NAME-Jayant Jhunja

CODING CHALLENGE

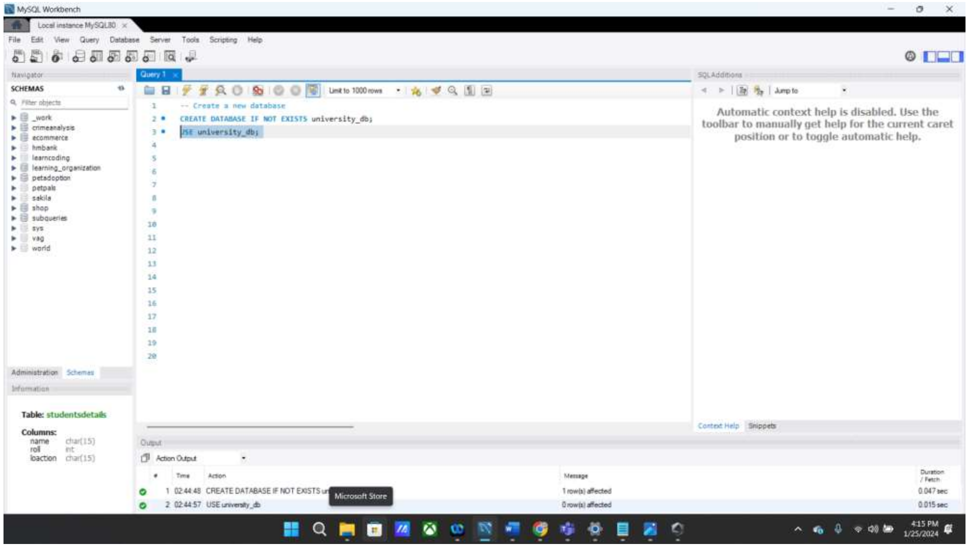
SQL

JOINS

*In relational databases, a join is an operation that combines rows from two or more tables based on a related column between them. The purpose of a join is to retrieve and consolidate information from multiple tables into a single result set.

TYPES OF JOINS --

*INNER JOIN-



```
-- Create a new database
CREATE DATABASE IF NOT EXISTS university_db;
USE university_db;
```

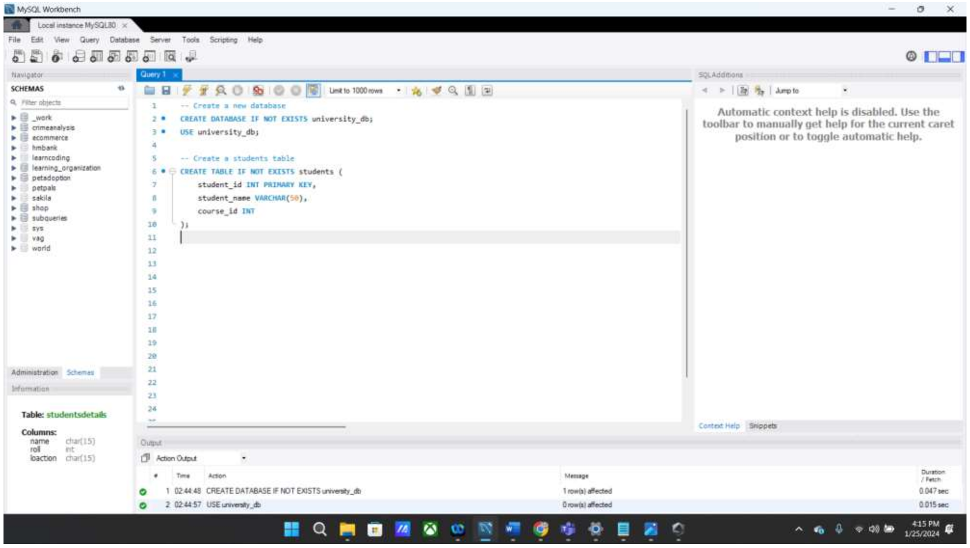
Table: studentsdetails

Columns:

name	char(15)
roll	int
loaction	char(15)

