

# HADOOP

## 1. Data Ingestion:

- Create a directory in HDFS and transfer the banking dataset from the local system to the HDFS directory.

Solution:-

Step-1 Create a directory in HDFS

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

[Hadoop](#) [Overview](#) [Datanodes](#) [Datanode Volume Failures](#) [Snapshot](#) [Startup Progress](#) [Utilities](#)

### Browse Directory

Show  entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	<a href="#">drwxr-xr-x</a>	<a href="#">titov</a>	<a href="#">supergroup</a>	0 B	Apr 02 00:47	<a href="#">0</a>	0 B	<a href="#">user</a>	<input type="button" value="🗑"/>

Showing 1 to 1 of 1 entries

Hadoop, 2023.

Step-2 Transfer the banking dataset from the local system to the HDFS directory.

```
C:\hadoop\sbin>hadoop fs -copyFromLocal "C:\Users\nilesh\Downloads\bank.csv" /user/banking_dataset
```

[Hadoop](#) [Overview](#) [Datanodes](#) [Datanode Volume Failures](#) [Snapshot](#) [Startup Progress](#) [Utilities](#)

### Browse Directory

Show  entries

Search:

<input type="checkbox"/>	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name	
<input type="checkbox"/>	<a href="#">-rw-r--r--</a>	<a href="#">titov</a>	<a href="#">supergroup</a>	366.74 KB	Apr 02 00:53	<a href="#">3</a>	128 MB	<a href="#">bank.csv</a>	<input type="button" value="🗑"/>

Showing 1 to 1 of 1 entries

Hadoop, 2023.

## 2. Data Transformation with MapReduce:

Write a MapReduce program in Python that calculates the average account balance for each job type.

```
C:\hadoop\sbin>hadoop fs -copyFromLocal "C:\Users\nilesh\Downloads\Hadoop\mapper.py" /user/nilesh/input/
```

```
C:\hadoop\sbin>hadoop fs -copyFromLocal "C:\Users\nilesh\Downloads\Hadoop\reducer.py" /user/nilesh/input/
```

```
C:\hadoop\sbin>hadoop jar C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.2.4.jar -files "file:///C:/Users/nilesh/Downloads/Hadoop/mapper.py",file:///C:/Users/nilesh/Downloads/Hadoop/reducer.py" -mapper "python mapper.py" -reducer "python reducer.py" -input /user/nilesh/input/bank.csv -output /user/hadoop/output_new
```

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output_new/part-*
```

```
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
housemaid 1504.7105263157894
```

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

```
C:\hadoop\sbin>hadoop fs -copyFromLocal "C:\Users\nilesh\Downloads\Hadoop\New folder\mapper.py" /user/nilesh/input/
copyFromLocal: `/user/nilesh/input/mapper.py': File exists
```

```
C:\hadoop\sbin>hadoop fs -copyFromLocal "C:\Users\nilesh\Downloads\Hadoop\New folder\reducer.py" /user/nilesh/input/
copyFromLocal: `/user/nilesh/input/reducer.py': File exists
```

```
at org.apache.hadoop.util.RunJar.main(RunJar.java:236)
C:\hadoop\sbin>hadoop jar C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.2.4.jar -files "file:///C:/Users/nilesh/Downloads/Hadoop/New folder/mapper.py",file:///C:/Users/nilesh/Downloads/Hadoop/New folder/reducer.py" -mapper "python mapper.py" -reducer "python reducer.py" -input /user/nilesh/input/bank.csv -output /user/hadoop/output_q2
```

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output_q2/part-*
```

```
primary 94 583
secondary 416 1889
tertiary 173 1176
unknown 7 179
```

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

```
C:\hadoop\sbin>hadoop jar C:\hadoop\share\hadoop\tools\lib\hadoop-streaming-3.2.4.jar -files "file:///C:/Users/titov/Downloads/Hadoop/problem3/mapper.py",file:///C:/Users/titov/Downloads/Hadoop/problem3/reducer.py" -mapper "python mapper.py" -reducer "python reducer.py" -input /user/tito/input/bank.csv -output /user/hadoop/output_q3
```

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output_q3/part-*
```

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

### 3. Data Analysis with MapReduce:

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

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output_q5/part-*
```

failure	254.38367346938776
other	273.83248730964465
success	338.6356589147287
unknown	262.1031039136302

