

Ratna RajyaLaxmi Campus

Pradashani Marga, Bhrikutimandap, Kathmandu

**DATABASE MANAGEMENT SYSTEM**

**CACS 251**

**Submitted By: - Submitted To: -**

Mrs. Kriti Nemkul

**Department of Database Management System**

Pramod gautam

BCA (4th Semester)

Roll No.: - 59

Ratna RajyaLaxmi Campus

Pradashani Marga, Bhrikutimandap, Kathmandu

***Submitted to the:***

Department of Humanities & Social Science, Ratna RajyaLaxmi campus

Pradashani Marga, Bhrikutimandap, Kathmandu

**LAB report on:**

***Database Management System***

**Submitted By: - Submitted To: -**

Mr. Kriti Nemkul

**Department of Database Management System**

Pramod Gautam

BCA (4th Semester)

Roll no.: -59

Submission Date: -2078/ /



Ratna RajyaLaxmi Campus

***Pradashani Marga, Bhrikutimandap, Kathmandu***

We certify that “This report is satisfactory in the Lab project of Database Management System”

**Evaluation Committee**

(Program Director)

Mrs. Kriti Nemkul

Department Of BCA

RR Campus

Pradashani Marga, Kathmandu

Department of Humanities&Socialscience

RR Campus

Pradashani Marga, Kathmandu

(External Examiner) (Internal Examiner)

Date: -2078/ / Date: -2078/ /

Table of Contents:

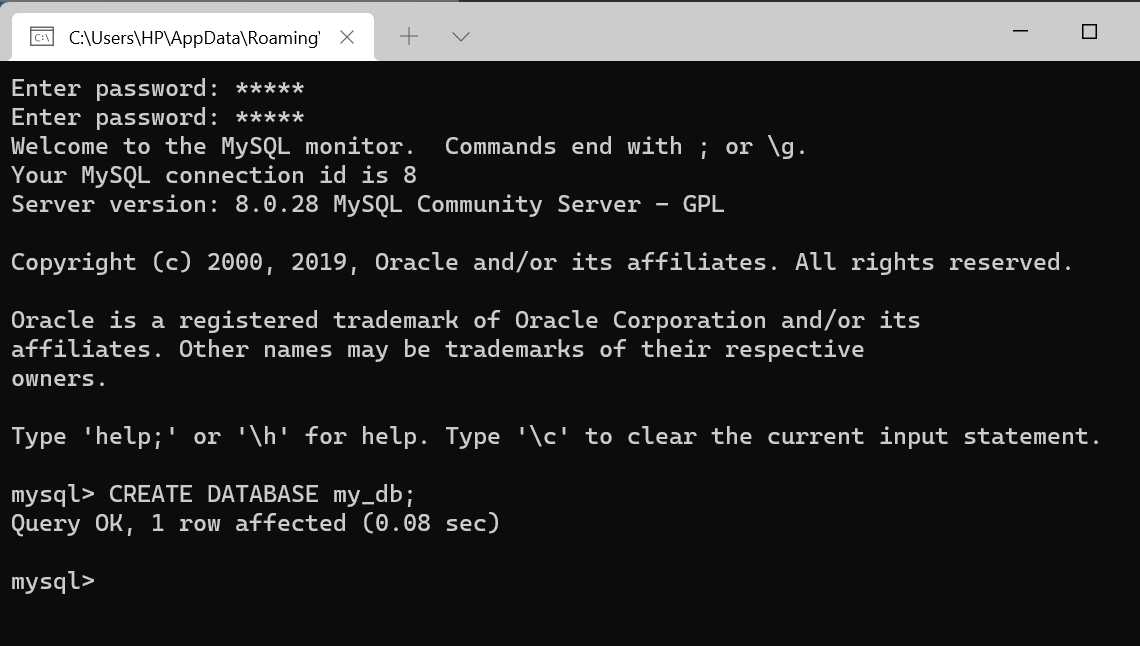
|  |  |  |  |
| --- | --- | --- | --- |
| S.N. | Lab No. | Date | Signature |
| 1. | LAB-1 |  |  |
| 2. | LAB-2 |  |  |
| 3. |  |  |  |
| 4. |  |  |  |

**A. Creating and Altering Databases and Tables**

1. **Create a new database.**

**CREATE DATABASE**

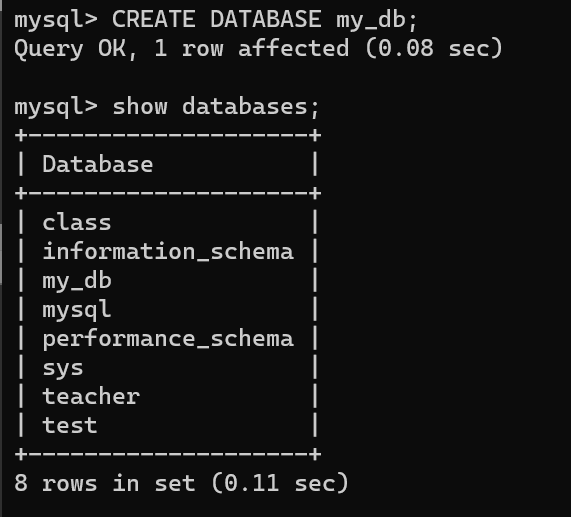
* It is used to create database
* Syntax: CREATE DATABASE database\_name;
* Example: CREATE DATABASE my\_db;



1. **Show all existing databases in your system.**

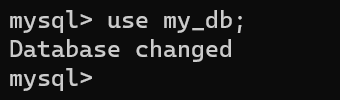
**SHOW DATABASES**

* This Commands is used to view all the databases name.
* Syntax:- SHOW DATABASES;
* Example:- SHOW DATABASES;



**USE**

* This Commands is used to select database which we can use.
* Syntax:- USE <database\_name>;
* Example:- USE my\_db;



1. **Create a new table in the database (table must contain primary key, foreign key, not null constraint and auto increment constraint)**

**CREATE TABLE**

* This Commands is used to .
* Syntax:-

CREATE TABLE table\_name

(

Column\_name1 data\_type (size) [constraints],

Column\_name2 data\_type (size) [constraints],

Column\_name3 data\_type (size) [constraints]

);

