LAB - 1

x. Difference b/w data and big data

Traditional data	Big data
a. Small amount of data that can be collected and analysed using traditional methods easily.	a. Big amount of data that cannot processing and analysed easily using traditional methods.
b. Usually structured data. Stored in databases spreed sheets.	b. Includes semi-structured and unstructured data too
c. Manual data collection	c Automatic data collection
d. Slow, gradual analysis	d. Fast, instant analysis
e Typically processed in batches	e. Developed and proceed in real time.

Applications of Big data

(a) Tracking customers spending habit, shopping
(b). Recommendation
(c). Smart traffic system
(d) Auto driving car
(e) Secure air traffic system
(f) Virtual personal assistant tool
(g) IOT Sensors
(h) Education sector.

Database available for big data

* AWS Dynamo DB
x Azure Cosmos DB
x. Amazon keyspaces
x. Amazon Document DB
x Amazon Redshift

x Write SQL queries to do the following

(1) Create + insert 2 values in the table

```
Create table department (
  empID INTEGER primary key.
  name VARCHAR (20) NOT NULL,
  loc_name VARCHAR (15) NOT NULL.
)

Create table employee (
 empID  INTEGER primary key.
 name   varchar (20) not null.
 jobname  varchar (20),
 manager id integer.
hire_data data.
salary varchar
commision varchar.
dep-id integer
)
```

```
insert into emplayee values (1, Ram, SDI, 1,
                             25-01-2024, 140000,
                             50000, 4):
insert into employee value (2, Raj, SDE, 2,
                            25-01-2014, 20000
                            50000, 3):
```

b. Find Salaries of all employee

```
Select salery name
  from employee:
```

c. Find those employees with hire data like February 22, 1991

```
  Select name
  From employee
where job-name = 'Analyst hire-data like'
  'February 22, 1991':
```

d. Calculate avg salary of employees who work as analysit

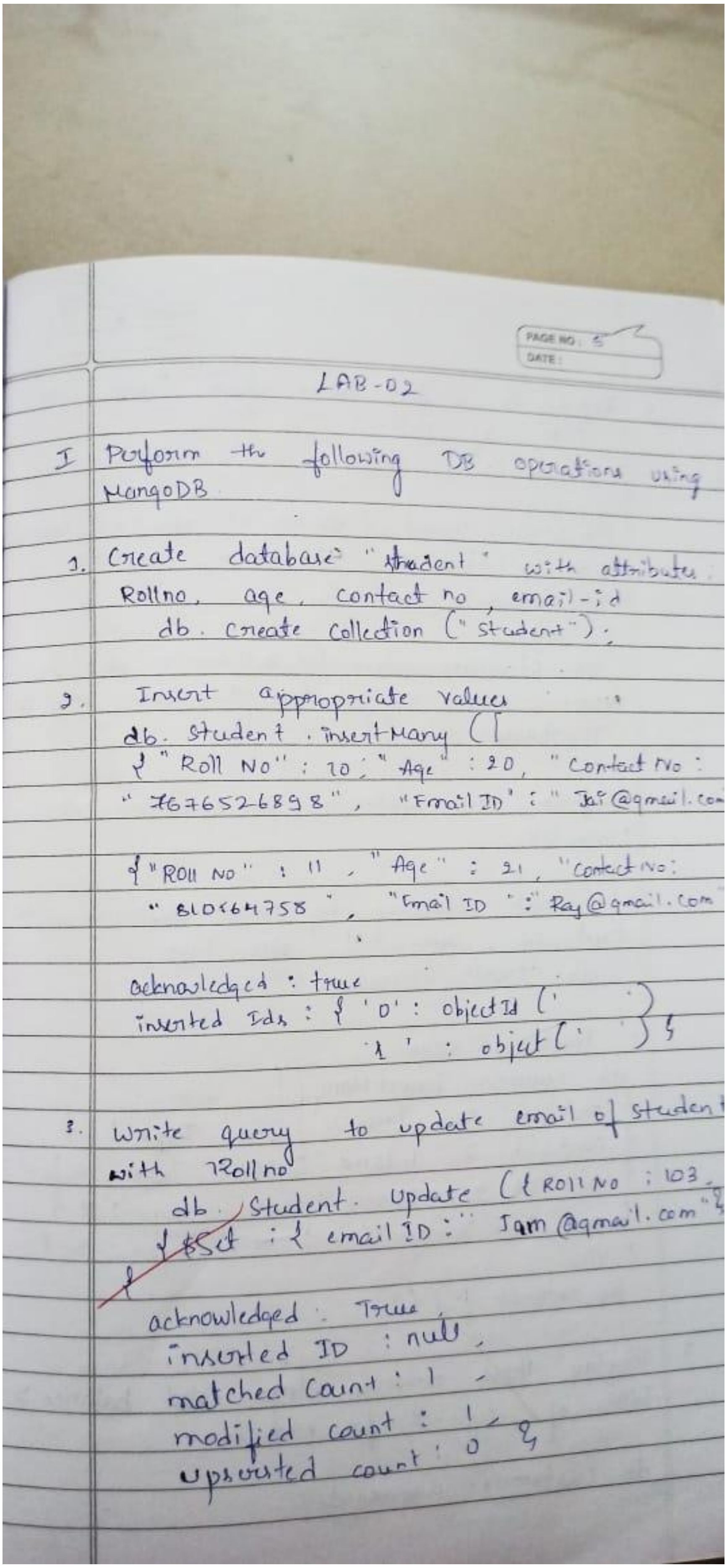
```
  select avg (salary)
  from employee
  where job-name = 'Analyst'
```