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
#  Time      Action                               Message              Duration / Fetch
1  02:44:48  CREATE DATABASE IF NOT EXISTS un     1 row(s) affected    0.047 sec
2  02:44:57  USE university_db                    0 row(s) affected    0.015 sec
```



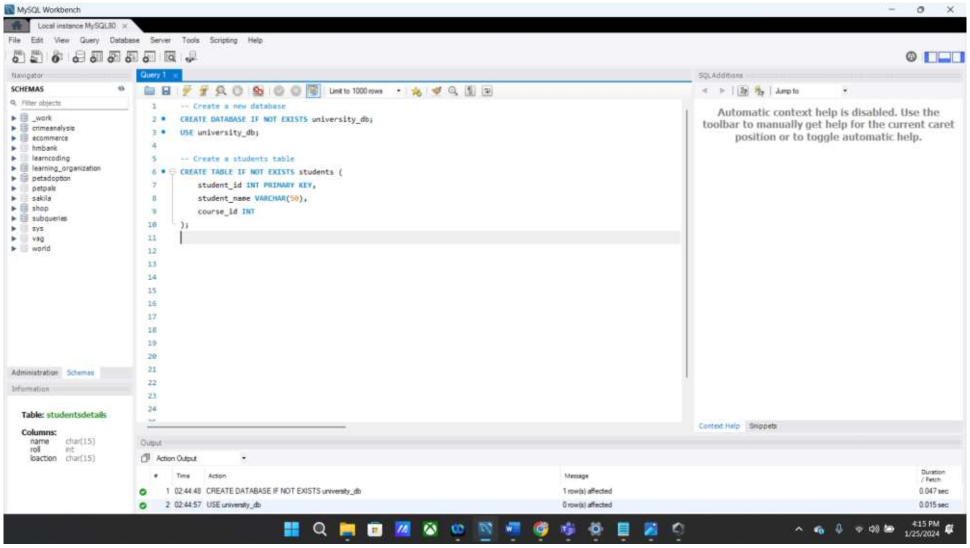
```sql
-- Create a new database
CREATE DATABASE IF NOT EXISTS university_db;
USE university_db;

-- Create a students table
CREATE TABLE IF NOT EXISTS students (
    student_id INT PRIMARY KEY,
    student_name VARCHAR(50),
    course_id INT
);
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: studentsdetails

Columns:
name char(15)
roll int
loaction char(15)

```
#  Time      Action                                          Message             Duration / Fetch
1  02:44:48  CREATE DATABASE IF NOT EXISTS university_db     1 row(s) affected   0.047 sec
2  02:44:57  USE university_db                               0 row(s) affected   0.015 sec
```



MySQL Workbench
Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
_work
crimeanalysis
ecommerce
hmbank
learncoding
learning_organization
petadoption
petpals
sakila
shop
subqueries
sys
vag
world
Query 1
Limit to 1000 rows
SQLAdditions
Jump to
Administration
Schemas
Information
Context Help
Snippets
Output
Action Output
4:15 PM
1/25/2024

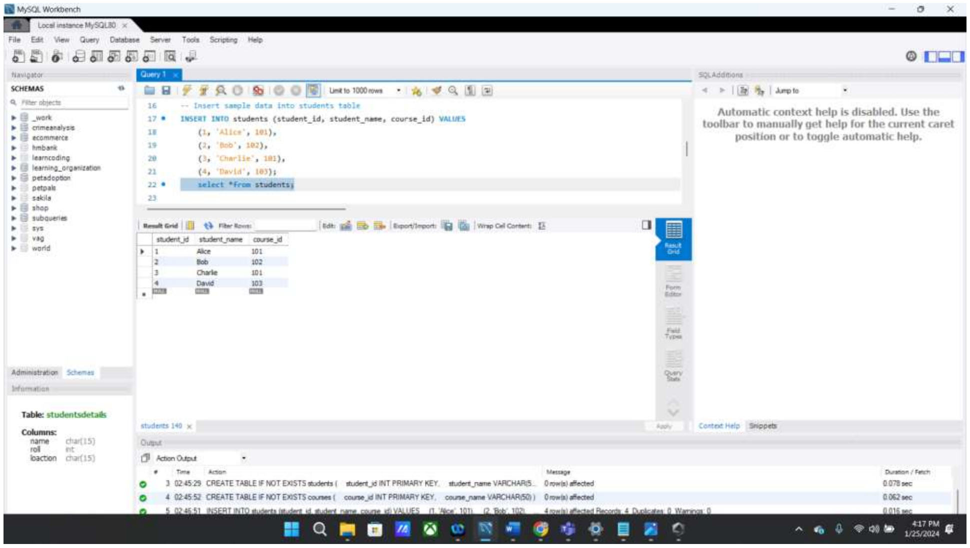
```sql
-- Create a new database
CREATE DATABASE IF NOT EXISTS university_db;
USE university_db;

-- Create a students table
CREATE TABLE IF NOT EXISTS students (
    student_id INT PRIMARY KEY,
    student_name VARCHAR(50),
    course_id INT
);
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: studentsdetails

Columns:
name char(15)
roll int
loaction char(15)

