NAME : SHARATH CHANDRA
BATCH : DATA ENGINEERING
DATE : 25-01-24

SQL CODING CHALLENGE

QUESTIONS:

1)

a)Execute OVER and PARTITION BY Clause in SQL Queries

b)creating subtotals

c)Total Aggregations using SQL Queries.

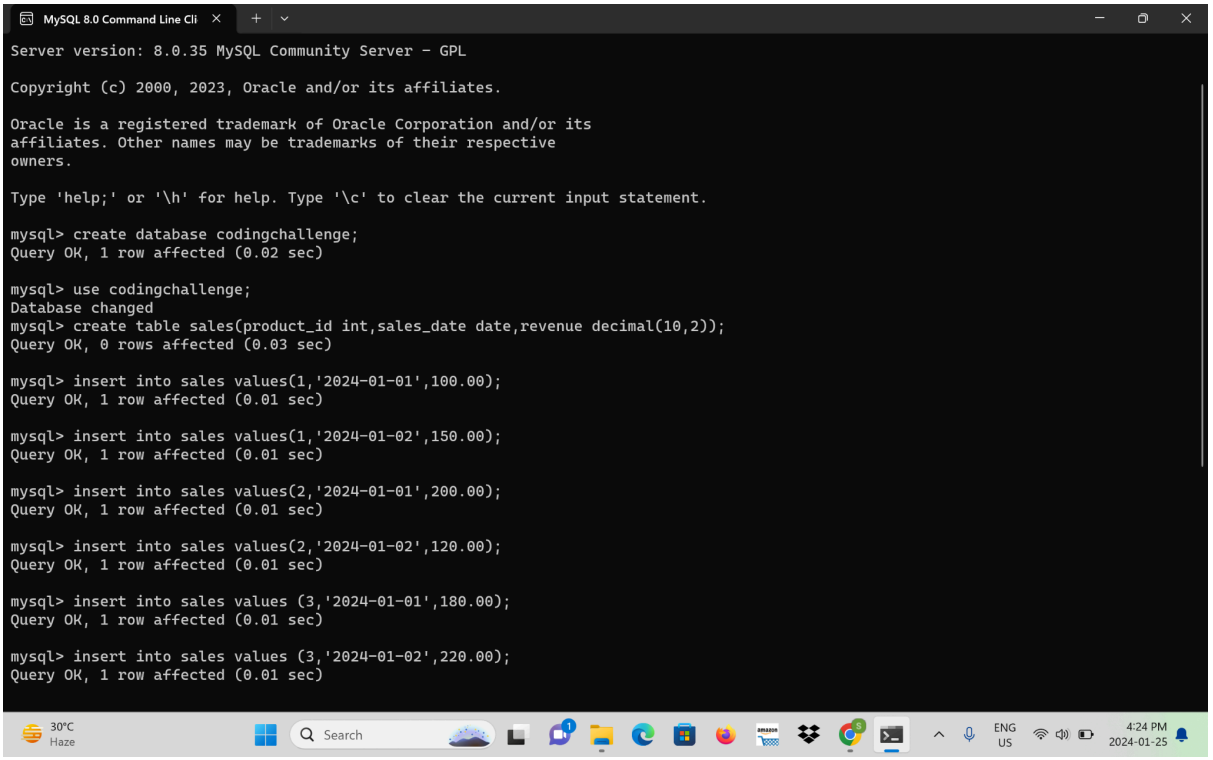
2)Execute all the join with examples.

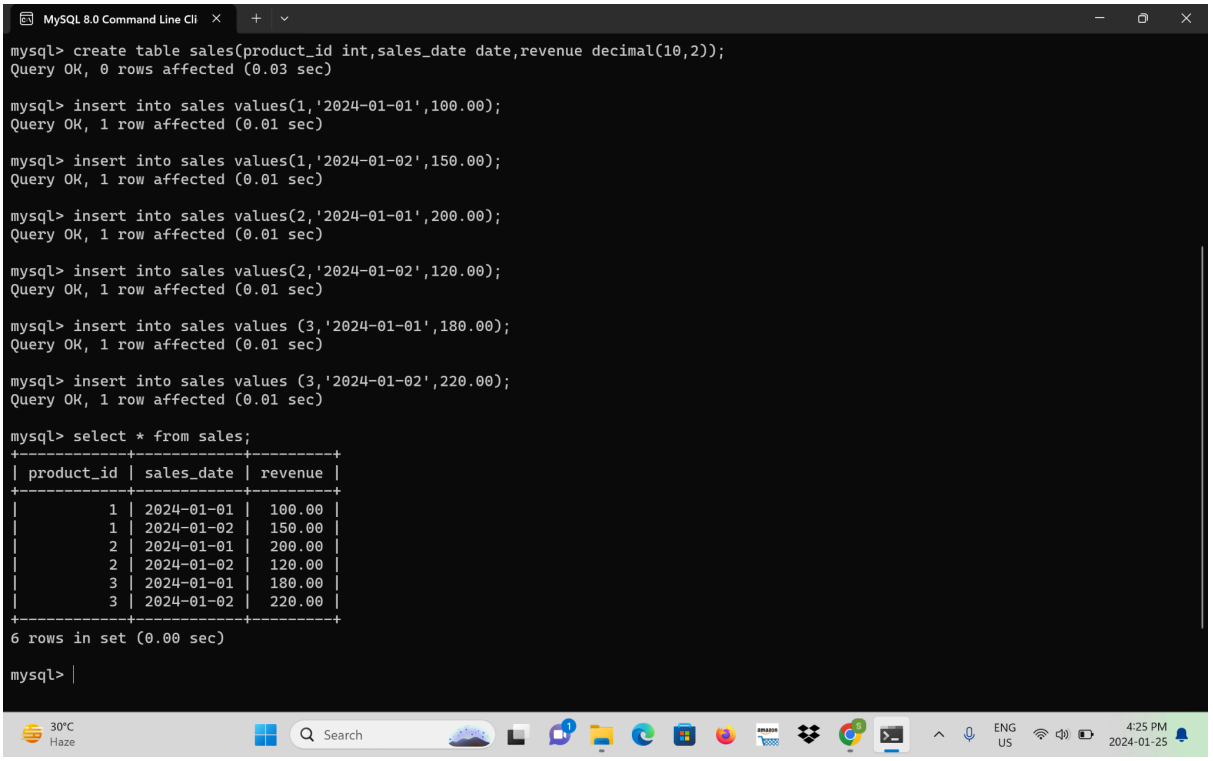
Solutions:

1)
a)Execute OVER and PARTITION BY Clause in SQL Queries:

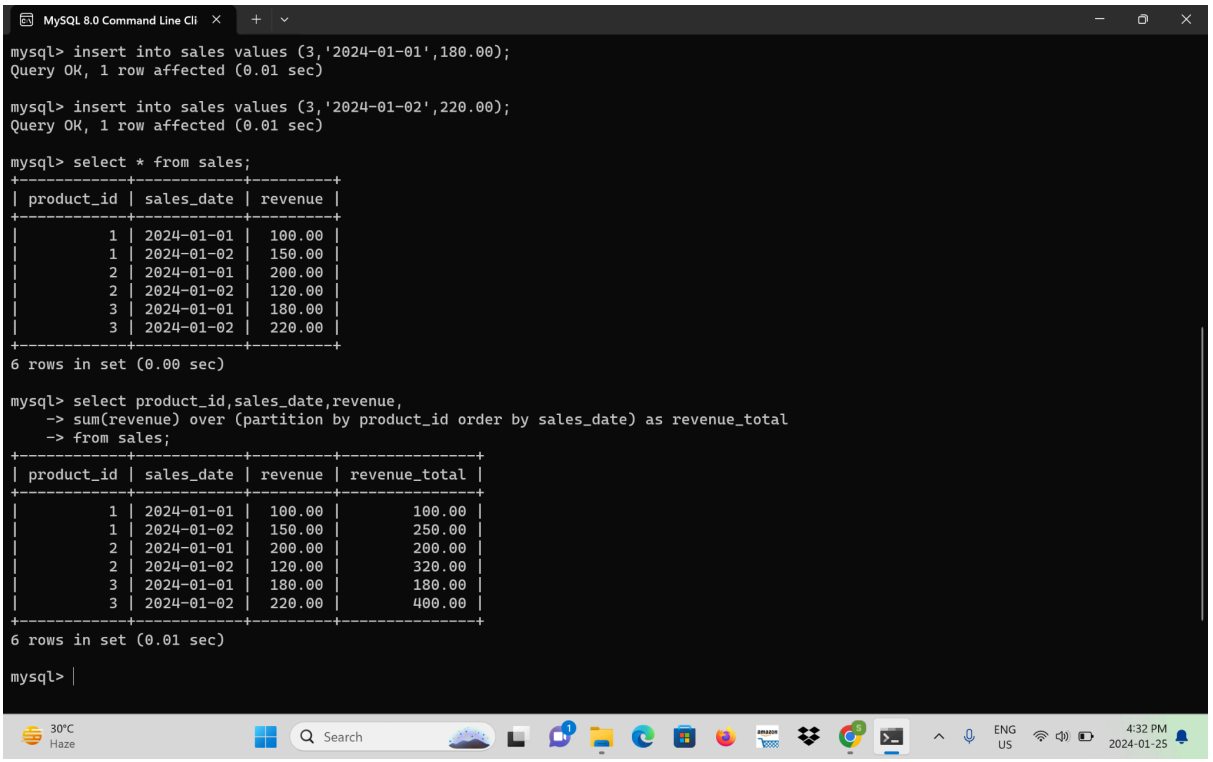
Answer:

-The over and partition by clauses are used together to perform window functions, which operate on a specific window of rows defined by the partition by clause.
-The partition by clause divides the result set into partitions based on specified columns.
-The over clause, working with partition by sets the range of rows for the window function. It decides the order of rows in each group, making it possible to do cumulative calculations or assign rankings within those groups.


MySQL 8.0 Command Line Cli
Server version: 8.0.35 MySQL Community Server - GPL
Copyright (c) 2000, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> create database codingchallenge;
Query OK, 1 row affected (0.02 sec)
mysql> use codingchallenge;
Database changed
mysql> create table sales(product_id int,sales_date date,revenue decimal(10,2));
Query OK, 0 rows affected (0.03 sec)
mysql> insert into sales values(1,'2024-01-01',100.00);
Query OK, 1 row affected (0.01 sec)
mysql> insert into sales values(1,'2024-01-02',150.00);
Query OK, 1 row affected (0.01 sec)
mysql> insert into sales values(2,'2024-01-01',200.00);
Query OK, 1 row affected (0.01 sec)
mysql> insert into sales values(2,'2024-01-02',120.00);
Query OK, 1 row affected (0.01 sec)
mysql> insert into sales values (3,'2024-01-01',180.00);
Query OK, 1 row affected (0.01 sec)
mysql> insert into sales values (3,'2024-01-02',220.00);
Query OK, 1 row affected (0.01 sec)
30°C
Haze
Search
ENG
US
4:24 PM
2024-01-25


