

**Name: Thorve Avishkar Shrikrushna**

**Roll No.: 62**

**Practical No.01**

**Query:**

-- 1. Create Employees Table with constraints

```
CREATE TABLE Employees (  
    EmpID INT PRIMARY KEY,  
    FirstName VARCHAR(50),  
    LastName VARCHAR(50),  
    Department VARCHAR(50),  
    Salary DECIMAL(10, 2),  
    JoiningDate DATE,  
    Email VARCHAR(100) UNIQUE,  
    CONSTRAINT chk_Salary CHECK (Salary > 0)  
);
```

-- 2. Create an Index on Department

```
CREATE INDEX idx_Department ON Employees (Department);
```

-- 3. Insert sample data

```
INSERT INTO Employees (EmpID, FirstName, LastName, Department, Salary, JoiningDate, Email)  
VALUES (1, 'John', 'Doe', 'IT', 60000, '2023-05-15', 'john.doe@example.com');
```

```
INSERT INTO Employees (EmpID, FirstName, LastName, Department, Salary, JoiningDate, Email)  
VALUES (2, 'Jane', 'Smith', 'HR', 45000, '2022-10-12', 'jane.smith@example.com');
```

-- 4. Select all data from Employees

```
SELECT * FROM Employees;
```

-- 5. Update Salary for a specific Employee

```
UPDATE Employees  
SET Salary = 70000  
WHERE EmpID = 1;
```

-- 6. Delete an Employee

```
DELETE FROM Employees  
WHERE EmpID = 2;
```

-- 7. Select Employees with Salary > 50000

```
SELECT FirstName, LastName, Salary
```

FROM Employees  
WHERE Salary > 50000;

**Output :**

**Output:**

EmpID	FirstName	LastName	Department	Salary	JoiningDate	Email
1	John	Doe	IT	60000	2023-05-15	<a href="mailto:john.doe@example.com">john.doe@example.com</a>
2	Jane	Smith	HR	45000	2022-10-12	<a href="mailto:jane.smith@example.com">jane.smith@example.com</a>

Then, after the update query where John's salary is increased:

**Updated Output:**

EmpID	FirstName	LastName	Department	Salary	JoiningDate	Email
1	John	Doe	IT	70000	2023-05-15	<a href="mailto:john.doe@example.com">john.doe@example.com</a>

After the delete query where Jane is removed:

**Final Output:**

EmpID	FirstName	LastName	Department	Salary	JoiningDate	Email
1	John	Doe	IT	70000	2023-05-15	<a href="mailto:john.doe@example.com">john.doe@example.com</a>

**Name: Thorve Avishkar Shrikrushna**

**Roll No.: 62**

## **Practical No.02**

### **Tables Setup:**

To demonstrate the joins, sub-queries, and views, let's assume we have the following two tables:

**Employees Table:**

EmpID	FirstName	LastName	Department	Salary	ManagerID
1	John	Doe	IT	70000	3
2	Jane	Smith	HR	45000	3
3	Mike	Johnson	IT	80000	NULL
4	Chris	Evans	Sales	55000	5

**Departments Table:**

DeptID	DeptName	DeptHead
1	IT	3
2	HR	2
3	Sales	5

### **1. INNER JOIN: Join Employees with Departments (matching records only)**

```
SELECT e.FirstName, e.LastName, d.DeptName
FROM Employees e
INNER JOIN Departments d
ON e.Department = d.DeptName;
```

### **Output:**

FirstName	LastName	DeptName
John	Doe	IT
Mike	Johnson	IT
Jane	Smith	HR

-- **LEFT JOIN:** Select all Employees with Departments, showing NULL where no department exists

```
SELECT e.FirstName, e.LastName, d.DeptName
FROM Employees e
LEFT JOIN Departments d
ON e.Department = d.DeptName;
```

**Output:**

FirstName	LastName	DeptName
John	Doe	IT
Mike	Johnson	IT
Jane	Smith	HR
Chris	Evans	NULL

