SQL Queries For Beginners

This free book is provided by courtesy of C# Corner and Mindcracker Network and its authors. Feel free to share this book with your friends and co-workers. Please do not reproduce, republish, edit or copy this book.

Syed Shanu
(C# Corner MVP)
<table>
<thead>
<tr>
<th>Index</th>
<th>Page No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create and Insert Query</td>
<td>1-4</td>
</tr>
<tr>
<td>Simple SQL Select Query</td>
<td>5-6</td>
</tr>
<tr>
<td>Aggregate and Scalar Function</td>
<td>7-8</td>
</tr>
<tr>
<td>Date Function</td>
<td>9-11</td>
</tr>
<tr>
<td>Other Select Function</td>
<td>12-14</td>
</tr>
<tr>
<td>Where Clause</td>
<td>15-18</td>
</tr>
<tr>
<td>Group BY and Having Clause</td>
<td>19-21</td>
</tr>
<tr>
<td>Sub Query</td>
<td>22-23</td>
</tr>
<tr>
<td>Joins</td>
<td>24-26</td>
</tr>
<tr>
<td>UNION and UNION ALL.</td>
<td>27-28</td>
</tr>
<tr>
<td>Common Table Expressions (CTE) – With</td>
<td>29-31</td>
</tr>
<tr>
<td>View</td>
<td>32-33</td>
</tr>
<tr>
<td>Pivot</td>
<td>34-36</td>
</tr>
<tr>
<td>Stored Procedure (SP)</td>
<td>37-41</td>
</tr>
<tr>
<td>Functions</td>
<td>42-44</td>
</tr>
<tr>
<td>SQL Server Coding Standards</td>
<td>44</td>
</tr>
<tr>
<td>Few Worked Examples</td>
<td>45</td>
</tr>
</tbody>
</table>
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.

```sql
1. CREATE TABLE [dbo].[ItemMasters](
2.   [Item_Code] [varchar](20) NOT NULL,
3.   [Item_Name] [varchar](100) NOT NULL,
4.   [Price] int NOT NULL,
5.   [TAX1] int NOT NULL,
6.   [Discount] int NOT NULL,
7.   [Description] [varchar](200) NOT NULL,
8.   [IN_DATE] [datetime] NOT NULL,
9.   [IN_USR_ID] [varchar](20) NOT NULL,
10.  [UP_DATE] [datetime] NOT NULL,
11.  [UP_USR_ID] [varchar](20) NOT NULL,
12.  CONSTRAINT [PK_ItemMasters] PRIMARY KEY CLUSTERED
13.  )
14. )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
15. )
16. )
```

Insert Item Master

```sql
1. INSERT INTO [ItemMasters] ([Item_Code],[Item_Name],[Price],[TAX1],[Discount],[Description],[IN_DATE]
```
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]
```
Insert Order Master

1. **INSERT INTO** [OrderMasters]
2. ([Order_No], [Table_ID], [Description], [IN_DATE], [IN_USR_ID], [UP_DATE], [UP_USR_ID])
3. **VALUES**
4. ('Ord_001', 'T1', '', GETDATE(), 'SHANU', GETDATE(), 'SHANU')
5. **INSERT INTO** [OrderMasters]
6. ([Order_No], [Table_ID], [Description], [IN_DATE], [IN_USR_ID], [UP_DATE], [UP_USR_ID])
7. **VALUES**
8. ('Ord_002', 'T2', '', GETDATE(), 'Mak', GETDATE(), 'MAK')
9. **INSERT INTO** [OrderMasters]
10. ([Order_No], [Table_ID], [Description], [IN_DATE], [IN_USR_ID], [UP_DATE], [UP_USR_ID])
11. **VALUES**
12. ('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.

1. **CREATE TABLE** [dbo].[OrderDetails]{
2. [Order_Detail_No] [varchar](20) NOT NULL,
3. [Order_No] [varchar](20) CONSTRAINT fk_OrderMasters FOREIGN KEY REFERENCES OrderMasters(Order_No),
4. [Item_Code] [varchar](20) CONSTRAINT fk_ItemMasters FOREIGN KEY REFERENCES ItemMasters(Item_Code),
5. [Notes] [varchar](200) NOT NULL,
6. [QTY] INT NOT NULL,
7. [IN_DATE] [datetime] NOT NULL,
8. [IN_USR_ID] [varchar](20) NOT NULL,
9. [UP_DATE] [datetime] NOT NULL,
10. [UP_USR_ID] [varchar](20) NOT NULL,
11. **CONSTRAINT** [PK_OrderDetails] PRIMARY KEY CLUSTERED
12. {
13. [Order_Detail_No] ASC
14. **WITH** (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
15. ) ON [PRIMARY]
16. 17.
18. --Now let's insert the 3 items for the above Order No 'Ord_001'.
19. **INSERT INTO** [OrderDetails]
20. ([Order_Detail_No], [Order_No], [Item_Code], [Notes], [QTY],
21. , [IN_DATE], [IN_USR_ID], [UP_DATE], [UP_USR_ID])
22. **VALUES**
Here we can see the same order_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.

