

DBMS END SEMESTER LAB EVALUATION

Praneesh R V

CB.SC.U4CYS23036

Scenario 1,

COURSES Table

```
CREATE TABLE COURSES (
  COURSEID INT,
  COURSENAME VARCHAR(50),
  DEPARTMENTID INT
);
```

Table COURSES created.

```
INSERT INTO COURSES (COURSEID,COURSENAME,DEPARTMENTID)
VALUES (201,'Algorithms',1);
INSERT INTO COURSES (COURSEID,COURSENAME,DEPARTMENTID)
VALUES (202,'Linear Algebra',2);
INSERT INTO COURSES (COURSEID,COURSENAME,DEPARTMENTID)
VALUES (203,'Quantum Physics',3);
```

```
SELECT * FROM COURSES;
```

All rows fetched: 3 in 0.064 seconds

	COURSEID	COURSENAME	DEPARTMENTID
1	201	Algorithms	1
2	202	Linear Algebra	2
3	203	Quantum Physics	3

DEPARTMENTS Table

```
Create TABLE DEPARTMENTS (
  DEPARTMENTID INT,
  DEPARTMENTNAME VARCHAR(50)
);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Table DEPARTMENTS created.

```
INSERT INTO DEPARTMENTS (DEPARTMENTID, DEPARTMENTNAME)
VALUES (1, 'Computer Science');
INSERT INTO DEPARTMENTS (DEPARTMENTID, DEPARTMENTNAME)
VALUES (2, 'Mathematics');
INSERT INTO DEPARTMENTS (DEPARTMENTID, DEPARTMENTNAME)
VALUES (3, 'Physics');
```

```
SELECT * FROM DEPARTMENTS;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR

All rows fetched: 3 in 0.012 seconds

	DEPARTMENTID		DEPARTMENTNAME
1		1	Computer Science
2		2	Mathematics
3		3	Physics

ENROLLMENTS Table

```
CREATE TABLE ENROLLMENTS (
  ENROLLMENTID INT,
  STUDENTID INT,
  COURSEID INT
)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Table ENROLLMENTS created.

```
INSERT INTO ENROLLMENTS (ENROLLMENTID,STUDENTID,COURSEID)
VALUES (1,101,201);
INSERT INTO ENROLLMENTS (ENROLLMENTID,STUDENTID,COURSEID)
VALUES (2,102,202);
INSERT INTO ENROLLMENTS (ENROLLMENTID,STUDENTID,COURSEID)
VALUES (3,103,203);
INSERT INTO ENROLLMENTS (ENROLLMENTID,STUDENTID,COURSEID)
VALUES (4,104,201);
```

SELECT * FROM ENROLLMENTS; Untitled-1

1 SELECT * FROM ENROLLMENTS;

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR COMMENTS GITLENS POSTMAN CONSOLE

All rows fetched: 4 in 0.017 seconds

	ENROLLMENTID	STUDENTID	COURSEID
1	1	101	201
2	2	102	202
3	3	103	203
4	4	104	201

Students table

```
CREATE TABLE STUDENTS (
  STUDENTID INT,
  STUDENTNAME VARCHAR(50),
  GPA DECIMAL(2,1),
  DEPARTMENTID INT
);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Table STUDENTS created.

```
INSERT INTO STUDENTS (STUDENTID,STUDENTNAME,GPA,DEPARTMENTID)
VALUES (101,'Alice',3.5,1);
INSERT INTO STUDENTS (STUDENTID,STUDENTNAME,GPA,DEPARTMENTID)
VALUES (102,'Bob',3.8,2);
INSERT INTO STUDENTS (STUDENTID,STUDENTNAME,GPA,DEPARTMENTID)
VALUES (103,'Charlie',2.9,3);
INSERT INTO STUDENTS (STUDENTID,STUDENTNAME,GPA,DEPARTMENTID)
VALUES (104,'Daisy',3.2,1);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

select * from STUDENTS; Untitled-1

```
1 select * from STUDENTS;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR COMMENTS GITLENS POSTMAN CONSOLE

All rows fetched: 4 in 0.027 seconds

	STUDENTID	STUDENTNAME	GPA	DEPARTMENTID
1	101	Alice	3.5	1
2	102	Bob	3.8	2
3	103	Charlie	2.9	3
4	104	Daisy	3.2	1

Qn 1, List the names of students along with the courses they are enrolled in using a LEFT JOIN.

