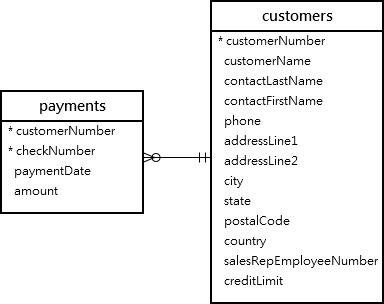
# **MySQL Views**

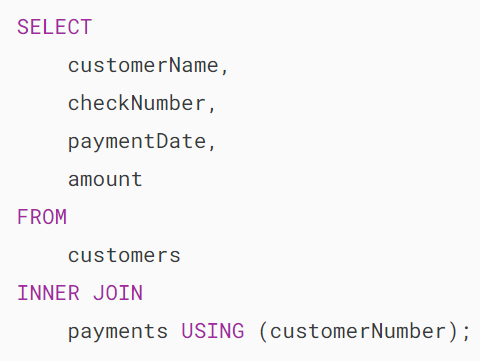
**Summary**: in this tutorial, you will learn about MySQL views and how to manipulate views effectively.

## **Introduction to MySQL Views**

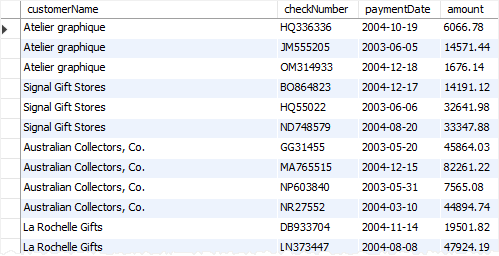
Let’s see the following tables customers and payments from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/).



This [query](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) returns data from both tables customers and payments using the [inner join](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/):



Here is the output:



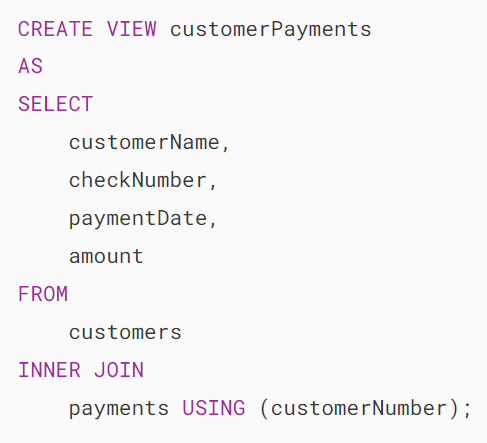
Next time, if you want to get the same information including customer name, check number, payment date, and amount, you need to issue the same query again.

One way to do this is to save the query in a file, either .txt or .sql file so that later you can open and execute it from MySQL Workbench or any other MySQL client tools.

A better way to do this is to save the query in the database server and assign a name to it. This named query is called a **database view,** or simply, **view**.

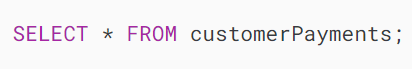
By definition, a view is a named query stored in the database catalog.

To create a new view you use the [CREATE VIEW](https://www.mysqltutorial.org/mysql-views/mysql-create-view/) statement. This statement creates a view customerPayments based on the above query above:



After you execute the CREATE VIEW statement, MySQL creates the view and stores it in the database.

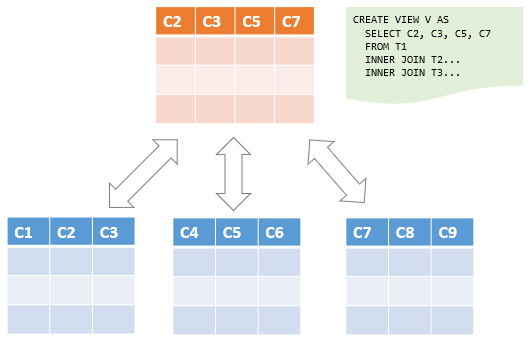
Now, you can reference the view as a table in SQL statements. For example, you can query data from the customerPayments view using the SELECT statement:



As you can see, the syntax is much simpler.

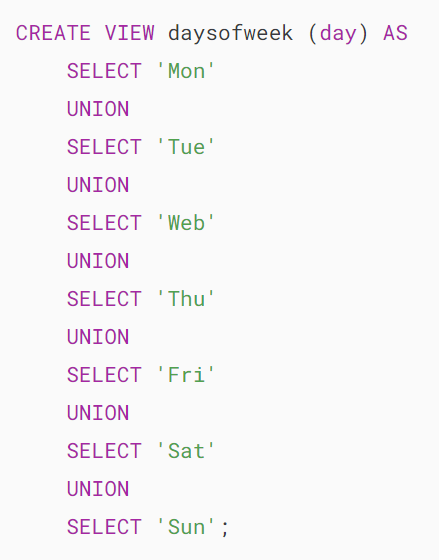
Note that a view does not physically store the data. When you issue the [SELECT](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) statement against the view, MySQL executes the underlying query specified in the view’s definition and returns the result set. For this reason, sometimes, a view is referred to as a virtual table.

MySQL allows you to create a view based on a SELECT statement that retrieves data from one or more tables. This picture illustrates a view based on columns of multiple tables:

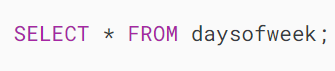


In addition, MySQL even allows you to create a view that does not refer to any table. But you will rarely find this kind of view in practice.

For example, you can create a view called daysofweek that return 7 days a week by executing the following query:



You can query data from the daysofweek view as follows:



This picture shows the output:



## **Advantages of MySQL Views**

MySQL views bring the following advantages.

### **1) Simplify complex query**

Views help simplify complex queries. If you have any frequently used complex query, you can create a view based on it so that you can reference the view by using a simple SELECT statement instead of typing the query all over again.

### **2) Make the business logic consistent**

Suppose you have to repeatedly write the same formula in every query.  Or you have a query that has complex business logic. To make this logic consistent across queries, you can use a view to store the calculation and hide the complexity.

### **3) Add extra security layers**

A table may expose a lot of data including sensitive data such as personal and banking information.

By using views and privileges, you can limit which data users can access by exposing only the necessary data to them.

For example, the table employees may contain SSN and address information, which should be accessible by the HR department only.

To expose general information such as first name, last name, and gender to the General Administration (GA) department, you can create a view based on these columns and grant the users of the GA department the view, not the entire table employees .

