My-SQL

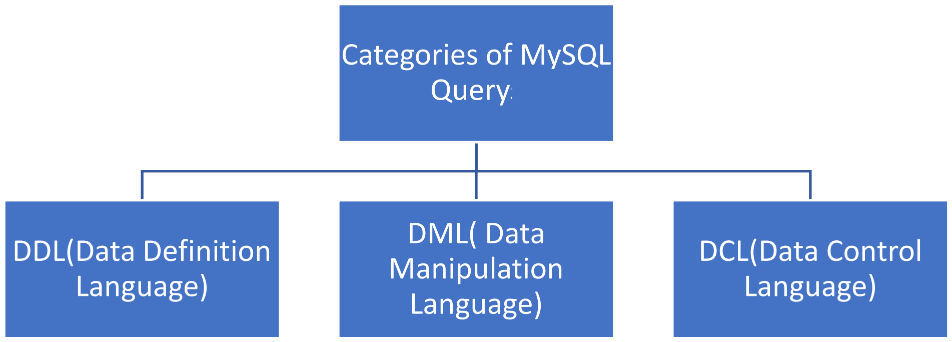
SQL: SQL stands for Structured Query Language. It is the language used to manipulate and manage databases and tables. Some of the important advantages of SQL are:

- **High speed:** SQL can be used to retrieve large amounts of records from a database quickly and efficiently.
- **Easy to learn and understand:** SQL generally consists of English statements and as such, it is very easy to learn and understand. Besides, it does not require much coding unlike in programming languages.
- **Portability**: SQL can be used in laptop, PCs, server and even some mobile phones.
- **Interactive Language**: SQL is a domain language used to communicate with the database. With the help of SQL answers to the complex questions can be received in seconds.

SQL queries/commands creates an interface between the user and database and helps to create, transform and retrieve information from Relational Database Management Systems.

CATEGORIES OF SQL QUERIES

SQL queries can be categorized into three categories:


Categories of MySQL Query
DDL(Data Definition Language)
DML(Data Manipulation Language)
DCL(Data Control Language)

Categories of SQL queries

DDL (Data Definition Language): All the queries which are used to create, destroy, or restructure databases and tables come under this category. Examples of DDL queries are - CREATE, DROP, ALTER.

DML (Data Manipulation Language): All the queries which are used to manipulate data within tables come under this category. Examples of DML queries are - INSERT, UPDATE, DELETE.

DCL (Data Control Language): All the queries which are used to control the access to databases and tables fall under this category. Examples of DCL queries are - GRANT, REVOKE.

DATA TYPES IN MySQL

Every data stored in SQL database has certain data type. MySQL uses many different data types broken into three categories –

- Numeric
- Character
- Date

Numeric data types are mainly used to store number with or without fraction part. The most commonly used numeric data types are:

1. INT/INTEGER
2. DECIMAL
3. FLOAT

INT/INTEGER: The int data type is the integer data type in SQL. This is used to store integer number (without any fraction part). We can specify a width of upto 11 digits.
FLOAT: This data type is used to store number with fraction part (real numbers).
DECIMAL: The decimal data type has fixed precision and scale.
SYNTAX: DECIMAL[(p[, s])]
where:

- 'p' is the precision or the total number of significant decimal digits, where the most significant digit is the left-most nonzero digit and the least significant digit is the right-most known digit.
- 's' is the scale or the number of digits from the decimal point to the least significant digit.
- Square brackets ([ ]) are optional.

EXAMPLE: Decimal(6,2) means maximum 6 digits can be there with 2 digits after decimal.

Character data types are used to store character data. They can store all alphanumeric values. The character data types are:

1. CHAR
2. VARCHAR
3. VARCHAR2

CHAR: CHAR is used for storing fix length character strings between 1 to 255 characters in length.. If this type is used to store variable length strings, it will waste a lot of disk space because it always allocates fixed memory.
SYNTAX: CHAR(10)
It can store fixed-length character string having maximum length 10.

VARCHAR: Varchar is a variable character string. If we declare data type as VARCHAR, then it will
occupy space for NULL values. It can have maximum of 2000 characters.
SYNTAX: VARCHAR (10)
It can store variable-length character string having maximum length 10.

VARCHAR2: VARCHAR2 is used to store variable length character strings. The string value's length will be stored on disk with the value itself. VARCHAR2 can store up to 4000 bytes of characters.
SYNTAX: VARCHAR2 (10)
It can store variable-length character string having maximum length 10.

EXAMPLE

Name char (15): If we store Name as "Kavita", then first six places of the fifteen characters are filled with Kavita and the remaining 9 spaces are filled with spaces. The size of name is always fifteen.
Name varchar (15): If we store Name as "Kavita", then first six places of the fifteen characters are filled with Kavita and the remaining 9 spaces are filled with NULL.
Name varchar2 (15): If we store Name as "Kavita", then first six places of the fifteen characters are filled with Kavita.

DATE data type is used to store valid date values, which is ranging from 1000-01-01 and 9999-
12-31. The valid date formats are: YYYY-MM-DD or DD/MON/YY or YYYY/MM/DD or MM/DD/YY or DD-MON-YYYY. Default format is: 'YYYY-MM-DD'

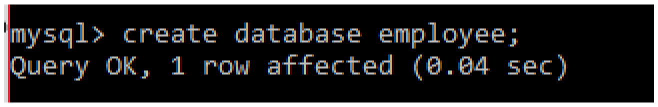
SYNTAX: DATE

EXAMPLE: 1945-11-22, 1998/05/15

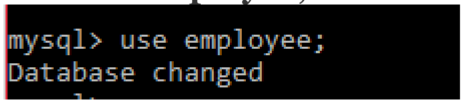
Before storing data in a database, we need to first create a database and then open it. Let us understand commands to create and open a database.

CREATING AND OPENING A DATABASE

- **Create database**: This query is used to create a database.
  Q. Write a query to create a database *employee*.
  And. create database employee;


```
mysql> create database employee;
Query OK, 1 row affected (0.04 sec)
```

- **Use <database name>:** This query is used to open the database.
  Q. Write a query to open the database *employee*.
  Ans. **use employee;**


```
mysql> use employee;
Database changed
```

Note: Please put semicolon after completing SQL query.

After creating a database next step is to create tables to store data.

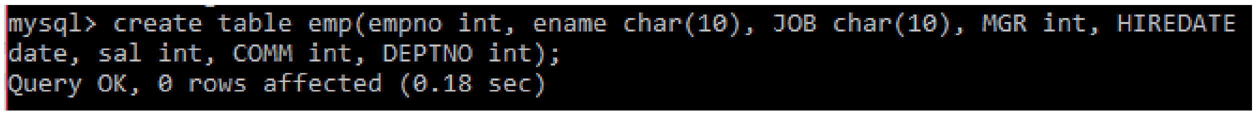
CREATING A TABLE

- **create table():** This query is used to create a table. In this query we give the names of all attributes along with their data types.

  SYNTAX:
  CREATE TABLE <TableName>(<ColumnName1> <Data Type1>, <ColumnName2> <Data Type2>,... ,<ColumnNameN> <Data TypeN>);

  NOTE: Before creating a table decide upon the data you are going to store in the table. Also think of the data types of the data which will be stored in the table.

  Q Write a query to create a table emp having columns employee number as integer, name as character(10), JOB as character(10), MGR as integer, Hire Date as date, salary as integer, commission as integer and depart number as integer.
  Ans. **create table emp(empno int, ename char(10), JOB char(10), MGR int, hiredate date, Sal int, comm int, deptno int);**

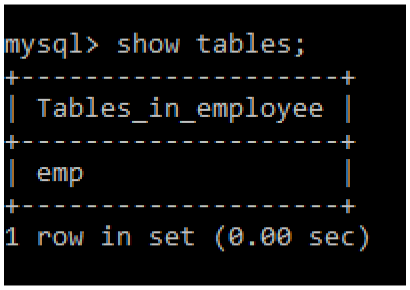

```
mysql> create table emp(empno int, ename char(10), JOB char(10), MGR int, HIREDATE
date, sal int, COMM int, DEPTNO int);
Query OK, 0 rows affected (0.18 sec)
```

  NOTE: Name of table and name of attribute should not have space in between. Also, they cannot have special characters. Only underscore is allowed. For eg.

  Invalid names: emp name, class-XII, grade*
  Valid names: emp_name, class_XII, grade

- To verify that the table has been created, give the query:

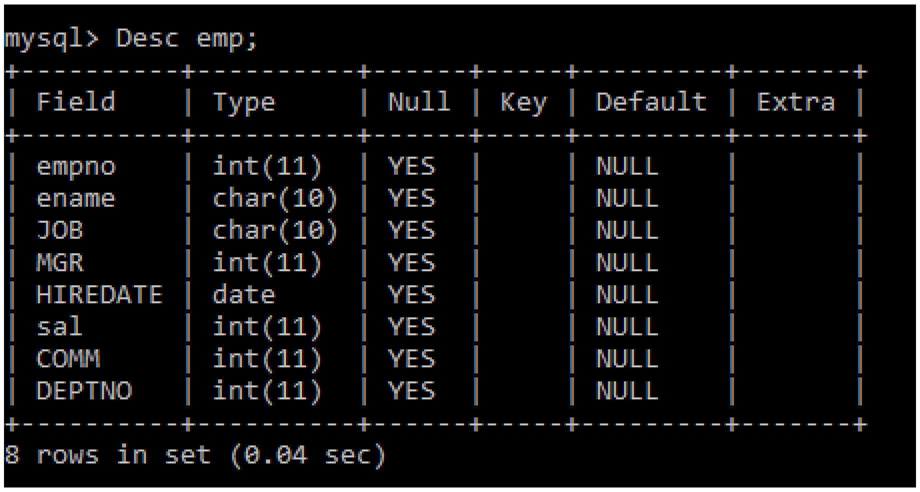
Show tables;



```
mysql> show tables;
+--------------------+
| Tables_in_employee |
+--------------------+
| emp                |
+--------------------+
1 row in set (0.00 sec)
```

The above command will display all the tables present in a current database.

- We can **view the structure of the table** emp by giving the query:
  Desc emp;



```
mysql> Desc emp;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| empno    | int(11)  | YES  |     | NULL    |       |
| ename    | char(10) | YES  |     | NULL    |       |
| JOB      | char(10) | YES  |     | NULL    |       |
| MGR      | int(11)  | YES  |     | NULL    |       |
| HIREDATE | date     | YES  |     | NULL    |       |
| sal      | int(11)  | YES  |     | NULL    |       |
| COMM     | int(11)  | YES  |     | NULL    |       |
| DEPTNO   | int(11)  | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
8 rows in set (0.04 sec)
```

Now that we have created a table, space for storing data has been reserved. Next step is to store data/records in this table.

ADDING CONSTRAINTS

Constraints can be easily added at the time of creation of tables. Let us discuss different types of constraints:

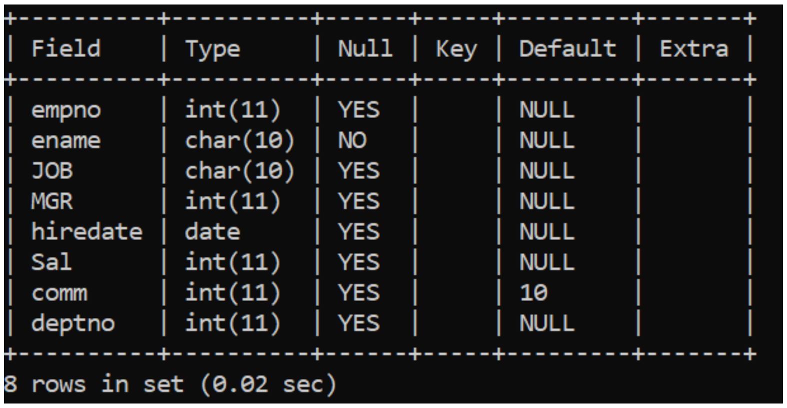
NOT NULL: This constraint when applied to an attribute makes sure that the attribute value is not NULL. For example, the name of the employee cannot be NULL. Hence NOT NULL constraint can be specified in this case.

```
create table emp
( empno int,
ename char(10) NOT NULL,
JOB char(10),
MGR int,
hiredate date,
```

Sal int,
comm int,
deptno int);

DEFAULT: If a user does not enter a value for an attribute, then default value specified while creating the table is used. For example, if the ***comm*** is not entered, then by default it should store 10.

create table emp2
(empno int,
ename char(10) NOT NULL,
JOB char(10),
MGR int,
hiredate date,
Sal int,
comm int DEFAULT 10,
deptno int);



```
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| empno    | int(11)  | YES  |     | NULL    |       |
| ename    | char(10) | NO   |     | NULL    |       |
| JOB      | char(10) | YES  |     | NULL    |       |
| MGR      | int(11)  | YES  |     | NULL    |       |
| hiredate | date     | YES  |     | NULL    |       |
| Sal      | int(11)  | YES  |     | NULL    |       |
| comm     | int(11)  | YES  |     | 10      |       |
| deptno   | int(11)  | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
8 rows in set (0.02 sec)
```

CHECK: It is used to restrict the values of an attribute so that they fall within a range. For example, salary of employee must not exceed 50000. This can be specified as follows:

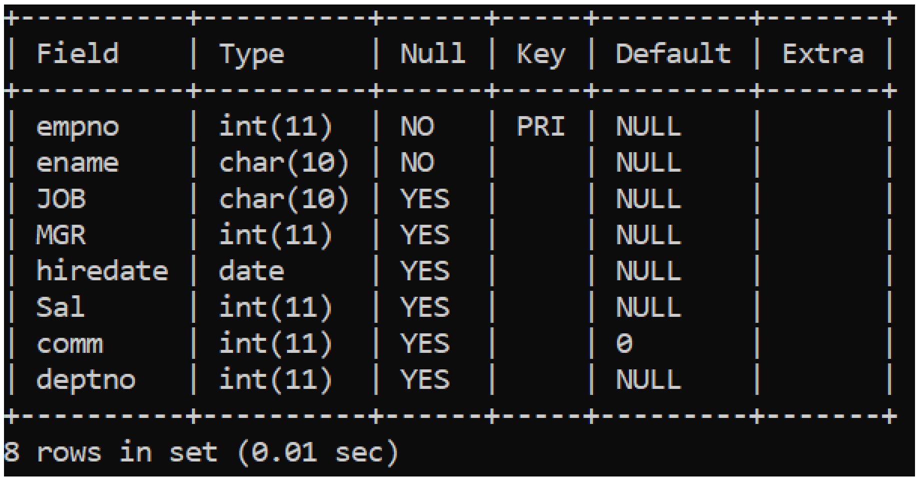
create table emp
(empno int,
ename char(10) NOT NULL,
JOB char(10),
MGR int,
hiredate date,
Sal int CHECK (Sal<=50000),
comm int DEFAULT 10,
deptno int);

Note: MySQL does not support check constraint.

KEY CONSTRAINT: Primary Key of a table can be specified in two ways.

If the ***primary key of the table consists of a single attribute***, then the corresponding attribute can be declared as primary key along with its description. For example, if empno attribute of the EMP relation is the PRIMARY KEY then it can be specified as follows:

```
create table emp
( empno int PRIMARY KEY,
ename char(10) NOT NULL,
JOB char(10),
MGR int,
hiredate date,
Sal int ,
comm int DEFAULT 0,
deptno int);
```



```
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| empno    | int(11)  | NO   | PRI | NULL    |       |
| ename    | char(10) | NO   |     | NULL    |       |
| JOB      | char(10) | YES  |     | NULL    |       |
| MGR      | int(11)  | YES  |     | NULL    |       |
| hiredate | date     | YES  |     | NULL    |       |
| Sal      | int(11)  | YES  |     | NULL    |       |
| comm     | int(11)  | YES  |     | 0       |       |
| deptno   | int(11)  | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
8 rows in set (0.01 sec)
```

As shown above empno has value NO in the Null column (primary key cannot be NULL) and PRI in the key column.

However, if ***primary key contains more than one attribute*** then it must be specified separately as a list of attributes it comprises of, within parenthesis, separated by commas. For example, suppose the primary key of the EMP relation comprises of EMPNO and ENAME.

```
create table emp
( empno int,
ename char(10),
JOB char(10),
MGR int,
hiredate date,
Sal int ,
comm int DEFAULT 10,
deptno int
PRIMARY KEY (empno, ename)
);
```

By default, Primary keys are NOT NULL and there is no need to mention this constraint separately.

REFERENTIAL INTEGRITY CONSTRAINT- This constraint is specified by using the foreign key clause. This clause contains the foreign key and the primary key referred to by this foreign key along with the name of the relation. For example consider the following tables:

EMP (empno, ename, JOB, MGR, hiredate, Sal, comm, deptno)
DEPARTMENT (Dept_ID, Dept_Name, Location)

In this example, ***deptno*** is the foreign key in EMP table that references ***Dept_ID*** of DEPARTMENT relation which is a primary key.

The SQL command for creating these tables would be as follows:

```
CREATE TABLE Department
(
Dept_ID INTEGER PRIMARY KEY,
Dept_Name VARCHAR(20) ,
Location VARCHAR(10)
);


CREATE TABLE EMP
( empno int PRIMARY KEY,
ename char(10),
JOB char(10),
MGR int,
hiredate date,
Sal int ,
comm int DEFAULT 10,
deptno int
FOREIGN KEY (deptno) REFERENCES DEPARTMENT(Dept_ID)
);
```

The foreign key is created separately by using the key words FOREIGN KEY followed by the attribute that is the foreign key within parenthesis, then the keyword REFERENCES followed by the name of the referred relation and its primary key within parenthesis.

