

ASSIGNMENT 3: PARALLEL PROCESSING

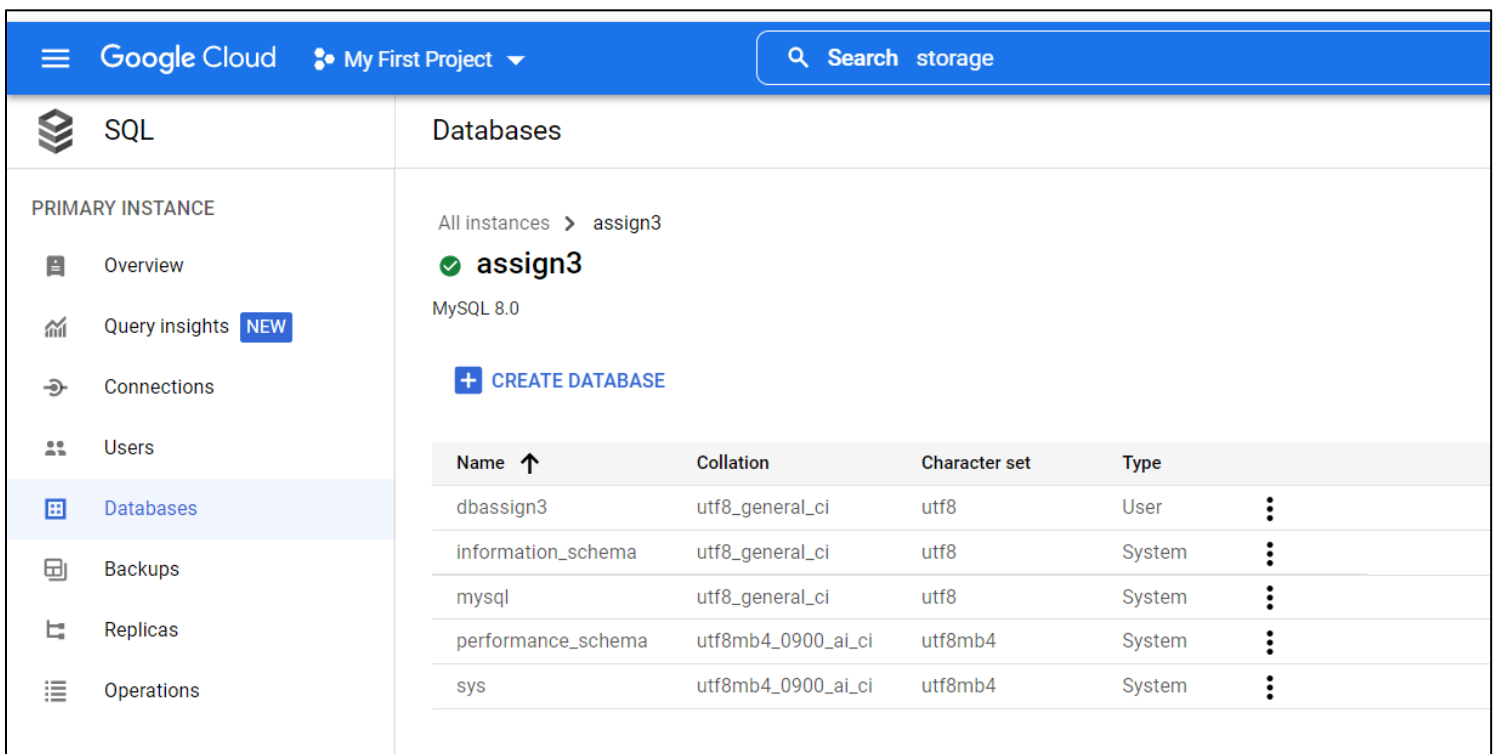
HARSHIKA AKULA (2075727)

In this report, we built a pyspark application in jupyter notebook to analyze the data and executed in EMR cluster. The dataset has been imported and taken from google cloud platform. Below are the steps for the process.

1. Uploading the dataset from local system to GOOGLE CLOUD

- i. In this step, we are creating an MySQL instance named “assign3” by giving specific details like Rootpassword, region and database version. We should also specify the ip address of jupyter notebook that we are using inorder to connect to this sql instance. I have uploaded mysql file from local system to cloud storage into a bucket named “**mysqlscript3**”
- ii. Under **assign3** sql instance, I have created a database named “**dbassign3**” into which we are importing sql file that we uploaded in the **mysqlscript3** bucket.

Below is the screenshot for **assign3** sql instance with **dbassign3** database



The screenshot shows the Google Cloud console interface for a MySQL instance named 'assign3'. The left sidebar contains navigation links for Overview, Query insights, Connections, Users, Databases, Backups, Replicas, and Operations. The 'Databases' link is selected. The main content area shows the instance 'assign3' with MySQL 8.0 version. Below this, there is a '+ CREATE DATABASE' button and a table listing the existing databases.

