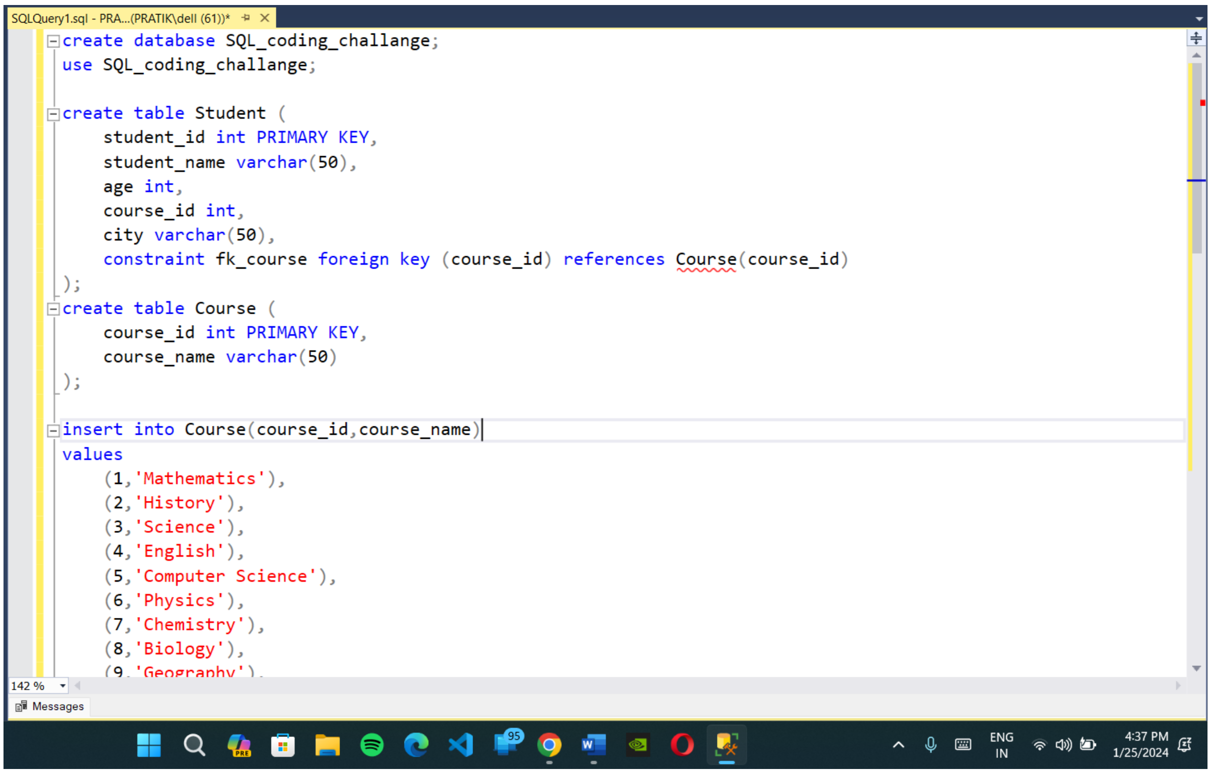
SQL Coding Challenge 1(Joins)

- Database Information:
    - Created Database name as SQL_coding_challange
    - Created two tables: Student and Course

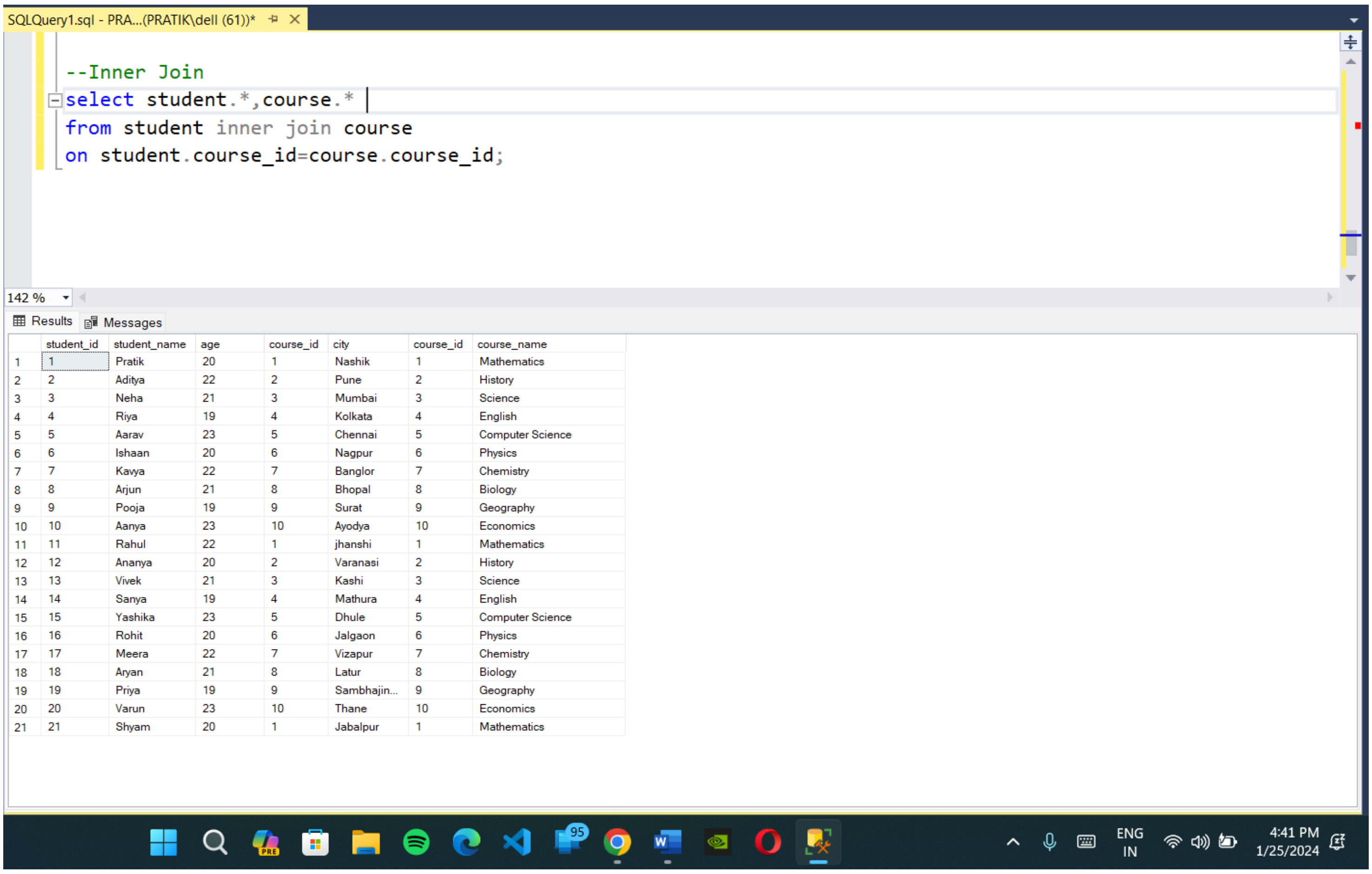


```
create database SQL_coding_challange;
use SQL_coding_challange;

create table Student (
    student_id int PRIMARY KEY,
    student_name varchar(50),
    age int,
    course_id int,
    city varchar(50),
    constraint fk_course foreign key (course_id) references Course(course_id)
);
create table Course (
    course_id int PRIMARY KEY,
    course_name varchar(50)
);

insert into Course(course_id,course_name)
values
    (1,'Mathematics'),
    (2,'History'),
    (3,'Science'),
    (4,'English'),
    (5,'Computer Science'),
    (6,'Physics'),
    (7,'Chemistry'),
    (8,'Biology'),
    (9,'Geography'),
```

