COMP1702-BIG DATA REPORT

FATMANUR ERTAS-001360077

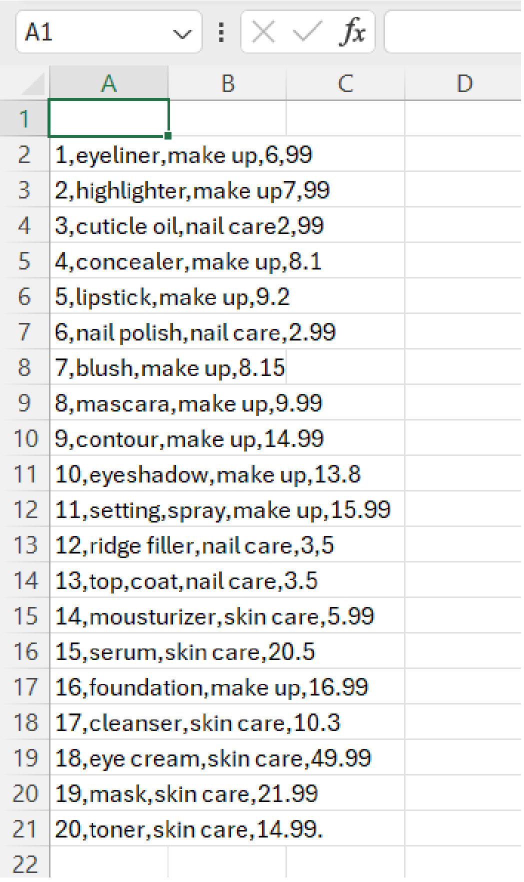
TASK-A Hive Data Warehouse Design

Creating the tables with hive

```
hive> create table Products(ProductID INT, ProductName STRING,
Category STRING, Price DOUBLE) ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',' STORED AS TEXTFILE;
OK
Time taken: 0.264 seconds
hive> create table Customers(CustomerID INT, Name STRING, City
STRING, Country STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY
',' STORED AS TEXTFILE;
OK
Time taken: 0.08 seconds
hive> create table Sales(SaleID INT, ProductID INT, CustomerID INT,
SaleDate DATE, Quantity INT) ROW FORMAT DELIMITED FIELDS TERMINATED
BY ',' STORED AS TEXTFILE;
OK
Time taken: 0.063 seconds
```

Loading the Tables from .csv files

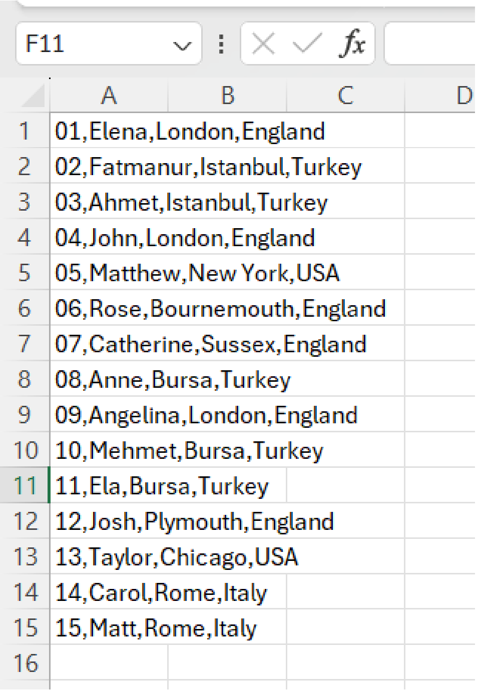
```
hive> LOAD DATA INPATH 'Products.txt' OVERWRITE INTO TABLE Products;
FAILED: SemanticException Line 1:17 Invalid path ''Products.txt'':
No files matching path
hdfs://localhost:9000/user/hadoop/Products.txt
hive> LOAD DATA INPATH 'Products.csv' OVERWRITE INTO TABLE Products;
Loading data to table default.products
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted hdfs://localhost:9000/user/hive/warehouse/products
Table default.products stats: [numFiles=1, numRows=0, totalSize=524,
rawDataSize=0]
OK
Time taken: 0.514 seconds
```



	A	B	C	D
1				
2	1,eyeliner,make up,6,99			
3	2,highlighter,make up7,99			
4	3,cuticle oil,nail care2,99			
5	4,concealer,make up,8.1			
6	5,lipstick,make up,9.2			
7	6,nail polish,nail care,2.99			
8	7,blush,make up,8.15			
9	8,mascara,make up,9.99			
10	9,contour,make up,14.99			
11	10,eyeshadow,make up,13.8			
12	11,setting,spray,make up,15.99			
13	12,ridge filler,nail care,3,5			
14	13,top,coat,nail care,3.5			
15	14,mousturizer,skin care,5.99			
16	15,serum,skin care,20.5			
17	16,foundation,make up,16.99			
18	17,cleanser,skin care,10.3			
19	18,eye cream,skin care,49.99			
20	19,mask,skin care,21.99			
21	20,toner,skin care,14.99.			
22				

Products table

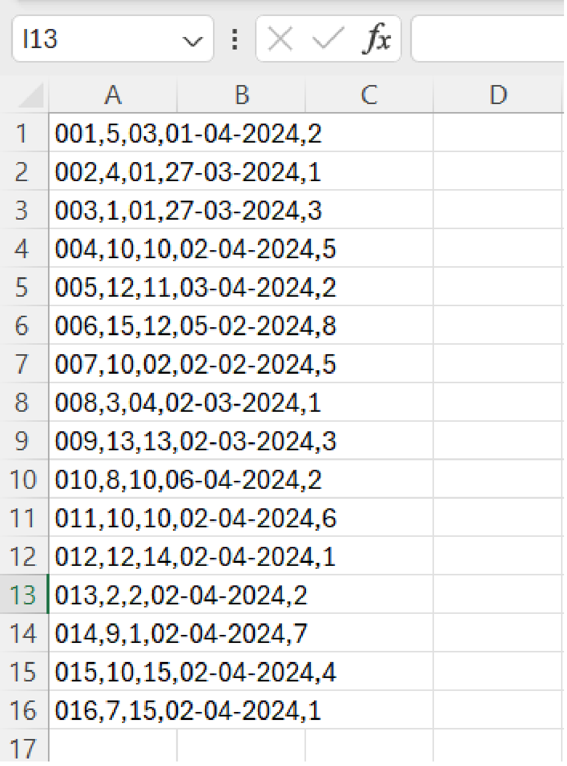
```
hive> LOAD DATA INPATH 'Customers.csv' OVERWRITE INTO TABLE
Customers;
Loading data to table default.customers
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted hdfs://localhost:9000/user/hive/warehouse/customers
Table default.customers stats: [numFiles=1, numRows=0,
totalSize=357, rawDataSize=0]
OK
Time taken: 0.287 seconds
```



