DML Statement

DML Stands for Data Manipulation Language and it is used for insert/delete/update or select operation on database means DML is used for performing operation on table.

Types of DML Command

1. **INSERT:** insert command or statement is used for store data in database table

There are two types of insert

a) **Wild card insert:** if we think about wild card insert we store data in all columns of the table using insert command.

Syntax:

Insert into tablename values(‘value1’,’value2’,’value…..n’)

Example: Suppose we have an employee table with field id,name,salary,age and we want to store data in all columns of the database table.

```
mysql> insert into employee values(1,'ram',20000,20);
Query OK, 1 row affected (0.04 sec)

mysql> select *from employee;
+------+------+--------+------+
| id   | name | salary | age  |
+------+------+--------+------+
|    1 | ram  |  20000 |   20 |
+------+------+--------+------+
1 row in set (0.04 sec)
```

Suppose we have four columns in the database table we insert values in all columns so this type insert is known as wild card insert.

b) **Partial insert:** Partial insert means we store data in specific column of database table called as partial insert

Syntax:

Insert into tablename (column1,column2…column)values(value1,value..n)
Example: Suppose we have a table name as an employee and a table containing 4 columns but we want to store 2 column values means we are not storing records in whole, just storing data in a few columns called as partial insert.

```
mysql> insert into employee (id,name)values(1,'ram');
Query OK, 1 row affected (0.04 sec)

mysql> select *from employee;
+------+------+--------+------+
| id   | name | salary | age  |
+------+------+--------+------+
|    1 | ram  |  20000 |   20 |
|    1 | ram  |   NULL | NULL |
+------+------+--------+------+
2 rows in set (0.00 sec)
```

Note: if we think about the above insert query we have four columns but we just insert id and name using insert command called as partial insertion.

2. DELETE: DELETE statement is used for delete record from database table or row from database table.

There are two types of delete

---

a) Wild card delete : we delete complete table records or all rows from the database table.
Syntax: delete from tablename;
Example: delete from employee;

```
mysql> delete from employee;
Query OK, 2 rows affected (0.01 sec)

mysql> select *from employee;
Empty set (0.00 sec)
```

b) Partial delete : Partial delete means deleting a specific record or row from a database table called partial delete.

Syntax: delete from tablename where condition;

```
mysql> select *from employee;
+------+-----------+--------+------+
| id   | name      | salary | age  |
+------+-----------+--------+------+
|    1 | ram       |  20000 |   20 |
|    2 | shyam     |  30000 |   20 |
|    3 | ghanshyam |  40000 |   20 |
+------+-----------+--------+------+
3 rows in set (0.00 sec)
```

Note: if we think about the above table we have three rows in the database table but we want to delete employees whose id is 2. Means we want to delete particular row then it is known as partial dependencies

```
mysql> delete from employee where id=2;
Query OK, 1 row affected (0.04 sec)

mysql> select *from employee;
+------+-----------+--------+------+
| id   | name      | salary | age  |
+------+-----------+--------+------+
|    1 | ram       |  20000 |   20 |
|    3 | ghanshyam |  40000 |   20 |
+------+-----------+--------+------+
2 rows in set (0.00 sec)
```

3. UPDATE : if we want to replace data or change data of a particular column then we can update statement

Syntax: update tablename set column=value: set new value in all cell in particular column

Example: Suppose consider we have employee table and we want to change salary of all employee

```
mysql> select *from employee;
+------+-----------+--------+------+
| id   | name      | salary | age  |
+------+-----------+--------+------+
|    1 | ram       |  20000 |   20 |
|    3 | ghanshyam |  40000 |   20 |
+------+-----------+--------+------+
2 rows in set (0.00 sec)

mysql> update employee set salary=50000;
Query OK, 2 rows affected (0.03 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select *from employee;
+------+-----------+--------+------+
| id   | name      | salary | age  |
+------+-----------+--------+------+
|    1 | ram       |  50000 |   20 |
|    3 | ghanshyam |  50000 |   20 |
+------+-----------+--------+------+
2 rows in set (0.00 sec)
```

Before Update

After Update

If we want to update particular record then we can use following syntax

Syntax: update tablename set column=value where condition

```
mysql> select *from employee;
+------+-----------+--------+------+
| id   | name      | salary | age  |
+------+-----------+--------+------+
|    1 | ram       |  50000 |   20 |
|    3 | ghanshyam |  50000 |   20 |
+------+-----------+--------+------+
2 rows in set (0.00 sec)
```

Note: Suppose consider we have table name as employee and we want to change salary of employee whose id is 1 from 50000 to 60000

```
mysql> update employee set salary=60000 where id=1;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select *from employee;
+------+-----------+--------+------+
| id   | name      | salary | age  |
+------+-----------+--------+------+
|    1 | ram       |  60000 |   20 |
|    3 | ghanshyam |  50000 |   20 |
+------+-----------+--------+------+
2 rows in set (0.00 sec)
```

After update table record

4. SELECT: select statement is used for retrieving data from database table or fetch data from database table as well as we can display anything by using select statement means we can display variable values,string etc

There are two types of select

---

1. **Wild card select :** fetch complete table using select query

Syntax: select *from tablename;

Note: * indicate wild card means fetch all records from the database table.

```
mysql> select *from employee;
+------+-----------+--------+------+
| id   | name      | salary | age  |
+------+-----------+--------+------+
|    1 | ram       |  60000 |   20 |
|    3 | ghanshyam |  50000 |   20 |
+------+-----------+--------+------+
2 rows in set (0.00 sec)
```

If we think about output we fetch all column values from the table. But if we want to fetch only name and id from table then we have to use partial selection.

2. **Partial select :** partial select means fetch specific columns or record from table

Syntax: select column1,column2….from tablename;

```
mysql> select id,name from employee;
+------+-----------+
| id   | name      |
+------+-----------+
|    1 | ram       |
|    3 | ghanshyam |
+------+-----------+
2 rows in set (0.00 sec)
```

DCL (Data Control Language): this command specify the access of database or table to end user or developer

a) Grant : grand command is used to provide access to the database by DBA means to provide permission for data access.

b) Revoke: To retain permission provided to users

TCL : TCL stands for Transaction control Language It contain three types of command

a) Commit : this command is used for save user work permanently

b) Rollback : if user perform some task but if user want to retain perform task then we can use rollback command

Note: rollback possible before commit operation.

c) Savepoint: save point means we can mark or save logic on specific point means it is partial commit

Where clause

Where clause is used for apply condition on DML statement means we can use where clause with select, delete or update statement etc

Syntax: SQL statement where condition;

Example: we want to fetch employee whose id is greater than 5

```
mysql> select  *from employee where id>5;
+------+------+--------+------+
| id   | name | salary | age  |
+------+------+--------+------+
|    6 | d    |  30000 |   30 |
|    7 | e    |  30000 |   30 |
+------+------+--------+------+
2 rows in set (0.03 sec)
```

Example: we want to delete employee whose id is greater than 5

```
mysql> delete from employee where id>5;
Query OK, 2 rows affected (0.04 sec)

mysql> select *from employee;
+------+-----------+--------+------+
| id   | name      | salary | age  |
+------+-----------+--------+------+
|    1 | ram       |  60000 |   20 |
|    3 | ghanshyam |  50000 |   20 |
+------+-----------+--------+------+
2 rows in set (0.00 sec)

mysql>
```

Logical Operator in SQL

Logical operator is used for combine more than one conditions and check it

Operator	Meaning

&& and	**Logical AND :** if all conditions are true then condition is true otherwise condition is false
|| or	**Logical OR:** if any one condition is true then condition is true otherwise condition is false
! <>>>

Example: WA SQL Query to show the employee whose salary is not 30000

```
mysql> select  *from employee where not sal=30000;
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     4 | STV  | 40000 |
|     5 | XYZ  | 50000 |
+-------+------+-------+
3 rows in set (0.02 sec)
```

```
mysql> select  *from employee where sal!=30000;
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     4 | STV  | 40000 |
|     5 | XYZ  | 50000 |
+-------+------+-------+
3 rows in set (0.00 sec)
```

```
mysql> select  *from employee where sal<>30000;
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     4 | STV  | 40000 |
|     5 | XYZ  | 50000 |
+-------+------+-------+
3 rows in set (0.00 sec)
```

IN & Between operator

IN: IN operator is used for avoid writing multiple OR condition with same column as well as it is used as conjunction in subquery

Syntax: select *from tablename where column IN(value1,value2…..value…n);

Or

select column1…….column..n from tablename where IN(value1,value2…value..n);

Example: WA SQL Query to fetch employee details whose salary is 5000 or 7000 or 20000 or 6000

```
mysql> select *from employee where sal=5000 or sal=6000 or sal=7000 or sal=20000;
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     6 | AAA  |  5000 |
|     7 | BBB  |  7000 |
+-------+------+-------+
3 rows in set (0.00 sec)
```

Note: if we think about query we use sal column four time in different conditions so better way you can use IN operator for avoiding using same column in different conditions shown in following screen shot

```
mysql> select *from employee where sal IN(5000,6000,7000,20000);
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     6 | AAA  |  5000 |
|     7 | BBB  |  7000 |
+-------+------+-------+
3 rows in set (0.00 sec)

mysql>
```

Example: WA SQL Query to show the employee details whose name is abc or mno or stv

```
mysql> select *from employee where name='abc' or name='mno' or name='stv';
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     2 | MNO  | 30000 |
|     4 | STV  | 40000 |
+-------+------+-------+
3 rows in set (0.00 sec)

mysql> select *from employee where name IN('abc','mno','stv');
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     2 | MNO  | 30000 |
|     4 | STV  | 40000 |
+-------+------+-------+
3 rows in set (0.00 sec)
```

Between operator

Between operator is used for fetch data between specified range or it shortcut of >= and <= condition

Syntax: select *from tablename where columnname between condition1 and condition

Example: WA SQL Query to fetch employee whose salary is greater than equal 7000 and less than equal 30000

```
mysql> select  *from employee where sal>=7000 and sal<=30000;
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     2 | MNO  | 30000 |
|     3 | PQR  | 30000 |
|     7 | BBB  |  7000 |
|     8 | DDD  |  8000 |
+-------+------+-------+
5 rows in set (0.00 sec)
```

```
mysql> select  *from employee where sal between 7000 and 30000;
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     2 | MNO  | 30000 |
|     3 | PQR  | 30000 |
|     7 | BBB  |  7000 |
|     8 | DDD  |  8000 |
+-------+------+-------+
5 rows in set (0.00 sec)
```

Aggregate function /group functions

Aggregate functions are the library function provided by SQL which is used to group the values using some specified column and generate the single value as a result called an aggregate function.

