

# **20MCA134 – ADVANCED DBMSLAB**

*Lab Report Submitted By*

**NITHIN JOSE**

**Reg. No.: AJC22MCA-2069**

*In Partial fulfilment for the Award of the Degree Of*

**MASTER OF COMPUTER APPLICATIONS (2 Year) (MCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING  
KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE,  
Accredited by NAAC with 'A' grade. Koovapally, Kanjirappally, Kottayam, Kerala – 686518]

**2022-2023**

## **DEPARTMENT OF COMPUTER APPLICATIONS**

### **AMAL JYOTHI COLLEGE OF ENGINEERING KANJIRAPPALLY**



### **CERTIFICATE**

This is to certify that the lab report, "**20MCA134 ADVANCED DBMS LAB**" is the bonafide work of **NITHIN JOSE (AJC22MCA-2069)** in partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year **2022-23**.

Ms. Navyamol K.T

**Lab In-Charge**

Rev. Fr. Dr. Rubin Thottupurathu Jose

**Head of the Department**

**Internal Examiner**

**External Examiner**

Course Code	Course Name	Syllabus Year	L-T-P-C
20MCA134	Advanced DBMS Lab	2020	0-1-3-2

## VISION

To promote an academic and research environment conducive for innovation centric technical education.

## MISSION

- MS1 - Provide foundations and advanced technical education in both theoretical and applied Computer Applications in-line with Industry demands.
- MS2 - Create highly skilled computer professionals capable of designing and innovating real life solutions.
- MS3 - Sustain an academic environment conducive to research and teaching focused to generate up-skilled professionals with ethical values.
- MS4 - Promote entrepreneurial initiatives and innovations capable of bridging and contributing with sustainable, socially relevant technology solutions.

## COURSE OUTCOME

CO	Outcome	Target
CO1	Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.	65
CO2	Apply PL/SQL for processing databases	65
CO3	Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.	65
CO4	Apply CRUD operations and retrieve data in a NoSQL environment.	65
CO5	Understand the basic storage architecture of distributed file systems.	65
CO6	Design and deployment of NoSQL databases with real time requirements.	60

## COURSE END SURVEY

CO	Survey Question	Answer Format
CO1	To what extend you are able to design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.	Excellent/Very Good/Good/Fair/Poor
CO2	To what extend you are able to apply PL/SQL for processing databases.	Excellent/Very Good/Good/Fair/Poor
CO3	To what extend you are able to compare relational and non-relational (NoSQL)databases and configuration of NoSQL Databases.	Excellent/Very Good/Good/Fair/Poor
CO4	To what extend you are able apply CRUD operations and retrieve data in a NoSQL environment.	Excellent/Very Good/Good/Fair/Poor
CO5	To what extent you are able to understand the basic storage architecture of distributed file systems.	Excellent/Very Good/Good/Fair/Poor
CO6	To what extend you are able to design and deployment of NoSQL databases with real time experiments.	Excellent/Very Good/Good/Fair/Poor

# CONTENT

<b>Sl. No.</b>	<b>Experiment</b>	<b>Date</b>	<b>CO</b>	<b>Page No.</b>
<b>1</b>	Creation of a database using DDL commands.	15-03-2023	CO1	1
<b>2</b>	Creation of a database using DML commands including integrity constraints.	17-03-2023	CO1	4
<b>3</b>	To familiarize with selecting data from single table	22-03-2023	CO1	7
<b>4</b>	To familiarize with set operations	24-03-2023	CO1	14
<b>5</b>	To familiarize with aggregate functions	29-03-2023	CO1	18
<b>6</b>	To familiarize with Join or Cartesian Product	31-03-2023	CO1	23
<b>7</b>	To familiarize with Group By and Having Clauses	31-03-2023	CO1	27
<b>8</b>	To familiarize with trigger functions	05-04-2023	CO2	30
<b>9</b>	To familiarize with Stored functions and Procedure	12-04-2023	CO2	32
<b>10</b>	Understand the installation and configuration of NoSQL Databases: MongoDB	14-06-2023	CO3	33
<b>11</b>	Create a database and use the insertMany method to insert the colours of the rainbow into the database	16-06-2023	CO4	39
<b>12</b>	Implementing CRUD operations using PHP-MongoDB	22-06-2023	CO4	41
<b>13</b>	Build sample collections/documents to perform the shell queries	12-07-2023	CO5	46
<b>14</b>	To familiarize with indexing in MongoDB	12-07-2023	CO5	50
<b>15</b>	Usage of Cloud Storage Management Systems: MongoDB Atlas	19-07-2023	CO6	53
<b>16</b>	Implementation of Replica Set on MongoDB	21-07-2023	CO5	59

## **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 fundamental tasks involved with modeling, designing and implementing a database.

### **Procedure**

#### 1. CREATE DATABASE

```
Query 1 ×
 1 create database office;
```

#### 2. USE DATABASE

```
Query 1 ×
 1 use office;
```

#### 3. CREATE TABLE - EMP

```
Query 1 ×
 1 create table emp (
 2     eno int primary key, ename varchar(20),
 3     ehiredate date, esalary int, ecommission int,
 4     check (length(eno)>=3), check (esalary<=5000)
 5 );
```

#### 4. INSERT VALUES INTO TABLE - EMP

```
Query 1 ×
 1 • insert into emp (eno, ename, ehiredate, esalary) values (101, 'Ramesh', '1980-01-07', 5000);
 2 • insert into emp (eno, ename, ehiredate, esalary, ecommission) values (102, 'Ajay', '1985-05-07', 5000, 500);
 3 • insert into emp (eno, ename, ehiredate, esalary) values (103, 'Ravi', '1981-12-08', 1500);
 4 • insert into emp (eno, ename, ehiredate, esalary, ecommission) values (104, 'Nikesh', '1983-03-03', 3000, 700);
 5 • insert into emp (eno, ename, ehiredate, esalary) values (105, 'Ravi', '1985-05-07', 3000);
 6 |
```

## 5. ADD COLUMN - SAL

Query 1

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

Result Grid

eno	ename	ehiredate	esalary	ecommission	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

## 6. DROP COLUMN - SAL

Query 1

```
1 • alter table emp drop sal;
2 • select * from emp;
```

Result Grid

eno	ename	ehiredate	esalary	ecommission
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

## 7. RENAME COLUMN - ename to name

Query 1

```
1 • ALTER TABLE emp RENAME COLUMN ename to name;
2 • SELECT * FROM emp;
3 |
```

Result Grid

eno	name	ehiredate	esalary	ecommission
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

### 8. RENAME TABLE

Query 1

```
1 • Alter table emp rename to emp_details;
2 • show tables;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Tables_in_office
emp_details

### 9. TRUNCATE TABLE

Query 1

```
1 • truncate emp_details;
2 • SELECT * FROM emp_details;
3
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

eno	name	ehiredate	esalary	ecomission
*	NULL	NULL	NULL	NULL

### 10. DROP TABLE

Query 1

```
1 • drop table emp_details;
2 • SELECT * FROM emp_details;
```

8 12:16:37 drop table emp\_details 0 row(s) affected

9 12:16:37 SELECT \* FROM emp\_details LIMIT 0, 1000 Error Code: 1146. Table 'office.emp\_details' doesn't exist

## 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.

### 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



```
Query 1 ×
CREATE DATABASE bank;
```

#### 2. USE DATABASE



```
Query 1 ×
use bank;
```

#### 3. CREATE TABLE - BRANCH



```
Query 1 ×
CREATE TABLE BRANCH (
    BNAME VARCHAR(20) PRIMARY KEY,
    CITY VARCHAR(30) CHECK(CITY IN ('NAGPUR', 'DELHI', 'BANGLORE', 'BOMBAY')) NOT NULL
);
```

#### 4. CREATE TABLE - CUSTOMER



```
Query 1 ×
CREATE TABLE CUSTOMER (
    CNAME VARCHAR(15) PRIMARY KEY,
    CITY VARCHAR(30) NOT NULL
);
```

## 5. CREATE TABLE - DEPOSIT

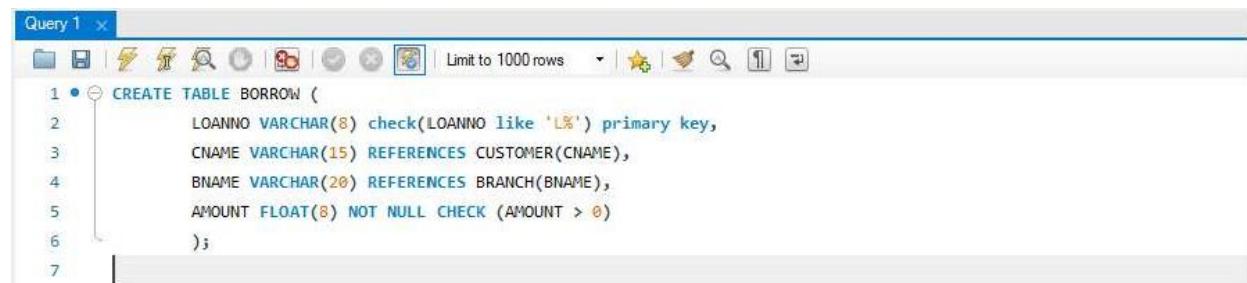


```

Query 1 ×
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

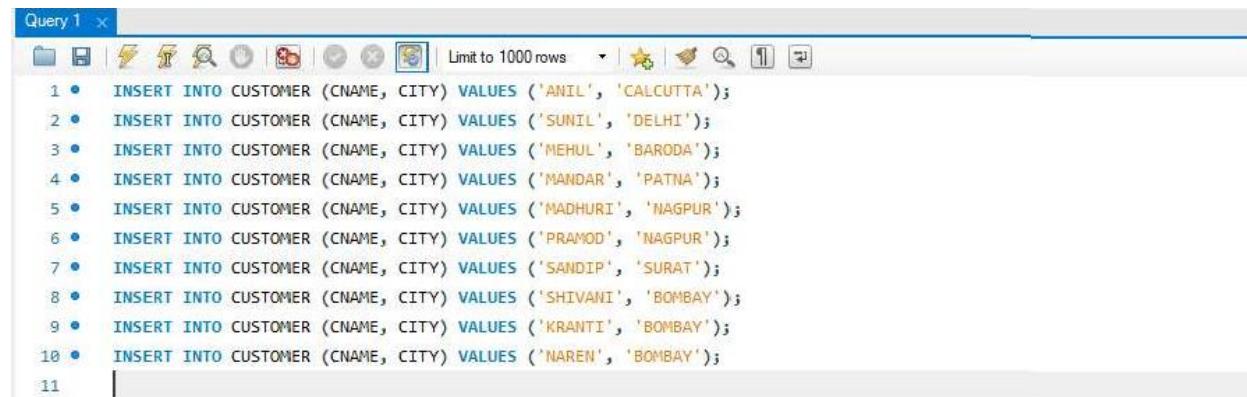


```

Query 1 ×
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



```

Query 1 ×
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



```

Query 1 ×
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

Query 1 ×

```

1 •  INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L201', 'ANIL', 'VRCE', 1000.00);
2 •  INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L206', 'MEHUL', 'AJNI', 5000.00);
3 •  INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L311', 'SUNIL', 'DHARAMPETH', 3000.00);
4 •  INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L321', 'MADHURI', 'ANDHERI', 2000.00);
5 •  INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L371', 'PRAMOD', 'VIRAR', 8000.00);
6 •  INSERT INTO borrow (LOANNO, CNAME, BNAME, AMOUNT) VALUES ('L481', 'KRANTI', 'NEHRU PLACE', 3000.00);
7

```

### 10. INSERT VALUES INTO TABLE - DEPOSIT

Query 1 ×

```

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

```

### 11. ADD CONSTRAINT - salary <= 5000

Query 1 ×

```

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

```

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

Query 1 ×

```

1 •  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:

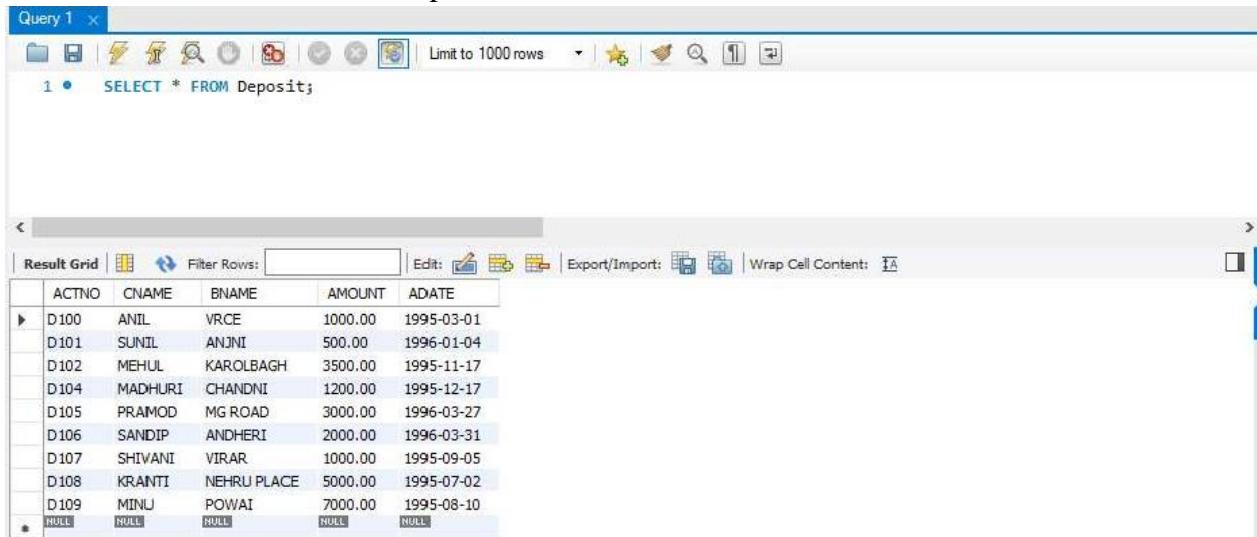
To familiarize with selecting data from single table

### CO1:

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

### Procedure

1. List all data from table deposit.



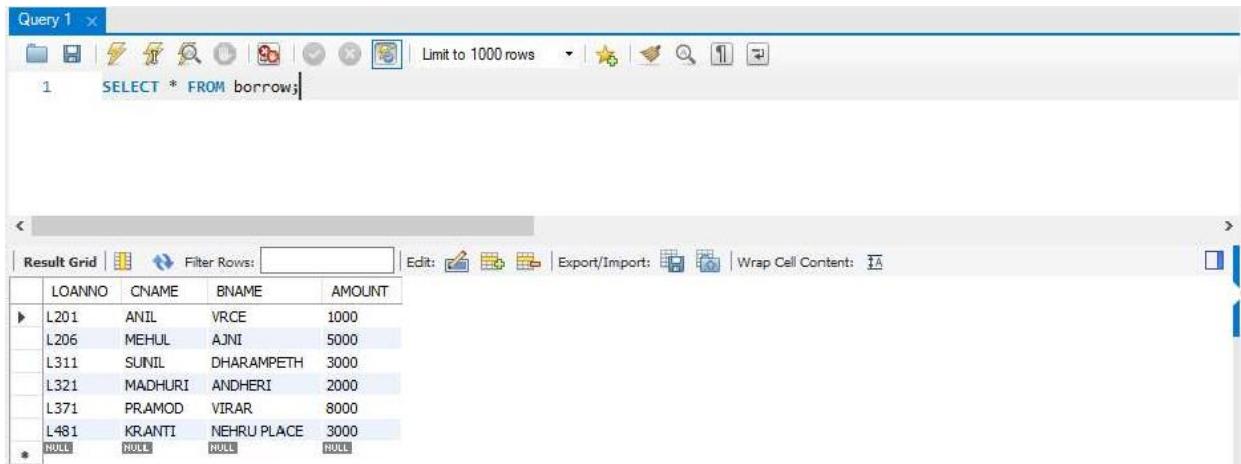
The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1 • SELECT * FROM Deposit;
```

