DBMS

CREATING TABLE AND DISPLAYING DETAILS (DESCRIBE)

DROP - DELETE

ALTER IS USED TO EDIT (ADD,DELETE)

DECIMAL - (A,B) - > ‘A’ - SIZE OF THE NUMBER ,’B’- DIGIT AFTER THE DECIMAL

ADDING NEW COLUMN GPA

```
CREATE TABLE student (
  student_id INT PRIMARY KEY,
  name VARCHAR(20),
  major VARCHAR(20)
  -- PRIMARY KEY(student_id)
);
-- DROP TABLE student;  (delete table - student)
-- DESCRIBE student;
-- DESCRIBE is used to dislay the table details

ALTER TABLE student ADD gpa DECIMAL(3,2);
DESCRIBE student;
```

Output:

```
Field   Type    Null    Key     Default Extra
student_id      int     NO      PRI     NULL
name    varchar(20)     YES             NULL
major   varchar(20)     YES             NULL
gpa     decimal(3,2)    YES             NULL
```

DELETING GPA USING DROP BY COLUMN NAME

```sql
CREATE TABLE student (
  student_id INT PRIMARY KEY,
  name VARCHAR(20),
  major VARCHAR(20)
  -- PRIMARY KEY(student_id)
);
-- DROP TABLE student;  (delete table - student)
-- DESCRIBE student;
-- DESCRIBE is used to dislay the table details

ALTER TABLE student ADD gpa DECIMAL(3,2);
ALTER TABLE student DROP COLUMN gpa;
DESCRIBE student;
```

INSERTING DATA :

Insert is used to insert values into the table

Cmd : INSERT INTO tableName VALUES(1,"Barath","CS");

Select * from student - > display the table content

```sql
CREATE TABLE student (
  student_id INT PRIMARY KEY,
  name VARCHAR(20),
  major VARCHAR(20)
  -- PRIMARY KEY(student_id)
);

INSERT INTO student VALUES(1,"Barath","CS");
SELECT * FROM student;
```

Output:

```
student_id      name    major
1       Barath  CS
```

INSERTING PARTICULAR FIELDS INTO THE TABLE.

```sql
CREATE TABLE student (
  id INT PRIMARY KEY,
  name VARCHAR(20),
  major VARCHAR(20)
  -- PRIMARY KEY(student_id)
);

INSERT INTO student VALUES(1,"Barath","CS");
INSERT INTO student VALUES(2,"Matthew","Maths");
INSERT INTO student(id,name) VALUES(3,"James");
SELECT * FROM student;
```

Output:

```
id      name    major
1       Barath  CS
2       Matthew Maths
3       James   NULL
```

Rename : (Table).

```
rename table tab_employee to tbl_employee;
```

Alter (add column) :

```
alter table tbl_employee add gender char(1);
```

Difference between char and varchar :

Char - static or fixed(faster);
Varchar - dynamic()

Modify :

```
alter table tbl_employee modify gender varchar(10);
```

Alter column in table :

```
alter table tbl_employee rename column gender to egender;
```

Drop table (delete):

```
drop table tbl_employee;
```

```
mysql> desc tbl_employee;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| eid     | int         | YES  |     | NULL    |       |
| ename   | varchar(20) | YES  |     | NULL    |       |
| esalary | int         | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> alter table tbl_employee add gender char(1);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc tbl_employee;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| eid     | int         | YES  |     | NULL    |       |
| ename   | varchar(20) | YES  |     | NULL    |       |
| esalary | int         | YES  |     | NULL    |       |
| gender  | char(1)     | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> alter table tbl_employee modify gender varchar(10);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc tbl_employee;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| eid     | int         | YES  |     | NULL    |       |
| ename   | varchar(20) | YES  |     | NULL    |       |
| esalary | int         | YES  |     | NULL    |       |
| gender  | varchar(10) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

```
mysql> alter table tbl_employee rename column gender to egender;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc tbl_employee;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| eid     | int         | YES  |     | NULL    |       |
| ename   | varchar(20) | YES  |     | NULL    |       |
| esalary | int         | YES  |     | NULL    |       |
| egender | varchar(10) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```

```
mysql> drop table tbl_employee;
Query OK, 0 rows affected (0.01 sec)

mysql> show tables;
Empty set (0.00 sec)
```

DDL : data definition language

- **CREATE**
- **RENAME**
- **ALTER**
    - ADD
    - MODIFY
    - RENAME
    - DROP
- **DROP -** DELETE ENTIRE TABLE record and structure
- **TRUNCATE -** DELETE ONLY TABLE RECORD

DML : Data Manipulation language

- **INSERT**
- **DELETE**
- **UPDATE**

TCL : Transaction control language

- **COMMIT**
- **ROLLBACK**

Link : https://www.javatpoint.com/dbms-er-model-concept

SQL is a standard query language of the database.

Where the PL/ SQL stands for "*Procedural Language extensions SQL*." It is used in the Oracle database and the extension of Structured Query Language (SQL). Whereas, T-SQL stands for "Transact-*SQL*.," which is the extension of Structured Query Language (SQL) used in Microsoft.

RDBMS :

A relational database management system (RDBMS) is a program used to create, update, and manage relational databases.

Some of the most well-known RDBMS include Maria DB, Oracle, MySQL.

What is Table :

The data in a RDBMS is stored in database object known as tables.

This table is basically a collection of related data entities and it consist of numerous columns and rows.
Remember , a table is the most common and simplest form of storing data.

Fields :

Evert table is broken up into small entities called fields.

A filend is a column in a table that is designed to maintain specific information about every record in the table.

For example, our CUSTOMERS table consist of different fields like ID,Name,Age,Salary,City and Country.

What is Record or Row ?

A record is also called as a row of data is each individual entry that exists in a table.

For EX : there are 3 records in the above CUSTOMERS table.
Following is a single row of data or record in the CUSTOMER table.

ID	Name	Age	Salary	City	Country
1	matthew	21	200000	Chennai	India

What is a Column ?

A column is a vertical entity in a table that contains all information associated with a specific field in a table For example, our CUSTOMER table have different column RESPECTED to their ID,Name,Age,Salary,City,Country.

What is NULL Value :

A Null value in a table is a value in a field that appears to be blank which means a field with null value is a field with no value.

Null is not zero(0), empty,It's a field left blank during the entry.

What are Constraints?

Constraints are the rules enforced on data columns on a table .These are used to limit the type of data that can go into a table.This ensures the accuracy and reliability of the data in the database.

Constraints can either be column either be column level or table level.Column level constraints are applied only to one column whereas table level constraints are applied to the entire table.

Types of Constraints:

S.NO	Constrains
1	Not NUll Constraint: Ensure that a column have a null value
2	**Default Constraint :** Provides a default value for a column when none is specified.

3	**Unique Key :** Ensure that all values in the column are different
4	**Primary Key** Unique identify each row/record in a table
5	**Foreign Key** Unique identify each row/record In any other database
6	**Check Constraint** Ensure all values in the a column Satisfy certain condition
7	**Index Constraints**

What is Normalization ?

Database Normalization is the process of efficient organizing data ina database

Two reasons for normalization :

1.Eliminating redundant data

2.Ensure data dependency exist.

First Normal Form :

A sets the basic rules to organize the data in a database. A database is said to be in first normal form if it satisfies the following conditions

Rule1 (Atomic Value) - Every column of a table should contain only atomic values.An atomic value that cannot be divided further.

Rule2 (No Repeating Groups) - There are no repeating groups of data .This means a table should not contain repeating columns.

Second Normal Form :

- Satisfy all rules of 1NF.
- There must be no partial dependencies of any of the column on the primary key.
- In the second normal form ,all non-key attributes are fully functional

‘

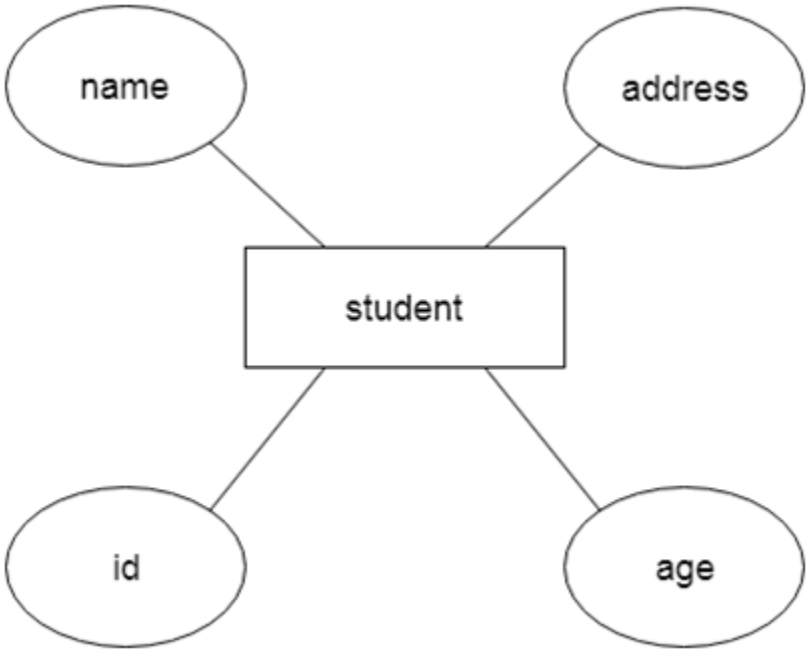
Third normal Form :

