**Video relacionado: 8.4, 8.5. Contenido: Encriptación en Redshift**

**Ejercicio encriptación en Redshift**

La empresa de telecomunicaciones donde trabajas como Data engineer recientemente ha añadido información externa de sus clientes. Dentro de las variables más importantes que se han añadido están el ssn el cúal es un código de verificación de ventas y credit\_card\_number que representa la tarjeta elegida por el usuario para hacer compras. Te piden lo siguiente:

**Requerimientos**

1. Una función función en SQL que permita encriptar la data sensible (PII) solo en un sentido (sin posibilidad de encriptación)
2. Otra función en SQL que permita encriptar y desencriptar la información sensible
3. Importante: debes elegir qué columnas con las que se consideran PII
4. Puedes tomar como base la siguiente estructura para la creación de la tabla

**CREATE** **TABLE** customer\_data (

customer\_id **INT**,

first\_name **VARCHAR**(50),

last\_name **VARCHAR**(50),

address **VARCHAR**(100),

city **VARCHAR**(50),

state **VARCHAR**(20),

postal\_code **VARCHAR**(10),

phone\_number **VARCHAR**(20),

email **VARCHAR**(100),

ssn **VARCHAR**(11),

date\_of\_birth **DATE**,

account\_balance **NUMERIC**(10, 2),

credit\_card\_number **VARCHAR**(20),

card\_expiration\_date **DATE**,

card\_cvv **VARCHAR**(5),

favorite\_color **VARCHAR**(20),

last\_login DATETIME,

security\_question **VARCHAR**(100),

security\_answer **VARCHAR**(100),

notes **VARCHAR**(200)

);

**Solución**

En este caso al analizar la data se puede considerar información PII por ejemplo a las variables: **address, poastal\_code, phone\_number, email, date\_of\_birth,credit\_card\_number**

**Función encriptadora en un solo sentido**

**CREATE** **OR** **REPLACE** **FUNCTION** encrypt\_sha256(input\_text **VARCHAR**)

**RETURNS** **VARCHAR**

**STABLE**

**AS** **$$**

import hashlib

**return** hashlib.sha256(input\_text.**encode**('utf-8')).hexdigest()

**$$** **LANGUAGE** plpythonu;

**CREATE** **OR** **REPLACE** **FUNCTION** decrypt\_sha256(input\_text **VARCHAR**)

**RETURNS** **VARCHAR**

**STABLE**

**AS** **$$**

**return** 'Decriptacion no es posible'

**$$** **LANGUAGE** plpythonu;

Verificando que funcione:

**SELECT encrypt\_sha256('Calle 104 13.56') AS encrypted\_text;**

**SELECT decrypt\_sha256('Calle 104 13.56') AS encrypted\_text;**

**Función encriptadora/decriptadora doble sentido**

**CREATE OR REPLACE FUNCTION encrypt\_base64(input\_text VARCHAR)**

**RETURNS VARCHAR**

**STABLE**

**AS $$**

**import base64**

**return base64.b64encode(input\_text.encode('utf-8')).decode('utf-8')**

**$$ LANGUAGE plpythonu;**

**CREATE OR REPLACE FUNCTION decrypt\_base64(input\_text VARCHAR)**

**RETURNS VARCHAR**

**STABLE**

**AS $$**

**import base64**

**try:**

**return base64.b64decode(input\_text).decode('utf-8')**

**except:**

**return 'Decriptacion no es posible'**

**$$ LANGUAGE plpythonu;**

Ahora podemos probar si funciona:

**SELECT** encrypt\_base64('Calle 104 13-56!') **AS** encrypted\_text;

**SELECT** decrypt\_base64('Q2FsbGUgMTA0IDEzLTU2IQ==') **AS** decrypted\_text;