1. **Create a table called Employee & execute the following.**
   Employee(EMPNO,ENAME,JOB, MANAGER_NO, SAL, COMMISSION)
   1. Create a user and grant all permissions to the user.
   2. Insert the any three records in the employee table contains attributes EMPNO,ENAME JOB, MANAGER_NO, SAL, COMMISSION and use rollback. Check the result.
   3. Add primary key constraint and not null constraint to the employee table.
   4. Insert null values to the employee table and verify the result.

>create user 'Name' identified by 'Password';
>grant all privileges on *.* to 'Name;
>create table Employee(EmpNo int,EName char(20),Job char(20),Manager_No int, Salary float,Commission int);
> desc Employee;

Field	Type	Null	Key	Default	Extra
EmpNo	int	YES		NULL	
EName	char(20)	YES		NULL	
Job	char(20)	YES		NULL	
Manager_No	int	YES		NULL	
Salary	float	YES		NULL	
Commission	int	YES		NULL	

> insert into Employee values(101,'John','Engineer',562547,35000,100);
> insert into Employee values(102,'Smith','CA',631548,50000,150);
> insert into Employee values(103,'Ravi','CEO',501524,55000,50);
> select * from Employee;

EmpNo	EName	Job	Manager_No	Salary	Commission
101	John	Engineer	562547	35000	100
102	Smith	CA	631548	50000	150
103	Ravi	CEO	501524	55000	50

>set autocommit=0;
> update Employee set Job='Manager' where EmpNo=102;
> select * from Employee;

>rollback;
> select * from Employee;

EmpNo	EName	Job	Manager_No	Salary	Commission
101	John	Engineer	562547	35000	100
102	Smith	CEO	631548	50000	150
103	Ravi	CEO	501524	55000	50

> alter table Employee modify column EmpNo int PRIMARY KEY;
> alter table Employee modify column EName char(20) not null;
> desc Employee;

Field	Type	Null	Key	Default	Extra
EmpNo	int	NO	PRI	NULL	
EName	char(20)	NO		NULL	
Job	char(20)	YES		NULL	
Manager_No	int	YES		NULL	
Salary	float	YES		NULL	
Commission	int	YES		NULL	

>

insert into Employee values(103,NULL,NULL,NULL,NULL,NULL);
ERROR 1048 (23000): Column 'EName' cannot be null

2. **Create a table called Employee that contain attributes EMPNO,ENAME,JOB, MGR,SAL & execute the following.**
   1. Add a column commission with domain to the Employeetable.
   2. Insert any five records into the table.
   3. Update the column details of job
   4. Rename the column of Employ table using alter command.
   5. Delete the employee whose Empno is 105.

> create table Employee(EmpNo int PRIMARY KEY,EName char(20),Job char(20),Manager_No int,Salary float);
> alter table Employee add column Commission decimal(10,2);
> desc Employee;

Field	Type	Null	Key	Default	Extra
EmpNo	int	NO	PRI	NULL	
EName	char(20)	YES		NULL	
Job	char(20)	YES		NULL	
Manager_No	int	YES		NULL	
Salary	float	YES		NULL	
Commission	decimal(10,2)	YES		NULL	

> insert into Employee values(101,'John','Emgineer',521458,35600,500);
> insert into Employee values(102,'Alice','Manager',582301,50000,400);
> insert into Employee values(103,'Michael','CEO',632581,65000,450);
> insert into Employee values(104,'Emily','Professor',812521,75000,250);
> insert into Employee values(105,'David','Assisant',745213,5000,200);
> select * from Employee;

EmpNo	EName	Job	Manager_No	Salary	Commission
101	John	Emgineer	521458	35600	500.00
102	Alice	Manager	582301	50000	400.00
103	Michael	CEO	632581	65000	450.00
104	Emily	Professor	812521	75000	250.00
105	David	Assisant	745213	5000	200.00

> update Employee set Job='Owner' where EmpNo=105;

EmpNo	EName	Job	Manager_No	Salary	Commission
101	John	Emgineer	521458	35600	500.00
102	Alice	Manager	582301	50000	400.00
103	Michael	CEO	632581	65000	450.00
104	Emily	Professor	812521	75000	250.00
105	David	Owner	745213	5000	200.00

> alter table Employee rename column Commission to Bonus;

Field	Type	Null	Key	Default	Extra
EmpNo	int	NO	PRI	NULL	
EName	char(20)	YES		NULL	
Job	char(20)	YES		NULL	
Manager_No	int	YES		NULL	
Salary	float	YES		NULL	
Bonus	decimal(10,2)	YES		NULL	