```
#  Time      Action                                          Message             Duration / Fetch
1  02:44:48  CREATE DATABASE IF NOT EXISTS university_db     1 row(s) affected   0.047 sec
2  02:44:57  USE university_db                               0 row(s) affected   0.015 sec
```



```sql
-- Insert sample data into students table
INSERT INTO students (student_id, student_name, course_id) VALUES
    (1, 'Alice', 101),
    (2, 'Bob', 102),
    (3, 'Charlie', 101),
    (4, 'David', 103);
    select *from students;
```

student_id	student_name	course_id
1	Alice	101
2	Bob	102
3	Charlie	101
4	David	103
NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: studentsdetails

Columns:
name char(15)
roll int
loaction char(15)

```
#  Time      Action                                                                                   Message                                                  Duration / Fetch
3  02:45:29  CREATE TABLE IF NOT EXISTS students (   student_id INT PRIMARY KEY,   student_name VARCHAR(5...   0 row(s) affected                                        0.078 sec
4  02:45:52  CREATE TABLE IF NOT EXISTS courses (   course_id INT PRIMARY KEY,   course_name VARCHAR(50) )   0 row(s) affected                                        0.062 sec
5  02:46:51  INSERT INTO students (student_id, student_name, course_id) VALUES   (1, 'Alice', 101),   (2, 'Bob', 102), ...   4 row(s) affected Records: 4  Duplicates: 0  Warnings: 0   0.016 sec
```

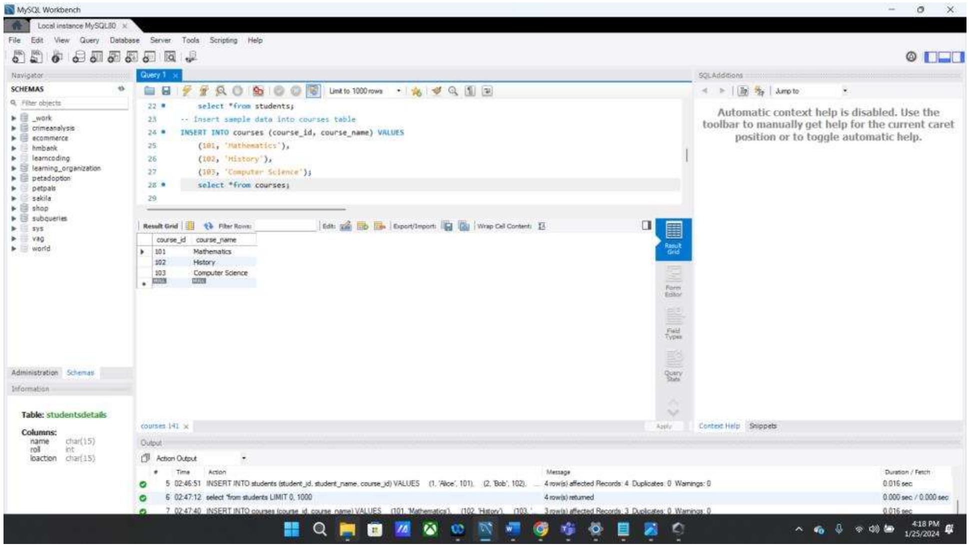


Table: studentsdetails

Columns:

name	char(15)
roll	int
loaction	char(15)

```sql
    select *from students;
-- Insert sample data into courses table
INSERT INTO courses (course_id, course_name) VALUES
    (101, 'Mathematics'),
    (102, 'History'),
    (103, 'Computer Science');
    select *from courses;
```

course_id	course_name
101	Mathematics
102	History
103	Computer Science
NULL	NULL

