CS 585 Big Data Management
Project 2

Name: Muralidharan Kumaravel (901004143), Shashwat Misra (824507755)

Problem 1 Spatial Join

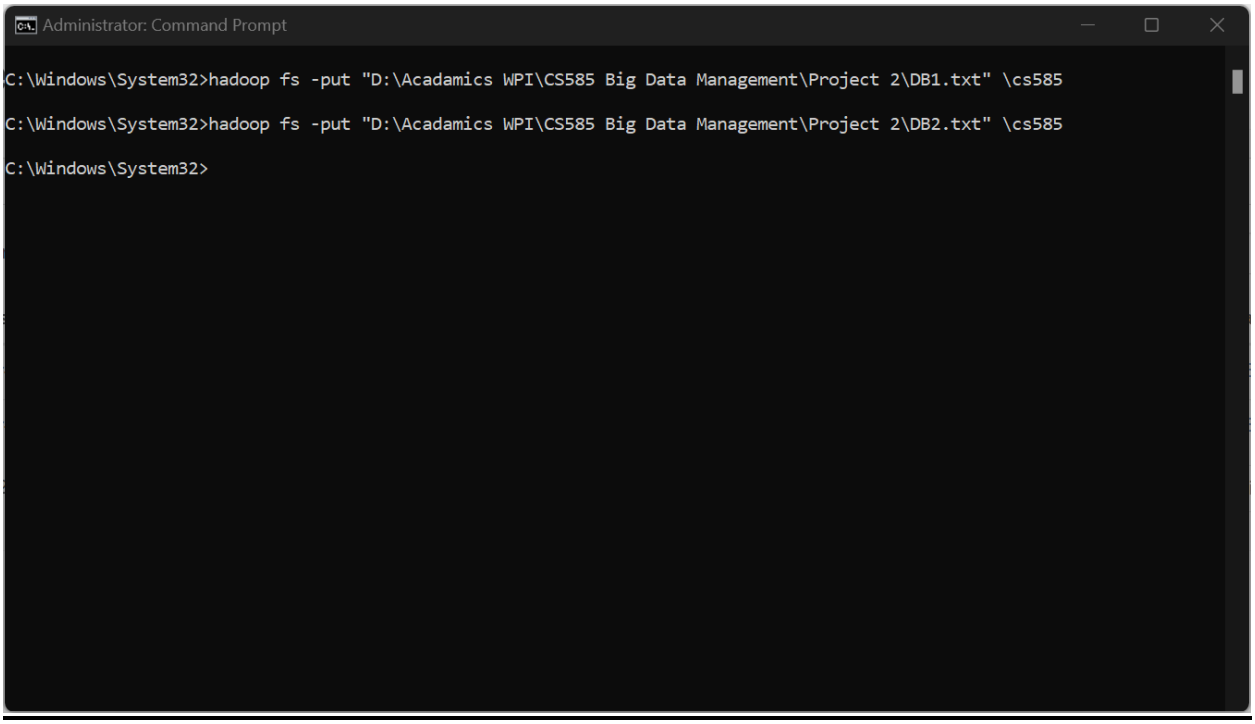
Step 1: Creating the dataset

- Here we created data set using random libraries and didn't keep brackets so that it is easier to do separation during MapReduce.

