UNIT – 3
Introduction of SQL

BCA SEM – 2

WEB PROGRAMMING

Code : CS-09

Presented By : Dhruvita Savaliya

Topics :

- Working with MySQL using PhpMyAdmin
- SQL DML Statement

1. Insert Command
2. Update Command
3. Select Command
4. Delete Command

- PHP-MySQLi Connectivity
- PHP-MySQLi Functions

1. mysqli_connect
2. mysqli_close
3. mysqli_error, msyqli_errno
4. mysqli_select_db
5. mysqli_query
6. mysqli_fetch_array
7. mysqli_num_rows
8. mysqli_affected_rows
9. mysqli_fetch_assoc
10. mysqli_fetch_field
11. mysqli_fetch_object
12. mysqli_fetch_row
13. mysqli_insert_id
14. mysqli_num_fields
15. mysqli_data_seek

Working with MySQL using PhpMyAdmin :

- phpMyAdmin is an open-source software tool introduced on **September 9, 1998**, which is written in PHP. Basically, it is a third-party tool to manage the tables and data inside the database. phpMyAdmin supports various type of operations on **MariaDB** and **MySQL**. The main purpose of phpMyAdmin is to handle the administration of MySQL over the web.
- It is the most popular application for MySQL database management. We can create, update, drop, alter, delete, import, and export MySQL database tables by using this software. phpMyAdmin also supports a wide range of operation like **managing databases, relations, tables, columns, indexes, permissions, and users**, etc., on MySQL and MariaDB. These operations can be performed via user interface, while we still have the ability to execute any SQL statement.
- **Database -** phpMyAdmin supports databases, i.e.
- MySQL 5.5 or latest versions
- MariaDB 5.5 or latest versions

Features of phpMyAdmin :

- phpMyAdmin can create, alter, browse, and drop databases, views, tables, columns, and indexes.
- It can display multiple results sets through queries and stored procedures.
- phpMyAdmin use stored procedure and queries to display multiple results sets.
- It supports foreign keys and In noDB tables.
- phpMyAdmin can track the changes done on databases, views, and tables.
- We can also create PDF graphics of our database layout.
- phpMyAdmin can be exported into various formats such as XML, CSV, PDF, ISO/IEC 26300 - OpenDocument Text and Spreadsheet.
- It supports mysqli, which is the improved MySQL extension.
- phpMyAdmin can interact with 80 different languages.
- phpMyAdmin can edit, execute, and bookmark any SQL-statements and even batch-queries.
- By using a set of pre-defined functions, it can transform stored data into any format. **For example** - BLOB-data as image or download-link.
- It provides the facility to backup the database into different forms.
- **PHP -** We also need to install PHP 5.3 or upper version to support different functionalities. It contains different extensions to provide support for these functionalities.

- **Data :** Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.
- **Database :** A database is **an organized collection of structured information, or data, typically stored electronically in a computer system**. A database is usually controlled by a database management system (DBMS).
- **Table : A table has records (rows) and fields (columns)**. Fields have different types of data, such as text, numbers, dates, and hyperlinks. A record: Contains specific data, like information about a particular employee or a product.
- **Query :** A query can either be **a request for data results from your database or for action on the data, or for both**. A query can give you an answer to a simple question, perform calculations, combine data from different tables, add, change, or delete data from a database.

- **What is SQL?**
- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987
- **What Can SQL do?**
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert, update, delete records in a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views
- **What is RDBMS ?**
- RDBMS stands for Relational Database Management System.
- RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

SQL Commands :

- Structured Query Language(SQL) as we all know is the database language by the use of which we can perform certain operations on the existing database and also we can use this language to create a database. SQL uses certain commands like Create, Drop, Insert, etc. to carry out the required tasks.
- These SQL commands are mainly categorized into four categories as:
  - **DDL** – Data Definition Language
  - **DQL** – Data Query Language
  - **DML** – Data Manipulation Language
  - **DCL** – Data Control Language

DDL :

- DDL(Data Definition Language) : DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema.
- It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.
- Examples of DDL commands:
- **CREATE** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- **DROP** – is used to delete objects from the database.
- **ALTER**- is used to alter the structure of the database.
- **TRUNCATE**–is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT** –is used to add comments to the data dictionary.
- **RENAME** –is used to rename an object existing in the database.

