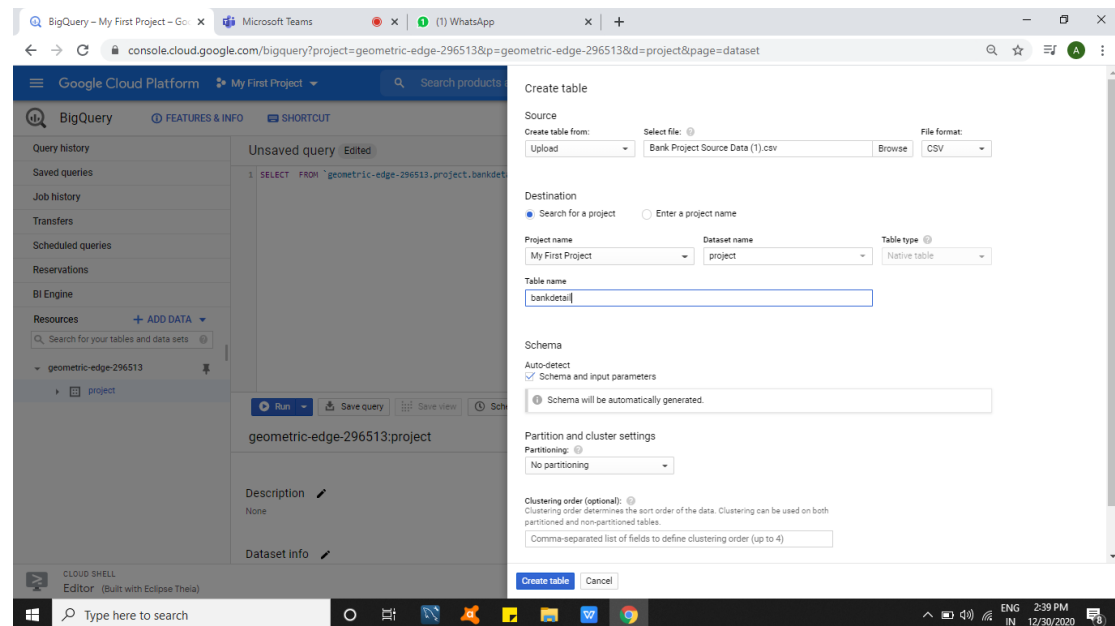


Project Analysis & Reporting System of Bank Account Details

Big query:

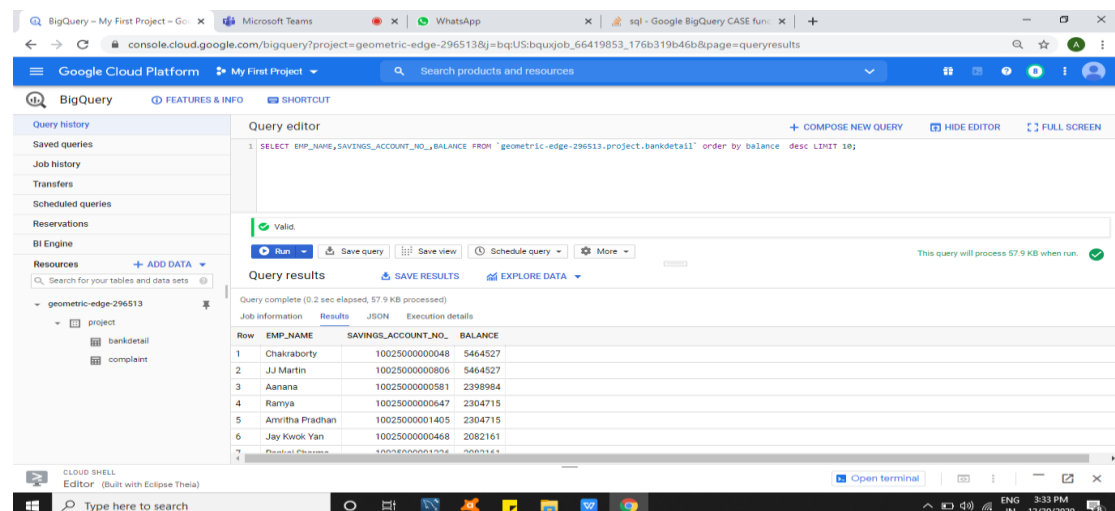
1. Analyse the dataset and find the top 10 account and its details based on salary



Code:

```
SELECT EMP_NAME,SAVINGS_ACCOUNT_NO_, SAVINGS_ACCOUNT_NO_BALANCE FROM  
'geometric-edge-296513.project.bankdetail' order by balance desc LIMIT 10;
```

Output:



6. Analyse the dataset and find the Customer's % increase in the average bank balance for 2 years.

Code:

```
SELECT EMP_NAME,SAVINGS_ACCOUNT_NO_,BALANCE,
((AVG_BALANCE_YEAR_3- AVG_BALANCE_YEAR1)/(AVG_BALANCE_YEAR1+
AVG_BALANCE_YEAR_2+ AVG_BALANCE_YEAR_3)) * 100 as average
FROM `geometric-edge-296513.project.bankdetail` where BALANCE is not null and
AVG_BALANCE_YEAR_3> AVG_BALANCE_YEAR1;
```

Output:

The screenshot shows the Google Cloud BigQuery console. The query editor contains the following SQL query:

```
1 SELECT EMP_NAME,SAVINGS_ACCOUNT_NO_,BALANCE,
2 ((AVG_BALANCE_YEAR_3- AVG_BALANCE_YEAR1)/(AVG_BALANCE_YEAR1+
3 AVG_BALANCE_YEAR_2+ AVG_BALANCE_YEAR_3)) * 100 as average
4 FROM `geometric-edge-296513.project.bankdetail` where BALANCE is not null and
5 AVG_BALANCE_YEAR_3> AVG_BALANCE_YEAR1;
```

The query results are displayed in a table with 8 rows. The columns are Row, EMP_NAME, SAVINGS_ACCOUNT_NO_, BALANCE, and average.

