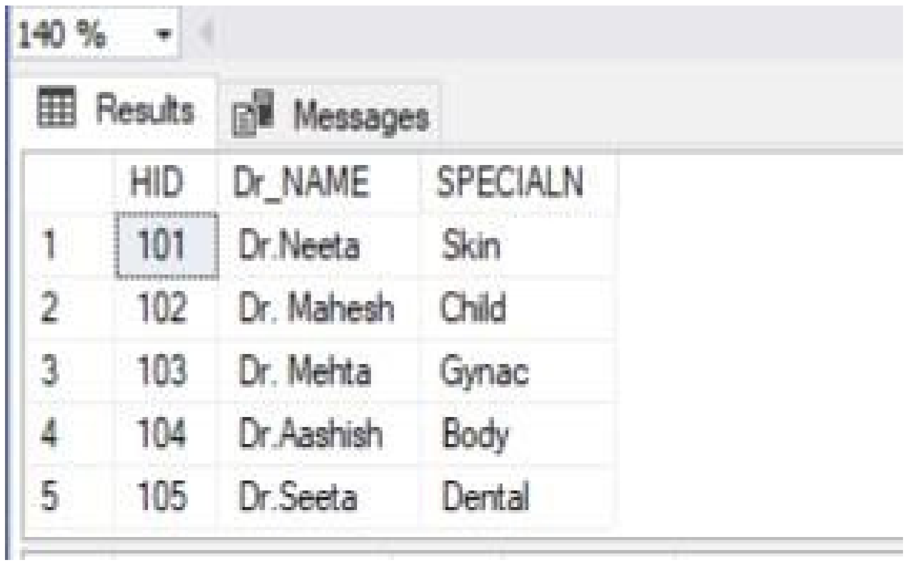
Atharv Anandrao Bhondave
A-17

Assignment No.1

Problem Statement : Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym, different constraints etc.
Write at least 10 SQL queries on the suitable database application using SQL DML statements.

Code:

```
USE Anand;

CREATE TABLE HOSPI (HID INT,Dr_NAME VARCHAR(20),SPECIALN
VARCHAR(20));

INSERT INTO HOSPI VALUES(101,'Dr.Neeta','Skin');

INSERT INTO HOSPI VALUES(102,'Dr. Mahesh','Child');

INSERT INTO HOSPI VALUES(103,'Dr. Mehta','Gynac');

INSERT INTO HOSPI VALUES(104,'Dr.Aashish','Body');

INSERT INTO HOSPI VALUES(105,'Dr.Seeta','Dental');

SELECT* FROM HOSPI;
```



	HID	Dr_NAME	SPECIALN
1	101	Dr.Neeta	Skin
2	102	Dr. Mahesh	Child
3	103	Dr. Mehta	Gynac
4	104	Dr.Aashish	Body
5	105	Dr.Seeta	Dental

CREATE TABLE HDATA(HNAME VARCHAR(20),HSN INT,CITY VARCHAR(10));

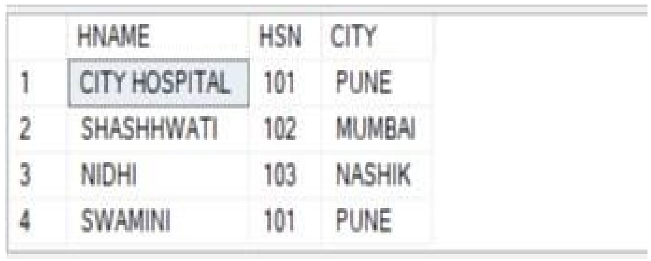
INSERT INTO HDATA VALUES('CITY HOSPITAL',101,'PUNE');

INSERT INTO HDATA VALUES('SHASHHWATI',102,'MUMBAI');

INSERT INTO HDATA VALUES('NIDHI',103,'NASHIK');

INSERT INTO HDATA VALUES('SWAMINI',101,'PUNE');

SELECT* FROM HDATA;



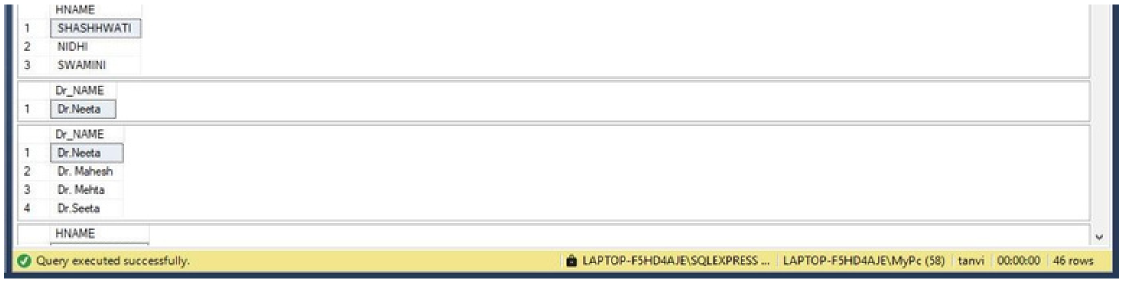
	HNAME	HSN	CITY
1	CITY HOSPITAL	101	PUNE
2	SHASHHWATI	102	MUMBAI
3	NIDHI	103	NASHIK
4	SWAMINI	101	PUNE

SELECT HNAME FROM HDATA WHERE HNAME LIKE '%I';

SELECT Dr_NAME FROM HOSPI WHERE SPECIALN='Skin';

SELECT Dr_NAME FROM HOSPI WHERE Dr_NAME NOT LIKE '%I%';

SELECT HNAME FROM HDATA WHERE HNAME NOT LIKE '%I';