-- **RIGHT JOIN:** Select all Departments with Employees, showing NULL where no employee exists

```
SELECT e.FirstName, e.LastName, d.DeptName
FROM Employees e
RIGHT JOIN Departments d
ON e.Department = d.DeptName;
```

**Output:**

FirstName	LastName	DeptName
John	Doe	IT
Mike	Johnson	IT
Jane	Smith	HR
NULL	NULL	Sales

-- **FULL OUTER JOIN:** Select all Employees and all Departments, showing NULL where no match exists

```
SELECT e.FirstName, e.LastName, d.DeptName
FROM Employees e
FULL OUTER JOIN Departments d
ON e.Department = d.DeptName;
```

**Output:**

FirstName	LastName	DeptName
John	Doe	IT
Mike	Johnson	IT
Jane	Smith	HR
Chris	Evans	NULL
NULL	NULL	Sales

-- **SELF JOIN:** Find Employees and their Managers

```
SELECT e.FirstName AS Employee, m.FirstName AS Manager
FROM Employees e
LEFT JOIN Employees m
ON e.ManagerID = m.EmpID;
```

**Output:**

Employee	Manager
John	Mike
Jane	Mike
Mike	NULL
Chris	NULL

-- **Sub-Query:** Select Employees whose salary is greater than the average salary of all Employees

```
SELECT FirstName, LastName, Salary
FROM Employees
WHERE Salary > (SELECT AVG(Salary) FROM Employees);
```

**Output:**

FirstName	LastName	Salary
John	Doe	70000
Mike	Johnson	80000

-- **Sub-Query:** Select Departments where Employees are currently working

```
SELECT DeptName
FROM Departments
WHERE DeptID IN (SELECT DISTINCT Department FROM Employees);
```

**Output:**

DeptName
IT
HR

-- **Correlated Sub-Query:** Select Employees whose salary is greater than the average salary of their department

```
SELECT e.FirstName, e.LastName, e.Salary
FROM Employees e
WHERE e.Salary > (SELECT AVG(Salary)
                  FROM Employees
```

WHERE Department = e.Department);

**Output:**

FirstName	LastName	Salary
Mike	Johnson	80000

-- **View:** Create a view for Employees with Salary greater than 60000

CREATE VIEW HighSalaryEmployees AS

SELECT FirstName, LastName, Department, Salary

FROM Employees

WHERE Salary > 60000;

-- Query to select from the view

SELECT \* FROM HighSalaryEmployees;

**Output:**

FirstName	LastName	Department	Salary
John	Doe	IT	70000
Mike	Johnson	IT	80000

-- **CROSS JOIN:** Select all possible combinations of Employees and Departments

SELECT e.FirstName, e.LastName, d.DeptName

FROM Employees e CROSS JOIN Departments d;

**Output:**

FirstName	LastName	DeptName
John	Doe	IT
John	Doe	HR
John	Doe	Sales
Jane	Smith	IT
Jane	Smith	HR
Jane	Smith	Sales
Mike	Johnson	IT
Mike	Johnson	HR
Mike	Johnson	Sales
Chris	Evans	IT
Chris	Evans	HR
Chris	Evans	Sales

**Name: Thorve Avishkar Shrikrushna**

**Roll No.: 62**

**Practical No.03**

**Insert Documents (Create Operation):**

// Insert a single document

```
db.employees.insertOne({  
  EmpID: 1,  
  FirstName: "John",  
  LastName: "Doe",  
  Department: "IT",  
  Salary: 70000,  
  ManagerID: 3  
});
```