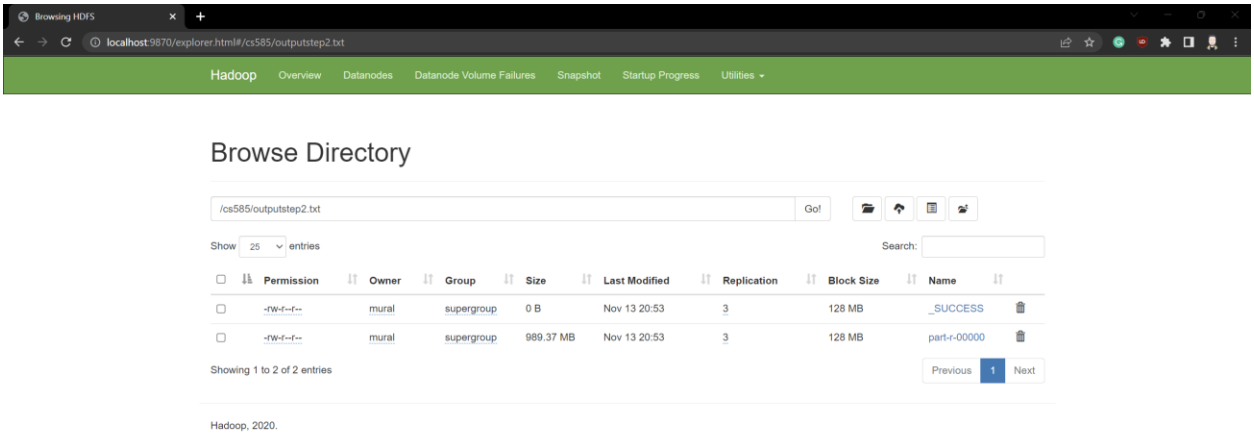
Step 2: Spatial Join

- Here we used 2 mappers for each dataset and one reducer for the joining. In the first mapper, we used a function findsection() to find the section of the graph where the point belongs to.
- In the next mapper, we did the same this is above for every corner of the rectangle as a rectangle can be present in many sections.
- We kept section id as the key so the reducer gets all the keys of the same sections at the same time.
- In reducer, we took 2 lists one of the points and one of the rectangles, and saved each rectangle and point in them respectively. Then we checked for each point to be inside the rectangle and if true we printed the output in the file.
- The code can be seen in the Step2.java file in the Step2SJ folder.
- We made a function to convert a string array consisting of points value to an integer array.

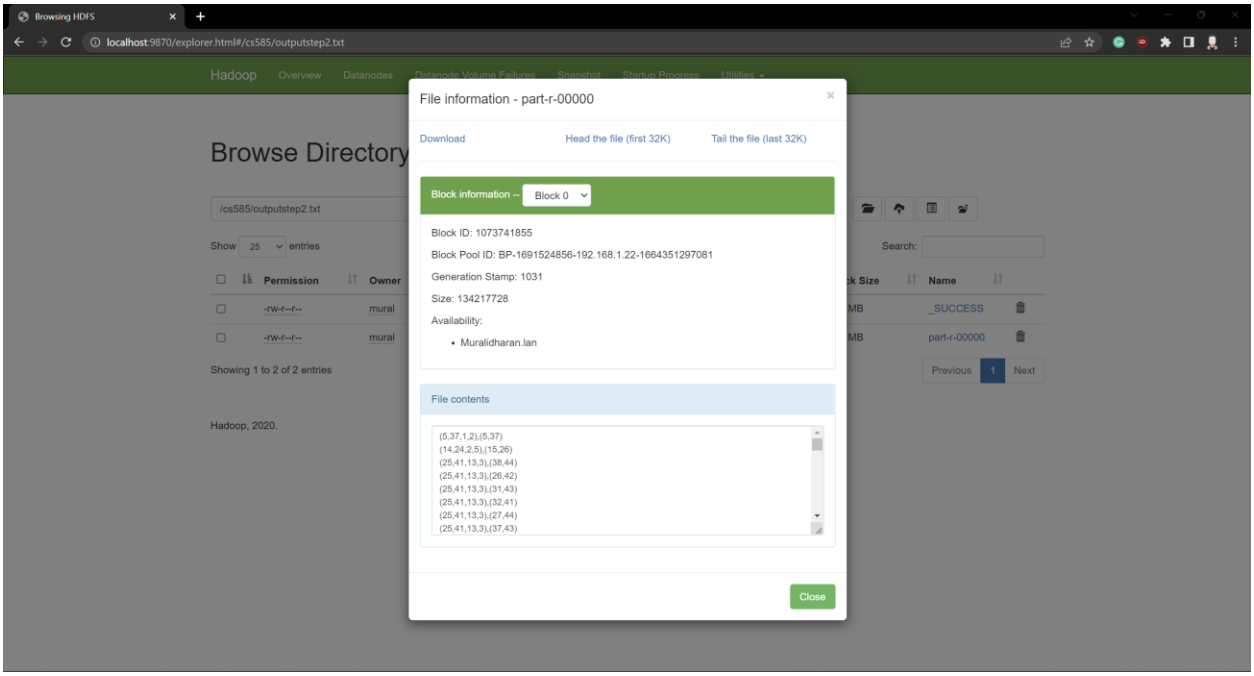
Uploading the generated data to HDFS:



Output of Spatial Join in Hadoop:



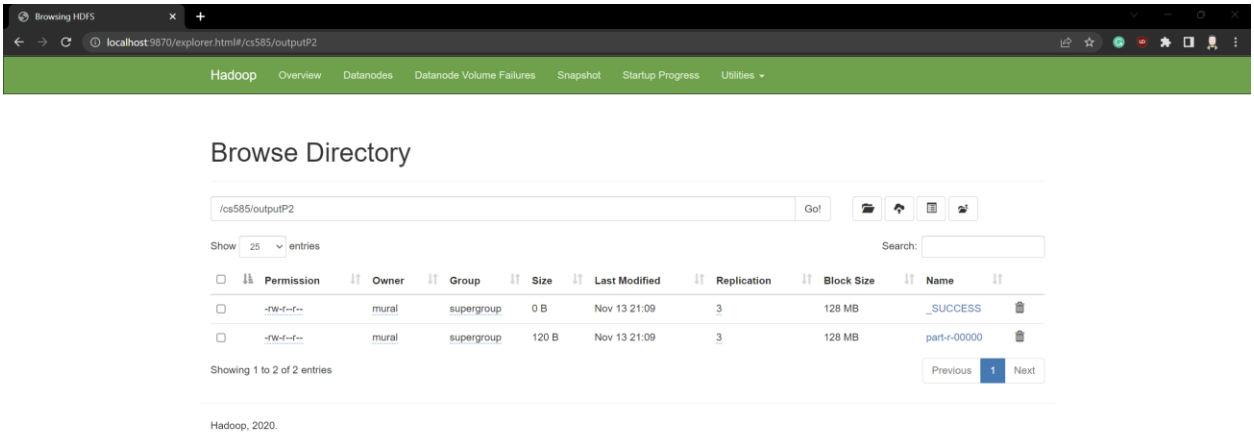
Contents of the output file for Spatial Join:



