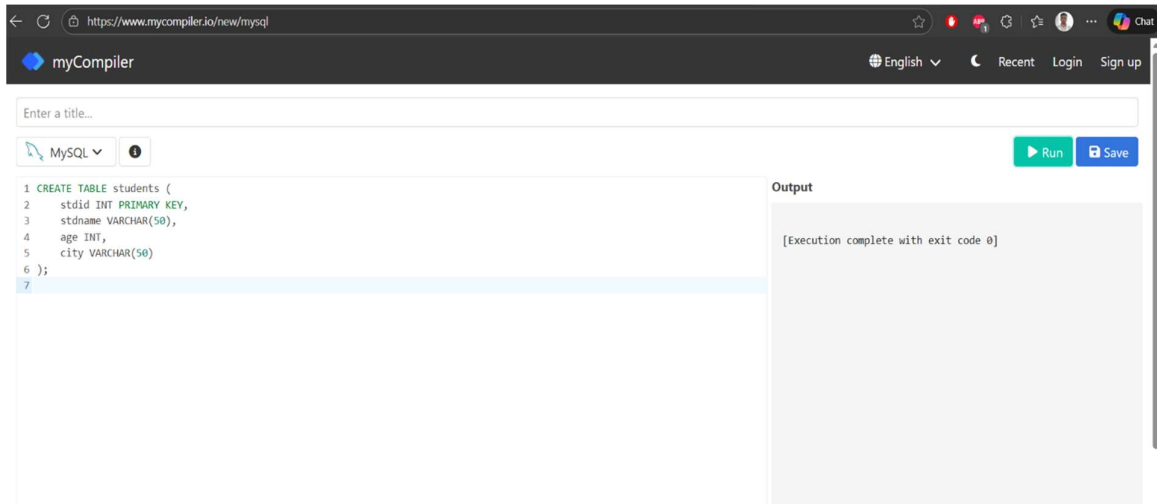


PART 1 – SQL Queries

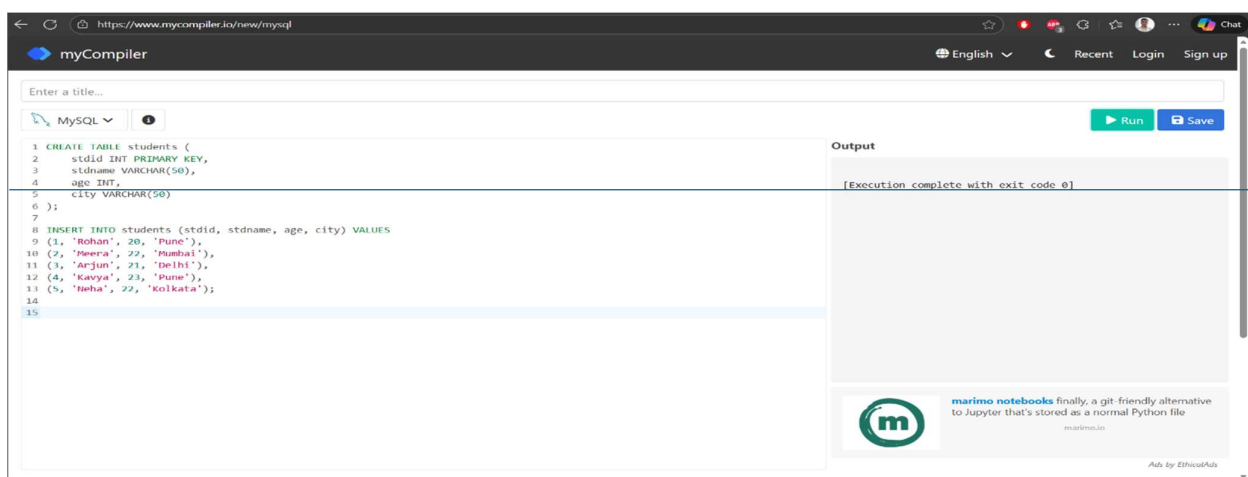
Q1. Create a table named students with fields:

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



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



Q3. Display all student records.

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

```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 SELECT * FROM students;
```

The output window displays the following 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

Below the output table, it states: [Execution complete with exit code 0].

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

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

```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 SELECT stdname, age  
16 FROM students;
```

The output window displays the following table:

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

Below the output table, it states: [Execution complete with exit code 0].

Q5. Display students who are from Pune.

myCompiler

Enter a title...

MySQL

```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 SELECT *  
16 FROM students  
17 WHERE city = 'Pune';  
18  
19
```

Run Save

Output

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

[Execution complete with exit code 0]

marimo notebooks finally, a git-friendly alternative to Jupyter that's stored as a normal Python file

marimo

Ads by EthicalAds

Q6. Display students whose age is greater than 21.

myCompiler

Enter a title...

MySQL

```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 SELECT *  
16 FROM students  
17 WHERE age > 21;  
18
```

Run Save

Output

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

[Execution complete with exit code 0]

marimo notebooks finally, a git-friendly alternative to Jupyter that's stored as a normal Python file

marimo

Ads by EthicalAds

Q7. Display students in descending order of age.