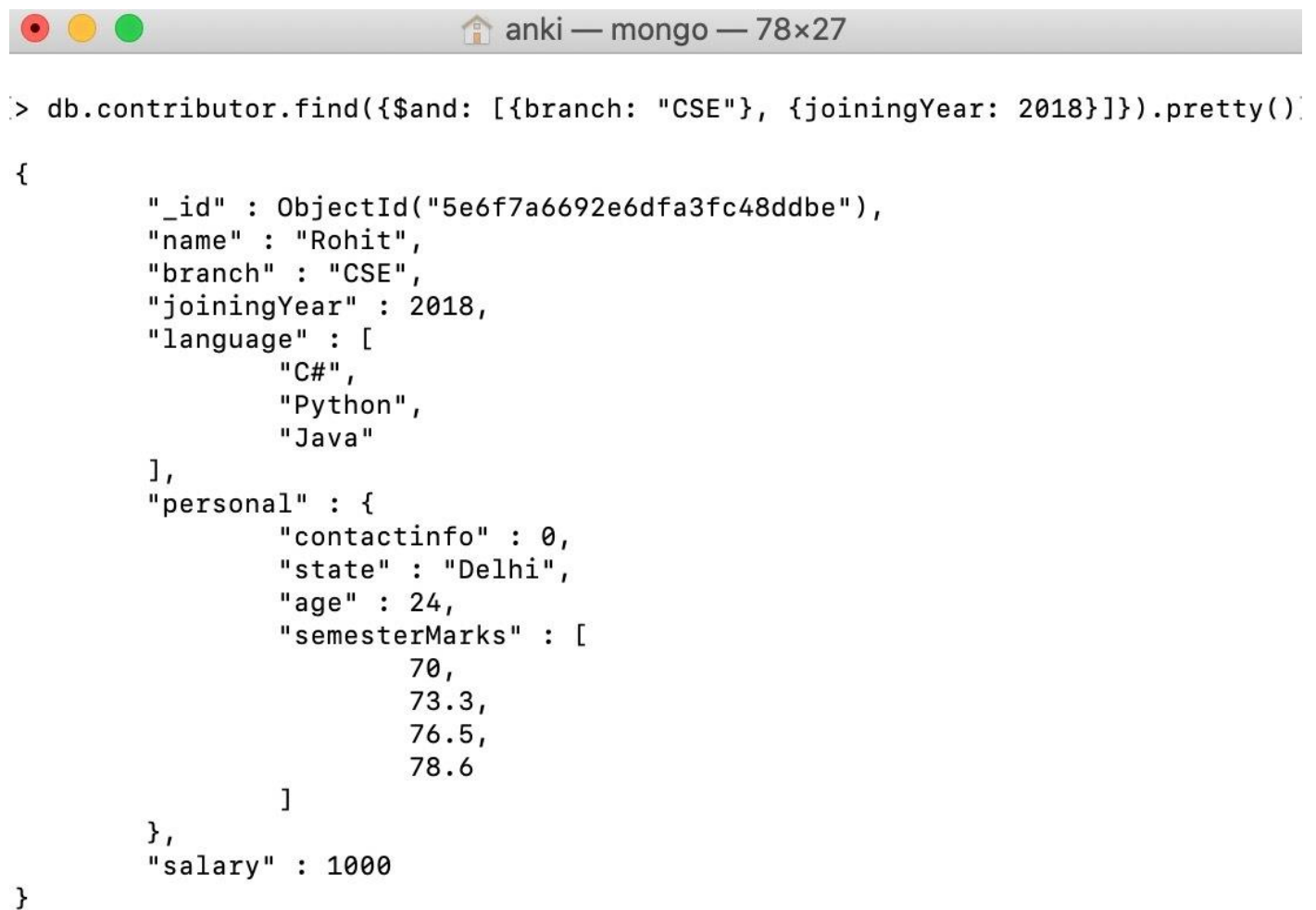
// Insert multiple documents

```
db.employees.insertMany([  
  {  
    EmpID: 2,  
    FirstName: "Jane",  
    LastName: "Smith",  
    Department: "HR",  
    Salary: 45000,  
    ManagerID: 3  
  },  
  {  
    EmpID: 3,  
    FirstName: "Mike",  
    LastName: "Johnson",  
    Department: "IT",  
    Salary: 80000,  
    ManagerID: null  
  }  
]);
```

**Query : Matching values using \$and operator**

```
db.contributor.find({$and: [{branch: "CSE"}, {joiningYear: 2062}]}).pretty()
```

