SET OPERATIONS

The SET Operators in MySQL are basically used to combine the result of more than 1 select statement and return the output as a single result set. In SQL, 4 types of set operators are. They are as follows:

* UNION: It is used to combine two or more result sets into a single set, without duplicates.
* UNION ALL: It is used to combine two or more result sets into a single set, including duplicates.
* INTERSECT: It is used to combine two result sets and returns the data which are common in both the result set.
* EXCEPT: It is used to combine two result sets and returns the data from the first result set which is not present in the second result set.

Example:

SELECT FirstName, LastName, Gender, Department FROM EmployeeUK

UNION

SELECT FirstName, LastName, Gender, Department FROM EmployeeUSA;

Graphical user interface, text, application, email

Description automatically generated

SUBQUERIES

A MySQL subquery is a query nested within another query such as SELECT, INSERT, UPDATE or DELETE. Also, a subquery can be nested within another subquery.

Example:

SELECT lastName, firstName FROM employeeuk WHERE department IN (SELECT department FROM employeeusa WHERE department= 'IT')

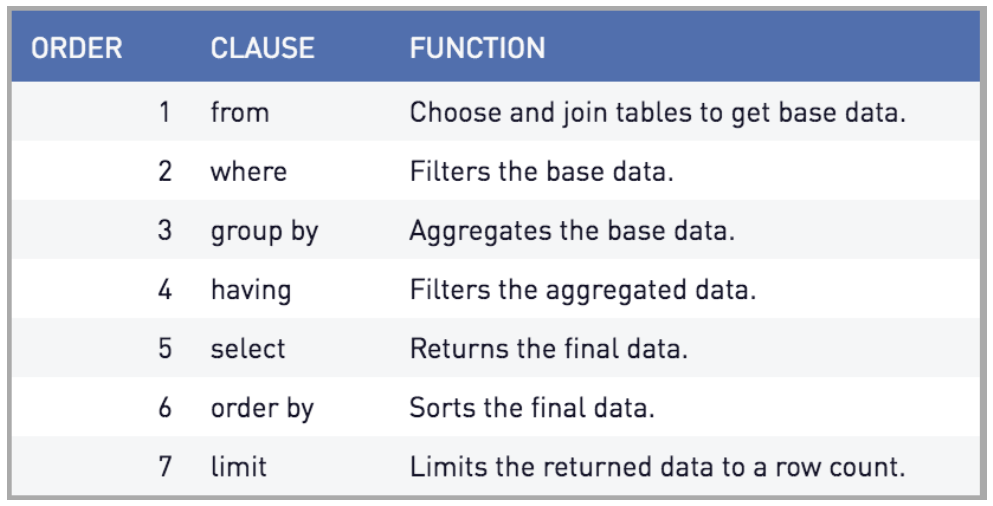
Graphical user interface, text, application, email

Description automatically generated

ORDER OF OPERATIONS OF QUERIES

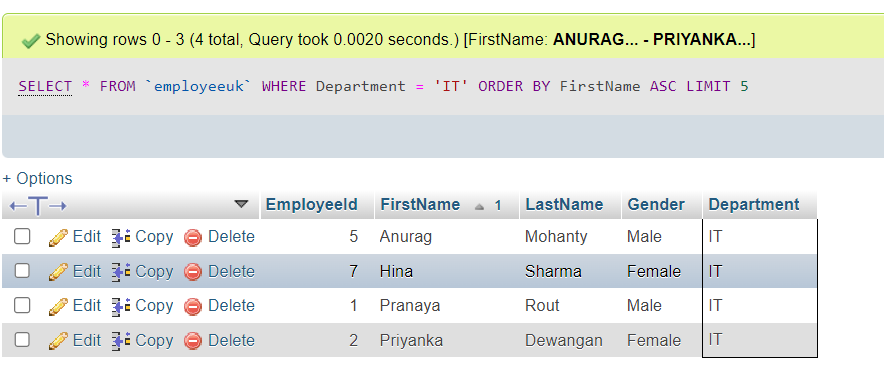
The SQL order of query execution/operation defines the order in which the clauses of a query are evaluated.

The order is as follows:



Example:

SELECT \* FROM `employeeuk` WHERE Department = 'IT' ORDER BY FirstName ASC LIMIT 5



creating, altering, and dropping tables

The CREATE, ALTER, and DROP statements are not limited to tables. We can use them to create other database objects like events, triggers, views, functions, or procedures. These statements are part of the Data Definition Language (DDL) of the SQL specification.

The CREATE statement is used to create tables. It is also used to create indexes, views, events, routines, and triggers. To create a table, we give a name to a table and to its columns. Each column has a data type. There are various data types in MySQL and choosing the correct datatype for the columns is part of the initial design of the database.

Example:

CREATE TABLE EmployeeUK

(

EmployeeId INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Gender VARCHAR(10),

Department VARCHAR(20)

);

The ALTER TABLE statement changes the structure of an existing table. It is possible to add a new column, delete a column, rename column, and table or change the data type of the table column.

Example:

ALTER TABLE EmployeeUK RENAME TO EmployeeUSA ;

The DROP TABLE statement drops or deletes a table from the database.

Example:

mysql> DROP TABLE EmployeeUSA;

Result: Query OK, 0 rows affected (0.00 sec) ;

ASSOCIATIONS

An association way in which two tables or classes are functionally connected to each other; it represents the possibility of a relationship between objects of the table.

Example: We want our customers to be able to place orders for the products that we sell in our online store, so we need to model the Order table and its association with the Customer table.

* An order is created when a customer decides to buy one or more of our products
* We need to know when the order was placed (date and time), and which customer representative sold the order

The association between the customer and the order will tell us which customer placed the order and correspondingly it will also tell us which orders were placed by a customer. Associations can also be called *relationships* and must include constraints about how few (at minimum) and how many (at maximum) individuals (objects) of one table may be connected to a single individual of the other table. This constraint is called *multiplicity* of the association.

* “Each customer places zero or more orders.” (The symbol \* in the diagram below means “many”, and any quantity more than one is considered “many” in a database.)
* “Each order is placed by one and only one customer.”

Diagram

Description automatically generated

joins and multiple table joins

A JOIN clause is used to combine rows from two or more tables, based on a related column between them. For example, let’s look at two tables, Orders, and Customers

Orders:

Background pattern

Description automatically generated

Customers:

Background pattern

Description automatically generated

Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship/association between the two tables above is the "CustomerID" column.

We can create *JOIN* statements that select records that have matching values in both tables.

Example:

SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate

FROM Orders

INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;

Background pattern

Description automatically generated

Other Types of Joins in MySQL

* INNER JOIN: Returns records that have matching values in both tables
* LEFT JOIN: Returns all records from the left table, and the matched records from the right table
* RIGHT JOIN: Returns all records from the right table, and the matched records from the left table
* CROSS JOIN: Returns all records from both tables

Diagram

Description automatically generated