20MCA134 - ADVANCED DATABASE MANAGEMENT SYSTEM LAB

Lab Report Submitted By

SANIO LUKE SEBASTIAN

Reg. No.: AJC21MCA-2093

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. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2021-2022

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Lab report, **"20MCA134 ADVANCED DATABASE MANAGEMENT SYSTEM LAB"** is the bonafide work of **SANIO LUKE SEBASTIAN (AJC21MCA-2093)** 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 2021-22.

Sr. Elsin Chakkalackal
Lab In-Charge

Rev. Fr. Dr. Rubin Thottupurathu Jose
Head of the Department

Internal Examiner

External Examiner

CONTENT

Program No: 01

Name: Sanio Luke Sebastian

Roll No: 35

Batch: B

Date: 08-04-2022

Aim: To study various DDL commands – CREATE, ALTER, DROP, TRUNGATE, RENAME.

Questions:

- ***CREATE***

1. Table1: Deposit
   - Actno varchar2(5) primary key, first letter must start with 'D' cname varchar2(15) foreign key references customer
   - Bname varchar2(20) foreign key references branch amount number (8,2) not null, cannot be 0
   - Adate date

2. Table 2: Branch
   - Bname varchar2(20) primary key
   - City varchar2(30) not null, any one of Nagpur, Delhi, Bangalore, Bombay

3. Table 3: Customer
   - Cname varchar2(15) primary key,
   - City varchar2(20) not null

4. Table 4: Borrow
   - loanno varchar2(8) primary key / first letter must start with 'L' cname varchar2(15) foreign key references customer
   - bname varchar2(20) foreign key references branch amount number (8,2) not null, cannot be 0

- ***ALTER, DROP, TRUNGATE, RENAME***

1. Create a table emp with attributes empno number (4) as primary key, ename char (10), hiredate, salary, commission. And insert the given 5 rows of data.
2. Modifying the structure of tables
   a. Add new columns: sal number(7,2)
   b. Dropping a column from a table: sal
   c. Modifying existing column: ename varchar2(15)
   d. Renaming the tables: emp to emp1
   e. Truncating the tables:emp1

f. Destroying tables:emp

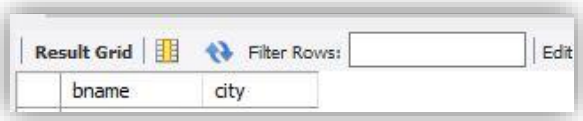
3. Create a table stud with sname varchar2(20) primary key , rollno number(10) not null,dob date not null
4. Create a table student as regno number (6), mark number (3) check constraint (mark >=0 and mark <=100));
5. Create a table cust with(custid number(6) constraint unique, name char(10)

Procedure & Outputs:

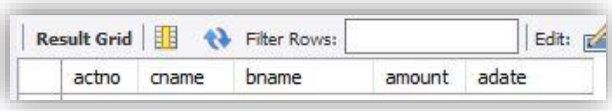
- ***CREATE***

1. CREATE TABLE customer (cname VARCHAR (15), city VARCHAR (20) NOT NULL, PRIMARY KEY (cname));

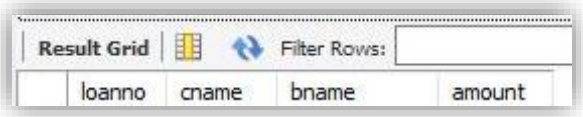
2. CREATE TABLE branch (bname VARCHAR (15), city VARCHAR (20) NOT NULL, PRIMARY KEY (bname), CHECK (city IN ('Nagpur', 'Delhi', 'Bangalore', 'Bombay'));



3. CREATE TABLE deposit (actno VARCHAR (25), cname VARCHAR (15), bname VARCHAR (20), amount BIGINT NOT NULL, adate DATE, check (actno like 'D%'), PRIMARY KEY (actno), FOREIGN KEY (cname) REFERENCES customer (cname), FOREIGN KEY (bname) REFERENCES branch (bname), CHECK (amount > 0));



4. CREATE TABLE borrow (loanno VARCHAR (25), cname VARCHAR (15), bname VARCHAR (20), amount FLOAT (8, 2) NOT NULL, CHECK (loanno LIKE 'L%'), PRIMARY KEY (loanno), FOREIGN KEY (cname) REFERENCES customer (cname), FOREIGN KEY (bname) REFERENCES branch (bname), CHECK (amount > 0));



- ***ALTER, DROP, TRUNGATE, RENAME***

create database lab_cycle_three;
use lab_cycle_three;

1. CREATE TABLE emp (empno int(5), ename varchar(10), hiredate date, salary bigint, commission bigint, PRIMARY KEY (empno));

   INSERT INTO emp VALUES (101,'Ramesh','1980-01-17',5000,null), (102,'Ajay','1985-07-05',5000,500), (103,'Ravi','1981-08-12',1500,null), (104,'Nikesh','1983-03-03',3000,700), (105,'Ravi','1985-07-05',3000,null);

   select * from emp1;

Result Grid | Filter Rows: | Edit:

empno	ename	hiredate	salary	commission
101	Ramesh	1980-01-17	5000	NULL
102	Ajay	1985-07-05	5000	500
103	Ravi	1981-08-12	1500	NULL
104	Nikesh	1983-03-03	3000	700
105	Ravi	1985-07-05	3000	NULL
NULL	NULL	NULL	NULL	NULL

2.
   a. alter table emp add sal numeric(7,2);

Result Grid | Filter Rows: | Edit:

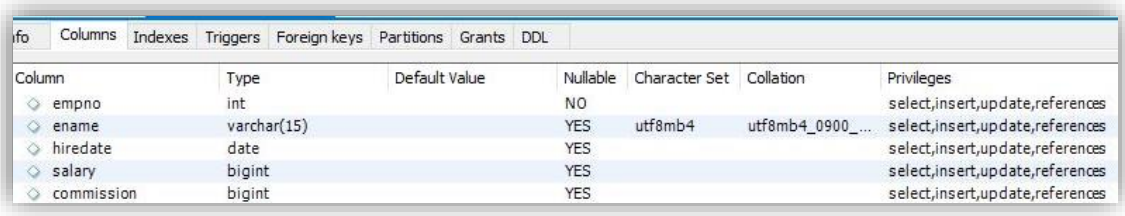
empno	ename	hiredate	salary	commission	sal
101	Ramesh	1980-01-17	5000	NULL	NULL
102	Ajay	1985-07-05	5000	500	NULL
103	Ravi	1981-08-12	1500	NULL	NULL
104	Nikesh	1983-03-03	3000	700	NULL
105	Ravi	1985-07-05	3000	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL

   b. alter table emp drop sal;

Result Grid | Filter Rows: | Edit:

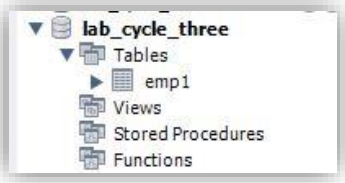
empno	ename	hiredate	salary	commission
101	Ramesh	1980-01-17	5000	NULL
102	Ajay	1985-07-05	5000	500
103	Ravi	1981-08-12	1500	NULL
104	Nikesh	1983-03-03	3000	700
105	Ravi	1985-07-05	3000	NULL
NULL	NULL	NULL	NULL	NULL

c. alter table emp modify column ename varchar(15);

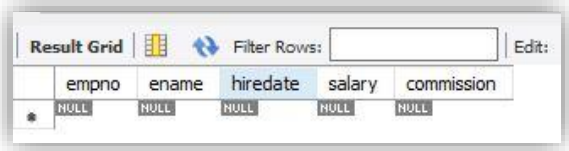


Column	Type	Default Value	Nullable	Character Set	Collation	Privileges
empno	int		NO			select,insert,update,references
ename	varchar(15)		YES	utf8mb4	utf8mb4_0900_...	select,insert,update,references
hiredate	date		YES			select,insert,update,references
salary	bigint		YES			select,insert,update,references
commission	bigint		YES			select,insert,update,references

d. alter table emp rename emp1;

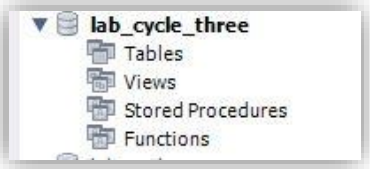


e. truncate table emp1;

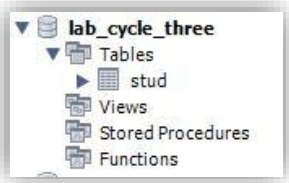


empno	ename	hiredate	salary	commission
NULL	NULL	NULL	NULL	NULL

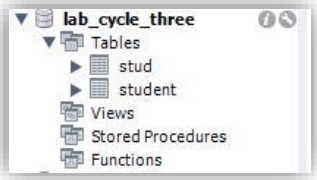
f. drop table emp1;



3. CREATE TABLE stud (sname varchar(20), rollno numeric(10) not null, dob date not null, PRIMARY KEY (sname));



4. CREATE TABLE student (regno numeric(6), mark numeric(3), check(mark >= 0 and mark <= 100));



Program No: 02

Name: Sanio Luke Sebastian

Roll No: 35

Batch: B

Date: 25-03-2022

Aim: To familiarize DML commands SELECT, INSERT, UPDATE, DELETE.

Questions:

- ***SELECT***

1. List all data from table deposit
2. List all data from borrow
3. List all data from customer
4. List all data from branch
5. Give account no and amount of deposite
6. Give customer name and account no of depositors
7. Give name of customers
8. Give name of branches
9. Give name of borrows
10. Give names of customer living in city Nagpur
11. Give names of depositors having amount greater than 4000
12. Give account date of Anil
13. Give name of all branches located in Bombay
14. Give name of borrower having loan number l205
15. Give names of depositors having account at VRCE
16. Give names of all branched located in city Delhi
17. Give name of the customers who opened account date ‘1-12-96’
18. Give account no and deposit number of customers having account opened between dates ‘1-12-96’ and ‘1-5-96’
19. Give name of the city where branch KAROLBAGH is located
20. Give details of customer ANIL

- ***INSERT, UPDATE, DELETE***

1. Inserting values to Branch
2. Inserting values into Customer table
3. Inserting values into Deposit table
4. Inserting values into borrow table
5. Update the branch table
6. Delete the borrow table

Procedure & Outputs:

- ***SELECT***

1. select * from deposit;

Result Grid | Filter Rows: | Edit:

actno	cname	bname	amount	adate
D100	Anil	VRCE	1000	1995-03-01
D101	Sunil	Ajni	5000	1996-12-01
D102	Mehul	Karolbagh	3500	1995-11-17
D104	Madhuri	Chandni	1200	1995-12-17
D105	Pramod	MG road	3000	1996-03-27
D106	Sandip	Andheri	2000	1996-03-31

2. select * from borrow;

Result Grid | Filter Rows:

loanno	cname	bname	amount
L105	Pramod	MG road	3000.00
L201	Anil	VRCE	1000.00
L206	Mehul	Ajni	5000.00
L311	Sunil	Dharampeth	3000.00
L321	Madhuri	Andheri	1200.00

3. select * from customer;

Result Grid | Filter Rows:

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. select * from branch;

Result Grid | Filter Rows:

bname	city
Ajni	Nagpur
Andheri	Bombay
Chandni	Delhi
Dharampeth	Nagpur
Karolbagh	Delhi
MG road	Bangalore
Nehru Palace	Delhi
Powai	Bombay
VRCE	Nagpur
NULL	NULL

5. select actno,amount from deposit;

Result Grid | Filter Row

actno	amount
D100	1000
D101	5000
D102	3500
D104	1200
D105	3000
D106	2000
D107	1000
D108	5000
D109	7000
NULL	NULL

5000

6. select cname,actno from deposit;

Result Grid | Filt

cname	actno
Anil	D100
Kranti	D108
Madhuri	D104
Mehul	D102
Naren	D109
Pramod	D105
Sandip	D106
Shivani	D107
Sunil	D101
NULL	NULL

7. select cname from customer;

Result Grid | Filter

cname
Anil
Kranti
Madhuri
Mandar

8. select bname from branch;

Result Grid

bname
Ajni
Andheri
Chandni
Dharampeth
Karolbagh
MG road
Nehru Palace
Powai
VRCE
NULL

9. select cname from borrow;

Result Grid

cname
Anil
Kranti
Madhuri
Mehul
Pramod
Sunil

10. select cname from customer where city='Nagpur';

Result Grid

cname
Madhuri
Pramod
NULL

11. select cname from deposit where amount > 4000;

Result Grid

cname
Sunil
Kranti
Naren

12. select adate from deposit where cname='Anil';

Result Grid

adate
1995-03-01

13. select bname from branch where city='Bombay';

Result Grid

bname
Andheri
Powai
NULL

14. select cname from borrow where loanno='L201';

Result Grid

cname
Anil

15. select cname from deposit where bname='VRCE';

Result Grid

cname
Anil

16. select bname from branch where city='Delhi';

Result Grid

bname
Chandni
Karolbagh
Nehru Palace
NULL

17. select cname from deposit where adate='1996-12-01';

Result Grid

cname
Sunil

18. select cname from deposit where adate between '1996-05-01' and '1996-12-01';

Result Grid

cname
Sunil

19. select city from branch where bname='karolbagh';

Result Grid

city
Delhi

20. select * from customer join borrow on customer.cname=borrow.cname join deposit on deposit.cname=borrow.cname WHERE customer.cname='Anil';

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

cname	city	loanno	cname	bname	amount	actno	cname	bname	amount	adate
Anil	Calcutta	L201	Anil	VRCE	1000.00	D100	Anil	VRCE	1000	1995-03-01

- ***INSERT, UPDATE, DELETE***

1. INSERT INTO branch VALUES ('VRCE','Nagpur'), ('Ajni','Nagpur'), ('Karolbagh','Delhi'), ('Chandni','Delhi'), ('Dharampeth','Nagpur'), ('MG road', 'Bangalore'), ('Andheri', 'Bombay'), ('Nehru Palace', 'Delhi'), ('Powai','Bombay');

