functions in mysql:
==========

--it is used to solve a particular task.

--a sql function must return a value.

--in sql we have 2 types of functions:

1.predefined functions
2.user-defined functions (PL/sql)

1.predefined functions:
==========

--it is divided into 4 categories:

1. number functions
2. charecter functions
3. group functions or aggregrate functions
4.date functions

1. number functions:
----

a. abs(): it returns the absolute number;

ex:

>select abs(-40) from dual;
or
>select abs(-40);

b. mod(m,n) : it returns the reminder of m/n;

ex:

>select mod(10,2) from dual;

c. round(m,n)
d.truncate(m,n);

ex:
>select round(12.343823,3) from dual;

>select truncate(12.343823,3) from dual;

e. ceil()
f. floor()

greatest() least();

--it will return biggest and smallest value from the list of arguments.

ex:

>select greatest(10,12,5, 9,5) from dual; // 12

Note: from a single column if we want to get the max and min value then we should use group functions like
min(), max();

2. charecter functions:
----

1. upper()
2. lower()
3. length()
4. replace()
5. concat()
6. substr()

>select name, lenght(name) len from student;
> select substr('ratan',3,2) from dual;

date function:
----

1. sysdate() : it will return the current data and time.

ex:

>select sysdate() from dual;

2. date_format()

>select date_format(sysdate(), '%d-%m-%Y');

3. adddate()

syn:

adddate(date, INTERVAL value unit);

DAY
HOUR
YEAR
MONTH
WEEK

ex:
> select adddate(sysdate(), INTERVAL 3 DAY) from dual;

group function or aggregrate functions:
==========

--these functions operates over several values of a single column and then results in a single value.

1. max()
2. min()
3. avg()
4. sum()
5. count(*) // it will count even null value also
6. count(columnname) // it will not count null values

> select max(marks) from student;

> select count(DISTINCT marks) from student;

group by clause:
==========

--the main purpose of group by clause is to group the records/ rows.

--the clause is mostly used with the group functions only.

--it is used to divide the simillar data items into set of logical groups

short syn:
----

select col_name(s) from table group by col_name(s);

long syn:
----

select col_names
from
tablename
[where condition] ---opt
group by col_names
[having <cond>] ---opt

ex:

+------+---------+---------+--------+--------+
| eid | ename | address | salary | deptid |
+------+---------+---------+--------+--------+
| 1000 | ravi | patna | 85000 | 10 |
| 1002 | amit | delhi | 82000 | 10 |
| 1003 | mukesh | pune | 84000 | 12 |
| 1004 | dinesh | mumbai | 78000 | 10 |
| 1005 | manoj | delhi | 72000 | 12 |
| 1006 | chnadan | pune | 62000 | 12 |
| 1007 | rakesh | goa | 82000 | 13 |
+------+---------+---------+--------+--------+

--the above data is called as detailed data and after performing the group by , we will get summerized data which is usefull for analysis.

> select sum(salary) from emp; // it will calculate the salary of whole emp.

-- to calculate the sum of salaries dept wise :

>select deptid, sum(salary) from emp group by deptid;

> select deptid,min(salary), max(salary),avg(salary) ,sum(salary) from emp group by deptid;

rules:
=====

1. group functions we can not use inside the where clause.

2. other than group function all the columns mentioned inside the select clause should be there after the group by clause otherwise (oracle db will give an error, and mysql db may give unexpected result).

ex:

>select deptid,ename, min(salary), max(salary),avg(salary) ,sum(salary) from emp group by deptid;
here we may get unexpected output

>select deptid,ename, min(salary), max(salary),avg(salary) ,sum(salary) from emp group by deptid,ename;

--here we gets the expected output.
--if 2 ravi is there working on 10 th number dept then both recored will be logically grouped.

Having clause:

--after group by clause we r not allowed to use where clause, in place of where clause we should use having clause after the group by clause.

--in where clause we can not use group functions. whereas inside having clause we can use group functions.

> select deptid,ename, min(salary), max(salary),avg(salary) ,sum(salary) from emp group by deptid,ename having sum(salary) > 80000;

using where clause and having clause also:

>select deptid, sum(salary) from emp where deptid IN(10,12,14) group by deptid having sum(salary) > 80000;

Joins:
======

--Join is used to receive data from multiple tables or by using join we can combine records from multipe tables.

--there are following types of joins:

1.Inner join
2.Outer Join
- **Left Outer join**
- **Right Outer join**
- **full outer join**

3.self join
4.cross join (cartesion product)

Note: when we try to get the data from more than one table without using joining condition. then it is called cross join, in this case every record of the first table will be mapped with the every record of the second table.

--with the cross join, we don't get the meaningfull data. in order to get the meaningfull data we need to use other types of joins.

INNER JOIN:
============

--here we need to apply joining condition on the common data from both table.

--if ambiguity is there in column name (if both table having the same column name)then we need to use alias support.

--this inner join returns the matching records from the DB tables based on the common column.