Note: The referenced table must be created before it is referred in any other table. You can also specify the action to be taken in case the foreign key attribute or the primary key attribute values are changed as that may result in violation of the referential integrity constraint. This may be done using the following commands:

1. ON DELETE
2. ON UPDATE

Actions that can be taken are as follows:
1. SET NULL
2. CASCADE
3. RESTRICT

Consider the following cases:

CASE I

```
CREATE TABLE EMP
( empno int PRIMARY KEY,
ename char(10),
JOB char(10),
MGR int,
hiredate date,
Sal int ,
comm int DEFAULT 10,
deptno int
FOREIGN KEY (deptno) REFERENCES DEPARTMENT(Dept_ID) ON
DELETE SET NULL ON UPDATE SET NULL
);
```

Thus if a department with a given value of **Dept_ID** is deleted in DEPARTMENT table, then the corresponding tuples that contains the deleted value for deptno attribute in EMP table would be set to NULL. Similarly, if **Dept_ID** value is updated then also the corresponding attribute in EMP table would be set to NULL.

CASE II

```
CREATE TABLE EMP
( empno int PRIMARY KEY,
ename char(10),
JOB char(10),
MGR int,
hiredate date,
Sal int ,
comm int DEFAULT 10,
deptno int
FOREIGN KEY (deptno) REFERENCES DEPARTMENT(Dept_ID)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

Thus if a department with a given value of **Dept_ID** is deleted in DEPARTMENT table, then the corresponding tuples that contains the deleted value for deptno attribute in EMP table would be deleted. Similarly, if **Dept_ID** value is updated then the change in corresponding value is also reflected in EMP table.

CASE III

```
CREATE TABLE EMP
( empno int PRIMARY KEY,
ename char(10),
JOB char(10),
MGR int,
hiredate date,
Sal int ,
comm int DEFAULT 10,
deptno int
FOREIGN KEY (deptno) REFERENCES DEPARTMENT(Dept_ID)
ON DELETE RESTRICT ON UPDATE RESTRICT
);
```

The RESTRICT option will reject the delete or update operation for the referenced table if there are one or more related foreign key values in a referencing table, i.e, you cannot delete or update a department if there are departments who belong to a particular employee.

SELF-REFERENCING TABLES: When a foreign key constraint references column within the same table then this table is known as self-referencing table. For example, consider a table Emp. Because the manager is also an employee, there is a foreign key relationship between the MGR and Empno as shown below:

```
CREATE TABLE EMP
( empno int PRIMARY KEY,
ename char(10) NOT NULL,
JOB char(10),
MGR int,
hiredate date,
Sal int ,
comm int DEFAULT 10,
deptno int
FOREIGN KEY (MGR) REFERENCES emp (empno)
);
```

NAMING A CONSTRAINT: We can name the constraints in the Create Table command. The advantage of naming is that named constraints can be easily deleted or updated using the Alter Table command. A constraint can be named by using the keyword

CONSTRAINT followed by the name of the constraint and its specification. For example consider the following Create Table command:

```
CREATE TABLE EMP
( empno int,
ename char(10),
JOB char(10),
MGR int,
hiredate date,
Sal int ,
comm int DEFAULT 10,
deptno int
CONSTRAINT EMP_PK PRIMARY KEY (empno),
CONSTRAINT EMP_FK FOREIGN KEY (deptno) REFERENCES
DEPARTMENT(Dept_ID) ON DELETE RESTRICT ON UPDATE
RESTRICT
);
```

In the above table, the primary key constraint is named as **EMP_PK** and the foreign key constraint is named as **EMP_FK**.

DROP TABLE

This command is used to delete tables. For example,

DROP TABLE EMP;

If the table has constraints then CASCADE and RESTRICT options can be used along with the drop command. For eg. Suppose you want to drop the Emp table having primary key and Foreign key constraints then the command would be:

DROP TABLE EMP CASCADE;

In this case the EMP table would be dropped and with the CASCADE option, i.e. all the constraints that refer this table would also be automatically dropped.

However, if the requirement is that the table should not be dropped if it is being referenced in some other table then RESTRICT option can be used as shown below:

DROP TABLE EMP RESTRICT;

Note: In MySQL server 5.6.20, RESTRICT and CASCADE options are not supported though they are permitted to make porting easier.

ALTER TABLE

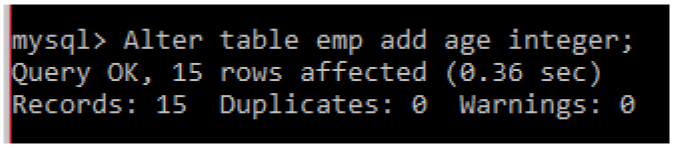
ADDING, MODIFYING AND DELETING COLUMNS

ALTER TABLE command is used is used to modify the structure of the table. It is used to add, modify and delete columns from a table.

ADDING COLUMNS

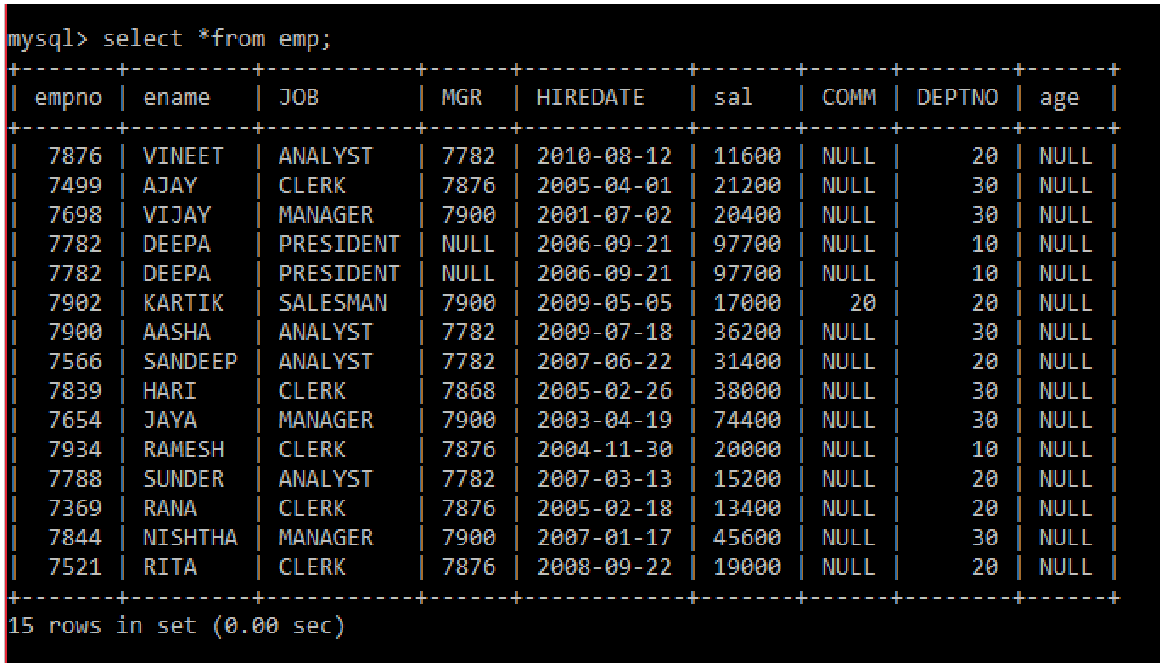
Q. WAC to add a new column age of type integer to the table emp;

Ans. **Alter table emp add age integer;**



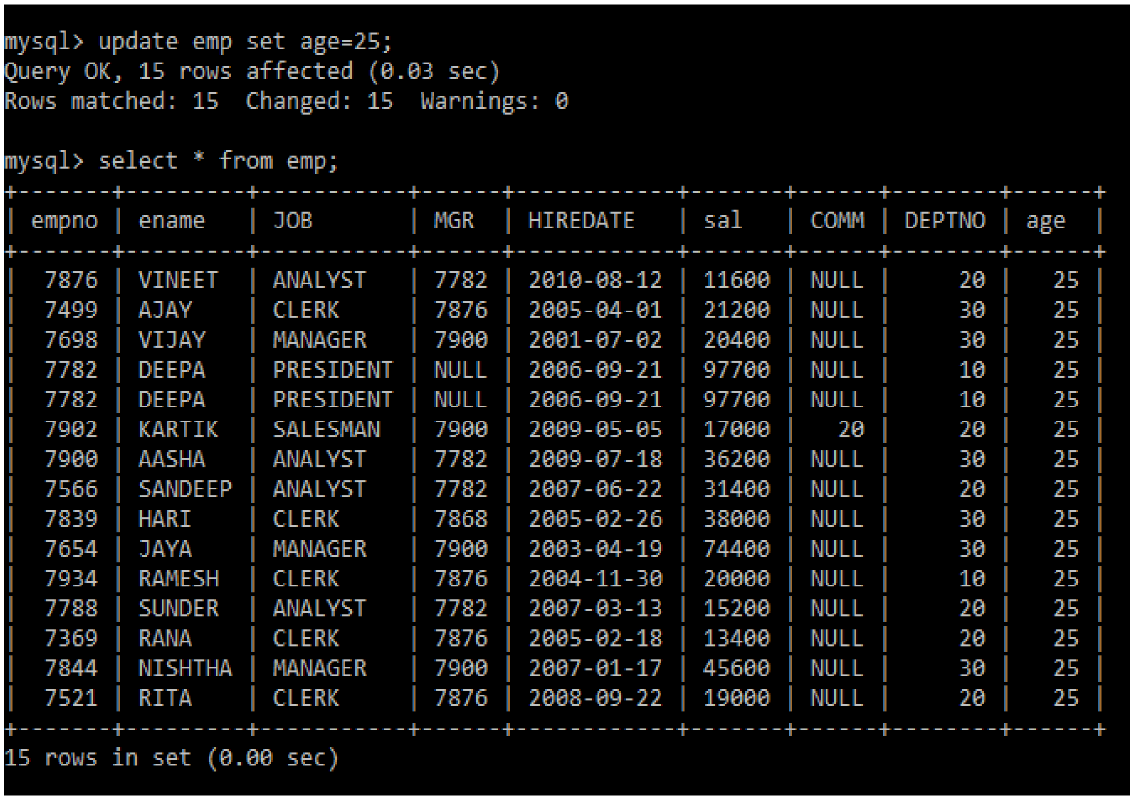
```
mysql> Alter table emp add age integer;
Query OK, 15 rows affected (0.36 sec)
Records: 15  Duplicates: 0  Warnings: 0
```

You can see below that age column has been added to the table but it holds NULL values for all employees.



```
mysql> select *from emp;
+-------+---------+-----------+------+------------+-------+------+--------+------+
| empno | ename   | JOB       | MGR  | HIREDATE   | sal   | COMM | DEPTNO | age  |
+-------+---------+-----------+------+------------+-------+------+--------+------+
|  7876 | VINEET  | ANALYST   | 7782 | 2010-08-12 | 11600 | NULL |     20 | NULL |
|  7499 | AJAY    | CLERK     | 7876 | 2005-04-01 | 21200 | NULL |     30 | NULL |
|  7698 | VIJAY   | MANAGER   | 7900 | 2001-07-02 | 20400 | NULL |     30 | NULL |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 97700 | NULL |     10 | NULL |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 97700 | NULL |     10 | NULL |
|  7902 | KARTIK  | SALESMAN  | 7900 | 2009-05-05 | 17000 |   20 |     20 | NULL |
|  7900 | AASHA   | ANALYST   | 7782 | 2009-07-18 | 36200 | NULL |     30 | NULL |
|  7566 | SANDEEP | ANALYST   | 7782 | 2007-06-22 | 31400 | NULL |     20 | NULL |
|  7839 | HARI    | CLERK     | 7868 | 2005-02-26 | 38000 | NULL |     30 | NULL |
|  7654 | JAYA    | MANAGER   | 7900 | 2003-04-19 | 74400 | NULL |     30 | NULL |
|  7934 | RAMESH  | CLERK     | 7876 | 2004-11-30 | 20000 | NULL |     10 | NULL |
|  7788 | SUNDER  | ANALYST   | 7782 | 2007-03-13 | 15200 | NULL |     20 | NULL |
|  7369 | RANA    | CLERK     | 7876 | 2005-02-18 | 13400 | NULL |     20 | NULL |
|  7844 | NISHTHA | MANAGER   | 7900 | 2007-01-17 | 45600 | NULL |     30 | NULL |
|  7521 | RITA    | CLERK     | 7876 | 2008-09-22 | 19000 | NULL |     20 | NULL |
+-------+---------+-----------+------+------------+-------+------+--------+------+
15 rows in set (0.00 sec)
```

To fill the data in the column age we can use update command. For e.g. to fill the value 25 in the column age of all employees we can use the following command:



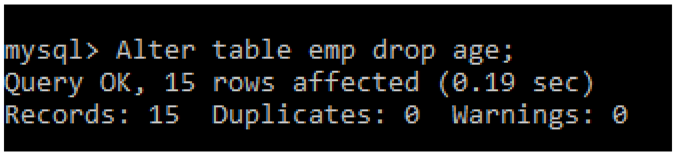
```
mysql> update emp set age=25;
Query OK, 15 rows affected (0.03 sec)
Rows matched: 15  Changed: 15  Warnings: 0

mysql> select * from emp;
+-------+---------+-----------+------+------------+-------+------+--------+------+
| empno | ename   | JOB       | MGR  | HIREDATE   | sal   | COMM | DEPTNO | age  |
+-------+---------+-----------+------+------------+-------+------+--------+------+
|  7876 | VINEET  | ANALYST   | 7782 | 2010-08-12 | 11600 | NULL |     20 |   25 |
|  7499 | AJAY    | CLERK     | 7876 | 2005-04-01 | 21200 | NULL |     30 |   25 |
|  7698 | VIJAY   | MANAGER   | 7900 | 2001-07-02 | 20400 | NULL |     30 |   25 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 97700 | NULL |     10 |   25 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 97700 | NULL |     10 |   25 |
|  7902 | KARTIK  | SALESMAN  | 7900 | 2009-05-05 | 17000 |   20 |     20 |   25 |
|  7900 | AASHA   | ANALYST   | 7782 | 2009-07-18 | 36200 | NULL |     30 |   25 |
|  7566 | SANDEEP | ANALYST   | 7782 | 2007-06-22 | 31400 | NULL |     20 |   25 |
|  7839 | HARI    | CLERK     | 7868 | 2005-02-26 | 38000 | NULL |     30 |   25 |
|  7654 | JAYA    | MANAGER   | 7900 | 2003-04-19 | 74400 | NULL |     30 |   25 |
|  7934 | RAMESH  | CLERK     | 7876 | 2004-11-30 | 20000 | NULL |     10 |   25 |
|  7788 | SUNDER  | ANALYST   | 7782 | 2007-03-13 | 15200 | NULL |     20 |   25 |
|  7369 | RANA    | CLERK     | 7876 | 2005-02-18 | 13400 | NULL |     20 |   25 |
|  7844 | NISHTHA | MANAGER   | 7900 | 2007-01-17 | 45600 | NULL |     30 |   25 |
|  7521 | RITA    | CLERK     | 7876 | 2008-09-22 | 19000 | NULL |     20 |   25 |
+-------+---------+-----------+------+------------+-------+------+--------+------+
15 rows in set (0.00 sec)
```

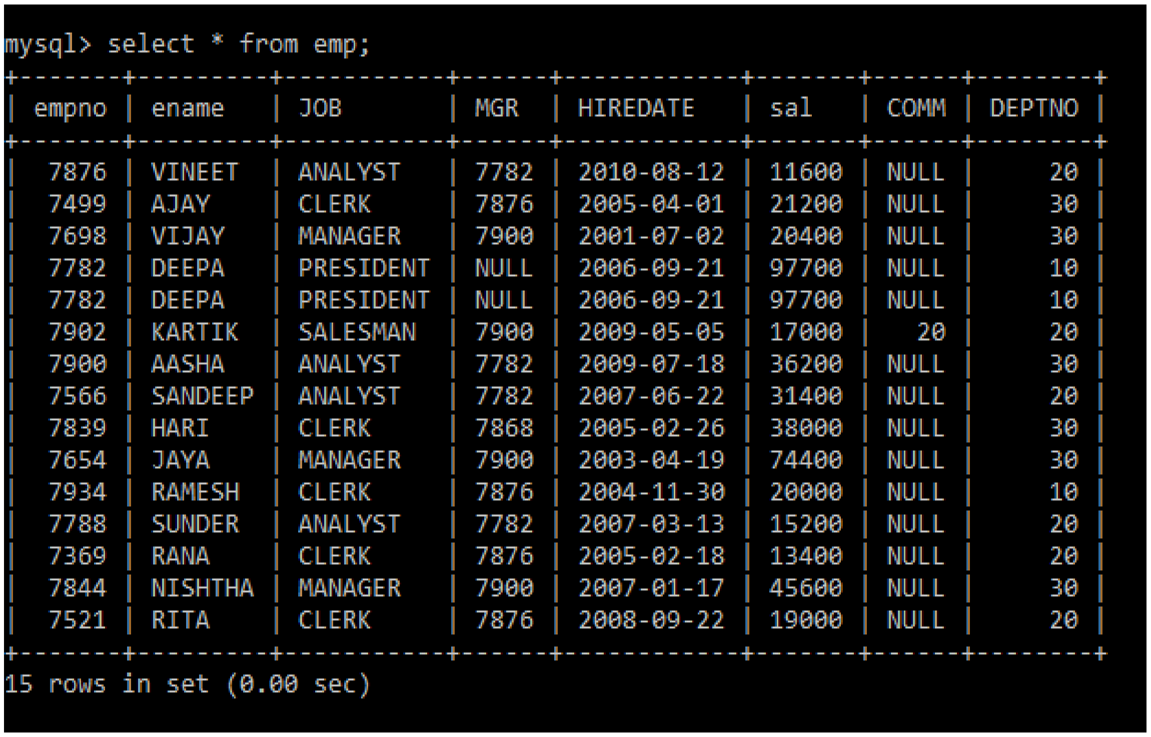
DELETING COLUMNS

Q. WAC to drop a column age from the table emp.

Ans. **Alter table emp drop age;**



