SQL

1.0

SQL

SQL is a standard language for accessing and manipulating databases.

- 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 records in a database
- SQL can update records in a database
- SQL can delete records from 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

Creating Database

The CREATE DATABASE statement is used to create a new SQL database.

Syntax

```
CREATE DATABASE databasename;
```

DROP DATABASE Statement

The DROP DATABASE statement is used to drop an existing SQL database.

Syntax

```
DROP DATABASE databasename;
```

CREATE TABLE Statement

The CREATE TABLE statement is used to create a new table in a database.

Syntax

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    ....
);
```

DROP TABLE Statement

The DROP TABLE statement is used to drop an existing table in a database.

Syntax

```
DROP TABLE table_name;
```

TRUNCATE TABLE Statement

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

Syntax

```
TRUNCATE TABLE table_name;
```

ALTER TABLE Statement

The ALTER TABLE statement is used

- to add, delete, or modify columns in an existing table.
- to add and drop various constraints on an existing table.

ALTER TABLE Statement

ALTER TABLE - ADD Column

To add a column in a table, use the following syntax:

Syntax

```
ALTER TABLE table_name

ADD column_name datatype;
```

ALTER TABLE Statement

ALTER TABLE - DROP COLUMN

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

Syntax

```
ALTER TABLE table_name

DROP COLUMN column_name;
```

ALTER TABLE Statement

ALTER TABLE - RENAME COLUMN

To rename a column in a table, use the following syntax:

Syntax

```
ALTER TABLE table_name

RENAME COLUMN old_name to new_name;
```

ALTER TABLE Statement

ALTER TABLE - ALTER/MODIFY DATATYPE

To change the data type of a column in a table, use the following syntax:

Syntax

```
ALTER TABLE table_name

MODIFY COLUMN column_name datatype;
```

SQL Constraints

The following constraints are commonly used in SQL:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK
- DEFAULT

SQL Create Constraints

Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

Syntax

```
CREATE TABLE table_name (
    column1 datatype constraint,
    column2 datatype constraint,
     ....
);
```

SQL NOT NULL Constraint

By default, a column can hold NULL values.

The NOT NULL constraint enforces a column to NOT accept NULL values.

This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

SQL NOT NULL on CREATE TABLE

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(25) NOT NULL,
    FirstName varchar(25) NOT NULL,
    Age int
);
```

SQL NOT NULL on ALTER TABLE

```
ALTER TABLE Persons
MODIFY COLUMN Age int NOT NULL;
```

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.

SQL UNIQUE Constraint on CREATE TABLE

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(25) NOT NULL,
    FirstName varchar(25),
    Age int UNIQUE
);
```

SQL UNIQUE Constraint on ALTER TABLE

```
ALTER TABLE Persons
ADD UNIQUE (ID);
```

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

SQL PRIMARY KEY Constraint

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(25) NOT NULL,
    FirstName varchar(25),
    Age int,
    PRIMARY KEY (ID)
);
```

SQL PRIMARY KEY on ALTER TABLE

```
ALTER TABLE Persons

ADD PRIMARY KEY (ID);
```

SQL FOREIGN KEY Constraint

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

SQL FOREIGN KEY Constraint

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(ID)
);
```

SQL FOREIGN KEY on ALTER TABLE

```
ALTER TABLE Orders
ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);
```

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 column it will allow 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.

SQL CHECK on CREATE TABLE

```
CREATE TABLE pets(
    ID INT PRIMARY KEY,
    Name VARCHAR(30) NOT NULL,
    Breed VARCHAR(20) NOT NULL,
    Age INT,
    GENDER VARCHAR(9),
    check(GENDER in ('Male', 'Female', 'Unknown'))
    );
```

SQL CHECK on ALTER TABLE

```
ALTER TABLE Persons

ADD CHECK (Age>=18);


ALTER TABLE Persons

ADD CHECK (Age>=18 AND Age<=60);
```

SQL DEFAULT Constraint

The DEFAULT constraint is used to set 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

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(25) NOT NULL,
    FirstName varchar(25),
    Age int,
    City varchar(25) DEFAULT 'Mangaluru'
);
```

SQL DEFAULT on ALTER TABLE

```
ALTER TABLE Persons

ALTER City SET DEFAULT 'Moodabidri';
```

DROP a DEFAULT Constraint

```
ALTER TABLE Persons

ALTER City DROP DEFAULT;
```

SQL AUTO INCREMENT

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

SQL AUTO INCREMENT

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
);
```

SQL AUTO INCREMENT

By default, the starting value for AUTO_INCREMENT is 1, and it will increment by 1 for each new record.

To let the AUTO_INCREMENT sequence start with another value, use the following SQL statement:

```
ALTER TABLE Persons AUTO_INCREMENT=100;
```

SQL Working With Dates

MySQL comes with the following data types for storing a date or a date/time value in the database:

DATE - format YYYY-MM-DD

DATETIME - format: YYYY-MM-DD HH:MI:SS

TIMESTAMP - format: YYYY-MM-DD HH:MI:SS

YEAR - format YYYY or YY

SQL Working With Dates

```
Create table dobyear(name varchar(10),dob year);
```

```
Create table dob(name varchar(10),dob date);
```

SQL Views

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 statements and functions to a view and present the data as if the data were coming from one single table.

SQL CREATE VIEW Statement

CREATE VIEW Syntax

```
CREATE VIEW view_name AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;
```

SQL CREATE VIEW Statement

```
CREATE VIEW DOBirth as

SELECT * FROM dobyear WHERE dob='1990';
```

SQL Updating a View

A view can be updated with the CREATE OR REPLACE VIEW statement.

```
CREATE OR REPLACE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

SQL Updating a View

```
CREATE or REPLACE VIEW DOBirth as

SELECT Name FROM dobyear WHERE dob='1990';
```

SQL Dropping a View

```
DROP VIEW view_name;
```