

# **ADVANCED DATABASE MANAGEMENT SYSTEM**

**SUBMITTED BY,**

**RONY KURIAN**

**S2 MCA**

**ROLL NO: 44**

# CYCLE 1

## QUESTION SET 1

1. Create an employee table ‘EMP’ with following fields :  
empno NUMBER(2)  
ename VARCHAR2(25)  
job VARCHAR2(12)  
salary NUMBER(10,2)  
commission NUMBER(7,2)  
deptno NUMBER(2)

### CODE

```
CREATE TABLE emp
(
  EMPNO INT(2),
  ENAME VARCHAR(25),
  JOB VARCHAR(12),
  SALARY INT(10.2),
  COMMISSION INT(7.2),
  DEPTNO INT(2)
);
select * from emp;
```

### output

	EMPNO	ENAME	JOB	SALARY	COMMISSION	DEPTNO

2. Display the structure of ‘EMP

### Code

```
DESCRIBE EMP;
```

## Output

3 14:30:03 DESCRIBE EMP

3. Insert the following record into 'EMP'

**EMPNO ENAME JOB SAL COMM DEPTNO**

7369 SMITH CLERK 800 20

## Code

```
INSERT INTO EMP(EMPNO,ENAME,JOB,SALARY,COMMISSION,DEPTNO)VALUES  
(7369,'SMITH','CLERK',800,NULL,20);
```

## Output

4 14:33:42 INSERT INTO EMP(EMPNO,ENAME,JOB,SALARY,COMMISSION,DEPTNO)VALUES (7369,'... 1 row(s) affected

4. Insert the rest of records using substitution variable.

**EMPNO ENAME JOB SAL COMM DEPTNO**

7499 ALLEN SALESMAN 1600 300 30  
7521 WARD SALESMAN 1250 500 30  
7566 JONES MANAGER 2975 20  
7654 MARTIN SALESMAN 1250 1400 30  
7698 BLAKE MANAGER 2850 30  
7782 CLARK MANAGER 2450 10  
7788 SCOTT ANALYST 3000 20  
7839 KING PRESIDENT 5000 10  
7844 TURNER SALESMAN 1500 30  
7876 ADAMS CLERK 1100 20  
7900 JAMES NULL 950 30  
7902 FORD ANALYST 3000 20  
7934 MILLER CLERK 1300 10

## Code

```
INSERT INTO EMP(EMPNO,ENAME,JOB,SALARY,COMMISSION,DEPTNO)VALUES  
(7369,'SMITH','CLERK',800,NULL,20);  
INSERT INTO EMP(EMPNO,ENAME,JOB,SALARY,COMMISSION,DEPTNO)VALUES  
(7499,'ALLEN','SALESMAN',1600,300,30),  
(7521,'WARD','SALESMAN',1250,500,30),  
(7566,'JONES','MANAGER',2975,NULL,20),  
(7654,'MARTIN','SALESMAN',1250,1400,30),  
(7698,'BLAKE','MANAGER',2850,NULL,30),  
(7782,'CLARK','MANAGER',2450,NULL,10),  
(7788,'SCOTT','ANALYST',3000,NULL,20),  
(7839,'KING','PRESIDENT',5000,NULL,10),  
(7844,'TURNER','SALESMAN',1500,NULL,30),  
(7876,'ADAMS','CLERK',1100,NULL,20),  
(7900,'JAMES',NULL,950,NULL,30),  
(7902,'FORD','ANALYST',3000,NULL,20),  
(7934,'MILLER','CLERK',1300,NULL,10);  
select * from emp;
```

## Output

✓ 5 14:40:14 INSERT INTO EMP(EMPNO,ENAME,JOB,SALARY,COMMISSION,DEPTNO)VALUES (7499,... 13 row(s) affected Records

5. Insert job as 'CLERK' for all 'NULL' job types.

## Code

```
UPDATE EMP  
SET JOB='CLERK' WHERE JOB=NULL;  
SELECT * FROM EMP;
```

## Output

✓ 6 14:42:28 UPDATE EMP SET JOB='CLERK' WHERE JOB=NULL

0 row(s) affected Rows ma

6.Add a new field 'date\_join' with following values

date\_join  
17-DEC-80  
20-FEB-81  
22-FEB-81  
02-APR-81  
28-SEP-81  
01-MAY-81  
09-JUN-81  
19-APR-87

**17-NOV-81  
08-SEP-81  
23-MAY-87  
03-DEC-81  
03-DEC-81  
23-JAN-82**

## **Code**

```
ALTER TABLE EMP

ADD DATE_JOIN DATE NOT NULL;

DESCRIBE EMP;

UPDATE EMP

SET DATE_JOIN='1980-12-17' WHERE EMPNO='7369';

UPDATE EMP

SET DATE_JOIN='1981-02-20' WHERE EMPNO='7499';

UPDATE EMP

SET DATE_JOIN='1981-02-22' WHERE EMPNO='7521';

UPDATE EMP

SET DATE_JOIN='1981-04-02' WHERE EMPNO='7566';

UPDATE EMP

SET DATE_JOIN='1981-09-28' WHERE EMPNO='7654';

UPDATE EMP

SET DATE_JOIN='1981-05-01' WHERE EMPNO='7698';

UPDATE EMP

SET DATE_JOIN='1981-06-09' WHERE EMPNO='7782';

UPDATE EMP

SET DATE_JOIN='1987-04-19' WHERE EMPNO='7788';

UPDATE EMP

SET DATE_JOIN='1981-11-17' WHERE EMPNO='7839';

UPDATE EMP

SET DATE_JOIN='1981-09-08' WHERE EMPNO='7844';

UPDATE EMP

SET DATE_JOIN='1987-05-23' WHERE EMPNO='7876';

UPDATE EMP
```

```
SET DATE_JOIN='1981-12-03' WHERE EMPNO='7900';
UPDATE EMP

SET DATE_JOIN='1981-12-03' WHERE EMPNO='7902';
UPDATE EMP

SET DATE_JOIN='1982-01-23' WHERE EMPNO='7934';
```

## Output

✓	13	15:10:57	UPDATE EMP SET DATE_JOIN='1980-12-17' WHERE EMPNO='7369'	1 row(s) affected Rows matched
✓	14	15:10:57	UPDATE EMP SET DATE_JOIN='1981-02-20' WHERE EMPNO='7499'	1 row(s) affected Rows matched
✓	15	15:10:57	UPDATE EMP SET DATE_JOIN='1981-02-22' WHERE EMPNO='7521'	1 row(s) affected Rows matched
✓	16	15:10:57	UPDATE EMP SET DATE_JOIN='1981-04-02' WHERE EMPNO='7566'	1 row(s) affected Rows matched
✓	17	15:10:57	UPDATE EMP SET DATE_JOIN='1981-09-28' WHERE EMPNO='7654'	1 row(s) affected Rows matched
✓	18	15:10:57	UPDATE EMP SET DATE_JOIN='1981-05-01' WHERE EMPNO='7698'	1 row(s) affected Rows matched
✓	19	15:10:57	UPDATE EMP SET DATE_JOIN='1981-06-09' WHERE EMPNO='7782'	1 row(s) affected Rows matched
✓	20	15:10:57	UPDATE EMP SET DATE_JOIN='1987-04-19' WHERE EMPNO='7788'	1 row(s) affected Rows matched
✓	21	15:10:57	UPDATE EMP SET DATE_JOIN='1981-11-17' WHERE EMPNO='7839'	1 row(s) affected Rows matched
✓	22	15:10:57	UPDATE EMP SET DATE_JOIN='1981-09-08' WHERE EMPNO='7844'	1 row(s) affected Rows matched
✓	23	15:10:58	UPDATE EMP SET DATE_JOIN='1987-05-23' WHERE EMPNO='7876'	1 row(s) affected Rows matched
✓	24	15:10:58	UPDATE EMP SET DATE_JOIN='1981-12-03' WHERE EMPNO='7900'	1 row(s) affected Rows matched
✓	25	15:10:58	UPDATE EMP SET DATE_JOIN='1981-12-03' WHERE EMPNO='7902'	1 row(s) affected Rows matched
✓	26	15:10:58	UPDATE EMP SET DATE_JOIN='1982-01-23' WHERE EMPNO='7934'	1 row(s) affected Rows matched

## 7. Display details of all employees

### Code

```
SELECT * FROM EMP;
```

## Output

	EMPNO	ENAME	JOB	SALARY	COMMISSION	DEPTNO	DATE_JOIN
	7369	SMITH	CLERK	800	NULL	20	1980-12-17
	7499	ALLEN	SALESMAN	1600	300	30	1981-02-20
	7521	WARD	SALESMAN	1250	500	30	1981-02-22
	7566	JONES	MANAGER	2975	NULL	20	1981-04-02
	7654	MARTIN	SALESMAN	1250	1400	30	1981-09-28
	7698	BLAKE	MANAGER	2850	NULL	30	1981-05-01
	7782	CLARK	MANAGER	2450	NULL	10	1981-06-09
	7788	SCOTT	ANALYST	3000	NULL	20	1987-04-19
	7839	KING	PRESIDENT	5000	NULL	10	1981-11-17
	7844	TURNER	SALESMAN	1500	NULL	30	1981-09-08
	7876	ADAMS	CLERK	1100	NULL	20	1987-05-23
	7900	JAMES	NULL	950	NULL	30	1981-12-03
	7902	FORD	ANALYST	3000	NULL	20	1981-12-03
	7934	MILLER	CLERK	1300	NULL	10	1982-01-23

8. Display all the distinct job types in 'EMP'.

## Code

```
66 •      SELECT DISTINCT JOB FROM EMP;
```

Result Grid | Filter Rows: [ ] | Export

JOB
CLERK
SALESMAN
MANAGER
ANALYST
PRESIDENT

## Output

```
✓ 28 15:19:26 SELECT DISTINCT JOB FROM EMP LIMIT 0, 1000 6 row(s) returned
```

9. Display names of all employees in dept 20 and 30

```
67 • SELECT ENAME FROM EMP WHERE DEPTNO IN(20,30);
```

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

ENAME
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
SCOTT
TURNER
ADAMS
JAMES
FORD

10. List name and Total of salary i.e sal+commission

### Code

```
SELECT ENAME,SALARY+COMMISSION AS TOTAL_OF_SALARY  
FROM EMP;
```

### Output

```
62      SET DATE JOIN='1981-12-03' WHERE EMPNO='7902';
```

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

ENAME	TOTAL_OF_SALARY
SMITH	NULL
ALLEN	1900
WARD	1750
JONES	NULL
MARTIN	2650
BLAKE	NULL
CLARK	NULL
SCOTT	NULL
KING	NULL
TURNER	NULL
ADAMS	NULL
JAMES	NULL
FORD	NULL
MILLER	NULL

11. List name and Annual Salary i.e sal\*12

### Output

69 • `SELECT ENAME, SALARY*12 AS ANNUAL_SALARY FROM EMP;`

ENAME	ANNUAL_SALARY
SMITH	9600
ALLEN	19200
WARD	15000
JONES	35700
MARTIN	15000
BLAKE	34200
CLARK	29400
SCOTT	36000
KING	60000
TURNER	18000
ADAMS	13200
JAMES	11400
FORD	36000
MILLER	15600

12. List the employee who joined in the date '03-DEC-81'

## Output

70 • `SELECT ENAME FROM EMP WHERE DATE_JOIN='1981-12-03';`

ENAME
JAMES
FORD

13. Display the total salary of 'Miller'

## Output

```
71      SELECT ENAME, SALARY FROM EMP WHERE ENAME = "MILLER";
```

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

ENAME	SALARY
MILLER	1300

14.Delete the employee 'Miller' from'EMP'

## Output

```
72 •    DELETE FROM EMP WHERE ENAME='MILLER';
73 •    SELECT * FROM EMP;
```

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

EMPNO	ENAME	JOB	SALARY	COMMISSION	DEPTNO	DATE_JOIN
7369	SMITH	CLERK	800	NULL	20	1980-12-17
7499	ALLEN	SALESMAN	1600	300	30	1981-02-20
7521	WARD	SALESMAN	1250	500	30	1981-02-22
7566	JONES	MANAGER	2975	NULL	20	1981-04-02
7654	MARTIN	SALESMAN	1250	1400	30	1981-09-28
7698	BLAKE	MANAGER	2850	NULL	30	1981-05-01
7782	CLARK	MANAGER	2450	NULL	10	1981-06-09
7788	SCOTT	ANALYST	3000	NULL	20	1987-04-19
7839	KING	PRESIDENT	5000	NULL	10	1981-11-17
7844	TURNER	SALESMAN	1500	NULL	30	1981-09-08
7876	ADAMS	CLERK	1100	NULL	20	1987-05-23
7900	JAMES	NULL	950	NULL	30	1981-12-03
7902	FORD	ANALYST	3000	NULL	20	1981-12-03

15.Display name and deptno of all employees

## Output

74 • SELECT ENAME,DEPTNO FROM EMP ;

The screenshot shows a database query result grid. At the top, there is a toolbar with icons for 'Result Grid' (selected), 'Filter Rows' (with a search bar), 'Export' (with a file icon), and 'Wrap Cell C'. The main area displays a table with two columns: 'ENAME' and 'DEPTNO'. The data rows are as follows:

	ENAME	DEPTNO
1	SMITH	20
2	ALLEN	30
3	WARD	30
4	JONES	20
5	MARTIN	30
6	BLAKE	30
7	CLARK	10
8	SCOTT	20
9	KING	10
10	TURNER	30
11	ADAMS	20
12	JAMES	30
13	FORD	20

16. Remove the field 'commission' from 'EMP' after updating salary with total salary, i.e sal+commission

## Output

```

75 • ALTER TABLE EMP ADD TOTAL_SALARY INT
76 ✘ UPDATE EMP SET COMMISSION=0 WHERE COMMISSION IS NULL;
77 • UPDATE EMP SET TOTAL_SALARY=(SELECT SALARY+COMMISSION AS TOTAL_SALARY);
78 • SELECT * FROM EMP;

```

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

EMPNO	ENAME	JOB	SALARY	COMMISSION	DEPTNO	DATE_JOIN	TOTAL_SALAR
7369	SMITH	CLERK	800	0	20	1980-12-17	800
7499	ALLEN	SALESMAN	1600	300	30	1981-02-20	1900
7521	WARD	SALESMAN	1250	500	30	1981-02-22	1750
7566	JONES	MANAGER	2975	0	20	1981-04-02	2975
7654	MARTIN	SALESMAN	1250	1400	30	1981-09-28	2650
7698	BLAKE	MANAGER	2850	0	30	1981-05-01	2850
7782	CLARK	MANAGER	2450	0	10	1981-06-09	2450
7788	SCOTT	ANALYST	3000	0	20	1987-04-19	3000
7839	KING	PRESIDENT	5000	0	10	1981-11-17	5000
7844	TURNER	SALESMAN	1500	0	30	1981-09-08	1500
7876	ADAMS	CLERK	1100	0	20	1987-05-23	1100
7900	JAMES	NULL	950	0	30	1981-12-03	950
7902	FORD	ANALYST	3000	0	20	1981-12-03	3000

Result Grid | Form Editor | Field Types | Query Stats

17. Display the name of employees having the same amount of salary ( don't use subqueries )

## Output

```
82 •   SELECT TOTAL_SALARY, GROUP_CONCAT(ENAME ORDER BY DATE_JOIN) AS ENAME  
83     FROM EMP  
84     GROUP BY TOTAL_SALARY  
85     HAVING COUNT(*) > 1;
```

Result Grid		
	TOTAL_SALARY	ENAME
▶	3000	FORD,SCOTT

18. Display the name and employee no as 'name' and 'emp\_id'

## Output

```
86 •   SELECT EMPNO AS EMP_ID, ENAME AS 'NAME'
```

Result Grid		
	EMP_ID	NAME
▶	7369	SMITH
	7499	ALLEN
	7521	WARD
	7566	JONES
	7654	MARTIN
	7698	BLAKE
	7782	CLARK
	7788	SCOTT
	7839	KING
	7844	TURNER
	7876	ADAMS
	7900	JAMES
	7902	FORD

19. Rename table 'EMP' to 'EMPLOYEE'

## Output

```
ALTER TABLE EMP
RENAME TO EMPLOYEE;
CREATE TABLE EMP_TABLE AS SELECT * FROM EMPLOYEE;
```

20.Create a new table ‘EMP\_TAB’ from table ‘EMPLOYEE

### Output

```
88 • ALTER TABLE EMP
89      RENAME TO EMPLOYEE;
90 • CREATE TABLE EMP_TABLE AS SELECT * FROM EMPLOYEE;
91 • DESCRIBE EMP_TABLE;
92 • SELECT * FROM EMP_TABLE;
93
```

EMPNO	ENAME	JOB	SALARY	DEPTNO	DATE_JOIN	TOTAL_SALARY
7369	SMITH	CLERK	800	20	1980-12-17	800
7499	ALLEN	SALESMAN	1600	30	1981-02-20	1900
7521	WARD	SALESMAN	1250	30	1981-02-22	1750
7566	JONES	MANAGER	2975	20	1981-04-02	2975
7654	MARTIN	SALESMAN	1250	30	1981-09-28	2650
7698	BLAKE	MANAGER	2850	30	1981-05-01	2850
7782	CLARK	MANAGER	2450	10	1981-06-09	2450
7788	SCOTT	ANALYST	3000	20	1987-04-19	3000
7839	KING	PRESIDENT	5000	10	1981-11-17	5000
7844	TURNER	SALESMAN	1500	30	1981-09-08	1500
7876	ADAMS	CLERK	1100	20	1987-05-23	1100
7900	JAMES	NULL	950	30	1981-12-03	950
7902	FORD	ANALYST	3000	20	1981-12-03	3000

21.List all the details of ‘EMPLOYEE’ and ‘EMP\_TAB’

### Output

```
▶ SELECT * FROM EMP_TABLE;  
▶ SELECT * FROM EMP_TABLE FULL JOIN EMPLOYEE;
```

The screenshot shows a database query results grid. At the top, there are buttons for 'Grid' (selected), 'Filter Rows', 'Export' (with icons for CSV and XML), and 'Wrap Cell Content'. The grid displays data from two tables: EMP\_TABLE and EMPLOYEE. The columns are: EMPNO, ENAME, JOB, SALARY, DEPTNO, DATE\_JOIN, TOTAL\_SALARY, and EMPNO. The data rows are:

EMPNO	ENAME	JOB	SALARY	DEPTNO	DATE_JOIN	TOTAL_SALARY	EMPNO
70	JAMES	NULL	950	30	1981-12-03	950	7369
76	ADAMS	CLERK	1100	20	1987-05-23	1100	7369
44	TURNER	SALESMAN	1500	30	1981-09-08	1500	7369
39	KING	PRESIDENT	5000	10	1981-11-17	5000	7369
38	SCOTT	ANALYST	3000	20	1987-04-19	3000	7369
32	CLARK	MANAGER	2450	10	1981-06-09	2450	7369

22.Delete all records from 'EMP'

### Output

```
94 • TRUNCATE TABLE EMPLOYEE;  
95 • SELECT * FROM EMPLOYEE;  
96  
97  
98
```

The screenshot shows a database query results grid. At the top, there are buttons for 'result Grid' (selected), 'Filter Rows', 'Export' (with icons for CSV and XML), and 'Wrap Cell Content'. The grid has a header row with columns: EMPNO, ENAME, JOB, SALARY, DEPTNO, DATE\_JOIN, and TOTAL\_SALARY. There are no data rows present.

