Ranking Functions

Ranking functions are a subset of the built-in functions in SQL Server. The ranking function helps us assign a rank value for each row in your result-set.

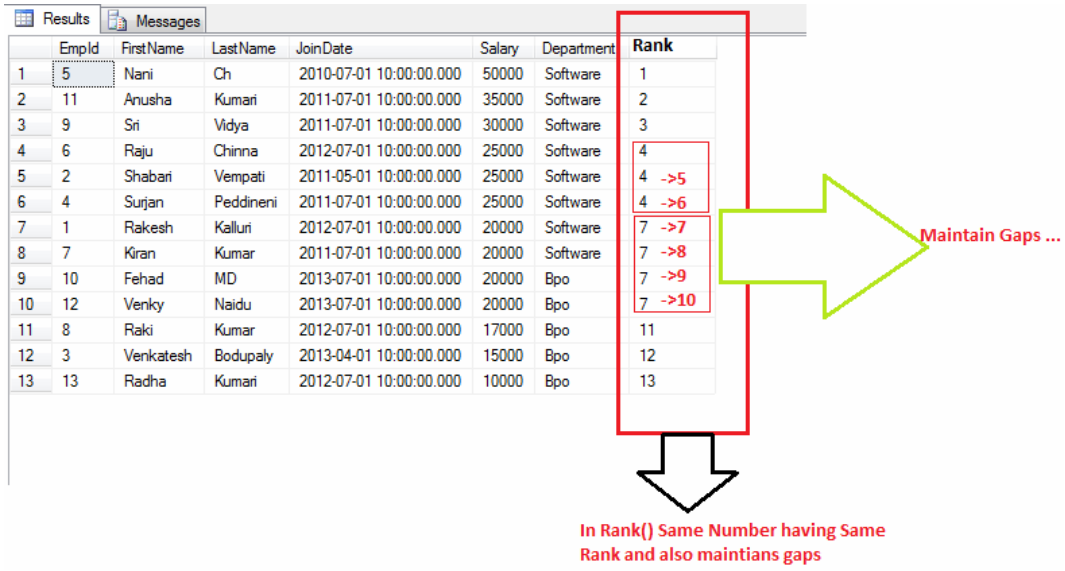
There are four ranking functions in SQL server: **RANK(), DENSE_RANK(),ROW_NUMBER() , NTILE()**

1. **RANK():** Returns the rank of each row within a result set. The rank of a row is one plus the number of ranks that come before the row in question. So RANK() may return values with gaps when there are ties. You could use the partition on the OVER clause to have partitioning in the rank values.

   Syntax: RANK() OVER ([<ORDER BY CLUASE >)

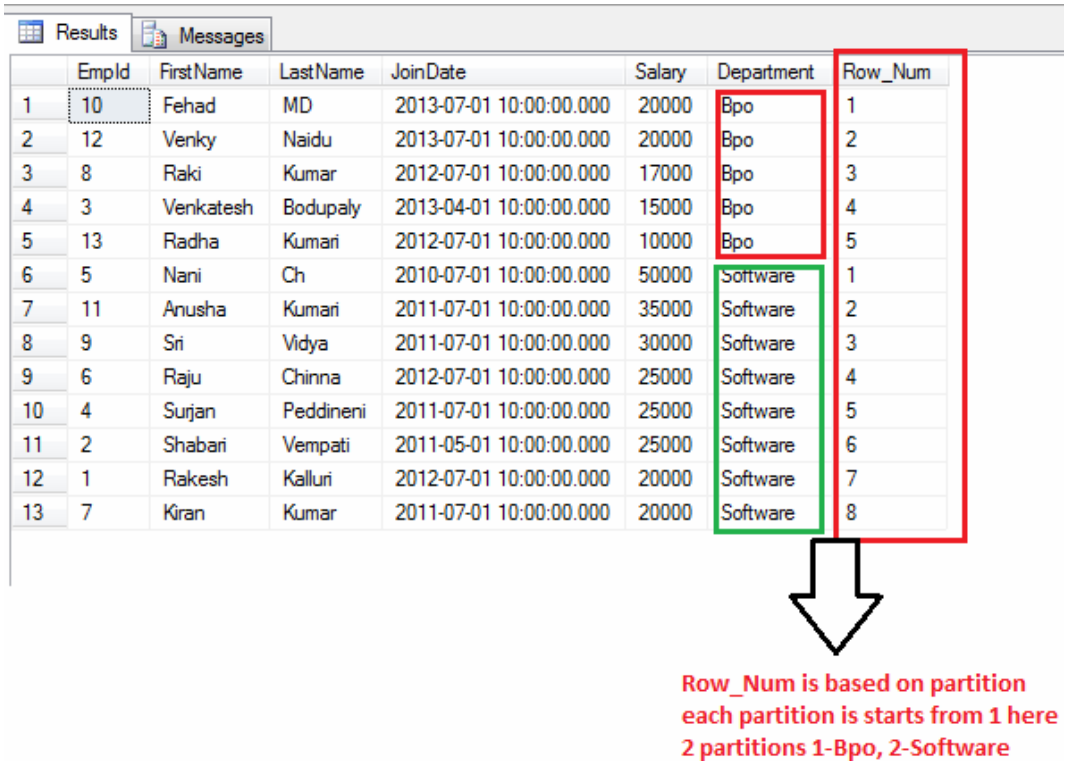
   RANK() OVER ([PARTITION BY CLAUSE] <ORDER BY CLUASE >)

