SQL Queries For Beginners

Syed Shanu

(C# Corner MVP)

Index	Page No

About Author

Syed Shanu is basically from Madurai, Tamil Nadu, India. He was been working in South Korea for past 8 years. He has 9+ years of Experience in Microsoft Technologies. His work experience with Language and Technology start's from ASP and SQL Server, Then VB.NET and C# for PDA Application, Touch Screen Application Development, Desktop Application, ASP.NET Web Application Development, MVC and WPF.

Syed Shanu

(C# Corner MVP)

Create and Insert Query

Before we start we will first create a table and insert some sample data into it so we can use these tables in our select class. I want to explain the table design with actual data since that will help the reader to understand this in detail.

In database design the important phase is to create a table with proper normalization with primary and foreign key relationships.

Now in this example we will assume we need to create a restaurant Order Management tables with relationships.

For this we need an Order Master, an Order Detail and an Item Master as the major tables. We will use these 3 tables in this article. First let's start by creating the tables. As I have said, we will use the 3 tables here in this article so we start by creating the Item Master as the first table. As the name suggests this table will be used to store the items.

Create Table

Item Master: Here we have created an ItemMaster with the necessary fields. As I already said, we need to make a plan about our project. List all the necessary tables we need to create for the project and describe and list all the fields to be created for each table. Here I used Item_Code as a primary key field that will be used in another table for the reference to this main table.

```
CREATE TABLE [dbo].[ItemMasters](
    [Item_Code] [varchar](20) NOT NULL,
    [Item_Name] [varchar](100) NOT NULL,
    [Price]  Int NOT NULL,
    [TAX1]  Int NOT NULL,
    [Discount]  Int NOT NULL,
    [Description] [varchar](200) NOT NULL,
    [IN_DATE] [datetime] NOT NULL,
    [IN_USR_ID] [varchar](20) NOT NULL,
    [UP_DATE] [datetime] NOT NULL,
    [UP_USR_ID] [varchar](20) NOT NULL,
 CONSTRAINT [PK_ItemMasters] PRIMARY KEY CLUSTERED
(
    [Item_Code] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

Insert Item Master

```
INSERT INTO [ItemMasters]   ([Item_Code],[Item_Name],[Price],[TAX1],[Discount],[Description],[IN_DATE]
```

```sql
            ,[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
            ('Item001','Coke',55,1,0,'Coke which need to be cold',GETDATE(),'SHANU'
            ,GETDATE(),'SHANU')

INSERT INTO [ItemMasters]   ([Item_Code],[Item_Name],[Price],[TAX1],[Discount],[Description],[IN_DATE]
            ,[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
            ('Item002','Coffee',40,0,2,'Coffe Might be Hot or Cold user choice',GETDATE(),'SHANU'
            ,GETDATE(),'SHANU')

INSERT INTO [ItemMasters]   ([Item_Code],[Item_Name],[Price],[TAX1],[Discount],[Description],[IN_DATE]
            ,[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
            ('Item003','Chiken Burger',125,2,5,'Spicy',GETDATE(),'SHANU'
            ,GETDATE(),'SHANU')

INSERT INTO [ItemMasters]   ([Item_Code],[Item_Name],[Price],[TAX1],[Discount],[Description],[IN_DATE]
            ,[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
            ('Item004','Potato Fry',15,0,0,'No Comments',GETDATE(),'SHANU'
            ,GETDATE(),'SHANU')
```

Order Master: Since this is a master table we will have one main record and all the subsequent related records will be stored in the Order Detail Table.

Note: First please understand what Master and Detail means. If you don't understand what a Master and a Detail are then I will explain that first. Master means there is one main record and in the details we have all the details of the main record.

Say for example we have a restaurant and an order for one Coke, one Burger and one Potato Fries. Which means I have make an order from the waiter with 3 Items. So the Order_No for the example can be “Ord0001” and this Order_No will have the 3 items so first we create the Order Master.

```sql
CREATE TABLE [dbo].[OrderMasters](
    [Order_No] [varchar](20) NOT NULL,
    [Table_ID] [varchar](20) NOT NULL,
    [Description] [varchar](200) NOT NULL,
    [IN_DATE] [datetime] NOT NULL,
    [IN_USR_ID] [varchar](20) NOT NULL,
    [UP_DATE] [datetime] NOT NULL,
    [UP_USR_ID] [varchar](20) NOT NULL,
 CONSTRAINT [PK_OrderMasters] PRIMARY KEY CLUSTERED
(
    [Order_No] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]
```

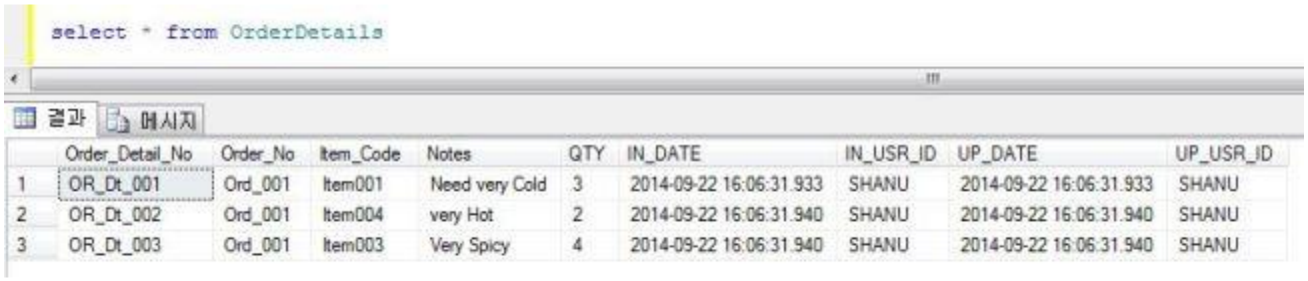
Insert Order Master

```
INSERT INTO [OrderMasters]
           ([Order_No],[Table_ID] ,[Description],[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
           ('Ord_001','T1','',GETDATE(),'SHANU' ,GETDATE(),'SHANU')

INSERT INTO [OrderMasters]
           ([Order_No],[Table_ID] ,[Description],[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
           ('Ord_002','T2','',GETDATE(),'Mak' ,GETDATE(),'MAK')

INSERT INTO [OrderMasters]
           ([Order_No],[Table_ID] ,[Description],[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
           ('Ord_003','T3','',GETDATE(),'RAJ' ,GETDATE(),'RAJ')
```

Order Detail: As table name suggests this will have the details of an order so for example we consider my preceding example order with the 3 items. First we create an Order Detail table.

```
CREATE TABLE [dbo].[OrderDetails](
    [Order_Detail_No] [varchar](20) NOT NULL,
    [Order_No] [varchar](20) CONSTRAINT  fk_OrderMasters FOREIGN KEY REFERENCES OrderMasters(Order_No),
    [Item_Code] [varchar](20) CONSTRAINT  fk_ItemMasters FOREIGN KEY REFERENCES ItemMasters(Item_Code),
    [Notes] [varchar](200) NOT NULL,
    [QTY]  INT NOT NULL,
    [IN_DATE] [datetime] NOT NULL,
    [IN_USR_ID] [varchar](20) NOT NULL,
    [UP_DATE] [datetime] NOT NULL,
    [UP_USR_ID] [varchar](20) NOT NULL,
 CONSTRAINT [PK_OrderDetails] PRIMARY KEY CLUSTERED
(
    [Order_Detail_No] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
) ON [PRIMARY]


--Now let’s insert the 3 items for the above Order No 'Ord_001'.
INSERT INTO [OrderDetails]
           ([Order_Detail_No],[Order_No],[Item_Code],[Notes],[QTY]
           ,[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
```

```
            ('OR_Dt_001','Ord_001','Item001','Need very Cold',3
            ,GETDATE(),'SHANU' ,GETDATE(),'SHANU')

INSERT INTO [OrderDetails]
            ([Order_Detail_No],[Order_No],[Item_Code],[Notes],[QTY]
            ,[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
            ('OR_Dt_002','Ord_001','Item004','very Hot ',2
            ,GETDATE(),'SHANU' ,GETDATE(),'SHANU')

INSERT INTO [OrderDetails]
            ([Order_Detail_No],[Order_No],[Item_Code],[Notes],[QTY]
            ,[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
            ('OR_Dt_003','Ord_001','Item003','Very Spicy',4
            ,GETDATE(),'SHANU' ,GETDATE(),'SHANU')
```



```
select * from OrderDetails
```

결과 | 메시지

	Order_Detail_No	Order_No	Item_Code	Notes	QTY	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	OR_Dt_001	Ord_001	Item001	Need very Cold	3	2014-09-22 16:06:31.933	SHANU	2014-09-22 16:06:31.933	SHANU
2	OR_Dt_002	Ord_001	Item004	very Hot	2	2014-09-22 16:06:31.940	SHANU	2014-09-22 16:06:31.940	SHANU
3	OR_Dt_003	Ord_001	Item003	Very Spicy	4	2014-09-22 16:06:31.940	SHANU	2014-09-22 16:06:31.940	SHANU

Here we can see the same ordre_No for 3 different details with different items. Refer to the Item Master for the Item Name details. We will see in detail the select query uses the following in this article. Now we insert another Order Master detail into the Detail tables.

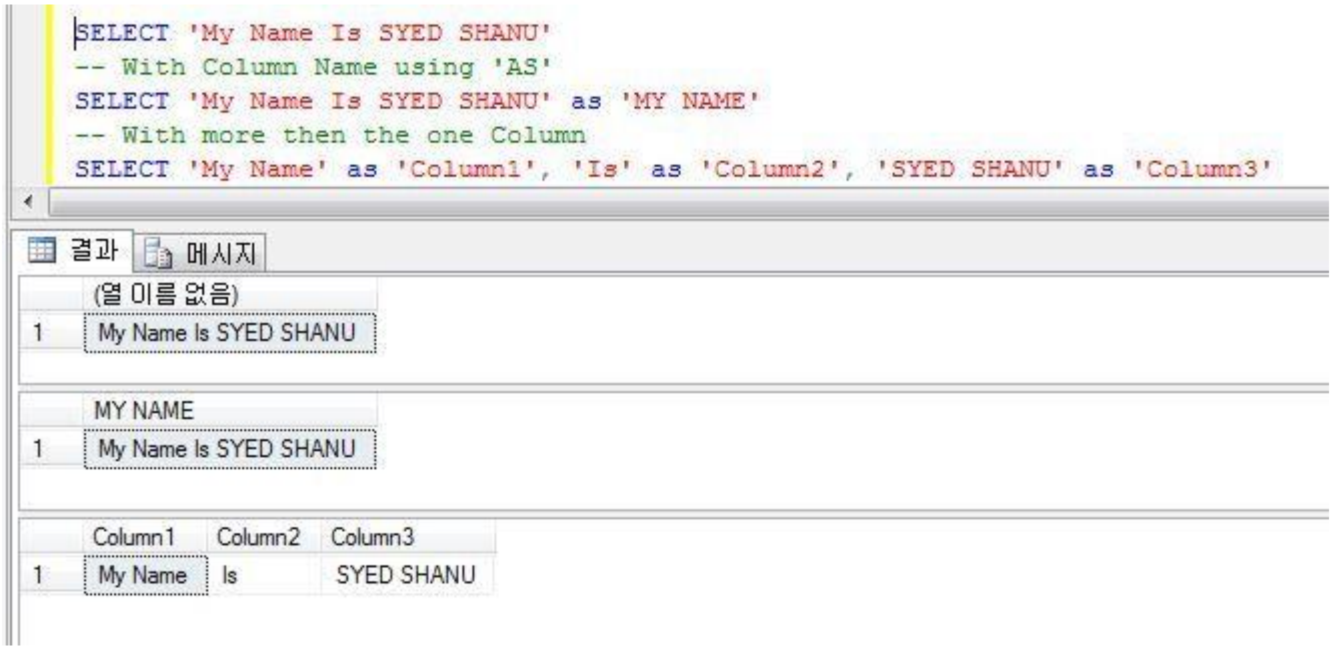
```
INSERT INTO [OrderDetails]
            ([Order_Detail_No],[Order_No],[Item_Code],[Notes],[QTY]
            ,[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
            ('OR_Dt_004','Ord_002','Item002','Need very Hot',2
            ,GETDATE(),'SHANU' ,GETDATE(),'SHANU')

INSERT INTO [OrderDetails]
            ([Order_Detail_No],[Order_No],[Item_Code],[Notes],[QTY]
            ,[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
            ('OR_Dt_005','Ord_002','Item003','very Hot ',2
            ,GETDATE(),'SHANU' ,GETDATE(),'SHANU')

INSERT INTO [OrderDetails]
            ([Order_Detail_No],[Order_No],[Item_Code],[Notes],[QTY]
            ,[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
            ('OR_Dt_006','Ord_003','Item003','Very Spicy',4
            ,GETDATE(),'SHANU' ,GETDATE(),'SHANU')
```

Simple SQL Select Query

Can someone tell me what a Select Query in SQL is? A Select is the one basic and most important DML statement of SQL. So what is DML in SQL? DML stands for Data Manipulation Language. That means these statements are used to manipulate the data that already exists. As the name suggests the SELECT statement selects one or more columns with one or more data to display from our DB with our optional filters.

For example now I want to display my Name in the SQL Server. So as a result I will use the select statement here as an example.

```sql
SELECT 'My Name Is SYED SHANU'
-- With Column Name using 'AS'
SELECT 'My Name Is SYED SHANU' as 'MY NAME'
-- With more then the one Column
SELECT 'My Name' as 'Column1', 'Is' as 'Column2', 'SYED SHANU' as 'Column3'
```



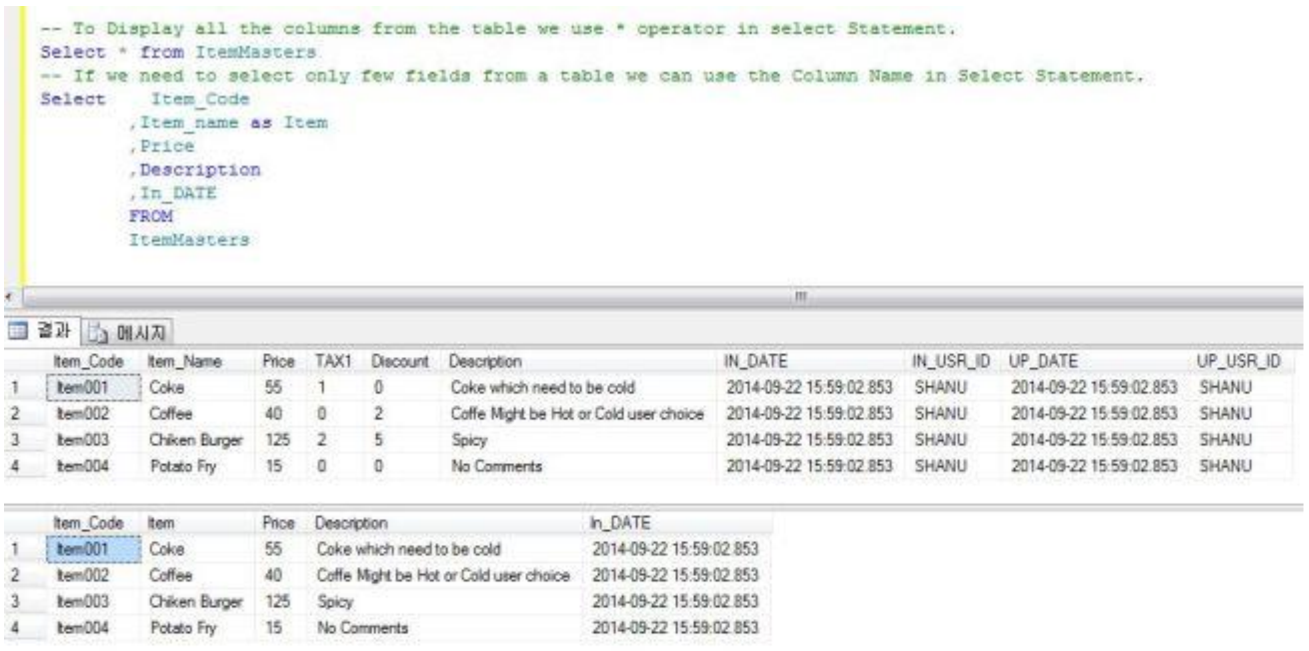
```sql
SELECT 'My Name Is SYED SHANU'
-- With Column Name using 'AS'
SELECT 'My Name Is SYED SHANU' as 'MY NAME'
-- With more then the one Column
SELECT 'My Name' as 'Column1', 'Is' as 'Column2', 'SYED SHANU' as 'Column3'
```

결과 | 메시지

```
   (열 이름 없음)
1  My Name Is SYED SHANU

   MY NAME
1  My Name Is SYED SHANU

   Column1   Column2  Column3
1  My Name   Is       SYED SHANU
```

Select Statement from Table

```
-- To Display all the columns from the table we use * operator in select Statement.
Select * from ItemMasters
-- If we need to select only few fields from a table we can use the Column Name in Select Statement.
Select     Item_Code
         ,Item_name as Item
         ,Price
         ,Description
         ,In_DATE
         FROM
         ItemMasters
```



