ADVANCED DATABASE SYSTEMS

JOURNAL

Name: Shaikh Faisal Alim.
Seat no: 31031522029

INDEX SECTION

Practical No. 1:

AIM: Demonstrate distributed databases environment by dividing given global conceptual schema, into vertical and Horizontal fragments and place them on different nodes. Execute queries on these fragments.

Step 1 Connect to the systems user

Step 2 Open another terminal, connect to the systems user and alter the hr user using:

ALTER USER hr identified by hr ACCOUNT UNLOCK;

LOGIN TO USER hr

connect

username: hr

password: hr

```
SQL> connect
Enter user-name: system
Enter password:
Connected.
SQL> ALTER USER hr identified by hr ACCOUNT UNLOCK;

User altered.
```

a)VERTICAL FRAGMENTATION

Step 3 Create tables and insert values in both the users.

i) Systems User:

CREATE TABLE emp_1(eno number(3) PRIMARY KEY, ename char(10), address varchar2(20));

INSERT INTO emp_1 VALUES(1, ‘Akansha’, 'Jogeshwari');

INSERT INTO emp_1 VALUES(2, ‘Faisal’, ‘Govandi');

INSERT INTO emp_1 VALUES(3, 'Aman', 'Vidhyavihar');

INSERT INTO emp_1 VALUES(4, 'Harry', 'Vidhyavihar');

INSERT INTO emp_1 VALUES(5, 'Khushi', 'Panvel');

```
SQL> create table emp_1(eno number(3) PRIMARY KEY, ename char(10), address varchar2(20));

Table created.

SQL> insert into emp_1 values(1, 'Akansha','Jogeshwari');

1 row created.

SQL> insert into emp_1 values(2, 'Faisal','Govandi');

1 row created.

SQL> insert into emp_1 values(3, 'Aman', 'Vidhyavihar');

1 row created.

SQL> insert into emp_1 values(4, 'Harry', 'Vidhyavihar');

1 row created.

SQL> insert into emp_1 values(5, 'Khushi', 'Panvel');

1 row created.
```

ii) HR User: Do this in the terminal where you have logged in as hr

CREATE TABLE emp_2(eno number(3) PRIMARY KEY, email varchar2(20), salary number(7,2));

INSERT INTO emp_2 VALUES(1, 'akansha@gmail.com',150);

INSERT INTO emp_2 VALUES(2, 'faisal@gmail.com',69);

INSERT INTO emp_2 VALUES(3, 'aman@gmail.com',420);

INSERT INTO emp_2 VALUES(4, 'harry@gmail.com',620);

INSERT INTO emp_2 VALUES(5, 'khushi@gmail.com',76);

```
SQL> create table emp_2(eno number(3) PRIMARY KEY, email varchar2(20), salary number(7,2));

Table created.

SQL> insert into emp_2 values(1, 'akansha@gmail.com',2400);

1 row created.

SQL> insert into emp_2 values(2, 'faisal@gmail.com',2400);

1 row created.

SQL> insert into emp_2 values(3, 'aman@gmail.com',1700);

1 row created.

SQL> insert into emp_2 values(4, 'harry@gmail.com',2100);

1 row created.

SQL> insert into emp_2 values(5, 'khushi@gmail.com',2100);

1 row created.
```

Step 3: Commit on both the users using COMMIT; command

Step 4: Create a database link between systems and hr user. For that you have to fire the following query in the systems terminal

CREATE PUBLIC DATABASE LINK "prac_1" CONNECT TO hr IDENTIFIED BY hr USING 'XE';

```
SQL> CREATE PUBLIC DATABASE LINK "prac_1" CONNECT TO hr IDENTIFIED BY hr USING 'XE';

Database link created.
```

Step 5: Now start quering in the systems user terminal

1. Find the salary of an employee where employee number is known.

SELECT salary FROM emp_1@prac_1 where eno = 1;

```
SQL> SELECT salary FROM emp_2@prac_1 WHERE eno=1;

    SALARY
----------
      2400
```

2. Find the Email where the employee name is known.

SELECT email FROM emp_2@prac_1 e2, emp_1 e1 WHERE e2.eno = e1.eno AND e1.ename = 'Aman';

```
SQL> SELECT email FROM emp_2@prac_1 e2, emp_1 e1 WHERE e2.eno = e1.eno AND e1.ename = 'Aman';

EMAIL
--------------------
aman@gmail.com
```

3. Find the employee name and Email where employee number is known.

SELECT ename,email FROM emp_2@prac_1 e2, emp_1 e1 WHERE e1.eno = e2.eno AND e1.eno = 4;

```
SQL> SELECT ename,email FROM emp_2@prac_1 e2, emp_1 e1 WHERE e1.eno = e2.eno AND e1.eno = 4;

ENAME      EMAIL
---------- --------------------
Harry      harry@gmail.com
```

4. Find the employee name whose salary is = 69

SELECT ename FROM emp_1 e1, emp_2@prac_1 e2 WHERE e1.eno = e2.eno AND e2.salary = 69

```
SQL> SELECT ename FROM emp_1 e1, emp_2@prac_1 e2 WHERE e1.eno = e2.eno AND e2.salary = 2400;

ENAME
----------
Akansha
Faisal
```

b) HORIZONTAL FRAGMENTATION

Step 3 Create tables and insert values in both the users.

1) System user

CREATE TABLE emp1(eno number(3) PRIMARY KEY, ename char(10), address varchar2(20), email varchar2(20), salary number(7));

INSERT INTO emp1 VALUES(1, 'Soham', 'Ghatkopar east', 'sohamhegde@gmail.com',150);

INSERT INTO emp1 VALUES(2, 'Ethan', 'Santacruz city', 'ethan@gmail.com',69);

```
SQL> create table emp1(eno number(3) PRIMARY KEY, ename char(10), address varchar2(20), email varchar2
(20), salary number(7));

Table created.

SQL> insert into emp1 values(1, 'Akansha', 'Jogeshwari', 'akansha@gmail.com', 2400);

1 row created.

SQL> insert into emp1 values(2, 'Faisal', 'Govandi', 'faisal@gmail.com', 2400);

1 row created.
```

2) hr user

CREATE TABLE emp2(eno number(3) PRIMARY KEY, ename char(10), address varchar2(20), email varchar2(20), salary number(7));

INSERT INTO emp2 VALUES(3, 'Aman', 'Vidhyavihar', 'aman@gmail.com',1700);

INSERT INTO emp2 VALUES(4, ‘Harry’, ‘Vidhyavihar’, 'harry@gmail.com', 2100);