Row	EMP_NAME	SAVINGS_ACCOUNT_NO_	BALANCE	average
1	Nagesh	10025000000002	2900	15.598762630921925
2	Gawand	10025000000010	59300	53.76007079515496
3	Manjunath	10025000000012	39052	60.66015611326962
4	Sharma	10025000000027	454527	16.263823756558573
5	Ambhore	10025000000029	60000	50.83603277439024
6	Mundargi	10025000000040	10000	41.51449817397809
7	mithila rai	10025000000046	52150	63.96437251993756
8	Ralendran	10025000000057	80000	39.49217237373458

9. Analyse the dataset and find the locations where more banking portals should be opened(number of employees for a particular location and number of bank branches shall be same, if less they need more banking portals)

Code:

```
with input as (SELECT COMPANY_LOCATION,count(COMPANY_LOCATION) as companycounted FROM
`geometric-edge-296513.project.bankdetail` where COMPANY_LOCATION is not null group by
COMPANY_LOCATION )SELECT BANK_BRANCH ,count( BANK_BRANCH ) as
bankcounted,a.companycounted FROM `geometric-edge-296513.project.bankdetail`,input as a
where BANK_BRANCH is not null and BANK_BRANCH =a.COMPANY_LOCATION group by
BANK_BRANCH,a.COMPANY_LOCATION,a.companycounted having a.companycounted>bankcounted
LIMIT 1000;
```

Output:

The screenshot displays the Google Cloud Platform BigQuery console. The top navigation bar shows the project name 'My First Project'. The left sidebar contains a 'Resources' section with a search bar and a list of tables: 'bankdetail' and 'complaint'. The main area is the 'Query editor', which contains a SQL query. Below the editor, the 'Query results' section shows a table with 5 rows and 4 columns: 'Row', 'BANK_BRANCH', 'bankcounted', and 'companycounted'. The query is valid and has been executed, processing 37.2 KB of data in 0.6 seconds.

Row	BANK_BRANCH	bankcounted	companycounted
1	Bangalore	184	278
2	Chennai	64	83
3	Gandhi Nagar	14	21
4	Navi Mumbai	166	190
5	Pune	52	69

11. Analyse the dataset and count the number of customers belonging to a particular type of account.

Code:

```
select a.CD_count,b.FD_count,c.OD_count,d.CURRENT_count,e.saving_count from
(SELECT count(CD_ACC_NO) as CD_count FROM
`geometric-edge-296513.project.bankdetail` where CD_ACC_NO not like "NULL")
a,(SELECT count(FD_ACC_NO) as FD_count FROM
`geometric-edge-296513.project.bankdetail` where FD_ACC_NO not like "NULL") as
b,(SELECT count(OD_ACC_NO) as OD_count FROM
`geometric-edge-296513.project.bankdetail` where OD_ACC_NO not like "NULL") as
c,(SELECT count(CURRENT_ACC_NO) as CURRENT_count FROM
`geometric-edge-296513.project.bankdetail` where CURRENT_ACC_NO not like "NULL")
as d,(SELECT count( SAVINGS_ACCOUNT_NO_) as saving_count FROM
`geometric-edge-296513.project.bankdetail` where SAVINGS_ACCOUNT_NO_!=0) as e ;
```

Output:

The screenshot shows the Google Cloud Platform BigQuery console. The query editor contains a complex SQL query that joins multiple subqueries to calculate various counts. The query results are displayed in a table with the following data:

Row	CD_count	FD_count	OD_count	CURRENT_count	saving_count
1	1038	1107	470	317	2000

13. Analyse the dataset and find the employees having both CD ACC and FD ACC.

Code:

```
SELECT EMP_NAME, CD_ACC_NO, FD_ACC_NO FROM
`geometric-edge-296513.project.bankdetail` where CD_ACC_NO not like "NULL" and
FD_ACC_NO not like "NULL" LIMIT 1000
```

Output:

The screenshot shows the Google Cloud Platform BigQuery console with a query that filters for employees having both CD_ACC_NO and FD_ACC_NO. The query results are displayed in a table with the following data:

Row	EMP_NAME	CD_ACC_NO	FD_ACC_NO
1	Raghu Poojary	1112510000308	1114020000209
2	Ali	1112510000310	1114020000246
3	Nayana	1112510000320	1114020000256
4	Tikudiya	1112510000330	1114020000266
5	Kubasad	1112510000331	1114020000267
6	ganesh Rao	1112510000344	1114020000288
7	Sarthi	1112510000349	1114020000293
8	Praikata (Gannarhar)	1112510000383	1114020000379

Dataprocedure:

4. Analyse the company and find the count of its employees registering for the bank.

Code:

```
package mypack;

import java.io.IOException;
import java.util.Iterator;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;

public class location {
    public static class Map extends MapReduceBase implements Mapper<LongWritable ,
    Text, Text, IntWritable>
    {

        public void map(LongWritable key, Text value, OutputCollector<Text,
    IntWritable> output, Reporter reporter) throws IOException {
            String line = value.toString();
            String [] count=line.split(",");

            output.collect(new Text(count[4]), new IntWritable(1));

        }
    }

    public static class Reduce extends MapReduceBase implements Reducer< Text, IntWritable,
    Text, IntWritable > {

        public void reduce( Text key, Iterator <IntWritable> values,
            OutputCollector<Text, IntWritable> output, Reporter reporter) throws
    IOException {
            int sum_result=0;
```

```

        while (values.hasNext()) {
            sum_result+=values.next().get();
        }
        output.collect(key, new IntWritable(sum_result));
    }
}

public static void main(String args[])throws Exception {
    JobConf conf = new JobConf(location.class);

    conf.setJobName("max_count");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);

    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
}
}

