

HIVE CASE STUDY

Submitted by : Abhas Ramteke, Renu Raj
Batch : DSC-42

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. Needless to say, the role of big data analysts is among the most sought-after job profiles of this decade. Therefore, as part of this assignment, we will be challenging you, as a big data analyst, to extract data and gather insights from a real-life data set of an e-commerce company.

OBJECTIVE :

The goal is to extract the data and gather insights from a real-life data set of an e-commerce company.

DATA :

This project makes use of a public clickstream dataset from a cosmetics business. The clickstream data contains all of the logs of how a user navigated the ecommerce website. It also includes information such as client time spent on each page, amount of clicks made, things added to the cart, customer id, and so on.

The data is available from the link provided:

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

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



OVERVIEW OF STEPS :

- Copying the data set into the HDFS:
 - Launch an EMR cluster that utilizes the Hive services.
 - Move the data from the S3 bucket into the HDFS.
- Creating the database and launching Hive queries on your EMR cluster:
 - Create the structure of your database,
 - Use optimized techniques to run your queries as efficiently as possible
 - Show the improvement of the performance after using optimization on any single query.
 - Run Hive queries to answer the questions given below.
- Cleaning up
 - Drop your database, and
 - Terminate your cluster

1. Key-pair creation and generation

1.1 Click on Create Key pair

The screenshot shows the AWS EC2 Key Pairs page. On the left, there's a sidebar with options like EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances, Instances (New), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances (New), Dedicated Hosts, and Scheduled Instances. The main area displays a table of key pairs:

Name	Type	Created	Fingerprint
demokeypairtesting	rsa	2022/11/27 15:45 GMT+5:30	75:9d:7b:3a:5d:88:6d:c1:db:b2:54:b0:4a...
vokey	rsa	2022/11/15 13:31 GMT+5:30	2f:59:c2:8d:13:63:08:e4:3a:2b:d4:0c:27...

1.2 Providing name of Key & format as .ppk further for use with putty.

The screenshot shows the 'Create key pair' wizard. Step 1: Set key pair information. The 'Name' field is filled with 'demokeypairtesting'. Below it, a note says: 'The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.' The 'Key pair type' section has 'RSA' selected. Under 'Private key file format', 'ppk' is selected, with a note: 'For use with PuTTY'. The 'Tags - optional' section is empty. At the bottom, there's an 'Add new tag' button and a note: 'You can add up to 50 more tags.' At the very bottom are 'Cancel' and 'Create key pair' buttons.

Successfully Key Pair created.

Name given to key-pair as shown below as "demokeypairtesting".

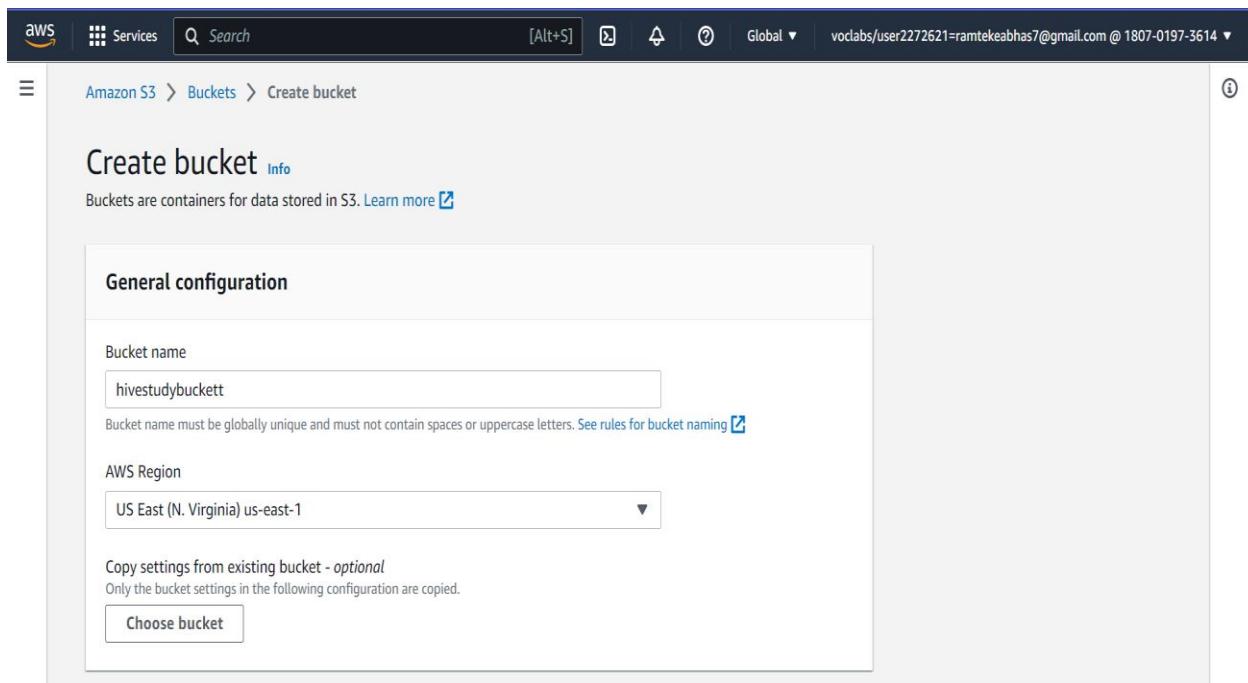
The screenshot shows the AWS EC2 Key Pairs page. The left sidebar includes links for EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (with sub-links for Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, and Scheduled Instances). The main content area displays a table titled "Key pairs (2) Info" with columns for Name, Type, Created, and Fingerprint. Two rows are listed: "demokeypairtesting" (rsa, 2022/11/27 15:45 GMT+5:30, 75:9d:7b:3a:5d:88:6d:c1:db:b2:54:b0:4a...) and "vockey" (rsa, 2022/11/15 13:31 GMT+5:30, 2f:59:c2:8d:13:63:08:e4:3a:2b:d4:0c:27...). An orange "Create key pair" button is at the top right. A black arrow points to the "demokeypairtesting" row.

2. Create S3 Bucket to store data files :

2.1 In search bar as we select s3 and clicked on the same, the below screen comes up in order to create bucket.

The screenshot shows the Amazon S3 Buckets page. The left sidebar includes links for Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, Access analyzer for S3, Block Public Access settings for this account, Storage Lens (with sub-links for Dashboards and AWS Organizations settings), and a note about AWS Lambda function triggers. The main content area displays a table titled "Buckets (3) Info" with columns for Name, AWS Region, Access, and Creation date. Three rows are listed: "abhas0demo" (US East (N. Virginia) us-east-1, Objects can be public, November 15, 2022, 16:28:42 ()), "aws-logs-180701973614-us-east-1" (US East (N. Virginia) us-east-1, Objects can be public, November 27, 2022, 15:58:49 ()), and "hivestudybuckett" (US East (N. Virginia) us-east-1, Objects can be public, December 2, 2022, 12:57:36 (U)). An orange "Create bucket" button is at the top right. A black arrow points to this button.

2.2 A unique name was given to create bucket which was not used earlier as instructed. Clicked on block to uncheck “block all public access”.



2.3 New Bucket created successfully with name “hivestudybuckett”.

The screenshot shows the 'Buckets' list page in the AWS S3 console. The left sidebar shows 'Buckets' under 'Amazon S3'. The main area displays a table of buckets:

Name	AWS Region	Access	Creation date
abhas0demo	US East (N. Virginia) us-east-1	Objects can be public	November 15, 2022, 16:28:42 (U)
aws-logs-180701973614-us-east-1	US East (N. Virginia) us-east-1	Objects can be public	November 27, 2022, 15:58:49 (U)
hivestudybuckett	US East (N. Virginia) us-east-1	Objects can be public	December 2, 2022, 12:57:36 (U)

An arrow points to the 'hivestudybuckett' row in the table.

