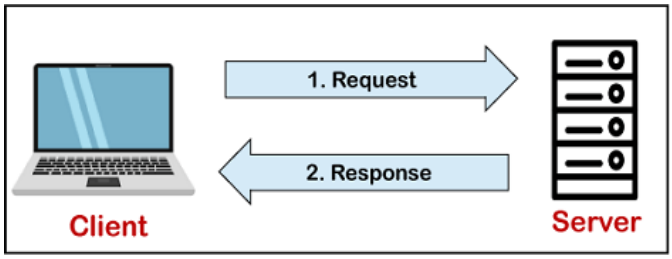
Database Handling Using PHP with MySQL: Introduction to MySQL: Database terms, Data Types. **Accessing MySQL** –Using MySQL Client and Using php MyAdmin, MySQL Commands, Using. PHP MySQL Functions, connecting to MySQL and Selecting the Database,
Executing Simple Queries, Retrieving Query Results, Counting Returned Records, Updating Records with PHP

What is MySQL?

MySQL is an open-source relational database management system (RDBMS). It is the most popular database system used with PHP. MySQL is developed, distributed, and supported by Oracle Corporation. MySQL was started in 1979 when MySQL's inventor **Michael Widenius** developed an in-house database tool called **UNIREG** for managing databases.

- The data in a MySQL database are stored in tables which consists of columns and rows.
- MySQL is a database system that runs on a server.
- MySQL is ideal for both small and large applications.
- MySQL is very fast, reliable, and easy to use database system. It uses standard SQL
- MySQL compiles on a number of platforms.

HOW MYSQL WORKS: MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services. Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched. The process of MySQL environment is the same as the client-server model.



The core of the MySQL database is the MySQL Server. This server is available as a separate program and responsible for handling all the database instructions, statements, or commands. The working of MySQL database with MySQL Server are as follows:

1. MySQL creates a database that allows you to build many tables to store and manipulate data and defining the relationship between each table.

2. Clients make requests through the GUI screen or command prompt by using specific SQL expressions on MySQL.

3. Finally, the server application will respond with the requested expressions and produce the desired result on the client-side.

MYSQL FEATURES: MySQL is a relational database management system (RDBMS) based on the SQL (Structured Query Language) queries. It is one of the most popular languages for accessing and managing the records in the table. MySQL is open-source and free software under the GNU license. Oracle Company supports it. The following are the most important features of MySQL

1. **Easy to use**: MySQL is easy to use. We have to get only the basic knowledge of SQL. We can build and interact with MySQL by using only a few simple SQL statements.
2. **It is secure**: MySQL consists of a solid data security layer that protects sensitive data from intruders. Also, passwords are encrypted in MySQL.
3. **Client/ Server Architecture**: MySQL follows the working of a client/server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they can query data, save changes, etc.
4. **Free to download**: MySQL is free to use so that we can download it from MySQL official website without any cost.
5. **It is scalable**: MySQL supports multi-threading that makes it easily scalable. It can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, we can increase this number to a theoretical limit of 8 TB of data.
6. **Speed**: MySQL is considered one of the very fast database languages, backed by a large number of the benchmark test.
7. **Compatible on many operating systems**: MySQL is compatible to run on many operating systems, like Windows, Linux, many varieties of UNIX (such as Sun Solaris, AIX, and DEC UNIX), OS/2, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).
8. **Allows roll-back**: MySQL allows transactions to be rolled back, commit, and crash recovery.
9. **Memory efficiency**: Its efficiency is high because it has a very low memory leakage problem.
10. **High Performance**: MySQL is faster, more reliable, and cheaper because of its unique storage engine architecture. It provides very high-performance results in comparison to other databases without losing an essential functionality of the software. It has fast loading utilities because of the different cache memory.
11. **Platform Independent**: It can download, install, and execute on most of the available

operating systems.

12. **Dual Password Support**: MySQL version 8.0 provides support for dual passwords: one is the current password, and another is a secondary password, which allows us to transition to the new password.

MySQL is an Open Source, Fast and Reliable Relational Database Management System (RDBMS) software like Oracle, Sybase, MS SQL Server etc. It was developed by Michael Widenius and AKA Monty and is alternative to many of the commercial RDBMS.

