**# Prueba Técnica - TechSupport Plurall**🧡

**Instrucciones**

Resuelve los siguientes problemas aplicando tus conocimientos en estructuras de datos y SQL. Se evaluará tanto la eficiencia de tu solución como la claridad en la explicación de tu razonamiento. Tienes un tiempo límite de 1 hora para completar la prueba.

---

**## \*\*Parte 1: SQL\*\***

*### Problema 1: Producto con más ingresos en los últimos 3 meses*

**Enunciado**

Tienes una tabla `ventas` con las siguientes columnas:

| id | producto | fecha\_venta | cantidad | precio |

|------|-----------|-------------|----------|---------|

| 1 | Televisor | 2024-06-15 | 3 | 500 |

| 2 | Laptop | 2024-07-10 | 2 | 800 |

| 3 | Televisor | 2024-09-05 | 1 | 500 |

| 4 | Teléfono | 2024-08-01 | 5 | 300 |

Escribe una consulta SQL para obtener el producto que generó más ingresos (cantidad \* precio) en los últimos 3 meses desde la fecha actual.

#### Ejemplo:

plaintext

Output: producto con más ingresos

select

    producto,

    sum(cantidad \* precio) as total\_price

from

    ventas

where

    fecha\_venta >= date\_sub( now(), interval 3 month)

group by

    producto

order by

    total\_price

limit 1

**## \*\*Parte 2: AWS\*\***

**Enunciado**

Imagina que encuentras un aumento significativo en los costos sobre Dynamo. ¿Qué componentes de AWS revisarías en primer lugar para identificar la causa raíz de este incremento y qué métricas específicas buscarías dentro de cada componente?

En dynamodb hay un servicio de monitoreo por el uso que se tiene como lo es ver el monitoreo de lectura y escritura y como se están comportando las métricas, aparte de ello si se creo de una forma provisionada o de consumo por uso, aparte de ello es necesario mirar si se están haciendo los filtros de búsqueda de una forma apropiada, por ejp utilizando los índices para no tener que recorrer cada fila en la tabla si no algo más concreto y de esa forma optimizar la búsqueda, incluso al crear una primary key y una sort key juntas siempre se optimizaran los filtros de búsqueda.

**##\*\*Parte 3: Programación\*\***

Crear una aplicación Node.js que genere contraseñas seguras y personalizadas. La aplicación debe permitir al usuario especificar los siguientes criterios:

* Longitud mínima: El número mínimo de caracteres de la contraseña.
* Caracteres incluidos: Indicar si se deben incluir mayúsculas, minúsculas, números y caracteres especiales.
* Cantidad de contraseñas: El número de contraseñas a generar.

Requisitos:

* Fortaleza: Las contraseñas generadas deben ser aleatorias y cumplir con criterios de seguridad (combinación de tipos de caracteres).
* Personalización: El usuario debe tener control total sobre la complejidad de la contraseña.
* Usabilidad: La interfaz de la aplicación debe ser sencilla e intuitiva.
* Buenas prácticas: Utilizar funciones, condicionales, bucles y estructuras de datos de manera eficiente.
* Modularidad: Dividir el código en funciones pequeñas y reutilizables.
* Manejo de errores: Implementar mecanismos para manejar posibles errores, como entradas inválidas.