- Joins:
    - joins are used to combine rows from two or more tables based on a related column between them.
    - Here the condition is that we need at least one common column through which we can join the tables.

- Types of Joins:

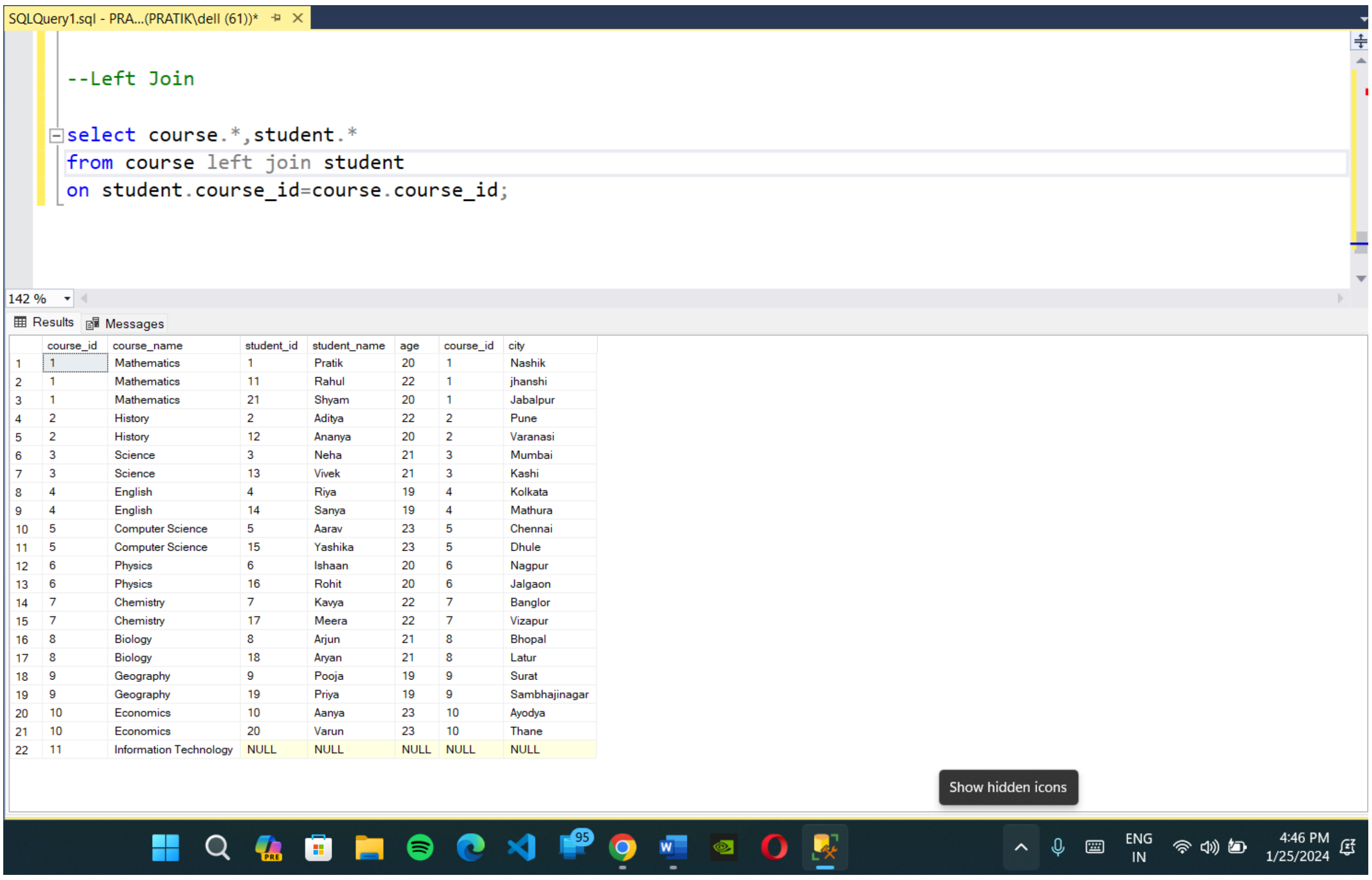
  - <u>Inner Join:</u>
    - The INNER JOIN keyword selects records that have matching values in both tables.
    - Here Both tables join based on course id.
    - We will get the information of all students along with there course details.



```sql
--Inner Join
select student.*,course.*
from student inner join course
on student.course_id=course.course_id;
```

