## Experiment No: 1

### Aim
Creation of a database using DDL commands.

### CO1
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

### Procedure

1. Create database
2. Use database

```
create database Employee_details;
use Employee_details;
```

Output

Action Output

| # | Time | Action | Message |
|---|---|---|---|
| 1 | 16:33:45 | create database Employee_details | 1 row(s) affected |
| 2 | 16:33:45 | use Employee_details | 0 row(s) affected |

3. Create table -emp

```
create table emp (
                eno int primary key, ename varchar(20),
                ehiredate date, esalary int, ecomission int,
                check (length(eno)>=3), check (esalary<=5000)
            );
```

4. Insert values into table

```
insert into emp (eno, ename, ehiredate, esalary) values (101, 'Ramesh', '1980-01-07', 5000);
insert into emp (eno, ename, ehiredate, esalary, ecomission) values (102, 'Ajay', '1985-05-07', 5000, 500);
insert into emp (eno, ename, ehiredate, esalary) values (103, 'Ravi', '1981-12-08', 1500);
insert into emp (eno, ename, ehiredate, esalary, ecomission) values (104, 'Nikesh', '1983-03-03', 3000, 700);
insert into emp (eno, ename, ehiredate, esalary) values (105, 'Ravi', '1985-05-07', 3000);
```

5. Add column-SAL

```
alter table emp add sal varchar(20);
select * from emp;
```

| eno | ename | ehiredate | esalary | ecomission | sal |
|---|---|---|---|---|---|
| 101 | Ramesh | 1980-01-07 | 5000 | NULL | NULL |
| 102 | Ajay | 1985-05-07 | 5000 | 500 | NULL |
| 103 | Ravi | 1981-12-08 | 1500 | NULL | NULL |
| 104 | Nikesh | 1983-03-03 | 3000 | 700 | NULL |
| 105 | Ravi | 1985-05-07 | 3000 | NULL | NULL |
| NULL | NULL | NULL | NULL | NULL | NULL |

6. Drop column-SAL

```
alter table emp drop sal;
select * from emp;
```

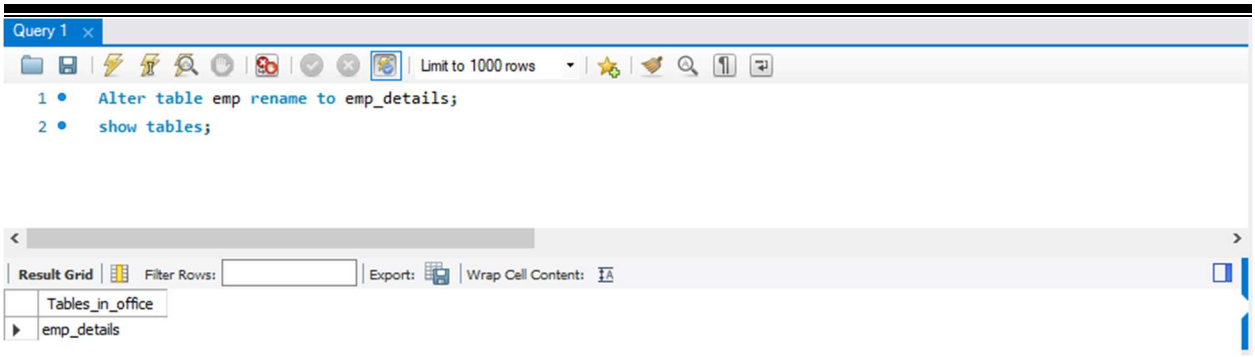
| eno | ename | ehiredate | esalary | ecomission |
|---|---|---|---|---|
| 101 | Ramesh | 1980-01-07 | 5000 | NULL |
| 102 | Ajay | 1985-05-07 | 5000 | 500 |
| 103 | Ravi | 1981-12-08 | 1500 | NULL |
| 104 | Nikesh | 1983-03-03 | 3000 | 700 |
| 105 | Ravi | 1985-05-07 | 3000 | NULL |
| NULL | NULL | NULL | NULL | NULL |

7. Rename column- ename to name

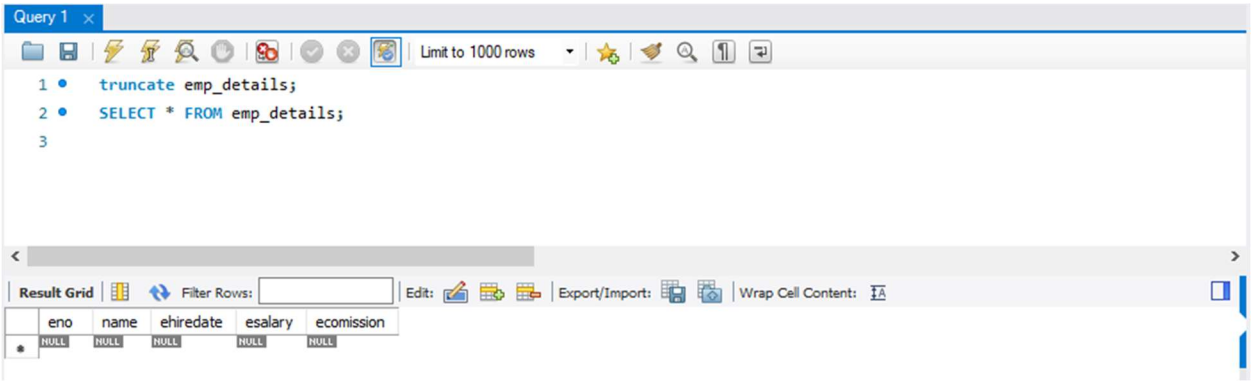
```
ALTER TABLE emp RENAME COLUMN ename to name;
SELECT * FROM emp;
```

| eno | name | ehiredate | esalary | ecomission |
|---|---|---|---|---|
| 101 | Ramesh | 1980-01-07 | 5000 | NULL |
| 102 | Ajay | 1985-05-07 | 5000 | 500 |
| 103 | Ravi | 1981-12-08 | 1500 | NULL |
| 104 | Nikesh | 1983-03-03 | 3000 | 700 |
| 105 | Ravi | 1985-05-07 | 3000 | NULL |
| NULL | NULL | NULL | NULL | NULL |

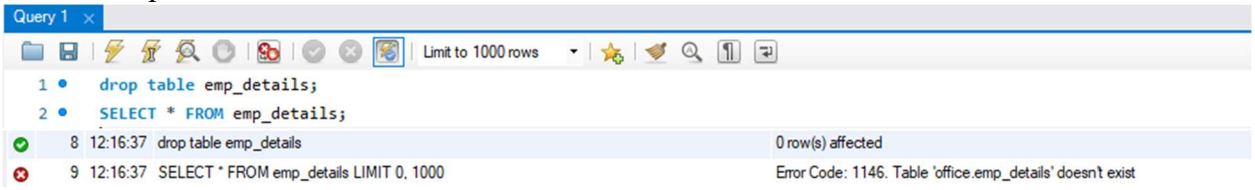
8. Rename table




9. Truncate table




10. Drop table




**Result**

The program was executed and the result was successfully obtained. Thus CO1 was obtained

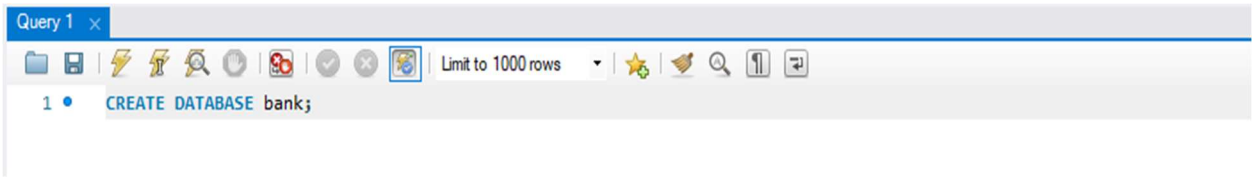
## Experiment No: 2

### Aim
Creation of a database using DML commands including integrity constraints.

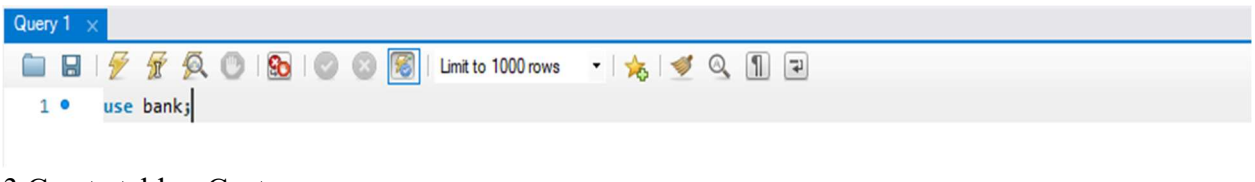
### CO2
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

### Procedure

1. Create database




2.Use database




3.Create table - Customer

```
CREATE TABLE BRANCH (
  BNAME VARCHAR(20) PRIMARY KEY,
  CITY VARCHAR(30) CHECK(CITY IN ('NAGPUR', 'DELHI', 'BANGLORE', 'BOMBAY')) NOT NULL
);
```

4. Create table - Customer

```
CREATE TABLE CUSTOMER (
  CNAME VARCHAR(15) PRIMARY KEY,
  CITY VARCHAR(30) NOT NULL
);
```

5.Create table - Deposit

```
CREATE TABLE DEPOSIT (
  ACTNO VARCHAR(5) PRIMARY KEY CHECK (ACTNO LIKE 'D%'),
  CNAME VARCHAR(15) REFERENCES CUSTOMER(CNAME),
  BNAME VARCHAR(20) REFERENCES BRANCH(BNAME),
  AMOUNT DECIMAL(8,2) NOT NULL CHECK (AMOUNT > 0),
  ADATE DATE
);
```

6. Create table - Borrow

```
CREATE TABLE BORROW (
        LOANNO VARCHAR(8) check(LOANNO like 'L%') primary key,
        CNAME VARCHAR(15) REFERENCES CUSTOMER(CNAME),
        BNAME VARCHAR(20) REFERENCES BRANCH(BNAME),
        AMOUNT FLOAT(8) NOT NULL CHECK (AMOUNT > 0)
        );
```

7. Insert values in table - Customer

```
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('ANIL', 'CALCUTTA');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('SUNIL', 'DELHI');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('MEHUL', 'BARODA');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('MANDAR', 'PATNA');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('MADHURI', 'NAGPUR');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('PRAMOD', 'NAGPUR');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('SANDIP', 'SURAT');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('SHIVANI', 'BOMBAY');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('KRANTI', 'BOMBAY');
INSERT INTO CUSTOMER (CNAME, CITY) VALUES ('NAREN', 'BOMBAY');
```

8. Insert values in table - Branch

```
INSERT INTO BRANCH (BNAME, CITY) VALUES ('VRCE', 'NAGPUR');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('AJNI', 'NAGPUR');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('KAROLBAGH', 'DELHI');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('CHANDNI', 'DELHI');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('DHARAMPETH', 'NAGPUR');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('MG ROAD', 'BANGLORE');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('ANDHERI', 'BOMBAY');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('NEHRU PALACE', 'DELHI');
INSERT INTO BRANCH (BNAME, CITY) VALUES ('POWAI', 'BOMBAY');
```

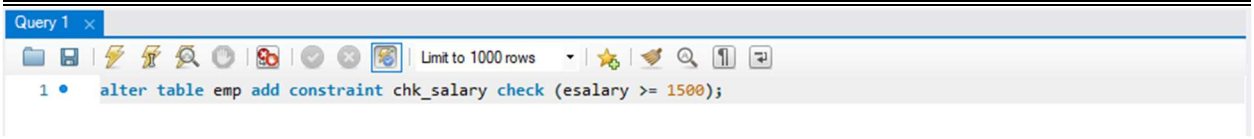
9. Insert values in table - Borrow

```
INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L201', 'ANIL', 'VRCE', 1000.00);
INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L206', 'MEHUL', 'AJNI', 5000.00);
INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L311', 'SUNIL', 'DHARAMPETH', 3000.00);
INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L321', 'MADHURI', 'ANDHERI', 2000.00);
INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L371', 'PRAMOD', 'VIRAR', 8000.00);
INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L481', 'KRANTI', 'NEHRU PLACE', 3000.00);
```

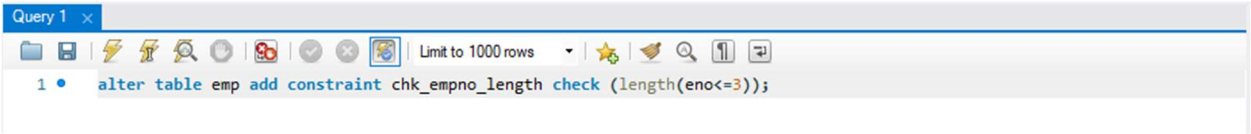
10. Insert values into table - Deposit

```
INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D100', 'ANIL', 'VRCE', 1000.00, '1995-03-01');
INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D101', 'SUNIL', 'ANJNI', 500.00, '1996-01-04');
INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D102', 'MEHUL', 'KAROLBAGH', 3500.00, '1995-11-17');
INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D104', 'MADHURI', 'CHANDNI', 1200.00, '1995-12-17');
INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D105', 'PRAMOD', 'MG ROAD', 3000.00, '1996-03-27');
INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D106', 'SANDIP', 'ANDHERI', 2000.00, '1996-03-31');
INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D107', 'SHIVANI', 'VIRAR', 1000.00, '1995-09-05');
INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D108', 'KRANTI', 'NEHRU PLACE', 5000.00, '1995-07-02');
INSERT INTO Deposit (ACTNO, CNAME, BNAME, AMOUNT, ADATE) VALUES ('D109', 'MINU', 'POWAI', 7000.00, '1995-08-10');
```

11. Add constraint - salary <= 5000




12. Add constraint - length(eno) >= 3




**Result**

The program was executed and the result was successfully obtained. Thus CO1 was obtained

**Experiment No: 3**

**Aim**

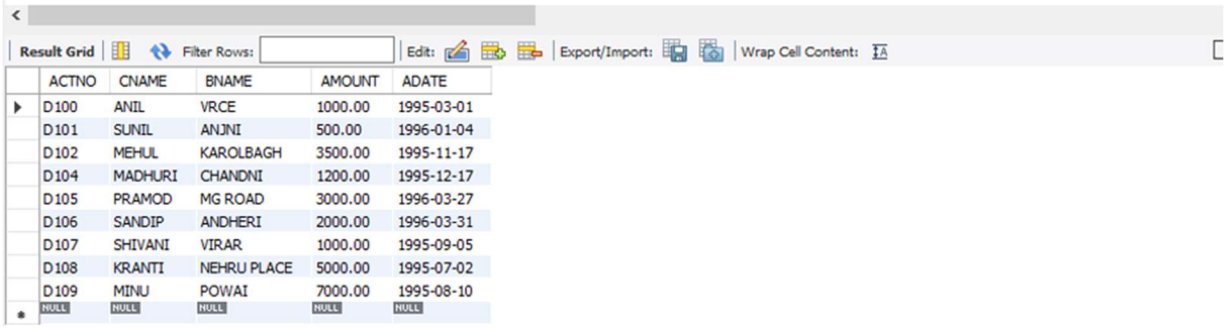
To familiarize with selecting data from single table

## CO1

Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

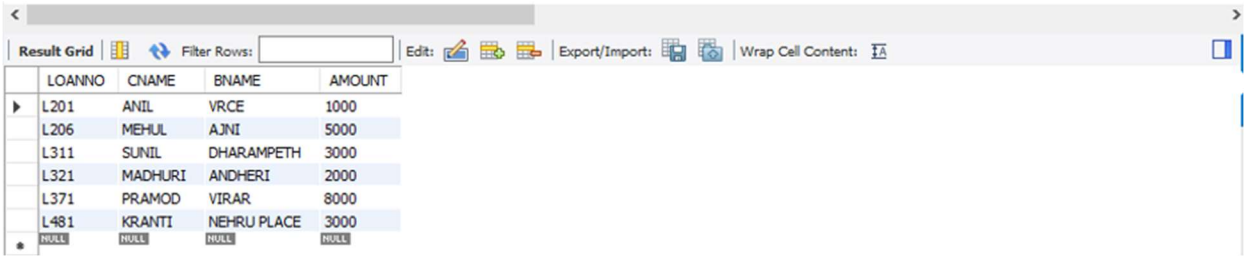
## Procedure

1. List all data from table deposit.
   select * from deposit;



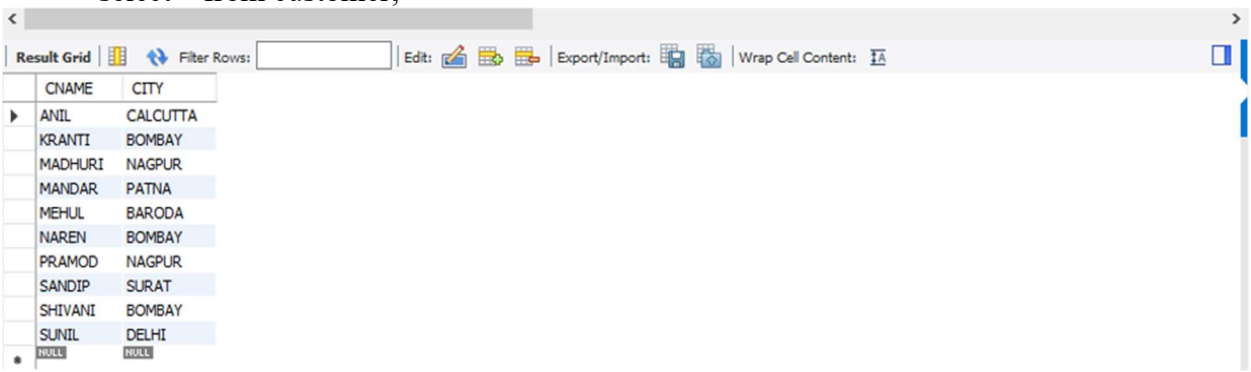
| ACTNO | CNAME | BNAME | AMOUNT | ADATE |
|---|---|---|---|---|
| D100 | ANIL | VRCE | 1000.00 | 1995-03-01 |
| D101 | SUNIL | ANJNI | 500.00 | 1996-01-04 |
| D102 | MEHUL | KAROLBAGH | 3500.00 | 1995-11-17 |
| D104 | MADHURI | CHANDNI | 1200.00 | 1995-12-17 |
| D105 | PRAMOD | MG ROAD | 3000.00 | 1996-03-27 |
| D106 | SANDIP | ANDHERI | 2000.00 | 1996-03-31 |
| D107 | SHIVANI | VIRAR | 1000.00 | 1995-09-05 |
| D108 | KRANTI | NEHRU PLACE | 5000.00 | 1995-07-02 |
| D109 | MINU | POWAI | 7000.00 | 1995-08-10 |
| NULL | NULL | NULL | NULL | NULL |

2. List all data from borrow
   select * from borrow;



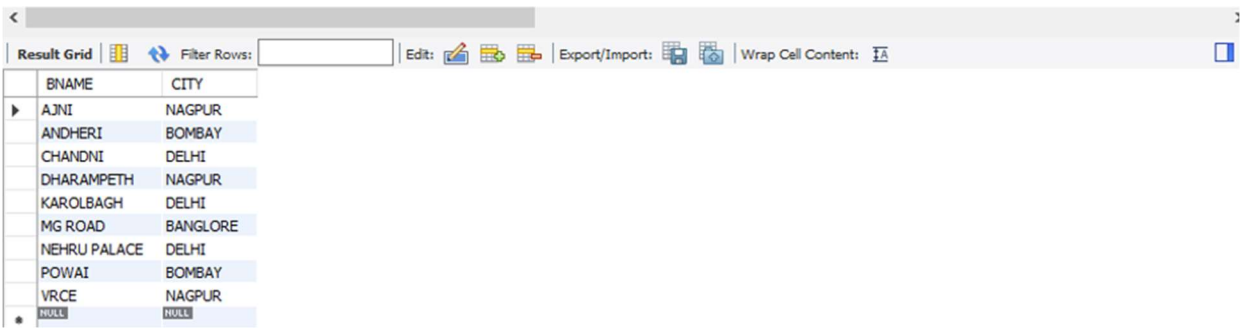
| LOANNO | CNAME | BNAME | AMOUNT |
|---|---|---|---|
| L201 | ANIL | VRCE | 1000 |
| L206 | MEHUL | AJNI | 5000 |
| L311 | SUNIL | DHARAMPETH | 3000 |
| L321 | MADHURI | ANDHERI | 2000 |
| L371 | PRAMOD | VIRAR | 8000 |
| L481 | KRANTI | NEHRU PLACE | 3000 |
| NULL | NULL | NULL | NULL |

3. List all data from customer
   select * from customer;



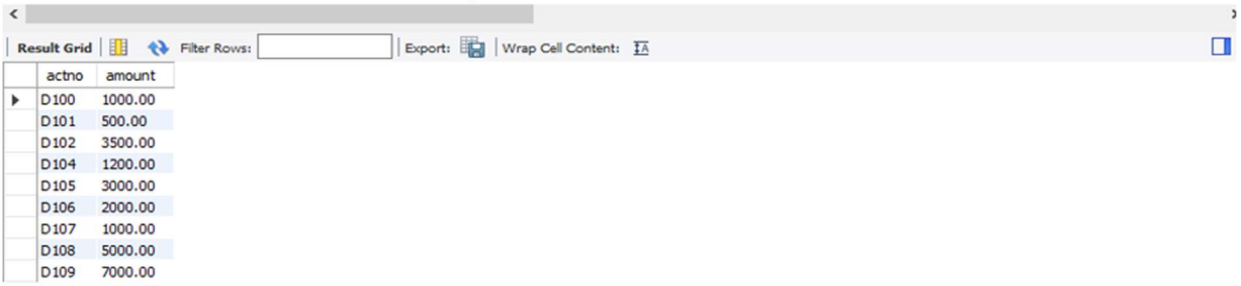
| CNAME | CITY |
|---|---|
| ANIL | CALCUTTA |
| KRANTI | BOMBAY |
| MADHURI | NAGPUR |
| MANDAR | PATNA |
| MEHUL | BARODA |
| NAREN | BOMBAY |
| PRAMOD | NAGPUR |
| SANDIP | SURAT |
| SHIVANI | BOMBAY |
| SUNIL | DELHI |
| NULL | NULL |

4. List all data from branch
   select * from branch;



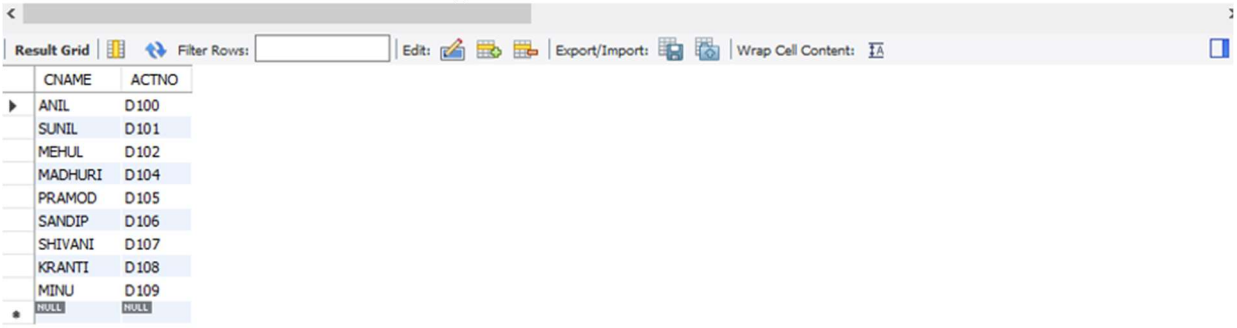
| BNAME | CITY |
|---|---|
| AJNI | NAGPUR |
| ANDHERI | BOMBAY |
| CHANDNI | DELHI |
| DHARAMPETH | NAGPUR |
| KAROLBAGH | DELHI |
| MG ROAD | BANGLORE |
| NEHRU PALACE | DELHI |
| POWAI | BOMBAY |
| VRCE | NAGPUR |

5. Give account no and amount of deposit
   select actno ,amount from deposit;



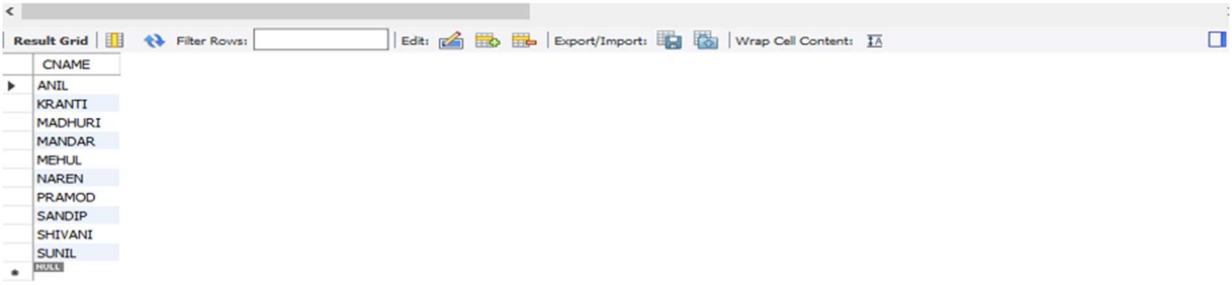
| actno | amount |
|---|---|
| D100 | 1000.00 |
| D101 | 500.00 |
| D102 | 3500.00 |
| D104 | 1200.00 |
| D105 | 3000.00 |
| D106 | 2000.00 |
| D107 | 1000.00 |
| D108 | 5000.00 |
| D109 | 7000.00 |

6. Give customer name and account no of depositors
   select actno ,cname from deposit;



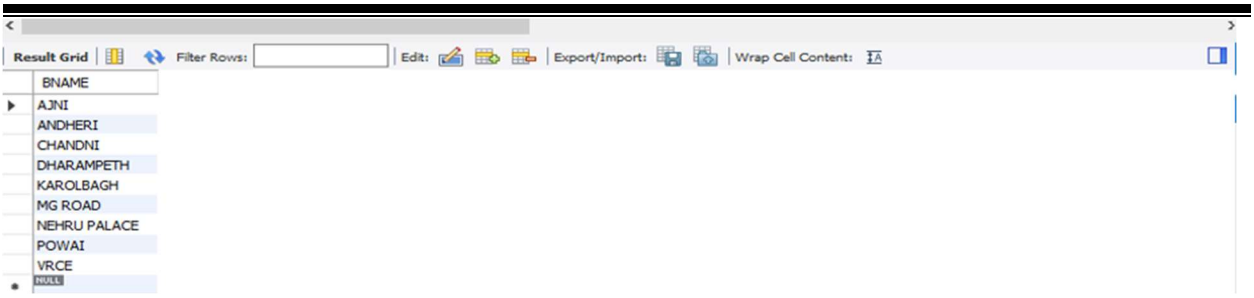
| CNAME | ACTNO |
|---|---|
| ANIL | D100 |
| SUNIL | D101 |
| MEHUL | D102 |
| MADHURI | D104 |
| PRAMOD | D105 |
| SANDIP | D106 |
| SHIVANI | D107 |
| KRANTI | D108 |
| MINU | D109 |