Result Grid | Filter Rows: | Edit:

loanno	cname	bname	amount
L105	Pramod	MG road	3000.00
L201	Anil	VRCE	1000.00
L206	Mehul	Ajni	5000.00
L311	Sunil	Dharampeth	3000.00
L321	Madhuri	Andheri	1200.00
L481	Kranti	Nehru Palace	5000.00
NULL	NULL	NULL	NULL

2. INSERT INTO customer VALUES ('Anil','Calcutta'), ('Sunil', 'Delhi'), ('Mehul','Baroda'), ('Mandar','Patna'), ('Madhuri','Nagpur'), ('Pramod','Nagpur'), ('Sandip','Surat'), ('Shivani','Bombay'), ('Kranti','Bombay'), ('Naren','Bombay');

Result Grid | Filter Rows: | Edit

bname	city
Ajni	Nagpur
Andheri	Bombay
Chandni	Delhi
Dharampeth	Nagpur
Karolbagh	Delhi
MG road	Bangalore
Nehru Palace	Delhi
Powai	Bombay
VRCE	Nagpur
NULL	NULL

3. INSERT INTO deposit VALUES ('D100','Anil','VRCE',1000.00,'1995-03-01'), ('D101','Sunil','Ajni',5000.00,'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','Andheri',1000.00,'1995-09-05'), ('D108','Kranti','Nehru Palace',5000.00,'1995-07-02'), ('D109','Naren','Powai',7000.00,'1995-08-10');

Result Grid | Filter Rows: | Edit:

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. INSERT INTO borrow VALUES ('L201','Anil','VRCE',1000.00), ('L206','Mehul','Ajni',5000.00), ('L311','Sunil','Dharampeth',3000.00), ('L321','Madhuri','Andheri',1200.00), ('L105','Pramod','MG road',3000.00), ('L481','Kranti','Nehru Palace',5000.00);

Result Grid | Filter Rows: | Edit: | Export/Import:

actno	cname	bname	amount	adate
D100	Anil	VRCE	1000	1995-03-01
D101	Sunil	Ajni	500	1996-01-04
D102	Mehul	Karolbagh	3500	1995-11-17
D104	Madhuri	Chandni	1200	1995-12-17
D105	Pramod	MG road	3000	1996-03-27
D106	Sandip	Andheri	2000	1996-03-31
D107	Shivani	Andheri	1000	1995-09-05
D108	Kranti	Nehru Palace	5000	1995-07-02
D109	Naren	Powai	7000	1995-08-10
NULL	NULL	NULL	NULL	NULL

Program No: 03

Name: Sanio Luke Sebastian

Roll No: 35

Batch: B

Date: 19-04-2022

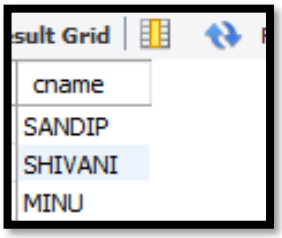
Aim: To familiarize with set operations.

Questions:

1. List all the customers who are depositors but not borrowers.
2. List all the customers who are both depositors and borrowers
3. List all the depositors having deposit in all the branches where Sunil is having Account.
4. List all the customers living in city NAGPUR and having branch city BOMBAY or DELHI
5. List all the depositors living in city NAGPUR
6. List all the depositors living in the city NAGPUR and having branch in city BOMBAY.
7. List the branch cities of Anil and Sunil.
8. List the customers having deposit greater than 1000 and loan less than 10000.
9. List the cities of depositors having branch VRCE.
10. List the depositors having amount less than 1000 and living in the same city as Anil.
11. List all the cities where branches of Anil and Sunil are located
12. List the amount for the depositors living in the city where Anil is living.

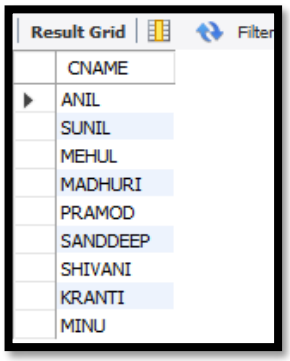
Procedure & Outputs:

1. select cname from deposite where cname not in (select cname from borrow);



cname
SANDIP
SHIVANI
MINU

2. select cname from deposite union (select cname from borrow);



CNAME
ANIL
SUNIL
MEHUL
MADHURI
PRAMOD
SANDDEEP
SHIVANI
KRANTI
MINU

3. select d1.cname from deposite d1 where d1.bname in (select d2.bname from deposite d2 where d2.cname = 'sunil');

Result Grid

CNAME
SUNIL

4. select c1.cname from customer c1,deposite d1, branch b1 where c1.city = 'nagpur' and c1.cname = d1.cname and d1.bname = b1.bname and b1.city in ('bombay','delhi');

Result Grid

CNAME
MADHURI

5. select distinct(customer.cname) from customer,deposite where city='nagpur';

Result Grid

CNAME
PRAMOD
MADHURI

6. select c1.cname from customer c1,deposite d1, branch b1 where c1.city = 'nagpur' and c1.cname = d1.cname and d1.bname = b1.bname and b1.city in ('bombay');

Result Grid

CNAME

7. select b1.city from deposit d1, branch b1 where d1.bname= b1.bname and d1.cname in ('sunil' ,'anil');

Result Grid

CITY
NAGPUR

8. select distinct d1.cname from deposite d1, borrow b1 where d1.amount>1000 and b1.amount<10000;

Result Grid

CNAME
MEHUL
MADHURI
PRAMOD
SANDIP
KRANTI
MINU

9. select b1.city from deposite d1, branch b1 where d1.bname=b1.bname and b1.bname='vrce';

Result Grid

CITY
NAGPUR

10. select d1.cname from deposite d1, customer c1 where amount<1000 and c1.city=(c1.cname='anil');

Result Grid

CNAME

11. select b1.city from branch b1 where b1.bname in (select d1.bname from deposit d1 where d1.cname in ('anil','sunil'));

Result Grid

CITY
NAGPUR

12. select distinct(d1.cname),d1.amount ,c1.city from deposite d1, customer c1, branch b1 where d1.cname=c1.cname and c1.city in(select c2.city from customer c2 where c2.cname='anil');

Result Grid Filter Rows:

CNAME	AMOUNT	CITY
ANIL	1000	CALCUTTA

Program No: 04

Name: Sanio Luke Sebastian

Roll No: 35

Batch: B

Date: 06-05-2022

Aim: To familiarize with join or cartesian product.

