

LAB QUESTIONS

SQL

CEHTAN

Q1. Create a table named students with fields:

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

Query:

```
CREATE TABLE students (  
    stdid int primary key,  
    stdname varchar(50),  
    age int,  
    city varchar(50)  
);
```

```

1 • create database Collage;
2 • use Collage;
3 • CREATE TABLE students (
4     stdid INT PRIMARY KEY,
5     stdname VARCHAR(50),
6     age INT,
7     city VARCHAR(50)
8 );
9 •
10 • select * from students;
11

```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

I A

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

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');

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

Q3. Display all student records.

Query:

```
select * from students
```

Result:

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

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

Query:

```
select stdname as Name, age as Age from students
```

Result:

Name	Age
Rohan	20
Meera	22
Arjun	21
Kavya	23
Neha	22

Q5. Display students who are from Pune.

Query:

```
select *  
from students  
where city == 'Pune'
```

Result:

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

Q6. Display students whose age is greater than 21.

Query:

```
select *  
from students  
where age>21
```

Result:

stdid	stdname	age	city
2	Meera	22	Mumbai
4	Kavya	23	Pune
5	Neha	22	Kolkata

Q7. Display students in descending order of age.

Query:

```
select *  
from students  
order by age desc
```

Result:

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

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

Query:

```
select city, count(stdid)  
from students  
group by city
```

Result:

city	count(stdid)
Delhi	1
Kolkata	1
Mumbai	1
Pune	2

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

Query:

```
SELECT *
```

```
FROM students
```

```
WHERE stdname LIKE 'K%';
```

Result:

stdid	stdname	age	city
4	Kavya	23	Pune

Q10. Delete student whose stdid = 5.

Query:

```
DELETE FROM students
```

```
WHERE stdid = 5;
```

Result:

SQL query successfully executed. However, the result set is empty.

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

Query:

```
ALTER TABLE students
```

```
ADD COLUMN contact VARCHAR(15);
```

Result:

SQL query successfully executed. However, the result set is empty.

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

Query:

```
ALTER TABLE students
```

```
MODIFY COLUMN city VARCHAR(100);
```

Q13. Rename the column stdname to student_name.

Query:

```
ALTER TABLE students
```

```
CHANGE COLUMN stdname student_name VARCHAR(255);
```

Q14. Drop the column contact from the table.

Query:

```
ALTER TABLE students
```

```
DROP COLUMN contact;
```

Result:

SQL query successfully executed. However, the result set is empty.

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

Query:

```
ALTER TABLE students
```

```
ADD gender ENUM('M', 'F');
```

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

Query:

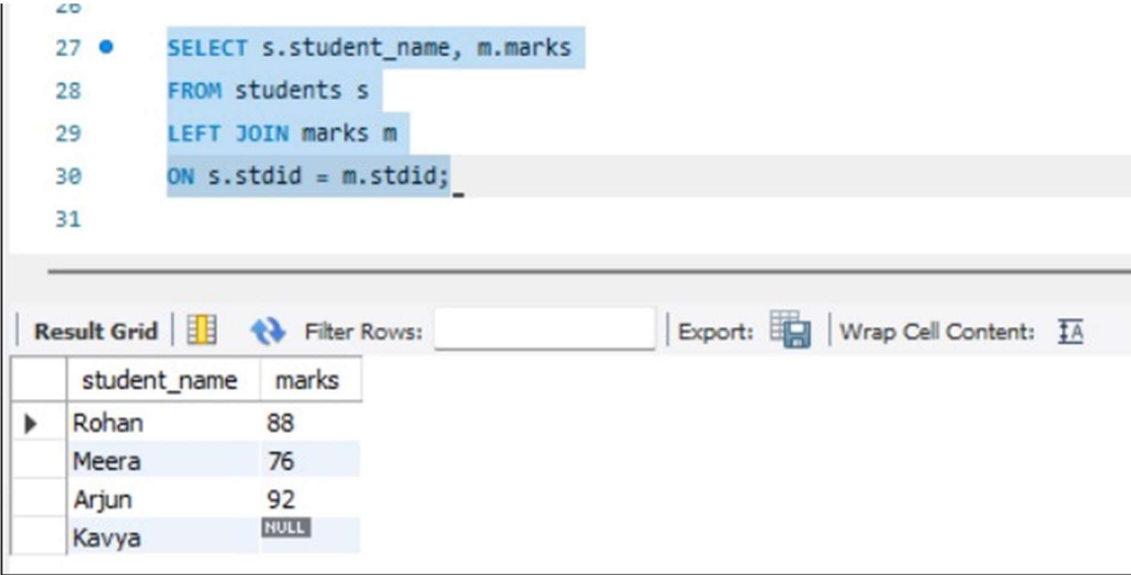
```
SELECT s.student_name, m.marks
```

```
FROM students s
```

```
INNER JOIN marks m
```

```
ON s.stdid = m.stdid;
```