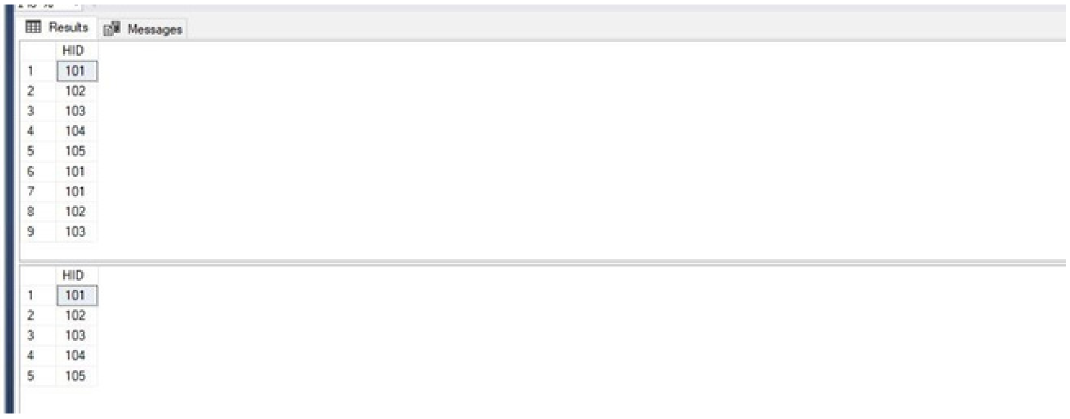


```
    HNAME
1   SHASHHWATI
2   NIDHI
3   SWAMINI

    Dr_NAME
1   Dr.Neeta

    Dr_NAME
1   Dr.Neeta
2   Dr. Mahesh
3   Dr. Mehta
4   Dr.Seeta

    HNAME
```

```
Query executed successfully.   LAPTOP-F5HD4AJE\SQLEXPRESS ...   LAPTOP-F5HD4AJE\MyPc (58)   tanvi   00:00:00   46 rows
```

CREATE INDEX IDX ON HDATA(HSN);
SELECT HID FROM HOSPI UNION ALL SELECT HSN FROM HDATA;

SELECT HID FROM HOSPI UNION SELECT HSN FROM HDATA;



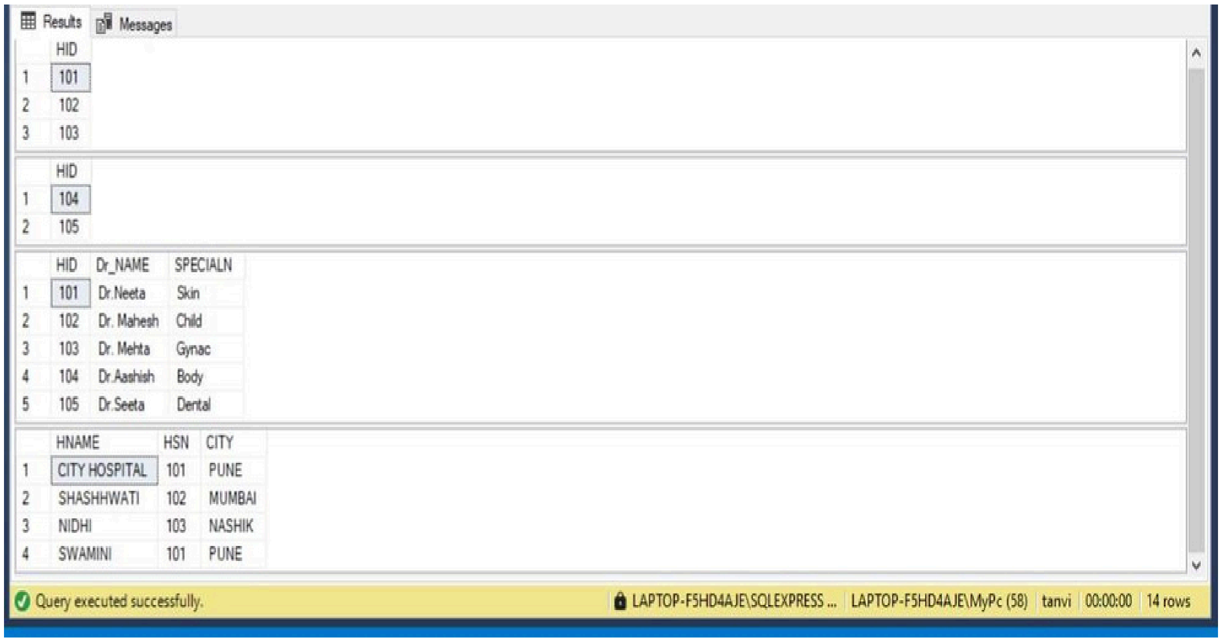
```
    HID
1   101
2   102
3   103
4   104
5   105
6   101
7   101
8   102
9   103

    HID
1   101
2   102
3   103
4   104
5   105
```

SELECT HID FROM HOSPI INTERSECT SELECT HSN FROM HDATA;

SELECT HID FROM HOSPI EXCEPT SELECT HSN FROM HDATA;

SELECT* FROM HOSPI;

SELECT* FROM HDATA;



Results | Messages

	HID
1	101
2	102
3	103

	HID
1	104
2	105

	HID	Dr_NAME	SPECIALN
1	101	Dr.Neeta	Skin
2	102	Dr. Mahesh	Child
3	103	Dr. Mehta	Gynac
4	104	Dr.Aashish	Body
5	105	Dr.Seeta	Dental

	HNAME	HSN	CITY
1	CITY HOSPITAL	101	PUNE
2	SHASHHWATI	102	MUMBAI
3	NIDHI	103	NASHIK
4	SWAMINI	101	PUNE

```
Query executed successfully.    LAPTOP-F5HD4AJE\SQLEXPRESS ...  LAPTOP-F5HD4AJE\MyPc (58)  tanvi  00:00:00  14 rows
```

ssignment No.2 //Anand Bora(A-20)

Problem Statement: Write at least 10 SQL queries for suitable database application using SQL DML statements.

