Experiment No: 1

Aim:

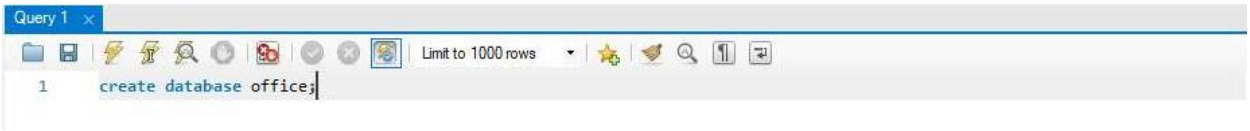
Creation of a database using DDL commands.

CO1:

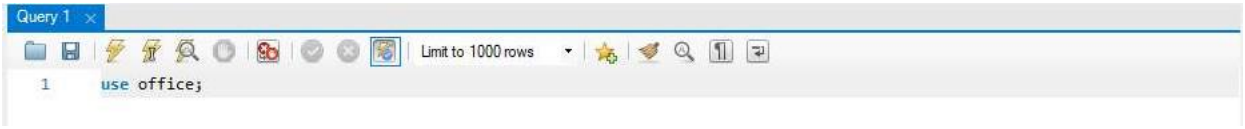
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

Procedure:

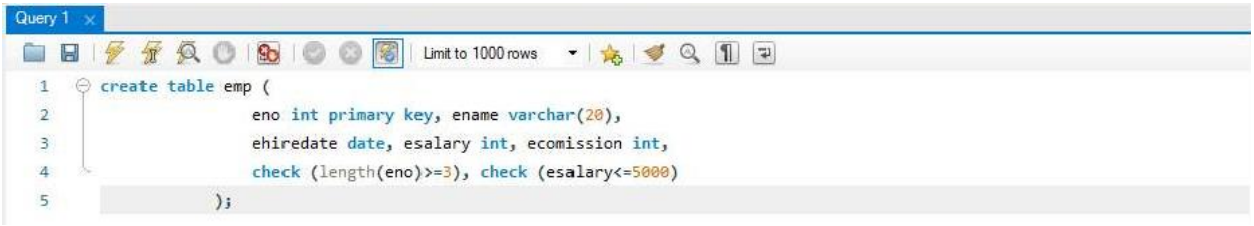
1. CREATE DATABASE



2. USE DATABASE



3. CREATE TABLE - EMP



4. INSERT VALUES INTO TABLE - EMP

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;
```

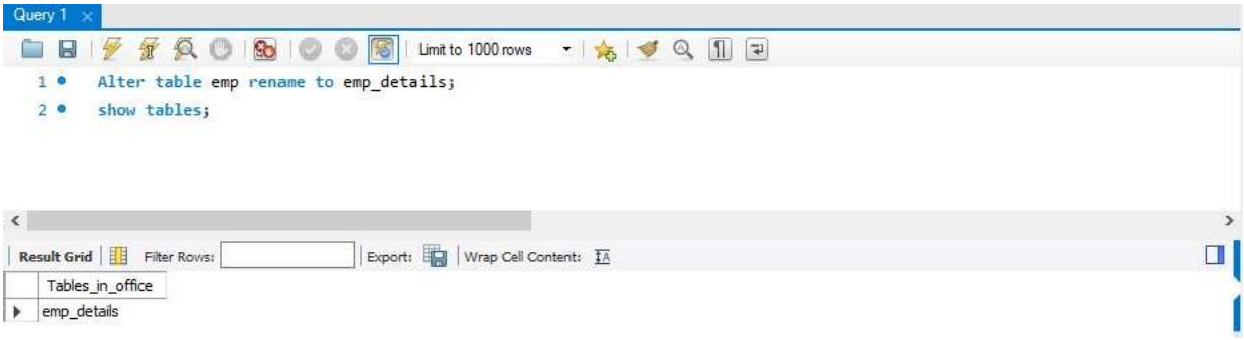
eno	ename	ehiredate	esalary	ecomission
101	Ramesh	1980-01-07	5000	NULL
102	Ajay	1985-05-07	5000	500
103	Ravi	1981-12-08	1500	NULL
104	Nikesh	1983-03-03	3000	700
105	Ravi	1985-05-07	3000	NULL
NULL	NULL	NULL	NULL	NULL

7. RENAME COLUMN - ename to name

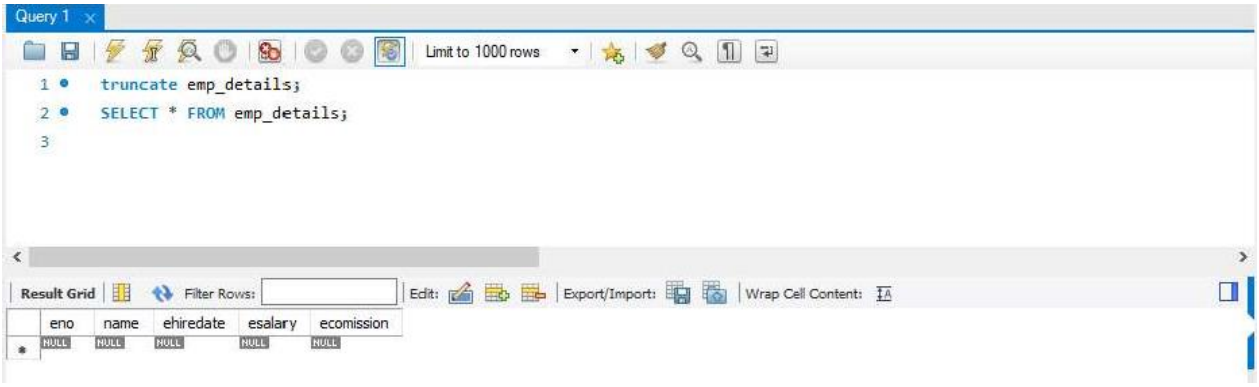
```
ALTER TABLE emp RENAME COLUMN ename to name;
SELECT * FROM emp;
```

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

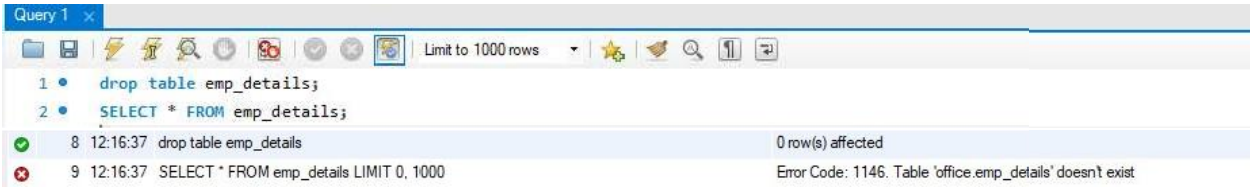
8. RENAME TABLE



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:

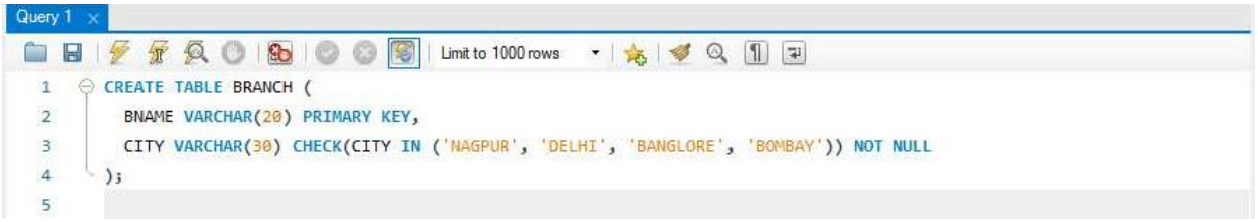
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

Procedure:

1. CREATE DATABASE

2. USE DATABASE

3. CREATE TABLE - BRANCH



4. CREATE TABLE - CUSTOMER

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

```
alter table emp add constraint chk_salary check (esalary >= 1500);
```

12. ADD CONSTRAINT - length(eno) >= 3