Types of Aggregate functions

---

a) Max : max this function is used for find the max value from column

Syntax: select max(columnname) from tablename;

Example: WA SQL query to find the max value of employee;

```
mysql> select *from employee;
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     2 | MNO  | 30000 |
|     3 | PQR  | 30000 |
|     4 | STV  | 40000 |
|     5 | XYZ  | 50000 |
|     6 | AAA  |  5000 |
|     7 | BBB  |  7000 |
|     8 | DDD  |  8000 |
+-------+------+-------+
8 rows in set (0.00 sec)

mysql> select max(sal) from employee;
+----------+
| max(sal) |
+----------+
|    50000 |
+----------+
1 row in set (0.00 sec)
```

b) Min : this function is used for finding the minimum value from a particular colum**n.**

Syntax: select min(columnname) from tablename;

```
mysql> select min(sal) from employee;
+----------+
| min(sal) |
+----------+
|     5000 |
+----------+
1 row in set (0.00 sec)
```

c) Count: this function is used for count the number of records from database table

Syntax: select count(columnname) from tablename: if we use column in select query then count function ignore the null values means not count null values just consider non null values.

Or

Select count(*) from tablename: if we use * In the count function then count function considers null as well as non null values in counting.

```
mysql> select *from employee;
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     2 | MNO  | 30000 |
|     3 | PQR  | 30000 |
|     4 | STV  | 40000 |
|     5 | XYZ  | 50000 |
|     6 | AAA  |  5000 |
|     7 | BBB  |  7000 |
|     8 | DDD  |  8000 |
|     9 | SSS  |  NULL |
|    10 | SSSD |  NULL |
|  NULL | NULL |  NULL |
+-------+------+-------+
11 rows in set (0.00 sec)
```

```
mysql> select count(sal) from employee;
+------------+
| count(sal) |
+------------+
|          8 |
+------------+
1 row in set (0.00 sec)
```

Note: if we think about above SQL query we get result 8 but we have 11 records in table but if we think sal column we have 3 null values in sal column so when we pass column name in count function then null not consider in counting.

```
mysql> select count(*) from employee;
+----------+
| count(*) |
+----------+
|       11 |
+----------+
1 row in set (0.00 sec)
```

Note: if we think about above screen shot we get 11 as result because we use * in select query and if we think about * then count function consider not null as well as null values.

d) Sum : this function is used for counting the sum of all values in a column.

Syntax: select sum(columnname) from tablename;

```
mysql> select sum(sal) from employee;
+----------+
| sum(sal) |
+----------+
|   190000 |
+----------+
1 row in set (0.00 sec)
```

e) Avg : this function is used for find the average value of particular column

Syntax: select avg(columnname) from tablename;

Example: Find average salary of employee;

```
mysql> select avg(sal) from employee;
+------------+
| avg(sal)   |
+------------+
| 23750.0000 |
+------------+
1 row in set (0.00 sec)

mysql> select sum(sal)/count(sal) from employee;
+---------------------+
| sum(sal)/count(sal) |
+---------------------+
|          23750.0000 |
+---------------------+
1 row in set (0.00 sec)
```

Group by , having clause

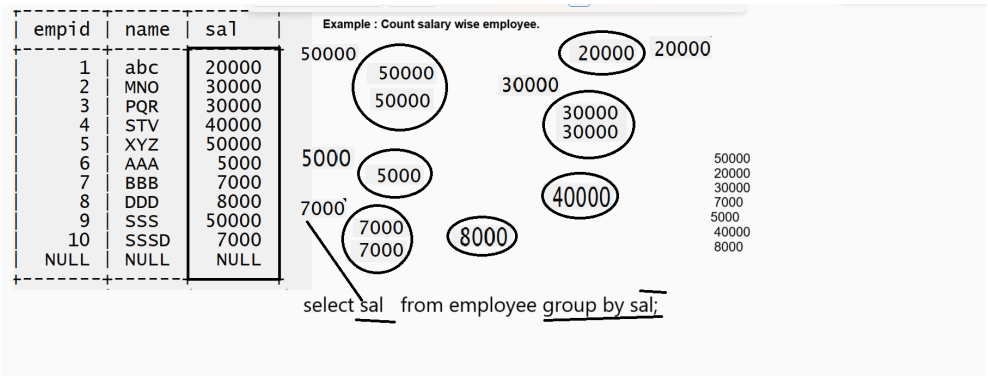
Q. What is a group by clause?

Group by clause work with specified column means using group by clause we can create the group of similar values using specified column and represent single value as group representative in output or using select query

Syntax: select column_name,aggregate function from table name group by column name;

Example:
1) Suppose consider we have employee database and we want to find the count of employee dept wise

2) Suppose consider we want to employee who is taking similar salaries
3) count the employee designation wise
4) Suppose consider we are working voting database and we want to find out the age wise voter count
Etc




```
mysql> select sal from employee group by sal;
+-------+
| sal   |
+-------+
| 20000 |
| 30000 |
| 40000 |
| 50000 |
|  5000 |
|  7000 |
|  8000 |
|  NULL |
+-------+
8 rows in set (0.05 sec)

mysql>
```

If we want to work with a group by clause we have some important points.

---

1. We can use column name in select whose name specify in group by clause
2. We can use aggregate function or group function select query if we use group by and use column in aggregate which we use in group by

Example: count the salary wise employee

```
mysql> select sal,count(sal) from employee group by sal;
+-------+------------+
| sal   | count(sal) |
+-------+------------+
| 20000 |          1 |
| 30000 |          2 |
| 40000 |          1 |
| 50000 |          2 |
|  5000 |          1 |
|  7000 |          2 |
|  8000 |          1 |
|  NULL |          0 |
+-------+------------+
8 rows in set (0.00 sec)
```

Example: find the name wise employee count

```
mysql> select name,count(name) from employee group by name;
+------+-------------+
| name | count(name) |
+------+-------------+
| abc  |           3 |
| STV  |           1 |
| XYZ  |           1 |
| AAA  |           1 |
| BBB  |           1 |
| DDD  |           1 |
| SSS  |           1 |
| SSSD |           1 |
| NULL |           0 |
+------+-------------+
9 rows in set (0.00 sec)
```

Having clause

Having clause is used to check the condition with aggregate function because we cannot use aggregate function as condition using where clause and if we want to check condition using aggregate function we can use having clause.

Syntax of having clause

Select columnname from tablename group by columnname having condition;

If we want to work with having clauses we have some important rules or points.

---

1. Having must be use with group by clause
2. Having can use group function or aggregate function in condition and if condition satisfies return true otherwise return false.
3. Having always write after group by

4. Normally we use column or group function in having which we use in select query

Example: display the name of employee whose names are same and count is more than 1

```
mysql> select name,count(name) from employee group by name having count(name)>1;
+------+-------------+
| name | count(name) |
+------+-------------+
| abc  |           3 |
| STV  |           2 |
+------+-------------+
2 rows in set (0.00 sec)

mysql>
```

Q. What is the difference between where clause and having clause?

---

1. Where clause cannot work with aggregate function or group function means we cannot use aggregate function as condition in where clause and having clause can use group function as condition

```
mysql> select name,count(name) from employee  where count(name)>1 group by name;
ERROR 1111 (HY000): Invalid use of group function
mysql>
```

   Note: if we think about above screen shot we get error at run time because we use group function with where clause which is not possible so SQL recommend us use having clause
2. Where clause can work without group by or with group by but having cannot work without group by clause
3. Where we can use for check the non grouping condition and having normally use for check the grouping condition
4. Where can use before group by and having can use after group by

Like operator

Like operator is used for pattern matching purpose or search by using patterns
Normally we use like operator with string data type
Normally we use like operators when we want to search particular data but using some specified portion of data then we can use like operator.

Example: Search employee whose name contain at least three letters or search employee whose name start with r and ends with sh

Syntax of like operator

Select *from tablename where columnname like pattern

If we want to work with like operator we have some wild card characters

1) **_ :** represent single character
2) **% :** represent 1 or more character

Example: fetch employee details whose name start with a

```
mysql> select *from employee where name like 'a%';
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
|     6 | AAA  |  5000 |
+-------+------+-------+
2 rows in set (0.01 sec)
```

Example: fetch employee details whose name ends with c

```
mysql> select *from employee where name like '%c';
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
+-------+------+-------+
1 row in set (0.00 sec)
```

Example: fetch employee details whose name start with a and ends with c

```
mysql> select *from employee where name like 'a%c';
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     1 | abc  | 20000 |
+-------+------+-------+
1 row in set (0.01 sec)
```

Example: Fetch employee whose name contains 4 letters?

```
mysql> select *from employee where name like '____';
+-------+------+-------+
| empid | name | sal   |
+-------+------+-------+
|     2 | uuuu | 30000 |
|     3 | uuuu | 30000 |
|    10 | SSSD |  7000 |
+-------+------+-------+
3 rows in set (0.00 sec)
```

Example:Fetch employee whose name contains at least 4 letters?

```
mysql> select *from employee where name like '____%';
+-------+--------+-------+
| empid | name   | sal   |
+-------+--------+-------+
|     2 | uuuu   | 30000 |
|     3 | uuuu   | 30000 |
|    10 | SSSD   |  7000 |
|    12 | ganesh |  9000 |
+-------+--------+-------+
4 rows in set (0.00 sec)
```

Order by clause

Order by clause is used for fetching data in ascending or descending order.

Syntax: select *from tablename order columnname by asc | desc;

Note: asc stands for ascending order and desc stands for descending order and by default order by work with ascending order means whe not specify any option with order by then by default order by work with asc.
Example: we want to show the employee details salary wise usign descending order.

```
mysql> select *from employee order by sal desc;
+-------+--------+-------+
| empid | name   | sal   |
+-------+--------+-------+
|     5 | STV    | 50000 |
|     9 | SSS    | 50000 |
|     4 | STV    | 40000 |
|     2 | uuuu   | 30000 |
|     3 | uuuu   | 30000 |
|     1 | abc    | 20000 |
|    12 | ganesh |  9000 |
|     8 | DDD    |  8000 |
|     7 | BBB    |  7000 |
|    10 | SSSD   |  7000 |
|     6 | AAA    |  5000 |
|  NULL | NULL   |  NULL |
+-------+--------+-------+
12 rows in set (0.00 sec)
```

Q1. explain the sequence of where,group by ,order by ,having and like?

Select *from tablename where condition conjuction columnnname like pattern group by column name having condition order by asc |desc

Example: fetch employee details whose name start with s and having same salary and salary should be greater than 15000 and arrange record salary wise and salary should be display in descending order.

mysql> select *from employee where sal IN(select sal from employee where sal>=15000 && name like 's%' group by sal having count(sal)>1 order by sal desc) order by sal desc;

```
+-------+---------+-------+
| empid | name    | sal   |
+-------+---------+-------+
|     5 | STV     | 50000 |
|     9 | SSS     | 50000 |
|     2 | sangram | 30000 |
|     3 | satish  | 30000 |
+-------+---------+-------+
4 rows in set, 1 warning (0.00 sec)
```

Constraints:

Q. What is constraints?

Constraints are the some rules and regulations which we can apply on databasse table and its column

Q. Why use constraints or What are the benefits of constraints?

---

1. Avoid store irelevant data in table column
2. Avoid to store null values if required
3. To maintain the unique ness of data
4. Provide relationship between two tables or more than two tables.
5. We can set default value for column at the time of table creation
6. We can check the data before inserting database table by specifying some conditions at the time of table creation.

etc

If we want to work with constraints in SQL we have types of constraints?