### **4) Enable backward compatibility**

In legacy systems, views can enable backward compatibility.

Suppose, you want to normalize a big table into many smaller ones. And you don’t want to impact the current applications that reference the table.

In this case, you can create a view whose name is the same as the table based on the new tables so that all applications can reference the view as if it were a table.

Note that a view and table cannot have the same name so you need to [drop the table](https://www.mysqltutorial.org/mysql-drop-table) first before creating a view whose name is the same as the deleted table.

## **Managing views in MySQL**

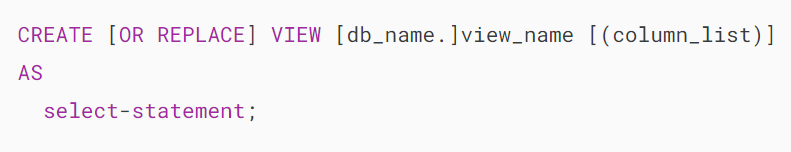
* [Create views](https://www.mysqltutorial.org/mysql-views/mysql-create-view/) – show you how to use the CREATE VIEW statement to create a new view in the database.
* [Understand view processing algorithms](https://www.mysqltutorial.org/mysql-views/mysql-view-processing-algorithms/) – learn how MySQL processes a view.
* [Create updatable views](https://www.mysqltutorial.org/mysql-views/mysql-updatable-views/) – learn how to create updatable views.
* Create views with a [WITH CHECK OPTION](https://www.mysqltutorial.org/mysql-views/mysql-view-with-check-option/) – ensure the consistency of views using the WITH CHECK OPTION clause.
* [LOCAL & CASCADED](https://www.mysqltutorial.org/mysql-view-local-cascaded-in-with-check-option) and WITH CHECK OPTION – specify the scope of the check with LOCAL and CASCADED options.
* [Show views](https://www.mysqltutorial.org/mysql-views/mysql-show-view/) – provide ways to find views in a database.
* [Show create view](https://www.mysqltutorial.org/mysql-views/mysql-show-create-view/) – learn how to display the statement that creates a view.
* [Rename views](https://www.mysqltutorial.org/mysql-views/mysql-rename-view/) – change the name of a view to another.
* [Drop views](https://www.mysqltutorial.org/mysql-views/mysql-drop-view/) – guide you on how to remove one or more existing views.

# **MySQL CREATE VIEW**

**Summary**: in this tutorial, you will learn how to use the MySQL CREATE VIEW  statement to create a new view in the database.

## **Introduction to MySQL CREATE VIEW statement**

The CREATE VIEW statement creates a new view in the database. Here is the basic syntax of the CREATE VIEW statement:



In this syntax:

First, specify the name of the view that you want to create after the CREATE VIEW keywords. The name of the view is unique in a database. Because views and tables in the same database share the same namespace, the name a view cannot the same as the name of an existing table.

Second, use the OR REPLACE option if you want to replace an existing view if the view already exists. If the view does not exist, the OR REPLACE has no effect.

Third, specify a list of columns for the view. By default, the columns of the view are derived from the select list of the SELECT statement. However, you can explicitly specify the column list for the view by listing them in parentheses following the view name.

Finally,  specify a [SELECT](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) statement that defines the view. The SELECT  statement can query data from tables or views. MySQL allows you to use the [ORDER BY](https://www.mysqltutorial.org/mysql-basics/mysql-order-by/) clause in the SELECT statement but ignores it if you select from the view with a query that has its own ORDER BY clause.

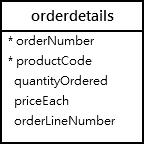
By default, the CREATE VIEW statement creates a view in the current database. If you want to explicitly create a view in a given database, you can qualify the view name with the database name.

## **MySQL CREATE VIEW examples**

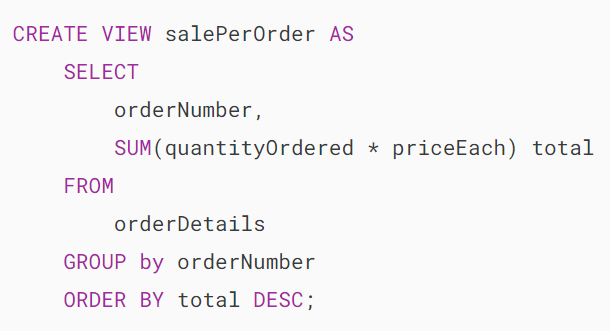
Let’s take some example of using the CREATE VIEW statement to create new views.

### **1) Creating a simple view example**

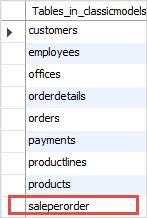
Let’s take a look at the orderDetails table from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/):



This statement uses the CREATE VIEW statement to create a view that represents total sales per order.

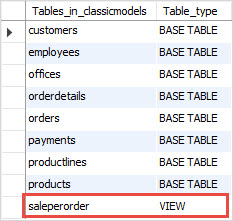
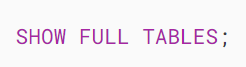


If you use the SHOW TABLE command to view all tables in the classicmodels database, you will see the viewsalesPerOrder is showing up in the list.



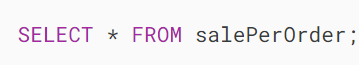
This is because the views and tables share the same namespace as mentioned earlier.

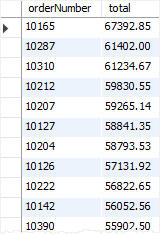
To know which object is a view or table, you use the [SHOW FULL TABLES](https://www.mysqltutorial.org/mysql-administration/mysql-show-tables/) command as follows:



The table\_type column in the result set specifies the type of the object: view or table (base table).

If you want to query total sales for each sales order, you just need to execute a simple SELECT  statement against the SalePerOrder  view as follows:

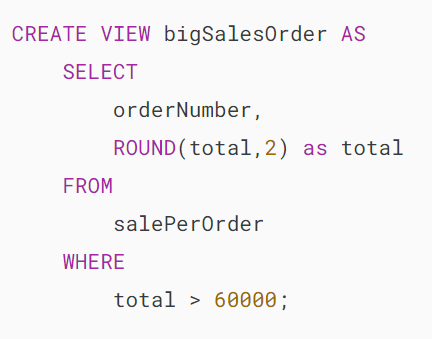




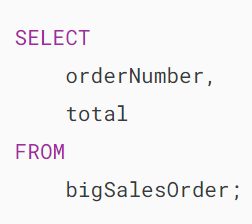
### **2) Creating a view based on another view example**

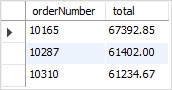
MySQL allows you to create a view based on another view.

