

INDEX

SR. NO	NAME OF PRACTICAL	DATE	SIGN
1	Create different types that include attributes and methods. Define tables for these types by adding a sufficient number of tuples. Demonstrate insert, update and delete operations on these tables. Execute queries on them		
2	Create an XML database and demonstrate insert, update and delete operations on these tables. Issue queries on it		
3	Create a table that stores spatial data and issues queries on it		
4	Create a temporal database and issue queries on it.		
5	Demonstrate the Accessing and Storing and performing CRUD operations in 1. MongoDB 2. Redis		
6	Demonstrate the Accessing and Storing and performing CRUD operations in • Redis 2. Apache Cassandra		
7	Demonstrating MapReduce in MongoDB to count the number of female (F) and male (M) respondents in the database		
8	Demonstrate the indexing and ordering operations in 1. MongoDB 2. CouchDB 3. Apache Cassandra		

PRACTICAL 1

Create different types that include attributes and methods. Define tables for these types by adding a sufficient number of tuples. Demonstrate insert, update and delete operations on these tables. Execute queries on them

Code:

```
CREATE TYPE Address AS OBJECT (  
    city VARCHAR2(35),  
    houseno number(5)  
);  
/
```

```
CREATE TYPE Person AS OBJECT (  
    first_name VARCHAR2(15),  
    last_name VARCHAR2(15),  
    home_address Address,  
    phone_number VARCHAR2(15),  
    MEMBER PROCEDURE display_address ( SELF IN OUT NOCOPY Person )  
  
)  
/
```

```
CREATE TYPE BODY Person AS
```

```
    MEMBER PROCEDURE display_address ( SELF IN OUT NOCOPY Person ) IS  
    BEGIN  
        DBMS_OUTPUT.PUT_LINE(first_name || ' ' || last_name);  
        DBMS_OUTPUT.PUT_LINE(home_address .city||' '||home_address .houseno);  
  
    END;  
END;  
/
```

```
CREATE TABLE Family1 OF Person;  
/  
SELECT * FROM Family1;  
/
```

```
DECLARE  
    p1 Person; -- p1 & p2 is atomically null  
    p2 Person;  
BEGIN  
    -- call the constructor for Person_type  
    p1 := Person('Akshay', 'Patil',Address('Pune',400),'987685434');  
    INSERT INTO Family1 VALUES(p1);  
  
    DBMS_OUTPUT.PUT_LINE(p1.first_name || ' ' || p1.last_name||' '||p1.home_address.city); -- display  
    details  
    p2 := Person('Megha', 'Joshi',Address('Nashik',500),'987685987');  
  
    INSERT INTO Family1 VALUES(p2);  
    DBMS_OUTPUT.PUT_LINE(p2.first_name || ' ' || p2.last_name||' '||p2.home_address.city); -- display  
    details
```

```

END;
/

DECLARE
  p1 Person;
  p2 Person;

BEGIN
  SELECT VALUE(f) INTO p1 FROM Family1 f
    WHERE f.last_name = 'Joshi';
  --- DBMS_OUTPUT.PUT_LINE(p1.first_name || ' ' || p1.last_name); -- display details
  p1.display_address();

  p2 := p1;
  p1.last_name := 'Kale';
  DBMS_OUTPUT.PUT_LINE(p1.first_name || ' ' || p1.last_name); -- display details

END;
/

SELECT * FROM Family1;
/

BEGIN
  UPDATE Family1 f SET f.first_name = 'Meenakshi'
    WHERE f.last_name = 'Joshi';

  END;
/
SELECT * FROM Family1;
/

BEGIN
  DELETE FROM Family1 f
    WHERE f.first_name = 'Akshay';
END;
/
SELECT * FROM Family1;
/

```

Output:

Type created.
Type created.
Type created.
Table created.
no data found

INSERT:

Statement processed.
Akshay Patil Pune
Megha Joshi Nashik

Statement processed.

Megha Joshi
Nashik 500
Megha Kale

READ:

FIRST_NAME	LAST_NAME	HOME_ADDRESS	PHONE_NUMBER
Akshay	Patil	[unsupported data type]	987685434
Megha	Joshi	[unsupported data type]	987685987

UPDATE:

FIRST_NAME	LAST_NAME	HOME_ADDRESS	PHONE_NUMBER
Akshay	Patil	[unsupported data type]	987685434
Meenakshi	Joshi	[unsupported data type]	987685987

DELETE:

Statement processed.