Questions:

1. Give name of customers having living city BOMBAY and branch city NAGPUR
2. Give names of customers having the same living city as their branch city
3. Give names of customers who are borrowers as well as depositors and having city NAGPUR.
4. Give names of borrowers having a deposit amount greater than 1000 and loan amount greater than 2000.
5. Give names of depositors having the same branch as the branch of Sunil
6. Give names of borrowers having loan amount greater than the loan amount of Pramod
7. Give the name of the customer living in the city where the branch of depositor Sunil is located.
8. Give branch city and living city of Pramod
9. Give branch city of Sunil and branch city of Anil
10. Give the living city of Anil and the living city of Sunil

Procedure & Outputs:

```
create database lab_cycle_six;
use lab_cycle_six;
```

1. select d1.cname from deposit d1,customer c1,branch b1 where c1.city = 'Bombay' and b1.city = 'Nagpur' and d1.cname = c1.cname and d1.bname = b1.bname;

Result Grid | Filter Ro

cname
Anil

2. select distinct(customer.cname), branch.city FROM branch, customer where branch.city = customer.city;

Result Grid | Filter Rows:

cname	city
Anil	Bombay
Kranti	Bombay
Madhuri	Nagpur
Naren	Bombay
Pramod	Nagpur
Shivani	Bombay
Sunil	Delhi

3. select c1.cname from customer c1,deposit d1,borrow b1 where c1.city='Nagpur' and c1.cname=d1.cname and d1.cname = b1.cname;

Result Grid | Filter Row

cname
Madhuri
Pramod

4. select br1.cname, br1.amount, d1.cname, d1.amount from borrow br1,deposit d1 where d1.cname = br1.cname and d1.amount > 1000 and br1.amount > 2000;

Result Grid | Filter Rows:

cname	amount	cname	amount
Pramod	3000.00	Pramod	3000
Mehul	5000.00	Mehul	3500
Sunil	3000.00	Sunil	5000
Kranti	5000.00	Kranti	5000

5. select d1.cname from deposit d1 where d1.bname in (select d2.bname from deposit d2 where d2.cname = 'Sunil');

Result Grid | Filter Rows:

cname
Sunil

6. select br1.cname,br1.amount from borrow br1 where br1.amount > all (select br2.amount from borrow br2 where br2.cname = 'Pramod');

Result Grid | Filter Row

cname	amount
Mehul	5000.00
Kranti	5000.00

7. select c.cname from customer c where c.city in (select b.city from branch b where b.bname in (select d.bname from deposit d where d.cname='Sunil'));

Result Grid | Filter Ro

cname
Madhuri
Pramod
NULL

8. select b1.city , c1.city from branch b1,customer c1, deposit d1 where c1.cname = 'Pramod' and c1.cname = d1.cname and d1.bname = b1.bname;

Result Grid | Filter Rows:

city	city
Bangalore	Nagpur

9. select d1.cname, b1.city from deposit d1, branch b1 where d1.bname = b1.bname and d1.cname in ('Sunil' ,'Anil');

Result Grid Filter Rows:

cname	city
Anil	Nagpur
Sunil	Nagpur

10. select c1.cname, c1.city from customer c1 where c1.cname = 'Anil' or c1.cname = 'Sunil';

Result Grid Filter Row

cname	city
Anil	Bombay
Sunil	Delhi
NULL	NULL

Program No: 05

Name: Sanio Luke Sebastian

Roll No: 35

Batch: B

Date: 06-05-2022

Aim: To familiarize with Group by and Having clause

Questions:

Using the lab cycle 06 database, perform the following queries.

1. List the branches having sum of deposit more than 5000.
2. List the branches having sum of deposit more than 500 and located in city BOMBAY
3. List the names of customers having deposited in the branches where the average deposit is more than 5000.
4. List the names of customers having maximum deposit
5. List the name of branch having highest number of depositors?
6. Count the number of depositors living in NAGPUR.
7. Give names of customers in VRCE branch having more deposite than any other customer in same branch
8. Give the names of branch where number of depositors is more than 5
9. Give the names of cities in which the maximum number of branches are located
10. Count the number of customers living in the city where branch is located

Procedure & Outputs:

use lab_cycle_six;

1. select d.bname from deposit d, branch b where d.bname=b.bname and b.city='Bombay' group by d.bname having sum(d.amount)>5000;

2. select d.bname from deposit d, branch b where d.bname=b.bname group by d.bname having sum(d.amount)>5000;

3. select cname from deposit where amount=(select avg(amount) from deposit group by bname having avg(amount)>5000);

Result Grid | Filter Ro

cname
Naren

4. select max(amount),cname from deposit;

Result Grid | Filter Rows:

max(amount)	cname
7000	Anil

5. select d1.bname from deposit d1 group by d1.bname having count(d1.cname) >= all (select count(d2.cname) from deposit d2 group by d2.bname);

Result Grid | Fi

bname
Andheri

6. select count(deposit.cname)from deposit,customer where customer.city='Nagpur';

Result Grid | Filter Rows:

count(deposit.cname)
18

7. select cname from deposit where bname='VRCE' and amount=(select max(amount) from deposit where bname='VRCE');

Result Grid | F

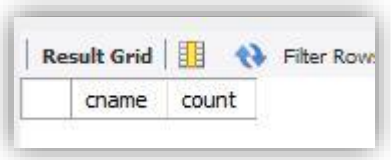
cname
Anil

8. select bname from deposit group by bname having count(bname) > 5;

Result Grid | Filter Row

cname	count

9. select c.cname ,count(b.bname) as count from customer c inner join branch b on c.cname=b.bname group by c.cname order by count(b.bname) desc;



10. select count(b1.bname) from deposit d1 , borrow b1 , customer c1 where c1.cname=d1.cname and d1.cname=b1.cname and c1.city in (select city from customer);

Program No: 06

Name: Sanio Luke Sebastian

Roll No: 35

Batch: B

Date: 10-06-2022

Aim: Implementation of triggers

Questions:

1. Create a Trigger for employee table it will update another table salary while updating values.
2. Create a Trigger for employe table it will update another table personal_updations while updating values.

Procedure & Outputs:

```
create database lab_trigger;
use lab_trigger;
```

1. **Question 01-**

```
CREATE TABLE student ( stud_id int(4) NOT NULL PRIMARY KEY auto_increment, name varchar(30), subject1 int(2), subject2 int(2), subject3 int(2), total int(3), per int(3));

CREATE TRIGGER mark
BEFORE INSERT ON student FOR EACH ROW
SET new.total=new.subject1+new.subject2+new.subject3, new.per= new.total / 300 * 100;

select * from student;
insert into student values (null,"Goblin",91, 89,82,0,0);
insert into student values (null,"Denvin",76, 91,67,0,0);
insert into student values (null,"Finny",78, 64,79,0,0);
```

Result Grid | Filter Rows: | Edit:

stud_id	name	subject1	subject2	subject3	total	per
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit:

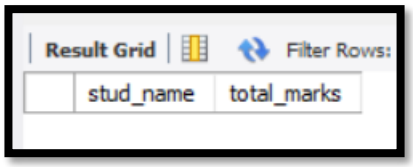
stud_id	name	subject1	subject2	subject3	total	per
1	Goblin	91	89	82	262	87
2	Goblin	91	89	82	262	87
3	Denvin	76	91	67	234	78
4	Finny	78	64	79	221	74
NULL	NULL	NULL	NULL	NULL	NULL	NULL