```
-- To Display all the columns from the table we use * operator in select Statement.
Select * from ItemMasters
-- If we need to select only few fields from a table we can use the Column Name in Select Statement.
Select     Item_Code
         ,Item_name as Item
         ,Price
         ,Description
         ,In_DATE
         FROM
         ItemMasters
```

	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
3	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
4	Item004	Potato Fry	15	0	0	No Comments	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

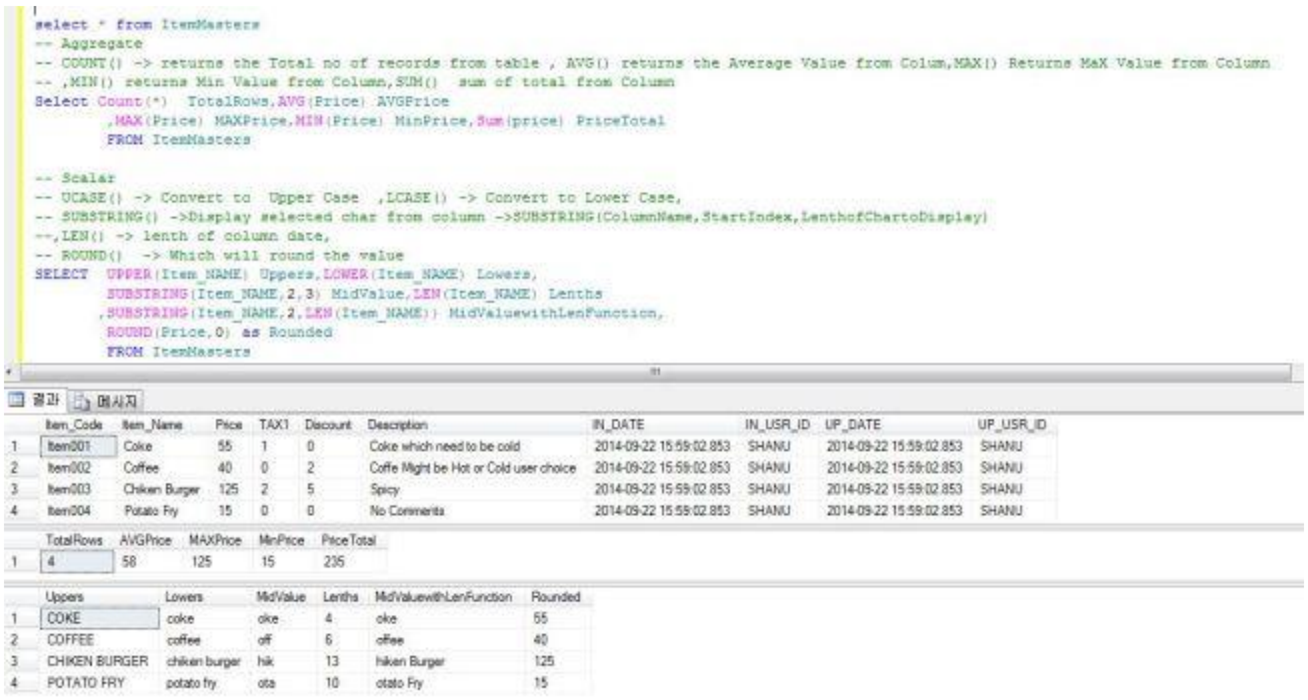
	Item_Code	Item	Price	Description	In_DATE
1	Item001	Coke	55	Coke which need to be cold	2014-09-22 15:59:02.853
2	Item002	Coffee	40	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853
3	Item003	Chiken Burger	125	Spicy	2014-09-22 15:59:02.853
4	Item004	Potato Fry	15	No Comments	2014-09-22 15:59:02.853

Simple Aggregate and Scalar Function

Aggregate and Scalar Functions are built-in functions of SQL Server that we can be used in our select statements. For example a few Aggregate and Scalar functions are Count, Max, Sum, Upper, lower, Round and so on. Here in comments I have explained each of its uses.

We can use our tblItemMaster with all these built-in functions.

```
select * from ItemMasters
-- Aggregate
-- COUNT() -
> returns the Total no of records from table , AVG() returns the Average Value from Col
um,MAX() Returns MaX Value from Column
-- ,MIN() returns Min Value from Column,SUM()  sum of total from Column
Select Count(*)  TotalRows,AVG(Price) AVGPrice
        ,MAX(Price) MAXPrice,MIN(Price) MinPrice,Sum(price) PriceTotal
        FROM ItemMasters

-- Scalar
-- UCASE() -> Convert to  Upper Case  ,LCASE() -> Convert to Lower Case,
-- SUBSTRING() ->Display selected char from column -
>SUBSTRING(ColumnName,StartIndex,LenthofChartoDisplay)
--,LEN() -> lenth of column date,
-- ROUND()  -> Which will round the value
SELECT  UPPER(Item_NAME) Uppers,LOWER(Item_NAME) Lowers,
        SUBSTRING(Item_NAME,2,3) MidValue,LEN(Item_NAME) Lenths
       ,SUBSTRING(Item_NAME,2,LEN(Item_NAME)) MidValuewithLenFunction,
        ROUND(Price,0) as Rounded
        FROM ItemMasters
```



```
select * from ItemMasters
-- Aggregate
-- COUNT() -> returns the Total no of records from table , AVG() returns the Average Value from Colum,MAX() Returns MaX Value from Column
-- ,MIN() returns Min Value from Column,SUM()  sum of total from Column
Select Count(*)  TotalRows,AVG(Price) AVGPrice
         ,MAX(Price) MAXPrice,MIN(Price) MinPrice,Sum(price) PriceTotal
         FROM ItemMasters

-- Scalar
-- UCASE() -> Convert to  Upper Case  ,LCASE() -> Convert to Lower Case,
-- SUBSTRING() ->Display selected char from column ->SUBSTRING(ColumnName,StartIndex,LenthofChartoDisplay)
--,LEN() -> lenth of column date,
-- ROUND()  -> Which will round the value
SELECT  UPPER(Item_NAME) Uppers,LOWER(Item_NAME) Lowers,
         SUBSTRING(Item_NAME,2,3) MidValue,LEN(Item_NAME) Lenths
        ,SUBSTRING(Item_NAME,2,LEN(Item_NAME)) MidValuewithLenFunction,
         ROUND(Price,0) as Rounded
         FROM ItemMasters
```

	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
3	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
4	Item004	Potato Fry	15	0	0	No Comments	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

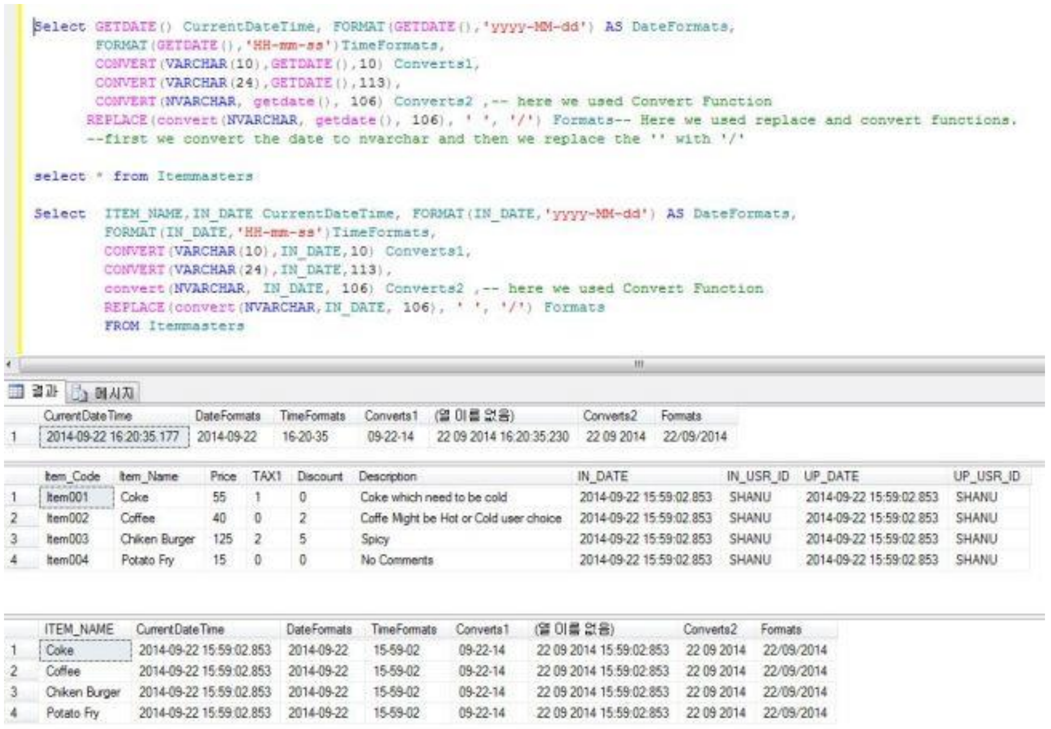
	TotalRows	AVGPrice	MAXPrice	MinPrice	PriceTotal
1	4	58	125	15	235

	Uppers	Lowers	MidValue	Lenths	MidValuewithLenFunction	Rounded
1	COKE	coke	oke	4	oke	55
2	COFFEE	coffee	off	6	offee	40
3	CHIKEN BURGER	chiken burger	hik	13	hiken Burger	125
4	POTATO FRY	potato fry	ota	10	otato Fry	15

Date Function

In all our projects and tables we use the date column. The Date function plays a very important role in all our projects. So we should be very careful with date functions since sometimes these functions will be trickier. We need to select appropriate date functions and formats in our projects. Let's see a few examples of those here.

```
-- GETDATE() -> to Display the Current Date and Time
-- Format() -> used to display our date in our requested format
Select GETDATE() CurrentDateTime, FORMAT(GETDATE(),'yyyy-MM-dd') AS DateFormats,
       FORMAT(GETDATE(),'HH-mm-ss')TimeFormats,
       CONVERT(VARCHAR(10),GETDATE(),10) Converts1,
       CONVERT(VARCHAR(24),GETDATE(),113),
       CONVERT(NVARCHAR, getdate(), 106) Converts2 ,-- here we used Convert Function
      REPLACE(convert(NVARCHAR, getdate(), 106), ' ', '/') Formats--
 Here we used replace and --convert functions.
      --first we convert the date to nvarchar and then we replace the '' with '/'

select * from Itemmasters

Select  ITEM_NAME,IN_DATE CurrentDateTime, FORMAT(IN_DATE,'yyyy-MM-
dd') AS DateFormats,
        FORMAT(IN_DATE,'HH-mm-ss')TimeFormats,
        CONVERT(VARCHAR(10),IN_DATE,10) Converts1,
        CONVERT(VARCHAR(24),IN_DATE,113),
        convert(NVARCHAR, IN_DATE, 106) Converts2 ,-- here we used Convert Function
        REPLACE(convert(NVARCHAR,IN_DATE, 106), ' ', '/') Formats
        FROM Itemmasters
```



```sql
Select GETDATE() CurrentDateTime, FORMAT(GETDATE(),'yyyy-MM-dd') AS DateFormats,
       FORMAT(GETDATE(),'HH-mm-ss')TimeFormats,
       CONVERT(VARCHAR(10),GETDATE(),10) Converts1,
       CONVERT(VARCHAR(24),GETDATE(),113),
       CONVERT(NVARCHAR, getdate(), 106) Converts2 ,-- here we used Convert Function
      REPLACE(convert(NVARCHAR, getdate(), 106), ' ', '/') Formats-- Here we used replace and convert functions.
      --first we convert the date to nvarchar and then we replace the '' with '/'

select * from Itemmasters

Select  ITEM_NAME,IN_DATE CurrentDateTime, FORMAT(IN_DATE,'yyyy-MM-dd') AS DateFormats,
        FORMAT(IN_DATE,'HH-mm-ss')TimeFormats,
        CONVERT(VARCHAR(10),IN_DATE,10) Converts1,
        CONVERT(VARCHAR(24),IN_DATE,113),
        convert(NVARCHAR, IN_DATE, 106) Converts2 ,-- here we used Convert Function
        REPLACE(convert(NVARCHAR,IN_DATE, 106), ' ', '/') Formats
        FROM Itemmasters
```

결과 메시지

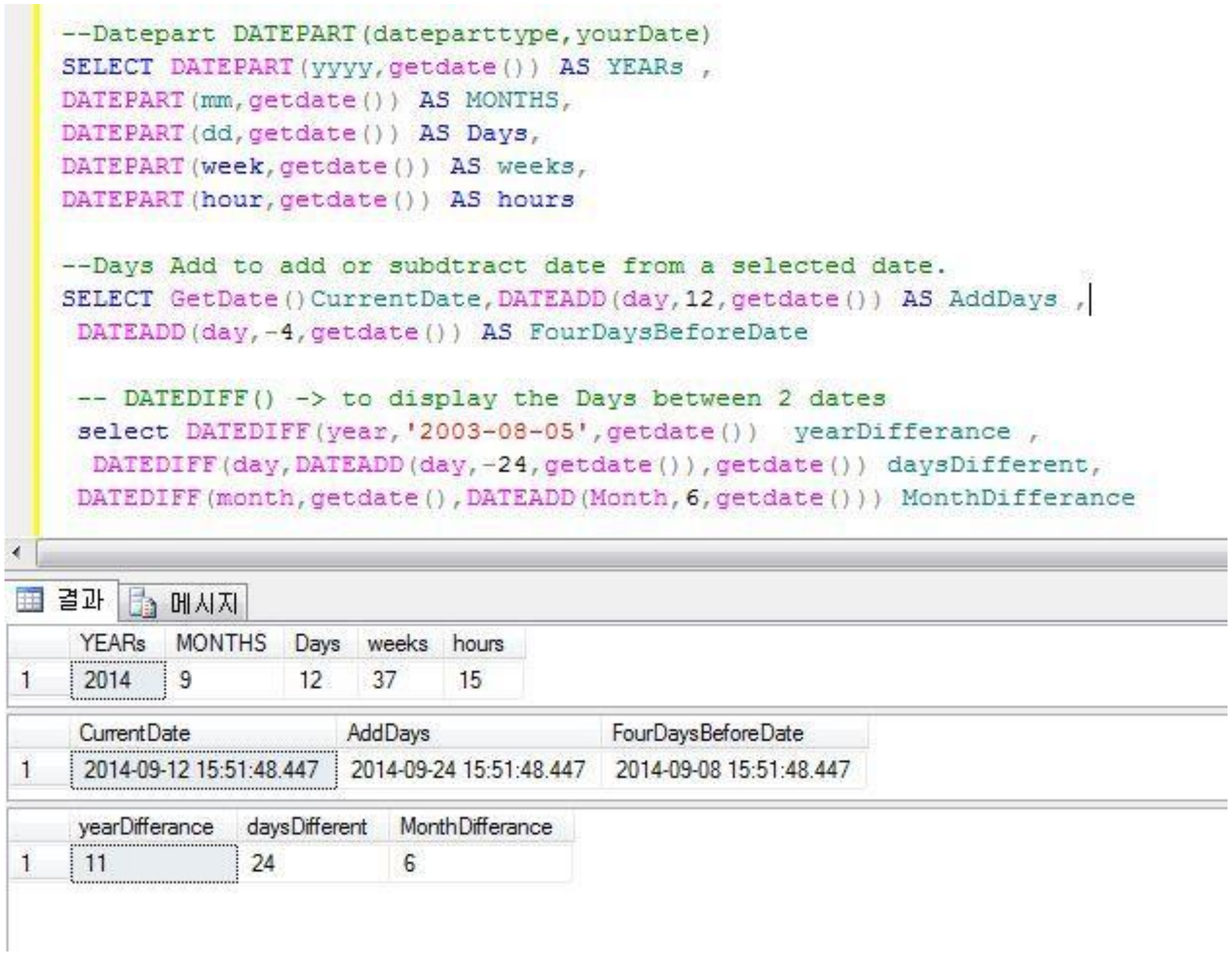
	CurrentDateTime	DateFormats	TimeFormats	Converts1	(열 이름 없음)	Converts2	Formats
1	2014-09-22 16:20:35.177	2014-09-22	16-20-35	09-22-14	22 09 2014 16:20:35:230	22 09 2014	22/09/2014

	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
3	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
4	Item004	Potato Fry	15	0	0	No Comments	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

	ITEM_NAME	CurrentDateTime	DateFormats	TimeFormats	Converts1	(열 이름 없음)	Converts2	Formats
1	Coke	2014-09-22 15:59:02.853	2014-09-22	15-59-02	09-22-14	22 09 2014 15:59:02:853	22 09 2014	22/09/2014
2	Coffee	2014-09-22 15:59:02.853	2014-09-22	15-59-02	09-22-14	22 09 2014 15:59:02:853	22 09 2014	22/09/2014
3	Chiken Burger	2014-09-22 15:59:02.853	2014-09-22	15-59-02	09-22-14	22 09 2014 15:59:02:853	22 09 2014	22/09/2014
4	Potato Fry	2014-09-22 15:59:02.853	2014-09-22	15-59-02	09-22-14	22 09 2014 15:59:02:853	22 09 2014	22/09/2014

- **DatePart:** We use the datepart to display the selected Year, Month and Day.
- **DateADD:** We use dateadd to display or add or subrtract days from the selected date.
- **DateDIff:** We use dateDiff to calculate the difference between 2 dates.

