Views Lab Manual + Northwind

Session: 2022 – 2026

Submitted by:

Saad Ahmad Malik 2022-CS-1

Supervised by:

Mr. Nazeef ul Haq

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Views

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. You 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.

Types of views in SQL Server:

There are the following two types of views:

1. User-Defined Views
2. System-Defined Views

First, we discuss User-Defined Views.

I will use Database Northwind to illustrate the concept of Views in DBMS.

Create SQL VIEW in SQL Server

1. CREATE VIEW view_name AS
2. SELECT columns
3. FROM tables
4. WHERE conditions; Let us create some views.

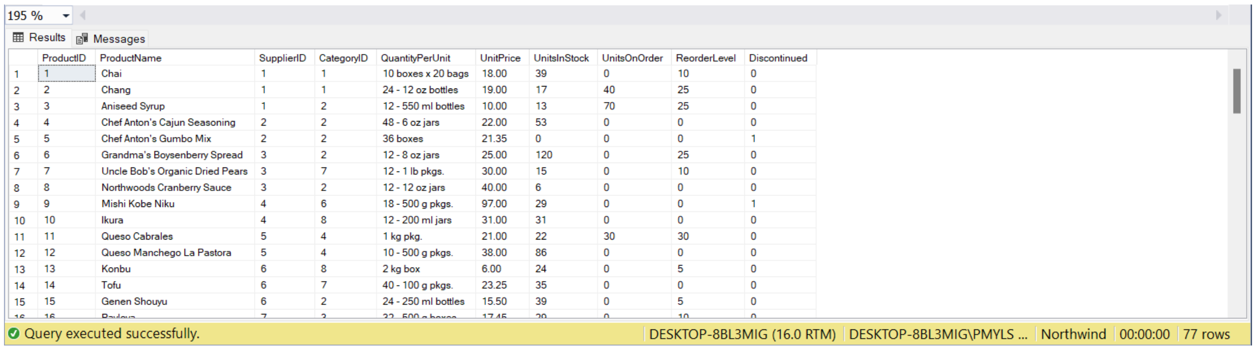
Method 1: We can select all columns of a table. The following example demonstrates that:

```
Create View Products_View AS
SELECT * FROM Products
GO
```

To Execute this View:

```
SELECT * FROM Products_View
```

Output:



195 %

Results | Messages

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	1	Chai	1	1	10 boxes x 20 bags	18.00	39	0	10	0
2	2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	0
3	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00	13	70	25	0
4	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.00	53	0	0	0
5	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1
6	6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.00	120	0	25	0
7	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.00	15	0	10	0
8	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	0
9	9	Mishi Kobe Niku	4	6	18 - 500 g pkgs.	97.00	29	0	0	1
10	10	Ikura	4	8	12 - 200 ml jars	31.00	31	0	0	0
11	11	Queso Cabrales	5	4	1 kg pkg.	21.00	22	30	30	0
12	12	Queso Manchego La Pastora	5	4	10 - 500 g pkgs.	38.00	86	0	0	0
13	13	Konbu	6	8	2 kg box	6.00	24	0	5	0
14	14	Tofu	6	7	40 - 100 g pkgs.	23.25	35	0	0	0
15	15	Genen Shouyu	6	2	24 - 250 ml bottles	15.50	39	0	5	0

Query executed successfully. | DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 77 rows

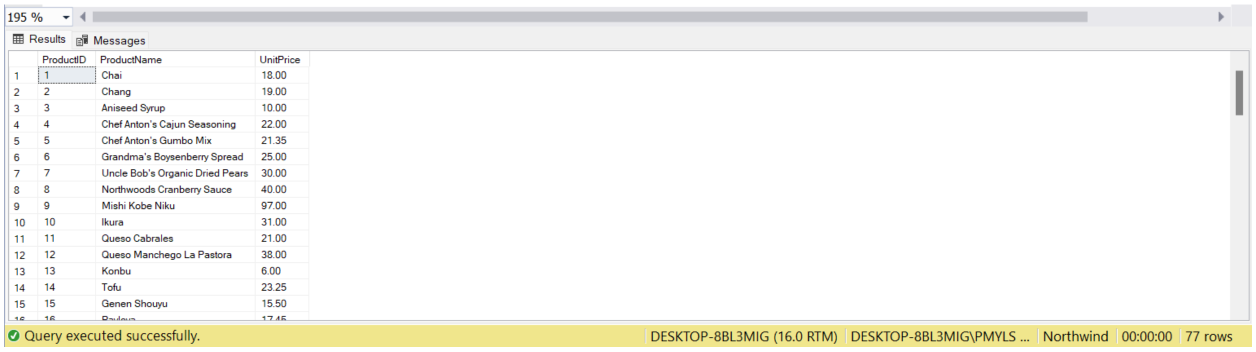
Method 2: We can select specific columns of a table. The following example demonstrates that:

```sql
Create View Products_View_1 AS
SELECT ProductID,ProductName,UnitPrice FROM Products
GO
```

To Execute this View:

```sql
SELECT * FROM Products_View_1
```

Output:



	ProductID	ProductName	UnitPrice
1	1	Chai	18.00
2	2	Chang	19.00
3	3	Aniseed Syrup	10.00
4	4	Chef Anton's Cajun Seasoning	22.00
5	5	Chef Anton's Gumbo Mix	21.35
6	6	Grandma's Boysenberry Spread	25.00
7	7	Uncle Bob's Organic Dried Pears	30.00
8	8	Northwoods Cranberry Sauce	40.00
9	9	Mishi Kobe Niku	97.00
10	10	Ikura	31.00
11	11	Queso Cabrales	21.00
12	12	Queso Manchego La Pastora	38.00
13	13	Konbu	6.00
14	14	Tofu	23.25
15	15	Genen Shouyu	15.50

```
Query executed successfully.   DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 77 rows
```

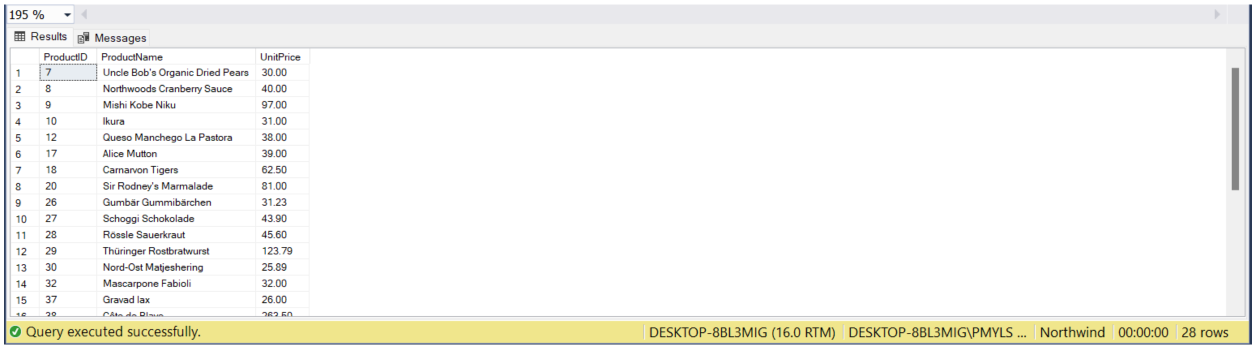
Method 3: We can select columns from a table with specific conditions. The following example demonstrates that:

```sql
Create View Products_View_2 AS
SELECT ProductID,ProductName,UnitPrice
FROM Products
WHERE UnitPrice > 25
GO
```

