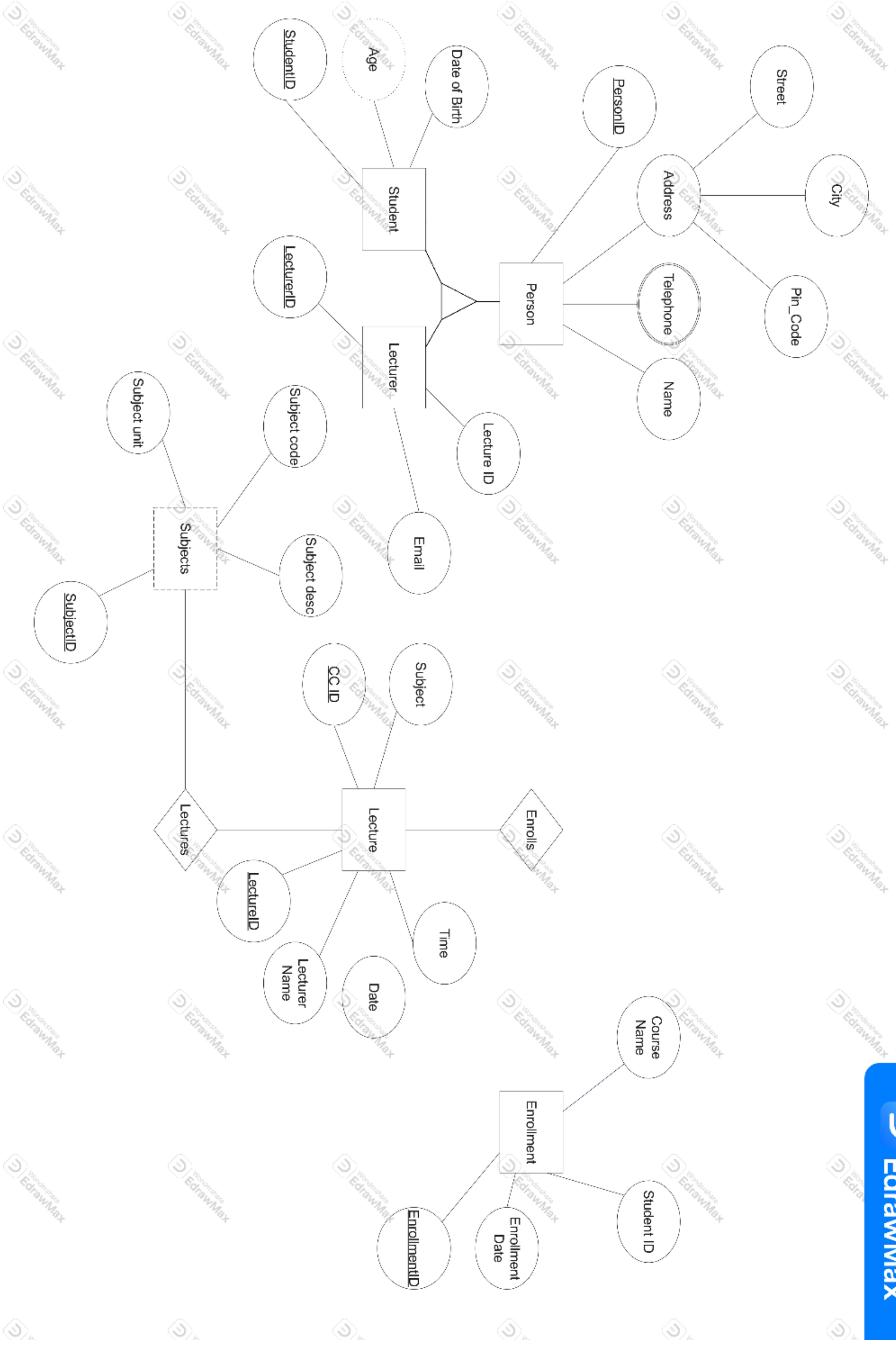
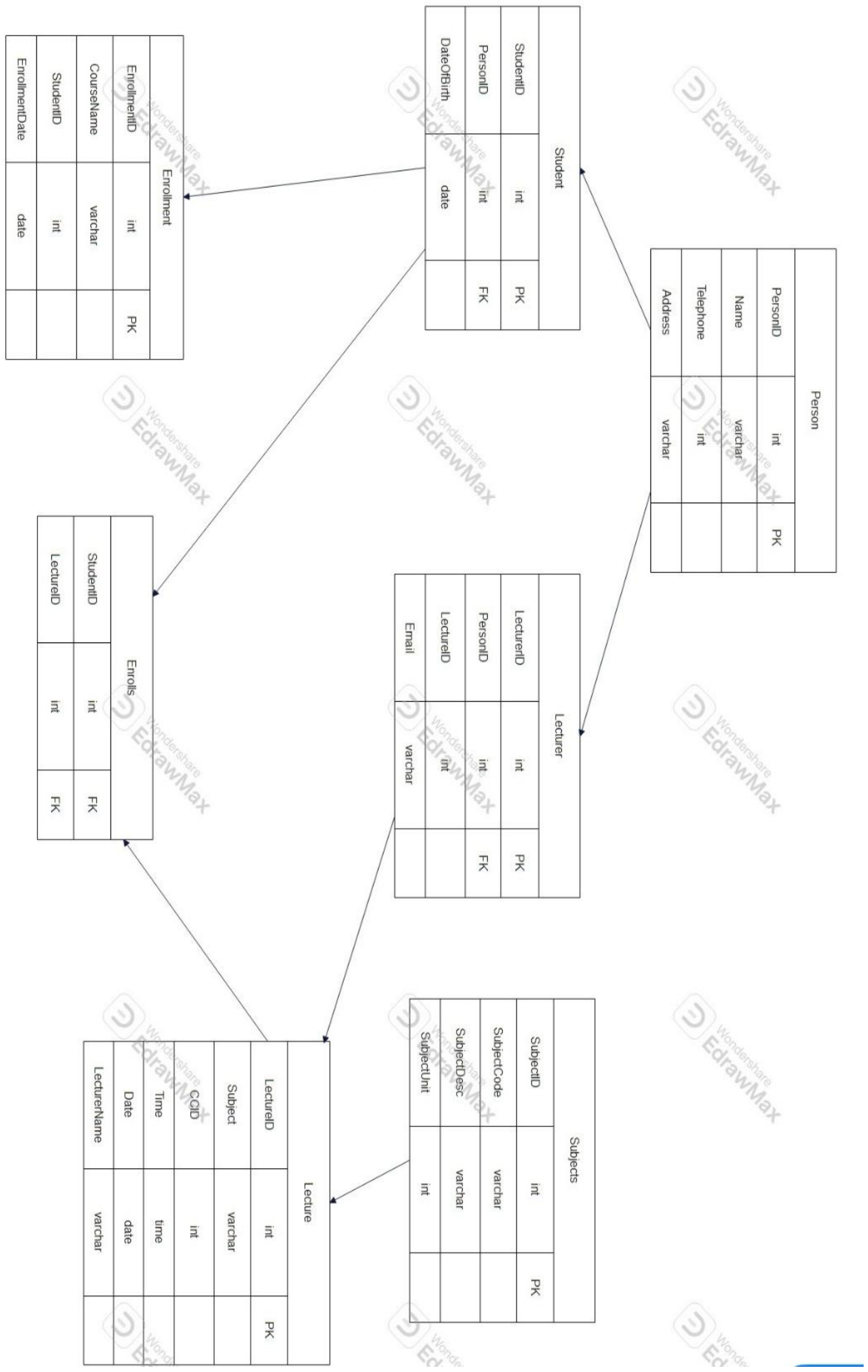
Name: Kale Prathmesh

Roll no: 55 Batch: A3

Practical no.:1



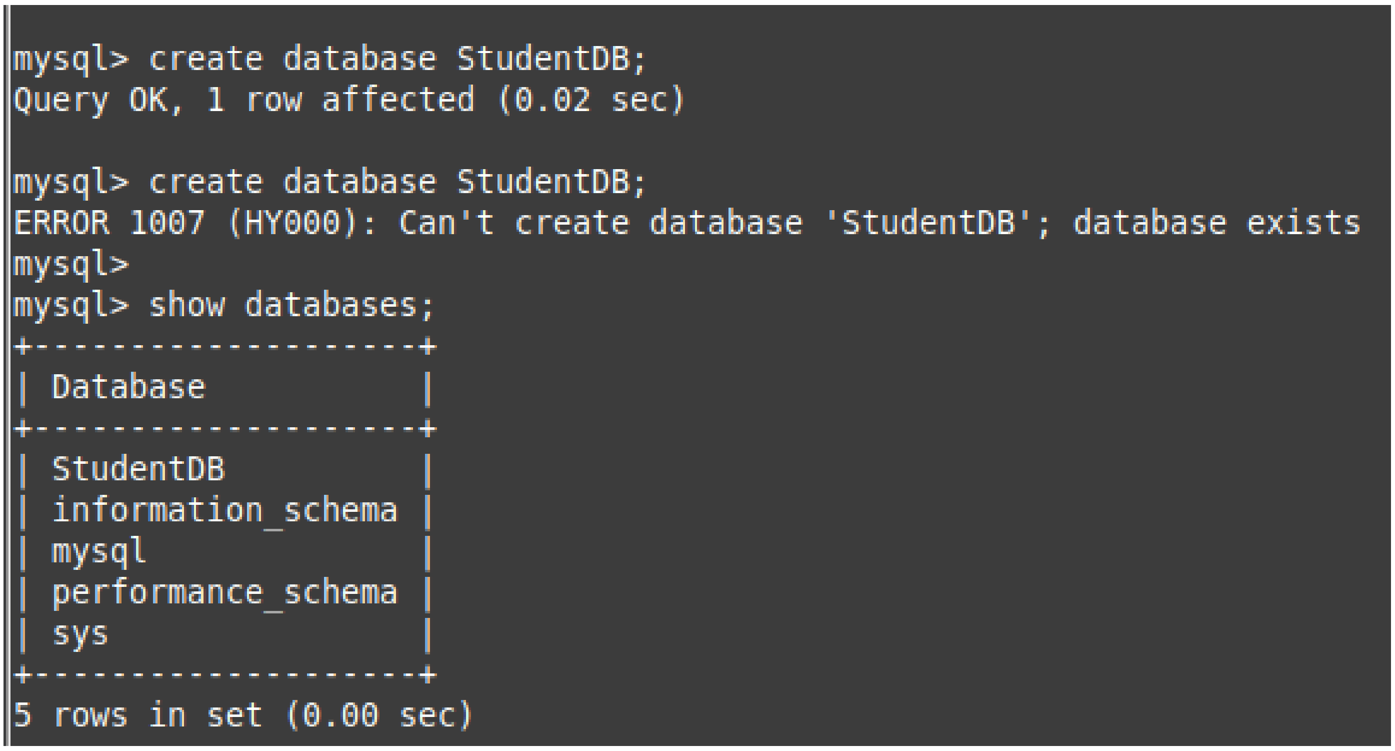

Person
PersonID int PK
Name varchar
Telephone int
Address varchar
Student
StudentID int PK
PersonID int FK
DateOfBirth date
Lecturer
LecturerID int PK
PersonID int FK
LectureID int
Email varchar
Enrollment
EnrollmentID int PK
CourseName varchar
StudentID int
EnrollmentDate date
Enrolls
StudentID int FK
LectureID int FK
Subjects
SubjectID int PK
SubjectCode varchar
SubjectDesc varchar
SubjectUnit int
Lecture
LectureID int PK
Subject varchar
CCID int
Time time
Date date
LecturerName varchar

Name: Kale Prathmesh

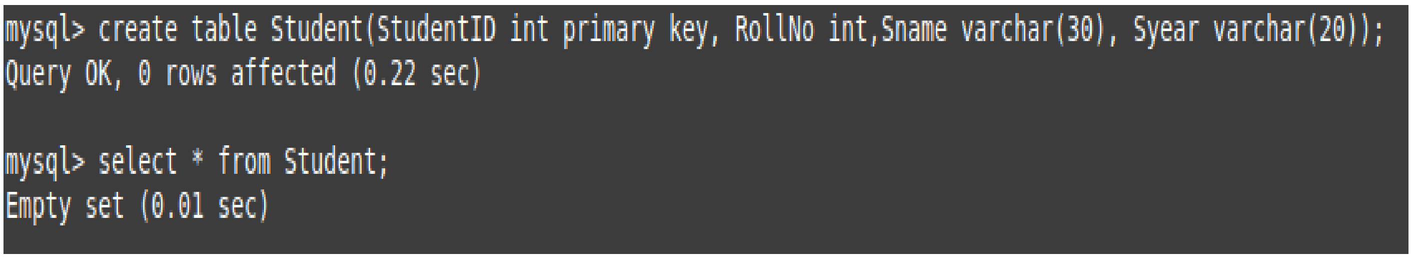
Div: A **Subject:** DBMSL

Practical no.: 2

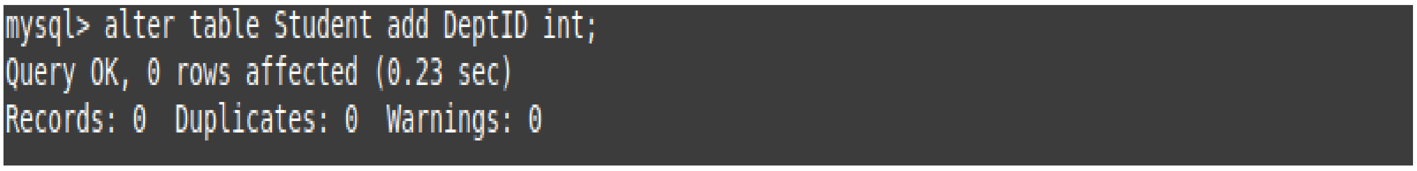
Output:



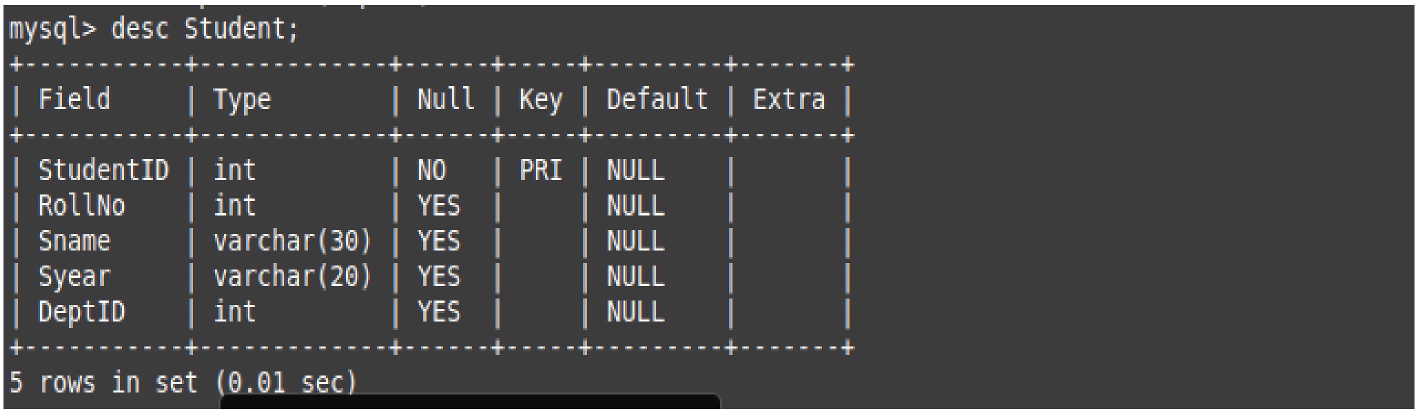
```
mysql> create database StudentDB;
Query OK, 1 row affected (0.02 sec)

mysql> create database StudentDB;
ERROR 1007 (HY000): Can't create database 'StudentDB'; database exists
mysql>
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| StudentDB          |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)
```



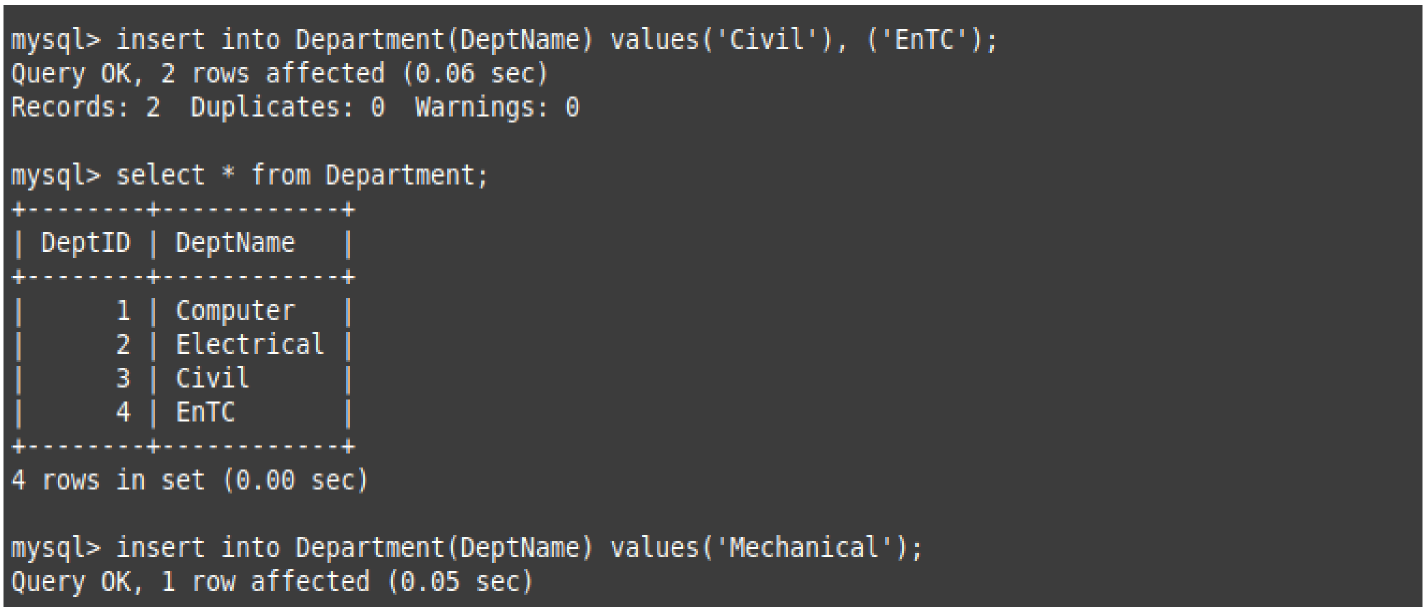
```
mysql> create table Student(StudentID int primary key, RollNo int,Sname varchar(30), Syear varchar(20));
Query OK, 0 rows affected (0.22 sec)

mysql> select * from Student;
Empty set (0.01 sec)
```



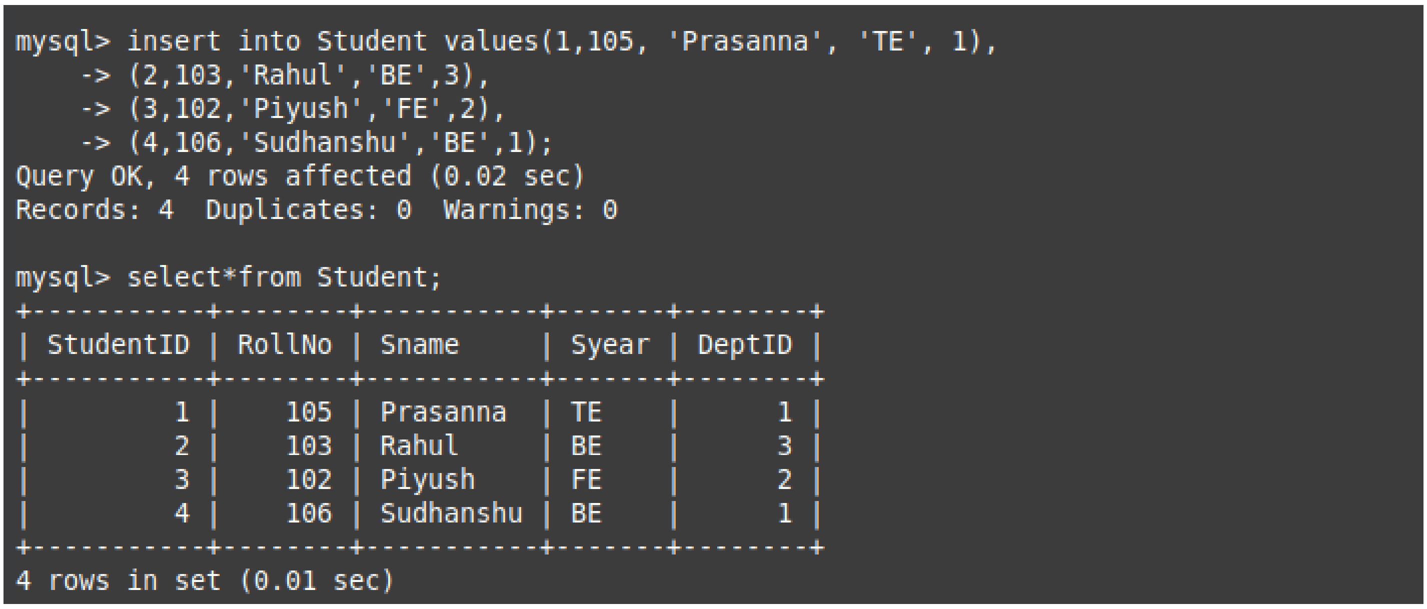
```
mysql> alter table Student add DeptID int;
Query OK, 0 rows affected (0.23 sec)
Records: 0  Duplicates: 0  Warnings: 0
```



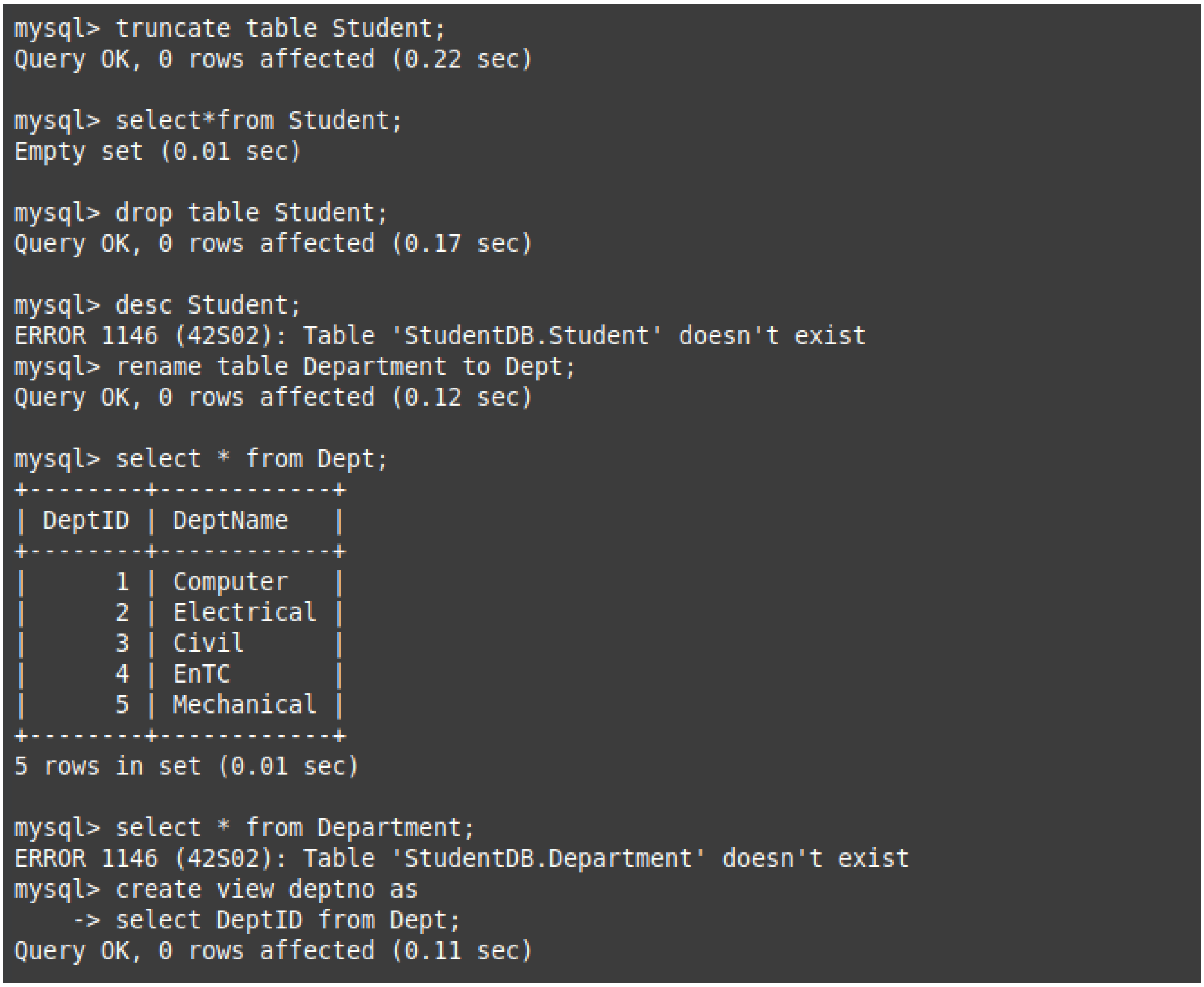
```
mysql> desc Student;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| StudentID | int         | NO   | PRI | NULL    |       |
| RollNo    | int         | YES  |     | NULL    |       |
| Sname     | varchar(30) | YES  |     | NULL    |       |
| Syear     | varchar(20) | YES  |     | NULL    |       |
| DeptID    | int         | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```



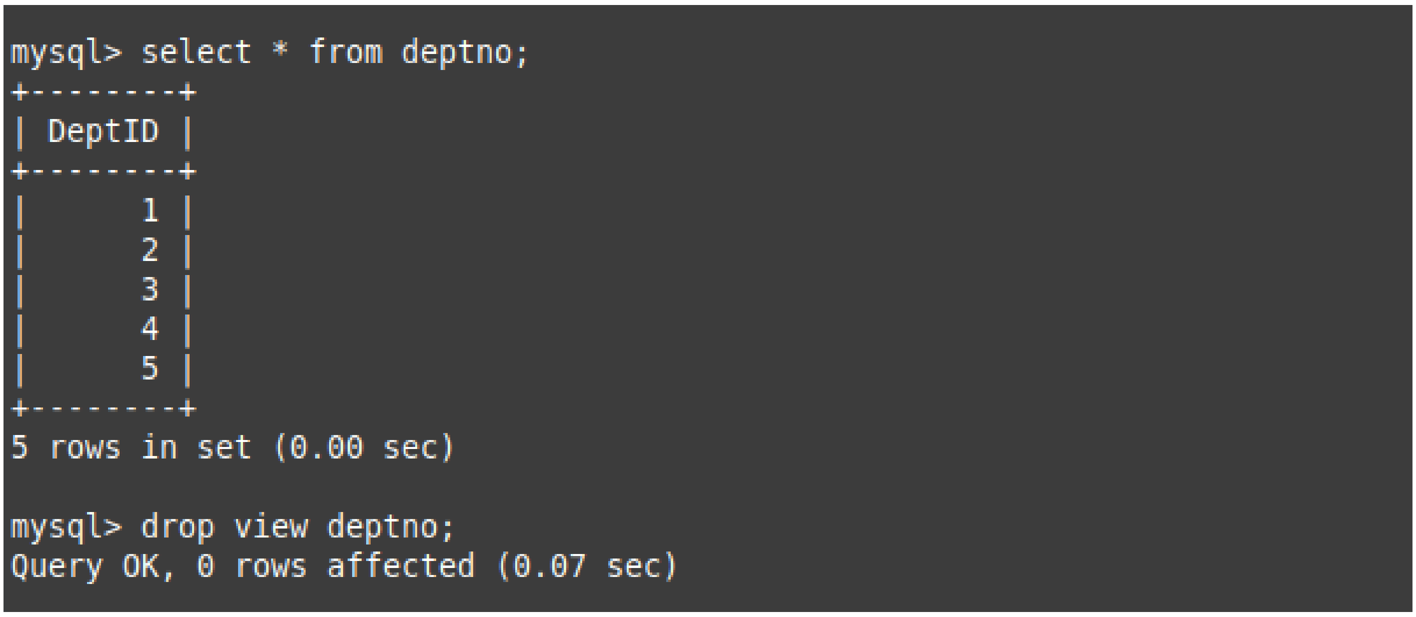
```
mysql> insert into Department(DeptName) values('Civil'), ('EnTC');
Query OK, 2 rows affected (0.06 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from Department;
+--------+------------+
| DeptID | DeptName   |
+--------+------------+
|      1 | Computer   |
|      2 | Electrical |
|      3 | Civil      |
|      4 | EnTC       |
+--------+------------+
4 rows in set (0.00 sec)