```
#  Time      Action                                                                                   Message                                                          Duration / Fetch
5  02:46:51  INSERT INTO students (student_id, student_name, course_id) VALUES   (1, 'Alice', 101),   (2, 'Bob', 102),   ...   4 row(s) affected Records: 4  Duplicates: 0  Warnings: 0   0.016 sec
6  02:47:12  select *from students LIMIT 0, 1000                                                      4 row(s) returned                                                0.000 sec / 0.000 sec
7  02:47:40  INSERT INTO courses (course_id, course_name) VALUES   (101, 'Mathematics'),   (102, 'History'),   (103, '...   3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0   0.016 sec
```

4:18 PM
1/25/2024

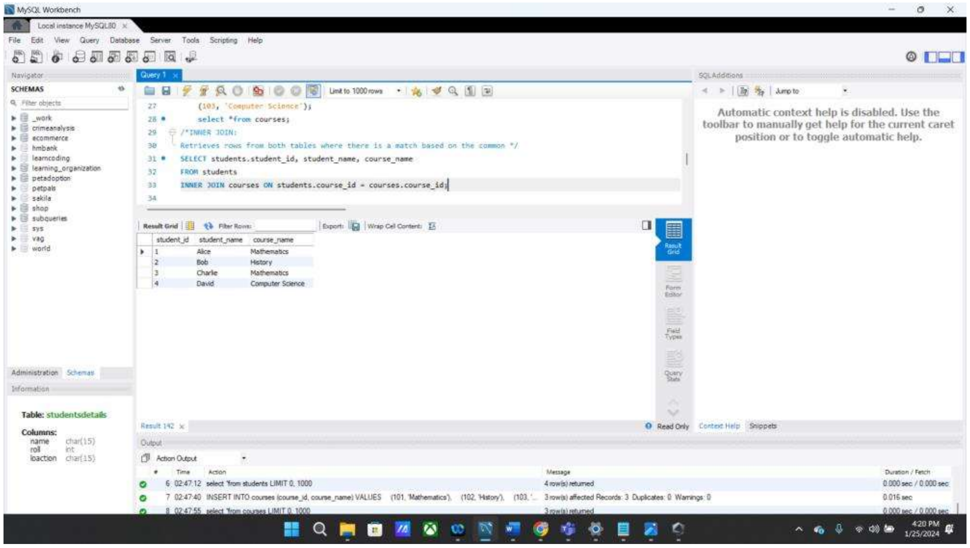


Table: studentsdetails

Columns:

name	char(15)
roll	int
loaction	char(15)

```sql
    (103, 'Computer Science');
    select *from courses;
/*INNER JOIN:
Retrieves rows from both tables where there is a match based on the common */
SELECT students.student_id, student_name, course_name
FROM students
INNER JOIN courses ON students.course_id = courses.course_id;
```

student_id	student_name	course_name
1	Alice	Mathematics
2	Bob	History
3	Charlie	Mathematics
4	David	Computer Science

```
#  Time      Action                                                                                   Message                                                          Duration / Fetch
6  02:47:12  select *from students LIMIT 0, 1000                                                      4 row(s) returned                                                0.000 sec / 0.000 sec
7  02:47:40  INSERT INTO courses (course_id, course_name) VALUES   (101, 'Mathematics'),   (102, 'History'),   (103, '...   3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0   0.016 sec
8  02:47:55  select *from courses LIMIT 0, 1000                                                       3 row(s) returned                                                0.000 sec / 0.000 sec
```

4:20 PM
1/25/2024

*LEFT JOIN--

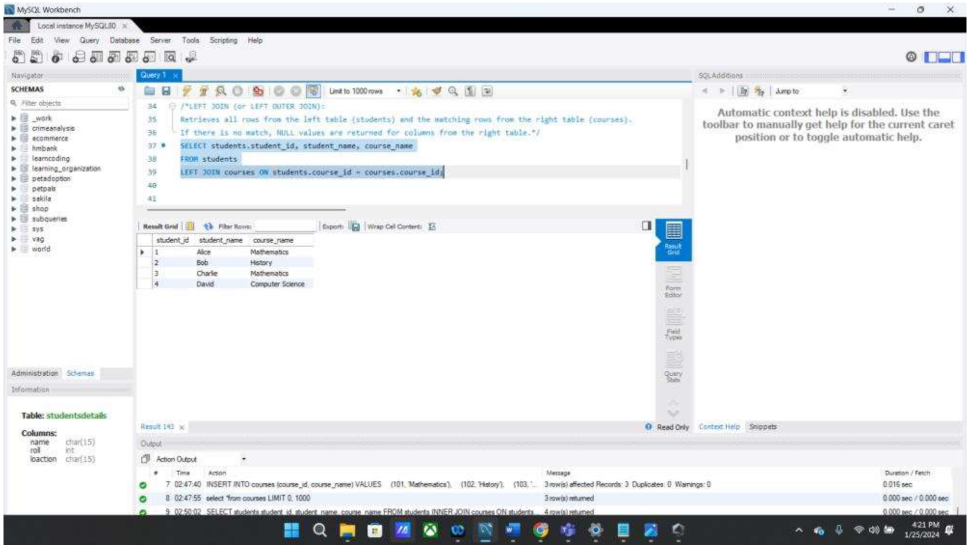


Table: studentsdetails

Columns:

name	char(15)
roll	int
loaction	char(15)

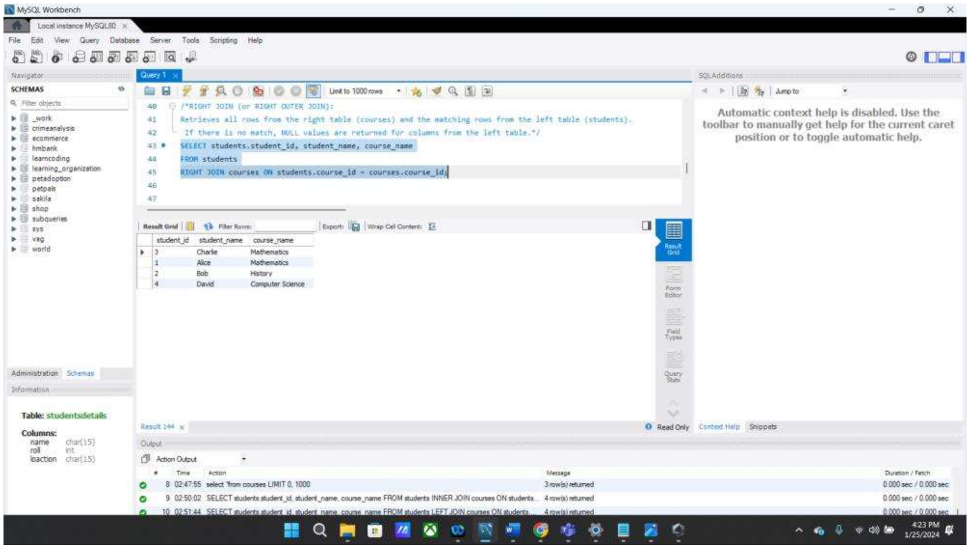
```sql
/*LEFT JOIN (or LEFT OUTER JOIN):
Retrieves all rows from the left table (students) and the matching rows from the right table (courses).
If there is no match, NULL values are returned for columns from the right table.*/
SELECT students.student_id, student_name, course_name
FROM students
LEFT JOIN courses ON students.course_id = courses.course_id;
```

student_id	student_name	course_name
1	Alice	Mathematics
2	Bob	History
3	Charlie	Mathematics
4	David	Computer Science

