**Types of SQL Server Views**

Views are virtual tables that are compiled at run time. The data associated with views are not physically stored in the view, but it is stored in the base tables of the view. A view can be made over one or more database tables. Generally we put those columns in view that we need to retrieve/query again and again. Once you have created the view, you can query view like as table. We can make index, trigger on view.

In SQL Server we make views for security purpose since it restricts the user to view some columns/fields of the table(s). Views show only those columns that are present in the query which is used to make view. One more advantage of Views is, data abstraction since the end user is not aware of all the data present in database table.

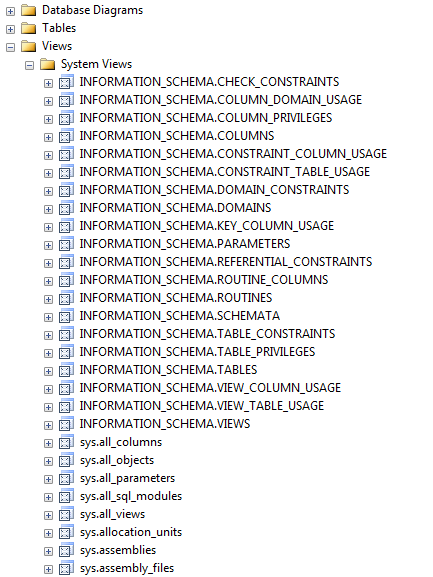
**Syntax for View**

1. **CREATE VIEW view\_name**
2. **AS**
3. **select\_statement[]**

**Types of Views**

In SQL Server there are 2 types of views.

**1. System Views**

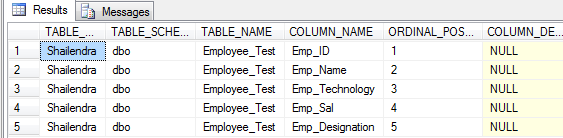
System defined Views are predefined Views that already exist in the Master database of SQL Server. These are also used as template Views for all newly created databases. These system Views will be automatically attached to any user defined database.  


**There are 3 types of System Views.**

1. ***Information Schema View***

In SQL Server we have twenty different schema views. These are used to display information of a database, like as tables and columns. This type of view starts with INFORMATION\_SCHEMA and after this view name.

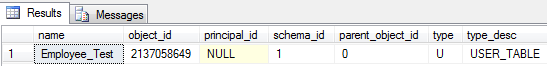
* + 1. ***--Create a table***
    2. **create table Employee\_Test**
    3. **(**
    4. **Emp\_ID int identity,**
    5. **Emp\_Name varchar(55),**
    6. **Emp\_Technology varchar(55),**
    7. **Emp\_Sal decimal (10,2),**
    8. **Emp\_Designation varchar(20)**
    9. **)**
    10. ***--To view detailed information of the columns of table Employee\_Test***
    11. **SELECT \* FROM INFORMATION\_SCHEMA.COLUMNS**
    12. **where TABLE\_NAME='Employee\_Test'**



1. ***Catalog View***

Catalog Views were introduced with SQL Server 2005. These are used to show database self-describing information.

**1. select \* from sys.tables**



1. ***Dynamic Management View***

Dynamic Management Views were introduced in SQL Server 2005. These Views give the administrator information of the database about the current state of the SQL Server machine. These values help the administrator to analyze problems and tune the server for optimal performance. These are of 2 types

1. ***Server-scoped Dynamic Management View***

These are stored only in the Master database.

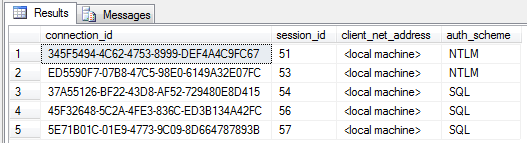
1. ***Database-scoped Dynamic Management View***

These are stored in each database.

***1. --To see all SQL Server connections***

**2. SELECT connection\_id,session\_id,client\_net\_address,auth\_scheme**

* + 1. **FROM sys.dm\_exec\_connections**



**2. User-Defined Views**

These types of view are defined by users. We have two types of user defined views.

***1. Simple View***

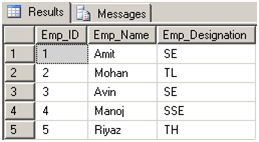
When we create a view on a single table, it is called simple view.

* + 1. ***--Now Insert data to table Employee\_Test***
    2. **Insert into Employee\_Test values ('Amit','PHP',12000,'SE');**
    3. **Insert into Employee\_Test values ('Mohan','ASP.NET',15000,'TL');**
    4. **Insert into Employee\_Test values ('Avin','C#',14000,'SE');**
    5. **Insert into Employee\_Test values ('Manoj','JAVA',22000,'SSE');**
    6. **Insert into Employee\_Test values ('Riyaz','VB',18000,'TH');**
    7. ***-- Now create view on single table Employee\_Test***
    8. **create VIEW vw\_Employee\_Test**
    9. **AS**
    10. **Select Emp\_ID ,Emp\_Name ,Emp\_Designation**
    11. **From Employee\_Test**

http://www.dotnet-tricks.com/Content/images/sqlserver/success.png

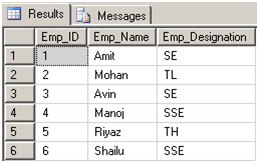
**1. *– Query view like as table***

**2. Select \* from vw\_Employee\_Test**

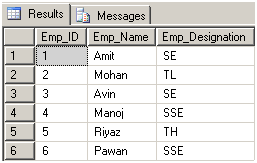


**In simple view we can insert, update, and delete data.** We can only insert data in simple view if we have primary key and all not null fields in the view.

