(T15)討論 View

CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc

==

(T15)討論 View

==

==

0. Summary

```
In Summary:
1.
1.1.
A view is considered as
a stored query or a virtual table
The usage is similar to Table.
1.2.
The table in FROM clause in VIEW is underlying base table.
A view does not store any data.
When execute a view, SQL server actually retrieves data,
from the underlying base table.
1.3.
Create/Alter/Drop View :
1.3.1.
ViewName is normally with prefix "vw"
1.3.2.
Syntax:
--Create View vwName
----ALTER View vwName
--AS
--   Select ...
--  From...
1.3.3.
--EXEC sys.sp_helptext @objname = N'vwName', @columnname = NULL;
sys.sp_helptext show the text definition of View.
1.3.4.
--DROP VIEW wName
1.3.5.
--SELECT  *
--FROM    wName;
2.
Good in View
2.1.
View can hides the complexity of joins and
make Non-IT users feel easier to view the data.
2.2.
View can implement the RowLevelSecurity and ColumnLevelSecurity.
DBA assign the user access to the view and not to the table directly.
2.2.1.
```

RowLevelSecurity can be achieved by using WHERE clause.

```
--WHERE   ColumnA = 'ColumnAValue1';
```

Let the user can only view the ColumnAValue1 data rows.
2.2.2.
ColumnLevelSecurity can be achieved by using SELECT clause.
E.g.
Do not SELECT ColumnA Column, because ColumnA is confidential.
Let user can not view ColumnA Column.
2.3.
Views can show only aggregated data and hide detailed data.

3.
3.1.
Insert/Update/Delete to the view
which does not contains derived or constant field
in ONE underlying base table is OK.
Derived or constant field means
the field which is the combination of multiple fields.
3.1.1.
E.g.

```
--CREATE VIEW vwName
--AS
--    SELECT  ID , FirstName + LastName AS Name ,C3 , C4
--    FROM    TableName;
--GO -- Run the prvious command and begins new batch
```

In this case, ID is the identity column,
so no need to provide value.
Name is the derived field of vwName,
we can not insert value to derived field.

```
--INSERT  INTO vwName
--VALUES  ( 'Name20', C3Value, C4Value );
```

This will return Error.
We may still sepcify the inserted column Name to avoid
the derived or constant field.
In this case, avoid the Name field.

```
--INSERT  INTO vwName
--( C3, C4)
--VALUES  ( C3Value, C4Value );
```

This will be inserted successfully.

3.2.
Insert/Update/Delete to the view in multiple underlying base tables
might cause something we don't expect.
In this case, it need to use trigger to ensure update correctly.
3.2.1.
E.g.

```
--CREATE VIEW vwName
--AS
--    SELECT  T1C1, T1C2, T1C3, T1.ColumnA, T2C1, T2C2, T2C3
--    FROM    T1 join T2 ON T1.ColumnA = T2.ColumnA;
--GO -- Run the prvious command and begins new batch
```

If we update the T2C1 in vwName,
it might cause something we don't expect.
In this case, it need to use trigger to ensure update correctly.

3.2.
Update VIEW :
E.g.

```
--CREATE VIEW vwName
--AS
--    SELECT  T1C1, T1C2, T1C3
--    FROM    T1;
--GO -- Run the prvious command and begins new batch
```

Then you can update as following

```
--Update  vwName
```

```
--Set     T1C2 = T1C2V1
--Where   T1C1 = T1C1V1
```
Or you can delete as following
```
--DELETE  FROM vwName
--where   T1C1 = T1C1V1
------------------------------------------------------------------------------
```
4.
WITH SchemaBinding View   AND   Indexed VIEW:
4.1.
WITH SchemaBinding View Syntax:
```
--CREATE VIEW vwName
--WITH SchemaBinding
--AS
--    SELECT  T1.T1C1 ,
--            SUM(ISNULL(( T2.T2C2 * T1.T1C2 ), 0)) AS AliasName ,
--            COUNT_BIG(*) AS NumberOfItemInEachGroup
--    FROM    dbo.T1
--            INNER JOIN dbo.T2 ON p.ColumnA = o.ColumnA
--    GROUP BY T1.T1C1;
--GO
```
4.1.1.
E.g.
```
--CREATE VIEW vwProductOrderDetail
--WITH SchemaBinding
--AS
--    SELECT  p.ProductName ,
--            SUM(ISNULL(( o.Quantity * p.UnitPrice ), 0)) AS TotalSales ,
--            COUNT_BIG(*) AS Transactions
--    FROM    dbo.Product p
--            INNER JOIN dbo.OrderDetail o ON p.ProductId = o.ProductId
--    GROUP BY p.ProductName;
--GO -- Run the prvious command and begins new batch
```
4.1.2.
--WITH SchemaBinding
Reference:
http://msdn.microsoft.com/en-us/library/ms191432(v=sql.105).aspx
https://www.mssqltips.com/sqlservertip/4673/benefits-of-schemabinding-in-sql-server/
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-procedure-transact-sql
https://blogs.msdn.microsoft.com/sqlprogrammability/2006/05/12/improving-query-plans-with-the-schemabinding-option-on-t-sql-udfs/
WITH SchemaBinding can be used in UserDefinedFunction, StoreProcedure, and VIEW.
WITH SchemaBinding prohibits the affected underlying base table from being dropped.
The VIEW which can include Indexes must using "with SchemaBinding".
4.1.3.
--SUM(ISNULL((o.Quantity * p.UnitPrice), 0)) AS TotalSales
The VIEW which can include Indexes must using "with SchemaBinding".
In addtion,
In order to let View includes Indexes,
Aggregate function in SELECT clause must NOT be NULL.
Therefore, In this case,
it need ISNULL() function to replace NULL values by ZERO.
4.1.4.
--COUNT_BIG(*) AS Transactions
COUNT_BIG(*) return the number of item in the group.
In order to let View includes Indexes,
if the VIEW contains GROUP BY clause,
then SELECT cluase must contain a COUNT_BIG(*).
4.1.5.
```
--FROM    dbo.Product p
--        INNER JOIN dbo.OrderDetail o ON p.ProductId = o.ProductId
```
In order to let View includes Indexes,
the view must use 2 parts name in FROM clause.
-- [schemaName].[TableName]
E.g.
--dbo.OrderDetail   and   dbo.Product

dbo stands for database owner.
It is a schema name just like a folder name

4.2.
Indexed VIEW Syntax:
```
--CREATE UNIQUE CLUSTERED INDEX UIX_vwName_ColumnName
--ON vwName(ColumnName);
```
4.2.1.
E.g.
```
--CREATE UNIQUE CLUSTERED INDEX UIX_vwProductOrderDetail_ProductName
--ON vwProductOrderDetail(ProductName);
```
In order to create Indexed View,
The view must following all the rules we discussed previously.
In this case,
create UNIQUE CLUSTERED INDEX "UIX_vwProductOrderDetail_ProductName",
and assign it to [vwProductOrderDetail].[ProductName] column.
4.2.2.
VIEW V.S. Indexed VIEW
4.2.1.
VIEW Syntax:
```
--CREATE VIEW vwName
--AS
--    SELECT  T1C1, T1C2, T1C3
--    FROM    T1;
--GO
```
A Non-indexed VIEW is a stored SQL query and stores no data.
the data is actually retrieved from the underlying base tables.
In this case, it is T1
4.2.2.
Indexed VIEW Syntax:
```
--CREATE VIEW vwName
--WITH SchemaBinding
--AS
--    SELECT  T1C1, T1C2, T1C3
--    FROM    T1;
--GO
--CREATE UNIQUE CLUSTERED INDEX UIX_vwName_ColumnName
--ON vwName(ColumnName);
--GO
```
In order to let View includes Indexes,
the View must use "WITH SchemaBinding"
When create an Index in VIEW,
The VIEW become materialized and can store data.
The data is actually retrieved from the Indexed VIEW,
rather than the underlying base table, in this case, T1.
Thus, Indexed VIEW improves the performace of fetching data.

4.3.
clustered index V.S. Non-Clustered index
4.3.1.
clustered index:
After the unique clustered index has been created,
then the additional nonclustered indexes could be created.
One VIEW or TABLE can only have ONE clustered index.
A Clustered index is stored with VIEW or TABLE and
does not need additional disk space.
it determines the storage order of data physically in the VIEW or TABLE.
4.3.2.
Non-Clustered index:
4.3.2.1.
One table can have many NonClustered Index.
4.3.2.2.
A Non-Clustered index is in one place and
refer to another place which stores data physically.
Because it need to refer back to the VIEW or TABLE,

Clustered index is slightly faster than a non-clustered index.
4.3.2.3.
A composite index is an index on two or more columns.
E.g.
One Student can enrole many courses.
One Course can be enroled by many students.
Thus, Studen and Course is in many to many relationship.
In this case, We will have 3 Tables,
Student table, Course table, and StudentCourse table in between.
StudentCourse table only contains 2 columns,
which are StudentID and CourseID.
In this case,
StudentID and CourseID in StudentCourse table are in the composite IndexA.
If the query SELECT only StudentID column and CourseID column,
then this is a covering query by the IndexA.
-->
In this case,
the data can simply be returned from the composite IndexA.
A Clustered Index always covers a query,
because it contains all data in a table.
This might be good for performance.

4.4.
Good and Bad of Indexed VIEW
4.4.1.
Run these Query, and see the "Include Actual Execution Plan"
Check the select VIEW query before and after adding Index.

```
--SELECT  *
--FROM    vwProductOrderDetail;
```

See the different in "Include Actual Execution Plan"
before and after adding Index.
4.4.2.
Indexed VIEW Syntax:

```
--CREATE VIEW vwName
--WITH SchemaBinding
--AS
--    SELECT  T1C1, T1C2, T1C3
--    FROM    T1;
--GO
--CREATE UNIQUE CLUSTERED INDEX UIX_vwName_ColumnName
--ON vwName(ColumnName);
--GO
```

Indexed views are good when
the data of underlying bease table, T1, is not frequently changed.
4.4.3.
If you insert or update Indexed views,
then it will need extra time to update the indexes.
The cost of maintaining an indexed view
is much higher than the cost of maintaining a table index.

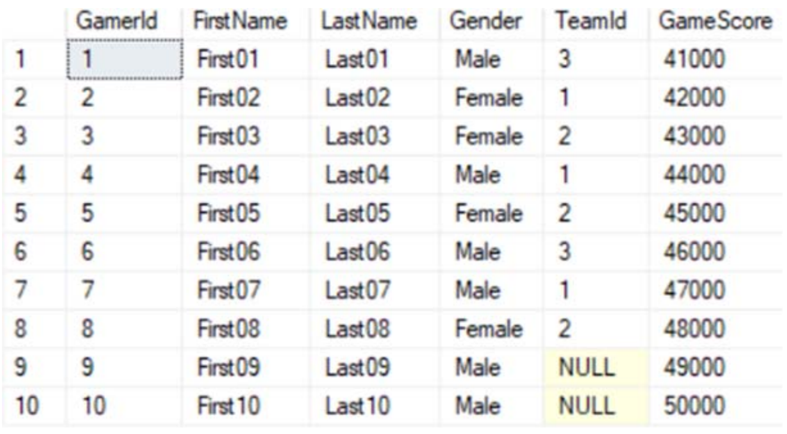
5.
VIEW Limitations
Reference:
https://technet.microsoft.com/en-us/library/ms189918(v=sql.105).aspx
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql
5.1.
View can not accept any parameters.
Table Valued functions are a replacement.
5.2.
We can not define PK, FK, or default value into View columns
5.3.
VIEW can not accept ORDER BY unless it also contains
TOP, OFFSET, or FOR XML.
5.4.
The underlying base table of VIEW must not be temporary tables.

