Table of Contents

1. Selecting distinct output rows

The keyword DISTINCT can be placed after SELECT to specify that any duplicate copies of an output row will not be displayed. The decision is based on the uniqueness of the rows to be displayed, not the uniqueness of rows in the table.

Using DISTINCT takes extra processing time and should be avoided unless it is necessary to avoid duplicate output lines. If you are writing a query that uses a single table and displays the primary key, do not include DISTINCT.

If you are using DISTINCT, then you can sort only by columns listed in the Select clause.

Demo 01: Display one output row per row in the table

```
select z_type
from zoo_2016;
```

```
Z_TYPE
-------------------------
Giraffe
Armadillo
Lion
Lion
Giraffe
Zebra
Zebra
Zebra
Horse
Giraffe
Giraffe
Giraffe
Giraffe
Giraffe
armadillo
armadillo
Lion
Lion
Lion
Lion

20 rows selected.
```

Demo 02: Display one output row for each different value of z_type. Note that we get one row for 'armadillo' and another row for 'Armadillo', these are considered distinct because Oracle is case specific.

```
select DISTINCT z_type
from zoo_2016;
```

```
Z_TYPE
-------------------------
Zebra
armadillo
Lion
Giraffe
Armadillo
Horse

6 rows selected.
```

Demo 03: Display one output row for each different combination of values for z_type and z_cost .

```
   select DISTINCT z_type, z_cost
   from zoo_2016
   order by z_type, z_cost ;
```

```
Z_TYPE                       Z_COST
------------------------ ----------
Armadillo                       490
Giraffe                        3750
Giraffe                        5000
Giraffe                     5000.25
Horse                           490
Lion                           1850
Lion                           5000
Zebra                       2500.25
armadillo                       490
armadillo                    490.01

10 rows selected.
```

Please note that Distinct is not a function **and it is inappropriate to use parentheses with Distinct**. You will often see queries that use the syntax Select Distinct (z_type) from zoo_2016, but those parentheses are simply parentheses you could use around any expression. You can write a query such as this where the parentheses are also legal but meaningless. Select (z_type) from zoo_2016;

1.1. Sorting and distinct

Return to the query

```
   select DISTINCT z_type from zoo_2016;
```

Can we sort the output? With some dbms, the way a Distinct operation is implemented the output is commonly sorted.

Demo 04: So we will do a descending sort

```
   select distinct z_type
   from zoo_2016
   order by z_type desc;
```

```
Z_TYPE
------------------------
armadillo
Zebra
Lion
Horse
Giraffe
Armadillo
```

What if we want to display the animal type and sort the output by the animal name? Before we try this, think about what this means. We are displaying one row that represents all of the zebra, one row that represents all of the giraffes. If we sort by z_name- how would the rows be returned? If we change the order by clause to sort by the name, we get an error message.

Demo 05:

```
   select distinct z_type
   from zoo_2016
   order by z_name desc;
```

```
order by z_name desc
         *
ERROR at line 3:
ORA-01791: not a SELECTed expression
```

You will often find this to be the case. You try to run a query that seems to make sense at first and find that it is blocked by the system. Most of the time, if you reflect on this, you will see the reasoning behind the decision to disallow the action.

2. Using RowNum

RowNum is a pseudo column that can be used to number and limit the rows as they are returned to SQL*Plus. You can also think of RowNum as a function that returns a row number value. The use of RowNum is limited to Oracle databases.

The tests that work correctly are limited to testing if RowNum is **< value** or **<= value**. Think of RowNum as counting off the rows as they are passed into the result set until the row count reaches the desired number.

RowNum is applied to the rows **before** any Order By or Distinct. This means the results might not be what you had hoped to achieved.

