HADOOP

Data Ingestion:

Question 1: Create a directory in HDFS and transfer the banking dataset from the localsystem to the HDFS directory.

Answer:

1. **Setting up hadoop in local environment:**

   1. Open a terminal on the system where Hadoop is installed. Or you can directly goto hadoop bin using the command line.

   2. I am running Hadoop in a Windows environment, we can use **start-dfs.cmd** and **start-yarn.cmd** to run the yarn demons.launch the necessary Hadoop services(NameNode, DataNode, ResourceManager, and NodeManager).

```
C:\Windows\System32>cd \

C:\>cd hadoop

C:\hadoop>cd sbin

C:\hadoop\sbin>start-dfs.cmd

C:\hadoop\sbin>start-yarn.cmd
starting yarn daemons

C:\hadoop\sbin>jps
2800 NameNode
15768 Jps
6552 NodeManager
2988 ResourceManager
8860 DataNode
```

2. **Create a Directory in HDFS:**

   Use the **hadoop fs -mkdir** command to create a directory in HDFS.

```
C:\hadoop\sbin>hadoop fs -mkdir -p /user/hadoop/input
```

3. **Transfer the Banking Dataset from Local to HDFS:**

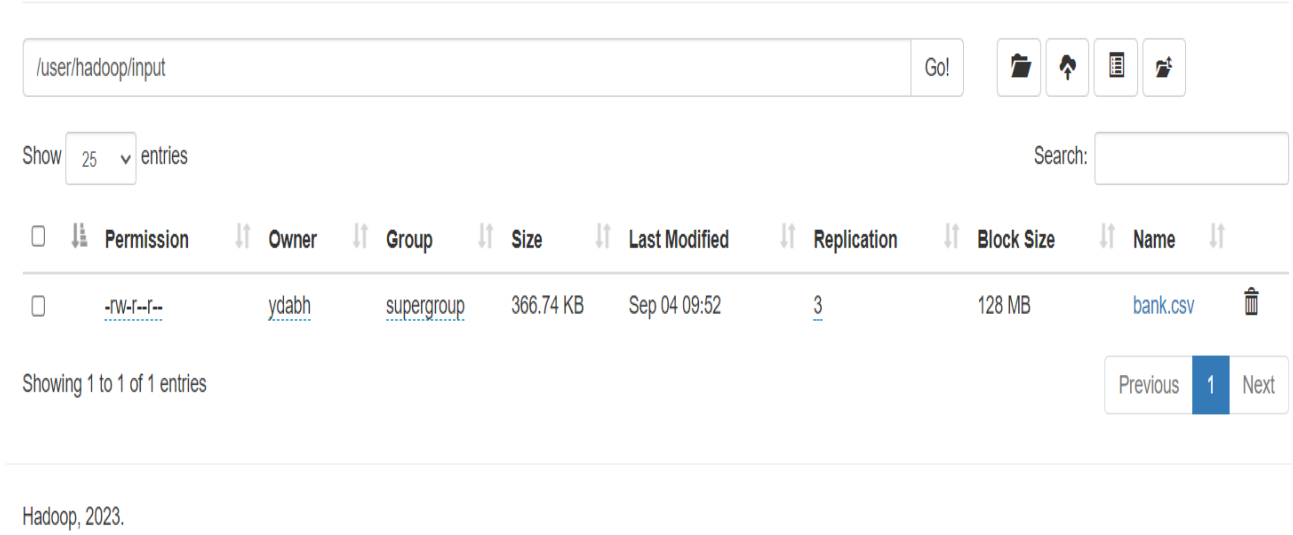
   Use the **hadoop fs -copyFromLocal** command to transfer the **bank.csv** datasetfrom your local file system to the newly created HDFS input directory.

```
C:\hadoop\sbin>hadoop fs -mkdir -p /user/hadoop/input

C:\hadoop\sbin>hadoop fs -copyFromLocal "C:\Users\ydabh\Downloads\Documents\bank.csv" /user/hadoop/input
```

4. **Check if data is loaded to the new directory:**

   We can visualise the data in ourlocalhost by accessing localhost**:9870** in our browser.



5. **Make the output directory for bank.csv:**
   `output /user/hadoop/output`: This command will specifies the output directory where the results of the MapReduce job will be stored. In this case, it's set to `/user/hadoop/output` on HDFS.

```
C:\hadoop\sbin>hadoop jar C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.3.6.jar -input /user/hadoop/input/bank.csv -output /user/hadoop/output
packageJobJar: [/C:/Users/ydabh/AppData/Local/Temp/hadoop-unjar8530752113659119477/] [] C:\Users\ydabh\AppData\Local\Temp\streamjob5768964188058727246.jar tmpDir=null
2024-09-04 10:03:36,810 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-04 10:03:36,964 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-04 10:03:37,419 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/ydabh/.staging/job_1725423105468_0002
2024-09-04 10:03:37,664 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-04 10:03:37,731 INFO mapreduce.JobSubmitter: number of splits:2
2024-09-04 10:03:37,840 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1725423105468_0002
2024-09-04 10:03:37,840 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-04 10:03:37,994 INFO conf.Configuration: resource-types.xml not found
2024-09-04 10:03:37,995 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-09-04 10:03:38,239 INFO impl.YarnClientImpl: Submitted application application_1725423105468_0002
2024-09-04 10:03:38,294 INFO mapreduce.Job: The url to track the job: http://Abhishek:8088/proxy/application_1725423105468_0002/
2024-09-04 10:03:38,296 INFO mapreduce.Job: Running job: job_1725423105468_0002
2024-09-04 10:03:47,444 INFO mapreduce.Job: Job job_1725423105468_0002 running in uber mode : false
2024-09-04 10:03:47,445 INFO mapreduce.Job:  map 0% reduce 0%
2024-09-04 10:03:53,598 INFO mapreduce.Job:  map 100% reduce 0%
2024-09-04 10:03:59,700 INFO mapreduce.Job:  map 100% reduce 100%
2024-09-04 10:03:59,705 INFO mapreduce.Job: Job job_1725423105468_0002 completed successfully
2024-09-04 10:03:59,811 INFO mapreduce.Job: Counters: 54
```

6. **Retrieve the results from the output directory:** use **hadoop fs -cat /user/hadoop/output/part-*** command to visualize and retrieve the output from the output directory.