The results are displayed in a "Result Grid" table:

	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
*	HULL	HULL	HULL	NULL	NULL

2. List all data from borrow



The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1 • SELECT * FROM borrow;
```

The results are displayed in a "Result Grid" table:

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

3. List all data from customer

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1  SELECT * FROM customers;
```

The results are displayed in a "Result Grid" table:

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

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1  SELECT * FROM branch;
```

The results are displayed in a "Result Grid" table:

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

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1  SELECT actno, amount FROM deposit;
```

The results are displayed in a "Result Grid" table:

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

Query 1 ×

1 • SELECT CNAME, ACTNO FROM DEPOSIT;

2

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

CNAME	ACTNO
ANIL	D 100
SUNIL	D 101
MEHUL	D 102
MADHURI	D 104
PRAMOD	D 105
SANDIP	D 106
SHIVANI	D 107
KRANTI	D 108
MINU	D 109
*	NULL
*	NULL

7. Give name of customers

Query 1 ×

1 • SELECT CNAME FROM CUSTOMER

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

CNAME	
ANIL	
KRANTI	
MADHURI	
MANDAR	
MEHUL	
NAREN	
PRAMOD	
SANDIP	
SHIVANI	
SUNIL	
*	NULL

## 8. Give name of branches

Query 1

```
1  SELECT BNAME FROM BRANCH;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

BNAME
AJNI
ANDHERI
CHANDNI
DHARAMPETH.
KAROLBAGH
MG ROAD
NEHRU PALACE
POWAI
VRCE
* NULL

## 9. Give name of borrows

Query 1

```
1  SELECT CNAME FROM BORROW;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

CNAME
ANIL
MEHUL
SUNIL
MADHURI
PRAMOD
KRANTI

## 10. Give names of customer living in city Nagpur

Query 1

```
1 •  SELECT CNAME FROM CUSTOMER WHERE CITY = 'NAGPUR';
Execute the selected portion of the script or everything, if there is no selection
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

CNAME
MADHURI
PRAMOD
* NULL

11. Give names of depositors having amount greater than 4000

Query 1

```
1 •   SELECT CNAME FROM DEPOSIT WHERE AMOUNT > 4000;
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

CNAME
KRANTI
MINU

12. Give account date of Anil

Query 1

```
1     SELECT ADATE FROM DEPOSIT WHERE CNAME = 'Anil';
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

ADATE
1995-03-01

13. Give name of all branches located in Bombay

Query 1

```
1     SELECT BNAME FROM BRANCH WHERE CITY = 'BOMBAY';
```

Result Grid | Filter Rows: Edit: Export/Import: Wrap Cell Content:

BNAME
ANDHERI
POWAI
*
NULL

14. Give name of borrower having loan number l205

```
Query 1
1   SELECT CNAME FROM BORROW WHERE LOANNO = '1205';
```

The screenshot shows the MySQL Workbench interface with a single query window titled "Query 1". The query itself is: "SELECT CNAME FROM BORROW WHERE LOANNO = '1205';". Below the query, the results are displayed in a "Result Grid" table. The table has one column labeled "CNAME" and one row containing the value "ANIL".

CNAME
ANIL

15. Give names of depositors having account at VRCE

```
Query 1
1   SELECT CNAME FROM DEPOSIT WHERE BNAME = 'VRCE';
```

The screenshot shows the MySQL Workbench interface with a single query window titled "Query 1". The query itself is: "SELECT CNAME FROM DEPOSIT WHERE BNAME = 'VRCE';". Below the query, the results are displayed in a "Result Grid" table. The table has one column labeled "CNAME" and one row containing the value "ANIL".

CNAME
ANIL

16. Give names of all branched located in city Delhi

```
Query 1
1   SELECT BNAME FROM BRANCH WHERE CITY = 'DELHI';
```

The screenshot shows the MySQL Workbench interface with a single query window titled "Query 1". The query itself is: "SELECT BNAME FROM BRANCH WHERE CITY = 'DELHI';". Below the query, the results are displayed in a "Result Grid" table. The table has one column labeled "BNAME" and four rows containing the values "CHANDNI", "KAROLBAGH", "NEHRU PALACE", and "DOWNTOWN".

BNAME
CHANDNI
KAROLBAGH
NEHRU PALACE
DOWNTOWN

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

```
Query 1
1   SELECT CNAME FROM DEPOSIT WHERE ADATE = '1996-12-01';
2
```

The screenshot shows the MySQL Workbench interface with a single query window titled "Query 1". The query itself is: "SELECT CNAME FROM DEPOSIT WHERE ADATE = '1996-12-01';". Below the query, the results are displayed in a "Result Grid" table. The table has one column labeled "CNAME" and one row containing the value "ANIL".

CNAME
ANIL

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

```
Query 1
1 •   SELECT ACTNO, AMOUNT FROM DEPOSIT WHERE ADATE BETWEEN '01-DEC-96' AND '01-MAY-96';
```

The screenshot shows a SQL query in the 'Query 1' window. The query selects 'ACTNO' and 'AMOUNT' from the 'DEPOSIT' table where 'ADATE' is between '01-DEC-96' and '01-MAY-96'. Below the query, the 'Result Grid' displays a single row with 'ACTNO' and 'AMOUNT' both set to NULL.

ACTNO	AMOUNT
NULL	NULL

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

```
Query 1
1 •   SELECT CITY FROM BRANCH WHERE BNAME = 'KAROLBAGH';
2
```

The screenshot shows a SQL query in the 'Query 1' window. The query selects 'CITY' from the 'BRANCH' table where 'BNAME' is 'KAROLBAGH'. Below the query, the 'Result Grid' displays a single row with 'CITY' set to 'DELHI'.

CITY
DELHI

20. Give details of customer ANIL

```
Query 1
1 •   SELECT * FROM DEPOSIT WHERE CNAME = 'Anil';
```

The screenshot shows a SQL query in the 'Query 1' window. The query selects all columns from the 'DEPOSIT' table where 'CNAME' is 'Anil'. Below the query, the 'Result Grid' displays a single row with columns: ACTNO (D100), CNAME (ANIL), BNAME (VRCE), AMOUNT (1000.00), and ADATE (1995-03-01).

ACTNO	CNAME	BNAME	AMOUNT	ADATE
D100	ANIL	VRCE	1000.00	1995-03-01

## Result

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

## **Experiment No: 4**

### **Aim:**

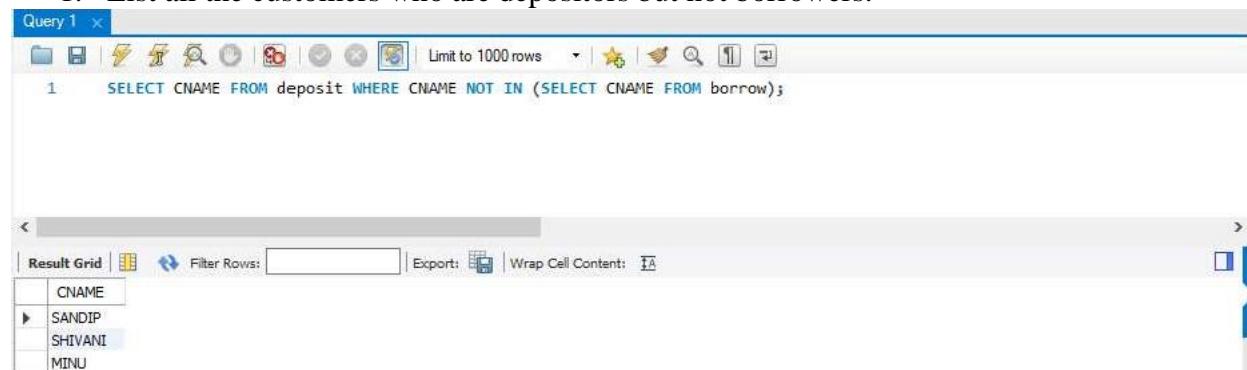
To familiarize with Set Operations.

### **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.



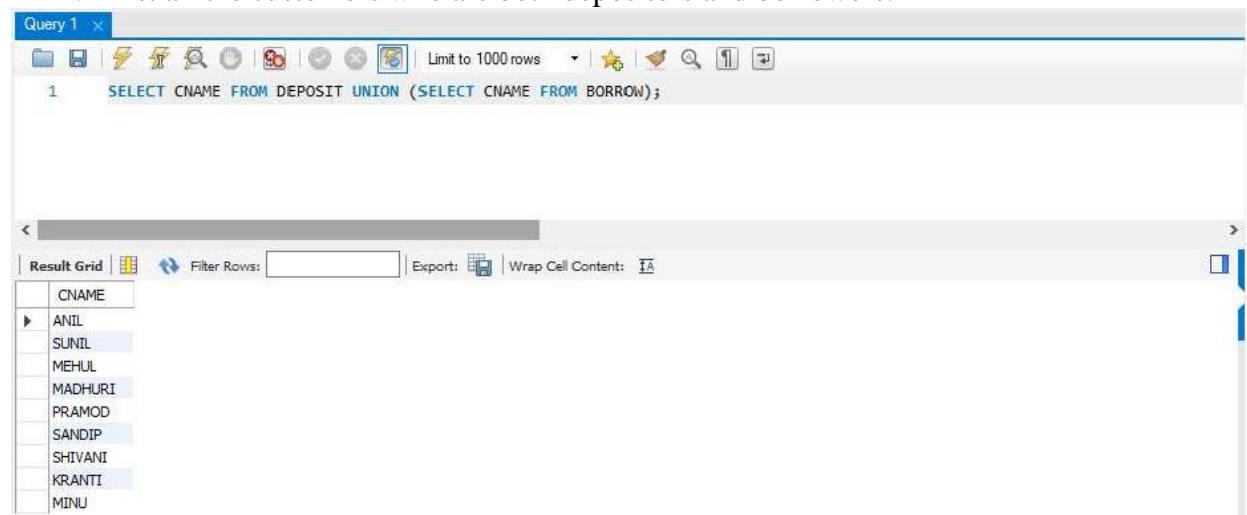
The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1  SELECT CNAME FROM deposit WHERE CNAME NOT IN (SELECT CNAME FROM borrow);
```

The result grid displays the following data:

CNAME
SANDIP
SHIVANI
MINU

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



The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1  SELECT CNAME FROM DEPOSIT UNION (SELECT CNAME FROM BORROW);
```

The result grid displays the following data:

CNAME
ANIL
SUNIL
MEHUL
MADHURI
PRAMOD
SANDIP
SHIVANI
KRANTI
MINU

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

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1   SELECT D1.CNAME FROM DEPOSIT D1 WHERE D1.BNAME IN (SELECT D2.BNAME FROM DEPOSIT D2 WHERE D2.CNAME = 'SUNIL' );
2
```

Below the query, the results are displayed in a "Result Grid" table:

CNAME
SUNIL

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

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1 •   SELECT C1.CNAME FROM CUSTOMER C1,DEPOSIT D1, BRANCH B1 WHERE C1.CITY = 'NAGPUR' AND
2     C1.CNAME = D1.CNAME AND D1.BNAME = B1.BNAME AND B1.CITY IN ('BOMBAY','DELHI');
3
```

Below the query, the results are displayed in a "Result Grid" table:

CNAME
MADHURI

5. List all the depositors living in city NAGPUR

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1 •   SELECT DISTINCT(CUSTOMER.CNAME) from CUSTOMER,DEPOSIT WHERE City='NAGPUR';
2
```

Below the query, the results are displayed in a "Result Grid" table:

CNAME
PRAMOD
MADHURI

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

The screenshot shows a MySQL Workbench interface with a query editor titled "Query 1". The query is:

```
1 •   SELECT C1.CNAME FROM CUSTOMER C1,DEPOSIT D1, BRANCH B1 WHERE C1.CITY = 'NAGPUR' AND C1.CNAME = D1.CNAME AND
2     D1.BNAME = B1.BNAME AND B1.CITY IN ('BOMBAY');
```

Below the query, the results are displayed in a "Result Grid" table:

CNAME
-------

7. List the branch cities of Anil and Sunil

```
Query 1 ×
SELECT B1.CITY FROM DEPOSIT D1, BRANCH B1 WHERE D1.BNAME = B1.BNAME AND D1.CNAME IN ('SUNIL', 'ANIL');
2
```

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query selects the city from the DEPOSIT and BRANCH tables where the branch name matches and the customer name is either 'SUNIL' or 'ANIL'. The result grid shows one row with the value 'NAGPUR'.

CITY
NAGPUR

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

```
Query 1 ×
SELECT DISTINCT D1.CNAME FROM deposit D1, borrow B1 WHERE D1.AMOUNT>1000 AND B1.AMOUNT<10000;
2
```

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query selects distinct customer names from the deposit and borrow tables where the deposit amount is greater than 1000 and the loan amount is less than 10000. The result grid shows six rows: MEHUL, MADHURI, PRAMOD, SANDIP, KRANTI, and MINU.

CNAME
MEHUL
MADHURI
PRAMOD
SANDIP
KRANTI
MINU

9. List the cities of depositors having branch VRCE.

```
Query 1 ×
SELECT B1.CITY FROM deposit D1, branch B1 WHERE D1.BNAME=B1.BNAME AND B1.BNAME='VRCE';
2
```

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query selects the city from the deposit and branch tables where the branch name is 'VRCE'. The result grid shows one row with the value 'NAGPUR'.

CITY
NAGPUR

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

```
Query 1 ×
SELECT D1.CNAME FROM deposit D1, customer C1 WHERE AMOUNT<1000 AND C1.CITY=(C1.CNAME='ANIL');
```

The screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The query selects the name of customers from the "deposit" table (alias D1) who have an amount less than 1000 and live in the same city as a customer named "ANIL" from the "customer" table (alias C1). The result grid displays one row with the value "SUNIL".

CNAME
SUNIL

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

```
Query 1 ×
SELECT B1.CITY FROM BRANCH B1 WHERE B1.BNAME IN (SELECT D1.BNAME FROM DEPOSIT D1 WHERE D1.CNAME IN ('ANIL','SUNIL'));
```

The screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The query selects the city from the "branch" table (alias B1) where the branch name is in the list of branch names belonging to customers named "ANIL" or "SUNIL". The result grid displays one row with the value "NAGPUR".