```sql
--Datepart DATEPART(dateparttype,yourDate)
SELECT DATEPART(yyyy,getdate()) AS YEARs ,
DATEPART(mm,getdate()) AS MONTHS,
DATEPART(dd,getdate()) AS Days,
DATEPART(week,getdate()) AS weeks,
DATEPART(hour,getdate()) AS hours

--Days Add to add or subdtract date from a selected date.
SELECT GetDate()CurrentDate,DATEADD(day,12,getdate()) AS AddDays ,
 DATEADD(day,-4,getdate()) AS FourDaysBeforeDate

 -- DATEDIFF() -> to display the Days between 2 dates
 select DATEDIFF(year,'2003-08-05',getdate())  yearDifferance ,
  DATEDIFF(day,DATEADD(day,-24,getdate()),getdate()) daysDifferent,
 DATEDIFF(month,getdate(),DATEADD(Month,6,getdate())) MonthDifferance
```



```
--Datepart DATEPART(dateparttype,yourDate)
SELECT DATEPART(yyyy,getdate()) AS YEARs ,
DATEPART(mm,getdate()) AS MONTHS,
DATEPART(dd,getdate()) AS Days,
DATEPART(week,getdate()) AS weeks,
DATEPART(hour,getdate()) AS hours

--Days Add to add or subdtract date from a selected date.
SELECT GetDate()CurrentDate,DATEADD(day,12,getdate()) AS AddDays ,
 DATEADD(day,-4,getdate()) AS FourDaysBeforeDate

 -- DATEDIFF() -> to display the Days between 2 dates
 select DATEDIFF(year,'2003-08-05',getdate())  yearDifferance ,
  DATEDIFF(day,DATEADD(day,-24,getdate()),getdate()) daysDifferent,
 DATEDIFF(month,getdate(),DATEADD(Month,6,getdate())) MonthDifferance
```

	YEARs	MONTHS	Days	weeks	hours
1	2014	9	12	37	15

	CurrentDate	AddDays	FourDaysBeforeDate
1	2014-09-12 15:51:48.447	2014-09-24 15:51:48.447	2014-09-08 15:51:48.447

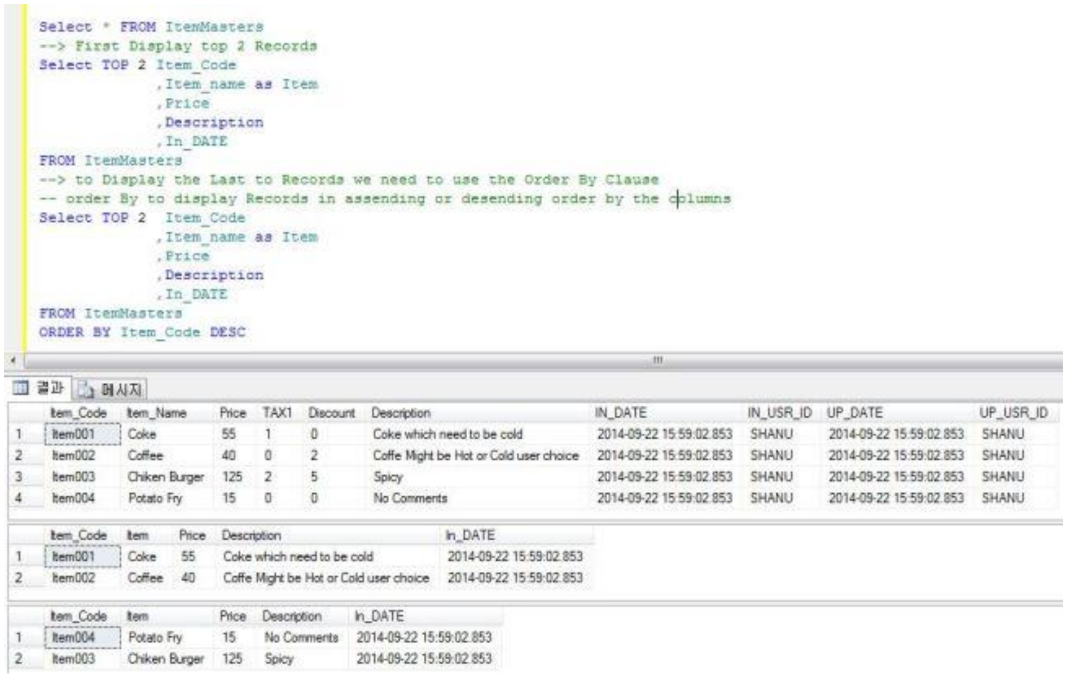
	yearDifferance	daysDifferent	MonthDifferance
1	11	24	6

Other Select Function

Top: To display the Top First or Last selected records first we see how to select the first or Top records and last to the Top records from the Select statement.

Order By: This is used in a SQL Select statement to display the records in ascending or descending order by the columns.

```
<a name="5">
--Top to Select Top first and last records using Select Statement.
Select * FROM ItemMasters
--> First Display top 2 Records
Select TOP 2 Item_Code
             ,Item_name as Item
             ,Price
             ,Description
             ,In_DATE
FROM ItemMasters
--> to Display the Last to Records we need to use the Order By Clause
-- order By to display Records in assending or desending order by the columns
Select TOP 2  Item_Code
             ,Item_name as Item
             ,Price
             ,Description
             ,In_DATE
FROM ItemMasters
ORDER BY Item_Code DESC</a>
```



```
Select * FROM ItemMasters
--> First Display top 2 Records
Select TOP 2 Item_Code
              ,Item_name as Item
              ,Price
              ,Description
              ,In_DATE
FROM ItemMasters
--> to Display the Last to Records we need to use the Order By Clause
-- order By to display Records in assending or desending order by the columns
Select TOP 2  Item_Code
              ,Item_name as Item
              ,Price
              ,Description
              ,In_DATE
FROM ItemMasters
ORDER BY Item_Code DESC
```

결과 | 메시지

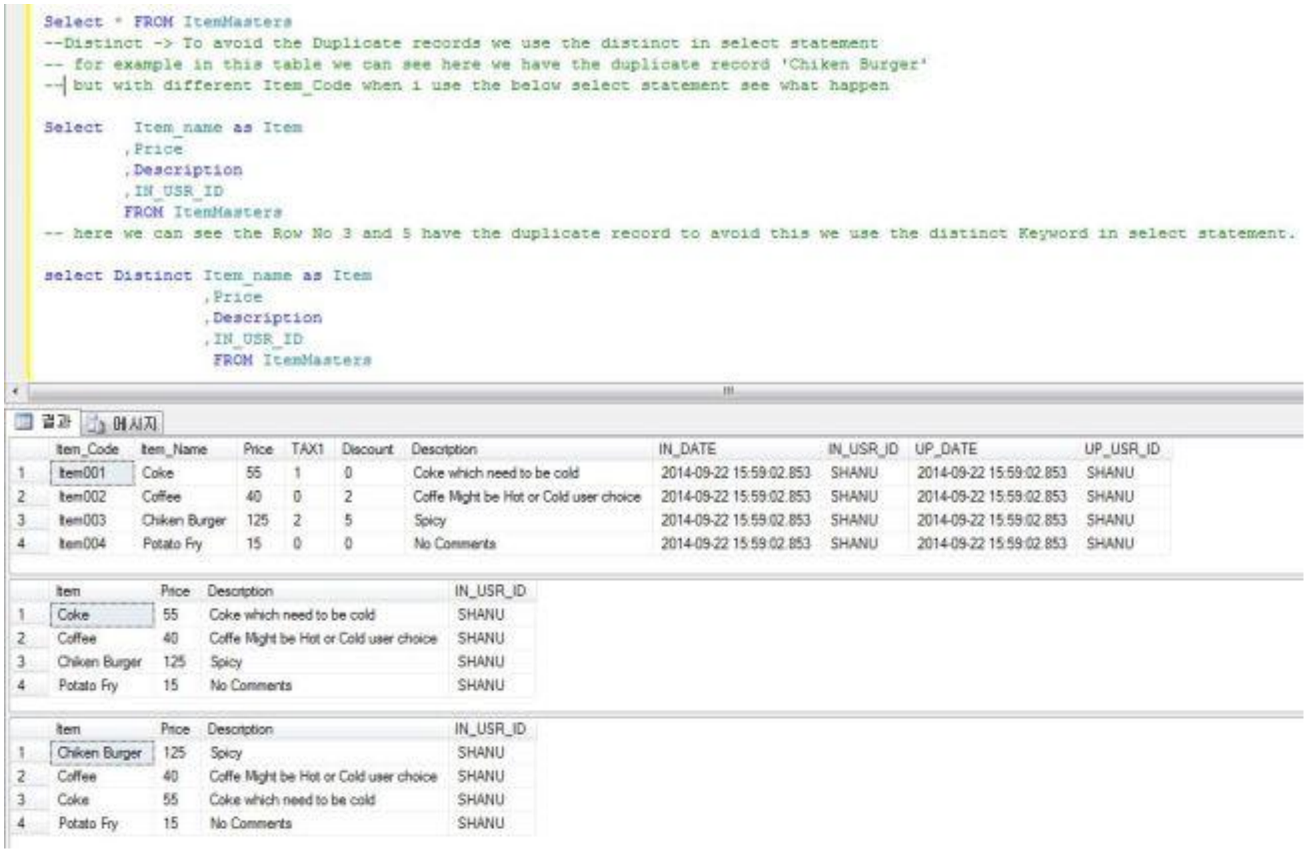
	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
3	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
4	Item004	Potato Fry	15	0	0	No Comments	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

	Item_Code	Item	Price	Description	In_DATE
1	Item001	Coke	55	Coke which need to be cold	2014-09-22 15:59:02.853
2	Item002	Coffee	40	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853

	Item_Code	Item	Price	Description	In_DATE
1	Item004	Potato Fry	15	No Comments	2014-09-22 15:59:02.853
2	Item003	Chiken Burger	125	Spicy	2014-09-22 15:59:02.853

Distinct: The distinct keyword is used in select statements to avoid the duplicate records.

```
<a name="5">
Select * FROM ItemMasters
--Distinct -> To avoid the Duplicate records we use the distinct in select statement
--
 for example in this table we can see here we have the duplicate record 'Chiken Burger'

-- but with different Item_Code when i use the below select statement see what happen

Select   Item_name as Item
        ,Price
        ,Description
        ,IN_USR_ID
        FROM ItemMasters
--
 here we can see the Row No 3 and 5 have the duplicate record to avoid this we use the
distinct Keyword in select statement.

select Distinct Item_name as Item
                ,Price
                ,Description
                ,IN_USR_ID
                 FROM ItemMasters</a>
```



```
Select * FROM ItemMasters
--Distinct -> To avoid the Duplicate records we use the distinct in select statement
-- for example in this table we can see here we have the duplicate record 'Chiken Burger'
-- but with different Item_Code when i use the below select statement see what happen

Select   Item_name as Item
        ,Price
        ,Description
        ,IN_USR_ID
        FROM ItemMasters
-- here we can see the Row No 3 and 5 have the duplicate record to avoid this we use the distinct Keyword in select statement.

select Distinct Item_name as Item
                ,Price
                ,Description
                ,IN_USR_ID
                 FROM ItemMasters
```

결과 | 메시지

	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
3	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
4	Item004	Potato Fry	15	0	0	No Comments	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

	Item	Price	Description	IN_USR_ID
1	Coke	55	Coke which need to be cold	SHANU
2	Coffee	40	Coffe Might be Hot or Cold user choice	SHANU
3	Chiken Burger	125	Spicy	SHANU
4	Potato Fry	15	No Comments	SHANU

	Item	Price	Description	IN_USR_ID
1	Chiken Burger	125	Spicy	SHANU
2	Coffee	40	Coffe Might be Hot or Cold user choice	SHANU
3	Coke	55	Coke which need to be cold	SHANU
4	Potato Fry	15	No Comments	SHANU

Where Clause

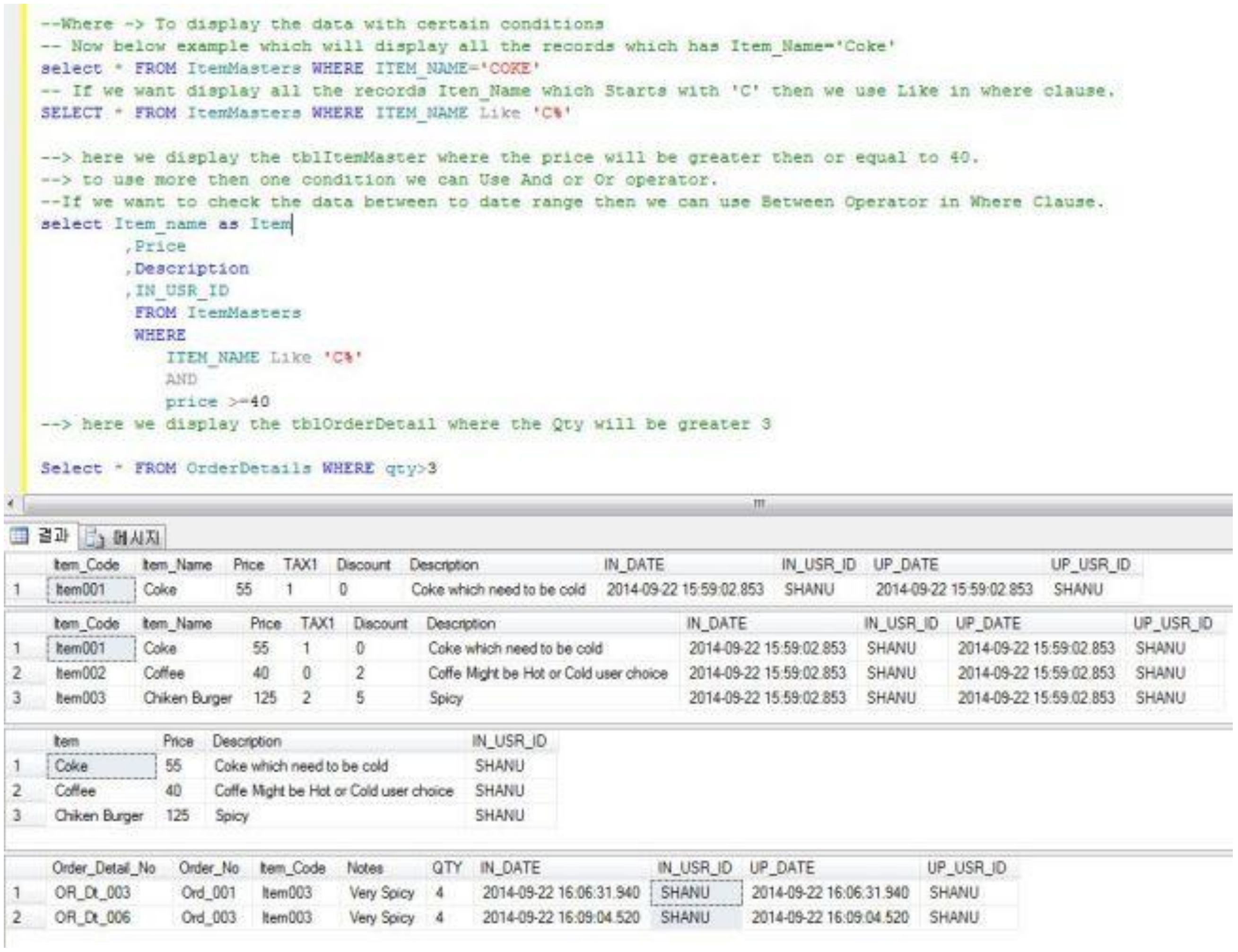
The where clause is very important in SQL Select statements. Why we use a where clause and what is use of where clause. Where clause is nothing but a filter of the records using some condition.

Now for example we consider a table has 10,000 records. If we use a select query to display the records then the load time might be long to display all the data. In that case we can use a condition and display some specific data.

For example we can use the Where Clause to display the records for the past 1 week or 1 month.

We can consider our restaurant Order Detail table, for example we consider an Order Detail table to have 10,000 records but for sure for one Order_No there will not be more than 100 records in the Order Detail. Now let's see a few where clause uses.

```
Select * from ItemMasters
Select * from OrderDetails
--Where -> To display the data with certain conditions
-- Now below example which will display all the records which has Item_Name='Coke'
select * FROM ItemMasters WHERE ITEM_NAME='COKE'
--
 If we want display all the records Iten_Name which Starts with 'C' then we use Like in
 where clause.
SELECT * FROM ItemMasters WHERE ITEM_NAME Like 'C%'

--
> here we display the ItemMasters where the price will be greater then or equal to 40.

--> to use more then one condition we can Use And or Or operator.
--
If we want to check the data between to date range then we can use Between Operator in
Where Clause.
select Item_name as Item
        ,Price
        ,Description
        ,IN_USR_ID
         FROM ItemMasters
         WHERE
            ITEM_NAME Like 'C%'
            AND
            price >=40
--> here we display the OrderDetails where the Qty will be greater 3

Select * FROM OrderDetails WHERE qty>3
```



```
--Where -> To display the data with certain conditions
-- Now below example which will display all the records which has Item_Name='Coke'
select * FROM ItemMasters WHERE ITEM_NAME='COKE'
-- If we want display all the records Iten_Name which Starts with 'C' then we use Like in where clause.
SELECT * FROM ItemMasters WHERE ITEM_NAME Like 'C%'

--> here we display the tblItemMaster where the price will be greater then or equal to 40.
--> to use more then one condition we can Use And or Or operator.
--If we want to check the data between to date range then we can use Between Operator in Where Clause.
select Item_name as Item
         ,Price
         ,Description
         ,IN_USR_ID
          FROM ItemMasters
          WHERE
             ITEM_NAME Like 'C%'
             AND
             price >=40
--> here we display the tblOrderDetail where the Qty will be greater 3

Select * FROM OrderDetails WHERE qty>3
```

결과 | 메시지

	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

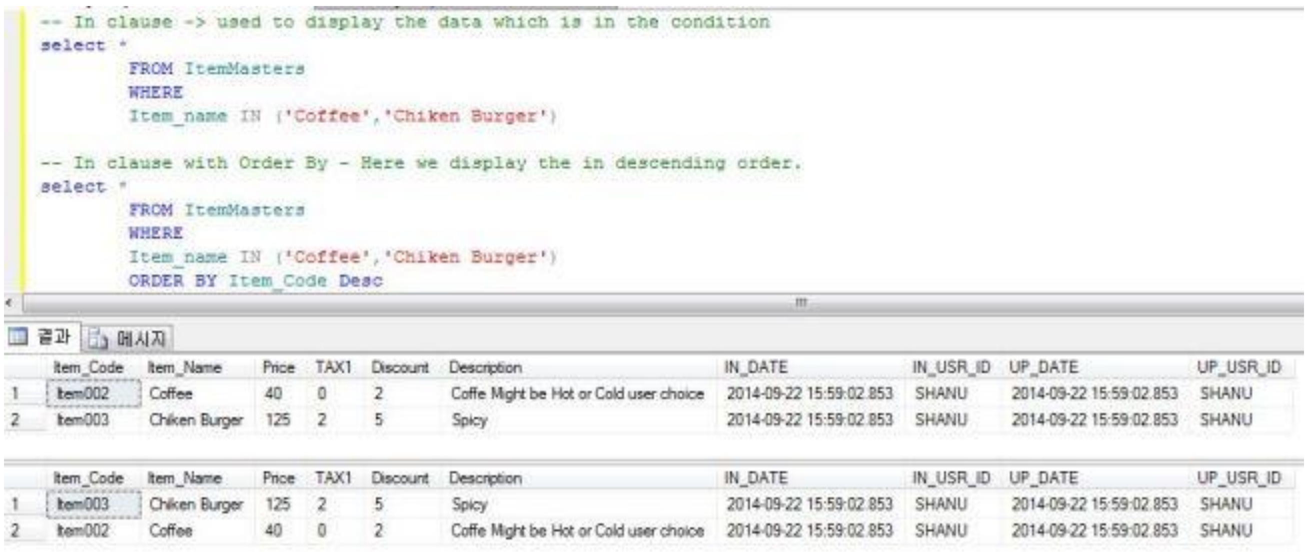
	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
3	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

	Item	Price	Description	IN_USR_ID
1	Coke	55	Coke which need to be cold	SHANU
2	Coffee	40	Coffe Might be Hot or Cold user choice	SHANU
3	Chiken Burger	125	Spicy	SHANU

	Order_Detail_No	Order_No	Item_Code	Notes	QTY	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	OR_Dt_003	Ord_001	Item003	Very Spicy	4	2014-09-22 16:06:31.940	SHANU	2014-09-22 16:06:31.940	SHANU
2	OR_Dt_006	Ord_003	Item003	Very Spicy	4	2014-09-22 16:09:04.520	SHANU	2014-09-22 16:09:04.520	SHANU

Where In Clause

