**Name -Monika Gariya**

**Email-** [**monikagariya2023@gmail.com**](mailto:monikagariya2023@gmail.com)

**Data Engineering Batch 1**

**Date – 19-01-2024**

**Topic – MySQL, Logical Operators, Joins**

MySQL is a relational database management system and open-source, free and cross platform. MySQL is ideal for both small and large applications and it is very fast, reliable, scalable, and easy to use.

**What Are the Data Languages Within SQL?**

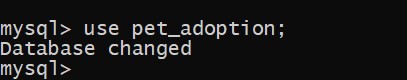
**DDL (Data Definition Language)** allows us to define what the structure of our databases looks like using commands such as create, drop, alter, truncate, rename.

**DML (Data Manipulation Language)** provides the methods for how to manipulate the data to actually do the adding, changing, and deleting through commands like select, insert, update and delete.

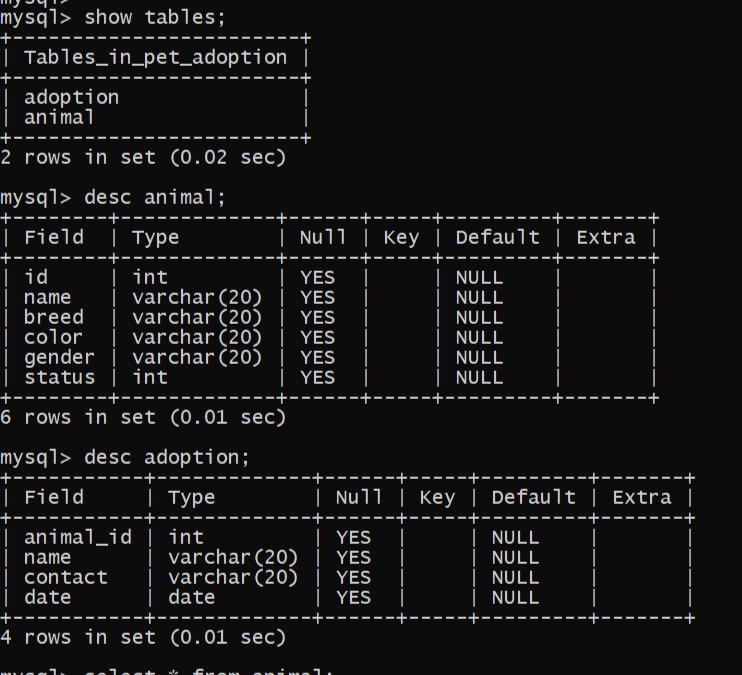
**DCL/TCL (data control language/transactional control language)** enables us to specify who controls our databases with rights and permissions. Like grant, revoke, commit, savepoint.

**HANDS-ON EXERCISE**

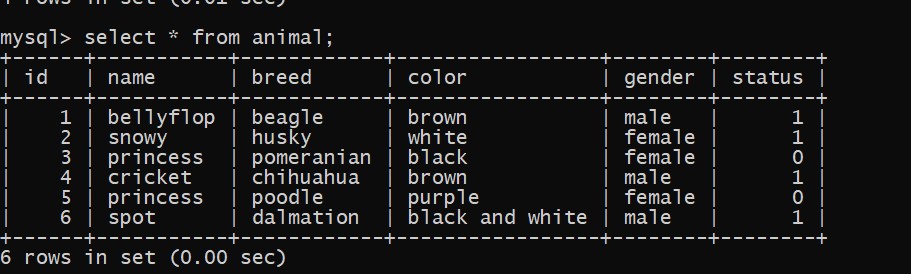
1)Create Database and use it.

