Sql Statements

**DDL - Data Definition Language**

These statements directly effect the table structure.

Once these statements are executed you cannot rollback

Create, alter, drop, rename, truncate, comment.

**DML - Data Manipulation Language.**

Used to add or manipulate existing data in the table.

You can do rollback for DML statements.

Select, insert, update, delete, merge

**DCL - Data Control Language.**

Used to give permissions based on the user role

Grant, revoke

**TCL - Transaction Control Language**

Used to control transactions.

Commit, Rollback, Savepoint

**The SQL SELECT Statement**

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

SELECT Syntax

SELECT column1, column2, ...  
FROM table\_name;

## The SQL SELECT DISTINCT Statement

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT Country FROM Customers;

## SQL WHERE Clause

The WHERE clause is used to filter records.

The WHERE clause is used to extract only those records that fulfill a specified condition.

SELECT \* FROM Customers  
WHERE Country='Mexico';

**The SQL AND, OR and NOT Operators**

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

* The AND operator displays a record if all the conditions separated by AND is TRUE.
* The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

SELECT \* FROM Customers  
WHERE Country='Germany' AND City='Berlin';

SELECT \* FROM Customers  
WHERE City='Berlin' OR City='München';

SELECT \* FROM Customers  
WHERE NOT Country='Germany';

[Try it Yourself »](https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_where_and)

## The SQL CREATE DATABASE Statement

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

CREATE DATABASE databasename;

## The SQL DROP DATABASE Statement

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

DROP DATABASE databasename;

## The SQL CREATE TABLE Statement

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

CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)   
);

**Data Types**

Data types define what type of data a column can contain

It is a label and guideline for sql to understand what type data is expected inside of each column and also it identifies how sql will interact with stored data.

https://www.tutorialspoint.com/mysql/mysql-data-types.htm

varchar - differentiates between null and empty string

varchar2 - cannot differentiate between null and empty string

Constraints

Primary key

Foreign key

Unique

Not null

Default

Create database

Create database databasename;

Use databasename;

Create

———-

CREATE TABLE table\_name (

column1 datatype,

column2 datatype,

column3 datatype,

....

);

CREATE TABLE Persons (

PersonID int,

LastName varchar(255),

FirstName varchar(255),

Address varchar(255),

City varchar(255)

);

**Create table using another table**

A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement.

The new table gets the same column definitions. All columns or specific columns can be selected.

If you create a new table using an existing table, the new table will be filled with the existing values from the old table.

CREATE TABLE new\_table\_name AS

SELECT column1, column2,...

FROM existing\_table\_name

WHERE ....;

**Constraints while creating table**

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.

CREATE TABLE table\_name (

column1 datatype constraint,

column2 datatype constraint,

column3 datatype constraint,

....

);

**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** - Uniquely identifies a row/record in another table

**CHECK** - Ensures that all values in a column satisfies a specific condition

**DEFAULT** - Sets a default value for a column when no value is specified

**INDEX** - Use to create and retrieve data from the database very quickly

**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.

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255) NOT NULL,

Age int

);

**Unique key 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.

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

UNIQUE (ID)

);

For multiple columns

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

CONSTRAINT UC\_Person UNIQUE (ID,LastName)

);

**Primary key constraint**

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

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

A table can have only one primary key, which may consist of single or multiple fields.

CREATE TABLE Persons (

IDint NOT NULL ,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

PRIMARY KEY (ID)

);

For multiple columns

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName)

);

**Foreign key constraint**

A FOREIGN KEY is a key used to link two tables together.

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

The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

CREATE TABLE Orders (

OrderID int NOT NULL,

OrderNumber int NOT NULL,

PersonID int,

PRIMARY KEY (OrderID),

FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)

);

For multiple columns

CREATE TABLE Orders (

OrderID int NOT NULL,

OrderNumber int NOT NULL,

PersonID int,

PRIMARY KEY (OrderID),

CONSTRAINT FK\_PersonOrder FOREIGN KEY (PersonID)

REFERENCES Persons(PersonID)

);

**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.

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

CHECK (Age>=18)

);

For multiple columns

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

City varchar(255),

CONSTRAINT CHK\_Person CHECK (Age>=18 AND City='Sandnes')

);

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.

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

City varchar(255) DEFAULT 'Sandnes'

);

The DEFAULT constraint can also be used to insert system values, by using functions like GETDATE():

CREATE TABLE Orders (

ID int NOT NULL,

OrderNumber int NOT NULL,

OrderDate date DEFAULT GETDATE()

);

**Auto increment field**

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.