FIRST_NAME	LAST_NAME	HOME_ADDRESS	PHONE_NUMBER
Meenakshi	Joshi	[unsupported data type]	987685987

PRACTICAL 2

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

Create table

```
CREATE TABLE EmployeeData OF xmltype
```

```
/
```

INSERT Record

```
BEGIN
```

```
INSERT INTO EmployeeData
```

```
VALUES ('<EmployeeData>'
```

```
    ||'<employee id="346">'
```

```
    ||'<firstname>Darren</firstname>'
```

```
    ||'<lastname>Dsouza</lastname>'
```

```
    ||'<hireDate>2/8/22</hireDate>'
```

```
    ||'</employee>'
```

```
    ||'<employee id="123">'
```

```
    ||'<firstname>Manoj</firstname>'
```

```
    ||'<lastname>Dsouza</lastname>'
```

```
    ||'<hireDate>5/8/22</hireDate>'
```

```
    ||'</employee>'
```

```
    ||'</EmployeeData>');
```

```
COMMIT;
```

```
END;
```

```
/
```

READ

```
SELECT xmlquery(
```

```
    '<Summary lineItemCount="{ count($XML/EmployeeData/employee) }">{'
```

```
    $XML/EmployeeData/employee
```

```
    }
```

```
    </Summary>'
```

```
    passing object_value AS "XML"
```

```
    returning content
```

```
    ).getclobval() initial_state
```

```
FROM EmployeeData
```

```
/
```

UPDATE

UPDATE EmployeeData

```
SET object_value = XMLQuery
(
    'copy $NEWXML := $XML modify (
        for $ED in $NEWXML/EmployeeData/employee[@id="123"] return (
            replace value of node $ED/firstname with $firstname1
        )
    )
    return $NEWXML'
    passing object_value as "XML",
    'Danny' as "firstname1"
    returning content
)
WHERE xmlExists(
'$XML/EmployeeData/employee/hiredate=$REF'
    passing object_value as "XML",
    '5/8/22' as "REF"
)
/
```

DELETE

UPDATE EmployeeData

```
SET object_value = XMLQuery
(
    'copy $NEWXML := $XML modify (
        delete nodes $NEWXML/EmployeeData/employee[@id=$id]
    )
    return $NEWXML'
    passing object_value as "XML",
    '123' as "id"
    returning content )
```

PRACTICAL 3

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

Code:

```
CREATE TABLE cola_markets (  
  mkt_id NUMBER PRIMARY KEY,  
  name VARCHAR2(32),  
  shape SDO_GEOMETRY);  
  
INSERT INTO cola_markets VALUES(  
  1,  
  'cola_a',  
  SDO_GEOMETRY(  
    2003, -- two-dimensional polygon  
    NULL,  
    NULL,  
    SDO_ELEM_INFO_ARRAY(1,1003,3), -- one rectangle (1003 = exterior)  
    SDO_ORDINATE_ARRAY(1,1, 5,7) -- only 2 points needed to  
      -- define rectangle (lower left and upper right) with  
      -- Cartesian-coordinate data  
  )  
);  
  
INSERT INTO cola_markets VALUES(  
  2,  
  'cola_b',  
  SDO_GEOMETRY(  
    2003, -- two-dimensional polygon  
    NULL,  
    NULL,  
    SDO_ELEM_INFO_ARRAY(1,1003,1), -- one polygon (exterior polygon ring)  
    SDO_ORDINATE_ARRAY(5,1, 8,1, 8,6, 5,7, 5,1)  
  )  
);  
  
INSERT INTO cola_markets VALUES(  
  3,  
  'cola_c',  
  SDO_GEOMETRY(  
    2003, -- two-dimensional polygon  
    NULL,  
    NULL,  
    SDO_ELEM_INFO_ARRAY(1,1003,1), -- one polygon (exterior polygon ring)  
    SDO_ORDINATE_ARRAY(3,3, 6,3, 6,5, 4,5, 3,3)  
  )  
);  
  
INSERT INTO cola_markets VALUES(  
  4,  
  'cola_d',  
  SDO_GEOMETRY(  
    2003, -- two-dimensional polygon  
    NULL,
```

```

NULL,
SDO_ELEM_INFO_ARRAY(1,1003,4), -- one circle
SDO_ORDINATE_ARRAY(8,7, 10,9, 8,11)
)
);
CREATE INDEX cola_spatial_idx
ON cola_markets(shape)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;

SELECT SDO_GEOM.SDO_INTERSECTION(c_a.shape, c_c.shape, 0.005)
FROM cola_markets c_a, cola_markets c_c
WHERE c_a.name = 'cola_a' AND c_c.name = 'cola_c';

-- Do two geometries have any spatial relationship?
SELECT SDO_GEOM.RELATE(c_b.shape, 'anyinteract', c_d.shape, 0.005)
FROM cola_markets c_b, cola_markets c_d
WHERE c_b.name = 'cola_b' AND c_d.name = 'cola_d';

-- Return the areas of all cola markets.
SELECT name, SDO_GEOM.SDO_AREA(shape, 0.005) FROM cola_markets;

-- Return the area of just cola_a.
SELECT c.name, SDO_GEOM.SDO_AREA(c.shape, 0.005) FROM cola_markets c
WHERE c.name = 'cola_a';

-- Return the distance between two geometries.
SELECT SDO_GEOM.SDO_DISTANCE(c_b.shape, c_d.shape, 0.005)
FROM cola_markets c_b, cola_markets c_d
WHERE c_b.name = 'cola_b' AND c_d.name = 'cola_d';

-- Is a geometry valid?
SELECT c.name, SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(c.shape, 0.005)
FROM cola_markets c WHERE c.name = 'cola_c';

-- Is a layer valid? (First, create the results table.)
CREATE TABLE val_results (sdo_rowid ROWID, result VARCHAR2(2000));
CALL SDO_GEOM.VALIDATE_LAYER_WITH_CONTEXT('COLA_MARKETS', 'SHAPE',
'VAL_RESULTS', 2);
SELECT * from val_results;