	student_id	student_name	age	course_id	city	course_id	course_name
1	1	Pratik	20	1	Nashik	1	Mathematics
2	2	Aditya	22	2	Pune	2	History
3	3	Neha	21	3	Mumbai	3	Science
4	4	Riya	19	4	Kolkata	4	English
5	5	Aarav	23	5	Chennai	5	Computer Science
6	6	Ishaan	20	6	Nagpur	6	Physics
7	7	Kavya	22	7	Banglor	7	Chemistry
8	8	Arjun	21	8	Bhopal	8	Biology
9	9	Pooja	19	9	Surat	9	Geography
10	10	Aanya	23	10	Ayodya	10	Economics
11	11	Rahul	22	1	jhanshi	1	Mathematics
12	12	Ananya	20	2	Varanasi	2	History
13	13	Vivek	21	3	Kashi	3	Science
14	14	Sanya	19	4	Mathura	4	English
15	15	Yashika	23	5	Dhule	5	Computer Science
16	16	Rohit	20	6	Jalgaon	6	Physics
17	17	Meera	22	7	Vizapur	7	Chemistry
18	18	Aryan	21	8	Latur	8	Biology
19	19	Priya	19	9	Sambhajin...	9	Geography
20	20	Varun	23	10	Thane	10	Economics
21	21	Shyam	20	1	Jabalpur	1	Mathematics

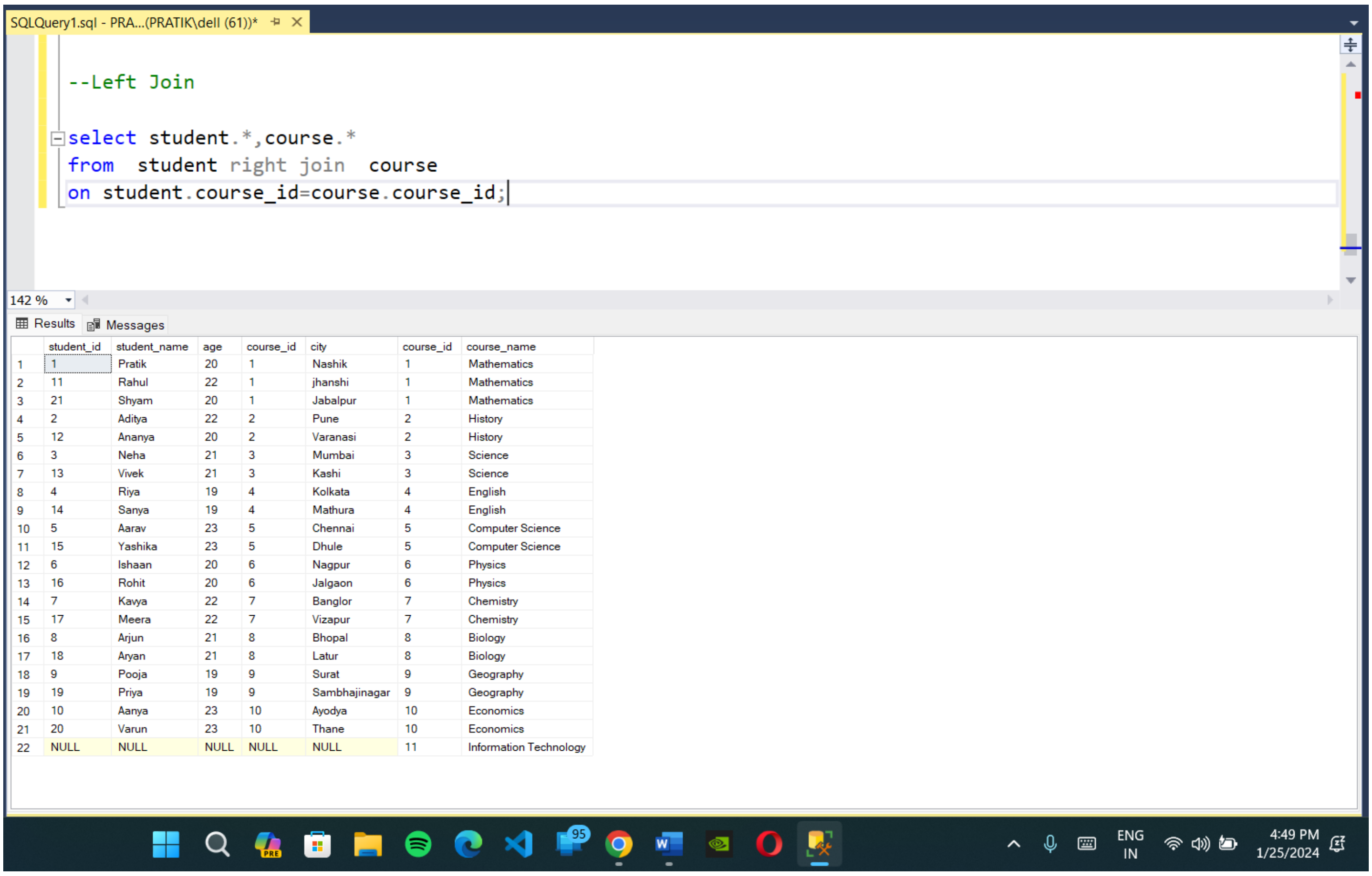
- <u>Left Join:</u>
    - The LEFT JOIN returns all records from the left table and the matched records from the right table.
    - If there is no match, NULL values are returned for columns from the right table.
    - Also known as LEFT OUTER JOIN
    - Here all the records from the left table(course) is returned and if there is no matching record in right table(student) it will return null in right table



```
--Left Join

select course.*,student.*
from course left join student
on student.course_id=course.course_id;
```

