

# Loan Analysis

Madhumathi Prakash

04.26.2019

## Introduction

Dataset link: [https://  
www.lendingclub.com/info/  
download-data.action](https://www.lendingclub.com/info/download-data.action)

Lending  
Club

Dataset

Overview

Putmerge

## About



- Help millions of people take control of their debt, grow their small businesses, and invest for the future
- Invest or Borrow
- Personal, business, education loans

## Dataset

- Loan data set
  - complete loan data for all loans issued through the time period stated, including the current loan status (Current, Late, Fully Paid, etc.) and latest payment information
- 2007-2018
- Members, homeowners, loan, interest, employment, verification, grade

Data  
fields

# Data Fields

Data Field	Description
addrState	The state provided by the borrower in the loan application
annualInc	The self-reported annual income provided by the borrower during registration.
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
delinq2Yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
delinqAmnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
empLength	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
expDefaultRate	The expected default rate of the loan.
ficoRangeHigh	The upper boundary range the borrower's FICO at loan origination belongs to.
ficoRangeLow	The lower boundary range the borrower's FICO at loan origination belongs to.
grade	LC assigned loan grade
homeOwnership	The home ownership status provided by the borrower during registration. Our values are: RENT, OWN, MORTGAGE, OTHER.
id	A unique LC assigned ID for the loan listing.
installment	The monthly payment owed by the borrower if the loan originates.
intRate	Interest Rate on the loan
loanAmnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
purpose	A category provided by the borrower for the loan request.
subGrade	LC assigned loan subgrade
term	The number of payments on the loan. Values are in months and can be either 36 or 60.
totalAcc	The total number of credit lines currently in the borrower's credit file
verified_status_joint	Indicates if the co-borrowers' joint income was verified by LC, not verified, or if the income source was verified
zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.

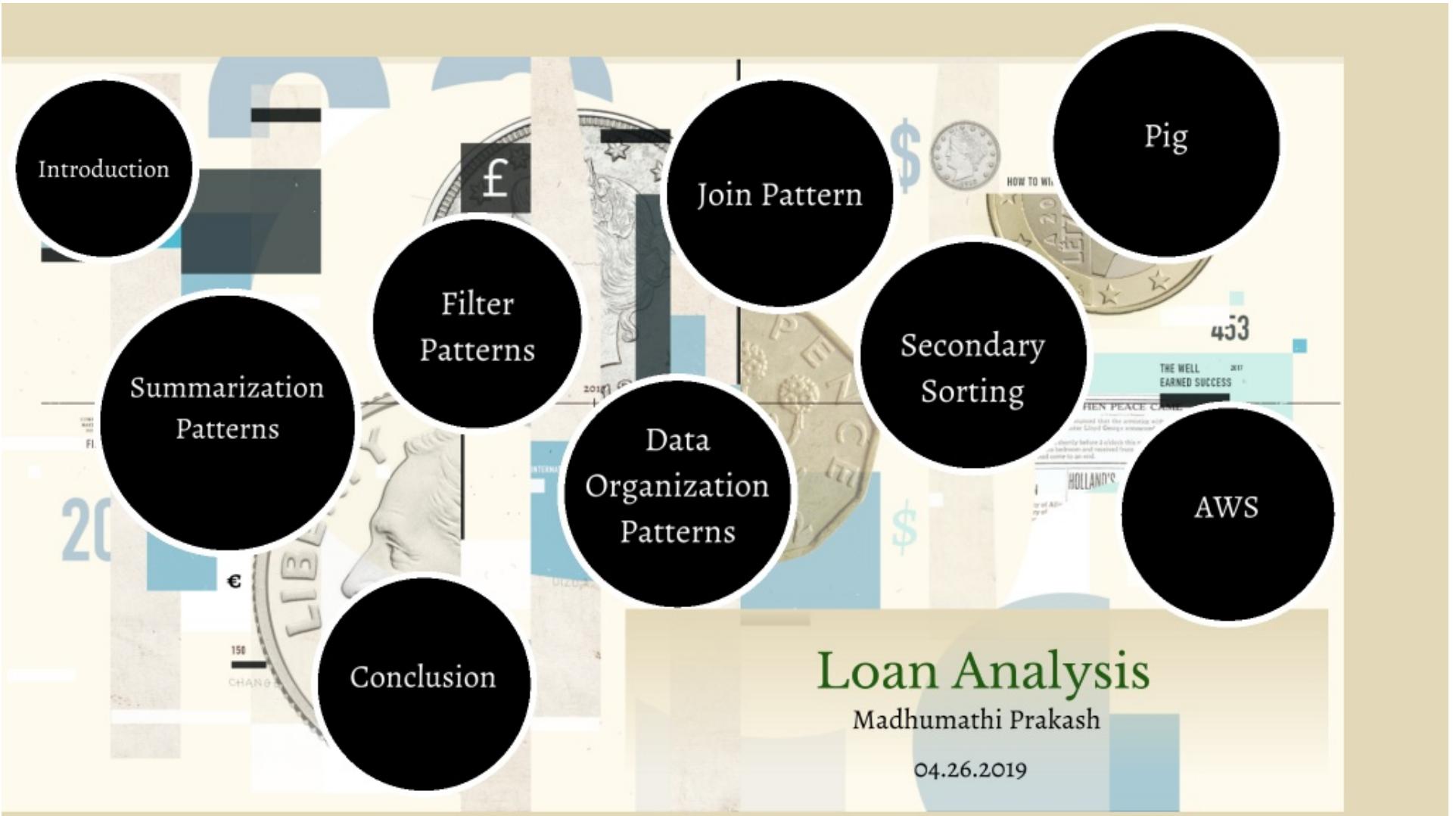
## What will I cover

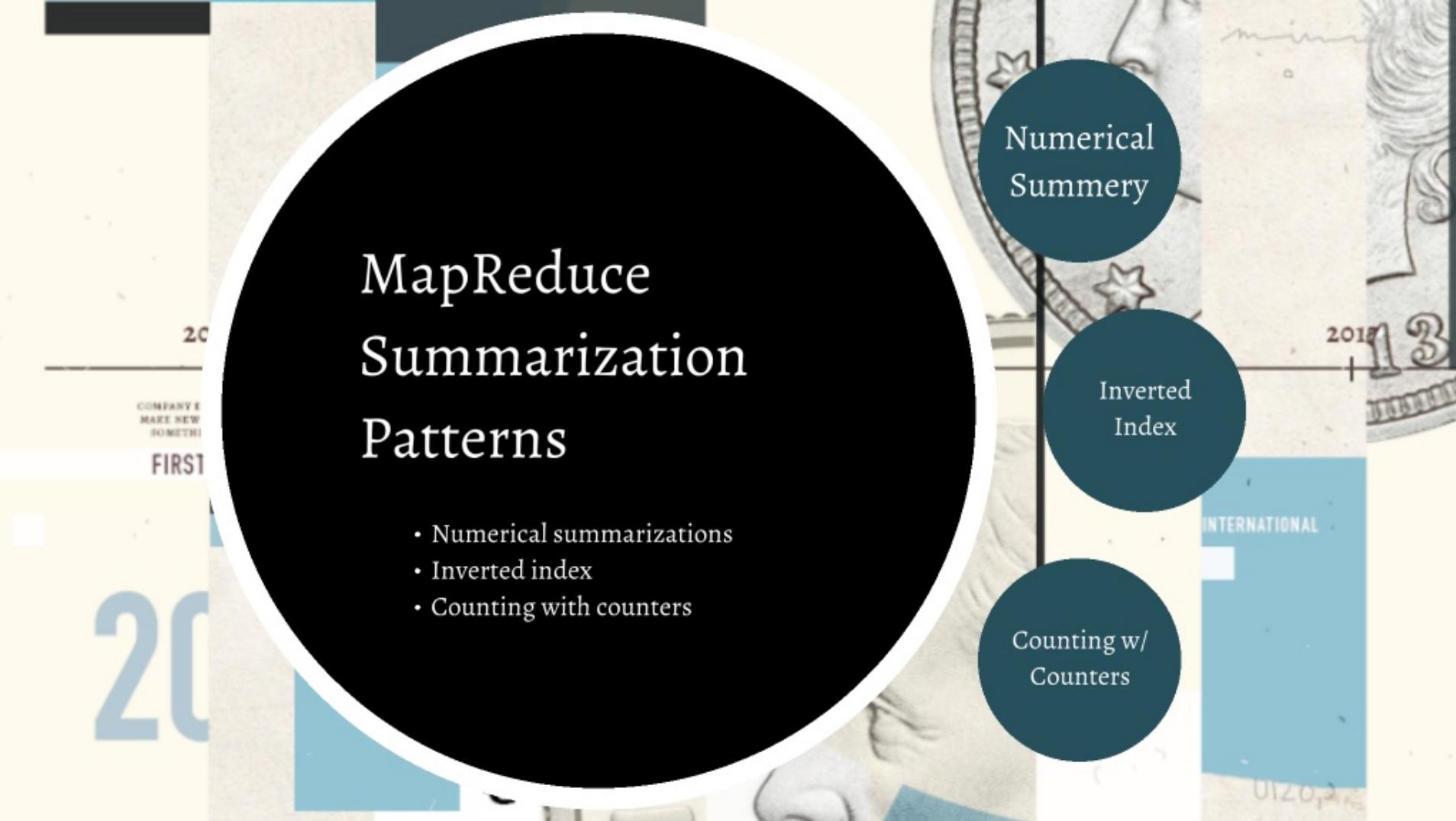
- Different data patterns
- Analysis of the different fields to determine the range or loans and interests given
- Look into tableau dashboard

# Putmerge

Combined all CSVs dealing with loan data - utilize just the driver

Take data csv as input and put in HDFS as 1 files that is output





# MapReduce Summarization Patterns

- Numerical summarizations
- Inverted index
- Counting with counters

Numerical  
Summery

Inverted  
Index

Counting w/  
Counters

## Numerical Summarization

MapReduce

1. Max and min
2. counting
3. Average
4. Median
5. Memory conscious Median

MaxMin

Counting

Average

Median

Memory  
Conscious  
Median

# Maximum and Minimum

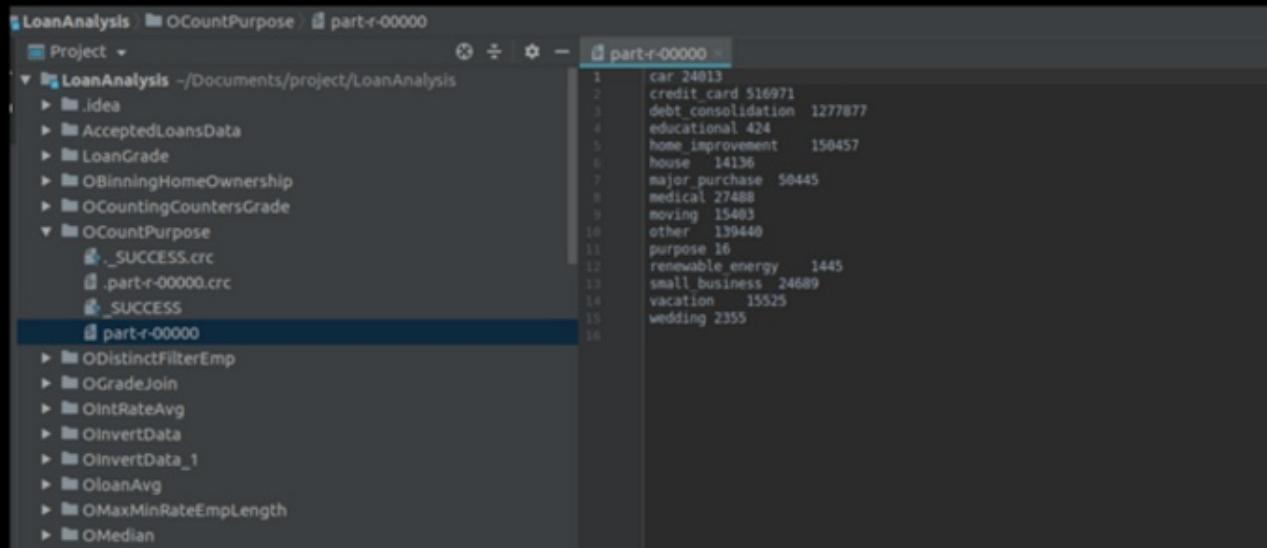
Found Minimum and Maximum interest Rate based on the Length of Employment in years

```
1 year 5.31 30.99
2 10+ years 5.31 30.99
3 2 years 5.31 30.99
4 3 years 5.31 30.99
5 4 years 5.31 30.99
6 5 years 5.31 30.99
7 6 years 5.31 30.99
8 7 years 5.31 30.99
9 8 years 5.31 30.99
10 9 years 5.31 30.99
11 < 1 year 5.31 30.99
12 n/a 5.31 30.99
```

Utilized NLine  
Mapper: Key as  
employment and value  
passing interest rate

Reducer: Key as  
employment and value  
concat the calculated  
max and min

# Determined number of data based on the Purpose



The screenshot shows a Java IDE interface with a project named 'LoanAnalysis'. The 'part-r-00000' file is open, displaying a list of loan purposes and their corresponding counts. The data is as follows:

Purpose	Count
car	24013
credit_card	516971
debt_consolidation	1277877
educational	424
home_improvement	150457
house	14136
major_purchase	50445
medical	27488
moving	15483
other	139440
purpose	16
renewable_energy	1445
small_business	24689
vacation	15525
wedding	2355

Key: purpose Value: sum of count

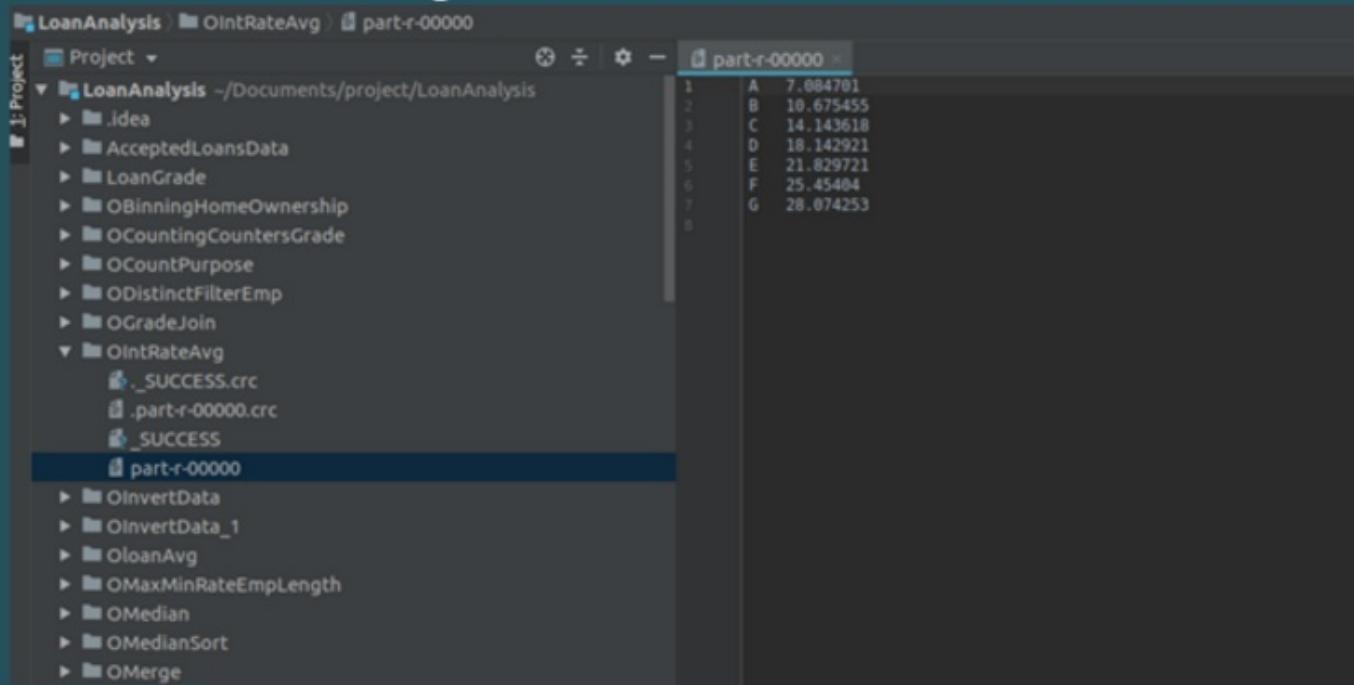


# Average

interest  
Rate

Loan

# Calculated Average for Interest rate based on grade



1	A	7.084701
2	B	10.675455
3	C	14.143618
4	D	18.142921
5	E	21.829721
6	F	25.45404
7	G	28.074253
8		

In map we create a pair of the sum and count of the average

Cleanup Context.write the key being grade and value being pair of sum and count

Allows the use of combiner that reduces the work of reducer

# Calculated Average of loan based on the Purpose

The screenshot shows a Java IDE interface with the following details:

- File menu:** File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help.
- Project tree:** LoanAnalysis (~/Documents/project/LoanAnalysis) contains .idea, AcceptedLoansData, LoanGrade, OBinningHomeOwnership, OCountingCountersGrade, OCountPurpose, ODistinctFilterEmp, OGradeJoin, OIntRateAvg, OInvertData, OInvertData\_1, OloanAvg, and part-r-00000.
- Code editor:** The part-r-00000 file displays the following data:

Loan Purpose	Count
car	9393
credit_card	-1296
debt_consolidation	-838
educational	6614
home_improvement	-13879
house	15703
major_purchase	12682
medical	9474
moving	8390
other	10481
renewable_energy	10757
small_business	16442
vacation	6357
wedding	10475

# Calculated Median of Fico score based on the Grade.

	A	729
1	B	709
2	C	699
3	D	699
4	E	694
5	F	694
6	G	694
7		

- Reducer adding everything to arraylist and finding the middle fico score for each grade
- The loan grade is the result of a formula that takes into account not only credit score, but also a combination of several indicators of credit risk from the credit report and loan application

# Memory Conscious Median of Loan Amount based on the Verification status

The screenshot shows a Java IDE interface with a project named "LoanAnalysis". The left sidebar displays the project structure, including various classes and methods such as AcceptedLoansData, LoanGrade, OCountingCountersGrade, OCountPurpose, ODistinctFilterEmp, OGradeJoin, OIntRateAvg, OInvertData, OInvertData\_1, OloanAvg, OMaxMinRateEmpLength, OMedian, and OMedianSort. The right side shows two code editor tabs: "OloanAvg/part-r-00000" and "OMedian/part-r-00000". The "OloanAvg" tab contains the following code:

```
36 months median:9987.5 Std_Div:426.5527
60 months median:19987.5 Std_Div:519.3332
```

The "OMedian" tab contains the following code:

```
17.
```

Utilizing Combiner and displaying both the median for 36 month and 60 month verification status