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output/part-*
0       age,job,marital,education,default,balance,housing,loan,contact,day,month,duration,campaign,pdays,previous,poutcome,y
118     30,unemployed,married,primary,no,1787,no,no,cellular,19,oct,79,1,-1,0,unknown,no
200     33,services,married,secondary,no,4789,yes,yes,cellular,11,may,220,1,339,4,failure,no
286     35,management,single,tertiary,no,1350,yes,no,cellular,16,apr,185,1,330,1,failure,no
371     30,management,married,tertiary,no,1476,yes,yes,unknown,3,jun,199,4,-1,0,unknown,no
455     59,blue-collar,married,secondary,no,0,yes,no,unknown,5,may,226,1,-1,0,unknown,no
537     35,management,single,tertiary,no,747,no,no,cellular,23,feb,141,2,176,3,failure,no
620     36,self-employed,married,tertiary,no,307,yes,no,cellular,14,may,341,1,330,2,other,no
706     39,technician,married,secondary,no,147,yes,no,cellular,6,may,151,2,-1,0,unknown,no
790     41,entrepreneur,married,tertiary,no,221,yes,no,unknown,14,may,57,2,-1,0,unknown,no
874     43,services,married,primary,no,-88,yes,yes,cellular,17,apr,313,1,147,2,failure,no
957     39,services,married,secondary,no,9374,yes,no,unknown,20,may,273,1,-1,0,unknown,no
1040    43,admin.,married,secondary,no,264,yes,no,cellular,17,apr,113,2,-1,0,unknown,no
1121    36,technician,married,tertiary,no,1109,no,no,cellular,13,aug,328,2,-1,0,unknown,no
1205    20,student,single,secondary,no,502,no,no,cellular,30,apr,261,1,-1,0,unknown,yes
1286    31,blue-collar,married,secondary,no,360,yes,yes,cellular,29,jan,89,1,241,1,failure,no
1373    40,management,married,tertiary,no,194,no,yes,cellular,29,aug,189,2,-1,0,unknown,no
1457    56,technician,married,secondary,no,4073,no,no,cellular,27,aug,239,5,-1,0,unknown,no
1542    37,admin.,single,tertiary,no,2317,yes,no,cellular,20,apr,114,1,152,2,failure,no
1623    25,blue-collar,single,primary,no,-221,yes,no,unknown,23,may,250,1,-1,0,unknown,no
1706    31,services,married,secondary,no,132,no,no,cellular,7,jul,148,1,152,1,other,no
```

Data Transformation with MapReduce:

Question 1: Write a MapReduce program in Python that calculates the averageaccount balance for each job type.

Answer:

Step: 1. Write the MapReduce Python script and save as mapper.py and reducer.py.
Step: 2. Since I already created a directory and uploaded as the banking dataset bank.csv from local toHDFS.
Step: 3. I will directly run the mapper and reducer python file from my local drive by specifying the actual path of the python and mapper,reducer file and it will create the the output directory as output_new for the frist question.

```
C:\hadoop\sbin>hadoop jar C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.3.6.jar -files "file:///D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question1/mapper.py,file:///D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question1/reducer.py" -mapper "C:/Users/ydabh/AppData/Local/Programs/Python/Python312/python.exe D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question1/mapper.py" -reducer "C:/Users/ydabh/AppData/Local/Programs/Python/Python312/python.exe D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question1/reducer.py" -input /user/hadoop/input/bank.csv -output /user/hadoop/output_new
packageJobJar: [/C:/Users/ydabh/AppData/Local/Temp/hadoop-unjar5076951737920181131/] [] C:\Users\ydabh\AppData\Local\Temp\streamjob5894810305650506437.jar tmpDir=null
2024-09-04 21:53:14,320 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-04 21:53:14,476 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-04 21:53:14,945 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/ydabh/.staging/job_1725463689644_0001
2024-09-04 21:53:15,289 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-04 21:53:15,351 INFO mapreduce.JobSubmitter: number of splits:2
2024-09-04 21:53:15,461 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1725463689644_0001
2024-09-04 21:53:15,461 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-04 21:53:15,601 INFO conf.Configuration: resource-types.xml not found
2024-09-04 21:53:15,601 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-09-04 21:53:15,883 INFO impl.YarnClientImpl: Submitted application application_1725463689644_0001
2024-09-04 21:53:15,945 INFO mapreduce.Job: The url to track the job: http://Abhishek:8088/proxy/application_1725463689644_0001/
2024-09-04 21:53:15,945 INFO mapreduce.Job: Running job: job_1725463689644_0001
2024-09-04 21:53:25,104 INFO mapreduce.Job: Job job_1725463689644_0001 running in uber mode : false
2024-09-04 21:53:25,104 INFO mapreduce.Job:  map 0% reduce 0%
2024-09-04 21:53:32,158 INFO mapreduce.Job:  map 100% reduce 0%
2024-09-04 21:53:37,164 INFO mapreduce.Job:  map 100% reduce 100%
2024-09-04 21:53:37,164 INFO mapreduce.Job: Job job_1725463689644_0001 completed successfully
2024-09-04 21:53:37,242 INFO mapreduce.Job: Counters: 54
```

Output: use **hadoop fs -cat /user/hadoop/output_new/part-00000** command to visualize and retrieve the output from the output directory.

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output_new/part-00000
admin.  1226.73640167364
blue-collar     1085.161733615222
entrepreneur    1645.125
housemaid       2083.8035714285716
management      1766.9287925696594
retired 2319.191304347826
self-employed   1392.4098360655737
services        1103.9568345323742
student 1543.8214285714287
technician      1330.99609375
unemployed      1089.421875
unknown 1501.7105263157894
```

Step 4: Delete the output directory: **It is generally a good practice to delete the output directory before running a new MapReduce job in Hadoop to avoid overwriting issues.**

*To delete the output directory, use the command **- hadoop fs -rm -r**

```
C:\hadoop\sbin>hadoop fs -rm -r /user/hadoop/output_new
Deleted /user/hadoop/output_new

C:\hadoop\sbin>
```

Question 2.2. Write another MapReduce program that counts the number of individuals with and without a housing loan in each education category?

Answer:

Step: 1. Write the MapReduce Python script and save as mapper.py and reducer.py.
Step: 2. Since I already created a directory and uploaded as the banking dataset bank.csv from local toHDFS.
Step: 3. I will directly run the mapper and reducer python file from my local drive by specifying the actual path of the python and mapper, reducer file and it will create the output directory as output_new2 for the second question.

```
C:\hadoop\sbin>hadoop jar C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.3.6.jar -files "file:///D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question2/mapper.py,file:///D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question2/reducer.py" -mapper "C:/Users/ydabh/AppData/Local/Programs/Python/Python312/python.exe D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question2/mapper.py" -reducer "C:/Users/ydabh/AppData/Local/Programs/Python/Python312/python.exe D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question2/reducer.py" -input /user/hadoop/input/bank.csv -output /user/hadoop/output_new2
packageJobJar: [/C:/Users/ydabh/AppData/Local/Temp/hadoop-unjar7963279749335195935/] [] C:\Users\ydabh\AppData\Local\Temp\streamjob4657754924804990251.jar tmpDir=null
2024-09-05 11:15:02,273 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-05 11:15:02,460 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-05 11:15:03,030 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/ydabh/.staging/job_1725514957792_0001
2024-09-05 11:15:04,070 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-05 11:15:04,145 INFO mapreduce.JobSubmitter: number of splits:2
2024-09-05 11:15:04,275 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1725514957792_0001
2024-09-05 11:15:04,275 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-05 11:15:04,443 INFO conf.Configuration: resource-types.xml not found
2024-09-05 11:15:04,444 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-09-05 11:15:04,898 INFO impl.YarnClientImpl: Submitted application application_1725514957792_0001
2024-09-05 11:15:04,936 INFO mapreduce.Job: The url to track the job: http://Abhishek:8088/proxy/application_1725514957792_0001/
2024-09-05 11:15:04,938 INFO mapreduce.Job: Running job: job_1725514957792_0001
2024-09-05 11:15:15,121 INFO mapreduce.Job: Job job_1725514957792_0001 running in uber mode : false
2024-09-05 11:15:15,122 INFO mapreduce.Job:  map 0% reduce 0%
2024-09-05 11:15:22,293 INFO mapreduce.Job:  map 100% reduce 0%
2024-09-05 11:15:28,364 INFO mapreduce.Job:  map 100% reduce 100%
2024-09-05 11:15:29,385 INFO mapreduce.Job: Job job_1725514957792_0001 completed successfully
2024-09-05 11:15:29,482 INFO mapreduce.Job: Counters: 54
```

Output:

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output_new2/part-00000
primary 94      583
secondary       416     1889
tertiary        173     1176
unknown 7       179
```

Question 2.3 - Perform a MapReduce job to determine the number of clients contactedin each month and their subscription status to term deposits ('y' column).

Answers:

Step: 1. Write the MapReduce Python script and save as mapper.py and reducer.py.
Step: 2. Since I already created a directory and uploaded as the banking dataset bank.csv from local toHDFS.
Step: 3. I will directly run the mapper and reducer python file from my local drive by specifying the actual path of the python and mapper, reducer file and it will create the output directory as output_new3 for the third question.
Step: 4. Deleted the output_new2 directory from the hadoop.

```
C:\hadoop\sbin>hadoop fs -rm -r /user/hadoop/output_new2
Deleted /user/hadoop/output_new2

C:\hadoop\sbin>hadoop jar C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.3.6.jar -files "file:///D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question3/mapper.py,file:///D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question3/reducer.py" -mapper "C:/Users/ydabh/AppData/Local/Programs/Python/Python312/python.exe D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question3/mapper.py" -reducer "C:/Users/ydabh/AppData/Local/Programs/Python/Python312/python.exe D:/Hadoop/Hadoop_MapReduce_Code/Data_transformation_MapReduce_codes/Question3/reducer.py" -input /user/hadoop/input/bank.csv -output /user/hadoop/output_new3
packageJobJar: [/C:/Users/ydabh/AppData/Local/Temp/hadoop-unjar2516355492650968893/] [] C:\Users\ydabh\AppData\Local\Temp\streamjob6318159417741873940.jar tmpDir=null
2024-09-05 11:27:54,787 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-05 11:27:54,935 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-09-05 11:27:55,413 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/ydabh/.staging/job_1725514957792_0002
2024-09-05 11:27:55,772 INFO mapred.FileInputFormat: Total input files to process : 1
2024-09-05 11:27:55,837 INFO mapreduce.JobSubmitter: number of splits:2
2024-09-05 11:27:55,958 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1725514957792_0002
2024-09-05 11:27:55,959 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-09-05 11:27:56,098 INFO conf.Configuration: resource-types.xml not found
2024-09-05 11:27:56,099 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-09-05 11:27:56,157 INFO impl.YarnClientImpl: Submitted application application_1725514957792_0002
2024-09-05 11:27:56,194 INFO mapreduce.Job: The url to track the job: http://Abhishek:8088/proxy/application_1725514957792_0002/
2024-09-05 11:27:56,195 INFO mapreduce.Job: Running job: job_1725514957792_0002
2024-09-05 11:28:03,362 INFO mapreduce.Job: Job job_1725514957792_0002 running in uber mode : false
2024-09-05 11:28:03,365 INFO mapreduce.Job:  map 0% reduce 0%
2024-09-05 11:28:09,500 INFO mapreduce.Job:  map 100% reduce 0%
2024-09-05 11:28:15,569 INFO mapreduce.Job:  map 100% reduce 100%
```

Output:

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output_new3/part-00000
apr     56      236
aug     79      553
dec     8       11
feb     38      183
jan     16      131
jul     61      644
jun     55      475
mar     21      27
may     93      1304
nov     38      350
oct     37      42
sep     16      35

C:\hadoop\sbin>
```

3. Data Analysis with MapReduce:

Question 3.1. Analyze the average duration of contact (in seconds) per campaign outcome ('poutcome').

Answers:

Step: 1. Write the MapReduce Python script and save as mapper.py and reducer.py.
Step: 2. Since I already created a directory and uploaded as the banking dataset bank.csv from local toHDFS.
Step: 3. I will directly run the mapper and reducer python file from my local drive by specifying the actual path of the python and mapper, reducer file and it will create the output directory as output_new4 for the third question.
Step: 4. Deleted the output_new3 directory from the hadoop.

```
C:\hadoop\sbin>hadoop fs -rm -r /user/hadoop/output_new3
Deleted /user/hadoop/output_new3

C:\hadoop\sbin>hadoop jar C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.3.6.jar -files "file:///D:/Hadoop/Hadoop_MapReduce_Code/data_analysis_MapReduce_codes/Question1/mapper.py,file:///D:/Hadoop/Hadoop_MapReduce_Code/data_analysis_MapReduce_codes/Question1/reducer.py" -mapper "C:/Users/ydabh/AppData/Local/Programs/Python/Python312/python.exe D:/Hadoop/Hadoop_MapReduce_Code/data_analysis_MapReduce_codes/Question1/mapper.py" -reducer "C:/Users/ydabh/AppData/Local/Programs/Python/Python312/python.exe D:/Hadoop/Hadoop_MapReduce_Code/data_analysis_MapReduce_codes/Question1/reducer.py" -input /user/hadoop/input/bank.csv -output /user/hadoop/output_new4
```

Output:

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output_new4/part-00000
failure 254.38
other   273.83
success 338.64
unknown 262.10

C:\hadoop\sbin>
```

Summary:

The MapReduce job analyzed the dataset to calculate the average contact duration (inseconds) for each campaign outcome. The results are as follows:

- Failure: Average contact duration is 254.38 seconds.
- Other: Average contact duration is 273.83 seconds.
- Success: Average contact duration is 338.64 seconds.
- Unknown: Average contact duration is 262.10 seconds.

Question 3.2 - Examine the relationship between the age of clients and their balance, and present findings in a summarized form.

Answers:

Step: 1. Write the MapReduce Python script and save as mapper.py and reducer.py.
Step: 2. Since I already created a directory and uploaded as the banking dataset bank.csv from local toHDFS.
Step: 3. I will directly run the mapper and reducer python file from my local drive by specifying the actual path of the python and mapper, reducer file and it will create the output directory as output_new5 for the third question.
Step: 4. Deleted the output_new4 directory from the hadoop.