DML :

- DML(Data Manipulation Language): The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.
- Examples of DML:
- **INSERT** – is used to insert data into a table.
- **UPDATE** – is used to update existing data within a table.
- **DELETE** – is used to delete records from a database table.

DQL :

- DQL (Data Query Language) :DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it.
- Example of DQL:
- **SELECT** – is used to retrieve data from the database.

DCL :

- DCL(Data Control Language): DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions and other controls of the database system.
- Examples of DCL commands:
- **GRANT** -gives user's access privileges to the database.
- **REVOKE-** withdraw user's access privileges given by using the GRANT command.

TCL :

- TCL(transaction Control Language): TCL commands deal with the transaction within the database.
- Examples of TCL commands:
- **COMMIT**– commits a Transaction.
- **ROLLBACK**– rollbacks a transaction in case of any error occurs.
- **SAVEPOINT**–sets a savepoint within a transaction.
- **SET TRANSACTION**–specify characteristics for the transaction.

PHP-MySQLi Connevtivity :

- In PHP you can easily do this using the **mysqli_connect()** function
- All communication between PHP and the MySQL database server takes place through this connection.
- Here're the basic syntaxes for connecting to MySQL using MySQLi and PDO extensions:

- **Syntax: MySQLi → Procedural way**

  $link = mysqli_connect("hostname", "username", "password", "database");
- **Syntax: MySQLi → Object Oriented way**

  $mysqli = new mysqli("hostname", "username", "password", "database");
- **Syntax: PHP Data Objects (PDO) way**

  $pdo = new PDO("mysql:host=hostname;dbname=database", "username", "password");

PHP-MySQLi Functions

1. mysqli_connect()
2. mysqli_close()
3. mysqli_error()
4. msyqli_errno()
5. mysqli_select_db()
6. mysqli_query()
7. mysqli_fetch_array()
8. mysqli_num_rows()
9. mysqli_affected_rows()
10. mysqli_fetch_assoc()
11. mysqli_fetch_field ()
12. mysqli_fetch_object()
13. mysqli_fetch_row()
14. mysqli_insert_id()
15. mysqli_num_fields()
16. mysqli_data_seek()

1. mysqli_connect() :

- The *mysqli_connect()* function in PHP is used to connect you to the database.
- In the previous version of the connection *mysql_connect()* was used for connection and then there comes *mysqli_connect()* where *i* means improved version of connection and is more secure than *mysql_connect()*.

➢ **Syntax :**

mysqli_connect ("host", "username", "password", "database_name")

- **Return values:**
- It returns an object which represent MySql connection. If connection failed then it return FALSE.

- ## Parameters used:
- **Host :** It is optional and it specify the host name or IP address. In case of local server <u>localhost</u> is used as a general keyword to connect local server and run the program.
- **Username :** It is optional and it specify mysql username. In local server username is *<u>root</u>*.
- **Password :** It is optional and it specify mysql password.(<u>blank</u>)
- **database_name :** It is database name where operation perform on data. It also optional.

➢ **Example :**

```
<?php
    mysqli_connect("localhost",
    "root", "", "dbname");
    if(mysqli_connect_error())
            echo "Connection Error.";
    else
            echo "Database Connection
    Successfully.";
?>
```

OUTPUT :

Warning: mysqli_connect(): (HY000/1049): Unknown database ‘dbname'
in **C:\xampp\htdocs\php2022 \Mysql\conn.php** on line **2**
Connection Error.