MySQL 8.0 Command Line Cli
mysql> create table sales(product_id int,sales_date date,revenue decimal(10,2));
Query OK, 0 rows affected (0.03 sec)
mysql> insert into sales values(1,'2024-01-01',100.00);
Query OK, 1 row affected (0.01 sec)
mysql> insert into sales values(1,'2024-01-02',150.00);
Query OK, 1 row affected (0.01 sec)
mysql> insert into sales values(2,'2024-01-01',200.00);
Query OK, 1 row affected (0.01 sec)
mysql> insert into sales values(2,'2024-01-02',120.00);
Query OK, 1 row affected (0.01 sec)
mysql> insert into sales values (3,'2024-01-01',180.00);
Query OK, 1 row affected (0.01 sec)
mysql> insert into sales values (3,'2024-01-02',220.00);
Query OK, 1 row affected (0.01 sec)
mysql> select * from sales;
+------------+------------+---------+
| product_id | sales_date | revenue |
+------------+------------+---------+
1	2024-01-01	100.00
1	2024-01-02	150.00
2	2024-01-01	200.00
2	2024-01-02	120.00
3	2024-01-01	180.00
3	2024-01-02	220.00
+------------+------------+---------+
6 rows in set (0.00 sec)
mysql>
30°C
Haze
Search
ENG
US
4:25 PM
2024-01-25

EXECUTING THE QUERY AND RESULTING THE OUTPUT:



Explaining the query:

→I Have taken the example to calculate the running total of revenue for each product.

→ so firstly i have created a database named codingchallenge

→then i have created a table named sales with columns product_id, sales_date, revenue and then i have inserted values into it.

→ Then i have written a query on over and partition by clause and executed it the code follows:

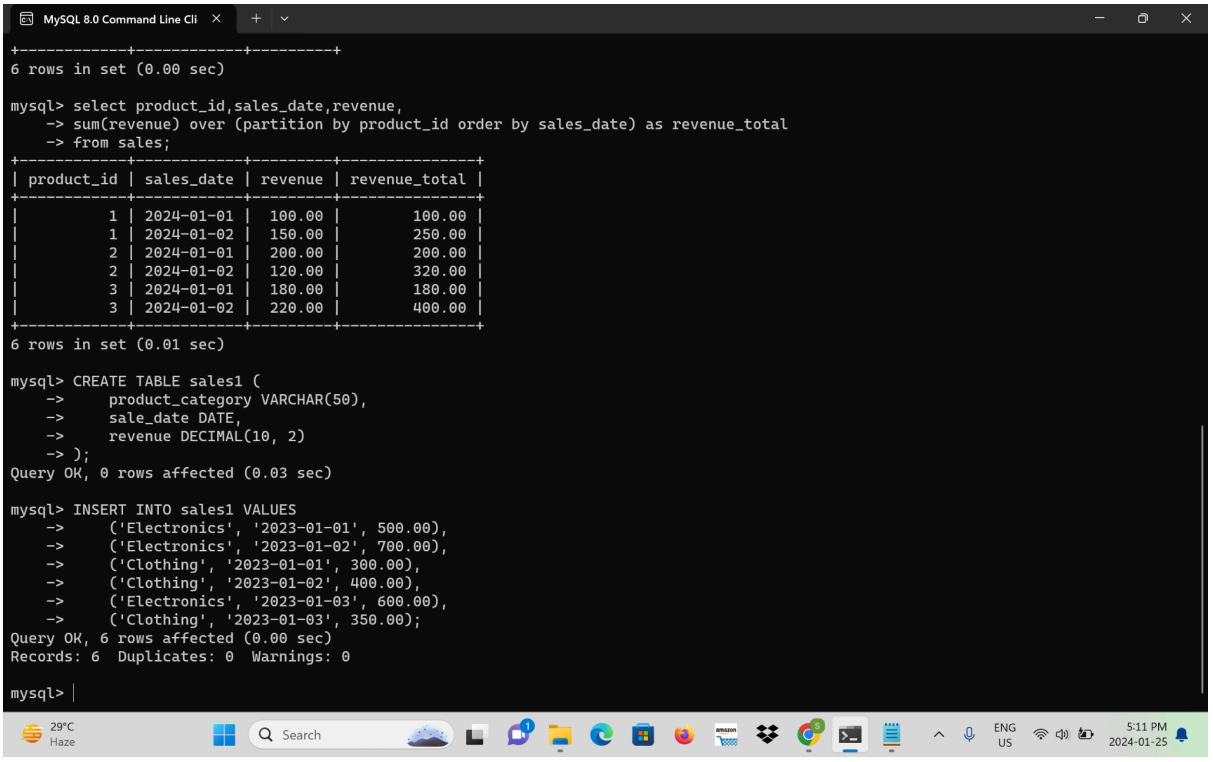
```
SELECT
    product_id,
    sales_date,
    revenue,
    SUM(revenue) OVER (PARTITION BY product_id ORDER BY sales_date) AS
running_total
FROM sales;
```