	course_id	course_name	student_id	student_name	age	course_id	city
1	1	Mathematics	1	Pratik	20	1	Nashik
2	1	Mathematics	11	Rahul	22	1	jhanshi
3	1	Mathematics	21	Shyam	20	1	Jabalpur
4	2	History	2	Aditya	22	2	Pune
5	2	History	12	Ananya	20	2	Varanasi
6	3	Science	3	Neha	21	3	Mumbai
7	3	Science	13	Vivek	21	3	Kashi
8	4	English	4	Riya	19	4	Kolkata
9	4	English	14	Sanya	19	4	Mathura
10	5	Computer Science	5	Aarav	23	5	Chennai
11	5	Computer Science	15	Yashika	23	5	Dhule
12	6	Physics	6	Ishaan	20	6	Nagpur
13	6	Physics	16	Rohit	20	6	Jalgaon
14	7	Chemistry	7	Kavya	22	7	Banglor
15	7	Chemistry	17	Meera	22	7	Vizapur
16	8	Biology	8	Arjun	21	8	Bhopal
17	8	Biology	18	Aryan	21	8	Latur
18	9	Geography	9	Pooja	19	9	Surat
19	9	Geography	19	Priya	19	9	Sambhajinagar
20	10	Economics	10	Aanya	23	10	Ayodya
21	10	Economics	20	Varun	23	10	Thane
22	11	Information Technology	NULL	NULL	NULL	NULL	NULL

- <u>Right Join:</u>
    - The Right JOIN returns all records from the right table and the matched records from the left table.
    - If there is no match, NULL values are returned for columns from the left table.
    - Also known as RIGHT OUTER JOIN
    - Here all the records from the right table(course) is returned and if there is no matching record in left table(student) it will return null in left table.

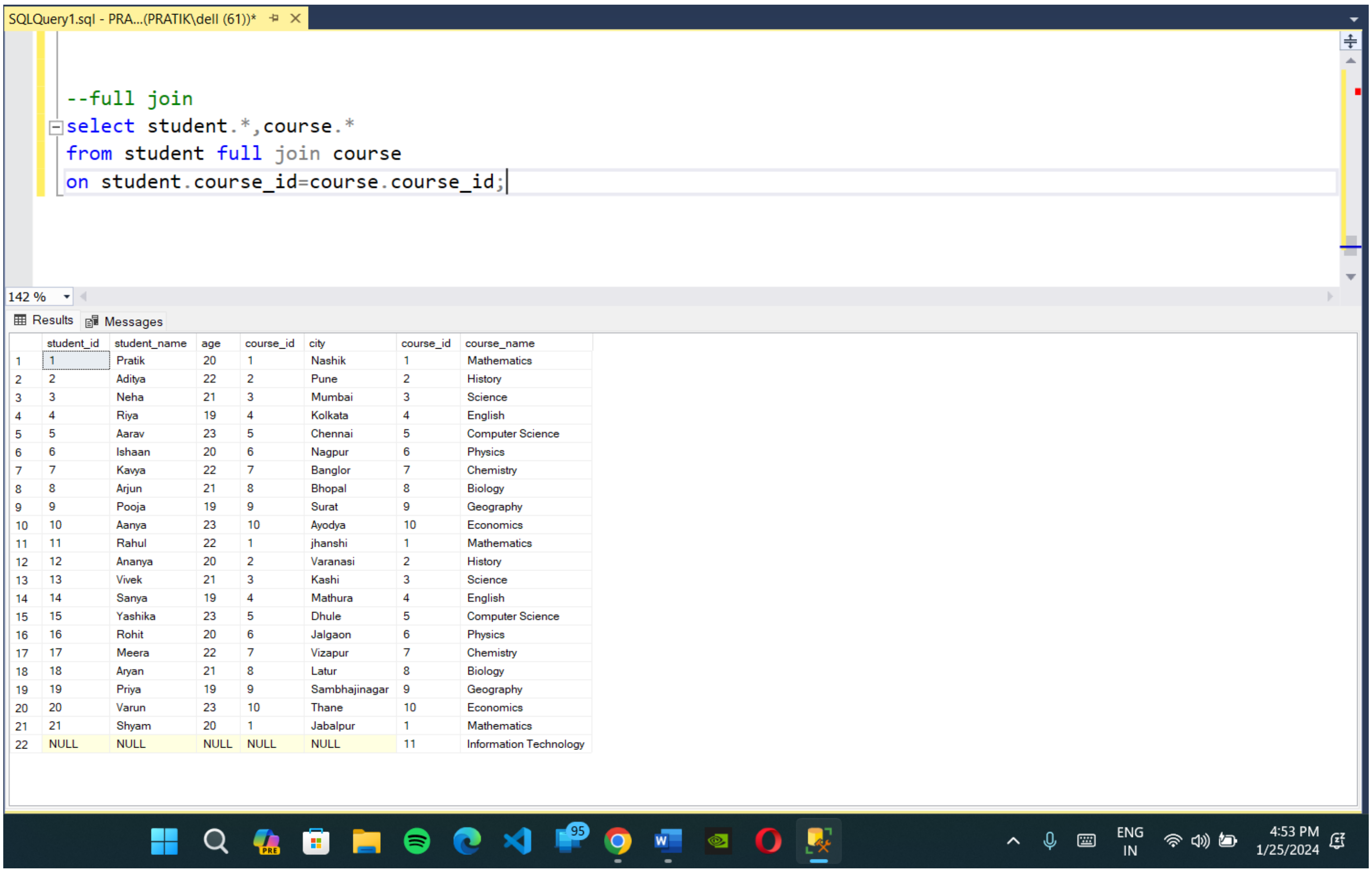


```sql
--Left Join

select student.*,course.*
from  student right join  course
on student.course_id=course.course_id;
```

