**Are Cast and Convert Different?**

CAST and CONVERT are both used to convert data from one data type to another

Here is an example using both CAST and CONVERT in the same statement:

SELECT CAST ('10' as int) \* 20,

CONVERT (int, '10') \* 20

## Differences between CAST and CONVERT

CAST is part of the ANSI-SQL specification; whereas, CONVERT is not.  In fact, CONVERT is SQL implementation specific.

CONVERT differences lie in that it accepts an optional style parameter which is used for formatting.

For example, when converting a DateTime datatype to Varchar, you can specify the resulting date’s format, such as YYYY/MM/DD or MM/DD/YYYY.

SELECT CONVERT(VARCHAR,GETDATE(),101) as MMDDYYYY,

CONVERT(VARCHAR,GETDATE(),111) as YYYYMMDD

## Should I use CAST or Convert?

Unless you have some specific formatting requirements you’re trying to address during the conversion, I would stick with using the CAST function.  There are several reason I can think of:

1. CAST is ANSI-SQL compliant; therefore, more apt to be used in other database implementation.
2. There is no performance penalty using CAST.
3. I think CAST is easier to read, and since it is part of the ANSI specification, your non-SQLServer DBA think so too!

String Function

[CHARINDEX (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms186323(v=sql.105).aspx)

[LEFT (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms177601(v=sql.105).aspx)

[LEN (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms190329(v=sql.105).aspx)

[LOWER (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms174400(v=sql.105).aspx)

[LTRIM (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms177827(v=sql.105).aspx)

[REPLACE (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms186862(v=sql.105).aspx)

[REVERSE (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms180040(v=sql.105).aspx)

[REVERT (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms178632(v=sql.105).aspx)

[RIGHT (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms177532(v=sql.105).aspx)

[RTRIM (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms178660(v=sql.105).aspx)

[STR (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms189527(v=sql.105).aspx)

[SUBSTRING (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms187748(v=sql.105).aspx)

[UPPER (Transact-SQL)](https://technet.microsoft.com/en-us/library/ms180055(v=sql.105).aspx)

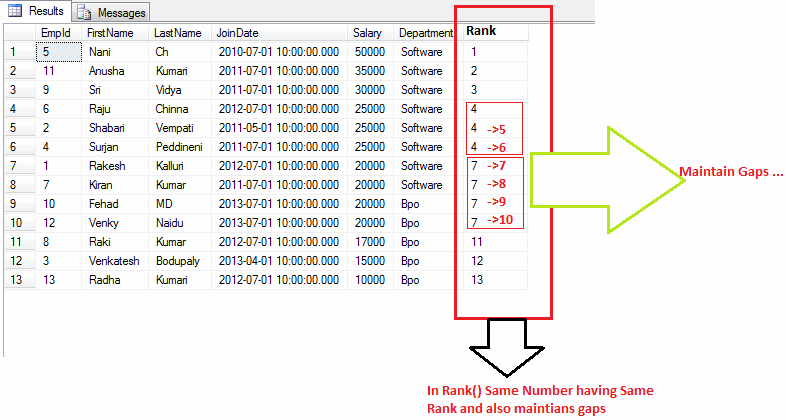
SUBSTRING()

Rank() and it’s different variant

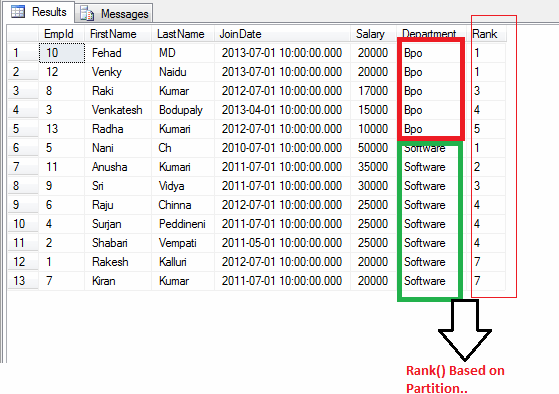
1. **ROW\_NUMBER() OVER ([PARTITION BY CLAUSE] <ORDER BY CLUASE>):**Returns the sequantial number of a row within the a partition of result set at 1 for the first row of the each partition.
2. **RANK() OVER ([PARTITION BY CLAUSE] <ORDER BY CLUASE >):**Returns rank for rows within the partition of result set.
3. **DENSE\_RANK() OVER ([PARTITION BY CLAUSE] <ORDER BY CLUASE >):**Returns rank for rows within the partition of result set.With out any gaps in the ranking.
4. **NTILE(INTEGER\_EXPRESSION) OVER ([PARTITION BY CLAUSE] <ORDER BY CLUASE >):**Distributes the rows in an ordered partition into a specified number of groups.