---

1. **Not null:** if we apply this constraints on column of database table then we cannot store null value in column.
   Syntax: create table tablename(columnname datatype(size) not null ….);

```
mysql> create table employee(eid int(5) not null,name varchar(200) not null, sal int(5));
Query OK, 0 rows affected, 2 warnings (0.09 sec)

mysql> insert into employee values(1,'ram',10000);
Query OK, 1 row affected (0.01 sec)

mysql> insert into employee (eid,sal)values(1,10000);
ERROR 1364 (HY000): Field 'name' doesn't have a default value
```

Note: if we think about this error Field name doesn't have default value because we apply not null constraints on name column name and when we not provide value to column then by default database engine provide null value to that column but we set rule on that column there should not null value so database engine not provide default value to that particular column so developer must be manually value to that except null

Conclusion: means if we are applying employee database table we must be known its name otherwise should not enter its data as employee record.

2. **Unique : if we apply unique constraints on column then we cannot allow duplicate values in that column**
   Example: suppose consider we want to store contact number of employee then contact of number of employee should not be duplicate
   Example: Suppose we are working company database then same name dept should be present in same company

Syntax: create table tablename(column name datatype(size) unique)

Example: mysql> create table employee(id int,name varchar(20),email varchar(20) not null unique,contact varchar(200) not null unique);
Query OK, 0 rows affected (0.11 sec)

```
mysql> insert into employee values(1,'ram','ram@gmail.com','12345');
Query OK, 1 row affected (0.04 sec)

mysql> insert into employee values(1,'ram','ram@gmail.com','12345');
ERROR 1062 (23000): Duplicate entry 'ram@gmail.com' for key 'email'
mysql>
```

Note: if we think about second insert statement we have error Duplicate entry for ram@gmail.com because we apply unique constraints email so we cannot store duplicate value in it.

3. **Primary** : primary key constraints are by default not null and unique means we cannot allow duplicate values in the primary key column as well as cannot store null values in the primary key.

Syntax: create table tablename(column_name datatype(size) primary key,.....);

```
mysql> create table employee(id int primary key,name varchar(20),email varchar(20) not null unique,contact varchar(200) not null unique);
Query OK, 0 rows affected (0.07 sec)

mysql> insert into employee values(1,'abc','abc@gmail.com','33333');
Query OK, 1 row affected (0.02 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into employee values(1,'abc','abc@gmail.com','33333');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
mysql> insert into employee values(null,'abc','abc@gmail.com','33333');
ERROR 1048 (23000): Column 'id' cannot be null
```

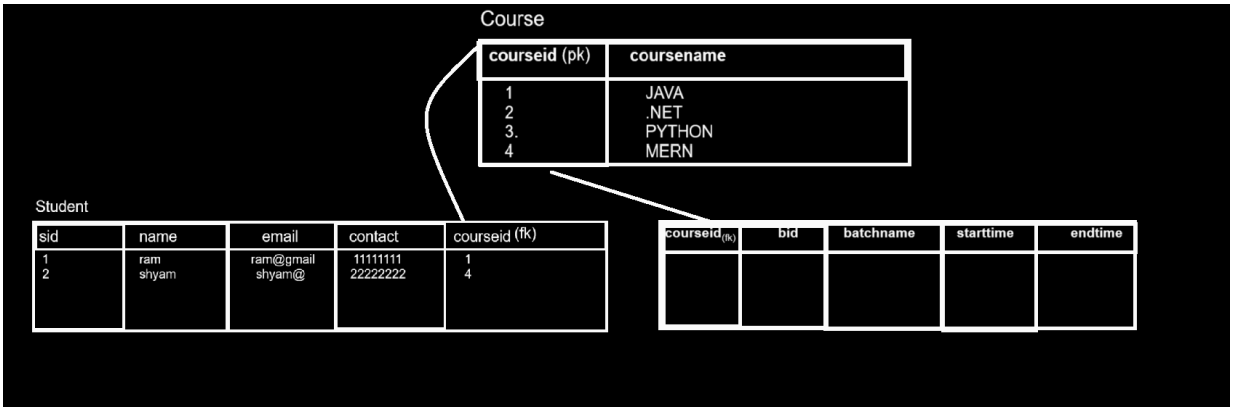
Note: if we think about second and third insert statement we get error because we try to store duplicate in primary key and we try to store null value in primary key column it is not possible

Q. What is the difference between primary key and unique key constraints?

Primary key	**Unique Key**
Primary key constraints is by default not null and unique	Unique constraints is not by default not null
Primary key constraints is not allow null values	Unique key constraints allow null values
Table can have only one primary key column	Table can have more than one unique key column

Primary key use clustered indexing	Unique used non clustered index
Primary key can use as reference in another table as foreign key	Unique key cannot use reference in another table as foreign key

4. **Foreign:** foreign key constraints is used for provide relationship between two different tables or more than two tables and every foreign key is primary key in some another table and if we use primary key as reference in another table known as foreign key and foreign key column table name known as child table and primary key column table name as parent table.



How to create foreign key

Syntax:

Create table tablename(column_name datatype(size),foreign key(columnname) references table name(primary key column name));

mysql> create table course(cid int(5) primary key,name varchar(200) not null unique);
Query OK, 0 rows affected, 1 warning (0.11 sec)

mysql> create table student(sid int(5) primary key ,name varchar(200) not null,email varchar(200) not null unique,contact varchar(200) not null unique,cid int(5),foreign key(cid) references course(cid));
Query OK, 0 rows affected, 2 warnings (0.08 sec)

```
mysql> select *from course order by cid;
+-----+--------+
| cid | name   |
+-----+--------+
|   1 | JAVA   |
|   2 | .NET   |
|   3 | PYTHON |
|   4 | MERN   |
+-----+--------+
4 rows in set (0.00 sec)

mysql> select *from student ;
+-----+------+---------------+---------+------+
| sid | name | email         | contact | cid  |
+-----+------+---------------+---------+------+
|   1 | abc  | abc@gmail.com | 343434  |    1 |
|   2 | mno  | mno@gmail.com | 3434344 |    4 |
+-----+------+---------------+---------+------+
2 rows in set (0.00 sec)
```

Note: we want to delete the java course from the course table.
mysql> delete from course where cid=1;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`sepoct2024`.`student`, CONSTRAINT `student_ibfk_1` FOREIGN KEY (`cid`) REFERENCES `course` (`cid`))
mysql>

Note: we want to delete the course whose id is 1 but we get error because cid=1 value present in its child table i.e in Student table in our database

Conclusion: if any parent reference of primary key is used by any child table or child record then we cannot delete parent record or we cannot update primary key column value before that we are required to delete or update child after that we can perform operation on parent.

```
mysql> delete from student where cid=1;
Query OK, 1 row affected (0.01 sec)

mysql> delete from course where cid=1;
Query OK, 1 row affected (0.01 sec)

mysql> _
```

Note: if we think about the above technique it is not very tedious when we have a large database with a number of child and parent relationships.

If we want to solve this problem we have three more constraints we can use the foreign key column name as on delete cascade , on update cascade, on delete set null

5. **On delete cascade:** on delete cascade we can use with foreign key column and when use on delete cascade with foreign key and if we delete parent record or primary key record then child key record automatically gets deleted.

Syntax: create table tablename(column name datatype(size),foreign key(columnname) references parenttable(primary key) on delete cascade….);

Example:

mysql> create table course(cid int(5) primary key,name varchar(200) not null unique);
Query OK, 0 rows affected, 1 warning (0.11 sec)

mysql> create table student(sid int(5) primary key ,name varchar(200) not null,email varchar(200) not null unique,contact varchar(200) not null unique,cid int(5),foreign key(cid) references course(cid) on delete cascade);
Query OK, 0 rows affected, 2 warnings (0.11 sec)

```
mysql> select *from course;
+-----+--------+
| cid | name   |
+-----+--------+
|   2 | .NET   |
|   4 | MERN   |
|   3 | PYTHON |
+-----+--------+
3 rows in set (0.00 sec)
```

```
mysql> delete from course where cid=2
mysql> select *from student;
+-----+-------+-----------------+---------+------+
| sid | name  | email           | contact | cid  |
+-----+-------+-----------------+---------+------+
|   1 | ram   | ram@gmail.com   | 12345   |    2 |
|   2 | shyam | shyam@gmail.com | 123445  |    4 |
+-----+-------+-----------------+---------+------+
2 rows in set (0.00 sec)
```

delete automatically when .NET Course delete

```
mysql> select *from course;
+-----+--------+
| cid | name   |
+-----+--------+
|   4 | MERN   |
|   3 | PYTHON |
+-----+--------+
2 rows in set (0.00 sec)

mysql> select *from student;
+-----+-------+-----------------+---------+------+
| sid | name  | email           | contact | cid  |
+-----+-------+-----------------+---------+------+
|   2 | shyam | shyam@gmail.com | 123445  |    4 |
+-----+-------+-----------------+---------+------+
1 row in set (0.00 sec)
```

Note: if we think about above screen short we delete course whose id is 2 i.e .NET course but database engine automatically delete record of .net student because we use on delete cascade with cid in student table means when cid parent delete then child record deleted automatically database engine.

6. **On update cascade:** on update cascade constraints help us to update the child key column automatically when we update the parent key record.

   Syntax: create table tablename(column name datatype(size),foreign key(columnname) references parenttable(primary key) on update cascade….);
   Note: we can combine on delete cascade and on update cascade at time with same column

   Example:
   create table course(cid int(5) primary key,name varchar(200) not null unique);

   create table student(sid int(5) primary key ,name varchar(200) not null,email varchar(200) not null unique,contact varchar(200) not null unique,cid int(5),foreign key(cid) references course(cid) on delete cascade on update cascade);

```
mysql> select *from course;
+-----+--------+
| cid | name   |
+-----+--------+
|   4 | MERN   |
|   3 | PYTHON |
+-----+--------+
2 rows in set (0.00 sec)

mysql> select *from student;
+-----+------+---------------+----------+------+
| sid | name | email         | contact  | cid  |
+-----+------+---------------+----------+------+
|   1 | abc  | abc@gmail.com | 123456   |    4 |
|   2 | mno  | mno@gmail.com | 1234456  |    4 |
|   3 | pqr  | pqr@gmail.com | 12344546 |    3 |
+-----+------+---------------+----------+------+
3 rows in set (0.00 sec)
```

Note: we want to change the course id of MERN course from 4 to 1

```
mysql> update course set cid=1 where cid=4;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select *from course;
+-----+--------+
| cid | name   |
+-----+--------+
|   1 | MERN   |
|   3 | PYTHON |
+-----+--------+
2 rows in set (0.00 sec)

mysql> select *from student;
+-----+------+---------------+----------+------+
| sid | name | email         | contact  | cid  |
+-----+------+---------------+----------+------+
|   1 | abc  | abc@gmail.com | 123456   |    1 |
|   2 | mno  | mno@gmail.com | 1234456  |    1 |
|   3 | pqr  | pqr@gmail.com | 12344546 |    3 |
+-----+------+---------------+----------+------+
3 rows in set (0.00 sec)
```

Note: if we think about right hand side output we not update the student table manually we update the course id from course table but student table get updated.**automatically his cid**

7. **On delete set null :** these constraints help us to set child column values as null when we delete parent records.