mysql> insert into Department(DeptName) values('Mechanical');
Query OK, 1 row affected (0.05 sec)
```



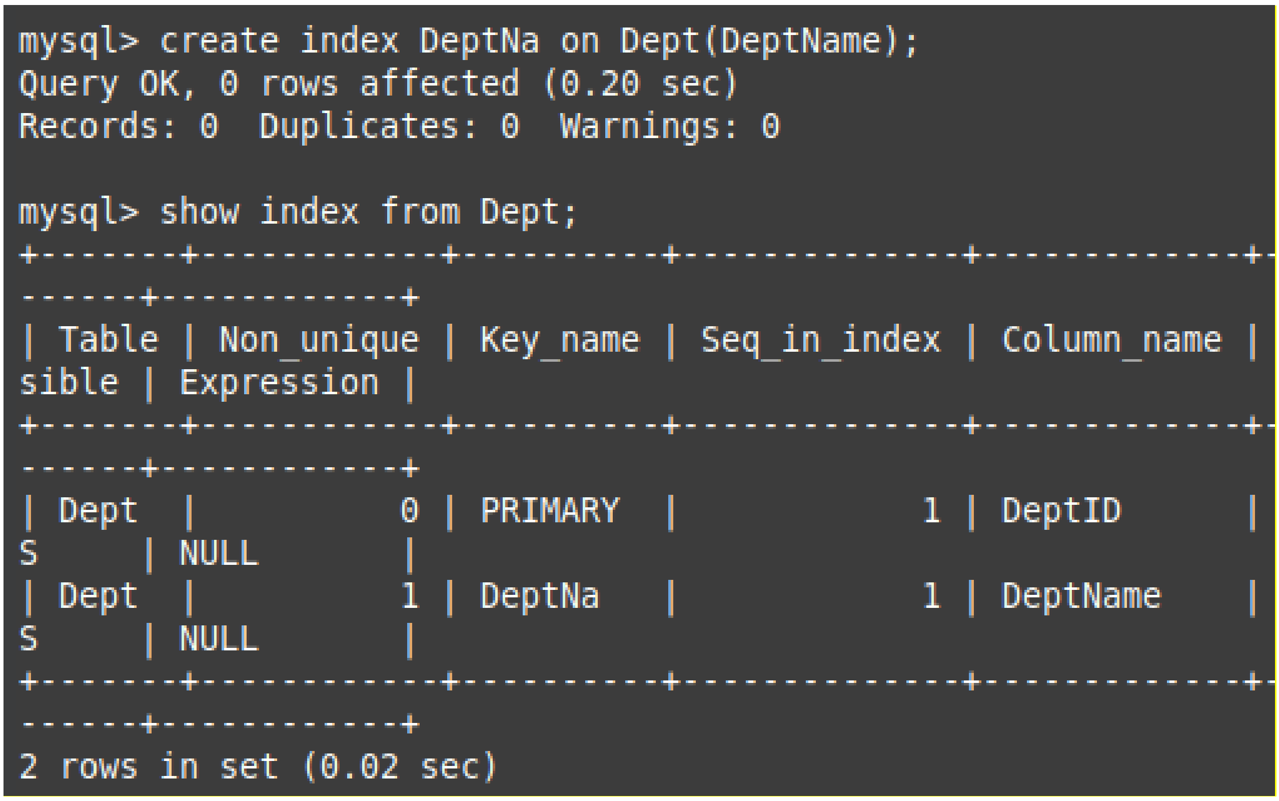
```
mysql> insert into Student values(1,105, 'Prasanna', 'TE', 1),
    -> (2,103,'Rahul','BE',3),
    -> (3,102,'Piyush','FE',2),
    -> (4,106,'Sudhanshu','BE',1);
Query OK, 4 rows affected (0.02 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select*from Student;
+-----------+--------+-----------+-------+--------+
| StudentID | RollNo | Sname     | Syear | DeptID |
+-----------+--------+-----------+-------+--------+
|         1 |    105 | Prasanna  | TE    |      1 |
|         2 |    103 | Rahul     | BE    |      3 |
|         3 |    102 | Piyush    | FE    |      2 |
|         4 |    106 | Sudhanshu | BE    |      1 |
+-----------+--------+-----------+-------+--------+
4 rows in set (0.01 sec)
```



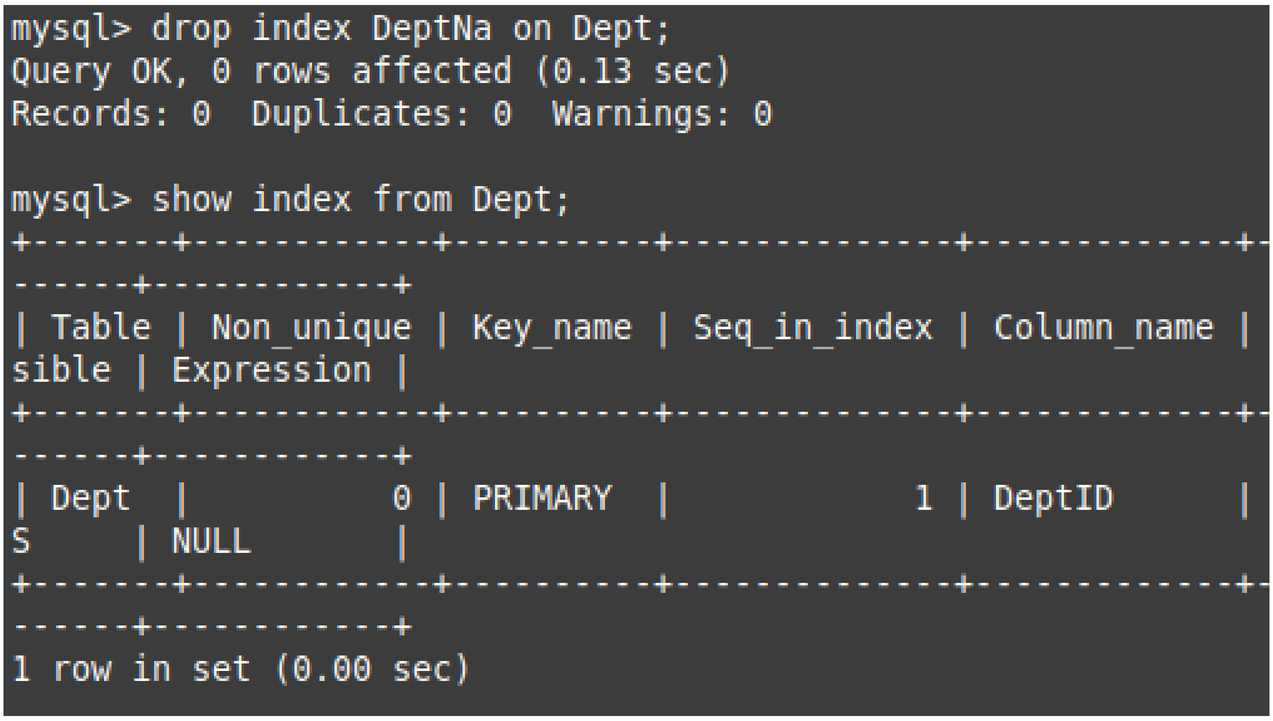
```
mysql> truncate table Student;
Query OK, 0 rows affected (0.22 sec)

mysql> select*from Student;
Empty set (0.01 sec)

mysql> drop table Student;
Query OK, 0 rows affected (0.17 sec)

mysql> desc Student;
ERROR 1146 (42S02): Table 'StudentDB.Student' doesn't exist
mysql> rename table Department to Dept;
Query OK, 0 rows affected (0.12 sec)

mysql> select * from Dept;
+--------+------------+
| DeptID | DeptName   |
+--------+------------+
|      1 | Computer   |
|      2 | Electrical |
|      3 | Civil      |
|      4 | EnTC       |
|      5 | Mechanical |
+--------+------------+
5 rows in set (0.01 sec)

mysql> select * from Department;
ERROR 1146 (42S02): Table 'StudentDB.Department' doesn't exist
mysql> create view deptno as
    -> select DeptID from Dept;
Query OK, 0 rows affected (0.11 sec)
```



```
mysql> select * from deptno;
+--------+
| DeptID |
+--------+
|      1 |
|      2 |
|      3 |
|      4 |
|      5 |
+--------+
5 rows in set (0.00 sec)

mysql> drop view deptno;
Query OK, 0 rows affected (0.07 sec)
```



```
mysql> create index DeptNa on Dept(DeptName);
Query OK, 0 rows affected (0.20 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from Dept;
+-------+------------+----------+--------------+-------------+-
------+------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name |
sible | Expression |
+-------+------------+----------+--------------+-------------+-
------+------------+
| Dept  |          0 | PRIMARY  |            1 | DeptID      |
S     | NULL       |
| Dept  |          1 | DeptNa   |            1 | DeptName    |
S     | NULL       |
+-------+------------+----------+--------------+-------------+-
------+------------+
2 rows in set (0.02 sec)
```

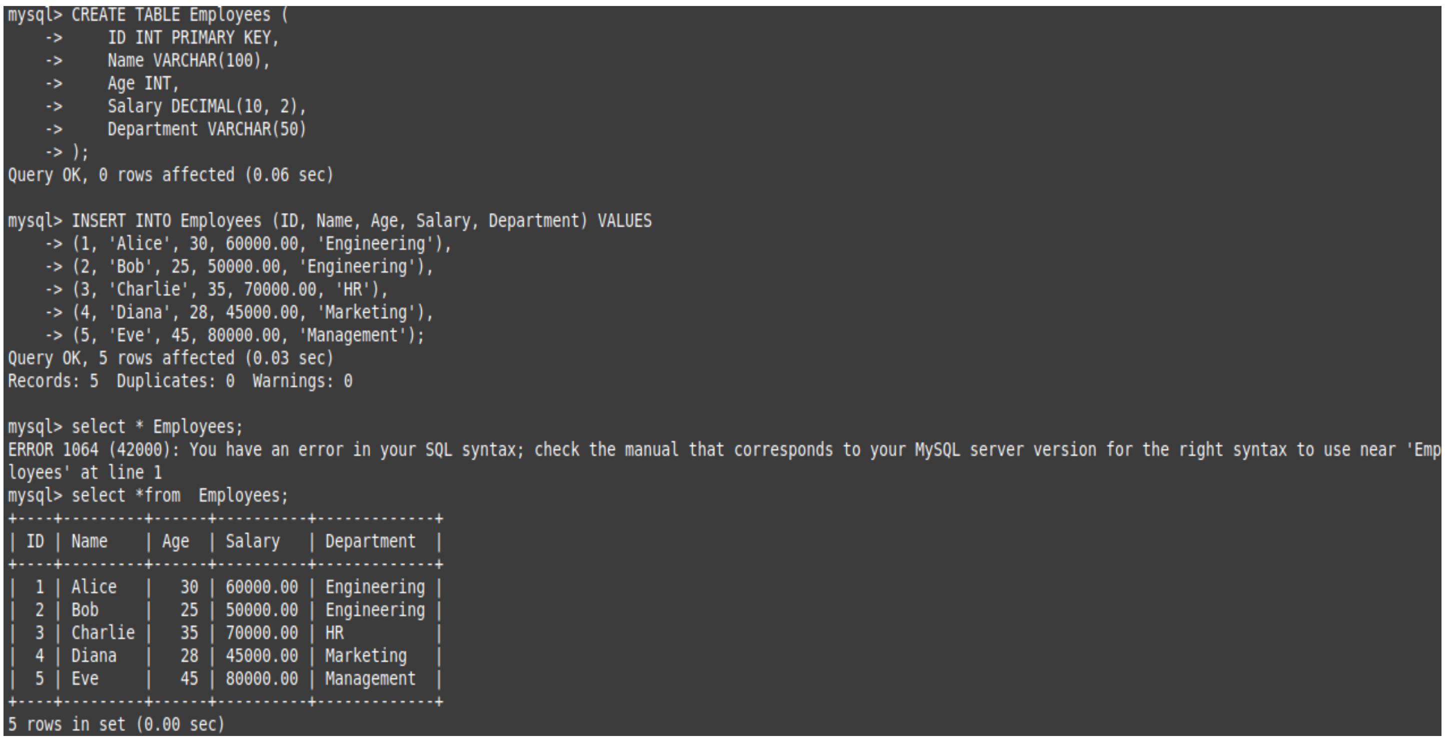


```
mysql> drop index DeptNa on Dept;
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show index from Dept;
+-------+------------+----------+--------------+-------------+-
------+------------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name |
sible | Expression |
+-------+------------+----------+--------------+-------------+-
------+------------+
| Dept  |          0 | PRIMARY  |            1 | DeptID      |
S     | NULL       |
+-------+------------+----------+--------------+-------------+-
------+------------+
1 row in set (0.00 sec)
```

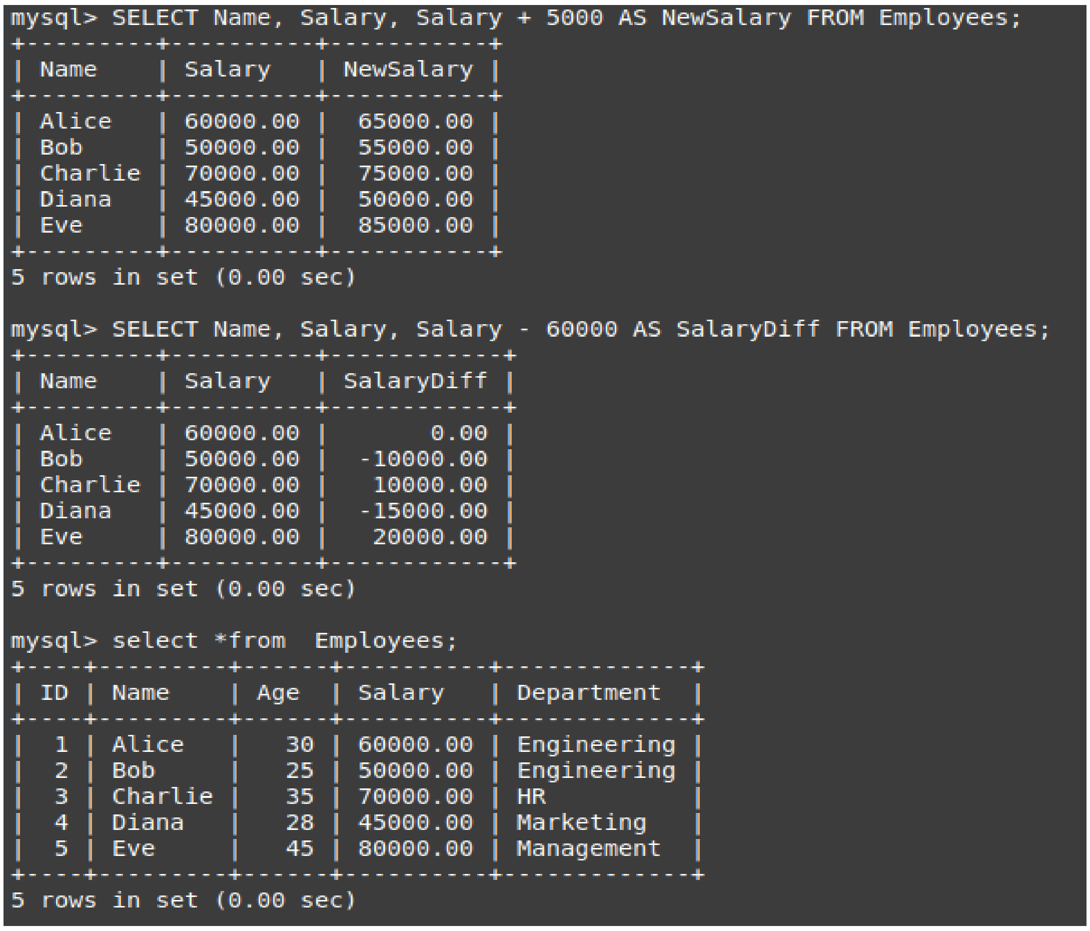
Practical 2_2

Name: Kale Prathmesh

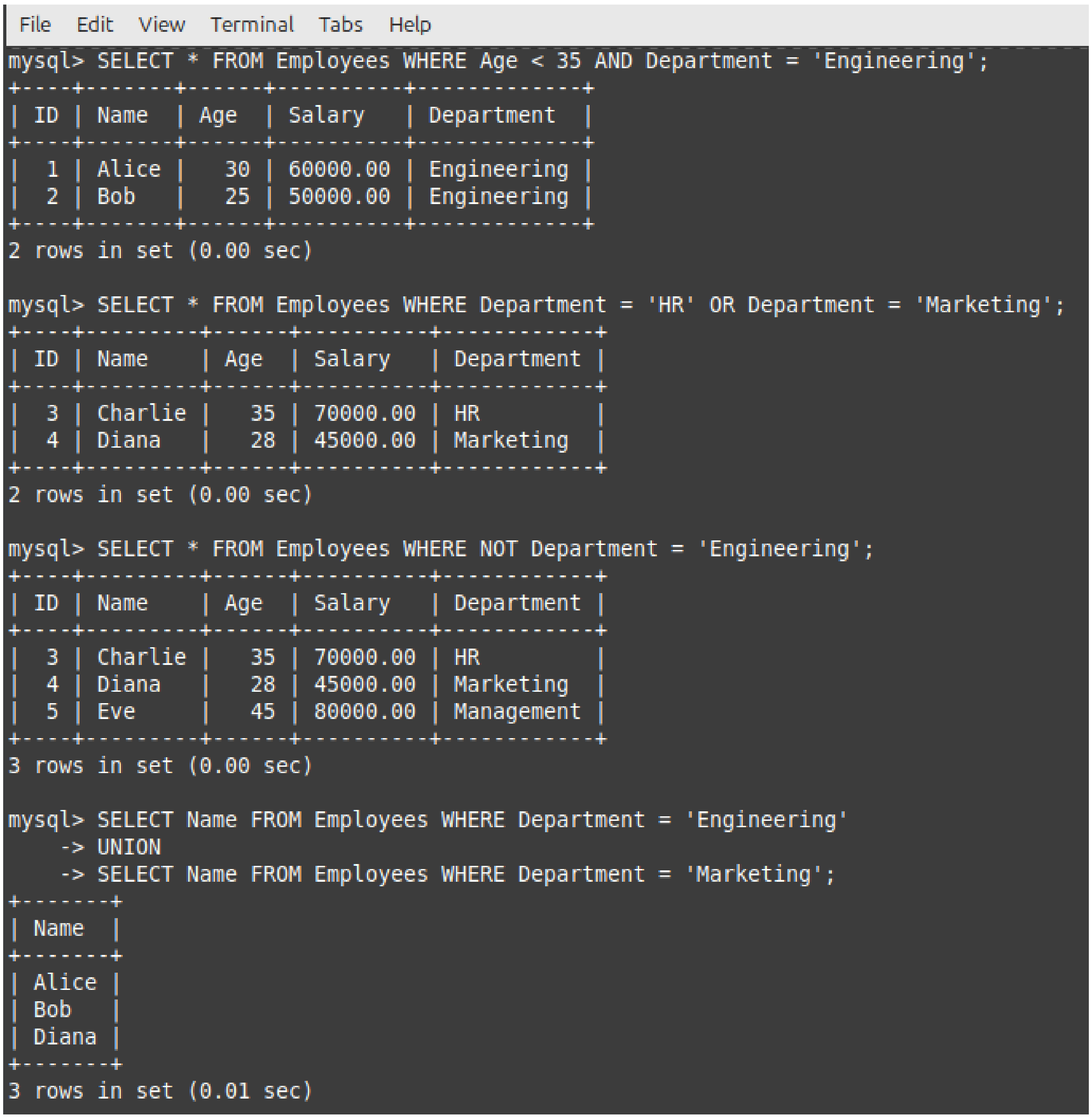
Roll no: 55   Batch: A3



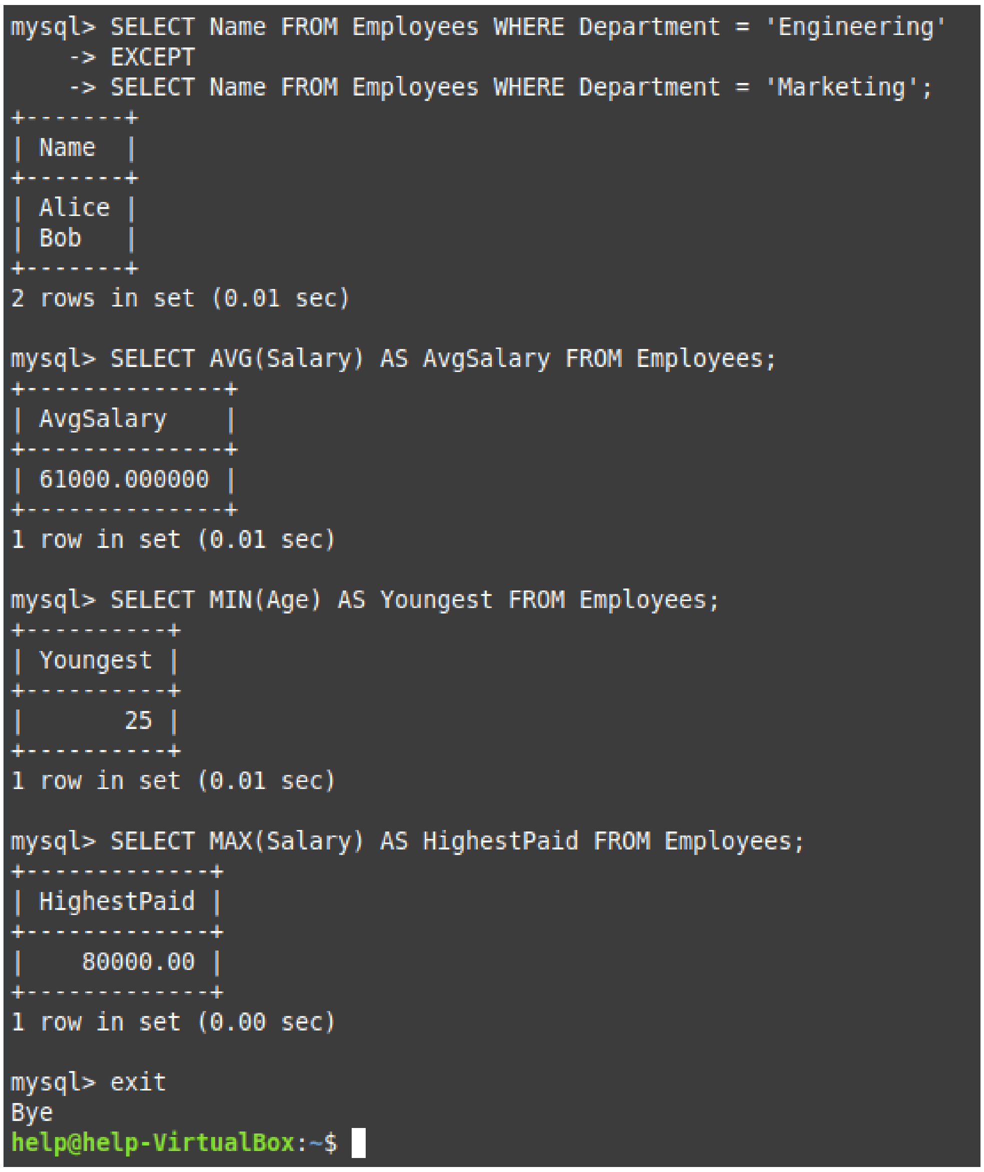
```
mysql> CREATE TABLE Employees (
    ->     ID INT PRIMARY KEY,
    ->     Name VARCHAR(100),
    ->     Age INT,
    ->     Salary DECIMAL(10, 2),
    ->     Department VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> INSERT INTO Employees (ID, Name, Age, Salary, Department) VALUES
    -> (1, 'Alice', 30, 60000.00, 'Engineering'),
    -> (2, 'Bob', 25, 50000.00, 'Engineering'),
    -> (3, 'Charlie', 35, 70000.00, 'HR'),
    -> (4, 'Diana', 28, 45000.00, 'Marketing'),
    -> (5, 'Eve', 45, 80000.00, 'Management');
Query OK, 5 rows affected (0.03 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> select * Employees;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Emp
loyees' at line 1
mysql> select *from  Employees;
+----+---------+------+----------+-------------+
| ID | Name    | Age  | Salary   | Department  |
+----+---------+------+----------+-------------+
|  1 | Alice   |   30 | 60000.00 | Engineering |
|  2 | Bob     |   25 | 50000.00 | Engineering |
|  3 | Charlie |   35 | 70000.00 | HR          |
|  4 | Diana   |   28 | 45000.00 | Marketing   |
|  5 | Eve     |   45 | 80000.00 | Management  |
+----+---------+------+----------+-------------+
5 rows in set (0.00 sec)
```



```
mysql> SELECT Name, Salary, Salary + 5000 AS NewSalary FROM Employees;
+---------+----------+-----------+
| Name    | Salary   | NewSalary |
+---------+----------+-----------+
| Alice   | 60000.00 |  65000.00 |
| Bob     | 50000.00 |  55000.00 |
| Charlie | 70000.00 |  75000.00 |
| Diana   | 45000.00 |  50000.00 |
| Eve     | 80000.00 |  85000.00 |
+---------+----------+-----------+
5 rows in set (0.00 sec)

mysql> SELECT Name, Salary, Salary - 60000 AS SalaryDiff FROM Employees;
+---------+----------+------------+
| Name    | Salary   | SalaryDiff |
+---------+----------+------------+
| Alice   | 60000.00 |       0.00 |
| Bob     | 50000.00 |  -10000.00 |
| Charlie | 70000.00 |   10000.00 |
| Diana   | 45000.00 |  -15000.00 |
| Eve     | 80000.00 |   20000.00 |
+---------+----------+------------+
5 rows in set (0.00 sec)

mysql> select *from  Employees;
+----+---------+------+----------+-------------+
| ID | Name    | Age  | Salary   | Department  |
+----+---------+------+----------+-------------+
|  1 | Alice   |   30 | 60000.00 | Engineering |
|  2 | Bob     |   25 | 50000.00 | Engineering |
|  3 | Charlie |   35 | 70000.00 | HR          |
|  4 | Diana   |   28 | 45000.00 | Marketing   |
|  5 | Eve     |   45 | 80000.00 | Management  |
+----+---------+------+----------+-------------+
5 rows in set (0.00 sec)
```



```
mysql> SELECT * FROM Employees WHERE Age < 35 AND Department = 'Engineering';
+----+-------+------+----------+-------------+
| ID | Name  | Age  | Salary   | Department  |
+----+-------+------+----------+-------------+
|  1 | Alice |   30 | 60000.00 | Engineering |
|  2 | Bob   |   25 | 50000.00 | Engineering |
+----+-------+------+----------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Employees WHERE Department = 'HR' OR Department = 'Marketing';
+----+---------+------+----------+------------+
| ID | Name    | Age  | Salary   | Department |
+----+---------+------+----------+------------+
|  3 | Charlie |   35 | 70000.00 | HR         |
|  4 | Diana   |   28 | 45000.00 | Marketing  |
+----+---------+------+----------+------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM Employees WHERE NOT Department = 'Engineering';
+----+---------+------+----------+------------+
| ID | Name    | Age  | Salary   | Department |
+----+---------+------+----------+------------+
|  3 | Charlie |   35 | 70000.00 | HR         |
|  4 | Diana   |   28 | 45000.00 | Marketing  |
|  5 | Eve     |   45 | 80000.00 | Management |
+----+---------+------+----------+------------+
3 rows in set (0.00 sec)

mysql> SELECT Name FROM Employees WHERE Department = 'Engineering'
    -> UNION
    -> SELECT Name FROM Employees WHERE Department = 'Marketing';
+-------+
| Name  |
+-------+
| Alice |
| Bob   |
| Diana |
+-------+
3 rows in set (0.01 sec)
```



```
mysql> SELECT Name FROM Employees WHERE Department = 'Engineering'
    -> EXCEPT
    -> SELECT Name FROM Employees WHERE Department = 'Marketing';
+-------+
| Name  |
+-------+
| Alice |
| Bob   |
+-------+
2 rows in set (0.01 sec)

mysql> SELECT AVG(Salary) AS AvgSalary FROM Employees;
+--------------+
| AvgSalary    |
+--------------+
| 61000.000000 |
+--------------+
1 row in set (0.01 sec)

mysql> SELECT MIN(Age) AS Youngest FROM Employees;
+----------+
| Youngest |
+----------+
|       25 |
+----------+
1 row in set (0.01 sec)

mysql> SELECT MAX(Salary) AS HighestPaid FROM Employees;
+-------------+
| HighestPaid |
+-------------+
|    80000.00 |
+-------------+
1 row in set (0.00 sec)