CITY
NAGPUR

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

```
Query 1 ×
1 •  SELECT DISTINCT(D1.CNAME),D1.AMOUNT ,C1.CITY FROM deposit D1,
2   CUSTOMER C1, BRANCH B1 WHERE D1.CNAME=C1.CNAME AND C1.CITY
3   IN(SELECT C2.CITY FROM customer C2 WHERE C2.CNAME='ANIL');
4
```

The screenshot shows the Oracle SQL Developer interface with a query window titled "Query 1". The query uses a complex join involving the "deposit" table (alias D1), "customer" table (alias C1), and "branch" table (alias B1). It selects distinct customer names, amounts, and cities, filtering by customer name and city, and further filtering the city by being in the list of cities where customers named "ANIL" live. The result grid displays one row with values "ANIL", "1000.00", and "CALCUTTA".

CNAME	AMOUNT	CITY
ANIL	1000.00	CALCUTTA

## 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

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query is: `SELECT SUM(AMOUNT) FROM BORROW;`. The results grid shows one row with the value 22000.

SUM(AMOUNT)
22000

2. List total deposit

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query is: `SELECT SUM(AMOUNT) FROM DEPOSIT;`. The results grid shows one row with the value 24200.00.

SUM(AMOUNT)
24200.00

3. List total loan taken from KAROLBAGH branch

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The query is: `SELECT MAX(AMOUNT) FROM BORROW WHERE BNAME = 'KAROLBAGH';`. The results grid shows one row with the value NULL.

MAX(AMOUNT)
NULL

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

```
Query 1
1  SELECT SUM(AMOUNT) from deposit where adate>'1995-03-01';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

SUM(AMOUNT)
23200.00

5. List total deposit of customers living in city NAGPUR

```
Query 1
1  SELECT SUM(D1.AMOUNT) FROM DEPOSIT D1 , CUSTOMER C1 WHERE C1.CITY = 'NAGPUR' AND C1.CNAME = D1.CNAME;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

SUM(D1.AMOUNT)
4200.00

6. List maximum deposit of customer living in Bombay

```
Query 1
1  SELECT MAX(D1.AMOUNT) FROM DEPOSIT D1 , CUSTOMER C1 WHERE C1.CITY
2    = 'Bombay' AND C1.CNAME = D1.CNAME;
3
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

MAX(D1.AMOUNT)
5000.00

7. List total deposit of customer having branch in BOMBAY

```
Query 1
1  SELECT SUM(AMOUNT) from deposit,BRANCH where city='BOMBAY';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

SUM(AMOUNT)
48400.00

## 8. Count total number of branch cities

Query 1 ×

```
1  SELECT COUNT(DISTINCT(CITY)) FROM BRANCH;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

COUNT(DISTINCT(CITY))
4

## 9. Count total number of customers cities

Query 1 ×

```
1  SELECT count(city) from CUSTOMER;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

count(city)
10

## 10. Give branch names and branch wise deposit

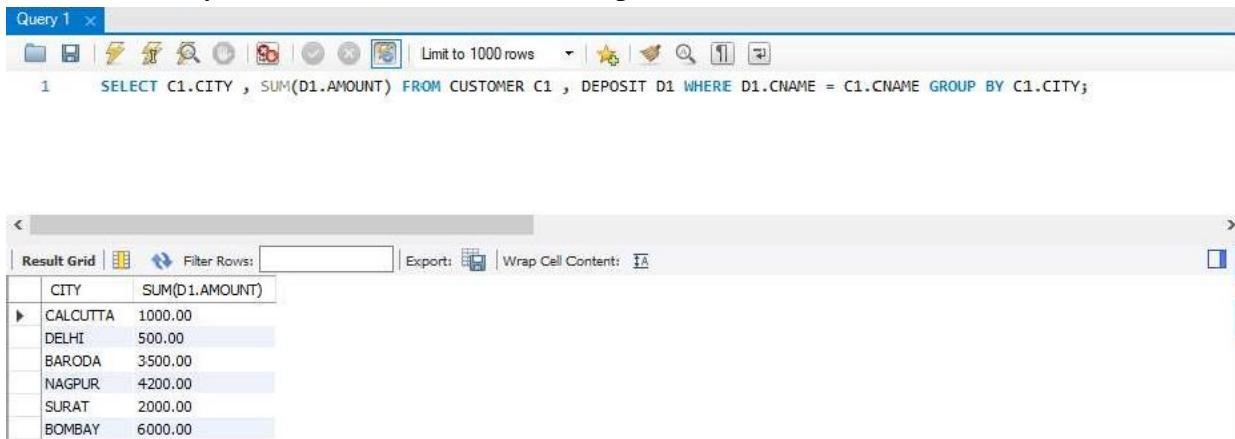
Query 1 ×

```
1  SELECT BNAME , SUM(AMOUNT) FROM DEPOSIT GROUP BY BNAME;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

BNAME	SUM(AMOUNT)
VRCE	1000.00
ANJNİ	500.00
KAROLBAGH	3500.00
CHANDNI	1200.00
MG ROAD	3000.00
ANDHERI	2000.00
VIRAR	1000.00
NEHRU PLACE	5000.00
POWAI	7000.00

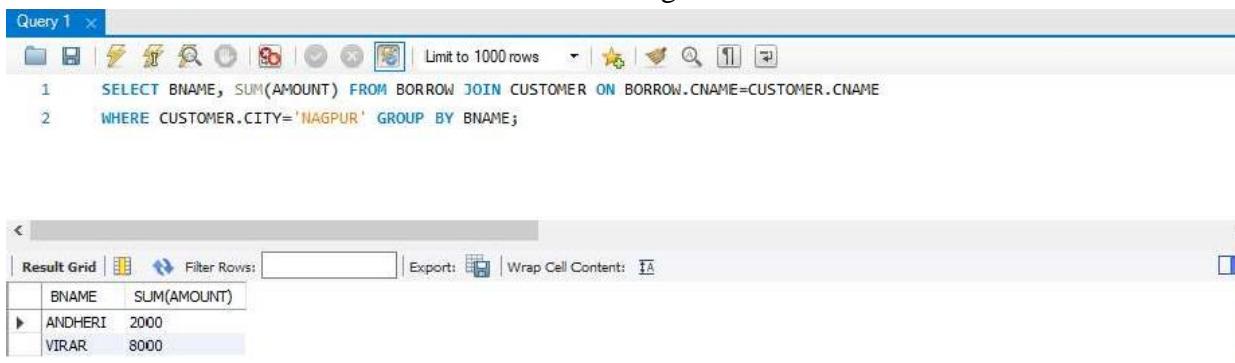
11. Give city wise name and branch wise deposit



```
Query 1
1  SELECT C1.CITY , SUM(D1.AMOUNT) FROM CUSTOMER C1 , DEPOSIT D1 WHERE D1.CNAME = C1.CNAME GROUP BY C1.CITY;
```

CITY	SUM(D1.AMOUNT)
CALCUTTA	1000.00
DELHI	500.00
BARODA	3500.00
NAGPUR	4200.00
SURAT	2000.00
BOMBAY	6000.00

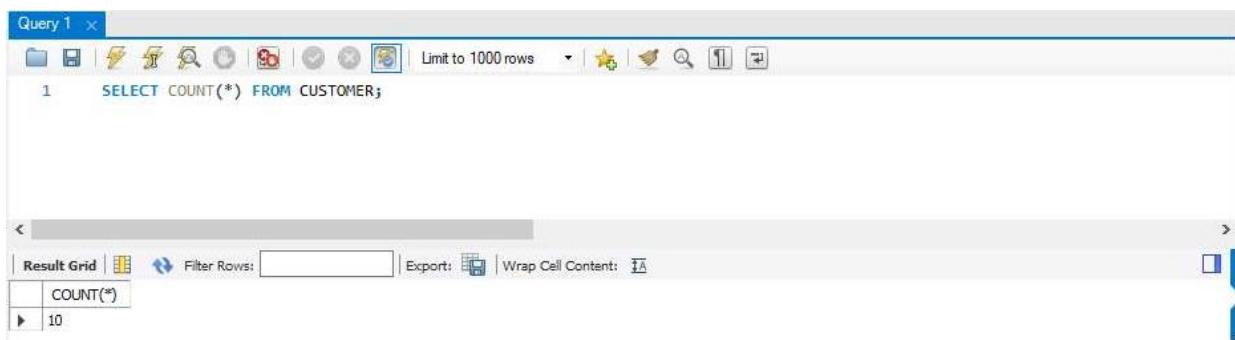
12. Give the branch wise loan of customer living in NAGPUR



```
Query 1
1  SELECT BNAME, SUM(AMOUNT) FROM BORROW JOIN CUSTOMER ON BORROW.CNAME=CUSTOMER.CNAME
2  WHERE CUSTOMER.CITY='NAGPUR' GROUP BY BNAME;
```

BNAME	SUM(AMOUNT)
ANDHERI	2000
VIRAR	8000

13. Count total number of customers



```
Query 1
1  SELECT COUNT(*) FROM CUSTOMER;
```

COUNT(*)
10

**14. Count total number of depositors branch wise**

```
Query 1
1   SELECT BNAME, COUNT(DISTINCT CNAME) FROM DEPOSIT GROUP BY BNAME;
```

The screenshot shows the MySQL Workbench interface with a query window titled "Query 1". The query is: "SELECT BNAME, COUNT(DISTINCT CNAME) FROM DEPOSIT GROUP BY BNAME;". The results are displayed in a "Result Grid" table:

BNAME	COUNT(DISTINCT CNAME)
ANDHERI	1
ANJNI	1
CHANDNI	1
KAROLBAGH	1
MG ROAD	1
NEHRU PLACE	1
POWAI	1
VIRAR	1
VRCE	1

**15. Give maximum loan from branch VRCE**

```
Query 1
1 •  SELECT BNAME, count(*) FROM DEPOSIT, CUSTOMER WHERE deposit.CNAME = CUSTOMER.CNAME GROUP BY BNAME;
```

The screenshot shows the MySQL Workbench interface with a query window titled "Query 1". The query is: "SELECT BNAME, count(\*) FROM DEPOSIT, CUSTOMER WHERE deposit.CNAME = CUSTOMER.CNAME GROUP BY BNAME;". The results are displayed in a "Result Grid" table:

BNAME	count(*)
VRCE	1
ANJNI	1
KAROLBAGH	1
CHANDNI	1
MG ROAD	1
ANDHERI	1
VIRAR	1
NEHRU PLACE	1

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

```
Query 1
1   select count(customer.CNAME) from customer where customer.CNAME IN (select deposit.cname from deposit)
2   and customer.CNAME IN (select borrow cname from borrow);
```

The screenshot shows the MySQL Workbench interface with a query window titled "Query 1". The query is: "select count(customer.CNAME) from customer where customer.CNAME IN (select deposit.cname from deposit) and customer.CNAME IN (select borrow cname from borrow);". The results are displayed in a "Result Grid" table:

count(customer.CNAME)
6

## **Result**

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

## Experiment no: 6

### Aim:

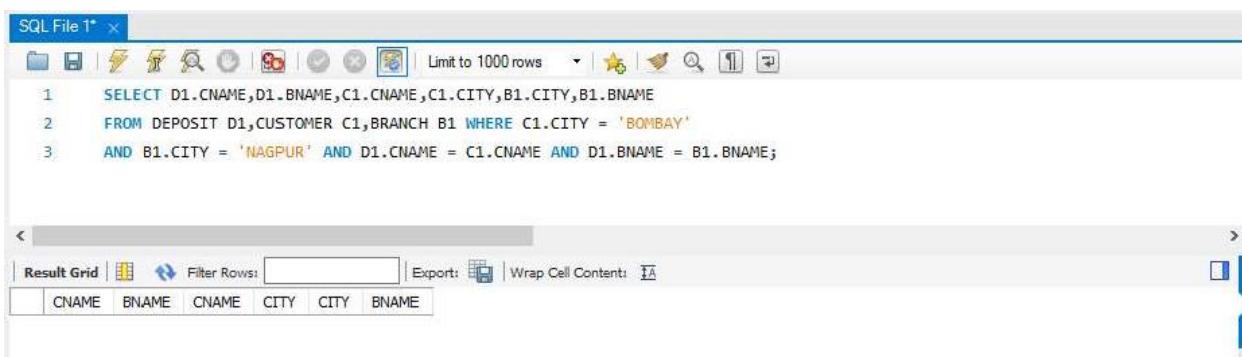
To familiarize with Join or Cartesian Product

### 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

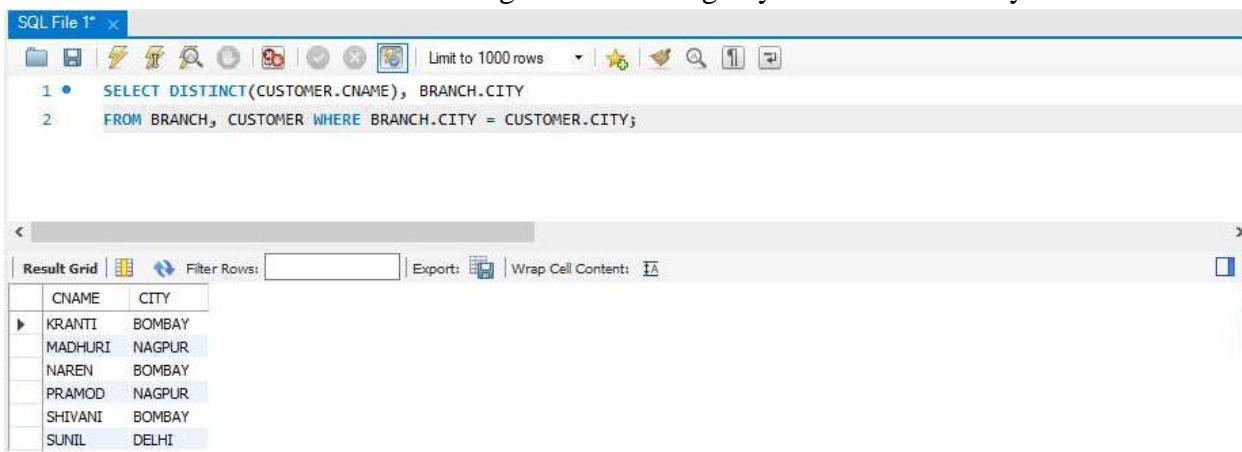


```
SQL File 1* 
1   SELECT D1.CNAME,D1.BNAME,C1.CNAME,C1.CITY,B1.CITY,B1.BNAME
2   FROM DEPOSIT D1,CUSTOMER C1,BRANCH B1 WHERE C1.CITY = 'BOMBAY'
3   AND B1.CITY = 'NAGPUR' AND D1.CNAME = C1.CNAME AND D1.BNAME = B1.BNAME;
```

The screenshot shows the SQL Editor window with the above query. Below the editor is the Result Grid pane, which displays the following data:

CNAME	BNAME	CNAME	CITY	CITY	BNAME
KRANTI	BOMBAY	MADHURI	NAGPUR	NAREN	BOMBAY
PRAMOD	NAGPUR	SHIVANI	BOMBAY	SUNIL	DELHI

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



```
SQL File 1* 
1 •  SELECT DISTINCT(CUSTOMER.CNAME), BRANCH.CITY
2   FROM BRANCH, CUSTOMER WHERE BRANCH.CITY = CUSTOMER.CITY;
```

The screenshot shows the SQL Editor window with the above query. Below the editor is the Result Grid pane, which displays the following data:

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.

SQL File 1\*

