DAS 839 NoSQL Systems

Assignment-2

Submitted by :

Group 2:

Abhay bhadouriya – MT2024003

jainish parmar - MT2024065

Jaimin Jadvani – MT2024064

Files Submitted :

1. Report File
2. Screenshot Folder
3. TimeChecker.py Python program for Question 3
4. Assignment_2_Q_2.zip -- Question 2
5. Assignment_2_Q_3.zip -- Question 3

Question 1:

Section 1 : M-Counter state-based object (SBO) qualifies as a CRDT?

To Qualify CRDT a Statebased object should follow these 4 properties.

1. Must Follow Associative law --> Here merge following

max(max(a,b),c)=max(a,max(b,c))

so here it is satisfied

2. Must Follow Commutativity law --> Here merge following the merge(x,y)=merge(y,x)

so here it is satisfied

3. Must follow Idempotency law --> here Merge following the

merge(x,x)=x

so here it is also satisfied

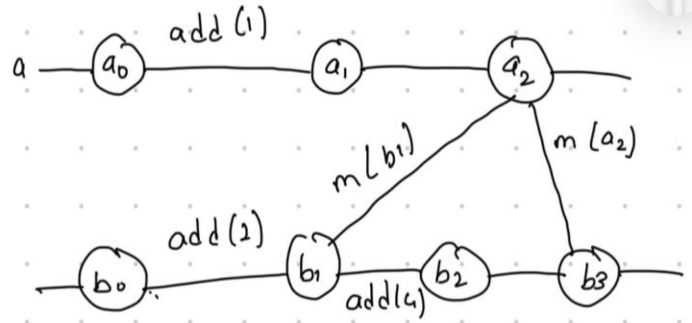
4. Must follow increasing Updates --> here it following

Merge(x,y)=y

so here it is also satisfied

So here it is satisfied so answer is true it is Following CRDT.

Section 2 :



=) 3 Servers

	state	query	history
a_0	Sum=0 cnt:0	0	{ }
a_1	Sum =1 Cnt1	1	{0}
a_2	Sum =3 Cnt 2	3	{0,1}
b_0	Sum=0 Cnt:0	0	{ }
b_1	Sum=2 Cnt:1	2	{1}
b_2	Sum=4 Cnt=2	4	{1 2}
b_3	Sum=7 Cnt3	7	{0, 1,2}

Section 3

This balance is a sensitive as formal correctness vs as intuitive application semantics

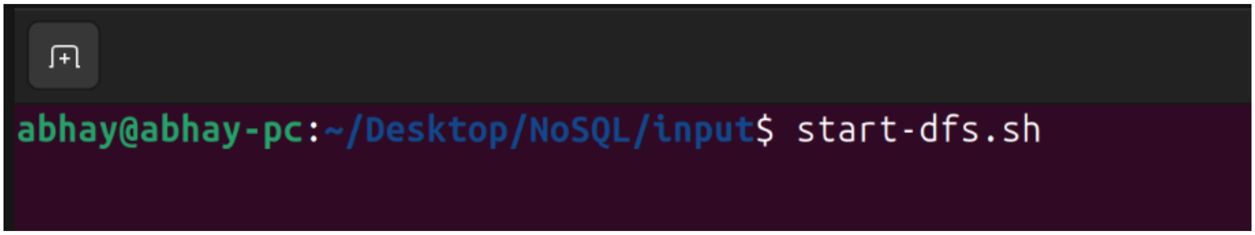
Trade-offs:

Focus on Correctness: Ensure global effectiveness while potentially producing an opposite result towards the user (e.g., reordering, collaborative editing.)

User Input based: Feels idiomatic to users while augmenting it with new custom logic, thus adding processing overhead and therefore complexity.

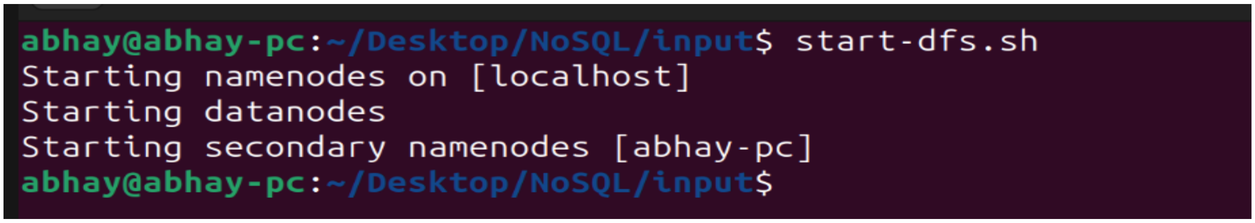
Question 2:

1 **Start Hadoop:** start-dfs.sh



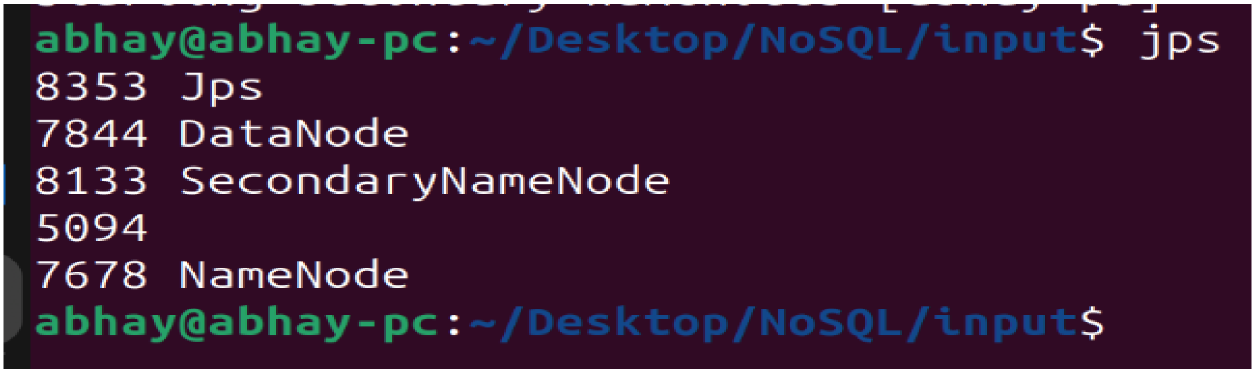
```
abhay@abhay-pc:~/Desktop/NoSQL/input$ start-dfs.sh
```

2. **Hadoop Running:**



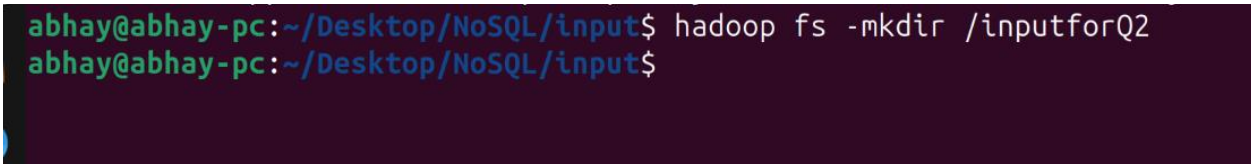
```
abhay@abhay-pc:~/Desktop/NoSQL/input$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [abhay-pc]
abhay@abhay-pc:~/Desktop/NoSQL/input$
```

3. **USE JPS** to check Running ,NameNode,DataNode,JPS,Secondary NameNode



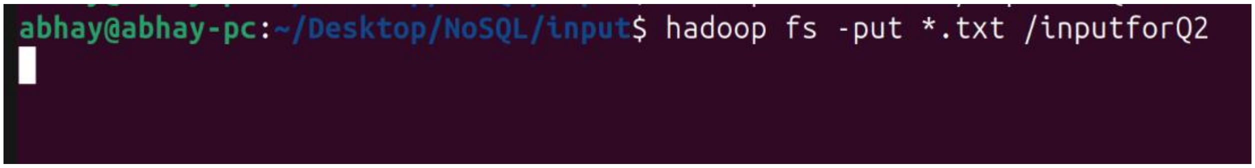
```
abhay@abhay-pc:~/Desktop/NoSQL/input$ jps
8353 Jps
7844 DataNode
8133 SecondaryNameNode
5094
7678 NameNode
abhay@abhay-pc:~/Desktop/NoSQL/input$
```

4. **Create a input directory:**



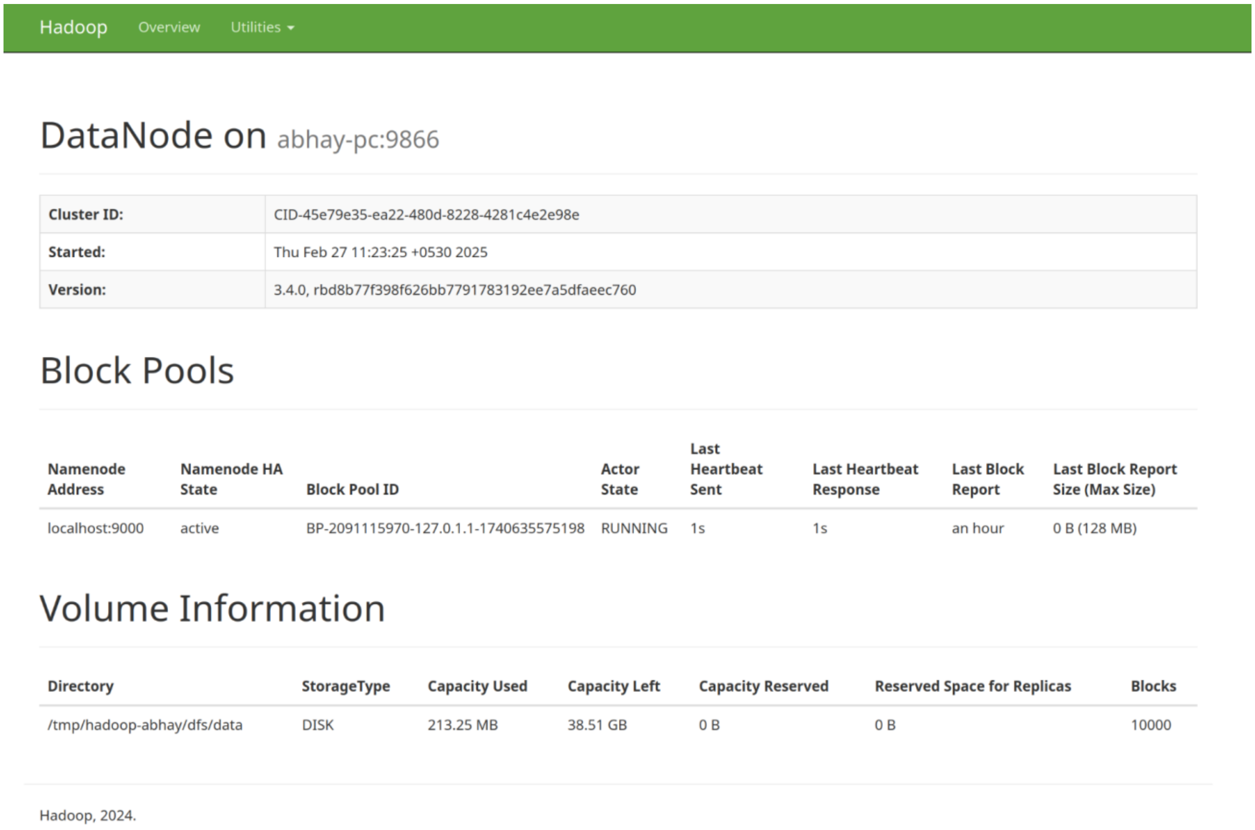
```
abhay@abhay-pc:~/Desktop/NoSQL/input$ hadoop fs -mkdir /inputforQ2
abhay@abhay-pc:~/Desktop/NoSQL/input$
```