* Example:-

create TABLE department

(

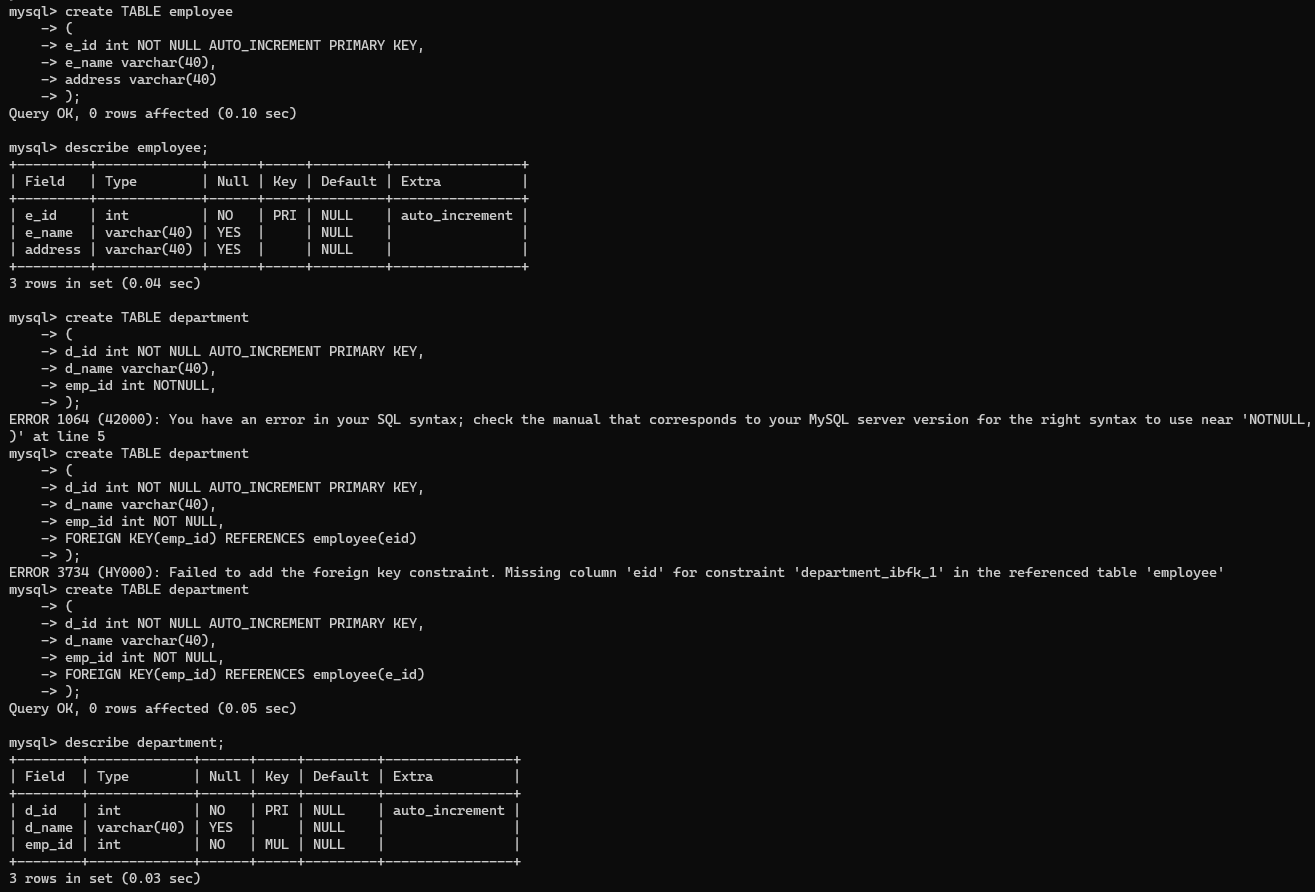
d\_id int NOT NULL AUTO\_INCREMENT PRIMARY KEY,

d\_name varchar(40),

emp\_id int NOT NULL,

FOREIGN KEY(emp\_id) REFERENCES employee(e\_id)

);



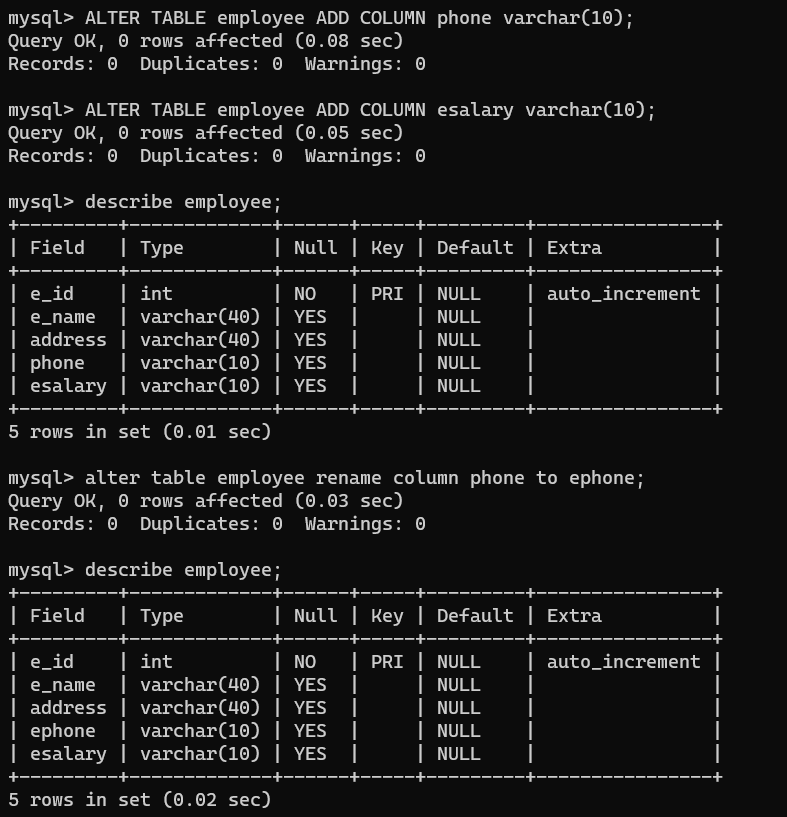
1. **Altering tables (Add column, modify existing column, delete column, rename column)**

**ALTER TABLE**

This command is used to ADD/Change/Modify/Drop existing structure table.

**ADD Column**

* When a new column is to be added to the table structure without constraints.
* Syntax:- ALTER TABLE table\_name ADD COLUMN column\_name datatype(size);
* Example:- ALTER TABLE employee ADD COLUMN phone varchar(10);



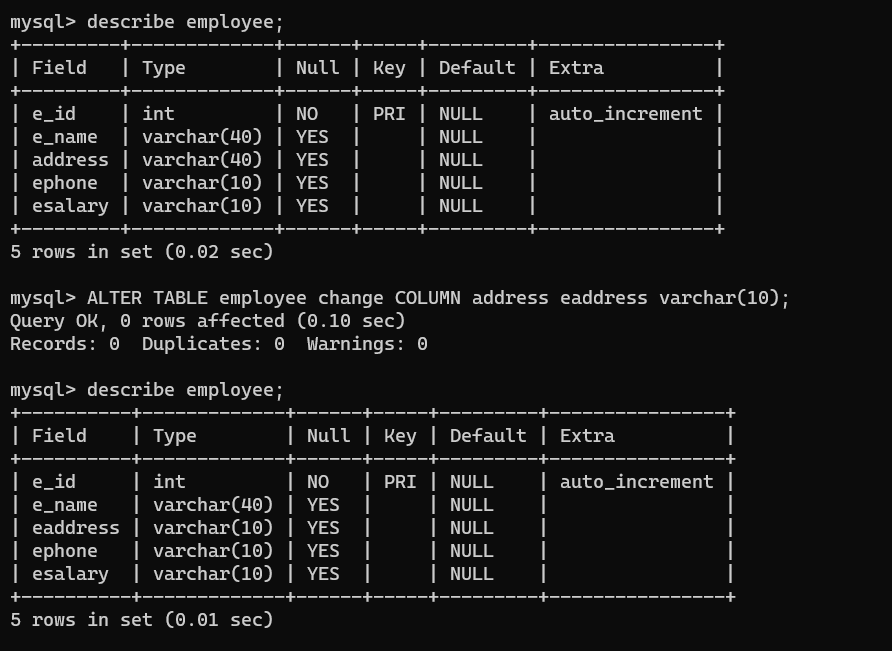
**Change Column**

* This is used to change name and data type of an existing column.
* Syntax:- ALTER TABLE table\_name

CHANGE COLUMN old\_column\_name new\_column\_name new\_data\_type(size);

* Example:- ALTER TABLE employee

CHANGE COLUMN address eaddresss varchar(5)



**MODIFY COLUMN**

* This is used to modify size of the data type or the data type itself of an existing column without changing column name.
* Syntax:-

