Table of Contents

1. Ranking

A ranking shows a position in a sorted list. Here we are ranking employees by their salary values. In the display below the person with the lowest salary has rank 1. There are several people with salary 15000; they have different ranks.

This uses a variable that is incremented for each row. There can be problems with using user-variables this way- so we will look at another approach also. The general rule is never to assign a value to a user variable in one part of a statement and use the same variable in some other part of the same statement. You might get the results you expect, but this is not guaranteed. This is not a very obvious way to run a query; you should consider this as learning to use a technique. Maybe someday MySQL will get good ranking fucntoins.

Demo 01: This uses a session variable to increment the rank for each row

```
set @rownum:= 0;

select
    emp_id
  , dept_id
  , salary
  , @rownum:= @rownum + 1 as Ranking
from  a_emp.adv_emp
order by salary;
+--------+---------+--------+---------+
| emp_id | dept_id | salary | Ranking |
+--------+---------+--------+---------+
|    150 |      80 |   6500 |       1 |
|    103 |     210 |   9000 |       2 |
|    108 |      30 |  12000 |       3 |
|    161 |     215 |  15000 |       4 |
|    160 |     215 |  15000 |       5 |
|    205 |      30 |  15000 |       6 |
|    109 |      30 |  15000 |       7 |
|    201 |      20 |  15000 |       8 |
|    204 |      30 |  15000 |       9 |
|    100 |      10 |  24000 |      10 |
|    110 |      30 |  30300 |      11 |
|    102 |     215 |  30300 |      12 |
|    203 |      30 |  44450 |      13 |
|    104 |     210 |  50000 |      14 |
|    200 |      35 |  65000 |      15 |
|    207 |      35 |  65000 |      16 |
|    145 |      80 |  65000 |      17 |
|    155 |      80 |  80000 |      18 |
|    146 |     215 |  88954 |      19 |
|    206 |      30 |  88954 |      20 |
|    162 |      35 |  98000 |      21 |
|    101 |      30 |  98005 |      22 |
+--------+---------+--------+---------+
```

Demo 02: Do a sort in a subquery and we can maintain the rownumbers. This generates the Rownumber based on a salary sort but displays the final result in year hired ordered.

```
set @salarynum:= 0;

select *
from (
   select
     emp_id
   , salary
   , year_hired
   , @salarynum:= @salarynum + 1 as RowNumber
   from a_emp.adv_emp
   order by salary
   ) tbl
order by year_hired desc;
+--------+--------+------------+-----------+
| emp_id | salary | year_hired | RowNumber |
+--------+--------+------------+-----------+
|    110 |  30300 |       2012 |        11 |
|    104 |  50000 |       2012 |        14 |
|    109 |  15000 |       2012 |         7 |
|    146 |  88954 |       2012 |        19 |
|    162 |  98000 |       2011 |        21 |
|    161 |  15000 |       2011 |         4 |
|    160 |  15000 |       2011 |         5 |
|    200 |  65000 |       2011 |        15 |
|    207 |  65000 |       2011 |        16 |
|    204 |  15000 |       2011 |         9 |
|    206 |  88954 |       2011 |        20 |
|    103 |   9000 |       2010 |         2 |
|    102 |  30300 |       2010 |        12 |
|    203 |  44450 |       2010 |        13 |
|    101 |  98005 |       2008 |        22 |
|    205 |  15000 |       2008 |         6 |
|    145 |  65000 |       2008 |        17 |
|    201 |  15000 |       2004 |         8 |
|    155 |  80000 |       2004 |        18 |
|    150 |   6500 |       2001 |         1 |
|    108 |  12000 |       1995 |         3 |
|    100 |  24000 |       1989 |        10 |
+--------+--------+------------+-----------+
```

Demo 03: We might want to rank employees within their department. This uses two session variables, We restart the rank for each new dept. Use a case structure to examine the value of the variable. You might recognize this logic as control-break logic. For each change in the dept_id value, the rank starts over as 1

```
set @dept := 0;
set @rank := 0;
select Dept_id, Emp_id, Salary, Rank
from (
   select
       Dept_id, Emp_id, Salary
     , case when @dept = dept_id then @rank := @rank +1
       else @rank :=1
       end as Rank
     , case when @dept <> dept_id then @dept:= dept_id
```

```
        end as  Brk
   from a_emp.adv_emp
   order by dept_id, salary) tbl ;

+---------+--------+--------+------+
| Dept_id | Emp_id | Salary | Rank |
+---------+--------+--------+------+
|      10 |    100 |  24000 |    1 |
|      20 |    201 |  15000 |    1 |
|      30 |    108 |  12000 |    1 |
|      30 |    205 |  15000 |    2 |
|      30 |    204 |  15000 |    3 |
|      30 |    109 |  15000 |    4 |
|      30 |    110 |  30300 |    5 |
|      30 |    203 |  44450 |    6 |
|      30 |    206 |  88954 |    7 |
|      30 |    101 |  98005 |    8 |
|      35 |    200 |  65000 |    1 |
|      35 |    207 |  65000 |    2 |
|      35 |    162 |  98000 |    3 |
|      80 |    150 |   6500 |    1 |
|      80 |    145 |  65000 |    2 |
|      80 |    155 |  80000 |    3 |
|     210 |    103 |   9000 |    1 |
|     210 |    104 |  50000 |    2 |
|     215 |    161 |  15000 |    1 |
|     215 |    160 |  15000 |    2 |
|     215 |    102 |  30300 |    3 |
|     215 |    146 |  88954 |    4 |
+---------+--------+--------+------+
```