For example, you can create a view called bigSalesOrder based on the salesPerOrder view to show every sales order whose total is greater than 60,000 as follows:



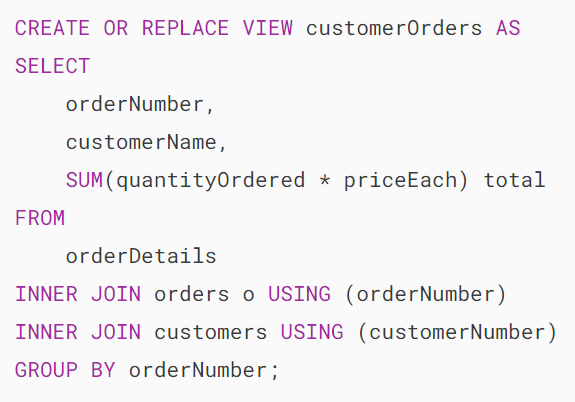
Now, you can query the data from the bigSalesOrder view as follows:





### **3) Creating a view with join example**

The following example uses the CREATE VIEW statement to create a view based on multiple tables. It uses the [INNER JOIN](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/) clauses to join tables.



This statement selects data from the customerOrders view:



This picture shows the partial output:

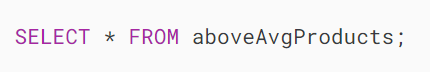


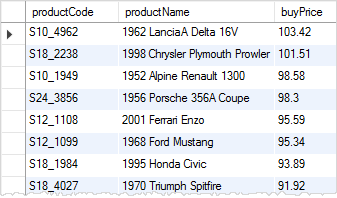
### **4) Creating a view with a subquery example**

The following example uses the CREATE VIEW statement to create a view whose SELECT statement uses a [subquery](https://www.mysqltutorial.org/mysql-basics/mysql-subquery/). The view contains products whose buy prices are higher than the average price of all products.



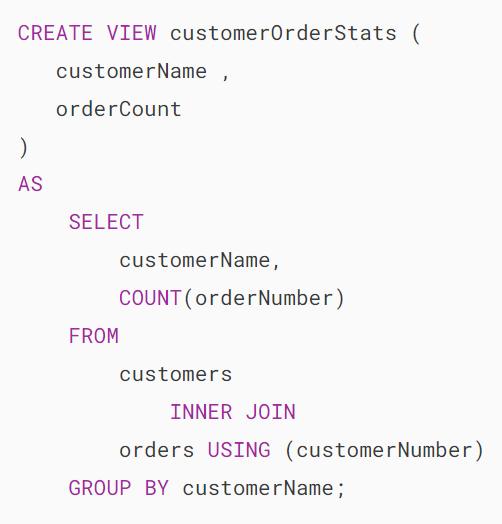
This query data from the aboveAvgProducts is simple as follows:



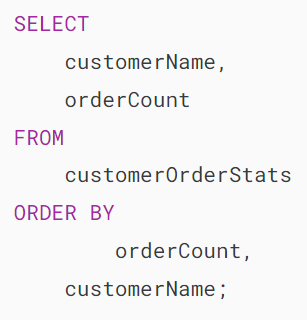


### **5) Creating a view with explicit view columns example**

This statement uses the CREATE VIEW statement to create a new view based on the customers and orders tables with explicit view columns:



This query returns data from the customerOrderStats view:



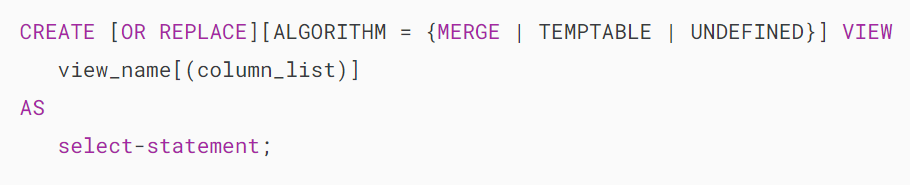


# **MySQL View Processing Algorithms**

**Summary**: in this tutorial, you will learn about MySQL view processing algorithms including MERGE, TEMPTABLE, and UNDEFINED.

The [CREATE VIEW](https://www.mysqltutorial.org/mysql-views/mysql-create-view/) and ALTER VIEW statements have an optional clause: ALGORITHM. The algorithm determines how MySQL process a view and can take one of three values MERGE, TEMPTABLE, and UNDEFINE.

Here is the CREATE VIEW statement with the ALGORITHM clause:



And this is the ALTER VIEW statement with the ALGORITHM clause:

## 

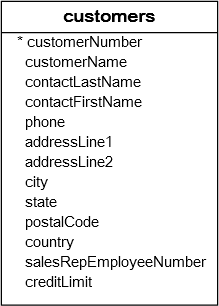
## **MERGE**

When you query from a MERGE view, MySQL processes the following steps:

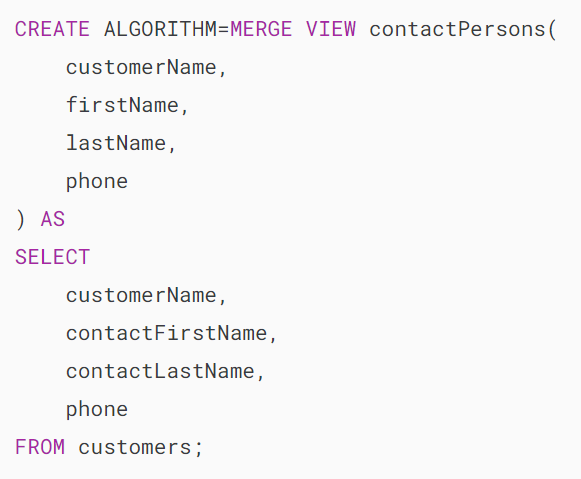
* First, merge the input query with the [SELECT](http://www.www.mysqltutorial.org/mysql-basics/mysql-select-from/) statement in the view definition into a single query.
* Then, execute the combined query to return the result set.