Code:

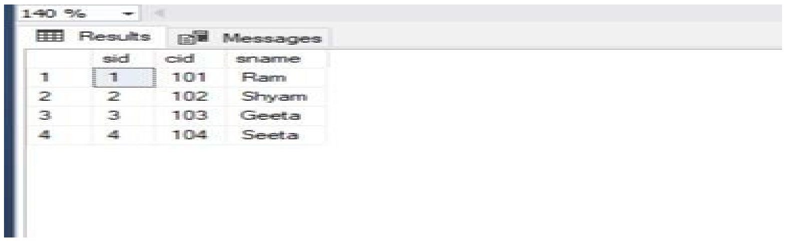
use

Anand;

LAPTOP-F5HD4AJE\SQLEXPRESS (SQL SDatabasesSecurity Server Objects

Replication Management XEvent Profiler aggregate_func.sq... HD4AJE\MyPc (55))*

select from student01;

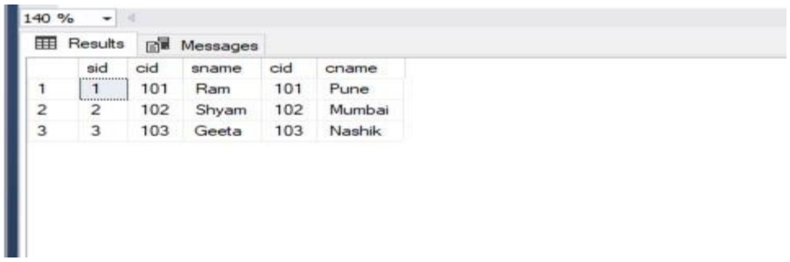


140 %

Results | Messages

	sid	cid	sname
1	1	101	Ram
2	2	102	Shyam
3	3	103	Geeta
4	4	104	Seeta

select from city;

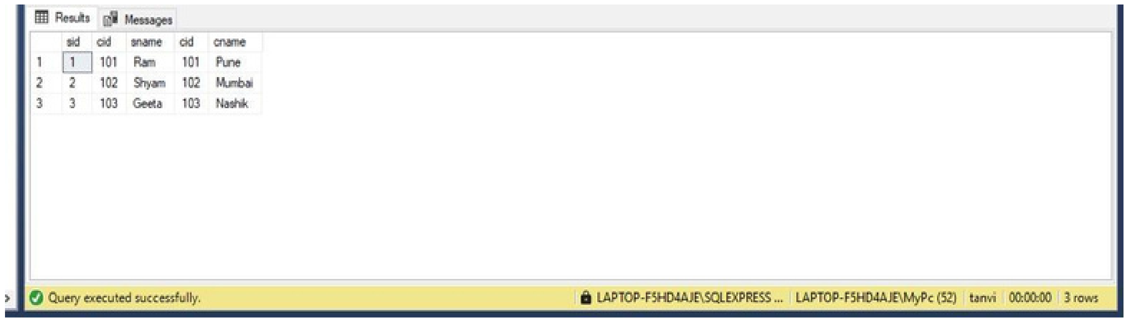


140 %

Results | Messages

	sid	cid	sname	cid	cname
1	1	101	Ram	101	Pune
2	2	102	Shyam	102	Mumbai
3	3	103	Geeta	103	Nashik

select from student01 s inner join city con s.cid=c.cid;

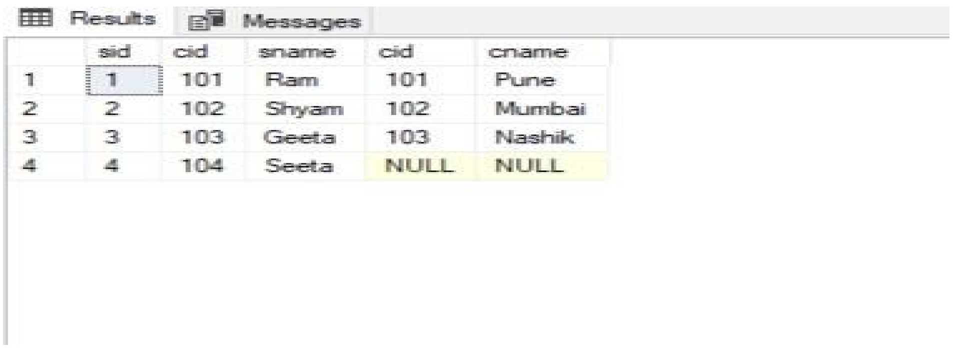


Results | Messages

	sid	cid	sname	cid	cname
1	1	101	Ram	101	Pune
2	2	102	Shyam	102	Mumbai
3	3	103	Geeta	103	Nashik

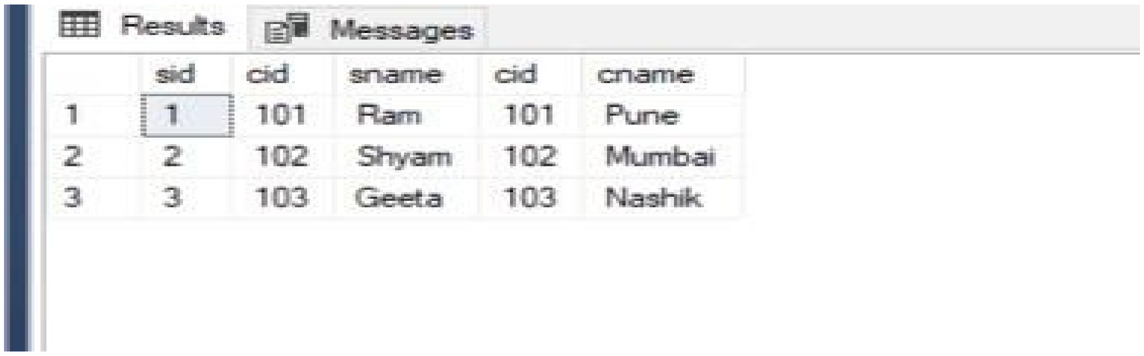
```
Query executed successfully.    LAPTOP-F5HD4AJE\SQLEXPRESS ... | LAPTOP-F5HD4AJE\MyPc (52) | tanvi | 00:00:00 | 3 rows
```

select from student01 s left join city con s.cid=c.cid;



