

MySQL Lab Questions - Complete Solutions

Davis Batch
21st November 2025

November 21, 2025

AAYUSH SOAM

1 PART 1 — SQL Queries

1.1 Q1. Create a table named students

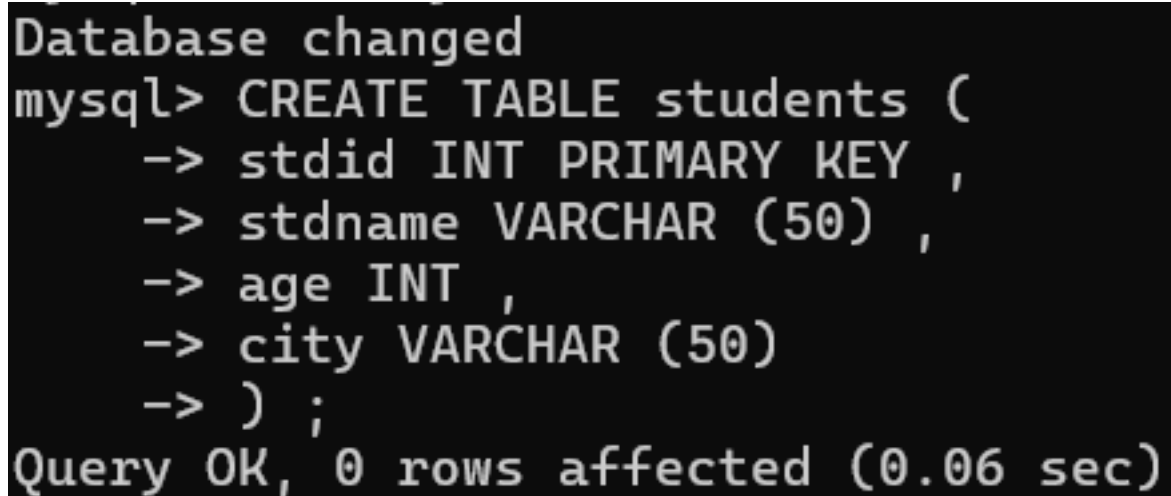
Question: Create a table named students with fields:

- stdid INT PRIMARY KEY
- stdname VARCHAR(50)
- age INT
- city VARCHAR(50)

SQL Query:

```
CREATE TABLE students (  
    stdid INT PRIMARY KEY,  
    stdname VARCHAR(50),  
    age INT,  
    city VARCHAR(50)  
);
```

Output:



```
Database changed  
mysql> CREATE TABLE students (  
    -> stdid INT PRIMARY KEY ,  
    -> stdname VARCHAR (50) ,  
    -> age INT ,  
    -> city VARCHAR (50)  
    -> ) ;  
Query OK, 0 rows affected (0.06 sec)
```

1.2 Q2. Insert records into the students table

Question: Insert the following records into the students table.

SQL Query:

```
INSERT INTO students (stdid, stdname, age, city) VALUES  
(1, 'Rohan', 20, 'Pune'),
```

```
(2, 'Meera', 22, 'Mumbai'),  
(3, 'Arjun', 21, 'Delhi'),  
(4, 'Kavya', 23, 'Pune'),  
(5, 'Neha', 22, 'Kolkata');
```

Output:

```
mysql> INSERT INTO students (stdid, stdname, age, city) VALUES  
-> (1, 'Rohan', 20, 'Pune'),  
-> (2, 'Meera', 22, 'Mumbai'),  
-> (3, 'Arjun', 21, 'Delhi'),  
-> (4, 'Kavya', 23, 'Pune'),  
-> (5, 'Neha', 22, 'Kolkata');  
Query OK, 5 rows affected (0.01 sec)  
Records: 5 Duplicates: 0 Warnings: 0
```

1.3 Q3. Display all student records

Question: Display all student records.