The main features of MySQL are-

- **Open Source & Free of Cost:**

  It is Open Source and available at free of cost.

- **Portability:**

  It can be installed and run on any types of Hardware and OS like Linux, MS Windows or Mac etc.

- **Security :**

  It creates secured database protected with password.

- **Connectivity**

  It may connect various types of Network client using different protocols and Programming Languages .

- **Query Language**

  It uses SQL (Structured Query Language) for handling database.

- In order to access data from the MySQL database, all program and user must use SQL (Structured Query Language). SQL is a set of commands that are recognized by all the RDBMSs and has become a standard language for database handling.
- SQL is a language that enables you to create and manage a relational database, in which all the information are kept in tables.
- There are numerous version of SQL. The original version was developed at IBM's San Jose Research Laboratory with a name of Sequel, as a part of System R project in 1970s. It was standardized by ANSI in 1986 by the name of SQL.

SQL is a Standard Query language whereas MySQL is a DBMS Software based on SQL.

Types of SQL Commands

MySQL follows SQL specifications for its commands. These SQL commands can be categorized as -

- **Data Definition Language (DDL)**

  These SQL commands are used to create, alter and delete database objects like table, views, index etc.

  Example : CREATE , ALTER , DROP etc.

- **Data Manipulation Language (DML)**

  These commands are used to insert, delete, update and retrieve the stored records from the table.

Ex. SELECT…., INSERT…, DELETE…, UPDATE…. etc.

- **Transaction Control Language (TCL)**

  These commands are used to control the transaction. Ex. COMMIT, ROLLBACK, SAVEPOINT etc.

- **Data Control Language (DCL)**

  These commands are used to manipulate permissions or access rights to the tables etc.

  Ex. GRANT , REVOKE etc.

Data type in MySQL

- Numeric Data Types:
  - **INTEGER or INT** – up to 11 digit number without decimal.
  - **SMALLINT** – up to 5 digit number without decimal.
  - **FLOAT (M,D) or DECIMAL(M,D) or NUMERIC(M,D)**

Stores Real numbers upto **M** digit length (including .) with **D** decimal places.

e.g. Float (10,2) can store 1234567.89

- Date & Time Data Types:
  - **DATE** - Stores date in YYYY-MM-DD format.
  - **TIME** - Stores time in HH:MM:SS format.

- String or Text Data Type:
  - **CHAR(Size)**
    - A fixed length string up to 255 characters. (default is 1)
  - **VARCHAR(Size)**
    - A variable length string up to 255 characters.

    Char, **Varchar**, **Date** and **Time** values should be enclosed with single (‘ ‘) or double (“”) quotes in MySQL.

MYSQL DATATYPES: Datatype means what kind of data that the variable will hold like integer, floating points, Boolean, etc. In MySQL, each database table has many columns and contains specific data types for each column. MySQL uses many different data types broken into three categories –

1. Numeric
2. Date and Time
3. String Types.

1. Nuemaric data type: MySQL has all essential SQL numeric data types. These data types can include the exact numeric data types (For example, integer, decimal, numeric, etc.), as well as the approximate numeric data types (For example, float, real, and double precision). It also supports BIT datatype to store bit values. In MySQL, numeric data types are categories into two types, either signed or unsigned except for bit data type. The following table contains all numeric data types that

support in MySQL:

Data Type Syntax	Description
INT	It is a normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. We can specify a width of up to 11 digits. It requires 4 bytes for storage.
BIGINT	It is a large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. We can specify a width of up to 20 digits. It requires 8 bytes for storage.
FLOAT(m,d)	It is a floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 10,2, where 2 is the number of decimals, and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a float type. It requires 2 bytes for storage.
DOUBLE(m,d)	It is a double-precision floating-point number that cannot be unsigned. You can define the display length (m) and the number of decimals (d). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a double. Real is a synonym for double. It requires 8 bytes for storage.
BIT(m)	It is used for storing bit values into the table column. Here, M determines the number of bit per value that has a range of 1 to 64.
BOOL	It is used only for the true and false condition. It considered numeric value 1 as true and 0 as false.