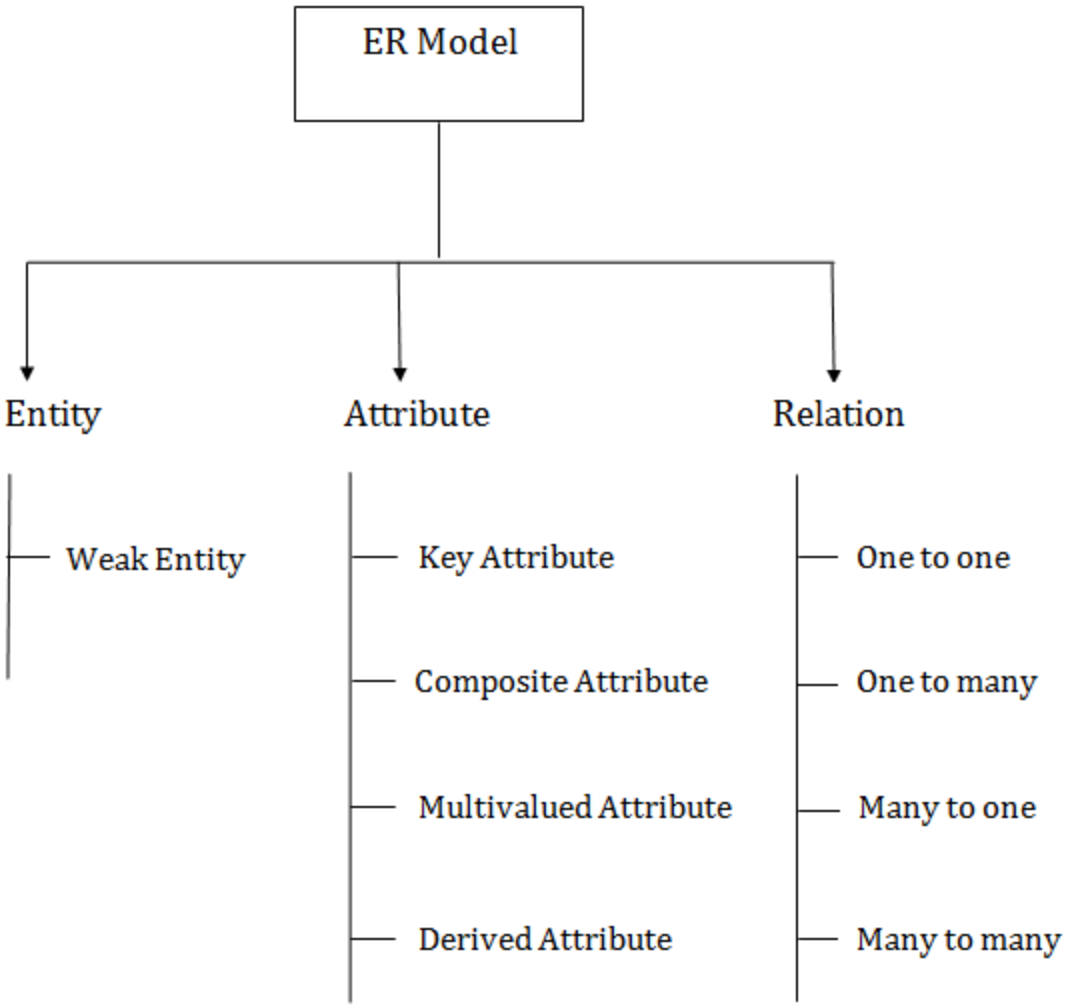
- Satisfy 2NF
- 3NF is used to reduce the data duplication.It is also used to achieve data integrity
- If there is no transitive dependency for non-prime attributes then the relation must be in third normal form.

What is an ER Diagram ?

ER model stands for an Entity-Relationship model.It is a high-level data model. This model is used to define the data elements and relationships

Suppose we design a school database.In this database,the student will be an entity with attributes like city ,street name, pin code,etc and there will be a relationship between them .

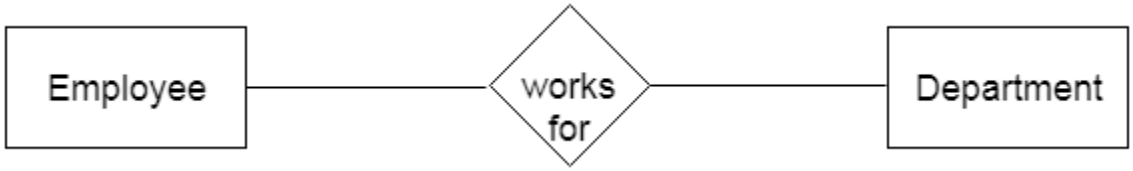




1. Entity:

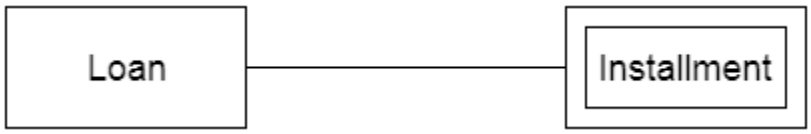
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

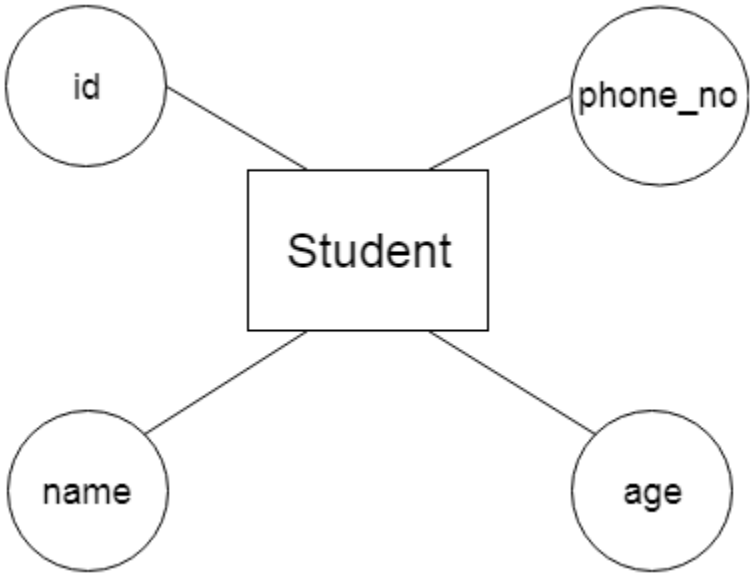
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

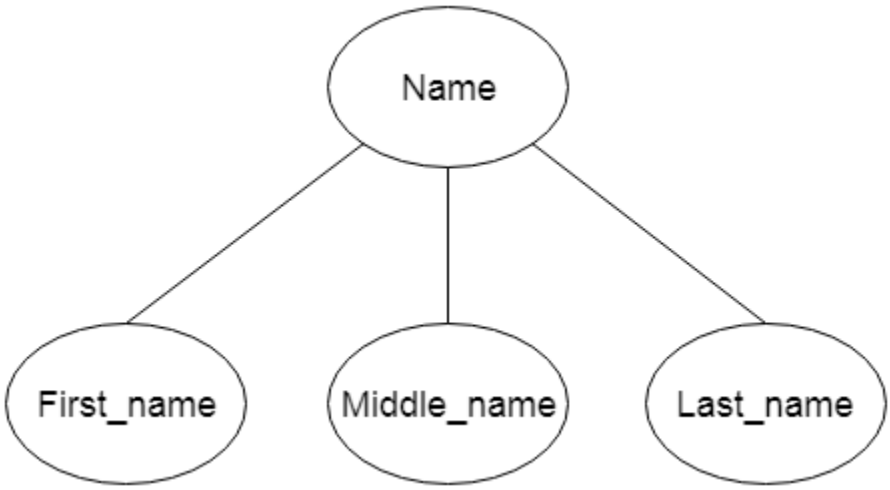
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



Composite Attribute

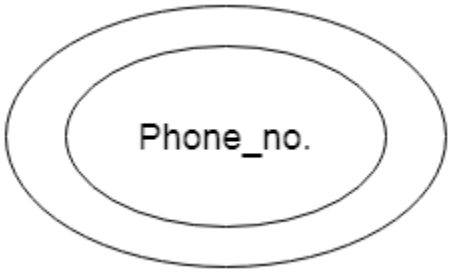
An attribute composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



Multivalued Attribute

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent a multivalued attribute.