   Note: we cannot use on delete set null and on delete cascade at same time on same column

   Syntax: create table tablename(column name datatype(size),foreign key(columnname) references parenttable(primary key) on delete set null ….);

   Example
   create table course(cid int(5) primary key,name varchar(200) not null unique);

   create table student(sid int(5) primary key ,name varchar(200) not null,email varchar(200) not null unique,contact varchar(200) not null unique,cid int(5),foreign key(cid) references course(cid) on delete set null on update cascade);

```
mysql> select *from course;
+-----+--------+
| cid | name   |
+-----+--------+
|   1 | MERN   |
|   3 | PYTHON |
+-----+--------+
2 rows in set (0.00 sec)

mysql> select *from student;
+-----+------+---------------+----------+------+
| sid | name | email         | contact  | cid  |
+-----+------+---------------+----------+------+
|   1 | abc  | abc@gmail.com | 123456   |    1 |
|   2 | mno  | mno@gmail.com | 1234456  |    1 |
|   3 | pqr  | pqr@gmail.com | 12344546 |    3 |
+-----+------+---------------+----------+------+
3 rows in set (0.00 sec)
```

```
mysql> delete from course where cid=1;
Query OK, 1 row affected (0.05 sec)

mysql> select *from student;
+-----+------+---------------+----------+------+
| sid | name | email         | contact  | cid  |
+-----+------+---------------+----------+------+
|   1 | abc  | abc@gmail.com | 123456   | NULL |
|   2 | mno  | mno@gmail.com | 1234456  | NULL |
|   3 | pqr  | pqr@gmail.com | 12344546 |    3 |
+-----+------+---------------+----------+------+
3 rows in set (0.00 sec)
```

8. **Auto_increment :** auto_increment this constraints help us to increase the primary key or unique key value automatically by database engine user not provide the manually value to primary or unique key

Syntax: create table tablename(column_name datatype(size) primary key auto_increment …);

Example: we want to create a player table and generate the player id automatically.

Sample code

mysql> create table player(pid int(5) primary key auto_increment,name varchar(200),run int(5));
Query OK, 0 rows affected, 2 warnings (0.05 sec)

```
mysql> insert into player values('0','ABC',10000);
Query OK, 1 row affected (0.03 sec)

mysql> insert into player values('0','MNO',10000);
Query OK, 1 row affected (0.01 sec)

mysql> select *from player;
+-----+------+-------+
| pid | name | run   |
+-----+------+-------+
|   1 | ABC  | 10000 |
|   2 | MNO  | 10000 |
+-----+------+-------+
2 rows in set (0.00 sec)
```

Note: when we work with auto_increment then we can use ‘0’ in column not need to specify value or null in column

9. **Default :** default constraints help us to set the default value with column means when user not provide default value to

column then database engine use value set by default constraints and when user provide value to the column then database engine override user value on default value.

Syntax: create table tablename(columnname datatype(size) default value…..);

10. **Check :** check constraints help us to apply condition on table column means using check constraints we can verify the data sent by user and after that store in column means condition satisfies otherwise not .

Syntax: create table tablename(column name datatype(size) check condition);

Example: we are working on voting database and when we store voter details in table then we required to set following things

1. **Age of voter should be greater than 18**
2. **When we not provide nationality then by default it should be indian**

mysql> create table voter(vid int(5) primary key auto_increment,name varchar(200) not null,age int(5) check(age>18), nationality varchar(200) default 'indian');

Query OK, 0 rows affected, 2 warnings (0.09 sec)

```
mysql>
mysql> insert into voter (vid,name,age)values(1,'ram',20);
Query OK, 1 row affected (0.04 sec)

mysql> select *from voter;
+-----+------+------+-------------+
| vid | name | age  | nationality |
+-----+------+------+-------------+
|   1 | ram  |   20 | indian      |
+-----+------+------+-------------+
1 row in set (0.00 sec)
```

Note: if we think about above screen shot we not provide value to column nationality but database engine add default value in it i.e indian

When we provide value to nationality then database engine override our value on default value.

```
mysql> insert into voter (vid,name,age,nationality)values(2,'shyam',20,'bhartiya');
Query OK, 1 row affected (0.01 sec)

mysql> select *from voter;
+-----+-------+------+-------------+
| vid | name  | age  | nationality |
+-----+-------+------+-------------+
|   1 | ram   |   20 | indian      |
|   2 | shyam |   20 | bhartiya    |
+-----+-------+------+-------------+
2 rows in set (0.00 sec)
```

If we think about the above screen shot we provide a value to nationality as bhartiya then the database engine overrides our value on default value.

```
mysql> insert into voter (vid,name,age,nationality)values(3,'ganesh',10,'bhartiya');
ERROR 3819 (HY000): Check constraint 'voter_chk_1' is violated.
mysql>
```

If we think about the above insert statement we have an age value is 10 and we set the condition that age should be greater than 18 so this condition is not satisfied so we get an error.

Joins in SQL

Joins are used to combine more than tables using select query and generate results as a single table and if we want to perform joins on multiple tables then tables must have some relationship using primary and foreign key columns or they must have some common column between two tables or more than two tables.

Types of Joins In SQL

---

1. **Inner join :** inner join means fetch common matching data from left hand side table and right side table
   Note: here we consider Course is our left hand side table and Student is our right hand side table.

Course

courseid	coursename
1	JAVA
2	.NET
3	PYTHON
4	HADOOP

INNER JOIN

Course Name	Student Name	Student Email	Student Contact
JAVA	ABC	abc@gmail.com	22334455
JAVA	MNO	mno@gmail.com	11223344
.NET	PQR	pqr@gmail.com	22222222

Student

sid	name	email	contact	courseid
1	ABC	abc@gmail.com	22334455	1
2	MNO	mno@gmail.com	11223344	1
3	PQR	pqr@gmail.com	22222222	2
4	STV	stv@gmail.com	33333333	null

2. **Left join:** Left join means fetch all records from left hand side table and common records from right hand side table.

Course

courseid	coursename
1	JAVA
2	.NET
3	PYTHON
4	HADOOP

INNER JOIN

Course Name	Student Name	Student Email	Student Contact
JAVA	ABC	abc@gmail.com	22334455
JAVA	MNO	mno@gmail.com	11223344
.NET	PQR	pqr@gmail.com	22222222
PYTHON	**NULL**	**NULL**	**NULL**
HADOOP	**NULL**	**NULL**	**NULL**

Student

sid	name	email	contact	courseid
1	ABC	abc@gmail.com	22334455	1
2	MNO	mno@gmail.com	11223344	1
3	PQR	pqr@gmail.com	22222222	2
4	STV	stv@gmail.com	33333333	null

3. **Right join : r**ight means fetch all records from right hand side table and common records from left hand side table

Course

courseid	coursename
1	JAVA
2	.NET
3	PYTHON
4	HADOOP

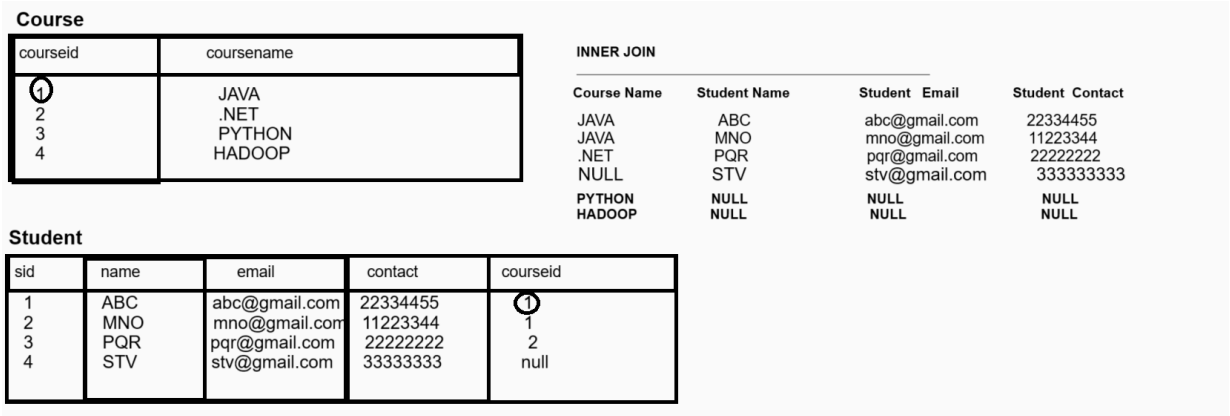
INNER JOIN

Course Name	Student Name	Student Email	Student Contact
JAVA	ABC	abc@gmail.com	22334455
JAVA	MNO	mno@gmail.com	11223344
.NET	PQR	pqr@gmail.com	22222222
NULL	STV	stv@gmail.com	333333333

Student

sid	name	email	contact	courseid
1	ABC	abc@gmail.com	22334455	1
2	MNO	mno@gmail.com	11223344	1
3	PQR	pqr@gmail.com	22222222	2
4	STV	stv@gmail.com	33333333	null

4. **Outer join :** outer join is combination of left join and right join means if we use the outer join then fetch all records from left hand side table and all records from right hand side table.



5. **Self join :** if we perform a join on the same table called as self join.

How to perform join practically

Syntax:

Select ltref.columnname,rtref.columnname from lefttable ltref join type righttable rtref on ltref.column=rtref.column;

Example of inner join

---

mysql> select c.name,s.name ,s.email from course c inner join student s on s.cid=c.cid;

```
+--------+------+---------------+
| name   | name | email         |
+--------+------+---------------+
| JAVA   | abc  | abc@gmail.com |
| JAVA   | mno  | mno@gmail.com |
| PYTHON | pqr  | pqr@gmail.com |
+--------+------+---------------+
3 rows in set (0.01 sec)
```

Example of left joins

---

mysql> select c.name,s.name ,s.email from course c left join student s on s.cid=c.cid;

```
mysql> select *from course order by cid;
+-----+--------+
| cid | name   |
+-----+--------+
|   1 | JAVA   |
|   2 | PHP    |
|   3 | PYTHON |
|   4 | MERN   |
|   5 | .NET   |
+-----+--------+
5 rows in set (0.02 sec)
```

mysql> select c.name,s.name ,s.email from course c left join student s on s.cid=c.cid;

```
mysql> select *from student order by sid;
+-----+--------+------------------+-----------+------+
| sid | name   | email            | contact   | cid  |
+-----+--------+------------------+-----------+------+
|   0 | rajesh | rajesh@gmail.com | 123456789 | NULL |
|   1 | abc    | abc@gmail.com    | 123456    |    1 |
|   2 | mno    | mno@gmail.com    | 1234456   |    1 |
|   3 | pqr    | pqr@gmail.com    | 12344546  |    3 |
|   4 | ganesh | ganesh           | 123456781 | NULL |
+-----+--------+------------------+-----------+------+
```

```
+--------+------+---------------+
| name   | name | email         |
+--------+------+---------------+
| .NET   | NULL | NULL          |
| JAVA   | abc  | abc@gmail.com |
| JAVA   | mno  | mno@gmail.com |
| MERN   | NULL | NULL          |
| PHP    | NULL | NULL          |
| PYTHON | pqr  | pqr@gmail.com |
+--------+------+---------------+
6 rows in set (0.02 sec)
```

Note: if we think about above output we fetch all records from left hand side table and common records from right hand side table.