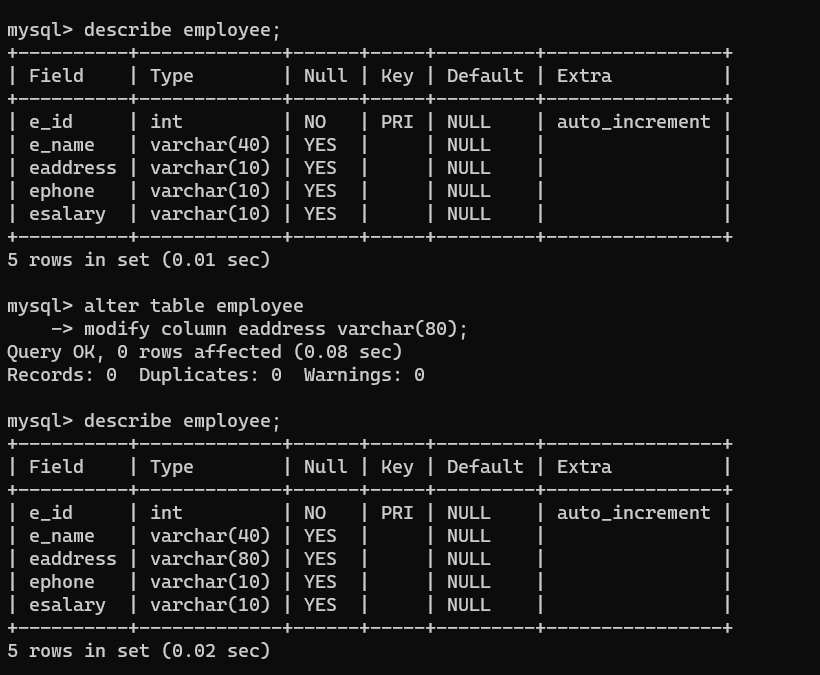
ALTER TABLE table\_name

MODIFY COLUMN column\_name datatype(size);

* Example:-

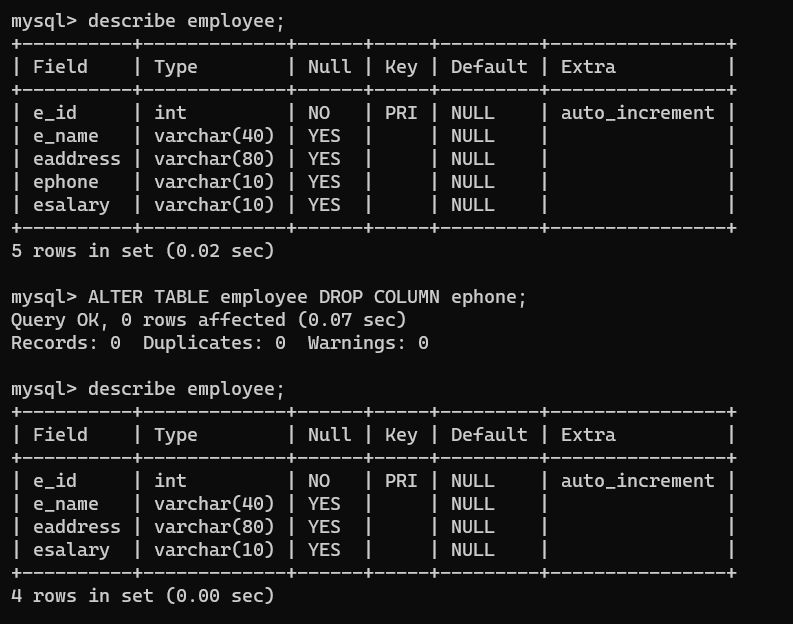
ALTER TABLE employee

MODIFY COLUMN eaddress varchar(80);



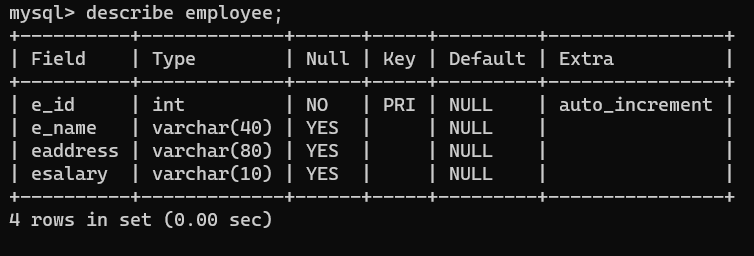
**DROP COLUMN**

* When a column in a table need to delete
* Syntax:- ALTER TABLE table\_name DROP COLUMN column\_name;
* Example: ALTER TABLE employee DROP COLUMN ephone;



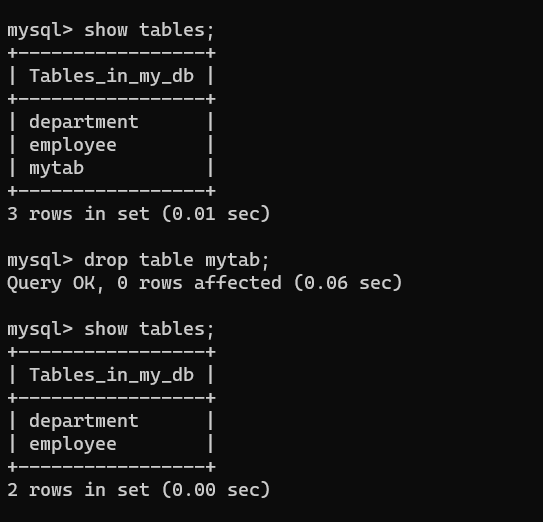
1. **Describe the table structure**

**DESCRIBE**

* This is used to describe table. DESCRIBE only describes structure of table not the information (rows) inside table.
* Syntax:- DESCRIBE table\_name; or DESC table\_name;
* Ex:- DESCRIBE employee;

1. **Drop table**

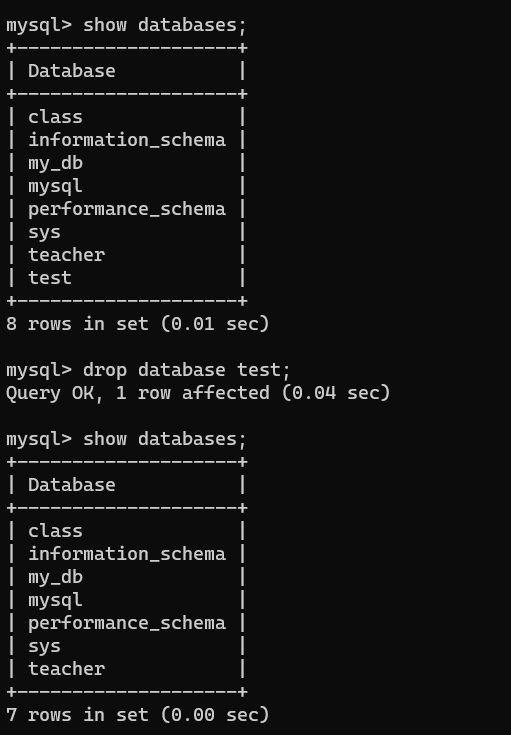
**DROP TABLE**

* This command is used to delete/remove the table from the database.
* Syntax:- DROP TABLE table\_name;
* Example:- DROP TABLE my\_tab;

1. **Drop database**

**DROP DATABASE**

* The DROP DATABASE statement is used to delete a database.
* Syntax:- DROP DATABASE database\_name;
* Ex:- DROP DATABASE test;



**Manipulating and Querying Data**

1. **Adding data into table with insert statement**

**INSERT INTO Statement**

* The INSERT INTO statement is used to insert new records/row/tuple in a table
* Syntax:-

INSERT INTO table\_name (column1, column2, column3, column4, ……….)