```

Output:

```

20/12/30 10:59:57 INFO client.RMProxy: Connecting to ResourceManager at cluster-0467-m/10.142.0.38:8032
20/12/30 10:59:57 INFO client.AHSProxy: Connecting to Application History server at cluster-0467-m/10.142.0.38:10200
20/12/30 10:59:58 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface to remedy this.
20/12/30 10:59:58 INFO mapred.FileInputFormat: Total input files to process : 1
20/12/30 10:59:58 INFO mapreduce.JobSubmitter: number of splits:15
20/12/30 10:59:58 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1609311045231_0005
20/12/30 10:59:59 INFO impl.YarnClientImpl: Submitted application application_1609311045231_0005
20/12/30 10:59:59 INFO mapreduce.Job: The url to track the job: http://cluster-0467-m:8088/proxy/application_1609311045231_0005/
20/12/30 10:59:59 INFO mapreduce.Job: Running job: job_1609311045231_0005
20/12/30 11:00:06 INFO mapreduce.Job: Job job_1609311045231_0005 running in uber mode : false
20/12/30 11:00:06 INFO mapreduce.Job: map 0% reduce 0%
20/12/30 11:00:15 INFO mapreduce.Job: map 13% reduce 0%
20/12/30 11:00:18 INFO mapreduce.Job: map 20% reduce 0%
20/12/30 11:00:19 INFO mapreduce.Job: map 33% reduce 0%
20/12/30 11:00:22 INFO mapreduce.Job: map 47% reduce 0%
20/12/30 11:00:29 INFO mapreduce.Job: map 60% reduce 0%
20/12/30 11:00:30 INFO mapreduce.Job: map 80% reduce 0%
20/12/30 11:00:34 INFO mapreduce.Job: map 87% reduce 0%
20/12/30 11:00:36 INFO mapreduce.Job: map 100% reduce 0%
20/12/30 11:00:45 INFO mapreduce.Job: map 100% reduce 20%
20/12/30 11:00:46 INFO mapreduce.Job: map 100% reduce 40%
20/12/30 11:00:47 INFO mapreduce.Job: map 100% reduce 60%
20/12/30 11:00:49 INFO mapreduce.Job: map 100% reduce 80%
20/12/30 11:00:50 INFO mapreduce.Job: map 100% reduce 100%
20/12/30 11:00:50 INFO mapreduce.Job: Job job_1609311045231_0005 completed successfully
20/12/30 11:00:50 INFO mapreduce.Job: Counters: 50
File System Counters
  FILE: Number of bytes read=448
  FILE: Number of bytes written=1161386
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
HDFS: Number of bytes read=649839
HDFS: Number of bytes written=151
HDFS: Number of read operations=70
HDFS: Number of large read operations=0
HDFS: Number of write operations=15
Job Counters
  Killed reduce tasks=1
  Launched map tasks=15
  Launched reduce tasks=5
  Data-local map tasks=15
  Total time spent by all maps in occupied slots (ms)=467292
  Total time spent by all reduces in occupied slots (ms)=154488

```

```

Total committed heap usage (bytes)=6969360384
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=648609
File Output Format Counters
Bytes Written=151
harshital262@cluster-0467-m:~$ hdfs dfs -cat /output
cat: '/output': Is a directory
harshital262@cluster-0467-m:~$ hdfs dfs -ls /output
Found 6 items
-rw-r--r-- 2 harshital262 hadoop 0 2020-12-30 11:00 /output/_SUCCESS
-rw-r--r-- 2 harshital262 hadoop 42 2020-12-30 11:00 /output/part-00000
-rw-r--r-- 2 harshital262 hadoop 36 2020-12-30 11:00 /output/part-00001
-rw-r--r-- 2 harshital262 hadoop 21 2020-12-30 11:00 /output/part-00002
-rw-r--r-- 2 harshital262 hadoop 35 2020-12-30 11:00 /output/part-00003
-rw-r--r-- 2 harshital262 hadoop 17 2020-12-30 11:00 /output/part-00004
harshital262@cluster-0467-m:~$ hdfs dfs -ls /output/part-00000
-rw-r--r-- 2 harshital262 hadoop 42 2020-12-30 11:00 /output/part-00000
harshital262@cluster-0467-m:~$ hdfs dfs -cat /output/part-00000
Cap Gemini 2
Qualcomm 30
TCS 32
Wipro 202
harshital262@cluster-0467-m:~$ hdfs dfs -cat /output/part-00001
Cognizant 100
Infosys 55
Seimens 31
harshital262@cluster-0467-m:~$ hdfs dfs -cat /output/part-00002
Accenture 96
IBM 601
harshital262@cluster-0467-m:~$ hdfs dfs -cat /output/part-00003
COMPANY 1
Global Edge 38
IGATE 749
harshital262@cluster-0467-m:~$ hdfs dfs -cat /output/part-00004
Tech Mahindra 64
harshital262@cluster-0467-m:~$
harshital262@cluster-0467-m:~$
harshital262@cluster-0467-m:~$

```

7. Analyse the dataset and find the company having most number of employees registering for the bank.

Code:

```

create table input2(EMPNAME String,EMPAGE String,MARITALSTATUS String,EMPID
String,COMPANY String,COMPANYLOCATION String,SPECIALIZATION
String,DESIGNATIONID String,PHONENO String,BANKBRANCH String,EXPERIENCE
String,EDUCATION String,VOTERID String,PANCARD String,HOUSING String,Loan
String,LOANTYPEID String,REWARDPOINTS String,SAVINGSACCOUNTNO
String,CDACCNO String,FDACCNO String,ODACCNO String,CURRENTACCNO
String,BALANCE String,CREDITCARDNO String,DEBITCARDNO
String,INTERNETBANKING String,AVGBALANCEYEAR1
String,AVGBALANCEYEAR2 String,AVGBALANCEYEAR3
String,ADDRESSOFEMPLOYEES String,ACCOUNTOPENINGDATE String)row format
delimited fields terminated by ',' stored as textfile;

```

```
load data local inpath '/home/ameer17_08/bank.csv' into table input2;
```

```
select company,count(*) as counted from input2 group by company order by counted desc
limit 1;
```

```
ssh.cloud.google.com/projects/geometric-edge-296513/zones/us-east-1-b/instances/cluster-0467-m?authuser=0&hl=en_US&projectNumber=398943591221&useAdminProxy=true
```