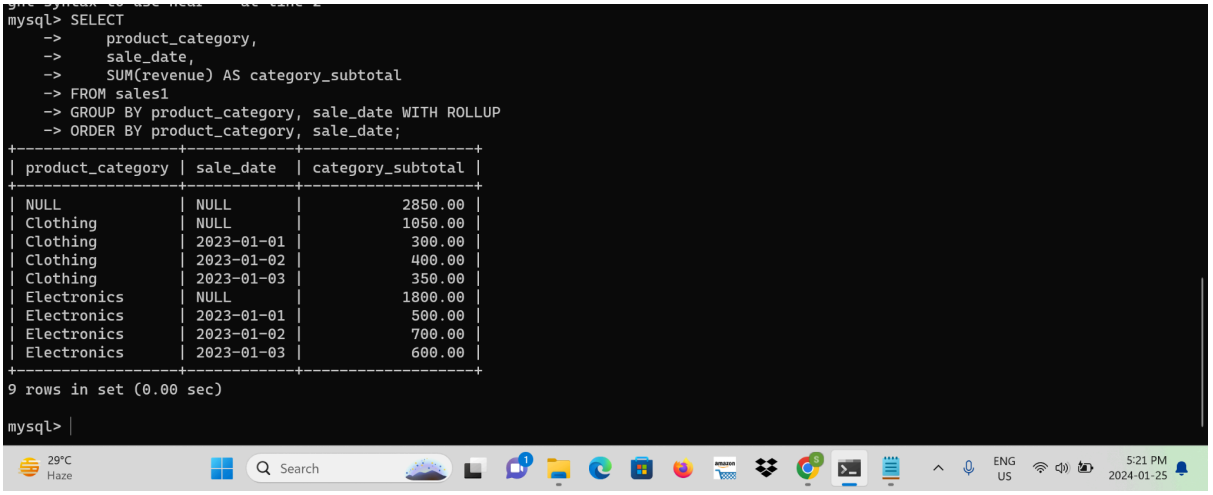
→ This code explains us by
—>The SUM(revenue) calculates the cumulative sum of revenue.
—>The OVER clause is used to define the window over which the cumulative sum is calculated.
—>The PARTITION BY product_id indicates that the window should be partitioned by product ID, so the running total resets for each product.
—>The ORDER BY sales_date specifies the order in which the running total is computed, ensuring it is cumulative based on the sales date.
→ Thereby it will calculate the running total of revenue for each product.

b)creating subtotals:

—>Subtotals are the intermediate or partial sums or calculations within a grouped set of data.
—>subtotals provide the sum or aggregation for each group, offering insights into specific segments of the data.

Example:calculate subtotals for each product category and a grand total for all records

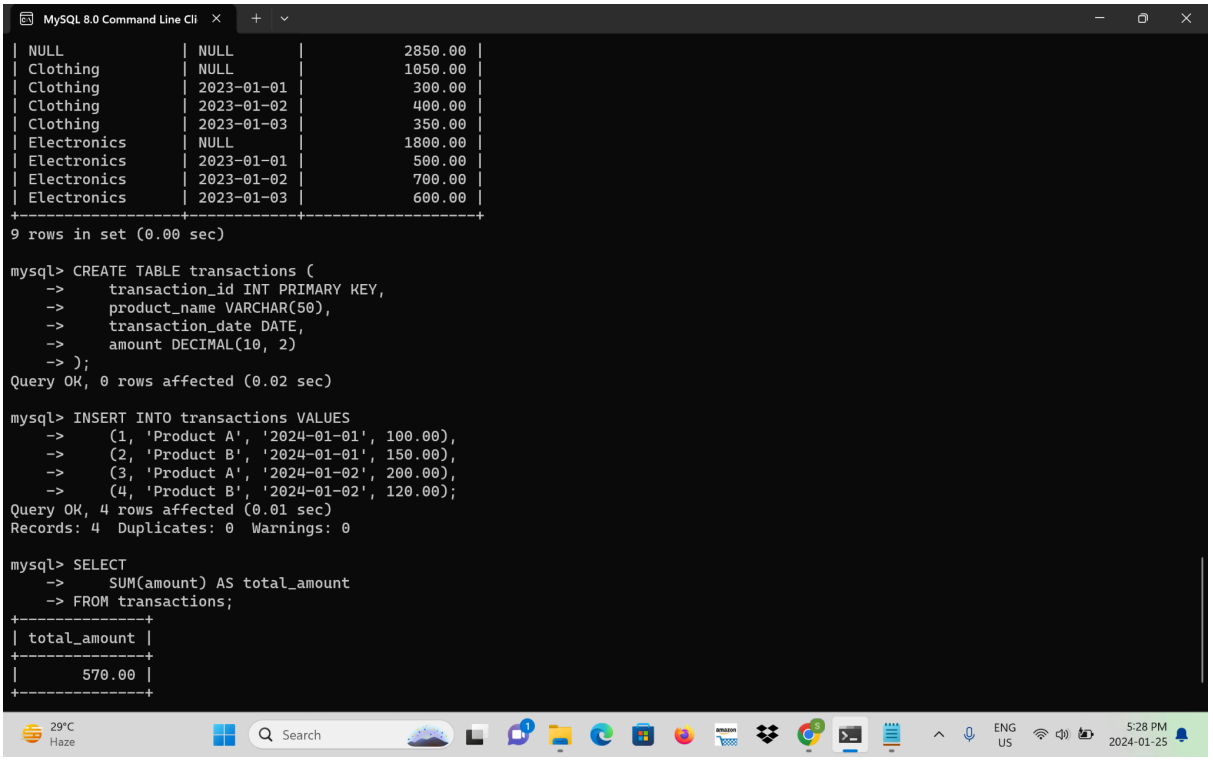




Summary:

The with rollup clause in the group by statement is used to generate subtotals and a grand total. It produces additional rows in the result set that represent subtotals and a grand total based on the specified grouping columns.