	student_id	student_name	age	course_id	city	course_id	course_name
1	1	Pratik	20	1	Nashik	1	Mathematics
2	11	Rahul	22	1	jhanshi	1	Mathematics
3	21	Shyam	20	1	Jabalpur	1	Mathematics
4	2	Aditya	22	2	Pune	2	History
5	12	Ananya	20	2	Varanasi	2	History
6	3	Neha	21	3	Mumbai	3	Science
7	13	Vivek	21	3	Kashi	3	Science
8	4	Riya	19	4	Kolkata	4	English
9	14	Sanya	19	4	Mathura	4	English
10	5	Aarav	23	5	Chennai	5	Computer Science
11	15	Yashika	23	5	Dhule	5	Computer Science
12	6	Ishaan	20	6	Nagpur	6	Physics
13	16	Rohit	20	6	Jalgaon	6	Physics
14	7	Kavya	22	7	Banglor	7	Chemistry
15	17	Meera	22	7	Vizapur	7	Chemistry
16	8	Arjun	21	8	Bhopal	8	Biology
17	18	Aryan	21	8	Latur	8	Biology
18	9	Pooja	19	9	Surat	9	Geography
19	19	Priya	19	9	Sambhajinagar	9	Geography
20	10	Aanya	23	10	Ayodya	10	Economics
21	20	Varun	23	10	Thane	10	Economics
22	NULL	NULL	NULL	NULL	NULL	11	Information Technology

○ Full Join:

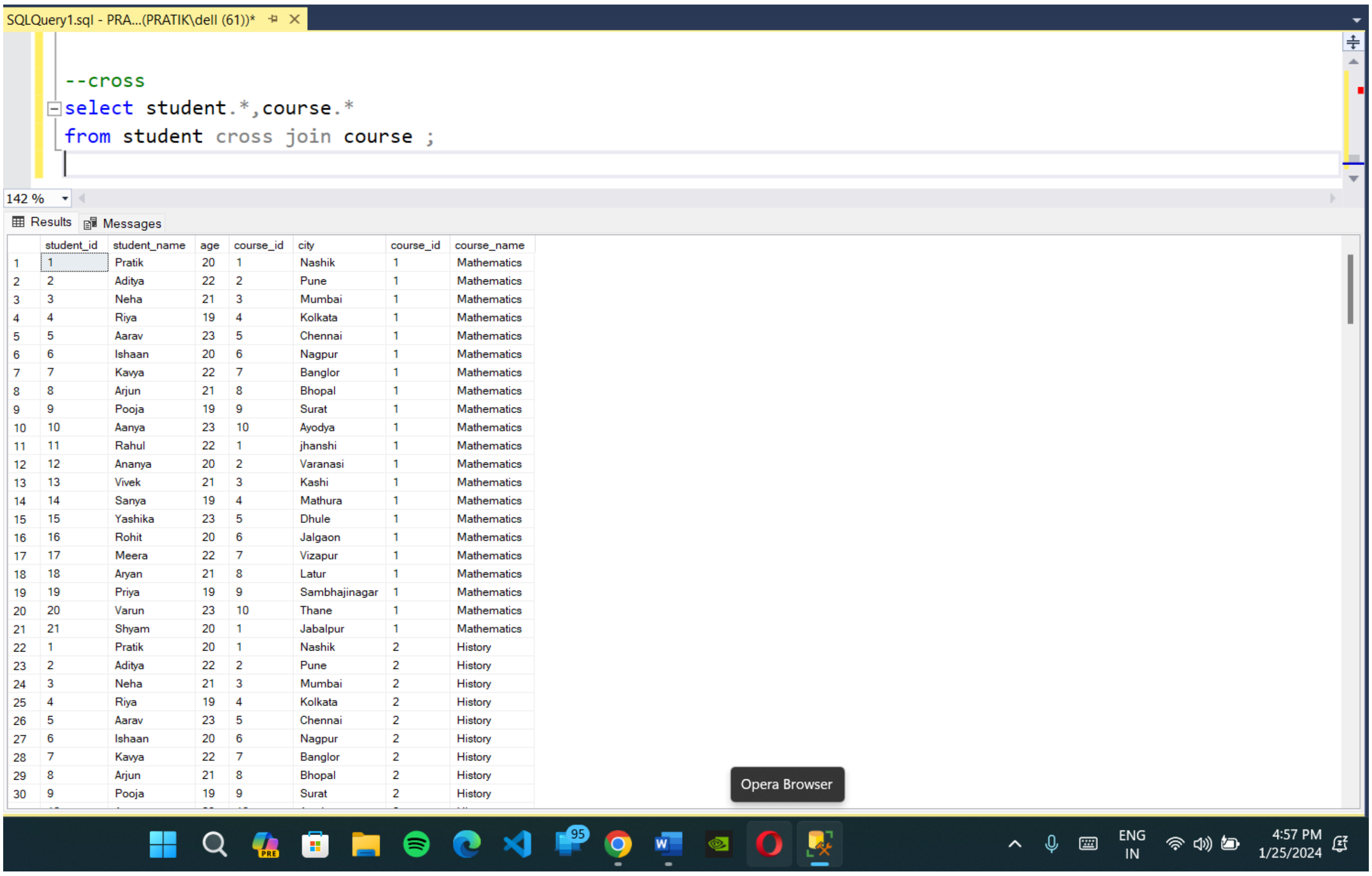
- The Full JOIN returns all records when there is a match in either left or right table records.
- If there is no match, NULL values are returned for columns.
- Also known as FULL OUTER JOIN
- Here all the records from the left table(student) and right table(course) is returned and if there is no matching record then NULL is return.



```
--full join
select student.*,course.*
from student full join course
on student.course_id=course.course_id;
```