If database not exists

2. Mysqli_close() :

- **MySQLi Procedural procedure:** To close the connection in mysql database we use php function mysqli_close() which disconnect from database. It require a parameter which is a connection returned by the mysql_connect function.

- **Syntax:**

  mysqli_close(conn);

- If the parameter is not specified in mysqli_close() function, then the last opened database is closed. This function returns true if it closes the connection successfully otherwise it returns false.

- **Example :**

```
<?php
// Creating connection
$conn =
    mysqli_connect(“localhost”,”root”,””);
// Checking connection
if (!$conn) {
    die("Connection failed: " .
    mysqli_connect_error());
}
// Creating a database named newDB
$sql = "CREATE DATABASE  newDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully with
    the name newDB";
} else {
    echo "Error creating database: " .
    mysqli_error($conn);
}
// closing connection
mysqli_close($conn);
?>
```

- **MySQLi Object-oriented procedure::**
- To close the connection in mysql database we use php function conn->close() which disconnect from database.

➢ **Syntax:**

- conn->close();

➢ **Example :**

```
 <?php
// checking connection
$conn = new mysqli("localhost",
    "root", " ", "dbname");

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " .
    $conn->connect_error);
}
//Close the connection
$conn->close();
?>
```

- **Using PDO procedure:**
  To close the connection in MySQL database in PDO procedure we set the connection name to null which disconnect from the database.

```
<?php
try {
        $conn = new PDO("mysql:host=localhost;",”root”,“ ”);
        //with database   $conn = new PDO("mysql:host=localhost;dbname=php_new",”root”, “ ”);

    // setting the PDO error mode to exception
        $conn->setAttribute(PDO::ATTR_ERRMODE,
                                  PDO::ERRMODE_EXCEPTION);
        $sql = "CREATE DATABASE abc";
        $conn->exec($sql);   // using exec() because no results are returned
        echo "Database created successfully with the name abc";
    }
    catch(PDOException $e)
    {
            echo $sql . "" . $e->getMessage();
    }
$conn = null;
?>
```

3. mysqli_error() :

- The **mysqli_error() function** is used to return the error in the most recent MySQL function call that failed. If there are multiple MySQL function calls, the error in the last statement is the one that is pointed out by the function.
- **Syntax:**
- mysqli_error("database_name") **Parameters:** This function accepts single parameter as mentioned above and described below:
- **database_name:** It is the database on which operations are being performed. It is a mandatory parameter.

```
<?php
  //Creating a connection
  $con = mysqli_connect("localhost", "root", "", "php_new");
  //Error
  $error = mysqli_error($con);
  if(mysqli_error($con))
        echo("Error Occurred: ".$error);
  else
        echo "hello";
  //Closing the connection
  mysqli_close($con);
?>
```

4. mysqli_errno() :

- The errno / mysqli_errno() function returns the last error code for the most recent function call, if any.
- **Syntax :**
  int mysqli_errno (mysqli $link)

```
<?php
 $conn = new mysqli("localhost","root","","php_new");
  if ($conn->connect_error) {
    die('Connect Error (' . mysqli_connect_errno() . ') '. mysqli_connect_error());
    exit();
  }
  else
     echo "Connected";

  mysqli_close($conn);
?>
```

5. mysqli_select_db() :

- The **mysqli_select_db()** function accepts a string value representing an existing database and, makes it as a the default database.

➢ **Syntax :**

```
mysqli_select_db($con, db_name)
```

➢ **Parameter :**

- **con(Mandatory) :** This is an object representing a connection to MySQL Server.
- **name(Mandatory) :** This is a string value representing the name of an existing database which you need to make as the default database.
- Return Values
- The PHP mysqli_select_db() function returns a boolean value which is, *true* if the operation is successful and, *false* if not.

```
<?php
  $conn = mysqli_connect("localhost","root","");
  $a=mysqli_select_db($conn, "php_new");
  if (!$a) {
    echo "not";
  }
  else
     echo "Connected";

  mysqli_close($conn);
?>
```

6. mysqli_query() :

- The **mysqli_query()** function accepts a string value representing a query as one of the parameters and, executes/performs the given query on the database.
- **Syntax :**

  mysqli_query($con, query)
- **Parameters :**
- **con(Mandatory) :** This is an object representing a connection to MySQL Server.
- **query(Mandatory) :** This is a string value representing the query to be executed.
- **mode(Optional) :** This is a integer value representing the result mode.