mysql> exit
Bye
help@help-VirtualBox:~$
```

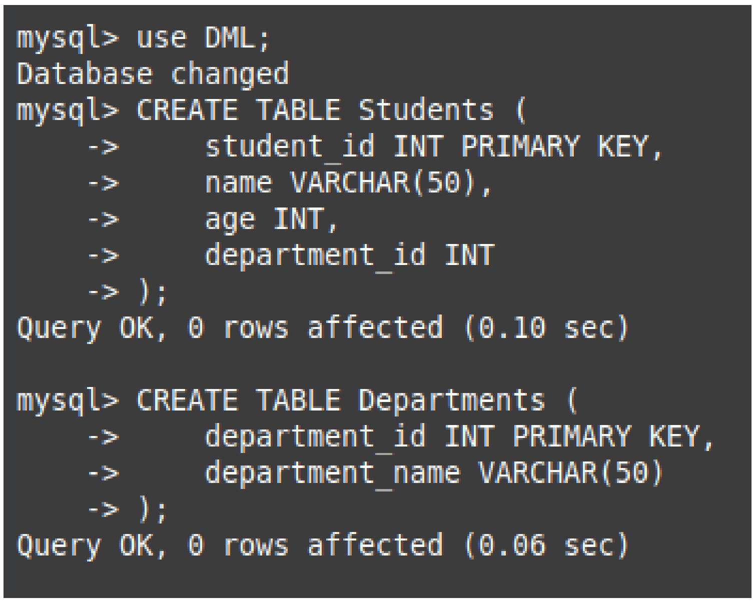
Name: Kale Prathmesh

Class: TE COMP (A)   Subject: DBMSL

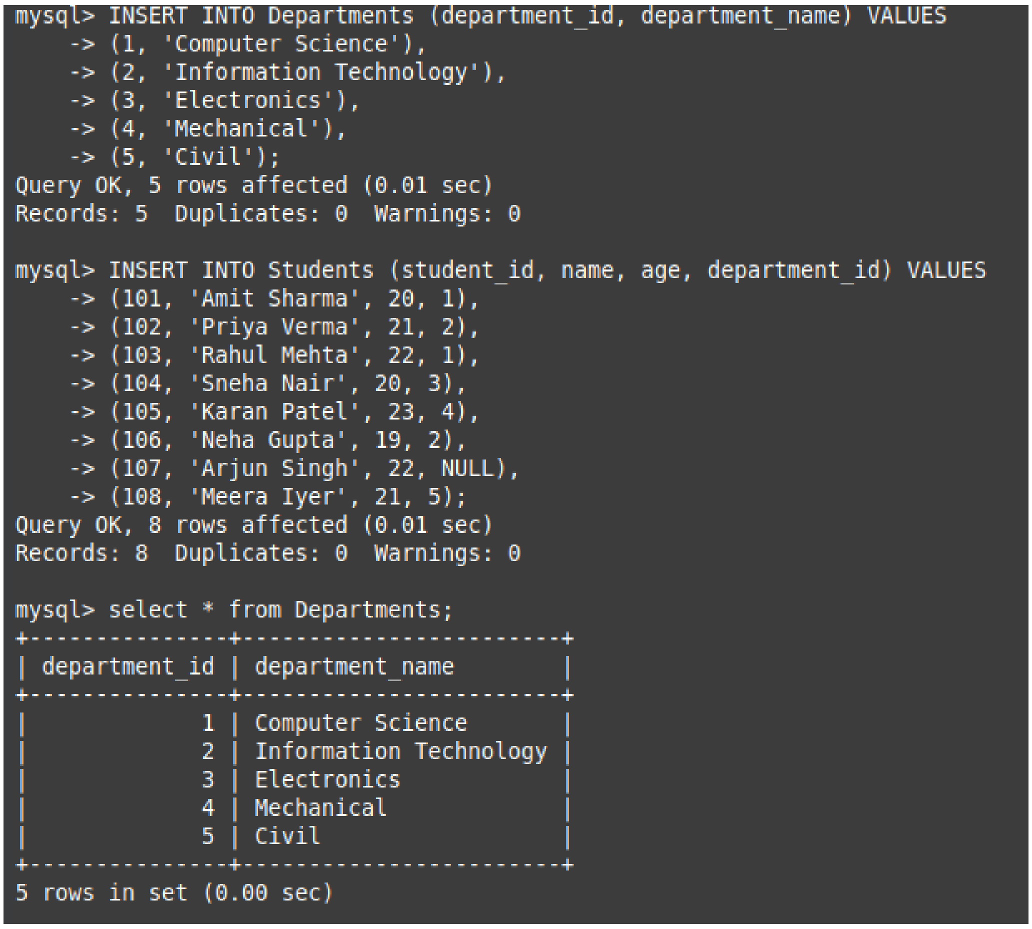
Roll no.: 55

PRACTICAL NO.: 3

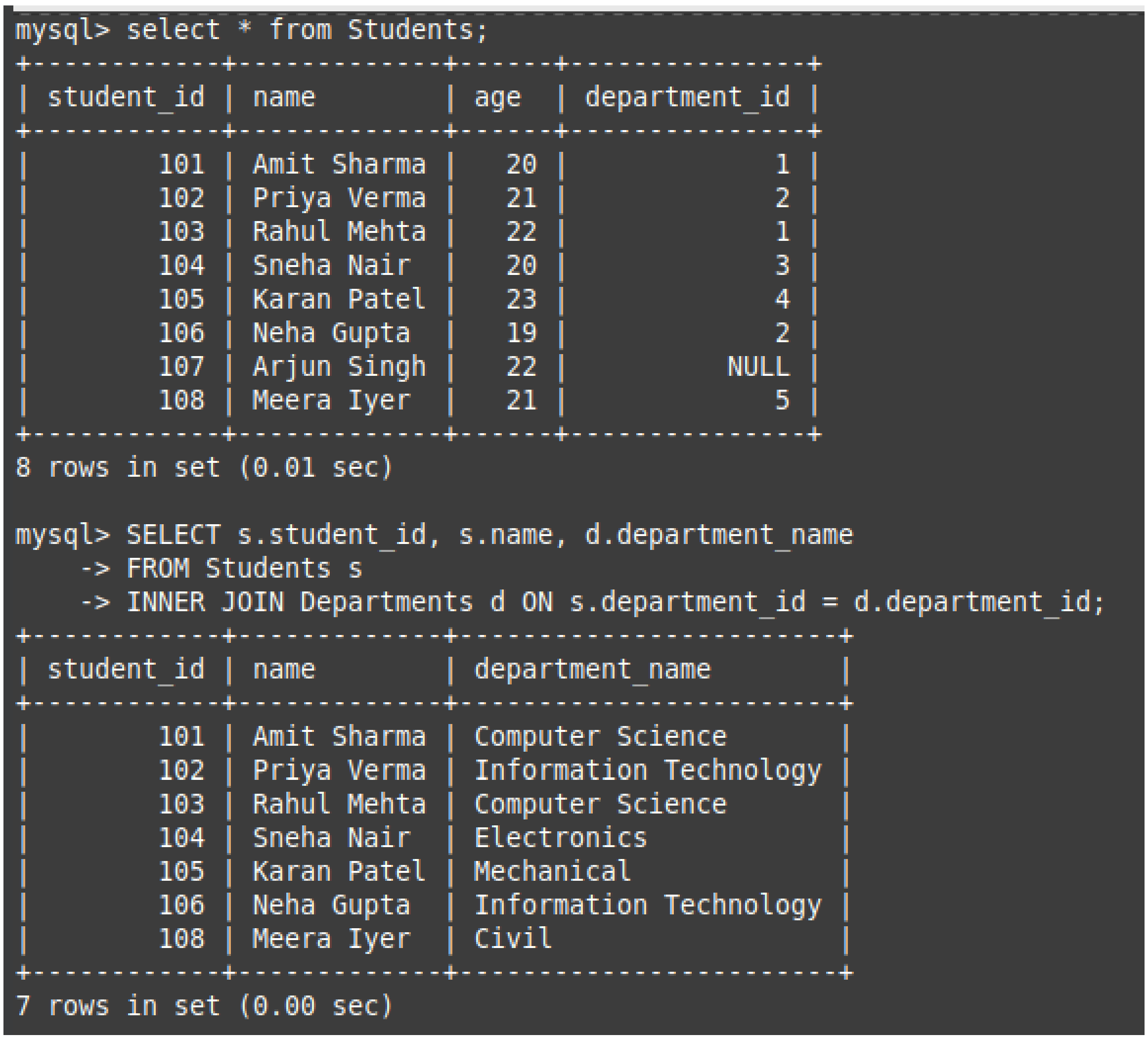
Output:



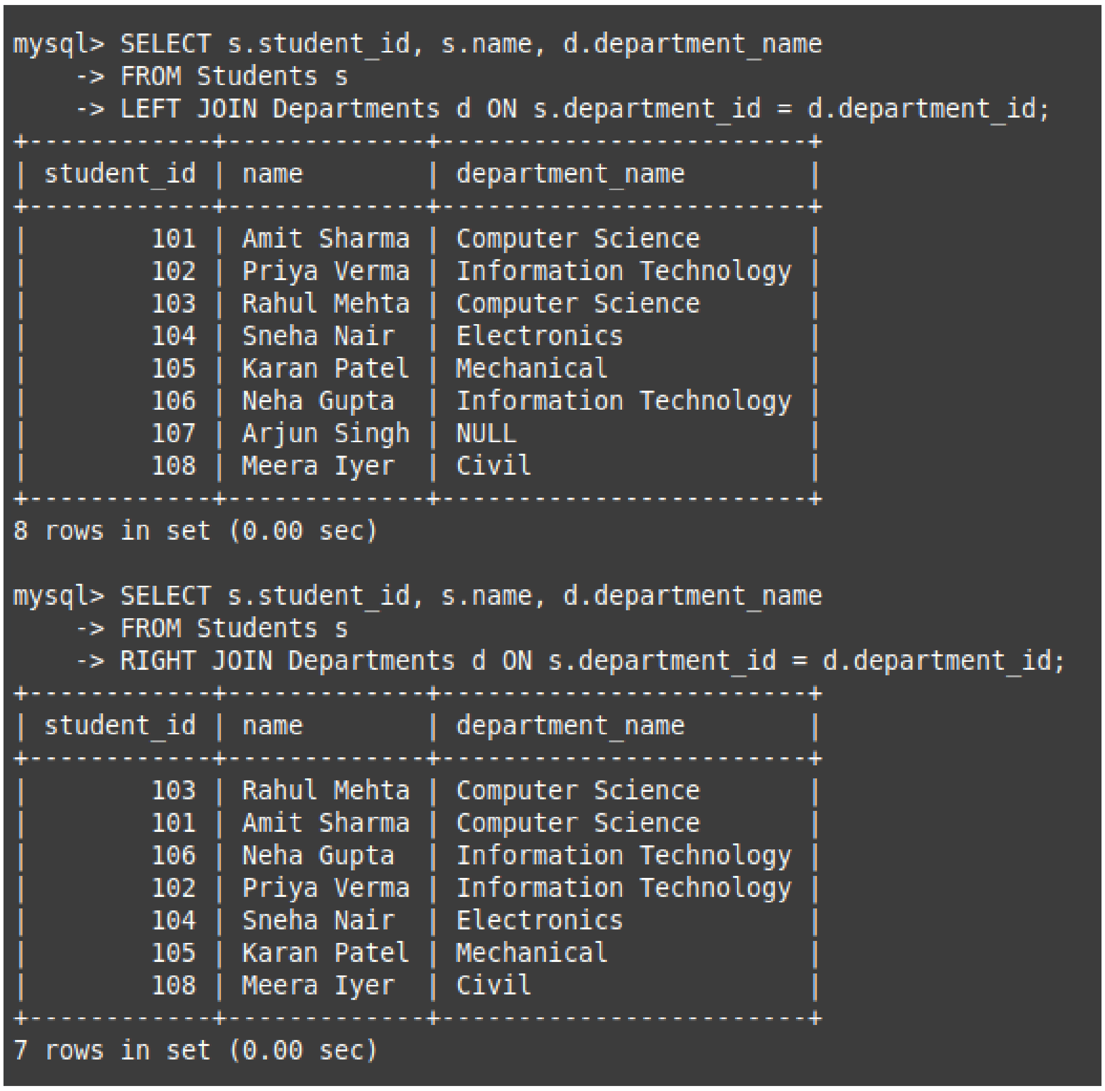
```
mysql> use DML;
Database changed
mysql> CREATE TABLE Students (
    ->     student_id INT PRIMARY KEY,
    ->     name VARCHAR(50),
    ->     age INT,
    ->     department_id INT
    -> );
Query OK, 0 rows affected (0.10 sec)

mysql> CREATE TABLE Departments (
    ->     department_id INT PRIMARY KEY,
    ->     department_name VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.06 sec)
```



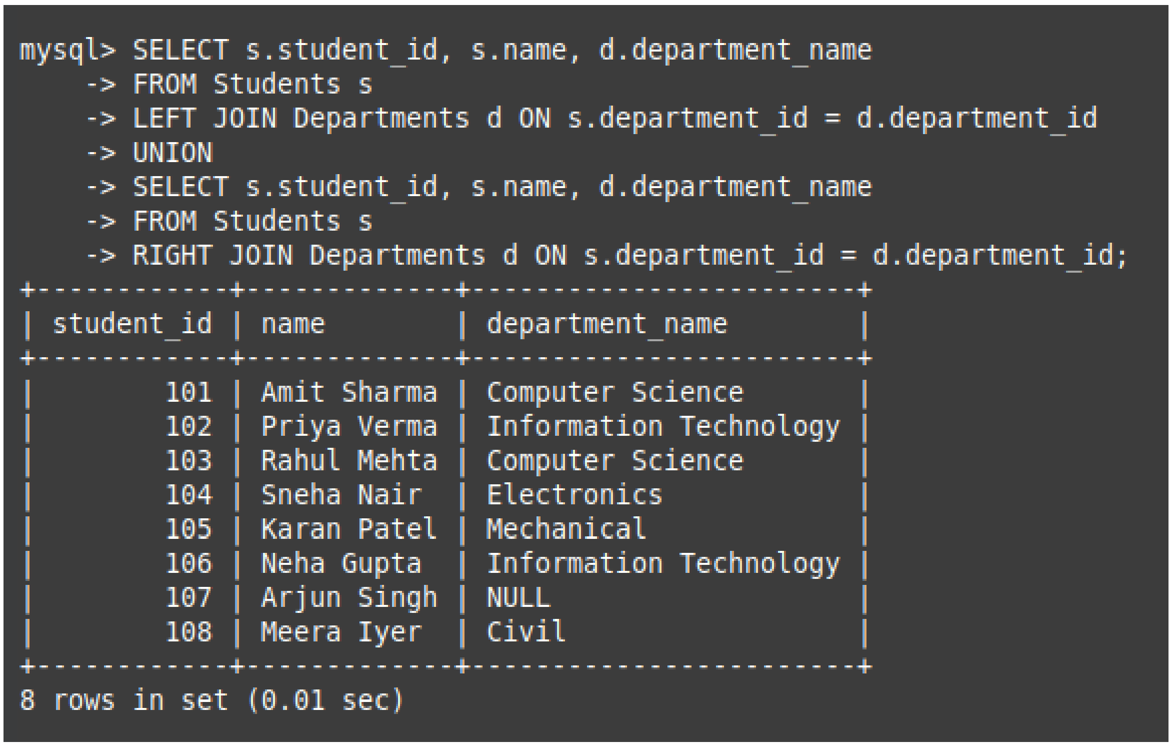
```
mysql> INSERT INTO Departments (department_id, department_name) VALUES
    -> (1, 'Computer Science'),
    -> (2, 'Information Technology'),
    -> (3, 'Electronics'),
    -> (4, 'Mechanical'),
    -> (5, 'Civil');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Students (student_id, name, age, department_id) VALUES
    -> (101, 'Amit Sharma', 20, 1),
    -> (102, 'Priya Verma', 21, 2),
    -> (103, 'Rahul Mehta', 22, 1),
    -> (104, 'Sneha Nair', 20, 3),
    -> (105, 'Karan Patel', 23, 4),
    -> (106, 'Neha Gupta', 19, 2),
    -> (107, 'Arjun Singh', 22, NULL),
    -> (108, 'Meera Iyer', 21, 5);
Query OK, 8 rows affected (0.01 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> select * from Departments;
+---------------+------------------------+
| department_id | department_name        |
+---------------+------------------------+
|             1 | Computer Science       |
|             2 | Information Technology |
|             3 | Electronics            |
|             4 | Mechanical             |
|             5 | Civil                  |
+---------------+------------------------+
5 rows in set (0.00 sec)
```



```
mysql> select * from Students;
+------------+-------------+------+---------------+
| student_id | name        | age  | department_id |
+------------+-------------+------+---------------+
|        101 | Amit Sharma |   20 |             1 |
|        102 | Priya Verma |   21 |             2 |
|        103 | Rahul Mehta |   22 |             1 |
|        104 | Sneha Nair  |   20 |             3 |
|        105 | Karan Patel |   23 |             4 |
|        106 | Neha Gupta  |   19 |             2 |
|        107 | Arjun Singh |   22 |          NULL |
|        108 | Meera Iyer  |   21 |             5 |
+------------+-------------+------+---------------+
8 rows in set (0.01 sec)

mysql> SELECT s.student_id, s.name, d.department_name
    -> FROM Students s
    -> INNER JOIN Departments d ON s.department_id = d.department_id;
+------------+-------------+------------------------+
| student_id | name        | department_name        |
+------------+-------------+------------------------+
|        101 | Amit Sharma | Computer Science       |
|        102 | Priya Verma | Information Technology |
|        103 | Rahul Mehta | Computer Science       |
|        104 | Sneha Nair  | Electronics            |
|        105 | Karan Patel | Mechanical             |
|        106 | Neha Gupta  | Information Technology |
|        108 | Meera Iyer  | Civil                  |
+------------+-------------+------------------------+
7 rows in set (0.00 sec)
```



```
mysql> SELECT s.student_id, s.name, d.department_name
    -> FROM Students s
    -> LEFT JOIN Departments d ON s.department_id = d.department_id;
+------------+-------------+------------------------+
| student_id | name        | department_name        |
+------------+-------------+------------------------+
|        101 | Amit Sharma | Computer Science       |
|        102 | Priya Verma | Information Technology |
|        103 | Rahul Mehta | Computer Science       |
|        104 | Sneha Nair  | Electronics            |
|        105 | Karan Patel | Mechanical             |
|        106 | Neha Gupta  | Information Technology |
|        107 | Arjun Singh | NULL                   |
|        108 | Meera Iyer  | Civil                  |
+------------+-------------+------------------------+
8 rows in set (0.00 sec)

mysql> SELECT s.student_id, s.name, d.department_name
    -> FROM Students s
    -> RIGHT JOIN Departments d ON s.department_id = d.department_id;
+------------+-------------+------------------------+
| student_id | name        | department_name        |
+------------+-------------+------------------------+
|        103 | Rahul Mehta | Computer Science       |
|        101 | Amit Sharma | Computer Science       |
|        106 | Neha Gupta  | Information Technology |
|        102 | Priya Verma | Information Technology |
|        104 | Sneha Nair  | Electronics            |
|        105 | Karan Patel | Mechanical             |
|        108 | Meera Iyer  | Civil                  |
+------------+-------------+------------------------+
7 rows in set (0.00 sec)
```



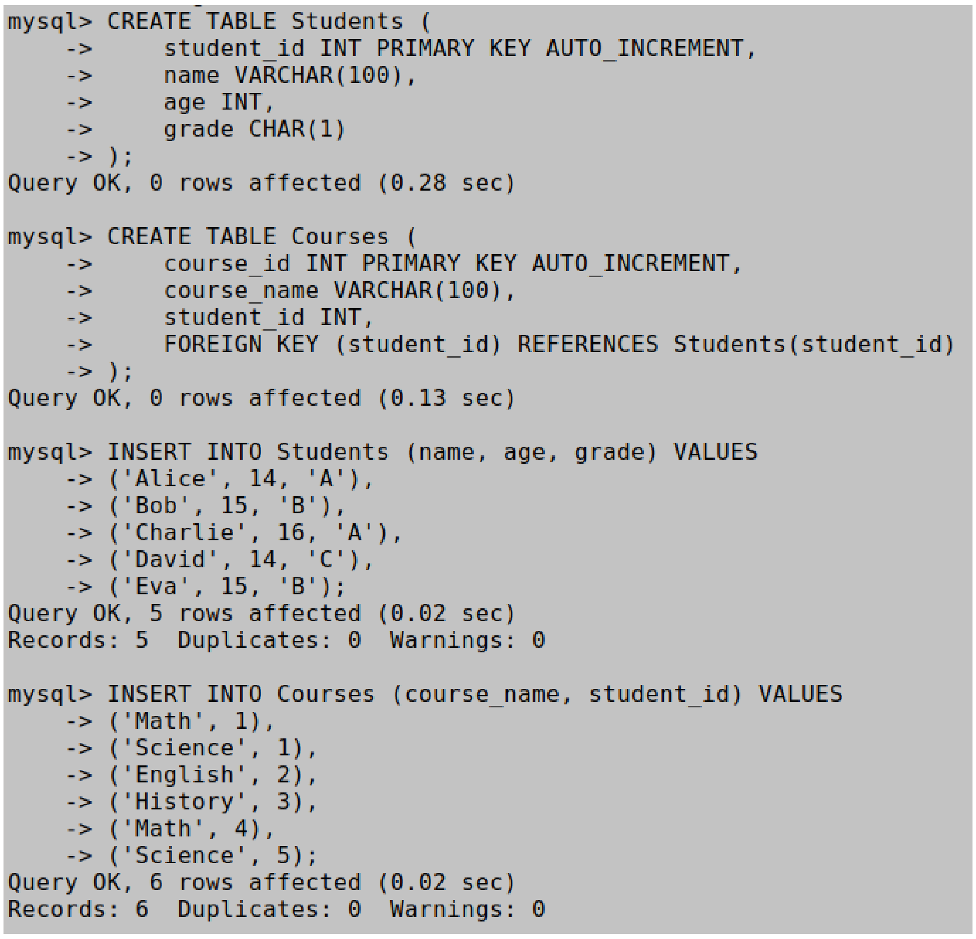
```
mysql> SELECT s.student_id, s.name, d.department_name
    -> FROM Students s
    -> LEFT JOIN Departments d ON s.department_id = d.department_id
    -> UNION
    -> SELECT s.student_id, s.name, d.department_name
    -> FROM Students s
    -> RIGHT JOIN Departments d ON s.department_id = d.department_id;
+------------+-------------+------------------------+
| student_id | name        | department_name        |
+------------+-------------+------------------------+
|        101 | Amit Sharma | Computer Science       |
|        102 | Priya Verma | Information Technology |
|        103 | Rahul Mehta | Computer Science       |
|        104 | Sneha Nair  | Electronics            |
|        105 | Karan Patel | Mechanical             |
|        106 | Neha Gupta  | Information Technology |
|        107 | Arjun Singh | NULL                   |
|        108 | Meera Iyer  | Civil                  |
+------------+-------------+------------------------+
8 rows in set (0.01 sec)
```

Name: Kale Prathmesh

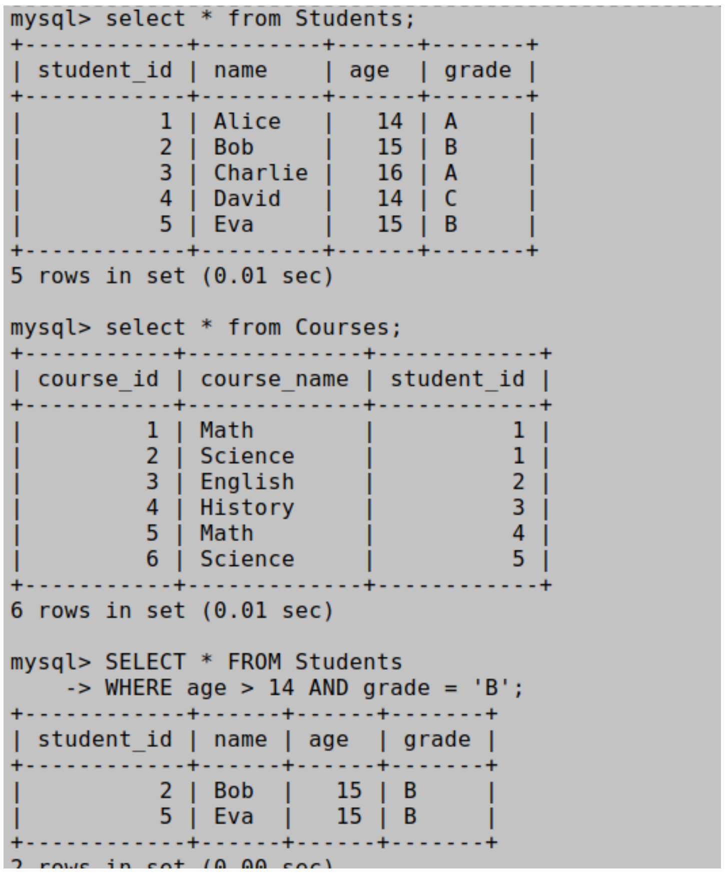
Class: TE COMP (A)   Subject: DBMSL

Roll no.: 55

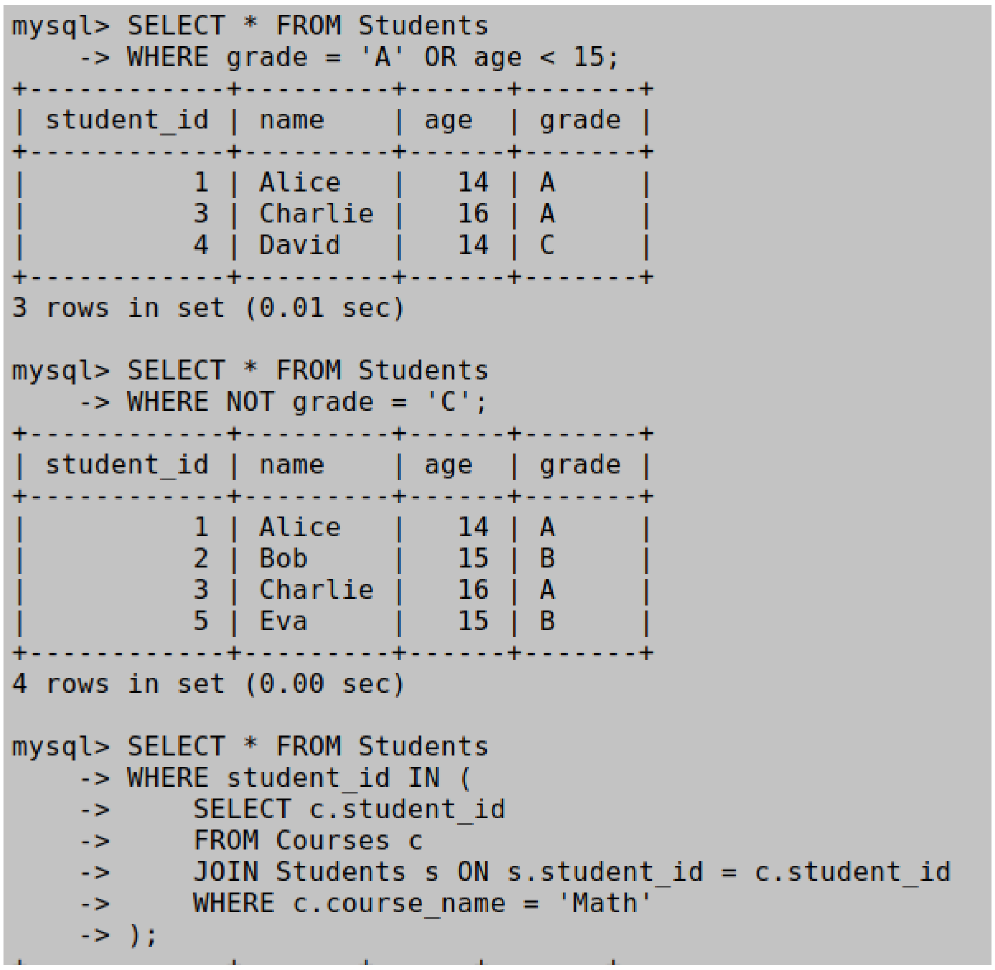
PRACTICAL NO.: 3_[2]



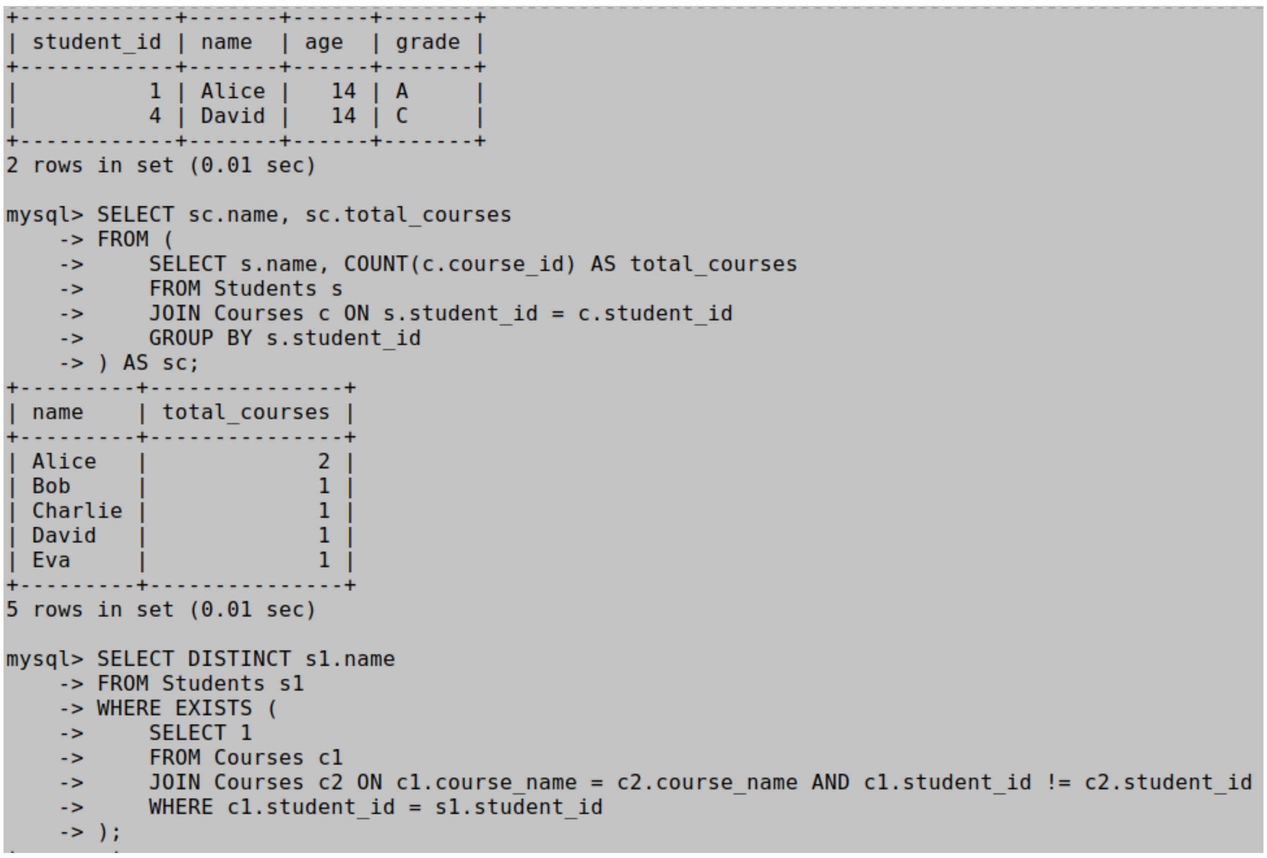
```
mysql> CREATE TABLE Students (
    ->     student_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     name VARCHAR(100),
    ->     age INT,
    ->     grade CHAR(1)
    -> );
Query OK, 0 rows affected (0.28 sec)

mysql> CREATE TABLE Courses (
    ->     course_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     course_name VARCHAR(100),
    ->     student_id INT,
    ->     FOREIGN KEY (student_id) REFERENCES Students(student_id)
    -> );
Query OK, 0 rows affected (0.13 sec)

mysql> INSERT INTO Students (name, age, grade) VALUES
    -> ('Alice', 14, 'A'),
    -> ('Bob', 15, 'B'),
    -> ('Charlie', 16, 'A'),
    -> ('David', 14, 'C'),
    -> ('Eva', 15, 'B');
Query OK, 5 rows affected (0.02 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql> INSERT INTO Courses (course_name, student_id) VALUES
    -> ('Math', 1),
    -> ('Science', 1),
    -> ('English', 2),
    -> ('History', 3),
    -> ('Math', 4),
    -> ('Science', 5);
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0
```



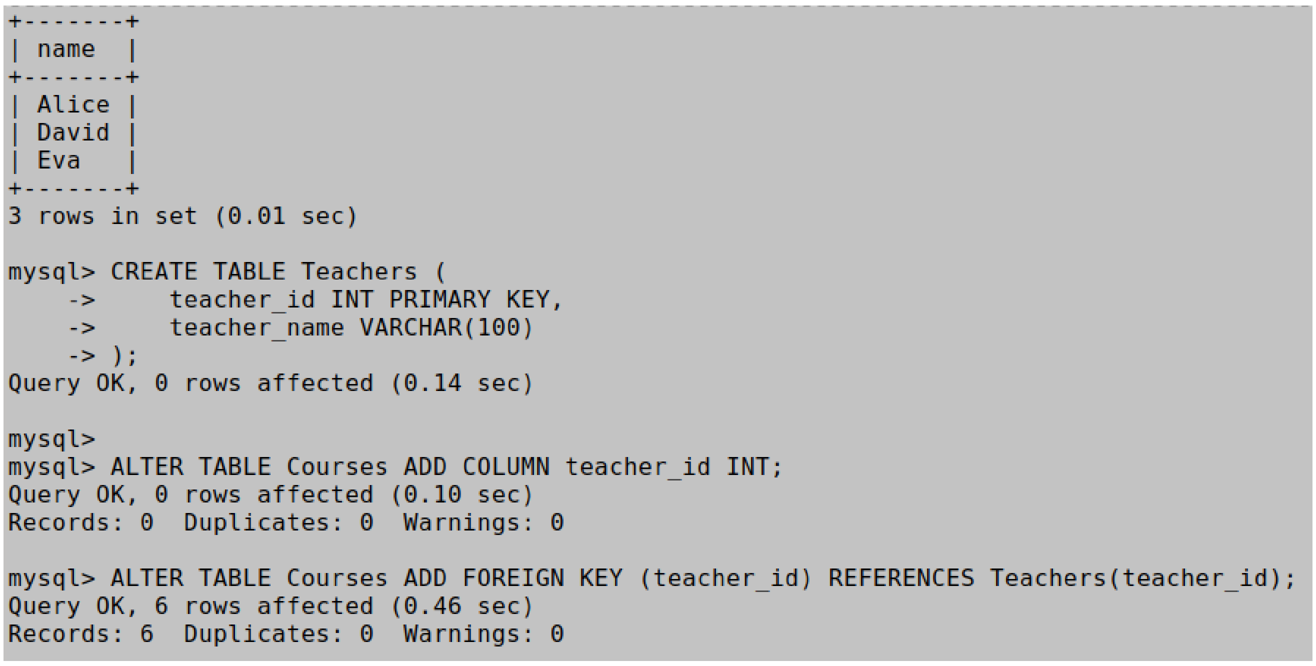
```
mysql> select * from Students;
+------------+---------+------+-------+
| student_id | name    | age  | grade |
+------------+---------+------+-------+
|          1 | Alice   |   14 | A     |
|          2 | Bob     |   15 | B     |
|          3 | Charlie |   16 | A     |
|          4 | David   |   14 | C     |
|          5 | Eva     |   15 | B     |
+------------+---------+------+-------+
5 rows in set (0.01 sec)

mysql> select * from Courses;
+-----------+-------------+------------+
| course_id | course_name | student_id |
+-----------+-------------+------------+
|         1 | Math        |          1 |
|         2 | Science     |          1 |
|         3 | English     |          2 |
|         4 | History     |          3 |
|         5 | Math        |          4 |
|         6 | Science     |          5 |
+-----------+-------------+------------+
6 rows in set (0.01 sec)

mysql> SELECT * FROM Students
    -> WHERE age > 14 AND grade = 'B';
+------------+------+------+-------+
| student_id | name | age  | grade |
+------------+------+------+-------+
|          2 | Bob  |   15 | B     |
|          5 | Eva  |   15 | B     |
+------------+------+------+-------+
2 rows in set (0.00 sec)
```