SELECT STUDENTS.STUDENTNAME, COURSES.COURSENAME Untitled-1

```
1 SELECT STUDENTS.STUDENTNAME, COURSES.COURSENAME
2 FROM STUDENTS
3 LEFT JOIN COURSES
4 ON STUDENTS.departmentid = COURSES.departmentid;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR

All rows fetched: 4 in 0.018 seconds

	STUDENTNAME	COURSENAME
1	Daisy	Algorithms
2	Alice	Algorithms
3	Bob	Linear Algebra
4	Charlie	Quantum Physics

Qn2, Show all departments that have no students.

SELECT d.DEPARTMENTNAME Untitled-1

```
1 SELECT d.DEPARTMENTNAME
2 FROM DEPARTMENTS d
3 WHERE d.DEPARTMENTID NOT IN (SELECT s.DEPARTMENTID FROM STUDENTS s);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR

All rows fetched: 0 in 0.045 seconds

	DEPARTMENTNAME
--	----------------

Qn3, Retrieve students enrolled in the 'Algorithms' course using a subquery with a join.

```
SELECT s.STUDENTNAME
FROM STUDENTS s
WHERE s.DEPARTMENTID IN (
    SELECT c.DEPARTMENTID
    FROM COURSES c
    WHERE c.COURSENAME = 'Algorithms'
);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT

All rows fetched: 2 in 0.026 seconds

	STUDENTNAME
1	Daisy
2	Alice

Qn 4, Find courses where all enrolled students have a GPA higher than 3.0.

```
2  SELECT s.STUDENTNAME
3  FROM STUDENTS s
4  WHERE s.GPA > 3.0;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT S

All rows fetched: 3 in 0.068 seconds

	STUDENTNAME
1	Alice
2	Bob
3	Daisy

Qn 5, Find departments with fewer than 2 students enrolled.

```
SELECT d.DEPARTMENTNAME Untitled-1 ●
1  SELECT d.DEPARTMENTNAME
2  FROM DEPARTMENTS d
3  WHERE d.DEPARTMENTID IN (
4      SELECT s.DEPARTMENTID
5      FROM STUDENTS s
6      GROUP BY s.DEPARTMENTID
7      HAVING COUNT(*) < 2
8  );
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT S

All rows fetched: 2 in 0.031 seconds

	DEPARTMENTNAME
1	Mathematics
2	Physics

VIEWS

SCENARIO 2

AUTHORS Table

A screenshot of a code editor with a dark theme. The editor shows a file named 'Untitled-1' with a document icon. The code is written in SQL and is color-coded: 'CREATE TABLE' is orange, 'AUTHORS' is white, 'AUTHORID' is light blue, 'INT' is dark blue, 'AUTHORNAME' is white, 'VARCHAR' is light blue, and '(50)' is pink. The code is spread across four lines. Below the code editor, there are four tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is active, showing the message 'Table AUTHORS created.' in a monospaced font.

```
CREATE TABLE AUTHORS(  
  AUTHORID INT,  
  AUTHORNAME VARCHAR(50)  
);
```

Table AUTHORS created.

INSERT INTO AUTHORS(AUTHORID,AUTHORNAME) Untitled-1 ●

1

INSERT INTO AUTHORS(AUTHORID,AUTHORNAME)

2

VALUES (1,'George Orwell');

3

INSERT INTO AUTHORS(AUTHORID,AUTHORNAME)

4

VALUES (2,'J.K. Rowling');

5

INSERT INTO AUTHORS(AUTHORID,AUTHORNAME)

6

VALUES (3,'J.R.R Tolkien');

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS

1 row inserted.

1 row inserted.

1 row inserted.

SELECT * FROM AUTHORS; Untitled-1 ●

1 SELECT * FROM AUTHORS;

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR

All rows fetched: 3 in 0.017 seconds

	AUTHORID		AUTHORNAME
1		1	George Orwell
2		2	J.K. Rowling
3		3	J.R.R Tolkien

BOOKS Table

```
CREATE TABLE BOOKS(
  BOOKID INT,
  TITLE VARCHAR(200),
  AUTHORID INT,
  GENRE VARCHAR(50),
  AVAILABLECOIPES INT
)
```

Table BOOKS created.