  You can pass *MYSQLI_USE_RESULT* or *MYSQLI_STORE_RESULT* as values to this parameter.

```
<?php
  $con = mysqli_connect("localhost", "root", "password",
  "mydb");
  mysqli_query($con, "CREATE TABLE IF NOT EXISTS
  my_team(ID INT, First_Name VARCHAR(255), Last_Name
  VARCHAR(255)");
  mysqli_close($con);
?>
```

7. mysqli_fetch_array() :

- The **mysqli_fetch_array() function** is used to fetch rows from the database and store them as an array. The array can be fetched as an associative array, as a numeric array or both.
- Associative arrays are the arrays where the indexes are the names of the individual columns of the table. On the other hand, numeric arrays are arrays where indexes are numbers, with 0 representing the first column and n-1 representing the last column of an n-column table.
- **Syntax:**

  mysqli_fetch_array ("query", "mode")
- **Parameters:**
- **database_name:** It is the database on which operations are being performed. It is a mandatory parameter.
- **mode:** It can have three values – MYSQLI_ASSOC, MYSQLI_NUM, and MYSQLI_BOTH. MYSQLI_ASSOC makes the function behave like mysqli_fetch_assoc() function, fetching an associative array, MYSQLI_NUM makes the function behave like mysqli_fetch_row() function, fetching a numeric array while MYSQLI_BOTH stores the data fetched in an array that can be accessed using both column indexes as well as column names.

```
<?php
    $con=mysqli_connect("localhost", "root", " ", "php_new");
    //or mysqli_select_db($con,$db);
    if($con)
    {
            echo "<br>connected<br>“
            $sel=mysqli_query($con,"select * from demo");
            while($row=mysqli_fetch_array($sel))
            {
                    echo $row['id'];
                    echo $row['name'];
                    echo "<br>";
            }
    }
    else
            die("sorry".mysqli_connect_error());
?>
```

8. mysqli_num_rows() :

- The mysqli_num_rows() function is an inbuilt function in PHP which is used to return the number of rows present in the result set. It is generally used to check if data is present in the database or not. To use this function, it is mandatory to first set up the connection with the MySQL database.
- **Syntax:**

```
mysqli_num_rows ( $result );
```

- **Parameters:** This function accepts single parameter $result (where $result is a MySQL query set up using mysqli_query()). It is a mandatory parameter and represents the result set returned by a fetch query in MySQL.
- **Return Value:** It returns the number of rows present in a result set. If no rows match the given criteria then it returns false instead.

```
<?php
$connection = mysqli_connect("localhost", "root", "","php_new");
    if (mysqli_connect_errno())        {
            echo "Database connection failed.";
    }
    $query = "SELECT *from demo";
    $result = mysqli_query($connection, $query);
    if ($result)        {
            // it return number of rows in the table.
            $row = mysqli_num_rows($result);
            if ($row)    {
                                    echo "Number of row in the table : " . $row;
                        }
            // close the result.
            mysqli_free_result($result);
    }
    mysqli_close($connection);
?>
```

9. mysqli_affected_rows() :

- It returns the number of affected rows in the previous SELECT, INSERT, UPDATE, REPLACE, or DELETE query.
- **Return Values :**
- An integer > 0 indicates the number of rows affected.0 indicates that no records were affected. -1 indicates that the query returned an error
- **Parameters :**
  connection : It specifies the number of connections to use in php

```
<?php
   $input = mysqli_connect("localhost","root","","php_new");
   mysqli_query($input,"SELECT * FROM demo");
   echo "Affected rows: " . mysqli_affected_rows($input);

   mysqli_close($input);
?>
```

10. mysqli_fetch_assoc() :

- The *mysqli_fetch_assoc()* function accepts a result object as a parameter and, retrieves the contents of current row in the given result object, and returns them as an associative array.