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

```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 SELECT *  
16 FROM students  
17 ORDER BY age DESC;  
18  
19
```

The output window displays the following table:

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

[Execution complete with exit code 0]

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

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

```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 SELECT city, COUNT(*) AS student_count  
16 FROM students  
17 GROUP BY city;  
18  
19  
20
```

The output window displays the following table:

city	student_count
Pune	2
Mumbai	1
Delhi	1
Kolkata	1

[Execution complete with exit code 0]

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

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

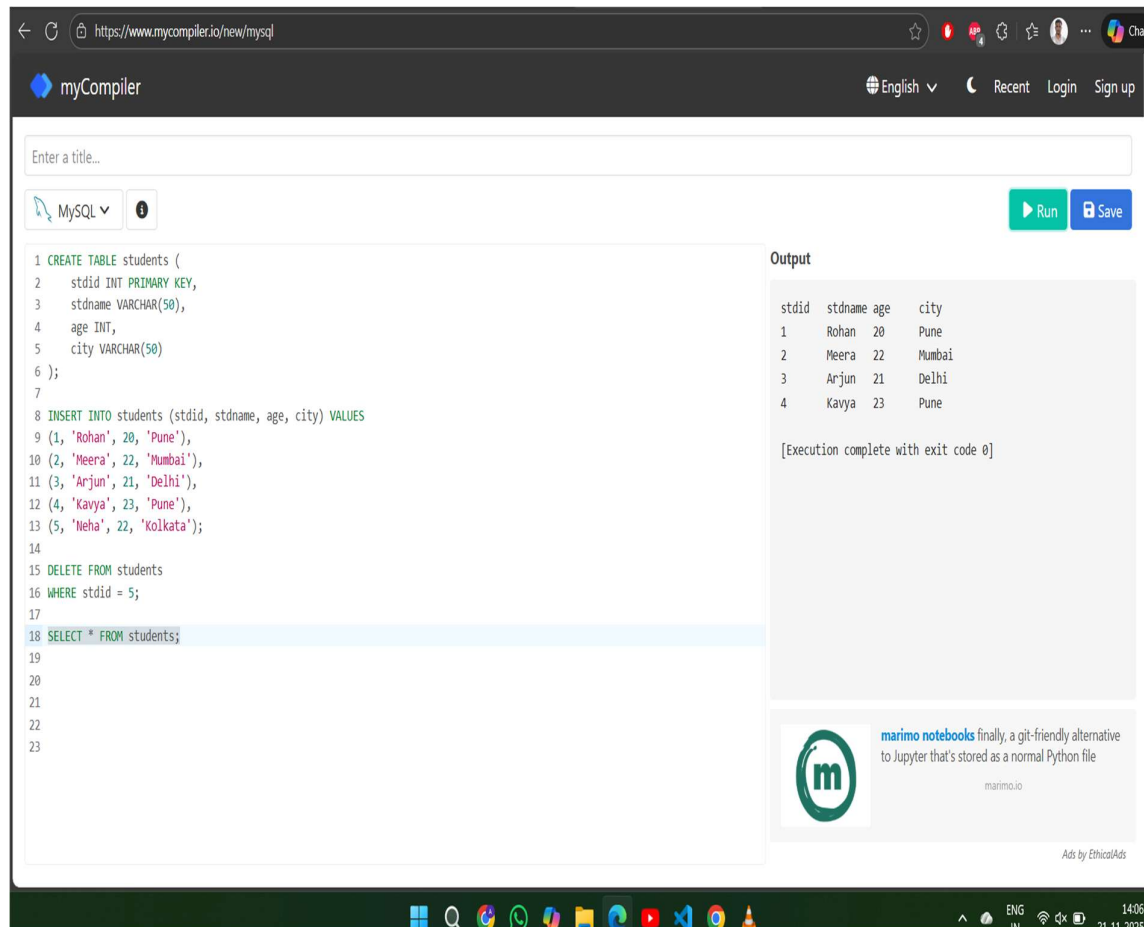
```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 SELECT *  
16 FROM students  
17 WHERE stdname LIKE 'K%';  
18  
19  
20  
21
```

The output window displays the following table:

stdid	stdname	age	city
4	Kavya	23	Pune

[Execution complete with exit code 0]

Q10. Delete student whose stdid = 5



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

```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 DELETE FROM students  
16 WHERE stdid = 5;  
17  
18 SELECT * FROM students;  
19  
20  
21  
22  
23
```

