📘 Banking Data Analysis with Hadoop, MRJob, and Apache Hive on GCP Dataproc

🧪 Project Overview

This project demonstrates how to process and analyze large-scale banking data using distributed computing tools such as **Hadoop**, **MapReduce** (via Python's `mrjob`), and **Apache Hive**, all deployed on **Google Cloud Platform (GCP) Dataproc**.

We explored various tasks including:

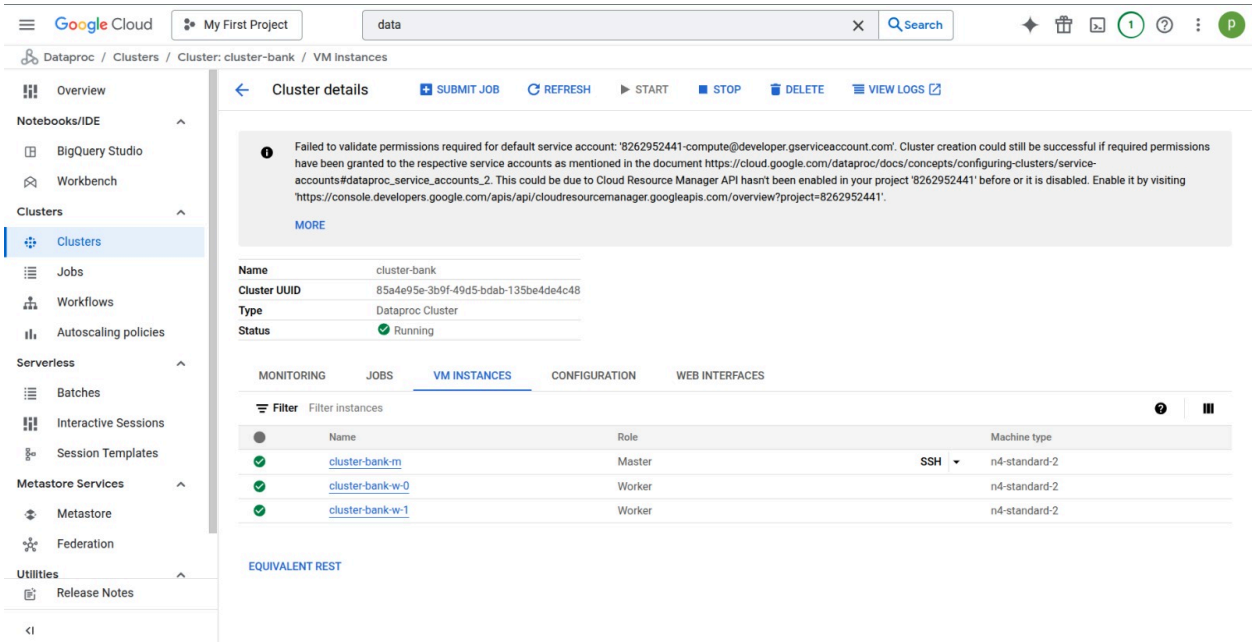
- Hadoop HDFS operations
- MapReduce jobs using `mrjob` in Python
- Apache Hive for SQL-based analysis