```
mysql> SELECT * FROM Students
    -> WHERE grade = 'A' OR age < 15;
+------------+---------+------+-------+
| student_id | name    | age  | grade |
+------------+---------+------+-------+
|          1 | Alice   |   14 | A     |
|          3 | Charlie |   16 | A     |
|          4 | David   |   14 | C     |
+------------+---------+------+-------+
3 rows in set (0.01 sec)

mysql> SELECT * FROM Students
    -> WHERE NOT grade = 'C';
+------------+---------+------+-------+
| student_id | name    | age  | grade |
+------------+---------+------+-------+
|          1 | Alice   |   14 | A     |
|          2 | Bob     |   15 | B     |
|          3 | Charlie |   16 | A     |
|          5 | Eva     |   15 | B     |
+------------+---------+------+-------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM Students
    -> WHERE student_id IN (
    ->     SELECT c.student_id
    ->     FROM Courses c
    ->     JOIN Students s ON s.student_id = c.student_id
    ->     WHERE c.course_name = 'Math'
    -> );
```



```
+------------+-------+------+-------+
| student_id | name  | age  | grade |
+------------+-------+------+-------+
|          1 | Alice |   14 | A     |
|          4 | David |   14 | C     |
+------------+-------+------+-------+
2 rows in set (0.01 sec)

mysql> SELECT sc.name, sc.total_courses
    -> FROM (
    ->     SELECT s.name, COUNT(c.course_id) AS total_courses
    ->     FROM Students s
    ->     JOIN Courses c ON s.student_id = c.student_id
    ->     GROUP BY s.student_id
    -> ) AS sc;
+---------+---------------+
| name    | total_courses |
+---------+---------------+
| Alice   |             2 |
| Bob     |             1 |
| Charlie |             1 |
| David   |             1 |
| Eva     |             1 |
+---------+---------------+
5 rows in set (0.01 sec)

mysql> SELECT DISTINCT s1.name
    -> FROM Students s1
    -> WHERE EXISTS (
    ->     SELECT 1
    ->     FROM Courses c1
    ->     JOIN Courses c2 ON c1.course_name = c2.course_name AND c1.student_id != c2.student_id
    ->     WHERE c1.student_id = s1.student_id
    -> );
```



```
+-------+
| name  |
+-------+
| Alice |
| David |
| Eva   |
+-------+
3 rows in set (0.01 sec)

mysql> CREATE TABLE Teachers (
    ->     teacher_id INT PRIMARY KEY,
    ->     teacher_name VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.14 sec)

mysql>
mysql> ALTER TABLE Courses ADD COLUMN teacher_id INT;
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE Courses ADD FOREIGN KEY (teacher_id) REFERENCES Teachers(teacher_id);
Query OK, 6 rows affected (0.46 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM Students s      WHERE EXISTS (          SELECT 1          FROM Courses c          JOIN Teachers t ON c.te
acher_id = t.teacher_id          WHERE c.student_id = s.student_id AND t.teacher_name = 'Mr. Smith'      );
+------------+-------+------+-------+
| student_id | name  | age  | grade |
+------------+-------+------+-------+
|          1 | Alice |   14 | A     |
+------------+-------+------+-------+
1 row in set (0.01 sec)
```

Name: Kale Prathmesh

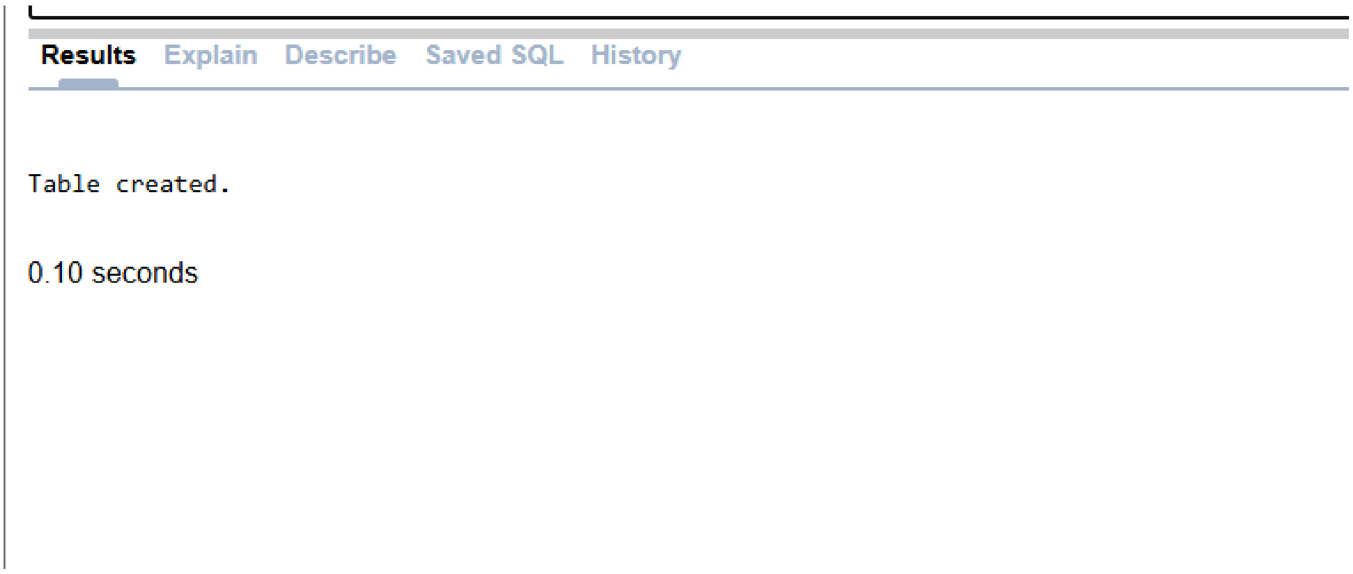
Batch: A3 **Roll no.:** 55

Subject: DBMSL

Practical no.: 4

Output:

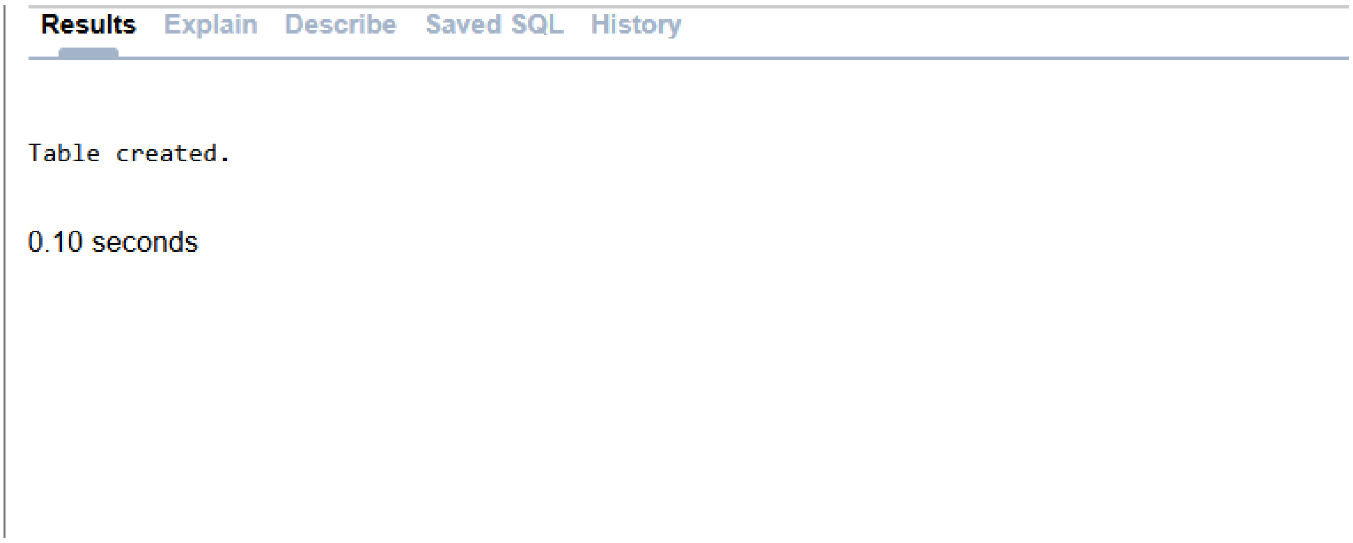
- CREATE TABLE borrower(roll_no NUMBER , name VARCHAR2(25), dateofissue DATE,name_of_book VARCHAR2(25), status VARCHAR2(20));



```
Table created.
```

0.10 seconds

- CREATE TABLE fine(roll_no NUMBER,date_of_return DATE,amt NUMBER);



```
Table created.
```

0.10 seconds

- INSERT INTO borrower VALUES(1, ’ASHUTOSH’, TO_DATE(‘01-08-2025’, ‘DD-MM-YYYY’), ’HARRY POTTER’, ’Issued’);

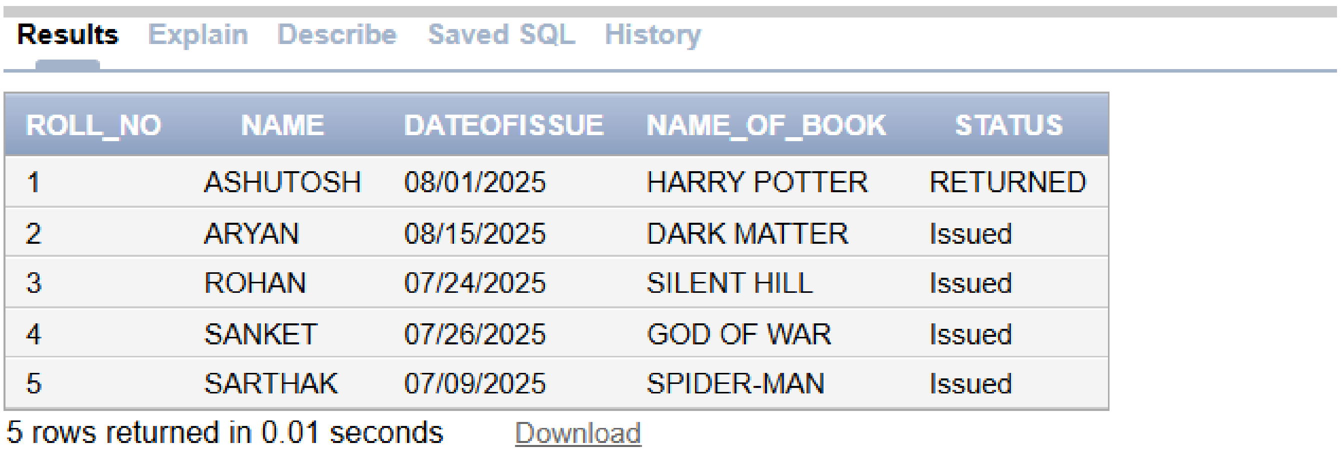
- INSERT INTO borrower VALUES(2, ’ARYAN’, TO_DATE(‘15-08-2025’ , ’DD-MM-YYYY’),’DARK MATTER’, ’Issued’);

- INSERT INTO borrower VALUES(3,’ROHAN’, TO_DATE(‘24-07-2025’, ‘DD-MM-YYYY’), ’SILENT HILL’, ’Issued’);

- INSERT INTO borrower VALUES(4, ’SANKET’, TO_DATE(‘26-07-2025’, ‘DD-MM-YYYY’),’GOD OF WAR’, ‘Issued’);

- INSERT INTO borrower VALUES(5, ’SARTHAK‘ , TO_DATE(‘09-07-2025’, ’DD-MM-YYYY’),’SPIDER-MAN’, ’Issued’);

- SELECT * from borrower;

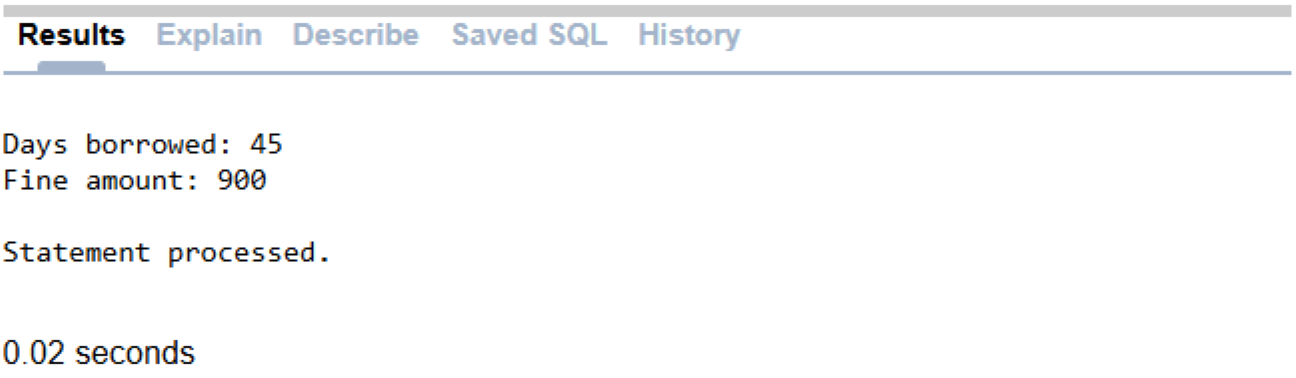


ROLL_NO	NAME	DATEOFISSUE	NAME_OF_BOOK	STATUS
1	ASHUTOSH	08/01/2025	HARRY POTTER	RETURNED
2	ARYAN	08/15/2025	DARK MATTER	Issued
3	ROHAN	07/24/2025	SILENT HILL	Issued
4	SANKET	07/26/2025	GOD OF WAR	Issued
5	SARTHAK	07/09/2025	SPIDER-MAN	Issued

5 rows returned in 0.01 seconds Download

- DECLARE
  i_roll_no NUMBER;
  name_of_book VARCHAR2(25);
  no_of_days NUMBER;
  return_date DATE := TO_DATE(SYSDATE, ‘DD-MM-YYYY’);
  temp NUMBER;
  doi DATE;
  fine NUMBER;
  BEGIN
  i_roll_no := &i_roll_no;

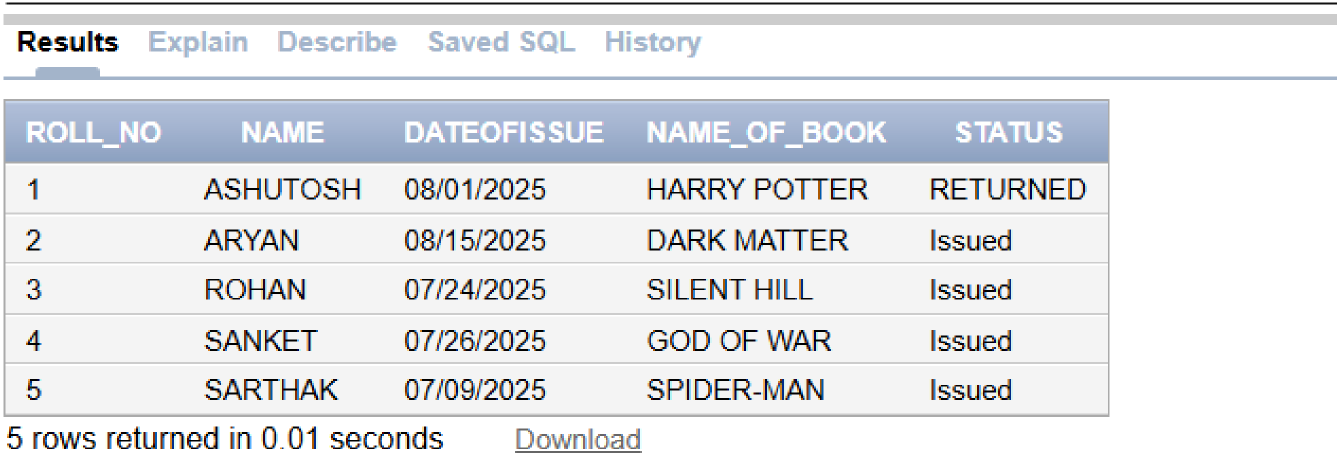
```
name_of_book := '&nameofbook';
--dbms_output.put_line(return_date);
SELECT to_date (borrower.dateofissue, 'DD-MM-YYYY') INTO doi FROM borrower
WHERE borrower.roll_no = i_roll_no AND borrower.name_of_book = name_of_book;
no_of_days := return_date-doi;
dbms_output.put_line(no_of_days);
IF (no_of_days >15 AND no_of_days <=30) THEN
fine := 5*no_of_days;
ELSIF (no_of_days >30 ) THEN
temp := no_of_days-30;
fine := 150 + temp*50;
END IF;
dbms_output.put_line(fine);
INSERT INTO fine VALUES(i_roll_no,return_date,fine);
UPDATE borrower SET status = &#39;RETURNED&#39; WHERE borrower.roll_no = i_roll_no;
END;
/
```



```
Days borrowed: 45
Fine amount: 900

Statement processed.
```

0.02 seconds

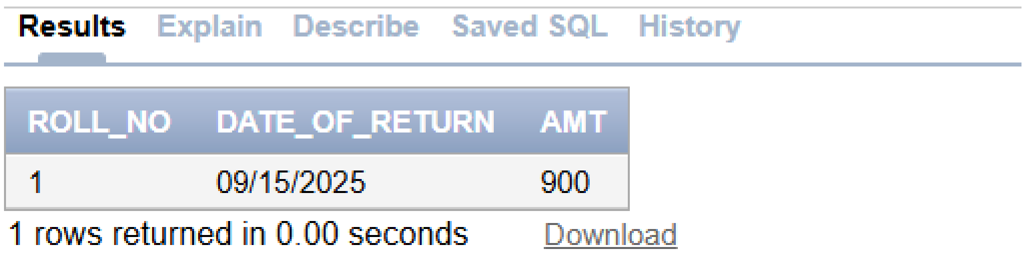
- SELECT * FROM borrower;



ROLL_NO	NAME	DATEOFISSUE	NAME_OF_BOOK	STATUS
1	ASHUTOSH	08/01/2025	HARRY POTTER	RETURNED
2	ARYAN	08/15/2025	DARK MATTER	Issued
3	ROHAN	07/24/2025	SILENT HILL	Issued
4	SANKET	07/26/2025	GOD OF WAR	Issued
5	SARTHAK	07/09/2025	SPIDER-MAN	Issued

5 rows returned in 0.01 seconds

- SELECT * FROM fine;



ROLL_NO	DATE_OF_RETURN	AMT
1	09/15/2025	900

1 rows returned in 0.00 seconds

Name: Kale Prathmesh

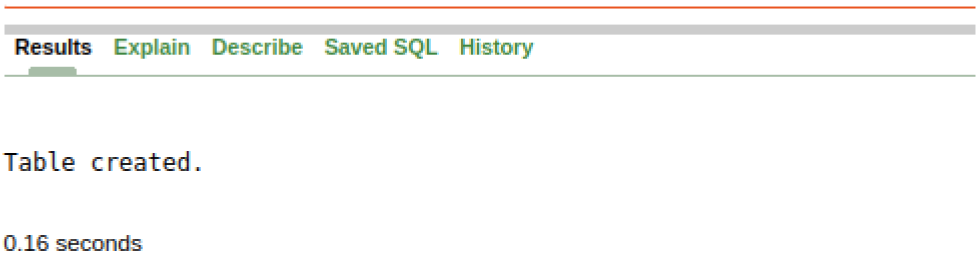
Batch: A3 **Roll no.:** 55

Subject: DBMSL

Practical no.: 5

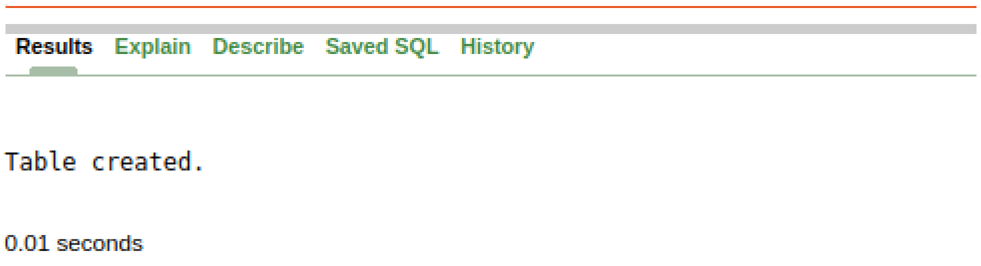
Output:

- CREATE TABLE stud_marks (name VARCHAR2(25),total_marks NUMBER);



```
Table created.

0.16 seconds
```

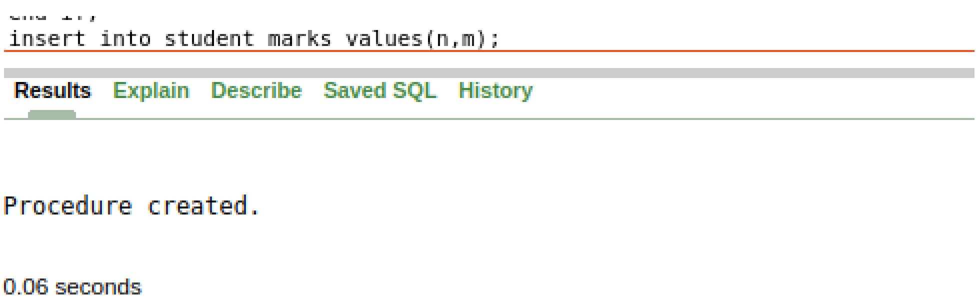
- CREATE TABLE result (roll NUMBER, name VARCHAR2(25), class VARCHAR2(30));



```
Table created.

0.01 seconds
```

- CREATE OR REPLACE PROCEDURE proc_Grade (
  r IN NUMBER,
  n IN VARCHAR2,
  m IN NUMBER,
  grade OUT VARCHAR2
) AS
  BEGIN
  IF m BETWEEN 990 AND 1500 THEN
  grade := ‘Distinction’;
  ELSIF m BETWEEN 900 AND 989 THEN
  grade := ‘First Class’;
  ELSIF m BETWEEN 825 AND 899 THEN
  grade := ‘Higher Second Class’;
  ELSE
  grade := ‘Fail’;
  END IF;

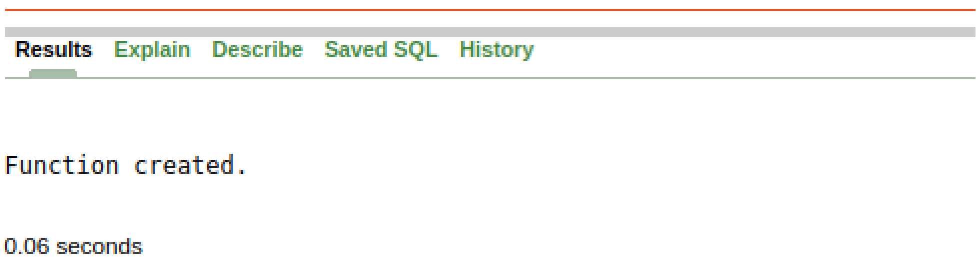
```
INSERT INTO stud_marks VALUES (n, m);
INSERT INTO result VALUES (r, n, grade);
END;
/
```



```
insert into student marks values(n,m);
```

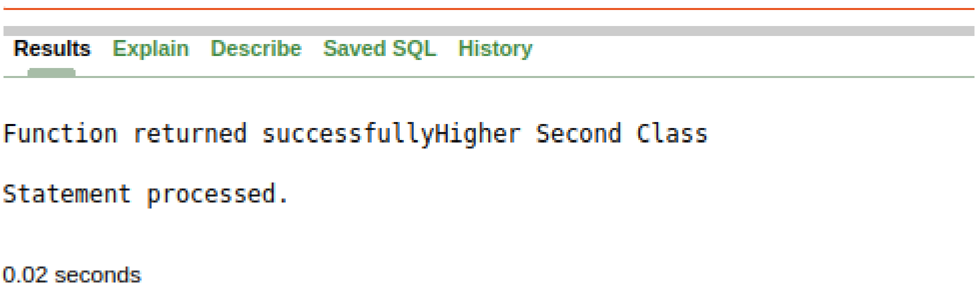
```
Procedure created.

0.06 seconds
```

-
  ```
  CREATE OR REPLACE FUNCTION func_Grade (
  r IN NUMBER,
  n IN VARCHAR2,
  m IN NUMBER )
  RETURN VARCHAR2 AS grade VARCHAR2(30);
  BEGIN
  proc_Grade(r, n, m, grade);
  RETURN grade;
  END;
  /
  ```



```
Function created.

0.06 seconds
```

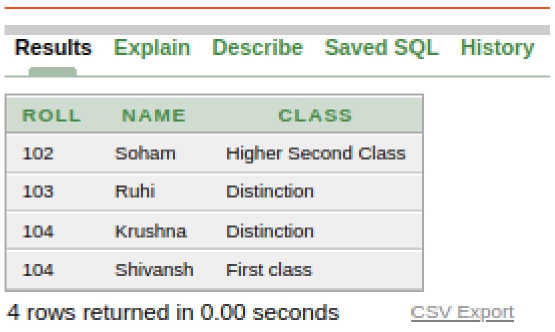
-
  ```
  DECLARE
  v_roll NUMBER := 102;
  v_name VARCHAR2(25) := &#39;Sai&#39;;
  v_marks NUMBER := 880;
  v_grade VARCHAR2(30);
  BEGIN
  v_grade := func_Grade(v_roll, v_name, v_marks);
  DBMS_OUTPUT.PUT_LINE(&#39;Function returned grade: &#39; || v_grade);
  COMMIT;
  END;
  /
  ```



```
Function returned successfullyHigher Second Class

Statement processed.
```

0.02 seconds

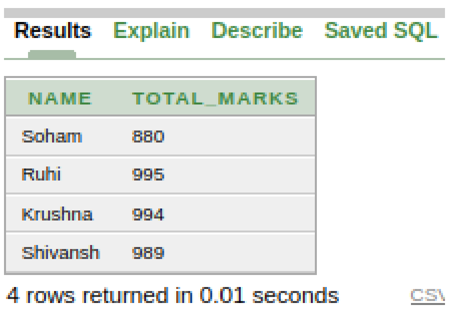
- SELECT * FROM stud_marks;



ROLL	NAME	CLASS
102	Soham	Higher Second Class
103	Ruhi	Distinction
104	Krushna	Distinction
104	Shivansh	First class

4 rows returned in 0.00 seconds CSV Export

- SELECT * FROM result;



NAME	TOTAL_MARKS
Soham	880
Ruhi	995
Krushna	994
Shivansh	989

4 rows returned in 0.01 seconds CSV

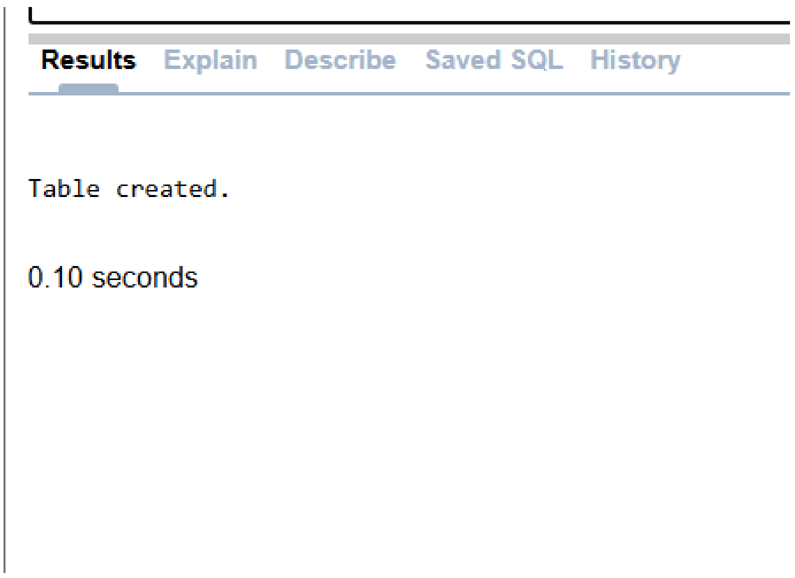
Name: Kale Prathmesh

Div.: A Batch: A3

Roll no.: 55 Subject: DBMSL

OUTPUT:

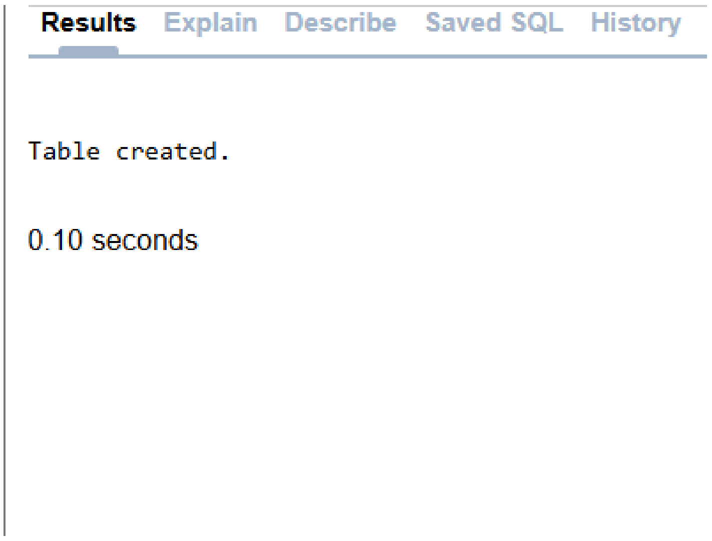
- CREATE TABLE o_rollcall(roll_no NUMBER, name VARCHAR2(25), div VARCHAR2(5));



```
Table created.
```

0.10 seconds

- CREATE TABLE n_rollcall(roll_no NUMBER, name VARCHAR2(25), div VARCHAR2(5));



```
Table created.
```

0.10 seconds

- INSERT INTO o_rollcall VALUES(1, ’ASHUTOSH’, ’A’);
- INSERT INTO o_rollcall VALUES(2, ’ARYAN’, ’A’);
- INSERT INTO o_rollcall VALUES(3, ‘SANKET’ ,’B’);

- INSERT INTO o_rollcall VALUES(1, ’ASHUTOSH’ ,’A’);
- INSERT INTO o_rollcall VALUES(5, ’ROHAN’, ’B’);
- INSERT INTO o_rollcall VALUES(1, ’ASHUTOSH’ ,’B’);

- CREATE OR REPLACE PROCEDURE cursor_imp AS
  c_r NUMBER;
  c_n VARCHAR2(25);
  c_d VARCHAR2(5);
  cursor c1 (roll NUMBER , n VARCHAR2 , d VARCHAR2) IS SELECT roll_no , COUNT (roll_no),name ,COUNT (name), div, COUNT (div) FROM o_rollcall GROUP BY

  roll_no,name,div HAVING (COUNT(roll_no)>1) AND (COUNT(name)>1) AND (COUNT(div)>1) ;

  temp c1%rowtype;

  BEGIN

  DELETE FROM n_rollcall;

  INSERT INTO n_rollcall SELECT * FROM o_rollcall;

  OPEN c1(c_r,c_n,c_d);

  LOOP

  FETCH c1 INTO temp;

  EXIT WHEN c1%NOTFOUND;

  DELETE FROM n_rollcall WHERE roll_no = temp.roll_no AND name =

  temp.name AND div = temp.div;

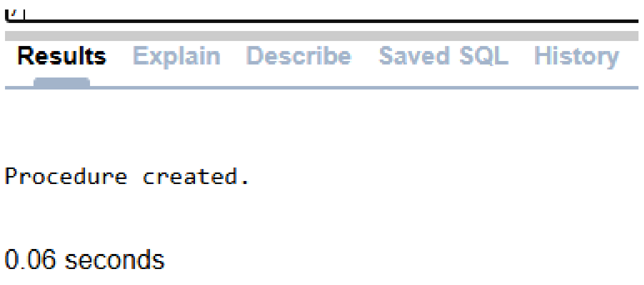
  INSERT INTO n_rollcall VALUES (temp.roll_no,temp.name,temp.div);

  dbms_output.put_line(temp.roll_no||temp.name||temp.div);

  END LOOP;

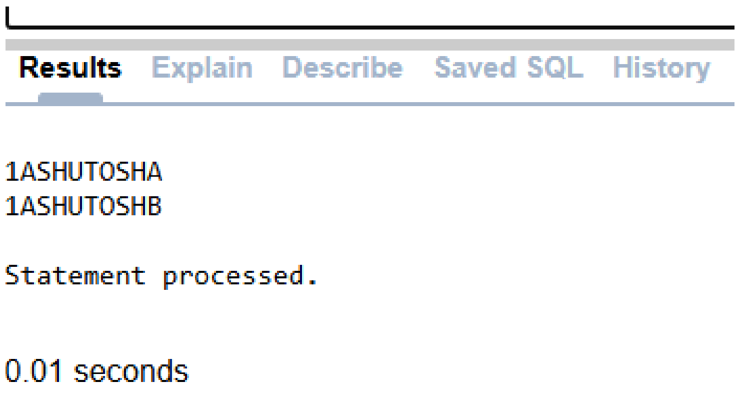
  END;

  /



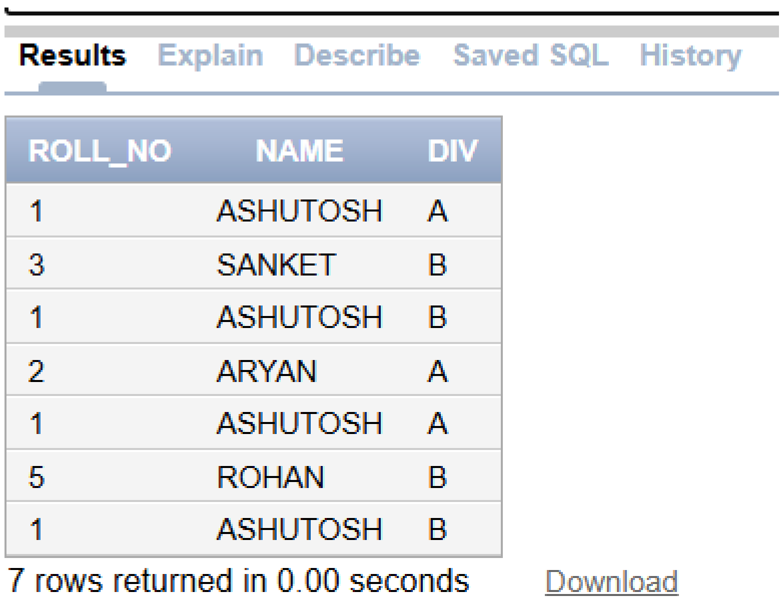
```
Procedure created.