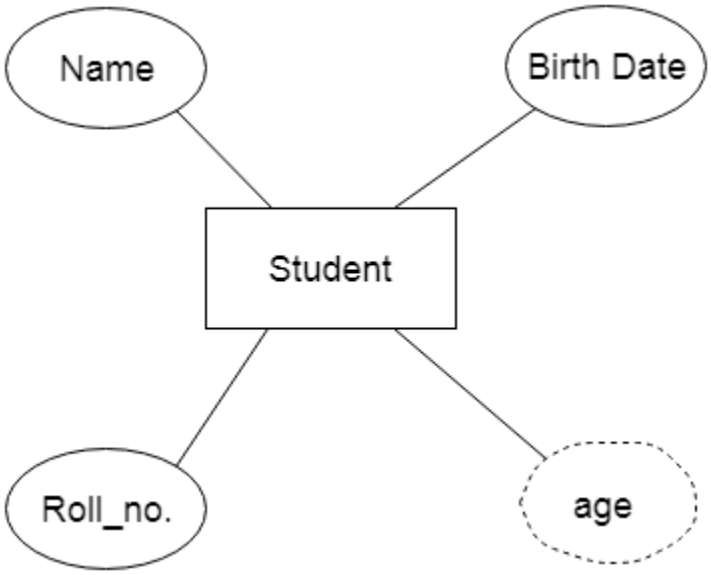
For example, a student can have more than one phone number.



Derived Attribute

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

One-to-many relationship

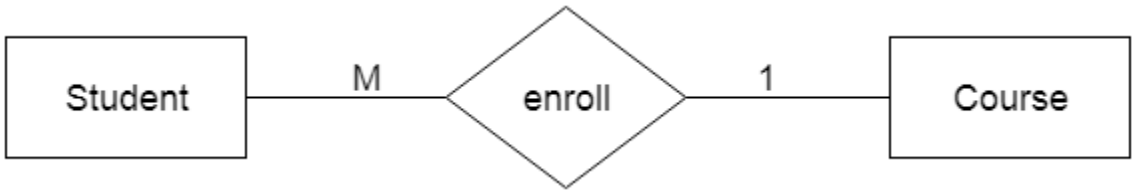
When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.

Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



Many-to-many relationship

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, employees can assign by many projects and project can have many employees.

DML :
Insert query :

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| lab_1              |
| matthew            |
| mydb               |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| world              |
+--------------------+
9 rows in set (0.00 sec)

mysql> use mydb;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> create table tbl_employee(eid int(5), ename varchar(20), esalary int(5));
Query OK, 0 rows affected, 2 warnings (0.02 sec)

mysql> insert into tbl_employee values(101,'Valan',2000);
Query OK, 1 row affected (0.01 sec)

mysql> insert into tbl_employee value(102,'Matthew',3000);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    3000 |
+------+---------+---------+
2 rows in set (0.00 sec)

mysql> insert into tbl_employee values(103,null,4000);
Query OK, 1 row affected (0.02 sec)

mysql> insert into tbl_employee(eid,ename) values(104,"Barath");
Query OK, 1 row affected (0.01 sec)

mysql> select * from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    3000 |
|  103 | NULL    |    4000 |
|  104 | Barath  |    NULL |
+------+---------+---------+
4 rows in set (0.00 sec)
```

SELECT QUERRY :

The SQL SELECT command is used to fetch data from the MySql Table.

Syntax :

Here is generic SQL syntax of SELECT command to fetch data from MYSql table -

QUERRY :

```
SELECT field1,field2,....fieldN
FROM table_name1,table_name2…
[WHERE Clause]
[OFFSET M][LIMIT N];
```

```
mysql> SELECT eid ,esalary from tbl_employee;
+------+---------+
| eid  | esalary |
+------+---------+
|  101 |    2000 |
|  102 |    3000 |
|  103 |    4000 |
|  104 |    NULL |
+------+---------+
4 rows in set (0.00 sec)
```

```
mysql> select * from tbl_employee where esalary > 2000;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  102 | Matthew |    3000 |
|  103 | NULL    |    4000 |
+------+---------+---------+
2 rows in set (0.00 sec)
```

```
mysql> select * from tbl_employee where esalary >= 2000;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    3000 |
|  103 | NULL    |    4000 |
+------+---------+---------+
3 rows in set (0.00 sec)
```

```
mysql> select * from tbl_employee where ename ='Matthew';
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  102 | Matthew |    3000 |
+------+---------+---------+
1 row in set (0.00 sec)
```

```
mysql> select * from tbl_employee where ename !='Matthew';
+------+--------+---------+
| eid  | ename  | esalary |
+------+--------+---------+
|  101 | Valan  |    2000 |
|  104 | Barath |    NULL |
+------+--------+---------+
2 rows in set (0.00 sec)
```

```
mysql> select * from tbl_employee where ename is null;
+------+-------+---------+
| eid  | ename | esalary |
+------+-------+---------+
|  103 | NULL  |    4000 |
+------+-------+---------+
1 row in set (0.00 sec)

mysql> select * from tbl_employee where ename is not null;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    3000 |
|  104 | Barath  |    NULL |
+------+---------+---------+
3 rows in set (0.00 sec)
```

LOGICAL AND :

```
mysql> select * from tbl_employee where ename is not null and esalary = 3000;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  102 | Matthew |    3000 |
+------+---------+---------+
1 row in set (0.00 sec)
```

LOGICAL OR

```
mysql> select * from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    3000 |
|  103 | NULL    |    4000 |
|  104 | Barath  |    NULL |
+------+---------+---------+
4 rows in set (0.00 sec)

mysql> select * from tbl_employee where ename is not null OR esalary = 3000;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    3000 |
|  104 | Barath  |    NULL |
+------+---------+---------+
3 rows in set (0.00 sec)
```

IN OPERATOR :

```
mysql> select * from tbl_employee where eid in(101,103,106);
+------+-------+---------+
| eid  | ename | esalary |
+------+-------+---------+
|  101 | Valan |    2000 |
|  103 | NULL  |    4000 |
+------+-------+---------+
2 rows in set (0.00 sec)
```

NOT IN :

```
mysql> select * from tbl_employee where eid not in(101,103,106);
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  102 | Matthew |    3000 |
|  104 | Barath  |    NULL |
+------+---------+---------+
2 rows in set (0.00 sec)
```

BETWEEN:

```
mysql> select * from tbl_employee where esalary between 2000 and 4000;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    3000 |
|  103 | NULL    |    4000 |
+------+---------+---------+
3 rows in set (0.00 sec)

mysql> select * from tbl_employee where esalary between 2000 and 3000;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    3000 |
+------+---------+---------+
2 rows in set (0.00 sec)

mysql> select * from tbl_employee where esalary not between 2000 and 3000;
+------+-------+---------+
| eid  | ename | esalary |
+------+-------+---------+
|  103 | NULL  |    4000 |
+------+-------+---------+
1 row in set (0.00 sec)

mysql> select * from tbl_employee where esalary not between 2000 and 4000;
Empty set (0.00 sec)
```

LIKE :

And not LIKE

% means 0 to n values

```
mysql> select * from tbl_employee where ename like 'M%';
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  102 | Matthew |    3000 |
+------+---------+---------+
1 row in set (0.00 sec)

mysql> select * from tbl_employee where ename like '_a%';
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    3000 |
|  104 | Barath  |    NULL |
+------+---------+---------+
3 rows in set (0.00 sec)

mysql> select * from tbl_employee where ename like '__t%';
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  102 | Matthew |    3000 |
+------+---------+---------+
1 row in set (0.00 sec)

mysql> select * from tbl_employee where ename not like '__t%';
+------+--------+---------+
| eid  | ename  | esalary |
+------+--------+---------+
|  101 | Valan  |    2000 |
|  104 | Barath |    NULL |
+------+--------+---------+
2 rows in set (0.00 sec)
```

```
mysql> select * from tbl_employee where ename not like 'M%';
+------+--------+---------+
| eid  | ename  | esalary |
+------+--------+---------+
|  101 | Valan  |    2000 |
|  104 | Barath |    NULL |
+------+--------+---------+
2 rows in set (0.00 sec)
```

UPDATE QUERY :

There may be required where the existing data in a MySQL table needs to be modified .You can do so by using the SQL UPDATE command .This will modify any field value of any MySql table.

Syntax :

The following code block has a generic sql syntax of the update command the data of the MySQl table -

UPDATE table_name SET field1 = new-value2 [WHERE Clause]

- You can update one or more fields altogether.
- You can specify any condition using the WHERE clause.
- You can update the values in table at a time

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

mysql> update tbl_employee set esalary =0 ;
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> select * from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |       0 |
|  102 | Matthew |       0 |
|  103 | NULL    |       0 |
|  104 | Barath  |       0 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

Setting auto commit to false;

```
mysql> select @@autocommit from dual;
+--------------+
| @@autocommit |
+--------------+
|            1 |
+--------------+
1 row in set (0.00 sec)

mysql> set autocommit =0;
Query OK, 0 rows affected (0.00 sec)

mysql> select @@autocommit from dual;
+--------------+
| @@autocommit |
+--------------+
|            0 |
+--------------+
1 row in set (0.00 sec)
```

After turning off auto commit :