7. Give name of customers
   select cname from customer;



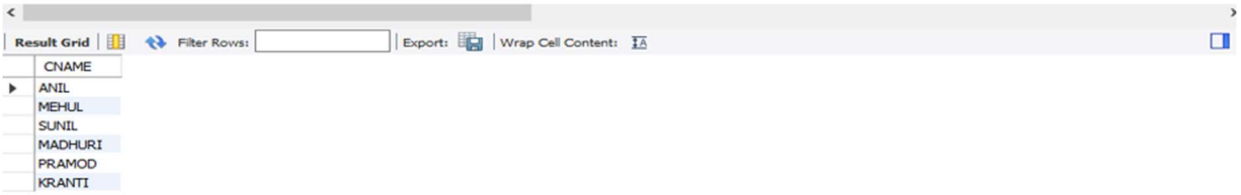
| CNAME |
|---|
| ANIL |
| KRANTI |
| MADHURI |
| MANDAR |
| MEHUL |
| NAREN |
| PRAMOD |
| SANDIP |
| SHIVANI |
| SUNIL |

8. Give name of branches
   select bname from branch;



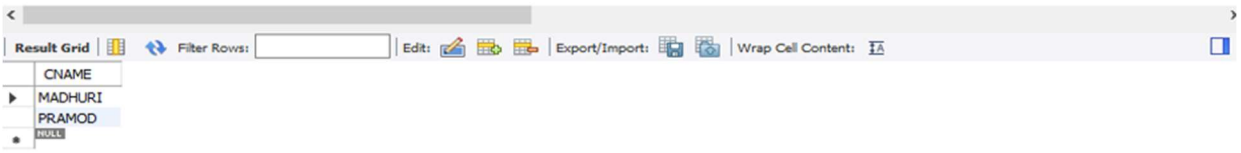
| BNAME |
|---|
| AJNI |
| ANDHERI |
| CHANDNI |
| DHARAMPETH |
| KAROLBAGH |
| MG ROAD |
| NEHRU PALACE |
| POWAI |
| VRCE |
| NULL |

9. Give name of borrows
   select cname from borrow;



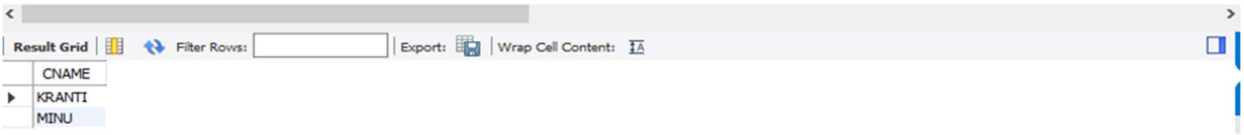
| CNAME |
|---|
| ANIL |
| MEHUL |
| SUNIL |
| MADHURI |
| PRAMOD |
| KRANTI |

10. Give names of customer living in city Nagpur
    select cname from customer where city="nagpur";



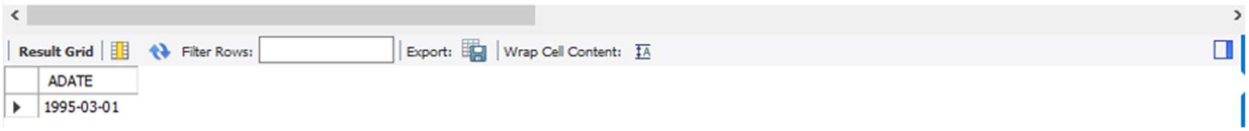
| CNAME |
|---|
| MADHURI |
| PRAMOD |
| NULL |

11. Give names of depositors having amount greater than 4000
    select cname from deposit where amount>4000;



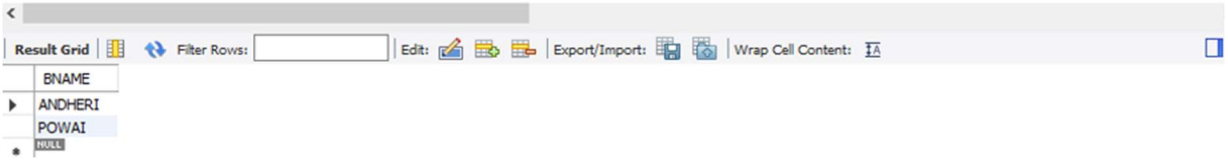
| CNAME |
|---|
| KRANTI |
| MINU |

12. Give account date of Anil
    select adate from deposite where cname="anil";



| ADATE |
|---|
| 1995-03-01 |

13. Give name of all branches located in Bombay
    select bname from branch where city="bombay";

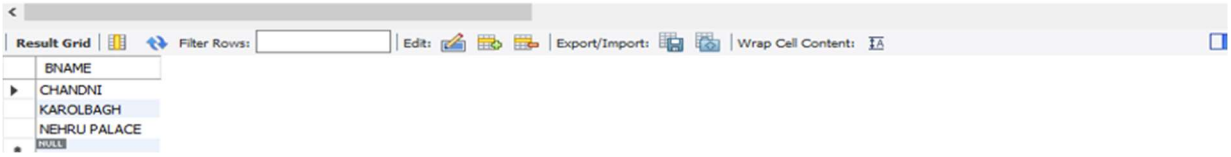


| BNAME |
|---|
| ANDHERI |
| POWAI |
| NULL |

14. Give name of borrower having loan number l205
    select cname from borrow where loanno="l205";

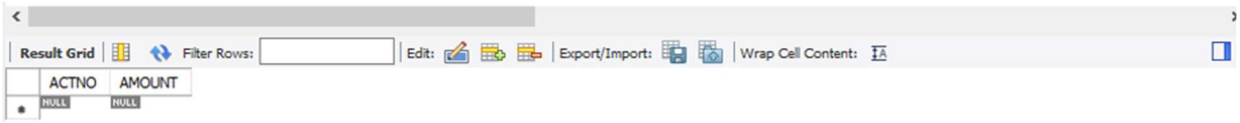

15. Give names of depositors having account at VRCE
    select cname from deposite where bname="vrce";


16. Give names of all branched located in city Delhi
    select bname from branch where city="delhi";



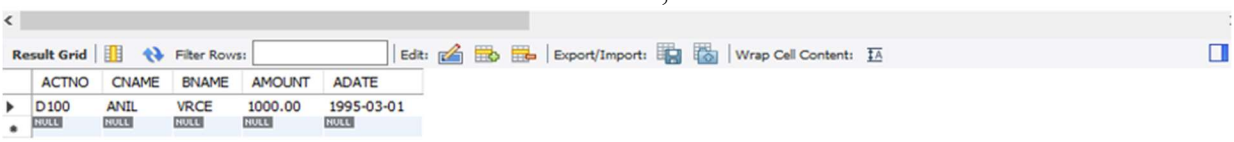

17. Give name of the customers who opened account date ‘1-12-96’
    select cname from deposite where adate='1996-12-01';


18. Give account no and deposit amount of customers having account opened between dates ‘1- 12-96’ and ‘1-5-96’
    select actno , cname from deposite where adate between '1996-12-01' and '1996-05-01';




19. Give name of the city where branch KAROL BAGH is located
    select city from branch where bname="karolbagh";


20. Give details of customer ANIL
    select * from customer where cname="anil";



**Result**

The program was executed and the result was successfully obtained. Thus CO1 was obtained

**Experiment No: 4**

**Aim**

To familiarize with Set Operations.

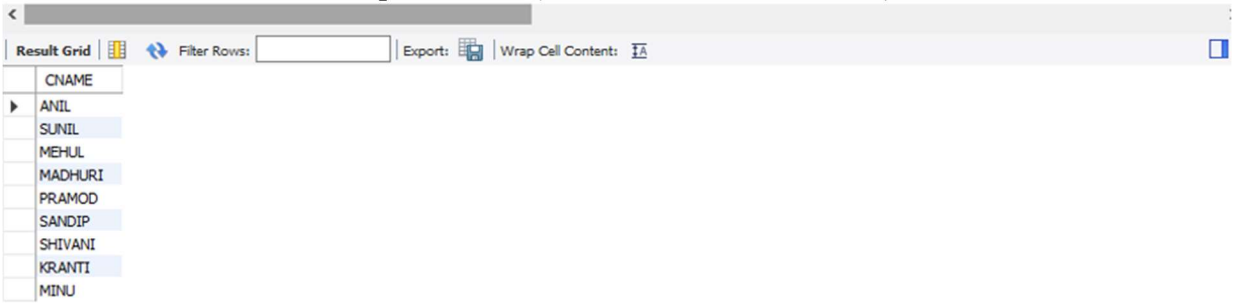
**CO1**

Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

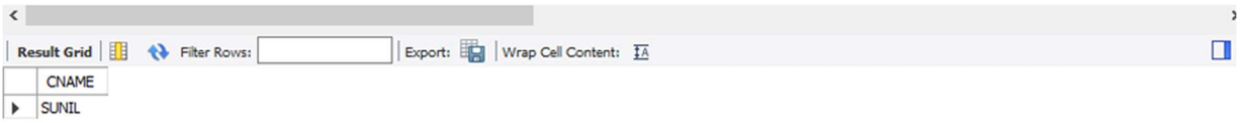
**Procedure**

1. List all the customers who are depositors but not borrowers.
   select cname from deposit where cname not in (select cname from borrow);

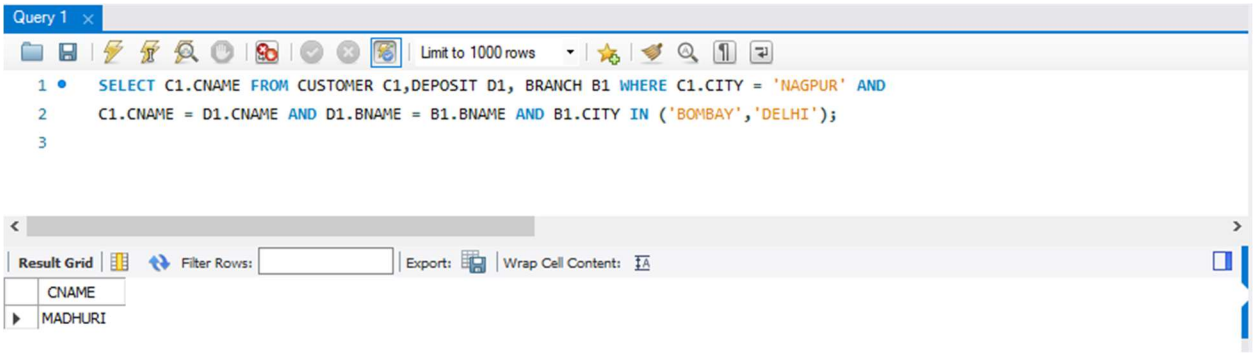

2. List all the customers who are both depositors and borrowers.
   select cname from deposit union (select cname from borrow);



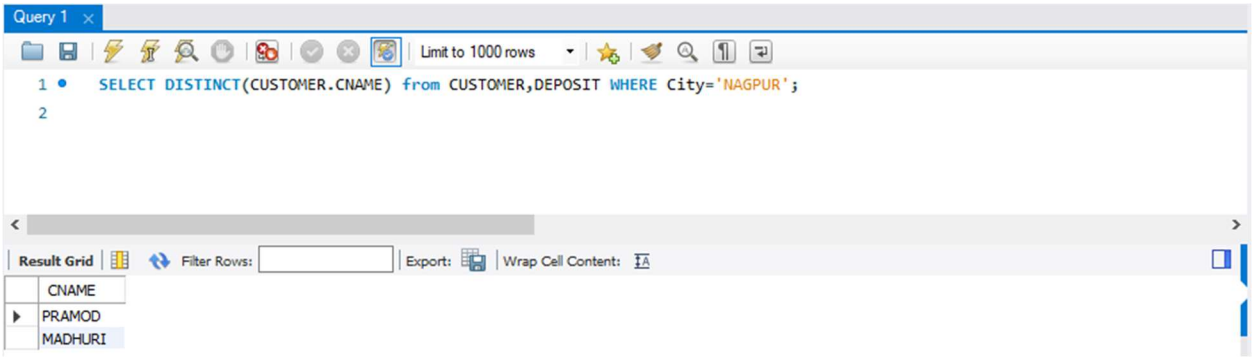

3. List all the depositors having deposit in all the branches where Sunil is having Account
   select D1.cname from deposit D1where D1.bname in (select D2.bname from deposit where D2.cname="sunil");



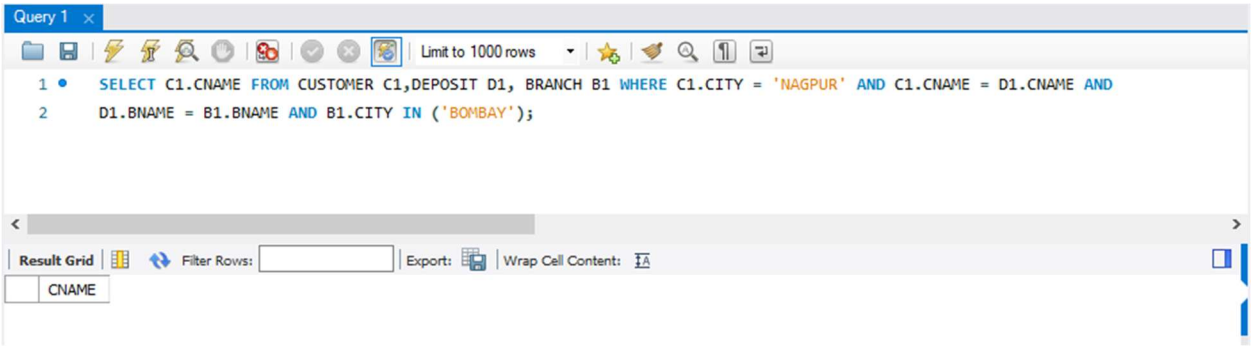
4. List all the customers living in city NAGPUR and having branch city BOMBAY or DELHI



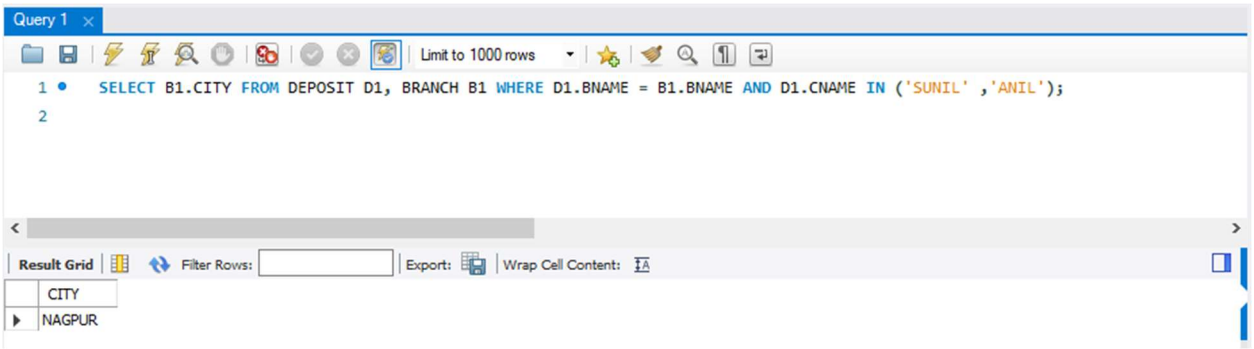

5. List all the depositors living in city NAGPUR




6. List all the depositors living in the city NAGPUR and having branch in city BOMBAY




7. List the branch cities of Anil and Sunil




8. List the customers having deposit greater than 1000 and loan less than 10000.

```
SELECT DISTINCT D1.CNAME FROM deposit D1, borrow B1 WHERE D1.AMOUNT>1000 AND B1.AMOUNT<10000;
```

| CNAME |
|---|
| MEHUL |
| MADHURI |
| PRAMOD |
| SANDIP |
| KRANTI |
| MINU |

9. List the cities of depositors having branch VRCE.

```
SELECT B1.CITY FROM deposit D1, branch B1 WHERE D1.BNAME=B1.BNAME AND B1.BNAME='VRCE';
```

| CITY |
|---|
| NAGPUR |

10. List the depositors having amount less than 1000 and living in the same city as Anil

```
SELECT D1.CNAME FROM deposit D1, customer C1 WHERE AMOUNT<1000 AND C1.CITY=(C1.CNAME='ANIL');
```

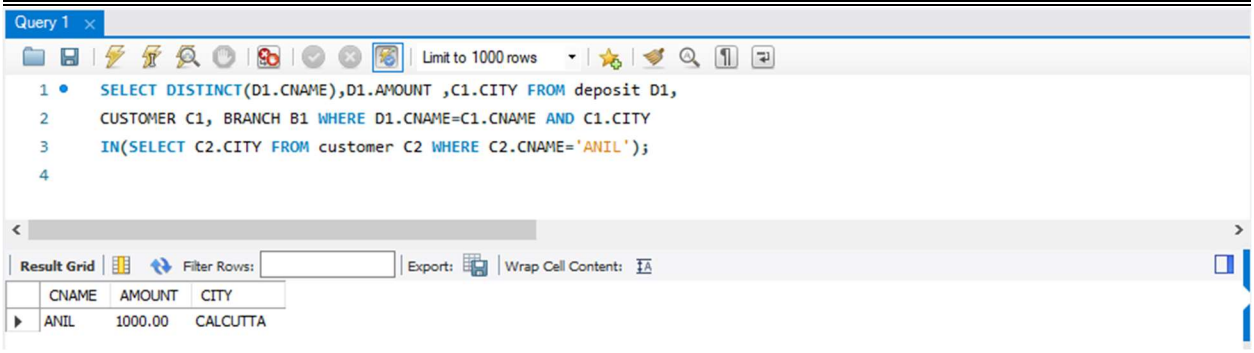
| CNAME |
|---|
| SUNIL |

11. List all the cities where branches of Anil and Sunil are locate

```
SELECT B1.CITY FROM BRANCH B1 WHERE B1.BNAME IN (SELECT D1.BNAME FROM DEPOSIT D1 WHERE D1.CNAME IN ('ANIL','SUNIL'));
```

| CITY |
|---|
| NAGPUR |

12. List the amount for the depositors living in the city where Anil is living




**Result**

The program was executed and the result was successfully obtained. Thus CO1 was obtained

## Experiment No: 5

**Aim**

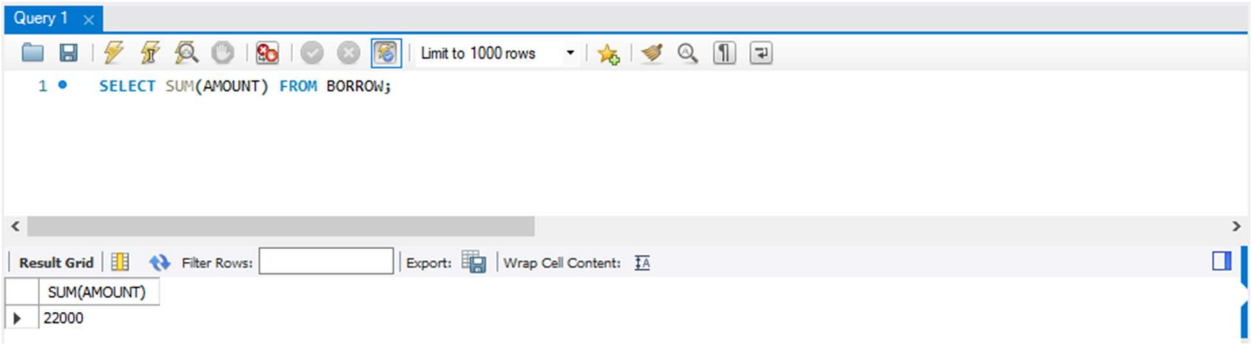
To familiarize with Aggregate Functions.

**CO1**

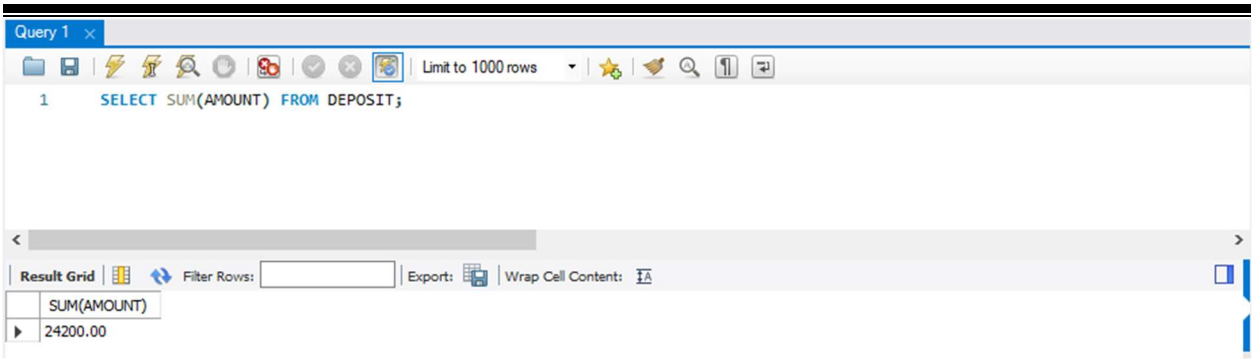
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

