Big Data

Assignment

Name: Maitreyi Manish Kulkarni

MSc. Big Data and Business Intelligence

Table of Contents

Task A: Hive Data Warehouse Design

For task A the data warehouse that I have created is of hospitals and patients. In this schema 'hospitals' table stores details about different hospitals like their cities and ratings. The 'doctors' table stores information about doctors working in these hospitals and their respective specializations. Lastly, the 'patients' table stores the patients' records regarding their age, date of admission to these hospitals, and also the doctor responsible for their treatment. This setup of 3 different tables allows for various types of queries and analyses, such as finding patients being treated by particular doctors, hospital ratings and doctors attached to these hospitals, and more. The details of the tables and their columns are as follows:

1) hospitals

    a) hospital_id (INT, Primary Key)
    b) name (STRING)
    c) location (STRING)
    d) rating (FLOAT)

2) doctors

a) doctor_id (INT, Primary Key)
b) name (STRING)
c) specialty (STRING)
d) hospital_id (INT, Foreign Key referencing hospitals)

3) patients

    a) patient_id (INT, Primary Key)
    b) name (STRING)
    c) age (INT)
    d) gender (STRING)
    e) doctor_id (INT, Foreign Key referencing doctors)
    f) admission_date (DATE)
    g) discharge_date (DATE)

Once the data is loaded into comma separated files, it is loaded into Hadoop and then stored into tables using hive queries. The steps for all 3 tables along with screenshot of successful data loading is shown below.

```
create table hospitals(Hospital_id STRING, Hospital_name STRING, City STRING, Rating float) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
```

load data inpath ‘hospitals.csv’ overwrite into table hospitals;
select * from hospitals;

```
hive> create table hospitals(Hospital_id STRING, Hospital_name STRING, City STRI
NG, Rating float) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFI
LE;
OK
Time taken: 1.081 seconds
hive> load data inpath 'hospitals.csv' overwrite into table hospitals;
Loading data to table default.hospitals
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted hdfs://localhost:9000/user/hive/warehouse/hospitals
Table default.hospitals stats: [numFiles=1, numRows=0, totalSize=396, rawDataSiz
e=0]
OK
Time taken: 0.87 seconds
hive> select * from hospitals;
OK
H001    City General Hospital   New York        4.2
H002    County Hospital Los Angeles     3.9
H003    Central Hospital        Chicago 4.5
H004    Community Hospital      Houston 3.7
H005    University Medical Center       Phoenix 4.8
H006    Memorial Hospital       Philadelphia    4.4
H007    Sunset Medical Center   San Antonio     4.1
H008    Riverside Community Hospital    San Diego       4.5
H009    Eastside Medical Center Dallas  4.6
H010    Midtown Hospital        San Jose        3.5
Time taken: 0.285 seconds, Fetched: 10 row(s)
```

create table doctors(Doctor_id STRING, Doctor_name STRING, Speciality STRING, Hospital_id STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

load data inpath ‘doctors.csv’ overwrite into table doctors;
select * from doctors;

```
hive> create table doctors(Doctor_id STRING, Doctor_name STRING, Speciality STRING, Hospital_id STRING) RO
W FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;
OK
Time taken: 0.28 seconds
hive> load data inpath 'doctors.csv' overwrite into table doctors;
Loading data to table default.doctors
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted hdfs://localhost:9000/user/hive/warehouse/doctors
Table default.doctors stats: [numFiles=1, numRows=0, totalSize=494, rawDataSize=0]
OK
Time taken: 0.567 seconds
hive> select * from doctors;
OK
d1      Dr. Smith       Cardiologist    H001
d2      Dr. Johnson     Pediatrician    H001
d3      Dr. Leah        Neurologist     H002
d4      Dr. Garcia      Surgeon H002
d5      Dr. Patel       Oncologist      H003
d6      Dr. Thompson    Orthopedic Surgeon      H006
d7      Dr. White       Dermatologist   H007
d8      Dr. Martin      Psychiatrist    H008
d9      Dr. Rodriguez   Endocrinologist H009
d10     Dr. Carter      Gynecologist    H010
d11     Dr. Ethans      Ophthalmologist H010
d12     Dr. Harris      Rheumatologist  H005
d13     Dr. Turing      Anesthesiologist        H007
d14     Dr. Baker       Allergist       H001
d15     Dr.Sharma       Pulmonologist   H006
Time taken: 0.086 seconds, Fetched: 15 row(s)
hive>
```

create table patients(Patient_id INT, Patient_name STRING, Age INT, Gender STRING, Doctor_id STRING, Admission_date DATE, Discharge_date DATE) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

load data inpath ‘patients.csv’ overwrite into table patients;
select * from patients;

```
hive> create table patients(Patient_id INT, Patient_name STRING, Age INT, Gender STRING, Doctor_id STRING,
 Admission_date DATE, Discharge_date DATE) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFIL
E;
OK
Time taken: 0.111 seconds
hive> load data inpath 'patients.csv' overwrite into table patients;
Loading data to table default.patients
rmr: DEPRECATED: Please use 'rm -r' instead.
Deleted hdfs://localhost:9000/user/hive/warehouse/patients
Table default.patients stats: [numFiles=1, numRows=0, totalSize=1771, rawDataSize=0]
OK
Time taken: 0.295 seconds
hive> select * from patients;
OK
1       David Joy       35      Male    d1      2024-04-01      2024-04-05
2       Austin Ames     45      Female  d2      2024-04-02      2024-04-07
3       Miguel Henson   50      Male    d3      2024-04-03      2024-04-06
4       Emilia Watson   28      Female  d4      2024-03-14      2024-03-18
5       Jack Garret     60      Male    d1      2024-02-15      2024-02-19
6       Sarah Wilford   42      Female  d6      2023-12-06      2023-12-10
7       Robert Moore    55      Male    d12     2024-03-27      2024-04-04
8       Karen Turner    30      Female  d8      2024-02-08      2024-02-12
9       Thomas Adams    65      Male    d9      2024-01-09      2024-01-13
10      Jennifer Clark  38      Female  d10     2023-04-10      2023-04-14
11      Steven Taylor   47      Male    d11     2023-04-11      2023-04-15
12      Audrey Reynolds 29      Female  d12     2023-02-12      2023-02-16
13      Martin Ezra     60      Male    d13     2023-01-13      2023-01-22
14      Rachel Evans    35      Female  d14     2023-04-14      2023-04-18
15      Daniel Miller   50      Male    d10     2023-07-15      2023-07-19
16      Lisa Brody      33      Female  d11     2023-06-06      2023-06-12
17      Arthur Williamson       70      Male    d12     2023-06-17      2023-06-26
18      Melinda Parker  25      Female  d13     2023-10-18      2023-10-22
19      Liam Price      52      Male    d14     2023-05-19      2023-05-23
20      Laura Hart      45      Female  d1      2023-04-20      2023-04-27
21      Christopher Fisher      55      Male    d2      2022-01-09      2022-01-15
22      Molly Sandos    28      Female  d3      2022-02-10      2022-02-20
23      Marcus Cole     40      Male    d4      2022-03-15      2022-03-24
```

```
24      Carol Hilton    35      Female  d5      2022-04-20      2022-04-26
25      Nicholas Warren 60      Male    d15     2022-05-25      2022-06-02
26      Charlotte Hexley        25      Female  d7      2022-06-27      2022-07-30
27      Joshua Lee      45      Male    d8      2022-08-05      2022-08-08
28      Anna Rogers     30      Female  d9      2022-09-10      2022-09-12
29      Mason Kirby     50      Male    d10     2022-10-15      2022-10-25
30      Sabrina Fellows 35      Female  d11     2022-11-20      2022-11-29
31      Oliver Green    27      Male    d12     2021-01-10      2021-01-19
32      Regina Adams    33      Female  d3      2021-03-11      2021-03-20
33      Billy Mckenzie  48      Male    d4      2021-08-12      2021-08-14
34      Ella Tyson      25      Female  d15     2021-10-13      2021-10-18
35      Noah Murrey     55      Male    d6      2021-12-14      2021-12-19
Time taken: 0.052 seconds, Fetched: 35 row(s)
```