```
-- In clause -> used to display the data which is in the condition
select *
        FROM ItemMasters
        WHERE
        Item_name IN ('Coffee','Chiken Burger')

-- In clause with Order By - Here we display the in descending order.
select *
        FROM ItemMasters
        WHERE
        Item_name IN ('Coffee','Chiken Burger')
        ORDER BY Item_Code Desc
```



```
-- In clause -> used to display the data which is in the condition
select *
        FROM ItemMasters
        WHERE
        Item_name IN ('Coffee','Chiken Burger')

-- In clause with Order By - Here we display the in descending order.
select *
        FROM ItemMasters
        WHERE
        Item_name IN ('Coffee','Chiken Burger')
        ORDER BY Item_Code Desc
```

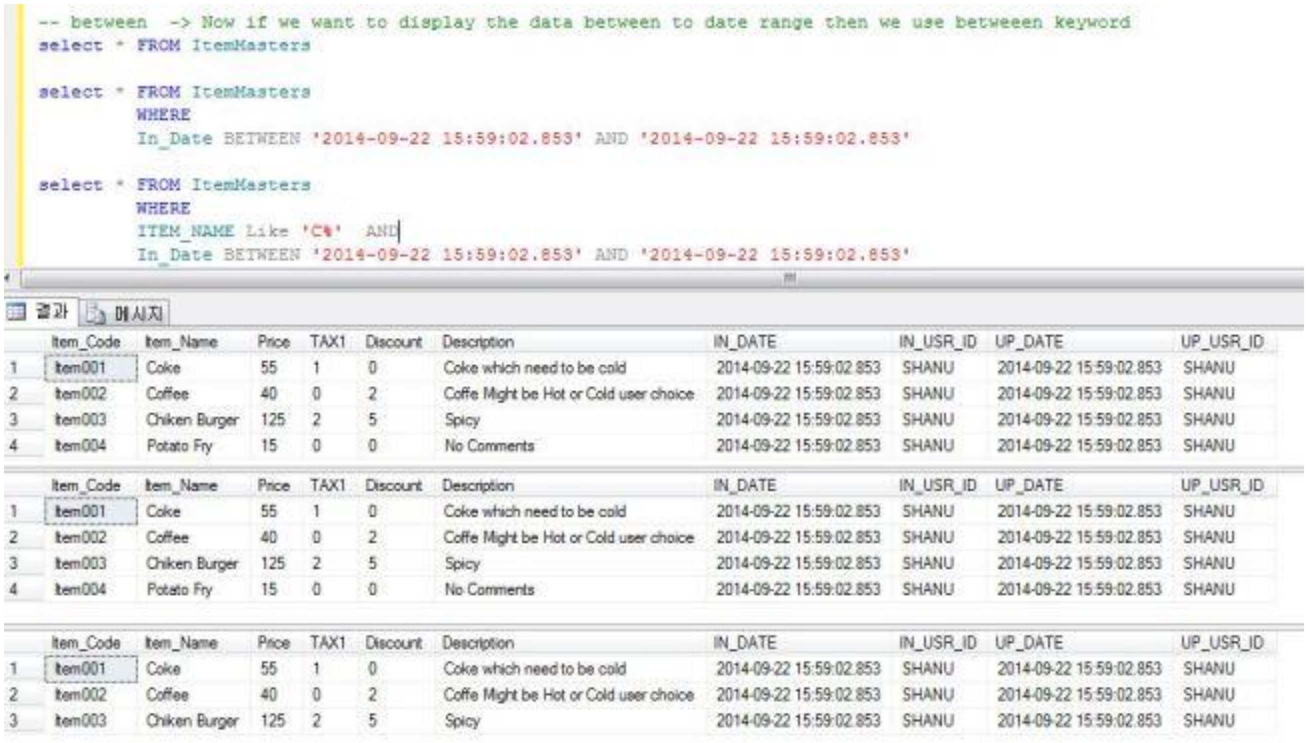
결과 | 메시지

	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

Where Between keyword

```
-- between  -
> Now if we want to display the data between to date range then we use betweeen keyword

select * FROM ItemMasters

select * FROM ItemMasters
        WHERE
        In_Date BETWEEN '2014-09-22 15:59:02.853' AND '2014-09-22 15:59:02.853'

select * FROM ItemMasters
        WHERE
        ITEM_NAME Like 'C%'
        AND
        In_Date BETWEEN '2014-09-22 15:59:02.853' AND '2014-09-22 15:59:02.853'
```



```
-- between  -> Now if we want to display the data between to date range then we use betweeen keyword
select * FROM ItemMasters

select * FROM ItemMasters
         WHERE
         In_Date BETWEEN '2014-09-22 15:59:02.853' AND '2014-09-22 15:59:02.853'

select * FROM ItemMasters
         WHERE
         ITEM_NAME Like 'C%'  AND
         In_Date BETWEEN '2014-09-22 15:59:02.853' AND '2014-09-22 15:59:02.853'
```

	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
3	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
4	Item004	Potato Fry	15	0	0	No Comments	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
3	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
4	Item004	Potato Fry	15	0	0	No Comments	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

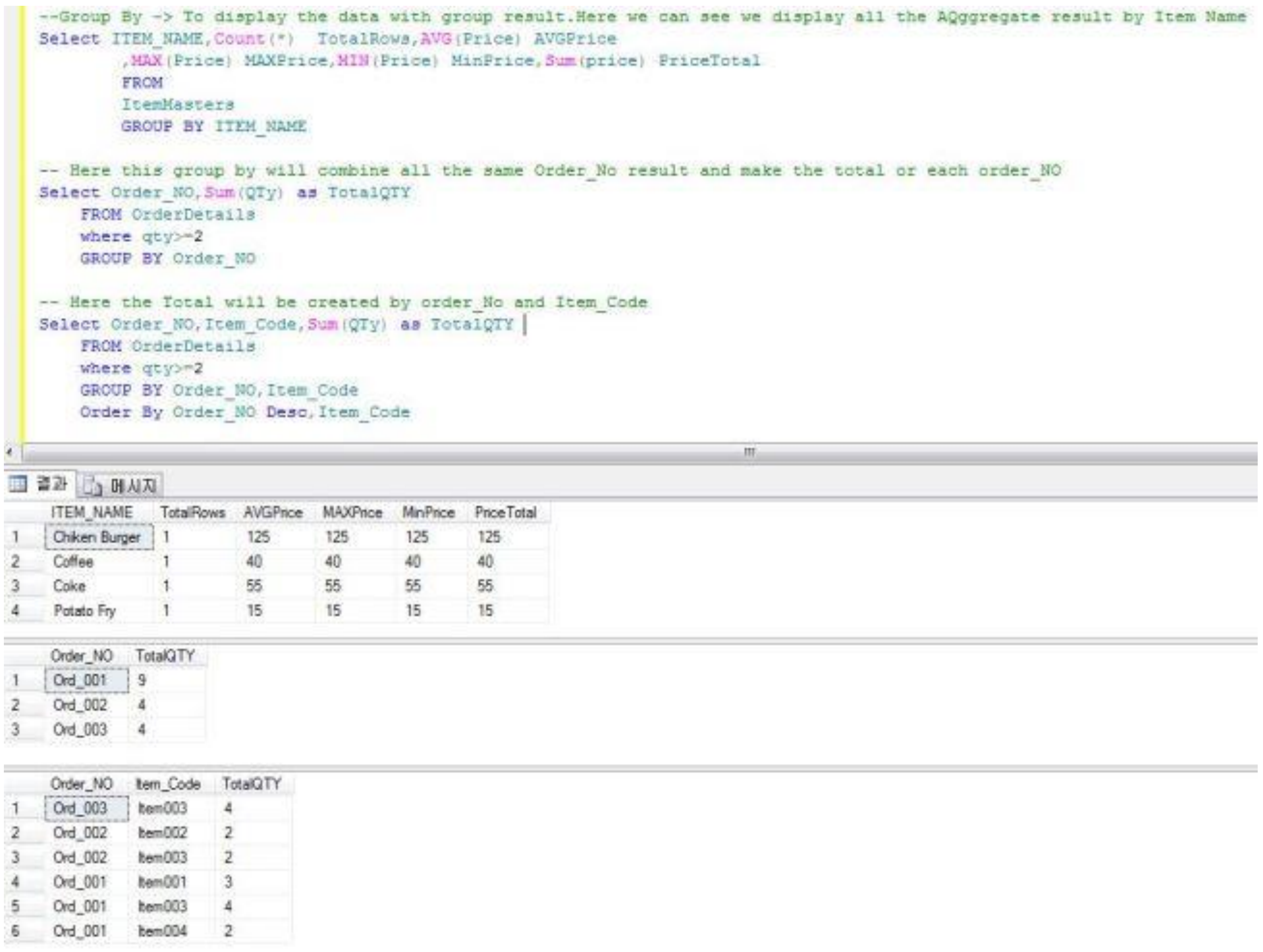
	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
3	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

To display the records within a range we use the between keyword in the where clause.

Group BY Clause

A Group by Clause will display records as a result of a combined set or result of the same related records.

```
--Group By -
> To display the data with group result.Here we can see we display all the AQggregate r
esult by Item Name
Select ITEM_NAME,Count(*)  TotalRows,AVG(Price) AVGPrice
        ,MAX(Price) MAXPrice,MIN(Price) MinPrice,Sum(price) PriceTotal
        FROM
        ItemMasters
        GROUP BY ITEM_NAME

--
 Here this group by will combine all the same Order_No result and make the total or eac
h order_NO
Select Order_NO,Sum(QTy) as TotalQTY
    FROM OrderDetails
    where qty>=2
    GROUP BY Order_NO

-- Here the Total will be created by order_No and Item_Code
Select Order_NO,Item_Code,Sum(QTy) as TotalQTY
    FROM OrderDetails
    where qty>=2
    GROUP BY Order_NO,Item_Code
    Order By Order_NO Desc,Item_Code
```



```
--Group By -> To display the data with group result.Here we can see we display all the AQggregate result by Item Name
Select ITEM_NAME,Count(*)  TotalRows,AVG(Price) AVGPrice
        ,MAX(Price) MAXPrice,MIN(Price) MinPrice,Sum(price) PriceTotal
        FROM
        ItemMasters
        GROUP BY ITEM_NAME

-- Here this group by will combine all the same Order_No result and make the total or each order_NO
Select Order_NO,Sum(QTy) as TotalQTY
    FROM OrderDetails
    where qty>=2
    GROUP BY Order_NO

-- Here the Total will be created by order_No and Item_Code
Select Order_NO,Item_Code,Sum(QTy) as TotalQTY
    FROM OrderDetails
    where qty>=2
    GROUP BY Order_NO,Item_Code
    Order By Order_NO Desc,Item_Code
```

결과 | 메시지

	ITEM_NAME	TotalRows	AVGPrice	MAXPrice	MinPrice	PriceTotal
1	Chiken Burger	1	125	125	125	125
2	Coffee	1	40	40	40	40
3	Coke	1	55	55	55	55
4	Potato Fry	1	15	15	15	15

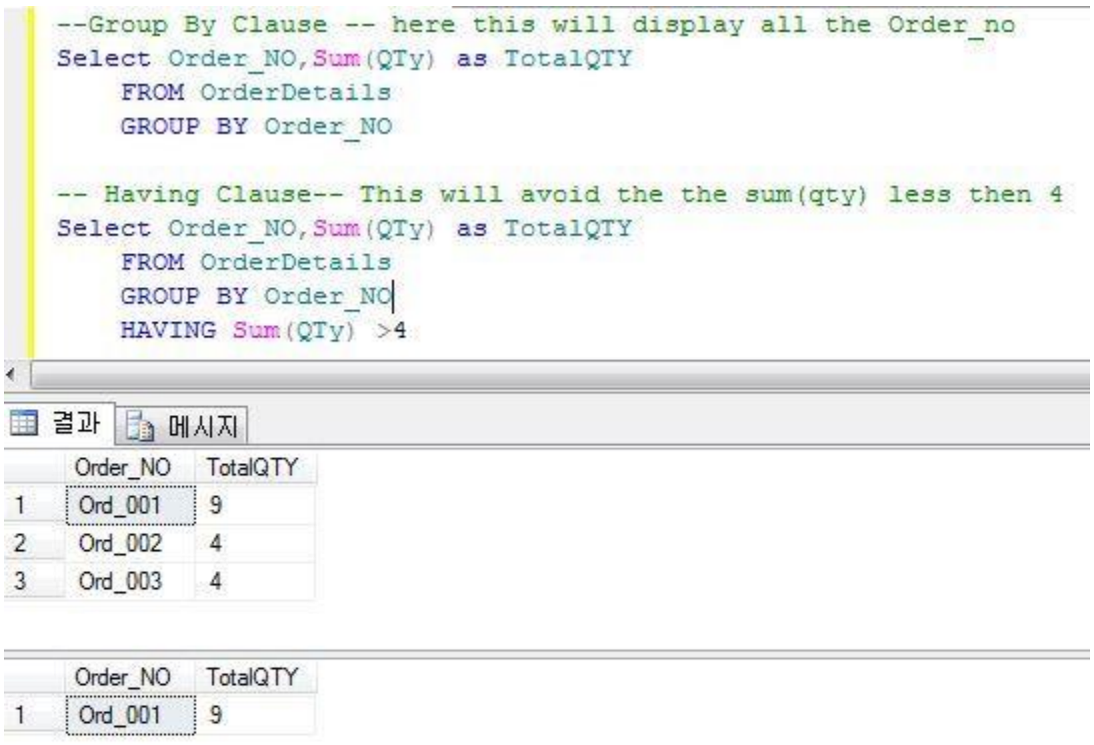
	Order_NO	TotalQTY
1	Ord_001	9
2	Ord_002	4
3	Ord_003	4

	Order_NO	Item_Code	TotalQTY
1	Ord_003	Item003	4
2	Ord_002	Item002	2
3	Ord_002	Item003	2
4	Ord_001	Item001	3
5	Ord_001	Item003	4
6	Ord_001	Item004	2

Group By and Having Clause

The Having and Group By clauses don't support aggregate functions. To use aggregate functions we use a having clause in a Select statement.

```
--Group By Clause -- here this will display all the Order_no
Select Order_NO,Sum(QTy) as TotalQTY
    FROM OrderDetails
    GROUP BY Order_NO

-- Having Clause-- This will avoid the the sum(qty) less then 4
Select Order_NO,Sum(QTy) as TotalQTY
    FROM OrderDetails
    GROUP BY Order_NO
    HAVING Sum(QTy) >4
```



```
--Group By Clause -- here this will display all the Order_no
Select Order_NO,Sum(QTy) as TotalQTY
    FROM OrderDetails
    GROUP BY Order_NO

-- Having Clause-- This will avoid the the sum(qty) less then 4
Select Order_NO,Sum(QTy) as TotalQTY
    FROM OrderDetails
    GROUP BY Order_NO
    HAVING Sum(QTy) >4
```

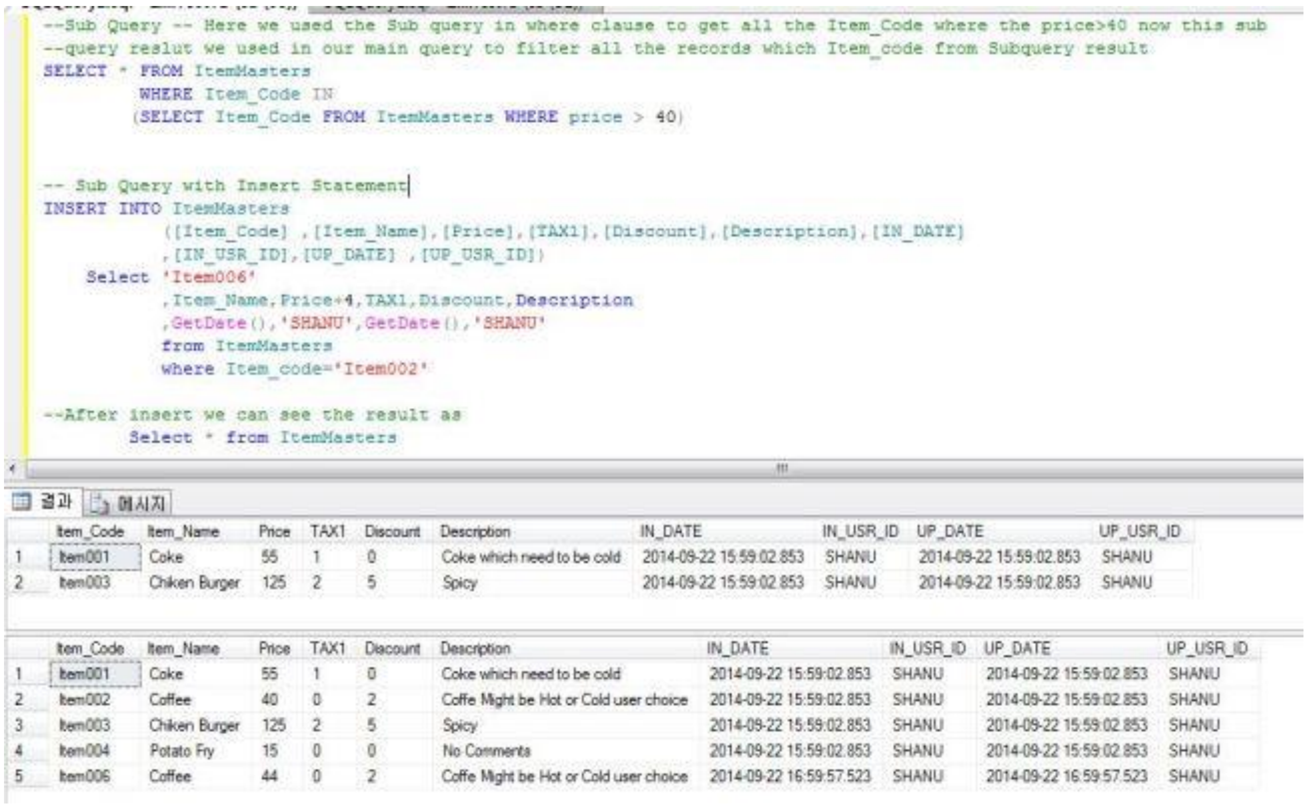
```
   Order_NO  TotalQTY
1  Ord_001   9
2  Ord_002   4
3  Ord_003   4
```

```
   Order_NO  TotalQTY
1  Ord_001   9
```

Sub Query

A Sub Query can be called an Inner Query or nested query that can be used in a Where clause of a select, insert, update or delete statement.