(3) Find those employee who are either clerk or Manager

```
Select emp_name
From employees
where job_name in ('clerk', 'Manager')
```

(4). Search for all the employees with an annual salary b/w 24000 and 50000

```
Select salary, emp_name from
employees
uphere salary BETWEEN 24000 AND
    50000 ;
```



LAB-02

I Perform the following DB operations using MongoDB

1. Create database "student" with attributes Rollno, age, contact no, email-id

```
db. create Collection ("student");
```

2. Insert appropriate values

```
db. Student . insertMany ([
{ "Roll No" : 10, "Age" : 20, "Contact No :
" 7676526898", "Email ID' : " Jai @gmail.com

{ "Roll No" : 11 , "Age" : 21 , "contact No:
" 8105647 58 " , "Email ID " : " Ray@gmail.com

acknowledged : true
inserted Ids : { '0' : objectId ('  ')
                '1' : object ('  ') }
```

3. Write query to update email of student with Rollno

```
db. Student. Update ({ Roll No : 103,
{ $Set : { email ID : " Jam @gmail.com" }
}

acknowledged : True,
inserted ID : null,
matched Count : 1,
modified count : 1,
upserted count : 0 }
```

4. Replace the student name from "ABC" to "IEM" of roll = 11

```
// insert apt data

db student insert ({ Roll No : 11, Age : 21, Na
ABC, contact No : 90076814, Email ID :
ABC@gmail.com }):
// replace
 db. student. replace ({ Roll No : 11 Age : 2
Name : IEM, contact No : 90076814, Emai
 Iem@gmail. com }) :
```

II Perform the following DB operations using Mango DB

1 Create collection by name customer with cust_id, Acc_bal, Acc_Type

```
 db. create collection (" customer")
```

2 Insert 5 values

```
db. customer insert Many ([
{ cust_id : 1 , balance : 200 , Type : s }
{ cust_id : 2, balance : 1000 , Type : z }
{ cust_id : 2, balance : 300 , type : z }
{ cust_id : 3, balance : 500 , type : s }
// view
 db customer find ()
```

3. Display those records whose total balance 1200 of type z for each id.

```
db. customer. aggregate (
{ $match : { type : 2}},
{ $group : {_id : "$cust_id"
  totAccbal : { $ sum : "$balance" }}},
{ $match : { totAccbal : { $gt : 1200}}});
_id : 2, totAccbal : 1300
```

6. Determine min and max account balance for each. 1

```
db. customers. aggregate(
{ $group: {_id : " $ cust_id ",
   minAccbal : { $min : " $balance" }
   maxAcc bal : { $max : " $balance" }}}
```

LAB - 03

Perform the following DB operations using Cassandra.

1. Create a keyspace by name employee

```
CREATE KEYSPACE "employee"
WITH replication = { 'class' : ' simpleStrategy' .
' replication _factor' : 2 } ;
```

```
Use employee ;
```

2. Create a column family by name employee info with attributes emp-id primary key, emp-name, Designation, Date-of-joining, Salary, Dept-name

```
    CREATE TABLE emp-info (
        emp-id int    PRIMARY KEY
        emp - Name,  emp - designation
        emp - doj , emp-sal , emp _dept_name ) ;
```

3. Insert the values into table in batch

```
  BEGIN BATCH
  INSERT INTO emp-info(empid, emp-name ,
 emp- designation , emp-doj , emp - sal , emp- deptname
value ( 1234, 'Hiya' , 'Manager', 20-01-24 , 21000
 . Biotech ' ) ;
```

4. update employee name and department of emp
1234

~~upde~~

```
UPDATE emp-info
    SET emp_name = 'Hiru' , emp-deptnam
      'chemistry'
    where emp id = 1234 ;
```

~~5~~.