**Procedure**

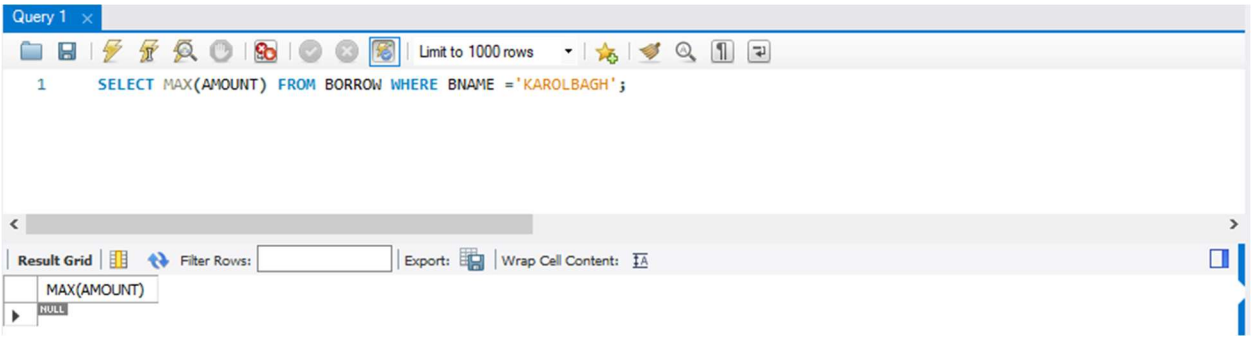
1. List total loan



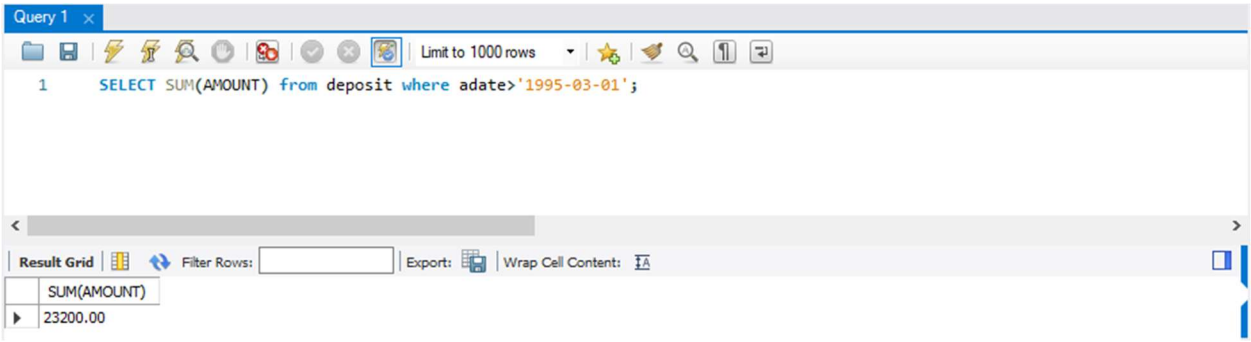

2. List total deposit



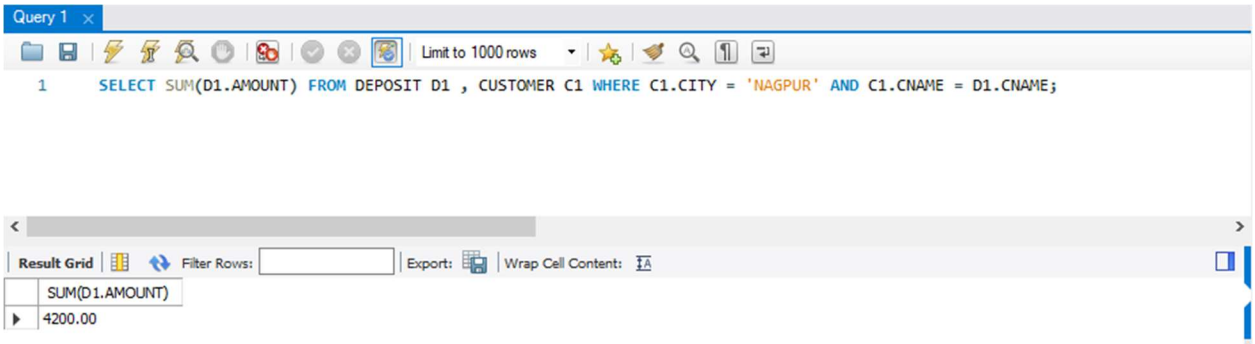

3. List total loan taken from KAROLBAGH branch



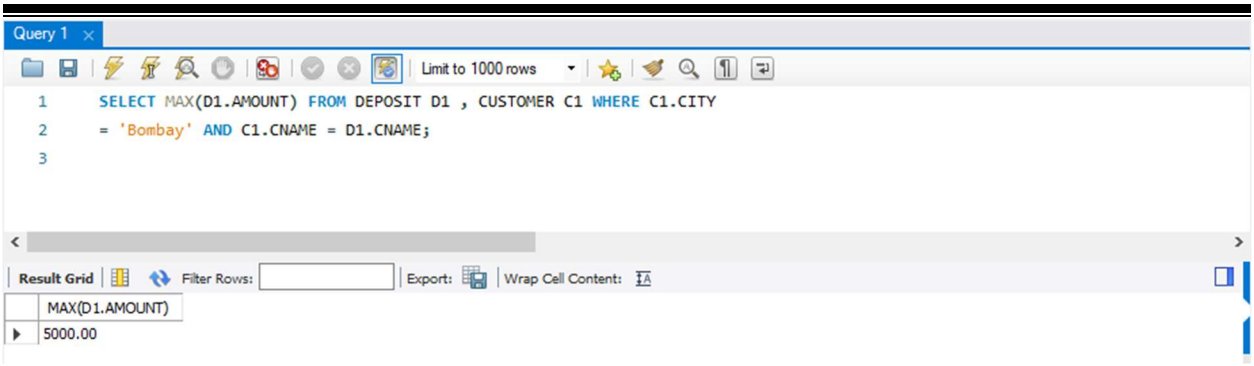

4. List total deposit of customers having account date later than 1-Jan-96



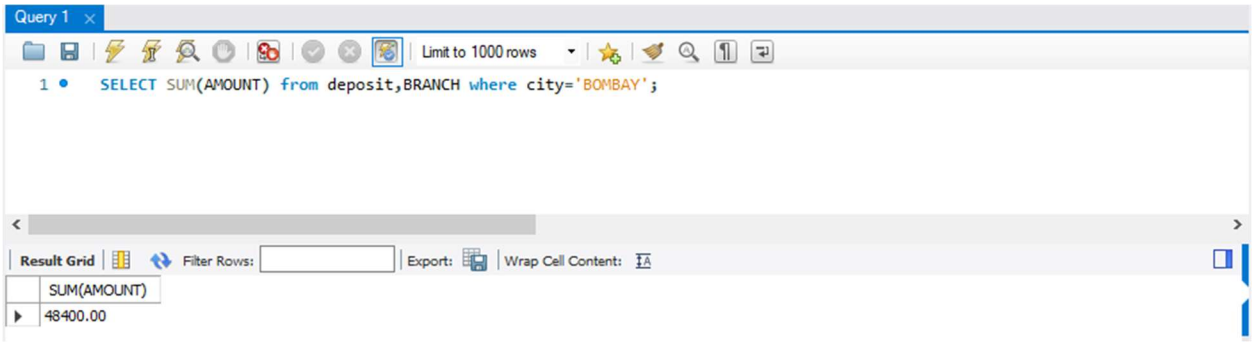

5. List total deposit of customers living in city NAGPUR



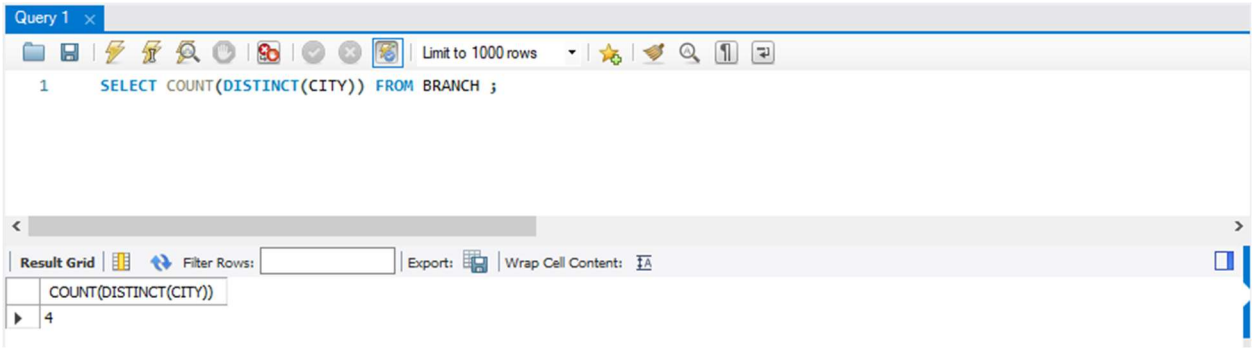

6. List maximum deposit of customer living in Bombay



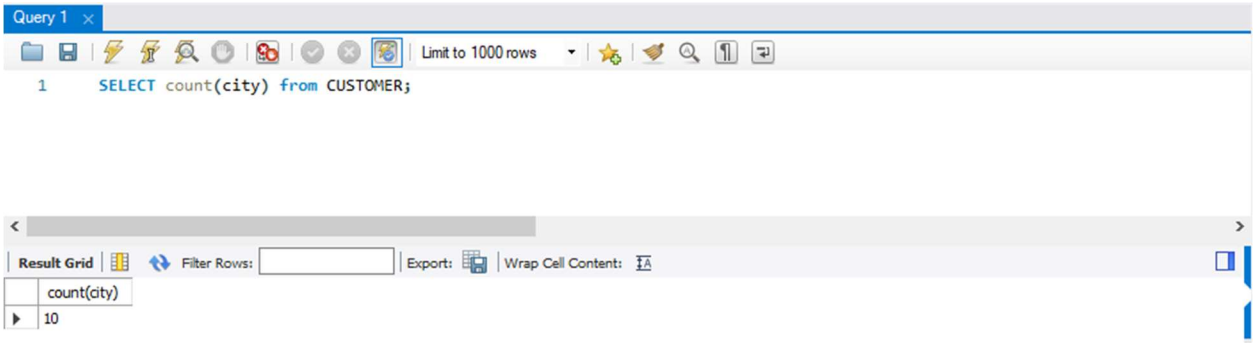

7. List total deposit of customer having branch in BOMBAY



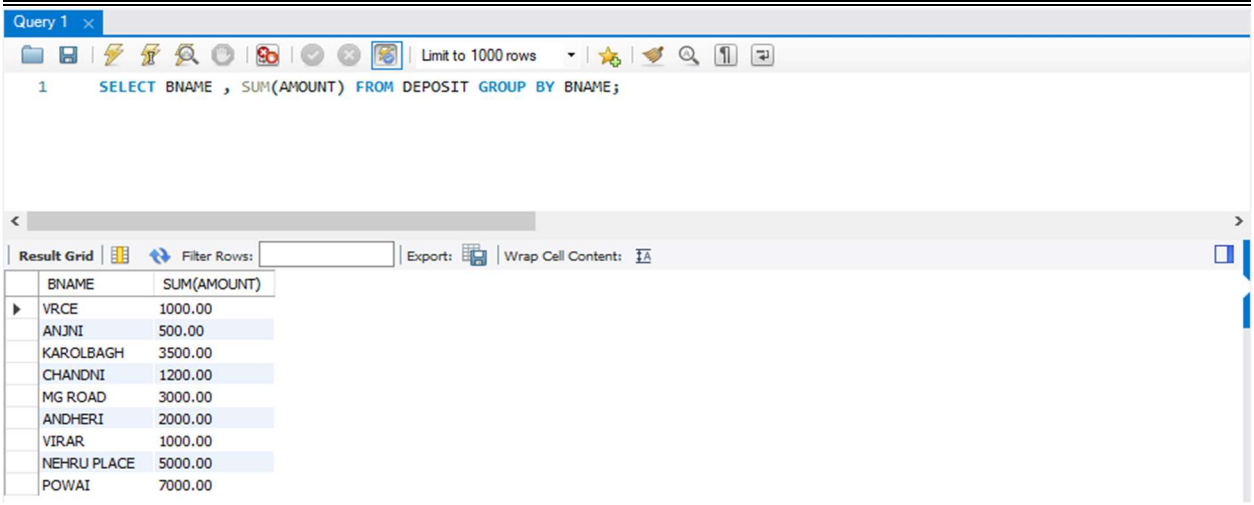

8. Count total number of branch cities



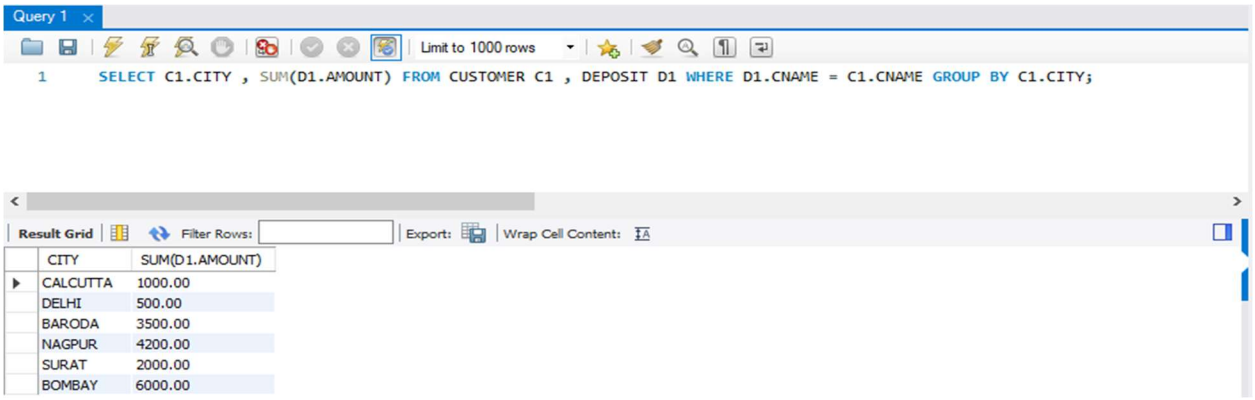

9. Count total number of customers cities



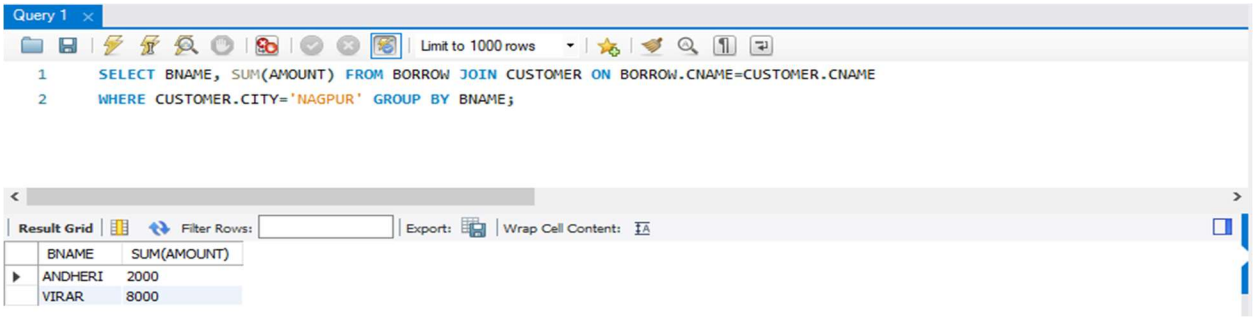

10. Give branch names and branch wise deposit



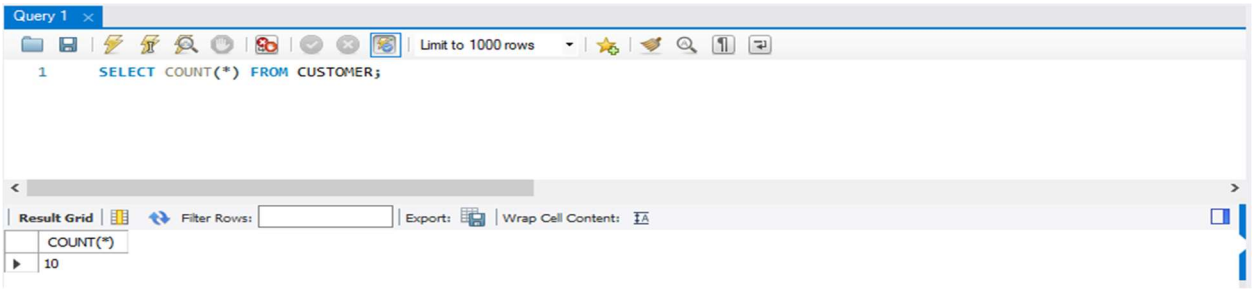

11. Give city wise name and branch wise deposit



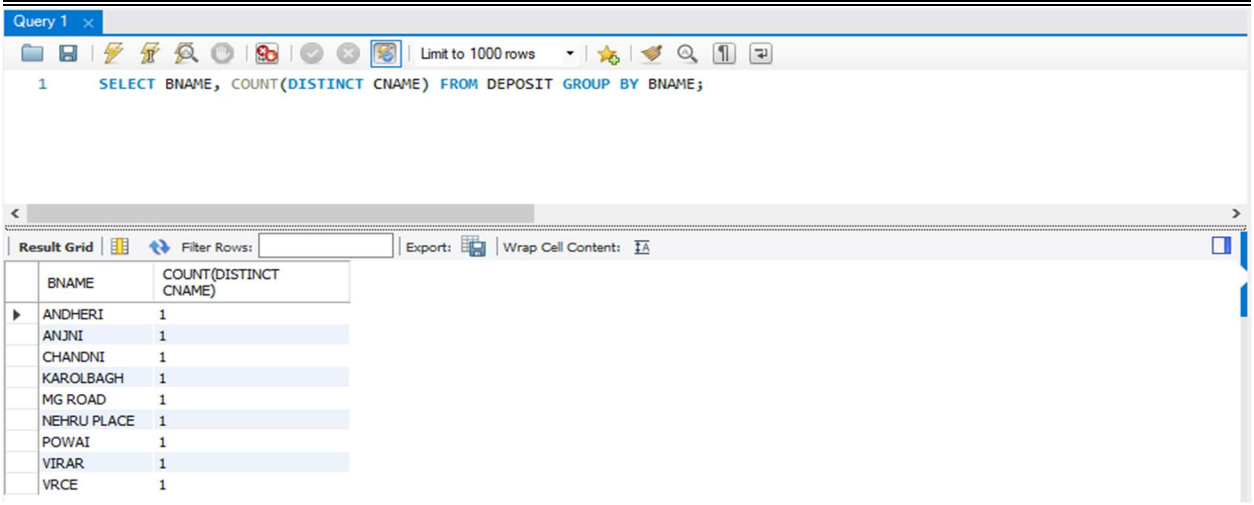

12. Give the branch wise loan of customer living in NAGPUR



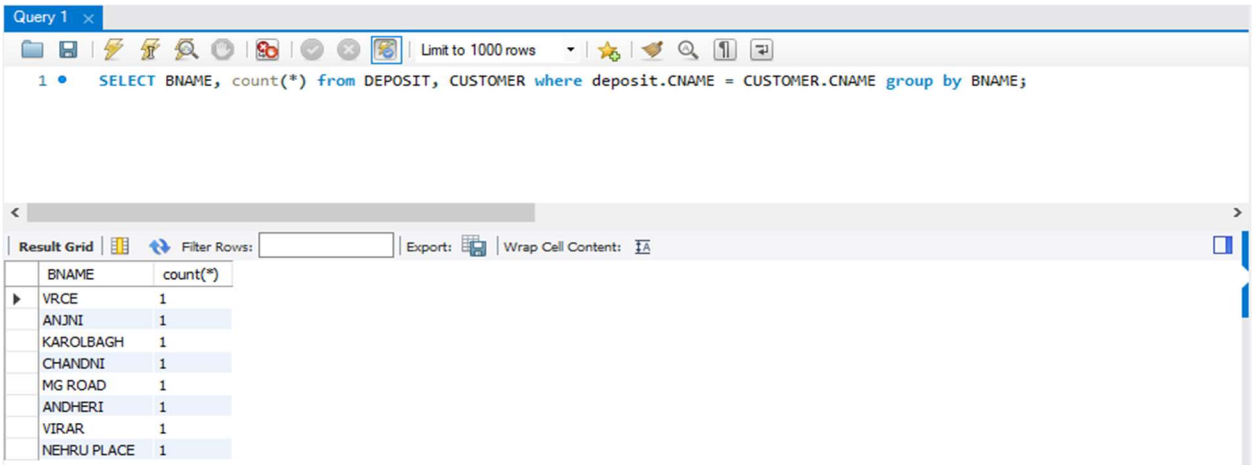

13. Count total number of customers



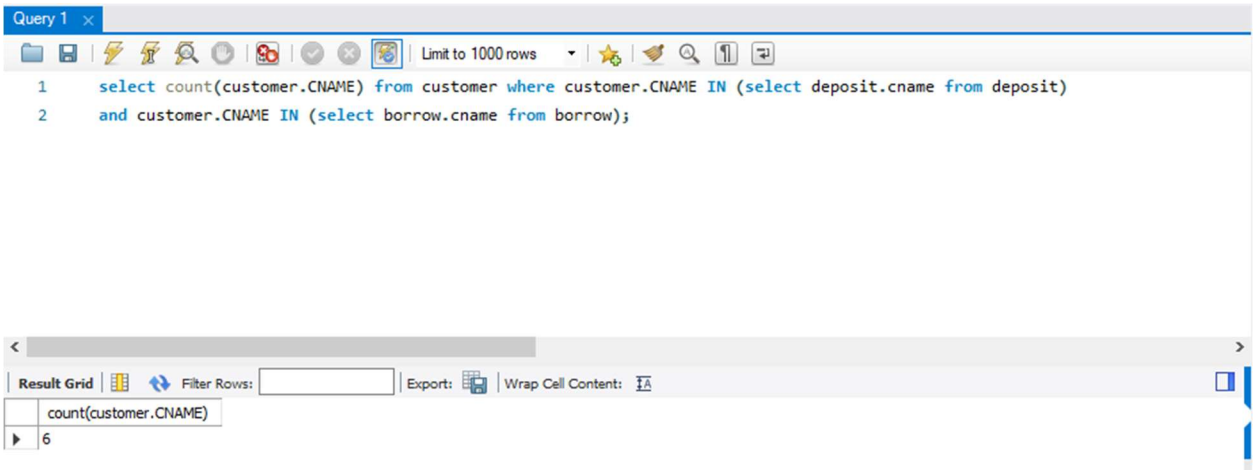

14. Count total number of depositors branch wise




15. Give maximum loan from branch VRCE




16. Give the number of customers who are depositors as well as borrowers




## Result

The program was executed and the result was successfully obtained. Thus CO1 was obtained

**Experiment no: 6**

**Aim**

To familiarize with Join or Cartesian Product

**CO1**

Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

**Procedure**

1. Give name of customers having living city BOMBAY and branch city NAGPUR

```
SELECT D1.CNAME,D1.BNAME,C1.CNAME,C1.CITY,B1.CITY,B1.BNAME
FROM DEPOSIT D1,CUSTOMER C1,BRANCH B1 WHERE C1.CITY = 'BOMBAY'
AND B1.CITY = 'NAGPUR' AND D1.CNAME = C1.CNAME AND D1.BNAME = B1.BNAME;
```

| CNAME | BNAME | CNAME | CITY | CITY | BNAME |
|---|---|---|---|---|---|

2. Give names of customers having the same living city as their branch city

```
SELECT DISTINCT(CUSTOMER.CNAME), BRANCH.CITY
FROM BRANCH, CUSTOMER WHERE BRANCH.CITY = CUSTOMER.CITY;
```

| CNAME | CITY |
|---|---|
| KRANTI | BOMBAY |
| MADHURI | NAGPUR |
| NAREN | BOMBAY |
| PRAMOD | NAGPUR |
| SHIVANI | BOMBAY |
| SUNIL | DELHI |

3. Give names of customers who are borrowers as well as depositors and having city NAGPUR.

```
SELECT C1.CNAME FROM CUSTOMER C1,DEPOSIT D1,BORROW B1
WHERE C1.CITY='NAGPUR' AND C1.CNAME=D1.CNAME AND D1.CNAME = B1.CNAME;
```

| CNAME |
|---|
| MADHURI |
| PRAMOD |

4. Give names of borrowers having deposit amount greater than 1000 and loan amount greater than 2000.

```
SELECT BR1.CNAME, BR1.AMOUNT, D1.CNAME, D1.AMOUNT
FROM BORROW BR1,DEPOSIT D1 WHERE D1.CNAME = BR1.CNAME AND D1.AMOUNT > 1000 AND BR1.AMOUNT > 2000;
```

| CNAME | AMOUNT | CNAME | AMOUNT |
|---|---|---|---|
| MEHUL | 5000 | MEHUL | 3500.00 |
| PRAMOD | 8000 | PRAMOD | 3000.00 |
| KRANTI | 3000 | KRANTI | 5000.00 |

5. Give names of depositors having the same branch as the branch of Sunil

```
SELECT D1.CNAME FROM DEPOSIT D1 WHERE D1.BNAME
IN (SELECT D2.BNAME FROM DEPOSIT D2 WHERE D2.CNAME = 'SUNIL');
```

| CNAME |
|---|
| SUNIL |

6. Give names of borrowers having loan amount greater than the loan amount of Pramod

```
SELECT BR1.CNAME, BR1.AMOUNT FROM BORROW BR1
WHERE BR1.AMOUNT > ALL (SELECT BR2.AMOUNT FROM BORROW BR2 WHERE BR2.CNAME = 'PRAMOD');
```

| CNAME | AMOUNT |
|---|---|

7. Give the name of the customer living in the city where branch of depositor Sunil is located.

```
SELECT C.CNAME FROM CUSTOMER C WHERE C.CITY IN (SELECT B.CITY FROM BRANCH B
WHERE B.BNAME IN (SELECT D.BNAME FROM DEPOSIT D WHERE D.CNAME='SUNIL'));
```

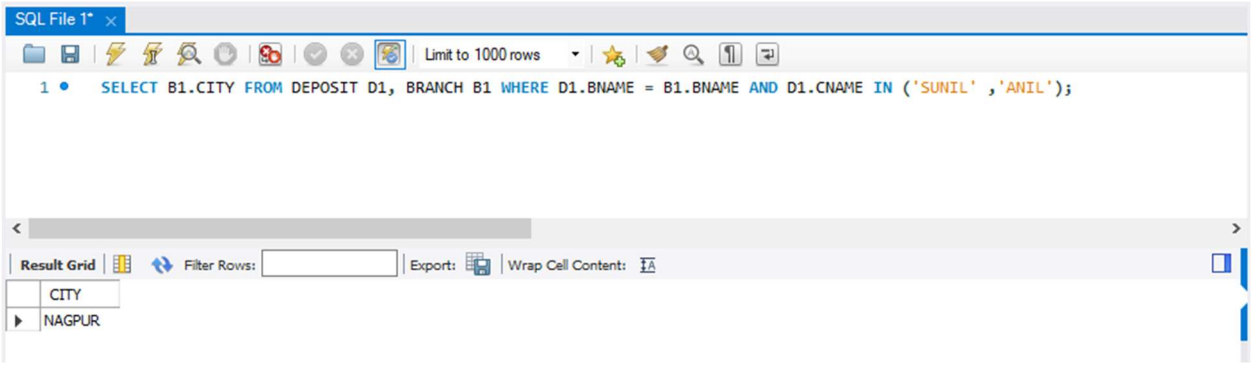
| CNAME | AMOUNT |
|---|---|

8. Give branch city and living city of Pramod

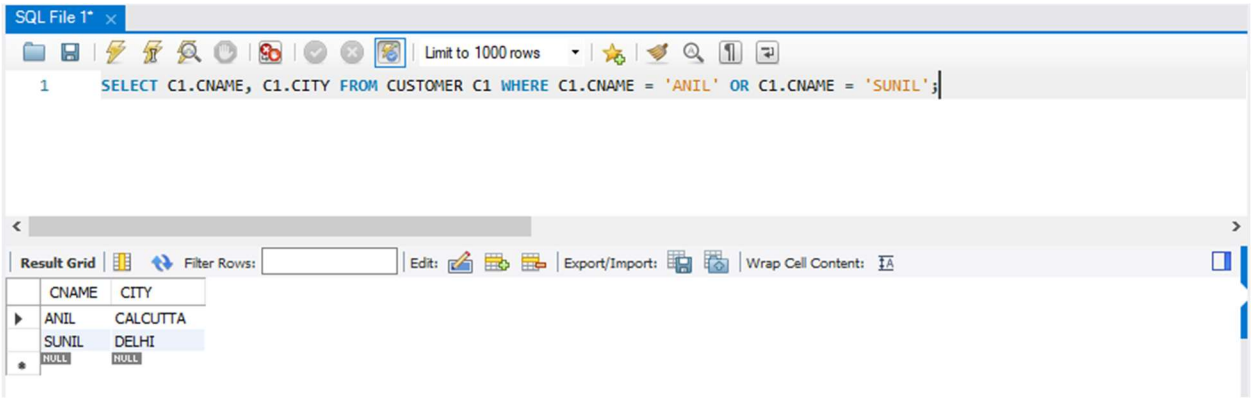
```
SELECT B1.CITY , C1.CITY FROM BRANCH B1,CUSTOMER C1, DEPOSIT D1
WHERE C1.CNAME = 'PRAMOD' AND C1.CNAME = D1.CNAME AND D1.BNAME = B1.BNAME;
```

| CNAME | AMOUNT |
|---|---|

9. Give branch city of Sunil and branch city of Anil




10. Give the living city of Anil and the living city of Sunil




**Result**

The program was executed and the result was successfully obtained. Thus CO1 was obtained

**Experiment no: 7**

**Aim**

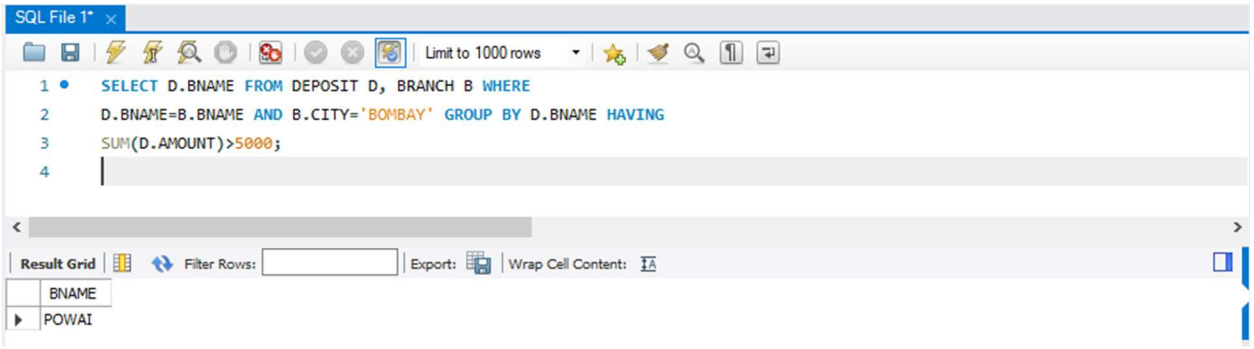
To familiarize with Group By and Having Clauses

## CO1

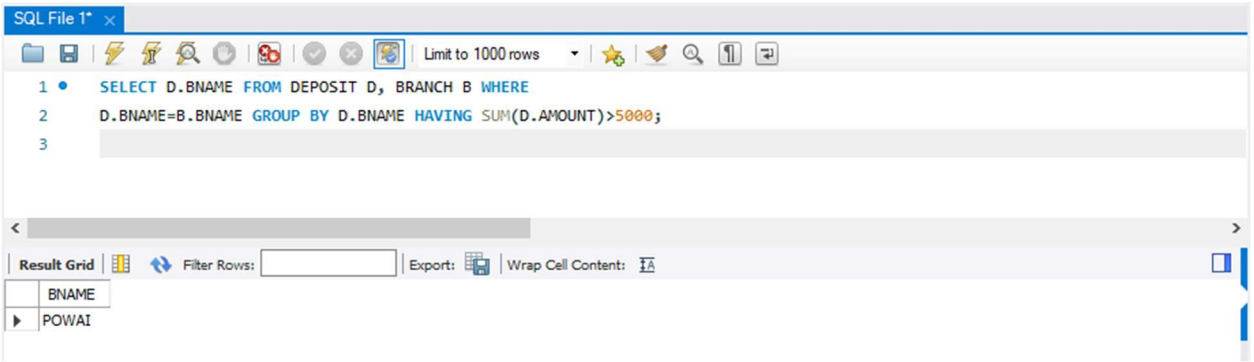
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

## Procedure

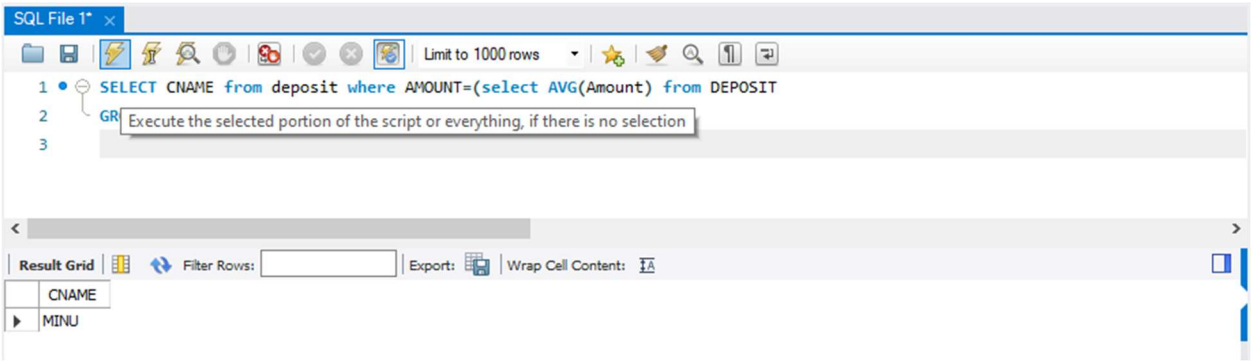
1. List the branches having sum of deposit more than 5000.