```
#  Time      Action                                                                                   Message                                                          Duration / Fetch
7  02:47:40  INSERT INTO courses (course_id, course_name) VALUES   (101, 'Mathematics'),   (102, 'History'),   (103, '...   3 row(s) affected Records: 3  Duplicates: 0  Warnings: 0   0.016 sec
8  02:47:55  select *from courses LIMIT 0, 1000                                                       3 row(s) returned                                                0.000 sec / 0.000 sec
9  02:50:02  SELECT students.student_id, student_name, course_name FROM students INNER JOIN courses ON students...   4 row(s) returned                                   0.000 sec / 0.000 sec
```

4:21 PM
1/25/2024

*RIGHT JOIN--



```
/*RIGHT JOIN (or RIGHT OUTER JOIN):
Retrieves all rows from the right table (courses) and the matching rows from the left table (students).
 If there is no match, NULL values are returned for columns from the left table.*/
SELECT students.student_id, student_name, course_name
FROM students
RIGHT JOIN courses ON students.course_id = courses.course_id;
```

student_id	student_name	course_name
3	Charlie	Mathematics
1	Alice	Mathematics
2	Bob	History
4	David	Computer Science

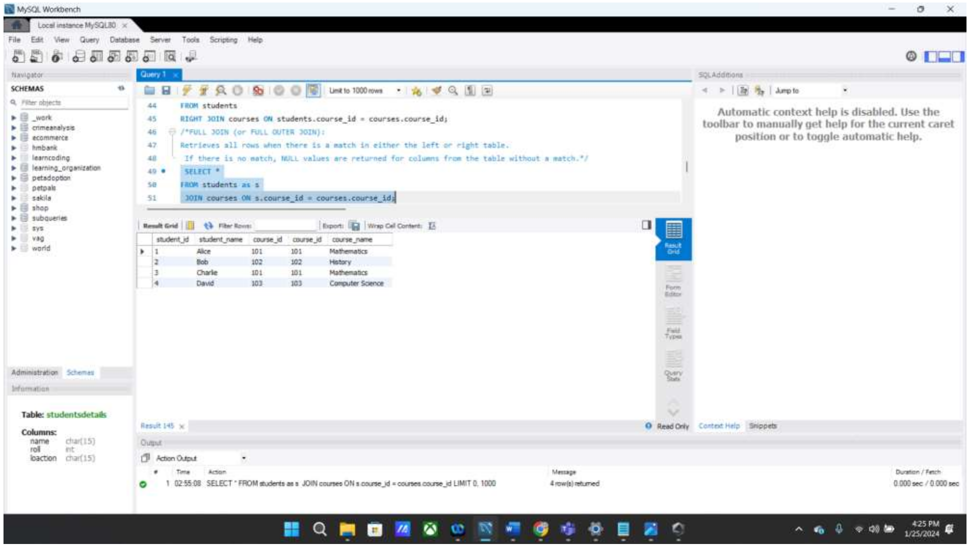
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: studentsdetails

Columns:
name char(15)
roll int
loaction char(15)

