```
-- PART 1 - SCALING NUMERICAL DATA
-- ================================================

-- Connect to the "transformaciones_db" database
-- psql -U postgres -d transformaciones_db

-- Initial scan of the "escalamiento" table
SELECT * FROM escalamiento;

-- Scaling by minimum and maximum
-- 1. Calculate the maximum and minimum of each column (max, min)
-- 2. Take each value in each column and apply the operation result =
(valor - min)/(max - min)

-- Example 1: Calculate the maximum and minimum of each column (max, min)
-- We will use what is known as a "Common Table Expression" (CTE), which is
-- like the equivalent of a function in other programming languages
-- and allows us to calculate and return the values we later need
-- in the query. In this case, we will use a CTE to calculate the
-- maximum and minimum of each column
WITH min_max AS (
      SELECT
            MIN(feat_a) AS min_a, MAX(feat_a) AS max_a,
            MIN(feat_b) AS min_b, MAX(feat_b) AS max_b
      FROM escalamiento
);
-- In the syntax above:
-- "min_max" is the alias we'll give to this CTE
-- A CTE always starts with WITH and the operations to be performed are
-- always in parentheses
```

```
-- Let's see what this CTE generates
WITH min_max AS (
      SELECT
            MIN(feat_a) AS min_a, MAX(feat_a) AS max_a,
            MIN(feat_b) AS min_b, MAX(feat_b) AS max_b
      FROM escalamiento
)
SELECT * FROM min_max;

-- Now let's see how to use this CTE to perform min/max scaling of
-- each column of data
-- Define the CTE
WITH min_max AS (
      SELECT
            MIN(feat_a) AS min_a, MAX(feat_a) AS max_a,
            MIN(feat_b) AS min_b, MAX(feat_b) AS max_b
      FROM escalamiento
)
-- And execute the query using this CTE
SELECT id, feat_a, feat_b,
      (feat_a - min_a) / (max_a - min_a) AS feat_a_escalada,
      (feat_b - min_b) / (max_b - min_b) AS feat_b_escalada
FROM
      escalamiento, min_max;

-- Example 2: Sometimes we are interested in scaling data not in the range
-- from 0 to 1 but from, for example, -1 to 1.
-- In this case, only the scaling formula changes:
-- 2*((min_value)/(max_min)) - 1
```

```
WITH min_max AS (
      SELECT
            MIN(feat_a) AS min_a, MAX(feat_a) AS max_a,
            MIN(feat_b) AS min_b, MAX(feat_b) AS max_b
      FROM escalamiento
)
SELECT id, feat_a, feat_b,
      2 * ((feat_a - min_a) / (max_a - min_a)) - 1 AS feat_a_escalada,
      2 * ((feat_b - min_b) / (max_b - min_b)) - 1 AS feat_b_escalada
FROM
escalamiento, min_max;


-- Example 3: Standardization (z-score)
-- 1. Calculate the mean (mu) and standard deviation (sigma) for each
column
-- 2. Take each value in each column and apply the operation result =
(value - mu)/(sigma)


WITH mu_sigma AS (
      SELECT
            AVG(feat_a) AS mu_a, STDDEV(feat_a) AS s_a,
            AVG(feat_b) AS mu_b, STDDEV(feat_b) AS s_b
      FROM escalamiento
)
SELECT id, feat_a, feat_b,
      (feat_a - mu_a)/s_a AS feat_a_estandarizada,
      (feat_b - mu_b)/s_b AS feat_b_estandarizada
FROM
escalamiento, mu_sigma;
```

```
-- PART 2 - TEXT NORMALIZATION (STRINGS)
-- ================================================

-- Display the "strings" table
SELECT * FROM strings;

-- Example 1: Let's use the TRIM() function to remove leading
-- and trailing whitespace
SELECT id, string, TRIM(string) AS trimmed
FROM strings;

-- Example 2: Let's combine LOWER() with TRIM() to remove leading
-- and trailing whitespace and to convert all characters to lowercase
SELECT id, string,
      LOWER(TRIM(string)) AS lower_trimmed
FROM strings;

-- Example 3: Use SPLIT_PART() to, for example, subdivide a string.
SELECT id, string,
      SPLIT_PART(TRIM(string), ' ', 1) AS word_1,
      SPLIT_PART(TRIM(string), ' ', 2) AS word_2
FROM strings;

-- PART 3 - TEXT NORMALIZATION (STRINGS) USING
-- REGULAR EXPRESSIONS
-- ================================================

-- Example 1: Remove all digits within a string
SELECT string,
      REGEXP_REPLACE(TRIM(string), '\d', '', 'g') AS sin_numeros
FROM strings;
```

```
-- 'd': searches for digits between 0-9

-- '': replaces digits with an empty string

-- 'g': replaces all occurrences (if not used, only the first occurrence is
replaced)


-- Example 2: remove non-alphanumeric characters (ñ, Ñ, tildes, @, and

-- other symbols)

SELECT string,

      REGEXP_REPLACE(TRIM(string), '[^a-zA-Z0-9\s]', '', 'g') AS
      solo_alfanumericos

FROM strings;

-- '[^a-zA-Z0-9\s]': search for any character that is not a letter, digit,
-- or space

-- '': remove those characters

-- 'g': to replace all occurrences


-- Example 3: replace multiple spaces with a single space

SELECT string,

      REGEXP_REPLACE(TRIM(string), '\s+', ' ', 'g') AS un_solo_espacio

FROM strings;


-- Example 4: Replace a specific word with another

SELECT string,

      REGEXP_REPLACE(string, 'PostgresSQL', 'postgres', 'g') AS
      palabra_reemplazada

FROM strings;
```