INSERT INTO emp2 VALUES(5, 'Khushi', 'Panvel', 'khushi@gmail.com',2100);

```
SQL> create table emp2(eno number(3) PRIMARY KEY, ename char(10), address varchar2(20), email varchar2
(20), salary number(7));

Table created.

SQL> insert into emp2 values(3, 'Aman', 'Vidhyavihar', 'aman@gmail.com', 1700);

1 row created.

SQL> insert into emp2 values(4, 'Harry', 'Vidhyavihar', 'harry@gmail.com', 2100);

1 row created.

SQL> insert into emp2 values(5, 'Khushi', 'Panvel', 'khushi@gmail.com', 2100);

1 row created.
```

Step 3: Commit on both the users using COMMIT; command

Step 4: Create a database link between systems and hr user. For that you have to fire the following query in the systems terminal

CREATE PUBLIC DATABASE LINK "prac_1" CONNECT TO hr IDENTIFIED BY hr USING 'XE';

```
SQL> CREATE PUBLIC DATABASE LINK "prac1" CONNECT TO hr IDENTIFIED BY hr USING 'XE';

Database link created.
```

STEP 5: Do all the queries

1. Find the salary of all employees.

SELECT salary FROM emp1 e1 UNION select salary FROM emp2@prac_1;

```
SQL> SELECT salary FROM emp1 e1 UNION select salary FROM emp2@prac1;

    SALARY
----------
      1700
      2100
      2400
```

2 Find the employee name and Email where employee number is known.

SELECT ename FROM emp1 WHERE eno = 1 UNION SELECT ename FROM emp2@prac_1 WHERE eno = 4;

```
SQL> SELECT ename FROM emp1 WHERE eno = 1 UNION SELECT ename FROM emp2@prac1 WHERE eno = 4;

ENAME
----------
Akansha
Harry
```

1. Find the employee name and address where employee number is known

SELECT ename,address FROM emp1 WHERE eno = 2 UNION SELECT ename,address FROM emp2@prac_1 WHERE eno = 3;

```
SQL> SELECT ename, address FROM emp1 WHERE eno = 2 UNION SELECT ename, address FROM emp2@prac1 WHERE e
no = 4;       4;

ENAME      ADDRESS
---------- --------------------
Faisal     Govandi
Harry      Vidhyavihar
```

PRACTICAL 2:

AIM: Create an XML database and demonstrate insert, update and delete operations on these tables. Issue queries on it.

CREATING AND INSERTING DATA IN TABLE

Code:

```
CREATE TABLE employee(dept_id number(5) PRIMARY KEY, emp_spec XMLType);
```

INSERTING VALUES

```
INSERT INTO employee VALUES

(1001,XMLType( '<employees>

   <emp id = "111">

   <name>Reena</name>

   <email>reena@gmail.com</email>

   <acc_no>1001</acc_no>
```

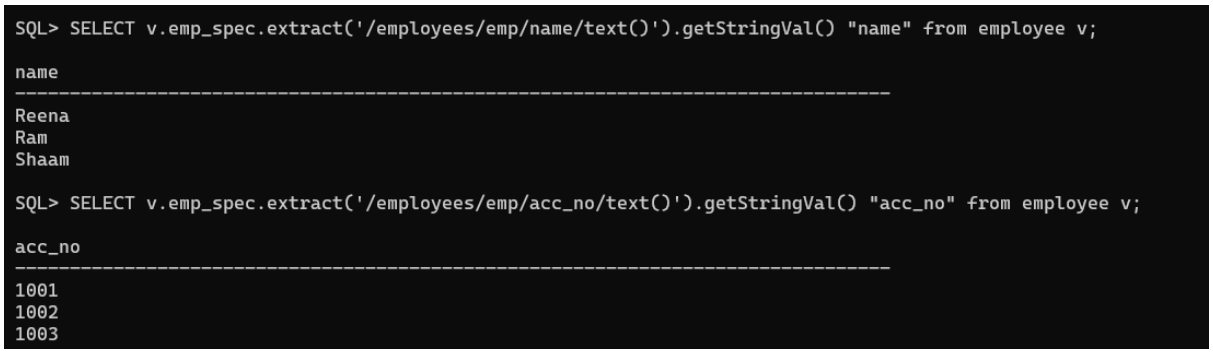
```
   <dateofjoining>24-12-2011</dateofjoining>
   </emp>
   </employees>'));

INSERT INTO employee VALUES
(1002,XMLType('<employees>
   <emp id = "112">
   <name>Ram</name>
   <email>ram@gmail.com</email>
   <acc_no>1002</acc_no>
   <dateofjoining>12-02-2011</dateofjoining>
   </emp>
   </employees>'));

INSERT INTO employee VALUES
(1003,XMLType('<employees>
   <emp id = "113">
   <name>Shaam</name>
   <email>shaam@gmail.com</email>
   <acc_no>1003</acc_no>
   <dateofjoining>02-04-2011</dateofjoining>
   </emp>
   </employees>'));
```

GETTING NAMES OF EMPLOYEES

SELECT v.emp_spec.extract('/employees/emp/name/text()').getStringVal() "name" from employee v;

GETTING ACCOUNT NO OF EMPLOYEES

SELECT v.emp_spec.extract('/employees/emp/acc_no/text()').getStringVal() "acc_no" from employee v;



GETTING ACCOUNT NAMES, EMAIL AND DATE OF JOINING

SELECT e.emp_spec.extract('/employees/emp/name/text()').getStringVal() "name",

e.emp_spec.extract('/employees/emp/email/text()').getStringVal "email",

e.emp_spec.extract('/employees/emp/dateofjoining/text()').getStringVal "dateofjoining" from employee e;

```
SQL> SELECT e.emp_spec.extract('/employees/emp/name/text()').getStringVal() "name",
  2  e.emp_spec.extract('/employees/emp/email/text()').getStringVal "email",
  3  e.emp_spec.extract('/employees/emp/dateofjoining/text()').getStringVal "dateofjoining" from employee e;

name
--------------------------------------------------------------------------------
email
--------------------------------------------------------------------------------
dateofjoining
--------------------------------------------------------------------------------
Reena
reena@gmail.com
24-12-2011

Ram
ram@gmail.com
12-02-2011

name
--------------------------------------------------------------------------------
email
--------------------------------------------------------------------------------
dateofjoining
--------------------------------------------------------------------------------

Shaam
shaam@gmail.com
02-04-2011
```

UPDATING XML

UPDATE employee e set emp_spec = XMLTYPE(

'<employees>

<emp id = "114">

<name>Ram</name>

<email>ram@gmail.com</email>

<acc_no>1002</acc_no>

<dateofjoining>12-02-2011</dateofjoining>

</emp>

</employees>'

) where e.emp_spec.extract('/employees/emp/@id').getStringVal() = '112';

```
SQL>  UPDATE employee e set emp_spec = XMLTYPE(
  2      '<employees>
  3      <emp id = "114">
  4      <name>Ram</name>
  5      <email>ram@gmail.com</email>
  6      <acc_no>1002</acc_no>
  7      <dateofjoining>12-02-2011</dateofjoining>
  8      </emp>
  9      </employees>'
 10   ) where e.emp_spec.extract('/employees/emp/@id').getStringVal() = '112';

1 row updated.
```

DELETING XML RECORDS

DELETE from employee e WHERE e.emp_spec.extract('/employees/emp/@id').getStringVal()='114';

```
SQL> DELETE from employee e WHERE e.emp_spec.extract('/employees/emp/@id').getStringVal()='114';

1 row deleted.
```

PRACTICAL 3

AIM: Create different types that include attributes and methods. Define tables for these types by adding enough tuples. Demonstrate insert, update and delete operations on these tables. Execute queries on them.