==

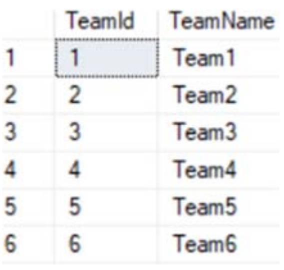
1. CreateAlterDrop_View

```
--==============================================================================
--T015_01_CreateAlterDrop_View
--==============================================================================
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Gamer' ) )
    BEGIN
        TRUNCATE TABLE Gamer;
        DROP TABLE Gamer;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Team' ) )
    BEGIN
        TRUNCATE TABLE Team;
        DROP TABLE Team;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE Team
(
  TeamId INT IDENTITY(1, 1)
             PRIMARY KEY
             NOT NULL ,
  TeamName NVARCHAR(50) NULL
)
GO -- Run the previous command and begins new batch
INSERT  Team
VALUES  ( N'Team1' )
INSERT  Team
VALUES  ( N'Team2' )
INSERT  Team
VALUES  ( N'Team3' )
INSERT  Team
VALUES  ( N'Team4' )
INSERT  Team
VALUES  ( N'Team5' )
INSERT  Team
VALUES  ( N'Team6' )
GO -- Run the previous command and begins new batch
CREATE TABLE Gamer
(
  GamerId INT IDENTITY(1, 1)
              PRIMARY KEY
              NOT NULL ,
  FirstName NVARCHAR(100) NULL ,
  LastName NVARCHAR(100) NULL ,
  Gender NVARCHAR(10) NOT NULL ,
  TeamId INT FOREIGN KEY REFERENCES Team ( TeamId )
```

```
            NULL ,
  GameScore INT NULL
)
GO -- Run the previous command and begins new batch
INSERT  Gamer
VALUES  ( N'First01', N'Last01', 'Male', 3, 41000 )
INSERT  Gamer
VALUES  ( N'First02', N'Last02', 'Female', 1, 42000 )
INSERT  Gamer
VALUES  ( N'First03', N'Last03', 'Female', 2, 43000 )
INSERT  Gamer
VALUES  ( N'First04', N'Last04', 'Male', 1, 44000 )
INSERT  Gamer
VALUES  ( N'First05', N'Last05', 'Female', 2, 45000 )
INSERT  Gamer
VALUES  ( N'First06', N'Last06', 'Male', 3, 46000 )
INSERT  Gamer
VALUES  ( N'First07', N'Last07', 'Male', 1, 47000 )
INSERT  Gamer
VALUES  ( N'First08', N'Last08', 'Female', 2, 48000 )
INSERT  Gamer
VALUES  ( N'First09', N'Last09', 'Male', NULL, 49000 )
INSERT  Gamer
VALUES  ( N'First10', N'Last10', 'Male', NULL, 50000 )
GO -- Run the previous command and begins new batch
SELECT  *
FROM    Gamer;
SELECT  *
FROM    Team;
GO -- Run the previous command and begins new batch
```



	GamerId	FirstName	LastName	Gender	TeamId	GameScore
1	1	First01	Last01	Male	3	41000
2	2	First02	Last02	Female	1	42000
3	3	First03	Last03	Female	2	43000
4	4	First04	Last04	Male	1	44000
5	5	First05	Last05	Female	2	45000
6	6	First06	Last06	Male	3	46000
7	7	First07	Last07	Male	1	47000
8	8	First08	Last08	Female	2	48000
9	9	First09	Last09	Male	NULL	49000
10	10	First10	Last10	Male	NULL	50000



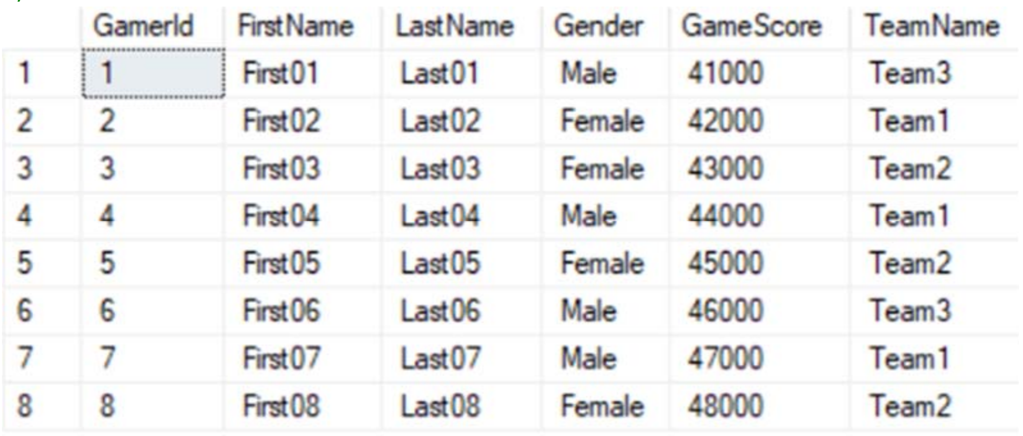
	TeamId	TeamName
1	1	Team1
2	2	Team2
3	3	Team3
4	4	Team4
5	5	Team5
6	6	Team6

==

2. CreateAlterDrop_View

```
--==============================================================================
--T015_02_CreateAlterDrop_View
--==============================================================================
/*
1.
1.1.
A view is considered as
a stored query or a virtual table
The usage is similar to Table.
1.2.
The table in FROM clause in VIEW is underlying base table.
A view does not store any data.
When execute a view, SQL server actually retrieves data,
from the underlying base table.
1.3.
Create/Alter/Drop View :
1.3.1.
ViewName is normally with prefix "vw"
1.3.2.
Syntax:
--Create View vwName
----ALTER View vwName
--AS
--      Select ...
--  From...
1.3.3.
--EXEC sys.sp_helptext @objname = N'vwName', @columnname = NULL;
sys.sp_helptext show the text definition of View.
1.3.4.
--DROP VIEW wName
1.3.5.
--SELECT  *
--FROM    wName;
2.
Good in View
2.1.
View can hides the complexity of joins and
make Non-IT users feel easier to view the data.
2.2.
View can implement the RowLevelSecurity and ColumnLevelSecurity.
DBA assign the user access to the view and not to the table directly.
2.2.1.
RowLevelSecurity can be achieved by using WHERE clause.
--WHERE   ColumnA = 'ColumnAValue1';
Let the user can only view the ColumnAValue1 data rows.
2.2.2.
ColumnLevelSecurity can be achieved by using SELECT clause.
E.g.
Do not SELECT ColumnA Column, because ColumnA is confidential.
Let user can not view ColumnA Column.
2.3.
Views can show only aggregated data and hide detailed data.
*/

--==============================================================================
--T015_02_01
--Drop View if it exist.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamerInTeam' ) )
    BEGIN
        DROP VIEW vwGamerInTeam;
    END;
GO -- Run the previous command and begins new batch
```

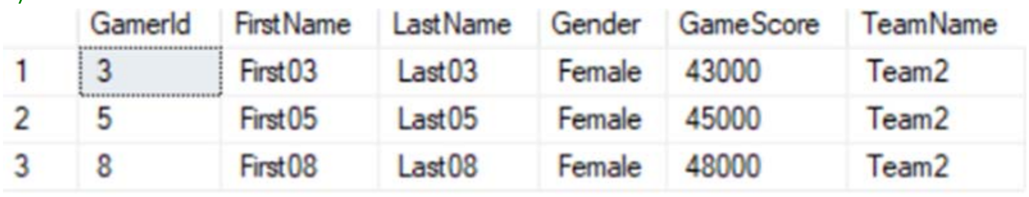
```
CREATE VIEW vwGamerInTeam
AS
    SELECT  g.GamerId ,
            g.FirstName ,
            g.LastName ,
            g.Gender ,
            g.GameScore ,
            t.TeamName
    FROM    Gamer g
            INNER JOIN Team t ON g.TeamId = t.TeamId;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamerInTeam;
GO -- Run the previous command and begins new batch
/*
1.
Syntax:
--Create View vwName
----ALTER View vwName
--AS
--      Select ...
--   From...
2.
2.1.
A view is considered as
a stored query or a virtual table
The usage is similar to Table.
2.2.
The table in FROM clause in VIEW is underlying base table.
A view does not store any data.
When execute a view, SQL server actually retrieves data,
from the underlying base table.
*/
```



	GamerId	FirstName	LastName	Gender	GameScore	TeamName
1	1	First01	Last01	Male	41000	Team3
2	2	First02	Last02	Female	42000	Team1
3	3	First03	Last03	Female	43000	Team2
4	4	First04	Last04	Male	44000	Team1
5	5	First05	Last05	Female	45000	Team2
6	6	First06	Last06	Male	46000	Team3
7	7	First07	Last07	Male	47000	Team1
8	8	First08	Last08	Female	48000	Team2

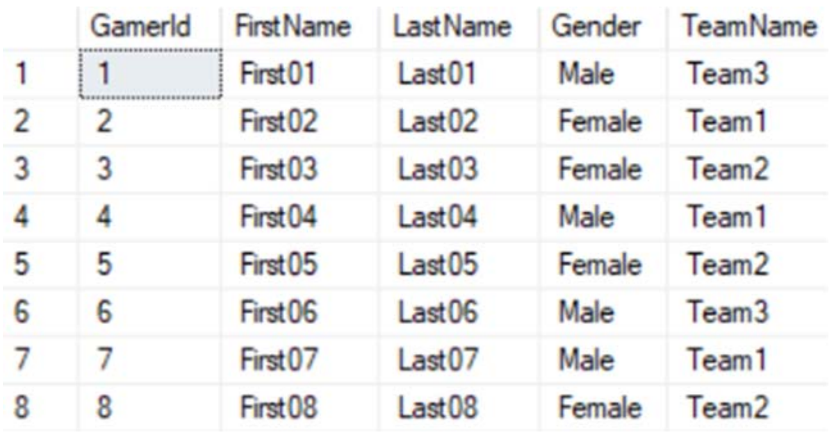
```
--==============================================================================
--T015_02_02
--Row Level Security:
--Drop View if it exist.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamerInTeam2' ) )
    BEGIN
        DROP VIEW vwGamerInTeam2;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwGamerInTeam2
AS
```

```
    SELECT  g.GamerId ,
            g.FirstName ,
            g.LastName ,
            g.Gender ,
            g.GameScore ,
            t.TeamName
    FROM    Gamer g
            INNER JOIN Team t ON g.TeamId = t.TeamId
    WHERE   t.TeamId = 2;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamerInTeam2;
GO -- Run the prvious command and begins new batch
/*
RowLevelSecurity can be achieved by using WHERE clause.
--WHERE   ColumnA = 'ColumnAValue1';
Let the user can only view the ColumnAValue1 data rows.
*/
```



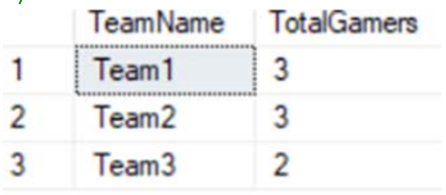
	GamerId	FirstName	LastName	Gender	GameScore	TeamName
1	3	First03	Last03	Female	43000	Team2
2	5	First05	Last05	Female	45000	Team2
3	8	First08	Last08	Female	48000	Team2

```
--==============================================================================
--T015_02_03
--Column Level Security:
--Drop View if it exist.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamerInTeam3' ) )
    BEGIN
        DROP VIEW vwGamerInTeam3;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwGamerInTeam3
AS
    SELECT  g.GamerId ,
            g.FirstName ,
            g.LastName ,
            g.Gender ,
            --g.GameScore ,
            t.TeamName
    FROM    Gamer g
            INNER JOIN Team t ON g.TeamId = t.TeamId;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamerInTeam3;
GO -- Run the prvious command and begins new batch
/*
ColumnLevelSecurity can be achieved by using SELECT clause.
E.g.
Do not SELECT ColumnA Column, because ColumnA is confidential.
Let user can not view ColumnA Column.
*/
```



	GamerId	FirstName	LastName	Gender	TeamName
1	1	First01	Last01	Male	Team3
2	2	First02	Last02	Female	Team1
3	3	First03	Last03	Female	Team2
4	4	First04	Last04	Male	Team1
5	5	First05	Last05	Female	Team2
6	6	First06	Last06	Male	Team3
7	7	First07	Last07	Male	Team1
8	8	First08	Last08	Female	Team2

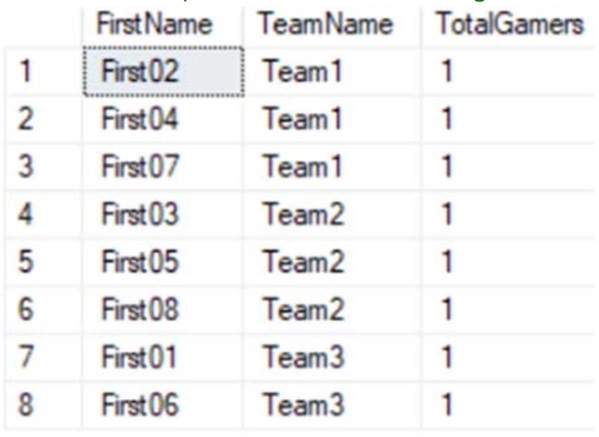
```
--==============================================================================
--T015_02_03
--aggregate with View.
--Drop View if it exist.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamerInTeam4' ) )
    BEGIN
        DROP VIEW vwGamerInTeam4;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwGamerInTeam4
AS
    SELECT  t.TeamName ,
            COUNT(g.GamerId) AS TotalGamers
    FROM    Gamer g
            INNER JOIN Team t ON g.TeamId = t.TeamId
    GROUP BY t.TeamName;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamerInTeam4;
GO -- Run the prvious command and begins new batch
/*
Views can show only aggregated data and hide detailed data.
*/
```



	TeamName	TotalGamers
1	Team1	3
2	Team2	3
3	Team3	2

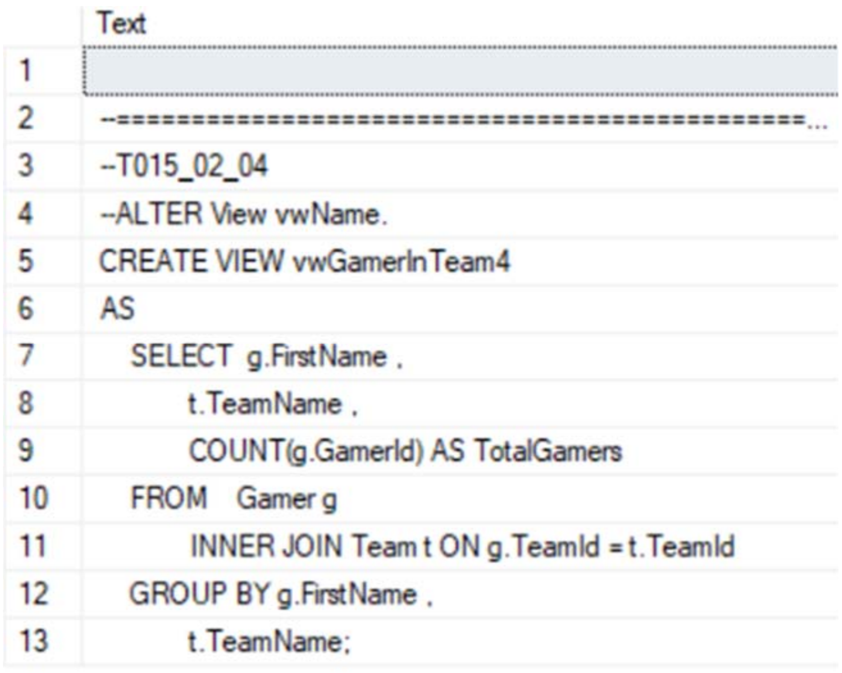
```
--==============================================================================
--T015_02_04
--ALTER View vwName.
ALTER VIEW vwGamerInTeam4
AS
    SELECT  g.FirstName ,
             t.TeamName ,
            COUNT(g.GamerId) AS TotalGamers
    FROM    Gamer g
            INNER JOIN Team t ON g.TeamId = t.TeamId
    GROUP BY g.FirstName ,
             t.TeamName;
```

```
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamerInTeam4;
GO -- Run the prvious command and begins new batch
```



