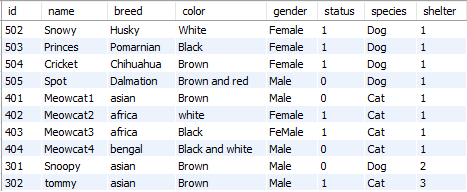
**VAISHALI BOKADIYA**

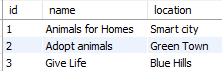
**DAY 5 ASSESSMENT**

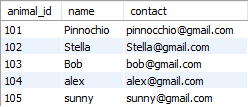
**DATA CLEANING & TRANSFORMATION,**

**STORED PROCEDURES AND RANKINGS**

**DATABASE:**







**RANKINGS:**

Rank functions assign the number (rank) to each row within the partition of an output.

**ROW\_NUMBER():**

This function is used to return the unique sequential number for each row within its partition.

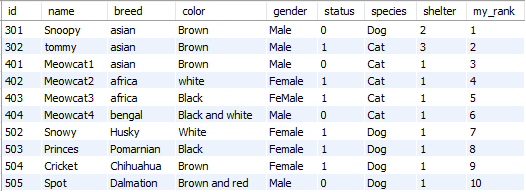
EXAMPLE:

select \*,

row\_number() over(order by id) as my\_rank

from animals;

OUTPUT:



**DENSE RANK():**

This function assigns a unique rank for each row within a **partition** as per the specified column value without any gaps.

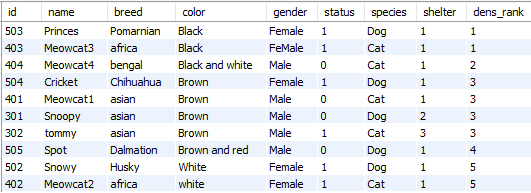
EXAMPLE:

select \*,

dense\_rank() over (order by color) as dens\_rank

from animals;

OUTPUT:



**RANK():**

This function is used to determine the rank for each row in the result set.

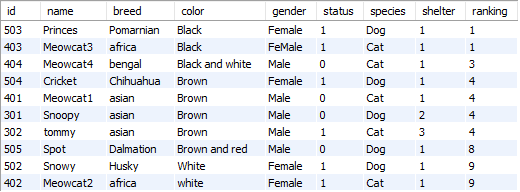
EXAMPLE:

select \*,

rank() over (order by color) as ranking

from animals;

OUTPUT:



**NTILE()**

This function is used to distribute rows of an ordered partition into a pre-defined number (N) of approximately equal groups. Each row group gets its rank based on the defined condition and starts numbering from one group.

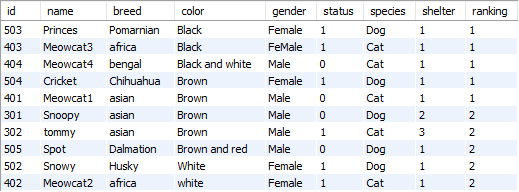
EXAMPLE:

select \*,

ntile(2) over(order by color) as ranking

from animals;

OUTPUT:



**STORED PROCEDURE:**

Stored procedures help group one or multiple SQL statements for reuse under a common name, encapsulating common business logic within the database itself. Such a procedure can be called from the application that accesses the database to retrieve or manipulate data in a consistent way.

EXAMPLE:

delimiter &&

create procedure select\_female\_animals ()

begin

select name from animals where gender='Female';

end &&

call select\_female\_animals();

OUTPUT:



**CORELATED SUBQUERIES:**

A correlated subquery is a subquery that contains a reference to a table that also appears in the outer query.

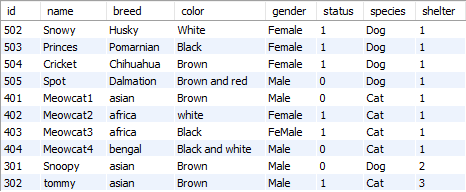
EXAMPLE:

SELECT \* FROM animals

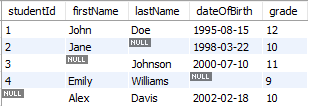
WHERE shelter = ANY (SELECT id FROM shelter

WHERE shelter.id = animals.shelter);

OUTPUT:



**DATABASE:**



**DATA CLEANING AND TRANSFORMATION:**

**STEP-1**

**Deleting the duplicate data:**

select firstName,count(studentId) as Actual\_count from students

group by studentId

having count(studentId)>1;

WITH duplicateStudents AS (

SELECT student\_id

FROM students

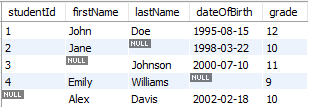
GROUP BY student\_id

HAVING COUNT(student\_id) > 1

)

delete from DuplicateStudents

where count(studentId)>1;

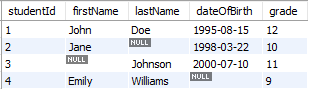


**STEP-2**

**Removing the null values:**

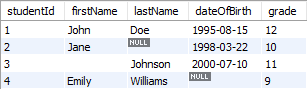
delete from students

where studentId is null;



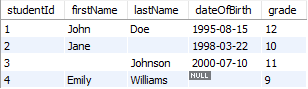
update students set firstName=''

where firstName is null;



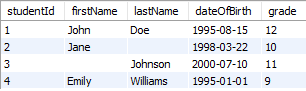
update students set lastName=''

where lastName is null;



update students set dateOfBirth='1995-01-01'

where dateOfBirth is null;



**STEP-3**

**Apply transformations:**

update students

set firstName=upper(firstName);