```
alter table emp add constraint chk_empno_length check (length(eno<=3));
```

Result:

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

Experiment No: 3

Aim:

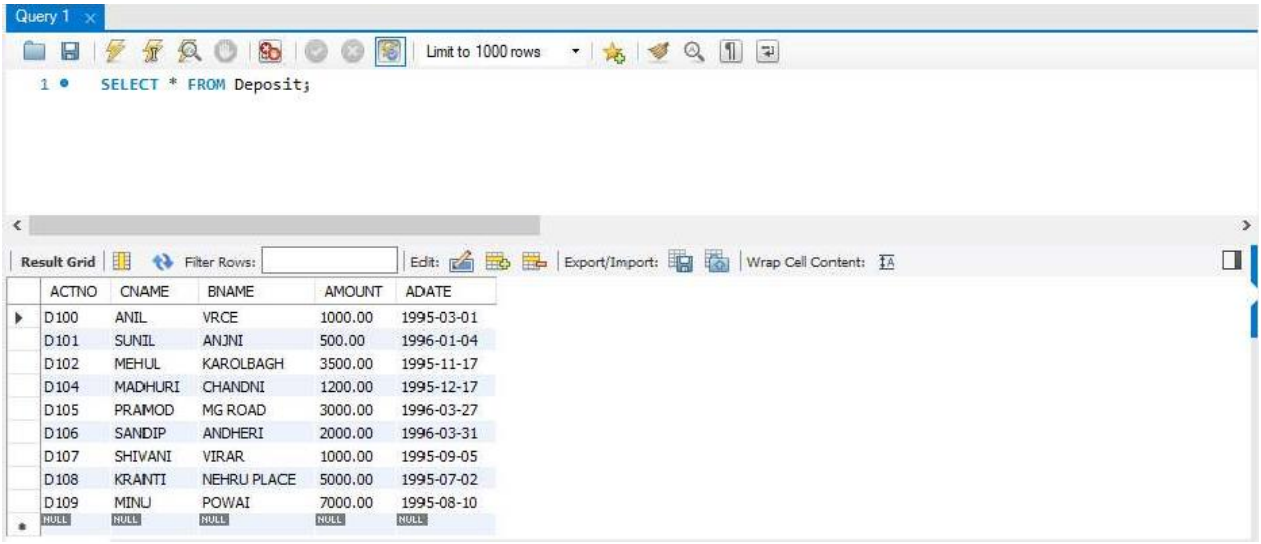
To familiarize with selecting data from single table

CO1:

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

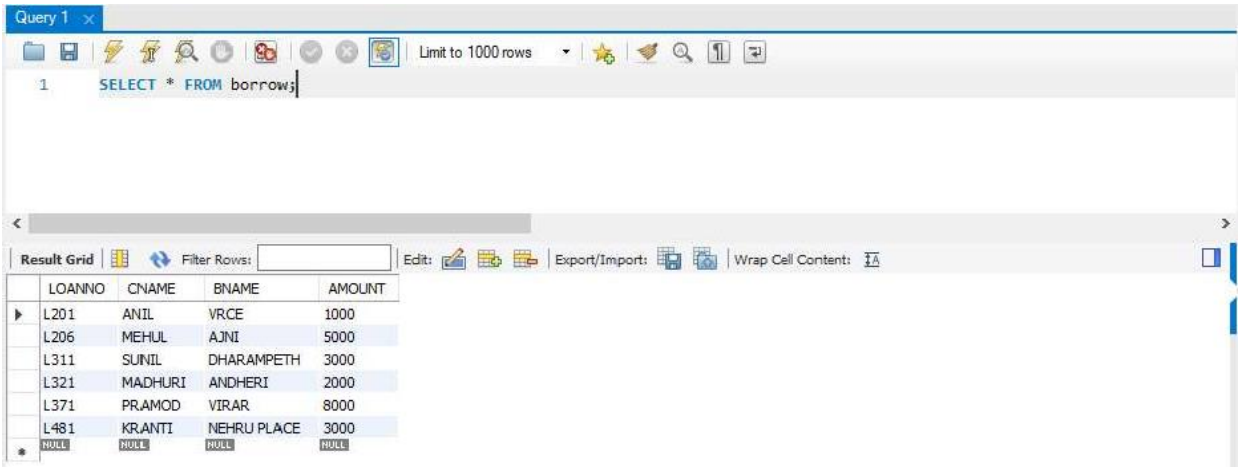
Procedure:

1. List all data from table deposit.



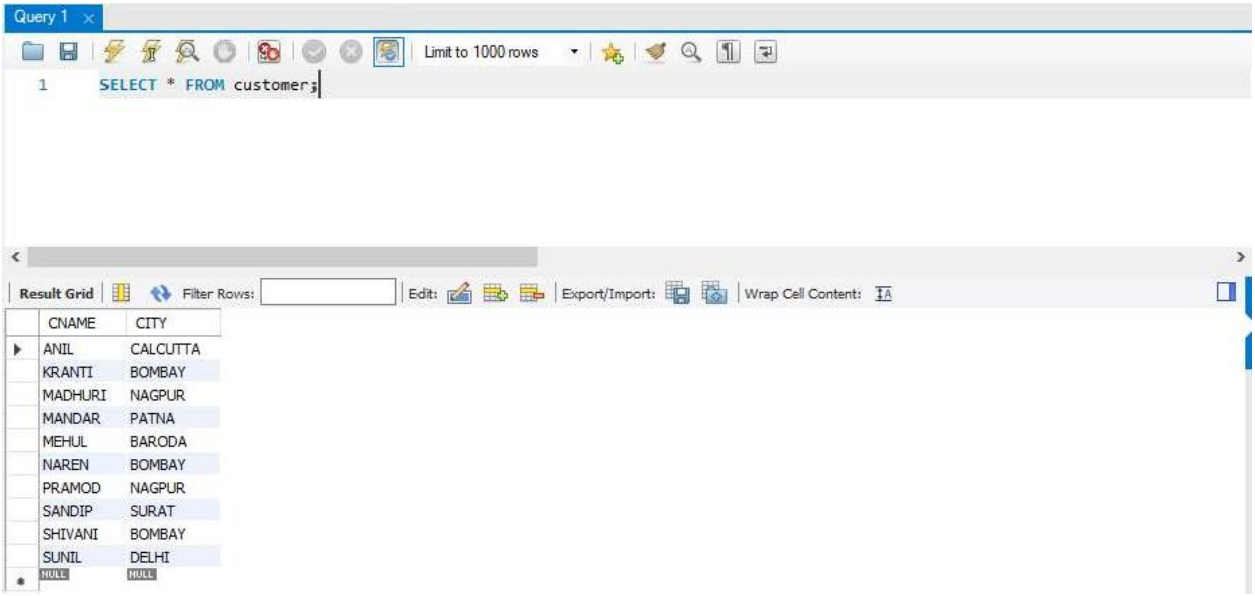
ACTNO	CNAME	BNAME	AMOUNT	ADATE
D100	ANIL	VRCE	1000.00	1995-03-01
D101	SUNIL	ANJNI	500.00	1996-01-04
D102	MEHUL	KAROLBAGH	3500.00	1995-11-17
D104	MADHURI	CHANDNI	1200.00	1995-12-17
D105	PRAMOD	MG ROAD	3000.00	1996-03-27
D106	SANDIP	ANDHERI	2000.00	1996-03-31
D107	SHIVANI	VIRAR	1000.00	1995-09-05
D108	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



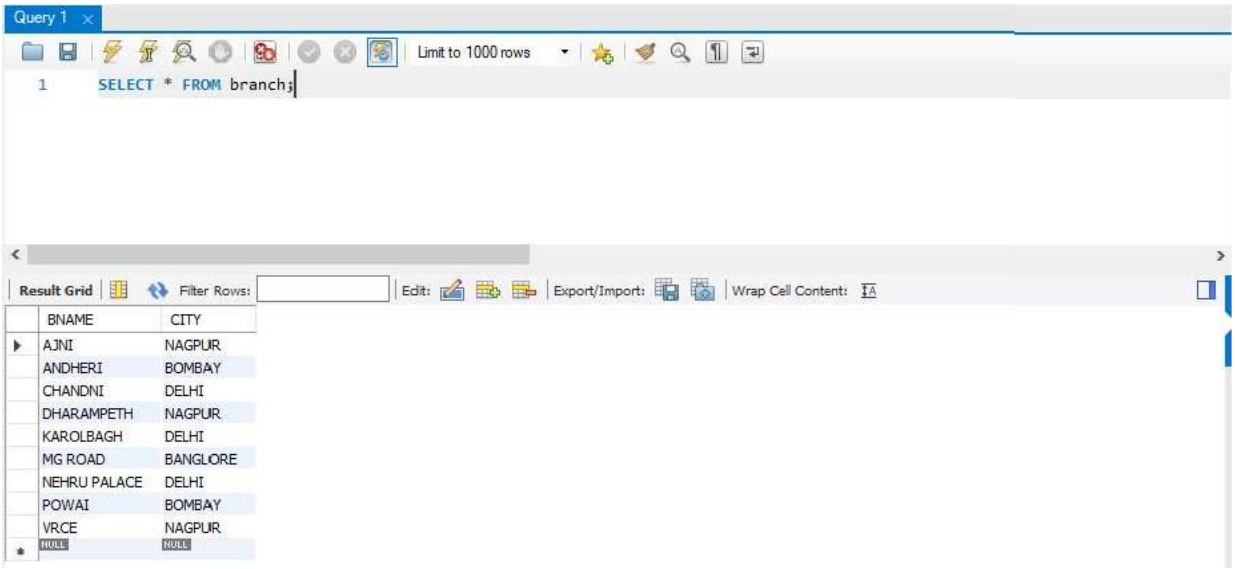
LOANNO	CNAME	BNAME	AMOUNT
L201	ANIL	VRCE	1000
L206	MEHUL	AJNI	5000
L311	SUNIL	DHARAMPETH	3000
L321	MADHURI	ANDHERI	2000
L371	PRAMOD	VIRAR	8000
L481	KRANTI	NEHRU PLACE	3000
NULL	NULL	NULL	NULL

3. List all data from customer



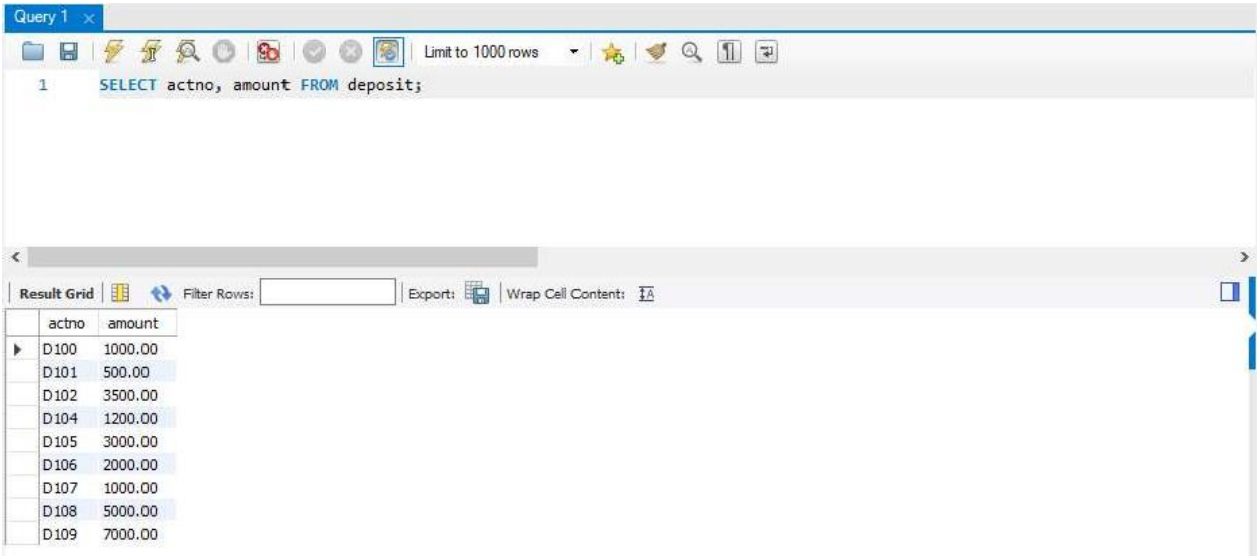
CNAME	CITY
ANIL	CALCUTTA
KRANTI	BOMBAY
MADHURI	NAGPUR
MANDAR	PATNA
MEHUL	BARODA
NAREN	BOMBAY
PRAMOD	NAGPUR
SANDIP	SURAT
SHIVANI	BOMBAY
SUNIL	DELHI
NULL	NULL

4. List all data from branch

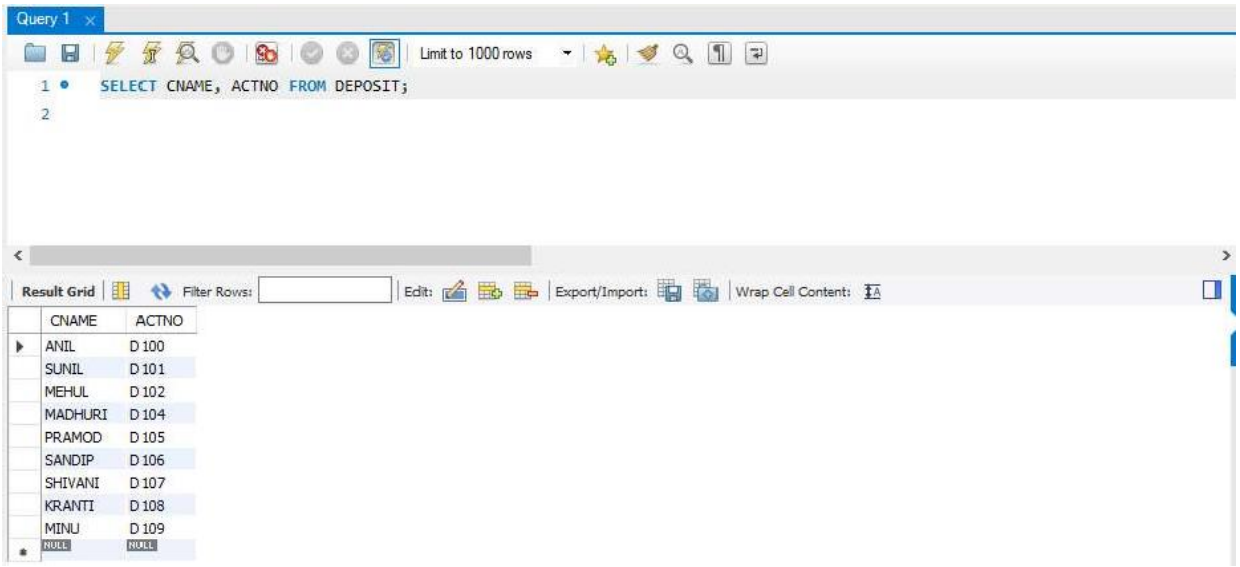


BNAME	CITY
AJNI	NAGPUR
ANDHERI	BOMBAY
CHANDNI	DELHI
DHARAMPETH	NAGPUR
KAROLBAGH	DELHI
MG ROAD	BANGLORE
NEHRU PALACE	DELHI
POWAI	BOMBAY
VRCE	NAGPUR
NULL	NULL

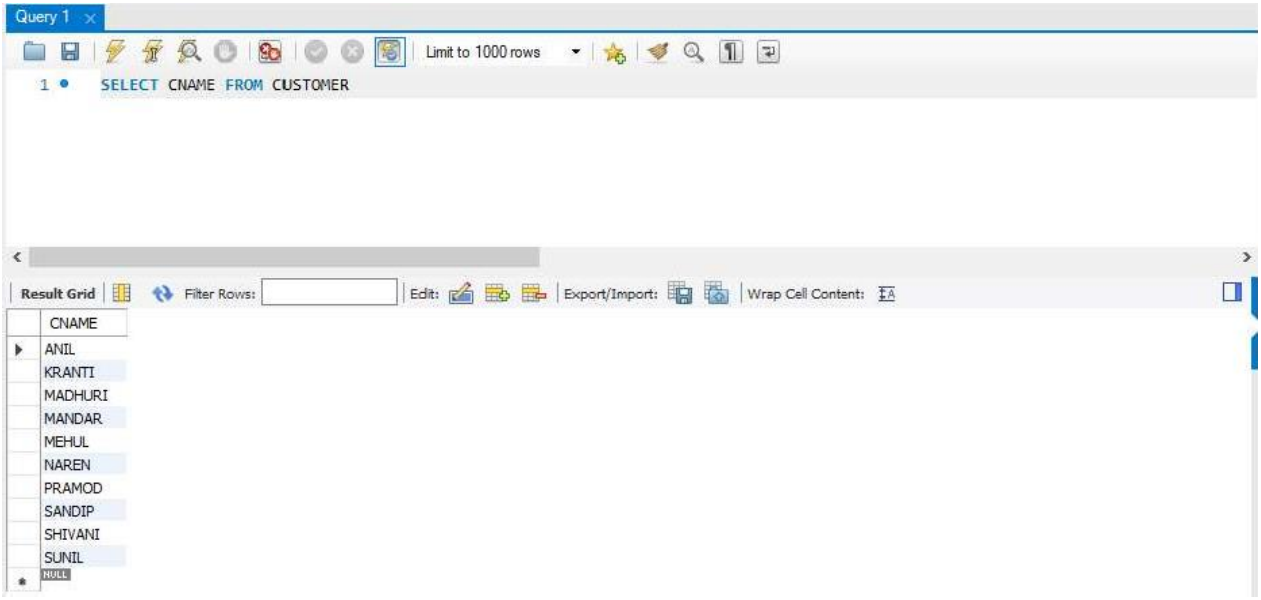
5. Give account no and amount of deposit



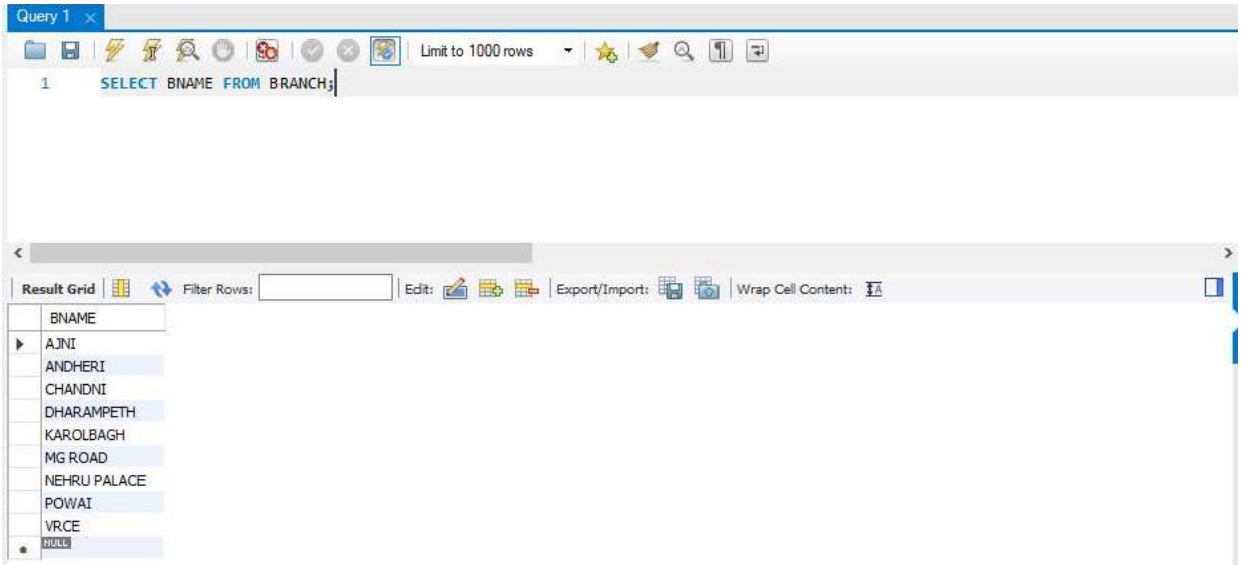
6. Give customer name and account no of depositors



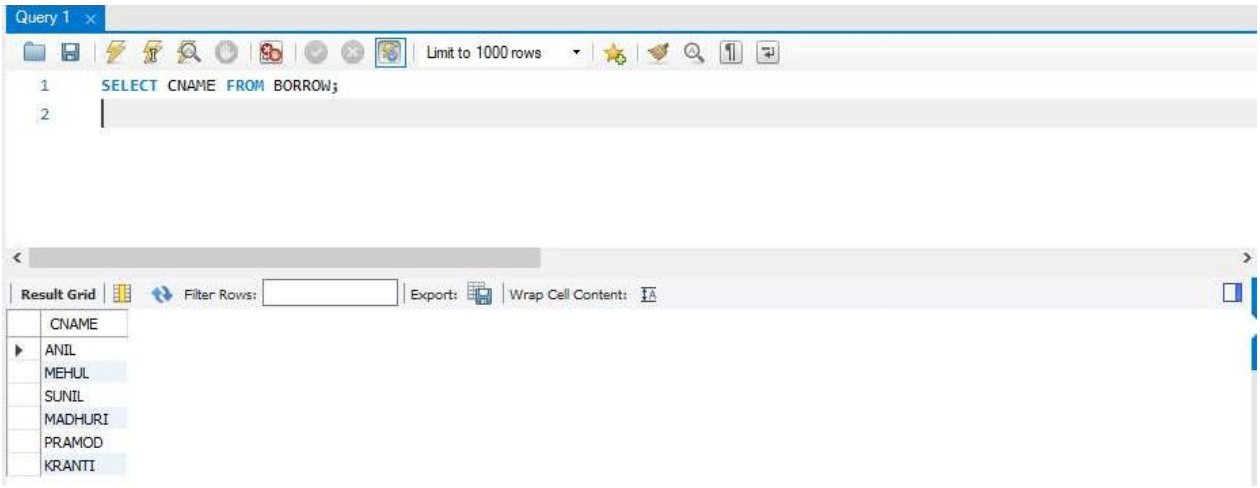
7. Give name of customers



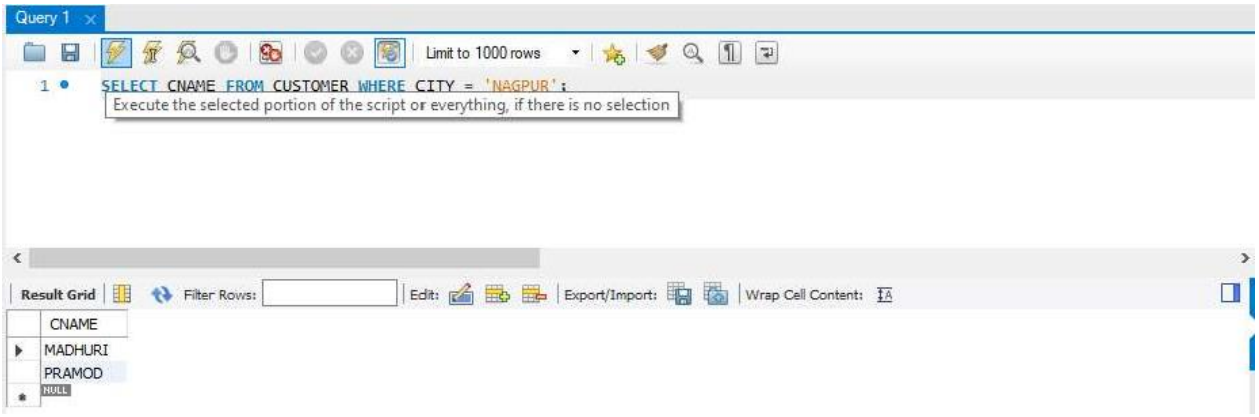
8. Give name of branches



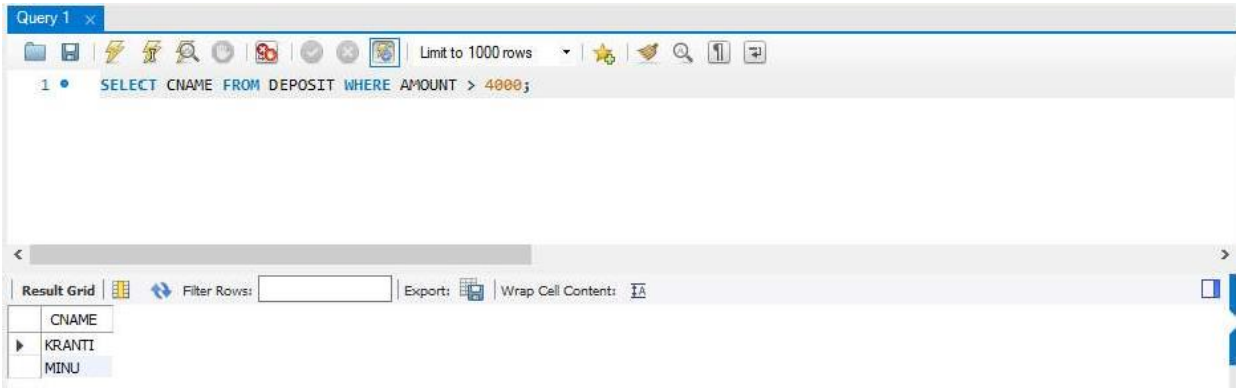
9. Give name of borrows



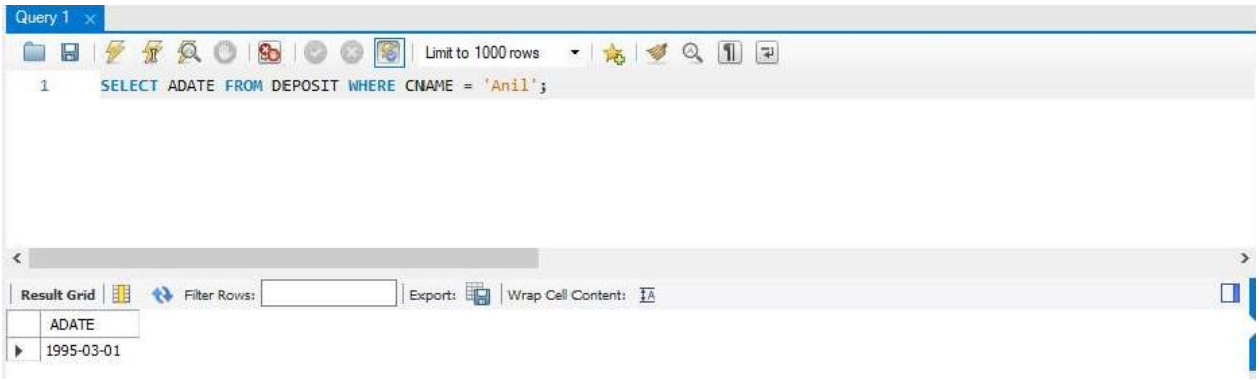
10. Give names of customer living in city Nagpur



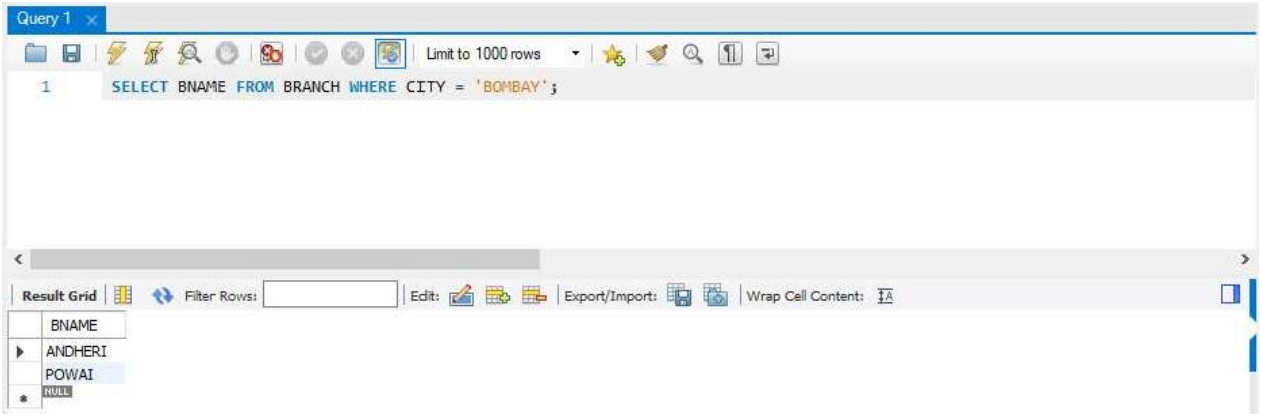
11. Give names of depositors having amount greater than 4000



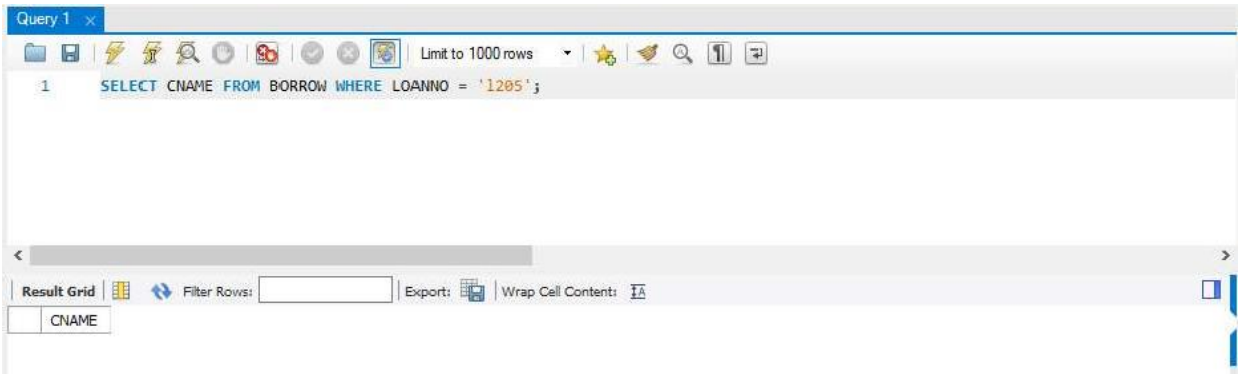
12. Give account date of Anil



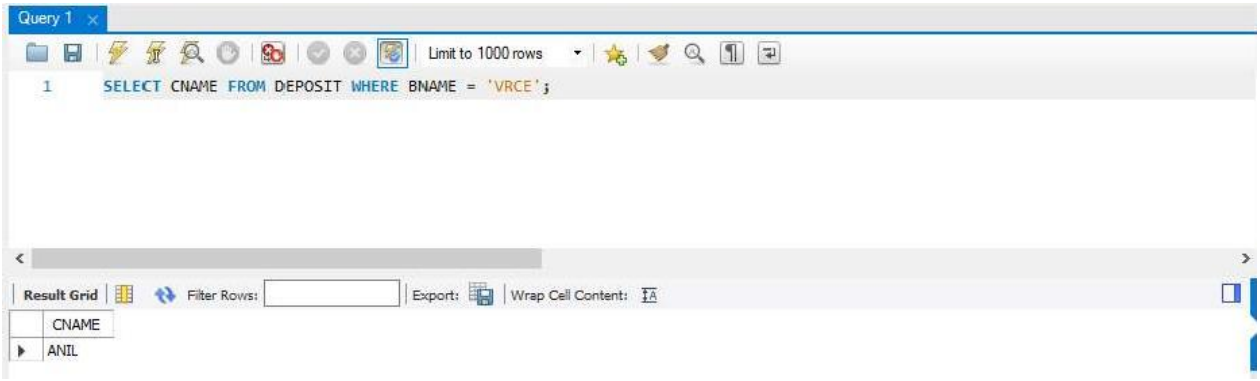
13. Give name of all branches located in Bombay



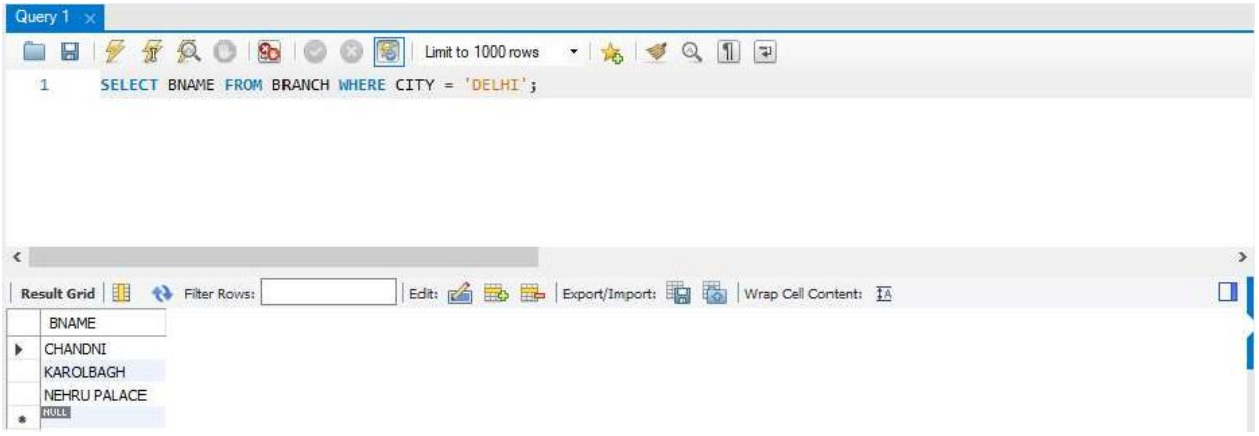
14. Give name of borrower having loan number l205



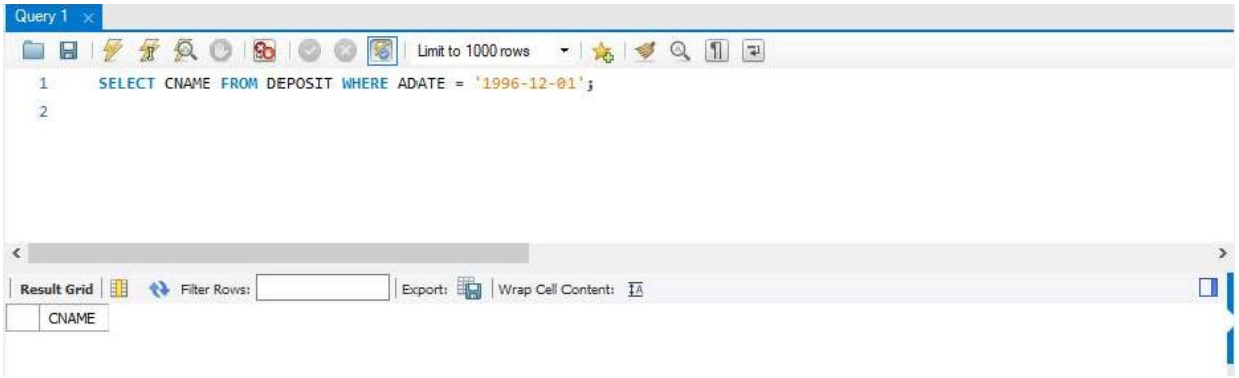
15. Give names of depositors having account at VRCE



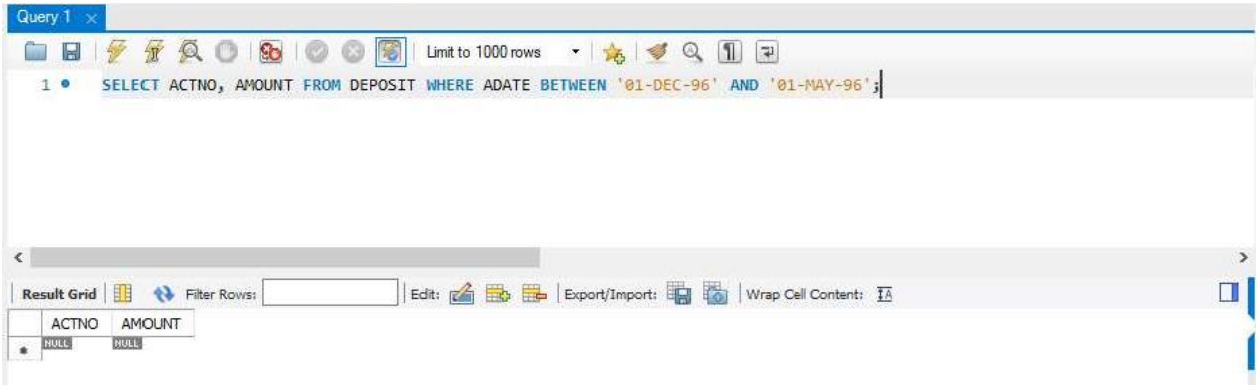
16. Give names of all branched located in city Delhi



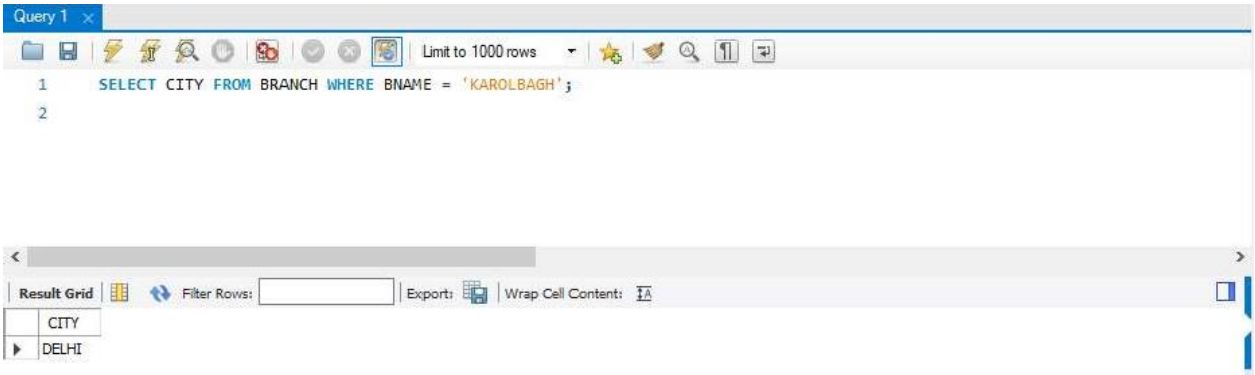
17. Give name of the customers who opened account date '1-12-96'



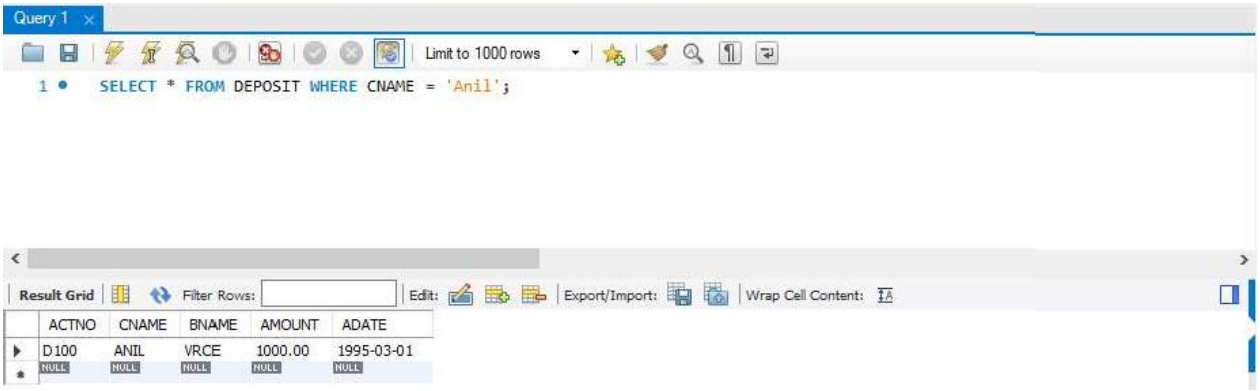
18. Give account no and deposit amount of customers having account opened between dates '1- 12-96' and '1-5-96'



19. Give name of the city where branch KAROL BAGH is located



20. Give details of customer ANIL



Result