To Execute this View:

```sql
SELECT * FROM Products_View_2
```

Output:



	ProductID	ProductName	UnitPrice
1	7	Uncle Bob's Organic Dried Pears	30.00
2	8	Northwoods Cranberry Sauce	40.00
3	9	Mishi Kobe Niku	97.00
4	10	Ikura	31.00
5	12	Queso Manchego La Pastora	38.00
6	17	Alice Mutton	39.00
7	18	Carnarvon Tigers	62.50
8	20	Sir Rodney's Marmalade	81.00
9	26	Gumbär Gummibärchen	31.23
10	27	Schoggi Schokolade	43.90
11	28	Rössle Sauerkraut	45.60
12	29	Thüringer Rostbratwurst	123.79
13	30	Nord-Ost Matjeshering	25.89
14	32	Mascarpone Fabioli	32.00
15	37	Gravad lax	26.00

```
Query executed successfully.   DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 28 rows
```

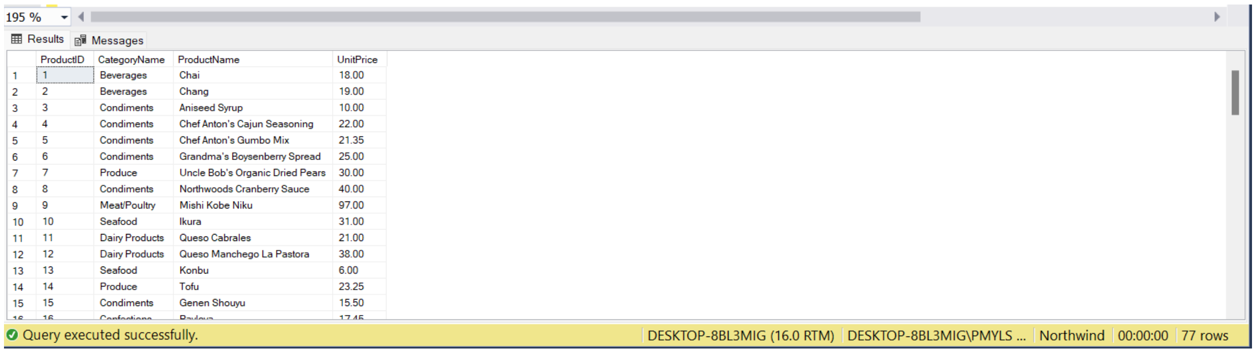
Method 4: We can create a view that will hold the columns of different tables. The following example demonstrates that:

```sql
Create View Products_View_3 AS
SELECT ProductID,CategoryName,ProductName,UnitPrice
FROM Products
JOIN Categories ON Categories.CategoryID = Products.CategoryID
GO
```

To Execute this View:

```sql
SELECT * FROM Products_View_3
```

Output:



195 %

Results | Messages

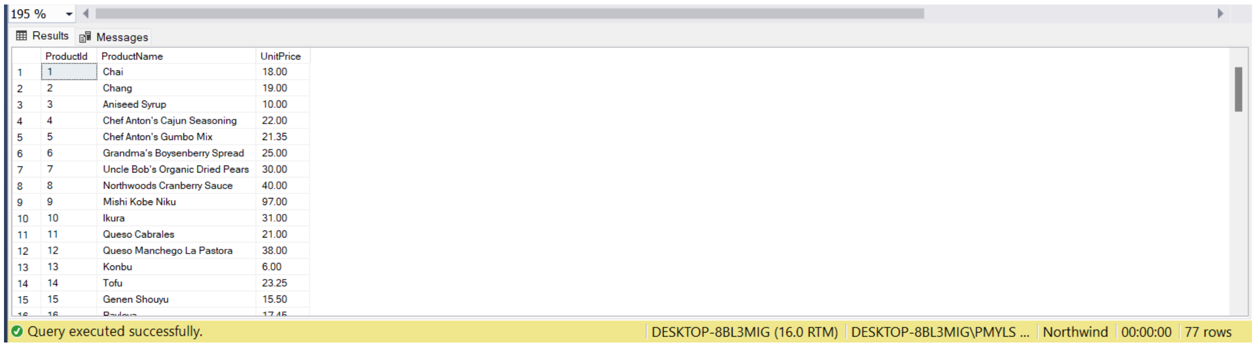
	ProductID	CategoryName	ProductName	UnitPrice
1	1	Beverages	Chai	18.00
2	2	Beverages	Chang	19.00
3	3	Condiments	Aniseed Syrup	10.00
4	4	Condiments	Chef Anton's Cajun Seasoning	22.00
5	5	Condiments	Chef Anton's Gumbo Mix	21.35
6	6	Condiments	Grandma's Boysenberry Spread	25.00
7	7	Produce	Uncle Bob's Organic Dried Pears	30.00
8	8	Condiments	Northwoods Cranberry Sauce	40.00
9	9	Meat/Poultry	Mishi Kobe Niku	97.00
10	10	Seafood	Ikura	31.00
11	11	Dairy Products	Queso Cabrales	21.00
12	12	Dairy Products	Queso Manchego La Pastora	38.00
13	13	Seafood	Konbu	6.00
14	14	Produce	Tofu	23.25
15	15	Condiments	Genen Shouyu	15.50

Query executed successfully. | DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 77 rows

Retrieve Data from View in SQL Server

This SQL CREATE VIEW example would create a virtual table based on the result set of the select statement. Now we can retrieve data from a view as follows:

1. Select * from Products_View
2. Select ProductId, ProductName, UnitPrice from Products_View



195 %

Results | Messages

	ProductId	ProductName	UnitPrice
1	1	Chai	18.00
2	2	Chang	19.00
3	3	Aniseed Syrup	10.00
4	4	Chef Anton's Cajun Seasoning	22.00
5	5	Chef Anton's Gumbo Mix	21.35
6	6	Grandma's Boysenberry Spread	25.00
7	7	Uncle Bob's Organic Dried Pears	30.00
8	8	Northwoods Cranberry Sauce	40.00
9	9	Mishi Kobe Niku	97.00
10	10	Ikura	31.00
11	11	Queso Cabrales	21.00
12	12	Queso Manchego La Pastora	38.00
13	13	Konbu	6.00
14	14	Tofu	23.25
15	15	Genen Shouyu	15.50

Query executed successfully. | DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 77 rows

Figure: Example of selecting specific columns from a View

The preceding query shows that we can select all the columns or some specific columns from a view.

Dropping a View in SQL Server

We can use the Drop command to drop a view. For example, to drop the view Products_View_3, we can use the following statement.

1. Drop View Products_View_3

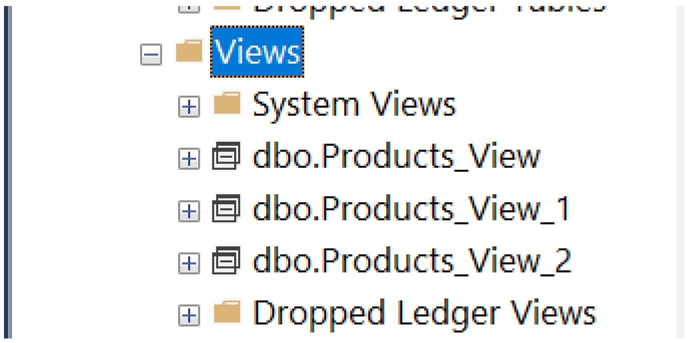


Figure: Drop View Products_View_3, after execution

For the moment we will create the view back.