> select * from dept INNER JOIN emp ON dept.did = emp.deptid;

Q/- get the emp details who is working in HR department.

> select eid, ename, address, salary from dept INNER JOIN emp ON dept.did= emp.deptid AND dept.dname = 'HR';

with help of alias:
----

select e.eid, e.ename, e.address, e.salary, d.dname from dept d INNER JOIN emp e ON d.did= e.deptid AND d.dname = 'HR';

Another syntax of INNER JOIN (without using INNER JOIN command):
==========

> select e.eid, e.ename, e.address, e.salary, d.dname from dept d,emp e where d.did= e.deptid AND d.dname = 'HR';

Outer Join:
==========

a. left outer join:

--to get the unmatched records from the left table use left outer join (it shows the full details of left table and null value for table of any unmatched record)

> select e.eid, e.ename, e.address, e.salary, d.dname from dept d LEFT OUTER JOIN emp e ON d.did= e.deptid;.

--here since Marketing dept we don't have any emp then all the dept details will be printed and unmatched emp from that dept will be null value.

+------+---------+---------+--------+-----------+
| eid | ename | address | salary | dname |
+------+---------+---------+--------+-----------+
| 1000 | ravi | patna | 85000 | HR |
| 1002 | amit | delhi | 82000 | HR |
| 1004 | dinesh | mumbai | 78000 | HR |
| 1008 | ravi | delhi | 55000 | HR |
| 1003 | mukesh | pune | 84000 | Account |
| 1005 | manoj | delhi | 72000 | Account |
| 1006 | chnadan | pune | 62000 | Account |
| 1007 | rakesh | goa | 82000 | Sales |
| NULL | NULL | NULL | NULL | Marketing |

Right Outer Join:

--to get the unmatched records from the right table use the right outer join (It shows all the details of right table and null values for the left table).

>select e.eid, e.ename, e.address, e.salary, d.dname from dept d RIGHT OUTER JOIN emp e ON d.did= e.deptid;

```
+------+---------+---------+--------+---------+
| eid  | ename   | address | salary | dname   |
+------+---------+---------+--------+---------+
| 1000 | ravi    | patna   |  85000 | HR      |
| 1002 | amit    | delhi   |  82000 | HR      |
| 1003 | mukesh  | pune    |  84000 | Account |
| 1004 | dinesh  | mumbai  |  78000 | HR      |
| 1005 | manoj   | delhi   |  72000 | Account |
| 1006 | chnadan | pune    |  62000 | Account |
| 1007 | rakesh  | goa     |  82000 | Sales   |
| 1008 | ravi    | delhi   |  55000 | HR      |
| 1009 | praksh  | delhi   |  85000 | NULL    |
| 1010 | vinay   | delhi   |  82000 | NULL    |
+------+---------+---------+--------+---------+
```

Full outer join:

--it displays the null values from both side for all the unmatched records.

Note: mysql does not support full outer join.

--in order to use full outer join in mysql, then we should use union of left outer join and right outer join.

select e.eid, e.ename, e.address, e.salary, d.dname from dept d Left OUTER JOIN emp e ON d.did= e.deptid union select e.eid, e.ename, e.address, e.salary, d.dname from dept d Right OUTER JOIN emp e ON d.did= e.deptid;

Self join:
========

--here we use joining a table to itself.

--here joining condition column must belongs to same datatype.

Note: --if we want to compare two tables same columns values then we use INNER JOIN where as if want to compare 2 diff column values within a single table then we must use self join.

****whenever a table contains hirarical data then only we allow to use
self join.**

emp ------> manager
student ---> monitor

--when we use self join, we must take the support of alias.

```
create table employee(
eid int primary key,
ename varchar(12),
salary int,
mgr int
);
```

mysql> insert into employee values(10,'Ram', 8000,null); // Ram does not have any manager

mysql> insert into employee values(11,'Ravi', 8200,10); // Ravi manager is Ram

mysql> insert into employee values(12,'amit', 8500,10); // Amit manager is Ram

mysql> insert into employee values(14,'sunil', 8700,11);// Sunil manager is Ravi

Q/- display the employee name and thier manager name.

--here we need to use self join.

```
> select e2.ename EMPLOYEE, e1.ename MANAGER from employee e1 ,employee e2
where e1.eid = e2.mgr;
```

Getting the data from 3 tables:
==========

```
create table course(

cid int primary key,
cname varchar(12),
fee int
);


create table teacher(
tid int primary key,
tname varchar(12),
taddress varchar(12),
course_id int,
foreign key (course_id) references course(cid)
);

create table student(

roll int primary key,
sname varchar(12),
saddress varchar(12),
course_id int,
foreign key (course_id) references course(cid)
);
```

Q/ get the details of all the student who is taught by Ramesh.

```
> select s.roll, s.sname, s.saddress, t.tname
from
course c INNER JOIN teacher t INNER JOIN student s
ON
c.cid = t.course_id AND c.cid = s.course_id AND t.tname = 'Ramesh';
```