Example of right join

```
mysql> select *from course order by cid;
+-----+--------+
| cid | name   |
+-----+--------+
|   1 | JAVA   |
|   2 | PHP    |
|   3 | PYTHON |
|   4 | MERN   |
|   5 | .NET   |
+-----+--------+
5 rows in set (0.02 sec)
```

mysql> select c.name,s.name ,s.email from course c right join student s on s.cid=c.cid;

```
mysql> select *from student order by sid;
+-----+--------+------------------+-----------+------+
| sid | name   | email            | contact   | cid  |
+-----+--------+------------------+-----------+------+
|   0 | rajesh | rajesh@gmail.com | 123456789 | NULL |
|   1 | abc    | abc@gmail.com    | 123456    |    1 |
|   2 | mno    | mno@gmail.com    | 1234456   |    1 |
|   3 | pqr    | pqr@gmail.com    | 12344546  |    3 |
|   4 | ganesh | ganesh           | 123456781 | NULL |
+-----+--------+------------------+-----------+------+
5 rows in set (0.00 sec)
```

```
+--------+--------+------------------+
| name   | name   | email            |
+--------+--------+------------------+
| NULL   | rajesh | rajesh@gmail.com |
| JAVA   | abc    | abc@gmail.com    |
| JAVA   | mno    | mno@gmail.com    |
| PYTHON | pqr    | pqr@gmail.com    |
| NULL   | ganesh | ganesh           |
+--------+--------+------------------+
5 rows in set (0.00 sec)
```

Note: if we think about above output we fetch all records from right hand side table
and common records from left hand side tables.

How to perform outer join using MYSQL

Outer join is combination of left join and right join means using outer join we can fetch all records from left hand side table all records from right hand side table

Note: if we want to work with outer join then MYSQL not provide outer keyword to us we have to perform union perform between two joins means if we want to perform outer join using MYSQL then we have write left join at left hand side of union keyword and right join at right hand side of union keyword

Syntax: select query left join union right join
Example:
mysql> select c.name,s.name ,s.email from course c left join student s on s.cid=c.cid union select c.name,s.name,s.email from course c right join student s on c.cid=s.cid;

name	name	email
JAVA	abc	abc@gmail.com
JAVA	mno	mno@gmail.com
PYTHON	pqr	pqr@gmail.com
.NET	NULL	NULL
MERN	NULL	NULL
PHP	NULL	NULL
NULL	rajesh	rajesh@gmail.com
NULL	ganesh	ganesh

8 rows in set (0.04 sec)

mysql> select *from course;

cid	name
5	.NET
1	JAVA
4	MERN
2	PHP
3	PYTHON

5 rows in set (0.00 sec)

mysql> select *from student;

sid	name	email	contact	cid
0	rajesh	rajesh@gmail.com	123456789	NULL
1	abc	abc@gmail.com	123456	1
2	mno	mno@gmail.com	1234456	1
3	pqr	pqr@gmail.com	12344546	3
4	ganesh	ganesh	123456781	NULL

5 rows in set (0.00 sec)

mysql> select c.name,s.name ,s.email from course c left join student s on s.cid=c.cid
union
select c.name,s.name,s.email from course c right join student s on c.cid=s.cid;

name	name	email
JAVA	abc	abc@gmail.com
JAVA	mno	mno@gmail.com
PYTHON	pqr	pqr@gmail.com
.NET	NULL	NULL
MERN	NULL	NULL
PHP	NULL	NULL

6 rows in set (0.00 sec)

union

name	name	email
NULL	rajesh	rajesh@gmail.com
JAVA	abc	abc@gmail.com
JAVA	mno	mno@gmail.com
PYTHON	pqr	pqr@gmail.com
NULL	ganesh	ganesh

5 rows in set (0.00 sec)

Example: WA SQL Query to fetch a course whose admission count is 0 or no student enroll in that course.

```
mysql> select c.name as 'Course Name',count(s.cid) from course c left join student s on c
.cid=s.cid group by c.cid having count(s.cid)=0;
+-------------+--------------+
| Course Name | count(s.cid) |
+-------------+--------------+
| PHP         |            0 |
| MERN        |            0 |
| .NET        |            0 |
+-------------+--------------+
3 rows in set (0.03 sec)
```

Example: WAP SQL Query to fetch course name start with j and having more than 1 admission

mysql> select c.name as 'Course Name',count(s.cid) from course c left join student s on c.cid=s.cid where c.name like 'j%' group by c.cid having count(s.cid)>1;

```
+-------------+--------------+
| Course Name | count(s.cid) |
+-------------+--------------+
| JAVA        |            2 |
+-------------+--------------+
1 row in set (0.03 sec)
```

Example: WAP SQL Query to fetch course name start with j and having more than 1 admission and fees more than 10000

mysql> select c.name as 'Course Name',count(s.cid) from course c left join student s on c.cid=s.cid where fees>10000 and c.name like 'j%' group by c.cid having count(s.cid)>1;

Output

```
+-------------+--------------+
| Course Name | count(s.cid) |
+-------------+--------------+
| JAVA        |            2 |
+-------------+--------------+
1 row in set (0.00 sec)
```

Example: WA SQL Query to fetch course whose contain more than one student and fees should be greater than 3000 and arrange student records in descending order course countwise

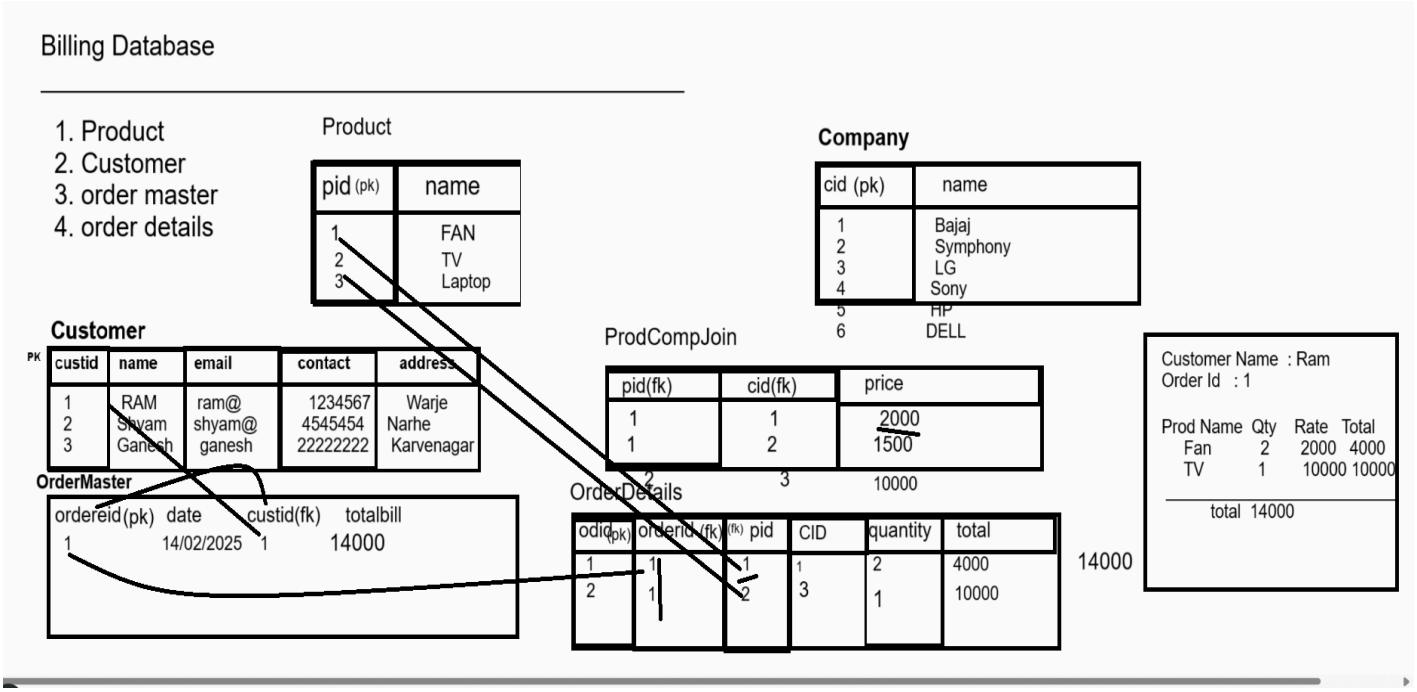
Query solution

mysql> select c.name as 'Course Name',count(s.cid) from course c left join student s on c.cid=s.cid where fees>3000 and c.name like 'j%' group by c.cid having count(s.cid)>1 order by count(c.cid) desc;

```
+-------------+--------------+
| Course Name | count(s.cid) |
+-------------+--------------+
| JAVA        |            3 |
| JS          |            2 |
+-------------+--------------+
2 rows in set (0.00 sec)
```

How to perform between more than two tables.

Case Study 1: For Join Practice in Class Room by Giri Sir



Database tables

```
mysql> create table product(pid int(5) primary key
auto_increment,name varchar(200) not null unique);


mysql> create table company(cid int(5) primary key
auto_increment,name varchar(200) not null unique);
Query OK, 0 rows affected, 1 warning (0.10 sec)

mysql> create table prodcompjoin(pid int(5),foreign key(pid)
references product(pid) on delete cascade on update cascade,cid
int(5),foreign key(cid) references company(cid) on delete cascade on
update cascade);
Query OK, 0 rows affected, 2 warnings (0.10 sec)


mysql> create table customer(cid int(5) primary key
auto_increment,name varchar(200) not null,email varchar(200)
unique,contact varchar(10) not null unique,address varchar(2000));
Query OK, 0 rows affected, 1 warning (0.10 sec)

mysql> create table ordermaster(oid int(5) primary key
auto_increment,odate date,cid int(5),foreign key(cid) references
customer(cid) on delete cascade on update cascade,totalbill int(5)
default 0);
Query OK, 0 rows affected, 3 warnings (0.09 sec)

mysql> create table orderdetails(tid int(5) primary key
auto_increment,oid int(5),foreign key(oid) references
ordermaster(oid),pid int(5),foreign key(pid) references
product(pid),cid int(5),foreign key(cid) references
company(cid),quantity int(5),total int(5));
Query OK, 0 rows affected, 6 warnings (0.10 sec)
```

Practice query

---

1. Fetch all product details
2. Fetch all company details
3. Fetch company wise product count
4. Fetch product wise company count
5. Fetch product whose price is same
6. Fetch product details whose company name start with b and product count is more than 2
7. Fetch customer list whose name start with r and at least name contain 3 or more than three letters
8. Fetch customer wise order details
9. Fetch order wise product details
10. Fetch order details bill wise in descending order
11. Give discount 20% on order whose bill amount is more 5000
12. Give 15% discount to customer who complete more than 1 order in shop
13. Fetch order detail customer wise in bill wise descending order