➢ **Syntax :**

mysqli_fetch_assoc($result);

➢ **Parameters :**

- **result(Mandatory) :** This is an identifier representing a result object.

➢ **Return Values :**

- The PHP mysqli_fetch_assoc() function returns an associative array which contains the current row of the result object. This function returns NULL if there are no more rows.

```
<?php
$con = mysqli_connect("localhost", "root", "", "php_new");
  //Retrieving the contents of the table
  $res = mysqli_query($con, "SELECT * FROM demo");

  //Fetching all the rows as objects
  while($obj = mysqli_fetch_assoc($res)){
    echo " ID: ".$obj["id"]."<br>";
    echo "First_Name: ".$obj["name"]."<br>";
  }
  //Closing the statement
  mysqli_free_result($res);

  //Closing the connection
  mysqli_close($con);
?>
```

11. mysqli_fetch_field() :

- The *mysqli_fetch_field()* function accepts a result object as a parameter and returns the definition information of the next column/field in the form of an object.

➢ **Syntax :**

```
mysqli_fetch_field($result);
```

➢ **Parameters :**

- **result(Mandatory) :**This is an identifier representing a result object.

➢ **Return Values :**

- The PHP mysqli_fetch_field() function returns an object containing the definition information of a field in the given result. This function returns *FALSE* in case of no information.

• Object properties :

Property	Description
name	The name of the column
orgname	Original column name if an alias was specified
table	The name of the table this field belongs to (if not calculated)
orgtable	Original table name if an alias was specified
def	Reserved for default value, currently always ""
db	The name of the database
catalog	The catalog name, always "def"
max_length	The maximum width of the field for the result set.
length	The width of the field, as specified in the table definition.
charsetnr	The character set number for the field.
flags	An integer representing the bit-flags for the field.
type	The data type used for this field
decimals	The number of decimals used (for integer fields)

```
<?php
$con = mysqli_connect("localhost", "root", "", "php_new");
  //Retrieving the contents of the table
  $res = mysqli_query($con, "SELECT * FROM demo");
  //Fetching all the rows as objects
  while($obj = mysqli_fetch_field($res)){
    echo(“Field Name : ".$obj->name."<br>");
    echo(“Table: ".$obj->table."<br>");
    echo(“max_length: ".$obj->max_length."<br>");
    echo("flags: ".$obj->flags."<br>");
    echo("type: ".$obj->type."<br>");
  }
  //Closing the statement
  mysqli_free_result($res);

  //Closing the connection
  mysqli_close($con);
?>
```

12. mysqli_fetch_object() :

- The *mysqli_fetch_object()* function accepts a result object as a parameter and, retrieves the contents of current row in the given result and returns them as an object.

- **Syntax :**

  mysqli_fetch_object($result, [$class_name, $params]);

- **Parameters :**
- **result(Mandatory) :** This is an identifier representing a result object.
- **class_name(Optional) :** The name of the class to instantiate, set the properties of and return.
- **params(Optional) :** An array representing the optional parameters.

