DBMS: LAB 7

Name: Shriya Konduru

SRN: PES2UG22CS546

JOINS AND AGGREGATE FUNCTIONS

University Fest Management System

INSTRUCTIONS

- In Lab 7 you are expected to solve 2 tasks that are to be completed and submitted.
- For this Lab, you would have to continue with the University Fest DB given to you last week for tasks 2.
- As a part of LAB 3, there are 2 tasks that are to be completed as described below:
    - **TASK 1:** A small case study on a random database has been given. You would have to understand or look through the entries in the tables and answer if the commands given are executable (then output) or not (then reason), along with the reasons. (NO NEED FOR ANY EXECUTION)
    - **Task 2:** This is the last task. Here, there will be 3 questions and you are expected to understand the questions and write the queries to solve them. Support each question with the corresponding screenshot.
- Ensure that your mysql **command client prompt is modified as per your SRN** using the command:

    prompt YOUR_SRN>

- An example of how your command line prompt should look is given in the "EXAMPLE" section. Note that this step is **mandatory**.
- The screenshots that are to be taken for each task are specified in detail below "EXAMPLE".
- As a part of the submission process, the following are to be submitted:
    - A **PDF** document, containing all the Screenshots for all 3 tasks as suggested
        - Name of the file: `<your SRN>_University_Fest_DB_Lab3.pdf`

Example:

Refer to the sample submissions given below. This will give you an idea about the details that must be included in your submissions

NOTE: Screenshots must be taken from "**Command Line".**

Changing your command line prompt:

Before :

```
mysql> _
```

prompt PES1UG20CS183>

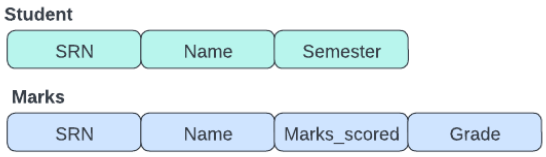
After:

```
mysql> prompt PES1UG20CS183>
PROMPT set to 'PES1UG20CS183> '
PES1UG20CS183> _
```

TASK 1:

CASE STUDY 1

Consider the following database as shown below: (Question 1&2)



- The "Student" table stores information about students like SRN, name, and semester.
- The "Marks" table contains information about which student gets what marks and grades for a given subject name.
- Note that there are no primary keys and foreign key constraints placed in the above case study

For the next two questions (questions 1 & 2) answer based on the above case study.

Question 1

Considering the entries in the "student" and "marks" tables, can the following query be executed successfully? If yes what is the output? If no what would be the error?

Student table:

SRN	Name	Semester
S001	Math	5
S002	Julie	3
S003	Martin	4
NULL	Science	8
S004	NULL	3
S010	Art	2
S011	Martha	5
NULL	NULL	6

Marks table:

SRN	Name	marks_scored	Grade
S001	Math	95	S
S001	Science	100	S
S001	Art	80	A
S002	Math	89	A
S002	Science	100	S
S005	Math	88	A
S005	Art	100	S
S010	NULL	80	A
NULL	NULL	100	S
NULL	Art	100	S
S012	Math	98	S

- *SELECT * FROM Student NATURAL JOIN Marks;*
- **SELECT e.event_id, e.event_name FROM event e INNER JOIN Registration r ON e.event_id = r.event_id INNER JOIN Participant p ON r.SRN = p.SRN GROUP BY e.event_id, e.event_name HAVING SUM(CASE WHEN p.gender = 'Female' THEN 1 ELSE 0 END) > SUM (CASE WHEN p.gender = 'Male' THEN 1 ELSE 0 END);**

Question 2:

Considering the entries in the "student" and "marks" tables as shown above. A student was asked to make a full outer join on the above two tables. The query written by the student and the output is given to you below:

Query :

- ***(SELECT * FROM student NATURAL LEFT OUTER JOIN marks)***

  UNION

  (SELECT * FROM student NATURAL RIGHT OUTER JOIN marks);