```
#   Time      Action                                                                              Message            Duration / Fetch
8   02:47:55  select *from courses LIMIT 0, 1000                                                  3 row(s) returned  0.000 sec / 0.000 sec
9   02:50:02  SELECT students.student_id, student_name, course_name FROM students INNER JOIN courses ON students...  4 row(s) returned  0.000 sec / 0.000 sec
10  02:51:44  SELECT students.student_id, student_name, course_name FROM students LEFT JOIN courses ON students...   4 row(s) returned  0.000 sec / 0.000 sec
```

*FULL JOIN--



```
FROM students
RIGHT JOIN courses ON students.course_id = courses.course_id;
/*FULL JOIN (or FULL OUTER JOIN):
Retrieves all rows when there is a match in either the left or right table.
 If there is no match, NULL values are returned for columns from the table without a match.*/
 SELECT *
FROM students as s
 JOIN courses ON s.course_id = courses.course_id;
```

student_id	student_name	course_id	course_id	course_name
1	Alice	101	101	Mathematics
2	Bob	102	102	History
3	Charlie	101	101	Mathematics
4	David	103	103	Computer Science

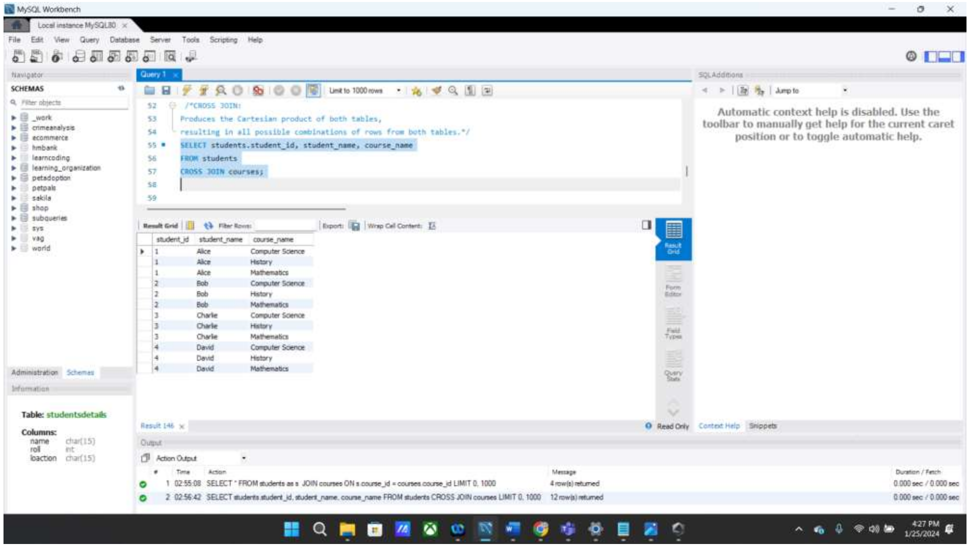
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: studentsdetails

Columns:
name char(15)
roll int
loaction char(15)

```
#  Time      Action                                                                        Message            Duration / Fetch
1  02:55:08  SELECT * FROM students as s  JOIN courses ON s.course_id = courses.course_id LIMIT 0, 1000  4 row(s) returned  0.000 sec / 0.000 sec
```

*CROSS JOIN--