```
select * from OrderDetails
```

<table>
<thead>
<tr>
<th>Order_Detail_No</th>
<th>Order_No</th>
<th>Item_Code</th>
<th>Notes</th>
<th>QTY</th>
<th>IN_DATE</th>
<th>IN_USR_ID</th>
<th>UP_DATE</th>
<th>UP_USR_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR_Dt_001</td>
<td>Ord_001</td>
<td>Item001</td>
<td>Need very Cold</td>
<td>3</td>
<td>2014-09-22 16:06:31.933</td>
<td>SHANU</td>
<td>2014-09-22 16:06:31.933</td>
<td>SHANU</td>
</tr>
<tr>
<td>OR_Dt_002</td>
<td>Ord_001</td>
<td>Item004</td>
<td>very Hot</td>
<td>2</td>
<td>2014-09-22 16:06:31.940</td>
<td>SHANU</td>
<td>2014-09-22 16:06:31.940</td>
<td>SHANU</td>
</tr>
<tr>
<td>OR_Dt_004</td>
<td>Ord_002</td>
<td>Item002</td>
<td>Need very Hot</td>
<td>2</td>
<td>2014-09-22 16:06:31.933</td>
<td>SHANU</td>
<td>2014-09-22 16:06:31.933</td>
<td>SHANU</td>
</tr>
<tr>
<td>OR_Dt_005</td>
<td>Ord_002</td>
<td>Item003</td>
<td>very Hot</td>
<td>2</td>
<td>2014-09-22 16:06:31.940</td>
<td>SHANU</td>
<td>2014-09-22 16:06:31.940</td>
<td>SHANU</td>
</tr>
</tbody>
</table>

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.
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.

```
1. SELECT 'My Name Is SYED SHANU'
2. -- With Column Name using 'AS'
3. SELECT 'My Name Is SYED SHANU' as 'MY NAME'
4. -- With more then the one Column
5. SELECT 'My Name' as 'Column1', 'Is' as 'Column2', 'SYED SHANU' as 'Column3'
```
Select Statement from Table

1. -- To Display all the columns from the table we use * operator in select Statement.
2. Select * from ItemMasters
3. --
4. If we need to select only few fields from a table we can use the Column Name in Select Statement.
5. Select Item_Code
6. , Item_name as Item
7. , Price
8. , Description
9. , In_DATE
10. FROM
11. ItemMasters
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.

```
1. select * from ItemMasters
2. -- Aggregate
3. -- COUNT() -> returns the Total no of records from table , AVG() returns the Average Value from Column,MAX() Returns Max Value from Column
4. -- ,MIN() returns Min Value from Column,SUM() sum of total from Column
5. Select Count(*) TotalRows,AVG(Price) AVGPrice
6. ,MAX(Price) MAXPrice,MIN(Price) MinPrice,Sum(price) PriceTotal
7. FROM ItemMasters
8. 
9. -- Scalar
10. -- UCASE() -> Convert to Upper Case ,LCASE() -> Convert to Lower Case,
11. -- SUBSTRING() -> Display selected char from column ,
12. --,LEN() -> length of column date,
13. -- ROUND() -> Which will round the value
14. SELECT UPPER(Item_NAME) Uppers,LOWER(Item_NAME) Lowers,
15. SUBSTRING(Item_NAME,2,3) MidValue,LEN(Item_NAME) Lents
16. ,SUBSTRING(Item_NAME,2,LEN(Item_NAME)) MidValuewithLenFunction,
17. ROUND(Price,0) as Rounded
18. FROM ItemMasters
```
```sql
SELECT * FROM ItemMasters

-- Aggregate
-- COUNT(*) -> returns the total no of records from table , AVG() returns the Average Value from Column, 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