☁️ Step 1: GCP Dataproc Cluster Creation

🔹 Navigate to Dataproc

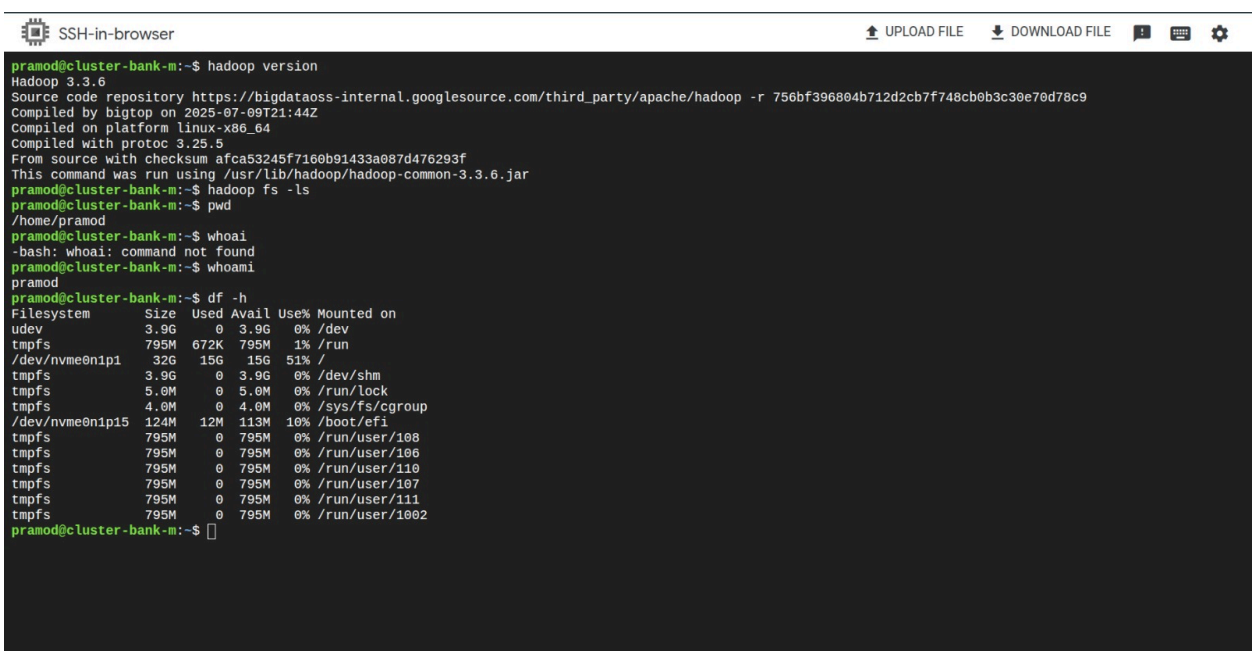
1. Go to GCP Console → Dataproc → Clusters
2. Click on "Create Cluster"
3. Choose the following main settings:

Setting	Value
Cluster Name	`bank-cluster`
Region	`us-central1`
Zone	`us-central1-a`
Image Version	`2.3.6-debian12`
Master Node Type	`n4-standard-2`
Worker Nodes	`2`
Worker Type	`n4-standard-2`
Optional Components	JUPYTER, HIVE_WEBHCAT



SSH into the Master Node

```
$ gcloud dataproc clusters list
$ gcloud compute ssh bank-cluster-m --zone=us-central1-a
```