STEP 1

Using Object Oriented databases create the following types:

a) AddrType (Pincode: number, Street :char, City : char, state :char)

b) BranchType (address: AddrType, phone1: integer,phone2:integer)

c) AuthorType (name:char,addr AddrType)

d) PublisherType (name: char, addr: AddrType, branches: BranchTableType

e) AuthorListType as varray, which is a reference to AuthorType

f) BranchTableType of BranchType

CODE:

```
CREATE TYPE AddrType AS OBJECT(pincode number(11), street char(20), city char(12), state char(20));

CREATE TYPE BranchType AS OBJECT(address AddrType, phone1 number(10), phone2 number(10));

CREATE TYPE AuthorType AS OBJECT(name char(15), addr AddrType);

CREATE TYPE BranchTableType AS TABLE of BranchType;

CREATE TYPE AuthorListType AS varray(2) OF REF AuthorType;

CREATE TYPE PublisherType AS OBJECT(name char(10), addr AddrType, branches BranchTableType);
```

STEP 2: Create the following tables

g) authors of AuthorType

h) books(title:varchar, year:date, published_by ref PublisherType,authors AuthorListType)

i) Publishers of PublisherType

CODE:

CREATE TABLE authors of AuthorType;

CREATE TABLE books(title varchar2(10), year date, published_by ref PublisherType, authors AuthorListType);

CREATE TABLE Publishers of PublisherType NESTED TABLE branches STORE AS BranchTable;

```
SQL> CREATE TABLE authors of AuthorType;

Table created.

SQL> CREATE TABLE books(title varchar2(10), year date, published_by ref PublisherType, authors AuthorL
istType);

Table created.

SQL> CREATE TABLE Publishers of PublisherType NESTED TABLE branches STORE AS BranchTable;

Table created.
```

```
SQL> CREATE TYPE AddrType AS OBJECT(pincode number(11), street char(20), city char(12), state char(20)
);
  2  /

Type created.

SQL> CREATE TYPE BranchType AS OBJECT(address AddrType, phone1 number(10), phone2 number(10));
  2  /

Type created.

SQL> CREATE TYPE AuthorType AS OBJECT(name char(15), addr AddrType);
  2  /

Type created.

SQL> CREATE TYPE BranchTableType AS TABLE of BranchType;
  2  /

Type created.

SQL> CREATE TYPE AuthorListType AS varray(2) OF REF AuthorType;
  2  /

Type created.

SQL> CREATE TYPE PublisherType AS OBJECT(name char(10), addr AddrType, branches BranchTableType);
  2  /

Type created.
```

STEP 3: PERFORMING CRUD OPERATIONS ON THE TABLE

Authors table

INSERT INTO authors VALUES('Akansha',
AddrType(400060,'Jogeshwari','Mumbai','Maharashtra'));

INSERT INTO authors VALUES('Faisal',
AddrType(400088,'Govandi','Mumbai','Maharashtra'));

INSERT INTO authors VALUES('Aman',
AddrType(400070,'Vidhyavihar','Mumbai','Maharashtra'));

INSERT INTO authors VALUES('Harry',
AddrType(400070,'Vidhayavihar','Mumbai','Maharashtra'));

INSERT INTO authors VALUES('Khushi',
AddrType(40000,'Panvel','Mumbai','Maharashtra'));

```
SQL> CREATE TABLE authors of AuthorType;

Table created.

SQL> CREATE TABLE books(title varchar2(10), year date, published_by ref PublisherType, authors AuthorL
istType);

Table created.

SQL> CREATE TABLE Publishers of PublisherType NESTED TABLE branches STORE AS BranchTable;

Table created.
```

```
SQL> INSERT INTO authors VALUES('Akansha',AddrType(400060,'Jogeshwari','Mumbai','Maharashtra'));

1 row created.

SQL> INSERT INTO authors VALUES('Faisal',AddrType(400088,'Govandi','Mumbai','Maharashtra'));

1 row created.

SQL> INSERT INTO authors VALUES('Aman',AddrType(400070,'Vidhyavihar','Mumbai','Maharashtra'));

1 row created.

SQL> INSERT INTO authors VALUES('Harry',AddrType(400070,'Vidhyavihar','Mumbai','Maharashtra'));

1 row created.

SQL> INSERT INTO authors VALUES('Khushi',AddrType(400070,'Panvel','Mumbai','Maharashtra'));

1 row created.
```

Publishers TABLES

CODE:

```
INSERT INTO Publishers VALUES('Penguin',
AddrType(123456,'Penguin Street','Penguin City','Penguin State'),
BranchTableType(BranchType(
    AddrType(123456,'Penguin Street','Penguin City','Penguin State'),
    123456789,123456780)
    )
)

INSERT INTO Publishers VALUES ('lion',
AddrType(223456,'Lion Street','Lion Ciy','Lion State'),
BranchTableType(BranchType(
    AddrType(223456,'Lion Street','Lion Ciy','Lion State'),
    223456789,
    223456780)));

INSERT INTO Publishers VALUES ('Tiger',
AddrType(323456,'Tiger Street','Tiger Ciy','Tiger State'),
BranchTableType(BranchType(
    AddrType(323456,'Tiger Street','Tiger Ciy','Tiger State'),
    323456789,
    323456780)));

INSERT INTO Publishers VALUES ('Monkey',
AddrType(423456,'Monkey Street','Monkey Ciy','Monkey State'),
BranchTableType(BranchType(
    AddrType(423456,'Monkey Street','Monkey Ciy','Monkey State'),
    423456789,
    423456780)));
```