```
mysql> update tbl_employee set esalary = 1000 where eid = 101;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    1000 |
|  102 | Matthew |       0 |
|  103 | NULL    |       0 |
|  104 | Barath  |       0 |
+------+---------+---------+
4 rows in set (0.00 sec)

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

mysql> select * from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |       0 |
|  102 | Matthew |       0 |
|  103 | NULL    |       0 |
|  104 | Barath  |       0 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

Rollback will work till the commit .

```
mysql> select *from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |       0 |
|  102 | Matthew |       0 |
|  103 | NULL    |       0 |
|  104 | Barath  |       0 |
+------+---------+---------+
4 rows in set (0.00 sec)

mysql> update tbl_employee set esalary = 2000;
Query OK, 4 rows affected (0.00 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> select *from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    2000 |
|  103 | NULL    |    2000 |
|  104 | Barath  |    2000 |
+------+---------+---------+
4 rows in set (0.00 sec)

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

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

mysql> select*from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | Valan   |    2000 |
|  102 | Matthew |    2000 |
|  103 | NULL    |    2000 |
|  104 | Barath  |    2000 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

```
mysql> update tbl_employee set ename = null, esalary = 0 where eid in(101,103,106);
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select*from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | NULL    |       0 |
|  102 | Matthew |    2000 |
|  103 | NULL    |       0 |
|  104 | Barath  |    2000 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

```
mysql> select*from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | NULL    |       0 |
|  102 | Matthew |    2000 |
|  103 | NULL    |       0 |
|  104 | Barath  |    2000 |
+------+---------+---------+
4 rows in set (0.00 sec)

mysql> update tbl_employee set ename = null, esalary =1000 where eid in(101,103,106);
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> update tbl_employee set ename = null, esalary =3000 where eid in(101,103,106);
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

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

mysql> select*from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | NULL    |       0 |
|  102 | Matthew |    2000 |
|  103 | NULL    |       0 |
|  104 | Barath  |    2000 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

DELETE :

Self notes:
The difference between delete and truncate is rollback will work in delete because its DML and it wont work for truncate because its DDL.

Another difference is in delete we can use condition to delete particular field or if we use only delete without condition it will delete all the records in the table like truncate,, but in truncate it will delete entire table records we can use condition to truncate a particular record.

DELETE FROM table_name [WHERE Clause]

- If the WHERE clause is not specified , then all the records will be deleted from the table .
- You can specify any condition using WHERE clause.
- You can delete records in a single table at a time

The WHERE clause is very useful when you want to delete selected rows in a table.

```
mysql> delete from tbl_employee where ename='Barath';
Query OK, 1 row affected (0.00 sec)

mysql> select*from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | NULL    |       0 |
|  102 | Matthew |    2000 |
|  103 | NULL    |       0 |
+------+---------+---------+
3 rows in set (0.00 sec)
```

Like Clause :

We have seen the SQL SELECT command to fetch the data from the MySql table .We can also use the conditional clause called as the WHERE clause to select the required records.

A WHERE clause with the ‘equal to’ sign = works fine where we want to do an exact match. Like if “employee_name = ‘Sanjay’”.But there may be a requirement where we want to filter out all the results where employee_name should contain “jay”.This can be handled using SQL LIKE Clause along with the WHERE clause.

Syntax :

SELECT field1,field2,....fieldN table_name2… WHERE field 1 like condition1 [AND[OR]]
field2 = 'somevalue'

- You can specify any condition using the where clause.
- You can use the LIKE Clause along with the where clause.
- You can use the LIKE clause in the place of the equal sign.
- When the LIKE clause is placed on the equals to sign.
- You can specify more than one condition using AND or OR operators.
- A WHERE ... .LIKE clauses can be used along with DELETE or UPDATE SQL commands also to specify a condition.

Sorting Results :

We have seen the SQL SELECT command to fetch data from a MySQL table.When you select rows,the MySQL server is free to return them in any order ,unless you instruct it otherwise by saying how to sort the result.

But you sort a result set by adding an ORDER BY clause the column or columns which you want to sort.

Syntax :

SELECT field1,field2,...fieldN table_name1,table_name2…
ORDER BY field1,[field2…][ASC[DESC]]

- You can sort the returned result in any field , if the field is being listed out.
- You can sort the result on more than one field.
- You can use the keyword ASC or DSC to get the result in ascending or descending order.By default it's the ascending order.
- You use the WHERE….LIKE clause in the usual way to put the condition.

```
mysql> select*from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | James   |    4000 |
|  102 | Matthew |    2000 |
|  103 | NULL    |       0 |
|  104 | Barath  |    2000 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

```
mysql> select*from tbl_employee order by ename;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  103 | NULL    |       0 |
|  104 | Barath  |    2000 |
|  101 | James   |    4000 |
|  102 | Matthew |    2000 |
+------+---------+---------+
4 rows in set (0.00 sec)

mysql> select*from tbl_employee order by eid;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | James   |    4000 |
|  102 | Matthew |    2000 |
|  103 | NULL    |       0 |
|  104 | Barath  |    2000 |
+------+---------+---------+
4 rows in set (0.00 sec)

mysql> select*from tbl_employee order by eid DESC;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  104 | Barath  |    2000 |
|  103 | NULL    |       0 |
|  102 | Matthew |    2000 |
|  101 | James   |    4000 |
+------+---------+---------+
4 rows in set (0.00 sec)

mysql> select*from tbl_employee order by ename DESC;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  102 | Matthew |    2000 |
|  101 | James   |    4000 |
|  104 | Barath  |    2000 |
|  103 | NULL    |       0 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

MySqlNull Values :

We have seen the SQL SELECT command along with the WHERE clause to fetch data from a MYSql table ,but when we try to give a condition ,which compares the field or the column value to NULL ,it does not work properly.
To handle such a situation ,MySql provides three options.

- **IS NULL** - This operator returns true,if the column value is NULL.
- **IS NOT NULL** - This operator returns true if column value is not NULL.
- **<=>** - This operator compares values,which (unlike the = operator) is true even for two NULL values.

AS :
As keyword is optional.It is used to display .It will not affect the original table.

```
mysql> select eid,ename from tbl_employee;
+------+---------+
| eid  | ename   |
+------+---------+
|  101 | NULL    |
|  102 | Matthew |
|  103 | NULL    |
|  104 | Barath  |
+------+---------+
4 rows in set (0.00 sec)

mysql> select eid as "Empoloyee Id",ename "Employee Name" from tbl_employee;
+--------------+---------------+
| Empoloyee Id | Employee Name |
+--------------+---------------+
|          101 | NULL          |
|          102 | Matthew       |
|          103 | NULL          |
|          104 | Barath        |
+--------------+---------------+
4 rows in set (0.00 sec)

mysql> select *from tbl_employee;
+------+---------+---------+
| eid  | ename   | esalary |
+------+---------+---------+
|  101 | NULL    |       0 |
|  102 | Matthew |    2000 |
|  103 | NULL    |       0 |
|  104 | Barath  |    2000 |
+------+---------+---------+
4 rows in set (0.00 sec)
```

Subqueries :

You can write a query with in a query in MySQL this is known as a query or an inner query or a nested query .Usually , a subquery is embedded within the where clause.

Subquery is used to return data that will be used in the main query as a condition to further restrict the data into

```
mysql> select now() ;
+---------------------+
| now()               |
+---------------------+
| 2023-10-03 09:51:05 |
+---------------------+
1 row in set (0.00 sec)
```

Sub_queries :

```
mysql> alter table tbl_employee add edno int(5);
Query OK, 0 rows affected, 1 warning (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> select * from tbl_employee;
+------+---------+---------+------+
| eid  | ename   | esalary | edno |
+------+---------+---------+------+
|  101 | NULL    |       0 | NULL |
|  102 | Matthew |    2000 | NULL |
|  103 | NULL    |       0 | NULL |
|  104 | Barath  |    2000 | NULL |
+------+---------+---------+------+
4 rows in set (0.00 sec)
```

Single row/value subquery :

```
mysql> update tbl_employee set edno = 10 where eid in (101,102);
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> update tbl_employee set edno = 10 where eid in (103,104);
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from tbl_employee;
+------+---------+---------+------+
| eid  | ename   | esalary | edno |
+------+---------+---------+------+
|  101 | NULL    |       0 |   10 |
|  102 | Matthew |    2000 |   10 |
|  103 | NULL    |       0 |   10 |
|  104 | Barath  |    2000 |   10 |
+------+---------+---------+------+
4 rows in set (0.00 sec)
```

```
mysql> update tbl_employee set ename = "james",esalary = 3000 where eid = 101;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from tbl_employee;
+------+---------+---------+------+
| eid  | ename   | esalary | edno |
+------+---------+---------+------+
|  101 | james   |    3000 |   10 |
|  102 | Matthew |    2000 |   10 |
|  103 | NULL    |       0 |   10 |
|  104 | Barath  |    2000 |   10 |
+------+---------+---------+------+
4 rows in set (0.00 sec)
```