23.Delete the table 'EMP'

### Output

```
96 •  DROP TABLE EMPLOYEE;  
97  
98  
99  
100  
101
```

Output :

Action Output		
#	Time	Action
55	14:25:58	SELECT * FROM EMP_TABLE LIMIT 0, 1000
56	14:28:48	SELECT * FROM EMP_TABLE FULL JOIN EMPLOYEE LIMIT 0, 1000
57	14:36:30	TRUNCATE TABLE EMPLOYEE
58	14:36:31	SELECT * FROM EMPLOYEE LIMIT 0, 1000
59	14:37:11	TRUNCATE TABLE EMPLOYEE
60	14:37:12	SELECT * FROM EMPLOYEE LIMIT 0, 1000
61	14:38:06	DROP TABLE EMPLOYEE

## SET 2

1. Display the id and name of youngest student.

CODE

```
SELECT SID,SNAME AS YOUNGEST_STUDENT,DOB as DATE_OF_BIRTH FROM STUDENT  
WHERE DOB=(SELECT MAX(DOB) FROM STUDENT);
```

OUTPUT

The screenshot shows the MySQL Workbench interface with a query editor window titled "SQL File 1". The code entered is:

```

15  ('SONY','1998-02-15',29,45,60),
16  ('SUBIN','1997-05-25',70,88,89),
17  ('JASMINE','2006-01-30',56,88,66);
18 • SELECT * FROM STUDENT;
19 • SELECT * FROM STUDENT WHERE Maths >=40 AND (Physics >=40 OR Chemistry >=40);
20 • ALTER TABLE STUDENT ADD TOTAL INT;
21 • ALTER TABLE STUDENT ADD AVERAGE FLOAT;
22 • DESCUTE STUDENT.

```

The result grid shows one row of data:

SID	SNAME	DOB	Physics	Chemistry	Maths
5	JASMINE	2006-01-30			

2. Display the details of students who have passed in maths and either in physics or chemistry.(pass mark = 40 marks and above)

## CODE

```
SELECT * FROM STUDENT WHERE Maths >=40 AND (Physics >=40 OR Chemistry >=40);
```

## Output:

The screenshot shows the MySQL Workbench interface with a query editor window titled "SQL File 1". The code entered is identical to the previous screenshot.

The result grid shows five rows of data:

SID	SNAME	DOB	Physics	Chemistry	Maths
1	JILJO	2000-07-08	50	85	80
2	NOAMI	1999-06-03	40	70	70
3	SONY	1998-02-15	29	45	60
4	SUBIN	1997-05-25	70	88	89
5	JASMINE	2006-01-30	56	88	66

3.Add two more columns total and average.

```

MySQL Workbench
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
    Filter objects
    company
    customers
    cycle2
    employee
    employees
        Tables
        Views
        Stored Procedures
        Functions
    lost_accounts
    rony
        Tables
        Views
        Stored Procedures
        Functions
Administration Schemas
Information
Schema: rony
Object Info Session

SQL File 1...x
16  (4,'SUBIN','1997-05-25',70,88,89),
17  (5,'JASHINE',2006-01-30',58,88,66);
18  • SELECT * FROM STUDENT;
19  • SELECT SID,SNAME AS YOUNGEST_STUDENT,DOB as DATE_OF_BIRTH FROM STUDENT WHERE DOB=(SELECT MAX(DOB) FROM STUDENT);
20  • SELECT * FROM STUDENT WHERE Maths >=40 AND (Physics >=40 OR Chemistry >=40);
21  • ALTER TABLE STUDENT ADD TOTAL INT;
22  • ALTER TABLE STUDENT ADD AVERAGE FLOAT;
23  • DESCRIBE STUDENT;
24  • SELECT CHARS COUNT STUDENT WHERE Maths>=40 AND Physics>=40 AND Chemistry>=40;

Result Grid
Field Type Null Key Default Extra
SID int YES NULL
SNAME varchar(25) YES NULL
DOB date YES NULL
Physics int YES NULL
Chemistry int YES NULL
Maths int YES NULL
TOTAL int YES NULL
AVERAGE float YES NULL

Result 5 x
Output
Action Output
# Time Action Message Duration / Fetch
14 17:04:10 SELECT * FROM STUDENT LIMIT 0, 1000 5 row(s) returned 0.062 sec / 0.000 sec
15 17:04:16 SELECT SID,SNAME AS YOUNGEST_STUDENT,DOB as DATE_OF_BIRTH FROM STUDENT WHERE D... 1 row(s) returned 0.140 sec / 0.000 sec
16 17:06:39 SELECT * FROM STUDENT WHERE Maths >=40 AND (Physics >=40 OR Chemistry >=40) LIMIT 0, 1000 5 row(s) returned 0.094 sec / 0.000 sec
17 17:07:21 ALTER TABLE STUDENT ADD TOTAL INT 0 rows affected Records: 0 Duplicates: 0 Warnings: 0 0.719 sec
18 17:07:25 ALTER TABLE STUDENT ADD AVERAGE FLOAT 0 rows affected Records: 0 Duplicates: 0 Warnings: 0 1.250 sec
19 17:07:46 DESCRIBE STUDENT 8 row(s) returned 0.000 sec / 0.000 sec

```

4. Display the name of student who scored highest marks in maths.

**Code:**

```
SELECT SNAME FROM STUDENT WHERE Maths=(SELECT MAX(Maths) FROM STUDENT);
```

**Output:**

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, there is a large multi-line query. In the Results Grid tab, the output of the query is displayed, showing a single row with columns SNAME and SUBN, both containing the value 'SONY'. The status bar at the bottom indicates 'Read Only'.

```

17 (S,'JASLINE', '2006-01-30',50,88,66);
18 • SELECT * FROM STUDENT;
19 • SELECT SID,SNAME AS YOUNGEST_STUDENT,DOB AS DATE_OF_BIRTH FROM STUDENT WHERE DOB=(SELECT MAX(DOB) FROM STUDENT);
20 • SELECT * FROM STUDENT WHERE Maths >=40 AND (Physics >=40 OR Chemistry >=40);
21 • ALTER TABLE STUDENT ADD TOTAL INT;
22 • ALTER TABLE STUDENT ADD AVERAGE FLOAT;
23 • DESCRIBE STUDENT;
24 • SELECT SNAME FROM STUDENT WHERE Maths=(SELECT MAX(Maths) FROM STUDENT);
25 • SELECT SNAME FROM STUDENT WHERE Chemistry=(SELECT MIN(Chemistry) FROM STUDENT);
26 • UPDATE STUDENT SET Total=Physics+Chemistry+Maths;

```

5. Display the name of student who scored least marks in chemistry.

**Code:**

```
SELECT SNAME FROM STUDENT WHERE Chemistry=(SELECT MIN(Chemistry)
FROM STUDENT);
```

**Output:**

The screenshot shows the MySQL Workbench interface. In the SQL Editor tab, there is a query. In the Output tab, the execution log is shown, detailing each step of the query execution. The log includes actions like selecting from STUDENT, altering the STUDENT table to add columns, describing the STUDENT table, selecting the maximum Maths mark, selecting the minimum Chemistry mark, and finally selecting the student name where Chemistry equals the minimum Chemistry mark. The log also shows the number of rows affected, duplicates, warnings, and execution duration for each step.

#	Action	Time	Message	Duration / Fetch
16	SELECT * FROM STUDENT WHERE Maths >=40 AND (Physics >=40 OR Chemistry >=40) LIMIT 0, 1000	17:06:39	5 rows returned	0.094 sec / 0.000 sec
17	ALTER TABLE STUDENT ADD TOTAL INT	17:07:21	0 rows affected, Records: 0 Duplicates: 0 Warnings: 0	0.719 sec
18	ALTER TABLE STUDENT ADD AVERAGE FLOAT	17:07:25	0 rows affected, Records: 0 Duplicates: 0 Warnings: 0	1.250 sec
19	DESCRIBE STUDENT	17:07:46	8 rows returned	0.000 sec / 0.000 sec
20	SELECT SNAME FROM STUDENT WHERE Maths=(SELECT MAX(Maths) FROM STUDENT) LIMIT 0, 1000	17:09:02	1 rows returned	0.110 sec / 0.000 sec
21	SELECT SNAME FROM STUDENT WHERE Chemistry=(SELECT MIN(Chemistry) FROM STUDENT) LIMIT 0, 1000	17:11:05	1 rows returned	0.125 sec / 0.000 sec

6. Update column total with total marks.

**Code:**

**UPDATE STUDENT SET Total=Physics+Chemistry+Maths;**

**SELECT \* FROM STUDENT;**

**Output:**

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure, including tables like company, customers, cycle2, employee, employees, and rony.
- SQL File 1:** Contains the following SQL code:

```
20 • SELECT * FROM STUDENT WHERE Maths >=40 AND (Physics >=40 OR Chemistry >=40);
21 • ALTER TABLE STUDENT ADD AVERAGE FLOAT;
22 • DESCRIBE STUDENT;
23 • SELECT SNAME FROM STUDENT WHERE Maths=(SELECT MAX(Maths) FROM STUDENT);
24 • SELECT SNAME FROM STUDENT WHERE Chemistry=(SELECT MIN(Chemistry) FROM STUDENT);
25 • UPDATE STUDENT SET Total=Physics+Chemistry+Maths;
26 • SELECT * FROM STUDENT;
27 • UPDATE STUDENT SET AVERAGE=TOTAL/3;
```
- Result Grid:** Displays the results of the SELECT query, showing student details and the calculated average.
- Output:** Shows the log of actions taken, including the creation of the AVERAGE column and the execution of the UPDATE statements.

**7. Display details of students in order of total merit.**

**Code:**

**UPDATE STUDENT SET AVERAGE=TOTAL/3;**

**SELECT \* FROM STUDENT;**

**Output:**

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema structure, including tables like company, customers, cycle2, employee, employees, and rony.
- SQL File 1:** Contains the following SQL code:

```
22 • ALTER TABLE STUDENT ADD AVERAGE FLOAT;
23 • DESCRIBE STUDENT;
24 • SELECT SNAME FROM STUDENT WHERE Maths=(SELECT MAX(Maths) FROM STUDENT);
25 • SELECT SNAME FROM STUDENT WHERE Chemistry=(SELECT MIN(Chemistry) FROM STUDENT);
26 • UPDATE STUDENT SET Total=Physics+Chemistry+Maths;
27 • SELECT * FROM STUDENT;
28 • UPDATE STUDENT SET AVERAGE=TOTAL/3;
29 • SELECT * FROM STUDENT;
30 • ALTER TABLE STUDENT REMOVE AVERAGE TO AVG_MATHS;
```
- Result Grid:** Displays the results of the SELECT query, showing student details and the calculated average.
- Output:** Shows the log of actions taken, including the creation of the AVERAGE column and the execution of the UPDATE statements.

## 8. Rename the column average with avg\_mark

### Code:

```
ALTER TABLE STUDENT RENAME COLUMN AVERAGE TO  
AVG_MARK;
```

```
SELECT * FROM STUDENT;
```

### Output:

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Navigator' shows the schema 'rony' with tables like company, customers, cycle2, employee, and STUDENT. The 'SQL File 1' tab contains the following SQL code:

```
24 • SELECT SNAME FROM STUDENT WHERE Maths=(SELECT MAX(Maths) FROM STUDENT);  
25 • SELECT SNAME FROM STUDENT WHERE Chemistry=(SELECT MIN(Chemistry) FROM STUDENT);  
26 • UPDATE STUDENT SET Total=Physics+Chemistry+Maths;  
27 • SELECT * FROM STUDENT;  
28 • UPDATE STUDENT SET AVERAGE=TOTAL/3;  
29 • SELECT * FROM STUDENT;  
30 • ALTER TABLE STUDENT RENAME COLUMN AVERAGE TO AVG_MARK;  
31 • SELECT * FROM STUDENT;
```

The 'Result Grid' pane displays the following data from the STUDENT table:

SID	SNAME	DOB	Physics	Chemistry	Maths	TOTAL	AVG_MARK
1	JLJO	2000-07-08	50	85	80	215	71.6667
2	NOAMI	1999-06-03	40	70	70	180	60
3	SONY	1998-02-15	29	45	60	134	44.6667
4	SUBIN	1997-05-25	70	88	89	247	82.3333
5	JASMINE	2006-01-30	50	88	66	204	68

The 'Output' pane at the bottom shows the execution log:

Action	Time	Message	Duration / Fetch
22	17:13:36	UPDATE STUDENT SET Total=Physics+Chemistry+Maths	0.188 sec
23	17:13:40	SELECT * FROM STUDENT LIMIT 0, 1000	0.094 sec / 0.000 sec
24	17:15:38	UPDATE STUDENT SET AVERAGE=TOTAL/3	0.141 sec
25	17:15:41	SELECT * FROM STUDENT LIMIT 0, 1000	0.062 sec / 0.000 sec
26	17:17:36	ALTER TABLE STUDENT RENAME COLUMN AVERAGE TO AVG_MARK	1.454 sec
27	17:17:40	SELECT * FROM STUDENT LIMIT 0, 1000	0.016 sec / 0.000 sec

## 9. Find out the overall average of class.

### Code:

```
SELECT AVG(AVG_MARK) AS OVERALL_AVERAGE FROM  
STUDENT;
```

### Output:

The screenshot shows the MySQL Workbench interface. In the left sidebar, under the 'Schemas' section, the 'employees' schema is selected. A query editor window titled 'SQL File 1' contains the following SQL code:

```
25 • SELECT * FROM STUDENT WHERE Chemistry > (SELECT MIN(Chemistry) FROM STUDENT);
26 • UPDATE STUDENT SET Total=Physics+Chemistry+Maths;
27 • SELECT * FROM STUDENT;
28 • UPDATE STUDENT SET AVERAGE=TOTAL/3;
29 • SELECT * FROM STUDENT;
30 • ALTER TABLE STUDENT RENAME COLUMN AVERAGE TO AVG_MARK;
31 • SELECT * FROM STUDENT;
32 • SELECT AVG(AVG_MARK) AS OVERALL_AVERAGE FROM STUDENT;
33 • SELECT * FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT);
```

The results pane shows a single row with the value 'OVERALL\_AVERAGE' followed by '65.3333358764648'. Below the results is a table titled 'Action Output' showing the execution history of the statements:

#	Time	Action	Message	Duration / Fetch
23	17:13:40	SELECT * FROM STUDENT LIMIT 0, 1000	5 row(s) returned	0.094 sec / 0.000 sec
24	17:15:38	UPDATE STUDENT SET AVERAGE=TOTAL/3	5 row(s) affected Rows matched: 5 Changed: 5 Warnings: 0	0.141 sec
25	17:15:41	SELECT * FROM STUDENT LIMIT 0, 1000	5 row(s) returned	0.052 sec / 0.000 sec
26	17:17:36	ALTER TABLE STUDENT RENAME COLUMN AVERAGE TO AVG_MARK	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	1.454 sec
27	17:17:40	SELECT * FROM STUDENT LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
28	17:18:37	SELECT AVG(AVG_MARK) AS OVERALL_AVERAGE FROM STUDENT LIMIT 0, 1000	1 row(s) returned	0.031 sec / 0.000 sec

10. Display details of students whose average is greater than overall average.

### Code:

```
SELECT * FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT);
```

### Output:

The screenshot shows the MySQL Workbench interface with a SQL editor containing the following code:

```

26 • UPDATE STUDENT SET Total=Physics+Chemistry+Maths;
27 • SELECT * FROM STUDENT;
28 • UPDATE STUDENT SET AVERAGE=TOTAL/3;
29 • SELECT * FROM STUDENT;
30 • ALTER TABLE STUDENT RENAME COLUMN AVERAGE TO AVG_MARK;
31 • SELECT * FROM STUDENT;
32 • SELECT AVG(AVG_MARK) AS OVERALL_AVERAGE FROM STUDENT;
33 • SELECT *FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT);
34 • SELECT COUNT(SNAME) FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT);

```

The Result Grid shows the following data:

SID	SNAME	DOB	Physics	Chemistry	Maths	TOTAL	AVG_MARK
1	JILJO	2000-07-08	50	85	80	215	71.6667
4	SUBIN	1997-05-25	70	88	89	247	82.3333
5	JASMINE	2006-01-30	50	88	66	204	68

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
24	17:15:38	UPDATE STUDENT SET AVERAGE=TOTAL/3	5 row(s) affected Rows matched: 5 Changed: 5 Warnings: 0	0.141 sec
25	17:15:41	SELECT * FROM STUDENT	5 row(s) returned	0.062 sec / 0.000 sec
26	17:17:36	ALTER TABLE STUDENT RENAME COLUMN AVERAGE TO AVG_MARK	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	1.454 sec
27	17:17:40	SELECT * FROM STUDENT LIMIT 0, 1000	5 row(s) returned	0.016 sec / 0.000 sec
28	17:18:37	SELECT AVG(AVG_MARK) AS OVERALL_AVERAGE FROM STUDENT LIMIT 0, 1000	1 row(s) returned	0.031 sec / 0.000 sec
29	17:19:31	SELECT *FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT) LIMIT 0, 1000	3 row(s) returned	0.015 sec / 0.000 sec
30	17:20:27	SELECT COUNT(SNAME) FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT)	1 row(s) returned	0.000 sec / 0.000 sec

11. Find the total no: of students whose average is greater than overall average.

### Code:

```
SELECT COUNT(SNAME) FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT);
```

### Output:

The screenshot shows the MySQL Workbench interface with a SQL editor containing the following code:

```

27 • SELECT * FROM STUDENT;
28 • UPDATE STUDENT SET AVERAGE=TOTAL/3;
29 • SELECT * FROM STUDENT;
30 • ALTER TABLE STUDENT RENAME COLUMN AVERAGE TO AVG_MARK;
31 • SELECT * FROM STUDENT;
32 • SELECT AVG(AVG_MARK) AS OVERALL_AVERAGE FROM STUDENT;
33 • SELECT *FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT);
34 • SELECT COUNT(SNAME) FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT);

```

The Result Grid shows the following data:

COUNT(SNAME)
3

The Output pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
25	17:15:41	SELECT * FROM STUDENT	5 row(s) returned	0.062 sec / 0.000 sec
26	17:17:36	ALTER TABLE STUDENT RENAME COLUMN AVERAGE TO AVG_MARK	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	1.454 sec
27	17:17:40	SELECT * FROM STUDENT	5 row(s) returned	0.016 sec / 0.000 sec
28	17:18:37	SELECT AVG(AVG_MARK) AS OVERALL_AVERAGE FROM STUDENT	1 row(s) returned	0.031 sec / 0.000 sec
29	17:19:31	SELECT *FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT)	3 row(s) returned	0.015 sec / 0.000 sec
30	17:20:27	SELECT COUNT(SNAME) FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM STUDENT)	1 row(s) returned	0.000 sec / 0.000 sec

## **SET 3**

Create the Table LOAN\_ACCOUNTS with the structure given below

Field Name Data Type Length

Accno CHAR 4

Cust\_name VARCHAR2 15

Loan\_Amount NUMBER 7 digits and 2

decimal places

Installments NUMBER

int\_rate NUMBER 2 digits and 2

decimal places

Start\_date DATE

Interest NUMBER 7 digits and 2

decimal places

Add another column ‘category’ of type varchar2(1) in the Loan Table.

```

2 • CREATE TABLE LOAN_ACCOUNTS
3   (
4     ACCNO INT,
5     CUST_NAME VARCHAR(15),
6     LOAN_AMOUNT DECIMAL(8,2),
7     INSTALLMENTS INT,
8     INT_RATE DECIMAL(4,2),
9     START_DATE DATE,
10    INTEREST DECIMAL(8,2)
11  );
12 • ALTER TABLE LOAN_ACCOUNTS
13   ADD COLUMN CATEGORY VARCHAR(1);

```

manually

Action Output			Message
#	Time	Action	
105	18:51:19	create database LOAN_ACCOUNTS	1 row(s) affected
106	18:55:05	CREATE TABLE LOAN_ACCOUNTS ( ACCNO INT, CUST_NAME VARCHAR(15), LOAN_...	0 row(s) affected
107	19:02:17	ALTER TABLE LOAN_ACCOUNTS ADD COLUMN CATEGORY VARCHAR(1)	0 row(s) affected Records: 0 Duration: 00:00:00.000