```
C:\hadoop\sbin>hadoop fs -rm -r /user/hadoop/output_new4
Deleted /user/hadoop/output_new4

C:\hadoop\sbin>hadoop jar C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.3.6.jar -files "file:///D:/Hadoop/Hadoop_M
apReduce_Code/data_analysis_MapReduce_codes/Question2/mapper.py,file:///D:/Hadoop/Hadoop_MapReduce_Code/data_analysis_Ma
pReduce_codes/Question2/reducer.py" -mapper "C:/Users/ydabh/AppData/Local/Programs/Python/Python312/python.exe D:/Hadoop
/Hadoop_MapReduce_Code/data_analysis_MapReduce_codes/Question2/mapper.py" -reducer "C:/Users/ydabh/AppData/Local/Program
s/Python/Python312/python.exe D:/Hadoop/Hadoop_MapReduce_Code/data_analysis_MapReduce_codes/Question2/reducer.py" -input
 /user/hadoop/input/bank.csv -output /user/hadoop/output_new5
```

Output:

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output_new5/part-00000
19      393.50
20      661.33
21      1774.29
22      1455.33
23      2117.95
24      634.62
25      1240.07
26      788.56
27      851.78
28      1025.10
29      1261.88
30      1113.03
31      1288.48
32      1256.55
33      1545.41
34      1111.54
35      1192.83
36      1226.89
37      1463.92
38      1718.99
39      1104.86
40      1399.51
41      1505.79
42      1612.36
43      1807.83
44      1836.55
45      1187.37
46      998.77
47      1363.05
48      1462.36
49      1591.11
50      1645.06
51      1528.57
52      782.29
53      1588.31
54      1656.66
55      1244.94
56      2120.14
57      1665.63
58      1755.08
59      1582.48
60      2964.57
61      2407.50
62      516.14
63      2286.38
64      1103.29
65      1638.17
66      3313.89
67      4149.40
68      11753.00
69      774.33
70      5084.57
71      3787.33
72      2526.00
73      525.83
74      1978.33
75      7046.50
76      1338.00
77      2405.17
78      318.00
79      4087.75
80      4183.50
81      1.00
83      380.50
84      639.00
86      1503.00
87      230.00

C:\hadoop\sbin>
```

Summary:

- Purpose: This analysis directly examines how the balance varies with each specific age.
- Implementation: The MapReduce job processes the data by outputting the age andbalance in the mapper and then calculates the average balance for each age in thereducer.
- Results: The results will give a detailed view of how balance varies with specificages, providing a more granular insight into the relationship between age and balance.

This approach avoids grouping by age bins and instead provides a direct average balance for each individual age.

Summary of Findings:

After running the MapReduce job, the output provides the average account balance for each specific age. Here's a summary of the key points:

- **Age-Specific Averages**: The output shows the average balance corresponding toeach client's age. For example, a 23-year-old might have an average balance of 2117.95, while a 25-year-old might have an average balance of 1240.05.
- **Trends**: If observed across ages, you might notice trends such as:Increase with Age: In some cases, there might be a gradual increase inaverage balance as age increases. Fluctuations: Certain ages might show higher or lower average balances dueto specific financial behaviors or life events.
- **Variability:** The average balances might fluctuate significantly across different ages, reflecting the diverse financial situations of clients at various life stages.

Conclusion:

The analysis reveals the average balance associated with each age, helping to identify any patterns or anomalies in financial behaviors across different age groups. This granular insight is valuable for financial institutions to tailor their products and services according to the needs of different age demographics.

HIVE

Data Ingestion and Table Creation

Question 1 - **Create a Hive database named bank_data.**

Answers:

Steps to Create a Hive Database:

1. Start Hive CLI
2. Create a new database named bank_data.

```
hive> CREATE DATABASE bank_data;
2024-09-06T09:35:46,242 INFO [main] org.
 dc733b09-e1c9-4b90-a69e-cbd7b25926bb
2024-09-06T09:35:46,242 INFO [main] org.
c9-4b90-a69e-cbd7b25926bb main
2024-09-06T09:35:46,366 WARN [dc733b09-e
- METASTORE_FILTER_HOOK will be ignored,
actory.
```

3. Switch to the newly created database:

```
hive> USE bank_data;
2024-09-06T09:36:59,763 INFO [main] org.apach
 dc733b09-e1c9-4b90-a69e-cbd7b25926bb
2024-09-06T09:36:59,763 INFO [main] org.apach
c9-4b90-a69e-cbd7b25926bb main
OK
```

Question 2 - Define and create a Hive table client_info with appropriate data types for the bank.csv dataset.

Answers:

- **Create the client_info table with appropriate data types based on the columns in the bank.csv file:**

```
hive> CREATE TABLE client_info(
    >     age INT,
    >     job STRING,
    >     marital STRING,
    >     education STRING,
    >     default STRING,
    >     balance FLOAT,
    >     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;
2024-09-06T09:42:24,540 INFO [main] org.a
 dc733b09-e1c9-4b90-a69e-cbd7b25926bb
2024-09-06T09:42:24,541 INFO [main] org.a
c9-4b90-a69e-cbd7b25926bb main
OK
```

Question 3 - Load the data from the bank.csv file into the client_info table.