	A	B	C	D
1	01,Elena,London,England			
2	02,Fatmanur,Istanbul,Turkey			
3	03,Ahmet,Istanbul,Turkey			
4	04,John,London,England			
5	05,Matthew,New York,USA			
6	06,Rose,Bournemouth,England			
7	07,Catherine,Sussex,England			
8	08,Anne,Bursa,Turkey			
9	09,Angelina,London,England			
10	10,Mehmet,Bursa,Turkey			
11	11,Ela,Bursa,Turkey			
12	12,Josh,Plymouth,England			
13	13,Taylor,Chicago,USA			
14	14,Carol,Rome,Italy			
15	15,Matt,Rome,Italy			
16				

Customers Table

```
hive> LOAD DATA INPATH 'Sales.csv' OVERWRITE INTO TABLE Sales;
Loading data to table default.sales
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted hdfs://localhost:9000/user/hive/warehouse/sales
Table default.sales stats: [numFiles=1, numRows=0, totalSize=358,
rawDataSize=0]
OK
Time taken: 0.257 seconds
```



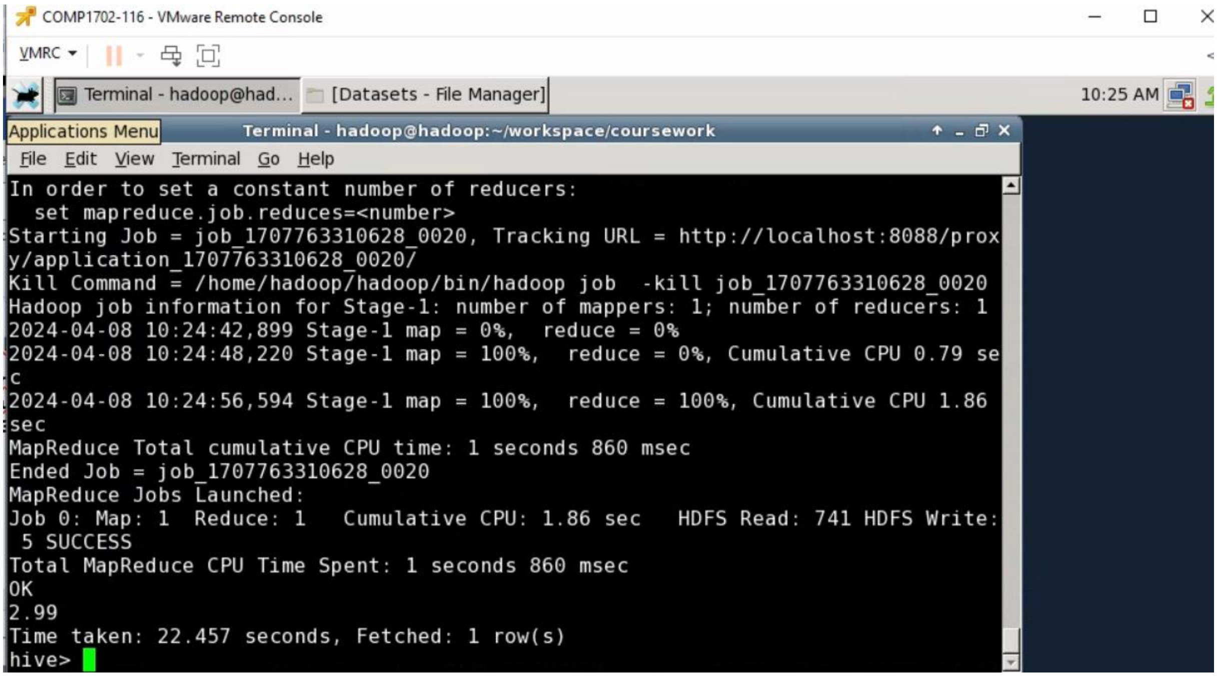
	A	B	C	D
1	001,5,03,01-04-2024,2			
2	002,4,01,27-03-2024,1			
3	003,1,01,27-03-2024,3			
4	004,10,10,02-04-2024,5			
5	005,12,11,03-04-2024,2			
6	006,15,12,05-02-2024,8			
7	007,10,02,02-02-2024,5			
8	008,3,04,02-03-2024,1			
9	009,13,13,02-03-2024,3			
10	010,8,10,06-04-2024,2			
11	011,10,10,02-04-2024,6			
12	012,12,14,02-04-2024,1			
13	013,2,2,02-04-2024,2			
14	014,9,1,02-04-2024,7			
15	015,10,15,02-04-2024,4			
16	016,7,15,02-04-2024,1			
17				

Sales Table

Result of the 1st querie

1-hive> SELECT MIN(Price) FROM Products;

This query finds the lowest price among the prices of products in the Products table.



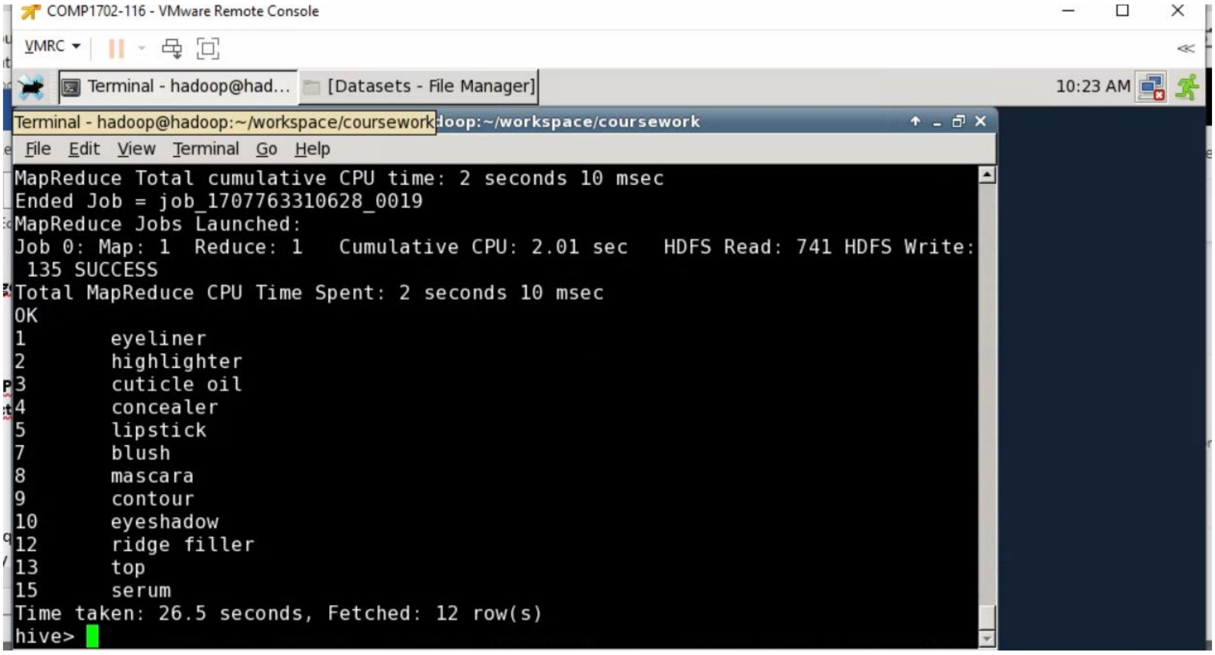
COMP1702-116 - VMware Remote Console
VMRC
Terminal - hadoop@had...
[Datasets - File Manager]
10:25 AM
Applications Menu
Terminal - hadoop@hadoop:~/workspace/coursework
File Edit View Terminal Go Help

```
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1707763310628_0020, Tracking URL = http://localhost:8088/prox
y/application_1707763310628_0020/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1707763310628_0020
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-08 10:24:42,899 Stage-1 map = 0%,  reduce = 0%
2024-04-08 10:24:48,220 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.79 se
c
2024-04-08 10:24:56,594 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.86
sec
MapReduce Total cumulative CPU time: 1 seconds 860 msec
Ended Job = job_1707763310628_0020
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.86 sec   HDFS Read: 741 HDFS Write:
 5 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 860 msec
OK
2.99
Time taken: 22.457 seconds, Fetched: 1 row(s)
hive>
```