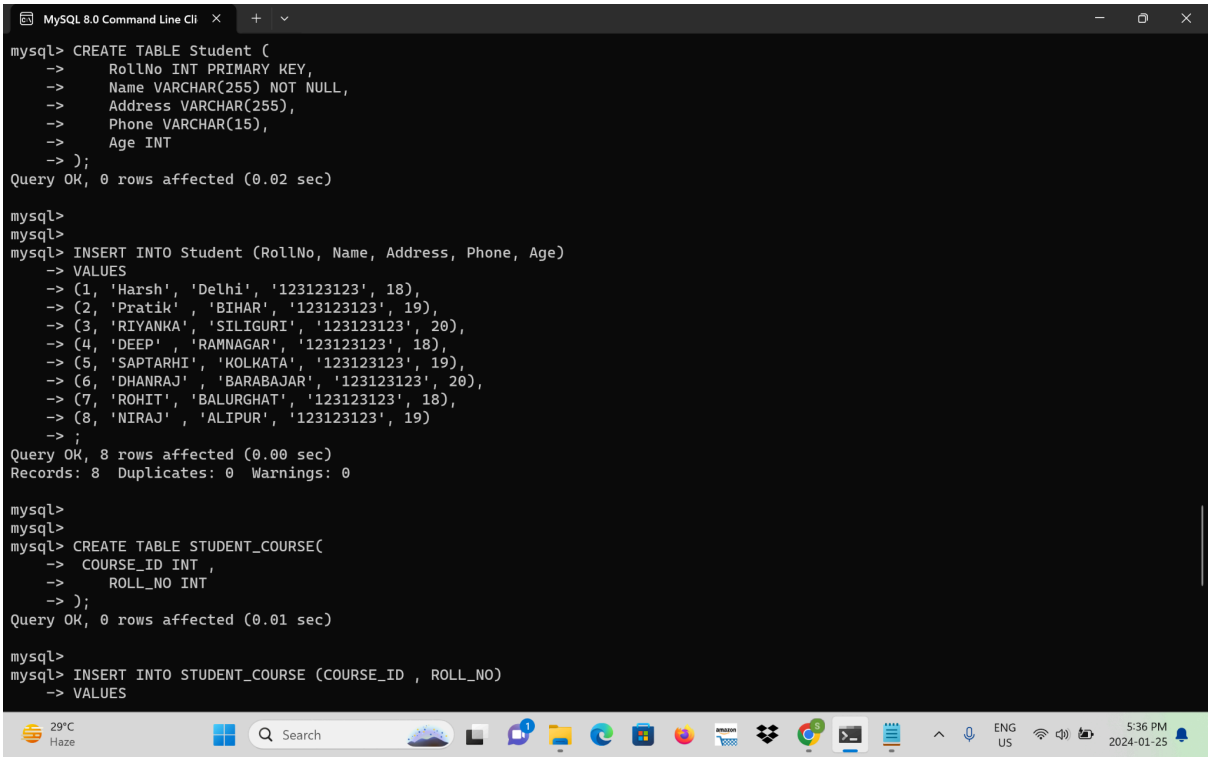
c)Total Aggregations using SQL Queries:



Summary : will provide the total amount across all records in the transactions table.

2)Execute all the join with examples.

i)JOINS IN SQL : a JOIN in SQL is a way to combine rows from two or more tables based on a related column between them. It allows you to retrieve data from multiple tables simultaneously.



```
mysql>
mysql> INSERT INTO STUDENT_COURSE (COURSE_ID , ROLL_NO)
    -> VALUES
    -> (1, 1),
    -> (2,2),
    -> (2,3),
    -> (3,4),
    -> (1,5),
    -> (1,5),
    -> (4,9),
    -> (5,10),
    -> (4,11)
    -> ;
Query OK, 9 rows affected (0.01 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> select * from Student;
+--------+----------+-----------+-----------+------+
| RollNo | Name     | Address   | Phone     | Age  |
+--------+----------+-----------+-----------+------+
|      1 | Harsh    | Delhi     | 123123123 |   18 |
|      2 | Pratik   | BIHAR     | 123123123 |   19 |
|      3 | RIYANKA  | SILIGURI  | 123123123 |   20 |
|      4 | DEEP     | RAMNAGAR  | 123123123 |   18 |
|      5 | SAPTARHI | KOLKATA   | 123123123 |   19 |
|      6 | DHANRAJ  | BARABAJAR | 123123123 |   20 |
|      7 | ROHIT    | BALURGHAT | 123123123 |   18 |
|      8 | NIRAJ    | ALIPUR    | 123123123 |   19 |
+--------+----------+-----------+-----------+------+
8 rows in set (0.00 sec)

mysql> select * from STUDENT_COURSE;
+-----------+---------+
```


```
mysql> select * from STUDENT_COURSE;
+-----------+---------+
| COURSE_ID | ROLL_NO |
+-----------+---------+
|         1 |       1 |
|         2 |       2 |
|         2 |       3 |
|         3 |       4 |
|         1 |       5 |
|         1 |       5 |
|         4 |       9 |
|         5 |      10 |
|         4 |      11 |
+-----------+---------+
9 rows in set (0.00 sec)

mysql>
```

INNER JOIN: It returns only the rows that have matching values in both tables.

```
mysql> SELECT Student_COURSE.COURSE_ID, Student.NAME, Student.AGE FROM Student
    -> INNER JOIN Student_Course
    -> ON Student.ROLLNO = Student_Course.ROLL_NO;
+-----------+----------+------+
| COURSE_ID | NAME     | AGE  |
+-----------+----------+------+
|         1 | Harsh    |   18 |
|         2 | Pratik   |   19 |
|         2 | RIYANKA  |   20 |
|         3 | DEEP     |   18 |
|         1 | SAPTARHI |   19 |
|         1 | SAPTARHI |   19 |
+-----------+----------+------+
6 rows in set (0.00 sec)

mysql>
```

