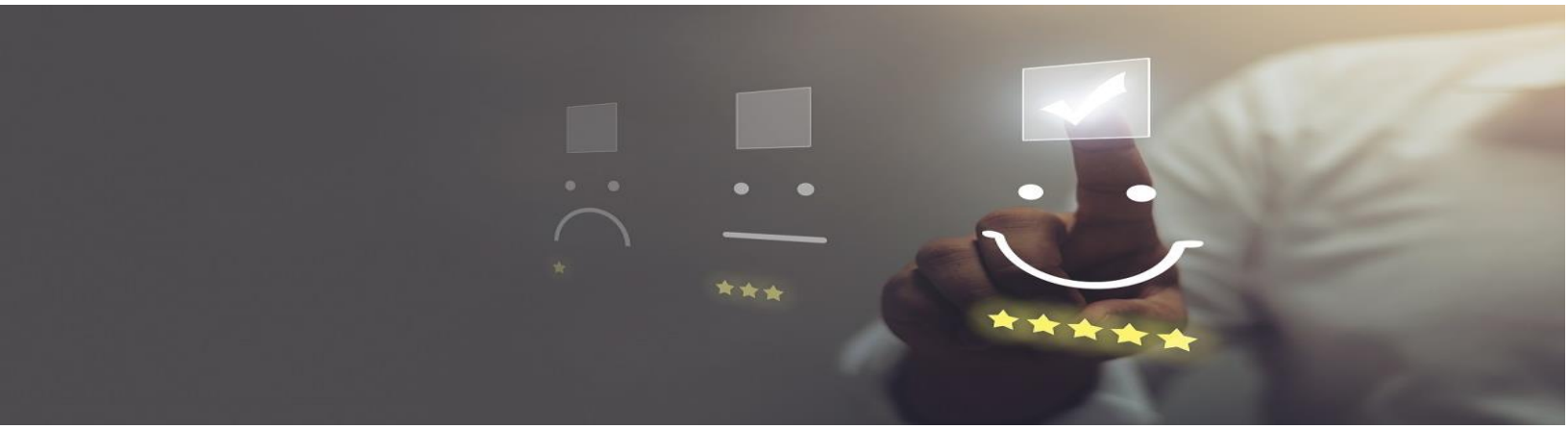
Customer Churn Prediction

Mini Project



NOVEMBER 21, 2021

DATA SCIENCE

Big Data Analytics - 18CSC403

Group-9

Name:	RollNo:
Gajapathy Selvaraj	**CB.SC.I5DAS18011**
Kamal G Anand	**CB.SC.I5DAS18015**
Karthik Rayan V	**CB.SC.I5DAS18017**
Lakshman P	**CB.SC.I5DAS18019**

Dataset Information :

Name - Bank Customer Churn data (Churn_Modelling.csv)
Source - Kaggle
Link - https://www.kaggle.com/adammaus/predicting-churn-for-bank-customers

About

Customer Churn is when customers leave a service in a given period of time, what is bad for business. This work has the objective to analyze the different aspects of the dataset so as to predict which customers will leave the service and the dataset used is the Banking Customer Churn, hosted at Kaggle.

- Each row represents a customer, each column contains customer's attributes described on the column Metadata.
- Customers who left within the last month – the column is called Exited
- Details that each customer has signed up for – Balance, Number of products, has credit card or not
- Customer account information – Tenure, Is an active member or not
- Demographic info about customers – Gender, Age range, Estimated salary, Location

- Understanding the trends and patterns of data.
  - Understanding the basic nature of customers is important in any business and understanding the loyal customers helps us to improve the business as well
  - So, these queries are done in order to sustain the bank, increase the business of the bank and widen the application and capabilities of the bank to a large audience

MapReduce

1. MapReduce program(written in java) to find the distinct tenure with the no of people in each group.

```
hduser@lman-VirtualBox:~$ nano DistinctValuesExample1.java
hduser@lman-VirtualBox:~$ export HADOOP_CLASSPATH=$JAVA_HOME/lib/tools.jar
hduser@lman-VirtualBox:~$ hadoop com.sun.tools.javac.Main /home/hduser/DistinctValuesExample1.java
Note: /home/hduser/DistinctValuesExample1.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
hduser@lman-VirtualBox:~$ rm -r ll.jar
hduser@lman-VirtualBox:~$ jar cf ll.jar DistinctValues*.class
hduser@lman-VirtualBox:~$ hadoop jar /home/hduser/ll.jar DistinctValuesExample1 /user/hduser/dataset/Churn_modelling.csv /user/hduser/distinct_output2
21/11/15 10:51:51 INFO mapred.LocalJobRunner: reduce task executor complete.
21/11/15 10:51:52 INFO mapreduce.Job:  map 100% reduce 100%
21/11/15 10:51:53 INFO mapreduce.Job: Job job_local595784836_0001 completed successfully
21/11/15 10:51:53 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=87428
                FILE: Number of bytes written=690703
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1369716
                HDFS: Number of bytes written=30
                HDFS: Number of read operations=13
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=4
        Map-Reduce Framework
                Map input records=10001
                Map output records=10001
                Map output bytes=20497
                Map output materialized bytes=40505
                Input split bytes=127
                Combine input records=0
                Combine output records=0
                Reduce input groups=12
                Reduce shuffle bytes=40505
                Reduce input records=10001
                Reduce output records=12
                Spilled Records=20002
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=104
                Total committed heap usage (bytes)=299245568
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                Bytes Written=30
hduser@lman-VirtualBox:~$ hdfs dfs -cat /user/hduser/distinct_output2/part*
21/11/15 10:52:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
0
1
10
2
3
4
5
6
7
8
9
Tenure
```

```
hduser@lman-VirtualBox:~$ hdfs dfs -cat /user/hduser/Driver1_output/part*
21/11/07 21:09:23 WARN util.NativeCodeLoader: Unable to load native-hadoop library for
orm... using builtin-java classes where applicable
Surname NULL
Azubuike        350
Campbell        350
Onyekachi       350
Lin     350
Maslow  350
Chou    351
Aikenhead       358
Panicucci       359
Thomas  363
Ozoemena        365
hduser@lman-VirtualBox:~$
```

PIG AND HIVE:

2. Using pig, selecting the active customers having bank balance greater than 1 lakh.

```
grunt> DATA = load '/user/hduser/dataset/Churn_Modelling.csv' USING PigStorage(',') as (RowNumber :int,CustomerId:long,Surname:chararr
e:int,Balance:float,NumOfProducts:int,HasCrCard:int,IsActiveMember:int,EstimatedSalary:float,Exited:int);
2021-11-07 14:45:34,194 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use
2021-11-07 14:45:34,195 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs
grunt> a= filter DATA by (EstimatedSalary > 100000.00) AND (IsActiveMember == 1);
2021-11-07 14:45:58,621 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 1 time(s).
grunt> b= foreach a generate Surname,CustomerId;
2021-11-07 14:46:33,754 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 1 time(s).
grunt> c= limit d 10;
```

```
(McDonald,15577657)
(Yen,15625047)
(Hargrave,15634602)
(Hill,15647311)
(Odinakachukwu,15706552)
(McWilliams,15728693)
(Clements,15732963)
(Young,15736816)
(Maclean,15738191)
(Watson,15788448)
```

3. Creating a view to find the important customers on the basis of number of products and bank balance ,segregating each one of them as “Important Client ” and “Not Important Client” ,which in turn help in the bank on how to deal with each one of them .

```
hive> create view imp_customers as select surname,balance,case exited when 1 then 'Imortant clie
nt' else 'Not Imp' end from churn where numofproducts = 1 and balance >100000;
OK
Time taken: 0.977 seconds
```

```
hive> select * from imp_customers limit 5;
OK
Mitchell        125510.82       Not Imp
H?      134603.88       Not Imp
Romeo   132602.88       Imortant client
Young   136815.64       Not Imp
McWilliams      141349.44       Not Imp
Time taken: 0.508 seconds, Fetched: 5 row(s)
```