```
INSERT INTO BOOKS(BOOKID, TITLE,AUTHORID,GENRE,AVAILABLECOIPES)
VALUES (1,'1984',1,'Dystopian',5);
INSERT INTO BOOKS(BOOKID, TITLE,AUTHORID,GENRE,AVAILABLECOIPES)
VALUES (2,'Harry Potter and the Sorcerers Stone',2,'Fantasy',10);
INSERT INTO BOOKS(BOOKID, TITLE,AUTHORID,GENRE,AVAILABLECOIPES)
VALUES (3,'The Hobbit',3,'Fantasy',3);
```

1 row inserted.

1 row inserted.

1 row inserted.

```
SELECT * FROM BOOKS;
```

All rows fetched: 3 in 0.029 seconds

	BOOKID	TITLE	AUTHORID	GENRE	AVAILABLECOIPES
1	1	1984	1	Dystopian	5
2	2	Harry Potter and the Sorcerers Stone	2	Fantasy	10
3	3	The Hobbit	3	Fantasy	3

LOANS Table

```
CREATE TABLE LOANS (
  LOANID INT,
  BOOKID INT,
  MEMBERID INT,
  RETURNSTATUS INT
)
```

PROBLEMS OUTPUT DEBUG CONSOLE T

Table LOANS created.

```
INSERT INTO LOANS (LOANID, BOOKID, MEMBE
Untitled-1 ●
1 INSERT INTO LOANS (LOANID, BOOKID, MEMBERID, RETURNSTATUS)
2 VALUES (1, 1, 1, 0);
3 INSERT INTO LOANS (LOANID, BOOKID, MEMBERID, RETURNSTATUS)
4 VALUES (2, 2, 2, 1);
5 INSERT INTO LOANS (LOANID, BOOKID, MEMBERID, RETURNSTATUS)
6 VALUES (3, 3, 3, 0);|
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT

1 row inserted.

1 row inserted.

1 row inserted.

SELECT * FROM LOANS; Untitled-1

1 SELECT * FROM LOANS;

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR COMMENTS GITLENS POSTMAN CONSOLE

All rows fetched: 3 in 0.051 seconds

	LOANID	BOOKID	MEMBERID	RETURNSTATUS
1	1	1	1	0
2	2	2	2	1
3	3	3	3	0

MEMBERS Table

```
CREATE TABLE MEMBERS(
  1 CREATE TABLE MEMBERS(
  2     MEMBERID INT,
  3     MEMBERNAME VARCHAR(50),
  4     MEMBERSHIPTYPE VARCHAR(50)
  5 )
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Table MEMBERS created.

INSERT INTO MEMBERS(MEMBERID, MEMBERNAME, MEMBERSHIPTYPE) •

1

2

3

4

5

6

INSERT INTO MEMBERS(MEMBERID, MEMBERNAME, MEMBERSHIPTYPE)

VALUES (1, 'Alice Smith', 'Gold');

INSERT INTO MEMBERS(MEMBERID, MEMBERNAME, MEMBERSHIPTYPE)

VALUES (2, 'Bob Johnson', 'Silver');

INSERT INTO MEMBERS(MEMBERID, MEMBERNAME, MEMBERSHIPTYPE)

VALUES (3, 'Charlie Brown', 'Gold');

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT

1 row inserted.

1 row inserted.

1 row inserted.

SELECT * FROM MEMBERS; Untitled-1 •

1 SELECT * FROM MEMBERS;

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR COMMENTS GITLENS

All rows fetched: 3 in 0.018 seconds

	MEMBERID		MEMBERNAME	MEMBERSHIPTYPE
1	1		Alice Smith	Gold
2	2		Bob Johnson	Silver
3	3		Charlie Brown	Gold

RESERVATIONS TABLE

