

HIVE CASE STUDY

STEP 1: CREATING KEY PAIR

The screenshot shows the AWS Management Console homepage. At the top, there's a navigation bar with links for Subscription Details, AWS Management Console, and various services like Apps, Gmail, YouTube, Maps, pandas-profiling, Risk Analytics, Tutorials List, Machine Learning R..., and SELECT basics. Below the navigation is a search bar and a sidebar titled "AWS services" with sections for Recently visited services (EC2, IAM, EMR, S3) and All services. To the right, there are promotional boxes for "Stay connected to your resources on-the-go" (AWS Console Mobile app), "Explore AWS" (Amazon S3 Multi-Region), and "Test Your Machine Learn".

-After opening the AWS console click on “EC2”.

The screenshot shows the EC2 Management Dashboard. The left sidebar includes "New EC2 Experience" (Tell us what you think), "EC2 Dashboard" (EC2 Global View, Events, Tags, Limits), and "Instances" (Instances, Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances). The main content area displays "Resources" (Info) showing EC2 resources in the US East (N. Virginia) Region, including 0 instances (running), 0 elastic IPs, 5 key pairs, 0 placement groups, and 0 snapshots. It also shows 0 dedicated hosts, 0 instances, 0 load balancers, 4 security groups, and 0 volumes. A note at the bottom says: "Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more". The right sidebar contains "Account attributes" (Supported platforms: VPC, Default VPC: vpc-cfc7fb2, Settings, EBS encryption, Zones, EC2 Serial Console, Default credit specification, Console experiments) and "Explore AWS".

On EC2 dashboard click on “Key Pairs”.

Key pairs (5) Info					
<input type="text"/> Filter key pairs					
	Name	Type	Fingerprint	ID	
<input type="checkbox"/>	Ayush_975406046528_2021-05-05 1...	rsa	34:af:53:ad:dd:6d:28:ccc:0:9:c:81:2d:8e...	key-0d3a15ed65708e934	Actions
<input type="checkbox"/>	ayush_ec2_key	rsa	94:d7:19:1d:67:69:24:58:46:41:a1:7fa...	key-02d9d121eadd039fb	Actions
<input type="checkbox"/>	test_key_pair	rsa	45:13:cd:4a:40:74:db:ad:c4:fd:83:56:0...	key-0d0c871e1751c1930	Actions
<input type="checkbox"/>	Test	rsa	fe:ea:b6:0b:fe:64:55:37:66:70:ad:81:49...	key-0226bd76ec421e293	Actions
<input type="checkbox"/>	Test_key_pair_1	rsa	89:99:20:83:39:85:21:1c:83:60:52:c4:d...	key-05309eee8c7261f80	Actions

-Click on "Create key Pair".

Create key pair [Info](#)

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

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

Key pair type [Info](#)
 RSA
 ED25519

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

Tags (Optional)
No tags associated with the resource.
[Add tag](#)
You can add 50 more tags.

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

Fill the name section as shown in the snapshot "**HiveCaseStudy- KeyPair**" and click on "**Create Key pair**".

The screenshot shows the AWS IAM Key Pairs page. At the top, a green banner displays the message "Successfully created key pair". Below this, the heading "Key pairs (6)" is followed by a "Create key pair" button. A search bar labeled "Filter key pairs" is present. The main table lists six key pairs with columns for Name, Type, Fingerprint, and ID. The entries are:

Name	Type	Fingerprint	ID
Ayush_975406046528_2021-05-05 1...	rsa	34:af:53:ad:dd:6d:28:cc:c0:9c:81:2d:8e...	key-0d3a15ed65708e934
ayush_ec2_key	rsa	94:d7:19:1d:67:69:24:58:46:41:a1:7f:a...	key-02d9d121eadd039fb
HiveCaseStudy-KeyPair	rsa	12:eb:60:b7:cd:bb:ea:76:19:c4:fe:22:ce...	key-0e00452f679337a54
Test	rsa	fe:ea:b6:0b:fe:64:55:37:66:70:ad:81:49...	key-0226bd76ec421e293
test_key_pair	rsa	45:13:cd:4a:40:74:db:ad:c4:fd:83:56:0...	key-0d0c871e1751c1930
Test_key_pair_1	rsa	89:99:20:83:39:85:21:1c:83:60:52:c4:d...	key-05309eee8c7261f80

The key-pair is successfully created and .ppk file is downloaded.

STEP 2- CREATING BUCKETS

The screenshot shows the AWS Management Console Services page. The search bar at the top is set to "s3". The page lists various AWS services under categories. The "Storage" category is expanded, showing S3, EFS, FSx, S3 Glacier, Storage Gateway, AWS Backup, and AWS Elastic Disaster Recovery. Other visible categories include Database (RDS, DynamoDB, ElastiCache), Media Services, AWS Cost Management, Front-end Web & Mobile, and several others like Systems Manager, AWS AppSync, Trusted Advisor, Control Tower, AWS License Manager, AWS Well-Architected Tool, Personal Health Dashboard, AWS Chatbot, Launch Wizard, AWS Compute Optimizer, Resource Groups & Tag Editor, Amazon Grafana, Amazon Prometheus, AWS Proton, AWS Resilience Hub, Incident Manager, CloudHSM, Directory Service, WAF & Shield, AWS Firewall Manager, Artifact, Security Hub, Detective, AWS Audit Manager, AWS Signer, AWS Network Firewall, AWS Cost Explorer, AWS Budgets, AWS Marketplace Subscriptions, AWS Application Cost Profiler, and AWS Lambda.

- First, we will find the **S3** services in amazon services and click on it to open.

The screenshot shows the AWS S3 console. On the left, there's a sidebar with links like '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', 'Dashboards', 'AWS Organizations settings', 'Feature spotlight', and 'AWS Marketplace for S3'. The main area has a title 'Amazon S3' and a sub-section 'Amazon S3'. Below that is 'Account snapshot' with metrics: Total storage (993.6 MB), Object count (11), Avg. object size (90.3 MB). A note says 'You can enable advanced metrics in the "default-account-dashboard" configuration.' There's a 'View Storage Lens dashboard' button. Below this is a table titled 'Buckets (5) Info' with columns: Name, AWS Region, Access, and Creation date. The buckets listed are: aws-logs-975406046528-us-east-1, dsc30, empdept, movisample, and samsachi-123. Each row shows the same access level ('Objects can be public') and creation date (November 8, 2021, or later). At the bottom of the page are standard footer links: Feedback, English (US), Support, Home, Regions, us-east-1, © 2021, Amazon Internet Services Private Ltd. or its affiliates., Privacy, Terms, and Cookie preferences.

- We will then click on "Create Bucket"

The screenshot shows the 'Create bucket' wizard. The first step is 'General configuration'. It has fields for 'Bucket name' (HiveCaseStudy-Rajat_Samprit), 'AWS Region' (US East (N. Virginia) us-east-1), and a 'Copy settings from existing bucket - optional' section with a 'Choose bucket' button. The second step is 'Block Public Access settings for this bucket', which contains a detailed description of what block public access does and how it applies to buckets and objects. It also includes a 'Learn more' link.

- We will then put in the appropriate bucket name. Like here we have put it as "**HiveCaseStudy-Rajat_Samprit**". Bucket name should be **unique**.

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 ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

Block public access to buckets and objects granted through *any* access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

Block public access to buckets and objects granted through *new* public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

Block public and cross-account access to buckets and objects through *any* public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

The screenshot shows the AWS S3 Bucket creation interface. The left sidebar lists navigation options like Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and more. The main panel is titled "Amazon S3" and contains sections for "Tags (0) - optional", "Default encryption", and "Advanced settings". Under "Default encryption", the "Server-side encryption" section has a radio button for "Disable" selected. A note at the bottom says, "After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings." At the bottom right are "Cancel" and "Create bucket" buttons.

- We need to enable the public access of the bucket. Therefore, we untick the "Block all public access" button.