Implementations

1.Fetch all product details

```
ysql> select *from product order by pid;
-----+--------+
 pid | name   |
-----+--------+
   1 | FAN    |
   2 | TV     |
   3 | Laptop |
-----+--------+
3 rows in set (0.00 sec)
```

2. Fetch all company details

```
mysql> select *from company order by cid;
+-----+----------+
| cid | name     |
+-----+----------+
|   1 | Bajaj    |
|   2 | Symphony |
|   3 | LG       |
|   4 | SONY     |
|   5 | HP       |
|   6 | DELL     |
+-----+----------+
6 rows in set (0.02 sec)
```

3. Fetch company wise product count

mysql> select c.name as 'company name' ,count(cpj.cid) from product p right join prodcompjoin cpj on p.pid=cpj.pid right join company c on c.cid=cpj.cid group by c.cid;

```
+---------------+-----------------+
| company name  | count(cpj.cid)  |
+---------------+-----------------+
| Bajaj         |               1 |
| DELL          |               0 |
| HP            |               0 |
| LG            |               1 |
| SONY          |               0 |
| Symphony      |               2 |
+---------------+-----------------+
6 rows in set (0.00 sec)
```

4. Example find the product wise company count.

```
mysql> select *from product;
+-----+--------+
| pid | name   |
+-----+--------+
|   1 | FAN    |
|   3 | Laptop |
|   2 | TV     |
+-----+--------+
3 rows in set (0.00 sec)
```

```
mysql> select *from company;
+-----+----------+
| cid | name     |
+-----+----------+
|   1 | Bajaj    |
|   6 | DELL     |
|   5 | HP       |
|   3 | LG       |
|   4 | SONY     |
|   2 | Symphony |
+-----+----------+
6 rows in set (0.00 sec)
```

```
mysql> select *from prodcompjoin;
+------+------+-------+
| pid  | cid  | price |
+------+------+-------+
|    1 |    1 |  2000 |
|    1 |    2 |  1500 |
|    2 |    3 | 10000 |
|    2 |    2 | 15000 |
+------+------+-------+
4 rows in set (0.00 sec)
```

mysql> select p.name as 'product name',count(pcj.pid) "company count " from product p left join prodcompjoin pcj on p.pid=pcj.pid left join company c on pcj.cid=c.cid group by pcj.pid;

5. Fetch product details whose price is the same.

mysql> select p.name from product p inner join prodcompjoin pcj on p.pid=pcj.pid where pcj.price IN(select pcj.price from product p inner join prodcompjoin pcj on p.pid=pcj.pid group by pcj.price having count(pcj.price)>1);

```
mysql> select *from product;
+-----+--------+
| pid | name   |
+-----+--------+
|   1 | FAN    |
|   3 | Laptop |
|   2 | TV     |
+-----+--------+
3 rows in set (0.00 sec)

mysql> select *from prodcompjoin;
+------+------+-------+
| pid  | cid  | price |
+------+------+-------+
|    1 |    1 |  2000 |
|    1 |    2 |  1500 |
|    2 |    3 |  2000 |
|    2 |    2 | 15000 |
|    3 |    5 | 60000 |
+------+------+-------+
5 rows in set (0.00 sec)
```

2000 (2)

1500(1)

15000(1)

60000(1)

select p.name from product p inner join prodcompjoin pcj on p.pid=pcj.pid where pcj.price IN(2000);
or
;

6.Fetch product details whose company name start with b and product count is more than 2

mysql> select p.name,pcj.price from product p inner join prodcompjoin pcj on p.pid=pcj.pid inner join company c on pcj.cid=c.cid where c.name IN(select c.name from product p inner join prodcompjoin pcj on p.pid=pcj.pid inner join company c on pcj.cid=c.cid where c.name like 'b%' group by p.pid having count(pcj.pid)>1);select c.name from product p inner join prodcompjoin pcj on p.pid=pcj.pid inner join company c on pcj.cid=c.cid where c.name like 'b%' group by p.pid having count(pcj.pid)>1;

7.Fetch customer list whose name start with r and at least name contain 3 or more than three letters

```
mysql> select *from customer where name like 'r__%';
+-----+------+----------------+---------+---------+
| cid | name | email          | contact | address |
+-----+------+----------------+---------+---------+
|   1 | ram  | ram@gmail.com  | 123456  | Warje   |
+-----+------+----------------+---------+---------+
1 row in set (0.00 sec)

mysql>
```

8. Fetch Customer wise order details?

mysql> select c.name,om.odate,od.quantity,p.name,cm.name,od.total,(od.total/od.quantity) from customer c inner join ordermaster om on c.cid=om.cid inner join orderdetails od on od.oid=om.oid inner join product p on od.pid=p.pid inner join company cm on od.cid=cm.cid;

name	odate	quantity	name	name	total	(od.total/od.quantity)
ram	2025-02-14	2	FAN	Bajaj	4000	2000.0000
ram	2025-02-14	1	FAN	Symphony	2000	2000.0000
ram	2025-02-14	1	TV	LG	10000	10000.0000

3 rows in set (0.04 sec)

9. Give 20% on order whose bill amount is more than 5000?

mysql> select om.odate,c.name ,om.totalbill , om.totalbill-(om.totalbill*20/100) from customer c inner join ordermaster om on om.cid=c.cid;

odate	name	totalbill	om.totalbill-(om.totalbill*20/100)
2025-02-14	ram	16000	12800.0000

1 row in set (0.00 sec)

Q10.Give 15% discount to customer who complete more than 1 order in shop

mysql> select om.totalbill ,om.totalbill-(om.totalbill*15/100),c.name,om.odate from customer c inner join ordermaster om on c.cid=om.cid where c.cid IN(select cid

from ordermaster group by cid having count(cid)>1) order by odate desc limit 1;

Output

```
+-----------+-----------------------------------+------+------------+
| totalbill | om.totalbill-(om.totalbill*15/100) | name | odate      |
+-----------+-----------------------------------+------+------------+
|     10000 |                         8500.0000 | ram  | 2025-02-17 |
+-----------+-----------------------------------+------+------------+
```

Normalization in Database

Normalization technique in databases where we can decompose the large database table into smaller database tables and provide relationships between them to avoid data redundancy or duplication called normalization.

Q. What are the benefits of normalization?

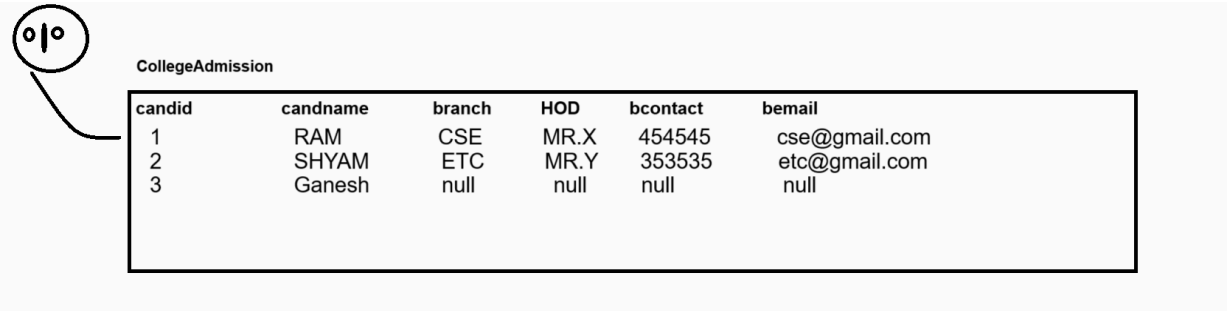
---

1. **Avoid data duplication:** if we normalize the database then we can avoid data duplication or redundancy.
2. **Provide integrity:** when we divide the larger tables into small tables and manage relationship between them then we can avoid insert/deletion or updation anomalies or confusion which occur without normalization
3. **Provide isolation for managing the data:** we can perform different operations on different types of tables and so the benefit is one table operation may not affect another table.

Q. What happens if we do not normalize databases?

If we not normalize your database then we have some anomalies or there may be some anomalies like as

1. **Insertion anomalies :** insertion anomalies means if not normalize the database and try to insert a record in it then there may be some problems we will discuss using examples.

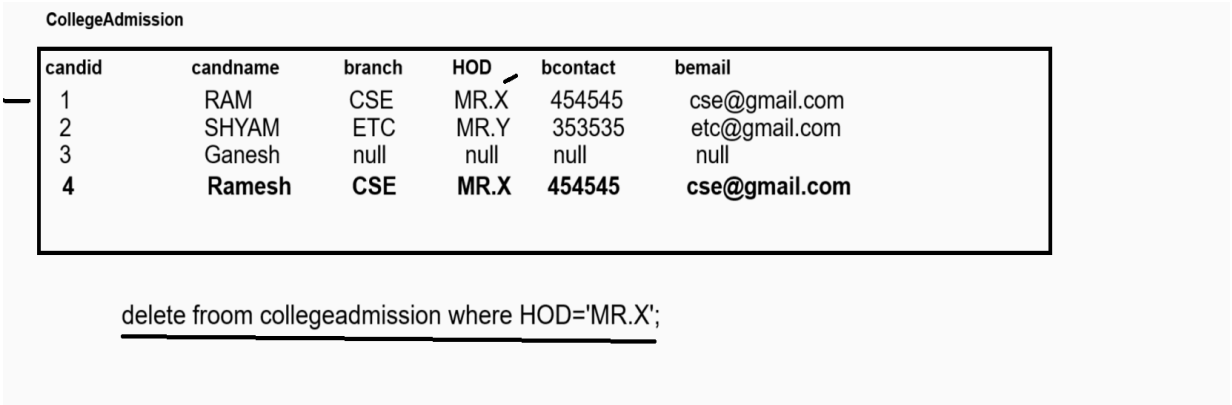


CollegeAdmission

candid	candname	branch	HOD	bcontact	bemail
1	RAM	CSE	MR.X	454545	cse@gmail.com
2	SHYAM	ETC	MR.Y	353535	etc@gmail.com
3	Ganesh	null	null	null	null

Note: if we think about the above database table we have the student name as ganesh but ganesh is confused about branch selection so we require set branch info null with ganesh or do not register ganesh in the system so this is an insertion anomaly.

2. **Deletion anomalies : we will using Example.**



CollegeAdmission

candid	candname	branch	HOD	bcontact	bemail
1	RAM	CSE	MR.X	454545	cse@gmail.com
2	SHYAM	ETC	MR.Y	353535	etc@gmail.com
3	Ganesh	null	null	null	null
4	**Ramesh**	**CSE**	**MR.X**	**454545**	**cse@gmail.com**

delete froom collegeadmission where HOD='MR.X';

Note: if we think about above table we consider MR.X want to resign his job from current organization so we want to delete record of MR.X using statement delete from collegeadmisission where HOD=MR.X so here we try to delete record of MR.X but all students of CSE dept record also deleted means we can say HOD resign job and student also stop learning or leave the college means here when we delete record of HOD then student record also get affected but when we delete record of HOD then student record should not delete this is called as deletion anomalies

3. **Updation anomalies : w**hen we try to update the single column value then all remaining column values get updating unnecessarily

Suppose consider we have hire MR.Z as replacement to MR.x then we want to update the MR.Z record on MR.X record
And MR.Z is hod of CSE and if CSE contain 1000 students and when update the column HOD value then database engine internally update 1000 records

If we want to work with normalization in DBMS we have the following types.