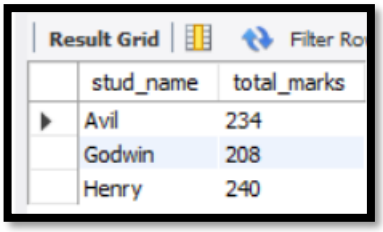
2. **Question 02-**

CREATE TABLE marks (stud_name VARCHAR(20), total_marks int(3));

CREATE TRIGGER mark_trigger
AFTER INSERT ON student FOR EACH ROW
INSERT INTO marks(stud_name,total_marks) VALUES(new.name,new.total);

INSERT INTO student (stud_id, name, subject1, subject2, subject3) values(null,'Avil',78, 65, 91),(null,'Godwin',67, 86, 55),(null,'Henry',81, 96, 63);
SELECT *FROM marks;





Program No: 07

Name: Sanio Luke Sebastian

Roll No: 35

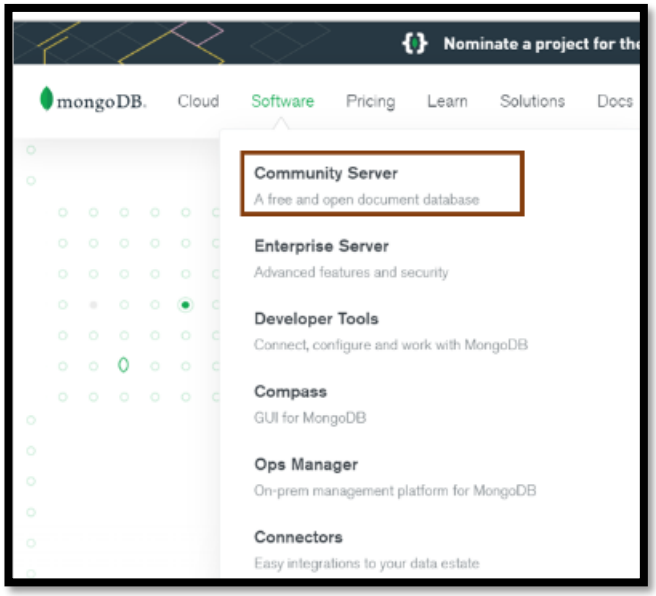
Batch: B

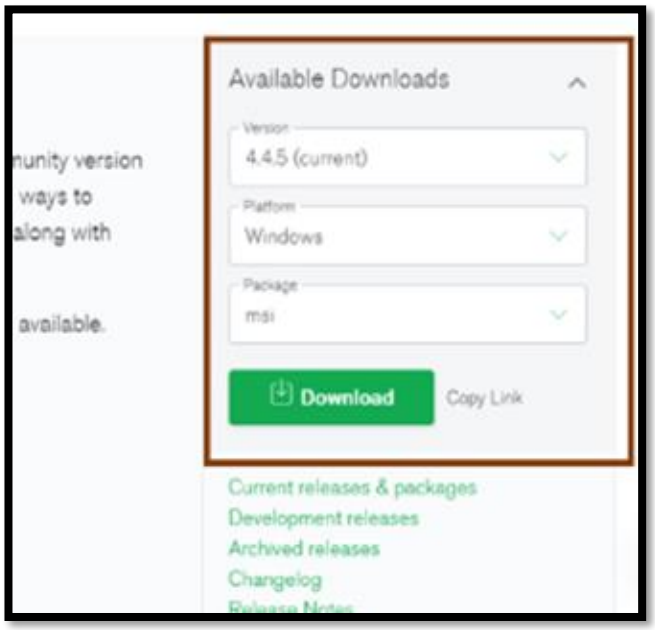
Date: 24-05-2022

Aim: Installation of MongoDB on Windows

Procedure & Outputs:

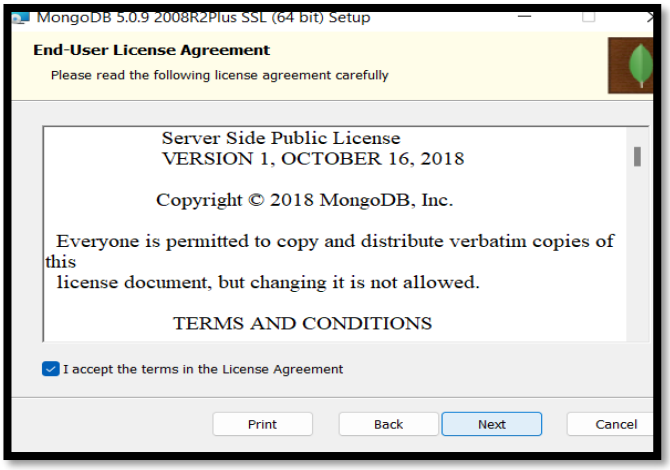
1. download the community server version of mongo db



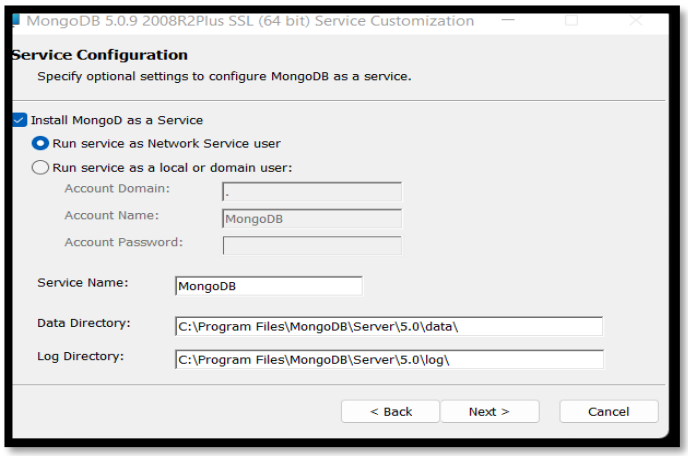


2. Now, Install the software on your pc

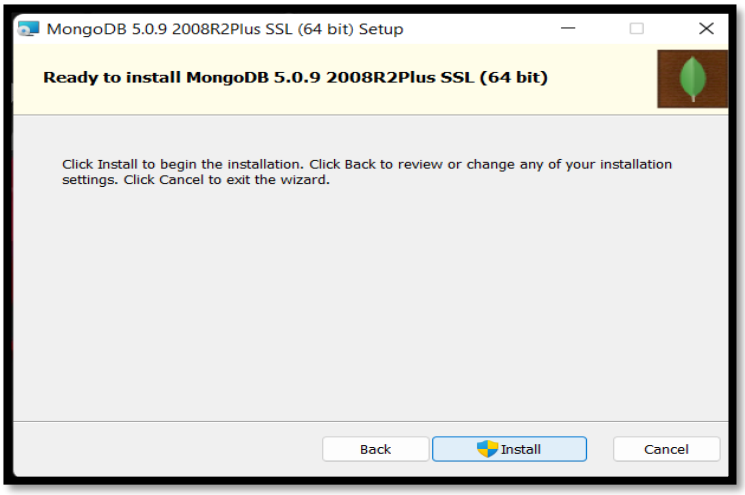
3. Accept the term and conditions.