1. **Date and Time data type:** The MySQL date and time datatypes are as follows −
    - **DATE** − A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.

- **DATETIME** − A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973

  would be stored as 1973-12-30 15:30:00.
- **TIME** − Stores the time in a HH:MM:SS format.
- **YEAR(M)** − Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

2. String data type: The string data type is used to hold plain text and binary data, for example, files, images, etc. The following table illustrates all string data types that support in MySQL:

- **CHAR(M)** − A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- **VARCHAR(M)** − A variable-length string between 1 and 255 characters in length. For example, VARCHAR(25). You must define a length when creating a VARCHAR field.
- **BLOB or TEXT** − A field with a maximum length of 65535 characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data. The difference between the two is that the sorts and comparisons on the stored data are **case sensitive** on BLOBs and are **not case sensitive** in TEXT fields. You do not specify a length with BLOB or TEXT.
- **ENUM** − An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field.

MYSQL DML COMMANDS: Data Manipulation Language (DML) statements are used for managing data in the database. DML statements include structured query statements like select, insert, update, delete, etc. DML allows adding / modifying / deleting data itself.

DML Commands

1. INSERT
2. SELECT

3. UPDATE
4. DELETE

1. INSERT: It is used to insert or add new rows or records in the existing table.

Syntax:

insert into <table_name>values(<value1>,<value2>,<value3>…….,<valuen>);

table name: The name of the table In which the data needs to be inserted.

values: values for each column of the table.

To insert values in the table we first need to create a table which is a DDL(Data definition language) statement. Here, in the below all the examples we have created a table named students. we will show a demonstration of the DML statement in this table only. So, let's start with creating the students table.

Below is the query for creating statement:

create table students (roll_no int,student_name varchar(150),course varchar(150)); Once the table is created we can now insert values into it.

Below is the query for insert statement:

insert into students values(1,'ashish','java'); insert into students values(2,'rahul','C++'); select * from students;

Output:

	roll_no	student_name	course
1	1	ashish	java
2	2	rahul	C++

2. SELECT: It used to display the contents of the tables. It is also used to select data from the database.

Syntax:

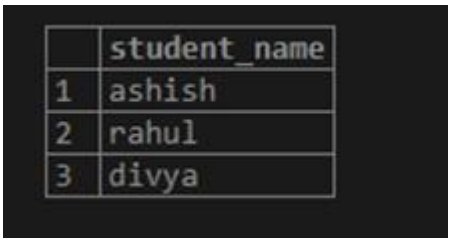
Select column1,column2,…..column n from table_table;

Where, column 1, column 2….column n are the attributes of the table

Example:

select student_name from students;

Output:



	student_name
1	ashish
2	rahul
3	divya

1. **UPDATE:** It is used to change the existing values of the column i.e, changing the name of the student or changing the course of any student.

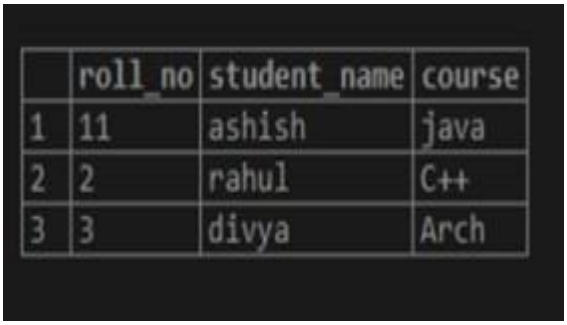
Syntax:

Update<table_name> set <column_name>=value where <condition>; **Table_name:** The name of the table in which the value is to be changed. **condition:** condition to get the specified row

Below is the query of the update statement:

update students set roll_no=roll_no+10 where student_name='ashish';

Output:



	roll_no	student_name	course
1	11	ashish	java
2	2	rahul	C++
3	3	divya	Arch

2. **DELETE:** Delete statement is used to delete rows of the table based on the specified conditions.