0.06 seconds
```

- BEGIN
  cursor_imp;
  END;
  /



```
1ASHUTOSHA
1ASHUTOSHB

Statement processed.


0.01 seconds
```

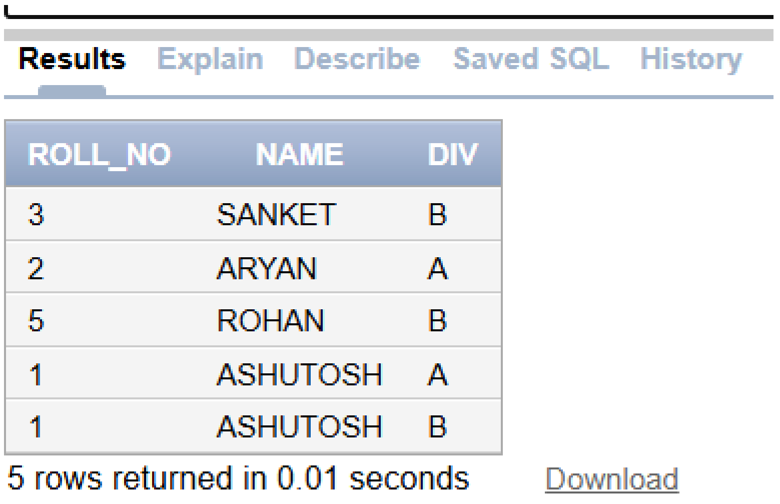
- select *from o_rollcall;



ROLL_NO	NAME	DIV
1	ASHUTOSH	A
3	SANKET	B
1	ASHUTOSH	B
2	ARYAN	A
1	ASHUTOSH	A
5	ROHAN	B
1	ASHUTOSH	B

7 rows returned in 0.00 seconds

- select *from n_rollcall;



ROLL_NO	NAME	DIV
3	SANKET	B
2	ARYAN	A
5	ROHAN	B
1	ASHUTOSH	A
1	ASHUTOSH	B

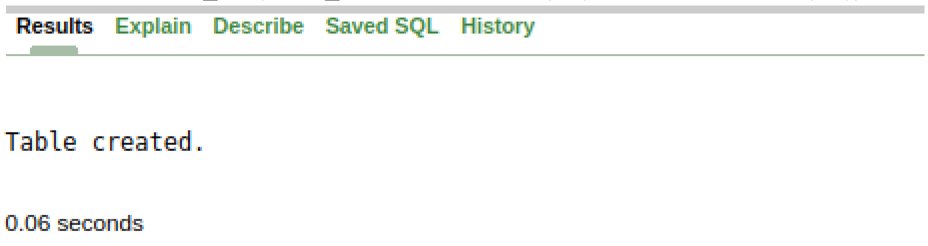
5 rows returned in 0.01 seconds

Name: Kale Prathmesh

Div.: A Batch: A3

Roll no.:55 Practical no.: 7

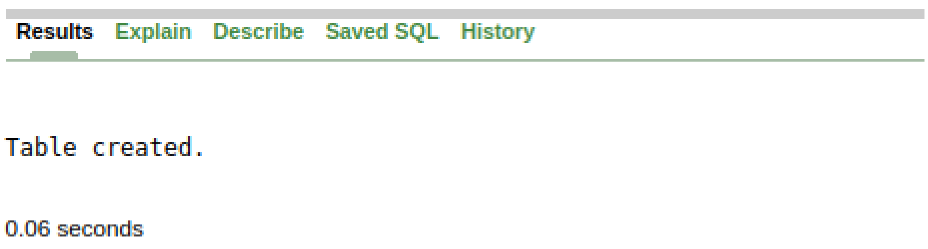
-
  ```
  CREATE TABLE lib_tab(book_name VARCHAR2(25),status VARCHAR2(15));
  ```



```
Table created.
```

0.06 seconds

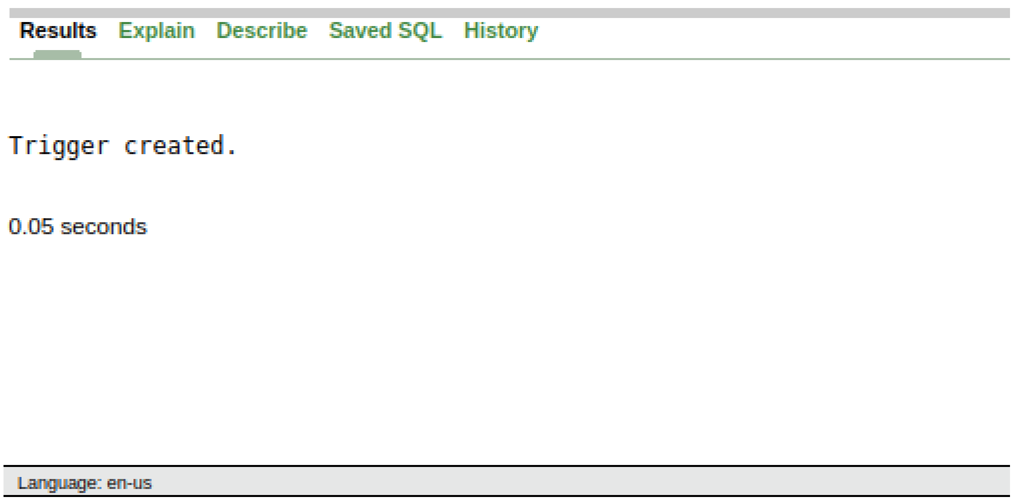
-
  ```
  CREATE TABLE library_audit(date_modified DATE, book_name
  VARCHAR2(25),old_status VARCHAR(15),new_status VARCHAR2(15),action
  VARCHAR2(25));
  ```



```
Table created.
```

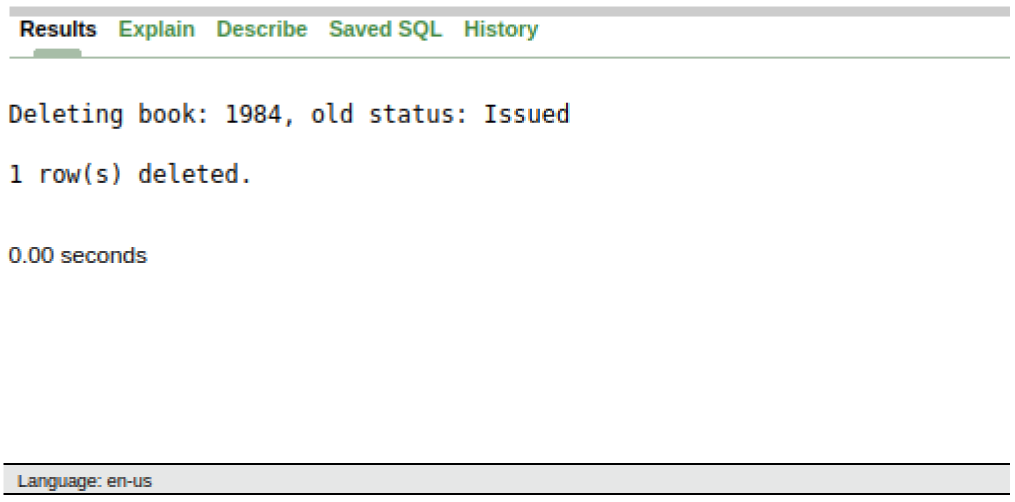
0.06 seconds

-
  ```
  CREATE OR REPLACE TRIGGER trigger_1
  AFTER INSERT OR UPDATE OR DELETE
  ON lib_tab
  FOR EACH ROW
  BEGIN
  IF UPDATING THEN
  DBMS_OUTPUT.PUT_LINE(:OLD.status);
  INSERT INTO library_audit
  VALUES (SYSDATE, :OLD.book_name, :OLD.status, :NEW.status, ‘UPDATE’);
  ELSIF INSERTING THEN
  DBMS_OUTPUT.PUT_LINE(:NEW.status);
  INSERT INTO library_audit
  VALUES (SYSDATE, :NEW.book_name, NULL, :NEW.status, ‘INSERT’);
  ELSIF DELETING THEN
  DBMS_OUTPUT.PUT_LINE(:OLD.book_name || ‘ deleting’);
  INSERT INTO library_audit
  VALUES (SYSDATE, :OLD.book_name, :OLD.status, NULL, ‘DELETE’);
  END IF;
  END;
  /
  ```



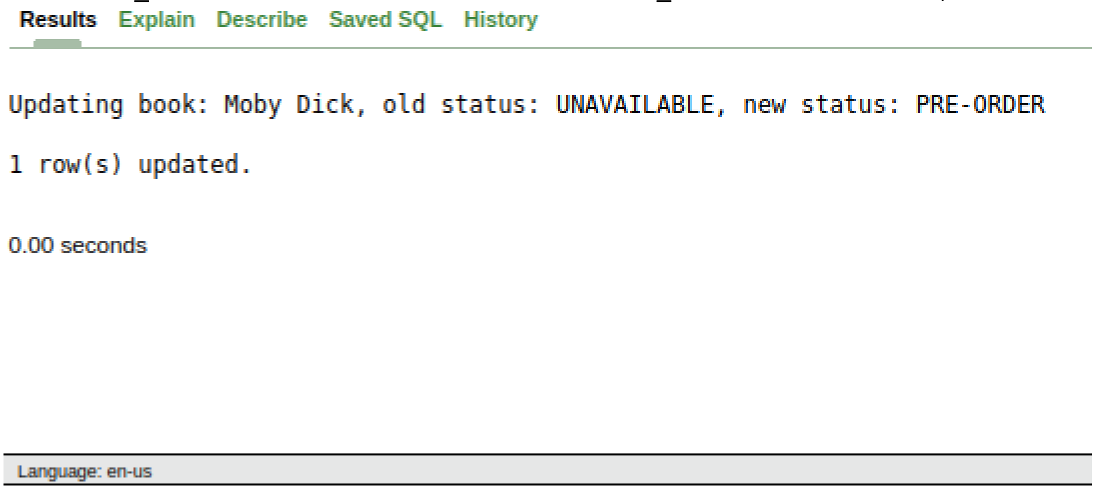
```
Trigger created.

0.05 seconds
```

- DELETE FROM lib_tab WHERE book_name = ‘1984’;



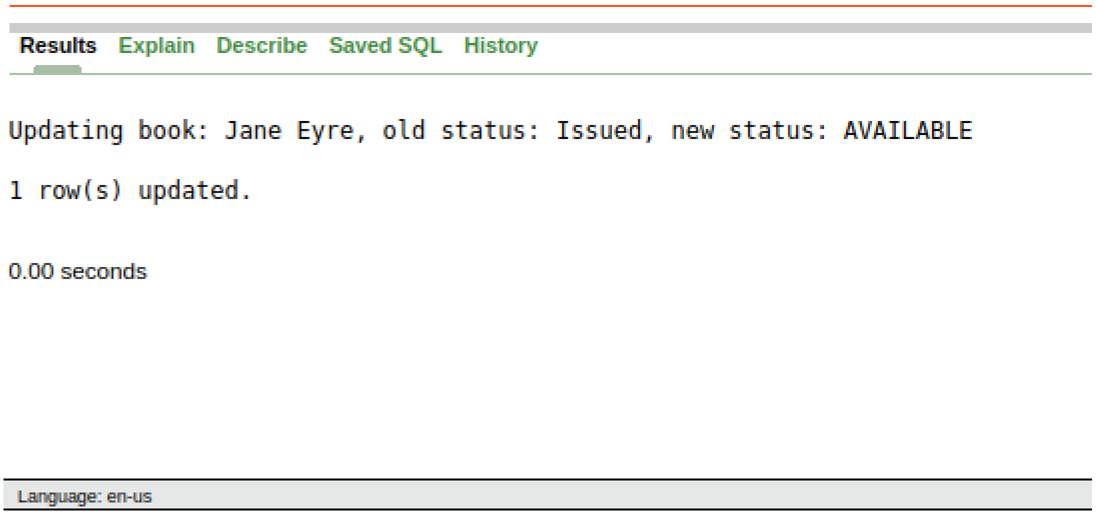
```
Deleting book: 1984, old status: Issued

1 row(s) deleted.

0.00 seconds
```

- UPDATE lib_tab SET status = ‘PRE-ORDER’ WHERE book_name = ‘GOD OF WAR’;



```
Updating book: Moby Dick, old status: UNAVAILABLE, new status: PRE-ORDER

1 row(s) updated.

0.00 seconds
```

- UPDATE lib_tab SET status = ‘AVAILABLE’ WHERE book_name = ‘Jane Eyre’;

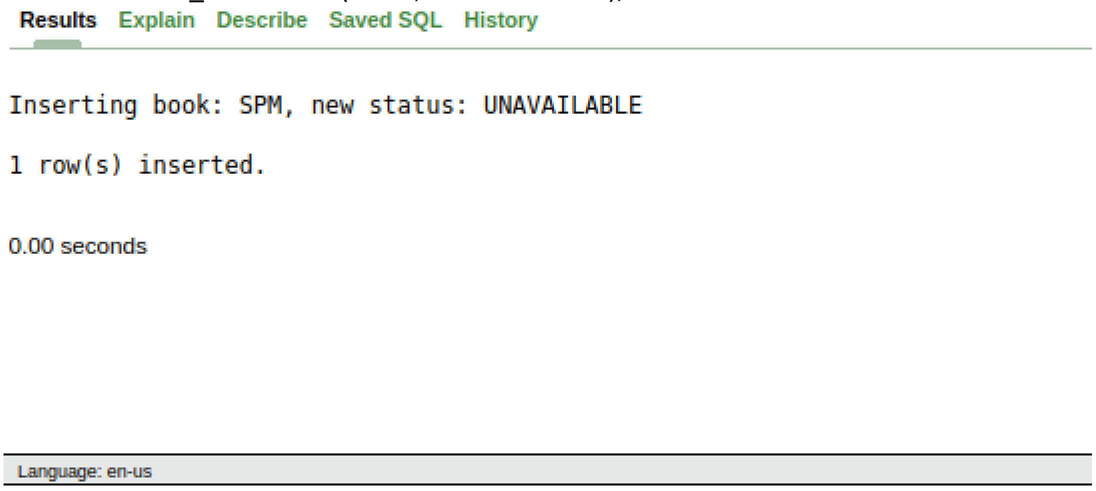


```
Updating book: Jane Eyre, old status: Issued, new status: AVAILABLE

1 row(s) updated.
```

0.00 seconds

Language: en-us

- INSERT INTO lib_tab VALUES(‘SPM’, ’UNAVAILABLE’);

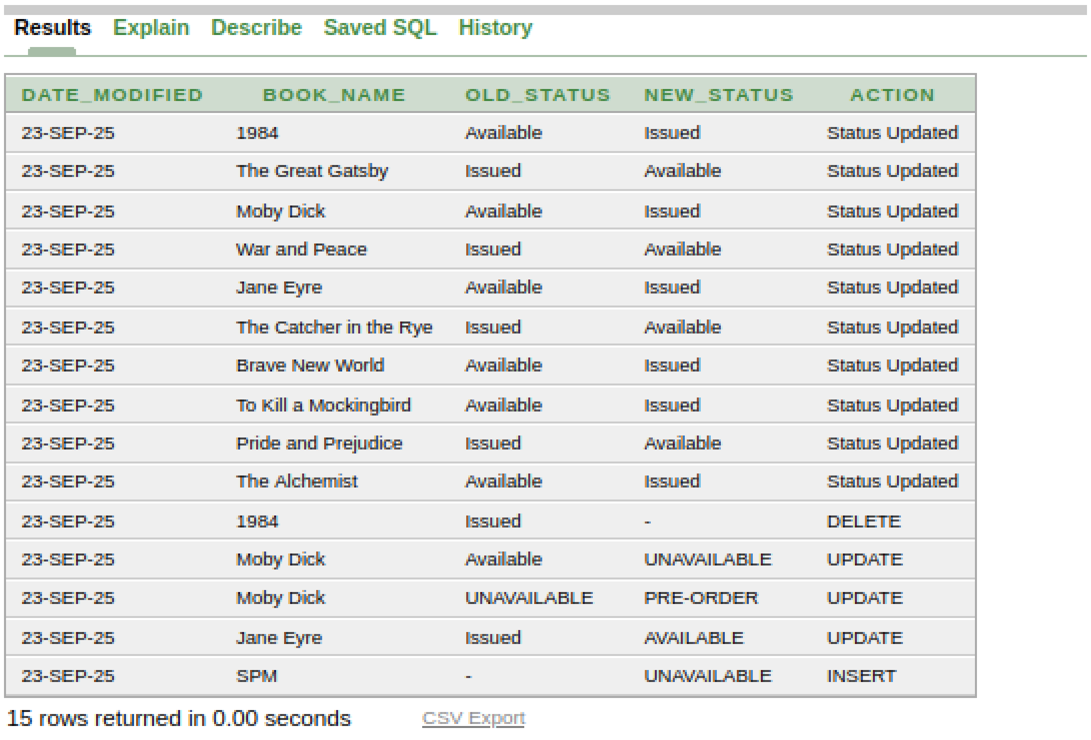


```
Inserting book: SPM, new status: UNAVAILABLE

1 row(s) inserted.
```

0.00 seconds

Language: en-us

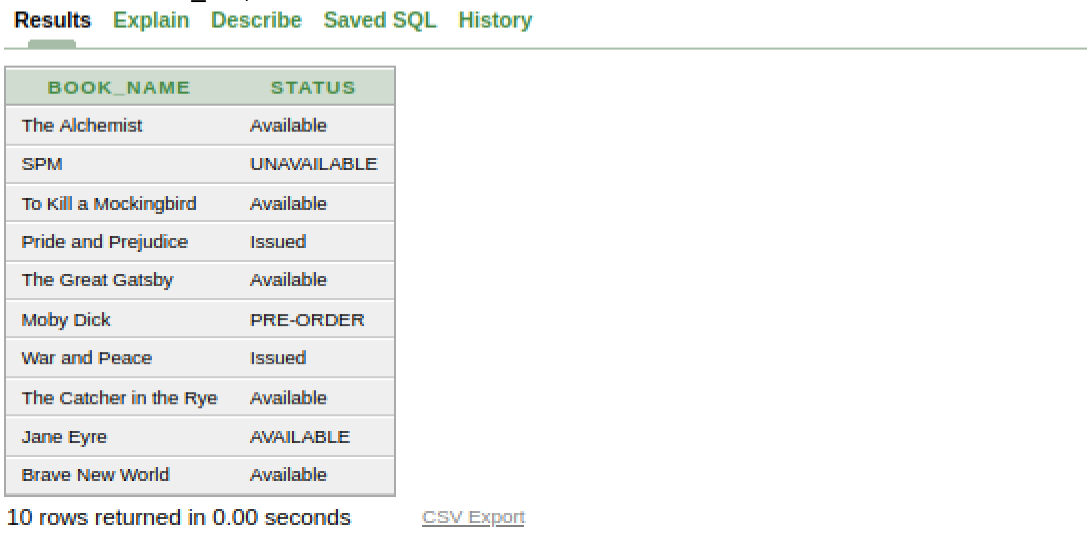
- Select * from library_audit;



DATE_MODIFIED	BOOK_NAME	OLD_STATUS	NEW_STATUS	ACTION
23-SEP-25	1984	Available	Issued	Status Updated
23-SEP-25	The Great Gatsby	Issued	Available	Status Updated
23-SEP-25	Moby Dick	Available	Issued	Status Updated
23-SEP-25	War and Peace	Issued	Available	Status Updated
23-SEP-25	Jane Eyre	Available	Issued	Status Updated
23-SEP-25	The Catcher in the Rye	Issued	Available	Status Updated
23-SEP-25	Brave New World	Available	Issued	Status Updated
23-SEP-25	To Kill a Mockingbird	Available	Issued	Status Updated
23-SEP-25	Pride and Prejudice	Issued	Available	Status Updated
23-SEP-25	The Alchemist	Available	Issued	Status Updated
23-SEP-25	1984	Issued	-	DELETE
23-SEP-25	Moby Dick	Available	UNAVAILABLE	UPDATE
23-SEP-25	Moby Dick	UNAVAILABLE	PRE-ORDER	UPDATE
23-SEP-25	Jane Eyre	Issued	AVAILABLE	UPDATE
23-SEP-25	SPM	-	UNAVAILABLE	INSERT

15 rows returned in 0.00 seconds

- Select * from lib_tab;



BOOK_NAME	STATUS
The Alchemist	Available
SPM	UNAVAILABLE
To Kill a Mockingbird	Available
Pride and Prejudice	Issued
The Great Gatsby	Available
Moby Dick	PRE-ORDER
War and Peace	Issued
The Catcher in the Rye	Available
Jane Eyre	AVAILABLE
Brave New World	Available

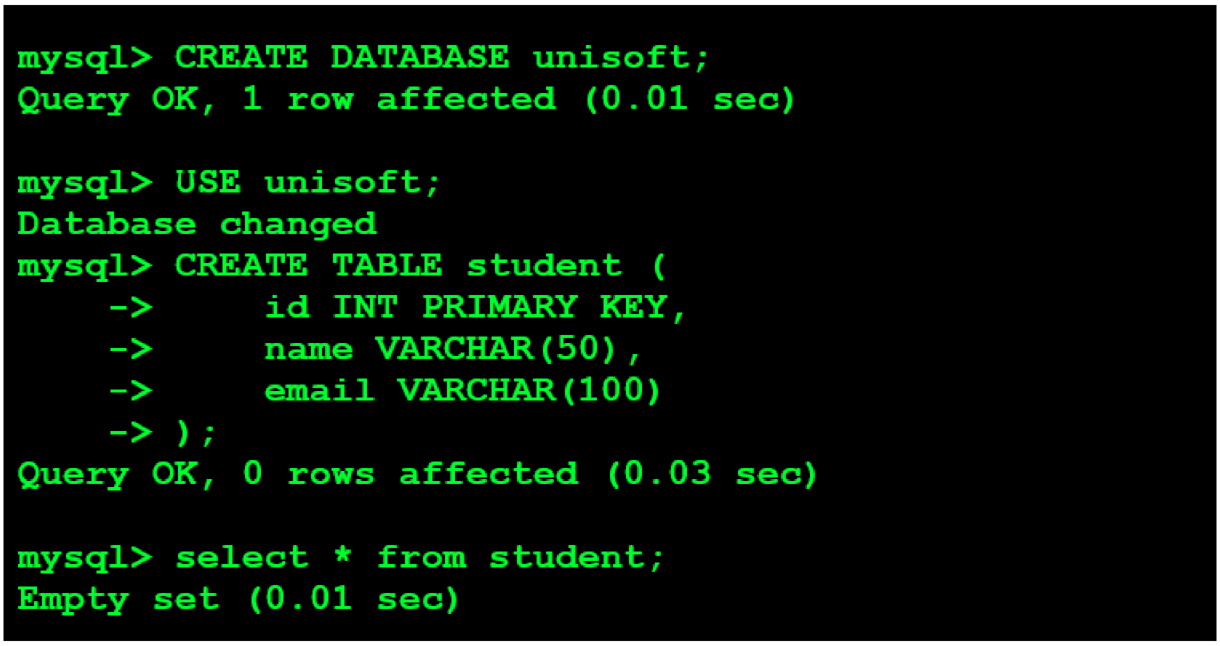
10 rows returned in 0.00 seconds

Name: Kale Prathmesh

Div.: A **Batch:** A3

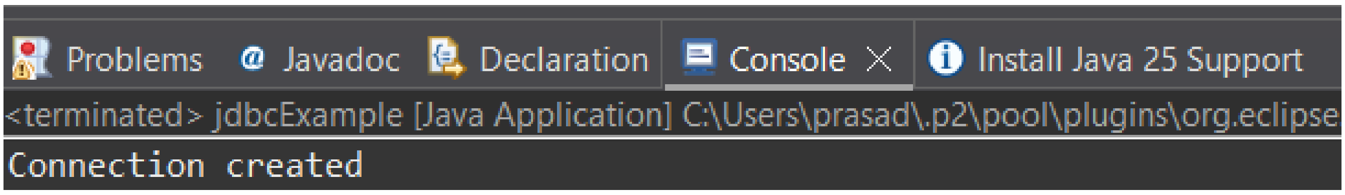
Roll no.: 55 **Practical no.:** 8

Output:



```
mysql> CREATE DATABASE unisoft;
Query OK, 1 row affected (0.01 sec)

mysql> USE unisoft;
Database changed
mysql> CREATE TABLE student (
    ->     id INT PRIMARY KEY,
    ->     name VARCHAR(50),
    ->     email VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> select * from student;
Empty set (0.01 sec)
```

```
package dbmsConn;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class jdbcExample {


	public static void main(String[] args)throws ClassNotFoundException, SQLException{

		Class.forName("com.mysql.cj.jdbc.Driver");

		Connection con=


	DriverManager.getConnection("jdbc:mysql://localhost:3306/unisoft","root","9075");

		System.out.println("Connection created");