Result:



The screenshot shows a SQL query editor with a query window and a result grid below it. The query window contains the following SQL code:

```
27 SELECT s.student_name, m.marks
28 FROM students s
29 LEFT JOIN marks m
30 ON s.stdid = m.stdid;
31
```

The result grid displays the output of the query. It has a toolbar with options: Result Grid, Filter Rows, Export, and Wrap Cell Content. The data is presented in a table with two columns: student_name and marks.

	student_name	marks
▶	Rohan	88
	Meera	76
	Arjun	92
	Kavya	NULL

Q17. Display all students and their marks.

Query:

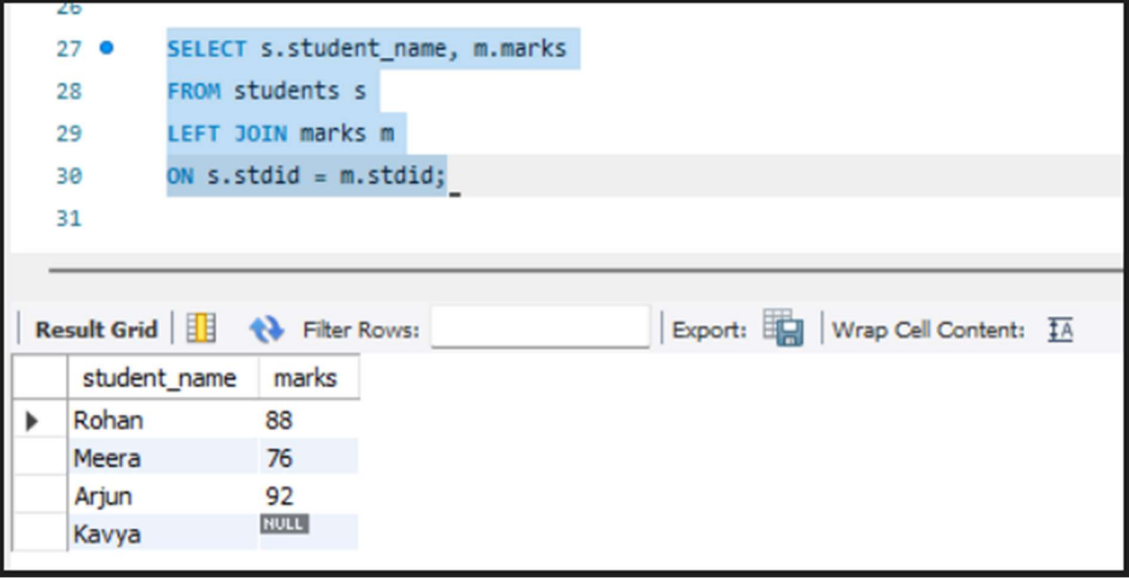
```
SELECT s.student_name, m.marks
```

```
FROM students s
```

```
LEFT JOIN marks m
```

```
ON s.stdid = m.stdid;
```

Result:



The screenshot shows a database query editor with a SQL query and its results. The query is a LEFT JOIN between the 'students' table and the 'marks' table. The results are displayed in a table grid with columns 'student_name' and 'marks'.

```
26
27 • SELECT s.student_name, m.marks
28 FROM students s
29 LEFT JOIN marks m
30 ON s.stdid = m.stdid;
31
```

	student_name	marks
▶	Rohan	88
	Meera	76
	Arjun	92
	Kavya	NULL

Q18. Display all marks records along with student names.

Query:

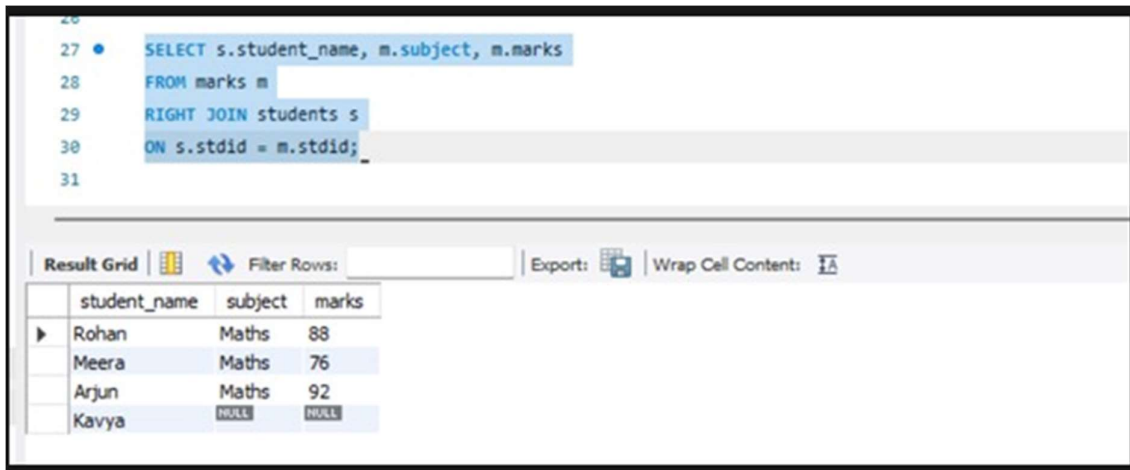
```
SELECT s.student_name, m.subject, m.marks
```

```
FROM marks m
```

```
RIGHT JOIN students s
```

```
ON s.stdid = m.stdid;
```

Result:



The screenshot shows a SQL query editor with the following code:

```
26  
27 • SELECT s.student_name, m.subject, m.marks  
28 FROM marks m  
29 RIGHT JOIN students s  
30 ON s.stdid = m.stdid;  
31
```

Below the query editor is a toolbar with the following options: Result Grid, Filter Rows, Export, and Wrap Cell Content. The Result Grid is active, displaying the following data:

	student_name	subject	marks
▶	Rohan	Maths	88
	Meera	Maths	76
	Arjun	Maths	92
	Kavya	NULL	NULL

Q19. Display all possible combinations of students and subjects.

Query:

```
SELECT s.student_name, m.subject
```

```
FROM students s
```

```
CROSS JOIN marks m;
```

Result:

The screenshot shows a database query execution interface. The SQL query is as follows:

```
27 SELECT s.student_name, m.subject
28 FROM students s
29 CROSS JOIN marks m;
30
31
```

The results are displayed in a table with two columns: student_name and subject. The data is as follows:

student_name	subject
Kavya	Maths
Arjun	Maths
Mipera	Maths
Rohan	Maths
Kavya	Maths

The interface also shows an 'Action Output' section with the following log:

#	Time	Action	Message
46	14:42:23	create database Collage	1 row(s) affected
47	14:42:23	use Collage	0 row(s) affected
48	14:42:23	CREATE TABLE students (stdid INT PRIMARY KEY, student_name VARCHAR(50), city VARCHAR(100))	0 row(s) affected
49	14:42:23	INSERT INTO students (stdid, student_name, city) VALUES (1, 'Rohan', 'Pune'), (2, 'Meera', 'Mumbai'), (3, 'Ajun', '...)	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0
50	14:42:23	CREATE TABLE marks (stdid INT, subject VARCHAR(50), marks INT)	0 row(s) affected
51	14:42:23	INSERT INTO marks (stdid, subject, marks) VALUES (1, 'Maths', 88), (2, 'Maths', 76), (3, 'Maths', 52), (5, 'Maths', ...)	4 row(s) affected Records: 4 Duplicates: 0 Warnings: 0
52	14:42:23	SELECT s.student_name, m.marks FROM students s INNER JOIN marks m ON s.stdid = m.stdid LIMIT 0, 1000	3 row(s) returned
53	14:45:04	SELECT s.student_name, m.marks FROM students s LEFT JOIN marks m ON s.stdid = m.stdid LIMIT 0, 1000	4 row(s) returned
54	14:46:45	SELECT s.student_name, m.subject, m.marks FROM marks m RIGHT JOIN students s ON s.stdid = m.stdid LIMIT ...	4 row(s) returned
55	14:47:25	SELECT s.student_name, m.subject FROM students s CROSS JOIN marks m LIMIT 0, 1000	16 row(s) returned

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

Result:

```
SELECT s.student_name, m.marks
```

```
FROM students s
```

```
INNER JOIN marks m
```

```
ON s.stdid = m.stdid
```

```
WHERE m.marks > 80;
```

Result:

The screenshot shows a database query execution interface. The SQL query is as follows:

```
26
27 SELECT s.student_name, m.marks
28 FROM students s
29 INNER JOIN marks m
30 ON s.stdid = m.stdid
31 WHERE m.marks > 80;
32
```

The results are displayed in a table with two columns: student_name and marks. The data is as follows:

student_name	marks
Rohan	88
Arjun	92