**Operators :**

|  |  |  |
| --- | --- | --- |
| **!=** | - | NOT EQUAL TO |
| **>** | - | GREATER THAN |
| **<** | - | LESS THAN |
| **AND / OR** | - |  |
| **<=** | - | LESS THAN OR EQUAL TO |
| **>=** |  | GREATER THAN OR EQUAL TO |

**Database**

|  |  |
| --- | --- |
| **Command** | **Description** |
| **CREATE DATABASE database\_name;** | Creating a new database |
| **SHOW DATABASES;** | Show available databases |
| **USE database\_name;** | Use Database |
| **DROP database\_name;** | Delete a database |

# Tables

|  |  |  |
| --- | --- | --- |
| **CREATE TABLE** table\_name ( column\_name **datatypes ] [column constraints],**  **…**  **[table constraints]**  **...);** | **-** | Creats a table with column, and constraints..  (min 1 column) |
| **ALTER TABLE** table\_name; | **-** | Modifies the table |
| **DROP table table\_name;** | **-** | Deletes a table |
| **DESCRIBE TABLE** tablename | **-** | Describes the specifications for the columns  with-in a table |
| **RENAME TABLE** OldName **TO** NewName | **-** | Gives the table a new name |
|  |  |  |
|  |  |  |
|  |  |  |

## Altering Tabels

|  |  |  |
| --- | --- | --- |
| **ALTER TABLE table\_name [DROP Column column\_name]**  **[ADD COLUMN column\_name conditions] /** | - | Dropping a column,  adds a new column. |
| **ALTER TABLE table\_name**  **[ADD CONSTRAINT]**  **[DROP INDEX];** | - | Adds a constraint or drops one |

# E.G

**CREATE TABLE** co\_employees (

ID **INT** **PRIMARY KEY AUTO\_INCREMENT,**

Employername **VARCHAR**(255) **NOT NULL**,

Gender **CHAR**(1) **NOT** **NULL**,

ContactNumber **VARCHAR**(255),

AGE **INT** **NOT** **NULL**,

Registered **TIMESTAMP** **NOT** **NULL** **DEFAULT** **NOW**());

# Constraints

## Table Constraints

|  |  |  |
| --- | --- | --- |
| **Command** | **-** | **Description** |
| **Primary KEY**(column\_one, column\_two, column\_three) | - | Specifies the column(s) is a primary key uniquely identifies each row in a table |
| **FOREIGN KEY(Column\_id) REFERENCES Table\_name(id) ON DELETE CASCADE ON UPDATE RESTRICT,** | - | A key whom connect two different tables |
|  | - |  |
|  | - |  |

## Column Constraints

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Support** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Column

### Alter table columns:

|  |  |  |
| --- | --- | --- |
| **ALTER TABLE table\_name ADD COLUMN column\_name data type [Constraints];** | **-** | Adding a new column |
| **ALTER TABLE table\_name MODIFY COLUMN column\_name datatype** | **-** | MODIFY an existing column |

## Column Data types

|  |  |  |
| --- | --- | --- |
| **CHAR(size)** | - | Implies there should be a text.  Used only for abbrevations with 1 or 2 characters |
| **VARCHAR(size)** | - | Implies there should be textural statements, uses for long text e.g above 5 characters |
| **NOT NULL** | - | Indicates the cell can not be empty |
| **UNIQUE** | - | The values with-in the cell has to be unique |
| **DEFAULT** | - | Sets a default value in a column |
| **AUTO\_INCREMENT** | - | Values in a column should automatically increase by one for each new record. |

### NUMERIC DATA TYPES :

|  |  |  |
| --- | --- | --- |
| **INT** | - | Implies there should be only INTEGERS ( 2147483648 -o -2147483647 ) |
| **FLOAT(m,d)** | **-** | Non integers / decimal.   m - Refers to the total number of digits the  FLOAT stores while.  d -refers to the number of digits after the decimal point. |
| **DOUBLE(m,d)** | **-** | is also used to store non integers as approximate values. It uses 8 bytes of storage and can be used to store numbers with higher precision. It is accurate up to about 14 decimal places. Similar to  FLOAT , you can specify the total number of digits ( m ) and the number of digits after the decimal point ( d ). |
| **DECIMAL(m,d)** | **-** | **DECIMAL** is used to store non-integers as exact values. Similar to **FLOAT** and **DOUBLE**, you can specify the total number of digits (m) and the number of digits after the decimal point (d) when using **DECIMAL** is commonly used to store monetary data where **precision is important** |