- **Return Values :**
- The PHP mysqli_fetch_object() function returns an object (with string properties) which holds the current row of the result object. This function returns NULL if there are no more rows.

```
<?php
  $con = mysqli_connect("localhost", "root", "", "php_new");
 //Retrieving the contents of the table
 $res = mysqli_query($con, "SELECT * FROM demo");

 while($obj = mysqli_fetch_object($res)){
          echo "ID: ".$obj->id."<br>";
          echo "First_Name: ".$obj->name."<br>";
 }
 //Closing the statement
 mysqli_free_result($res);

 //Closing the connection
 mysqli_close($con);
?>
```

Difference between mysqli_fetch_array() and mysqli_fetch_object() Function:

	mysqli_fetch_array()	mysqli_fetch_object()
1.	It fetches a result row as an associative array and a numeric array.	It is used to return the row of a result-set
2.	Its syntax is -: **$mysqli_result -> fetch_array(result_type)**	Its syntax is -: **$mysqli_result -> fetch_object(classname, params)**
3.	It takes one parameter that is a result.	It takes two parameters that are class name and array of parameters
4.	Its return value is array of string.	It returns an object with string properties for the fetched row.
5.	It is supported in PHP version 5+	It is supported in PHP version 5+

13. mysqli_fetch_row() :

- The *mysqli_fetch_row()* function accepts a result object as a parameter, retrieves the contents of its current row as an array of strings.

- **Syntax :**

  mysqli_fetch_row($result);

- **Parameters :**

- **result(Mandatory) :** This is an identifier representing a result object.

- **Return Values :**

- The PHP mysqli_fetch_row() function returns an array (string) which contains the values in the row to which the data seek is currently pointed.

```
<?php
  $con = mysqli_connect("localhost", "root", "", "php_new");
 //Retrieving the contents of the table
 $res = mysqli_query($con, "SELECT * FROM demo");

 while ($row = mysqli_fetch_row($res)) {
   echo "ID: ".$row[0]."<br>" ;
   echo "Name: ".$row[1]."<br> ";
 }
 //Closing the statement
 mysqli_free_result($res);

 //Closing the connection
 mysqli_close($con);
?>
```

14. mysqli_insert_id() :

- If you have a table with an AUTO_INCREMENT attribute for a column and, if your last MySQLi function call executes INSERT or UPDATE statement. The **mysqli_insert_id()** function returns the auto generated id of the last executed query.

➢ **Syntax :**

mysqli_insert_id($con)

➢ **Parameters :**

- **con(Mandatory) :**This is an object representing a connection to MySQL Server.

➢ **Return Values :**

- PHP mysqli_insert_id() function returns the value of the "**Auto Increment**" column in the last query In case it is INSERT or, UPDATE operation.
- If the last executed query is not INSERT or, UPDATE or, if the table <u>**doesn't** have any column/field with "AUTO_INCREMENT" attribute, this function returns *0*.</u>

```
<?php
   $con = mysqli_connect("localhost", "root", "", "php_new");
   //Retrieving the contents of the table
   $res = mysqli_query($con, "insert into demo values('?','xyz')");
   $id = mysqli_insert_id($con);
   echo "Insert ID: ".$id;
   mysqli_close($con);
?>
```

15. mysqli_num_fields() :

- The *mysqli_num_fields()* function accepts a result object as a parameter, retrieves and returns the number of fields in the given object.

➢ **Syntax :**

```
mysqli_num_fields($result);
```

➢ **Parameters :**

- **result(Mandatory) :** This is an identifier representing a result object.

➢ **Return Values :**

- The PHP mysqli_num_fields() function returns an integer value specifying the **number of fields** in the given result object.

```
<?php
  $con = mysqli_connect("localhost", "root", "", "php_new");
  //Retrieving the contents of the table
  //$res = mysqli_query($con, "insert into demo values('?','xyz')");
  $res=mysqli_query($con,"select *from demo");
  $count = mysqli_num_fields($res);
?>
```

16. mysqli_data_seek() :

- The *mysqli_data_seek()* function accepts a result object and an integer value representing an offset, as parameters, and moves the data seek of the given result object to the specified row.

➢ **Syntax :**

```
        mysqli_data_seek($result, $offset);
```

➢ **Parameters :**

- **result(Mandatory) :** This is an identifier representing a result object.
- **offset(Mandatory) :** This is an integer value representing an offset to which you need to move the data seek in the given result object.