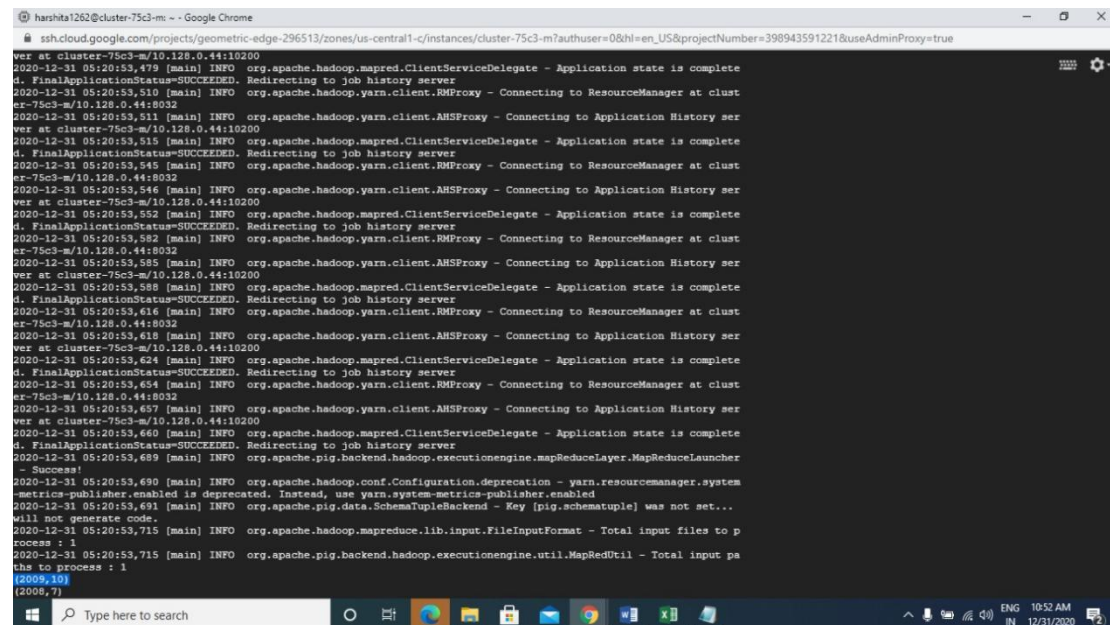
```
>  
hive> create table databank(EMPNAME String,EMPAJE int,MARITALSTATUS String,EMFID String,COMPANY String,COMPANYLOCATION String,SPECIALIZATION String,DESIGNATIONID String,INSA  
O String,BANKBRANCH String,EXPERIENCE String,EDUCATION String,VOTERID String,BANCARD String,HOUSING String,Loan String,LOANTYPEID String,REWARDPOINTS String,SAVINGSACCOUNTNO  
tring,CDACCNO String,FDCACCNO String,ODACCNO String,CURRENTACCNO String,BALANCE String,CREDITCARDNO String,DEBITCARDNO String,INTERNETBANKING String,AVGBALANCEYEAR1 String,AVG  
BALANCEYEAR2 String,AVGBALANCEYEAR3 String,ADDRESSOFEMPLOYEES String,ACCONTOPEXPIRINGDATE String)row format delimited fields terminated by ',' stored as textfile;  
OK  
Time taken: 0.261 seconds  
hive> Load data local inpath '/home/harshital262/bank.csv' into table databank;  
Loading data to table default.databank  
OK  
Time taken: 1.057 seconds  
hive> select COMPANY,count(*) as empcount from databank group by COMPANY order by empcount desc limit 1:  
Query ID = harshital262_20201230112039_fb19e7b1-d0be-4beb-bd5c-c37d5947ad0e  
Total Jobs = 1  
Launching Job 1 out of 1  
Tex session was closed. Reopening...  
Session re-established.  
Status: Running (Executing on YARN cluster with App id application_1609311045231_0012)
```

```
-----  
VERTICES      MODE          STATUS    TOTAL    COMPLETED   RUNNING   PENDING   FAILED   KILLED  
Map 1 ..... Container SUCCEEDED     1         1           0         0         0         0  
Reducer 2 ..... Container SUCCEEDED     1         1           0         0         0         0  
Reducer 3 ..... Container SUCCEEDED     1         1           0         0         0         0  
-----  
VERTICES: 03/03 [=====>>>] 100% ELAPSED TIME: 6.72 s  
-----  
OK  
HIGATE 749  
Time taken: 20.741 seconds, Fetched: 1 row(s)  
hive>  
>  
>  
>  
>  
>  
>  
>  
>
```

Code:

```
group1 =group substring_data by year;
grouping= foreach group1 generate group,COUNT(substring_data.Loan)as compliant;
final = LIMIT final_result 1;
store final into 'secondprogram.pig';
dump final;
```


Output:



12. Analyse the dataset and find the maximum salary and the details of the employee having that salary for a particular specialization.

Code:

```
spark.sql("create table data(EMPNAME String,EMPAGE int,MARITALSTATUS String,EMPID String,COMPANY String,COMPANYLOCATION String,SPECIALIZATION String,DESIGNATIONID String,PHONENO String,BANKBRANCH String,EXPERIENCE String,EDUCATION String,VOTERID String,PANCARD String,HOUSING String,Loan String,LOANTYPEID String,REWARDPOINTS String,SAVINGSACCOUNTNO String,CDACCNO String,FDACCNO String,ODACCNO String,CURRENTACCNO String,BALANCE String,CREDITCARDNO String,DEBITCARDNO String,INTERNETBANKING String,AVGBALANCEYEAR1 String,AVGBALANCEYEAR2 String,AVGBALANCEYEAR3 String,ADDRESSOFEMPLOYEES String,ACCOUNTOPENINGDATE String) row format delimited fields terminated by ',' stored as textfile")
```

```
spark.sql("Load data local inpath '/bankfinal.csv' into table data")
```

```
spark.sql("create table data1(SPECIALIZATION String, BALANCE String) row format delimited fields terminated by ',' stored as textfile")
```

```
spark.sql("INSERT INTO data1 SELECT SPECIALIZATION ,MAX(BALANCE) as max FROM data where group by SPECIALIZATION ")
```

```
spark.sql("SELECT a.SPECIALIZATION, a.EMPNAME, a.EMPID ,b.max as max FROM data a join data1 b on a.BALANCE=b.max where a. SPECIALIZATION=b.
```

SPECIALIZATION and a. SPECIALIZATION!= 'SPECIALIZATION' order by max desc").show()