## Inverted Index Pattern

Used chaining for filtered Data and found the list loan ids based on the Fico score as key

# Counting the number of data based on grades using counters

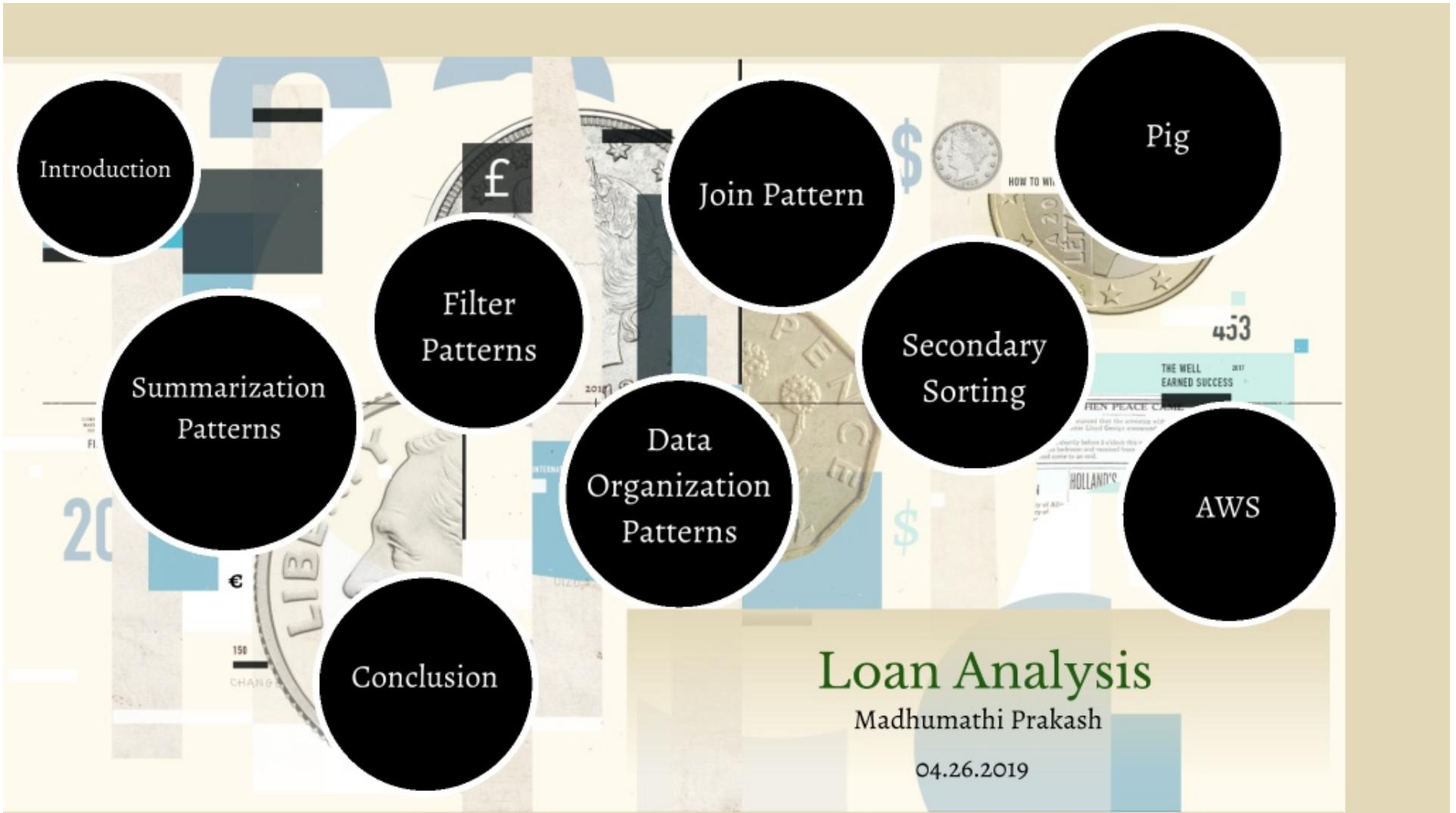
The screenshot shows an IDE interface with a code editor and a run output window.

**Code Editor:** The code is part of a class named `Driver` in the `main` package. It includes imports for `org.apache.hadoop.mapreduce.Job`, `org.apache.hadoop.mapreduce.lib.input.FileInputFormat`, `org.apache.hadoop.mapreduce.lib.output.FileOutputFormat`, and `org.apache.hadoop.mapreduce.Counters`. The code sets up a job, defines a group for counters, and prints out counter values.

```
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35 int code = job.waitForCompletion( verbose: true)? 1 : 0;
36 if (code==1) {
37
38     for(Counter counter: job.getCounters().getGroup( groupName: "MonthCounter")) {
39         //System.out.println("inside this");
40         System.out.println("the counter "+counter.getDisplayName()+" is "+counter.getValue());
41         // System.out.println("the counter values is "+counter.getValue());
42     }
43
44 }
45 }
```

**Run Output:** The output window shows the results of the run, including file input and output counters, and a list of counters with their values.

```
Run: CountingWithCountersGrade x
File Input Format Counters
  Bytes Read=1093535734
File Output Format Counters
  Bytes Written=8
the counter A is 439827
the counter B is 663557
the counter C is 650853
the counter D is 324424
the counter E is 135639
the counter F is 41800
the counter G is 12168
the counter unknown is 16
Process finished with exit code 0
```



# Filtering Patterns

1. Bloom filtering
2. Top K
3. Distinct

Bloom  
Filtering

Top K

Distinct

Used Bloom Filter to filter the data and select only the data related to MA and NY

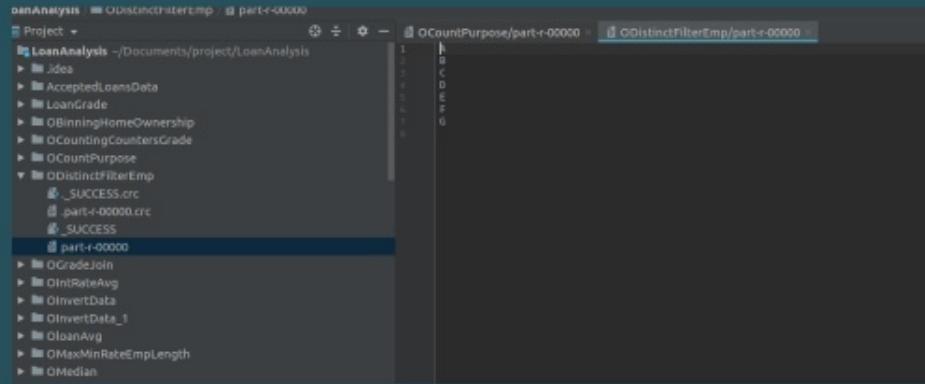
- Put the 2 items in the `this.origin = BloomFilter` in the setup
  - In map checking if data coming in is matching the BloomFilter

# Identified the Top 10 members based on the Loan Amount

```
LoanAnalysis [-/Documents/project/LoanAnalysis] - .../OTopK/part-r-00000 [LoanAnalysis] - IntelliJ IDEA
File Edit View Navigate Code Analyze Befactor Build Run Tools VCS Window Help
LoanAnalysis OTopK part-r-00000
Project OShufflesData/part-r-00000 OTopK/part-r-00000
1 141942827,,38000,38000,38000,36 months,11.55%,990,0.84,1 year,RENT,90000,Not Verified,Oct-18,Current,n,other,880xx,C0,1.6,0,Jan-15,705,709,0,27,,9,0,2354,7.40%,12,w,24580,24500,4
2 138812769,,32000,32000,32000,36 months,11.55%,1056,8,84,< 1 year,RENT,150000,Source Verified,Aug-18,Current,n,small business,773xx,TX,4,05,0,Sep-97,678,674,1,,75,16,1,13784,36,40
3 139832556,,33000,33000,33000,36 months,11.55%,1089,8,84,< 1 year,MORTGAGE,21000,Not Verified,Sep-18,Current,n,credit_card,159xx,PA,77,77,0,Aug-96,690,694,1,46,,-6,0,1485,30,99%,35
4 75728706,,35000,35000,35000,36 months,7.89%,1095,A,45,n,o,MORTGAGE,75000,Verified,Mar-16,Current,n,credit_card,117xx,NY,8,43,0,Mar-03,690,694,0,27,,6,0,19259,69,30%,11,w,1087,75
5 137881711,,34000,34000,34000,36 months,11.55%,1122,8,84,10+ years,MORTGAGE,90000,Not Verified,Aug-18,Current,n,home improvement,189xx,PA,36,71,0,Apr-01,738,734,0,71,,11,0,201700,
6 96376153,,35000,35000,35000,36 months,11.49%,1154,B,85,2 years,RENT,40000,Source Verified,Jan-17,Current,n,other,398xx,GA,15,92,1,Aug-07,700,704,0,37,,21,0,2792,16,209,31,w,12638
7 139884969,,35000,35000,34750,36 months,11.55%,1155,8,84,3 years,MORTGAGE,63000,Source Verified,Oct-18,Current,n,credit_card,327xx,FL,22,29,0,Mar-05,739,734,0,66,,11,0,13986,35%,2
8 137880846,,36000,36000,36000,36 months,11.55%,1186,8,84,10+ years,MORTGAGE,110000,Source Verified,Aug-18,Current,n,credit_card,453xx,OH,16,45,0,May-01,718,714,0,51,,11,0,14128,37
9 74523492,,40000,40000,40000,36 months,9.75%,1286,8,85,10+ years,MORTGAGE,183000,Not Verified,Mar-16,In Grace Period,n,credit_card,221xx,VA,13,61,2,Feb-02,690,694,0,7,,11,0,13722
10 141546266,,48000,48000,48000,36 months,11.55%,1320,8,84,< 1 year,MORTGAGE,98000,Verified,Oct-18,Current,n,credit_card,305xx,NY,26,19,0,Jul-99,695,690,0,39,,18,0,14445,52,30%,33,4
```

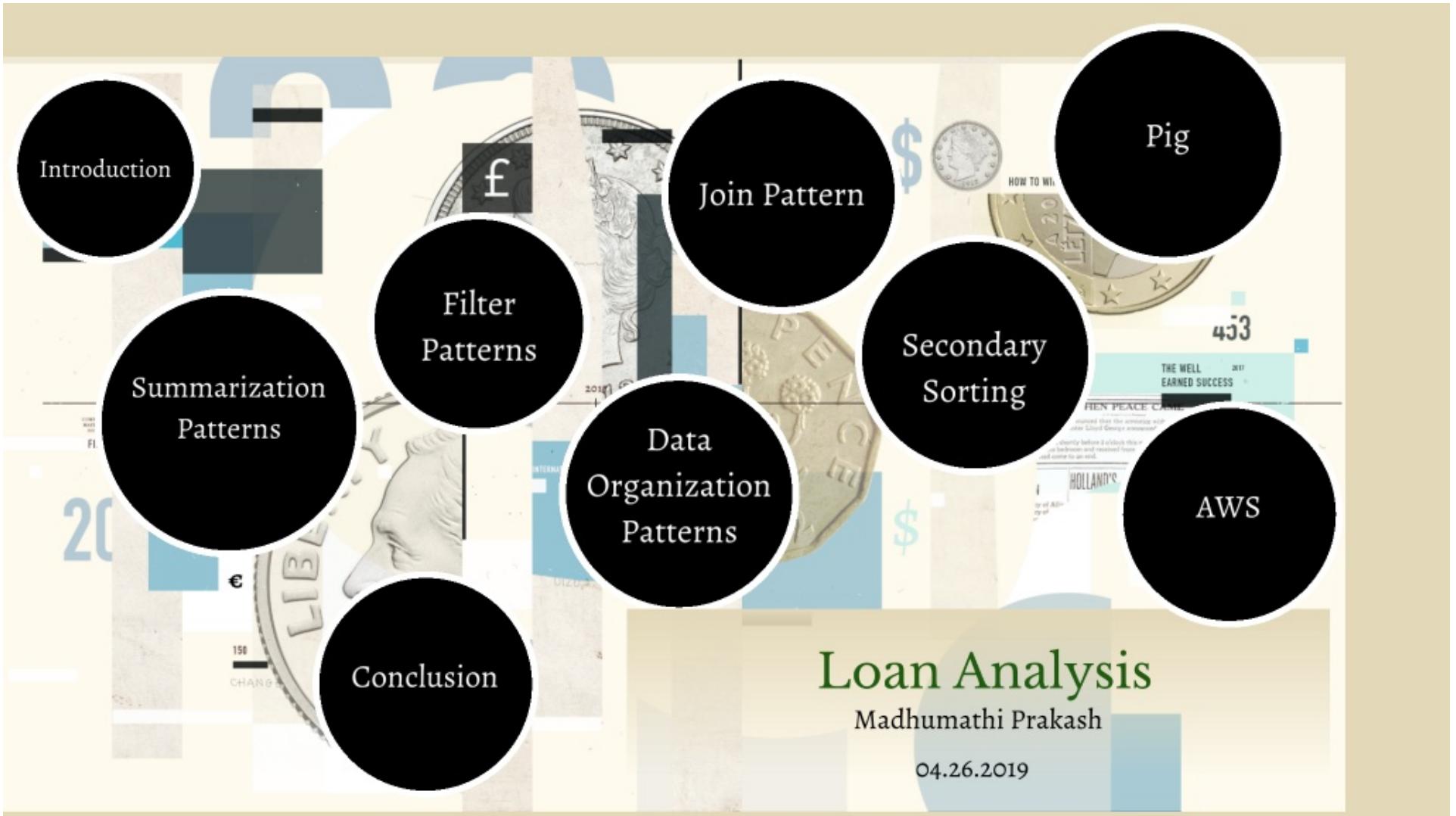
To get the top 10 checking that tMap.size() > K in the map and doing the context.write in the cleanup of the mapper

# Distinct grades



The key will have the grade passed and the value will be NullWritable to keep it unique

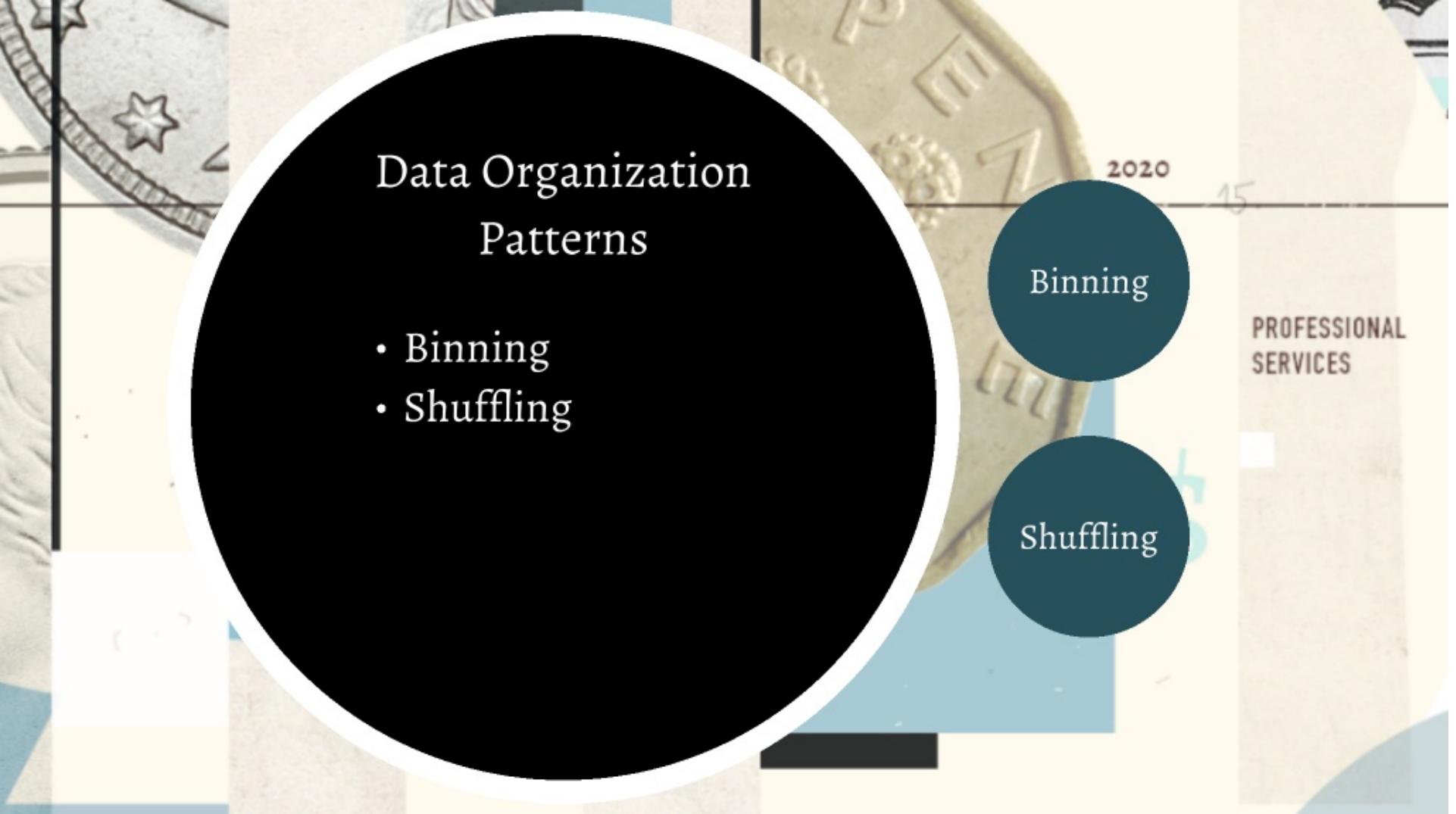
```
@Override  
protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, NullWritable>.Context context)  
    throws IOException, InterruptedException {  
    String line = value.toString();  
  
    String[] tokens = line.split(regex);  
    if(tokens.length > 15 && !line.contains("grade")) {  
        | context.write(new Text(tokens[8]), NullWritable.get()); //10  
    }  
}
```