Demo 06: These are the rows from the zoo_2016 table sorted by price with the most expensive first.

```
select z_id, z_name, z_cost
from zoo_2016
order by z_cost desc;
```

```
      Z_ID Z_NAME                       Z_COST
---------- ------------------------- ----------
        85 Sally Robinson              5000.25
        56 Leon                           5000
        57 Lenora                         5000
       259                                5000
       258                                5000
        23 Sam                            5000
       260 Geoff                          5000
       257 Arnold                         5000
        52 Dewey                          3750
        45 Louie                       2500.25
        44 Dewey                       2500.25
        43 Huey                        2500.25
       374                                1850
       375                                1850
       373                                1850
       372 Leon                           1850
       371 Anne                         490.01
       370 Anders                          490
        25 Abigail                         490
        47                                 490

20 rows selected.
```

Demo 07: Now we try to get the five most expensive animals using RowNum

```
select z_id, z_name, z_cost
from zoo_2016
where ROWNUM <=5
order by z_cost desc;
```

```
      Z_ID Z_NAME                       Z_COST
---------- ------------------------- ----------
        85 Sally Robinson              5000.25
        23 Sam                            5000
        56 Leon                           5000
        57 Lenora                         5000
        25 Abigail                         490

5 rows selected.
```

It runs but these are not the most expensive animals- we have many animals at $5000. That is not what we wanted. What RowNum does is take the first 5 rows that come from the zoo_2016 table and are passed to the result set and then it sorts those 5 rows. We will eventually solve the problem of the five most expensive items. What you need to remember at this time is that **Rownum is a Where clause test and that is processed before the Order By clause.**

A few more examples might help.

Demo 08: First show the different animal types in the zoo_2016 table sorted. I got 6 rows

```
select DISTINCT z_type
from zoo_2016
order by z_type;
```

```
Z_TYPE
-------------------------
Armadillo
Giraffe
Horse
Lion
Zebra
armadillo

6 rows selected.
```

Demo 09: Then add a where rownum <= 2 filter; the output looks ok. I got two rows.

```
select DISTINCT z_type
from zoo_2016
where rownum <=2
order by z_type;
```

```
Z_TYPE
-------------------------
Armadillo
Giraffe
```

Demo 10: But If I change this to rownum < =8, I got only four rows. Why didn't I get the Horse or armadillo?

```
select DISTINCT z_type
from zoo_2016
where rownum <=8
order by z_type;
```

```
Z_TYPE
-------------------------
Armadillo
Giraffe
Lion
Zebra

4 rows selected.
```

With `Where rownum<=8` why do I get only four rows? The From clause and the Where clause were executed first and 8 rows were returned, but then Distinct was applied to those 8 rows and apparently there were only 4 distinct z_type values in those rows. (you might get a different set of rows.)

Demo 11: Including RowNum in the select may help you see what is happening. I removed the DISTINCT. We get 8 rows coming from the table in whatever order the dbsm delivers them. After the 8 rows are retrieved, the Order By clause is applied- here the sort is by the price. We do get rownum values 1, 2, 3, 4, 5,6,7, and 8.

```
select ROWNUM, z_id, z_cost, z_type
from zoo_2016
where ROWNUM <=8
order by z_cost desc;
```

```
    ROWNUM       Z_ID     Z_COST Z_TYPE
---------- ---------- ---------- ----------
         5         85    5000.25 Giraffe
         3         56       5000 Lion
         1         23       5000 Giraffe
         4         57       5000 Lion
         6         43    2500.25 Zebra
         7         44    2500.25 Zebra
         8         45    2500.25 Zebra
         2         25        490 Armadillo

8 rows selected.
```

If I now manually looked at the rows and got the distinct values for z_type from that display, I get Giraffe, Lion, Zebra, Armadillo

Some other tests might help you see how this works. All of these return 0 rows. This is due to the way that Oracle generates the rownum values as rows are pulled from the table.

Demo 12:

```
select ROWNUM, z_id, z_name, z_cost
from zoo_2016
where rownum = 5;
```

Demo 13:

```
select ROWNUM, z_id, z_name, z_cost
from zoo_2016
where rownum between 5 and 10;
```

Demo 14:

```
select ROWNUM, z_id, z_name, z_cost
from zoo_2016
where rownum >= 5;
```

The tests that work as we probably want are limited to testing if RowNum is **equal to or less than a value**. RowNum is not the same as a Top or Limit keyword that you might be familiar with from another dbms.

One common use of RowNum is to limit the output to a few rows to see what type of data we have in a large table.