### DATE AND TIME DATA TYPES:

|  |  |  |
| --- | --- | --- |
| **DATE** | **-** | The data type is used to store time in the **YYYY-MM-DD** format, with a supported range of '-**1000 -01-01** to **'9999-12-31**' |
| **DATETIME** | **-** | The data type is used to store a date and time combination in the **YYYY-MM-DD** **HH:MI:SS** format. The supported range is from **'1000-01-01** **00:00:00'** to '**9999-12-31 23:59:59**' . |
| **TIME** | **-** | The data type is used to store time in the **HH:MI:SS** format, with a supported range of '-**838:59:59'** to '**838:59:59**' . |
| **TIMESTAMP** | **-** | The data type is also used to store a date and time combination in the **YYYY-MM-DD** **HH:MI:SS** format. The supported range is from '**1970-01-01 00:00:01**' UTC to '**2038-01-09 03:14:07**' UTC. |
| **YEAR** | **-** | The data type is used to store a year in either a two-digit or a four-digit format. Values allowed in four-digit format are from **1901** to **2155** . Values allowed in two-digit format are from 1 to 69 (representing years from 2001 to 2069) and 70 to 99 (representing years from 1970 to 1999). |

# SELECT Syntax

**SELECT** \* ( Everything ) ( columnName ) {**AS** new alias} [**ORDER BY** column] [**DESC** / other way] **FROM** table\_name [**WHERE** condition];

## Basic Select Syntax

|  |  |  |  |
| --- | --- | --- | --- |
| **Command** |  | **Description** | **Example** |
| **SELECT** \* **FROM** TableName | - | Selects everything in the chosen table | **SELECT** \* **FROM** MyTable; |
| **SELECT** column, column… **FROM** table\_name | - | Selects (a) chosen column(s) from the chosen table | **SELECT** Name, ContactInfo **FROM** MyTable; |
| **SELECT** column **AS** NickName **FROM** Table\_name | - | Selecting a column, adding a nickname from the choosen table | **SELECT** Worker AS Employee **FROM** MyTable |
| **SELECT** \* **FROM** table\_ name **LIMIT ( 1-999 )** | - | The condition limits how many rows shown | **SELECT** \* **FROM** MyTable **LIMIT** 10 |
| **SELECT DISTINCT**(column) **FROM** TableName | - | Removes any duplicates in the column |  |
| **SELECT** \* **FROM** table\_name **WHERE** id!= 1 | - | Selects everything except id 1 |  |
| **SELECT** \* **FROM** table\_name **WHERE** ID **BETWEEN** 1 **AND** 3; | - | Selects only ids between 1 - 3 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Command** | **-** | **Description** | **Example** |
| **SELECT** \* **FROM** table\_name **WHERE** column\_name **LIKE ‘(**letters**)%’** | - | SELECTS only columns with the same identity |  |
| **SELECT** \* **FROM table\_name WHERE** id **IN** (5,4,2); | **-** | Selects only the information its given. |  |
| **SELECT** \* **FROM table\_name WHERE** id **NOT IN** (7.8) | **-** | Selects everything except id 7 and Description |  |
| **SELECT** column\_name **FROM** table\_name **WHERE** ID **IN (SELECT** column\_name **FROM** table\_name **WHERE column\_name** = ‘String’ | **-** | Selects the id of the unknown id in another table. |  |
| **SELECT** \* **FROM** table\_name **ORDER BY** column\_name…; | **-** | Sorting rows In a choosen order. |  |

