```sql
-- PART 1: TIME SERIES EXPLORATION AND SCALING
-- =====================================================

-- We'll use the "series" table (time series with 1 feature)
-- and assume we want to train a model that takes
-- 3 consecutive records and predicts record number
-- 4. The idea is to use SQL to prepare this dataset.

-- Step 0: Handle missing data, extreme values,
-- and repeated records. In this case, the table is "ideal."

-- Step 1: Initial time series exploration
SELECT * FROM serie;

-- The series contains 16 records and the columns "datetime" and
-- "feature_1"
-- The series values range from 10.1 to 11.6

-- Step 2: Scaling the data and storing the minimum and maximum values
-- calculated during scaling

-- Create and store table "escalamiento_1f"

CREATE TABLE escalamiento_1f AS
WITH escalamiento AS (
SELECT
MIN(feature_1) AS min_f1, MAX(feature_1) AS max_f1
FROM serie
)
SELECT * FROM escalamiento;
```

```
SELECT * FROM escalamiento_1f;

-- Create table "serie_esc" and store the result
-- of scaling the original table ("serie_1f")

-- Create table
CREATE TABLE serie_esc AS SELECT * FROM serie;

-- Scale and update the previous table
WITH escalation AS (
SELECT
MIN(feature_1) AS min_f1, MAX(feature_1) as max_f1
FROM series
)
UPDATE esc_series
SET
feature_1 = 2*(feature_1-min_f1)/(max_f1-min_f1)-1
FROM escalation;

SELECT * FROM esc_series;

-- PART 2: CREATING THE DATASET FOR THE MACHINE LEARNING MODEL
-- ======================================================

SELECT
-- Generate lags of 3, 2, and 1
LAG(st.feature_1, 3) OVER (ORDER BY st.datetime) AS f1_l3,
LAG(st.feature_1, 2) OVER (ORDER BY st.datetime) AS f1_l2,
LAG(st.feature_1, 1) OVER (ORDER BY st.datetime) AS f1_l1,

-- And the current value will be the value to be predicted ("target")
```

```
st.feature_1 AS target
FROM serie_esc st;

-- In the previous case, the first three rows have incomplete data.
-- This is because for time points 1, 2, and 3, we won't have
-- yet 3 historical data points to generate this new prediction.

-- Let's modify the previous query so that the
-- incomplete rows
SELECT *
FROM (
SELECT
-- Generate "lags" of 3, 2 and 1
LAG(st.feature_1, 3) OVER (ORDER BY st.datetime) AS f1_l3,
LAG(st.feature_1, 2) OVER (ORDER BY st.datetime) AS f1_l2,
LAG(st.feature_1, 1) OVER (ORDER BY st.datetime) AS f1_l1,

-- And the current value will be the value to be predicted ("target")
st.feature_1 AS target

FROM esc_series st
)
WHERE f1_l3 IS NOT NULL;

SELECT * FROM esc_series;

-- And finally let's save the new table generated with the previous query
CREATE TABLE dataset_forecasts AS
SELECT *
FROM (
SELECT
```

```sql
-- Generate "lags" of 3, 2 and 1
LAG(st.feature_1, 3) OVER (ORDER BY st.datetime) AS f1_l3,
LAG(st.feature_1, 2) OVER (ORDER BY st.datetime) AS f1_l2,
LAG(st.feature_1, 1) OVER (ORDER BY st.datetime) AS f1_l1,

-- And the current value will be the value to be predicted ("target")
st.feature_1 AS target

FROM esc_series st
)
WHERE f1_l3 IS NOT NULL;

SELECT * FROM dataset_forecasts;

-- Limitations of the previous approach:
-- - Generally, when building an ML model, we don't know in advance
--   the most appropriate number of input/output time instants
--   to generate predictions. The previous code is static: if we want
--   to change these input and output sizes, we'll have to modify the code.
-- - For relatively large input/output sizes, we'll have relatively
     extensive code.
```