Output :

SRN	Name	Semester	marks_scored	Grade
S001	Math	5	95	S
S002	Julie	3	NULL	NULL
S003	Martin	4	NULL	NULL
NULL	Science	8	NULL	NULL
S004	NULL	3	NULL	NULL
S010	Art	2	NULL	NULL
S011	Martha	5	NULL	NULL
NULL	NULL	6	NULL	NULL
S001	Math	95	S	5
S001	Science	100	S	NULL
S001	Art	80	A	NULL
S002	Math	89	A	NULL
S002	Science	100	S	NULL
S005	Math	88	A	NULL
S005	Art	100	S	NULL
S010	NULL	80	A	NULL
NULL	NULL	100	S	NULL
NULL	Art	100	S	NULL
S012	Math	98	S	NULL

Examining the above-given query and the output screenshot, is the task assigned to the student satisfied? If not what could be the possible reason for the same? And also suggest what would be the correct way that the student should have taken to do it.

CASE STUDY 2

Consider the following database as shown below: (Questions 3&4)

- The "Demo" table stores random information that includes ID, name, and grade.
- The description of the demo table is as shown:

Field	Type	Null	Key	Default	Extra
ID	int	YES		NULL	
NAME	varchar(10)	YES		NULL	
GRADE	enum('S','A','B')	YES		NULL	

- Note that there are no primary keys and foreign key constraints placed in the above case study

For the next two questions (questions 3 & 4) answer based on the above case study. The screenshots of input values for the demo table are as shown.

ID	NAME	GRADE
1	1abc	A
2	Cde	S
3	567	B
4	cde	S

Question 3:

Considering the above case study, what would be the output for the given below commands?

- ***SELECT * FROM Demo ORDER BY Name DESC;***
- ***SELECT * FROM Demo ORDER BY Name;***

Question 4:

A student was asked to order the entries of the demo table given above by the Grade column. The query written by the student and the output is given to you below:

Query:

- ***SELECT * FROM Demo ORDER BY Grade;***
-

Output:

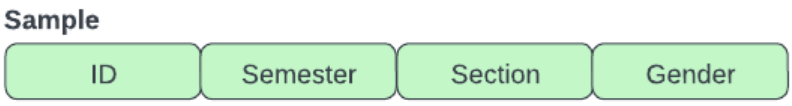
	ID	NAME	GRADE
▶	2	Cde	S
	4	cde	S
	1	1abc	A
	3	567	B

The result, however, was different from what the student expected. Since A occurs before B and B occurs before S in alphabetical order, the student assumed that all records with grade 'A' would be listed first, followed by all records with grade 'B' and finally by all records with grade 'S'. Can you explain why the result is different from the student's expectation?

Answer: It does not order grade column alphabetically because the it has a datatype of enum. Therefore it will arrange based on the order in which the grades are arranged in the enum. i.e S grade is defined first.

CASE STUDY 3:

Consider the following database as shown below: (Question 5)



- The "Sample" table stores random information that includes ID, Semester, Section, Gender.

- The description of the demo table is as shown:

	Field	Type	Null	Key	Default	Extra
▶	ID	int	YES		NULL	
	Semester	enum('1','2','3')	YES		NULL	
	Section	char(1)	YES		NULL	
	Gender	enum('M','F','O')	YES		NULL	

- Note that there are no primary keys and foreign key constraints placed in the above case study

The values in the Sample table are as shown below:

	ID	Semester	Section	Gender
▶	1	1	A	M
	2	1	B	F
	4	1	A	F
	6	1	A	F
	9	1	B	M
	10	1	B	M
	12	1	B	M
	100	NULL	NULL	NULL
	NULL	NULL	NULL	NULL
	NULL	NULL	NULL	F
	NULL	NULL	A	NULL
	NULL	3	NULL	NULL
	NULL	3	B	M
	NULL	NULL	B	M

Two students (Student A and Student B) were given the task of finding out the number of students in each section of the semester. The two students had come up with the following queries:

Student A:

```
SELECT semester, section, COUNT(gender) AS section_total
FROM Sample_group
GROUP BY semester, section
ORDER BY semester, section;
```

Student B:

```
SELECT semester, section, COUNT(*) AS section_total
FROM Sample_group
GROUP BY semester, section
ORDER BY semester, section;
```

You have to examine how many of the above queries are correct and able to achieve the given task. If any query is incorrect what would be the reason for the same?