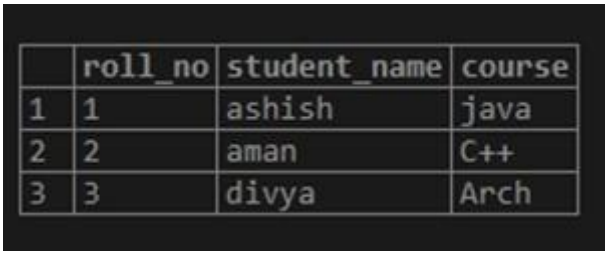
Syntax:

delete from <table_name> where <condition>;

table_name: Name of the table from which the data needs to be deleted. condition: Condition based on which the data is to be deleted.

select * from students;

Output:



	roll_no	student_name	course
1	1	ashish	java
2	2	aman	C++
3	3	divya	Arch

Database Handling commands in MySQL

- **Creating a Database.**
- The following command will create School database in MySQL. mysql> **CREATE DATABASE School;**
- **Opening a database**
- To open an existing database, following command is used. mysql> **USE school ;**
- **Getting listings of database and tables**
- mysql> **SHOW DATABASES;**
- mysql> **SHOW TABLES;**
- **Deleting a Database and Table** mysql> **DROP DATABASE School;** mysql> **DROP TABLE Student;**
- **Viewing Table Structure**
- mysql> **DESCRIBE Student;**

`Select database();` Shows the name of currently open

Creating Tables & Inserting records

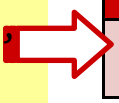
- Creating Simple Tables:

CREATE TABLE < Table Name>

(<Col name1><data type>[(size)],….);

Data types- INTEGER, NUMERIC(P,D), CHAR(n), VARCHAR(n), DATE etc.

```
mysql> CREATE TABLE Employee (empID
          integer,ename char(30),
            city char(25),
            pay decimal(10,2));
```



Employee			
empID	ename	city	pay

Inserting Records:

INSERT INTO <Table Name> VALUES (value1, vale2,……);

String and Date type values must be enclosed in single or double quotes. **mysql> INSERT INTO Employee VALUES (1,'Amitabh','Allahabad',15000); mysql> INSERT INTO Employee VALUES (2, 'Akbar', 'Dehradun',20000); mysql> INSERT INTO Employee VALUES (3, 'Anthony', 'Mumbai',10500);**

Making Simple Queries Using SELECT

The SELECT command of SQL, empower you to make a request (queries) to retrieve stored records from the database.

The syntax of SQL is given below-

SELECT < [Distinct | ALL] *| column name(s)> FROM <table(s)>

WHERE <condition>

ORDER BY <column name> [ASC | DESC] ;

Consider the table ***Student*** having some records as –

StID	Name	Fname	DOB	City	Class
S1	Amitabh	Harivansh Rai	1948-11-10	Allahabad	12
S2	Sharukh	Firoz	1970-05-10	Delhi	11
S3	Irphan	Akbar	1970-10-05	Jaipur	11
S4	Salman	Salim Javed	1972-04-10	Mumbai	10
S5	Abhishek	Amitabh	1975-03-12	Mumbai	10

- **Selecting all columns**

If you want to view all columns of the student table, then you should give the following command-

mysql> **SELECT * FROM Student ;**

MySQL will display the all records with all columns in the Student table.

***** Is used to represent all columns.

StID	Name	Fname	DOB	City	Class
S1	Amitabh	Harivansh Rai	1948-11-10	Allahabad	12
S2	Sharukh	Firoz	1970-05-10	Delhi	11
S3	Irphan	Akbar	1970-10-05	Jaipur	11
S4	Salman	Salim Javed	1972-04-10	Mumbai	10
S5	Abhishek	Amitabh	1975-03-12	Mumbai	10

Selecting columns

If you want to view only **Name** and **City** columns of the student table mysql> **SELECT Name, City FROM Student ;**

Name	City
Amitabh	Allahabad
Sharukh	Delhi
Irphan	Jaipur
Salman	Mumbai
Abhishek	Mumbai

mysql> **SELECT City, Name FROM Student ;**

City	Name
Allahabad	Amitabh
Delhi	Sharukh
Jaipur	Irphan
Mumbai	Salman
Mumbai	Abhishek

- **Eliminating Duplicate values in a column – DISTINCT**

mysql> **SELECT City FROM Student ;**

City
Allahabad
Delhi
Jaipur
Mumbai
Mumbai

Mumbai is repeated

MySQL assumes ALL keyword, if you are not using DISTICT keyword.