	FirstName	TeamName	TotalGamers
1	First02	Team1	1
2	First04	Team1	1
3	First07	Team1	1
4	First03	Team2	1
5	First05	Team2	1
6	First08	Team2	1
7	First01	Team3	1
8	First06	Team3	1

```
--===========================================================================
--T015_02_05
--sys.sp_helptext show the text definition of View.
EXEC sys.sp_helptext @objname = N'vwGamerInTeam4', -- nvarchar(776)
    @columnname = NULL;
GO -- Run the prvious command and begins new batch
/*
--EXEC sys.sp_helptext @objname = N'vwName', @columnname = NULL;
sys.sp_helptext show the text definition of View.
*/
```



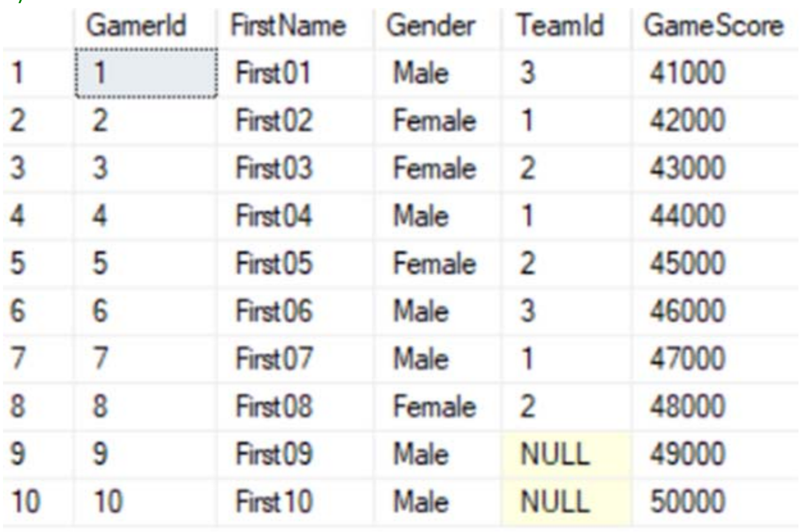
	Text
1	
2	--==...
3	--T015_02_04
4	--ALTER View vwName.
5	CREATE VIEW vwGamerInTeam4
6	AS
7	SELECT g.FirstName ,
8	t.TeamName ,
9	COUNT(g.GamerId) AS TotalGamers
10	FROM Gamer g
11	INNER JOIN Team t ON g.TeamId = t.TeamId
12	GROUP BY g.FirstName ,
13	t.TeamName;

==

3. Insert/Update/Delete in ONE underlying base table

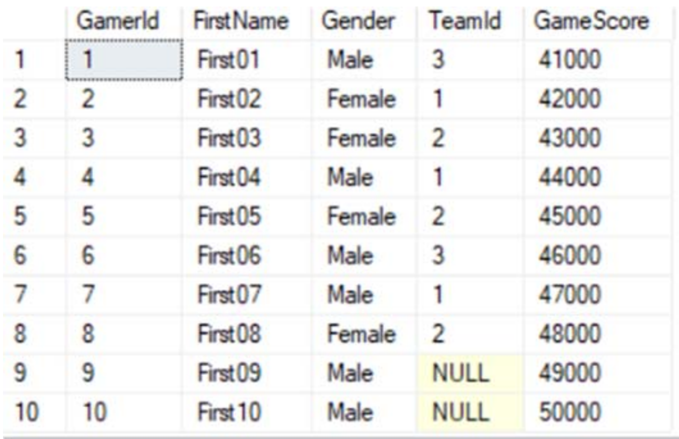
```
--===========================================================================
--T015_03_Insert/Update/Delete in ONE underlying base table
--==============================================================================
/*
3.
3.1.
Insert/Update/Delete to the view
which does not contains derived or constant field
in ONE underlying base table is OK.
Derived or constant field means
the field which is the combination of multiple fields.
3.1.1.
E.g.
--CREATE VIEW vwName
--AS
--    SELECT  ID , FirstName + LastName AS Name ,C3 , C4
--    FROM    TableName;
--GO -- Run the prvious command and begins new batch
In this case, ID is the identity column,
so no need to provide value.
Name is the derived field of vwName,
we can not insert value to derived field.
--INSERT  INTO vwName
--VALUES  ( 'Name20', C3Value, C4Value );
This will return Error.
We may still sepcify the inserted column Name to avoid
the derived or constant field.
In this case, avoid the Name field.
--INSERT  INTO vwName
--( C3, C4)
--VALUES  ( C3Value, C4Value );
This will be inserted successfully.
----------------------------------------
3.2.
Insert/Update/Delete to the view in multiple underlying base tables
might cause something we don't expect.
In this case, it need to use trigger to ensure update correctly.
3.2.1.
E.g.
--CREATE VIEW vwName
--AS
--    SELECT  T1C1, T1C2, T1C3, T1.ColumnA, T2C1, T2C2, T2C3
--    FROM    T1 join T2 ON T1.ColumnA = T2.ColumnA;
--GO -- Run the prvious command and begins new batch
If we update the T2C1 in vwName,
it might cause something we don't expect.
In this case, it need to use trigger to ensure update correctly.
----------------------------------------
3.2.
Update VIEW :
E.g.
--CREATE VIEW vwName
--AS
--    SELECT  T1C1, T1C2, T1C3
--    FROM    T1;
--GO -- Run the prvious command and begins new batch
Then you can update as following
--Update  vwName
--Set     T1C2 = T1C2V1
--Where   T1C1 = T1C1V1
Or you can delete as following
--DELETE  FROM vwName
--where   T1C1 = T1C1V1
*/

--==============================================================================
--T015_03_01
--Create a view which contains ONE underlying base table without derived field
```

```
IF ( EXISTS ( SELECT    *
             FROM      INFORMATION_SCHEMA.TABLES
             WHERE     TABLE_NAME = 'vwGamer' ) )
   BEGIN
       DROP VIEW vwGamer;
   END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwGamer
AS
   SELECT  GamerId ,
           FirstName ,
           Gender ,
           TeamId ,
           GameScore
   FROM    Gamer;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamer;
/*
1.
Create View
--Create View vwName
--AS
--      Select ...
--   From...
2.
2.1.
A view is considered as
a stored query or a virtual table
The usage is similar to Table.
2.2.
The table in FROM clause in VIEW is underlying base table.
A view does not store any data.
When execute a view, SQL server actually retrieves data,
from the underlying base table.
*/
```

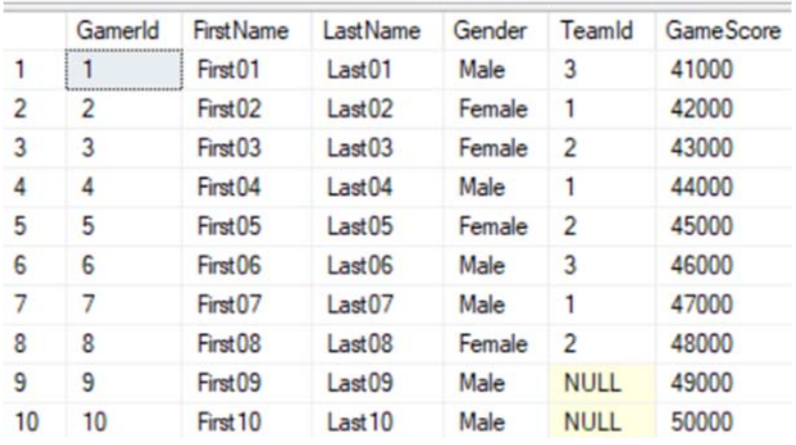


	GamerId	FirstName	Gender	TeamId	GameScore
1	1	First01	Male	3	41000
2	2	First02	Female	1	42000
3	3	First03	Female	2	43000
4	4	First04	Male	1	44000
5	5	First05	Female	2	45000
6	6	First06	Male	3	46000
7	7	First07	Male	1	47000
8	8	First08	Female	2	48000
9	9	First09	Male	NULL	49000
10	10	First10	Male	NULL	50000

```
--=============================================================================
--T015_03_02
--Insert to View which contains ONE underlying base table without derived field
SELECT  *
FROM    vwGamer;
SELECT  *
FROM    Gamer;
```



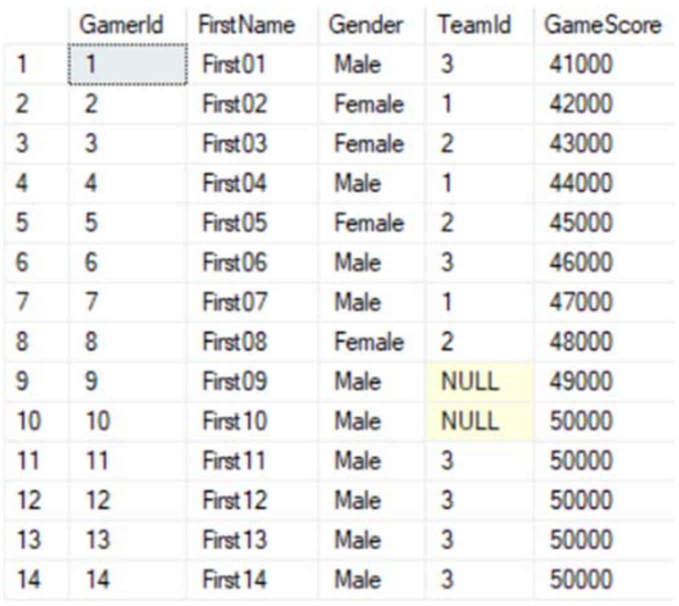
	GamerId	FirstName	Gender	TeamId	GameScore
1	1	First01	Male	3	41000
2	2	First02	Female	1	42000
3	3	First03	Female	2	43000
4	4	First04	Male	1	44000
5	5	First05	Female	2	45000
6	6	First06	Male	3	46000
7	7	First07	Male	1	47000
8	8	First08	Female	2	48000
9	9	First09	Male	NULL	49000
10	10	First10	Male	NULL	50000



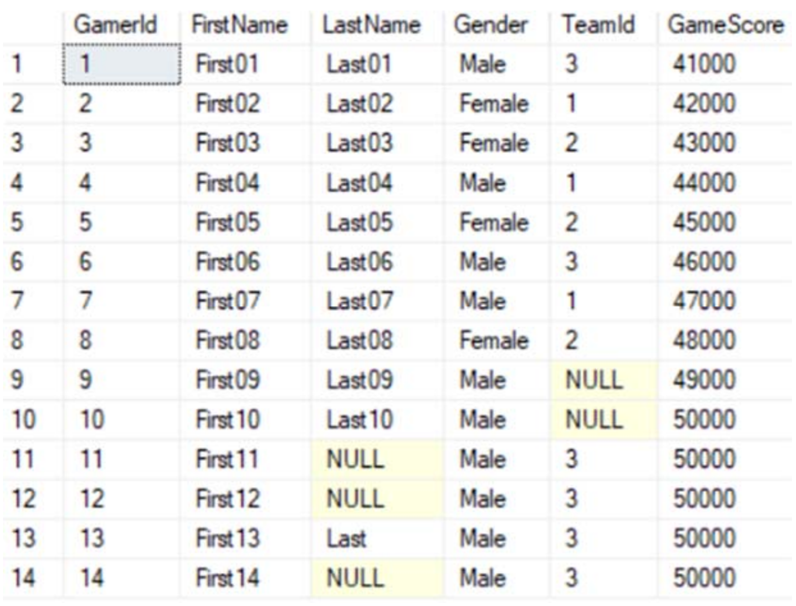
	GamerId	FirstName	LastName	Gender	TeamId	GameScore
1	1	First01	Last01	Male	3	41000
2	2	First02	Last02	Female	1	42000
3	3	First03	Last03	Female	2	43000
4	4	First04	Last04	Male	1	44000
5	5	First05	Last05	Female	2	45000
6	6	First06	Last06	Male	3	46000
7	7	First07	Last07	Male	1	47000
8	8	First08	Last08	Female	2	48000
9	9	First09	Last09	Male	NULL	49000
10	10	First10	Last10	Male	NULL	50000

```sql
--Insert into VIEW
INSERT  INTO vwGamer
VALUES  ( N'First11', 'Male', 3, 50000 )
INSERT  [dbo].[vwGamer]
        ( FirstName ,
          Gender ,
          TeamId ,
          GameScore
        )
VALUES  ( N'First12' ,
          'Male' ,
          3 ,
          50000
        );
--Insert into Table
INSERT  INTO Gamer
VALUES  ( N'First13', N'Last', 'Male', 3, 50000 )
INSERT  INTO Gamer
        ( FirstName ,
          Gender ,
          TeamId ,
          GameScore
        )
VALUES  ( N'First14' ,
          'Male' ,
          3 ,
          50000
        )
SELECT  *
FROM    vwGamer;
SELECT  *
```

```
FROM    Gamer;
GO -- Run the prvious command and begins new batch
/*
When we insert data to VIEW vwName,
We actually insert the data to its underlying base table "Gamer".
The ID is the Identity Column, thus, we do not supply the ID data.
*/
```