# Loan Analysis

Madhumathi Prakash

04.26.2019



## Data Organization Patterns

- Binning
- Shuffling

Binning

Shuffling

# Binning to separate the file based on Home Ownership

```

1 | 6419273, 21500, 21500, 36 months, 11.39%, 707, 86, B, B1, 10+ years, MORTGAGE, 125000, Not Verified, Dec-16, Current, n, debt consolidation, 750xx, OK, 20, 18, B, Dec-96, 710, 714, 0, , 1
2 | 96191765, 9300, 9300, 36 months, 8, 24%, 292, 46, B, B1, 9 years, MORTGAGE, 62000, Source Verified, Dec-16, Current, n, debt consolidation, 481xx, MI, 16, 12, B, Jun-03, 666, 664, 1, 71, 99, 3
3 | 96419283, 14400, 14400, 14400, 60 months, 7, 99%, 291, 92, A, A5, 9 years, MORTGAGE, 135000, Not Verified, Dec-16, Current, n, home improvement, 870xx, NJ, 6, 6, 1, Oct-99, 695, 699, 0, 11, 15, 0, 8
4 | 95180015, 13480, 13480, 13480, 36 months, 13.49%, 454, 67, C, C2, 5 years, MORTGAGE, 71000, Not Verified, Dec-16, Fully Paid, n, home improvement, 951xx, CA, 32, 94, 8, Dec-78, 690, 694, 0, 79, 76
5 | 96379251, 10000, 10000, 10000, 36 months, 11.39%, 129, 24, II, H3, < 1 year, MORTGAGE, 703xx, LA, 29, 93, 0, Sep-03, 720, 724, 0, , 13, 6
6 | 96419291, 13800, 13800, 13800, 36 months, 5, 32%, 391, 5, A, A1, 10+ years, MORTGAGE, 161000, Not Verified, Dec-16, Charged Off, n, credit card, 461xx, IN, 18, 91, 0, Jul-97, 728, 724, 0, , 79, 8, 3
7 | 96209228, 10000, 10000, 10000, 36 months, 7, 99%, 563, 98, A, A5, 2 years, MORTGAGE, 52000, Source Verified, Dec-16, Charged Off, n, credit card, 461xx, FL, 30, 76, 0, Oct-00, 775, 779, 0, , 13, 0
8 | 94366756, 25900, 25900, 25900, 60 months, 24.49%, 767, 66, E, E4, 10+ years, MORTGAGE, 54000, Verified, Dec-16, Charged Off, n, credit card, 433xx, OH, 20, 9, Jan-85, 685, 689, 1, 56, 0, 9, 0, 27346
9 | 96409250, 40000, 40000, 40000, 36 months, 7, 99%, 563, 98, A, A5, 2 years, MORTGAGE, 60000, Not Verified, Dec-16, Fully Paid, n, credit card, 666xx, KS, 28, 45, 3, Oct-96, 680, 684, 0, 7, 7, 6, 56
10 | 96349353, 39000, 39000, 39000, 36 months, 15.99%, 1370, 94, C, C5, 10+ years, MORTGAGE, 78000, Source Verified, Dec-16, Current, n, debt consolidation, 453xx, DE, 13, 14, 0, May-00, 715, 719, 0
11 | 95845940, 20000, 20000, 20000, 36 months, 16.99%, 250, 66, A, A5, 10+ years, MORTGAGE, 63000, Not Verified, Dec-16, Current, n, debt consolidation, 511xx, MA, 29, 03, 1, Mar-90, 660, 664, 2, 23, 7
12 | 93999587, 19200, 19200, 19200, 36 months, 13.99%, 656, 12, C, C3, 10+ years, MORTGAGE, 75000, Not Verified, Dec-16, Current, n, home improvement, 531xx, MI, 16, 98, 0, Dec-03, 748, 744, 0, , 8, 8
13 | 95547393, 25675, 25675, 25675, 60 months, 14.99%, 598, 41, C, C4, 10+ years, MORTGAGE, 64000, Not Verified, Dec-16, Fully Paid, n, other, 272xx, NC, 10, 70, 0, Oct-03, 740, 744, 1, , 23, 0, 8922, 9
14 | 96164691, 18000, 18000, 18000, 36 months, 8, 24%, 357, 65, B, B1, 8 years, MORTGAGE, 95000, Not Verified, Dec-16, Fully Paid, n, credit card, 881xx, CO, 21, 22, B, May-98, 734, 0, 56, 6, 0, 22
15 | 96259182, 8400, 8400, 8400, 36 months, 11.44%, 276, 76, B, B4, 10+ years, MORTGAGE, 70000, Not Verified, Dec-16, Fully Paid, n, credit card, 486xx, MI, 20, 69, 0, Jan-01, 695, 699, 0, , 118, 19, 1, 7
16 | 95983888, 5000, 5000, 5000, 36 months, 7.49%, 151, 51, A, A4, 10+ years, MORTGAGE, 20000, Verified, Dec-16, Current, n, debt consolidation, 658xx, MO, 6, 66, 0, Jan-99, 733, 739, 0, 31, 5, 0, 4729
17 | 95831658, 20000, 20000, 20000, 36 months, 7.99%, 626, 64, A, A5, 8 years, MORTGAGE, 120000, Not Verified, Dec-16, Current, n, debt consolidation, 204xx, MO, 3, 38, 0, Jan-07, 710, 714, 0, , 12, 0
18 | 96389124, 15000, 15000, 15000, 36 months, 5, 32%, 451, 73, A, A1, 10+ years, MORTGAGE, 98000, Source Verified, Dec-16, Fully Paid, n, credit card, 198xx, PA, 4, 76, 1, Feb-03, 705, 709, 0, 2, 1, 2
19 | 96674335, 11000, 11000, 11000, 36 months, 5, 32%, 331, 27, A, A1, 6 years, MORTGAGE, 52000, Source Verified, Dec-16, Current, n, credit card, 327xx, FL, B, A3, 0, Sep-03, 785, 789, 0, , 0, 0, 17785
20 | 95347510, 20000, 20000, 20000, 36 months, 11.49%, 659, 43, B, B5, 10+ years, MORTGAGE, 78000, Not Verified, Dec-16, Fully Paid, n, credit card, 378xx, TN, 15, 81, 0, Aug-00, 715, 719, 2, 64, 105, 3
21 | 96379167, 12000, 12000, 12000, 60 months, 13.99%, 279, 16, C, C2, 3 years, MORTGAGE, 285000, Source Verified, Dec-16, Fully Paid, n, debt consolidation, 943xx, CA, 11, 91, 0, Jun-90, 666, 664, 0
22 | 95638862, 7175, 7175, 7175, 36 months, 13.49%, 243, 46, C, C2, 10+ years, MORTGAGE, 65000, Source Verified, Dec-16, Fully Paid, n, debt consolidation, 331xx, FL, 3, 4, 8, Aug-02, 665, 669, 0, 59
23 | 96173850, 4050, 8050, 8050, 36 months, 17.99%, 290, 99, B, B2, 10+ years, MORTGAGE, 120000, Not Verified, Dec-16, Current, n, debt consolidation, 852xx, AZ, 14, 79, 2, Aug-99, 705, 709, 2, 17, 13
24 | 96339220, 4080, 4080, 4080, 36 months, 7, 24%, 123, 75, A, A3, 10+ years, MORTGAGE, 64000, Source Verified, Dec-16, Current, n, home improvement, 282xx, NC, 25, 73, 3, Mar-88, 708, 784, 0, 15, 18
25 | 95010241, 6050, 6050, 6050, 36 months, 22.74%, 233, 38, E, E1, 5 years, MORTGAGE, 20000, Not Verified, Dec-16, Fully Paid, n, debt consolidation, 319xx, GA, 32, 27, 0, Mar-89, 685, 689, 0, 83, 12
26 | 95324641, 21000, 21000, 21000, 36 months, 13.99%, 717, 63, C, C3, 10+ years, MORTGAGE, 78000, Not Verified, Dec-16, Fully Paid, n, debt consolidation, 891xx, NV, 7, 19, 0, Jul-05, 685, 689, 1, 45
27 | 96229311, 20000, 20000, 20000, 36 months, 6.99%, 61, 75, A, A2, 7 years, MORTGAGE, 60000, Not Verified, Dec-16, Fully Paid, n, medical, 584xx, ND, 23, 04, 0, Jul-99, 770, 774, 0, , 14, 0, 17081, 27, 20
28 | 95994281, 18000, 18000, 18000, 36 months, 12.74%, 335, 69, C, C1, 10+ years, MORTGAGE, 75000, Not Verified, Dec-16, Fully Paid, n, debt consolidation, 972xx, OR, 30, 96, 5, Jun-99, 666, 664, 0, 1
29 | 94203000, 12000, 12000, 12000, 36 months, 12.74%, 402, 83, C, C1, 5 years, MORTGAGE, 180000, Source Verified, Dec-16, Current, n, debt consolidation, 928xx, CA, 11, 86, 0, Mar-90, 680, 684, 0, 7
30 | 96229242, 18000, 18000, 18000, 36 months, 26.24%, 484, 19, E, E5, 3 years, MORTGAGE, 52000, Source Verified, Dec-16, Fully Paid, n, other, 131xx, VA, 20, 79, 0, May-11, 675, 679, 2, , 13, 0, 8278, 3
31 | 95179761, 24000, 24000, 24000, 60 months, 18.99%, 622, 45, D, D3, 7 years, MORTGAGE, 77000, Not Verified, Dec-16, Current, n, debt consolidation, 770xx, TX, 17, 49, 0, Aug-00, 705, 709, 0, , 13, 0

```

Positive is the use of only a Mapper and no need of partitioner and reducer

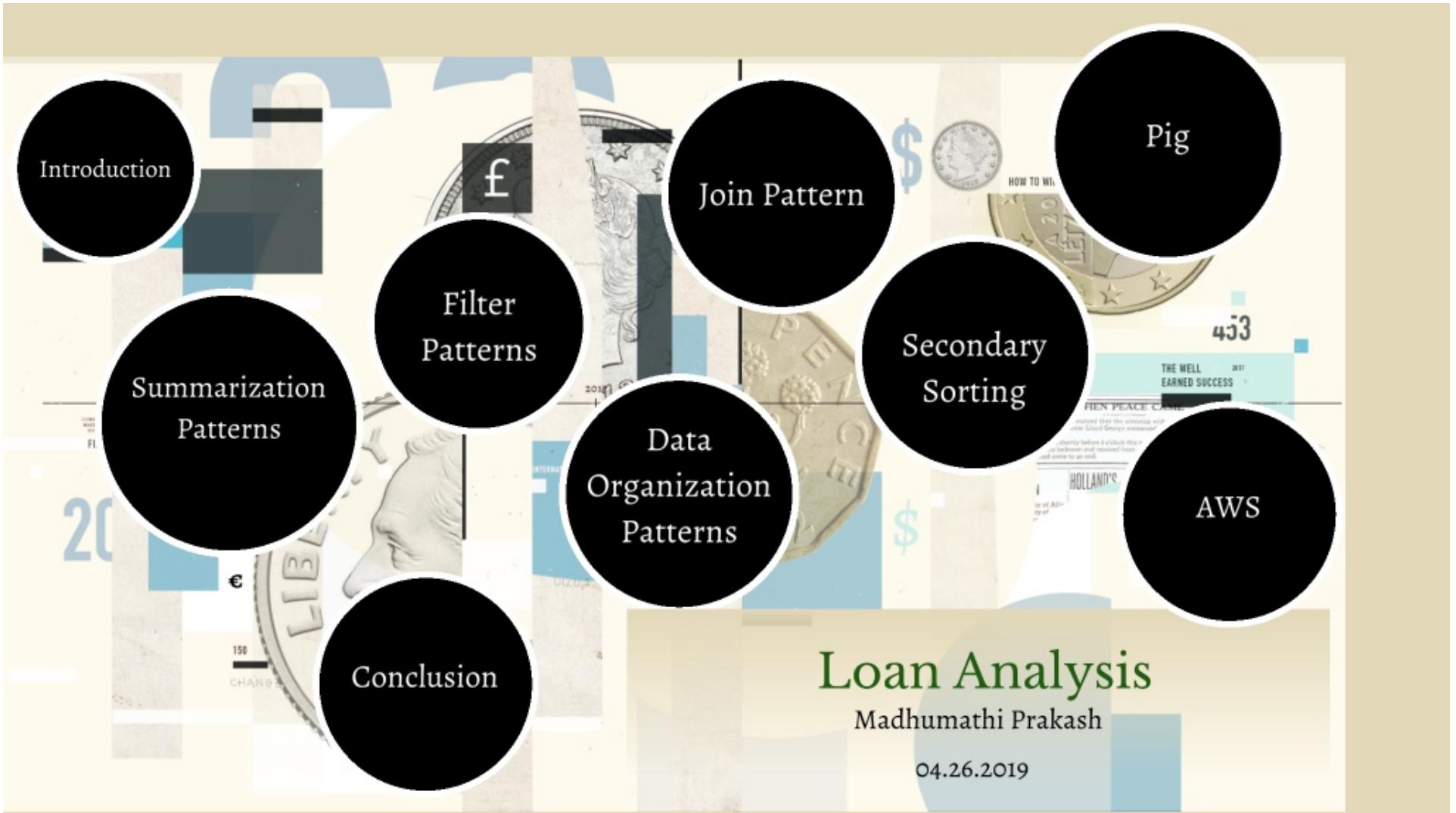
```

@Override
protected void setup(Context context) throws IOException, InterruptedException {
    mos = new MultipleOutputs(context);
}

@Override
public void map(Object key, Text value, Context context)
    throws IOException, InterruptedException {
    String line = value.toString();
    String[] row[] = value.toString().split("\\r\\n");
    if(row.length > 15 && !line.contains("home_ownership")) {
        String cancellationCode = row[11];
        // System.out.println(cancellationCode);
        if(cancellationCode.contains("MORTGAGE"))
            mos.write(namesOutput, "bins", value, NullWritable.get(), baseOutputPath, "loans-mortgage-home");
        if(cancellationCode.contains("OWN"))
            mos.write(namesOutput, "bins", value, NullWritable.get(), baseOutputPath, "loans-own-home");
        if(cancellationCode.contains("RENT"))
            mos.write(namesOutput, "bins", value, NullWritable.get(), baseOutputPath, "loans-rent-home");
    }
}

```

Shuffled the Data in order to randomize the data

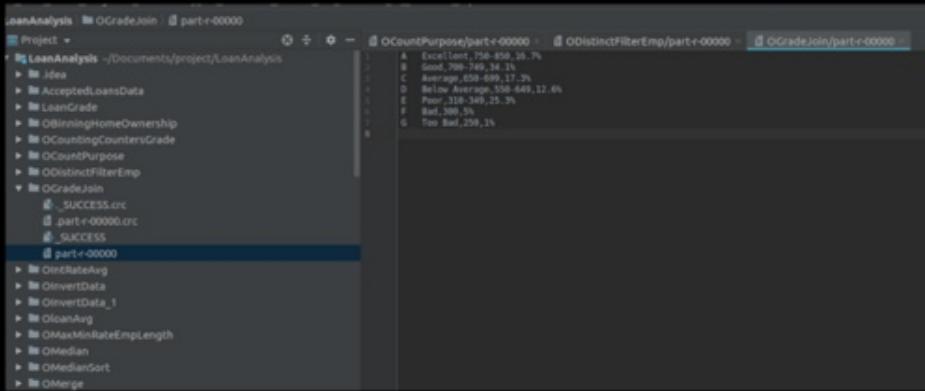


# Loan Analysis

Madhumathi Prakash

04.26.2019

# Used Joins to join the distinct grade to its data in different csv



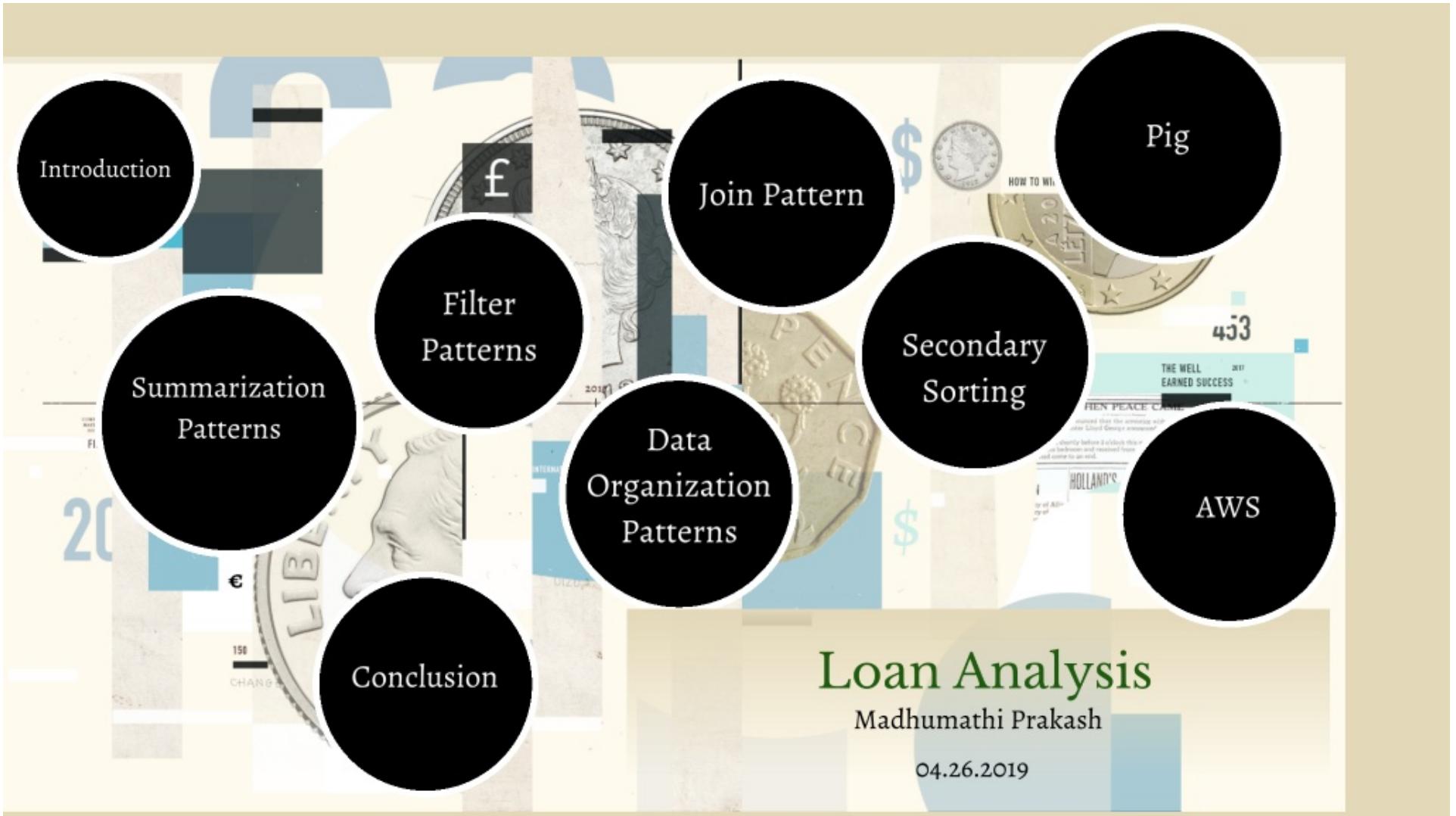
The other csv contained data on what the grade means and the credit score as well as the percent of people who usually have the grade

