Windowing Functions

*(Also known as Analytical functions)*

Whenever we want to see pure summary outputs like department number wise highest salaries then we use the Group By technique and aggregate function. But the Group By clause can only give you pure summary result. **It is not capable of showing summary values along with actual records.**

What if we want to show the employee name, salary, that employee’s department’s highest salary for each record? Group by fails in that case. In such a scenario Window Ranking, Offset, and Aggregate Functions will help us!!

Though analytic functions like Over() give aggregate result **they do not group the result set.** They return the group value multiple times with each record. As such any other non-"group by" column or expression can be present in the select clause.

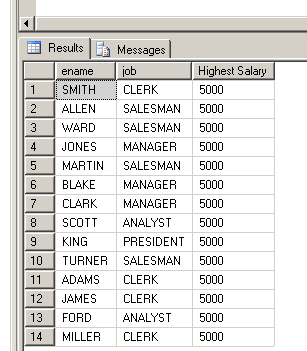
Partition by will do aggregation and display it for all the records (After aggregating the value grouping is not done)

**Over () clause**

If OVER( ) function is NOT having any parameters the function acts on entire record set returned by the where clause.

**select ename, job, max(sal) over () "Highest Salary" from emp**

***(The max(sal) value is repeated for all the rows)***



What Over() Clause does?

## Over() clause creates a specific window, i.e. user defined set of rows provided by the select statement.

## Then after creating the window of those rows, it further computes a value for each row in that result set. Now that computation can be any aggregate function like min, max, sum, running totals, moving averages, etc. or that computation can be generating auto numbering, generating ranks based on data values.

## Finally the computed value is given to each row within that set.

**A scalar function such as IsNull, Case, Cast cannot be given before Over function.**

select ename, job, **IsNull(Comm,0) Over () "Comm"** from emp

**Error Message 🡪 The function 'IsNull' is not a valid windowing function, and cannot be used with the OVER clause.**

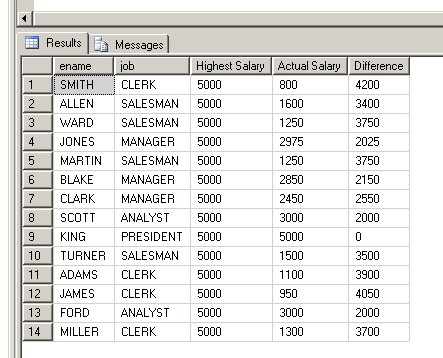
**To perform calculations with aggregate values and actual values**

To see the difference in the max(sal) and sal for each employee.

**select ename, job, max(sal) over () "Highest Salary", sal "Actual Salary",**

**(max(sal) over() – sal) "Difference"**

**from emp**



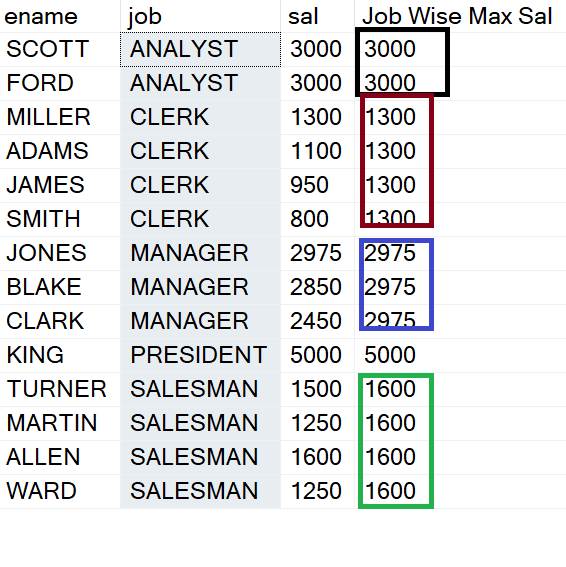
**Over (Partition by) clause**

To see name, job, salary and maximum salary within each job.

