



100 DAYS OF SQL

Day 6 – SQL query for finding the nth highest value of a column in the table

By Mallela Preethi

Question: Consider a zoo's AnimalPopulation table, which records the population of various animal species. The table schema includes Animal_ID, Species, and Population columns. Given the sample data provided below, write SQL queries to:

- 1. Find the species with the second-highest population among all animal species.
- 2. Find the species with the fourth highest population among all animal species.

Show Species and population in result

| Animal_ID | Species | Population |
|-----------|--------------|------------|
| 1 | Lion | 10 |
| 2 | Elephant | 5 |
| 3 | Giraffe | 8 |
| 4 | Tiger | 12 |
| 5 | Penguin | 20 |
| 6 | Gorilla | 7 |
| 7 | Zebra | 15 |
| 8 | Kangaroo | 9 |
| 9 | Crocodile | 6 |
| 10 | Polar Bear | 4 |
| 11 | Leopard | 11 |
| 12 | Hippopotamus | 14 |
| 13 | Koala | 3 |
| 14 | Rhino | 16 |
| 15 | Panda | 13 |

Creation of table and Insertion of records

```
1 • use _100days_sql;
2 • create table if not exists AnimalPopulation(Animal_ID int, Species varchar(30), Population int);
3 • insert into AnimalPopulation(Animal_ID, Species, Population) values(1, "Lion", 10), (2, "Elephant", 5), (3, "Giraffe", 8),
4   (4, "Tiger", 12), (5, "Penguin", 20), (6, "Gorilla", 7), (7, "Zebra", 15), (8, "Kangaroo", 9), (9, "Crocodile", 6),
5   (10, "Polar Bear", 4), (11, "Leopard", 11), (12, "Hippopotamus", 14), (13, "Koala", 3), (14, "Rhino", 16),
6   (15, "Panda", 13);
7
```

1) Second highest (First way – Correlated Subquery)

```
SELECT
    Species, Population
FROM
    AnimalPopulation AS a
WHERE
    2 = (SELECT
        COUNT(DISTINCT Population)
        FROM
            AnimalPopulation AS e
        WHERE
            a.Population <= e.Population);
```

Output:

| Result Grid | | | Filter Rows: |
|-------------|---------|------------|--------------|
| | Species | Population | |
| ▶ | Rhino | 16 | |

1) Second highest (Second way – Using DENSE_RANK)

```
24
25 • SELECT Species, Population FROM
26     (SELECT *, dense_rank() OVER (ORDER BY Population DESC) rn FROM AnimalPopulation ) as a
27 WHERE rn = 2;
```

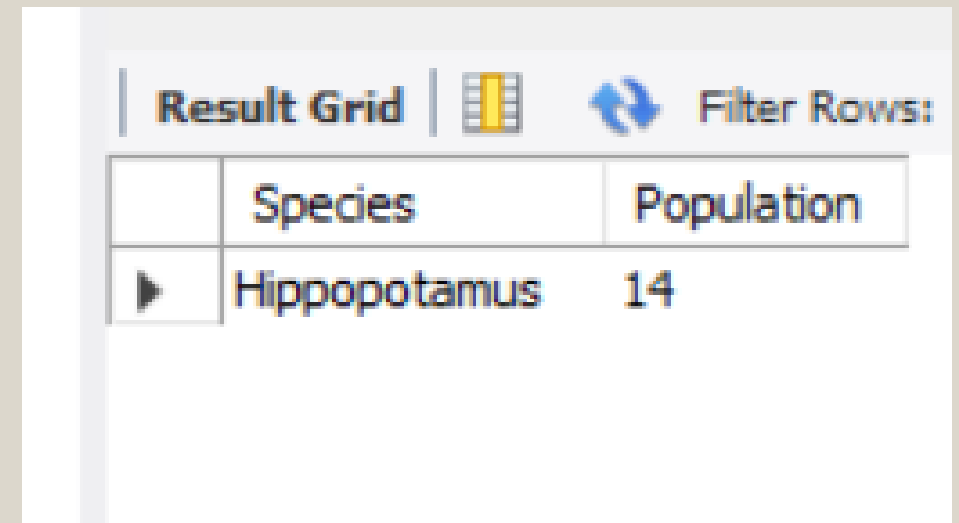
Output:

| Result Grid | | | Filter Rows: |
|-------------|---------|------------|--------------|
| | Species | Population | |
| ▶ | Rhino | 16 | |