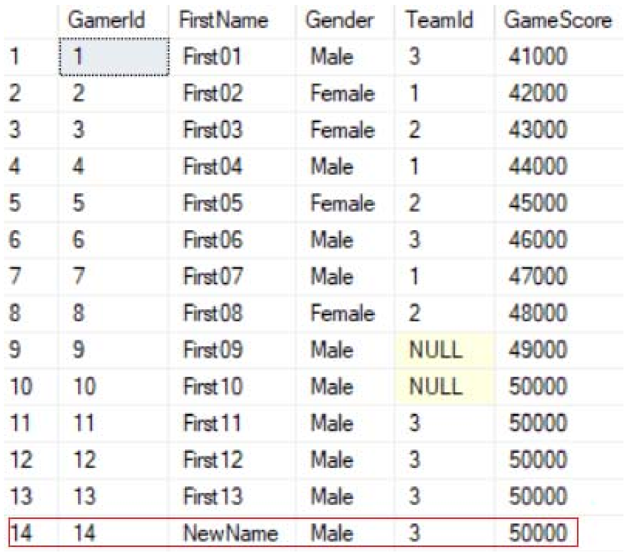
	GamerId	FirstName	Gender	TeamId	GameScore
1	1	First01	Male	3	41000
2	2	First02	Female	1	42000
3	3	First03	Female	2	43000
4	4	First04	Male	1	44000
5	5	First05	Female	2	45000
6	6	First06	Male	3	46000
7	7	First07	Male	1	47000
8	8	First08	Female	2	48000
9	9	First09	Male	NULL	49000
10	10	First10	Male	NULL	50000
11	11	First11	Male	3	50000
12	12	First12	Male	3	50000
13	13	First13	Male	3	50000
14	14	First14	Male	3	50000



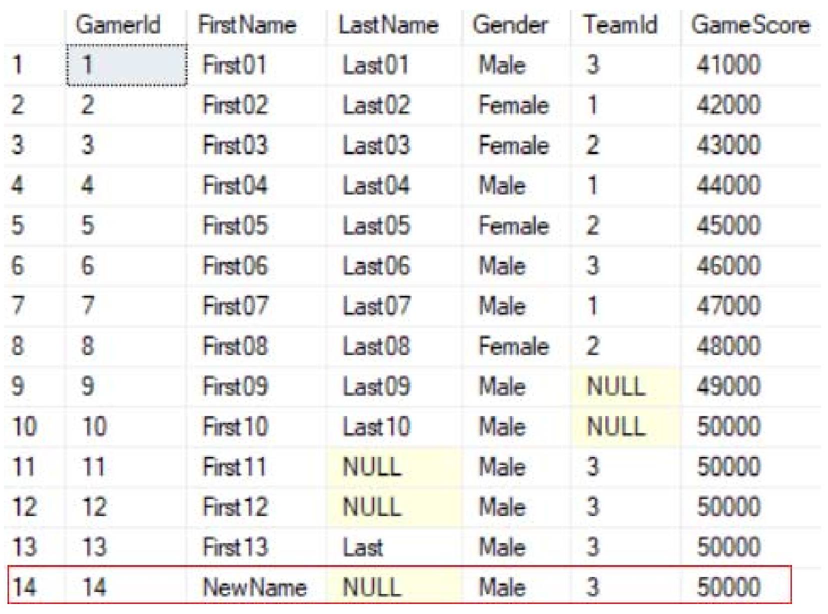
	GamerId	FirstName	LastName	Gender	TeamId	GameScore
1	1	First01	Last01	Male	3	41000
2	2	First02	Last02	Female	1	42000
3	3	First03	Last03	Female	2	43000
4	4	First04	Last04	Male	1	44000
5	5	First05	Last05	Female	2	45000
6	6	First06	Last06	Male	3	46000
7	7	First07	Last07	Male	1	47000
8	8	First08	Last08	Female	2	48000
9	9	First09	Last09	Male	NULL	49000
10	10	First10	Last10	Male	NULL	50000
11	11	First11	NULL	Male	3	50000
12	12	First12	NULL	Male	3	50000
13	13	First13	Last	Male	3	50000
14	14	First14	NULL	Male	3	50000

```
--==============================================================================
--T015_03_03
--Update and Delete to View which contains ONE underlying base table without derived field
SELECT  *
FROM    vwGamer;
SELECT  *
FROM    Gamer;
GO -- Run the prvious command and begins new batch
--UPDATE from VIEW
UPDATE  vwGamer
SET     FirstName = 'NewName'
WHERE   GamerId = ( SELECT  MAX(GamerId)
                    FROM    vwGamer
                  );
GO -- Run the prvious command and begins new batch
```

```sql
SELECT  *
FROM    vwGamer;
SELECT  *
FROM    Gamer;
GO -- Run the prvious command and begins new batch
```

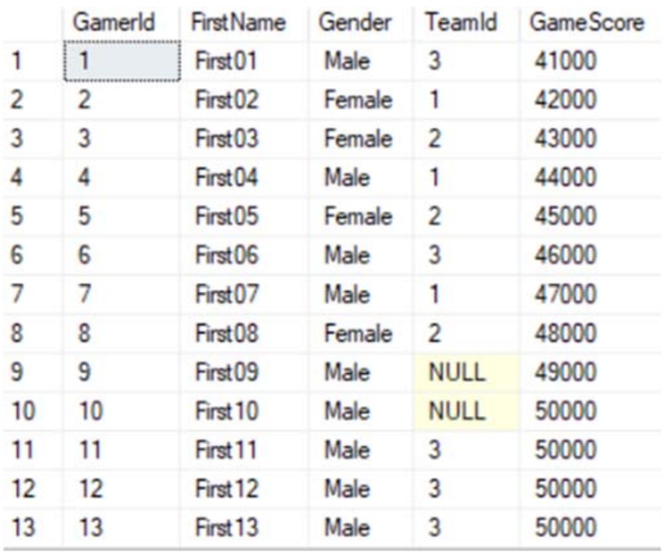


	GamerId	FirstName	Gender	TeamId	GameScore
1	1	First01	Male	3	41000
2	2	First02	Female	1	42000
3	3	First03	Female	2	43000
4	4	First04	Male	1	44000
5	5	First05	Female	2	45000
6	6	First06	Male	3	46000
7	7	First07	Male	1	47000
8	8	First08	Female	2	48000
9	9	First09	Male	NULL	49000
10	10	First10	Male	NULL	50000
11	11	First11	Male	3	50000
12	12	First12	Male	3	50000
13	13	First13	Male	3	50000
14	14	NewName	Male	3	50000

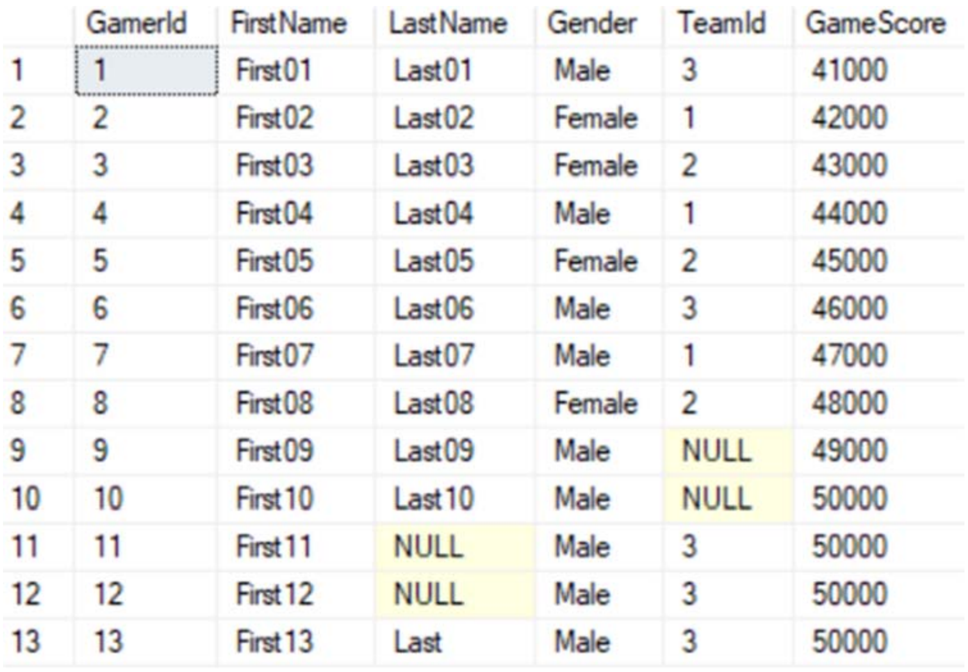


	GamerId	FirstName	LastName	Gender	TeamId	GameScore
1	1	First01	Last01	Male	3	41000
2	2	First02	Last02	Female	1	42000
3	3	First03	Last03	Female	2	43000
4	4	First04	Last04	Male	1	44000
5	5	First05	Last05	Female	2	45000
6	6	First06	Last06	Male	3	46000
7	7	First07	Last07	Male	1	47000
8	8	First08	Last08	Female	2	48000
9	9	First09	Last09	Male	NULL	49000
10	10	First10	Last10	Male	NULL	50000
11	11	First11	NULL	Male	3	50000
12	12	First12	NULL	Male	3	50000
13	13	First13	Last	Male	3	50000
14	14	NewName	NULL	Male	3	50000

```sql
--Delete from VIEW
DELETE  FROM vwGamer
WHERE    FirstName = 'NewName';
SELECT  *
FROM    vwGamer;
SELECT  *
FROM    Gamer;
```

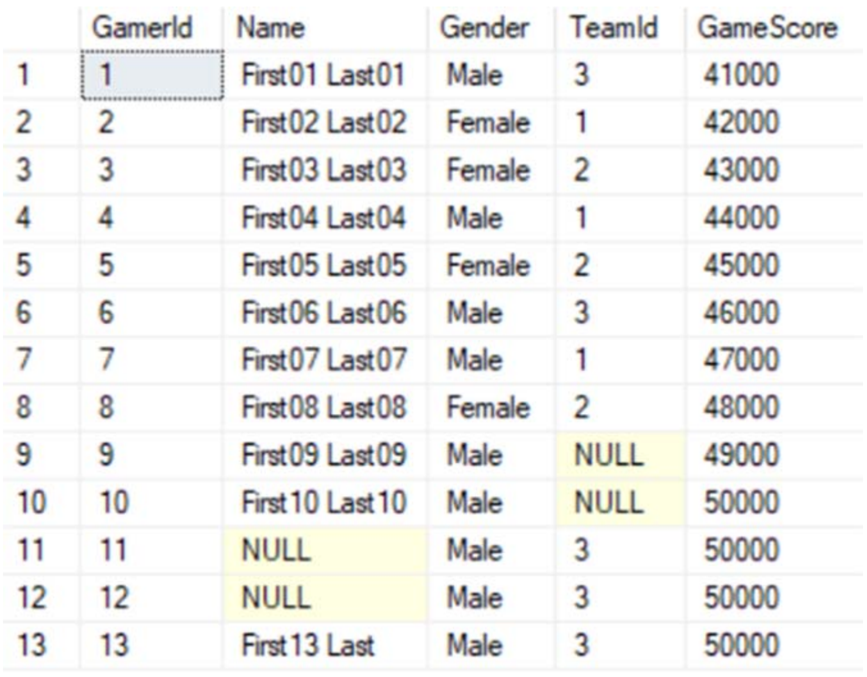


	GamerId	FirstName	Gender	TeamId	GameScore
1	1	First01	Male	3	41000
2	2	First02	Female	1	42000
3	3	First03	Female	2	43000
4	4	First04	Male	1	44000
5	5	First05	Female	2	45000
6	6	First06	Male	3	46000
7	7	First07	Male	1	47000
8	8	First08	Female	2	48000
9	9	First09	Male	NULL	49000
10	10	First10	Male	NULL	50000
11	11	First11	Male	3	50000
12	12	First12	Male	3	50000
13	13	First13	Male	3	50000



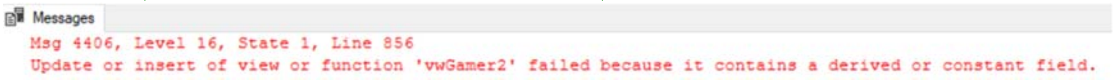
	GamerId	FirstName	LastName	Gender	TeamId	GameScore
1	1	First01	Last01	Male	3	41000
2	2	First02	Last02	Female	1	42000
3	3	First03	Last03	Female	2	43000
4	4	First04	Last04	Male	1	44000
5	5	First05	Last05	Female	2	45000
6	6	First06	Last06	Male	3	46000
7	7	First07	Last07	Male	1	47000
8	8	First08	Last08	Female	2	48000
9	9	First09	Last09	Male	NULL	49000
10	10	First10	Last10	Male	NULL	50000
11	11	First11	NULL	Male	3	50000
12	12	First12	NULL	Male	3	50000
13	13	First13	Last	Male	3	50000

```
--==================================================================================
--T015_03_04
--Create a view which contains ONE underlying base table with derived field
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamer2' ) )
    BEGIN
        DROP VIEW vwGamer2;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwGamer2
AS
    SELECT  GamerId ,
            FirstName + ' ' + LastName AS Name ,
            Gender ,
            TeamId ,
            GameScore
    FROM    Gamer;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamer2;
```