The screenshot shows the AWS S3 console. A green banner at the top indicates that a bucket named "hivecasestudy-rajan-samprit" has been successfully created. Below the banner, the "Account snapshot" section provides an overview of storage usage: Total storage is 993.6 MB, Object count is 11, and Avg. object size is 90.3 MB. A note says you can enable advanced metrics in the "default-account-dashboard" configuration. The main area displays a list of buckets, including "aws-logs-975406046528-us-east-1", "dsc30", "empdept", "hivecasestudy-rajan-samprit", "movisample", and "samsachi-123". Each bucket entry includes its name, AWS Region (US East (N. Virginia)), Access level (Objects can be public), and Creation date.

- As our new bucket has been created, we are ready to upload files to this bucket. We will click on our newly created bucket.

The screenshot shows the AWS S3 console with the newly created bucket "hivecasestudy-rajan-samprit" selected. The "Objects" tab is active, showing a table with one row: "No objects". A note below the table states, "You don't have any objects in this bucket." At the bottom of the table is a large orange "Upload" button.

- We click on **upload** and select the files/dataset we want to upload by clicking on "Add files".

The screenshot shows the AWS S3 'Upload: status' page. At the top, a green banner indicates 'Upload succeeded'. Below it, a message says 'The information below will no longer be available after you navigate away from this page.' A 'Summary' section shows the destination 's3://hivecasestudy-rajan-samprit' with two succeeded files (980.7 MB) and zero failed files. Below this, tabs for 'Files and folders' and 'Configuration' are visible. Under 'Files and folders', a table lists one file: '2019-Nov.csv' (text/csv, 520.6 MB, Succeeded). The bottom of the page includes a feedback link, language selection ('English (US)'), and a copyright notice.

- We have successfully uploaded both our files **2019-Nov.csv** and **2019-Oct.csv** in S3 bucket.
- So, we can see 2 FILES 980.7MB files has been uploaded to S3.
- We can now start with the process of our cluster creation

STEP 3- EMR (ELASTIC MAP REDUCE)

The screenshot shows the AWS EMR management console. On the left, a sidebar lists options: Amazon EMR, Amazon Studio, EMR on EC2, Clusters (which is selected), Notebooks, Git repositories, Security configurations, and Block public access. The main area displays a table of clusters. A button for 'Create cluster' is at the top. The table shows one cluster: 'demo_cluster' (ID: j-WSM372B0SV80), which is 'Terminated with errors' due to 'Instance failure'. Other columns include Name, Status, Creation time, Elapsed time, and Normalized instance hours.

- Search for **EMR** on AWS management console and then click on "CREATE CLUSTER".

- Then we click on “**Go to advanced options**”.
- **CLUSTER CREATION: STEP 1- SOFTWARE AND STEPS.**

-In software configuration we choose the **EMR release 5.29.0** for this case study. After that we go on the bottom of this page and click on **Next**.

STEP 2- HARDWARE.

Cluster Nodes and Instances

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

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

Node type	Instance type	Instance count	Purchasing option
Master Master - 1	m4.large 2 vCore, 8 GiB memory, EBS only storage EBS Storage: 32 GiB	1 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price
Core Core - 2	m4.large 2 vCore, 8 GiB memory, EBS only storage EBS Storage: 32 GiB	1 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price
Task Task - 3	m5.xlarge 4 vCore, 16 GiB memory, EBS only storage EBS Storage: 32 GiB	0 Instances	<input checked="" type="radio"/> On-demand <input type="radio"/> Spot Use on-demand as max price

+ Add task instance group

Total core and task units 1 Total units

- In Cluster Nodes and Instances option we choose our **Master Node Instance type as m4.large** and **Core node as m4.large** and **instance count as 1**. After selecting appropriate nodes, we click on **Next**.

STEP 3- GENERAL CLUSTER SETTING.

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

Debugging [?](#)

Termination protection [?](#)

Tags [?](#)

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

Additional Options

EMRFS consistent view [?](#)
Custom AMI ID [?](#)

Bootstrap Actions

In "GENERAL CLUSTER SETTING" we fill our cluster name as "**CaseStudy-Hive**"--> We click on **Next**.

STEP 4- SECURITY.

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

Security Options

EC2 key pair **HiveCaseStudy-KeyPair**

Cluster visible to all IAM users in account

Permissions

Default Custom

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

EMR role **EMR_DefaultRole** Use EMR_DefaultRole_V2

EC2 instance profile **EMR_EC2_DefaultRole**

Auto Scaling role **EMR_AutoScaling_DefaultRole**

► Security Configuration
► EC2 security groups

Create cluster

- In **SECURITY** options we fill our **EC2 key pair** as "**HiveCaseStudy-KeyPair**" which we downloaded and then we click on **CREATE CLUSTER**.

Amazon EMR
EMR Studio
EMR on EC2
Clusters
Notebooks
Git repositories
Security configurations
Block public access
VPC subnets
Events
EMR on EKS
Virtual clusters

Help
What's new

Cluster: CaseStudy-Hive Starting

Summary **Configuration details**

ID: J-1YGNYUWR0J9RL
Creation date: 2021-11-19 23:31 (UTC+5:30)
Elapsed time: 0 seconds
After last step completes: Cluster waits
Termination protection: On Change
Tags: -- View All / Edit
Master public DNS: --

Release label: emr-5.29.0
Hadoop distribution: Amazon 2.8.5
Applications: Hive 2.3.6, Pig 0.17.0, Hue 4.4.0
Log URI: s3://aws-logs-975406046528-us-east-1/elasticmapreduce/

EMRFS consistent view: Disabled
Custom AMI ID: --

Application user interfaces **Network and hardware**

Persistent user interfaces: --
On-cluster user interfaces: --

Availability zone: --
Subnet ID: subnet-0179744c
Master: Provisioning 1 m4.large
Core: Provisioning 1 m4.large
Task: --
Cluster scaling: Not enabled

Security and access

Key name: HiveCaseStudy-KeyPair
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Auto Scaling role: EMR_AutoScaling_DefaultRole
Visible to all users: All

-As we can see here that our cluster is showing the status **STARTING** with 1 master and 1 core node.

Amazon EMR

EMR Studio

EMR on EC2

Clusters

Notebooks

Git repositories

Security configurations

Block public access

VPC subnets

Events

EMR on EKS

Virtual clusters

Help

What's new

Cluster: CaseStudy-Hive Waiting Cluster ready after last step completed.

Summary Application user interfaces Monitoring Hardware Configurations Events Steps Bootstrap actions

Summary		Configuration details	
ID: j-1YGNYUWROJ9RL	Release label: emr-5.29.0	Creation date: 2021-11-19 23:31 (UTC+5:30)	Hadoop distribution: Amazon 2.8.5
Elapsed time: 11 minutes	Applications: Hive 2.3.6, Pig 0.17.0, Hue 4.4.0	After last step completes: Cluster waits	Log URI: s3://aws-logs-975406046528-us-east-1/elasticmapreduce/
Termination protection: On Change	EMRFS consistent view: Disabled	Tags: - View All / Edit	Custom AMI ID: -
Master public DNS: ec2-23-22-247-157.compute-1.amazonaws.com	Connect to the Master Node Using SSH		
Application user interfaces		Network and hardware	
Persistent user interfaces: -	On-cluster user Not Enabled	Enable an SSH Connection	Availability zone: us-east-1d
	Interfaces: -		Subnet ID: subnet-0179744c
			Master: Running 1 m4.large
			Core: Running 1 m4.large
			Task: -
			Cluster scaling: Not enabled
Security and access			
Key name: HiveCaseStudy-KeyPair			
EC2 instance profile: EMR_EC2_DefaultRole			
EMR role: EMR_DefaultRole			
Auto Scaling role: EMR_AutoScaling_DefaultRole			

- Here the cluster shows the status **WAITING** which means the cluster is ready now and the **master** and **core node** is running.

STEP 4- SECURITY GROUPS.