2.4 Firstly, a folder “hivedataset2019” is created under bucket & then uploaded both dataset in the folder.

The screenshot shows the AWS S3 console interface. The left sidebar has a 'Buckets' section with various options like Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and Access analyzer for S3. Below that is a 'Storage Lens' section with Dashboards and AWS Organizations settings. The main content area shows a bucket named 'hivestudybuckett'. The 'Objects' tab is selected. A single object, 'hivedataset2019/' (a folder), is listed. Below the object list are buttons for Create folder, Upload (which is highlighted in orange), Copy S3 URI, Copy URL, Download, Open, Delete, and Actions. There's also a search bar labeled 'Find objects by prefix' and a pagination indicator showing page 1 of 1.

The screenshot shows the AWS S3 console interface, similar to the previous one but with a different bucket. The left sidebar is identical. The main content area shows a bucket named 'hivedataset2019/'. The 'Objects' tab is selected. Two objects are listed: '2019-Nov.csv' and '2019-Oct.csv', both of which are CSV files. They were last modified on December 2, 2022, at 13:16:38 (UTC+05:30). Their sizes are 520.6 MB and 460.2 MB respectively, and they are stored in the Standard storage class. Above the object list are buttons for Create folder, Upload (highlighted in orange), Copy S3 URI, Copy URL, Download, Open, Delete, and Actions. There's also a search bar labeled 'Find objects by prefix' and a pagination indicator showing page 1 of 1.

3. EMR CLUSTER CREATION:

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

The screenshot shows the Amazon EMR console interface. On the left, there's a sidebar with navigation links for EMR Studio, EMR Serverless (New), EMR on EC2 (Clusters, Notebooks, Git repositories, Security configurations, Block public access, VPC subnets, Events), EMR on EKS, Virtual clusters, and an ellipsis. The main area has a header with 'Create cluster' (highlighted in blue), 'View details', 'Clone', and 'Terminate'. A search bar says 'Filter: All clusters' and shows '22 clusters (all loaded)'. The table lists the following clusters:

	Name	ID	Status	Creation time (UTC+5:30)	Elapsed time
<input type="checkbox"/>	My_cluster	j-7VPOLGS5A4AT	Terminated User request	2022-12-05 01:46 (UTC+5:30)	11 hours, 6 minutes
<input type="checkbox"/>	Hive_Cluster	j-DGLIVELPIWUF	Terminated User request	2022-12-05 01:42 (UTC+5:30)	11 hours, 9 minutes
<input type="checkbox"/>	Hive_Cluster	j-1DWFZ7D4WJ9Y6	Terminated Auto-terminate	2022-12-05 01:39 (UTC+5:30)	1 hour, 16 minutes
<input checked="" type="checkbox"/>	Hive_Cluster	j-3U593JEM528X3	Terminated with errors Instance failure	2022-12-04 23:06 (UTC+5:30)	2 hours, 20 minutes
<input checked="" type="checkbox"/>	Hive_Cluster	j-2U6PUYKD3G4I9	Terminated with errors Validation error	2022-12-04 23:03 (UTC+5:30)	41 seconds
<input type="checkbox"/>	Hive_Cluster	j-3FT1YO59GRT3U	Terminated Auto-terminate	2022-12-04 21:27 (UTC+5:30)	1 hour, 20 minutes
<input type="checkbox"/>	Hive_Cluster	j-3VZBMJUNQUIJF	Terminated Auto-terminate	2022-12-02 19:18 (UTC+5:30)	2 hours, 5 minutes
<input checked="" type="checkbox"/>	Hive_Cluster	j-OOCOW3FQFP4V	Terminated with errors	2022-12-02 19:14 (UTC+5:30)	33 seconds

3.2 Then click on 'Go to advanced options'.

The screenshot shows the 'Create Cluster - Quick Options' page. At the top, it says 'Create Cluster - Quick Options' and 'Go to advanced options'. Below that is a 'General Configuration' section with the following fields:

- Cluster name: my_hive_cluster
- Logging i
- S3 folder: s3://aws-logs-180701973614-us-east-1/elasticmap copy
- Launch mode: Cluster i Step execution i

Below that is a 'Software configuration' section with the following settings:

- Release: emr-5.29.0
- Applications:
 - Core Hadoop: Hadoop 2.8.5 with Ganglia 3.7.2, Hive 2.3.6, Hue 4.4.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
 - HBase: HBase 1.4.10 with Ganglia 3.7.2, Hadoop 2.8.5, Hive 2.3.6, Hue 4.4.0, Phoenix 4.14.3, and ZooKeeper 3.4.14
 - Presto: Presto 0.227 with Hadoop 2.8.5 HDFS and Hive 2.3.6 Metastore
 - Spark: Spark 3.4.1 on Hadoop 2.8.5 YARN with

At the bottom, there are links for Feedback, Unified Settings, Copyright notice (© 2022, Amazon Web Services, Inc. or its affiliates.), Privacy, Terms, and Cookie preferences.

3.3 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.

3.4 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 master and 2 for core node.

Node type	Instance type	Instance count	Purchasing option
Master Master - 1	m4.large 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 64 GiB	1 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price
Core Core - 2	m4.large 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 64 GiB	2 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price

3.5 Here we have given the name of the cluster as “my_hive_cluster”.

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

Logging [?](#)

S3 folder [File browser](#)

Debugging [?](#)

Termination protection [?](#)

Tags [?](#)

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

Additional Options

3.6 We have selected the created EC2 key pair named “demokeypairtesting” and then click on the create cluster option.

Step 1: Software and Steps

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

Security Options

EC2 key pair [?](#)

Cluster visible to all IAM users in account [?](#)

Permissions [?](#)

Default Custom

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

EMR role [?](#) Use EMR_DefaultRole_V2 [?](#)

EC2 instance profile [?](#)

Auto Scaling role [?](#)