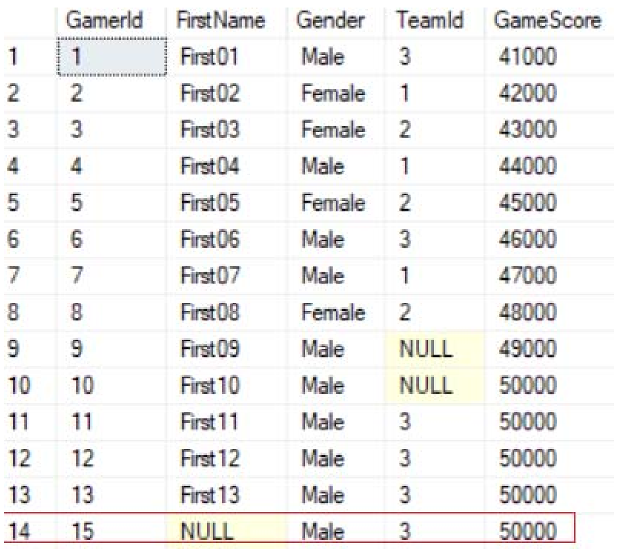
	GamerId	Name	Gender	TeamId	GameScore
1	1	First01 Last01	Male	3	41000
2	2	First02 Last02	Female	1	42000
3	3	First03 Last03	Female	2	43000
4	4	First04 Last04	Male	1	44000
5	5	First05 Last05	Female	2	45000
6	6	First06 Last06	Male	3	46000
7	7	First07 Last07	Male	1	47000
8	8	First08 Last08	Female	2	48000
9	9	First09 Last09	Male	NULL	49000
10	10	First10 Last10	Male	NULL	50000
11	11	NULL	Male	3	50000
12	12	NULL	Male	3	50000
13	13	First13 Last	Male	3	50000

```
--==============================================================================
--T015_03_05
--Insrt to the view which contains ONE underlying base table with derived field
SELECT  *
FROM    vwGamer;
SELECT  *
FROM    vwGamer2;
SELECT  *
FROM    Gamer;
INSERT  INTO vwGamer2
VALUES  ( N'Name15', 'Male', 3, 50000 )
--Return Error,
--because (FirstName + ' ' + LastName AS Name) is a derived field
```



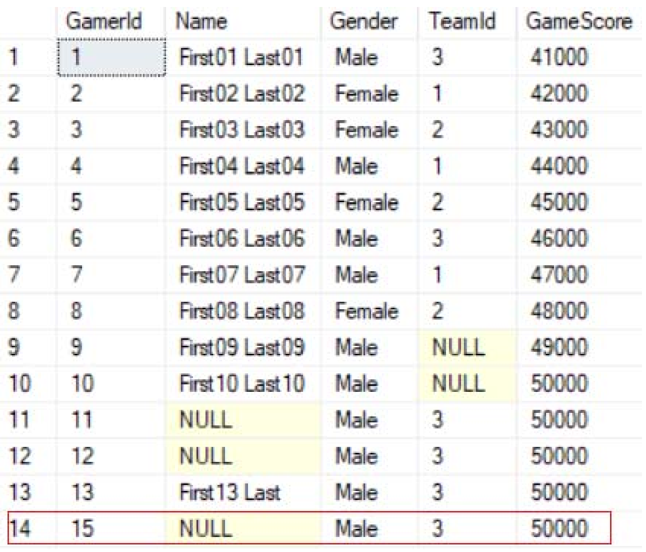
```
Messages
  Msg 4406, Level 16, State 1, Line 856
  Update or insert of view or function 'vwGamer2' failed because it contains a derived or constant field.
```

```
INSERT  INTO vwGamer2
        ( Gender, TeamId, GameScore )
VALUES  ( 'Male', 3, 50000 )
--Insert Success
SELECT  *
FROM    vwGamer;
```



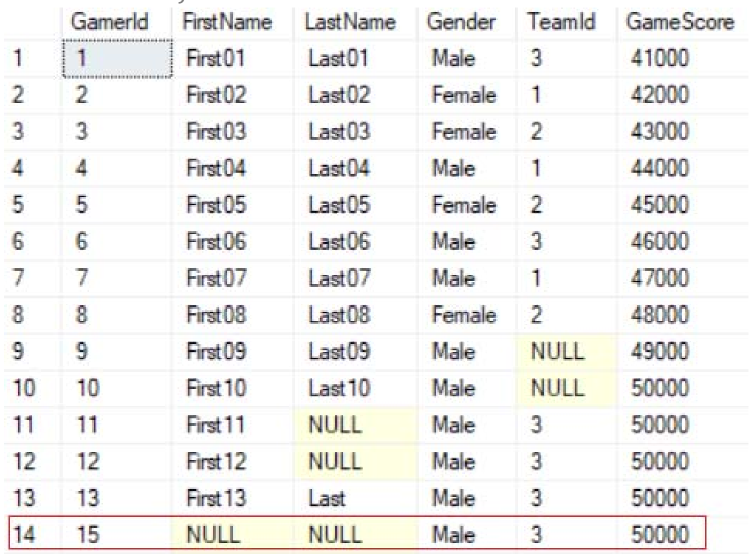
	GamerId	FirstName	Gender	TeamId	GameScore
1	1	First01	Male	3	41000
2	2	First02	Female	1	42000
3	3	First03	Female	2	43000
4	4	First04	Male	1	44000
5	5	First05	Female	2	45000
6	6	First06	Male	3	46000
7	7	First07	Male	1	47000
8	8	First08	Female	2	48000
9	9	First09	Male	NULL	49000
10	10	First10	Male	NULL	50000
11	11	First11	Male	3	50000
12	12	First12	Male	3	50000
13	13	First13	Male	3	50000
14	15	NULL	Male	3	50000

```
SELECT  *
FROM    vwGamer2;
```



	GamerId	Name	Gender	TeamId	GameScore
1	1	First01 Last01	Male	3	41000
2	2	First02 Last02	Female	1	42000
3	3	First03 Last03	Female	2	43000
4	4	First04 Last04	Male	1	44000
5	5	First05 Last05	Female	2	45000
6	6	First06 Last06	Male	3	46000
7	7	First07 Last07	Male	1	47000
8	8	First08 Last08	Female	2	48000
9	9	First09 Last09	Male	NULL	49000
10	10	First10 Last10	Male	NULL	50000
11	11	NULL	Male	3	50000
12	12	NULL	Male	3	50000
13	13	First13 Last	Male	3	50000
14	15	NULL	Male	3	50000

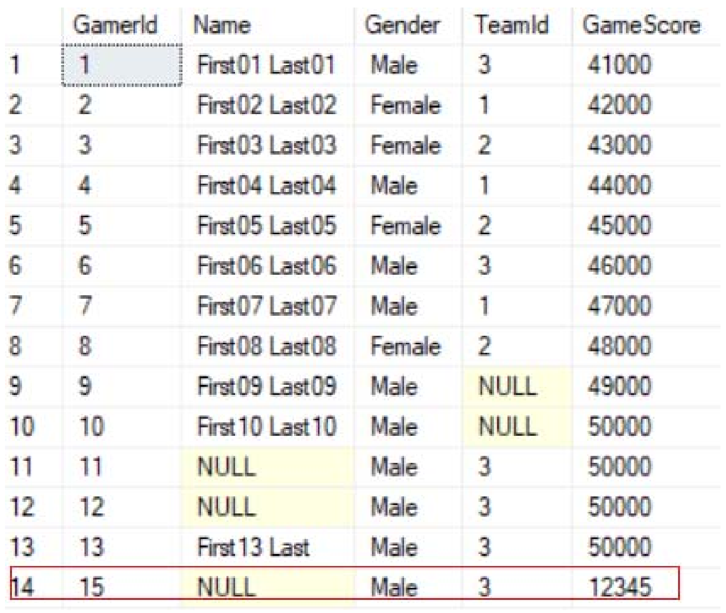
```
SELECT  *
FROM    Gamer;
```



	GamerId	FirstName	LastName	Gender	TeamId	GameScore
1	1	First01	Last01	Male	3	41000
2	2	First02	Last02	Female	1	42000
3	3	First03	Last03	Female	2	43000
4	4	First04	Last04	Male	1	44000
5	5	First05	Last05	Female	2	45000
6	6	First06	Last06	Male	3	46000
7	7	First07	Last07	Male	1	47000
8	8	First08	Last08	Female	2	48000
9	9	First09	Last09	Male	NULL	49000
10	10	First10	Last10	Male	NULL	50000
11	11	First11	NULL	Male	3	50000
12	12	First12	NULL	Male	3	50000
13	13	First13	Last	Male	3	50000
14	15	NULL	NULL	Male	3	50000

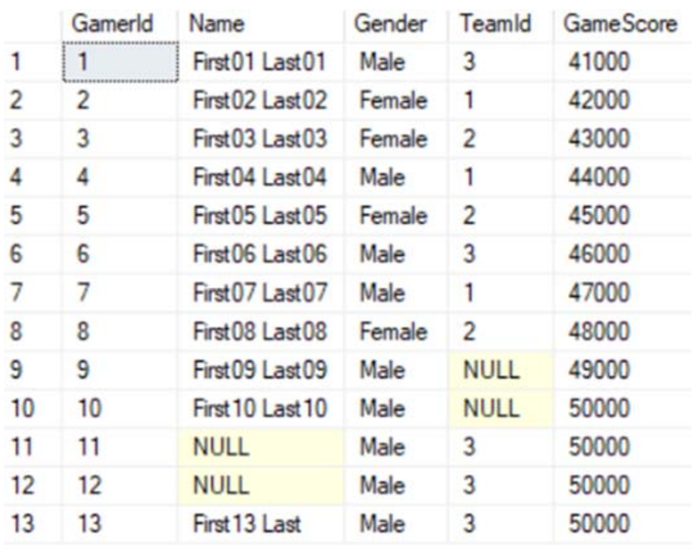
```
/*
3.
3.1.
Insert/Update/Delete to the view
```

```
which does not contains derived or constant field
in ONE underlying base table is OK.
Derived or constant field means
the field which is the combination of multiple fields.
3.1.1.
E.g.
--CREATE VIEW vwName
--AS
--    SELECT  ID , FirstName + LastName AS Name ,C3 , C4
--    FROM    TableName;
--GO -- Run the prvious command and begins new batch
In this case, ID is the identity column,
so no need to provide value.
Name is the derived field of vwName,
we can not insert value to derived field.
--INSERT  INTO vwName
--VALUES  ( 'Name20', C3Value, C4Value );
This will return Error.
We may still sepcify the inserted column Name to avoid
the derived or constant field.
In this case, avoid the Name field.
--INSERT  INTO vwName
--( C3, C4)
--VALUES  ( C3Value, C4Value );
This will be inserted successfully.
*/

--==============================================================================
--T015_03_06
--Update and Delete to the view which contains ONE underlying base table with derived field
SELECT  *
FROM    vwGamer2;
SELECT  *
FROM    Gamer;
GO -- Run the prvious command and begins new batch
--UPDATE from VIEW
UPDATE  vwGamer2
SET     GameScore = 12345
WHERE   GamerId = ( SELECT  MAX(GamerId)
                    FROM    vwGamer
                  );
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamer2;
```



	GamerId	Name	Gender	TeamId	GameScore
1	1	First01 Last01	Male	3	41000
2	2	First02 Last02	Female	1	42000
3	3	First03 Last03	Female	2	43000
4	4	First04 Last04	Male	1	44000
5	5	First05 Last05	Female	2	45000
6	6	First06 Last06	Male	3	46000
7	7	First07 Last07	Male	1	47000
8	8	First08 Last08	Female	2	48000
9	9	First09 Last09	Male	NULL	49000
10	10	First10 Last10	Male	NULL	50000
11	11	NULL	Male	3	50000
12	12	NULL	Male	3	50000
13	13	First13 Last	Male	3	50000
14	15	NULL	Male	3	12345

```
SELECT  *
FROM    Gamer;
GO -- Run the prvious command and begins new batch
--Delete from VIEW
DELETE  FROM vwGamer
WHERE   GameScore = 12345
SELECT  *
FROM    vwGamer2;
```



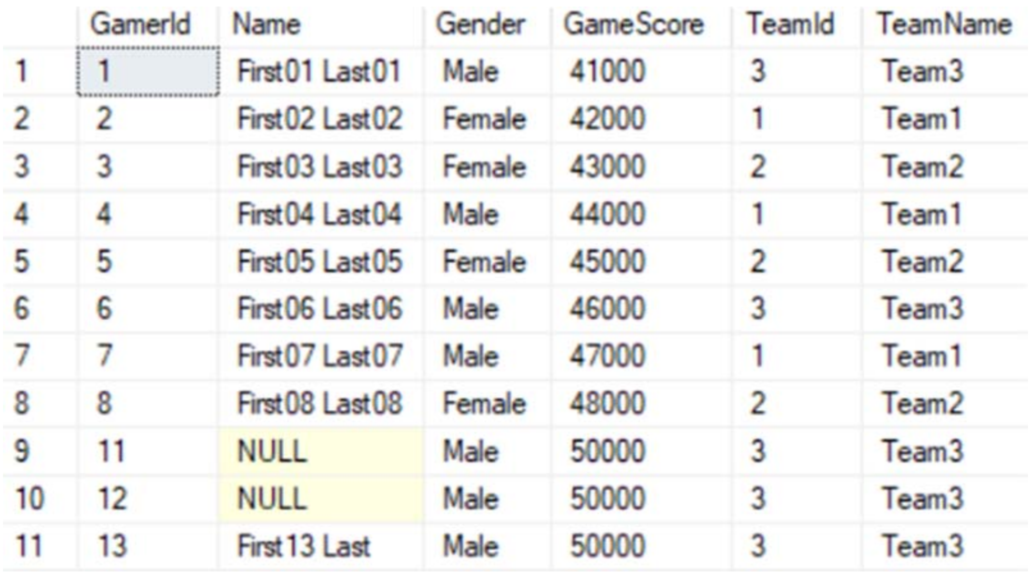
	GamerId	Name	Gender	TeamId	GameScore
1	1	First01 Last01	Male	3	41000
2	2	First02 Last02	Female	1	42000
3	3	First03 Last03	Female	2	43000
4	4	First04 Last04	Male	1	44000
5	5	First05 Last05	Female	2	45000
6	6	First06 Last06	Male	3	46000
7	7	First07 Last07	Male	1	47000
8	8	First08 Last08	Female	2	48000
9	9	First09 Last09	Male	NULL	49000
10	10	First10 Last10	Male	NULL	50000
11	11	NULL	Male	3	50000
12	12	NULL	Male	3	50000
13	13	First13 Last	Male	3	50000

```
SELECT  *
FROM    Gamer;
/*
As long as you don't update or delete the derived field,
(FirstName + LasName AS Name), then it will be fine.
*/
```