```

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

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

CNAME
MADHURI
PRAMOD

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

SQL File 1\*

```

1 •  SELECT BR1.CNAME, BR1.AMOUNT, D1.CNAME, D1.AMOUNT
2     FROM BORROW BR1,DEPOSIT D1 WHERE D1.CNAME = BR1.CNAME AND D1.AMOUNT > 1000 AND BR1.AMOUNT > 2000;
3

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

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

SQL File 1\*

```

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

```

Result Grid | Filter Rows: Export: Wrap Cell Content:

CNAME
SUNIL

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

The screenshot shows the SQL Developer interface with a SQL file named 'SQL File 1'. The code is:

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

The result grid shows two columns: CNAME and AMOUNT. There are no results displayed.

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

The screenshot shows the SQL Developer interface with a SQL file named 'SQL File 1'. The code is:

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

The result grid shows two columns: CNAME and AMOUNT. There are no results displayed.

8. Give branch city and living city of Pramod

The screenshot shows the SQL Developer interface with a SQL file named 'SQL File 1'. The code is:

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

The result grid shows two columns: CNAME and AMOUNT. There are no results displayed.

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

The screenshot shows a SQL query window titled "SQL File 1\*". The query is:

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

The result grid shows one row:

CITY
NAGPUR

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

The screenshot shows a SQL query window titled "SQL File 1\*". The query is:

```
1 • SELECT C1.CNAME, C1.CITY FROM CUSTOMER C1 WHERE C1.CNAME = 'ANIL' OR C1.CNAME = 'SUNIL';
```

The result grid shows three rows:

CNAME	CITY
ANIL	CALCUTTA
SUNIL	DELHI
*	NULL

## **Result**

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

## Experiment no: 7

### Aim:

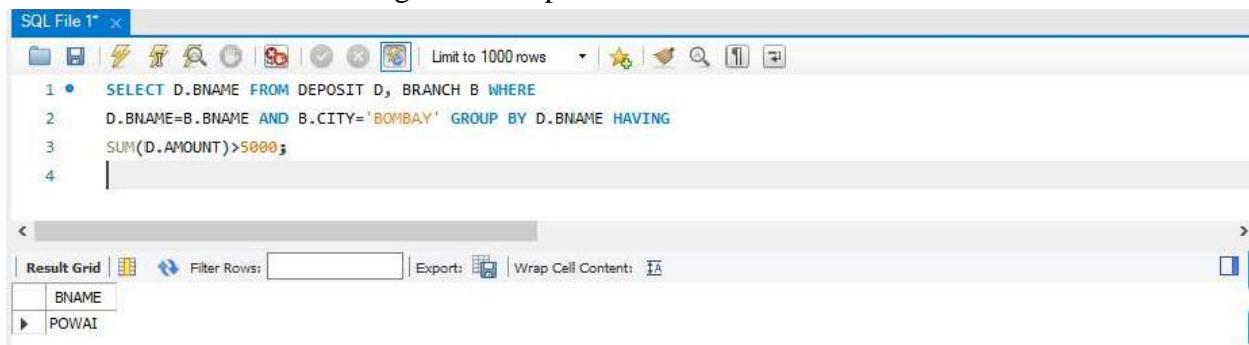
To familiarize with Group by and Having Clauses

### 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.



The screenshot shows the SQL Developer interface with a SQL file named "SQL File 1". The code is as follows:

```

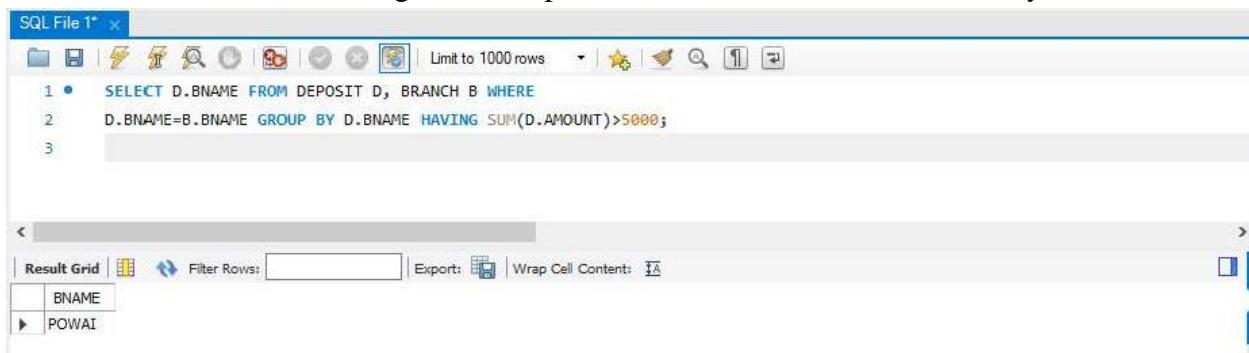
1 •  SELECT D.BNAME FROM DEPOSIT D, BRANCH B WHERE
2     D.BNAME=B.BNAME AND B.CITY='BOMBAY' GROUP BY D.BNAME HAVING
3     SUM(D.AMOUNT)>5000;
4

```

The result grid shows two rows:

BNAME
POWAI

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



The screenshot shows the SQL Developer interface with a SQL file named "SQL File 1". The code is as follows:

```

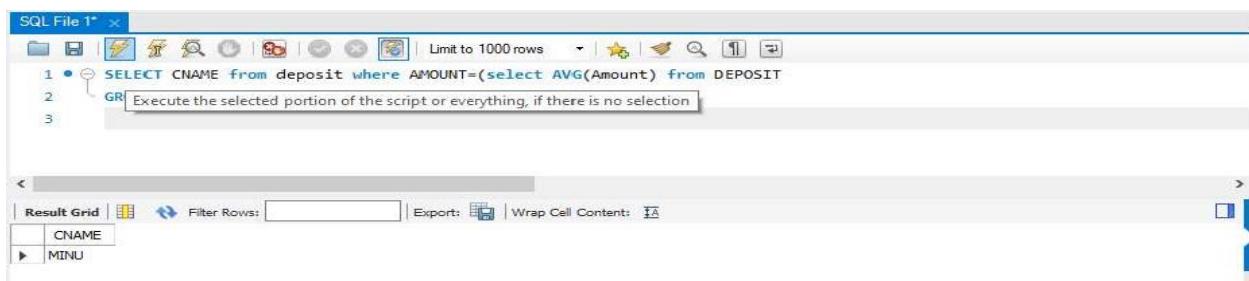
1 •  SELECT D.BNAME FROM DEPOSIT D, BRANCH B WHERE
2     D.BNAME=B.BNAME GROUP BY D.BNAME HAVING SUM(D.AMOUNT)>5000;
3

```

The result grid shows two rows:

BNAME
POWAI

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



The screenshot shows the SQL Developer interface with a SQL file named "SQL File 1". The code is as follows:

```

1 •  SELECT CNAME from deposit where AMOUNT=(select AVG(Amount) from DEPOSIT
2     GR Execute the selected portion of the script or everything, if there is no selection
3

```

The result grid shows one row:

CNAME
MINU

4. List the names of customers having maximum deposit

```
SQL File 1* 
1   SELECT MAX(AMOUNT) CNAME FROM deposit;
```

The screenshot shows the SQL Editor window with the query above. Below the editor is the Result Grid pane, which displays a single row of data:

CNAME
7000.00

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

```
SQL File 1* 
1 •  SELECT D1.BNAME FROM DEPOSIT D1 GROUP BY D1.BNAME
2     HAVING COUNT(D1.CNAME) >= ALL (SELECT COUNT(D2.CNAME) FROM DEPOSIT D2 GROUP BY D2.BNAME);
3
```

The screenshot shows the SQL Editor window with the query above. Below the editor is the Result Grid pane, which displays a list of branch names:

BNAME
VRCE
ANJINI
KAROLBAGH
CHANDNI
MG ROAD
ANDHERI
VIRAR
NEHRU PLACE
POWAI

6. Count the number of depositors living in NAGPUR.

```
SQL File 1* 
1 •  SELECT COUNT(DEPOSIT.CNAME)FROM DEPOSIT,CUSTOMER WHERE CUSTOMER.CITY='NAGPUR';
```

The screenshot shows the SQL Editor window with the query above. Below the editor is the Result Grid pane, which displays a single row of data:

COUNT(DEPOSIT.CNAME)
18

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

```
SQL File 1* 
1 • SELECT CNAME FROM DEPOSIT WHERE BNAME='VRCE' AND AMOUNT=(SELECT
2     MAX(AMOUNT) FROM DEPOSIT WHERE BNAME='VRCE');
3 
```

The screenshot shows the SQL Editor with the following query:

```
SELECT CNAME FROM DEPOSIT WHERE BNAME='VRCE' AND AMOUNT=(SELECT MAX(AMOUNT) FROM DEPOSIT WHERE BNAME='VRCE');
```

The Result Grid shows one row:

CNAME
ANIL

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

```
SQL File 1* 
1 • SELECT BNAME from deposit GROUP BY BNAME HAVING
2     COUNT(BNAME)>5;
3 
```

The screenshot shows the SQL Editor with the following query:

```
SELECT BNAME from deposit GROUP BY BNAME HAVING COUNT(BNAME)>5;
```

The Result Grid shows one row:

BNAME
-------

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

```
SQL File 1* 
1 • SELECT C.CNAME ,COUNT(B.BNAME) FROM CUSTOMER C JOINED TOTAL_BRANCH B ON
2     C.CNAME=B.BNAME GROUP BY C.CNAME ORDER BY COUNT(B.BNAME) DESC;
3 
```

The screenshot shows the SQL Editor with the following query:

```
SELECT C.CNAME ,COUNT(B.BNAME) FROM CUSTOMER C JOINED TOTAL_BRANCH B ON C.CNAME=B.BNAME GROUP BY C.CNAME ORDER BY COUNT(B.BNAME) DESC;
```

The Result Grid shows one row:

CNAME	COUNT(B.BNAME)

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

```
SQL File 1* 
1 • SELECT COUNT(B1.BNAME) FROM DEPOSIT D1 BORROW B1 CUSTOMER C1 WHERE
2     C1.CITY=D1.CITY AND D1.BNAME=B1.BNAME IN (SELECT CITY FROM CUSTOMER);
3 
```

The screenshot shows the SQL Editor with the following query:

```
SELECT COUNT(B1.BNAME) FROM DEPOSIT D1 BORROW B1 CUSTOMER C1 WHERE C1.CITY=D1.CITY AND D1.BNAME=B1.BNAME IN (SELECT CITY FROM CUSTOMER);
```

The Result Grid shows one row:

COUNT(B1.BNAME)
6

## 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

```

Query 1
CREATE DATABASE employee_db;
USE employee_db;
CREATE TABLE `employee` (
    `emp_id` int NOT NULL,
    `emp_name` varchar(45) DEFAULT NULL,
    `dob` date DEFAULT NULL,
    `address` varchar(45) DEFAULT NULL,
    `designation` varchar(45) DEFAULT NULL,
    `mobile_no` int DEFAULT NULL,
    `dept_no` int DEFAULT NULL,
    `salary` int DEFAULT NULL,
    PRIMARY KEY (`emp_id`)
);
SELECT * FROM employee;

```

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The code listed creates a database named "employee\_db", selects it, and then creates a table named "employee" with columns for emp\_id, emp\_name, dob, address, designation, mobile\_no, dept\_no, and salary. A primary key is defined on emp\_id. Below the code, a result grid shows a single row with all columns set to NULL.

```

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;

```

The screenshot shows the MySQL Workbench interface with a query editor titled "Query 1". The code listed creates a table named "salary" with columns for employee\_id, old\_sal, new\_sal, and rev\_date. A primary key is defined on employee\_id. Below the code, a result grid shows a single row with all columns set to NULL.

The screenshot shows the MySQL Workbench interface. At the top, there is a query editor window titled 'employee' containing the following SQL code:

```

1   INSERT INTO `employee_db`.`employee`
2     (`emp_id`, `emp_name`, `dob`, `address`, `designation`, `mobile_no`, `dept_no`, `salary`)
3     VALUES ('1', 'amal', '1995-04-29', 'wayanad', 'developer', '9469664422', '7', '40000');
4
5 •  SELECT * FROM employee;
6

```

Below the query editor is a 'Result Grid' showing the data inserted into the 'employee' table:

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

On the left side of the interface, there is a table configuration panel for the 'employee' table. It includes fields for Table Name (employee), Schema (employee\_db), Charset/Collation (utf8mb4\_0900\_ai\_ci), and Engine (InnoDB). There is also a 'Comments:' section.

Below the table configuration is a trigger definition for the 'employee' table:

```

BEFORE INSERT
AFTER INSERT
BEFORE UPDATE
▼ AFTER UPDATE
  employe_AFTER_UPDATE
BEFORE DELETE
AFTER DELETE

```

```

1 •  CREATE DEFINER='root'@'localhost'
2  ◇ TRIGGER `employee_AFTER_UPDATE` AFTER UPDATE ON `employee` FOR EACH ROW BEGIN
3    if(new.salary != old.salary)
4    then
5      INSERT INTO salary (employee_id,old_sal,new_sal,rev_date) values
6      (new.emp_id,old.salary,new.salary,sysdate());
7    END if;
8  END

```

The screenshot shows the MySQL Workbench interface. At the top, there is a query editor window titled 'salary' containing the following SQL code:

```

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

```

Below the query editor is a 'Result Grid' showing the data in the 'salary' table:

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

## Result

The program was executed and the result was successfully obtained. Thus, CO<sub>2</sub> 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()
```

The screenshot shows the MySQL Workbench interface. In the SQL editor tab, the code for the stored procedure WhoBorrow is shown. The second line, which contains the call to the procedure, is highlighted with a blue dot. The result grid below displays the output of the procedure, which is a list of borrow records from the BORROW table.