> delete from Employee where EmpNo=105;
> select * from Employee;

EmpNo	EName	Job	Manager_No	Salary	Bonus
101	John	Emgineer	521458	35600	500.00
102	Alice	Manager	582301	50000	400.00
103	Michael	CEO	632581	65000	450.00
104	Emily	Professor	812521	75000	250.00

3. **Queries using aggregate functions(COUNT,AVG,MIN,MAX,SUM),Group by,Orderby. Employee(E_id, E_name, Age, Salary)**

1. Create Employee table containing all Records E_id, E_name, Age, Salary.

2. Count number of employee names from employeetable

3. Find the Maximum age from employee table.

4. Find the Minimum age from employeetable.

5. Find salaries of employee in Ascending Order.

6. Find grouped salaries of employees.

>create table Employee (E_id int PRIMARY KEY, E_Name char(20), Age int, Salary decimal(10,2));

> desc Employee;

```
+--------+---------------+------+-----+---------+-------+
| Field  | Type          | Null | Key | Default | Extra |
+--------+---------------+------+-----+---------+-------+
| E_id   | int           | NO   | PRI | NULL    |       |
| E_Name | char(20)      | YES  |     | NULL    |       |
| Age    | int           | YES  |     | NULL    |       |
| Salary | decimal(10,2) | YES  |     | NULL    |       |
+--------+---------------+------+-----+---------+-------+
```

>insert into Employee values(101,"Smith",28,55000);

>insert into Employee values(102,"John",24,60000);

> insert into Employee values(103,"Alice",32,75000);

> insert into Employee values(104,"Jack",32,75000);

> insert into Employee values(105,"David",25,60000);

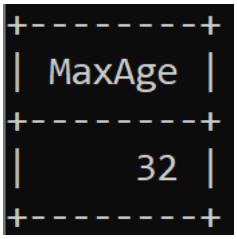
> select * from Employee;

```
+------+--------+------+----------+
| E_id | E_Name | Age  | Salary   |
+------+--------+------+----------+
|  101 | Smith  |   28 | 55000.00 |
|  102 | John   |   24 | 60000.00 |
|  103 | Alice  |   32 | 75000.00 |
|  104 | Jack   |   32 | 75000.00 |
|  105 | David  |   25 | 60000.00 |
+------+--------+------+----------+
```

>select count(E_Name) as TotalEmployee from Employee;

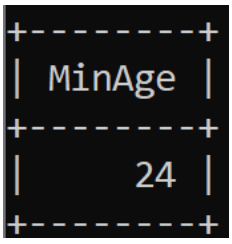
```
+---------------+
| TotalEmployee |
+---------------+
|             5 |
+---------------+
```

>select MAX(Age) as MaxAge from Employee;



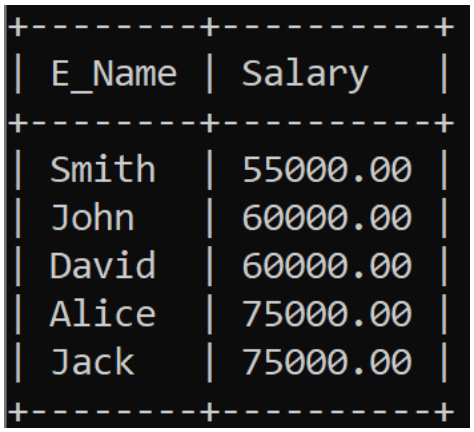
MaxAge
32

>select MIN(Age) as MinAge from Employee;



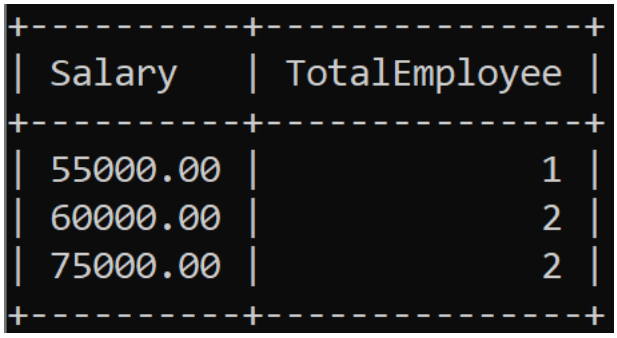
MinAge
24

>select E_Name, Salary from Employee order by Salary;



E_Name	Salary
Smith	55000.00
John	60000.00
David	60000.00
Alice	75000.00
Jack	75000.00