# FUNCTIONS :

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Example** |
| **CONCAT()** | Allows us to concatenating the string (combie two or more strings together) | SELECT CONCAT(‘Hello’, ‘World’) output : HelloWorld |
| **SUBSTRING()** | Allows us to extract letters from a string | SELECT SUBSTRING (‘String’, INT start, INT end); |
| **NOW()** | Record the present time. |  |
| **CURDATE()** | Record the current date. |  |
| **CURTIME()** | Record the current time |  |

## Aggregate Functions

**What is Aggregate functions?**

A function which performs calculations on a set

of values and return the result as a single value

|  |  |  |  |
| --- | --- | --- | --- |
| **Command** |  | **Description** | **Example** |
| **COUNT(VALUE)** | - | Returns numbered rows in the table. | **SELECT** **COUNT**(\* / column\_name) **FROM** table\_name |
| **AVG()** | - | AVGO() returns the average of a set of value | **SELECT** AVG(column\_name) **FROM** table\_name |
| **ROUND()** | - | Rounds up yo closest 10’er in decimal | **SELECT** ROUND(AVGO(column\_name, INT value)) **FROM** table\_name; |
| **MAX()** | - | Return the maximum set of values in a column | SELECT MAX(column\_name) FROM table\_name; |
| **MIN()** | - | Return the minimum set of values in a column | SELECT MIN(column\_name) FROM table\_name; |
| **SUM()** | - | Returns the sum of a value | SELECT SUM(column\_name) FROM table\_name; |

### Group By, Join & Unions

|  |  |
| --- | --- |
| **GROUP BY** | |
| Groups the data to compare data with a and b | **SELECT** ColumnName, ColumnName ( compared with), **FROM** TableName **GROUP BY** columnName |
| Groups the data to compare data with **A** and **B** | **SELECT** ColumnName, ColumnName ( compared with), **FROM** TableName **GROUP BY** columnName Having |

|  |  |
| --- | --- |
| **UNIONS** | |
| **UNION** used to comine the results of two or more **SELECT** statements each statement must have exact the same number of columns | **SELECT** statement\_one **UNION** SELECT statement\_two |
| Groups the data to compare data with **A** and **B** | **SELECT** statement\_one **FROM** table\_name **UNION ALL** **SELECT** statement\_two FROM table\_name |

# View

|  |  |  |
| --- | --- | --- |
| **Command** |  | **Description** |
| **CREATE VIEW** view\_name **AS SELECT** statement; | **-** | Creating a view, SQL virtual Table |
| **SELECT \* FROM** view\_name; | **-** | Viewing the virtual Table |
| **ALTER VIEW** view\_name **AS SELECT** statement; | **-** | Altering a view table |
| **DROP VIEW IF EXISTS** view\_name | **-** | Deletes a view |

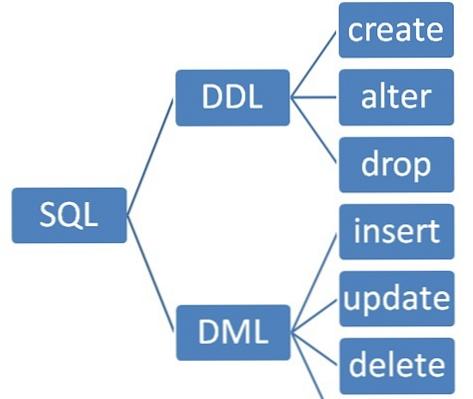
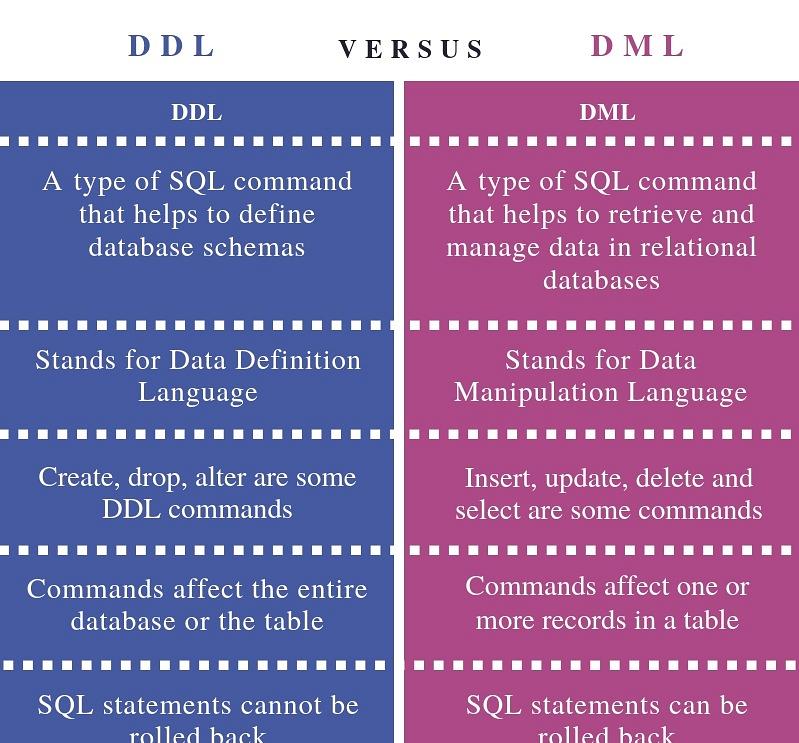
# Triggers