Renaming the View in SQL Server

We can use the sp_rename system procedure to rename a view. The syntax of the sp_rename command is given below:

Sp_Rename OldViewName , NewViewName

Example:

We will rename the Products_View

```
Sp_Rename Products_View, View_Products
```

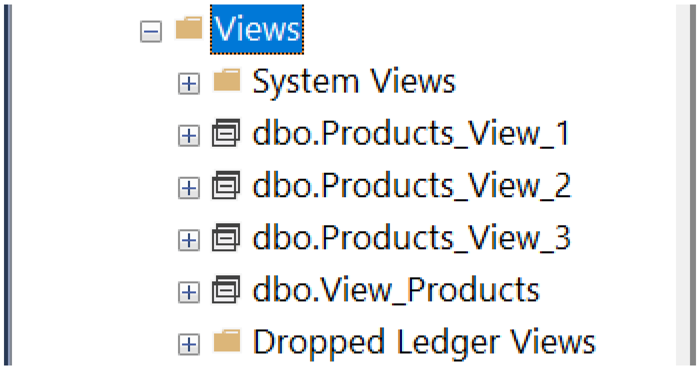
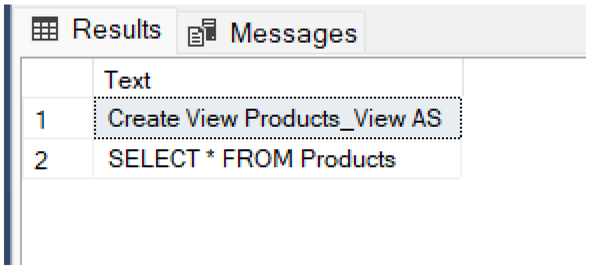


Figure: Products_View after renaming

Getting Information about a view:

We can retrieve all the information of a view using the Sp_Helptext system Stored Procedure. Let us see an example.

```
Sp_Helptext View_Products
```



	Text
1	Create View Products_View AS
2	SELECT * FROM Products

Figure: Output of the example.

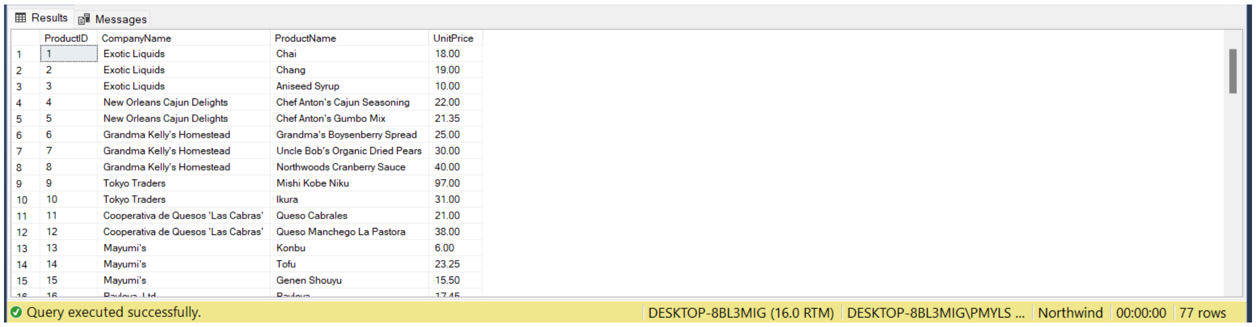
Alter View in SQL Server

We can alter the schema or structure of a view. In other words, we can add or remove some columns or change some conditions that are applied in a predefined view. Let us see an example.

```
Alter View Products_View_3 AS
SELECT ProductID, CompanyName, ProductName, UnitPrice
FROM Products
JOIN Suppliers ON Suppliers.SupplierID = Products.SupplierID
GO
```

Output:

```
SELECT * FROM Products_View_3
```



	ProductID	CompanyName	ProductName	UnitPrice
1	1	Exotic Liquids	Chai	18.00
2	2	Exotic Liquids	Chang	19.00
3	3	Exotic Liquids	Aniseed Syrup	10.00
4	4	New Orleans Cajun Delights	Chef Anton's Cajun Seasoning	22.00
5	5	New Orleans Cajun Delights	Chef Anton's Gumbo Mix	21.35
6	6	Grandma Kelly's Homestead	Grandma's Boysenberry Spread	25.00
7	7	Grandma Kelly's Homestead	Uncle Bob's Organic Dried Pears	30.00
8	8	Grandma Kelly's Homestead	Northwoods Cranberry Sauce	40.00
9	9	Tokyo Traders	Mishi Kobe Niku	97.00
10	10	Tokyo Traders	Ikura	31.00
11	11	Cooperativa de Quesos 'Las Cabras'	Queso Cabrales	21.00
12	12	Cooperativa de Quesos 'Las Cabras'	Queso Manchego La Pastora	38.00
13	13	Mayumi's	Konbu	6.00
14	14	Mayumi's	Tofu	23.25
15	15	Mayumi's	Genen Shouyu	15.50
16	16	Pavlova, Ltd.	Pavlova	17.45

```
Query executed successfully.    DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 77 rows
```

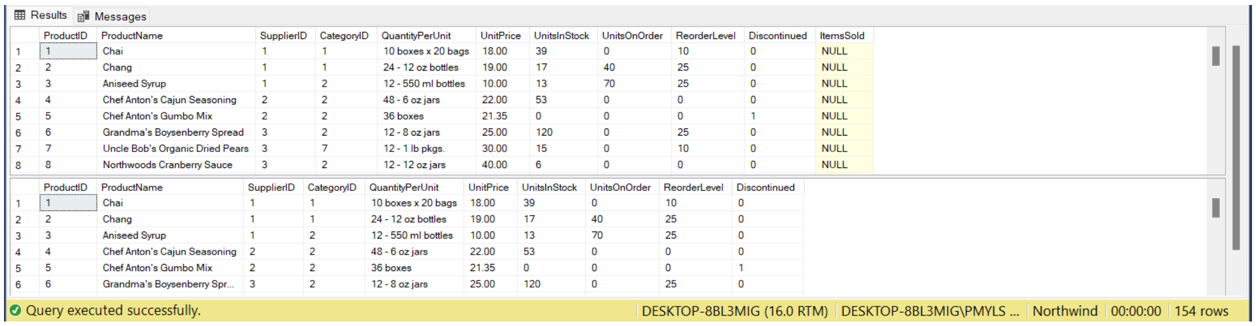
Refreshing a View in SQL Server:

Let us consider the scenario now by adding a new column to the table Products and examine the effect.

```
Alter Table Products Add ItemsSold nvarchar(50)
```

```
Select * from Products
```

```sql
Select * from View_Products
```



Results | Messages

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued	ItemsSold
1	1	Chai	1	1	10 boxes x 20 bags	18.00	39	0	10	0	NULL
2	2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	0	NULL
3	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00	13	70	25	0	NULL
4	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.00	53	0	0	0	NULL
5	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1	NULL
6	6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.00	120	0	25	0	NULL
7	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.00	15	0	10	0	NULL
8	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	0	NULL