4. Creating a view of the people having rich social dilemma in these bank according to different geography ,so that one can easily concentrate on the which region to concentrate more on to increase the business

```
hive> create view geography_rich as select geography,sum(Isactivemember),count(*) from churn group by geography;
OK
Time taken: 0.853 seconds
hive> select * from geography_rich;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different
Query ID = hduser_20211027214110_915b7535-455e-421a-aa1e-5027895b77f2
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2021-10-27 21:41:12,939 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1214870157_0006
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 8234680 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
France  2591    5014
Geography       NULL    1
Germany 1248    2509
Spain   1312    2477
Time taken: 2.226 seconds, Fetched: 4 row(s)
```

5. From the query before to check on the active members of each geography, specifying a particular geography and getting the important clients surname ,balance and number of products on the basis of credit score, balance not null and number of products.

```
hive> select surname,balance,numofproducts from churn where creditscore >700 and numofproducts >
2 and balance !=0.0 and geography='France';
OK
Walker  100749.5        3
Colman  115414.19       3
Fedorov 176845.4        3
Chinedum        88308.87        3
Galkin  127785.17       4
Aksenov 95813.76        3
Osonduagwuike   118022.06       3
Price   123681.32       3
Lin     128289.7        3
Lavrentyev      117280.23       3
Voronoff        139914.6        4
Bolton  79475.3 3
Oluchukwu       141252.19       4
Yen     128548.49       4
Banks   132084.66       3
Greece  136492.92       3
Osborne 108309.0        4
Craig   46388.16        3
Time taken: 0.434 seconds, Fetched: 18 row(s)
```

6. Selecting the proportion of gender(male, female) having tenure more than 5 years ,in order to understand the proportion of each and every member in the bank and how many years they are invested in the bank .

```
hive> SELECT gender,count(*) FROM CHURN where tenure >5 group by gender;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20211027142543_41827f78-7abe-48e6-844c-93c7b883d93e
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2021-10-27 14:25:44,874 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local302352858_0007
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 12335636 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Female  2022
Male    2472
Time taken: 1.776 seconds, Fetched: 2 row(s)
```

7. Joining two view inorder to find the top 10 class of customers and why they left

```
hive> create view j1 as select surname,creditscore from churn where isactivemember ==1 order by creditscore desc limit 25;
OK
Time taken: 0.441 seconds
hive> create view j2 as select surname,numofproducts,exited from churn;
OK
Time taken: 0.407 seconds
```

```
hive> select distinct(c.surname),c.creditscore,d.numofproducts from j1  c  join j2 d on (c.surname = d.surname) where d.exited ==0;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution eng
Query ID = hduser_20211115191547_761b5931-c50c-465d-8055-61aace8f2086
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2021-11-15 19:15:49,991 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local623919519_0009
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/apache-hive-2.3.7-bin/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinc
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
```

```
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 8
Stage-Stage-3:  HDFS Read: 8
Total MapReduce CPU Time Spe
OK
Amadi   850     1
Amadi   850     2
Armstrong       850     1
Benjamin        850     2
Browne  850     1
Browne  850     2
Cameron 850     1
Cameron 850     2
Ch'eng  850     1
Ch'eng  850     2
Clogstoun       850     1
Davidson        850     1
Davidson        850     2
De Salis        850     2
Degtyarev       850     1
Degtyarev       850     2
Ellis   850     1
Ellis   850     2
Hansen  850     1
Hansen  850     2
Hudson  850     2
Hughes  850     1
Hughes  850     2
Jones   850     1
Jones   850     2
Li Fonti        850     1
Li Fonti        850     2
Major   850     2
Maslova 850     1
Maslova 850     2
McKay   850     1
McKay   850     2
Nkemjika        850     1
Nkemjika        850     2
She     850     1
She     850     2
Theus   850     1
Ts'ui   850     1
Ts'ui   850     2
Ts'ui   850     3
Vorobyova       850     1
Vorobyova       850     2
Time taken: 15.949 seconds,
```

8.

Encording the important people(active member and credit score) names in the bank(printf,reverse,space,ascii,soundex).

```
hive> create view crypto as select surname,printf("------>"),space(5),printf("encoreded as : %d",ascii(reverse(surname))),printf("Or"),soundex(surnam
OK
Time taken: 0.331 seconds
hive> select * from crypto limit 10;
OK
Hargrave        ------>         encoreded as : 101      Or      H626
Hill    ------>         encoreded as : 108      Or      H400
Mitchell        ------>         encoreded as : 108      Or      M324
Bartlett        ------>         encoreded as : 116      Or      B634
He      ------>         encoreded as : 101      Or      H000
H?      ------>         encoreded as : 63       Or      H000
Scott   ------>         encoreded as : 116      Or      S300
Goforth ------>         encoreded as : 104      Or      G163
Henderson       ------>         encoreded as : 110      Or      H536
Hao     ------>         encoreded as : 111      Or      H000
Time taken: 0.351 seconds, Fetched: 10 row(s)
hive>
```

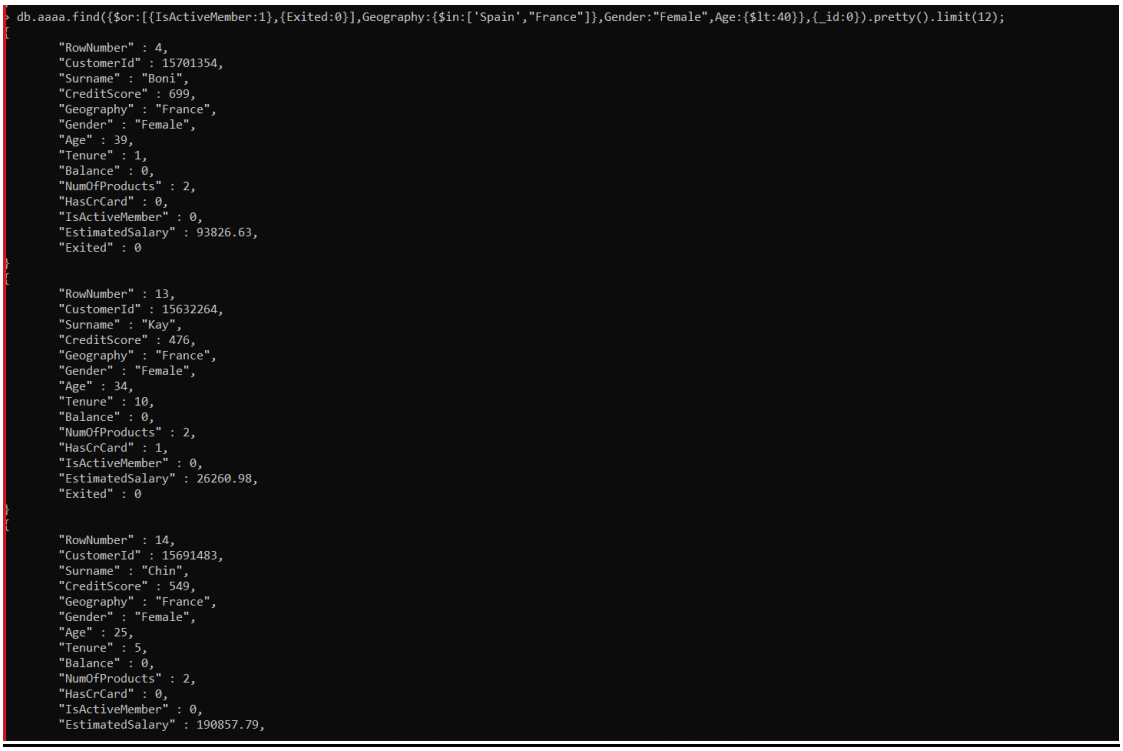
CB.SC.I5DAS18019 Lakshman P

MONGO

1.