Insert the following details into the table

Accno Cust\_name

e

Loan\_Amount

t

Installment

s

int\_rate Start\_date Interest

1001 R.K Gupta 300,000.00 36 12.00 July 19, 2009

1002 S.P

Sharma

500,000.00 48 10.00 March 22, 2008

1003 K.P Jain 300,000.00 36 NULL August 3, 2007

1004 M.P

Yadav  
 800,000.00 60 10.00 June 12,  
 2008  
 1005 S.P Sinha 200,000.00 36 12.50 March 1,  
 2010  
 1006 P. Sharma 700,000.00 60 12.50 May 6,  
 2008  
 1007 K.S Dhall 500,000.00 48 NULL May 3,  
 2008

```

14 • INSERT INTO LOAN_ACCOUNTS(ACCNO,CUST_NAME,LOAN_AMOUNT,INSTALLMENTS,INT_RATE
15     (1001,'R.K GUPTA',300000.00,36,12.00,'2009-07-19'),
16     (1002,'S.P SHARMA',500000.00,48,10.00,'2008-03-22'),
17     (1003,'K.P JAIN',300000.00,36,NULL,'2007-08-03'),
18     (1004,'M.P YADAV',800000.00,60,10.00,'2008-06-12'),
19     (1005,'S.P SINHA',200000.00,36,12.50,'2010-03-01'),
20     (1006,'P SHARMA',700000.00,60,12.50,'2008-05-06'),
21     (1007,'K.S DHALL',500000.00,48,NULL,'2008-05-03');
22 • SELECT * FROM LOAN_ACCOUNTS
  
```

The screenshot shows the results of the SQL query in a grid format. The columns are:

ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
1001	R.K GUPTA	300000.00	36	12.00	2009-07-19	NULL
1002	S.P SHARMA	500000.00	48	10.00	2008-03-22	NULL
1003	K.P JAIN	300000.00	36	NULL	2007-08-03	NULL
1004	M.P YADAV	800000.00	60	10.00	2008-06-12	NULL
1005	S.P SINHA	200000.00	36	12.50	2010-03-01	NULL
1006	P SHARMA	700000.00	60	12.50	2008-05-06	NULL
1007	K.S DHALL	500000.00	48	NULL	2008-05-03	NULL

Below the grid, there is a message bar indicating "accounts 1" and "output". At the bottom, there is an "Action Output" section showing two log entries:

#	Time	Action	Message
111	19:33:35	SELECT *FROM STUDENT WHERE AVG_MARK > (SELECT AVG(AVG_MARK) FROM ST...	3 row(s) returned
112	19:44:09	SELECT * FROM loan_accounts LIMIT 0, 1000	7 row(s) returned

1. Put the interest rate 11.50% for all the loans for which the interest rate is NULL.

### OUTPUT:

```
update loan_accounts set int_rate = 11.50 where int_rate is null;
```

| Output

Time Action

19:44:09 SELECT \* FROM loan\_accounts LIMIT 0, 1000

19:50:15 update loan\_accounts set int\_rate = 11.50 where int\_rate is null

2. Increase the interest rate by 0.5% for all the Loans for which the Loan amount is more than 400000.

**OUTPUT:**

```
24 • update loan_accounts set int_rate = int_rate + 0.5 where loan_amount > 400000
25 • SELECT * FROM loan_accounts;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |  |  Result Grid |  Form Editor |  Read Only |  Context

	ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
▶	1001	R.K GUPTA	300000.00	36	12.00	2009-07-19	NULL
	1002	S.P SHARMA	500000.00	48	10.50	2008-03-22	NULL
	1003	K.P JAIN	300000.00	36	11.50	2007-08-03	NULL
	1004	M.P YADAV	800000.00	60	10.50	2008-06-12	NULL
	1005	S.P SINHA	200000.00	36	12.50	2010-03-01	NULL
	1006	P SHARMA	700000.00	60	13.00	2008-05-06	NULL
	1007	K.S DHALL	500000.00	48	12.00	2008-05-03	NULL

loan\_accounts 2 ×

Output ::::::::::::

Action Output

#	Time	Action	Message
114	19:54:11	update loan_accounts set int_rate = int_rate + 0.5 where loan_amount > 400000.00	4 row(s) affected R
115	19:54:11	SELECT * FROM loan_accounts LIMIT 0, 1000	7 row(s) returned

3. For each Loan replace Interest with (Loan\_amount \* Int\_rate\* installments)/(12\*100).

**OUTPUT:**

```

26 • update loan_accounts set interest = (loan_amount * int_rate* installments)/(12*100);
27 • SELECT * FROM loan_accounts;
28

```

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

	ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
▶	1001	R.K GUPTA	300000.00	36	12.00	2009-07-19	108000.00
	1002	S.P SHARMA	500000.00	48	10.50	2008-03-22	210000.00
	1003	K.P JAIN	300000.00	36	11.50	2007-08-03	103500.00
	1004	M.P YADAV	800000.00	60	10.50	2008-06-12	420000.00
	1005	S.P SINHA	200000.00	36	12.50	2010-03-01	75000.00
	1006	P SHARMA	700000.00	60	13.00	2008-05-06	455000.00
	1007	K.S DHALL	500000.00	48	12.00	2008-05-03	240000.00

loan\_accounts 3 × Read Only Cont.

Action Output

#	Time	Action	Message
✓	116 19:57:07	update loan_accounts set interest = (loan_amount * int_rate* installments)/(12*100) where int...	7 row(s) affected
✓	117 19:57:07	SELECT * FROM loan_accounts LIMIT 0, 1000	7 row(s) returned

4. Delete the records of all the Loans whose start date is before 2008.

### OUTPUT:

```

28 • DELETE FROM loan_accounts WHERE start_date < '2008-01-01';
29 • SELECT * FROM loan_accounts;

```

Output ::::::::::::::::::::

Action Output

#	Time	Action
▶	117 19:57:07	SELECT * FROM loan_accounts LIMIT 0, 1000
▶	118 19:59:00	DELETE FROM loan_accounts WHERE start_date < '2008-01-01'

5. Delete the records of all the Loans whose name starts with 'K'

### OUTPUT:

```
30 • delete FROM loan_accounts where cust_name like 'K%';
```

Output :::::

Action Output

#	Time	Action	Message
119	20:00:32	SELECT * FROM loan_accounts LIMIT 0, 1000	6 row(s) returned
120	20:00:46	delete FROM loan_accounts where cust_name like 'K%'	1 row(s) affected

6. Display the details of all the Loans with less than 40 installments.

### OUTPUT:

```
31 • SELECT * FROM loan_accounts where installments < 40;
```

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

ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
1001	R.K GUPTA	300000.00	36	12.00	2009-07-19	108000.00
1005	S.P SINHA	200000.00	36	12.50	2010-03-01	75000.00

7. Display the Accno and Loan\_amount of all the loans started before 01-04-2009.

### OUTPUT:

32 • `SELECT accno,loan_amount, start_date from loan_accounts where start_date < '2009-04-01'`

accno	loan_amount	start_date
1002	500000.00	2008-03-22
1004	800000.00	2008-06-12
1006	700000.00	2008-05-06

Action Output

#	Time	Action	Message
121	20:01:58	<code>SELECT * FROM loan_accounts where installments &lt; 40 LIMIT 0, 1000</code>	2 row(s) returned
122	20:04:22	<code>SELECT accno,loan_amount, start_date from loan_accounts where start_date &lt; '2009-04-01'</code>	3 row(s) returned

8. Display the int\_rate of all Loans started after 01-04-2009.

### OUTPUT:

33 • `select int_rate ,start_date from loan_accounts where start_date > '2009-04-01'`

34

int_rate	start_date
12.00	2009-07-19
12.50	2010-03-01

9. Display the Accno, cust\_name and Loan amount for all the Loans for which the cust\_name ends with 'Sharma'.

### OUTPUT:

```
34 •   SELECT accno,cust_name,loan_amount FROM loan_accounts WHERE cust_name LIKE '%a'
```

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

accno	cust_name	loan_amount
1002	S.P SHARMA	500000.00
1006	P SHARMA	700000.00

loan\_accounts 8 × Read Only

Action Output

#	Time	Action	Message
123	20:06:06	select int_rate ,start_date from loan_accounts where start_date > '2009-04-01' LIMIT 0, 1000	2 row(s) returned
124	20:06:54	SELECT accno,cust_name,loan_amount FROM loan_accounts WHERE cust_name LIKE '%a' LIMIT 0, 1000	2 row(s) returned

10. Loan\_Amount of all the Loans for which the Cust\_name ends with 'a'.

## OUTPUT:

```
35 •   select loan_amount from loan_accounts where cust_name like '%a';
```

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

loan_amount
300000.00
500000.00
200000.00
700000.00

loan\_accounts 9 × Read Only Context Help

Action Output

#	Time	Action	Message
124	20:06:54	SELECT accno,cust_name,loan_amount FROM loan_accounts WHERE cust_name LIKE '%a' LIMIT 0, 1000	2 row(s) returned
125	20:07:50	select loan_amount from loan_accounts where cust_name like '%a' LIMIT 0, 1000	4 row(s) returned

11. Display the Accno, Cust\_name and Loan\_Amount for the Loans for which the Cust\_name contains 'a'.

**OUTPUT:**

The screenshot shows the MySQL Workbench interface. At the top, a SQL query is displayed: "select accno,cust\_name,loan\_amount from loan\_accounts where cust\_name like '%a%'". Below the query is a "Result Grid" table with the following data:

	accno	cust_name	loan_amount
▶	1001	R.K GUPTA	300000.00
	1002	S.P SHARMA	500000.00
	1004	M.P YADAV	800000.00
	1005	S.P SINHA	200000.00
	1006	P SHARMA	700000.00

On the right side of the interface, there is a sidebar with icons for "Result Grid", "Form Editor", and "Context Help". Below the table, the status bar shows "loan\_accounts 10 ×" and "Read Only". The bottom section shows the "Action Output" log with two entries:

#	Time	Action	Message
125	20:07:50	select loan_amount from loan_accounts where cust_name like "%a%" LIMIT 0, 1000	4 row(s) returned
126	20:11:18	select accno,cust_name,loan_amount from loan_accounts where cust_name like "%a%" LIMIT 0, 1000	5 row(s) returned

12. Display the Accno, Cust\_name and Loan\_Amount for all the Loans for which the Cust\_name does not contain 'P'.

**OUTPUT:**

```
37 • select accno,cust_name,loan_amount from loan_accounts where cust_name not like "%a%"
```

accno	cust_name	loan_amount
-------	-----------	-------------

loan\_accounts 11 ×

Output

Action Output

#	Time	Action	Message
126	20:11:18	select accno,cust_name,loan_amount from loan_accounts where cust_name like "%a%" LIMIT 5	5 row(s) returned
127	20:12:02	select accno,cust_name,loan_amount from loan_accounts where cust_name not like "%p%"	0 row(s) returned

13. Display the structure of table LOAN\_ACCOUNTS so that you can verify that the table is created as required.

## OUTPUT:

```
38 • describe loan_accounts;
```

Field	Type	Null	Key	Default	Extra
ACCNO	int	YES		NULL	
CUST_NAME	varchar(15)	YES		NULL	
LOAN_AMOUNT	decimal(8,2)	YES		NULL	
INSTALLMENTS	int	YES		NULL	
INT_RATE	decimal(4,2)	YES		NULL	
START_DATE	date	YES		NULL	
INTEREST	decimal(8,2)	YES		NULL	
CATEGORY	varchar(1)	YES		NULL	

result 12 ×

Output

Action Output

#	Time	Action
127	20:12:02	select accno,cust_name,loan_amount from loan_accounts v
128	20:13:16	describe loan_accounts

14. Display the details of all the loans in the ascending order of their Loan\_Amount.

### OUTPUT:

The screenshot shows the MySQL Workbench interface. At the top, a SQL query is displayed: "39 • select \* from loan\_accounts order by loan\_amount asc;". Below the query is a "Result Grid" table with the following data:

ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
1005	S.P SINHA	200000.00	36	12.50	2010-03-01	75000.00
1001	R.K GUPTA	300000.00	36	12.00	2009-07-19	108000.00
1002	S.P SHARMA	500000.00	48	10.50	2008-03-22	210000.00
1006	P SHARMA	700000.00	60	13.00	2008-05-06	455000.00
1004	M.P YADAV	800000.00	60	10.50	2008-06-12	420000.00

Below the result grid, there is a "loan\_accounts 13" tab and an "Output" section. The "Output" section contains the following log entries:

#	Time	Action	Count
128	20:13:16	describe loan_accounts	8
129	20:13:52	select *from loan_accounts order by loan_amount asc LIMIT 0, 1000	5

15. Display the details of all the Loans in the descending order of their Start\_date.

### OUTPUT:

```
40 • select * from loan_accounts order by start_date desc;
```

	ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
▶	1005	S.P SINHA	200000.00	36	12.50	2010-03-01	75000.00
	1001	R.K GUPTA	300000.00	36	12.00	2009-07-19	108000.00
	1004	M.P YADAV	800000.00	60	10.50	2008-06-12	420000.00
	1006	P SHARMA	700000.00	60	13.00	2008-05-06	455000.00
	1002	S.P SHARMA	500000.00	48	10.50	2008-03-22	210000.00

loan\_accounts 14 × Read Only

Output ::::::::::::

Action Output

#	Time	Action	Message
129	20:13:52	select * from loan_accounts order by loan_amount asc LIMIT 0, 1000	5 row(s) returned
130	20:14:39	select * from loan_accounts order by start_date desc LIMIT 0, 1000	5 row(s) returned

16. Display the details of all the Loans in the ascending order of their Loan\_amount and within Loan\_amount in the descending order of their Start\_date.

### OUTPUT:

```
41 • select * from loan_accounts order by loan_amount,start_date desc;
```

	ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
▶	1005	S.P SINHA	200000.00	36	12.50	2010-03-01	75000.00
	1001	R.K GUPTA	300000.00	36	12.00	2009-07-19	108000.00
	1002	S.P SHARMA	500000.00	48	10.50	2008-03-22	210000.00
	1006	P SHARMA	700000.00	60	13.00	2008-05-06	455000.00
	1004	M.P YADAV	800000.00	60	10.50	2008-06-12	420000.00

loan\_accounts 16 × Read Only

Output ::::::::::::

Action Output

#	Time	Action	Message
131	20:17:30	select accno,cust_name,loan_amount from loan_accounts where cust_name like '%%' LIMIT ...	0 row(s) returned
132	20:31:41	select * from loan_accounts order by loan_amount,start_date desc LIMIT 0, 1000	5 row(s) returned

17. Display the Accno, Cust\_name and Loan\_Amount of all the Loans for which the Cust\_name starts with 'K'.

**OUTPUT:**

The screenshot shows the MySQL Workbench interface. At the top, a SQL query is displayed: "select accno,cust\_name,loan\_amount from loan\_accounts where cust\_name like 'K%'". Below the query is a "Result Grid" pane with a header row containing "accno", "cust\_name", and "loan\_amount". The main body of the grid is currently empty. To the right of the grid is a vertical toolbar with icons for "Result Grid", "Form Editor", and "Read Only". Below the grid, the status bar shows "loan\_accounts 17 x" and "Read Only". Underneath the grid, there is an "Output" section with a "Action Output" tab. It contains a table with two rows, both of which have a green checkmark next to them, indicating successful execution. The first row shows a query to select all columns from the loan\_accounts table ordered by loan\_amount and start\_date desc, limited to 0, 1000 rows. The second row shows a query to select accno, cust\_name, and loan\_amount from loan\_accounts where cust\_name starts with 'K%', also limited to 0, 1000 rows. Both rows show "5 row(s) r" in the message column.

#	Time	Action	Message
132	20:31:41	select * from loan_accounts order by loan_amount,start_date desc LIMIT 0, 1000	5 row(s) r
133	20:33:23	select accno,cust_name,loan_amount from loan_accounts where cust_name like 'K%' LIMIT ...	0 row(s) r

18. Display the details of all the Loans whose rate of interest in NULL.

**OUTPUT:**

43 • select \* from loan\_accounts where int\_rate is null;

ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
-------	-----------	-------------	--------------	----------	------------	----------

loan\_accounts 18 ×      Read Only

Output :::::::::::::

Action Output

#	Time	Action	Message
133	20:33:23	select accno,cust_name,loan_amount from loan_accounts where cust_name like 'k%' LIMIT ...	0 row(s) return
134	20:34:54	select * from loan_accounts where int_rate is null LIMIT 0, 1000	0 row(s) return

19. Display the details of all the loans whose rate of interest is not NULL.

### OUTPUT:

44 • select \* from loan\_accounts where int\_rate is not null;

ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
1001	R.K GUPTA	300000.00	36	12.00	2009-07-19	108000.00
1002	S.P SHARMA	500000.00	48	10.50	2008-03-22	210000.00
1004	M.P YADAV	800000.00	60	10.50	2008-06-12	420000.00
1005	S.P SINHA	200000.00	36	12.50	2010-03-01	75000.00
1006	P SHARMA	700000.00	60	13.00	2008-05-06	455000.00

loan\_accounts 19 ×

Output :::::::::::::

Action Output

#	Time	Action
134	20:34:54	select * from loan_accounts where int_rate is null LIMIT 0, 1000
135	20:36:16	select * from loan_accounts where int_rate is not null LIMIT 0, 1000

20. Display the amounts of various loans from the table Loan\_Accounts.  
A Loan\_Amount should appear only once.

### OUTPUT:

```
45 • select distinct loan_amount from loan_accounts ;
```

loan_amount
300000.00
500000.00
800000.00
200000.00
700000.00

21. Display the details of all the loans started after 31-12-2008 for which the number of installments are more than 36.

### OUTPUT:

```
46 • SELECT * FROM loan_accounts where (start_date > '2008-12-31' and installmen
```

ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST

loan\_accounts 21 ×      Read Only      Context Help

Action Output

#	Time	Action	Message
136	20:39:17	select distinct loan_amount from loan_accounts LIMIT 0, 1000	5 row(s) returned
137	20:44:46	SELECT * FROM loan_accounts where (start_date > '2008-12-31' and installments > 36) LIM...	0 row(s) returned

22. Display the Customer\_name and Loan\_amount for all the Loans for which the Loan amount is less than 500000 or int\_rate is more than 12.

### OUTPUT:

```
47 • select cust_name,loan_amount from loan_accounts where (loan_amount < 500000
48 •
49 •
50 •
51 •
52 •
53 •
54 •
55 •
56 •
57 •
58 •
59 •
60 •
61 •
62 •
63 •
64 •
65 •
66 •
67 •
68 •
69 •
70 •
71 •
72 •
73 •
74 •
75 •
76 •
77 •
78 •
79 •
80 •
81 •
82 •
83 •
84 •
85 •
86 •
87 •
88 •
89 •
90 •
91 •
92 •
93 •
94 •
95 •
96 •
97 •
98 •
99 •
100 •
101 •
102 •
103 •
104 •
105 •
106 •
107 •
108 •
109 •
110 •
111 •
112 •
113 •
114 •
115 •
116 •
117 •
118 •
119 •
120 •
121 •
122 •
123 •
124 •
125 •
126 •
127 •
128 •
129 •
130 •
131 •
132 •
133 •
134 •
135 •
136 •
137 •
138 •
```

Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ] | Result Grid | Form Editor | Action Output

cust_name	loan_amount
R.K GUPTA	300000.00
S.P SINHA	200000.00
P SHARMA	700000.00

loan\_accounts 22 × | Read Only | Context Help

Output:

Action Output

#	Time	Action	Message
137	20:44:46	SELECT * FROM loan_accounts where (start_date > '2008-12-31' and installments > 36) LIMIT ...	0 row(s) returned
138	20:45:55	select cust_name,loan_amount from loan_accounts where (loan_amount < 500000.00 or int_...	3 row(s) returned

23. Display the details of all Loans which started in the year 2009.

## OUTPUT:

```
48 • select * from loan_accounts where start_date between '2009-01-01' and '2009
49 •
50 •
51 •
52 •
53 •
54 •
55 •
56 •
57 •
58 •
59 •
60 •
61 •
62 •
63 •
64 •
65 •
66 •
67 •
68 •
69 •
70 •
71 •
72 •
73 •
74 •
75 •
76 •
77 •
78 •
79 •
80 •
81 •
82 •
83 •
84 •
85 •
86 •
87 •
88 •
89 •
90 •
91 •
92 •
93 •
94 •
95 •
96 •
97 •
98 •
99 •
100 •
101 •
102 •
103 •
104 •
105 •
106 •
107 •
108 •
109 •
110 •
111 •
112 •
113 •
114 •
115 •
116 •
117 •
118 •
119 •
120 •
121 •
122 •
123 •
124 •
125 •
126 •
127 •
128 •
129 •
130 •
131 •
132 •
133 •
134 •
135 •
136 •
137 •
138 •
139 •
```

Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ] | Result Grid | Form Editor | Action Output

ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
1001	R.K GUPTA	300000.00	36	12.00	2009-07-19	108000.00

loan\_accounts 23 × | Read Only | Context Help

Output:

Action Output

#	Time	Action	Message
138	20:45:55	select cust_name,loan_amount from loan_accounts where (loan_amount < 500000.00 or int_...	3 row(s) returned
139	20:46:32	select * from loan_accounts where start_date between '2009-01-01' and '2009-12-31' LIMIT ...	1 row(s) returned

24. Display the details of all the Loans whose Loan amount is in the Range

400000 to 500000.

### OUTPUT:

```
49 • select * from loan_accounts where loan_amount between 400000.00 and 500000.
```

ACCNO	CUST_NAME	LOAN_AMOUNT	INSTALLMENTS	INT_RATE	START_DATE	INTEREST
1002	S.P SHARMA	500000.00	48	10.50	2008-03-22	210000.00

25. Display the Customer\_name and Loan\_amount of all the Loans for which the number of installments are 26, 36 and 48.

### OUTPUT:

```
select cust_name,installments, loan_amount from loan_accounts where installments in (26,36,48);
update loan_accounts set int_rate =0.0 where int_rate is null;
```

cust_name	installments	loan_amount
KUPTA	36	300000.00
HARMA	48	500000.00
INHA	36	200000.00

26. Display the customer name, loan\_amount and interest rate. If interest rate is NULL, display it as 0.

### OUTPUT:

```

52 • select cust_name,loan_amount,int_rate from loan_accounts where int_rate = 0
53
54
Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]
cust_name | loan_amount | int_rate |

```

loan\_accounts 26 × Read Only

Output:

Action Output

#	Time	Action	Message
142	20:49:04	update loan_accounts set int_rate =0.0 where int_rate is null	0 row(s)
143	20:50:37	select cust_name,loan_amount,int_rate from loan_accounts where int_rate = 0.0 LIMIT 0, 10...	0 row(s)

27. Display the customer name, loan\_amount and interest rate. If interest rate is NULL, display it as “No Interest”

```

53 • select cust_name,loan_amount,ifnull(int_rate,'No Interest') as int_rate from loan_accounts where int_rate = 0.0 LIMIT 0, 10...
54
Result Grid | Filter Rows: [ ] Export: [ ] Wrap Cell Content: [ ]
cust_name | loan_amount | int_rate |
R.K GUPTA | 300000.00 | 12.00
S.P SHARMA | 500000.00 | 10.50
M.P YADAV | 800000.00 | 10.50
S.P SINHA | 200000.00 | 12.50
P SHARMA | 700000.00 | 13.00

```

result 28 × Read Only Context

Output:

Action Output

#	Time	Action	Message
145	20:59:53	select cust_name,loan_amount,int_rate from loan_accounts where int_rate = 0.0 LIMIT 0, 10...	0 row(s) returned
146	21:01:07	select cust_name,loan_amount,ifnull(int_rate,'No Interest') as int_rate from loan_accounts LI...	5 row(s) returned

# CYCLE 2

## Q.SET 1

Create the following tables and execute the queries given below

### SAILORS

sid	sname	rating	age
22	Dustin	7	45
29	Brutas	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horatio	9	35
85	Art	3	26
95	Bob	3	64

### BOATS

Bid	bname	color
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red

### RESERVES

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98

31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

- Find the names and ages of all sailors

## OUTPUT

```

4 • INSERT INTO SAILORS VALUES (22,'DUSTIN',7,45),(29,'BRUTAS',1,33),(31,'LUBBER',8,55),(32,'ANDY',8,25),
5   (58,'RUSTY',10,35),(64,'HORATIO',7,35),(71,'ZORBA',10,16),(74,'HORATIO',9,35),(85,'ART',3,26),(95,'BOB',3,64);
6 • CREATE TABLE BOATS
7   (
8     BID INT,BNAME VARCHAR(40),COLOR VARCHAR(33));
9 • INSERT INTO BOATS VALUES(101,'INTERLAKE','BLUE'),(102,'INTERLAKE','RED'),(103,'CLIPPER','GREEN'),(104,'MARINE','RED');
10 • CREATE TABLE RESERVES (
11   SID INT,BID INT,DAY DATE);
12 • SELECT * FROM RESERVES;
13 • INSERT INTO RESERVES VALUES (22,101,'1998-10-10'),(22,102,'1998-10-10');
14 • INSERT INTO RESERVES VALUES (22,103,'1998-08-10'),(22,104,'1998-07-10'),(31,102,'1998-10-11'),(31,103,'1998-06-11'),
15   (31,104,'1998-12-11'),(64,101,'1998-05-09'),(64,102,'1998-08-09'),(74,103,'1998-08-09');
16 • SELECT SNAME,AGE FROM SAILORS;

```

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

SNAME	AGE
DUSTIN	45
BRUTAS	33
LUBBER	55
ANDY	25
RUSTY	35
HORATIO	35
ZORBA	16
HORATIO	35
ART	26
BOB	64

SAILORS 3 ×

- Find all information of sailors who have reserved boat number 101.

## OUTPUT

```

5      (58,'RUSTY',10,35),(64,'HORATIO',7,35),(71,'ZORBA',10,16),(74,'HORATIO',9,35),(85,'ART',3,26),(95,'BOB',3,64);
6 • CREATE TABLE BOATS
7   (
8     BID INT,BNAME VARCHAR(40),COLOR VARCHAR(33));
9 • INSERT INTO BOATS VALUES(101,'INTERLAKE','BLUE'),(102,'INTERLAKE','RED'),(103,'CLIPPER','GREEN'),(104,'MARINE','RED');
10 • CREATE TABLE RESERVES (
11   SID INT,BID INT,DAY DATE);
12 • SELECT * FROM RESERVES;
13 • INSERT INTO RESERVES VALUES (22,101,'1998-10-10'),(22,102,'1998-10-10');
14 • INSERT INTO RESERVES VALUES (22,103,'1998-08-10'),(22,104,'1998-07-10'),(31,102,'1998-10-11'),(31,103,'1998-06-11'),
15   (31,104,'1998-12-11'),(64,101,'1998-05-09'),(64,102,'1998-08-09'),(74,103,'1998-08-09');
16 • SELECT SNAME,AGE FROM SAILORS;
17 • SELECT *FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=101;
--
```

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

SID	SNAME	RATING	AGE	SID	BID	DAY
22	DUSTIN	7	45	22	101	1998-10-10
64	HORATIO	7	35	64	101	1998-05-09

3. Find all sailors with rating above 7.

## OUTPUT

```

6 • CREATE TABLE BOATS
7   (
8     BID INT,BNAME VARCHAR(40),COLOR VARCHAR(33));
9 • INSERT INTO BOATS VALUES(101,'INTERLAKE','BLUE'),(102,'INTERLAKE','RED'),(103,'CLIPPER','GREEN'),(104,'MARINE','RED');
0 • CREATE TABLE RESERVES (
1   SID INT,BID INT,DAY DATE);
2 • SELECT * FROM RESERVES;
3 • INSERT INTO RESERVES VALUES (22,101,'1998-10-10'),(22,102,'1998-10-10');
4 • INSERT INTO RESERVES VALUES (22,103,'1998-08-10'),(22,104,'1998-07-10'),(31,102,'1998-10-11'),(31,103,'1998-06-11'),
5   (31,104,'1998-12-11'),(64,101,'1998-05-09'),(64,102,'1998-08-09'),(74,103,'1998-08-09');
6 • SELECT SNAME,AGE FROM SAILORS;
7 • SELECT *FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=101;
8 • SELECT *FROM SAILORS WHERE RATING>7;
```

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

SID	SNAME	RATING	AGE
31	LUBBER	8	55
32	ANDY	8	25
58	RUSTY	10	35
71	ZORBA	10	16
74	HORATIO	9	35

4. Find the names of sailors who have reserved boat no 103.

## OUTPUT

```

7   (
8     BID INT,BNAME VARCHAR(40),COLOR VARCHAR(33));
9 •  INSERT INTO BOATS VALUES(101,'INTERLAKE','BLUE'),(102,'INTERLAKE','RED'),(103,'CLIPPER','GREEN'),(104,'MARINE','RED');
10 • CREATE TABLE RESERVES (
11   SID INT,BID INT,DAY DATE);
12 •  SELECT * FROM RESERVES;
13 •  INSERT INTO RESERVES VALUES (22,101,'1998-10-10'),(22,102,'1998-10-10');
14 •  INSERT INTO RESERVES VALUES (22,103,'1998-08-10'),(22,104,'1998-07-10'),(31,102,'1998-10-11'),(31,103,'1998-06-11'),
15   (31,104,'1998-12-11'),(64,101,'1998-05-09'),(64,102,'1998-08-09'),(74,103,'1998-08-09');
16 •  SELECT SNAME,AGE FROM SAILORS;
17 •  SELECT *FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=101;
18 •  SELECT *FROM SAILORS WHERE RATING>7;
19 •  SELECT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=103;
...

```

Result Grid	
	SNAME
•	DUSTIN
•	LUBBER
•	HORATIO

5. Find the names of sailors who have reserved a red boat, and list in the order of age.

## OUTPUT

```

8   (
9     BID INT,BNAME VARCHAR(40),COLOR VARCHAR(33));
10 •  INSERT INTO BOATS VALUES(101,'INTERLAKE','BLUE'),(102,'INTERLAKE','RED'),(103,'CLIPPER','GREEN'),(104,'MARINE','RED');
11 • CREATE TABLE RESERVES (
12   SID INT,BID INT,DAY DATE);
13 •  SELECT * FROM RESERVES;
14 •  INSERT INTO RESERVES VALUES (22,101,'1998-10-10'),(22,102,'1998-10-10');
15 •  INSERT INTO RESERVES VALUES (22,103,'1998-08-10'),(22,104,'1998-07-10'),(31,102,'1998-10-11'),(31,103,'1998-06-11'),
16   (31,104,'1998-12-11'),(64,101,'1998-05-09'),(64,102,'1998-08-09'),(74,103,'1998-08-09');
17 •  SELECT SNAME,AGE FROM SAILORS;
18 •  SELECT *FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=101;
19 •  SELECT *FROM SAILORS WHERE RATING>7;
20 •  SELECT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=103;
...

```

Result Grid	
	SNAME
•	DUSTIN
•	LUBBER
•	HORATIO

6. Find the names of sailors who have reserved either a red or green boat

## OUTPUT

```

9 •   INSERT INTO BOATS VALUES(101, INTERLAKE , BLUE ),(102, INTERLAKE , RED ),(103, CLIPPER , GREEN ),(104, MARINE , RED );
10 • CREATE TABLE RESERVES (
11     SID INT,BID INT,DAY DATE);
12 •   SELECT * FROM RESERVES;
13 •   INSERT INTO RESERVES VALUES (22,101,'1998-10-10'),(22,102,'1998-10-10');
14 •   INSERT INTO RESERVES VALUES (22,103,'1998-08-10'),(22,104,'1998-07-10'),(31,102,'1998-10-11'),(31,103,'1998-06-11'),
15     (31,104,'1998-12-11'),(64,101,'1998-05-09'),(64,102,'1998-08-09'),(74,103,'1998-08-09');
16 •   SELECT SNAME,AGE FROM SAILORS;
17 •   SELECT *FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=101;
18 •   SELECT *FROM SAILORS WHERE RATING>7;
19 •   SELECT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=103;
20 •   SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND COLOR='RED';
21 •   SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND (COLOR='RED' OR COLOR='GREEN');
22
23
24

```

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

SNAME
DUSTIN
LUBBER
HORATIO

Result Grid | Form Editor

7. Find the colors of boats reserved by “Lubber”.

## OUTPUT

```

12 •   SELECT * FROM RESERVES;
13 •   INSERT INTO RESERVES VALUES (22,101,'1998-10-10'),(22,102,'1998-10-10');
14 •   INSERT INTO RESERVES VALUES (22,103,'1998-08-10'),(22,104,'1998-07-10'),(31,102,'1998-10-11'),(31,103,'1998-06-11'),
15     (31,104,'1998-12-11'),(64,101,'1998-05-09'),(64,102,'1998-08-09'),(74,103,'1998-08-09');
16 •   SELECT SNAME,AGE FROM SAILORS;
17 •   SELECT *FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=101;
18 •   SELECT *FROM SAILORS WHERE RATING>7;
19 •   SELECT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=103;
20 •   SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND COLOR='RED';
21 •   SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND (COLOR='RED' OR COLOR='GREEN');
22
23 •   SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND SNAME='LUBBER';
24

```

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

COLOR
RED
GREEN

Result Grid | Form Editor

8. Find the names of sailors who have reserved both red and green boats.

## OUTPUT

```

20 •  SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND COLOR='RED';
21 •  SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND (COLOR='RED' OR COLO
22
23 •  SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND SNAME='LUBBER';
24 •  SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE COLOR='RED' AND SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID
25   AND EXISTS( SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS
26   WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID
27   AND COLOR='GREEN'));
28 •  SELECT DISTINCT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID ;
29 •  SELECT SNAME ,RESERVES.SID FROM RESERVES,SAILORS WHERE SAILORS.SID=RESERVES.SID GROUP BY DAY,

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
SNAME				
DUSTIN				
LUBBER				
HORATIO				

9. Find the names of sailors who have reserved at least one boat.

## OUTPUT

```

19 •  SELECT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=103;
20 •  SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND COLOR='RED';
21 •  SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND (COLOR='RED' OR COLOR='GR
22
23 •  SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND SNAME='LUBBER';
24 •  SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE COLOR='RED' AND SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID
25   IN( SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS
26   WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID
27   AND COLOR='GREEN');
28 •  SELECT DISTINCT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID ;

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
SNAME				
DUSTIN				
LUBBER				
HORATIO				

10. Find the ids and names of sailors who have reserved two different boats on the same day.

## OUTPUT

```

1 Limit to 1000 rows
20 • SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND COLOR='RED';
21 • SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND (COLOR='RED' OR COL
22
23 • SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND SNAME='LUBBER';
24 • SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE COLOR='RED' AND SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID
25 • IN( SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS
26 WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID
27 AND COLOR='GREEN');
28 • SELECT DISTINCT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID ;
29 • SELECT SNAME ,RESERVES.SID FROM RESERVES,SAILORS WHERE SAILORS.SID=RESERVES.SID GROUP BY DAY,
30 SNAME ,RESERVES.SID HAVING COUNT(DAY)>1 ;
31
32

```

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

SNAME	SID
DUSTIN	22

11. Find the name and the age of the youngest sailor.

## OUTPUT

```

19 • SELECT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID AND BID=103;
20 • SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND COLOR='RED';
21 • SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND (COLOR='RED' OR COL
22
23 • SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND SNAME='LUBBER';
24 • SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE COLOR='RED' AND SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID
25 • IN( SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS
26 WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID
27 AND COLOR='GREEN');
28 • SELECT DISTINCT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID ;
29 • SELECT SNAME ,RESERVES.SID FROM RESERVES,SAILORS WHERE SAILORS.SID=RESERVES.SID GROUP BY DAY,
30 SNAME ,RESERVES.SID HAVING COUNT(DAY)>1 ;
31 • SELECT SNAME,AGE FROM SAILORS WHERE AGE =(SELECT MIN(AGE) FROM SAILORS ) ;
32

```

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

SNAME	AGE
ZORBA	16

12. Find the names and ratings of sailor whose rating is better than some sailor called Horatio.

## OUTPUT

```

20 •   SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND COLOR='RED';
21 •   SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND (COLOR='RED' OR
22
23 •   SELECT DISTINCT COLOR FROM SAILORS,RESERVES,BOATS WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID AND SNAME='LUBBER'
24 •   SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS WHERE COLOR='RED' AND SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID
25     IN( SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS
26       WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID = RESERVES.BID
27       AND COLOR='GREEN');
28 •   SELECT DISTINCT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID ;
29 •   SELECT SNAME ,RESERVES.SID FROM RESERVES,SAILORS WHERE SAILORS.SID=RESERVES.SID GROUP BY DAY,
30   SNAME ,RESERVES.SID HAVING COUNT(DAY)>1 ;
31 •   SELECT SNAME,AGE FROM SAILORS WHERE AGE =(SELECT MIN(AGE) FROM SAILORS ) ;
32 •   SELECT SNAME FROM SAILORS WHERE RATING > (SELECT MAX(RATING ) FROM SAILORS WHERE SNAME = 'HORATIO');


```

Result Grid	
Filter Rows:	<input type="text"/>
Export:	
Wrap Cell Content:	
SNAME	
RUSTY	
ZORBA	

13. Find the names of sailors who have reserved all boats.

## OUTPUT

```

28 •   SELECT DISTINCT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID ;
29 •   SELECT SNAME ,RESERVES.SID FROM RESERVES,SAILORS WHERE SAILORS.SID=RESERVES.SID GROUP BY DAY,
30   SNAME ,RESERVES.SID HAVING COUNT(DAY)>1 ;
31 •   SELECT SNAME,AGE FROM SAILORS WHERE AGE =(SELECT MIN(AGE) FROM SAILORS ) ;
32 •   SELECT SNAME FROM SAILORS WHERE RATING > (SELECT MAX(RATING ) FROM SAILORS WHERE SNAME = 'HORATIO');
33 •   SELECT SNAME FROM (SELECT SNAME,RESERVES.SID,COUNT(BID) AS id FROM RESERVES,SAILORS WHERE
34   SAILORS.SID=RESERVES.SID GROUP BY RESERVES.SID,SNAME) a
35   WHERE id =( SELECT COUNT(BID) FROM BOATS);
36 •   SELECT COUNT(C.SNAME) FROM (SELECT DISTINCT SNAME FROM SAILORS) C;
37 •   SELECT AVG(AGE) FROM SAILORS;


```

Result Grid	
Filter Rows:	<input type="text"/>
Export:	
Wrap Cell Content:	
SNAME	
DUSTIN	

14. Count the number of different sailor names.

## OUTPUT