mysql> **SELECT DISTINCT City FROM Student ;**

City
Allahabad
Delhi
Jaipur
Mumbai

Only Unique Cities are displayed

Doing simple calculations

We can also perform simple calculations with SQL Select command. SQL provide a dummy table named DUAL, which can be used for this purpose.

mysql> **SELECT 4*3 ;**

We can also extend this idea with a columns of the existing table. mysql> **SELECT Name, Sal *12 FROM EMP ;**

Using Column Aliases

We can give a different name to a column or expression (Alias) in the output of a query.

mysql> **SELECT Name, Sal*12 AS 'Annual Salary' FROM EMP;** mysql> **SELECT Name, DOB AS 'Date of Birth' FROM Student;** mysql> **SELECT 22/7 AS PI FROM Dual;**

Note : When Alias name is a single word then single quotes is not required.

Selecting Specific Records – WHERE clause

WHERE <Condition>

We can select specific records by specifying conditions with WHERE clause.

mysql> **SELECT * FROM Student WHERE City='Mumbai';**

StID	Name	Fname	DOB	City	Class
S4	Salman	Salim Javed	1972-04-10	Mumbai	10
S5	Abhishek	Amitabh	1975-03-12	Mumbai	10

mysql**> SELECT Name, Fname, City from Student WHERE Class >10;**

Name	Fname	City	Class
Amitabh	Harivansh Rai	Allahabad	12
Sharukh	Firoz	Delhi	11
Irphan	Akbar	Jaipur	11

Selecting Specific Records – WHERE clause

Relational Operators

We can use the following Relational operators in condition.

=, > , < , >=, <=, <>, IS , LIKE, IN, BETWEEN

Logical Operators

We can use the following Logical Operators to connect two conditions.

OR , AND , NOT (!)

mysql> **SELECT Name, City from Student WHERE City <> 'Mumbai' AND Class>10;**

mysql> **SELECT * FROM Emp**
WHERE Sal >10000 OR Job ='Manager';

mysql> **SELECT * FROM Student WHERE NOT Grade='A';**
Selecting Specific Rows – WHERE clause

Specifying Range of Values – BETWEEN Operator
mysql> **SELECT * FROM Emp**
WHERE Sal BETWEEN 5000 AND 10000 ;
The same query can also be written as -
mysql> **SELECT * FROM Emp**
WHERE Sal >= 5000 AND Sal<=10000 ;
Other Logical operators also can be applied- mysql> **SELECT * FROM Emp**
WHERE NOT Sal BETWEEN 5000 AND 10000 ;

Specifying List – IN Operator
mysql> **SELECT * FROM Emp**
WHERE Sal IN (5000, 10000) ;
The same query can also be written as -
mysql> **SELECT * FROM Emp**
WHERE Sal = 5000 OR Sal =10000 ;

mysql> **SELECT * FROM Student**
WHERE City IN ('Mumbai', 'Delhi','Kanpur') ;

Selecting Specific Rows – WHERE clause

Pattern Matching – LIKE Operator

A string pattern can be used in SQL using the following wild card

- **%** **Represents a substring in any length**
- **_** **Represents a single character**

Example:
'A%' represents any string starting with 'A' character. **'_ _A'**
represents any 3-character string ending with 'A'. **'_B%'** represents any string having second character 'B' **'_ _ _'** represents any 3-letter string.

A pattern is case sensitive and can be used with LIKE operator.

mysql> **SELECT * FROM Student WHERE Name LIKE 'A%';** mysql> **SELECT * FROM Student WHERE Name LIKE '%Singh%';** mysql**> SELECT Name, City FROM Student**

WHERE Class>=9 AND Name LIKE '%Kumar%' ;

Searching NULL Values – IS Operator

mysql**> SELECT * FROM Student WHERE City IS NULL ;**

The **NOT** Operator can also be applied -

mysql**> SELECT * FROM Student WHERE City IS NOT NULL;**

Ordering Query Result – ORDER BY Clause

A query result can be orders in ascending (A-Z) or descending (Z-A) order as per any column. Default is Ascending order.

mysql**> SELECT * FROM Student ORDER BY City;**

To get descending order use **DESC** key word. mysql**> SELECT * FROM Student ORDER BY City DESC;**

mysql**> SELECT Name, Fname, City FROM Student**

Where Name LIKE 'R%' ORDER BY Class;

mysql**> SELECT Name, Basic+DA AS 'PAY' FROM Student ORDER BY F**

You can also use Alias column with ORDER BY clause

Inserting Records in a Table

You can insert record in the table by using by using the following DML command.