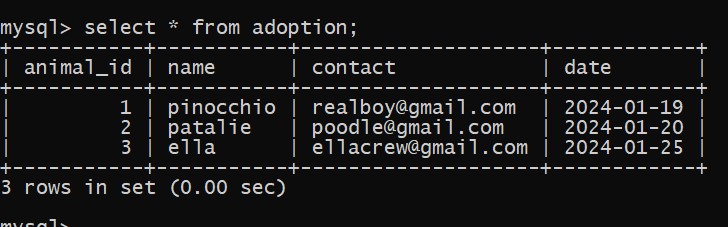


2)create tables

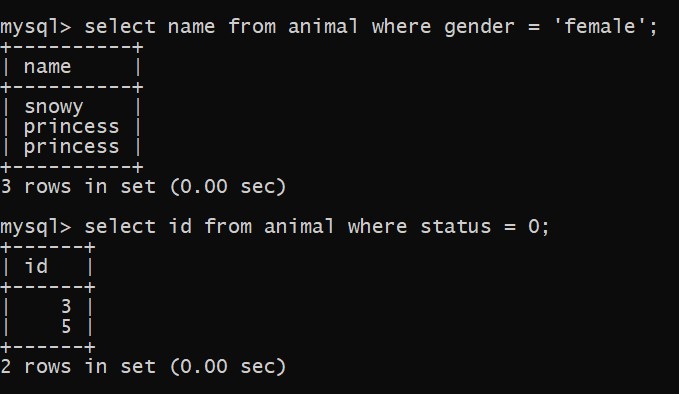


3)Insert values into the tables

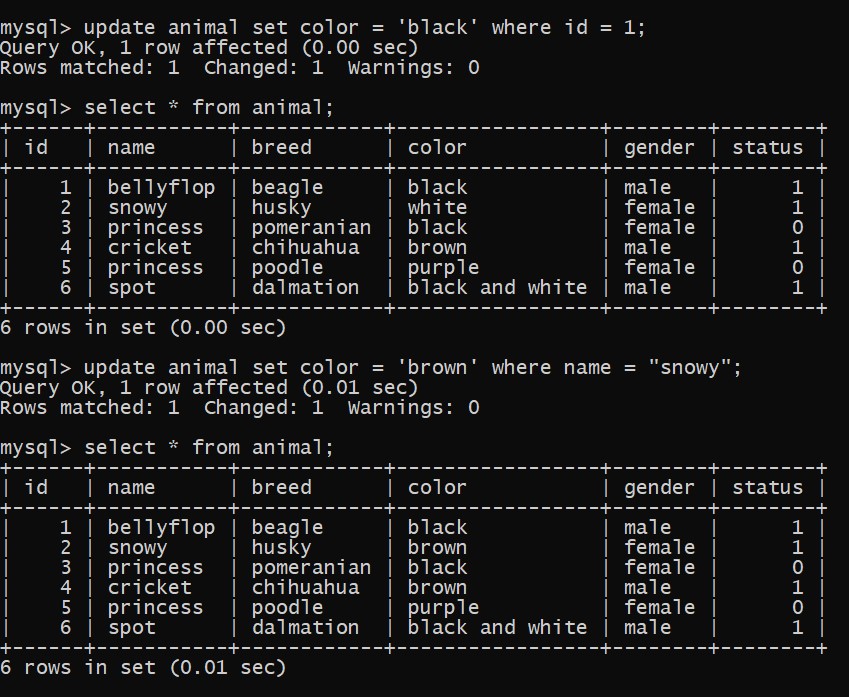




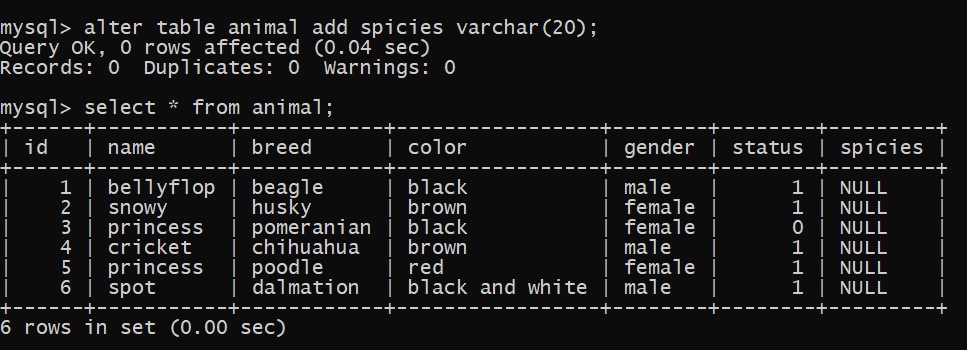
4) Using where clause



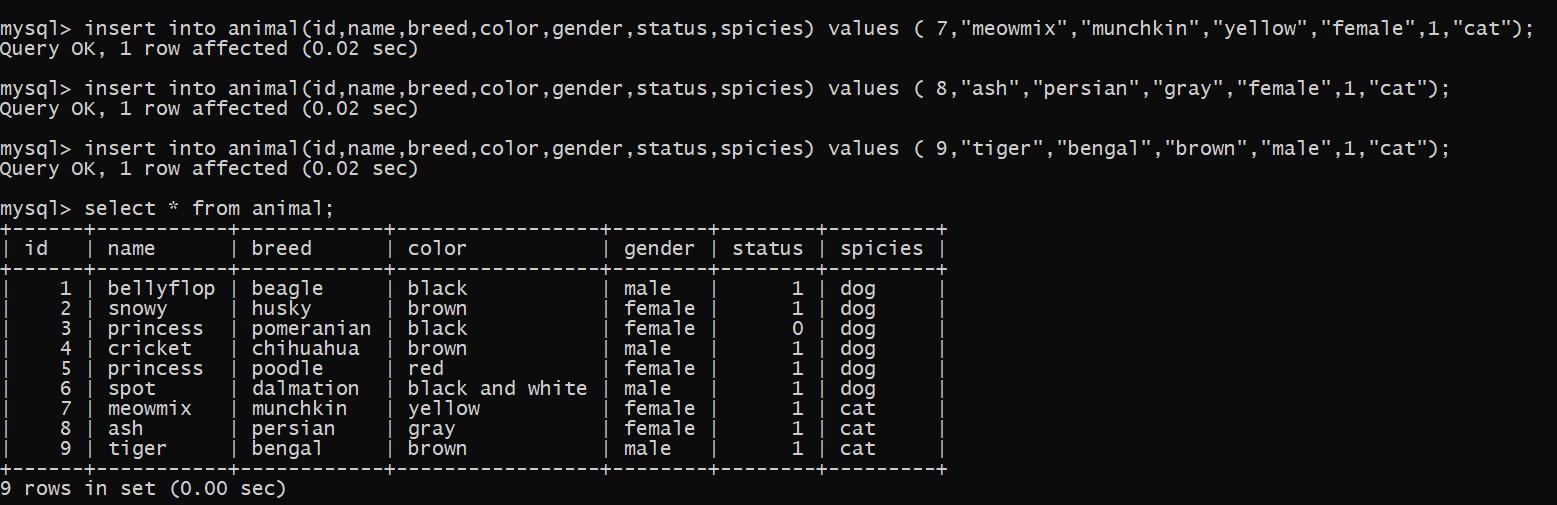