SQL Query:

```
SELECT * FROM students;
```

Output:

```
mysql> SELECT * FROM students ;  
+-----+-----+-----+-----+  
| stdid | stdname | age | city |  
+-----+-----+-----+-----+  
| 1 | Rohan | 20 | Pune |  
| 2 | Meera | 22 | Mumbai |  
| 3 | Arjun | 21 | Delhi |  
| 4 | Kavya | 23 | Pune |  
| 5 | Neha | 22 | Kolkata |  
+-----+-----+-----+-----+  
5 rows in set (0.00 sec)
```

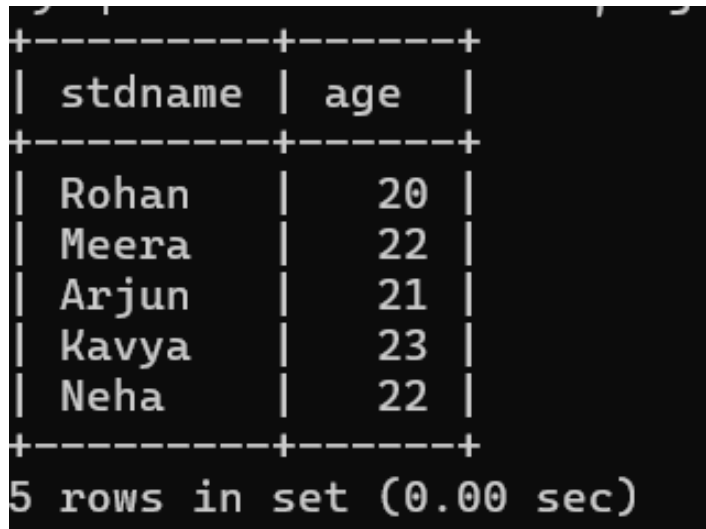
1.4 Q4. Display only the name and age of all students

Question: Display only the name and age of all students.

SQL Query:

```
SELECT stdname, age FROM students;
```

Output:

A screenshot of a MySQL terminal window showing the output of the SQL query 'SELECT stdname, age FROM students;'. The output is a table with two columns: 'stdname' and 'age'. The data rows are: Rohan (20), Meera (22), Arjun (21), Kavya (23), and Neha (22). The terminal also shows '5 rows in set (0.00 sec)' at the bottom.

stdname	age
Rohan	20
Meera	22
Arjun	21
Kavya	23
Neha	22

5 rows in set (0.00 sec)

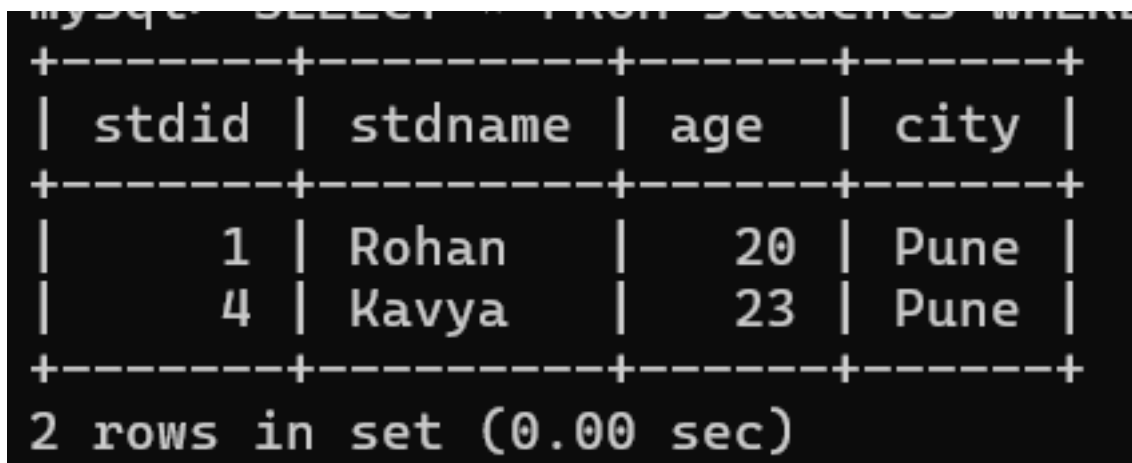
1.5 Q5. Display students who are from Pune

Question: Display students who are from Pune.