2. Various ranking schemes

The next demos do not use the session variables for ranking and they produce somewhat different results.

These just look at dept 30 to keep the row count down. Note the filters for dept_id in the various query components.

2.1. Version A

In this result set, we have several people with salary 15000 and they get rank 5. The next salary gets rank 6. This is called dense ranking since none of the rank numbers are skipped.

Again this uses a correlated subquery and uses one copy of the table to get the first few columns and the second to get the rank column,

Demo 04:

```
select
  emp_1.emp_id
, dept_id
, emp_1.salary
, (
   select count(distinct salary)
   from  a_emp.adv_emp as emp_2
   where emp_2.salary >= emp_1.salary
   and  dept_id = 30
   )as Ranking
from  a_emp.adv_emp as emp_1
```

```
where dept_id = 30
order by ranking
;
+--------+---------+--------+---------+
| emp_id | dept_id | salary | Ranking |
+--------+---------+--------+---------+
|    101 |      30 |  98005 |       1 |
|    206 |      30 |  88954 |       2 |
|    203 |      30 |  44450 |       3 |
|    110 |      30 |  30300 |       4 |
|    109 |      30 |  15000 |       5 |
|    204 |      30 |  15000 |       5 |
|    205 |      30 |  15000 |       5 |
|    108 |      30 |  12000 |       6 |
+--------+---------+--------+---------+
```

2.2. Version B

Demo 05: If you wanted to start the rank at 0 use > instead of >=

```
select
  emp_1.emp_id
, dept_id
, emp_1.salary
, (
   select count(distinct salary)
   from a_emp.adv_emp as emp_2
   where emp_2.salary > emp_1.salary
   and dept_id = 30
  )as ranking
from  a_emp.adv_emp as emp_1
where dept_id = 30
order by ranking;
+--------+---------+--------+---------+
| emp_id | dept_id | salary | ranking |
+--------+---------+--------+---------+
|    101 |      30 |  98005 |       0 |
|    206 |      30 |  88954 |       1 |
|    203 |      30 |  44450 |       2 |
|    110 |      30 |  30300 |       3 |
|    109 |      30 |  15000 |       4 |
|    204 |      30 |  15000 |       4 |
|    205 |      30 |  15000 |       4 |
|    108 |      30 |  12000 |       5 |
+--------+---------+--------+---------+
```

2.3. Version C

There is another way to count the ranks. This demo includes the previous rank column- with the alias Ranking1 and adds a second ranking column. Ranking2 also find ties but skips some of the numbers when there are ties. In the output, there ties for salary 15000 these all get rank 7. Ranks 5 and 6 were skipped. We have 8 rows and the column for Rank 2 goes to rank 8.

Demo 06:

```
select emp_1.emp_id, dept_id, emp_1.salary
, (
   select count(distinct salary)
   from a_emp.adv_emp as emp_2
```

```
      where emp_2.salary >= emp_1.salary
      and dept_id = 30
      )as ranking1
   , (
      select count(salary)
      from    a_emp.adv_emp as emp_2
      where  emp_2.salary >= emp_1.salary
      and    dept_id = 30
     )as ranking2
   from  a_emp.adv_emp as emp_1
   where dept_id = 30
   order by ranking1
   ;
   +--------+---------+--------+----------+----------+
   | emp_id | dept_id | salary | ranking1 | ranking2 |
   +--------+---------+--------+----------+----------+
   |    101 |      30 |  98005 |        1 |        1 |
   |    206 |      30 |  88954 |        2 |        2 |
   |    203 |      30 |  44450 |        3 |        3 |
   |    110 |      30 |  30300 |        4 |        4 |
   |    109 |      30 |  15000 |        5 |        7 |
   |    204 |      30 |  15000 |        5 |        7 |
   |    205 |      30 |  15000 |        5 |        7 |
   |    108 |      30 |  12000 |        6 |        8 |
   +--------+---------+--------+----------+----------+
```