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

Experiment No: 4

Aim

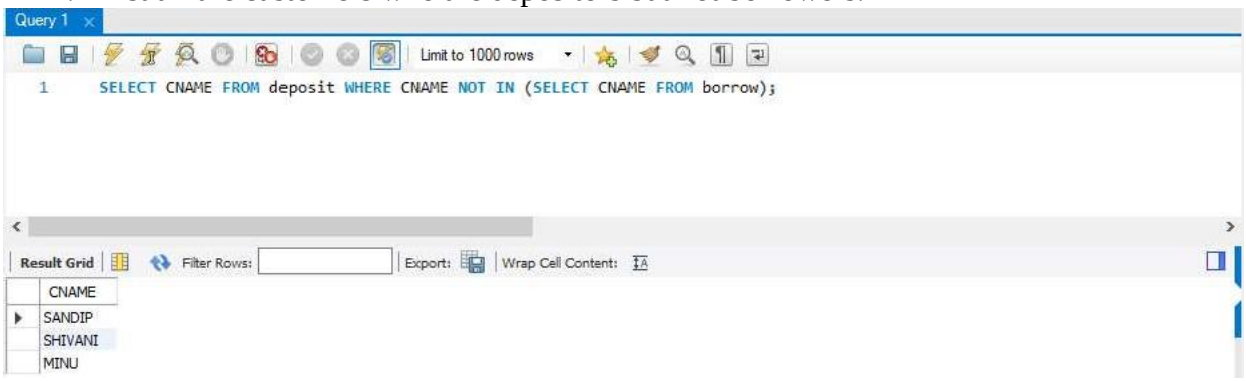
To familiarize with Set Operations.

CO1

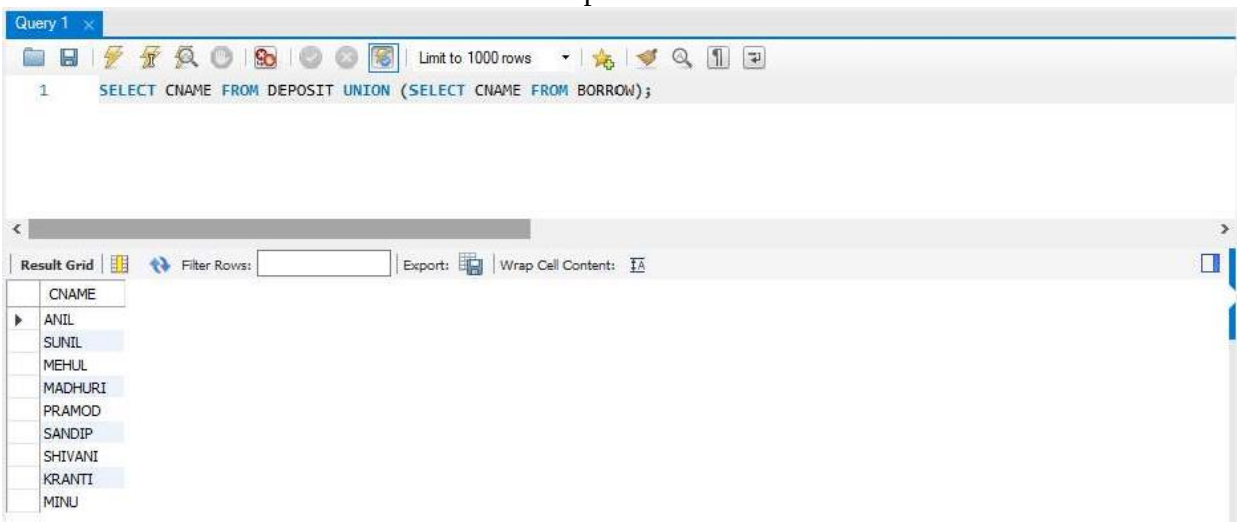
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

Procedure

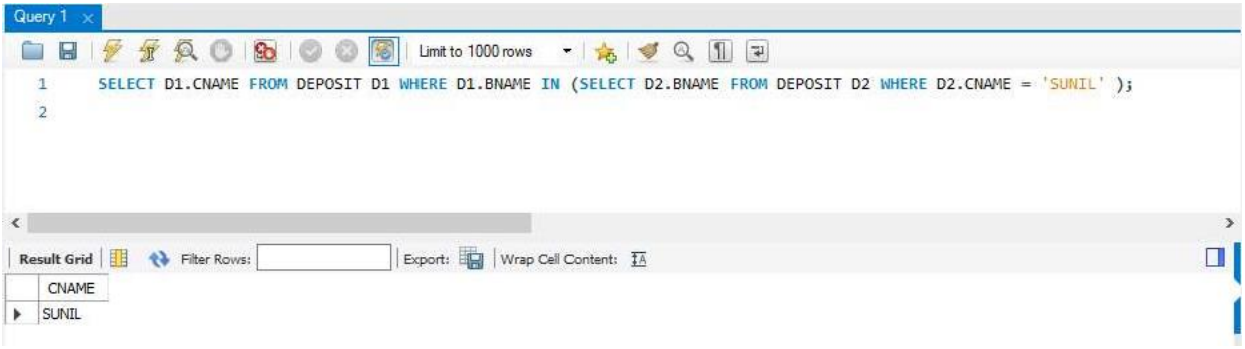
1. List all the customers who are depositors but not borrowers.



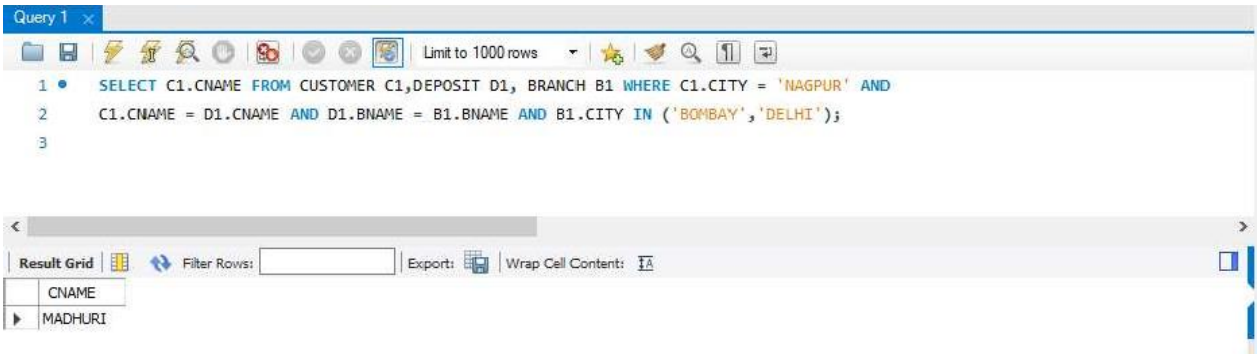
2. List all the customers who are both depositors and borrowers.



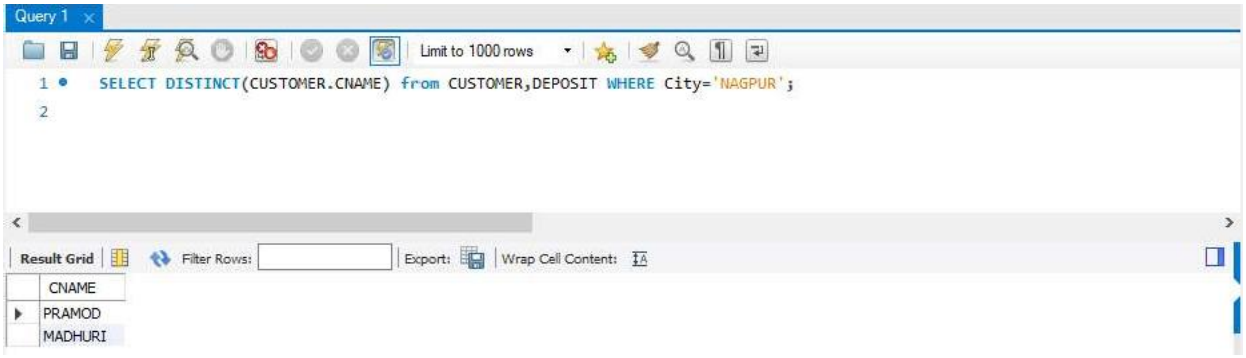
3. List all the depositors having deposit in all the branches where Sunil is having Account



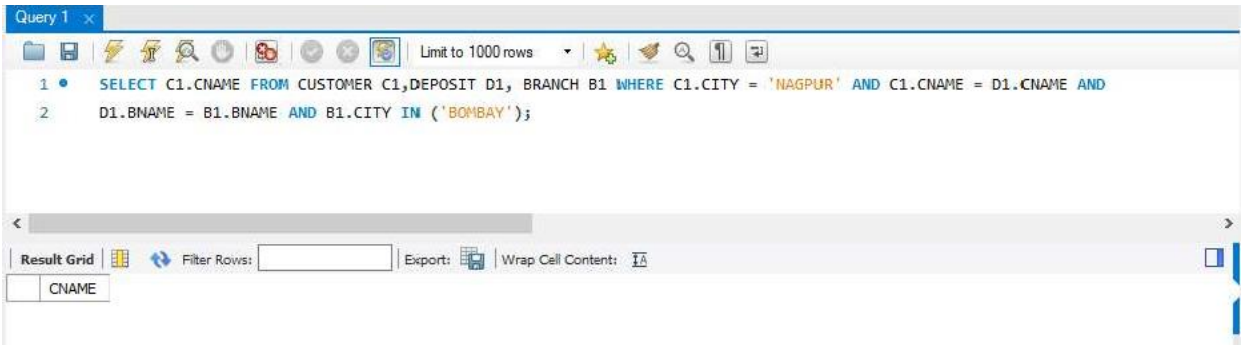
4. List all the customers living in city NAGPUR and having branch city BOMBAY or DELHI



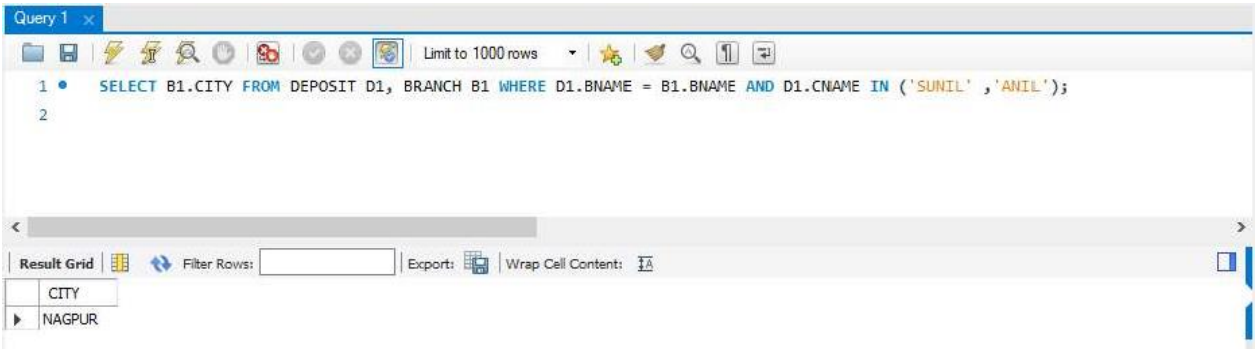
5. List all the depositors living in city NAGPUR



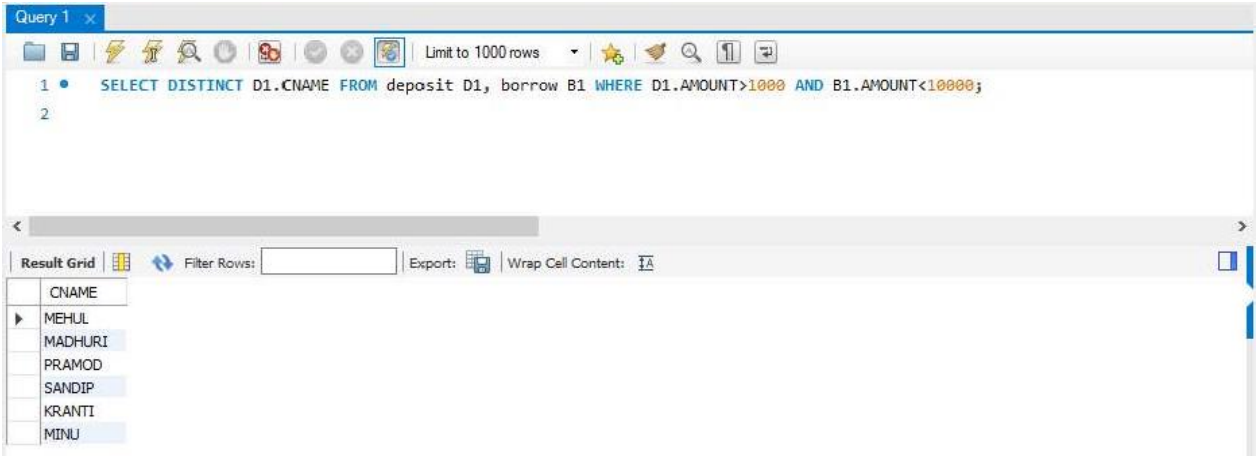
6. List all the depositors living in the city NAGPUR and having branch in city BOMBAY



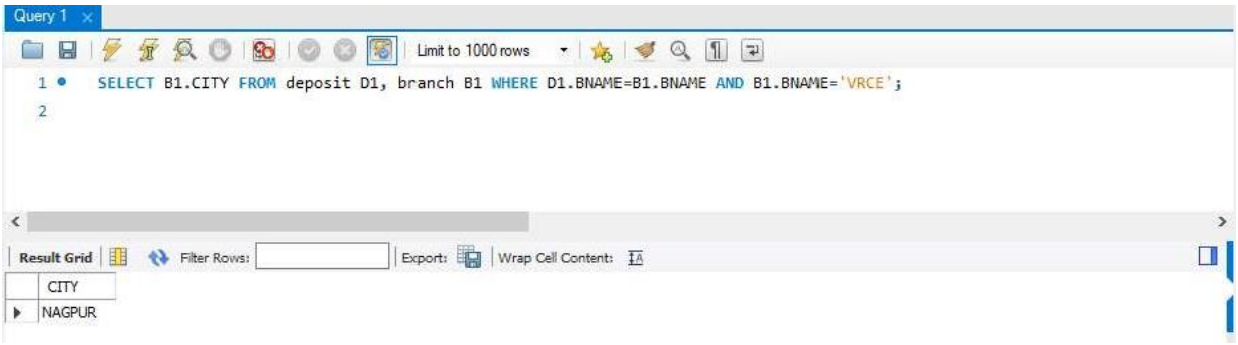
7. List the branch cities of Anil and Sunil



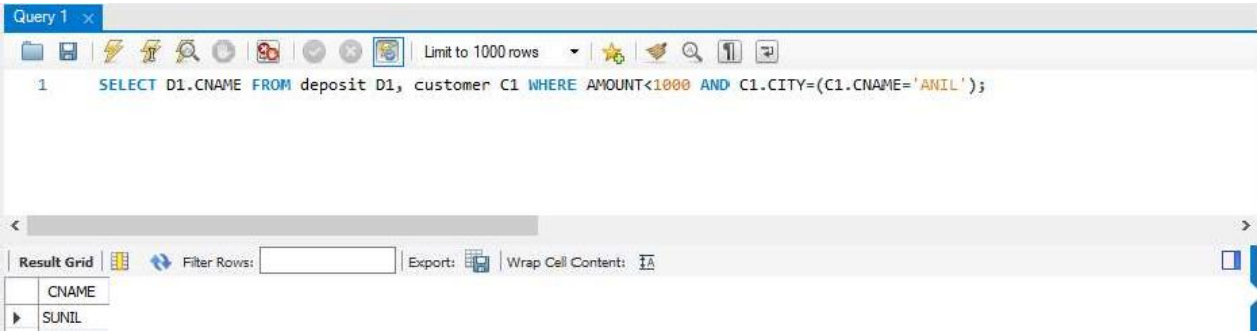
8. List the customers having deposit greater than 1000 and loan less than 10000.



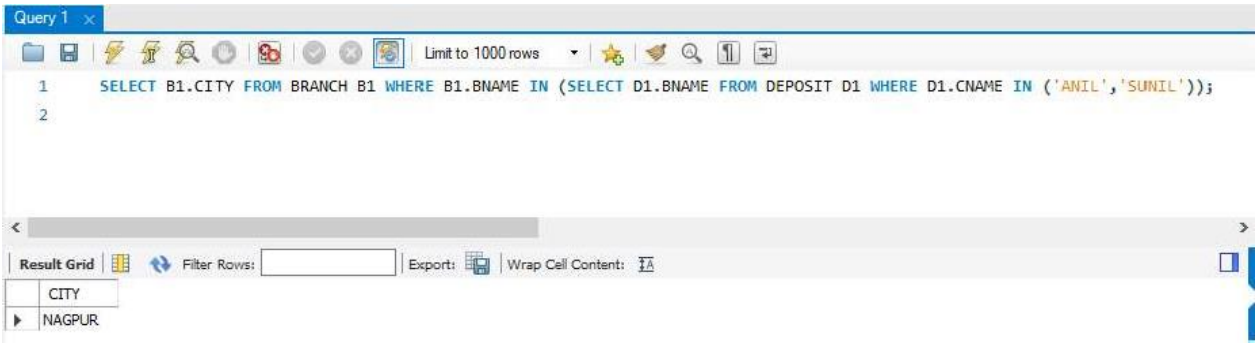
9. List the cities of depositors having branch VRCE.



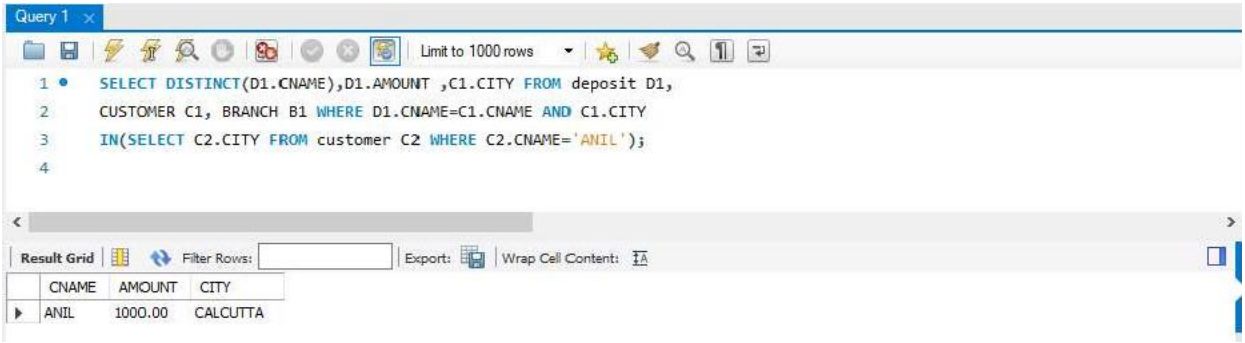
10. List the depositors having amount less than 1000 and living in the same city as Anil



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



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

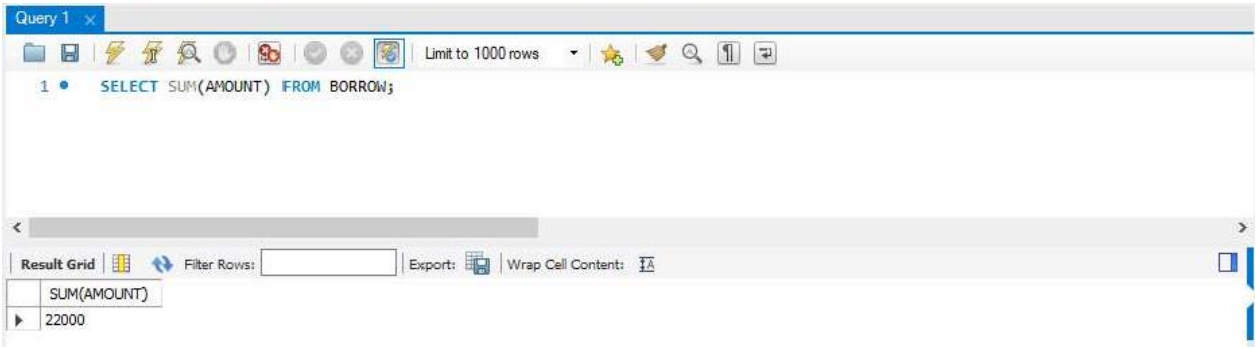
To familiarize with Aggregate Functions.

CO1

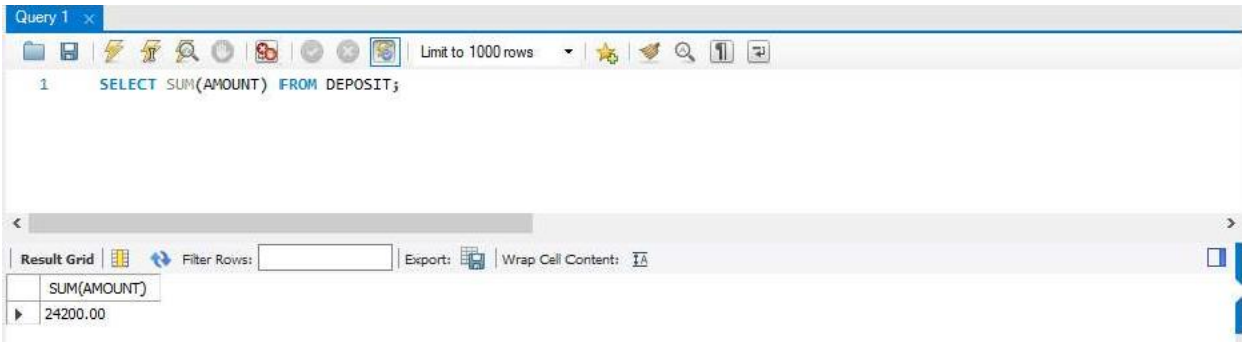
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

Procedure

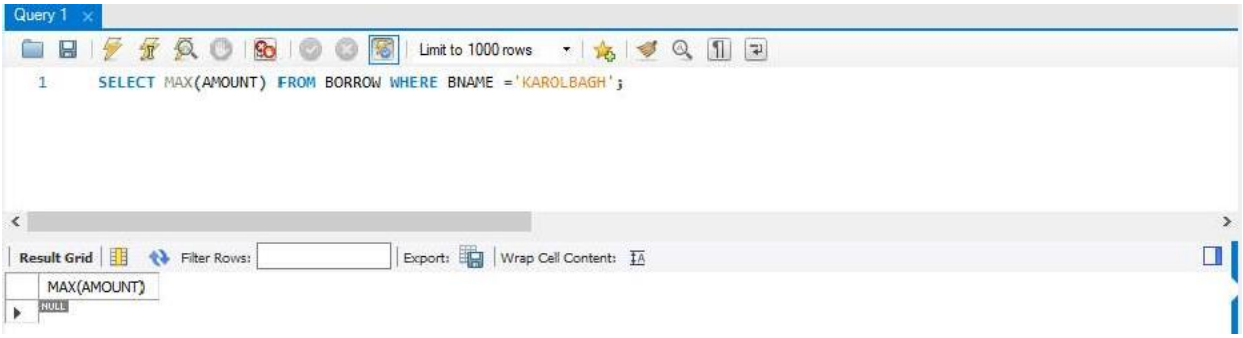
1. List total loan



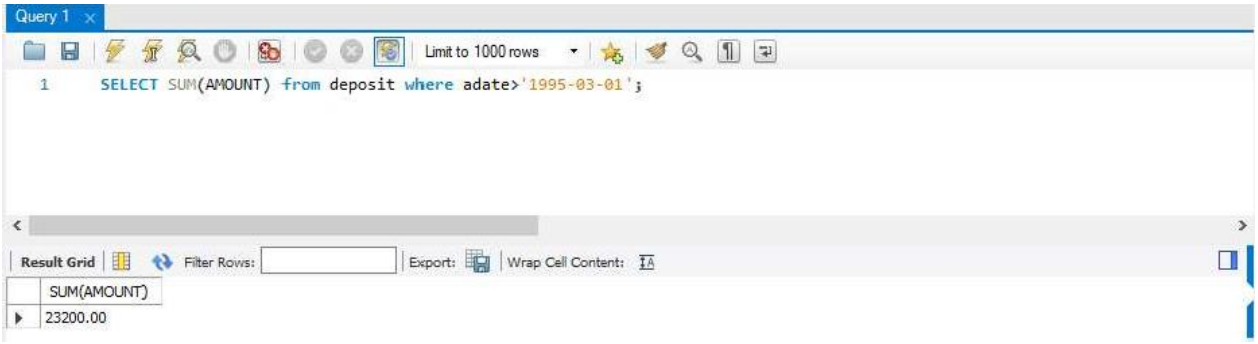
2. List total deposit



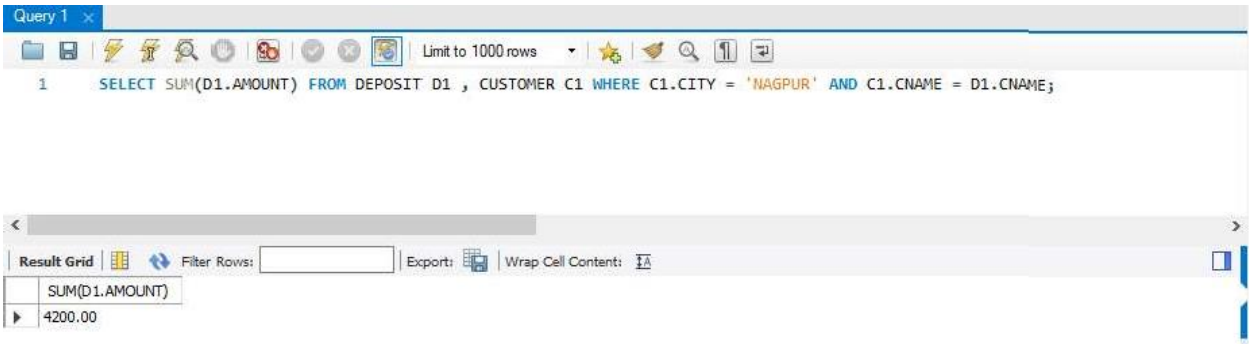
3. List total loan taken from KAROLBAGH branch



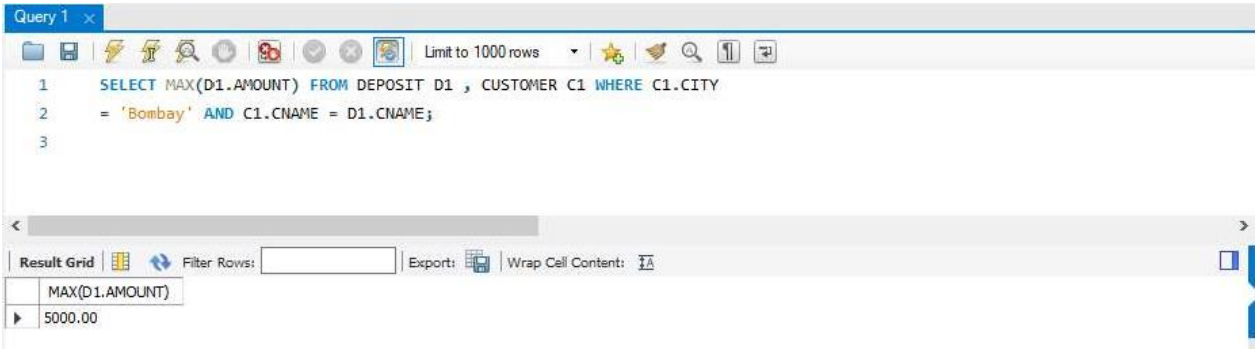
4. List total deposit of customers having account date later than 1-Jan-96



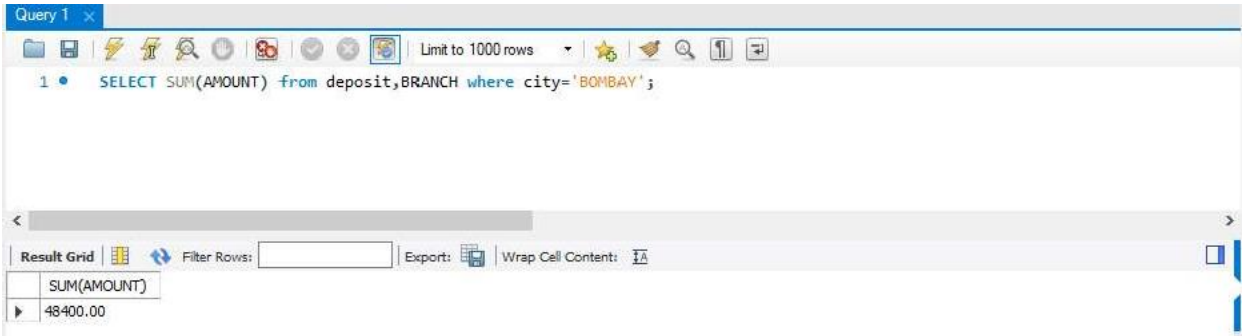
5. List total deposit of customers living in city NAGPUR



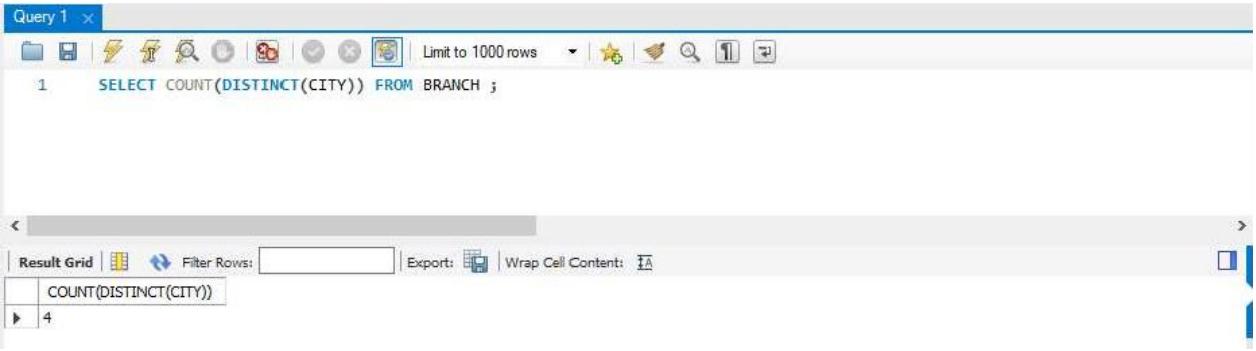
6. List maximum deposit of customer living in Bombay



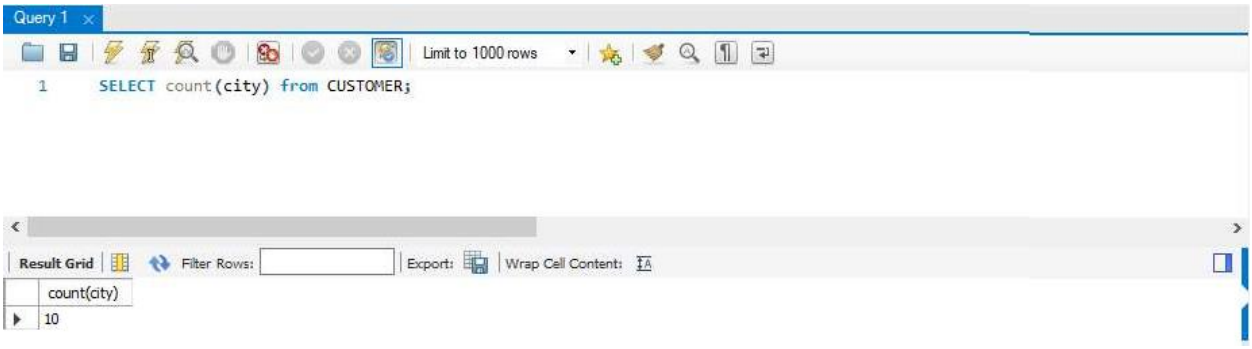
7. List total deposit of customer having branch in BOMBAY



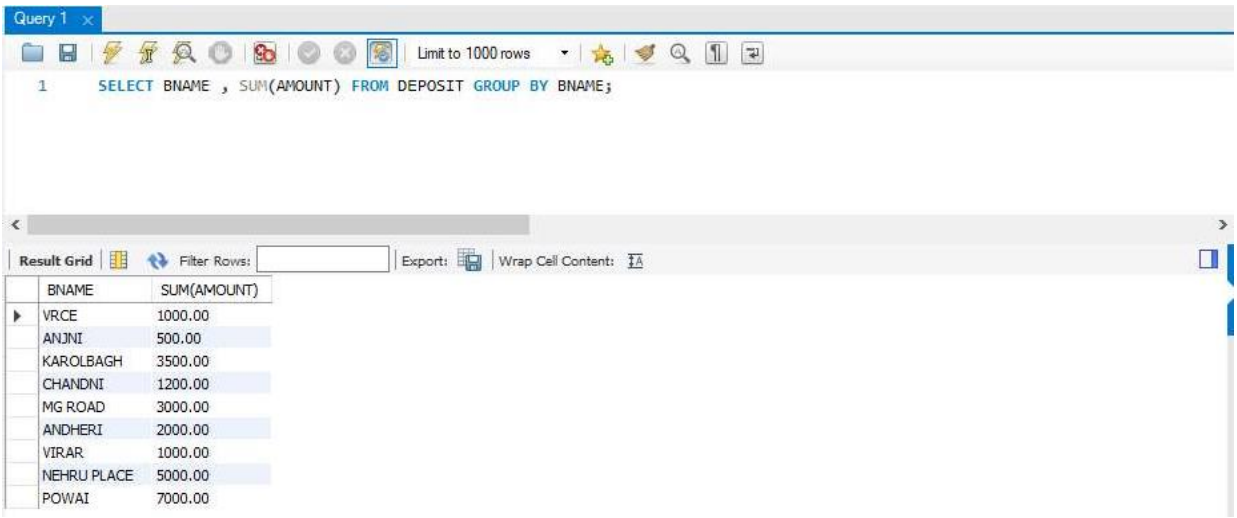
8. Count total number of branch cities



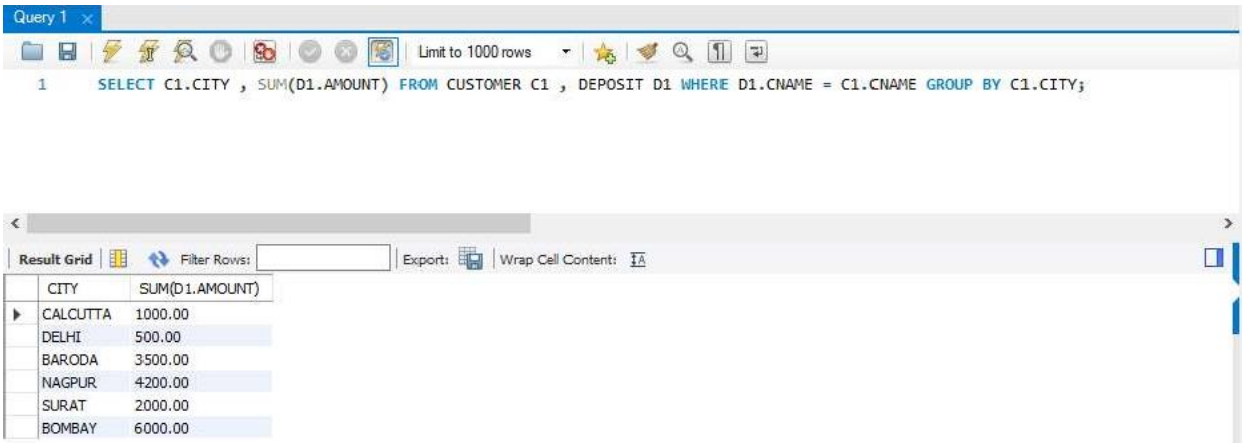
9. Count total number of customers cities



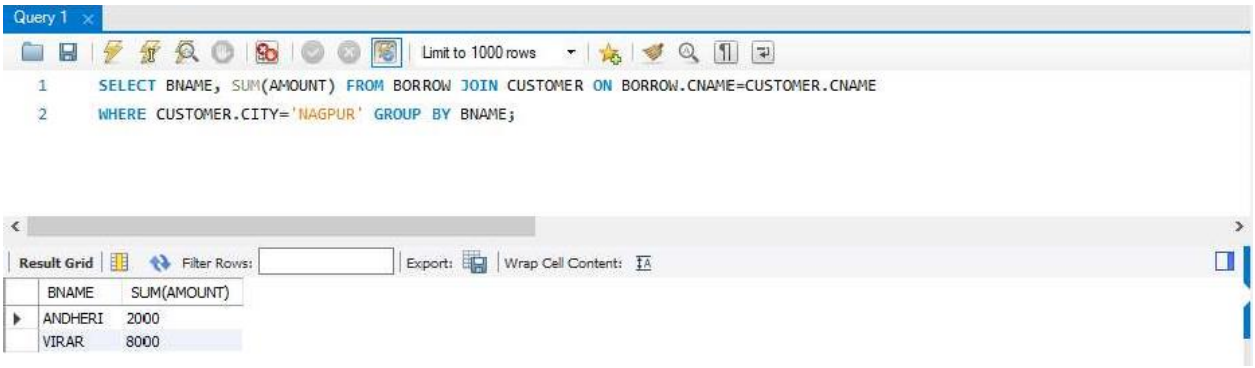
10. Give branch names and branch wise deposit



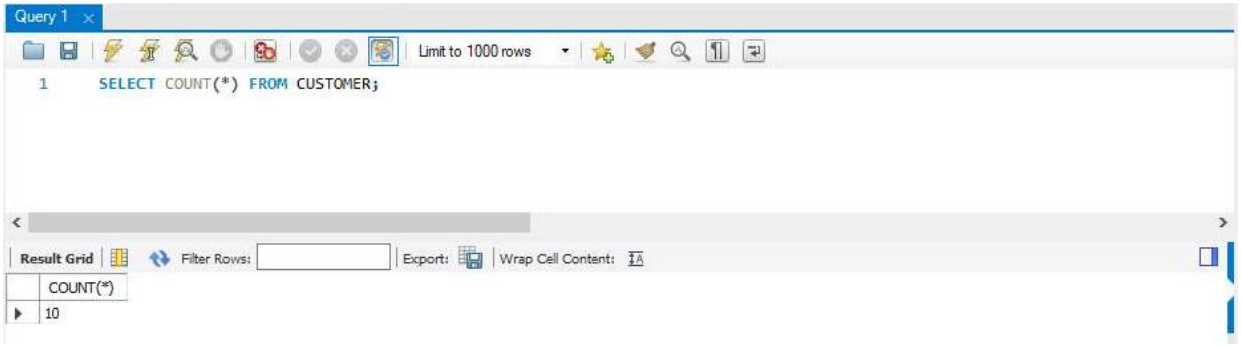
11. Give city wise name and branch wise deposit



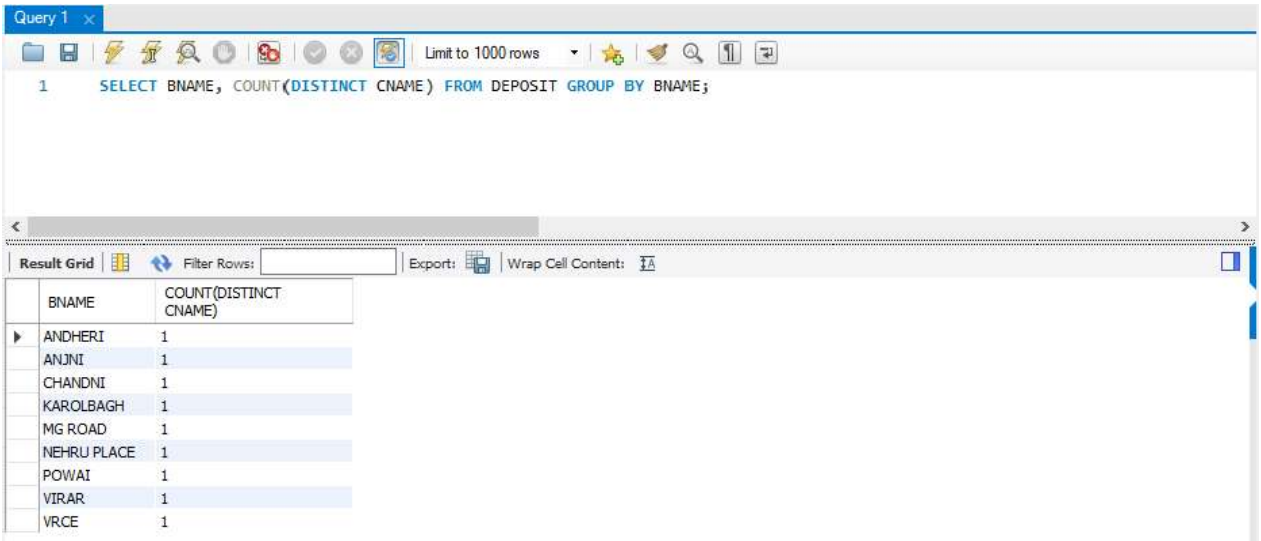
12. Give the branch wise loan of customer living in NAGPUR



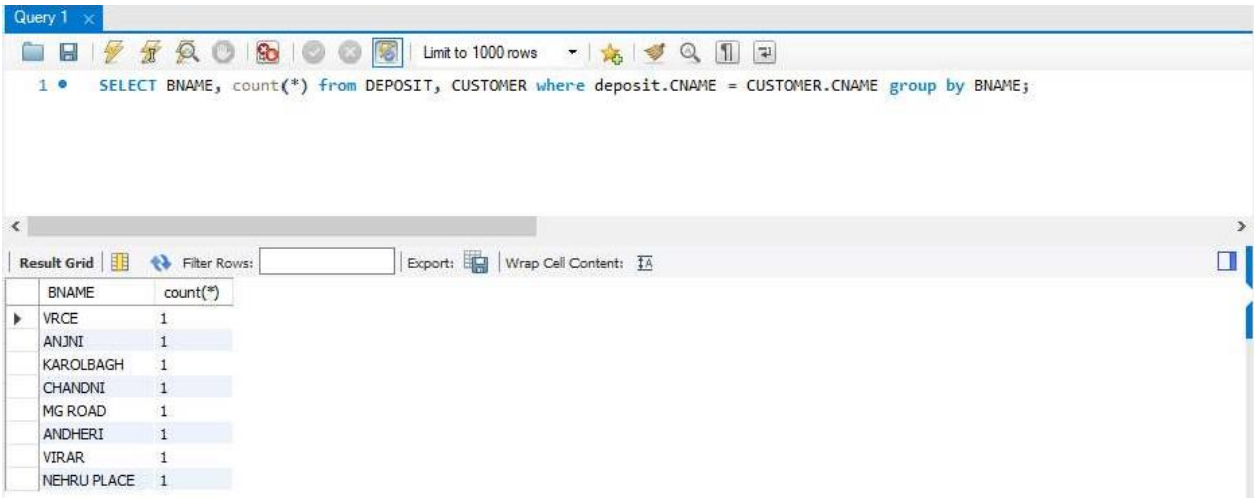
13. Count total number of customers



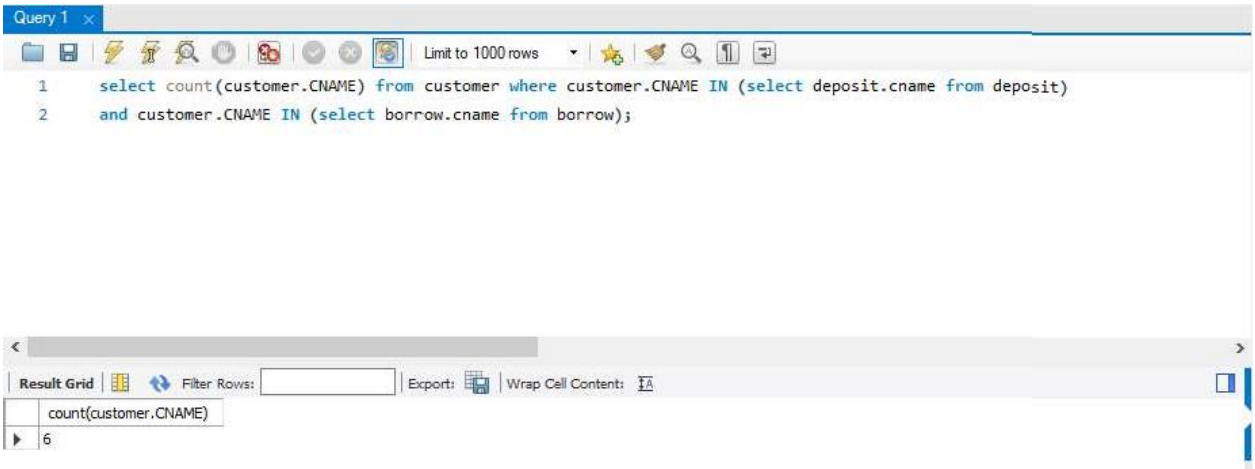