```
--Sub Query --
 Here we used the Sub query in where clause to get all the Item_Code where the price>40
 now this sub
--
query reslut we used in our main query to filter all the records which Item_code from S
ubquery result
SELECT * FROM ItemMasters
         WHERE Item_Code IN
        (SELECT Item_Code FROM ItemMasters WHERE price > 40)


-- Sub Query with Insert Statement
INSERT INTO ItemMasters           ([Item_Code] ,[Item_Name],[Price],[TAX1],[Discount],[
Description],[IN_DATE]
           ,[IN_USR_ID],[UP_DATE] ,[UP_USR_ID])
    Select 'Item006'
           ,Item_Name,Price+4,TAX1,Discount,Description
           ,GetDate(),'SHANU',GetDate(),'SHANU'
           from ItemMasters
           where Item_code='Item002'

--After insert we can see the result as
        Select * from ItemMasters
```



```
--Sub Query -- Here we used the Sub query in where clause to get all the Item_Code where the price>40 now this sub
--query reslut we used in our main query to filter all the records which Item_code from Subquery result
SELECT * FROM ItemMasters
         WHERE Item_Code IN
        (SELECT Item_Code FROM ItemMasters WHERE price > 40)


-- Sub Query with Insert Statement
INSERT INTO ItemMasters
           ([Item_Code] ,[Item_Name],[Price],[TAX1],[Discount],[Description],[IN_DATE]
           ,[IN_USR_ID],[UP_DATE] ,[UP_USR_ID])
    Select 'Item006'
           ,Item_Name,Price+4,TAX1,Discount,Description
           ,GetDate(),'SHANU',GetDate(),'SHANU'
           from ItemMasters
           where Item_code='Item002'

--After insert we can see the result as
        Select * from ItemMasters
```

결과 | 메시지

	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU

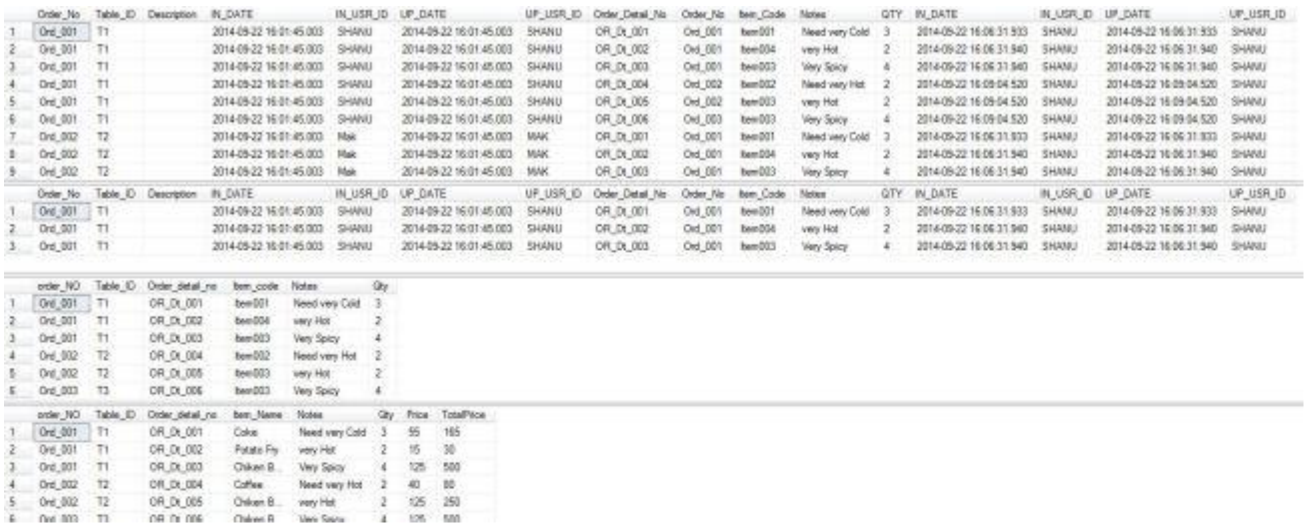
	Item_Code	Item_Name	Price	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Item001	Coke	55	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
2	Item002	Coffee	40	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
3	Item003	Chiken Burger	125	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
4	Item004	Potato Fry	15	0	0	No Comments	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU
5	Item006	Coffee	44	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 16:59:57.523	SHANU	2014-09-22 16:59:57.523	SHANU

Joins

So far we have seen all the something related to one table. Now we see how to join more than one table and display the results. Select statements will not be effective until we use a Join. The main purpose of using SQL Server is to use the normalization and increase the performance by displaying the records. With normalization we split large tables into small related tables. Now we need to display the related table data as a result of one single select statement. To accomplish this we use a Join in our SQL Select statement and combine all the data and display it.

Simple Join

```
--Now we have used the simple join with out any condition this will display all the
-- records with duplicate data to avaoid this we see our next example with condition
SELECT * FROM Ordermasters,OrderDetails
--
 Simple Join with Condition  now here we can see the duplicate records now has been avo
ided by using the where checing with both table primaryKey field
SELECT *
        FROM
        Ordermasters as M, OrderDetails as D
        where M.Order_NO=D.Order_NO
        and M.Order_NO='Ord_001'

-- Now to make more better understanding we need to select the need fields from both
--table insted of displaying all column.
SELECT M.order_NO,M.Table_ID,D.Order_detail_no,Item_code,Notes,Qty
                FROM
                Ordermasters as M, OrderDetails as D
                where M.Order_NO=D.Order_NO
 -- Now lets Join 3 table
 SELECT  M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,
                I.Price*D.Qty as TotalPrice
                FROM
                Ordermasters as M, OrderDetails as D,ItemMasters as I
                where
                M.Order_NO=D.Order_NO AND D.Item_Code=I.Item_Code
```



	Order_No	Table_ID	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID	Order_Detail_No	Order_No	Item_Code	Notes	QTY	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Ord_001	T1		2014-09-22 16:01:45.003	SHANU	2014-09-22 16:01:45.003	SHANU	OR_Dt_001	Ord_001	Item001	Need very Cold	3	2014-09-22 16:06:31.933	SHANU	2014-09-22 16:06:31.933	SHANU
2	Ord_001	T1		2014-09-22 16:01:45.003	SHANU	2014-09-22 16:01:45.003	SHANU	OR_Dt_002	Ord_001	Item004	very Hot	2	2014-09-22 16:06:31.940	SHANU	2014-09-22 16:06:31.940	SHANU
3	Ord_001	T1		2014-09-22 16:01:45.003	SHANU	2014-09-22 16:01:45.003	SHANU	OR_Dt_003	Ord_001	Item003	Very Spicy	4	2014-09-22 16:06:31.940	SHANU	2014-09-22 16:06:31.940	SHANU
4	Ord_001	T1		2014-09-22 16:01:45.003	SHANU	2014-09-22 16:01:45.003	SHANU	OR_Dt_004	Ord_002	Item002	Need very Hot	2	2014-09-22 16:09:04.520	SHANU	2014-09-22 16:09:04.520	SHANU
5	Ord_001	T1		2014-09-22 16:01:45.003	SHANU	2014-09-22 16:01:45.003	SHANU	OR_Dt_005	Ord_002	Item003	very Hot	2	2014-09-22 16:09:04.520	SHANU	2014-09-22 16:09:04.520	SHANU
6	Ord_001	T1		2014-09-22 16:01:45.003	SHANU	2014-09-22 16:01:45.003	SHANU	OR_Dt_006	Ord_003	Item003	Very Spicy	4	2014-09-22 16:09:04.520	SHANU	2014-09-22 16:09:04.520	SHANU
7	Ord_002	T2		2014-09-22 16:01:45.003	Mak	2014-09-22 16:01:45.003	MAK	OR_Dt_001	Ord_001	Item001	Need very Cold	3	2014-09-22 16:06:31.933	SHANU	2014-09-22 16:06:31.933	SHANU
8	Ord_002	T2		2014-09-22 16:01:45.003	Mak	2014-09-22 16:01:45.003	MAK	OR_Dt_002	Ord_001	Item004	very Hot	2	2014-09-22 16:06:31.940	SHANU	2014-09-22 16:06:31.940	SHANU
9	Ord_002	T2		2014-09-22 16:01:45.003	Mak	2014-09-22 16:01:45.003	MAK	OR_Dt_003	Ord_001	Item003	Very Spicy	4	2014-09-22 16:06:31.940	SHANU	2014-09-22 16:06:31.940	SHANU

	Order_No	Table_ID	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID	Order_Detail_No	Order_No	Item_Code	Notes	QTY	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID
1	Ord_001	T1		2014-09-22 16:01:45.003	SHANU	2014-09-22 16:01:45.003	SHANU	OR_Dt_001	Ord_001	Item001	Need very Cold	3	2014-09-22 16:06:31.933	SHANU	2014-09-22 16:06:31.933	SHANU
2	Ord_001	T1		2014-09-22 16:01:45.003	SHANU	2014-09-22 16:01:45.003	SHANU	OR_Dt_002	Ord_001	Item004	very Hot	2	2014-09-22 16:06:31.940	SHANU	2014-09-22 16:06:31.940	SHANU
3	Ord_001	T1		2014-09-22 16:01:45.003	SHANU	2014-09-22 16:01:45.003	SHANU	OR_Dt_003	Ord_001	Item003	Very Spicy	4	2014-09-22 16:06:31.940	SHANU	2014-09-22 16:06:31.940	SHANU

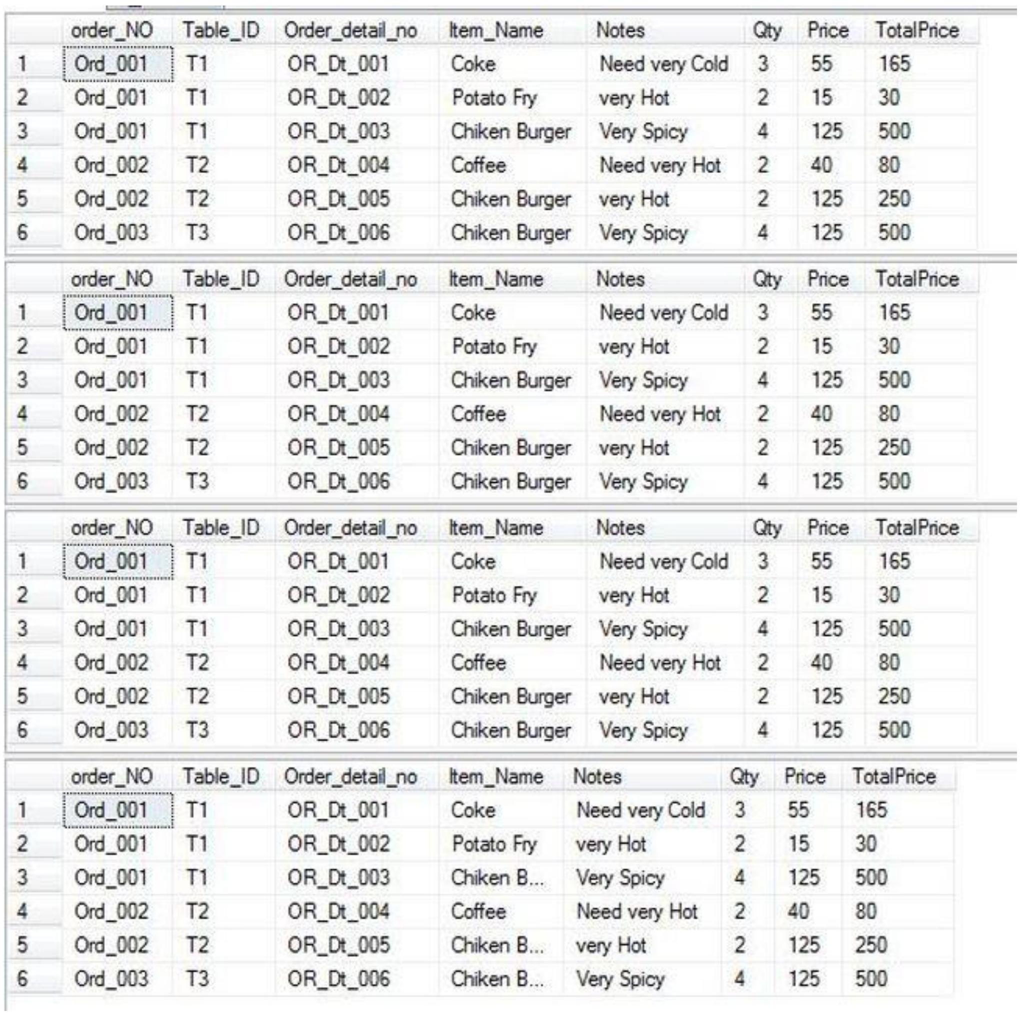
	order_NO	Table_ID	Order_detail_no	Item_code	Notes	Qty
1	Ord_001	T1	OR_Dt_001	Item001	Need very Cold	3
2	Ord_001	T1	OR_Dt_002	Item004	very Hot	2
3	Ord_001	T1	OR_Dt_003	Item003	Very Spicy	4
4	Ord_002	T2	OR_Dt_004	Item002	Need very Hot	2
5	Ord_002	T2	OR_Dt_005	Item003	very Hot	2
6	Ord_003	T3	OR_Dt_006	Item003	Very Spicy	4

	order_NO	Table_ID	Order_detail_no	Item_Name	Notes	Qty	Price	TotalPrice
1	Ord_001	T1	OR_Dt_001	Coke	Need very Cold	3	55	165
2	Ord_001	T1	OR_Dt_002	Potato Fry	very Hot	2	15	30
3	Ord_001	T1	OR_Dt_003	Chiken B...	Very Spicy	4	125	500
4	Ord_002	T2	OR_Dt_004	Coffee	Need very Hot	2	40	80
5	Ord_002	T2	OR_Dt_005	Chiken B...	very Hot	2	125	250
6	Ord_003	T3	OR_Dt_006	Chiken B	Very Spicy	4	125	500

Inner Join, Left Outer Join, Right Outer Join and Full outer Join

We can see each join example as in the following with comments.