**Examples**

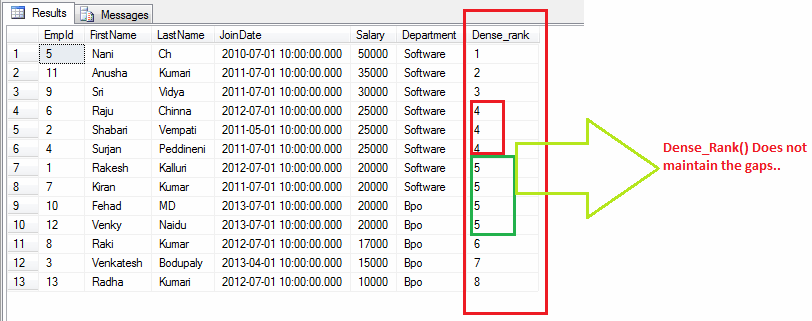
1. --create Employee table
2. **create** **table** Employee
3. (
4. EmpId **int** identity(1,1) **primary** **key**,
5. FirstName **varchar**(100),
6. LastName **varchar**(100),
7. JoinDate datetime ,
8. Salary **int** ,
9. Department **varchar**(20)
10. )
11. --Insert data to Employee table
13. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department)**values**('Rakesh','Kalluri','2012-07-01 10:00:00.000',20000,'Software')
14. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department)**values**('Shabari','Vempati','2011-05-01 10:00:00.000',25000,'Software')
15. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department)**values**('Venkatesh','Bodupaly','2013-04-01 10:00:00.000',15000,'Bpo')
16. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department)**values**('Surjan','Peddineni','2011-07-01 10:00:00.000',25000,'Software')
17. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department) **values**('Nani','Ch','2010-07-01 10:00:00.000',50000,'Software')
18. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department) **values**('Raju','Chinna','2012-07-01 10:00:00.000',25000,'Software')
19. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department) **values**('Kiran','Kumar','2011-07-01 10:00:00.000',20000,'Software')
20. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department) **values**('Raki','Kumar','2012-07-01 10:00:00.000',17000,'Bpo')
21. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department) **values**('Sri','Vidya','2011-07-01 10:00:00.000',30000,'Software')
22. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department) **values**('Fehad','MD','2013-07-01 10:00:00.000',20000,'Bpo')
23. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department)**values**('Anusha','Kumari','2011-07-01 10:00:00.000',35000,'Software')
24. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department) **values**('Venky','Naidu','2013-07-01 10:00:00.000',20000,'Bpo')
25. **insert** **into** Employee(FirstName,LastName,JoinDate,Salary,Department)**values**('Radha','Kumari','2012-07-01 10:00:00.000',10000,'Bpo')
26. --selecting data from Employee table
27. **select** \* **from** Employee
28. --Row\_Number() with out using partition cluase
29. **select** \* ,row\_number() over (**order** **by** Salary **desc**) **as** Row\_Num **from** Employee
30. --Row\_Number() with using partition cluase
31. **select** \* ,row\_number() over (partition **by** Department **order** **by** Salary **desc**) **as** Row\_Num **from** Employee
32. --rank() with out using partition cluase
33. **select** \* ,rank() over (**order** **by** Salary **desc**) **as** [Rank] **from** Employee



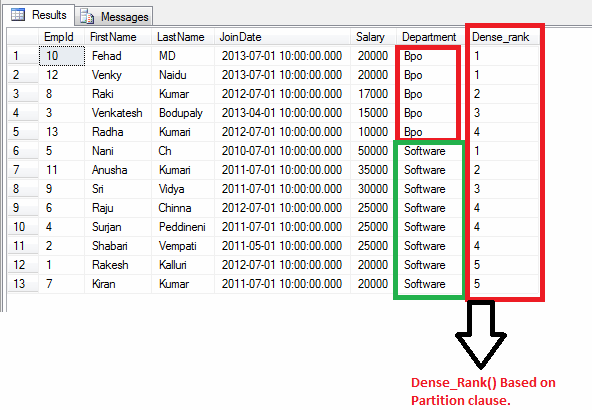
1. --rank() with using partition cluase
2. **select** \* ,rank() over (partition **by** Department **order** **by** Salary **desc**) **as** [Rank] **from** Employee



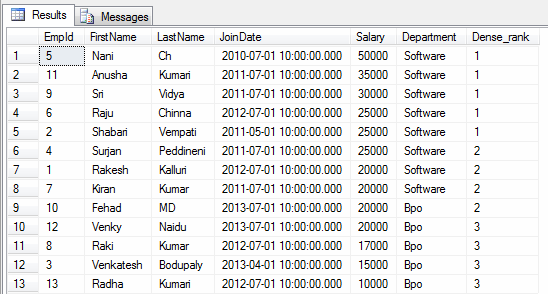
1. --dense\_rank() with out using partition cluase
2. **select** \* ,dense\_rank() over (**order** **by** Salary **desc**) **as** [Dense\_rank] **from** Employee



1. --dense\_rank() with using partition cluase
2. **select** \* ,dense\_rank() over (partition **by** Department **order** **by** Salary **desc**) **as** [Dense\_rank] **from** Employee



1. --ntile(input\_exp) with out using partition cluase
2. **select** \* ,ntile(3) over (**order** **by** Salary **desc**) **as** [ntile] **from** Employee

In Ntile it accepts the input parameter based on input it divides the row ranking.  
  


1. --ntile(input\_exp) with using partition cluase
2. **select** \* ,ntile(3) over (partition **by** De6partment **order** **by** Salary **desc**) **as** [ntile] **from** Employee