Output:

```
scala>
scala>
scala>
scala>
scala> spark.sql("SELECT a.SPECIALIZATION, a.EMPNAME, a.EMPID, b.BALANCE as max FROM data1 a join data1 b on a.BALANCE=b.BALANCE where a. SPECIALIZATION!=b. SPECIALIZATION and a.SPECIALIZATION != 'SPECIALIZATION' order by max desc").show()
+-----+-----+-----+-----+
| SPECIALIZATION | EMPNAME | EMPID | max |
+-----+-----+-----+-----+
| Android | EZIMALI | 6204 | 85000 |
| V&V | VISHAL CHAMARIA | 6010 | 8440 |
| V&V | SUDIKSHA KAPOOR | 6333 | 8440 |
| MySQL Server | KIRTIKUMAR JAYPAL | 3888 | 82 |
| Location Allotments | TUSHAR AGRAWAL | 24610 | 799 |
| Project Allotments | KSHITIJ RANGANATH | 24602 | 729 |
| Blackberry 10 | RAHUL TICKOO | 6132 | 6456 |
| Chrome OS App | ASHWINANDAM MALHOTRA | 6126 | 6130 |
| Windows 8.1 PC | ASHISH GANDOTRA | 6124 | 6128 |
| Windows 8.0 Mobile | RAGHAV SHARMA | 6123 | 6127 |
| Hiring | KUNDAN GELA | 25547 | 6127 |
| Windows 7.5 | SIDIN THOMAS | 6115 | 6119 |
| iOS | ISHITA KINARWALA | 6107 | 6111 |
| Step 7 lite Tool | Tejas Gnanani | 26001 | 6 |
| Communication ports | Preeti Desai | 26006 | 531235 |
| Exception Handling | ANANDITA BHUSHAN | 26026 | 26030 |
| On-site Relations | Aditya Nair | 25373 | 21274 |
+-----+-----+-----+-----+

scala>
scala>
scala>
scala>
scala>
scala>
```

Dataprep:

5. Analyse the dataset and find the Customer's membership(gold, platinum, diamond) based on the account opening date(for year 2005-2007->gold, 2008-2011->platinum, 2012-2015->diamond).

Code:

IF({ACCOUNT OPENING DATE1} >= 2005 && {ACCOUNT OPENING DATE1} <= 2007, 'Gold', IF({ACCOUNT OPENING DATE1} >= 2008 && {ACCOUNT OPENING DATE1} <= 2011, 'Platnum', IF({ACCOUNT OPENING DATE1} >= 2012 && {ACCOUNT OPENING DATE1} <= 2015, 'Diamond', 'NA'))

Output:

The screenshot shows the Google Cloud Data Studio interface. A new formula is being created for a column named 'membership'. The formula type is set to 'Single row formula'. The formula being entered is:

```
IF((ACCOUNT OPENING DATE1) >= 2005 && (ACCOUNT OPENING DATE1) <= 2007, 'Gold', IF((ACCOUNT OPENING DATE1) >= 2008 && (ACCOUNT OPENING DATE1) <= 2011, 'Platinum', IF((ACCOUNT OPENING DATE1) >= 2012 && (ACCOUNT OPENING DATE1) <= 2015, 'Diamond', 'NA'))
```

The new column name is 'membership'.

The screenshot shows the Google Cloud Data Studio interface. The details of the 'membership' column are displayed. The column is of type 'Text' and has 3 categories: Platinum, Diamond, and Gold. The quality of the data is shown as 100% Valid, 0% Mismatched, and 0% Missing. The unique values are Platinum (178), Diamond (69), and Gold (62). The patterns are Upper:Lower(6) (24), Upper:Lower(3) (62), and Show pattern details... The suggestions are Recently used and Replace missing values in membership with 'NA'.

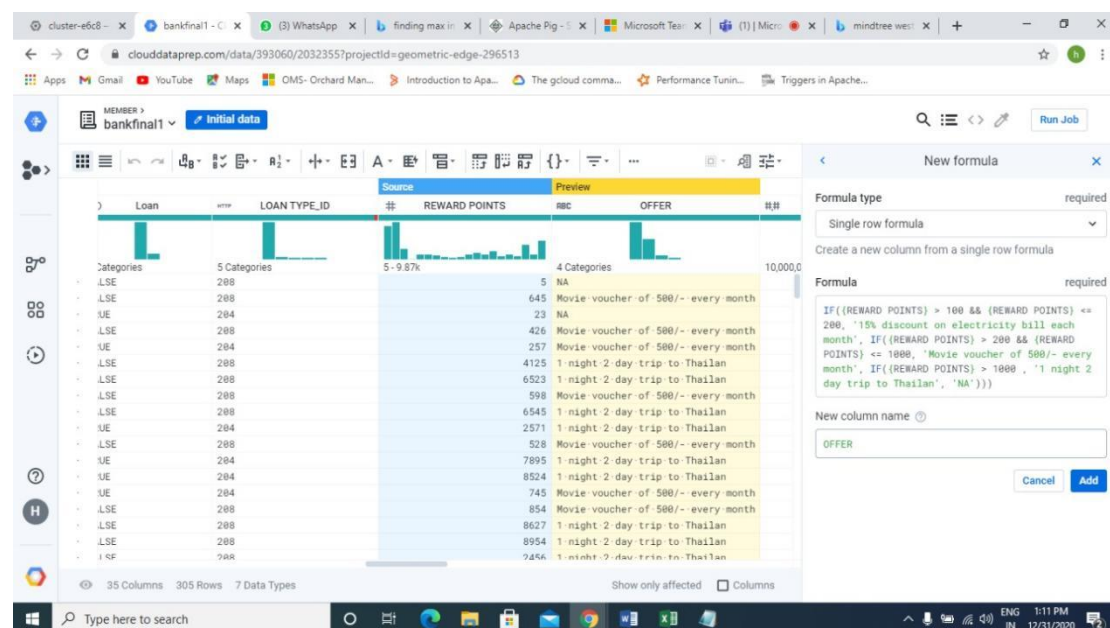
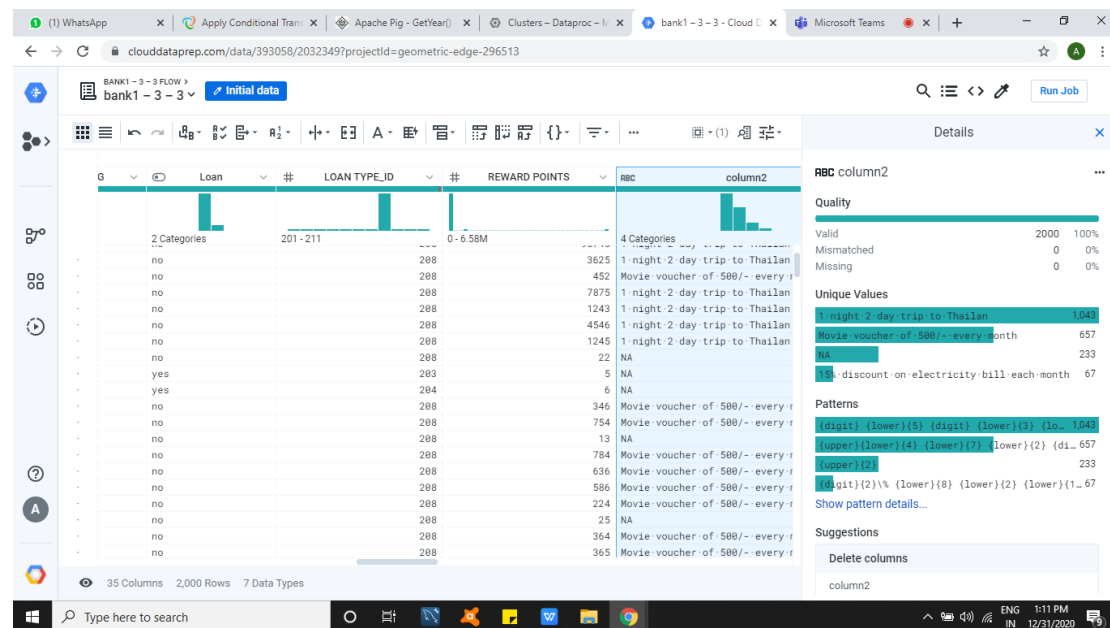
10. Analyse the dataset and provide offers to the customers(if reward points in the range of 100-200k then