INSERT INTO Publishers VALUES ('Eagle',

AddrType(523456,'Eagle Street','Eagle Ciy','Eagle State'),

BranchTableType(BranchType(

AddrType(523456,'Eagle Street','Eagle Ciy','Eagle State'),

523456789,

523456780)));

```
SQL> INSERT INTO Publishers VALUES('Penguin',AddrType(123456,'Penguin Street','Penguin city','Penguin
State'),BranchTableType(BranchType(AddrType(123456,'Penguin Street','Penguin City','Penguin State'),12
3456789,123456780)))
  2  /

1 row created.

SQL> INSERT INTO Publishers VALUES('Lion',AddrType(223456,'Lion Street','Lion city','Lion State'),Bran
chTableType(BranchType(AddrType(223456,'Lion Street','Lion City','Lion State'),223456789,223456780)));


1 row created.

SQL> INSERT INTO Publishers VALUES('Tiger',AddrType(323456,'Tiger Street','Tiger city','Tiger State'),
BranchTableType(BranchType(AddrType(323456,'Tiger Street','Tiger City','Tiger State'),423456789,423456
780)));

1 row created.

SQL> INSERT INTO Publishers VALUES('Monkey',AddrType(423456,'Monkey Street','Monkey city','Monkey Stat
e'),BranchTableType(BranchType(AddrType(423456,'Monkey Street','Monkey City','Monkey State'),523456789
,523456780)));

1 row created.

SQL> INSERT INTO Publishers VALUES('Eagle',AddrType(523456,'Eagle Street','Eagle city','Eagle State'),
BranchTableType(BranchType(AddrType(523456,'Eagle Street','Eagle City','Eagle State'),423456789,423456
780)));

1 row created.
```

Books table:

CODE:

INSERT INTO books SELECT 'Python','24-feb-2001',

ref(pub),

AuthorListType(ref(aut))

FROM Publishers pub, Authors aut

WHERE pub.name = 'Penguin' AND aut.name = 'Akansha';

INSERT INTO books SELECT 'Java','2-may-2011',

```
ref(pub),
AuthorListType(ref(aut))
FROM Publishers pub, Authors aut
WHERE pub.name = 'lion' AND aut.name = 'Faisal';

INSERT INTO books SELECT 'Ruby','4-jan-1999',
ref(pub),
AuthorListType(ref(aut))
FROM Publishers pub, Authors aut
WHERE pub.name = 'Tiger' AND aut.name = 'Aman';

INSERT INTO books SELECT 'Carbon','3-feb-2021',
ref(pub),
AuthorListType(ref(aut))
FROM Publishers pub, Authors aut
WHERE pub.name = 'Monkey' AND aut.name = 'Harry';

INSERT INTO books SELECT 'Julia','2-feb-2001',
ref(pub),
AuthorListType(ref(aut))
FROM Publishers pub, Authors aut
WHERE pub.name = 'Eagle' AND aut.name = 'Khushi';
INSERT INTO books SELECT 'Javascript','3-jan-2001',
ref(pub),
AuthorListType(ref(aut))
FROM Publishers pub, Authors aut
WHERE pub.name = 'Eagle' AND aut.name = 'Akansha';

INSERT INTO books SELECT 'Typesctipt','4-jan-2001',
```

```
ref(pub),
AuthorListType(ref(aut))
FROM Publishers pub, Authors aut
WHERE pub.name = 'Monkey' AND aut.name = 'Faisal';

INSERT INTO books SELECT 'Scala','4-may-2001',
ref(pub),
AuthorListType(ref(aut))
FROM Publishers pub, Authors aut
WHERE pub.name = 'lion' AND aut.name = 'Aman';

INSERT INTO books SELECT 'C sharp','4-aug-2001',
ref(pub),
AuthorListType(ref(aut))
FROM Publishers pub, Authors aut
WHERE pub.name = 'Tiger' AND aut.name = 'Harry';

INSERT INTO books SELECT 'C','4-jan-2001',
ref(pub),
AuthorListType(ref(aut))
FROM Publishers pub, Authors aut
WHERE pub.name = 'Penguin' AND aut.name = 'Khushi';
```

```
SQL> INSERT INTO books SELECT 'Python','24-feb-2001',ref(pub),AuthorListType(ref(aut)) FROM Publishers
 pub, Authors aut WHERE pub.name = 'Penguin' AND aut.name = 'Akansha';

1 row created.

SQL> INSERT INTO books SELECT 'Java','2-may-2011',ref(pub),AuthorListType(ref(aut)) FROM Publishers pu
b, Authors aut WHERE pub.name = 'Lion' AND aut.name = 'Faisal';

1 row created.

SQL> INSERT INTO books SELECT 'Ruby','4-jan-1999',ref(pub),AuthorListType(ref(aut)) FROM Publishers pu
b, Authors aut WHERE pub.name = 'Tiger' AND aut.name = 'Aman';

1 row created.

SQL> INSERT INTO books SELECT 'Carbon','3-feb-2021',ref(pub),AuthorListType(ref(aut)) FROM Publishers
pub, Authors aut WHERE pub.name = 'Monkey' AND aut.name = 'Harry';

1 row created.

SQL> INSERT INTO books SELECT 'Julia','2-feb-2001',ref(pub),AuthorListType(ref(aut)) FROM Publishers p
ub, Authors aut WHERE pub.name = 'Eagle' AND aut.name = 'Khushi';

1 row created.

SQL> INSERT INTO books SELECT 'Javascript','3-jan-2001',ref(pub),AuthorListType(ref(aut)) FROM Publish
ers pub, Authors aut WHERE pub.name = 'Eagle' AND aut.name = 'Akansha';

1 row created.

SQL> INSERT INTO books SELECT 'Typescript','4-jan-2001',ref(pub),AuthorListType(ref(aut)) FROM Publish
ers pub, Authors aut WHERE pub.name = 'Monkey' AND aut.name = 'Faisal';

1 row created.

SQL> INSERT INTO books SELECT 'Scala','4-may-2001',ref(pub),AuthorListType(ref(aut)) FROM Publishers p
ub, Authors aut WHERE pub.name = 'Lion' AND aut.name = 'Aman';

1 row created.

SQL> INSERT INTO books SELECT 'C sharp','4-aug-2001',ref(pub),AuthorListType(ref(aut)) FROM Publishers
 pub, Authors aut WHERE pub.name = 'Tiger' AND aut.name = 'Harry';

1 row created.

SQL> INSERT INTO books SELECT 'C','4-jan-2001',ref(pub),AuthorListType(ref(aut)) FROM Publishers pub,
Authors aut WHERE pub.name = 'Penguin' AND aut.name = 'Khushi';

1 row created.
```

STEP 4 Performing queries

a) List all of the authors that have the same address as their publisher:

SELECT a.name FROM Authors a, Publishers p WHERE a.addr = p.addr;

b) List all books that have 2 or more authors:

SELECT b.title FROM books b WHERE 1 < (SELECT COUNT(*) FROM TABLE (b.authors));

c) List the name of the publisher that has the most branches

Select p.name from publishers p, table(p.branches) group by p.name having count(*)>=all (select count(*) from publishers p, table(p.branches) group by name);

d) Name of authors who have not published a book

select (a.name) from authors a minus select a.name from books b, table(b.authors) b1, authors a where b1.column_value=ref(a);

e) List all authors who have published more than one book:

select a.name from authors a, books b, table(b.authors) b1 where b1.column_value =ref(a) group by a.name having count(*) > 1;

g) List all books (title) where the same author appears more than once on the list of authors (assuming that an integrity constraint requiring that the name of an author is unique in a list of authors has not been specified).

select title from authors a, books b, table(b.authors) v where v.column_value = ref(a) group by title having count(*) > 1;

```
SQL> select p.name from publishers p, table(p.branches)group by p.name having count(*)>=all(select cou
nt(*) from publishers p, table(p.branches)group by  name);

NAME
----------
Lion
Monkey
Penguin
Eagle
Tiger
```

```
SQL> select a.name from authors a, books b, table(b.authors)b1 where b1.column_value=ref(a) group by a
.name having count(*) > 1;

NAME
---------------
Aman
Harry
Khushi
Akansha
Faisal
```

PRACTICAL 4:

AIM: Create a table that stores spatial data and issues queries on it.

1) Create tables

CREATE TABLE market(mkt_no number(5) PRIMARY KEY, mkt_name VARCHAR2(10),mkt_shape MDSYS.SDO_GEOMETRY);

2) Inserting values into the table

A) Creating abc area

INSERT INTO market VALUES(1,'abc', MDSYS.SDO_GEOMETRY(

```
    2003,
    NULL,
    NULL,
    MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
    MDSYS.SDO_ORDINATE_ARRAY(1,1,5,7)
))
```

B) Creating pqr area

```
INSERT INTO market VALUES(2,'pqr', MDSYS.SDO_GEOMETRY(
    2003,
    NULL,
    NULL,
    MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
    MDSYS.SDO_ORDINATE_ARRAY(1,3,4,7,8)
));
```

C) Creating mno area

```
INSERT INTO market VALUES(3,'mno', MDSYS.SDO_GEOMETRY(
    2003,
    NULL,
    NULL,
    MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),
    MDSYS.SDO_ORDINATE_ARRAY(1,4,7,2,0)
));
```

D) Creating xyz

```
INSERT INTO market VALUES(4,'xyz', MDSYS.SDO_GEOMETRY(
    2003,
    NULL,
```

NULL,

MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,4),

MDSYS.SDO_ORDINATE_ARRAY(8,7, 10,9, 8,11)

));

```
SQL> CREATE TABLE market(mkt_no number(5) PRIMARY KEY,mkt_name VARCHAR2(10),mkt_shape MDYSY.SDO_GEOMETRY);

Table created.

SQL> INSERT INTO market VALUES(1, 'abc', MDSYS.SDO_GEOMETRY(
  2  2003,
  3  NULL,
  4  NULL,
  5  MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
  6  MDSYS.SDO_ORDINATE_ARRAY(1,1,5,7)
  7  ))
  8  /

1 row created.

SQL> INSERT INTO market VALUES(2, 'pqr', MDSYS.SDO_GEOMETRY(
  2  2003,
  3  NULL,
  4  NULL,
  5  MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),
  6  MDSYS.SDO_ORDINATE_ARRAY(1,3,4,7,8)
  7  ));

1 row created.

SQL> INSERT INTO market VALUES(3, 'mno', MDSYS.SDO_GEOMETRY(
  2  2003,
  3  NULL,
  4  NULL,
  5  MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,3),
  6  MDSYS.SDO_ORDINATE_ARRAY(1,4,7,2,0)
  7  ));

1 row created.

SQL> INSERT INTO market VALUES(4, 'xyz', MDSYS.SDO_GEOMETRY(
  2  2003,
  3  NULL,
  4  NULL,
  5  MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,4),
  6  MDSYS.SDO_ORDINATE_ARRAY(8,7,10,9,8,11)
  7  ));

1 row created.
```

3) Adding market in the USER_SDO_GEOM_METADATA. Doing this will allow us to query our spatial data.

INSERT INTO USER_SDO_GEOM_METADATA VALUES(

'market',

'mkt_shape',

MDSYS.SDO_DIM_ARRAY(

MDSYS.SDO_DIM_ELEMENT('X',0,20,0.005),

MDSYS.SDO_DIM_ELEMENT('Y',0,20,0.005)

),

NULL

);

```
SQL> INSERT INTO USER_SDO_GEOM_METADATA VALUES(
  2  'market',
  3  'mkt_shape',
  4  MDSYS.SDO_DIM_ARRAY(
  5  MDYSY.SDO_DIM_ELEMENT('X',0,20,0.005),
  6  MDYSY.SDO_DIM_ELEMENT('Y',0,20,0.005)
  7  ),
  8  NULL
  9  );

1 row created.
```

4) Create spatial index on market

CREATE INDEX market_spatial_index ON market(market_shape) INDEXTYPE IS MDSYS.SDO_SPATIAL_INDEX;

```
SQL>
SQL> CREATE INDEX market_spatial_index ON market(mkt_shape) INDEXTYPE IS MDSYS.SPATIAL_INDEX;

Index created.
```

5) Running queries

a) Return the topological intersection of two geometries.

SELECT SDO_GEOM.SDO_INTERSECTION(a.mkt_shape,b.mkt_shape,0.005) FROM market a, market b WHERE a.mkt_name = 'abc' AND b.mkt_name = 'mno';

```
SQL> SELECT SDO_GEOM.SDO_INTERSECTION(a.mkt_shape, b.mkt_shape,0.005) FROM market a, market b WHERE a.mkt_name = 'abc'AND b.mkt_name = 'pqr';

SDO_GEOM.SDO_INTERSECTION(A.MKT_SHAPE,B.MKT_SHAPE,0.005)(SDO_GTYPE, SDO_SRID, SD
--------------------------------------------------------------------------------
SDO_GEOMETRY(2002, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 2, 1), SDO_ORDINATE_ARRAY(
5, 1, 5, 7))


SQL> SELECT SDO_GEOM.SDO_INTERSECTION(a.mkt_shape,b.mkt_shape,0.005) FROM market a, market b WHERE a.mkt_name = 'abc' AND b.mkt_name = 'mno';

SDO_GEOM.SDO_INTERSECTION(A.MKT_SHAPE,B.MKT_SHAPE,0.005)(SDO_GTYPE, SDO_SRID, SD
--------------------------------------------------------------------------------
SDO_GEOMETRY(2003, NULL, NULL, SDO_ELEM_INFO_ARRAY(1, 1003, 1), SDO_ORDINATE_ARR
AY(4, 5, 3, 3, 5, 3, 5, 5, 4, 5))
```

b) Do two geometries have any spatial relationship?