VALUES(value1, 'value2', 'value3', value4…..);

* Example:-

INSERT INTO employee(eid, efullname, address)

VALUES(1, 'Ram', 'Ktm');



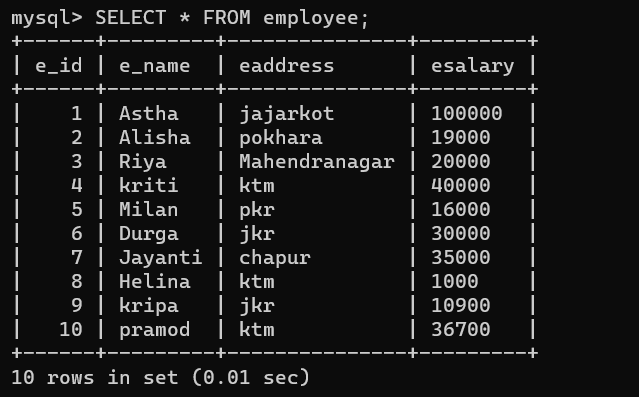
1. **Retrieving data from database using select statement, from clause and filter data with where clause**
   * **Select all data from the table**

**SELECT**

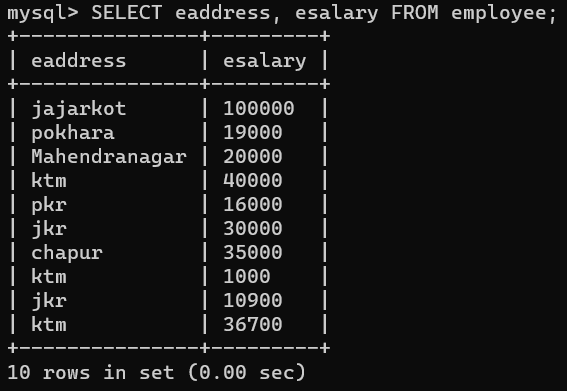
* + The SELECT statement is used to select data from a database and retrieve the information.

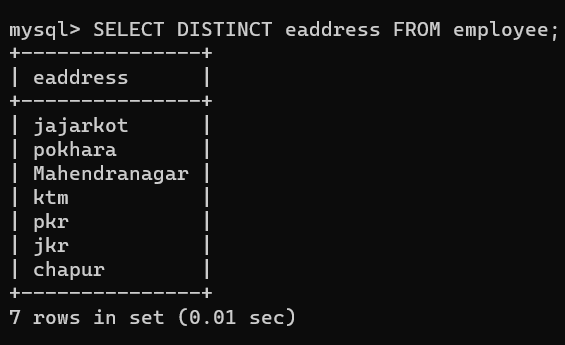
**1. Select all columns from the table**

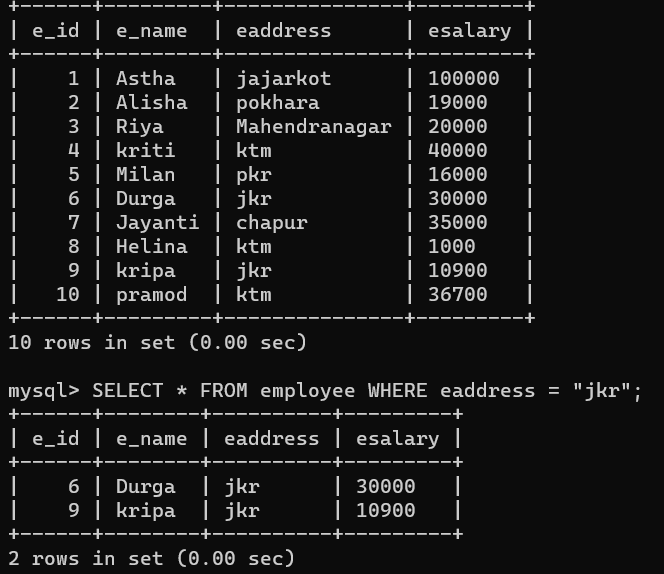
* + Syntax: SELECT \* FROM table\_name;
  + Ex: SELECT \* FROM employee;



* + **Select specific column data from the table**
    - Syntax:- SELECT column\_name1, column\_name2,……. From table\_name;
    - Ex: SELECT eaddress, esalary FROM employee;



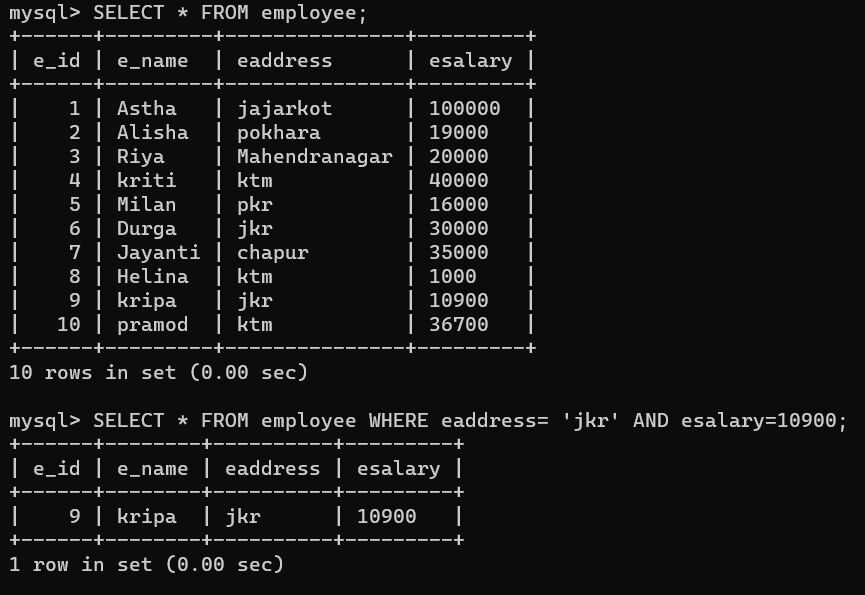
* + **Select distinct data from table using select distinct query**
* SELECT DISTINCT statement is used to return only distinct (different) values.
* Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.
* Syntax: SELECT DISTINCT column1, column2, ……FROM table\_name;
* Ex:- SELECT DISTINCT eaddress FROM employee;
  + **Where clause** 
    - The WHERE clause is used to filter records. It is used to extract only those records that fulfill a specified condition.
    - WHERE Syntax:-
    - SELECT column1, column2, ... FROM table\_name WHERE condition;
    - Example:- SELECT \* FROM employee WHERE eaddress = "jkr";



* + **Use logical connectives (and, or, not) in where clause to retrieve data from table**
* **Logical Operators**
* The Logical operators are those that are true or false. They return a true or false values to combine one or more true or false values.
* The WHERE clause can be combined with AND, OR, and NOT operators.
* **AND Operator**
* The AND operator displays a record if all the conditions separated by AND are TRUE.
* Syntax:-

SELECT column1, column2, ... FROM table\_name WHERE condition1 AND condition2 AND condition3 ...;

* Example: SELECT \* FROM employee WHERE eaddress= 'jkr' AND esalary=10900;