Implementation and queries-

A data warehouse of hospitals, patients and doctors will be highly beneficial in the healthcare industry. Healthcare organisations can analyse patient demographics, diagnoses, treatments and patterns between them. This will help in improving patient care and optimising allocation of doctors in various hospitals. All records of patients can be maintained efficiently. Innovative approaches for mass treatments can be thought of.

1. Suppose, we want to find details of patients and doctors. This query demonstrates obtaining patient's name, gender, age, and their doctor with the specialisation. The join is performed on doctor_id column which is primary key in doctors table and foreign key in patients table. This helps in knowing what each patient is being treated for and by whom.

```
select patients.patient_name, patients.gender, patients.age,
doctors.doctor_name, doctors.speciality from patients
join doctors on patients.doctor_id=doctors.doctor_id;
```

```
hive> select patients.patient_name, patients.gender, patients.age, doctors.doctor_name, doctors.speciality
    > from patients
    > join doctors on patients.doctor_id=doctors.doctor_id;
Total jobs = 1
24/04/07 18:26:01 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... us
ing builtin-java classes where applicable
24/04/07 18:26:02 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_18-25-58_604_34247141590644433
02-1/-local-10006/jobconf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.r
etry.interval;  Ignoring.
24/04/07 18:26:02 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_18-25-58_604_34247141590644433
02-1/-local-10006/jobconf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.a
ttempts;  Ignoring.
Execution log at: /tmp/hadoop/hadoop_20240407182525_4d8edd72-2eed-43c4-a148-a026f99138f1.log
2024-04-07 06:26:03     Starting to launch local task to process map join;      maximum memory = 518979584
2024-04-07 06:26:03     Dump the side-table into file: file:/tmp/hadoop/hive_2024-04-07_18-25-58_604_34247
14159064443302-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile01--.hashtable
2024-04-07 06:26:03     Uploaded 1 File to: file:/tmp/hadoop/hive_2024-04-07_18-25-58_604_3424714159064443
302-1/-local-10003/HashTable-Stage-3/MapJoin-mapfile01--.hashtable (952 bytes)
2024-04-07 06:26:03     End of local task; Time Taken: 0.958 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1712502557830_0001, Tracking URL = http://localhost:8088/proxy/application_171250255783
0_0001/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1712502557830_0001
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2024-04-07 18:26:13,783 Stage-3 map = 0%,  reduce = 0%
2024-04-07 18:26:21,517 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 1.14 sec
MapReduce Total cumulative CPU time: 1 seconds 140 msec
Ended Job = job_1712502557830_0001
MapReduce Jobs Launched:
Job 0: Map: 1   Cumulative CPU: 1.14 sec   HDFS Read: 1988 HDFS Write: 1641 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 140 msec
OK
David Joy       Male    35      Dr. Smith       Cardiologist
Austin Ames     Female  45      Dr. Johnson     Pediatrician
Miguel Henson   Male    50      Dr. Leah        Neurologist
```

```
Emilia Watson   Female  28      Dr. Garcia      Surgeon
Jack Garret     Male    60      Dr. Smith       Cardiologist
Sarah Wilford   Female  42      Dr. Thompson    Orthopedic Surgeon
Robert Moore    Male    55      Dr. Harris      Rheumatologist
Karen Turner    Female  30      Dr. Martin      Psychiatrist
Thomas Adams    Male    65      Dr. Rodriguez   Endocrinologist
Jennifer Clark  Female  38      Dr. Carter      Gynecologist
Steven Taylor   Male    47      Dr. Ethans      Ophthalmologist
Audrey Reynolds Female  29      Dr. Harris      Rheumatologist
Martin Ezra     Male    60      Dr. Turing      Anesthesiologist
Rachel Evans    Female  35      Dr. Baker       Allergist
Daniel Miller   Male    50      Dr. Carter      Gynecologist
Lisa Brody      Female  33      Dr. Ethans      Ophthalmologist
Arthur Williamson       Male    70      Dr. Harris      Rheumatologist
Melinda Parker  Female  25      Dr. Turing      Anesthesiologist
Liam Price      Male    52      Dr. Baker       Allergist
Laura Hart      Female  45      Dr. Smith       Cardiologist
Christopher Fisher      Male    55      Dr. Johnson     Pediatrician
Molly Sandos    Female  28      Dr. Leah        Neurologist
Marcus Cole     Male    40      Dr. Garcia      Surgeon
Carol Hilton    Female  35      Dr. Patel       Oncologist
Nicholas Warren Male    60      Dr.Sharma       Pulmonologist
Charlotte Hexley        Female  25      Dr. White       Dermatologist
Joshua Lee      Male    45      Dr. Martin      Psychiatrist
Anna Rogers     Female  30      Dr. Rodriguez   Endocrinologist
Mason Kirby     Male    50      Dr. Carter      Gynecologist
Sabrina Fellows Female  35      Dr. Ethans      Ophthalmologist
Oliver Green    Male    27      Dr. Harris      Rheumatologist
Regina Adams    Female  33      Dr. Leah        Neurologist
Billy Mckenzie  Male    48      Dr. Garcia      Surgeon
Ella Tyson      Female  25      Dr.Sharma       Pulmonologist
Noah Murrey     Male    55      Dr. Thompson    Orthopedic Surgeon
Time taken: 24.021 seconds, Fetched: 35 row(s)
hive>
```