-- Sqler
-- UCase() -> Convert to Upper Case , LCase() -> Convert to Lower Case,
-- SUBSTRING() ->Display selected char from column ->SUBSTRING(ColumnName, StartIndex, Length)CharToDisplay
-- LEN() -> length of column date,
-- ROUND() -> Which will round the value
SELECT UPPER(Item_NAME), Uppers,
      LOWER(Item_NAME) Lowers,
      SUBSTRING(Item_NAME, 1, 3) MidPrice,
      LEN(Item_NAME) Lengths,
      SUBSTRING(Item_NAME, 1, LEN(Item_NAME)) MidValueLenLenFunction,
      ROUND(Price, 0) as Rounded
FROM ItemMasters
```

<table>
<thead>
<tr>
<th>Item_Code</th>
<th>Item_Name</th>
<th>Price</th>
<th>TAX</th>
<th>Discount</th>
<th>Description</th>
<th>IN_DATE</th>
<th>IN_USR_ID</th>
<th>UP_DATE</th>
<th>UP_USR_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item001</td>
<td>Coke</td>
<td>55</td>
<td>1</td>
<td>0</td>
<td>Coke which need to be cold</td>
<td>2014-09-22 15:59:02.853</td>
<td>SHANU</td>
<td>2014-09-22 15:59:02.853</td>
<td>SHANU</td>
</tr>
<tr>
<td>Item002</td>
<td>Coffee</td>
<td>40</td>
<td>0</td>
<td>2</td>
<td>coffin might be Hot or Cold user choice</td>
<td>2014-09-22 15:59:02.853</td>
<td>SHANU</td>
<td>2014-09-22 15:59:02.853</td>
<td>SHANU</td>
</tr>
<tr>
<td>Item004</td>
<td>Potato Fry</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>No Comments</td>
<td>2014-09-22 15:59:02.853</td>
<td>SHANU</td>
<td>2014-09-22 15:59:02.853</td>
<td>SHANU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TotalRows</th>
<th>AVGPrice</th>
<th>MAXPrice</th>
<th>MinPrice</th>
<th>PriceTotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>58</td>
<td>123</td>
<td>16</td>
<td>235</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item_Name</th>
<th>Uppers</th>
<th>Lowers</th>
<th>MidPrice</th>
<th>Lengths</th>
<th>MidValueLenLenFunction</th>
<th>Rounded</th>
</tr>
</thead>
<tbody>
<tr>
<td>COKE</td>
<td>coke</td>
<td>coke</td>
<td>4</td>
<td>coke</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>COFFEE</td>
<td>coffee</td>
<td>coffee</td>
<td>6</td>
<td>coffee</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>CHICKEN BURGER</td>
<td>chicken burger</td>
<td>13</td>
<td>chicken Burger</td>
<td>125</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POTATO FRY</td>
<td>potato fry</td>
<td>10</td>
<td>potato fry</td>
<td>15</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

1. -- GETDATE() -> to Display the Current Date and Time
2. -- Format() -> used to display our date in our requested format
3. Select GETDATE() CurrentDateTime, FORMAT(GETDATE(),'yyyy-MM-dd') AS DateFormats,
4. FORMAT(GETDATE(),'HH-mm-ss')TimeFormats,
5. CONVERT(VARCHAR(10),GETDATE(),10) Converts1,
6. CONVERT(VARCHAR(24),GETDATE(),113),
7. CONVERT(NVARCHAR, getdate(), 106) Converts2 ,-- here we used Convert Function
8. REPLACE(convert(NVARCHAR, getdate(), 106), '', '/') Formats--
9. Here we used replace and --convert functions.
10. -- first we convert the date to nvarchar and then we replace the '' with ' /'
11. select * from Itemmasters
12.
13. Select ITEM_NAME, IN_DATE CurrentDateTime, FORMAT(IN_DATE,'yyyy-MM-
    dd') AS DateFormats,
14. FORMAT(IN_DATE,'HH-mm-ss')TimeFormats,
15. CONVERT(VARCHAR(10),IN_DATE,10) Converts1,
16. CONVERT(VARCHAR(24),IN_DATE,113),
17. convert(NVARCHAR, IN_DATE, 106) Converts2 ,-- here we used Convert Function
18. REPLACE(convert(NVARCHAR,IN_DATE, 106), '', '/') Formats
19. FROM Itemmasters
- **DatePart**: We use the datepart to display the selected Year, Month and Day.

- **DateADD**: We use dateadd to display or add or subtract days from the selected date.

- **DateDiff**: We use dateDiff to calculate the difference between 2 dates.

```
1. --Datepart
   DATEPART(dateparttype,yourDate)
2. SELECT DATEPART(yyyy,getdate()) AS YEARS ,
3.    DATEPART(mm,getdate()) AS MONTHS ,
4.    DATEPART(dd,getdate()) AS Days ,
5.    DATEPART(week,getdate()) AS weeks ,
6.    DATEPART(hour,getdate()) AS hours
7. --Days Add to add or subdtract date from a selected date.
8. SELECT GetDate() currentDate,DATEADD(day,12,getdate()) AS AddDays ,
9.    DATEADD(day,-4,getdate()) AS FourDaysBeforeDate
10. -- DATEDIFF() -> to display the Days between 2 dates
11. select DATEDIFF(year,'2003-08-05',getdate()) yearDifference ,
12.    DATEDIFF(day,DATEADD(day,-24,getdate()),getdate()) daysDifferent ,
13.    DATEDIFF(month,getdate(),DATEADD(Month,6,getdate())) MonthDifference
```
--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 subtract 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, '2003-09-05', getdate()) as yearDifference,
       DATEDIFF(day, DATEADD(day, -24, getdate()), getdate()) as daysDifferent,
       DATEDIFF(month, getdate(), DATEADD(Month, 6, getdate())) as MonthDifference

<table>
<thead>
<tr>
<th></th>
<th>YEARS</th>
<th>MONTHS</th>
<th>Days</th>
<th>weeks</th>
<th>hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2014</td>
<td>9</td>
<td>12</td>
<td>37</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Current Date</th>
<th>Add Days</th>
<th>FourDaysBeforeDate</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>yearDifference</th>
<th>daysDifferent</th>
<th>MonthDifference</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>24</td>
<td>6</td>
</tr>
</tbody>
</table>
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.