► Security Configuration

► EC2 security groups

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

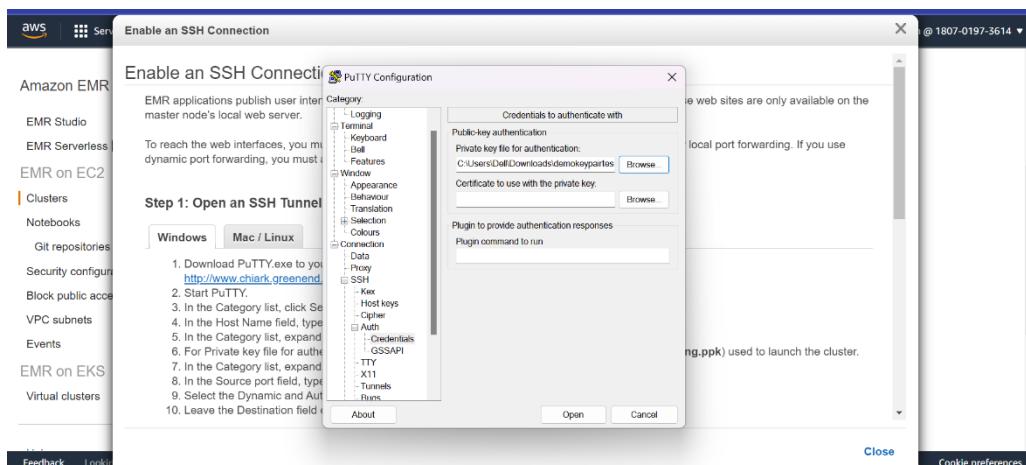
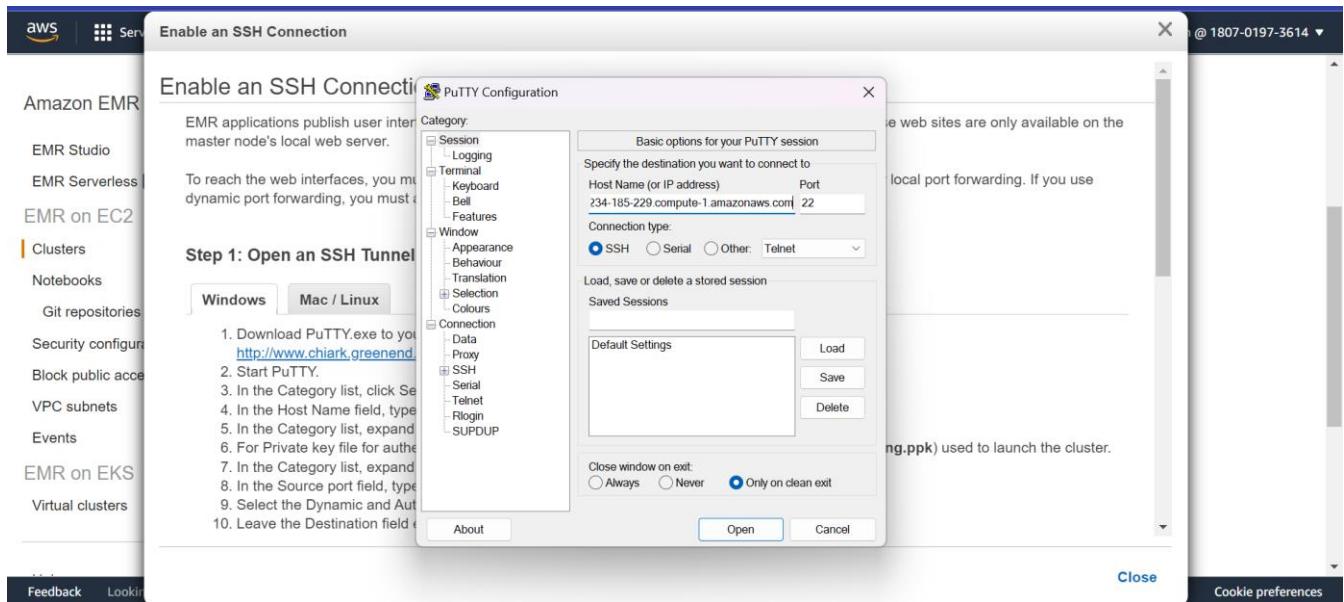
3.7 Then cluster has been created and will be in the “Starting” state and will change to ‘Waiting’ state after sometime.

The screenshot shows the AWS EMR Cluster Overview page. On the left, there's a sidebar with navigation links for Amazon EMR, EMR Studio, EMR Serverless (New), EMR on EC2 (selected), Clusters, Notebooks, Git repositories, Security configurations, Block public access, VPC subnets, and Events. Under EMR on EC2, there are also links for EMR on EKS and Virtual clusters. The main content area shows a cluster named "my_hive_cluster" in the "Starting" state. A red warning box at the top right says "Auto-termination is not available for this account when using this release of EMR." Below the cluster name, there are tabs for Summary, Application user interfaces, Monitoring, Hardware, Configurations, Events, Steps, and Bootstrap actions. The Summary tab is selected. It displays the cluster ID (j-3OJSTHLDPEL), creation date (2022-12-05 22:15 UTC+5:30), elapsed time (9 seconds), and status message "After last step completes: Cluster waits". It also shows termination protection set to "On" and a master public DNS entry. The Configuration details section lists the 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), and log URI (s3://aws-logs-180701973614-us-east-1/elasticmapreduce/). At the bottom, there are links for Feedback, Unified Settings, and various AWS links like Privacy, Terms, and Cookie preferences.

3.8 Now cluster is started & connected to Master Node using SSH connection.

The screenshot shows the AWS EMR Cluster Overview page for the same cluster, now in the "Running" state. The main content area shows the cluster details again, but the status message now says "After last step completes: Cluster is running". The Configuration details section remains the same. The Application user interfaces section shows a persistent user interface entry with a link to "Enable an SSH Connection". The Network and hardware section shows the availability zone (us-east-1a), subnet ID (subnet-02436d2c95aaa5fe1), and master and core instance types (m4.xlarge). There are also links for "Edit" and "View All" throughout the configuration sections.

3.9 Once cluster in running state we have to click on Master public DNS(Connect to master stroke using SSH). 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 → credentials. Selected browse option for selecting key pair.



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

```
hadoop@ip-172-31-55-24:~  
Using username "hadoop".  
Authenticating with public key "demokeypairtesting"  
Last login: Sun Dec 4 16:04:01 2022  
  
_ _|_|_ _|_) Amazon Linux 2 AMI  
_ _\_\_||_|_|  
  
https://aws.amazon.com/amazon-linux-2/  
22 package(s) needed for security, out of 32 available  
Run "sudo yum update" to apply all updates.  
  
EEEEEEEEEEEEEEEEEE MMMMMMM M:-----M M:-----M R:-----R  
E:-----:-----:-----:-----:-----:-----:-----:-----:R  
EE:-----:-----:-----:-----:-----:-----:-----:RRRRRRRR:-----R  
E:-----:-----:-----:-----:-----:-----:-----:R:-----R  
E:-----:-----:-----:-----:-----:-----:-----:R:-----:R  
E:-----:-----:-----:-----:-----:-----:-----:R:-----:RR  
E:-----:-----:-----:-----:-----:-----:-----:R:-----:RRRRRRRR:-----R  
E:-----:-----:-----:-----:-----:-----:-----:R:-----:R  
E:-----:-----:-----:-----:-----:-----:-----:R:-----:R  
EE:-----:-----:-----:-----:-----:-----:-----:R:-----:R  
E:-----:-----:-----:-----:-----:-----:-----:R:-----:R  
EEEEEEEEEEEEEEEEEE MMMMMMM M:-----M M:-----M R:-----R  
E:-----:-----:-----:-----:-----:-----:-----:RRRRRRRR:-----R  
E:-----:-----:-----:-----:-----:-----:-----:R:-----:R  
EEEEEEEEEEEEEEEEEE MMMMMMM RRRRRRRR RRRRRR
```

4. Copying the data into HDFS

4.1 To access the public s3bucket.

Command : aws s3 ls hivestudybuckett

Output : PRE hivedataset2019/ (this is folder where both the dataset is uploaded).

```
[hadoop@ip-172-31-55-24 ~]$ aws s3 ls hivestudybuckett
                                PRE hivedataset2019/
```

4.2 Command to check already present directories in HDFS.

Command: hadoop fs -ls /

```
[hadoop@ip-172-31-55-24 ~]$ hadoop fs -ls /
Found 4 items
drwxr-xr-x  - hdfs hdfsadmingroup          0 2022-12-04 16:04 /apps
drwxrwxrwt  - hdfs hdfsadmingroup          0 2022-12-04 16:07 /tmp
drwxr-xr-x  - hdfs hdfsadmingroup          0 2022-12-04 16:04 /user
drwxr-xr-x  - hdfs hdfsadmingroup          0 2022-12-04 16:04 /var
```