	}

}
```

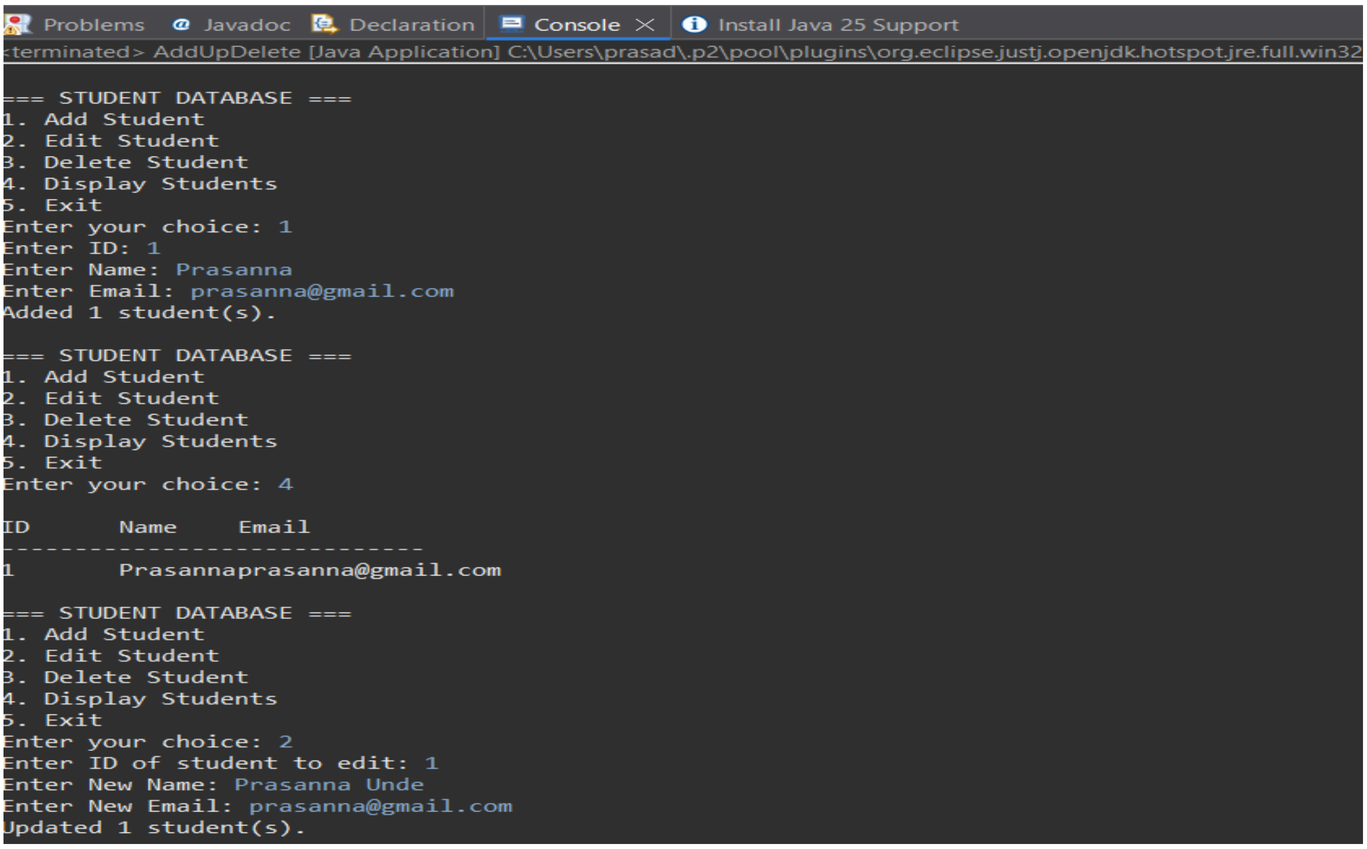


```
<terminated> jdbcExample [Java Application] C:\Users\prasad\.p2\pool\plugins\org.eclipse
Connection created
```

```
package dbmsConn;

import java.sql.*;

import java.util.Scanner;

public class AddUpDelete {

  static final String URL = "jdbc:mysql://localhost:3306/unisoft?useSSL=false&serverTimezone=UTC";

  static final String USER = "root";

  static final String PASSWORD = "9075";

  public static void main(String[] args) {

    try {

      Class.forName("com.mysql.cj.jdbc.Driver");

      try (Connection con = DriverManager.getConnection(URL, USER, PASSWORD);

         Scanner sc = new Scanner(System.in)) {

        int choice;

        do {

          System.out.println("\n=== STUDENT DATABASE ===");

          System.out.println("1. Add Student");

          System.out.println("2. Edit Student");

          System.out.println("3. Delete Student");

          System.out.println("4. Display Students");

          System.out.println("5. Exit");

          System.out.print("Enter your choice: ");

          choice = sc.nextInt();

          sc.nextLine();

          switch (choice) {

            case 1 -> {
```

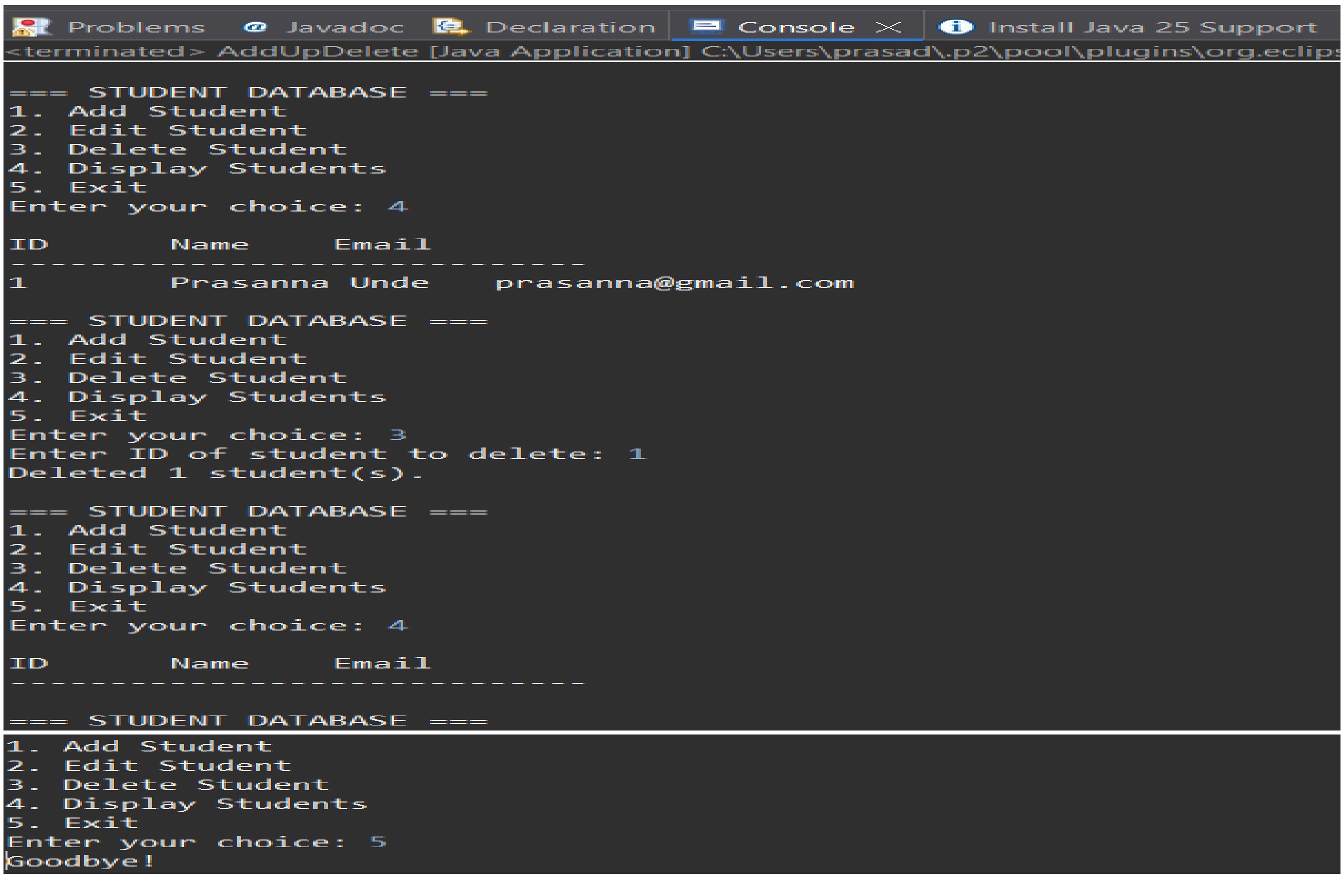
```
            System.out.print("Enter ID: ");
            int id = sc.nextInt();
            sc.nextLine();
            System.out.print("Enter Name: ");
            String name = sc.nextLine();
            System.out.print("Enter Email: ");
            String email = sc.nextLine();
            addStudent(con, id, name, email);
          }
          case 2 -> {
            System.out.print("Enter ID of student to edit: ");
            int id = sc.nextInt();
            sc.nextLine();
            System.out.print("Enter New Name: ");
            String name = sc.nextLine();
            System.out.print("Enter New Email: ");
            String email = sc.nextLine();
            updateStudent(con, id, name, email);
          }
          case 3 -> {
            System.out.print("Enter ID of student to delete: ");
            int id = sc.nextInt();
            deleteStudent(con, id);
          }
          case 4 -> displayStudents(con);
          case 5 -> System.out.println("Goodbye!");
          default -> System.out.println("Invalid choice.");
        }
      } while (choice != 5);
    }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public static void addStudent(Connection con, int id, String name, String email) throws SQLException {
    String sql = "INSERT INTO student (id, name, email) VALUES (?, ?, ?)";
    try (PreparedStatement ps = con.prepareStatement(sql)) {
      ps.setInt(1, id);
      ps.setString(2, name);
      ps.setString(3, email);
      int rows = ps.executeUpdate();
      System.out.println("Added " + rows + " student(s).");
    }
  }
  public static void updateStudent(Connection con, int id, String name, String email) throws
SQLException {
    String sql = "UPDATE student SET name = ?, email = ? WHERE id = ?";
    try (PreparedStatement ps = con.prepareStatement(sql)) {
      ps.setString(1, name);
      ps.setString(2, email);
      ps.setInt(3, id);
      int rows = ps.executeUpdate();
      System.out.println("Updated " + rows + " student(s).");
    }
  }
  public static void deleteStudent(Connection con, int id) throws SQLException {
    String sql = "DELETE FROM student WHERE id = ?";
    try (PreparedStatement ps = con.prepareStatement(sql)) {
      ps.setInt(1, id);
      int rows = ps.executeUpdate();
```

```
        System.out.println("Deleted " + rows + " student(s).");
      }
    }
  public static void displayStudents(Connection con) throws SQLException {
    String sql = "SELECT * FROM student";
    try (Statement stmt = con.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {
        System.out.println("\nID\tName\tEmail");
        System.out.println("----------------------------");
        while (rs.next()) {
          System.out.printf("%d\t%s\t%s%n", rs.getInt("id"), rs.getString("name"), rs.getString("email"));
        }
    }
  }
}
```



Problems Javadoc Declaration Console Install Java 25 Support

```
<terminated> AddUpDelete [Java Application] C:\Users\prasad\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32

=== STUDENT DATABASE ===
1. Add Student
2. Edit Student
3. Delete Student
4. Display Students
5. Exit
Enter your choice: 1
Enter ID: 1
Enter Name: Prasanna
Enter Email: prasanna@gmail.com
Added 1 student(s).

=== STUDENT DATABASE ===
1. Add Student
2. Edit Student
3. Delete Student
4. Display Students
5. Exit
Enter your choice: 4

ID      Name     Email
----------------------------
1       Prasannaprasanna@gmail.com

=== STUDENT DATABASE ===
1. Add Student
2. Edit Student
3. Delete Student
4. Display Students
5. Exit
Enter your choice: 2
Enter ID of student to edit: 1
Enter New Name: Prasanna Unde
Enter New Email: prasanna@gmail.com
Updated 1 student(s).
```



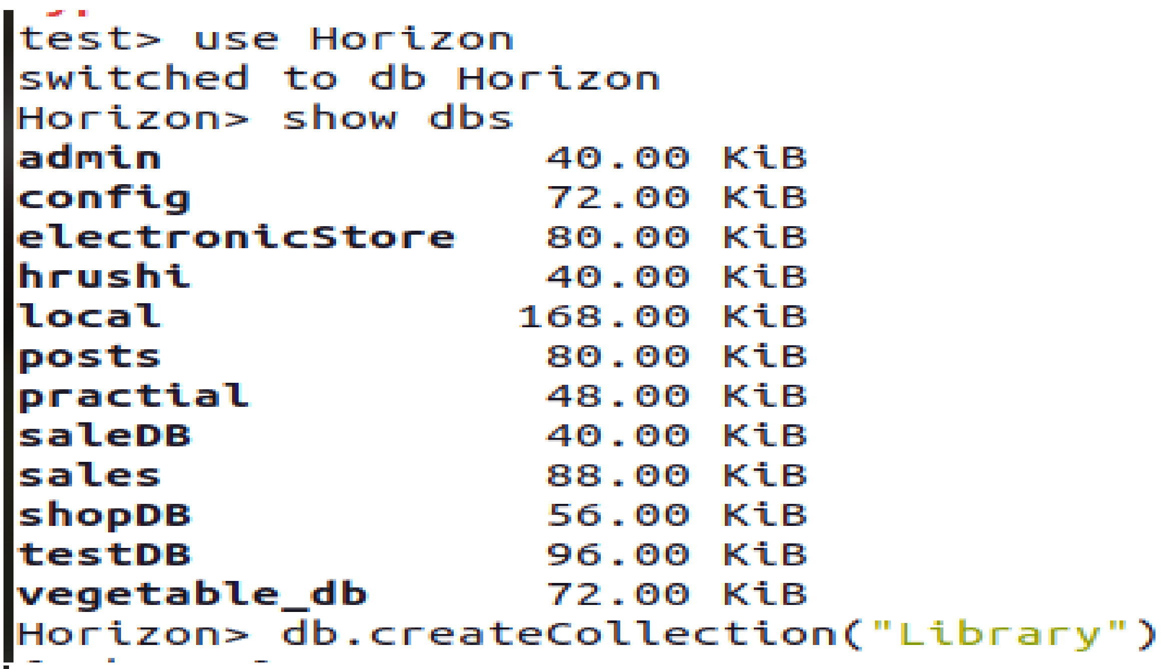
```
<terminated> AddUpDelete [Java Application] C:\Users\prasad\.p2\pool\plugins\org.eclips

=== STUDENT DATABASE ===
1. Add Student
2. Edit Student
3. Delete Student
4. Display Students
5. Exit
Enter your choice: 4

ID        Name      Email
------------------------------
1         Prasanna Unde    prasanna@gmail.com

=== STUDENT DATABASE ===
1. Add Student
2. Edit Student
3. Delete Student
4. Display Students
5. Exit
Enter your choice: 3
Enter ID of student to delete: 1
Deleted 1 student(s).

=== STUDENT DATABASE ===
1. Add Student
2. Edit Student
3. Delete Student
4. Display Students
5. Exit
Enter your choice: 4

ID        Name      Email
------------------------------

=== STUDENT DATABASE ===
1. Add Student
2. Edit Student
3. Delete Student
4. Display Students
5. Exit
Enter your choice: 5
Goodbye!
```

Name: Kale Prathmesh

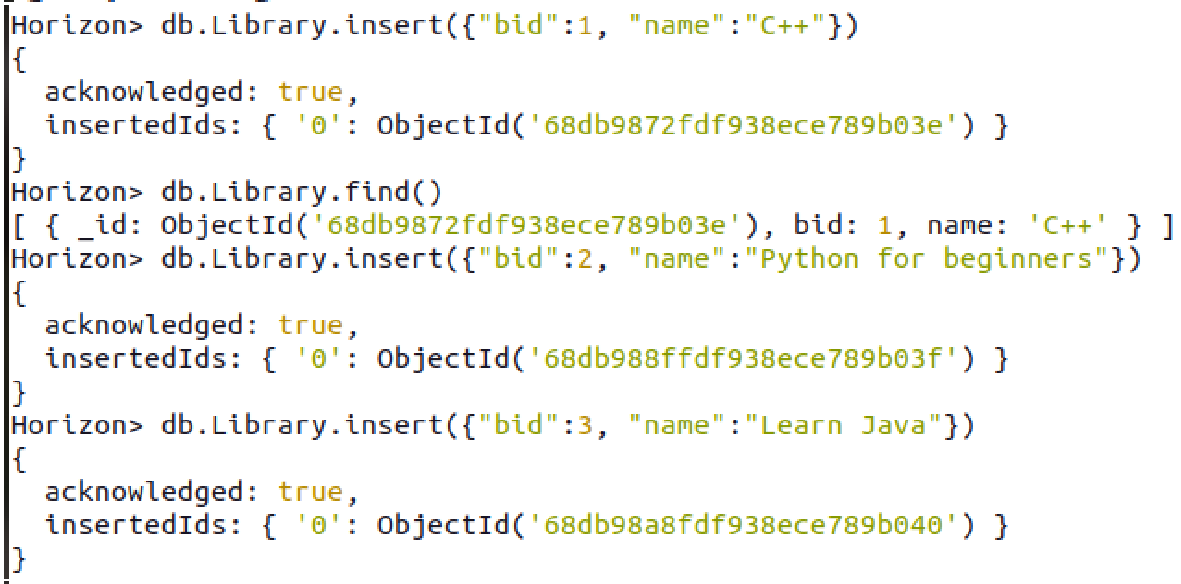
Div.: A **Batch:** A3

Roll no.: 55 **Practical no.:** 9

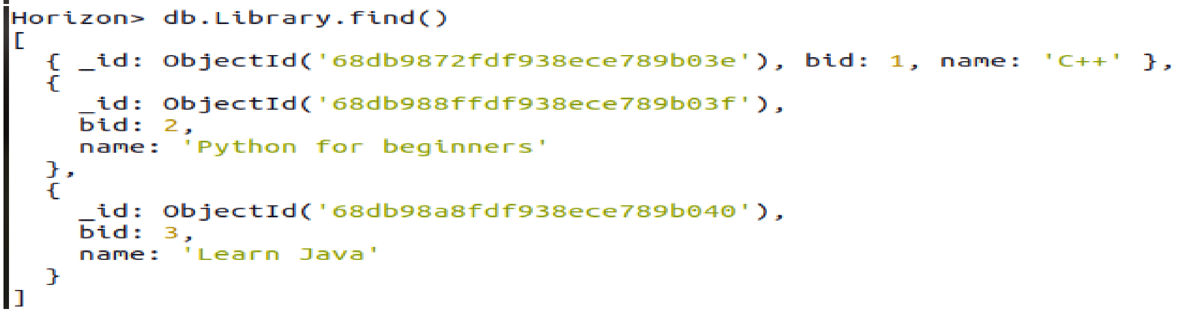
Output:



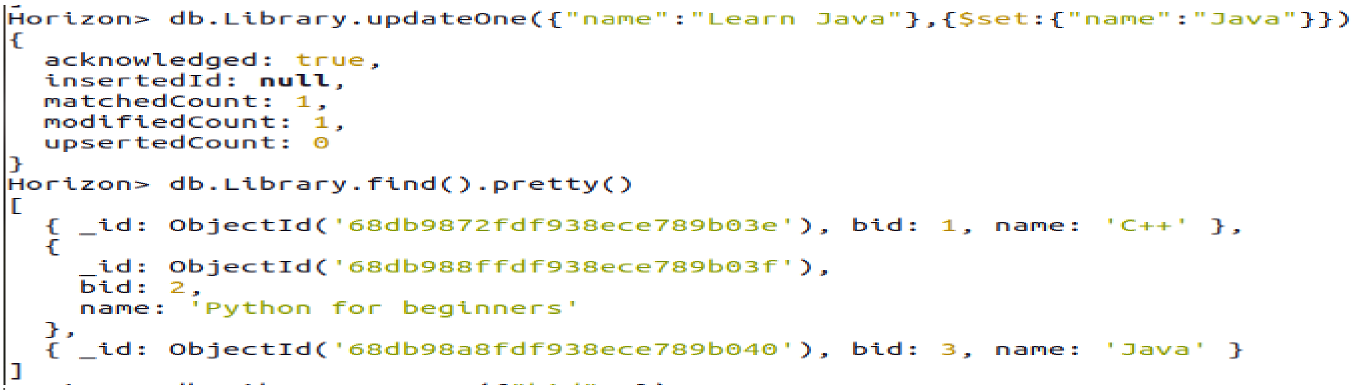
```
test> use Horizon
switched to db Horizon
Horizon> show dbs
admin                40.00 KiB
config               72.00 KiB
electronicStore      80.00 KiB
hrushi               40.00 KiB
local               168.00 KiB
posts                80.00 KiB
practial             48.00 KiB
saleDB               40.00 KiB
sales                88.00 KiB
shopDB               56.00 KiB
testDB               96.00 KiB
vegetable_db         72.00 KiB
Horizon> db.createCollection("Library")
```



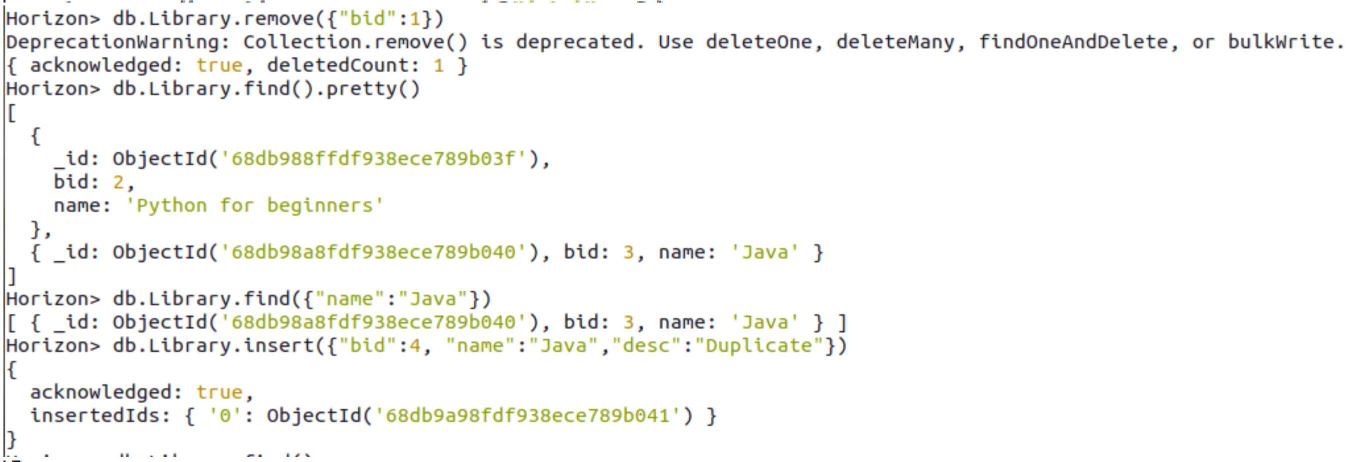
```
Horizon> db.Library.insert({"bid":1, "name":"C++"})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68db9872fdf938ece789b03e') }
}
Horizon> db.Library.find()
[ { _id: ObjectId('68db9872fdf938ece789b03e'), bid: 1, name: 'C++' } ]
Horizon> db.Library.insert({"bid":2, "name":"Python for beginners"})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68db988ffdf938ece789b03f') }
}
Horizon> db.Library.insert({"bid":3, "name":"Learn Java"})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68db98a8fdf938ece789b040') }
}
```



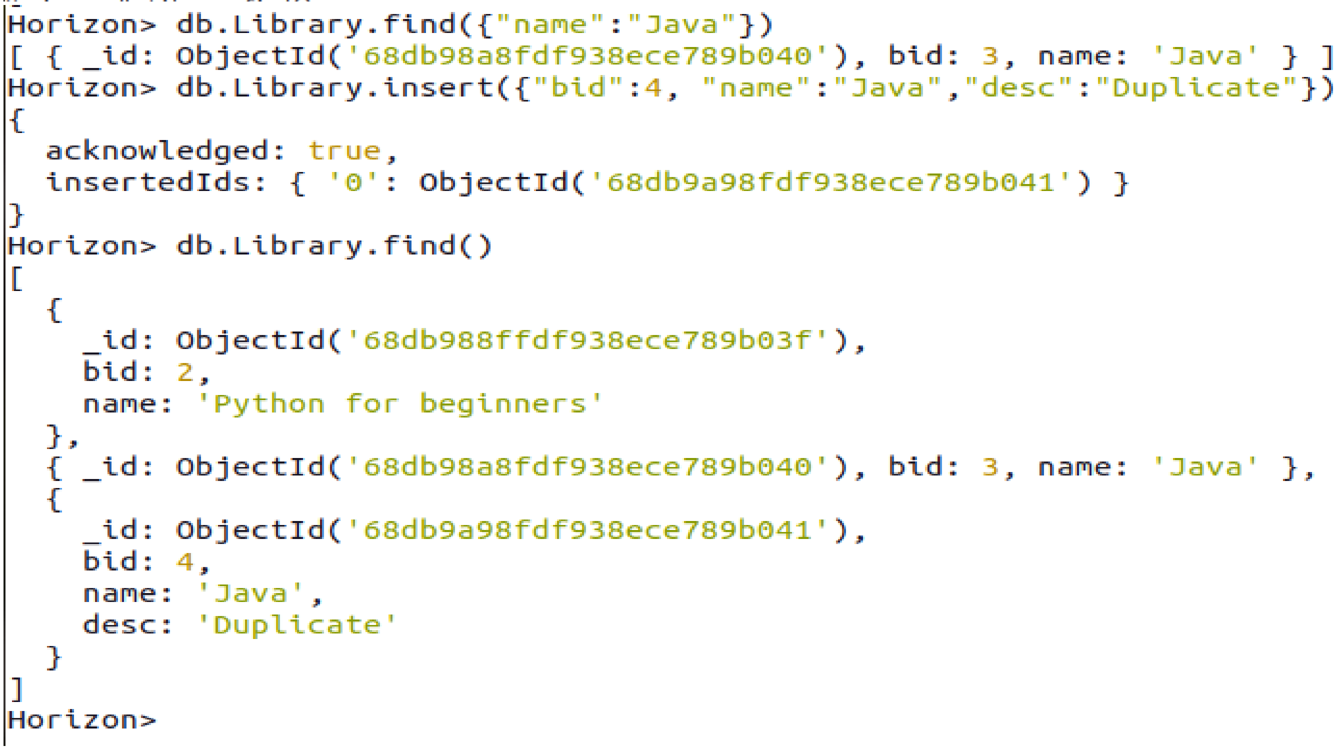
```
Horizon> db.Library.find()
[
  { _id: ObjectId('68db9872fdf938ece789b03e'), bid: 1, name: 'C++' },
  {
    _id: ObjectId('68db988ffdf938ece789b03f'),
    bid: 2,
    name: 'Python for beginners'
  },
  {
    _id: ObjectId('68db98a8fdf938ece789b040'),
    bid: 3,
    name: 'Learn Java'
  }
]
```



```
Horizon> db.Library.updateOne({"name":"Learn Java"},{$set:{"name":"Java"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Horizon> db.Library.find().pretty()
[
  { _id: ObjectId('68db9872fdf938ece789b03e'), bid: 1, name: 'C++' },
  {
    _id: ObjectId('68db988ffdf938ece789b03f'),
    bid: 2,
    name: 'Python for beginners'
  },
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' }
]
```

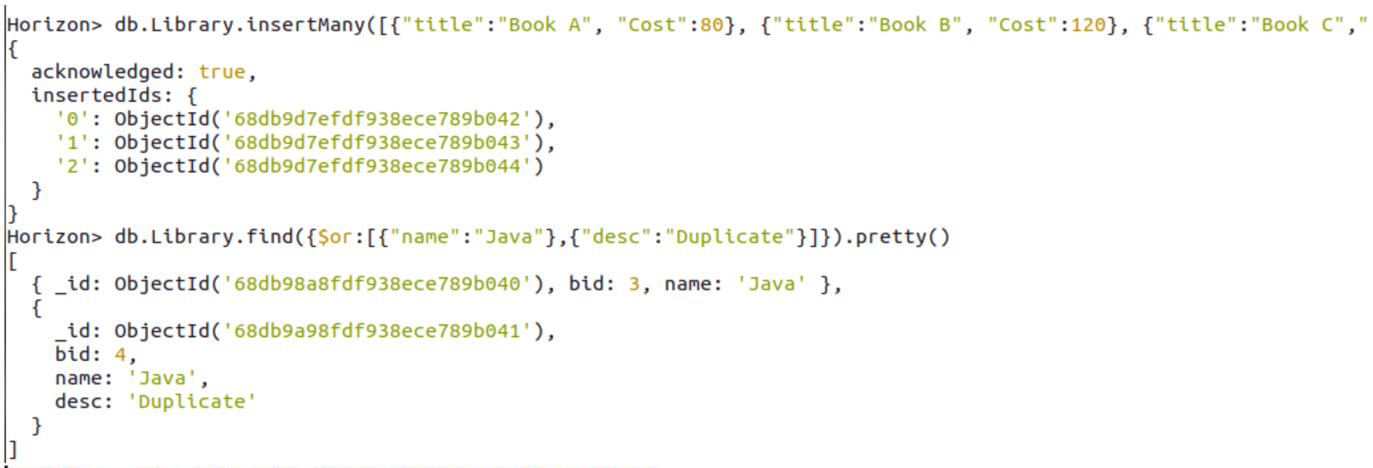


```
Horizon> db.Library.remove({"bid":1})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 1 }
Horizon> db.Library.find().pretty()
[
  {
    _id: ObjectId('68db988ffdf938ece789b03f'),
    bid: 2,
    name: 'Python for beginners'
  },
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' }
]
Horizon> db.Library.find({"name":"Java"})
[ { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' } ]
Horizon> db.Library.insert({"bid":4, "name":"Java","desc":"Duplicate"})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68db9a98fdf938ece789b041') }
}
```

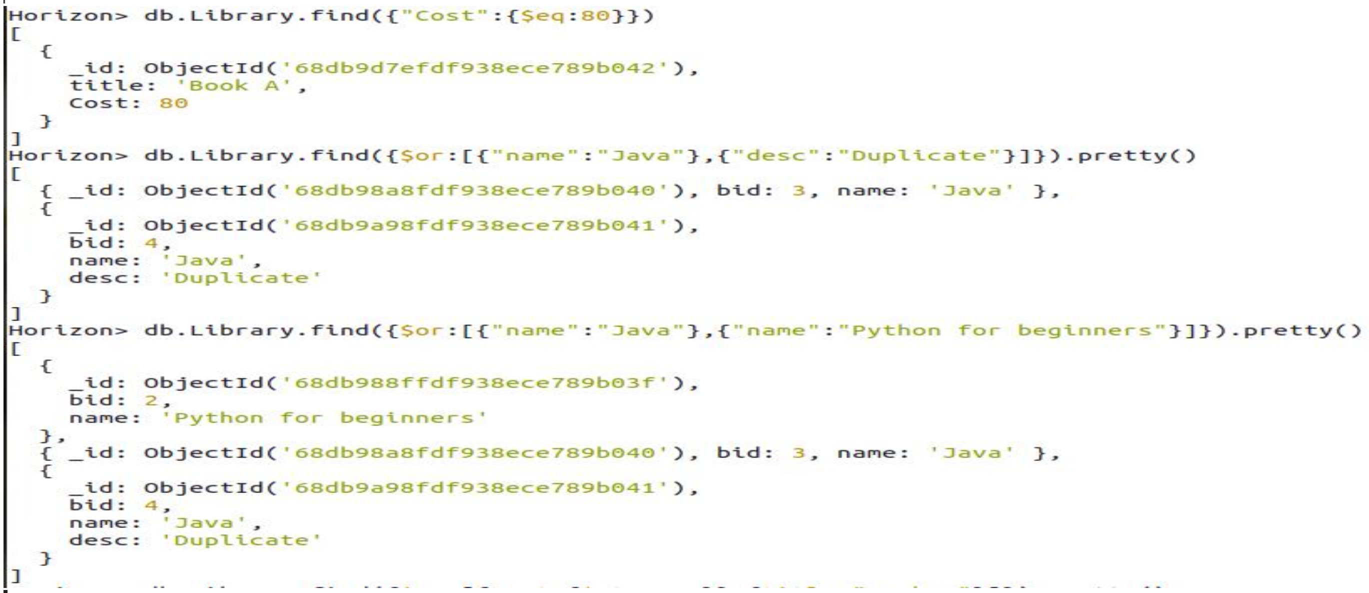