Finding the loyal customers who are important for the bank to prevail

db.aaaa.find({$or:[{IsActiveMember:1},{Exited:0}],Geography:{$in:['Spain',"France"]},Gender:"Female",Age:{$lt:40}},{_id:0}).pretty().limit(12);



```
> db.aaaa.find({CreditScore: {$gte: 640,$lt: 660},Age: {$gte: 35,$lt: 40},Exited: 1,HasCrCard: 1,IsActiveMember: 1,NumOfProducts: 3}).count()
2
> db.aaaa.find({CreditScore: {$gte: 640,$lt: 660},Age: {$gte: 35,$lt: 40},Exited: 1,HasCrCard: 1,IsActiveMember: 1,NumOfProducts: 3}).pretty()
{
        "_id" : ObjectId("61a3a72450c988518099e873"),
        "RowNumber" : 6525,
        "CustomerId" : 15743293,
        "Surname" : "Waters",
        "CreditScore" : 651,
        "Geography" : "Germany",
        "Gender" : "Female",
        "Age" : 35,
        "Tenure" : 1,
        "Balance" : 163700.78,
        "NumOfProducts" : 3,
        "HasCrCard" : 1,
        "IsActiveMember" : 1,
        "EstimatedSalary" : 29583.48,
        "Exited" : 1
}
{
        "_id" : ObjectId("61a3a72550c988518099ef93"),
        "RowNumber" : 8349,
        "CustomerId" : 15796230,
        "Surname" : "Morley",
        "CreditScore" : 642,
        "Geography" : "Germany",
        "Gender" : "Female",
        "Age" : 36,
        "Tenure" : 2,
        "Balance" : 124495.98,
        "NumOfProducts" : 3,
        "HasCrCard" : 1,
        "IsActiveMember" : 1,
        "EstimatedSalary" : 57904.22,
        "Exited" : 1
}
>
```

2.

Making a Contingency table in mongo in order to understand about the credit card and Is an active member

db.aaaa.aggregate([{$group:{_id:{CreditCard:"$HasCrCard",Member:"$IsActive Member"},count:{$sum:1}}},{$sort:{"_id.Member":1}}]);

```
> db.aaaa.aggregate([{$group:{_id:{CreditCard:"$HasCrCard",Member:"$IsActiveMember"},count:{$sum:1}}},{$sort:{"_id.Member":1}}]);
{ "_id" : { "CreditCard" : 0, "Member" : 0 }, "count" : 1401 }
{ "_id" : { "CreditCard" : 1, "Member" : 0 }, "count" : 3448 }
{ "_id" : { "CreditCard" : 1, "Member" : 1 }, "count" : 3607 }
{ "_id" : { "CreditCard" : 0, "Member" : 1 }, "count" : 1544 }
>
```

3.

Finding male citizens of spain whos age is greater than 40 and salary greater tha 5L can enable the growthof business and feature extraction on active.

db.aaaa.aggregate([{$match:{$or:[{"EstimatedSalary":{$gt:500000}},{$and:[{"Gender":"Male"}]}]}},{$match:{$or:[{$and:[{"Geography" : "Spain"},{"Age":{"$lt":40}}]}]}},{$project:{rateMultiply:{$multiply:["$CreditScore","$Exited"]}, Tenure:1, Balance:1,Surname:1,_id:0}}])

```
> db.aaaa.aggregate([{$match:{$or:[{"EstimatedSalary":{$gt:500000}},{$and:[{"Gender":"Male"}]}]}},{$match:{$or:[{$and:[{"Geography" : "Spain"},{"Age":{"$lt" : 40}}]}]}},{$project:{rateMultiply:{$multiply:["$Cred
itScore","$Exited"]}, Tenure:1, Balance:1,Surname:1,_id:0}}])
{ "Surname" : "Andrews", "Tenure" : 3, "Balance" : 0, "rateMultiply" : 0 }
{ "Surname" : "Watson", "Tenure" : 3, "Balance" : 145260.23, "rateMultiply" : 0 }
{ "Surname" : "Lorenzo", "Tenure" : 7, "Balance" : 76548.6, "rateMultiply" : 0 }
{ "Surname" : "Jeffrey", "Tenure" : 1, "Balance" : 56084.69, "rateMultiply" : 0 }
{ "Surname" : "Bushell", "Tenure" : 5, "Balance" : 77253.22, "rateMultiply" : 0 }
{ "Surname" : "Gant", "Tenure" : 3, "Balance" : 121681.82, "rateMultiply" : 750 }
{ "Surname" : "Fiorentini", "Tenure" : 10, "Balance" : 176273.95, "rateMultiply" : 0 }
{ "Surname" : "Allard", "Tenure" : 8, "Balance" : 0, "rateMultiply" : 0 }
{ "Surname" : "Bradley", "Tenure" : 7, "Balance" : 0, "rateMultiply" : 0 }
{ "Surname" : "Walkom", "Tenure" : 5, "Balance" : 150092.8, "rateMultiply" : 0 }
{ "Surname" : "Chiemela", "Tenure" : 9, "Balance" : 106190.55, "rateMultiply" : 0 }
{ "Surname" : "Kennedy", "Tenure" : 6, "Balance" : 120193.42, "rateMultiply" : 0 }
{ "Surname" : "Taverner", "Tenure" : 0, "Balance" : 133702.89, "rateMultiply" : 683 }
{ "Surname" : "Marshall", "Tenure" : 5, "Balance" : 0, "rateMultiply" : 0 }
{ "Surname" : "Crawford", "Tenure" : 9, "Balance" : 61710.44, "rateMultiply" : 0 }
{ "Surname" : "Chiemezie", "Tenure" : 2, "Balance" : 141040.01, "rateMultiply" : 0 }
{ "Surname" : "Ginikanwa", "Tenure" : 3, "Balance" : 176666.62, "rateMultiply" : 0 }
{ "Surname" : "Yang", "Tenure" : 7, "Balance" : 0, "rateMultiply" : 0 }
{ "Surname" : "Grant", "Tenure" : 4, "Balance" : 0, "rateMultiply" : 0 }
{ "Surname" : "Madison", "Tenure" : 7, "Balance" : 0, "rateMultiply" : 0 }
Type "it" for more
```

4.

Feature extraction and Generation the estimating the how a person should save each month and the average amount the person has on the basis of the bank and projecting the surname ,credit score and the features extracted.

db.aaaa.aggregate([{$project:{Surname:1,_id:0,Sould_Invest_In_A_Month:{$divide:["$Balance",12]},Avg_Amount_ThePersonHas:{$avg:"$Balance"},CreditScore:1}}]).pretty()

```
> db.aaaa.aggregate([{$project:{Surname:1,_id:0,Sould_Invest_In_A_Month:{$divide:["$Balance",12]},Avg_Amount_ThePersonHas:{$avg:"$Balance"},CreditScore:1}}]).pretty()
{
        "Surname" : "Hargrave",
        "CreditScore" : 619,
        "Sould_Invest_In_A_Month" : 0,
        "Avg_Amount_ThePersonHas" : 0
}
{
        "Surname" : "Hill",
        "CreditScore" : 608,
        "Sould_Invest_In_A_Month" : 6983.988333333334,
        "Avg_Amount_ThePersonHas" : 83807.86
}
{
        "Surname" : "Onio",
        "CreditScore" : 502,
        "Sould_Invest_In_A_Month" : 13305.066666666666,
        "Avg_Amount_ThePersonHas" : 159660.8
}
{
        "Surname" : "Boni",
        "CreditScore" : 699,
        "Sould_Invest_In_A_Month" : 0,
        "Avg_Amount_ThePersonHas" : 0
}
{
        "Surname" : "Mitchell",
        "CreditScore" : 850,
        "Sould_Invest_In_A_Month" : 10459.235,
        "Avg_Amount_ThePersonHas" : 125510.82
}
{
        "Surname" : "Chu",
        "CreditScore" : 645,
        "Sould_Invest_In_A_Month" : 9479.648333333333,
        "Avg_Amount_ThePersonHas" : 113755.78
}
{
        "Surname" : "Bartlett",
        "CreditScore" : 822,
        "Sould_Invest_In_A_Month" : 0,
        "Avg_Amount_ThePersonHas" : 0
}
```

CB.SC.I5DAS18019 Lakshman P

5.