Inbound rules (19)						
	Name	Security group rule...	IP version	Type	Protocol	Port range
	-	sgr-0c7d7a5b7af92e961	IPv4	Custom TCP	TCP	8443
	-	sgr-00e21d468dc495fd	-	All UDP	UDP	0 - 65535
	-	sgr-08dbb6629947a3...	-	All TCP	TCP	0 - 65535
	-	sgr-09c36d02737823a...	IPv4	Custom TCP	TCP	8443
	-	sgr-0c9ad2173efdc0c95	-	All TCP	TCP	0 - 65535
	-	sgr-0dc5eaed33917e50	IPv4	Custom TCP	TCP	8443
	-	sgr-08465463eabb89...	IPv4	Custom TCP	TCP	8443
	-	sgr-09cd3134209f27c8	IPv4	Custom TCP	TCP	8443
	-	sgr-0ede43a1bbfe736cb	IPv4	Custom TCP	TCP	8443
	-	sgr-08dfb91a918f2b19e	-	All UDP	UDP	0 - 65535
	-	sgr-002aa3a566792066...	IPv4	Custom TCP	TCP	8443
	-	sgr-0408f71447fefac24	-	All ICMP - IPv4	ICMP	All
	-	sgr-04bcfa52c612d3b67	IPv4	Custom TCP	TCP	8443
	-	sgr-04923a137255ff8e9	IPv4	Custom TCP	TCP	8443
	-	sgr-0b7d464af2c2b7a3f	IPv4	Custom TCP	TCP	8443
	-	sgr-0a3077085db2d1...	IPv4	Custom TCP	TCP	8443

- Click on EC2 on AWS management console--> On EC2 dashboard select **SECURITY GROUPS**--> Select **ElasticMapReduce-master**--> Click on **edit inbound rules**.

The screenshot shows the AWS Management Console with the Services dropdown selected. The main area displays a list of security groups (sgr-04bc...67, sgr-0492...8e9, etc.) and their inbound rules. A dropdown menu is open over one of the rules, showing options such as Custom TCP, SSH, and SMTP. The SSH option is highlighted. The interface includes search bars, filter dropdowns, and delete buttons for each rule.

- Delete all by default SSH rules and click on **ADD RULES** that you see on the left side of the screen. Drop down **Custom TCP** and select **SSH**. Select the source as **Anywhere IPv4**. After that click on **Save Rules**.

The screenshot shows the AWS EC2 Security Groups page. A green banner at the top indicates that inbound security group rules have been successfully modified. Below this, the details for a security group named "sg-0747e5bd3f2994a13 - ElasticMapReduce-master" are displayed, including its name, ID, owner, and VPC ID. The "Inbound rules" tab is selected, showing 19 permission entries. A note at the bottom encourages users to check network connectivity with the Reachability Analyzer.

- After clicking on save rules you will get the above screen showing that **inbound security group rules successfully modified**.

STEP 5- CONNECTING THE MASTER NODE WITH PUTTY

- Click on the latest cluster that we have created and after that click on the '**Connect to the master node using SSH**'

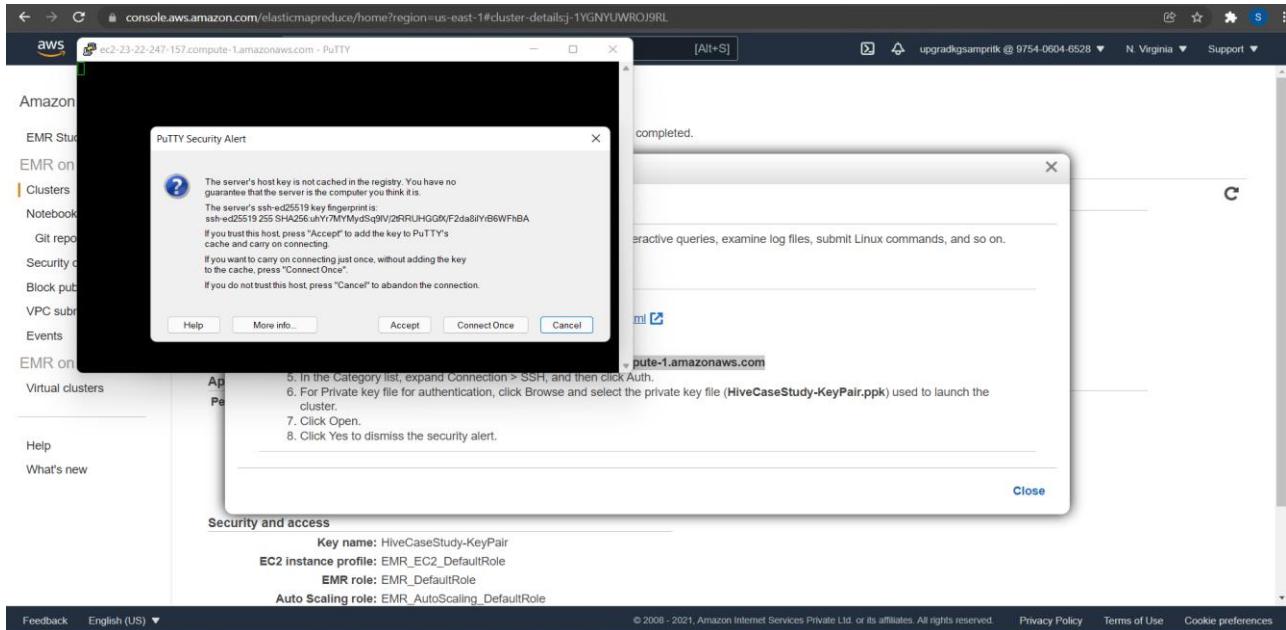
HOST- hadoop@ec2-54-173-34-139.compute-1.amazonaws.com

The screenshot shows the AWS EMR Cluster details page for a cluster named "CaseStudy-Hive". The status is "Waiting" and it says "Cluster ready after last step completed". Below the cluster details, there is a "Connect to the Master Node Using SSH" dialog. This dialog includes instructions for connecting via SSH, mentioning the host name "hadoop@ec2-23-22-247-157.compute-1.amazonaws.com". To the right of this dialog, the "Putty Configuration" window is open. In the "Session" category, the host name is set to "hadoop@ec2-23-22-247-157.compute-1.amazonaws.com" and the port is set to 22. The "Connection type" is set to "SSH". The "Auth" section is expanded, showing the private key file "HiveCaseStudy- KeyPair.ppk" selected. Other tabs in the Putty Configuration window include "Logging", "Terminal", "Keyboard", "Features", "Window", "Appearance", "Behaviour", "Translation", "Selection", "Colours", "Data", "Proxy", and "Connections".

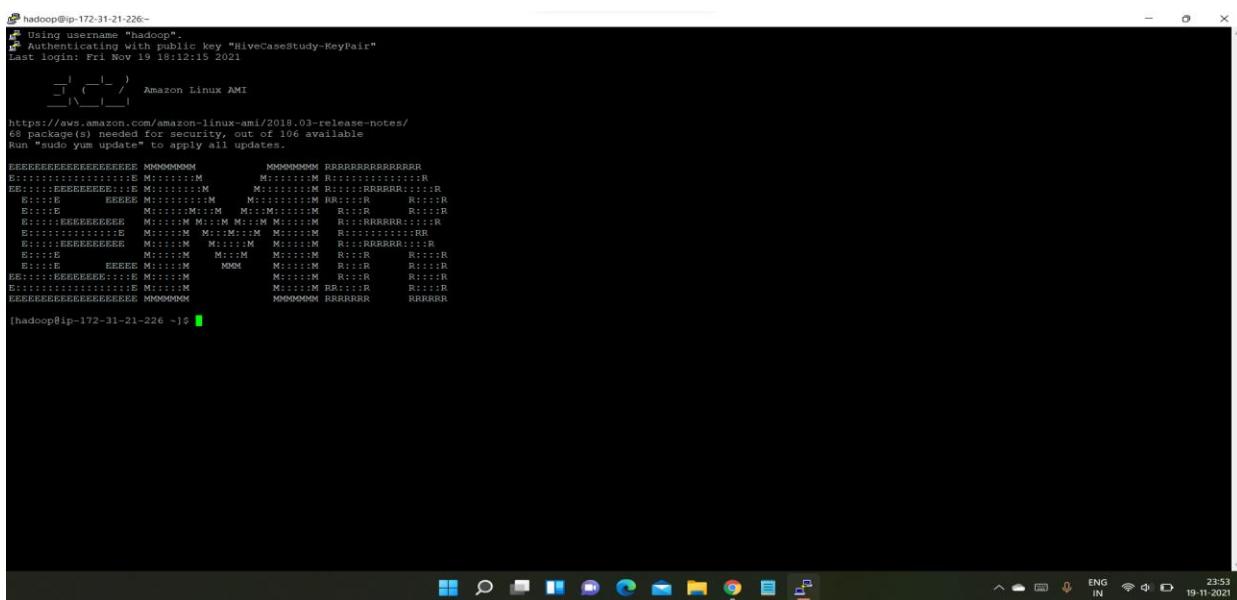
- Launch PUTTY on your system and enter the '**host name**'. On the Category side click on '**SSH**' --> '**Auth**'.