	student_id	student_name	age	course_id	city	course_id	course_name
1	1	Pratik	20	1	Nashik	1	Mathematics
2	2	Aditya	22	2	Pune	2	History
3	3	Neha	21	3	Mumbai	3	Science
4	4	Riya	19	4	Kolkata	4	English
5	5	Aarav	23	5	Chennai	5	Computer Science
6	6	Ishaan	20	6	Nagpur	6	Physics
7	7	Kavya	22	7	Banglor	7	Chemistry
8	8	Arjun	21	8	Bhopal	8	Biology
9	9	Pooja	19	9	Surat	9	Geography
10	10	Aanya	23	10	Ayodya	10	Economics
11	11	Rahul	22	1	jhanshi	1	Mathematics
12	12	Ananya	20	2	Varanasi	2	History
13	13	Vivek	21	3	Kashi	3	Science
14	14	Sanya	19	4	Mathura	4	English
15	15	Yashika	23	5	Dhule	5	Computer Science
16	16	Rohit	20	6	Jalgaon	6	Physics
17	17	Meera	22	7	Vizapur	7	Chemistry
18	18	Aryan	21	8	Latur	8	Biology
19	19	Priya	19	9	Sambhajinagar	9	Geography
20	20	Varun	23	10	Thane	10	Economics
21	21	Shyam	20	1	Jabalpur	1	Mathematics
22	NULL	NULL	NULL	NULL	NULL	11	Information Technology

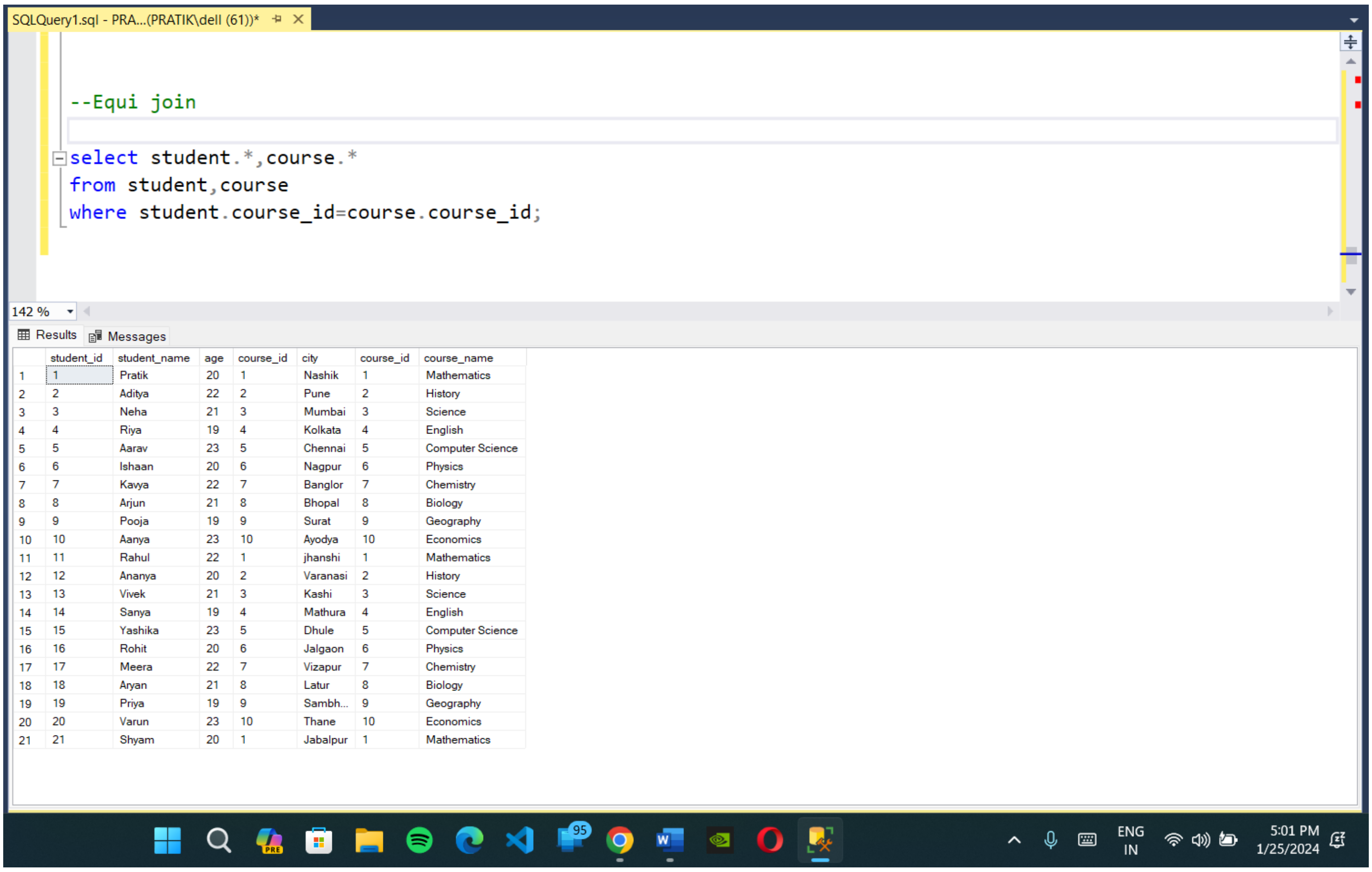
- <u>Cross Join:</u>
  - The CROSS JOIN returns the Cartesian product of both tables.
  - All possible combinations of rows from both tables.
  - It does not require a specific column for the join condition.



```sql
--cross
select student.*,course.*
from student cross join course ;
```