**select ename, job, sal,**

**max(sal) over (partition by job) "Job Wise Max Sal"**

**from emp**

****

**Partition By Clause:** The Partition By clause will sort the rows mentioned as per that column (s) and then do the aggregation, ranking as per the distinct values of that column.

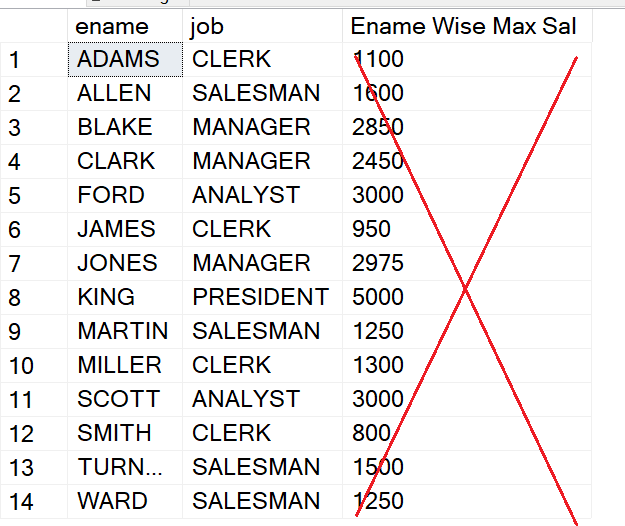
**Example of wrong column in Partition By:**

If a column having **unique values** is given in the Partition by clause then it will result in logically wrong output.

select ename, job,

max(sal) over (partition by ename) "Ename Wise Max Sal"

from emp -- **Logically Wrong**, although no error is given



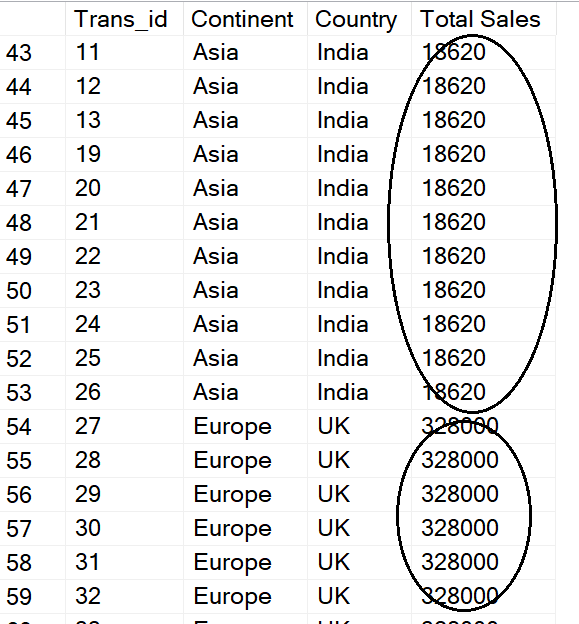
**Multiple Columns in Partition By Clause**

Select Trans\_id, Continent, Country,

Sum(Sales) Over(Partition By **Continent, Country)** as "Total Sales"

from Products2





If the column used in Partition by clause has null values then for all those nulls it will do the sorting and aggregation.

Update emp

set job = null

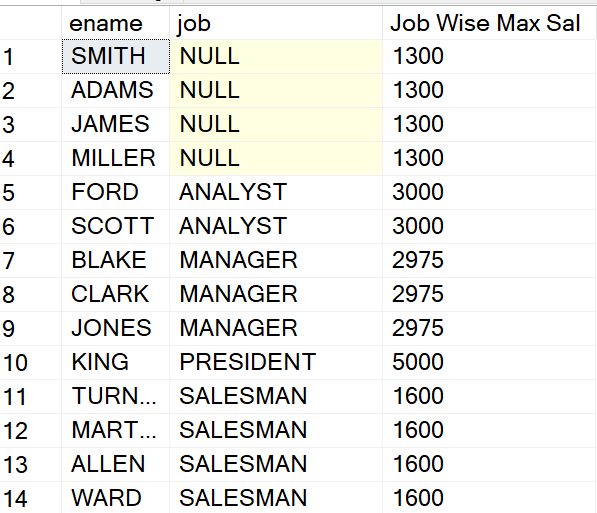
where job = 'CLERK'

select \* from emp

select ename, job,

max(sal) over(partition by job) "Job Wise Max Sal"

from emp



Let’s restore the default values in the emp table.