```

25   IN( SELECT DISTINCT SNAME FROM SAILORS,RESERVES,BOATS
26     WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID    = RESERVES.BID
27     AND COLOR='GREEN');
28 •  SELECT DISTINCT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID ;
29 •  SELECT SNAME ,RESERVES.SID FROM RESERVES,SAILORS WHERE SAILORS.SID=RESERVES.SID GROUP BY DAY,
30   SNAME ,RESERVES.SID HAVING COUNT(DAY)>1 ;
31 •  SELECT SNAME,AGE FROM SAILORS WHERE AGE  =(SELECT MIN(AGE) FROM SAILORS ) ;
32 •  SELECT SNAME FROM SAILORS WHERE RATING > (SELECT MAX(RATING ) FROM SAILORS WHERE SNAME = 'HORATIO');
33 ✘  SELECT SNAME,SID FROM (SELECT SNAME,RESERVES.SID COUNT(BID) AS id FROM RESERVES,SAILORS WHERE
34   SAILORS.SID=RESERVES.SID GROUP BY RESERVES.SID,SNAME)
35   WHERE id =( SELECT COUNT(BID) FROM BOATS);
36 •  SELECT COUNT(C.SNAME) FROM (SELECT DISTINCT SNAME FROM SAILORS) C;

```

Result Grid		Filter Rows:	<input type="text"/>	Export:		Wrap Cell Content:	
COUNT(C.SNAME)							

15. Calculate the average age of all sailors.

## OUTPUT

```

20 -  SELECT DISTINCT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID ,
29 •  SELECT SNAME ,RESERVES.SID FROM RESERVES,SAILORS WHERE SAILORS.SID=RESERVES.SID GROUP BY DAY,
30   SNAME ,RESERVES.SID HAVING COUNT(DAY)>1 ;
31 •  SELECT SNAME,AGE FROM SAILORS WHERE AGE  =(SELECT MIN(AGE) FROM SAILORS ) ;
32 •  SELECT SNAME FROM SAILORS WHERE RATING > (SELECT MAX(RATING ) FROM SAILORS WHERE SNAME = 'HORATIO');
33 ✘  SELECT SNAME,SID FROM (SELECT SNAME,RESERVES.SID COUNT(BID) AS id FROM RESERVES,SAILORS WHERE
34   SAILORS.SID=RESERVES.SID GROUP BY RESERVES.SID,SNAME)
35   WHERE id =( SELECT COUNT(BID) FROM BOATS);
36 •  SELECT COUNT(C.SNAME) FROM (SELECT DISTINCT SNAME FROM SAILORS) C;
37 •  SELECT AVG(AGE) FROM SAILORS;

```

Result Grid		Filter Rows:	<input type="text"/>	Export:		Wrap Cell Content:	
AVG(AGE)							

16. Find the average age of sailors for each rating level.

## OUTPUT

```

26   WHERE SAILORS.SID=RESERVES.SID AND BOATS.BID    = RESERVES.BID
27   AND COLOR='GREEN');
28 •  SELECT DISTINCT SNAME FROM SAILORS,RESERVES WHERE SAILORS.SID=RESERVES.SID ;
29 •  SELECT SNAME ,RESERVES.SID FROM RESERVES,SAILORS WHERE SAILORS.SID=RESERVES.SID GROUP BY DAY,
30   SNAME ,RESERVES.SID HAVING COUNT(DAY)>1 ;
31 •  SELECT SNAME,AGE FROM SAILORS WHERE AGE  =(SELECT MIN(AGE) FROM SAILORS ) ;
32 •  SELECT SNAME FROM SAILORS WHERE RATING > (SELECT MAX(RATING ) FROM SAILORS WHERE SNAME = 'HORATIO');
33 ✘  SELECT SNAME,SID FROM (SELECT SNAME,RESERVES.SID COUNT(BID) AS id FROM RESERVES,SAILORS WHERE
34   SAILORS.SID=RESERVES.SID GROUP BY RESERVES.SID,SNAME)
35   WHERE id =( SELECT COUNT(BID) FROM BOATS);
36 •  SELECT COUNT(C.SNAME) FROM (SELECT DISTINCT SNAME FROM SAILORS) C;
37 •  SELECT AVG(AGE) FROM SAILORS;
38 •  SELECT RATING,Avg(AGE) FROM SAILORS GROUP BY RATING;

```

Result Grid		
	RATING	Avg(AGE)
▶	7	40.0000
	1	33.0000
	8	40.0000
	10	25.5000
	9	35.0000
	3	45.0000

17. Find the average age of sailors for each rating level that has at least two sailors.

## OUTPUT

```

34   SAILORS.SID=RESERVES.SID GROUP BY RESERVES.SID,SNAME)
35   WHERE id =( SELECT COUNT(BID) FROM BOATS);
36 •  SELECT COUNT(C.SNAME) FROM (SELECT DISTINCT SNAME FROM SAILORS) C;
37 •  SELECT AVG(AGE) FROM SAILORS;
38 •  SELECT RATING,Avg(AGE) FROM SAILORS GROUP BY RATING;
39 •  SELECT A.RATING,B.MEAN FROM
40   (SELECT COUNT(SNAME) AS NUM,RATING FROM SAILORS
41   GROUP BY RATING HAVING COUNT(sname)>1 ) A,
42   (SELECT RATING,Avg(age) AS mean FROM SAILORS GROUP BY RATING ) B WHERE A.RATING = B.RATING;
43
44
45

```

Result Grid		
	RATING	MEAN
▶	7	40.0000
	8	40.0000
	10	25.5000
	3	45.0000

## Q.SET 2

1. Create the table STUDENT\_INFO with Columns: Sid, Stud\_name & stude\_score.
  - Insert values into STUDENT\_INFO with the following constraints: Sid should be unique, Stud name NOT NULL and stude\_score DEFAULT value of 20.

## OUTPUT

```
1 • CREATE TABLE STUDENT_INFO(Sid  INT UNIQUE ,Stud_name  VARCHAR(20) NOT NULL,
2           stude_score numeric(5,2) DEFAULT 20);
3
4 •   ALTER TABLE STUDENT_INFO ADD PRIMARY KEY (Sid);
5 •   DESCRIBE STUDENT_INFO;
```

The screenshot shows the 'Result Grid' tab of MySQL Workbench displaying the description of the STUDENT\_INFO table. The table has three columns: Sid, Stud\_name, and stude\_score. The 'Field' column lists the column names, 'Type' lists their data types (int, varchar(20), and decimal(5,2)), 'Null' indicates whether they can be null (NO for all), 'Key' shows the primary key status (PRI for Sid), and 'Default' and 'Extra' provide additional metadata.

Field	Type	Null	Key	Default	Extra
Sid	int	NO	PRI	NULL	
Stud_name	varchar(20)	NO		NULL	
stude_score	decimal(5,2)	YES		20.00	

- Update stude\_score by adding a value of 5 to stude\_score in the table STUDENT\_INFO for the rows satisfying the condition of stude\_score >150 (Using CASE)

## OUTPUT

```
0 • UPDATE STUDENT_INFO SET stude_score = stude_score +5 WHERE stude_score > 1
1 • SELECT * FROM STUDENT_INFO;
```

The screenshot shows the 'Result Grid' tab of MySQL Workbench displaying the updated data in the STUDENT\_INFO table. The table now contains four rows with the following data: (1, annu, 203.00), (2, angel, 123.00), (3, aswini, 148.00), and (4, nithya, 160.00). The 'sid' column contains integer values, while 'stud\_name' and 'stude\_score' contain their respective string and decimal values.

sid	stud_name	stude_score
1	annu	203.00
2	angel	123.00
3	aswini	148.00
4	nithya	160.00
NULL	NULL	NULL

2. Create the tables **worker** and **bonus** with the following fields. The primary key of Worker table is Worker\_ID. Set Worker\_id as foreign key of bonus on update and delete cascade constraints. Each constraint should be given a name

```

21 •    INSERT INTO WORKER VALUE
22      (1, 'Monika', 'Arora', 100000, '2014-02-20', 'HR'), (2, 'Niharika', 'Verma', 80000
23      (3, 'Vishal', 'Singhal', 300000, '2014-02-20', 'HR'), (4, 'Amitabh', 'Singh', 50
24      (5, 'Vivek', 'Bhati', 500000, '2014-06-11', 'Admin'), (6, 'Vipul', 'Diwan', 200000,
25      (7, 'Satish', 'Kumar', 75000, '2014-01-20', 'Account'), (8, 'Geetika', 'Chauhan',
26 •    SELECT * FROM WORKER;
27

```

The screenshot shows a database interface with a command window at the top and a result grid below it. The command window contains the SQL code for creating the WORKER table and inserting data. The result grid displays the inserted data with columns: WORKER\_ID, FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, and DEPARTMENT. The data is as follows:

WORKER_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20	HR
2	Niharika	Verma	80000	2014-06-11	Admin
3	Vishal	Singhal	300000	2014-02-20	HR
4	Amitabh	Singh	500000	2014-02-20	Admin
5	Vivek	Bhati	500000	2014-06-11	Admin

Below the result grid, there is an 'Action Output' section showing the history of actions taken:

#	Time	Action	Message
10	00:32:24	SELECT * FROM STUDENT_INFO LIMIT 0, 1000	4 row(s) returned
11	00:33:53	CREATE TABLE WORKER (WORKER_ID int(5), FIRST_NAME VARCHAR(15), LAST_NAME VARCHAR(15), SALARY int(10), JOINING_DATE DATE, DEPARTMENT VARCHAR(10))	0 row(s) affected
12	00:34:30	CREATE TABLE WORKER (WORKER_ID int, FIRST_NAME VARCHAR(15), LAST_NAME VARCHAR(15), SALARY int, JOINING_DATE DATE, DEPARTMENT VARCHAR(10))	Error Code: 1
13	00:36:12	INSERT INTO WORKER VALUE (1, 'Monika', 'Arora', 100000, '2014-02-20', 'HR'), (2, 'Niharika', 'Verma', 80000, '2014-06-11', 'Admin'), (3, 'Vishal', 'Singhal', 300000, '2014-02-20', 'HR'), (4, 'Amitabh', 'Singh', 500000, '2014-02-20', 'Admin'), (5, 'Vivek', 'Bhati', 500000, '2014-06-11', 'Admin')	8 row(s) affected
14	00:36:24	SELECT * FROM WORKER LIMIT 0, 1000	8 row(s) returned

### 3. Sample Table – Bonus

WORKER_ID	BONUS_DATE	BONUS_AMOUNT
1	2016-02-20	5000
2	2016-06-11	3000
3	2016-02-20	4000

1	2016-02-20	4500
2	2016-06-11	3500

```

25      FOREIGN KEY(Worker_ID) REFERENCES
26      worker(Worker_ID) ON DELETE CASCADE ;
27 •  INSERT INTO worker VALUES (1,'monika','arora',100000,'2014-02-20','hr'),(2,'nih
28      (3,'vishal','singhal',300000,'2014-02-20','hr'),(4,'amithabh','singh',500000,'2
29      (5,'vivek','bhati',500000,'2014-06-11','admin'),
30      (6,'vipul','diwan',200000,'2014-06-11','account'),(7,'satich','kumar',75000,'20
31 •  select *from worker;
32 •  INSERT INTO bonus VALUES (1,'2016-02-20',5000),(2,'2016-06-11',3000),(3,'2016-0
33      (2,'2016-06-11',3500);
34 •  select *from bonus;
```

Result Grid		
Worker_ID	bonus_date	bonus_amount
1	2016-02-20	5000
2	2016-06-11	3000
3	2016-02-20	4000
1	2016-02-20	4500
2	2016-06-11	3500

4. Write An SQL Query To Fetch  
 “FIRST\_NAME” From Worker Table Using The  
 Alias Name As <WORKER\_NAME>.

## Output

```

26      worker(Worker_ID) ON DELETE CASCADE ;
27 •  INSERT INTO worker VALUES (1,'monika','arora',100000,'2014-02-20','hr'),(2,
28      (3,'vishal','singhal',300000,'2014-02-20','hr'),(4,'amithabh','singh',500000,
29      (5,'vivek','bhati',500000,'2014-06-11','admin'),
30      (6,'vipul','diwan',200000,'2014-06-11','account'),(7,'satich','kumar',75000,'2016-06-11','hr');
31 •  select *from worker;
32 •  INSERT INTO bonus VALUES (1,'2016-02-20',5000),(2,'2016-06-11',3000),(3,'2016-06-11',3500);
33 •  select *from bonus;
34 •  SELECT first_name AS worker_name FROM worker;
```

Result Grid		
worker_name		
monika		
niharika		
vishal		
amithabh		
vivek		
vipul		
satich		

5. Write An SQL Query To Print All Worker Details From The Worker Table Order By FIRST\_NAME Ascending

### Output

```
27 •   INSERT INTO worker VALUES (1,'monika','arora',100000,'2014-02-20','hr'),(2,
28     (3,'vishal','singhal',300000,'2014-02-20','hr'),(4,'amithabh','singh',50000
29     (5,'vivek','bhati',500000,'2014-06-11','admin'),
30     (6,'vipul','diwan',200000,'2014-06-11','account'),(7,'satich','kumar',75000
31 •   select *from worker;
32 •   INSERT INTO bonus VALUES (1,'2016-02-20',5000),(2,'2016-06-11',3000),(3,'20
33     (2,'2016-06-11',3500);
34 •   select *from bonus;
35 •   SELECT first_name AS worker_name FROM worker;
36 •   SELECT * FROM worker ORDER BY trim(first_name) ASC ;
```

	Worker_ID	first_name	last_name	salary	joining_date	department
▶	4	amithabh	singh	500000	2014-02-20	admin
	1	monika	arora	100000	2014-02-20	hr
	2	niharika	verma	80000	2014-06-11	admin
	7	satich	kumar	75000	2014-01-20	admin
	6	vipul	diwan	200000	2014-06-11	account
	3	vishal	singhal	300000	2014-02-20	hr
	5	vivek	bhati	500000	2014-06-11	admin
*	NULL	NULL	NULL	NULL	NULL	NULL

6. Write An SQL Query To Print Details Of Workers Excluding First Names, “Vipul” And “Satish” From Worker Table.

### Output

```

28      (3,'vishal','singhal',300000,'2014-02-20','hr'),(4,'amithabh','singh',50000,
29      (5,'vivek','bhati',500000,'2014-06-11','admin'),
30      (6,'vipul','diwan',200000,'2014-06-11','account'),(7,'satich','kumar',75000,
31 •   select *from worker;
32 •   INSERT INTO bonus VALUES (1,'2016-02-20',5000),(2,'2016-06-11',3000),(3,'2016-02-20',3500);
33      (2,'2016-06-11',3500);
34 •   select *from bonus;
35 •   SELECT first_name AS worker_name FROM worker;
36 •   SELECT * FROM worker ORDER BY trim(first_name) ASC ;
37 •   SELECT * FROM worker WHERE trim(first_name) != 'vipul' AND trim(first_name)

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

Worker_ID	first_name	last_name	salary	joining_date	department
1	monika	arora	100000	2014-02-20	hr
2	niharika	verma	80000	2014-06-11	admin
3	vishal	singhal	300000	2014-02-20	hr
4	amithabh	singh	500000	2014-02-20	admin
5	vivek	bhati	500000	2014-06-11	admin
*	NULL	NULL	NULL	NULL	NULL

## 7. Write An SQL Query To Print Details Of Workers With DEPARTMENT Name As “Admin”.

### Output

```

29      (5,'vivek','bhati',500000,'2014-06-11','admin'),
30      (6,'vipul','diwan',200000,'2014-06-11','account'),(7,'satich','kumar',75000,
31 •   select *from worker;
32 •   INSERT INTO bonus VALUES (1,'2016-02-20',5000),(2,'2016-06-11',3000),(3,'2016-02-20',3500);
33      (2,'2016-06-11',3500);
34 •   select *from bonus;
35 •   SELECT first_name AS worker_name FROM worker;
36 •   SELECT * FROM worker ORDER BY trim(first_name) ASC ;
37 •   SELECT * FROM worker WHERE trim(first_name) != 'vipul' AND trim(first_name)
38 •   select *from worker where department='admin';

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content

Worker_ID	first_name	last_name	salary	joining_date	department
2	niharika	verma	80000	2014-06-11	admin
4	amithabh	singh	500000	2014-02-20	admin
5	vivek	bhati	500000	2014-06-11	admin
7	satich	kumar	75000	2014-01-20	admin
*	NULL	NULL	NULL	NULL	NULL

## 8. Write An SQL Query To Print Details Of The Workers Whose SALARY Lies Between 100000 And 500000

### Output

```

30      (6,'vipul','diwan',200000,'2014-06-11','account'),(7,'satich','kumar',7500
31 •   select *from worker;
32 •   INSERT INTO bonus VALUES (1,'2016-02-20',5000),(2,'2016-06-11',3000),(3,'2
33     (2,'2016-06-11',3500);
34 •   select *from bonus;
35 •   SELECT first_name AS worker_name FROM worker;
36 •   SELECT * FROM worker ORDER BY trim(first_name) ASC ;
37 •   SELECT * FROM worker WHERE trim(first_name) != 'vipul' AND trim(first_name)
38 •   select *from worker where department='admin';
39 •   SELECT * FROM worker WHERE salary BETWEEN 100000 AND 500000;

```

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

	Worker_ID	first_name	last_name	salary	joining_date	department
▶	1	monika	arora	100000	2014-02-20	hr
	3	vishal	singhal	300000	2014-02-20	hr
	4	amithabh	singh	500000	2014-02-20	admin
	5	vivek	bhati	500000	2014-06-11	admin
*	6	vipul	diwan	200000	2014-06-11	account
*	NULL	NULL	NULL	NULL	NULL	NULL

9. Write An SQL Query To Fetch  
“FIRST\_NAME” From Worker Table In Upper  
Case. (upper())

## Output

31 • select \*from worker;
32 • INSERT INTO bonus VALUES (1,'2016-02-20',5000),(2,'2016-06-11',3000)
33 (2,'2016-06-11',3500);
34 • select \*from bonus;
35 • SELECT first\_name AS worker\_name FROM worker;
36 • SELECT \* FROM worker ORDER BY trim(first\_name) ASC ;
37 • SELECT \* FROM worker WHERE trim(first\_name) != 'vipul' AND trim(first\_name)
38 • select \*from worker where department='admin';
39 • SELECT \* FROM worker WHERE salary BETWEEN 100000 AND 500000;
40 • SELECT upper(first\_name) FROM worker;

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

	upper(first_name)
▶	MONIKA
	NIHARIKA
	VISHAL
	AMITHABH
	VIVEK
	VIPUL
	SATICH

10. Write An SQL Query To Fetch Unique  
Values Of DEPARTMENT From Worker Table.

## Output

```
32 •    INSERT INTO bonus VALUES (1,'2016-02-20',5000),(2,'2016-06-11',3000
33     (2,'2016-06-11',3500);
34 •    select *from bonus;
35 •    SELECT first_name AS worker_name FROM worker;
36 •    SELECT * FROM worker ORDER BY trim(first_name) ASC ;
37 •    SELECT * FROM worker WHERE trim(first_name) != 'vipul' AND trim(fir
38 •    select *from worker where department='admin';
39 •    SELECT * FROM worker WHERE salary BETWEEN 100000 AND 500000;
40 •    SELECT upper(first_name) FROM worker;
41 •    SELECT distinct department FROM worker;
```

<   Result Grid   Filter Rows:   Export:   Wrap Cell Content:   >	
Result Grid	
Filter Rows:	[ ]
Export:	[ ]
Wrap Cell Content:	[ ]
department	
hr	
admin	
account	

11. Write An SQL Query To Print First Three Characters Of FIRST\_NAME From Worker Table.(substring())

## Output

```
33     (2,'2016-06-11',3500);
34 •    select *from bonus;
35 •    SELECT first_name AS worker_name FROM worker;
36 •    SELECT * FROM worker ORDER BY trim(first_name) ASC ;
37 •    SELECT * FROM worker WHERE trim(first_name) != 'vipul' AND trim(first_nam
38 •    select *from worker where department='admin';
39 •    SELECT * FROM worker WHERE salary BETWEEN 100000 AND 500000;
40 •    SELECT upper(first_name) FROM worker;
41 •    SELECT distinct department FROM worker;
42 •    SELECT SUBSTR(first_name, 1, 3) AS small FROM worker;
```

<   Result Grid   Filter Rows:   Export:   Wrap Cell Content:   >	
Result Grid	
Filter Rows:	[ ]
Export:	[ ]
Wrap Cell Content:	[ ]
small	
mon	
nih	
vis	
ami	
viv	
vip	
sat	

12. Write An SQL Query To Print The FIRST\_NAME From Worker Table After Removing White Spaces From The Right Side( RTRIM ( ))

### Output