==

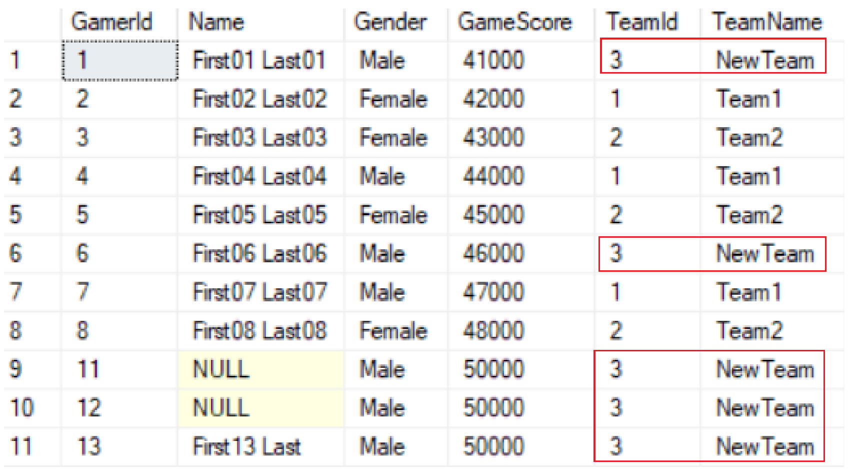
4. Insert/Update/Delete in multiple underlying base table

```
--==================================================================================
--T015_04_Insert/Update/Delete in multiple underlying base table
--==================================================================================
/*
3.2.
Insert/Update/Delete to the view in multiple underlying base tables
might cause something we don't expect.
In this case, it need to use trigger to ensure update correctly.
3.2.1.
E.g.
--CREATE VIEW vwName
--AS
--    SELECT  T1C1, T1C2, T1C3, T1.ColumnA, T2C1, T2C2, T2C3
--    FROM    T1 join T2 ON T1.ColumnA = T2.ColumnA;
--GO -- Run the prvious command and begins new batch
If we update the T2C1 in vwName,
it might cause something we don't expect.
In this case, it need to use trigger to ensure update correctly.
*/

--==================================================================================
--T015_04_01
--Create a view which contains multiple underlying base tables with derived field
--Drop View if it exist.
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamerInTeam5' ) )
    BEGIN
        DROP VIEW vwGamerInTeam5;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwGamerInTeam5
AS
    SELECT  g.GamerId ,
            g.FirstName + ' ' + g.LastName AS Name ,
            g.Gender ,
            g.GameScore ,
            t.TeamId ,
            t.TeamName
    FROM    Gamer g
            INNER JOIN Team t ON g.TeamId = t.TeamId;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamerInTeam5;
GO -- Run the prvious command and begins new batch
/*
ColumnLevelSecurity can be achieved by using SELECT clause.
E.g.
Do not SELECT ColumnA Column, because ColumnA is confidential.
Let user can not view ColumnA Column.
*/
```



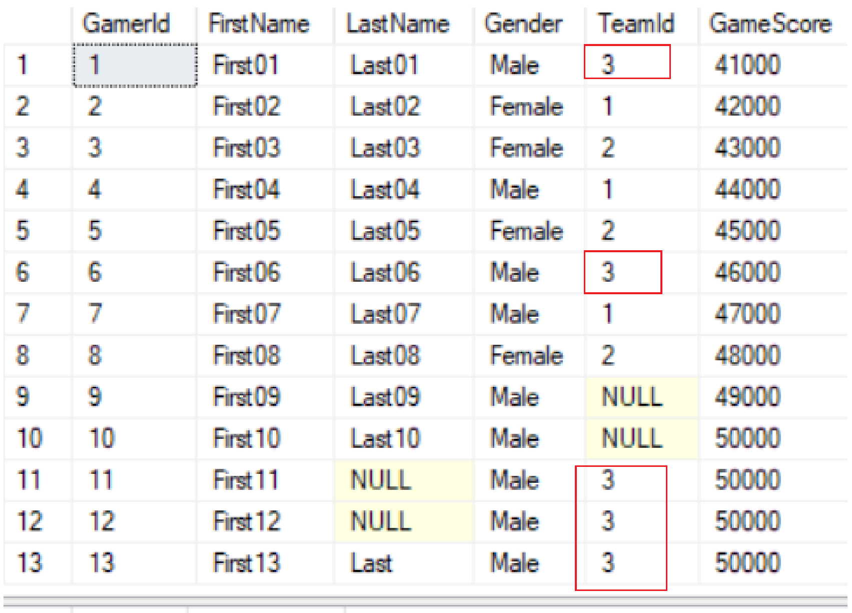
	GamerId	Name	Gender	GameScore	TeamId	TeamName
1	1	First01 Last01	Male	41000	3	Team3
2	2	First02 Last02	Female	42000	1	Team1
3	3	First03 Last03	Female	43000	2	Team2
4	4	First04 Last04	Male	44000	1	Team1
5	5	First05 Last05	Female	45000	2	Team2
6	6	First06 Last06	Male	46000	3	Team3
7	7	First07 Last07	Male	47000	1	Team1
8	8	First08 Last08	Female	48000	2	Team2
9	11	NULL	Male	50000	3	Team3
10	12	NULL	Male	50000	3	Team3
11	13	First13 Last	Male	50000	3	Team3

```
--==============================================================================
--T015_04_02
--Incorrectly Update VIEW
SELECT  *
FROM    vwGamerInTeam5;
UPDATE  vwGamerInTeam5
SET     TeamName = 'NewTeam'
WHERE   GamerId = ( SELECT  MAX(GamerId)
                    FROM    vwGamerInTeam5
                  );
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwGamerInTeam5;
```

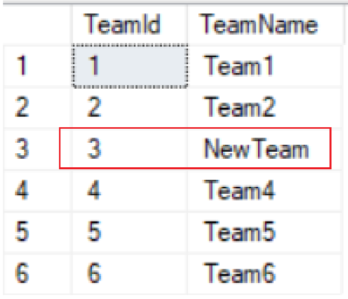


	GamerId	Name	Gender	GameScore	TeamId	TeamName
1	1	First01 Last01	Male	41000	3	NewTeam
2	2	First02 Last02	Female	42000	1	Team1
3	3	First03 Last03	Female	43000	2	Team2
4	4	First04 Last04	Male	44000	1	Team1
5	5	First05 Last05	Female	45000	2	Team2
6	6	First06 Last06	Male	46000	3	NewTeam
7	7	First07 Last07	Male	47000	1	Team1
8	8	First08 Last08	Female	48000	2	Team2
9	11	NULL	Male	50000	3	NewTeam
10	12	NULL	Male	50000	3	NewTeam
11	13	First13 Last	Male	50000	3	NewTeam

```
SELECT  *
FROM    Gamer
SELECT  *
FROM    Team
GO -- Run the prvious command and begins new batch
```



	GamerId	FirstName	LastName	Gender	TeamId	GameScore
1	1	First01	Last01	Male	3	41000
2	2	First02	Last02	Female	1	42000
3	3	First03	Last03	Female	2	43000
4	4	First04	Last04	Male	1	44000
5	5	First05	Last05	Female	2	45000
6	6	First06	Last06	Male	3	46000
7	7	First07	Last07	Male	1	47000
8	8	First08	Last08	Female	2	48000
9	9	First09	Last09	Male	NULL	49000
10	10	First10	Last10	Male	NULL	50000
11	11	First11	NULL	Male	3	50000
12	12	First12	NULL	Male	3	50000
13	13	First13	Last	Male	3	50000



	TeamId	TeamName
1	1	Team1
2	2	Team2
3	3	NewTeam
4	4	Team4
5	5	Team5
6	6	Team6

==

5. IndexedViews

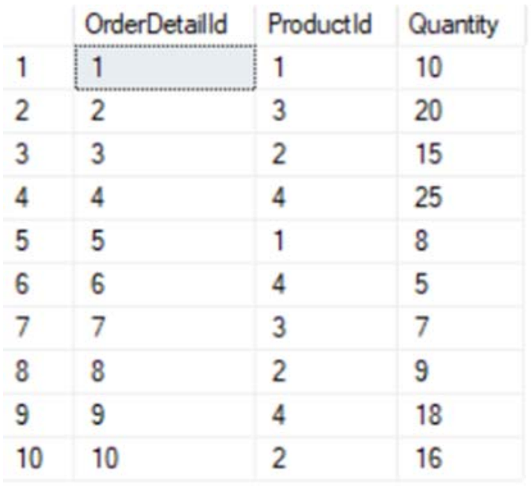
```
--==============================================================================
--T015_05_IndexedViews
--==============================================================================
/*
4.
WITH SchemaBinding View   AND   Indexed VIEW:
4.1.
WITH SchemaBinding View Syntax:
--CREATE VIEW vwName
--WITH SchemaBinding
--AS
--     SELECT  T1.T1C1 ,
--             SUM(ISNULL(( T2.T2C2 * T1.T1C2 ), 0)) AS AliasName ,
--             COUNT_BIG(*) AS NumberOfItemInEachGroup
--     FROM    dbo.T1
--             INNER JOIN dbo.T2 ON p.ColumnA = o.ColumnA
--     GROUP BY T1.T1C1;
--GO
4.1.1.
E.g.
--CREATE VIEW vwProductOrderDetail
--WITH SchemaBinding
--AS
--     SELECT  p.ProductName ,
--             SUM(ISNULL(( o.Quantity * p.UnitPrice ), 0)) AS TotalSales ,
--             COUNT_BIG(*) AS Transactions
--     FROM    dbo.Product p
--             INNER JOIN dbo.OrderDetail o ON p.ProductId = o.ProductId
--     GROUP BY p.ProductName;
--GO -- Run the prvious command and begins new batch
```

```
4.1.2.
--WITH SchemaBinding
Reference:
http://msdn.microsoft.com/en-us/library/ms191432(v=sql.105).aspx
https://www.mssqltips.com/sqlservertip/4673/benefits-of-schemabinding-in-sql-server/
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-procedure-transact-sql
https://blogs.msdn.microsoft.com/sqlprogrammability/2006/05/12/improving-query-plans-with-the-
schemabinding-option-on-t-sql-udfs/
WITH SchemaBinding can be used in UserDefinedFunction, StoreProcedure, and VIEW.
WITH SchemaBinding prohibits the affected underlying base table from being dropped.
The VIEW which can include Indexes must using "with SchemaBinding".
4.1.3.
--SUM(ISNULL(( o.Quantity * p.UnitPrice ), 0)) AS TotalSales
The VIEW which can include Indexes must using "with SchemaBinding".
In addtion,
In order to let View includes Indexes,
Aggregate function in SELECT clause must NOT be NULL.
Therefore, In this case,
it need ISNULL() function to replace NULL values by ZERO.
4.1.4.
--COUNT_BIG(*) AS Transactions
COUNT_BIG(*) return the number of item in the group.
In order to let View includes Indexes,
if the VIEW contains GROUP BY clause,
then SELECT cluase must contain a COUNT_BIG(*).
4.1.5.
--FROM    dbo.Product p
--        INNER JOIN dbo.OrderDetail o ON p.ProductId = o.ProductId
In order to let View includes Indexes,
the view must use 2 parts name in FROM clause.
-- [schemaName].[TableName]
E.g.
--dbo.OrderDetail   and   dbo.Product
dbo stands for database owner.
It is a schema name just like a folder name
---------------------------------
4.2.
Indexed VIEW Syntax:
--CREATE UNIQUE CLUSTERED INDEX UIX_vwName_ColumnName
--ON vwName(ColumnName);
4.2.1.
E.g.
--CREATE UNIQUE CLUSTERED INDEX UIX_vwProductOrderDetail_ProductName
--ON vwProductOrderDetail(ProductName);
In order to create Indexed View,
The view must following all the rules we discussed previously.
In this case,
create UNIQUE CLUSTERED INDEX "UIX_vwProductOrderDetail_ProductName",
and assign it to [vwProductOrderDetail].[ProductName] column.
4.2.2.
VIEW  V.S. Indexed VIEW
4.2.1.
VIEW Syntax:
--CREATE VIEW vwName
--AS
--    SELECT  T1C1, T1C2, T1C3
--    FROM    T1;
--GO
A Non-indexed VIEW is a stored SQL query and stores no data.
the data is actually retrieved from the underlying base tables.
In this case, it is T1
4.2.2.
Indexed VIEW Syntax:
--CREATE VIEW vwName
--WITH SchemaBinding
--AS
```