Result of the 2nd querie

2-hive> SELECT DISTINCT Products.ProductID, Products.ProductName FROM Products JOIN Sales ON Products.ProductID = Sales.ProductID;

This query returns the list of products that have been sold. The JOIN operator combines the Products and Sales tables based on ProductID.



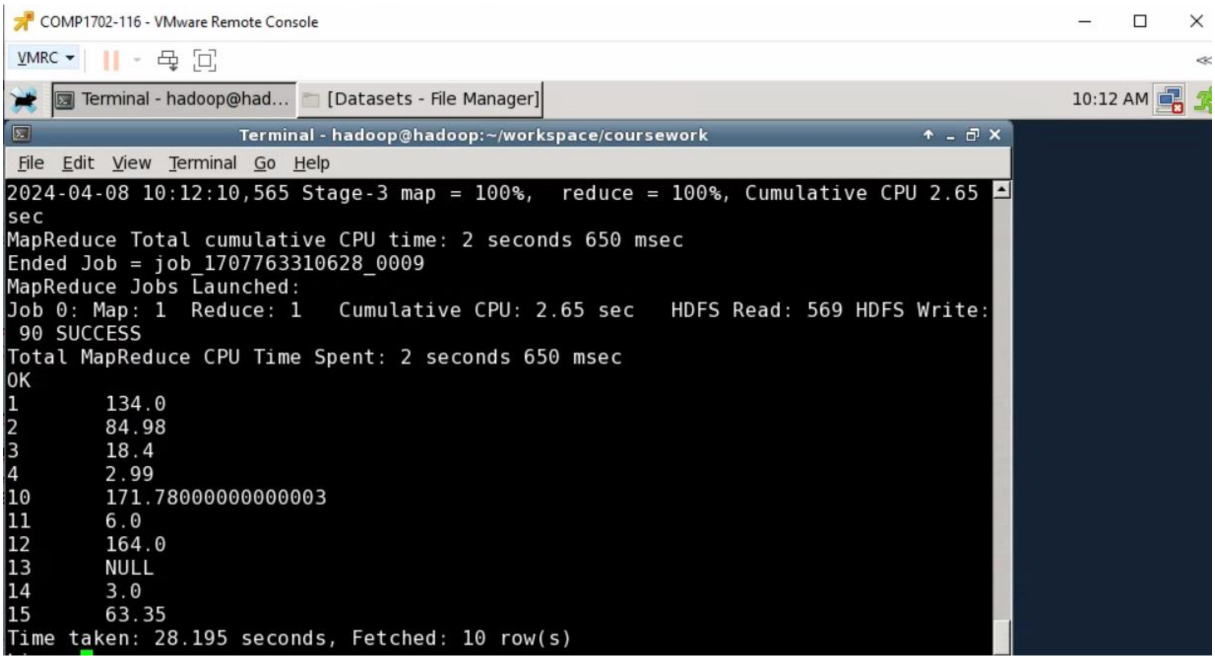
COMP1702-116 - VMware Remote Console
VMRC
Terminal - hadoop@had...
[Datasets - File Manager]
10:23 AM
Terminal - hadoop@hadoop:~/workspace/coursework
File Edit View Terminal Go Help

```
MapReduce Total cumulative CPU time: 2 seconds 10 msec
Ended Job = job_1707763310628_0019
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.01 sec   HDFS Read: 741 HDFS Write:
 135 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 10 msec
OK
1       eyeliner
2       highlighter
3       cuticle oil
4       concealer
5       lipstick
7       blush
8       mascara
9       contour
10      eyeshadow
12      ridge filler
13      top
15      serum
Time taken: 26.5 seconds, Fetched: 12 row(s)
hive>
```

Result of the 3rd querie

3-hive> SELECT Customers.CustomerID, SUM(Products.Price * Sales.Quantity) as TotalSpent FROM Customers JOIN Sales ON Customers.CustomerID = Sales.CustomerID JOIN Products ON Sales.ProductID = Products.ProductID GROUP BY Customers.CustomerID;

This query calculates the total spending of each customer. The GROUP BY clause groups the results by customer ID.

