol' brand has the highest total increment i.e., 36,027/- and 'Ovale' seems to have least increment of 0.56/- from October to November.
- Among all these brands list, 'Runail' which was the best brand in terms of selling in October and November combined is also in the top 10 brands with high increment for October (71539.28/-) to November (76758.61/-) i.e., increment of total 5219.38/-.
- This implies that 'Runail' is the best and popular brand among all other brands within people.

**8.Your Company wants to reward the top 10 users in its website with a Golden Customer plan.  
Write a query to generate a list of top 10 users who spent the most.**

**Code:**

```
SELECT user_id, SUM(price) as Total_Expense  
FROM ecom_data_part  
WHERE event_type='purchase'  
GROUP BY user_id  
ORDER BY Total_Expense DESC  
LIMIT 10;
```

```
hive> SELECT user_id, SUM(price) as Total_Expense  
> FROM ecom_data_part  
> WHERE event_type='purchase'  
> GROUP BY user_id  
> ORDER BY Total_Expense DESC  
> LIMIT 10;  
Query ID = hadoop_20210404131249_41788d71-42b8-49f2-b6a1-8fa5c3b43544  
Total jobs = 1  
Launching Job 1 out of 1  
Status: Running (Executing on YARN cluster with App id application_1617537007743_0007)  
-----  
 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  
Reducer 3 ..... container  SUCCEEDED    1        1        0        0        0        0  
-----  
VERTICES: 03/03  [=====>>] 100%  ELAPSED TIME: 17.63 s  
-----  
OK  
user_id total_expense  
557790271      2715.8699999999963  
150318419      1645.969999999998  
562167663      1352.8500000000004  
531900924      1329.45  
557850743      1295.4799999999998  
522130011      1185.3900000000012  
561592095      1109.7000000000001  
431950134      1097.5899999999997  
566576008      1056.3600000000001  
521347209      1040.9099999999999  
Time taken: 18.433 seconds, Fetched: 10 row(s)  
hive> █
```

**Insights:**

- Here is the list of the top 10 users or buyers who have spent the most and could be rewarded with a Golden Customer plan to attract more people in the coming future.

- **Cleaning Up**

1. Drop the database ECOMDB

```
hive> DROP DATABASE IF EXISTS ECOMDB;
OK
Time taken: 0.218 seconds
hive> █
```

2. Once the analysis is done, we can terminate the cluster by changing the Termination protection from **ON** to **OFF** and then click on the terminate button.

The first screenshot shows the initial state where 'Termination protection' is set to 'On'. The second screenshot shows the 'Change' link being clicked, which leads to the third screenshot where the protection is now set to 'Off'.

**Screenshot 1 (Initial State):**

ID: j-1MVQZ1L7BE1TP	Creation date: 2021-04-04 17:12 (UTC+5:30)	Elapsed time: 3 hours, 41 minutes
After last step completes: Cluster waits	Termination protection: On <a href="#">Change</a>	Tags: -- <a href="#">View All / Edit</a>
Master public DNS: ec2-3-235-181-210.compute-1.amazonaws.com <a href="#">Connect to the Master Node Using SSH</a>		

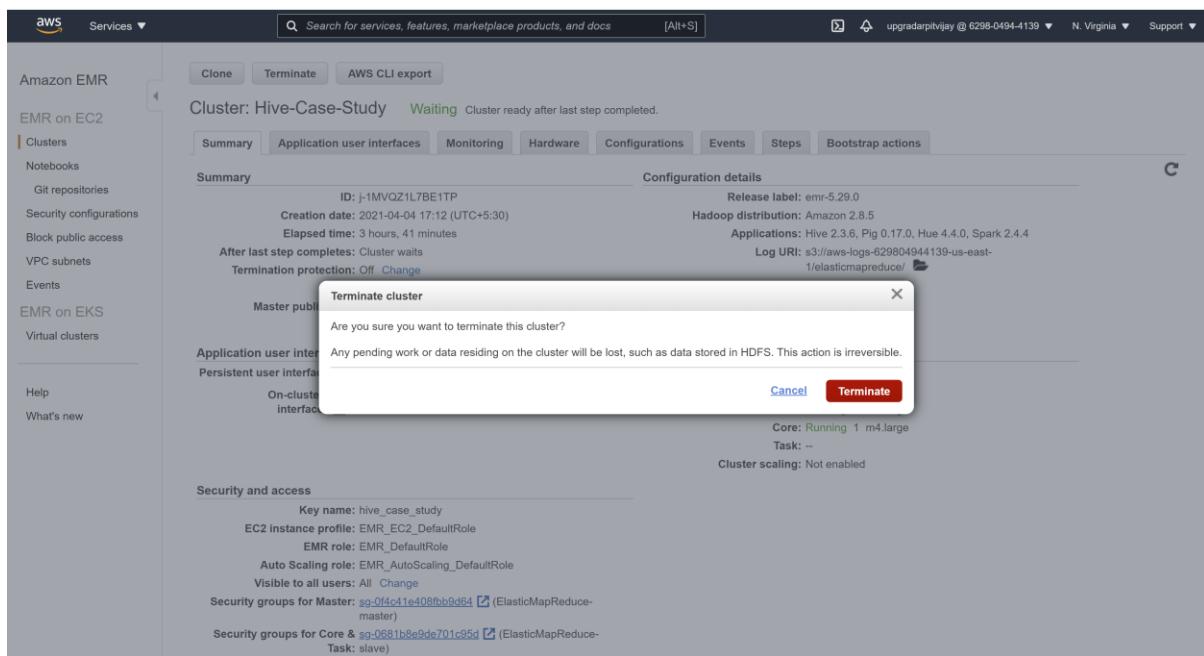
**Screenshot 2 (Change State):**

ID: j-1MVQZ1L7BE1TP	Creation date: 2021-04-04 17:12 (UTC+5:30)	Elapsed time: 3 hours, 41 minutes
After last step completes: Cluster waits	Termination protection: Off <a href="#">Change</a>	Tags: -- <a href="#">View All / Edit</a>
Master public DNS: ec2-3-235-181-210.compute-1.amazonaws.com <a href="#">Connect to the Master Node Using SSH</a>		

**Screenshot 3 (Final State):**

ID: j-1MVQZ1L7BE1TP	Creation date: 2021-04-04 17:12 (UTC+5:30)	Elapsed time: 3 hours, 26 minutes
After last step completes: Cluster waits	Termination protection: <input type="radio"/> On <input checked="" type="radio"/> Off	Tags: -- <a href="#">View All / Edit</a>
Master public DNS: ec2-3-235-181-210.compute-1.amazonaws.com <a href="#">Connect to the Master Node Using SSH</a>		

3. Click on Terminate.



4. Once we click on the terminate button the status of the cluster changes to terminating and then the cluster is terminated.

## SUMMARY

For this case study we have worked with public clickstream dataset of a cosmetics store.

Steps taken are:

- 1.Copying the data set into the HDFS from S3 Bucket.
- 2.Creating the database and launching Hive queries on your EMR cluster.  
[we made necessary optimizations, such as selecting the appropriate table format and using partitioned/bucketed tables]  
[use CSVSerde with the default properties value for loading the dataset into a Hive table]
- 3.Cleaning up.

**Overall, the increase in sales happened from October to November.**