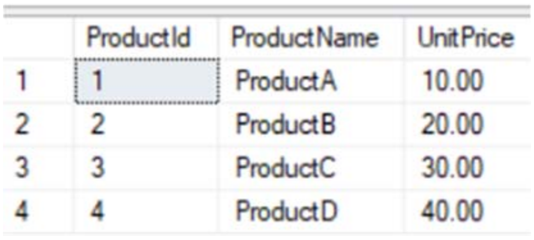
```
--    SELECT  T1C1, T1C2, T1C3
--    FROM    T1;
--GO
--CREATE UNIQUE CLUSTERED INDEX UIX_vwName_ColumnName
--ON vwName(ColumnName);
--GO
In order to let View includes Indexes,
the View must use  "WITH SchemaBinding"
When create an Index in VIEW,
The VIEW become materialized and can store data.
The data is actually retrieved from the Indexed VIEW,
rather than the underlying base table, in this case, T1.
Thus, Indexed VIEW improves the performace of fetching data.
---------------------------------
4.3.
clustered index V.S. Non-Clustered index
4.3.1.
clustered index:
After the unique clustered index has been created,
then the additional nonclustered indexes could be created.
One VIEW or TABLE can only have ONE clustered index.
A Clustered index is stored with VIEW or TABLE and
does not need additional disk space.
it determines the storage order of data physically in the VIEW or TABLE.
4.3.2.
Non-Clustered index:
4.3.2.1.
One table can have many NonClustered Index.
4.3.2.2.
A Non-Clustered index is in one place and
refer to another place which stores data physically.
Because it need to refer back to the VIEW or TABLE,
Clustered index is slightly faster than a non-clustered index.
4.3.2.3.
A composite index is an index on two or more columns.
E.g.
One Student can enrole many courses.
One Course can be enroled by many students.
Thus, Studen and Course is in many to many relationship.
In this case, We will have 3 Tables,
Student table, Course table, and StudentCourse table in between.
StudentCourse table only contains 2 columns,
which are StudentID and CourseID.
In this case,
StudentID and CourseID in StudentCourse table are in the composite IndexA.
If the query SELECT only StudentID column and CourseID column,
then this is a covering query by the IndexA.
-->
In this case,
the data can simply be returned from the composite IndexA.
A Clustered Index always covers a query,
because it contains all data in a table.
This might be good for performance.
---------------------------------
4.4.
Good and Bad of Indexed VIEW
4.4.1.
Run these Query, and see the "Include Actual Execution Plan"
Check the select VIEW query before and after adding Index.
--SELECT  *
--FROM    vwProductOrderDetail;
See the different in "Include Actual Execution Plan"
before and after adding Index.
4.4.2.
Indexed VIEW Syntax:
--CREATE VIEW vwName
--WITH SchemaBinding
```

```
--AS
--    SELECT  T1C1, T1C2, T1C3
--    FROM    T1;
--GO
--CREATE UNIQUE CLUSTERED INDEX UIX_vwName_ColumnName
--ON vwName(ColumnName);
--GO
Indexed views are good when
the data of underlying bease table, T1, is not frequently changed.
4.4.3.
If you insert or update Indexed views,
then it will need extra time to update the indexes.
The cost of maintaining an indexed view
is much higher than the cost of maintaining a table index.
*/



--==============================================================================
--T015_05_01
--Create Sample Data
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwProductOrderDetail' ) )
    BEGIN
        DROP VIEW vwProductOrderDetail;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'OrderDetail' ) )
    BEGIN
        TRUNCATE TABLE OrderDetail;
        DROP TABLE OrderDetail;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Product' ) )
    BEGIN
        TRUNCATE TABLE Product;
        DROP TABLE Product;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE Product
(
  ProductId INT IDENTITY(1, 1)
                PRIMARY KEY
                NOT NULL ,
  ProductName NVARCHAR(100) ,
  UnitPrice MONEY
)
GO -- Run the previous command and begins new batch
INSERT  INTO Product
VALUES  ( 'ProductA', 10 );
INSERT  INTO Product
VALUES  ( 'ProductB', 20 );
INSERT  INTO Product
```

```sql
VALUES  ( 'ProductC', 30 );
INSERT  INTO Product
VALUES  ( 'ProductD', 40 );
GO -- Run the previous command and begins new batch
CREATE TABLE OrderDetail
(
  OrderDetailId INT IDENTITY(1, 1)
                    PRIMARY KEY
                    NOT NULL ,
  ProductId INT FOREIGN KEY REFERENCES Product ( ProductId ) ,
  Quantity SMALLINT
);
GO -- Run the previous command and begins new batch
INSERT  INTO OrderDetail
VALUES  ( 1, 10 );
INSERT  INTO OrderDetail
VALUES  ( 3, 20 );
INSERT  INTO OrderDetail
VALUES  ( 2, 15 );
INSERT  INTO OrderDetail
VALUES  ( 4, 25 );
INSERT  INTO OrderDetail
VALUES  ( 1, 8 );
INSERT  INTO OrderDetail
VALUES  ( 4, 5 );
INSERT  INTO OrderDetail
VALUES  ( 3, 7 );
INSERT  INTO OrderDetail
VALUES  ( 2, 9 );
INSERT  INTO OrderDetail
VALUES  ( 4, 18 );
INSERT  INTO OrderDetail
VALUES  ( 2, 16 );
GO -- Run the previous command and begins new batch
SELECT  *
FROM    OrderDetail;
SELECT  *
FROM    Product;
GO -- Run the previous command and begins new batch
```

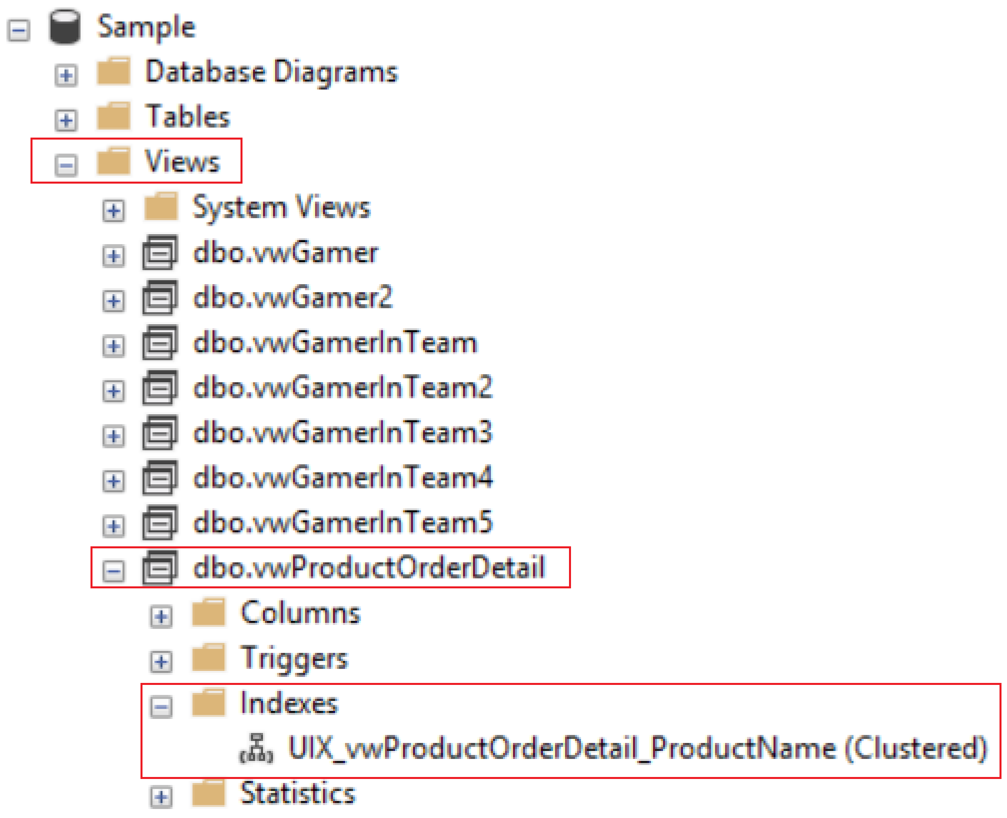


	OrderDetailId	ProductId	Quantity
1	1	1	10
2	2	3	20
3	3	2	15
4	4	4	25
5	5	1	8
6	6	4	5
7	7	3	7
8	8	2	9
9	9	4	18
10	10	2	16

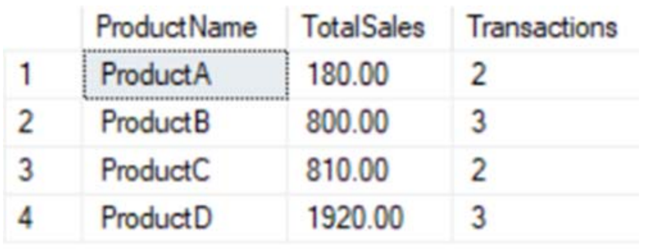


	ProductId	ProductName	UnitPrice
1	1	ProductA	10.00
2	2	ProductB	20.00
3	3	ProductC	30.00
4	4	ProductD	40.00

```
--=============================================================================
--T015_05_02
--Create a View which can include Index.
IF ( EXISTS ( SELECT   *
              FROM     INFORMATION_SCHEMA.TABLES
              WHERE    TABLE_NAME = 'vwProductOrderDetail' ) )
    BEGIN
        DROP VIEW vwProductOrderDetail;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwProductOrderDetail
WITH SchemaBinding
AS
    SELECT  p.ProductName ,
            SUM(ISNULL(( o.Quantity * p.UnitPrice ), 0)) AS TotalSales ,
            COUNT_BIG(*) AS Transactions
    FROM    dbo.Product p
            INNER JOIN dbo.OrderDetail o ON p.ProductId = o.ProductId
    GROUP BY p.ProductName;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    vwProductOrderDetail;
--Create Index for a View
CREATE UNIQUE CLUSTERED INDEX UIX_vwProductOrderDetail_ProductName
ON vwProductOrderDetail(ProductName);
```


Sample
Database Diagrams
Tables
Views
System Views
dbo.vwGamer
dbo.vwGamer2
dbo.vwGamerInTeam
dbo.vwGamerInTeam2
dbo.vwGamerInTeam3
dbo.vwGamerInTeam4
dbo.vwGamerInTeam5
dbo.vwProductOrderDetail
Columns
Triggers
Indexes
UIX_vwProductOrderDetail_ProductName (Clustered)
Statistics

```
SELECT  *
FROM    vwProductOrderDetail;
```

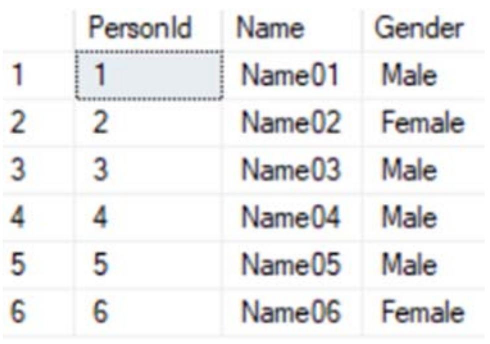


	ProductName	TotalSales	Transactions
1	ProductA	180.00	2
2	ProductB	800.00	3
3	ProductC	810.00	2
4	ProductD	1920.00	3

6. VIEW Limitations

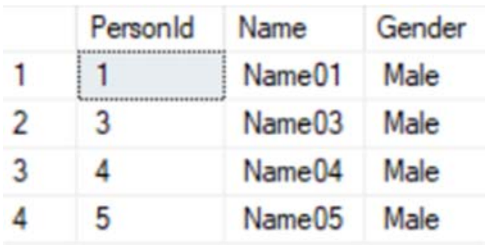
```
--==========================================================================
--T015_06_VIEW Limitations
--==========================================================================
/*
5.
VIEW Limitations
Reference:
https://technet.microsoft.com/en-us/library/ms189918(v=sql.105).aspx
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql
5.1.
View can not accept any parameters.
Table Valued functions are a replacement.
5.2.
We can not define PK, FK, or default value into View columns
5.3.
VIEW can not accept ORDER BY unless it contains
TOP, OFFSET, or FOR XML.
5.4.
```

```
The underlying base table of VIEW must not be temporary tables.
*/

--==============================================================================
--T015_06_01
--Create Sample Data
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Person' ) )
    BEGIN
        TRUNCATE TABLE Person;
        DROP TABLE Person;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE Person
(
  PersonId INT IDENTITY(1, 1)
               PRIMARY KEY
               NOT NULL ,
  [Name] NVARCHAR(100) NULL ,
  Gender NVARCHAR(10) NULL
)
GO -- Run the previous command and begins new batch
INSERT  Person
VALUES  ( N'Name01', 'Male' );
INSERT  Person
VALUES  ( N'Name02', 'Female' );
INSERT  Person
VALUES  ( N'Name03', 'Male' );
INSERT  Person
VALUES  ( N'Name04', 'Male' );
INSERT  Person
VALUES  ( N'Name05', 'Male' );
INSERT  Person
VALUES  ( N'Name06', 'Female' );
GO -- Run the previous command and begins new batch
SELECT  *
FROM    Person
GO -- Run the previous command and begins new batch
```



	PersonId	Name	Gender
1	1	Name01	Male
2	2	Name02	Female
3	3	Name03	Male
4	4	Name04	Male
5	5	Name05	Male
6	6	Name06	Female

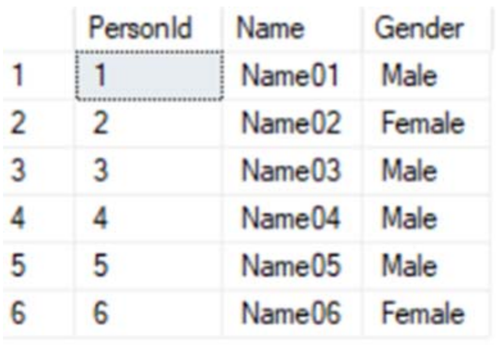
```
--==============================================================================
--T015_06_02
--View can not accept any parameters.
--Table Valued functions are a replacement.
--T015_06_02_01
--Table Valued functions
IF ( EXISTS ( SELECT    *
                FROM      INFORMATION_SCHEMA.ROUTINES
                WHERE     ROUTINE_TYPE = 'FUNCTION'
                          AND LEFT(ROUTINE_NAME, 2) NOT IN ( '@@' )
                          AND SPECIFIC_NAME = 'fnPerson' ) )
    BEGIN
        DROP FUNCTION fnPerson;
    END;
GO -- Run the previous command and begins new batch
CREATE FUNCTION fnPerson ( @Gender NVARCHAR(10) )
RETURNS TABLE
AS
RETURN
    ( SELECT    PersonId ,
                 [Name] ,
                 Gender
      FROM      Person
      WHERE     Gender = @Gender
    );
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    fnPerson('Male');
GO -- Run the prvious command and begins new batch
```