Note that the combination of input query and the SELECT statement of the view definition into a single query is referred to as view resolution.

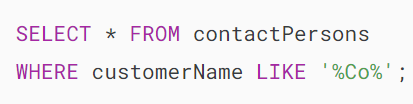
See the following customers from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/):



The following statement creates a view based on the customers table with the name contactPersons with the MERGE algorithm:



Suppose that you issue the following statement:



MySQL performs these steps:

* Convert view name contactPersons to table name customers.
* Convert askterisk (\*)  to a list column names customerName, firstName, lastName, phone, which corresponds to customerName, contactFirstName, contactLastName, phone.
* Add the [WHERE](https://www.mysqltutorial.org/mysql-basics/mysql-where/) clause.

The resulting statement is:

## 

## **TEMPTABLE**

When you issue a query to a TEMPTABLE view, MySQL performs these steps:

* First, [create a temporary table](https://www.mysqltutorial.org/mysql-basics/mysql-temporary-table/) to store the result of the SELECT in the view definition.
* Then, execute the input query against the temporary table.

Because MySQL has to create the temporary table to store the result set and moves the data from the base tables to the temporary table, the algorithm TEMPTABLE  is less efficient than the MERGE algorithm.

Note that TEMPTABLE views cannot be [updatable](https://www.mysqltutorial.org/mysql-views/mysql-updatable-views/).

## **UNDEFINED**

The UNDEFINED is the default algorithm when you create a view without specifying the ALGORITHM clause or you explicitly specify ALGORITHM=UNDEFINED.

In addition, when you create a view with ALGORITHM = MERGE and MySQL can only process the view with a temporary table, MySQL automatically sets the algorithm to UNDEFINED and generates a warning.

TheUNDEFINED allows MySQL to choose either MERGE or TEMPTABLE. And MySQL prefers MERGE  over TEMPTABLE  if possible because MERGE is often more efficient than TEMPTABLE.

In this tutorial, you have learned about the MySQL view processing algorithms including MERGE, TEMPTABLE, and UNDEFINED.

# **MySQL ALTER VIEW**

**Summary**: in this tutorial, you will learn how to use the MySQL ALTER VIEW to modify the definition of an existing view.

## **Introduction to MySQL ALTER VIEW statement**

The MySQL ALTER VIEW statement changes the definition of an existing view. The syntax of the ALTER VIEW is similar to the CREATE VIEW statement:

## 

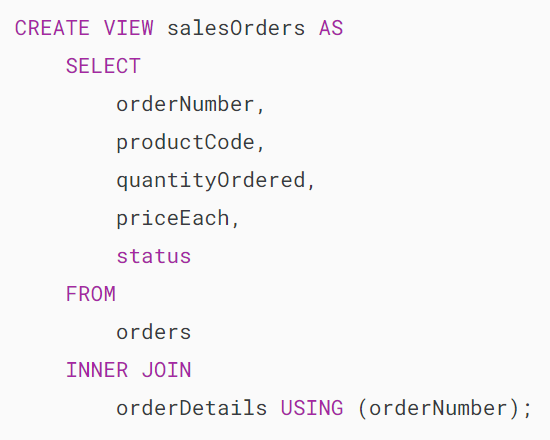
## **MySQL ALTER VIEW examples**

See the following tables orders and orderdetails from the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/).

### **orders orderdetails table**

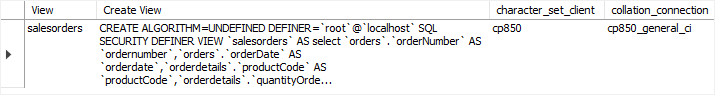
### **1) ALTER VIEW simple example**

First, [create a view](https://www.mysqltutorial.org/mysql-views/mysql-create-view/) based on the orders and orderdetails tables:

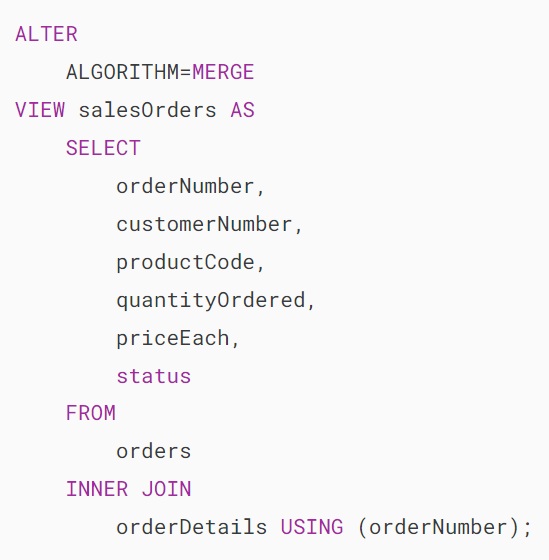


Second, [query data](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) from the view salesorders:





Third, use the ALTER VIEW statement to change the [processing algorithm of the view](https://www.mysqltutorial.org/mysql-views/mysql-view-processing-algorithms/) from UNDEFINED to MERGE and add customerNumber column to the view:



Finally, show the view information to see the effect:

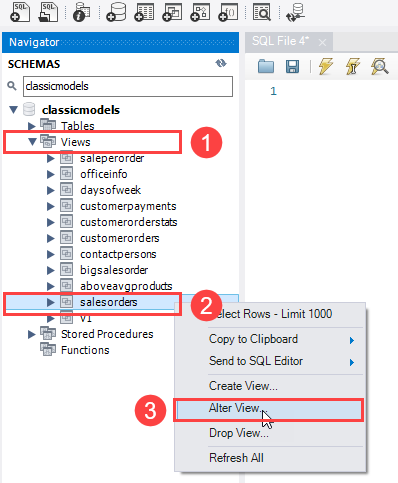


### 

### **2) ALTER VIEW using MySQL Workbench**

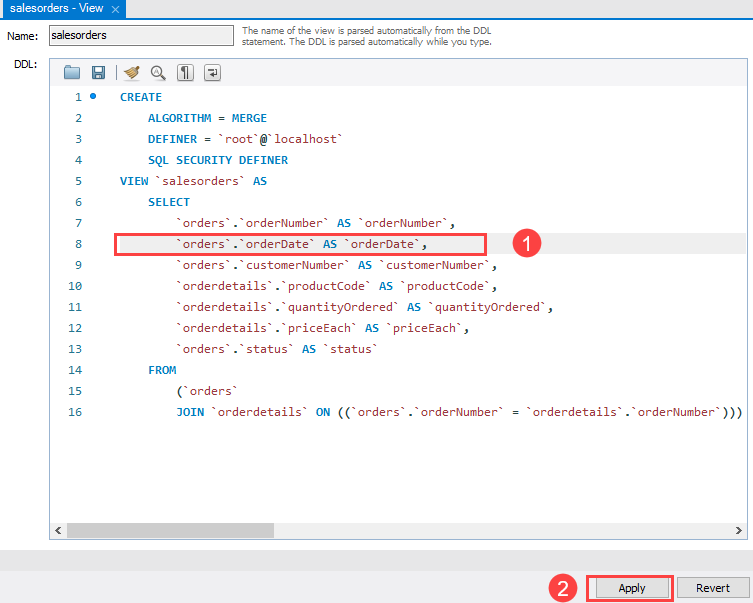
Using MySQL Workbench to modify an existing view is more practical. Because you need to see the whole view definition before changing it.

First, open the Views, right-click the view that you want to modify, and select Alter View… menu item:



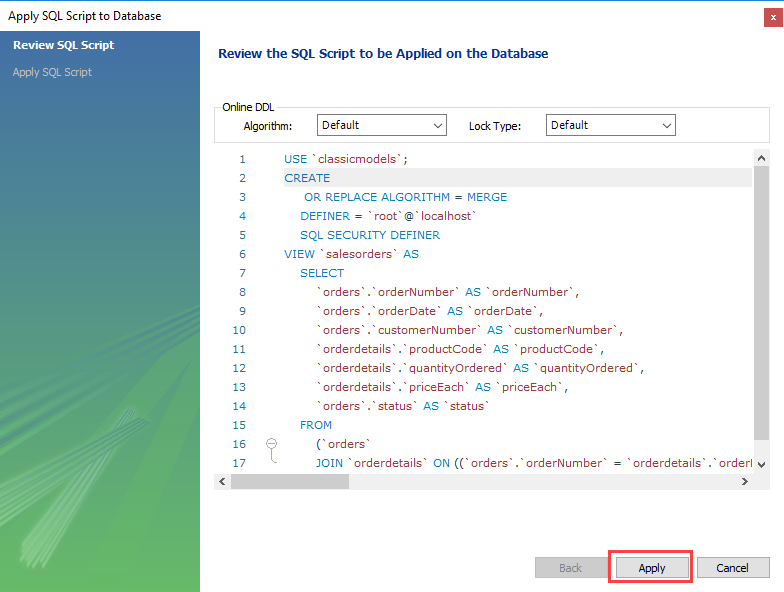
MySQL Workbench will open an editor that displays the view’s DDL.

Second, change the DDL of the view and click the Apply button to confirm the changes. If you want to revert the change, click the Revert button:



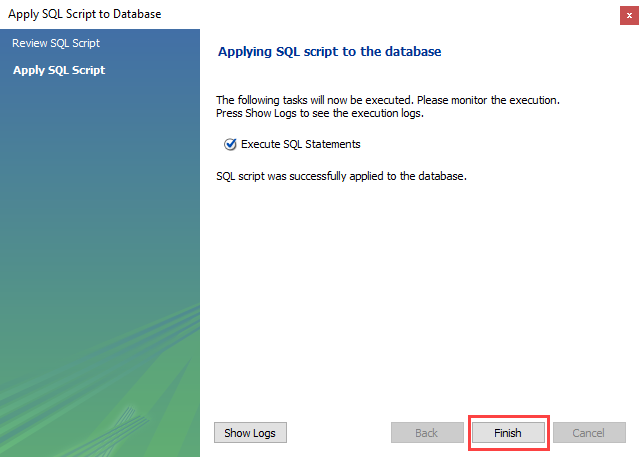
MySQL Workbench opens a confirmation window.

Third, click the Apply button to apply the change.



Note that MySQL Workbench uses [CREATE OR REPLACE VIEW](https://www.mysqltutorial.org/mysql-views/mysql-create-view/) statement instead of ALTER VIEW statement

Finally, click the Finish button to close the window.



# **Create MySQL Updatable Views**

**Summary**: in this tutorial, we will show you how to **create an updatable view** and update data in the underlying table through the view**.**

## **Introduction to MySQL updatable views**

In MySQL, views are not only queryable but also updatable. This implies that you can use the [INSERT](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) or [UPDATE](https://www.mysqltutorial.org/mysql-basics/mysql-update/) statement to add or modify rows of the base table through the updatable view.

Additionally, you can use the [DELETE](https://www.mysqltutorial.org/mysql-basics/mysql-delete/) statement to remove rows of the underlying table via the view.

However, to create an updatable view, the [SELECT statement](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) defining the view must not contain any of the following elements:

* [Aggregate functions](https://www.mysqltutorial.org/mysql-aggregate-functions/) such as [MIN](https://www.mysqltutorial.org/mysql-aggregate-functions/mysql-min/), [MAX](https://www.mysqltutorial.org/mysql-aggregate-functions/mysql-max-function/), [SUM](https://www.mysqltutorial.org/mysql-aggregate-functions/mysql-sum/), [AVG](https://www.mysqltutorial.org/mysql-aggregate-functions/mysql-avg/), and [COUNT](https://www.mysqltutorial.org/mysql-aggregate-functions/mysql-count/).
* [DISTINCT](https://www.mysqltutorial.org/mysql-basics/mysql-distinct/)
* [GROUP BY](https://www.mysqltutorial.org/mysql-basics/mysql-group-by/) clause.
* [HAVING](https://www.mysqltutorial.org/mysql-basics/mysql-having/) clause.
* [UNION](https://www.mysqltutorial.org/mysql-basics/mysql-union/) or UNION ALL clause.
* [Left join](https://www.mysqltutorial.org/mysql-basics/mysql-left-join/) or outer join.
* [Subquery](https://www.mysqltutorial.org/mysql-basics/mysql-subquery/)in the [SELECT](https://www.mysqltutorial.org/mysql-basics/mysql-select-from/) clause or in the [WHERE](https://www.mysqltutorial.org/mysql-basics/mysql-where/) clause that refers to the table appeared in the FROM clause.
* Reference non-updatable views in the FROM clause.
* Use literal values.
* Multiple references to any column of the base table.