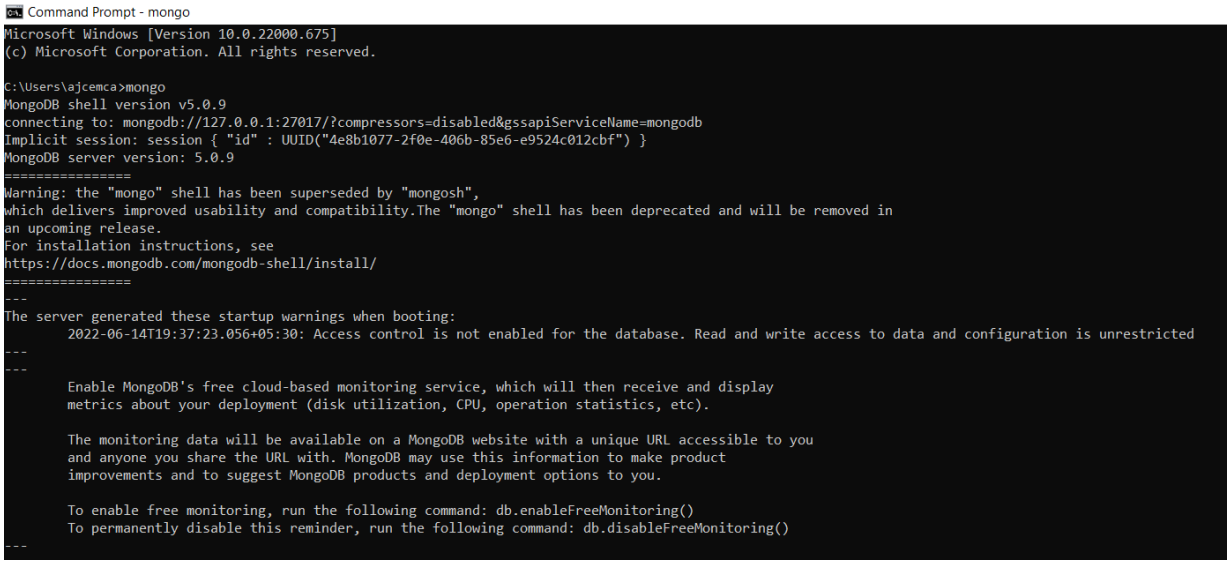
4. choose service configruation



5. Make necessary changes and install



Step 6: verify the installation by typing mongo on cmd



Program No: 08

Name: Sanio Luke Sebastian

Roll No: 35

Batch: B

Date: 24-05-2022

Aim: Designing Databases using NoSQL: MongoDB

Procedure & Outputs:

- To show Database

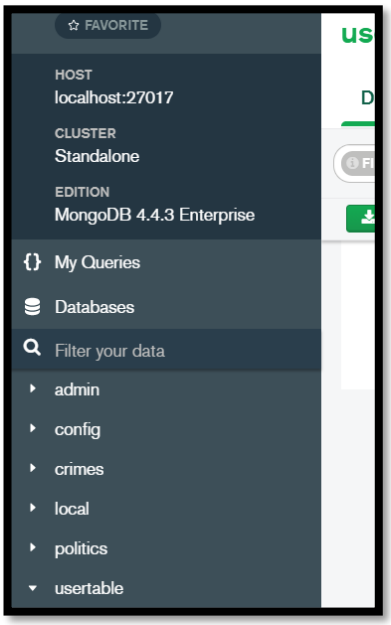
```
> show dbs
AJCE            0.000GB
admin           0.000GB
config          0.000GB
dbms            0.000GB
local           0.000GB
mongo-sanio-db  0.000GB
```

- To create new Database

```
> use mongo-sanio-db
switched to db mongo-sanio-db
```

- To create collection and show it

```
> db.createCollection("movie")
{ "ok" : 1 }
```



Program No: 09

Name: Sanio Luke Sebastian

Roll No: 35

Batch: B

Date: 03-06-2022

Aim: Query Processing : Performing CRUD operations with NoSQL database.

Procedure & Outputs:

1. Display available Databases and selecting the desired database.

```
> show dbs
AJCE            0.000GB
admin           0.000GB
config          0.000GB
dbms            0.000GB
local           0.000GB
mongo-sanio-db  0.000GB
> use mongo-sanio-db
switched to db mongo-sanio-db
> show dbs
AJCE            0.000GB
admin           0.000GB
config          0.000GB
dbms            0.000GB
local           0.000GB
mongo-sanio-db  0.000GB
```

2. Creating collection in the database.

```
> db.createCollection("movie")
{ "ok" : 1 }
```

```
> db.createCollection("developers")
{ "ok" : 1 }
```

3. Insertion of fields
   a. Insert: Inserts entered queried documents into the collection.

```
> db.movie.insert({"name":"Avengers: Endgame"})
WriteResult({ "nInserted" : 1 })
> db.movie.insert([{"name":"Avengers: 2012"},{"name":"Captain Amer
ble Hulk"},{"name":"Captain Marvel"}])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 7,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

b. InsertMany: Inserts multiple documents into the collection at same time.

```
> db.developers.insertMany([{_id:1001,name:"Danatorh",des:"Mobile A
: "JUnit, Selenium",yoj:2010}])
{ "acknowledged" : true, "insertedIds" : [ 1001, 1002 ] }
```

c. InsertOne: Inserts only one documents into the collection.

```
> db.developers.insertOne({_id:1003,name:"Gabriel Kinemon",des:"Cyb
{ "acknowledged" : true, "insertedId" : 1003 }
>
```

4. Find fields and documents using filters

```
> db.developers.find()
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2011 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2010 }
{ "_id" : 1003, "name" : "Gabriel Kinemon", "des" : "Cyber Security", "tool" : "Kali Linux", "yoj" : "2005" }
>
```

```
> db.developers.find({ yoj: {$gte: 2010}})
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2011 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2010 }
>
```

5. Update fields according to the filter query conditions

a. UpdateOne: Updates only one document.