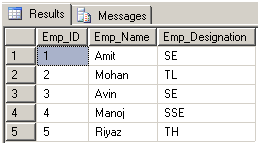
* + 1. ***-- Insert data to view vw\_Employee\_Test***
    2. **insert into vw\_Employee\_Test(Emp\_Name, Emp\_Designation) values ('Shailu','SSE')**
    3. ***-- Now see the affected view***
    4. **Select \* from vw\_Employee\_Test**



* + 1. ***-- Update data to view vw\_Employee\_Test***
    2. **Update vw\_Employee\_Test set Emp\_Name = 'Pawan' where Emp\_ID = 6**
    3. ***-- Now see the affected view***
    4. **Select \* from vw\_Employee\_Test**



* + 1. ***-- Delete data from view vw\_Employee\_Test***
    2. **delete from vw\_Employee\_Test where Emp\_ID = 6**
    3. ***-- Now see the affected view***
    4. **Select \* from vw\_Employee\_Test**



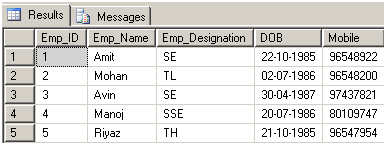
***2. Complex View***

When we create a view on more than one table, it is called complex view.

* + 1. ***--Create another table***
    2. **create table Personal\_Info**
    3. **(**
    4. **Emp\_Name varchar(55),**
    5. **FName varchar(55),**
    6. **DOB varchar(55),**
    7. **Address varchar(55),**
    8. **Mobile int,**
    9. **State varchar(55)**
    10. **)**
    11. ***-- Now Insert data***
    12. **Insert into Personal\_Info values ('G.Chaudary','22-10-1985','Ghaziabad',96548922,'UP');**
    13. **Insert into Personal\_Info values ('B.S.Chauhan','02-07-1986','Haridwar',96548200,'UK');**
    14. **Insert into Personal\_Info values ('A.Panwar','30-04-1987','Noida',97437821,'UP');**
    15. **Insert into Personal\_Info values ('H.C.Patak','20-07-1986','Rampur',80109747,'UP');**
    16. **Insert into Personal\_Info values ('M.Shekh','21-10-1985','Delhi',96547954,'Delhi');**
    17. ***-- Now create view on two tables Employee\_Test and Personal\_Info***
    18. **Create VIEW vw\_Employee\_Personal\_Info**
    19. **As**
    20. **Select e.Emp\_ID, e.Emp\_Name,e.Emp\_Designation,p.DOB,p.Mobile**
    21. **From Employee\_Test e INNER JOIN Personal\_Info p**
    22. **On e.Emp\_Name = p. Emp\_Name**

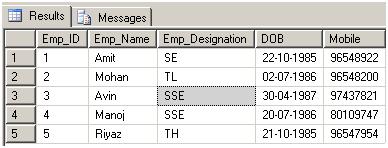
http://www.dotnet-tricks.com/Content/images/sqlserver/success.png

* + 1. ***-- Now Query view like as table***
    2. **Select \* from vw\_Employee\_Personal\_Info**



**We can only update data in complex view.** We can't insert, update data in complex view.

* + 1. ***--Update view***
    2. **update vw\_Employee\_Personal\_Info set Emp\_Designation = 'SSE' where Emp\_ID = 3**
    3. ***-- See affected view***
    4. **Select \* from vw\_Employee\_Personal\_Info**



Note

1. We make views for security purpose since it restricts the user to view some columns/fields of the table(s).
2. One more advantage of Views is, data abstraction since the end user is not aware of all the data present in database table

# Types of Views

**SQL Server 2008 R2**

[Other Versions](javascript:;)



* [SQL Server 2008](http://technet.microsoft.com/en-us/library/ms190426(d=printer,v=sql.100).aspx)
* [SQL Server 2005](http://technet.microsoft.com/en-us/library/ms190426(d=printer,v=sql.90).aspx)

You can create standard views, indexed views, and partitioned views.

[Standard Views](javascript:void(0))

Combining data from one or more tables through a standard view lets you satisfy most of the benefits of using views. These include focusing on specific data and simplifying data manipulation.

[Indexed Views](javascript:void(0))

An indexed view is a view that has been materialized. This means it has been computed and stored. You index a view by creating a unique clustered index on it. Indexed views dramatically improve the performance of some types of queries. Indexed views work best for queries that aggregate many rows. They are not well-suited for underlying data sets that are frequently updated.

[Partitioned Views](javascript:void(0))

A partitioned view joins horizontally partitioned data from a set of member tables across one or more servers. This makes the data appear as if from one table. A view that joins member tables on the same instance of SQL Server is a local partitioned view.

|  |
| --- |
| **NoteNote** |
| The preferred method for partitioning data locally is through partitioned tables. For more information, see [Partitioned Tables and Indexes](http://technet.microsoft.com/en-us/library/ms188706(v=sql.105).aspx). |

When a view joins data from tables across servers, it is a distributed partitioned view. Distributed partitioned views are used to implement a federation of database servers. A federation is a group of servers administered independently, but which cooperate to share the processing load of a system. Forming a federation of database servers by partitioning data is the mechanism that lets you scale out a set of servers to support the processing requirements of large, multi-tiered Web sites.