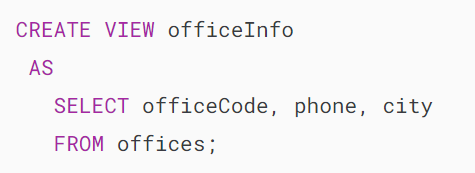
If you create a view with the TEMPTABLE algorithm, the view is not updatable.

Note that it is possible to create updatable views based on multiple tables using an [inner join](https://www.mysqltutorial.org/mysql-basics/mysql-inner-join/).

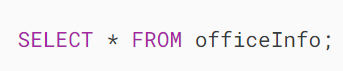
## **MySQL updatable view example**

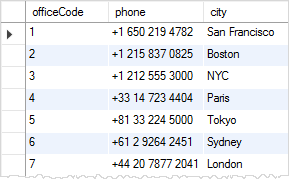
Let’s create an updatable view.

First, create a view named officeInfo  based on the offices  table in the [sample database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/). The view refers to three columns of the offices table: officeCode, phone, and city.

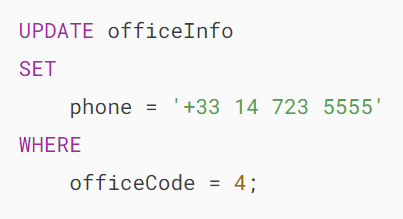


Second, retrieve data from the officeInfo view using the following statement:

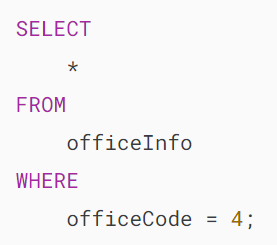




Third, change the phone number of the office with officeCode  4 through the officeInfo view using the following [UPDATE](https://www.mysqltutorial.org/mysql-basics/mysql-update/)statement.



Finally, query the data from the officeInfo  view to verify the change:

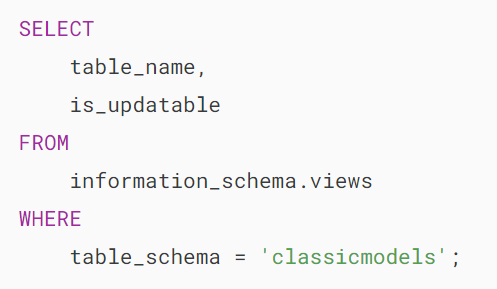


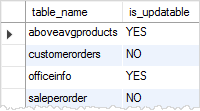
mysql updateable view example with officeInfo View

## **Checking updatable view information**

You can check if a view in a database is updatable by querying the is\_updatable column from the views table in the information\_schema database.

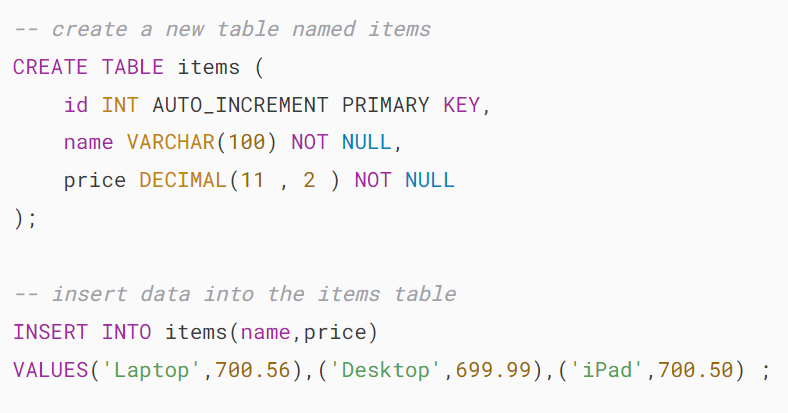
The following query gets all views from the [classicmodels database](https://www.mysqltutorial.org/getting-started-with-mysql/mysql-sample-database/) and shows which views are updatable.





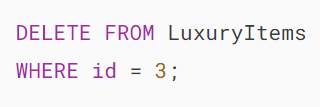
## **Removing rows through the view**

First, [create a table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) named items, [insert](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) some rows into the items table, and [create a view](https://www.mysqltutorial.org/mysql-views/mysql-create-view/) that contains items whose prices are greater than 700.



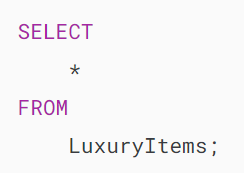


Second, use the DELETE statement to remove a row with id value 3.



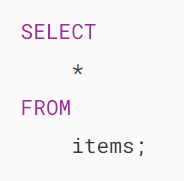
MySQL returns a message saying that 1 row(s) is affected.

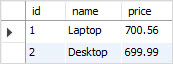
Third, check the data through the view again:



MySQL DELETE through View

Finally, query the data from the base table items to verify if the DELETE statement deleted the row.





The output shows that the row with id 3 was removed from the base table.

# **MySQL WITH CHECK OPTION Clause**

**Summary**: in this tutorial, you will learn how to ensure consistency of the database views using the WITH CHECK OPTION clause.

## **Introduction to WITH CHECK OPTION clause**

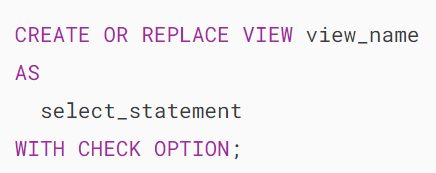
Sometimes, you [create a view](https://www.mysqltutorial.org/mysql-views/mysql-create-view/) to reveal the partial data of a table. However, a simple view is [updatable](https://www.mysqltutorial.org/mysql-views/mysql-updatable-views/), and therefore, it is possible to [update](https://www.mysqltutorial.org/mysql-basics/mysql-update/) data that is not visible through the view. This update makes the view inconsistent.

To ensure the consistency of the view, you use the WITH CHECK OPTION clause when you [create or modify the view](https://www.mysqltutorial.org/mysql-views/mysql-create-view/).

The WITH CHECK OPTION is an optional clause of the CREATE VIEW statement. This WITH CHECK OPTION prevents you from [updating](https://www.mysqltutorial.org/mysql-basics/mysql-update/) or [inserting](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) rows that are not visible through the view.

In other words, whenever you update or insert a row of the base tables through a view, MySQL ensures that the insert or update operation conforms with the definition of the view.

