# MySQL Views: How to Create View from Tables with Examples

## What are Views in MySQL?

**VIEWS** are virtual tables that do not store any data of their own but display data stored in other tables. In other words, VIEWS are nothing but SQL Queries. A view can contain all or a few rows from a table. A MySQL view can show data from one table or many tables.

### MySQL Views syntax

Let's now look at the basic syntax used to create a view in MySQL.

```
CREATE VIEW `view_name` AS SELECT statement;
```

**WHERE**

- **"CREATE VIEW `view_name`"** tells MySQL server to create a view object in the database named `view_name`
- **"AS SELECT statement"** is the SQL statement to be packed in the MySQL Views. It can be a SELECT statement that can contain data from one table or multiple tables.

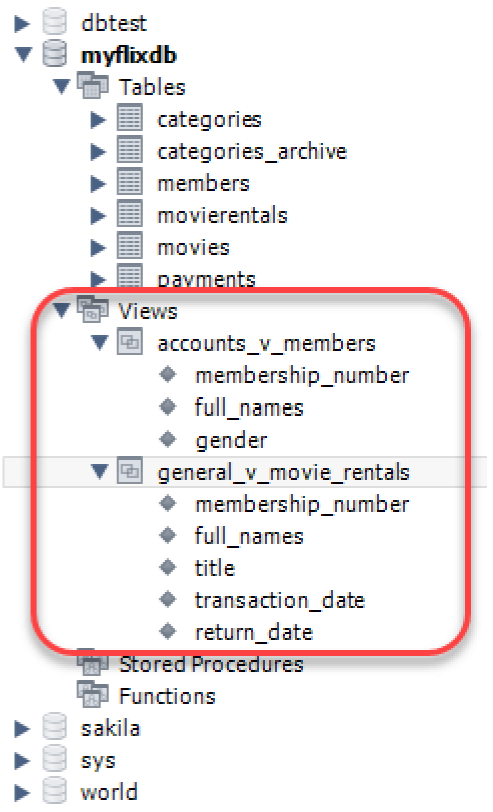
## How to Create Views in MySQL

Following is a step by step process to create view in MySQL:

Let's now create our first view using the "myflixdb" . We will create a simple view that restricts the columns seen in the members table.

Suppose authorization requirements state that the accounts department can only see member's number , name and gender from the member's table. To achieve this you can create a VIEW -

```
CREATE VIEW `accounts_v_members` AS SELECT
`membership_number`,`full_names`,`gender` FROM `members`;
```
Executing the above script in MySQL workbench against the myflixdb and expanding the views node in the database explorer gives us the following results.



Note the accounts_v_members object is now visible in the database views objects. Let's now execute a SELECT statement that selects all the fields from the view as shown in the below MySQL create view example.

```
SELECT * FROM `accounts_v_members`;
```
Executing the above script in MySQL workbench against myflixdb gives us the following results shown below.

| **membership_number** | **full_names** | **gender** |
|---|---|---|
| 1 | Janet Jones | Female |

| | | |
|---|---|---|
| 2 | Janet Smith Jones | Female |
| 3 | Robert Phil | Male |
| 4 | Gloria Williams | Female |
| 5 | Leonard Hofstadter | Male |
| 6 | Sheldon Cooper | Male |
| 7 | Rajesh Koothrappali | Male |
| 8 | Leslie Winkle | Male |
| 9 | Howard Wolowitz | Male |

Only the authorized columns for the accounts department have been returned. Other details found in the members table have been hidden .

If we want to see the SQL statements that make up a particular view, we can use the script shown below to do that.

```
SHOW CREATE VIEW `accounts_v_members`;
```

Executing the above script gives you the view name and the SQL SELECT statements used to create the view.

## Joins and Views in MySQL

Let's now look at a fairly complex example which involves multiple tables and uses joins.

We will package the JOIN created that gets information from three (3) tables namely members, movies and movie rentals. Below is the script that helps us to achieve that.

```
CREATE VIEW `general_v_movie_rentals` AS SELECT
mb.`membership_number`,mb.`full_names`,mo.`title`,mr.`tra
```

```
nsaction_date`,mr.`return_date` FROM `movierentals` AS mr
INNER JOIN `members` AS mb ON mr.`membership_number` =
mb.`membership_number` INNER JOIN `movies` AS mo ON
mr.`movie_id` = mo.`movie_id`;
```

Executing the above scripts creates the view named general_v_movie_rentals in our myflixdb

Let's now select all the fields from a table named general_v_movie_rentals.

```
SELECT * FROM `general_v_movie_rentals`;
```

Executing the above script in MySQL workbench against the myflixdb gives us the following results shown below.

| **membership_number** | **full_names** | **title** | **transaction_date** | **return_date** |
|---|---|---|---|---|
| 1 | Janet Jones | Pirates of the Caribbean 4 | 20-06-2012 | 28-06-2012 |
| 1 | Janet Jones | Forgetting Sarah Marshall | 22-06-2012 | 25-06-2012 |
| 3 | Robert Phil | Forgetting Sarah Marshall | 22-06-2012 | 25-06-2012 |
| 2 | Janet Smith Jones | Forgetting Sarah Marshall | 21-06-2012 | 24-06-2012 |
| 3 | Robert Phil | X-Men | 23-06-2012 | 28-06-2012 |

Note we didn't have to write the complex JOIN query to get information about members, movies and movie rental details. We simply used the view in a regular SELECT statement as any other ordinary table. The view can be called from anywhere in the application system running on top of the myflixdb.

## Dropping Views in MySQL

The DROP command can be used to delete a view from the database that is no longer required. The basic syntax to drop a view is as follows.

```
DROP VIEW ` general_v_movie_rentals `;
```

Why use views?

You may want to use views primarily for following 3 reasons

- Ultimately , you will use your SQL knowledge to create applications , which will use a  database for data requirements. It's recommended that you use VIEWS of the original table structure in your application instead of using the tables themselves. This ensures that when you refactor your DB, your legacy code will see the original schema via the view without breaking the application.
- VIEWS increase reusability. You will not have to create complex queries involving joins repeatedly. All the complexity is converted into a single line of query use VIEWS. Such condensed code will be easier to integrate in your application. This will eliminate chances of typos and your code will be more readable.
- VIEWS help in data security. You can use views to show only authorized information to users and hide sensitive data like credit card numbers.

## Summary

- Views are virtual tables; they do not contain the data that is returned. The data is stored in the tables referenced in the SELECT statement.
- Views improve security of the database by showing only intended data to authorized users. They hide sensitive data.
- Views make life easy as you do not have to write complex queries time and again.
- It's possible to use INSERT, UPDATE  and DELETE on a VIEW. These operations will change the underlying tables of the VIEW.  The only consideration is that VIEW should contain all NOT NULL columns of the tables it references. Ideally, you should not use VIEWS for updating