LOANNO	CNAME	BNAME	AMOUNT
L201	ANIL	VRCE	1000.00
L206	MEHUL	AJNI	5000.00
L311	SUNIL	DHARAMPETH	3000.00
L321	MADHURI	ANDHERI	2000.00
L371	PRAMOD	VIRAR	8000.00
L481	KRANTI	NEHRU PLACE	3000.00

### Result

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

## Experiment No:10

### Aim:

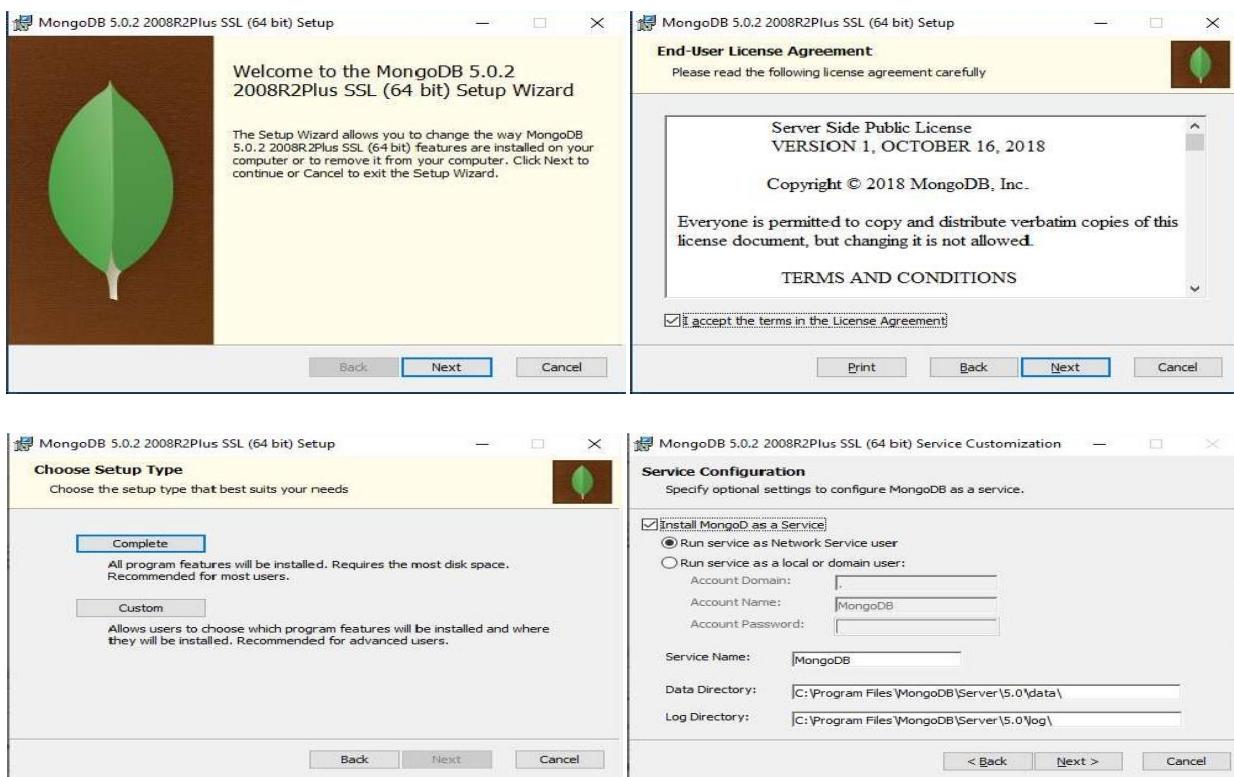
Understand the installation and configuration of NoSQL Databases: MongoDB

### CO3:

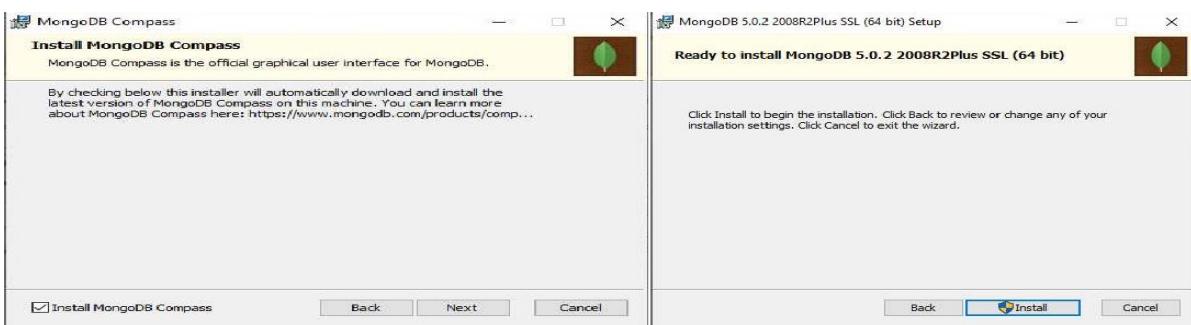
Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.

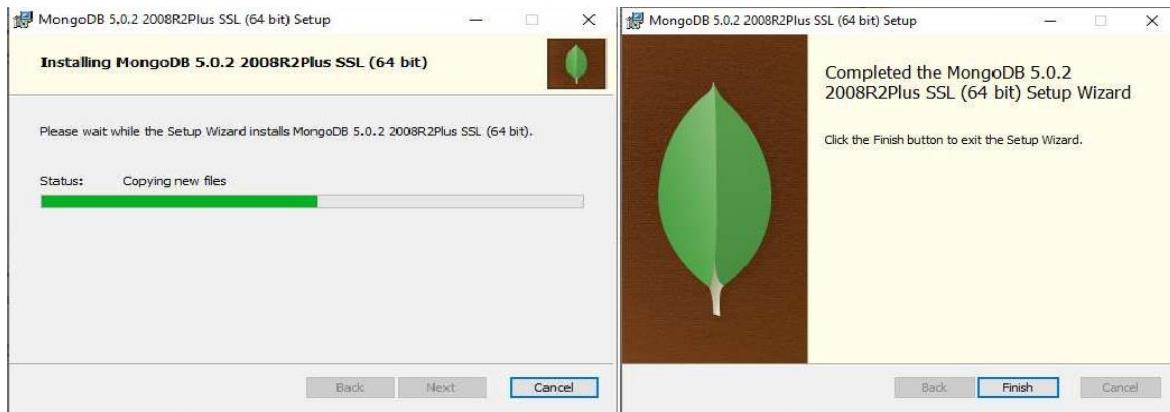
### Procedure

#### 1. Setup Wizard



#### 2. Installing MongoDB Compass

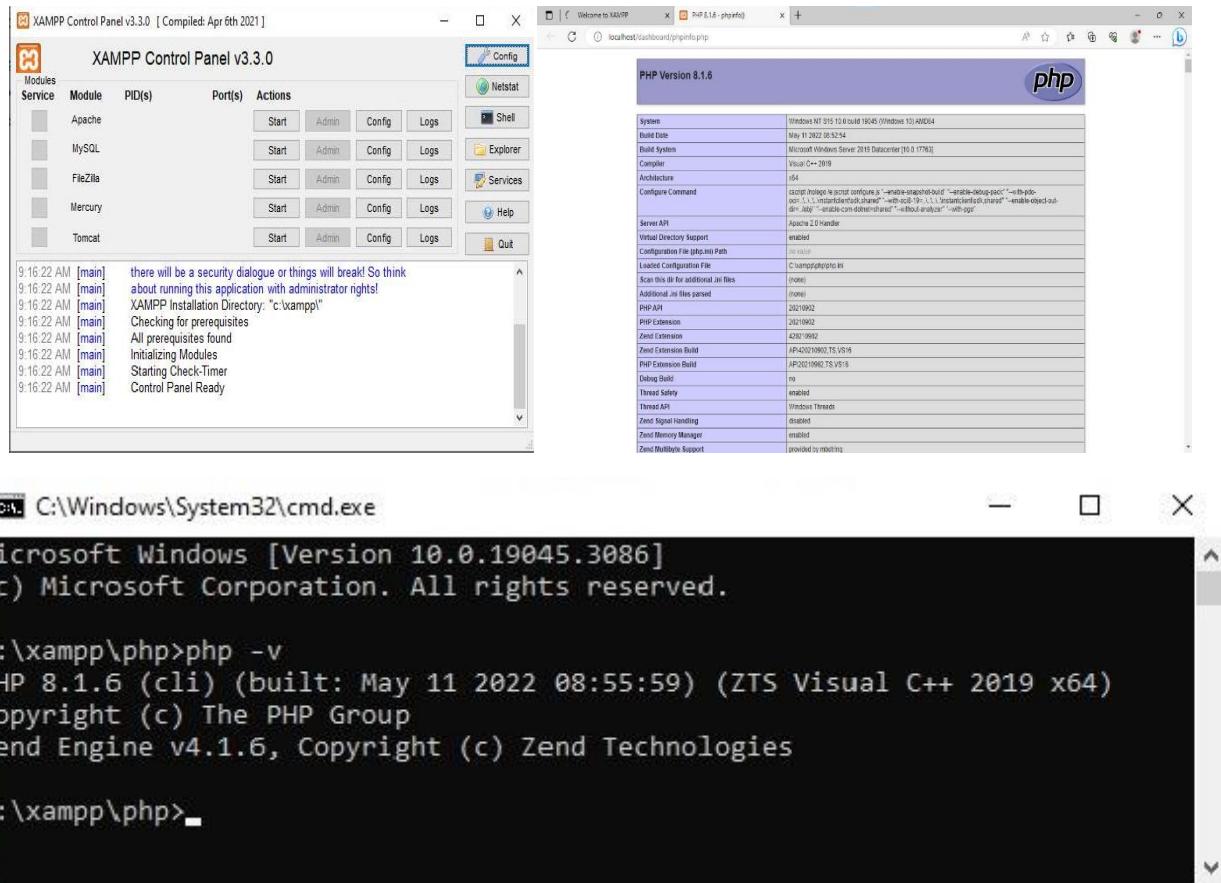




### 3. MongoDB Compass Home Page

The screenshot shows the MongoDB Compass application interface. The top part features a 'Welcome to Compass' dialog box with a 'Start' button. To the right, a 'New Connection' dialog box is open, showing a connection string 'mongodb://localhost:27017/'. The main workspace displays three databases: 'admin', 'config', and 'local'. Each database card provides details like storage size, collections, and indexes. At the bottom, a terminal window shows the prompt '>\_MONGOSH'.

#### 4. Configuring MongoDB to Connect to PHP



#### 5. Installing MongoDB Package for PHP

The screenshot shows the PECL package manager interface for the MongoDB driver:

- Documentation:** Includes links for Home, News, Support, Browse Packages, Search Packages, and Download Statistics.
- Package Information:**
  - Maintainers:** Jeremy Mikola (lead), Katherine Walker (developer), Andreas Brauer (lead), Derick Rethans (lead at phpb.dot.net), Hannes Magnusson (biori at phpb.dot.net) (lead).
  - License:** Apache License.
  - Description:** The purpose of this driver is to provide exceptionally thin glue between MongoDB and PHP, implementing only fundamental and performance-critical components necessary to build a fully-functional MongoDB driver.
  - Homepage:** <http://docs.mongodb.org/ecosystem/drivers/php/>
- Available Releases:** A table listing MongoDB versions and their details:

Version	State	Release Date	Downloads
1.16.1	stable	2023-06-22	<a href="#">mongodb-1.16.1.tgz (1862.9kB)</a>
1.16.0	stable	2023-06-22	<a href="#">mongodb-1.16.0.tgz (1521.9kB)</a>
1.15.3	stable	2023-05-12	<a href="#">mongodb-1.15.3.tgz (1701.9kB)</a>
1.15.2	stable	2023-04-21	<a href="#">mongodb-1.15.2.tgz (1702.1kB)</a>
1.15.1	stable	2023-02-09	<a href="#">mongodb-1.15.1.tar (1701.4kB)</a>

## 6. Configuring Apache

File Home Share View

File > This PC > Local Disk (C:) >xampp > php > ext

Name	Date modified	Type	Size
php_bs2.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_curd.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_fti.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_idn.dll	5/11/2022 2:59 PM	Application exten...	232 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_myqli.dll	5/11/2022 2:59 PM	Application exten...	119 KB
php_ocall_19.dll	5/11/2022 2:59 PM	Application exten...	193 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...	826 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]

File Edit Format View Help

extension=mbstring

extension=zip ; Must be after mbstring as it depends on it

extension=mysqli ; Use with Oracle Database 12c Instant Client

extension=oci19 ; Use with Oracle Database 19 Instant Client

extension=odbc

extension=pdo

extension=pdo\_dblib

extension=pdo\_firebird

extension=pdo\_oci

extension=pdo\_oci8

extension=pdo\_pgsql

extension=pdo\_sqlite

extension=png

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 (CR/LF) UTF-8

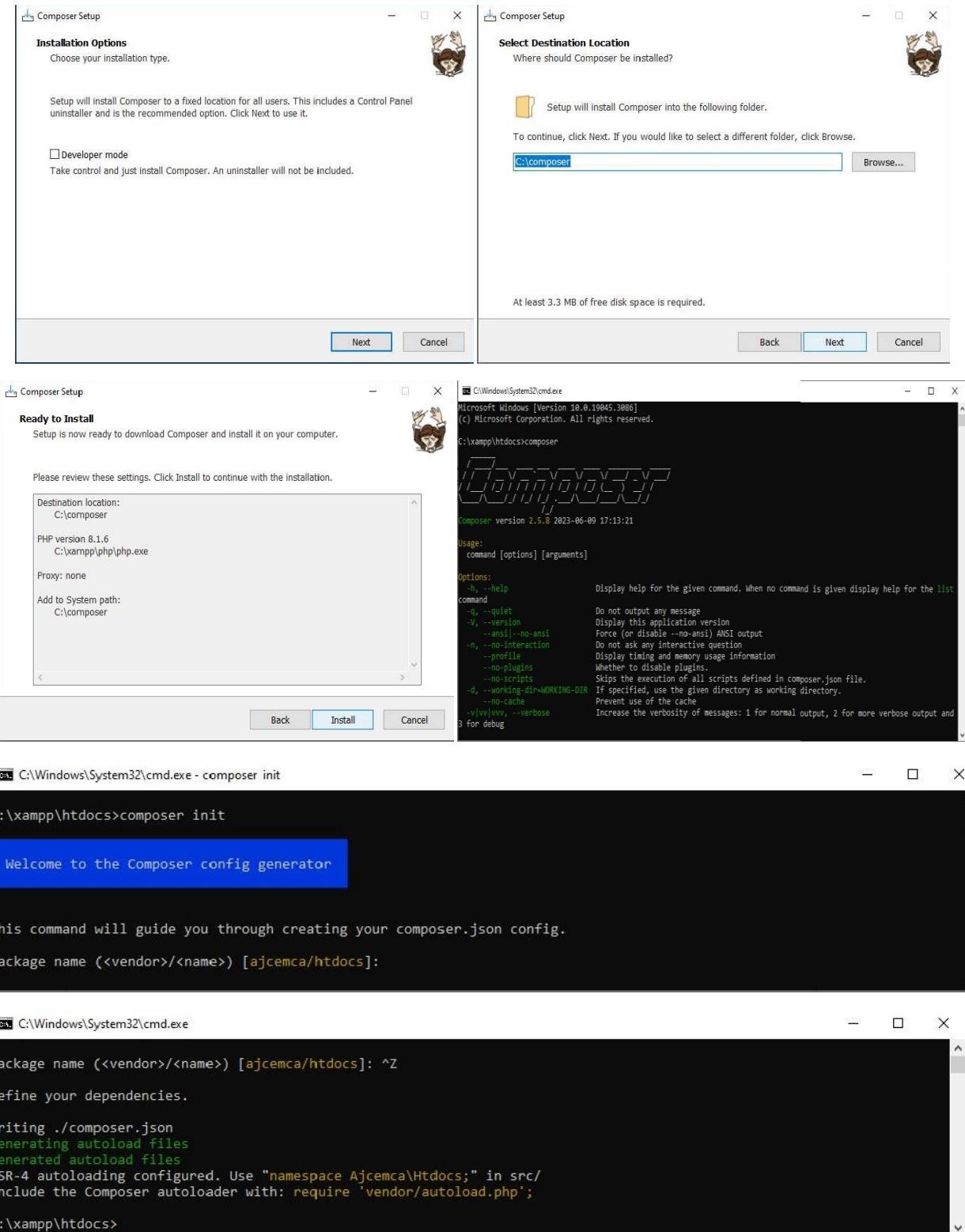
Directive	Local Value	Master Value
mbstring.regex_stack_limit	1000000	1000000
mbstring.stack_limit	1000000	1000000
mbstring.strict_detection	Off	Off
mbstring.substitute_character	no value	no value