```
Horizon> db.Library.find({"name":"Java"})
[ { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' } ]
Horizon> db.Library.insert({"bid":4, "name":"Java","desc":"Duplicate"})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('68db9a98fdf938ece789b041') }
}
```

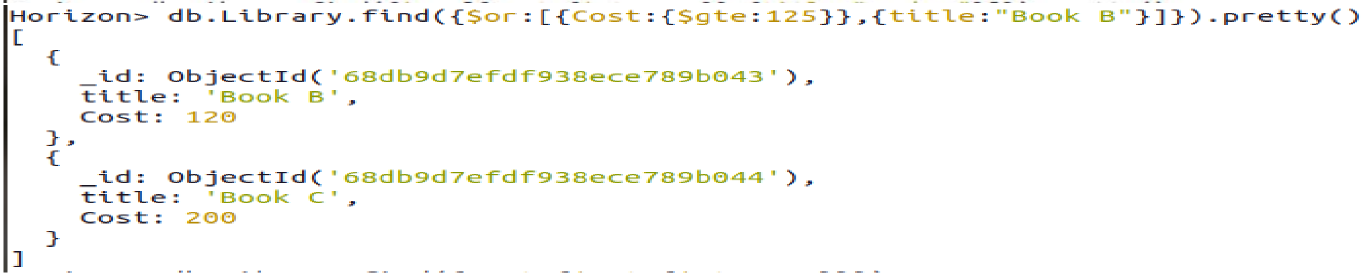
```
Horizon> db.Library.find()
[
  {
    _id: ObjectId('68db988ffdf938ece789b03f'),
    bid: 2,
    name: 'Python for beginners'
  },
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' },
  {
    _id: ObjectId('68db9a98fdf938ece789b041'),
    bid: 4,
    name: 'Java',
    desc: 'Duplicate'
  }
]
Horizon>
```



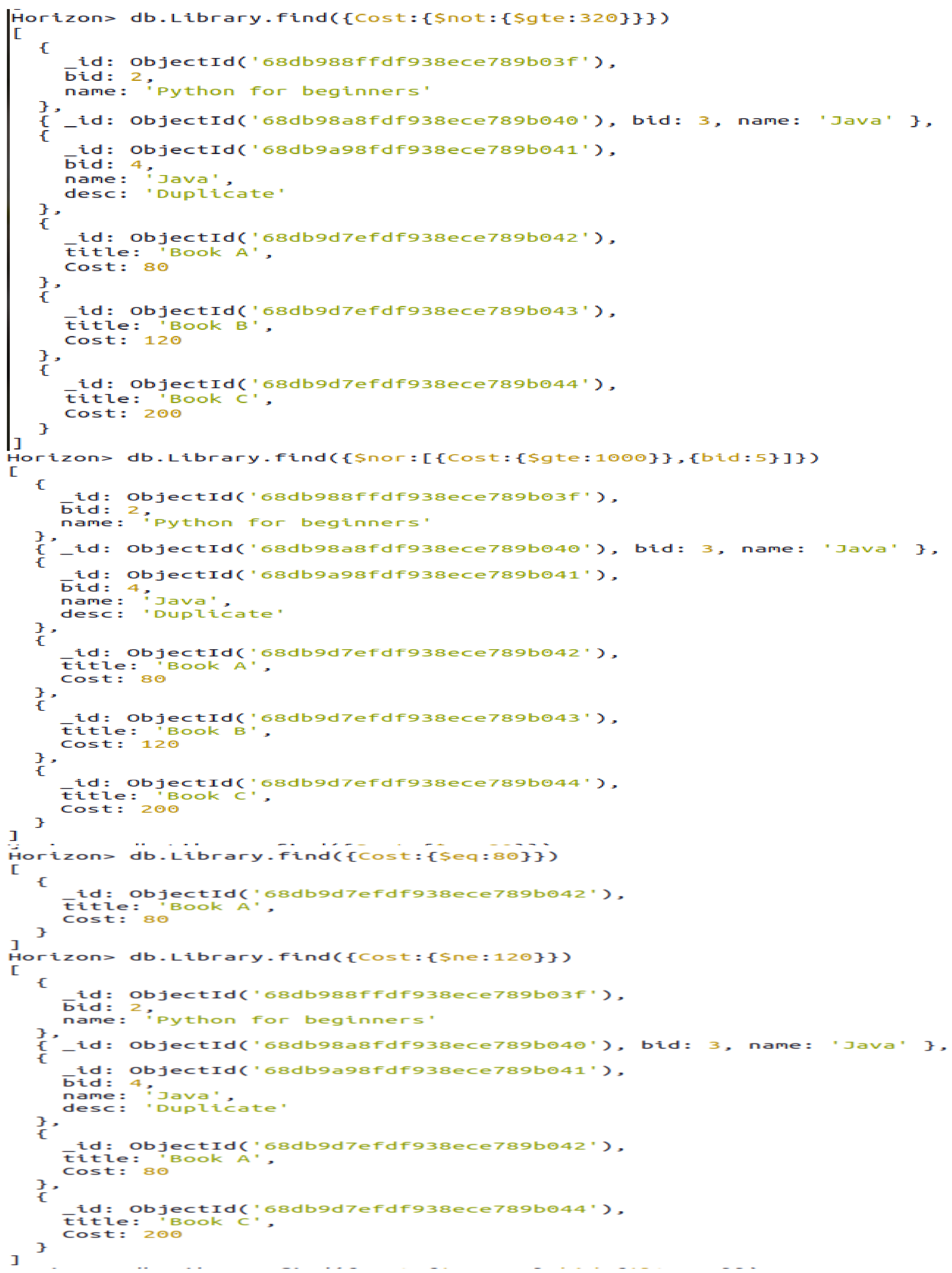
```
Horizon> db.Library.insertMany([{"title":"Book A", "Cost":80}, {"title":"Book B", "Cost":120}, {"title":"Book C","
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68db9d7efdf938ece789b042'),
    '1': ObjectId('68db9d7efdf938ece789b043'),
    '2': ObjectId('68db9d7efdf938ece789b044')
  }
}
Horizon> db.Library.find({$or:[{"name":"Java"},{"desc":"Duplicate"}]}).pretty()
[
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' },
  {
    _id: ObjectId('68db9a98fdf938ece789b041'),
    bid: 4,
    name: 'Java',
    desc: 'Duplicate'
  }
]
```



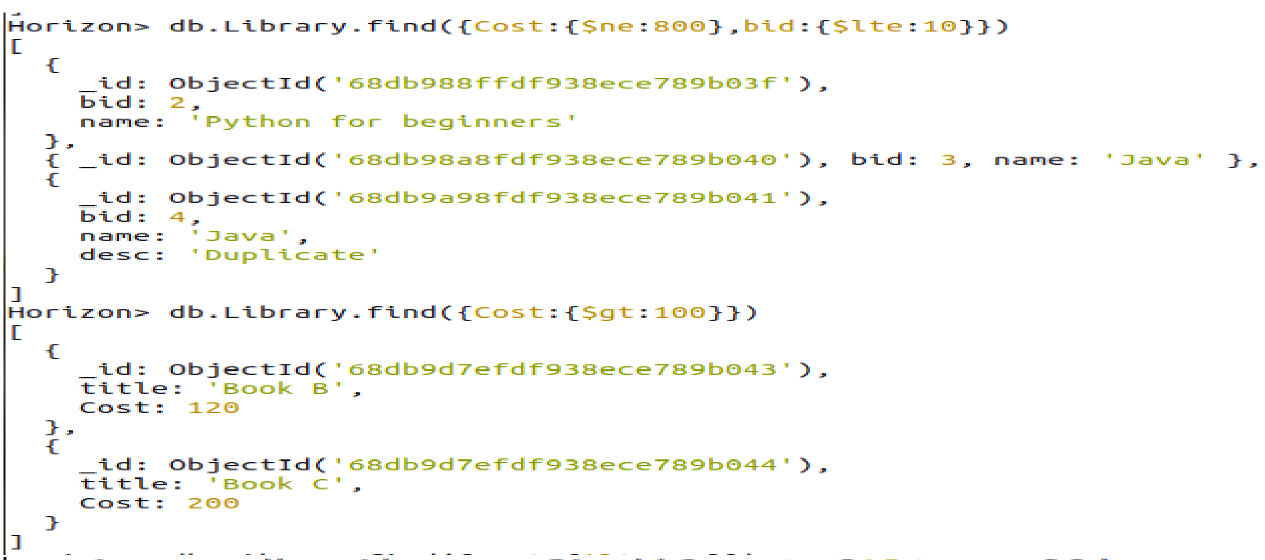
```
Horizon> db.Library.find({"Cost":{$eq:80}})
[
  {
    _id: ObjectId('68db9d7efdf938ece789b042'),
    title: 'Book A',
    Cost: 80
  }
]
Horizon> db.Library.find({$or:[{"name":"Java"},{"desc":"Duplicate"}]}).pretty()
[
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' },
  {
    _id: ObjectId('68db9a98fdf938ece789b041'),
    bid: 4,
    name: 'Java',
    desc: 'Duplicate'
  }
]
Horizon> db.Library.find({$or:[{"name":"Java"},{"name":"Python for beginners"}]}).pretty()
[
  {
    _id: ObjectId('68db988ffdf938ece789b03f'),
    bid: 2,
    name: 'Python for beginners'
  },
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' },
  {
    _id: ObjectId('68db9a98fdf938ece789b041'),
    bid: 4,
    name: 'Java',
    desc: 'Duplicate'
  }
]
```



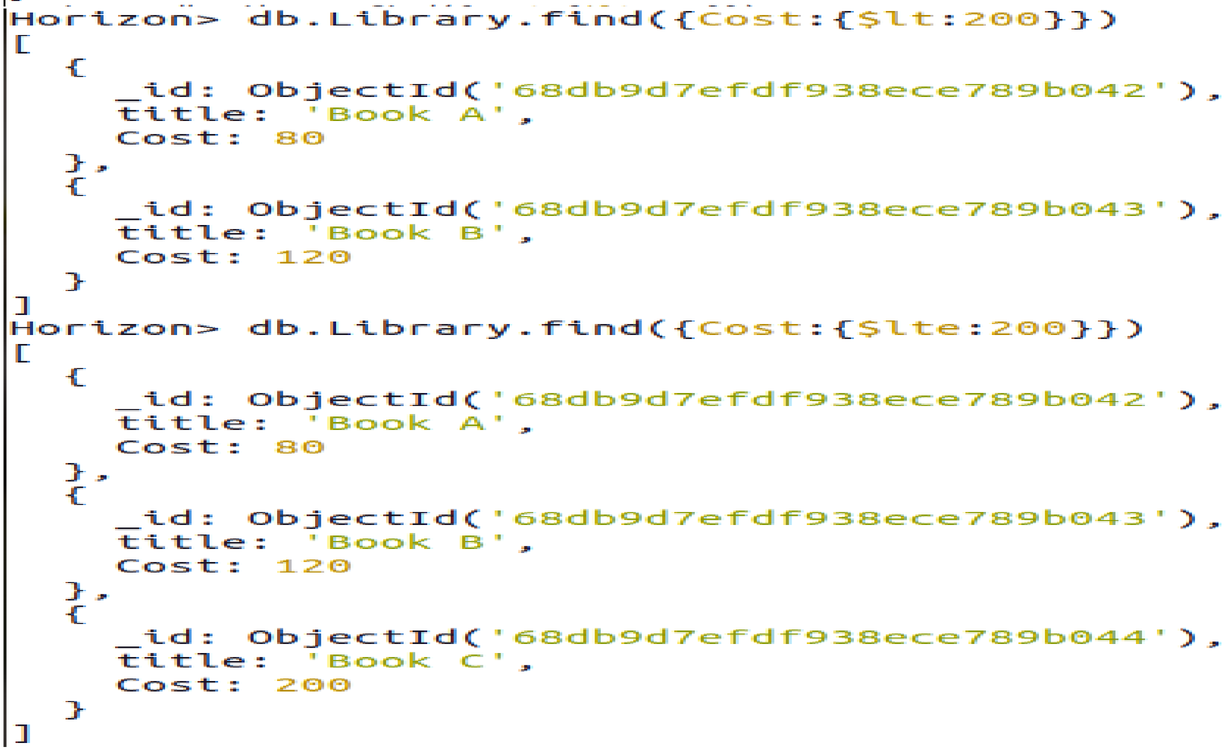
```
Horizon> db.Library.find({$or:[{Cost:{$gte:125}},{title:"Book B"}]}).pretty()
[
  {
    _id: ObjectId('68db9d7efdf938ece789b043'),
    title: 'Book B',
    Cost: 120
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b044'),
    title: 'Book C',
    Cost: 200
  }
]
```



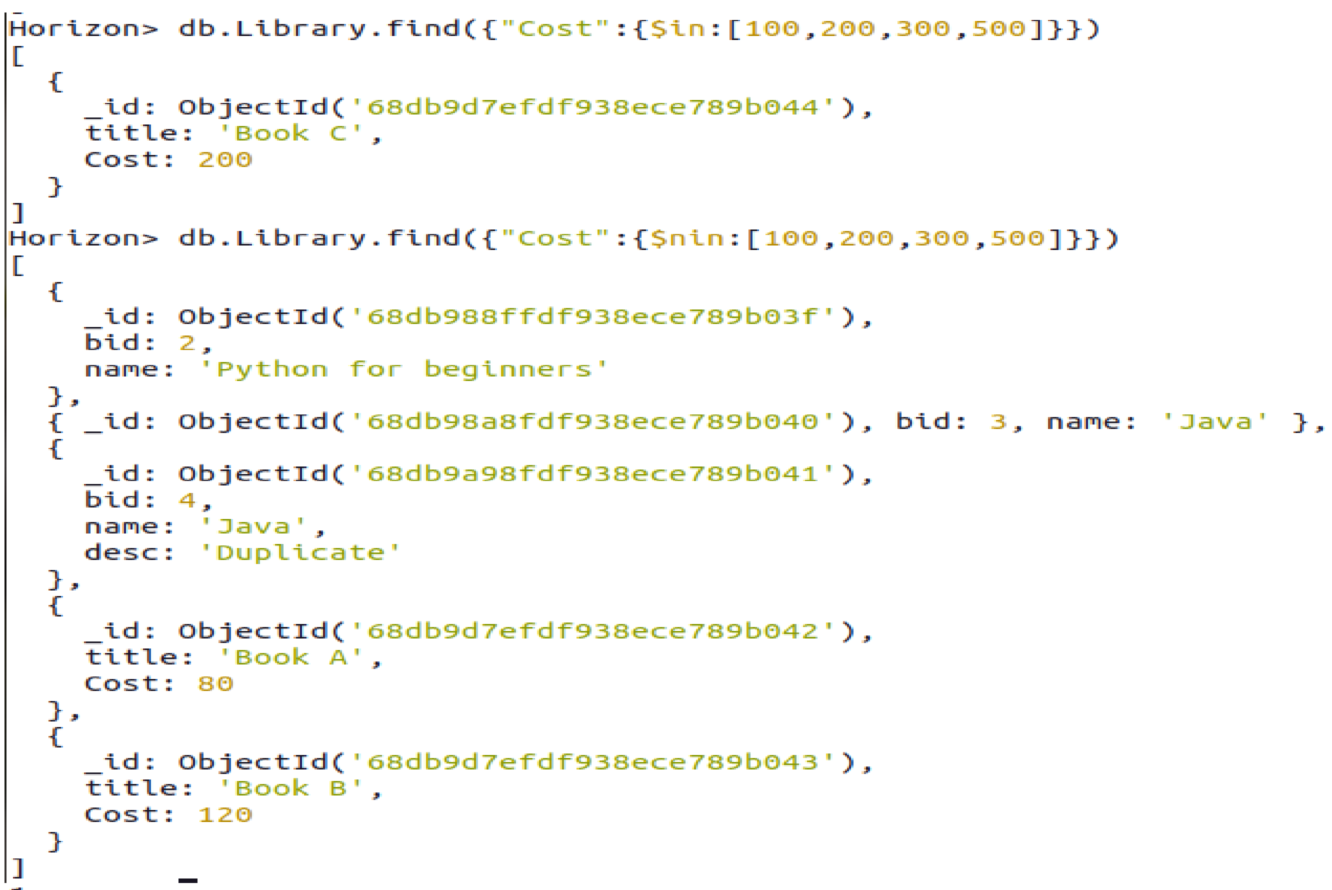
```
Horizon> db.Library.find({Cost:{$not:{$gte:320}}})
[
  {
    _id: ObjectId('68db988ffdf938ece789b03f'),
    bid: 2,
    name: 'Python for beginners'
  },
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' },
  {
    _id: ObjectId('68db9a98fdf938ece789b041'),
    bid: 4,
    name: 'Java',
    desc: 'Duplicate'
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b042'),
    title: 'Book A',
    Cost: 80
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b043'),
    title: 'Book B',
    Cost: 120
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b044'),
    title: 'Book C',
    Cost: 200
  }
]
Horizon> db.Library.find({$nor:[{Cost:{$gte:1000}},{bid:5}]})
[
  {
    _id: ObjectId('68db988ffdf938ece789b03f'),
    bid: 2,
    name: 'Python for beginners'
  },
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' },
  {
    _id: ObjectId('68db9a98fdf938ece789b041'),
    bid: 4,
    name: 'Java',
    desc: 'Duplicate'
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b042'),
    title: 'Book A',
    Cost: 80
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b043'),
    title: 'Book B',
    Cost: 120
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b044'),
    title: 'Book C',
    Cost: 200
  }
]
Horizon> db.Library.find({Cost:{$eq:80}})
[
  {
    _id: ObjectId('68db9d7efdf938ece789b042'),
    title: 'Book A',
    Cost: 80
  }
]
Horizon> db.Library.find({Cost:{$ne:120}})
[
  {
    _id: ObjectId('68db988ffdf938ece789b03f'),
    bid: 2,
    name: 'Python for beginners'
  },
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' },
  {
    _id: ObjectId('68db9a98fdf938ece789b041'),
    bid: 4,
    name: 'Java',
    desc: 'Duplicate'
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b042'),
    title: 'Book A',
    Cost: 80
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b044'),
    title: 'Book C',
    Cost: 200
  }
]
```



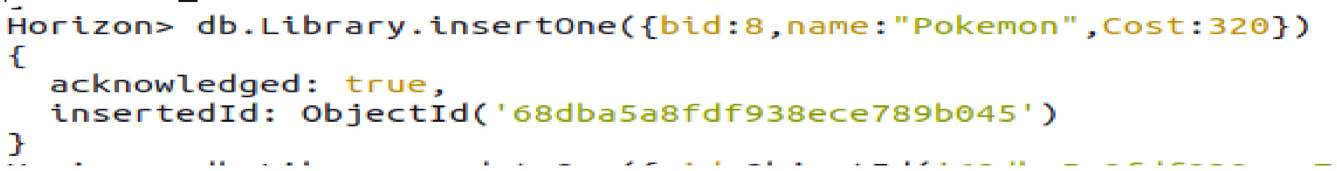
```
Horizon> db.Library.find({Cost:{$ne:800},bid:{$lte:10}})
[
  {
    _id: ObjectId('68db988ffdf938ece789b03f'),
    bid: 2,
    name: 'Python for beginners'
  },
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' },
  {
    _id: ObjectId('68db9a98fdf938ece789b041'),
    bid: 4,
    name: 'Java',
    desc: 'Duplicate'
  }
]
Horizon> db.Library.find({Cost:{$gt:100}})
[
  {
    _id: ObjectId('68db9d7efdf938ece789b043'),
    title: 'Book B',
    Cost: 120
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b044'),
    title: 'Book C',
    Cost: 200
  }
]
```



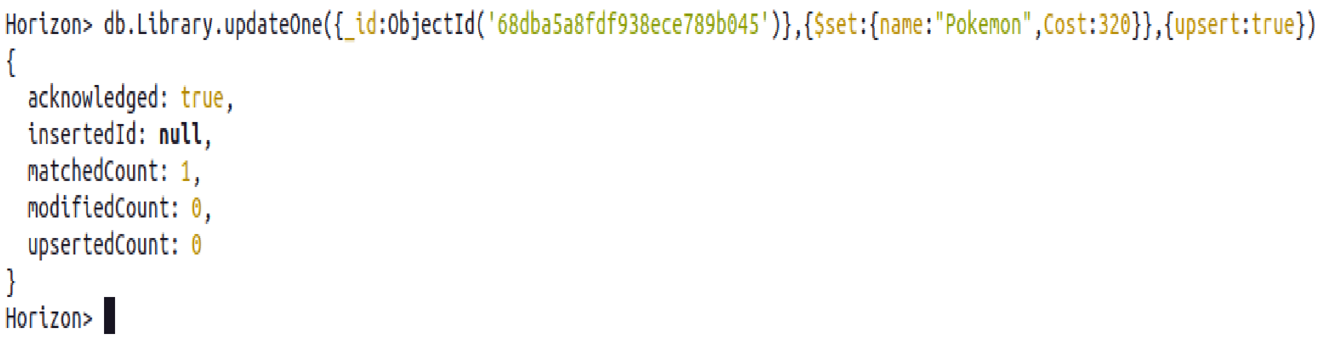
```
Horizon> db.Library.find({Cost:{$lt:200}})
[
  {
    _id: ObjectId('68db9d7efdf938ece789b042'),
    title: 'Book A',
    Cost: 80
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b043'),
    title: 'Book B',
    Cost: 120
  }
]
Horizon> db.Library.find({Cost:{$lte:200}})
[
  {
    _id: ObjectId('68db9d7efdf938ece789b042'),
    title: 'Book A',
    Cost: 80
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b043'),
    title: 'Book B',
    Cost: 120
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b044'),
    title: 'Book C',
    Cost: 200
  }
]
```



```
Horizon> db.Library.find({"Cost":{$in:[100,200,300,500]}})
[
  {
    _id: ObjectId('68db9d7efdf938ece789b044'),
    title: 'Book C',
    Cost: 200
  }
]
Horizon> db.Library.find({"Cost":{$nin:[100,200,300,500]}})
[
  {
    _id: ObjectId('68db988ffdf938ece789b03f'),
    bid: 2,
    name: 'Python for beginners'
  },
  { _id: ObjectId('68db98a8fdf938ece789b040'), bid: 3, name: 'Java' },
  {
    _id: ObjectId('68db9a98fdf938ece789b041'),
    bid: 4,
    name: 'Java',
    desc: 'Duplicate'
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b042'),
    title: 'Book A',
    Cost: 80
  },
  {
    _id: ObjectId('68db9d7efdf938ece789b043'),
    title: 'Book B',
    Cost: 120
  }
]
```



```
Horizon> db.Library.insertOne({bid:8,name:"Pokemon",Cost:320})
{
  acknowledged: true,
  insertedId: ObjectId('68dba5a8fdf938ece789b045')
}
```



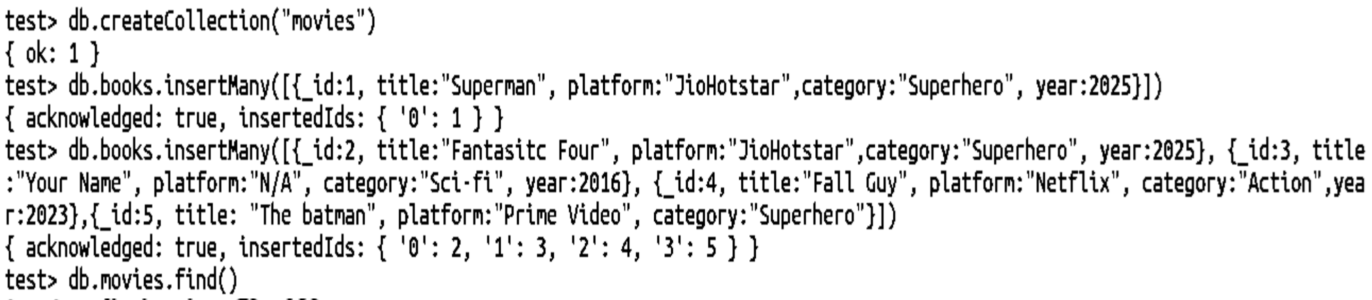
```
Horizon> db.Library.updateOne({_id:ObjectId('68dba5a8fdf938ece789b045')},{$set:{name:"Pokemon",Cost:320}},{upsert:true})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
Horizon>
```

Name: Kale Prathmesh

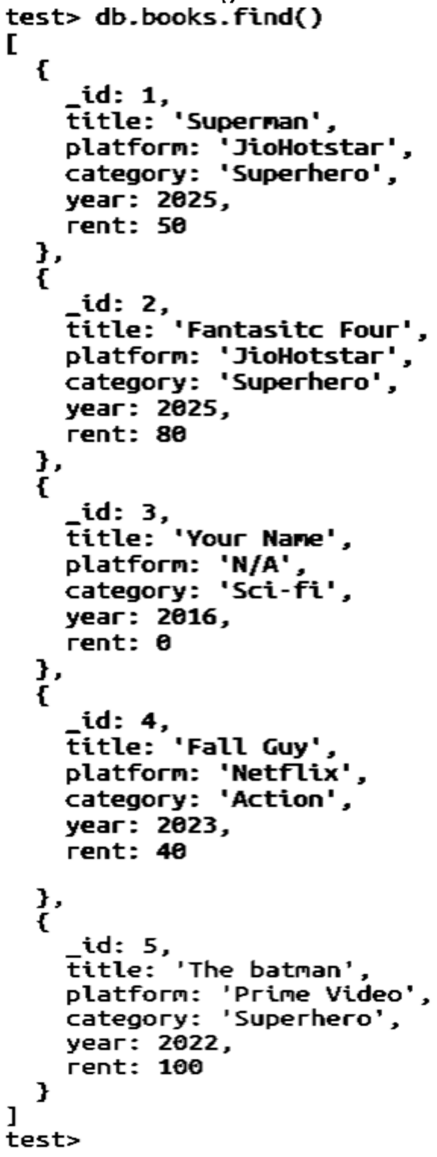
Div.: A **Batch:** A3

Roll no.: 55 **Practical no.:** 10

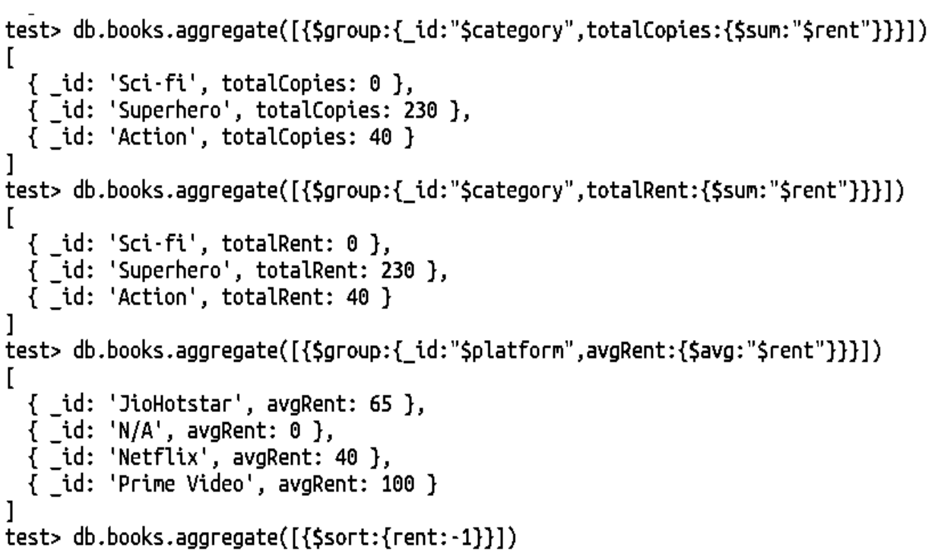
Output:



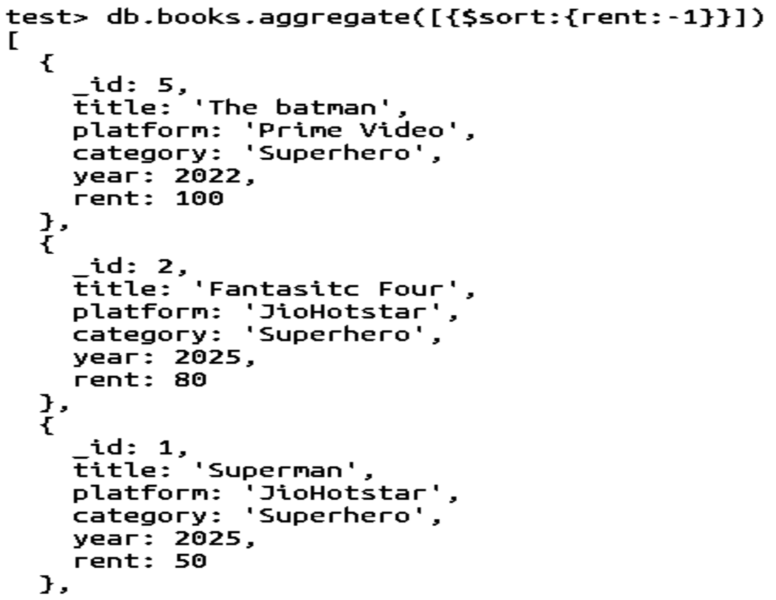
```
test> db.createCollection("movies")
{ ok: 1 }
test> db.books.insertMany([{_id:1, title:"Superman", platform:"JioHotstar",category:"Superhero", year:2025}])
{ acknowledged: true, insertedIds: { '0': 1 } }
test> db.books.insertMany([{_id:2, title:"Fantasitc Four", platform:"JioHotstar",category:"Superhero", year:2025}, {_id:3, title
:"Your Name", platform:"N/A", category:"Sci-fi", year:2016}, {_id:4, title:"Fall Guy", platform:"Netflix", category:"Action",yea
r:2023},{_id:5, title: "The batman", platform:"Prime Video", category:"Superhero"}])
{ acknowledged: true, insertedIds: { '0': 2, '1': 3, '2': 4, '3': 5 } }
test> db.movies.find()
```



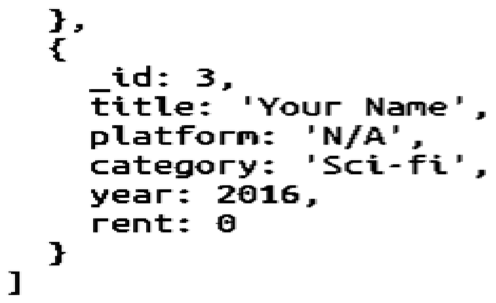
```
test> db.books.find()
[
  {
    _id: 1,
    title: 'Superman',
    platform: 'JioHotstar',
    category: 'Superhero',
    year: 2025,
    rent: 50
  },
  {
    _id: 2,
    title: 'Fantasitc Four',
    platform: 'JioHotstar',
    category: 'Superhero',
    year: 2025,
    rent: 80
  },
  {
    _id: 3,
    title: 'Your Name',
    platform: 'N/A',
    category: 'Sci-fi',
    year: 2016,
    rent: 0
  },
  {
    _id: 4,
    title: 'Fall Guy',
    platform: 'Netflix',
    category: 'Action',
    year: 2023,
    rent: 40

  },
  {
    _id: 5,
    title: 'The batman',
    platform: 'Prime Video',
    category: 'Superhero',
    year: 2022,
    rent: 100
  }
]
test>
```



