**SQL UNIQUE Constraint**

The UNIQUE constraint ensures that all values in a column are different.

Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint.

However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

Example :

create table student( id int **unique**, student\_name text);

insert into student (id,student\_name) values (1,"benedict")

insert into student (id,student\_name) values (1,"json") //error

## **SQL CHECK Constraint**

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a single column it allows only certain values for this column.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that you can not have any person below 18 years:

create table student (id int unique,name text, age int, check(age > 18))

**NOTE :**

**Unfortunately MySQL does not support SQL check constraints.**

**SQL DEFAULT Constraint**

The DEFAULT constraint is used to provide a default value for a column.

The default value will be added to all new records IF no other value is specified.

SQL DEFAULT on CREATE TABLE

The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:

**Example:**

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

City varchar(255) DEFAULT 'Sandnes'

);

**SQL CREATE VIEW Statement**

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

**CREATE VIEW Syntax**

CREATE VIEW view\_name AS

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

Example :

Creating view :

CREATE VIEW product\_information AS

SELECT productName, price

FROM products

WHERE price > 20;

Displaying view:

select \* from product\_information

**INDEX**

The CREATE INDEX statement is used to create indexes in tables.

Indexes are used to retrieve data from the database very fast. The users cannot see the indexes, they are just used to speed up searches/queries.

Syntax :

CREATE INDEX *index\_name*

ON *table\_name* (*column1*, *column2*, ...);

*Create table music (id primary key auto increment, band text, song text);*

//insert records

By default primary key column will have index. No need to create index for primary key column.

Searching records from table without indexing takes more time than with index.

Create index for column :

***Create index idx\_band on music (band);***

Now searching band from table music takes less time.

***Select band from music where band = “maroon”;***

**Bool datatype**

True and 1 are interchangeable

False and 0 are interchangeable

Create table student( id int, fees\_paid bool);

Insert into student(id ,fees\_paid) values (1,true); //stored fees\_paid value as 1

Insert into student(id ,fees\_paid) values (2,1);

Insert into student(id ,fees\_paid) values (3,false); //stored fees\_paid value as 0

Insert into student(id ,fees\_paid) values (1,0);

**Year Date Time DataTypes**

Format of Year : YYYY

Format of Date : MM

Format of Time : HH:MM:SS

To get current year : year(now())

To get current date : date(now())

To get current time : time(now())

Creating a table with datatypes year , date and time

***create table worldcup***

***(id int primary key auto\_increment, winning\_year year, winning\_date date, winning\_time time)***

Inserting date Example using functions:

*insert into worldcup*

*(winning\_year, winning\_date, winning\_time)*

*values*

*(year(now()),date(now()),time(now()));*

Inserting date Example without using functions:

*insert into worldcup*

*(winning\_year, winning\_date, winning\_time)*

*values*

*("2018","2018-06-06","12:06:55");*

Selecting a record based on timing

*select \* from worldcup where winning\_time < "12:06:55";*

**TimeStamp datatype**

The MySQL TIMESTAMP is a temporal data type that holds the combination of date and time. The format of a TIMESTAMP column is YYYY-MM-DD HH:MM:SS which is fixed at 19 characters.

Used to record every event.

**DATETIME datatype**

Same as TimeStamp. I.e, "YYYY-MM-DD HH:MM:SS" format.

**TIMESTAMP** differs with current timezone settings while **DATETIME** remains constant.

create table transaction\_

(id int primary key auto\_increment,starting\_ timestamp , ending\_ datetime default now());

insert into transaction\_ (starting\_) values ("2018-06-07 11:11:11");



**Locking tables**

Lock keyword allows us to lock tables. This will avoid other users to modify other table while a transaction(crud) is going on.

Once the table is unlocked then other users can access other tables.

Syntax :

Lock tables table1,table2….

//do your work with table1, table2 ….

Unlock tables

Ex :

lock tables products write

insert into products (productName, price) values ("Honey",137)

Unlock tables

Ex :

Lock tables products read

//read only products table, you cannot insert data into products table

Unlock tables