```
/*CROSS JOIN:
Produces the Cartesian product of both tables,
resulting in all possible combinations of rows from both tables.*/
SELECT students.student_id, student_name, course_name
FROM students
CROSS JOIN courses;
```

student_id	student_name	course_name
1	Alice	Computer Science
1	Alice	History
1	Alice	Mathematics
2	Bob	Computer Science
2	Bob	History
2	Bob	Mathematics
3	Charlie	Computer Science
3	Charlie	History
3	Charlie	Mathematics
4	David	Computer Science
4	David	History
4	David	Mathematics

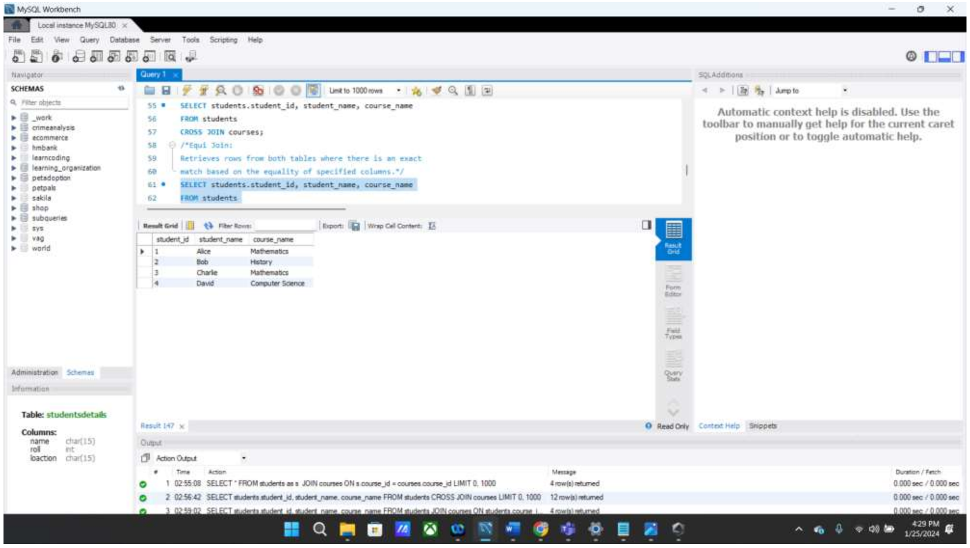
Table: studentsdetails

Columns:
name char(15)
roll int
loaction char(15)

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
#  Time      Action                                                                                            Message             Duration / Fetch
1  02:55:08  SELECT * FROM students as s  JOIN courses ON s.course_id = courses.course_id LIMIT 0, 1000          4 row(s) returned   0.000 sec / 0.000 sec
2  02:56:42  SELECT students.student_id, student_name, course_name FROM students CROSS JOIN courses LIMIT 0, 1000  12 row(s) returned  0.000 sec / 0.000 sec
```

*EQUI JOIN--



```
SELECT students.student_id, student_name, course_name
FROM students
CROSS JOIN courses;
/*Equi Join:
Retrieves rows from both tables where there is an exact
match based on the equality of specified columns.*/
SELECT students.student_id, student_name, course_name
FROM students
```

student_id	student_name	course_name
1	Alice	Mathematics
2	Bob	History
3	Charlie	Mathematics
4	David	Computer Science

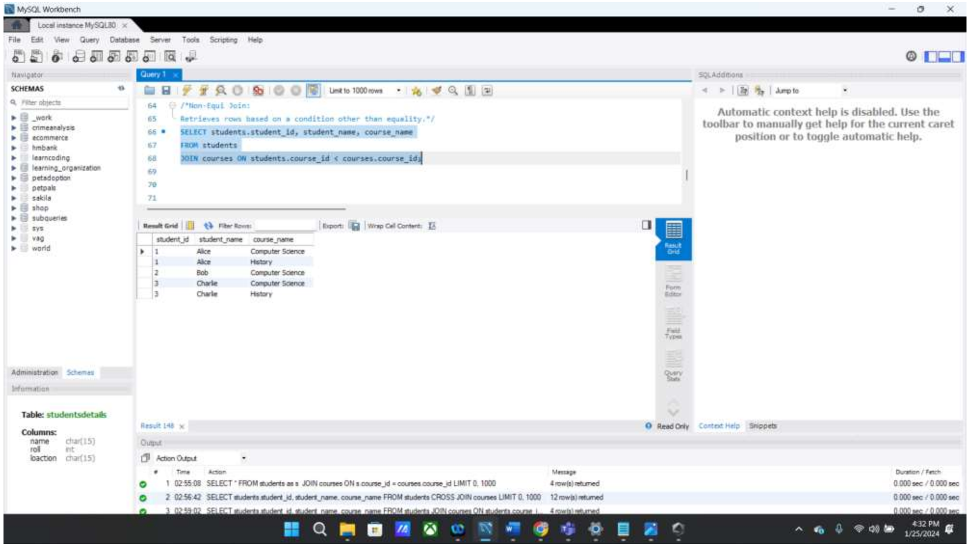
Table: studentsdetails

Columns:
name char(15)
roll int
loaction char(15)