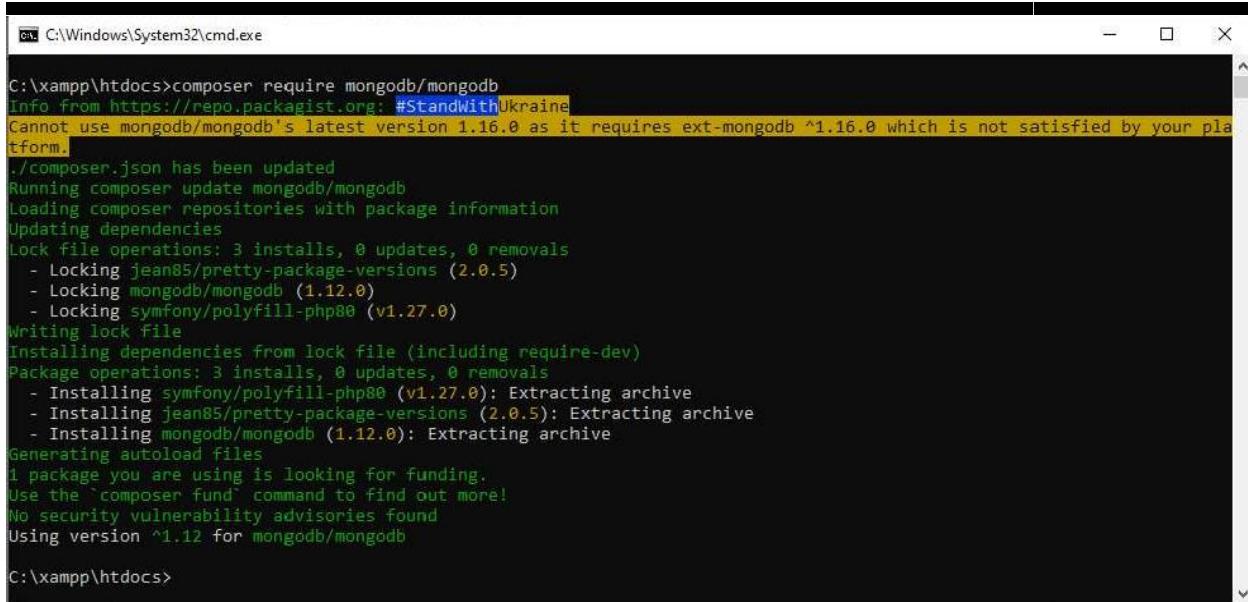
**mongoDB**

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**





```
C:\xampp\htdocs>composer require mongodb/mongodb
Info from https://repo.packagist.org: #StandWithUkraine
Cannot use mongodb/mongodb's latest version 1.16.0 as it requires ext-mongodb ^1.16.0 which is not satisfied by your platform.
./composer.json has been updated
Running composer update mongodb/mongodb
Loading composer repositories with package information
Updating dependencies
Lock file operations: 3 installs, 0 updates, 0 removals
- Locking jean85/pretty-package-versions (2.0.5)
- Locking mongodb/mongodb (1.12.0)
- Locking symfony/polyfill-php80 (v1.27.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 3 installs, 0 updates, 0 removals
- Installing symfony/polyfill-php80 (v1.27.0): Extracting archive
- Installing jean85/pretty-package-versions (2.0.5): Extracting archive
- Installing mongodb/mongodb (1.12.0): Extracting archive
Generating autoload files
1 package you are using is looking for funding.
Use the 'composer fund' command to find out more!
No security vulnerability advisories found
Using version ^1.12 for mongodb/mongodb

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

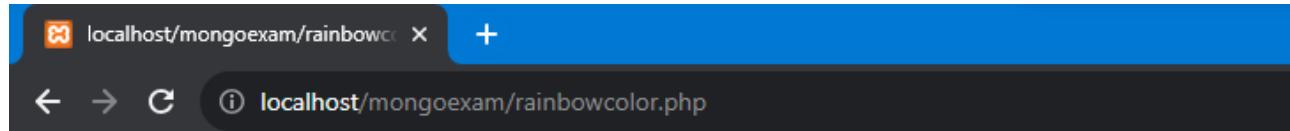
### **CO4:**

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("Rainbow");
echo "Collection Creation Successful !!!";
echo "<br>";
$collection = $db -> Rainbow;
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 "Data Inserted !!!";
echo "<br>";
?>
```

## Output



Connection Successful !!!  
 Database Creation Successful !!!  
 Collection Creation Successful !!!  
 Collection Selected Successful !!!  
 Date Inserted !!!

_id	color
ObjectId('64ea0dd2082c5af4b80ae569')	Violet
ObjectId('64ea0dd2082c5af4b80ae56a')	Indigo
ObjectId('64ea0dd2082c5af4b80ae56b')	Blue
ObjectId('64ea0dd2082c5af4b80ae56c')	Green
ObjectId('64ea0dd2082c5af4b80ae56d')	Yellow

## 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 "<br>";
    $db = $con->Contact;
    $collection = $db->profile;
?>
```

#### Register.php

```
<html>
    <head>
        <title>Registration Form</title>
        <link rel="stylesheet" href="bootstrap/css/bootstrap.css">
    </head>
    <body>
        <center>
            <br><br>
            <h1 class="font-weight-bold">Registration Form</h1>
            <br>

            <div>
                <table>
                    <form action="insert.php" method="post">
                        <tr>
                            <td><input type="text" style="width: 400px;" name="name" placeholder="Full Name" class="form-control" required></td>
                        </tr>
                        <tr>
                            <td><input type="email" name="email" placeholder="Email" class="form-control" required></td>
                        </tr>
                        <tr>
                            <td><input type="text" name="place" placeholder="Place" class="form-control" required></td>
                        </tr>
                    </form>
                </table>
            </div>
        </center>
    </body>
</html>
```

```

        </tr>
        <tr>
            <td></td>
        </tr>
        <tr>
            <td>
                <input type="submit" name="submit" class="btn btn-success btn-lg float-left" value="Register">
            </td>
            <form action="select.php" method="get">
                <input type="submit" name="viewButton" class="btn btn-warning ml-2 btn-lg float-right" value="View Data">
            </form>
        </td>
    </tr>
</table>
</div>
</center>
</body>
</html>

```

**Select.php**

```

<html>
    <head>
        <link rel="stylesheet" href="bootstrap/css/bootstrap.css">
    </head>
</html>
<?php
include'connection.php';
$result = $collection->find(); //All Data are Stored into 'result'
?>

<center>
<h1 class="font-weight-bold">INFORMATIONS REGISTERED</h1>
<br>
<table border="2" class="table table-hover" style="width:80%">
<tr>
<th>NAME</th>
<th>E-MAIL</th>
<th>PLACE</th>
<th>EDIT</th>
<th>DELETE</th>
</tr>
<?php
    foreach ($result as $i){
?>
    <tr>
        <td> <?php echo $i['name']?> </td>

```

```

<td> <?php echo $i['email']?> </td>
<td> <?php echo $i['place']?> </td>
<td>
    <a href ="<?php echo "delete.php?id=$i[_id]";?>" 
    onClick="return confirm ('Are You sure you want to Delete the Data ?')">Delete</a>
</td>
<td>
    <a href ="<?php echo "edit.php?id=$i[_id]";?>">Edit</a>
</td>
</tr>
<?php
    }
?>
</table>
<br>
<form action="register.php" method="get">
    <input type="submit" name="viewButton" class="btn btn-danger" value="Register New
Data">
</form>

```

## Edit.php

```

<?php
include "connection.php";

$id = $_GET['id'];
$objectId = new MongoDB\BSON\ObjectId($id);
$item = $collection->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 = $collection->updateOne($myid, $updateData);
    header("Location: select.php");
}

?>
<html>
    <head>
        <link rel="stylesheet" href="bootstrap/css/bootstrap.css">
    </head>
    <body>

```

```

<center>
    <h1 class="font-weight-bold" style="font-family: Comic Sans MS, Comic Sans, cursive;">
Edit Info</h1>
    <br>
    <table class="table table-success table-striped" style="width: 500px; border-radius: 25px;">
        <form method="POST" action="#">
            <tr>
                <td><label for="name">Full Name:</label></td>
                <td><input type="text" class="form-control" value=<?php echo $item["name"]; ?>" name="name" id="name" required></td>
            </tr>
            <tr>
                <td><label for="email">Email:</label></td>
                <td><input type="email" class="form-control" value=<?php echo $item["email"]; ?>" name="email" id="email" required></td>
            </tr>
            <tr>
                <td><label for="name">Place:</label></td>
                <td><input type="text" class="form-control" value=<?php echo $item["place"]; ?>" name="place" id="place" required></td>
            </tr>
        </table>
        <table width="500px">
            <tr>
                <td><input type="submit" name="submit" class="btn btn-success btn-lg float-left" value="Submit">
                </td>
            </tr>
        </table>
    </center>
</body>
</html>

```

## Insert.php

```

<?php
    include "connection.php";
    if (isset($_POST['submit']))
    {
        $name = $_POST['name'];
        $email = $_POST['email'];
        $place = $_POST['place'];

```

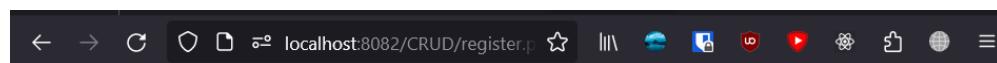
```

echo"<br>Data Inserted";
echo "<br>";
echo $name ;
echo "<br>";
echo $email;
echo "<br>";
echo $place;
$data = array('name'=>$name,'place'=>$place,'email'=>$email);

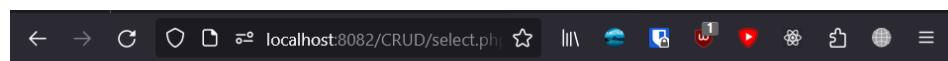
$collection ->insertOne($data);
header("Location: select.php");
}

?>

```

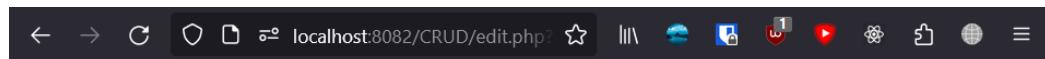
**Output:****Registration Form**

Shon George	
shong@gmail.com	
Kappadu	
<b>Register</b>	<b>View Data</b>

**INFORMATION REGISTERED**

NEME	E-MAIL	PLACE	EDIT	DELETE
Nithin Rajappan	rajappan@gmail.com	Vagamon	<a href="#">Delete</a>	<a href="#">Edit</a>
Nithin Jose	nithin@email.com	Chirakkadavu	<a href="#">Delete</a>	<a href="#">Edit</a>
Shon George	shong@gmail.com	Kappadu	<a href="#">Delete</a>	<a href="#">Edit</a>

[Register New Data](#)



## Edit Info

Full Name:	Nithin Rajappan
Email:	rajappan@gmail.com
Place:	Vagamon

[Submit](#)

[Go Back](#)

NEME	E-MAIL	PLACE	EDIT	DELETE
Nithin Rajappan	rajappan@gmail.com	Vagamon	<a href="#">Delete</a>	<a href="#">Edit</a>
Nithin Jose	nithin@email.com	Chirakkadavu	<a href="#">Delete</a>	<a href="#">Edit</a>
Shon George	shong@gmail.com	Kappadu	<a href="#">Delete</a>	<a href="#">Edit</a>

⊕ localhost:8082  
 Are You sure you want to Delete the Data ?  
[OK](#)   [Cancel](#)

## 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

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> use expmongo
switched to db expmongo
>
```

2. Display current database

>db

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db
expmongo
> -
```

3. Create a collection

>db.createCollection("actors")

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.createCollection("actors")
{ "ok" : 1 }
>
```

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": "schniederova", "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": "schniederova", "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)
```

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find().sort({ name: 1 }).skip(1).limit(2)
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
>
```

### 21. Display

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

```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.actors.find().sort({ name: 1 }).limit(2).skip(1)
{ "_id" : "trojan", "name" : "Ivan Trojan", "year" : 1964, "movies" : [ "samotari", "medvidek" ] }
{ "_id" : "machacek", "name" : "Jiri Machacek", "year" : 1966, "movies" : [ "medvidek", "vratnelahve", "samotari" ] }
>
```

## 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

### C05:

Apply CRUD operations and retrieve data in a NoSQL environment.

### Procedure

#### 1. Create and Use a Database



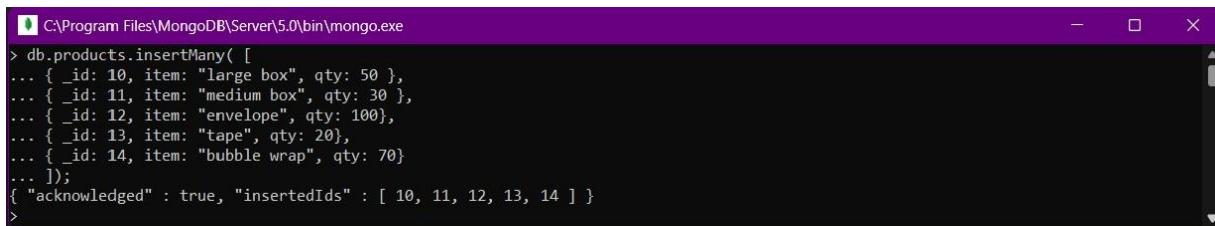
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> use indexdemo;
switched to db indexdemo
> -
```

#### 2. Show Databases



```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.createCollection("indexdm");
{ "ok" : 1 }
> show dbs
Exam          0.000GB
MicroProject  0.000GB
admin          0.000GB
bloodbank      0.000GB
config         0.000GB
indexdemo      0.000GB
local          0.000GB
```

#### 3. Input data



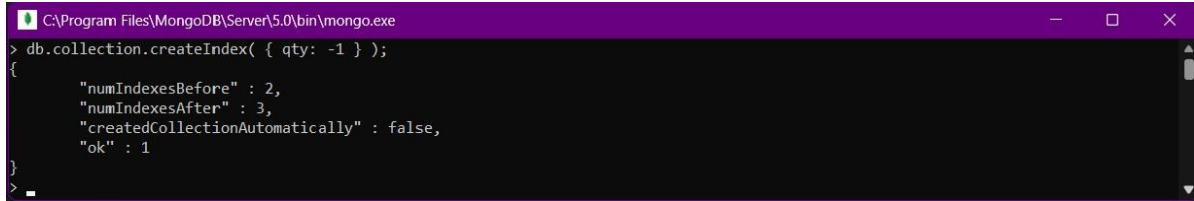
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.products.insertMany([
... { _id: 10, item: "large box", qty: 50 },
... { _id: 11, item: "medium box", qty: 30 },
... { _id: 12, item: "envelope", qty: 100 },
... { _id: 13, item: "tape", qty: 20},
... { _id: 14, item: "bubble wrap", qty: 70}
... ]);
{ "acknowledged" : true, "insertedIds" : [ 10, 11, 12, 13, 14 ] }
```