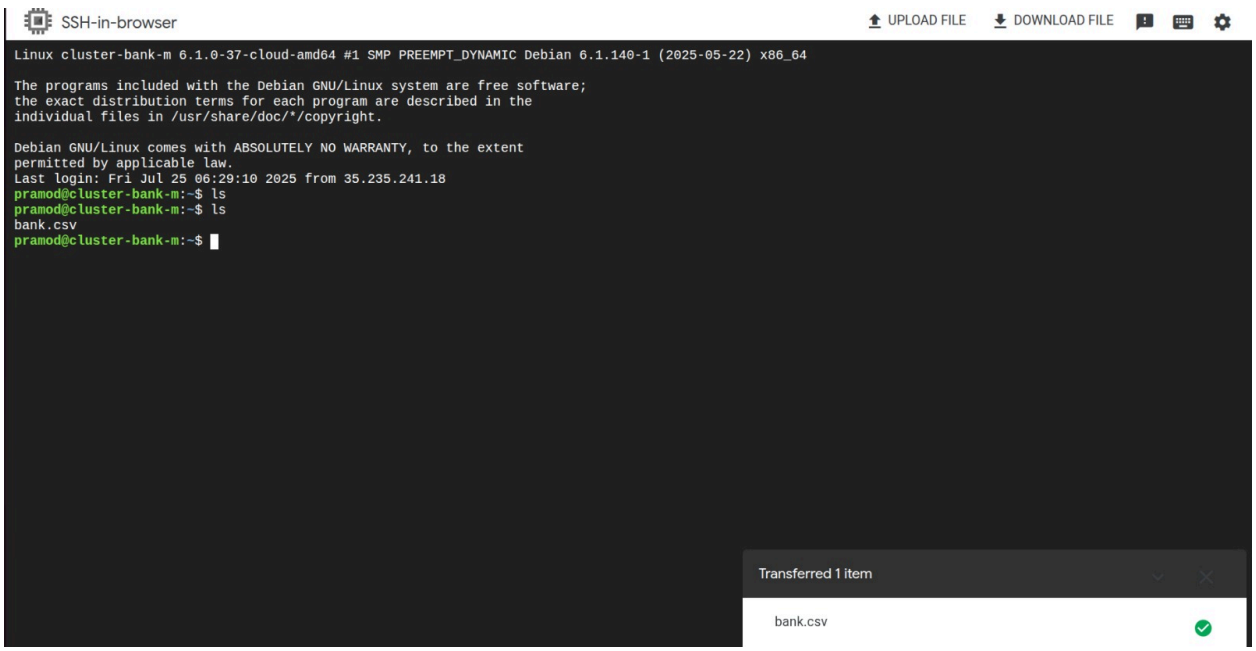
📂 Step 2: Upload Dataset to HDFS

- Place dataset on the master node

```
$ scp bank.csv pramod@<master-node-ip>:/home/pramod/
```

- Put dataset in HDFS

```
$ hadoop fs -put bank.csv /user/pramod/
$ hadoop fs -ls /user/pramod
```



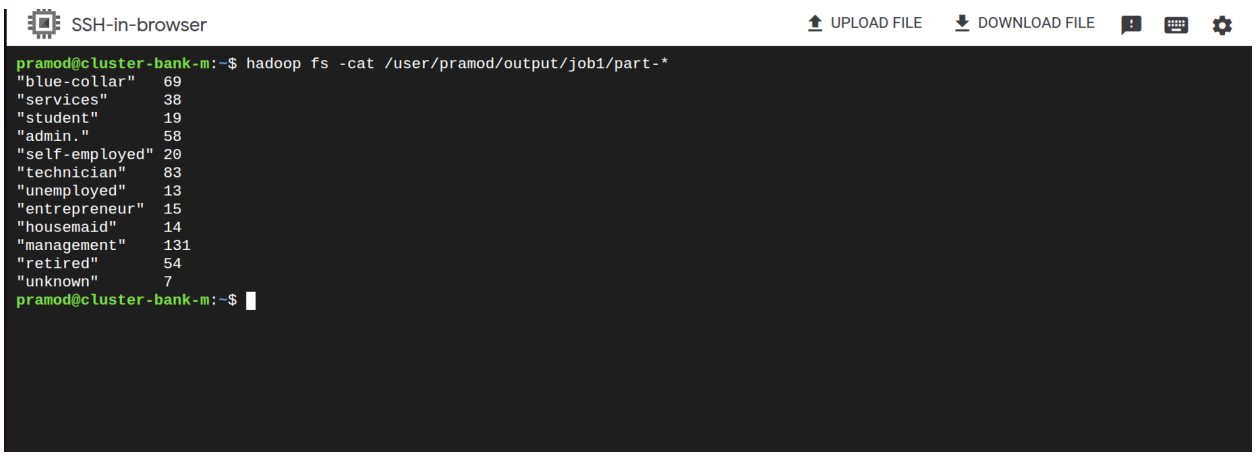
🛠️ Step 3: Run MapReduce Jobs using `mrjob`

Job	Directory	Python File	Output Dir	Command
1️⃣ Subscriptions by Job	`job1_subscription_by_job`	`job1_subscription_by_job.py`	`/user/pramod/output/job1`	`python3 job1_subscription_by_job.py bank.csv -r hadoop --output-dir hdfs:///user/pramod/output/job1`