```
> db.developers.find()
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2011 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2010 }
{ "_id" : 1003, "name" : "Gabriel Kinemon", "des" : "Cyber Security", "tool" : "Kali Linux", "yoj" : "2005" }
{ "_id" : 1004, "name" : "Avil", "des" : "Software Developer", "tool" : "Eclipse", "yoj" : 2020 }
{ "_id" : 1005, "name" : "Nebin Thomas", "des" : "Web Developer", "tool" : "PHP, HTML, CSS", "yoj" : 2019 }
{ "_id" : 1006, "name" : "Tejas", "des" : "Back-end Developer", "tool" : "SQL, PSQl, MongoDB", "yoj" : 2018 }
{ "_id" : 1007, "name" : "Valle", "des" : "DBMS", "tool" : "Django", "yoj" : 2021 }
{ "_id" : 1008, "name" : "Sanio", "des" : "Mobile App Developer", "tool" : "Android, Java, Kotlin, Flutter", "yoj" : 2022 }
{ "_id" : 1009, "name" : "Vikram", "des" : "Hardware Server manager", "tool" : "kali Linux", "yoj" : 2011 }
>
>
> db.developers.updateOne({tool: {$eq : "Kali Linux"}},{$set: {yoj: 2010}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
>
> db.developers.find()
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2011 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2010 }
{ "_id" : 1003, "name" : "Gabriel Kinemon", "des" : "Cyber Security", "tool" : "Kali Linux", "yoj" : 2010 }
{ "_id" : 1004, "name" : "Avil", "des" : "Software Developer", "tool" : "Eclipse", "yoj" : 2020 }
{ "_id" : 1005, "name" : "Nebin Thomas", "des" : "Web Developer", "tool" : "PHP, HTML, CSS", "yoj" : 2019 }
{ "_id" : 1006, "name" : "Tejas", "des" : "Back-end Developer", "tool" : "SQL, PSQl, MongoDB", "yoj" : 2018 }
{ "_id" : 1007, "name" : "Valle", "des" : "DBMS", "tool" : "Django", "yoj" : 2021 }
{ "_id" : 1008, "name" : "Sanio", "des" : "Mobile App Developer", "tool" : "Android, Java, Kotlin, Flutter", "yoj" : 2022 }
{ "_id" : 1009, "name" : "Vikram", "des" : "Hardware Server manager", "tool" : "kali Linux", "yoj" : 2011 }
```

b. UpdateMany: Updates multiple documents according to the query from the collection.

```
> db.developers.find()
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2022 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2022 }
{ "_id" : 1003, "name" : "Gabriel Kinemon", "des" : "Cyber Security", "tool" : "Kali Linux", "yoj" : 2022 }
{ "_id" : 1004, "name" : "Avil", "des" : "Software Developer", "tool" : "Eclipse", "yoj" : 2020 }
{ "_id" : 1005, "name" : "Nebin Thomas", "des" : "Web Developer", "tool" : "PHP, HTML, CSS", "yoj" : 2019 }
{ "_id" : 1006, "name" : "Tejas", "des" : "Back-end Developer", "tool" : "SQL, PSQl, MongoDB", "yoj" : 2018 }
{ "_id" : 1007, "name" : "Valle", "des" : "DBMS", "tool" : "Django", "yoj" : 2021 }
{ "_id" : 1008, "name" : "Sanio", "des" : "Mobile App Developer", "tool" : "Android, Java, Kotlin, Flutter", "yoj" : 2022 }
{ "_id" : 1009, "name" : "Vikram", "des" : "Hardware Server manager", "tool" : "kali Linux", "yoj" : 2022 }
>
>
> db.developers.replaceOne({name : {$eq: "Vikram"}},{name: "Vikram",des:"Client-Side Architect",tool:"Laravel",yoj:2015})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
>
> db.developers.find()
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2022 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2022 }
{ "_id" : 1003, "name" : "Gabriel Kinemon", "des" : "Cyber Security", "tool" : "Kali Linux", "yoj" : 2022 }
{ "_id" : 1004, "name" : "Avil", "des" : "Software Developer", "tool" : "Eclipse", "yoj" : 2020 }
{ "_id" : 1005, "name" : "Nebin Thomas", "des" : "Web Developer", "tool" : "PHP, HTML, CSS", "yoj" : 2019 }
{ "_id" : 1006, "name" : "Tejas", "des" : "Back-end Developer", "tool" : "SQL, PSQl, MongoDB", "yoj" : 2018 }
{ "_id" : 1007, "name" : "Valle", "des" : "DBMS", "tool" : "Django", "yoj" : 2021 }
{ "_id" : 1008, "name" : "Sanio", "des" : "Mobile App Developer", "tool" : "Android, Java, Kotlin, Flutter", "yoj" : 2022 }
{ "_id" : 1009, "name" : "Vikram", "des" : "Client-Side Architect", "tool" : "Laravel", "yoj" : 2015 }
```

c. replaceOne: Replace the whole specified document with the new values.

```
> db.developers.find()
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2022 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2022 }
{ "_id" : 1003, "name" : "Gabriel Kinemon", "des" : "Cyber Security", "tool" : "Kali Linux", "yoj" : 2022 }
{ "_id" : 1004, "name" : "Avil", "des" : "Software Developer", "tool" : "Eclipse", "yoj" : 2020 }
{ "_id" : 1005, "name" : "Nebin Thomas", "des" : "Web Developer", "tool" : "PHP, HTML, CSS", "yoj" : 2019 }
{ "_id" : 1006, "name" : "Tejas", "des" : "Back-end Developer", "tool" : "SQL, PSQl, MongoDB", "yoj" : 2018 }
{ "_id" : 1007, "name" : "Valle", "des" : "DBMS", "tool" : "Django", "yoj" : 2021 }
{ "_id" : 1008, "name" : "Sanio", "des" : "Mobile App Developer", "tool" : "Android, Java, Kotlin, Flutter", "yoj" : 2022 }
{ "_id" : 1009, "name" : "Vikram", "des" : "Hardware Server manager", "tool" : "kali Linux", "yoj" : 2022 }
>
>
> db.developers.replaceOne({name : {$eq: "Vikram"}},{name: "Vikram",des:"Client-Side Architect",tool:"Laravel",yoj:2015})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
>
> db.developers.find()
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2022 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2022 }
{ "_id" : 1003, "name" : "Gabriel Kinemon", "des" : "Cyber Security", "tool" : "Kali Linux", "yoj" : 2022 }
{ "_id" : 1004, "name" : "Avil", "des" : "Software Developer", "tool" : "Eclipse", "yoj" : 2020 }
{ "_id" : 1005, "name" : "Nebin Thomas", "des" : "Web Developer", "tool" : "PHP, HTML, CSS", "yoj" : 2019 }
{ "_id" : 1006, "name" : "Tejas", "des" : "Back-end Developer", "tool" : "SQL, PSQl, MongoDB", "yoj" : 2018 }
{ "_id" : 1007, "name" : "Valle", "des" : "DBMS", "tool" : "Django", "yoj" : 2021 }
{ "_id" : 1008, "name" : "Sanio", "des" : "Mobile App Developer", "tool" : "Android, Java, Kotlin, Flutter", "yoj" : 2022 }
{ "_id" : 1009, "name" : "Vikram", "des" : "Client-Side Architect", "tool" : "Laravel", "yoj" : 2015 }
```

6. Delete documents according to specified conditions.

a. DeleteOne: Deletes only one documents according to the condition.