SQL Query:

```
SELECT * FROM students WHERE city = 'Pune';
```

Output:

A screenshot of a MySQL terminal window showing the output of the SQL query 'SELECT * FROM students WHERE city = 'Pune';'. The output is a table with four columns: 'stdid', 'stdname', 'age', and 'city'. The data rows are: 1 (Rohan, 20, Pune) and 4 (Kavya, 23, Pune). The terminal also shows '2 rows in set (0.00 sec)' at the bottom.

stdid	stdname	age	city
1	Rohan	20	Pune
4	Kavya	23	Pune

2 rows in set (0.00 sec)

1.6 Q6. Display students whose age is greater than 21

Question: Display students whose age is greater than 21.

SQL Query:

```
SELECT * FROM students WHERE age > 21;
```

Output:

```
+-----+-----+-----+-----+
| stdid | stdname | age  | city   |
+-----+-----+-----+-----+
|      2 | Meera   | 22   | Mumbai |
|      4 | Kavya   | 23   | Pune   |
|      5 | Neha    | 22   | Kolkata |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

1.7 Q7. Display students in descending order of age

Question: Display students in descending order of age.

SQL Query:

```
SELECT * FROM students ORDER BY age DESC;
```

Output:

```
+-----+-----+-----+-----+
| stdid | stdname | age  | city   |
+-----+-----+-----+-----+
|      4 | Kavya   | 23   | Pune   |
|      2 | Meera   | 22   | Mumbai |
|      5 | Neha    | 22   | Kolkata |
|      3 | Arjun   | 21   | Delhi  |
|      1 | Rohan   | 20   | Pune   |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

1.8 Q8. Count how many students belong to each city

Question: Count how many students belong to each city. (Use GROUP BY)

SQL Query:

```
SELECT city, COUNT(*) AS student_count
FROM students
GROUP BY city;
```

Output:

```
+-----+-----+
| city  | student_count |
+-----+-----+
| Pune  | 2             |
| Mumbai | 1             |
| Delhi  | 1             |
| Kolkata | 1             |
+-----+-----+
4 rows in set (0.01 sec)
```

1.9 Q9. Display students whose name starts with 'K'

Question: Display students whose name starts with 'K'. (Use LIKE)

SQL Query:

```
SELECT * FROM students WHERE stdname LIKE 'K%';
```

Output:

```
+-----+-----+-----+-----+
| stdid | stdname | age  | city  |
+-----+-----+-----+-----+
| 4     | Kavya  | 23   | Pune  |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

1.10 Q10. Delete student whose stdid = 5

Question: Delete student whose stdid = 5.

SQL Query:

```
DELETE FROM students WHERE stdid = 5;
```

Output:

```
mysql> DELETE FROM students WHERE stdid = 5;  
Query OK, 1 row affected (0.01 sec)
```

2 PART 2 — ALTER COMMAND QUESTIONS

2.1 Q11. Add a new column contact

Question: Add a new column contact VARCHAR(15) to the students table.

SQL Query:

```
ALTER TABLE students ADD contact VARCHAR(15);
```

Output:

```
mysql> ALTER TABLE students ADD contact VARCHAR(15);  
Query OK, 0 rows affected (0.02 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

2.2 Q12. Modify the data type of city column

Question: Modify the data type of city column to VARCHAR(100).

SQL Query:

```
ALTER TABLE students MODIFY city VARCHAR(100);
```

Output:

```
mysql> ALTER TABLE students MODIFY city VARCHAR(100);  
Query OK, 4 rows affected (0.07 sec)  
Records: 4 Duplicates: 0 Warnings: 0
```

2.3 Q13. Rename the column stdname to student_name

Question: Rename the column stdname to student_name.

SQL Query:

```
ALTER TABLE students RENAME COLUMN stdname TO student_name;
```

Output:

```
mysql> ALTER TABLE students RENAME COLUMN stdname TO student_name;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

2.4 Q14. Drop the column contact from the table

Question: Drop the column contact from the table.

SQL Query:

```
ALTER TABLE students DROP COLUMN contact;
```

Output:

```
mysql> ALTER TABLE students DROP COLUMN contact;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

2.5 Q15. Add a new column gender

Question: Add a new column gender ENUM('M','F').