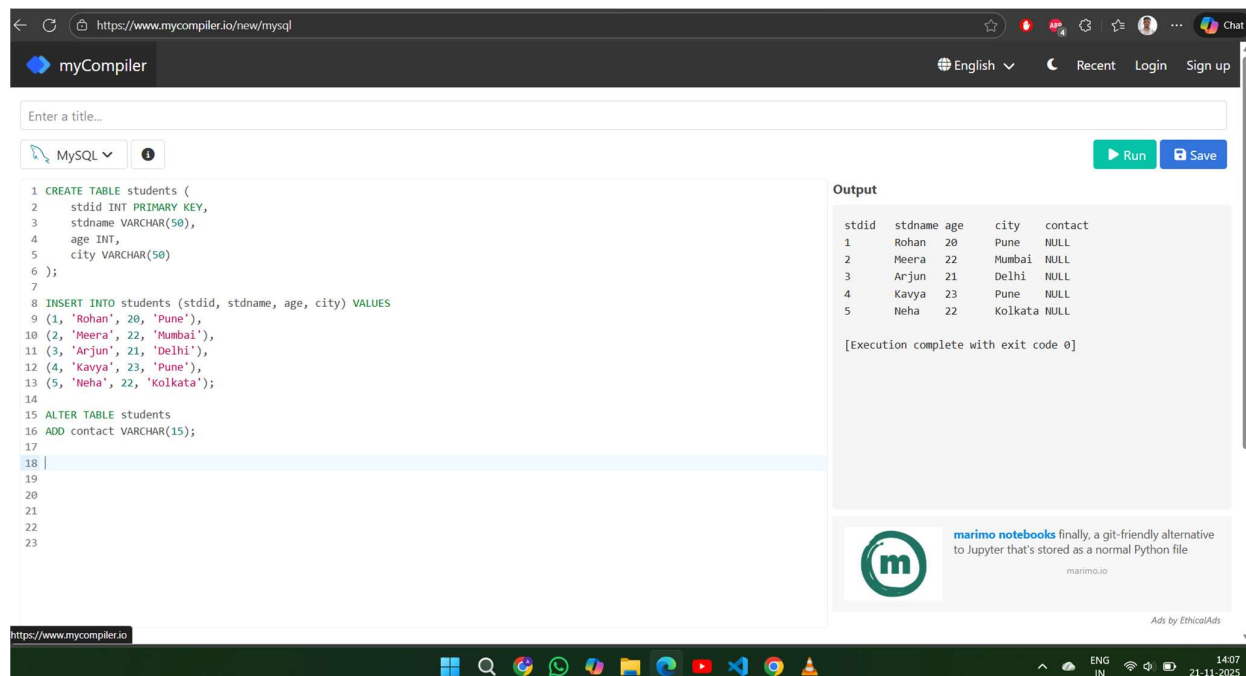
The output window displays the result of the SELECT query:

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

[Execution complete with exit code 0]

PART 2 – ALTER COMMAND QUESTIONS

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



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

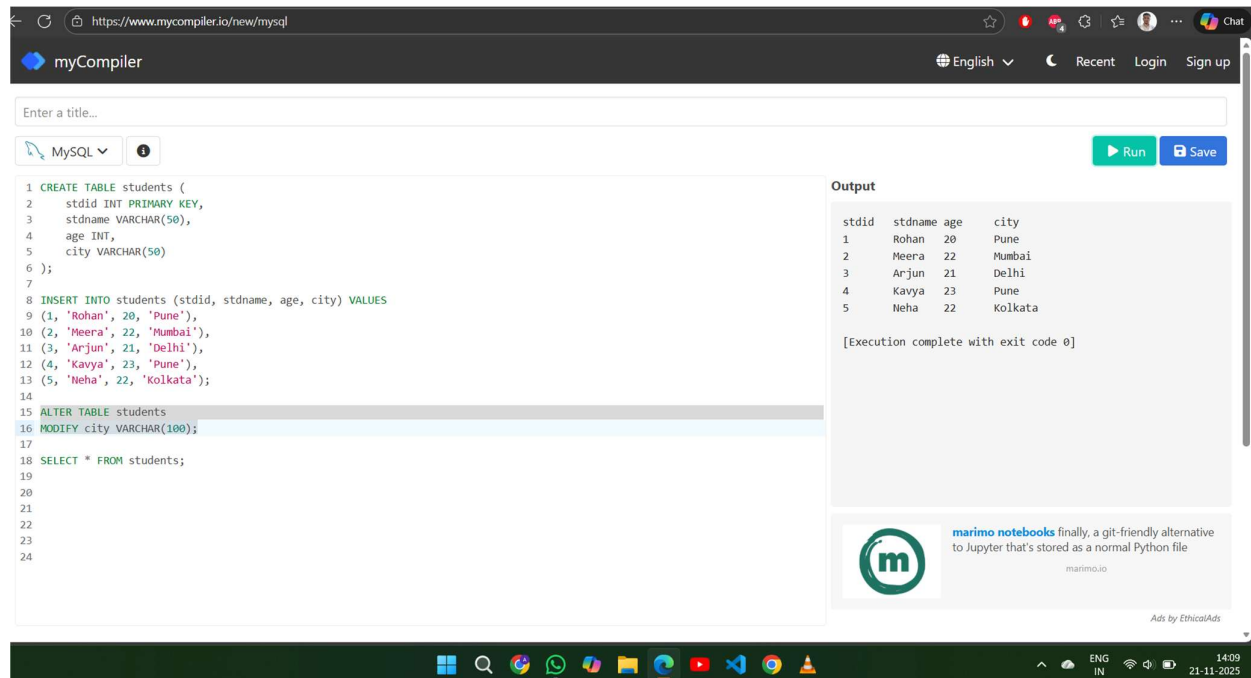
```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 ALTER TABLE students  
16 ADD contact VARCHAR(15);  
17  
18 |  
19  
20  
21  
22  
23
```