```
> db.developers.find()
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2022 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2022 }
{ "_id" : 1003, "name" : "Gabriel Kinemon", "des" : "Cyber Security", "tool" : "Kali Linux", "yoj" : 2022 }
{ "_id" : 1004, "name" : "Avil", "des" : "Software Developer", "tool" : "Eclipse", "yoj" : 2020 }
{ "_id" : 1005, "name" : "Nebin Thomas", "des" : "Web Developer", "tool" : "PHP, HTML, CSS", "yoj" : 2019 }
{ "_id" : 1006, "name" : "Tejas", "des" : "Back-end Developer", "tool" : "SQL, PSQl, MongoDB", "yoj" : 2018 }
{ "_id" : 1007, "name" : "Valle", "des" : "DBMS", "tool" : "Django", "yoj" : 2021 }
{ "_id" : 1008, "name" : "Sanio", "des" : "Mobile App Developer", "tool" : "Android, Java, Kotlin, Flutter", "yoj" : 2022 }
{ "_id" : 1009, "name" : "Vikram", "des" : "Client-Side Architect", "tool" : "Laravel", "yoj" : 2015 }
>
>
>
> db.developers.deleteOne({name: "Sanio"})
{ "acknowledged" : true, "deletedCount" : 1 }
>
> db.developers.find()
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2022 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2022 }
{ "_id" : 1003, "name" : "Gabriel Kinemon", "des" : "Cyber Security", "tool" : "Kali Linux", "yoj" : 2022 }
{ "_id" : 1004, "name" : "Avil", "des" : "Software Developer", "tool" : "Eclipse", "yoj" : 2020 }
{ "_id" : 1005, "name" : "Nebin Thomas", "des" : "Web Developer", "tool" : "PHP, HTML, CSS", "yoj" : 2019 }
{ "_id" : 1006, "name" : "Tejas", "des" : "Back-end Developer", "tool" : "SQL, PSQl, MongoDB", "yoj" : 2018 }
{ "_id" : 1007, "name" : "Valle", "des" : "DBMS", "tool" : "Django", "yoj" : 2021 }
{ "_id" : 1009, "name" : "Vikram", "des" : "Client-Side Architect", "tool" : "Laravel", "yoj" : 2015 }
```

b. DeleteMany: Deletes all the documents that satisfies the condition.

```
> db.developers.find()
{ "_id" : 1001, "name" : "Danatorh", "des" : "Mobile App Developer", "tool" : "React Native", "yoj" : 2022 }
{ "_id" : 1002, "name" : "Frankinstien", "des" : "Software Tester", "tool" : "JUnit, Selenium", "yoj" : 2022 }
{ "_id" : 1003, "name" : "Gabriel Kinemon", "des" : "Cyber Security", "tool" : "Kali Linux", "yoj" : 2022 }
{ "_id" : 1004, "name" : "Avil", "des" : "Software Developer", "tool" : "Eclipse", "yoj" : 2020 }
{ "_id" : 1005, "name" : "Nebin Thomas", "des" : "Web Developer", "tool" : "PHP, HTML, CSS", "yoj" : 2019 }
{ "_id" : 1006, "name" : "Tejas", "des" : "Back-end Developer", "tool" : "SQL, PSQl, MongoDB", "yoj" : 2018 }
{ "_id" : 1007, "name" : "Valle", "des" : "DBMS", "tool" : "Django", "yoj" : 2021 }
{ "_id" : 1009, "name" : "Vikram", "des" : "Client-Side Architect", "tool" : "Laravel", "yoj" : 2015 }
>
>
> db.developers.deleteMany({yoj: 2022})
{ "acknowledged" : true, "deletedCount" : 3 }
>
>
> db.developers.find()
{ "_id" : 1004, "name" : "Avil", "des" : "Software Developer", "tool" : "Eclipse", "yoj" : 2020 }
{ "_id" : 1005, "name" : "Nebin Thomas", "des" : "Web Developer", "tool" : "PHP, HTML, CSS", "yoj" : 2019 }
{ "_id" : 1006, "name" : "Tejas", "des" : "Back-end Developer", "tool" : "SQL, PSQl, MongoDB", "yoj" : 2018 }
{ "_id" : 1007, "name" : "Valle", "des" : "DBMS", "tool" : "Django", "yoj" : 2021 }
{ "_id" : 1009, "name" : "Vikram", "des" : "Client-Side Architect", "tool" : "Laravel", "yoj" : 2015 }
```

Program No: 10

Name: Sanio Luke Sebastian

Roll No: 35

Batch: B

Date: 03-06-2022

Aim: NoSQL and Front-End: PHP: Create a PHP form and insert data to mongodb

Procedure & Outputs:

index.php

```
<?php
   $mongo = new MongoDB\Driver\Manager("mongodb://localhost:27017");
   if(isset($_POST["form-submit"])){
      $fullname=$_POST["fullname"];
      $username=$_POST["username"];
      $emailid=$_POST["emailid"];
      $passwrd=$_POST["passwrd"];
      $writer=new MongoDB\Driver\Bulkwrite;
      $writer-
>insert(["fullname"=>$fullname,"username"=>$username,"emailid"=>$emailid,"passwrd"=>$passwrd]
);
      $mongo->executeBulkWrite('usertable.userinfo',$writer);
      header("Location:success.html");
      die();
   }
?>
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>PHP - MongoDB Connection</title>
   <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;600&display=swap"
rel="stylesheet">
   <link rel="stylesheet" href="css.css">
</head>
<body>
   <center>
      <h3 style="margin-top: 40px">The PHP - MongoDB Database Connection</h3>
      <p>The php-mongodb connection is explained via form submission. Please fill the form and
submit to upload the data to MongoDB.</p>
```

```
        <form action="index.php" method="POST" id="index_form">
            <h2>Profile Information</h2>
            <br>

            <table id="form-table">
                <tr>
                    <td><label for="fullname" class="formlabel">Full Name : </label></td>
                    <td><input type="text" class="forminput" name="fullname" id="fullname"
placeholder="Enter your first name..."></td>
                </tr>
                <tr>
                    <td><label for="username" class="formlabel">Username : </label></td>
                    <td><input type="text" class="forminput" name="username" id="username"
placeholder="Enter a new username..."></td>
                </tr>
                <tr>
                    <td><label for="emailid" class="formlabel">Email ID : </label></td>
                    <td><input type="email" class="forminput" name="emailid" id="emailid"
placeholder="Enter your Email-ID..."></td>
                </tr>
                <tr>
                    <td><label for="passwrd" class="formlabel">Password : </label></td>
                    <td><input type="password" class="forminput" name="passwrd" id="passwrd"
placeholder="Enter a new password..."></td>
                </tr>
                <tr>
                    <td colspan="2"><input type="submit" name="form-submit" id="form-submit"></td>
                </tr>
            </table>

        </form>
    </center>
</body>
</html>
```

success.html

```
<html>
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Success Page - Connection</title>
```

```
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;600&display=swap"
rel="stylesheet">
    <link rel="stylesheet" href="css.css">
</head>
<body>
    <center>
        <br><br>
        <h2>successfully created</h2>
        <p>The Data has been successfully uploaded to the MongoDB Database. Thanks for providing
your information and support & time.</p>
        <p id="success-emoji">&#128513;</p>
    </center>
</body>
</html>
```