SQL Query:

```
ALTER TABLE students ADD gender ENUM('M', 'F');
```

Output:

```
mysql> ALTER TABLE students ADD gender ENUM('M', 'F');  
Query OK, 0 rows affected (0.02 sec)  
Records: 0 Duplicates: 0 Warnings: 0
```

3 PART 3 — JOIN PRACTICE

3.1 Setup for Join Questions

Create and populate required tables:

```
-- Recreate students table with updated schema
CREATE TABLE students (
    stdid INT PRIMARY KEY,
    student_name VARCHAR(50),
    city VARCHAR(50)
);

INSERT INTO students VALUES
(1, 'Rohan', 'Pune'),
(2, 'Meera', 'Mumbai'),
(3, 'Arjun', 'Delhi'),
(4, 'Kavya', 'Pune');

-- Create marks table
CREATE TABLE marks (
    stdid INT,
    subject VARCHAR(50),
    marks INT
);

INSERT INTO marks VALUES
(1, 'Maths', 88),
(2, 'Maths', 76),
(3, 'Maths', 92),
(5, 'Maths', 67);
```

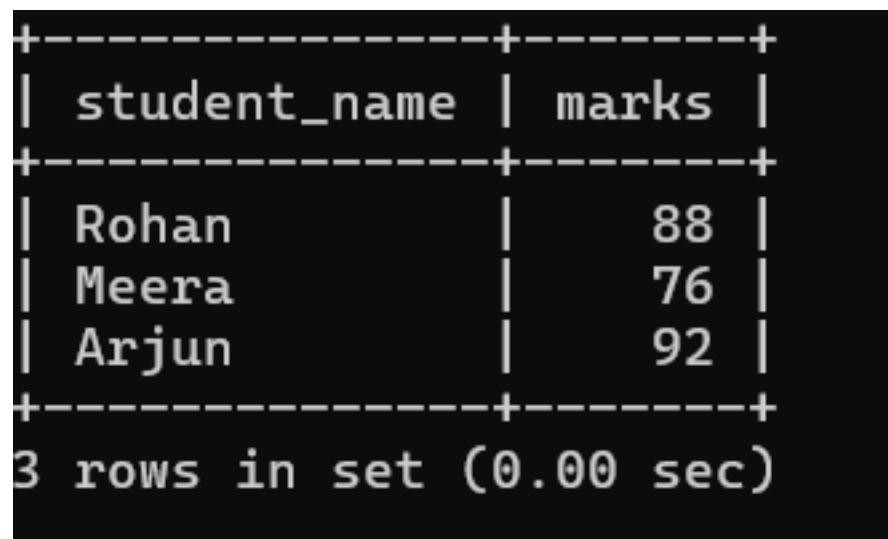
3.2 Q16. INNER JOIN

Question: Display student name and marks of only those students who have matching IDs in both tables. (Students without marks should not appear.)

SQL Query:

```
SELECT s.student_name, m.marks
FROM students s
INNER JOIN marks m ON s.stdid = m.stdid;
```

Output:



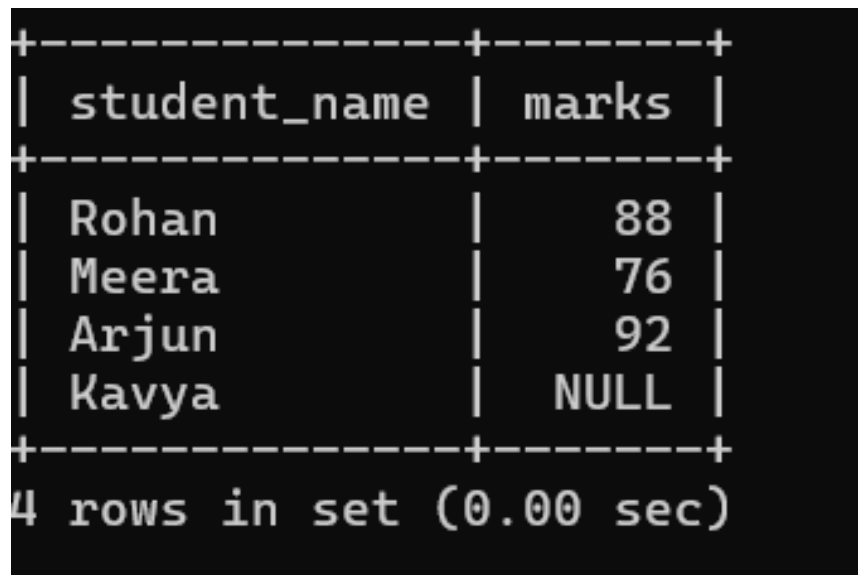
```
+-----+-----+
| student_name | marks |
+-----+-----+
| Rohan       | 88    |
| Meera       | 76    |
| Arjun       | 92    |
+-----+-----+
3 rows in set (0.00 sec)
```

