adding a constraints to the existing table:

>create table a1(id int, name varchar(12));

>alter table a1 modify id int primary key;

adding FK to the existing table:

>create table b1(bid int);

>alter table b1 add foreign key (bid) references a1(id) on delete cascade;

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

--it is used to solve a perticular task.

--a sql function must return a value.

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

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

1 it is categoriezed into following categories:

1. number functions
2. charecter functions
3. date functions
4. group functions or aggregate functions
5. special functions

1. number functions:
---

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

ex:

>select abs(-10) from dual; // here dual is a sudo table, in mysql it is optional where as in oralce db it is mandatory.

>select 10+10 from dual;
or
>select 10+10;

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

ex:
>select mod(10,2) from dual;

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

ex:

select round(12.4248243,3) from dual; // 12.425

e. ceil()
f. floor()

```
greatest() and least():
--------------------------
-- it will return biggest and smallest number from the list of argument

ex:

>select greatest(10,20,40,60,12,15) from dual;

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

ex:

>select max(marks) from student;  // Ok
>select greatest(marks) from student;  // error

2. charecter functions:
-----------------------------
a. upper()
b. lower()
c. length()
d. replace()
e. concat()
f. substr()

etc..

>select name, length(name) len from student;

>select substr('ratan',3,2) from dual; // ta


date functions:
------------------

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

>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

> select adddate(sysdate(), INTERVAL 10 DAY) from dual;


Group functions/ aggregate function:
==============================

--these functions operates over the several values of a single column and then
```

results a single value.

1. max()
2. min()
3. sum()
4. avg()
5. count(*) // it will count the record, and consider the null value also
6. count(columnName) // it will not count the null value

>select max(marks), min(marks), sum(marks), avg(marks) from student;

> select count(marks) from student; // null value will not be considers

>select count(*) from student;

>select count(DISTINCT marks) from student;

Group By clause:
==========

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

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

--it is used to divide the similar data item into the set of logical groups.

short syn:

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

long syn:

```
select col_name(s)
from
tablename(s)
[where condition] ---opt
group by col_name(s)
[having <cond>] ---opt
```

ex:

eid	ename	salary	deptId
1000	ravi	80000	10
1001	amit	82000	10
1002	mukesh	84000	12
1003	dinesh	78000	10
1004	manoj	72000	12
1005	chandan	62000	14
1006	rakesh	82000	11

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

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

--to calcualte the sum of salaries dept wise.

```
 select deptid, sum(salary) from emp group by deptid;
+--------+-------------+
| deptid | sum(salary) |
+--------+-------------+
|     10 |      240000 |
|     11 |       82000 |
|     12 |      156000 |
|     14 |       62000 |
+--------+-------------+
```

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

rules of using group by:
---

1. we should not use group functions, with normal columns without using group by clause, it will not give the correct result.

```
> select deptid, sum(salary) from emp;
```

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

ex:

```
>select * from emp where avg(salary) > 80000;
ERROR 1111 (HY000): Invalid use of group function
```

3. 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 will not give the expected result.)

ex:

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

--here we have used deptid and ename other than group function sum(), so these both column name should be there inside the group by clause.

--if 2 emp with the same name works in the same deptid then their salaried will be grouped together.

Having clause:

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

--having clause is always used with the group by clause.

--in where clause we can not use group functions, where as inside the having clause we can use group functions.

```
>select deptid, min(salary), max(salary), avg(salary), sum(salary) from emp group by deptid having sum(salary) > 100000;
```

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) > 100000;
```

Special functions:
==========

```
1. IF() function:
--------------------

--if we want ot generate some data/column logically based on some decision
then we can use IF() function.

syn:

IF(condition, value-IF_true, value_IF_false);

ex:

> select IF(10 % 2 = 1, 'correct', 'incorecet') result from dual;

> select *, if(salary > 60000 , 'High Sal', 'Low sal') desciption from emp;

>select *, if(salary < 70000, salary, salary+5000) result from emp;


CASE() function:
=============

--it is similar to the switch case in any programming language.

--the CASE statments goes through the multiple conditions and return a value
when the conditon is met, other wise it will return the value defined inside the
ELSE part.

syntax:

CASE
   WHEN condition1 THEN result1
   WHEN condition2 THEN result2
   WHEN condition3 THEN result3
   WHEN condition4 THEN result4
   ELSE result;
END;


> select *, (CASE WHEN salary <= 50000 THEN 'LOW' WHEN salary > 50000 AND salary < 70000 THEN
'medium' ELSE 'HIGH' END) description from emp;


create table product
(
pid int primary key,
pname varchar(12),
price int,
quantity int
);


mysql> insert into product values(1, 'pen' ,10, 100);
Query OK, 1 row affected (0.16 sec)

mysql> insert into product values(2, 'pencil' ,5, 200);
Query OK, 1 row affected (0.15 sec)

mysql> insert into product values(3, 'notebook' ,20, 10);
Query OK, 1 row affected (0.11 sec)

mysql> insert into product values(4, 'rubber' ,12, 0);
Query OK, 1 row affected (0.17 sec)
```

```
mysql> select * from product;
+-----+----------+-------+----------+
| pid | pname    | price | quantity |
+-----+----------+-------+----------+
|   1 | pen      |    10 |      100 |
|   2 | pencil   |     5 |      200 |
|   3 | notebook |    20 |       10 |
|   4 | rubber   |    12 |        0 |
+-----+----------+-------+----------+
4 rows in set (0.00 sec)



> select *, (

CASE
 WHEN quantity >= 100 THEN 'Sufficient Stock'
 WHEN quantity < 100  AND quantity > 0 THEN 'Selling Fast'
 ELSE 'sold out'
END
) description  from product;


Join:
====


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

there are following types of joins:

1. Inner Join    // it is mostly used 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
the joining condition, then it is called as 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 any meaningfull data, in order to get the
meaningfull data we need to use other types of joins.

> select * from dept, emp;


Inner Join:
==========

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

--if ambiguity is there among the common column name then we can take the  help alias support.

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

ex:

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

with the help of alias support:
-----------------------------------

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