```
34 •    select *from bonus;
35 •    SELECT first_name AS worker_name FROM worker;
36 •    SELECT * FROM worker ORDER BY trim(first_name) ASC ;
37 •    SELECT * FROM worker WHERE trim(first_name) != 'vipul' AND trim(f
38 •    select *from worker where department='admin';
39 •    SELECT * FROM worker WHERE salary BETWEEN 100000 AND 500000;
40 •    SELECT upper(first_name) FROM worker;
41 •    SELECT distinct department FROM worker;
42 •    SELECT SUBSTR(first_name, 1, 3) AS small FROM worker;
43 •    SELECT rtrim(first_name) FROM worker;
```

rtrim(first_name)
monika
niharika
vishal
amithabh
vivek
vipul
satich

13. Write An SQL Query To Print The DEPARTMENT From Worker Table After Removing White Spaces From The Left Side. ( LTRIM ( ))

### Output

```

35 •   SELECT first_name AS worker_name FROM worker;
36 •   SELECT * FROM worker ORDER BY trim(first_name) ASC ;
37 •   SELECT * FROM worker WHERE trim(first_name) != 'vipul' AND trim(firs
38 •   select *from worker where department='admin';
39 •   SELECT * FROM worker WHERE salary BETWEEN 100000 AND 500000;
40 •   SELECT upper(first_name) FROM worker;
41 •   SELECT distinct department FROM worker;
42 •   SELECT SUBSTR(first_name, 1, 3) AS small FROM worker;
43 •   SELECT rtrim(first_name) FROM worker;
44 •   SELECT ltrim(department) FROM worker;

```

Result Grid	
	ltrim(department)
▶	hr
	admin
▶	hr
	admin
	admin
	account
	admin

14. Write An SQL Query That Fetches The Unique Values Of DEPARTMENT From Worker Table And Prints Its Length.( length())

## Output

```

36 •   SELECT * FROM worker ORDER BY trim(first_name) ASC ;
37 •   SELECT * FROM worker WHERE trim(first_name) != 'vipul' AND trim
38 •   select *from worker where department='admin';
39 •   SELECT * FROM worker WHERE salary BETWEEN 100000 AND 500000;
40 •   SELECT upper(first_name) FROM worker;
41 •   SELECT distinct department FROM worker;
42 •   SELECT SUBSTR(first_name, 1, 3) AS small FROM worker;
43 •   SELECT rtrim(first_name) FROM worker;
44 •   SELECT ltrim(department) FROM worker;
45 •   SELECT distinct department, LENGTH(department) FROM worker ;

```

Result Grid		
	department	LENGTH(department)
▶	hr	2
	admin	5
	account	7

15. Write An SQL Query To Print The FIRST\_NAME From Worker Table After Replacing ‘a’ With ‘A’.( REPLACE( ))

## Output

```
37 •   SELECT * FROM worker WHERE trim(first_name) != 'vipul' AND trim(first_
38 •   select *from worker where department='admin';
39 •   SELECT * FROM worker WHERE salary BETWEEN 100000 AND 500000;
40 •   SELECT upper(first_name) FROM worker;
41 •   SELECT distinct department FROM worker;
42 •   SELECT SUBSTR(first_name, 1, 3) AS small FROM worker;
43 •   SELECT rtrim(first_name) FROM worker;
44 •   SELECT ltrim(department) FROM worker;
45 •   SELECT distinct department, LENGTH(department) FROM worker ;
46 •   SELECT REPLACE(first_name, 'a', 'A') AS fname FROM worker;
```

fname
monikA
nihArikA
vishAl
AmithAbh
vivek
vipul
sAtich

16. Find the First name , last name ,Department, Salary and Bonus of employees whose bonus amount is greater than 4000

## Output

```

38 •   select *from worker where department='admin';
39 •   SELECT * FROM worker WHERE salary BETWEEN 100000 AND 500000;
40 •   SELECT upper(first_name) FROM worker;
41 •   SELECT distinct department FROM worker;
42 •   SELECT SUBSTR(first_name, 1, 3) AS small FROM worker;
43 •   SELECT rtrim(first_name) FROM worker;
44 •   SELECT ltrim(department) FROM worker;
45 •   SELECT distinct department, LENGTH(department) FROM worker ;
46 •   SELECT REPLACE(first_name, 'a', 'A') AS fname FROM worker;
47 •   SELECT first_name,last_name,department,salary,bonus_amount  FROM worker,b

```

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

	first_name	last_name	department	salary	bonus_amount
▶	monika	arora	hr	100000	5000
	monika	arora	hr	100000	4500

17. Delete the employee with worker\_id=7 from worker and display the details of both tables.

## Output

```

40 •   SELECT upper(first_name) FROM worker;
41 •   SELECT distinct department FROM worker;
42 •   SELECT SUBSTR(first_name, 1, 3) AS small FROM worker;
43 •   SELECT rtrim(first_name) FROM worker;
44 •   SELECT ltrim(department) FROM worker;
45 •   SELECT distinct department, LENGTH(department) FROM worker ;
46 •   SELECT REPLACE(first_name, 'a', 'A') AS fname FROM worker;
47 •   SELECT first_name,last_name,department,salary,bonus_amount  FROM worker;
48 •   DELETE from worker WHERE worker_id=7;
49 •   SELECT * FROM worker;

```

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

	Worker_ID	first_name	last_name	salary	joining_date	department
1	monika	arora	100000	2014-02-20	hr	
2	niharika	verma	80000	2014-06-11	admin	
3	vishal	singhal	300000	2014-02-20	hr	
4	amithabh	singh	500000	2014-02-20	admin	
5	vivek	bhati	500000	2014-06-11	admin	
6	vipul	diwan	200000	2014-06-11	account	
*	NULL	NULL	NULL	NULL	NULL	NULL

18. Drop the foreign key constraint and add a new referential integrity constraint with ‘on update or delete with no action’

## Output

```

41 •   SELECT DISTINCT department FROM worker;
42 •   SELECT SUBSTR(first_name, 1, 3) AS small FROM worker;
43 •   SELECT rtrim(first_name) FROM worker;
44 •   SELECT ltrim(department) FROM worker;
45 •   SELECT distinct department, LENGTH(department) FROM worker ;
46 •   SELECT REPLACE(first_name, 'a', 'A') AS fname FROM worker;
47 •   SELECT first_name, last_name, department, salary, bonus_amount FROM worker;
48 •   DELETE from worker WHERE worker_id=7;
49 •   SELECT * FROM worker;
50 •   ALTER TABLE bonus DROP CONSTRAINT fk_cod_csd;
51 •   ALTER TABLE bonus ADD CONSTRAINT fk_cod_na FOREIGN KEY(Worker_ID)
      REFERENCES worker(Worker_ID) ON DELETE no action ;
52
53

```

Output			
#	Time	Action	Message
✓ 1	09:27:16	SELECT distinct cust_name, COUNT(item_id), bill_date FROM customer c, sales s WHERE c....	3 row(s) returned
✗ 2	09:42:49	ALTER TABLE bonus ADD CONSTRAINT fk_cod_na FOREIGN KEY(Worker_ID) REFEREN...	Error Code: 1826. Duplicate foreign key constraint
✓ 3	09:43:36	ALTER TABLE bonus ADD CONSTRAINT fk_cod_nai FOREIGN KEY(Worker_ID) REFERE...	5 row(s) affected Records: 5 Duplicates: 0 W

19. Delete the employee with worker\_id = 8 from worker.

## Output

```

46 •   SELECT DISTINCT department, LENGTH(department) FROM worker
47 •   SELECT REPLACE(first_name, 'a', 'A') AS fname FROM worker
48 •   SELECT first_name, last_name, department, salary, bonus_amount
49 •   DELETE from worker WHERE worker_id=7;
50 •   SELECT * FROM worker;
51 •   ALTER TABLE bonus DROP CONSTRAINT fk_cod_csd;
52 •   ALTER TABLE bonus ADD CONSTRAINT fk_cod_na FOREIGN KEY(
53     REFERENCES worker(Worker_ID) ON DELETE no action ;
54 •   DELETE from worker WHERE worker_id=8;
55 •   SELECT * FROM worker;

```

Result Grid | Filter Rows:  Edit: Export/Import

	Worker_ID	first_name	last_name	salary	joining_date	department
▶	1	monika	arora	100000	2014-02-20	hr
	2	niharika	verma	80000	2014-06-11	admin
	3	vishal	singhal	300000	2014-02-20	hr
	4	amithabh	singh	500000	2014-02-20	admin
	5	vivek	bhati	500000	2014-06-11	admin
*	6	vipul	diwan	200000	2014-06-11	account
	NULL	NULL	NULL	NULL	NULL	NULL

### Q.SET 3

Create the tables given below and execute the queries:

**Customer(Cust id : integer, cust\_name: string)**

**Item(item\_id: integer, item\_name: string, price: integer)**

**Sale(bill\_no: integer, bill\_date: date, cust\_id: integer, item\_id: integer, qty\_sold: integer)**

For the above schema, perform the following—

- a) Create the tables with the appropriate integrity constraints

**Output**

```

2 • CREATE TABLE CUSTOMER(CUST_ID INT PRIMARY KEY,CUST_NAME VARCHAR(10));
3 • CREATE TABLE ITEM(ITEM_ID INT PRIMARY KEY,ITEM_NAME VARCHAR(20),PRICE INT);
4 • CREATE TABLE SALE(BILL_NO INT PRIMARY KEY,BILL_DATE DATE,CUST_ID INT,ITEM_ID
5   CONSTRAINT FK_CUST FOREIGN KEY(CUST_ID) REFERENCES CUSTOMER(CUST_ID),CONSTRAINT
6

```

Input : Action Output

#	Time	Action	Message
43	01:28:03	drop database sq2set3	3 row(s) affected
44	01:28:47	create database sq2set3	1 row(s) affected
45	01:29:40	CREATE TABLE CUSTOMER(CUST_ID INT PRIMARY KEY,CUST_NAME VARCHAR(10))	0 row(s) affected
46	01:29:49	CREATE TABLE ITEM(ITEM_ID INT PRIMARY KEY,ITEM_NAME VARCHAR(20),PRICE INT)	0 row(s) affected
47	01:29:59	CREATE TABLE SALE(BILL_NO INT PRIMARY KEY,BILL_DATE DATE,CUST_ID INT,ITEM_ID	0 row(s) affected

- b) Insert details of 5 customers, 5 items and 10 sales details. There should be one customer 'rekha' who had purchased 3 different products on the same date. And there should be atleast one customer who had purchased 2 different products on the same date in the year '2018'.

## Output

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** A tree view showing the structure of the database, including the 'customers' schema which contains tables like 'customer', 'item', and 'sale'.
- SQL File 2:** A tab containing SQL code for creating tables and inserting data into them.
- Result Grid:** A table displaying the data inserted into the 'sale' table.
- Action Output:** A log of actions taken, including the creation of the database and tables, and the insertion of data.

**SQL File 2 Content:**

```

26 • select * from customer;
27 • INSERT INTO item VALUES( 1, 'Rusk', 120),(2,'banana',50),(3,'handwash',60),(4,'cake',420),(5,'sweets',25);
28 • select * from item;
29 • INSERT INTO sale VALUES ( 10, '2020-10-01', 1, 1, 3),( 11, '2020-10-01', 1, 3, 2),
30   ( 12, '2020-10-01', 1, 5, 7),( 13, '2018-10-01', 4, 4, 1),
31   ( 14, '2018-10-11', 4, 2, 2),( 15, '2018-09-29', 5, 1, 5),( 16, '2019-12-25', 3, 1, 5),( 17, '1995-06-21', 5, 1),
32   ( 18, '2002-04-01', 4, 5, 5),( 19, '2020-02-12', 1, 2, 1);
33 • select * from sale;
34 • SELECT * FROM item WHERE price > 200 AND sale.item_id = item.item_id AND

```

**Result Grid Data:**

bill_no	bill_date	cust_id	item_id	qty_sold
10	2020-10-01	1	1	3.000
11	2020-10-01	1	3	2.000
12	2020-10-01	1	5	7.000
13	2018-10-01	4	4	1.000
14	2018-10-11	4	2	2.000
15	2018-09-29	5	1	5.000
16	2019-12-25	3	1	5.000
17	1995-06-21	5	4	4.000
18	2002-04-01	4	5	5.000
19	2020-02-12	1	2	1.000

**Action Output Log:**

#	Time	Action	Message
17	17:39:49	create database customers	Error Code: 1007. Can't create database 'customers'
18	17:39:13	INSERT INTO item VALUES( 1, 'Rusk', 120),(2,'banana',50),(3,'handwash',60),(4,'cake',420),(5,'sweets',25)	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
19	17:39:16	select * from item LIMIT 0, 1000	5 row(s) returned
20	17:39:43	INSERT INTO sale VALUES ( 10, '2020-10-01', 1, 1, 3),( 11, '2020-10-01', 1, 3, 2), ( 12, '2020-10-01', 1, 5, 7)	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0
21	17:39:47	create database customers	Error Code: 1007. Can't create database 'customers'
22	17:39:53	select * from sale LIMIT 0, 1000	10 row(s) returned

- c) List the details of the customer who have bought a product which has a price>200 .

CODE:

```

SELECT *FROM customer,item,sale WHERE
price > 200 AND sale.item_id = item.item_id
AND
sale.cust_id = customer.cust_id;

```

## Output

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema with the 'customers' schema selected.
- SQL Editor:** Contains the SQL query provided above.
- Result Grid:** Displays the output of the query, showing two rows of data:

cust_id	cust_name	item_id	item_name	price	bill_no	bill_date	cust_id	item_id	qty_sold
4	nirmal	4	cake	420	13	2019-10-01	4	4	1.000
5	Jasmine	4	cake	420	17	1995-06-21	5	4	4.000

- Action Output:** Shows the history of actions taken in the session, including insertions into the 'item' and 'sale' tables, and the creation of the 'customers' database.

- d) Give a count of how many products have been bought by each customer group by bill date.

## Output

```

15 •   SELECT C.CUST_ID,CUST_NAME,SC.COUNT,SC.BILL_DATE FROM CUSTOMER C,
16   (SELECT COUNT(ITEM_ID) AS COUNT,BILL_DATE,CUST_ID FROM SALE GROUP BY BILL_DATE)

```

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

	CUST_ID	CUST_NAME	COUNT	BILL_DATE
▶	1	Rekha	4	2020-10-01
▶	1	Rekha	1	2020-02-12
▶	3	Renju	1	2018-12-25
▶	4	renjini	1	2018-10-01
▶	4	renjini	1	2018-10-11

Result 6 × Read Only

Output

Action Output

#	Time	Action
57	01:37:28	SELECT * FROM ITEM LIMIT 0, 1000
58	01:38:52	INSERT INTO SALE(BILL_NO,BILL_DATE,CUST_ID,ITEM_ID,QTY_SOLID) VALUES(10,'2020-10-01',1,1,1)
59	01:39:53	SELECT * FROM SALE LIMIT 0, 1000
60	01:42:05	SELECT * FROM CUSTOMER C,ITEM I,SALE S WHERE C.CUST_ID=S.CUST_ID AND I.ITEM_ID=S.ITEM_ID
61	01:43:30	SELECT C.CUST_ID,CUST_NAME,SC.COUNT,SC.BILL_DATE FROM CUSTOMER C, (SELECT * FROM CUSTOMER C,ITEM I,SALE S WHERE C.CUST_ID=S.CUST_ID AND I.ITEM_ID=S.ITEM_ID) AS T GROUP BY C.CUST_ID,SC.BILL_DATE HAVING extract(YEAR FROM SC.BILL_DATE)=2018;

- e) Give a count of how many products have been bought by each customer group by bill date only for the year 2018.

#### CODE:

```

SELECT      cust_name,      COUNT(item_id),
bill_date  FROM customer c,  sale s WHERE
c.cust_id = s.cust_id
GROUP BY cust_name,    bill_date HAVING
extract(YEAR FROM bill_date) = 2018;

```

#### Output

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Schemas:** A tree view showing the 'customers' schema expanded, containing Tables, Views, Stored Procedures, and Functions.
- SQL Editor:** Contains a multi-line SQL query for selecting items from the 'item' and 'sale' tables based on specific criteria like price > 200 and bill\_date between 2018-04-01 and 2020-02-12.
- Result Grid:** Displays the results of the query, showing columns: cust\_name, COUNT(item\_id), and bl\_date. The data includes rows for 'nirmal' (1 item, 2018-10-01, 2018-10-11) and 'jasmine' (1 item, 2018-09-29).
- Output Window:** Shows the execution history of the session, including the creation of the 'customers' database and various SELECT statements.

- f) Give a list of products bought by a customer having cust\_id as 5

## CODE:

```
SELECT item_name FROM item, sale WHERE  
sale.item_id = item.item_id AND sale.cust_id  
= 5;
```

## Output

The screenshot shows the MySQL Workbench interface. The top navigation bar includes File, Edit, View, Query, Database, Server, Tools, Scripting, and Help. Below the menu is a toolbar with various icons for database management. The left sidebar, titled 'Navigator', displays the database schema with 'customers' as the selected schema. It lists tables like 'customer', 'item', 'sale', and 'order'. The main area contains a query editor with the following SQL code:

```

31      ( 14, '2018-10-11', 4, 2, 2), ( 15, '2018-09-29', 5, 3, 5),( 16, '2018-12-25', 3, 1, 5),( 17, '1995-06-21', 5, 1, 5),
32      ( 18, '2002-04-01', 4, 5, 5),( 19, '2020-02-12', 1, 2, 1);
33  •      select * from sale;
34  •      SELECT * FROM customer,item,sale WHERE price > 200 AND sale.item_id = item.item_id AND
35      sale.cust_id = customer.cust_id;
36  •      SELECT cust_name, COUNT(item_id), bill_date FROM customer c, sale s WHERE c.cust_id = s.cust_id
37      GROUP BY cust_name, bill_date HAVING extract(YEAR FROM bill_date) = 2018;
38  •      SELECT item_name FROM item, sale WHERE sale.item_id = item.item_id AND sale.cust_id = 5;
39  •      CREATE item_name < http://data.com/TTM?T=CAFE<NAME>T item_id< item_id & name < http://data.MYBROWZER>.
```

The 'Result Grid' tab is active, showing the results of the last query:

item_name
Rusk
cake

Below the result grid, there are 'Filter Rows:' and 'Wrap Cell Content:' buttons. The bottom right corner has a 'Read Only' button. The status bar at the bottom shows 'Object Info' and 'Session'.

g) List the item details which are sold as of today

CODE:

```
SELECT item_name,S.bill_date FROM ITEM I,SALE S  
WHERE I.item_id=S.item_id AND  
S.bill_date=CURDATE();
```

Output

20 • `SELECT * FROM ITEM I,SALE S WHERE I.ITEM_ID=S.ITEM_ID AND S.BILL_DATE=CURDATE();`

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar is a results grid window titled "Result Grid" with two columns: "ITEM\_NAME" and "BILL\_DATE". The results grid is currently empty. Below the results grid is an "Output" pane titled "Action Output" which lists several database actions with their times and descriptions. The last action listed is the execution of the query from the previous step.

#	Time	Action	Message
60	01:42:05	SELECT * FROM CUSTOMER C,ITEM I,SALE S WHERE C.CUST_ID=S.CUST_ID AND I.I...	2 rows(s)
61	01:43:30	SELECT C.CUST_ID,CUST_NAME,SC.COUNT,SC.BILL_DATE FROM CUSTOMER C, (SE...	8 rows(s)
62	01:45:33	SELECT C.CUST_ID,CUST_NAME,SC.COUNT,SC.BILL_DATE FROM CUSTOMER C, (SE...	4 rows(s)
63	01:47:19	SELECT ITEM_NAME,I.ITEM_ID FROM ITEM I,CUSTOMER C,SALE S WHERE S.CUST_I...	2 rows(s)
64	01:49:00	SELECT ITEM_NAME,S.BILL_DATE FROM ITEM I,SALE S WHERE I.ITEM_ID=S.ITEM_I...	0 rows(s)

h) Print the bill in a neat format with the quantity sold, price of the item and the final amount of customer 'rekha'

CODE:

```
SELECT item_name,qty_sold,    price,(qty_sold *  
price) AS total_amount FROM customer, item, sale  
WHERE cust_name = 'Rekha' AND sale.item_id =  
item.item_id AND sale.cust_id = customer.cust_id;
```

Output

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Local instance MySQL80

SCHEMAS: customers

```

SELECT * FROM customer, item, sale WHERE price > 200 AND sale.item_id = item.item_id AND
sale.cust_id = customer.cust_id;
SELECT cust_name, COUNT(item_id), bill_date FROM customer c, sale s WHERE c.cust_id = s.cust_id
GROUP BY cust_name, bill_date HAVING EXTRACT(YEAR FROM bill_date) = 2018;
SELECT item_name FROM item, sale WHERE sale.item_id = item.item_id AND sale.cust_id = 5;
SELECT item_name, S.bill_date FROM ITEM I, SALE S WHERE I.item_id=S.item_id AND S.bill_date=CURDATE();
SELECT item_name,qty_sold, price,(qty_sold * price) AS total_amount FROM customer, item, sale
WHERE cust_name = 'Ruskin' AND sale.item_id = item.item_id AND sale.cust_id = customer.cust_id;
    
```

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

item_name	qty_sold	price	total_amount
Rusk	3.000	120	360.000
handwash	2.000	60	120.000
sweets	7.000	25	175.000
banana	1.000	50	50.000

Schema: customers

Result 7 | Read Only

Action Output

#	Time	Action	Message
22	17:39:53	selected * from sale LIMIT 0, 1000	10 row(s) returned
23	17:41:04	SELECT * FROM customer, item, sale WHERE price > 200 AND sale.item_id = item.item_id AND sale.cust_id = customer.cust_id;	2 row(s) returned
24	17:43:38	SELECT cust_name, COUNT(item_id), bill_date FROM customer c, sale s WHERE c.cust_id = s.cust_id GROUP BY cust_name, bill_date HAVING EXTRACT(YEAR FROM bill_date) = 2018;	3 row(s) returned
25	17:46:41	SELECT item_name FROM item, sale WHERE sale.item_id = item.item_id AND sale.cust_id = 5 LIMIT 0, 1000	2 row(s) returned
26	17:48:10	SELECT item_name, S.bill_date FROM ITEM I, SALE S WHERE I.item_id=S.item_id AND S.bill_date=CURDATE();	0 row(s) returned
27	17:49:12	SELECT item_name,qty_sold, price,(qty_sold * price) AS total_amount FROM customer, item, sale WHERE cust_name = 'Ruskin' AND sale.item_id = item.item_id AND sale.cust_id = customer.cust_id;	4 row(s) returned

Action Info Session

# CYCLE 3

## ***QUESTION SET 3***

Create the following tables.

- Primary key, SSN of EMPLOYEE should be created as a sequence starting at 1.
- There should be at least 8 employees and 5 departments
- Check salary range of employees is between 30,000 and 75,000 using check predicate.

### **EMPLOYEE**

Column	Constraint	Data Type	Remarks
SSN	PRIMARY KEY	NUMBER	Employee Number
ENAME	NOT NULL	CHARACTER	Employee Name
DESIG	---	CHARACTER	Designation
DNO	FOREIGN KEY (DEPARTMENT)	NUMBER	Dept. Number
DOJ	---	DATE	Date of Join
SALARY	---	NUMBER	Basic Salary

### **DEPARTMENT**

Column	Constraint	Data Type	Remarks
DNUMBER	PRIMARY KEY	NUMBER	Department Number
DNAME	NOT NULL	CHARACTER	Department Name
LOC	---	CHARACTER	Dept. Location
MGRSSN	FOREIGN KEY (EMPLOYEE)	NUMBER	Dept. Manager Number

### **PROJECT**

Column	Constraint	Data Type	Remarks
PNUMBER	PRIMARY KEY	NUMBER	Project Number
PNAME	NOT NULL	CHARACTER	Project Name
DNUM	FOREIGN KEY (DEPARTMENT)	NUMBER	Dept. Number

### **WORKS\_IN**

Column	Constraint	Data Type	Remarks
ESSN	FOREIGN KEY (EMPLOYEE)	NUMBER	Employee Number
PNO	FOREIGN KEY (PROJECT)	NUMBER	Project Number
HOURS	FOREIGN KEY (DEPARTMENT)	NUMBER	Total Hours

1. Retrieve all employees in department 5 whose salary is between Rs 30,000 and Rs 40,000.

### Output

```

15 • UPDATE department SET mgrssn=2 WHERE dnumber=1;
16 • UPDATE department SET mgrssn=1 WHERE dnumber=2;
17 • UPDATE department SET mgrssn=3 WHERE dnumber=3;
18 • UPDATE department SET mgrssn=6 WHERE dnumber=4;
19 • UPDATE department SET mgrssn=7 WHERE dnumber=2;
20 • UPDATE department SET mgrssn=4 WHERE dnumber=5;
21 • UPDATE department SET mgrssn=5 WHERE dnumber=5;
22 • insert into Proj(pnumber,pname,dnum)values(11,'Bancs Trsry',3),(12,'Nielesan',5),(13,'World Bnk',1),(14,'Airlines',1),(15,'Amex',4);
23 • insert into Work_in(esn,pno)values(1,14),(4,13),(8,12),(6,15),(2,11),(3,13);
24 • select * from Work_in;
25 • SELECT e.ename FROM employee e LEFT OUTER JOIN department d on d.dnumber=e.dno WHERE e.salary BETWEEN 30000 AND 40000 AND d.dnumber=5;
26

```

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

ename
Diya
Govind

2. Retrieve a list of employees and the projects they are working on, where the departments and the employees within the department are alphabetically by name.

### Output

```

15 • UPDATE department SET mgrssn=2 WHERE dnumber=1;
16 • UPDATE department SET mgrssn=1 WHERE dnumber=2;
17 • UPDATE department SET mgrssn=3 WHERE dnumber=3;
18 • UPDATE department SET mgrssn=6 WHERE dnumber=4;
19 • UPDATE department SET mgrssn=7 WHERE dnumber=2;
20 • UPDATE department SET mgrssn=4 WHERE dnumber=5;
21 • UPDATE department SET mgrssn=5 WHERE dnumber=5;
22 • insert into Proj(pnumber,pname,dnum)values(11,'Bancs Trsry',3),(12,'Nielesan',5),(13,'World Bnk',1),(14,'Airlines',1),(15,'Amex',4);
23 • insert into Work_in(esn,pno)values(1,14),(4,13),(8,12),(6,15),(2,11),(3,13);
24 • select * from Work_in;
25 • SELECT e.ename FROM employee e LEFT OUTER JOIN department d on d.dnumber=e.dno WHERE e.salary BETWEEN 30000 AND 40000 AND d.dnumber=5;
26 • SELECT e.ename,d.dname FROM employee e LEFT OUTER JOIN department d on e.dno=d.dnumber ORDER BY d.dname ASC,e.ename ASC;
27

```

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

ename	dname
Bhama	Admin
Hima	Finance
Hima	Finance
Abhi	HR
Diya	Production
Govind	Production
Hima	Production
Chriz	Sales

3. Retrieve the project number, the project name, and the number of employees who work in each project.

### Output

```

20 • UPDATE department SET mgrssn=4 WHERE dnumber=5;
21 • UPDATE department SET mgrssn=5 WHERE dnumber=5;
22 • insert into Proj(pnumber,pname,dnum)values(11,'Bancs Trsry',3),(12,'Nielesan',5),(13,'World Bnk',1),(14,'Airlines',1),(15,'Amex',4);
23 • insert into Work_in(eSSN,pno)values(1,14),(4,13),(8,12),(6,15),(2,11),(3,13);
24 • select * from Work_in;
25 • SELECT e.ename FROM employee e LEFT OUTER JOIN department d ON d.dnumber=e.dno WHERE e.salary BETWEEN 30000 AND 40000 AND d.dnumber=5;
26 • SELECT e.ename,d.dname FROM employee e LEFT OUTER JOIN department d ON e.dno=d.dnumber ORDER BY d.dname ASC,e.ename ASC;
27 • SELECT p.pnumber,p.pname,COUNT(e.eSSN) FROM Work_in w LEFT OUTER JOIN Proj p ON w.pno=p.pnumber LEFT OUTER JOIN employee e ON w.eSSN=e.eSSN GROUP BY p.pname,p.pnumber;
28
29

```

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

pnumber	pname	count(e.eSSN)
14	Airlines	1
13	World Bnk	2
12	Nielesan	1
15	Amex	1
11	Bancs Trsry	1

4. For the project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.

### Output

```

19 • UPDATE department SET mgrssn=7 WHERE dnumber=2;
20 • UPDATE department SET mgrssn=4 WHERE dnumber=5;
21 • UPDATE department SET mgrssn=5 WHERE dnumber=5;
22 • insert into Proj(pnumber,pname,dnum)values(11,'Bancs Trsry',3),(12,'Nielesan',5),(13,'World Bnk',1),(14,'Airlines',1),(15,'Amex',4);
23 • insert into Work_in(eSSN,pno)values(1,14),(4,13),(8,12),(6,15),(2,11),(3,13);
24 • select * from Work_in;
25 • SELECT e.ename FROM employee e LEFT OUTER JOIN department d ON d.dnumber=e.dno WHERE e.salary BETWEEN 30000 AND 40000 AND d.dnumber=5;
26 • SELECT e.ename,d.dname FROM employee e LEFT OUTER JOIN department d ON e.dno=d.dnumber ORDER BY d.dname ASC,e.ename ASC;
27 • SELECT p.pnumber,p.pname,COUNT(e.eSSN) FROM Work_in w LEFT OUTER JOIN Proj p ON w.pno=p.pnumber LEFT OUTER JOIN employee e ON w.eSSN=e.eSSN GROUP BY p.pname,p.pnumber;
28 • SELECT p.pnumber,p.pname,COUNT(e.eSSN) FROM Work_in w LEFT OUTER JOIN Proj p ON w.pno=p.pnumber LEFT OUTER JOIN employee e ON w.eSSN=e.eSSN GROUP BY p.pname,p.pnumber HAVING COUNT(e.eSSN)>2;

```

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

pnumber	pname	count(e.eSSN)

5. For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.

### Output

```

19 • UPDATE department SET mgrssn=7 WHERE dnumber=2;
20 • UPDATE department SET mgrssn=4 WHERE dnumber=5;
21 • UPDATE department SET mgrssn=5 WHERE dnumber=5;
22 • insert into Proj(pnumber,pname,dnum)values(11,'Bancs Trsry',3),(12,'Nielesan',5),(13,'World Bnk',1),(14,'Airlines',1),(15,'Amex',4);
23 • insert into Work_in(eSSN,pno)values(1,14),(4,13),(8,12),(6,15),(2,11),(3,13);
24 • select * from Work_in;
25 • SELECT e.ename FROM employee e LEFT OUTER JOIN department d ON d.dnumber=e.dno WHERE e.salary BETWEEN 30000 AND 40000 AND d.dnumber=5;
26 • SELECT e.ename,d.dname FROM employee e LEFT OUTER JOIN department d ON e.dno=d.dnumber ORDER BY d.dname ASC,e.ename ASC;
27 • SELECT p.pnumber,p.pname,COUNT(e.eSSN) FROM Work_in w LEFT OUTER JOIN Proj p ON w.pno=p.pnumber LEFT OUTER JOIN employee e ON w.eSSN=e.eSSN GROUP BY p.pname,p.pnumber;
28 • SELECT p.pnumber,p.pname,d.dnumber,COUNT(e.eSSN) FROM Proj p LEFT OUTER JOIN department d ON d.dnumber=p.dnum LEFT OUTER JOIN employee e ON e.dno=p.dnum GROUP BY p.pname,p.pnumber,d.dnumber;

```

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

pnumber	pname	dnumber	count(e.eSSN)
12	Nielesan	5	3

6. For the departments having more than five employees, display the department id and the number and details of employees earning more than Rs 40,000 per month.

## Output

```
21 • UPDATE department SET mgrssn=5 WHERE dnumber=5;
22 • insert into Proj(pnumber,pname,dnum)values(11,'Bancs Trsry',3),(12,'Nileesan',5),(13,'World Bnk',1),(14,'Airlines',1),(15,'Amex',4);
23 • insert into Work_in(ssn,pno)values(1,14),(4,13),(8,12),(6,15),(2,11),(3,13);
24 • select * from Work_in;
25 • SELECT e.ename FROM employee e LEFT OUTER JOIN department d on d.dnumber=e.dno WHERE e.salary BETWEEN 30000 AND 40000 AND d.dnumber=5;
26 • SELECT e.ename,d.dname FROM employee e LEFT OUTER JOIN department d on e.dno=d.dnumber ORDER BY d.dname ASC,e.ename ASC;
27 • SELECT p.pnumber,p.pname,count(e.ssn) FROM Work_in w LEFT OUTER JOIN Proj p on w.pno=p.pnumber LEFT OUTER JOIN employee e on w.ssn=e.ssn GROUP BY p.pname,p.pnumber;
28 • SELECT d.dname,d.dnumber,e.ssn,e.ename,e.design,e.doj,e.salary FROM department d,employee e WHERE
29 (SELECT COUNT(*)) FROM employee e WHERE e.dno = d.dnumber AND e.salary> 40000) > 4 AND e.dno=d.dnumber GROUP BY d.dname,d.dnumber,e.ssn,e.ename,e.design,e.doj,e.salary;
30 •
```

7. Create a synonym for the VIEW created on natural join of emp and dept tables.

## Output

8. Use the tables Employee, and Department. Perform the operations as mentioned below:

- (a) Display the employee details, departments that the departments are same in both the emp and dept. (Equi-join)

## Output

```
22 • insert into Proj(pnumber,pname,dnum)values(11,'Bancs Trsry',3),(12,'Nileesan',5),(13,'World Bnk',1),(14,'Airlines',1),(15,'Amex',4);
23 • insert into Work_in(ssn,pno)values(1,14),(4,13),(8,12),(6,15),(2,11),(3,13);
24 • select * from Work_in;
25 • SELECT e.ename FROM employee e LEFT OUTER JOIN department d on d.dnumber=e.dno WHERE e.salary BETWEEN 30000 AND 40000 AND d.dnumber=5;
26 • SELECT e.ename,d.dname FROM employee e LEFT OUTER JOIN department d on e.dno=d.dnumber ORDER BY d.dname ASC,e.ename ASC;
27 • SELECT p.pnumber,p.pname,count(e.ssn) FROM Work_in w LEFT OUTER JOIN Proj p on w.pno=p.pnumber LEFT OUTER JOIN employee e on w.ssn=e.ssn GROUP BY p.pname,p.pnumber;
28 • create VIEW emp_dept_view as select * from employee NATURAL JOIN department;
29 • CREATE ANY SYNONYM emp_dept FOR emp_dept_view;
30 • select * from emp_dept;
31 • Select * From employee e,department d WHERE e.dno=d.dnumbers;
```

- (b) Display the employee details, departments that the departments are not same in both the emp and dept. (Non Equi-join)

## Output

```

23 • insert into Work_in(ssn,pno)values(1,14),(4,13),(8,12),(6,15),(2,11),(3,13);
24 • select * from Work_in;
25 • SELECT e.ename FROM employee e LEFT OUTER JOIN department d ON d.dnumber=e.dno WHERE e.salary BETWEEN 30000 AND 40000 AND d.dnumber=5;
26 • SELECT e.ename,d.dname FROM employee e LEFT OUTER JOIN department d ON e.dno=d.dnumber ORDER BY d.dname ASC,e.ename ASC;
27 • SELECT p.pnumber,p.pname,count(e.ssn) FROM Work_in w LEFT OUTER JOIN Proj p ON w.pno=p.pnumber LEFT OUTER JOIN employee e ON w.essn=e.ssn GROUP BY p.pname,p.pnumber;
28 • create VIEW emp_dept_view AS select * from employee NATURAL JOIN department;
29 ☐ CREATE ANY SYNONYM emp_dept FOR employee_dept_view;
30 • select * from emp_dept;
31 • Select * From employee e,department d WHERE e.dno=d.dnumber;
32 • SELECT * FROM employee e,department d WHERE NOT(e.dno=d.dnumber);
33 • SELECT * FROM employee e,department d ON e.dno=d.dnumber;

```

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

ssn	ename	dno	doj	salary	dnumber	dname	loc	mgrssn
1	Abhi	HR	2	2009-07-12	70000	5	Production	Thiruvananthapuram
1	Abhi	HR	2	2009-07-12	70000	4	Finance	Delhi
1	Abhi	HR	2	2009-07-12	70000	3	Sales	Kochi
1	Abhi	HR	2	2009-07-12	70000	1	Admin	Chennai
2	Bhama	Admin	1	2008-03-10	75000	5	Production	Thiruvananthapuram
2	Bhama	Admin	1	2008-03-10	75000	4	Finance	Delhi

(c) Perform Left outer join on the emp and dept tables.

### Output

```

24 • select * from Work_in;
25 • SELECT e.ename FROM employee e LEFT OUTER JOIN department d ON d.dnumber=e.dno WHERE e.salary BETWEEN 30000 AND 40000 AND d.dnumber=5;
26 • SELECT e.ename,d.dname FROM employee e LEFT OUTER JOIN department d ON e.dno=d.dnumber ORDER BY d.dname ASC,e.ename ASC;
27 • SELECT p.pnumber,p.pname,count(e.ssn) FROM Work_in w LEFT OUTER JOIN Proj p ON w.pno=p.pnumber LEFT OUTER JOIN employee e ON w.essn=e.ssn GROUP BY p.pname,p.pni
28 • create VIEW emp_dept_view AS select * from employee NATURAL JOIN department;
29 ☐ CREATE ANY SYNONYM emp_dept FOR employee_dept_view;
30 • select * from emp_dept;
31 • Select * From employee e,department d WHERE e.dno=d.dnumber;
32 • SELECT * FROM employee e,department d WHERE NOT(e.dno=d.dnumber);
33 • SELECT * FROM employee e LEFT OUTER JOIN department d ON e.dno=d.dnumber;

```

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

ssn	ename	design	dno	doj	salary	dnumber	dname	loc	mgrssn
1	Abhi	HR	2	2009-07-12	70000	2	HR	Chennai	7
2	Bhama	Admin	1	2008-03-10	75000	1	Admin	Chennai	2
3	Chriz	Sales	3	2011-06-23	35000	3	Sales	Kochi	3
4	Diya	Production	5	2018-09-29	32000	5	Production	Thiruvananthapuram	5
5	Govind	Production	5	1995-06-21	35000	5	Production	Thiruvananthapuram	5
6	Hima	Finance	4	2002-04-01	51000	4	Finance	Delhi	6

(d) Perform Right outer join on the emp and dept tables.

### Output