This screenshot is similar to the one above, showing the AWS EMR Cluster details page for the "CaseStudy-Hive" cluster. The "Putty Configuration" window is open, and the "Auth" tab is selected. The "Key" field is populated with "HiveCaseStudy- KeyPair.ppk". Other tabs in the Putty Configuration window include "Logging", "Terminal", "Keyboard", "Features", "Window", "Appearance", "Behaviour", "Translation", "Selection", "Colours", "Data", "Proxy", and "Connections". The "Session" category in the sidebar shows the host name "hadoop@ec2-23-22-247-157.compute-1.amazonaws.com" and port 22.

- Browse the .ppk file i.e **HiveCaseStudy- KeyPair** --> Click on **OPEN**.



-Click on Open and Click on **ACCEPT**.



- EMR(ELASTIC MAP REDUCE) CLI is launched successfully.

STEP 6- QUERYING: Loading data in HDFS

-Checking for the services running in hadoop cluster using command “**sudo initctl list**”.

```
print-image-id stop/waiting
oozie start/running, process 16322
elastic-network-interfaces stop/waiting
splash-manager stop/waiting
hadoop-yarn-proxyserver start/running, process 12103
start-ttys stop/waiting
ranger-kms-server stop/waiting
amazon-ssm-agent start/running, process 3899
hadoop-hdfs-namenode start/running, process 8094
rcs-sulogin stop/waiting
scout-server stop/waiting
serial (ttyS0) start/running, process 4977
[hadoop@ip-172-31-21-226 ~]$ hadoop fs -ls /
Found 4 items
drwxr-xr-x  - hdfs hadoop      0 2021-11-19 18:08 /apps
drwxrwxrwt  - hdfs hadoop      0 2021-11-19 18:10 /tmp
drwxr-xr-x  - hdfs hadoop      0 2021-11-19 18:08 /user
drwxr-xr-x  - hdfs hadoop      0 2021-11-19 18:08 /var
```

- Verifying the Hadoop file system using command “**hadoop fs -ls /**”.

```
drwxr-xr-x - hdfs hadoop 0 2021-11-20 10:08 /apps  
drwxrwxrwt - hdfs hadoop 0 2021-11-20 10:11 /tmp  
drwxr-xr-x - hdfs hadoop 0 2021-11-20 10:08 /user  
drwxr-xr-x - hdfs hadoop 0 2021-11-20 10:08 /var  
[hadoop@ip-172-31-17-238 ~]$ hadoop fs -mkdir /user/hive/hivetestcasestudy
```

- Creating a new folder for our case study using command "**hadoop fs -mkdir /user/hive/hivecasestudy**".

```
[hadoop@ip-172-31-17-238 ~]$ hadoop fs -ls /user/hive/
Found 2 items
drwxr-xr-x  - hadoop hadoop          0 2021-11-20 10:21 /user/hive/hivecasestudy
drwxrwxrwt  - hdfs  hadoop          0 2021-11-20 10:08 /user/hive/warehouse
```

- We verified the directory using the command “**hadoop fs -ls /user/hive/**”.

```
[hadoop@ip-172-31-17-238 ~]$ hadoop distcp s3://hivecasestudy-rajam-samprit/2019-Oct.csv /user/hive/hivecasestudy/2019-Oct.csv
21/11/20 10:27:44 INFO tools.DistCp: Input Options: DistCpOptions{atomicCommit=false, syncFolder=false, deleteMissing=false, ignoreFailures=false, overwrite=false, skipCRC=false, blocking=true, numListStatusThreads=0, maxMaps=20, maxBandwidth=100, sslConfigurationFile='null', copyStrategy='uniformsize', preserveStatus=[], preserveRawXattrs=false, atomicWorkPath=null, logPath=null, sourceFileList=null, sourcePaths=[s3://hivecasestudy-rajam-samprit/2019-Oct.csv], targetPath=/user/hive/hivecasestudy/2019-Oct.csv, targetPathExists=false, filtersFile='null'}
21/11/20 10:27:44 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-17-238.ec2.internal/172.31.17.238:8032
21/11/20 10:27:51 INFO tools.SimpleCopyListing: Paths (files+dirs) cnt = 1; dirCnt = 0
21/11/20 10:27:51 INFO tools.SimpleCopyListing: Build file listing completed.
21/11/20 10:27:51 INFO Configuration.deprecation: io.sort.mb is deprecated. Instead, use mapreduce.task.io.sort.mb
21/11/20 10:27:51 INFO Configuration.deprecation: io.sort.factor is deprecated. Instead, use mapreduce.task.io.sort.factor
21/11/20 10:27:51 INFO tools.DistCp: Number of paths in the copy list: 1
21/11/20 10:27:51 INFO tools.DistCp: Number of paths in the copy list: 1
21/11/20 10:27:51 INFO client.RMProxy: Connecting to ResourceManager at ip-172-31-17-238.ec2.internal/172.31.17.238:8032
21/11/20 10:27:52 INFO mapreduce.JobSubmitter: number of splits:1
21/11/20 10:27:52 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1637403004108_0001
21/11/20 10:27:53 INFO impl.YarnClientImpl: Submitted application application_1637403004108_0001
21/11/20 10:27:53 INFO mapreduce.Job: The url to track the job: http://ip-172-31-17-238.ec2.internal:20888/proxy/application_1637403004108_0001
21/11/20 10:27:53 INFO tools.DistCp: DistCp job-id: job_1637403004108_0001
21/11/20 10:27:53 INFO mapreduce.Job: Running job: job_1637403004108_0001
21/11/20 10:28:02 INFO mapreduce.Job: Job job_1637403004108_0001 running in uber mode : false
21/11/20 10:28:02 INFO mapreduce.Job: map 0% reduce 0%
21/11/20 10:28:21 INFO mapreduce.Job: map 100% reduce 0%
21/11/20 10:28:23 INFO mapreduce.Job: Job job_1637403004108_0001 completed successfully
21/11/20 10:28:23 INFO mapreduce.Job: Counters: 38
    File System Counters
        FILE: Number of bytes read=0
        FILE: Number of bytes written=172507
        FILE: Number of read operations=0
        FILE: Number of large read operations=0
        FILE: Number of write operations=0
        HDFS: Number of bytes read=366
        HDFS: Number of bytes written=482542278
        HDFS: Number of read operations=12
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=4
        S3: Number of bytes read=482542278
        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)=564416
        Total time spent by all reduces in occupied slots (ms)=0
        Total time spent by all map tasks (ms)=17638
        Total vcore-milliseconds taken by all map tasks=17638
        Total megabyte-milliseconds taken by all map tasks=18061312
    Map-Reduce Framework
        Map input records=1
```

- Copying the data from S3 to HDFS Using distributed copy command:

hadoop	distcp	s3://hivecasestudy-rajam-samprit/2019-Oct.csv
/user/hive/hivecasestudy/2019-Oct.csv		

**hadoop distcp s3:// hivecasesstudy-rajat-samprit /2019-Nov.csv
/user/hive/hivecasesstudy/2019-Nov.csv**

```
[hadoop@ip-172-31-17-238 ~]$ hadoop fs -ls /user/hive/hivecasesstudy
Found 2 items
-rw-r--r-- 1 hadoop hadoop 545839412 2021-11-20 10:42 /user/hive/hivecasesstudy/2019-Nov.csv
-rw-r--r-- 1 hadoop hadoop 482542278 2021-11-20 10:28 /user/hive/hivecasesstudy/2019-Oct.csv
```

- Verifying the loaded data in HDFS using command: **hadoop fs -ls /user/hive/hivecasesstudy**

It shows that 2 items are found which means both our files are loaded successfully.