   e.g: select * , rank() over (order by Salary desc) as [Rank] from Employee



	EmpId	FirstName	LastName	JoinDate	Salary	Department	Rank
1	5	Nani	Ch	2010-07-01 10:00:00.000	50000	Software	1
2	11	Anusha	Kumari	2011-07-01 10:00:00.000	35000	Software	2
3	9	Sri	Vidya	2011-07-01 10:00:00.000	30000	Software	3
4	6	Raju	Chinna	2012-07-01 10:00:00.000	25000	Software	4
5	2	Shabari	Vempati	2011-05-01 10:00:00.000	25000	Software	4 ->5
6	4	Surjan	Peddineni	2011-07-01 10:00:00.000	25000	Software	4 ->6
7	1	Rakesh	Kalluri	2012-07-01 10:00:00.000	20000	Software	7 ->7
8	7	Kiran	Kumar	2011-07-01 10:00:00.000	20000	Software	7 ->8
9	10	Fehad	MD	2013-07-01 10:00:00.000	20000	Bpo	7 ->9
10	12	Venky	Naidu	2013-07-01 10:00:00.000	20000	Bpo	7 ->10
11	8	Raki	Kumar	2012-07-01 10:00:00.000	17000	Bpo	11
12	3	Venkatesh	Bodupaly	2013-04-01 10:00:00.000	15000	Bpo	12
13	13	Radha	Kumari	2012-07-01 10:00:00.000	10000	Bpo	13

Select * ,row_number() over (partition by Department order by Salary desc) as Row_Num frm Employee



Results | Messages

	EmpId	FirstName	LastName	JoinDate	Salary	Department	Row_Num
1	10	Fehad	MD	2013-07-01 10:00:00.000	20000	Bpo	1
2	12	Venky	Naidu	2013-07-01 10:00:00.000	20000	Bpo	2
3	8	Raki	Kumar	2012-07-01 10:00:00.000	17000	Bpo	3
4	3	Venkatesh	Bodupaly	2013-04-01 10:00:00.000	15000	Bpo	4
5	13	Radha	Kumari	2012-07-01 10:00:00.000	10000	Bpo	5
6	5	Nani	Ch	2010-07-01 10:00:00.000	50000	Software	1
7	11	Anusha	Kumari	2011-07-01 10:00:00.000	35000	Software	2
8	9	Sri	Vidya	2011-07-01 10:00:00.000	30000	Software	3
9	6	Raju	Chinna	2012-07-01 10:00:00.000	25000	Software	4
10	4	Surjan	Peddineni	2011-07-01 10:00:00.000	25000	Software	5
11	2	Shabari	Vempati	2011-05-01 10:00:00.000	25000	Software	6
12	1	Rakesh	Kalluri	2012-07-01 10:00:00.000	20000	Software	7
13	7	Kiran	Kumar	2011-07-01 10:00:00.000	20000	Software	8

Row_Num is based on partition
each partition is starts from 1 here
2 partitions 1-Bpo, 2-Software

2. **ROW_NUMBER():** Returns the sequential number for each row within a result set. ROW_NUMBER always starts with one.