```sql
1. <a name="5"/>
2. --Top to Select Top first and last records using Select Statement.
3. Select * FROM ItemMasters
4. --> First Display top 2 Records
5. Select TOP 2 Item_Code
6. ,Item_name as Item
7. ,Price
8. ,Description
9. ,In_DATE
10. FROM ItemMasters
11. --> to Display the Last to Records we need to use the Order By Clause
12. -- order By to display Records in ascending or desending order by the columns
13. Select TOP 2 Item_Code
14. ,Item_name as Item
15. ,Price
16. ,Description
17. ,In_DATE
18. FROM ItemMasters
19. ORDER BY Item_Code DESC</a>
```
**Distinct:** The distinct keyword is used in select statements to avoid the duplicate records.

1. `<a name="5">`
2. `Select * FROM ItemMasters`
3. `-Distinct -> To avoid the Duplicate records we use the distinct in select statement`
4. `-` for example in this table we can see here we have the duplicate record 'Chiken Burger'
5. `- but with different Item_Code when i use the below select statement see what happen`
6. `Select Item_name as Item`
7. `,Price`
8. `,Description`
9. `,IN_USR_ID`
10. `FROM ItemMasters`
11. `-` 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.
12. `select Distinct Item_name as Item`
13. `,Price`
14. `,Description`
15. `,IN_USR_ID`
16. `FROM ItemMasters`
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 'Chicken 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.

<table>
<thead>
<tr>
<th>Item_Code</th>
<th>Item_Name</th>
<th>Price</th>
<th>TAX1</th>
<th>Discount</th>
<th>Description</th>
<th>IN_DATE</th>
<th>IN_USR_ID</th>
<th>UP_DATE</th>
<th>UP_USR_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Coke</td>
<td>55</td>
<td>1</td>
<td>0</td>
<td>Coke which need to be cold</td>
<td>2014-09-22 15:19:02.853</td>
<td>SHANU</td>
<td>2014-09-22 15:19:02.853</td>
<td>SHANU</td>
</tr>
<tr>
<td>2</td>
<td>Coffee</td>
<td>40</td>
<td>0</td>
<td>2</td>
<td>Coffee Might be Hot or Cold user choice</td>
<td>2014-09-22 15:19:02.853</td>
<td>SHANU</td>
<td>2014-09-22 15:19:02.853</td>
<td>SHANU</td>
</tr>
<tr>
<td>4</td>
<td>Potato Fry</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>No Comments</td>
<td>2014-09-22 15:19:02.853</td>
<td>SHANU</td>
<td>2014-09-22 15:19:02.853</td>
<td>SHANU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
<th>Description</th>
<th>IN_USR_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Coke</td>
<td>Coke which need to be cold</td>
<td>SHANU</td>
</tr>
<tr>
<td>2</td>
<td>Coffee</td>
<td>Coffee Might be Hot or Cold user choice</td>
<td>SHANU</td>
</tr>
<tr>
<td>3</td>
<td>Chicken Burger</td>
<td>Spicy</td>
<td>SHANU</td>
</tr>
<tr>
<td>4</td>
<td>Potato Fry</td>
<td>No Comments</td>
<td>SHANU</td>
</tr>
</tbody>
</table>

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REUBLISH, CHANGE OR COPY.
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.

1. Select * from ItemMasters
2. Select * from OrderDetails
3. -- Where -> To display the data with certain conditions
4. -- Now below example which will display all the records which has Item_Name='Coke'
5. select * FROM ItemMasters WHERE ITEM_NAME='COKE'
6. --
   If we want display all the records Item_Name which Starts with 'C' then we use Like in where clause.
7. SELECT * FROM ItemMasters WHERE ITEM_NAME Like 'C%'
8. --
   > here we display the ItemMasters where the price will be greater then or equal to 40.
9. 10. --> to use more than one condition we can Use And or Or operator.
11. --
   If we want to check the data between to date range then we can use Between Operator in Where Clause.
12. select Item_name as Item
13.   ,Price
14.   ,Description
15.   ,IN_USR_ID
16. FROM ItemMasters
17. WHERE
18.   ITEM_NAME Like 'C%'
19.   AND
20.   price >=40
21. --> here we display the OrderDetails where the Qty will be greater 3
22. 23. Select * FROM OrderDetails WHERE qty>3
Where In Clause

1. -- In clause -> used to display the data which is in the condition
   2. select *
   3. FROM ItemMasters
   4. WHERE
   5. Item_name IN ('Coffee','Chiken Burger')

6. -- In clause with Order By - Here we display the in descending order.
7. select *
   8. FROM ItemMasters
   9. WHERE
   10. Item_name IN ('Coffee','Chiken Burger')
   11. ORDER BY Item_Code Desc
Where Between keyword

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

2. select * FROM ItemMasters

3. 

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

7. 

8. select * FROM ItemMasters
   WHERE
   ITEM_NAME Like 'C%'
11. AND
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.

```
1. --Group By -
   > To display the data with group result. Here we can see we display all the Aggregate result by Item Name
2. Select ITEM_NAME,Count(*) TotalRows,AVG(Price) AVGPrice
3.   ,MAX(Price) MAXPrice,MIN(Price) MinPrice,Sum(price) PriceTotal
4. FROM ItemMasters
5.   GROUP BY ITEM_NAME
6.
7.
8. --
   Here this group by will combine all the same Order_No result and make the total or each order_NO
9. Select Order_NO,Sum(QTy) as TotalQTY
10. FROM OrderDetails
11. where qty>=2
12. GROUP BY Order_NO
13.
14. -- Here the Total will be created by order_No and Item_Code
15. Select Order_NO,Item_Code,Sum(QTy) as TotalQTY
16. FROM OrderDetails
17. where qty>=2
18. GROUP BY Order_NO,Item_Code
19. Order By Order_NO Desc,Item_Code
```
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.

1. --Group By Clause -- here this will display all the Order_no
2. Select Order_NO,Sum(QTy) as TotalQTY
3. FROM OrderDetails
4. GROUP BY Order_NO
5. 
6. -- Having Clause-- This will avoid the the sum(qty) less then 4
7. Select Order_NO,Sum(QTy) as TotalQTY
8. FROM OrderDetails
9. GROUP BY Order_NO
10. 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

<table>
<thead>
<tr>
<th>Order_NO</th>
<th>TotalQTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ord_001</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>Ord_002</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Ord_003</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order_NO</th>
<th>TotalQTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ord_001</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>
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.

1. **Sub Query**
   
   Here we used the Sub query in where clause to get all the Item_Code where the price>40
   now this sub

2. **Sub query result**
   
   query result we used in our main query to filter all the records which Item_code from Subquery result

3. ```
   SELECT * FROM ItemMasters
   WHERE Item_Code IN
   (SELECT Item_Code FROM ItemMasters WHERE price > 40)
   ```