Answers:

1. Load the banking data from hadoop directory to the client_info table.

```
hive> LOAD DATA INPATH '/user/hadoop/input/bank.csv' INTO TABLE client_info;
2024-09-06T09:48:55,210 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Us
 dc733b09-e1c9-4b90-a69e-cbd7b25926bb
2024-09-06T09:48:55,211 INFO [main] org.apache.hadoop.hive.ql.session.Session
c9-4b90-a69e-cbd7b25926bb main
Loading data to table bank_data.client_info
OK
Time taken: 0.797 seconds
```

2. Query the client_info table to verify that the data has been loaded correctly. Here, we can see the asked ten rows in the output. So, it is verified that the data is loaded successfully.

```
hive> SELECT * FROM client_info LIMIT 10 ;
2024-09-06T09:50:58,001 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id:
 dc733b09-e1c9-4b90-a69e-cbd7b25926bb
2024-09-06T09:50:58,001 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to dc733b09-e1
c9-4b90-a69e-cbd7b25926bb main
2024-09-06T09:50:59,185 INFO [dc733b09-e1c9-4b90-a69e-cbd7b25926bb main] org.apache.hadoop.hive.common.FileUtils - Creat
ing directory if it doesn't exist: hdfs://localhost:9000/tmp/hive/ydabh/dc733b09-e1c9-4b90-a69e-cbd7b25926bb/hive_2024-0
9-06_09-50-58_019_4142651933164400544-1/-mr-10001/.hive-staging_hive_2024-09-06_09-50-58_019_4142651933164400544-1
2024-09-06T09:50:59,315 INFO [dc733b09-e1c9-4b90-a69e-cbd7b25926bb main] org.apache.hadoop.conf.Configuration.deprecatio
n - mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
OK
2024-09-06T09:50:59,327 INFO [dc733b09-e1c9-4b90-a69e-cbd7b25926bb main] org.apache.hadoop.conf.Configuration.deprecatio
n - mapred.input.dir is deprecated. Instead, use mapreduce.input.fileinputformat.inputdir
NULL    job     marital education       default NULL    housing loan    contact NULL    month   NULL    NULL    NULL   N
ULL     poutcome        y
30      unemployed      married primary no      1787.0  no      no      cellular        19      oct     79      1      -
1       0       unknown no
33      services        married secondary       no      4789.0  yes     yes     cellular        11      may     220    1
339     4       failure no
35      management      single  tertiary        no      1350.0  yes     no      cellular        16      apr     185    1
330     1       failure no
30      management      married tertiary        no      1476.0  yes     yes     unknown 3       jun     199     4      -
1       0       unknown no
59      blue-collar     married secondary       no      0.0     yes     no      unknown 5       may     226     1      -
1       0       unknown no
35      management      single  tertiary        no      747.0   no      no      cellular        23      feb     141    2
176     3       failure no
```

Basic Data Exploration

Question 1: Write a HiveQL query to count the total number of clients in the dataset.

Answers:-

Steps:

1. **FROM client_info**: The query selects data from the client_infotable, whichcontains the records of all clients.

2. **COUNT (*) AS total_clients**: The COUNT(*)function counts the total number of rows in the client_infotable. The result is labelled as total_clientsfor clarity.

```
hive> SELECT COUNT(*) AS total_clients FROM client_info;
2024-09-06T09:56:24,920 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Usi
 dc733b09-e1c9-4b90-a69e-cbd7b25926bb
2024-09-06T09:56:24,920 INFO [main] org.apache.hadoop.hive.ql.session.SessionS
c9-4b90-a69e-cbd7b25926bb main
```

Output:

```
Ended Job = job_1725595040841_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU:
Total MapReduce CPU Time Spent: 3 seconds 671 msec
OK
4522
```

Summary of the Results:

- **Total Number of Clients**: The query returns a single number representing the total number of clients in the dataset. This number gives you a quick overview of the dataset size, indicating how many client records are available for analysis. So, here we can see that the total number of clients is 4522.

Question 2: Display the first 10 rows of the dataset.

Answers:-

1. **FROM client_info**: The query selects data from the client_infotable, which contains all the records in the dataset.

2. ****SELECT ***:** The query selects all columns (*) from the client_info table. This means that every piece of information available for each client will be included in the result.

3. **LIMIT 10**: The query restricts the output to only the first 10 rows of the dataset. This is useful for quickly examining a sample of the data without retrieving the entire dataset.

```
hive> SELECT * FROM client_info LIMIT 10 ;
2024-09-06T09:50:58,001 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id:
 dc733b09-e1c9-4b90-a69e-cbd7b25926bb
2024-09-06T09:50:58,001 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to dc733b09-e1
c9-4b90-a69e-cbd7b25926bb main
2024-09-06T09:50:59,185 INFO [dc733b09-e1c9-4b90-a69e-cbd7b25926bb main] org.apache.hadoop.hive.common.FileUtils - Creat
ing directory if it doesn't exist: hdfs://localhost:9000/tmp/hive/ydabh/dc733b09-e1c9-4b90-a69e-cbd7b25926bb/hive_2024-0
9-06_09-50-58_019_4142651933164400544-1/-mr-10001/.hive-staging_hive_2024-09-06_09-50-58_019_4142651933164400544-1
2024-09-06T09:50:59,315 INFO [dc733b09-e1c9-4b90-a69e-cbd7b25926bb main] org.apache.hadoop.conf.Configuration.deprecatio
n - mapred.task.is.map is deprecated. Instead, use mapreduce.task.ismap
OK
2024-09-06T09:50:59,327 INFO [dc733b09-e1c9-4b90-a69e-cbd7b25926bb main] org.apache.hadoop.conf.Configuration.deprecatio
n - mapred.input.dir is deprecated. Instead, use mapreduce.input.fileinputformat.inputdir
NULL    job     marital education       default NULL    housing loan    contact NULL    month   NULL    NULL    NULL   N
ULL     poutcome        y
30      unemployed      married primary no      1787.0  no      no      cellular        19      oct     79      1      -
1       0       unknown no
33      services        married secondary       no      4789.0  yes     yes     cellular        11      may     220    1
339     4       failure no
35      management      single  tertiary        no      1350.0  yes     no      cellular        16      apr     185    1
330     1       failure no
30      management      married tertiary        no      1476.0  yes     yes     unknown 3       jun     199     4      -
1       0       unknown no
59      blue-collar     married secondary       no      0.0     yes     no      unknown 5       may     226     1      -
1       0       unknown no
35      management      single  tertiary        no      747.0   no      no      cellular        23      feb     141    2
176     3       failure no
```

Summary of the Results:

- **First 10 Rows of the Dataset:** Here, we can see that the output displays all columnsand their values for the first 10 clients in the client_info table. These rows represent a small sample of the overall dataset, providing a snapshot of the data structure and contents.

Data Filtering and Sorting:

Question 1: Retrieve all records of clients who are married and have a personal loan.