Name ↑	Collation	Character set	Type	
dbassign3	utf8_general_ci	utf8	User	⋮
information_schema	utf8_general_ci	utf8	System	⋮
mysql	utf8_general_ci	utf8	System	⋮
performance_schema	utf8mb4_0900_ai_ci	utf8mb4	System	⋮
sys	utf8mb4_0900_ai_ci	utf8mb4	System	⋮

Below is the screenshot for bucket creation named “**mysqlscript3**”

Filter Filter buckets							
<input type="checkbox"/>	Name ↑	Created	Location type	Location	Default storage class ?	Last modified ?	Public access ?
<input type="checkbox"/>	mysqscript3	Nov 7, 2022, 12:20:45 PM	Multi-region	us	Standard	Nov 7, 2022, 1:20:32 PM	Not public

2. Creation an EMR cluster in AWS

- In this step,we log in to aws console and go to EMR service. We create a cluster named “**demo**” and specific details like emr version(emr-5.36.0) , Applications(Spark), No EC2 key-pair.Under advanced options, we also make changes in software configuration to run spark application in jupyter notebook.

General Configuration

Cluster name

☒ Logging ⓘ

S3 folder

Launch mode
☒ Cluster ⓘ
☐ Step execution ⓘ

Software configuration

Release
 ⓘ

Applications

☐ Core Hadoop: Hadoop 2.10.1, Hive 2.3.9, Hue 4.10.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2
☐ HBase: HBase 1.4.13, Hadoop 2.10.1, Hive 2.3.9, Hue 4.10.0, Phoenix 4.14.3, and ZooKeeper 3.4.14
☐ Presto: Presto 0.267 with Hadoop 2.10.1 HDFS and Hive 2.3.9 Metastore
☒ Spark: Spark 2.4.8 on Hadoop 2.10.1 YARN and Zeppelin 0.10.0
☐ Use AWS Glue Data Catalog for table metadata ⓘ

Hardware configuration

Instance type

The selected instance type adds 64 GiB of GP2 EBS storage per instance by default. [Learn more](#)

Number of instances
 (1 master and 0 core nodes)

Cluster scaling
☐ scale cluster nodes based on workload

Auto-termination
☒ Enable auto-termination [Learn more](#)

Terminate cluster when it is idle after
 hours
 minutes

Security and access

EC2 key pair
 ⓘ [Learn how to create an EC2 key pair.](#)

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

Step 1: Software and Steps

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

Software Configuration

Release emr-5.36.0

☒ Hadoop 2.10.1
☒ JupyterHub 1.4.1
☐ Ganglia 3.7.2
☒ Hive 2.3.9
☒ JupyterEnterpriseGateway 2.1.0
☐ Mahout 0.13.0
☐ Oozie 5.2.1
☐ TensorFlow 2.4.1

☐ Zeppelin 0.10.0
☐ Tez 0.9.2
☐ HBase 1.4.13
☐ Presto 0.267
☐ MXNet 1.8.0
☒ Hue 4.10.0
☒ Spark 2.4.8

☐ Livy 0.7.1
☐ Flink 1.14.2
☒ Pig 0.17.0
☐ ZooKeeper 3.4.14
☐ Sqoop 1.4.7
☐ Phoenix 4.14.3
☐ HCatalog 2.3.9

- ii. Under IAM Console, We also need to attach two policies for EMR_DefaultRole, those policies are “**AmazonElasticMapReduceEditorsRole**” and “**AmazonS3FullAccess**”.
- iii. After clicking on create button, we can monitor the status of cluster creation. If the status says “waiting cluster ready”, cluster is up and running.

Create cluster

View details

Clone

Terminate

Filter: All clusters

Filter clusters ...

2 clusters (all loaded)

	Name	ID	Status	Creation time (UTC-6)	Elapsed time	Normalized instance hours
<input type="checkbox"/>	demo	j-1J6CJR9AK8TS1	Waiting Cluster ready	2022-11-08 18:46 (UTC-6)	26 minutes	0

- iv. After creation of cluster, we create a notebook named “**demo_note**”. since we have an active cluster “**demo**” running, we choose the existing cluster to attach to the notebook. choose the default service IAM role and click on create notebook. If the status of the notebook says ready then we can launch the Jupyter.


Notebook name*


Names may only contain alphanumeric characters, hyphens (-), or underscores (_).


Description

256 characters max.



Cluster* ☒ Choose an existing cluster

demo [j-1J6CJR9AK8TS1](#) 

☐ Create a cluster 

Security groups ☒ Use default security groups 

☐ Choose security groups (vpc-0b59e8ceb4b208a34)

AWS service role*  

3. Connecting the GCP cloud SQL in Jupyter using Pyspark

After launching the jupyter notebook, we select the pyspark kernel amongst various jupyter kernels specified. We need to connect to database of GCP instance by installing “sql-connector-python” package and importing sql.connector. Then we use connect function in which we specify the username, password, public ip address of **assign3** instance and database name(“**dbassign3**”). Then we connect with the cursor and write the query. We also instantiate the spark using SparkSession and create object for both spark context and spark session. After that we retrieve all the tables with proper columns and data from gcp instance and convert it to spark dataframe and start ETL operations on those dataframes according to the problem statement given.