```

25 • SELECT e.ename FROM employee e LEFT OUTER JOIN department d ON d.dnumber=e.dno WHERE e.salary BETWEEN 30000 AND 40000 AND d.dnumber=5;
26 • SELECT e.ename,d.dname FROM employee e LEFT OUTER JOIN department d ON e.dno=d.dnumber ORDER BY d.dname ASC,e.ename ASC;
27 • SELECT p.pnumber,p.pname,count(e.ssn) FROM Work_in w LEFT OUTER JOIN Proj p ON w.pno=p.pnumber LEFT OUTER JOIN employee e ON w.essn=e.ssn GROUP BY p.pname,p.pnumber;
28 • create VIEW emp_dept_view AS select * from employee NATURAL JOIN department;
29 ☐ CREATE ANY SYNONYM emp_dept FOR employee_dept_view;
30 • select * from emp_dept;
31 • Select * From employee e,department d WHERE e.dno=d.dnumber;
32 • SELECT * FROM employee e,department d WHERE NOT(e.dno=d.dnumber);
33 • SELECT * FROM employee e LEFT OUTER JOIN department d ON e.dno=d.dnumber;
34 • SELECT * FROM employee e RIGHT OUTER JOIN department d ON e.dno=d.dnumber;

```

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

ssn	ename	design	dno	doj	salary	dnumber	dname	loc	mgrssn
2	Bhama	Admin	1	2008-03-10	75000	1	Admin	Chennai	2
1	Abhi	HR	2	2009-07-12	70000	2	HR	Chennai	7
3	Chriz	Sales	3	2011-06-23	35000	3	Sales	Kochi	3
8	Hima	Finance	4	2002-04-01	45000	4	Finance	Delhi	6
6	Hima	Finance	4	2002-04-01	51000	4	Finance	Delhi	6
7	Hima	HR	5	2002-04-01	49000	5	Production	Thiruvananthapuram	5
5	Govind	Production	5	1995-06-21	35000	5	Production	Thiruvananthapuram	5
4	Diya	Production	5	2018-09-29	32000	5	Production	Thiruvananthapuram	5

(e) Perform inner join on the emp and dept tables.

### Output

```

26 • SELECT e.ename,d.dname FROM employee e LEFT OUTER JOIN department d on e.dno=d.dnumber ORDER BY d.dname ASC,e.ename ASC;
27 • SELECT p.pnumber,p.pname,count(e.ssn) FROM Work_in w LEFT OUTER JOIN Proj p on w.pno=p.pnumber LEFT OUTER JOIN employee e on w.essn=e.ssn GROUP BY
28 • create VIEW emp_dept_view as select * from employee NATURAL JOIN department;
29 ✘ CREATE ANY SYNONYM emp_dept for employee_dept_view;
30 • select * from emp_dept;
31 • Select * From employee e,department d WHERE e.dno=d.dnumber;
32 • SELECT * FROM employee e,department d WHERE NOT(e.dno=d.dnumber);
33 • SELECT * FROM employee e LEFT OUTER JOIN department d ON e.dno=d.dnumber;
34 • SELECT * FROM employee e RIGHT OUTER JOIN department d ON e.dno=d.dnumber;
35 • SELECT * FROM employee e INNER JOIN department d ON e.dno=d.dnumber;

```

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

ssn	ename	design	dno	doj	salary	dnumber	dname	loc	mgrssn
1	Abhi	HR	2	2009-07-12	70000	2	HR	Chennai	7
2	Bhama	Admin	1	2008-03-10	75000	1	Admin	Chennai	2
3	Chriz	Sales	3	2011-06-23	35000	3	Sales	Kochi	3
4	Diya	Production	5	2018-09-29	32000	5	Production	Thiruvananthapuram	5
5	Govind	Production	5	1995-06-21	35000	5	Production	Thiruvananthapuram	5
6	Hima	Finance	4	2002-04-01	51000	4	Finance	Delhi	6
7	Hima	HR	5	2002-04-01	49000	5	Production	Thiruvananthapuram	5
8	Hima	Finance	4	2002-04-01	45000	4	Finance	Delhi	6

# CYCLE 4

## QUESTION SET 6

Consider the database for a banking enterprise. Write the queries for the below questions.

- (i) Create the following tables

Table	Attributes
customer	cid, cname, loc, sex, dob
Bank_brn	bcode, bloc, bstate
Deposit	Dacno, dtype, ddate, damt
Loan	Lacno, ltype, ldate, lamt
Accounts_in	Bcode, cid
depositor	cid, dacno
borrower	cid, lacno

### Output

The screenshot shows the MySQL Workbench interface with the SQL editor tab active. The code area contains the following SQL statements:

```
1 • CREATE TABLE customer(cid INT PRIMARY KEY,cname VARCHAR(25) NOT NULL,loc VARCHAR(25),sex VARCHAR(25),dob DATE );
2 • CREATE TABLE bank_brn
3     (bcode INT PRIMARY KEY,bloc VARCHAR(25) NOT NULL,bstate VARCHAR(25) NOT NULL);
4 • CREATE TABLE deposit
5     (Dacno INT PRIMARY KEY,ddate DATE NOT NULL,damt INT NOT NULL, dtype VARCHAR(25) NOT NULL);
6 • CREATE TABLE loan
7     (Lacno INT PRIMARY KEY,ldate DATE NOT NULL,lamt INT NOT NULL, ltype VARCHAR(25) NOT NULL);
8 • CREATE TABLE accounts_in
9     (cid INT NOT NULL, Bcode INT NOT NULL, FOREIGN KEY(Bcode) REFERENCES bank_brn(bcode), FOREIGN KEY(cid) REFERENCES customer(cid));
10
11 • CREATE TABLE depositor
12     (cid INT NOT NULL, dacno INT NOT NULL, FOREIGN KEY(dacno) REFERENCES deposit(Dacno), FOREIGN KEY(cid) REFERENCES customer(cid));
13 • CREATE TABLE borrower
14     (cid INT NOT NULL, lacno INT NOT NULL, FOREIGN KEY(lacno) REFERENCES loan(Lacno), FOREIGN KEY(cid) REFERENCES customer(cid));
```

The output pane shows the results of the execution:

#	Time	Action	Message	Duration / Fetch
4	15:04:33	CREATE TABLE deposit (Dacno INT PRIMARY KEY,ddate DATE NOT NULL,damt INT N...)	0 row(s) affected	0.422 sec
5	15:04:41	CREATE TABLE loan (Lacno INT PRIMARY KEY,ldate DATE NOT NULL,lamt INT NOT N...)	0 row(s) affected	0.516 sec
6	15:04:48	CREATE TABLE accounts_in (cid INT NOT NULL, Bcode INT NOT NULL, FOREIGN KEY(...)	0 row(s) affected	1.859 sec
7	15:04:58	CREATE TABLE depositor (cid INT NOT NULL, dacno INT NOT NULL, FOREIGN KEY(dac...)	0 row(s) affected	0.718 sec
8	15:05:03	CREATE TABLE borrower (cid INT NOT NULL, lacno INT NOT NULL, FOREIGN KEY(lacn...)	0 row(s) affected	1.594 sec

A message at the bottom right says: "Activate Windows Go to Settings to activate Windows."

- (ii) Include necessary constraints.

## Output

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a text area displaying SQL code. The code includes several CREATE TABLE statements for tables like loan, accounts\_in, depositor, borrower, and customer. A DESCRIBE command is also present. The number 15 is highlighted in blue, indicating the current query being run. Below the code is a result grid showing the structure of the 'customer' table.

```
6 • CREATE TABLE loan
7     (Lacno INT PRIMARY KEY,ldate DATE NOT NULL,amt INT NOT NULL, ltype VARCHAR(25) NOT NULL);
8 • CREATE TABLE accounts_in
9     (cid INT NOT NULL, Bcode INT NOT NULL, FOREIGN KEY(Bcode) REFERENCES bank_brn(bcode), FOREIGN KEY
10
11 • CREATE TABLE depositor
12     (cid INT NOT NULL, dacno INT NOT NULL, FOREIGN KEY(dacno) REFERENCES deposit(Dacno), FOREIGN KEY(
13 • CREATE TABLE borrower
14     (cid INT NOT NULL, lacno INT NOT NULL, FOREIGN KEY(lacno) REFERENCES loan(Lacno), FOREIGN KEY(cid
15 • DESCRIBE customer;
16
17
```

Field	Type	Null	Key	Default	Extra
cid	int	NO	PRI	NULL	
cname	varchar(25)	NO		NULL	
loc	varchar(25)	YES		NULL	
sex	varchar(25)	YES		NULL	
dob	date	YES		NULL	

(iii)Tables are created under the database 'bank'

## Output:

```

4 • CREATE TABLE deposit
5     (Dacno  INT PRIMARY KEY,ddate DATE NOT NULL,damt  INT NOT NULL, dtype VARCHAR(25) NOT NULL);
6 • CREATE TABLE loan
7     (Lacno  INT PRIMARY KEY,ldate DATE NOT NULL,amt  INT NOT NULL, ltype VARCHAR(25) NOT NULL);
8 • CREATE TABLE accounts_in
9     (cid  INT NOT NULL, Bcode INT NOT NULL, FOREIGN KEY(Bcode) REFERENCES bank_branch(bcode), FOREIGN KEY(cid) RE
10
11 • CREATE TABLE depositor
12     (cid  INT NOT NULL, dacno INT NOT NULL, FOREIGN KEY(dacno) REFERENCES deposit(Dacno), FOREIGN KEY(cid) RE
13 • CREATE TABLE borrower
14     (cid  INT NOT NULL, lacno INT NOT NULL, FOREIGN KEY(lacno) REFERENCES loan(Lacno), FOREIGN KEY(cid) REFER
15 • DESCRIBE borrower;

```

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

	Field	Type	Null	Key	Default	Extra
▶	cid	int	NO	MUL	NULL	
	lacno	int	NO	MUL	NULL	

(iv) Display all the tables in bank database

### Output;

```

5     (Dacno  INT PRIMARY KEY,ddate DATE NOT NULL,damt  INT NOT NULL, dtype VARCHAR(25) NOT NULL);
6 • CREATE TABLE loan
7     (Lacno  INT PRIMARY KEY,ldate DATE NOT NULL,amt  INT NOT NULL, ltype VARCHAR(25) NOT NULL);
8 • CREATE TABLE accounts_in
9     (cid  INT NOT NULL, Bcode INT NOT NULL, FOREIGN KEY(Bcode) REFERENCES bank_branch(bcode), FOREIGN KEY(cid) RE
10
11 • CREATE TABLE depositor
12     (cid  INT NOT NULL, dacno INT NOT NULL, FOREIGN KEY(dacno) REFERENCES deposit(Dacno), FOREIGN KEY(cid) RE
13 • CREATE TABLE borrower
14     (cid  INT NOT NULL, lacno INT NOT NULL, FOREIGN KEY(lacno) REFERENCES loan(Lacno), FOREIGN KEY(cid) REFERENC
15 • DESCRIBE bank.borrower;
16 • show tables;
17

```

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

Tables_in_bank
accounts_in
bank_branch
borrower
customer
deposit
depositor
loan

(v) Describe the structure of all tables

**Output:**

```
5      (Dacno  INT PRIMARY KEY,ddate DATE NOT NULL,damt   INT NOT NULL, dtype V
6 •  CREATE TABLE loan
7      (Lacno  INT PRIMARY KEY,ldate DATE NOT NULL,lamt   INT NOT NULL, ltype V
8 •  CREATE TABLE accounts_in
9      (cid    INT NOT NULL, Bcode INT NOT NULL, FOREIGN KEY(Bcode) REFERENCES ba
10
11 •  CREATE TABLE depositor
12      (cid    INT NOT NULL, dacno INT NOT NULL, FOREIGN KEY(dacno) REFERENCES de
13 •  CREATE TABLE borrower
14      (cid    INT NOT NULL, lacno INT NOT NULL, FOREIGN KEY(lacno) REFERENCES lo
15 •  DESCRIBE bank.borrower;
16 •  show tables;
17 •  DESCRIBE loan;
```

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

	Field	Type	Null	Key	Default	Extra
▶	Lacno	int	NO	PRI	NULL	
	ldate	date	NO		NULL	
	lamt	int	NO		NULL	
	ltype	varchar(25)	NO		NULL	

```
6 • CREATE TABLE loan
7     (lacno INT PRIMARY KEY, ldate DATE NOT NULL, lamt INT NOT NULL)
8 • CREATE TABLE accounts_in
9     (cid INT NOT NULL, Bcode INT NOT NULL, FOREIGN KEY(Bcode) REFERENCES
10
11 • CREATE TABLE depositor
12     (cid INT NOT NULL, dacno INT NOT NULL, FOREIGN KEY(dacno) REFERENCES
13 • CREATE TABLE borrower
14     (cid INT NOT NULL, lacno INT NOT NULL, FOREIGN KEY(lacno) REFERENCES
15 • DESCRIBE bank.borrower;
16 • show tables;
17 • DESCRIBE loan;
18 • DESCRIBE accounts_in;
```

Result Grid					
Field	Type	Null	Key	Default	Extra
cid	int	NO	MUL	NULL	
Bcode	int	NO	MUL	NULL	

(vi) Delete tables.

**Output;**

```

2 • CREATE TABLE bank_brn
3     (bcode  INT PRIMARY KEY,bloc VARCHAR(25) NOT NULL,bstate VARCHAR(25) NOT NULL);
4 • CREATE TABLE deposit
5     (Dacno  INT PRIMARY KEY,ddate DATE NOT NULL,damt  INT NOT NULL, dtype VARCHAR(25) NOT NULL);
6 • CREATE TABLE loan
7     (Lacno  INT PRIMARY KEY,ldate DATE NOT NULL,lamt  INT NOT NULL, ltype VARCHAR(25) NOT NULL);
8 • CREATE TABLE accounts_in
9     (cid  INT NOT NULL, Bcode INT NOT NULL, FOREIGN KEY(Bcode) REFERENCES bank_brn(bcode), FOREIGN KEY(cid)
10
11 • CREATE TABLE depositor
12     (cid  INT NOT NULL, dacno INT NOT NULL, FOREIGN KEY(dacno) REFERENCES deposit(Dacno), FOREIGN KEY(cid)
13 • CREATE TABLE borrower
14     (cid  INT NOT NULL, lacno INT NOT NULL, FOREIGN KEY(lacno) REFERENCES loan(Lacno), FOREIGN KEY(cid) RE
15 • DESCRIBE bank.borrower;
16 • show tables;
17 • DESCRIBE loan;
18 • DESCRIBE accounts_in;
19 • drop table borrower,depositor,accounts_in,loan,customer;

```

<

Output

Action Output			Message
#	Time	Action	
✓	24 15:51:08	DESCRIBE loan	4 row(s) returned
✓	25 15:51:52	DESCRIBE accounts_in	2 row(s) returned
✗	26 15:54:03	drop table borrower,depositor,account_in,loan,customer	Error Code: 1051. Unknown table 'bank.account_in'
✓	27 15:54:29	drop table borrower,depositor,accounts_in,loan,customer	0 row(s) affected

## Q.SET 8

Consider the following database for a banking enterprise.

- BRANCH (bid:int, branch-name: String, branch-city: String, assets: int)
- ACCOUNTS (accno: int, bid:int, balance: int)
- DEPOSITOR (cid:int, accno: int)
- CUSTOMER(cid:int, customer-name:String, customer-street:String, customer-city: String)

Set primary key and foreign keys and insert valid records based on questions.

Write SQL queries to

1. Find all the customers who have at least two accounts at the Mainbranch.

#### Output

cycle 2 question 3 cycle 3 question 1 cycle 4 question 6 cycle 4 question 8\* SET4.1

```
80 (22,'CUST2','S2','CITY2'),
81 (33,'CUST3','S3','CITY3'),
82 (44,'CUST4','S4','CITY4'),
83 (55,'CUST5','S4','CITY5');
84 • SELECT * FROM CUSTOMER;
85
86 • SELECT CUSTOMER.CID,CUSTOMER_NAME,BRANCH_NAME FROM BRANCH,CUSTOMER,DEPOSITOR,ACCOUNTS WHERE CUSTOMER.CID=DEPOSITOR.CID AND DEPOSITOR.ACCTNO=ACCOUNTS.ACCTNO
87 GROUP BY ACCOUNTS.BID HAVING COUNT(ACCOUNTS.BID)>=2;
88
```

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

CID	CUSTOMER_NAME	BRANCH_NAME
22	CUST2	MAIN

2. Find all the customers who have an account at all the branches located in a specific city.

#### Output

cycle 2 question 3 cycle 3 question 1 cycle 4 question 6 cycle 4 question 8\* SET4.1

```
86 • SELECT CUSTOMER.CID,CUSTOMER_NAME,BRANCH_NAME FROM BRANCH,CUSTOMER,DEPOSITOR,ACCOUNTS WHERE CUSTOMER.CID=DEPOSITOR.CID AND DEPOSITOR.ACCTNO=ACCOUNTS.ACCTNO
87 GROUP BY ACCOUNTS.BID HAVING COUNT(ACCOUNTS.BID)>=2;
88
89
90 • SELECT CUSTOMER.CID,CUSTOMER_NAME,ACCOUNTS.ACCTNO,BRANCH_CITY FROM ACCOUNTS,BRANCH,DEPOSITOR,CUSTOMER WHERE BRANCH.BID=ACCOUNTS.BID AND ACCOUNTS.ACCTNO=DEPOSITOR.ACCTNO AND BRANCH.BRANCH_CITY='C3'
91
92 GROUP BY DEPOSITOR.CID HAVING COUNT(DISTINCT BRANCH.BID)=(SELECT COUNT(BID) FROM BRANCH WHERE BRANCH_CITY='C3');
93
94
```

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

CID	CUSTOMER_NAME	ACCTNO	BRANCH_CITY
11	CUST1	107	C3
44	CUST4	106	C3

3. Find the branch with greatest asset.

#### Output

cycle 2 question 3   cycle 3 question 1   cycle 4 question 6   **cycle 4 question 8\***   SE14.1

```

87     GROUP BY ACCOUNTS.BID HAVING COUNT(ACCOUNTS.BID)>=2;
88
89
90 •   SELECT CUSTOMER.CID,CUSTOMER_NAME,ACCOUNTS.ACCTNO,BRANCH_CITY FROM ACCOUNTS,BRANCH,DEPOSITOR,CUSTOMER WHERE BRANCH.BID=ACCOUNTS.BID AND BRANCH.BRANCH_CITY='C3'
91
92     GROUP BY DEPOSITOR.CID HAVING COUNT(DISTINCT BRANCH.BID)=(SELECT COUNT(BID) FROM BRANCH WHERE BRANCH_CITY='C3');
93
94
95 •   SELECT * FROM BRANCH WHERE ASSETS=(SELECT MAX(ASSETS) FROM BRANCH);

```

**Result Grid** | Filter Rows:  | Edit: | Export/Import: | Wrap Cell Content:

	BID	BRANCH_NAME	BRANCH_CITY	ASSETS
▶	5	B5	C5	50000
*	NULL	NULL	NULL	NULL

4. Find the customer with highest balance.

## Output

cycle 2 question 3   cycle 3 question 1   cycle 4 question 5   **cycle 4 question 8\***   SE14.1

```

90 •   SELECT CUSTOMER.CID,CUSTOMER_NAME,ACCOUNTS.ACCTNO,BRANCH_CITY FROM ACCOUNTS,BRANCH,DEPOSITOR,CUSTOMER WHERE BRANCH.BID=ACCOUNTS.BID AND AC
91     AND BRANCH.BRANCH_CITY='C3'
92     GROUP BY DEPOSITOR.CID HAVING COUNT(DISTINCT BRANCH.BID)=(SELECT COUNT(BID) FROM BRANCH WHERE BRANCH_CITY='C3');
93
94
95 •   SELECT * FROM BRANCH WHERE ASSETS=(SELECT MAX(ASSETS) FROM BRANCH);
96
97
98 •   SELECT CUSTOMER.CID,CUSTOMER_NAME,ACCOUNTS.ACCTNO,BALANCE FROM CUSTOMER,DEPOSITOR,ACCOUNTS

```

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

	CID	CUSTOMER_NAME	ACCTNO	BALANCE
▶	55	CUST5	111	10000