```

Output:

```
SDO_GEOM.SDO_INTERSECTION(C_A.SHAPE,C_C.SHAPE,0.005)
```

```
{"polygon" : {"boundary" : [{"line" : {"datapoints" : [[5,3],[5,5],[4,5],[3,3],[5,3]]}}]}}
```

```
SDO_GEOM.RELATE(C_B.SHAPE, 'ANYINTERACT', C_D.SHAPE, 0.005)
```

```
FALSE
```


NAME	SDO_GEOM.SDO_AREA(SHAPE,0.005)
cola_a	24
cola_b	16.5
cola_c	5
cola_d	12.5663706143592

SDO_GEOM.SDO_DISTANCE(C_B.SHAPE,C_D.SHAPE,0.005)
.846049894151541

NAME	SDO_GEOM.VALIDATE_GEOMETRY_WITH_CONTEXT(C.SHAPE,0.005)
cola_c	TRUE

SDO_ROWID	RESULT
-	Rows Processed <4>

PRACTICAL 4

Create a temporal database and issue queries on it.

Statement 1:

Create the EMPLOYEES_TRACK_TIME table

```
CREATE TABLE employees_track_time (  
    employee_id NUMBER(6) NOT NULL, first_name VARCHAR2(20), last_name VARCHAR2(25) NOT  
NULL,  
    email VARCHAR2(25) NOT NULL, phone_number VARCHAR2(20), hire_date DATE NOT NULL,  
    job_id VARCHAR2(10) NOT NULL, salary NUMBER(8,2), commission_pct NUMBER(2,2),  
    manager_id NUMBER(6), department_id NUMBER(4),  
    PERIOD FOR emp_track_time)
```

Output:

Table created.

Statement 2:

```
SELECT SUBSTR(COLUMN_NAME,1,22) NAME, SUBSTR(DATA_TYPE,1,28) DATA_TYPE, COLUMN_ID AS  
COL_ID,  
    SEGMENT_COLUMN_ID AS SEG_COL_ID, INTERNAL_COLUMN_ID AS INT_COL_ID,  
HIDDEN_COLUMN  
FROM USER_TAB_COLS WHERE TABLE_NAME='EMPLOYEES_TRACK_TIME'
```

Output:

NAME	DATA_TYPE	COL_ID D	SEG_COL_I D	INT_COL_I D	HIDDEN_COLUM N
EMP_TRACK_TIME_START	TIMESTAMP(6)) WITH TIME ZONE	-	1	1	YES
EMP_TRACK_TIME_END	TIMESTAMP(6)) WITH TIME ZONE	-	2	2	YES
EMP_TRACK_TIME	NUMBER	-	-	3	YES
EMPLOYEE_ID	NUMBER	1	3	4	NO
FIRST_NAME	VARCHAR2	2	4	5	NO
LAST_NAME	VARCHAR2	3	5	6	NO
EMAIL	VARCHAR2	4	6	7	NO
PHONE_NUMBER	VARCHAR2	5	7	8	NO
HIRE_DATE	DATE	6	8	9	NO
JOB_ID	VARCHAR2	7	9	10	NO
SALARY	NUMBER	8	10	11	NO
COMMISSION_PCT	NUMBER	9	11	12	NO

MANAGER_ID	NUMBER	10	12	13	NO
DEPARTMENT_ID	NUMBER	11	13	14	NO

Statement 3:

```
INSERT INTO employees_track_time (emp_track_time_start, emp_track_time_end, employee_id,
first_name,
last_name, email, hire_date, job_id, salary, manager_id, department_id)
VALUES (TIMESTAMP '2009-06-01 12:00:01 Europe/Paris',
TIMESTAMP '2012-11-30 12:00:01 Europe/Paris', 251, 'Scott', 'Tiger',
'scott.tiger@example.com', DATE '2009-05-21', 'IT_PROG', 60000, 103, 60)
```

Output:

1 row(s) inserted.