### Output :



```
> db.contributor.find({$and: [{branch: "CSE"}, {joiningYear: 2018}]}).pretty()
{
  "_id" : ObjectId("5e6f7a6692e6dfa3fc48ddbe"),
  "name" : "Rohit",
  "branch" : "CSE",
  "joiningYear" : 2018,
  "language" : [
    "C#",
    "Python",
    "Java"
  ],
  "personal" : {
    "contactinfo" : 0,
    "state" : "Delhi",
    "age" : 24,
    "semesterMarks" : [
      70,
      73.3,
      76.5,
      78.6
    ]
  },
  "salary" : 1000
}
```



**Name: Thorve Avishkar Shrikrushna**

**Roll No.: 62**

**Practical No. 04**

**Schema:**

1. **Borrower(Rollin, Name, DateofIssue, NameofBook, Status)**
2. **Fine(Roll\_no,Date,Amt)**
  - *Accept roll\_no & name of book from user.*
  - *Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.*
  - *If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.*
  - *After submitting the book, status will change from I to R.*
  - *If condition of fine is true, then details will be stored into fine table.*

```
SQL> create table Borrower(  
2 rollin integer,  
3 name varchar(20) not null,  
4 dateofissue date not null,  
5 nameofbook varchar(20) not null,  
6 status varchar(2) default 'I');
```

Table created.

```
insert into Borrower values (1,'Shree',TO_DATE('08/09/2020','dd/mm/yyyy'),'CN','I');  
insert into Borrower values (2,'Krishna',TO_DATE('20/09/2020','dd/mm/yyyy'),'DBMS','I');  
insert into Borrower values (3,'Madhav',TO_DATE('25/10/2020','dd/mm/yyyy'),'ISEE','I');  
insert into Borrower values (4,'Mukund',TO_DATE('03/10/2020','dd/mm/yyyy'),'SEPM','I');  
insert into Borrower values (5,'Mohan',TO_DATE('27/10/2020','dd/mm/yyyy'),'TOC','I');  
insert into Borrower values (1,'Shree',TO_DATE('25/09/2020','dd/mm/yyyy'),'TOC','I');
```

```
SQL> create table fine(  
2 rollno integer,  
3 fine_date date not null,  
4 amt float not null);
```

Table created.

```
create or replace procedure finer(r in number, book_name in varchar) as days number;  
begin  
select to_number(trunc(sysdate-dateofissue)) into days from Borrower where rollin=r and nameofbook=book_name;  
if days < 15 then  
insert into Fine values (r,sysdate,0);  
elsif days > 15 and days < 30 then
```

```

insert into Fine values (r,sysdate,0);

update fine set amt = amt + (days*5) where rollno=r;
else

days := days-30;

insert into Fine values (r,sysdate,0);

update fine set amt = amt + (days*50)+75 where rollno=r;

end if;

update borrower set status='R' where rollin=r and nameofbook=book_name;

end;

/

```

SQL> execute finer(1,'CN');

SQL> execute finer(1,'TOC');

SQL> execute finer(2,'DBMS');

PL/SQL procedure successfully completed.

SQL> select \* from fine;

ROLLNO	FINE_DATE	AMT
1	08-OCT-20	75
1	08-OCT-20	0
2	08-OCT-20	90

SQL> select \* from borrower;

ROLLIN	NAME	DATEOFISS	NAMEOFBOOK	ST
1	Shree	08-SEP-20	CN	R
2	Krishna	20-SEP-20	DBMS	R
3	Madhav	25-OCT-20	ISEE	I
4	Mukund	03-OCT-20	SEPM	I
5	Mohan	27-OCT-20	TOC	I
1	Shree	25-SEP-20	TOC	R

6 rows selected.

**Name: Thorve Avishkar Shrikrushna**

**Roll No.: 62**

**Practical No.05**

**Cursor**

```
SQL> create table oroll(  
2 roll integer primary key,  
3 name varchar(10) not null,  
4 year varchar(3) not null);
```

Table created.

INSERT some values

```
insert into oroll values(1,'Shree','TE');  
insert into oroll values(2,'Krishna','BE');  
insert into oroll values(3,'Madhav','SE');  
insert into oroll values(4,'Govind','TE');
```

```
SQL> create table nroll(  
2 roll integer primary key,  
3 name varchar(10) not null,  
4 year varchar(3) not null);
```

Table created.