14. Count total number of depositors branch wise



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

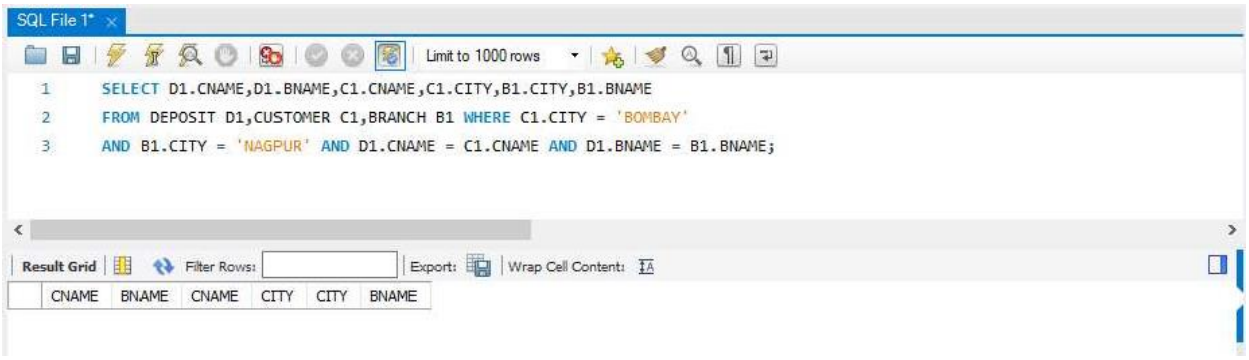
To familiarize with Join or Cartesian Product

CO1

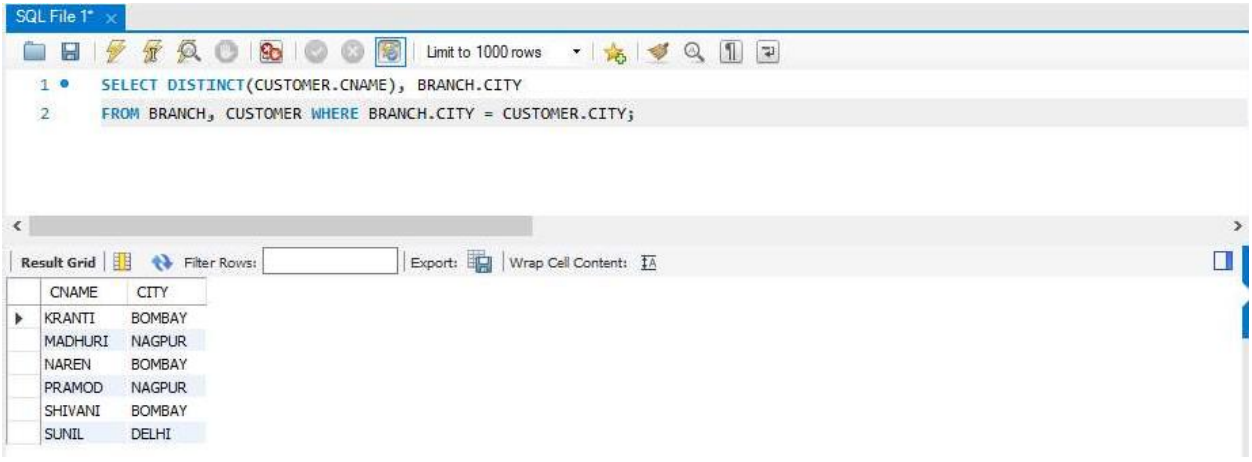
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

Procedure

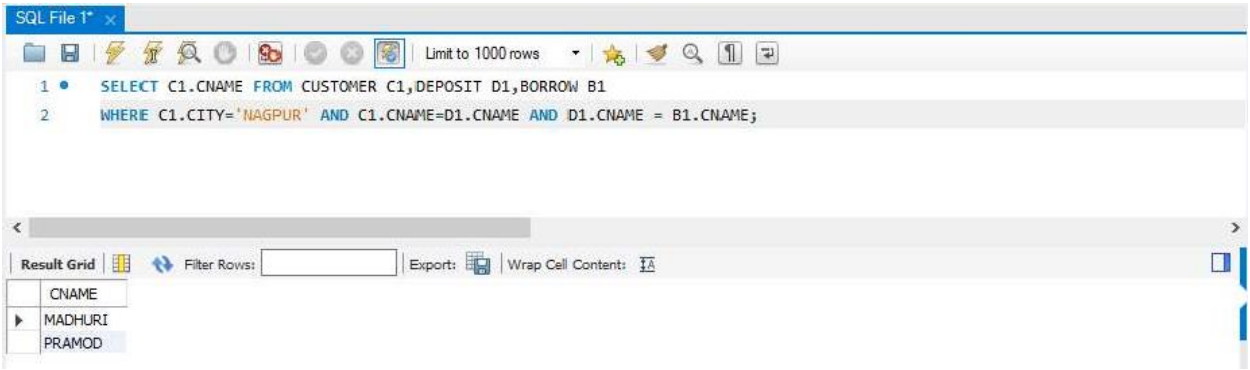
1. Give name of customers having living city BOMBAY and branch city NAGPUR



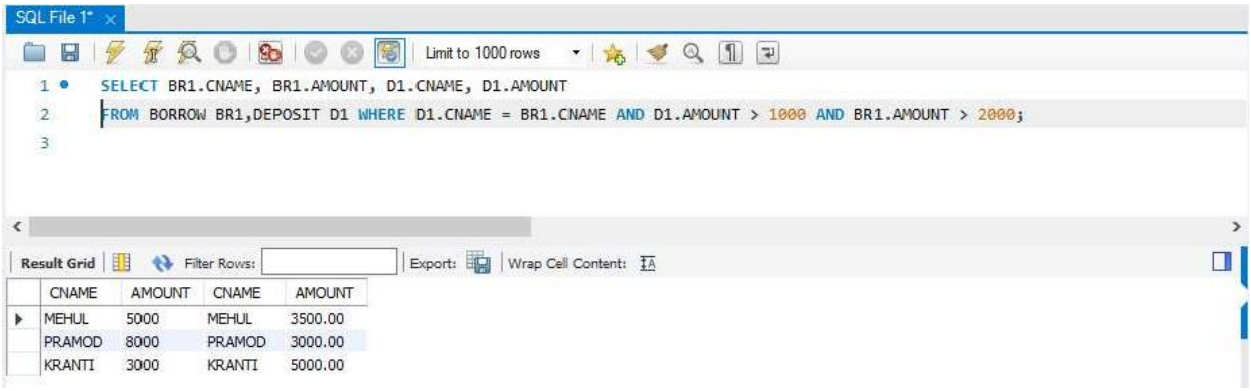
2. Give names of customers having the same living city as their branch city



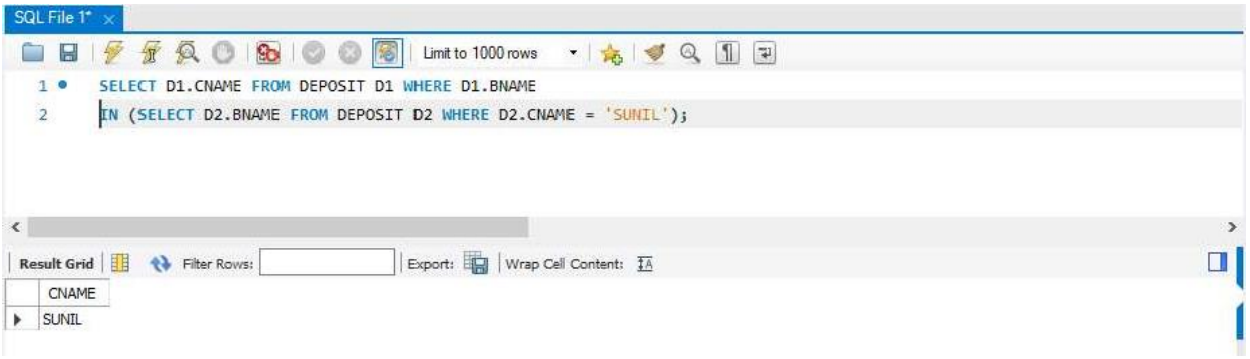
3. Give names of customers who are borrowers as well as depositors and having city NAGPUR.



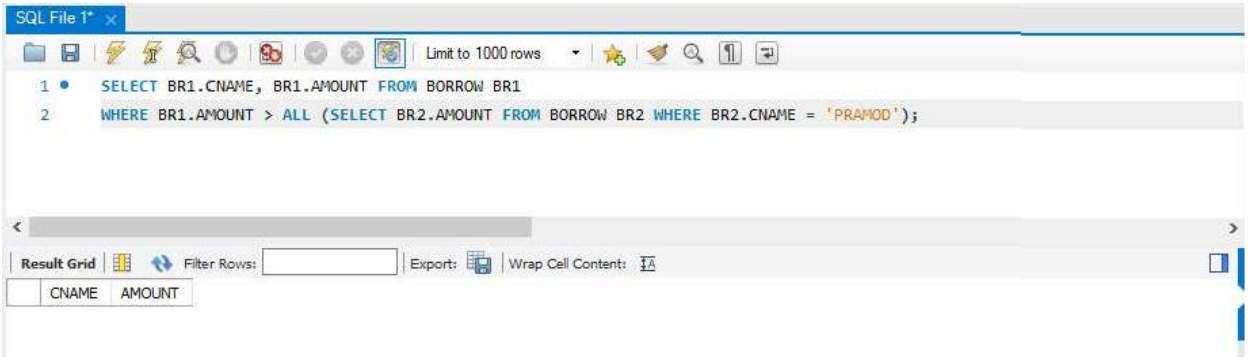
4. Give names of borrowers having deposit amount greater than 1000 and loan amount greater than 2000.



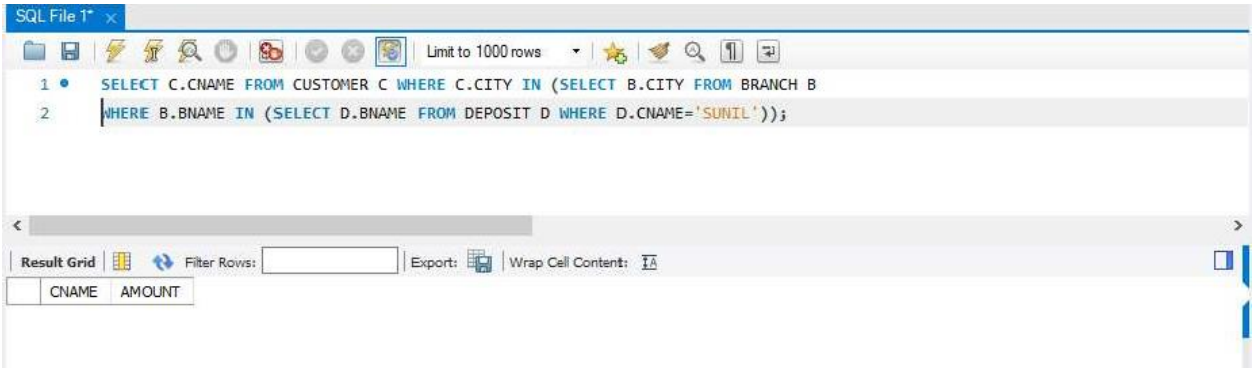
5. Give names of depositors having the same branch as the branch of Sunil



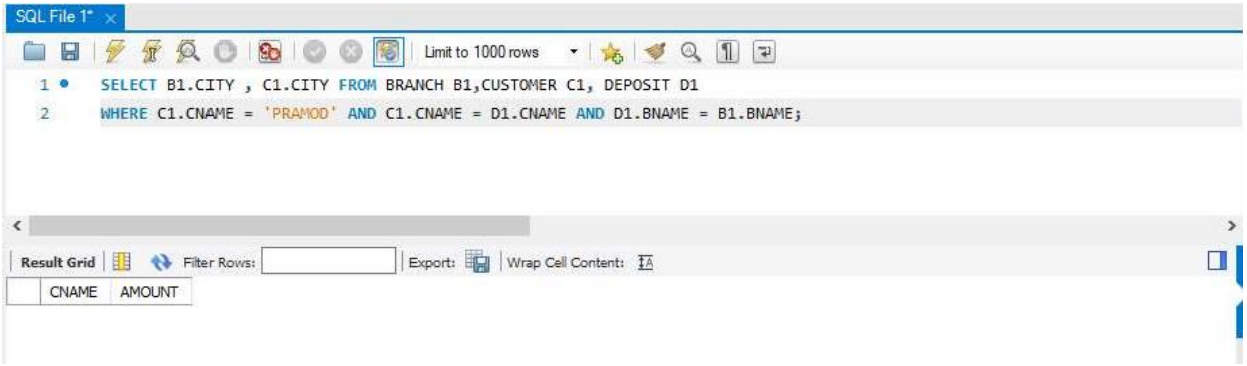
6. Give names of borrowers having loan amount greater than the loan amount of Pramod



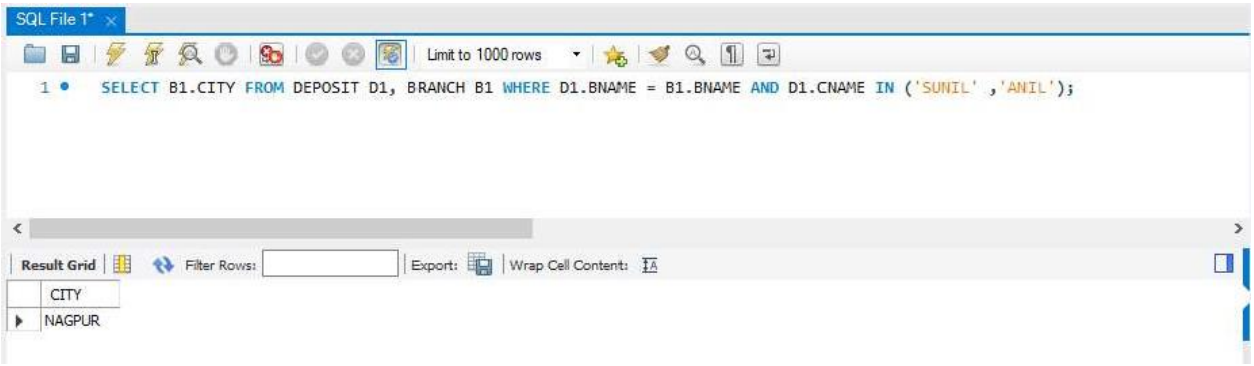
7. Give the name of the customer living in the city where branch of depositor Sunil is located.



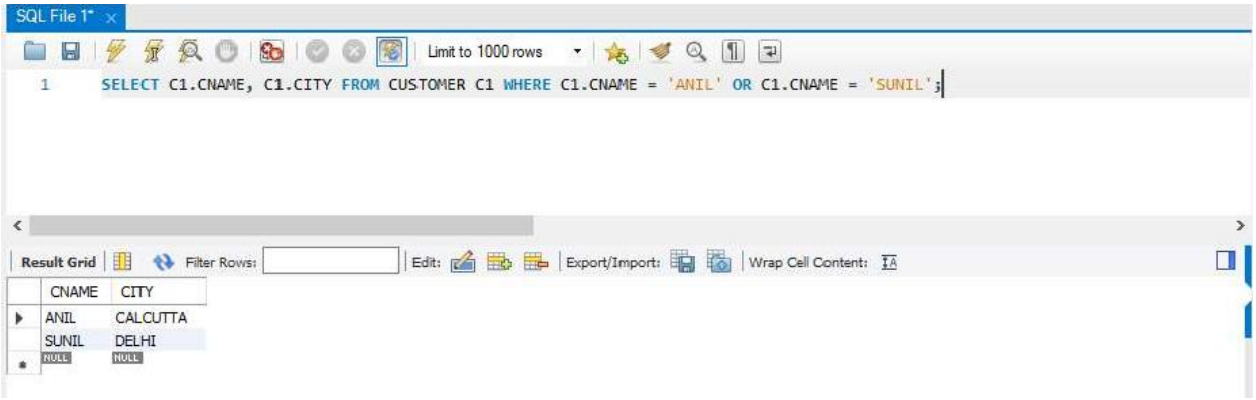
8. Give branch city and living city of Pramod



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

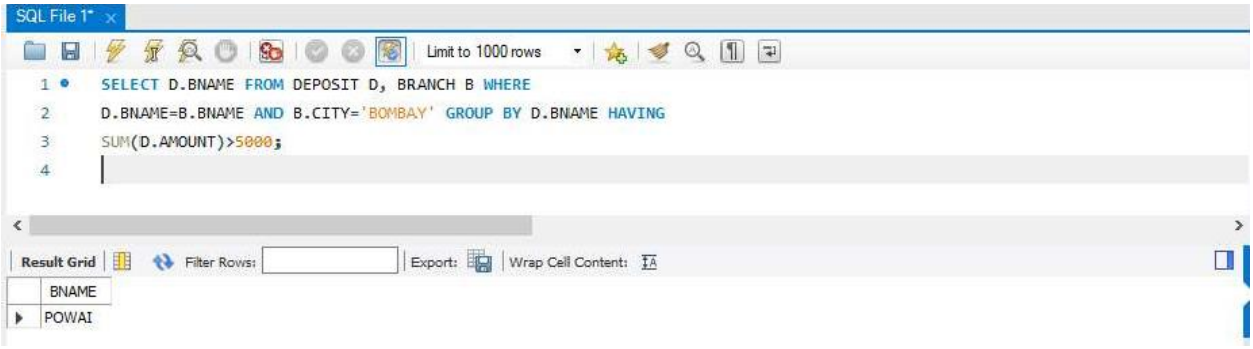
To familiarize with Group By and Having Clauses

CO1

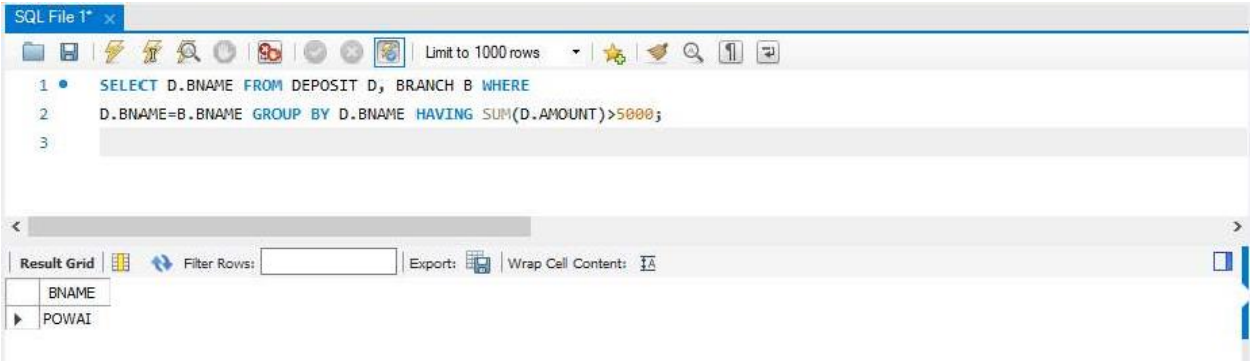
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modeling, designing and implementing a database.

Procedure

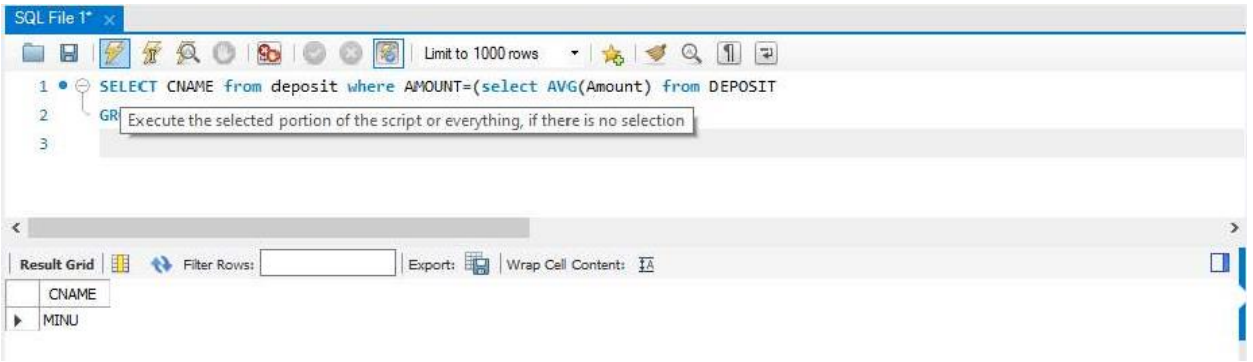
1. List the branches having sum of deposit more than 5000.



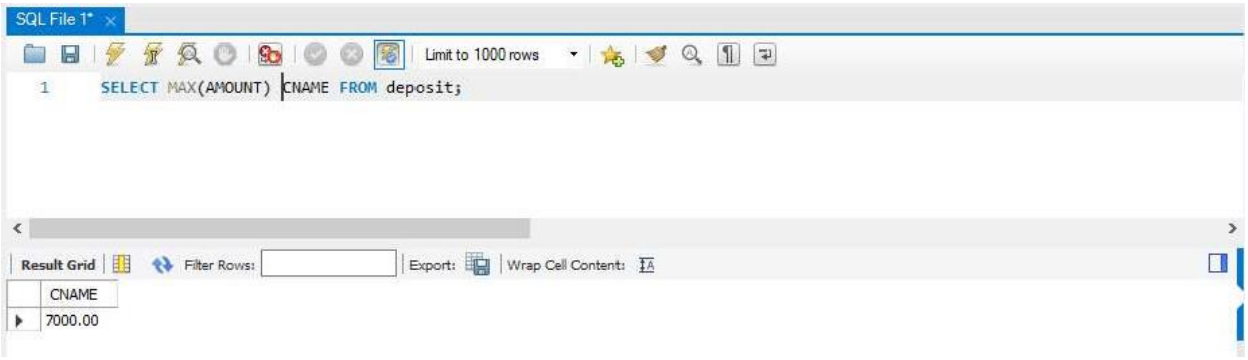
2. List the branches having sum of deposit more than 500 and located in city BOMBAY



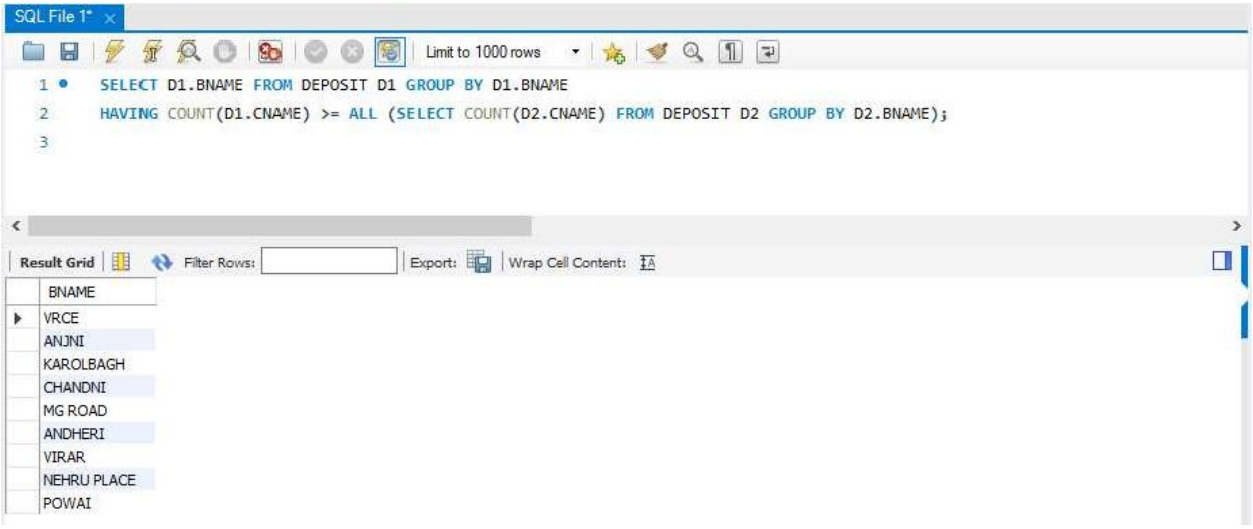
3. List the names of customers having deposited in the branches where the average deposit is more than 5000.



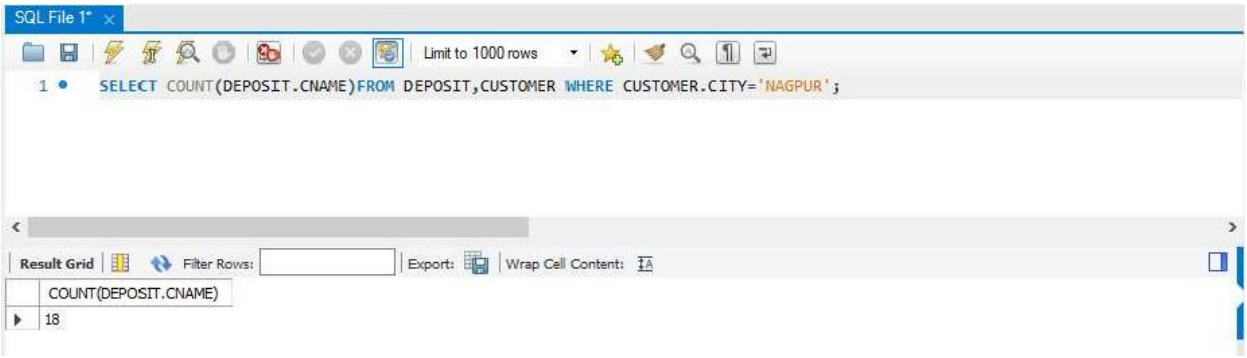
4. List the names of customers having maximum deposit



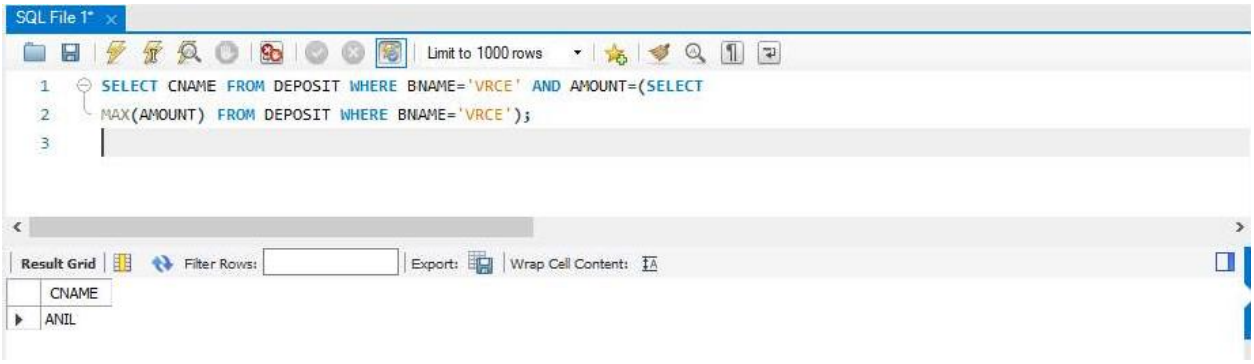
5. List the name of branch having highest number of depositors?



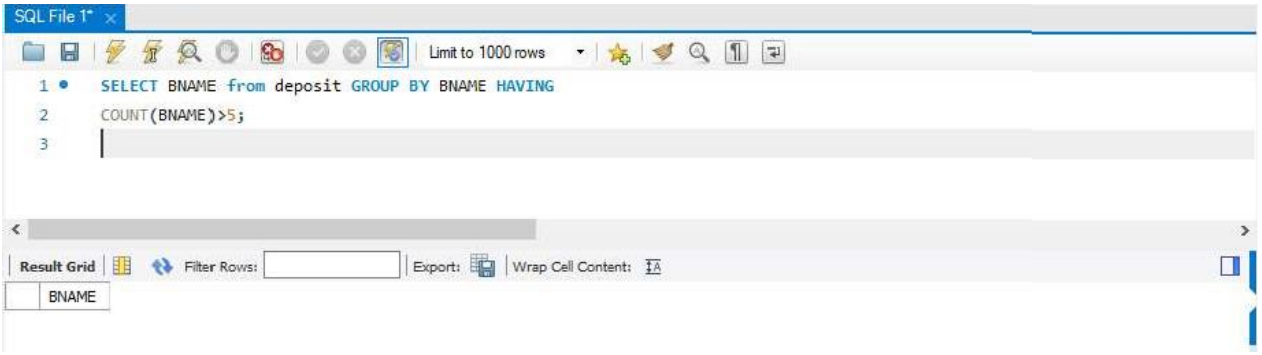
6. Count the number of depositors living in NAGPUR.



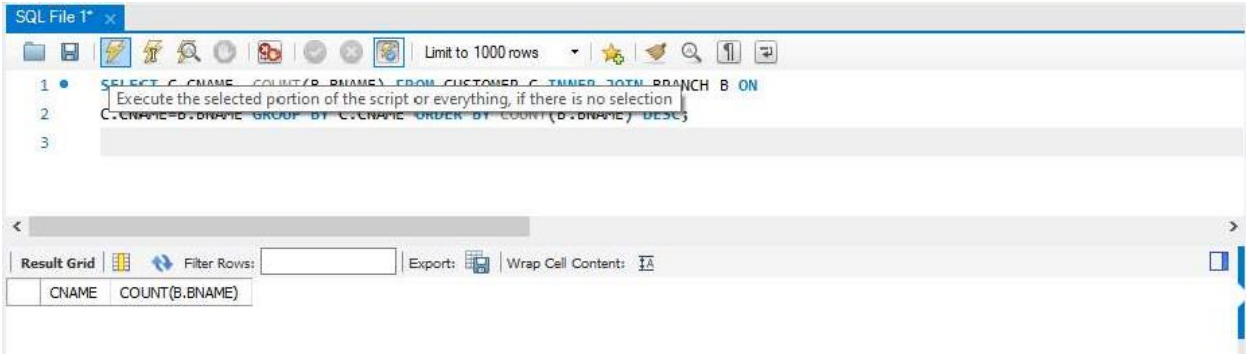
7. Give names of customers in VRCE branch having more deposite than any other customer in same branch



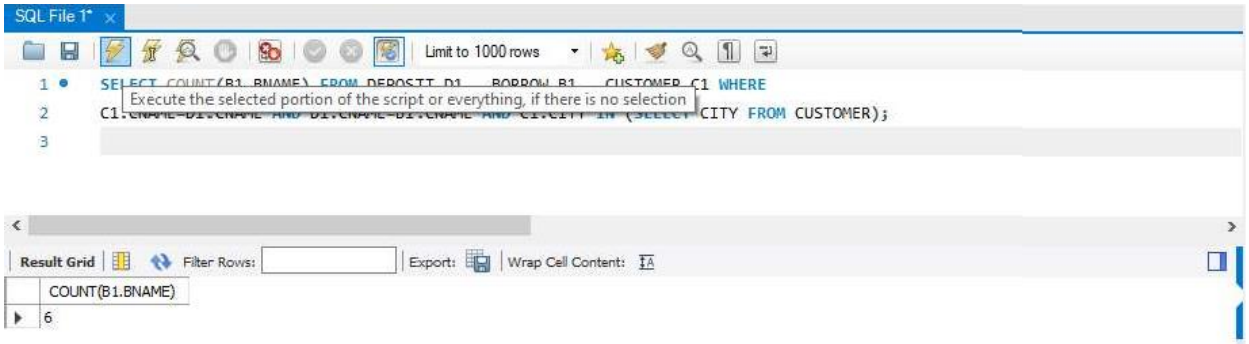
8. Give the names of branch where number of depositors is more than 5



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

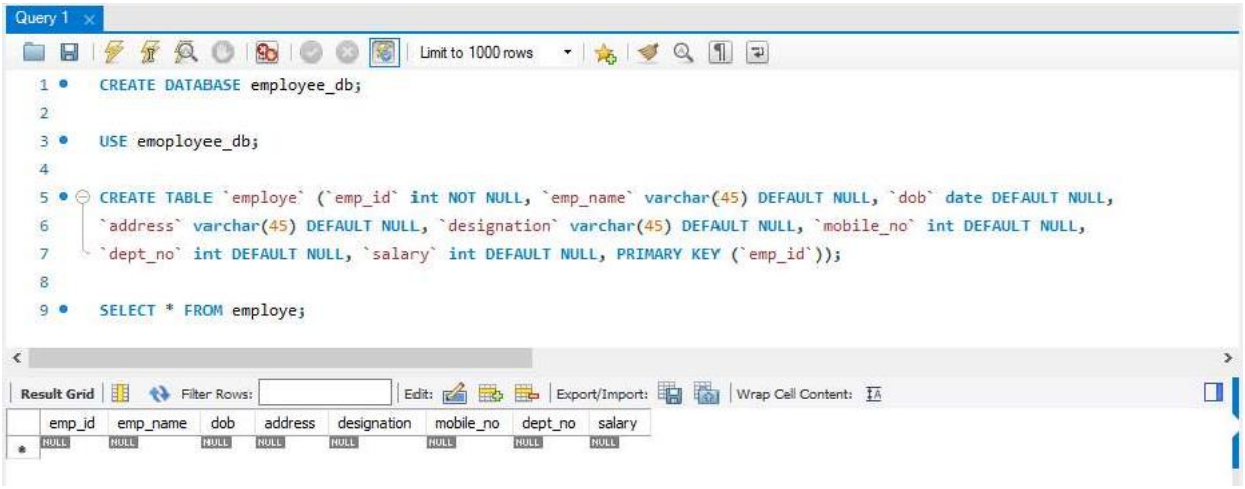
To have familiarize with trigger functions
Create a Trigger for employee table it will update another table salary while updating values

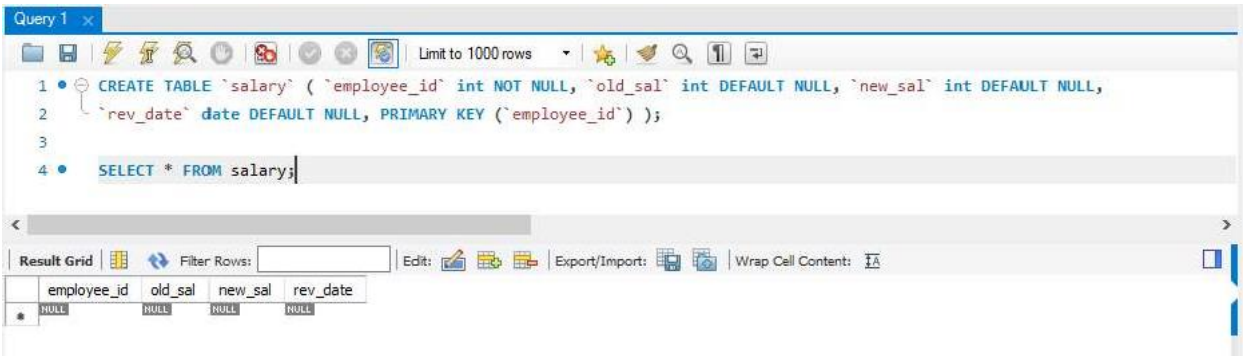
CO2

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





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;
```