Statement 4:

```
INSERT INTO employees_track_time (emp_track_time_start, emp_track_time_end, employee_id,
first_name,
last_name, email, hire_date, job_id, salary, manager_id, department_id)
VALUES (TIMESTAMP '2009-06-01 12:00:01 Europe/Paris',
TIMESTAMP '2012-12-31 12:00:01 Europe/Paris', 252, 'Jane', 'Lion',
'jane.lion@example.com', DATE '2009-06-11', 'IT_PROG', 60000, 103, 60)
```

Output:

1 row(s) inserted.

Statement 5:

```
INSERT INTO employees_track_time (emp_track_time_start, emp_track_time_end, employee_id,
first_name,
last_name, email, hire_data, job_id, salary, manager_id, department_id)
VALUES (TIMESTAMP '2011-07-01 12:00:01 Europe/Paris',
TIMESTAMP '2014-08-31 12:00:01 Europe/Paris', 253, 'Marie', 'Smith',
'marie.smith@example.com', DATE '2011-06-10', 'IT_PROG', 60000, 103, 60)
```

Output:

1 row(s) inserted.

Statement 6:

```
UPDATE employees_track_time set manager_id = 105
WHERE emp_track_time_start <= TIMESTAMP '2009-06-01 12:00:01 Europe/Paris'
```

Output:

2 row(s) updated.

Statement 7:

```
SELECT employee_id, last_name, first_name, manager_id FROM employees_track_time
```

Output:

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	MANAGER_ID
251	Tiger	Scott	105
252	Lion	Jane	105

253	Smith	Marie	103
-----	-------	-------	-----

Statement 8:

```
DELETE employees_track_time WHERE emp_track_time_end < TIMESTAMP '2001-12-31 12:00:01 Europe/Paris'
```

Output:

0 row(s) deleted.

Statement 9:

```
SELECT employee_id FROM employee_track_time
      WHERE emp_track_time_start > TIMESTAMP '2009-05-31 12:00:01 Europe/Paris' AND
            emp_track_time_end < TIMESTAMP '2012-12-01 12:00:01 Europe/Paris'
```

Output:

EMPLOYEE_ID
251

Statement 10:

```
SELECT employee_id FROM employee_track_time AS OF PERIOD FOR
      emp_track_time TIMESTAMP '2012-12-01 12:00:01 Europe/Paris'
```

Output:

EMPLOYEE_ID
252
253

PRACTICAL 5

Demonstrate the Accessing and Storing and performing CRUD operations in 1. MongoDB

```
> show dbs
```

```
admin    0.000 GB
config   0.000 GB
local    0.000 GB
test     0.000 GB
```

```
> use students
```

```
switched to db students
```

```
> show dbs
```

```
admin  0.000 GB
config 0.000 GB
local  0.000 GB
test   0.000 GB
```

```
> db.studentsData.insertOne({"name": "Vrushali", "contact": "98765423", "Course": "CS"})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("63b92171d2f8ccbe65c46ce0")
}
```

```
> db.studentsData.find()
{ "_id" : ObjectId("63b92171d2f8ccbe65c46ce0"), "name" : "Vrushali", "contact" : "98765423",
"Course" : "CS" }
```

```
> db.studentsData.insertOne({"name": "Yogesh", "contact": "778634783", "Course": "CIVIL"},
{"name": "Atul", "contact": "2863468", "Course": "MECH"})
{
  "acknowledged" : true,
  "insertedId" : ObjectId("63b9243ad2f8ccbe65c46ce1")
}
```

```
> db.studentsData.find()
{ "_id" : ObjectId("63b92171d2f8ccbe65c46ce0"), "name" : "Vrushali", "contact" : "98765423",
"Course" : "CS" }
{ "_id" : ObjectId("63b9243ad2f8ccbe65c46ce1"), "name" : "Yogesh", "contact" : "778634789",
"Course" : "CIVIL" }
> db.studentsData.insertMany([{"name": "Yogesh", "contact": "778634783", "Course": "CIVIL"},
{"name": "Atul", "contact": "2863468", "Course": "MECH"}])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("63ba3c9e3741a427f994b067"),
    ObjectId("63ba3c9e3741a427f994b068")
  ]
}
> db.studentsData.insertOne({"name": "Asha", "contact": "9923875"})
{
```

```
    "acknowledged" : true,
    "insertedId" : ObjectId("63ba3cc73741a427f994b069")
}
```

```
> db.studentsData.find()
{ "_id" : ObjectId("63b92171d2f8ccbe65c46ce0"), "name" : "Vrushali", "contact" : "98765423",
  "Course" : "CS" }
{ "_id" : ObjectId("63ba3c9e3741a427f994b067"), "name" : "Yogesh", "contact" : "778634783",
  "Course" : "CIVIL" }
{ "_id" : ObjectId("63ba3c9e3741a427f994b068"), "name" : "Atul", "contact" : "2863468",
  "Course" : "MECH" }
{ "_id" : ObjectId("63ba3cc73741a427f994b069"), "name" : "Asha", "contact" : "9923875" }
```

```
> db.studentsData.find().pretty()
{
  "_id" : ObjectId("63b92171d2f8ccbe65c46ce0"),
  "name" : "Vrushali",
  "contact" : "98765423",
  "Course" : "CS"
}
{
  "_id" : ObjectId("63ba3c9e3741a427f994b067"),
  "name" : "Yogesh",
  "contact" : "778634783",
  "Course" : "CIVIL"
}
{
  "_id" : ObjectId("63ba3c9e3741a427f994b068"),
  "name" : "Atul",
  "contact" : "2863468",
  "Course" : "MECH"
}
{
  "_id" : ObjectId("63ba3cc73741a427f994b069"),
  "name" : "Asha",
  "contact" : "9923875"
}
```

```
> db.studentsData.insertOne({name:"Bhaskar",contact:"99234845",_id:"1234"}) //id has to be unique
{ "acknowledged" : true, "insertedId" : "1234" }
```

```
> db.studentsData.find()
{ "_id" : ObjectId("63b92171d2f8ccbe65c46ce0"), "name" : "Vrushali", "contact" : "98765423",
  "Course" : "CS" }
{ "_id" : ObjectId("63ba3c9e3741a427f994b067"), "name" : "Yogesh", "contact" : "778634783",
  "Course" : "CIVIL" }
{ "_id" : ObjectId("63ba3c9e3741a427f994b068"), "name" : "Atul", "contact" : "2863468",
  "Course" : "MECH" }
{ "_id" : ObjectId("63ba3cc73741a427f994b069"), "name" : "Asha", "contact" : "9923875" }
{ "_id" : "1234", "name" : "Bhaskar", "contact" : "99234845" }
```

```
> db.studentsData.find({name:"Atul"}) //Filtering Based on Criteria
{ "_id" : ObjectId("63ba3c9e3741a427f994b068"), "name" : "Atul", "contact" : "2863468",
  "Course" : "MECH" }
```

```

> db.studentsData.updateOne({name:"Yogesh"},{$set:{name:"Yogesh Patil"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.studentsData.find().pretty()
{
  "_id" : ObjectId("63b92171d2f8ccbe65c46ce0"),
  "name" : "Vrushali",
  "contact" : "98765423",
  "Course" : "CS"
}
{
  "_id" : ObjectId("63ba3c9e3741a427f994b067"),
  "name" : "Yogesh Patil",
  "contact" : "778634783",
  "Course" : "CIVIL"
}
{
  "_id" : ObjectId("63ba3c9e3741a427f994b068"),
  "name" : "Atul",
  "contact" : "2863468",
  "Course" : "MECH"
}
{
  "_id" : ObjectId("63ba3cc73741a427f994b069"),
  "name" : "Asha",
  "contact" : "9923875"
}
{ "_id" : "1234", "name" : "Bhaskar", "contact" : "99234845" }

> db.studentsData.updateOne({name:"Atul"},{$set:{Marks:[90,30]}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.studentsData.updateOne({name:"Atul"},{$set:{Address:{House_no:101, City:"Pune"}}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

> db.studentsData.updateMany({},{$set:{College:"SIES"}})
{ "acknowledged" : true, "matchedCount" : 5, "modifiedCount" : 5 }

> db.studentsData.aggregate({$addFields:{hobby:"Cricket"}})
{ "_id" : ObjectId("63b92171d2f8ccbe65c46ce0"), "name" : "Vrushali", "contact" : "98765423",
"Course" : "CS", "College" : "SIES", "hobby" : "Cricket" }
{ "_id" : ObjectId("63ba3c9e3741a427f994b067"), "name" : "Yogesh Patil", "contact" :
"778634783", "Course" : "CIVIL", "College" : "SIES", "hobby" : "Cricket" }
{ "_id" : ObjectId("63ba3c9e3741a427f994b068"), "name" : "Atul", "contact" : "2863468",
"Course" : "MECH", "Marks" : [ 90, 30 ], "Address" : { "House_no" : 101, "City" : "Pune" },
"College" : "SIES", "hobby" : "Cricket" }
{ "_id" : ObjectId("63ba3cc73741a427f994b069"), "name" : "Asha", "contact" : "9923875",
"College" : "SIES", "hobby" : "Cricket" }
{ "_id" : "1234", "name" : "Bhaskar", "contact" : "99234845", "College" : "SIES", "hobby" :
"Cricket" }

```

```
db.studentsData.deleteMany()
```

query executed successfully

db.studentDataDeleteOne()

query executed successfully

2. Redis

127.0.0.1:6379> set name flosia

OK

127.0.0.1:6379> set name flosia simon

(error) ERR syntax error

127.0.0.1:6379> set name 'flosia simon'

OK

127.0.0.1:6379> get name

"flosia simon"