2. Suppose we want to find out which hospitals have the minimum number of doctors so as to allocate them properly. This query shows how to get information about how many doctors are attached in every hospital by selecting name of the hospital and count of doctors. The join is performed on hospital_id column which is a primary key in hospitals table and foreign key in doctors table. The count is grouped by name of hospitals.

```
select hospitals.hospital_name, count(doctors.doctor_id) as num_doctors
from hospitals join doctors on hospitals.hospital_id = doctors.hospital_id
group by hospitals.hospital_name;
```

```
hive> select hospitals.hospital_name, count(doctors.doctor_id) as num_doctors from hospitals
    > join doctors on hospitals.hospital_id = doctors.hospital_id group by hospitals.hospital_name;
Total jobs = 1
24/04/07 19:06:31 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java clas
ses where applicable
24/04/07 19:06:31 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_19-06-28_801_8085218411062808936-1/-local-10007/job
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
24/04/07 19:06:31 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_19-06-28_801_8085218411062808936-1/-local-10007/job
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
Execution log at: /tmp/hadoop/hadoop_20240407190606_17734da0-1b7f-42c2-a57f-ff265a0178af.log
2024-04-07 07:06:32     Starting to launch local task to process map join;      maximum memory = 518979584
2024-04-07 07:06:33     Dump the side-table into file: file:/tmp/hadoop/hive_2024-04-07_19-06-28_801_8085218411062808936-1/-loc
al-10004/HashTable-Stage-2/MapJoin-mapfile30--.hashtable
2024-04-07 07:06:33     Uploaded 1 File to: file:/tmp/hadoop/hive_2024-04-07_19-06-28_801_8085218411062808936-1/-local-10004/Ha
shTable-Stage-2/MapJoin-mapfile30--.hashtable (703 bytes)
2024-04-07 07:06:33     End of local task; Time Taken: 1.059 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1712502557830_0005, Tracking URL = http://localhost:8088/proxy/application_1712502557830_0005/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1712502557830_0005
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2024-04-07 19:06:40,709 Stage-2 map = 0%,  reduce = 0%
2024-04-07 19:06:47,038 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.79 sec
2024-04-07 19:06:53,371 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 1.95 sec
MapReduce Total cumulative CPU time: 1 seconds 950 msec
Ended Job = job_1712502557830_0005
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.95 sec   HDFS Read: 709 HDFS Write: 209 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 950 msec
OK
Central Hospital        1
City General Hospital   3
```

```
Total MapReduce CPU Time Spent: 1 seconds 950 msec
OK
Central Hospital        1
City General Hospital   3
County Hospital 2
Eastside Medical Center 1
Memorial Hospital       2
Midtown Hospital        2
Riverside Community Hospital    1
Sunset Medical Center   2
University Medical Center       1
Time taken: 25.716 seconds, Fetched: 9 row(s)
hive>
```

3. Suppose we want to know number of patients admitted to any of the hospitals in a year-wise manner, this query can be used. It extracts the year from the entire date of admission and then fetches the count of patients which are then grouped by each year.

```
select year(admission_date) as admission_year, count(*) as no_of_patients_ad
from patients
group by year(patients.admission_date);
```

```
hive> select year(admission_date) as admission_year, count(*) as no_of_patients_ad
    > from patients
    > group by year(patients.admission_date);
FAILED: SemanticException [Error 10025]: Line 1:7 Expression not in GROUP BY key 'admission_date'
hive> select year(patients.admission_date) as admission_year, count(*) as no_of_patients_ad
    > from patients
    > group by year(patients.admission_date);
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1712502557830_0002, Tracking URL = http://localhost:8088/proxy/application_171250255783
0_0002/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1712502557830_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-07 18:43:40,177 Stage-1 map = 0%,  reduce = 0%
2024-04-07 18:43:46,645 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.29 sec
2024-04-07 18:43:54,173 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.51 sec
MapReduce Total cumulative CPU time: 2 seconds 510 msec
Ended Job = job_1712502557830_0002
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.51 sec   HDFS Read: 1988 HDFS Write: 30 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 510 msec
OK
2021    5
2022    10
2023    12
2024    8
Time taken: 23.504 seconds, Fetched: 4 row(s)
hive>
```

4. Suppose the details of only female patients are required. This includes their name and age as well as the gender and also outputs their treatment, name of their doctor and the hospital they are admitted in. The name, age and gender of the patients is selected along with name of their doctors and hospitals. The 'where' clause puts a condition on the gender and 'join' joins the common columns.

```
select patients.patient_name, patients.age, patients.gender,
doctors.doctor_name, doctors.speciality, hospitals.hospital_name
from patients
join doctors on patients.doctor_id=doctors.doctor_id
join hospitals on doctors.hospital_id=hospitals.hospital_id
where patients.gender='Female';
```

```
hive> select patients.patient_name, patients.age, patients.gender, doctors.doctor_name, doctors.speciality, hospitals.hospital_
name
    > from patients
    > join doctors on patients.doctor_id=doctors.doctor_id
    > join hospitals on doctors.hospital_id=hospitals.hospital_id
    > where patients.gender='Female';
Total jobs = 1
24/04/07 19:48:02 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java clas
ses where applicable
24/04/07 19:48:02 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_19-47-59_064_2961094440729014847-1/-local-10007/job
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
24/04/07 19:48:02 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_19-47-59_064_2961094440729014847-1/-local-10007/job
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
Execution log at: /tmp/hadoop/hadoop_20240407194747_8d5570b9-3a50-446b-8137-f3c032df1d08.log
2024-04-07 07:48:03     Starting to launch local task to process map join;      maximum memory = 518979584
2024-04-07 07:48:04     Dump the side-table into file: file:/tmp/hadoop/hive_2024-04-07_19-47-59_064_2961094440729014847-1/-loc
al-10004/HashTable-Stage-4/MapJoin-mapfile91--.hashtable
2024-04-07 07:48:04     Uploaded 1 File to: file:/tmp/hadoop/hive_2024-04-07_19-47-59_064_2961094440729014847-1/-local-10004/Ha
shTable-Stage-4/MapJoin-mapfile91--.hashtable (703 bytes)
2024-04-07 07:48:04     Dump the side-table into file: file:/tmp/hadoop/hive_2024-04-07_19-47-59_064_2961094440729014847-1/-loc
al-10004/HashTable-Stage-4/MapJoin-mapfile101--.hashtable
2024-04-07 07:48:04     Uploaded 1 File to: file:/tmp/hadoop/hive_2024-04-07_19-47-59_064_2961094440729014847-1/-local-10004/Ha
shTable-Stage-4/MapJoin-mapfile101--.hashtable (1027 bytes)
2024-04-07 07:48:04     End of local task; Time Taken: 1.117 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1712502557830_0011, Tracking URL = http://localhost:8088/proxy/application_1712502557830_0011/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1712502557830_0011
Hadoop job information for Stage-4: number of mappers: 1; number of reducers: 0
2024-04-07 19:48:12,183 Stage-4 map = 0%,  reduce = 0%
2024-04-07 19:48:19,645 Stage-4 map = 100%,  reduce = 0%, Cumulative CPU 1.34 sec
MapReduce Total cumulative CPU time: 1 seconds 340 msec
Ended Job = job_1712502557830_0011
MapReduce Jobs Launched:
Job 0: Map: 1   Cumulative CPU: 1.34 sec   HDFS Read: 1988 HDFS Write: 1156 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 340 msec
OK
Austin Ames     45      Female  Dr. Johnson     Pediatrician    City General Hospital
Emilia Watson   28      Female  Dr. Garcia      Surgeon County Hospital
Sarah Wilford   42      Female  Dr. Thompson    Orthopedic Surgeon      Memorial Hospital
```

```
Job 0: Map: 1   Cumulative CPU: 1.34 sec   HDFS Read: 1988 HDFS Write: 1156 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 340 msec
OK
Austin Ames     45      Female  Dr. Johnson     Pediatrician    City General Hospital
Emilia Watson   28      Female  Dr. Garcia      Surgeon County Hospital
Sarah Wilford   42      Female  Dr. Thompson    Orthopedic Surgeon      Memorial Hospital
Karen Turner    30      Female  Dr. Martin      Psychiatrist    Riverside Community Hospital
Jennifer Clark  38      Female  Dr. Carter      Gynecologist    Midtown Hospital
Audrey Reynolds 29      Female  Dr. Harris      Rheumatologist  University Medical Center
Rachel Evans    35      Female  Dr. Baker       Allergist       City General Hospital
Lisa Brody      33      Female  Dr. Ethans      Ophthalmologist Midtown Hospital
Melinda Parker  25      Female  Dr. Turing      Anesthesiologist        Sunset Medical Center
Laura Hart      45      Female  Dr. Smith       Cardiologist    City General Hospital
Molly Sandos    28      Female  Dr. Leah        Neurologist     County Hospital
Carol Hilton    35      Female  Dr. Patel       Oncologist      Central Hospital
Charlotte Hexley        25      Female  Dr. White       Dermatologist   Sunset Medical Center
Anna Rogers     30      Female  Dr. Rodriguez   Endocrinologist Eastside Medical Center
Sabrina Fellows 35      Female  Dr. Ethans      Ophthalmologist Midtown Hospital
Regina Adams    33      Female  Dr. Leah        Neurologist     County Hospital
Ella Tyson      25      Female  Dr.Sharma       Pulmonologist   Memorial Hospital
Time taken: 21.642 seconds, Fetched: 17 row(s)
hive>
```