Syntax: ROW_NUMBER() OVER ([PARTITION BY CLAUSE] <ORDER BY CLUASE>)
ROW_NUMBER() OVER <ORDER BY CLUASE>)

```
Select * ,row_number() over (order by Salary desc) as Row_Num from Employee
```

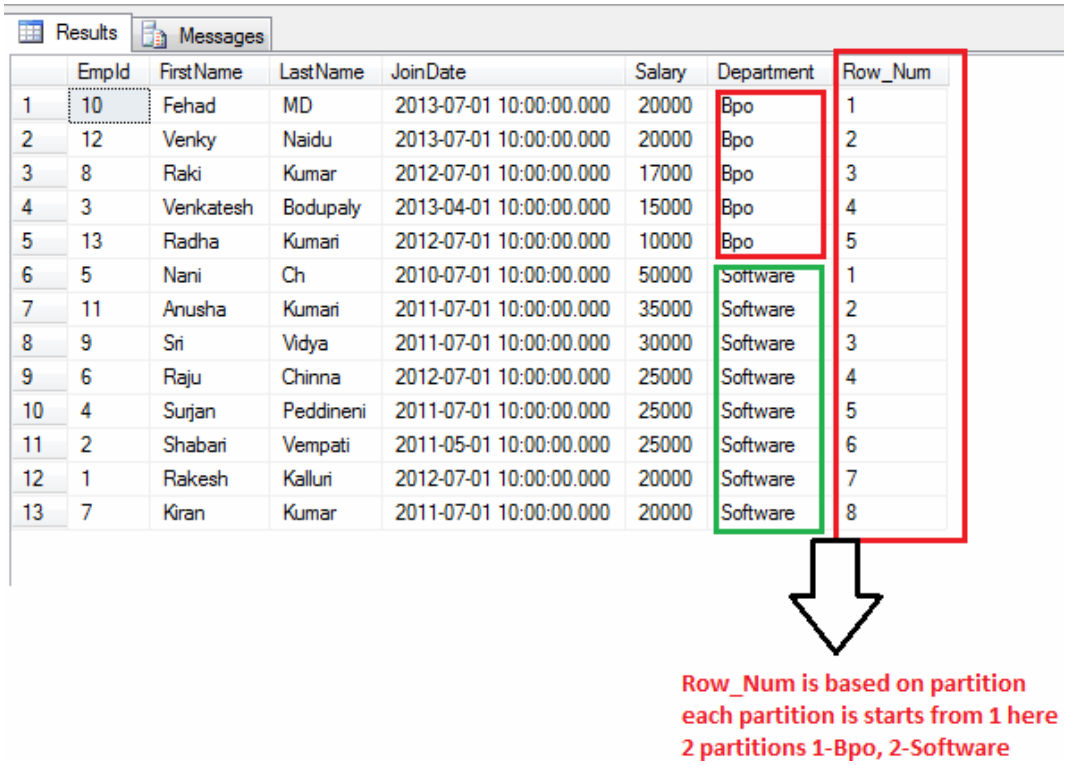
Results | Messages

	EmpId	FirstName	LastName	JoinDate	Salary	Department	Row_Num
1	5	Nani	Ch	2010-07-01 10:00:00.000	50000	Software	1
2	11	Anusha	Kumari	2011-07-01 10:00:00.000	35000	Software	2
3	9	Sri	Vidya	2011-07-01 10:00:00.000	30000	Software	3
4	6	Raju	Chinna	2012-07-01 10:00:00.000	25000	Software	4
5	2	Shabari	Vempati	2011-05-01 10:00:00.000	25000	Software	5
6	4	Surjan	Peddineni	2011-07-01 10:00:00.000	25000	Software	6
7	1	Rakesh	Kalluri	2012-07-01 10:00:00.000	20000	Software	7
8	7	Kiran	Kumar	2011-07-01 10:00:00.000	20000	Software	8
9	10	Fehad	MD	2013-07-01 10:00:00.000	20000	Bpo	9
10	12	Venky	Naidu	2013-07-01 10:00:00.000	20000	Bpo	10
11	8	Raki	Kumar	2012-07-01 10:00:00.000	17000	Bpo	11
12	3	Venkatesh	Bodupaly	2013-04-01 10:00:00.000	15000	Bpo	12
13	13	Radha	Kumari	2012-07-01 10:00:00.000	10000	Bpo	13