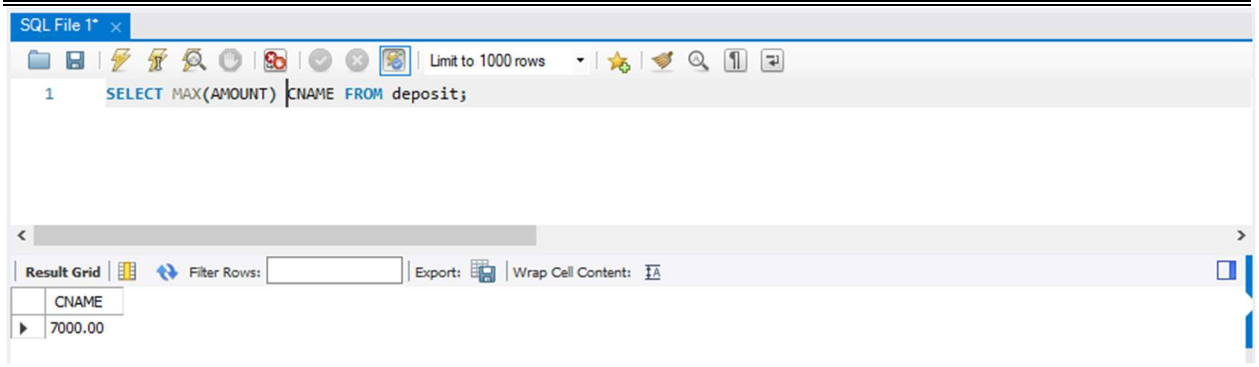

2. List the branches having sum of deposit more than 500 and located in city BOMBAY



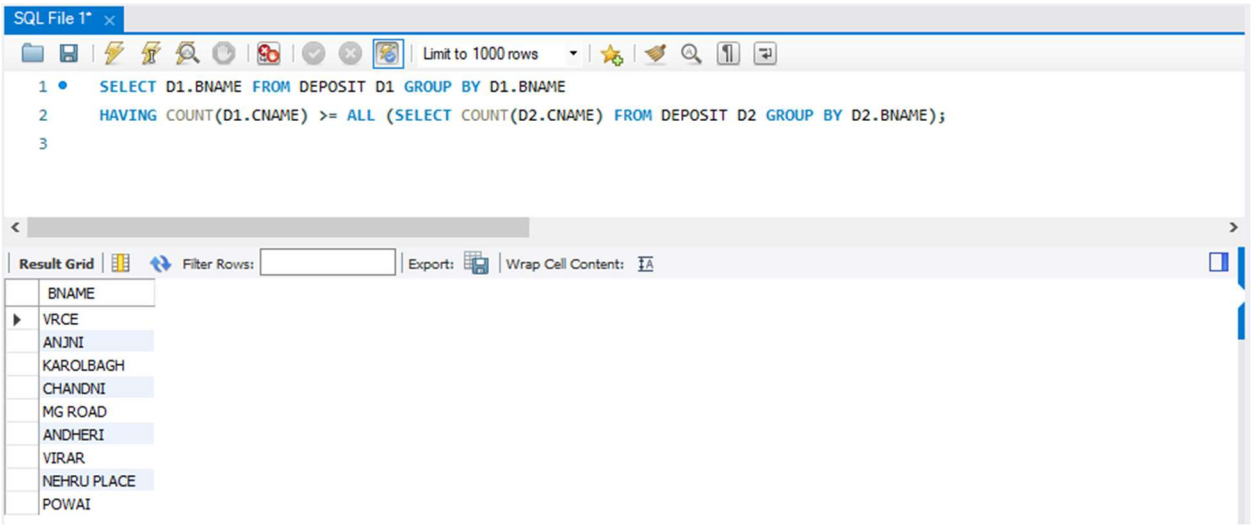

3. List the names of customers having deposited in the branches where the average deposit is more than 5000.



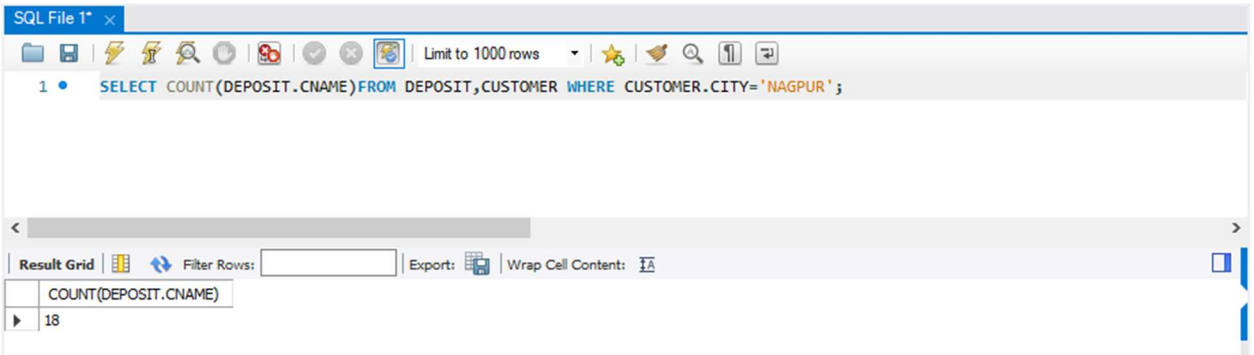

4. List the names of customers having maximum deposit



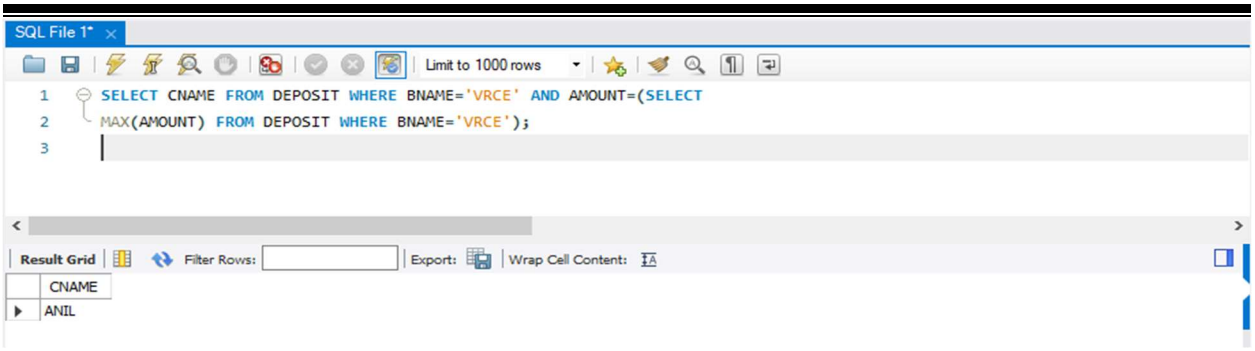

5. List the name of branch having highest number of depositors?



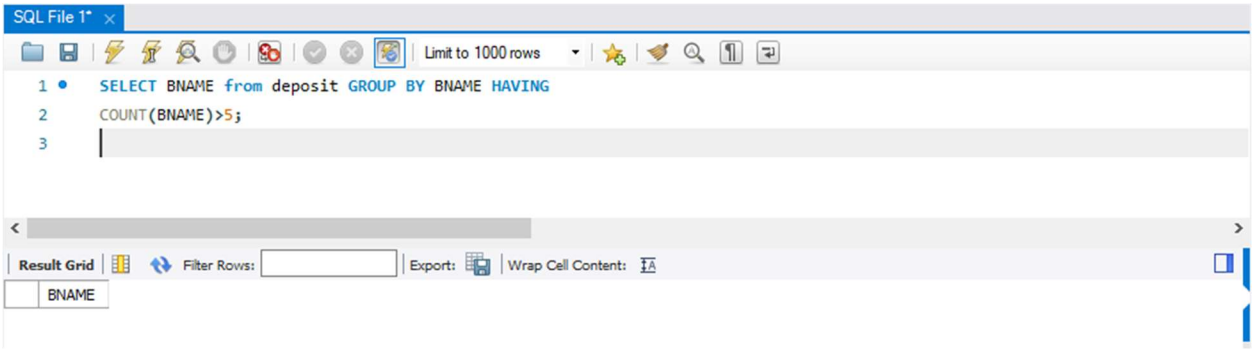

6. Count the number of depositors living in NAGPUR.



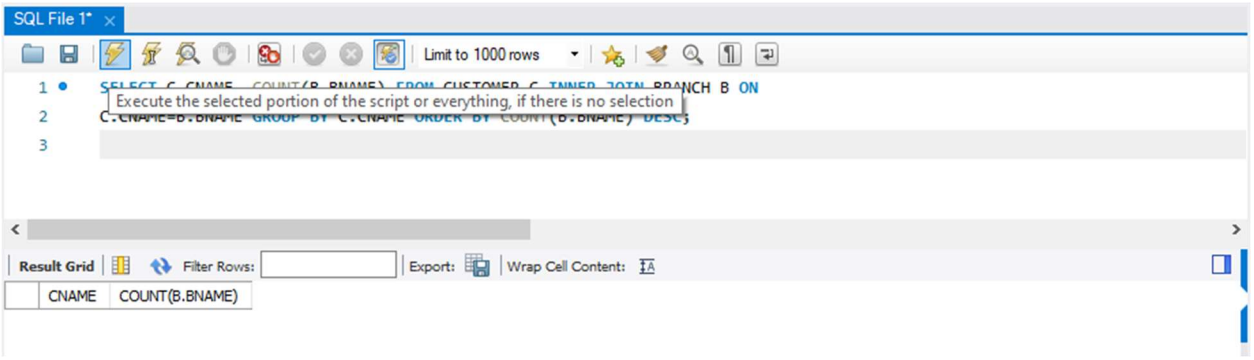

7. Give names of customers in VRCE branch having more deposite than any other customer in same branch



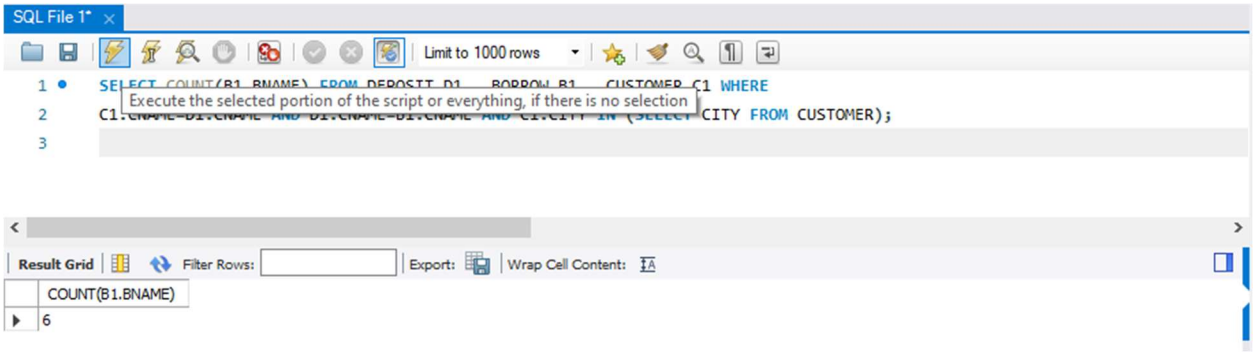

8. Give the names of branch where number of depositors is more than 5




9. Give the names of cities in which the maximum number of branches are located




10. Count the number of customers living in the city where branch is located




## Result

The program was executed and the result was successfully obtained. Thus CO1 was obtained

## Experiment No: 8

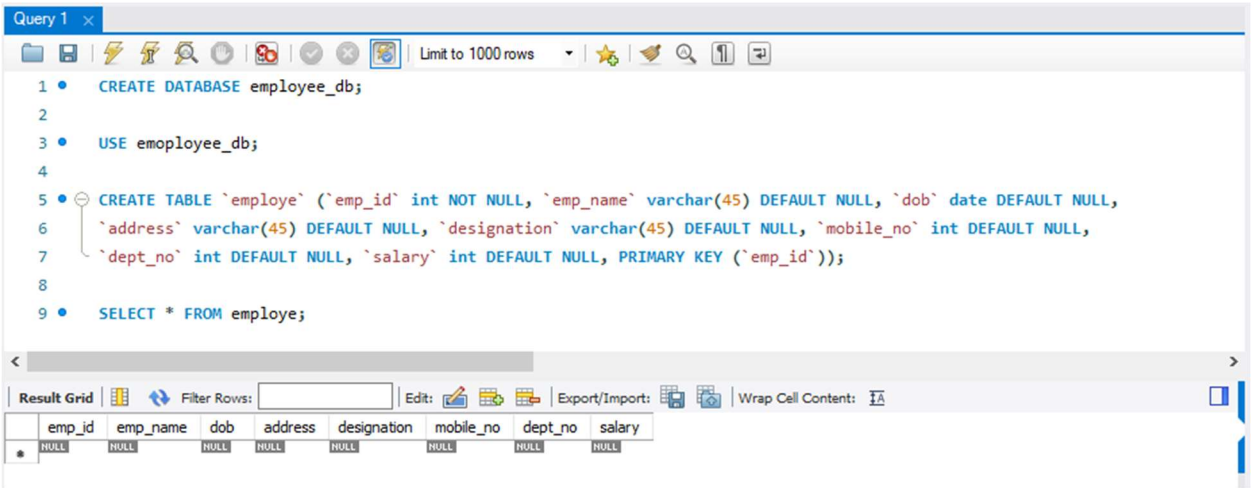
### Aim
To familiarize with trigger functions

### CO2
Apply PL/SQL for processing databases

### Procedure

Step 1: Start
Step 2: Initialize the trigger.
Step 3: On update the trigger has to be executed.
Step 4: Execute the trigger procedure after updation
Step 5: Carry out the operation on the table to check for trigger execution.
Step 6: Stop

Create a Trigger for employee table it will update another table salary while updating values



Query 1

```
CREATE TABLE `salary` ( `employee_id` int NOT NULL, `old_sal` int DEFAULT NULL, `new_sal` int DEFAULT NULL,
`rev_date` date DEFAULT NULL, PRIMARY KEY (`employee_id`) );

SELECT * FROM salary;
```

Result Grid

| employee_id | old_sal | new_sal | rev_date |
|---|---|---|---|
| NULL | NULL | NULL | NULL |

employe

```
INSERT INTO `employee_db`.`employe`
(`emp_id`, `emp_name`, `dob`, `address`, `designation`, `mobile_no`, `dept_no`, `salary`)
VALUES ('1', 'amal', '1995-04-29', 'wayanad', 'developer', '9469664422', '7', '40000');

SELECT * FROM employe;
```

Result Grid

| emp_id | emp_name | dob | address | designation | mobile_no | dept_no | salary |
|---|---|---|---|---|---|---|---|
| 1 | amal | 199... | wayanad | developer | 9469664... | 7 | 40000 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Table Name: employe Schema: **employee_db**

Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci Engine: InnoDB

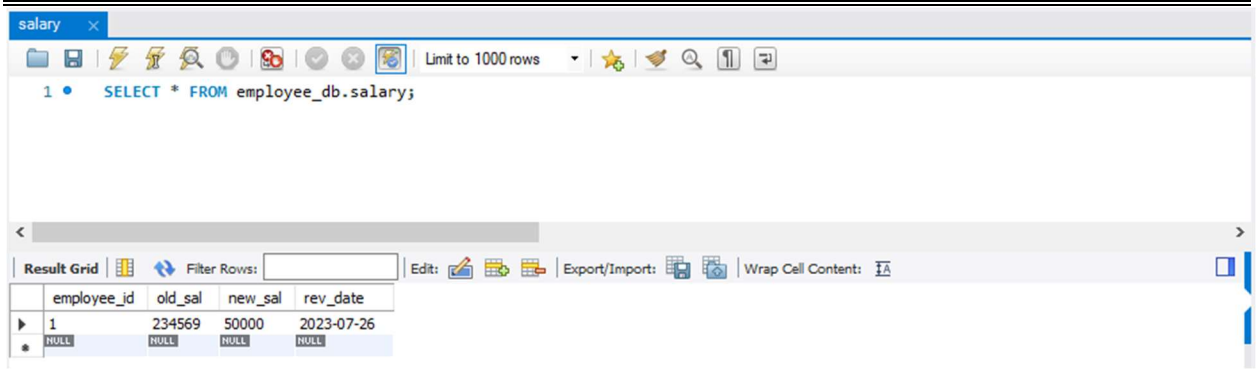
Comments:

- BEFORE INSERT
- AFTER INSERT
- BEFORE UPDATE
- AFTER UPDATE
  - employe_AFTER_UPDATE
- BEFORE DELETE
- AFTER DELETE

```
CREATE DEFINER=`root`@`localhost`
TRIGGER `employe_AFTER_UPDATE` AFTER UPDATE ON `employe` FOR EACH ROW BEGIN
if(new.salary != old.salary)
then
INSERT INTO salary (employee_id,old_sal,new_sal,rev_date) values
(new.emp_id,old.salary,new.salary,sysdate());
END if;
END
```

salary

```
UPDATE employe SET salary=50000 WHERE emp_id=1;
```




**Result**

The program was executed and the result was successfully obtained. Thus CO2 was obtained

**Experiment No: 9**

**Aim**

To familiarize with Stored functions and Procedure

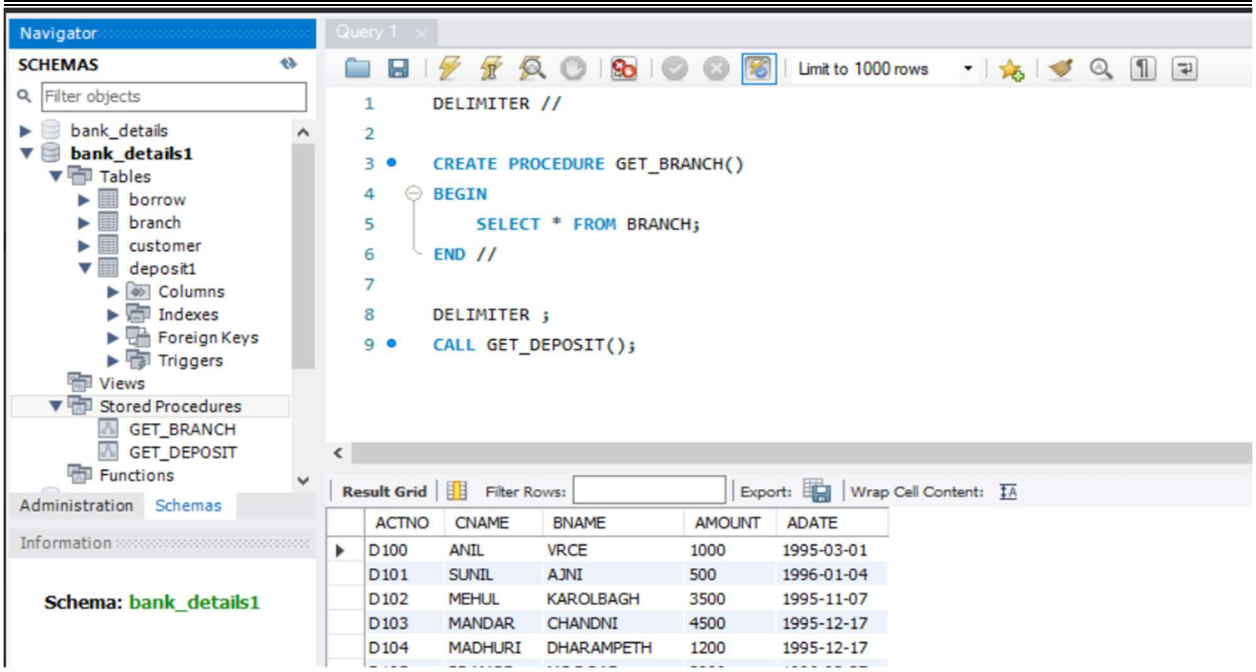
**CO2**

Apply PL/SQL for processing databases

**Procedure**

```
DELIMITER //
CREATE PROCEDURE GET_BRANCH()
BEGIN
   SELECT *  FROM BORROW;
END//

DELIMITER ;
CALL GET_BRANCH ()
```




**Result**

The program was executed and the result was successfully obtained. Thus CO2 was obtained

**Experiment No: 10**

**Aim**

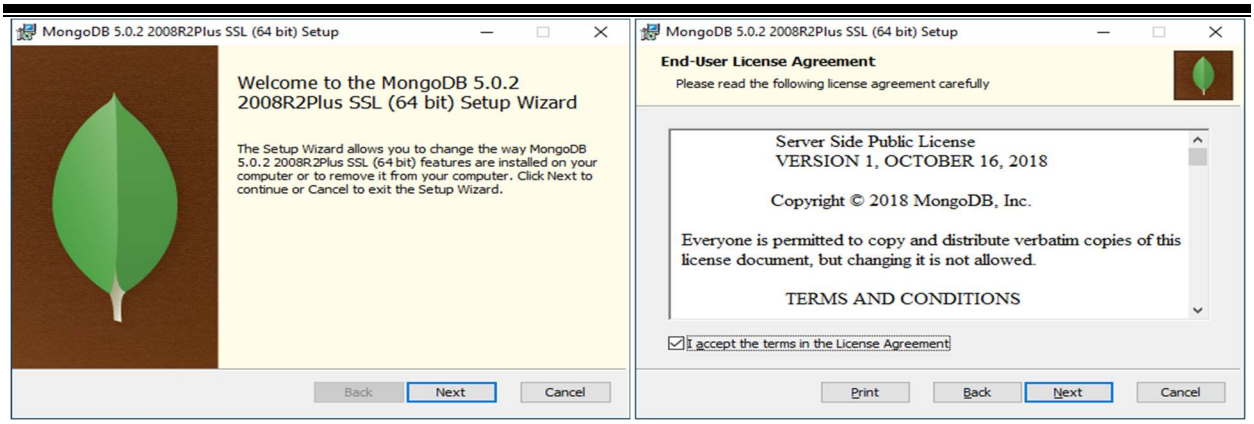
Understand the installation and configuration of NoSQL Databases: MongoDB

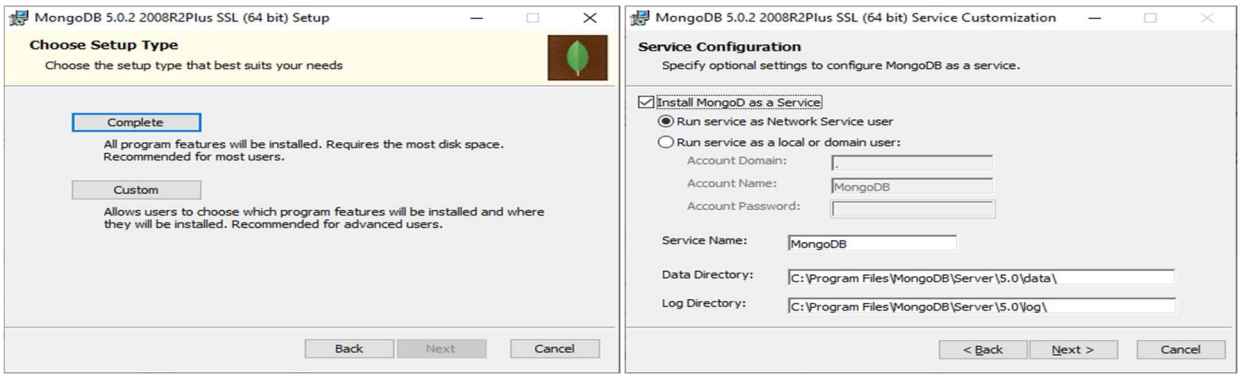
**CO3**

Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.