salary

```
SELECT * FROM employee_db.salary;
```

Result Grid

employee_id	old_sal	new_sal	rev_date
1	234569	50000	2023-07-26
NULL	NULL	NULL	NULL

Result

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

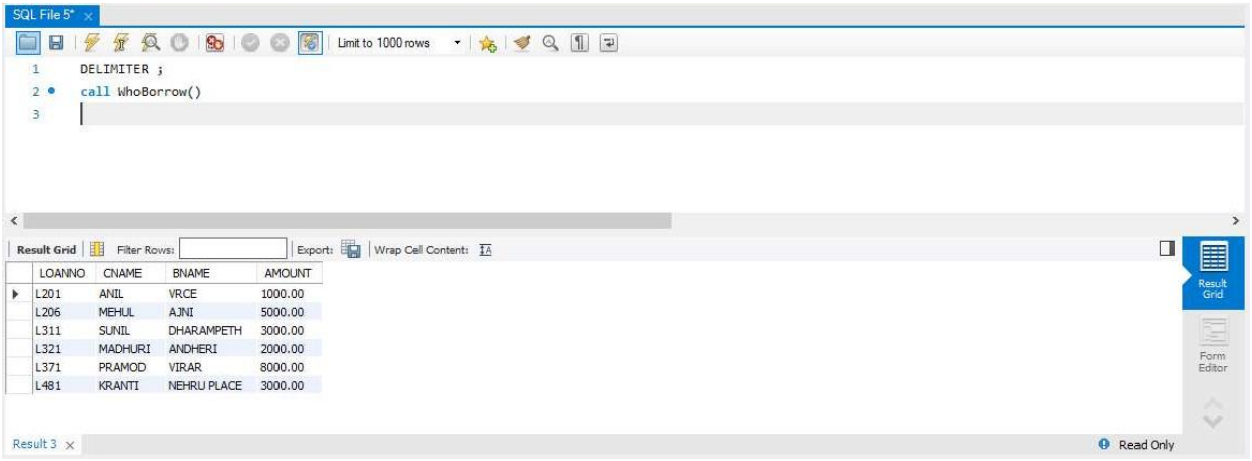
Experiment No: 9

Aim
To familiarize with Stored functions and Procedure

CO2
Apply PL/SQL for processing databases

Procedure

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

DELIMITER ;
call WhoBorrow()
```



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

Experiment No: 10

Aim

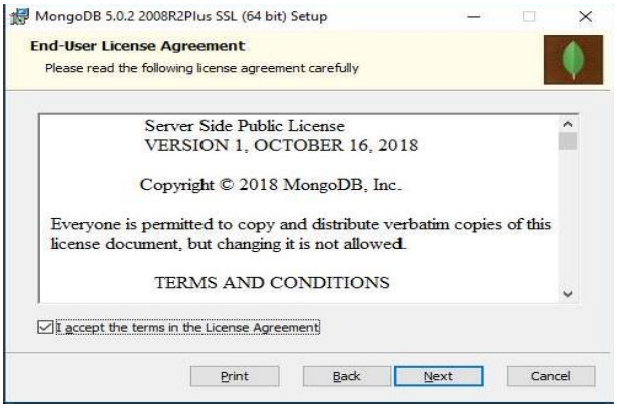
Understand the installation and configuration of NoSQL Databases: MongoDB

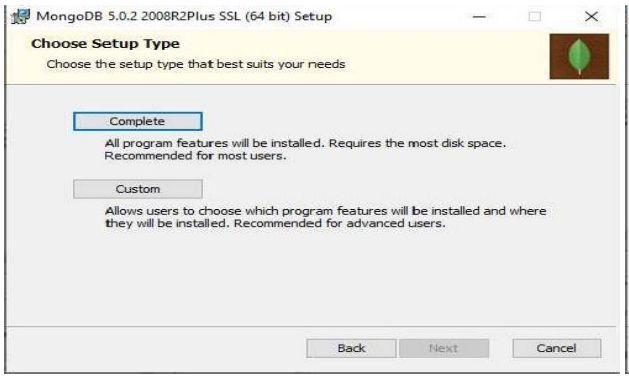
CO3

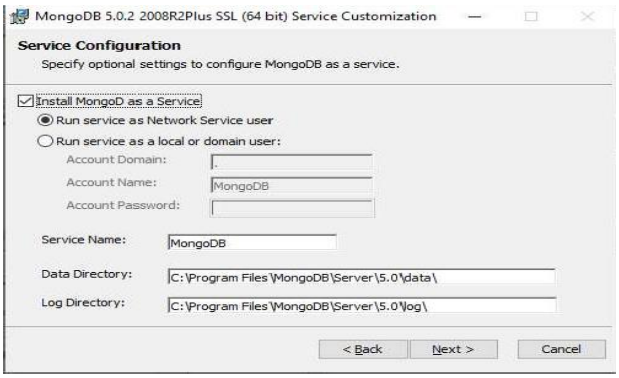
Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.

Procedure

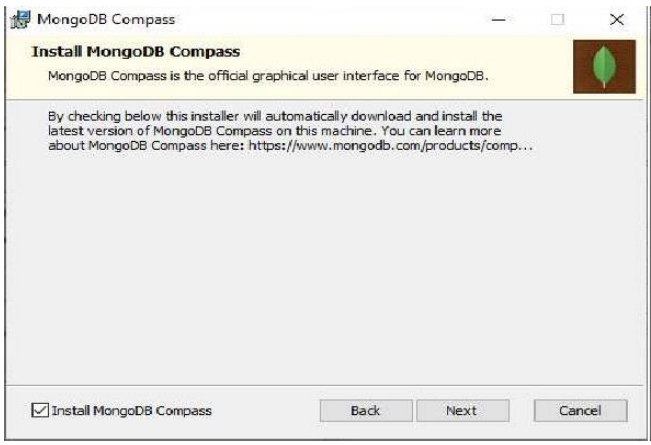
1. Setup Wizard

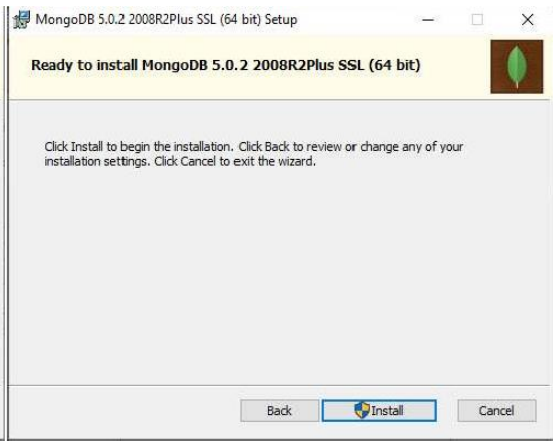


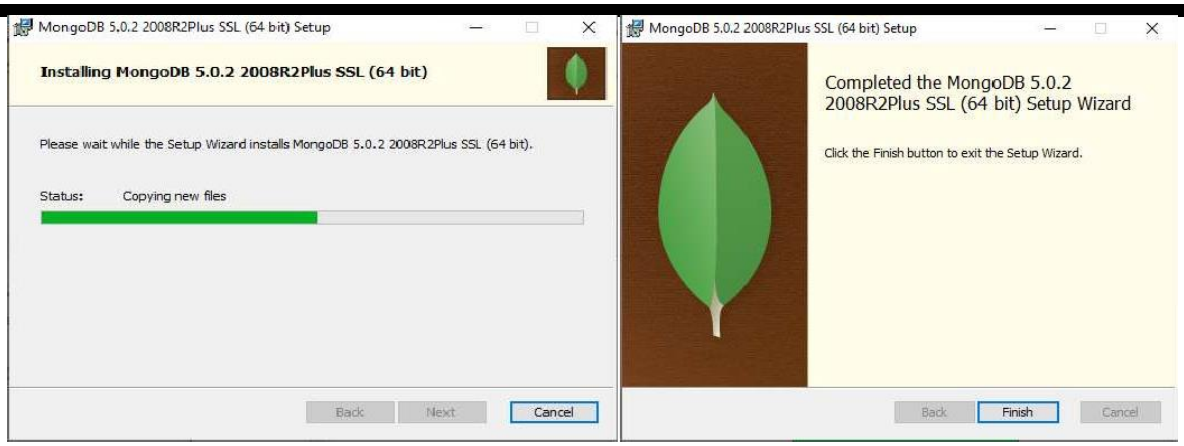




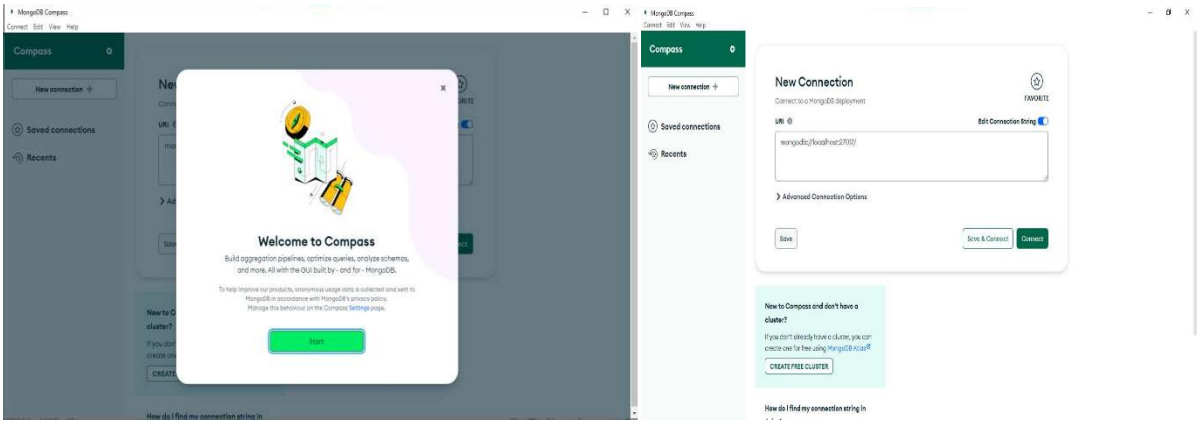
2. Installing MongoDB Compass

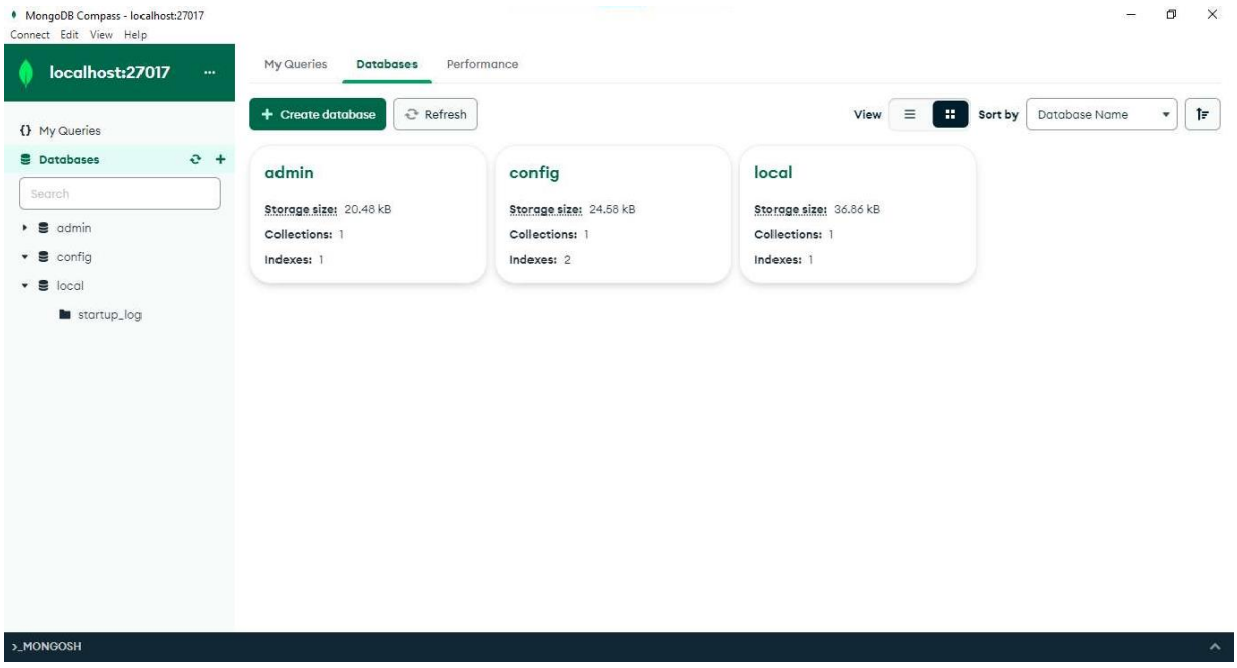




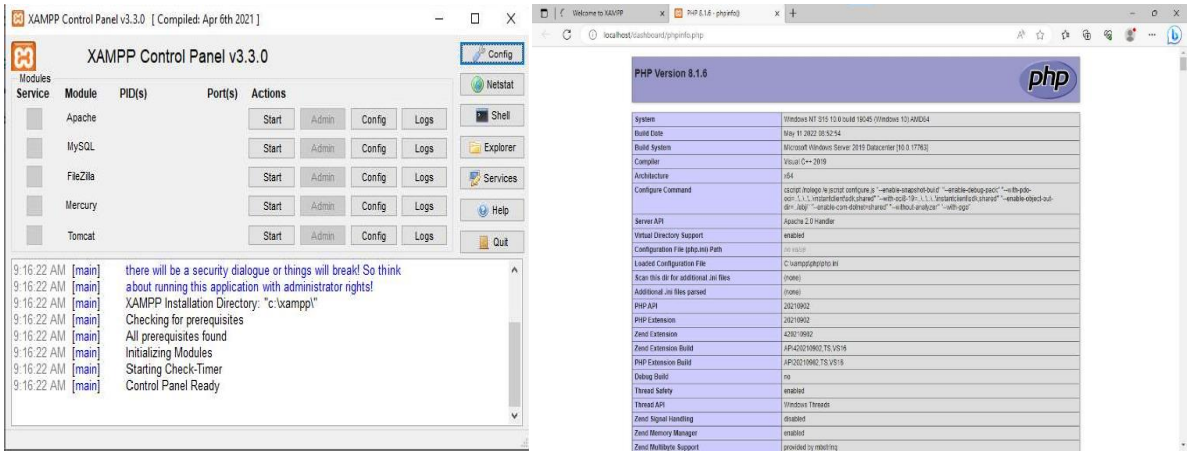


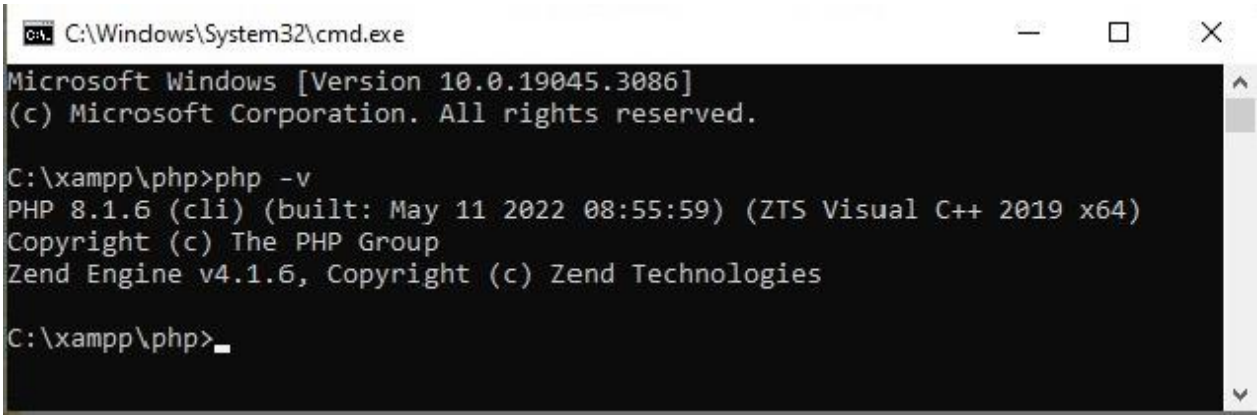
3. MongoDB Compass Home Page



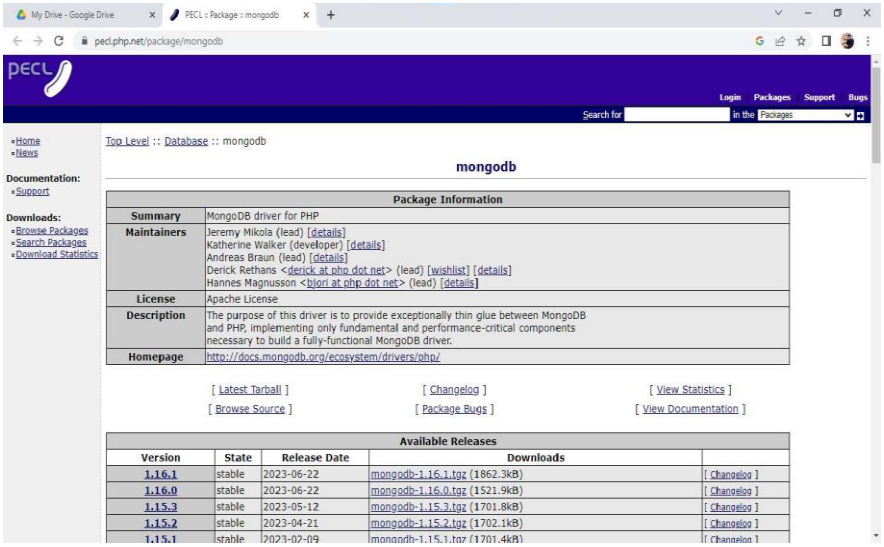


4. Configuring MongoDB to Connect to PHP

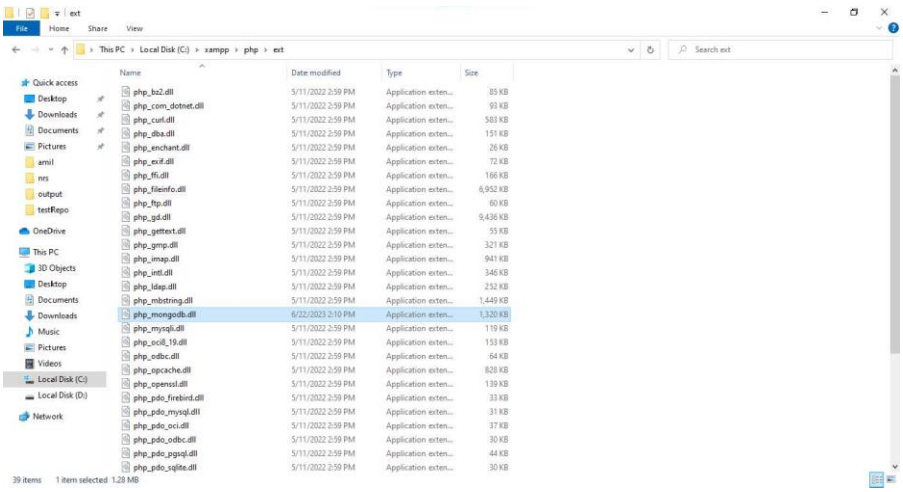


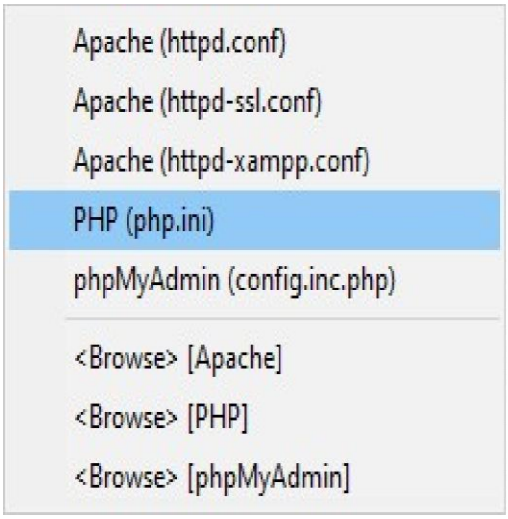


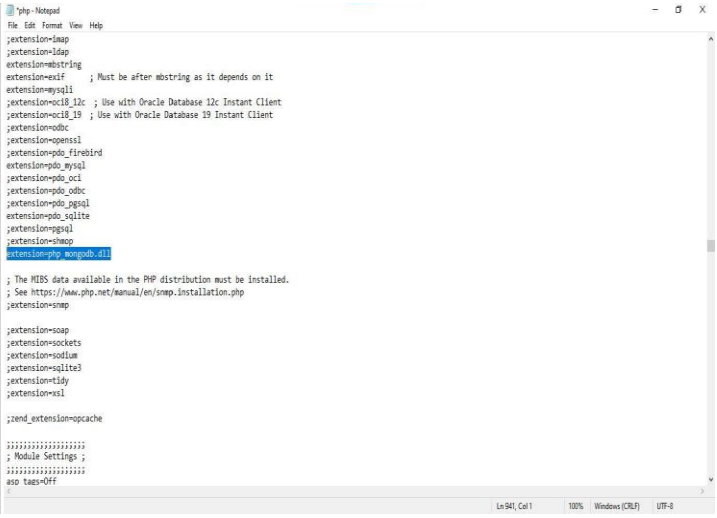
5. Installing MongoDB Package for PHP

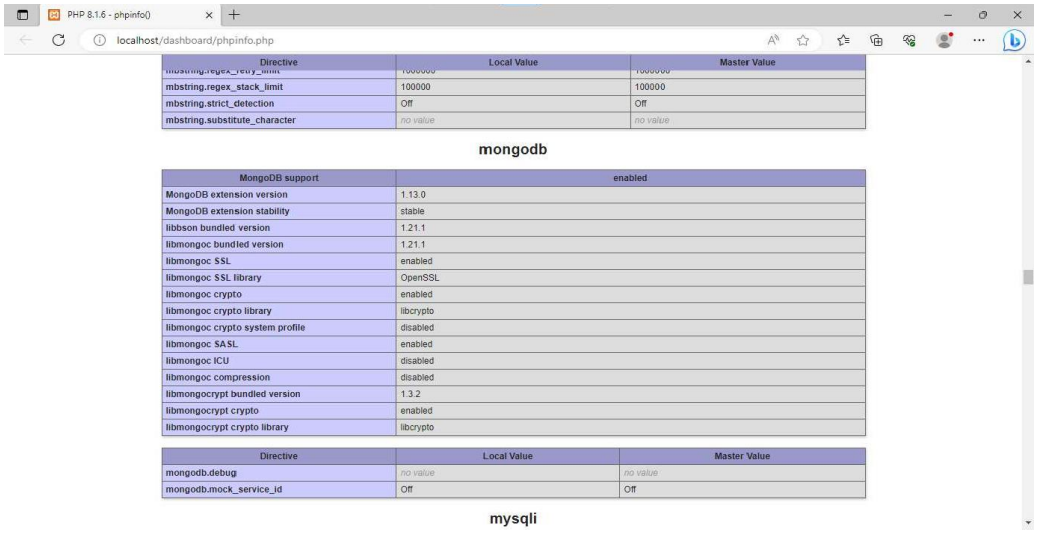


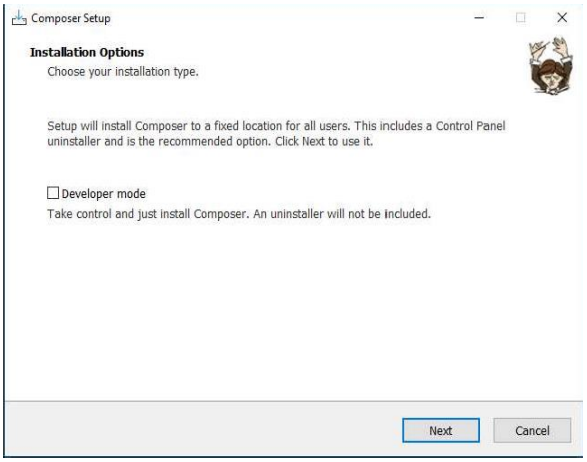
6. Configuring Apache

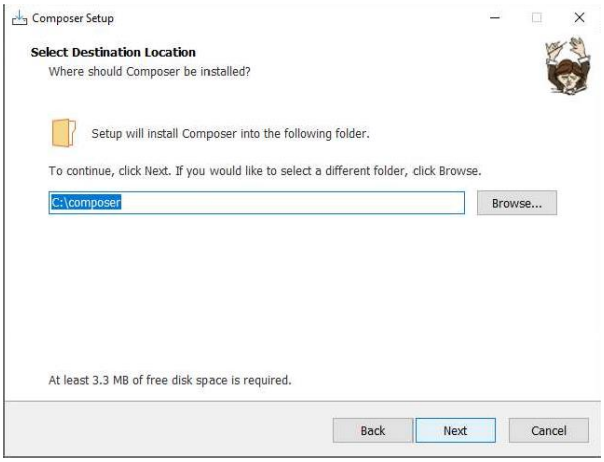


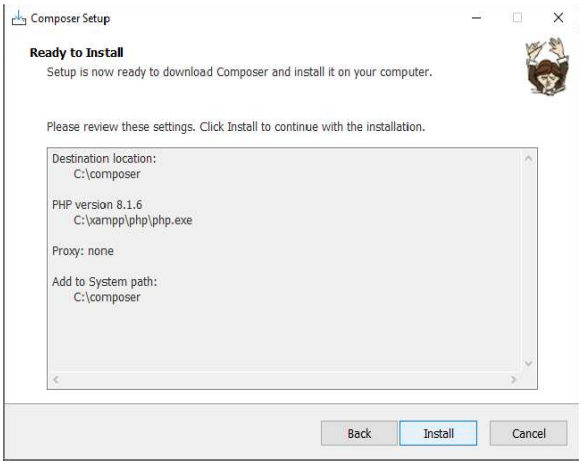


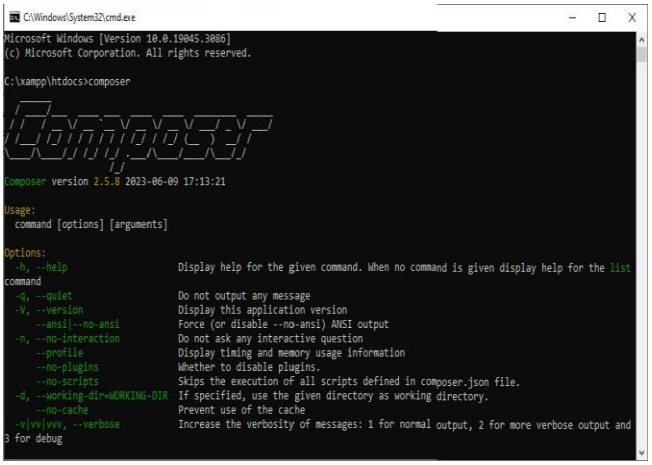




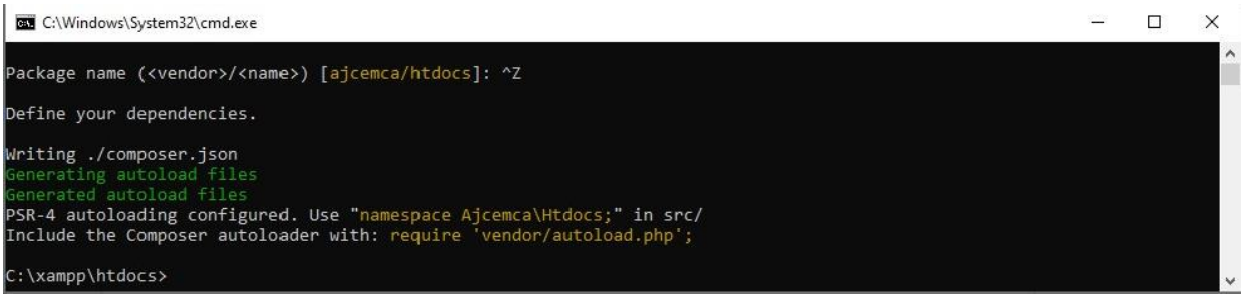

Composer Setup
Installation Options
Choose your installation type.
Setup will install Composer to a fixed location for all users. This includes a Control Panel uninstaller and is the recommended option. Click Next to use it.
Developer mode
Take control and just install Composer. An uninstaller will not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

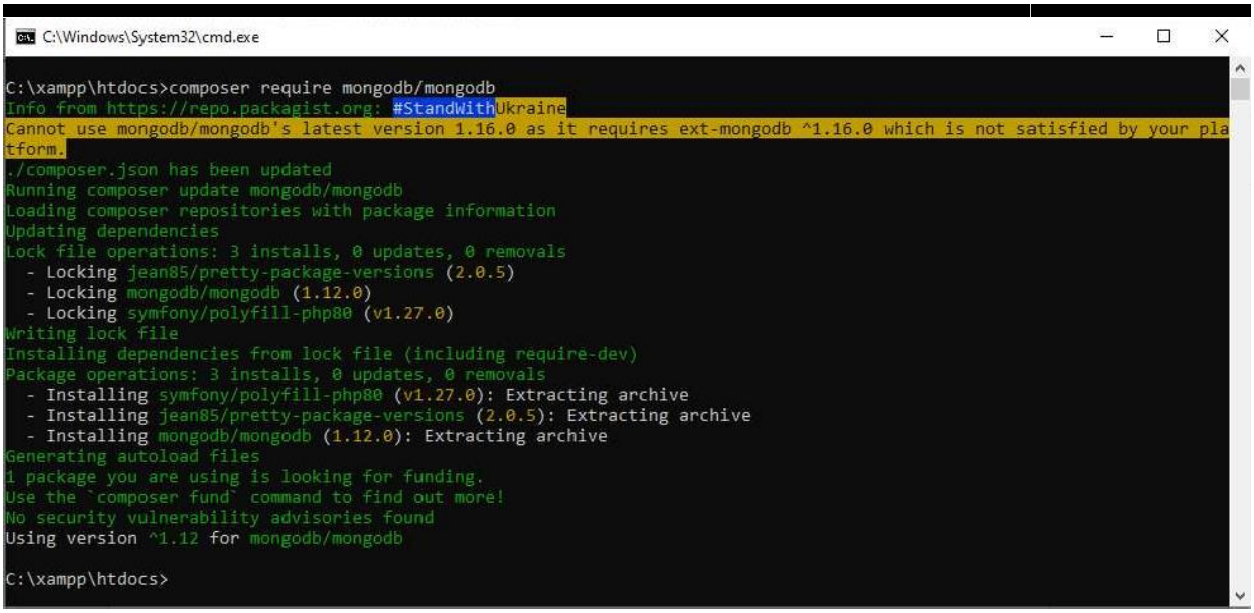

Composer Setup
Select Destination Location
Where should Composer be installed?
Setup will install Composer into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\composer
Browse...
At least 3.3 MB of free disk space is required.
Back
Next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

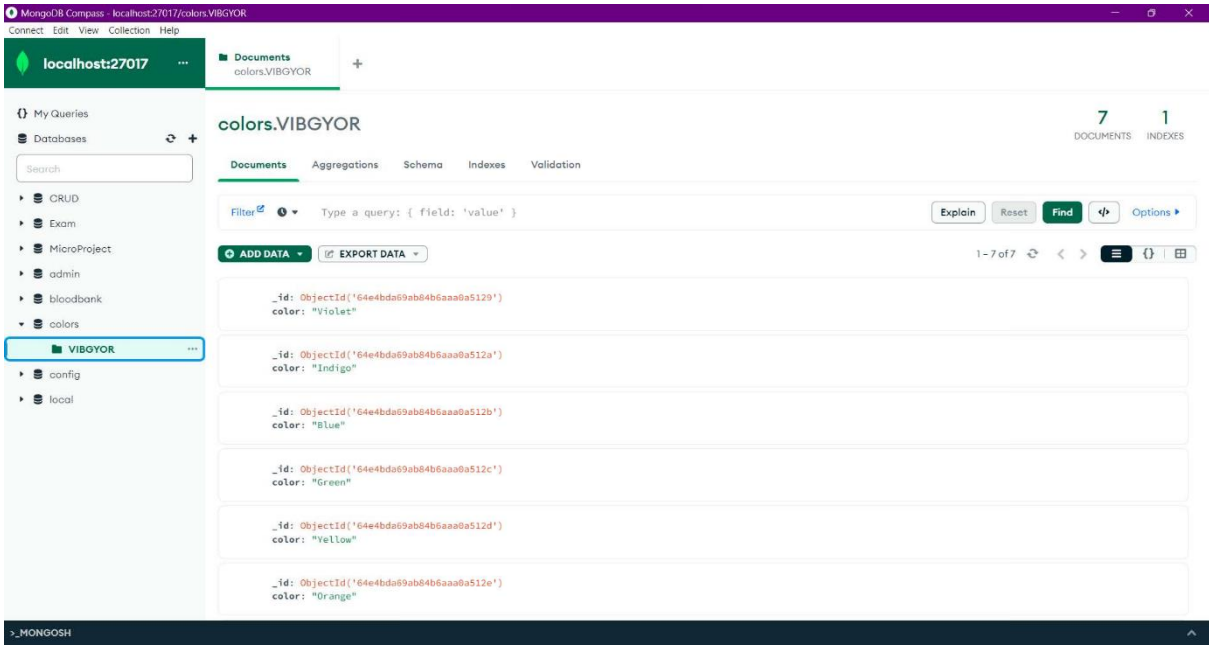
CO4

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 -> colors;
echo "Database Creation Successful !!!";
echo "<br>";
$collection = $db -> createCollection("VIBGYOR");
echo "Collection Creation Successful !!!";
echo "<br>";
$collection = $db -> VIBGYOR;
echo "Collection Selected Successful !!!";
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 "Date 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
include 'connection.php';
$result=$conn->find();
?>
<html>
<body>
<head>
<style>

</style>
</head>
<center>
<table border = 1>
<tr>
<th>NAME</th>
<th>EMAIL</th>
<th>PLACE</th>
<th>DELETE</th>
<th>EDIT</th>
</tr>
<?php
        foreach($result as $res){
?>
<tr>
        <td><?php echo $res ['name'];?></td>
        <td><?php echo $res ['email'];?></td>
```

```
        <td><?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>

</tr>
        <?php
        }
        ?>
</table>
</body>
</html>
```