NOTE: Both queries could be incorrect.

Query will be executed but the solution is not what we expect. This query considers only common columns, so here there is a mismatch.

Correct Query : (select student_srn, student.name as student_name,semester, marks.name as ssubject_name, marks_scored, grade FROM student LEFT JOIN marks ON student.SRN=marks.SRN) UNION (select marks.SRN, student.name as student_name,semester, marks.name AS subject_name, marks_scored, grade FROM student student RIGHT JOIN marks ON Student.SRN=marks.SRN);

Task 2:

Understand the various scenarios described in the given questions and write the corresponding SQL queries for the same. Support your answers with the help of the screenshot.

Question 1

There are many events that are hosted in the university. You are an analyst who has been given the task of identifying which events have more female participants than male participants.

NOTE:

- The output should have the event_id and the event_name only.

//Add screen shot of query and output

Question 2:

Every fest has a number of stalls associated with it. Each stall could offer items that are only VEG, only NON-VEG, or both. As a stall management head, you are expected to find out the total number of veg items, and non-veg items sold in each stall present in the database.

NOTE:

- If there are no items of a particular category, do not display that category.
- Order the output by the name of the items in descending order, and then type.
- In ordering the items always VEG should come before NON-VEG if the stall sells both items
- The output should contain name of the stall, type of item, and the count value corresponding called as Item_count

SELECT s.name, type, COUNT(*) AS item_count FROM stall s JOIN stall_items SI on s.stall_id = SI.stall_id JOIN item I on si.item_name = I.name GROUP BY s.name, i.type ORDER BY s.name DESC, i.type;

SELECT P.name, GROUP_CONCAT(v.name SEPARATOR, ‘,’) AS Visitor_list COUNT (v.name) AS visitor_count FROM participant P LEFT OUTER JOIN visitor V ON p.SRN = v.srn GROUP BY p.srn ORDER BY COUNT (v.name) desc;

Question 3 for students to solve:

Every participant can have zero or more number of visitors. As an analyst, you are given a task to find out the participants along with the list of all the visitors and the corresponding visitors count.

NOTE:

- If there is a participant P with visitors V1, V2 then the table entry corresponding to it would be as shown:

Participant	Visitor_list	Visitor_count
P	V1, V2	2

- If there is a participant P1 with no visitors then the table entry corresponding to it would be as shown:

Participant	Visitor_list	Visitor_count
P1	NULL	0

Expected OUTPUT:

name	Visitor_list	Visitor_count
John Smith	David Wilson, John Doe, Michael Johnson	3
Michael Williams	Andrew Thomas, Daniel Taylor	2
Mia Davis	Emily Davis, Sophia Brown	2
Charlotte Baker	Isabella Martinez, Olivia Anderson	2
Emily Johnson	Jane Smith	1
Sophia Brown	NULL	0
Jacob Jones	NULL	0
Olivia Davis	NULL	0
Ethan Miller	NULL	0
Ava Wilson	NULL	0
William Taylor	NULL	0
Emma Anderson	NULL	0
Liam Martinez	NULL	0
Isabella Thompson	NULL	0
James Harris	NULL	0
Benjamin Clark	NULL	0
Daniel Lopez	NULL	0
Amelia Turner	NULL	0
Henry Hill	NULL	0
Victoria Young	NULL	0
David Lee	NULL	0
Sofia Green	NULL	0
Christopher Moore	NULL	0
Scarlett Evans	NULL	0
Andrew Martinez	NULL	0

25 rows in set (0.03 sec)