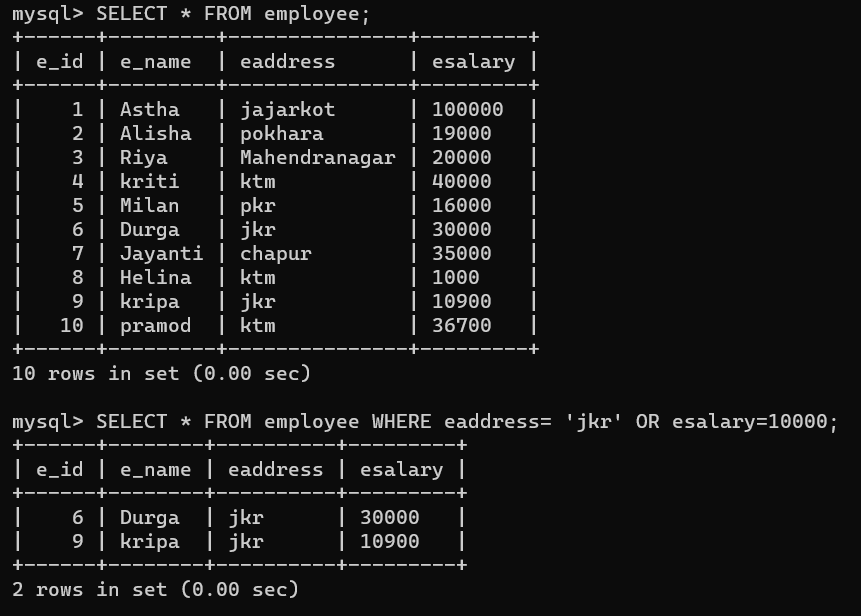


* **OR Operator**
* The OR operator displays a record if any of the conditions separated by OR is TRUE.
* OR Syntax

SELECT column1, column2, ... FROM table\_name WHERE condition1 OR condition2 OR condition3 ...;

* Example:

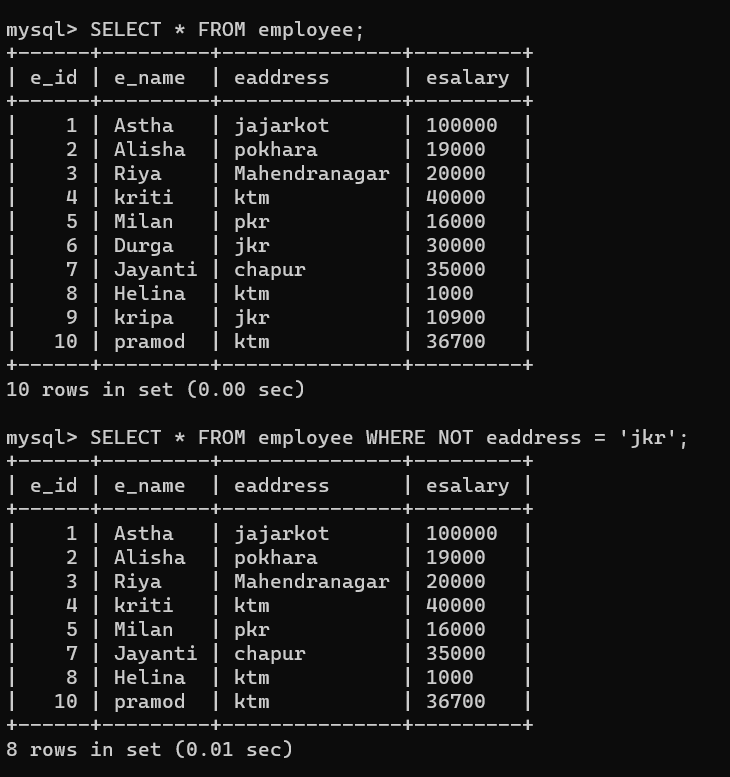
SELECT \* FROM employee WHERE eaddress= 'jkr' OR esalary=10000;



* **NOT Operator**
* The NOT operator displays a record if the condition(s) is NOT TRUE.
* Syntax

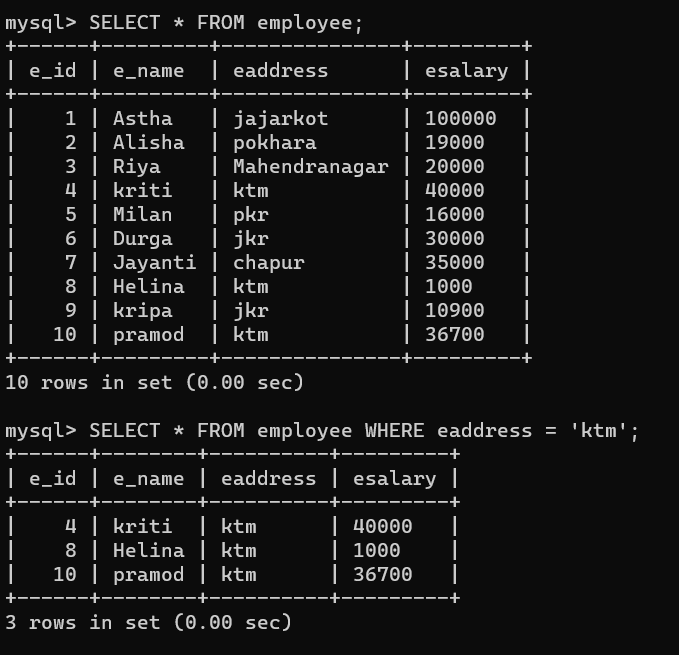
SELECT column1, column2, ... FROM table\_name WHERE NOT condition;

* Example:

SELECT \* FROM employee WHERE NOT eaddress = 'jkr';

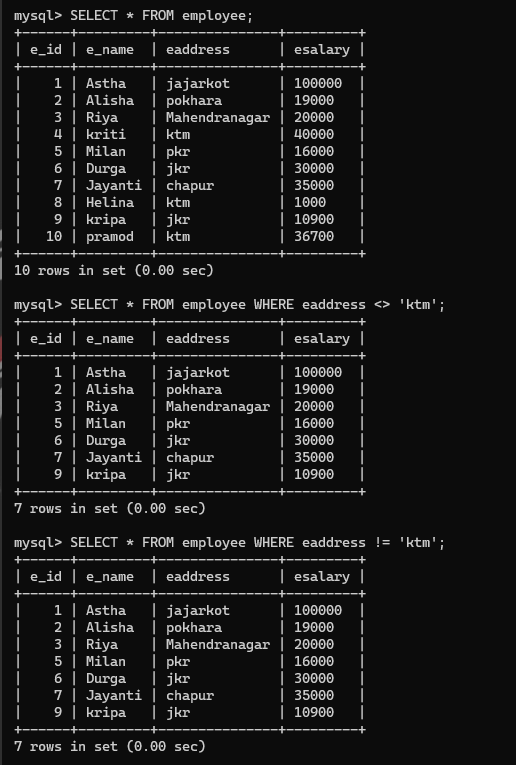
* + Use comparison operators (<, <=, >=, >, =, <>) in where clause to retrieve data from the table
* The comparison operators are used to test for equality and inequality. These operators are used in the WHERE clause to determine which records to select.