INSERT.PHP

```
<?php
include 'connection.php';
if(isset($_POST['submit'])){
        $name=$_POST['name'];
        $email=$_POST['email'];
        $place=$_POST['place'];
        $doc=array("name"=>$name,"email"=>$email,"place"=>$place);
        $conn->insertOne($doc);
        echo "inserted";
        header( "Location: select.php" );
}

?>
```

Registration.php

```
<!DOCTYPE html>

<html lang="en" dir="ltr">
  <head>
    <style>
url('https://fonts.googleapis.com/css?family=Poppins:400,500,600,700&display=swap');
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Poppins', sans-serif;
```

```
}
body{
  min-height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
  background: #0e1229;
}
.wrapper{
  position: relative;
  max-width: 430px;
  width: 100%;
  background: #fff;
  padding: 34px;
  border-radius: 6px;
  box-shadow: 0 5px 10px rgba(0,0,0,0.2);
}
.wrapper h2{
  position: relative;
  font-size: 22px;
  font-weight: 600;
  color: #333;
}
.wrapper h2::before{
  content: '';
  position: absolute;
  left: 0;
  bottom: 0;
  height: 3px;
  width: 28px;
  border-radius: 12px;
  background: #4070f4;
}
.wrapper form{
  margin-top: 30px;
}
.wrapper form .input-box{
```

```
  height: 52px;
  margin: 18px 0;
}
form .input-box input{
  height: 100%;
  width: 100%;
  outline: none;
  padding: 0 15px;
  font-size: 17px;
  font-weight: 400;
  color: #333;
  border: 1.5px solid #C7BEBE;
  border-bottom-width: 2.5px;
  border-radius: 6px;
  transition: all 0.3s ease;
}
.input-box input:focus,
.input-box input:valid{
  border-color: #4070f4;
}
form .policy{
  display: flex;
  align-items: center;
}
form h3{
  color: #707070;
  font-size: 14px;
  font-weight: 500;
  margin-left: 10px;
}
.input-box.button input{
  color: #fff;
  letter-spacing: 1px;
  border: none;
  background: #5540f4;
  cursor: pointer;
}
```

```
.input-box.button input:hover{
  background: #0e4bf1;
}
form .text h3{
 color: #333;
 width: 100%;
 text-align: center;
}
form .text h3 a{
  color: #4070f4;
  text-decoration: none;
}
form .text h3 a:hover{
  text-decoration: underline;
}
    </style>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Registration or Sign Up form in HTML CSS | CodingLab </title>
    <link rel="stylesheet" href="style.css">
   </head>
<body>
  <div class="wrapper">
    <h2>Registration</h2>
    <form method="post" action="insert.php">
      <div class="input-box">
        <input type="text" name="name" placeholder="Enter your name"  required>
      </div
      <div class="input-box">
        <input type="Email" name="email" placeholder="Enter your email" required>
      </div
      <div class="input-box">
        <input type="text" name="place" placeholder="Place" required>
      </div
      <div class="input-box button">
        <input type="submit" name="submit">
      </div>
```

```
    <div class="text">
      <h3>Already have an account? <a href="insert.php">Login now</a></h3>
    </div>
   </form>
  </div>
</body>
</html>
```

Select.php

```
<?php
include 'connection.php';
$result=$conn->find();
?>
<html>
<body>
<head>
<style>


</style>
</head>
<center>
<table border = 1>
<tr>
<th>NAME</th>
<th>EMAIL</th>
<th>PLACE</th>
<th>DELETE</th>
<th>EDIT</th>
</tr>
<?php
        foreach($result as $res){
?>
<tr>
        <td><?php echo $res ['name'];?></td>
        <td><?php echo $res ['email'];?></td>
        <td><?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}?>
</table>
</body>
</html>
```

Delete.php

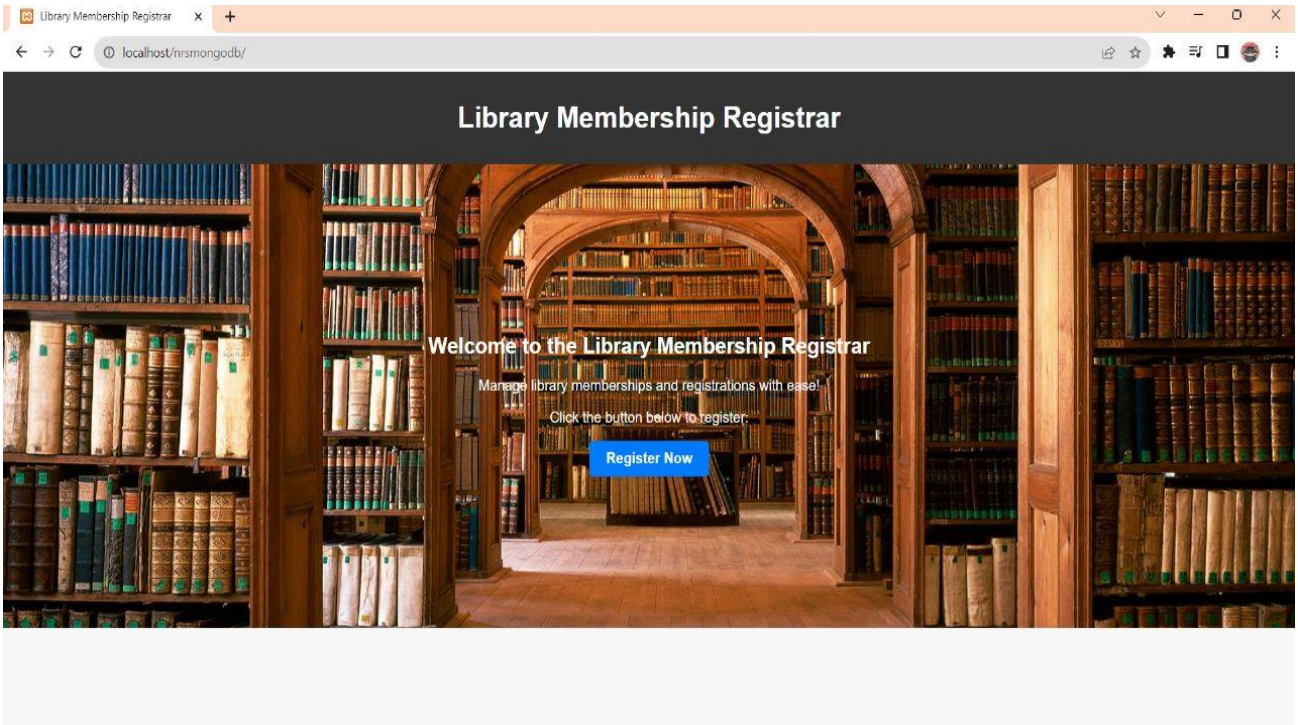
```
<?php
include 'connection.php';
$id = $_GET['id'];
$conn->deleteOne(['_id' => new MongoDB\BSON\ObjectID($id)]);
header( "Location: select.php" );
?>
```

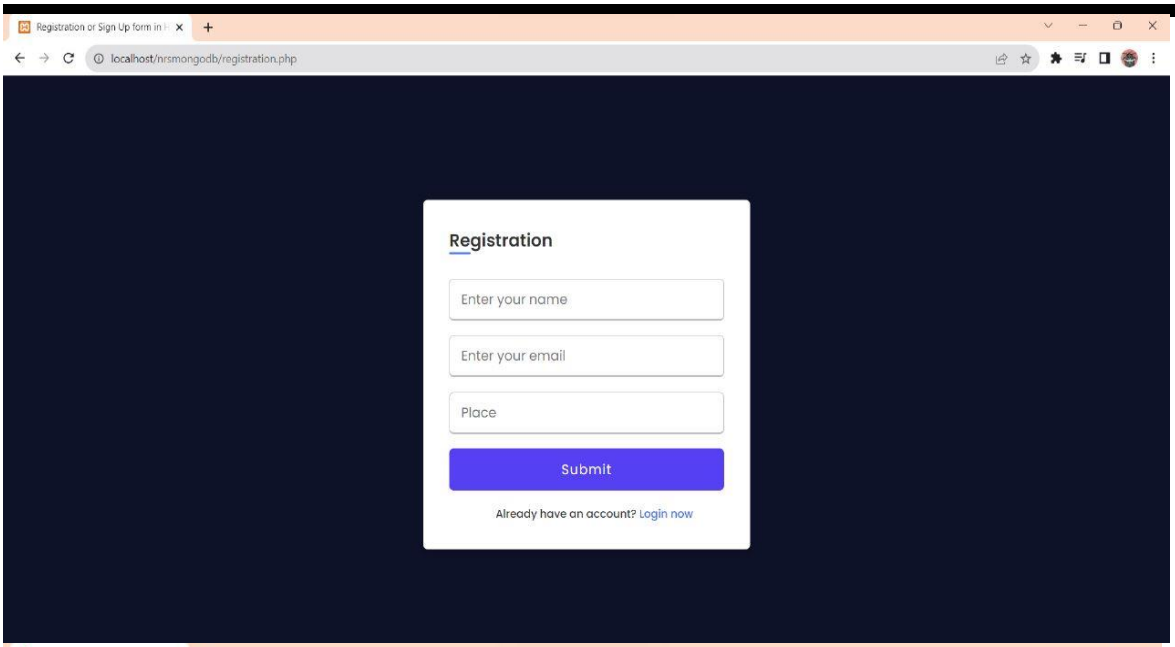
Edit.php

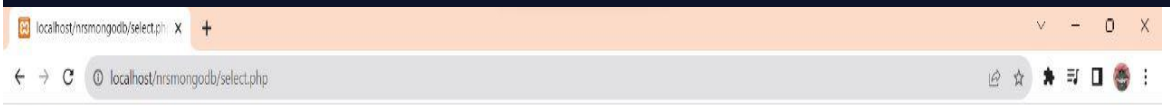
```
<?php
include "connection.php";

$id = $_GET['id'];
$objectId = new MongoDB\BSON\ObjectID($id);
$item = $conn->findOne(['_id' => $objectId]);

if (isset($_POST['submit']))
{
   $name = $_POST['name'];
   $email = $_POST['email'];
        $place = $_POST['place'];
        $updateData = ['$set' => ['name' => $name, 'email' => $email,'place' =>$place]];

        $myid = ['_id' => $objectId];

        $result = $conn->updateOne($myid, $updateData);
        header("Location: select.php");
}?>
<html>
<head>
   <title>Form </title>
        <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
```

```
<center>
    <h1>Registration Form</h1>
    <form method="POST" action="#">
        <label for="name">Full Name:</label>
        <input type="text" value="<?php echo $item["name"]; ?>" name="name" id="name"
required><br><br>
        <label for="email">Email:</label>
        <input type="email"  value="<?php echo $item["email"]; ?>" name="email" id="email"
required><br><br>
<label for="name">Place:</label>
        <input type="text"  value="<?php echo $item["place"]; ?>" name="place" id="place"
required><br><br>
<input type="submit" name="submit" value="Submit">
    </form>
        <a href="viewdata.php">Go Back</a>
        </center>
</body>
</html>
```

OUTPUT







admin.sample

NAME	EMAIL	PLACE	DELETE	EDIT
Amil Mary Joseph	amil@gmail.com	Idukki	Delete	Edit
Anina Elizabeth	anina@gmail.com	Thrissur	Delete	Edit
Anitta Rose	anitta@gmail.com	Pathanmthitta	Delete	Edit
Chithira	cb@gmail.com	Wayanad	Delete	Edit
Gayathri	gayu@gmail.com	Kottayam	Delete	Edit
Nandana	nandana@gmail.com	Trivandrum	Delete	Edit
Sanila	san@gmail.com	Thrissur	Delete	Edit
SnehaElsa	elsa@gmail.com	Calicut	Delete	Edit
Neha	neha@gmail.com	Kannur	Delete	Edit
Nandana R.S	nandana@gmail.com	Trivandrum	Delete	Edit
Shruthy	sru@gmail.com	Trivandrum	Delete	Edit

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

Apply CRUD operations and retrieve data in a NoSQL environment.

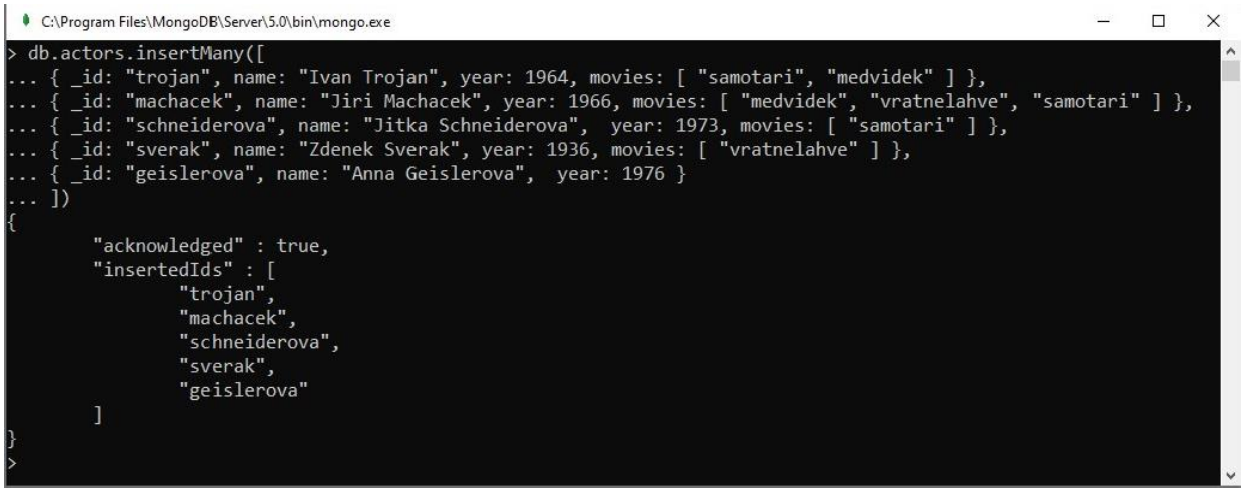
Procedure

1. Create/Use a database
   > use expmongo

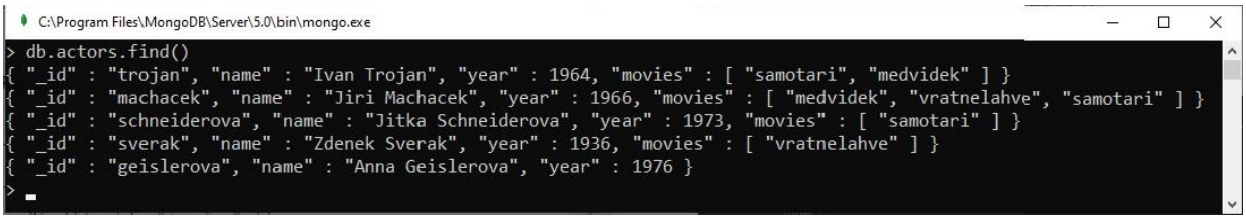
2. Display current database
   >db

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

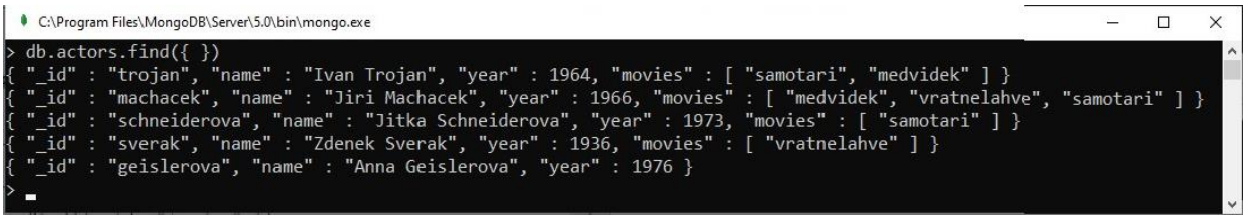
4. Insert data into the collection



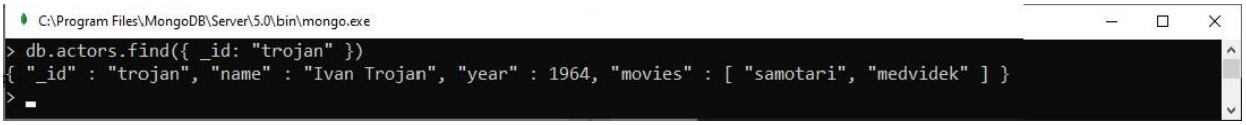
5. Display documents in collection
   >db.actors.find()



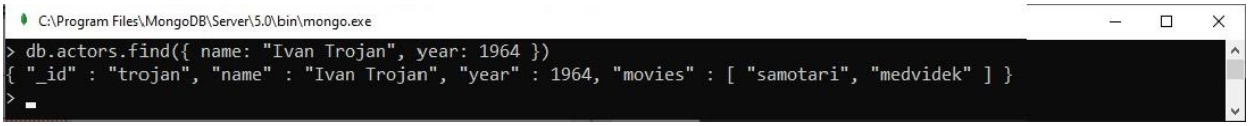
6. Display documents in collection
   >db.actors.find({ })



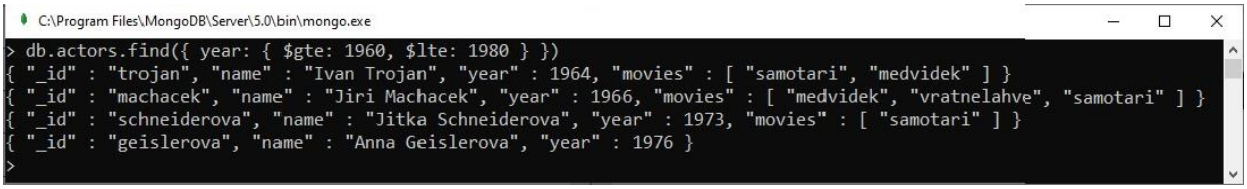
7. Display
   >db.actors.find({ _id: "trojan" })



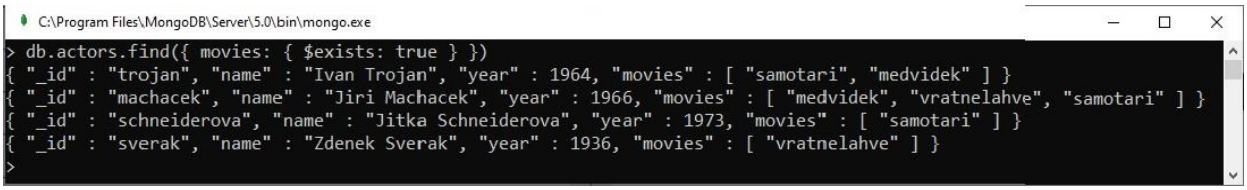
8. Display
   >db.actors.find({ name: "Ivan Trojan", year: 1964 })