#### 4. Create ascending index on a field



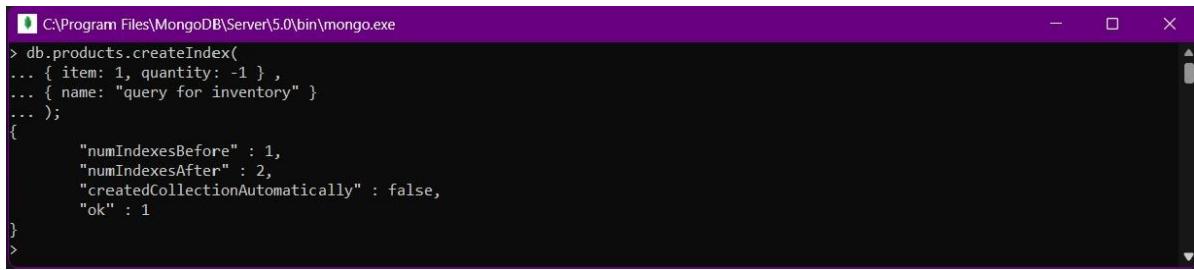
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.collection.createIndex( { item: 1 } );
{
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "createdCollectionAutomatically" : true,
  "ok" : 1
}>
```

### 5. Create descending index on a field



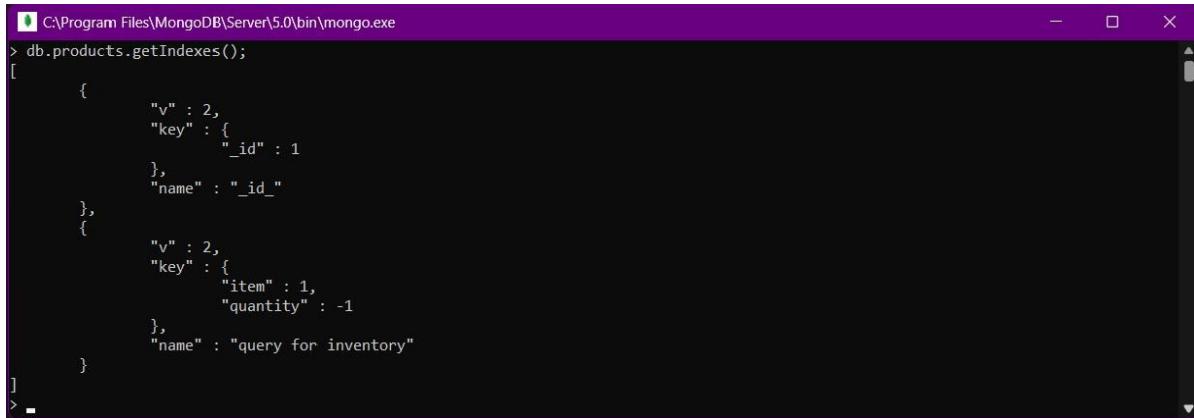
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.collection.createIndex( { qty: -1 } );
{
  "numIndexesBefore" : 2,
  "numIndexesAfter" : 3,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
>
```

### 6. Create index with the index name



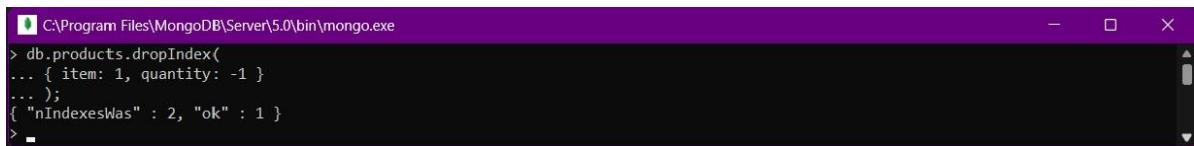
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.products.createIndex(
... { item: 1, quantity: -1 },
... { name: "query for inventory" }
... );
{
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
>
```

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



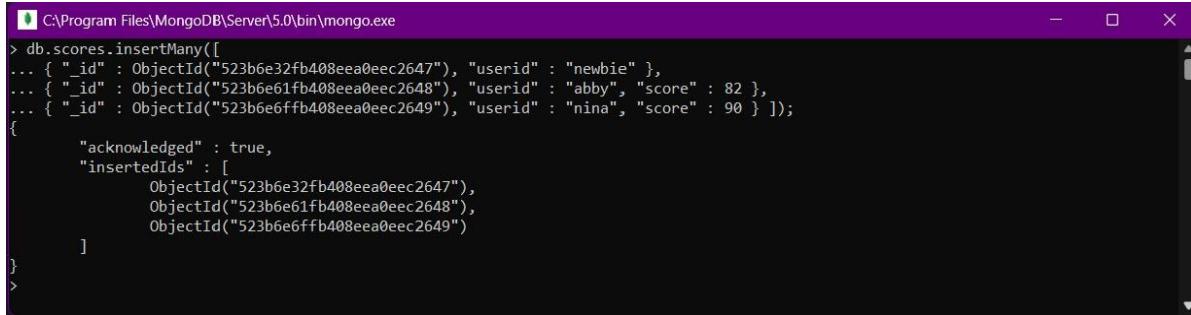
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.products.getIndexes();
[
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "item" : 1,
      "quantity" : -1
    },
    "name" : "query for inventory"
  }
]
>
```

### 8. Drop index by index document



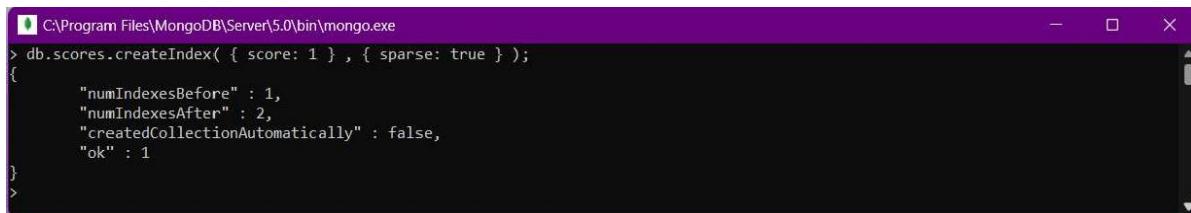
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.products.dropIndex(
... { item: 1, quantity: -1 }
... );
{ "nIndexesWas" : 2, "ok" : 1 }
>
```

## 9. Insert data



```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.scores.insertMany([
... { "_id" : ObjectId("523b6e32fb408eea0eec2647"), "userid" : "newbie" },
... { "_id" : ObjectId("523b6e61fb408eea0eec2648"), "userid" : "abby", "score" : 82 },
... { "_id" : ObjectId("523b6e6ffb408eea0eec2649"), "userid" : "nina", "score" : 90 } ]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("523b6e32fb408eea0eec2647"),
    ObjectId("523b6e61fb408eea0eec2648"),
    ObjectId("523b6e6ffb408eea0eec2649")
  ]
}
>
```

## 10. Create a sparse index



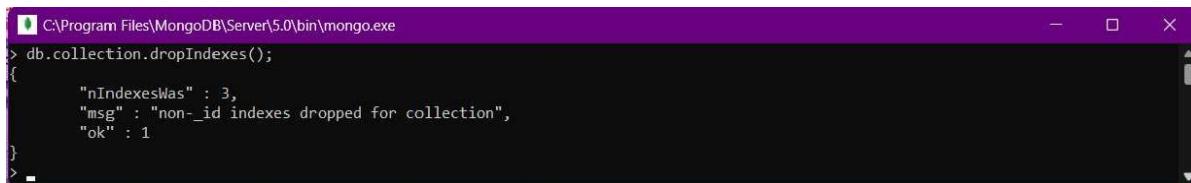
```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.scores.createIndex( { score: 1 } , { sparse: true } );
{
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "createdCollectionAutomatically" : false,
  "ok" : 1
}
>
```

## 11. Use Sparse index on the score field



```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.scores.find( { score: { $lt: 90 } } );
{ "_id" : ObjectId("523b6e61fb408eea0eec2648"), "userid" : "abby", "score" : 82 }
> -
```

## 12. Drop all indexes but \_id index



```
C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe
> db.collection.dropIndexes();
{
  "nIndexesWas" : 3,
  "msg" : "non-_id indexes dropped for collection",
  "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

### CO5:

Apply CRUD operations and retrieve data in a NoSQL environment.

### Procedure

The screenshot shows the MongoDB Atlas cluster creation interface. It displays three plan options: M10 (\$0.08/hour), SERVERLESS (\$0.10/1M reads), and MO (FREE). The MO plan is selected. The MO plan details are: For learning and exploring MongoDB in a cloud environment. Storage: Up to 1TB, RAM: Auto-scale, vCPU: Shared. Below the plans, it shows the provider options (AWS, Google Cloud, Azure) and the recommended region (Mumbai (ap-south-1)). A 'Create' button is present at the bottom left.

The screenshot shows the MongoDB Atlas security quickstart interface. It displays two authentication methods: Username and Password and Certificate. A message indicates an autogenerated username and password was created. It then guides the user to create a database user with a specific privilege and provides fields for Username (nithinjose2024b), Password (SRb9XWm0nulmy7op), and Create User. The sidebar shows various project settings like Deployment, Services, Security, and Quickstart.

The screenshot shows the MongoDB Atlas Security Quickstart interface. On the left sidebar, under the SECURITY section, the 'Quickstart' option is selected. In the main area, there's a form for adding IP addresses to an 'IP Access List'. It includes fields for 'IP Address' (with placeholder 'Enter IP Address'), 'Description' (placeholder 'Enter description'), and a button 'Add My Current IP Address'. Below this is a table showing existing entries in the 'IP Access List': '117.202.227.63/32' and '117.93.79.225/32', each with 'EDIT' and 'REMOVE' buttons. At the bottom right is a 'Finish and Close' button.

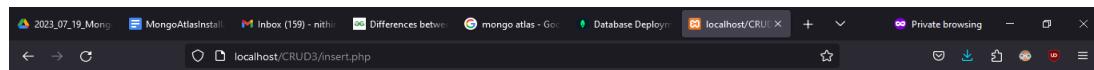
The screenshot shows the MongoDB Atlas Database Deployments interface. The left sidebar shows the 'Deployment' section is selected. The main area displays a 'Database Deployments' dashboard for 'Cluster0'. It features a 'Load sample datasets to Cluster0' section with a 'Load sample dataset' button. Below this are sections for 'Connect' (Cluster0), 'View Monitoring' (with metrics like R: 0, W: 0, Connections: 0, In: 0.0 B/s, Out: 0.0 B/s, Data Size: 6.0 B / 512.0 MB (0%)), and 'Explore Charts'. A 'FREE' badge is visible next to the 'Cluster0' connection button. At the bottom right is a 'Data Size' summary: '6.0 B / 512.0 MB (0%) Last 2 minutes'.

The screenshot shows the 'Connect to Cluster0' interface. It consists of three main steps: 'Set up connection security', 'Choose a connection method', and 'Connect'. The 'Choose a connection method' step is currently active, displaying a list of tools and drivers for connecting to the cluster. These include 'Drivers' (MongoDB native drivers for Node.js, Go, etc.), 'Compass' (GUI for exploring data), 'Shell' (CLI for quick data operations), 'MongoDB for VS Code' (extension for VS Code), and 'Atlas SQL' (SQL tools for data analysis). The background shows the same MongoDB Atlas deployment dashboard as the previous screenshot.

```

connection.php
1 <?php
2 require "./vendor/autoload.php";
3
4 $con = new MongoDB\Client("mongodb+srv://philipantony:4KJva0oBr4nDu9kc@cluster0.26tn2x0.mongodb.net/?retryWrites=true&w=majority");
5 if($con)
6 {
7     echo "Connected";
8 }
9 else{
10    echo "failed";
11 }
12 $db = $con->contact;
13 $collection = $db->profile;
14
15
16 ?>

```



Printing connection String mongodb+srv://nithinjose2024b.SRb9XWm0nulmy7cp@cluster0.zsc03fv.mongodb.net/?retryWrites=true&w=majority  
DB Created Successfully  
Collection Created Successfully

Data Inserted  
Shon G  
shon@gmail.com  
Kappadu



The screenshot shows the Realm App Services interface. On the left, there's a sidebar with sections like DATA ACCESS, BUILD, MANAGE, and HELP. The main area features a heading 'Build better apps faster with App Services' and a sub-section 'Start with an app template'. It lists five templates:

- Build your own App: Set up your own app services to fit your development needs.
- Mobile app using a Realm SDK: Sync data to your backend.
- Manage Database Views: Event-driven Database Trigger template to update a view in a separate collection.
- React.js + Realm Web Boilerplate: Todo web app using the Realm Web SDK.
- GraphQL + React App Boilerplate: Todo web app using the GraphQL API.

**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 **Link your Data Source** Template Starter Applications let you build on top of data in a MongoDB Atlas cluster. Cluster0
- ② **Name your Application** This name will be used internally and cannot be changed later. Application-0
- ③ **App Deployment Model** You can change this later in your Application Settings. View Docs Single Region • Mumbai (ap-south-1) • AWS | Region chosen based on your cluster's region Change

**Recent**

Type here to search

10:32 AM 7/19/2023

**Total App Usage** Jul 01 - Jul 31, Free Tier resets in 13 days. Learn more Last Updated: Jul 19, 2023, 10:32:43 AM GMT-5:30

Requests	Data Transfer	Compute Runtime	Sync Runtime
0 / IM FREE TIER USED	0.00 / 10 GB FREE TIER USED	0.00 / 500 Hr FREE TIER USED	0.00 / 10K Hr FREE TIER USED

**Application-0**

**NO ENVIRONMENT**

Total Users Requests (hr) 0  
0 Success Fail

This Month Last modified: 1 minute ago

Requests 0  
Data Transfer (GB) 0.00  
Compute Runtime (Hr) 0.00  
Sync Runtime (Hr) 0.00

Type here to search

10:32 AM 7/19/2023

**Create an App Service**

- Name** This name will be used internally and cannot be changed later. Application-1 Names must only include ASCII letters, numbers, \_.
- Link your Database** App Services securely stores your application data in your linked MongoDB Atlas cluster(s).
  - We'll automatically create a free-tier 4.0 Atlas cluster for you with 832 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)

**Create App Service**

NITHIN JOSE

Type here to search

10:32 AM 7/19/2023

The screenshot shows the MongoDB Atlas App Services interface for the 'myapp' project. The left sidebar includes sections for DATA ACCESS (Rules, Schema, App Users, Authentication), BUILD (Realm SDKs, Device Sync, GraphQL, Functions), and Apps (myapp). The main content area displays the 'Get Started With App Services' section, which includes four cards: 'REALM CLI' (Enable Local Development with the Command Line Interface), 'SDKs' (Install SDKs and get familiar with the SDK syntax.), 'DOCUMENTATION' (Visit App Services documentation to learn how to build with App Services.), and 'DEPLOYMENT' (Connect to a GitHub account and repo to use Git).

The screenshot shows the MongoDB Atlas App Services interface for the 'NITHIN JOSE'S ORG' project. The left sidebar includes sections for DATA ACCESS (Rules, Schema, App Users, Authentication), BUILD (Realm SDKs, Device Sync, GraphQL, Functions, Triggers), and HTTPS Endpoints (Values, MANAGE, Linked Data Sources, Deployment). The main content area displays the 'Custom HTTPS Endpoints' section, which includes a message about the re-naming of 'HTTP Service Incoming Webhooks' to 'HTTPS Endpoints'. It also features a button to 'Add An Endpoint' and a note explaining that HTTPS Endpoints allow you to execute code in response to HTTP requests from an external app or service.