4. **Sub Query with Insert Statement**

5. ```
   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'
   ```

6. **After insert we can see the result as**

7. ```
   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 resul we used in our main query to filter all the records whihc 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
VALUES
((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='Item001'
```

--- After insert we can see the result as

```
SELECT * FROM ItemMasters
```

<table>
<thead>
<tr>
<th>Item_Code</th>
<th>Item_Name</th>
<th>Price</th>
<th>TAX1</th>
<th>Discount</th>
<th>Description</th>
<th>IN_DATE</th>
<th>IN_USR_ID</th>
<th>UP_DATE</th>
<th>UP_USR_ID</th>
<th>Description</th>
<th>IN_DATE</th>
<th>IN_USR_ID</th>
<th>UP_DATE</th>
<th>UP_USR_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item001</td>
<td>Coke</td>
<td>55</td>
<td>1</td>
<td>0</td>
<td>Coke which need to be cold</td>
<td>2014-09-22 15:59:02 853</td>
<td>SHANU</td>
<td>2014-09-22 15:59:02 853</td>
<td>SHANU</td>
<td>Coke which need to be cold</td>
<td>2014-09-22 15:59:02 853</td>
<td>SHANU</td>
<td>2014-09-22 15:59:02 853</td>
<td>SHANU</td>
</tr>
<tr>
<td>Item002</td>
<td>Coffee</td>
<td>45</td>
<td>0</td>
<td>2</td>
<td>Coffee Might be Hot or Cold user choice</td>
<td>2014-09-22 15:59:02 853</td>
<td>SHANU</td>
<td>2014-09-22 15:59:02 853</td>
<td>SHANU</td>
<td>Coffee Might be Hot or Cold user choice</td>
<td>2014-09-22 15:59:02 853</td>
<td>SHANU</td>
<td>2014-09-22 15:59:02 853</td>
<td>SHANU</td>
</tr>
</tbody>
</table>

©2016 C# CORNER.

SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.
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

1. --Now we have used the simple join with out any condition this will display all the
2. --records with duplicate data to avoid this we see our next example with condition
3. SELECT * FROM Ordermasters,OrderDetails
4. --
5. SELECT * FROM Ordermasters as M, OrderDetails as D
6. WHERE M.Order_NO=D.Order_NO
7. AND M.Order_NO='Ord_001'

10. -- Now to make more better understanding we need to select the need fields from both
11. --table insted of displaying all column.
12. SELECT M.order_NO,M.Table_ID,D.Order_detail_no,Item_code,Notes,Qty
13. FROM Ordermasters as M, OrderDetails as D
14. WHERE M.Order_NO=D.Order_NO
15. -- Now let's Join 3 table
16. SELECT M.Order_NO,M.Table_ID,D.Order_detail_no,I.Item_Name,D.Notes,D.Qty,I.Price,
17. I.Price*D.Qty as TotalPrice
18. FROM Ordermasters as M, OrderDetails as D,ItemMasters as I
19. WHERE M.Order_NO=D.Order_NO AND D.Item_Code=I.Item_Code
Inner Join, Left Outer Join, Right Outer Join and Full outer Join

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

1. --INNER JOIN

   This will display the records which in both table Satisfy here i have used Like in where clause which display the

   \[
   \text{SELECT M.order_NO, M.Table_ID, D.Order_detail_no, I.Item_Name, D.Notes, D.Qty, I.Price, I.Price*D.Qty as TotalPrice}
   \]

   \[
   \text{FROM}
   \]

   \[
   \text{Ordermasters as M INNER JOIN OrderDetails as D WHERE M.Order_NO=D.Order_NO}
   \]

2. --LEFT OUTER JOIN

   This will display the records which Left side table Satisfy

   \[
   \text{SELECT M.order_NO, M.Table_ID, D.Order_detail_no, I.Item_Name, D.Notes, D.Qty, I.Price, I.Price*D.Qty as TotalPrice}
   \]

   \[
   \text{FROM}
   \]

   \[
   \text{Ordermasters as M LEFT OUTER JOIN OrderDetails as D WHERE M.Order_NO=D.Order_NO}
   \]

3. --RIGHT OUTER JOIN

   This will display the records which Left side table Satisfy

   \[
   \text{SELECT M.order_NO, M.Table_ID, D.Order_detail_no, I.Item_Name, D.Notes, D.Qty, I.Price, I.Price*D.Qty as TotalPrice}
   \]

   \[
   \text{FROM}
   \]

   \[
   \text{Ordermasters as M RIGHT OUTER JOIN OrderDetails as D WHERE M.Order_NO=D.Order_NO}
   \]
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.Price*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%'