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

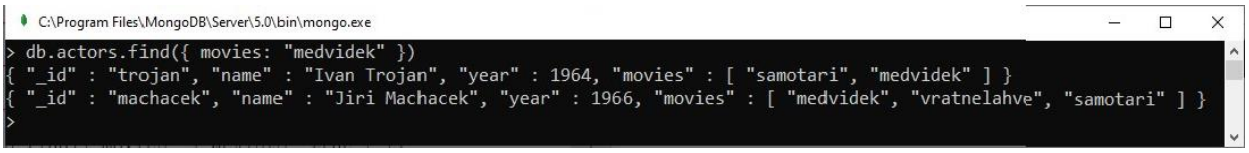


10. Display
    >db.actors.find({ movies: { $exists: true } })



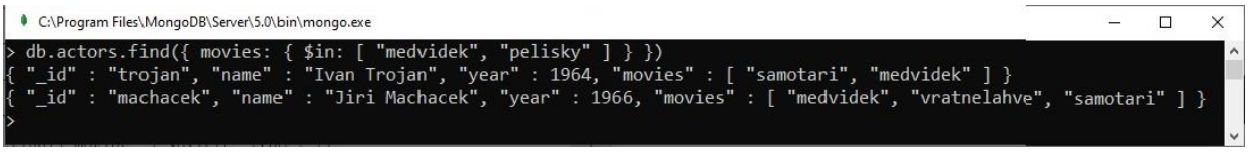
11. Display

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



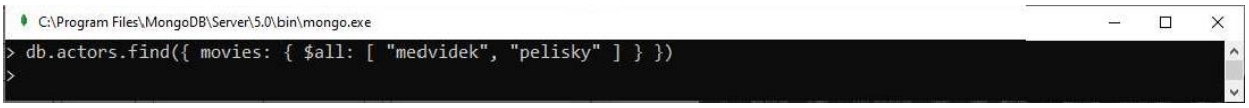
12. Display

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



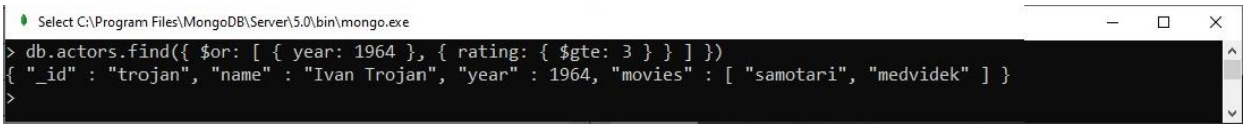
13. Display

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



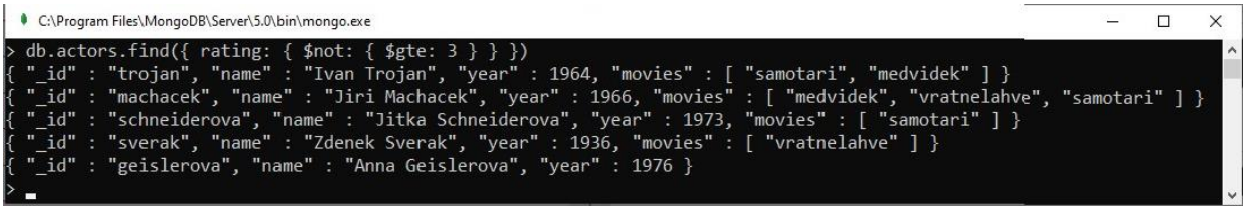
14. Display

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



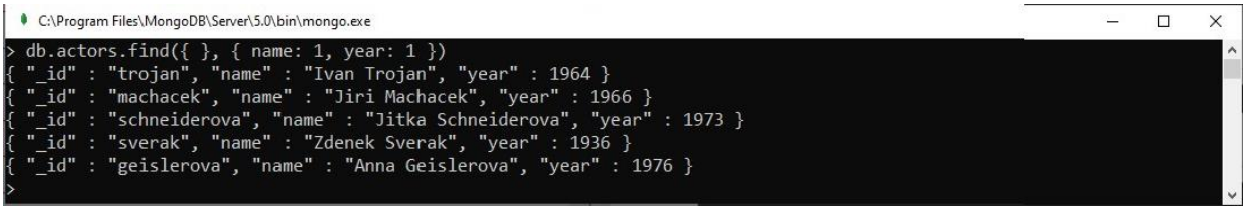
15. Display

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



16. Display

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



17. Display

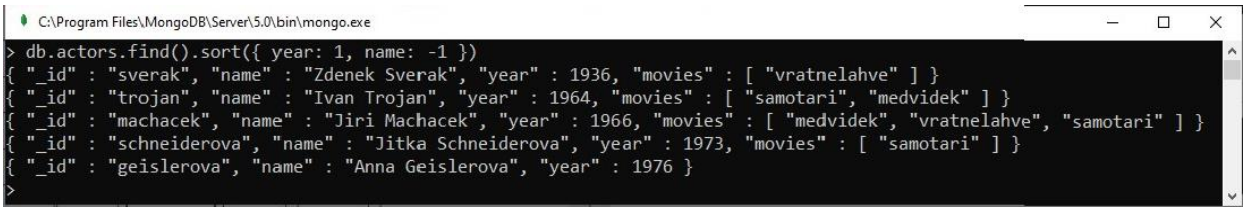
>db.actors.find({ }, { movies: 0, _id: 0 })

18. Display

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

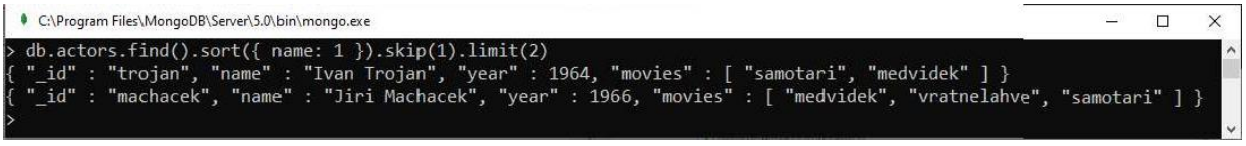
19. Display

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



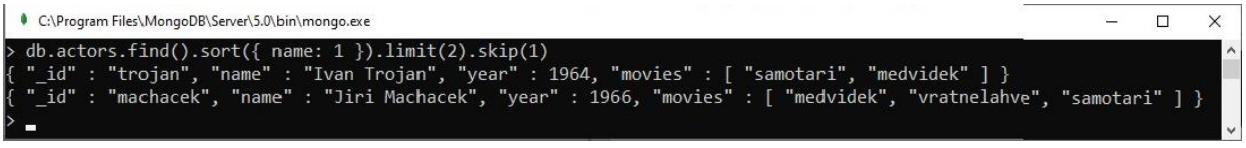
20. Display

>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

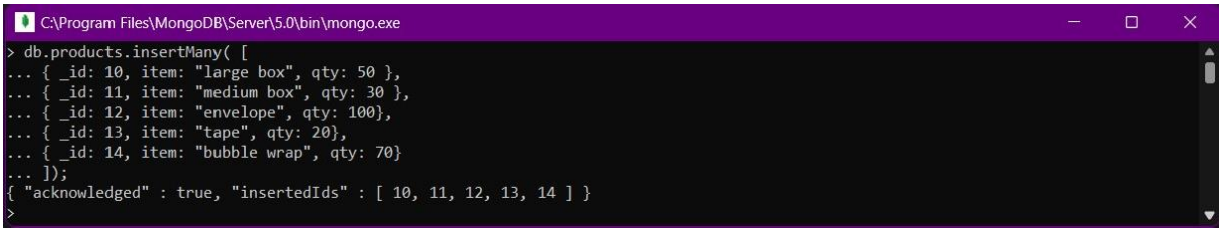
Apply CRUD operations and retrieve data in a NoSQL environment.

Procedure

1. Create and Use a Database

2. Show Databases

3. Input data

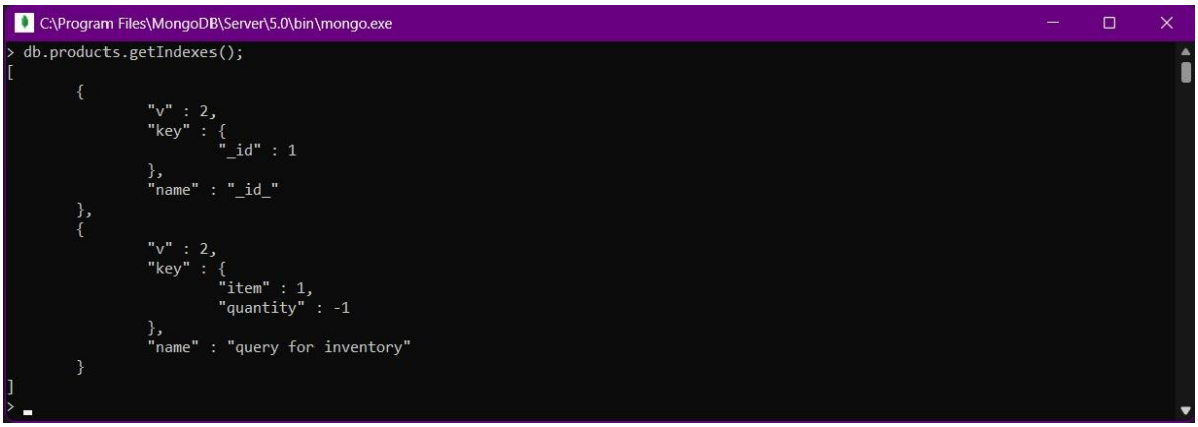


4. Create ascending index on a field

5. Create descending index on a field

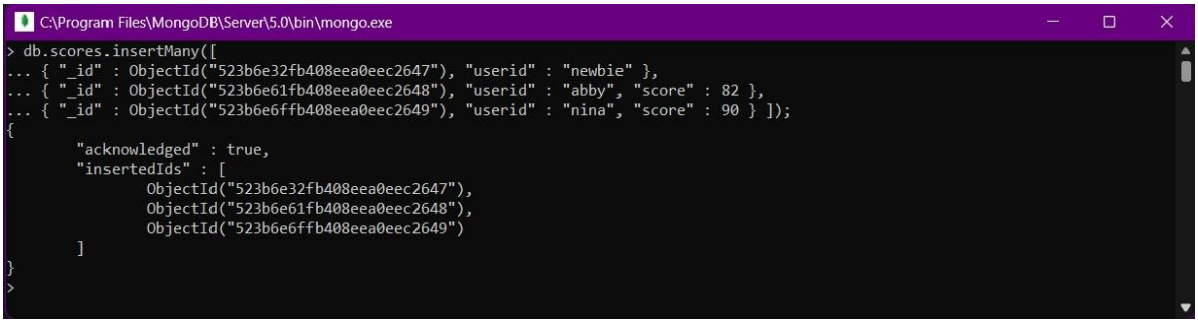
6. Create index with the index name

7. To list out indexes on the <collection>,



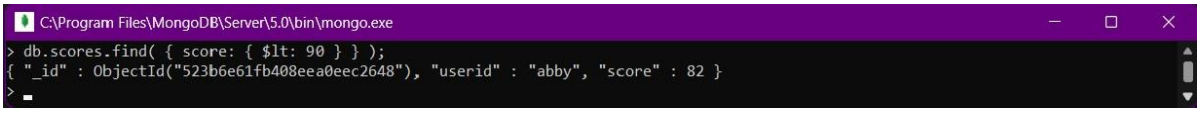
8. Drop index by index document

9. Insert data



10. Create a sparse index

11. Use Sparse index on the score field



12. Drop all indexes but _id index

Result

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

Experiment No: 15

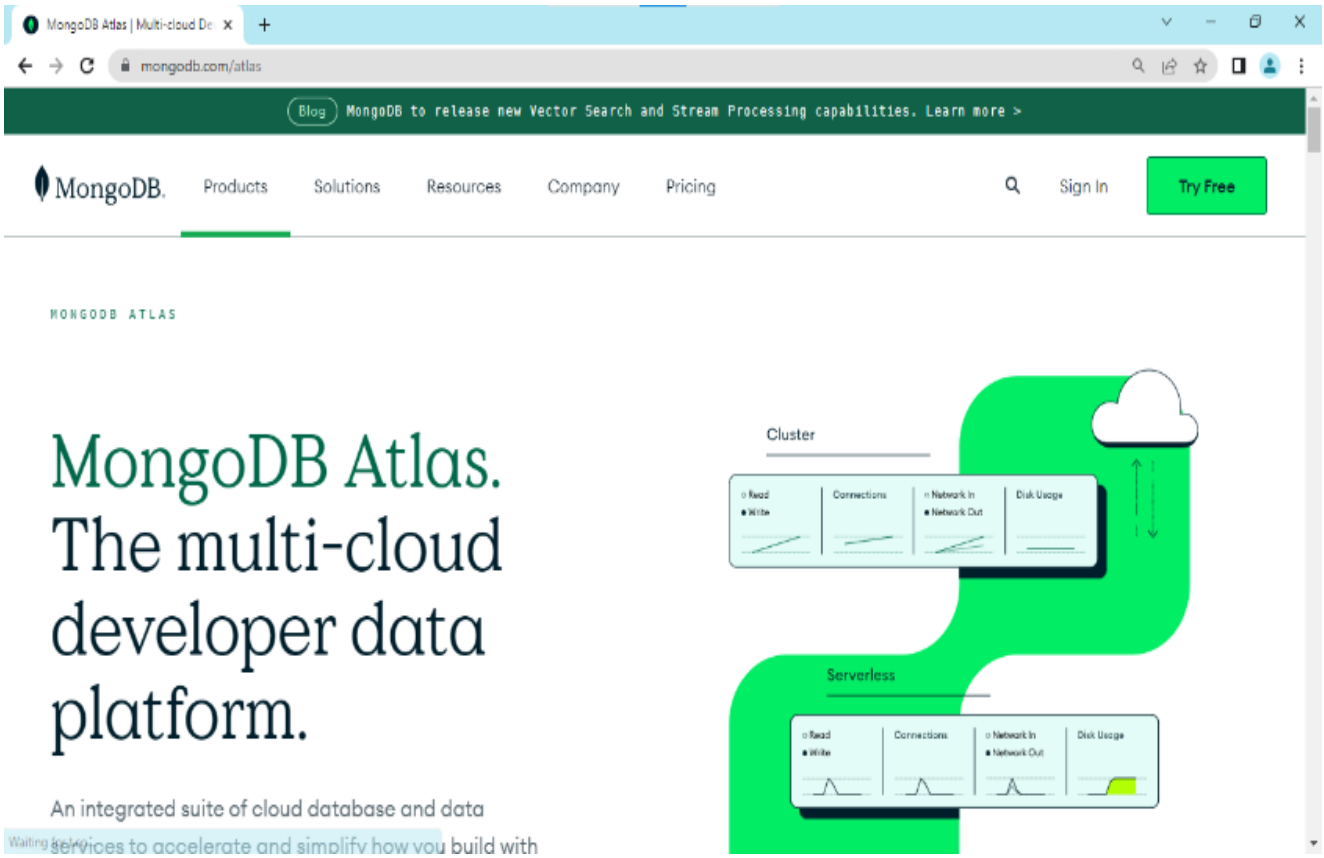
Aim

Usage of Cloud Storage Management Systems: MongoDB Atlas

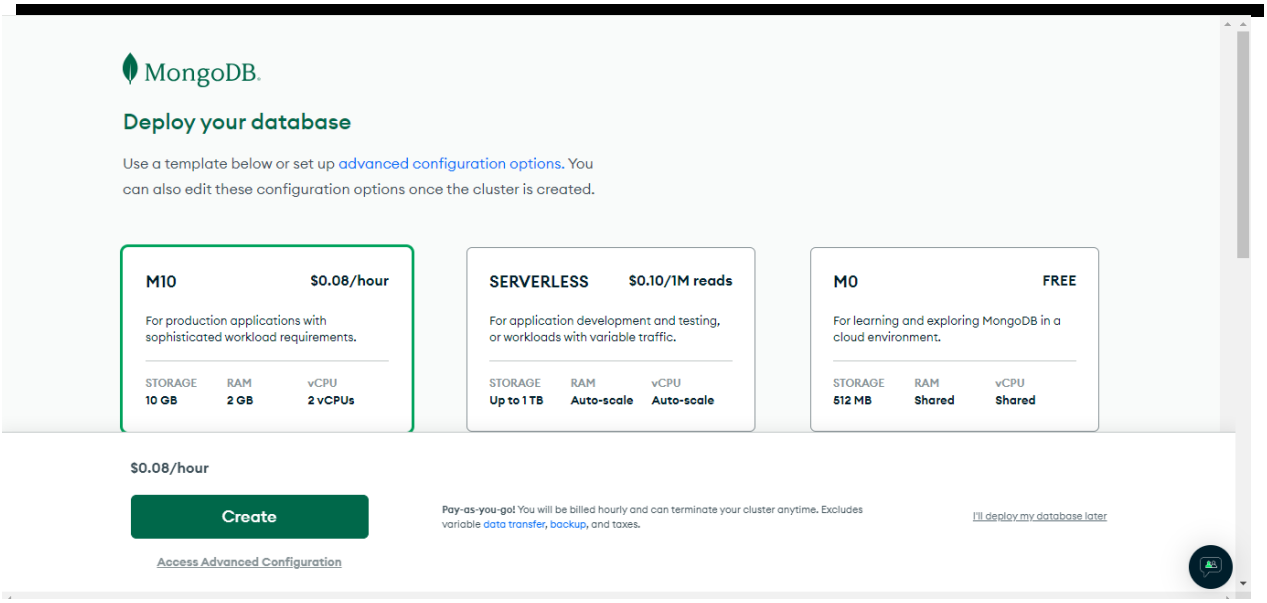
CO5

Apply CRUD operations and retrieve data in a NoSQL environment.

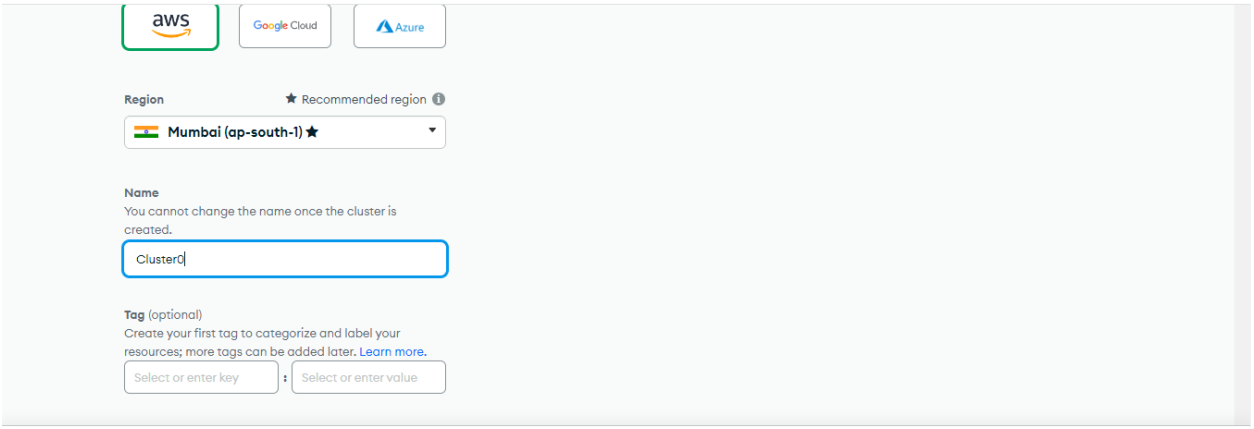
Procedure



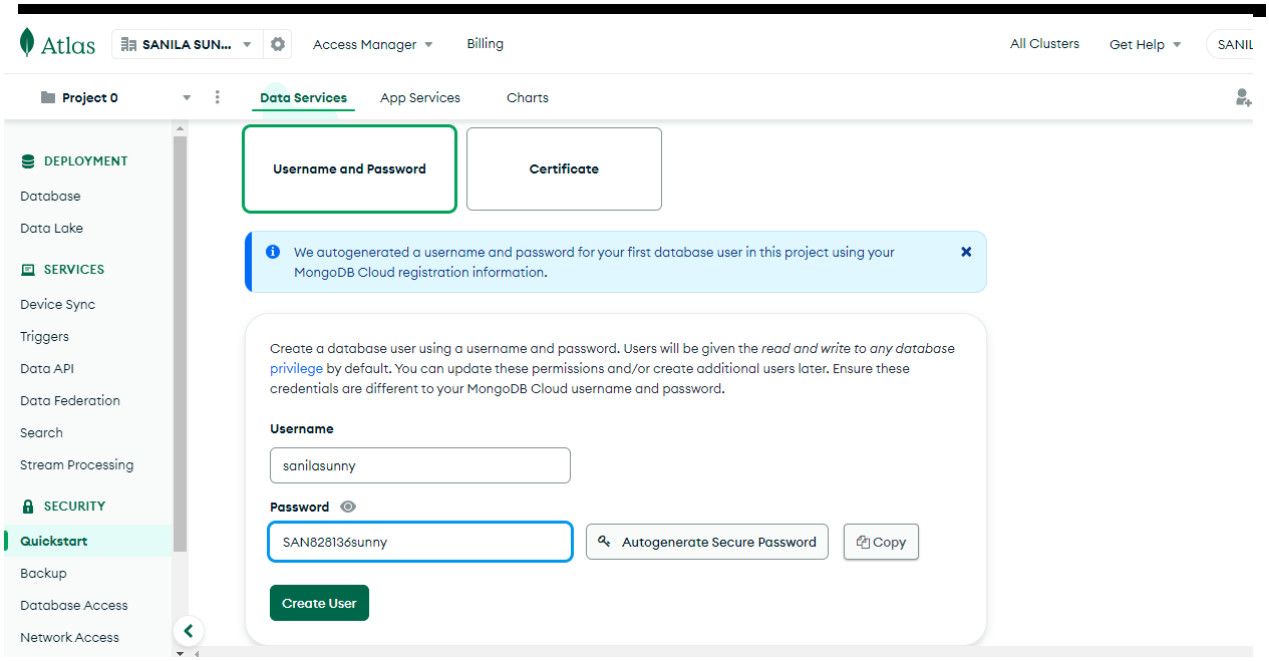
Creating Account :

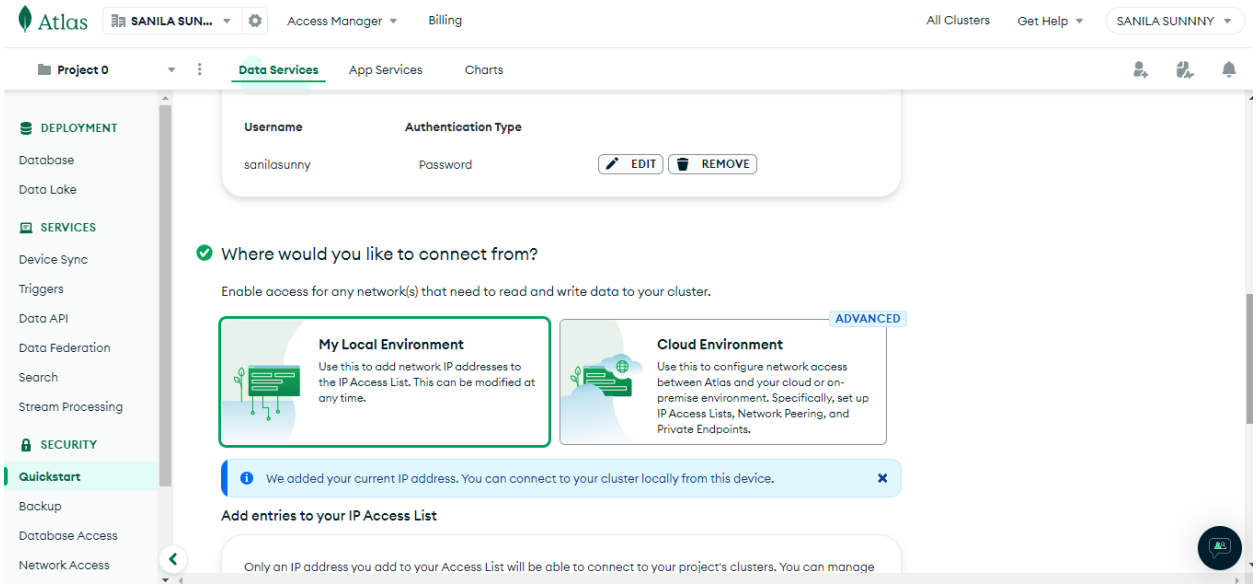


Cluster name : cluster0

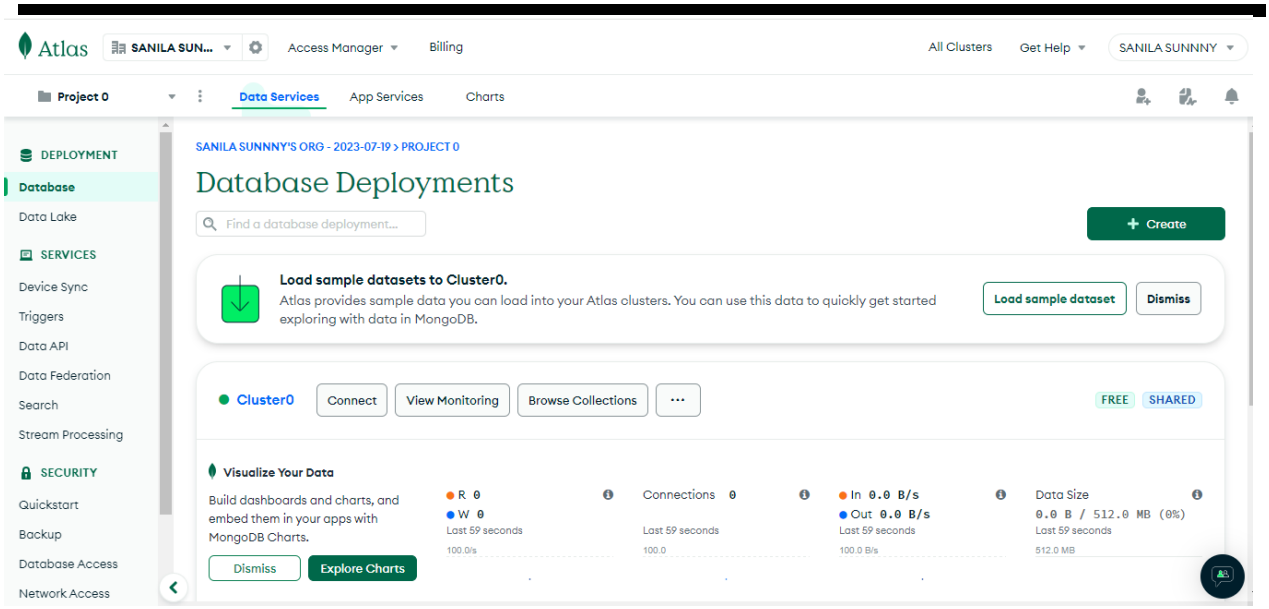


Setting password and username for accessing the DB :

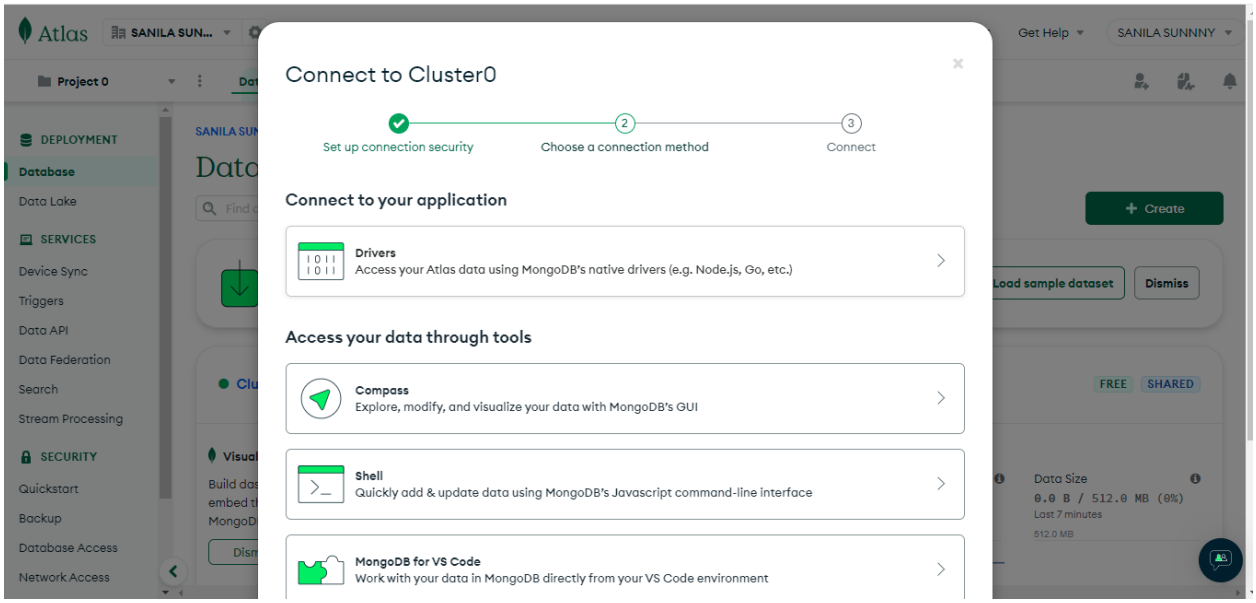




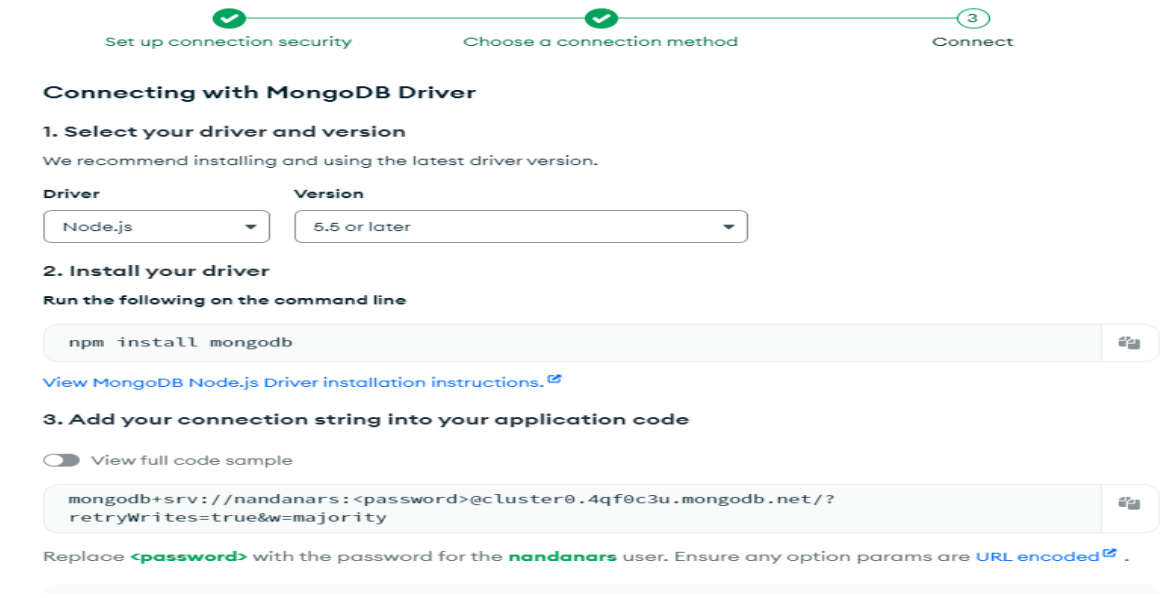
Connecting Database :