INSERT some values

```
insert into nroll values(1,'Shree','TE');  
insert into nroll values(2,'Krishna','BE');
```

Key Statement here is :

```
select * from oroll minus select * from nroll;
```

*OR*

```
select * from oroll where roll not in(select roll from nroll);
```

```
SQL> select * from oroll minus select * from nroll;
```

ROLL NAME	YEA
3 Madhav	SE
4 Govind	TE

declare

```
c_roll oroll.roll%type;  
c_name oroll.name%type;
```

```

c_year oroll.year%type;

cursor c_temp is select * from oroll minus select * from nroll;

begin

open c_temp;

loop

fetch c_temp into c_roll,c_name,c_year;

exit when c_temp%notfound;

insert into nroll values(c_roll,c_name,c_year);

dbms_output.put_line(c_roll || ' ' || c_name || ' ' || c_year);

end loop;

close c_temp;

end;

/

```

#### Output

3 Madhav SE

4 Govind TE

PL/SQL procedure successfully completed.

SQL> select \* from nroll;

ROLL NAME	YEA
1 Shree	TE
2 Krishna	BE
3 Madhav	SE
4 Govind	TE

**Name: Thorve Avishkar Shrikrushna**

**Roll No.: 62**

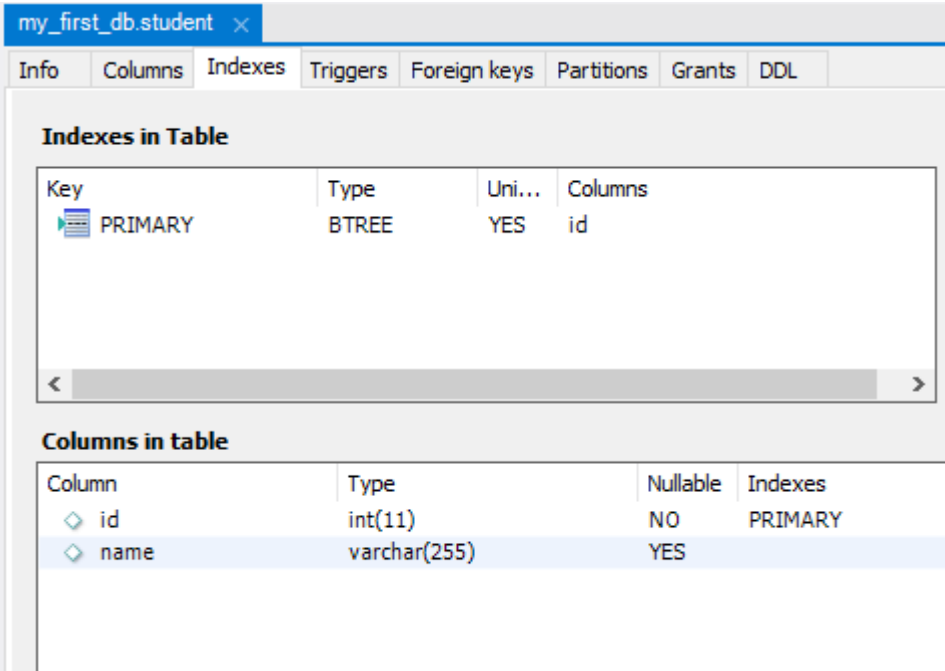
**Practical No.06**

```
import mysql.connector

db_connection = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="root",
    database="my_first_db"
)

db_cursor = db_connection.cursor()
#Here we modify existing column id
db_cursor.execute("ALTER TABLE student MODIFY id INT PRIMARY KEY")
```

**Output:**



The screenshot shows the MySQL Workbench interface for the 'my\_first\_db.student' table. The 'Indexes' tab is selected, displaying the 'Indexes in Table' section. Below this, a table lists the primary key index. The 'Columns in table' section is also visible, showing the structure of the table.

Key	Type	Uni...	Columns
PRIMARY	BTREE	YES	id

Column	Type	Nullable	Indexes
id	int(11)	NO	PRIMARY
name	varchar(255)	YES	