>select Salary,Count(*) as TotalEmployee from Employee group by Salary;



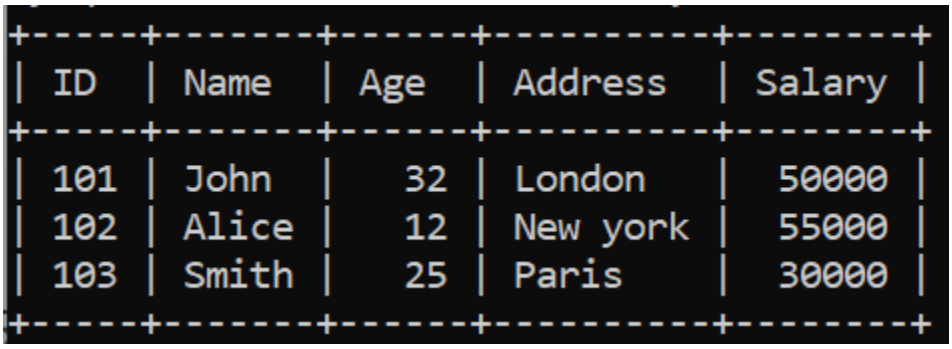
Salary	TotalEmployee
55000.00	1
60000.00	2
75000.00	2

4. **Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old & new Salary. CUSTOMERS(ID,NAME,AGE,ADDRESS,SALARY)**

>Create table Customers(ID int primary key, Name varchar(20), Age int, Address varchar(50), Salary float);

> insert into Customers values(101,'John',32,'London',50000),(102,'Alice',12,'New york',55000),(103,'Smith',25,'Paris',30000);

>select * from Customers;



ID	Name	Age	Address	Salary
101	John	32	London	50000
102	Alice	12	New york	55000
103	Smith	25	Paris	30000

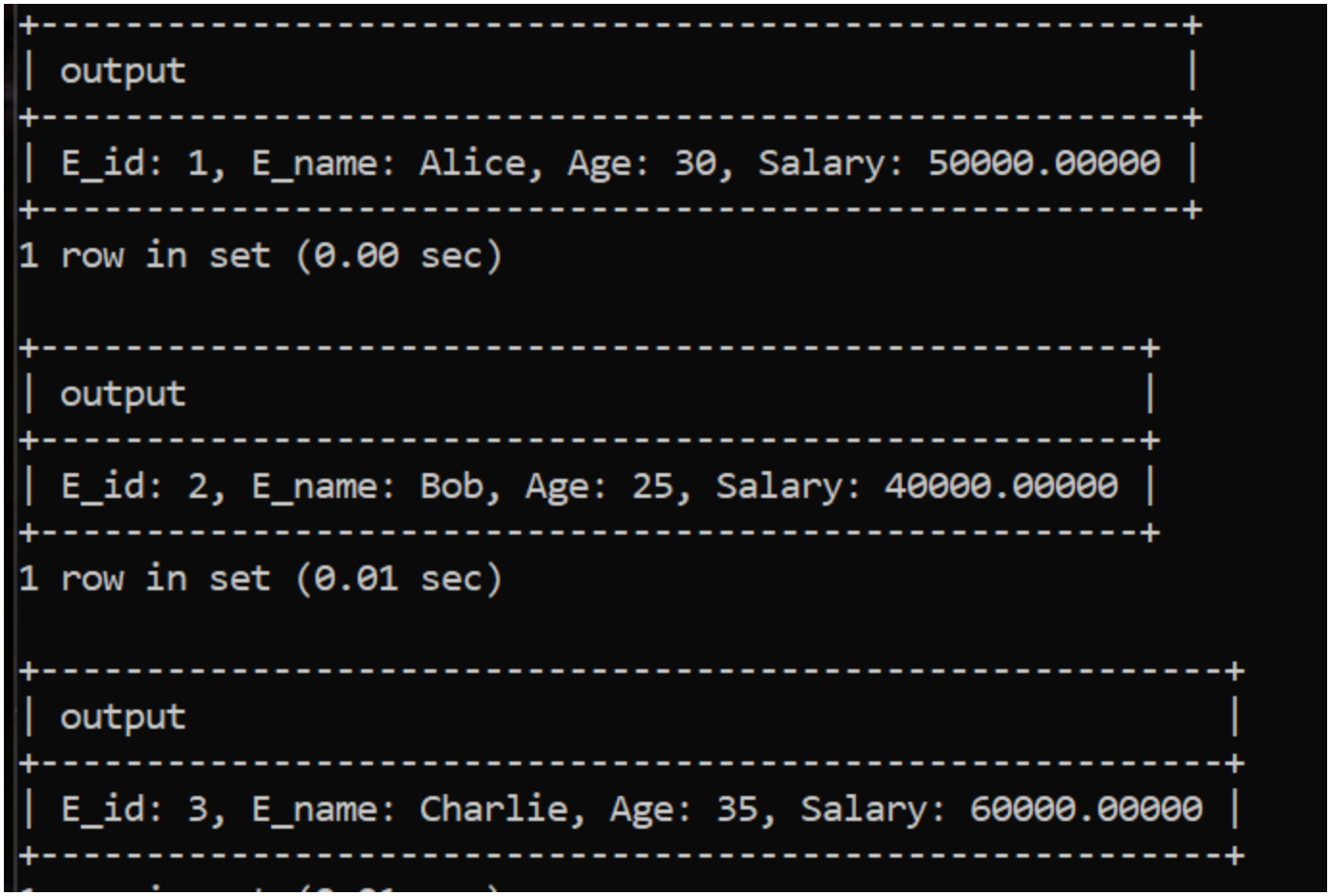
```
>Delimiter $$
  create trigger Update_customer
  before update on Customers
  for each row
  begin
        declare Salary_diff decimal(10,2);
        set Salary_diff = New.Salary-Old.Salary;
        set @meess=Concat('Salary Difference: ', Salary_diff);
        signal sqlstate '45000'
        set message_text=@meess;
  end$$
>Delimiter ;
>update Customers set Salary=60000 where ID=102;
ERROR 1644 (45000): Salary Difference: 5000.00
```