```
mysql> Alter table emp drop age;
Query OK, 15 rows affected (0.19 sec)
Records: 15  Duplicates: 0  Warnings: 0
```


```
mysql> select * from emp;
+-------+---------+-----------+------+------------+-------+------+--------+
| empno | ename   | JOB       | MGR  | HIREDATE   | sal   | COMM | DEPTNO |
+-------+---------+-----------+------+------------+-------+------+--------+
|  7876 | VINEET  | ANALYST   | 7782 | 2010-08-12 | 11600 | NULL |     20 |
|  7499 | AJAY    | CLERK     | 7876 | 2005-04-01 | 21200 | NULL |     30 |
|  7698 | VIJAY   | MANAGER   | 7900 | 2001-07-02 | 20400 | NULL |     30 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 97700 | NULL |     10 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 97700 | NULL |     10 |
|  7902 | KARTIK  | SALESMAN  | 7900 | 2009-05-05 | 17000 |   20 |     20 |
|  7900 | AASHA   | ANALYST   | 7782 | 2009-07-18 | 36200 | NULL |     30 |
|  7566 | SANDEEP | ANALYST   | 7782 | 2007-06-22 | 31400 | NULL |     20 |
|  7839 | HARI    | CLERK     | 7868 | 2005-02-26 | 38000 | NULL |     30 |
|  7654 | JAYA    | MANAGER   | 7900 | 2003-04-19 | 74400 | NULL |     30 |
|  7934 | RAMESH  | CLERK     | 7876 | 2004-11-30 | 20000 | NULL |     10 |
|  7788 | SUNDER  | ANALYST   | 7782 | 2007-03-13 | 15200 | NULL |     20 |
|  7369 | RANA    | CLERK     | 7876 | 2005-02-18 | 13400 | NULL |     20 |
|  7844 | NISHTHA | MANAGER   | 7900 | 2007-01-17 | 45600 | NULL |     30 |
|  7521 | RITA    | CLERK     | 7876 | 2008-09-22 | 19000 | NULL |     20 |
+-------+---------+-----------+------+------------+-------+------+--------+
15 rows in set (0.00 sec)
```

Q. WAC to drop column age and job from the table emp.

Ans. **Alter table emp drop age, drop job;**

If the column to be dropped is being referenced in other tables, we can specify the options (RESTRICT or CASCADE) for the drop behavior.

The RESTRICT option would not let the column be dropped if it is being referenced in other tables and CASCADE would drop the constraint associated with this column in this relation as well as all the constraints that refer this column. For eg.

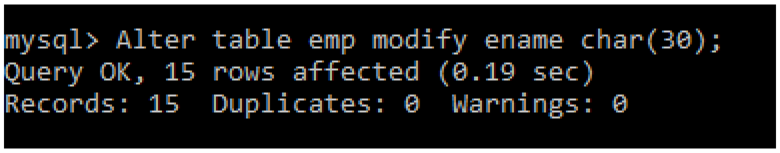
ALTER TABLE DEPARTMENT DROP Dept_No CASCADE;

This will drop the Dept_No column in the DEPARTMENT Table and it would also drop the foreign key constraint EMP_FK in the EMP table as it uses this column.

MODIFYING COLUMNS

Q. WAC to change the size of ename column to char(30).

Ans. **Alter table emp modify ename char(30);**



```
mysql> Alter table emp modify ename char(30);
Query OK, 15 rows affected (0.19 sec)
Records: 15  Duplicates: 0  Warnings: 0
```

Alter table command can be used for dropping the default value or defining a new default value. For example, in the EMP table the default value of Comm is 10. If you want to drop this default value or change this value to 20 then it can be done by using the following commands:

ALTER TABLE EMP ALTER comm DROP DEFAULT;

ALTER TABLE EMP ALTER comm SET DEFAULT 20;

DROPPING KEYS

A foreign key/primary key/key can be dropped by using ALTER TABLE command. For example, if you want to delete the foreign key EMP_FK in the EMP table then following command can be used:

ALTER TABLE EMP DROP FOREIGN KEY EMP_FK;

CASCADE or RESTRICT option can be specified for the drop behavior.

.

ADDING RECORDS TO TABLE

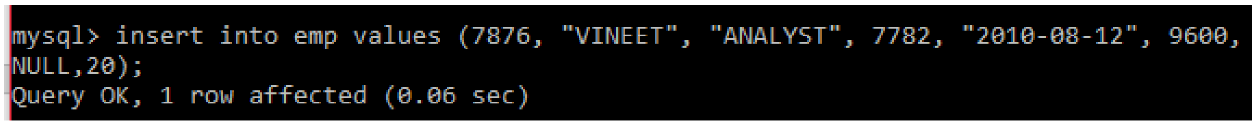
Insert into: This query is used to add row to the table. In this command we give the values for all columns of the table.

SYNTAX:
INSERT INTO <tablename> (<column1>, <column2>, ..., <columnn>) VALUES (<value1>, <value2>, ... <value n>);

Q. Insert a record into the table emp.
Ans. **insert into emp values(7876, "VINEET","ANALYST", 7782, "2010-08-12",9600, NULL,20);**

or

insert into emp (empno, ename, job, mgr, hiredate, sal, comm, deptno) values (7876, "VINEET","ANALYST", 7782, "2010-08-12",9600, NULL,20);



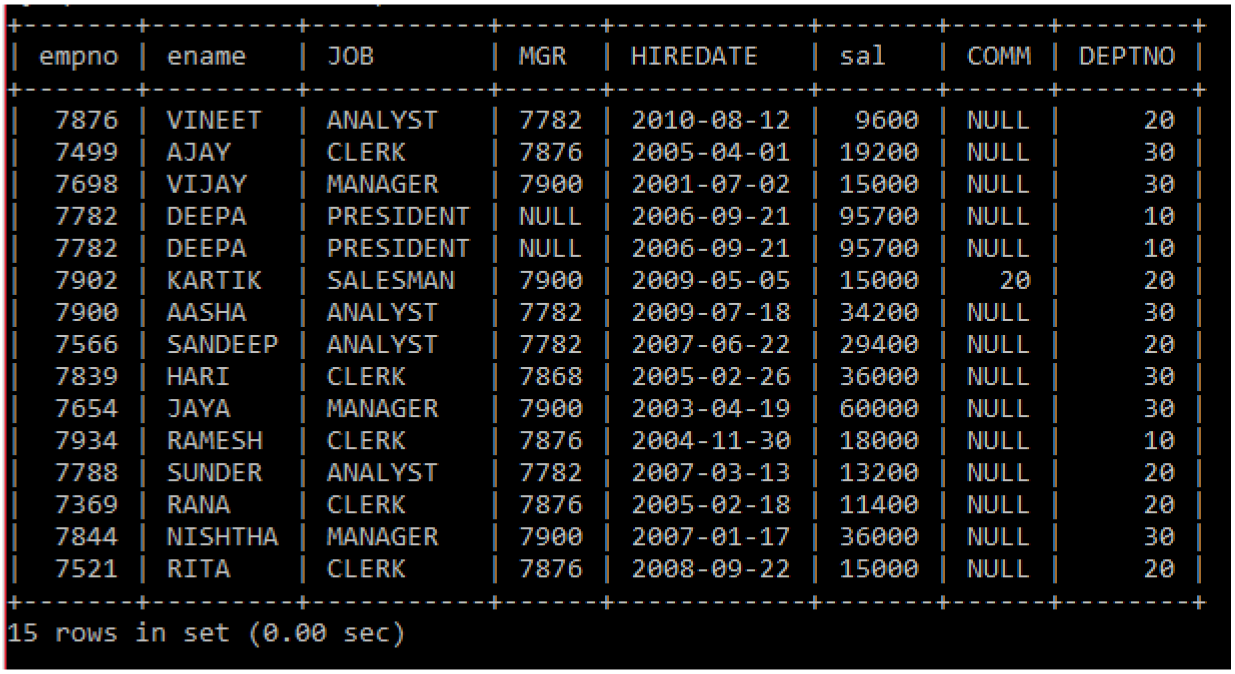
```
mysql> insert into emp values (7876, "VINEET", "ANALYST", 7782, "2010-08-12", 9600,
NULL,20);
Query OK, 1 row affected (0.06 sec)
```

Note:

- ❑ In the first case the data values for each column must match exactly the default order in which they appear in the table.
- ❑ The values for char and date data types must be enclosed in quotes. Standard date format is "yyyy-mm-dd".
- ❑ If you do not want to provide value for any particular column then you can store NULL in that column.

Insert command needs to be given as many times as the number of records you want to store in the table.

Consider the table EMP to show the results of all further queries.



```
+-------+---------+-----------+------+------------+-------+------+--------+
| empno | ename   | JOB       | MGR  | HIREDATE   | sal   | COMM | DEPTNO |
+-------+---------+-----------+------+------------+-------+------+--------+
|  7876 | VINEET  | ANALYST   | 7782 | 2010-08-12 |  9600 | NULL |     20 |
|  7499 | AJAY    | CLERK     | 7876 | 2005-04-01 | 19200 | NULL |     30 |
|  7698 | VIJAY   | MANAGER   | 7900 | 2001-07-02 | 15000 | NULL |     30 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 95700 | NULL |     10 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 95700 | NULL |     10 |
|  7902 | KARTIK  | SALESMAN  | 7900 | 2009-05-05 | 15000 |   20 |     20 |
|  7900 | AASHA   | ANALYST   | 7782 | 2009-07-18 | 34200 | NULL |     30 |
|  7566 | SANDEEP | ANALYST   | 7782 | 2007-06-22 | 29400 | NULL |     20 |
|  7839 | HARI    | CLERK     | 7868 | 2005-02-26 | 36000 | NULL |     30 |
|  7654 | JAYA    | MANAGER   | 7900 | 2003-04-19 | 60000 | NULL |     30 |
|  7934 | RAMESH  | CLERK     | 7876 | 2004-11-30 | 18000 | NULL |     10 |
|  7788 | SUNDER  | ANALYST   | 7782 | 2007-03-13 | 13200 | NULL |     20 |
|  7369 | RANA    | CLERK     | 7876 | 2005-02-18 | 11400 | NULL |     20 |
|  7844 | NISHTHA | MANAGER   | 7900 | 2007-01-17 | 36000 | NULL |     30 |
|  7521 | RITA    | CLERK     | 7876 | 2008-09-22 | 15000 | NULL |     20 |
+-------+---------+-----------+------+------------+-------+------+--------+
15 rows in set (0.00 sec)
```

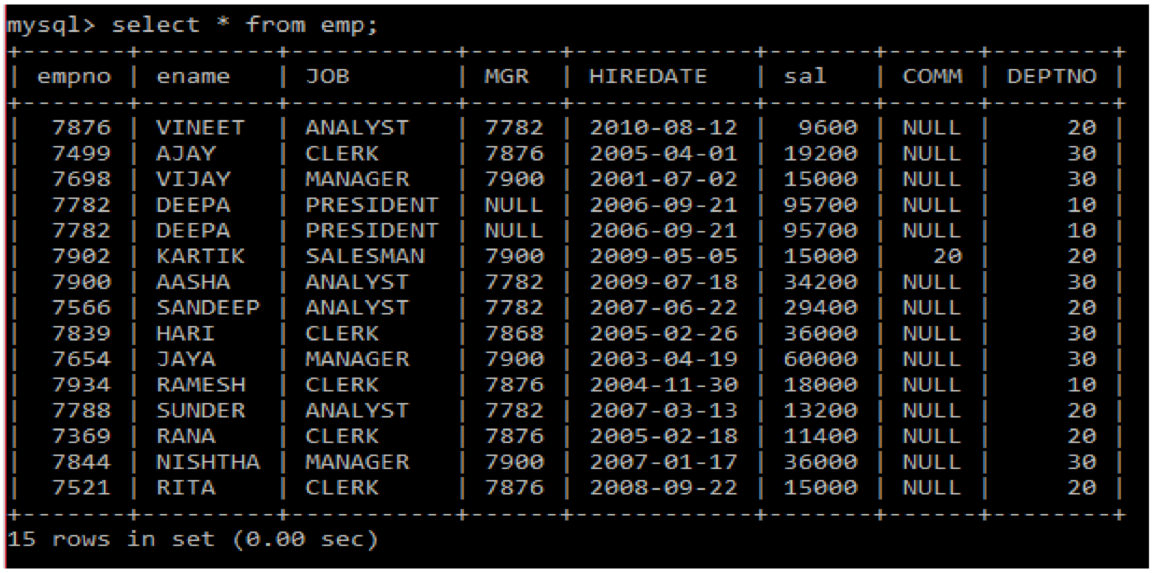
DISPLAYING RECORDS

SELECT COMMAND

The select command is used to display data stored in a table.

- ❑ **To display all attributes of all records:**

Q. Write a command to display all records from the table emp.
Ans. **Select * from emp;**

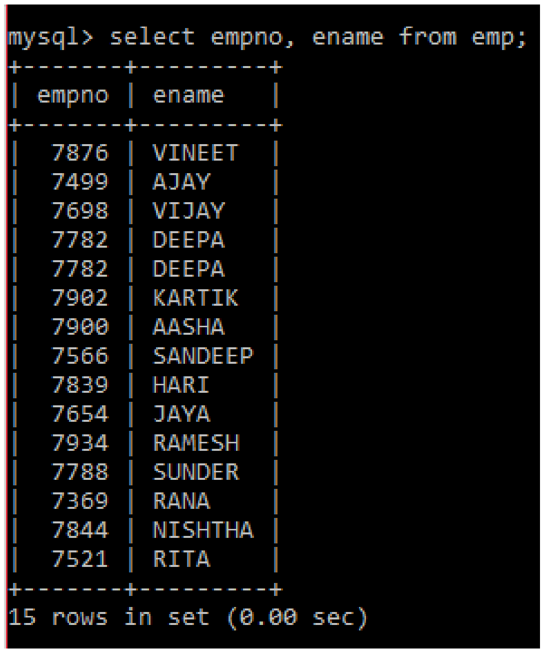


```
mysql> select * from emp;
+-------+---------+-----------+------+------------+-------+------+--------+
| empno | ename   | JOB       | MGR  | HIREDATE   | sal   | COMM | DEPTNO |
+-------+---------+-----------+------+------------+-------+------+--------+
|  7876 | VINEET  | ANALYST   | 7782 | 2010-08-12 |  9600 | NULL |     20 |
|  7499 | AJAY    | CLERK     | 7876 | 2005-04-01 | 19200 | NULL |     30 |
|  7698 | VIJAY   | MANAGER   | 7900 | 2001-07-02 | 15000 | NULL |     30 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 95700 | NULL |     10 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 95700 | NULL |     10 |
|  7902 | KARTIK  | SALESMAN  | 7900 | 2009-05-05 | 15000 |   20 |     20 |
|  7900 | AASHA   | ANALYST   | 7782 | 2009-07-18 | 34200 | NULL |     30 |
|  7566 | SANDEEP | ANALYST   | 7782 | 2007-06-22 | 29400 | NULL |     20 |
|  7839 | HARI    | CLERK     | 7868 | 2005-02-26 | 36000 | NULL |     30 |
|  7654 | JAYA    | MANAGER   | 7900 | 2003-04-19 | 60000 | NULL |     30 |
|  7934 | RAMESH  | CLERK     | 7876 | 2004-11-30 | 18000 | NULL |     10 |
|  7788 | SUNDER  | ANALYST   | 7782 | 2007-03-13 | 13200 | NULL |     20 |
|  7369 | RANA    | CLERK     | 7876 | 2005-02-18 | 11400 | NULL |     20 |
|  7844 | NISHTHA | MANAGER   | 7900 | 2007-01-17 | 36000 | NULL |     30 |
|  7521 | RITA    | CLERK     | 7876 | 2008-09-22 | 15000 | NULL |     20 |
+-------+---------+-----------+------+------------+-------+------+--------+
15 rows in set (0.00 sec)
```

As you can see the above command will display all attributes/columns of all records present in the table.

❑ **To display selected attributes of all records:**

Q. Write a command to display employee number and name of all employees.
Ans. **Select empno, ename from emp;**



```
mysql> select empno, ename from emp;
+-------+---------+
| empno | ename   |
+-------+---------+
|  7876 | VINEET  |
|  7499 | AJAY    |
|  7698 | VIJAY   |
|  7782 | DEEPA   |
|  7782 | DEEPA   |
|  7902 | KARTIK  |
|  7900 | AASHA   |
|  7566 | SANDEEP |
|  7839 | HARI    |
|  7654 | JAYA    |
|  7934 | RAMESH  |
|  7788 | SUNDER  |
|  7369 | RANA    |
|  7844 | NISHTHA |
|  7521 | RITA    |
+-------+---------+
15 rows in set (0.00 sec)
```

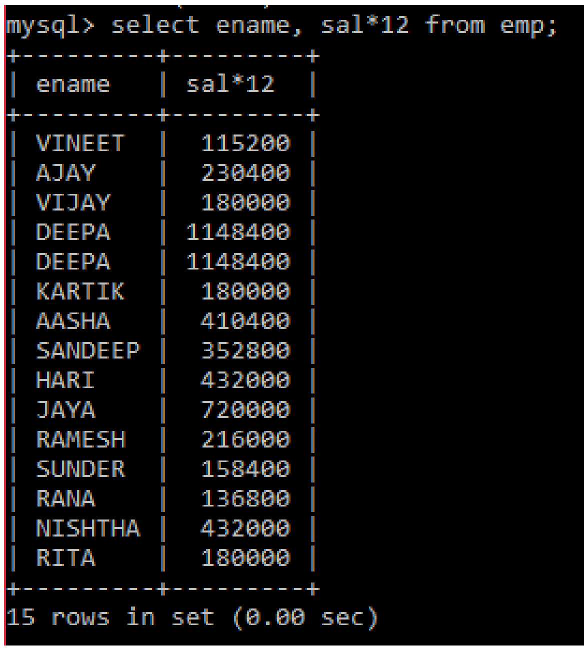
The above command will display the data stored in the columns empno and ename of all records.

Note: Please use the same column names as given in the table in the SQL command.

SELECT WITH AIRTHMETIC OPERATORS

Arithmetic operators can be used in the select command to perform calculations.

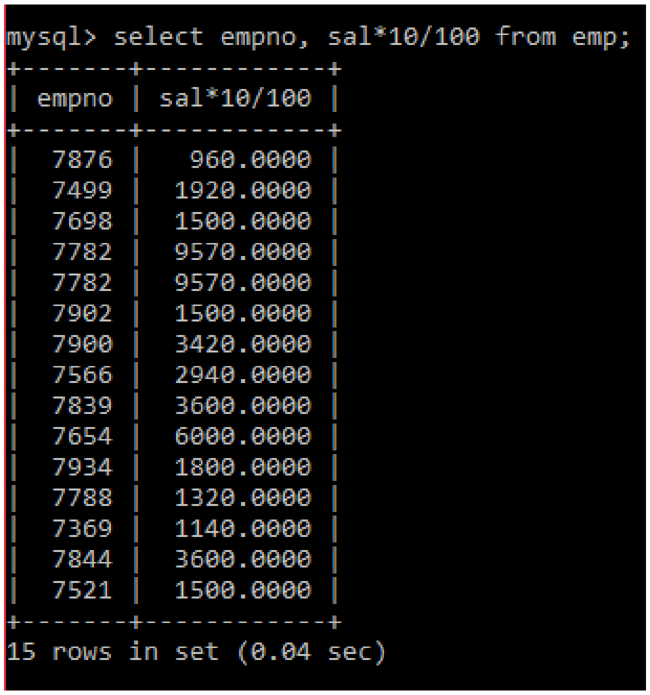
Q. Write a command to display the name and annual salary of all employees from the table emp.
Ans. **Select ename, sal*12 from emp;**