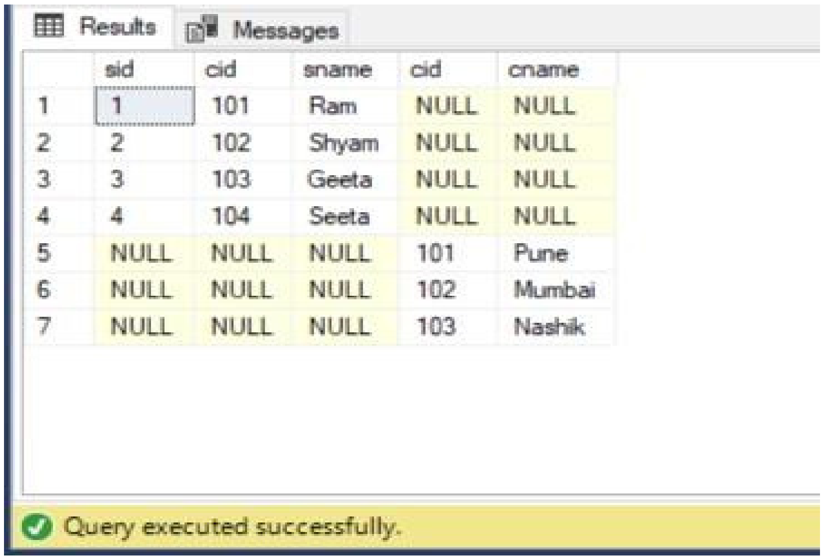
	sid	cid	sname	cid	cname
1	1	101	Ram	101	Pune
2	2	102	Shyam	102	Mumbai
3	3	103	Geeta	103	Nashik
4	4	104	Seeta	NULL	NULL

select from student01 right join city c on c.cid=s.cid;



	sid	cid	sname	cid	cname
1	1	101	Ram	101	Pune
2	2	102	Shyam	102	Mumbai
3	3	103	Geeta	103	Nashik

select from student01 s full join city con s.sid=c.cid;



	sid	cid	sname	cid	cname
1	1	101	Ram	NULL	NULL
2	2	102	Shyam	NULL	NULL
3	3	103	Geeta	NULL	NULL
4	4	104	Seeta	NULL	NULL
5	NULL	NULL	NULL	101	Pune
6	NULL	NULL	NULL	102	Mumbai
7	NULL	NULL	NULL	103	Nashik

Query executed successfully.

Assignment no.3 //Anand Bora(A-20)

Problem Statement: Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scoredby students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and 900 category is first class, if marks 899 and 825 category is Higher Second Class.
Write a PL/SQL block to use procedure created with above requirement.
Stud_Marks(name, total_marks) Result(Roll,Name, Class).

Code:

```
use abc;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed mysql> create table stud_marks( roll_no int primary key, name
varchar(10),total_marks int);
ERROR 1050 (42S01): Table 'stud_marks' already exists
mysql> create table stud( roll_no int primary key, name varchar(10),total_marks int);
Query OK, 0 rows affected (0.34 sec)

mysql> insert into stud values (101,'abc',933);
Query OK, 1 row affected (0.06 sec)

mysql> insert into stud values (102,'ghj',356);
Query OK, 1 row affected (0.31 sec)

mysql> insert into stud values (103,'dgt',987);
Query OK, 1 row affected (0.30 sec)

mysql> insert into stud values (104,'drt',450);
Query OK, 1 row affected (0.35 sec)

mysql> insert into stud values (105,'tyu',675);

Query OK, 1 row affected (0.06 sec) mysql>

select* from stud; +---------+------+-------------+
| roll_no | name | total_marks |
+---------+------+-------------+
|    101  | abc  |        933 |
|    102  |  ghj  |        356 |
|    103  | dgt  |       987 |
|    104  |  drt  |      450 |
|    105 | tyu |        675 |
+---------+------+-------------+
5 rows in set (0.00 sec)

mysql> create table result(roll_no int, name char(30),class varchar(20 -
    > ));
ERROR 1050 (42S01): Table 'result' already exists
mysql> create table result1(roll_no int, name char(30),class varchar(20 ));
Query OK, 0 rows affected (0.36 sec)

mysql> delimiter // mysql> create procedure grade(in marks int,
out class char(10))
    -> begin
    -> if marks<=1500 and marks>=990 then set class='DIST';
    -> end if;
    -> if marks<=989 and marks>=900 then set class='FC';
    -> end if;
    -> if marks<=899 and marks>=825 then set class='SC';
    -> end if;
    -> if marks<=749 and marks>=650 then set class='fail';
    -> end if;
    -> end;
    -> //
Query OK, 0 rows affected (0.12 sec)

mysql> create function find(roll_no int) returns int deterministic
    -> begin
    -> declare fmarks int;
    -> declare grade char(10);
    -> declare stud_name char(10);
    -> select stud_marks.total_marks, stud_marks.name into fmarks,stud_name from
stud_marks where stud_marks.roll_no=roll_in;
```

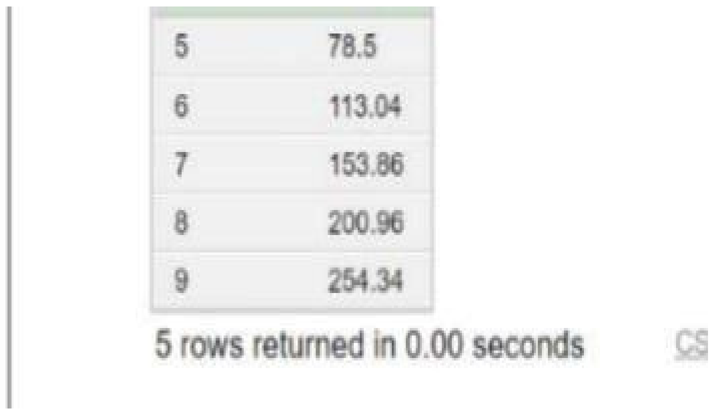
```
    -> call grade(fmarks,@grade);
    -> insert into result1 values(roll_in,stud_name,@grade);
    -> return roll_in;
    -> end;
    -> //
Query OK, 0 rows affected (0.08 sec)
```

Assignment No.4

Problem Statement: Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.

