When you use the [SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select/) statement to query data from one or more tables, you get a result set.

For example, the following statement returns the product name, brand, and list price of all products from the products and brands tables:

SELECT

product\_name,

brand\_name,

list\_price

FROM

production.products p

INNER JOIN production.brands b

ON b.brand\_id = p.brand\_id;

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

Next time, if you want to get the same result set, you can save this query into a text file, open it, and execute it again.

SQL Server provides a better way to save this query in the database catalog through a view.

A view is a named query stored in the database catalog that allows you to refer to it later.

So the query above can be stored as a view using the [CREATE VIEW](https://www.sqlservertutorial.net/sql-server-views/sql-server-create-view/) statement as follows:

CREATE VIEW sales.product\_info

AS

SELECT

product\_name,

brand\_name,

list\_price

FROM

production.products p

INNER JOIN production.brands b

ON b.brand\_id = p.brand\_id;

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

Later, you can reference to the view in the SELECT statement like a table as follows:

SELECT \* FROM sales.product\_info;

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

When receiving this query, SQL Server executes the following query:

SELECT

\*

FROM (

SELECT

product\_name,

brand\_name,

list\_price

FROM

production.products p

INNER JOIN production.brands b

ON b.brand\_id = p.brand\_id;

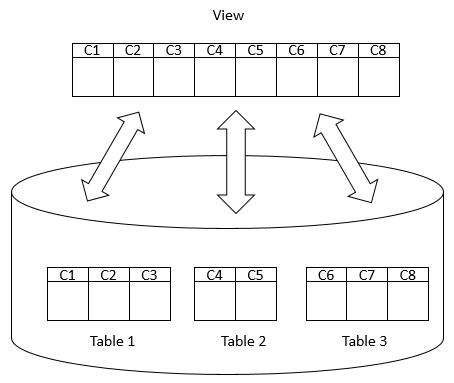
);

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

By definition, views do not store data except for [indexed views](https://www.sqlservertutorial.net/sql-server-views/sql-server-indexed-view/).

A view may consist of columns from multiple tables using joins or just a subset of columns of a single table. This makes views useful for abstracting or hiding complex queries.

The following picture illustrates a view that includes columns from multiple tables:



Advantages of views

Generally speaking, views provide the following advantages:

Security

You can restrict users to access directly to a table and allow them to access a subset of data via views.

For example, you can allow users to access customer name, phone, email via a view but restrict them to access the bank account and other sensitive information.

Simplicity

A relational database may have many tables with complex relationships e.g., one-to-one and one-to-many that make it difficult to navigate.

However, you can simplify the complex queries with joins and conditions using a set of views.

Consistency

Sometimes, you need to write a complex formula or logic in every query.

To make it consistent, you can hide the complex queries logic and calculations in views.

Once views are defined, you can reference the logic from the views rather than rewriting it in separate queries.

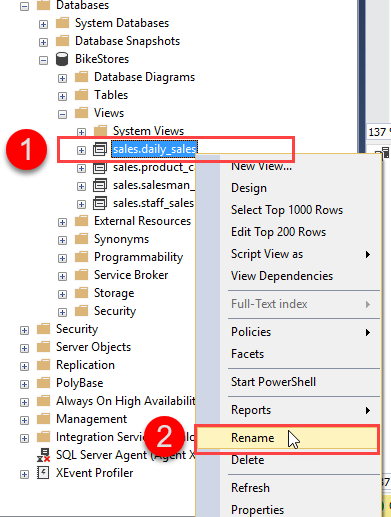
Managing views in SQL Server

* [Creating a new view](https://www.sqlservertutorial.net/sql-server-views/sql-server-create-view/) – show you how to create a new view in a SQL Server database.
* [Renaming a view](https://www.sqlservertutorial.net/sql-server-views/sql-server-rename-view/) – learn how to rename a view using the SQL Server Management Studio (SSMS) or Transact-SQL command.
* [Listing views in SQL Server](https://www.sqlservertutorial.net/sql-server-views/sql-server-list-views/) – discuss the various way to list all views in a SQL Server Database.
* [Getting view information](https://www.sqlservertutorial.net/sql-server-views/sql-server-get-view-information/) – how to get information about a view.
* [Removing a view](https://www.sqlservertutorial.net/sql-server-views/sql-server-drop-view/) – guide you how to use the DROP VIEW statement to remove one or more views from the database.
* [Creating an indexed view](https://www.sqlservertutorial.net/sql-server-views/sql-server-indexed-view/) – show you how to create an indexed view against tables that have infrequent data modification to optimize the performance of the view.