1) Fourth highest (First way – Correlated Subquery)

Output:

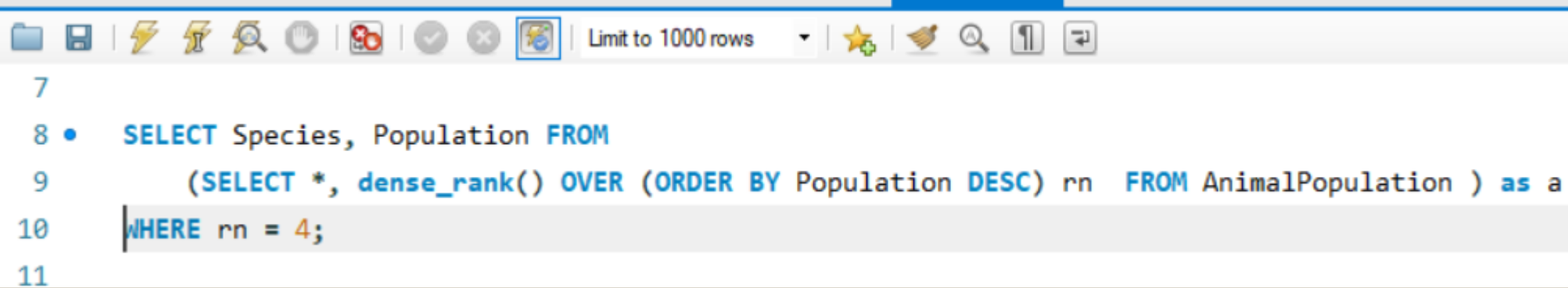
```
12 • SELECT
13     Species, Population
14 FROM
15     AnimalPopulation AS a
16 WHERE
17     4 = (SELECT
18         COUNT(DISTINCT Population)
19     FROM
20         AnimalPopulation AS e
21     WHERE
22         a.Population <= e.Population);
23
```



The screenshot shows a database interface with a 'Result Grid' tab. The grid contains two columns: 'Species' and 'Population'. The first row of data shows 'Hippopotamus' with a population of '14'. There are icons for 'Filter Rows' and a 'Result Grid' tab.

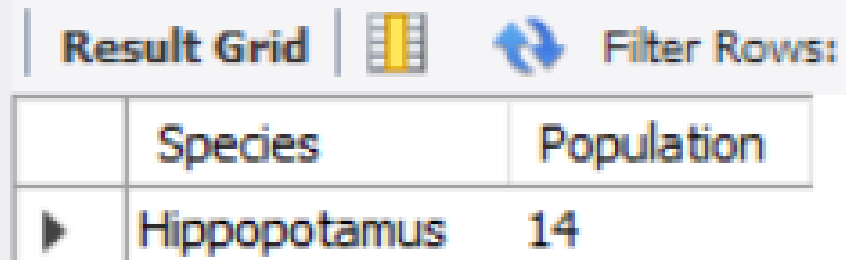
| | Species | Population |
|---|--------------|------------|
| ▶ | Hippopotamus | 14 |

1) Fourth highest (Second way – Using DENSE_RANK)



```
7
8 • SELECT Species, Population FROM
9     (SELECT *, dense_rank() OVER (ORDER BY Population DESC) rn FROM AnimalPopulation ) as a
10 WHERE rn = 4;
11
```

Output:



| | Species | Population |
|---|--------------|------------|
| ▶ | Hippopotamus | 14 |

For nth highest Queries can be written as follows:

- 1)

```
SELECT Species, Population FROM
(SELECT *, dense_rank() OVER (ORDER BY Population DESC) rn
FROM AnimalPopulation ) a WHERE rn = n;
```
- 2)

```
SELECT Species, Population FROM AnimalPopulation a
WHERE n = (SELECT COUNT(DISTINCT Population) FROM AnimalPopulation e
WHERE a.Population <= e.Population);
```