Row_Num starts from 1 based on order by cluase

--Row_Number() with using partition clause

```
Select *,row_number() over (partition by Department order by Salary desc) as Row_Num
 from Employee
```



Results | Messages

	EmpId	FirstName	LastName	JoinDate	Salary	Department	Row_Num
1	10	Fehad	MD	2013-07-01 10:00:00.000	20000	Bpo	1
2	12	Venky	Naidu	2013-07-01 10:00:00.000	20000	Bpo	2
3	8	Raki	Kumar	2012-07-01 10:00:00.000	17000	Bpo	3
4	3	Venkatesh	Bodupaly	2013-04-01 10:00:00.000	15000	Bpo	4
5	13	Radha	Kumari	2012-07-01 10:00:00.000	10000	Bpo	5
6	5	Nani	Ch	2010-07-01 10:00:00.000	50000	Software	1
7	11	Anusha	Kumari	2011-07-01 10:00:00.000	35000	Software	2
8	9	Sri	Vidya	2011-07-01 10:00:00.000	30000	Software	3
9	6	Raju	Chinna	2012-07-01 10:00:00.000	25000	Software	4
10	4	Surjan	Peddineni	2011-07-01 10:00:00.000	25000	Software	5
11	2	Shabari	Vempati	2011-05-01 10:00:00.000	25000	Software	6
12	1	Rakesh	Kalluri	2012-07-01 10:00:00.000	20000	Software	7
13	7	Kiran	Kumar	2011-07-01 10:00:00.000	20000	Software	8

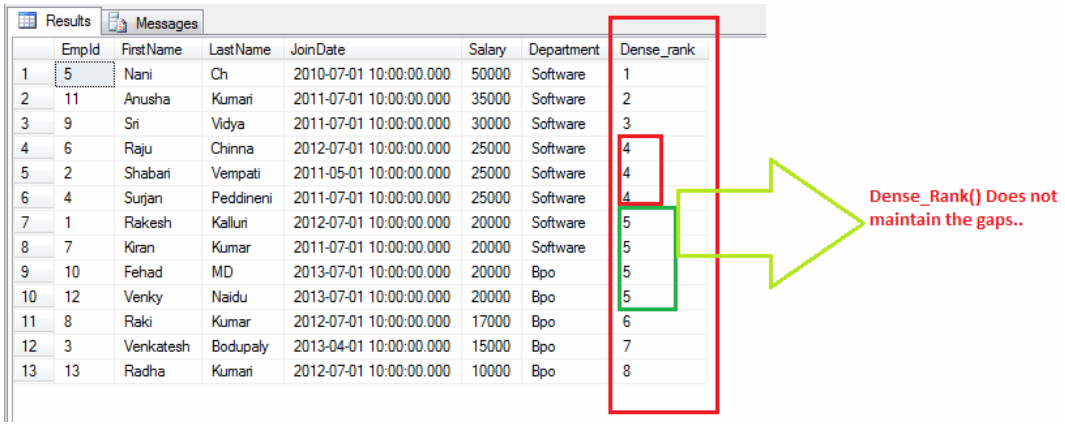
Row_Num is based on partition
each partition is starts from 1 here
2 partitions 1-Bpo, 2-Software

3. **DENSE_RANK():** Returns the rank of rows within the partition of a result set. The rank of a row is one plus the number of distinct ranks that come before the row in question. As the name suggests, DENSE_RANK returns the rank values without gaps and this is the difference between RANK() and DENSE_RANK().

