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.

One 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;

This is the generic syntax of create view.

A view always shows up-to-date data! The database engine recreates the data, using the view's SQL statement, every time a user queries a view.

## SQL CREATE VIEW Examples:

The first example is interesting. Suppose, a table named ‘Products’ exists. That table has a flag variable indicating whether the product is currently available or not.

Suppose, the table Products is created like this:

**create table Products(ProductID INT NOT NULL AUTO\_INCREMENT PRIMARY KEY, ProductName VARCHAR(100) NOT NULL, CurrentlyAvailable enum("YES","NO"));**

And show columns from Products looks like:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **Null** | **Key** | **Default** | **Extra** |
| ProductID | INT(11) | NO | PRI | NULL | AUTO\_  INCREMENT |
| Product  Name | VARCHAR(100) | No |  | NULL |  |
| CurrentlyAvailable | enum('YES','NO') | YES |  | NULL |  |

Now, some data is inserted in the table:

**insert into Products(ProductID, ProductName, CurrentlyAvailable) Values(1, "Redmi 4a", 'NO');**

**insert into Products(ProductID, ProductName, CurrentlyAvailable) Values(2, "Redmi 3S", 'NO');**

**insert into Products(ProductID, ProductName, CurrentlyAvailable) Values(3, "Moto E", 'Yes');**

**insert into Products(ProductID, ProductName, CurrentlyAvailable) Values(4, "Moto G4", 'Yes');**

Now, a view is created:

**View Creation Syntax:**

**create View Available\_Product\_List As Select ProductID, ProductName From Products Where CurrentlyAvailable='Yes';**

Now, it enables every kind of queries in it. Including Insertion, deletion, Update.

**Replace View Syntax:**

**Generic Syntax:**

REPLACE VIEW view\_name AS SELECT column1, column2, … FROM table\_name  
WHERE condition;

**Drop View Syntax:**

**Genetic Syntax:**

Drop view view\_name;

However, this is basics.

**Generic View Creation Syntax:**

**CREATE**

**[OR REPLACE]**

**[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]**

**[DEFINER = { user | CURRENT\_USER }]**

**[SQL SECURITY { DEFINER | INVOKER }]**

**VIEW view\_name [(column\_list)]**

**AS select\_statement**

**[WITH [CASCADED | LOCAL] CHECK OPTION]**

However, View does not exactly behaves like a table:

**A view definition is subject to the following restrictions:**

* Before MySQL 5.7.7, the [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html" \o "13.2.9 SELECT Syntax) statement cannot contain a sub-query in the FROM clause.
* The [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html" \o "13.2.9 SELECT Syntax) statement cannot refer to system variables or user-defined variables.
* Within a stored program, the [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html" \o "13.2.9 SELECT Syntax) statement cannot refer to program parameters or local variables.
* The [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html" \o "13.2.9 SELECT Syntax) statement cannot refer to prepared statement parameters.
* Any table or view referred to in the definition must exist. If, after the view has been created, a table or view that the definition refers to is dropped, use of the view results in an error. To check a view definition for problems of this kind, use the [CHECK TABLE](https://dev.mysql.com/doc/refman/5.7/en/check-table.html" \o "13.7.2.2 CHECK TABLE Syntax) statement.
* The definition cannot refer to a TEMPORARY table, and you cannot create a TEMPORARY view.
* You cannot associate a trigger with a view.
* Aliases for column names in the [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html" \o "13.2.9 SELECT Syntax) statement are checked against the maximum column length of 64 characters (not the maximum alias length of 256 characters).
* ORDER BY is permitted in a view definition, but it is ignored if you select from a view using a statement that has its own ORDER BY.

(These are from mysql reference page).