	student_id	student_name	age	course_id	city	course_id	course_name
1	1	Pratik	20	1	Nashik	1	Mathematics
2	2	Aditya	22	2	Pune	1	Mathematics
3	3	Neha	21	3	Mumbai	1	Mathematics
4	4	Riya	19	4	Kolkata	1	Mathematics
5	5	Aarav	23	5	Chennai	1	Mathematics
6	6	Ishaan	20	6	Nagpur	1	Mathematics
7	7	Kavya	22	7	Banglor	1	Mathematics
8	8	Arjun	21	8	Bhopal	1	Mathematics
9	9	Pooja	19	9	Surat	1	Mathematics
10	10	Aanya	23	10	Ayodya	1	Mathematics
11	11	Rahul	22	1	jhanshi	1	Mathematics
12	12	Ananya	20	2	Varanasi	1	Mathematics
13	13	Vivek	21	3	Kashi	1	Mathematics
14	14	Sanya	19	4	Mathura	1	Mathematics
15	15	Yashika	23	5	Dhule	1	Mathematics
16	16	Rohit	20	6	Jalgaon	1	Mathematics
17	17	Meera	22	7	Vizapur	1	Mathematics
18	18	Aryan	21	8	Latur	1	Mathematics
19	19	Priya	19	9	Sambhajinagar	1	Mathematics
20	20	Varun	23	10	Thane	1	Mathematics
21	21	Shyam	20	1	Jabalpur	1	Mathematics
22	1	Pratik	20	1	Nashik	2	History
23	2	Aditya	22	2	Pune	2	History
24	3	Neha	21	3	Mumbai	2	History
25	4	Riya	19	4	Kolkata	2	History
26	5	Aarav	23	5	Chennai	2	History
27	6	Ishaan	20	6	Nagpur	2	History
28	7	Kavya	22	7	Banglor	2	History
29	8	Arjun	21	8	Bhopal	2	History
30	9	Pooja	19	9	Surat	2	History