Driver: have 2 mapper classes

```
MultipleInputs.addInputPath(job, new Path(args[0]), TextInputFormat.class, DistinctMapper.class);
MultipleInputs.addInputPath(job, new Path(args[1]), TextInputFormat.class, CSVMapper.class);
```

```
public void executeJoin(Context context) throws IOException, InterruptedException {
    if(joinType.equalsIgnoreCase("inner")){
        if(!r.isEmpty() && !m.isEmpty()){
            for(Text ra: r){
                for(Text mo: m){
                    context.write(ra,mo);
                }
            }
        }
    }
}
```

Reducer has the inner join

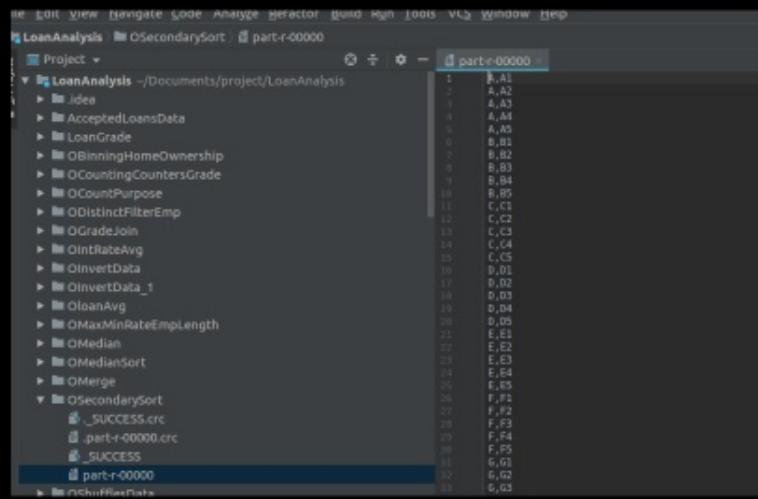


# Loan Analysis

Madhumathi Prakash

04.26.2019

# Sorted the Grade and Subgrade using Secondary Sorting Technique



```
job.setGroupingComparatorClass(GroupComparator.class);
job.setSortComparatorClass(SecodarySortComparator.class);
job.setPartitionerClass(KeyPartitioner.class);
```

```
job.setOutputKeyClass(CompositeKeyWritable.class);
job.setOutputValueClass(NullWritable.class);
```

Driver: group comparator, sec sort comparator,  
composite key partitionor

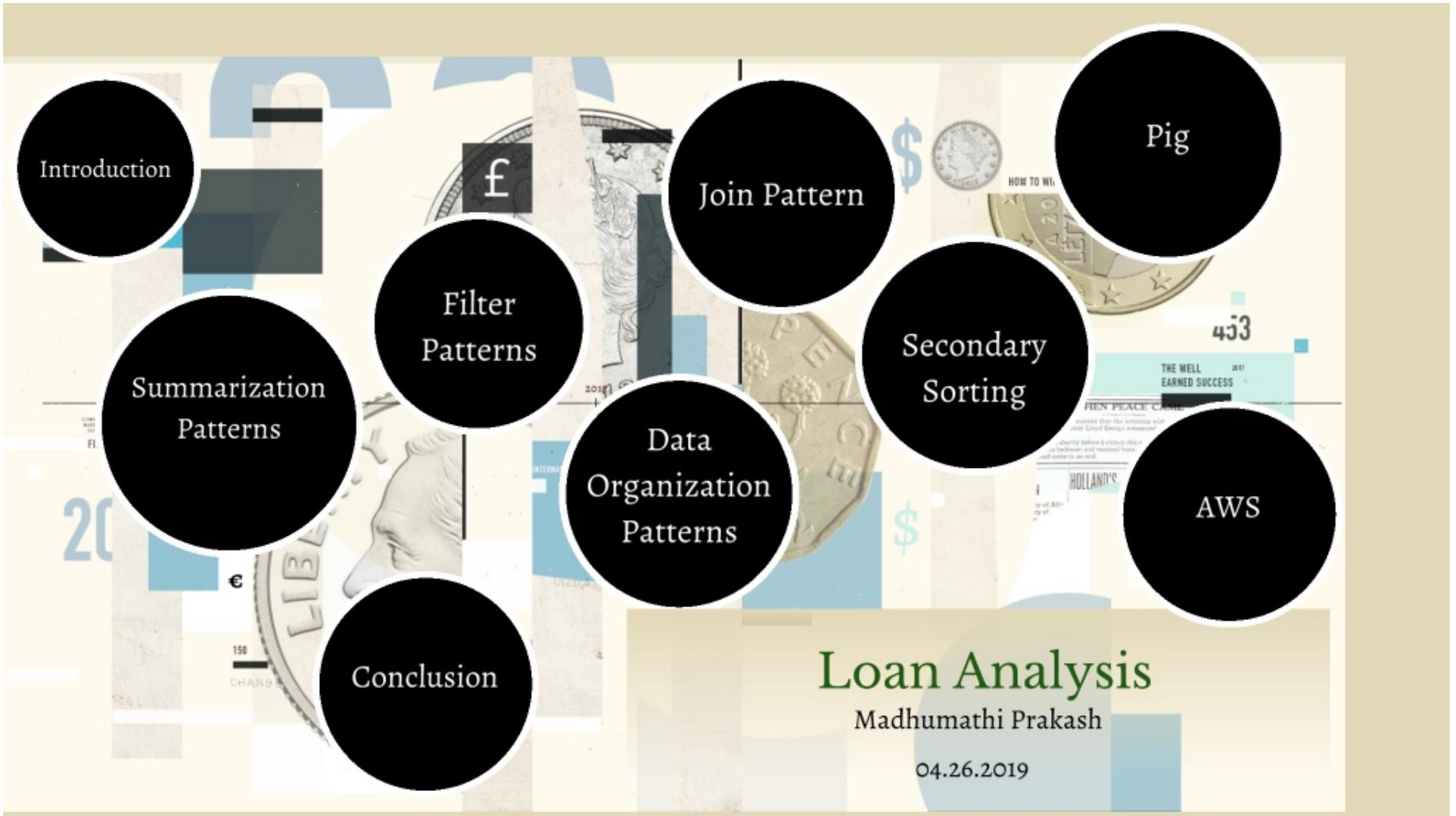
Sec sort has compositeKey class that implements  
WritableComparable

```
protected GroupComparator() { super(CompositeKeyWritable.class, createInstances: true); }

@Override
public int compare(Object a, Object b) {
    CompositeKeyWritable ckw1 = (CompositeKeyWritable)a;
    CompositeKeyWritable ckw2 = (CompositeKeyWritable)b;
    return ckw1.getGrade().compareTo(ckw2.getGrade());
}
```

```
@Override
public int getPartition(CompositeKeyWritable key, NullWritable value, int numPartitions) {
    return key.getGrade().hashCode()%numPartitions;
}
```

Group comparator and the sec sort comparator  
extend WritableComparable



# Loan Analysis

Madhumathi Prakash

04.26.2019



## Pig - data flow language

1. Found Top 25 fico scores based on count using Pig
2. Filtered the data based on the loan less than 3000
3. Join to show grade info

Top K

Filter

Join

# Top 25 fico scores based on count using Pig

The image shows two terminal windows side-by-side. The left window displays a Pig Latin script named `topk.pig` with the following code:

```
users = LOAD 'LoanStats3a_securev1.csv' using PigStorage(',');
grouped = GROUP users BY $24;
summed = FOREACH grouped GENERATE group, COUNT(users) AS cntd;
sorted = ORDER summed BY cntd DESC;
top25 = LIMIT sorted 25;
STORE top25 into 'topk_ficoscore_pig.csv';
```

The right window shows the output of the script, listing 25 fico scores in descending order:

FICO Score	Count
589	2318
704	2267
584	2228
599	2202
694	2196
579	1994
709	1970
724	1949
719	1891
729	1891
674	1854
569	1821
714	1771
734	1670
739	1653
564	1584
744	1490
749	1319
754	1294
759	1100
764	984
774	799
769	792
779	654
784	573

# Filtered the data based on the loan less than 3000

The screenshot shows a Jupyter Notebook interface with two tabs. The left tab, titled 'SmallLoan.pig', contains a Pig Latin script for filtering data from a CSV file. The right tab, titled 'part-m-00000', displays the filtered data as a table.

**Pig Script (SmallLoan.pig):**

```
users = LOAD 'LoanStats3a_securev1.csv' using PigStorage(',');
s1 = FILTER users BY $2 < 3000;
s2 = FOREACH s1 GENERATE $0,$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,'T';
STORE s2 into 'SmallLoanAmount';
```

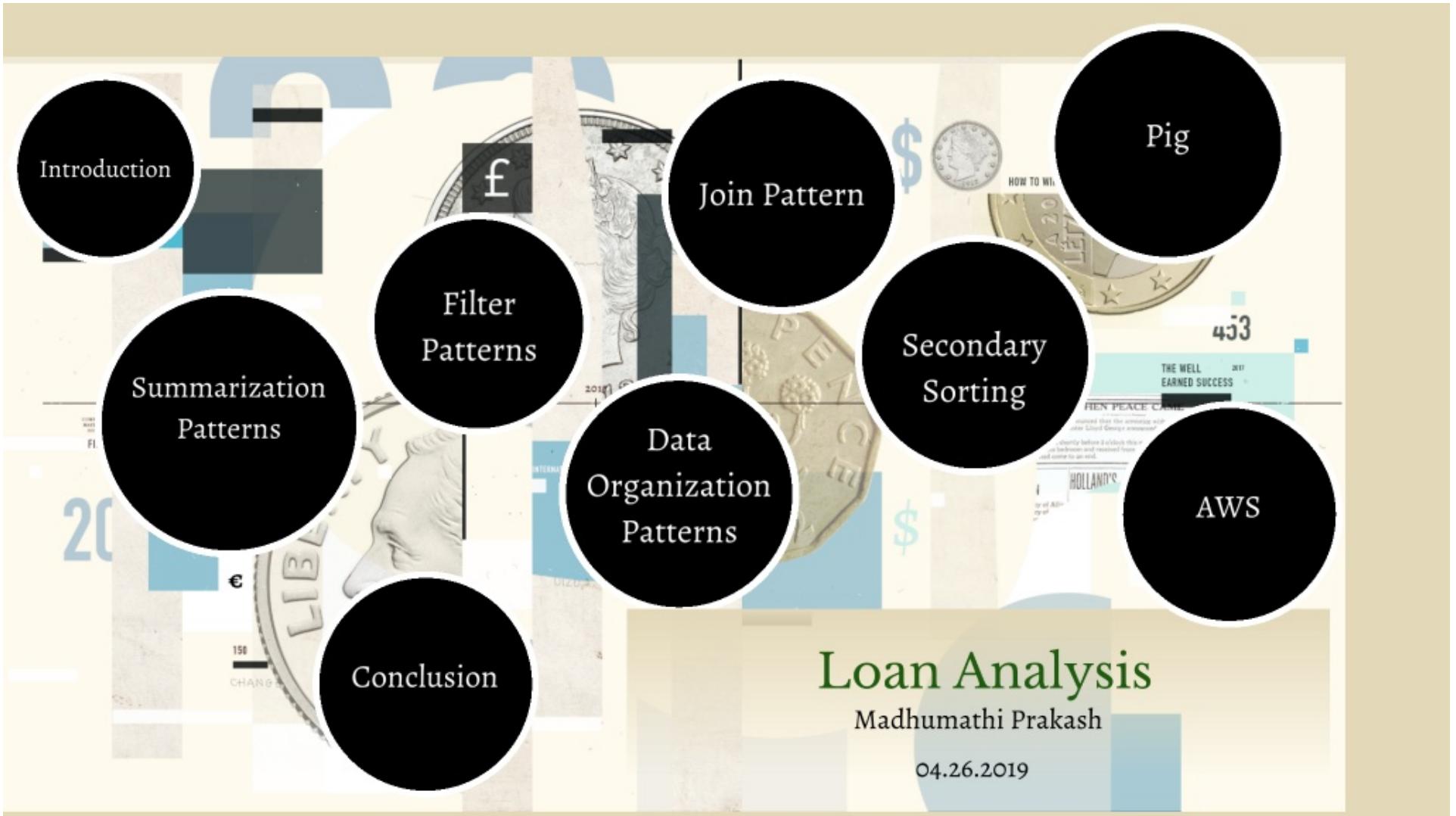
**Output Table (part-m-00000):**

ID	Type	Amount	Source	Verified	Month	Rate	Score	Category	Category2	Category3
1077430	RENT	2500	2500	08 months	15.27%	59.83	C	C4	< 1	
1077175	RENT	30000	Source	Verified	Dec-11 T	15.96%	84.33	C	C5	10+
1069759	RENT	12252	Not Verified		Dec-11 T					
1068292	RENT	1000	1000	36 months	16.29%	35.31	D	D1	< 1	
1068292	MORTGAGE	28000	Not Verified		Dec-11 T	12.42%	78.18	B	B4	10+
1068180	RENT	2100	2100	36 months	14.27%	85.78	C	C2	7 years	
1067644	OWN	49500	Source	Verified	Dec-11 T					
1067644	MORTGAGE	2500	2500	36 months	7.51%	77.78	A	A3	n/a	
1066836	RENT	12000	Source	Verified	Dec-11 T					
1066617	RENT	27200	Not Verified		Dec-11 T	13.49%	84.83	C	C1	4 years
1066617	MORTGAGE	41000	Verified		Dec-11 T					
1066617	RENT	2000	2000	36 months	15.27%	69.6	C	C4	2 years	
1066617	MORTGAGE	1500	1500	36 months	17.27%	53.69	D	D3	10+	
1066598	RENT	144000	Verified		Dec-11 T					
1066598	MORTGAGE	2500	2500	36 months	11.71%	82.69	B	B3	7 years	
1065382	RENT	29000	Verified		Dec-11 T					
1065382	MORTGAGE	1400	1400	36 months	7.90%	43.81	A	A4	1 year	
1066081	RENT	28800	Not Verified		Dec-11 T					
1066081	MORTGAGE	2000	2000	36 months	7.90%	43.81	A	A4	4 years	
1066081	RENT	1800	1800	36 months	6.62%	55.27	A	A2	10+	
1064453	RENT	18000	Not Verified		Dec-11 T					
1064453	MORTGAGE	2000	2000	36 months	7.90%	62.59	A	A4	1 year	
1064251	RENT	39600	Not Verified		Dec-11 T					
1064251	MORTGAGE	1500	1500	36 months	18.64%	54.72	E	E1	< 1	
1064366	RENT	17000	Source	Verified	Dec-11 T					
1064366	MORTGAGE	1000	1000	36 months	14.65%	34.5	C	C3	n/a	
1064008	RENT	1000	Not Verified		Dec-11 T					
1064008	MORTGAGE	18408	Verified		Dec-11 T					
1064008	RENT	2625	2625	36 months	11.71%	86.83	B	B3	10+	
1063448	RENT	52800	Not Verified		Dec-11 T					
1063448	MORTGAGE	2200	2200	36 months	15.27%	76.56	C	C4	10+	
1062897	RENT	48000	Not Verified		Dec-11 T					
1062897	MORTGAGE	1700	1700	36 months	7.90%	53.2	A	A4	n/a	
1062510	RENT	37800	Not Verified		Dec-11 T					
1062510	MORTGAGE	2000	2000	36 months	9.91%	64.45	B	B1	1 year	

# Join to display grade with more info

```
grade = LOAD 'grade' using PigStorage(',') as (grd);
detail = LOAD 'gradedetails.csv' using PigStorage(',') as (id,desc,name,creditscore);
joinop = JOIN grade BY grd, detail BY id;
out = FOREACH joinop GENERATE $0,$2,$3,$4;
STORE summed out 'gradeoutput';
```

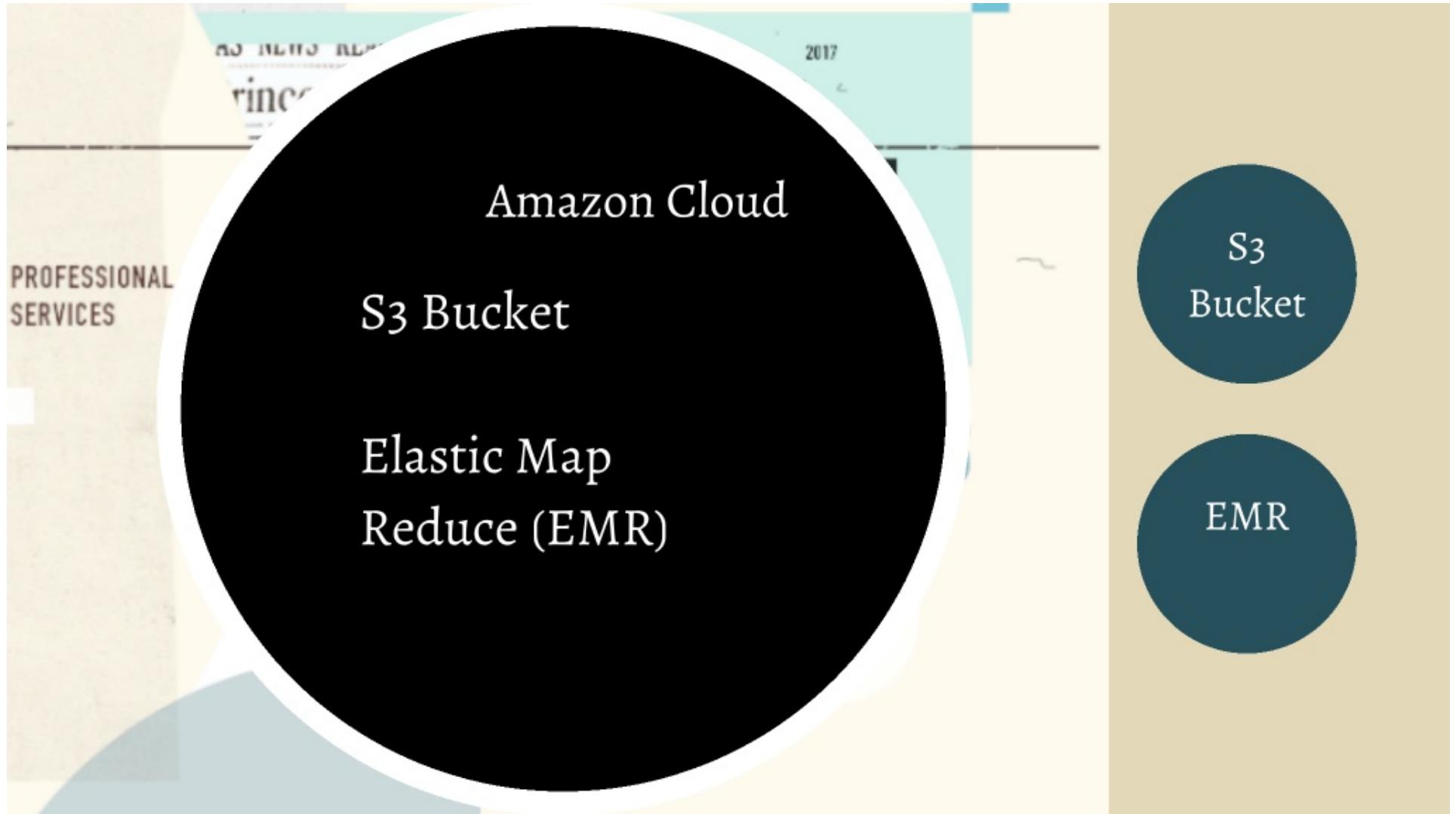
A	Excellent,750-850,16.7%
B	Good,700-749,34.1%
C	Average,650-699,17.3%
D	Below Average,550-649,12.6%
E	Poor,310-349,25.3%
F	Bad,300,5%
G	Too Bad,250,1%



