

DBMS: LAB 7

JOINS AND AGGREGATE FUNCTIONS

University Fest Management System

ABUTHWAHIR H M,
PES1UG22CS022.

TASK 1:

CASE STUDY 1:

Question 1:

Yes, The query runs successfully.

Question 2:

NO , The code doesn't run successfully.

The right approach would be to use an **explicit Full Outer Join** for the SRN column.

CASE STUDY 2:

Question 3:

For command:

| ID | NAME | GRADE | |
|----|--------|-------|--|
| 4 | cde | s | |
| 2 | RAJU | s | |
| 1 | MADARA | a | |
| 3 | 576 | b | |

| Id | Name | Grade |
|----|--------|-------|
| 1 | MADARA | a |
| 2 | RAJU | s |
| 3 | Cde | S |
| 4 | 567 | d |

SELECT * FROM Demo ORDER BY Name

So basically command 1 , Command sorts the rows by the Name column in descending order.

Command 2, This command sorts the rows by the Name column in ascending order .which is the default

Question 4:

The student's expectation is wrong because the Grade column is probably set as an ENUM type. This means the grades are stored as numbers based on the order in which they were defined, not by alphabetical order.

So , While sorting, it follows the internal number order, not the letter order. To fix this, the student would need to change how the sorting is done to make it alphabetical

CASE STUDY 3:

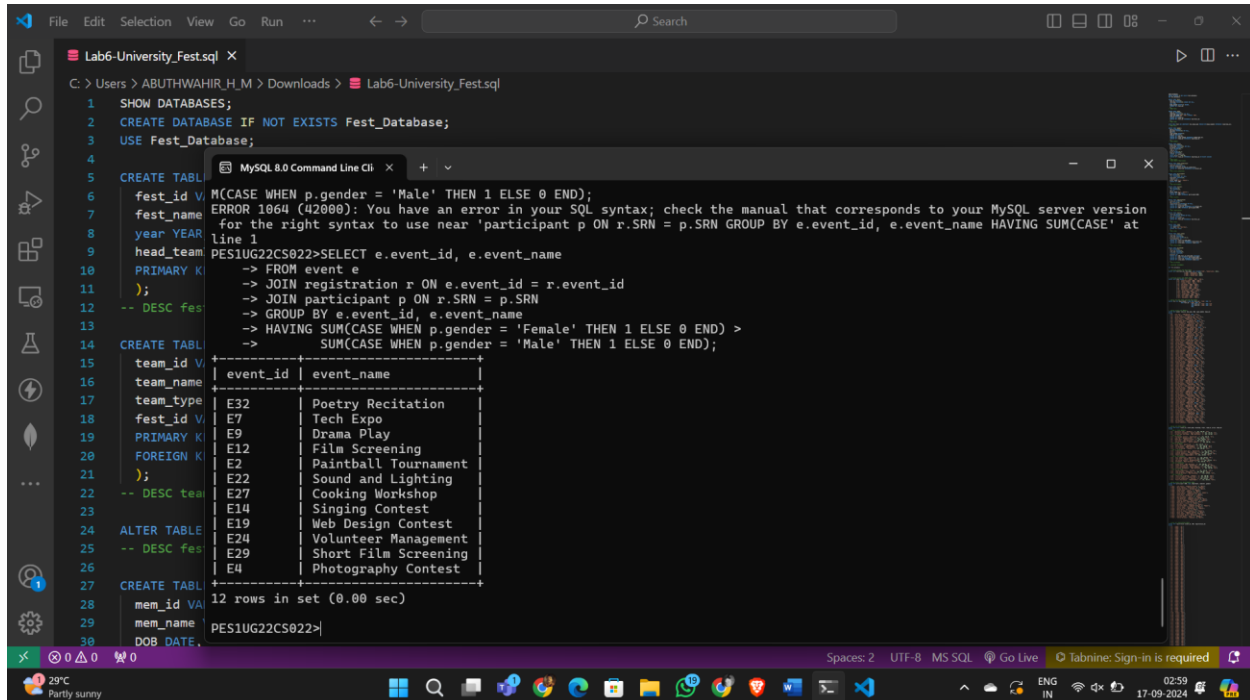
Student A's query counts based on the gender column, which is fine if the gender column doesn't have any missing values.

Student B's query counts all rows using count(), which is accurate even if some rows have missing values in the gender column.*

Both Student A and Student B wrote correct queries to find out how many students are in each section for each semester...