5)Using update command



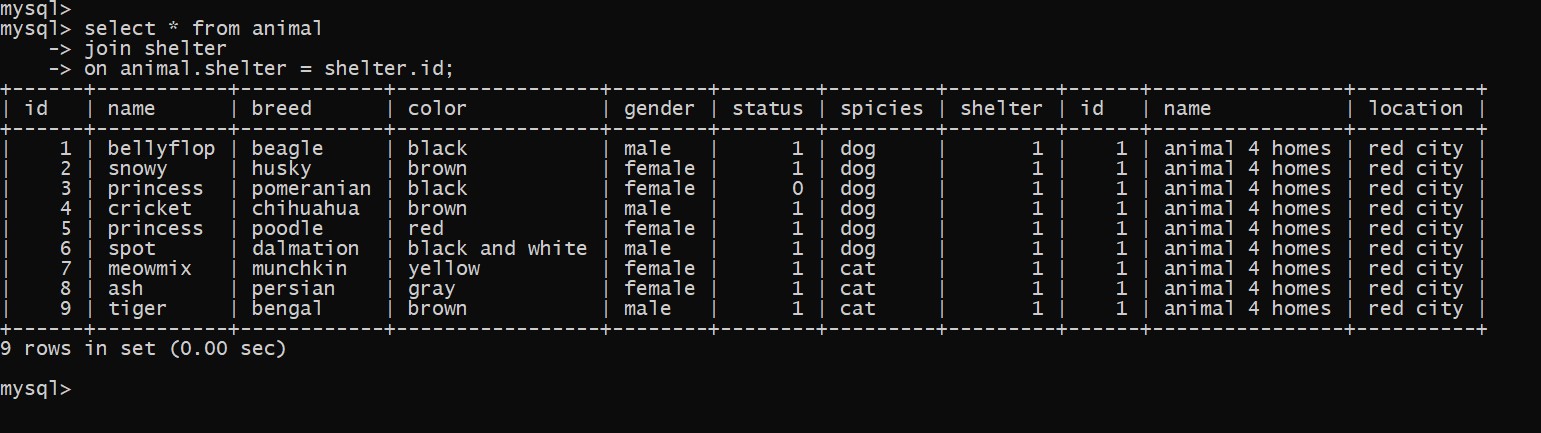
6) Using alter command



7) Insert more column on existing table



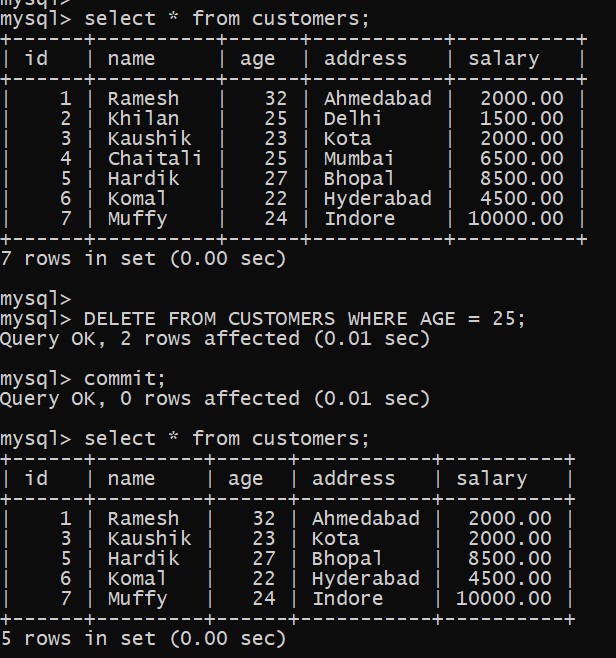
8) applying join



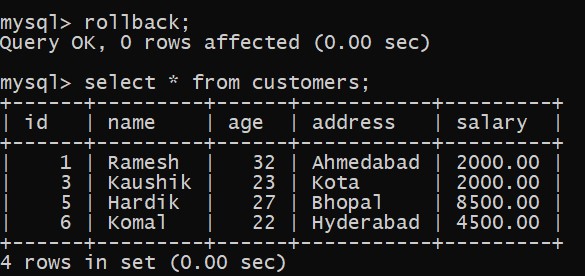
9) Having Clause :

Having Clause is basically like the aggregate function with the GROUP BY clause. The HAVING clause is used instead of WHERE with aggregate functions.

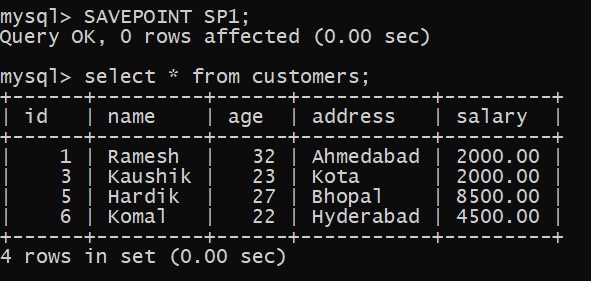
10) COMMIT − to save the changes.



11)ROLLBACK − to roll back the changes.



