

# Davis- MySQL(Practice queries )

Date - 21-11-25

Q1. Create a table named students with fields:

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

```
create table student(
stdid int primary key,
stdname varchar(50),
age int,
city varchar(50)
);
```

```
select * from student
```

The screenshot shows the MySQL Workbench interface. The SQL tab contains the following code:

```
1 •  create database school;
2 •  use school;
3
4 •  create table student(
5      stdid int primary key,
6      stdname varchar(50),
7      age int,
8      city varchar(50)
9 );
10
11 •  select * from student;
```

The results tab shows the output of the select query:

	stdid	stdname	age	city
*	NULL	NULL	NULL	NULL

Q2. Insert the following records into the students table:

stdid	stdname	age	city
1	Rohan	20	Pune
2	Meera	22	Mumbai
3	Arjun	21	Delhi
4	Kavya	23	Pune
5	Neha	22	Kolkata

```
INSERT INTO student (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');
```

```
10  
11 • select * from student  
12  
13 ✘ INSERT INTO student (stdid, stdname, age, city) VALUES  
14     (1, 'Rohan', 20, 'Pune'),  
15     (2, 'Meera', 22, 'Mumbai'),  
16     (3, 'Arjun', 21, 'Delhi'),  
17     (4, 'Kavya', 23, 'Pune'),  
18     (5, 'Neha', 22, 'Kolkata');  
19 • select * from student  
20
```

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

	stdid	stdname	age	city
*	1	Rohan	20	Pune
*	2	Meera	22	Mumbai
*	3	Arjun	21	Delhi
*	4	Kavya	23	Pune
*	5	Neha	22	Kolkata
*	NULL	NULL	NULL	NULL

student 2 ×

### Q3. Display all student records

```
select * from student
```

```
10  
11 • select * from student  
12  
13 ✘ INSERT INTO student (stdid, stdname, age, city) VALUES  
14     (1, 'Rohan', 20, 'Pune'),  
15     (2, 'Meera', 22, 'Mumbai'),  
16     (3, 'Arjun', 21, 'Delhi'),  
17     (4, 'Kavya', 23, 'Pune'),  
18     (5, 'Neha', 22, 'Kolkata');  
19 • select * from student
```

```
20
```

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

	stdid	stdname	age	city
1	1	Rohan	20	Pune
2	2	Meera	22	Mumbai
3	3	Arjun	21	Delhi
4	4	Kavya	23	Pune
5	5	Neha	22	Kolkata
	NULL	NULL	NULL	NULL

:student 2 x

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

```
select stdname, age  
from student;
```

```
16      (3, 'Arjun', 21, 'Delhi'),  
17      (4, 'Kavya', 23, 'Pune'),  
18      (5, 'Neha', 22, 'Kolkata');  
19 •  select * from student  
20  
21 ✖  select * from student  
22  
23  select stdname, age  
24  from student;  
25  
26
```

The screenshot shows a database interface with a code editor at the top and a result grid below. The code editor contains several lines of SQL and some numbered lines (16-26). Lines 16-20 show tuple definitions. Line 19 is a comment. Lines 21-22 show a failed attempt to select all columns. Lines 23-24 show the correct query to select stdname and age from the student table. Line 25 is a blank line. Line 26 is another blank line. Below the code editor is a toolbar with buttons for Result Grid, Filter Rows, Export, and Wrap Cell Content. The result grid displays the following data:

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

At the bottom left of the interface, it says "student 4 X".

## Q5. Display students who are from Pune

```
select *  
from student  
where city = 'Pune';
```

The screenshot shows the MySQL Workbench interface. The query editor window displays the following SQL code:

```
19 •  select * from student  
20  
21 ✘  select * from student  
22  
23  select stdname, age  
24  from student;  
25  
26 •  select *  
27  from student  
28  where city = 'Pune';  
29
```

The line `28 where city = 'Pune';` is highlighted with a blue selection bar. Below the editor is the result grid, which shows the following data:

	stdid	stdname	age	city
▶	1	Rohan	20	Pune
	4	Kavya	23	Pune
*	NULL	NULL	NULL	NULL

Q6. Display students whose age is greater than 21.

```
select name,age  
from student  
where age >21;
```

The screenshot shows the MySQL Workbench interface. The query editor window contains the following code:

```
22  
23      select stdname, age  
24      from student;  
25  
26 •      select *  
27      from student  
28      where city = 'Pune';  
29  
30 •      select age  
31      from student  
32      where age >21;
```

The result grid shows the following data:

age
22
23
22

Q7. Display students in descending order of age.