# Loan Analysis

Madhumathi Prakash

04.26.2019



# S3 Bucket

input folder

The screenshot shows the AWS S3 console with the path "Amazon S3 > Nameofmybucket > input". The "Overview" tab is selected. The table lists the following objects:

Name	Last modified	Size	Storage class
LoadBatch_secure1.csv	Apr 24, 2019 9:33:00 PM GMT-0400	15.4 MB	Standard
LoadBatch_secure2.csv	Apr 24, 2019 9:33:00 PM GMT-0400	81.1 MB	Standard
LoadBatch_secure3.csv	Apr 24, 2019 9:33:00 PM GMT-0400	104.5 MB	Standard
LoadBatch_secure4.csv	Apr 24, 2019 9:33:00 PM GMT-0400	140.0 MB	Standard
LoadBatch_secure5.csv	Apr 24, 2019 9:33:00 PM GMT-0400	62.8 MB	Standard
LoadBatch_secure6_2019Q2.csv	Apr 24, 2019 9:33:51 PM GMT-0400	46.0 MB	Standard
LoadBatch_secure7_2019Q2.csv	Apr 24, 2019 9:36:13 PM GMT-0400	46.7 MB	Standard
LoadBatch_secure8_2019Q4.csv	Apr 24, 2019 9:36:27 PM GMT-0400	45.9 MB	Standard
LoadBatch_secure9_2019Q4.csv	Apr 24, 2019 9:37:00 PM GMT-0400	45.8 MB	Standard

JAR File

The screenshot shows the AWS S3 console with the path "Amazon S3 > Nameofmybucket > pr". The "Overview" tab is selected. The table lists the following object:

Name	Last modified	Size	Storage class
loadanalysis.LD.GraduateTest.jar	Apr 24, 2019 9:22:40 PM GMT-0400	59.4 KB	Standard

Output Folder

The screenshot shows the AWS S3 console with the path "Amazon S3 > Nameofmybucket > part". The "Overview" tab is selected. The table lists the following objects:

Name	Last modified	Size	Storage class
_SUCCESS	Apr 24, 2019 9:49:48 PM GMT-0400	0 B	Standard
part-00000	Apr 24, 2019 9:49:48 PM GMT-0400	251.0 B	Standard

# EMR

Cluster: My cluster Waiting Cluster ready after last step completed.

Summary Application history Monitoring Hardware Configurations Events Steps Bootstrap actions

Add step Clone step Cancel step

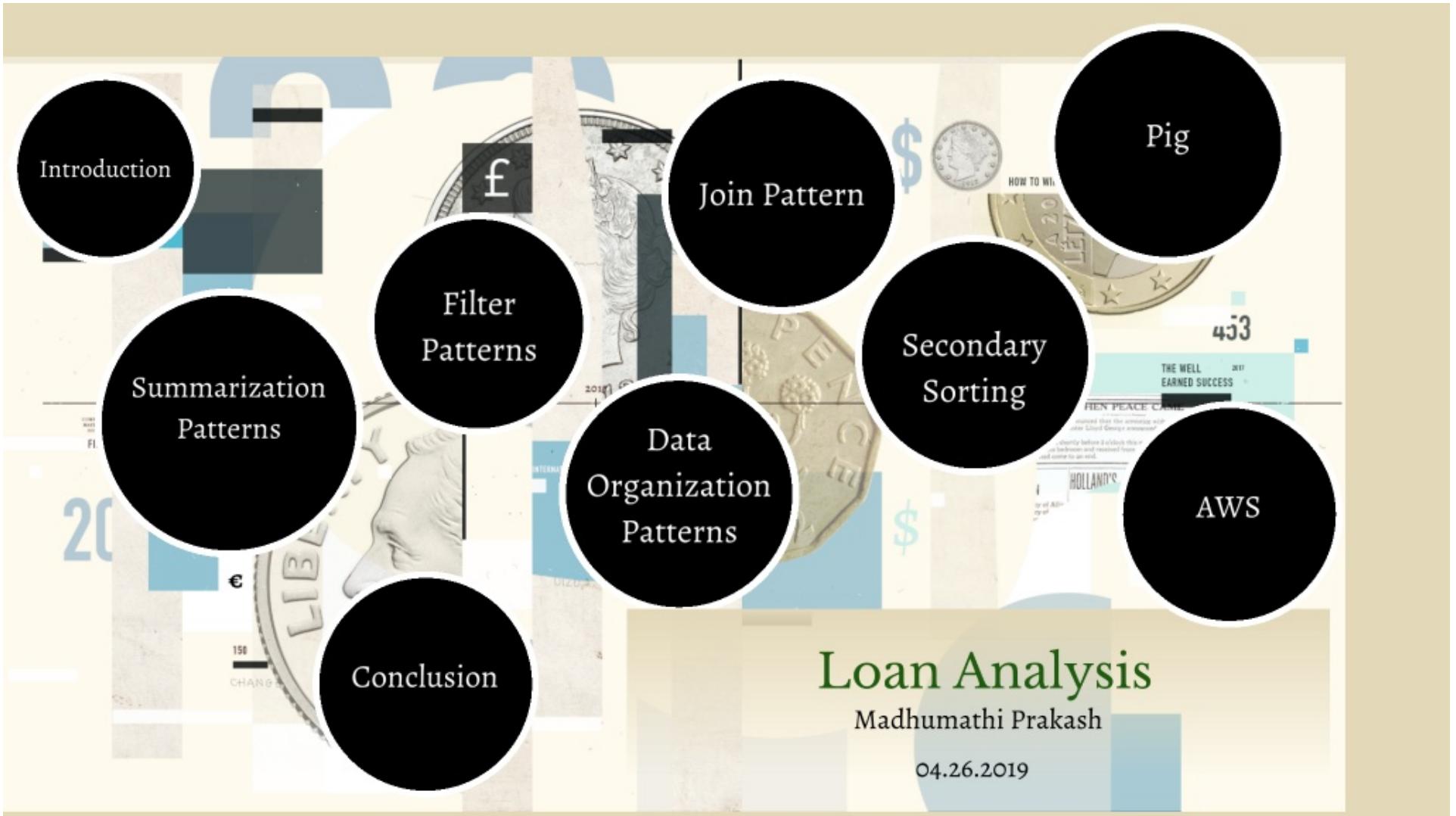
Steps

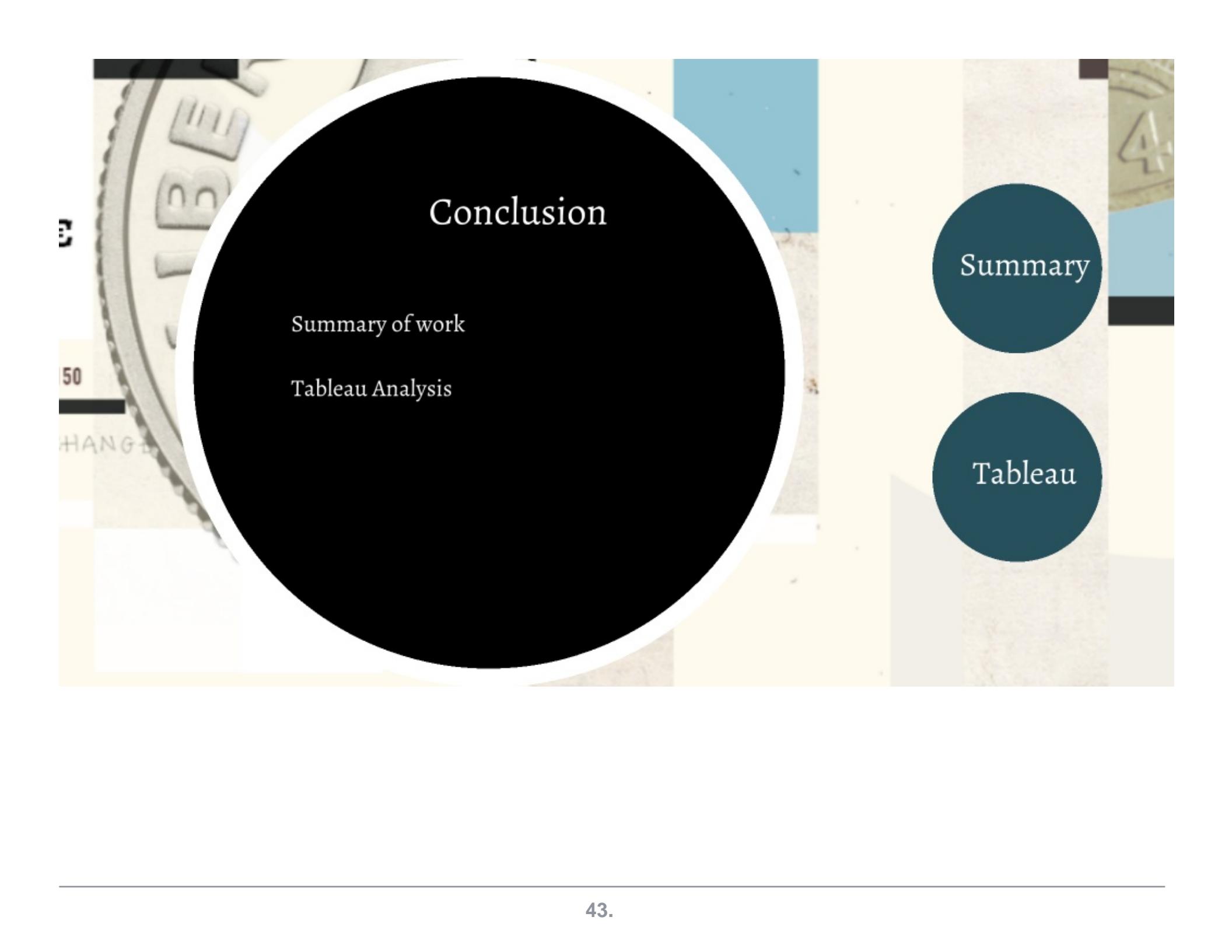
Filter: All steps ▾ Filter steps ... 2 steps (all loaded) C View all interactive jobs

ID	Name	Status	Start time (UTC-4)	Elapsed time	Log files
s-12EDD1UM8F02P	Custom JAR	Completed	2019-04-24 21:47 (UTC-4)	2 minutes	<a href="#">View logs</a>

JAR location : s3://loananalysishadoop/jar/LoanAnalysis-1.0-SNAPSHOT.jar  
Main class : None  
Arguments : CountPurposeDetermineNumberLoans.CountPurposeDriver s3://loananalysishadoop/input s3://loananalysishadoop/output  
Action on failure: Continue

```
car 24013
credit_card 516971
debt_consolidation 1277877
educational 424
home_improvement 150457
house 14136
major_purchase 50445
medical 27488
moving 15403
other 139440
purpose 16
renewable_energy 1445
small_business 24689
vacation 15525
wedding 2355
```





# Conclusion

Summary of work

Tableau Analysis

Summary

Tableau

# Summary Of Work Done

Overview	Work done
Summarization patterns	Min and max
	Nline
	Counting
	Average
	Median
	Memory conscious Median
	Custom Combiner
	Inverted index
	Counting with counters
Filter Patterns	
	Bloom filtering
	Top K
	Distinct
Data org Patterns	
	Binning
	Shuffling
Joins	inner joins
Secondary Sort	WritableComparable
Chaining	
Pig	
	Top K
	Filtering
	Join
AWS Cloud	

# Tableau Analysis

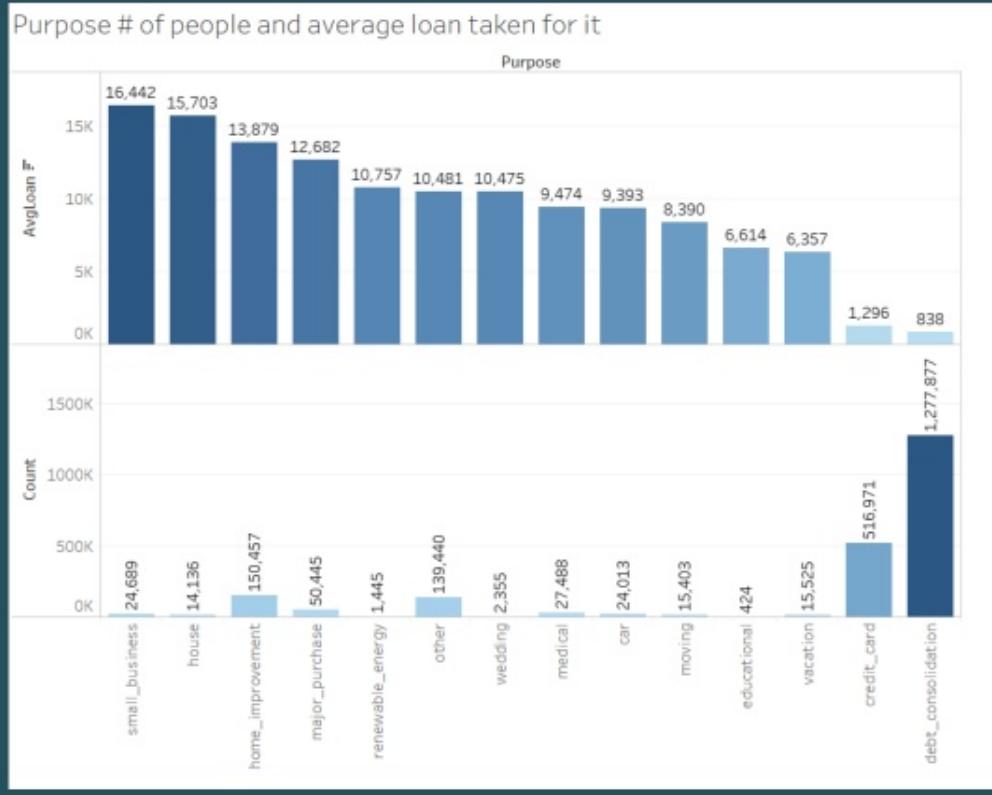
Analysis 1

Analysis 2

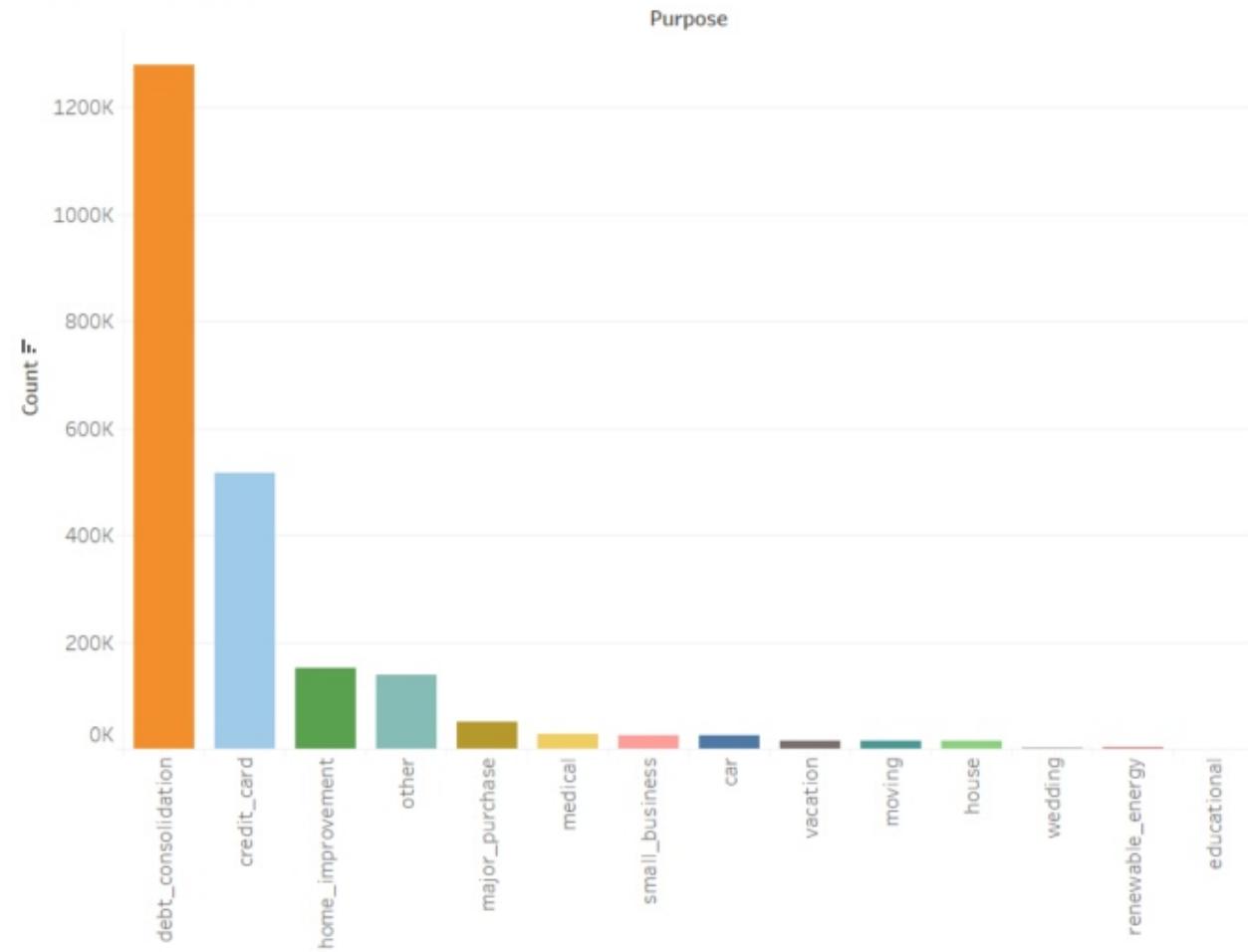
Analysis 3

Analysis 4

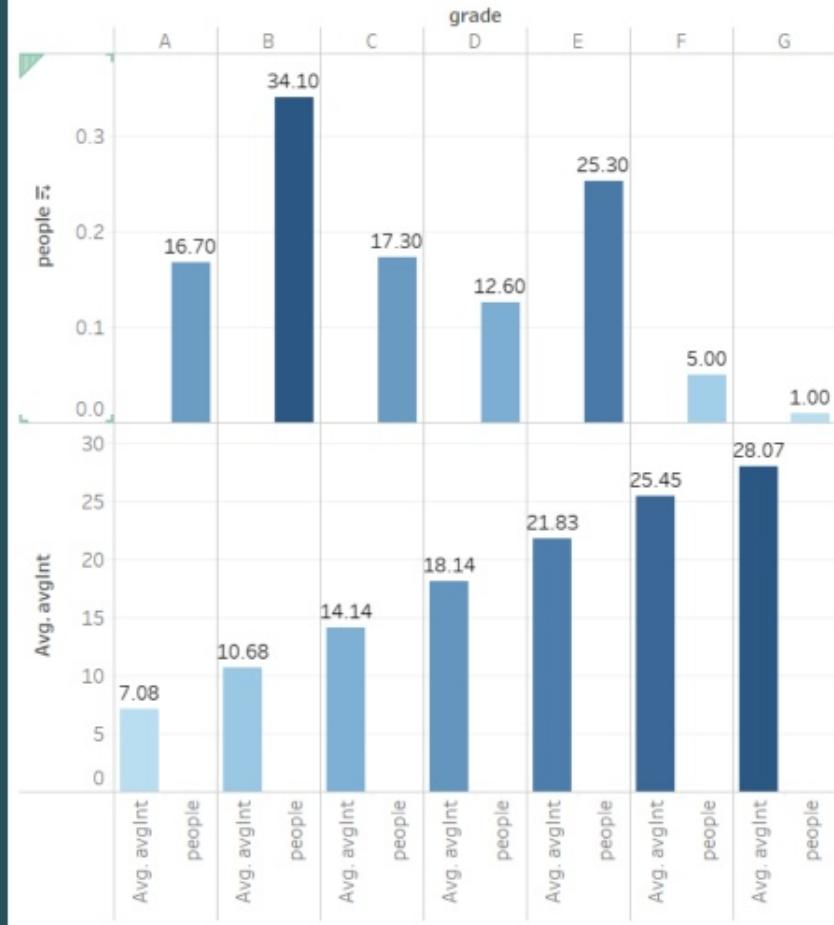
# Purpose # of people and average loan taken for it