Code:

```
use Anand;
CREATE TABLE AREAS
RADIUS NUMBER(5), AREA NUMBER(14,2)):
DECLARE
pi constant number(4.2):=3.14: radius mumber(5); area number(14.2);
BEGIN
radius:5 while radius <=9 loop area pi*power(radius,2); insert into areas
values(radius, area); radius: radius+1: end loop:
end;
SELECT* FROM AREAS:
```



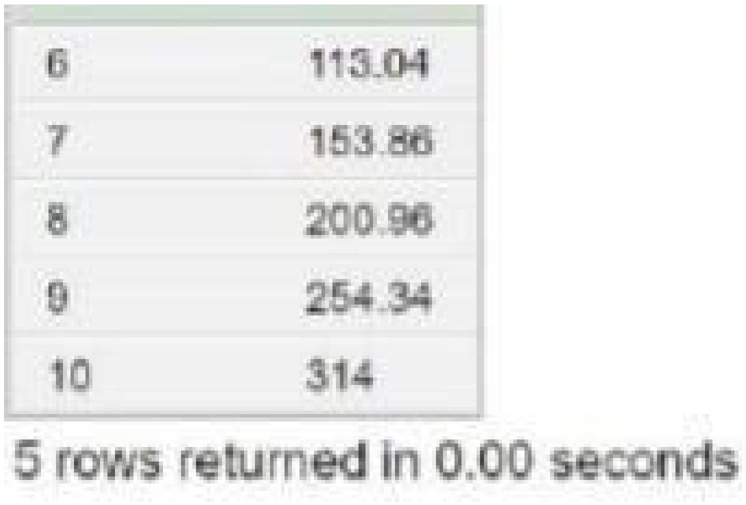
5	78.5
6	113.04
7	153.86
8	200.96
9	254.34

5 rows returned in 0.00 seconds

```
CREATE TABLE AREAS
( RADIUS NUMBER(5), AREA NUMBER(14.2));
DECLARE
pi constant number(4.2): 3.14; radius number(5): arca number(14,2);
BEGIN
```

radius:6; while radius <=10) loop arca: pi*power(radius, 2); insert into areas values(radius area); radius: radius+1; end loop; end:

SELECT * FROM AREAS:



6	113.04
7	153.86
8	200.96
9	254.34
10	314

5 rows returned in 0.00 seconds

Assignment no.6 //Anand Bora(A-20)

Problem Statement: Write a PL/SQL block of code using parameterized Cursor that will merge the data availablein the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.

Code:

```
use abc;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> create table old_r(rno int primary key,name varchar(20),Address varchar(50));
Query OK, 0 rows affected (0.32 sec)

mysql> insert into old_r values(1,'Amit'.'Pune');
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near '.'Pune')' at
line 1
mysql> insert into old_r values(1,'Amit','pune');
Query OK, 1 row affected (0.05 sec)

mysql> insert into old_r values(2,'preti','nashik');
Query OK, 1 row affected (0.53 sec)

mysql> insert into old_r values(3,'amol','nagpur');
Query OK, 1 row affected (0.41 sec)

mysql> insert into old_r values(4,'pritm','ahmbad');
Query OK, 1 row affected (0.07 sec)

mysql> insert into old_r values(5,'Rubina','mumbai');

Query OK, 1 row affected (0.71 sec) mysql> selcrt*

from old_r;
```

```
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 'selcrt* from
old_r' at line 1
mysql> select* from old_r;
+-----+--------+---------+
| rno | name | Address |
+-----+--------+---------+
| 1 | Amit | pune |
|    2 | preti | nashik |
| 3 | amol | nagpur |
|    4 | pritm | ahmbad |
| 5 | Rubina | mumbai |
+-----+--------+---------+
5 rows in set (0.00 sec)

mysql> create table new_r (rno int, name varchar(20),add varchar(10));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 'add
varchar(10))' at line 1
mysql> create table new_r (rno int, name varchar(20),add varchar(20));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 'add
varchar(20))' at line 1
mysql> create table new_r (rno int, name varchar(20),address varchar(20));
Query OK, 0 rows affected (0.52 sec)

mysql> insert into new_r values(1,'amit','pune');
Query OK, 1 row affected (0.39 sec)

mysql> insert into new_r values(2,'preti','nashik');
Query OK, 1 row affected (0.06 sec)

mysql> insert into new_r values(3,'amol','nagpur');
Query OK, 1 row affected (0.37 sec)

mysql> select* from new_r;
+------+-------+---------+
| rno | name | address |
+------+-------+---------+
|     1 | amit | pune     |
|     2 | preti | nashik |
|3 | amol | nagpur | +-----
-+-------+---------+       3
rows in set (0.00 sec)

mysql> delimiter //
mysql> create procedure N(in rno1 int)
    -> begin
    -> declare rno2 int;
    -> declare exit_con boolean;
    -> declare c1 cursor for select rno from old_r where rno>rno1;
    -> declare continue handler for not found set exit_con=TRUE;
    -> open c1;
    -> l1:loop
    -> fetch c1 into rno2;
    -> if not exists(select* from new_r where rno=rno2) then
    -> insert into new_r select* from old_r where rno=rno2;
    -> end if;
    -> if exit_con then
    -> close c1;
    -> leave l1;
    -> end if;
    -> end loop l1;
    -> end;
    -> //
Query OK, 0 rows affected (0.41 sec)

mysql> call N(3); -
    > //
Query OK, 0 rows affected (0.21 sec)

mysql> select* from new_r; -
    > //
+------+--------+---------+
| rno | name     | address |
+------+--------+---------+
|      1 | amit | pune |
|     2 | preti | nashik |
|   3 | amol | nagpur |
|    4 | pritm | ahmbad |
|    5 | Rubina | mumbai |
+------+--------+---------+
5 rows in set
```

Assignment no.7 //Anand Bora(A-20)

Problem Statement: Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added inLibrary_Audit table.