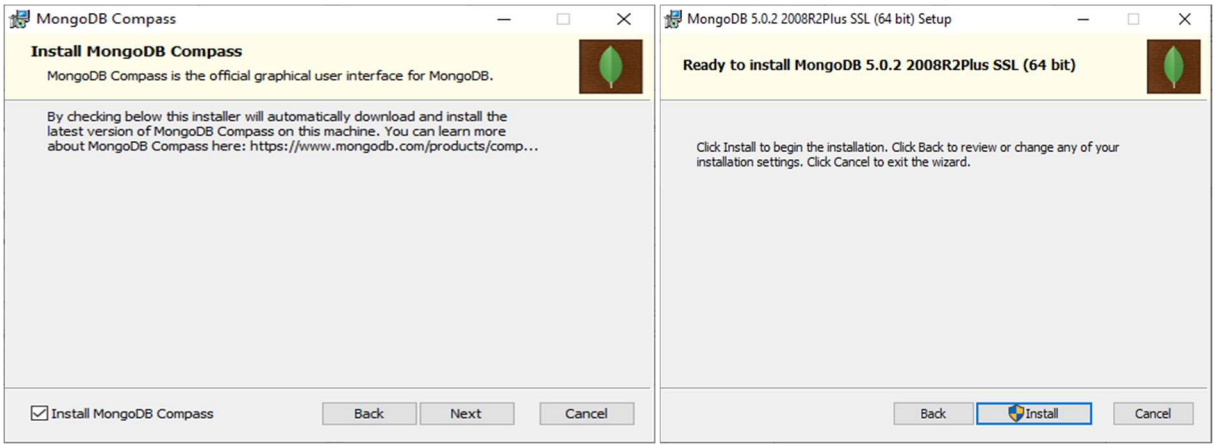
**Procedure**

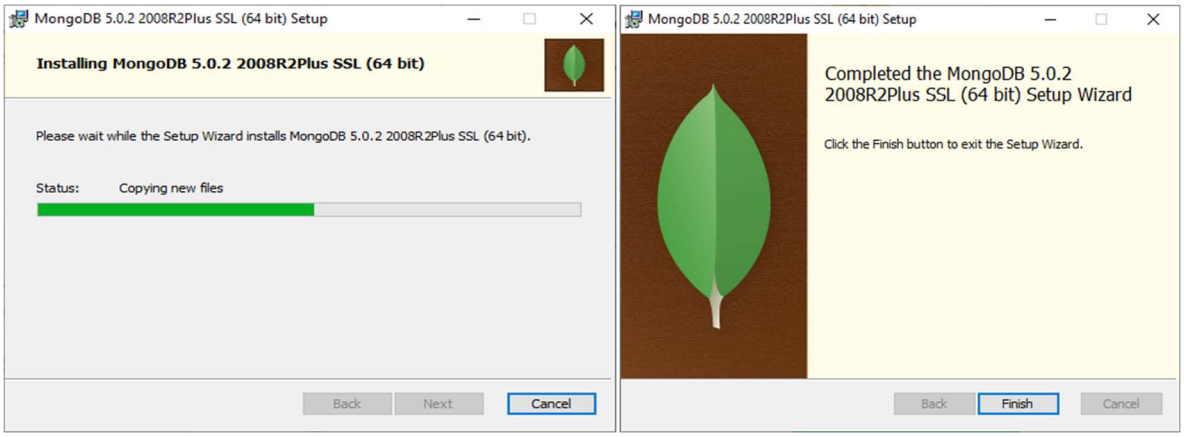
1. Setup Wizard



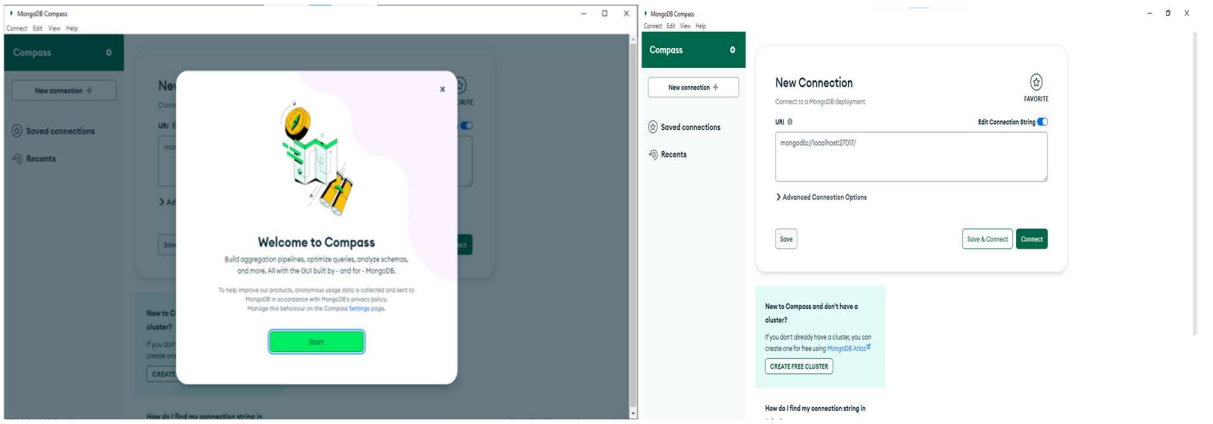



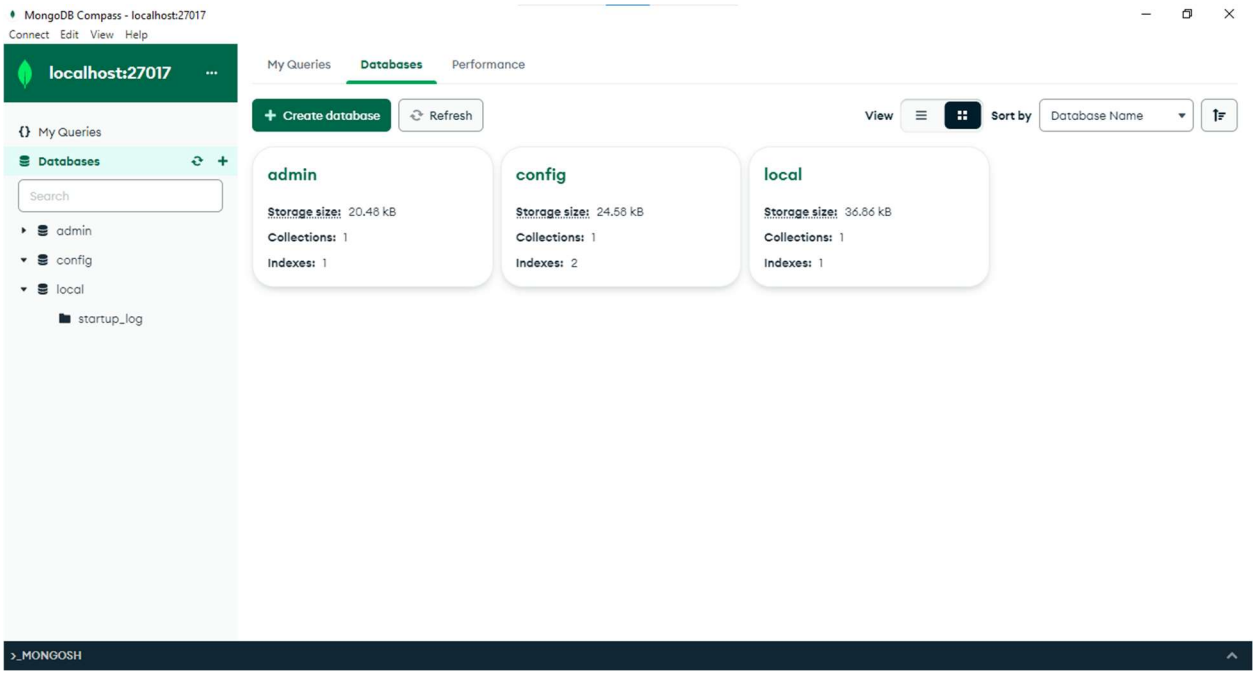

2. Installing MongoDB Compass



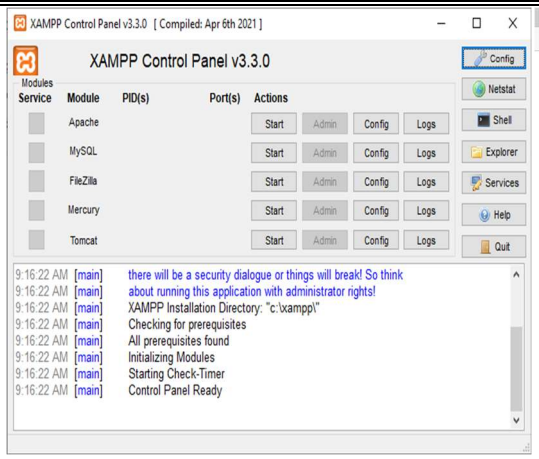



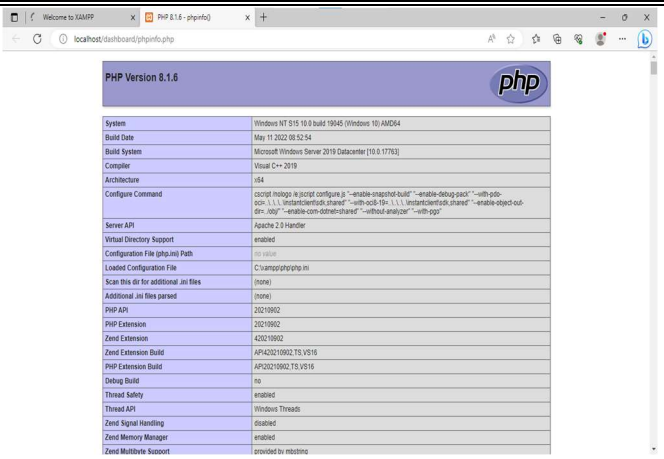
3. MongoDB Compass Home Page

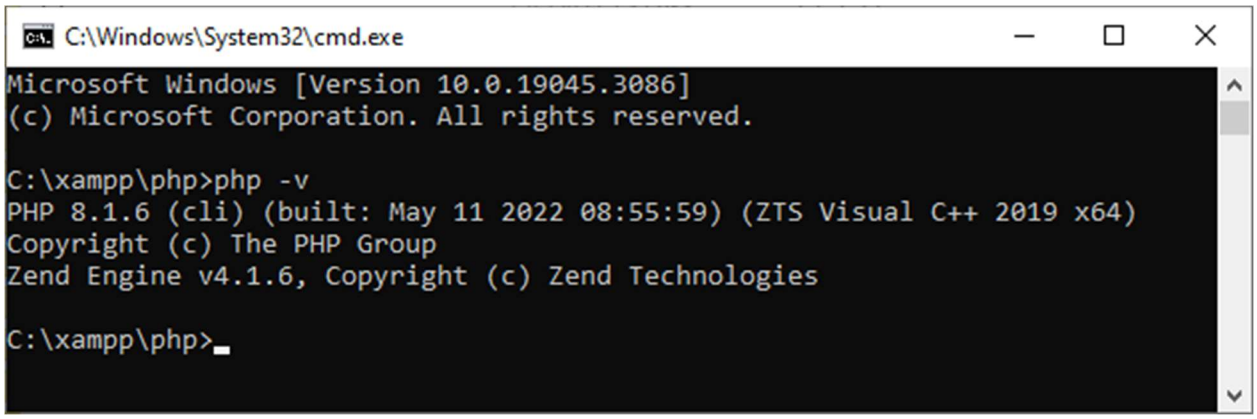


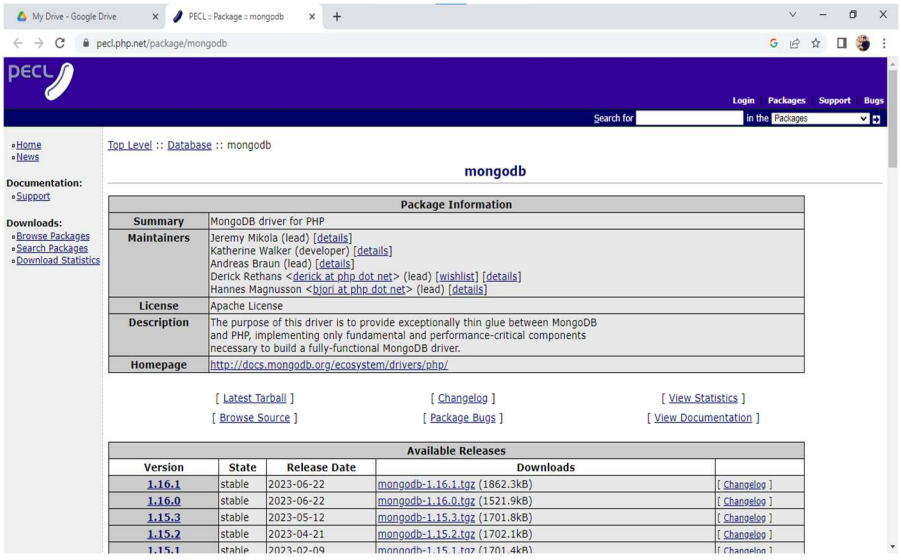

4. Configuring MongoDB to Connect to PHP



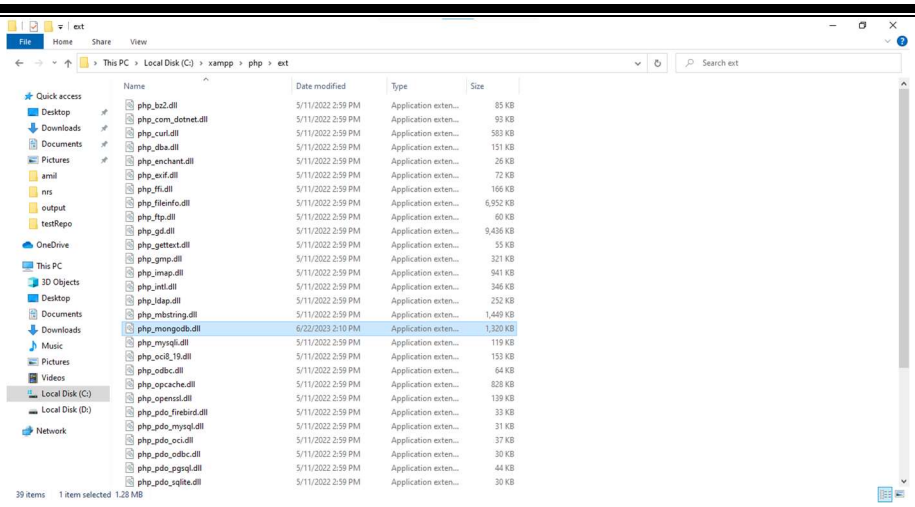



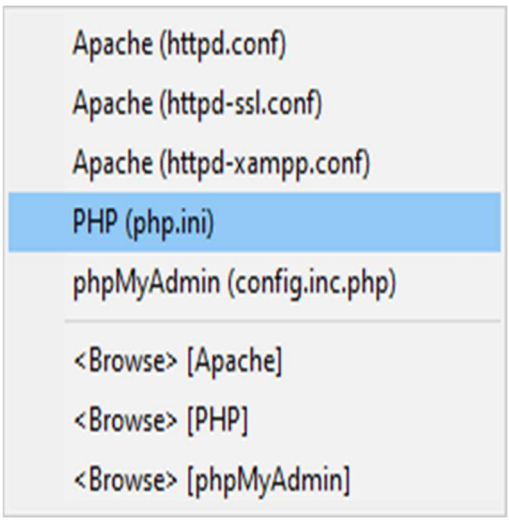



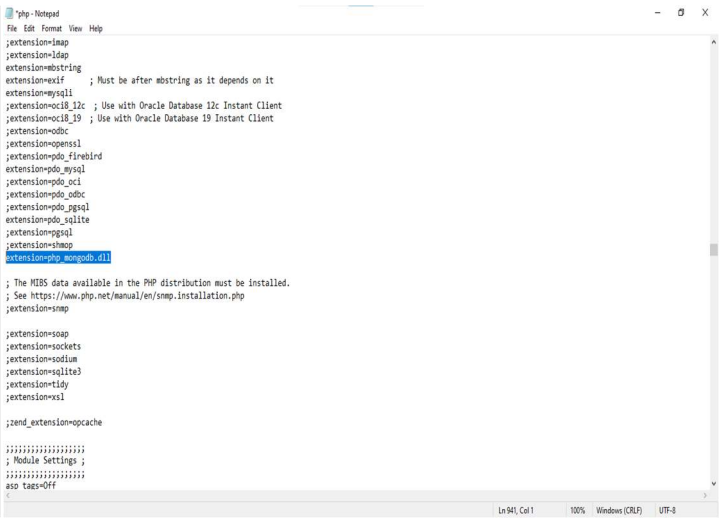

5. Installing MongoDB Package for PHP

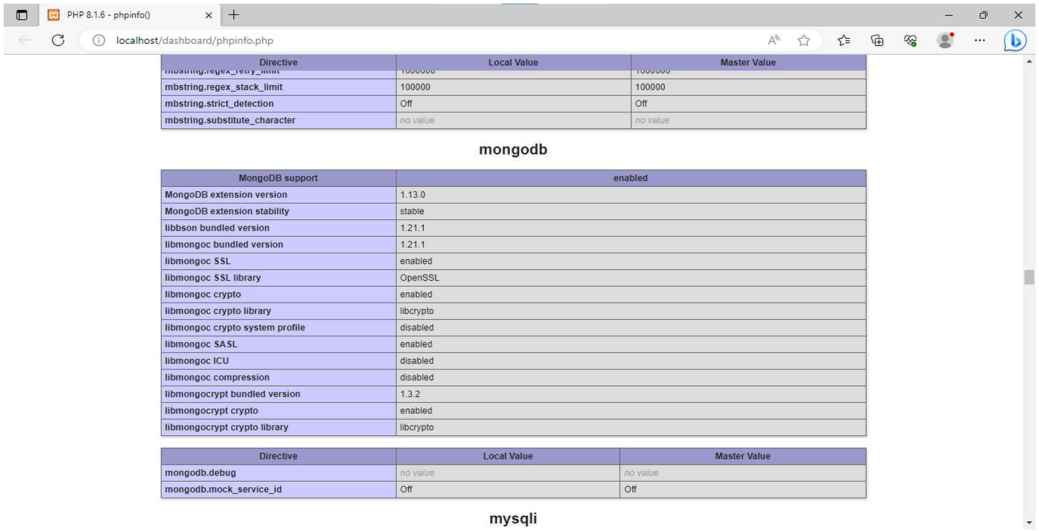


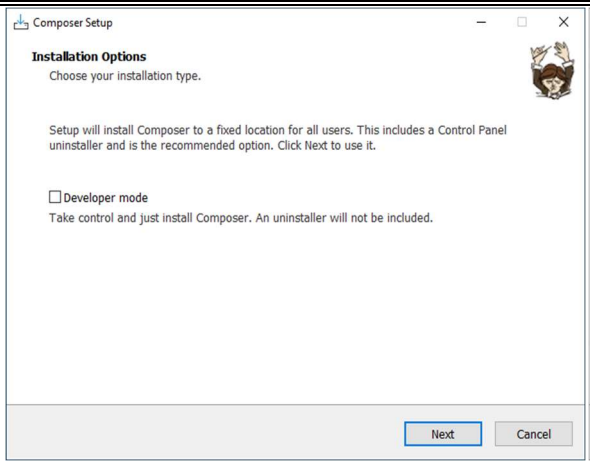

6. Configuring Apache

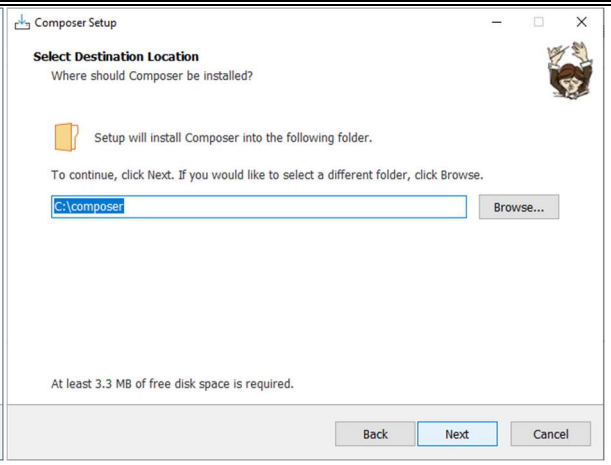

ext
File Home Share View
This PC > Local Disk (C:) > xampp > php > ext
Search ext
Quick access
Desktop
Downloads
Documents
Pictures
amil
nrs
output
testRepo
OneDrive
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)
Network
Name
Date modified
Type
Size
php_bz2.dll 5/11/2022 2:59 PM Application exten... 85 KB
php_com_dotnet.dll 5/11/2022 2:59 PM Application exten... 93 KB
php_curl.dll 5/11/2022 2:59 PM Application exten... 583 KB
php_dba.dll 5/11/2022 2:59 PM Application exten... 151 KB
php_enchant.dll 5/11/2022 2:59 PM Application exten... 26 KB
php_exif.dll 5/11/2022 2:59 PM Application exten... 72 KB
php_ffi.dll 5/11/2022 2:59 PM Application exten... 166 KB
php_fileinfo.dll 5/11/2022 2:59 PM Application exten... 6,952 KB
php_ftp.dll 5/11/2022 2:59 PM Application exten... 60 KB
php_gd.dll 5/11/2022 2:59 PM Application exten... 9,436 KB
php_gettext.dll 5/11/2022 2:59 PM Application exten... 55 KB
php_gmp.dll 5/11/2022 2:59 PM Application exten... 321 KB
php_imap.dll 5/11/2022 2:59 PM Application exten... 941 KB
php_intl.dll 5/11/2022 2:59 PM Application exten... 346 KB
php_ldap.dll 5/11/2022 2:59 PM Application exten... 252 KB
php_mbstring.dll 5/11/2022 2:59 PM Application exten... 1,449 KB
php_mongodb.dll 6/22/2023 2:10 PM Application exten... 1,320 KB
php_mysqli.dll 5/11/2022 2:59 PM Application exten... 119 KB
php_oci8_19.dll 5/11/2022 2:59 PM Application exten... 153 KB
php_odbc.dll 5/11/2022 2:59 PM Application exten... 64 KB
php_opcache.dll 5/11/2022 2:59 PM Application exten... 828 KB
php_openssl.dll 5/11/2022 2:59 PM Application exten... 139 KB
php_pdo_firebird.dll 5/11/2022 2:59 PM Application exten... 33 KB
php_pdo_mysql.dll 5/11/2022 2:59 PM Application exten... 31 KB
php_pdo_oci.dll 5/11/2022 2:59 PM Application exten... 37 KB
php_pdo_odbc.dll 5/11/2022 2:59 PM Application exten... 30 KB
php_pdo_pgsql.dll 5/11/2022 2:59 PM Application exten... 44 KB
php_pdo_sqlite.dll 5/11/2022 2:59 PM Application exten... 30 KB
39 items 1 item selected 1.28 MB

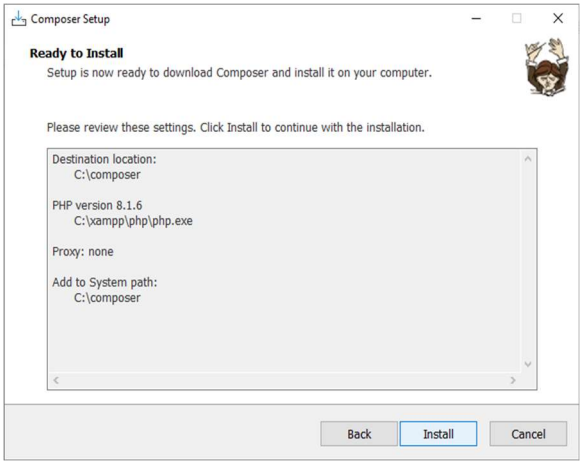

Apache (httpd.conf)
Apache (httpd-ssl.conf)
Apache (httpd-xampp.conf)
PHP (php.ini)
phpMyAdmin (config.inc.php)
<Browse> [Apache]
<Browse> [PHP]
<Browse> [phpMyAdmin]

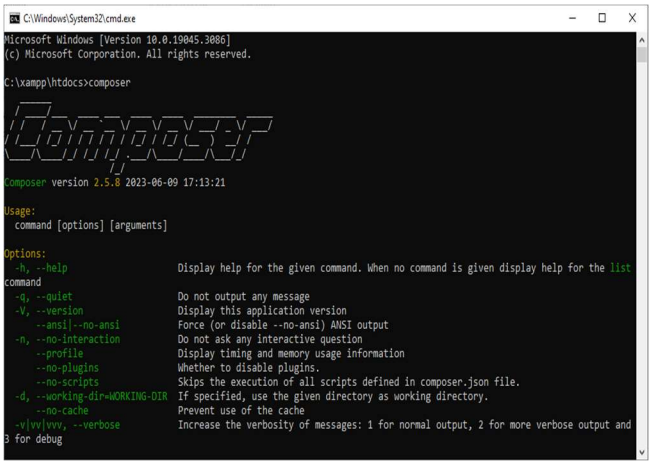

*php - Notepad
File Edit Format View Help
;extension=imap
;extension=ldap
extension=mbstring
extension=exif ; Must be after mbstring as it depends on it
extension=mysqli
;extension=oci8_12c ; Use with Oracle Database 12c Instant Client
;extension=oci8_19 ; Use with Oracle Database 19 Instant Client
;extension=odbc
;extension=openssl
;extension=pdo_firebird
extension=pdo_mysql
;extension=pdo_oci
;extension=pdo_odbc
;extension=pdo_pgsql
extension=pdo_sqlite
;extension=pgsql
;extension=shmop
extension=php_mongodb.dll
; The MIBS data available in the PHP distribution must be installed.
; See https://www.php.net/manual/en/snmp.installation.php
;extension=snmp
;extension=soap
;extension=sockets
;extension=sodium
;extension=sqlite3
;extension=tidy
;extension=xsl
;zend_extension=opcache
; Module Settings ;
asp_tags=Off
Ln 941, Col 1
100%
Windows (CRLF)
UTF-8

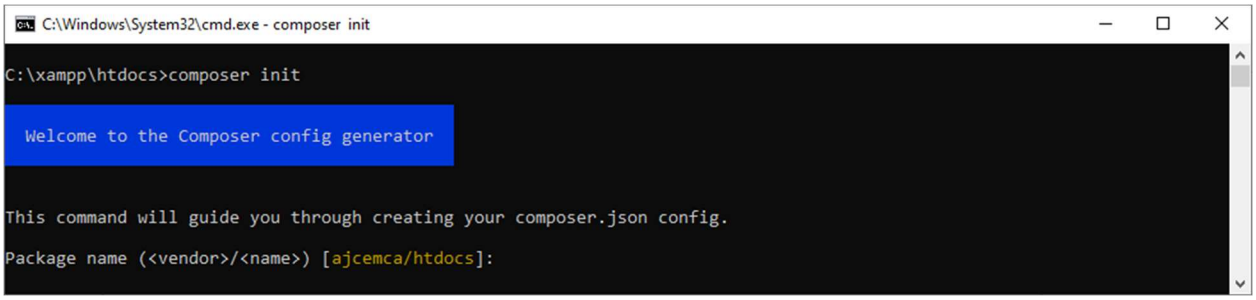

PHP 8.1.6 - phpinfo()
localhost/dashboard/phpinfo.php
Directive
Local Value
Master Value
mbstring.regex_stack_limit
100000
100000
mbstring.strict_detection
Off
Off
mbstring.substitute_character
no value
no value
mongodb
MongoDB support
enabled
MongoDB extension version
1.13.0
MongoDB extension stability
stable
libbson bundled version
1.21.1
libmongoc bundled version
1.21.1
libmongoc SSL
enabled
libmongoc SSL library
OpenSSL
libmongoc crypto
enabled
libmongoc crypto library
libcrypto
libmongoc crypto system profile
disabled
libmongoc SASL
enabled
libmongoc ICU
disabled
libmongoc compression
disabled
libmongocrypt bundled version
1.3.2
libmongocrypt crypto
enabled
libmongocrypt crypto library
libcrypto
Directive
Local Value
Master Value
mongodb.debug
no value
no value
mongodb.mock_service_id
Off
Off
mysqli

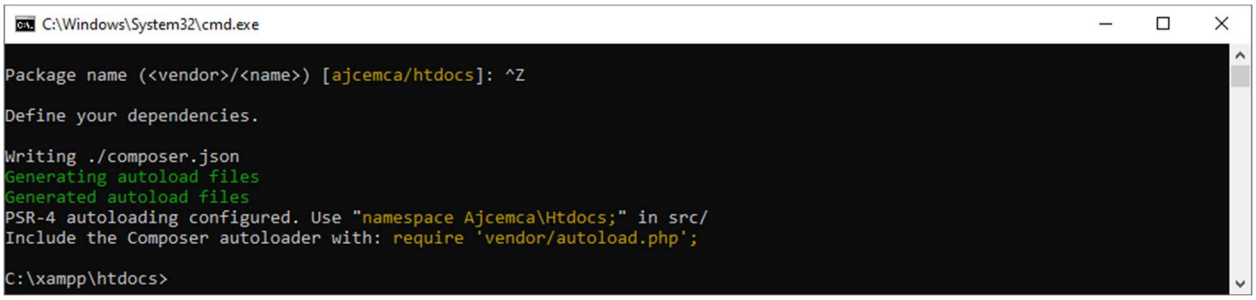

Composer Setup
Installation Options
Choose your installation type.
Setup will install Composer to a fixed location for all users. This includes a Control Panel uninstaller and is the recommended option. Click Next to use it.
Developer mode
Take control and just install Composer. An uninstaller will not be included.
Next
Cancel

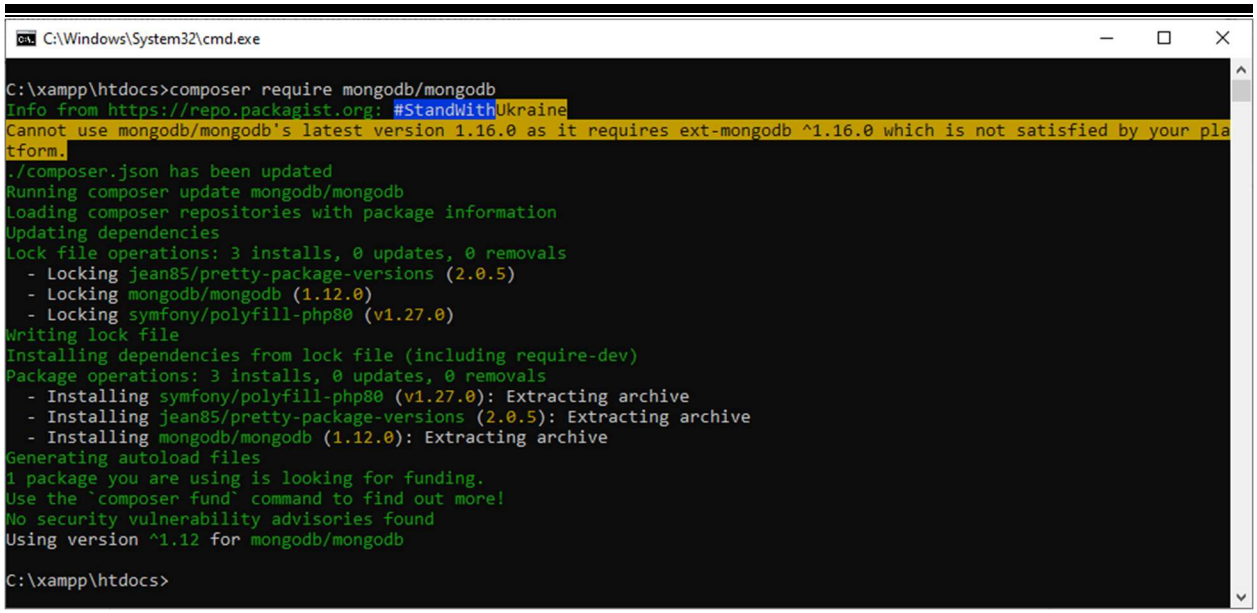

Composer Setup
Select Destination Location
Where should Composer be installed?
Setup will install Composer into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\composer
Browse...
At least 3.3 MB of free disk space is required.
Back
Next
Cancel


Composer Setup
Ready to Install
Setup is now ready to download Composer and install it on your computer.
Please review these settings. Click Install to continue with the installation.
Destination location:
C:\composer
PHP version 8.1.6
C:\xampp\php\php.exe
Proxy: none
Add to System path:
C:\composer
Back
Install
Cancel


C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19045.3086]
(c) Microsoft Corporation. All rights reserved.
C:\xampp\htdocs>composer
Composer version 2.5.8 2023-06-09 17:13:21
Usage:
command [options] [arguments]
Options:
-h, --help Display help for the given command. When no command is given display help for the list command
-q, --quiet Do not output any message
-V, --version Display this application version
--ansi|--no-ansi Force (or disable --no-ansi) ANSI output
-n, --no-interaction Do not ask any interactive question
--profile Display timing and memory usage information
--no-plugins Whether to disable plugins.
--no-scripts Skips the execution of all scripts defined in composer.json file.
-d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
--no-cache Prevent use of the cache
-v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug


C:\Windows\System32\cmd.exe - composer init
C:\xampp\htdocs>composer init
Welcome to the Composer config generator
This command will guide you through creating your composer.json config.
Package name (<vendor>/<name>) [ajcemca/htdocs]:


C:\Windows\System32\cmd.exe
Package name (<vendor>/<name>) [ajcemca/htdocs]: ^Z
Define your dependencies.
Writing ./composer.json
Generating autoload files
Generated autoload files
PSR-4 autoloading configured. Use "namespace Ajcemca\Htdocs;" in src/
Include the Composer autoloader with: require 'vendor/autoload.php';
C:\xampp\htdocs>




**Result**

The program was executed and the result was successfully obtained. Thus CO3 was obtained

**Experiment No: 11**

**Aim**

Create a database and use the insertMany method to insert the colors of the rainbow into the database

**CO4**

Apply CRUD operations and retrieve data in a NoSQL environment.