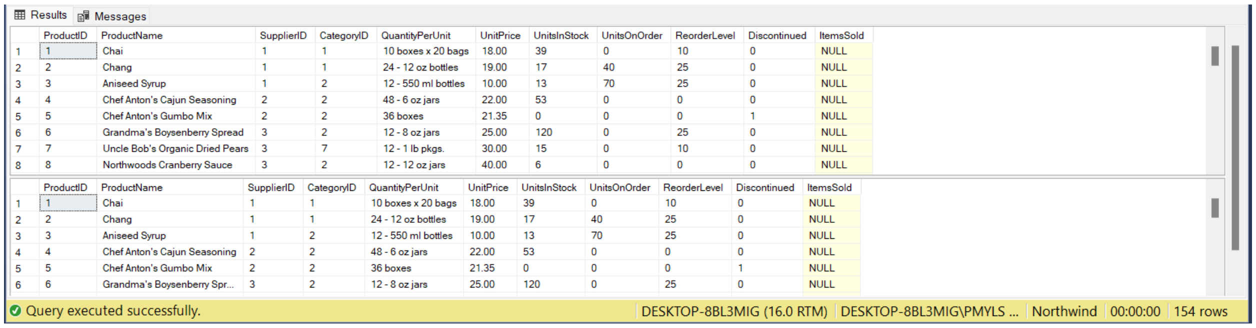
	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
1	1	Chai	1	1	10 boxes x 20 bags	18.00	39	0	10	0
2	2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	0
3	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00	13	70	25	0
4	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.00	53	0	0	0
5	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1
6	6	Grandma's Boysenberry Spr...	3	2	12 - 8 oz jars	25.00	120	0	25	0

Query executed successfully. | DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 154 rows

We don't get the results we expected because the schema of the view is already defined. So when we add a new column into the table it will not change the schema of the view and the view will contain the previous schema. For removing this problem, we use the system-defined Stored Procedure sp_refreshview.

sp_refreshview is a system-level Stored Procedure that refreshes the metadata of any view once you edit the schema of the table. Let's execute the following:

```sql
Exec sp_Refreshview View_Products
Select * from Products
Select * from View_Products
```



Results | Messages

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued	ItemsSold
1	1	Chai	1	1	10 boxes x 20 bags	18.00	39	0	10	0	NULL
2	2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	0	NULL
3	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00	13	70	25	0	NULL
4	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.00	53	0	0	0	NULL
5	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1	NULL
6	6	Grandma's Boysenberry Spread	3	2	12 - 8 oz jars	25.00	120	0	25	0	NULL
7	7	Uncle Bob's Organic Dried Pears	3	7	12 - 1 lb pkgs.	30.00	15	0	10	0	NULL
8	8	Northwoods Cranberry Sauce	3	2	12 - 12 oz jars	40.00	6	0	0	0	NULL

	ProductID	ProductName	SupplierID	CategoryID	QuantityPerUnit	UnitPrice	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued	ItemsSold
1	1	Chai	1	1	10 boxes x 20 bags	18.00	39	0	10	0	NULL
2	2	Chang	1	1	24 - 12 oz bottles	19.00	17	40	25	0	NULL
3	3	Aniseed Syrup	1	2	12 - 550 ml bottles	10.00	13	70	25	0	NULL
4	4	Chef Anton's Cajun Seasoning	2	2	48 - 6 oz jars	22.00	53	0	0	0	NULL
5	5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35	0	0	0	1	NULL
6	6	Grandma's Boysenberry Spr...	3	2	12 - 8 oz jars	25.00	120	0	25	0	NULL

Query executed successfully. | DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 154 rows

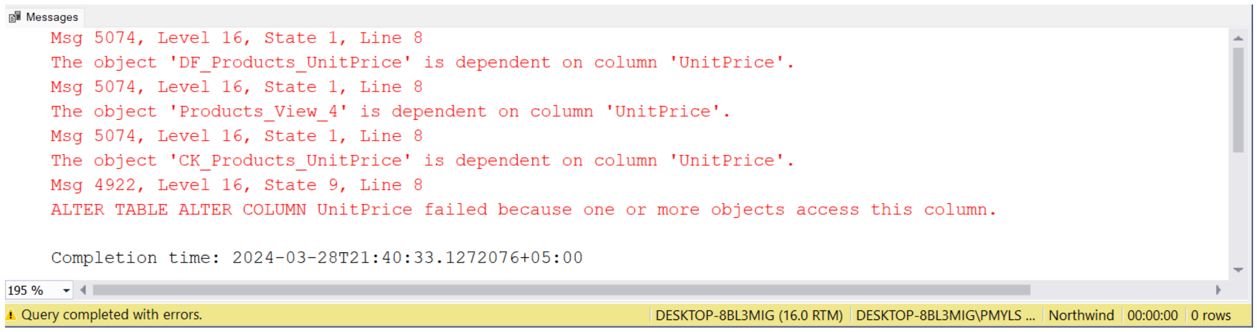
Schema Binding a VIEW

In the previous example, we saw that if we add a new column into the table then we must refresh the view.

Such a way if we change the data type of any column in a table then we should refresh the view. If we want to prevent any type of change in a base table then we can use the concept of SCHEMABINDING. It will lock the tables being referred to by the view and restrict all kinds of changes that may change the table schema (no Alter command).

```
Create View Products_View_4
WITH SCHEMABINDING
AS
SELECT ProductID,ProductName,UnitPrice
FROM dbo.Products
GO
```

In the preceding example, we create a view using Schema binding. Now we try to change the datatype of UnitPrice from money to int in the Base Table.


Messages

```
Msg 5074, Level 16, State 1, Line 8
The object 'DF_Products_UnitPrice' is dependent on column 'UnitPrice'.
Msg 5074, Level 16, State 1, Line 8
The object 'Products_View_4' is dependent on column 'UnitPrice'.
Msg 5074, Level 16, State 1, Line 8
The object 'CK_Products_UnitPrice' is dependent on column 'UnitPrice'.
Msg 4922, Level 16, State 9, Line 8
ALTER TABLE ALTER COLUMN UnitPrice failed because one or more objects access this column.

Completion time: 2024-03-28T21:40:33.1272076+05:00
```

195 %

Query completed with errors. | DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 0 rows

We find that we cannot change the data type because we used the SCHEMABINDING that prevents any type of change in the base table.