```
mysql> create table tbl_dept (dno int(5),dname varchar(10));
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> insert into tbl_dept values(10,"L&D"),(20,"Project");
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from tbl_dept;
+------+---------+
| dno  | dname   |
+------+---------+
|   10 | L&D     |
|   20 | Project |
+------+---------+
2 rows in set (0.00 sec)
```

```
mysql> select * from tbl_employee where edno = (select dno from tbl_dept where dname ="L&D");
+------+---------+---------+------+
| eid  | ename   | esalary | edno |
+------+---------+---------+------+
|  101 | james   |    3000 |   10 |
|  102 | Matthew |    2000 |   10 |
|  103 | NULL    |       0 |   10 |
|  104 | Barath  |    2000 |   10 |
+------+---------+---------+------+
4 rows in set (0.00 sec)

mysql> update tbl_employee set edno = 20 where eid in (103,104);
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from tbl_employee;
+------+---------+---------+------+
| eid  | ename   | esalary | edno |
+------+---------+---------+------+
|  101 | james   |    3000 |   10 |
|  102 | Matthew |    2000 |   10 |
|  103 | NULL    |       0 |   20 |
|  104 | Barath  |    2000 |   20 |
+------+---------+---------+------+
4 rows in set (0.00 sec)
```

```
mysql> select * from tbl_employee where edno = (select dno from tbl_dept where dname ="L&D");
+------+---------+---------+------+
| eid  | ename   | esalary | edno |
+------+---------+---------+------+
|  101 | james   |    3000 |   10 |
|  102 | Matthew |    2000 |   10 |
+------+---------+---------+------+
2 rows in set (0.00 sec)
```

```
mysql> select * from tbl_employee where edno = (select dno from tbl_dept where dname ="Project");
+------+--------+---------+------+
| eid  | ename  | esalary | edno |
+------+--------+---------+------+
|  103 | NULL   |       0 |   20 |
|  104 | Barath |    2000 |   20 |
+------+--------+---------+------+
2 rows in set (0.00 sec)
```

```
mysql> select dname from tbl_dept where dno = (select edno from tbl_employee where ename = "Barath");
+---------+
| dname   |
+---------+
| Project |
+---------+
1 row in set (0.00 sec)

mysql> select dname from tbl_dept where dno = (select edno from tbl_employee where ename is null);
+---------+
| dname   |
+---------+
| Project |
+---------+
1 row in set (0.00 sec)
```

```
mysql> update tbl_employee set esalary=esalary+200 where edno=(select dno from tbl_dept where dname = "Project")
    -> ;
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select* from tbl_employee;
+------+---------+---------+------+
| eid  | ename   | esalary | edno |
+------+---------+---------+------+
|  101 | james   |    3000 |   10 |
|  102 | Matthew |    2000 |   10 |
|  103 | NULL    |     200 |   20 |
|  104 | Barath  |    2200 |   20 |
+------+---------+---------+------+
4 rows in set (0.00 sec)
```

Multi row/value subquery :

Use “**IN**” to achieve multi row subquery.

```
mysql> select dname from tbl_dept where dno in(select edno from tbl_employee where eid in(101,103));
+---------+
| dname   |
+---------+
| L&D     |
| Project |
+---------+
2 rows in set (0.00 sec)
```

```
mysql> select* from tbl_employee;
+------+---------+---------+------+
| eid  | ename   | esalary | edno |
+------+---------+---------+------+
|  101 | james   |    3000 |   10 |
|  102 | Matthew |    2000 |   10 |
|  103 | NULL    |     200 |   20 |
|  104 | Barath  |    2200 |   20 |
+------+---------+---------+------+
4 rows in set (0.00 sec)

mysql> select* from tbl_dept;
+------+---------+
| dno  | dname   |
+------+---------+
|   10 | L&D     |
|   20 | Project |
+------+---------+
2 rows in set (0.00 sec)
```

```
mysql> select dname from tbl_dept where dno in (select edno from tbl_employee where esalary > 2000 );
+---------+
| dname   |
+---------+
| L&D     |
| Project |
+---------+
2 rows in set (0.00 sec)

mysql> select dname from tbl_dept where dno in (select edno from tbl_employee where esalary = 2000 );
+-------+
| dname |
+-------+
| L&D   |
+-------+
1 row in set (0.00 sec)
```

SQL - CONSTRAINTS :

The constraint in my-sql is used to specify the rules that allow or restrict what values/data will be stored in the table.

They provide a suitable method to ensure data accuracy and integrity inside the table.

It also helps to limit the type of data that will be inserted inside the table .If any interruption occurs between the constraint and data action, the action is failed.

Constraints Used in MySQL:

The following are the most common constraints used in MY SQL :

- NOT NULL
- CHECK
- DEFAULT
- PRIMARY KEY
- AUTO_INCREMENT
- UNIQUE

```
mysql> create table tbl_student(rno int(5) primary key auto_increment,sname varchar(20) not null,smark int(3) check (smark>0),smno int(10) unique, sage int(3) default 15);
Query OK, 0 rows affected, 4 warnings (0.02 sec)

mysql> desc tbl_student;
+-------+-------------+------+-----+---------+----------------+
| Field | Type        | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| rno   | int         | NO   | PRI | NULL    | auto_increment |
| sname | varchar(20) | NO   |     | NULL    |                |
| smark | int         | YES  |     | NULL    |                |
| smno  | int         | YES  | UNI | NULL    |                |
| sage  | int         | YES  |     | 15      |                |
+-------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)
```

```
mysql> insert into tbl_student(sname,smark,smno) values("Barath",760,6357623);
Query OK, 1 row affected (0.00 sec)

mysql> select* from tbl_student;
+-----+---------+-------+------------+------+
| rno | sname   | smark | smno       | sage |
+-----+---------+-------+------------+------+
|   1 | Matthew |    60 | 1234567891 |   15 |
|   2 | Barath  |   760 |    6357623 |   15 |
+-----+---------+-------+------------+------+
2 rows in set (0.00 sec)
```

```
mysql> insert into tbl_student(rno,sname,smark,smno) values(20,"Barath",760,6357323);
Query OK, 1 row affected (0.00 sec)

mysql> select* from tbl_student;
+-----+---------+-------+------------+------+
| rno | sname   | smark | smno       | sage |
+-----+---------+-------+------------+------+
|   1 | Matthew |    60 | 1234567891 |   15 |
|   2 | Barath  |   760 |    6357623 |   15 |
|  20 | Barath  |   760 |    6357323 |   15 |
+-----+---------+-------+------------+------+
3 rows in set (0.00 sec)
```

```
mysql> insert into tbl_student(sname,smark,smno) values("James",80,46578912);
Query OK, 1 row affected (0.00 sec)

mysql> select* from tbl_student;
+-----+---------+-------+------------+------+
| rno | sname   | smark | smno       | sage |
+-----+---------+-------+------------+------+
|   1 | Matthew |    60 | 1234567891 |   15 |
|   2 | Barath  |   760 |    6357623 |   15 |
|  20 | Barath  |   760 |    6357323 |   15 |
|  21 | James   |    80 |   46578912 |   15 |
+-----+---------+-------+------------+------+
4 rows in set (0.00 sec)
```

Foreign key :

```
mysql> create table tbl_dept(dno int primary key,dname varchar (20));
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> create table tbl_employee(id int primary key,name varchar (20),salary int,dno int,foreign key (dno) references tbl_dept(dno));
Query OK, 0 rows affected (0.02 sec)

mysql> desc tbl_employee;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int         | NO   | PRI | NULL    |       |
| name   | varchar(20) | YES  |     | NULL    |       |
| salary | int         | YES  |     | NULL    |       |
| dno    | int         | YES  | MUL | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> desc tbl_dept;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| dno   | int         | NO   | PRI | NULL    |       |
| dname | varchar(20) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

```
mysql> insert into tbl_dept values(10,"LD");
Query OK, 1 row affected (0.00 sec)

mysql> insert into tbl_employee values(101,"Barath",3000,10);
Query OK, 1 row affected (0.01 sec)

mysql> select * from tbl_employee;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   3000 |   10 |
+-----+--------+--------+------+
1 row in set (0.00 sec)
```

Duplicating table :

```
mysql> show tables;
+----------------+
| Tables_in_mydb |
+----------------+
| tbl_dept       |
| tbl_employee   |
+----------------+
2 rows in set (0.00 sec)

mysql> select * from tbl_employee;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   3000 |   10 |
+-----+--------+--------+------+
1 row in set (0.00 sec)
```

```
mysql> create table tbl_employee1 as select *from tbl_employee;
Query OK, 1 row affected (0.02 sec)
Records: 1  Duplicates: 0  Warnings: 0

mysql> select * from tbl_employee1;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   3000 |   10 |
+-----+--------+--------+------+
1 row in set (0.00 sec)
```

