**Connect from mysql shell to mysql server**

\connect root@localhost:3306

It will ask for password , provide the password given at time of install

**Default prompt is for js, switch to sql**

\sql

**Create database**

create database databasename ;

**Switch to created database for work**

use databasename;

**CHAR(size)**

A FIXED length string (can contain letters, numbers, and special characters). The size parameter specifies the column length in characters - can be from 0 to 255. Default is 1

**VARCHAR(size)**

A VARIABLE length string (can contain letters, numbers, and special characters). The size parameter specifies the maximum column length in characters - can be from 0 to 65535

**BINARY(size)**

Equal to CHAR(), but stores binary byte strings. The size parameter specifies the column length in bytes. Default is 1

**VARBINARY(size)**

Equal to VARCHAR(), but stores binary byte strings. The size parameter specifies the maximum column length in bytes.

**TEXT(size)**

Holds a string with a maximum length of 65,535 bytes

**BLOB(size)**

For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data

**LONGTEXT**

Holds a string with a maximum length of 4,294,967,295 characters

**LONGBLOB**

For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data

**Create Table**

SQL> create table employees(id int ,name varchar(20), balance double);

**See tables in database**

show tables;

**Create user**

CREATE USER 'anuj'@'%' IDENTIFIED BY 'scooby';

**Granting access to a user**

GRANT ALL PRIVILEGES ON testdb.\* TO 'anuj'@'%' WITH GRANT OPTION;

**Schema Description**

desc or describe command

describe employees;

**Insert row into table**

SQL> insert into employees(id,name, balance) values(1 ,'ram' , 300.50);

**Select data from table**

SQL> select \* from employee;

**Select columnnames for which result should be fetched**

SQL> select id from employee;

**Delete row from the table**

delete from employees where name='dharma';

This will delete rows for employees with name dharma

**Change Schema structure**

Use Alter command

**Rename table**

alter table oldtablename rename to newtablename;

**Adding column**

alter table employees add columnname datatype;

**Drop column**

alter table employees drop columnname ;

**SQL Constraints**

1. used to specify rules for the data in a table.
2. used to limit the type of data that can go into a table.
3. Ensures the accuracy and reliability of the data in the table

NOT NULL - Ensures that a column cannot have a NULL value

UNIQUE - Ensures that all values in a column are different

PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

FOREIGN KEY - enforces referential integrity, which essentially says that if column value A refers to column value B, then column value B must exist.

CHECK - Ensures that the values in a column satisfies a specific condition

**Add Check constraint on balance**

alter table employees add check (balance>0);

insert into employees(id,name, balance) values(8 ,'asish' , -800);

This will throw check constraint violation error because we have added check constraint on balance greater than 0

**Add Primary key**

SQL> alter table employees add constraint employees\_id\_pk primary key(id);

**Update row(s)**

SQL> update employees set name='bluetooth' where id=6;

**Create another Table**

SQL> Create table departments(id int, deptname varchar(20));

**Add Primary key constraint**

SQL> alter table departments add constraint dept\_id\_pk primary key(id);

Table altered.

Adding column

SQL> alter table employees add did int;

Table altered.

**Dropping column**

Alter table employees drop column did;

**Add Foreign key constraint**

SQL>alter table employees add constraint employees\_did\_fk foreign key(did) references departments(id);

Now you cant add a row in employees table with did value which does NOT exist as primary key in references table departments

Null Foreign key is permissible though

**Select matching rows**

Fetch rows where employees id exist in departments

SQL> select \* from employees e,departments d where e.did=d.id;

Fetch rows where employees id exist in departments and employee’s id should be 2

SQL> select \* from employees e,departments d

where e.did=d.id and e.id=2;

**IN clause**

Only find employees where id is 1 or 2 or 3 using in clause

select \* from employees where id in (1,2,3);

select \* from employees where id in (select id from managers);

**Not in clause**

Only find employees where id is NOT 1 or 2 or 3 using not in clause

select \* from employees where id not in (1,2,3);

**LIKE clause**

Find employees where names start from sc

select \* from employees where name like 'sc%';

Find employees where names ends with sc

select \* from employees where name like '%sc';

Find employees where names contain sc

select \* from employees where name like '%sc%';

**Between**

select \* from employees where salary between 1000 and 2000;

**On Clause**

**INNER JOIN**

SELECT \* FROM employees e INNER JOIN departments d ON e.did=d.id;

SELECT \* FROM employees e INNER JOIN departments d ON e.did=d.id;

ID NAME DID ID NAME

---------- -------------------- ---------- ---------- --------------------

1 akash 11 11 it

2 ab 12 12 hr

On clause with where to restrict

SELECT \* FROM employees e INNER JOIN departments d ON e.did=d.id where e.id=2;

**LEFT JOIN**

SELECT \* FROM employees e LEFT JOIN departments d ON e.did=d.id;

This will fetch the rows where there is a foreign key and primary key match and also the rows where foreign key is null

insert into employees(id,name) values(3,'chandra');

SELECT \* FROM employees e LEFT JOIN departments d ON e.did=d.id;

ID NAME DID ID deptNAME

---------- -------------------- ---------- ---------- --------------------

1 akash 11 11 it

2 ab 12 12 hr

3 chandra

**Functions**

select \* from employees;

ID NAME DID SALARY

---------- -------------------- ---------- ----------

1 akash 11 1000

2 ab 12 2000

3 chandra 3000

select sum(salary) from employees;

SUM(SALARY)

-----------

6000

select avg(salary) from employees;

AVG(SALARY)

-----------

2000

select max(salary) from employees;

MAX(SALARY)

-----------

3000

select min(salary) from employ.ees;

MIN(SALARY)

-----------

1000

Count of number of rows

select count(id) from employees;

COUNT(ID)

----------

4

**Order by**

Descending order

select \* from employees order by salary desc;

**Ascending Order**

select \* from employees order by salary;

**Group By**

The GROUP BY Statement in SQL is used to arrange identical data into groups with the help of some functions.

Total salaries paid in every department

select did, sum(salary) from employees group by did;

Average salary of employees with same name

select name, avg(salary) from employees group by name;

Count of Employees with same name

select name, count(name) from employees

group by name;

**Having**

Where clause is used to put conditions on columns, Having clause is used to put conditions on groups

Only show those names which are repeated

select name, count(name) from employees

group by name having count(name) >1;

Total salaries paid in every department

select did, sum(salary) from employees group by did;