5. There are many patients who are being treated by the same doctor. In order to find the total number of patients treated by each doctor, join can be performed on the doctor_id column in patients and doctors table. And the names of the doctors are grouped using ‘group by’.

```
select doctors.doctor_name, count(patients.patient_id) as patients_treated
from doctors
join patients on doctors.doctor_id=patients.doctor_id
group by doctors.doctor_name;
```

```
hive> select doctors.doctor_name, count(patients.patient_id) as patients_treated from doctors
    > join patients on doctors.doctor_id=patients.doctor_id
    > group by doctors.doctor_name;
Total jobs = 1
24/04/07 19:29:18 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java clas
ses where applicable
24/04/07 19:29:18 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_19-29-15_637_8300658560223710288-1/-local-10007/job
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
24/04/07 19:29:18 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_19-29-15_637_8300658560223710288-1/-local-10007/job
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
Execution log at: /tmp/hadoop/hadoop_20240407192929_88b6a018-c3ff-4324-a2b3-8282ed49831a.log
2024-04-07 07:29:19     Starting to launch local task to process map join;      maximum memory = 518979584
2024-04-07 07:29:20     Dump the side-table into file: file:/tmp/hadoop/hive_2024-04-07_19-29-15_637_8300658560223710288-1/-loc
al-10004/HashTable-Stage-2/MapJoin-mapfile60--.hashtable
2024-04-07 07:29:20     Uploaded 1 File to: file:/tmp/hadoop/hive_2024-04-07_19-29-15_637_8300658560223710288-1/-local-10004/Ha
shTable-Stage-2/MapJoin-mapfile60--.hashtable (748 bytes)
2024-04-07 07:29:20     End of local task; Time Taken: 0.925 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1712502557830_0009, Tracking URL = http://localhost:8088/proxy/application_1712502557830_0009/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1712502557830_0009
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2024-04-07 19:29:27,260 Stage-2 map = 0%,  reduce = 0%
2024-04-07 19:29:33,498 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.91 sec
2024-04-07 19:29:40,754 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 2.0 sec
MapReduce Total cumulative CPU time: 2 seconds 0 msec
Ended Job = job_1712502557830_0009
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.0 sec   HDFS Read: 1988 HDFS Write: 194 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 0 msec
OK
Dr. Baker       2
Dr. Carter      3
Dr. Ethans      3
```

```
OK
Dr. Baker       2
Dr. Carter      3
Dr. Ethans      3
Dr. Garcia      3
Dr. Harris      4
Dr. Johnson     2
Dr. Leah        3
Dr. Martin      2
Dr. Patel       1
Dr. Rodriguez   2
Dr. Smith       3
Dr. Thompson    2
Dr. Turing      2
Dr. White       1
Dr.Sharma       2
Time taken: 26.243 seconds, Fetched: 15 row(s)
hive>
```

6. To find out from all records how many are males and how many are females can be useful in knowing about patient demographics. This query gives a gender-wise count from total patients using ‘count’ and ‘group by’.

```
select patients.gender, count(*) as no_of_patients from patients
group by patients.gender;
```

```
hive> select patients.gender, count(*) as no_of_patients from patients
    > group by patients.gender;
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1712502557830_0007, Tracking URL = http://localhost:8088/proxy/application_1712502557830_0007/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1712502557830_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2024-04-07 19:13:25,178 Stage-1 map = 0%,  reduce = 0%
2024-04-07 19:13:31,493 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.91 sec
2024-04-07 19:13:38,787 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.07 sec
MapReduce Total cumulative CPU time: 2 seconds 70 msec
Ended Job = job_1712502557830_0007
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 2.07 sec   HDFS Read: 1988 HDFS Write: 18 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 70 msec
OK
Female  17
Male    18
Time taken: 21.327 seconds, Fetched: 2 row(s)
hive>
```