```
mysql> select ename, sal*12 from emp;
+---------+---------+
| ename   | sal*12  |
+---------+---------+
| VINEET  |  115200 |
| AJAY    |  230400 |
| VIJAY   |  180000 |
| DEEPA   | 1148400 |
| DEEPA   | 1148400 |
| KARTIK  |  180000 |
| AASHA   |  410400 |
| SANDEEP |  352800 |
| HARI    |  432000 |
| JAYA    |  720000 |
| RAMESH  |  216000 |
| SUNDER  |  158400 |
| RANA    |  136800 |
| NISHTHA |  432000 |
| RITA    |  180000 |
+---------+---------+
15 rows in set (0.00 sec)
```

The above command will display name and annual salary (which is obtained by multiplying salary of employees by 12) of all records.

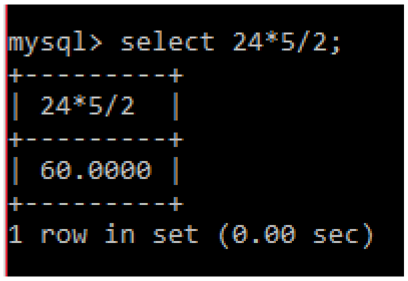
Q. Write a command to display employee number and bonus of all employees where bonus is calculated as 10% of salary.
Ans. **Select empno, sal*10/100 from emp;**



```
mysql> select empno, sal*10/100 from emp;
+-------+------------+
| empno | sal*10/100 |
+-------+------------+
|  7876 |   960.0000 |
|  7499 |  1920.0000 |
|  7698 |  1500.0000 |
|  7782 |  9570.0000 |
|  7782 |  9570.0000 |
|  7902 |  1500.0000 |
|  7900 |  3420.0000 |
|  7566 |  2940.0000 |
|  7839 |  3600.0000 |
|  7654 |  6000.0000 |
|  7934 |  1800.0000 |
|  7788 |  1320.0000 |
|  7369 |  1140.0000 |
|  7844 |  3600.0000 |
|  7521 |  1500.0000 |
+-------+------------+
15 rows in set (0.04 sec)
```

The above command will display employee number and bonus (which is obtained as 10% of salary) of all records.

Q. Write a command to display the result of the expression: 24*5/2
Ans. **Select 24*5/2;**



```
mysql> select 24*5/2;
+---------+
| 24*5/2  |
+---------+
| 60.0000 |
+---------+
1 row in set (0.00 sec)
```

The above command will display the result of the calculation 24*5/2.

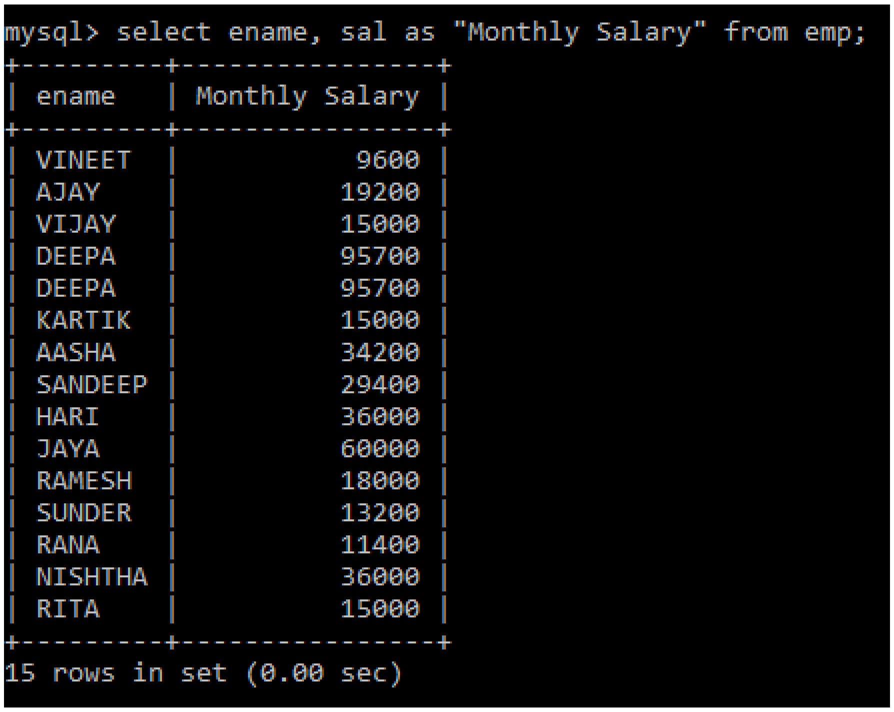
USING COLUMN ALIAS

Column alias lets different name to appear for a column then the actual one in the output. Column alias does not rename the column. It simply displays a different column name in the output. You can use AS keyword to specify the alias name. But As keyword is optional.

Q. Write a command to display employee name and salary and display salary as Monthly salary.
Ans. **Select ename, sal as "Monthly Salary" from emp;**

or

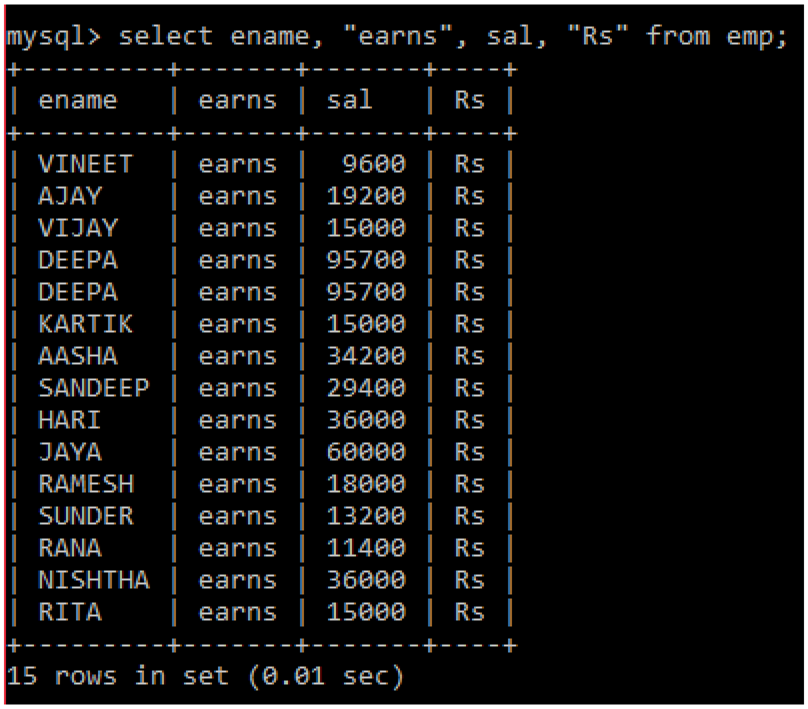
Select ename , sal from emp;


```
mysql> select ename, sal as "Monthly Salary" from emp;
+---------+----------------+
| ename   | Monthly Salary |
+---------+----------------+
| VINEET  |           9600 |
| AJAY    |          19200 |
| VIJAY   |          15000 |
| DEEPA   |          95700 |
| DEEPA   |          95700 |
| KARTIK  |          15000 |
| AASHA   |          34200 |
| SANDEEP |          29400 |
| HARI    |          36000 |
| JAYA    |          60000 |
| RAMESH  |          18000 |
| SUNDER  |          13200 |
| RANA    |          11400 |
| NISHTHA |          36000 |
| RITA    |          15000 |
+---------+----------------+
15 rows in set (0.00 sec)
```

In the above command you can see that instead of ***sal*** as the column name ***Monthly Salary*** is displayed.

PUTTING TEXT IN QUERY OUTPUT

Text can also be displayed in the query output. For eg.
Select ename, “earns” , sal , “Rs” from emp;

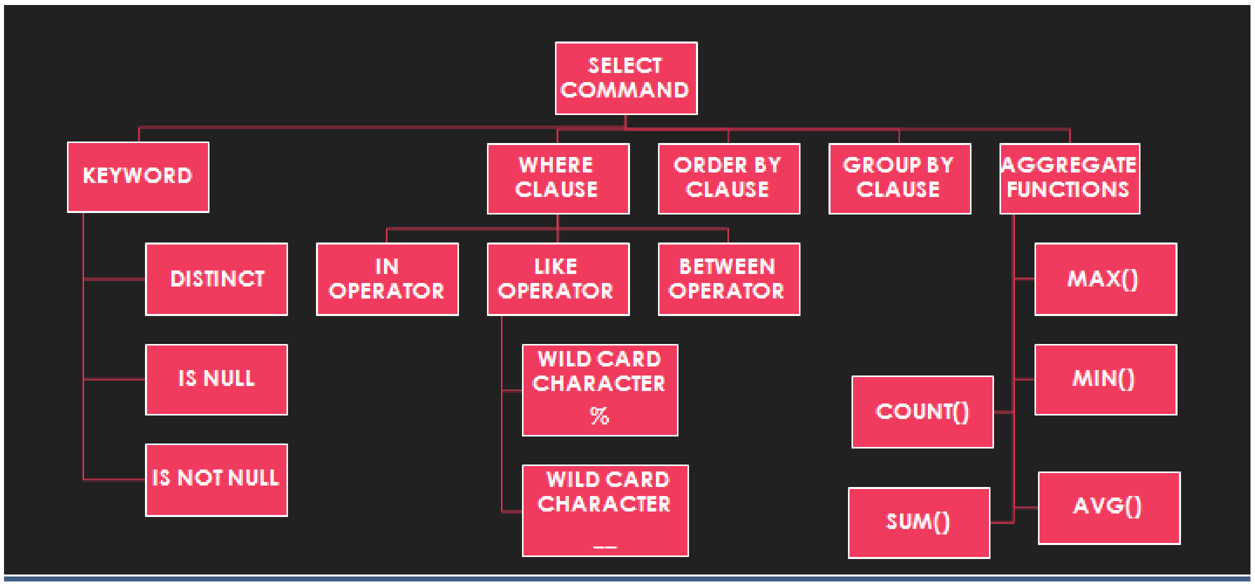


```
mysql> select ename, "earns", sal, "Rs" from emp;
+---------+-------+-------+----+
| ename   | earns | sal   | Rs |
+---------+-------+-------+----+
| VINEET  | earns |  9600 | Rs |
| AJAY    | earns | 19200 | Rs |
| VIJAY   | earns | 15000 | Rs |
| DEEPA   | earns | 95700 | Rs |
| DEEPA   | earns | 95700 | Rs |
| KARTIK  | earns | 15000 | Rs |
| AASHA   | earns | 34200 | Rs |
| SANDEEP | earns | 29400 | Rs |
| HARI    | earns | 36000 | Rs |
| JAYA    | earns | 60000 | Rs |
| RAMESH  | earns | 18000 | Rs |
| SUNDER  | earns | 13200 | Rs |
| RANA    | earns | 11400 | Rs |
| NISHTHA | earns | 36000 | Rs |
| RITA    | earns | 15000 | Rs |
+---------+-------+-------+----+
15 rows in set (0.01 sec)
```

The above command will display the text ***earns*** and ***Rs*** in the output.

KEYWORDS/CLAUSES IN SELECT COMMAND

Various clauses/keywords/aggregate functions associated with SELECT command are as follows:



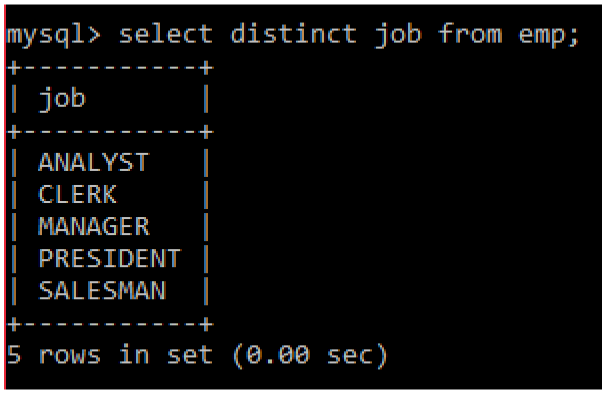
CLAUSE/KEYWORD	USAGE
DISTINCT	Used to display distinct values from a column of a table.
WHERE	Used to specify the condition based on which rows of a table are displayed.
BETWEEN	Used to define the range of values within which the column values must fall to make a condition true. Range includes both the upper and the lower values.
IN	Used to select values that match any value in a list of Specified values.
LIKE	Used for pattern matching of string data using wildcard characters **%** and **_ .**
IS NULL	Used to select rows in which the specified column is NULL (or is NOT NULL).
ORDER BY	Used to display the selected rows in ascending or in descending order of the specified column/expression.

ELIMINATING DUPLICATE VALUES

DISTINCT keyword: The distinct keyword is used to eliminate duplicate values and display the duplicated data only once.

Q. Give a command to display different types of jobs present in the table emp.

Ans. **Select distinct job from emp;**



```
mysql> select distinct job from emp;
+-----------+
| job       |
+-----------+
| ANALYST   |
| CLERK     |
| MANAGER   |
| PRESIDENT |
| SALESMAN  |
+-----------+
5 rows in set (0.00 sec)
```

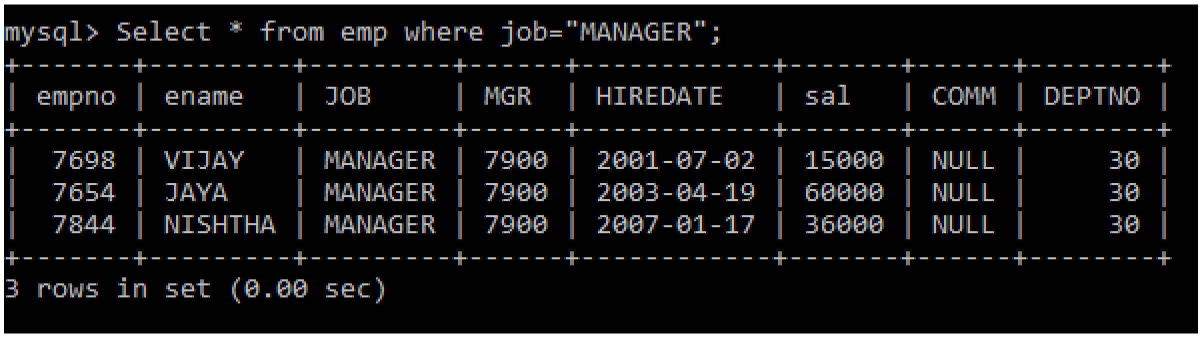
The above command will display different types of jobs only once and eliminate duplicate entries.

SELECT COMMAND: WHERE CLAUSE

Where clause is used to display selected rows based upon the condition specified.

Q. Give a command to display records of those employees whose job is manager.

Ans. **Select * from emp where job="MANAGER";**

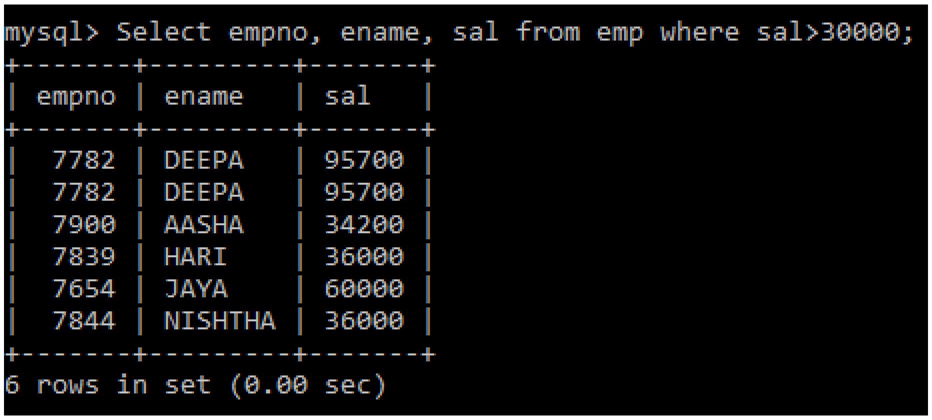