- Checking the top rows for the month of October from our loaded file.

```
[hadoop@ip-172-31-17-238 ~]$ hadoop fs -cat /user/hive/hivecasesstudy/2019-Oct.csv |head
event_time,event_type,product_id,category_id,category_code,brand,price,user_id,user_session
2019-10-01 00:00:00 UTC,cart,5773203,1487580005134238553,,runail,2.62,463240011,26dd6e6e-4dac-4778-8d2c-92e149dab885
2019-10-01 00:00:03 UTC,cart,5773353,1487580005134238553,,runail,2.62,463240011,26dd6e6e-4dac-4778-8d2c-92e149dab885
2019-10-01 00:00:07 UTC,cart,5881589,2151191071051219817,,lovely,13.48,429681830,49e8d843-adf3-428b-a2c3-fe8bc6a307c9
2019-10-01 00:00:07 UTC,cart,5723490,1487580005134238553,,runail,2.62,463240011,26dd6e6e-4dac-4778-8d2c-92e149dab885
2019-10-01 00:00:15 UTC,cart,5881449,148758000513522845895,,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,5739055,1487580008246412266,,kapous,4.75,377667011,81326ac6-daa4-4f0a-b488-fd0956ca78733
2019-10-01 00:00:24 UTC,cart,5825598,1487580009445982239,,,0.56,467916800,2f5b5546-b8cb-9ee7-7ecd-8276f8ef486
2019-10-01 00:00:25 UTC,cart,5698989,1487580006317032337,,,1.27,385985999,d30965e8-1101-44ab-b45d-cc1bb5fae694
```

- Checking the top rows for the month of October from our loaded file.
- Command "**hadoop fs -cat /user/hive/hivecasesstudy/2019-Oct.csv |head**"

```
[hadoop@ip-172-31-17-238 ~]$ hadoop fs -cat /user/hive/hivecasesstudy/2019-Nov.csv |head
event_time,event_type,product_id,category_id,category_code,brand,price,user_id,user_session
2019-11-01 00:00:02 UTC,view,5802432,1487580009286598681,,,0.32,562076640,09fafd6c-0c99-46b1-834f-33527f4de241
2019-11-01 00:00:09 UTC,cart,5844397,1487580006317032337,,,2.38,553329724,2067216c-31b5-455d-a1cc-af0575a34ffb
2019-11-01 00:00:10 UTC,view,5837166,178399064103190764,.pnb,22.22,556138645,57ed22ae-a54a-4907-9944-5a875c2d7f4f
2019-11-01 00:00:11 UTC,cart,5876812,1487580010100293687,,jessmail,3.16,564506666,186c1951-8052-4b37-adce-dd9644b1d5f7
2019-11-01 00:00:24 UTC,remove_from_cart,5826182,1487580007483048900,,,3.33,553329724,2067216c-31b5-455d-a1cc-af0575a34ffb
2019-11-01 00:00:24 UTC,remove_from_cart,5826182,1487580007483048900,,,3.33,553329724,2067216c-31b5-455d-a1cc-af0575a34ffb
2019-11-01 00:00:25 UTC,view,5856189,1487580009026551821,,runail,15.71,562076640,09fafd6c-0c99-46b1-834f-33527f4de241
2019-11-01 00:00:32 UTC,view,5837835,1933472286753424063,,,3.49,514649199,432ade95-375c-4b40-bd36-0fc039e77580
2019-11-01 00:00:34 UTC,remove_from_cart,5870838,1487580007675986993,.milv,0.79,429913900,2f0bf3c-252f-4fe6-afcd-5d8a6a92839a
cat: Unable to write to output stream.
```

- Checking the top rows for the month of November from our loaded file.
- Command "**hadoop fs -cat /user/hive/hivecasesstudy/2019-Nov.csv |head**"

Creating a database.

```
[hadoop@ip-172-31-17-238 ~]$ hive  
  
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false  
hive> |
```

- First we will enter the **hive CLI**(command Line Interface).

```
hive> show databases ;  
OK  
default  
Time taken: 0.639 seconds, Fetched: 1 row(s)  
hive> |
```

- After entering into hive we will Check databases

```
hive> create database if not exists retailstore ;  
OK  
Time taken: 0.503 seconds
```

- We will create database called '**retailstore**' in our databases.
- Command – "**create database if not exists retailstore**".

```
hive> show databases ;  
OK  
default  
retailstore  
Time taken: 0.018 seconds, Fetched: 2 row(s)
```

- Now we will Verify our newly created databases using show databases.
- Command- "**show database**".

```
hive> describe database retailstore ;
OK
retailstore          hdfs://ip-172-31-17-238.ec2.internal:8020/user/hive/warehouse/retailstore.db      hadoop  USER
Time taken: 0.05 seconds, Fetched: 1 row(s)
```

- We will describe our databases.
- Command- “**describe database retailstore**”

```
hive> use retailstore ;
OK
Time taken: 0.02 seconds
```

- We will use **retailstore** database for further query.
- Command- “**use retailstore**”

```
hive> CREATE EXTERNAL TABLE IF NOT EXISTS retailstore (event_time timestamp, event_type string, product_id string, category_id string, category_code string, brand string, price float, user_id bigint, user_session string) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE LOCATION '/user/hive/hivecasestudy' tblproperties("skip.header.line.count"="1");
OK
```

- Now we will create external new Base table
- Command- **CREATE EXTERNAL TABLE IF NOT EXISTS retailstore (event_time timestamp, event_type string, product_id string, category_id string, category_code string, brand string, price float, user_id bigint, user_session string) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE LOCATION '/user/hive/hivecasestudy' tblproperties("skip.header.line.count"="1");**

```
hive> show tables ;
OK
retailstore
Time taken: 0.06 seconds, Fetched: 1 row(s)
```

- Show tables.

```
hive> set hive.cli.print.header = true ;
hive> select * from retailstore limit 5 ;
OK
retailstore.event_time    retailstore.event_type    retailstore.product_id    retailstore.category_id    retailstore.user_session
2019-11-01 00:00:02 UTC    view                  5802432    1487580009286598681          0.32      562076640
2019-11-01 00:00:09 UTC    cart                  5844397    1487580006317032337          2.38      553329724
2019-11-01 00:00:10 UTC    view                  5837166    1783999064103190764          pnb       22.22      556138645
2019-11-01 00:00:11 UTC    cart                  5876812    1487580010100293687          jessnail   3.16      5645
2019-11-01 00:00:24 UTC    remove_from_cart     5826182    1487580007483048900          3.33
Time taken: 2.121 seconds, Fetched: 5 row(s)
```

- We will add **hive.cli.print.header** for putting headers.

```
hive> > CREATE TABLE IF NOT EXISTS dynpart_bucket_retailstore(event_time timestamp, product_id string, r_session string) PARTITIONED BY (event_type string) CLUSTERED BY (price) INTO 10 BUCKETS ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t' LINES TERMINATED BY '\n' STORED AS TEXTFILE tblproperties('skip.header.line.count' = '1');
OK
Time taken: 0.095 seconds
```

- Creating an external table called `dynpart_bucket_retailstore`.
 - Command- `CREATE EXTERNAL TABLE IF NOT EXISTS dynpart_bucket_retailstore (event_time timestamp, event_type string, product_id string, category_id string, category_code string, brand string, price float, user_id bigint, user_session string) ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde' STORED AS TEXTFILE LOCATION '/user/hive/hivecasestudy' tblproperties("skip.header.line.count"="1");`

- Copying the dataset from retailstore table and inserting it into **dyn_part_bucket_retailstore**.
 - Command- `INSERT INTO TABLE dynpart_bucket_retailstore PARTITION (event_type) SELECT event_time,product_id,category_id,category_code,brand , price,user_id,user_session,event_type FROM retail;`

```

Time taken: 0.033 seconds
hive> show tables ;
OK
tab_name
dynpart_bucket_retailstore
retailstore
Time taken: 0.033 seconds, Fetched: 2 row(s)