7. Suppose we wish to know about hospitals and how their performance has been, how many patients are being admitted on an average. This can also help in knowing which hospitals need attention, we can execute this query. It shows the average ratings of hospitals and the total count of patients treated. Join is performed on all common columns and it is grouped by name of hospitals.

```
select hospitals.hospital_name, avg(hospitals.rating),
count(patients.patient_id) as patients_treated
from hospitals
join doctors on hospitals.hospital_id=doctors.hospital_id
join patients on doctors.doctor_id=patients.doctor_id
group by hospitals.hospital_name;
```

```
hive> select hospitals.hospital_name, avg(hospitals.rating), count(patients.patient_id) as patients_treated
    > from hospitals
    > join doctors on hospitals.hospital_id=doctors.hospital_id
    > join patients on doctors.doctor_id=patients.doctor_id
    > group by hospitals.hospital_name;
Total jobs = 1
24/04/07 19:20:00 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java clas
ses where applicable
24/04/07 19:20:00 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_19-19-58_044_1505879741381506767-1/-local-10008/job
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
24/04/07 19:20:00 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_19-19-58_044_1505879741381506767-1/-local-10008/job
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
Execution log at: /tmp/hadoop/hadoop_20240407191919_88d0e44d-7c32-48a6-a8be-12b8dafd9ddc.log
2024-04-07 07:20:01     Starting to launch local task to process map join;      maximum memory = 518979584
2024-04-07 07:20:02     Dump the side-table into file: file:/tmp/hadoop/hive_2024-04-07_19-19-58_044_1505879741381506767-1/-loc
al-10005/HashTable-Stage-3/MapJoin-mapfile50--.hashtable
2024-04-07 07:20:02     Uploaded 1 File to: file:/tmp/hadoop/hive_2024-04-07_19-19-58_044_1505879741381506767-1/-local-10005/Ha
shTable-Stage-3/MapJoin-mapfile50--.hashtable (743 bytes)
2024-04-07 07:20:02     Dump the side-table into file: file:/tmp/hadoop/hive_2024-04-07_19-19-58_044_1505879741381506767-1/-loc
al-10005/HashTable-Stage-3/MapJoin-mapfile41--.hashtable
2024-04-07 07:20:02     Uploaded 1 File to: file:/tmp/hadoop/hive_2024-04-07_19-19-58_044_1505879741381506767-1/-local-10005/Ha
shTable-Stage-3/MapJoin-mapfile41--.hashtable (719 bytes)
2024-04-07 07:20:02     End of local task; Time Taken: 0.961 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1712502557830_0008, Tracking URL = http://localhost:8088/proxy/application_1712502557830_0008/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1712502557830_0008
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 1
2024-04-07 19:20:09,536 Stage-3 map = 0%,  reduce = 0%
2024-04-07 19:20:14,870 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 0.9 sec
2024-04-07 19:20:22,110 Stage-3 map = 100%,  reduce = 100%, Cumulative CPU 1.97 sec
MapReduce Total cumulative CPU time: 1 seconds 970 msec
```

```
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.97 sec   HDFS Read: 709 HDFS Write: 330 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 970 msec
OK
Central Hospital        4.5     1
City General Hospital   4.199999809265137       7
County Hospital 3.9000000953674316      6
Eastside Medical Center 4.599999904632568       2
Memorial Hospital       4.400000095367432       4
Midtown Hospital        3.5     6
Riverside Community Hospital    4.5     2
Sunset Medical Center   4.099999904632568       3
University Medical Center       4.800000190734863       4
Time taken: 25.209 seconds, Fetched: 9 row(s)
hive>
```

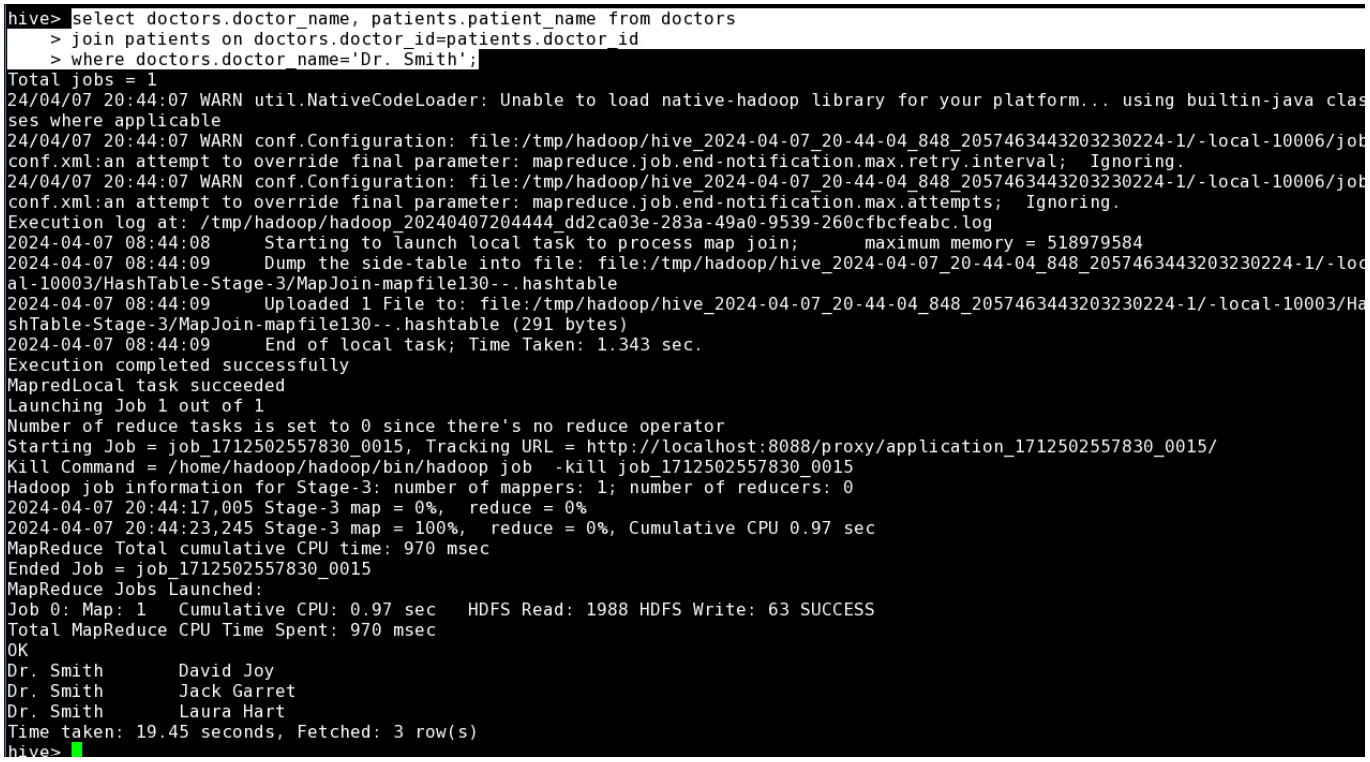
8. For instance, if there a need to obtain information of the patients who were admitted after a specific time and the records are needed only of a specific hospital the query will show. Here, only patients admitted after 1st January 2023 and admitted to hospital H010 are shown. There are two 'where' conditions used in this query on hospital_id and date of admission,

```
select patients.patient_name, patients.admission_date,
patients.discharge_date, doctors.doctor_name
from patients
join doctors on patients.doctor_id=doctors.doctor_id
where doctors.hospital_id='H010' and patients.admission_date > '2023-01-
01';
```

```
hive> select patients.patient_name, patients.admission_date, patients.discharge_date, doctors.doctor_name
    > from patients
    > join doctors on patients.doctor_id=doctors.doctor_id
    > where doctors.hospital_id='H010' and patients.admission_date > '2023-01-01';
Total jobs = 1
24/04/07 20:11:36 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cla
ses where applicable
24/04/07 20:11:36 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_20-11-34_322_6781964677929316660-1/-local-10006/jo
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
24/04/07 20:11:36 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_20-11-34_322_6781964677929316660-1/-local-10006/jo
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
Execution log at: /tmp/hadoop/hadoop_20240407201111_a10003b4-fe7b-4aeb-8dce-710b63abc348.log
2024-04-07 08:11:37     Starting to launch local task to process map join;      maximum memory = 518979584
2024-04-07 08:11:39     Dump the side-table into file: file:/tmp/hadoop/hive_2024-04-07_20-11-34_322_6781964677929316660-1/-lo
al-10003/HashTable-Stage-3/MapJoin-mapfile111--.hashtable
2024-04-07 08:11:39     Uploaded 1 File to: file:/tmp/hadoop/hive_2024-04-07_20-11-34_322_6781964677929316660-1/-local-10003/H
shTable-Stage-3/MapJoin-mapfile111--.hashtable (326 bytes)
2024-04-07 08:11:39     End of local task; Time Taken: 1.305 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1712502557830_0013, Tracking URL = http://localhost:8088/proxy/application_1712502557830_0013/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1712502557830_0013
Hadoop job information for Stage-3: number of mappers: 1; number of reducers: 0
2024-04-07 20:11:46,022 Stage-3 map = 0%,  reduce = 0%
2024-04-07 20:11:52,360 Stage-3 map = 100%,  reduce = 0%, Cumulative CPU 1.44 sec
MapReduce Total cumulative CPU time: 1 seconds 440 msec
Ended Job = job_1712502557830_0013
MapReduce Jobs Launched:
Job 0: Map: 1   Cumulative CPU: 1.44 sec   HDFS Read: 1988 HDFS Write: 186 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 440 msec
OK
Jennifer Clark  2023-04-10      2023-04-14      Dr. Carter
Steven Taylor   2023-04-11      2023-04-15      Dr. Ethans
Daniel Miller   2023-07-15      2023-07-19      Dr. Carter
Lisa Brody      2023-06-06      2023-06-12      Dr. Ethans
Time taken: 20.134 seconds, Fetched: 4 row(s)
hive>
```