**Procedure**

```
<?php
require '../vendor/autoload.php';
$conn = new MongoDB\Client("mongodb://localhost:27017");
echo "Connection Successful.";
echo "<br>";
$db = $conn -> Rainbow;
echo " Rainbow database created.";
echo "<br>";
$collection = $db -> createCollection("Colors");
echo "Collection Colors created";
echo "<br>";
$collection = $db -> Colors;
echo "<br>";
$c1 = array('color' => 'Violet');
$c2 = array('color' => 'Indigo');
$c3 = array('color' => 'Blue');
$c4 = array('color' => 'Green');
$c5 = array('color' => 'Yellow');
$c6 = array('color' => 'Orange');
$c7 = array('color' => 'Red');
```

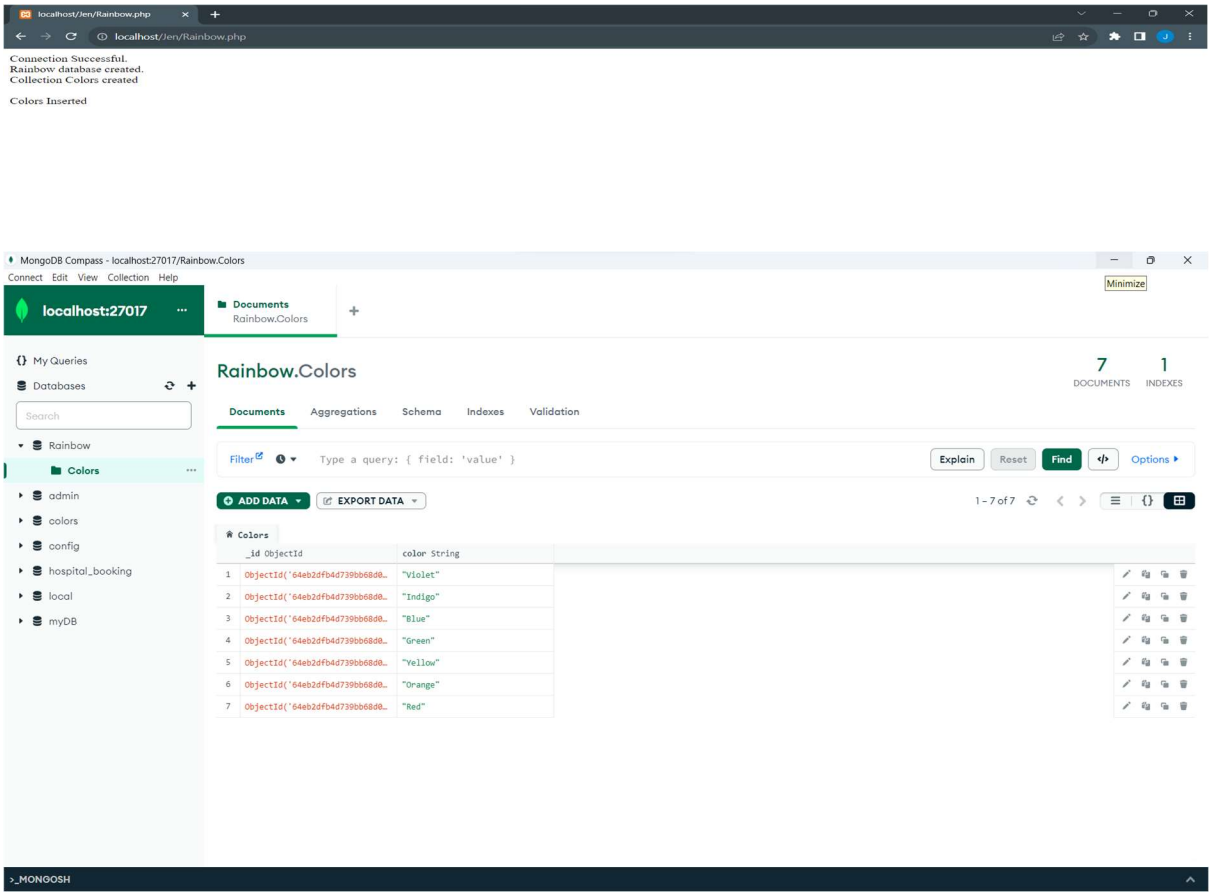
```
$collection -> insertMany([$c1, $c2, $c3, $c4, $c5, $c6, $c7]);

echo "Colors Inserted ";

echo "<br>";

?>
```

**Output**




**Result**

The program was executed and the result was successfully obtained. Thus CO4 was obtained

**Experiment No: 12**

**Aim**
Implementing CRUD operations using PHP-MongoDB

**CO4**
Apply CRUD operations and retrieve data in a NoSQL environment.

**Procedure**
**connection.php**

```
<?php
require '../vendor/autoload.php';
$con=new MongoDB\Client("mongodb://localhost:27017");
//echo $con;
$db=$con->admin;
$collection=$db->Profile;
?>
```

**register.php**

```
<html>
<head><title>Registration form</title>
</head>
```

```
<body><table border="1" align="center">
<tr><td>
<h1>Registrarion Form</h1>
<form action="insert.php" method="post">
Name:<input type="text" name="name" required><br><br>
Email:<input type="text" name="email" required><br><br>
Place :<input type="text" name="place" required><br><br>
<input type="submit" name="submit" class="button" value="Register" >
</form>
</td><tr>
</table>
</body>
</html>
```

**edit.php**

```
<?php
include_once("connection.php");
if(isset($_POST['update'])){
echo $id = $_GET['id'];
$name = $_POST['name'];
$place = $_POST['place'];
$email = $_POST['email'];
$db->Profile->updateOne(['_id' => new MongoDB\BSON\ObjectID($id)],['$set' => ['name'=>
$name, 'place' => $place, 'email'=>$email,]]);
header("Location: select.php");
}?>
<html><head>
<title>Registration Form</title>
</head>
<?php
$id = $_GET['id'];
```

```
$result = $db->Profile->findOne(['_id' => new MongoDB\BSON\ObjectID($id)]);
$name = $result['name'];
$place = $result['place'];
$email = $result['email'];
?>
<body>
<h2>Profile Data</h2>
<form action="#" method="post">
<?php $id = $_GET['id'];?>
Name: <input type="text" name="name" value="<?php echo $name; ?>" required><br>
Place:<input type="place" name="place" value="<?php echo $place; ?>" required><br>
Email:<input type="email" name="email" value="<?php echo $email; ?>" required><br>
<input type="submit" name="update" class="button" value="Update">
</form></body>
</html>
```

**select.php**

```
<center><br><form action=register.php>
<input type="submit" value=register></form>
<form action=select.php>
<input type="submit" value=View>
</form><br>
<?php
include_once("connection.php");
$result=$collection->find();
?>
<html>
<head><title>select.php</title>
</head><body>
<table border="0" >
```

```
<tr>
<th align="center" bgcolor=green><font color=white size=4><b>SlNo.</th>
<th align="center" bgcolor=green><font color=white size=4><b>Name</th>
<th align="center" bgcolor=green><font color=white size=4><b>Email</th>
<th align="center" bgcolor=green><font color=white size=4><b>Place</th>
<th align="center" bgcolor=green><font color=white size=4><b>Delete</th>
<th align="center" bgcolor=green><font color=white size=4><b>Edit</th>
</tr><?php
$no=1;
foreach($result as $res){
?> <tr>
<td><font color=DeepSkyBlue size=4><?php echo $no;?>
<td><font color=steelblue size=4><?php echo $res['name'];?></td>
<td><font color=pink size=4><?php echo $res['email'];?></td>
<td><font color=RosyBrown size=4><?php echo $res['place'];?></td>
<td><a href="<?php echo "delete.php?id=$res[_id]";?>" onClick="return confirm('Are you sure you want to delete?')">Delete</a></td>
<td><a href="<?php echo "edit.php?id=$res[_id]";?>">Edit</a></td>
<?php
$no++;
} ?>
</tr></table></body></html>
```

**insert.php**

```
<?php
include_once("connection.php");
if(isset($_POST["submit"])){
        $user=array(
        'name'=>$_POST['name'],
        'email'=>$_POST['email'],
        'place'=>$_POST['place']
```

```
        );
        $collection->insertOne($user);
        echo "Inserted";
}
header ("location:select.php");
?>
```

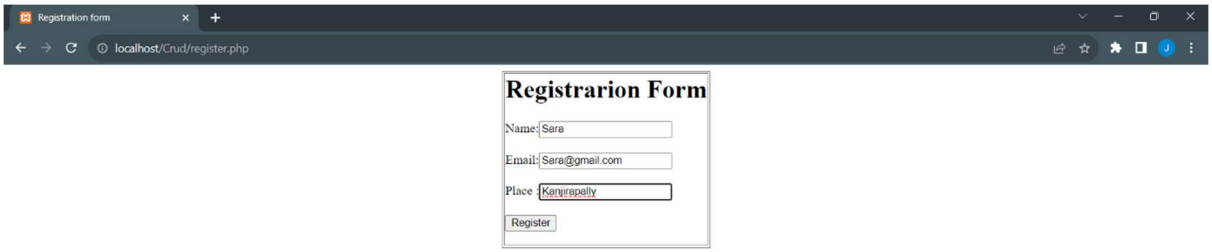
**delete.php**

```
<?php
include_once("connection.php");
$id = $_GET['id'];
$collection->deleteOne(['_id' => new MongoDB\BSON\ObjectID($id)]);
header("location:select.php");
?>
```

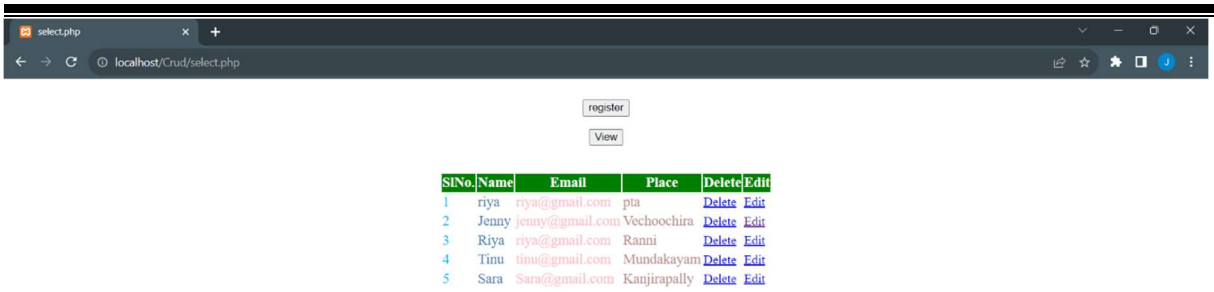
**Output Screenshot**

**Connection.php**

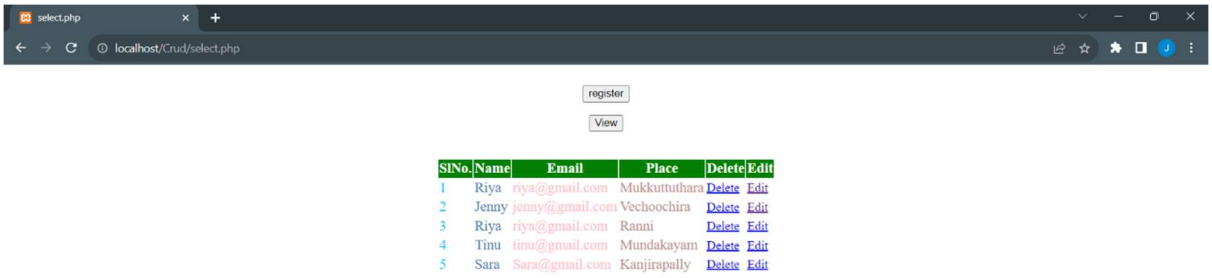

**Register.php**



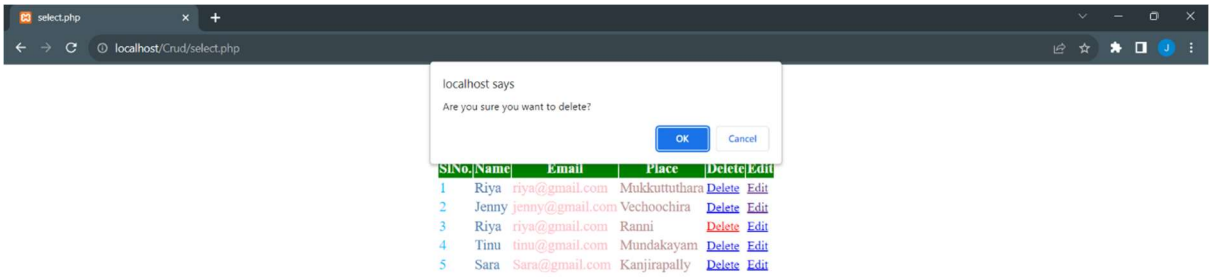

**Select.php**

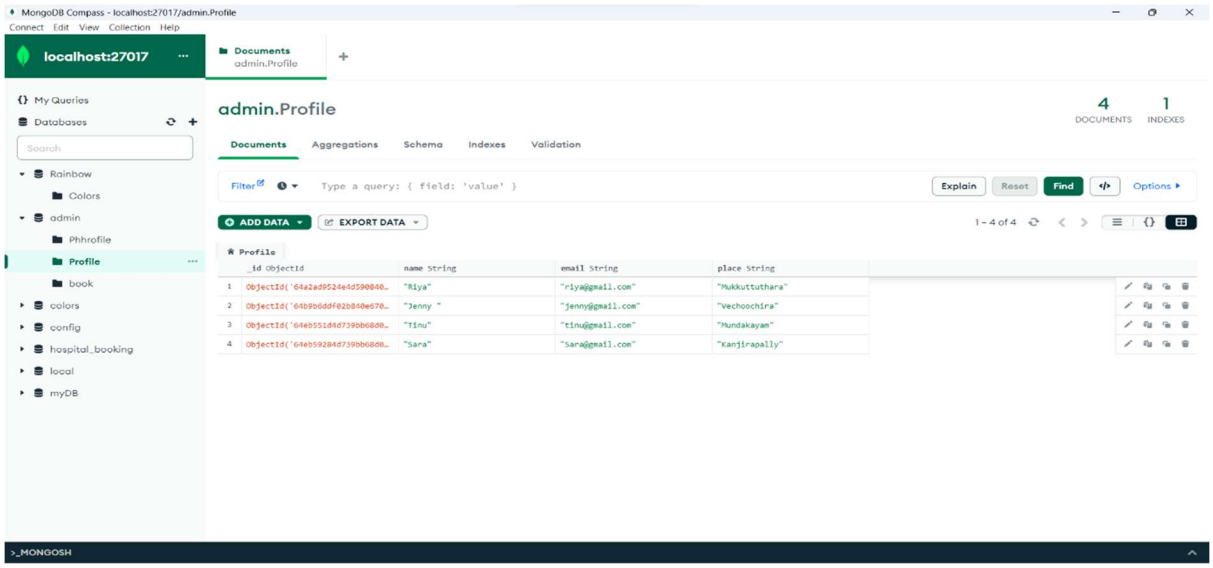



## Edit.php


## Delete.php




## Result

The program was executed and the result was successfully obtained. Thus CO4 was obtained

**Experiment No: 13**

**Aim**

Build sample collections/documents to perform the shell queries

**CO5**

Understand the basic storage architecture of distributed file systems.

**Procedure**

1. Create/Use a database
   > use expmongo


2. Display current database
   >db


3. Create a collection
   >db.createCollection("actors")


4. Insert data into the collection

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.insertMany([
... { _id: "trojan", name: "Ivan Trojan", year: 1964, movies: [ "samotari", "medvidek" ] },
... { _id: "machacek", name: "Jiri Machacek", year: 1966, movies: [ "medvidek", "vratnelahve", "samotari" ] },
... { _id: "schneiderova", name: "Jitka Schneiderova",  year: 1973, movies: [ "samotari" ] },
... { _id: "sverak", name: "Zdenek Sverak", year: 1936, movies: [ "vratnelahve" ] },
... { _id: "geislerova", name: "Anna Geislerova",  year: 1976 }
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                "trojan",
                "machacek",
                "schneiderova",
                "sverak",
                "geislerova"
        ]
}
>
```

5. Display documents in collection
   >db.actors.find()

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find()
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936, "movies" : [ "vratnelahve" ] }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
>
```

6. Display documents in collection
   >db.actors.find({ })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936, "movies" : [ "vratnelahve" ] }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
>
```

7. Display
   >db.actors.find({ _id: "trojan" })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ _id: "trojan" })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
>
```

8. Display
   >db.actors.find({ name: "Ivan Trojan", year: 1964 })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ name: "Ivan Trojan", year: 1964 })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
>
```

9. Display
   >db.actors.find({ year: { $gte: 1960, $lte: 1980 } })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ year: { $gte: 1960, $lte: 1980 } })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
>
```

10. Display

>db.actors.find({ movies: { $exists: true } })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ movies: { $exists: true } })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936, "movies" : [ "vratnelahve" ] }
>
```

11. Display

>db.actors.find({ movies: "medvidek" })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ movies: "medvidek" })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
>
```

12. Display

>db.actors.find({ movies: { $in: [ "medvidek", "pelisky" ] } })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ movies: { $in: [ "medvidek", "pelisky" ] } })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
>
```

13. Display

>db.actors.find({ movies: { $all: [ "medvidek", "pelisky" ] } })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ movies: { $all: [ "medvidek", "pelisky" ] } })
>
```

14. Display

>db.actors.find({ $or: [ { year: 1964 }, { rating: { $gte: 3 } } ] })

```
Select C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ $or: [ { year: 1964 }, { rating: { $gte: 3 } } ] })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
>
```

15. Display

>db.actors.find({ rating: { $not: { $gte: 3 } } })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ rating: { $not: { $gte: 3 } } })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936, "movies" : [ "vratnelahve" ] }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
>
```

16. Display
    >db.actors.find({ }, { name: 1, year: 1 })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ }, { name: 1, year: 1 })
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964 }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966 }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973 }
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936 }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
>
```

17. Display
    >db.actors.find({ }, { movies: 0, _id: 0 })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ }, { movies: 0, _id: 0 })
{ "name" : "Ivan Trojan", "year" : 1964 }
{ "name" : "Jiri Machacek", "year" : 1966 }
{ "name" : "Jitka Schneiderova", "year" : 1973 }
{ "name" : "Zdenek Sverak", "year" : 1936 }
{ "name" : "Anna Geislerova", "year" : 1976 }
>
```

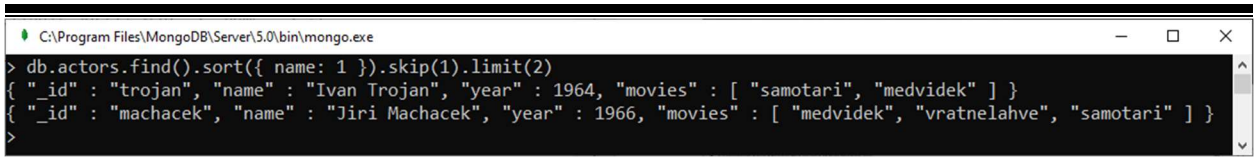
18. Display
    >db.actors.find({ }, { name: 1, movies: { $slice: 2 }, _id: 0 })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find({ }, { name: 1, movies: { $slice: 2 }, _id: 0 })
{ "name" : "Ivan Trojan", "movies" : [ "samotari", "medvidek" ] }
{ "name" : "Jiri Machacek", "movies" : [ "medvidek", "vratnelahve" ] }
{ "name" : "Jitka Schneiderova", "movies" : [ "samotari" ] }
{ "name" : "Zdenek Sverak", "movies" : [ "vratnelahve" ] }
{ "name" : "Anna Geislerova" }
>
```

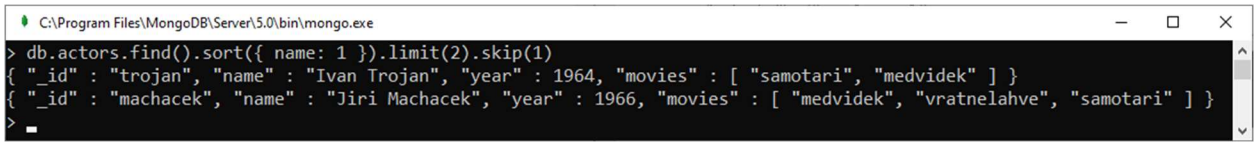
19. Display
    >db.actors.find().sort({ year: 1, name: -1 })

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find().sort({ year: 1, name: -1 })
{ "_id" : "sverak", "name" : "Zdenek Sverak", "year" : 1936, "movies" : [ "vratnelahve" ] }
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
{ "_id" : "schneiderova", "name" : "Jitka Schneiderova", "year" : 1973, "movies" : [ "samotari" ] }
{ "_id" : "geislerova", "name" : "Anna Geislerova", "year" : 1976 }
>
```

20. Display
    >db.actors.find().sort({ name: 1 }).skip(1).limit(2)




21. Display
>db.actors.find().sort({ name: 1 }).limit(2).skip(1)




**Result**

The program was executed and the result was successfully obtained. Thus CO5 was obtained

**Experiment No: 14**

**Aim**

To familiarize with indexing in MongoDB

**CO5**

Understand the basic storage architecture of distributed file systems.

**Procedure**

1.Input data –

db.products.insertMany( [

{ _id: 10, item: "large box", qty: 50 },

{ _id: 11, item: "medium box", qty: 30 },

{ _id: 12, item: "envelope", qty: 100},

{ _id: 13, item: "tape", qty: 20},

{ _id: 14, item: "bubble wrap", qty: 70}

])

```
> db.products.insertMany( [
... { _id: 10, item: "large box", qty: 50 },
... { _id: 11, item: "medium box", qty: 30 },
... { _id: 12, item: "envelope", qty: 100},
... { _id: 13, item: "tape", qty: 20},
... { _id: 14, item: "bubble wrap", qty: 70}
... ])
{ "acknowledged" : true, "insertedIds" : [ 10, 11, 12, 13, 14 ] }
```

```
> db.products.find()
{ "_id" : 10, "item" : "large box", "qty" : 50 }
{ "_id" : 11, "item" : "medium box", "qty" : 30 }
{ "_id" : 12, "item" : "envelope", "qty" : 100 }
{ "_id" : 13, "item" : "tape", "qty" : 20 }
{ "_id" : 14, "item" : "bubble wrap", "qty" : 70 }
```

Example 1 – Create ascending index on a field

db.collection.createIndex( { item: 1 } )

```
> db.collection.createIndex( { item: 1 } )
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : true,
        "ok" : 1
}
```

Example 2 – Create descending index on a field

db.collection.createIndex( { qty: -1 } )

```
> db.collection.createIndex( { qty: -1 } )
{
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "createdCollectionAutomatically" : false,
        "ok" : 1
}
```

```
> db.products.getIndexes()
[ { "v" : 2, "key" : { "_id" : 1 }, "name" : "_id_" } ]
```

2.Specify the name to the Index

Example – Create index with the index name
db.products.createIndex(

{ item: 1, quantity: -1 } ,

{ name: "query for inventory" }

)

```
> db.products.createIndex( { item: 1, quantity: -1 } , { name: "query for inventory" } )
{
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "createdCollectionAutomatically" : true,
        "ok" : 1
}
```

