**MySQL Notes: Functions and Subqueries with Output**

**🔹 1. Table Schemas**

**🗂️ students**

CREATE TABLE students (

student\_id INT PRIMARY KEY,

name VARCHAR(50),

age INT,

marks INT,

city VARCHAR(30)

);

**🗂️ courses**

CREATE TABLE courses (

course\_id INT PRIMARY KEY,

course\_name VARCHAR(50),

credit\_hours INT

);

**🗂️ enrollments**

CREATE TABLE enrollments (

enroll\_id INT PRIMARY KEY,

student\_id INT,

course\_id INT,

grade CHAR(1),

FOREIGN KEY (student\_id) REFERENCES students(student\_id),

FOREIGN KEY (course\_id) REFERENCES courses(course\_id)

);

**🔹 2. Sample Data**

**👨‍🎓 students**

INSERT INTO students VALUES

(1, 'Ali', 18, 70, 'Lahore'),

(2, 'Sara', 19, 85, 'Karachi'),

(3, 'Ahmed', 20, 40, 'Lahore'),

(4, 'Zara', 21, 90, 'Islamabad'),

(5, 'Usman', 22, 50, 'Multan');

**📘 courses**

INSERT INTO courses VALUES

(101, 'Database Systems', 3),

(102, 'Computer Networks', 4),

(103, 'Data Structures', 3);

**📝 enrollments**

INSERT INTO enrollments VALUES

(1, 1, 101, 'B'),

(2, 2, 101, 'A'),

(3, 3, 102, 'C'),

(4, 2, 102, 'A'),

(5, 4, 103, 'A');

**🔹 3. MySQL Functions – With Output**

**A. String Functions**

SELECT UPPER(name), LOWER(city), LENGTH(name)

FROM students;

| **UPPER(name)** | **LOWER(city)** | **LENGTH(name)** |
| --- | --- | --- |
| ALI | lahore | 3 |
| SARA | karachi | 4 |
| AHMED | lahore | 5 |
| ZARA | islamabad | 4 |
| USMAN | multan | 5 |

**B. Numeric Functions**

SELECT ABS(-25), CEIL(4.3), FLOOR(4.8), ROUND(3.567, 2);

| **ABS(-25)** | **CEIL(4.3)** | **FLOOR(4.8)** | **ROUND(3.567, 2)** |
| --- | --- | --- | --- |
| 25 | 5 | 4 | 3.57 |

**C. Aggregate Functions**

SELECT COUNT(\*), MAX(marks), MIN(marks), AVG(marks)

FROM students;

| **COUNT(\*)** | **MAX(marks)** | **MIN(marks)** | **AVG(marks)** |
| --- | --- | --- | --- |
| 5 | 90 | 40 | 67 |

**D. Conditional Function (IF)**

SELECT name, marks, IF(marks >= 50, 'Pass', 'Fail') AS status

FROM students;

| **name** | **marks** | **status** |
| --- | --- | --- |
| Ali | 70 | Pass |
| Sara | 85 | Pass |
| Ahmed | 40 | Fail |
| Zara | 90 | Pass |
| Usman | 50 | Pass |

**🔹 4. Subqueries – With Output**

**A. WHERE Clause Subquery**

SELECT name, marks FROM students

WHERE marks > (SELECT AVG(marks) FROM students);

**Avg Marks = 67**

| **name** | **marks** |
| --- | --- |
| Sara | 85 |
| Zara | 90 |

**B. SELECT Clause Subquery**

SELECT name, marks,

(SELECT AVG(marks) FROM students) AS avg\_marks

FROM students;

| **name** | **marks** | **avg\_marks** |
| --- | --- | --- |
| Ali | 70 | 67 |
| Sara | 85 | 67 |
| Ahmed | 40 | 67 |
| Zara | 90 | 67 |
| Usman | 50 | 67 |

**C. FROM Clause Subquery**

SELECT name, marks FROM (

SELECT \* FROM students

) AS temp

WHERE marks > (SELECT AVG(marks) FROM students);

Same output as WHERE clause subquery:

| **name** | **marks** |
| --- | --- |
| Sara | 85 |
| Zara | 90 |

**D. Subquery with IN**

SELECT name FROM students

WHERE student\_id IN (

SELECT student\_id FROM enrollments

WHERE course\_id = (

SELECT course\_id FROM courses

WHERE course\_name = 'Database Systems'

)

);

| **name** |
| --- |
| Ali |
| Sara |

**🧠 Practice Tasks**

1. **Display students with total enrolled courses (using GROUP BY):**

SELECT s.name, COUNT(e.course\_id) AS total\_courses

FROM students s

JOIN enrollments e ON s.student\_id = e.student\_id

GROUP BY s.name;

| **name** | **total\_courses** |
| --- | --- |
| Ali | 1 |
| Sara | 2 |
| Ahmed | 1 |
| Zara | 1 |