**What is a trigger ?**

Triggers is an automatic series of actions that is activated when a defined event occurs in a specific table such as **INSERT**, **UPDATE** or **DELETE**.

|  |  |  |
| --- | --- | --- |
| **Command** |  | **Description** |
| **DELIMITER** $$  **CREATE TRIGGER** trigger\_name **BEFORE** / **AFTER** / **UPDATE** / **DELETE** / **INSERT ON** table\_name **FOR EACH ROW BEGIN** – action **END** $$  **DELIMETER** ; | **-** | Creating a trigger to handle an event |
|  |  |  |

## DML / DDL



### Data Definition Language

A type of **SQL command** to **define** database schemas, by creating, drop or altering the database / table structure. The SQL Statements are permanent and are **not** able to be rolled back at.

### Data Manipulation Language

A type of **SQL command** which helps to **retrieve & Manage** data in relational databases by Inserting, updating, delete & selecting. Its basically commands that affects the **rows** in a table. The SQL-Statement is temporary and **can** be rolled back on.

# VARIABLE

A Name given to data which we need to store and use

|  |  |  |  |
| --- | --- | --- | --- |
| **Command** |  | **Description** | **Example** |
| **SET @variable\_name** = value | - | Assign the variable a value | **SELECT** \* **FROM** table\_name **WHERE** columnID = **@VariableName** |
| **SELECT @variable\_name** = value; | - | SET and SELECT a variable |  |

# Stored routines & Procedures

## Stored Procedures

|  |  |  |
| --- | --- | --- |
| **Command** |  | **Description** |
| **DELIMITER** $$  **CREATE PROCEDURE** procedureName ( [Parameters] ) **BEGIN** ---- SQL Statements **END** $$  **DELIMITER ;** | - | A Procedure syntax |
| **DELIMITER** $$  **CREATE PROCEDURE** ProcedureName( **IN** variabel, DATATYPE)  **BEGIN** ---- SQL Statements **END** $$  **DELIMITER ;** | **-** | **IN -**  Used to pass information to the procedure. |
| **DELIMITER** $$  **CREATE PROCEDURE** ProcedureName( **OUT** variabel, DATATYPE)  **BEGIN** ---- SQL Statements **END** $$  **DELIMITER ;** |  | **OUT:**  Used to get information from the stored procedure. |
| **DELIMITER** $$  **CREATE PROCEDURE** ProcedureName( **INOUT** variabel, DATATYPE)  **BEGIN** ---- SQL Statements **END** $$  **DELIMITER ;** |  | **INOUT:**   Serves both in and out parameters |

## STORED FUNCTIONS :

|  |  |  |
| --- | --- | --- |
| **COMMAND** |  | **Description** |
| **DELIMITER** $$  **CREATE FUNCTION** NameOfFunction ( parameters ) **RETURNS** datatype, Chartaristics of the function  **BEGIN** – details of the function **RETURN** result; **END** $$ | **-** | Create a function |
| **DECLEAR** variable dataType [ **DEFAULT** if value is default]; | **-** | Declears a local variable |