5. **Create cursor for Employee table & extract the values from the table. Declare the variables ,Open the cursor & extrct the values from the cursor. Close the cursor. Employee(E_id, E_name, Age, Salary)**

> create table Employee(E_Id int primary key, E_name varchar(20), Age int, Salary float);
>insert into Employee values(1, 'Alice', 30, 50000.00), (2, 'Bob', 25, 40000.00),
(3, 'Charlie', 35, 60000.00);
>select * from Employee;

E_Id	E_name	Age	Salary
1	Alice	30	50000
2	Bob	25	40000
3	Charlie	35	60000

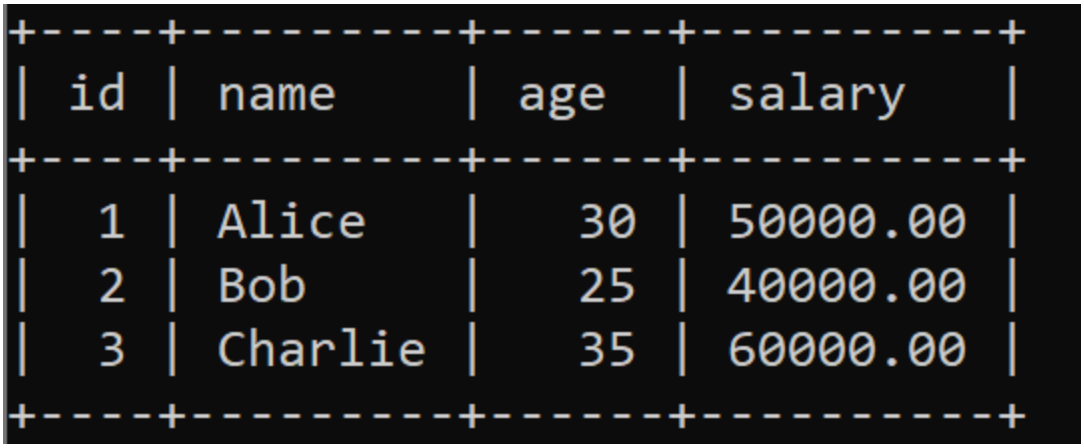
```
>Delimiter //
create procedure FetchEmployeeData()
Begin
        Declare v_E_id int;
        Declare v_E_name varchar(50);
        Declare v_Age int;
        Declare v_Salary decimal(10,5);
        declare done int Default FALSE;
        Declare emp_cursor cursor for
                Select E_id,E_name,Age,Salary from Employee;
        Declare continue handler for not found set done =True;
        open emp_cursor;
        Fetch_loop: loop
                Fetch emp_cursor into v_E_id, v_E_name, v_Age, v_Salary;
                if done then
                        leave Fetch_loop;
                End if;
                select concat('E_id: ', v_E_id, ', E_name: ', v_E_name, ', Age: ', v_Age, ', Salary: ',
v_Salary) AS output;
        End loop;
        close emp_cursor;
End //
Delimiter ;
>call FetchEmployeeData();
```


output
E_id: 1, E_name: Alice, Age: 30, Salary: 50000.00000
1 row in set (0.00 sec)
output
E_id: 2, E_name: Bob, Age: 25, Salary: 40000.00000
1 row in set (0.01 sec)
output
E_id: 3, E_name: Charlie, Age: 35, Salary: 60000.00000