Code:

```
mysql> use abc;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed mysql> create table stud1(sid int primary key,name
varchar(20),class varchar(20),age int,issued varchar(10),bname varchar(20)); Query
OK, 0 rows affected (0.40 sec)

mysql> create table returned(sid int,bname varchar(10),author varchar(20));
Query OK, 0 rows affected (1.48 sec)

mysql> create table issue(sid int,bname varchar(20));
Query OK, 0 rows affected (1.12 sec)

mysql> insert into stud1 values(1,'amit','1-A',20,'I','korth');
Query OK, 1 row affected (0.08 sec)

mysql> insert into stud1 values(2,'kranti','2-A',21,'I','DBMS');
Query OK, 1 row affected (0.07 sec)

mysql> select* from stud1;
+-----+--------+-------+------+--------+-------+
| sid | name | class | age | issued | bname |
+-----+--------+-------+------+--------+-------+
| 1 | amit | 1-A | 20 | I          | korth |
| 2 | kranti | 2-A | 21 | I          | DBMS |
+-----+--------+-------+------+--------+-------+
2 rows in set (0.00 sec)
mysql> delimiter // mysql>
```

```
create trigger tr_stud ->
before insert -> on stud1
    -> for each row
    -> begin
    -> insert into issue values(new.sid,new.bname);
    -> end;
    -> //
Query OK, 0 rows affected (0.11 sec)

mysql> insert into stud1 values(3,'geeta','3-A',22,'I','JS'); -
    > //
Query OK, 1 row affected (0.50 sec)

mysql> select* from issue; -
    > //
+------+-------+
| sid | bname |
+------+-------+
|    3 | JS     |
+------+-------+
1 row in set (0.00 sec)
mysql> delimiter // mysql>
create trigger tr_stud
    -> before insert -
    > on stud1
    -> for each row
    -> begin
    -> insert into issue values(new.sid,new.bname);
    -> end;
    -> //
Query OK, 0 rows affected (0.11 sec)

mysql> insert into stud1 values(3,'geeta','3-A',22,'I','JS'); -
    > //
Query OK, 1 row affected (0.50 sec)

mysql> select* from issue; -
    > //
```

```
+------+--------+
| sid | bname |
+------+--------+
|    3 | JS     |
+------+--------+
1 row in set (0.00 )
```

Assignment No.7 //Anand Bora(A-20)

Problem Statement: Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations,SAVE method, logical operators etc.).

Code:

```
use assb02; switched
to db assb02
assb02> db.assb02.insert({roll:1,name:'navin',subject:'c++',marks:70});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or
bulkWrite.
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb45c9fc4505e08e35722") }
}
assb02> db.assb02.insert({roll:2,name:'anusha',subject:'dsa',marks:75});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb4839fc4505e08e35723") }
}
assb02> db.assb02.insert({roll:3,name:'ravi',subject:'toc',marks:69});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb4cb9fc4505e08e35724") }
}
assb02> db.assb02.find();
[
{
_id: ObjectId("634fb45c9fc4505e08e35722"),
roll: 1, name: 'navin', subject: 'c++', marks:
70},
{
_id: ObjectId("634fb4839fc4505e08e35723"),
roll: 2,
name: 'anusha',
subject: 'dsa', marks:
75
},
{
_id: ObjectId("634fb4cb9fc4505e08e35724"),
roll: 3, name: 'ravi', subject: 'toc', marks: 69
}
]
assb02> db.assb02.insert({roll:4,name:'veena',subject:'c++',marks:70});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb5649fc4505e08e35725") }
}
assb02> db.assb02.insert({roll:5,name:'pravin',subject:'CN',marks:80});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb5969fc4505e08e35726") }
}
assb02> db.assb02.insert({roll:6,name:'geeta',subject:'CN',marks:90});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb5b39fc4505e08e35727") }
}
assb02> db.assb02.find();
[
{
_id: ObjectId("634fb45c9fc4505e08e35722"),
roll: 1, name: 'navin', subject: 'c++', marks:
70
},
{
_id: ObjectId("634fb4839fc4505e08e35723"),
roll: 2,
name: 'anusha',
subject: 'dsa',
marks: 75
},
{
_id: ObjectId("634fb4cb9fc4505e08e35724"),
roll: 3, name: 'ravi', subject: 'toc', marks: 69
},
{
_id: ObjectId("634fb5649fc4505e08e35725"),roll: 4,
name: 'veena', subject: 'c++', marks: 70
},
{
_id: ObjectId("634fb5969fc4505e08e35726"),
roll: 5, name: 'pravin', subject: 'CN', marks:
80
},
{
_id: ObjectId("634fb5b39fc4505e08e35727"),
roll: 6, name: 'geeta', subject: 'CN', marks: 90
}
]
ssb02> db.assb02.createIndex({name:1});
name_1
assb02> db.assb02.getIndexes();
[
{ v: 2, key: { _id: 1 }, name: '_id_' },
{ v: 2, key: { name: 1 }, name: 'name_1' }
]
assb02> db.assb02.find().sort({name:1});
[
{
_id: ObjectId("634fb4839fc4505e08e35723"),
roll: 2,
name: 'anusha',
subject: 'dsa',
marks: 75
},
{
_id: ObjectId("634fb5b39fc4505e08e35727"),
roll: 6, name: 'geeta', subject: 'CN', marks: 90
},
{
_id: ObjectId("634fb45c9fc4505e08e35722"),
roll: 1, name: 'navin', subject: 'c++', marks:
70
},
{
_id: ObjectId("634fb5969fc4505e08e35726"),
roll: 5, name: 'pravin', subject: 'CN',marks: 80
},
{
_id: ObjectId("634fb4cb9fc4505e08e35724"),
roll: 3, name: 'ravi', subject: 'toc', marks: 69
},
{
_id: ObjectId("634fb5649fc4505e08e35725"),
roll: 4,
name: 'veena',
subject: 'c++',
marks: 70
}
]
assb02> db.assb02.find().sort({name:-1});
[
{
_id: ObjectId("634fb5649fc4505e08e35725"),
roll: 4,
name: 'veena',
subject: 'c++',
marks: 70
},
{
_id: ObjectId("634fb4cb9fc4505e08e35724"),
roll: 3, name: 'ravi', subject: 'toc', marks: 69
},
{
_id: ObjectId("634fb5969fc4505e08e35726"),
roll: 5, name: 'pravin', subject: 'CN', marks:
80
```

```
},
{
_id: ObjectId("634fb45c9fc4505e08e35722"),
roll: 1, name: 'navin', subject: 'c++', marks:
70
},
{
_id: ObjectId("634fb5b39fc4505e08e35727"),
roll: 6, name: 'geeta', subject: 'CN', marks: 90
},
{
_id: ObjectId("634fb4839fc4505e08e35723"),roll: 2,
name: 'anusha',
subject: 'dsa',
marks: 75
}
]
ssb02> db.assb02.find({name:'pravin'});
[
{
_id: ObjectId("634fb5969fc4505e08e35726"),
roll: 5, name: 'pravin', subject: 'CN', marks:
80
}
]
assb02> db.assb02.ensureIndex({roll:1},{unique:true});
[ 'roll_1' ]
assb02> db.assb02.insert({roll:6,name:'geeta',subject:'CN',marks:90});
Uncaught:
MongoBulkWriteError: E11000 duplicate key error collection: assb02.assb02 index:
roll_1 dup key:
{
roll: 6 }
Result: BulkWriteResult { result: { ok: 1, writeErrors: [ WriteError { err: { index: 0,
code: 11000, errmsg: 'E11000 duplicate key error collection: assb02.assb02 index:
roll_1 dup key: { roll: 6 }', errInfo: undefined, op: {
roll: 6, name:
'geeta',
subject: 'CN',
marks: 90,
_id: ObjectId("634fb7859fc4505e08e35728")
}}}],
```