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

The screenshot shows the MySQL Workbench interface. The SQL editor window contains the following code:

```
26 •    select *  
27      from student  
28      where city = 'Pune';  
29  
30 •    select age  
31      from student  
32      where age >21;  
33  
34 •    SELECT *  
35      FROM student  
36      ORDER BY age DESC;
```

The Result Grid shows the following data:

	stdid	stdname	age	city
▶	4	Kavya	23	Pune
	2	Meera	22	Mumbai
	5	Neha	22	Kolkata
	3	Arjun	21	Delhi
	1	Rohan	20	Pune
*	NULL	NULL	NULL	NULL

Q8. Count how many students belong to each city.  
(Use GROUP BY)

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

```
34 •      SELECT *  
35      FROM student  
36      ORDER BY age DESC;  
37
```

	stdid	stdname	age	city
▶	4	Kavya	23	Pune
	2	Meera	22	Mumbai
	5	Neha	22	Kolkata
	3	Arjun	21	Delhi
◀	1	Rohan	20	Pune
◀	NULL	NULL	NULL	NULL

Q9. Display student whose name starts with 'K'.  
(Use LIKE)

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

The screenshot shows a MySQL Workbench interface. The SQL editor window contains the following code:

```
-- -----
41
42 •    SELECT *
43      FROM student
44      WHERE stdname LIKE 'K%' ;
```

The line starting with '•' indicates the current active statement. The results grid below shows the output of the query:

	stdid	stdname	age	city
▶	4	Kavya	23	Pune
✳	NULL	NULL	NULL	NULL

Q10. Delete student whose stdid = 5.

```
DELETE FROM student
```

```
WHERE stdid = 5;
```

```
select * from student
```

```
46 •    DELETE FROM student
```

```
47      WHERE stdid = 5;
```

```
48
```

```
49 •    select * from student
```

```
50
```

The screenshot shows the MySQL Workbench interface. At the top, there is a command line with several numbered lines of SQL code. Lines 46 and 47 are standard delete statements. Line 49 is a select statement, which is highlighted with a blue background. Line 50 is a blank line. Below the command line is a toolbar with icons for Result Grid, Filter Rows, and Edit. The main area displays a result grid for the select statement. The grid has four columns: stdid, stdname, age, and city. It contains five rows of data, indexed from 1 to 5. Row 5 is a new row with all columns set to NULL. The data is as follows:

	stdid	stdname	age	city
1	1	Rohan	20	Pune
2	2	Meera	22	Mumbai
3	3	Arjun	21	Delhi
4	4	Kavya	23	Pune
*	HULL	HULL	HULL	HULL

## PART 2 — ALTER COMMAND QUESTIONS

Q11. Add a new column contact VARCHAR(15) to the student table.

```
ALTER TABLE student  
ADD contact VARCHAR(15);  
select * from student
```

The screenshot shows the MySQL Workbench interface. In the SQL editor pane, there are four numbered statements:

- 50 (unnumbered)
- 51 (red X) ALTER TABLE student
- 52 ADD contact VARCHAR(15);
- 53 (green dot) select \* from student

The 'Result Grid' tab is selected, displaying the contents of the 'student' table. The table has five columns: stdid, stdname, age, city, and contact. The data is as follows:

	stdid	stdname	age	city	contact
▶	1	Rohan	20	Pune	NULL
	2	Meera	22	Mumbai	NULL
	3	Arjun	21	Delhi	NULL
	4	Kavya	23	Pune	NULL
●	NULL	NULL	NULL	NULL	NULL

Q12. Modify the data type of city column to VARCHAR(100).

```
ALTER TABLE student  
MODIFY city VARCHAR(100);  
select * from student
```

```
55 • ALTER TABLE student  
56     MODIFY city VARCHAR(100);  
57 • select * from student  
58 |
```

Result Grid | Filter Rows: . | Edit:     

stdid	stdname	age	city	contact
1	Rohan	20	Pune	NULL
2	Meera	22	Mumbai	NULL
3	Arjun	21	Delhi	NULL
4	Kavya	23	Pune	NULL
HULL	HULL	HULL	HULL	HULL

Q13. Rename the column stdname to student\_name.

```
ALTER TABLE student
```

```
CHANGE stdname student_name VARCHAR(255);  
select * from student
```

58

```
59 ✘ ALTER TABLE student  
60      CHANGE stdname student_name VARCHAR(255);  
61 •  select * from student
```

62

The screenshot shows the MySQL Workbench interface with a result grid. The grid has five columns: stdid, student\_name, age, city, and contact. The data consists of five rows:

	stdid	student_name	age	city	contact
▶	1	Rohan	20	Pune	NULL
	2	Meera	22	Mumbai	NULL
	3	Arjun	21	Delhi	NULL
	4	Kavya	23	Pune	NULL
⋮	NULL	NULL	NULL	NULL	NULL

Q14. Drop the column contact from the table.

```
ALTER TABLE student  
DROP COLUMN contact;  
select * from student
```

```
63 ✘ ALTER TABLE student  
64     DROP COLUMN contact;  
65 • select * from student  
66
```

Result Grid



Filter Rows:

Edit:



	stdid	student_name	age	city
1	Rohan	20	Pune	
2	Meera	22	Mumbai	
3	Arjun	21	Delhi	
4	Kavya	23	Pune	
	NULL	NULL	NULL	NULL

Q15. Add a new column gender ENUM('M','F').

```
ALTER TABLE student  
ADD gender ENUM('M','F');  
select * from student
```

```
66
67 * ALTER TABLE student
68     ADD gender ENUM('M','F');
69 • select * from student
70
```

Result Grid | Filter Rows: | Edit: |

	stdid	student_name	age	city	gender
▶	1	Rohan	20	Pune	NULL
	2	Meera	22	Mumbai	NULL
	3	Arjun	21	Delhi	NULL
	4	Kavya	23	Pune	NULL
*	NULL	NULL	NULL	NULL	NULL

student 15 x

## PART 3 — JOIN PRACTICE

### INNER JOIN

Q16. Display student name and marks of only those students who have matching IDs in both tables.