- 15% discount on electricity bill each month – range : 200k-1000k
- Movie voucher of 500/- every month – range : 1000k and above
- 1 night 2 day trip to Thailand

Code:

```
IF({REWARD POINTS} >= 100 && {REWARD POINTS} <= 200000, "15% discount on electricity bill each month", IF({REWARD POINTS} >= 200000 && {REWARD POINTS} <= 1000000, "Movie voucher of 500/- every month", IF({REWARD POINTS} >= 1000000, "1 night 2 day trip to Thailand", "NA"))
```

Output:



Dataflow:

14. Analyse the dataset and find the most preferred account created by the customers

Code:

```
package org.apache.beam.examples;

import org.apache.beam.sdk.Pipeline;
import org.apache.beam.sdk.options.PipelineOptions;
```

```

import org.apache.beam.sdk.options.PipelineOptionsFactory;
import org.apache.beam.sdk.io.TextIO;
import org.apache.beam.sdk.transforms.*;
import org.apache.beam.sdk.values.*;

public class problem14 {
    private static final String CSV_HEADER =
        "Operator,Indoor_Outdoor_Travelling,Network Type,Rating,Call
Drop Category,Latitude,Longitude,State Name";
    static int count=0;
    public static void main(String[] args) {
        PipelineOptions options = PipelineOptionsFactory.fromArgs(args)
.withValidation().create();
        Pipeline pipeline = Pipeline.create(options);

        pipeline.apply("ReadAds", TextIO.read().from("src/main
/project_resources/bank1.csv"))
            .apply("FilterHeader", ParDo.of(new FilterHeaderFn(CSV_
HEADER)))
            .apply("MakeAgeKVFn", ParDo.of(new MakeAgeKVFn()))
            .apply(Sum.<String>integersPerKey())
            .apply("MAXKEY",Max.integersPerKey())
            .apply(ParDo.of(new ConvertToStringFn()))
            .apply("PrintToConsole", ParDo.of(new DoFn<String, Void
>() {
                @ProcessElement
                public void processElement(ProcessContext c) {
                    System.out.println(c.element());
                }
            }));

        //value.apply("Writeresult to file", TextIO.write().to("gs://al
binbucket/amita/output"));

        pipeline.run().waitUntilFinish();
    }

    private static class FilterHeaderFn extends DoFn<String, String> {

        private final String header;

        public FilterHeaderFn(String header) {
            this.header = header;
        }

        @ProcessElement
        public void processElement(ProcessContext c) {
            String row = c.element();

```

```

        if (!row.isEmpty() && !row.equals(this.header)) {
            c.output(row);
        }
    }
}

private static class MakeAgeKVFn extends DoFn<String, KV<String, Integer>> {

    @ProcessElement
    public void processElement(ProcessContext c) {
        String[] fields = c.element().split(",");

        String account="";
        if(!fields[19].equals("NULL")){
            account=" CD ACC NO";
            c.output(KV.of(account, 1));
        }
        if(!fields[20].equals("NULL")){
            account=" FD ACC NO";
            c.output(KV.of(account,1));
        }
        if(!fields[20].equals("NULL")){
            account=" OD ACC NO";
            c.output(KV.of(account,1));
        }
    }
}

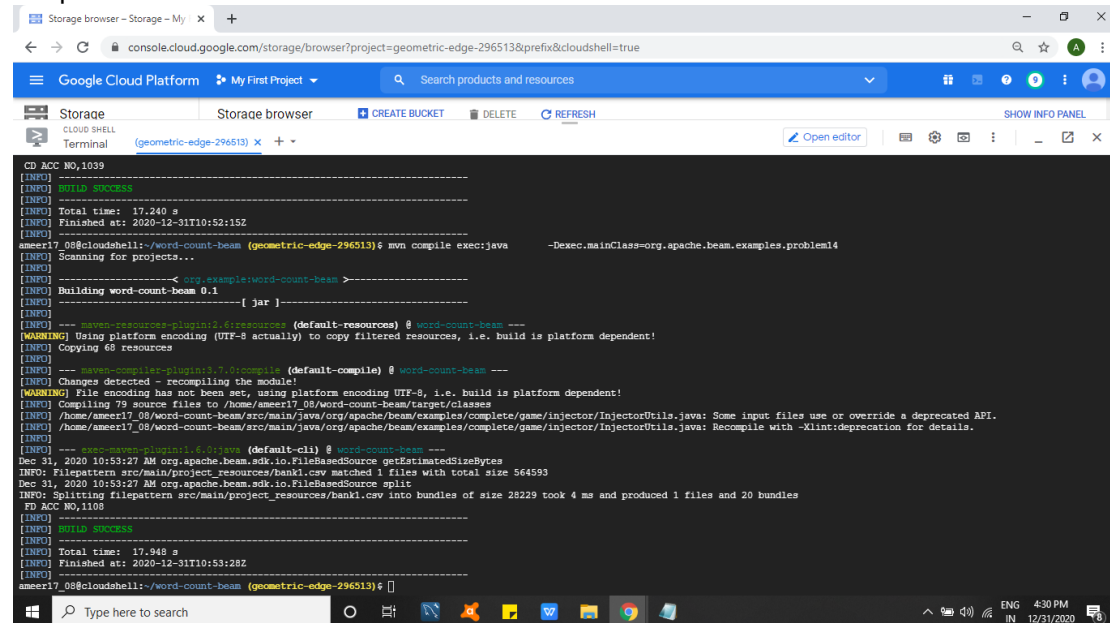
private static class ConvertToStringFn extends DoFn<KV<String,Integer>, String> {

    @ProcessElement
    public void processElement(ProcessContext c) {

        if(count==0){
            c.output(c.element().getKey() + "," + c.element().getValue());
            count++;
        }
    }
}
}