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
#  Time      Action                                                                                            Message             Duration / Fetch
1  02:55:08  SELECT * FROM students as s  JOIN courses ON s.course_id = courses.course_id LIMIT 0, 1000          4 row(s) returned   0.000 sec / 0.000 sec
2  02:56:42  SELECT students.student_id, student_name, course_name FROM students CROSS JOIN courses LIMIT 0, 1000  12 row(s) returned  0.000 sec / 0.000 sec
3  02:59:02  SELECT students.student_id, student_name, course_name FROM students JOIN courses ON students.course_i...  4 row(s) returned   0.000 sec / 0.000 sec
```

*NON EQUI JOIN--



```sql
/*Non-Equi Join:
Retrieves rows based on a condition other than equality.*/
SELECT students.student_id, student_name, course_name
FROM students
JOIN courses ON students.course_id < courses.course_id;
```

student_id	student_name	course_name
1	Alice	Computer Science
1	Alice	History
2	Bob	Computer Science
3	Charlie	Computer Science
3	Charlie	History

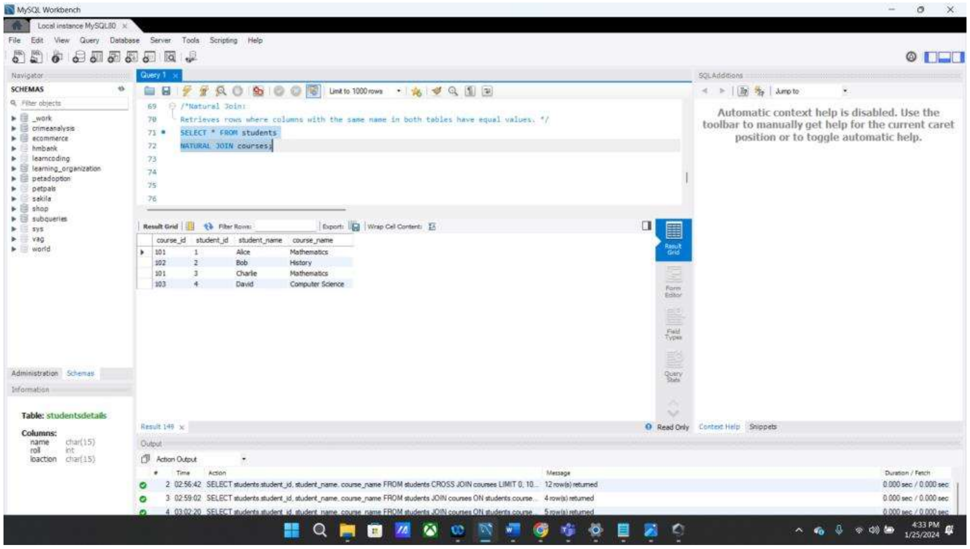
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: studentsdetails

Columns:
name char(15)
roll int
loaction char(15)

```
#  Time      Action                                                                                   Message            Duration / Fetch
1  02:55:08  SELECT * FROM students as s  JOIN courses ON s.course_id = courses.course_id LIMIT 0, 1000   4 row(s) returned  0.000 sec / 0.000 sec
2  02:56:42  SELECT students.student_id, student_name, course_name FROM students CROSS JOIN courses LIMIT 0, 1000   12 row(s) returned  0.000 sec / 0.000 sec
3  02:59:02  SELECT students.student_id, student_name, course_name FROM students JOIN courses ON students.course_i...   4 row(s) returned  0.000 sec / 0.000 sec
```

*NATURAL JOIN--



```sql
/*Natural Join:
Retrieves rows where columns with the same name in both tables have equal values. */
SELECT * FROM students
NATURAL JOIN courses;
```

course_id	student_id	student_name	course_name
101	1	Alice	Mathematics
102	2	Bob	History
101	3	Charlie	Mathematics
103	4	David	Computer Science

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: studentsdetails

Columns:
name char(15)
roll int
loaction char(15)

```
#  Time      Action                                                                                   Message            Duration / Fetch
2  02:56:42  SELECT students.student_id, student_name, course_name FROM students CROSS JOIN courses LIMIT 0, 10...   12 row(s) returned  0.000 sec / 0.000 sec
3  02:59:02  SELECT students.student_id, student_name, course_name FROM students JOIN courses ON students.course...   4 row(s) returned  0.000 sec / 0.000 sec
4  03:02:20  SELECT students.student_id, student_name, course_name FROM students JOIN courses ON students.course...   5 row(s) returned  0.000 sec / 0.000 sec
```