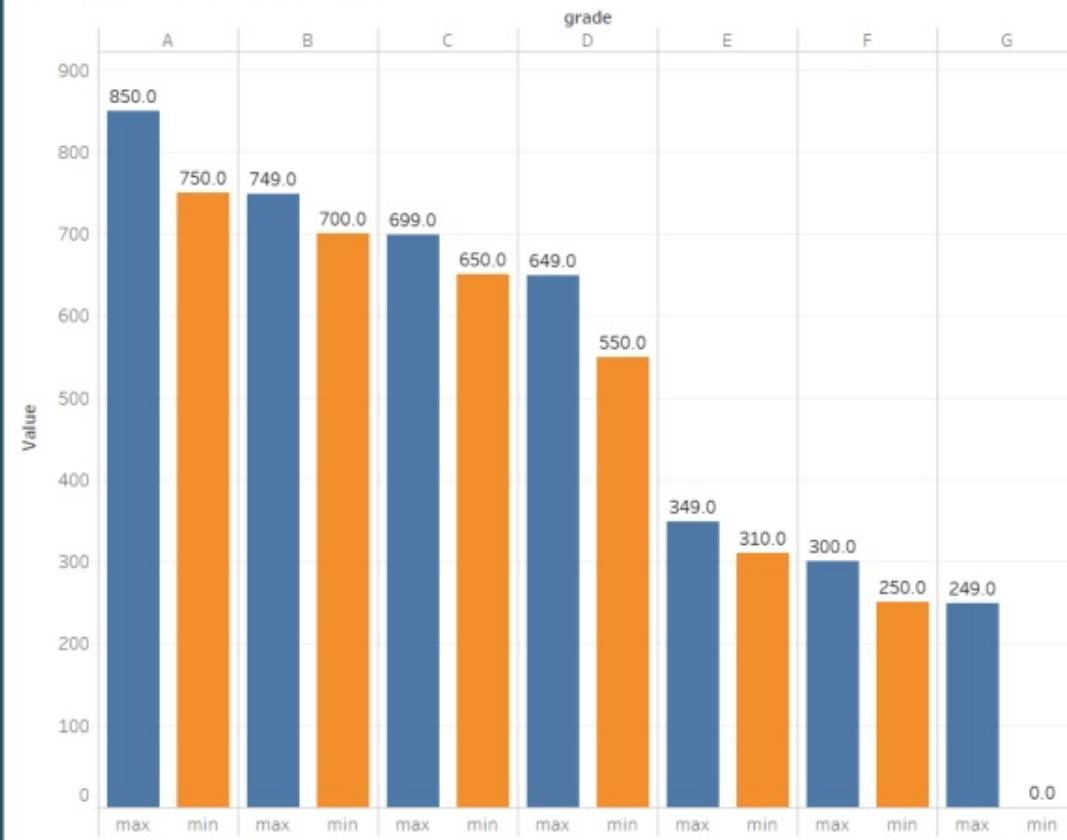
## Count based on Purpose

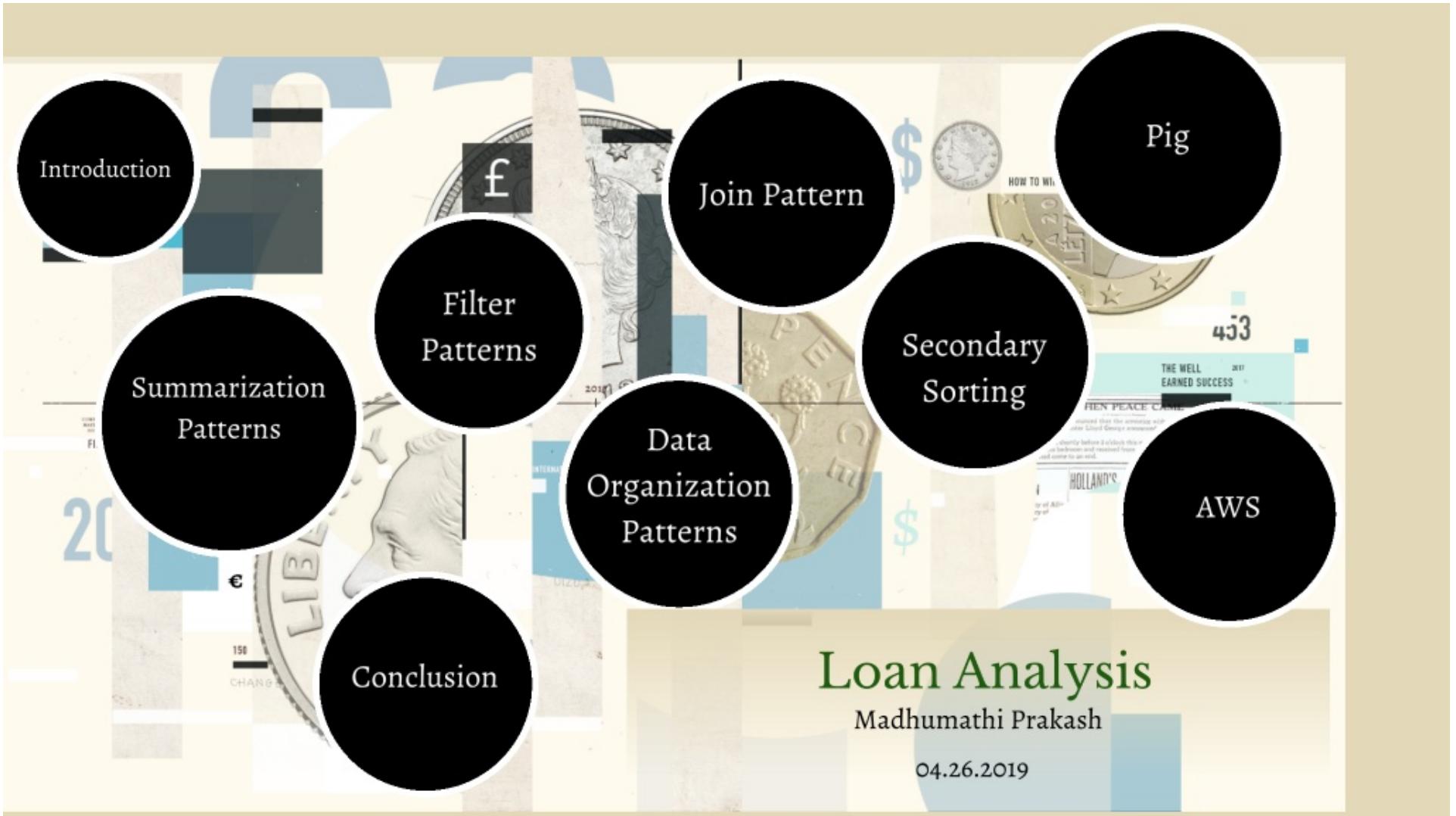


Avg interest per grade and % people



Max and Min fico of each grade





# Loan Analysis

Madhumathi Prakash

04.26.2019

# Appendix

## AverageInterestRateByGrade

```
package AverageInterestRateByGrade;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.Configuration;
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 AverageInterestRateByGrade {
    public static class MapLoan extends Mapper<LongWritable, Text, Text, LoanAveragingPair> {
        private Text outText = new Text();
        private Map<String,LoanAveragingPair> pairMap = new HashMap<String,LoanAveragingPair>();
        private LoanAveragingPair pair = new LoanAveragingPair();

        public void map(LongWritable key, Text value, Context context) throws IOException,
        InterruptedException {
            String line = value.toString();
            String[] columnValues = line.split(",");

```

```

if (columnValues.length > 6) {
    if (!columnValues[0].contains("id")) {
        outText.set(columnValues[8]);
        Float LoanTime = Float.parseFloat(columnValues[6].replace("%", ""));
        LoanAveragingPair pair = pairMap.get(columnValues[8]);
        if (pair == null) {
            pair = new LoanAveragingPair();
            pairMap.put(columnValues[8], pair);
        }
        float Loan = pair.getLoan().get() + LoanTime;
        int count = pair.getCount().get() + 1;
        pair.set(Loan, count);
    }
}
}

@Override
protected void cleanup(Context context) throws IOException, InterruptedException {
    Set<String> keys = pairMap.keySet();
    Text keyText = new Text();
    for (String key : keys) {
        keyText.set(key);
        context.write(keyText, pairMap.get(key));
    }
}
}

public static class Reduce extends Reducer<Text, LoanAveragingPair, Text, FloatWritable> {

```

```

public void reduce(Text key, Iterable<LoanAveragingPair> values
    , Context context)
    throws IOException, InterruptedException {
    float sum = 0;
    int count = 0;
    for (LoanAveragingPair val : values) {
        sum += val.getLoan().get();
        count += val.getCount().get();
    }
    context.write(key, new FloatWritable(sum/count));
}

public static class LoanAveragingPair implements Writable, WritableComparable<LoanAveragingPair>
{
    private FloatWritable Loan;
    private IntWritable count;

    public LoanAveragingPair() {
        set(new FloatWritable(0), new IntWritable(0));
    }

    public LoanAveragingPair(Float Loan, int count) {
        set(new FloatWritable(Loan), new IntWritable(count));
    }

    public void write(DataOutput out) throws IOException {
        Loan.write(out);
    }
}

```

```
        count.write(out);

    }

public void readFields(DataInput in) throws IOException {
    Loan.readFields(in);
    count.readFields(in);
}

public int compareTo(LoanAveragingPair other) {
    int compareVal = this.Loan.compareTo(other.getLoan());
    if (compareVal != 0) {
        return compareVal;
    }
    return this.count.compareTo(other.getCount());
}

public void set(Float Loan, int count){
    this.Loan.set(Loan);
    this.count.set(count);
}

public void set(FloatWritable Loan, IntWritable count) {
    this.Loan = Loan;
    this.count = count;
}

public static LoanAveragingPair read(DataInput in) throws IOException {
```

```
    LoanAveragingPair averagingPair = new LoanAveragingPair();
    averagingPair.readFields(in);
    return averagingPair;
}
```

```
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    LoanAveragingPair that = (LoanAveragingPair) o;

    if (!count.equals(that.count)) return false;
    if (!Loan.equals(that.Loan)) return false;

    return true;
}
```

```
@Override
public int hashCode() {
    int result = Loan.hashCode();
    result = 163 * result + count.hashCode();
    return result;
}
```

```
@Override
public String toString() {
    return "LoanAveragingPair{" +
        "Loan=" + Loan +
        ", count=" + count +
```

```
'}';

}

public FloatWritable getLoan() {
    return Loan;
}

public IntWritable getCount() {
    return count;
}

}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

    Job job = new Job(conf, "int rate count");
    job.setJarByClass(AverageInterestRateByGrade.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(LoanAveragingPair.class);

    job.setMapperClass(MapLoan.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);

    }

}
```

### AverageLoanAmtByPurpose

```
package AverageLoanAmtByPurpose;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.Configuration;
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 AverageLoanAmtByPurpose {

    public static class MapLoan extends Mapper<LongWritable, Text, Text, LoanAveragingPair> {

        private Text outText = new Text();

        private Map<String,LoanAveragingPair> pairMap = new HashMap<String,LoanAveragingPair>();

        private LoanAveragingPair pair = new LoanAveragingPair();

        protected void map(LongWritable key, Text value, Mapper.Context context) throws IOException, InterruptedException {
            String[] line = value.toString().split(",");
            if(line.length > 1) {
                pair.setAmount(new LongWritable(Double.parseDouble(line[1])));
                pair.setCategory(line[0]);
                pairMap.put(pair.getCategory(), pair);
            }
        }

        protected void reduce(Text key, Iterable<LoanAveragingPair> values, Reducer.Context context) throws IOException, InterruptedException {
            long sum = 0;
            int count = 0;
            for(LoanAveragingPair val : values) {
                sum += val.getAmount().get();
                count++;
            }
            if(count > 0) {
                double average = sum / count;
                outText.set(String.valueOf(average));
                context.write(key, outText);
            }
        }
    }
}
```

```

public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
    String line = value.toString();
    String[] columnValues = line.split(",");
    // System.out.println(columnValues[0] + " " + columnValues[2]);
    if (columnValues.length > 2) {
        if (!columnValues[2].contains("loan")) {
            outText.set(columnValues[17]);
            int LoanTime = Integer.parseInt(columnValues[2]);
            LoanAveragingPair pair = pairMap.get(columnValues[17]);
            if (pair == null) {
                pair = new LoanAveragingPair();
                pairMap.put(columnValues[17], pair);
            }
            int Loan = pair.getLoan().get() + LoanTime;
            int count = pair.getCount().get() + 1;
            pair.set(Loan, count);
        }
    }
}

@Override
protected void cleanup(Context context) throws IOException, InterruptedException {
    Set<String> keys = pairMap.keySet();
    Text keyText = new Text();
    for (String key : keys) {
        keyText.set(key);
        context.write(keyText, pairMap.get(key));
    }
}

```

```

        }

    }

}

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

    public void reduce(Text key, Iterable<LoanAveragingPair> values
                      , Context context)
                      throws IOException, InterruptedException {

        int sum = 0;
        int count = 0;
        for (LoanAveragingPair val : values) {
            sum += val.getLoan().get();
            count += val.getCount().get();
        }
        context.write(key, new IntWritable(sum/count));
    }
}

public static class LoanAveragingPair implements Writable, WritableComparable<LoanAveragingPair> {

    private IntWritable loan;
    private IntWritable count;

    public LoanAveragingPair() {
        set(new IntWritable(0), new IntWritable(0));
    }

    public LoanAveragingPair(int loan, int count) {
        set(new IntWritable(loan), new IntWritable(count));
    }

    ...
}

```

```
}
```

```
public void write(DataOutput out) throws IOException {  
    Loan.write(out);  
    count.write(out);  
}
```

```
public void readFields(DataInput in) throws IOException {  
    Loan.readFields(in);  
    count.readFields(in);  
}
```

```
public int compareTo(LoanAveragingPair other) {  
    int compareVal = this.Loan.compareTo(other.getLoan());  
    if (compareVal != 0) {  
        return compareVal;  
    }  
    return this.count.compareTo(other.getCount());  
}  
public void set(int Loan, int count){  
    this.Loan.set(Loan);  
    this.count.set(count);  
}
```

```
public void set(IntWritable Loan, IntWritable count) {  
    this.Loan = Loan;
```

```
    this.count = count;
}

public static LoanAveragingPair read(DataInput in) throws IOException {
    LoanAveragingPair averagingPair = new LoanAveragingPair();
    averagingPair.readFields(in);
    return averagingPair;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    LoanAveragingPair that = (LoanAveragingPair) o;

    if (!count.equals(that.count)) return false;
    if (!Loan.equals(that.Loan)) return false;

    return true;
}

@Override
public int hashCode() {
    int result = Loan.hashCode();
    result = 163 * result + count.hashCode();
    return result;
}
```

```
@Override
public String toString() {
    return "LoanAveragingPair{" +
        "Loan=" + Loan +
        ", count=" + count +
        '}';
}

public IntWritable getLoan() {
    return Loan;
}

public IntWritable getCount() {
    return count;
}

}

public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

    Job job = new Job(conf, "loan count");
    job.setJarByClass(AverageLoanAmtByPurpose.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(LoanAveragingPair.class);

    job.setMapperClass(MapLoan.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);
    }
}

```

### **BinningHomeOwnership**

```

package BinningHomeOwnership;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.MultipleOutputs;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.io.IOException;

public class BinningHomeOwnership {
    public static class TMapper extends Mapper<Object, Text, Text, NullWritable>

```

```
{  
  
    private MultipleOutputs<Text, NullWritable> mos=null;  
  
    @Override  
    protected void setup(Context context) throws IOException, InterruptedException  
    {  
        mos = new MultipleOutputs(context);  
    }  
  
    @Override  
    public void map(Object key, Text value, Context context)  
        throws IOException, InterruptedException  
    {  
  
        String line = value.toString();  
        String row[] = value.toString().split(",");  
  
        if(row.length > 15 && !line.contains("home_ownership")) {  
            String cancellationCode = row[11];  
  
            // System.out.println(cancellationCode);  
            if(cancellationCode.contains("MORTGAGE"))  
                mos.write("bins", value, NullWritable.get(),"loans-mortgage-home");  
            if(cancellationCode.contains("OWN"))  
                mos.write("bins", value, NullWritable.get(),"loans-own-home");  
            if(cancellationCode.contains("RENT"))  
                mos.write("bins", value, NullWritable.get(),"loans-rent-home");  
        }  
    }  
}
```

```
    }

}

@Override
protected void cleanup(Context context) throws IOException, InterruptedException
{
    mos.close();
}

}

public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException
{
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Cancellation Binning ");
    job.setJarByClass(BinningHomeOwnership.class);
    job.setMapperClass(TMapper.class);

    MultipleOutputs.addNamedOutput(job, "bins", TextOutputFormat.class, Text.class,
    IntWritable.class);
```

```
    MultipleOutputs.setCountersEnabled(job, true);

    job.setNumReduceTasks(0);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(NullWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);

}

}
```