Assignment No.9 //Anand Bora(A-20)

Problem Statement:DesignandDevelopMongoDBQueriesusingCRUDoperations.(Use CRUD operations, SAVE method, logical operators etc.).

Code:

```
db.createCollection("library");
{ ok: 1 }
assb01> db.library.insert({"bid":1,"name":"c++"});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or
bulkWrite.
{
acknowledged: true, insertedIds: { '0':
ObjectId("634f8ac74ec9cc905a3ea1cf") }
}
assb01> db.library.insertOne({"bid":1,"name":"c++"});
{
acknowledged: true, insertedId:
ObjectId("634f8b294ec9cc905a3ea1d0")
}
assb01> db.library.insertOne({"bid":2,"name":"c"});
{
acknowledged: true, insertedId:
ObjectId("634f8b4c4ec9cc905a3ea1d1")
}assb01> db.library.find();
[
{ _id: ObjectId("634f8ac74ec9cc905a3ea1cf"), bid: 1, name: 'c++' },
{ _id: ObjectId("634f8b294ec9cc905a3ea1d0"), bid: 1, name: 'c++' },
{ _id: ObjectId("634f8b4c4ec9cc905a3ea1d1"), bid: 2, name: 'c' }
]
assb01> db.library.remove({"bid":1});
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne,
deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 2 }
assb01> db.library.find();
```

```
[ { _id: ObjectId("634f8b4c4ec9cc905a3ea1d1"), bid: 2, name: 'c' } ]
assb01> db.library.update({"bid":2},{$set:{"name":"Python"} });
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
acknowledged: true, insertedId: null, matchedCount: 1, modifiedCount: 1, upsertedCount: 0
}
assb01> db.library.find();
[
{ _id: ObjectId("634f8b4c4ec9cc905a3ea1d1"), bid: 2, name: 'Python' }
]
```

Assignment no.10 //Anand Bora(A-20)

Problem Statement: Design and Develop MongoDB Queries using aggregation and indexing with suitable exampleusing MongoDB.