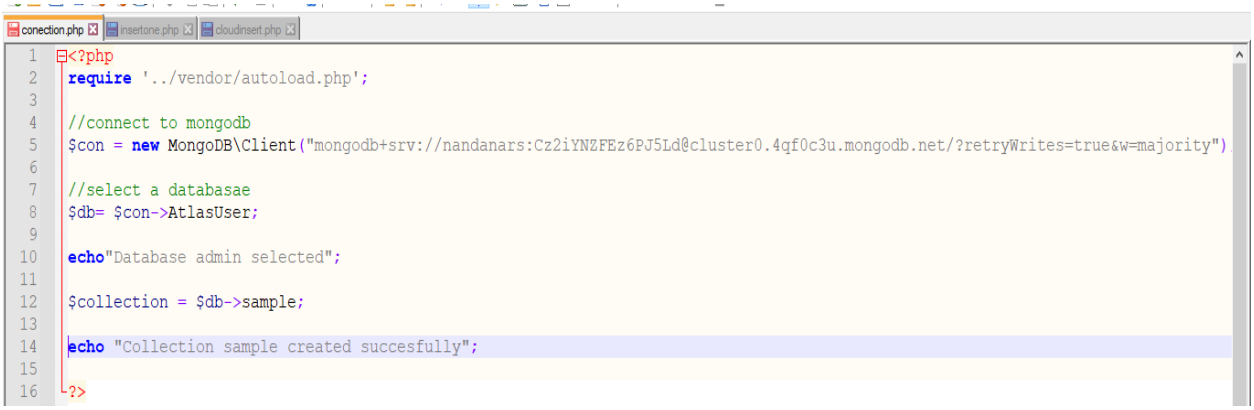
Connection Services provided by :



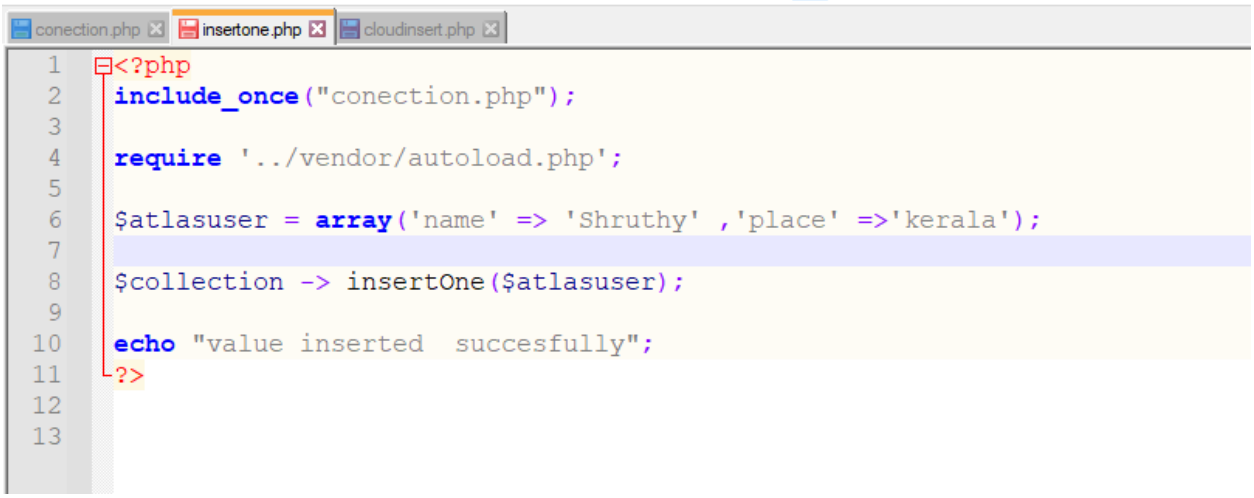
Choose connecting option Driver :



Connection code



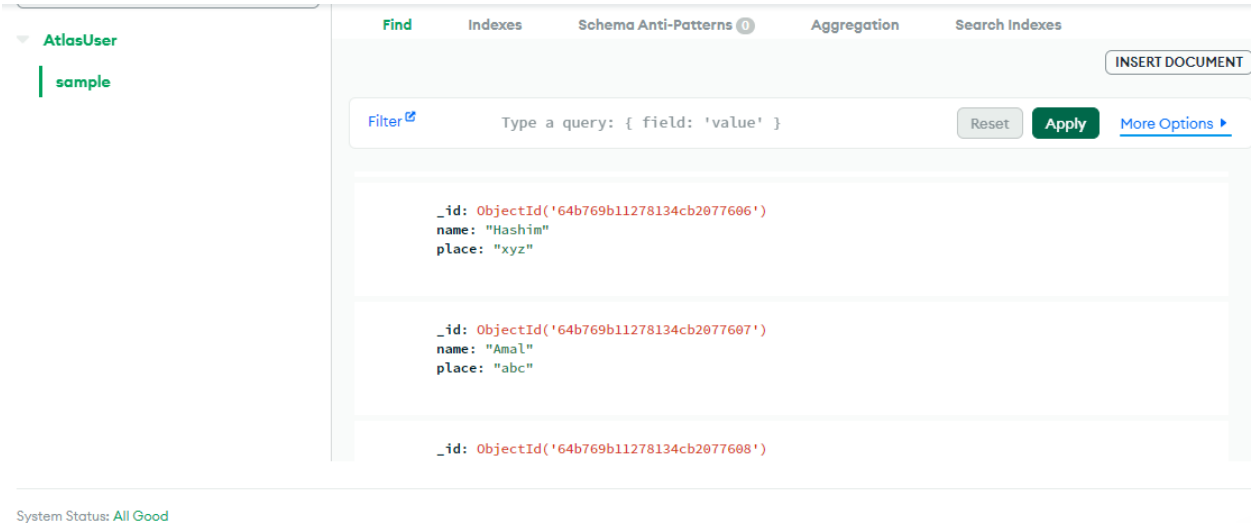
Insert one code



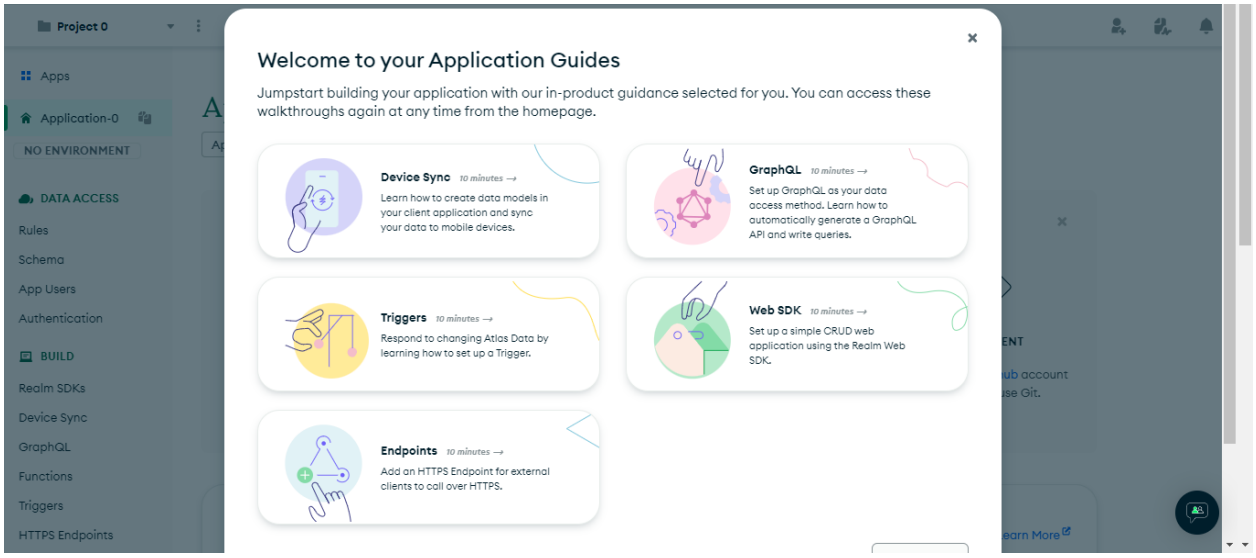
Output of InsertOne

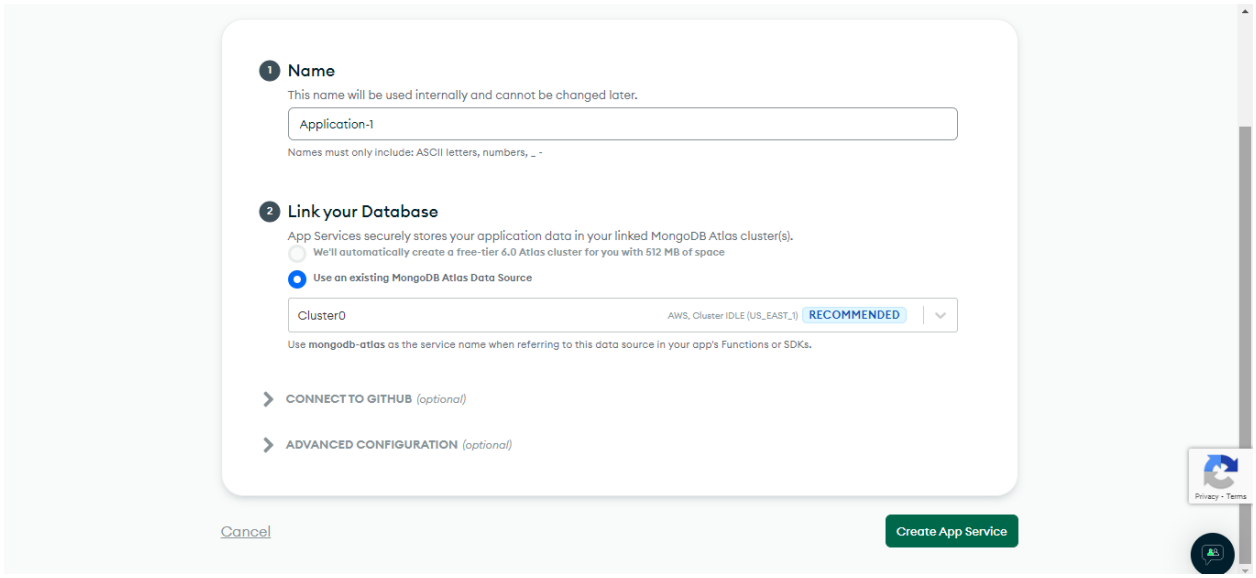
InsertMany code

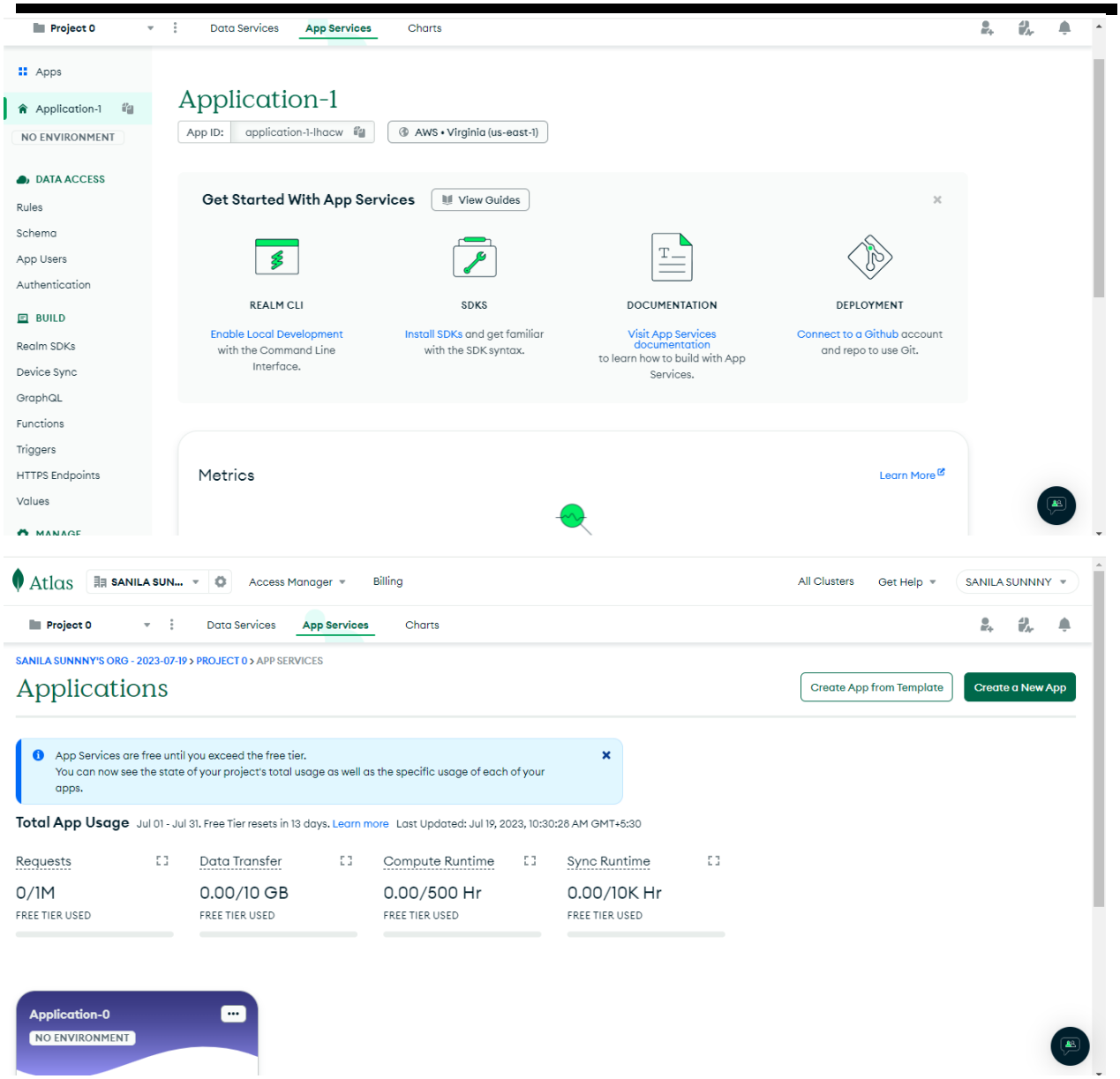
Output of insertMany:



MONGO ATALS APP SERVICE







Result:

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

Experiment No: 16

Aim:

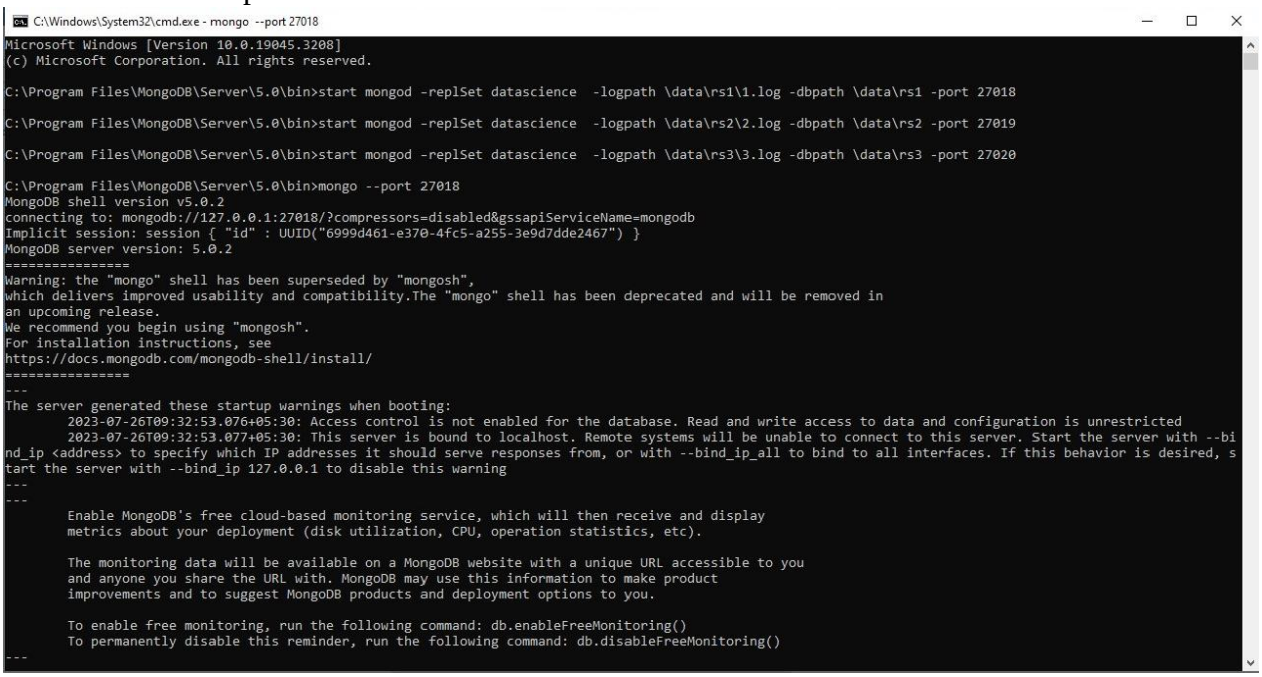
Implementation of Replica Set on MongoDB

CO5:

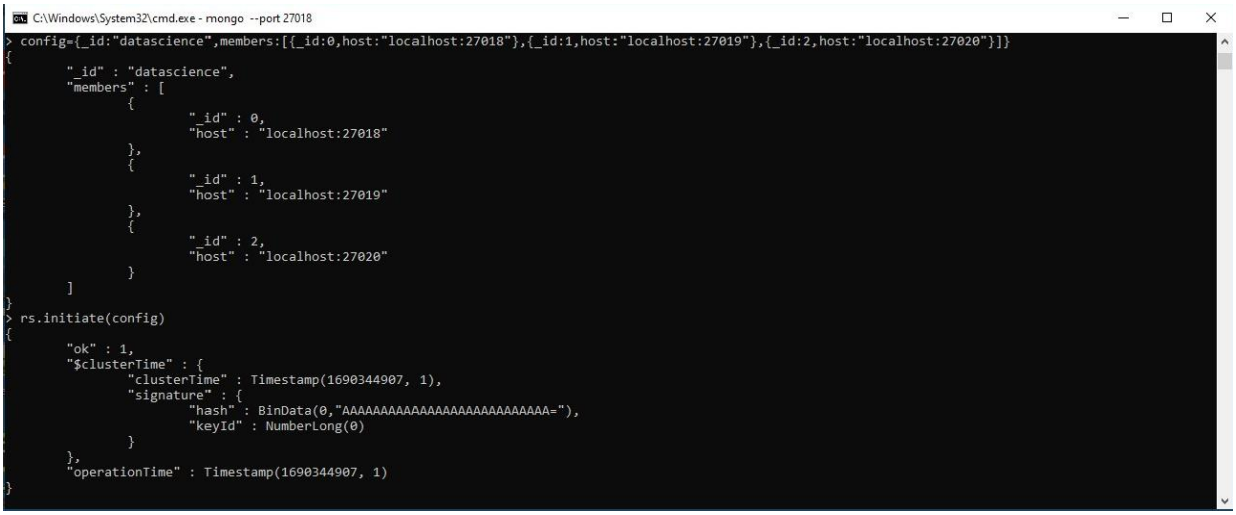
Apply CRUD operations and retrieve data in a NoSQL environment.

Procedure:

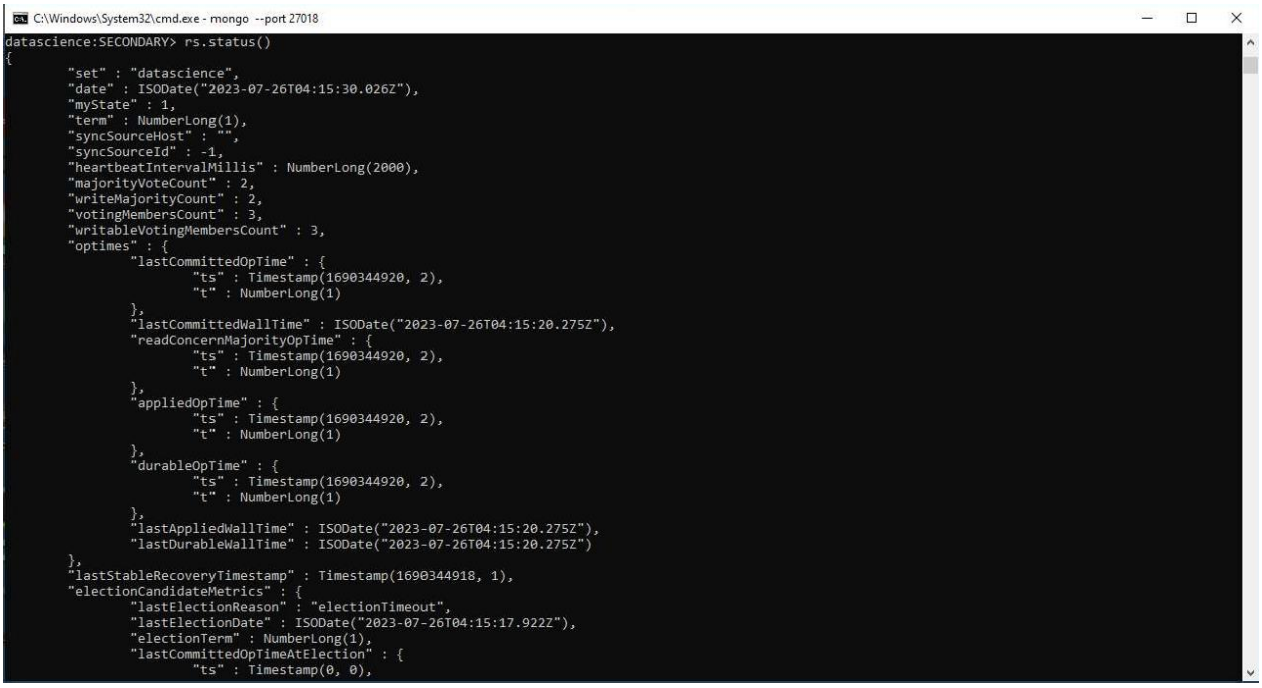
1. Create Replica Set



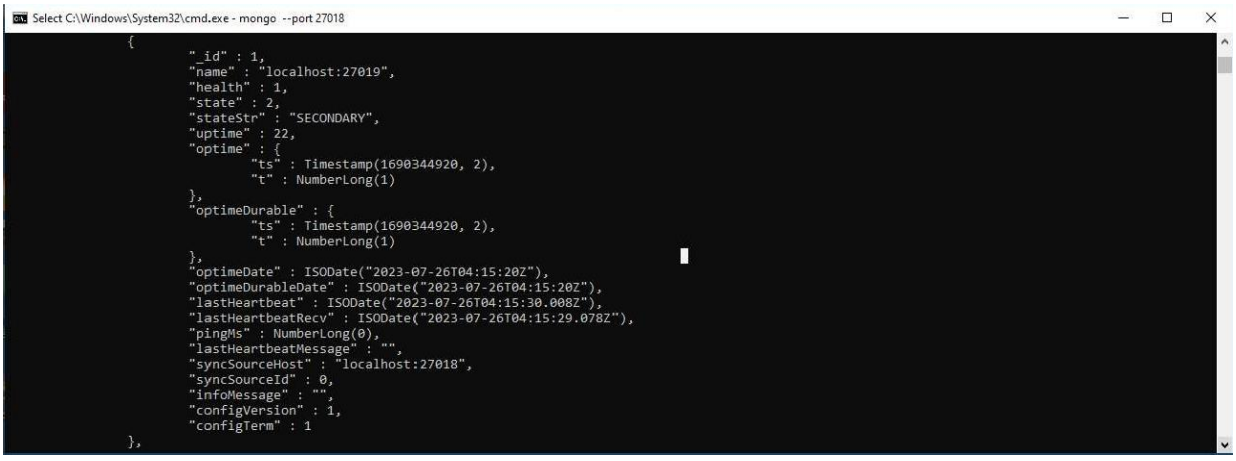
2. Configure Replica Set

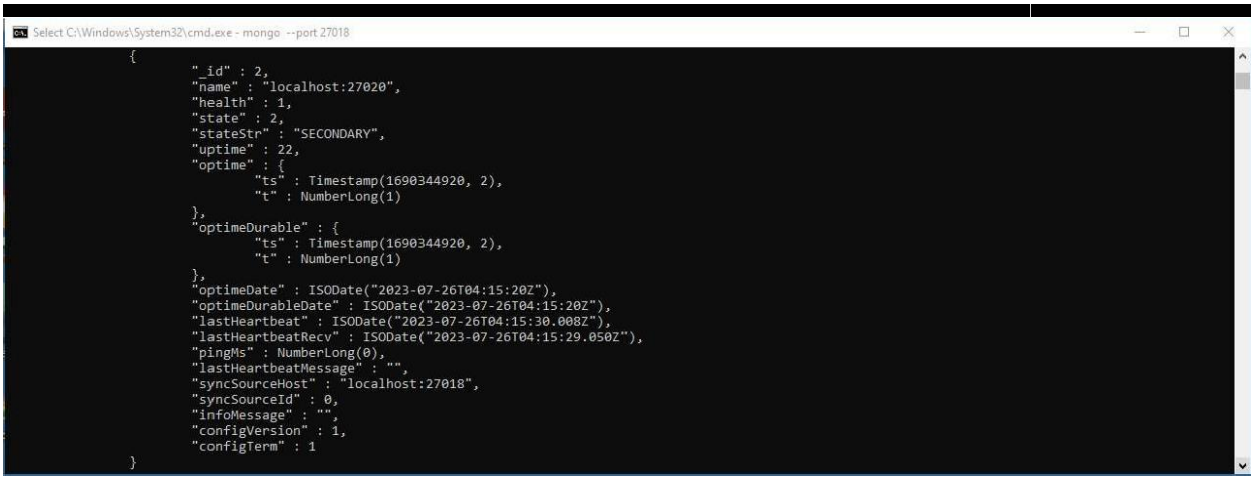


3. Replica Set Status



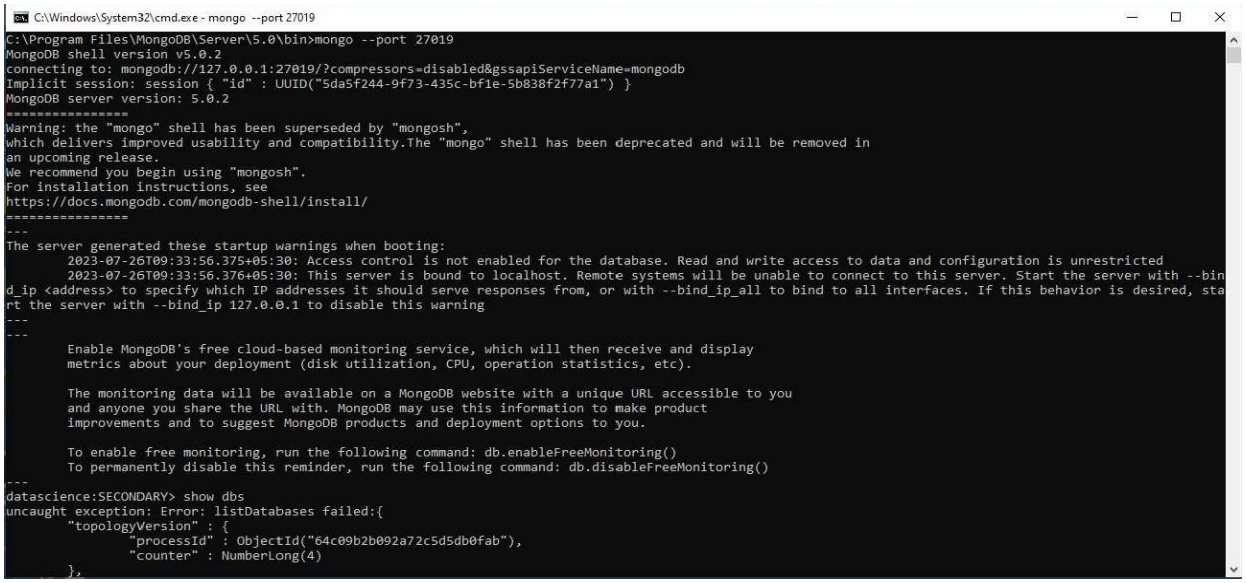
4. List Replica Set





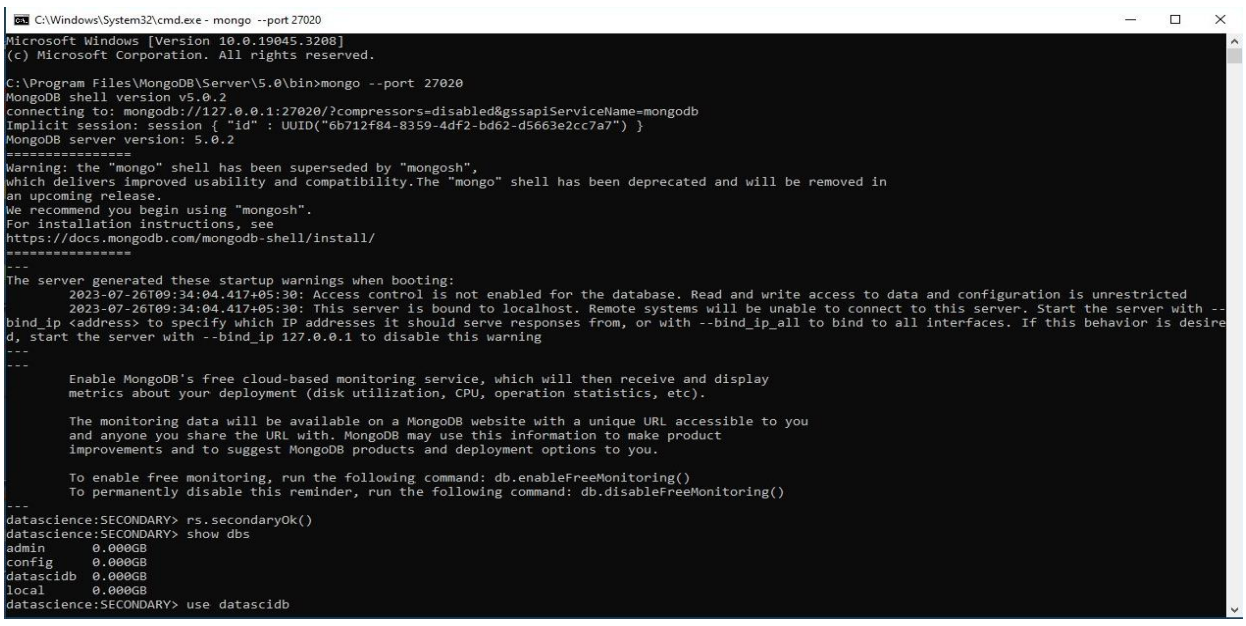
5. Performing operations at Primary

6. Performing operations at Secondary




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

Result:

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