2.4. Version D

In this version we have the tied rows getting the smaller rank number. There are ties for salary 15000 which all get rank 5 and the next rank used is 8

Demo 07:

```
   select
     Emp_1.emp_id
   , dept_id
   , Emp_1.salary
   , (
      select count(salary)
      from a_emp.adv_emp as Emp_2
      where Emp_2.salary > Emp_1.salary
      and dept_id = 30) + 1 as Ranking2
   from a_emp.adv_emp as Emp_1
   where dept_id = 30
   order by ranking2
   ;
   +--------+---------+--------+----------+
   | emp_id | dept_id | salary | Ranking2 |
   +--------+---------+--------+----------+
   |    101 |      30 |  98005 |        1 |
   |    206 |      30 |  88954 |        2 |
   |    203 |      30 |  44450 |        3 |
   |    110 |      30 |  30300 |        4 |
   |    109 |      30 |  15000 |        5 |
   |    204 |      30 |  15000 |        5 |
   |    205 |      30 |  15000 |        5 |
   |    108 |      30 |  12000 |        8 |
   +--------+---------+--------+----------+
```

3. MySQL Approach

This is a very MySQL approach to this which uses some additional MySQL functions. You can read more about this and some of the issues with user variables from the following site

http://rpbouman.blogspot.com/2009/09/mysql-another-ranking-trick.html

These give us rank and dense rank. What essentially happens here is that this uses group_concat to get a csv list of all of the salaries.

Demo 08: This uses group_concat which concatenates all the salaries separated by commas.

```
select
group_concat(salary order by salary desc)  as salarylist
from a_emp.adv_emp \G
*************************** 1. row ***************************
salarylist: 98005,98000,88954,88954,80000,65000,65000,65000,50000,44450,
30300,30300,24000,15000,15000,15000,15000,15000,15000,12000,9000,6500
1 row in set (0.00 sec)
```

Demo 09: Add Distinct to get only one copy of each salary value

```
select
group_concat(distinct salary order by salary desc)  as salarylist
from  a_emp.adv_emp \G
*************************** 1. row ***************************
salarylist: 98005,98000,88954,80000,65000,50000,44450,30300,24000,15000,
12000,9000,6500
1 row in set (0.00 sec)
```

Demo 10: Now use Find_in_set to pick out the position of a salary in that list, giving the rank

```
select
  emp_id
 , salary
 , find_in_set(
       salary
     , (select group_concat( distinct salary order by salary  desc  )
        from  a_emp.adv_emp ) ) as rank
from   a_emp.adv_emp
order by rank
;
+--------+--------+------+
| emp_id | salary | rank |
+--------+--------+------+
|    101 |  98005 |    1 |
|    162 |  98000 |    2 |
|    206 |  88954 |    3 |
|    146 |  88954 |    3 |
|    155 |  80000 |    4 |
|    200 |  65000 |    5 |
|    207 |  65000 |    5 |
|    145 |  65000 |    5 |
|    104 |  50000 |    6 |
|    203 |  44450 |    7 |
|    110 |  30300 |    8 |
|    102 |  30300 |    8 |
|    100 |  24000 |    9 |
|    161 |  15000 |   10 |
|    160 |  15000 |   10 |
|    201 |  15000 |   10 |
|    204 |  15000 |   10 |
|    205 |  15000 |   10 |
|    109 |  15000 |   10 |
|    108 |  12000 |   11 |
|    103 |   9000 |   12 |
|    150 |   6500 |   13 |
+--------+--------+------+
```

Demo 11: What if I skip Distinct? What happens to the Rank column?

```
select
  emp_id
 , salary
 , find_in_set(
       salary
     , (select group_concat(salary order by salary  desc  )
        from  a_emp.adv_emp ) ) as rank
from   a_emp.adv_emp
order by rank
;
+--------+--------+------+
| emp_id | salary | rank |
+--------+--------+------+
|    101 |  98005 |    1 |
|    162 |  98000 |    2 |
|    206 |  88954 |    3 |
|    146 |  88954 |    3 |
|    155 |  80000 |    5 |
|    200 |  65000 |    6 |
|    207 |  65000 |    6 |
|    145 |  65000 |    6 |
|    104 |  50000 |    9 |
|    203 |  44450 |   10 |
|    110 |  30300 |   11 |
|    102 |  30300 |   11 |
|    100 |  24000 |   13 |
|    161 |  15000 |   14 |
|    160 |  15000 |   14 |
|    201 |  15000 |   14 |
|    204 |  15000 |   14 |
|    205 |  15000 |   14 |
|    109 |  15000 |   14 |
|    108 |  12000 |   20 |
|    103 |   9000 |   21 |
|    150 |   6500 |   22 |
+--------+--------+------+
```