```
mysql> Select * from emp where job="MANAGER";
+-------+---------+---------+------+------------+-------+------+--------+
| empno | ename   | JOB     | MGR  | HIREDATE   | sal   | COMM | DEPTNO |
+-------+---------+---------+------+------------+-------+------+--------+
|  7698 | VIJAY   | MANAGER | 7900 | 2001-07-02 | 15000 | NULL |     30 |
|  7654 | JAYA    | MANAGER | 7900 | 2003-04-19 | 60000 | NULL |     30 |
|  7844 | NISHTHA | MANAGER | 7900 | 2007-01-17 | 36000 | NULL |     30 |
+-------+---------+---------+------+------------+-------+------+--------+
3 rows in set (0.00 sec)
```

Above command will display only those records present in the table where ***job*** is ***manager***.

Q. Give a command to display employee number, name and salary of those employees whose salary is more than 30000.

Ans. **Select empno, ename, sal from emp where sal>30000;**



```
mysql> Select empno, ename, sal from emp where sal>30000;
+-------+---------+-------+
| empno | ename   | sal   |
+-------+---------+-------+
|  7782 | DEEPA   | 95700 |
|  7782 | DEEPA   | 95700 |
|  7900 | AASHA   | 34200 |
|  7839 | HARI    | 36000 |
|  7654 | JAYA    | 60000 |
|  7844 | NISHTHA | 36000 |
+-------+---------+-------+
6 rows in set (0.00 sec)
```

Above command will display ***employee number, name*** and ***salary*** of those employees whose ***salary*** is more than 30000.

RELATIONAL OPERATORS IN WHERE CLAUSE

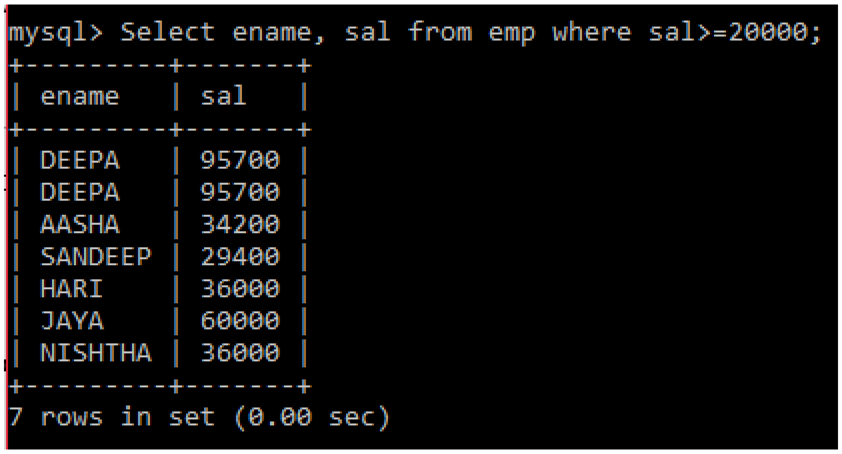
Relational operators that can be used in the where clause are as follows:

=	**Equal to**

<	**Less than**
>	**Greater than**
>=	**Greater than equal to**
<=	**Less than equal to**
!= or <>	**Not equal to**

Q. Write a command (WAC) to display name and salary of those employees whose salary is greater than or equal to 10000.

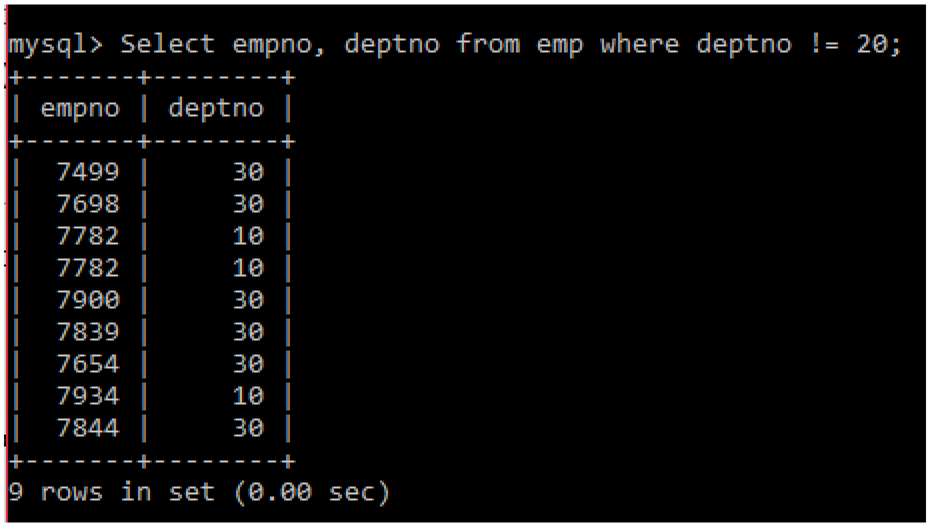
Ans. Select ename, sal from emp where sal>=20000;



```
mysql> Select ename, sal from emp where sal>=20000;
+---------+-------+
| ename   | sal   |
+---------+-------+
| DEEPA   | 95700 |
| DEEPA   | 95700 |
| AASHA   | 34200 |
| SANDEEP | 29400 |
| HARI    | 36000 |
| JAYA    | 60000 |
| NISHTHA | 36000 |
+---------+-------+
7 rows in set (0.00 sec)
```

Q. Write a command to display employee number, department number of those employees whose department number is not 20.

Ans. Select empno, deptno from emp where deptno != 20;



```
mysql> Select empno, deptno from emp where deptno != 20;
+-------+--------+
| empno | deptno |
+-------+--------+
|  7499 |     30 |
|  7698 |     30 |
|  7782 |     10 |
|  7782 |     10 |
|  7900 |     30 |
|  7839 |     30 |
|  7654 |     30 |
|  7934 |     10 |
|  7844 |     30 |
+-------+--------+
9 rows in set (0.00 sec)
```

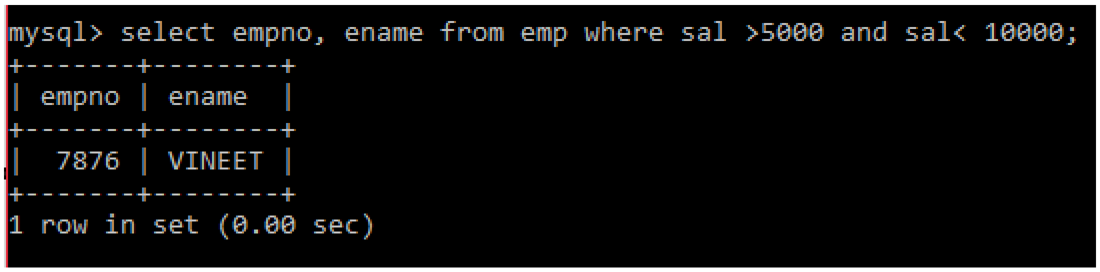
LOGICAL OPERATORS IN WHERE CLAUSE

NOT	**Negates a condition**
AND / &&	**The condition is true if both the conditions joined using AND are true**
OR / ||	**The condition is true if any of the conditions joined using OR are true**

Logical operators are used to join two conditions.

Q. WAC to display employee number and name of those employees whose salary is greater than 10000 and less than 5000.

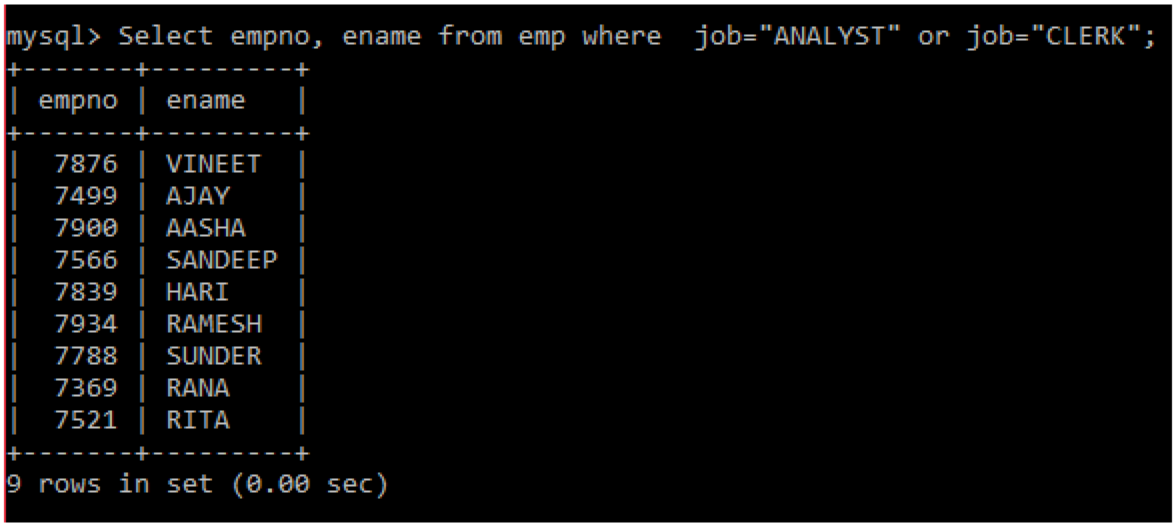
Ans. **select empno, ename from emp where sal >5000 and sal< 10000;**



```
mysql> select empno, ename from emp where sal >5000 and sal< 10000;
+-------+--------+
| empno | ename  |
+-------+--------+
|  7876 | VINEET |
+-------+--------+
1 row in set (0.00 sec)
```

Q. WAC to display employee number and name of those employees who are ANALYST or CLERK.

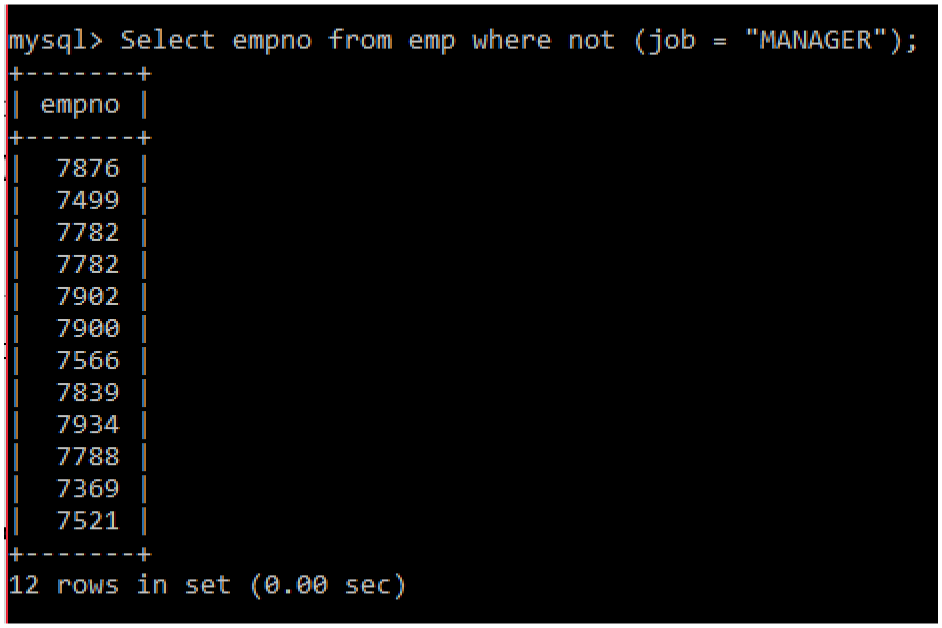
Ans. **Select empno, ename from emp where job="ANALYST" or job="CLERK";**



```
mysql> Select empno, ename from emp where  job="ANALYST" or job="CLERK";
+-------+---------+
| empno | ename   |
+-------+---------+
|  7876 | VINEET  |
|  7499 | AJAY    |
|  7900 | AASHA   |
|  7566 | SANDEEP |
|  7839 | HARI    |
|  7934 | RAMESH  |
|  7788 | SUNDER  |
|  7369 | RANA    |
|  7521 | RITA    |
+-------+---------+
9 rows in set (0.00 sec)
```

Q. WAC to display employee number of those employees who are not manager.

Ans. **Select empno from emp where not (job = “MANAGER”);**



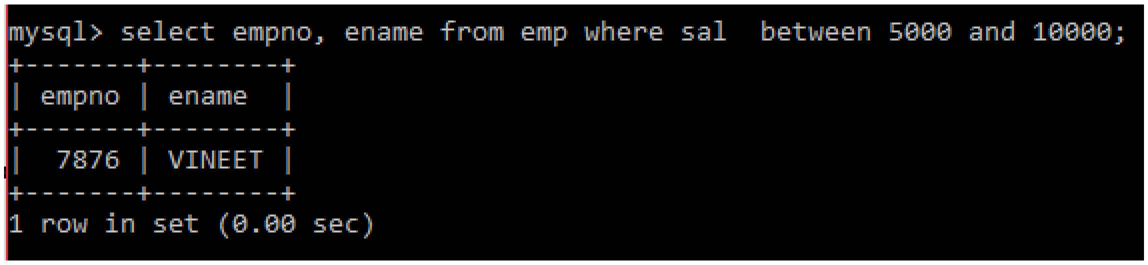
```
mysql> Select empno from emp where not (job = "MANAGER");
+-------+
| empno |
+-------+
|  7876 |
|  7499 |
|  7782 |
|  7782 |
|  7902 |
|  7900 |
|  7566 |
|  7839 |
|  7934 |
|  7788 |
|  7369 |
|  7521 |
+-------+
12 rows in set (0.00 sec)
```

BETWEEN OPERATOR – RANGE OF VALUES

The between operator is used to define the range of values. The range includes both the upper and lower values.

Q. WAC to display employee number and name of those employees whose salary is greater than equal to 10000 and less than equal to 5000.

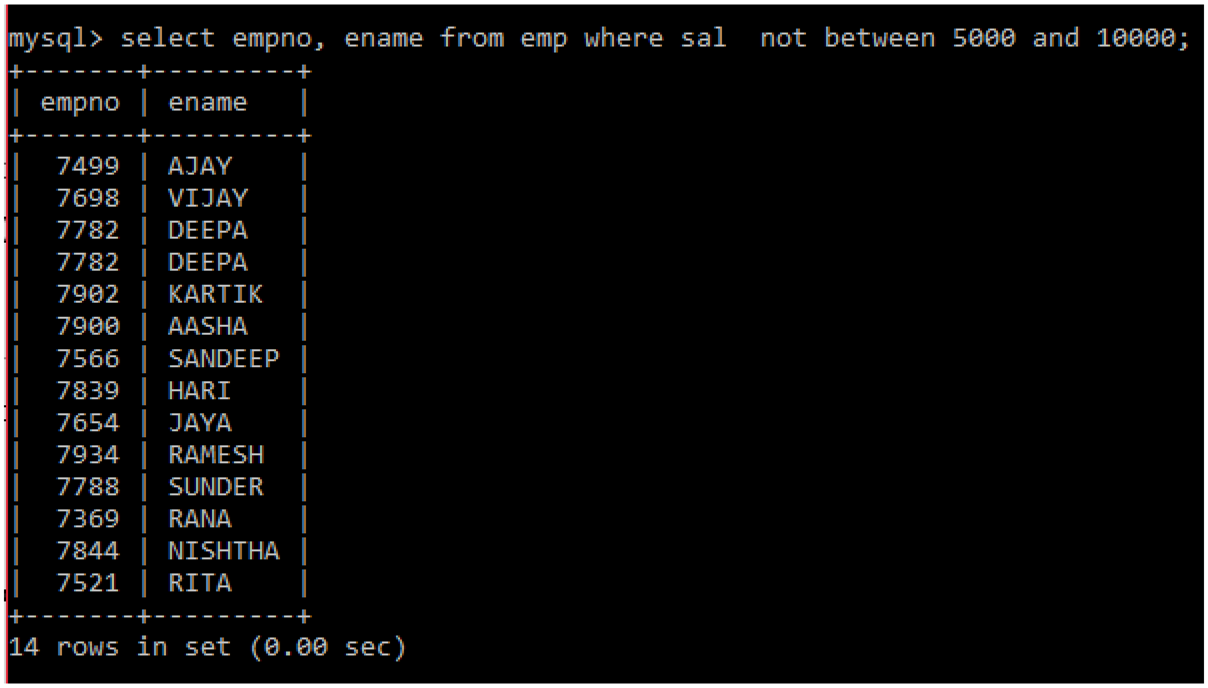
Ans. **select empno, ename from emp where sal between 5000 and 10000;**