### **CountingWithCountersGrade**

```
package CountingWithCountersGrade;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.mapreduce.Counter;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

public class Driver {

    public static void main( String[] args ) throws IOException, ClassNotFoundException,
    InterruptedException
    {
        Job job = Job.getInstance();

```

```
job.setJarByClass(Driver.class);

FileInputFormat.addInputPath(job, new Path(args[0]));
Path outDir = new Path(args[1]);
FileOutputFormat.setOutputPath(job, outDir);

job.setMapperClass(MyMapper.class);

job.setOutputKeyClass(NullWritable.class);
job.setOutputValueClass(NullWritable.class);
;

int code = job.waitForCompletion(true)? 1 : 0;
if (code==1 {

    for(Counter counter: job.getCounters().getGroup("MonthCounter")) {
        //System.out.print("inside this");
        System.out.println("the counter "+counter.getDisplayName()+" is "+counter.getValue());
        // System.out.println("the counter values is "+counter.getValue());
    }
}
```

```
 }  
 }
```

```
package CountingWithCountersGrade;  
  
import org.apache.hadoop.io.LongWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Mapper;  
  
import java.io.IOException;  
import java.util.Arrays;  
import java.util.HashSet;  
  
public class MyMapper extends Mapper<LongWritable, Text, NullWritable,NullWritable> {  
    @Override  
    protected void map(LongWritable key, Text value, Context context) throws IOException,  
    InterruptedException {  
  
        String grades[] = {"A", "B", "C", "D", "E", "F", "G"};  
        HashSet<String> gradesHashSet = new HashSet<String>(Arrays.asList(grades));  
  
        String input = value.toString();  
        String splitString[] = input.split(",");  
        if(splitString.length > 8) {  
            String grade = splitString[8];  
            // System.out.print(month);  
  
            if (gradesHashSet.contains(grade) && splitString[8] != "grade") {
```

```
// System.out.print(month);

context.getCounter("MonthCounter", grade).increment(1);

} else {

    context.getCounter("MonthCounter", "unknown").increment(1);

}

}

}

}
```

### **CountPurposeDetermineNumberLoans**

```
package CountPurposeDetermineNumberLoans;

import org.apache.hadoop.fs.FileSystem;
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.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;

import java.io.IOException;

public class CountPurposeDriver {
```

```
public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException {
    Job job = Job.getInstance();
    job.setJarByClass(CountPurposeDriver.class);

    job.setMapperClass(CountPurposeMapper.class);
    job.setCombinerClass(CountPurposeReducer.class);
    job.setReducerClass(CountPurposeReducer.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setNumReduceTasks(1);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    FileSystem fs = FileSystem.get(job.getConfiguration());
    /*
    Path outDir = new Path(args[1]);
    if(fs.exists(outDir)) {
        fs.delete(outDir, true);
    }
    Submit the job, then poll for progress until the job is complete
*/
```

```
        job.waitForCompletion(true);

    }

}

package CountPurposeDetermineNumberLoans;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class CountPurposeMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    private static final IntWritable one = new IntWritable(1);

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
        String line = value.toString();

        String[] tokens = line.split(",");
        if(tokens.length > 15) {
            // for(String s : tokens) {
            // System.out.println(tokens[17]);
            context.write(new Text(tokens[17]), one);
        }
    //}

```

```
    }

@Override
protected void cleanup(Mapper<LongWritable, Text, Text, IntWritable>.Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.cleanup(context);
}

@Override
public void run(Mapper<LongWritable, Text, Text, IntWritable>.Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.run(context);
}

@Override
protected void setup(Mapper<LongWritable, Text, Text, IntWritable>.Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.setup(context);
}

}

package CountPurposeDetermineNumberLoans;
```

```
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class CountPurposeReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    @Override
    protected void cleanup(Reducer<Text, IntWritable, Text, IntWritable>.Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.cleanup(context);
    }

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
        Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException,
        InterruptedException {
        int sum = 0;

        for (IntWritable value : values) {
            sum += value.get();
        }

        context.write(key, new IntWritable(sum));
    }
}
```

```

@Override
public void run(Reducer<Text, IntWritable, Text, IntWritable>.Context arg0)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.run(arg0);
}

@Override
protected void setup(Reducer<Text, IntWritable, Text, IntWritable>.Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.setup(context);
}

}

```

### **DistinctFilteringByGrade**

```

package DistinctFilteringByGrade;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
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;

```

```
import java.io.IOException;

public class DistinctDriver {

    public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException {

        Job job = Job.getInstance();
        job.setJarByClass(DistinctDriver.class);

        job.setMapperClass(DistinctMapper.class);
        job.setCombinerClass(DistinctReducer.class);
        job.setReducerClass(DistinctReducer.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(NullWritable.class);

        job.setNumReduceTasks(1);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        FileSystem fs = FileSystem.get(job.getConfiguration());

        job.waitForCompletion(true);
    }
}
```

```
package DistinctFilteringByGrade;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class DistinctMapper extends Mapper<LongWritable, Text, Text, NullWritable> {

    private static final IntWritable one = new IntWritable(1);

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, NullWritable>.Context context)
        throws IOException, InterruptedException {
        String line = value.toString();

        String[] tokens = line.split(",");
        if(tokens.length > 15 && !line.contains("grade")) {
            context.write(new Text(tokens[8]), NullWritable.get()); //10
        }
    }

    @Override
    protected void cleanup(Mapper<LongWritable, Text, Text, NullWritable>.Context context)
```

```
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.cleanup(context);
}

@Override
public void run(Mapper<LongWritable, Text, Text, NullWritable>.Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.run(context);
}

@Override
protected void setup(Mapper<LongWritable, Text, Text, NullWritable>.Context context)
    throws IOException, InterruptedException {
    // TODO Auto-generated method stub
    super.setup(context);
}

}

package DistinctFilteringByGrade;

import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
```

```
public class DistinctReducer extends Reducer<Text, NullWritable, Text, NullWritable> {

    @Override
    protected void cleanup(Reducer<Text, NullWritable, Text, NullWritable>.Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.cleanup(context);
    }

    @Override
    protected void reduce(Text key, Iterable<NullWritable> values,
        Reducer<Text, NullWritable, Text, NullWritable>.Context context) throws IOException,
    InterruptedException {
        context.write(key, NullWritable.get());
    }

    @Override
    public void run(Reducer<Text, NullWritable, Text, NullWritable>.Context arg0)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.run(arg0);
    }

    @Override
    protected void setup(Reducer<Text, NullWritable, Text, NullWritable>.Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
    }
}
```

```
super.setup(context);

}

}
```

### **InnerJoin**

```
package InnerJoin;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;
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 JoinPattern {

    public static void main(String[] args){

        try {
            Job job = Job.getInstance();
            job.setJarByClass(JoinPattern.class);

            job.getConfiguration().set("type",args[3]);

            MultipleInputs.addInputPath(job, new Path(args[0]),TextInputFormat.class,
DistinctMapper.class);
        }
    }
}
```

```
MultipleInputs.addInputPath(job,new Path(args[1]), TextInputFormat.class, CSVMapper.class);

//Reducer
job.setReducerClass(JoinReducer.class);
job.setOutputFormatClass(TextOutputFormat.class);

//Number of Reducers
job.setNumReduceTasks(1);

//Specify Key value
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);

//Specify Output Path
Path output = new Path(args[2]);
FileOutputFormat.setOutputPath(job, output);

FileSystem fs = FileSystem.get(job.getConfiguration());
if(fs.exists(output)) {
    fs.delete(output, true);
}

System.exit(job.waitForCompletion(true) ? 0 : 1);

}catch(Exception ex){
    ex.printStackTrace();
}
}
```

```
package InnerJoin;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class CSVMapper extends Mapper<LongWritable, Text, Text, Text> {

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String[] fields = value.toString().split(",");
        Text t1 = new Text(fields[0]);
        String val = "M"+fields[1]+","+fields[2]+","+fields[3];
        Text t2 = new Text(val);

        context.write(t1,t2);
    }
}

package InnerJoin;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
```

```
import java.io.IOException;

public class DistinctMapper extends Mapper<LongWritable, Text, Text, Text> {

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String[] fields = value.toString().split(" ");
        Text t1 = new Text(fields[0].replace("\n", "").trim());
        String val = "R" + fields[0].replace("\n", "").trim();
        Text t2 = new Text(val);
        context.write(t1, t2);
    }
}

package InnerJoin;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class DistinctMapper extends Mapper<LongWritable, Text, Text, Text> {

    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

```

```

String[] fields = value.toString().split(" ");
Text t1 = new Text(fields[0].replace("\n", "").trim());
String val = "R"+fields[0].replace("\n", "").trim();
Text t2 = new Text(val);
context.write(t1,t2);
}
}

```

### InvertedIndex

```

package InvertedIndex;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
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;

import java.io.IOException;

public class Driver {
    public static void main( String[] args ) throws IOException, ClassNotFoundException, InterruptedException
    {
        Job job = Job.getInstance();

```

```
job.setJarByClass(Driver.class);

FileInputFormat.addInputPath(job, new Path(args[0]));
Path outDir = new Path(args[1]+"_1");
FileOutputFormat.setOutputPath(job, outDir);

job.setMapperClass(FilterMapper.class);
// job.setReducerClass(MyReducer.class);
// job.setCombinerClass(MyReducer.class);

job.setOutputFormatClass(TextOutputFormat.class);
job.setMapOutputValueClass(NullWritable.class);
job.setMapOutputKeyClass(Text.class);
job.setInputFormatClass(TextInputFormat.class);

// job.setNumReduceTasks(1);

Boolean a=job.waitForCompletion(true);

if(a==true)
{
    Job job2 = Job.getInstance();

    job2.setJarByClass(Driver.class);

    // job2.setGroupingComparatorClass(GroupComparator.class);
    // job2.setSortComparatorClass(SecodarySortComparator.class);
    //job2.setPartitionerClass(KeyPartition.class);
```

```
    FileInputFormat.addInputPath(job2, new Path(args[1]+"_1"));

    Path outDir1 = new Path(args[1]);
    FileOutputFormat.setOutputPath(job2, outDir1);

    job2.setMapperClass(MyMapper.class);
    job2.setReducerClass(MyReducer.class);
    job2.setCombinerClass(MyReducer.class);

    job2.setOutputFormatClass(TextOutputFormat.class);
    job2.setMapOutputValueClass(Text.class);
    job2.setMapOutputKeyClass(Text.class);
    job2.setInputFormatClass(TextInputFormat.class);

    job2.setNumReduceTasks(1);

    // job2.setOutputKeyClass(CompositeKeyWritable.class);
    // job2.setOutputValueClass(NullWritable.class);

    job2.waitForCompletion(true);
}

}

}

package InvertedIndex;

import com.google.common.base.Charsets;
import com.google.common.hash.BloomFilter;
```

```
import com.google.common.hash.Funnel;
import com.google.common.hash.Sink;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;
import java.util.ArrayList;

public class BloomMapper extends Mapper<LongWritable, Text, Text, NullWritable> {

    private BloomFilter<String> origin;

    Funnel<String> p = new Funnel<String>() {

        public void funnel(String from, Sink into) {

            into.putString(from,Charsets.UTF_8);

        }
    };

    @Override
    protected void setup(Context context) throws IOException, InterruptedException {

        this.origin = BloomFilter.create(p, 300000, 0.1);

        String p1 = "MA";
    }
}
```

```
String p2 = "NY";
// Person p2 = new Person("Jamie", "Scott");

ArrayList<String> originList = new ArrayList<String>();
originList.add(p1);
originList.add(p2);

for (String ps : originList) {
    origin.put(ps);
}

}

@Override
protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
    try {
        String input = value.toString();

        String splitString[] = input.split(",");
        String origins = splitString[19];
        //System.out.println(origins);
        //String values[] = value.toString().split(",");
        //Person p = new Person(values[1], values[2]);

        if (origin.mightContain(origins) && !splitString[0].contains("id")) {
            context.write(value, NullWritable.get());
        }
    }
}
```

```
        } catch(Exception e) {
            System.out.println(e);
        }
    }

@Override
protected void cleanup(Context context) throws IOException, InterruptedException
{
    super.cleanup(context);
}

package InvertedIndex;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MyMapper extends Mapper<LongWritable, Text, Text, Text> {
    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        try {
            String input = value.toString();

            String splitString[] = input.split(",");
            String origin = splitString[24];
        }
    }
}
```

```

        context.write(new Text(origin), new Text(splitString[0]));

    } catch(Exception e) {
        System.out.println(e);
    }
}

}

package InvertedIndex;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class MyReducer extends Reducer<Text, Text, Text, Text> {

    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
    InterruptedException {

        String concatString = "";

        for(Text s:values) {
            if(!concatString.contains(s.toString())) {
                concatString = s.toString() + "," + concatString;
            }
        }

        context.write(key,new Text(concatString));
    }
}

```

```
 }  
 }
```

### **MaxAndMinIntRateForEmpLength**

```
package MaxAndMinIntRateForEmpLength;  
  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
import org.apache.hadoop.mapreduce.lib.input.NLineInputFormat;  
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  
import java.io.IOException;  
  
public class MaxMinRateDriver {  
    public static void main( String[] args ) throws IOException, ClassNotFoundException, InterruptedException  
    {  
        Job job = Job.getInstance();  
  
        job.setJarByClass(MaxMinRateDriver.class);  
  
        FileInputFormat.addInputPath(job, new Path(args[0]));  
        Path outDir = new Path(args[1]);  
        FileOutputFormat.setOutputPath(job, outDir);
```

```
job.setMapperClass(MaxMinRateMapper.class);
job.setReducerClass(MaxMinRateReducer.class);
job.setCombinerClass(MaxMinRateReducer.class);

job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setInputFormatClass(NLineInputFormat.class);
NLineInputFormat.setNumLinesPerSplit(job, 10000);
job.setNumReduceTasks(1);

job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);

job.waitForCompletion(true);
}

}

package MaxAndMinIntRateForEmpLength;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MaxMinRateMapper extends Mapper<LongWritable,Text,Text,Text> {
```

```
@Override  
protected void map(LongWritable key, Text value, Context context) throws IOException,  
InterruptedException {  
    Text symbol;  
    String line = value.toString();  
    String[] fields = line.split(",");  
    if (fields.length > 5) {  
        if (!line.contains("int_rate")) {  
            symbol = new Text(fields[10]);  
            context.write(symbol, new Text(fields[6] + " " + fields[6]));  
        }  
    }  
}  
}
```

```
package MaxAndMinIntRateForEmpLength;  
  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Reducer;  
  
import java.io.IOException;  
  
public class MaxMinRateReducer extends Reducer<Text,Text,Text,Text> {  
    @Override  
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException,  
InterruptedException {  
    Float currentMaxRateValue;  
    Float currentMinRateValue;
```

```

Float maxRateValue = (float) 0;
Float minRateValue = (float) Integer.MAX_VALUE;

for (Text value : values) {
    String a[] = value.toString().split(" ");
    currentMaxRateValue = Float.parseFloat(a[1].trim().replace("%", ""));
    currentMinRateValue = Float.parseFloat(a[0].trim().replace("%", ""));

    if (currentMaxRateValue > maxRateValue) {
        maxRateValue = currentMaxRateValue;
    }
    if (currentMinRateValue < minRateValue) {
        minRateValue = currentMinRateValue;
    }
}

// String result = "Min Rate: " + String.valueOf(minRateValue) + " Max Rate: "
//      + String.valueOf(maxRateValue);
context.write(key, new Text(minRateValue + "" + maxRateValue));
}
}

```

## Median

```

package Median;

import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;

```

```
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;

import java.io.IOException;

public class Driver {
    public static void main( String[] args ) throws IOException, ClassNotFoundException, InterruptedException
    {
        Job job = Job.getInstance();

        job.setJarByClass(Driver.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        Path outDir = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outDir);

        job.setMapperClass(FilterMapper.class);
        job.setReducerClass(FilterReducer.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);

        job.setNumReduceTasks(1);
```

```

        FileInputStream.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        FileSystem fs = FileSystem.get(job.getConfiguration());
        job.waitForCompletion(true);
    }

}

package Median;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class FilterMapper extends Mapper<LongWritable, Text, Text, Text> {
    @Override
    protected void map(LongWritable key, Text value, Context context) throws IOException,
    InterruptedException {
        try {
            String split[] = value.toString().split(",");
            // IntWritable one = new IntWritable(1);
            String symbol = split[0];
            if(!symbol.contains("member_id") && split.length > 20 && !split[5].equalsIgnoreCase("0") &&
            !split[20].equalsIgnoreCase("") && !split[20].equalsIgnoreCase("")) {
                context.write(new Text(split[8]), new Text(split[24]));
            }
        }
    }
}

```

```
        }

    } catch(Exception ex) {
        System.out.println("Exception in mapper"+ex);
    }
}

package Median;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;

public class FilterReducer extends Reducer<Text, Text, Text,Text> {

    ArrayList<Integer> array= new ArrayList<Integer>();
    @Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {
        Integer median=0;
        //int sum = 0;
        //int count = 0;

        //Integer temp= (int) Float.parseFloat(key.toString());
        try {
            for (Text value:values) {
```

```

        Integer temp = (int) Float.parseFloat(value.toString());
        if (temp != 0){
            array.add(temp);
        }
        //context.write(new Text(key), NullWritable.get());
    }

    Collections.sort(array);
    System.out.println(array.size()+":"+array);
    int len=array.size();
    if(array.size()%2==0){
        median= (array.get((int) len/2 - 1)+array.get((int) len/2)) /2;
    }
    else{
        median= array.get(len/2);
    }

    context.write(key,new Text(median+""));

}
catch (Exception e){
    System.out.println(key+e.getMessage());
}
}
}