1. **First Normal Form**

   Rules of First Normal Form

   a) Column must be unique
   b) Column must have atomic values
   c) Column sequence matter
   d) Domain value of column must be same

Admission

aid	name	coursename
1	RAM	C
2	SHYAM	C,C++
3	Ganesh	C,C++,JAVA

1. **Column must be unique**
2. **Column must have atomic values**
3. **Column sequence matter**
4. **Domain value of column must be same**

Note: if we think about above database table it is not in first normal form because above table not satisfy the 2nd rule
Because Admission table coursename column store more than one values by separating command and first normal form say column must have atomic values.**or single value and we not satisfy 2nd rule**

Note: if we want to manage the above table using the first normal form we must have to store unique or atomic values in the column so your table looks as .

Admission

aid	name	coursename
1	RAM	C
2	SHYAM	C
2	SHYAM	C++
3	Ganesh	C
3	Ganesh	C++
3	Ganesh	JAVA

Note: if we think about above database table it is in first normal form because satisfy all rules of first normal form but there is data duplication so can avoid duplication by store separate info of student and course in different tables and provide relationship between

If we think about above table it create many to many relationship

Student

sid	name
1	RAM
2	SHYAM
3	Ganesh

CouseStudJoin

sid	cid
1	1
2	1
2	2
3	1
3	2
3	3

Course

cid	name
1	C
2	C++
3	JAVA

2. Second Normal Form

Rules of second normal form

a) Table must be in first normal form

b) Table should not have partial dependencies

Note: before understanding partial dependency we must know what dependency is.

Q. What is dependency on the database table?

If a particular column is responsible for identifying the complete record/row/tuple called as dependency and normally this dependency is achieved by using the primary key column.

Q. What are partial dependencies?

Partial dependencies means if non prime attribute is dependent on half portion of candidate key called as partial dependencies

Q. What is the candidate key?

Candidate key is a minimal set of columns or attributes which is responsible for identifying uniqueness of record called as candidate key.

Example for understanding concept of partial dependency

Student

sid (pk)	name
1	RAM
2	SHYAM

Subject

subid (pk)	subname
1	C
2	C++
3	JAVA

Score

non prime attribute

sid(fk)	subid(fk)	marks	teachname
1	1	70	C Teacher
1	2	60	CPP Teacher
2	1	80	C Teacher
2	2	70	CPP Teacher

If we think about above database we have score table which contain two foreign keys sid and subid (i.e primary key in another table) and we create combination of sid and subid for identify non prime attribute means here we can say sid and subid is candidate key
And in this table we have two non prime attribute i.e marks and teachname
We want to identify marks column
Suppose we have question

Q. 70 marks obtained?

We need to know the two things for identity above marks
a) Subject name
b) Student name

For identify subject name in score table we have subid and for identity student name we have sid column means marks column value can properly identity by using combination of sid and subid

means marks column is dependent on complete candidate key so here in score table marks column not create partial dependency

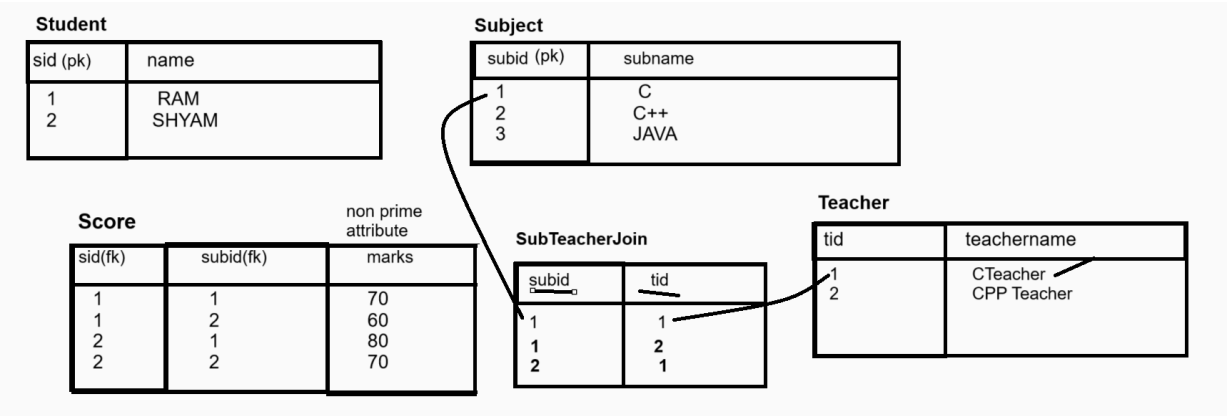
We have one more non primary key column or non prime attribute i.e teachname and we want to identity

C teacher subject name?

If we want to identify subject of teacher then we need to know subid only according to our database not need to know the student id i.e sid means here non primary key column teachname identity is dependent on only subid means dependent on half portion of candidate key not complete portion of candidate key means here we can say teachname column key partial dependency in score table and result is teachname is repeatedly occur in teachname column means we have some duplication is there so we can say score table is not in second normal form because it contain partial dependency So if we want to arrange the score table in the second normal form we must remove the partial dependency from the score table.

How can we remove the partial dependency from the score table in our example?

Suppose if we have scenario single teacher can teach multiple subject or single subject teach by multiple teachers i.e many to many relationship between teacher and subject then we have to create separate table for teacher and provide relationship between subject and teacher by using third intermediate table so your database table like as



Second scenario if your requirement is that a single teacher can teach only one subject means we have a one to one relationship then we are not required to create a separate table for teachers just you can create a column for teacher in subject table.

Student

sid (pk)	name
1	RAM
2	SHYAM

Subject

subid (pk)	subname	teachname
1	C	C Teacher
2	C++	CPP Teacher
3	JAVA	J Teacher

Score

non prime attribute

sid(fk)	subid(fk)	marks
1	1	70
1	2	60
2	1	80
2	2	70

3. Third Normal Form

Rules

---

a) Table must be in second normal form
b) Table should not have transitive dependency

Q. What is transitive dependency?

Transitive dependency means if a non prime attribute is dependent on a non prime attribute called as transitive dependency.

Student

sid (pk)	name
1	RAM
2	SHYAM

Subject

subid (pk)	subname	teachname
1	C	C Teacher
2	C++	CPP Teacher
3	JAVA	J Teacher

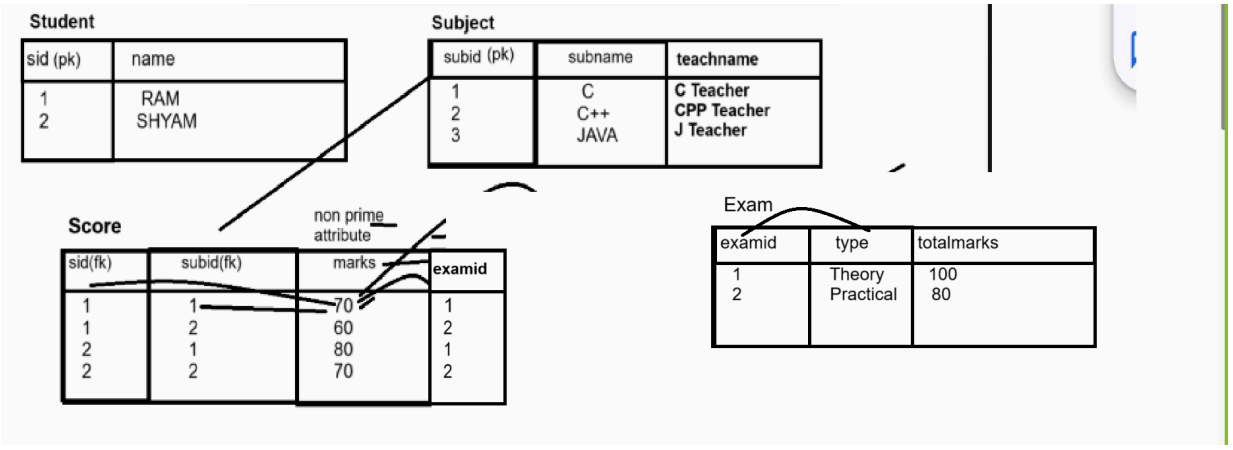
Score

non prime attribute

sid(fk)	subid(fk)	marks	exam type	total marks
1	1	70	Theory	100
1	2	60	Practical	80
2	1	80	Theory	100
2	2	70	practical	80

1. kise
2. konse subject me
3. konse exam
4. total marks

If we think about database table we can score table is not third normal form it contain transitive dependency because if we think about score table it contain three non prime column total marks , examtype and marks so here marks column is dependent on examtype and totalmarks also dependent on examtype here marks is non prime column and examtype is also non prime attribute so here one prime attribute is dependent on another non primary attribute called as transitive dependency so if we want to avoid this problem create separate table exam and store examtype and total marks in it and share the exampid as child relation in score table shown in following code



Note: remaining normal form we will discuss after SQL/PLSQL all topics

4. Boycee Code or 3.5 normal form
5. Fourth Normal form
6. Fifth Normal form

View In SQL

View in SQL is like a virtual table means view does not store permanently just view can retrieve data from actual and show to the end user.

Q. Why do we use views if we have a database table?

Restricting data access: view provide additional level of table security by restricting access to a predetermined set of rows and column of table

Hiding complex queries & simplifying user commands : means if we have some complex query like as using join or subquery etc so better way can hide the that complex query in view and when we want to execute that query then we can execute view

Multiple view facility : we can apply more than one view on a single table.