The output window displays the result of the ALTER TABLE query:

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
5	Neha	22	Kolkata	NULL

[Execution complete with exit code 0]

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



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

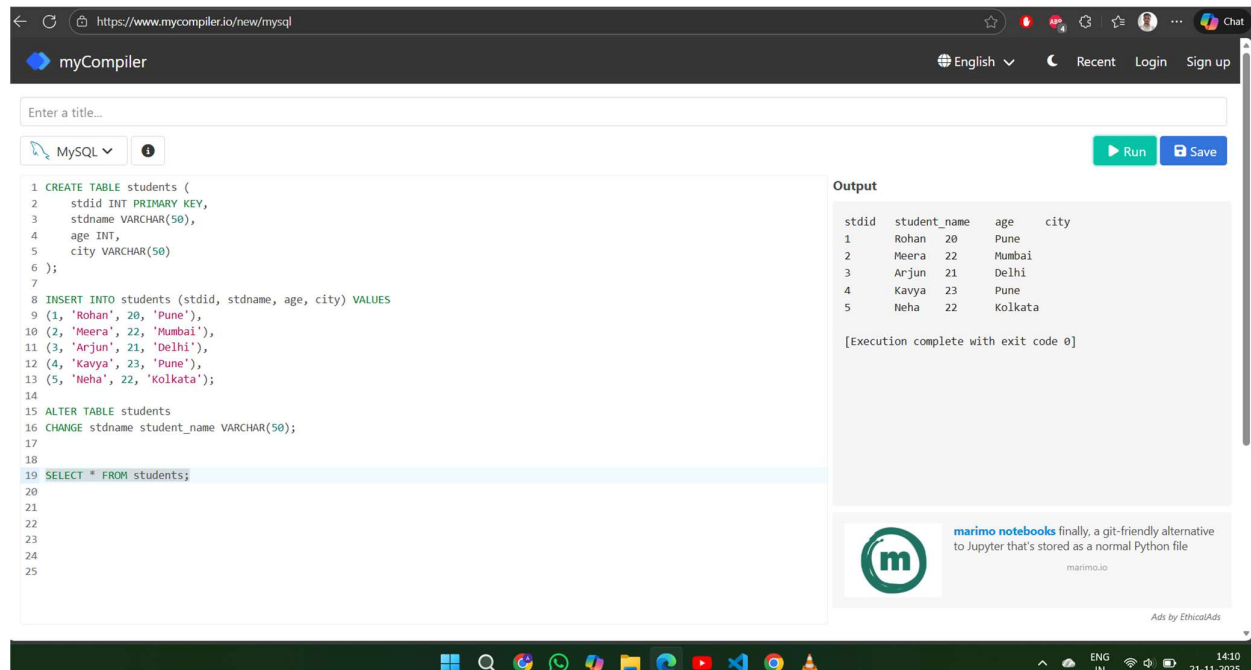
```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 ALTER TABLE students  
16 MODIFY city VARCHAR(100);  
17  
18 SELECT * FROM students;  
19  
20  
21  
22  
23  
24
```

The output section displays the following 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

[Execution complete with exit code 0]

Q13. Rename the column stdname to student_name.



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

```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 ALTER TABLE students  
16 CHANGE stdname student_name VARCHAR(50);  
17  
18  
19 SELECT * FROM students;  
20  
21  
22  
23  
24  
25
```

The output section displays the following table:

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

[Execution complete with exit code 0]

Q14. Drop the column contact from the table.

The screenshot shows the myCompiler.io MySQL interface. The SQL editor contains the following code:

```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 ALTER TABLE students  
16 ADD contact VARCHAR(15);  
17  
18 ALTER TABLE students  
19 DROP COLUMN contact;  
20  
21  
22 SELECT * FROM students;  
23  
24  
25  
26  
27  
28
```

The output window displays the result of the SELECT statement:

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

[Execution complete with exit code 0]

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

The screenshot shows the myCompiler.io MySQL interface. The SQL editor contains the following code:

```
1 CREATE TABLE students (  
2   stdid INT PRIMARY KEY,  
3   stdname VARCHAR(50),  
4   age INT,  
5   city VARCHAR(50)  
6 );  
7  
8 INSERT INTO students (stdid, stdname, age, city) VALUES  
9 (1, 'Rohan', 20, 'Pune'),  
10 (2, 'Meera', 22, 'Mumbai'),  
11 (3, 'Arjun', 21, 'Delhi'),  
12 (4, 'Kavya', 23, 'Pune'),  
13 (5, 'Neha', 22, 'Kolkata');  
14  
15 ALTER TABLE students  
16 ADD contact VARCHAR(15);  
17  
18 ALTER TABLE students  
19 DROP COLUMN contact;  
20  
21 ALTER TABLE students  
22 ADD gender ENUM('M', 'F');  
23  
24  
25  
26 SELECT * FROM students;  
27  
28
```