➢ **Return Values :**

- The PHP mysqli_data_seek() function returns a boolean value which is, *TRUE* incase of success and *FALSE* incase of failure.

```
<?php
    $con = mysqli_connect("localhost", "root", "", "php_new");
   //Retrieving the contents of the table
   //$res = mysqli_query($con, "insert into demo values('?','xyz')");
   $res=mysqli_query($con,"select *from demo");
   mysqli_data_seek($res, 1); //it will return 2nd record
    $row = mysqli_fetch_row($res);
   print_r($row);
?>
```

- **SQL Constraints :**
- SQL constraints are used to specify rules for the data in a table.
- Constraints 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 table. If there is any violation between the constraint and the data action, the action is aborted.
- Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.
- The following constraints are commonly used in SQL:
- NOT NULL - Ensures that a column cannot have a NULL value
- UNIQUE - Ensures that all values in a column are different
- PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- FOREIGN KEY - Prevents actions that would destroy links between tables
- CHECK - Ensures that the values in a column satisfies a specific condition
- DEFAULT - Sets a default value for a column if no value is specified

- **Create Command :**
- **Create Database :**

CREATE DATABASE *databasename*;

- <u>**Create Table :**</u>

```
CREATE TABLE table_name (
      column1 datatype[size],
      column2 datatype[size],
      column3 datatype[size],
     ....
  );
```

- **Drop Command:**
- **Drop Database :**

DROP DATABASE *databasename*;

- **Drop Table :**

DROP TABLE *table_name*;

- **Insert :**

  INSERT INTO *table_name* VALUES (*value1*, *value2*, *value3*, ...);

- **Update :**

  UPDATE *table_name*

  SET *column1* = *value1*, *column2* = *value2*, ...

  WHERE *condition*;

- **Delete :**

  DELETE FROM *table_name* WHERE *condition*;

- **Select :**

  SELECT *FROM table_name;

  SELECT *column_name(s)* FROM *table_name* WHERE *condition*

  LIMIT *number*;

  (With where clause)

- **Truncate :**

  TRUNCATE TABLE table_name;

- **Alter :**

➢ **ALTER TABLE - ADD Column**

ALTER TABLE *table_name*
ADD *column_name datatype*;

➢ **ALTER TABLE - DROP COLUMN**

ALTER TABLE *table_name*
DROP COLUMN *column_name*;

➢ **ALTER TABLE - ALTER COLUMN**

ALTER TABLE *table_name*
ALTER COLUMN *column_name datatype*;

- **NOT NULL / UNIQUE :**

```
CREATE TABLE table_name (
    column1 datatype[size]  AUTO_INCREMENT,
    column1 datatype[size] NOT NULL,
    column1 datatype[size] UNIQUE,
   ....
);
```

- **CHECK :**

```
CREATE TABLE table_name (
    column1 datatype[size],
    column1 datatype[size],
CHECK (field condition)
....
);
```

- **DEFAULT :**

```
CREATE TABLE table_name (
    column1 datatype[size],
    column1 datatype[size] DEFAULT 'VALUE',
....
);
```

- **PRIMARY KEY :**

CREATE TABLE *table_name* (

column1 datatype[size] PRIMARY KEY, *column1 datatype[size]*,

OR PRIMARY KEY(unique field name)

);

- **FORIEGN KEY :**

CREATE TABLE table_name1

(

Column_Name_1 data type (**size of** the column_1),

Column_Name_2 data type (**size of** the column_2),

……,

Column_Name_N data type (**size of** the column_N)

FOREIGN KEY REFERENCES Table_Name2 (Column_Name)

) ;

```
CREATE TABLE Orders (
   OrderID int NOT NULL,
   name varchar(10),
   PRIMARY KEY (OrderID),
   FOREIGN KEY (OrderID) REFERENCES sales(id)
);
```