Update emp

set job = 'CLERK'

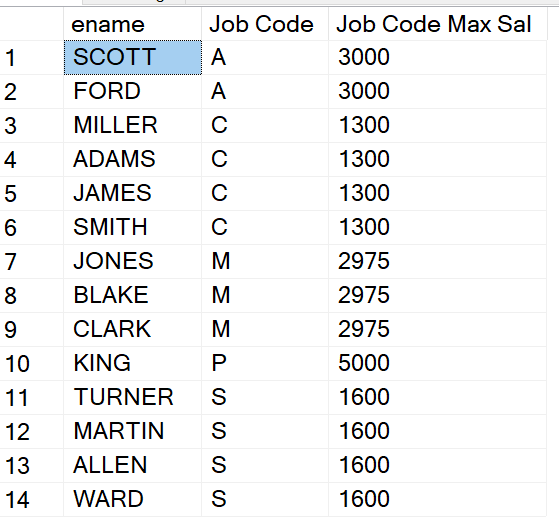
where job is null

**Partitioning on Expression Values:**

select ename, Left(Job,1) as "Job Code",

max(sal) over (partition by **Left(Job, 1)**) "Job Code Max Sal"

from emp

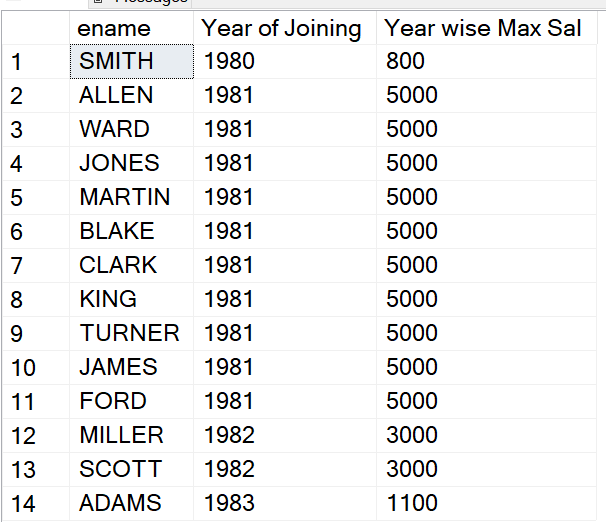


**Especially suitable for date value analysis.**

select ename, Year(Hiredate) as "Year of Joining",

max(sal) over (partition by **Year(Hiredate)**) "Year wise Max Sal"

from emp

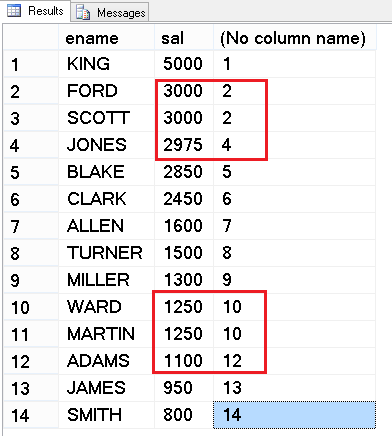


**Ranking Functions**

RANK and DENSE\_RANK both provide rank to the records based on some column value or expression. In case of a tie of 2 records at position N, RANK declares 2 positions N and skips position N+1 and gives position N+2 to the next record. While DENSE\_RANK declares 2 positions N but does not skip position N+1.