3.3 Q17. LEFT JOIN

Question: Display all students and their marks. (If marks not available, show NULL.)
SQL Query:

```
SELECT s.student_name, m.marks
FROM students s
LEFT JOIN marks m ON s.stdid = m.stdid;
```

Output:



```
+-----+-----+
| student_name | marks |
+-----+-----+
| Rohan       | 88    |
| Meera       | 76    |
| Arjun       | 92    |
| Kavya       | NULL  |
+-----+-----+
4 rows in set (0.00 sec)
```

3.4 Q18. RIGHT JOIN

Question: Display all marks records along with student names. (If student doesn't exist in students table, show NULL.)

SQL Query:

```
SELECT s.student_name, m.marks
FROM students s
RIGHT JOIN marks m ON s.stidid = m.stdid;
```

Output:

student_name	marks
Rohan	88
Meera	76
Arjun	92
NULL	67

4 rows in set (0.00 sec)

3.5 Q19. CROSS JOIN

Question: Display all possible combinations of students and subjects. (Use CROSS JOIN between students and marks table to show every pair.)

SQL Query:

```
SELECT s.student_name, m.subject, m.marks
FROM students s
CROSS JOIN marks m;
```

Output:

student_name	subject	marks
Kavya	Maths	88
Arjun	Maths	88
Meera	Maths	88
Rohan	Maths	88
Kavya	Maths	76
Arjun	Maths	76
Meera	Maths	76
Rohan	Maths	76
Kavya	Maths	92
Arjun	Maths	92
Meera	Maths	92
Rohan	Maths	92
Kavya	Maths	67
Arjun	Maths	67
Meera	Maths	67
Rohan	Maths	67

16 rows in set (0.00 sec)

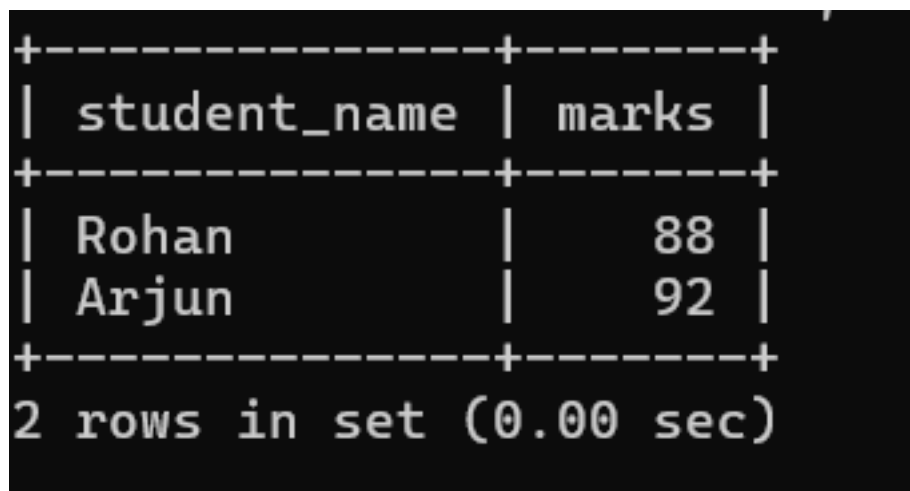
3.6 Q20. JOIN with Filtering

Question: Using INNER JOIN, display students who scored more than 80.

SQL Query:

```
SELECT s.student_name, m.marks
FROM students s
INNER JOIN marks m ON s.stdid = m.stdid
WHERE m.marks > 80;
```

Output:



student_name	marks
Rohan	88
Arjun	92

2 rows in set (0.00 sec)