**Equals To Operator (=)**

* = operator is used to test for equality in a query.
* Example: SELECT \* FROM employee WHERE eaddress = 'ktm';

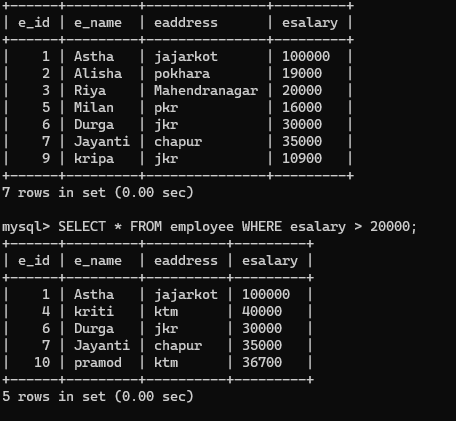
**Inequality Operator (<> or !=)**

* In SQL, there are two ways to test for inequality in a query. We can use either the <> or != operator. Both will return the same results.
* Syntax:
* Example1: SELECT \* FROM employee WHERE eaddress <> 'ktm';
* Example2: SELECT \* FROM employee WHERE eaddress != 'ktm';



**Greater Than Operator (>)**

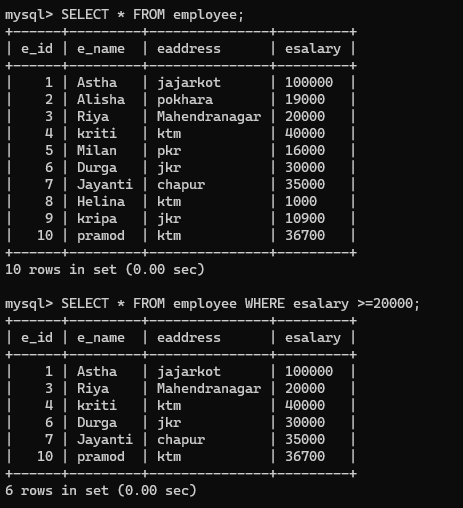
* (>) operator in SQL to test for an expression greater than.
* Example: SELECT \* FROM employee WHERE esalary > 20000;



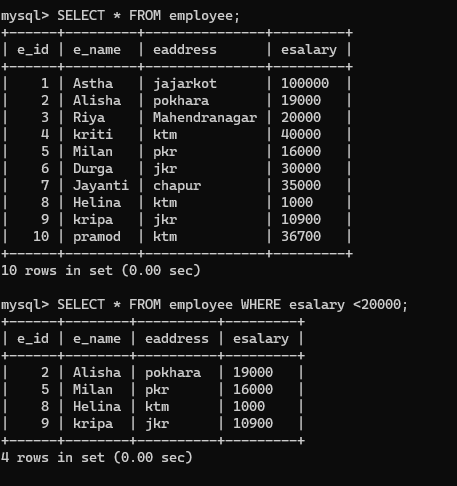
**Greater Than or Equal Operator**

* (>=) operator to test for an expression greater than or equal to.
* Example:

SELECT \* FROM employee WHERE esalary > =20000;

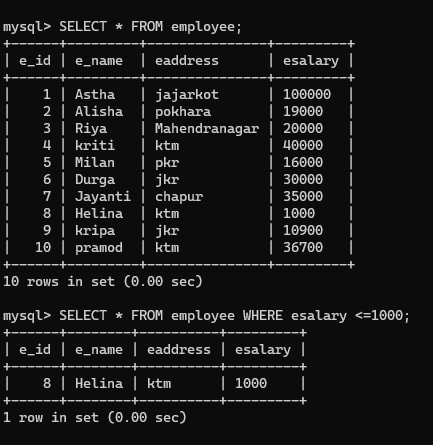


**Less Than (<) Operator**

* + - * We can use the < operator in SQL to test for an expression less than.
      * Example: SELECT \* FROM employee WHERE esalary < 20000;

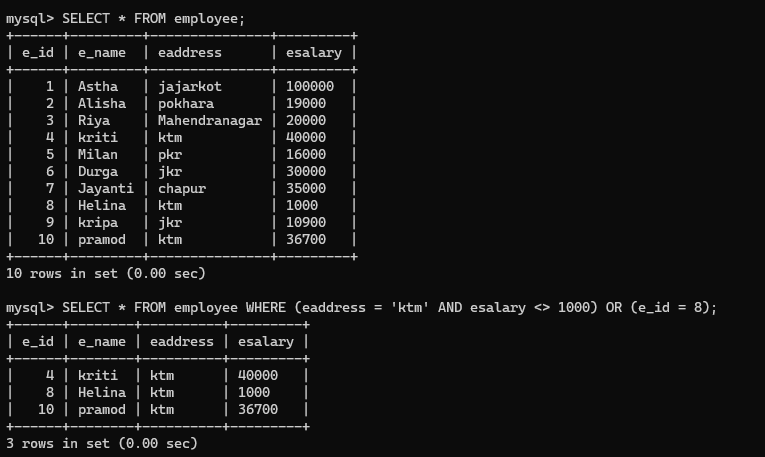
**Less then or equals to (<=) Operator**

* + - * We can use the <= operator to test for an expression less than or equal to.
      * Example:

SELECT \* FROM employee WHERE salary <=1000;

* + Combine and and or operator to retrieve data from table
* Example:

SELECT \* FROM employee WHERE (eaddress = 'ktm' AND esalary <> 1000) OR (e\_id = 8);



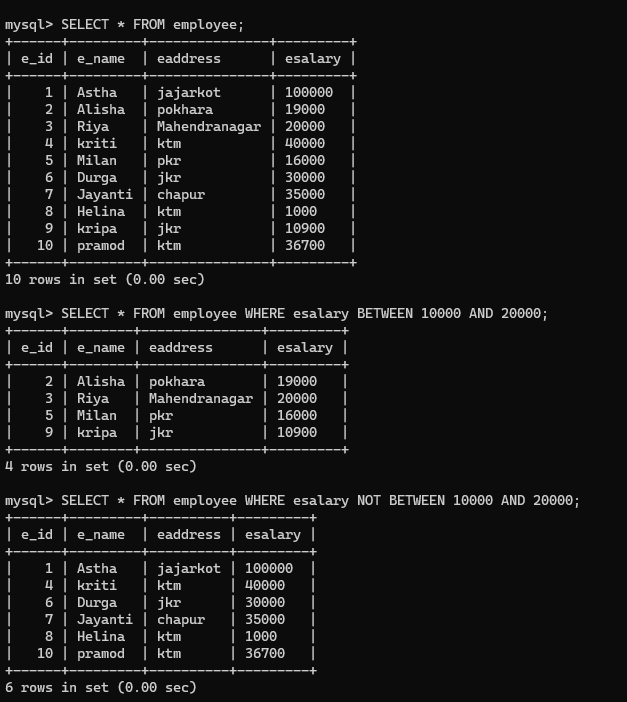
* + Use between and not between operator

BETWEEN operator is used to easily test if an expression is within a range of values (inclusive). It can be used in a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax: expression BETWEEN value1 AND value2;

Example:

SELECT \* FROM employee WHERE esalary BETWEEN 10000 AND 20000;

SELECT \* FROM employee WHERE esalary NOT BETWEEN 10000 AND 20000;