**Rank()**

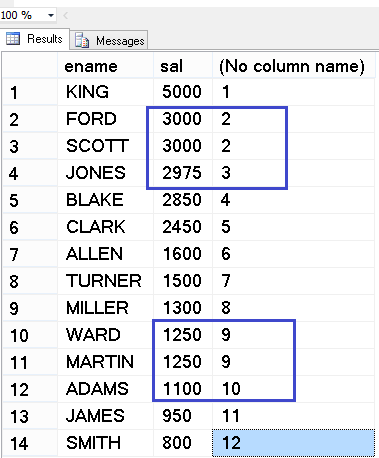
selectename,sal,Rank()over(orderbysaldesc)fromemp

****

**In Rank function in case of ties the next value will get the physical row number and not the logical row number.**

**Dense\_Rank()**

selectename,sal,Dense\_Rank()over(orderbysaldesc)fromemp



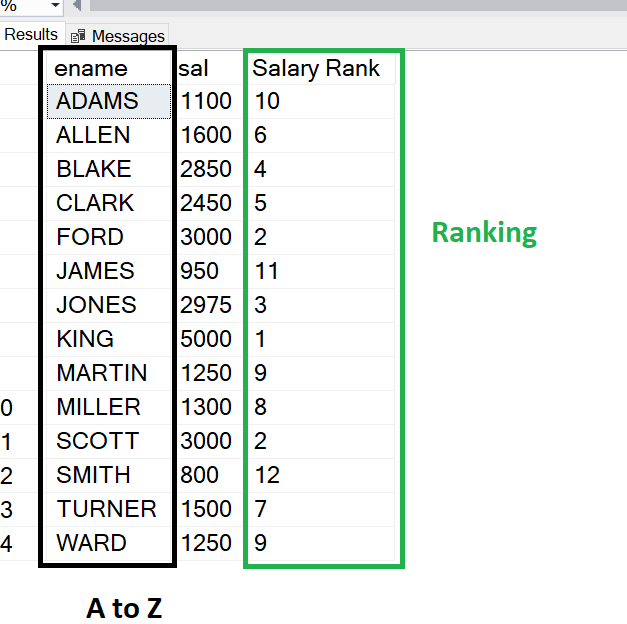
**In Dense\_Rank function in case of ties the next value will get the logical row number and not the physical row number.**

**Example of Order By in Over() function and at the end of Select Statement:**

select ename, sal, Dense\_Rank() over(order by sal desc) as "Salary Rank" -- Order By inside Over() is used for ranking

from emp

Order by ename -- The Order by at the end is for sorting the result ename wise A to Z



**Ranking along with Partition By:**

select ename,

sal,

deptno,

dense\_rank() over(partition by deptno order by sal desc) "Salary Rank as Deptno"

from emp

order by 3,4

ENAME SAL DEPTNO Salary Rank as Deptno

---------- ---------- ---------- -----------------------

KING 5000 10 1

CLARK 2450 10 2

MILLER 1300 10 3

SCOTT 3000 20 1

FORD 3000 20 1

JONES 2975 20 **2**

ADAMS 1100 20 3

SMITH 800 20 4

BLAKE 2850 30 1

ALLEN 1600 30 2

TURNER 1500 30 3

WARD 1250 30 4

MARTIN 1250 30 4

JAMES 950 30 5

For DEPTNO 20 there are two contenders for the first position Scott and Ford. So it has given the same number 1 for those and directly given rank number 2 for Jones of the same deptno 20.

Same case is for last 3 records of deptno 30.

**Row\_Number()**

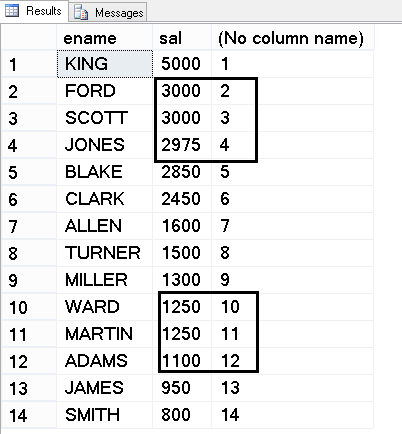
It will provide the row numbers for the result set once the records are sorted.