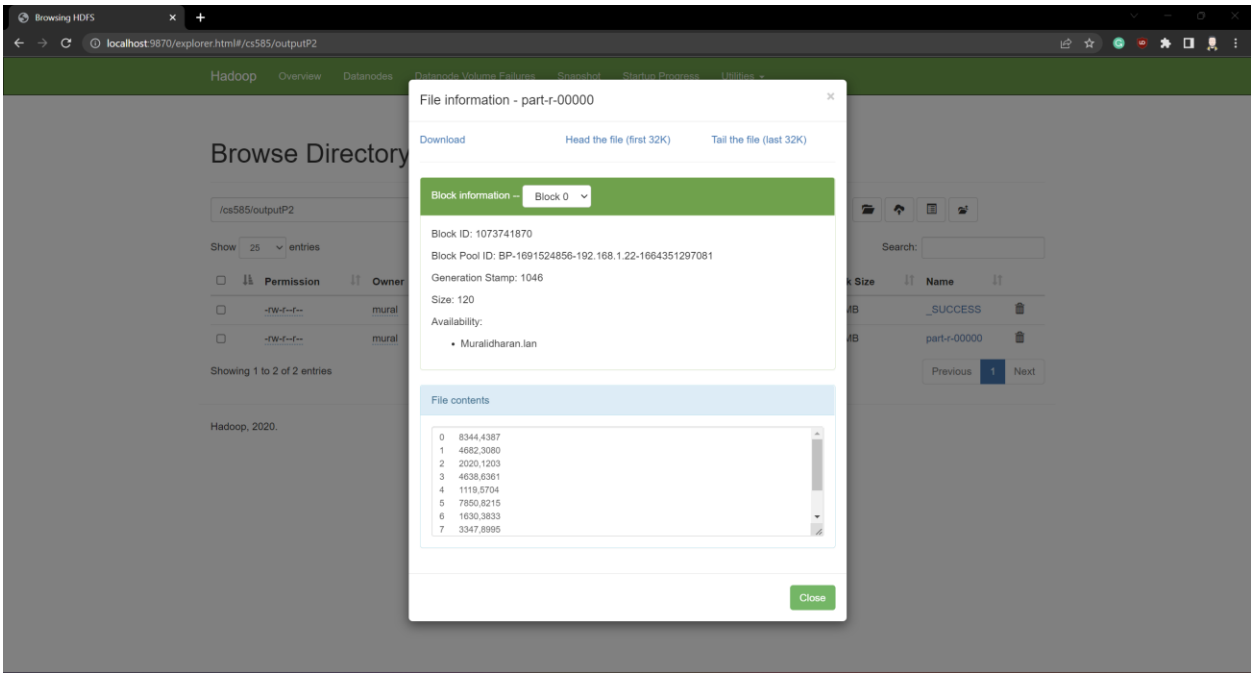
Problem 2 K-Means Clustering

- We made a function to convert a string array consisting of points value to an integer array.
- We made another function to calculate the difference between points.
- In the mapper, we check for each point closest center from the list of centers we generate randomly and assign the center's index as key and point as value.
- In the reducer, for each center, we calculate the mean of the x-axis and y-axis values and assign them as the new center points.
- In the main method first, we generate an integer array consisting of random center points.
- Then we start a loop for 6 iterations. For each iteration, we check whether there is a previous output. As the job won't run if there is already an output present.
- Then we run a job taking the input as the first database.
- We then check if the output is given by the job and we take the path of the output and check whether the center points given by the output are the same as the present center point if yes the loop ends and the output is final. If not the centers of the outputs are assigned as the new center points for the next iteration.

Output of the Map-Reduce in Hadoop:



Contents of the output file for K-Means:



Problem 3 Spark SQL

The entire workflow is in Problem3.ipynb notebook.

T1:

```
# T1
T1 = spark.sql("SELECT * FROM Transactions WHERE TransTotal>=200")
T1.show()
T1.createOrReplaceTempView("T1")
```

```
+-------+------+----------+-------------+--------------------+
|TransId|CustID|TransTotal|TransNumItems|           TransDesc|
+-------+------+----------+-------------+--------------------+
|      1|     1| 705.72736|            8|IiBvLScnYtaLgtwzL...|
|      2|     2|  816.9572|            8|VQTCgKqGPQdHdwRFz...|
|      4|     4|  304.8461|            8|JKogNbXDBmtVdkuui...|
|      5|     5|  279.5298|            1|rdPqMyMUlncnQHdMr...|
|      6|     6|  669.3279|            1|CqUydTbAExwUXnluL...|
|      7|     7| 998.74567|            1|YHSUDuReQpYHHHbuG...|
|      8|     8| 447.03073|            7|TtbxKBQLpxwBfGVOZ...|
|     10|    10| 387.45612|            9|YJAAGruWJykyQUUYK...|
|     11|    11| 838.51074|            5|eEhoLdYXiToLyzvag...|
|     12|    12| 765.49005|            5|sGlrLfSzYcRruRcpo...|
|     13|    13| 442.60776|            7|wvbYObWCzrfRLYLeq...|
|     16|    16|   584.431|            6|ZFnPyAFyGkczXzxRS...|
|     17|    17| 586.54364|            8|uBcrSKVcmLsxDDpdS...|
|     18|    18|  659.2834|           10|psvCvwlPsdfmcOXNn...|
|     20|    20|  278.5204|            3|EUJILaYCgvMiIRpAk...|
|     22|    22|  833.0222|            9|mMrkTYYqNOlqMcTiN...|
|     23|    23| 565.34393|            2|mYRDdpgtEqGFjFhYg...|
|     24|    24| 536.58563|            7|nYXTJxCbvoJjFTSwV...|
|     26|    26| 460.59375|            7|vefGzFjxTBNyDnsCV...|
|     27|    27| 205.28831|            3|fvcnmMtkemACTYGHP...|
+-------+------+----------+-------------+--------------------+
only showing top 20 rows
```

