SQL View – A complete introduction and walk-through

July 1, 2019 by Rajendra Gupta

In relational databases, data is structured using various database objects like tables, stored procedure, views, clusters etc. This article aims to walk you through 'SQL VIEW' – one of the widely-used database objects in SQL Server.

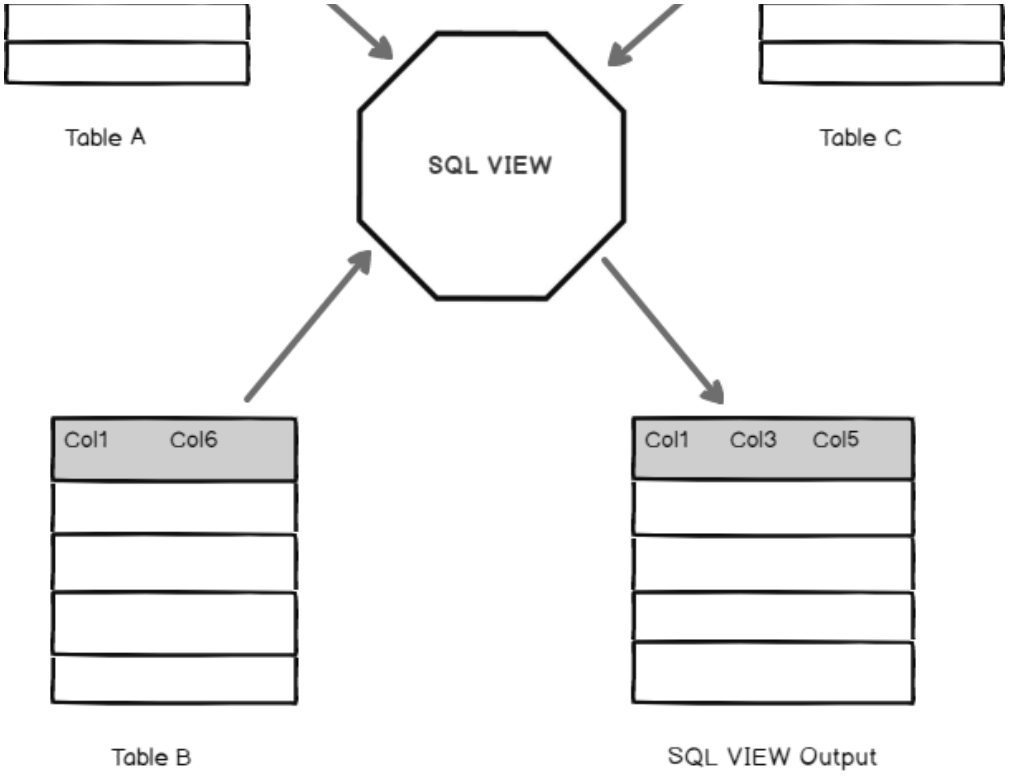
It is a good practice to organize tables in a database to reduce redundancy and dependency in SQL database. Normalization is a database process for organizing the data in the database by splitting large tables into smaller tables. These multiple tables are linked using the relationships. Developers write queries to retrieve data from multiple tables and columns. In the query, we might use multiple joins and queries could become complicated and overwhelming to understand. Users should also require permissions on individual objects to fetch the data.

Let's go ahead and see how SQL VIEW help to resolve these issues in SQL Server.

Introduction

A VIEW in SQL Server is like a virtual table that contains data from one or multiple tables. It does not hold any data and does not exist physically in the database. Similar to a SQL table, the view name should be unique in a database. It contains a set of predefined SQL queries to fetch data from the database. It can contain database tables from single or multiple databases as well.

In the following image, you can see the VIEW contains a query to join three relational tables and fetch the data in a virtual table.



A VIEW does not require any storage in a database because it does not exist physically. In a VIEW, we can also control user security for accessing the data from the database tables. We can allow users to get the data from the VIEW, and the user does not require permission for each table or column to fetch data.

Let's explore user-defined VIEW in SQL Server.

Note: In this article, I am going to use sample database **AdventureWorks** for all examples.