**Order By is the mandatory clause required for row\_number().**

select ename, row\_number() over(order by ename) from emp



selectename,sal,row\_number()over(orderbysaldesc)fromemp



**Even if there is tie (same values) RowNumber() will always give the next serial number only.**

**So RowNumber() gives us True serial or auto number independent of tied values.**

**As a best practice RowNumber() column should be ideally the first column in Reports.**

select row\_number() over(order by ename) as "RowNumber",

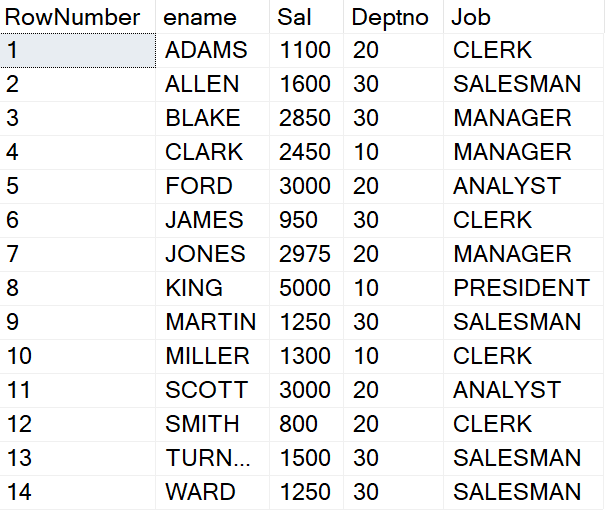
ename,

Sal,

Deptno,

Job

from emp;



**RowNumber() function along with Partition By clause:**

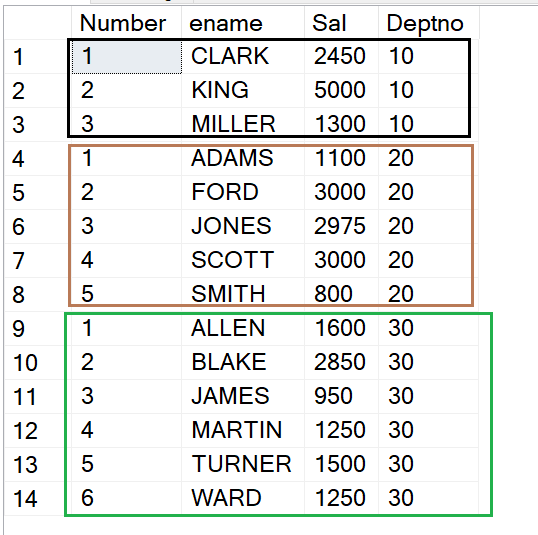
select row\_number() over(**Partition by deptno** order by ename) as "Number in Deptno",

ename,

Sal,

Deptno

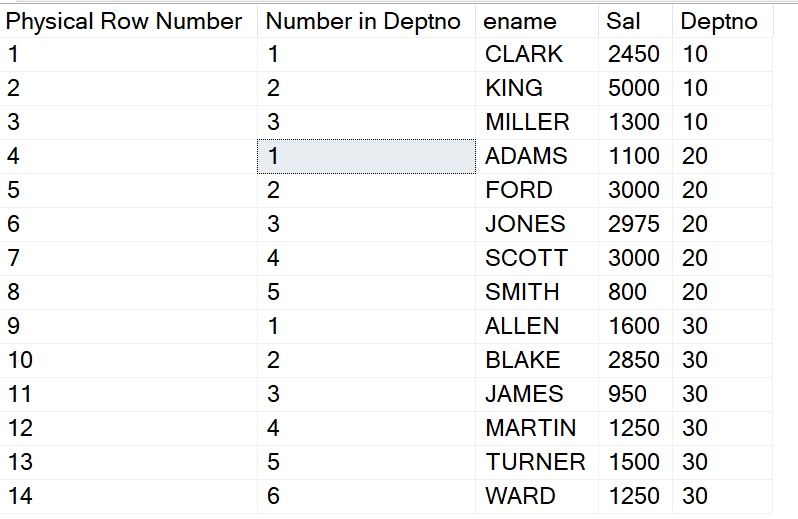
from emp;



In the above example the row numbers are within the current Deptno values.