```
mysql> select empno, ename from emp where sal  between 5000 and 10000;
+-------+--------+
| empno | ename  |
+-------+--------+
|  7876 | VINEET |
+-------+--------+
1 row in set (0.00 sec)
```

Q. WAC to display employee number and name of those employees whose salary is NOT greater than equal to 10000 and less than equal to 5000.

Ans. **select empno, ename from emp where sal not between 5000 and 10000;**



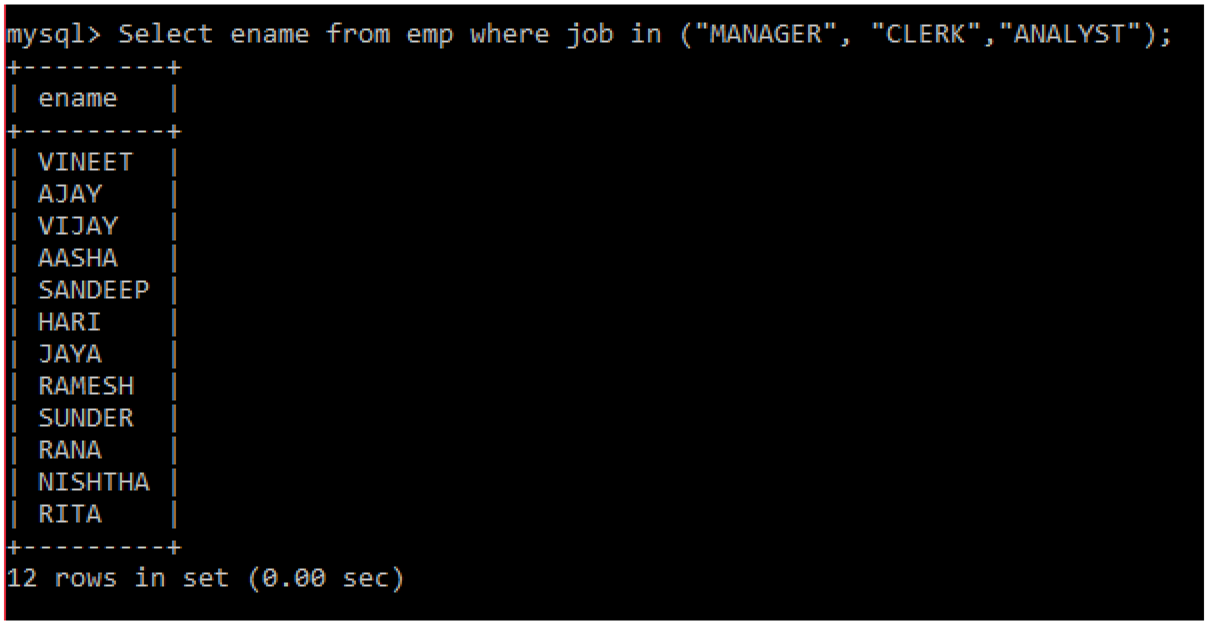
```
mysql> select empno, ename from emp where sal  not between 5000 and 10000;
+-------+---------+
| empno | ename   |
+-------+---------+
|  7499 | AJAY    |
|  7698 | VIJAY   |
|  7782 | DEEPA   |
|  7782 | DEEPA   |
|  7902 | KARTIK  |
|  7900 | AASHA   |
|  7566 | SANDEEP |
|  7839 | HARI    |
|  7654 | JAYA    |
|  7934 | RAMESH  |
|  7788 | SUNDER  |
|  7369 | RANA    |
|  7844 | NISHTHA |
|  7521 | RITA    |
+-------+---------+
14 rows in set (0.00 sec)
```

IN OPERATOR – LIST OF VALUES

IN operator is used to select value/values from a list of values.

Q. WAC to display name of employees who are manager, clerk and analyst.

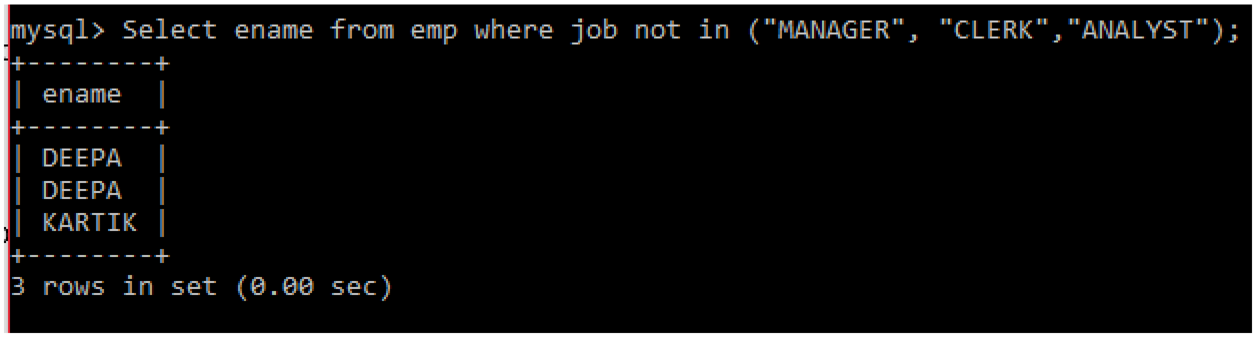
Ans**. Select ename from emp where job in (“MANAGER”, “CLERK”,”ANALYST”);**



```
mysql> Select ename from emp where job in ("MANAGER", "CLERK","ANALYST");
+---------+
| ename   |
+---------+
| VINEET  |
| AJAY    |
| VIJAY   |
| AASHA   |
| SANDEEP |
| HARI    |
| JAYA    |
| RAMESH  |
| SUNDER  |
| RANA    |
| NISHTHA |
| RITA    |
+---------+
12 rows in set (0.00 sec)
```

Q. WAC to display name of all employees except manager, clerk and analyst.

Ans. **Select ename from emp where job not in (“MANAGER”, “CLERK”,”ANALYST”);**



```
mysql> Select ename from emp where job not in ("MANAGER", "CLERK","ANALYST");
+--------+
| ename  |
+--------+
| DEEPA  |
| DEEPA  |
| KARTIK |
+--------+
3 rows in set (0.00 sec)
```

We can have another query in the WHERE clause of SQL query if the condition is based on the result of another query as shown below:

Q: To retrieve the names of all the employee whose department is located in SOUTH.

SELECT DISTINCT ename

FROM EMP

WHERE deptno IN (Select Dept_ID

FROM DEPARTMENT

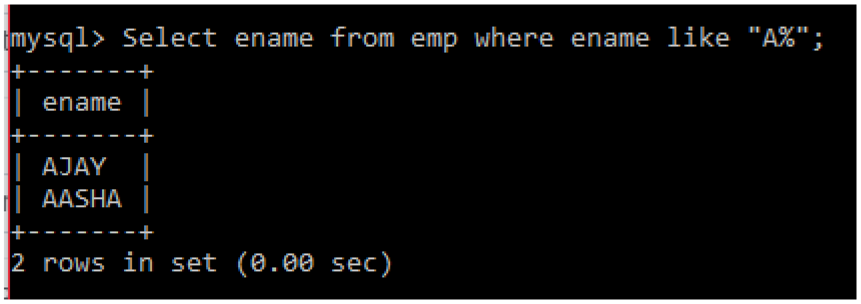
WHERE Location = 'SOUTH');

LIKE OPERATOR – PATTERN MATCHING

The LIKE operator is used to fetch data from the table which matches a specified pattern. It uses two wild card characters:

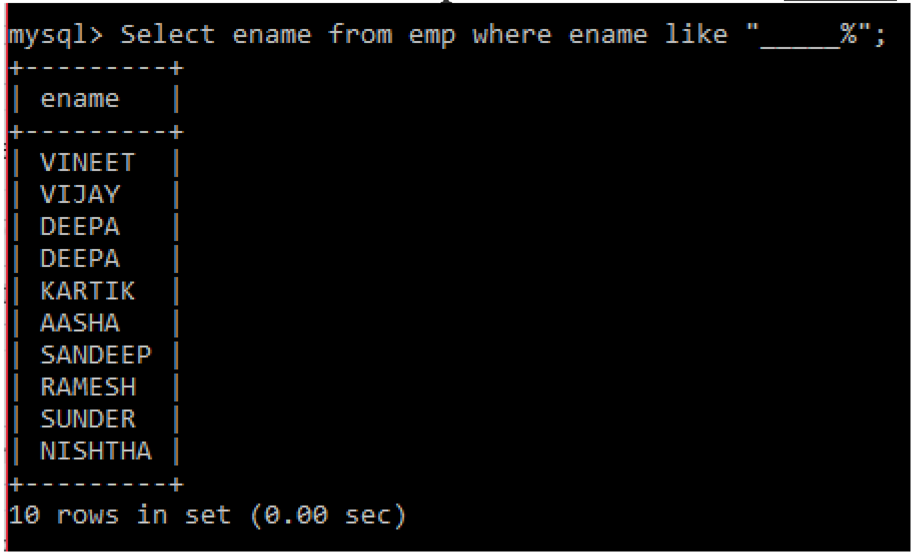
% (percentage): It is used to represent any sequence of zero or more characters.
_ (underscore): It is used to represent a single character.

Q. WAC to display names of those employees whose name starts with A.
Ans. **Select ename from emp where ename like "A%";**



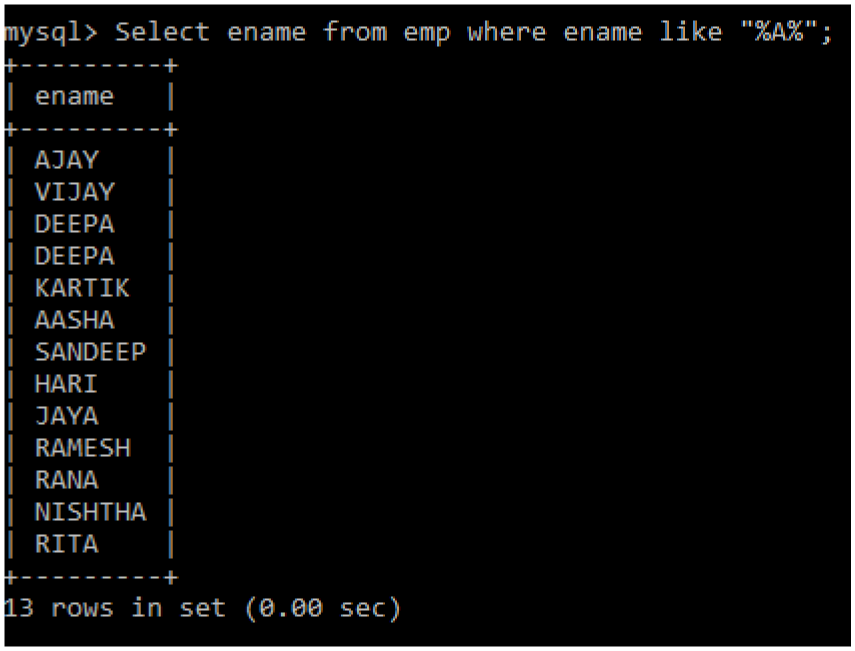
```
mysql> Select ename from emp where ename like "A%";
+-------+
| ename |
+-------+
| AJAY  |
| AASHA |
+-------+
2 rows in set (0.00 sec)
```

Q. WAC to display names of those employees whose name is at least 5 characters long.
Ans. **Select ename from emp where ename like "_____%";**



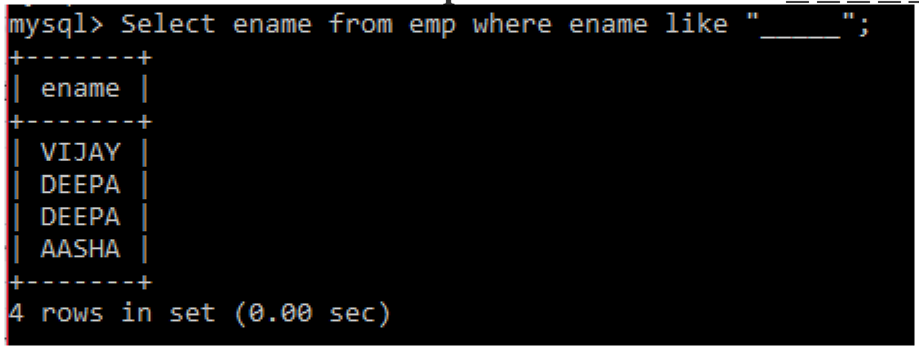
```
mysql> Select ename from emp where ename like "_____%";
+---------+
| ename   |
+---------+
| VINEET  |
| VIJAY   |
| DEEPA   |
| DEEPA   |
| KARTIK  |
| AASHA   |
| SANDEEP |
| RAMESH  |
| SUNDER  |
| NISHTHA |
+---------+
10 rows in set (0.00 sec)
```

Q. WAC to display names of those employees whose name contains A.
Ans. **Select ename from emp where ename like "%A%";**



```
mysql> Select ename from emp where ename like "%A%";
+---------+
| ename   |
+---------+
| AJAY    |
| VIJAY   |
| DEEPA   |
| DEEPA   |
| KARTIK  |
| AASHA   |
| SANDEEP |
| HARI    |
| JAYA    |
| RAMESH  |
| RANA    |
| NISHTHA |
| RITA    |
+---------+
13 rows in set (0.00 sec)
```

Q. WAC to display names of those employees whose name is exactly 5 characters long.
Ans. **Select ename from emp where ename like "_ _ _ _ _";**



```
mysql> Select ename from emp where ename like "_____";
+-------+
| ename |
+-------+
| VIJAY |
| DEEPA |
| DEEPA |
| AASHA |
+-------+
4 rows in set (0.00 sec)
```

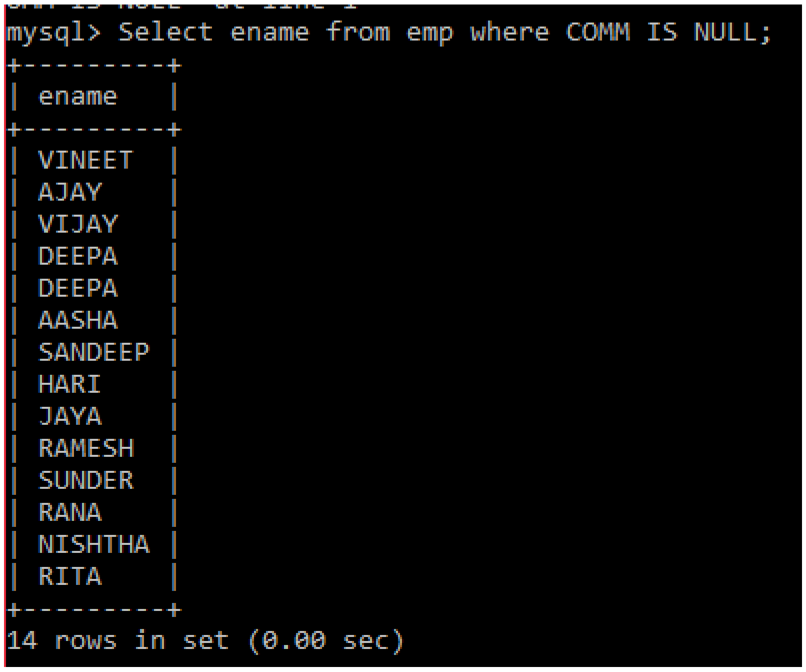
The keyword NOT LIKE is used to select the rows that do not match the specified pattern.

IS NULL/IS NOT NULL

NULL is used to represent unknown value. NULL is not the same as zero or a space or any other character. In a table NULL is searched for using IS NULL/IS NOT NULL keywords.

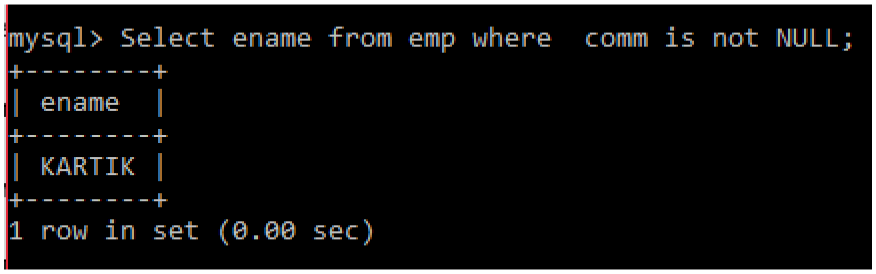
Q. WAC to display name of those employees whose commission is NULL.

Ans. **Select ename from emp where comm is NULL;**


```
mysql> Select ename from emp where COMM IS NULL;
+---------+
| ename   |
+---------+
| VINEET  |
| AJAY    |
| VIJAY   |
| DEEPA   |
| DEEPA   |
| AASHA   |
| SANDEEP |
| HARI    |
| JAYA    |
| RAMESH  |
| SUNDER  |
| RANA    |
| NISHTHA |
| RITA    |
+---------+
14 rows in set (0.00 sec)
```

Q. WAC to display name of those employees whose commission is not NULL.

Ans. **Select ename from emp where comm is not NULL;**

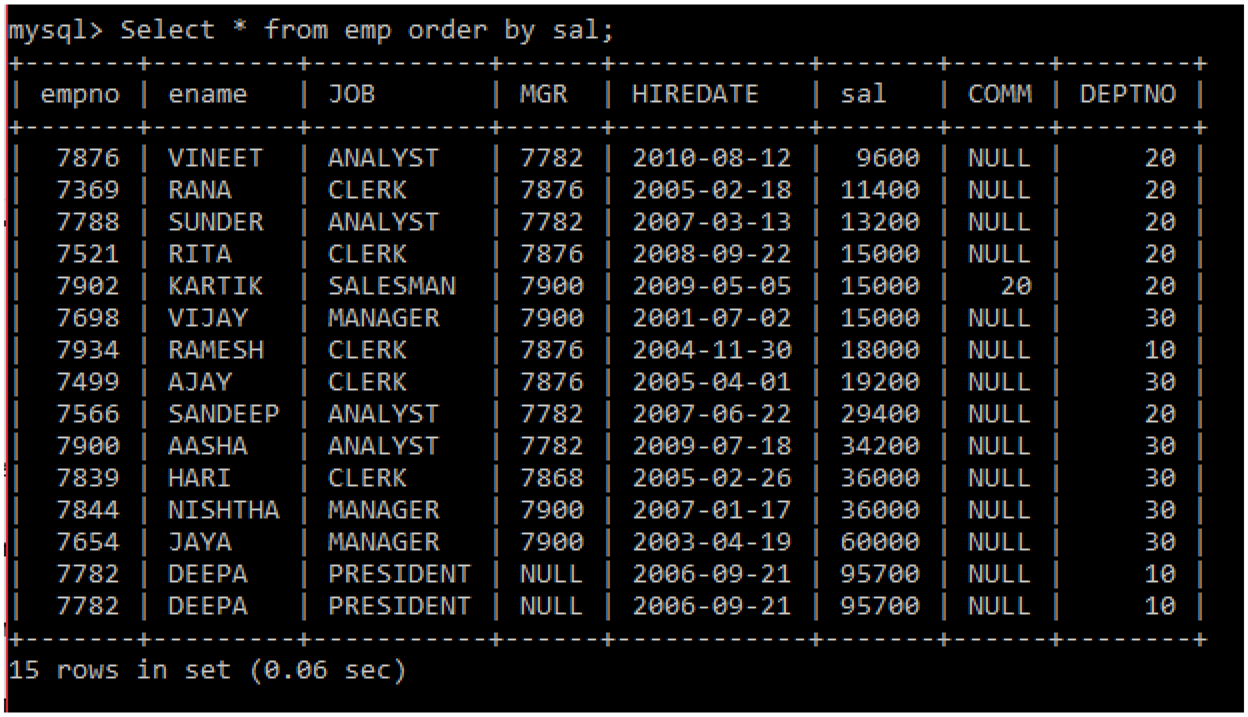