In our dataset the names of the customers is given as a last name ,It may irradicate the analysis in future So we have to generalise all the surname and its replicas .

Here we are finding the duplicate names along with their ids.

db.aaaa.aggregate([{$group: {_id: {Duplicated_Names: "$Surname"},uniqueIds: {$addToSet: "$_id"},count: {$sum: 1}}},{$match: { count: {"$gt": 1}}}]).pretty();

```
> db.aaaa.aggregate([{$group: {_id: {Duplicated_Names: "$Surname"},uniqueIds: {$addToSet: "$_id"},count: {$sum: 1}}},{$match: { count: {"$gt": 1}}}]).pretty();
{
        "_id" : {
                "Duplicated_Names" : "Begum"
        },
        "uniqueIds" : [
                ObjectId("61a3a72350c988518099da6b"),
                ObjectId("61a3a72350c988518099d13f")
        ],
        "count" : 2
}
{
        "_id" : {
                "Duplicated_Names" : "Chimaraoke"
        },
        "uniqueIds" : [
                ObjectId("61a3a72450c988518099ea5a"),
                ObjectId("61a3a72550c988518099efba"),
                ObjectId("61a3a72550c988518099f531"),
                ObjectId("61a3a72550c988518099f1de")
        ],
        "count" : 4
}
{
        "_id" : {
                "Duplicated_Names" : "Chinweuba"
        },
        "uniqueIds" : [
                ObjectId("61a3a72450c988518099de2d"),
                ObjectId("61a3a72350c988518099d77d"),
                ObjectId("61a3a72450c988518099edb5"),
                ObjectId("61a3a72450c988518099de2c")
        ],
        "count" : 4
}
{
        "_id" : {
                "Duplicated_Names" : "Hay"
        },
        "uniqueIds" : [
                ObjectId("61a3a72350c988518099d28e"),
                ObjectId("61a3a72450c988518099e015"),
                ObjectId("61a3a72350c988518099cfd3"),
                ObjectId("61a3a72350c988518099d75f"),
                ObjectId("61a3a72350c988518099d3d6"),
                ObjectId("61a3a72450c988518099ea73"),
                ObjectId("61a3a72450c988518099eb4f"),
                ObjectId("61a3a72350c988518099d733")
        ],
```

CB.SC.I5DAS18017 Karthik Rayan V

Mapreduce:

To find out the frequency of values in exited column.

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
public class Exited_count{
public static class FrequencyMapper extends Mapper<LongWritable, Text, Text, In$
  private final static IntWritable one = new IntWritable(1);
  @Override
  public void map(LongWritable offset, Text lineText, Context context)
        throws IOException, InterruptedException {
 String line = lineText.toString();
        String exited = line.split(",")[13];
        context.write(new Text(exited), one);
  }
}

public class FrequencyReducer extends Reducer<Text ,  IntWritable ,  Text ,  IntWritable > {
        @Override public void reduce( Text eventID,  Iterable<IntWritable> counts,  Context
context)
        throws IOException,  InterruptedException {
        int sum = 0;
        for ( IntWritable count : counts) {
        sum += count.get();
        }
        context.write(exited, new IntWritable(sum));
        }
}
public static void main(String[] args) throws Exception{
        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Exited_count");
        job.setJarByClass(Exited_count.class);
```

```
        job.setMapperClass(FrequencyMapper.class);
        job.setCombinerClass(FrequencyReducer.class);
        job.setReducerClass(FrequencyReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        System.exit(job.waitForCompletion(true) ? 0 : 1);
}}
```

```
hduser@krv_server:~$ nano Exited_count.java
hduser@krv_server:~$ hadoop com.sun.tools.javac.Main Exited_count.java
hduser@krv_server:~$ ls
 apache-hive-2.3.7-bin.tar.gz           Exited_count.java     metastore_db
 derby.log                              hadoop-2.7.3          metastore_db.tmp
 Exited_count.class                     hadoop-2.7.3.tar.gz   pig-0.16.0
'Exited_count$FrequencyMapper.class'    hive                  pig-0.16.0.tar.gz
'Exited_count$FrequencyReducer.class'   input                 pig_1632497403647.log
```

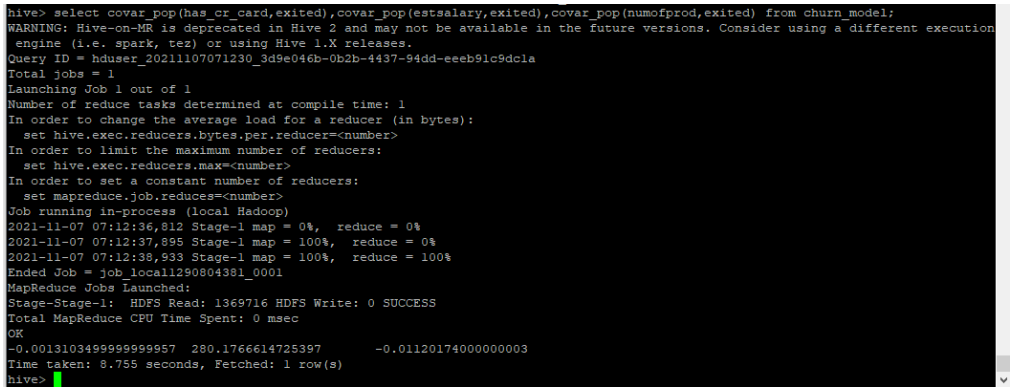
```
hduser@krv_server:~$ hadoop jar ec.jar Exited_count /user/inputs/Churn_Modelling.csv /user/output_dir
21/11/01 10:53:56 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin
-java classes where applicable
21/11/01 10:53:56 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.metrics.session-id
21/11/01 10:53:56 INFO jvm.JvmMetrics: Initializing JVM Metrics with processName=JobTracker, sessionId=
21/11/01 10:53:56 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the
 Tool interface and execute your application with ToolRunner to remedy this.
21/11/01 10:53:56 INFO input.FileInputFormat: Total input paths to process : 1
21/11/01 10:53:57 INFO mapreduce.JobSubmitter: number of splits:1
21/11/01 10:53:57 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1040222482_0001
```