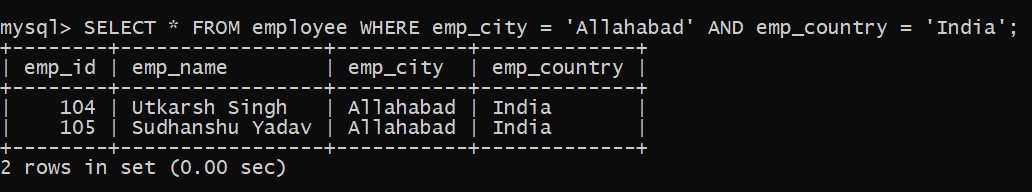
12)SAVEPOINT − creates points within the groups of transactions in which to ROLLBACK.



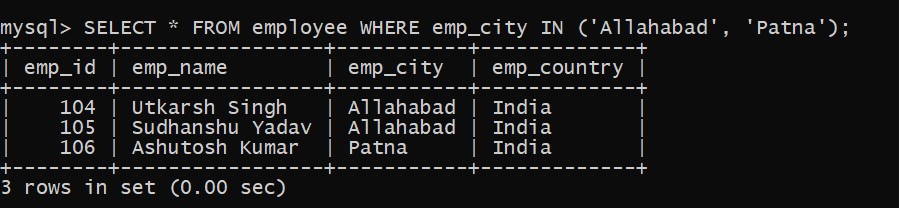
13) SET TRANSACTION − Places a name on a transaction

**Logical Operators in SQL**

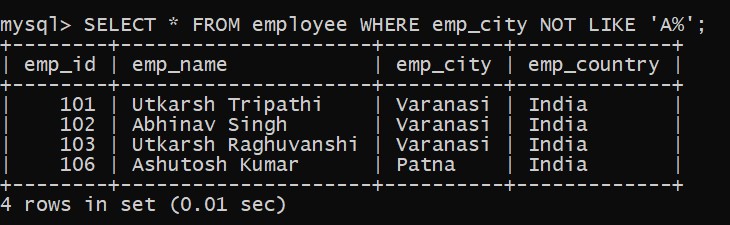
**1)AND -** TRUE if both Boolean expressions are TRUE



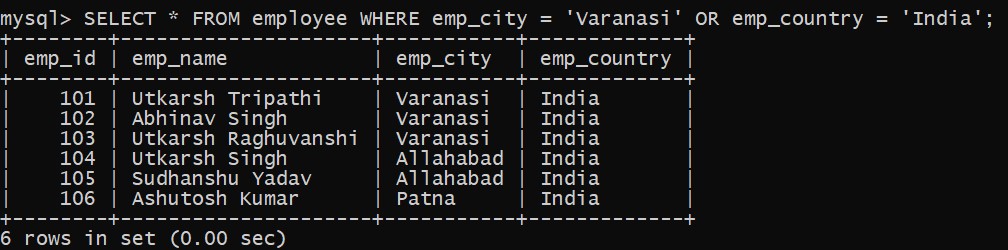
**2)IN -** TRUE if the operand is equal to one of a list of expressions.



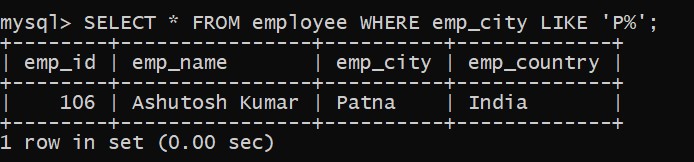
**3)NOT** - Reverses the value of any other Boolean operator.



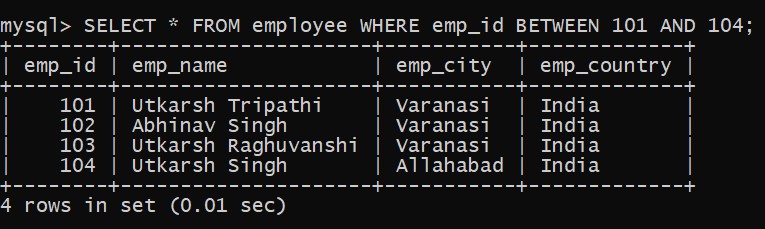
**4)OR** - TRUE if either Boolean expression is TRUE.