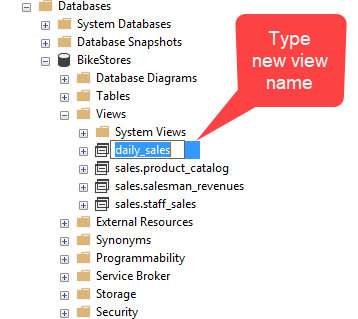
To rename the name of a view you follow these steps:

First, in Object Explorer, expand the Databases, choose the database name which contains the view that you want to rename and expand the Views folder.

Second, right-click the view that you want to rename and select **Rename**.



Third, enter the new name for the view.



SQL Server rename view using Transact-SQL

If you want to rename a view programmatically, you can use the sp\_rename [stored procedure](https://www.sqlservertutorial.net/sql-server-stored-procedures/):

EXEC sp\_rename

@objname = 'sales.product\_catalog',

@newname = 'product\_list';

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

In this statement:

* First, pass the name of the view which you want to rename using the @objname parameter and the new view name to using the @newname parameter. Note that in the @objectname you must specify the schema name of the view. However, in the @newname parameter, you must not.
* Second, execute the statement.

To list all views in a SQL Server Database, you query the sys.views or sys.objects catalog view. Here is an example:

SELECT

OBJECT\_SCHEMA\_NAME(v.object\_id) schema\_name,

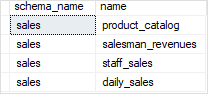
v.name

FROM

sys.views as v;

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

The query returns the following list of schema names and view names:



In this example, we used the OBJECT\_SCHEMA\_NAME() function to get the schema names of the views.

The following query returns a list of views through the sys.objects view:

SELECT

OBJECT\_SCHEMA\_NAME(o.object\_id) schema\_name,

o.name

FROM

sys.objects as o

WHERE

o.type = 'V';

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

## Creating a stored procedure to show views in SQL Server Database

The following stored procedure wraps the query above to list all views in the SQL Server Database based on the input schema name and view name:

CREATE PROC usp\_list\_views(

@schema\_name AS VARCHAR(MAX) = NULL,

@view\_name AS VARCHAR(MAX) = NULL

)

AS

SELECT

OBJECT\_SCHEMA\_NAME(v.object\_id) schema\_name,

v.name view\_name

FROM

sys.views as v

WHERE

(@schema\_name IS NULL OR

OBJECT\_SCHEMA\_NAME(v.object\_id) LIKE '%' + @schema\_name + '%') AND

(@view\_name IS NULL OR

v.name LIKE '%' + @view\_name + '%');

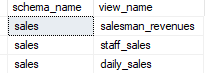
Code language: SQL (Structured Query Language) (sql)

For example, if you want to know the views that contain the word sales, you can call the stored procedure usp\_list\_views:

EXEC usp\_list\_views @view\_name = 'sales'

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

Here is the result:



## Getting the view information using the sql.sql\_module catalog

To get the information of a view, you use the system catalog sys.sql\_module and the OBJECT\_ID() function:

SELECT

definition,

uses\_ansi\_nulls,

uses\_quoted\_identifier,

is\_schema\_bound

FROM

sys.sql\_modules

WHERE

object\_id

= object\_id(

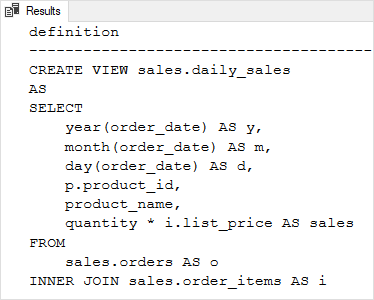
'sales.daily\_sales'

);

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

In this query, you pass the name of the view to the OBJECT\_ID() function in the WHERE clause. The OBJECT\_ID() function returns an identification number of a schema-scoped database object.