INSERT INTO <Table Name> [<Column list>] VALUES <list of values>

If value is not available for a column, NULL can be used.

Suppose a table STUDENT has been created as per given structure-

StID	NAME	FNAME	DOB	CITY	CLASS

We can insert a record as follows-

The sequence of values should match with order of columns.

mysql> INSERT INTO Student VALUES

('s1','Amitabh', 'Harivansh','1955-10-25', 'Mumbai', 12); mysql> INSERT INTO Student VALUES

('s2','Sharukh Khan', NULL,'1972-5-25', 'Delhi', 10);

mysql> INSERT INTO Student (StID, FName, Name, Class) VALUES ('s3','Amitabh', 'Abhishek', 10);

You can also define order of columns.

Inserting Records from Other Table

You can insert all or selected record(s) in the table from another table by using Select … command in place of Values. Suppose a table named NEWSTUDENT has been created and records to be inserted from OLDSTUDENT table having the same structure of columns.

mysql> INSERT INTO Newstudent VALUES (SELECET * FROM Oldstudent);

Both tables must have same column structure

mysql>INSERT INTO Newstudent VALUES

(SELECT * FROM Oldstudent WHERE City='Mumbai'); mysql> INSERT INTO Newstudent (StID, Name, Class)

VALUES (Select StID, Name,Class FROM Oldstudent WHERE Class>=11);

You can also select columns from both tables.

Deleting Records from the Table

You can delete all or selected record(s) from the table by using the following DML command.

DELETE FROM <Table Name> [WHERE <Condition>]

mysql> DELETE FROM Student

note : this command will delete all the records.

mysql> DELETE FROM Student WHERE City='Mumbai' ;

mysql> DELETE FROM Student WHERE Class >=11 ;

mysql> DELETE FROM Student WHERE Class <9 AND City='Delhi';

- **You can recall (Undelete) records by giving ROLLBACK command.**

  mysql> ROLLBACK ;

- **You can issue COMMIT command to record the changes permanently.**

  mysql> COMMIT;

Modifying Records –UPDATE Command

You can modify the values of columns of all or selected records in the table by using the following DML command.

UPDATE <Table Name>

SET <Column> = <Expression> [WHERE <Condition>]

mysql> UPDATE Student SET Class =10 ;

mysql> UPDATE Student SET FName= CONACT('Mr.', FName') ;

mysql> UPDATE Emp SET Sal = Sal+(Sal*10/100);

mysql> UPDATE Emp SET Sal = Sal+(Sal*10/100)

WHERE Sal <=10000;

mysql> UPDATE Emp SET City = 'Dehradun' WHERE CITY IS NULL;

Working with Tables

Creating Tables:

CREATE TABLE < Table Name> (<Col name><data type>[(size)][Constraints], ...)

Data types - Commonly used data types are- INTEGER, DECIMAL(P,D), NUMERIC(P,D), CHAR(n), VARCHAR(n), DATE etc.

Employee (EmpID, Ename, Sex, DOB, Pay)

mysql> CREATE TABLE Employee (**EmpID** integer, **Ename** char(20), **Sex** char(1), **Dob** Date, **Pay** decimal (8,2));

Creating Table with Constraints

One of the major responsibility of a DBMS is to maintain the Integrity of the data i.e. Data being stored in the Database must be correct and valid.

An Integrity Constraints are condition or checks applicable to a column or table which ensures the integrity and validity of data. The following constraints are available in MySQL.