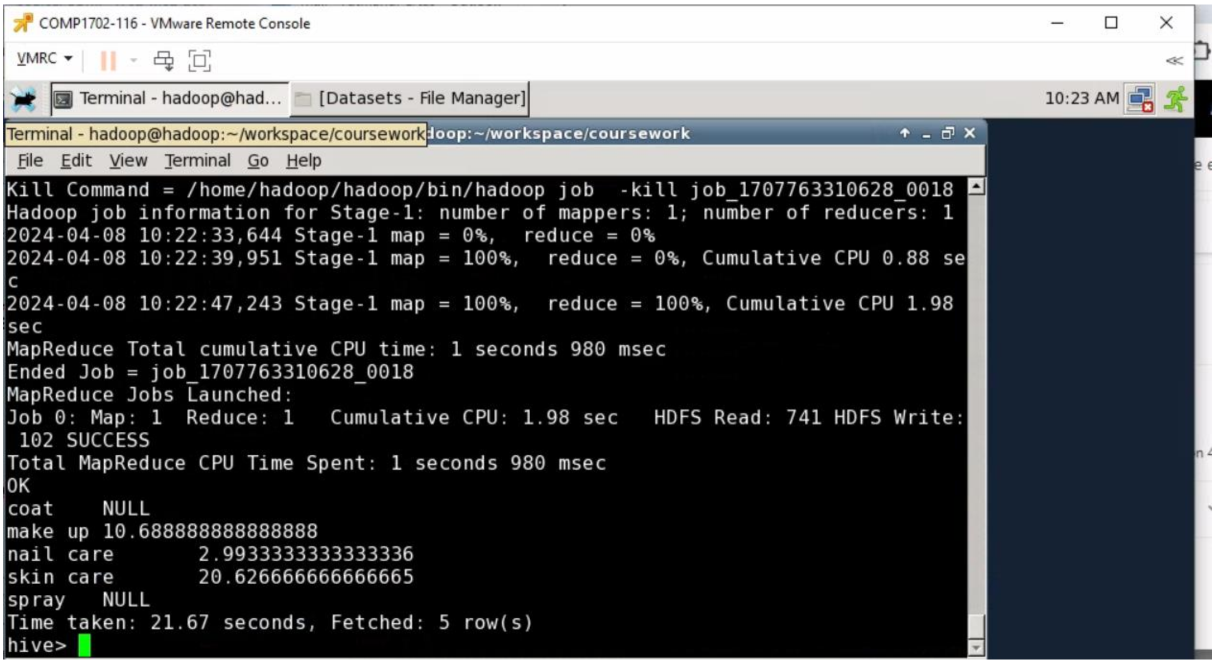

COMP1702-116 - VMware Remote Console
VMRC
Terminal - hadoop@had...
[Datasets - File Manager]
10:12 AM
Terminal - hadoop@hadoop:~/workspace/coursework
File Edit View Terminal Go Help

```
2024-04-08 10:12:10,565 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 2.65
sec
MapReduce Total cumulative CPU time: 2 seconds 650 msec
Ended Job = job_1707763310628_0009
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.65 sec   HDFS Read: 569 HDFS Write:
 90 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 650 msec
OK
1       134.0
2       84.98
3       18.4
4       2.99
10      171.78000000000003
11      6.0
12      164.0
13      NULL
14      3.0
15      63.35
Time taken: 28.195 seconds, Fetched: 10 row(s)
```

Result of the 4th querie

4-hive> SELECT Category, AVG(Price) as AveragePrice FROM Products GROUP BY Category;

This query finds the average price of products for each category.

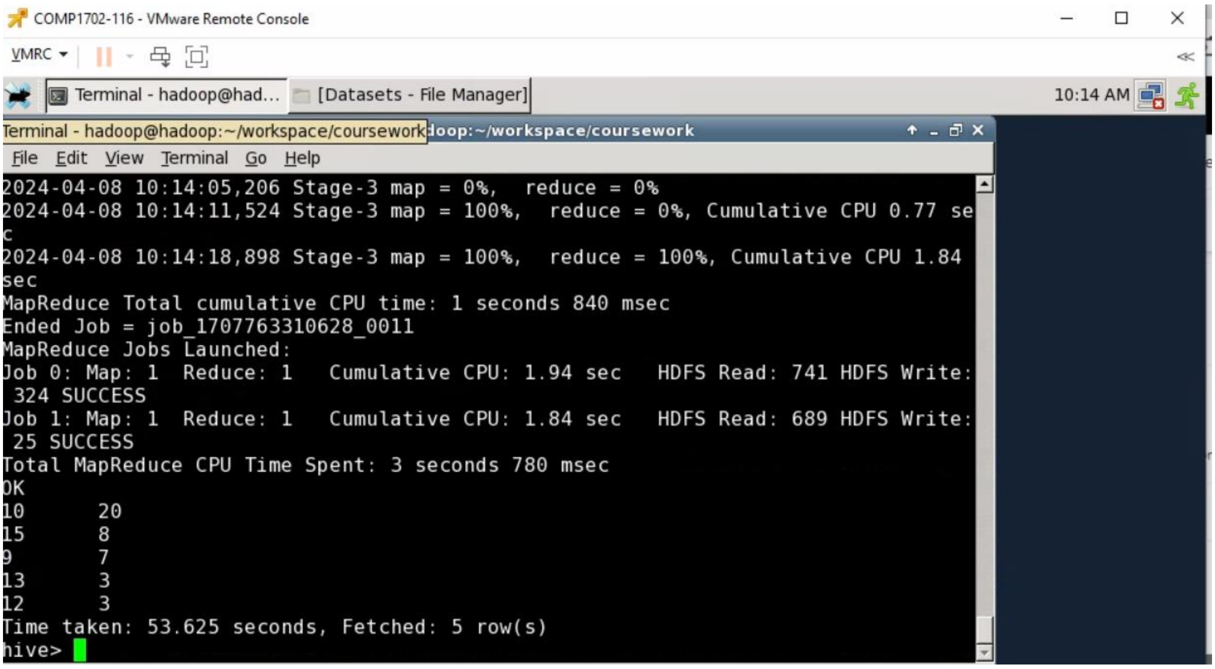

COMP1702-116 - VMware Remote Console
VMRC
Terminal - hadoop@had...
[Datasets - File Manager]
10:23 AM
Terminal - hadoop@hadoop:~/workspace/coursework
File Edit View Terminal Go Help

```
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1707763310628_0018
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-08 10:22:33,644 Stage-1 map = 0%,  reduce = 0%
2024-04-08 10:22:39,951 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.88 se
c
2024-04-08 10:22:47,243 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.98
sec
MapReduce Total cumulative CPU time: 1 seconds 980 msec
Ended Job = job_1707763310628_0018
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.98 sec   HDFS Read: 741 HDFS Write:
 102 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 980 msec
OK
coat    NULL
make up 10.688888888888888
nail care       2.9933333333333336
skin care       20.626666666666665
spray   NULL
Time taken: 21.67 seconds, Fetched: 5 row(s)
hive>
```

Result of the 5th querie

5-hive> SELECT Products.ProductID, SUM(Sales.Quantity) as TotalQuantity FROM Products JOIN Sales ON Products.ProductID = Sales.ProductID GROUP BY Products.ProductID ORDER BY TotalQuantity DESC LIMIT 5;

This query lists the top 5 products by sales quantity.