* + - **Use in and not in operator**

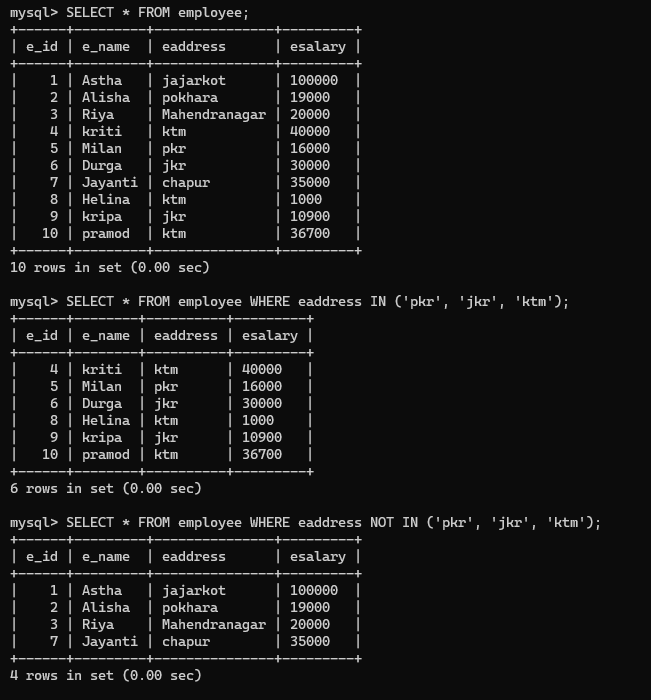
IN condition (sometimes called the IN operator) allows to easily test if an expression matches any value in a list of values. It is used to help reduce the need for multiple OR conditions in a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax: expression IN (value1, value2, .... value\_n);

Example:

SELECT \* FROM employee WHERE eaddress IN ('Dang', 'Rolpa', 'Pyuthan');

SELECT \* FROM employee WHERE eaddress NOT IN ('Dang', 'Rolpa', 'Pyuthan');



* + - **Use order by statement to order data in ascending and descending order**

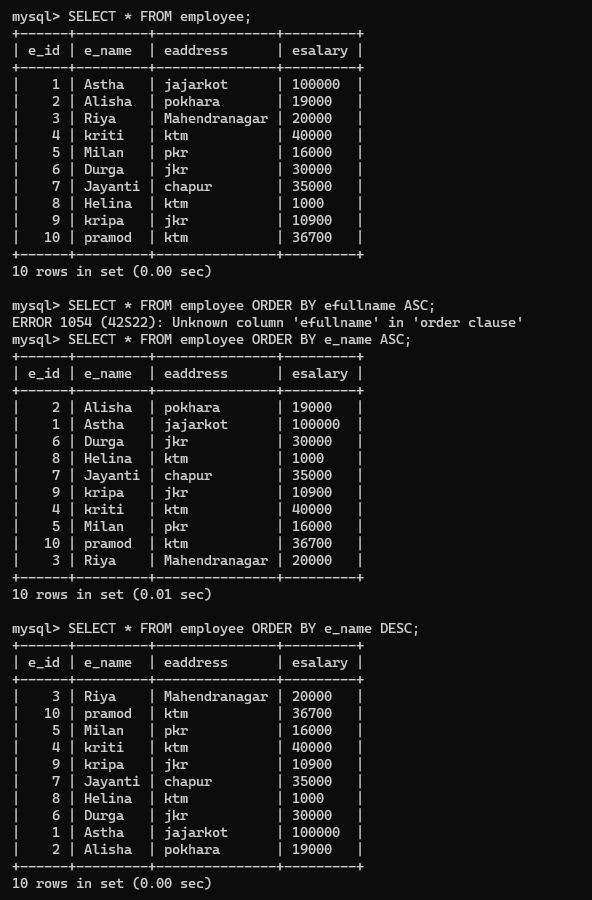
The ORDER BY keyword is used to sort the result-set in ascending or descending order. The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

Syntax: SELECT column1, column2, ... FROM table\_name ORDER BY column1, column2, ... ASC|DESC;

Example:

SELECT \* FROM employee ORDER BY efullname ASC;

SELECT \* FROM employee ORDER BY ename DESC;



* + - Use null and not null condition in where clause to select null data and not null data

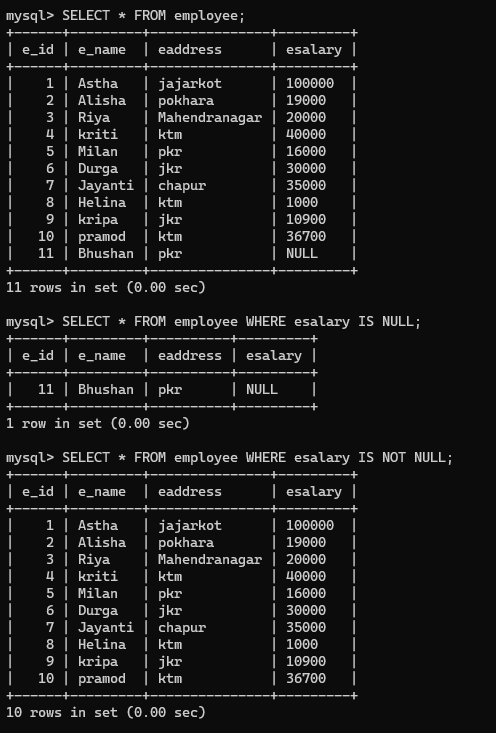
The IS NULL condition is used in SQL to test for a NULL value. It returns TRUE if a NULL value is found, otherwise it returns FALSE. It can be used in a SELECT, INSERT, UPDATE, or DELETE statement.

Syntax: expression IS NULL

Example:

SELECT \* FROM employee WHERE esalary IS NULL;

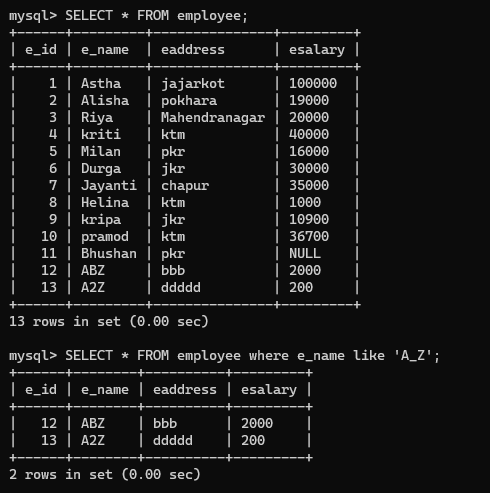
SELECT \* FROM employee WHERE esalary IS NOT NULL;

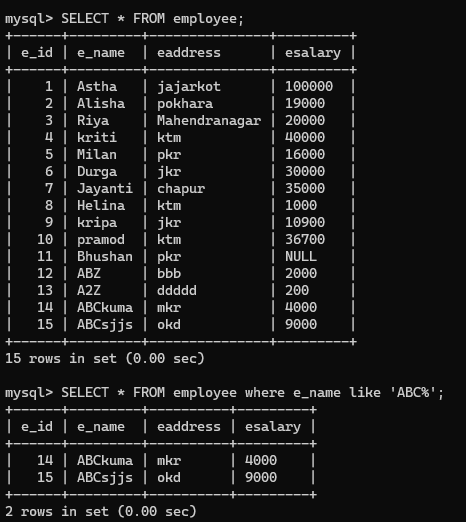


* + - Show String operation using like comparison operator

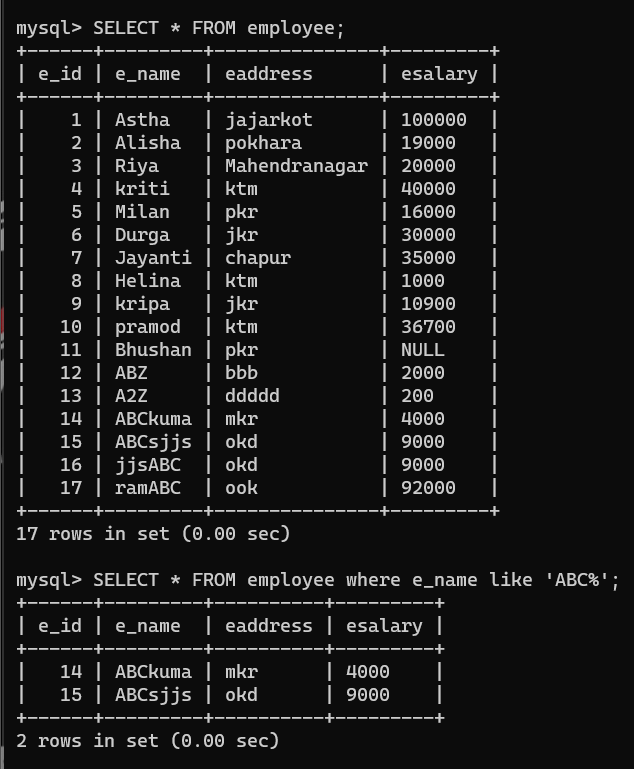
Eg.