Answer:

Steps:

1. **FROM client_info**: The query selects data from the client_info table, which includes various details about the clients such as their marital status, loan status, and other attributes.
2. ****SELECT ***:** The query selects all columns (*) from the client_info table. This means that every piece of information available for each client will be included in the result, such as age, job, balance, etc.
3. **WHERE marital = 'married' AND loan = 'yes'**: The query filters the data to includeonly those clients who meet both of the following conditions:

- **marital = 'married'**: The client must be married.
- **loan = 'yes'**: The client must have a personal loan.

```
hive> SELECT * FROM client_info WHERE marital ='married' AND loan = 'yes';
2024-09-06T10:03:42,240 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: dc733b09-e1c9-4b90-a69e-cb
b
2024-09-06T10:03:42,240 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to dc733b09-e1c9-4b90-a69e-cbd7b25926bb m
2024-09-06T10:03:42,470 INFO [dc733b09-e1c9-4b90-a69e-cbd7b25926bb main] org.apache.hadoop.hive.common.FileUtils - Creating directory if it doesn't
dfs://localhost:9000/tmp/hive/ydabh/dc733b09-e1c9-4b90-a69e-cbd7b25926bb/hive_2024-09-06_10-03-42_256_1916351001245885362-1/-mr-10001/.hive-staging
4-09-06_10-03-42_256_1916351001245885362-1
OK
33      services        married secondary       no      4789.0  yes     yes     cellular        11      may     220     1339    4       failure no
30      management      married tertiary        no      1476.0  yes     yes     unknown 3       jun     199     4       -1      0       unknown no
43      services        married primary no      -88.0   yes     yes     cellular        17      apr     313     1       147     2       failure no
31      blue-collar     married secondary       no      360.0   yes     yes     cellular        29      jan     89      1241    1       failure no
40      management      married tertiary        no      194.0   no      yes     cellular        29      aug     189     2-1     0       unknown no
56      self-employed   married secondary       no      784.0   no      yes     cellular        30      jul     149     2-1     0       unknown no
53      admin.  married secondary       no      105.0   no      yes     cellular        21      aug     74      2       -1      0       unknown no
57      management      married secondary       no      82.0    no      yes     telephone       4       feb     140     1-1     0       unknown no
41      blue-collar     married primary no      -516.0  no      yes     telephone       8       jul     554     3       -1      0       unknown no
41      management      married secondary       no      0.0     no      yes     cellular        7       jul     630     3-1     0       unknown no
37      blue-collar     married secondary       no      427.0   yes     yes     unknown 9       jun     371     3       -1      0       unknown no
32      self-employed   married secondary       no      217.0   yes     yes     cellular        15      jul     317     5-1     0       unknown no
36      blue-collar     married secondary       no      -231.0  no      yes     cellular        15      jul     779     2-1     0       unknown no
42      admin.  married secondary       no      323.0   yes     yes     unknown 8       may     280     2       -1      0unknown no
35      management      married tertiary        no      106.0   no      yes     cellular        11      aug     588     2-1     0       unknown no
56      retired married primary no      1906.0  no      yes     unknown 19      jun     45      9       -1      0       unknown  no
43      services        married secondary       no      978.0   yes     yes     unknown 26      may     82      2       -1      0       unknown no
43      admin.  married secondary       no      -465.0  yes     yes     cellular        23      jul     166     1       -1      0       unknown no
44      admin.  married secondary       no      5181.0  yes     yes     cellular        31      jul     18      7       -1      0       unknown no
49      blue-collar     married primary no      0.0     yes     yes     telephone       23      jul     97      6       -1      0       unknown no
38      services        married secondary       no      1.0     no      yes     cellular        21      nov     152     2-1     0       unknown no
42      technician      married secondary       no      2030.0  yes     yes     cellular        9       jul     196     1-1     0       unknown no
49      blue-collar     married primary no      305.0   yes     yes     telephone       10      jul     834     10      -1      0       unknown no
35      technician      married tertiary        no      0.0     yes     yes     cellular        23      sep     112     162     6       other   no
27      admin.  married secondary       no      -247.0  yes     yes     unknown 4       jun     344     2       -1      0unknown no
43      technician      married secondary       no      0.0     no      yes     cellular        8       may     9       2172    5       failure no
55      blue-collar     married secondary       no      989.0   yes     yes     unknown 23      may     246     4       -1      0       unknown no
34      management      married tertiary        no      415.0   no      yes     cellular        23      jul     361     2-1     0       unknown no
54      housemaid       married secondary       no      209.0   yes     yes     cellular        25      jul     97      1-1     0       unknown no
53      entrepreneur    married tertiary        no      624.0   no      yes     cellular        21      jul     180     4-1     0       unknown no
```

Summary of result:

- This query filters the data to retrieve records of clients who are married and have a personal loan.
- **Output**: We can see the output is a list of all married clients who have taken a personal loan, including all columns of data.

Question 2: List the top 10 clients with the highest balance, displaying their job, marital status, and balance.

Answers:

1. **FROM client_info**: The query selects data from the client_info table, which contains information about clients, including their job, marital status, and account balance.
2. **SELECT job, marital, balance**: The query specifies that it wants to retrieve the job, marital, and balancecolumns from the client_info table.
3. **ORDER BY balance DESC**: The query orders the results by the balance column indescending order (DESC), meaning that the clients with the highest balances will appear first.
4. **LIMIT 10**: The query limits the results to the top 10 records. This means only the 10clients with the highest balances will be shown.

```
hive> SELECT job,marital,balance FROM client_info ORDER BY balance DESC LIMIT 10;
```

Output:

```
Total MapReduce CPU Time Spent: 3 seconds 327 m
OK
retired married 71188.0
entrepreneur    married 42045.0
technician      single  27733.0
management      married 27359.0
technician      married 27069.0
housemaid       single  26965.0
retired married 26452.0
services        married 26394.0
management      divorced        26306.0
retired single  25824.0
Time taken: 21.859 seconds, Fetched: 10 row(s)
```

Summary of the Results:

- **Top 10 Clients by Balance**: The output of this query will display the job, marital status, and balance of the 10 clients who have the highest account balances. So, here we see that retired clients are generally married and have the highest balance.

Data Aggregation and Grouping:

Question 1: Calculate the average age of clients for each job category.

Answers:-

The query retrieves data from the client_info table, which contains information aboutclients, including their job and age.

GROUP BY job: The query groups the data by the jobcolumn. This means that the data will be aggregated separately for each unique job category.

AVG(age) AS average_age: For each job category, the query calculates the average age of the clients using the AVG(age) function. The result is stored in a column named average_age.

SELECT job, AVG(age) AS average_age: Finally, the query selects and displays the jobcategory (job) alongside the calculated average age (average_age) for each group.

```
hive> SELECT job, AVG(age) AS average_age FROM client_info GROUP BY job;
```

Output:

```
OK
admin.  39.68200836820084
blue-collar     40.15644820295983
entrepreneur    42.011904761904 76
housemaid       47.339285714285715
job     NULL
management      40.54076367389061
retired 61.869565217391305
self-employed   41.45355191256831
services        38.57074340527578
student 26.821428571428573
technician      39.470052083333336
unemployed      40.90625
unknown 48.10526315789474
```

Summary of the Results:

- **Average Age by Job Category**: The query outputs the average age of clients for each job category. This provides insight into the typical age of clients in different professions. Here, we can see that the average age of the majority of clients fordifferent job categories is between 35 to 45.