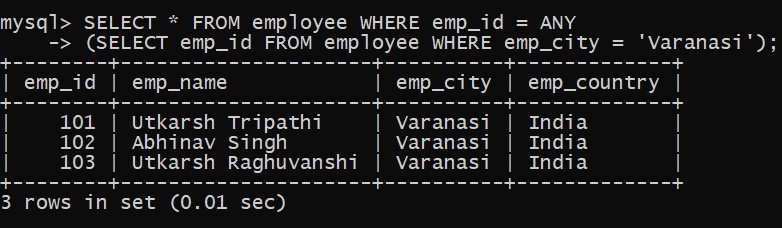
**5)LIKE** - TRUE if the operand matches a pattern.



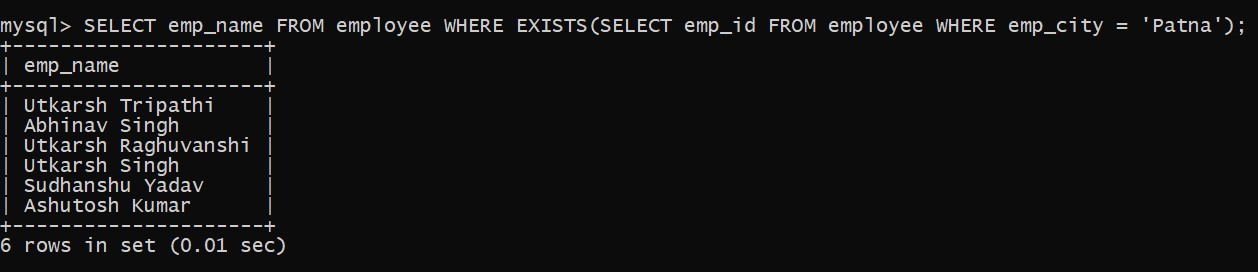
**6)BETWEEN** - TRUE if the operand is within a range.

****

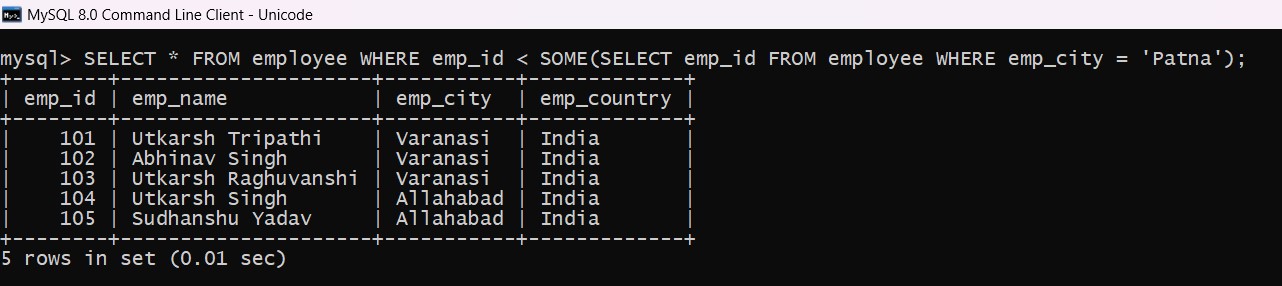
**7) ANY** - TRUE if any one of a set of comparisons is TRUE.



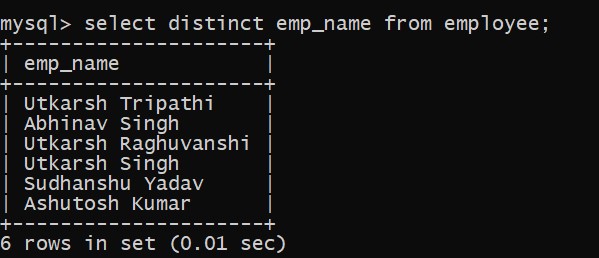
**8)EXISTS** - TRUE if a subquery contains any rows.