   Syntax: DENSE_RANK() OVER ([PARTITION BY CLAUSE] <ORDER BY CLUASE >)

--dense_rank() with out using partition cluase

```
Select * ,dense_rank() over (order by Salary desc) as [Dense_rank] from Employee
```



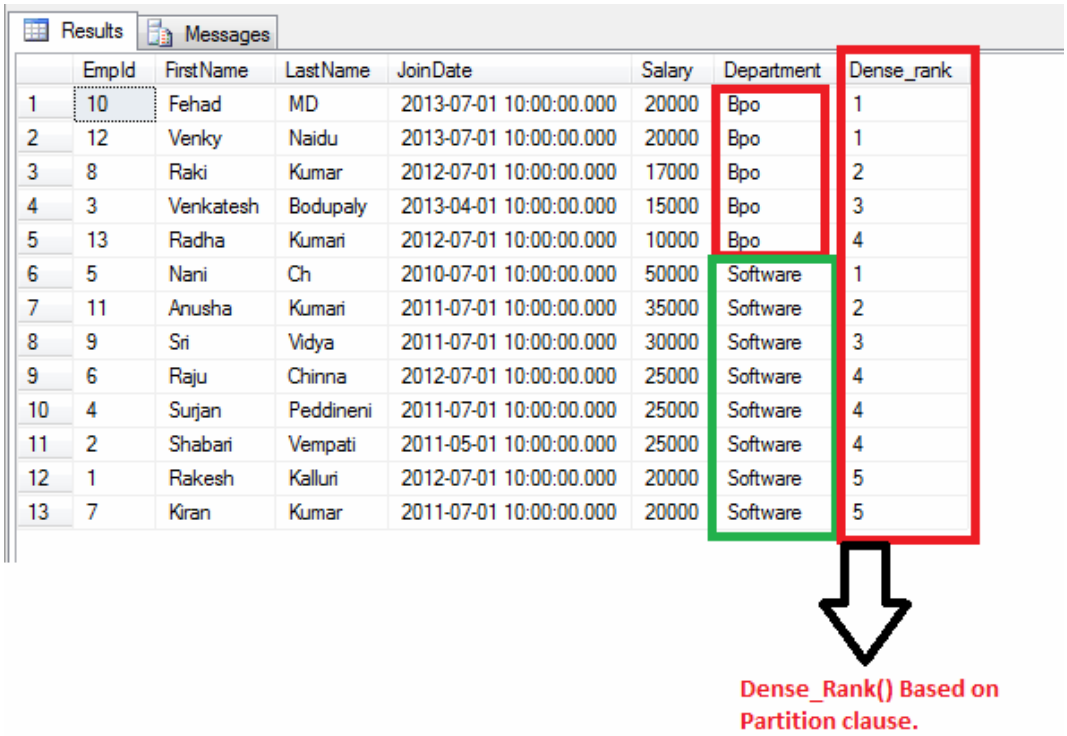
Results | Messages

	EmpId	FirstName	LastName	JoinDate	Salary	Department	Dense_rank
1	5	Nani	Ch	2010-07-01 10:00:00.000	50000	Software	1
2	11	Anusha	Kumari	2011-07-01 10:00:00.000	35000	Software	2
3	9	Sri	Vidya	2011-07-01 10:00:00.000	30000	Software	3
4	6	Raju	Chinna	2012-07-01 10:00:00.000	25000	Software	4
5	2	Shabari	Vempati	2011-05-01 10:00:00.000	25000	Software	4
6	4	Surjan	Peddineni	2011-07-01 10:00:00.000	25000	Software	4
7	1	Rakesh	Kalluri	2012-07-01 10:00:00.000	20000	Software	5
8	7	Kiran	Kumar	2011-07-01 10:00:00.000	20000	Software	5
9	10	Fehad	MD	2013-07-01 10:00:00.000	20000	Bpo	5
10	12	Venky	Naidu	2013-07-01 10:00:00.000	20000	Bpo	5
11	8	Raki	Kumar	2012-07-01 10:00:00.000	17000	Bpo	6
12	3	Venkatesh	Bodupaly	2013-04-01 10:00:00.000	15000	Bpo	7
13	13	Radha	Kumari	2012-07-01 10:00:00.000	10000	Bpo	8

Dense_Rank() Does not maintain the gaps..

