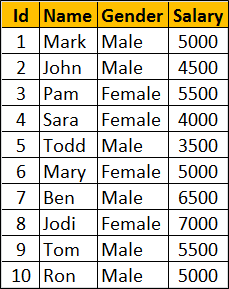
In this video we will discuss the power and use of Over clause in SQL Server.   
  
   
  
The **OVER** clause combined with **PARTITION BY** is used to break up data into partitions.   
**Syntax :** function (...) OVER (PARTITION BY col1, Col2, ...)

The specified function operates for each partition.   
  
**For example :**  
COUNT(Gender) OVER (PARTITION BY Gender) will partition the data by **GENDER**i.e there will 2 partitions (Male and Female) and then the COUNT() function is applied over each partition.   
  
Any of the following functions can be used. Please note this is not the complete list.  
COUNT(), AVG(), SUM(), MIN(), MAX(), ROW\_NUMBER(), RANK(), DENSE\_RANK() etc.   
  
**Example :**We will use the following **Employees table**for the examples in this video.   
   
  
**SQl Script to create Employees table**

Create Table Employees

(

     Id int primary key,

     Name nvarchar(50),

     Gender nvarchar(10),

     Salary int

)

Go

Insert Into Employees Values (1, 'Mark', 'Male', 5000)

Insert Into Employees Values (2, 'John', 'Male', 4500)

Insert Into Employees Values (3, 'Pam', 'Female', 5500)

Insert Into Employees Values (4, 'Sara', 'Female', 4000)

Insert Into Employees Values (5, 'Todd', 'Male', 3500)

Insert Into Employees Values (6, 'Mary', 'Female', 5000)

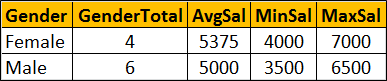
Insert Into Employees Values (7, 'Ben', 'Male', 6500)

Insert Into Employees Values (8, 'Jodi', 'Female', 7000)

Insert Into Employees Values (9, 'Tom', 'Male', 5500)

Insert Into Employees Values (10, 'Ron', 'Male', 5000)

Go

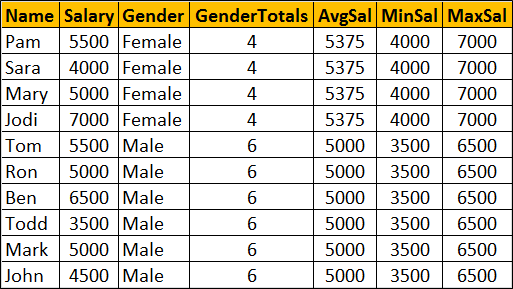
Write a query to retrieve total count of employees by Gender. Also in the result we want Average, Minimum and Maximum salary by Gender. The result of the query should be as shown below.   
   
  
This can be very easily achieved using a simple **GROUP BY** query as show below.

SELECT Gender, COUNT(\*) AS GenderTotal, AVG(Salary) AS AvgSal,

        MIN(Salary) AS MinSal, MAX(Salary) AS MaxSal

FROM Employees

GROUP BY Gender

What if we want **non-aggregated values** (like employee Name and Salary) in result set along with aggregated values   
   
  
You cannot include **non-aggregated** columns in the **GROUP BY** query.

SELECT Name, Salary, Gender, COUNT(\*) AS GenderTotal, AVG(Salary) AS AvgSal,

        MIN(Salary) AS MinSal, MAX(Salary) AS MaxSal

FROM Employees

GROUP BY Gender

The above query will result in the following error  
Column 'Employees.Name' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause   
  
One way to achieve this is by including the aggregations in a subquery and then **JOINING** it with the main query as shown in the example below. Look at the amount of T-SQL code we have to write.

SELECT Name, Salary, Employees.Gender, Genders.GenderTotals,

        Genders.AvgSal, Genders.MinSal, Genders.MaxSal

FROM Employees

INNER JOIN

(SELECT Gender, COUNT(\*) AS GenderTotals,

          AVG(Salary) AS AvgSal,

         MIN(Salary) AS MinSal, MAX(Salary) AS MaxSal

FROM Employees

GROUP BY Gender) AS Genders

ON Genders.Gender = Employees.Gender

Better way of doing this is by using the **OVER**clause combined with **PARTITION BY**

SELECT Name, Salary, Gender,

        COUNT(Gender) OVER(PARTITION BY Gender) AS GenderTotals,

        AVG(Salary) OVER(PARTITION BY Gender) AS AvgSal,

        MIN(Salary) OVER(PARTITION BY Gender) AS MinSal,

        MAX(Salary) OVER(PARTITION BY Gender) AS MaxSal

FROM Employees