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

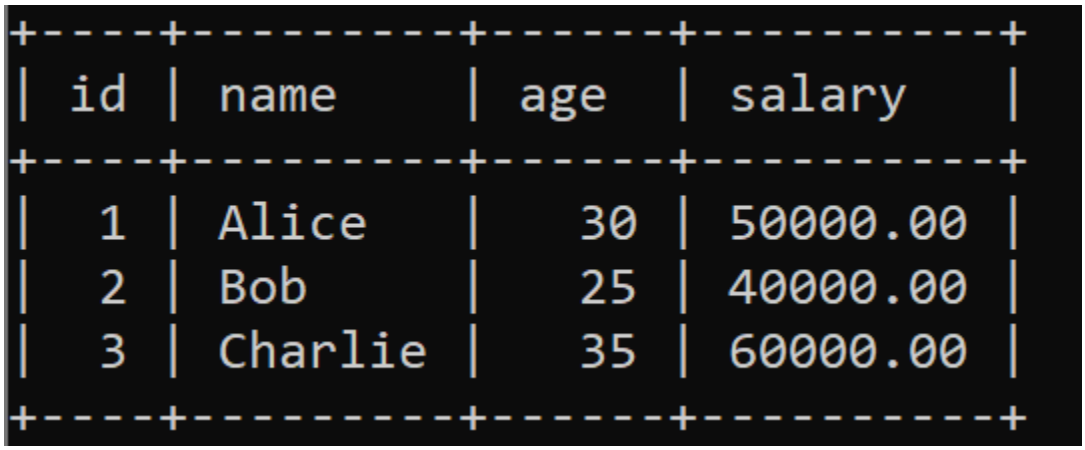
```
>create table n_rollcall(id int primary key, name varchar(20), age int, salary decimal(10,2));
>create table o_rollcall(id int primary key, name varchar(20), age int, salary decimal(10,2));

>Delimiter //
Create procedure Merge_N_RollCall_to_O_RollCall()
        Begin
        Declare done int default false;
        Declare v_id int;
        Declare v_name varchar(20);
        Declare v_Age int;
        Declare v_Salary decimal(10,2);
        Declare n_cursor cursor for
        select id,Name,Age,Salary from N_RollCall;
        Declare continue handler for not found set done=True;
        open n_cursor;
        Fetch_loop: Loop
        Fetch n_cursor into v_id, v_name, v_Age, v_Salary;
        if done then
        leave Fetch_loop;
        end if;
        select count(*) into @count from O_RollCall where id=v_id;
        if @count=0 then
        insert into O_RollCall values (v_id,v_name,v_Age,v_Salary);
        end if;
        end loop;
        close n_cursor;
        End//
        Delimiter ;

>select * from n_rollcall;
Empty set (0.00 sec)
>select * from o_rollcall;
Empty set (0.00 sec)
> insert into n_rollcall values (1,'Alice',30,50000), (2,'Bob',25,40000), (3,'Charlie',35,60000);
```

>select * from n_rollcall;



id	name	age	salary
1	Alice	30	50000.00
2	Bob	25	40000.00
3	Charlie	35	60000.00

> call Merge_N_RollCall_to_O_RollCall();
>select * from o_rollcall;



id	name	age	salary
1	Alice	30	50000.00
2	Bob	25	40000.00
3	Charlie	35	60000.00

7. **Install an Open Source NoSQL Data base MangoDB & perform basic CRUD(Create, Read, Update & Delete) operations. Execute MangoDB basic Queries using CRUD operations.**

To start using MongoDB, follow these steps:

1. Install MongoDB: Download and install MongoDB from the official MongoDB website.

2. Start MongoDB: Start the MongoDB server using the mongod command.

3. Use MongoDB Shell or Compass: Use the MongoDB shell (mongosh) for command line operations or MongoDB Compass for a graphical interface.