--dense_rank() with using partition cluase

Select * ,dense_rank() over (partition by Department order by Salary desc) as [Dense_rank] from Employee



Results | Messages

	EmpId	FirstName	LastName	JoinDate	Salary	Department	Dense_rank
1	10	Fehad	MD	2013-07-01 10:00:00.000	20000	Bpo	1
2	12	Venky	Naidu	2013-07-01 10:00:00.000	20000	Bpo	1
3	8	Raki	Kumar	2012-07-01 10:00:00.000	17000	Bpo	2
4	3	Venkatesh	Bodupaly	2013-04-01 10:00:00.000	15000	Bpo	3
5	13	Radha	Kumari	2012-07-01 10:00:00.000	10000	Bpo	4
6	5	Nani	Ch	2010-07-01 10:00:00.000	50000	Software	1
7	11	Anusha	Kumari	2011-07-01 10:00:00.000	35000	Software	2
8	9	Sri	Vidya	2011-07-01 10:00:00.000	30000	Software	3
9	6	Raju	Chinna	2012-07-01 10:00:00.000	25000	Software	4
10	4	Surjan	Peddineni	2011-07-01 10:00:00.000	25000	Software	4
11	2	Shabari	Vempati	2011-05-01 10:00:00.000	25000	Software	4
12	1	Rakesh	Kalluri	2012-07-01 10:00:00.000	20000	Software	5
13	7	Kiran	Kumar	2011-07-01 10:00:00.000	20000	Software	5

Dense_Rank() Based on Partition clause.

4. **NTILE()**

This distributes the result set into a specified number of groups. The groups are sequential numbers, starting at one. For each row, NTILE returns the number of the group to which the row belongs. The rows are evenly distributed into each group when the total

rows in the result set is divisible by the number of groups specified. If this is not the case there size of group may vary.

NTILE(INTEGER_EXPRESSION) OVER ([PARTITION BY CLAUSE] <ORDER BY CLUASE >):

--ntile(input_exp) with out using partition cluase

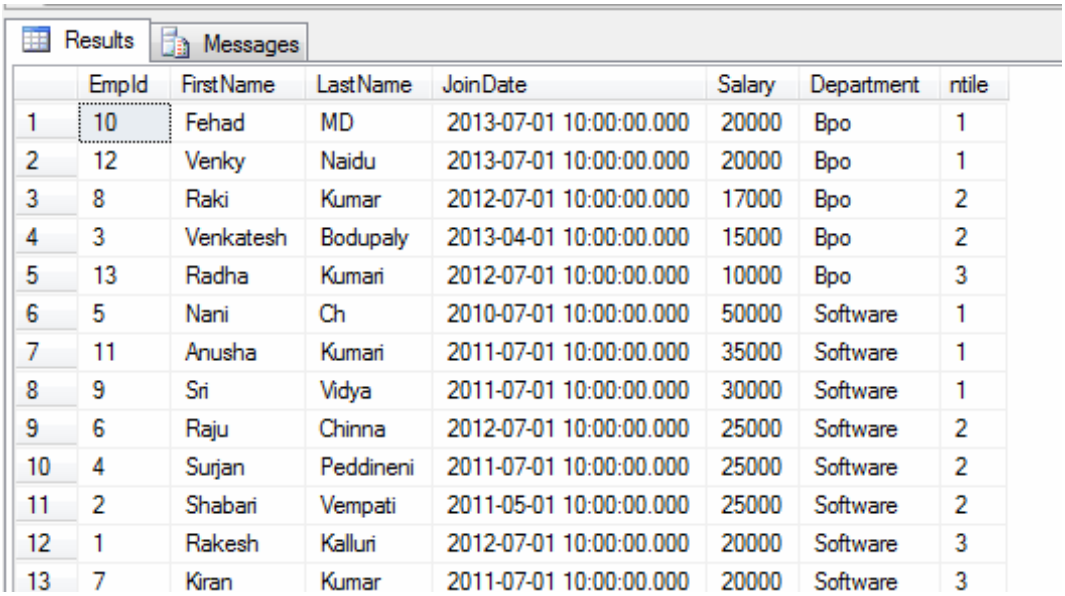
Select * ,ntile(3) over (order by Salary desc) as [ntile] from Employee