**9) SOME** - TRUE if some of a set of comparisons are TRUE



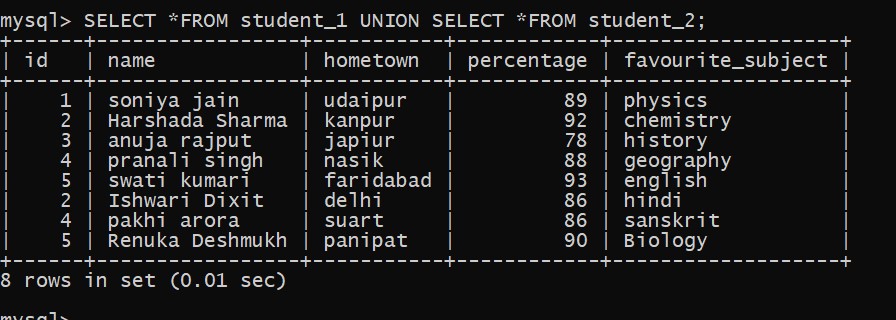
**10) DISTINICT**- The distinct keyword is used in conjunction with the select keyword



**SET Operators in SQL**

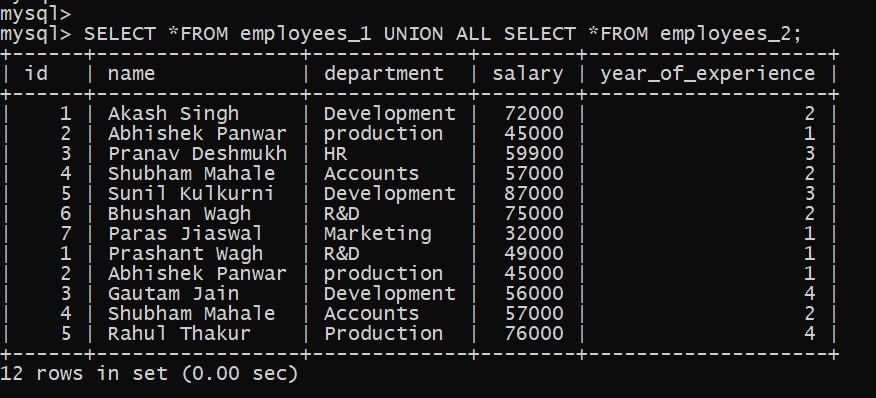
Operators covered under SET operators are:

1. **UNION -** UNION will be used to combine the result of two select statements. Duplicate rows will be eliminated from the results obtained after performing the UNION operation.

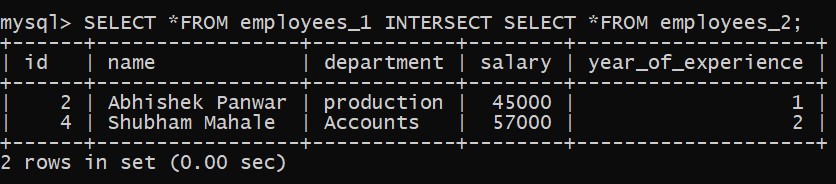


**2)UNION ALL -** This operator combines all the records from both the queries.

Duplicate rows will be not be eliminated from the results obtained after performing the UNION ALL operation.



**3.INTERSECT-** It is used to combine two SELECT statements, but it only returns the records which are common from both SELECT statements



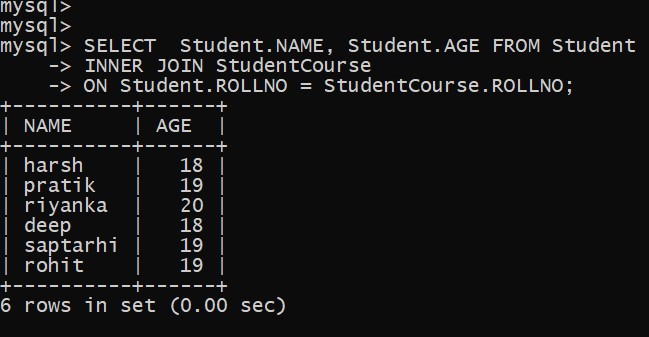
**4.MINUS -** It displays the rows which are present in the first query but absent in the second query with no duplicates.