```
    Map input records=10001
    Map output records=10001
    Map output bytes=60011
    Map output materialized bytes=35
    Input split bytes=119
    Combine input records=10001
    Combine output records=3
    Reduce input groups=3
    Reduce shuffle bytes=35
    Reduce input records=3
    Reduce output records=3
    Spilled Records=6
    Shuffled Maps =1
    Failed Shuffles=0
    Merged Map outputs=1
    GC time elapsed (ms)=5
    Total committed heap usage (bytes)=471859200
```

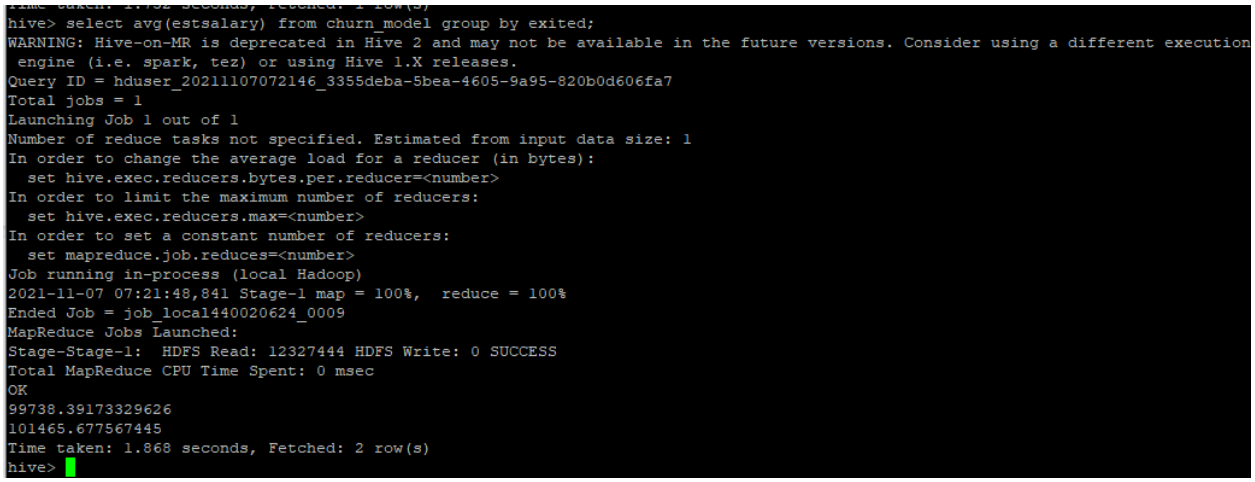
```
hduser@krv_server:~$ hdfs dfs -ls /user/output_dir
21/11/01 10:54:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin
-java classes where applicable
Found 2 items
-rw-r--r--   1 hduser supergroup          0 2021-11-01 10:53 /user/output_dir/_SUCCESS
-rw-r--r--   1 hduser supergroup         23 2021-11-01 10:53 /user/output_dir/part-r-00000
hduser@krv_server:~$ hdfs dfs -cat /user/output_dir/part-r-00000
21/11/01 10:54:33 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin
-java classes where applicable
0       7963
1       2037
```

Hive and Pig

1. Find out the covariance of creditcard,estimated salary and num of products with exited.



2. Since estimated salary showed higher positive covariance, find out the difference in average salary for people who left and didn't leave.



Select avg(estsalary) from churn_model group by exited;

We can see people who, exited have a higher average salary compared to people who are still in the service, even considering the fact that exited people are less in number compared to people who stay.

3. Create a contingency table for location,gender and exited and analyse the frequency

```
Total MapReduce CPU Time Spent: 0 msec
OK
0       France  Female  1801
1       France  Female  460
0       France  Male    2403
1       France  Male    350
0       Germany Female  745
1       Germany Female  448
0       Germany Male    950
1       Germany Male    366
0       Spain   Female  858
1       Spain   Female  231
0       Spain   Male    1206
1       Spain   Male    182
Time taken: 2.971 seconds, Fetched: 12 row(s)
```

Putting having exited =1 ;

```
OK
France  Female  1       460
France  Male    1       350
Germany Female  1       448
Germany Male    1       366
Spain   Female  1       231
Spain   Male    1       182
Time taken: 1.77 seconds, Fetched: 6 row(s)
hive>
```

4. select covar_pop(age,estsalary),covar_pop(age,exited) from churn_model;

```
OK
-4342.938266805138      1.2051293400000043
Time taken: 1.6 seconds, Fetched: 1 row(s)
hive> hduser@krv server:~$
```

Here there is a negative covariance showing that as age increases the salary of the people decreases and when age increases the chance of people exiting is also there , which proves our finding that estimated salary's weak influence.

5. Select geo,avg(estsalary) from churn_model group by geo;

```
France  99899.18080236696
Germany 101113.43509471782
Spain   99440.57221695951
Time taken: 8.311 seconds, Fetched: 3 row(s)
```

6. Top 5 people with high tenure but can see that they are still in service (exited = 0)

```
Details at logfile: /home/hduser/input/pig_1636293381624.log
grunt> ordered = order data by tenure DESC;
grunt> top5 = limit ordered 5;
grunt> dump top5;
```

```
 process : 1
(3187,15649668,Wilhelm,637,Germany,Female,36,10,145750.45,2,1,1,96660.76,0)
(2501,15713378,Brownless,711,France,Male,38,10,0.0,2,0,0,53311.78,0)
(7896,15660571,Halpern,668,Spain,Male,43,10,113034.31,1,1,1,100423.88,0)
(4097,15758775,Vasilyeva,820,Spain,Male,34,10,97208.46,1,1,1,59553.34,0)
(4095,15760880,Edman,513,France,Male,29,10,0.0,2,0,1,25514.77,0)
```

Mongo Queries

1. **Finding reliable customers : people who have more than 800 credit scores and a bank balance of at least 100000$, and still our customers.**

db.bank.aggregate([{$**match** : {CreditScore :{$gt:800},Balance:{$gt:100000},Exited:{$eq:0}}},{**$count**:"reliable_customers"}}])

```
> db.bank.aggregate([{$match : {CreditScore :{$gt : 800},Balance:{$gt:100000},Exited:{$eq:0}}},{$count:"reliable_cust
omers"}])
{ "reliable_customers" : 239 }
>
```

2. **Converting the reliable customer information into a separate collection using $out**

db.bank.aggregate(
[{**$match**:{CreditScore:{$gt:800},Balance:{$gt:100000},Exited:{$eq:0}}},
{**$count**:"reliable_customers"},
{**$out:**"Reliable"}}])

```
> db.bank.aggregate([{$match : {CreditScore :{$gt : 800},Balance:{$gt:100000},Exited:{$eq:0}}},{$out:"Reliable"}])
> show collections
Reliable
bank
>
```

3. **Finding out the age group of people who have less variety of tenure to target them with more tenure oriented plans.**

> db.bank.aggregate([
{**$group**:{_id:"$Age",
Tenure:{**$push**:"$Tenure"}}},
{**$out**:"Tenure_acc_age"}])
>

4. db.Tenure_acc_age.find({"**$expr**":{$lte:[{**$size**:"$Tenure"},3]}}).pretty()

```
> db.bank.aggregate([{$group:{_id:"$Age",Tenure:{$push:"$Tenure"}}},{$out:"Tenure_acc_age"}])
> db.Tenure_acc_age.find({"$expr":{$lte:[{$size:"$Tenure"},3]}}).pretty()
{ "_id" : 92, "Tenure" : [ 3, 1 ] }
{ "_id" : 83, "Tenure" : [ 6 ] }
{ "_id" : 80, "Tenure" : [ 4, 8, 3 ] }
{ "_id" : 84, "Tenure" : [ 8, 8 ] }
{ "_id" : 88, "Tenure" : [ 10 ] }
{ "_id" : 85, "Tenure" : [ 10 ] }
{ "_id" : 82, "Tenure" : [ 2 ] }
>
```

5. **Adding tenure as an index and query faster**

> db.Tenure_acc_age.createIndex({"Tenure":1})

```
> db.Tenure_acc_age.createIndex({"Tenure":1})
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

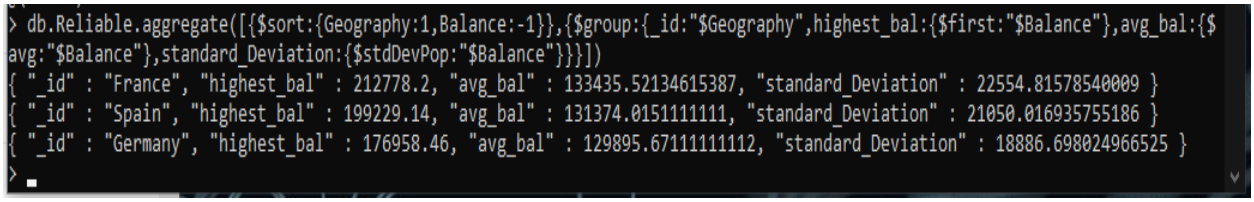
>db.Tenure_acc_age.find({Tenure:1},{Tenure:0}).pretty()
>db.Tenure_acc_age.find({Tenure:1},{Tenure:0}).count()

```
> db.Tenure_acc_age.find({Tenure:1},{Tenure:0}).pretty()
{ "_id" : 92 }
{ "_id" : 33 }
{ "_id" : 44 }
{ "_id" : 28 }
{ "_id" : 59 }
{ "_id" : 49 }
{ "_id" : 72 }
{ "_id" : 26 }
{ "_id" : 69 }
{ "_id" : 46 }
{ "_id" : 36 }
{ "_id" : 62 }
{ "_id" : 29 }
```

```
> db.Tenure_acc_age.find({Tenure:1},{Tenure:0}).count()
61
>
```