```

- Verifying the inserted table using show tables . We can see tables have been inserted.
- Command- **show table;**

```

[hadoop@ip-172-31-21-36 ~]$ hadoop fs -ls /user/hive/hivecasestudy
Found 6 items
-rw-r--r-- 1 hadoop hadoop 545839412 2021-11-21 10:30 /user/hive/hivecasestudy/2019-Nov.csv
-rw-r--r-- 1 hadoop hadoop 482542278 2021-11-21 10:29 /user/hive/hivecasestudy/2019-Oct.csv
drwxr-xr-x - hadoop hadoop 0 2021-11-21 11:28 /user/hive/hivecasestudy/event_type=cart
drwxr-xr-x - hadoop hadoop 0 2021-11-21 11:28 /user/hive/hivecasestudy/event_type=purchase
drwxr-xr-x - hadoop hadoop 0 2021-11-21 11:28 /user/hive/hivecasestudy/event_type=remove_from_cart
drwxr-xr-x - hadoop hadoop 0 2021-11-21 11:28 /user/hive/hivecasestudy/event_type=view
[hadoop@ip-172-31-21-36 ~]$ hadoop fs -ls /user/hive/hivecasestudy/event_type=cart
Found 10 items
-rwxr-xr-x 1 hadoop hadoop 27512579 2021-11-21 11:26 /user/hive/hivecasestudy/event_type=cart/000000_0
-rwxr-xr-x 1 hadoop hadoop 32190447 2021-11-21 11:26 /user/hive/hivecasestudy/event_type=cart/000001_0
-rwxr-xr-x 1 hadoop hadoop 33302805 2021-11-21 11:27 /user/hive/hivecasestudy/event_type=cart/000002_0
-rwxr-xr-x 1 hadoop hadoop 32602023 2021-11-21 11:27 /user/hive/hivecasestudy/event_type=cart/000003_0
-rwxr-xr-x 1 hadoop hadoop 34104132 2021-11-21 11:26 /user/hive/hivecasestudy/event_type=cart/000004_0
-rwxr-xr-x 1 hadoop hadoop 32538513 2021-11-21 11:27 /user/hive/hivecasestudy/event_type=cart/000005_0
-rwxr-xr-x 1 hadoop hadoop 39257340 2021-11-21 11:27 /user/hive/hivecasestudy/event_type=cart/000006_0
-rwxr-xr-x 1 hadoop hadoop 24825787 2021-11-21 11:27 /user/hive/hivecasestudy/event_type=cart/000007_0
-rwxr-xr-x 1 hadoop hadoop 28504487 2021-11-21 11:27 /user/hive/hivecasestudy/event_type=cart/000008_0
-rwxr-xr-x 1 hadoop hadoop 35410315 2021-11-21 11:27 /user/hive/hivecasestudy/event_type=cart/000009_0

```

- Verifying partitioned files and bucketed files.
- Command- **hadoop fs -ls /user/hive/hivecasestudy**

```

hive> use retailstore ;
OK
Time taken: 0.019 seconds
hive> select * from retailstore limit 5 ;
OK
2019-11-01 00:00:02 UTC view    5802432 1487580009286598681      0.32   562076640  09fafd6c-6c99-
2019-11-01 00:00:09 UTC cart    5844397 1487580006317032337      2.38   553329724  2067216c-31b5-
2019-11-01 00:00:10 UTC view    5837166 1783999064103190764      pnb    22.22  556138645  57ed222e-a54a-
2019-11-01 00:00:11 UTC cart    5876812 1487580010100293687      jessnail 3.16   564506666  186c19-
2019-11-01 00:00:24 UTC remove_from_cart 5826182 1487580007483048900      3.33   553329724
Time taken: 2.38 seconds, Fetched: 5 row(s)
hive> select * from dynpart_bucket_retailstore limit 5 ;
OK
2019-10-08 09:19:19 UTC 89350  1487580011652186237      runail  1.27   232701853  3f1469f5-d926-44ce-a9f
2019-10-10 05:29:47 UTC 5866208 1487580013841613016      concept  3.16   493381333  535bb6b7-08f4-4021-acd
2019-10-08 12:25:50 UTC 5821183 1487580007717929935      1.27   546703849  3daf4d64-5ffa-46cc-827
2019-10-10 08:19:06 UTC 5848901 1487580007675986893      bpw.style 1.27   439370683  9aeb4d9a-1bed-
2019-10-09 18:32:50 UTC 5869152 1487580005268456287      cosmoprofi 7.94   558533352  cfde0f74-8705-
Time taken: 0.225 seconds, Fetched: 5 row(s)
hive>

```

- After partitioned files we will use retailstore and check the first row for **retailstore** and **dynpart_bucket_retailstore**.

Command-

- Without optimization- **select * from retailstore limit 5;**
- With optimization- **select * from dynpart_bucket_retailstore limit 5;**

QUESTIONS....

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

Base Table: **retailstore** table

```
SELECT SUM(price) AS tot_revenue_oct FROM retailstore WHERE MONTH(event_time) = '10'  
AND event_type = 'purchase';
```

OptimisedTable: **dynpart_bucket_retailstore** table

```
SELECT SUM(price) AS tot_revenue_oct FROM dynpart_bucket_retailstore WHERE  
MONTH(event_time) = '10' AND event_type = 'purchase';
```

```
hive> SELECT SUM(price) AS tot_revenue_oct FROM retailstore WHERE MONTH(event_time) = '10' AND event_type = 'purchase';  
Query ID = hadoop_20211122053035_0fe72c98-e268-4clf-8b40-51b1179e5103  
Total jobs = 1  
Launching Job 1 out of 1  
Status: Running (Executing on YARN cluster with App id application_1637556007874_0006)  
-----  
 VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  
Map 1 ..... container  SUCCEEDED    5       5       0       0       0       0       0  
Reducer 2 ..... container  SUCCEEDED    1       1       0       0       0       0       0  
-----  
VERTICES: 02/02  [=====>>>] 100%  ELAPSED TIME: 125.18 s  
-----  
OK  
1211538.4299997438  
Time taken: 126.027 seconds, Fetched: 1 row(s)  
hive> SELECT SUM(price) AS tot_revenue_oct from dynpart_bucket_retailstore WHERE MONTH(event_time) = '10' AND event_type = 'purchase';  
Query ID = hadoop_20211122053648_d507ee77-9ceb-4ce2-bffa-7222eb633603  
Total jobs = 1  
Launching Job 1 out of 1  
Status: Running (Executing on YARN cluster with App id application_1637556007874_0006)  
-----  
 VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  
Map 1 ..... container  SUCCEEDED    3       3       0       0       0       0       0  
Reducer 2 ..... container  SUCCEEDED    1       1       0       0       0       0       0  
-----  
VERTICES: 02/02  [=====>>>] 100%  ELAPSED TIME: 23.69 s  
-----  
OK  
1211532.4500002791  
Time taken: 24.715 seconds, Fetched: 1 row(s)
```

Output & Observation-

- Total revenue generated in the month of October is 1211538.43 +/- on purchase.
- Time taken to execute the query is 125.18 seconds when table is not optimized whereas time taken to execute is 23.69 when the table is optimized.
- The optimized table reduces the execution time to a significant level. Thus, we can say that performance of the optimized table is better.