```sql
--INNER JOIN
--
This will display the records which in both table Satisfy here i have used Like in where class which display the
SELECT M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,I.Price*D.Qty as TotalPrice
        FROM
        Ordermasters as M Inner JOIN OrderDetails as D
        ON M.Order_NO=D.Order_NO
        INNER JOIN  ItemMasters as I
        ON   D.Item_Code=I.Item_Code
        WHERE
        M.Table_ID like 'T%'
--LEFT OUTER JOIN
--This will display the records which Left side table Satisfy
 SELECT M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,I.Price*D.Qty as TotalPrice
        FROM
        Ordermasters as M LEFT OUTER JOIN OrderDetails as D
        ON M.Order_NO=D.Order_NO
        LEFT OUTER JOIN     ItemMasters as I
        ON   D.Item_Code=I.Item_Code
        WHERE
        M.Table_ID like 'T%'
--RIGHT OUTER JOIN
--This will display the records which Left side table Satisfy
 SELECT M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,I.Price*D.Qty as TotalPrice
        FROM
        Ordermasters as M RIGHT OUTER JOIN OrderDetails as D
        ON M.Order_NO=D.Order_NO
        RIGHT OUTER JOIN    ItemMasters as I
        ON   D.Item_Code=I.Item_Code
        WHERE
        M.Table_ID like 'T%'

--FULL OUTER JOIN
--This will display the records which Left side table Satisfy
 SELECT M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,I.Pri
ce*D.Qty as TotalPrice
        FROM
        Ordermasters as M FULL OUTER JOIN OrderDetails as D
        ON M.Order_NO=D.Order_NO
        FULL OUTER JOIN     ItemMasters as I
        ON   D.Item_Code=I.Item_Code
        WHERE
        M.Table_ID like 'T%'
```



	order_NO	Table_ID	Order_detail_no	Item_Name	Notes	Qty	Price	TotalPrice
1	Ord_001	T1	OR_Dt_001	Coke	Need very Cold	3	55	165
2	Ord_001	T1	OR_Dt_002	Potato Fry	very Hot	2	15	30
3	Ord_001	T1	OR_Dt_003	Chiken Burger	Very Spicy	4	125	500
4	Ord_002	T2	OR_Dt_004	Coffee	Need very Hot	2	40	80
5	Ord_002	T2	OR_Dt_005	Chiken Burger	very Hot	2	125	250
6	Ord_003	T3	OR_Dt_006	Chiken Burger	Very Spicy	4	125	500

	order_NO	Table_ID	Order_detail_no	Item_Name	Notes	Qty	Price	TotalPrice
1	Ord_001	T1	OR_Dt_001	Coke	Need very Cold	3	55	165
2	Ord_001	T1	OR_Dt_002	Potato Fry	very Hot	2	15	30
3	Ord_001	T1	OR_Dt_003	Chiken Burger	Very Spicy	4	125	500
4	Ord_002	T2	OR_Dt_004	Coffee	Need very Hot	2	40	80
5	Ord_002	T2	OR_Dt_005	Chiken Burger	very Hot	2	125	250
6	Ord_003	T3	OR_Dt_006	Chiken Burger	Very Spicy	4	125	500

	order_NO	Table_ID	Order_detail_no	Item_Name	Notes	Qty	Price	TotalPrice
1	Ord_001	T1	OR_Dt_001	Coke	Need very Cold	3	55	165
2	Ord_001	T1	OR_Dt_002	Potato Fry	very Hot	2	15	30
3	Ord_001	T1	OR_Dt_003	Chiken Burger	Very Spicy	4	125	500
4	Ord_002	T2	OR_Dt_004	Coffee	Need very Hot	2	40	80
5	Ord_002	T2	OR_Dt_005	Chiken Burger	very Hot	2	125	250
6	Ord_003	T3	OR_Dt_006	Chiken Burger	Very Spicy	4	125	500

	order_NO	Table_ID	Order_detail_no	Item_Name	Notes	Qty	Price	TotalPrice
1	Ord_001	T1	OR_Dt_001	Coke	Need very Cold	3	55	165
2	Ord_001	T1	OR_Dt_002	Potato Fry	very Hot	2	15	30
3	Ord_001	T1	OR_Dt_003	Chiken B...	Very Spicy	4	125	500
4	Ord_002	T2	OR_Dt_004	Coffee	Need very Hot	2	40	80
5	Ord_002	T2	OR_Dt_005	Chiken B...	very Hot	2	125	250
6	Ord_003	T3	OR_Dt_006	Chiken B...	Very Spicy	4	125	500

UNION and UNION ALL

Before we see joins in SQL. In a join we join 2 or more tables and display all the common related result. If we need to display 2 or more tables to be combined and return all the data then we use a UNION or UNION ALL. Here we can see UNION and UNION ALL. So what do you think about the difference between these two?

A **Union** will display only the unique results but a **Union ALL** will display the data with duplicate results also.

If we only need the distinct results of combining tables then we can use a **UNION**. If we need the results with duplicate data then we use the **UNION ALL**. That means it will display all the data.

Note: Both or All Select Column count, Name and data type should be the same to use a Union in SQL.

The syntax for using a union is like:

```sql
Select column1,Colum2 from Table1
Union
Select Column1,Column2 from Table2

Select column1,Colum2 from Table1
Union All
Select Column1,Column2 from Table2
```

Example

```sql
--Select with different where condition which display the result as 2 Table result
select Item_Code,Item_Name,Price,Description FROM ItemMasters where price <=44
select Item_Code,Item_Name,Price,Description FROM ItemMasters where price >44

--
 Union with same table but with different where condition now which result as one table
 which combine both the result.
select Item_Code,Item_Name,Price,Description FROM ItemMasters where price <=44
UNION
select Item_Code,Item_Name,Price,Description FROM ItemMasters where price >44

-- Union ALL with Join sample
 SELECT M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,I.Pri
ce*D.Qty as TotalPrice
    FROM
        Ordermasters as M (NOLOCK)   Inner JOIN OrderDetails as D
        ON M.Order_NO=D.Order_NO INNER JOIN     ItemMasters as I
        ON   D.Item_Code=I.Item_Code WHERE      I.Price <=44
```

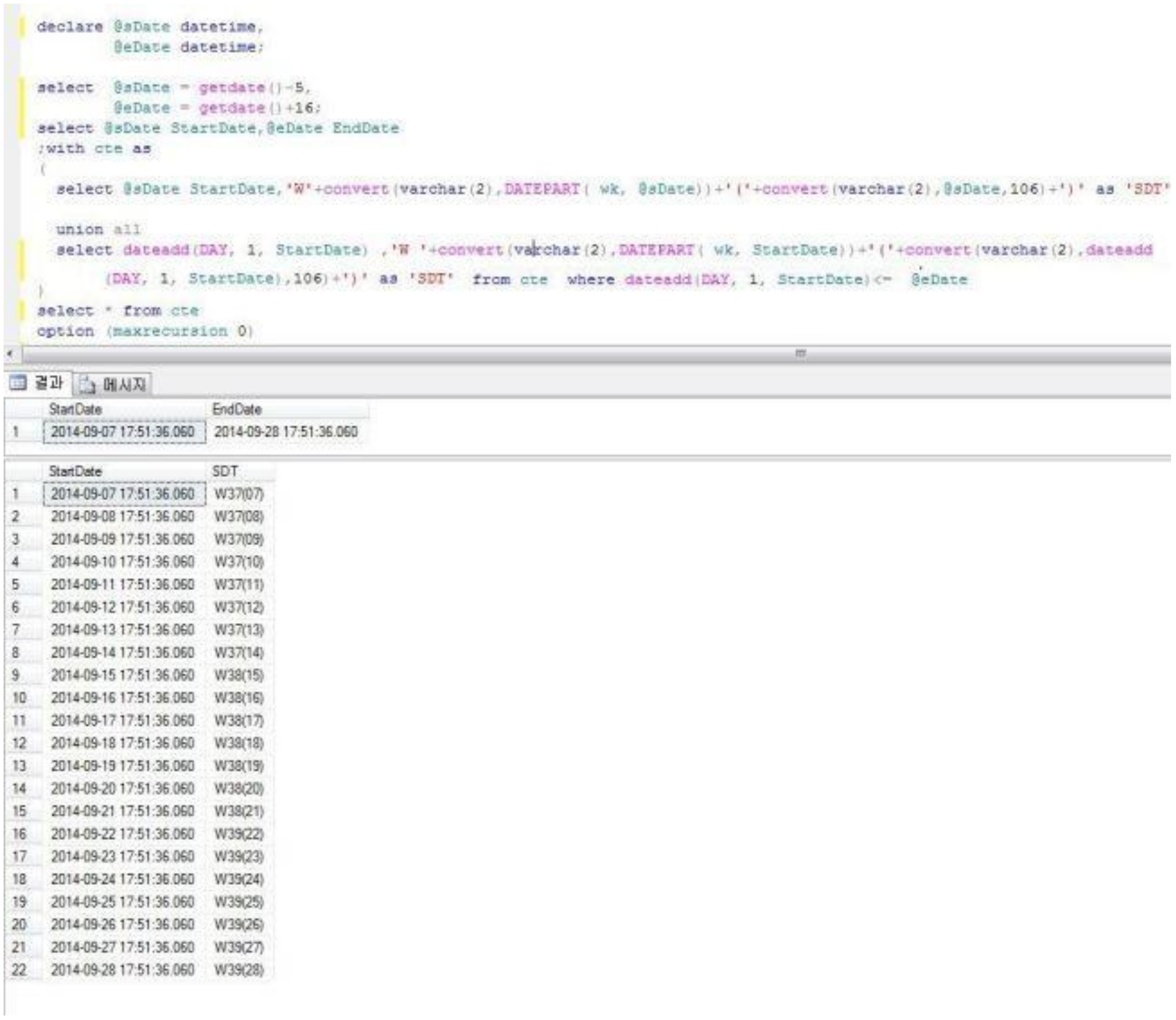
```
Union ALL
 SELECT M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,I.Price*D.Qty as TotalPrice
    FROM
        Ordermasters as M (NOLOCK)   Inner JOIN OrderDetails as D
        ON M.Order_NO=D.Order_NO    INNER JOIN  ItemMasters as I
        ON   D.Item_Code=I.Item_Code    WHERE   I.Price>44
```

Common Table Expressions (CTE) – With

A Common Table Expression (CTE) is a temporary named result set that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement.

We need to use the With clause in CTE. CTE will be very useful for displaying the records in some certain range. Here now for a simple example I have 2 dates with certain ranges, for example I have a Start_Date and an End_Date. Both have a 20 days difference. Now if I want to display all the dates with weekNo and WeekStart Day then as a result here we can see this query created using CTE.

```sql
declare @sDate datetime,
        @eDate datetime;

select  @sDate = getdate()-5,
        @eDate = getdate()+16;
--select @sDate StartDate,@eDate EndDate
;with cte as
   (
      select @sDate StartDate,'W'+convert(varchar(2),
            DATEPART( wk, @sDate))+'('+convert(varchar(2),@sDate,106)+')' as 'SDT'

  union all
       select  dateadd(DAY, 1, StartDate) ,
              'W'+convert(varchar(2),DATEPART( wk, StartDate))+'('+convert(varchar(2),

               dateadd(DAY, 1, StartDate),106)+')' as 'SDT'
  FROM  cte
  WHERE dateadd(DAY, 1, StartDate)<=  @eDate
    )
select * from cte
option (maxrecursion 0)
```



```
declare @sDate datetime,
        @eDate datetime;

select  @sDate = getdate()-5,
        @eDate = getdate()+16;
select @sDate StartDate,@eDate EndDate
;with cte as
(
  select @sDate StartDate,'W'+convert(varchar(2),DATEPART( wk, @sDate))+'('+convert(varchar(2),@sDate,106)+')' as 'SDT'

  union all
  select dateadd(DAY, 1, StartDate) ,'W '+convert(varchar(2),DATEPART( wk, StartDate))+'('+convert(varchar(2),dateadd
       (DAY, 1, StartDate),106)+')' as 'SDT'  from cte  where dateadd(DAY, 1, StartDate)<=  @eDate
)
select * from cte
option (maxrecursion 0)
```

결과 메시지

```
    StartDate                EndDate
1   2014-09-07 17:51:36.060  2014-09-28 17:51:36.060
```

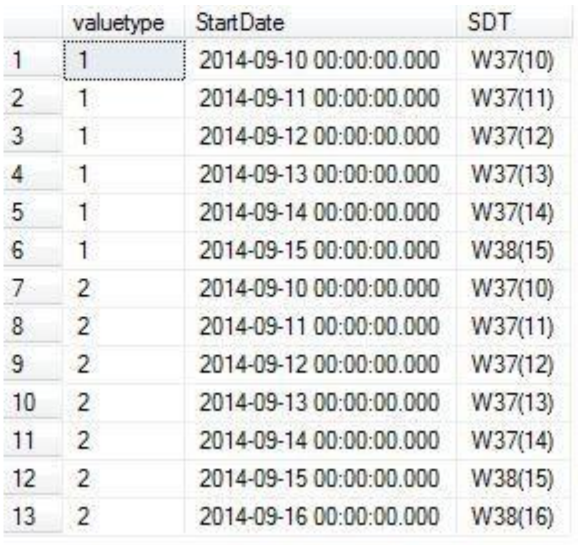
```
    StartDate                SDT
1   2014-09-07 17:51:36.060  W37(07)
2   2014-09-08 17:51:36.060  W37(08)
3   2014-09-09 17:51:36.060  W37(09)
4   2014-09-10 17:51:36.060  W37(10)
5   2014-09-11 17:51:36.060  W37(11)
6   2014-09-12 17:51:36.060  W37(12)
7   2014-09-13 17:51:36.060  W37(13)
8   2014-09-14 17:51:36.060  W37(14)
9   2014-09-15 17:51:36.060  W38(15)
10  2014-09-16 17:51:36.060  W38(16)
11  2014-09-17 17:51:36.060  W38(17)
12  2014-09-18 17:51:36.060  W38(18)
13  2014-09-19 17:51:36.060  W38(19)
14  2014-09-20 17:51:36.060  W38(20)
15  2014-09-21 17:51:36.060  W38(21)
16  2014-09-22 17:51:36.060  W39(22)
17  2014-09-23 17:51:36.060  W39(23)
18  2014-09-24 17:51:36.060  W39(24)
19  2014-09-25 17:51:36.060  W39(25)
20  2014-09-26 17:51:36.060  W39(26)
21  2014-09-27 17:51:36.060  W39(27)
22  2014-09-28 17:51:36.060  W39(28)
```

In the following Multiple CTE sample I have used more than one CTE. Now let's see how to use the Multiple CTE. Here you can see in the following sample as in the preceding that will display all the days but with one more CTE added.

In this example I have used a **UNION ALL**.

```
declare @sDate datetime,@eDate datetime;
        declare @sDate1 datetime,@eDate1 datetime;
select  @sDate = '2014-09-10', @eDate = '2014-09-15';
select  @sDate1 = '2014-09-10', @eDate1 = '2014-09-18';
WITH    cte
          AS (
              select '1' valuetype ,@sDate StartDate,'W'+convert(varchar(2),DATEPART( w
k, @sDate))+'('+convert(varchar(2),@sDate,106)+')' as 'SDT'
```

```

  union all
  select valuetype, dateadd(DAY, 1, StartDate) ,'W'+convert(varchar(2),DATEPART( wk, StartDate))+'('+convert(varchar(2),dateadd(DAY, 1, StartDate),106)+')' as 'SDT'

  from cte
  where dateadd(DAY, 1, StartDate)<=  @eDate
              ),
              cte2
          AS (
              select '2' valuetype, @sDate StartDate,'W'+convert(varchar(2),DATEPART( wk, @sDate1))+'('+convert(varchar(2),@sDate1,106)+')' as 'SDT'

  union all
  select '2' valuetype, dateadd(DAY, 1, StartDate) ,'W'+convert(varchar(2),DATEPART( wk, StartDate))+'('+convert(varchar(2),dateadd(DAY, 1, StartDate),106)+')' as 'SDT'

  from cte
  where dateadd(DAY, 1, StartDate)<=  @eDate1
              )
    SELECT  *    FROM    cte
    union all
      SELECT  *    FROM    cte2
    option (maxrecursion 0)
```



	valuetype	StartDate	SDT
1	1	2014-09-10 00:00:00.000	W37(10)
2	1	2014-09-11 00:00:00.000	W37(11)
3	1	2014-09-12 00:00:00.000	W37(12)
4	1	2014-09-13 00:00:00.000	W37(13)
5	1	2014-09-14 00:00:00.000	W37(14)
6	1	2014-09-15 00:00:00.000	W38(15)
7	2	2014-09-10 00:00:00.000	W37(10)
8	2	2014-09-11 00:00:00.000	W37(11)
9	2	2014-09-12 00:00:00.000	W37(12)
10	2	2014-09-13 00:00:00.000	W37(13)
11	2	2014-09-14 00:00:00.000	W37(14)
12	2	2014-09-15 00:00:00.000	W38(15)
13	2	2014-09-16 00:00:00.000	W38(16)

View

Many people will be confused whether a view is the same as a select. In a view we use the same select query but we need a view. So what is the use of a view?

I will say a view will be very useful in actual projects.

For example we have a long select query with more than 20 columns and more than 4 table joins. We write one big query and for that suppose we need to use that query in other places we need to call that query. Let's see the uses of views in SQL.

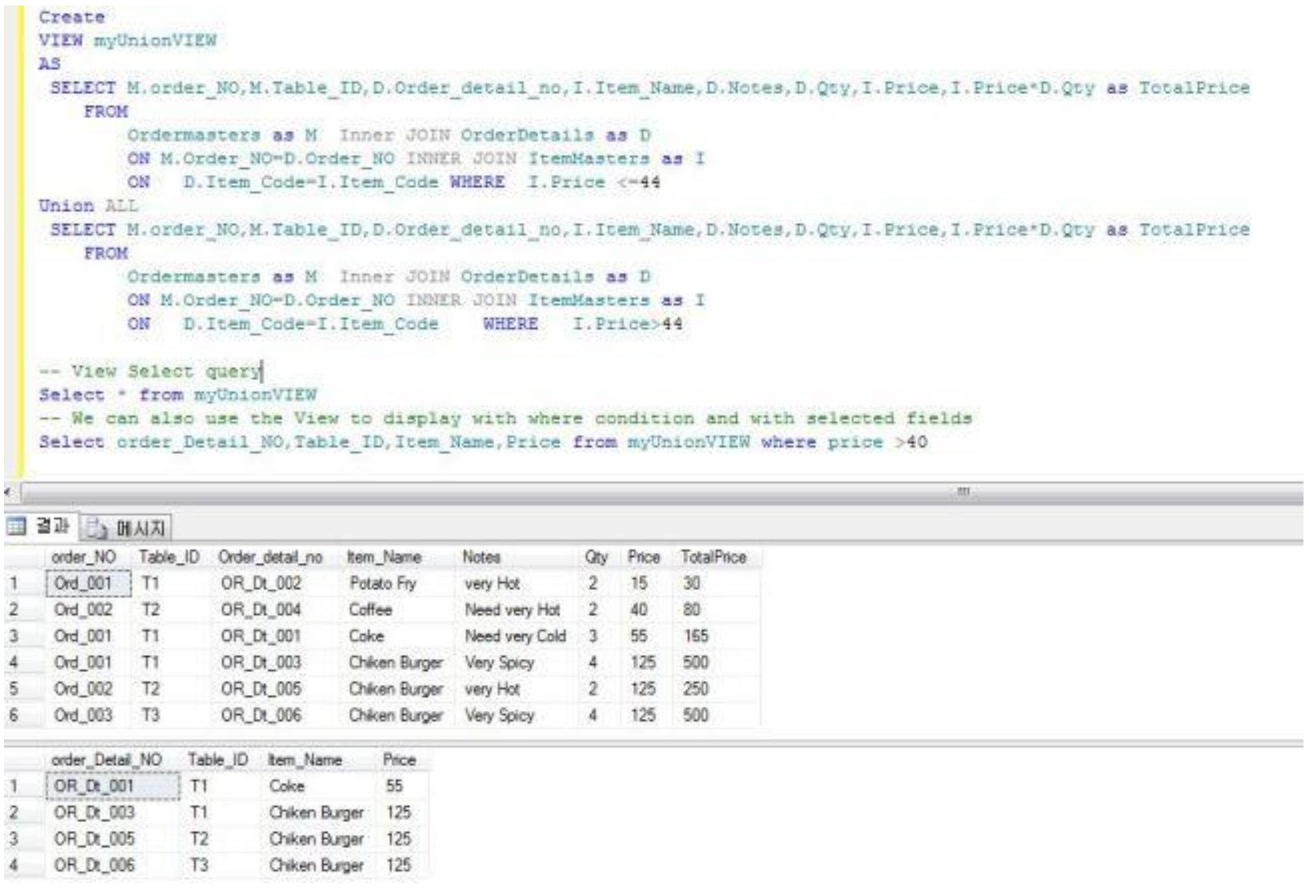
1. **Speed:** Views can be used to increase the performance.

2. **Security:** If suppose we have an Order Detail Table and it has a total of items sold, a Total price and so on. Since those fields can be viewed only by the administrator or manager and for the user they just need to only see the itemname and ItemPrice. Then in that case we can create a view for the user to display only the itemName and Price.

3. If we need to join more than one table and need to display that in several places then in that case we can use a View.