4. Connect to MongoDB: Connect to your MongoDB instance using a MongoDB driver or client.

1. Create Collection

```
db.createCollection("students");
```

Output:- { ok: 1 }

2. Insert Documents (Create)

i. Insert a single student document

```
db.students.insertOne({
usn: "001",
name: "Johnson",
sem: 5,
subjects: ["Mathematics", "Physics", "Chemistry"]
});
```

Output:-

```
{
 acknowledged: true,
 insertedId: ObjectId('669d13531cd03857e74ec05e')
}
```

ii. Insert multiple student documents

```
db.students.insertMany([
   {
      usn: "002",
      name: "Smith",
      sem: 6,
      subjects: ["Biology", "Chemistry", "English"]
   },
   {
```

```
        usn: "003",
        name: "Charlie",
        sem: 4,
        subjects: ["History", "Geography", "Political Science"]
    },
    {
        usn: "004",
        name: "Miller",
        sem: 5,
        subjects: ["Mathematics", "Computer Science", "Physics"]
    }
]);
```

Output:-

```
{
acknowledged: true,
  insertedIds: {
    '0': ObjectId('669d139be3a7d3f1594ec05f'),
    '1': ObjectId('669d139be3a7d3f1594ec060'),
    '2': ObjectId('669d139be3a7d3f1594ec061')
  }
}
```

3. Query Documents (Read)

Find all students

```
db.students.find({});
```

Output:-

```
[
  {
    _id: ObjectId('669d13fd6b0c041d674ec05e'),
    usn: '001',
    name: 'Johnson',
    sem: 5,
    subjects: [ 'Mathematics', 'Physics', 'Chemistry' ]
  },
  {
    _id: ObjectId('669d13fe6b0c041d674ec05f'),
    usn: '002',
    name: 'Smith',
    sem: 6,
    subjects: [ 'Biology', 'Chemistry', 'English' ]
```

```
  },
  {
    _id: ObjectId('669d13fe6b0c041d674ec060'),
    usn: '003',
    name: 'Charlie',
    sem: 4,
    subjects: [ 'History', 'Geography', 'Political Science' ]
  },
  {
    _id: ObjectId('669d13fe6b0c041d674ec061'),
    usn: '004',
    name: 'Miller',
    sem: 5,
    subjects: [ 'Mathematics', 'Computer Science', 'Physics' ]
  }
]
```

Find students in a specific semester

```
db.students.find({ sem: 5 });
```

Output:-

```
[
  {
    _id: ObjectId('669d1423d301ff6e404ec05e'),
    usn: '001',
    name: 'Johnson',
    sem: 5,
    subjects: [ 'Mathematics', 'Physics', 'Chemistry' ]
  },
  {
    _id: ObjectId('669d1423d301ff6e404ec061'),
    usn: '004',
    name: 'Miller',
    sem: 5,
    subjects: [ 'Mathematics', 'Computer Science', 'Physics' ]
  }
]
```

Find a single student by USN

```
db.students.findOne({ usn: "001" });
```

Output:-

```
{
  _id: ObjectId('669d14511dcfe211144ec05e'),
  usn: '001',
  name: 'Johnson',
  sem: 5,
  subjects: [ 'Mathematics', 'Physics', 'Chemistry' ]
}
```

Find students enrolled in a specific subject

```
db.students.find({ subjects: "Mathematics" });
```

Output:-

```
[
  {
    _id: ObjectId('669d1486dde375e6344ec05e'),
    usn: '001',
    name: 'Johnson',
    sem: 5,
    subjects: [ 'Mathematics', 'Physics', 'Chemistry' ]
  },
  {
    _id: ObjectId('669d1486dde375e6344ec061'),
    usn: '004',
    name: 'Miller',
    sem: 5,
    subjects: [ 'Mathematics', 'Computer Science', 'Physics' ]
  }
]
```

4. Update Documents (Update)

Update a single student's semester

```
db.students.updateOne( { usn: "001" },  { $set: { sem: 6 } } );
```

Output:-

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
```

}

Update multiple students' subjects

```
db.students.updateMany( { sem: 5 },  { $addToSet: { subjects: "Elective Course" } } );
```

Output:-

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

5. Delete Documents (Delete)

Delete a single student document

```
db.students.deleteOne({ usn: "003" });
```

Output:-

```
{ acknowledged: true, deletedCount: 1 }
```

Delete multiple students from a specific semester

```
db.students.deleteMany({ sem: 6 });
```

Output:-

```
{ acknowledged: true, deletedCount: 1 }
```