4.3 Checking the available directory.

Command: hadoop fs -ls /user/

```
[hadoop@ip-172-31-55-24 ~]$ hadoop fs -mkdir /user/hive/hive_case_study_abhas
[hadoop@ip-172-31-55-24 ~]$ hadoop fs -ls /user/
Found 6 items
drwxrwxrwx  - hadoop hdfsadmingroup          0 2022-12-04 16:04 /user/hadoop
drwxr-xr-x  - mapred mapred                 0 2022-12-04 16:04 /user/history
drwxrwxrwx  - hdfs hdfsadmingroup          0 2022-12-04 16:40 /user/hive
drwxrwxrwx  - hue  hue                      0 2022-12-04 16:04 /user/hue
drwxrwxrwx  - oozie oozie                  0 2022-12-04 16:06 /user/oozie
drwxrwxrwx  - root  hdfsadmingroup          0 2022-12-04 16:04 /user/root
[hadoop@ip-172-31-55-24 ~]$ █
```

4.4 Creating a directory in HDFS.

Command: hadoop fs -mkdir /user/hive/hive_case_study_abhas

4.5 Load the s3 public data set to created directory “hive_case_study_abhas” in hadoop .

Command: hadoop distcp s3://hivestudybuckett/hivedataset2019/2019-Oct.csv
/user/new_hive/2019-Oct.csv

```
[hadoop@ip-172-31-53-223 ~]$ hadoop distcp s3://hivestudybuckett/hivedataset2019/2019-Oct.csv /user/new_hive/2019-Oct.csv
22/12/04 18:00:45 INFO tools.OptionsParser: parseChunkSize: blocksperchunk false
22/12/04 18:00:47 INFO tools.DistCp: Input Options: DistCpOptions{atomicCommit=false, syncFolder=false, deleteMissing=false, ignoreFailures=false, overwrite=false, append=false, useDiff=false, useRdiff=false, fromSnapshot=null, toSnapshot=null, 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://hivestudybuckett/hivedataset2019/2019-Oct.csv], targetPath=/user/new_hive/2019-Oct.csv, targetPathExists=false, filtersFile='null', blocksPerChunk=0, copyBufferSize=8192, verboseLog=false}
22/12/04 18:00:47 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-53-223.ec2.internal/172.31.53.223:8032
22/12/04 18:00:47 INFO client.AHSProxy: Connecting to Application History server at ip-172-31-53-223.ec2.internal/172.31.53.223:10200
22/12/04 18:00:54 INFO tools.SimpleCopyListing: Paths (files+dirs) cnt = 1; dirCnt = 0
22/12/04 18:00:54 INFO tools.SimpleCopyListing: Build file listing completed.
22/12/04 18:00:54 INFO Configuration.deprecation: io.sort.mb is deprecated. Instead, use mapreduce.ask.io.sort.mb
22/12/04 18:00:54 INFO Configuration.deprecation: io.sort.factor is deprecated. Instead, use mapreduce.task.io.sort.factor
22/12/04 18:00:54 INFO tools.DistCp: Number of paths in the copy list: 1
22/12/04 18:00:54 INFO tools.DistCp: Number of paths in the copy list: 1
22/12/04 18:00:54 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-53-223.ec2.internal/172.31.53.223:8032
22/12/04 18:00:54 INFO client.AHSProxy: Connecting to Application History server at ip-172-31-53-223.ec2.internal/172.31.53.223:10200
22/12/04 18:00:55 INFO mapreduce.JobSubmitter: number of splits:1
22/12/04 18:00:55 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1670175863837_0001
22/12/04 18:00:55 INFO conf.Configuration: resource-types.xml not found
22/12/04 18:00:55 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/12/04 18:00:55 INFO resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE
22/12/04 18:00:55 INFO resource.ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE
22/12/04 18:00:56 INFO impl.YarnClientImpl: Submitted application application_1670175863837_0001
22/12/04 18:00:56 INFO mapreduce.Job: The url to track the job: http://ip-172-31-53-223.ec2.internal:2088/proxy/application_1670175863837_0001/
22/12/04 18:00:56 INFO tools.DistCp: DistCp job-id: job_1670175863837_0001
22/12/04 18:00:56 INFO mapreduce.Job: Running job: job_1670175863837_0001
22/12/04 18:01:07 INFO mapreduce.Job: Job job_1670175863837_0001 running in uber mode : false
22/12/04 18:01:07 INFO mapreduce.Job: map 0% reduce 0%
22/12/04 18:01:25 INFO mapreduce.Job: map 100% reduce 0%
22/12/04 18:01:29 INFO mapreduce.Job: Job job_1670175863837_0001 completed successfully
22/12/04 18:01:29 INFO mapreduce.Job: Counters: 38
```

Bytes Read=250
File Output Format Counters
Bytes Written=0
DistCp Counters
Bytes Copied=482542278
Bytes Expected=482542278
Files Copied=1

Command: hadoop distcp s3://hivestudybuckett/hivedataset2019/2019-Nov.csv
 /user/new_hive/2019-Nov.csv

```
[hadoop@ip-172-31-53-223 ~]$ hadoop distcp s3://hivestudybuckett/hivedataset2019/2019-Nov.csv /user/new_hive/2019-Nov.csv
22/12/04 18:04:09 INFO tools.OptionsParser: parseChunkSize: blocksperchunk false
22/12/04 18:04:11 INFO tools.DistCp: Input Options: DistCpOptions{atomicCommit=false, syncFolder=false, deleteMissing=false, ignoreFailures=false, overwrite=false, append=false, useDiff=false, useRdiff=false, fromSnapshot=null, toSnapshot=null, 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://hivestudybuckett/hivedataset2019/2019-Nov.csv], targetPath=/user/new_hive/2019-Nov.csv, targetPathExists=false, filtersFile='null', blocksPerChunk=0, copyBufferSize=8192, verboseLog=false}
22/12/04 18:04:11 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-53-223.ec2.internal/172.31.53.223:8032
22/12/04 18:04:11 INFO client.AHSProxy: Connecting to Application History server at ip-172-31-53-223.ec2.internal/172.31.53.223:10200
22/12/04 18:04:16 INFO tools.SimpleCopyListing: Paths (files+dirs) cnt = 1; dirCnt = 0
22/12/04 18:04:16 INFO tools.SimpleCopyListing: Build file listing completed.
22/12/04 18:04:16 INFO Configuration.deprecation: io.sort.mb is deprecated. Instead, use mapreduce.ask.io.sort.mb
22/12/04 18:04:16 INFO Configuration.deprecation: io.sort.factor is deprecated. Instead, use mapreduce.task.io.sort.factor
22/12/04 18:04:16 INFO tools.DistCp: Number of paths in the copy list: 1
22/12/04 18:04:16 INFO tools.DistCp: Number of paths in the copy list: 1
```

```

ce.task.io.sort.factor
22/12/04 18:04:16 INFO tools.DistCp: Number of paths in the copy list: 1
22/12/04 18:04:16 INFO tools.DistCp: Number of paths in the copy list: 1
22/12/04 18:04:16 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-53-223.ec2.internal
1/172.31.53.223:8032
22/12/04 18:04:16 INFO client.AHSProxy: Connecting to Application History server at ip-172-31-53-223
.ec2.internal/172.31.53.223:10200
22/12/04 18:04:17 INFO mapreduce.JobSubmitter: number of splits:1
22/12/04 18:04:17 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1670175863837_0002
22/12/04 18:04:18 INFO conf.Configuration: resource-types.xml not found
22/12/04 18:04:18 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/12/04 18:04:18 INFO resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi,
type = COUNTABLE
22/12/04 18:04:18 INFO resource.ResourceUtils: Adding resource type - name = vcores, units = , type
= COUNTABLE
22/12/04 18:04:18 INFO impl.YarnClientImpl: Submitted application application_1670175863837_0002
22/12/04 18:04:18 INFO mapreduce.Job: The url to track the job: http://ip-172-31-53-223.ec2.internal
:20888/proxy/application_1670175863837_0002/
22/12/04 18:04:18 INFO tools.DistCp: DistCp job-id: job_1670175863837_0002
22/12/04 18:04:18 INFO mapreduce.Job: Running job: job_1670175863837_0002
22/12/04 18:04:26 INFO mapreduce.Job: Job job_1670175863837_0002 running in uber mode : false
22/12/04 18:04:26 INFO mapreduce.Job: map 0% reduce 0%
22/12/04 18:04:44 INFO mapreduce.Job: map 100% reduce 0%
22/12/04 18:04:48 INFO mapreduce.Job: Job job_1670175863837_0002 completed successfully
22/12/04 18:04:48 INFO mapreduce.Job: Counters: 38
22/12/04 18:04:48 INFO mapreduce.Job: Counters: 38
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=224554
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=385
    HDFS: Number of bytes written=545839412
    HDFS: Number of read operations=11
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=5
    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)=646624
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=20207
    Total vcore-milliseconds taken by all map tasks=20207
    Total megabyte-milliseconds taken by all map tasks=20691968
  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)=359
    CPU time spent (ms)=21250
    Physical memory (bytes) snapshot=618848256