LEFT JOIN: This type of JOIN returns all the rows from the left table and the matched rows from the right table.

```
mysql> SELECT Student.NAME,Student_Course.COURSE_ID
    -> FROM Student
    -> LEFT JOIN Student_Course
    -> ON Student_Course.ROLL_NO = Student.ROLLNO;
+----------+-----------+
| NAME     | COURSE_ID |
+----------+-----------+
| Harsh    |         1 |
| Pratik   |         2 |
| RIYANKA  |         2 |
| DEEP     |         3 |
| SAPTARHI |         1 |
| SAPTARHI |         1 |
| DHANRAJ  |      NULL |
| ROHIT    |      NULL |
| NIRAJ    |      NULL |
+----------+-----------+
9 rows in set (0.00 sec)

mysql>
```

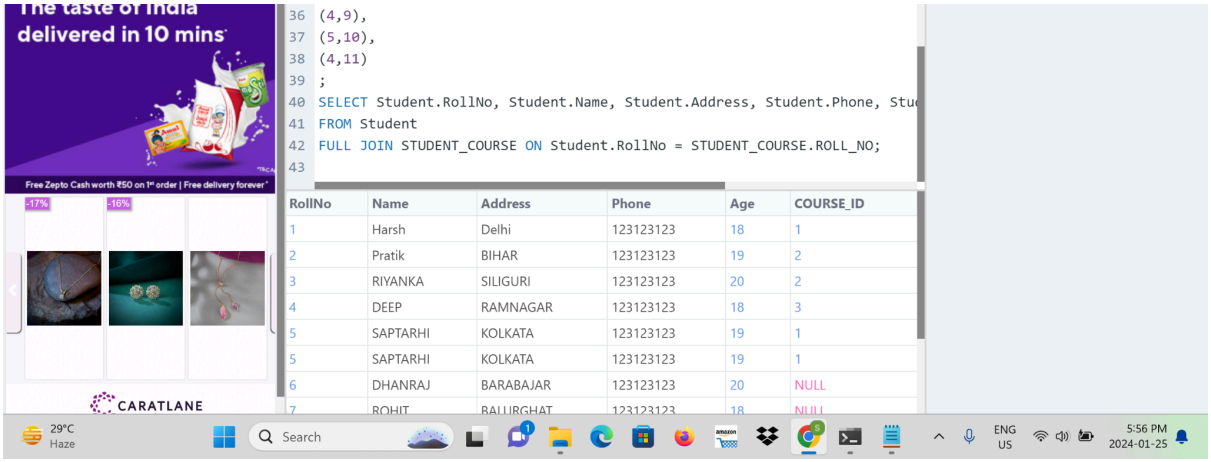
RIGHT JOIN : it returns all the rows from the right table and the matched rows from the left table.

```
mysql> SELECT Student.NAME,Student_Course.COURSE_ID
    -> FROM Student
    -> RIGHT JOIN Student_Course
    -> ON Student_Course.ROLL_NO = Student.ROLLNO;
+----------+-----------+
| NAME     | COURSE_ID |
+----------+-----------+
| Harsh    |         1 |
| Pratik   |         2 |
| RIYANKA  |         2 |
| DEEP     |         3 |
| SAPTARHI |         1 |
| SAPTARHI |         1 |
| NULL     |         4 |
| NULL     |         5 |
| NULL     |         4 |
+----------+-----------+
9 rows in set (0.00 sec)

mysql>
```

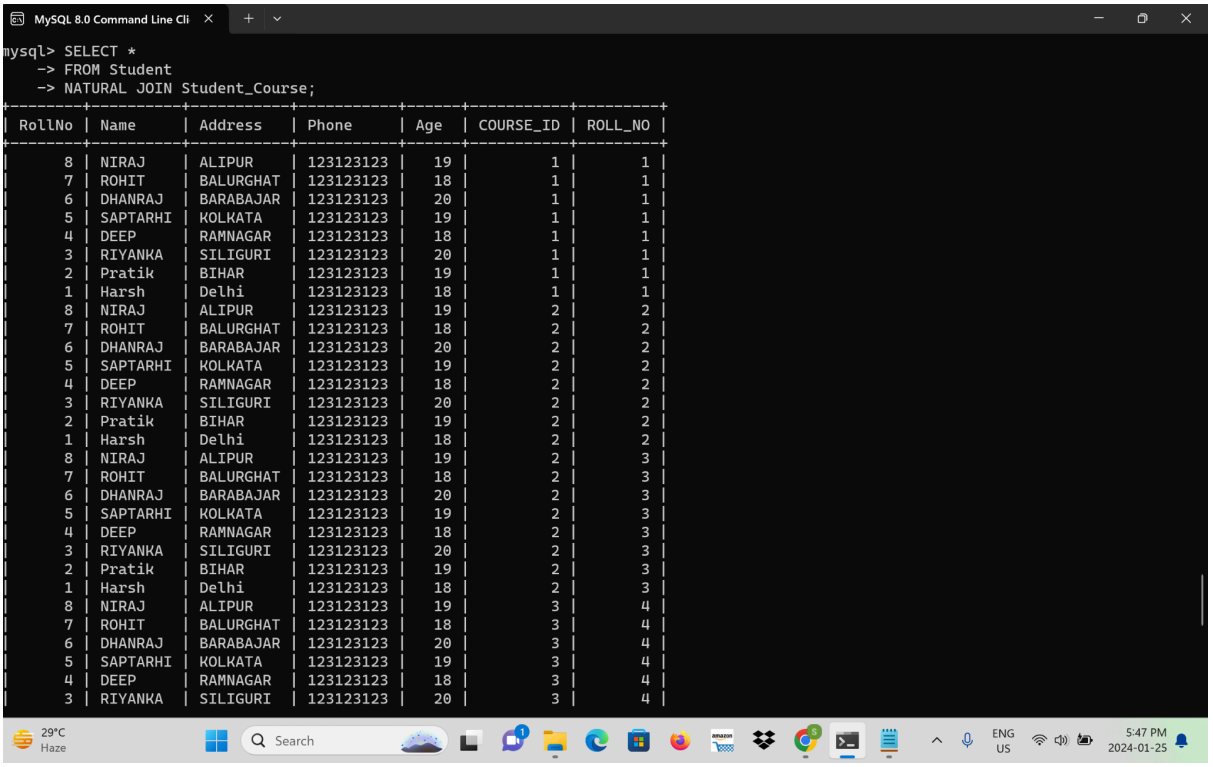
FULL JOIN: This JOIN returns all rows when there is a match in either the left or right table.