Job	Directory	Python File	Output Dir	Command
2 Avg Balance by Education	job2_avg_balance_by_education	job2_avg_balance_by_education.py	/user/pramod/output/job2	python3 job2_avg_balance_by_education.py bank.csv -r hadoop --output-dir hdfs:///user/pramod/output/job2
3 Conversion Rate by Month	job3_conversion_rate_by_month	job3_conversion_rate_by_month.py	/user/pramod/output/job3	python3 job3_conversion_rate_by_month.py bank.csv -r hadoop --output-dir hdfs:///user/pramod/output/job3

- View Output Example

```
$ hadoop fs -cat /user/pramod/output/job1/part-*
```




```
$ hadoop fs -cat /user/pramod/output/job2/part-*
```

SSH-in-browser UPLOAD FILE DOWNLOAD FILE

```
pramod@cluster-bank-m:~$ hadoop fs -cat /user/pramod/output/job2/part-*
"secondary"     1196.81
"tertiary"      1775.42
"primary"       1411.54
"unknown"       1701.25
pramod@cluster-bank-m:~$
```

```
$ hadoop fs -cat /user/pramod/output/job3/part-*
```

SSH-in-browser UPLOAD FILE DOWNLOAD FILE

```
pramod@cluster-bank-m:~$ hadoop fs -cat /user/pramod/output/job3/part-*
"dec"   "45.00%"
"jun"   "10.36%"
"may"   "6.65%"
"nov"   "10.03%"
"feb"   "17.12%"
"jan"   "10.81%"
"jul"   "8.64%"
"sep"   "32.69%"
"apr"   "19.11%"
"aug"   "12.48%"
"mar"   "42.86%"
"oct"   "46.25%"
pramod@cluster-bank-m:~$
```

🐝 Step 4: Apache Hive Setup and Queries

Apache Hive is a distributed, fault-tolerant data warehouse system that enables analytics at a massive scale. Hive Metastore(HMS) provides a central repository of metadata that can easily be analyzed to make informed, data driven decisions, and therefore it is a critical component of many data lake architectures. Hive is built on top of Apache Hadoop and

supports storage on S3, adls, gs etc though hdfs. Hive allows users to read, write, and manage petabytes of data using SQL.

🔹 Launch HiveShell or Beeline

```
$ hive
```

OR

```
$ beeline
!connect jdbc:hive2://localhost:10000
```

🔹 Create Database and Table

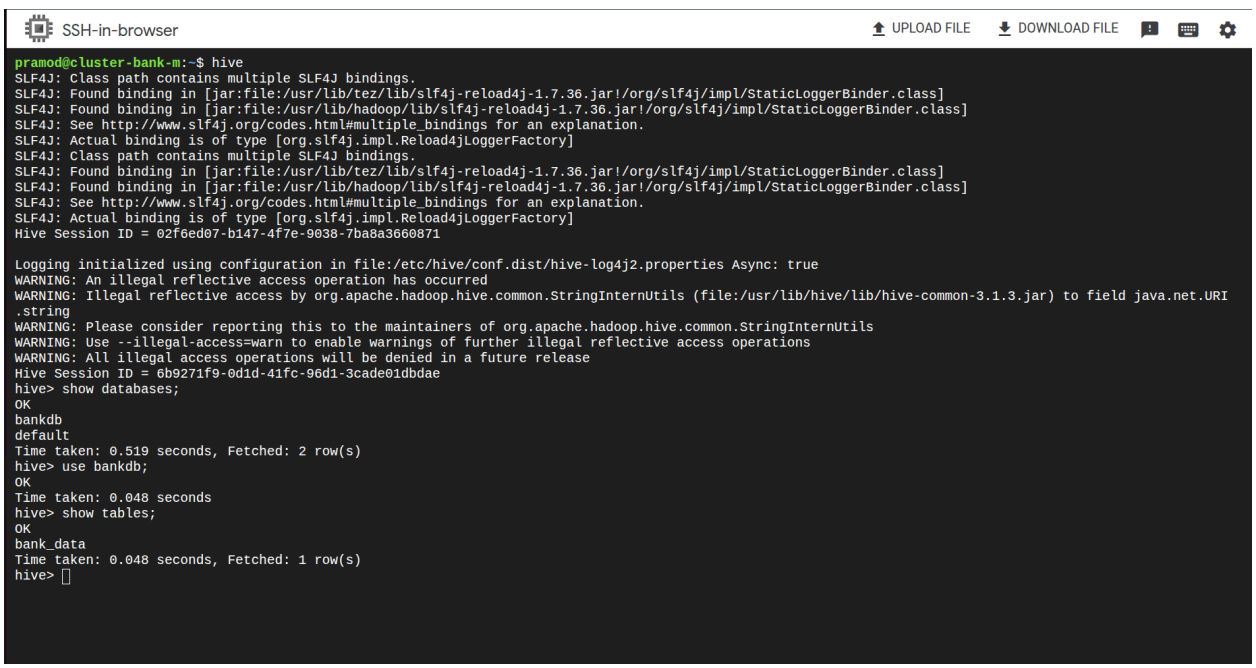
```
CREATE DATABASE IF NOT EXISTS bankdb;
USE bankdb;

CREATE EXTERNAL TABLE bank_data (
    age INT,
    job STRING,
    marital STRING,
    education STRING,
    default STRING,
    balance INT,
    housing STRING,
    loan STRING,
    contact STRING,
    day INT,
    month STRING,
    duration INT,
    campaign INT,
    pdays INT,
    previous INT,
    poutcome STRING,
    y STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
LOCATION '/user/pramod/';
```