```
CREATE
VIEW viewname
AS
Select ColumNames from yourTable

Example :
-- Here we create view for our Union ALL example
Create
VIEW myUnionVIEW
AS
 SELECT M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,
        I.Price*D.Qty as TotalPrice
        FROM
          Ordermasters as M  Inner JOIN OrderDetails as D
          ON M.Order_NO=D.Order_NO INNER JOIN ItemMasters as I
          ON   D.Item_Code=I.Item_Code WHERE    I.Price <=44
Union ALL
 SELECT M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,
        I.Price*D.Qty as TotalPrice
        FROM
          Ordermasters as M  Inner JOIN OrderDetails as D
          ON M.Order_NO=D.Order_NO INNER JOIN ItemMasters as I
          ON   D.Item_Code=I.Item_Code  WHERE   I.Price>44

-- View Select query
Select * from myUnionVIEW
```

```sql
-- We can also use the View to display with where condition and with selected fields
Select order_Detail_NO,Table_ID,Item_Name,Price from myUnionVIEW where price >40
```



```sql
Create
VIEW myUnionVIEW
AS
 SELECT M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,I.Price*D.Qty as TotalPrice
    FROM
        Ordermasters as M  Inner JOIN OrderDetails as D
        ON M.Order_NO=D.Order_NO INNER JOIN ItemMasters as I
        ON   D.Item_Code=I.Item_Code WHERE  I.Price <=44
Union ALL
 SELECT M.order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,I.Price*D.Qty as TotalPrice
    FROM
        Ordermasters as M  Inner JOIN OrderDetails as D
        ON M.Order_NO=D.Order_NO INNER JOIN ItemMasters as I
        ON   D.Item_Code=I.Item_Code    WHERE   I.Price>44

-- View Select query
Select * from myUnionVIEW
-- We can also use the View to display with where condition and with selected fields
Select order_Detail_NO,Table_ID,Item_Name,Price from myUnionVIEW where price >40
```

	order_NO	Table_ID	Order_detail_no	Item_Name	Notes	Qty	Price	TotalPrice
1	Ord_001	T1	OR_Dt_002	Potato Fry	very Hot	2	15	30
2	Ord_002	T2	OR_Dt_004	Coffee	Need very Hot	2	40	80
3	Ord_001	T1	OR_Dt_001	Coke	Need very Cold	3	55	165
4	Ord_001	T1	OR_Dt_003	Chiken Burger	Very Spicy	4	125	500
5	Ord_002	T2	OR_Dt_005	Chiken Burger	very Hot	2	125	250
6	Ord_003	T3	OR_Dt_006	Chiken Burger	Very Spicy	4	125	500

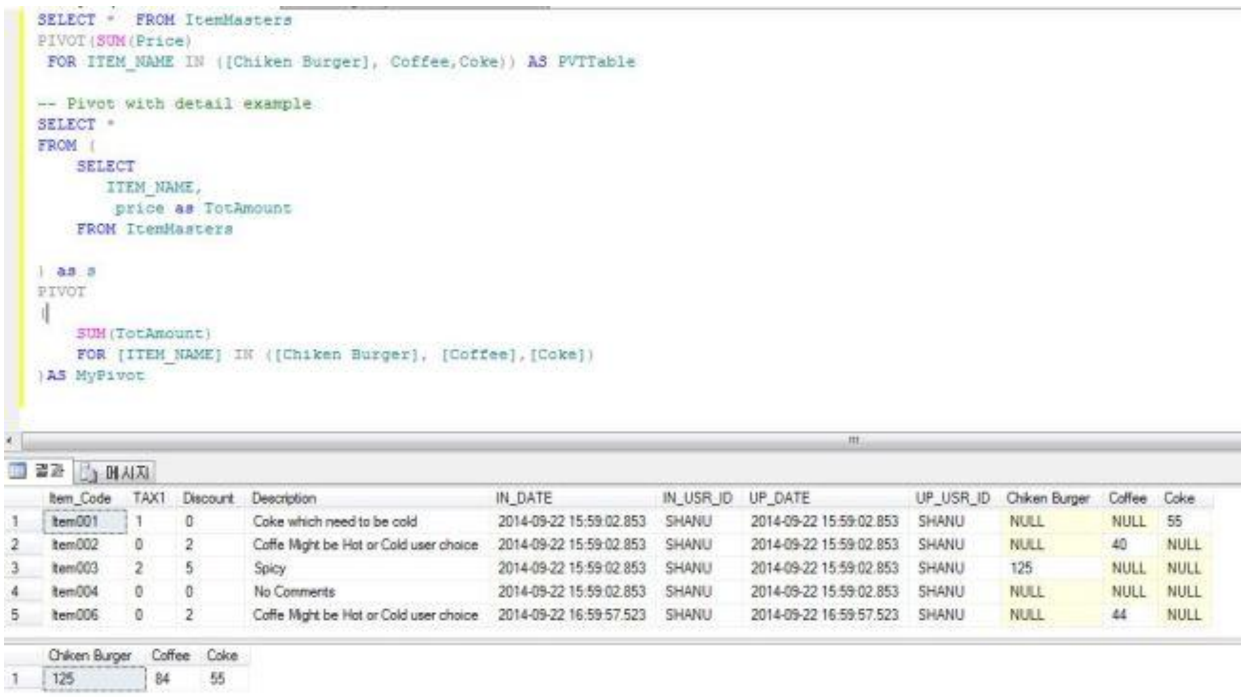
	order_Detail_NO	Table_ID	Item_Name	Price
1	OR_Dt_001	T1	Coke	55
2	OR_Dt_003	T1	Chiken Burger	125
3	OR_Dt_005	T2	Chiken Burger	125
4	OR_Dt_006	T3	Chiken Burger	125

Pivot

Why do we use a pivot in our SQL? So many people ask how to display the data from a row to a column. Yes we can do that using a Pivot. Using a Pivot we can display the rows as columns and display the average or sum of items of that. For example I want to display all the items with the price as a column and for that we can use a pivot. Now let's see an sample.

Here we can see an example that displays all the records of tblItemMaster with ItemName as a column and display the price in each row of items.

```sql
-- Simple Pivot Example
SELECT *  FROM ItemMasters
PIVOT(SUM(Price)
 FOR ITEM_NAME IN ([Chiken Burger], Coffee,Coke)) AS PVTTable

-- Pivot with detail example
SELECT *
FROM (
    SELECT
       ITEM_NAME,
        price as TotAmount
    FROM ItemMasters

) as s
PIVOT
(
    SUM(TotAmount)
    FOR [ITEM_NAME] IN ([Chiken Burger], [Coffee],[Coke])
)AS MyPivot
```



```
SELECT *  FROM ItemMasters
PIVOT(SUM(Price)
 FOR ITEM_NAME IN ([Chiken Burger], Coffee,Coke)) AS PVTTable

-- Pivot with detail example
SELECT *
FROM (
    SELECT
       ITEM_NAME,
        price as TotAmount
    FROM ItemMasters

) as s
PIVOT
(
    SUM(TotAmount)
    FOR [ITEM_NAME] IN ([Chiken Burger], [Coffee],[Coke])
)AS MyPivot
```

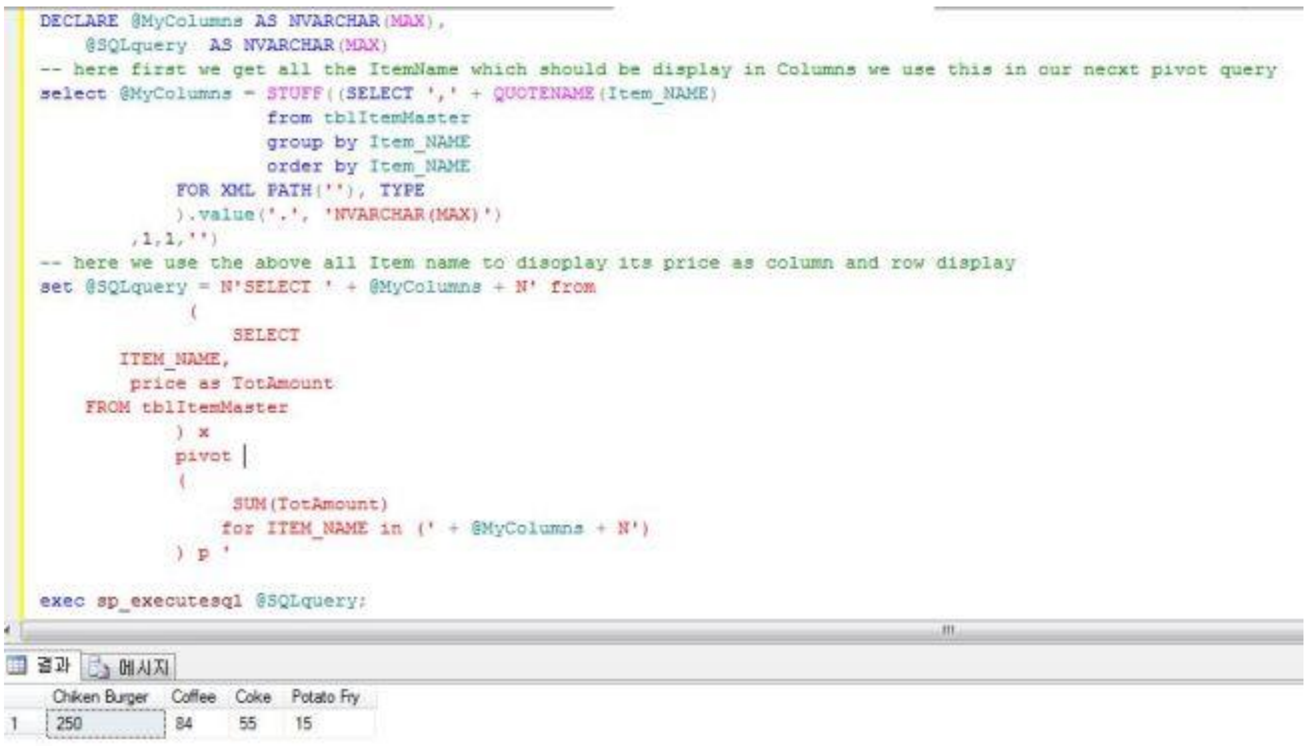
	Item_Code	TAX1	Discount	Description	IN_DATE	IN_USR_ID	UP_DATE	UP_USR_ID	Chiken Burger	Coffee	Coke
1	Item001	1	0	Coke which need to be cold	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU	NULL	NULL	55
2	Item002	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU	NULL	40	NULL
3	Item003	2	5	Spicy	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU	125	NULL	NULL
4	Item004	0	0	No Comments	2014-09-22 15:59:02.853	SHANU	2014-09-22 15:59:02.853	SHANU	NULL	NULL	NULL
5	Item006	0	2	Coffe Might be Hot or Cold user choice	2014-09-22 16:59:57.523	SHANU	2014-09-22 16:59:57.523	SHANU	NULL	44	NULL

	Chiken Burger	Coffee	Coke
1	125	84	55

Here we can see one more important example that will display all the Item Names as columns. To use this first we need to add all the item names to a variable. Then we can use this variable to display as a pivot. See the following example.

```
DECLARE @MyColumns AS NVARCHAR(MAX),
    @SQLquery  AS NVARCHAR(MAX)
-- here first we get all the ItemName which should be display in Columns we use this in our necxt pivot query
select @MyColumns = STUFF((SELECT ',' + QUOTENAME(Item_NAME)
                    FROM ItemMasters
                    GROUP BY Item_NAME
                    ORDER BY Item_NAME
            FOR XML PATH(''), TYPE
            ).value('.', 'NVARCHAR(MAX)')
        ,1,1,'')
-- here we use the above all Item name to disoplay its price as column and row display
set @SQLquery = N'SELECT ' + @MyColumns + N' from
             (
                 SELECT
       ITEM_NAME,
        price as TotAmount
    FROM ItemMasters
            ) x
            pivot
            (
                 SUM(TotAmount)
                for ITEM_NAME in (' + @MyColumns + N')
            ) p '

```

```
exec sp_executesql @SQLquery;
```



```
DECLARE @MyColumns AS NVARCHAR(MAX),
    @SQLquery  AS NVARCHAR(MAX)
-- here first we get all the ItemName which should be display in Columns we use this in our necxt pivot query
select @MyColumns = STUFF((SELECT ',' + QUOTENAME(Item_NAME)
                    from tblItemMaster
                    group by Item_NAME
                    order by Item_NAME
            FOR XML PATH(''), TYPE
            ).value('.', 'NVARCHAR(MAX)')
        ,1,1,'')
-- here we use the above all Item name to disoplay its price as column and row display
set @SQLquery = N'SELECT ' + @MyColumns + N' from
             (
                SELECT
       ITEM_NAME,
        price as TotAmount
    FROM tblItemMaster
            ) x
            pivot
            (
                SUM(TotAmount)
               for ITEM_NAME in (' + @MyColumns + N')
            ) p '

exec sp_executesql @SQLquery;
```

결과 | 메시지

```
   Chiken Burger  Coffee  Coke  Potato Fry
1  250            84      55    15
```

Stored Procedure (SP)

I saw many times in Code Project many people asking questions about how to write more than one query in SQL Server and use them in our C# program. For example many people ask how to run a Select, Insert and Update at the same time in SQL. Well we actually cannot run all the queries at the same time but we can use Store Procedures to execute them all one by one. A SP is nothing but a function we can write once and use many times and also perform various tasks with different parameter values.

In our example now I want to insert a new item into a table Item Master but before inserting I want to generate my itemCode. To do that for example as a simple method I will use the count of records and add one value to it and generate my item Code.

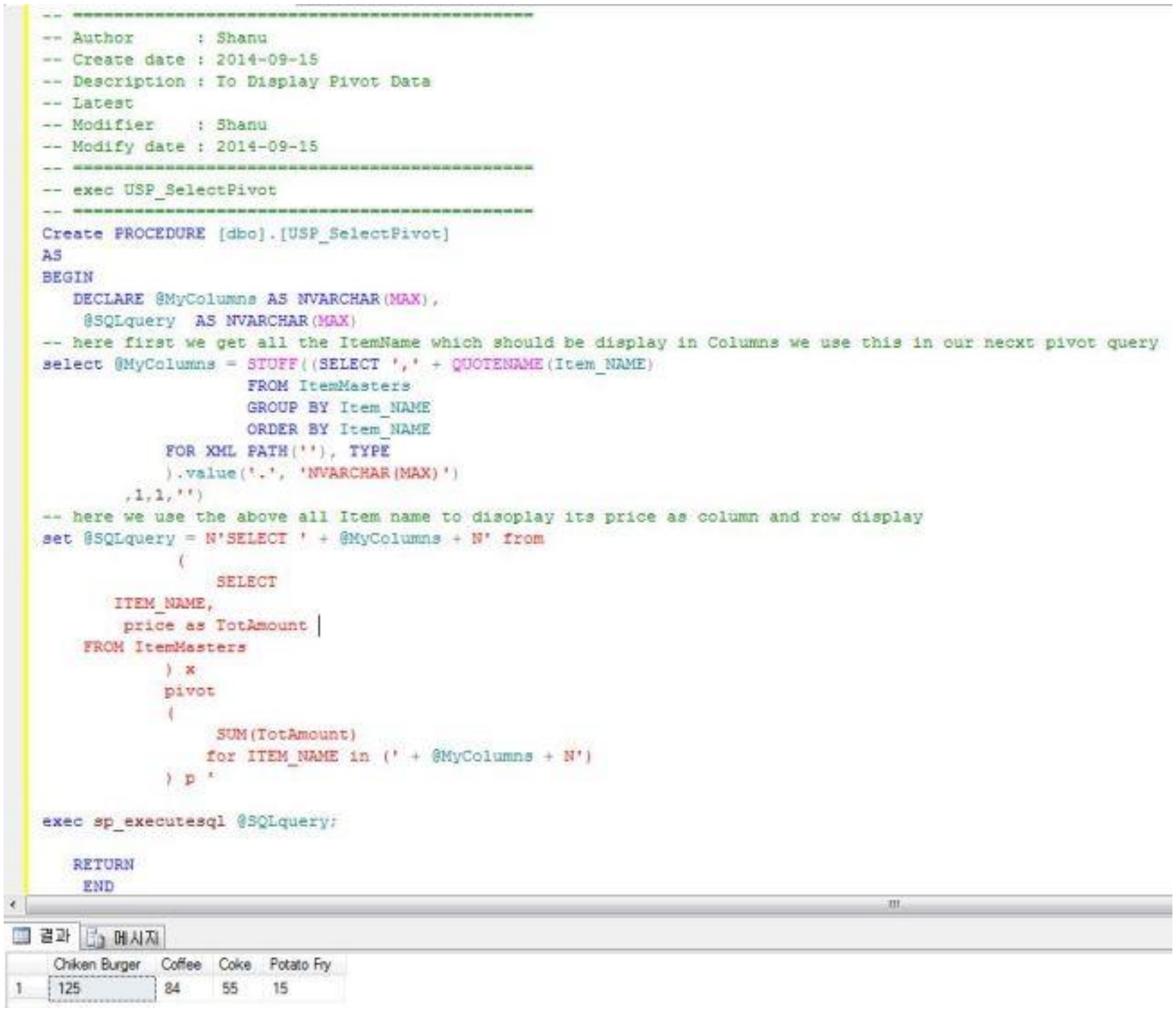
Syntax