127.0.0.1:6379> del name

(integer) 1

127.0.0.1:6379> mset names flosia color yellow rating 10

OK

127.0.0.1:6379> get rating

"10"

127.0.0.1:6379> getrange names 0 3

"flos"

127.0.0.1:6379> getrange names -3 -1

"sia"

127.0.0.1:6379> setrange names 2 abc

(integer) 6

127.0.0.1:6379>

127.0.0.1:6379>

127.0.0.1:6379>

127.0.0.1:6379> incr rating

(integer) 11

127.0.0.1:6379> decr rating

(integer) 10

127.0.0.1:6379> decrby rating 10

(integer) 0

127.0.0.1:6379> incrby rating 5

(integer) 5

127.0.0.1:6379> set name flosia ex 5

OK

127.0.0.1:6379> get name

"flosia"

127.0.0.1:6379> get name

(nil)

127.0.0.1:6379> sadd naam dd bb

(integer) 2

127.0.0.1:6379> sadd naam cc

(integer) 1

127.0.0.1:6379> srem naam dd

(integer) 1

127.0.0.1:6379> sadd morenaam aa cc

(integer) 2

127.0.0.1:6379> sunion morenaam naam

1) "bb"

2) "aa"

3) "cc"

127.0.0.1:6379> sismember morenaam aa

(integer) 1

127.0.0.1:6379>

127.0.0.1:6379>

127.0.0.1:6379>

127.0.0.1:6379> lpush order ff

(integer) 5

127.0.0.1:6379> rpush order sagat

(integer) 6

127.0.0.1:6379> lrange order 0 4

1) "ff"

2) "rq"

3) "rr"

4) "rt"

5) "rm"

127.0.0.1:6379> lpop order

"ff"

127.0.0.1:6379>

PRACTICAL 6

Demonstrate the Accessing and Storing and performing CRUD operations in

1. HBase

hbase shell

Create TABLE:

hbase(main):003:0>create 'emp','personal data','public data'

Insert Records:

hbase(main):010:0> put 'emp','sam','personal data:active','stockreport.jpg'

Took 0.0097 seconds

hbase(main):011:0> put 'emp','sam','personal data:backup','Java.jar'

Took 0.0096 seconds

hbase(main):012:0>get 'emp','sam'

```
hbase(main):012:0> get 'emp','sam'
COLUMN                                CELL
personal data:active                  timestamp=1673002870257, value=stockreport.jpg
personal data:backup                  timestamp=1673002880427, value=Java.jar
1 row(s)
Took 0.0160 seconds
hbase(main):013:0> scan 'emp'
```

Delete record:

```
hbase(main):013:0> scan 'emp'
ROW          COLUMN+CELL
row1         column=personal data:active, timestamp=1673002796132, value=stockreport.jpg
row2         column=personal data:backup, timestamp=1673002823419, value=Java.jar
sam          column=personal data:active, timestamp=1673002870257, value=stockreport.jpg
sam          column=personal data:backup, timestamp=1673002880427, value=Java.jar
3 row(s)
Took 0.0643 seconds
```

hbase(main):015:0> deleteall 'emp','row1'

Took 0.0081 seconds

```
hbase(main):016:0> scan 'emp'
ROW          COLUMN+CELL
row2         column=personal data:backup, timestamp=1673002823419, value=Java.jar
sam          column=personal data:active, timestamp=1673002870257, value=stockreport.jpg
sam          column=personal data:backup, timestamp=1673002880427, value=Java.jar
2 row(s)
Took 0.0194 seconds
```

Update record:

hbase(main):017:0> put 'emp','sam','personal data:active','Redis.jar'

hbase(main):018:0> scan 'emp'

```
hbase(main):018:0> scan 'emp'
ROW COLUMN+CELL
 row2 column=personal data:backup, timestamp=1673002823419, value=Java.jar
 sam column=personal data:active, timestamp=1673003127226, value=Redis.jar
 sam column=personal data:backup, timestamp=1673002880427, value=Java.jar
2 row(s)
Took 0.0174 seconds
```

2. Apache Cassandra

cqlsh

```
cqlsh>CREATE KEYSPACE SIES WITH REPLICATION ={ 'class':'SimpleStrategy','replication_factor':1 };
```

```
cqlsh> USE SIES;
```

```
cqlsh:sies> CREATE TABLE student(
... student_id int PRIMARY KEY,
... student_name text,
... student_city text,
... student_fees varint,
... student_phone varint
... );
```

```
cqlsh:sies> SELECT * FROM student;
```

```
student_id | student_city | student_fees | student_name | student_phone
-----+-----+-----+-----+-----
(0 rows)
```

```
cqlsh:sies> INSERT INTO student(student_id,student_city,student_fees,student_name,student_phone)
VALUES (88,'NERUL',45055,'Darren',4566633);
```

```
cqlsh:sies> INSERT INTO student(student_id,student_city,student_fees,student_name,stu
arren',98789);
cqlsh:sies> SELECT * FROM student;

student_id | student_city | student_fees | student_name | student_phone
-----+-----+-----+-----+-----
      88 |      NERUL |      345 |      Darren |      98789
(1 rows)
```

```
cqlsh:sies> UPDATE student SET student_city='THANE' WHERE student_id IN(88);
```

```
cqlsh:sies> UPDATE student SET student_city='THANE' WHERE student_id IN(88);
cqlsh:sies> SELECT * FROM student;
```

student_id	student_city	student_fees	student_name	student_phone
88	THANE	345	Darren	98789