🔹 Create External Table

Defines a table named bank_data that reads CSV data from HDFS without owning the file.

- **External Table** → Hive won't delete data if table is dropped
- **ROW FORMAT DELIMITED** → Data is plain text
- **FIELDS TERMINATED BY ','** → Columns separated by commas (CSV)
- **STORED AS TEXTFILE** → Stored as simple text
- **LOCATION '/user/pramod/'** → Data resides in HDFS at this path



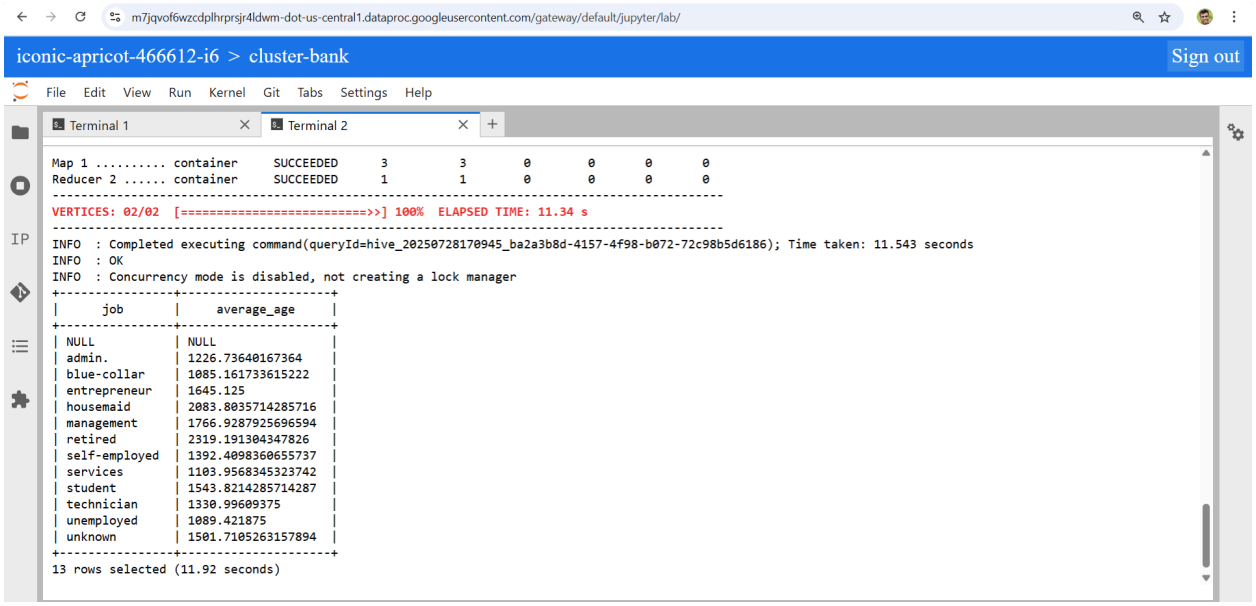

```
pramod@cluster-bank-m:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/tez/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Reload4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/lib/tez/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/hadoop/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Reload4jLoggerFactory]
Hive Session ID = 02f6ed07-b147-4f7e-9038-7ba8a3660871

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: true
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.hive.common.StringInternUtils (file:/usr/lib/hive/lib/hive-common-3.1.3.jar) to field java.net.URI
.string
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.hive.common.StringInternUtils
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Hive Session ID = 6b9271f9-0d1d-41fc-96d1-3cade01dbdae
hive> show databases;
OK
bankdb
default
Time taken: 0.519 seconds, Fetched: 2 row(s)
hive> use bankdb;
OK
Time taken: 0.048 seconds
hive> show tables;
OK
bank_data
Time taken: 0.048 seconds, Fetched: 1 row(s)
hive>
```