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

Base Table : **retailstore** table

```
SELECT MONTH(event_time) AS month, COUNT(event_type) AS total_purchases FROM  
retailstore WHERE event_type = 'purchase' GROUP BY MONTH(event_time);
```

Optimisation Table **dynpart_bucket_retailstore** table

```
SELECT MONTH(event_time) AS month, COUNT(event_type) AS total_purchases FROM
dynpart_bucket_retailstore WHERE event_type = 'purchase' GROUP BY MONTH(event_time);
```

```
hive> SELECT MONTH(event_time) AS month, COUNT(event_type) AS total_purchases FROM retailstore WHERE event_type = 'purchase' GROUP BY MONTH(event_time);
Query ID = hadoop_20211122054616_6711018d-a66a-497c-bbc9-a557c72a7323
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637556007874_0007)

-----  

      VERTICES    MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

Map 1 ..... container    SUCCEEDED   5      5      0      0      0      0  

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

-----  

VERTICES: 02/02 [=====>>>] 100% ELAPSED TIME: 100.25 s  

-----  

OK  

10      245624  

11      322417  

Time taken: 101.193 seconds, Fetched: 2 row(s)
hives> SELECT MONTH(event_time) AS month, COUNT(event_type) AS total_purchases FROM dynpart_bucket_retailstore WHERE event_type = 'purchase' GROUP BY MONTH(event_time);
Query ID = hadoop_20211122054959_e5cf3704-b06f-4221-5f61-93ffffae2a77a
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637556007874_0007)

-----  

      VERTICES    MODE      STATUS TOTAL COMPLETED RUNNING PENDING FAILED KILLED  

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

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

-----  

VERTICES: 02/02 [=====>>>] 100% ELAPSED TIME: 22.98 s  

-----  

OK  

10      245619  

11      322412  

Time taken: 23.904 seconds, Fetched: 2 row(s)
```

Output & Observation-

- Total purchase for the month of October is 245624 and for November it is 322417. There is a significant increase in November purchase, around 30%.
- Execution time for un-optimized table is 100.25 seconds and 22.98 seconds for optimized table which is again a very significant difference. Thus, our optimized table again gives better performance in respect to time.

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

Base Table **retailstore** table

```
SELECT (SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END) - SUM(CASE WHEN MONTH(event_time)=10 THEN price ELSE 0 END)) AS change_rev FROM retailstore
WHERE event_type = 'purchase' AND MONTH(event_time) in ('10','11');
```

Optimised Table: **dynpart_bucket_retailstore** table

```
SELECT (SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END) - SUM(CASE WHEN MONTH(event_time)=10 THEN price ELSE 0 END)) AS change_rev FROM
dynpart_bucket_retailstore WHERE event_type = 'purchase' AND MONTH(event_time) in ('10','11');
```

```

hive> SELECT (SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END) - SUM(CASE WHEN MONTH(event_time)=10 THE
se' AND MONTH(event_time) in ('10','11'));
Query ID = hadoop_20211122071550_6eeee7dc-1483-4cd8-9b55-514f0eea3998
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637564392791_0004)

-----  

      VERTICES    MODE     STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  

Map 1 ..... container  SUCCEEDED   5       5       0       0       0       0       0  

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

-----  

VERTICES: 02/02  [=====>>] 100%  ELAPSED TIME: 102.84 s  

-----  

OK  

319478.4700003781  

Time taken: 105.478 seconds, Fetched: 1 row(s)
hive> SELECT (SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END) - SUM(CASE WHEN MONTH(event_time)=10 THE
_type = 'purchase' AND MONTH(event_time) in ('10','11'));
Query ID = hadoop_20211122071750_f2f8e609-4bd4-401c-b734-928d9ba76e5c
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637564392791_0004)

-----  

      VERTICES    MODE     STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  

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

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

-----  

VERTICES: 02/02  [=====>>] 100%  ELAPSED TIME: 25.46 s  

-----  

OK  

319437.7899997565  

Time taken: 26.363 seconds, Fetched: 1 row(s)

```

Output & Observation-

- We can observe that the difference in revenue between October purchases and November Purchases is 319478.
- Execution time for un-optimized table is 102.84 seconds and 25.46 seconds for optimized table which is again a very significant difference. Thus, our optimized table again gives better performance in respect to time.

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

With Optimisztion(**dynpart_bucket_retailstore** table)-

```

SELECT DISTINCT SPLIT(category_code,'\\.')[0] AS Category FROM
dynpart_bucket_retailstore WHERE category_code != "";

```

```

hive> SELECT DISTINCT SPLIT(category_code,'\\.')[0] AS Category FROM dynpart_bucket_retailstore WHERE cat
Query ID = hadoop_20211122072148_730c7c71-621b-4641-8a71-b6ecc8ea3255
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637564392791_0004)

-----
      VERTICES     MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container  SUCCEEDED    6        6          0        0        0        0        0
Reducer 2 ..... container  SUCCEEDED    5        5          0        0        0        0        0
-----
VERTICES: 02/02  [======>>] 100%  ELAPSED TIME: 68.46 s
-----
OK
furniture
appliances
accessories
apparel
sport
stationery
Time taken: 69.344 seconds, Fetched: 6 row(s)

```

Output & Observation-

- The distinct categories of the products are:
 - furniture
 - Appliances
 - Accessories
 - Apparel
 - Sport
 - Stationary
- The execution time here is 69.344 seconds.

5. Find the total number of products available under each category.

With Optimisation(dynpart_bucket_retailstore table)-

```

SELECT SPLIT(category_code,'\\.')[0] AS Category, COUNT(product_id) AS
number_of_products FROM dynpart_bucket_retailstore WHERE category_code != " GROUP

```

```

BY SPLIT(category_code,'\\.')[0] ORDER BY number_of_products DESC;
hive> SELECT SPLIT(category_code,'\\.') [0] AS Category, COUNT(product_id) AS number_of_products FROM dynpart_bu
[0] ORDER BY number_of_products DESC;
Query ID = hadoop_20211122072441_7cb34ce2-579c-4117-8488-3a51e66aa367
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637564392791_0004)

-----
      VERTICES    MODE     STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container  SUCCEEDED   6       6        0        0        0        0
Reducer 2 ..... container  SUCCEEDED   5       5        0        0        0        0
Reducer 3 ..... container  SUCCEEDED   1       1        0        0        0        0
-----
VERTICES: 03/03  [=====>>>] 100%  ELAPSED TIME: 64.90 s
-----
OK
appliances      61736
stationery       26722
furniture        23604
apparel          18232
accessories      12928
sport             2
Time taken: 65.966 seconds, Fetched: 6 row(s)

```

Output & Observation-

No.	Category	Number of Products
1.	Appliances	61736
2.	Stationery	26722
3.	Furniture	23604
4.	Apparel	18232
5.	Accessories	12928
6.	Sports	2

- Appliances category have the most number of products followed by Stationary. Sports category has the least number of products.
- Execution time here is 65.966 seconds.

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

With Optimisztion(**dynpart_bucket_retailstore** table)-

```

WITH total_sales_summary AS( SELECT brand, round((SUM(CASE WHEN
MONTH(event_time)=10 THEN price ELSE 0 END) + SUM(CASE WHEN
MONTH(event_time)=11 THEN PRICE ELSE 0 END)),2) AS total_sales FROM
dynpart_bucket_retailstore WHERE event_type = 'purchase' AND MONTH(event_time) in
('10','11') AND brand != " GROUP BY brand)
```

```
SELECT brand, total_sales FROM total_sales_summary ORDER BY total_sales DESC LIMIT 1;
```

```
hive> WITH total_sales_summary AS(
    > SELECT brand, round((SUM(CASE WHEN MONTH(event_time)=10 THEN price ELSE 0 END) + SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END)) / 2) AS total_sales
    > FROM event_log WHERE event_type = 'purchase' AND MONTH(event_time) in ('10','11') AND brand != ''
    > GROUP BY brand
    > ORDER BY total_sales DESC LIMIT 1;
Query ID = hadoop_20211122072729_edcaf121-6fa0-4785-80a4-3be1cb407f65
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637564392791_0004)

-----
      VERTICES      MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
-----
Map 1 ..... container  SUCCEEDED      3        3          0          0          0          0
Reducer 2 ..... container  SUCCEEDED      1        1          0          0          0          0
Reducer 3 ..... container  SUCCEEDED      1        1          0          0          0          0
-----
VERTICES: 03/03  [=====>>] 100%  ELAPSED TIME: 25.94 s
-----
OK
runail 148292.46
Time taken: 26.614 seconds, Fetched: 1 row(s)
```

Output & Observation-

- It can be seen that **runail** has the highest number of sales for the month of October and November with total sales of 148292.
- Time taken to execute the query is 25.95 seconds.