CREATE TABLE Persons (

ID int NOT NULL AUTO\_INCREMENT,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

PRIMARY KEY (ID)

);

MySQL uses the AUTO\_INCREMENT keyword to perform an auto-increment feature.

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;

**To get structure of table**

Describe table name;

**Insert into statement**

INSERT INTO table\_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

INSERT INTO table\_name

VALUES (value1, value2, value3, ...);

Insert into select statement

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

INSERT INTO SELECT requires that data types in source and target tables match

The existing records in the target table are unaffected

INSERT INTO table2

SELECT \* FROM table1

WHERE condition;

INSERT INTO table2 (column1, column2, column3, ...)

SELECT column1, column2, column3, ...

FROM table1

WHERE condition;

**Alter table statement**

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

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/modify column

ALTER TABLE table\_name

MODIFY COLUMN column\_name datatype;

Alter table unique constraint

ALTER TABLE Persons

ADD UNIQUE (ID);

ALTER TABLE Persons

ADD CONSTRAINT UC\_Person UNIQUE (ID,LastName);

Drop unique constraint

ALTER TABLE Persons

DROP INDEX UC\_Person;

Alter table primary key constraint

ALTER TABLE Persons

ADD PRIMARY KEY (ID);

ALTER TABLE Persons

ADD CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName);

Alter table drop primary key

ALTER TABLE Persons

DROP PRIMARY KEY;

Alter table foreign key constraint

ALTER TABLE Orders

ADD FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

ALTER TABLE Orders

ADD CONSTRAINT FK\_PersonOrder

FOREIGN KEY (PersonID) REFERENCES Persons(PersonID);

Alter table drop foreign key constraint

ALTER TABLE Orders

DROP FOREIGN KEY FK\_PersonOrder;

Alter table check constraint

ALTER TABLE Persons

ADD CHECK (Age>=18);

Alter table drop check constraint

ALTER TABLE Persons

DROP CHECK CHK\_PersonAge;

Alter table default constraint

ALTER TABLE Persons

ALTER City SET DEFAULT 'Sandnes';

Alter table drop default constraint

ALTER TABLE Persons

ALTER City DROP DEFAULT;

**Drop, delete and truncate**

Drop and truncate

are DDL statements

We cannot rollback

Delete

is DML statement

We can rollback

Drop deletes table and its structure

Truncate deletes data but table structure will still be there

Delete will delete data but not table structure.you can limit or filter rows by using where clause. it also invokes all the triggers.

Drop and truncate deletes all data. it does not invoke the triggers, it ignores the triggers.

If you want to delete entire data go with truncate because delete is slow in terms of performance. it will take backup or cache the data because once u rollback it has to get back the deleted data.

Delete statement

DELETE FROM table\_name;

DELETE \* FROM table\_name;

DELETE FROM table\_name

WHERE condition;

Rollback;

Truncate statement

Truncate table table name;

Drop table table name;

Update statement

UPDATE table\_name

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

WHERE condition;

You will get update warning when you don’t use where clause because it updates all records.

UPDATE Customers

SET ContactName='Juan';

You can do arithmetic operations

If you want to increment salary of employees.

Select employee\_id, first\_name, salary+10 as increment\_sal from employees;

Select employee\_id, first\_name, (salary+10)\*2 as increment\_sal from employees;

Get all employees who are working as IT\_PROG.

Select \* from employees where job\_id=“IT\_PROG”;

Get all employees who joined after year 1998

Select \* from employees where hire\_date>’1998-12-31’;

Get all employees whose salary is greater than 10000

Select \* from employees where salary>10000;

Get all employees who have commission pct between 0.4 and 5

Select \* from employees where commission\_pct between 0.5 and 5;

Get all employees who are it prog and whose salary is greater than 5000

Select \* from employees where job\_id=“IT\_PROG” and salary>5000;

Get all employees who are not it prog

Select \* from employees where job\_id!=“IT\_PROG”;

Get all employees who are it prog, hr and clerk

Select \* from employees where job\_id in (“IT\_PROG”, “hr”, “clerk”);

Get full name of all employees

Select concat(first\_name, ‘ ‘, last\_name) as full name from employees;

Get all unique last names

Select distinct last\_name from employees;

Get all employees whose first name starts with ‘a’ and ends with ’n’

Select \* from employees where first\_name like “a%n”;

% - any number of characters

\_ - for one character