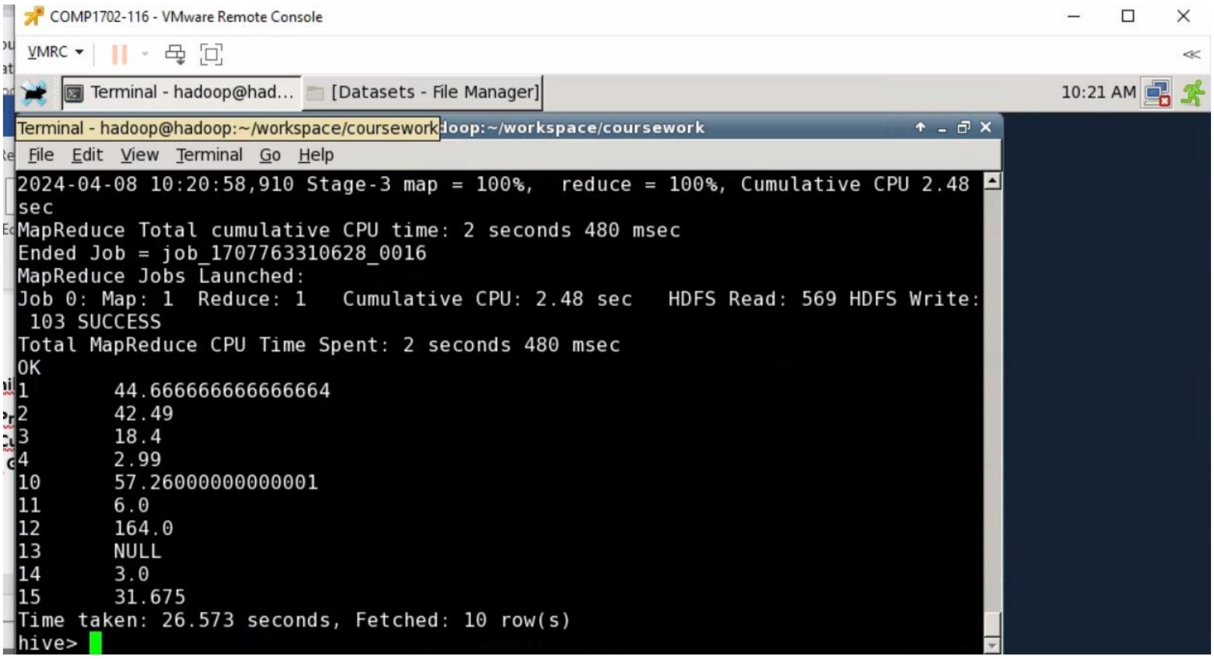

COMP1702-116 - VMware Remote Console
VMRC
Terminal - hadoop@had...
[Datasets - File Manager]
10:14 AM
Terminal - hadoop@hadoop:~/workspace/coursework
File Edit View Terminal Go Help

```
2024-04-08 10:14:05,206 Stage-3 map = 0%,  reduce = 0%
2024-04-08 10:14:11,524 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 0.77 se
c
2024-04-08 10:14:18,898 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 1.84
sec
MapReduce Total cumulative CPU time: 1 seconds 840 msec
Ended Job = job_1707763310628_0011
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.94 sec   HDFS Read: 741 HDFS Write:
 324 SUCCESS
Job 1: Map: 1  Reduce: 1   Cumulative CPU: 1.84 sec   HDFS Read: 689 HDFS Write:
 25 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 780 msec
OK
10      20
15      8
9       7
13      3
12      3
Time taken: 53.625 seconds, Fetched: 5 row(s)
hive>
```

Result of the 6th querie

```
6-hive> SELECT Customers.CustomerID, AVG(Products.Price *
Sales.Quantity) as AvgSpent FROM Customers JOIN Sales ON
Customers.CustomerID = Sales.CustomerID JOIN Products ON
Sales.ProductID = Products.ProductID GROUP BY
Customers.CustomerID;
```

This query calculates the average spending of each customer.



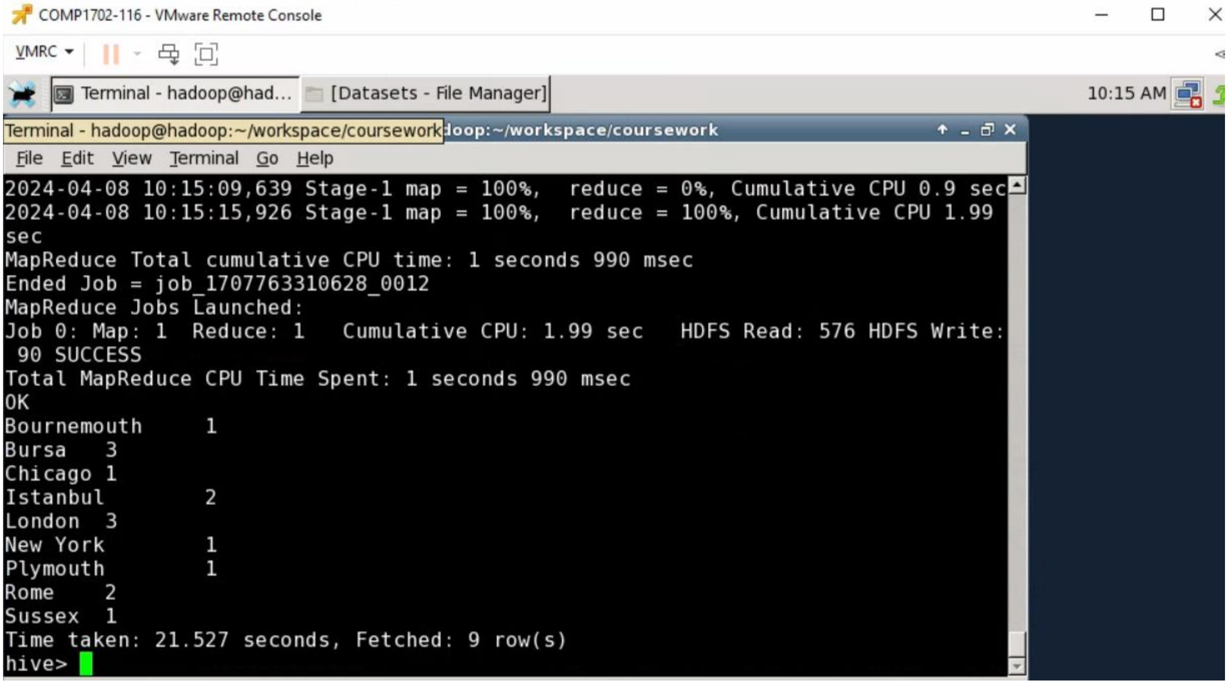
COMP1702-116 - VMware Remote Console
VMRC
Terminal - hadoop@had...
[Datasets - File Manager]
10:21 AM
Terminal - hadoop@hadoop:~/workspace/coursework
File Edit View Terminal Go Help

```
2024-04-08 10:20:58,910 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 2.48
sec
MapReduce Total cumulative CPU time: 2 seconds 480 msec
Ended Job = job_1707763310628_0016
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.48 sec   HDFS Read: 569 HDFS Write:
 103 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 480 msec
OK
1       44.666666666666664
2       42.49
3       18.4
4       2.99
10      57.26000000000001
11      6.0
12      164.0
13      NULL
14      3.0
15      31.675
Time taken: 26.573 seconds, Fetched: 10 row(s)
hive>
```

Result of the 7th querie

7-hive> SELECT Customers.City, COUNT(*) as CustomerCount FROM Customers GROUP BY Customers.City;

This query finds the number of customers in each city.