5. **Copy all 10000 files to InputforQ2 directory**



```
abhay@abhay-pc:~/Desktop/NoSQL/input$ hadoop fs -put *.txt /inputforQ2
█
```

6. **Check if data is copied or not :** Goto http://localhost:9870 -> on overview Tab Click -> LiveNode and click on datanode address -> useally it is http://localhost:9864



DataNode on abhay-pc:9866

Cluster ID:	CID-45e79e35-ea22-480d-8228-4281c4e2e98e
Started:	Thu Feb 27 11:23:25 +0530 2025
Version:	3.4.0, rbd8b77f398f626bb7791783192ee7a5dfaeec760

Block Pools

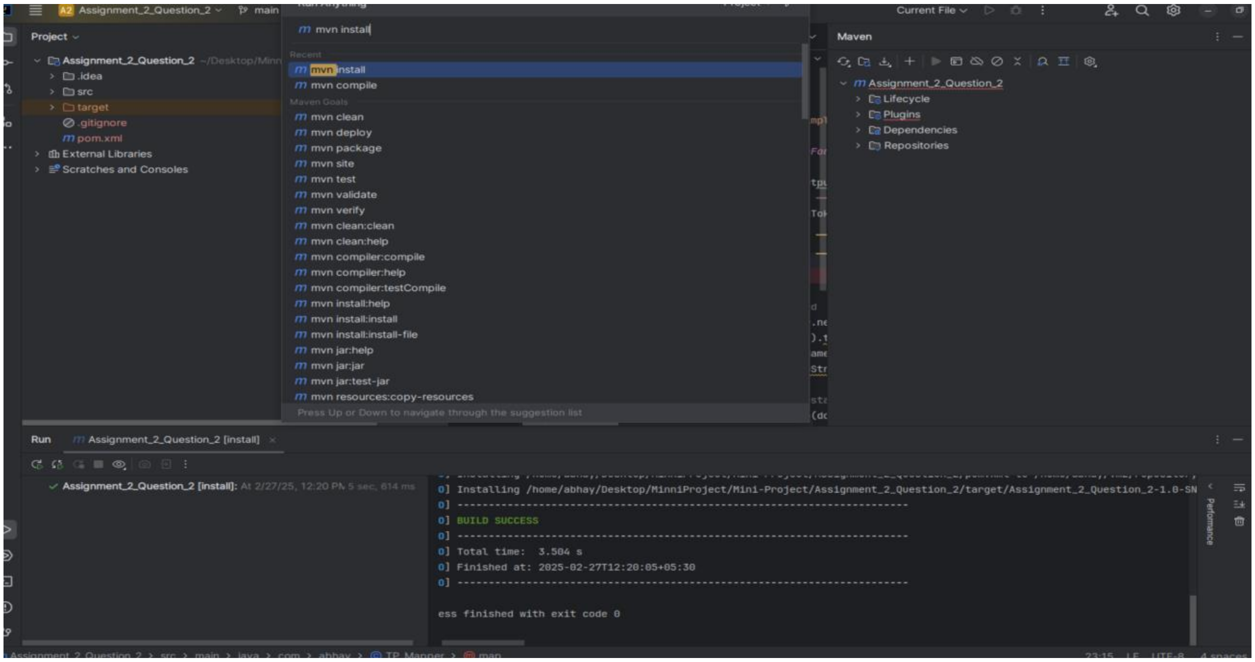
Namenode Address	Namenode HA State	Block Pool ID	Actor State	Last Heartbeat Sent	Last Heartbeat Response	Last Block Report	Last Block Report Size (Max Size)
localhost:9000	active	BP-2091115970-127.0.1.1-1740635575198	RUNNING	1s	1s	an hour	0 B (128 MB)

Volume Information

Directory	StorageType	Capacity Used	Capacity Left	Capacity Reserved	Reserved Space for Replicas	Blocks
/tmp/hadoop-abhay/dfs/data	DISK	213.25 MB	38.51 GB	0 B	0 B	10000

Hadoop, 2024.