<table>
<thead>
<tr>
<th>order_NO</th>
<th>Table_ID</th>
<th>Order_detail_no</th>
<th>Item_Name</th>
<th>Notes</th>
<th>Qty</th>
<th>Price</th>
<th>TotalPrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ord_001</td>
<td>T1</td>
<td>OR_Dt_001</td>
<td>Coke</td>
<td>Need very Cold</td>
<td>3</td>
<td>55</td>
<td>165</td>
</tr>
<tr>
<td>Ord_001</td>
<td>T1</td>
<td>OR_Dt_002</td>
<td>Potato Fry</td>
<td>very Hot</td>
<td>2</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>Ord_001</td>
<td>T1</td>
<td>OR_Dt_003</td>
<td>Chiken Burger</td>
<td>Very Spicy</td>
<td>4</td>
<td>125</td>
<td>500</td>
</tr>
<tr>
<td>Ord_002</td>
<td>T2</td>
<td>OR_Dt_004</td>
<td>Coffee</td>
<td>Need very Hot</td>
<td>2</td>
<td>40</td>
<td>80</td>
</tr>
<tr>
<td>Ord_002</td>
<td>T2</td>
<td>OR_Dt_005</td>
<td>Chiken Burger</td>
<td>very Hot</td>
<td>2</td>
<td>125</td>
<td>250</td>
</tr>
<tr>
<td>Ord_003</td>
<td>T3</td>
<td>OR_Dt_006</td>
<td>Chiken Burger</td>
<td>Very Spicy</td>
<td>4</td>
<td>125</td>
<td>500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>order_NO</th>
<th>Table_ID</th>
<th>Order_detail_no</th>
<th>Item_Name</th>
<th>Notes</th>
<th>Qty</th>
<th>Price</th>
<th>TotalPrice</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ord_001</td>
<td>T1</td>
<td>OR_Dt_001</td>
<td>Coke</td>
<td>Need very Cold</td>
<td>3</td>
<td>55</td>
<td>165</td>
</tr>
<tr>
<td>Ord_001</td>
<td>T1</td>
<td>OR_Dt_002</td>
<td>Potato Fry</td>
<td>very Hot</td>
<td>2</td>
<td>15</td>
<td>30</td>
</tr>
<tr>
<td>Ord_001</td>
<td>T1</td>
<td>OR_Dt_003</td>
<td>Chiken Burger</td>
<td>Very Spicy</td>
<td>4</td>
<td>125</td>
<td>500</td>
</tr>
<tr>
<td>Ord_002</td>
<td>T2</td>
<td>OR_Dt_004</td>
<td>Coffee</td>
<td>Need very Hot</td>
<td>2</td>
<td>40</td>
<td>80</td>
</tr>
<tr>
<td>Ord_002</td>
<td>T2</td>
<td>OR_Dt_005</td>
<td>Chiken Burger</td>
<td>very Hot</td>
<td>2</td>
<td>125</td>
<td>250</td>
</tr>
<tr>
<td>Ord_003</td>
<td>T3</td>
<td>OR_Dt_006</td>
<td>Chiken Burger</td>
<td>Very Spicy</td>
<td>4</td>
<td>125</td>
<td>500</td>
</tr>
</tbody>
</table>
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:

1. `Select column1,Column2 from Table1`
2. `Union`
3. `Select Column1,Column2 from Table2`
4. `Select column1,Column2 from Table1`
5. `Union All`
6. `Select Column1,Column2 from Table2`