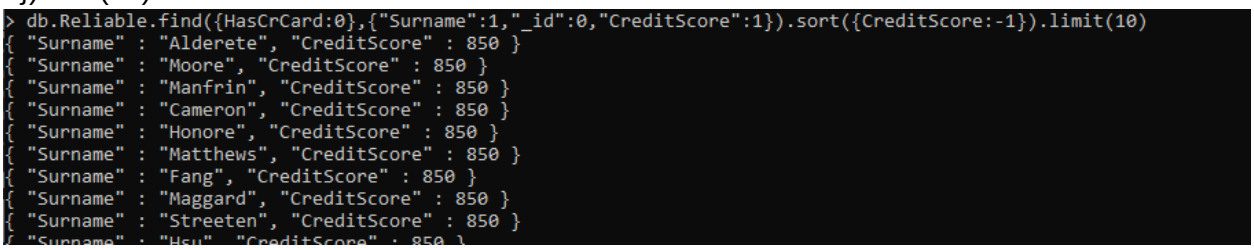
6. **From the reliable customers, find the highest balance in an account for each geography, comparing it with avg balance and checking the standard deviation.**

```
>
db.Reliable.aggregate([ {$sort:{Geography:1,Balance:-1}},
{       $group:{_id:"$Geography",
        highest_bal:{$first:"$Balance"},
        avg_bal:{$avg:"$Balance"},
        standard_Deviation:{$stdDevPop:{"$Balance"}}
        }
}])
```



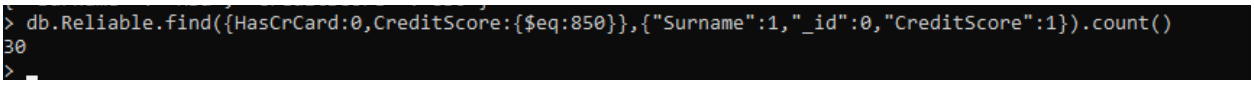
7. **From reliable customers finding out people who don't have a credit card but a high credit score to target them with new plans.**

```
>
db.Reliable.find({HasCrCard:0},{"Surname":1,"_id":0,"CreditScore":1}).sort({CreditScore:-1}).limit(10)
```

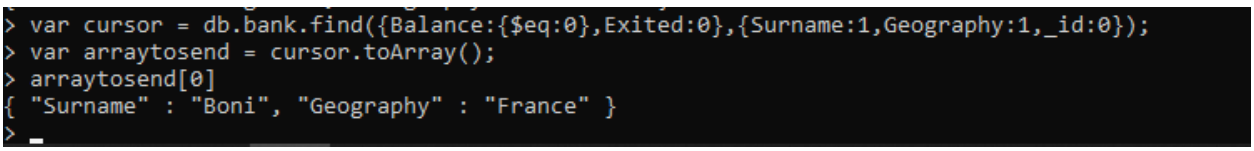


The count of people =>

```
>db.Reliable.find({HasCrCard:0,CreditScore:{$eq:850}},{"Surname":1,"_id":0,"CreditScore":1}).count()
```



8. **Creating array of people who have zero balance currently and still present in our bank**



Hive and Pig

Q1. Find the percentile of Age column?

```
hive> select percentile_approx(Age,0.5) from churn;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
 using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20211108103738_f69b7878-ae82-429e-9bc1-46197eb8e39e
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2021-11-08 10:37:41,527 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1218207873_0001
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 5552632 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
36.81799163179916
Time taken: 2.617 seconds, Fetched: 1 row(s)
```

Q2. Find Histogram numeric of column (EstimatedSalary)?

```
hive> select histogram_numeric(EstimatedSalary,10) from churn;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
 using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20211108104349_bf53a6b7-b787-409c-a2aa-b7d56a741b5a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2021-11-08 10:43:51,421 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local1412721454_0006
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 19249792 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
[{"x":11155.732039281856,"y":2231.0},{"x":33817.39016549394,"y":2172.0},{"x":55815.44833997756,"y":21
87.0},{"x":76659.18928159874,"y":2061.0},{"x":96923.7988142205,"y":2107.0},{"x":117801.60869053197,"y
":2123.0},{"x":138264.28863345634,"y":2003.0},{"x":157212.93639107022,"y":1719.0},{"x":175083.8583108
3843,"y":1806.0},{"x":192141.35616357593,"y":1591.0}]
Time taken: 1.58 seconds, Fetched: 1 row(s)
hive>
```

Q3. Find Standard Deviation of CreditScore Column?

```
hive> select stddev_pop(CreditScore) from churn;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider
 using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = hduser_20211108110057_6b9c8a7a-1486-45dd-b0ba-4ef2e4e08b14
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2021-11-08 11:00:58,890 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local671736245_0010
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 30207520 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
96.64846595037068
Time taken: 1.605 seconds, Fetched: 1 row(s)
hive>
```

Q4. Map Ages on a range of 10

```
hive> select CustomerId,Surname,concat(cast(floor((Age)/20)*20 as string),'-',cast(floor((Age)/20)*20
+9 as string)) from churn where Age BETWEEN 20 AND 29;
OK
15656148        Obinna  20-29
15592389        H?      20-29
15737173        Andrews 20-29
15691483        Chin    20-29
15788218        Henderson       20-29
15568982        Hao     20-29
15738191        Maclean 20-29
15656300        Lucciano        20-29
15732963        Clements        20-29
15602280        Martin  20-29
15773469        Clark   20-29
15592461        Jackson 20-29
15755648        Pisano  20-29
15620344        McKee   20-29
15779052        Ballard 20-29
15780961        Cavenagh        20-29
15762418        Gant    20-29
15767954        Osborne 20-29
15640635        Capon   20-29
15693683        Yuille  20-29
15604348        Allard  20-29
15609618        Fanucci 20-29
15779659        Zetticci        20-29
15591607        Fernie  20-29
15642004        Alekseeva       20-29
15800703        Madukwe 20-29
15705707        Bennelong       20-29
15623595        Clayton 20-29
15692132        Wilkinson       20-29
15658929        Taverner        20-29
15724623        Taubman 20-29
15611325        Wood    20-29
15587562        Hawkins 20-29
15613172        Sun     20-29
15679200        Crawford        20-29
15785542        Kornilova       20-29
15605461        Lucas   20-29
15723886        Fiore   20-29
15711540        Pacheco 20-29
15795149        Stevens 20-29
15685500        Glazkov 20-29
15599792        Dimauro 20-29
```

MongoDB

Q1. Total Balance based on Geography (using MapReduce)

```
>
> var map1=function() {emit(this.Geography,this.Balance);};
> var reduce1=function(key1,value1) {return Array.sum(value1);};
> db.project1.mapReduce(map1,reduce1,{out:"Geography_Balance"})
{ "result" : "Geography_Balance", "ok" : 1 }
> db.Geography_Balance.find().limit(5);
{ "_id" : "France", "value" : 311332479.49000007 }
{ "_id" : "Spain", "value" : 153123552.0099998 }
{ "_id" : "Germany", "value" : 300402861.38000035 }
> db.Geography_Balance.find();
```