**Example of Physical Row Number of Record and row number within deptno**

|  |
| --- |
| select row\_number() over(order by deptno) as"Physical Row Number",  row\_number() over(Partition by deptno order by ename) as "Number in Deptno",  ename,  Sal,  Deptno  from emp |



**The Windowed Functions cannot be used in the where clause.**

To get the records of top 3 earners:

select ename, sal,

Dense\_Rank() Over(Order by Sal desc) as Salary\_Rank

from emp

**where Dense\_Rank() Over(Order by Sal desc) > 3**

Error Message 🡪 **Windowed functions can only appear in the SELECT or ORDER BY clauses.**

select ename, row\_number() over(order by ename)

from emp

**where row\_number() over(order by ename) <= 5**

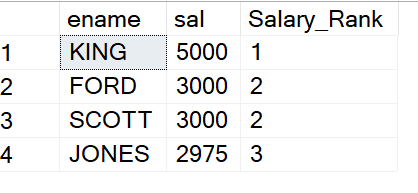
Same Error Message 🡪 **Windowed functions can only appear in the SELECT or ORDER BY clauses.**

**Solution is using a Derived Table!!!**

**Top – N Analysis becomes very easy using the Windowing and Ranking functions.**

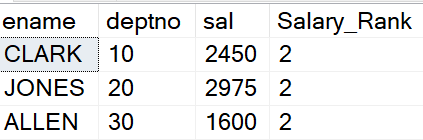
To show records of top 3 salary earners.

|  |
| --- |
| Select \*  From (select ename, sal,  Dense\_Rank() Over(Order by Sal desc) as Salary\_Rank  from emp) as DT  where Salary\_Rank <= 3 |



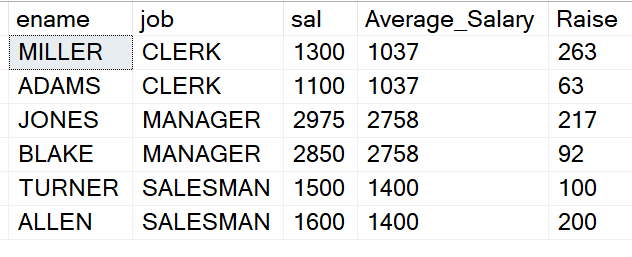
To show the 2nd highest earner record within each department number.

|  |
| --- |
| Select \*  From (select ename, deptno, sal,  Dense\_Rank() Over(**Partition by Deptno** Order by Sal desc) as Salary\_Rank  from emp) as DT  where Salary\_Rank = 2 |



To see names, salaries, job, average salary and difference (raise) with average salary of those employees who earn more than the average salary in their jobs.

|  |
| --- |
| select ename, job, sal, Average\_Salary, Sal - Average\_Salary as "Raise"  From **(select ename, job, sal,**  **Avg(Sal) Over(Partition by job) as Average\_Salary**  **from emp) as DT**  where Sal > Average\_Salary |

****

**View and Window Functions:**

Generally, in the projects the complex select statement used along with Window functions are put inside a View.

User will query the view simply be mentioning select \* from viewname.

create view **Employees\_Above\_Average\_Salary**

as

select ename, job, sal, Average\_Salary, Sal - Average\_Salary as "Raise"

From (select ename, job, sal,

Avg(Sal) Over(Partition by job) as Average\_Salary

from emp) as DT

where Sal > Average\_Salary

**Later we will just query the view:**

select \* from **Employees\_Above\_Average\_Salary**

Further customization in output also becomes easy:

select \* from **Employees\_Above\_Average\_Salary**

where Average\_Salary > 2000