```
mysql> Select ename from emp where  comm is not NULL;
+--------+
| ename  |
+--------+
| KARTIK |
+--------+
1 row in set (0.00 sec)
```

ORDER BY – ARRANGING THE RESULT

ORDER BY clause is used to arrange the results of the select command in either ascending or descending order. The default order is ascending order. However you can also add ASC keyword for ascending order. To arrange the data in descending order use the keyword DESC. It can also be used to arrange the data in particular order on multiple columns.

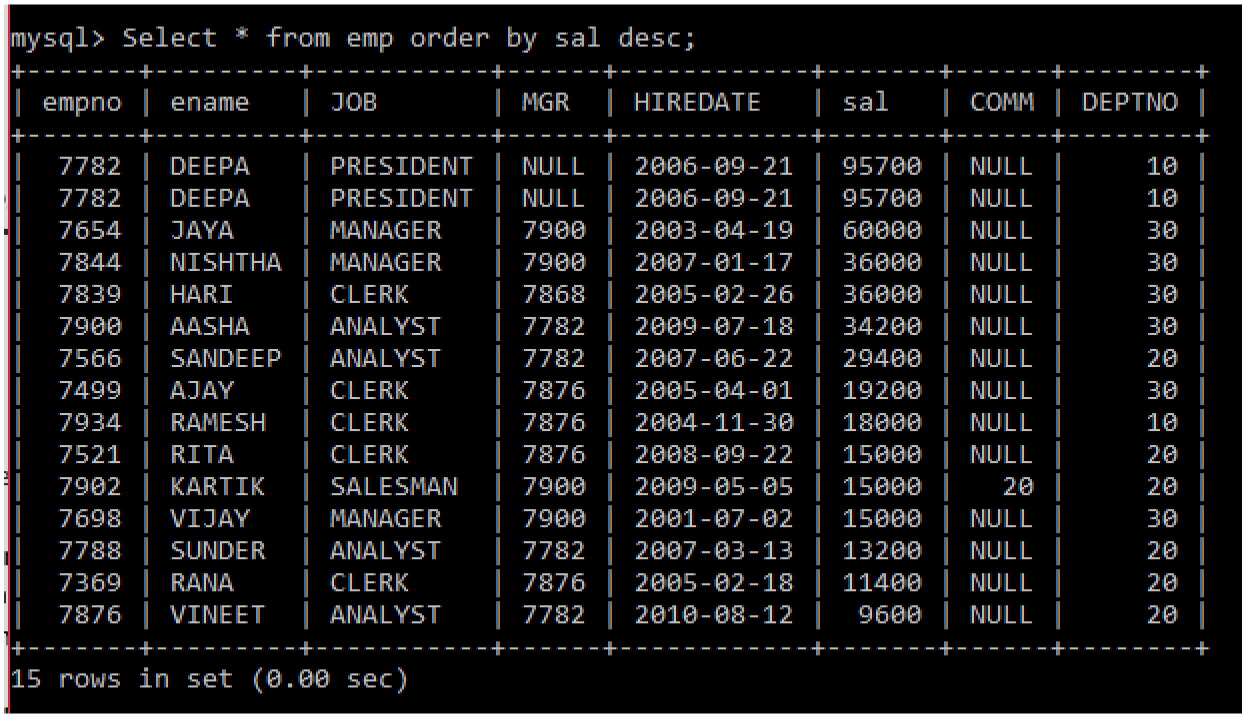
Q. WAC to display the data of employees in ascending order of their salary.

Ans. **Select * from emp order by sal;**
or
select * from emp order by sal asc;



```
mysql> Select * from emp order by sal;
+-------+---------+-----------+------+------------+-------+------+--------+
| empno | ename   | JOB       | MGR  | HIREDATE   | sal   | COMM | DEPTNO |
+-------+---------+-----------+------+------------+-------+------+--------+
|  7876 | VINEET  | ANALYST   | 7782 | 2010-08-12 |  9600 | NULL |     20 |
|  7369 | RANA    | CLERK     | 7876 | 2005-02-18 | 11400 | NULL |     20 |
|  7788 | SUNDER  | ANALYST   | 7782 | 2007-03-13 | 13200 | NULL |     20 |
|  7521 | RITA    | CLERK     | 7876 | 2008-09-22 | 15000 | NULL |     20 |
|  7902 | KARTIK  | SALESMAN  | 7900 | 2009-05-05 | 15000 |   20 |     20 |
|  7698 | VIJAY   | MANAGER   | 7900 | 2001-07-02 | 15000 | NULL |     30 |
|  7934 | RAMESH  | CLERK     | 7876 | 2004-11-30 | 18000 | NULL |     10 |
|  7499 | AJAY    | CLERK     | 7876 | 2005-04-01 | 19200 | NULL |     30 |
|  7566 | SANDEEP | ANALYST   | 7782 | 2007-06-22 | 29400 | NULL |     20 |
|  7900 | AASHA   | ANALYST   | 7782 | 2009-07-18 | 34200 | NULL |     30 |
|  7839 | HARI    | CLERK     | 7868 | 2005-02-26 | 36000 | NULL |     30 |
|  7844 | NISHTHA | MANAGER   | 7900 | 2007-01-17 | 36000 | NULL |     30 |
|  7654 | JAYA    | MANAGER   | 7900 | 2003-04-19 | 60000 | NULL |     30 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 95700 | NULL |     10 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 95700 | NULL |     10 |
+-------+---------+-----------+------+------------+-------+------+--------+
15 rows in set (0.06 sec)
```

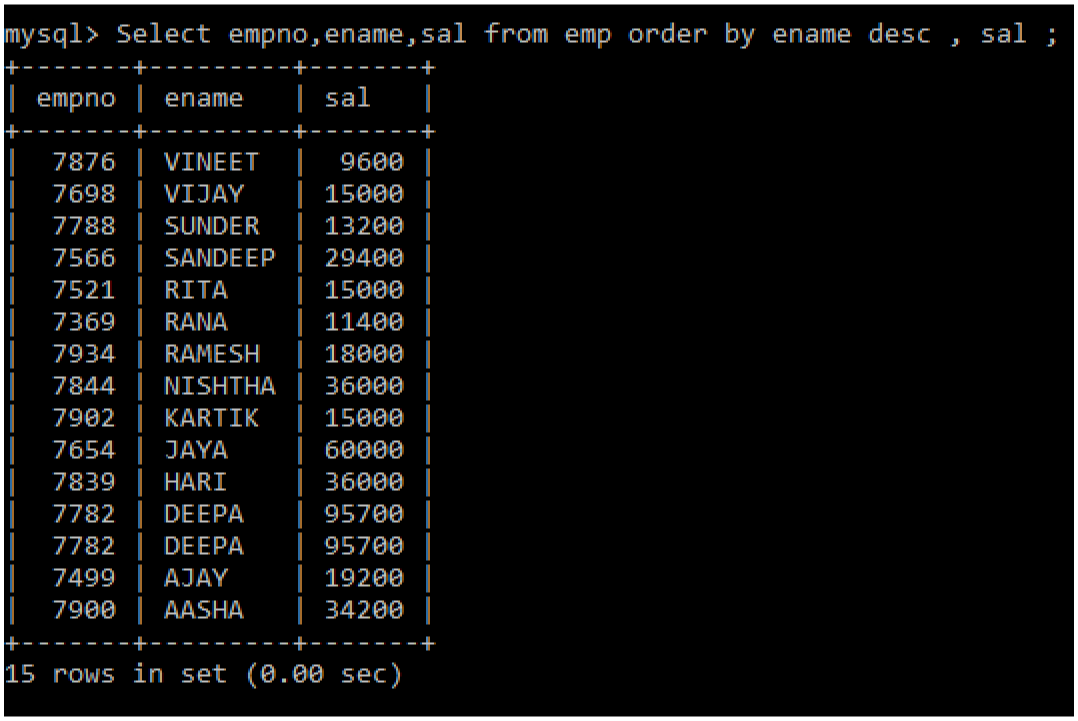
Q. WAC to display the data of employees in descending order of their name.
Ans. **Select * from emp order by ename desc;**



```
mysql> Select * from emp order by sal desc;
+-------+---------+-----------+------+------------+-------+------+--------+
| empno | ename   | JOB       | MGR  | HIREDATE   | sal   | COMM | DEPTNO |
+-------+---------+-----------+------+------------+-------+------+--------+
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 95700 | NULL |     10 |
|  7782 | DEEPA   | PRESIDENT | NULL | 2006-09-21 | 95700 | NULL |     10 |
|  7654 | JAYA    | MANAGER   | 7900 | 2003-04-19 | 60000 | NULL |     30 |
|  7844 | NISHTHA | MANAGER   | 7900 | 2007-01-17 | 36000 | NULL |     30 |
|  7839 | HARI    | CLERK     | 7868 | 2005-02-26 | 36000 | NULL |     30 |
|  7900 | AASHA   | ANALYST   | 7782 | 2009-07-18 | 34200 | NULL |     30 |
|  7566 | SANDEEP | ANALYST   | 7782 | 2007-06-22 | 29400 | NULL |     20 |
|  7499 | AJAY    | CLERK     | 7876 | 2005-04-01 | 19200 | NULL |     30 |
|  7934 | RAMESH  | CLERK     | 7876 | 2004-11-30 | 18000 | NULL |     10 |
|  7521 | RITA    | CLERK     | 7876 | 2008-09-22 | 15000 | NULL |     20 |
|  7902 | KARTIK  | SALESMAN  | 7900 | 2009-05-05 | 15000 |   20 |     20 |
|  7698 | VIJAY   | MANAGER   | 7900 | 2001-07-02 | 15000 | NULL |     30 |
|  7788 | SUNDER  | ANALYST   | 7782 | 2007-03-13 | 13200 | NULL |     20 |
|  7369 | RANA    | CLERK     | 7876 | 2005-02-18 | 11400 | NULL |     20 |
|  7876 | VINEET  | ANALYST   | 7782 | 2010-08-12 |  9600 | NULL |     20 |
+-------+---------+-----------+------+------------+-------+------+--------+
15 rows in set (0.00 sec)
```

Q. WAC to display the data of employees in descending order of their name and within that in ascending order of salary.

Ans. **Select empno,ename,sal from emp order by ename desc , sal ;**



```
mysql> Select empno,ename,sal from emp order by ename desc , sal ;
+-------+---------+-------+
| empno | ename   | sal   |
+-------+---------+-------+
|  7876 | VINEET  |  9600 |
|  7698 | VIJAY   | 15000 |
|  7788 | SUNDER  | 13200 |
|  7566 | SANDEEP | 29400 |
|  7521 | RITA    | 15000 |
|  7369 | RANA    | 11400 |
|  7934 | RAMESH  | 18000 |
|  7844 | NISHTHA | 36000 |
|  7902 | KARTIK  | 15000 |
|  7654 | JAYA    | 60000 |
|  7839 | HARI    | 36000 |
|  7782 | DEEPA   | 95700 |
|  7782 | DEEPA   | 95700 |
|  7499 | AJAY    | 19200 |
|  7900 | AASHA   | 34200 |
+-------+---------+-------+
15 rows in set (0.00 sec)
```

INSERTING NULL VALUES

There are two ways of inserting NULL values in the column/columns of the table.

- If we don't have data for particular columns, we can leave them at the time of giving INSERT INTO command. For those columns NULL values are automatically inserted.

Q. WAC to insert employee number as 7771, name as AKSHAT and depart number as 20. All other columns will have Null values.

Ans. **Insert into emp (empno, ename, deptno) values (7771, "AKSHAT",20);**

- We can also insert NULL values explicitly.

Q. WAC to insert employee number as 7771, name as AKSHAT and depart number as 20. All other columns will have Null values.

Ans**. Insert into emp (empno, ename,job, mge, hiredate, comm, deptno) values (7771, "AKSHAT",NULL, NULL, NULL, NULL, 20);**

UPDATE STATEMENT

The UPDATE statement is used to modify the data present in the table.

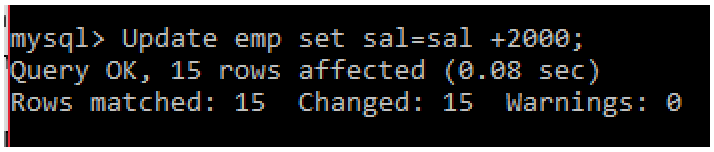
Syntax:

UPDATE <table_name>

SET <column name> = <value>, [<column name> = <value>, …] [WHERE <condn>];

Q. WAC to increase the salary of all employees by Rs 2000.

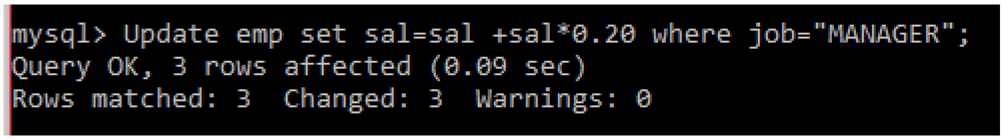
Ans. **Update emp set sal=sal +2000;**



```
mysql> Update emp set sal=sal +2000;
Query OK, 15 rows affected (0.08 sec)
Rows matched: 15  Changed: 15  Warnings: 0
```

Q. WAC to increase the salary by 20% of all MANAGER.

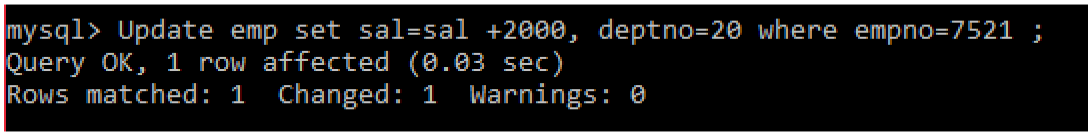
Ans. **Update emp set sal=sal +sal*0.20 where job= 'MANAGER';**



```
mysql> Update emp set sal=sal +sal*0.20 where job="MANAGER";
Query OK, 3 rows affected (0.09 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

Q. WAC to increase the salary by 2000Rs and change department number to 20 for employee whose employee number is 7521.

Ans. **Update emp set sal=sal +2000, deptno=20 where empno=7521 ;**



```
mysql> Update emp set sal=sal +2000, deptno=20 where empno=7521 ;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

DELETE STATEMENT

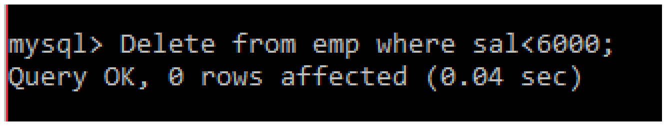
DELETE is used to delete rows from the table.

Syntax:

DELETE FROM < tablename> [Where < condn>];

Q. WAC to delete data of those employee whose salary is less than 6000.

Ans. **Delete from emp where sal<6000;**