(1 rows)

```
cqlsh:sies> DELETE student_fees FROM student WHERE student_id=88;
```

```
cqlsh:sies> DELETE student_fees FROM student WHERE student_id=88;
cqlsh:sies> SELECT * FROM student;
```

student_id	student_city	student_fees	student_name	student_phone
88	THANE	null	Darren	98789

(1 rows)

```
cqlsh:sies>
```

PRACTICAL 7

Demonstrating MapReduce in MongoDB to count the number of female (F) and male (M) respondents in the database

```
> db.customer.find()
```

```
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd2"), "rollno" : 1, "gender" : "F", "class" : "fy", "marks" : 60 }
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd3"), "rollno" : 2, "gender" : "F", "class" : "fy", "marks" : 70 }
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd4"), "rollno" : 1, "gender" : "M", "class" : "sy", "marks" : 40 }
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd5"), "rollno" : 2, "gender" : "F", "class" : "sy", "marks" : 50 }
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd6"), "rollno" : 3, "gender" : "F", "class" : "ty", "marks" : 90 }
```

```
var m1=function(){emit(this.gender,this.gender)}
```

```
> var r1=function(key,values){return Array.sum(values)}
```

```
> db.customer.mapReduce(m1,r1,{ 'out': 'Result1' })
```

```
{ "result" : "Result1", "ok" : 1 }
```

```
> db.Result1.find()
```

```
{ "_id" : "M", "value" : "M" }
```

```
{ "_id" : "F", "value" : "FFFF" }
```

```
var m1=function(){emit(this.class,this.gender)}
```

```
> var r1=function(key,values){return Array.sum(values)}
```

```
> db.customer.mapReduce(m1,r1,{ 'out': 'Result1' })
```

```
{ "result" : "Result1", "ok" : 1 }
```

```
> db.Result1.find()
```

```
{ "_id" : "ty", "value" : "F" }
```

```
{ "_id" : "fy", "value" : "FF" }
```

```
{ "_id" : "sy", "value" : "FM" }
```

PRACTICAL 8

Demonstrate the indexing and ordering operations in

1. MongoDB

```
>db.student.insertMany([{rollno:4,Name:"Nikita",marks:10,city:"mumbai"},{rollno:5,Name:"Neeraj",marks:15,city:"mumbai"},{rollno:6,Name:"Aman",marks:0,city:"Panvel"},{rollno:7,Name:"Gopal",marks:50,city:"Thane"},{rollno:8,Name:"Rohit",marks:55,city:"Thane"},{rollno:9,Name:"Shrihari",marks:33,city:"Nerul"},{rollno:10,Name:"Sarvesh",marks:0,city:"Thane"}])
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("6371d044d6261288bd217ec4"),
    ObjectId("6371d044d6261288bd217ec5"),
    ObjectId("6371d044d6261288bd217ec6"),
    ObjectId("6371d044d6261288bd217ec7"),
    ObjectId("6371d044d6261288bd217ec8"),
    ObjectId("6371d044d6261288bd217ec9"),
    ObjectId("6371d044d6261288bd217eca")
  ]
}

> db.student.createIndex({Name:1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}

> db.student.find({Name:"Aman"})
{ "_id" : ObjectId("6371d044d6261288bd217ec6"), "rollno" : 6, "Name" : "Aman", "marks" : 0, "city" : "Panvel" }
```

MongoDB Ordering

```
> db.student.find({marks:{"$gt":55}}).sort({marks:-1})
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd6"), "rollno" : 3, "gender" : "F", "class" : "ty", "marks" : 90 }
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd3"), "rollno" : 2, "gender" : "F", "class" : "fy", "marks" : 70 }
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd2"), "rollno" : 1, "gender" : "F", "class" : "fy", "marks" : 60 }

> db.customer.find({marks:{"$gt":55}}).sort({marks:1})
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd2"), "rollno" : 1, "gender" : "F", "class" : "fy", "marks" : 60 }
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd3"), "rollno" : 2, "gender" : "F", "class" : "fy", "marks" : 70 }
{ "_id" : ObjectId("62ef64b3b5d06df968c18dd6"), "rollno" : 3, "gender" : "F", "class" : "ty", "marks" : 90 }
```

2. CouchDB

Code:

Entry1:

```
{
  "_id": "00a271787f89c0ef2e10e88a0c0001f4",
  "type": "movie",
  "title": "My Neighbour Totoro",
  "year": 1988,
  "director": "miyazaki",
  "rating": 8.2
}
```

Table

Metadata

{ } JSON

Create Document

id "5392a62418b69762c8e98d711300d9cc"

```
{
  "id": "5392a62418b69762c8e98d711300d9cc",
  "key": "5392a62418b69762c8e98d711300d9cc",
  "value": {
    "rev": "1-c7f3ec5e61c7a461d83da2d42f64a9e4"
  },
  "doc": {
    "_id": "5392a62418b69762c8e98d711300d9cc",
    "_rev": "1-c7f3ec5e61c7a461d83da2d42f64a9e4",
    "type": "movie",
    "title": "My Neighbour Totoro",
    "year": 1988,
    "director": "miyazaki",
    "rating": 8.2
  }
}
```

Entry2:

```
{
  "_id": "00a271787f89c0ef2e10e88a0c0003f0",
  "type": "movie",
  "title": "Kikis Delivery Service",
  "year": 1989,
  "director": "miyazaki",
  "rating": 7.8
}
```

Table

Metadata

{ } JSON

Create Document

id "5392a62418b69762c8e98d711300fc64"

```
{
  "id": "5392a62418b69762c8e98d711300fc64",
  "key": "5392a62418b69762c8e98d711300fc64",
  "value": {
    "rev": "1-5facb9c84b721a5fe8d667c1140a54d3"
  },
  "doc": {
    "_id": "5392a62418b69762c8e98d711300fc64",
    "_rev": "1-5facb9c84b721a5fe8d667c1140a54d3",
    "type": "movie",
    "title": "Kikis Delivery Service",
    "year": 1989,
    "director": "miyazaki",
    "rating": 7.8
  }
}
```

Entry3:

```
{
  "_id": "00a271787f89c0ef2e10e88a0c00048b",
  "type": "movie",
  "title": "Princess Mononoke",
  "year": 1997,
  "director": "miyazaki",
}
```

```
}
  "rating": 8.4
}
```

		_id	director	rating	title	type
<input type="checkbox"/>		5392a62418b69762c8e98...	miyazaki	8.2	My Neighbour Totoro	movie
<input type="checkbox"/>		5392a62418b69762c8e98...	miyazaki	7.8	Kikis Delivery Service	movie
<input type="checkbox"/>		5392a62418b69762c8e98...	miyazaki	8.4	Princess Mononoke	movie
<input type="checkbox"/>		_design/4a53d1a437fa6d9...				

Create index

```
{
  "index": {
    "fields": [
      "year"
    ]
  },
  "name": "year-json-index",
  "type": "json"
}
```

This defines an index on the field year and allows us to send queries for documents from a specific year. Next, click on “edit query” and change the Mango Query to look like this:

```
{
  "selector": {
    "year": {
      "$eq": 1988
    }
  }
}
```

movies > Mango Query

Query history

Mango Query

```
1 {
2   "selector": {
3     "year": {
4       "$eq": 1988
5     }
6   }
7 }
```

Run Query

Explain

Executed in 1 ms

manage indexes

Table

{ } JSON

Create Document

		_id	director	rating	title	type
<input checked="" type="checkbox"/>		5392a62418b69762c8...	miyazaki	8.2	My Neighbour Totoro	movie

3. Apache Cassandra

Creating an index on a column

```
cqlsh> CREATE KEYSPACE IF NOT EXISTS myschema WITH REPLICATION = { 'class' :
'SimpleStrategy', 'replication_factor' :
'1' };
```



```
cqlsh> CREATE TABLE myschema.users (  
    ...  userID uuid,  
    ...  fname text,  
    ...  lname text,  
    ...  email text,  
    ...  address text,  
    ...  zip int,  
    ...  state text,  
    ...  PRIMARY KEY (userID)  
    ... );
```

```
cqlsh>
```

```
cqlsh> CREATE INDEX user_state  
    ...  ON myschema.users (state);
```

Creating an index on a clustering column

```
cqlsh> CREATE KEYSPACE IF NOT EXISTS mykeyspace WITH REPLICATION = { 'class' :  
'SimpleStrategy', 'replication_factor'  
: '1' };
```

```
cqlsh> CREATE TABLE mykeyspace.users (  
    ...  userID uuid,  
    ...  fname text,  
    ...  lname text,  
    ...  email text,  
    ...  address text,  
    ...  zip int,  
    ...  state text,  
    ...  PRIMARY KEY ((userID, fname), state)  
    ... );
```

```
cqlsh> CREATE INDEX ON mykeyspace.users (state);
```

Creating an index on a set or list collection

```
cqlsh> ALTER TABLE mykeyspace.users ADD phones set<text>;
```

```
cqlsh> CREATE INDEX ON mykeyspace.users (phones);
```