I first worked on cleaning the data obtained for dblp by converting the data to having one xml input per line. This allowed reading and parsing to the reducer to be substantially faster and more simple to understand. The new data takes approximately a minute to create how ever this allows us to not use the built in xml parser that scala offers as using this will increase run time by 25+ minutes.

The first step in getting ready to map and reduce was to install hortonworks and vmware as this allowed us to set up the environment necessary much more quickly and more simple than installing Hadoop manually would be. With that installed we can begin coding the reader we will use for the class. This was a simple reader that just takes one line at a time and checks if it contains the word author. If it does the string is parsed and sent to the mapper. If it does not contain the word Author, the line is skipped, and we check the next line.

Now at the mapper I split the passed in value into different sections such as type of input (book www etc) ,an array of authors, and finally the year of current entry. These are then written to mapper to be sent to the reducer. I did have to give the histograms a strange key in order to print them towards the top of the file as the final output is determined by the key order sent to the mapper not the reducer. The order is determined by ascii so histograms are give keys such as “A ,”B , “C ,”D to ensure they will print in the order required to have a good output.

Now at the reducer each author gets reduced with various functions provided In the reducer class such as updatex() where x is the value stored in the class. This made reducing a simple foreach on the iterable provided by the reducer for each key. How ever if a histogram key was entered this will cause a different form of reducing to happen where it updates the bins stored in the words section of the author class for each histogram. After the reducer is finished we override the cleanup method and add an if statement that checks the progress of the job. If the progress of the job is 99% we then write our top 100 and bottom 100 to the reducer to be printed out as we finally have all the input required for that. We check the progress in cleanup because the reducer gets call multiple times to support pipelining and if we did not check the progress to be basically finished then in cleanup we will write multiple times to the reducer causing a bad output.

Finally in the print section I first handle printing all of the histograms then I head to the Authors. I feel like this is the best format to have as printing all those authors first really is a waste of time to scroll through them and the important data is at the top. Sadly no matter how hard I try the top and bottom authors refuse to print to the top so they are left at the very bottom of the output

This was all possible because of extending the class writable and creating a class that can store text,int,and a double this allowed for transferring almost any data in a format you can parse later in the reducer and in the printing section.

My program runs pretty fast at about 3 minutes to process the entire 2.5gb file and to display all the information required. These speeds were possible because of only using 1 job to get all data required instead of multiple. Also the custom parser played a role in it as well.

The histograms working in 1 job was difficult to manage at first at it used a lot of memory inside of yarn but after creating and adding bins into text instead of storing all the years and counts for the entire file it became much more efficient and i didn’t need to run another task. I also added commas to support csv files better as this will but things in new elements

Output from entire file est Time for all data to finish is ~4 minutes(1 minute for the java parser to set up the data, 3 minutes for actual map and reduce (that's pretty fast if you ask me made testing easier))

```
Year Count histogram
1930:,57
1940:,155
1950:,2333
1960:,16183
1970:,62425
1980:,220505
1990:,936816
2000:,3516844
2010:,7957877
2020:,2822
```

```
Journal Author Count histogram
0:,541783
2:,1045215
4:,342868
6:,70962
8:,17388
10:,5637
12:,2439
14:,1243
16:,668
18:,435
20:,272
22:,207
24:,143
28:,73
32:,44
34:,34
38:,18
40:,2
42:,1
44:,6
46:,8
48:,9
54:,4
60:,2
64:,1
70:,1
72:,1
78:,1
```

Conference Author Count histogram
0:,393052
2:,1332182
4:,556334
6:,117882
8:,25995
10:,7580
12:,2942
14:,1281
16:,719
18:,396
20:,239
22:,147
24:,117
26:,68
28:,55
30:,41
32:,13
34:,13
36:,13
38:,6
40:,7
42:,10
46:,6
52:,3
54:,1
56:,3
60:,3
64:,2

Author Count histogram
0:,3368053
2:,2454259
4:,905100
6:,190017
8:,43705
10:,13345
12:,5430
14:,2559
16:,1398
18:,842
20:,514
22:,355
24:,262
26:,165
28:,131
30:,97
32:,66
34:,52
36:,49
38:,29
40:,28
42:,26
44:,12
46:,10
50:,2
54:,1
60:,5
62:,3
64:,3
68:,1
70:,4
76:,3
78:,1
80:,2
94:,1
102:,2
110:,1
252:,1

```
all,Max: 268,Mean: 2.168885062665923
article,Max: 268,Mean: 2.608120658135908
book,Max: 18,Mean: 1.6071411191969505
incollection,Max: 48,Mean: 2.1852199654733324
inproceedings,Max: 153,Mean: 2.958440608533185
mastersthesis,Max: 1,Mean: 1.0
phdthesis,Max: 3,Mean: 1.0046808630764246
proceedings,Max: 1,Mean: 1.0
www,Max: 10,Mean: 1.0230444349437549
```

Then from here all the authors scores are printed for the entire file

An example would be below

```
Author,Authorship,numberOfPublications,MaxCoauthors,MedianCoauthors,MinCoauthors
&AElig;leen Frisch,5.041666666666666,7,4,1,1.8571428571428572
Á. B. Nagy,1.625,2,2,2,1.5
Á. Baran,2.875,5,3,2,2.0
Á. Birkisson,1.3333333333333333,2,3,3,2.0
Á. F&ouml;ldváry,1.3333333333333333,2,3,3,2.0
Á. González,1.3333333333333333,2,3,3,2.0
Á. Hernández,1.1875,2,4,4,2.5
Á. Irabien,1.0,1,1,1,1.0
Á. Kovács,1.625,2,2,2,1.5
Á. Kriston,1.25,2,3,3,2.0
Á. L. Gallego,1.3125,2,4,4,2.5
Á. Makay,2.625,4,3,2,1.75
Á. Michels,1.4166666666666665,2,3,3,2.0
Á. Nunes,2.0,2,1,1,1.0
```

And then sadly at the bottom of the file

```
Bottom 100 ascending:,Bruce &quot;Tog&quot; Tognazzini:0.09090909090909091,T
cute;ndez-Pampill&oacute;n:0.325,Jen Dong:0.32999999999999996,Vasilis Stefan
45454545453,Todd N. Wylie:0.3922413793103448,Tolga Nazyok:0.3928571428571428
nk Revercomb:0.4076086956521739,Tessy Cerratto Pargman:0.4083333333333333,To
Top 100 descending,H. Vincent Poor:456.07289377289385,Ronald R. Yager:448.02
dier Dubois:272.1977018079959,Nadia Magnenat-Thalmann:268.8547324203574,Yan
hang:237.94056520170648,Yan Wang:237.51076146076144,Saharon Shelah:235.65833
iou:218.19642857142856,Xin Liu:217.5619639202261,
```