Task 2:

Question1:



```

1 SHOW DATABASES;
2 CREATE DATABASE IF NOT EXISTS Fest_Database;
3 USE Fest_Database;
4
5 CREATE TABLE
6   fest_id VARCHAR(10) PRIMARY KEY,
7   fest_name VARCHAR(100),
8   year YEAR,
9   head_team VARCHAR(100),
10  PRIMARY KEY (fest_id, year),
11  -- DESC Fest;
12
13 CREATE TABLE
14   team_id VARCHAR(10) PRIMARY KEY,
15   team_name VARCHAR(100),
16   team_type VARCHAR(100),
17   fest_id VARCHAR(10),
18   PRIMARY KEY (team_id, fest_id),
19   FOREIGN KEY (fest_id) REFERENCES fest_id (fest_id);
20
21 -- DESC team;
22
23 ALTER TABLE
24   team_id
25   ADD COLUMN (fest_id VARCHAR(10));
26
27 CREATE TABLE
28   mem_id VARCHAR(10) PRIMARY KEY,
29   mem_name VARCHAR(100),
30   DOB DATE,

```

MySQL 8.0 Command Line Cli

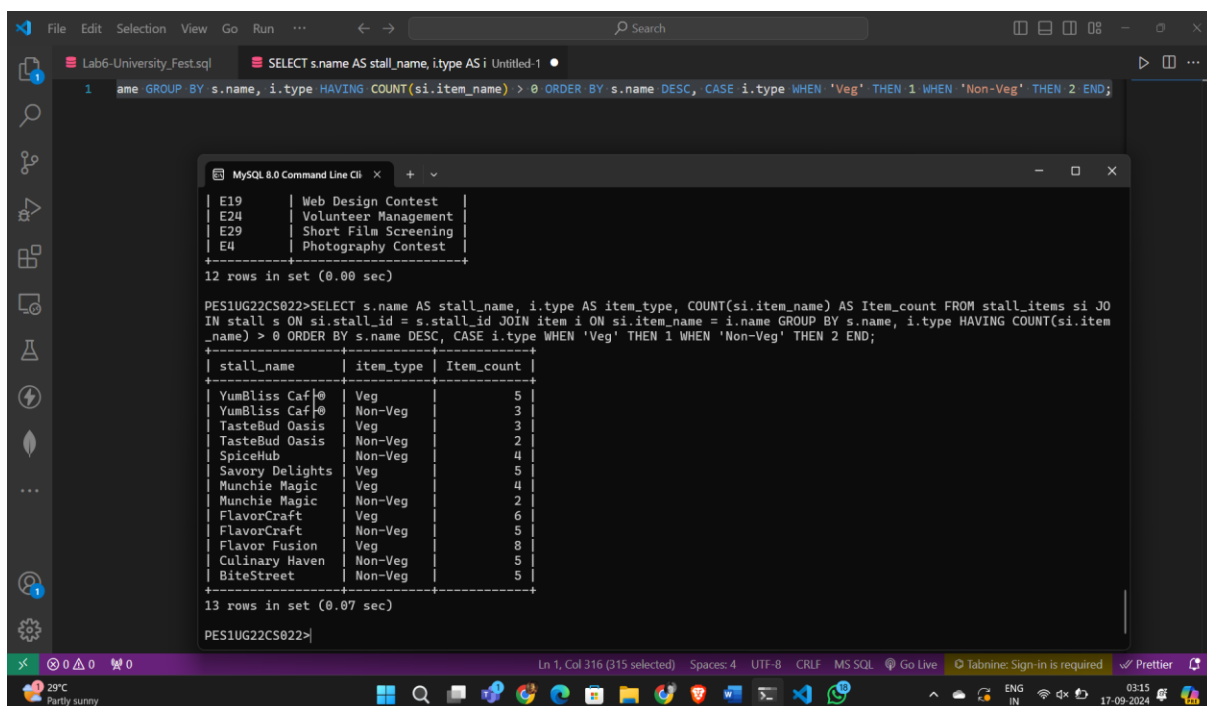
```

M(CASE WHEN p.gender = 'Male' THEN 1 ELSE 0 END);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
for the right syntax to use near 'participant p ON r.SRN = p.SRN GROUP BY e.event_id, e.event_name HAVING SUM(CASE' at
line 1
PES1UG22CS022>SELECT e.event_id, e.event_name
-> FROM event e
-> JOIN registration r ON e.event_id = r.event_id
-> 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);
12 rows in set (0.00 sec)
PES1UG22CS022>