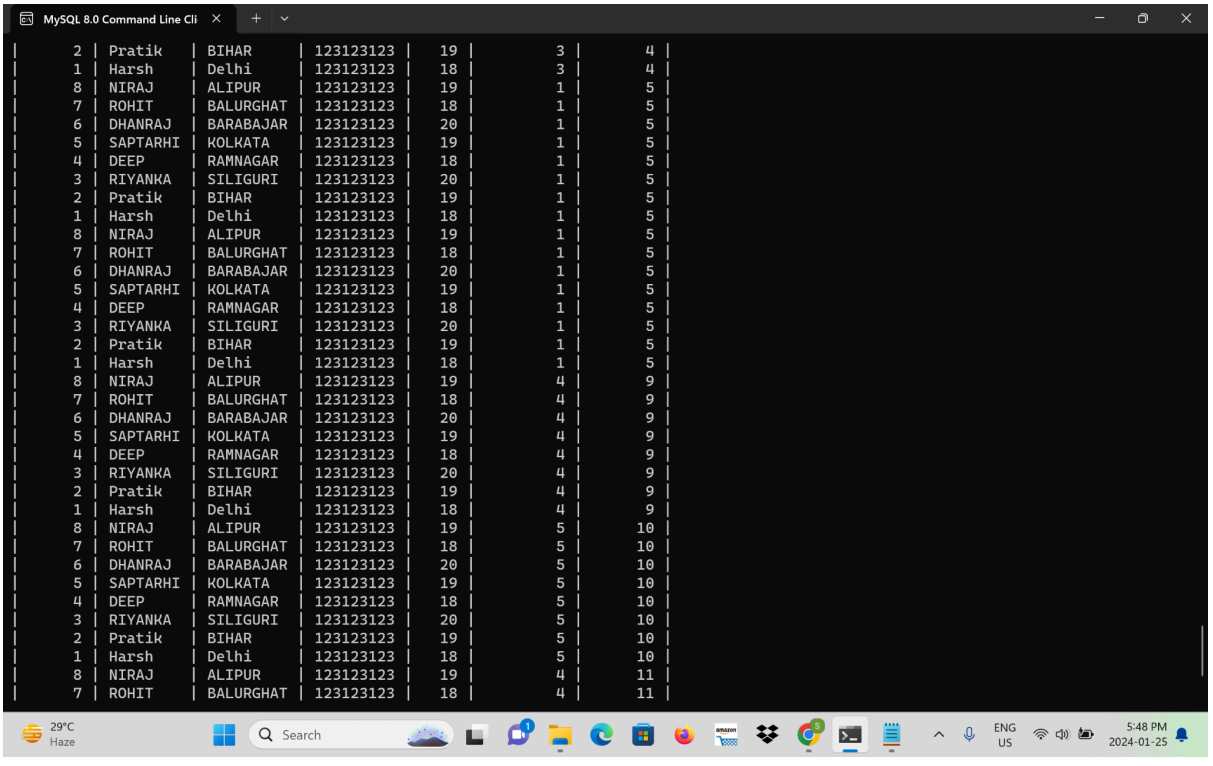
MAM FOR FULL JOIN IN MY SQL CLIENT NOT WORKING SO I USED ONLINE COMPILER MAM.

```
72 rows in set (0.00 sec)

mysql> SELECT Student.Name, Student_Course.COURSE_ID
    -> FROM Student
    -> FULL JOIN Student_Course ON Student_Course.ROLL_NO = Student.ROLLNO;
ERROR 1054 (42S22): Unknown column 'Student.Name' in 'field list'
mysql> SELECT Student.Name, STUDENT_COURSE.COURSE_ID
    -> FROM Student
    -> FULL JOIN STUDENT_COURSE ON Student_Course.ROLL_NO = Student.ROLLNO;
ERROR 1054 (42S22): Unknown column 'Student.Name' in 'field list'
mysql> SELECT Student.RollNo, Student.Name, Student.Address, Student.Phone, Student.Age, STUDENT_COURSE.COURSE_ID
    -> FROM Student
    -> FULL JOIN STUDENT_COURSE ON Student.RollNo = STUDENT_COURSE.ROLL_NO;
ERROR 1054 (42S22): Unknown column 'Student.RollNo' in 'field list'
mysql>
```



NATURAL JOIN:a natural join in SQL is a type of join that automatically combines tables based on columns with the same name.