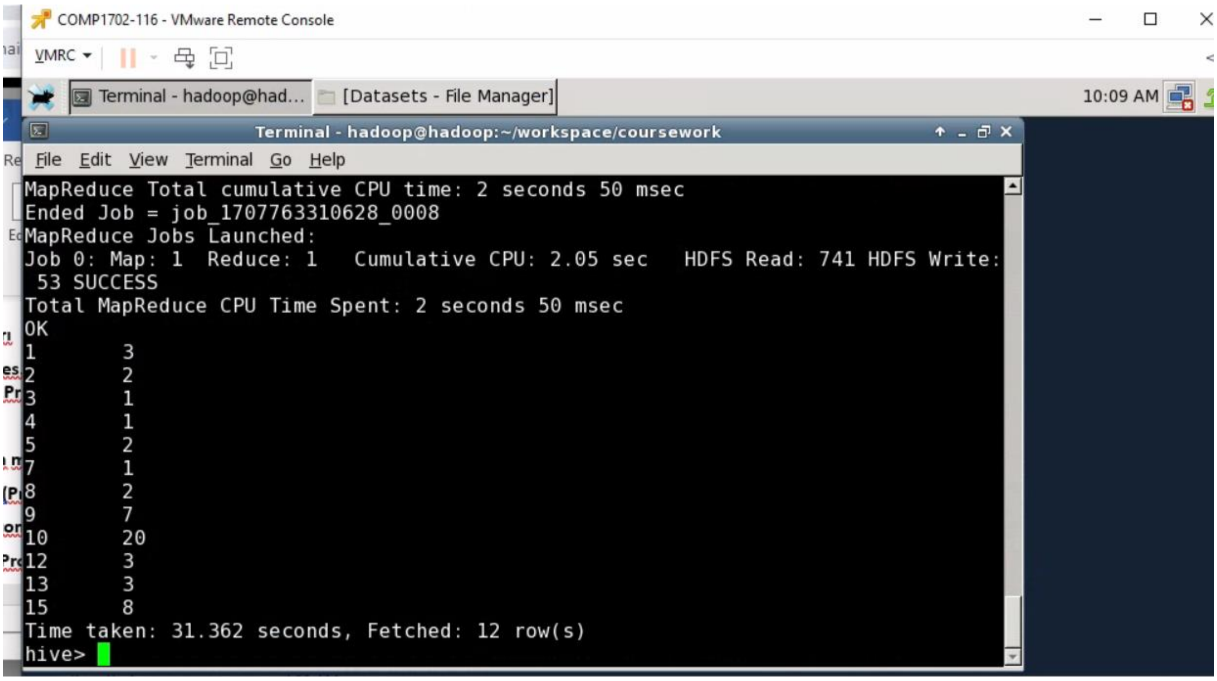

COMP1702-116 - VMware Remote Console
VMRC
Terminal - hadoop@had...
[Datasets - File Manager]
10:15 AM
Terminal - hadoop@hadoop:~/workspace/coursework
File Edit View Terminal Go Help

```
2024-04-08 10:15:09,639 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.9 sec
2024-04-08 10:15:15,926 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.99
sec
MapReduce Total cumulative CPU time: 1 seconds 990 msec
Ended Job = job_1707763310628_0012
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.99 sec   HDFS Read: 576 HDFS Write:
 90 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 990 msec
OK
Bournemouth     1
Bursa   3
Chicago 1
Istanbul        2
London  3
New York        1
Plymouth        1
Rome    2
Sussex  1
Time taken: 21.527 seconds, Fetched: 9 row(s)
hive>
```

Result of the 8th querie

8-hive> SELECT Products.ProductID, SUM(Sales.Quantity) as TotalQuantity FROM Products JOIN Sales ON Products.ProductID = Sales.ProductID GROUP BY Products.ProductID;

This query calculates the total sales quantity for each product.

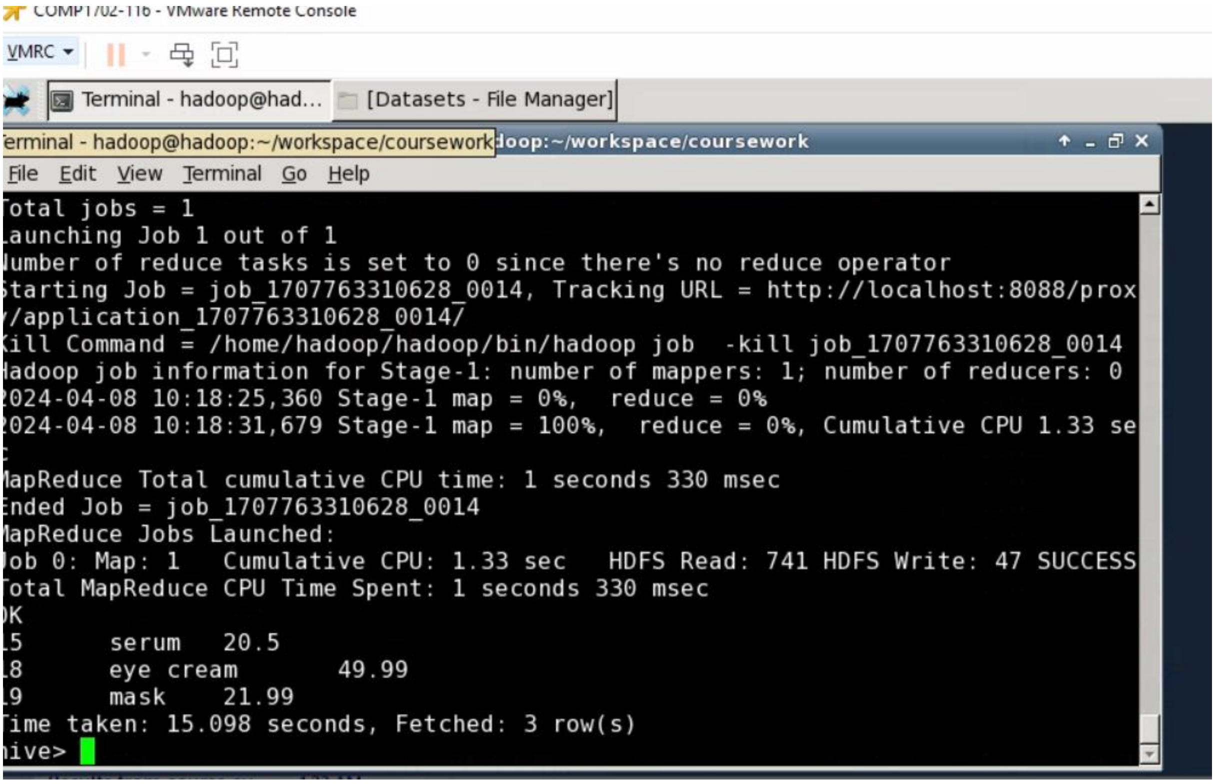

COMP1702-116 - VMware Remote Console
VMRC
Terminal - hadoop@had...
[Datasets - File Manager]
10:09 AM
Terminal - hadoop@hadoop:~/workspace/coursework
File Edit View Terminal Go Help

```
MapReduce Total cumulative CPU time: 2 seconds 50 msec
Ended Job = job_1707763310628_0008
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.05 sec   HDFS Read: 741 HDFS Write:
 53 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 50 msec
OK
1       3
2       2
3       1
4       1
5       2
7       1
8       2
9       7
10      20
12      3
13      3
15      8
Time taken: 31.362 seconds, Fetched: 12 row(s)
hive> 
```