Encrypt a view in SQL Server:

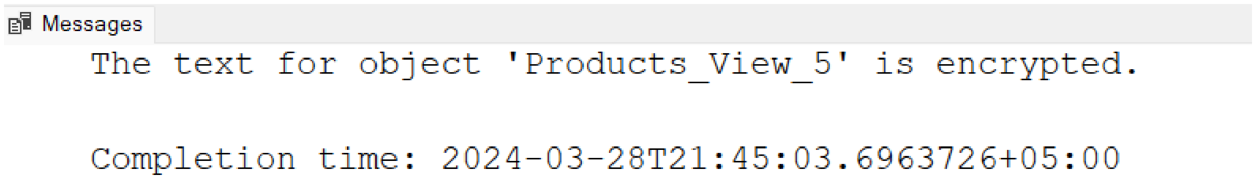
The “WITH ENCRYPTION” option can encrypt any views. That means it will not be visible via SP_HELPTEXT. This option encrypts the definition. This option encrypts the definition of the view. Users will not be able to see the definition of the view after it is created. This is the main advantage of the view where we can make it secure.

```
Create View Products_View_5
WITH Encryption
AS
SELECT ProductID,ProductName,UnitPrice
FROM Products
GO
```

Now we try to retrieve the definition of the view.

```
sp_Helptext Products_View_5
```

Output:


Messages

```
The text for object 'Products_View_5' is encrypted.

Completion time: 2024-03-28T21:45:03.6963726+05:00
```

Check Option:

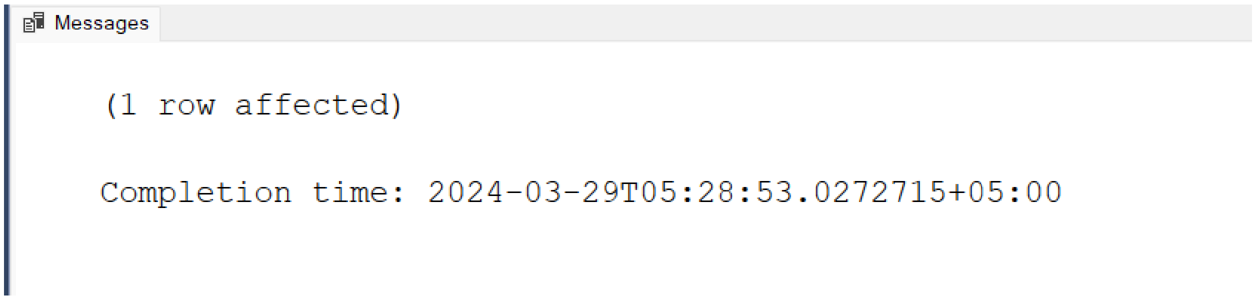
The use of the Check Option in a view is to ensure that all the Update and Insert commands must satisfy the condition in the view definition.

Let us see with an example.

```
Create View Products_View_6 AS
SELECT ProductID,ProductName,UnitPrice
FROM Products
WHERE UnitPrice < 250
GO
```

In the preceding example, we create a view that contains all the data for which UnitPrice < 250 but we can insert the data for a product having Unit price more than 250 as follows.

```
Insert Into Products_View_6 values ('Caramel',270)
```


Messages

```
(1 row affected)

Completion time: 2024-03-29T05:28:53.0272715+05:00
```

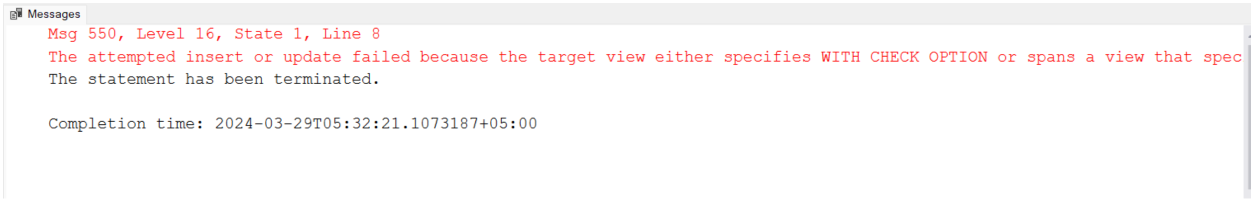
Now we drop the View and create it using Check option to prevent this issue as:

```
Create View Products_View_6 AS
SELECT ProductID,ProductName,UnitPrice
FROM Products
WHERE UnitPrice < 250
WITH Check Option
GO
```

Now if we try to execute the preceding query then it will throw an error such as:

```
Insert Into Products_View_6 values ('Caramel',270)
```

Output:



```
Messages
    Msg 550, Level 16, State 1, Line 8
    The attempted insert or update failed because the target view either specifies WITH CHECK OPTION or spans a view that spec
    The statement has been terminated.

    Completion time: 2024-03-29T05:32:21.1073187+05:00
```

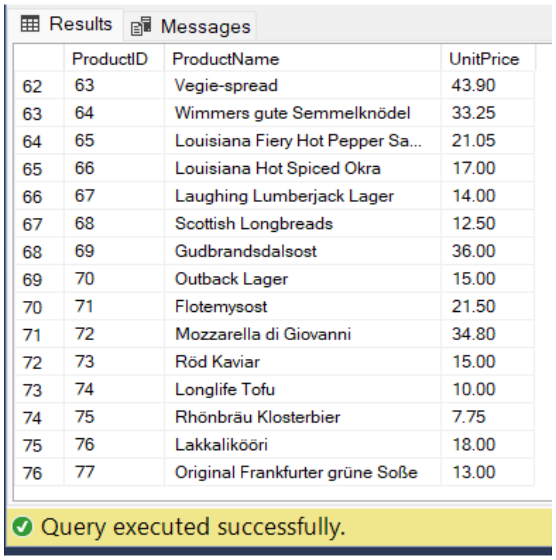
DML Query in View

In a view we can implement many types of DML query like insert, update and delete. But for a successful implementation of a DML query we should use some conditions like:

1. View should not contain multiple tables.
2. View should not contain set function.
3. View should not use the Distinct keyword.
4. View should not contain Group By, having clauses.
5. View should not contain Sub query.
6. View should not use Set Operators.
7. All NOT NULL columns from the base table must be included in the view in order for the INSERT query to function.

If we use the preceding conditions then we can implement a DML Query in the view without any problem. Let us see an example.

```
SELECT * FROM Products_View_6
```

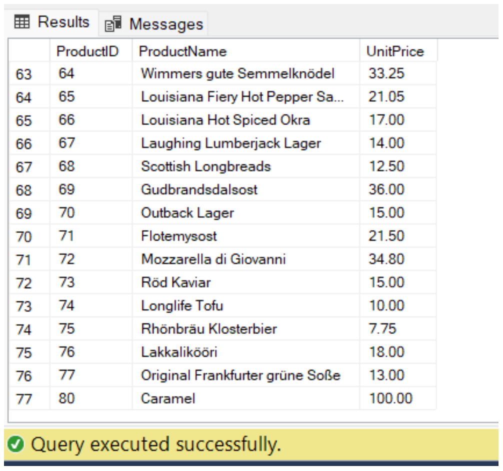


	ProductID	ProductName	UnitPrice
62	63	Vegie-spread	43.90
63	64	Wimmers gute Semmelknödel	33.25
64	65	Louisiana Fiery Hot Pepper Sa...	21.05
65	66	Louisiana Hot Spiced Okra	17.00
66	67	Laughing Lumberjack Lager	14.00
67	68	Scottish Longbreads	12.50
68	69	Gudbrandsdalsost	36.00
69	70	Outback Lager	15.00
70	71	Flotemysost	21.50
71	72	Mozzarella di Giovanni	34.80
72	73	Röd Kaviar	15.00
73	74	Longlife Tofu	10.00
74	75	Rhönbräu Klosterbier	7.75
75	76	Lakkalikööri	18.00
76	77	Original Frankfurter grüne Soße	13.00

Query executed successfully.