```
CREATE TABLE RESERVATIONS(
  RESERVATIONID INT,
  BOOKID INT,
  MEMBERID INT,
  STATUS VARCHAR(20)
);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Table RESERVATIONS created.

```
INSERT INTO RESERVATIONS(RESERVATIONID,BOOKID,MEMBERID,STATUS)
VALUES (1,1,2,'Pending');
INSERT INTO RESERVATIONS(RESERVATIONID,BOOKID,MEMBERID,STATUS)
VALUES (2,2,1,'Completed');
INSERT INTO RESERVATIONS(RESERVATIONID,BOOKID,MEMBERID,STATUS)
VALUES (3,3,3,'Cancelled');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL H

1 row inserted.

1 row inserted.

1 row inserted.

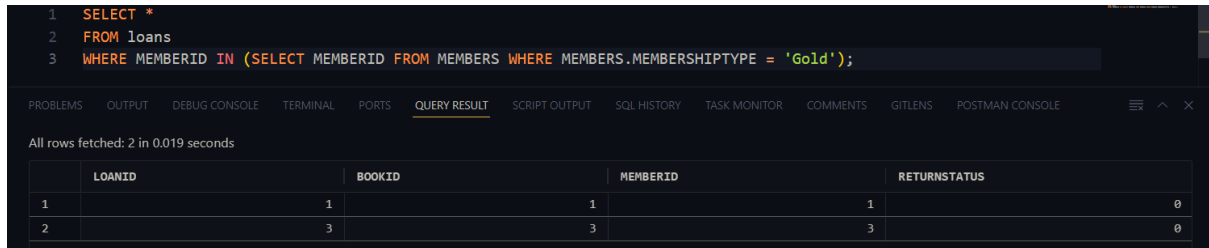
```
SELECT * FROM RESERVATIONS;
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR COMMENTS GITLENS POSTMAN CONSOLE

All rows fetched: 3 in 0.026 seconds

	RESERVATIONID	BOOKID	MEMBERID	STATUS
1	1	1	2	Pending
2	2	2	1	Completed
3	3	3	3	Cancelled

Qn1, Get the details of loans for members with 'Gold' membership.



The screenshot shows a SQL query in a dark-themed IDE. The query is: `SELECT * FROM loans WHERE MEMBERID IN (SELECT MEMBERID FROM MEMBERS WHERE MEMBERSHIPSTYPE = 'Gold');`. The results pane shows 2 rows fetched in 0.019 seconds. The table has columns: LOANID, BOOKID, MEMBERID, and RETURNSTATUS.