Examine the relationship between the age of clients and their balance, and present findings in a summarised form.

```
C:\hadoop\sbin>hadoop fs -cat /user/hadoop/output_q6/part-*
19      393.5
20      661.33333333333334
21      1774.2857142857142
22      1455.3333333333333
23      2117.95
24      634.625
25      1240.0681818181818
26      788.5584415584416
27      851.7765957446809
28      1025.0970873786407
29      1261.8762886597938
30      1113.0333333333333
31      1288.4824120603016
32      1256.549107142857
33      1545.4139784946237
34      1111.5367965367966
35      1192.8277777777778
36      1226.8936170212767
37      1463.9192546583852
38      1718.993710691824
39      1104.8615384615384
40      1399.5070422535211
41      1505.7925925925927
42      1612.3617021276596
43      1807.8347826086956
44      1836.5523809523809
45      1187.3660714285713
46      998.7731092436975
47      1363.0462962962963
48      1462.359649122807
49      1591.107142857143
50      1645.0594059405942
51      1528.5714285714287
52      782.2906976744187
53      1588.3085106382978
54      1656.661971830986
55      1244.9444444444443
56      2120.135135135135
57      1665.6263736263736
58      1755.0823529411764
59      1582.4788732394366
60      2964.574468085106
61      2407.5
62      516.1428571428571
63      2286.375
64      1103.2857142857142
```

## HIVE

### 1. Data Ingestion and Table Creation:

- Create a Hive database named **banking\_data**.
- Define and create a Hive table **client\_info** with appropriate data types for the **bank.csv** dataset.

- Load the data from the **bank.csv** file into the **client\_info** table.

```
hive> SELECT * FROM client_info LIMIT 5;
```

	job	marital	education	default	NULL	housing	loan	contact	NULL	month	NULL	NULL	NULL	NULL	poutcome	y
30	unemployed	married	primary	no	1787	no	no	cellular		19	oct	79	1	-1	0	unknown no
33	services	married	secondary	no	4789	yes	yes	cellular		11	may	220	1	339	4	failure no
35	management	single	tertiary	no	1350	yes	no	cellular		16	apr	185	1	330	1	failure no
30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown no	

## 2. Basic Data Exploration:

- Write a HiveQL query to count the total number of clients in the dataset.

```
hive> select count(*) from client_info;
```

```
OK
2024-05-05 17:27:50,472 INFO ql.Driver: OK
2024-05-05 17:27:50,472 INFO ql.Driver: Concurrency mode is disabled, not creating a lock manager
2024-05-05 17:27:50,478 INFO mapred.FileInputFormat: Total input files to process : 1
2024-05-05 17:27:50,485 INFO exec.ListSinkOperator: RECORDS_OUT_OPERATOR_LIST_SINK_10:1, RECORDS_OUT_INTERMEDIATE:0, 4522
Time taken: 24.905 seconds, Fetched: 1 row(s)
```

- Display the first 10 rows of the dataset.

```
hive> select * from client_info limit 10;
```

2024-05-05 17:05:11.599 INFO mapred.FileInputFormat: Total input files to process : 1																
NULL	job	marital	education	default	NULL	housing	loan	contact	NULL	month	NULL	NULL	NULL	NULL	poutcome	y
30	unemployed	married	primary	no	1787	no	no	cellular		19	oct	79	1	-1	0	unknown no
33	services	married	secondary	no	4789	yes	yes	cellular		11	may	220	1	339	4	failure no
35	management	single	tertiary	no	1350	yes	no	cellular		16	apr	185	1	330	1	failure no
30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown no	
59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown no	
35	management	single	tertiary	no	747	no	no	cellular		23	feb	141	2	176	3	failure no
36	self-employed	married	tertiary	no	307	yes	no	cellular		14	may	341	1	330	2	other no
39	technician	married	secondary	no	147	yes	no	cellular		6	may	151	2	-1	0	unknown no
41	entrepreneur	married	tertiary	no	221	yes	no	unknown	14	may	57	2	-1	0	unknown no	