SELECT SDO_GEOM.RELATE(a.mkt_shape,'anyinteract',b.mkt_shape,0.005) FROM market a, market b WHERE a.mkt_name = 'abc' AND b.mkt_name = 'pqr';

SELECT SDO_GEOM.RELATE(a.mkt_shape,'anyinteract',b.mkt_shape,0.005) FROM market a, market b WHERE a.mkt_name = 'abc' AND b.mkt_name = 'xyz';

```
SQL> SELECT SDO_GEOM.RELATE(a.mkt_shape,'anyinteract',b.mkt_shape,0.005) FROM market a, market b WHERE a.mkt_name = 'abc' AND b.mkt_name = 'pqr';

SDO_GEOM.RELATE(A.MKT_SHAPE,'ANYINTERACT',B.MKT_SHAPE,0.005)
--------------------------------------------------------------------------------
TRUE

SQL> SELECT SDO_GEOM.RELATE(a.mkt_shape,'anyinteract',b.mkt_shape,0.005) FROM market a, market b WHERE a.mkt_name = 'abc' AND b.mkt_name = 'xyz';

SDO_GEOM.RELATE(A.MKT_SHAPE,'ANYINTERACT',B.MKT_SHAPE,0.005)
--------------------------------------------------------------------------------
FALSE
```

c) Return the areas of all cola markets.

SELECT SDO_GEOM.SDO_AREA(mkt_shape,0.005) FROM market;

```
SQL> SELECT SDO_GEOM.SDO_AREA(mkt_shape,0.005) FROM market;

SDO_GEOM.SDO_AREA(MKT_SHAPE,0.005)
----------------------------------
                                24
                              16.5
                                 5
                         12.5663706
```

d) Return the area of just cola_a.

SELECT SDO_GEOM.SDO_AREA(a.mkt_shape,0.005) from market a WHERE a.mkt_name = 'pqr';

```
DO_GEOM.SDO_AREA(A.MKT_SHAPE,0.005)
-----------------------------------
                              16.5
```

PRACTICAL 5:

AIM: Create a table that stores temporal data and issues queries on it.

CREATING TABLES

CREATE TABLE emp_data(emp_id number(5) PRIMARY KEY, emp_name char(10), join_date date, retire_date date);

INSERTING VALUES

INSERT INTO emp_data VALUES(1,'xyz',to_date(sysdate,'DD-MM-YYYY'), to_date('2-3-2039','DD-MM-YYYY'));

INSERT INTO emp_data VALUES(2,'xyz2',to_date('19-2-1980','DD-MM-YYYY'), to_date(sysdate,'DD-MM-YYYY'));

INSERT INTO emp_data VALUES(3,'xyz3',to_date(sysdate,'DD-MM-YYYY'), to_date('4-5-2040','DD-MM-YYYY'));

INSERT INTO emp_data VALUES(4,'xyz4',to_date('24-2-2001','DD-MM-YYYY'), to_date(sysdate,'DD-MM-YYYY'));

INSERT INTO emp_data VALUES(5,'xyz5',to_date(sysdate,'DD-MM-YYYY'), to_date('24-2-2039','DD-MM-YYYY'));

INSERT INTO emp_data VALUES(6,'xyz6',to_date(sysdate,'DD-MM-YYYY'), to_date('2-12-2069','DD-MM-YYYY'));

INSERT INTO emp_data VALUES(7,'xyz7',to_date('24-2-2001','DD-MM-YYYY'), to_date('2-3-2039','DD-MM-YYYY'));

INSERT INTO emp_data VALUES(8,'xyz8',to_date('24-3-2001','DD-MM-YYYY'), to_date('2-4-2039','DD-MM-YYYY'));

INSERT INTO emp_data VALUES(9,'xyz9',to_date('24-4-2001','DD-MM-YYYY'), to_date('2-5-2039','DD-MM-YYYY'));

INSERT INTO emp_data VALUES(10,'xyz10',to_date('24-5-2001','DD-MM-YYYY'), to_date('2-6-2039','DD-MM-YYYY'));

```
SQL> CREATE TABLE emp_data(emp_id number(5) PRIMARY KEY, emp_name char(10), join_date date, retire_date date);

Table created.

SQL> INSERT INTO emp_data VALUES(1,'xyz',to date(sysdate,'DD-MM-YYYY'),to_date('2-3-2039','DD-MM-YYYY'));
INSERT INTO emp_data VALUES(1,'xyz',to date(sysdate,'DD-MM-YYYY'),to_date('2-3-2039','DD-MM-YYYY'))
                                   *
ERROR at line 1:
ORA-00936: missing expression


SQL> INSERT INTO emp_data VALUES(1,'xyz',to_date(sysdate,'DD-MM-YYYY'),to_date('2-3-2039','DD-MM-YYYY'));

1 row created.

SQL> INSERT INTO emp_data VALUES(2,'xyz2',to_date('19-2-1980','DD-MM-YYYY'),to_date(sysdate,'DD-MM-YYYY'));

1 row created.

SQL> INSERT INTO emp_data VALUES(3,'xyz3',to_date(sysdate,'DD-MM-YYYY'),to_date('4-5-2040','DD-MM-YYYY'));

1 row created.
```

```
SQL> select * from emp_data;

    EMP_ID EMP_NAME   JOIN_DATE RETIRE_DA
---------- ---------- --------- ---------
         1 xyz        26-DEC-22 02-MAR-39
         2 xyz2       19-FEB-80 26-DEC-22
         3 xyz3       26-DEC-22 04-MAY-40
         4 xyz4       24-FEB-01 26-DEC-22
         5 xyz5       26-DEC-22 24-FEB-39
         6 xyz6       26-DEC-22 02-DEC-69
         7 xyz7       24-FEB-01 02-MAR-39
         8 xyz8       24-MAR-01 02-APR-39
         9 xyz9       24-APR-01 02-MAY-39
        10 xyz10      24-MAY-01 02-JUN-39

10 rows selected.
```

CREATING TABLES AND INSERTING DATA

alter session

set nls_date_format = 'HH24:MI';

CREATE TABLE share_det(cust_name varchar(10),no_shares number(10),cost number(10,2), time_of_tran date);

INSERT INTO share_det VALUES('xyz',10,1000,to_date(sysdate,'HH24:Mi'));

INSERT INTO share_det VALUES('xyz1',10,1000,to_date(sysdate,'HH24:Mi'));

INSERT INTO share_det VALUES('xyz2',10,1000,to_date(sysdate,'HH24:Mi'));