7. **Creating Jar Files:-** unzip the Assignment_2_Q_2.zip -> open in intellij -> Click on maven on right side tab -> click on Mvn install -> it will create the Jar file



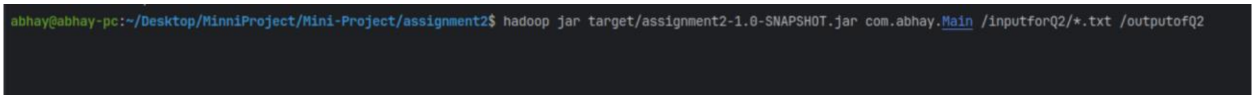
```
O] Installing /home/abhay/Desktop/MinniProject/Mini-Project/Assignment_2_Question_2/target/Assignment_2_Question_2-1.0-SN
O] ------------------------------------------------------------------------
O] BUILD SUCCESS
O] ------------------------------------------------------------------------
O] Total time:  3.504 s
O] Finished at: 2025-02-27T12:20:05+05:30
O] ------------------------------------------------------------------------

ess finished with exit code 0
```

8. **Running the Program 2 :** use the command displayed in below screentshot to run

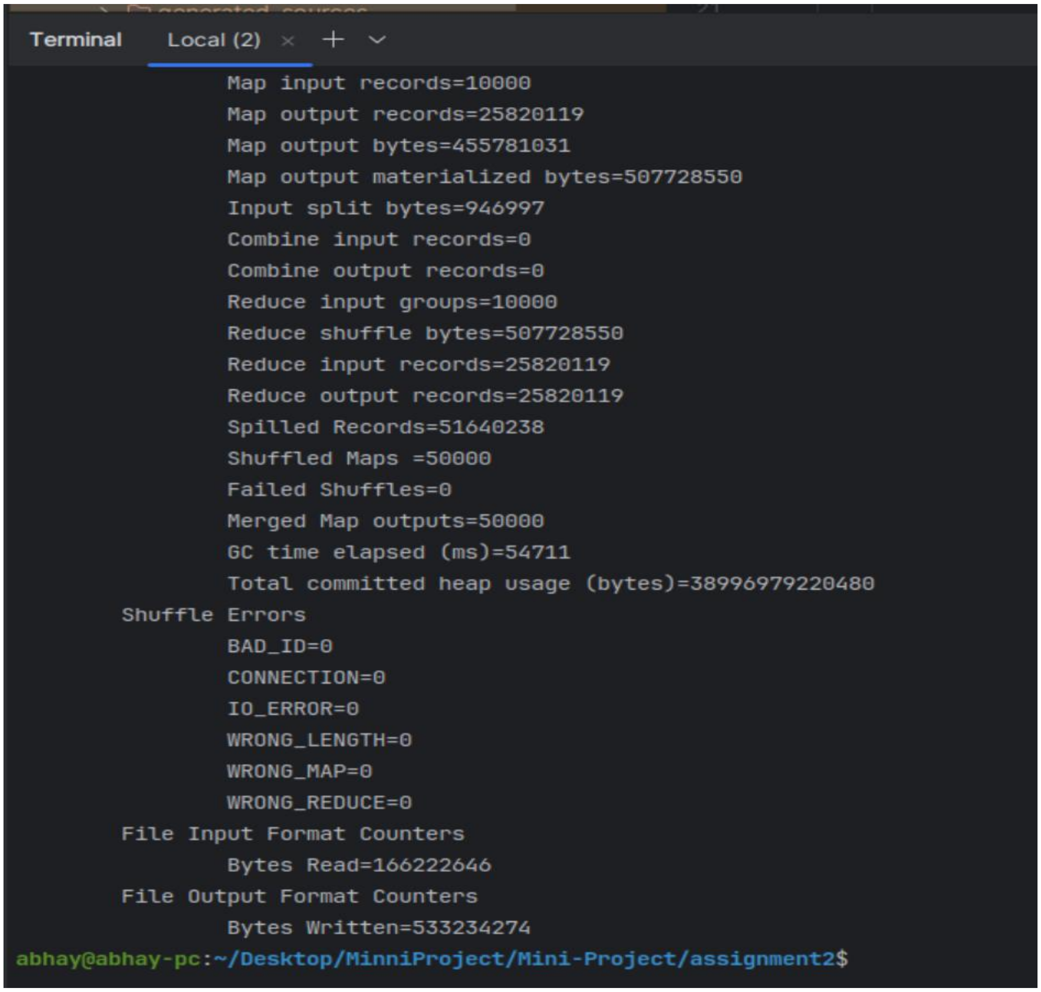
/inputforQ2 --- is input data directory

/ outputofQ2 --- is output directory



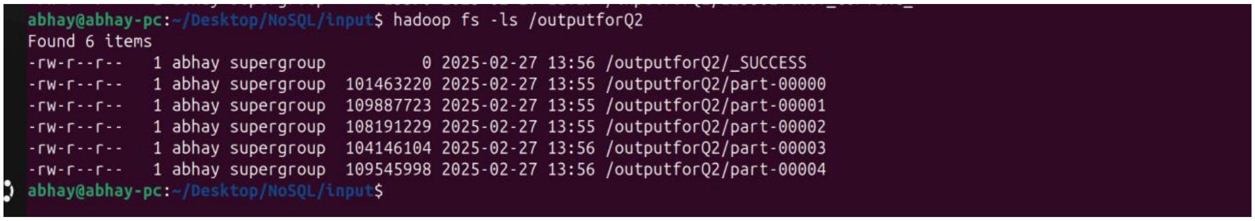
```
abhay@abhay-pc:~/Desktop/MinniProject/Mini-Project/assignment2$ hadoop jar target/assignment2-1.0-SNAPSHOT.jar com.abhay.Main /inputforQ2/*.txt /outputofQ2