Result of the 9th querie

9-hive>SELECT ProductID, ProductName, Price FROM Products WHERE Price > 20;

This query lists the products that are priced above 20



COMP1702-116 - VMware Remote Console
VMRC
Terminal - hadoop@had...
[Datasets - File Manager]
erminal - hadoop@hadoop:~/workspace/coursework doop:~/workspace/coursework
File Edit View Terminal Go Help

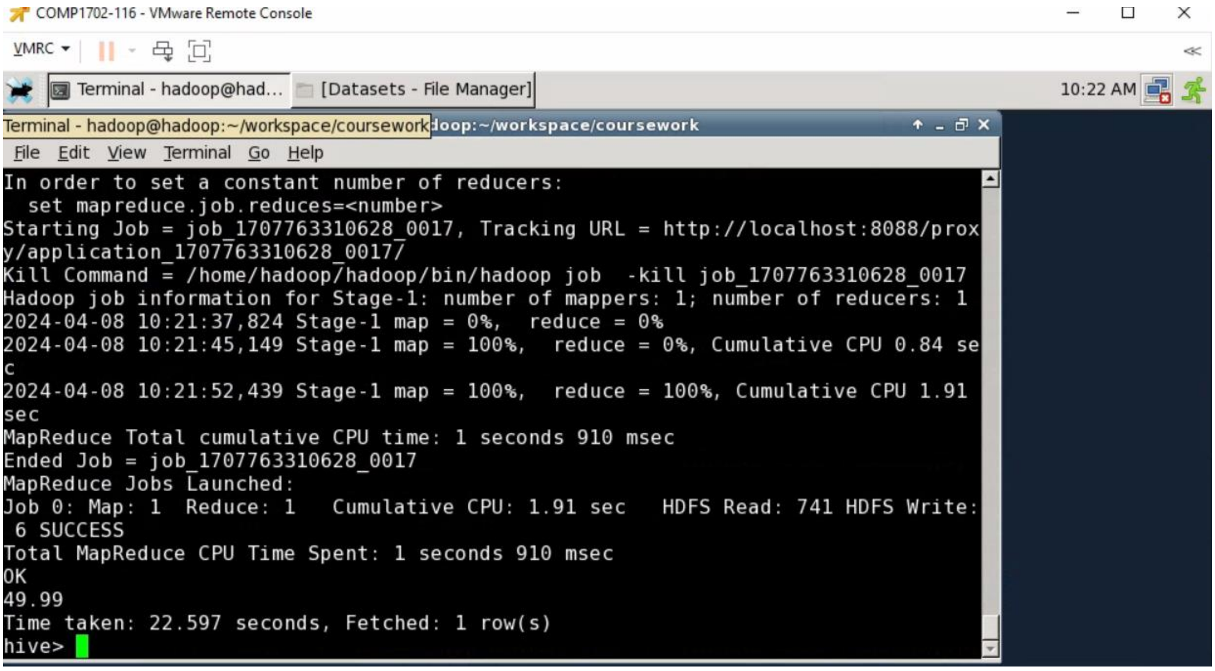
```
otal jobs = 1
aunching Job 1 out of 1
umber of reduce tasks is set to 0 since there's no reduce operator
tarting Job = job_1707763310628_0014, Tracking URL = http://localhost:8088/prox
/application_1707763310628_0014/
ill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1707763310628_0014
adoop job information for Stage-1: number of mappers: 1; number of reducers: 0
024-04-08 10:18:25,360 Stage-1 map = 0%,  reduce = 0%
024-04-08 10:18:31,679 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.33 se


apReduce Total cumulative CPU time: 1 seconds 330 msec
nded Job = job_1707763310628_0014
apReduce Jobs Launched:
ob 0: Map: 1   Cumulative CPU: 1.33 sec   HDFS Read: 741 HDFS Write: 47 SUCCESS
otal MapReduce CPU Time Spent: 1 seconds 330 msec
K
5       serum   20.5
8       eye cream       49.99
9       mask    21.99
ime taken: 15.098 seconds, Fetched: 3 row(s)
ive>
```

Result of the 10th querie

`10-hive> SELECT MAX(Price) FROM Products;`

This query finds the highest price among the prices of products in the Products table.


COMP1702-116 - VMware Remote Console
VMRC
Terminal - hadoop@had...
[Datasets - File Manager]
10:22 AM
Terminal - hadoop@hadoop:~/workspace/coursework
File Edit View Terminal Go Help

```
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1707763310628_0017, Tracking URL = http://localhost:8088/prox
y/application_1707763310628_0017/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1707763310628_0017
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-08 10:21:37,824 Stage-1 map = 0%,  reduce = 0%
2024-04-08 10:21:45,149 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.84 se
c
2024-04-08 10:21:52,439 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.91
sec
MapReduce Total cumulative CPU time: 1 seconds 910 msec
Ended Job = job_1707763310628_0017
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.91 sec   HDFS Read: 741 HDFS Write:
 6 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 910 msec
OK
49.99
Time taken: 22.597 seconds, Fetched: 1 row(s)
hive>
```

TASK-B.4(Output the average number of authors per paper for each year) MapReduce Programming

```
class Mapper {

// defines a HashMap that will store the total number of authors for each year

  private Map<Integer, Integer> authorCountPerYear;

// setup method is called when the mapper object is initialized

  public void setup() {

    authorCountPerYear = new HashMap<>();

  }

// map method is called for each line of the dataset
// merge method is used to update the total author count for each year in the authorCountPerYear HashMap
  public void map(String key, String value) {
    String[] parts = value.split("\\|");
    int year = Integer.parseInt(parts[3]);
    int authorCount = parts[0].split(",").length;
    authorCountPerYear.merge(year, authorCount, Integer::sum);
  }
// cleanup method is called
  public void cleanup(Context context) {
    authorCountPerYear.forEach((year, totalAuthors) -> {
      context.emit(year, new Tuple(totalAuthors, 1));
    });
  }
}


class Reducer {
  public void reduce(Integer year, Iterable<Tuple> values) {
    int totalAuthors = 0;
    int paperCount = 0;
//These lines initialize variables to hold the total number of authors and the number of papers.
    for (Tuple value : values) {
      totalAuthors += value.getFirst();
      paperCount += value.getSecond();
    }
//This loop sums up the total number of authors and the number of papers.
 float averageAuthorsPerPaper = paperCount > 0 ? (float) totalAuthors / paperCount : 0;
    context.emit(year, averageAuthorsPerPaper);
    }
}
```

In the map phase, after taking article information as input, the year information is used as the key, and the number of authors is used as the value for the key-value pair. For the output, the cleanup method sends the total number of authors and the number of articles for each year as a tuple to the reducer. In the reduce phase, a loop is performed over all the tuple values for each year, and the total number of authors and the total number of articles are calculated. Then, the average number of authors per article for each year is calculated and outputted.