Here is the output:



Note that you need to output the result to the text format in order to see the SELECT statement clearly as the above picture.

To show the results as text, from the query editor, you press **Ctrl-T** keyboard shortcut or click the **Results to Text** button as shown in the following screenshot:

SQL Server Getting view definition - show results to text

## Getting view information using the sp\_helptext stored procedure

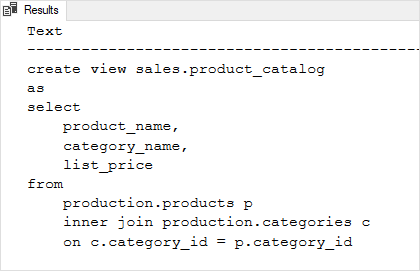
The sp\_helptext stored procedure returns the definition of a user-defined object such as a view.

To get a view’s information, you pass the view name to the sp\_helptext stored procedure. For example, the following statement returns the information of the sales.product\_catalog view:

EXEC sp\_helptext 'sales.product\_catalog' ;

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

The following picture shows the output:



## Getting the view information using OBJECT\_DEFINITION() function

Another way to get the view information is to use the OBJECT\_DEFINITION() and OBJECT\_ID() functions as follows:

SELECT

OBJECT\_DEFINITION(

OBJECT\_ID(

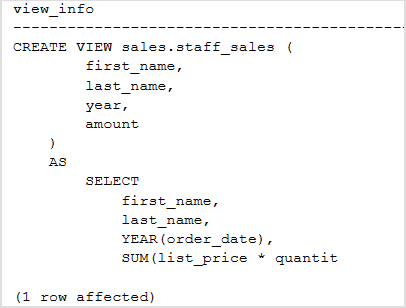
'sales.staff\_sales'

)

) view\_info;

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

The following picture shows the output:



To remove a [view](https://www.sqlservertutorial.net/sql-server-views/) from a database, you use the DROP VIEW statement as follows:

DROP VIEW [IF EXISTS] schema\_name.view\_name;

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

In this syntax, you specify the name of the view that you want to drop after the DROP VIEW keywords. If the view belongs to a schema, you must also explicitly specify the name of the schema to which the view belongs.

If you attempt to remove a view that does not exist, SQL Server will issue an error. The IF EXISTS clause prevents an error from occurring when you delete a view that does not exist.

To remove multiple views, you use the following syntax:

DROP VIEW [IF EXISTS]

schema\_name.view\_name1,

schema\_name.view\_name2,

...;

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

In this syntax, the views are separated by commas.

Note that when you drop a view, SQL Server removes all permissions for the view.

## SQL Server DROP VIEW examples

We will use the sales.daily\_sales and sales.staff\_sales views created in the [CREATE VIEW](https://www.sqlservertutorial.net/sql-server-views/sql-server-create-view/) tutorial for the demonstration.

### Removing one view example

The following example shows how to drop the sales.daily\_sales view from the sample database:

DROP VIEW IF EXISTS sales.daily\_sales;

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

### Removing multiple views example

The following statement [creates a view](https://www.sqlservertutorial.net/sql-server-views/sql-server-create-view/) named product\_catalogs for demonstration purpose:

CREATE VIEW sales.product\_catalog

AS

SELECT

product\_name,

category\_name,

brand\_name,

list\_price

FROM

production.products p

INNER JOIN production.categories c

ON c.category\_id = p.category\_id

INNER JOIN production.brands b

ON b.brand\_id = p.brand\_id;

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

The following statement removes both sales.staff\_sales and sales.product\_catalog views at the same time:

DROP VIEW IF EXISTS

sales.staff\_sales,

sales.product\_catalogs;