CREATE TABLE student(  
 stdid int primary key,

```

student_name varchar(50),
city varchar(55)
);
insert into student(stdid,student_name,city)
values(1,"Rohan","Pune"),
(2,"Meera","Mumbai"),
(3,"Arjun","Delhi"),
(4,"Kavya","Pune");
CREATE TABLE marks(
stdid int primary key,
subject varchar(25),
marks int not null
);
insert into marks(stdid,subject,marks)
values(1,"Maths",88),
(2,"Maths",76),
(3,"Maths",92),
(5,"Maths",67);
-- --INNER JOIN---
SELECT student.student_name, marks.marks
FROM student
INNER JOIN marks
ON student.stdid = marks.stdid;

```

Result Grid | Filter Rows:  Export

	student_name	marks
▶	Rohan	88
	Meera	76
	Arjun	92

## LEFT JOIN

Q17. Display all students and their marks.

```
CREATE TABLE student(
    stdid int primary key,
    student_name varchar(50),
    city varchar(55)
);
insert into student(stdid,student_name,city)
values(1,"Rohan","Pune"),
(2,"Meera","Mumbai"),
(3,"Arjun","Delhi"),
(4,"Kavya","Pune");
CREATE TABLE marks(
    stdid int primary key,
    subject varchar(25),
    marks int not null
);
insert into marks(stdid,subject,marks)
values(1,"Maths",88),
(2,"Maths",76),
(3,"Maths",92),
(5,"Maths",67);
---LEFT JOIN---
SELECT students.student_name, marks.marks
FROM students
LEFT JOIN marks
ON students.stdid = marks.stdid;
```

The screenshot shows a database result grid with the following structure:

	student_name	marks
▶	Rohan	88
	Meera	76
	Arjun	92
	Kavya	NULL

## RIGHT JOIN

Q18. Display all marks records along with student names.

```

CREATE TABLE student(
stdid int primary key,
student_name varchar(50),
city varchar(55)
);
insert into student(stdid,student_name,city)
values(1,"Rohan","Pune"),
(2,"Meera","Mumbai"),
(3,"Arjun","Delhi"),
(4,"Kavya","Pune");
CREATE TABLE marks(
stdid int primary key,
subject varchar(25),
marks int not null
);
insert into marks(stdid,subject,marks)
values(1,"Maths",88),
(2,"Maths",76),
(3,"Maths",92),
(5,"Maths",67);

```

-----RIGHT JOIN-----

```
SELECT student.student_name, marks.marks
FROM student
RIGHT JOIN marks
ON student.stdid = marks.stdid;
```

	student_name	marks
▶	Rohan	88
	Meera	76
	Arjun	92
	HULL	67

## CROSS JOIN

Q19. Display all possible combinations of students and subjects.

```
CREATE TABLE student(
stdid int primary key,
student_name varchar(50),
city varchar(55)
);
insert into student(stdid,student_name,city)
values(1,"Rohan","Pune"),
(2,"Meera","Mumbai"),
(3,"Arjun","Delhi"),
(4,"Kavya","Pune");
CREATE TABLE marks(
stdid int primary key,
subject varchar(25),
marks int not null
);
insert into marks(stdid,subject,marks)
values(1,"Maths",88),
(2,"Maths",76),
```

```
(3,"Maths",92),
(5,"Maths",67)
-----CROS JOIN-----
SELECT student.student_name, marks.subject
FROM student
CROSS JOIN marks;
```

Result Grid | Filter Rows: [ ]

	student_name	subject
▶	Kavya	Maths
	Arjun	Maths
	Meera	Maths
	Rohan	Maths
	Kavya	Maths
	Arjun	Maths

## JOIN with Filtering

Q20. Using INNER JOIN, display students who scored more than 80.

```
CREATE TABLE student(
stdid int primary key,
student_name varchar(50),
city varchar(55)
);
insert into student(stdid,student_name,city)
values(1,"Rohan","Pune"),
(2,"Meera","Mumbai"),
(3,"Arjun","Delhi"),
(4,"Kavya","Pune");
CREATE TABLE marks(
stdid int primary key,
subject varchar(25),
marks int not null
```

```
};

insert into marks(stdid,subject,marks)
values(1,"Maths",88),
(2,"Maths",76),
(3,"Maths",92),
(5,"Maths",67)
-----INNER JOIN-----
SELECT student.student_name, marks.marks
FROM student
INNER JOIN marks
ON student.stdid = marks.stdid
WHERE marks.marks > 80;
```

Result Grid | Filter Rows:  | Export:

	student_name	marks
▶	Rohan	88
	Arjun	92