9. This query shows patients being treated by a specific doctor. The name of the doctor i.e. ‘Dr. Smith’ is given in where condition.

```
select doctors.doctor_name, patients.patient_name from doctors
join patients on doctors.doctor_id=patients.doctor_id
where doctors.doctor_name='Dr. Smith';
```



10. This query demonstrates the patient details such as patient ID, their name and age who are under the treatment of Dr. Harris. The results are also ordered by the date of admission of each patient in descending order.

```
select patients.patient_id, patients.patient_name, patients.age,
patients.admission_date from patients
join doctors on patients.doctor_id=doctors.doctor_id
where doctors.doctor_name='Dr. Harris'
order by patients.admission_date desc;
```

```
hive> select patients.patient_id, patients.patient_name, patients.age, patients.admission_date
    > from patients
    > join doctors on patients.doctor_id=doctors.doctor_id
    > where doctors.doctor_name='Dr. Harris'
    > order by patients.admission_date desc;
Total jobs = 1
24/04/07 21:21:13 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java clas
ses where applicable
24/04/07 21:21:14 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_21-21-11_495_4812960794201402272-1/-local-10007/job
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.retry.interval;  Ignoring.
24/04/07 21:21:14 WARN conf.Configuration: file:/tmp/hadoop/hive_2024-04-07_21-21-11_495_4812960794201402272-1/-local-10007/job
conf.xml:an attempt to override final parameter: mapreduce.job.end-notification.max.attempts;  Ignoring.
Execution log at: /tmp/hadoop/hadoop_20240407212121_002ab47f-0274-4002-8384-6443546ca5f3.log
2024-04-07 09:21:15     Starting to launch local task to process map join;      maximum memory = 518979584
2024-04-07 09:21:16     Dump the side-table into file: file:/tmp/hadoop/hive_2024-04-07_21-21-11_495_4812960794201402272-1/-loc
al-10004/HashTable-Stage-2/MapJoin-mapfile141--.hashtable
2024-04-07 09:21:16     Uploaded 1 File to: file:/tmp/hadoop/hive_2024-04-07_21-21-11_495_4812960794201402272-1/-local-10004/Ha
shTable-Stage-2/MapJoin-mapfile141--.hashtable (281 bytes)
2024-04-07 09:21:16     End of local task; Time Taken: 1.311 sec.
Execution completed successfully
MapredLocal task succeeded
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1712502557830_0016, Tracking URL = http://localhost:8088/proxy/application_1712502557830_0016/
Kill Command = /home/hadoop/hadoop/bin/hadoop job  -kill job_1712502557830_0016
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2024-04-07 21:21:23,305 Stage-2 map = 0%,  reduce = 0%
2024-04-07 21:21:29,551 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 0.9 sec
2024-04-07 21:21:35,875 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 1.96 sec
MapReduce Total cumulative CPU time: 1 seconds 960 msec
Ended Job = job_1712502557830_0016
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.96 sec   HDFS Read: 1988 HDFS Write: 127 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 960 msec
OK
7       Robert Moore    55      2024-03-27
```

```
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.96 sec   HDFS Read: 1988 HDFS Write: 127 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 960 msec
OK
7       Robert Moore    55      2024-03-27
17      Arthur Williamson       70      2023-06-17
12      Audrey Reynolds 29      2023-02-12
31      Oliver Green    27      2021-01-10
Time taken: 26.59 seconds, Fetched: 4 row(s)
hive>
```

Task B: MapReduce Programming

Output the number of papers by each author for each year.' Given below is the pseudo code in MapReduce.

```
class pprcount:

    def (self):
        self.output = {}

    def emit(self, key, value):
        if key not in self.output:
            self.output[key] = 0
        self.output[key] += value

    def map(self, line):
        fields = line.split('|')
        authors = fields[0].split(',')
        year = fields[3]

        for author in authors:
            key = author + '|' + year
            self.emit(key, 1)

    def reduce(self):
        for key, values in self.output.items():
            yield (key, value)
```

(Mishra, et al., 2023)

We know that the different fields are separated by the "|" character. In the above pseudo-code, computation of keys and values is done by the 'pprcount' class. The map function takes each line of input and splits it into fields using the '|' delimiter. Then, it obtains the authors and the year from the input line. 'Authors' is the first attribute in the input data and indexing starts from 0. Hence, it is accessed using fields[0]. Similarly, the year is the fourth attribute in the input data, and it is accessed using fields[3]. Value associated to each key is set as 1 by default. This value indicates the presence of an author for 1 paper in a specific year. For each author in the authors list, a key is constructed using the author's name and the publication year (author|year). So, each (key, value) pair emitted by the map stage represents a unique author and the year they published a paper, with the value set to 1. In the reduce function, the output pairs from the map stage are received as input. The function calculates over the output dictionary, which contains the emitted key-value pairs. Further, the reduce function adds up each of the values for all unique keys. This displays the count of papers each author has published in each year. The final output is imagined to be a series of (author|year, count) pairs.