## 3. Data Filtering and Sorting:

- Retrieve all records of clients who are married and have a personal loan.

```
hive> select * from client_info where marital='married' AND loan='yes';
```

```

36      -1      0      unknown no      secondary no      -872      yes      yes      cellular      20      nov      153
      unemployed      1      183      1      failure no
38      services      6      -1      0      unknown no
      management      1      -1      0      tertiary no
48      management      2      -1      0      tertiary no
      management      1      -1      0      tertiary no
37      management      2      -1      0      tertiary no
      management      1      -1      0      tertiary no
34      management      1      -1      0      tertiary no
      technician      6      -1      0      tertiary no
31      technician      6      -1      0      tertiary no
      services      2      -1      0      tertiary no
32      services      2      -1      0      tertiary no
      retired married      309      2      failure no
58      retired married      309      2      failure no
      technician      2      253      1      failure no
50      technician      2      253      1      failure no
      admin. married      184      7      failure no
35      admin. married      184      7      failure no
      technician      2      -1      0      tertiary no
54      technician      2      -1      0      tertiary no
      housemaid      -1      0      tertiary no
31      housemaid      -1      0      tertiary no
      technician      14      -1      0      tertiary no
50      technician      14      -1      0      tertiary no
      services      -1      0      tertiary no
36      services      -1      0      tertiary no
      services      2      -1      0      tertiary no
37      services      2      -1      0      tertiary no
      blue-collar      13      -1      0      tertiary no
58      blue-collar      13      -1      0      tertiary no
      blue-collar      -1      0      tertiary no
41      blue-collar      -1      0      tertiary no
      management      1      -1      0      tertiary no
50      management      1      -1      0      tertiary no
      admin. married      195      2      failure no
32      admin. married      195      2      failure no
      blue-collar      0      unknown no
57      blue-collar      0      unknown no
      blue-collar      -1      0      tertiary no
28      blue-collar      -1      0      tertiary no
      blue-collar      1      168      4      failure no
31      blue-collar      1      168      4      failure no
      management      4      -1      0      tertiary no
34      management      4      -1      0      tertiary no
      technician      3      -1      0      tertiary no
30      technician      3      -1      0      tertiary no
      unemployed      1      -1      0      tertiary no
33      unemployed      1      -1      0      tertiary no
      management      -1      0      tertiary no
34      management      -1      0      tertiary no
      services      -1      0      tertiary no
35      services      -1      0      tertiary no
      blue-collar      0      unknown no
37      blue-collar      0      unknown no
      management      1      256      1      success yes
40      management      1      256      1      success yes
      self-employed      12      -1      0      tertiary no
55      self-employed      12      -1      0      tertiary no
      services      1      -1      0      tertiary no
42      services      1      -1      0      tertiary no
      self-employed      -1      0      tertiary no
60      self-employed      -1      0      tertiary no
      admin. married      unknown no
42      admin. married      unknown no
      self-employed      -1      0      tertiary no
57      self-employed      -1      0      tertiary no
Time taken: 0.219 seconds, Fetched: 453 row(s)
2024-05-05 17:32:25,436 INFO CliDriver: Time taken: 0.219 seconds, Fetched: 453 row(s)
2024-05-05 17:32:25,436 INFO conf.HiveConf: Using the default value passed in for log id: 14dc1a57-a54e-4b11-98ad-aac325bcacbc
2024-05-05 17:32:25,436 INFO session.SessionState: Resetting thread name to main
hive>

```

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

```
hive> select job,marital,balance from client info order by balance desc limit 10;
```

```

2024-05-05 17:50:42,035 INFO ql.Driver: OK
2024-05-05 17:50:42,036 INFO ql.Driver: Concurrency mode is disabled, not creating a lock manager
2024-05-05 17:50:42,046 INFO mapred.FileInputFormat: Total input files to process : 1
retired married 71188
entrepreneur married 42045
technician single 27733
management married 27359
technician married 27069
housemaid single 26965
retired married 26452
services married 26394
management divorced 26306
retired single 25824
2024-05-05 17:50:42,066 INFO exec.ListSinkOperator: RECORDS_OUT_INTERMEDIATE:0, RECORDS_OUT_OPERATOR_LIST_SINK_8:10,
Time taken: 40.449 seconds, Fetched: 10 row(s)
2024-05-05 17:50:42,077 INFO CliDriver: Time taken: 40.449 seconds, Fetched: 10 row(s)
2024-05-05 17:50:42,078 INFO conf.HiveConf: Using the default value passed in for log id: 14dc1a57-a54e-4b11-98ad-aac325bcacbc
2024-05-05 17:50:42,079 INFO session.SessionState: Resetting thread name to main