Create a SQL VIEW

The syntax to create a VIEW is as follows:

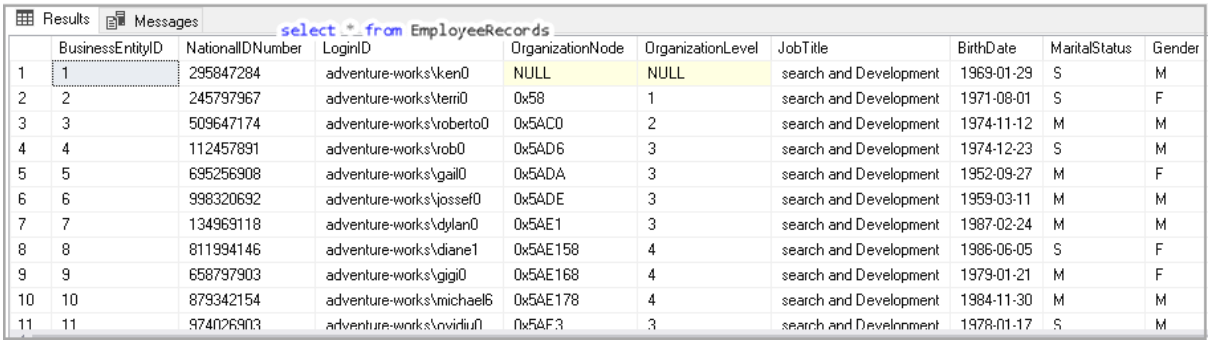
```
CREATE VIEW Name AS
Select column1, Column2...Column N From tables
Where conditions;
```

Example 1: SQL VIEW to fetch all records of a table

It is the simplest form of a VIEW. Usually, we do not use a VIEW in SQL Server to fetch all records from a single table.

```
CREATE VIEW EmployeeRecords
AS
     SELECT *
     FROM [HumanResources].[Employee];
```

Once a VIEW is created, you can access it like a SQL table.



Results Messages

```
select * from EmployeeRecords
```

	BusinessEntityID	NationalIDNumber	LoginID	OrganizationNode	OrganizationLevel	JobTitle	BirthDate	MaritalStatus	Gender
1	1	295847284	adventure-works\ken0	NULL	NULL	search and Development	1969-01-29	S	M
2	2	245797967	adventure-works\terri0	0x58	1	search and Development	1971-08-01	S	F
3	3	509647174	adventure-works\roberto0	0x5AC0	2	search and Development	1974-11-12	M	M
4	4	112457891	adventure-works\rob0	0x5AD6	3	search and Development	1974-12-23	S	M
5	5	695256908	adventure-works\gail0	0x5ADA	3	search and Development	1952-09-27	M	F
6	6	998320692	adventure-works\jossef0	0x5ADE	3	search and Development	1959-03-11	M	M
7	7	134969118	adventure-works\dylan0	0x5AE1	3	search and Development	1987-02-24	M	M
8	8	811994146	adventure-works\diane1	0x5AE158	4	search and Development	1986-06-05	S	F
9	9	658797903	adventure-works\gigi0	0x5AE168	4	search and Development	1979-01-21	M	F
10	10	879342154	adventure-works\michael6	0x5AE178	4	search and Development	1984-11-30	M	M
11	11	974026903	adventure-works\ovidiu0	0x5AE3	3	search and Development	1978-01-17	S	M

Example 2: SQL VIEW to fetch a few columns of a table

We might not be interested in all columns of a table. We can specify required column names in the select statement to fetch those fields only from the table.

```
CREATE VIEW EmployeeRecords
AS
     SELECT NationalIDNumber,LoginID,JobTitle
     FROM [HumanResources].[Employee];
```

Example 3: SQL VIEW to fetch a few columns of a table and filter results using WHERE clause

We can filter the results using a Where clause condition in a Select statement. Suppose we want to get EmployeeRecords with Martial status 'M'.

```
CREATE VIEW EmployeeRecords
AS
     SELECT NationalIDNumber,
            LoginID,
            JobTitle,
            MaritalStatus
     FROM [HumanResources].[Employee]
     WHERE MaritalStatus = 'M';
```

Example 4: SQL VIEW to fetch records from multiple tables

We can use VIEW to have a select statement with Join condition between multiple tables. It is one of the frequent uses of a VIEW in SQL Server.