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

With Optimisztion(**dynpart_bucket_retailstore** table)-

WITH brand_sales_summary AS(

```
SELECT brand, round(SUM(CASE WHEN MONTH(event_time)=10 THEN price ELSE 0 END),2) AS Oct_sales,round(SUM(CASE WHEN MONTH(event_time)=11 THEN price ELSE 0 END),2) AS Nov_sales
FROM dynpart_bucket_retailstore WHERE event_type = 'purchase' AND MONTH(event_time) in ('10','11') AND brand != '' GROUP BY brand)
```

```
SELECT brand, Oct_sales, Nov_sales, round((Nov_sales-Oct_sales),2)AS Change_in_Sales
FROM brand_sales_summary WHERE Nov_sales-Oct_sales > 0 ORDER BY Change_in_Sales DESC;
```

```

hive> WITH brand_sales_summary AS(
    > SELECT brand, round(SUM(CASE WHEN MONTH(event_time)=10 THEN price ELSE 0 END),2) AS Oct_sales,round(SUM
art_bucket_retailstore WHERE event_type = 'purchase' AND MONTH(event_time) in ('10','11') AND brand != '' GROUP
-> SELECT brand, Oct_sales, Nov_sales, round((Nov_sales-Oct_sales),2)AS Change_in_Sales FROM brand_sales_su
Query ID = hadoop_20211122072925_b0c1c291-9c05-4ada-86ef-cb142a9b6e46
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637564392791_0004)

-----  

      VERTICES      MODE      STATUS     TOTAL     COMPLETED     RUNNING     PENDING     FAILED     KILLED  

-----  

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

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

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

-----  

VERTICES: 03/03 [=====>>>] 100% ELAPSED TIME: 25.58 s  

-----  

OK  

grattol 35445.54 71472.71 36027.17  

uno 35302.03 51039.75 15737.72  

lianail 5892.84 16394.24 10501.4  

ingarden 23161.39 33566.21 10404.82  

strong 29196.63 38671.27 9474.64  

jessnail 26287.84 33345.23 7057.39  

cosmoprofi 8322.81 14536.99 6214.18  

polarus 6013.72 11371.93 5358.21  

runail 71537.77 76754.69 5216.92  

freedecor 3421.78 7671.8 4250.02  

staleks 8519.73 11875.61 3355.88  

bpw.style 11572.15 14837.44 3265.29  

lovely 8704.38 11939.06 3234.68  

marathon 7280.75 10273.1 2992.35  

haruyama 9390.69 12352.91 2962.22  

yoko 8756.91 11707.88 2950.97  

italwax 21940.24 24799.37 2859.13  

benovy 409.62 3259.97 2850.35  

kaypro 881.34 3268.7 2387.36  

estel 21756.75 24142.67 2385.92  

concept 11032.14 13380.4 2348.26  

kapous 11927.16 14093.08 2165.92  

f.o.x 6624.23 8577.28 1953.05  

masura 31266.08 33058.47 1792.39  

milv 3904.94 5642.01 1737.07  

beautix 10493.95 12222.95 1729.0  

artex 2730.64 4327.25 1596.61  

domix 10472.05 12009.17 1537.12  

shik 3341.2 4839.72 1498.52  

-----  

nirvel 163.04 234.33 71.29  

konad 739.83 810.67 70.84  

egomania 77.47 146.04 68.57  

Cutrin 299.37 367.62 68.25  

laboratorium 246.5 312.52 66.02  

inn 288.02 351.21 63.19  

dewal 0.0 61.29 61.29  

marutaka-foot 49.22 109.33 60.11  

kares 0.0 59.45 59.45  

profhenna 679.23 736.85 57.62  

koelcia 55.5 112.75 51.25  

balbcare 155.33 212.38 57.05  

elskin 307.65 56.56  

foamie 35.04 80.49 45.45  

ladykin 125.65 170.57 44.92  

likato 296.06 340.97 44.91  

mavala 409.04 446.32 37.28  

vilenta 197.6 231.21 33.61  

beautyblender 78.74 109.41 30.67  

biore 60.65 90.31 29.66  

orly 902.38 931.09 28.71  

estelare 444.81 471.87 27.06  

prorepil 93.36 118.02 24.66  

blixz 38.95 63.4 24.45  

binacil 0.0 24.26 24.26  

godefroy 401.22 425.12 23.9  

glysolid 69.73 91.59 21.86  

veraclara 50.11 71.21 21.1  

juno 0.0 21.08 21.08  

kamill 63.01 81.49 18.48  

treaclemoon 163.37 181.49 18.12  

supertan 50.37 66.51 16.14  

barbie 0.0 12.39 12.39  

deoproce 316.84 329.17 12.33  

rasyan 18.8 28.94 10.14  

fly 17.14 27.17 10.03  

tertio 236.16 245.8 9.64  

jaguar 1102.11 1110.65 8.54  

soleo 204.2 212.53 8.33  

neoleor 43.41 51.7 8.29  

moyou 5.71 10.28 4.57  

bodyton 1376.34 1380.64 4.3  

skinity 8.88 12.44 3.56  

helloganic 0.0 3.1 3.1  

grace 100.92 102.61 1.69  

cosima 20.23 20.93 0.7  

ovale 2.54 3.1 0.56  

Time taken: 26.222 seconds, Fetched: 160 row(s)

```

Output & Observation-

- 'Grattol' brand has the highest total increment from October to November i.e., 36,027 which is around 20% . 'Ovale' seems to have the least increment of 0.56 from October to November.
- From the output we can see there are total of 160 brands that has increased their sales from October to November.
- Total time taken for execution is 26.222 seconds.

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.

With Optimisztion(**dynpart_bucket_retailstore** table)-

```
SELECT user_id, round(SUM(price),2) AS total_spends FROM dynpart_bucket_estore WHERE event_type = 'purchase' GROUP BY user_id ORDER BY total_spends DESC LIMIT 10;
```

```
hive> SELECT user_id, round(SUM(price),2) AS total_spends FROM dynpart_bucket_retailstore WHERE event_type = 'purchase'
Query ID = hadoop_20211122073204_ef0542ef-e08e-496e-be87-613c7a96c3f8
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1637564392791_0004)

-----  

      VERTICES    MODE      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED  

-----  

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

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

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

-----  

VERTICES: 03/03  [=====>>] 100%  ELAPSED TIME: 25.64 s  

-----  

OK  

557790271      2715.87  

150318419      1645.97  

562167663      1352.85  

531900924      1329.45  

557850743      1295.48  

522130011      1185.39  

561592095      1109.7  

431950134      1097.59  

566576008      1056.36  

521347209      1040.91  

Time taken: 26.311 seconds, Fetched: 10 row(s)
hive> █
```

Output & Observation-

- Here is the list of top 10 users in the output who deserve Golden Customer Plan. They have spent in the range of 2716 and 1040.
- The execution time is 26.311 seconds.

Termination of EMR Cluster.

After completing the analysis it is important to terminate the cluster. If the cluster is left running aws billing will keep on increasing for the users account even when it is not being used. So here are the steps how one can terminate the cluster after finishing their work.....

Search for EMR--> Click on the cluster that has to be terminated--> Click on terminate.

In the window above we can see that the “Termination” is in progress.

Termination is complete. Both our master and core node has been terminated.

The screenshot shows the AWS EMR console. On the left, there's a sidebar with links: Amazon EMR, EMR Studio, EMR on EC2, Clusters (which is selected), Notebooks, and Git repositories. The main area has a header with a search bar ('Search for services, features, blogs, docs, and more'), a help icon, and account information ('N. Virginia' and 'upgradikgsampitk @ 9754-0604-6528'). A banner at the top says 'Get fine-grained access control for your Apache Spark applications. Learn more' with a link icon. Below the banner is a table titled 'Clusters'. The table has columns: Name, ID, Status, Creation time (UTC+5:30), Elapsed time, and Normalized instance hours. There is one row visible: 'CaseStudy-Hive' (ID: j-3QOUY72U8F262), Status: 'Terminated User request', Creation time: '2021-11-22 12:22 (UTC+5:30)', Elapsed time: '52 minutes', and Normalized instance hours: '8'.

Name	ID	Status	Creation time (UTC+5:30)	Elapsed time	Normalized instance hours
CaseStudy-Hive	j-3QOUY72U8F262	Terminated User request	2021-11-22 12:22 (UTC+5:30)	52 minutes	8

The status here shows terminated.

With this we have completed analysis on our dataset successfully.

Thank You!