```

#### 4. Data Aggregation and Grouping:

- Calculate the average age of clients for each job category.

```

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

```

```

2024-05-05 19:55:16,731 INFO mapred.FileInputFormat: Total input files to process : 1
2024-05-05 19:55:16,743 INFO exec.ListSinkOperator: RECORDS_OUT_OPERATOR_LIST_SINK_10:13, RECORDS_OUT_INTERMEDIATE:0,
admin. 39.68200836820084
blue-collar 40.15644820295983
entrepreneur 42.01190476190476
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
Time taken: 51.054 seconds, Fetched: 13 row(s)
2024-05-05 19:55:16,765 INFO CliDriver: Time taken: 51.054 seconds, Fetched: 13 row(s)
2024-05-05 19:55:16,765 INFO conf.HiveConf: Using the default value passed in for log id: 14dc1a57-a54e-4b11-98ad-aac325bcacbc
2024-05-05 19:55:16,766 INFO session.SessionState: Resetting thread name to main

```

- Find the total number of clients for each education level who have defaulted on credit.

```

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

```

```

OK
2024-05-05 20:00:06,421 INFO ql.Driver: OK
2024-05-05 20:00:06,421 INFO ql.Driver: Concurrency mode is disabled, not creating a lock manager
2024-05-05 20:00:06,424 INFO mapred.FileInputFormat: Total input files to process : 1
2024-05-05 20:00:06,432 INFO exec.ListSinkOperator: RECORDS_OUT_INTERMEDIATE:0, RECORDS_OUT_OPERATOR_LIST_SINK_13:4,
primary yes 10
secondary yes 46
tertiary yes 17
unknown yes 3
Time taken: 24.233 seconds, Fetched: 4 row(s)
2024-05-05 20:00:06,441 INFO CliDriver: Time taken: 24.233 seconds, Fetched: 4 row(s)
2024-05-05 20:00:06,441 INFO conf.HiveConf: Using the default value passed in for log id: 14dc1a57-a54e-4b11-98ad-aac325bcacbc
2024-05-05 20:00:06,442 INFO session.SessionState: Resetting thread name to main

```

## 5. Complex Queries for Insights:

- 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.

We directly calculate the average balance (`AVG(ci.balance)`) for each job category.

We also calculate the total number of clients (`COUNT(*)`) and the number of clients who have subscribed to a term deposit (`SUM(CASE WHEN ci.y = 'yes' THEN 1 ELSE 0 END)`).

We calculate the subscription rate as the percentage of subscribed clients out of the total number of clients.

The results are grouped by job category and sorted in descending order of average balance.

We limit the output to the top 5 job categories with the highest average balance using the `LIMIT` clause.

This query will give you 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.

Identify the top 5 job categories with the highest average balance:

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

```
2024-05-05 22:37:47,922 INFO exec.ListSinkOperator: RECORDS_OUT_INTERMEDIATE:0, RECORDS_OUT_OPERATOR_LIST_SINK_18:5,
retired 2319.191304347826 230 54 23.47826086956522
housemaid 2083.8035714285716 112 14 12.5
management 1766.9287925696594 969 131 13.519091847265221
entrepreneur 1645.125 168 15 8.928571428571429
student 1543.8214285714287 84 19 22.61904761904762
Time taken: 50.115 seconds, Fetched: 5 row(s)
2024-05-05 22:37:47,929 INFO CliDriver: Time taken: 50.115 seconds, Fetched: 5 row(s)
```

- 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).



We directly calculate the success rate for each month by counting the total number of contacts and the number of clients who subscribed to a term deposit within each month.

We use the `GROUP BY` clause to group the data by month.

The results are sorted in descending order of the number of contacts, and we limit the output to only the first row, which represents the month with the highest number of contacts.

This query will give you the month with the highest number of contacts and the success rate of the campaign in that month.