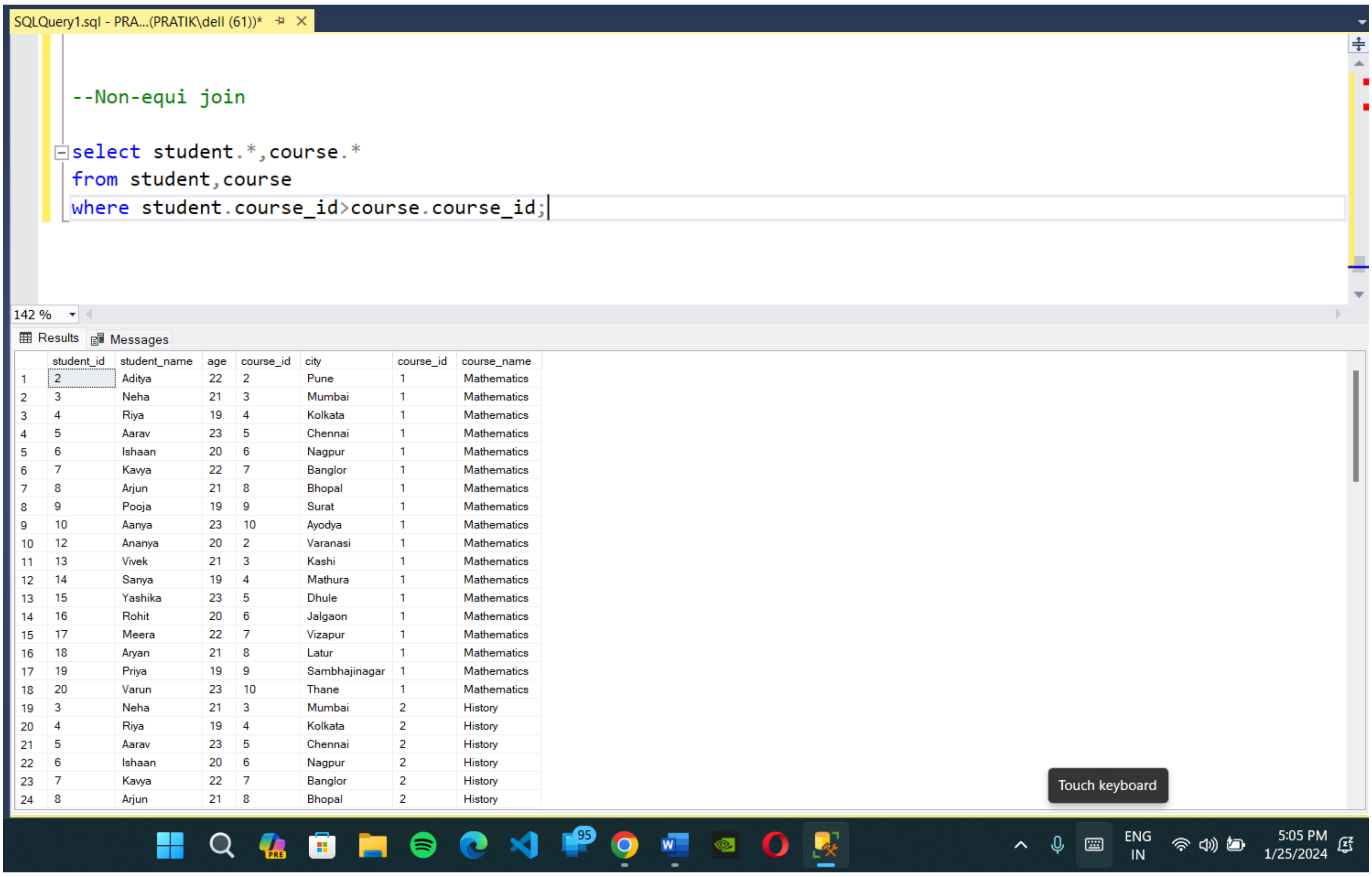
- <u>Equi Join:</u>
  - The EQUI JOIN involves equality between columns in two different tables.
  - An equi join is similar to INNER JOIN but here it works on '=' Operator.



```
--Equi join

select student.*,course.*
from student,course
where student.course_id=course.course_id;
```

	student_id	student_name	age	course_id	city	course_id	course_name
1	1	Pratik	20	1	Nashik	1	Mathematics
2	2	Aditya	22	2	Pune	2	History
3	3	Neha	21	3	Mumbai	3	Science
4	4	Riya	19	4	Kolkata	4	English
5	5	Aarav	23	5	Chennai	5	Computer Science
6	6	Ishaan	20	6	Nagpur	6	Physics
7	7	Kavya	22	7	Banglor	7	Chemistry
8	8	Arjun	21	8	Bhopal	8	Biology
9	9	Pooja	19	9	Surat	9	Geography
10	10	Aanya	23	10	Ayodya	10	Economics
11	11	Rahul	22	1	jhanshi	1	Mathematics
12	12	Ananya	20	2	Varanasi	2	History
13	13	Vivek	21	3	Kashi	3	Science
14	14	Sanya	19	4	Mathura	4	English
15	15	Yashika	23	5	Dhule	5	Computer Science
16	16	Rohit	20	6	Jalgaon	6	Physics
17	17	Meera	22	7	Vizapur	7	Chemistry
18	18	Aryan	21	8	Latur	8	Biology
19	19	Priya	19	9	Sambh...	9	Geography
20	20	Varun	23	10	Thane	10	Economics
21	21	Shyam	20	1	Jabalpur	1	Mathematics

- <u>Non-Equi Join:</u>
    - The NON-EQUI JOIN involves a comparison other than equality between the columns of two tables.
    - non-equi joins use other comparison operators such as <, >, <=, >=, or <>
    - Here I used the > operator
    - Hence got the record for students having course id greater than there course id.


SQLQuery1.sql - PRA...(PRATIK\dell (61))*

```sql
--Non-equi join

select student.*,course.*
from student,course
where student.course_id>course.course_id;
```

142 %

Results | Messages

	student_id	student_name	age	course_id	city	course_id	course_name
1	2	Aditya	22	2	Pune	1	Mathematics
2	3	Neha	21	3	Mumbai	1	Mathematics
3	4	Riya	19	4	Kolkata	1	Mathematics
4	5	Aarav	23	5	Chennai	1	Mathematics
5	6	Ishaan	20	6	Nagpur	1	Mathematics
6	7	Kavya	22	7	Banglor	1	Mathematics
7	8	Arjun	21	8	Bhopal	1	Mathematics
8	9	Pooja	19	9	Surat	1	Mathematics
9	10	Aanya	23	10	Ayodya	1	Mathematics
10	12	Ananya	20	2	Varanasi	1	Mathematics
11	13	Vivek	21	3	Kashi	1	Mathematics
12	14	Sanya	19	4	Mathura	1	Mathematics
13	15	Yashika	23	5	Dhule	1	Mathematics
14	16	Rohit	20	6	Jalgaon	1	Mathematics
15	17	Meera	22	7	Vizapur	1	Mathematics
16	18	Aryan	21	8	Latur	1	Mathematics
17	19	Priya	19	9	Sambhajinagar	1	Mathematics
18	20	Varun	23	10	Thane	1	Mathematics
19	3	Neha	21	3	Mumbai	2	History
20	4	Riya	19	4	Kolkata	2	History
21	5	Aarav	23	5	Chennai	2	History
22	6	Ishaan	20	6	Nagpur	2	History
23	7	Kavya	22	7	Banglor	2	History
24	8	Arjun	21	8	Bhopal	2	History