	LOANID	BOOKID	MEMBERID	RETURNSTATUS
1	1	1	1	0
2	3	3	3	0

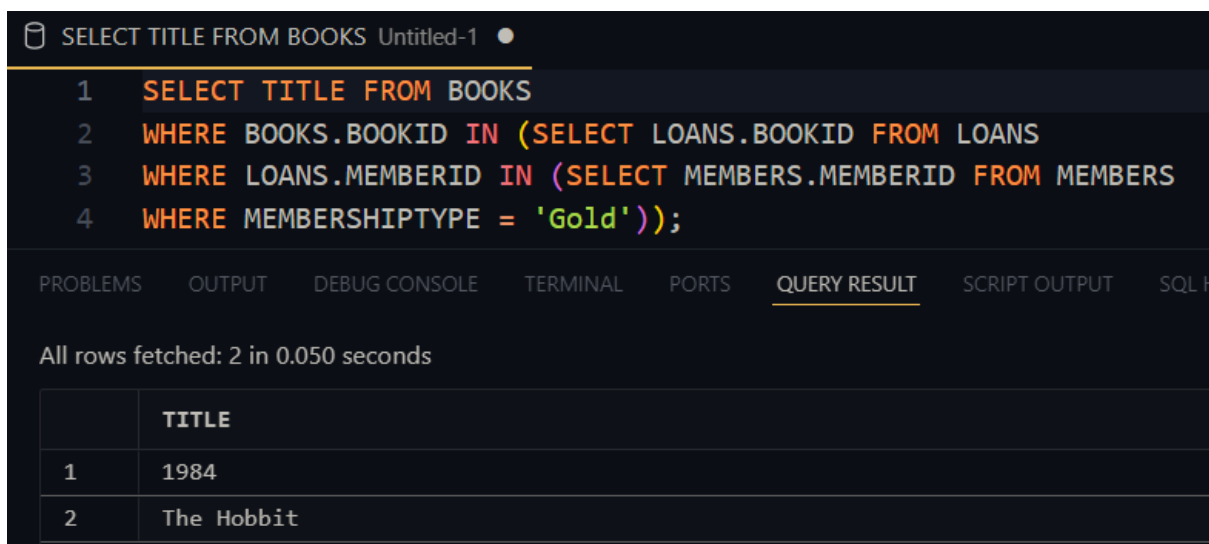
Qn2, Find all members who have a reservation for a book that is currently available.



The screenshot shows a SQL query in a dark-themed IDE. The query is: `SELECT m.MEMBERID, m.MEMBERNAME FROM MEMBERS m INNER JOIN RESERVATIONS r ON m.MEMBERID=r.MEMBERID WHERE r.STATUS='Pending' OR r.STATUS='Completed';`. The results pane shows 2 rows fetched in 0.083 seconds. The table has columns: MEMBERID and MEMBERNAME.

	MEMBERID	MEMBERNAME
1	1	Alice Smith
2	2	Bob Johnson

Qn3, Find books that have been loaned by 'Gold' members



The screenshot shows a SQL query in a dark-themed IDE. The query is: `SELECT TITLE FROM BOOKS WHERE BOOKS.BOOKID IN (SELECT LOANS.BOOKID FROM LOANS WHERE LOANS.MEMBERID IN (SELECT MEMBERS.MEMBERID FROM MEMBERS WHERE MEMBERSHIPSTYPE = 'Gold'));`. The results pane shows 2 rows fetched in 0.050 seconds. The table has columns: TITLE.

	TITLE
1	1984
2	The Hobbit

Qn4, Retrieve the members who have borrowed books that were also reserved by other members.

```
SELECT MEMBERS.MEMBERNAME
```

```
1 SELECT MEMBERS.MEMBERNAME
2 FROM MEMBERS
3 WHERE MEMBERS.MEMBERID IN (
4     SELECT LOANS.MEMBERID
5     FROM LOANS
6     WHERE LOANS.MEMBERID IN (
7         SELECT RESERVATIONS.MEMBERID
8         FROM RESERVATIONS
9     )
10 );
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT

All rows fetched: 3 in 0.113 seconds

	MEMBERNAME
1	Alice Smith
2	Bob Johnson
3	Charlie Brown

Qn 5, Show books that have been both reserved and loaned

```
SELECT *
```

```
1 SELECT *
2 FROM books b
3 WHERE b.bookID IN (SELECT bookID FROM reservations)
4 AND b.bookID IN (SELECT bookID FROM loans);
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULT SCRIPT OUTPUT SQL HISTORY TASK MONITOR COMMENTS GITLENS POSTMAN CONSOLE

All rows fetched: 3 in 0.020 seconds

	BOOKID	TITLE	AUTHORID	GENRE	AVAILABLECOPIES
1	1	1984	1	Dystopian	5
2	2	Harry Potter and the Sorcerers Stone	2	Fantasy	10
3	3	The Hobbit	3	Fantasy	3

Qn 6, Find the total number of books that are available and have not been reserved

```
SELECT SUM(BOOKS.AVAILABLECOPIES) -  
COUNT(RESERVATIONS.BOOKID);
```

VIEWS