Example

1. `--Select with different where condition which display the result as 2 Table result`
2. `select Item_Code,Item_Name,Price,Description FROM ItemMasters where price <=44`
3. `select Item_Code,Item_Name,Price,Description FROM ItemMasters where price >44`
4. `-- Union with same table but with different where condition now which result as one table which combine both the result.`
5. `select Item_Code,Item_Name,Price,Description FROM ItemMasters where price <=44`
6. `UNION`
7. `select Item_Code,Item_Name,Price,Description FROM ItemMasters where price >44`
8. `-- Union ALL with Join sample`
9. `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`
10. `FROM Ordermasters as M (NOLOCK) INNER JOIN  OrderDetails as D INNER JOIN ItemMasters as I`
11. `ON M.Order_NO=D.Order_NO INNER JOIN ItemMasters as I ON D.Item_Code=I.Item_Code WHERE I.Price <=44`
16. **Union** ALL
17. **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
18. **FROM**
19. Ordermasters as M (NOLOCK)  **Inner JOIN** OrderDetails as D
20. **ON** M.Order_NO=D.Order_NO **INNER JOIN** ItemMasters as I
21. **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
1. declare @sDate datetime,
2. @eDate datetime;
3.
4. select @sDate = getdate()-5,
5. @eDate = getdate()+16;
6. --select @sDate StartDate,@eDate EndDate
7. ;with cte as
8. 9. (select @sDate StartDate,'W'+convert(varchar(2),
10. DATEPART( wk, @sDate))+(''+convert(varchar(2),@sDate,106)+'') as 'SDT'
11. union all
12. select dateadd(DAY, 1, StartDate),
13. 'W'+convert(varchar(2),DATEPART( wk, StartDate))+(''+convert(varchar(2),
14. dateadd(DAY, 1, StartDate),106)+'') as 'SDT'
15. FROM cte
16. WHERE dateadd(DAY, 1, StartDate)<= @eDate
17. )
18. select * from cte
19. option (maxrecursion 0)
```
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**.
8.  union all
9.  select valuertype, dateadd(DAY, 1, StartDate),'
10.  W'+convert(varchar(2),DATEPART( wk, StartDate))+'
11.  ('+convert(varchar(2),dateadd(DAY, 1, StartDate),106)+')' as 'SDT'
12.  from cte
13.  where dateadd(DAY, 1, StartDate)<= @eDate)
14.  cte2
15.  AS
16.  (select '2' valuertype, @sDate StartDate,'
17.  W'+convert(varchar(2),DATEPART( wk, @sDate1))+'
18.  ('+convert(varchar(2),@sDate1,106)+')' as 'SDT'
19.  union all
20.  select '2' valuertype, dateadd(DAY, 1, StartDate),'
21.  W'+convert(varchar(2),DATEPART( wk, StartDate))+'
22.  ('+convert(varchar(2),dateadd(DAY, 1, StartDate),106)+')' as 'SDT'
23.  from cte
24.  where dateadd(DAY, 1, StartDate)<= @eDate1)
25.  )
26.  SELECT * FROM cte
27.  union all
28.  SELECT * FROM cte2
29.  option (maxrecursion 0)

<table>
<thead>
<tr>
<th>valuertype</th>
<th>StartDate</th>
<th>SDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2014-09-10 00:00:00.000</td>
<td>W37(10)</td>
</tr>
<tr>
<td>2</td>
<td>2014-09-11 00:00:00.000</td>
<td>W37(11)</td>
</tr>
<tr>
<td>3</td>
<td>2014-09-12 00:00:00.000</td>
<td>W37(12)</td>
</tr>
<tr>
<td>4</td>
<td>2014-09-13 00:00:00.000</td>
<td>W37(13)</td>
</tr>
<tr>
<td>5</td>
<td>2014-09-14 00:00:00.000</td>
<td>W37(14)</td>
</tr>
<tr>
<td>6</td>
<td>2014-09-15 00:00:00.000</td>
<td>W37(15)</td>
</tr>
<tr>
<td>7</td>
<td>2014-09-10 00:00:00.000</td>
<td>W37(10)</td>
</tr>
<tr>
<td>8</td>
<td>2014-09-11 00:00:00.000</td>
<td>W37(11)</td>
</tr>
<tr>
<td>9</td>
<td>2014-09-12 00:00:00.000</td>
<td>W37(12)</td>
</tr>
<tr>
<td>10</td>
<td>2014-09-13 00:00:00.000</td>
<td>W37(13)</td>
</tr>
<tr>
<td>11</td>
<td>2014-09-14 00:00:00.000</td>
<td>W37(14)</td>
</tr>
<tr>
<td>12</td>
<td>2014-09-15 00:00:00.000</td>
<td>W37(15)</td>
</tr>
<tr>
<td>13</td>
<td>2014-09-16 00:00:00.000</td>
<td>W37(16)</td>
</tr>
</tbody>
</table>
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 ColumnNames from yourTable

-- 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
```
27. We can also use the View to display with where condition and with selected fields.

28. Select order_Detail_NO, Table_ID, Item_Name, Price from myUnionVIEW where price > 40
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
1. -- Simple Pivot Example
2. SELECT * FROM ItemMasters
3. PIVOT(SUM(Price)
4. FOR ITEM_NAME IN ([Chiken Burger], [Coffee],[Coke]) AS PVTTable
5. 
6. -- Pivot with detail example
7. SELECT *
8. FROM ( 
9. SELECT ITEM_NAME, 
10. price as TotAmount 
11. FROM ItemMasters 
12. ) as s
13. PIVOT ( 
14. SUM(TotAmount) 
15. FOR [ITEM_NAME] IN ([Chiken Burger], [Coffee],[Coke])
16. ) AS MyPivot
```
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.