```

Output:



```
CD ACC NO,1039
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 17.240 s
[INFO] Finished at: 2020-12-31T10:52:15Z
[INFO] -----
ameer17_08@cloudshell:~/word-count-beam (geometric-edge-296513) $ mvn compile exec:java -Dexec.mainClass=org.apache.beam.examples.problem14
[INFO] Scanning for projects...
[INFO] -----< org.apache:word-count-beam >-----
[INFO] Building word-count-beam 0.1
[INFO] [ jar ]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ word-count-beam ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 68 resources
[INFO] --- maven-compiler-plugin:3.7.0:compile (default-compile) @ word-count-beam ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 76 source files to /home/ameer17_08/word-count-beam/target/classes
[INFO] /home/ameer17_08/word-count-beam/src/main/java/org/apache/beam/examples/complete/game/injector/InjectorUtils.java: Some input files use or override a deprecated API.
[INFO] /home/ameer17_08/word-count-beam/src/main/java/org/apache/beam/examples/complete/game/injector/InjectorUtils.java: Recompile with -Xlint:deprecation for details.
[INFO] --- exec-maven-plugin:1.6.0:java (default-cli) @ word-count-beam ---
Dec 31, 2020 10:53:27 AM org.apache.beam.sdk.io.FileBasedSource getEstimatedSizeBytes
INFO: Filepattern src/main/project_resources/bank1.csv matched 1 files with total size 564593
Dec 31, 2020 10:53:27 AM org.apache.beam.sdk.io.FileBasedSource split
INFO: Splitting filepattern src/main/project_resources/bank1.csv into bundles of size 28229 took 4 ms and produced 1 files and 20 bundles
FD ACC NO,1108
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 17.948 s
[INFO] Finished at: 2020-12-31T10:53:28Z
[INFO] -----
ameer17_08@cloudshell:~/word-count-beam (geometric-edge-296513) $
```

2. Analyse which situation is more favourable to get the loan sanctioned based on marital status and housing

Code:

```
package org.apache.beam.examples;

import org.apache.beam.sdk.Pipeline;
import org.apache.beam.sdk.options.PipelineOptions;
import org.apache.beam.sdk.options.PipelineOptionsFactory;
import org.apache.beam.sdk.io.TextIO;
import org.apache.beam.sdk.transforms.*;
import org.apache.beam.sdk.values.*;

import java.util.Calendar;

public class BankLoan {

    public static void main(String[] args) {
        PipelineOptions options = PipelineOptionsFactory.fromArgs(args).withValidation().create();
        Pipeline pipeline = Pipeline.create(options);

        PCollection<String> value = pipeline.apply("ReadAds",
            TextIO.read().from("src/main/resources/bank1.csv"))
            // .apply("FilterHeader", ParDo.of(new FilterHeaderFn(CSV_HEADER)))
            .apply("MakeAgeKVFn", ParDo.of(new MakeAgeKVFn()))
            .apply("Sum", Sum.integersPerKey())
```



```

        .apply(ParDo.of(new ConvertToStringFn()));

value.apply("PrintToConsole", ParDo.of(new DoFn<String, Void>() {
    @ProcessElement
    public void processElement(ProcessContext c) {
        System.out.println(c.element());
    }
}));

// value.apply("Writeresult to file", TextIO.write().to("gs://albinbucket/amita/output"));

pipeline.run().waitUntilFinish();
}

private static class FilterHeaderFn extends DoFn<String, String> {

    private final String header;

    public FilterHeaderFn(String header) {
        this.header = header;
    }

    @ProcessElement
    public void processElement(ProcessContext c) {
        String row = c.element();

        if (!row.isEmpty() && !row.equals(this.header)) {
            c.output(row);
        }
    }
}

private static class MakeAgeKVFn extends DoFn<String, KV<String, Integer>> {

    @ProcessElement
    public void processElement(ProcessContext c) {
        String[] fields = c.element().split(",");

        String marriage = fields[2];
        String housing = fields[14];
        String val1="Married,Housing=yes,Loan=yes";
        String val2 = "Married,Housing=no,Loan=yes";
        String val3 ="Married,Housing=yes,Loan=no";
        String val4 ="Married,Housing=no,Loan=no";
        String val5 ="Single,Housing=no,Loan=no";
        String val6="Single,Housing=no,Loan=Yes";
        String val7 = "Single,Housing=no,Loan=Yes";
        String val8 = "Single,Housing=yes,Loan=Yes";
        if(marriage.equals("married") && housing.equals("yes"))
        {
            if(fields[15].equals("yes"))
            {
                c.output(KV.of(val1, 1));
            }
        }
        else if(marriage.equals("married") && housing.equals("no"))
        {

```



```

        if(fields[15].equals("yes"))
        {
            c.output(KV.of(val2, 1));
        }
    }
    else if(marriage.equals("married") && housing.equals("yes"))
    {
        if(fields[15].equals("no"))
        {
            c.output(KV.of(val3, 1));
        }
    }
    else if(marriage.equals("married") && housing.equals("no"))
    {
        if(fields[15].equals("no"))
        {
            c.output(KV.of(val4, 1));
        }
    }
    else if(marriage.equals("single") && housing.equals("no"))
    {
        if(fields[15].equals("no"))
        {
            c.output(KV.of(val5, 1));
        }
    }
    else if(marriage.equals("single") && housing.equals("yes"))
    {
        if(fields[15].equals("no"))
        {
            c.output(KV.of(val6, 1));
        }
    }
    else if(marriage.equals("single") && housing.equals("no"))
    {
        if(fields[15].equals("yes"))
        {
            c.output(KV.of(val7, 1));
        }
    }
    else if(marriage.equals("single") && housing.equals("yes"))
    {
        if(fields[15].equals("yes"))
        {
            c.output(KV.of(val8, 1));
        }
    }
    }
}