The following illustrates the syntax of the WITH CHECK OPTION clause:

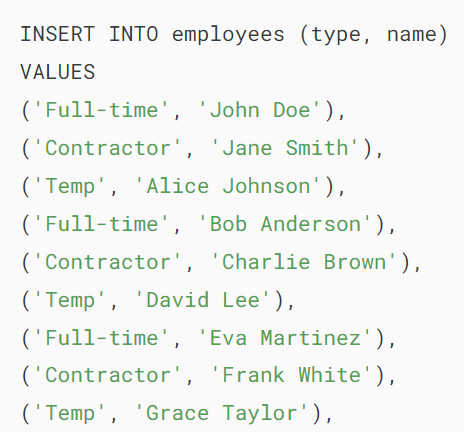
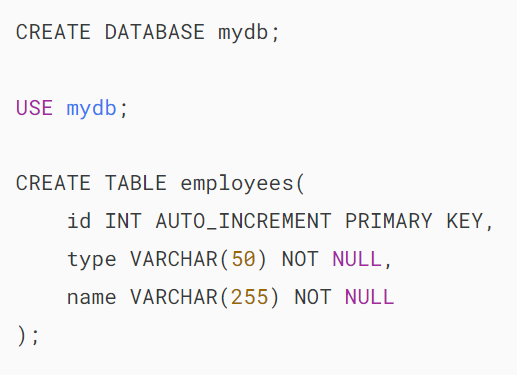


In this syntax, you add the WITH CHECK OPTION clause at the end of the CREATE VIEW statement.

## **MySQL WITH CHECK OPTION example**

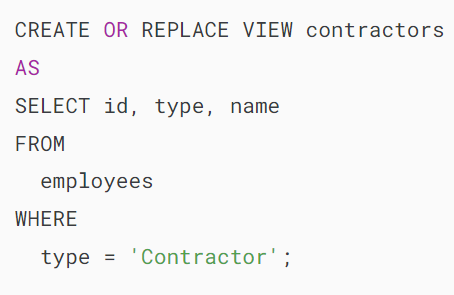
Let’s explore an example of using the WITH CHECK OPTION clause.

First, create a table called contacts:

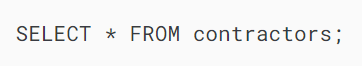


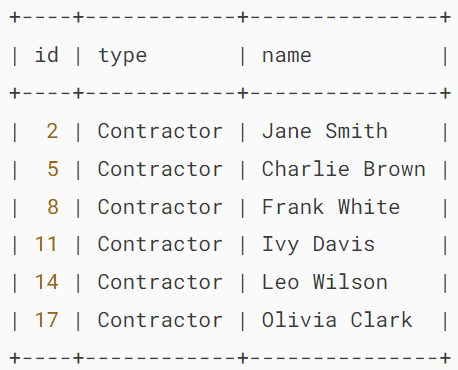


Second, create a view called contractors based on the employees table:



Third, query data from the contractors view:





The contractors view is [updatable](https://www.mysqltutorial.org/mysql-views/mysql-updatable-views/). For example, you can insert a new row into the employees table via the contractor view:

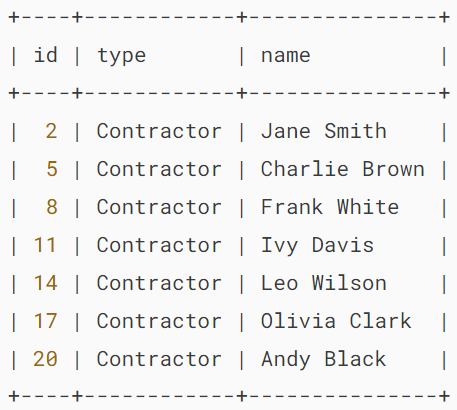


Output:



The statement inserts a new employee with the type Contractor into the employees table via the contractors view.

Fourth, retrieve data from the contractors view:



The output shows that Andy Black has been added successfully.

The problem is that you can add an employee with other types such as Full-time into the employees table via the contractors view. For example:

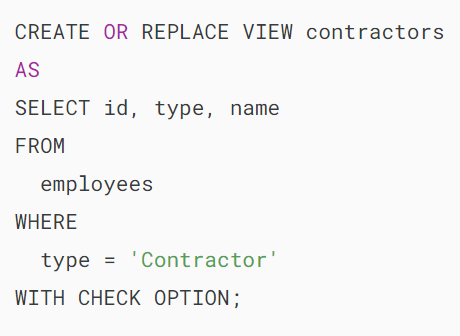


Output:

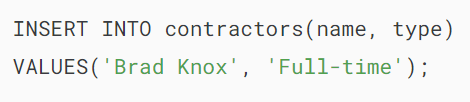


This statement successfully inserts a new row that is not visible via the contractors view into the employees table.

To prevent this, you need to add the WITH CHECK OPTION clause to the CREATE OR REPLACE VIEW statement like this:



If you attempt to insert or update rows that are not visible via the contractor, you’ll get an error. For example:



Output:



Because of the WITH CHECK OPTION, MySQL checks if the INSERT statement conforms with the SELECT statement that defines the view. If the INSERT statement does not conform, MySQL rejects it and issues an error.

## **Summary**

* Include the WITH CHECK OPTION clause in the CREATE VIEW statement to ensure the consistency of the view.

# **Understanding LOCAL & CASCADED in WITH CHECK OPTION Clause**

**Summary**: in this tutorial, you will learn the differences between LOCAL and CASCADED in WITH CHECK OPTION clause.

Note that you should be familiar with the WITH CHECK OPTION clause before going forward with this tutorial. If this is not the case, you can follow the WITH CHECK OPTION tutorial.

## **Introduction to scope of check LOCAL & CASCADED**

When you [create a view](https://www.mysqltutorial.org/mysql-views/mysql-create-view/) with the WITH CHECK OPTION clause and issue a DML statement against the view such as [INSERT](https://www.mysqltutorial.org/mysql-basics/mysql-insert/), [UPDATE](https://www.mysqltutorial.org/mysql-basics/mysql-update/), and [DELETE](https://www.mysqltutorial.org/mysql-basics/mysql-delete/), MySQL checks to ensure that the rows that are being changed are conformable to the definition of the view.

Because a view can be created based on other views, MySQL also checks the rules in the dependent views for data consistency.