LAB - 04

Execution of HDFS Commands for interaction w
Hadoop Environment

1. mkdir 2. ls 3. get 4. put
5. copyfromlocal 6. Copytolocal 7

1. 1. To create directories

⇒
```
hadoop fs -mkdir /newDataFlair
```

2. To enlist the files and directories presen
in HDFS

⇒
```
hadoop fs -ls/
```

3. To copy localfile of the localfile system to the Hadoop filesystem

⇒
```
hadoop fs -put ~/localfile /first.text
```

4 To copy the ~~first1~~ file present in the local file system to the newdataFlair directory of Hadoop

⇒
```
hadoop fs -copyFromLocal ~/first /newDataFlair/copy text
```

⇒
```
hadoop fs -cat /newDataFlair/copytext
```

6. To copy the file of the hadoop filesystem to the local file system

⇒ hadoop fs -get /first ~/ copyfromhadoop

13/3

7. Hadoop HDFS get command Example hadoop fs -getfact /abl

This apache hadoop command shows the acces control lists of files and directories

8. Cat , hdfs dfs - cat /abc/wc.txt

9. mr , HDFS mr Command Example hadoop fs -mr /abc /FFF .b

10. CP , hadoop fs -cp /CSE /LLL hadoop fs -ls /LLL

Wordcount MapReduce program

Terminal

-> start - all. sh

-> jps

-> sudo snap installe eclipe--classic

-> su hadoop

1. open eclipse create new java project

2. click right on the wordcount project -> build configure build path -> libraries -> add jar files from (share / common and share /mapreduce)

3. Right click on the wordcount add classes called WCDriver, WCMapper, WCReducer

4. Go to Home / etc/Hadoop / .xml and modify it

5. Fight click on the wordcount -> export -> jar file -> Create jar file wordcount.jar

6. Open terminal
-> hadoop fs - mkdir / rgo
-> hadoop fs - copy FromLocal /home/hadoop/ Desktop / sample .txt /rgn / test. txt
-> hadoop jar /home /hadoop / Desktop / word. count.jar WCDriver /rgs /text.~~txt~~ / output
-> hadoop . output
-> hadoop - cat /output / part -0000

x. Create Map Reduce program to
(a) Find average temperature for each year
from NCDC data set.

Mapper :-

```
import sys

for line in sys.stdin :
        line = line.strip()
        part = line.split ("\t")
        date = part [0]
      temperature = part [1]
        year    = date [:4]

try :
    temperature = float (temperature)
    print (f" {year} \t {temperature}")
except valueError :
    continue
```

Reducer :-

```
import sys

current_year = None
 Sum_temp  = 0
 count_temp  = 0

 for line in   sys. stdin :
      line  =  line . strip ()
```

```
    year, temperature = line.split ("\t")
try :
  temperature = float (temperature)
except ValueError:
  continue
 if current - year = year:
       sum - temp + = temperature
       Count-temp + = 1
else :
      if current_year:
      average _temp = Sum_temp / Count-temp

    Current-year = year
     Sum_temp = temperature
     count-temp = 1
if  current -year:
     average-temp = Sum-temp / counttemp
 print (f" {current_year } \t {average-
 temp }")
```

*. create MapReduce program to
Q. Find the mean max temperature for every month, give

Mapper :

```
import sys

for line in sys.stdin:
  line = line.strip()
  parts = line.strip()
  date = parts[0)
 temperature = parts[1]
  year_month = date[:7]

try :
  temperature = float (temperature)
  print (f" {year_month}\t {temperature}")
Except valueError :
  continue
```

Reducer :

```
import sys

current_month = None
 sum_temp = 0
 count_temp = 0

 for line in sys.stdin:
     line = line.strip()
```