Constraints	Description
NOT NULL	Ensures that a column cannot have NULL value.
PRIMARY KEY	Used to identify a row uniquely.
DEFAULT	Provides a default value for a column, if no value is given.
UNIQUE	Ensures that all values in a column are different.
CHECK	Ensures that value for a column should satisfy certain condition.
FOREIGN KEY	Used to ensure Referential Integrity of the data.

Implementing Constraints in the Table

- **NOT NULL**

  This constraints specifies that column must not contain NULL value i.e. value for the column must be given (**mandatory**)

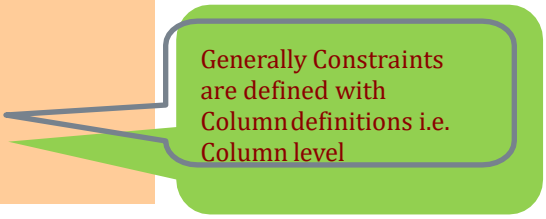
- **PRIMARY KEY**

  This constraints declare a column as the primary key. Since **Primary key must not have NULL value**, so it is used with NOT NULL constraints.

- **UNIQUE**

  This constraints ensures that the value for the column should be Unique i.e. **no two records have the same** (duplicate) value.

```
mysql> CREATE TABLE Student
(StCode char(3) NOT NULL PRIMARY KEY,
Stname char(20)  NOT NULL, StAdd  varchar(40),
AdmNo  char(5) UNIQUE,
StAge  integer CHECK (StAge>=5) );
```



Unique v/s Primary Key

- UNIQUE allows NULL values but PRIMERY KEY does not.
- A table may have multiple UNIQUE constraints, but there must be only one PRIMERY KEY constraints in a table.

Implementing Primary Key Constraints

Defining Primary Key at Column Level:

```
mysql> CREATE TABLE Student
( StCode    char(3)     NOT NULL PRIMARY KEY,
Stname      char(20)    NOT NULL,
StAge       int(2) );
```

Defining Primary Key at Table Level:

```
mysql> CREATE TABLE Student
( StCode  char(3)  NOT NULL, Stname char(20)  NOT NULL, StAge  int(2), PRIMARY KEY (StCode) );
```

Note : PRIMARY KEY Constraint is defined after all column definitions (Table Level).

A Composite (multi-column) Primary key can be defined as only a Table level whereas Single-column Primary key can be defined in both way i.e. Column level or Table level.

Handling Tables

☐ *Viewing Table Structure:*

You can view structure of any table after using database as-

DESC[RIBE] <table name>

mysql> DESC Student;

☐ *Deleting Table:*

You can delete an existing table as-

DROP TABLE [IF EXIST] <table name>

mysql> DROP TABLE Student;

☐ *Creating Table from Existing Table:*

CREATE TABLE <Table name> AS (<Select Query>);

mysql> CREATE TABLE Staff

(Select **empID, ename**, **sex** From Emp);

mysql> CREATE TABLE Staff

(Select * From Emp);

It will create identical table as Emp

Modifying Table Struct

You can alter (modify) the structure of existing table by the using **ALTER TABLE….** Command of MySQL. You can do the following with the help of ALTER TABLE.. Command.

- **Add a new Column or Constraints**
- **Modifying existing column (name, data type, size etc.)**
- **Delete an existing column or Constraints**
- **Changing Column Name**

ALTER TABLE <Table Name> ADD|MODIFY|DROP|CHANGE <Column Definition(s)>

Note : You can add/Delete/Modify multiple columns with single ALTER Command.

- **Adding new column**

ALTER TABLE <Table Name>
ADD <Column>[<data type> <size>][<Constraints>]

mysql> ALTER TABLE Student ADD (TelNo Integer);

mysql> ALTER TABLE Student ADD (Age Integer DEFAUL 10);

- **Modifying Existing Column**

ALTER TABLE <Table Name>
MODIFY <Column>[<data type> <size>] [<Constraints>] mysql> ALTER TABLE Student MODIFY Name VARCHAR(40); mysql> ALTER TABLE Employee MODIFY (Pay DECIMAL (10,2));

- **Removing Column & Constraints**

ALTER TABLE <Table Name>
DROP <Column name> |<Constraints> mysql> ALTER TABLE Student DROP TelNo; mysql> ALTER TABLE Emp DROP JOB, DROP Pay;