Code:

```
db.assb02.insert({roll:1,name:'navin',subject:'c++',marks:70});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or
bulkWrite.
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb45c9fc4505e08e35722") }
}
assb02> db.assb02.insert({roll:2,name:'anusha',subject:'dsa',marks:75});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb4839fc4505e08e35723") }
}
assb02> db.assb02.insert({roll:3,name:'ravi',subject:'toc',marks:69});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb4cb9fc4505e08e35724") }
}
assb02> db.assb02.find();
[
{
_id: ObjectId("634fb45c9fc4505e08e35722"),
roll: 1, name: 'navin', subject: 'c++', marks:
70},
{
_id: ObjectId("634fb4839fc4505e08e35723"),
roll: 2,
name: 'anusha',
subject: 'dsa',
marks: 75
```

```
},
{
_id: ObjectId("634fb4cb9fc4505e08e35724"),
roll: 3, name: 'ravi', subject: 'toc', marks: 69
}
]
assb02> db.assb02.insert({roll:4,name:'veena',subject:'c++',marks:70});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb5649fc4505e08e35725") }
}
assb02> db.assb02.insert({roll:5,name:'pravin',subject:'CN',marks:80});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb5969fc4505e08e35726") }
}
assb02> db.assb02.insert({roll:6,name:'geeta',subject:'CN',marks:90});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fb5b39fc4505e08e35727") }
}
assb02> db.assb02.find();
[
{
_id: ObjectId("634fb45c9fc4505e08e35722"),
roll: 1, name: 'navin', subject: 'c++', marks:
70
},
{
_id: ObjectId("634fb4839fc4505e08e35723"),
roll: 2,
name: 'anusha',
subject: 'dsa',
marks: 75
},
{
_id: ObjectId("634fb4cb9fc4505e08e35724"),
roll: 3, name: 'ravi', subject: 'toc', marks: 69
```

```
},
{
_id: ObjectId("634fb5649fc4505e08e35725"),roll: 4,
name: 'veena', subject: 'c++', marks: 70
},
{
_id: ObjectId("634fb5969fc4505e08e35726"),
roll: 5, name: 'pravin', subject: 'CN', marks:
80
},
{
_id: ObjectId("634fb5b39fc4505e08e35727"),
roll: 6, name: 'geeta', subject: 'CN', marks: 90
}
]
ssb02> db.assb02.createIndex({name:1});
name_1
assb02> db.assb02.getIndexes();
[
{ v: 2, key: { _id: 1 }, name: '_id_' },
{ v: 2, key: { name: 1 }, name: 'name_1' }
]
assb02> db.assb02.find().sort({name:1});
[
{
_id: ObjectId("634fb4839fc4505e08e35723"),
roll: 2,
name: 'anusha',
subject: 'dsa',
marks: 75
},
{
_id: ObjectId("634fb5b39fc4505e08e35727"),
roll: 6, name: 'geeta', subject: 'CN', marks: 90
},
{
_id: ObjectId("634fb45c9fc4505e08e35722"),
roll: 1, name: 'navin', subject: 'c++',
},
{
_id: ObjectId("634fb5b39fc4505e08e35727"),
roll: 6, name: 'geeta', subject: 'CN', marks: 90
},
{
_id: ObjectId("634fb4839fc4505e08e35723"),roll: 2,
name: 'anusha', subject: 'dsa', marks: 75
}
]
ssb02> db.assb02.find({name:'pravin'});
[
{
_id: ObjectId("634fb5969fc4505e08e35726"),
roll: 5, name: 'pravin', subject: 'CN', marks:
80
}
]
assb02> db.assb02.ensureIndex({roll:1},{unique:true});
[ 'roll_1' ]
assb02> db.assb02.insert({roll:6,name:'geeta',subject:'CN',marks:90});
Uncaught:
MongoBulkWriteError: E11000 duplicate key error collection: assb02.assb02 index:
roll_1 dup key:
{
roll: 6 }
Result: BulkWriteResult { result: { ok: 1, writeErrors: [ WriteError { err: { index: 0,
code: 11000, errmsg: 'E11000 duplicate key error collection: assb02.assb02 index:
roll_1 dup key: { roll: 6 }', errInfo: undefined, op: {
roll: 6, name:
'geeta',
subject: 'CN',
marks: 90,
_id: ObjectId("634fb7859fc4505e08e35728")
}
}
}
],
writeConcernErrors: [], insertedIds: [ { index: 0, _id:
ObjectId("634fb7859fc4505e08e35728") } ], nInserted: 0, nUpserted: 0, nMatched:
0, nModified: 0, nRemoved: 0, upserted: []
}
}
assb02> db.assb02.aggregate([{$group:{_id:"$subject",marks:{$min:"$marks"}}}]);
[{ _id: 'CN', marks: 80 },
{ _id: 'dsa', marks: 75 },
{ _id: 'c++', marks: 70 },
{ _id: 'toc', marks: 69 }
]
assb02> db.assb02.aggregate([{$group:{_id:"$subject",marks:{$max:"$marks"}}}]);
[
{ _id: 'CN', marks: 90 },
{ _id: 'dsa', marks: 75 },
{ _id: 'c++', marks: 70 },
{ _id: 'toc', marks: 69 }
]
assb02> db.assb02.aggregate([{$group:{_id:"$subject",marks:{$avg:"$marks"}}}]);
[
{ _id: 'dsa', marks: 75 },
{ _id: 'toc', marks: 69 },
{ _id: 'CN', marks: 85 },
{ _id: 'c++', marks: 70 }
]
assb02> db.assb02.aggregate([{$group:{_id:"$subject",marks:{$first:"$marks"}}}]);
[
{ _id: 'dsa', marks: 75 },
{ _id: 'toc', marks: 69 },
{ _id: 'CN', marks: 80 },
{ _id: 'c++', marks: 70 }
]
assb02> db.assb02.aggregate([{$group:{_id:"$subject",marks:{$last:"$marks"}}}]);
[
{ _id: 'dsa', marks: 75 },
{ _id: 'toc', marks: 69 },
{ _id: 'CN', marks: 90 },
{ _id: 'c++', marks: 70 }
]
assb02> db.assb02.aggregate([{$group:{_id:"$subject",marks:{$sum:"$marks"}}}]);
[
{ _id: 'dsa', marks: 75 },
{ _id: 'toc', marks: 69 },
{ _id: 'CN', marks: 170 },
{ _id: 'c++', marks: 140 }]
```

Assignment No.11 //Anand Bora(A-20)

Problem Statement: Write a program to implement Mongo DB database connectivity with any front end language to implement Database navigation operations(add, delete, edit etc.)

Code:

```
db.createCollection("stud");
{ ok: 1 }
assb03> db.stud.insert({name:'amit',marks:80});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fd4c49fc4505e08e35729") }
}
assb03> db.stud.insert({name:'amit',marks:90});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fd4d19fc4505e08e3572a") }
}
assb03> db.stud.insert({name:'shreya',marks:40});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fd4e59fc4505e08e3572b") }
}
assb03> db.stud.insert({name:'neha',marks:80});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fd4f89fc4505e08e3572c") }
}
assb03> db.stud.insert({name:'neha',marks:35});
{
acknowledged: true, insertedIds: { '0':
ObjectId("634fd5009fc4505e08e3572d") }
}
```

```
assb03> db.stud.find();
[
{
_id: ObjectId("634fd4c49fc4505e08e35729"),
name: 'amit',
marks: 80
},
{
_id: ObjectId("634fd4d19fc4505e08e3572a"),
name: 'amit', marks: 90
},
{
_id: ObjectId("634fd4e59fc4505e08e3572b"),
name: 'shreya', marks: 40
},
{
_id: ObjectId("634fd4f89fc4505e08e3572c"),
name: 'neha', marks: 80
},
{
_id: ObjectId("634fd5009fc4505e08e3572d"),
name: 'neha', marks: 35
}]
assb03> var mapfunc=function(){emit(this.name,1)} assb03> var
redfunc = function(key,values){return Array.sum(values);} assb03>
db.stud.mapReduce(mapfunc,redfunc,{out:"name_total"})
{ result: 'name_total', ok: 1 }
assb03> db.name_total.find();
[
{ _id: 'shreya', value: 1 },
{ _id: 'neha', value: 2 },
{ _id: 'amit', value: 2 }
]
assb03> var mapfuncm=function(){emit(this.name,this.marks);}
assb03> db.stud.mapReduce(mapfuncm,redfunc,{out:"name_total_marks"});
{ result: 'name_total_marks', ok: 1 }
assb03> db.name_total_marks.find();
[
{ _id: 'shreya', value: 40 },
{ _id: 'neha', value: 115 },
{ _id: 'amit', value: 170 }]
```