T2:

```
# T2
T2 = spark.sql("SELECT TransNumItems, MIN(TransTotal) As Min, Max(TransTotal) As Max, AVG(TransTotal) as AverageTotal FROM T1 GRO
T2.show()
T2.createOrReplaceTempView("T2")
```

```
+-------------+---------+---------+-----------------+
|TransNumItems|      Min|      Max|     AverageTotal|
+-------------+---------+---------+-----------------+
|            1|200.00201|1000.99963|600.6975258443335|
|            6|200.00002| 1000.9973|600.4016342750211|
|            3|200.00038| 1000.9982|599.8147830360873|
|            5|200.00053|1000.99585|600.4341447166371|
|            9|  200.003| 1000.9989|600.5540300106376|
|            4|200.00154|    1001.0|600.4196428638362|
|            8|200.00656| 1000.9999|600.4962424765906|
|            7|200.00629|1000.99664|600.4110138265504|
|           10| 200.0014|1000.99774|600.3107869375992|
|            2|200.00095| 1000.9995|600.0706089062178|
+-------------+---------+---------+-----------------+
```

T3:

```
#T3
T3 = spark.sql("SELECT CustID, COUNT(*) as TransCount FROM T1 GROUP BY CustID")
T3.show()
T3.createOrReplaceTempView("T3")
```

```
+------+----------+
|CustID|TransCount|
+------+----------+
|   148|        82|
|   463|        77|
|   471|        76|
|  1088|        81|
|  1238|        74|
|  1580|        76|
|  1591|        83|
|  1645|        86|
|  1829|        73|
|  1959|        86|
|  2122|        85|
|  2142|        83|
|  2366|        83|
|  2659|        80|
|  2866|        84|
|  3175|        82|
|  3749|        74|
|  3794|        87|
|  3918|        77|
|  4101|        83|
+------+----------+
only showing top 20 rows
```

T4:

```
#T4
T4 = spark.sql("SELECT * FROM Transactions WHERE TransTotal>600")
T4.show()
T4.createOrReplaceTempView("T4")
```

```
+-------+------+----------+-------------+--------------------+
|TransId|CustID|TransTotal|TransNumItems|           TransDesc|
+-------+------+----------+-------------+--------------------+
|      1|     1| 705.72736|            8|IiBvLScnYtaLgtwzL...|
|      2|     2|  816.9572|            8|VQTCgKqGPQdHdwRFz...|
|      6|     6|  669.3279|            1|CqUydTbAExwUXnluL...|
|      7|     7| 998.74567|            1|YHSUDuReQpYHHHbuG...|
|     11|    11| 838.51074|            5|eEhoLdYXiToLyzvag...|
|     12|    12| 765.49005|            5|sGlrLfSzYcRruRcpo...|
|     18|    18|  659.2834|           10|psvCvwlPsdfmcOXNn...|
|     22|    22|  833.0222|            9|mMrkTYYqNOlqMcTiN...|
|     33|    33| 763.04846|            5|IyOWroOYGUJIXpLJB...|
|     35|    35|  916.9663|            7|zJadpgvTMDbScwqyv...|
|     41|    41|   677.482|            4|phAFEURrmpwkbtxeH...|
|     42|    42| 787.64264|            4|DIVbaTIhkqOhVOPHz...|
|     45|    45| 632.58484|            3|nTECtynjQMGPEMeqX...|
|     46|    46|  658.2769|           10|mopvtyLmbQcMuInmS...|
|     48|    48|  850.5452|            2|TVczkknSzlmjDBaXA...|
|     50|    50| 965.36584|            2|eiAOtCXsiXuxeAcMJ...|
|     52|    52| 764.26117|            5|uMOqtwPUDmIsSYRaO...|
|     55|    55| 835.81506|            6|YHZQKgFUHBvrNdCMuoWQ|
|     56|    56| 859.24786|            3|eZhMwVmxALZoyheCQ...|
|     57|    57| 811.30756|            3|LsdoTaljHBHRQmHlV...|
+-------+------+----------+-------------+--------------------+
only showing top 20 rows
```

