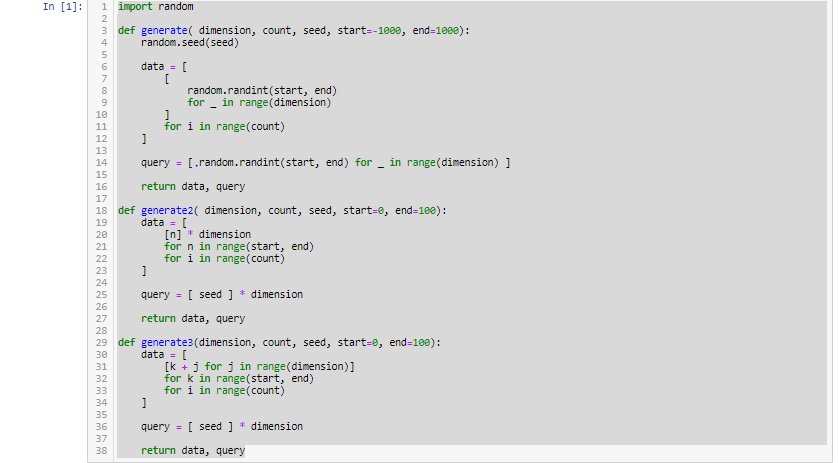
1)Implementation details of your c2lsh(). Explain how your major transform function works.

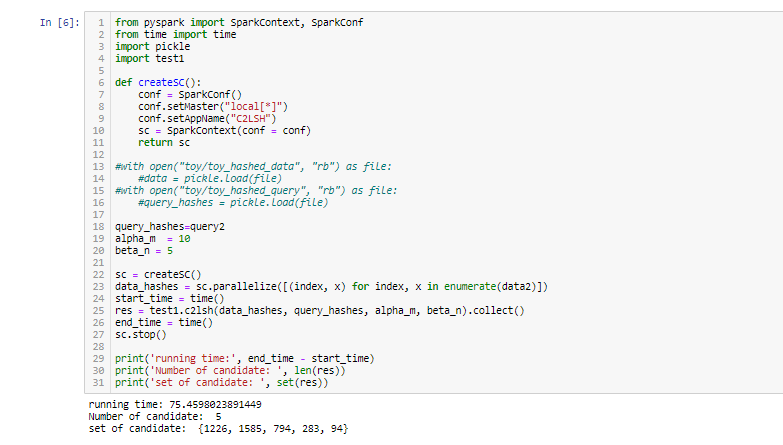
I used only Filter , Count and Keys functions in my final submission.

Step:

1. I am filtering datahashes rdd using Count function (Defined) . Count function returns number of collisions between 2 lists and sends the number . Filter checks whether the number is greater than or equal to Alpha\_m . If “YES” it will be in rdd or else omitted .
2. Now counting total number of Candidates in RDD using rdd.count() function if rdd.count()< beat\_n then we are incrementing offset value by 1 and again doing step until we got enough nearest neighbours.
3. If we got enough Neighbours return rdd.keys()

2) Show the evaluation result of your implementation using **your own test cases**.

data2, query2 = generate(20, 2000, 0, 0, 100)



3)What did you do to improve the efficiency of your implementation?

I had done by Tracking the candidates model . For every increase in the Offset number we are running while loop and applying filter on datahashes . I had declared candidates and stored the keys of rdd for each iteration and then used SubtractByKey to delete those keys from Datahashes since we already know them they are near . But time complexity is more for SubtractByKey and we need to Union Candidates for each iteration . Union also costs time. In the count function used Python Map Instead of for loop to reduce the complexity.