```

9. **Completed Running**



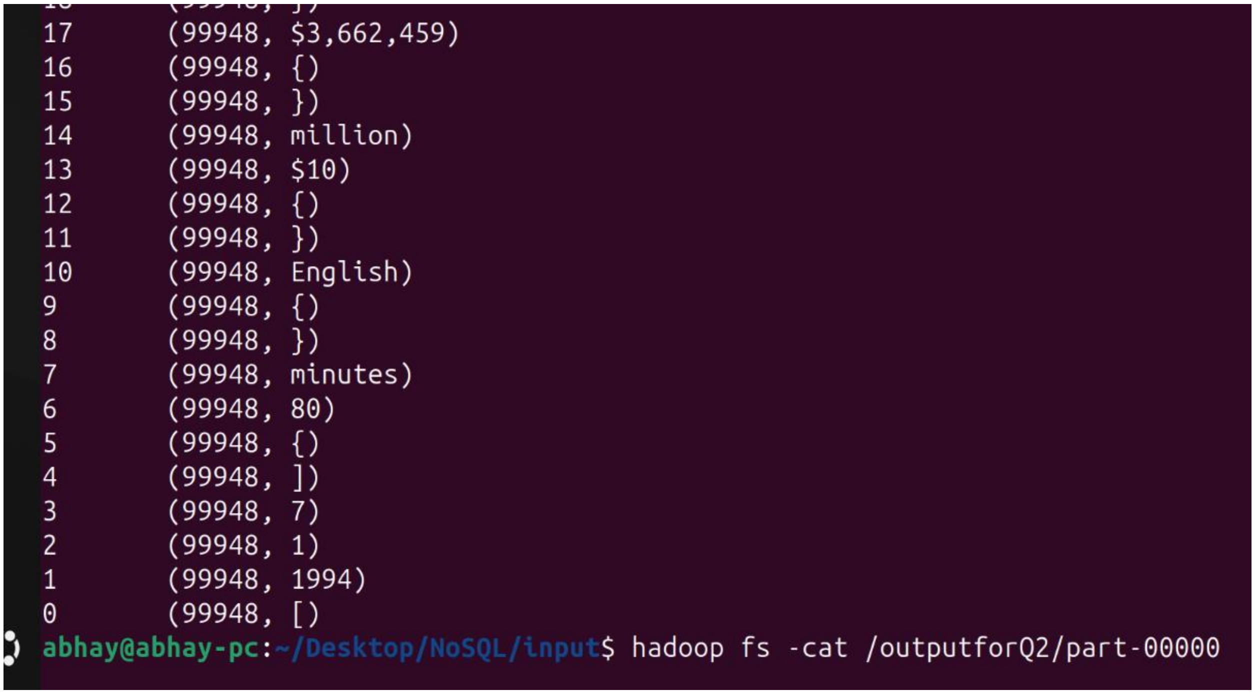
```
                Map input records=10000
                Map output records=25820119
                Map output bytes=455781031
                Map output materialized bytes=507728550
                Input split bytes=946997
                Combine input records=0
                Combine output records=0
                Reduce input groups=10000
                Reduce shuffle bytes=507728550
                Reduce input records=25820119
                Reduce output records=25820119
                Spilled Records=51640238
                Shuffled Maps =50000
                Failed Shuffles=0
                Merged Map outputs=50000
                GC time elapsed (ms)=54711
                Total committed heap usage (bytes)=38996979220480
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=166222646
        File Output Format Counters
                Bytes Written=533234274
abhay@abhay-pc:~/Desktop/MinniProject/Mini-Project/assignment2$
```