```
CREATE PROCEDURE [ProcedureName]

AS
BEGIN
-- Select or Update or Insert query.
END

To execute SP we use
exec ProcedureName
```

Example Select Sp with no parameter

```
-- =============================================
-- Author      : Shanu
-- Create date : 2014-09-15
-- Description : To Display Pivot Data
-- Latest
-- Modifier    : Shanu
-- Modify date : 2014-09-15
-- =============================================
-- exec USP_SelectPivot
```

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

Create PROCEDURE [dbo].[USP_SelectPivot]
AS
BEGIN
   DECLARE @MyColumns AS NVARCHAR(MAX),
    @SQLquery  AS NVARCHAR(MAX)
--
 here first we get all the ItemName which should be display in Columns we use this in o
ur necxt pivot query
select @MyColumns = STUFF((SELECT ',' + QUOTENAME(Item_NAME)
                    FROM ItemMasters
                    GROUP BY Item_NAME
                    ORDER BY Item_NAME
            FOR XML PATH(''), TYPE
            ).value('.', 'NVARCHAR(MAX)')
        ,1,1,'')
--
 here we use the above all Item name to disoplay its price as column and row display
set @SQLquery = N'SELECT ' + @MyColumns + N' from
              (
                 SELECT
       ITEM_NAME,
        price as TotAmount
    FROM ItemMasters
            ) x
            pivot
            (
                 SUM(TotAmount)
                for ITEM_NAME in (' + @MyColumns + N')
            ) p '

exec sp_executesql @SQLquery;

   RETURN
    END
```



```sql
-- ==============================================
-- Author      : Shanu
-- Create date : 2014-09-15
-- Description : To Display Pivot Data
-- Latest
-- Modifier    : Shanu
-- Modify date : 2014-09-15
-- ==============================================
-- exec USP_SelectPivot
-- ==============================================
Create PROCEDURE [dbo].[USP_SelectPivot]
AS
BEGIN
   DECLARE @MyColumns AS NVARCHAR(MAX),
    @SQLquery  AS NVARCHAR(MAX)
-- here first we get all the ItemName which should be display in Columns we use this in our necxt pivot query
select @MyColumns = STUFF((SELECT ',' + QUOTENAME(Item_NAME)
                    FROM ItemMasters
                    GROUP BY Item_NAME
                    ORDER BY Item_NAME
            FOR XML PATH(''), TYPE
            ).value('.', 'NVARCHAR(MAX)')
        ,1,1,'')
-- here we use the above all Item name to disoplay its price as column and row display
set @SQLquery = N'SELECT ' + @MyColumns + N' from
             (
                 SELECT
       ITEM_NAME,
        price as TotAmount
    FROM ItemMasters
            ) x
            pivot
            (
                 SUM(TotAmount)
                for ITEM_NAME in (' + @MyColumns + N')
            ) p '

exec sp_executesql @SQLquery;

   RETURN
    END
```

결과 | 메시지

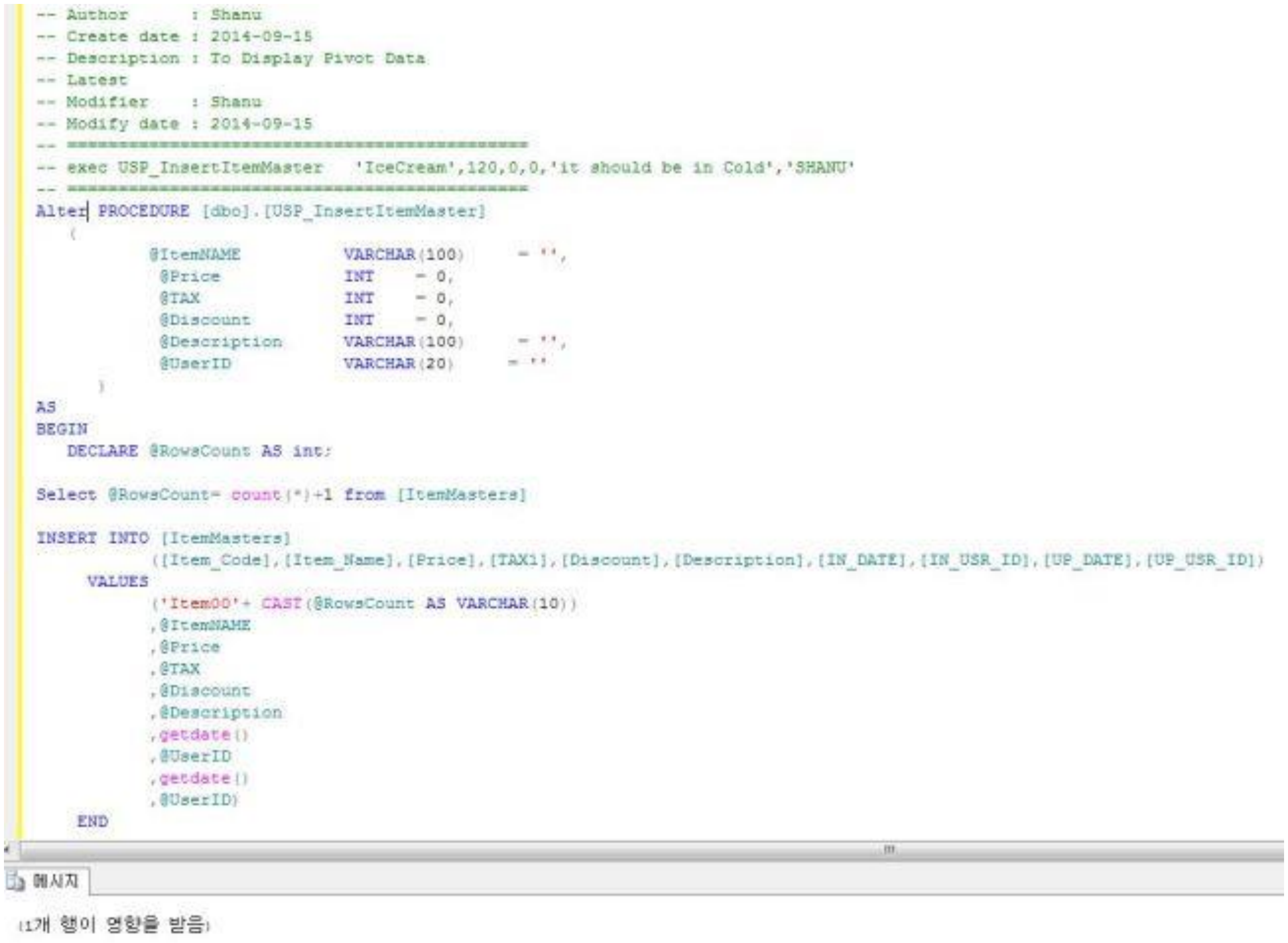
	Chiken Burger	Coffee	Coke	Potato Fry
1	125	84	55	15

Note to alter the SP we use an Alter procedure procedureName.

Select and insert SP

```
--
 Author       : Shanu
-- Create date : 2014-09-
15
--
 Description : To Display Pivot Data

-- Latest
--
 Modifier     : Shanu
```

```
-- Modify date : 2014-09-15
-- =============================================
-- exec USP_InsertItemMaster   'IceCream',120,0,0,'it should be in Cold','SHANU'
-- =============================================
Create PROCEDURE [dbo].[USP_InsertItemMaster]
   (
           @ItemNAME          VARCHAR(100)     = '',
            @Price            INT     = 0,
            @TAX              INT     = 0,
            @Discount         INT     = 0,
            @Description      VARCHAR(100)     = '',
            @UserID           VARCHAR(20)     = ''
      )
AS
BEGIN
   DECLARE @RowsCount AS int;

Select @RowsCount= count(*)+1 from [ItemMasters]


INSERT INTO [ItemMasters]
           ([Item_Code],[Item_Name],[Price],[TAX1],[Discount],[Description],[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
           ('Item00'+ CAST(@RowsCount AS VARCHAR(10))
           ,@ItemNAME
           ,@Price
           ,@TAX
           ,@Discount
           ,@Description
           ,getdate()
           ,@UserID
           ,getdate()
           ,@UserID)
    END
```



```
-- Author       : Shanu
-- Create date : 2014-09-15
-- Description : To Display Pivot Data
-- Latest
-- Modifier     : Shanu
-- Modify date : 2014-09-15
-- =============================================
-- exec USP_InsertItemMaster   'IceCream',120,0,0,'it should be in Cold','SHANU'
-- =============================================
Alter PROCEDURE [dbo].[USP_InsertItemMaster]
   (
           @ItemNAME           VARCHAR(100)     = '',
            @Price             INT     = 0,
            @TAX               INT     = 0,
            @Discount          INT     = 0,
            @Description       VARCHAR(100)     = '',
            @UserID            VARCHAR(20)     = ''
      )
AS
BEGIN
   DECLARE @RowsCount AS int;

Select @RowsCount= count(*)+1 from [ItemMasters]

INSERT INTO [ItemMasters]
           ([Item_Code],[Item_Name],[Price],[TAX1],[Discount],[Description],[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
     VALUES
           ('Item00'+ CAST(@RowsCount AS VARCHAR(10))
           ,@ItemNAME
           ,@Price
           ,@TAX
           ,@Discount
           ,@Description
           ,getdate()
           ,@UserID
           ,getdate()
           ,@UserID)
    END
```

메시지

```
(1개 행이 영향을 받음)
```

Functions

In this article we have already seen a few pre-defined system functions like MAX(), SUM(), GetDate() and and so on. Now let's see how to create a user defined function.

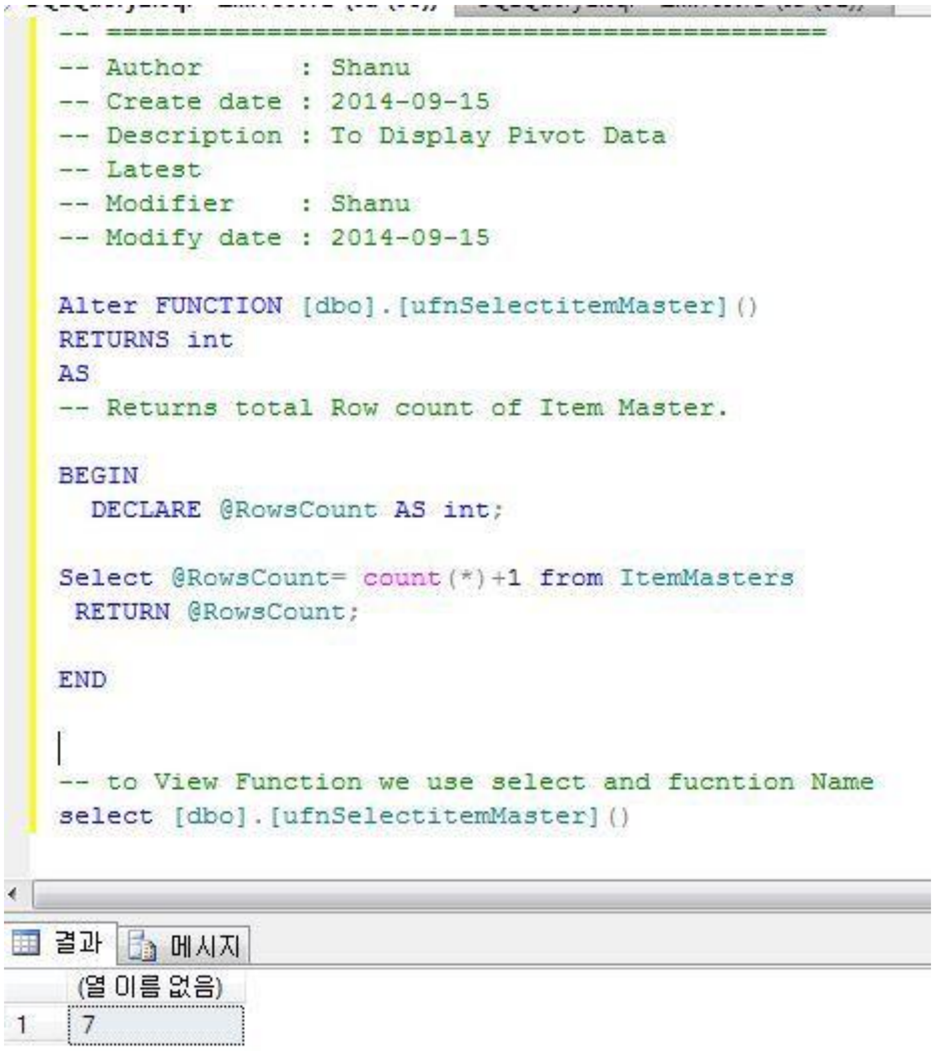
If someone were to ask me what a function is and what is the use of a function then in a simple way I will say that if I want to execute or perform some action several times with a different meaning then I will create a function and call it whenever I need it. The following is the syntax to create a function.

```
<a style="font-size: 14px; color: #111111">

Create Function functionName
As
Begin
END

</a>
```

Here we will see a simple function that will return the max row count of tblItemMaster.

```
<a style="font-size: 14px; color: #111111">
-- =============================================
-- Author       : Shanu
-- Create date : 2014-09-15
-- Description : To Display Pivot Data
-- Latest
-- Modifier     : Shanu
-- Modify date : 2014-09-15

Alter FUNCTION [dbo].[ufnSelectitemMaster]()
RETURNS int
AS
-- Returns total Row count of Item Master.

BEGIN
  DECLARE @RowsCount AS int;

Select @RowsCount= count(*)+1 from ItemMasters
 RETURN @RowsCount;

END
```

```


-- to View Function we use select and fucntion Name
select [dbo].[ufnSelectitemMaster]()

</a>
```



```
-- =============================================
-- Author        : Shanu
-- Create date : 2014-09-15
-- Description : To Display Pivot Data
-- Latest
-- Modifier      : Shanu
-- Modify date : 2014-09-15

Alter FUNCTION [dbo].[ufnSelectitemMaster]()
RETURNS int
AS
-- Returns total Row count of Item Master.

BEGIN
  DECLARE @RowsCount AS int;

Select @RowsCount= count(*)+1 from ItemMasters
 RETURN @RowsCount;

END

-- to View Function we use select and fucntion Name
select [dbo].[ufnSelectitemMaster]()
```

결과 | 메시지

	(열 이름 없음)
1	7

Here we can see another function that will return the last date of a month from a given date.

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

--
 Author       : Shanu
-- Create date : 2014-09-
15
--
 Description : To Display Pivot Data
```

```
-- Latest
--
 Modifier    : Shanu
-- Modify date : 2014-09-15

ALTER FUNCTION [dbo].[ufn_LastDayOfMonth]
(
   @DATE NVARCHAR(10)
)
RETURNS NVARCHAR(10)
AS
BEGIN
   RETURN CONVERT(NVARCHAR(10), DATEADD(D, -
1, DATEADD(M, 1, CAST(SUBSTRING(@DATE,1,7) + '-01' AS DATETIME))), 120)
END
SELECT dbo.ufn_LastDayOfMonth('2014-09-01')AS LastDay

</a>
```

SQL Server Coding Standards

Here are a few SQL Coding Standards sites. Kindly refer to these links and follow the SQL Standards in your queries.

- SQL Server Database Coding Standards and Guidelines.
- SQL Server – Database Coding Standards and Guidelines – Part 1

Few Working Examples

Example 1: To Display Every Week Sunday.

In some case we need to display all the days that should be Sunday, Monday and so on. For example to list all the days of work that start on Sunday.

Here now let's see our following example. I have the From Date and the To Date. Now I want to display all the Sundays alone between this range.

For this we can use our preceding CTE, the same example. Now using the CTE we get all the dates between 2 dates. Now what we need to do is to select only the Sunday dates. I have used the temp table to store all the Sundays and display the result.

```
declare @FromDate datetime,
       @ToDate datetime;

IF OBJECT_ID('tempdb..#TEMP_EveryWk_Snday') IS NOT NULL

    DROP TABLE #TEMP_EveryWk_Snday


DECLARE @TOTALCount INT

         SET @FromDate = getdate();
         SET @ToDate = DATEADD(Month, 3,getdate());

Select  @TOTALCount= DATEDIFF(dd,@FromDate,@ToDate);
  WITH d AS
             (
              SELECT top (@TOTALCount) AllDays = DATEADD(DAY, ROW_NUMBER()

                OVER (ORDER BY object_id), REPLACE(@FromDate,'-
','',''))
              FROM sys.all_objects
             )
         SELECT Distinct DATEADD(DAY, 1 - DATEPART(WEEKDAY, AllDays),
                     CAST(AllDays AS DATE))WkStartSundays
            INTO #TEMP_EveryWk_Snday

            FROM d
            WHERE
            AllDays  <= @ToDate

    Select  WkStartSundays WEEKSTART_Sunday,
            DATENAME(dw,WkStartSundays) Day_Name
            FROM #TEMP_EveryWk_Snday
```