The screenshot shows the MongoDB Atlas App Services interface for the 'NITHIN JOSE'S ORG' project, specifically the 'Add Endpoint' form for creating a new HTTPS endpoint. The form includes fields for 'Route' (set to '/person'), 'Enabled' (checked), and 'Operation Type' (callback URL: <https://ap-south-1.aws.data.mongodb-api.com/app/application-1-xnoco/endpoint/person>). Below the form, there is a terminal-like interface with a 'curl' command example:

```
curl \
-H "Content-Type: application/json" \
-d '{"foo": "bar"}' \
https://ap-south-1.aws.data.mongodb-api.com/app/application-1-xnoco/endpoint/person
```

**HTTP Method:** POST

**Respond With Result:**

**Return Type:** **JSON** (selected) | EJSON | [VIEW EXAMPLE](#)

**FUNCTION**

**Authentication:** Functions for new Endpoints will use [Application Authentication](#) by default. Authentication can be updated by navigating to the settings page for a particular Function (found on the [Functions](#) page) once the Function is saved.

Select the function to be linked to the endpoint. Selecting a new function will create a default function.

HTTP Method	Endpoint Route	Linked Function	Last Modified	Actions
POST	/person	createPerson	07/19/2023 05:13:10	...

**Custom HTTPS Endpoints**

HTTP Service Incoming Webhooks are now re-named 'HTTPS Endpoints' and will be found in this section of the UI moving forward. All existing HTTP Service Incoming Webhooks can be converted to 'HTTPS Endpoints' and can still be used with no changes in functionality from an end user perspective. To develop and deploy HTTPS Endpoints from your local machine, please install realm-cliv2.3+. [Learn more](#)

## 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

```
C:\Windows\System32\cmd.exe - mongo --port 27018
Microsoft Windows [Version 10.0.19045.3208]
(c) Microsoft Corporation. All rights reserved.

C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replicaSet dataservice -logpath \data\rs1\1.log -dbpath \data\rs1 -port 27018
C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replicaSet dataservice -logpath \data\rs2\2.log -dbpath \data\rs2 -port 27019
C:\Program Files\MongoDB\Server\5.0\bin>start mongod -replicaSet dataservice -logpath \data\rs3\3.log -dbpath \data\rs3 -port 27020

C:\Program Files\MongoDB\Server\5.0\bin>mongo --port 27018
MongoDB shell version v5.0.2
connecting to: mongodb://127.0.0.1:27018/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("6999d461-e370-4fc5-a255-3e9d7dde2467") }
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:
2023-07-26T09:32:53.076+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2023-07-26T09:32:53.077+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()
---
```

#### 2. Configure Replica Set

```
C:\Windows\System32\cmd.exe - mongo --port 27018
> config={_id:"dataservice",members:[{_id:0,host:"localhost:27018"},{_id:1,host:"localhost:27019"},{_id:2,host:"localhost:27020"}]}
{
  "_id" : "dataservice",
  "members" : [
    {
      "_id" : 0,
      "host" : "localhost:27018"
    },
    {
      "_id" : 1,
      "host" : "localhost:27019"
    },
    {
      "_id" : 2,
      "host" : "localhost:27020"
    }
  ]
}
> rs.initiate(config)
{
  "ok" : 1,
  "$clusterTime" : {
    "clusterTime" : Timestamp(1690344907, 1),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  },
  "operationTime" : Timestamp(1690344907, 1)
}
```

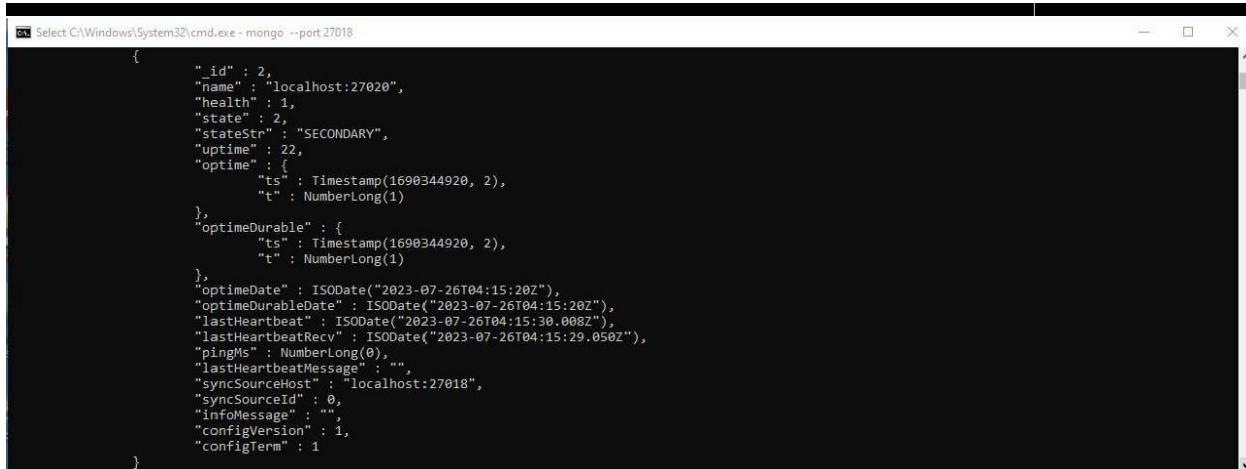
### 3. Replica Set Status

```
C:\Windows\System32\cmd.exe - mongo --port 27018
data:SECONDARY> rs.status()
{
    "set" : "datascience",
    "date" : ISODate("2023-07-26T04:15:30.026Z"),
    "myState" : 1,
    "term" : NumberLong(1),
    "syncSourceHost" : "",
    "syncSourceId" : -1,
    "heartbeatIntervalMillis" : NumberLong(2000),
    "majorityVoteCount" : 2,
    "writeMajorityCount" : 2,
    "votingMembersCount" : 3,
    "writableVotingMembersCount" : 3,
    "optimes" : {
        "lastCommittedOptime" : {
            "ts" : Timestamp(1690344920, 2),
            "t" : NumberLong(1)
        },
        "lastCommittedWallTime" : ISODate("2023-07-26T04:15:20.275Z"),
        "readConcernMajorityOptime" : {
            "ts" : Timestamp(1690344920, 2),
            "t" : NumberLong(1)
        },
        "appliedOptime" : {
            "ts" : Timestamp(1690344920, 2),
            "t" : NumberLong(1)
        },
        "durableOptime" : {
            "ts" : Timestamp(1690344920, 2),
            "t" : NumberLong(1)
        },
        "lastAppliedWallTime" : ISODate("2023-07-26T04:15:20.275Z"),
        "lastDurableWallTime" : ISODate("2023-07-26T04:15:20.275Z")
    },
    "lastStableRecoveryTimestamp" : Timestamp(1690344918, 1),
    "electionCandidateMetrics" : {
        "lastElectionReason" : "electiontimeout",
        "lastElectionDate" : ISODate("2023-07-26T04:15:17.922Z"),
        "electionTerm" : NumberLong(1),
        "lastCommittedOptimeAtElection" : {
            "ts" : Timestamp(0, 0),
            "t" : NumberLong(0)
        }
    }
}
```

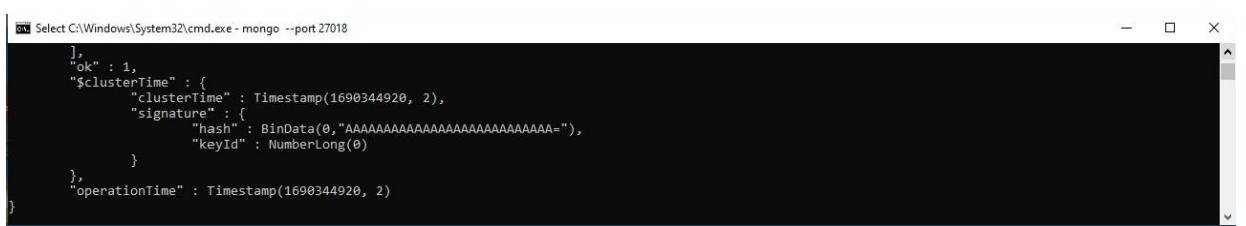
### 4. List Replica Set

```
C:\Windows\System32\cmd.exe - mongo --port 27018
{
    "members" : [
        {
            "_id" : 0,
            "name" : "localhost:27018",
            "health" : 1,
            "state" : 1,
            "stateStr" : "PRIMARY",
            "uptime" : 759,
            "optime" : {
                "ts" : Timestamp(1690344920, 2),
                "t" : NumberLong(1)
            },
            "optimeDate" : ISODate("2023-07-26T04:15:20Z"),
            "syncSourceHost" : "",
            "syncSourceId" : -1,
            "infoMessage" : "Could not find member to sync from",
            "electionTime" : Timestamp(1690344917, 1),
            "electionDate" : ISODate("2023-07-26T04:15:17Z"),
            "configVersion" : 1,
            "configTerm" : 1,
            "self" : true,
            "lastHeartbeatMessage" : ""
        }
    ]
}
```

```
Select C:\Windows\System32\cmd.exe - mongo --port 27018
{
    "_id" : 1,
    "name" : "localhost:27019",
    "health" : 1,
    "state" : 2,
    "stateStr" : "SECONDARY",
    "uptime" : 22,
    "optime" : {
        "ts" : Timestamp(1690344920, 2),
        "t" : NumberLong(1)
    },
    "optimeDurable" : {
        "ts" : Timestamp(1690344920, 2),
        "t" : NumberLong(1)
    },
    "optimeDate" : ISODate("2023-07-26T04:15:20Z"),
    "optimeDurableDate" : ISODate("2023-07-26T04:15:20Z"),
    "lastheartbeat" : ISODate("2023-07-26T04:15:30.008Z"),
    "lastheartbeatRecv" : ISODate("2023-07-26T04:15:29.078Z"),
    "pingMs" : NumberLong(0),
    "lastheartbeatMessage" : "",
    "syncSourceHost" : "localhost:27018",
    "syncSourceId" : 0,
    "infoMessage" : "",
    "configVersion" : 1,
    "configTerm" : 1
},
```

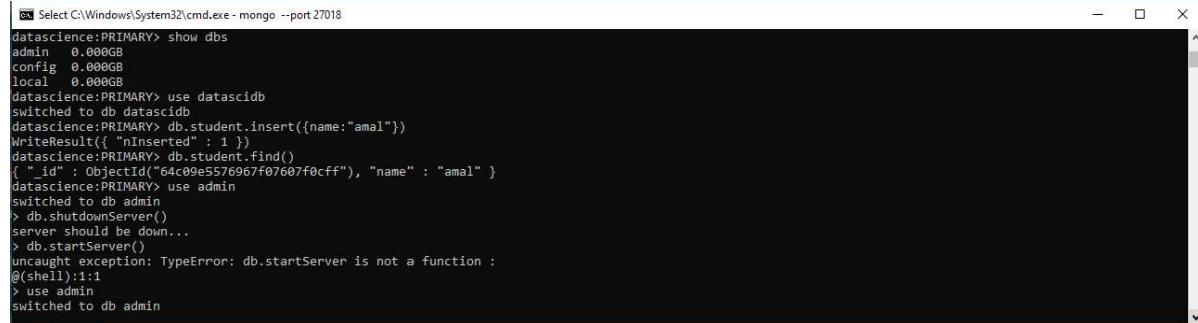


```
PS Select C:\Windows\System32\cmd.exe - mongo --port 27020
{
    "_id" : 2,
    "name" : "localhost:27020",
    "health" : 1,
    "state" : 2,
    "stateStr" : "SECONDARY",
    "uptime" : 22,
    "optime" : {
        "ts" : Timestamp(1690344920, 2),
        "t" : NumberLong(1)
    },
    "optimeDurable" : {
        "ts" : Timestamp(1690344920, 2),
        "t" : NumberLong(1)
    },
    "optimeDate" : ISODate("2023-07-26T04:15:20Z"),
    "optimeDurableDate" : ISODate("2023-07-26T04:15:20Z"),
    "lastHeartbeat" : ISODate("2023-07-26T04:15:30.008Z"),
    "lastHeartbeatRecv" : ISODate("2023-07-26T04:15:29.050Z"),
    "pingMs" : NumberLong(0),
    "lastHeartbeatMessage" : "",
    "syncSourceHost" : "localhost:27018",
    "syncSourceId" : 0,
    "infoMessage" : "",
    "configVersion" : 1,
    "configTerm" : 1
}
```

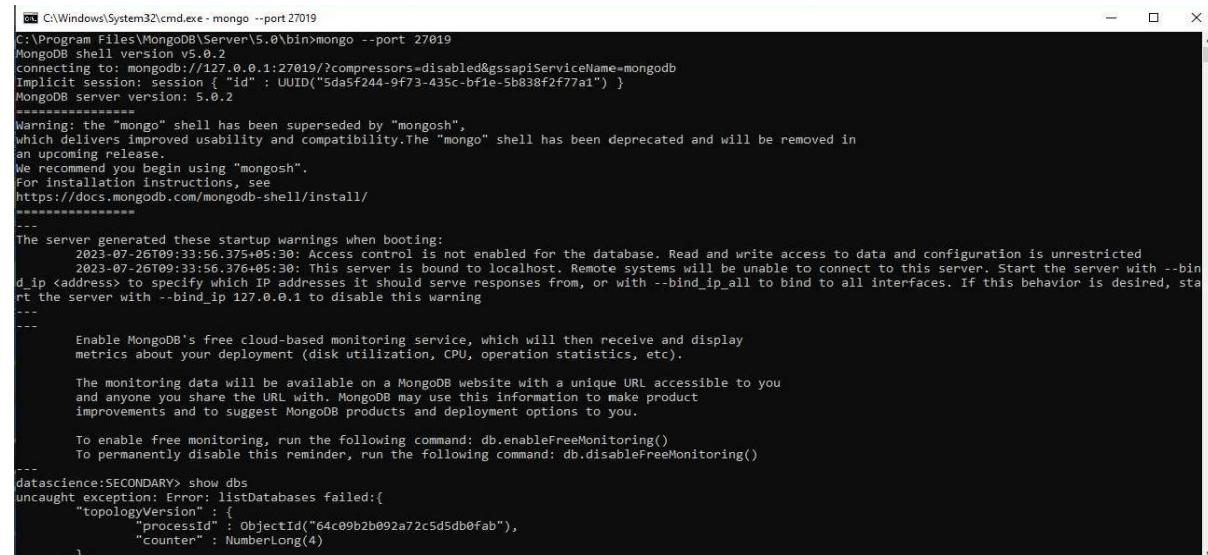
```
PS Select C:\Windows\System32\cmd.exe - mongo --port 27018
{
    "ok" : 1,
    "$clusterTime" : {
        "clusterTime" : Timestamp(1690344920, 2),
        "signature" : {
            "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAA="),
            "keyId" : NumberLong(0)
        }
    },
    "operationTime" : Timestamp(1690344920, 2)
}
```

## 1. Performing operations at Primary



```
PS 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
```

## 2. Performing operations at Secondary



```
PS 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: 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 --bind_ip
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, 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> 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,"AAAAAAAAAAAAAAAAAAAAAA="),
        "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.getBS@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:i1:
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,
})
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
d, 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,"AAAAAAAAAAAAAAAAAAAAAAA="),
            "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("64c0a1e23151b8630c0e8860"), "name" : "akhil" }
dataScience:SECONDARY>
```

## Result

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