10. **OutPut for Question 2 is 5 files because we used 5 reducers:**



```
abhay@abhay-pc:~/Desktop/NoSQL/input$ hadoop fs -ls /outputforQ2
Found 6 items
-rw-r--r--   1 abhay supergroup          0 2025-02-27 13:56 /outputforQ2/_SUCCESS
-rw-r--r--   1 abhay supergroup  101463220 2025-02-27 13:55 /outputforQ2/part-00000
-rw-r--r--   1 abhay supergroup  109887723 2025-02-27 13:55 /outputforQ2/part-00001
-rw-r--r--   1 abhay supergroup  108191229 2025-02-27 13:55 /outputforQ2/part-00002
-rw-r--r--   1 abhay supergroup  104146104 2025-02-27 13:56 /outputforQ2/part-00003
-rw-r--r--   1 abhay supergroup  109545998 2025-02-27 13:56 /outputforQ2/part-00004
abhay@abhay-pc:~/Desktop/NoSQL/input$
```

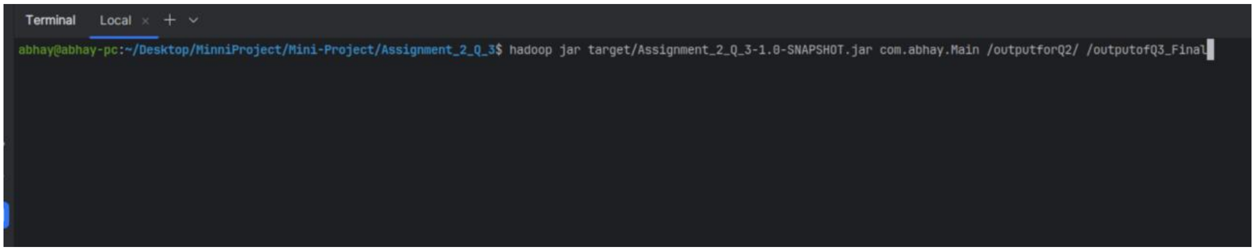
11. **Content of output files as specified in Question (Index,(Fileid,word))**



```
17      (99948, $3,662,459)
16      (99948, {)
15      (99948, })
14      (99948, million)
13      (99948, $10)
12      (99948, {)
11      (99948, })
10      (99948, English)
9       (99948, {)
8       (99948, })
7       (99948, minutes)
6       (99948, 80)
5       (99948, {)
4       (99948, ])
3       (99948, 7)
2       (99948, 1)
1       (99948, 1994)
0       (99948, [)
abhay@abhay-pc:~/Desktop/NoSQL/input$ hadoop fs -cat /outputforQ2/part-00000
```

Question 3

1. **Now Run the Third Program also: Running Question No 3**



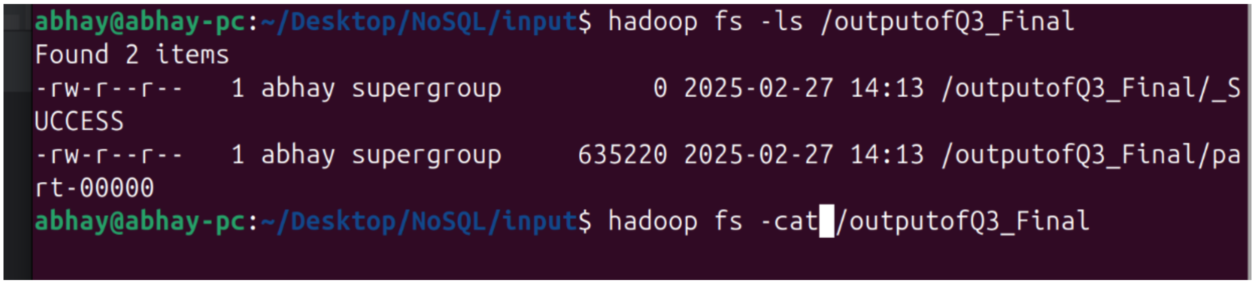
```
abhay@abhay-pc:~/Desktop/MinniProject/Mini-Project/Assignment_2_Q_3$ hadoop jar target/Assignment_2_Q_3-1.0-SNAPSHOT.jar com.abhay.Main /outputforQ2/ /outputofQ3_Final
```

/outputforQ2 -- is output data generated by question 2 in (index,(fileid,word)) manner

/outputofQ3_final – is output data generated by question 3 in (index,(time,word))

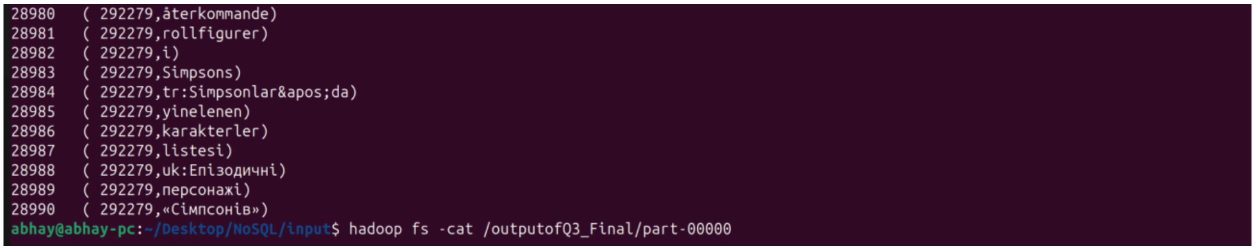
Here word is representing the word at x index of maximum time based on fileid