Q2. Average Credit Score based on Age (using MapReduce)

```
> var map2=function() {emit(this.Age,this.CreditScore);};
> var reduce2=function(key2,value2) {return Array.avg(value2);};
> db.project1.mapReduce(map2,reduce2,{out:"Age_CS"})
{ "result" : "Age_CS", "ok" : 1 }
> db.Age_CS.find().limit(10);
{ "_id" : 80, "value" : 695 }
{ "_id" : 43, "value" : 655.989898989899 }
{ "_id" : 26, "value" : 648.785 }
{ "_id" : 58, "value" : 653.2537313432836 }
{ "_id" : 65, "value" : 640.7777777777778 }
{ "_id" : 37, "value" : 651.7782426778243 }
{ "_id" : 72, "value" : 661.1904761904761 }
{ "_id" : 53, "value" : 637.9459459459459 }
{ "_id" : 47, "value" : 658.7828571428571 }
{ "_id" : 84, "value" : 472.5 }
>
```

Q3. Total number of Active Members between the credit score 300 and 500

```
> db.project1.aggregate([{$match:{CreditScore:{$gte : 300,$lte : 500}}},{$group:{_id:null,total:{$sum:"$Is
ActiveMember"}}}]);
{ "_id" : null, "total" : 301 }
>
```

Q4. Average Credit Score with Age greater than or equal to 50 and Gender : Female

```
> db.project1.aggregate([{$match:{ $and:[{"Age":{$gte : 50}},{"Gender":"Female"}]}},{$group:{_id:null,av
erage:{$avg:"$CreditScore"}}}]);
{ "_id" : null, "average" : 648.0543806646525 }
>
```

Q5. Number of Active Members with credit score greater than 600 in France

```
> db.project1.find({"IsActiveMember":1,"Geography":"France","CreditScore":{$gt:600}}).count();
1833
>
```

MAP REDUCE PROGRAM

```
import java.io.IOException;

import java.util.*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.conf.*;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.*;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class AvBalance{

 public static class Map extends Mapper<LongWritable, Text, Text, FloatWritable> {

    private FloatWritable balance = new FloatWritable();

    private Text geography= new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {

        String line = value.toString();

        String details[] = line.split("\t");

        if(details.length>7){

                geography.set(details[4]);

                float i = Float.parseFloat(details[13]);
                balance.set(i);

        }
```

```
        context.write(geography, balance);
    }
}
 public static class Reduce extends Reducer<Text, FloatWritable, Text, FloatWritable> {

    public void reduce(Text key, Iterable<FloatWritable> values, Context context)
      throws IOException, InterruptedException {
        float sum= 0;
        int n=0;
        for (FloatWritable val : values){
                n +=1;
                sum+=val.get();
        }
        float avg = sum/n;
       context.write(key, new FloatWritable(avg));
    }
}
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = new Job(conf, "AvBalance");
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(FloatWritable.class);
    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);
    job.setInputFormatClass(TextInputFormat.class);
```

```
    job.setOutputFormatClass(TextOutputFormat.class);
   FileInputFormat.addInputPath(job, new Path(args[0]));

    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);

 }

}
```

```
hduser@kamal-VirtualBox:~$ hdfs dfs -cat /user/hduser/output1/part-r-00000
21/11/15 12:53:31 WARN util.NativeCodeLoader: Unable to load native-hadoop libr
ary for your platform... using builtin-java classes where applicable
France  62092.61
Germany 119730.164
Spain   61818.152
hduser@kamal-VirtualBox:~$
```

HIVE AND PIG

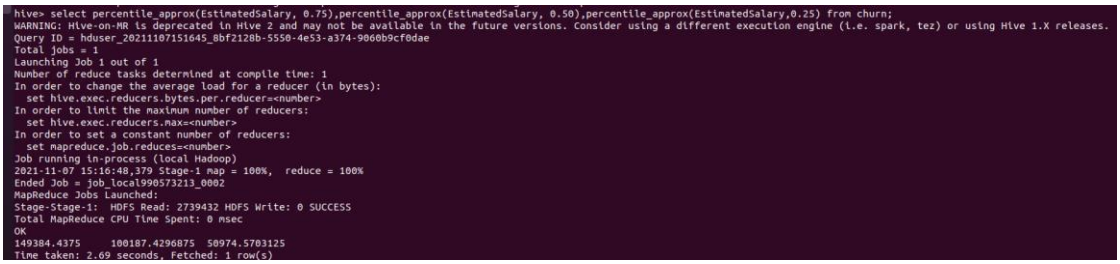
1. Finding the maximum and minimum balance in the account

   ✞ select max(balance), min(balance) from churn;

```
hive> select max(balance),min(balance) from churn;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a differ
Query ID = hduser_20211107151807_4cbd7236-3541-4f60-b2ca-a2370969961f
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2021-11-07 15:18:10,204 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local81133635_0003
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 4109148 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
250898.1        0.0
Time taken: 2.756 seconds, Fetched: 1 row(s)
```

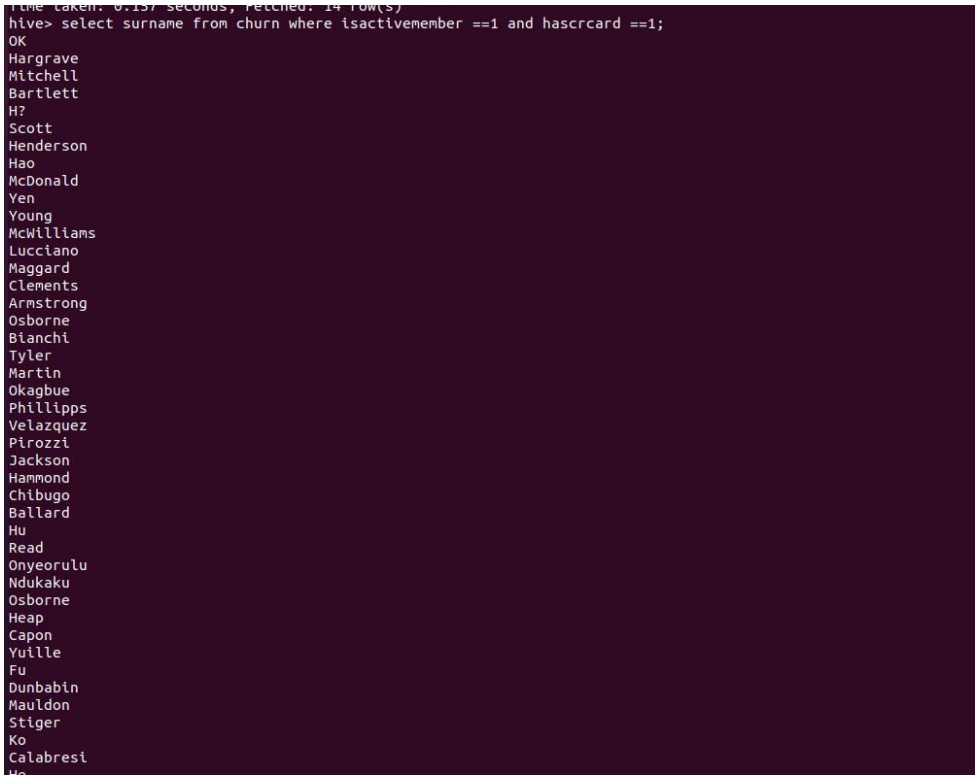
2. Finding the 25th ,50th and 75th quartile of estimated salary

   ✞ select percentile_approx(EstimatedSalary, 0.75),percentile_approx(EstimatedSalary, 0.50), percentile_approx(EstimatedSalary, 0.25) from churn;



3. Name of people who is both an active member and has credit card

- ✞ select surname from churn where isactivemember ==1 and hascrcard ==1;



4. Finding the total bank balance of the region with highest balance

- ✞ select geography, max(balance) as ba from churn where geography = ‘spain’ group by geography;

```
hive> select geography,max(balance) as ba from churn group by geography where geography=="Spain";
FAILED: ParseException line 1:66 missing EOF at 'where' near 'geography'
hive> select geography,max(balance) as ba from churn where geography ="Spain"  group by geography;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a
Query ID = hduser_20211107153658_ac9b5292-eaf5-4644-8b4f-adfbdd839cd2
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2021-11-07 15:37:00,707 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local334578822_0012
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 19184216 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
OK
Spain   250898.1
Time taken: 1.974 seconds, Fetched: 1 row(s)
```