To determine the scope of the check, MySQL provides two options: LOCAL and CASCADED. If you don’t specify the keyword explicitly in the WITH CHECK OPTION clause, MySQL uses CASCADED by default.

## **MySQL WITH CASCADED CHECK OPTION**

Consider the following example to understand the effect of the WITH CASCADED CHECK OPTION:

First, [create a table](https://www.mysqltutorial.org/mysql-basics/mysql-create-table/) named t1 with one column c whose data type is an integer.

CREATE TABLE t1 (

c INT

);Code language: SQL (Structured Query Language) (sql)

Next, [create a view](https://www.mysqltutorial.org/mysql-views/mysql-create-view/) v1 based on the t1 table with the data in the c column greater than 10.

CREATE OR REPLACE VIEW v1

AS

SELECT

c

FROM

t1

WHERE

c > 10;Code language: SQL (Structured Query Language) (sql)

Because we did not specify the WITH CHECK OPTION, the following statement works even though it does not conform with the definition of the v1 view.

INSERT INTO v1(c)

VALUES (5);Code language: SQL (Structured Query Language) (sql)

Then, create a view v2 based on the v1 view with WITH CASCADED CHECK OPTION clause.

CREATE OR REPLACE VIEW v2

AS

SELECT c FROM v1

WITH CASCADED CHECK OPTION;Code language: SQL (Structured Query Language) (sql)

Now, [insert a row](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) with value 5 into the t1 table through the v2 view.

INSERT INTO v2(c)

VALUES (5);Code language: SQL (Structured Query Language) (sql)

MySQL issued the following error message:

Error Code: 1369. CHECK OPTION failed 'classicmodels.v2'Code language: SQL (Structured Query Language) (sql)

It fails the new row that does not conform with the definition of v2 view.

After that, create a new view named v3 based on v2.

CREATE OR REPLACE VIEW v3

AS

SELECT c

FROM v2

WHERE c < 20;Code language: SQL (Structured Query Language) (sql)

Insert a new row into the t1 table through the v3 view with value 8.

INSERT INTO v3(c)

VALUES (8);Code language: SQL (Structured Query Language) (sql)

MySQL issued the following error message:

Error Code: 1369. CHECK OPTION failed 'classicmodels.v3'Code language: SQL (Structured Query Language) (sql)

The [insert statement](https://www.mysqltutorial.org/mysql-basics/mysql-insert/) fails even though the row seems to conform with the definition of the v3 view.

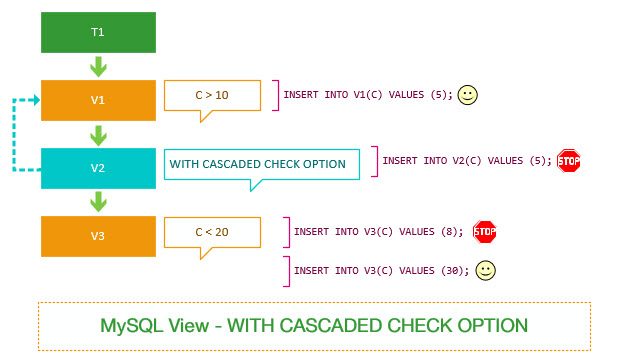
Because the view v3 is dependent on the v2 view, and the v2 view has the option WITH CASCADED CHECK OPTION.

However, the following insert statement works.

INSERT INTO v3(c) VALUES (30);Code language: SQL (Structured Query Language) (sql)

Because the v3 view does not have a WITH CHECK OPTION, and the statement conforms with the definition of the v2 view.

In conclusion, when a view uses a WITH CASCADED CHECK OPTION, MySQL checks the rules of the view and also the rules of the underlying views recursively.



## **MySQL WITH LOCAL CHECK OPTION**

Let’s use the same example above for the WITH LOCAL CHECK OPTION to see the differences.

First, change the v2 view to use the WITH LOCAL CHECK OPTION instead.

ALTER VIEW v2 AS

SELECT

c

FROM

v1

WITH LOCAL CHECK OPTION;Code language: SQL (Structured Query Language) (sql)

Second, insert the same row that we did with the example above.

INSERT INTO v2(c)

VALUES (5);Code language: SQL (Structured Query Language) (sql)

It succeeded.

Because v2 view does not have any rules. The view v2 is dependent on the v1 view. However, v1 view does not specify a check option, therefore, MySQL skips checking the rules in v1 view.

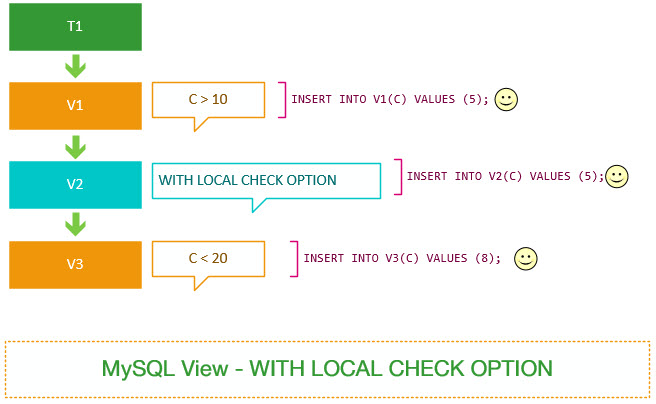
Notice that this statement failed in the v2 view created with a WITH CASCADED CHECK OPTION.

Third, insert the same row to the t1 table through the v3 view.

INSERT INTO v3(c) VALUES (8);Code language: SQL (Structured Query Language) (sql)

It succeeds in this case because MySQL did not check the rules of v1 view due to the WITH LOCAL CHECK OPTION of the v2 view.

Also, notice that this statement failed in the example that v2 created with a WITH CASCADED CHECK OPTION.



So if a view uses a WITH LOCAL CHECK OPTION, MySQL checks the rules of views that have a WITH LOCAL CHECK OPTION and a WITH CASCADED CHECK OPTION.

It is different from the view that uses a WITH CASCADED CHECK OPTION that MySQL checks the rules of all dependent views.

Notice that before MySQL 5.7.6, if you use a view with a WITH LOCAL CHECK OPTION, MySQL only checks the rules of the current view and it does not check the rules of the underlying views.

https://www.mysqltutorial.org/mysql-views/mysql-view-with-local-check-option-vs-with-cascaded-check-option/