```

```

Physical memory (bytes) snapshot=618848256
Virtual memory (bytes) snapshot=3334119424
Total committed heap usage (bytes)=500695040
File Input Format Counters
  Bytes Read=250
File Output Format Counters
  Bytes Written=0
DistCp Counters
  Bytes Copied=545839412
  Bytes Expected=545839412
  Files Copied=1

```

5. Hive query > hive

5.1 create database ABDB;

5.2 use ABDB;

5.3 Create table hive_table;

5.4 Once base table is created we need to go for optimizing the table for quick query result.

Optimised table name is hive_table_part.

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

5.5 Created partitioning table named hive_table_part.

```
hadoop@ip-172-31-53-223:~  
at org.apache.hadoop.hive.cli.CliDriver.run(CliDriver.java:759)  
at org.apache.hadoop.hive.cli.CliDriver.main(CliDriver.java:666)  
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)  
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  
at java.lang.reflect.Method.invoke(Method.java:498)  
at org.apache.hadoop.util.RunJar.run(RunJar.java:244)  
at org.apache.hadoop.util.RunJar.main(RunJar.java:158)  
FAILED: ParseException line 1:0 cannot find identifier near 'hive'">' hive'  
hive> CREATE TABLE IF NOT EXISTS hive_table (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/new_hive/' tblproperties('skip.header.line.count'=1')  
OK  
Time taken: 0.774 seconds  
hive> select * from hive_table limit 5;  
OK  
2019-11-01 00:00:02 UTC view 5802432 1487580009286598681 0.32 562076640 0  
9fafdd6c-6e99-46b1-834f-33527f4de241  
2019-11-01 00:00:09 UTC cart 5844397 1487580006317032337 2.38 553329724 2  
067216c-31b5-455d-a1cc-af0575a34ffb  
2019-11-01 00:00:10 UTC view 5837166 1783999064103190764 pnb 22.22 556138645 5  
7ed222e-a54a-4907-9944-5a875c2d7ff4f  
2019-11-01 00:00:11 UTC cart 5876812 14875800010100293687 jessnail 3.16 5645  
06666 186c1951-8052-4b37-adce-dd9644bd1d5f7  
2019-11-01 00:00:24 UTC remove_from_cart 5826182 1487580007483048900 3.33  
553329724 2067216c-31b5-455d-a1cc-af0575a34ffb  
Time taken: 3.229 seconds, Fetched: 5 row(s)  
hive> SET hive.exec.dynamic.partition=true;  
hive> SET hive.exec.dynamic.partition.mode=nonstrict;  
hive> SET hive.enforce.bucketing =true;  
hive> CREATE TABLE IF NOT EXISTS hive_table (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/new_hive/' tblproperties('skip.header.line.count'=1');  
OK  
Time taken: 0.146 seconds  
hive> Insert into table hive_table_part partition (event_type) select event_time , product_id , category_id , category_code , brand , price , user_id , user_session , event_type from ecom_data ;  
FAILED: SemanticException [Error 10001]: Line 1:177 Table not found 'ecom_data'  
hive> Insert into table hive_table_part partition (event_type) select event_time , product_id , category_id , category_code , brand , price , user_id , user_session , event_type from hive_table ;  
Query ID = hadoop_20221204182529_ib8a3c5d-1ce5-4380-9c8e-6f3ed1f34ea3  
[REDACTED]
```

5.5 Inserting records into hive_table_part from hive_table.

```
hadoop@ip-172-31-53-223:~  
2019-11-01 00:00:10 UTC view 5837166 1783999064103190764 pnb 22.22 556138645 5  
7ed222e-a54a-4907-9944-5a875c2d7ff4f  
2019-11-01 00:00:11 UTC cart 5876812 148758000100293687 jessnail 3.16 5645  
06666 186c1951-8052-4b37-adce-dd9644bd1d5f7  
2019-11-01 00:00:24 UTC remove_from_cart 5826182 1487580007483048900 3.33  
553329724 2067216c-31b5-455d-a1cc-af0575a34ffb  
Time taken: 3.229 seconds, Fetched: 5 row(s)  
hive> SET hive.exec.dynamic.partition=true;  
hive> SET hive.exec.dynamic.partition.mode=nonstrict;  
hive> SET hive.enforce.bucketing =true;  
hive> CREATE TABLE IF NOT EXISTS hive_table (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/new_hive/' tblproperties('skip.header.line.count'=1);  
OK  
Time taken: 0.146 seconds  
hive> Insert into table hive_table_part partition (event_type) select event_time , product_id , category_id , category_code , brand , price , user_id , user_session , event_type from ecom_data ;  
FAILED: SemanticException [Error 10001]: Line 1:177 Table not found 'ecom_data'  
hive> Insert into table hive_table_part partition (event_type) select event_time , product_id , category_id , category_code , brand , price , user_id , user_session , event_type from hive_table ;  
Query ID = hadoop_20221204182529_ib8a3c5d-1ce5-4380-9c8e-6f3ed1f34ea3  
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_1670175863837_0004)  
-----  
 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: 0/2 [=====>>>] 100% ELAPSED TIME: 174.14 s  
-----  
Loading data to table default.hive_table_part partition (event_type=null)  
Loaded : 4/4 partitions.  
Time taken to load dynamic partitions: 1.019 seconds  
Time taken for adding to write entity : 0.003 seconds  
OK  
Time taken: 185.285 seconds  
[REDACTED]
```

5.6 Print the column headers for the queries.

Command:

Set hive.cli.print.header=true;

```

hadoop@ip-172-31-53-223:~ 
hive> set hive.cli.print.header=true;
hive> select * from hive_table_part limit 5;
OK
hive_table_part.event_time      hive_table_part.product_id      hive_table_part.category_id      hive
hive_table_part.event_time      hive_table_part.brand      hive_table_part.price      hive_table_part.user
_id    hive_table_part.user_session  hive_table_part.event_type
2019-10-08 18:54:37 UTC 5749149 1487580007592100809      0.22  556330912  9e6a
e8ea-ff7c-47c9-80a4-59ade79f9578  cart
2019-10-08 18:54:38 UTC 5751423 1511892746070131099      uno   9.37  532169378  2786
2764-a3bf-4ef2-8e72-dff0b3ba8c18  cart
2019-10-08 18:54:38 UTC 5789896 1487580006132482952      runail 0.14  468486843  df0c
98b8-bd9b-4bb8-a3f1-379e477aca0  cart
2019-10-08 18:54:39 UTC 5746444 148758000575499573      4.44  437836184  f12c
d33c-ac3e-4d6e-9879-eed8d5624749  cart
2019-10-08 18:54:40 UTC 5563798 1487580005595612013      3.00  557513931  bcee
b5d3-aac0-4c4a-9639-fc16e5fb5c35  cart
Time taken: 0.276 seconds, Fetched: 5 row(s)
hive> Select sum(price) as total_revenue from hive_table where month(event_time)=10 and event_type = 'purchase';
Query ID = hadoop_20221204183201_la2f9879-b36b-420c-a7f1-fd68b59cf61d
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1670175863837_0004)

-----  

VERTICES      MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED
Map 1 ..... container  SUCCEEDED  5      5      0      0      0      0
Reducer 2 .... container  SUCCEEDED  1      1      0      0      0      0
-----  

VERTICES: 02/02  [=====>>] 100% ELAPSED TIME: 129.74 s
-----  

OK
total_revenue
1211538.4299997438
Time taken: 131.358 seconds, Fetched: 1 row(s)

```

Following are the screen shots of the query outputs:

Question 1:

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

Ans 1- Total revenue generated with the help of base table created as in below image.

Query - Select sum(price) as total_revenue from hive_table where month(event_time)=10 and event_type = 'purchase' ;

```

hadoop@ip-172-31-53-223:~ 
hive> Select sum(price) as total_revenue from hive_table where month(event_time)=10 and event_type = 'purchase';
Query ID = hadoop_20221204183201_la2f9879-b36b-420c-a7f1-fd68b59cf61d
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1670175863837_0004)

-----  

VERTICES      MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED
Map 1 ..... container  SUCCEEDED  5      5      0      0      0      0
Reducer 2 .... container  SUCCEEDED  1      1      0      0      0      0
-----  

VERTICES: 02/02  [=====>>] 100% ELAPSED TIME: 129.74 s
-----  

OK
total_revenue
1211538.4299997438
Time taken: 131.358 seconds, Fetched: 1 row(s)

```

Ans2 - Total revenue generated with the help of partition table created as in below image.

Select sum(price) as total_revenue from hive_table_part where month(event_time)=10 and event_type = 'purchase' ;

```

hive> select sum(price) as total_revenue from hive_table_part where month(event_time)=10 and event_type = 'purchase' ;
Query ID = hadoop_20221207190213_ddb111bc-144a-4fec-9e10-bf9dbe0dcda7
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1670438307348_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: 9.07 s
-----  

OK
total_revenue
1211525.179999663
Time taken: 9.958 seconds, Fetched: 1 row(s)
hive>

```

Inference:-

The total revenue made by the ECOM company in the month of October is 1211538/- & We attempted to obtain the result by using both the base table hive_table and the optimised table hive_table_part. We discovered that the Bucketed table improves query time by around 120 seconds when compared to the base table. Hence using hive_table_part partitioning table for further query for optimization.

Question 2)

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

Query 2:

```
select date_format(event_time, 'MM') as month, count(event_type) as Sum_of_purchases from  
hive_table_part where event_type='purchase' group by date_format(event_time, 'MM');
```

```
1211538.4299997438  
Time taken: 131.358 seconds, Fetched: 1 row(s)  
hive> SELECT date_format(event_time, 'MM') AS Months, COUNT(event_type) AS Sum_of_Purchases FROM hiv  
e_table_part WHERE event_type='purchase' GROUP BY date_format(event_time, 'MM');  
Query ID = hadoop_20221204183640_d447eb88-6bla-4015-8e8f-ec9edc60a7el  
Total jobs = 1  
Launching Job 1 out of 1  
Status: Running (Executing on YARN cluster with App id application_1670175863837_0004)  
  
-----  
 VERTICES  MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  
Map 1 ..... container  SUCCEEDED   2       2       0       0       0       0  
Reducer 2 ..... container  SUCCEEDED   2       2       0       0       0       0  
-----  
VERTICES: 02/02  [=====>>>] 100% ELAPSED TIME: 21.76 s  
  
OK  
months  sum_of_purchases  
10      245623  
11      322411  
Time taken: 22.832 seconds, Fetched: 2 row(s)  
hive> Display all 574 possibilities? (y or n)  
!  
$KEY$          !=          $ELEM$  
$KEY$          $VALUE$     $elem$  
$key$          $sum0       $value$  
%             &           ()
```

Inference:

More purchases are done in the month of November compared to October, with a difference of 76788/-

Que 3:

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

Query- 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 hive_table_part  
WHERE event_type= 'purchase') SELECT (Nov_purchase - Oct_purchase) as difference_revenue  
FROM rev_difference ;
```

```
hadoop@ip-172-31-53-223:~  
hive> WITH rev_difference AS (SELECT SUM(case when MONTH(event_time) = '10' then price else 0 end) A  
S Oct_purchase, SUM(case when MONTH(event_time) = '11' then price else 0 end) AS Nov_purchase FROM h  
ive_table_part WHERE event_type= 'purchase') SELECT (Nov_purchase - Oct_purchase) as difference_reve  
nue FROM rev_difference;  
Query ID = hadoop_20221204184954_93dd0541-fbf5-4a99-ba95-62a319cf34a8  
Total jobs = 1  
Launching Job 1 out of 1  
Status: Running (Executing on YARN cluster with App id application_1670175863837_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: 20.78 s  
  
OK  
difference_revenue  
319298.3600005086  
Time taken: 21.465 seconds, Fetched: 1 row(s)
```

Inference:

The difference in revenue between October and November is 319298/- . The month of November is more profitable.

Que 4:

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

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

```

hadoop@ip-172-31-53-223:~ 
hive> SELECT DISTINCT SPLIT(category_code,'\\.') [0]
> AS Category
> FROM hive_table_part
> WHERE SPLIT(category_code,'\\.') [0] <> '';
Query ID = hadoop_20221204185323_7ca9c521-e1cf-4547-98d9-544601c194f6
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1670175863837_0005)

-----  

 VERTICES      MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

Map 1 ..... container    SUCCEEDED   4       4       0       0       0       0  

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

-----  

VERTICES: 02/02  [=====>>>] 100% ELAPSED TIME: 69.16 s  

-----  

OK
category
accessories
apparel
appliances
furniture
sport
stationery
Time taken: 70.08 seconds, Fetched: 6 row(s)
hive> WITH Monthly_Revenue AS ( SELECT brand, SUM(CASE WHEN date_format(event_time, 'MM')=10 THEN pr

```

Furniture, appliances, accessories, apparel, sport and stationery are the different categories in the table.

Query 5:

Find the total number of products available under each category.

SELECT SPLIT(category_code,'\\.') [0] AS Category, COUNT(product_id) AS No_of_products FROM hive_table_part WHERE SPLIT(category_code,'\\.') [0] <> '' GROUP BY SPLIT(category_code,'\\.') [0] ORDER BY No_of_products DESC;

```

Time taken: 27.017 seconds, Fetched: 1 row(s)
hive> SELECT SPLIT(category_code,'\\.') [0] AS Category, COUNT(product_id) AS No_of_products FROM hive_table_part WHERE SPLIT(category_code,'\\.') [0] <> '' GROUP BY SPLIT(category_code,'\\.') [0] ORDER BY No_of_products DESC;
Query ID = hadoop_20221204194207_9c03eec3-9lee-4b6e-a00-e03755f0adcc
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1670175863837_0007)