In the following query, we use INNER JOIN and LEFT OUTER JOIN between multiple tables to fetch a few columns as per our requirement.

```
CREATE VIEW [Sales].[vStoreWithContacts]
```

```
AS
     SELECT s.[BusinessEntityID],
            s.[Name],
            ct.[Name] AS [ContactType],
            p.[Title],
            p.[FirstName],
            p.[MiddleName],
            p.[LastName],
            p.[Suffix],
            pp.[PhoneNumber],
            ea.[EmailAddress],
            p.[EmailPromotion]
     FROM [Sales].[Store] s
          INNER JOIN [Person].[BusinessEntityContact] bec ON bec.[BusinessEntityID
] = s.[BusinessEntityID]
          INNER JOIN [Person].[ContactType] ct ON ct.[ContactTypeID] = bec.[Con-
tactTypeID]
          INNER JOIN [Person].[Person] p ON p.[BusinessEntityID] = bec.[PersonID]
          LEFT OUTER JOIN [Person].[EmailAddress] ea ON ea.[BusinessEntityID] = p.
[BusinessEntityID]
          LEFT OUTER JOIN [Person].[PersonPhone] pp ON pp.[BusinessEntityID] = p.[
BusinessEntityID];
GO
```

Suppose you need to execute this query very frequently. Using a VIEW, we can simply get the data with a single line of code.

```
select * from  [Sales].[vStoreWithContacts]
```

	BusinessEntityID	Name	ContactType	Title	FirstName	MiddleName	LastName
1	292	Next-Door Bike Store	Owner	Mr.	Gustavo	NULL	Achong
2	294	Professional Sales and Service	Owner	Ms.	Catherine	R.	Abel
3	296	Riders Company	Owner	Ms.	Kim	NULL	Abercrombie
4	298	The Bike Mechanics	Owner	Sr.	Humberto	NULL	Acevedo
5	300	Nationwide Supply	Owner	Sra.	Pilar	NULL	Ackerman
6	302	Area Bike Accessories	Owner	Ms.	Frances	B.	Adams
7	304	Bicycle Accessories and Kits	Owner	Ms.	Margaret	J.	Smith
8	306	Clamps & Brackets Co.	Owner	Ms.	Carla	J.	Adams
9	316	Fun Toys and Bikes	Owner	Mr.	Robert	E.	Ahlering
10	318	Great Bikes	Owner	Mr.	François	NULL	Ferrier
11	320	Metropolitan Sales and Rental	Owner	Ms.	Kim	NULL	Akers

Example 5: SQL VIEW to fetch specific column

In the previous example, we created a VIEW with multiple tables and a few column from those tables. Once we have a view, it is not required to fetch all columns from the view. We can select few columns as well from a VIEW in SQL Server similar to a relational table.

In the following query, we want to get only two columns name and contract type from the view.

```
SELECT Name,
       ContactType
FROM [Sales].[vStoreWithContacts];
```

Example 6: Use Sp_helptext to retrieve VIEW definition

Example 6: Use Sp_helptext to retrieve VIEW definition

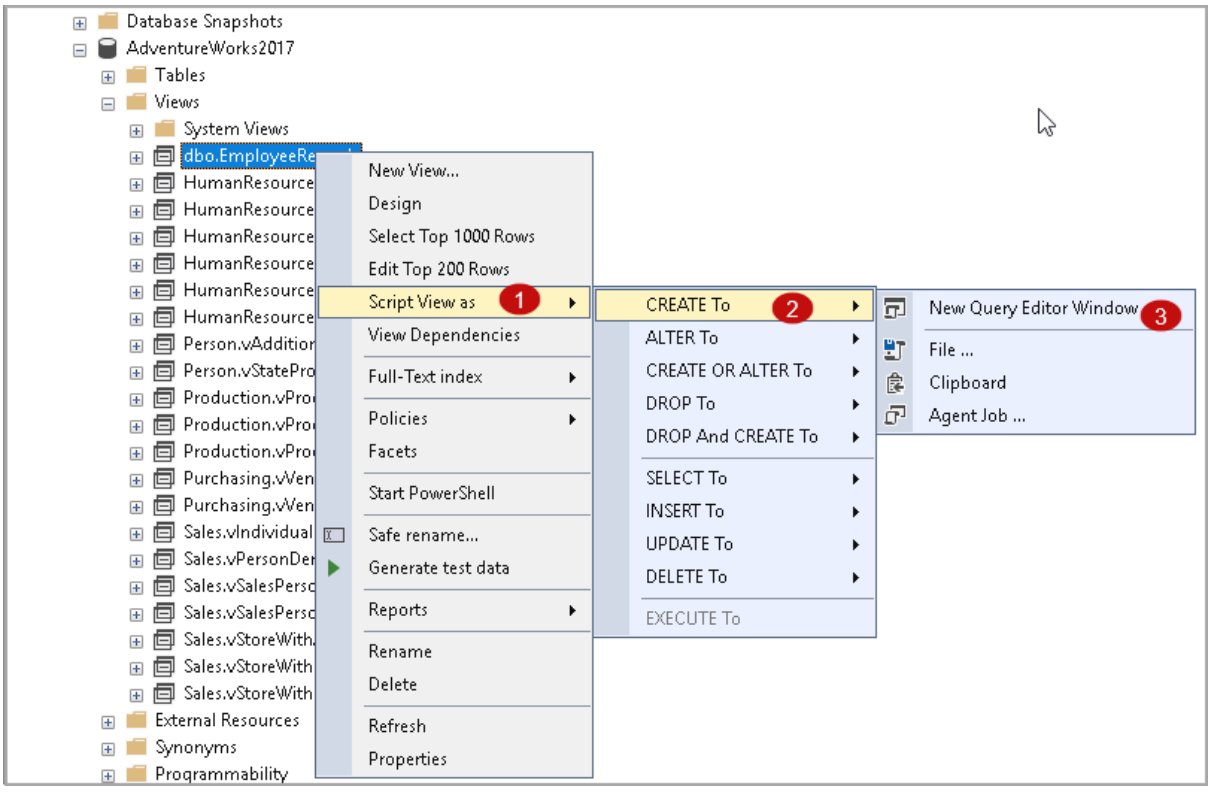
We can use sp_helptext system stored procedure to get VIEW definition. It returns the complete definition of a SQL VIEW.