Question:2: Find the total number of clients for each education level who have defaultedon credit.

Answers:-

FROM client_info: The query starts by selecting data from the client_info table, which contains information about the clients, including their education level and whether they have defaulted on credit.

WHERE default = 'yes': The query filters the data to include only those clients who have defaulted on credit. The defaultcolumn is checked, and only records where default ='yes' are selected. This ensures that the query is only counting clients who have actuallydefaulted.

GROUP BY education and default: The query then groups the filtered data by the `education` column and default column.

COUNT(*) AS total_defaulted_clients: For each education level, the query counts the number of clients who have defaulted on credit using **`COUNT(*).`** The result is stored in a column named total_defaulted_clients.

SELECT education, default, COUNT(*) AS total_defaulted_clients: Finally, the queryselects and displays the education level alongside the total number of clients who have defaulted in that education level.

```
hive> SELECT education,default, COUNT(*) AS total_default_client FROM client_info WHERE default = 'yes' GROUP BY educati
on, default;
```

Output:

```
OK
primary yes     10
secondary       yes     46
tertiary        yes     17
unknown yes     3
```

Summary of result:

- This query finds the total number of clients for each education level who have defaulted on credit. Here, we can see that the majority of clients who defaulted have a secondary education level.

Complex Queries for Insights:

Question 1: Identify the top 5 job categories with the highest average balance and the percentage of clients in each of these job categories who have subscribed to a term deposit.

Answer:

1. **AVG(balance)**: Calculates the average balance for each job category.
2. **SUM(CASE WHEN y = 'yes' THEN 1 ELSE 0 END)**: Counts the number of clients who subscribed to a term deposit.
3. **(subscribed_clients / total_clients) * 100**: Calculates the percentage of clients in each job category who subscribed to a term deposit.
4. **ORDER BY avg_balance DESC**: Sorts the job categories in descending order by average balance.
5. **LIMIT 5**: Limits the result to the top 5 job categories.

```
hive> WITH job_balance AS (
    >     SELECT
    >         job,
    >         AVG(balance) AS avg_balance,
    >         COUNT(*) AS total_clients,
    >         SUM(CASE WHEN y = 'yes' THEN 1 ELSE 0 END) AS subscribed_clients
    >     FROM client_info
    >     GROUP BY job
    > )
    > SELECT
    >     job,
    >     avg_balance,
    >     (subscribed_clients / total_clients) * 100 AS percentage_subscribed
    > FROM job_balance
    > ORDER BY avg_balance DESC
    > LIMIT 5;
```

Output:

```
OK
retired 2319.191304347826       23.47826086956522
housemaid       2083.8035714285716      12.5
management      1766.9287925696594      13.519091847265221
entrepreneur    1645.125        8.928571428571429
student 1543.8214285714287      22.619047619047619
```

Summary of the Results:

- **Top 5 Job Categories by Average Balance**: The query identifies the five job categories with the highest average balances. These are the clients with the mostsignificant average account balances across all job types. So, here retired and housemaids have the highest average balance.
- **Subscription Percentage**: The query then calculates how successful the current campaign was in converting clients in these high-balance job categories into term deposit subscribers. The percentage_subscribed indicates the effectiveness of the campaign for each job category. E.g. Here it was not very successful for entrepreneurs because the subscribed_clients was only 8.928 percent.

Question 2: Determine the month with the highest number of contacts and the success rate of the campaign in that month (percentage of clients who subscribed to a term deposit).

Answer:

- **COUNT(*):** Counts the total number of contacts made in each month.
- **SUM(CASE WHEN y = 'yes' THEN 1 ELSE 0 END):** Counts the number of clients who subscribed to a term deposit in each month.
- **(subscribed_clients / total_contacts) * 100:** Calculates the success rate of the campaign in each month as a percentage.
- **ORDER BY total_contacts DESC:** Orders the result by the number of contacts in descending order.
- **LIMIT 1:** Limits the result to the month with the highest number of contacts.

```
hive> WITH month_contacts AS (
    >     SELECT
    >         month,
    >         COUNT(*) AS total_contacts,
    >         SUM(CASE WHEN y = 'yes' THEN 1 ELSE 0 END) AS subscribed_clients
    >     FROM client_info
    >     GROUP BY month
    > )
    > SELECT
    >     month,
    >     total_contacts,
    >     (subscribed_clients / total_contacts) * 100 AS success_rate
    > FROM month_contacts
    > ORDER BY total_contacts DESC
    > LIMIT 1;
```

Output:

```
OK
may     1398    6.652360515021459
```

Summary:

- This query determines the month with the highest number of client contacts and calculates the success rate of the campaign in that month (the percentage of clients who subscribed to a term deposit).
- **Output**: May, is the month with the highest contacts, the total number of contacts are 1398, and the success rate (percentage of clients who subscribed to a term deposit) during that month is 6.652.

Correlation Analysis:

Question 6: Calculate the correlation between age and balance for the clients.

Answer:

The CORR(age, balance) function calculates the Pearson correlation coefficient betweenthe ageand balancecolumns.

This coefficient will range from -1 to 1, where:

- **1** indicates a perfect positive correlation,
- **-1** indicates a perfect negative correlation, and
- **0** indicates no correlation.

```
hive> SELECT
    >     CORR(age, balance) AS age_balance_correlation
    > FROM client_info;
```

Output:

```
OK
0.08382014224477742
```

Summary Of Result:

- This result gives you an idea of how strongly the age of clients is related to their account balance. Here, Since the output is 0.0838 i.e. between 0 and 1 so, we can say that age isslightly related to balance.

Trend Analysis:

Question 1: Analyse the year-over-year trend in the number of clients contacted:

***There is no data in the bank_data.csv dataset which represents the year. But let's say the first four characters from the `month` column represent the year (e.g., `2023` from `2023-Jan`).**

Answers:-

- **SUBSTR(month, 1, 4):** Extracts the first four characters from the month column to represent the year.
- **COUNT(*):** Counts the number of clients contacted in each year.
- **GROUP BY SUBSTR(month, 1, 4):** Groups the data by the extracted year.
- **ORDER BY year:** Sorts the results by year to show the year-over-year trend.

```
hive> WITH year_contacts AS (
    >     SELECT
    >         SUBSTR(month, 1, 4) AS year,
    >         COUNT(*) AS total_contacts
    >     FROM client_info
    >     GROUP BY SUBSTR(month, 1, 4)
    > )
    > SELECT
    >     year,
    >     total_contacts
    > FROM year_contacts
    > ORDER BY year;
```

Output:

```
OK
apr     293
aug     633
dec     20
feb     222
jan     148
jul     706
jun     531
mar     49
may     1398
mont    1
nov     389
oct     80
sep     52
```

Summary of result: This output will give us a clear view of the year-over-year trend in the number of client contacts. Here, we did not have year information in the bank_data.csv file. So, we could not get the desired result

Anomaly Detection:

Question 1: Identify any unusual patterns in the average yearly balance across different education levels.