INSERT INTO share_det VALUES('xyz3',10,1000,to_date(sysdate,'HH24:Mi'));

```
SQL> CREATE TABLE share_det(cust_name varchar(10), no_shares number(10),cost number(10,2), time_of_tran date);

Table created.

SQL> INSERT INTO share_det VALUES('xyz',10,1000,to_date(sysdate,'HH24:Mi'));

1 row created.

SQL> INSERT INTO share_det VALUES('xyz1',10,1000,to_date(sysdate,'HH24:Mi'));

1 row created.

SQL> INSERT INTO share_det VALUES('xyz2',10,1000,to_date(sysdate,'HH24:Mi'));

1 row created.

SQL> INSERT INTO share_det VALUES('xyz3',10,1000,to_date(sysdate,'HH24:Mi'));

1 row created.
```

```
SQL> select * from share_det;

CUST_NAME   NO_SHARES       COST TIME_
---------- ---------- ---------- -----
xyz                10       1000 12:21
xyz1               10       1000 12:21
xyz2               10       1000 12:22
xyz3               10       1000 12:22
```

QUERYING

1) Selecting employees going to retire on 2-3-2039

SELECT emp_name from emp_data where retire_date = to_date('2-3-2039','DD-MM-YYYY');

2) Select employees who have started in 24-2-2001

SELECT emp_name from emp_data where join_date = to_date('24-2-2001','DD-MM-YYYY');

```
SQL> SELECT emp_name from emp_data WHERE retire_date = to_date('2-3-2039','DD-MM-YYYY');

EMP_NAME
----------
xyz
xyz7

SQL> SELECT emp_name from emp_data WHERE join_date = to_date('24-2-2001','DD-MM-YYYY');

EMP_NAME
----------
xyz4
xyz7
```

3) Find all the names of a company whose share price is more than Rs. 900 at 15:49 A.M.

SELECT cust_name FROM share_det where(no_shares*cost)>900 AND time_of_tran = to_date('15:49','HH24:Mi');

PRACTICAL 6:

AIM: Demonstrate the Accessing and Storing and performing CRUD operations in a. MongoDB

BASIC COMMANDS

1) Seeing all the existing database:

show dbs

2) Switching/Creating a database use <dbname>

use students

3) Seeing the collections in the database

show collections

4) Dropping the table

db.dropDatabase()

```
test> show dbs
FskDb   144.00 KiB
admin    40.00 KiB
config   72.00 KiB
fsk2Db   72.00 KiB
local    72.00 KiB
test> use students
switched to db students
students> show collections

students> db.dropDatabase()
{ ok: 1, dropped: 'students' }
```

STEP 1 creating and entering a database

use studentInfo

STEP 2 Creating a collection and inserting values in it

db.studentInfo.insertMany([{studentName:’Akansha’,studentLname:'Kotian',dob:"24-dec-2001",pincode:400060},

{studentName:'Fsk',studentLname:'Shaikh',dob:"24-aug-2001",pincode:400088},

{studentName:'Aman',studentLname:'Dandekar',dob:"17-dec-2001",pincode:400070},

{studentName:'Harry',studentLname:'Sapariya',dob:"8-oct-2001",pincode:400070}])

```
students> use studentInfo
switched to db studentInfo
studentInfo> db.studentInfo.insertMany([{studentName:'Akansha',studentLname:'Kotian',dob:"24-dec-2001",pincode:400060},
... {studentName:'Fsk',studentLname:'Shaikh',dob:"24-aug-2001",pincode:400088},
... {studentName:'Aman',studentLname:'Dandekar',dob:"17-dec-2001",pincode:400070},
... {studentName:'Harry',studentLname:'Sapariya',dob:"8-oct-2001",pincode:400070}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("63a949ba7a1cbe21e78eee31"),
    '1': ObjectId("63a949ba7a1cbe21e78eee32"),
    '2': ObjectId("63a949ba7a1cbe21e78eee33"),
    '3': ObjectId("63a949ba7a1cbe21e78eee34")
  }
}
```

STEP 3: Performing read operations

a) Reading all the data in a collection

db.studentInfo.find()

```
studentInfo> db.studentInfo.find()
[
  {
    _id: ObjectId("63a949ba7a1cbe21e78eee31"),
    studentName: 'Akansha',
    studentLname: 'Kotian',
    dob: '24-dec-2001',
    pincode: 400060
  },
  {
    _id: ObjectId("63a949ba7a1cbe21e78eee32"),
    studentName: 'Fsk',
    studentLname: 'Shaikh',
    dob: '24-aug-2001',
    pincode: 400088
  },
  {
    _id: ObjectId("63a949ba7a1cbe21e78eee33"),
    studentName: 'Aman',
    studentLname: 'Dandekar',
    dob: '17-dec-2001',
    pincode: 400070
  },
  {
    _id: ObjectId("63a949ba7a1cbe21e78eee34"),
    studentName: 'Harry',
    studentLname: 'Sapariya',
    dob: '8-oct-2001',
    pincode: 400070
  }
]
```

b) Find info about student whose name is "Soham" or "Sanjana"

db.studentInfo.find({

$or:[{studentName:"Akansha"},

{studentName:"Fsk"}]

}).pretty()

```
studentInfo> db.studentInfo.find({$or:[{studentName:"Akansha"},{studentName:"Fsk"}]}).pretty()
[
  {
    _id: ObjectId("63a949ba7a1cbe21e78eee31"),
    studentName: 'Akansha',
    studentLname: 'Kotian',
    dob: '24-dec-2001',
    pincode: 400060
  },
  {
    _id: ObjectId("63a949ba7a1cbe21e78eee32"),
    studentName: 'Fsk',
    studentLname: 'Shaikh',
    dob: '24-aug-2001',
    pincode: 400088
  }
]
```

c) Count the number of documents who have the pincode 400045

db.studentInfo.countDocuments({pincode:400088})

```
studentInfo> db.studentInfo.countDocuments({pincode:400088})
1
studentInfo>
```

Step 4: Perform Update operations

a) Update harry's birthday to 14-oct-2001

db.studentInfo.updateOne(

{studentName:'Harry'},

{$set:{dob:"14-oct-2001"}}

)

```
studentInfo> db.studentInfo.updateOne({studentName:'Harry'},
... {$set:{dob:"14-oct-2001"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Step 5: Perform Delete Operation

a) Delete document of student named "Harry"

db.studentInfo.deleteOne({studentName:"Harry"})

```
studentInfo> db.studentInfo.deleteOne({studentName:"Harry"})
{ acknowledged: true, deletedCount: 1 }
```

b) Delete all documents with pincode 400086 or 400045

db.studentInfo.deleteMany({$or:[

{pincode:400086},

{pincode:400045}

]

})