```
mysql> Delete from emp where sal<6000;
Query OK, 0 rows affected (0.04 sec)
```

Q. WAC to delete all rows of table employee.

Ans. **Delete from emp;**

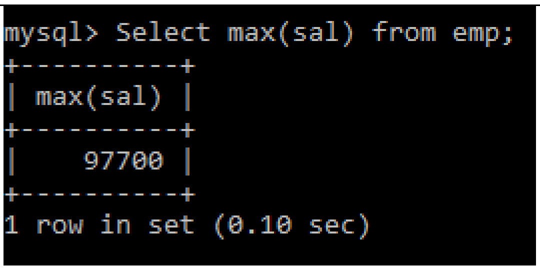
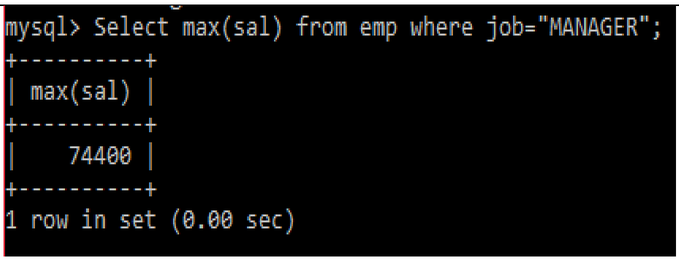
AGGREGATE FUNCTIONS

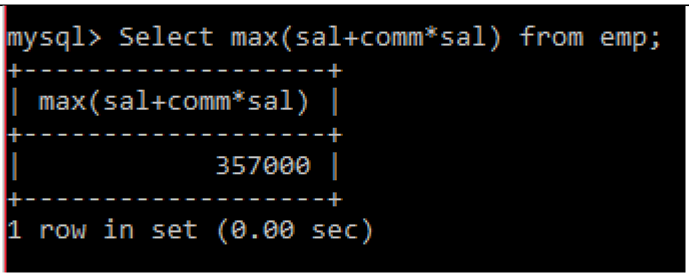
Aggregate functions work on multiple rows. There are 5 types of aggregate functions:

Aggregate function	Purpose
MAX()	To find the maximum value under the specified column
MIN()	To find the minimum value under the specified column
AVG()	To find the average of values under the specified column
SUM()	To find the sum of values under the specified column
COUNT()	To count the values under the specified column

AGGREGATE FUNCTION – MAX()

Please refer to table employee for the queries given below:

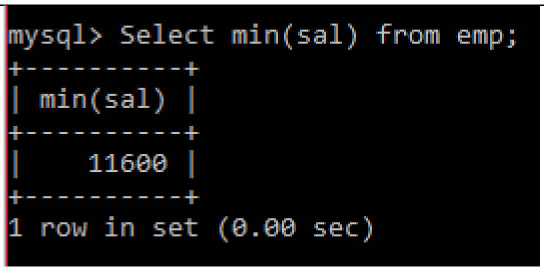
Purpose	Command	Result
To find the highest salary paid to an employee	Select max(sal) from emp;	 ``mysql> Select max(sal) from emp;`` ``+----------+`` ``| max(sal) |`` ``+----------+`` ``|    97700 |`` ``+----------+`` ``1 row in set (0.10 sec)``
To find the highest salary paid to MANAGERS	Select max(sal) from emp where job="MANAGER";	 ``mysql> Select max(sal) from emp where job="MANAGER";`` ``+----------+`` ``| max(sal) |`` ``+----------+`` ``|    74400 |`` ``+----------+`` ``1 row in set (0.00 sec)``

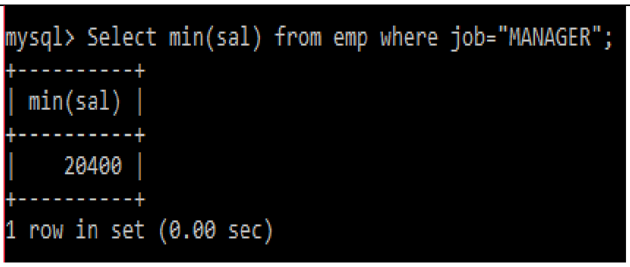
To find the highest salary + commision paid to employee	Select max(sal+comm*sal) from emp;	
		```
		mysql> Select max(sal+comm*sal) from emp;
		+--------------------+
		| max(sal+comm*sal) |
		+--------------------+
		|             357000 |
		+--------------------+
		1 row in set (0.00 sec)
		```

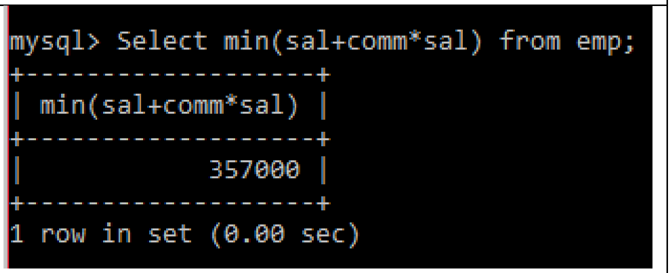
AGGREGATE FUNCTION – MIN()

Please refer to table employee for the queries given below:

Purpose	Command	Result

To find the lowest salary paid to an employee	Select min(sal) from emp;	
		```
		mysql> Select min(sal) from emp;
		+----------+
		| min(sal) |
		+----------+
		|    11600 |
		+----------+
		1 row in set (0.00 sec)
		```

To find the lowest salary paid to MANAGERS	Select min(sal) from emp where job="MANAGER";	
		```
		mysql> Select min(sal) from emp where job="MANAGER";
		+----------+
		| min(sal) |
		+----------+
		|    20400 |
		+----------+
		1 row in set (0.00 sec)
		```

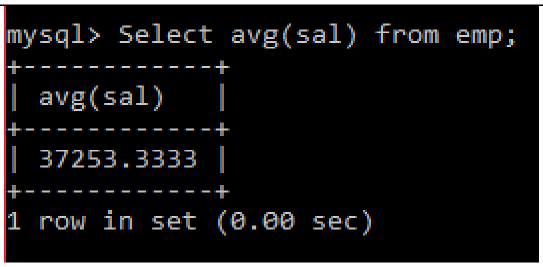
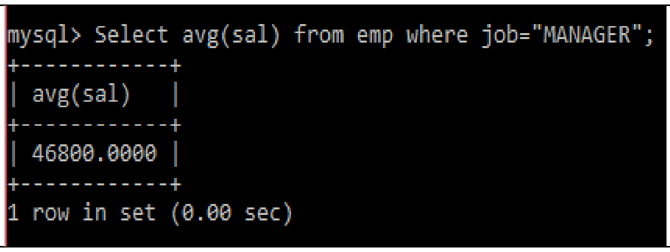
To find the lowest salary + commision paid to employee	Select min(sal+comm*sal) from emp;	
		```
		mysql> Select min(sal+comm*sal) from emp;
		+--------------------+
		| min(sal+comm*sal) |
		+--------------------+
		|             357000 |
		+--------------------+
		1 row in set (0.00 sec)
		```

AGGREGATE FUNCTION – AVG()

The argument of AVG() function can be of numeric (int/decimal) type only. Averages of String and Date type data are not defined.

AVG() function considers only non NULL values in that column.

Please refer to table employee for the queries given below:

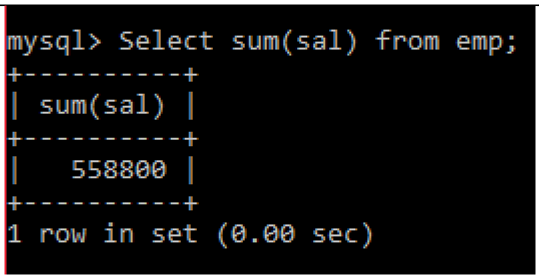
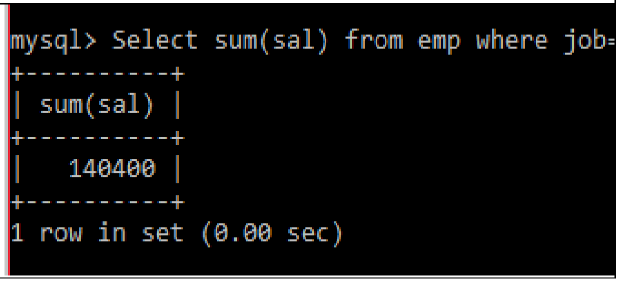
Purpose	Command	Result
To find the average salary paid to an employee	Select avg(sal) from emp;	
To find the average salary paid to MANAGERS	Select avg(sal) from emp where job='MANAGER';	

```
mysql> Select avg(sal) from emp;
+------------+
| avg(sal)   |
+------------+
| 37253.3333 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> Select avg(sal) from emp where job="MANAGER";
+------------+
| avg(sal)   |
+------------+
| 46800.0000 |
+------------+
1 row in set (0.00 sec)
```

AGGREGATE FUNCTION – SUM()

Please refer to table employee for the queries given below:

Purpose	Command	
To find the sum of salary paid to all employees	Select sum(sal) from emp;	
To find the sum of salary paid to MANAGERS	Select sum(sal) from emp where job='MANAGER';	

```
mysql> Select sum(sal) from emp;
+----------+
| sum(sal) |
+----------+
|   558800 |
+----------+
1 row in set (0.00 sec)
```

```
mysql> Select sum(sal) from emp where job=
+----------+
| sum(sal) |
+----------+
|   140400 |
+----------+
1 row in set (0.00 sec)
```

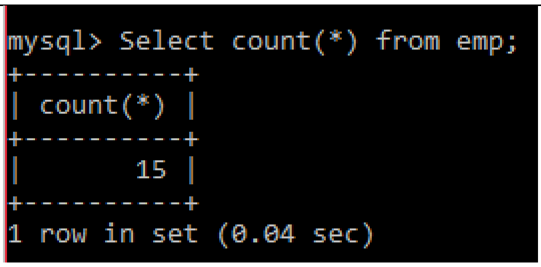
AGGREGATE FUNCTION – COUNT()

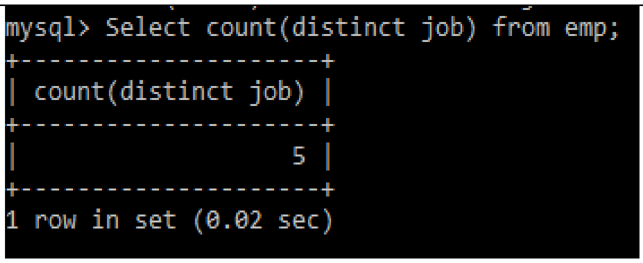
COUNT() takes one argument which can be any column name, an expression based on a column, or an asterisk (*).

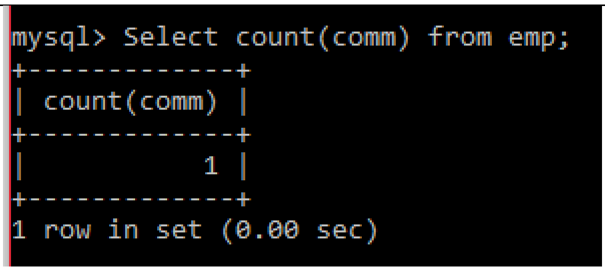
- ❑ COUNT(column name) returns the number of non-NULL values in that column
- ❑ COUNT(*) returns the total number of rows satisfying the condition

Please refer to table employee for the queries given below:

Purpose	Command	Result

To count the total number of employees in the table	Select count(*) from emp;	
		```
		mysql> Select count(*) from emp;
		+----------+
		| count(*) |
		+----------+
		|       15 |
		+----------+
		1 row in set (0.04 sec)
		```

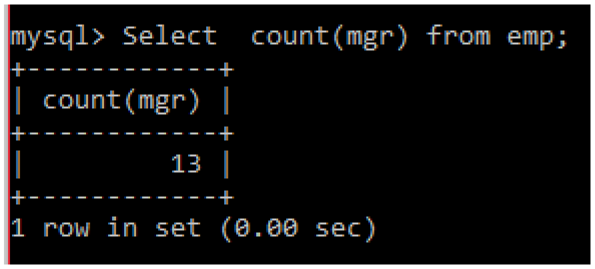
To count the different types of jobs that the table has	Select count(distinct job) from emp;	
		```
		mysql> Select count(distinct job) from emp;
		+---------------------+
		| count(distinct job) |
		+---------------------+
		|                   5 |
		+---------------------+
		1 row in set (0.02 sec)
		```

To count the number of employees who are paid commission	Select count(comm) from emp;	
		```
		mysql> Select count(comm) from emp;
		+-------------+
		| count(comm) |
		+-------------+
		|           1 |
		+-------------+
		1 row in set (0.00 sec)
		```

DIFFERENCE BETWEEN COUNT() AND COUNT(*)

- ❑ When the argument is a column name or an expression based on a column, COUNT() returns the number of non-NULL values in that column.
- ❑ If the argument is a *, then COUNT() counts the total number of rows satisfying the condition, if any, in the table.

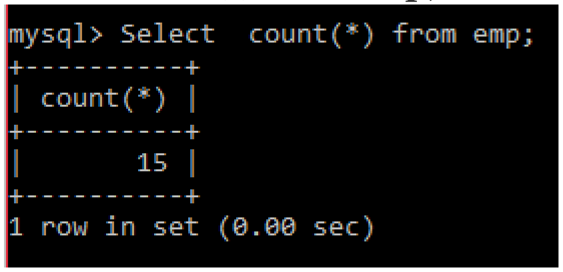
For eg.
Consider the table emp:
The command
Select count(mgr) from emp;



```
mysql> Select  count(mgr) from emp;
+------------+
| count(mgr) |
+------------+
|         13 |
+------------+
1 row in set (0.00 sec)
```

will give the output as 13 as there is NULL in one column.

However the command
Select count(*) from emp;



```
mysql> Select  count(*) from emp;
+----------+
| count(*) |
+----------+
|       15 |
+----------+
1 row in set (0.00 sec)
```

will give the output as 14 as it counts the total number of rows in the table.

GROUP BY CLAUSE

Group by clause is used together with the SQL SELECT statement to group the data of the table based on certain type. It arranges identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

The queries that contain the GROUP BY clause return only one single row for every grouped item. Optionally it is used in conjunction with aggregate functions to produce summary reports from the database.

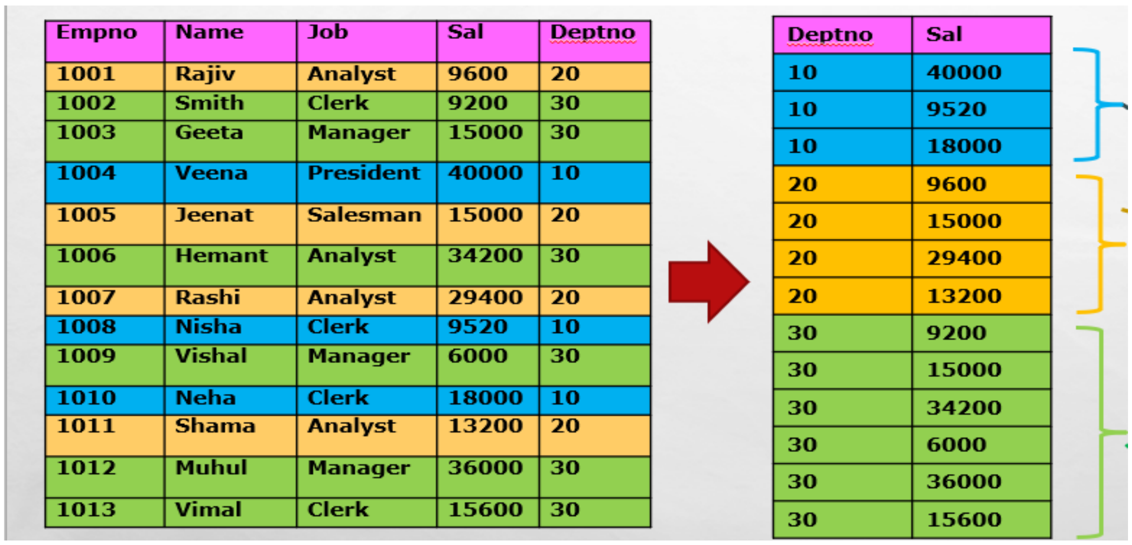
SYNTAX

SELECT statement... GROUP BY column_name1[,column_name2,...] [HAVING condition];

HERE

- "SELECT statement..." is the standard SQL SELECT command query.
- "**GROUP BY** *column_name1*" is the clause that performs the grouping based on column_name1.
- "[,column_name2,...]" is optional; represents other column names when the grouping is done on more than one column.
- "[HAVING condition]" is optional; it is used to restrict the rows affected by the GROUP BY clause. It is similar to the WHERE clause.

To understand group by clause let us consider EMP table containg Empno, Name, Job, Sal and Deptno fields. The following diagram shows how we can group the data based on the column Deptno.



Empno	Name	Job	Sal	Deptno
1001	Rajiv	Analyst	9600	20
1002	Smith	Clerk	9200	30
1003	Geeta	Manager	15000	30
1004	Veena	President	40000	10
1005	Jeenat	Salesman	15000	20
1006	Hemant	Analyst	34200	30
1007	Rashi	Analyst	29400	20
1008	Nisha	Clerk	9520	10
1009	Vishal	Manager	6000	30
1010	Neha	Clerk	18000	10
1011	Shama	Analyst	13200	20
1012	Muhul	Manager	36000	30
1013	Vimal	Clerk	15600	30

Deptno	Sal
10	40000
10	9520
10	18000
20	9600
20	15000
20	29400
20	13200
30	9200
30	15000
30	34200
30	6000
30	36000
30	15600

Here, you can see how groupby clause first makes groups based on the column Deptno. After making groups the desired query is executed on these groups and one row per group is returned.

Q WAC to display the maximum salary paid to employee of each type of job.

Select job, max(sal) from emp group by job;

Job	Max(sal)
Analyst	34200
Clerk	18000
Manager	36000
President	40000
Salesman	15000

We can include more than one aggregate function in one command.

Q WAC to display maximum, minimum, sum of salary paid to employee of each type of department.

Select deptno, max(sal), min(sal), sum(sal) from emp group by deptno;

Deptno	Max(sal)	Min(sal)	Sum(sal)
10	4000	9250	67520
20	29400	9600	67200

30	**6000**	**36000**	**116000**

Note: Please make sure to include the column name on which we are grouping in the select command.

HAVING CLAUSE

Having clause is used to apply condition on groups i.e. when condition is applied on aggregate function. We can include more than one aggregate function in one command.

EXAMPLES

Q WAC to display the maximum salary paid to employee of each type of job where maximum salary is greater than 20000.

Select job, max(sal) from emp group by job having max(sal)>20000;

Job	**Max(sal)**
Analyst	34200
Manager	36000
President	40000

Here, you can see that records with job as CLERK and SALESMAN are not present in the output as their maximum salary is less than 20000.

Q WAC to display maximum, minimum, sum of salary paid to employee of each type of department for departments 10 and 20.

Select deptno, max(sal), min(sal), sum(sal) from emp group by deptno having deptno=10 or deptno=20;

Deptno	**Max(sal)**	**Min(sal)**	**Sum(sal)**
10	**4000**	**9250**	**67520**
20	**29400**	**9600**	**67200**

Here, the output does not contain a record corresponding to deptno 30.

DIFFERENCE BETWEEN WHERE & HAVING

WHERE is used to put a condition on individual row of a table whereas HAVING is used to put condition on individual group formed by GROUP BY clause in a SELECT statement.

EQUI JOIN

Consider a table Department with the following data:

Dept_ID	Dept_Name	Location
10	Marketing	North
20	Sales	South
30	Production	East
40	HR	West

- ❑ When we extract data from two tables, they must have one column which is present in both the tables. An equi join of two tables is obtained by putting an equality condition on the Cartesian product of two tables.
- ❑ This equality condition is put on the common column of the tables and is called equi join condition.
- ❑ It is mandatory to give equi join condition when we want to extract data from two tables. Let us consider two tables: emp and dept

Q. WAC to display employee number, job, salary, department name and loction from the tables emp and dept.

Ans. **Select empno, job, sal, dname, loc from emp, dept where emp.deptno = dept.deptno;**

Note: When a column exist with same name in both the tables it is preceded by table name and dot to avoid ambiguity.

Q. WAC to display employee name, salary and depart name of all MANAGERS from emp and dept tables.

Ans. **Select ename, sal, dname from emp, dept where emp.deptno = dept.deptno and job = "MANAGER";**

Q. WAC to display employee name, salary and depart name of all employee in ascending order of salary from emp and dept tables.

Ans. **Select ename, sal, dname from emp, dept where emp.deptno = dept.deptno order by sal;**