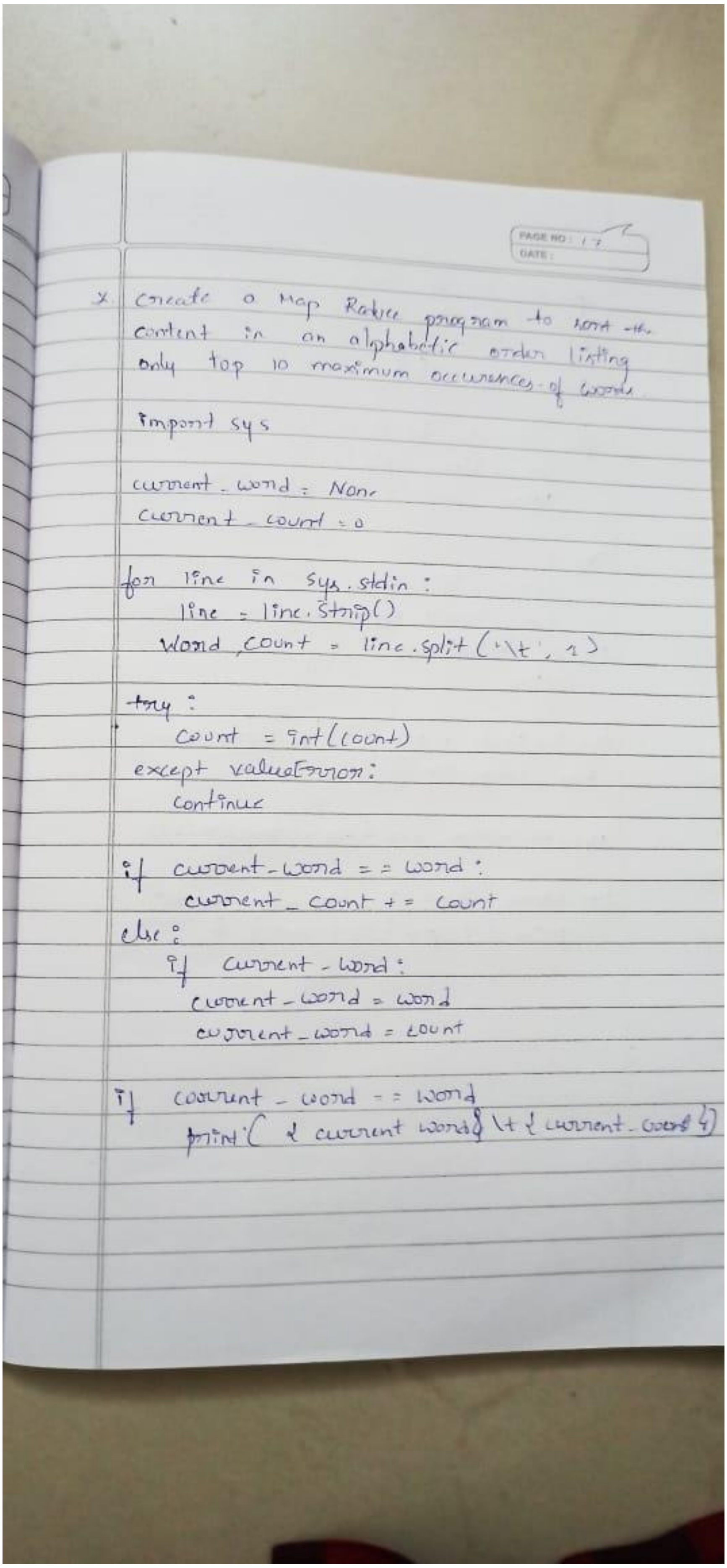
```
month, temperature = line.split ("\t")

try :
  temperature = float (temperature)
except valueError :
  continue

if  current-month == month :
    sum-temp + = temperatur
    count-temp  + = 1
else :
   if  current-month :
      average-temp  =  sum-temp / count-temp
   print { current -month average temp }

  current-month =month
   sum-temp   = temperature
   count-temp  = 1

if  current-month :
   average-temp  =  sum-temp / count-temp
   print ( current-month   average-temp )
```



x. Create a Map Reduce program to sort the content in an alphabetic order listing only top 10 maximum occurences of words.

```
import sys

current_word = None
current_count = 0

for line in sys.stdin :
    line = line.strip()
   word ,count = line.split ('\t', 1)

try :
   count = int(count)
except valueError:
   continue

if current-word == word :
    current_count += count
else :
   if current - word :
     current - word = word
     current_word = count

if current - word == word
   print(f{current word}\t{current_count})
```

```
for line in sys.stdin:
print (line.strip())

word-counts = [ ]

for line in sys.stdin:
    line = line.strip()
    word, count = line.split ('\t', 1)

try:
    count = int(count)
    word-counts.append((word, count))
except ValueError:
  continue

top_10_words = nlargest (10, word_counts,
 key = itemgetter (1))

top-10-words.sort (key = itemgetter (0))

for word, count in top-10-words:
    print ("{word}\t {count}"}
```