For example, let's check the view definition for EmployeeRecords VIEW.

```
Text
----------------------------------------------------------------------
CREATE VIEW EmployeeRecords
AS
     SELECT NationalIDNumber,
            LoginID,
            JobTitle,
            MaritalStatus
     FROM [HumanResources].[Employee]
     WHERE MaritalStatus = 'M';
```

Sp_helptext 'EmployeeRecords'

We can use SSMS as well to generate the script for a VIEW. Expand database -> Views -> Right click and go to Script view as -> Create To -> New Query Editor Window.



Example 7: sp_refreshview to update the Metadata of a SQL VIEW

Suppose we have a VIEW on a table that specifies select * statement to get all columns of that table.

```
CREATE VIEW DemoView
```

```
CREATE VIEW DemoView
AS
     SELECT *
     FROM [AdventureWorks2017].[dbo].[MyTable];
```

Once we call the VIEW DemoView, it gives the following output.

	TableID	ForeignID	Value	CodeOne	CodeTwo
1	1	1	FooBar	Foo	Bar

Let's add a new column in the table using the Alter table statement.

```
Alter Table [AdventureWorks2017].[dbo].[MyTable] Add City nvarchar(50)
```

Rerun the select statement to get records from VIEW. It should display the new column as well in the output. We still get the same output, and it does not contain the newly added column.

	TableID	ForeignID	Value	CodeOne	CodeTwo
1	1	1	FooBar	Foo	Bar

By Default, SQL Server does not modify the schema and metadata for the VIEW. We can use the system stored procedure **sp_refreshview** to refresh the metadata of any view.

```
Exec sp_refreshview DemoView
```

Rerun the select statement to get records from VIEW. We can see the City column in the output.

	TableID	ForeignID	Value	CodeOne	CodeTwo	City
1	1	1	FooBar	Foo	Bar	NULL

After executing Exec sp_refreshview DemoView

Example 8: Schema Binding a SQL VIEW

In the previous example, we modify the SQL table to add a new column. Suppose in the production instance, and you have a view in the application. You are not aware of the changes in the table design for the new column. We do not want any changes to be made in the tables being used in the VIEW. We can use SCHEMABINDING option to lock all tables used in the VIEW and deny any alter table statement against those tables.

Let's execute the following query with an option SCHEMABINDING.

```
CREATE VIEW DemoView
WITH SCHEMABINDING
AS
```

```
SELECT *
FROM [AdventureWorks2017].[dbo].[MyTable];
```

It gives an error message.

Msg 1054, Level 15, State 6, Procedure DemoView, Line 4 [Batch Start Line 2]
Syntax '*' is not allowed in schema-bound objects.

We cannot call all columns (Select *) in a VIEW with SCHEMABINDING option. Let's specify the columns in the following query and execute it again.

```
CREATE VIEW DemoView
WITH SCHEMABINDING
AS
     SELECT TableID, ForeignID ,Value, CodeOne
     FROM [AdventureWorks2017].[dbo].[MyTable];
```

We again get the following error message.