5. Name of the people with the highest credit score

- ✞ select surname, creditscore from churn where creditscore =850;

```
hive> select surname,creditscore from churn where creditscore=850;
OK
Mitchell        850
Armstrong       850
Rozier  850
Chiemezie       850
Stevenson       850
Welch   850
Simmons 850
Pino    850
Uchechukwu      850
Olisanugo       850
Macartney       850
Madukwe 850
Ritchie 850
Alderete        850
Douglas 850
Summers 850
Cameron 850
Hsu     850
Maclean 850
Zikoranaudodimma        850
Kline   850
Philip  850
Paterson        850
Genovese        850
Sorokina        850
Longo   850
Morres  850
Degtyarev       850
Shelton 850
Matthews        850
Kaeppel 850
Honore  850
Moore   850
Onyemachukwu    850
Le Grand        850
Kao     850
Mackay  850
P'eng   850
```

6. Name and balance of the person with the highest bank balance

✞ select surname, balance from churn where balance = 250898.1;

```
hive> select surname,balance from churn where balance ==250898.1
    > ;
OK
Lo      250898.1
```

MONGODB

1. Finding the total number of documents using built-in queries

```
> db.churn.estimatedDocumentCount()
10000
>
```

2. Average estimated salary of males and females respectively using aggregate and group

```
> db.churn.find( { $expr: { $gt: [ "$EstimatedSalary" , "$Balance" ] } } ).pretty()
{
        "_id" : ObjectId("61a3ab9b576fd4da66110047"),
        "RowNumber" : 1,
        "CustomerId" : 15634602,
```

3. Finding the maximum credit score grouping by geography using aggregate functions

```
> db.churn.aggregate(
...     [
...       {
...         $group:
...           {
...             _id: "$Geography",
...             maxTotalAmount:{$max:"$CreditScore"}}
...       }
...     ])
{ "_id" : "France", "maxTotalAmount" : 850 }
{ "_id" : "Spain", "maxTotalAmount" : 850 }
{ "_id" : "Germany", "maxTotalAmount" : 850 }
>
```

4. Finding a potentially important customer who closed the account recently

```
> db.churn.find({"Age": { $gt: 60} , "HasCrCard": 1, "CreditScore": {$gt: 800} , "Exited": 1 , "Balance": {$gt: 100000}}
).pretty()
{
        "_id" : ObjectId("61a3ab9b576fd4da66110ece"),
        "RowNumber" : 3720,
        "CustomerId" : 15577999,
        "Surname" : "Sleeman",
        "CreditScore" : 850,
        "Geography" : "France",
        "Gender" : "Female",
        "Age" : 62,
        "Tenure" : 1,
        "Balance" : 124678.35,
        "NumOfProducts" : 1,
        "HasCrCard" : 1,
        "IsActiveMember" : 0,
        "EstimatedSalary" : 70916,
        "Exited" : 1
}
>
```

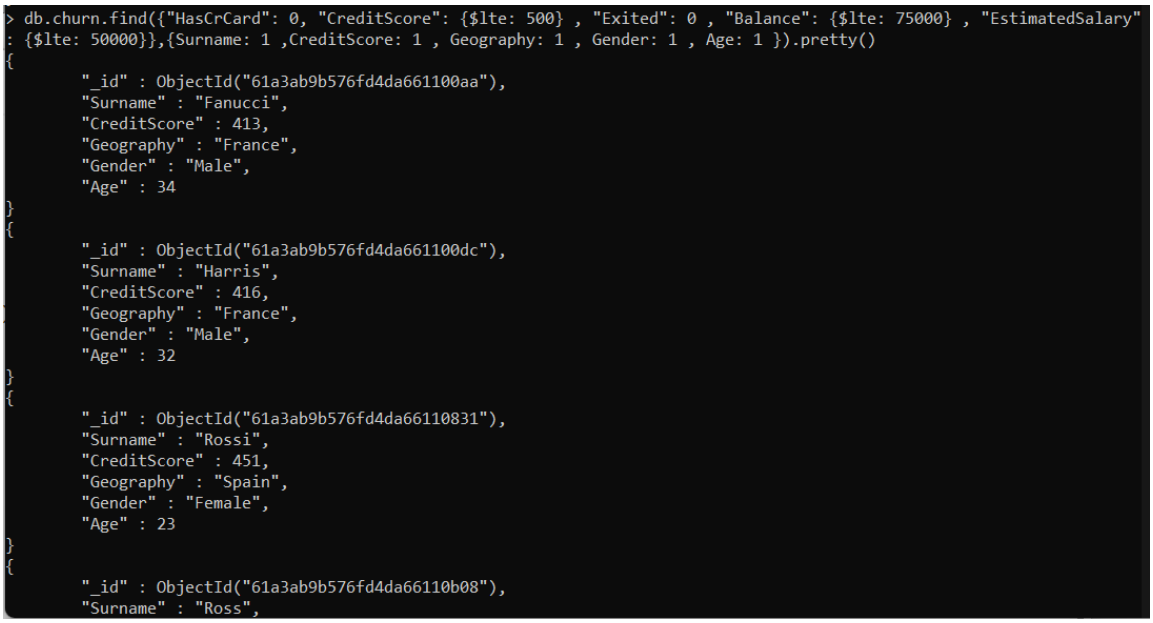
5. Finding the maximum and minimum balance and grouping it by geography using aggregate function

```
> db.churn.aggregate(
...     [
...       {
...         $group:
...           {
...             _id: "$Geography",
...             MaxBalance: {$max: "$Balance"},
...             MinBalance: {$min: "$Balance"}
...
...           }
...       }
...     ]
... )
{ "_id" : "France", "MaxBalance" : 238387.56, "MinBalance" : 0 }
{ "_id" : "Spain", "MaxBalance" : 250898.09, "MinBalance" : 0 }
{ "_id" : "Germany", "MaxBalance" : 214346.96, "MinBalance" : 27288.43 }
>
```

6. Finding customers whose Estimated Salary is greater than Balance using **$expr Evaluation Query Operators**

```
> db.churn.find( { $expr: { $gt: [ "$EstimatedSalary" , "$Balance" ] } } ).pretty()
{
        "_id" : ObjectId("61a3ab9b576fd4da66110047"),
        "RowNumber" : 1,
        "CustomerId" : 15634602,
        "Surname" : "Hargrave",
        "CreditScore" : 619,
        "Geography" : "France",
        "Gender" : "Female",
        "Age" : 42,
        "Tenure" : 2,
        "Balance" : 0,
        "NumOfProducts" : 1,
        "HasCrCard" : 1,
        "IsActiveMember" : 1,
        "EstimatedSalary" : 101348.88,
        "Exited" : 1
}
```

7. Finding the potentially least important customers and printing their necessary detail



```
> db.churn.find({"HasCrCard": 0, "CreditScore": {$lte: 500} , "Exited": 0 , "Balance": {$lte: 75000} , "EstimatedSalary"
: {$lte: 50000}},{Surname: 1 ,CreditScore: 1 , Geography: 1 , Gender: 1 , Age: 1 }).pretty()
{
        "_id" : ObjectId("61a3ab9b576fd4da661100aa"),
        "Surname" : "Fanucci",
        "CreditScore" : 413,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 34
}
{
        "_id" : ObjectId("61a3ab9b576fd4da661100dc"),
        "Surname" : "Harris",
        "CreditScore" : 416,
        "Geography" : "France",
        "Gender" : "Male",
        "Age" : 32
}
{
        "_id" : ObjectId("61a3ab9b576fd4da66110831"),
        "Surname" : "Rossi",
        "CreditScore" : 451,
        "Geography" : "Spain",
        "Gender" : "Female",
        "Age" : 23
}
{
        "_id" : ObjectId("61a3ab9b576fd4da66110b08"),
        "Surname" : "Ross",
```