In conclusion, the key is the combination of author and year of publication and the value is initially 1 but after aggregation it will be summed then the final value will be the count of papers by each author in each year.

This algorithm is efficient for counting the number of papers each author has published in each year. The MapReduce model efficiently distributes the workload across multiple compute nodes in a distributed computing environment, enabling parallel processing of large datasets. By emitting key-value pairs with a value of 1 in the map stage, the algorithm minimizes the amount of data shuffled and processed in the reduce stage, which helps improve performance (Samples, 2023).

However, if the dataset is extremely large or the number of unique authors and years is very high, the performance of the algorithm may degrade due to increased memory and processing requirements. In such cases, optimizations like combiners or partitioning strategies may be employed to improve efficiency.

Task C: Big Data Project Analysis

Task C.1

Consider the ABC Investment Bank Ltd which seeks to use social media and other various data sources to generate trading and portfolio balancing strategies. The volume of such kind of data is enormous. For this purpose, building a data lake is essential. A data lake is a centralized storage facility that enables the storage of both organized and unorganized data, regardless of its size. Without initially structuring the data, it can be saved in its current state and various forms of real-time analytics, big data processing, and creating dashboards can be performed. This also helps in better decision-making (AWS). Jotted below are some major advantages of data lakes and how they may benefit this company.

1. Flexibility - Data lakes facilitate the storage of structured, semi-structured, and raw data. Data that is going to be collected is going to be in the form of different data types including textual, pictographic, streams and batches etc. In this situation, a data lake will act as an ideal storage house to stock various forms of huge data. Any volume of data can be imported in real time through data lakes (Dutta, 2024). The processing required to modify the structure to fit an enterprise scheme is minimal or non-existent.
2. Data Integration - Data lakes are an integrated system since several sources of data are gathered, and the original format of the data is retained and transferred into the data lake. Data can be consolidated, and this will provide a comprehensive view of trends in the market. We not only require only market data but a combination of corporate data and say customer sentiment data too. Therefore, this feature will be helpful because with the help of data lakes, information will not be used in isolation but in fact in unison.
3. Low Cost - As compared to traditional data warehouses, data lakes provide a faster approach at a lesser cost. They make use of open-source technology and scalable, cloud-based storage solutions to reduce infrastructure costs and offer an affordable platform for data processing and analysis. In a data lake architecture, we can access data at adequate rates and receive a considerably larger storage volume for less money (AWS).
4. Scalability - The idea of scalability is of the utmost importance because, as any business expands, it must have an architecture that can continue to provide all clients with the same level of service without compromising on efficiency (Gopalan, 2022). Large volumes of structured data are typically managed by a data warehouse, which is designed for workloads that need a lot of reading. A data lake, on the other hand, is intended for storing enormous amounts of unstructured, raw data and can scale seamlessly to accommodate growing data volumes (Dutta, 2024).

Based on the points above, we can conclude that the reason why we are building a data lake is that our data is a mixture of structured and unstructured, texts and videos, stable and raw

data and data lakes can help in its storage. Having it all together in one place would also make further analysis easy. Tools can be applied directly, and it outputs faster queries as compared to a data warehouse. Moreover, another main requirement is to accommodate over 200 petabytes of data which requires a technology which can extend horizontally to meet expanded amounts of data (Dutta, 2024). Using data lakes for schema, transformations and scaling will be time-saving.

Building a data lake requires a thoughtful approach that progresses through multiple stages, with an emphasis on scalability, integration, and sophisticated analytics. Once data lake is implemented it can be accessed by data scientists, analysts and developers (AWS).

- Raw-data zone- This initial stage describes the purpose and involves setting up a low-cost, scalable environment for capturing raw data without affecting existing IT systems. It serves as a zone where data is stored in for an indefinite time before being prepared for use. Implement strong governance measures, including thorough tagging and classification of data, to avoid creating a data swamp. This is crucial for managing the data effectively and ensuring it can be easily accessed and used.
- Storage and processing- The data lake becomes a platform for data scientists to experiment with data. It allows for easy access to raw data, enabling them to focus on analysis and experimentation using a range of open-source and commercial tools (Hagstroem, Roggendorf, Saleh, & Sharma, 2017). Data pipelines can be implemented to transform raw data into actionable insights using machine learning models and algorithms.
- Data Architecture- The data ingestion layer enables the seamless collection of data from various sources using methods like batch processing, real-time streaming, data connectors, and APIs. It allows for the efficient ingestion of data without prior transformation or schema design (Gomede, 2023). As the data lake matures, it becomes a core part of the data infrastructure, replacing existing data marts or running data stores. It enables the provision of data as a service and supports enhanced analytics and machine-learning programs (Hagstroem, Roggendorf, Saleh, & Sharma, 2017). In the storage layer, data is stored in its raw form. It's built on scalable cloud-based storage solutions like AWS, Azure Data Lake Storage, or Google Cloud Storage, accommodating massive volumes of data cost-effectively.
- Security- Data governance policies must be in place to ensure data quality and secure storage. Since confidentiality must be kept regarding decisions taken by the bank. Metadata management, cataloguing, and organizing data assets within the data lake can be executed (Gomede, 2023). This will also require regular updation so that performance can be optimized, and the analytical models can be adapted to meet the fluctuating data.

These strategies can help the bank to proceed with implementing data lake.

Task C.2

In this business case of inventing a platform for stocks prediction, holdings and trading in high end markets there is going to be a continuous shift in the type and content of data generated. To ensure that ABC bank's trading software is the most accurate it is going to need fast-paced and real-time solutions.