Creating table structure without data. By giving the wrong condition.

```
mysql> create table tbl_employee2 as select * from tbl_employee where 1=2;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc tbl_employee2;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int         | NO   |     | NULL    |       |
| name   | varchar(20) | YES  |     | NULL    |       |
| salary | int         | YES  |     | NULL    |       |
| dno    | int         | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> select * from tbl_employee2;
Empty set (0.00 sec)
```

GROUP BY:

```
mysql> select * from tbl_employee1;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   3000 |   10 |
| 102 | Naveen |   4000 |   10 |
| 103 | Geo    |   5000 |   20 |
| 104 | Suriya |   6000 |   20 |
| 105 | Ganesh |   7000 |   20 |
+-----+--------+--------+------+
5 rows in set (0.00 sec)

mysql> select count(*) from tbl_employee1;
+----------+
| count(*) |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> select count(*) from tbl_employee1 group by dno;
+----------+
| count(*) |
+----------+
|        2 |
|        3 |
+----------+
2 rows in set (0.00 sec)

mysql> select dno, count(*) from tbl_employee1 group by dno;
+------+----------+
| dno  | count(*) |
+------+----------+
|   10 |        2 |
|   20 |        3 |
+------+----------+
2 rows in set (0.00 sec)

mysql> select dno as "Department No", count(*) as "No of Employees" from tbl_employee1 group by dno;
+---------------+-----------------+
| Department No | No of Employees |
+---------------+-----------------+
|            10 |               2 |
|            20 |               3 |
+---------------+-----------------+
2 rows in set (0.00 sec)
```

```
mysql> select dno as "Department No", count(*) as "No of Employees" from tbl_employee1 group by dno order by dno desc;
+---------------+-----------------+
| Department No | No of Employees |
+---------------+-----------------+
|            20 |               3 |
|            10 |               2 |
+---------------+-----------------+
2 rows in set (0.00 sec)
```

GROUP FUNCTION :

- Count()
- Sum()
- Avg()
- min()
- max()

TABLE :

```
mysql> select * from tbl_employee1;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   3000 |   10 |
| 102 | Naveen |   4000 |   10 |
| 103 | Geo    |   5000 |   20 |
| 104 | Suriya |   6000 |   20 |
| 105 | Ganesh |   7000 |   20 |
+-----+--------+--------+------+
5 rows in set (0.00 sec)
```

SUM :

```
mysql> select  sum(dno)  from tbl_employee1 group by dno;
+----------+
| sum(dno) |
+----------+
|       20 |
|       60 |
+----------+
2 rows in set (0.00 sec)
```

```
mysql> select dno as "Department No", sum(dno),count(*) as "sum of Employees" from tbl_employee1 group by dno;
+---------------+----------+------------------+
| Department No | sum(dno) | sum of Employees |
+---------------+----------+------------------+
|            10 |       20 |                2 |
|            20 |       60 |                3 |
+---------------+----------+------------------+
2 rows in set (0.00 sec)
```

AVG :

```
mysql> select dno as "Department No", avg(dno),count(*) as "sum of Employees" from tbl_employee1 group by dno;
+---------------+----------+------------------+
| Department No | avg(dno) | sum of Employees |
+---------------+----------+------------------+
|            10 |  10.0000 |                2 |
|            20 |  20.0000 |                3 |
+---------------+----------+------------------+
2 rows in set (0.02 sec)
```

```
mysql> select dno, avg(salary) from tbl_employee1 group by dno;
+------+--------------+
| dno  | avg(salary) |
+------+--------------+
|   10 |   3500.0000 |
|   20 |   6000.0000 |
+------+--------------+
2 rows in set (0.00 sec)
```

MIN :

```
mysql> select dno as "Department No", min(dno),count(*) as "sum of Employees" from tbl_employee1 group by dno;
+---------------+----------+------------------+
| Department No | min(dno) | sum of Employees |
+---------------+----------+------------------+
|            10 |       10 |                2 |
|            20 |       20 |                3 |
+---------------+----------+------------------+
2 rows in set (0.00 sec)
```

```
mysql> select min(dno) from tbl_employee1 group by dno;
+-----------+
| min(dno) |
+-----------+
|        10 |
|        20 |
+-----------+
2 rows in set (0.00 sec)
```

```
mysql> select min(salary) from tbl_employee1 group by dno;
+-------------+
| min(salary) |
+-------------+
|        3000 |
|        5000 |
+-------------+
2 rows in set (0.00 sec)

mysql> select min(salary) from tbl_employee1 group by salary;
+-------------+
| min(salary) |
+-------------+
|        3000 |
|        4000 |
|        5000 |
|        6000 |
|        7000 |
+-------------+
5 rows in set (0.00 sec)

mysql> select min(salary) from tbl_employee1 group by name;
+-------------+
| min(salary) |
+-------------+
|        3000 |
|        4000 |
|        5000 |
|        6000 |
|        7000 |
+-------------+
5 rows in set (0.00 sec)

mysql> select min(salary) from tbl_employee1 group by dno;
+-------------+
| min(salary) |
+-------------+
|        3000 |
|        5000 |
+-------------+
2 rows in set (0.00 sec)
```

```
mysql> select dno, min(salary) from tbl_employee1 group by dno;
+------+-------------+
| dno  | min(salary) |
+------+-------------+
|   10 |        3000 |
|   20 |        5000 |
+------+-------------+
2 rows in set (0.00 sec)
```

MAX :

```
mysql> select dno, max(salary) from tbl_employee1 group by dno;
+------+-------------+
| dno  | max(salary) |
+------+-------------+
|   10 |        4000 |
|   20 |        7000 |
+------+-------------+
2 rows in set (0.00 sec)
```

Practice : (Explore)

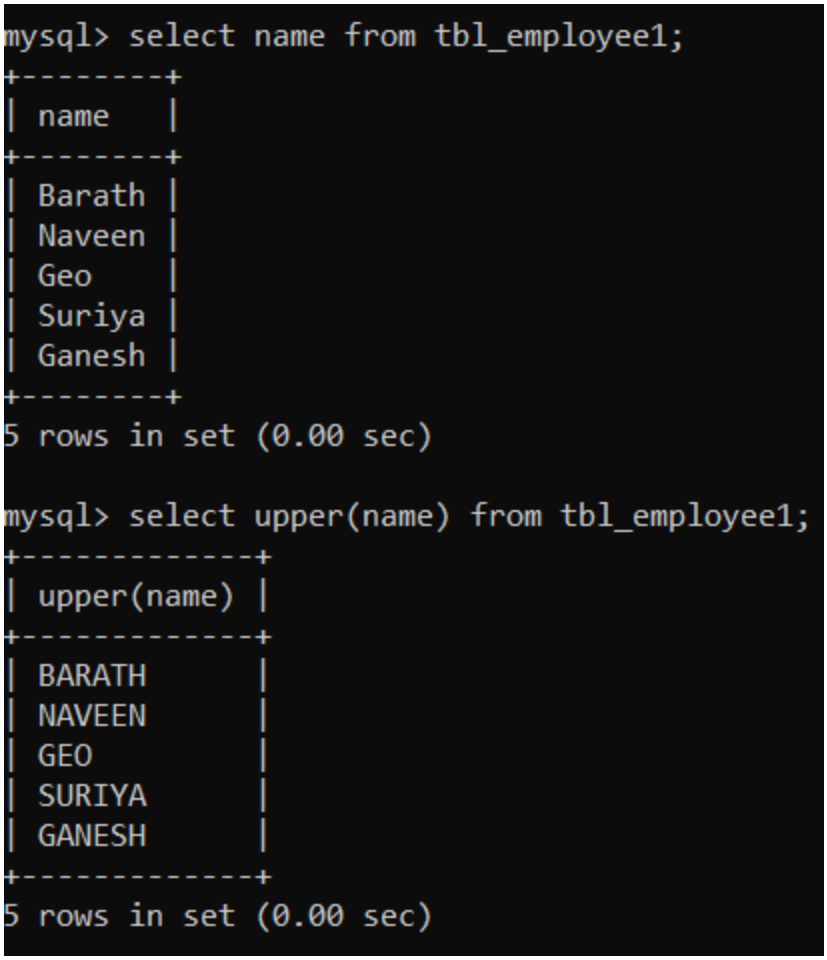
Order by col1,col2;
Order by col1,col2 desc;

Group by col1,col2;

Group by : (having)

```
mysql> select * from tbl_employee1;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   3000 |   10 |
| 102 | Naveen |   4000 |   10 |
| 103 | Geo    |   5000 |   20 |
| 104 | Suriya |   6000 |   20 |
| 105 | Ganesh |   7000 |   20 |
+-----+--------+--------+------+
5 rows in set (0.00 sec)

mysql> select dno,count(*) from tbl_employee1 group by dno;
+------+----------+
| dno  | count(*) |
+------+----------+
|   10 |        2 |
|   20 |        3 |
+------+----------+
2 rows in set (0.00 sec)

mysql> select dno,count(*) from tbl_employee1 group by dno having min(salary)=3000;
+------+----------+
| dno  | count(*) |
+------+----------+
|   10 |        2 |
+------+----------+
1 row in set (0.00 sec)
```

```
mysql> select dno,count(*) from tbl_employee1 group by dno having min(salary)=5000;
+------+----------+
| dno  | count(*) |
+------+----------+
|   20 |        3 |
+------+----------+
1 row in set (0.00 sec)

mysql> select dno,count(*) from tbl_employee1 group by dno having min(salary)=1000;
Empty set (0.00 sec)

mysql> select dno,count(*) from tbl_employee1 group by dno having max(salary)=4000;
+------+----------+
| dno  | count(*) |
+------+----------+
|   10 |        2 |
+------+----------+
1 row in set (0.00 sec)

mysql> select dno,count(*) from tbl_employee1 group by dno having max(salary)=1000;
Empty set (0.00 sec)
```



VIEW :

A view is a database object that has no value .Its contents are based on the base table.