```

```

private static class ConvertToStringFn extends DoFn<KV<String, Integer>, String> {

    @ProcessElement
    public void processElement(ProcessContext c) {
        c.output(c.element().getKey() + "," + c.element().getValue());
    }
}

```

```
}
}
```

Output:

```

[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 16.036 s
[INFO] Finished at: 2020-12-31T11:43:14Z
[INFO] -----
[WARNING] The requested profile "directflow-runner" could not be activated because it does not exist.
mitaison29@cloudshell:~/word-count-beam (geometric-edge-296513)$ mv -P directflow-runner compile exec:java -Deam.mainClass=org.apache.beam.examples.Bankfion
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building word-count-beam 0.1
[INFO] -----
[INFO] --- beam:resources:plugin2:fileResources (default resources) @ word-count-beam ---
[INFO] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 24 resources
[INFO] --- beam:compiler:plugin2:fileResources (default compiler) @ word-count-beam ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!
[INFO] Compiling 89 source files to /home/mitaison29/word-count-beam/target/classes
[INFO] /home/mitaison29/word-count-beam/src/main/java/org/apache/beam/examples/complete/game/injector/InjectorUtil.java: /home/mitaison29/word-count-beam/src/main/java/org/apache/beam/examples/complete/game/inject
or/InjectorUtil.java uses or overrides a deprecated API.
[INFO] /home/mitaison29/word-count-beam/src/main/java/org/apache/beam/examples/complete/game/injector/InjectorUtil.java: Recompile with -Xlint:deprecation for details.
[INFO] --- beam:compiler:plugin2:fileResources (default-compile) @ word-count-beam ---
Dec 31, 2020 11:43:43 AM org.apache.beam.sdk.io.FileBasedSource getPlatformedFileType
INFO: Filepattern src/main/resources/bank1.csv matched 1 files with total size 564593
Dec 31, 2020 11:43:43 AM org.apache.beam.sdk.io.FileBasedSource split
INFO: Splitting filepattern src/main/resources/bank1.csv into bundles of size 28229 took 3 ms and produced 1 files and 20 bundles
Married, Housing: yes, loan: yes, 16
Single, Housing: no, loan: no, 28
Married, Housing: yes, loan: yes, 158
Single, Housing: no, loan: no, 425
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 16.584 s
[INFO] Finished at: 2020-12-31T11:43:44Z
[INFO] -----
[WARNING] The requested profile "directflow-runner" could not be activated because it does not exist.
mitaison29@cloudshell:~/word-count-beam (geometric-edge-296513)$

```

3. Analyse based on age and education the bank balance of the employees

Code:

```
package org.apache.beam.examples;
```

```
import org.apache.beam.sdk.Pipeline;
import org.apache.beam.sdk.options.PipelineOptions;
import org.apache.beam.sdk.options.PipelineOptionsFactory;
import org.apache.beam.sdk.io.TextIO;
import org.apache.beam.sdk.transforms.*;
import org.apache.beam.sdk.values.*;
```

```
import java.util.Calendar;
```

```
public class account {
```

```
    private static final String CSV_HEADER =
        "car,price,body,mileage,engV,engType,registration,year,model,drive";
```

```
    public static void main(String[] args) {
        PipelineOptions options = PipelineOptionsFactory.fromArgs(args).withValidation().create();
        Pipeline pipeline = Pipeline.create(options);
```

```
        pipeline.apply("ReadAds",
            TextIO.read().from("/home/harshita1262/word-count-beam/src/main/resources/bankfinal1.csv"))
```

```

        .apply("FilterHeader", ParDo.of(new FilterHeaderFn(CSV_HEADER)))
        .apply("MakeAgeKVFn", ParDo.of(new MakeAgeKVFn()))
        .apply(Sum.integersPerKey())
        .apply(ParDo.of(new ConvertToStringFn()))
        .apply("PrintToConsole", ParDo.of(new DoFn<String, Void>() {
            @ProcessElement
            public void processElement(ProcessContext c) {
                System.out.println(c.element());
            }
        }));

//value.apply("Writeresult to file", TextIO.write().to("gs://albinbucket/amita/output"));

pipeline.run().waitUntilFinish();
}

private static class FilterHeaderFn extends DoFn<String, String> {

    private final String header;

    public FilterHeaderFn(String header) {
        this.header = header;
    }

    @ProcessElement
    public void processElement(ProcessContext c) {
        String row = c.element();

        if (!row.isEmpty() && !row.equals(this.header)) {
            c.output(row);
        }
    }
}

private static class MakeAgeKVFn extends DoFn<String, KV<String, Integer>> {

    @ProcessElement
    public void processElement(ProcessContext c) {
        String[] fields = c.element().split(",");
        if(!(fields[23].equals("BALANCE"))){
            int sal=Integer.parseInt(fields[23]);

            c.output(KV.of(fields[1]+" ", "+fields[11],sal));
        }
    }
}
}

```

```

private static class ConvertToStringFn extends DoFn<KV<String, Integer>, String> {

    @ProcessElement
    public void processElement(ProcessContext c) {
        c.output(c.element().getKey() + "," + c.element().getValue());
    }
}
}

```

The screenshot shows a Google Cloud Platform console window with a terminal output. The terminal displays the following test cases:

```

32 , MSA, 21281
24 , MCA, 5053
34 , mba, 4132
27 , mba, 153320
31 , mba, 248319
32 , mba, 205244
30 , MSA, 242576
32 , bac, 3384
25 , MSC, 5044
32 , MSC, 53240
31 , BE/BTech, 287363
31 , MTECH, 359234
27 , MCA, 16781
29 , MTECH, 919618
26 , MSA, 221274
38 , MTECH, 240612
25 , mba, 729
24 , MTECH, 58807
27 , MTECH, 788390
30 , MCA, 11000
28 , BE/BTech, 752627
26 , MTECH, 794732
29 , MSA, 200000
28 , mba, 23783
30 , mba, 598
29 , bac, 20509
37 , MSA, 21305
28 , MCA, 8000
24 , MSC, 20000
22 , MTECH, 163204
31 , bac, 34511
27 , MSC, 71989

```

The build process completed successfully. The terminal output shows:

```

[INFO] BUILD SUCCESS
[INFO] Total time: 9.074 s
[INFO] Finished at: 2020-12-31T10:15:49Z

```