```sql
1. DECLARE @MyColumns AS NVARCHAR(MAX),
2. @SQLquery AS NVARCHAR(MAX)
3. -- here first we get all the ItemName which should be display in Columns we use this in our next pivot query
4. select @MyColumns = STUFF((SELECT ', ' + QUOTENAME(Item_NAME)
5. FROM ItemMasters
6. GROUP BY Item_NAME
7. ORDER BY Item_NAME
8. FOR XML PATH(''), TYPE
9. ).value('. ', 'NVARCHAR(MAX)')
10. ,1,1,'')
11. -- here we use the above all Item name to display its price as column and row display
12. set @SQLquery = N'SELECT ' + @MyColumns + N' FROM
13. ( SELECT
14. ITEM_NAME,
15. price as TotAmount
16. FROM ItemMasters
17. ) x
18. pivot
19. ( SUM(TotAmount)
20. for ITEM_NAME in (' + @MyColumns + N')
21. ) p '
```

25. `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 next 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 display 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;

<table>
<thead>
<tr>
<th></th>
<th>Chicken Burger</th>
<th>Coffee</th>
<th>Coke</th>
<th>Potato Fry</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>250</td>
<td>84</td>
<td>55</td>
<td>15</td>
</tr>
</tbody>
</table>
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

1. CREATE PROCEDURE [ProcedureName]
2. AS
3. BEGIN
4. -- Select or Update or Insert query.
5. END
6. To execute SP we use
7. exec ProcedureName

Example Select Sp with no parameter

1. --
2. ==============================================================
3. -- Author      : Shanu
4. -- Create date : 2014-09-15
5. -- Description : To Display Pivot Data
6. --
7. -- Modifier    : Shanu
8. -- Modify date : 2014-09-15
9. -- 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 next pivot query
    select @MyColumns = STUFF((
        SELECT ',' + QUOTENAME(Item_NAME)
        FROM ItemMasters
        GROUP BY Item_NAME
        ORDER BY Item_NAME
    ).value('.', 'NVARCHAR(MAX)')
          ,1,1,'')
    -- here we use the above all Item name to disoply 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
Note to alter the SP we use an Alter procedure procedureName.

Select and insert SP

1. -- Author : Shanu
2. -- Create date : 2014-09-15
3. -- Description : To Display Pivot Data
4. -- Latest
5. -- Modifier : Shanu
6. -- Modify date: 2014-09-15
7. 
8. -- exec USP_InsertItemMaster 'IceCream',120,0,0,'it should be in Cold','SHANU'
9. 
10. Create PROCEDURE [dbo].[USP_InsertItemMaster]
11. 
12. @ItemNAME VARCHAR(100) = '',
13. @Price INT = 0,
14. @TAX INT = 0,
15. @Discount INT = 0,
16. @Description VARCHAR(100) = '',
17. @UserID VARCHAR(20) = ''
18. )
19. AS
20. BEGIN
21. DECLARE @RowsCount AS int;
22. 
23. SELECT @RowsCount = count(*)+1 from [ItemMasters]
24. 
25. INSERT INTO [ItemMasters]
26. ([Item_Code],[Item_Name],[Price],[TAX1],[Discount],[Description],[IN_DATE],[IN_USR_ID],[UP_DATE],[UP_USR_ID])
27. VALUES
28. ('Item00'+ CAST(@RowsCount AS VARCHAR(10)))
29. ,@ItemNAME
30. ,@Price
31. ,@TAX
32. ,@Discount
33. ,@Description
34. ,getdate()
35. ,@UserID
36. ,getdate()
37. ,@UserID
38. 
39. END
```
-- Procedure: USEP_InsertItemMaster

ALTER PROCEDURE [dbo].[USEP_InsertItemMaster]

BEGIN

DECLARE @RowCount AS int;

SELECT @RowCount = count(*) - 1 FROM [ItemMasters]

INSERT INTO [ItemMasters]

VALUES

('Item00' + CAST(@RowCount AS VARCHAR(10)),

PRICE, 0,

TAX, 0,

Discount, 0,

Description, ''

UserID, ''

END
```

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.

```
1. <a style="font-size: 14px; color: #111111">
2. Create Function functionName
3. As
4. Begin
5. END
6. </a>
```

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

```
1. <a style="font-size: 14px; color: #111111">
2. --=================================================================================
3. -- Author : Shanu
4. -- Create date : 2014-09-15
5. -- Description : To Display Pivot Data
6. -- Latest
7. -- Modifier  : Shanu
8. -- Modify date : 2014-09-15
9. Alter FUNCTION [dbo].[ufnSelectitemMaster]()
10. RETURNS int
11. AS
12. -- Returns total Row count of Item Master.
13. BEGIN
14. DECLARE @RowsCount AS int;
15. SELECT @RowsCount = count(*)+1 FROM ItemMasters
16. RETURN @RowsCount;
17. END
```

©2016 C# CORNER.
SHARE THIS DOCUMENT AS IT IS. PLEASE DO NOT REPRODUCE, REPUBLISH, CHANGE OR COPY.
-- to View Function we use select and function Name
select [dbo].[ufnSelectitemMaster]()

--- function to get the last date of 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].[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 function Name
select [dbo].[ufnSelectitemMaster]()

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

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.

```
1. declare @FromDate datetime,
2. @ToDate datetime;
3. IF OBJECT_ID('tempdb..#TEMP_EveryWk_Snday') IS NOT NULL
4.   DROP TABLE #TEMP_EveryWk_Snday
5. DECLARE @TOTALCount INT
6. SET @FromDate = getdate();
7. SET @ToDate = DATEADD(Month, 3, getdate());
8. SELECT @TOTALCount = DATEDIFF(dd, @FromDate, @ToDate);
9. WITH d AS
10. ( SELECT top (@TOTALCount) AllDays = DATEADD(DAY, ROW_NUMBER() OVER (ORDER BY object_id), REPLACE(@FromDate, '-', '',''))
11. FROM sys.all_objects
12. )
13. SELECT Distinct DATEADD(DAY, 1 - DATEPART(WEEKDAY, AllDays),
14.   CAST(AllDays as DATE))WkStartSundays
15. INTO #TEMP_EveryWk_Snday
16. FROM d
17. WHERE AllDays <= @ToDate
18. Select WkStartSundays WEEKSTART_Sunday,
19. DATENAME(dw, WkStartSundays) Day_Name
20. FROM #TEMP_EveryWk_Snday
```