MySQL 8.0 Command Line Cli
mysql> SELECT *
-> FROM Student
-> NATURAL JOIN Student_Course;
RollNo	Name	Address	Phone	Age	COURSE_ID	ROLL_NO
8	NIRAJ	ALIPUR	123123123	19	1	1
7	ROHIT	BALURGHAT	123123123	18	1	1
6	DHANRAJ	BARABAJAR	123123123	20	1	1
5	SAPTARHI	KOLKATA	123123123	19	1	1
4	DEEP	RAMNAGAR	123123123	18	1	1
3	RIYANKA	SILIGURI	123123123	20	1	1
2	Pratik	BIHAR	123123123	19	1	1
1	Harsh	Delhi	123123123	18	1	1
8	NIRAJ	ALIPUR	123123123	19	2	2
7	ROHIT	BALURGHAT	123123123	18	2	2
6	DHANRAJ	BARABAJAR	123123123	20	2	2
5	SAPTARHI	KOLKATA	123123123	19	2	2
4	DEEP	RAMNAGAR	123123123	18	2	2
3	RIYANKA	SILIGURI	123123123	20	2	2
2	Pratik	BIHAR	123123123	19	2	2
1	Harsh	Delhi	123123123	18	2	2
8	NIRAJ	ALIPUR	123123123	19	2	3
7	ROHIT	BALURGHAT	123123123	18	2	3
6	DHANRAJ	BARABAJAR	123123123	20	2	3
5	SAPTARHI	KOLKATA	123123123	19	2	3
4	DEEP	RAMNAGAR	123123123	18	2	3
3	RIYANKA	SILIGURI	123123123	20	2	3
2	Pratik	BIHAR	123123123	19	2	3
1	Harsh	Delhi	123123123	18	2	3
8	NIRAJ	ALIPUR	123123123	19	3	4
7	ROHIT	BALURGHAT	123123123	18	3	4
6	DHANRAJ	BARABAJAR	123123123	20	3	4
5	SAPTARHI	KOLKATA	123123123	19	3	4
4	DEEP	RAMNAGAR	123123123	18	3	4
3	RIYANKA	SILIGURI	123123123	20	3	4
29°C Haze
Search
ENG US
5:47 PM 2024-01-25


MySQL 8.0 Command Line Cli
2	Pratik	BIHAR	123123123	19	3	4
1	Harsh	Delhi	123123123	18	3	4
8	NIRAJ	ALIPUR	123123123	19	1	5
7	ROHIT	BALURGHAT	123123123	18	1	5
6	DHANRAJ	BARABAJAR	123123123	20	1	5
5	SAPTARHI	KOLKATA	123123123	19	1	5
4	DEEP	RAMNAGAR	123123123	18	1	5
3	RIYANKA	SILIGURI	123123123	20	1	5
2	Pratik	BIHAR	123123123	19	1	5
1	Harsh	Delhi	123123123	18	1	5
8	NIRAJ	ALIPUR	123123123	19	1	5
7	ROHIT	BALURGHAT	123123123	18	1	5
6	DHANRAJ	BARABAJAR	123123123	20	1	5
5	SAPTARHI	KOLKATA	123123123	19	1	5
4	DEEP	RAMNAGAR	123123123	18	1	5
3	RIYANKA	SILIGURI	123123123	20	1	5
2	Pratik	BIHAR	123123123	19	1	5
1	Harsh	Delhi	123123123	18	1	5
8	NIRAJ	ALIPUR	123123123	19	4	9
7	ROHIT	BALURGHAT	123123123	18	4	9
6	DHANRAJ	BARABAJAR	123123123	20	4	9
5	SAPTARHI	KOLKATA	123123123	19	4	9
4	DEEP	RAMNAGAR	123123123	18	4	9
3	RIYANKA	SILIGURI	123123123	20	4	9
2	Pratik	BIHAR	123123123	19	4	9
1	Harsh	Delhi	123123123	18	4	9
8	NIRAJ	ALIPUR	123123123	19	5	10
7	ROHIT	BALURGHAT	123123123	18	5	10
6	DHANRAJ	BARABAJAR	123123123	20	5	10
5	SAPTARHI	KOLKATA	123123123	19	5	10
4	DEEP	RAMNAGAR	123123123	18	5	10
3	RIYANKA	SILIGURI	123123123	20	5	10
2	Pratik	BIHAR	123123123	19	5	10
1	Harsh	Delhi	123123123	18	5	10
8	NIRAJ	ALIPUR	123123123	19	4	11
7	ROHIT	BALURGHAT	123123123	18	4	11
29°C Haze
Search
ENG US
5:48 PM 2024-01-25

```
|      1 | Harsh     | Delhi      | 123123123 |   18 |          4 |       9 |
|      8 | NIRAJ     | ALIPUR     | 123123123 |   19 |          5 |      10 |
|      7 | ROHIT     | BALURGHAT  | 123123123 |   18 |          5 |      10 |
|      6 | DHANRAJ   | BARABAJAR  | 123123123 |   20 |          5 |      10 |
|      5 | SAPTARHI  | KOLKATA    | 123123123 |   19 |          5 |      10 |
|      4 | DEEP      | RAMNAGAR   | 123123123 |   18 |          5 |      10 |
|      3 | RIYANKA   | SILIGURI   | 123123123 |   20 |          5 |      10 |
|      2 | Pratik    | BIHAR      | 123123123 |   19 |          5 |      10 |
|      1 | Harsh     | Delhi      | 123123123 |   18 |          5 |      10 |
|      8 | NIRAJ     | ALIPUR     | 123123123 |   19 |          4 |      11 |
|      7 | ROHIT     | BALURGHAT  | 123123123 |   18 |          4 |      11 |
|      6 | DHANRAJ   | BARABAJAR  | 123123123 |   20 |          4 |      11 |
|      5 | SAPTARHI  | KOLKATA    | 123123123 |   19 |          4 |      11 |
|      4 | DEEP      | RAMNAGAR   | 123123123 |   18 |          4 |      11 |
|      3 | RIYANKA   | SILIGURI   | 123123123 |   20 |          4 |      11 |
|      2 | Pratik    | BIHAR      | 123123123 |   19 |          4 |      11 |
|      1 | Harsh     | Delhi      | 123123123 |   18 |          4 |      11 |
+--------+-----------+------------+-----------+------+------------+---------+
72 rows in set (0.00 sec)

mysql>
```