	PersonId	Name	Gender
1	1	Name01	Male
2	3	Name03	Male
3	4	Name04	Male
4	5	Name05	Male

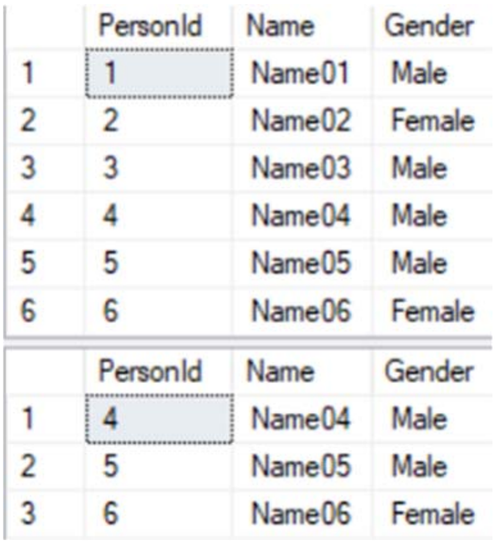
```
/*
----T015_06_02_02
----Syntax ERROR : View can not accept any parameters.
--CREATE VIEW vwPerson( @Gender NVARCHAR(10) )
--AS
--    SELECT    PersonId ,
--                [Name] ,
--                Gender
--      FROM      Person
--      WHERE     Gender = @Gender
--GO
*/
--T015_06_02_03
--View can not accept any parameters.
IF ( EXISTS ( SELECT    *
                FROM      INFORMATION_SCHEMA.TABLES
                WHERE     TABLE_NAME = 'vwPerson' ) )
    BEGIN
        DROP VIEW vwPerson;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwPerson
AS
    SELECT    PersonId ,
                 [Name] ,
                 Gender
      FROM      Person
GO -- Run the prvious command and begins new batch
```

```
--==============================================================================
--T015_06_03
--We can not define PK, FK, or default value into View columns.
/*
----Syntax ERROR
--CREATE VIEW vwPerson2
--AS
--      SELECT PersonId ,
--             [Name] ,
--             Gender DEFAULT 'Male' -- Syntax ERROR
--      FROM    Person
--GO
*/

--==============================================================================
--T015_06_04
--VIEW can not accept ORDER BY unless it contains
--TOP, OFFSET, or FOR XML.
----------------------------------------------------
--T015_06_04_01
/*
---- Syntax ERROR
--CREATE VIEW vwPerson3
--AS
--     SELECT    PersonId ,
--                 [Name] ,
--                 Gender
--       FROM      Person
--        ORDER BY PersonId  -- Syntax ERROR
--GO
*/
----------------------------------------------------
--T015_06_04_02
--VIEW can not accept ORDER BY unless the it contains TOP
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwPerson4' ) )
    BEGIN
        DROP VIEW vwPerson4;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwPerson4
AS
    SELECT TOP 8
            PersonId ,
            [Name] ,
            Gender
    FROM    Person
    ORDER BY PersonId  -- Syntax ERROR
GO -- Run the previous command and begins new batch
SELECT  *
FROM    vwPerson4
GO -- Run the previous command and begins new batch
```



	PersonId	Name	Gender
1	1	Name01	Male
2	2	Name02	Female
3	3	Name03	Male
4	4	Name04	Male
5	5	Name05	Male
6	6	Name06	Female

```
--==============================================================================
--T015_06_05
--VIEW can not accept ORDER BY unless the it contains OFFSET
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwPerson5' ) )
    BEGIN
        DROP VIEW vwPerson5;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwPerson5
AS
    SELECT  PersonId ,
            [Name] ,
            Gender
    FROM    Person
    ORDER BY PersonId
            OFFSET 3 ROWS
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    Person
SELECT  *
FROM    vwPerson5
GO -- Run the prvious command and begins new batch
```

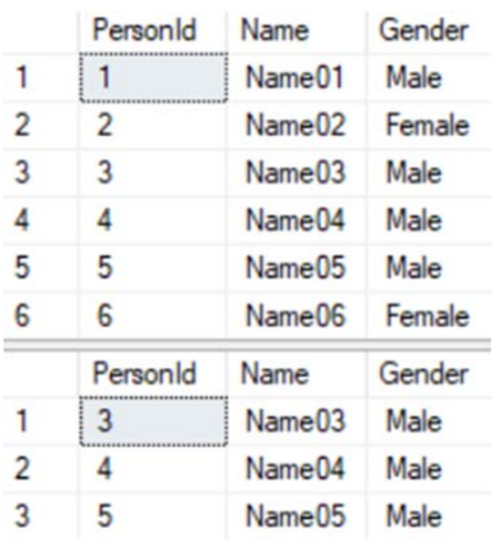


	PersonId	Name	Gender
1	1	Name01	Male
2	2	Name02	Female
3	3	Name03	Male
4	4	Name04	Male
5	5	Name05	Male
6	6	Name06	Female

	PersonId	Name	Gender
1	4	Name04	Male
2	5	Name05	Male
3	6	Name06	Female

```
/*
1.
--ORDER BY PersonId
-- OFFSET 3 ROWS
1.1.
Skip first 3 rows from the sorted result set and return the remaining rows.
```

```
1.2.
[ORDER BY { order_by_expression [ ASC | DESC ] } [ ,...n][<offset_fetch>] ]
Reference:
https://technet.microsoft.com/en-us/library/gg699618(v=sql.110).aspx
The OFFSET-FETCH clause provides you with an option to
fetch only a window or page of results from the result set.
OFFSET-FETCH can be used only with the ORDER BY clause.
2.
VIEW can not accept ORDER BY unless the it also contains
TOP, OFFSET, or FOR XML.
*/


--==============================================================================
--T015_06_06
--VIEW can not accept ORDER BY unless the it contains OFFSET
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwPerson6' ) )
    BEGIN
        DROP VIEW vwPerson6;
    END;
GO -- Run the previous command and begins new batch
CREATE VIEW vwPerson6
AS
    SELECT  PersonId ,
            [Name] ,
            Gender
    FROM    Person
    ORDER BY PersonId
            OFFSET 2 ROWS FETCH NEXT 3 ROWS ONLY;
GO -- Run the prvious command and begins new batch
SELECT  *
FROM    Person
SELECT  *
FROM    vwPerson6
GO -- Run the prvious command and begins new batch
```

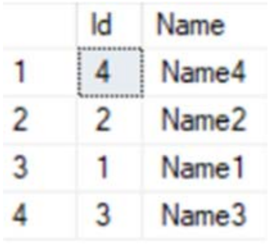


	PersonId	Name	Gender
1	1	Name01	Male
2	2	Name02	Female
3	3	Name03	Male
4	4	Name04	Male
5	5	Name05	Male
6	6	Name06	Female

	PersonId	Name	Gender
1	3	Name03	Male
2	4	Name04	Male
3	5	Name05	Male

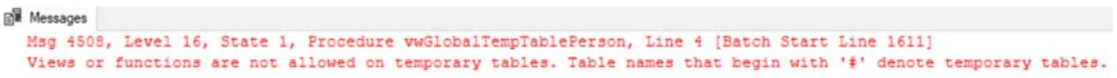
```
/*
1.
--ORDER BY PersonId
--OFFSET 2 ROWS FETCH NEXT 3 ROWS ONLY;
1.1.
Skip first 2 rows from the sorted resultset and return next 3 rows..
```

```
1.2.
[ORDER BY { order_by_expression [ ASC | DESC ] } [ ,...n][<offset_fetch>] ]
Reference:
https://technet.microsoft.com/en-us/library/gg699618(v=sql.110).aspx
The OFFSET-FETCH clause provides you with an option to
fetch only a window or page of results from the result set.
OFFSET-FETCH can be used only with the ORDER BY clause.
2.
VIEW can not accept ORDER BY unless the it also contains
TOP, OFFSET, or FOR XML.
*/

--==============================================================================
--T015_06_07
--The underlying base table of VIEW must not be temporary tables.
IF OBJECT_ID('tempdb..##GlobalTempTablePerson') IS NOT NULL
    BEGIN
        TRUNCATE TABLE ##GlobalTempTablePerson;
        DROP TABLE ##GlobalTempTablePerson;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE ##GlobalTempTablePerson ( Id INT, Name NVARCHAR(20) );
INSERT  INTO ##GlobalTempTablePerson
VALUES  ( 4, 'Name4' );
INSERT  INTO ##GlobalTempTablePerson
VALUES  ( 2, 'Name2' );
INSERT  INTO ##GlobalTempTablePerson
VALUES  ( 1, 'Name1' );
INSERT  INTO ##GlobalTempTablePerson
VALUES  ( 3, 'Name3' );
SELECT  *
FROM    ##GlobalTempTablePerson
GO -- Run the prvious command and begins new batch
```



	Id	Name
1	4	Name4
2	2	Name2
3	1	Name1
4	3	Name3

```
--Error
CREATE VIEW vwGlobalTempTablePerson
AS
    Select  *
    from    ##GlobalTempTablePerson
GO -- Run the prvious command and begins new batch
```



```
Messages
Msg 4508, Level 16, State 1, Procedure vwGlobalTempTablePerson, Line 4 [Batch Start Line 1611]
Views or functions are not allowed on temporary tables. Table names that begin with '#' denote temporary tables.
```

==

7. Clean up

```
--==============================================================================
```

```sql
--T015_07_Clean up
--==============================================================================
--Clean up
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Gamer' ) )
    BEGIN
        TRUNCATE TABLE Gamer;
        DROP TABLE Gamer;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Team' ) )
    BEGIN
        TRUNCATE TABLE Team;
        DROP TABLE Team;
    END;
GO -- Run the previous command and begins new batch
--------------------------------------------------------------------
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamerInTeam' ) )
    BEGIN
        DROP VIEW vwGamerInTeam;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamerInTeam2' ) )
    BEGIN
        DROP VIEW vwGamerInTeam2;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamerInTeam3' ) )
    BEGIN
        DROP VIEW vwGamerInTeam3;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamerInTeam4' ) )
    BEGIN
        DROP VIEW vwGamerInTeam4;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'vwGamerInTeam5' ) )
    BEGIN
        DROP VIEW vwGamerInTeam5;
    END;
```

```sql
GO -- Run the previous command and begins new batch
--------------------------------------------------------------------
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'vwGamer' ) )
    BEGIN
        DROP VIEW vwGamer;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'vwGamer2' ) )
    BEGIN
        DROP VIEW vwGamer2;
    END;
GO -- Run the previous command and begins new batch
--------------------------------------------------------------------
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'vwProductOrderDetail' ) )
    BEGIN
        DROP VIEW vwProductOrderDetail;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'OrderDetail' ) )
    BEGIN
        TRUNCATE TABLE OrderDetail;
        DROP TABLE OrderDetail;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'Product' ) )
    BEGIN
        TRUNCATE TABLE Product;
        DROP TABLE Product;
    END;
GO -- Run the previous command and begins new batch
--------------------------------------------------------------------
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE      TABLE_NAME = 'Person' ) )
    BEGIN
        TRUNCATE TABLE Person;
        DROP TABLE Person;
    END;
GO -- Run the previous command and begins new batch
--------------------------------
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.ROUTINES
              WHERE      ROUTINE_TYPE = 'FUNCTION'
                         AND LEFT(ROUTINE_NAME, 2) NOT IN ( '@@' )
```

```sql
                        AND SPECIFIC_NAME = 'fnPerson' ) )
   BEGIN
       DROP FUNCTION fnPerson;
   END;
GO -- Run the previous command and begins new batch
--------------------------------
IF ( EXISTS ( SELECT    *
             FROM      INFORMATION_SCHEMA.TABLES
             WHERE      TABLE_NAME = 'vwPerson' ) )
   BEGIN
       DROP VIEW vwPerson;
   END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
             FROM      INFORMATION_SCHEMA.TABLES
             WHERE      TABLE_NAME = 'vwPerson2' ) )
   BEGIN
       DROP VIEW vwPerson2;
   END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
             FROM      INFORMATION_SCHEMA.TABLES
             WHERE      TABLE_NAME = 'vwPerson3' ) )
   BEGIN
       DROP VIEW vwPerson3;
   END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
             FROM      INFORMATION_SCHEMA.TABLES
             WHERE      TABLE_NAME = 'vwPerson4' ) )
   BEGIN
       DROP VIEW vwPerson4;
   END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
             FROM      INFORMATION_SCHEMA.TABLES
             WHERE      TABLE_NAME = 'vwPerson5' ) )
   BEGIN
       DROP VIEW vwPerson5;
   END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
             FROM      INFORMATION_SCHEMA.TABLES
             WHERE      TABLE_NAME = 'vwPerson6' ) )
   BEGIN
       DROP VIEW vwPerson6;
   END;
GO -- Run the previous command and begins new batch
-----------------------------
IF OBJECT_ID('tempdb..##GlobalTempTablePerson') IS NOT NULL
   BEGIN
       TRUNCATE TABLE ##GlobalTempTablePerson;
       DROP TABLE ##GlobalTempTablePerson;
```

```
    END;
GO -- Run the previous command and begins new batch
```