```

## **PutMerge**

```

package PutMerge;

import org.apache.hadoop.conf.Configuration;

```

```

import org.apache.hadoop.fs.*;import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class MergeFiles {

    public static void main(String[] args) throws IOException, ClassNotFoundException,
InterruptedException { //Hadoop File System instance

        Configuration conf = new Configuration();

        FileSystem hdfs = FileSystem.get(conf); //local File System instance

        FileSystem local = FileSystem.getLocal(conf); //Path to the directory in local file System

        Path inputDir = new Path(args[0]);

        Path hdfsFile = new Path(args[1]); try {

            //List of Files Names from FileStatus

            FileStatus[] inputFiles = local.listStatus(inputDir);

            //Writing to hdfs using OutputStream

            FSDataOutputStream out = hdfs.create(hdfsFile); //Reading the input files using

            FSDataInputStream

            for (int i = 0; i < inputFiles.length; i++) {

                FSDataInputStream in = local.open(inputFiles[i].getPath());

                // BufferedReader br = new BufferedReader(new InputStreamReader(in, UTF8));

                //int lineCount = 0;

                byte buffer[] = new byte[256];

                int bytesRead = 0; while ((bytesRead = in.read(buffer)) > 0 ) {

                out.write(buffer, 0, bytesRead);

            }

            in.close();

        }

        out.close(); } catch (IOException e) {

        e.printStackTrace();

    }

}

```

```
 }  
 }
```

### **Secondary Sort**

```
package SecSort;  
  
import org.apache.hadoop.fs.FileSystem;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
  
import java.io.IOException;  
  
public class Driver  
{  
    public static void main( String[] args ) throws IOException, ClassNotFoundException,  
    InterruptedException  
    {  
        Job job = Job.getInstance();  
  
        job.setJarByClass(Driver.class);  
  
        job.setGroupingComparatorClass(GroupComparator.class);  
        job.setSortComparatorClass(SecodarySortComparator.class);  
        job.setPartitionerClass(KeyPartition.class);  
  
        FileInputFormat.addInputPath(job, new Path(args[0]));
```

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

job.setMapperClass(MyMapper.class);
job.setReducerClass(MyReducer.class);

job.setNumReduceTasks(1);

job.setOutputKeyClass(CompositeKeyWritable.class);
job.setOutputValueClass(NullWritable.class);

FileSystem fs = FileSystem.get(job.getConfiguration());
if(fs.exists(outDir)) {
    fs.delete(outDir, true);
}

job.waitForCompletion(true);

// }

}

package SecSort;

import org.apache.hadoop.io.WritableComparable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
```

```
public class CompositeKeyWritable implements WritableComparable<CompositeKeyWritable> {

    private String grade;
    private String subgrade;

    public CompositeKeyWritable() {
        super();
    }

    public CompositeKeyWritable(String grade, String id) {
        super();
        this.grade = grade;
        this.subgrade = id;
    }

    public void write(DataOutput out) throws IOException {
        out.writeUTF(grade);
        out.writeUTF(subgrade);
    }

    public void readFields(DataInput in) throws IOException {
        grade = in.readUTF();
        subgrade = in.readUTF();
    }

    public int compareTo(CompositeKeyWritable o) {
```

```
int result = this.grade.compareTo(o.grade);
if (result == 0) {
    return this.subgrade.compareTo(o.subgrade);
}
return result;
}

public String getGrade() {
    return grade;
}

public void setGrade(String grade) {
    this.grade = grade;
}

public String getSubgrade() {
    return subgrade;
}

public void setSubgrade(String subgrade) {
    this.subgrade = subgrade;
}

@Override
public String toString() {

    return grade + "," + subgrade;
}
```

```
}
```

```
package SecSort;
```

```
import org.apache.hadoop.io.WritableComparator;
```

```
public class GroupComparator extends WritableComparator {
```

```
    protected GroupComparator() {
```

```
        super(CompositeKeyWritable.class, true);
```

```
    }
```

```
    @Override
```

```
    public int compare(Object a, Object b) {
```

```
        CompositeKeyWritable ckw1 = (CompositeKeyWritable)a;
```

```
        CompositeKeyWritable ckw2 = (CompositeKeyWritable)b;
```

```
        return ckw1.getGrade().compareTo(ckw2.getGrade());
```

```
    }
```

```
}
```

```
package SecSort;
```

```
import org.apache.hadoop.io.NullWritable;
```

```
import org.apache.hadoop.mapreduce.Partitioner;
```

```
public class KeyPartition extends Partitioner<CompositeKeyWritable, NullWritable> {
```

```
    @Override
    public int getPartition(CompositeKeyWritable key, NullWritable value, int numPartitions) {
        return key.getGrade().hashCode()%numPartitions;
    }
}

package SecSort;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MyMapper extends Mapper<LongWritable, Text, CompositeKeyWritable, NullWritable> {

    @Override
    protected void map(LongWritable key, Text value,
                      Context context)
            throws IOException, InterruptedException {
        String line = value.toString();
        String[] tokens = line.split(",");
        String grade = null;
```

```
String subbgrade = null;  
if(!value.toString().contains("member_id") && tokens.length>14) {  
    try {  
  
        grade = tokens[8];  
        subbgrade = tokens[9];  
  
    } catch (Exception e) {  
  
    }  
  
    if (grade != null && subbgrade != null) {  
  
        CompositeKeyWritable outKey = new CompositeKeyWritable(grade, subbgrade);  
  
        context.write(outKey, NullWritable.get());  
  
    }  
}  
  
}  
  
package SecSort;  
  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.mapreduce.Reducer;  
  
import java.io.IOException;
```

```
public class MyReducer extends Reducer<CompositeKeyWritable, NullWritable, CompositeKeyWritable, NullWritable> {

    @Override
    protected void reduce(CompositeKeyWritable key, Iterable<NullWritable> values,
        Context context)
        throws IOException, InterruptedException {
        //for (NullWritable v : values) {
        //    context.write(key, NullWritable.get());
        //}
    }

}

package SecSort;

import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableComparator;

public class SecodarySortComparator extends WritableComparator {

    protected SecodarySortComparator() {
        super(CompositeKeyWritable.class, true);
    }

    @Override
```

```
public int compare(WritableComparable a, WritableComparable b) {  
  
    CompositeKeyWritable ckw1 = (CompositeKeyWritable) a;  
    CompositeKeyWritable ckw2 = (CompositeKeyWritable) b;  
  
    int result = ckw1.getGrade().compareTo(ckw2.getGrade());  
  
    if (result == 0) {  
        return ckw1.getSubgrade().compareTo(ckw2.getSubgrade());  
    }  
  
    return result;  
}  
  
}
```

### **ShuffleMemberId**

```
package ShuffleMemberId;  
  
import org.apache.hadoop.fs.FileSystem;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Job;  
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;

import java.io.IOException;

public class ShuffleDriver {
    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Job job = Job.getInstance();
        job.setJarByClass(ShuffleDriver.class);

        job.setMapperClass(ShuffleMapper.class);
        job.setReducerClass(ShuffleReducer.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        job.setOutputKeyClass(IntWritable.class);
        job.setOutputValueClass(Text.class);

        job.setNumReduceTasks(1);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        FileSystem fs = FileSystem.get(job.getConfiguration());
        job.waitForCompletion(true);
```

```
    }

}

package ShuffleMemberId;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;
import java.util.Random;

public class ShuffleMapper extends Mapper<LongWritable, Text, IntWritable, Text> {

    private static final IntWritable one = new IntWritable(1);
    private Random random= new Random();
    @Override
    protected void map(LongWritable key, Text value, Context context)
        throws IOException, InterruptedException {
        String line = value.toString();

        String[] tokens = line.split(",");
        if(tokens.length > 15) {
            context.write(new IntWritable(random.nextInt()), value);
        }
    }
}
```

```
    @Override  
    protected void cleanup(Context context)  
        throws IOException, InterruptedException {  
        // TODO Auto-generated method stub  
        super.cleanup(context);  
    }
```

```
    @Override  
    public void run(Context context)  
        throws IOException, InterruptedException {  
        // TODO Auto-generated method stub  
        super.run(context);  
    }
```

```
    @Override  
    protected void setup(Context context)  
        throws IOException, InterruptedException {  
        // TODO Auto-generated method stub  
        super.setup(context);  
    }
```

```
}
```

```
package ShuffleMemberId;
```

```
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;
```

```
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

public class ShuffleReducer extends Reducer<IntWritable, Text, Text, NullWritable> {

    @Override
    protected void cleanup(Context context)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.cleanup(context);
    }

    @Override
    protected void reduce(IntWritable key, Iterable<Text> values,
        Context context) throws IOException, InterruptedException {
        for (Text val: values) {
            // context.write(new IntWritable(),val);
            context.write(val,NullWritable.get());
        }
    }

    @Override
    public void run(Context arg0)
        throws IOException, InterruptedException {
        // TODO Auto-generated method stub
        super.run(arg0);
    }
}
```

```
    @Override  
    protected void setup(Context context)  
        throws IOException, InterruptedException {  
        // TODO Auto-generated method stub  
        super.setup(context);  
    }  
  
}
```

### **Top10MembersLoanAmt**

```
package Top10MembersLoanAmt;  
  
import org.apache.hadoop.fs.FileSystem;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Job;  
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;  
  
import java.io.IOException;  
  
/**  
 * Hello world!  
 *  
 */
```

```
public class App
{
    public static void main( String[] args ) throws IOException, ClassNotFoundException,
InterruptedException
    {
        Job job = Job.getInstance();
        job.setJarByClass(App.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        Path input = new Path(args[0]);
        Path output = new Path(args[1]);

        job.setOutputKeyClass(NullWritable.class);
        job.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(job, input);
        FileOutputFormat.setOutputPath(job, output);

        job.setNumReduceTasks(1);

        job.setMapperClass(TopKMapper.class);
        job.setReducerClass(TopKReducer.class);

        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(output)) {
            fs.delete(output, true);
        }
    }
}
```

```
        job.waitForCompletion(true);

    }

}

package Top10MembersLoanAmt;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;
import java.util.TreeMap;

public class TopKMapper extends Mapper<LongWritable, Text, NullWritable, Text> {

    private static final int K = 10;
    private TreeMap<Integer, Text> tMap;

    @Override
    protected void setup(Mapper<LongWritable, Text, NullWritable, Text>.Context context)
            throws IOException, InterruptedException {
        this.tMap = new TreeMap<Integer, Text>();
    }

    @Override
    protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, NullWritable,
Text>.Context context)
```

```
        throws IOException, InterruptedException {

    String[] tokens = value.toString().split(",");
    if(!value.toString().contains("id") && tokens.length>8) {
        try {

            String score = tokens[7].replace("\\"", "").trim();
            tMap.put(Integer.parseInt(score), new Text(value));

        } catch (Exception ex) {

    }

    if (tMap.size() > K) {
        tMap.remove(tMap.firstKey());
    }
}

@Override
protected void cleanup(Mapper<LongWritable, Text, NullWritable, Text>.Context context)
        throws IOException, InterruptedException {

    for (Text t : tMap.values()) {
        context.write(NullWritable.get(), t);
    }

}
```

```
}
```

```
package Top10MembersLoanAmt;
```

```
import org.apache.hadoop.io.NullWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Reducer;
```

```
import java.io.IOException;  
import java.util.TreeMap;
```

```
public class TopKReducer extends Reducer<NullWritable, Text, NullWritable, Text> {
```

```
    private static final int K = 10;  
    private TreeMap<Integer, Text> tMap;
```

```
    @Override  
    protected void setup(Reducer<NullWritable, Text, NullWritable, Text>.Context context)  
        throws IOException, InterruptedException {
```

```
        this.tMap = new TreeMap<Integer, Text>();
```

```
}
```

```
    @Override  
    protected void reduce(NullWritable arg0, Iterable<Text> values,  
        Reducer<NullWritable, Text, NullWritable, Text>.Context arg2) throws IOException,  
        InterruptedException {
```

```

        for (Text t : values) {

            String[] tokens = t.toString().split(",");
            tMap.put(Integer.parseInt(tokens[7]), new Text(t));

            if (tMap.size() > K) {
                tMap.remove(tMap.firstKey());
            }
        }

    }

@Override
protected void cleanup(Reducer<NullWritable, Text, NullWritable, Text>.Context context)
    throws IOException, InterruptedException {

    for(Text t : tMap.values()) {
        context.write(NullWritable.get(), t);
    }
}

}

```

### **VerificationStatusMedianStd**

```

package VerificationStatusMedianStd;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.SortedMapWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;

```

```
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 java.io.IOException;

public class Driver {
    public static void main( String[] args ) throws IOException, ClassNotFoundException, InterruptedException
    {
        Job job = Job.getInstance();

        job.setJarByClass(Driver.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        Path outDir = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outDir);

        job.setMapperClass(MyMapper.class);
        job.setReducerClass(MyReducer.class);
        job.setCombinerClass(myCombiner.class);

        //job.setOutputFormatClass(TextOutputFormat.class);
        job.setMapOutputValueClass(SortedMapWritable.class);
        job.setMapOutputKeyClass(Text.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(MedianStdDevTuple.class);
    }
}
```

```
        job.setNumReduceTasks(1);

        job.waitForCompletion(true);
    }

}

package VerificationStatusMedianStd;

import org.apache.hadoop.io.Writable;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

public class MedianStdDevTuple implements Writable {

    private float median;

    private float stdDev;

    public MedianStdDevTuple(float median, float stdDev) {
        this.median = median;
        this.stdDev = stdDev;
    }

    public float getMedian() {
        return median;
    }

    public void setMedian(float median) {
        this.median = median;
    }
}
```

```
}

public float getStdDev() {
    return stdDev;
}

public void setStdDev(float stdDev) {
    this.stdDev = stdDev;
}

public void write(DataOutput out) throws IOException {
    out.writeFloat(this.median);
    out.writeFloat(this.stdDev);
}

@Override
public String toString() {

    return "median:"+ this.median +" Std_Div:"+this.stdDev;
}

public void readFields(DataInput in) throws IOException {
    this.median=in.readFloat();
    this.stdDev=in.readFloat();
}

public MedianStdDevTuple() {
}
```

```
}
```

```
package VerificationStatusMedianStd;

import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
import java.util.Map;

public class myCombiner extends Reducer<Text, SortedMapWritable, Text, SortedMapWritable> {

    @Override
    protected void reduce(Text key, Iterable<SortedMapWritable> values, Context context) throws
    IOException, InterruptedException {
        // int sum=0;
        SortedMapWritable map = new SortedMapWritable();
        for(SortedMapWritable v:values) {
            for(Object entry : v.entrySet() ) {
                if(entry instanceof Map.Entry) {
                    Map.Entry entry1 = (Map.Entry) entry;
                    IntWritable count = (IntWritable) map.get(entry1.getKey());
                    FloatWritable mapKey = ((FloatWritable) entry1.getKey());
                    Integer mapValue = ((IntWritable) entry1.getValue()).get();
                    if (count != null) {
                        map.put(mapKey, new IntWritable(count.get() + ((IntWritable) entry1.getValue()).get()));
                    } else {
                        map.put(mapKey, new IntWritable(((IntWritable) entry1.getValue()).get()));
                    }
                }
            }
        }
    }
}
```

```

        }

        v.clear();

    }

    context.write(key,map);

}

}

package VerificationStatusMedianStd;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

public class MyMapper extends Mapper<LongWritable, Text, Text, SortedMapWritable> {

    @Override

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

        try {

            String split[]=value.toString().split(",");
            IntWritable one = new IntWritable(1);

            String symbol = split[13];

            if(!split[0].contains("id") && split.length>6 ) {

                Float avg = Float.parseFloat(split[2]);

                SortedMapWritable map = new SortedMapWritable();
                map.put(new FloatWritable(avg),one);
                context.write(new Text(symbol.replace("\n","")),map);

            }

        }
    }
}

```

```

} catch(Exception ex) {
    System.out.println("Exception in mapper"+ex);
}
}

}

package VerificationStatusMedianStd;

import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
import java.util.Map;
import java.util.TreeMap;

public class MyReducer extends Reducer<Text, SortedMapWritable, Text, MedianStdDevTuple> {
    @Override
    protected void reduce(Text key, Iterable<SortedMapWritable> values, Context context) throws
    IOException, InterruptedException {
        //int sum = 0;
        //int count = 0;
        MedianStdDevTuple result = new MedianStdDevTuple();
        TreeMap<Float, Integer> finalMap= new TreeMap<Float, Integer>();

        try{
            Float sum = 0f;
            Integer totalCount = 0;

```

```
for(SortedMapWritable v:values) {  
    for(Object entry: v.entrySet() ) {  
        if(entry instanceof Map.Entry) {  
            Map.Entry entry1 = (Map.Entry) entry;  
            Float mapKey = ((FloatWritable) entry1.getKey()).get();  
            Integer mapValue = ((IntWritable) entry1.getValue()).get();  
            Integer count = finalMap.get(mapKey);  
  
            sum = sum + mapValue * mapKey;  
            totalCount = totalCount + mapValue;  
  
            if (count != null) {  
                finalMap.put(mapKey, mapValue + count);  
            } else {  
                finalMap.put(mapKey, mapValue);  
            }  
        }  
    }  
    v.clear();  
}  
Integer middleIndex = totalCount/2;  
int prev=0;  
int current = 0;  
Float prevKey = 0f;  
Float median=0f;
```

```

for(Map.Entry<Float,Integer> entry: finalMap.entrySet()) {
    current = entry.getValue();
    if(middleIndex > prev && middleIndex<=(current+prev)) {
        if(totalCount%2==0) {
            median = (entry.getKey()+prevKey)/2;
        } else {
            median = entry.getKey();
        }
        break;
    }
    prev= current+prev;
    prevKey = entry.getKey();
}

result.setMedian(median);

Float mean = sum/totalCount;

Float sumOfSquares=0.f;
for(Map.Entry<Float,Integer> entry:finalMap.entrySet()) {
    sumOfSquares = sumOfSquares + (entry.getKey()-mean)*(entry.getKey()-mean);
}
Float stdDev = (float) Math.sqrt(sumOfSquares/(totalCount-1));

result.setStdDev(stdDev);

context.write(key,result);

```

```
    } catch(Exception e) {  
        System.out.println("exception in reducer "+ e);  
    }  
}
```

Pig:

## Filtering:

```
users = LOAD 'LoanStats3a_securev1.csv' using PigStorage(',');
s1 = FILTER users BY $2 < 3000;
s2 = FOREACH s1 GENERATE $0,$1,$2,$3,$4,$5,$6,$7,$8,$9,$10,$11,$12,$13,$14,'T';
STORE s2 into 'SmallLoanAmount';
```

## Joins:

```
grade = LOAD 'grade' using PigStorage(',') as (grd);  
  
detail = LOAD 'gradedetails.csv' using PigStorage(',') as (id,desc,name,creditscore);  
  
joinop = JOIN grade BY grd, detail BY id;  
  
out = FOREACH joinop GENERATE $0,$2,$3,$4;  
  
STORE summed out 'gradeoutput';
```

## Top K:

```
users = LOAD 'LoanStats3a_securev1.csv' using PigStorage(',');
grouped = GROUP users BY $24;
summed = FOREACH grouped GENERATE group, COUNT(users) AS cntd;
sorted = ORDER summed BY cntd DESC;
top25 = LIMIT sorted 25;
STORE top25 into 'topk_ficoscore_pig.csv';
```