The output window displays the result of the SELECT statement:

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

[Execution complete with exit code 0]

PART 3 – JOIN PRACTICE

Tables:

Table: students

stdid	student_name	city
1	Rohan	Pune
2	Meera	Mumbai
3	Arjun	Delhi
4	Kavya	Pune

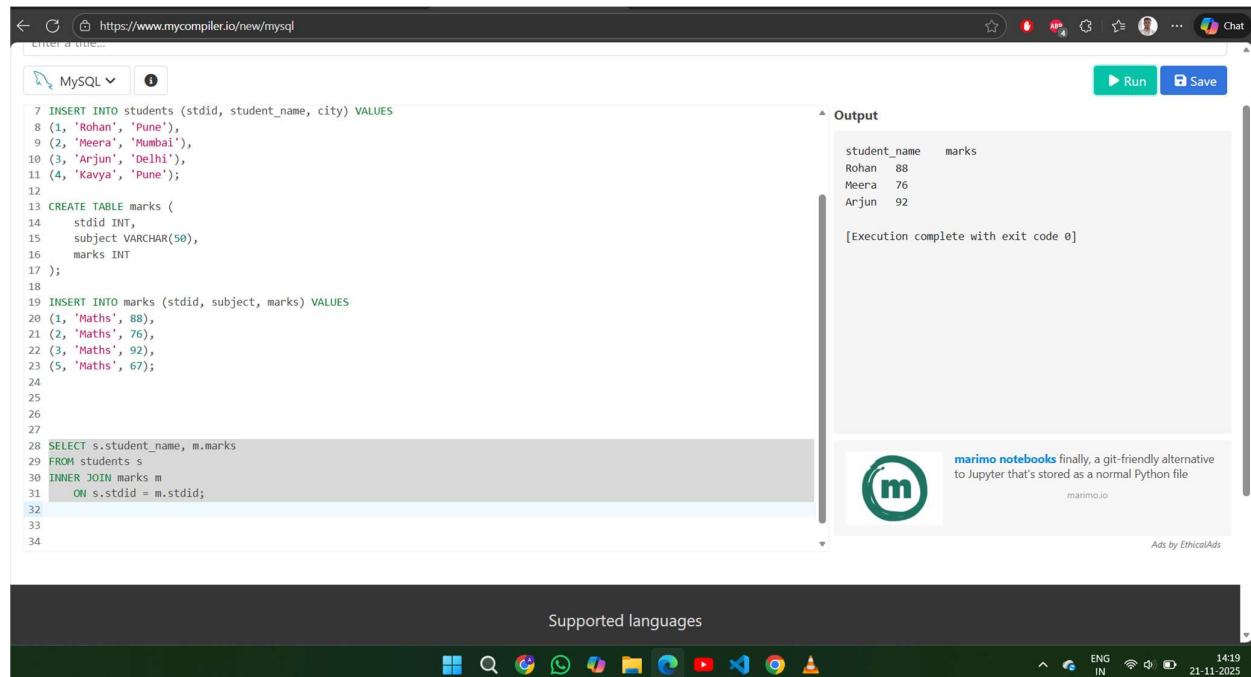
Table: marks

stdid	subject	marks
1	Maths	88
2	Maths	76
3	Maths	92
5	Maths	67

INNER JOIN

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

(Students without marks should not appear.)



The screenshot shows a web-based MySQL compiler interface. The code editor contains the following SQL statements:

```
7 INSERT INTO students (stdid, student_name, city) VALUES
8 (1, 'Rohan', 'Pune'),
9 (2, 'Meera', 'Mumbai'),
10 (3, 'Arjun', 'Delhi'),
11 (4, 'Kavya', 'Pune');
12
13 CREATE TABLE marks (
14   stdid INT,
15   subject VARCHAR(50),
16   marks INT
17 );
18
19 INSERT INTO marks (stdid, subject, marks) VALUES
20 (1, 'Maths', 88),
21 (2, 'Maths', 76),
22 (3, 'Maths', 92),
23 (5, 'Maths', 67);
24
25
26
27
28 SELECT s.student_name, m.marks
29 FROM students s
30 INNER JOIN marks m
31 ON s.stdid = m.stdid;
```

The output panel shows the result of the inner join query:

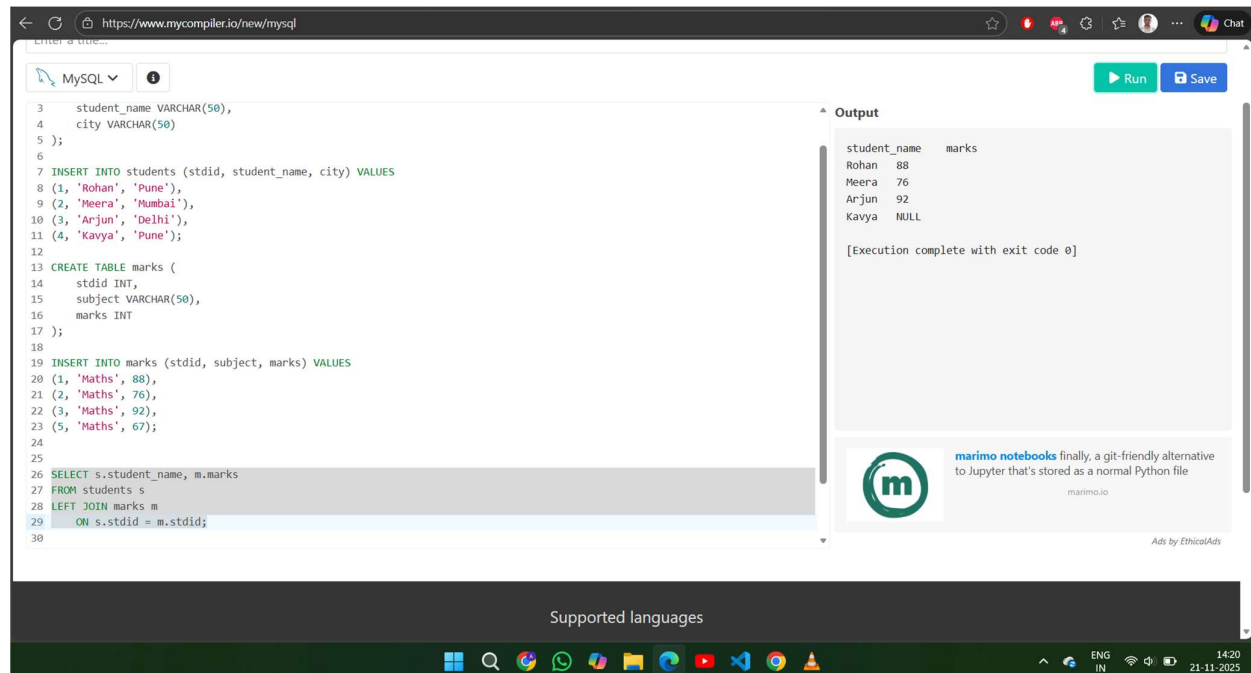
student_name	marks
Rohan	88
Meera	76
Arjun	92

The output also includes the message: "[Execution complete with exit code 0]".

LEFT JOIN

Q17. Display all students and their marks.

(If marks not available, show NULL.)



The screenshot shows a web-based MySQL query editor. The query is as follows:

```
3 student_name VARCHAR(50),
4 city VARCHAR(50)
5 );
6
7 INSERT INTO students (stdid, student_name, city) VALUES
8 (1, 'Rohan', 'Pune'),
9 (2, 'Meera', 'Mumbai'),
10 (3, 'Arjun', 'Delhi'),
11 (4, 'Kavya', 'Pune');
12
13 CREATE TABLE marks (
14 stdid INT,
15 subject VARCHAR(50),
16 marks INT
17 );
18
19 INSERT INTO marks (stdid, subject, marks) VALUES
20 (1, 'Maths', 88),
21 (2, 'Maths', 76),
22 (3, 'Maths', 92),
23 (5, 'Maths', 67);
24
25
26 SELECT s.student_name, m.marks
27 FROM students s
28 LEFT JOIN marks m
29 ON s.stdid = m.stdid;
30
```

The output of the query is displayed in a table:

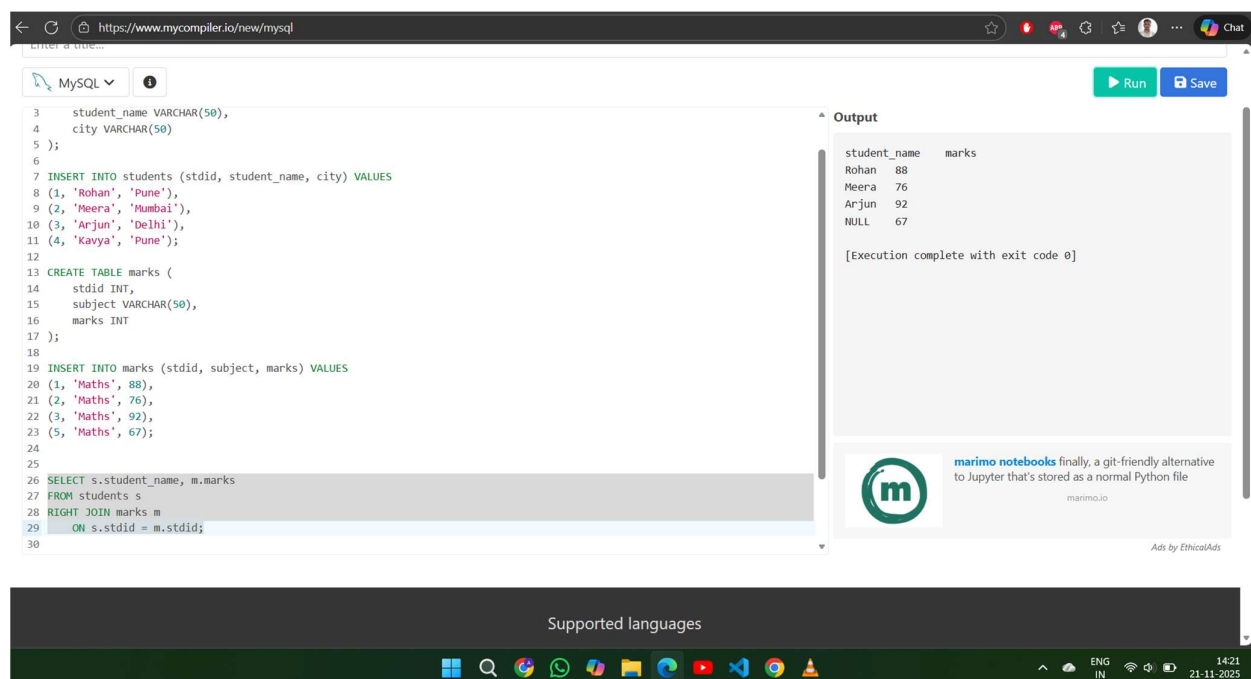
student_name	marks
Rohan	88
Meera	76
Arjun	92
Kavya	NULL

