CREATE TABLE umbrales (

umbral\_id INT AUTO\_INCREMENT PRIMARY KEY,

sensor\_id INT NOT NULL,

estacion\_id INT NOT NULL,

umbral\_minimo FLOAT NULL,

umbral\_maximo FLOAT NULL,

tiene\_umbrales BOOLEAN DEFAULT TRUE,

observacion TEXT NULL,

creado\_en TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

enable INT DEFAULT 1,

FOREIGN KEY (sensor\_id) REFERENCES Sensores(sensor\_id),

FOREIGN KEY (estacion\_id) REFERENCES Estaciones(estacion\_id)

);

INSERT INTO `gp-mlp-telemtry`.umbrales (

sensor\_id, estacion\_id, umbral\_minimo, umbral\_maximo, tiene\_umbrales, observacion, creado\_en, enable

)

SELECT

s.sensor\_id,

s.estacion\_id,

5.5 AS umbral\_minimo,

9.0 AS umbral\_maximo,

TRUE AS tiene\_umbrales,

'Umbral asignado por tipo pH' AS observacion,

NOW() AS creado\_en,

TRUE AS enable

FROM `gp-mlp-telemtry`.sensores s

WHERE s.tipo = 'pH';

CREATE TABLE criterio\_alerta (

criterio\_id INT AUTO\_INCREMENT PRIMARY KEY,

nombre VARCHAR(100) NOT NULL

);

INSERT INTO criterio\_alerta (nombre) VALUES

('detencion'),

('umbral'),

('anomalia');

CREATE TABLE Alertas (

alerta\_id INT AUTO\_INCREMENT PRIMARY KEY,

sensor\_id INT NOT NULL,

estacion\_id INT NOT NULL,

timestamp DATETIME NOT NULL,

valor FLOAT NULL,

criterio\_id INT NOT NULL,

contador INT DEFAULT 1,

enable TINYINT(1) DEFAULT 1,

timestamp\_arrive TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

observacion TEXT NULL,

FOREIGN KEY (sensor\_id) REFERENCES Sensores(sensor\_id),

FOREIGN KEY (estacion\_id) REFERENCES Estaciones(estacion\_id),

FOREIGN KEY (criterio\_id) REFERENCES Criterio\_alerta(criterio\_id)

);

CREATE TABLE run\_history (

id\_history INT AUTO\_INCREMENT PRIMARY KEY,

run\_instance DATETIME NOT NULL,

rows\_written INT DEFAULT 0,

duration INT DEFAULT 0,

host VARCHAR(100)

);

CREATE TABLE log\_inserts (

log\_id INT AUTO\_INCREMENT PRIMARY KEY,

row\_vigency INT DEFAULT 0, -- Cantidad de sensores vigentes revisados

registers INT DEFAULT 0, -- Total de lecturas procesadas en esa corrida

date\_log DATE NOT NULL, -- Fecha del log

timestamp\_log DATETIME NOT NULL, -- Timestamp exacto del evento/log

event\_count INT DEFAULT 0, -- Eventos de interés detectados (valores fuera de umbral, por ejemplo)

count\_alarm INT DEFAULT 0, -- Total de alarmas generadas

start\_alarm DATETIME DEFAULT '0000-00-00 00:00:00', -- Inicio del primer evento de alarma

vigency\_alarm DATETIME DEFAULT '0000-00-00 00:00:00' -- Hasta cuándo sigue activa la alarma

);

CREATE PROCEDURE GetSensorDetenido(IN horas INT)

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE sensor INT;

DECLARE estacion INT;

DECLARE ultima\_lectura DATETIME;

DECLARE cur CURSOR FOR

SELECT sensor\_id, estacion\_id, MAX(timestamp)

FROM mediciones

GROUP BY sensor\_id, estacion\_id;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

-- Desactiva alertas si el sensor volvió a enviar datos

UPDATE alertas a

JOIN (

SELECT sensor\_id, MAX(timestamp) as ultima

FROM mediciones

GROUP BY sensor\_id

) m ON a.sensor\_id = m.sensor\_id

SET a.enable = 0

WHERE a.criterio\_id = 1 AND a.enable = 1

AND TIMESTAMPDIFF(HOUR, m.ultima, NOW()) < horas;

-- Detectar sensores sin datos recientes y crear alerta si no existe

OPEN cur;

lectura\_loop: LOOP

FETCH cur INTO sensor, estacion, ultima\_lectura;

IF done THEN

LEAVE lectura\_loop;

END IF;

IF TIMESTAMPDIFF(HOUR, ultima\_lectura, NOW()) >= horas THEN

IF NOT EXISTS (

SELECT 1 FROM alertas

WHERE sensor\_id = sensor AND criterio\_id = 1 AND enable = 1

) THEN

INSERT INTO alertas(sensor\_id, estacion\_id, timestamp, valor, criterio\_id, contador, enable, timestamp\_arrive)

VALUES(sensor, estacion, ultima\_lectura, NULL, 1, 1, 1, NOW());

END IF;

END IF;

END LOOP;

CLOSE cur;

END;