🔹 Hive Queries

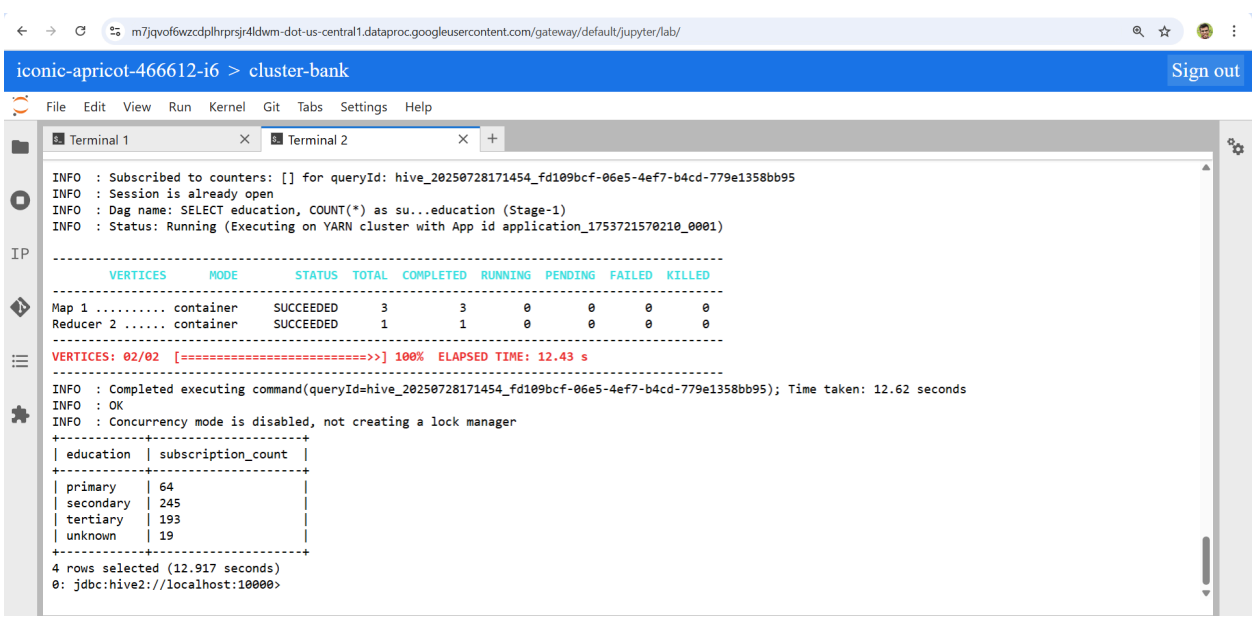
🔸 *Query 1: Average balance grouped by job*

```
SELECT job, AVG(balance) FROM bank_data GROUP BY job;
```



- *Query 2: Count of term deposit subscriptions by education*

```
SELECT education, COUNT(*) FROM bank_data WHERE y = 'yes' GROUP BY education;
```



- *Query 3: Month-wise campaign success*

```sql
SELECT month, COUNT(*) FROM bank_data WHERE y = 'yes' GROUP BY month;
```