2. **OutputFiles genereted :** we used 1 reducer so only file generated



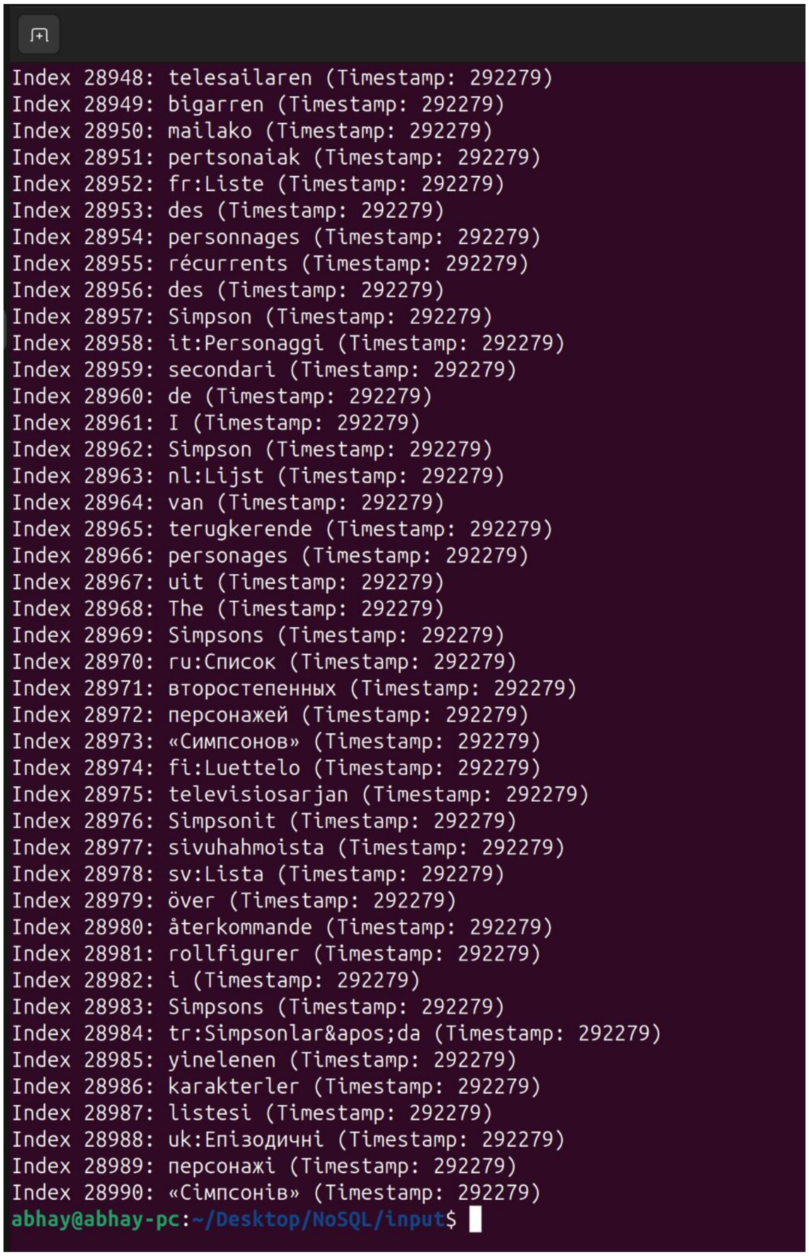
```
abhay@abhay-pc:~/Desktop/NoSQL/input$ hadoop fs -ls /outputofQ3_Final
Found 2 items
-rw-r--r--   1 abhay supergroup          0 2025-02-27 14:13 /outputofQ3_Final/_S
UCCESS
-rw-r--r--   1 abhay supergroup     635220 2025-02-27 14:13 /outputofQ3_Final/pa
rt-00000
abhay@abhay-pc:~/Desktop/NoSQL/input$ hadoop fs -cat /outputofQ3_Final
```

3. Output of file :



```
28980   ( 292279,återkommande)
28981   ( 292279,rollfigurer)
28982   ( 292279,i)
28983   ( 292279,Simpsons)
28984   ( 292279,tr:Simpsonlar&apos;da)
28985   ( 292279,yinelenen)
28986   ( 292279,karakterler)
28987   ( 292279,listesi)
28988   ( 292279,uk:Епізодичні)
28989   ( 292279,персонажі)
28990   ( 292279,«Сімпсонів»)
abhay@abhay-pc:~/Desktop/NoSQL/input$ hadoop fs -cat /outputofQ3_Final/part-00000
```