- MapReduce and Hadoop have lagging hard drives for a lot of read/write operations. These delays get less tolerable as user expectations and data quantities rise. Additionally, due to the intrinsic batch-processing nature of MapReduce, it seems more designed to process large volumes of data in parallel. MapReduce jobs typically involve data shuffling, map and reduce phases, and disk I/O, which can infer more response time. These sub-tasks involved will increase the data analysis time and batch processing environments are also limited.
- Complexity might arise during the implementation and maintenance of a MapReduce-based near real-time processing system, particularly when handling resource management, scalability, and fault tolerance. Overall, it is a complex technology that may need a thorough knowledge of programming languages like Java, Scala, and Python.
- Social media data streams can be high volume, high velocity, and unstructured, demanding real-time analysis to extract relevant insights. MapReduce may not be ideal for dealing with continuous streams of data and responding in real-time to quickly changing patterns or occurrences.

Considering the above factors, MapReduce might not be the optimal choice to process this requirement. Agreed, that MapReduce is good for processing huge, unorganized data and it is cheaper. But for the specific task of achieving real-time answers about financial products and also for dealing with social media data which increases exponentially in less time, MapReduce is not a good fit. For real-time processing, alternative technologies might provide more straightforward and adaptable options.

Technologies like Apache Spark or Apache Tez may be more suited. Apache Tez provides an added performance over MapReduce and also maintains the ability to extend as per changing volumes of data.

Apache Spark uses in-memory processing, and this reduces the response time as compared to disk-based processing of MapReduce. The stream processing of Apache Spark is notable and provides support for processing continuous data streams like streaming prices of financial products. All queries on structured streams are routed through the Catalyst query optimizer, and they can even be conducted interactively, allowing users to execute SQL queries against live streaming data. In Spark 3.1 and later, streams and tables can be treated identically. The ability to merge numerous streams and perform SQL-like stream-to-stream connections opens up new possibilities for ingestion and transformation (Pointer, 2024). Spark also supports micro-batch processing, fault tolerance and low latency making this a sound choice for real-time performance.

Task C.3

The implementation of a Big Data solution for ABC Investment Bank Ltd.'s project, a hybrid cloud hosting strategy will be the right approach. This combines the benefits of on-premises infrastructure with those of public cloud services. The bank will be able to take advantage of the cloud's scalability and flexibility while maintaining control over sensitive data and adhering to legal regulations. The utilization of data lakes and cloud computing can contribute to the creation of a robust and flexible technological invention.

If a public cloud environment is chosen it could provide easy scalability, reliability and it would be available to all institutions. But devising the entire project on a public cloud will not be safe. We know that sensitive information about financial instruments and shares is involved. Not only is the outcome to be kept confidential but the raw data collected also consists of sensitive and valuable information. Security cannot be compromised in this case. The public cloud neither provides security nor does it cater to custom-made solutions which is actually required for this task.

Let us talk about the private cloud which does provide an enhanced level of security and there is complete control over own servers. But it can be a very advanced technology especially if most employees have only a limited understanding of it and requires constant monitoring by IT experts. This will make it increasingly expensive for big businesses. It will be inaccessible or tough to access from worldwide locations.

Considering that this business is going to be on a global scale and the application needs to be highly accessible from across the world a private cloud system will not be helpful. It needs to be feasible to collect and process huge amounts of structured and unstructured data. Moreover, the confidentiality should be maintained to give the bank an upper edge over its competitors. Some more benefits are:

- Hybrid cloud model- In a hybrid model, the business can employ the benefits of private and public cloud deployment models. It is like the ‘best of both worlds’. The combination allows for a more tailored and fitted resolution that will meet the requirements precisely.
- On-premises and cloud-based- Keeping an on-site data center up to date for vital applications and data that demand low latency and stringent security measures should be done. The trading systems and sensitive financial data that need to stay inside the bank's physical infrastructure can all be kept in this data center. At the same time cloud-based backup can provide faster recovery in case of any disaster (Susnjara, 2023). Hence, agility of the cloud will come in handy and enhanced security will be ensured.
- Storage- Store data in a hybrid cloud storage architecture that spans both on-premises and cloud-based storage solutions. Use on-premises storage systems for sensitive and regulated data, while leveraging cloud object storage services for scalable and cost-effective storage of unstructured data, logs, and backups.
- Other features- Utilizing a hybrid cloud platform approach will enable the company to take advantage of all available services. A hybrid cloud management platform can

operate on any cloud which will prove to be cost-effective. Workloads can be deployed effortlessly across a multi-cloud environment. One of the pros of a hybrid cloud is also being scalable which will be beneficial. Horizontal scaling or adding compute nodes so as to accommodate the growing data.

Concluding from the above features we can say that for ABC bank, a hybrid cloud model will prove to be fruitful. The main requirements which are security, accessibility and scalability are all fulfilled by a hybrid cloud deployment model.

References

AWS. (n.d.) *What is a Data Lake?* Available at: https://aws.amazon.com/what-is/data-lake/ (Accessed: April 2024)

AWS. (n.d.) *What's the Difference Between a Data Warehouse, Data Lake, and Data Mart?*, Available at: https://aws.amazon.com/compare/the-difference-between-a-data-warehouse-data-lake-and-data-mart/ (Accessed: April 2024)

Dutta, S. (2024) *A Comprehensive Guide on Data Lake Architecture.*, Available at: https://www.sprinkledata.com/blogs/data-lake-architecture (Accessed: April 2024)

Gomede, E. (2023) *Data Lake Architecture: A Flexible and Scalable Approach to Modern Data Management.*, Available at: https://medium.com/@evertongomede/data-lake-architecture-a-flexible-and-scalable-approach-to-modern-data-management-2b3c5048c808 (Accessed: April 2024)

Gopalan, R. (2022) *The Cloud Data Lake: A Guide to Building Robust Cloud Data Architecture.* O'Reilly Media, ISBN-13: 978-1098116583, Available at: https://www.oreilly.com/library/view/the-cloud-data/9781098116576/ch04.html (Accessed: April 2024)

Hagstroem, M., Roggendorf, M., Saleh, T., & Sharma, J. (2017) *A smarter way to jump into data lakes.*, Available at: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/a-smarter-way-to-jump-into-data-lakes (Accessed: April 2024)

Mishra, M., Kumar, P., Bhat, R., Murthy V, R., Contractor, D., & Tamilselvam, S. (2023) *Prompting with Pseudo-Code Instructions.*, Available at: https://arxiv.org/pdf/2305.11790.pdf (Accessed: April 2024)

Pointer, I. (2024) *What is Apache Spark? The big data platform that crushed Hadoop.*, Available at: https://www.infoworld.com/article/3236869/what-is-apache-spark-the-big-data-platform-that-crushed-hadoop.html (Accessed: April 2024)

Samples, Q. (2023) *What Is MapReduce In Big Data.,* Available at: https://robots.net/fintech/what-is-mapreduce-in-big-data/ (Accessed: April 2024)

Susnjara, S. (2023) *How to build a successful hybrid cloud strategy.,* Available at: https://www.ibm.com/blog/hybrid-cloud-strategy/ (Accessed: April 2024)