**SQL | Join (Inner, Left, Right and Full Joins)**

**SQL Join** statement is used to combine data or rows from two or more tables based on a common field between them. Different types of Joins are as follows:

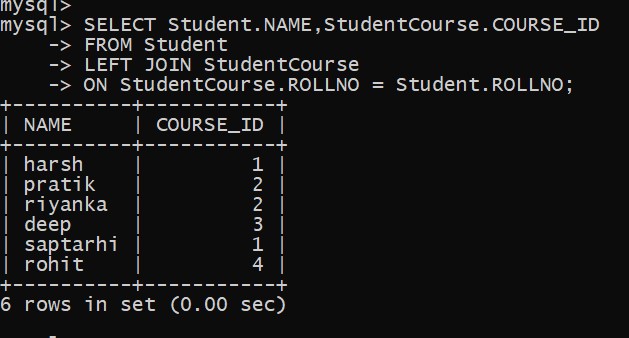
**1)INNER JOIN**

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result-set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.



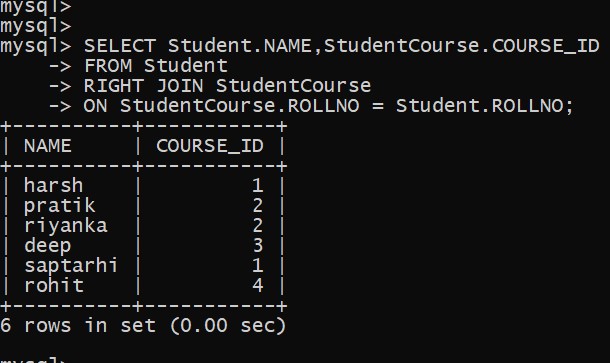
**2)LEFT JOIN**

This join returns all the rows of the table on the left side of the join and matches rows for the table on the right side of the join. For the rows for which there is no matching row on the right side, the result-set will contain *null*. LEFT JOIN is also known as LEFT OUTER JOIN.



**3) RIGHT JOIN**

RIGHT JOIN is similar to LEFT JOIN. This join returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. For the rows for which there is no matching row on the left side, the result-set will contain *null*. RIGHT JOIN is also known as RIGHT OUTER JOIN.

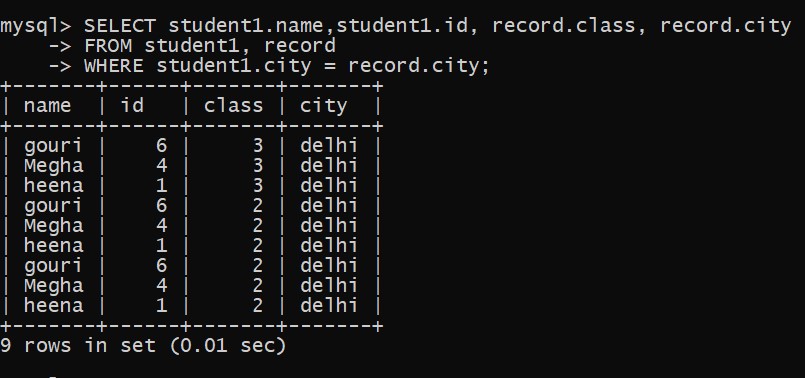


**4)** **FULL JOIN**

FULL JOIN creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain *NULL* values.

1. **EQUI JOIN :**

EQUI JOIN creates a JOIN for equality or matching column(s) values of the relative tables. EQUI JOIN also create JOIN by using JOIN with ON and then providing the names of the columns with their relative tables to check equality using equal sign (=).



1. **NON EQUI JOIN:**

NON EQUI JOIN performs a JOIN using comparison operator other than equal (=) sign like >, <, >=, <= with conditions.