4. Now Cross check the out of file Question 3 with our own Custom code for generated key value pair program



```
Index 28948: telesailaren (Timestamp: 292279)
Index 28949: bigarren (Timestamp: 292279)
Index 28950: mailako (Timestamp: 292279)
Index 28951: pertsonaiak (Timestamp: 292279)
Index 28952: fr:Liste (Timestamp: 292279)
Index 28953: des (Timestamp: 292279)
Index 28954: personnages (Timestamp: 292279)
Index 28955: récurrents (Timestamp: 292279)
Index 28956: des (Timestamp: 292279)
Index 28957: Simpson (Timestamp: 292279)
Index 28958: it:Personaggi (Timestamp: 292279)
Index 28959: secondari (Timestamp: 292279)
Index 28960: de (Timestamp: 292279)
Index 28961: I (Timestamp: 292279)
Index 28962: Simpson (Timestamp: 292279)
Index 28963: nl:Lijst (Timestamp: 292279)
Index 28964: van (Timestamp: 292279)
Index 28965: terugkerende (Timestamp: 292279)
Index 28966: personages (Timestamp: 292279)
Index 28967: uit (Timestamp: 292279)
Index 28968: The (Timestamp: 292279)
Index 28969: Simpsons (Timestamp: 292279)
Index 28970: ru:Список (Timestamp: 292279)
Index 28971: второстепенных (Timestamp: 292279)
Index 28972: персонажей (Timestamp: 292279)
Index 28973: «Симпсонов» (Timestamp: 292279)
Index 28974: fi:Luettelo (Timestamp: 292279)
Index 28975: televisiosarjan (Timestamp: 292279)
Index 28976: Simpsonit (Timestamp: 292279)
Index 28977: sivuhahmoista (Timestamp: 292279)
Index 28978: sv:Lista (Timestamp: 292279)
Index 28979: över (Timestamp: 292279)
Index 28980: återkommande (Timestamp: 292279)
Index 28981: rollfigurer (Timestamp: 292279)
Index 28982: i (Timestamp: 292279)
Index 28983: Simpsons (Timestamp: 292279)
Index 28984: tr:Simpsonlar&apos;da (Timestamp: 292279)
Index 28985: yinelenen (Timestamp: 292279)
Index 28986: karakterler (Timestamp: 292279)
Index 28987: listesi (Timestamp: 292279)
Index 28988: uk:Епізодичні (Timestamp: 292279)
Index 28989: персонажі (Timestamp: 292279)
Index 28990: «Сімпсонів» (Timestamp: 292279)
abhay@abhay-pc:~/Desktop/NoSQL/input$
```

Mini Report for Question 2 and Question 3

Question 2: Hadoop Implementation and Execution

Objective: Process 10,000 text files using Hadoop to generate an output with the format (Index, (FileID, Word)).

Execution Steps:

1. **Hadoop Startup:**
    a. Command: start-all.sh
    b. Result: Started NameNode, DataNode, and SecondaryNameNode on localhost (abhay-pc).
    c. Verification: Used jps to confirm:
        i. NameNode (7678)
        ii. DataNode (7844)
        iii. SecondaryNameNode (8133)
        iv. Jps (8353)
2. **Data Preparation:**
    a. Created input directory: hadoop fs -mkdir /inputforQ2
    b. Copied 10,000 text files: hadoop fs -put *.txt /inputforQ2
    c. Verified via Hadoop UI (http://localhost:9864): 213.25 MB used, 10,000 blocks.
3. **Program Compilation:**
    a. Unzipped Assignment_2_Q_2.zip, opened in IntelliJ, and built JAR file with Maven (mvn install).
4. **Program Execution:**
    a. Ran Hadoop job with input /inputforQ2 and output /outputofQ2.
    b. Used 5 reducers, producing 5 output files.
5. **Output Statistics:**
    a. Screenshot attached Please refer to "Completed program 2.png"
6. **Output Format:**
    a. Format: (Index, (FileID, Word)).

Summary: Successfully processed 10,000 files into 25.8 million records across 5 files, demonstrating distributed computing efficiency.

Question 3: Processing and Output Generation

Objective: Likely involves further processing or indexing of Question 2's output

Execution Steps:

1. **Preparation:**
    a. Unzipped Assignment_2_Q_3.zip to access code/resources.
    b. Confirmed Hadoop cluster availability with jps.
2. **Input Data Setup:**
    a. Used Question 2's output: hadoop fs -ls /outputofQ2 to verify 5 output files.
3. **Program Execution:**
    a. Ran the Hadoop job from Assignment_2_Q_3.zip (assumed to be another JAR file):
        i. Ran Hadoop job with input /outputofQ2 and output /outputofQ3_final
4. **Output Verification:**
    a. Checked results in /outputofQ3_final via hadoop fs -cat /outputofQ3_final/part-r-00000 (example).

    b. Sample output :

        i. 28948, (292279 , telesatiaren)
        ii. 28949,(292279 ,bigarren)

    c. Indicates word indexing with timestamps.

Note on TimeChecker.py:

- run after Question 3 jar to verify the output:
    - usage: copy TimeChecker.py in txt file directory which consist all 10000 files
    - Then just run :- python 3 TimeChecker.py
    - Purpose: Could measure execution time or validate output timestamps and we need to cross check the output of python program and question 3 output