Now we implement a DML Query as in the following:

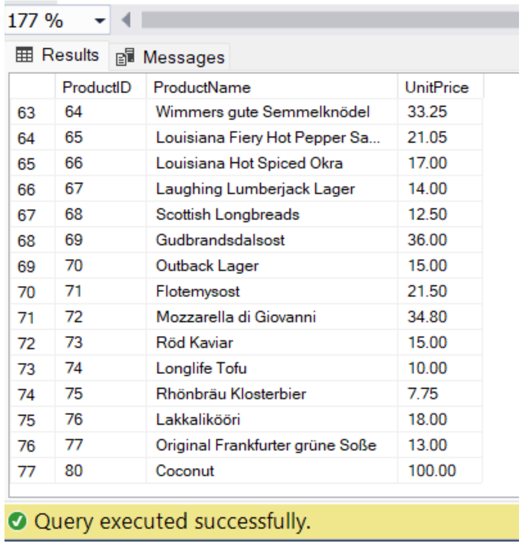
1. `Insert Into Products_View_6 values ('Caramel',270)`



	ProductID	ProductName	UnitPrice
63	64	Wimmers gute Semmelknödel	33.25
64	65	Louisiana Fiery Hot Pepper Sa...	21.05
65	66	Louisiana Hot Spiced Okra	17.00
66	67	Laughing Lumberjack Lager	14.00
67	68	Scottish Longbreads	12.50
68	69	Gudbrandsdalsost	36.00
69	70	Outback Lager	15.00
70	71	Flotemysost	21.50
71	72	Mozzarella di Giovanni	34.80
72	73	Röd Kaviar	15.00
73	74	Longlife Tofu	10.00
74	75	Rhönbräu Klosterbier	7.75
75	76	Lakkalikööri	18.00
76	77	Original Frankfurter grüne Soße	13.00
77	80	Caramel	100.00

Query executed successfully.

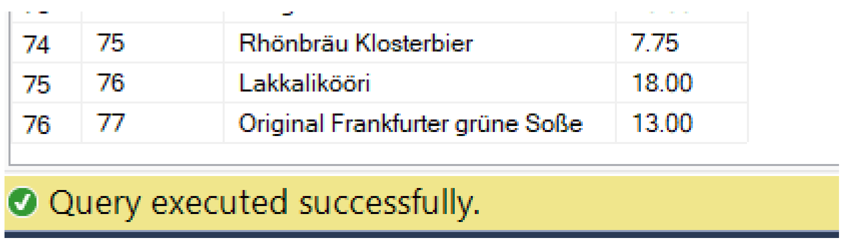
2. `Update Products_View_6 SET ProductName = 'Coconut' WHERE ProductId = 80;`



	ProductID	ProductName	UnitPrice
63	64	Wimmers gute Semmelknödel	33.25
64	65	Louisiana Fiery Hot Pepper Sa...	21.05
65	66	Louisiana Hot Spiced Okra	17.00
66	67	Laughing Lumberjack Lager	14.00
67	68	Scottish Longbreads	12.50
68	69	Gudbrandsdalsost	36.00
69	70	Outback Lager	15.00
70	71	Flotemysost	21.50
71	72	Mozzarella di Giovanni	34.80
72	73	Röd Kaviar	15.00
73	74	Longlife Tofu	10.00
74	75	Rhönbräu Klosterbier	7.75
75	76	Lakkalikööri	18.00
76	77	Original Frankfurter grüne Soße	13.00
77	80	Coconut	100.00

Query executed successfully.

3. `DELETE FROM Products_View_6 WHERE ProductId = 80`

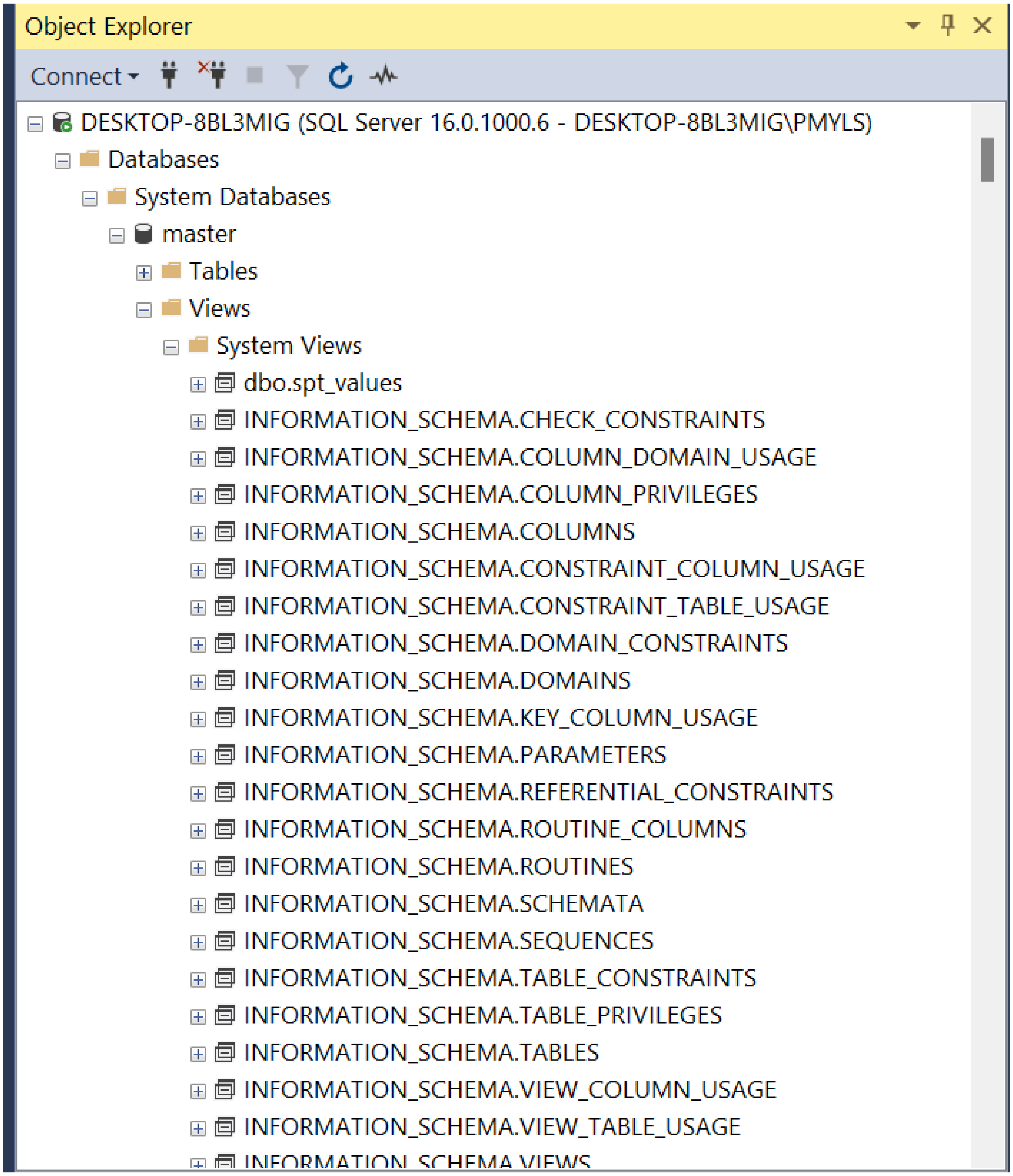


74	75	Rhönbräu Klosterbier	7.75
75	76	Lakkalikööri	18.00
76	77	Original Frankfurter grüne Soße	13.00

Query executed successfully.