Msg 4512, Level 16, State 3, Procedure DemoView, Line 5 [Batch Start Line 1]
Cannot schema bind VIEW 'DemoView' because of the name 'AdventureWorks2017.dbo.MyTable' is invalid for schema binding.
Names must be in a two-part format, and an object cannot reference itself.

In my query, I used a three-part object name in the format [DBName.Schema.Object]. We cannot use this format with SCHEMABINDING option in a VIEW. We can use the two-part name as per the following query.

```
CREATE VIEW DemoView
WITH SCHEMABINDING
AS
     SELECT TableID, ForeignID ,Value, CodeOne
     FROM [dbo].[MyTable];
```

Once you have created a VIEW with SCHEMABINDING option, try to add a modify a column data type using Alter table command.

```
ALTER TABLE dbo.MyTable ALTER COLUMN ForeignID BIGINT;
```

```
Msg 5074, Level 16, State 1, Line 12
The object 'DemoView' is dependent on column 'ForeignID'.
Msg 4922, Level 16, State 9, Line 12
ALTER TABLE ALTER COLUMN ForeignID failed because one or more objects access this column.
```

We need to drop the VIEW definition itself along with other dependencies on that table before making a change to the existing table schema.

Example 8: SQL VIEW ENCRYPTION

We can encrypt the VIEW using the WITH ENCRYPTION clause. Previously, we checked users can see the view definition using the sp_helptext command. If we do not want users to view the definition, we can encrypt it.

```
CREATE VIEW DemoView
WITH ENCRYPTION
AS
     SELECT TableID, ForeignID ,Value, CodeOne
     FROM [dbo].[MyTable];
```

Now if you run the sp_helptext command to view the definition, you get the following error message.

```
Exec sp_helptext DemoView
```

The text for the object 'DemoView' is encrypted.

Example 9: SQL VIEW for DML (Update, Delete and Insert) queries

We can use SQL VIEW to insert, update and delete data in a single SQL table. We need to note the following things regarding this.

1. We can use DML operation on a single table only
2. VIEW should not contain Group By, Having, Distinct clauses
3. We cannot use a subquery in a VIEW in SQL Server
4. We cannot use Set operators in a SQL VIEW

Use the following queries to perform DML operation using VIEW in SQL Server.

- Insert DML

```
Insert into DemoView values(4,'CC','KK','RR')
```

- Delete DML

```
Delete from DemoView where TableID=7
```

- Update DML

```
Update DemoView set value='Raj' where TableID=5
```

Example 10: SQL VIEW and Check Option

We can use WITH CHECK option to check the conditions in VIEW are inline with the DML statements.

- It prevents to insert rows in the table where the condition in the Where clause is not satisfied
- If the condition does not satisfy, we get an error message in the insert or update statement

In the following query, we use the CHECK option, and we want only values starting with letter F in the

[Codeone] column.

```
CREATE VIEW DemoView
AS
     SELECT *
     FROM [dbo].[MyTable]
     WHERE [Codeone] LIKE 'F%'
WITH CHECK OPTION;
```

If we try to insert a value that does not match the condition, we get the following error message.

```
Insert into DemoView values (5,'CC','Raj','Raj')
```

```
  Msg 550, Level 16, State 1, Line 1
  The attempted insert or update failed because the target view either specifies
   WITH CHECK OPTION or spans a view that specifies WITH CHECK OPTION and one or more rows
   resulting from the operation did not qualify under the CHECK OPTION constraint.
  The statement has been terminated.
```

Example 11: Drop SQL VIEW

We can drop a VIEW using the DROP VIEW statement. In the following query, we want to drop the VIEW **demoview** in SQL Server.

```
DROP VIEW demoview;
```

Example 12: Alter a SQL VIEW

We can change the SQL statement in a VIEW using the following alter VIEW command. Suppose we want to change the condition in the where clause of a VIEW. Execute the following query.

```
Alter VIEW DemoView
AS
     SELECT *
     FROM [dbo].[MyTable]
     WHERE [Codeone] LIKE 'C%'
WITH CHECK OPTION;
```

Starting from SQL Server 2016 SP1, we can use the CREATE or ALTER statement to create a SQL VIEW or modify it if already exists. Prior to SQL Server 2016 SP1, we cannot use both CREATE or Alter together.

