Problem I: MapReduce implementation using Python and mrjob

You will use two dataset: Two versions of the data are provided, (1)a small trial version with only 1000 integers to use while developing your method, and (2)a test that goes over 1 million integers to test your data on a larger dataset:

trial_incomes.csv
test_incomes.csv.zip

The aim is to calculate total incomes, mean of incomes, highest income, lowest income, and standard deviation of incomes.

1. **total incomes** – The sum of all incomes in the dataset
2. **Mean** – The mean of all incomes in the dataset. The mean is given by the following formula:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Where n is the number of elements in the dataset, and x_i is the variable associated to incomes.

3. **Generalized mean** – The generalized mean of all incomes in the dataset given by the following formula:

$$M_p = \left(\frac{1}{n}\sum_{i=1}^{n} {x_i}^p\right)^p$$

Where n is the number of elements in the dataset, x_i is the variable associated to incomes, and p is the order of the generalized mean.

4. **Maximum** – highest income in the dataset.
5. **Minimum** – lowest income in the dataset.
6. **Standard deviation** – the standard deviation of the incomes using the following formula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n}}$$

For each question, you must write a Python program in MapReduce using mrjob. Which means that you have to provide ***six Python programs.***

Problem II: Sparkifying world cities

In this problem, you will use the dataset `worldcitiespop.txt` where you will have to compute some features and queries using `pyspark`. For that aim, you are restricted to only doing so with at most four shufflable transformation – a transformation which may cause a shuffle: `reduceByKey`, `groupByKey`, `combineByKey`, `sortByKey` `join`, `leftOuterJoin`, `rightOuterJoin`, `intersection`, `cogroup`, `groupWith distinct`, `repartition`, `coals`.

1. **Simple cleaning worldcitiespo**p: Write a Spark python program that cleans the "worldcitiespop.txt" file to keep only valid lines. By valid we only consider rows with a given population.
2. **Statistics:** Write a Spark Python program to display the following statistics for city populations: min, max, sum, and average.
3. **Histograms:** Write a Spark Python program to calculate a frequency histogram of city populations. For the histogram, the equivalence classes will be chosen using a logarithmic scale. (class 0, cities of size [0..10[, class 1 cities of size [10..100[...).
4. **TopK:** Write a Spark Python program to calculate and display the 10 cities with the largest population.
5. Calculate the percentage of the total population represented by the top K populous cities globally.
6. Calculate the total population of each region and find the region with the highest population.
7. Find the countries where the sum of the populations of their cities is greater than a specified value.
8. Determine the average population for cities in each country, considering only cities with a population above a certain threshold.
9. Identify cities with a population above the average population of their respective countries.

10. **Re-cleaning:** By analyzing the results of 4), we see that "Delhi" and "New Delhi" are the same cities. Propose a solution to remove duplicates (different cities in the same place) by not keeping the cities with the highest population in case of overlap. Recalculate your histograms etc... You should get the following result:

```
(count: 37774, mean: 56685.35026737963, stdev: 335272.73059020063, max: 31480498.0,
min: 7.0)
[(0, 4), (1, 65), (2, 1324), (3, 14545), (4, 18456), (5, 3107), (6, 264), (7, 9)]
jp,tokyo,Tokyo,40,31480498,35.685,139.751389
cn,shanghai,Shanghai,23,14608512,31.045556,121.399722
in,bombay,Bombay,16,12692717,18.975,72.825833
pk,karachi,Karachi,05,11627378,24.9056,67.0822
in,delhi,Delhi,07,10928270,28.666667,77.216667
ph,manila,Manila,D9,10443877,14.6042,120.9822
ru,moscow,Moscow,48,10381288,55.752222,37.615556
kr,seoul,Seoul,11,10323448,37.5985,126.9783
br,sao paulo,S□o Paulo,27,10021437,-23.473293,-46.665803
```

```
tr,istanbul,Istanbul,34,9797536,41.018611,28.964722
ng,lagos,Lagos,05,8789133,6.453056,3.395833
mx,mexico,Mexico,09,8720916,19.434167,-99.138611
id,jakarta,Jakarta,04,8540306,-6.174444,106.829444
us,new york,New York,NY,8107916,40.7141667,-74.0063889
cd,kinshasa,Kinshasa,06,7787832,-4.3,15.3
eg,cairo,Cairo,11,7734602,30.05,31.25
pe,lima,Lima,15,7646786,-12.05,-77.05
cn,peking,Peking,22,7480601,39.928889,116.388333
gb,london,London,H9,7421228,51.514125,-.093689

co,bogota,Bogot�,34,7102602,4.649178,-74.062827
```

You can rely on the demonstration code `install_And_Test_Spark.ipynb` that contains different tools of Spark provided in a notebook that can be run on Google Colab. In fact, it contains the installation of Spark, and demos for the following functions defined in Spark: `reduceByKey`, `groupByKey`, `combineByKey`, `sortByKey`, `join`, `leftOuterJoin`, `rightOuterJoin`, `intersection`, `cogroup`, `groupWith`, `distinct`, `repartition`, `coalesce`. You are required to rely on functions in spark to resolve the task of this assignment. You should write your pyspark code in a notebook where the response for each task can be run successfully on Google Colab.

Submission

You must submit the *pdf* file of your report. In fact, it must contain the responses in form of outputs for the two problems. In addition, you must submit all your six Python programs of Problem I, and in a `.ipynb` file of the Problem II that can be run on Google Colab.