Results | Messages

	EmpId	FirstName	LastName	JoinDate	Salary	Department	Dense_rank
1	5	Nani	Ch	2010-07-01 10:00:00.000	50000	Software	1
2	11	Anusha	Kumari	2011-07-01 10:00:00.000	35000	Software	1
3	9	Sri	Vidya	2011-07-01 10:00:00.000	30000	Software	1
4	6	Raju	Chinna	2012-07-01 10:00:00.000	25000	Software	1
5	2	Shabari	Vempati	2011-05-01 10:00:00.000	25000	Software	1
6	4	Surjan	Peddineni	2011-07-01 10:00:00.000	25000	Software	2
7	1	Rakesh	Kalluri	2012-07-01 10:00:00.000	20000	Software	2
8	7	Kiran	Kumar	2011-07-01 10:00:00.000	20000	Software	2
9	10	Fehad	MD	2013-07-01 10:00:00.000	20000	Bpo	2
10	12	Venky	Naidu	2013-07-01 10:00:00.000	20000	Bpo	3
11	8	Raki	Kumar	2012-07-01 10:00:00.000	17000	Bpo	3
12	3	Venkatesh	Bodupaly	2013-04-01 10:00:00.000	15000	Bpo	3
13	13	Radha	Kumari	2012-07-01 10:00:00.000	10000	Bpo	3

--ntile(input_exp) with using partition cluase

select * ,ntile(3) over (partition by De6partment order by Salary desc) as [ntile] from Employee



Results | Messages

	EmpId	FirstName	LastName	JoinDate	Salary	Department	ntile
1	10	Fehad	MD	2013-07-01 10:00:00.000	20000	Bpo	1
2	12	Venky	Naidu	2013-07-01 10:00:00.000	20000	Bpo	1
3	8	Raki	Kumar	2012-07-01 10:00:00.000	17000	Bpo	2
4	3	Venkatesh	Bodupaly	2013-04-01 10:00:00.000	15000	Bpo	2
5	13	Radha	Kumari	2012-07-01 10:00:00.000	10000	Bpo	3
6	5	Nani	Ch	2010-07-01 10:00:00.000	50000	Software	1
7	11	Anusha	Kumari	2011-07-01 10:00:00.000	35000	Software	1
8	9	Sri	Vidya	2011-07-01 10:00:00.000	30000	Software	1
9	6	Raju	Chinna	2012-07-01 10:00:00.000	25000	Software	2
10	4	Surjan	Peddineni	2011-07-01 10:00:00.000	25000	Software	2
11	2	Shabari	Vempati	2011-05-01 10:00:00.000	25000	Software	2
12	1	Rakesh	Kalluri	2012-07-01 10:00:00.000	20000	Software	3
13	7	Kiran	Kumar	2011-07-01 10:00:00.000	20000	Software	3

```
--create Employee table

create table Employee

(

  EmpId int identity(1,1) primary key,

  FirstName varchar(100),

  LastName varchar(100),

  JoinDate datetime ,

  Salary int ,

  Department varchar(20)

)
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department)values('Rakesh','Kalluri','2012-07-01 10:00:00.000',20000,'Software')
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department)values('Shabari','Vempati','2011-05-01 10:00:00.000',25000,'Software')
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department)values('Venkatesh','Bodupaly','2013-04-01 10:00:00.000',15000,'Bpo')
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department)values('Surjan','Peddineni','2011-07-01 10:00:00.000',25000,'Software')
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department) values('Nani','Ch','2010-07-01 10:00:00.000',50000,'Software')
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department) values('Raju','Chinna','2012-07-01 10:00:00.000',25000,'Software')
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department) values('Kiran','Kumar','2011-07-01 10:00:00.000',20000,'Software')
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department) values('Raki','Kumar','2012-07-01 10:00:00.000',17000,'Bpo')
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department) values('Sri','Vidya','2011-07-01 10:00:00.000',30000,'Software')
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department) values('Fehad','MD','2013-07-01 10:00:00.000',20000,'Bpo')
```

```
insert into Employee(FirstName,LastName,JoinDate,Salary,Department)values('Anusha','Kumari','2011-07-01 10:00:00.000',35000,'Software')
```

insert into Employee(FirstName,LastName,JoinDate,Salary,Department) values('Venky','Naidu','2013-07-01 10:00:00.000',20000,'Bpo')

insert into Employee(FirstName,LastName,JoinDate,Salary,Department)values('Radha','Kumari','2012-07-01 10:00:00.000',10000,'Bpo')