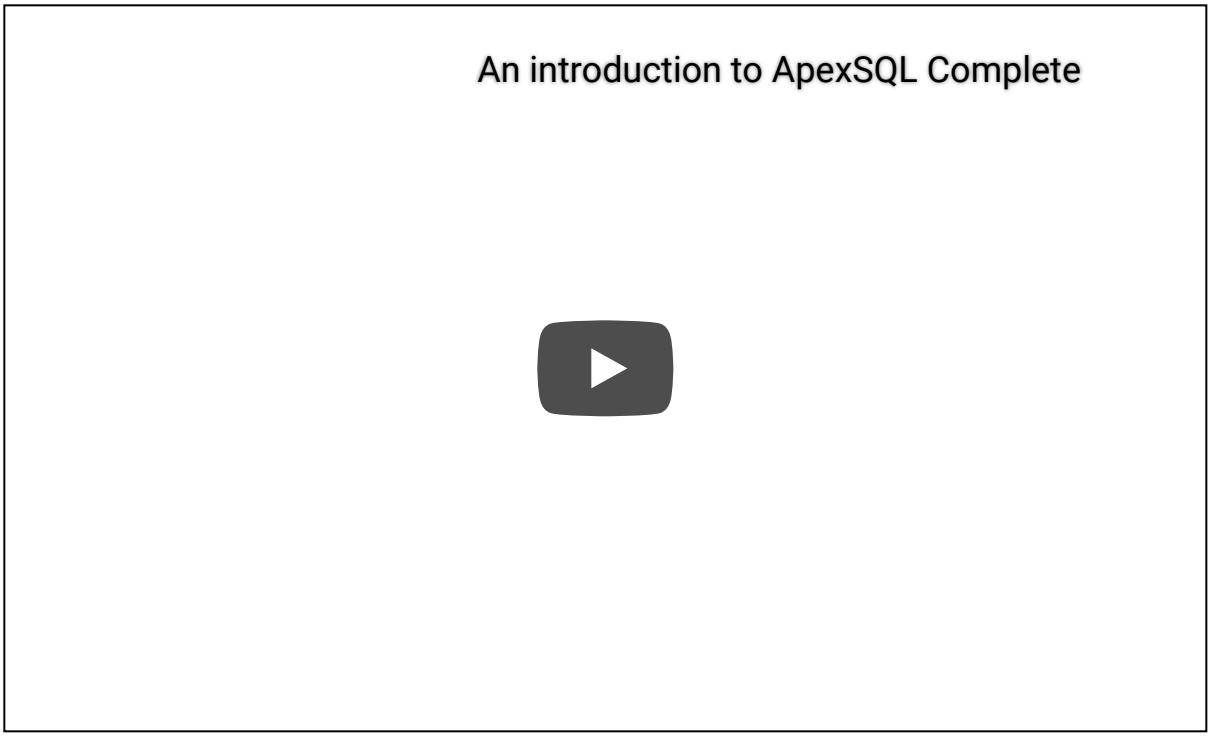
```
CREATE OR ALTER VIEW DemoView
AS SELECT *
   FROM [dbo].[MyTable]
   WHERE [Codeone] LIKE 'C%'
WITH CHECK OPTION;
```

Conclusion

In this article, we explored SQL View with various examples. You should be familiar with the view in SQL Server as a developer or DBA as well. Further, you can learn more on how to create view in SQL Server and SQL Server indexed view. If you have any comments or questions, feel free to leave them in the comments below.

See more

ApexSQL Complete is a SQL code complete tool that includes features like code snippets, SQL auto-replacements, tab navigation, saved queries and more for SSMS and Visual Studio



Rajendra Gupta

As an MCSA certified and Microsoft Certified Trainer in Gurgaon, India, with 13 years of experience, Rajendra works for a variety of large companies focusing on performance optimization, monitoring, high availability, and disaster recovery strategies and implementation. He is the author of hundreds of authoritative articles on SQL Server, Azure, MySQL, Linux, Power BI, Performance tuning, AWS/Amazon RDS, Git, and related technologies that have been viewed by over 10m readers to date.

He is the creator of one of the biggest free online collections of articles on a single topic, with his 50-part series on SQL Server Always On Availability Groups. Based on his contribution to the SQL Server community, he has been recognized with various awards including the prestigious "Best author of the year" continuously in 2020 and 2021 at SQLShack.

Raj is always interested in new challenges so if you need consulting help on any subject covered in his writings, he can be reached at rajendra.gupta16@gmail.com

View all posts by Rajendra Gupta

Related Posts:

1. **Top SQL Server Books**
2. **Introduction to pagination in SQL Server**
3. **The benefits, costs, and documentation of database constraints**
4. **Term Extraction Transformation in SSIS**
5. **What is causing database slowdowns?**

Functions, SQL commands, T-SQL

29,481 Views