The output also includes the message: [Execution complete with exit code 0].

RIGHT JOIN

Q18. Display all marks records along with student names.

(If student doesn't exist in students table, show NULL.)



The screenshot shows a web-based MySQL query editor. The query is as follows:

```
3 student_name VARCHAR(50),
4 city VARCHAR(50)
5 );
6
7 INSERT INTO students (stdid, student_name, city) VALUES
8 (1, 'Rohan', 'Pune'),
9 (2, 'Meera', 'Mumbai'),
10 (3, 'Arjun', 'Delhi'),
11 (4, 'Kavya', 'Pune');
12
13 CREATE TABLE marks (
14 stdid INT,
15 subject VARCHAR(50),
16 marks INT
17 );
18
19 INSERT INTO marks (stdid, subject, marks) VALUES
20 (1, 'Maths', 88),
21 (2, 'Maths', 76),
22 (3, 'Maths', 92),
23 (5, 'Maths', 67);
24
25
26 SELECT s.student_name, m.marks
27 FROM students s
28 RIGHT JOIN marks m
29 ON s.stdid = m.stdid;
30
```

The output of the query is displayed in a table:

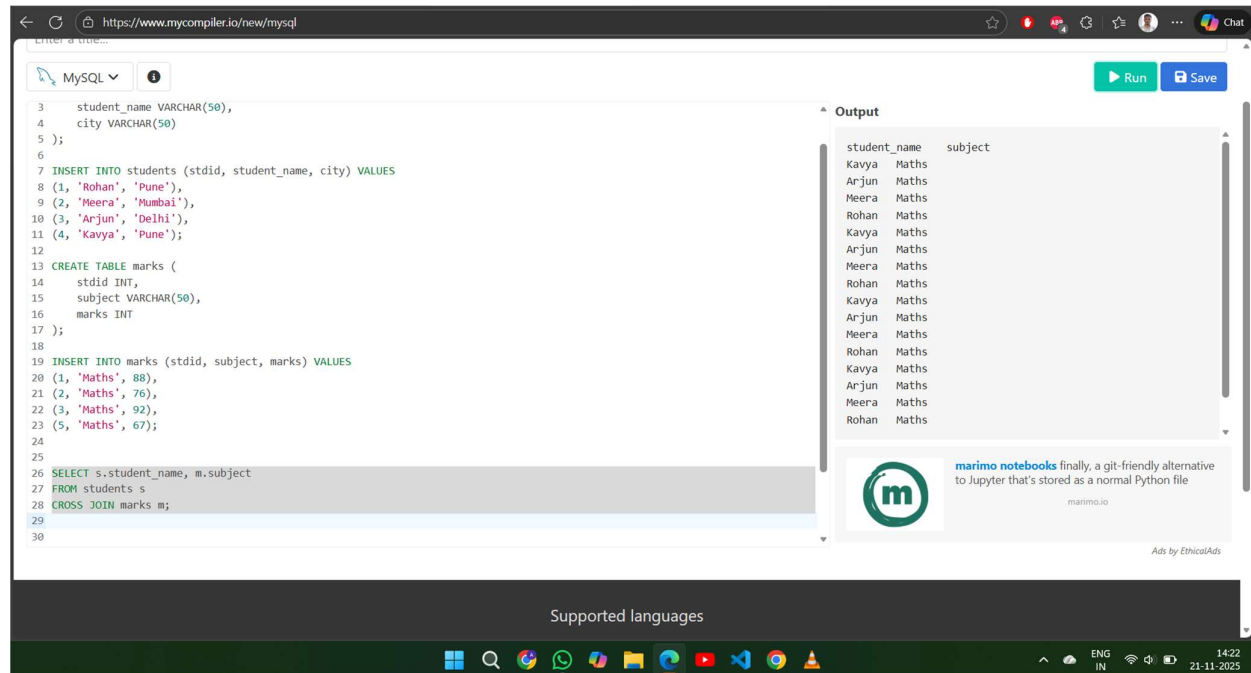
student_name	marks
Rohan	88
Meera	76
Arjun	92
NULL	67

The output also includes the message: [Execution complete with exit code 0].