How to create view in MYSQL

If we want to create view in MYSQL we have following syntax

Syntax: create view viewname as select query;

```
mysql> create view custview as select name,contact from customer;
Query OK, 0 rows affected (0.12 sec)
```

```
mysql> select *from custview;
+--------+----------+
| name   | contact  |
+--------+----------+
| ram    | 123456   |
| shyam  | 1234567  |
| ganesh | 12345697 |
| ui     | 45454545 |
+--------+----------+
4 rows in set (0.00 sec)
```

We can delete data from original table using view

```
mysql> delete from custview where name='ui';
Query OK, 1 row affected (0.09 sec)
```

```
mysql> delete from custview where name='ui';
Query OK, 1 row affected (0.09 sec)

mysql> select *from customer;
+-----+--------+------------------+----------+------------+
| cid | name   | email            | contact  | address    |
+-----+--------+------------------+----------+------------+
|   1 | ram    | ram@gmail.com    | 123456   | Warje      |
|   2 | shyam  | shyam@gmail.com  | 1234567  | Narhe      |
|   3 | ganesh | ganesh@gmail.com | 12345697 | Karvenagar |
+-----+--------+------------------+----------+------------+
3 rows in set (0.00 sec)

mysql>
```

We can update data in database table by using view

```
mysql> update custview set name='ganu' where contact='1234567';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select *from customer;
+-----+--------+------------------+----------+------------+
| cid | name   | email            | contact  | address    |
+-----+--------+------------------+----------+------------+
|   1 | ram    | ram@gmail.com    | 123456   | Warje      |
|   2 | ganu   | shyam@gmail.com  | 1234567  | Narhe      |
|   3 | ganesh | ganesh@gmail.com | 12345697 | Karvenagar |
+-----+--------+------------------+----------+------------+
3 rows in set (0.00 sec)

mysql>
```

Note: you can hide the complex query using view

mysql> create view test as select c.name,c.email,c.contact,m.odate from customer c inner join ordermaster m on c.cid=m.cid;
Query OK, 0 rows affected (0.04 sec)

```
mysql> select *from test;
+------+------------------+---------+------------+
| name | email            | contact | odate      |
+------+------------------+---------+------------+
| ram  | ram@gmail.com    | 123456  | 2025-02-14 |
| ram  | ram@gmail.com    | 123456  | 2025-02-17 |
| ganu | shyam@gmail.com  | 1234567 | 2025-02-17 |
+------+------------------+---------+------------+
3 rows in set (0.00 sec)
```

How to drop the view

If we want to drop the view we have to use the following syntax.
Syntax: drop view viewname;

```
mysql> drop view test;
Query OK, 0 rows affected (0.05 sec)

mysql> select *from test;
ERROR 1146 (42S02): Table 'billing.test' doesn't exist
mysql>
```

How to update the view

If we want to change the view structure we can modify the view and if we want to modify the view we have to use following syntax

Syntax: create or replace view viewname as select query;

```
mysql> create or replace view storecust as select name,email,contact from customer;
Query OK, 0 rows affected (0.01 sec)

mysql> select *from storecust;
+--------+------------------+----------+
| name   | email            | contact  |
+--------+------------------+----------+
| ram    | ram@gmail.com    | 123456   |
| ganu   | shyam@gmail.com  | 1234567  |
| ganesh | ganesh@gmail.com | 12345697 |
+--------+------------------+----------+
3 rows in set (0.00 sec)

mysql>
```

Procedure

Procedure is a set of SQL statements or it is group of SQL statement which we can reuse more than one time
Means once we write logic in procedure we can execute it more than one time.

How to create procedure in MYSQL;

Steps.

1. **Set delimiters**

   Q. What are delimiters?

   Delimiter is character in SQL which decide the end of SQL statement and by default database use semicolon as delimiter character
   If we want to set delimiter we have some following syntax
   Syntax: delimiter character ;

Note: when we use procedure/trigger/function then we must be set delimiter other than semi colon
Because when we write logic in procedure then we have multiple statements and we are required to end that statement semicolon then database engine cannot predicate end of procedure so we need to set delimiter other than semi colon.

2. **Create procedure using following syntax**

If we want to create procedure we have following syntax

```
Delimiter character;
Create procedure procedurename(IN | OUT  | IN OUT
variablename datatype(size))
Begin
   Write here your logics
End
Character
```

Syntax: if we think about above syntax we have create is DDL statement which is used for create procedure using procedure keyword and procedurename is user choice means user can give any his procedure and procedure has three types of parameter
IN : this is default of parameter in procedure using this parameter we pass value to procedure as input parameter
OUT: OUT is parameter we use in procedure which help us to get data from procedure definition to procedure calling point like as return statement in c language
IN OUT: IN OUT parameter means single parameter can work as input parameter as well as output parameter

Example: we want to create a procedure to accept the product data and store it in a database table.

```
mysql> delimiter //
mysql> create procedure storeprod(prodid int(5),prodname varchar(200))
    -> begin
    -> insert into product values(prodid,prodname);
    -> end
    -> //
Query OK, 0 rows affected, 1 warning (0.05 sec)

mysql>
```

Once we create procedure we can reuse it and if we want to reuse procedure we have to call the procedure

3. **Call the procedure if we want to reuse it.**

If we want to reuse procedure we have to use following syntax
Syntax to call procedure: call procedure name(parameters);

```
mysql> call storeprod(4,'Keyboard')//
Query OK, 1 row affected (0.09 sec)
```

```
mysql> select *from product;
+-----+----------+
| pid | name     |
+-----+----------+
|   1 | FAN      |
|   4 | Keyboard |
|   3 | Laptop   |
|   5 | Mouse    |
|   2 | TV       |
+-----+----------+
5 rows in set (0.00 sec)

mysql>
```

Example: we want to create a procedure to store company information.

```
mysql> delimiter //
mysql> create procedure storecomp(compid int(5),compname varchar(200))
    -> begin
    -> insert into company values(compid,compname);
    -> end
    -> //
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql>
```

```
mysql> call storecomp(7,'Wipro');
    -> //
Query OK, 1 row affected (0.04 sec)
```

How to work with out parameter using procedure

Example: we want to pass product id to procedure as input parameter and we want to get product name as output parameter.

Steps to work with OUT parameter using procedure

---

1. **Create procedure and use OUT keyword with out parameter**

```
mysql>
mysql> delimiter $$
mysql> create procedure getProdName(IN prodid int(5),OUT prodName varchar(200))

    -> begin
    -> 
```

2. **Use INTO keyword for copy data in OUT parameter**
   INTO keyword normally use for copy the data in out parameter

```
mysql> delimiter $$
mysql> create procedure getProdName(IN prodid int(5),OUT prodName varchar(200))

    -> begin
    -> select name INTO prodName from product where pid=prodid;
    -> end
    -> $$
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> 
```

Note: if we think about select statements we have a name column and we want to copy the value of the name column into the prodName OUT parameter.

3. **End the procedure**

```
mysql> delimiter $$
mysql> create procedure getProdName(IN prodid int(5),OUT prodName varchar(200))

    -> begin
    -> select name INTO prodName from product where pid=prodid;
    -> end
    -> $$
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> _
```

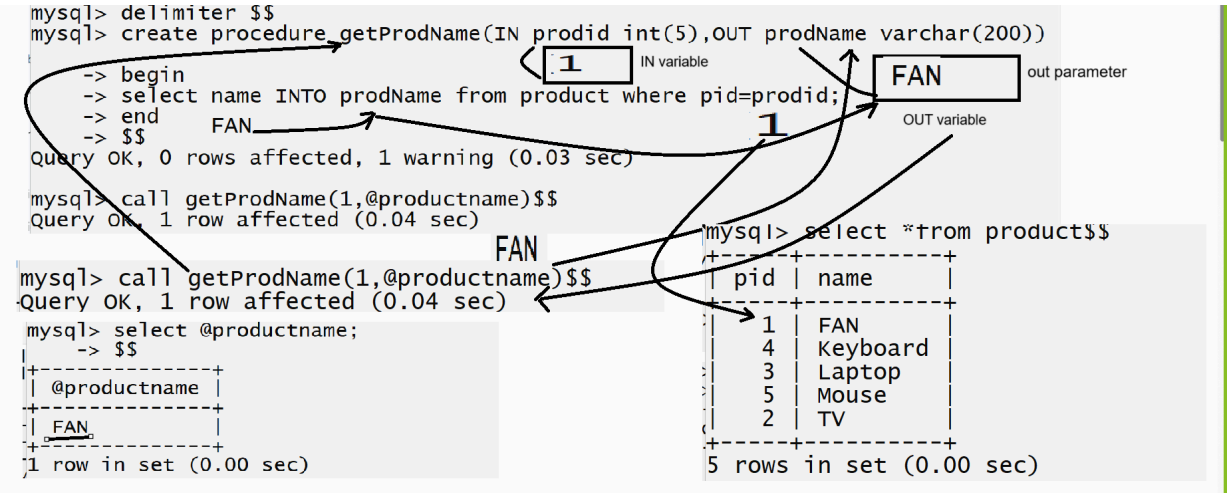
4. When we call the procedure then catch OUT parameter in procedure by passing variables as parameter in it.

```
mysql> call getProdName(1,@productname)$$
Query OK, 1 row affected (0.04 sec)

mysql> _
```

5. Print OUT variable using select statement;

```
mysql> select @productname;
    -> $$
+---------------+
| @productname  |
+---------------+
| FAN           |
+---------------+
1 row in set (0.00 sec)
```



Example: select name of customer using email address.

```
mysql> delimiter $$
mysql> create procedure getNameByEmail(IN e varchar(200),OUT n varchar(200))
    -> begin
    -> select name INTO n from customer where email=e;
    -> end
    -> $$
Query OK, 0 rows affected (0.04 sec)

mysql> call getNameByEmail('ram@gmail.com',@custname)$$
Query OK, 1 row affected (0.01 sec)

mysql> select @custname$$
+-----------+
| @custname |
+-----------+
| ram       |
+-----------+
1 row in set (0.00 sec)
```

```
mysql> SET @custemail='ganesh@gmail.com'$$
Query OK, 0 rows affected (0.04 sec)

mysql> call getNameByEmail(@custemail,@Custname)$$
Query OK, 1 row affected (0.00 sec)

mysql> select @custemail as 'Customer Email',@custname as 'Customer name '$$
+------------------+----------------+
| Customer Email   | Customer name  |
+------------------+----------------+
| ganesh@gmail.com | ganesh         |
+------------------+----------------+
1 row in set (0.00 sec)

mysql> _
```

Trigger in SQL

Trigger is same like as procedure only difference is procedure need to call manually and trigger call automatically by database engine on DML statements like as call on insert/delete etc statement
Trigger is a bunch of SQL statement logic or blocks of SQL statements executed automatically by database engine on DML operations.

Types of trigger

---

1. **Before update :** this types of trigger automatically executed by database engine but before updation on destination table on which apply trigger

2. **Before delete** : this types of trigger automatically executed by database engine but before deletion on destination table on which apply trigger

3. **Before insert:** this types of trigger automatically executed by database engine but before insertion in destination table on which apply trigger

4. **After update :** this types of trigger automatically executed by database engine but after updation on destination table on which apply trigger

5. **After insert:** this types of trigger automatically executed by database engine but after insertion on destination table on which apply trigger

6. **After delete :** this types of trigger automatically executed by database engine but after deletion on destination table on which apply trigger

How to create trigger using MYSQL

```
delimiter character
create trigger trigger_name [before | after ] [ insert  | delete | update ]
on tablename
For each row
Begin
  Write here your logics
End
Delimiter
```

Note: when we use trigger then we have to use two fixed constant
OLD : we can access values from table before operation like as delete/update or insertion

NEW : we can access value after operation from table on which trigger executed

Example: Suppose consider we are working on simple banking application and in banking application we want to maintain mini statement record of customer means we have two tables

1. **Account :** account table is used for maintain account information like as acc no , cust name ,cust email, cust contact , balance of account
2. **Mini_statement :** mini statement table can track the all activity perform on account like deposited amount , withdraw amount etc

Our goal is if someone deposit or withdraw amount in account table means perform transaction on account table then automatically transaction should record in mini statement and we want to hand over this responsibility to database engine via trigger