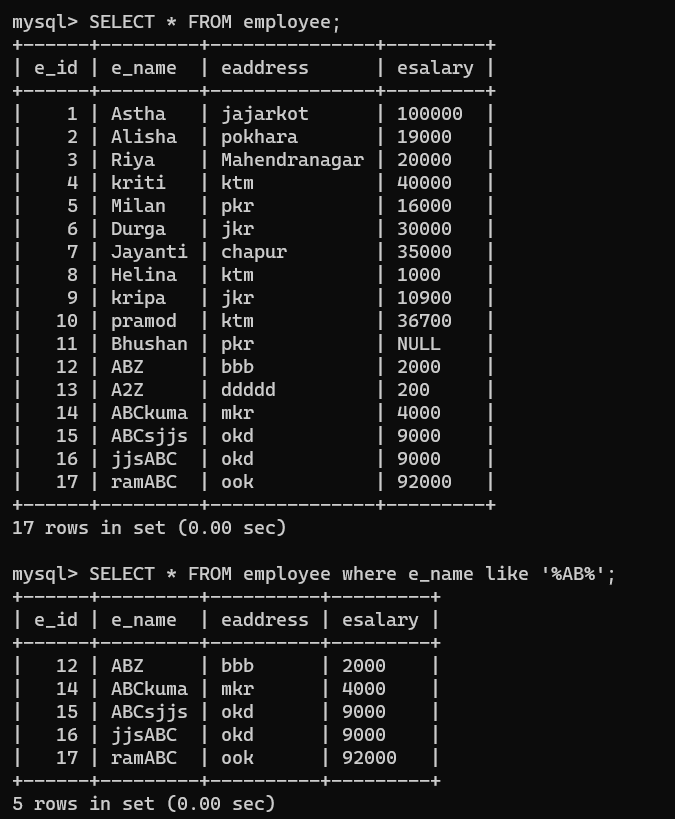
• ‘A\_Z‘: All string that starts with ‘A‘, another character and end with ‘Z‘. For example, ‘ABZ’ and ‘A2Z’ both satisfy this condition but ‘ABHZ’ does not because between A and Z there are two characters are present instead of one.



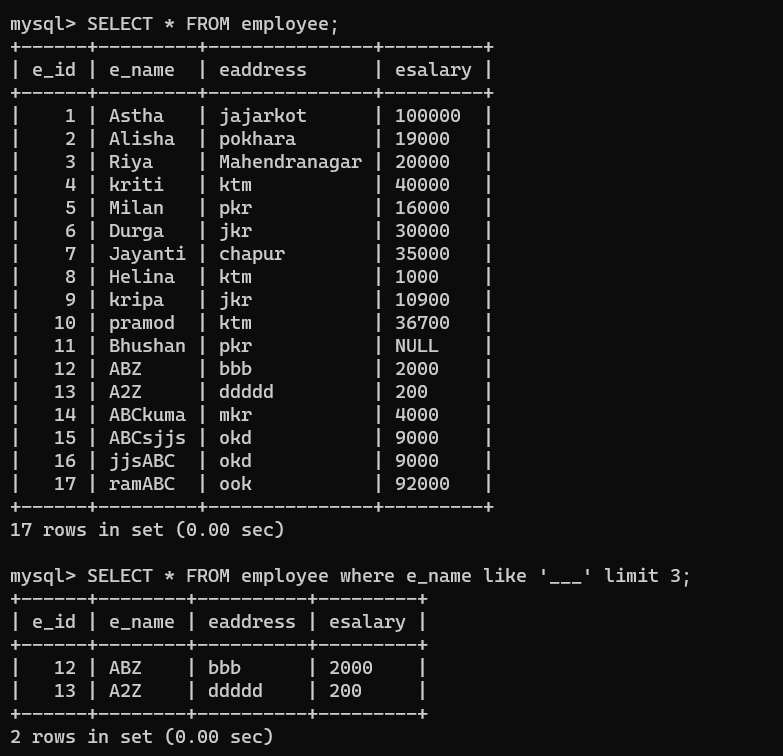
• ‘ABC%’: All strings that start with ‘ABC’.

• ‘%ABC’: All strings that ends with ‘ABC’.

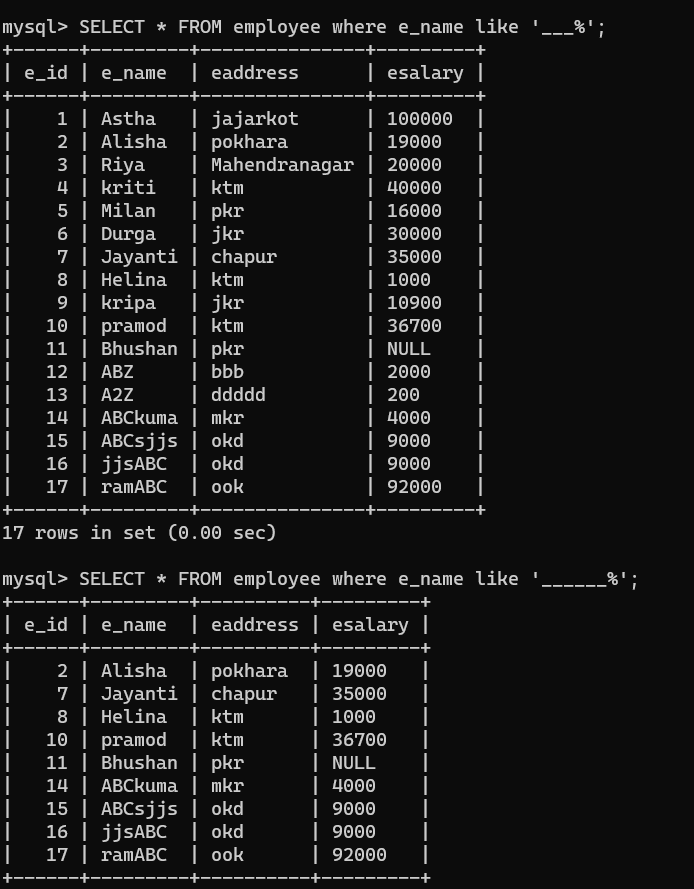


• ‘%AN%’: All strings that contains the pattern ‘AN’ anywhere. For example, ‘ANGELS’ ,‘SAN’, ‘FRANCISCO’ etc.

• ‘\_ \_ \_’: matches any strings of exactly three characters.



• ‘\_ \_ \_ \_ \_%’: matches any strings of at least five characters



* + - Use limit clause
    - GROUP BY Aggregate Functions: COUNT, MAX, MIN, AVG, SUM
    - Having clause

1. Modify Data – use Update query to update the date of the table
2. Deleting rows