Answer:

- **SUBSTR**(month, 1, 4): Extracts the year from the month column.
- **AVG**(balance): Calculates the average balance for each combination of year and education level.
- **GROUP BY SUBSTR**(month, 1, 4), education: Groups the data by year and education to compute the average balance for each group.
- **ORDER BY** year, education: Orders the results by year and education to help you see trends over time.

```
hive> WITH yearly_balance_education AS (
    >     SELECT
    >         SUBSTR(month, 1, 4) AS year,
    >         education,
    >         AVG(balance) AS avg_balance
    >     FROM client_info
    >     GROUP BY SUBSTR(month, 1, 4), education
    > )
    > SELECT
    >     year,
    >     education,
    >     avg_balance
    > FROM yearly_balance_education
    > ORDER BY year, education;
```

Output:

```
OK
apr     primary 2646.8888888888887
apr     secondary       1208.3764705882354
apr     tertiary        2204.609756097561
apr     unknown 864.4
aug     primary 1214.4237288135594
aug     secondary       1361.3705035971223
aug     tertiary        1525.5587188612099
aug     unknown 2796.133333333333
dec     primary 980.0
dec     secondary       2885.5555555555557
dec     tertiary        2384.0
dec     unknown 12660.5
feb     primary 980.4705882352941
feb     secondary       1040.009900990099
feb     tertiary        1766.6538461538462
feb     unknown 1850.111111111111
jan     primary 1189.3684210526317
jan     secondary       885.3055555555555
jan     tertiary        1106.6888888888889
jan     unknown 696.75
jul     primary 866.542372881356
jul     secondary       741.2950391644908
jul     tertiary        860.8222222222222
jul     unknown 649.92
jun     primary 1845.1238938053098
jun     secondary       1453.1106382978724
jun     tertiary        2647.0340136054424
jun     unknown 1714.75
mar     primary 1238.0
mar     secondary       1717.6363636363637
mar     tertiary        2812.4444444444443
mar     unknown 1674.5
may     primary 947.4026548672566
may     secondary       966.1274752475248
may     tertiary        1475.9901639344262
may     unknown 1673.1186440677966
mont    education       NULL
nov     primary 2163.0
nov     secondary       2504.7272727272725
```

Summary of the Results:

- **year:** The year extracted from the month column.
- **education:** The education level of the clients.
- **z_score:** The z-score indicates how the average yearly balance for a specificeducation level compares to the overall average for that year. A positive
  z-score means the average balance is above the overall average, while a negative z-score indicates it is below the overall average.

Output Interpretation:

- **Positive Z-Scores:** Indicate that the clients with a specific education level havea higher-than-average balance compared to others in the same year.
- **Negative Z-Scores:** Indicate that the clients with a specific education level havea lower-than-average balance compared to others in the same year.

Advanced Analysis:

Question 1: Analyze the impact of previous campaign outcomes (poutcome) on the current campaign's success. Calculate the subscription rate (to term deposits) for each poutcome category.

Answer:

- **COUNT(*)**: Counts the total number of clients for each poutcome category.
- **SUM(CASE WHEN y = 'yes' THEN 1 ELSE 0 END)**: Counts the number of clients who subscribed to a term deposit in each poutcome category.
- **(subscribed_clients / total_clients) * 100**: Calculates the subscription rate (as a percentage) for each poutcome category.
- **ORDER BY subscription_rate DESC**: Orders the results by the subscription rate in descending order, so you can see which poutcome categories had the highest success rates.

```
hive> WITH poutcome_analysis AS (
    >     SELECT
    >         poutcome,
    >         COUNT(*) AS total_clients,
    >         SUM(CASE WHEN y = 'yes' THEN 1 ELSE 0 END) AS subscribed_clients
    >     FROM client_info
    >     GROUP BY poutcome
    > )
    > SELECT
    >     poutcome,
    >     total_clients,
    >     subscribed_clients,
    >     (subscribed_clients / total_clients) * 100 AS subscription_rate
    > FROM poutcome_analysis
    > ORDER BY subscription_rate DESC;
```

Output:

```
OK
success 129     83      64.341085271317840
other   197     38      19.289340101522843
failure 490     63      12.857142857142856
unknown 3705    337     9.095816464237517
poutcome        1       0       0.0
```

Summary of the Results:

- **poutcome**: Represents different outcomes of the previous marketing campaigns,such as success or failure.
- **total_clients**: The total number of clients associated with each outcome.
- **subscribed_clients**: The number of clients who subscribed to the term depositwithin each `poutcome` category.
- **subscription_rate**: The percentage of clients who subscribed to the termdeposit, calculated per `poutcome` category.

The results of this query help to understand the effectiveness of previous marketing outcomes (`poutcome`) by showing how many clients subscribed to the term deposit in each category. By sorting the results by `subscription_rate`, it highlights which previous outcomes were most successful in leading to subscriptions.

Interpretation:

- **High Subscription Rate:** A high subscription rate in poutcome category (e.g.,`success`) indicates that clients who had a positive outcome in the previous campaign are more likely to subscribe again in the current campaign.
- **Low Subscription Rate:** A low subscription rate (e.g., `failure` or `unknown`)suggests that clients with unsuccessful or unknown outcomes in the previous campaign are less likely to subscribe in the current campaign.

Here, we can see that the subscription_rate is very high i.e. 64.34 for success poutcome. So, we can say that clients who had a positive outcome in the previous campaign are more likely to subscribe again in the current campaign.

HIVE

Question 2: Compare the average contact duration for clients who subscribed and who did not subscribe to a term deposit.

Answer:

- **AVG(duration):** Calculates the average contact duration for each group.
- **GROUP BY y:** Groups the data by the subscription status (y), which represents whether the client subscribed ('yes') or did not subscribe ('no').

```
hive> SELECT
    >     y AS subscription_status,
    >     AVG(duration) AS avg_contact_duration
    > FROM client_info
    > GROUP BY y;
```

Output:

```
OK
no      226.3475
y       NULL
yes     552.7428023032629
```

Summary of the Results:

- **subscription_status**: This indicates whether the client subscribed to the termdeposit (yesor no).
- **avg_contact_duration**: This is the average duration (in seconds) of contact withthe client, grouped by whether they subscribed or not.

The result provides insight into whether there is a difference in the average contact duration between clients who subscribed to the term deposit and those who did not. For instance, here higher average contact duration for the `yes` group suggests that longer interactions aremore effective in convincing clients to subscribe.

MADE BY: - ABHISEK YADAV