-----  

 VERTICES      MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

Map 1 ..... container    SUCCEEDED   4       4       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: 71.53 s  

-----  

OK
appliances      61730
stationery      26720
furniture       23594
apparel         18232
accessories     12923
sport            1
Time taken: 72.422 seconds, Fetched: 6 row(s)
hive>

```

appliances	61730
stationery	26720
furniture	23594
apparel	18232
accessories	12923
sport	1

Inference: - Appliances category has the highest number of products 61730, and the least is sport which is only has 1.

Query 6:

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

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

```
Time taken: 24.93 seconds, Fetched: 10 row(s)
hive> SELECT brand,sum(price) as total_price from hive_table_part where brand !='' and event_type ='purchase' group by brand order by total_price desc limit 1;
Query ID = hadoop_20221204193644_abc8ebdd-71fa-4d02-b667-ab8fd589d919
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_1670175863837_0007)

-----  

 VERTICES      MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

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

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

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

-----  

VERTICES: 03/03 [=====>>>] 100% ELAPSED TIME: 18.43 s  

-----  

OK
runail 148297.9400000021
Time taken: 27.017 seconds, Fetched: 1 row(s)
hive> 
```

Inference:-

Runail is the brand which has the maximum price of 148297/- in the two months October and November. Looks like cosmetics addicts more towards runail brand.

Query 7:

Which brands increased their sales from October to November?

```
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 hive_table_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;
```

```
hadoop@ip-172-31-53-223:~  

hive> WITH Monthly_Revenue AS ( SELECT brand, SUM(CASE WHEN date_format(event_time, 'MM')=10 THEN pr
ice ELSE 0 END) AS Oct_Revenue, SUM(CASE WHEN date_format(event_time, 'MM')=11 THEN price ELSE 0 END
) AS Nov_Revenue FROM hive_table_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 S
ales_Difference FROM Monthly_Revenue WHERE (Nov_Revenue - Oct_Revenue)>0 ORDER BY Sales_Difference;
Query ID = hadoop_20221204185724_f6fd9174-2c08-8ed8-9914c712befd
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1670175863837_0005)

-----  

 VERTICES      MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

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

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

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

-----  

VERTICES: 03/03 [=====>>>] 100% ELAPSED TIME: 33.55 s  

-----  

OK
brand oct_revenue nov_revenue sales_difference
ovale 2.54 3.1 0.56
cosima 20.23 20.929999999999993 0.6999999999999922
grace 100.92000000000002 102.61 1.6899999999999835
helloganic 0.0 3.1 3.1
skinity 8.88 12.44000000000001 3.560000000000005
bodyton 1376.3399999999979 1380.6399999999992 4.3000000000001319
moyou 5.71 10.28000000000001 4.570000000000001
neoleor 43.41 51.7 8.2900000000006
soleo 204.200000000003 212.5299999999994 8.32999999999643
jaguar 1102.110000000001 1110.6499999999999 8.53999999999736
tertio 236.160000000001 245.799999999998 9.63999999999873
fly 17.14 27.17 10.03000000000001
rasyan 18.79999999999997 28.93999999999998 10.14
deoproce 316.84 329.1699999999996 12.329999999999984
barbie 0.0 12.39 12.39
supertan 50.37000000000001 66.51000000000002 16.140000000000008
treaclemoon 163.3699999999995 181.4899999999995 18.120000000000005
kamill 63.0099999999998 81.4899999999998 18.48000000000004
juno 0.0 21.08 21.08
veraclara 50.1100000000014 71.2100000000001 21.09999999999994
glysolid 69.7299999999998 91.5899999999998 21.85999999999987
godefroy 401.2200000000025 425.1200000000023 23.89999999999987
Binacil 0.0 24.25999999999998 24.25999999999998
-----  

OK
22°C Partly cloudy 
```

hadoop@ip-172-31-53-223:~

binacil_0.0	24.259999999999998	24.259999999999998
blixz	38.95	63.39999999999996
profepil	93.360000000000001	118.020000000000001
estelare	444.8099999999996	471.8700000000005
orly	902.3800000000005	931.0900000000006
biore	60.650000000000006	90.3100000000002
beautyblender	78.740000000000001	109.41 30.66999999999987
vilenta	197.6000000000016	231.2100000000002
mavala	409.03999999999974	446.3200000000001
likato	296.059999999999	340.9699999999997
ladykin	125.64999999999999	170.57 44.92
foamie	35.04	80.49 45.4499999999996
elskin	251.0900000000057	307.6500000000083
balbcare	155.3299999999996	212.379999999998
koelcia	55.5	112.75 57.25
profhenna	679.229999999998	736.8500000000006
kares_0.0	59.45	59.45
marutaka-foot	49.21999999999999	109.33 60.1100000000001
dewal	0.0	61.28999999999999
imm	288.02	351.2100000000001
laboratorium	246.4999999999997	312.51999999999987
cutrin	299.37	367.62 68.25
egomania	77.47	146.04 68.57
konad	739.829999999999	810.669999999997
nirvel	163.04	234.3299999999987
koelf	422.729999999998	507.289999999999
plazan	101.37	194.0100000000002
aura	83.95	177.51 93.5599999999999
kerasys	430.91	525.2 94.2900000000002
enjoy	41.35	136.5700000000002
depilflax	2707.669999999996	2803.779999999996
eos	54.33999999999996	152.61 98.2700000000001
carmex	145.08	243.3600000000004
batiste	772.399999999999	874.169999999995
osmo	645.58	762.309999999998
dizao	819.130000000000	945.5100000000008
igrobeauty	513.660000000007	645.069999999998
finish	98.38	230.380000000005
nefertiti	203.5200000000007	366.64 133.1199999999992
elizavecca	70.51	204.3000000000004
muskin	58.04	93.0700000000005
latinol	249.5199999999998	384.5900000000003
farmiona	1692.4599999999996	1843.4300000000003
crystalinas	427.629999999998	584.349999999995

22°C Partly cloudy ENG IN 00:58 05-12-2022

hadoop@ip-172-31-53-223:~

crystalinas	427.629999999998	584.949999999995
chi	358.940000000002	538.610000000002
matreshka	0.0	182.6700000000007
freshbubble	318.7	502.340000000002
mane	66.78999999999999	260.2599999999993
keen	236.350000000005	435.620000000046
ecocraft	41.1600000000004	241.95 200.79
fedua	52.38	263.81 211.43
provoc	827.990000000001	1063.820000000002
skinlite	651.939999999998	890.449999999999
entity	479.7100000000134	719.260000000001
trind	298.070000000005	542.96000000002
protokeratin	201.2500000000003	456.79 255.54
beaugreen	511.509999999999	768.349999999999
bluesky	10307.239999999874	10565.529999999915
candy	534.96	799.379999999998
insight	1443.700000000001	1721.960000000001
kocostar	310.850000000001	594.930000000002
happyfons	801.920000000006	1091.589999999997
kims	330.039999999996	632.04 302.0
shary	871.96	1176.490000000002
nitrile	847.279999999999	1162.679999999982
lowence	242.84	567.749999999999
jas	3318.959999999994	3657.429999999997
ellips	245.849999999994	606.04 360.1900000000005
lador	2083.610000000047	2471.53000000003
naomi_0.0	388.999999999994	388.999999999994
kiss	421.549999999995	817.329999999995
yu-r	271.41	673.709999999998
sophin	1067.86	1515.519999999993
farmavita	837.369999999999	1291.969999999999
bioaqua	942.890000000002	1398.120000000006
greyrum	29.21	489.489999999999
gehwoi	1089.07	1557.679999999998
matrix	3243.250000000014	3726.739999999984
limoni	1308.9	1796.600000000004
s.care	412.68	913.069999999999
coifin	903.000000000002	1428.489999999996
usku	5142.270000000008	5690.309999999993
airnails	5118.899999999939	5691.519999999967
brownenna	14331.369999999954	14916.729999999972
kinetics	6334.249999999994	6945.260000000015
kosmekka	1181.440000000003	1813.369999999999
kaoral	4412.430000000003	5086.070000000006

22°C Partly cloudy ENG IN 00:58 05-12-2022

hadoop@ip-172-31-53-223:~

kaoral	4412.430000000003	5086.070000000006
refectocil	2716.180000000057	3475.580000000067
rosi	3077.039999999994	3841.559999999992
solomeya	1899.669999999993	2685.799999999992
misssha	1293.829999999997	2150.279999999993
levissime	2227.500000000064	3085.309999999999
art-visage	2092.710000000011	2997.8000000000106
ecolab	262.8499999999997	1214.299999999993
nagaraku	4369.7400000000525	5327.680000000057
sanoto	157.14	1209.679999999998
markell	1768.749999999998	2834.429999999998
metzger	5373.449999999996	6457.159999999997
de.lux	1659.6599999999678	2775.5099999999643
swarovski	1887.929999999981	3043.159999999986
zeitun	708.660000000004	2009.629999999999
irisk	45591.96000000514	46898.61000000187
joico	705.52	2015.100000000001
severina	4775.879999999991	6120.480000000105
oniq	8425.41000000002	9841.650000000025
levrana	2243.560000000018	3664.100000000006
smart	4457.259999999994	5902.140000000019
roublöff	3468.340000000015	4913.770000000008
shik	3341.200000000003	4839.720000000005
domix	10472.049999999947	1209.170000000011
artex	2730.6399999999976	4327.25 1596.610000000024
beautix	10493.94999999997	1222.949999999913
milv	3904.939999999905	5642.0100000001175
masura	31266.079999998175	33058.469999999778
f.o.x	6624.229999999971	8577.27999999995
kapous	11927.159999999913	14093.07999999989
concept	11032.139999999934	1380.399999999954
estel	21756.7500000000404	24142.670000000355
kaypro	881.3400000000001	3268.7 2387.359999999997
benovy	409.620000000001	3259.969999999993
italwax	21940.240000000133	24797.39000000026
yoko	8756.909999999953	11707.879999999974
haruyama	9390.90000000006	12352.90999999936
marathon	12801.459999999997	10273.039999999988
lovely	8704.379000000009	1193.060000000001
bpw.style	11572.1500000001746	14837.440000001989
staleks	8519.730000000005	11875.610000000059
freedoor	3421.7799999999643	7671.80000000231
ruinail	71539.2800000005	76758.66000000118

22°C Partly cloudy ENG IN 00:59 05-12-2022