System Define Views:

SQL Server also contains various predefined databases like Tempdb, Master, temp. Each database has their own properties and responsibility. Master data is a template database for all other user-defined databases. A Master database contains many Predefine_View that work as templates for other databases and tables. Master databases contain nearly 230 predefined views.



```
DESKTOP-8BL3MIG (SQL Server 16.0.1000.6 - DESKTOP-8BL3MIG\PMYLS)
  Databases
    System Databases
      master
        Tables
        Views
          System Views
            dbo.spt_values
            INFORMATION_SCHEMA.CHECK_CONSTRAINTS
            INFORMATION_SCHEMA.COLUMN_DOMAIN_USAGE
            INFORMATION_SCHEMA.COLUMN_PRIVILEGES
            INFORMATION_SCHEMA.COLUMNS
            INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE
            INFORMATION_SCHEMA.CONSTRAINT_TABLE_USAGE
            INFORMATION_SCHEMA.DOMAIN_CONSTRAINTS
            INFORMATION_SCHEMA.DOMAINS
            INFORMATION_SCHEMA.KEY_COLUMN_USAGE
            INFORMATION_SCHEMA.PARAMETERS
            INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
            INFORMATION_SCHEMA.ROUTINE_COLUMNS
            INFORMATION_SCHEMA.ROUTINES
            INFORMATION_SCHEMA.SCHEMATA
            INFORMATION_SCHEMA.SEQUENCES
            INFORMATION_SCHEMA.TABLE_CONSTRAINTS
            INFORMATION_SCHEMA.TABLE_PRIVILEGES
            INFORMATION_SCHEMA.TABLES
            INFORMATION_SCHEMA.VIEW_COLUMN_USAGE
            INFORMATION_SCHEMA.VIEW_TABLE_USAGE
            INFORMATION_SCHEMA.VIEWS
```

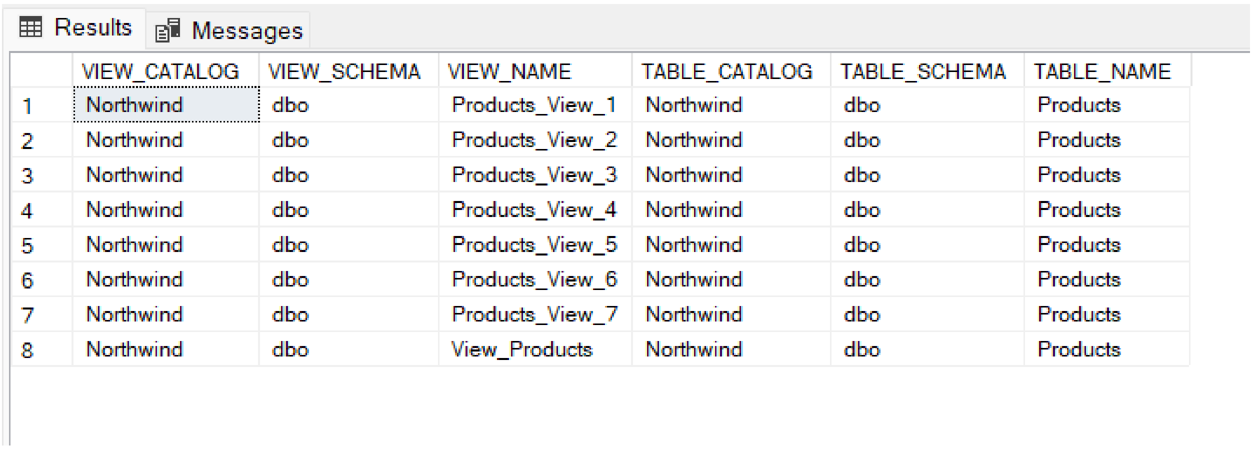
These predefined views are very useful to us. Mainly we divide system views into the following two parts.

1. Information Schema
2. Catalog View

Information schema: There are nearly 21 Information Schemas in the System. These are used for displaying the most physical information of a database, such as table and columns. An Information Schema starts from INFORMATION_SCHEMA.[View Name]. Let us see an example.

```
select * from INFORMATION_SCHEMA.VIEW_TABLE_USAGE
where TABLE_NAME='Products'
```

Output:

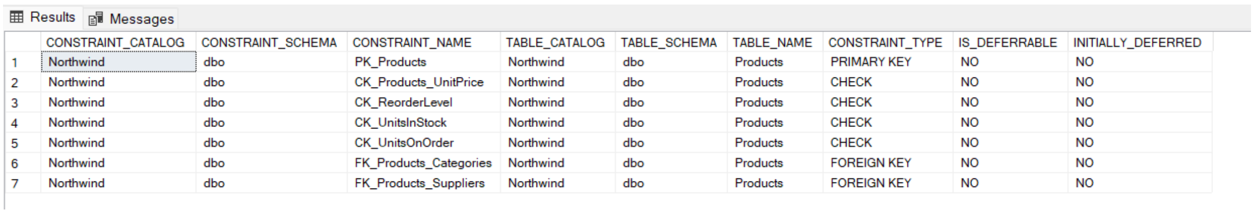


	VIEW_CATALOG	VIEW_SCHEMA	VIEW_NAME	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME
1	Northwind	dbo	Products_View_1	Northwind	dbo	Products
2	Northwind	dbo	Products_View_2	Northwind	dbo	Products
3	Northwind	dbo	Products_View_3	Northwind	dbo	Products
4	Northwind	dbo	Products_View_4	Northwind	dbo	Products
5	Northwind	dbo	Products_View_5	Northwind	dbo	Products
6	Northwind	dbo	Products_View_6	Northwind	dbo	Products
7	Northwind	dbo	Products_View_7	Northwind	dbo	Products
8	Northwind	dbo	View_Products	Northwind	dbo	Products

This Information_Schema returns the details of all the views used by table Products.

```
select * from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
where TABLE_NAME='Products'
```

Output:

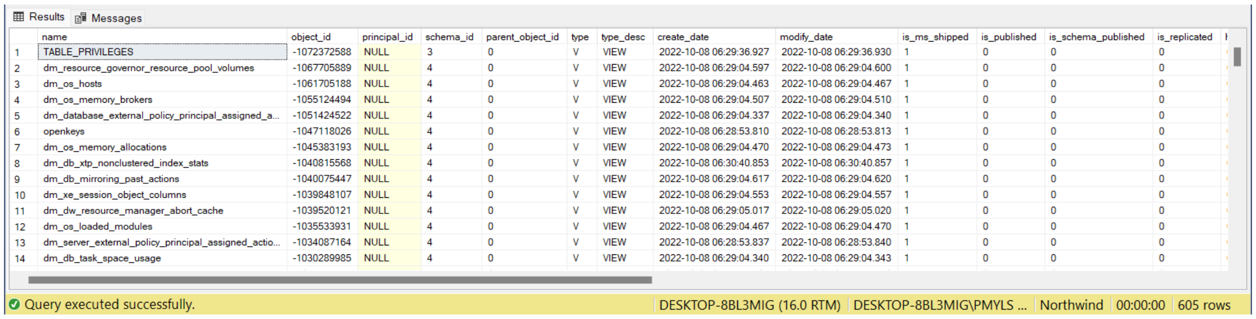


	CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE	IS_DEFERRABLE	INITIALLY_DEFERRED
1	Northwind	dbo	PK_Products	Northwind	dbo	Products	PRIMARY KEY	NO	NO
2	Northwind	dbo	CK_Products_UnitPrice	Northwind	dbo	Products	CHECK	NO	NO
3	Northwind	dbo	CK_ReorderLevel	Northwind	dbo	Products	CHECK	NO	NO
4	Northwind	dbo	CK_UnitsInStock	Northwind	dbo	Products	CHECK	NO	NO
5	Northwind	dbo	CK_UnitsOnOrder	Northwind	dbo	Products	CHECK	NO	NO
6	Northwind	dbo	FK_Products_Categories	Northwind	dbo	Products	FOREIGN KEY	NO	NO
7	Northwind	dbo	FK_Products_Suppliers	Northwind	dbo	Products	FOREIGN KEY	NO	NO

This **Information_Schema** returns the information about the constraints of a table.