It contains rows and columns similar to the real world .In MySQl,the View is a virtual table created by a query by joining one or more tables.It is opened similarly to the base table but does not contain any data of its own.

The view and table have one main difference that the view are definitions built on top of other tables (or views).If any changes occur in the underlying table,the same changes reflected in the view also.

```
mysql> create view myview as select * from tbl_employee1 where dno=10;
Query OK, 0 rows affected (0.01 sec)

mysql> select * from myview;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   3000 |   10 |
| 102 | Naveen |   4000 |   10 |
+-----+--------+--------+------+
2 rows in set (0.00 sec)

mysql> update myview set salary = 7000 where id = 101;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from myview;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   7000 |   10 |
| 102 | Naveen |   4000 |   10 |
+-----+--------+--------+------+
2 rows in set (0.00 sec)

mysql> select * from tbl_employee1;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   7000 |   10 |
| 102 | Naveen |   4000 |   10 |
| 103 | Geo    |   5000 |   20 |
| 104 | Suriya |   6000 |   20 |
| 105 | Ganesh |   7000 |   20 |
+-----+--------+--------+------+
5 rows in set (0.00 sec)
```

```
mysql> update tbl_employee1 set salary = null where dno =10;
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select * from tbl_employee1;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   NULL |   10 |
| 102 | Naveen |   NULL |   10 |
| 103 | Geo    |   5000 |   20 |
| 104 | Suriya |   6000 |   20 |
| 105 | Ganesh |   7000 |   20 |
+-----+--------+--------+------+
5 rows in set (0.00 sec)

mysql> select * from myview;
+-----+--------+--------+------+
| id  | name   | salary | dno  |
+-----+--------+--------+------+
| 101 | Barath |   NULL |   10 |
| 102 | Naveen |   NULL |   10 |
+-----+--------+--------+------+
2 rows in set (0.00 sec)
```

JOINS :

Types of joins :

- Inner Join,(default)
- Outer Join
- Cross Join,
- Self Join.

Two Tables :

```
mysql> select * from customers;
+---------------+---------------+---------------+
| Customer_Code | Customer_Name | Customer_Area |
+---------------+---------------+---------------+
| C101          | customer1     | chennai       |
| C102          | customer2     | chennai       |
| C103          | customer3     | chennai       |
| C104          | customer4     | bangalore     |
| C105          | customer5     | bangalore     |
+---------------+---------------+---------------+
5 rows in set (0.00 sec)

mysql> select * from agents;
+------------+------------+--------------+
| Agent_Code | Agent_Name | Working_Area |
+------------+------------+--------------+
| A101       | agent1     | chennai      |
| A102       | agent2     | chennai      |
| A103       | agent3     | bangalore    |
| A104       | agent4     | bangalore    |
+------------+------------+--------------+
4 rows in set (0.00 sec)
```

Type -1 :(Using table name)

```
mysql> select agents.Agent_code,agents.Agent_Name,customers.Customer_Name from agents,customers where agents.Working_Area = customers.Customer_Area;
+------------+------------+---------------+
| Agent_code | Agent_Name | Customer_Name |
+------------+------------+---------------+
| A102       | agent2     | customer1     |
| A101       | agent1     | customer1     |
| A102       | agent2     | customer2     |
| A101       | agent1     | customer2     |
| A102       | agent2     | customer3     |
| A101       | agent1     | customer3     |
| A104       | agent4     | customer4     |
| A103       | agent3     | customer4     |
| A104       | agent4     | customer5     |
| A103       | agent3     | customer5     |
+------------+------------+---------------+
10 rows in set (0.00 sec)
```

Type-1(Using order by):

```
mysql> select agents.Agent_code,agents.Agent_Name,customers.Customer_Name from agents,customers where agents.Working_Area = customers.Customer_Area order by Customer_Name,Agent_code;
+------------+------------+---------------+
| Agent_code | Agent_Name | Customer_Name |
+------------+------------+---------------+
| A101       | agent1     | customer1     |
| A102       | agent2     | customer1     |
| A101       | agent1     | customer2     |
| A102       | agent2     | customer2     |
| A101       | agent1     | customer3     |
| A102       | agent2     | customer3     |
| A103       | agent3     | customer4     |
| A104       | agent4     | customer4     |
| A103       | agent3     | customer5     |
| A104       | agent4     | customer5     |
+------------+------------+---------------+
10 rows in set (0.00 sec)
```

Type -2 (Using alias):

```
mysql> select a.Agent_code,a.Agent_Name,c.Customer_Name from agents a,customers c where a.Working_Area = c.Customer_Area order by Customer_Name,Agent_code;
+------------+------------+---------------+
| Agent_code | Agent_Name | Customer_Name |
+------------+------------+---------------+
| A101       | agent1     | customer1     |
| A102       | agent2     | customer1     |
| A101       | agent1     | customer2     |
| A102       | agent2     | customer2     |
| A101       | agent1     | customer3     |
| A102       | agent2     | customer3     |
| A103       | agent3     | customer4     |
| A104       | agent4     | customer4     |
| A103       | agent3     | customer5     |
| A104       | agent4     | customer5     |
+------------+------------+---------------+
10 rows in set (0.00 sec)
```

Type - 3 (Using alias and Join):

```
mysql> select a.Agent_code,a.Agent_Name,c.Customer_Name from agents a join customers c on a.Working_Area = c.Customer_Area order by Customer_Name,Agent_code;
+------------+------------+---------------+
| Agent_code | Agent_Name | Customer_Name |
+------------+------------+---------------+
| A101       | agent1     | customer1     |
| A102       | agent2     | customer1     |
| A101       | agent1     | customer2     |
| A102       | agent2     | customer2     |
| A101       | agent1     | customer3     |
| A102       | agent2     | customer3     |
| A103       | agent3     | customer4     |
| A104       | agent4     | customer4     |
| A103       | agent3     | customer5     |
| A104       | agent4     | customer5     |
+------------+------------+---------------+
10 rows in set (0.00 sec)
```

INNER JOIN :
Equi Join

```
mysql> select a.Agent_code,a.Agent_Name,c.Customer_Name from agents a inner join customers c on a.Working_Area = c.Customer_Area order by Customer_Name,Agent_code;
+------------+------------+---------------+
| Agent_code | Agent_Name | Customer_Name |
+------------+------------+---------------+
| A101       | agent1     | customer1     |
| A102       | agent2     | customer1     |
| A101       | agent1     | customer2     |
| A102       | agent2     | customer2     |
| A101       | agent1     | customer3     |
| A102       | agent2     | customer3     |
| A103       | agent3     | customer4     |
| A104       | agent4     | customer4     |
| A103       | agent3     | customer5     |
| A104       | agent4     | customer5     |
+------------+------------+---------------+
10 rows in set (0.00 sec)
```

Two types of inner join:

- Equi Join (using = in the command),
- Non-Equi Join (using other than =,like(<>,<,>)).

Non-Equi Join :

```
mysql> select a.Agent_code,a.Agent_Name,c.Customer_Name from agents a inner join customers c on a.Working_Area <> c.Customer_Area order by Customer_Name,Agent_code;
+------------+------------+--------------+
| Agent_code | Agent_Name | Customer_Name |
+------------+------------+--------------+
| A103       | agent3     | customer1    |
| A104       | agent4     | customer1    |
| A103       | agent3     | customer2    |
| A104       | agent4     | customer2    |
| A103       | agent3     | customer3    |
| A104       | agent4     | customer3    |
| A101       | agent1     | customer4    |
| A102       | agent2     | customer4    |
| A101       | agent1     | customer5    |
| A102       | agent2     | customer5    |
+------------+------------+--------------+
10 rows in set (0.00 sec)
```

Outer Join :

- **Left Outer Join,**
- **Right Outer Join,**

Cross Join:

```
mysql> select * from agents a CROSS JOIN customers c on a.Working_Area = c.Customer_Area;
+------------+------------+--------------+---------------+---------------+---------------+
| Agent_Code | Agent_Name | Working_Area | Customer_Code | Customer_Name | Customer_Area |
+------------+------------+--------------+---------------+---------------+---------------+
| A102       | agent2     | chennai      | C101          | agent1        | chennai       |
| A101       | agent1     | chennai      | C101          | agent1        | chennai       |
| A102       | agent2     | chennai      | C102          | agent2        | chennai       |
| A101       | agent1     | chennai      | C102          | agent2        | chennai       |
| A102       | agent2     | chennai      | C103          | agent3        | chennai       |
| A101       | agent1     | chennai      | C103          | agent3        | chennai       |
| A104       | agent4     | bangalore    | C104          | agent4        | bangalore     |
| A103       | agent3     | bangalore    | C104          | agent4        | bangalore     |
| A104       | agent4     | bangalore    | C105          | agent5        | bangalore     |
| A103       | agent3     | bangalore    | C105          | agent5        | bangalore     |
+------------+------------+--------------+---------------+---------------+---------------+
10 rows in set (0.00 sec)
```

Self Join:

DATABASE DESIGN:

Database design can be generally defined as a collection of tasks or processes that enhance the designing,development, implementation,and maintenance of enterprise data management systems.

Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space.

Therefore there has to be a brilliant concept of designing a database. The designer should follow the consistency and decide how the elements correlate and what kind of data must be stored.

Why Database Design is Important ?

1. Database designs provide the blueprint of how the data is going to be stored in a system.A proper design of a database highly affects the overall performance of any application.

2. The designing principles defined for a database gives a clear idea of the behavior of any application and how the requests are processed.

3. Another instance to emphasize the database design is that a proper database design meets all the requirements of users.

4. Lastly the processing time of an application is greatly reduced if the consistency of designing highly efficient databases are properly implemented.

5. A good database design starts with a list of the data that you want to include in your database and what you want to be able to do with the database later on.

6. This can be written in your won language , without any SQL.

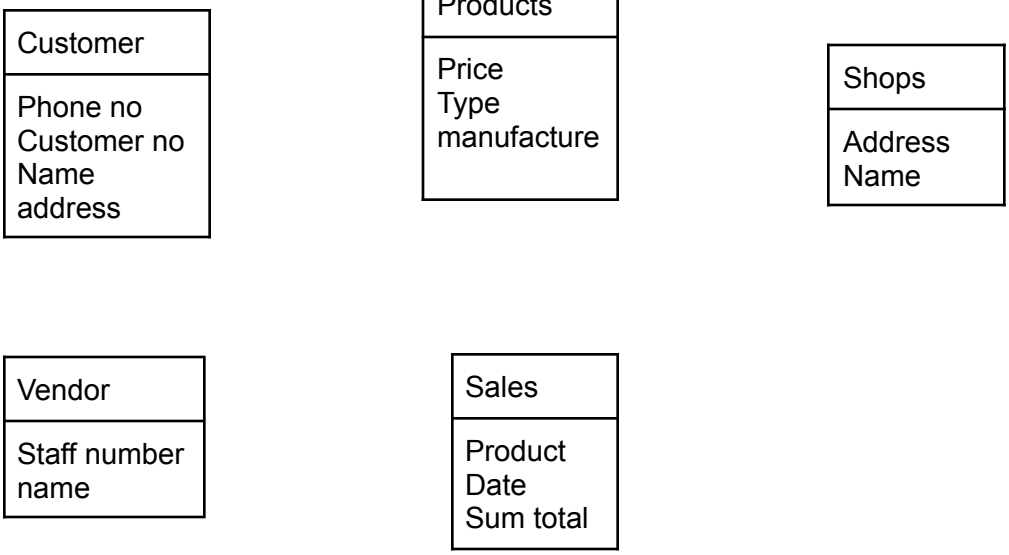
7. In this stage you must try not to think in tables or columns but just think : “What do I need to know ?” Don’t take this too lightly, because if you find out later that you forgot something ,usually you need to start all over. Adding thighs to your database is mostly a lot of work.

1.Identifying the Entities :

- The types of information that are stored in the database are called ‘entities’.
- These entities are of four kinds : people,things,events,locations.
- Everything you want to put in a database fits into one of these three categories.
- If the information you want to include doesn't fit into these categories then it is probably not an entity but a property of an entity,an attribute.
- Imagine that you are creating a website for a shop,what kind of information do you have to deal with?
- In a shop you sell your product to customers.The “**shop”** is a location; “**Sales”** is an event: “**Product”** are things : and “**Customer”** are people . These are all entities that need to be included in your database.
- But what other things are happening when selling a product ? A customer comes into the shop ,approaches the vendor ,asks a question and gets an answer .**Vendors** also participate and because vendors are people we need a vendor entity.

2.Identifying Attributes :

The data elements that you want to save for each entity are called “**attributes**”.

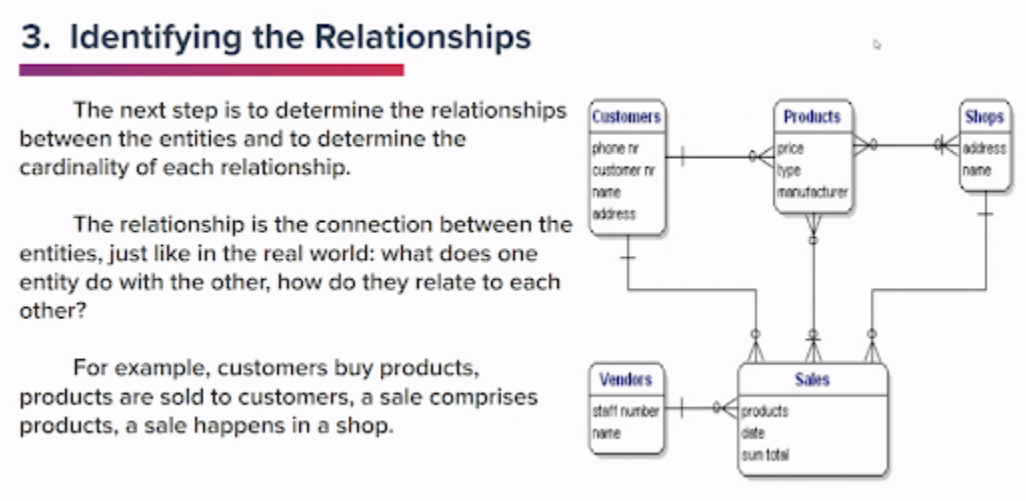


3.Identifying the relationships.

The next step is to determine the relationships between the entities and to determine the cardinality of each relationship.

The relationship is the connection between the entities,just like in the real world : what does one entity do with the other,how do they relate to each other?

For example, customers buy products,products are sold to customers, a sale comprises products and a sale happens in the shop.



4.Removing the Redundant Relationship :

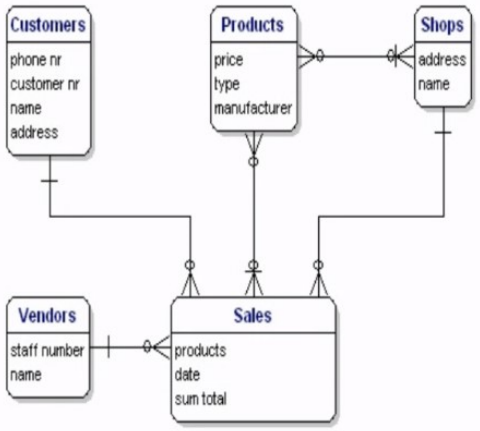
Sometimes in your model you will get a redundant relationship. These relationships are already indicated by other relationships,although not directly.

In the case of our example there is a direct relationship between customer and products.But there are also relationships between customer and products,But there are also relationships from customers to sales and from sales to products so indirectly there already is a relationship between customer and products through sales.The relationship ‘Customer<....>Product’ is made twice ,and one of them is therefore redundant .In this case, products are only purchased through sale, so the relationship ‘Customer <> Product’ can be deleted.

4. Removing the Redundant Relationships

Sometimes in your model you will get a 'redundant relationship'. These are relationships that are already indicated by other relationships, although not directly.

In the case of our example there is a direct relationships between customers and products. But there are also relationships from customers to sales and from sales to products, so indirectly there already is a relationship between customers and products through sales. The relationship 'Customers <----> Products' is made twice, and one of them is therefore redundant. In this case, products are only purchased through a sale, so the relationships 'Customers <----> Products' can be deleted.



5. Solving Many -to Many Relationships

Many-to-Many relationships (M:N) are not directly possible in a database .What a M:N relationShip says is that a number of records from one table belong to a number of records from another table.Somewhere you need to save which record these are and the solution is to split the relationship up in two one-to-many relationships.

This can be done by creating a new entity that is in between the related entities.In our example,there is a many-to many relationship s.

In the example there are two many -to-many relationships that need to be solved : ‘Product <....> Sales’ and ‘Product <.....> Shops'.For both situations there need to be created a new entity ,but what is that entity?

For the product <.....> Sales relationship, every sale includes more products.The relationship shows the content of the sale.So the entity is called ‘Sales details’.You could also name it ‘sold products’.

The Products<.....>Shops relationship shows which products are available in which shops ,also known as ‘stock’.

7. Defining the attributes Data Types :