T5:

```
#T5
T5 = spark.sql("SELECT CustID, COUNT(*) as TransCount FROM T4 GROUP BY CustID")
T5.show()
T5.createOrReplaceTempView("T5")
```

```
+------+----------+
|CustID|TransCount|
+------+----------+
|  1238|        37|
|  1645|        53|
|  1959|        39|
|  2866|        37|
|  3175|        41|
|  4101|        42|
|  4818|        32|
|  6357|        38|
|  6620|        46|
|  6658|        47|
|  7253|        36|
|  7554|        34|
|  7880|        47|
|  7982|        49|
|  8389|        32|
|  8592|        31|
|  9376|        42|
|  9900|        45|
| 10206|        43|
| 10623|        49|
+------+----------+
only showing top 20 rows
```

T6:

```
#T6
T6 = spark.sql("SELECT T5.CustID, COUNT(*) as TransCount FROM T3, T5 WHERE T3.TransCount > T5.TransCount * 5 GROUP BY T5.CustID")
T6.show()
```

```
+------+----------+
|CustID|TransCount|
+------+----------+
+------+----------+
```

Problem 4 Spark RDDs

The data is created using Java. The file “Problem4DataGeneration.java” was used to generate the required data. The code has been reused from problem 1.

The entire spark workflow is in “Problem4.ipynb” notebook.

Step 2 Output (Top 50 grid cells w.r.t Relative-Density Index):

```
In [26]: rddFinal.sortBy(lambda x: x[1], ascending= False).take(50)

Out[26]: [(57253, 1.8909090909090909),
          (160955, 1.7926421404682273),
          (45314, 1.771986970684039),
          (55626, 1.7668711656441718),
          (85474, 1.75),
          (192689, 1.748502994011976),
          (133351, 1.7483443708609272),
          (187309, 1.7425742574257426),
          (56480, 1.7391304347826086),
          (144870, 1.7297297297297298),
          (233049, 1.7297297297297298),
          (45119, 1.7245508982035929),
          (86355, 1.7230769230769232),
          (115221, 1.7218543046357615),
          (148771, 1.7215189873417722),
          (198808, 1.7206703910614525),
          (186662, 1.7181208053691275),
          (132279, 1.7177914110429449),
          (71721, 1.7124600638977636),
```

Step 3 Output (Relative Density of the Neighbors of the Top 50 grid cells):

```
In [34]: finalList

Out[34]: [[57253,
           [[56752, [0.6091370558375635]],
            [57252, [1.229050279329609]],
            [57752, [0.7924528301886793]],
            [56753, [0.7035175879396985]],
            [57753, [0.7536231884057971]],
            [56754, [0.9487870619946092]],
            [57254, [0.7741935483870968]],
            [57754, [0.9702970297029703]]]],
          [160955,
           [[160454, [0.8328767123287671]],
            [160954, [0.7661971830985915]],
            [161454, [0.9662921348314607]],
            [160455, [0.7314285714285714]],
            [161455, [0.9341317365269461]],
            [160456, [1.032967032967033]],
            [160956, [0.7017543859649122]],
            [161456, [0.8228571428571428]]]],
          [45314,
```