```
> db.products.getIndexes()
[
        {
                "v" : 2,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_"
        },
        {
                "v" : 2,
                "key" : {
                        "item" : 1,
                        "quantity" : -1
                },
                "name" : "query for inventory"
        }
]
```

3.dropIndex() Method Example

Example 1 – Drop index by index document

db.products.dropIndex(

{ item: 1, quantity: -1 }

)

```
> db.products.dropIndex(
... { item: 1, quantity: -1 }
... )
{ "nIndexesWas" : 2, "ok" : 1 }
```

Example 2 – Drop index by index name

db.products.dropIndex( "query for inventory“ )

```
> db.products.dropIndex( "query for inventory" )
{ "nIndexesWas" : 2, "ok" : 1 }
```

**Result**

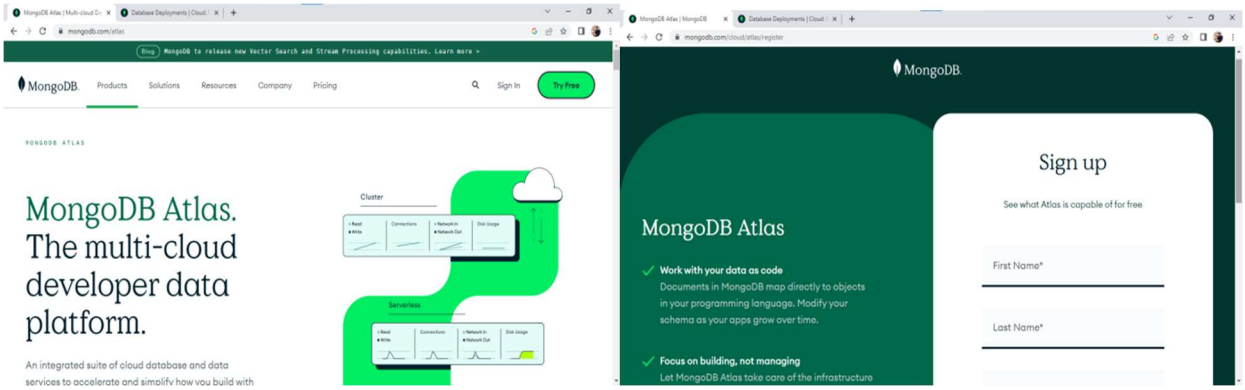
The program was executed and the result was successfully obtained. Thus CO5 was obtained

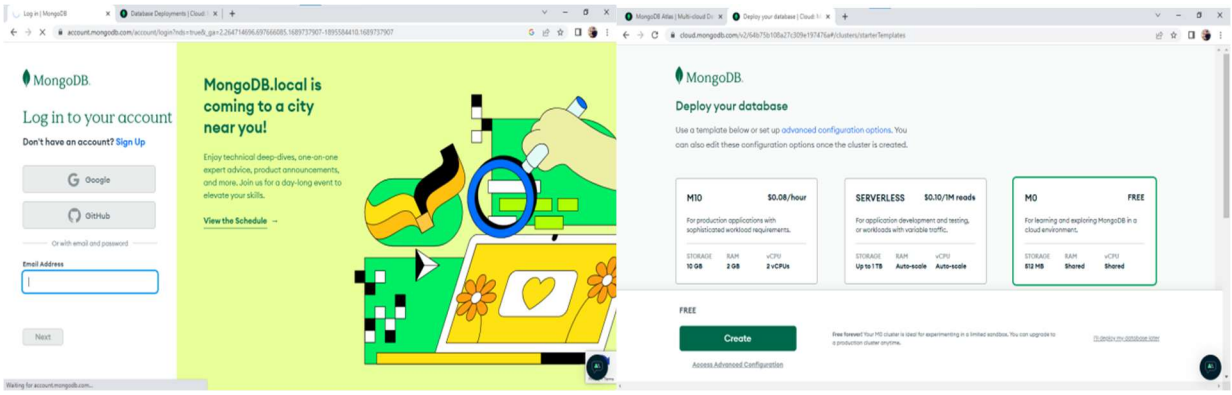
## Experiment No: 15

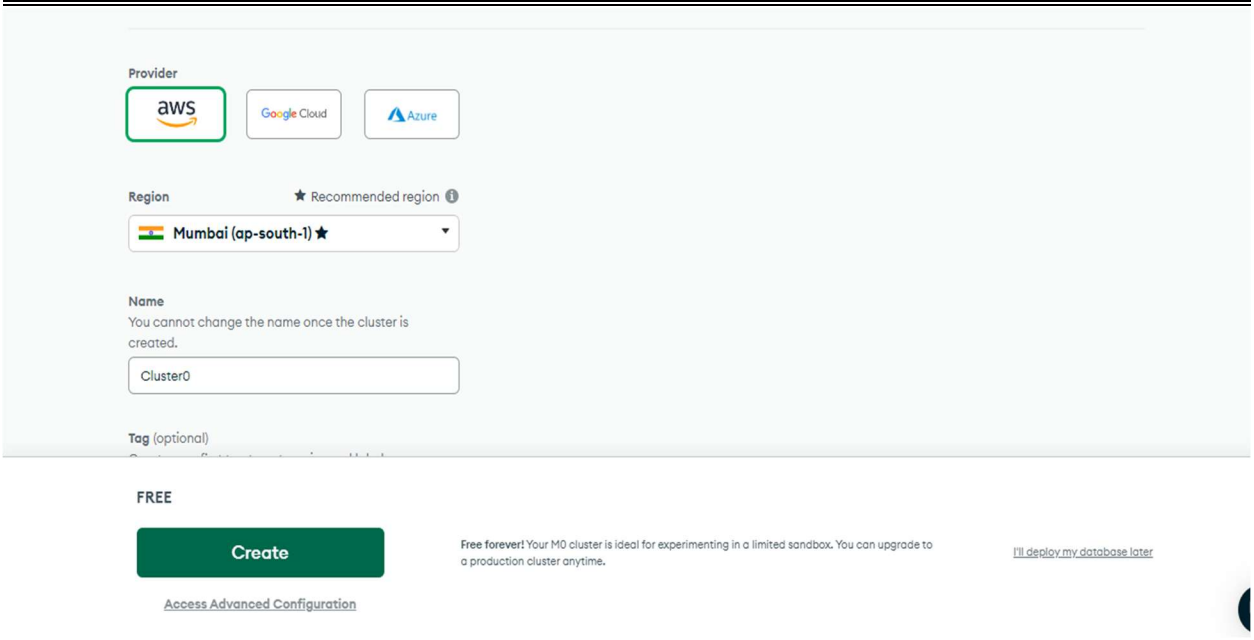
### Aim
Usage of Cloud Storage Management Systems: MongoDB Atlas

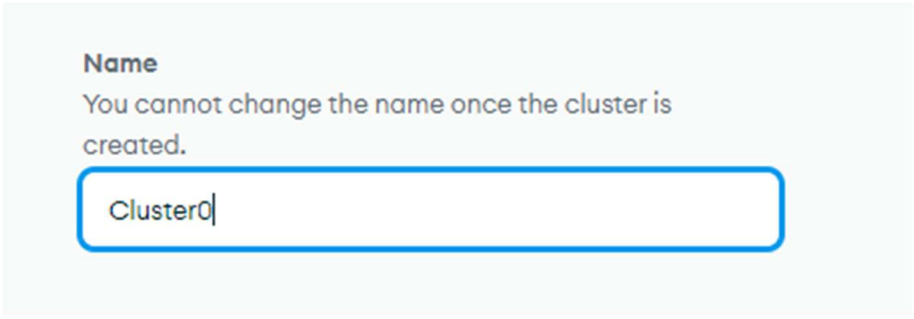
### CO6
Design and deployment of NoSQL databases with real time requirements.

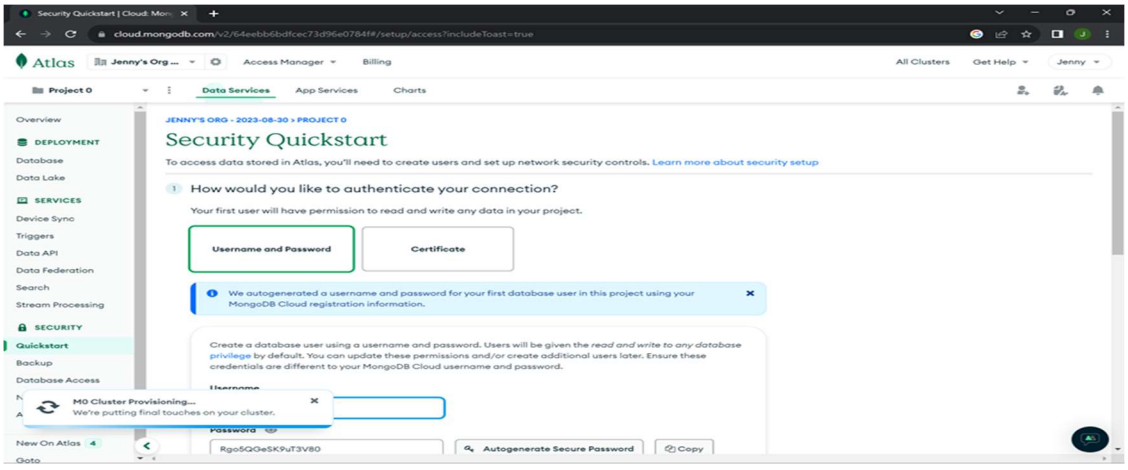
### Procedure

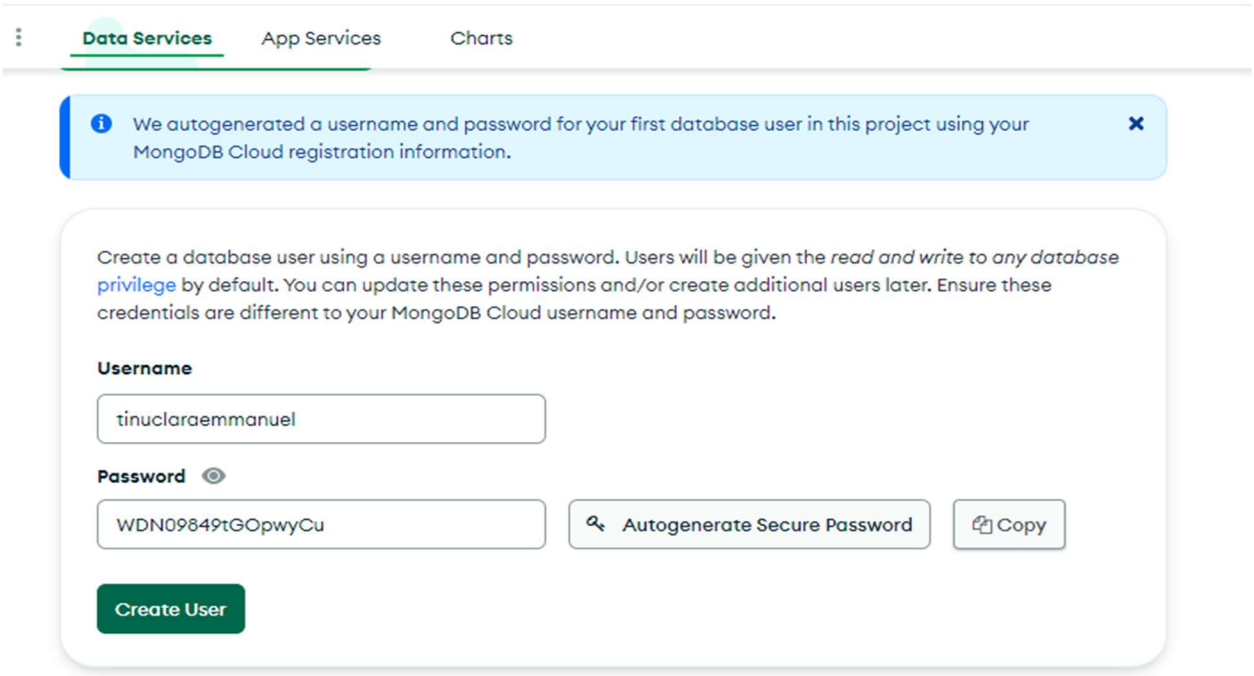


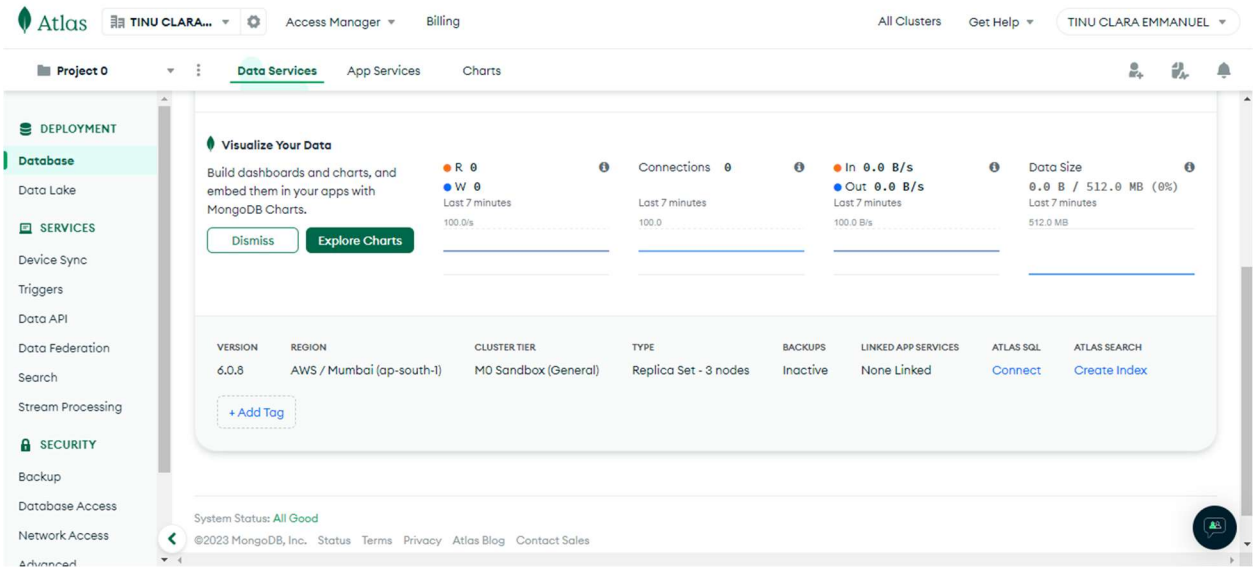



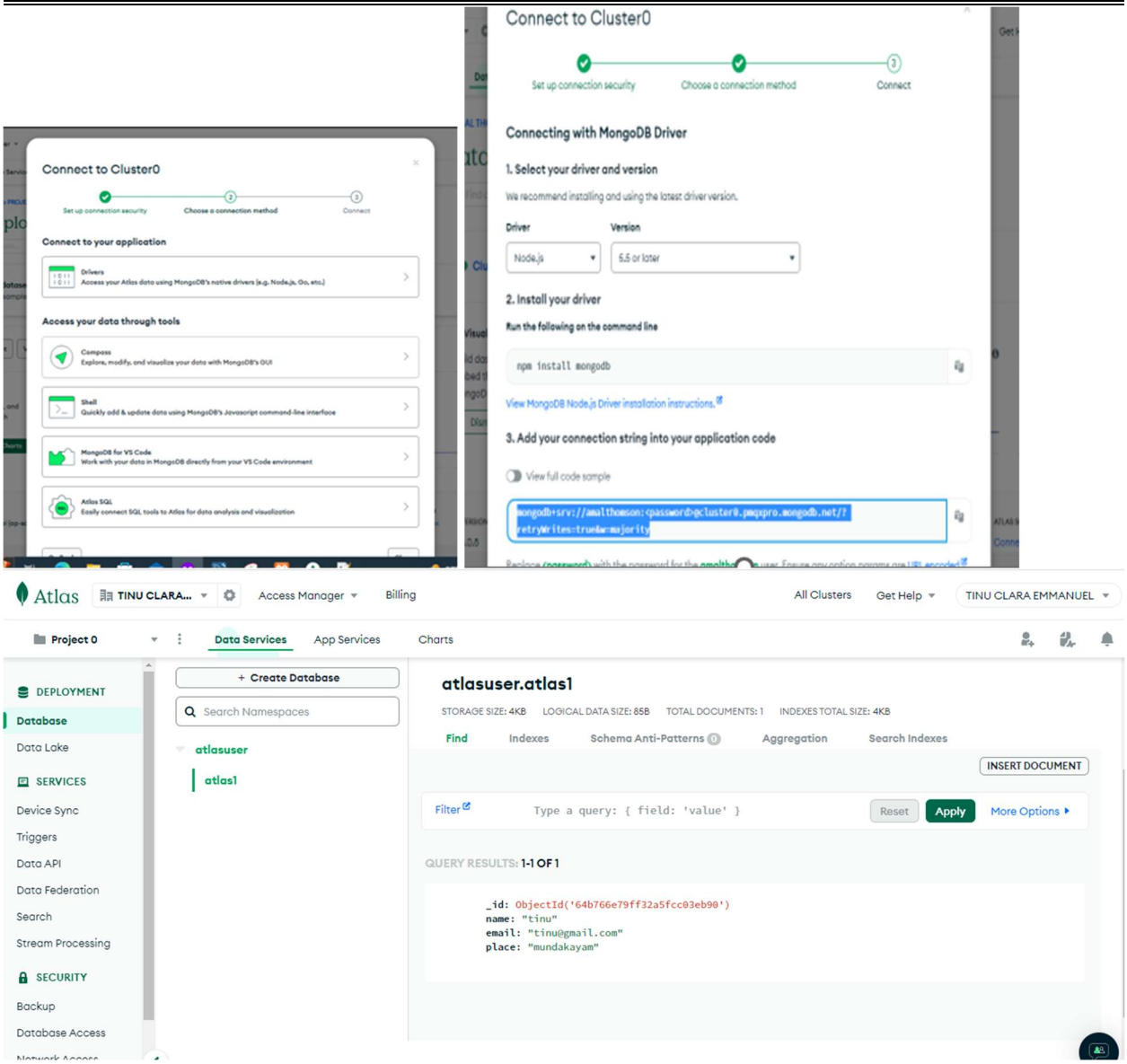

Provider
aws
Google Cloud
Azure
Region
Recommended region
Mumbai (ap-south-1)
Name
You cannot change the name once the cluster is created.
Cluster0
Tag (optional)
FREE
Create
Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.
I'll deploy my database later
Access Advanced Configuration


Name
You cannot change the name once the cluster is created.
Cluster0

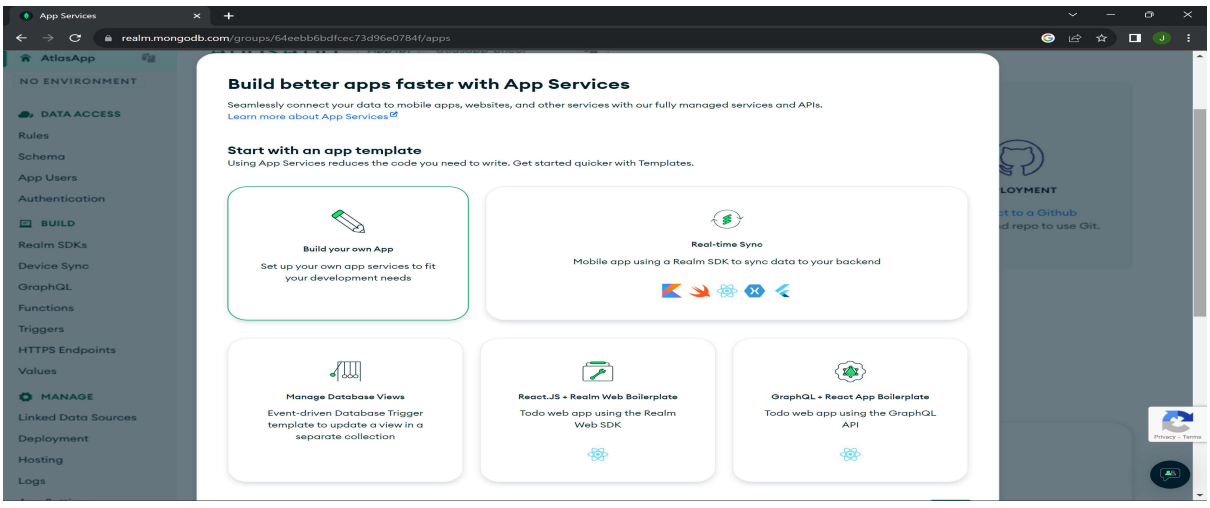

Atlas
Security Quickstart
To access data stored in Atlas, you'll need to create users and set up network security controls.
How would you like to authenticate your connection?
Your first user will have permission to read and write any data in your project.
Username and Password
Certificate
M0 Cluster Provisioning...
We're putting final touches on your cluster.
Autogenerate Secure Password
Copy

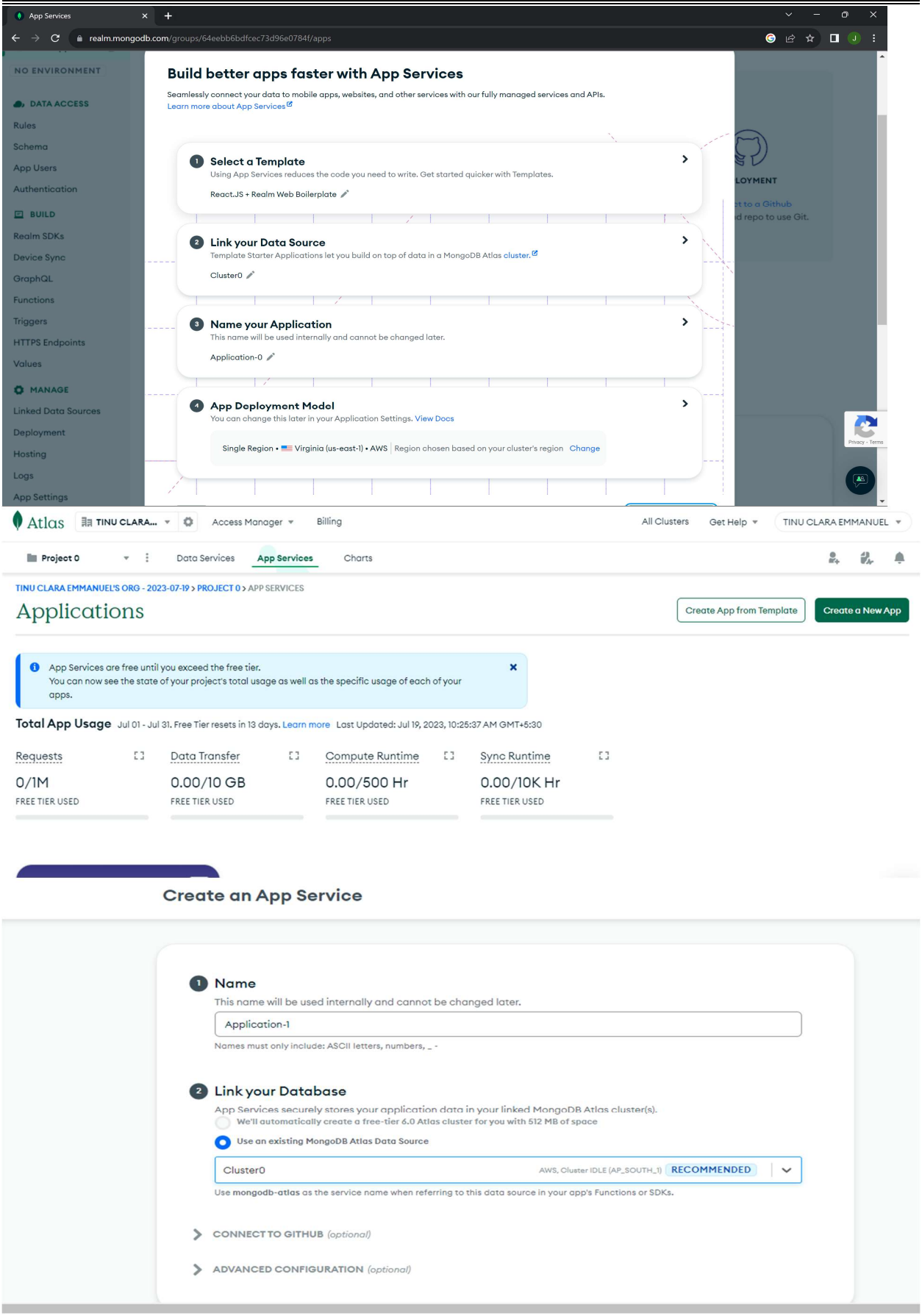

Data Services
App Services
Charts
We autogenerated a username and password for your first database user in this project using your MongoDB Cloud registration information.
Create a database user using a username and password. Users will be given the read and write to any database privilege by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.
Username
tinuclaraemmanuel
Password
WDN09849tGOpwyCu
Autogenerate Secure Password
Copy
Create User

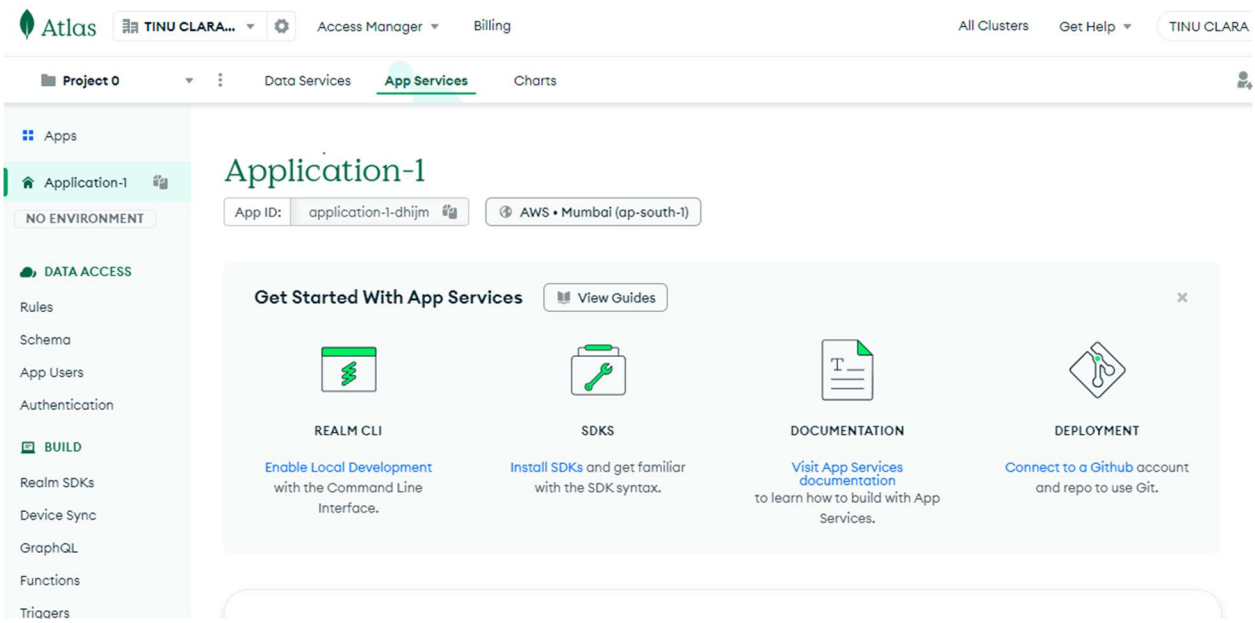

Atlas
TINU CLARA...
Access Manager
Billing
All Clusters
Get Help
TINU CLARA EMMANUEL
Project 0
Data Services
App Services
Charts
DEPLOYMENT
Database
Data Lake
SERVICES
Device Sync
Triggers
Data API
Data Federation
Search
Stream Processing
SECURITY
Backup
Database Access
Network Access
Visualize Your Data
Build dashboards and charts, and embed them in your apps with MongoDB Charts.
Dismiss
Explore Charts
R 0
W 0
Connections 0
In 0.0 B/s
Out 0.0 B/s
Data Size
0.0 B / 512.0 MB (0%)
Last 7 minutes
VERSION
REGION
CLUSTER TIER
TYPE
BACKUPS
LINKED APP SERVICES
ATLAS SQL
ATLAS SEARCH
6.0.8
AWS / Mumbai (ap-south-1)
M0 Sandbox (General)
Replica Set - 3 nodes
Inactive
None Linked
Connect
Create Index
+ Add Tag
System Status: All Good
©2023 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales




APP services




App Services
realm.mongodb.com/groups/64eebb6bdfcec73d96e0784f/apps
NO ENVIRONMENT
DATA ACCESS
Rules
Schema
App Users
Authentication
BUILD
Realm SDKs
Device Sync
GraphQL
Functions
Triggers
HTTPS Endpoints
Values
MANAGE
Linked Data Sources
Deployment
Hosting
Logs
App Settings
Build better apps faster with App Services
Seamlessly connect your data to mobile apps, websites, and other services with our fully managed services and APIs.
Learn more about App Services
1 Select a Template
Using App Services reduces the code you need to write. Get started quicker with Templates.
React.JS + Realm Web Boilerplate
2 Link your Data Source
Template Starter Applications let you build on top of data in a MongoDB Atlas cluster.
Cluster0
3 Name your Application
This name will be used internally and cannot be changed later.
Application-0
4 App Deployment Model
You can change this later in your Application Settings. View Docs
Single Region • Virginia (us-east-1) • AWS | Region chosen based on your cluster's region Change
Atlas
TINU CLARA...
Access Manager
Billing
All Clusters
Get Help
TINU CLARA EMMANUEL
Project 0
Data Services
App Services
Charts
TINU CLARA EMMANUEL'S ORG - 2023-07-19 > PROJECT 0 > APP SERVICES
Applications
Create App from Template
Create a New App
App Services are free until you exceed the free tier.
You can now see the state of your project's total usage as well as the specific usage of each of your apps.
Total App Usage Jul 01 - Jul 31. Free Tier resets in 13 days. Learn more Last Updated: Jul 19, 2023, 10:25:37 AM GMT+5:30
Requests
0/1M
FREE TIER USED
Data Transfer
0.00/10 GB
FREE TIER USED
Compute Runtime
0.00/500 Hr
FREE TIER USED
Sync Runtime
0.00/10K Hr
FREE TIER USED
Create an App Service
1 Name
This name will be used internally and cannot be changed later.
Application-1
Names must only include: ASCII letters, numbers, _ -
2 Link your Database
App Services securely stores your application data in your linked MongoDB Atlas cluster(s).
We'll automatically create a free-tier 6.0 Atlas cluster for you with 512 MB of space
Use an existing MongoDB Atlas Data Source
Cluster0
AWS, Cluster IDLE (AP_SOUTH_1)
RECOMMENDED
Use mongodb-atlas as the service name when referring to this data source in your app's Functions or SDKs.
CONNECT TO GITHUB (optional)
ADVANCED CONFIGURATION (optional)




## Result

The program was executed and the result was successfully obtained. Thus CO5 was obtained

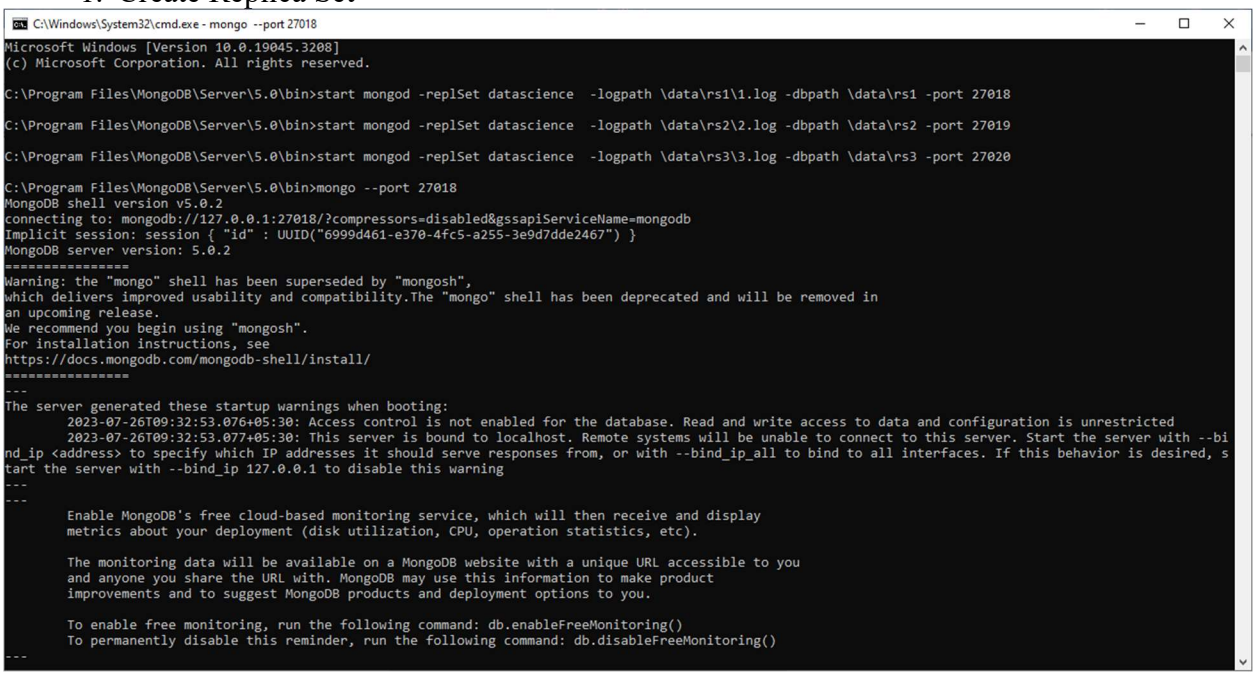
## Experiment No: 16

### Aim
Implementation of Replica Set on MongoDB

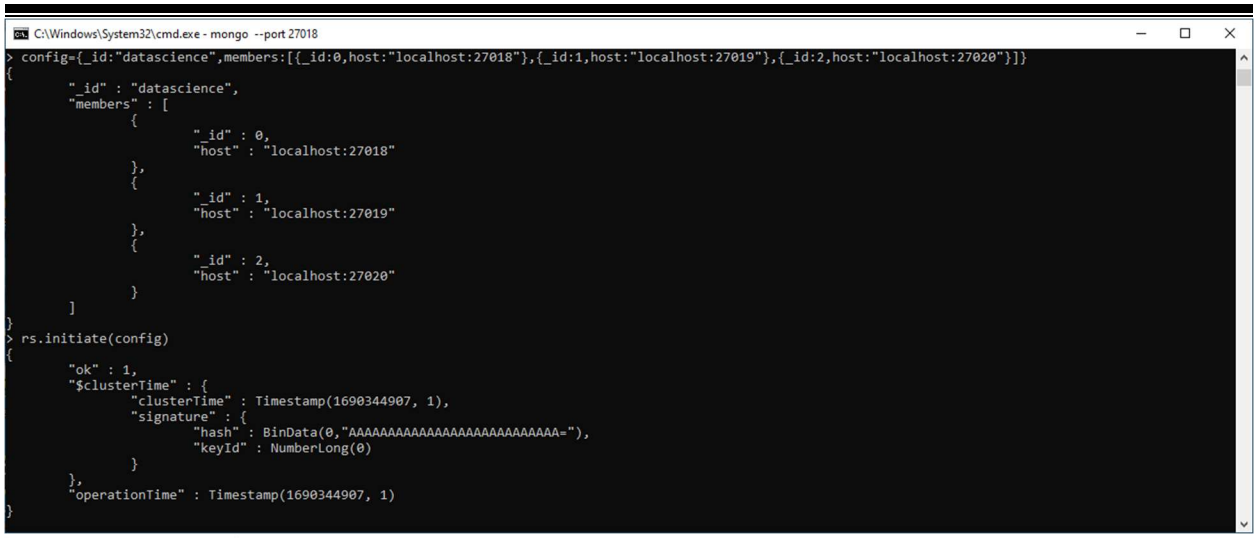
### CO5
To what extent you are able to understand the basic storage architecture of distributed file systems.

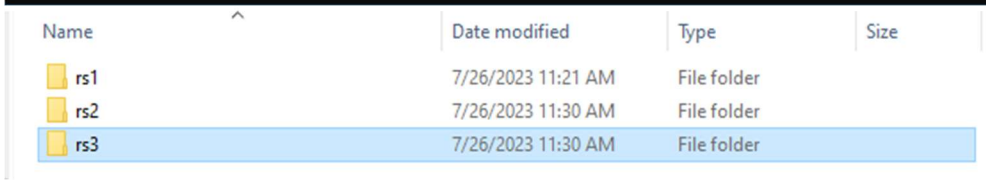
### Procedure

1. Create Replica Set

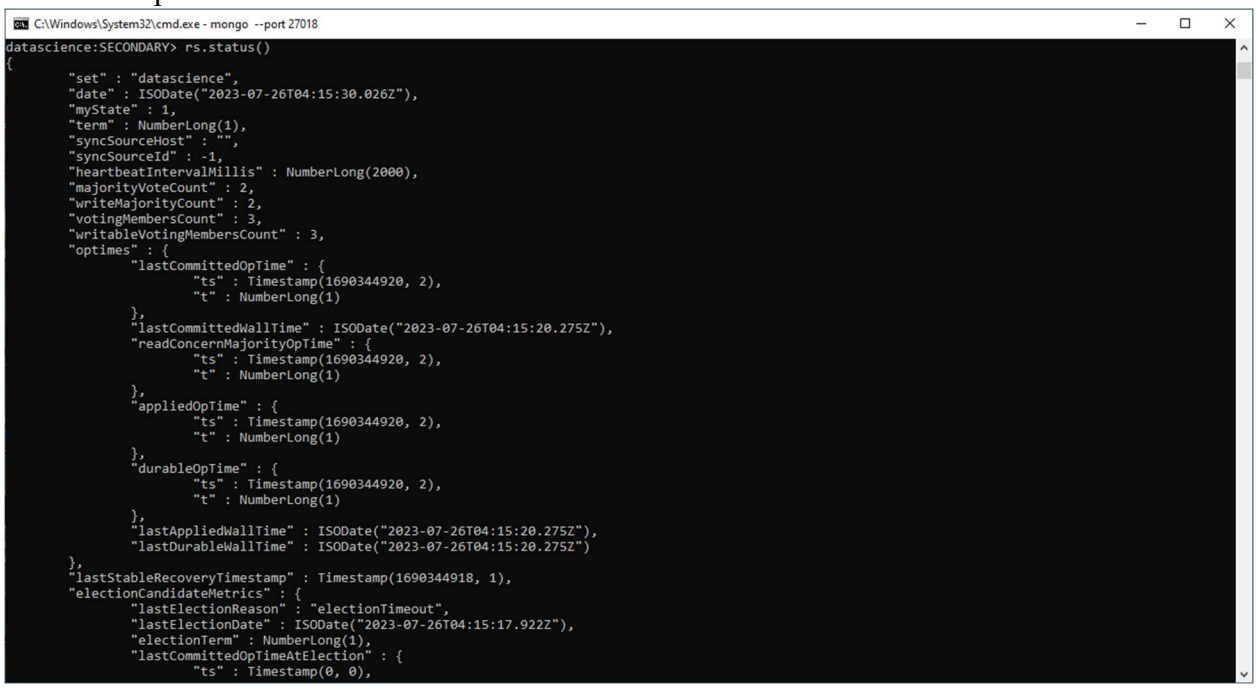



2. Configure Replica Set

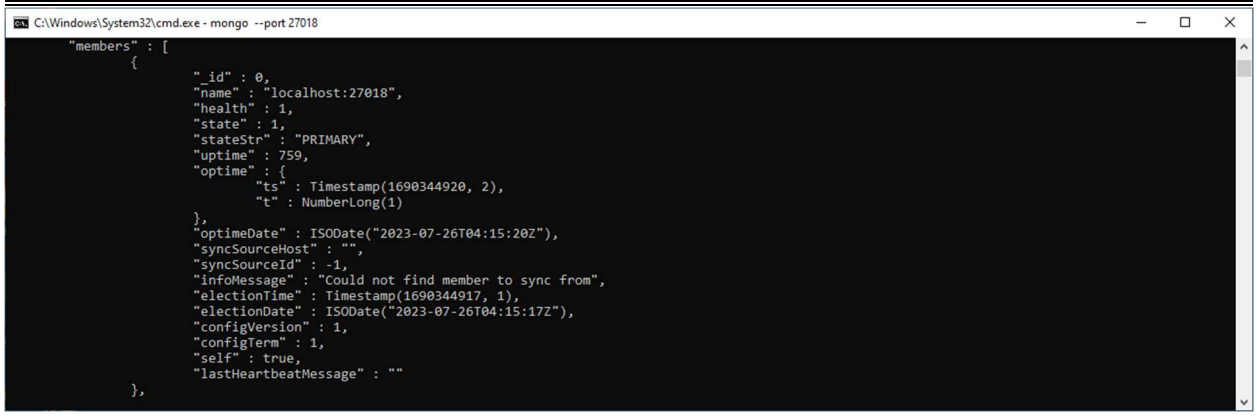


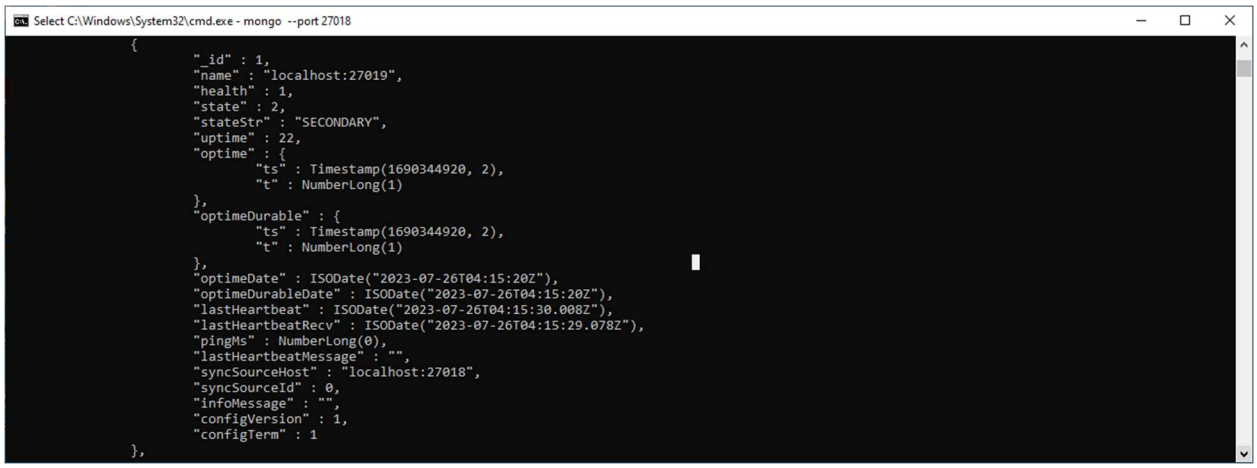



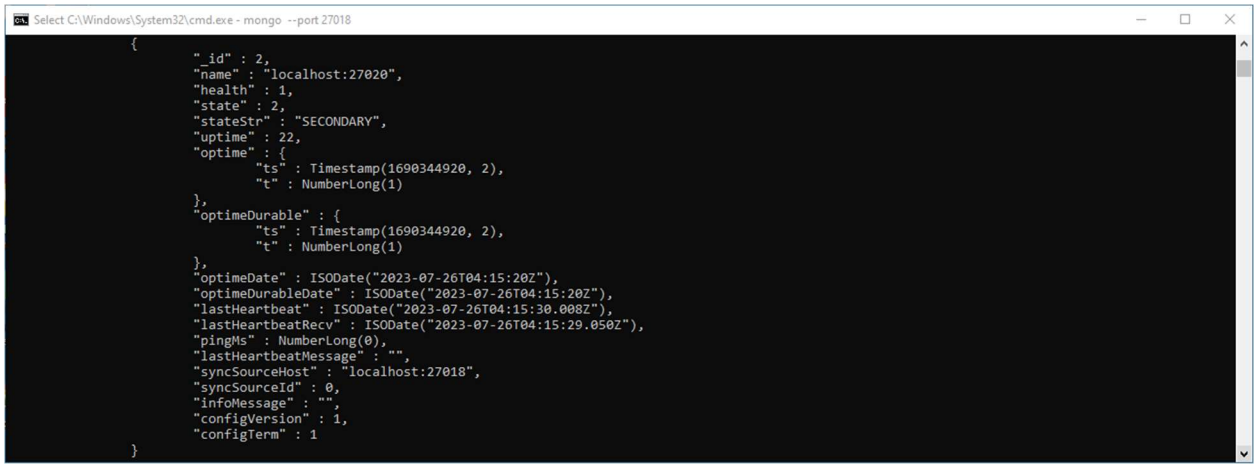

3. Replica Set Status




4. List Replica Set





5. Performing operations at Primary