```
test> db.books.aggregate([{$group:{_id:"$category",totalCopies:{$sum:"$rent"}}}])
[
  { _id: 'Sci-fi', totalCopies: 0 },
  { _id: 'Superhero', totalCopies: 230 },
  { _id: 'Action', totalCopies: 40 }
]
test> db.books.aggregate([{$group:{_id:"$category",totalRent:{$sum:"$rent"}}}])
[
  { _id: 'Sci-fi', totalRent: 0 },
  { _id: 'Superhero', totalRent: 230 },
  { _id: 'Action', totalRent: 40 }
]
test> db.books.aggregate([{$group:{_id:"$platform",avgRent:{$avg:"$rent"}}}])
[
  { _id: 'JioHotstar', avgRent: 65 },
  { _id: 'N/A', avgRent: 0 },
  { _id: 'Netflix', avgRent: 40 },
  { _id: 'Prime Video', avgRent: 100 }
]
test> db.books.aggregate([{$sort:{rent:-1}}])
```

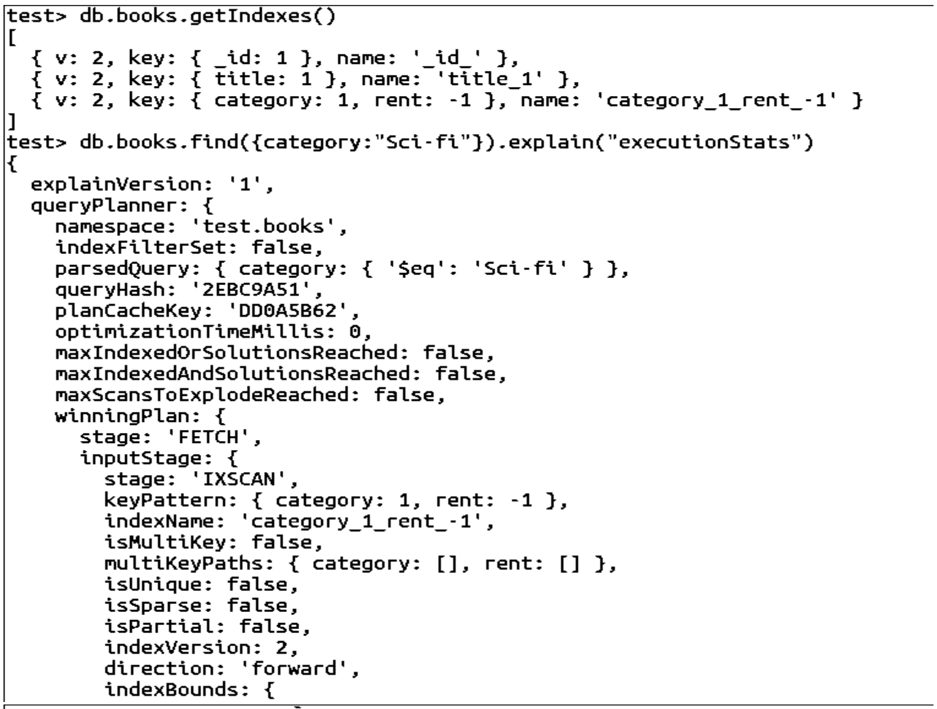


```
test> db.books.aggregate([{$sort:{rent:-1}}])
[
  {
    _id: 5,
    title: 'The batman',
    platform: 'Prime Video',
    category: 'Superhero',
    year: 2022,
    rent: 100
  },
  {
    _id: 2,
    title: 'Fantasitc Four',
    platform: 'JioHotstar',
    category: 'Superhero',
    year: 2025,
    rent: 80
  },
  {
    _id: 1,
    title: 'Superman',
    platform: 'JioHotstar',
    category: 'Superhero',
    year: 2025,
    rent: 50
  },
```

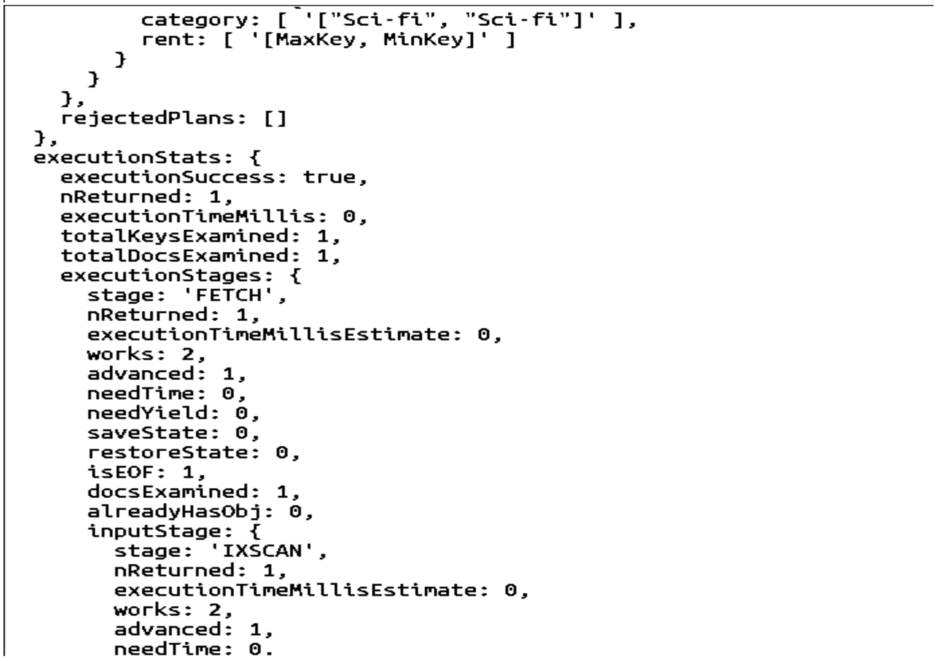


```
  },
  {
    _id: 3,
    title: 'Your Name',
    platform: 'N/A',
    category: 'Sci-fi',
    year: 2016,
    rent: 0
  }
]
```

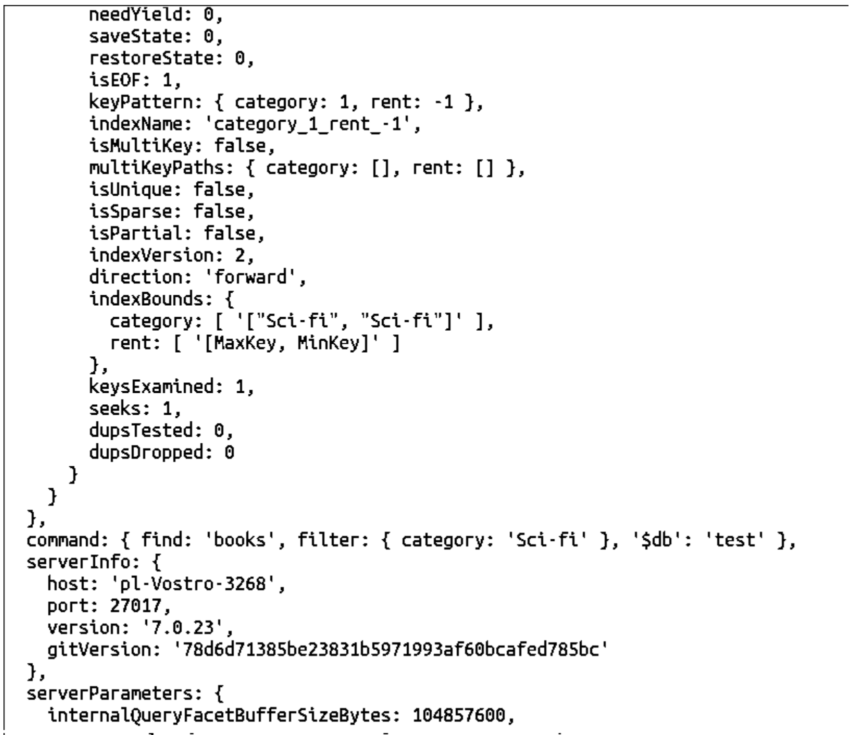
```
test> db.books.aggregate([{$sort:{rent:1}}])
[
  {
    _id: 3,
    title: 'Your Name',
    platform: 'N/A',
    category: 'Sci-fi',
    year: 2016,
    rent: 0
  },
  {
    _id: 4,
    title: 'Fall Guy',
    platform: 'Netflix',
    category: 'Action',
    year: 2023,
    rent: 40
  },
  {
    _id: 1,
    title: 'Superman',
    platform: 'JioHotstar',
    category: 'Superhero',
    year: 2025,
    rent: 50
  },
  {
    _id: 2,
    title: 'Fantasitc Four',
    platform: 'JioHotstar',
    category: 'Superhero',
    year: 2025,
    rent: 80
  },
  {
    _id: 5,
    title: 'The batman',
    platform: 'Prime Video',
    category: 'Superhero',
    year: 2022,
    rent: 100
  }
]
test> db.books.aggregate([{$project:{_id:0,title:1, platform:1,rent:1}}])
[
  { title: 'Superman', platform: 'JioHotstar', rent: 50 },
  { title: 'Fantasitc Four', platform: 'JioHotstar', rent: 80 },
  { title: 'Your Name', platform: 'N/A', rent: 0 },
  { title: 'Fall Guy', platform: 'Netflix', rent: 40 },
  { title: 'The batman', platform: 'Prime Video', rent: 100 }
]
test> db.books.aggregate([{$match:{category:"Superhero",rent:{$gt:10}}},{$sort:{rent:-1}},{$project:{_id:0,title:1,rent:1}}])
[
  { title: 'The batman', rent: 100 },
  { title: 'Fantasitc Four', rent: 80 },
  { title: 'Superman', rent: 50 }
]
test> db.books.aggregate([{$match:{category:"Superhero",rent:{$gt:50}}},{$sort:{rent:-1}},{$project:{_id:0,title:1,rent:1}}])
[
  { title: 'The batman', rent: 100 },
  { title: 'Fantasitc Four', rent: 80 }
]
test> db.books.createIndex({title:1})
title_1
test> db.books.createIndex({category:1,rent:-1})
category_1_rent_-1
```



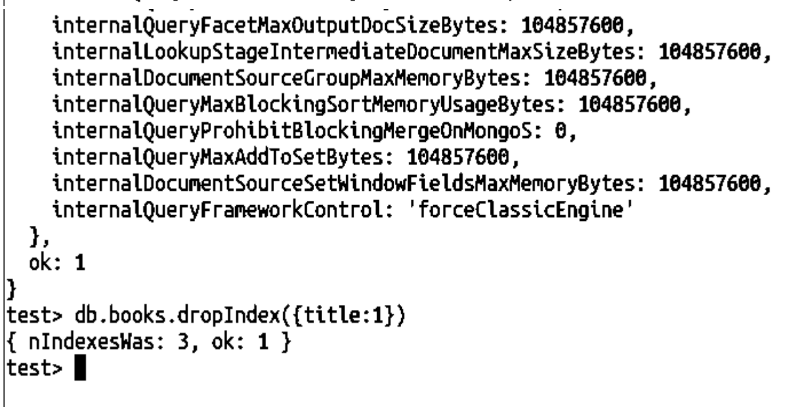
```
test> db.books.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { title: 1 }, name: 'title_1' },
  { v: 2, key: { category: 1, rent: -1 }, name: 'category_1_rent_-1' }
]
test> db.books.find({category:"Sci-fi"}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'test.books',
    indexFilterSet: false,
    parsedQuery: { category: { '$eq': 'Sci-fi' } },
    queryHash: '2EBC9A51',
    planCacheKey: 'DD0A5B62',
    optimizationTimeMillis: 0,
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { category: 1, rent: -1 },
        indexName: 'category_1_rent_-1',
        isMultiKey: false,
        multiKeyPaths: { category: [], rent: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: {
```



```
          category: [ '["Sci-fi", "Sci-fi"]' ],
          rent: [ '[MaxKey, MinKey]' ]
        }
      }
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 0,
    totalKeysExamined: 1,
    totalDocsExamined: 1,
    executionStages: {
      stage: 'FETCH',
      nReturned: 1,
      executionTimeMillisEstimate: 0,
      works: 2,
      advanced: 1,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      docsExamined: 1,
      alreadyHasObj: 0,
      inputStage: {
        stage: 'IXSCAN',
        nReturned: 1,
        executionTimeMillisEstimate: 0,
        works: 2,
        advanced: 1,
        needTime: 0.
```



```
        needYield: 0,
        saveState: 0,
        restoreState: 0,
        isEOF: 1,
        keyPattern: { category: 1, rent: -1 },
        indexName: 'category_1_rent_-1',
        isMultiKey: false,
        multiKeyPaths: { category: [], rent: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: {
          category: [ '["Sci-fi", "Sci-fi"]' ],
          rent: [ '[MaxKey, MinKey]' ]
        },
        keysExamined: 1,
        seeks: 1,
        dupsTested: 0,
        dupsDropped: 0
      }
    }
  },
  command: { find: 'books', filter: { category: 'Sci-fi' }, '$db': 'test' },
  serverInfo: {
    host: 'pl-Vostro-3268',
    port: 27017,
    version: '7.0.23',
    gitVersion: '78d6d71385be23831b5971993af60bcafed785bc'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
```



```
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600,
    internalQueryFrameworkControl: 'forceClassicEngine'
  },
  ok: 1
}
test> db.books.dropIndex({title:1})
{ nIndexesWas: 3, ok: 1 }
test> 
```

Name: Kale Prathmesh
Class: TE Comp (A)   Roll no.:55
Subject: DBMSL   Practical no.:11

Output:

```
test> use populationDB
switched to db populationDB
populationDB> db.cities.insertMany([
...
...     { _id: 1, city: "CityA", district: "North", population: 150000 },
...
...     { _id: 2, city: "CityB", district: "South", population: 220000 },
...
...     { _id: 3, city: "CityC", district: "North", population: 80000 },
...
...     { _id: 4, city: "CityD", district: "West", population: 350000 },
...
...     { _id: 5, city: "CityE", district: "South", population: 130000 },
...
...     { _id: 6, city: "CityF", district: "North", population: 120000 }
...
... ]);
{
  acknowledged: true,
  insertedIds: { '0': 1, '1': 2, '2': 3, '3': 4, '4': 5, '5': 6 }
}
```

```
populationDB> var mapFunction = function() {
...
...     emit(this.district, { totalPopulation: this.population, count: 1 });
...
... };

populationDB> var reduceFunction = function(district, values) {
...
...     var result = { totalPopulation: 0, count: 0 };
...
...     values.forEach(function(value) {
...
...         result.totalPopulation += value.totalPopulation;
...
...         result.count += value.count;
...
...     });
...
...     return result;
...
... };

populationDB> var finalizeFunction = function(district, reducedValue) {
...
...     reducedValue.avgCityPopulation = reducedValue.totalPopulation / reducedValue.count;
...
...     return reducedValue;
...
... };
```

```
populationDB> db.cities.mapReduce(
...
...     mapFunction,
...
...     reduceFunction,
...
...     {
...
...         out: "district_population_stats",
...
...         finalize: finalizeFunction
...
...     }
...
... );
```

```
populationDB> db.district_population_stats.find().pretty();
[
  {
    _id: 'South',
    value: { totalPopulation: 350000, count: 2, avgCityPopulation: 175000 }
  },
  {
    _id: 'West',
    value: { totalPopulation: 350000, count: 1, avgCityPopulation: 350000 }
  },
  {
    _id: 'North',
    value: {
      totalPopulation: 350000,
      count: 3,
      avgCityPopulation: 116666.66666666667
    }
  }
]
```

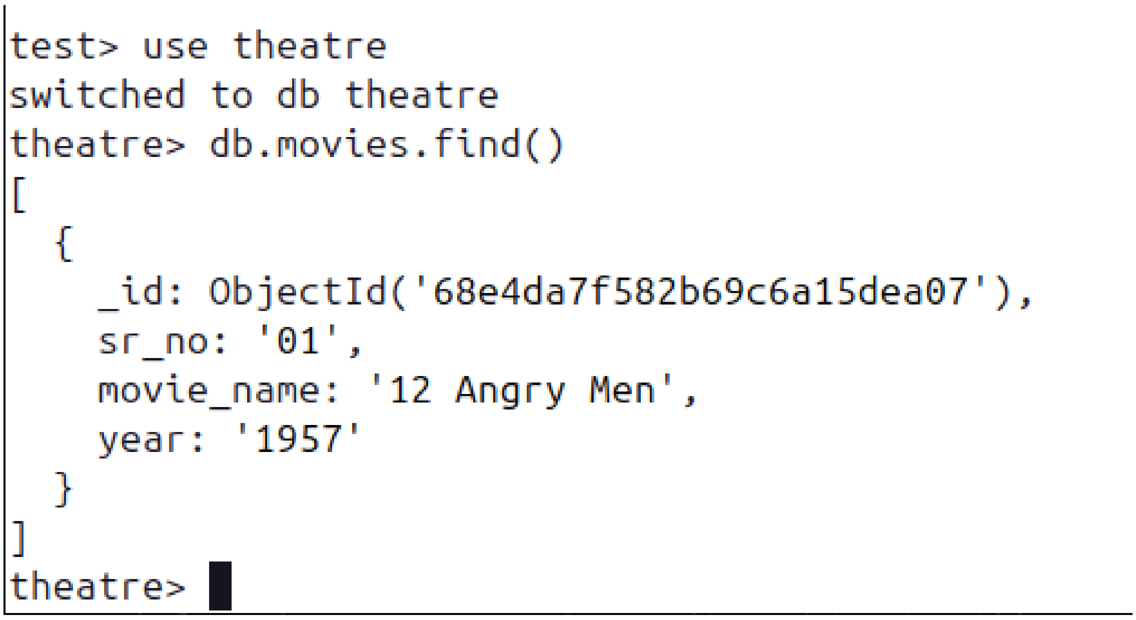
Name: Kale Prathmesh

Class: TE COMP (A)   Batch: A3

Practical no.:12

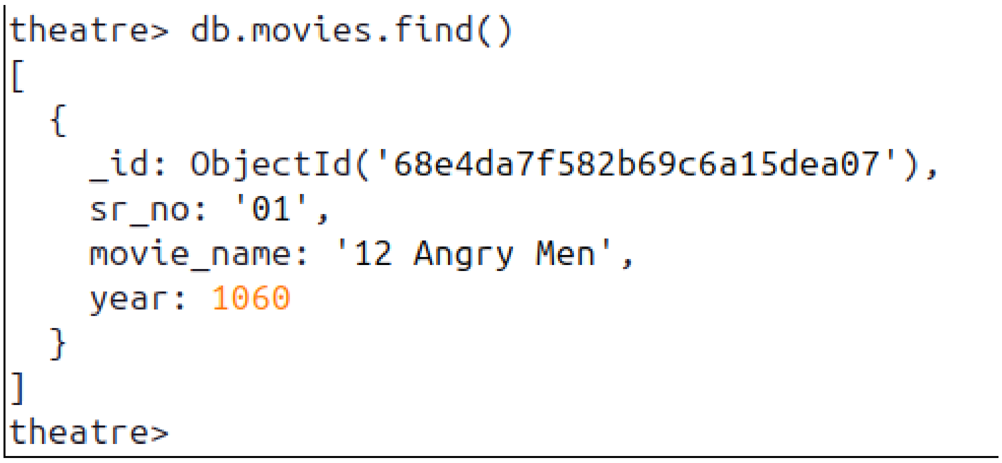
Program and Output:

```
from pymongo import MongoClient

client=MongoClient("mongodb://localhost:27017/")

database = client.theatre

collection=database.movies

#1st

collection1 ={

'sr_no':"01",

'movie_name':"12 Angry Men",

'year':"1957"

}

collection.insert_one(collection1)

print(collection.find_one())

print()
```



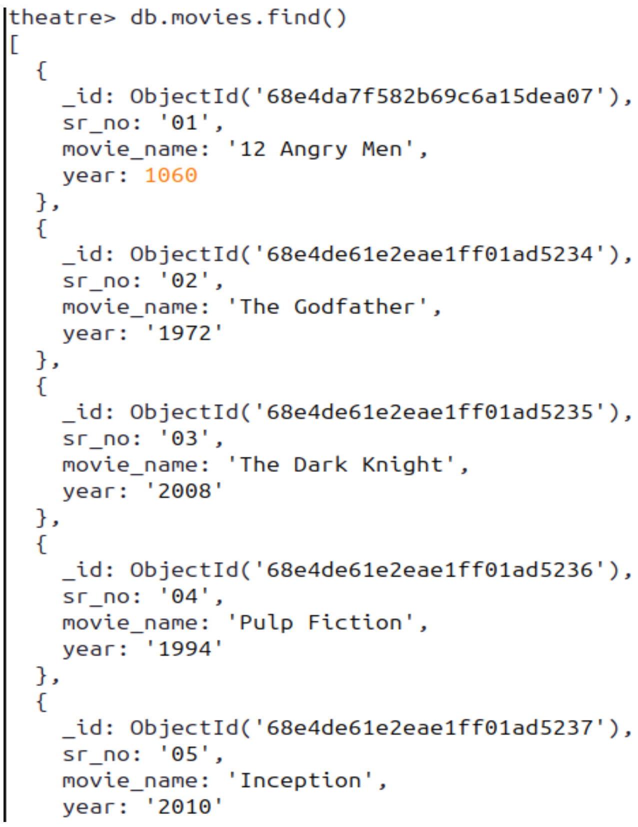
```
test> use theatre
switched to db theatre
theatre> db.movies.find()
[
  {
    _id: ObjectId('68e4da7f582b69c6a15dea07'),
    sr_no: '01',
    movie_name: '12 Angry Men',
    year: '1957'
  }
]
theatre> 
```

```
#2nd

 collection.update_one(

 {

 'sr_no': "01",
'movie_name': "12 Angry Men",
'year': "1957"
},
{
'$set': { 'year': 1060 }
}
)
collection.insert_one(collection1)
print(collection.find_one())
print()
```



```
theatre> db.movies.find()
[
  {
    _id: ObjectId('68e4da7f582b69c6a15dea07'),
    sr_no: '01',
    movie_name: '12 Angry Men',
    year: 1060
  }
]
theatre>
```

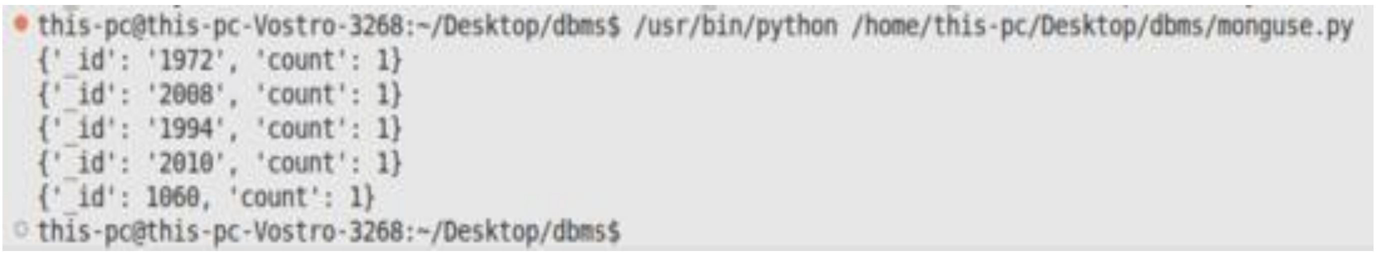
```
#3rd
result = collection.aggregate(pipeline)
documents = [
{'sr_no': "02", 'movie_name': "The Godfather", 'year': "1972"},
{'sr_no': "03", 'movie_name': "The Dark Knight", 'year': "2008"},
{'sr_no': "04", 'movie_name': "Pulp Fiction", 'year': "1994"},
{'sr_no': "05", 'movie_name': "Inception", 'year': "2010"}
]
insert_result = collection.insert_many(documents)
print("\nAll documents in 'movies' collection:")
for doc in collection.find():
print(doc)
```



```
theatre> db.movies.find()
[
  {
    _id: ObjectId('68e4da7f582b69c6a15dea07'),
    sr_no: '01',
    movie_name: '12 Angry Men',
    year: 1060
  },
  {
    _id: ObjectId('68e4de61e2eae1ff01ad5234'),
    sr_no: '02',
    movie_name: 'The Godfather',
    year: '1972'
  },
  {
    _id: ObjectId('68e4de61e2eae1ff01ad5235'),
    sr_no: '03',
    movie_name: 'The Dark Knight',
    year: '2008'
  },
  {
    _id: ObjectId('68e4de61e2eae1ff01ad5236'),
    sr_no: '04',
    movie_name: 'Pulp Fiction',
    year: '1994'
  },
  {
    _id: ObjectId('68e4de61e2eae1ff01ad5237'),
    sr_no: '05',
    movie_name: 'Inception',
    year: '2010'
```

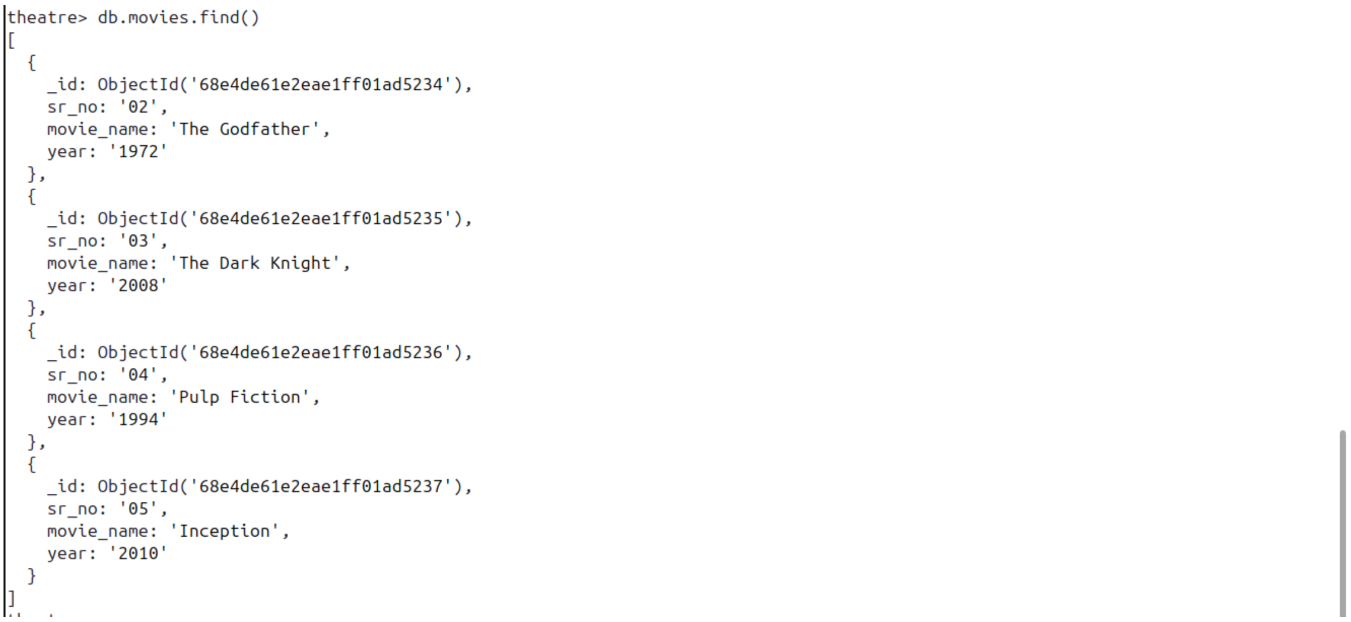
#4rd

```
pipeline = [

{

'$group': {

'_id': '$year', # Group by year

'count': {'$sum': 1} # Count the number of movies in each year

}

},

{

'$sort': {'count': -1} # Sort the results by count in descending order

}

]

print("\nAll documents in 'movies' collection:")

for doc in collection.find():

print(doc)
```


```
this-pc@this-pc-Vostro-3268:~/Desktop/dbms$ /usr/bin/python /home/this-pc/Desktop/dbms/monguse.py
{'_id': '1972', 'count': 1}
{'_id': '2008', 'count': 1}
{'_id': '1994', 'count': 1}
{'_id': '2010', 'count': 1}
{'_id': 1060, 'count': 1}
this-pc@this-pc-Vostro-3268:~/Desktop/dbms$
```

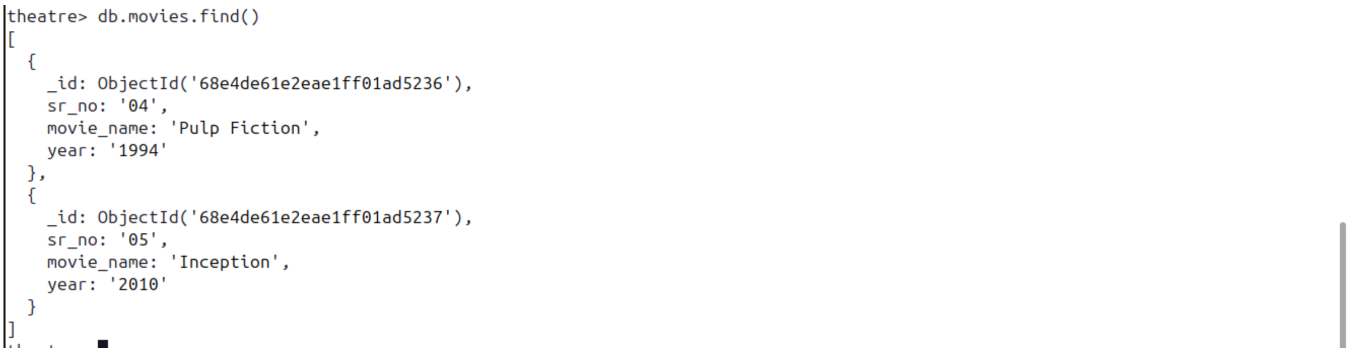
#5th

delete_result = collection.delete_one({'sr_no': '01'})


```
theatre> db.movies.find()
[
  {
    _id: ObjectId('68e4de61e2eae1ff01ad5234'),
    sr_no: '02',
    movie_name: 'The Godfather',
    year: '1972'
  },
  {
    _id: ObjectId('68e4de61e2eae1ff01ad5235'),
    sr_no: '03',
    movie_name: 'The Dark Knight',
    year: '2008'
  },
  {
    _id: ObjectId('68e4de61e2eae1ff01ad5236'),
    sr_no: '04',
    movie_name: 'Pulp Fiction',
    year: '1994'
  },
  {
    _id: ObjectId('68e4de61e2eae1ff01ad5237'),
    sr_no: '05',
    movie_name: 'Inception',
    year: '2010'
  }
]
```

# 6th

delete_result = collection.delete_many({'year': {'$in': ['1957', '1972', '2008']}})


```
theatre> db.movies.find()
[
  {
    _id: ObjectId('68e4de61e2eae1ff01ad5236'),
    sr_no: '04',
    movie_name: 'Pulp Fiction',
    year: '1994'
  },
  {
    _id: ObjectId('68e4de61e2eae1ff01ad5237'),
    sr_no: '05',
    movie_name: 'Inception',
    year: '2010'
  }
]
```