Using the in-mapper combiner approach in this code reduces network traffic, as each mapper performs its own aggregation and sends the results to the reducer. This reduces the amount of data sent over the network because each author count and year combination is only sent once. This is important in large data sets where network traffic can be a bottleneck. The in-mapper combiner ensures that intermediate results are stored in memory rather than on disk, which reduces disk read-write operations and lowers I/O costs. Since each mapper aggregates its own data, the amount of data sent to the reducers is reduced, meaning that the reducers have less processing to do and can produce results faster. As the mappers operate independently and perform their own aggregation, this allows for parallelism, helping to process large data sets more quickly. In-mapper combiners can also be scaled as the data set grows by increasing the number of mappers to expand processing capacity.

However, there are potential issues with this code. Since each mapper performs its own aggregation, memory usage can increase, which might lead to memory insufficiency in some cases. In terms of data distribution, if the data set is concentrated around certain years or author counts, some mappers might have a heavier workload than others. This workload imbalance can reduce the overall efficiency of the process.

In conclusion, although this code benefits from the advantages of the in-mapper combiner function in terms of efficiency, it might suffer from reduced efficiency due to memory usage and workload imbalance.

TASK-C Big Data Project Analysis

C1-

A data lake contains data of all types of structures, including raw and unprocessed data, while a data warehouse stores data processed and transformed for a specific purpose, which can then be used as a source for analytical or operational reporting so this makes databases ideal for building more standard forms of BI analysis or serving already-defined business use cases.[1]

Firstly, data lakes are typically built on distributed file systems such as Hadoop or Amazon S3, which can easily store this 200 petabytes of data and scale effortlessly as data volume increases. Since we will be storing various types of data (social media, market data, online news feeds, broker notes, and corporate data), data lakes can accommodate unstructured data like text, images, videos, as well as semi-structured data like JSON, XML, and structured data (SQL tables) all in one place. In data warehouses, we need to define data schemas in advance, and data must conform to these schemas during the loading process (Schema-on-write). In data lakes, we use a schema-on-read approach, where we define our data schema while reading the data. This provides us with a significant advantage in situations where our data changes and varies, as is the case with our data. Data lakes are compatible with big data processing frameworks (Apache Spark, Hadoop) and machine learning libraries, which will enable ABC Investment Bank to analyze market trends more effectively, optimize trading strategies, and better analyze customer portfolios. Data lakes also use lower-cost storage solutions and provide a cost advantage as data increases, which will result in lower costs for our data compared to data warehouses. In summary, considering factors such as managing a large data volume of 200 petabytes, processing various types of data, needing flexible data modeling, requiring advanced analytics capabilities, and cost efficiency, it is logical for ABC Investment Bank Ltd to opt for a data lake solution. This will help the bank maintain its competitive advantage and adapt quickly to market conditions.

C2-

Although Hadoop, i.e. the standard MapReduce, has many strengths, it also has several disadvantages. MapReduce by itself cannot do recursive or repeating jobs.[2]There is also a problem with completely unplanned behavior. All input must be prepared before the job starts, which prevents MapReduce from using network and stream processing use cases.[2] The job initiation framework, such as code copying and scheduling, is another thing that prevents it from running interactive jobs and near real-time queries, also MapReduce cannot perform continuous calculations and queries.[2] The processes of reading data, processing it, and writing the results to disk can introduce significant latency. Running and managing MapReduce jobs for real-time analytics can be more complex and may not provide the desired quick response.

Alternatively, using stream processing frameworks can be more beneficial. Frameworks such as Apache Storm, Apache Flink, or Apache Kafka Streams are designed for low-latency processing, which is crucial for our real-time analytics operations. They can process data in milliseconds, allowing the bank to act quickly based on social media discussions. Like MapReduce, these frameworks can also handle large volumes of data across a distributed cluster and provide fault tolerance to ensure continuous processing. They also support stateful computations, which are necessary for tracking and analyzing trends over time.

In conclusion, although MapReduce is powerful for batch processing, it may not fulfill the real-time performance requirements of ABC Investment Bank. Due to their low latency, scalability, and fault tolerance, using stream processing frameworks like Apache Storm, Apache Flink, or Apache Kafka Streams can enable the bank to perform real-time analysis and make timely trading decisions.

C3-

Using cloud services such as Amazon Web Services, Google Cloud Platform, or Microsoft Azure will not only reduce the bank's infrastructure management burden but also enable automatic scaling of resources according to demand. Cloud computing refers to the processing phase of everything in the cloud. It can also be related to Big Data processing in the cloud. The cloud consists of a number of high-powered servers. These servers may be provided by one or more service providers. In the cloud, it is possible to view large data sets and then make queries faster than on a regular computer[3]. The presence of multiple data centers worldwide from cloud providers will allow the bank to serve users in different locations more closely and ensure low latency. Cloud services can automatically increase and decrease resources in situations like traffic surges, meeting the bank's scalability requirements while optimizing costs. The redundancy and fault tolerance offered by cloud providers in cases of service outages and hardware failures will also be beneficial. For data processing, Hadoop or Spark can be used as they process data in parallel, providing fast analysis and processing. Using a data lake architecture will allow us to store various types of data in a single storage area. From a security perspective, encrypting data both in transit and at rest will be beneficial. Role-based access controls and authentication mechanisms will prevent unauthorized access and enhance security. This hosting strategy, with the scalability of cloud-based solutions, the accessibility of global data centers, the speed of distributed data processing, and the protection of security measures, will meet the global business requirements of ABC Investment Bank Ltd. Additionally, it will allow the bank to easily adapt as data volume increases and business needs change.

References

[1] Microsoft Azure Available at:https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-data-lake#:~:text=While%20a%20data%20lake%20holds,source%20analytic%20or%20operational%20reporting.

[2]- Saeed Shahrivari ,“Beyond Batch Processing: Towards Real-Time and Streaming Big Data “, 2014

[3]- Ajay Yadav ,“ The Role Played by Cloud Computing in Big Data”,2022