```
-- PART 1 - DETECTING EXTREME VALUES IN NUMERIC VARIABLES
-- ======================================================

-- First, we connect to the "limpieza_db" database.
psql -U postgres -d limpieza_db

-- Detecting extreme values in numeric variables can be done
-- using Tukey's method or the z-score method
-- Tukey's method is recommended because it is less sensitive to extreme
-- values.

-- Example 1: Detection with Tukey's method

-- Tukey's method requires:
-- 1. Calculate the 25th percentile (quartile 1, Q1)
-- 2. Calculate the 75th percentile (3rd quartile, Q3)
-- 3. Calculate the interquartile range (IQR = Q3-Q1)
-- 4. Mark values below Q1 - 1.5*IQR as “outliers”
--    or above Q1 + 1.5*IQR
-- Let's start by implementing a Common Table Expression to
-- calculate the 1st and 2nd quartiles of the numeric variable
WITH q1_q3_edad AS (
      SELECT
            -- 25th percentile. "WITH GROUP" is used to generate
            -- aggregations on the ordered numeric variable
            PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY edad) AS q1_edad,
            PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY edad) AS q3_edad
      FROM inversionistas
)
-- And let's see the result of this CTE.
SELECT * FROM q1_q3_edad;
```

```
-- And let's create a CTE similar to the previous one, but for the variable
-- "amount"
WITH q1_q3_monto AS (
      SELECT
            -- 25th percentile. "WITHIN GROUP" is used to generate
            -- aggregations on the ordered numeric variable
      PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY monto) AS q1_monto,
      PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY monto) AS q3_monto
FROM inversionistas
)
-- And let's see the result of this CTE.
SELECT * FROM q1_q3_monto;


-- Very well. Now the idea is to use these two CTEs and create a query
-- where:
-- 1. Select id, age, and amount
-- 2. For each variable, use "CASE" to determine if the variable is below
-- Q1 - 1.5*IQR or above Q3 + 1.5*IQR, and if so, label it as
-- "NULL". Otherwise, we leave the value as is.


-- Let's combine the 2 CTEs created previously into a single one.
WITH q1_q3 AS (
      SELECT
            -- Quartiles 1 and 3, age variable
      PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY edad) AS q1_edad,
      PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY edad) AS q3_edad,
            -- Quartiles 1 and 3, amount variable
      PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY monto) AS q1_monto,
      PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY monto) AS q3_monto
      FROM inversionistas
)
```

```
-- Query to detect outliers by age
SELECT id, nombre, edad,
-- Query to detect outliers by age
CASE
      WHEN edad < q1_edad - 1.5*(q3_edad-q1_edad) OR edad > q3_edad +
      1.5*(q3_edad-q1_edad)
      THEN 10000
      ELSE edad
END AS edad_outliers,
monto,
-- Query to detect outliers by age
-- Query to detect outliers by amount
CASE
      WHEN monto < q1_monto - 1.5*(q3_monto-q1_monto) OR monto > q3_monto +
      1.5*(q3_monto-q1_monto)
      THEN 10000
      ELSE monto
      END AS monto_outliers
FROM inversionistas, q1_q3;

-- PART 2 - HANDLING EXTREME VALUES IN NUMERIC VARIABLES
-- ====================================================

-- The simplest way to handle extreme values is
-- 1. Perform detection
-- 2. Mark extreme values as "missing"
-- 3. Apply one of the techniques like elimination or imputation

-- Let's slightly modify the previous query to:
-- - Return only id, name, age, amount
-- - And return only those rows with complete records (IS NOT NULL)
--   and that are NOT outliers
```

```
WITH q1_q3 AS (
SELECT
      -- Quartiles 1 and 3 variable age
      PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY edad) AS q1_edad,
      PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY edad) AS q3_edad,
      -- Quartiles 1 and 3, variable amount
      PERCENTILE_CONT(0.25) WITHIN GROUP (ORDER BY monto) AS q1_monto,
      PERCENTILE_CONT(0.75) WITHIN GROUP (ORDER BY monto) AS q3_monto
FROM inversionistas
)
-- Query to return only values that are not outliers
SELECT id, nombre, edad, monto
FROM investors, q1_q3
WHERE
      -- Remove missing data
      edad IS NOT NULL AND monto IS NOT NULL
      -- Filter outliers by age
      AND edad >= q1_edad - 1.5*(q3_edad - q1_edad)
      AND edad <= q3_edad + 1.5*(q3_edad - q1_edad)
-- Filter outliers by amount
AND monto >= q1_monto - 1.5*(q3_monto - q1_monto)
AND monto <= q3_monto + 1.5*(q3_monto - q1_monto);

-- And for comparison, let's show the original table
SELECT id, nombre, edad, monto FROM inversionistas;



-- PART 3 - DETECTING AND HANDLING EXTREME VALUES IN CATEGORICAL VARIABLES
-- ======================================================
```

```sql
-- In categorical variables, the "outlier" can sometimes
-- refer to the minority category (although this is not always true)

-- Let's perform the detection assuming the "outlier" is precisely the
-- minority category

-- CTE to determine the least frequent category
WITH menos_frec AS (
      SELECT categoria
      FROM inversionistas
      WHERE categoria IS NOT NULL
      GROUP BY categoria
      ORDER BY COUNT(*)
      LIMIT 1
)
SELECT * FROM menos_frec;

-- Now let's use the previous CTE to mark the outliers.
WITH menos_frec AS (
      SELECT categoria
      FROM inversionistas
      WHERE categoria IS NOT NULL
      GROUP BY categoria
      ORDER BY COUNT(*) ASC
      LIMIT 1
)
SELECT id, nombre, categoria,
      CASE
            WHEN categoria = (SELECT categoria FROM menos_frec) THEN
            'Outlier' ELSE categoria
```

```
      END AS categoria_outlier
FROM inversionistas;


-- And to perform the elimination process, we can make a slight
modification to the previous query.
-- CTE
WITH menos_frec AS (
      SELECT category
      FROM inversionistas
      WHERE categoria IS NOT NULL
      GROUP BY categoria
      ORDER BY COUNT(*) ASC
      LIMIT 1
)
SELECT id, nombre, categoria
FROM inversionistas
WHERE
      -- Filtering missing values
      categoria IS NOT NULL AND
      -- Filtering extreme categorical values
      categoria != (SELECT categoria FROM menos_frec);


-- And display the original table (unique id, name, and category)
SELECT id, nombre, categoria
FROM inversionistas;
```