CROSS JOIN

Q1G. Display all possible combinations of students and subjects.

(Use CROSS JOIN between students and marks table to show every pair.)



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

```
3 student_name VARCHAR(50),
4 city VARCHAR(50)
5 );
6
7 INSERT INTO students (stdid, student_name, city) VALUES
8 (1, 'Rohan', 'Pune'),
9 (2, 'Meera', 'Mumbai'),
10 (3, 'Arjun', 'Delhi'),
11 (4, 'Kavya', 'Pune');
12
13 CREATE TABLE marks (
14 stdid INT,
15 subject VARCHAR(50),
16 marks INT
17 );
18
19 INSERT INTO marks (stdid, subject, marks) VALUES
20 (1, 'Maths', 88),
21 (2, 'Maths', 76),
22 (3, 'Maths', 92),
23 (5, 'Maths', 67);
24
25
26 SELECT s.student_name, m.subject
27 FROM students s
28 CROSS JOIN marks m;
29
30
```

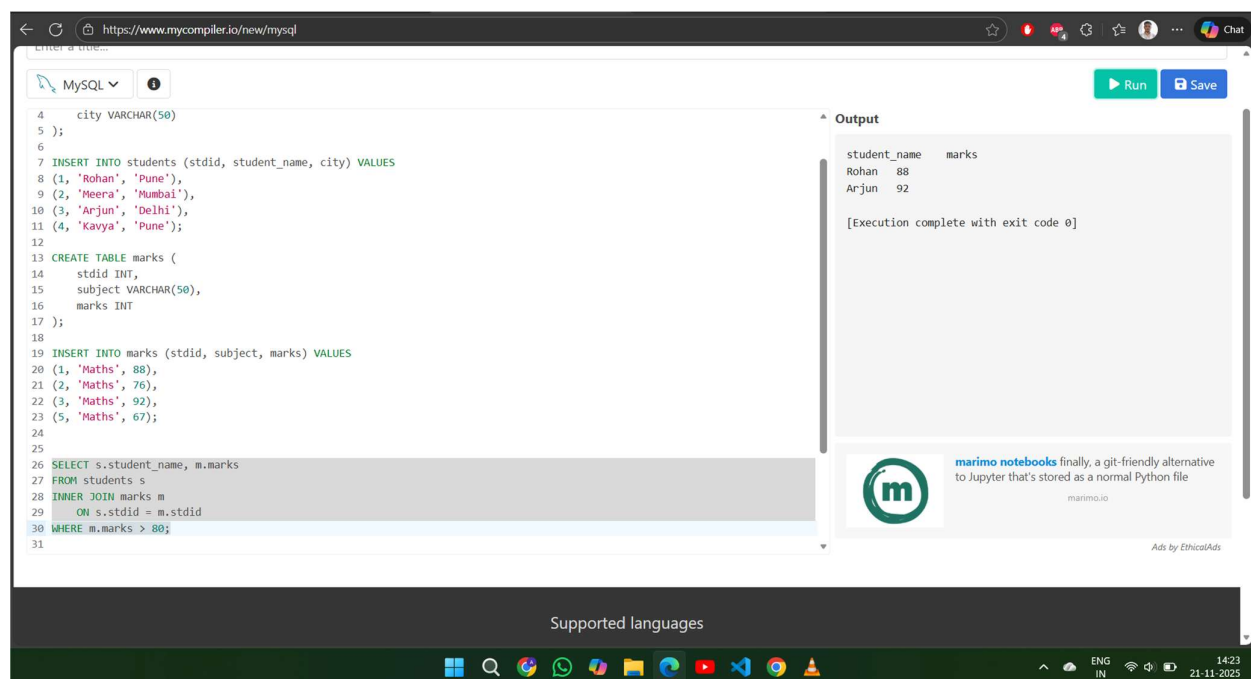
The output shows a table with two columns: student_name and subject. The rows are:

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

The interface also includes a 'Run' button, a 'Save' button, and a 'Supported languages' section at the bottom.

JOIN with Filtering

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



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

```
4 city VARCHAR(50)
5 );
6
7 INSERT INTO students (stdid, student_name, city) VALUES
8 (1, 'Rohan', 'Pune'),
9 (2, 'Meera', 'Mumbai'),
10 (3, 'Arjun', 'Delhi'),
11 (4, 'Kavya', 'Pune');
12
13 CREATE TABLE marks (
14 stdid INT,
15 subject VARCHAR(50),
16 marks INT
17 );
18
19 INSERT INTO marks (stdid, subject, marks) VALUES
20 (1, 'Maths', 88),
21 (2, 'Maths', 76),
22 (3, 'Maths', 92),
23 (5, 'Maths', 67);
24
25
26 SELECT s.student_name, m.marks
27 FROM students s
28 INNER JOIN marks m
29 ON s.stdid = m.stdid
30 WHERE m.marks > 80;
31
```

The output shows a table with two columns: student_name and marks. The rows are:

student_name	marks
Rohan	88
Arjun	92

The interface also includes a 'Run' button, a 'Save' button, and a 'Supported languages' section at the bottom.