```
Select C:\Windows\System32\cmd.exe - mongo --port 27018
datascience:PRIMARY> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
datascience:PRIMARY> use datascidb
switched to db datascidb
datascience:PRIMARY> db.student.insert({name:"amal"})
WriteResult({ "nInserted" : 1 })
datascience:PRIMARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
datascience:PRIMARY> use admin
switched to db admin
> db.shutdownServer()
server should be down...
> db.startServer()
uncaught exception: TypeError: db.startServer is not a function :
@(shell):1:1
> use admin
switched to db admin
```

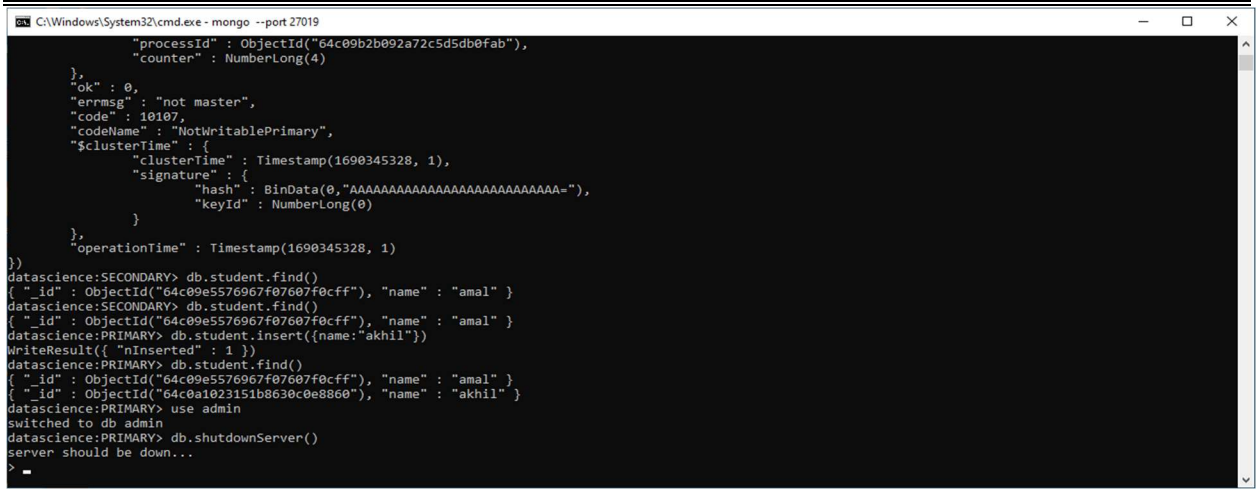
6. Performing operations at Secondary

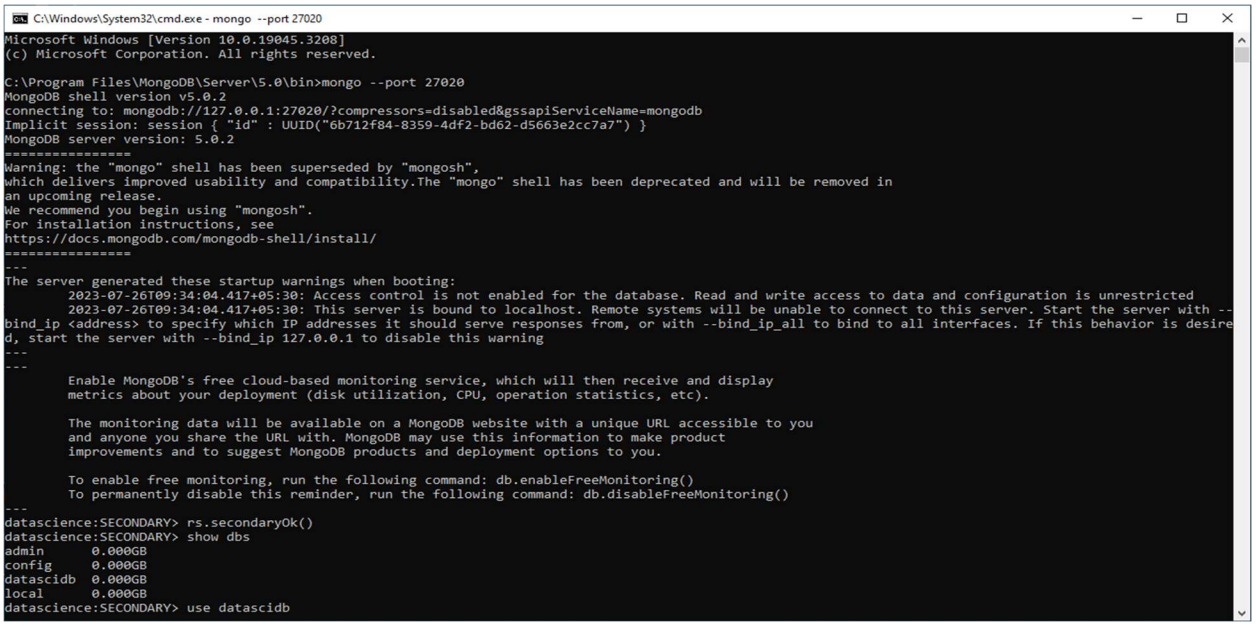
```
C:\Windows\System32\cmd.exe - mongo --port 27019
C:\Program Files\MongoDB\Server\5.0\bin>mongo --port 27019
MongoDB shell version v5.0.2
connecting to: mongodb://127.0.0.1:27019/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("5da5f244-9f73-435c-bf1e-5b838f2f77a1") }
MongoDB server version: 5.0.2
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
We recommend you begin using "mongosh".
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
---
The server generated these startup warnings when booting:
        2023-07-26T09:33:56.375+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
        2023-07-26T09:33:56.376+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bin
d_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, sta
rt the server with --bind_ip 127.0.0.1 to disable this warning
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
datascience:SECONDARY> show dbs
uncaught exception: Error: listDatabases failed:{
        "topologyVersion" : {
                "processId" : ObjectId("64c09b2b092a72c5d5db0fab"),
                "counter" : NumberLong(4)
        },
```

```
C:\Windows\System32\cmd.exe - mongo --port 27019
        },
        "ok" : 0,
        "errmsg" : "not master and slaveOk=false",
        "code" : 13435,
        "codeName" : "NotPrimaryNoSecondaryOk",
        "$clusterTime" : {
                "clusterTime" : Timestamp(1690345198, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1690345198, 1)
} :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
Mongo.prototype.getDBs/<@src/mongo/shell/mongo.js:145:19
Mongo.prototype.getDBs@src/mongo/shell/mongo.js:97:12
shellHelper.show@src/mongo/shell/utils.js:956:13
shellHelper@src/mongo/shell/utils.js:838:15
@(shellhelp2):1:1
datascience:SECONDARY> rs.slaveOk()
WARNING: slaveOk() is deprecated and may be removed in the next major release. Please use secondaryOk() instead.
datascience:SECONDARY> rs.secondaryOk()
datascience:SECONDARY> show dbs
admin      0.000GB
config     0.000GB
datascidb  0.000GB
local      0.000GB
datascience:SECONDARY> use datascidb
switched to db datascidb
datascience:SECONDARY> db.student.insert({name:"vimal"})
WriteCommandError({
        "topologyVersion" : {
                "processId" : ObjectId("64c09b2b092a72c5d5db0fab"),
                "counter" : NumberLong(4)
        },
        "ok" : 0,
```


C:\Windows\System32\cmd.exe - mongo --port 27019
"processId" : ObjectId("64c09b2b092a72c5d5db0fab"),
"counter" : NumberLong(4)
},
"ok" : 0,
"errmsg" : "not master",
"code" : 10107,
"codeName" : "NotWritablePrimary",
"$clusterTime" : {
"clusterTime" : Timestamp(1690345328, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
},
"operationTime" : Timestamp(1690345328, 1)
})
datascience:SECONDARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
datascience:SECONDARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
datascience:PRIMARY> db.student.insert({name:"akhil"})
WriteResult({ "nInserted" : 1 })
datascience:PRIMARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
{ "_id" : ObjectId("64c0a1023151b8630c0e8860"), "name" : "akhil" }
datascience:PRIMARY> use admin
switched to db admin
datascience:PRIMARY> db.shutdownServer()
server should be down...
>


C:\Windows\System32\cmd.exe - mongo --port 27020
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.
C:\Program Files\MongoDB\Server\5.0\bin>mongo --port 27020
MongoDB shell version v5.0.2
connecting to: mongodb://127.0.0.1:27020/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("6b712f84-8359-4df2-bd62-d5663e2cc7a7") }
MongoDB server version: 5.0.2
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
We recommend you begin using "mongosh".
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
The server generated these startup warnings when booting:
2023-07-26T09:34:04.417+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2023-07-26T09:34:04.417+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable this warning
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
datascience:SECONDARY> rs.secondaryOk()
datascience:SECONDARY> show dbs
admin 0.000GB
config 0.000GB
datascidb 0.000GB
local 0.000GB
datascience:SECONDARY> use datascidb


C:\Windows\System32\cmd.exe - mongo --port 27020
datascience:SECONDARY> use datascidb
switched to db datascidb
datascience:SECONDARY> db.student.insert({name:"vikas"})
WriteCommandError({
"topologyVersion" : {
"processId" : ObjectId("64c09b33cf106257bbb46f97"),
"counter" : NumberLong(5)
},
"ok" : 0,
"errmsg" : "not master",
"code" : 10107,
"codeName" : "NotWritablePrimary",
"$clusterTime" : {
"clusterTime" : Timestamp(1690345478, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
},
"operationTime" : Timestamp(1690345478, 1)
})
datascience:SECONDARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
datascience:SECONDARY> db.student.find()
{ "_id" : ObjectId("64c09e5576967f07607f0cff"), "name" : "amal" }
{ "_id" : ObjectId("64c0a1023151b8630c0e8860"), "name" : "akhil" }
datascience:SECONDARY>

## Result

The program was executed and the result was successfully obtained. Thus CO5 was obtained