```

| event_id | event_name |
|----------|----------------------|
| E32 | Poetry Recitation |
| E7 | Tech Expo |
| E9 | Drama Play |
| E12 | Film Screening |
| E2 | Paintball Tournament |
| E22 | Sound and Lighting |
| E27 | Cooking Workshop |
| E14 | Singing Contest |
| E19 | Web Design Contest |
| E24 | Volunteer Management |
| E29 | Short Film Screening |
| E4 | Photography Contest |

Question 2:



```

1 SELECT s.name AS stall_name, i.type AS item_type, COUNT(si.item_name) AS Item_count FROM stall_items si JOIN
2   stall s ON si.stall_id = s.stall_id JOIN item i ON si.item_name = i.name GROUP BY s.name, i.type HAVING COUNT(si.item
3   name) > 0 ORDER BY s.name DESC, CASE i.type WHEN 'Veg' THEN 1 WHEN 'Non-Veg' THEN 2 END;

```

MySQL 8.0 Command Line Cli

```

PES1UG22CS022>SELECT s.name AS stall_name, i.type AS item_type, COUNT(si.item_name) AS Item_count FROM stall_items si JOIN
IN stall s ON si.stall_id = s.stall_id JOIN item i ON si.item_name = i.name GROUP BY s.name, i.type HAVING COUNT(si.item
_name) > 0 ORDER BY s.name DESC, CASE i.type WHEN 'Veg' THEN 1 WHEN 'Non-Veg' THEN 2 END;
13 rows in set (0.07 sec)
PES1UG22CS022>

```

| stall_name | item_type | Item_count |
|-----------------|-----------|------------|
| YumBliss Caffe | Veg | 5 |
| YumBliss Caffe | Non-Veg | 3 |
| TasteBud Oasis | Veg | 3 |
| TasteBud Oasis | Non-Veg | 2 |
| SpiceHub | Non-Veg | 4 |
| Savory Delights | Veg | 5 |
| Munchie Magic | Veg | 4 |
| Munchie Magic | Non-Veg | 2 |
| FlavorCraft | Veg | 6 |
| FlavorCraft | Non-Veg | 5 |
| Flavor Fusion | Veg | 8 |
| Culinary Haven | Non-Veg | 5 |
| BiteStreet | Non-Veg | 5 |

Database Management System

Question 3 for students to solve:

```
MySQL 8.0 Command Line Cli
PES1UG22CS022>SELECT p.name AS Participant,
-> GROUP_CONCAT(v.name ORDER BY v.name SEPARATOR ', ') AS Visitor_list,
-> IF(COUNT(v.name) > 0, COUNT(v.name), 0) AS Visitor_count
-> FROM participant p
-> LEFT JOIN visitor v ON p.SRN = v.SRN
-> GROUP BY p.name
-> ORDER BY Visitor_count DESC;
+-----+-----+-----+
| Participant | Visitor_list | Visitor_count |
+-----+-----+-----+
| John Smith | David Wilson, John Doe, Michael Johnson | 3 |
| Charlotte Baker | Isabella Martinez, Olivia Anderson | 2 |
| Mia Davis | Emily Davis, Sophia Brown | 2 |
| Michael Williams | Andrew Thomas, Daniel Taylor | 2 |
| Emily Johnson | Jane Smith | 1 |
| Amelia Turner | NULL | 0 |
| Andrew Martinez | NULL | 0 |
| Ava Wilson | NULL | 0 |
| Benjamin Clark | NULL | 0 |
| Christopher Moore | NULL | 0 |
| Daniel Lopez | NULL | 0 |
| David Lee | NULL | 0 |
| Emma Anderson | NULL | 0 |
| Ethan Miller | NULL | 0 |
| Henry Hill | NULL | 0 |
| Isabella Thompson | NULL | 0 |
| Jacob Jones | NULL | 0 |
| James Harris | NULL | 0 |
| Liam Martinez | NULL | 0 |
| Olivia Davis | NULL | 0 |
| Scarlett Evans | NULL | 0 |
| Sofia Green | NULL | 0 |
| Sophia Brown | NULL | 0 |
| Victoria Young | NULL | 0 |
| William Taylor | NULL | 0 |
+-----+-----+-----+
25 rows in set (0.00 sec)

PES1UG22CS022>
```