```

hadoop@ip-172-31-53-223:~ 
runail 71539.28000000055 76758.66000000118 5219.38000000063
polarus 6013.720000000001 11371.930000000017 5358.2100000000155
cosmoprofi 8322.81000000005 14536.990000000173 6214.180000000122
jessmail 26287.84000000002 33323.799999999785 7035.959999999766
strong 29196.62999999995 38671.269999999924 9474.639999999974
ingarden 23161.39000000003 33566.209999999344 10404.819999999316
lianall 5892.83999999995 16394.240000000384 10501.400000000389
uno 35302.0299999978 51039.74999999882 15737.71999999904
grattol 35445.54000000099 71472.709999997 36027.169999996004
474679.05999996286 619376.949999914 144697.88999995153
Time taken: 34.255 seconds, Fetched: 161 row(s)
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
    > GROUP BY user_id
[1]+  Stopped                  hive
[hadoop@ip-172-31-53-223 ~]$ hive

```

Inference: There are 161 brands for which the sales increased from October to November. Ovale showing the least difference and Grattol showing the maximum difference. Runail is also among the brands which showed increase in sales. So runail can be claimed as a popular brand.

Query 8.

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

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

```

use
hive>
> SELECT user_id, SUM(price) as Total_Expense FROM hive_table_part WHERE event_type='purchase' G
ROUP BY user_id ORDER BY Total_Expense DESC LIMIT 10;
Query ID = hadoop_20221204190044_4ebc72a9-89d0-4aae-b5b8a8549ca7
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1670175863837_0006)

-----  

  VERTICES      MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

-----  

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

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

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

-----  

VERTICES: 03/03  [=====>>>] 100% ELAPSED TIME: 18.18 s  

-----  

OK  

557790271  2715.8699999999917  

150318419  1645.969999999996  

562167663  1352.850000000001  

531900924  1329.449999999987  

557850743  1295.48  

522130011  1185.39  

561592095  1109.700000000007  

431950134  1097.589999999999  

566576008  1056.360000000017  

521347209  1040.91  

Time taken: 24.93 seconds, Fetched: 10 row(s)
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;

```

Inference:

The output gives the list of top 10 customers who can be rewarded the Golden customer plan.

6. Cleaning Up:

Dropping the database.

Drop database ABDB;

```
hive> DROP DATABASE IF EXISTS ABDB;
OK
Time taken: 0.159 seconds
hive> [green bar]
```

6.1. 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 screenshot shows the AWS EMR Cluster Summary page for a cluster named 'my_hive_cluster'. The cluster is in a 'Waiting' state. On the right, there's a summary card with the following details:

- ID: j-1HLCLYY3CJME
- Creation date: 2022-12-08 00:01 (UTC+5:30)
- Elapsed time: 59 minutes
- After last step completes: Cluster waits
- Termination protection: On Off ✓ ✗
- Tags: -- View All / Edit
- Master public DNS: ec2-3-80-121-110.compute-1.amazonaws.com [Connect to the Master Node Using SSH](#)

A red warning box at the top right states: "Auto-termination is not available for this account when using this release of EMR."

The screenshot shows the AWS EMR Cluster Summary page for the same cluster 'my_hive_cluster'. The cluster is now in a 'Terminating' state. The termination protection setting has been changed to 'Off'. The summary card details are identical to the previous screenshot, except for the state and the termination protection setting.

The screenshot shows the AWS Amazon EMR console. On the left sidebar, there are links for 'Amazon EMR', 'EMR Studio', 'EMR Serverless (New)', and 'EMR on EC2'. Under 'Clusters', there is a link to 'Clusters'. The main area has a 'Create cluster' button and tabs for 'View details', 'Clone', and 'Terminate'. A filter bar at the top says 'Filter: All clusters' and shows '36 clusters (all loaded)'. A table below lists one cluster: 'my_hive_cluster' (ID: j-1HLCLYY3CJKE, Status: Terminated, Creation time: 2022-12-08 00:01 (UTC+5:30), Elapsed time: 1 hour, 44 minutes).

Summary:

The public clickstream dataset of a cosmetics store is used for this analysis.

Steps involved:

1. Copying the data set into the HDFS from S3 Bucket.
2. Creating the database and launching Hive queries on EMR cluster.
3. Necessary optimizations are done, such as selecting the appropriate table format and using partitioned/bucketed tables. CSV Serde is used with the default properties value for loading the dataset into a Hive table.
4. Runail seems to have increased sales from October to November
5. Cleaning up.

Overall, there is increase in sales from October to November.