Catalog View: Catalog Views are categorized into various groups also. These are used to show the self-describing information of a database. These start with **"sys"**.

```
select * from sys.all_views
```



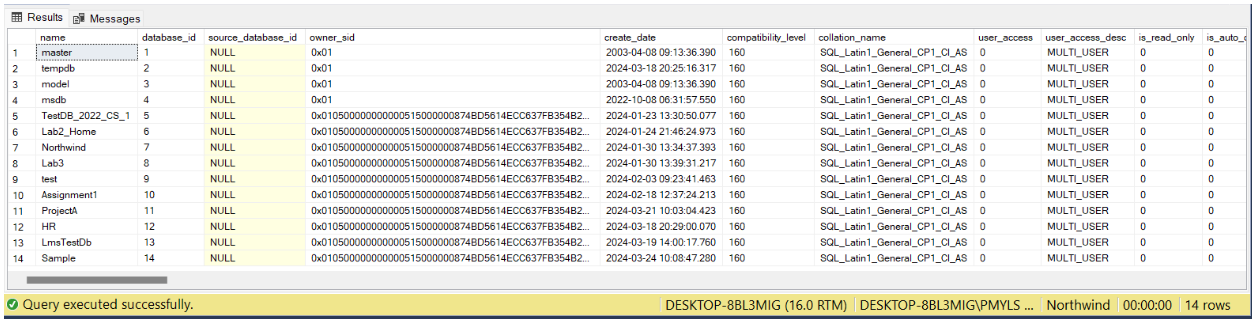
Results | Messages

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date	is_ms_shipped	is_published	is_schema_published	is_replicated
1	TABLE_PRIVILEGES	-1072372588	NULL	3	0	V	VIEW	2022-10-08 06:29:36.927	2022-10-08 06:29:36.930	1	0	0	0
2	dm_resource_governor_resource_pool_volumes	-1067705889	NULL	4	0	V	VIEW	2022-10-08 06:29:04.597	2022-10-08 06:29:04.600	1	0	0	0
3	dm_os_hosts	-1061705188	NULL	4	0	V	VIEW	2022-10-08 06:29:04.463	2022-10-08 06:29:04.467	1	0	0	0
4	dm_os_memory_brokers	-1055124494	NULL	4	0	V	VIEW	2022-10-08 06:29:04.507	2022-10-08 06:29:04.510	1	0	0	0
5	dm_database_external_policy_principal_assigned_a...	-1051424522	NULL	4	0	V	VIEW	2022-10-08 06:29:04.337	2022-10-08 06:29:04.340	1	0	0	0
6	openkeys	-1047118026	NULL	4	0	V	VIEW	2022-10-08 06:28:53.810	2022-10-08 06:28:53.813	1	0	0	0
7	dm_os_memory_allocations	-1045383193	NULL	4	0	V	VIEW	2022-10-08 06:29:04.470	2022-10-08 06:29:04.473	1	0	0	0
8	dm_db_xtp_nonclustered_index_stats	-1040815568	NULL	4	0	V	VIEW	2022-10-08 06:30:40.853	2022-10-08 06:30:40.857	1	0	0	0
9	dm_db_mirroring_past_actions	-1040075447	NULL	4	0	V	VIEW	2022-10-08 06:29:04.617	2022-10-08 06:29:04.620	1	0	0	0
10	dm_xe_session_object_columns	-1039848107	NULL	4	0	V	VIEW	2022-10-08 06:29:04.553	2022-10-08 06:29:04.557	1	0	0	0
11	dm_dw_resource_manager_abort_cache	-1039520121	NULL	4	0	V	VIEW	2022-10-08 06:29:05.017	2022-10-08 06:29:05.020	1	0	0	0
12	dm_os_loaded_modules	-1035533931	NULL	4	0	V	VIEW	2022-10-08 06:29:04.467	2022-10-08 06:29:04.470	1	0	0	0
13	dm_server_external_policy_principal_assigned_actio...	-1034087164	NULL	4	0	V	VIEW	2022-10-08 06:28:53.837	2022-10-08 06:28:53.840	1	0	0	0
14	dm_db_task_space_usage	-1030289985	NULL	4	0	V	VIEW	2022-10-08 06:29:04.340	2022-10-08 06:29:04.343	1	0	0	0

Query executed successfully. | DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 605 rows

This query provides information to all types of views using a database.

```
select * from sys.databases
```



Results | Messages

	name	database_id	source_database_id	owner_sid	create_date	compatibility_level	collation_name	user_access	user_access_desc	is_read_only	is_auto_
1	master	1	NULL	0x01	2003-04-08 09:13:36.390	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
2	tempdb	2	NULL	0x01	2024-03-18 20:25:16.317	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
3	model	3	NULL	0x01	2003-04-08 09:13:36.390	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
4	msdb	4	NULL	0x01	2022-10-08 06:31:57.550	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
5	TestDB_2022_CS_1	5	NULL	0x010500000000000515000000874BD5614ECC637FB354B2...	2024-01-23 13:30:50.077	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
6	Lab2_Home	6	NULL	0x010500000000000515000000874BD5614ECC637FB354B2...	2024-01-24 21:46:24.973	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
7	Northwind	7	NULL	0x010500000000000515000000874BD5614ECC637FB354B2...	2024-01-30 13:34:37.393	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
8	Lab3	8	NULL	0x010500000000000515000000874BD5614ECC637FB354B2...	2024-01-30 13:39:31.217	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
9	test	9	NULL	0x010500000000000515000000874BD5614ECC637FB354B2...	2024-02-03 09:23:41.463	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
10	Assignment1	10	NULL	0x010500000000000515000000874BD5614ECC637FB354B2...	2024-02-18 12:37:24.213	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
11	ProjectA	11	NULL	0x010500000000000515000000874BD5614ECC637FB354B2...	2024-03-21 10:03:04.423	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
12	HR	12	NULL	0x010500000000000515000000874BD5614ECC637FB354B2...	2024-03-18 20:29:00.070	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
13	LmsTestDb	13	NULL	0x010500000000000515000000874BD5614ECC637FB354B2...	2024-03-19 14:00:17.760	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0
14	Sample	14	NULL	0x010500000000000515000000874BD5614ECC637FB354B2...	2024-03-24 10:08:47.280	160	SQL_Latin1_General_CP1_CI_AS	0	MULTI_USER	0	0

Query executed successfully. | DESKTOP-8BL3MIG (16.0 RTM) | DESKTOP-8BL3MIG\PMYLS ... | Northwind | 00:00:00 | 14 rows

This query will provide the information about all the databases defined by the system, including user-defined and system defined database.