```
studentInfo> db.studentInfo.deleteMany({$or:[
... {pincode:400070},
... {pincode:400070}]})
{ acknowledged: true, deletedCount: 1 }
studentInfo>
```

HBASE Practical

Step1: Install VM Oracle. intall Ubuntu on VM oracle

Installing Java

1. In Ubuntu, open the terminal and enter the following command:
   $ sudo apt install default-jdk default-jre -y
2. Verify the installation via
   $ java -version

Create new user Hadoop and configure password-less SSH

1. Create the new user
   $ sudo adduser hadoop
2. Add the hadoop user to the sudo group.
   $ sudo usermod -aG sudo hadoop
3. Log out of the current user and switch to the newly created Hadoop user
4. In terminal, install openssh client and server
   $ apt install openssh-server openssh-client -y
5. Generate public and private key pairs.
   $ ssh-keygen -t rsa
6. Add the generated public key from id_rsa.pub to authorized_keys.
   $ sudo cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
7. Change the permissions of the authorized_keys file.
   $ sudo chmod 640 ~/.ssh/authorized_keys
8. Verify if the password-less SSH is functional.
   $ ssh localhost

 # Install Apache Hadoop

1. Download the latest stable version of Hadoop. To get the latest version, go to ApacheHadoop official download page.
2. Extract the downloaded file.
   $ tar -xvzf hadoop-3.3.1.tar.gz
3. Move the extracted directory to the /usr/local/ directory.
   $ sudo mv hadoop-3.3.1 /usr/local/hadoop
4. Create directory to store system logs.
   $ sudo mkdir /usr/local/hadoop/logs
5. Change the ownership of the hadoop directory.
   $ sudo chown -R hadoop:hadoop /usr/local/Hadoop

 # Configure Hadoop

1. Edit file ~/.bashrc to configure the Hadoop environment variables.
   $ sudo nano ~/.bashrc
2. Add the following lines to the file. Save and close the file.

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_INSTALL=$HADOOP_HOME

export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME


export YARN_HOME=$HADOOP_HOME

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/nativeexport
PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

3. Add the changes to the memory
   $ source ~/.bashrc

 # Configure Java Environment Variables

1. Find the OpenJDK directory and copy the output
   $ readlink -f /usr/bin/javac
2. Edit the hadoop-env.sh file.
   $ sudo nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
3. Add the following lines to the file. Then, close and save the file.

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
export HADOOP_CLASSPATH+=" $HADOOP_HOME/lib/*.jar"
```

4. Browse to the hadoop lib directory.
   $ cd /usr/local/hadoop/lib
5. Download the Javax activation file.
   $ sudo wget https://jcenter.bintray.com/javax/activation/javax.activation-

api/1.2.0/javax.activation-api-1.2.0.jar

6. Verify the Hadoop version.

```
$ hadoop version
```

7. Edit the core-site.xml configuration file to specify the URL for your NameNode.

```
$ sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml
```

8. Add the following lines. Save and close the file.

```
<configuration>
    <property>
        <name>fs.default.name</name>
        <value>hdfs://0.0.0.0:9000</value>
        <description>The default file system URI</description>
    </property>
</configuration>
```

9. Create a directory for storing node metadata and change the ownership to hadoop.

```
$ sudo mkdir -p /home/hadoop/hdfs/{namenode,datanode}
$ sudo chown -R hadoop:hadoop /home/hadoop/hdfs
```

10. Edit hdfs-site.xml configuration file to define the location for storing node metadata,fs-image file.

```
$ sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

11. Add the following lines. Close and save the file.

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>
    <property>
        <name>dfs.name.dir</name>
        <value>file:///home/hadoop/hdfs/namenode</value>
    </property>
    <property>
        <name>dfs.data.dir</name>
        <value>file:///home/hadoop/hdfs/datanode</value>
    </property>
    <property>
        <name>dfs.permissions.enabled</name>
        <value>false</value>
    </property>
</configuration>
```

12. Edit mapred-site.xml configuration file to define MapReduce values.

```
$ sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

13. Add the following lines. Save and close the file.

```
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
```

```
        <name>yarn.app.mapreduce.am.env</name>
        <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
        <description>Change this to your hadoop location.</description>
    </property>
    <property>
        <name>mapreduce.map.env</name>
        <value>HADOOP_MAPRED_HOME=/usr/local/hadoop</value>
        <description>Change this to your hadoop location.</description>
    </property>
    <property>
        <name>mapreduce.reduce.env</name>
        <value>HADOOP_MAPRED_HOME=/usr/local/hadoop/<value>
        <description>Change this to your hadoop location.</description>

    </property>
</configuration>
```

14. Edit the yarn-site.xml configuration file and define YARN-related settings.
    $ sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
15. Add the following lines. Save and close the file.

```
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

16. Validate the Hadoop configuration and format the HDFS NameNode.
    $ hdfs namenode -format

 # Start the Apache Hadoop Cluster

1. Start the NameNode and DataNode.
   $ start-dfs.sh
2. Start the YARN resource and node managers.
   $ start-yarn.sh
3. Verify all the running components.
   $ jps

 # Access Apache Hadoop Web Interface

In your web browser enter the URLs given below to access the interface

http://localhost:9870

http://localhost:8088

Implementation:

#HBase installation

1. Go to the official site and download the most stable version of hbase
2. Extract the zip file and place it in Hadoop home directory
3. Open hbase-env.sh in hbase/conf and assign the JAVA_HOME path

   $ sudo nano hbase/conf/hbase-env.sh

   export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

4. Edit the .bashrc file

   $ nano ~/.basrhc

   export
   HBASE_HOME=/home/hbuser/hbase
   export PATH= $PATH:$HBASE_HOME/bin

5. Read the edited bashrc file to the running memory

   $ source ~/.bashrc

6. Add the following properties below the existing ones in the hbase/conf/hbase-site.xml file

   <property>

   <name>hbase.rootdir</name>

   <value>file:///home/hbuser/hbase</value>

   </property>

   <property>

   <name>hbase.zookeeper.property.dataDir</name>

   <value>/home/hbuser/hbase</value>

   </property>

7. Run hbase by typing

   $ start-hbase.sh

   $ hbase shell

Storing and retrieval of data

Enter the following command in the hbase shell

- Create ‘emp’, ‘pri_data’, ‘pro_data’
- Put ‘emp’, ‘1’, ‘pri_data:name’, ‘Andy’
- e =get_table

- e.put ‘1’, ‘pri_data:age’, ‘22’
- e.put ‘1’, ‘pro_data:post’, ‘asst. manager’
- e.put ‘1’, ‘pro_data:salary’, ‘40k’
- e.put ‘2’, ‘pri_data:name’, ‘Icarus’
- e.put ‘2’, ‘pri_data:age’, ‘22’
- e.put ‘2’, ‘pro_data:post’, ‘manager’
- e.get ‘1’
- e.get ‘2’