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

```
2024-05-05 22:29:00,692 INFO ql.Driver: OK
2024-05-05 22:29:00,692 INFO ql.Driver: Concurrency mode is disabled, not creating a lock manager
2024-05-05 22:29:00,692 INFO mapred.FileInputFormat: Total input files to process : 1
2024-05-05 22:29:00,708 INFO exec.ListSinkOperator: RECORDS_OUT_INTERMEDIATE:0, RECORDS_OUT_OPERATOR_LIST_SINK_18:1,
may 1398 93 6.652360515021459
Time taken: 46.87 seconds, Fetched: 1 row(s)
2024-05-05 22:29:00,708 INFO CliDriver: Time taken: 46.87 seconds, Fetched: 1 row(s)
```

## 6. Correlation Analysis:

- Calculate the correlation between age and balance for the clients.

In this simplified query:

1. We calculate the numerator of the correlation formula: the sum of the products of age and balance minus the count of records times the average of age times the average of balance.
2. We calculate the denominator of the correlation formula: the square root of the difference between the sum of the squares of age and the count times the square of the average of age, multiplied by the square root of the difference between the sum of the squares of balance and the count times the square of the average of balance.
3. We divide the numerator by the denominator to get the correlation coefficient.

This query should provide you with the correlation coefficient between age and balance for the clients in your dataset.

```
hive> SELECT (
  > SUM(age*balance)-COUNT(*)*AVG(age)*AVG(balance)
  > )/(
  > Sqrt(SUM(POW(age,2))-COUNT(*)*POW(AVG(age),2))*
  > Sqrt(SUM(POW(balance,2))-COUNT(*)*POW(AVG(balance),2))
  > )AS correlation_age_balance
  > FROM client_info;
2024-05-05 21:03:10,887 INFO conf.HiveConf: Using the default value passed in for log id: 2fe3dd08-4d00-4eec-a267-7f5cbbf8629c
2024-05-05 21:03:10,887 INFO session.SessionState: Updating thread name to 2fe3dd08-4d00-4eec-a267-7f5cbbf8629c main
2024-05-05 21:03:10,889 INFO ql.Driver: Compiling command(queryId=titov_20240505210310_5d5c2c37-75d7-4d3b-a6bf-f37b1c95a93c): SELECT (
SUM(age*balance)-COUNT(*)*AVG(age)*AVG(balance)
)/(
Sqrt(SUM(POW(age,2))-COUNT(*)*POW(AVG(age),2))*
Sqrt(SUM(POW(balance,2))-COUNT(*)*POW(AVG(balance),2))
)AS correlation_age_balance
FROM client_info
2024-05-05 21:03:56,534 INFO ql.Driver: OK
2024-05-05 21:03:56,537 INFO ql.Driver: Concurrency mode is disabled, not creating a lock manager
2024-05-05 21:03:56,550 INFO mapred.FileInputFormat: Total input files to process : 1
2024-05-05 21:03:56,637 INFO exec.ListSinkOperator: RECORDS_OUT_INTERMEDIATE:0, RECORDS_OUT_OPERATOR_LIST_SINK_11:1,
0.0835552815119586
Time taken: 45.649 seconds, Fetched: 1 row(s)
2024-05-05 21:03:56,733 INFO CliDriver: Time taken: 45.649 seconds, Fetched: 1 row(s)
```

## 7. Trend Analysis:(TO BE DONE)

- Analyze the year-over-year trend in the number of clients contacted.

In this query:

We use the `SUBSTRING` function to extract the year from the `month` column. Assuming the `month` column is in the format "YYYY-MM", we extract the first four characters to get the year.

We count the number of clients contacted (`COUNT(*)`) for each year.

We group the results by the extracted year.

Finally, we order the results by year to see the trend over time.

This query will give you the year-over-year trend in the number of clients contacted based on the data in your `client_info` table.

```
hive> SELECT SUBSTRING(month,1,4) AS year,
  > COUNT(*) AS num_clients_contacted
  > FROM
  > client_info
  > GROUP BY SUBSTRING(month,1,4)
  > ORDER BY year;
```

```

2024-05-05 21:12:45,360 INFO mapred.FileInputFormat: Total input files to process : 1
2024-05-05 21:12:45,377 INFO exec.ListSinkOperator: RECORDS_OUT_INTERMEDIATE:0, RECORDS_OUT_OPERATOR_LIST_SINK_16:13,
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
Time taken: 100.264 seconds, Fetched: 13 row(s)

```

## 8. Anomaly Detection:

- Identify any unusual patterns in the average yearly balance across different education levels.

We calculate the average yearly balance for each education level.

We also calculate the overall average yearly balance for each year using the window function `AVG(AVG(balance)) OVER (PARTITION BY SUBSTRING(month, 1, 4))`.

We calculate the z-score for each average yearly balance within each education level by subtracting the overall average yearly balance and dividing by the standard deviation, both calculated over the partition of years.

By examining the z-scores, you can identify any unusual patterns in the average yearly balance across different education levels. A z-score significantly higher or lower than zero indicates that the average yearly balance for a particular education level is unusually high or low compared to the overall average yearly balance.

```

hive> SELECT
> SUBSTRING(month,1,4) AS year,
> education,
> AVG(balance) AS avg_yearly_balance,
> AVG(AVG(balance)) OVER (PARTITION BY SUBSTRING(month,1,4)) AS overall_avg_balance,
> (AVG(balance)-AVG(AVG(balance)) OVER (PARTITION BY SUBSTRING(month,1,4)))/STDDEV(balance) OVER(PARTITION BY SUBSTRING(month,1,4)) AS z_score
> FROM client_info
> GROUP BY SUBSTRING(month,1,4),education;

```

## 9. Advanced Analysis:

- 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.

In this query:

1. We group the data by the poutcome column to analyze the impact of previous campaign outcomes.
2. We count the total number of clients (`total_clients`) and the number of clients who subscribed to term deposits (`subscribed_clients`) for each poutcome category.
3. We calculate the subscription rate (`subscription_rate`) as the percentage of clients who subscribed to term deposits out of the total number of clients for each poutcome category.

This query will provide you with the subscription rate for each poutcome category, allowing you to analyze the impact of previous campaign outcomes on the current campaign's success.

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

```
2024-05-05 21:20:32,943 INFO ql.Driver: OK
2024-05-05 21:20:32,943 INFO ql.Driver: Concurrency mode is disabled, not creating a lock manager
2024-05-05 21:20:32,948 INFO mapred.FileInputFormat: Total input files to process : 1
2024-05-05 21:20:32,956 INFO exec.ListSinkOperator: RECORDS_OUT_INTERMEDIATE:0, RECORDS_OUT_OPERATOR_LIST_SINK_11:5,
failure 490      63      12.857142857142856
other 197      38      19.289340101522843
poutcome 1      0      0.0
success 129     83      64.34108527131784
unknown 3705    337     9.095816464237517
Time taken: 24.245 seconds, Fetched: 5 row(s)
2024-05-05 21:20:32,967 INFO CliDriver: Time taken: 24.245 seconds, Fetched: 5 row(s)
```

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

In this query:

1. We group the data by the subscription status (`y` column), which indicates whether the client subscribed to a term deposit.
2. We calculate the average contact duration (`avg_contact_duration`) for each group separately.

This query will provide you with the average contact duration for clients who subscribed and who did not subscribe to a term deposit, allowing you to compare the contact durations between the two groups.

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

```
2024-05-05 21:26:25,506 INFO mapred.FileInputFormat: Total input files to process : 1  
2024-05-05 21:26:25,511 INFO exec.ListSinkOperator: RECORDS_OUT_INTERMEDIATE:0, RECORDS_OUT_OPERATOR_LIST_SINK_16:3,  
yes 552.7428023032629  
no 226.3475  
y NULL  
Time taken: 54.453 seconds, Fetched: 3 row(s)  
2024-05-05 21:26:25,517 INFO CliDriver: Time taken: 54.453 seconds, Fetched: 3 row(s)  
2024-05-05 21:26:25,517 INFO conf.HiveConf: Using the default value passed in for log id: 2fe3dd08-4d00-4eec-a267-7f5cbbf8629c  
2024-05-05 21:26:25,518 INFO session.SessionState: Resetting thread name to main
```

## Submission Guidelines:

- Make a copy of this doc file.
- Perform the analysis in your local system using Hadoop and Hive and provide screenshots of both the **code** and the **output** under each question.
- Upload the doc file with other files and submit it in the submission dashboard.