Enunciado Prueba de Desempeño – Módulo 4 (Bases de Datos SQL)

"Primero se lee, se entiende, se planea y después se desarrolla.
Lo último que se hace en un proyecto es el código fuente."

Introducción

Eres parte del equipo de desarrollo asignado por AgroData Solutions, una empresa especializada en soluciones tecnológicas para el sector agroindustrial en Latinoamérica. Actualmente, uno de los clientes de AgroData —la empresa *AgroVida*— enfrenta dificultades para gestionar la información relacionada con sus cultivos, fincas, sensores de monitoreo y reportes de producción.

La información se encuentra dispersa y desorganizada en múltiples archivos de Excel (.xlsx), lo que dificulta su análisis y seguimiento técnico.

Tu misión como desarrollador es proponer e implementar una solución que permita organizar y estructurar esta información en una base de datos SQL, facilitando su carga, almacenamiento y administración mediante un sistema CRUD, junto con consultas clave que respondan a las necesidades del cliente.

Objetivo

El objetivo de esta prueba es que, como desarrollador:

- Analices la información entregada en el archivo Excel y la normalices manualmente aplicando las tres primeras formas normales (1FN, 2FN, 3FN).
- Construyas un modelo relacional que represente la estructura final de la información normalizada.
- Crees y configures la base de datos SQL según tu modelo.
- Implementes la carga masiva de datos desde CSV a la base ya creada.
- Desarrolles un CRUD para administrar una de las entidades de tu modelo.
- Implementes consultas avanzadas que permitan resolver requerimientos planteados por el cliente.

Requisitos técnicos

Normalización y modelo relacional

- Analizar el archivo Excel desorganizado.
- Convertirlo a CSV para facilidad de carga.
- Diseñar un modelo relacional aplicando 1FN, 2FN y 3FN.
- El modelo debe hacerse manualmente en draw.io (sugerido) o herramienta similar, no autogenerado por Workbench.
- Guardar el modelo como imagen o PDF.

Creación de la base de datos (prueba_nombre_coder _clan)

- Implementar el DDL completo de la base de datos.
- Todas las tablas y atributos deben estar en inglés.
- Incluir claves primarias, foráneas y restricciones (NOT NULL, UNIQUE, etc.).
- La base de datos debe existir antes de la carga masiva.
- Entregar archivo .sql con el script de creación.

Carga masiva desde CSV (solo datos)

- El archivo original se entregará en formato **Excel (.xlsx)**.
- Debes convertirlo a **CSV** para facilitar la carga en la base de datos.
- El proceso de carga debe ejecutarse **de manera local**, pudiendo implementarse de las siguientes formas:
  - Mediante un **script** que se ejecute manualmente.
  - Mediante un **botón en el frontend** que dispare la ejecución del script local.
- La correcta inserción de los datos debe estar documentada en el **README.md**.

Sistema CRUD y Dashboard

- Implementar **Create, Read, Update y Delete** para una de las entidades debiles de tu modelo mediante la creación de un api (express).
- Validar datos antes de insertarlos o actualizarlos.
- Incluir un **dashboard** que funcionará como **frontend mínimo y sencillo** para la gestión de esta entidad.

- El frontend puede desarrollarse con **frameworks/librerías CSS** como Bootstrap, Tailwind o Bulma.
- El diseño visual no es prioritario, pero el funcionamiento del CRUD debe ser completo.

Consultas avanzadas (solo desde Postman)

1. **Producción por finca:**
   "Como administrador de operaciones, necesito saber cuántas toneladas de producción ha generado cada finca durante el mes, para poder planificar la distribución y venta."

2. **Alertas por sensores en cultivos específicos:**
   "Como ingeniero agrónomo, necesito identificar todos los sensores que hayan superado límites críticos (temperatura, humedad, etc.) en cultivos específicos, junto con la fecha y hora del evento, para tomar decisiones preventivas."

3. **Listado de cultivos por región:**
   "Como analista de datos, necesito generar un informe que me permita ver todos los cultivos registrados por región, indicando el tipo de cultivo y el responsable técnico de la finca."

Colección de Postman

Debe incluir:

- Endpoints del CRUD.
- Endpoints de las 3 consultas avanzadas.
- Debe adjuntarse en el repositorio.

Documentación

Incluir un **README.md** en inglés con:

- Descripción del sistema.
- Instrucciones para ejecutar el proyecto.
- Tecnologías utilizadas.
- Explicación de la normalización.
- Instrucciones para la carga masiva desde CSV.
- Explicación de las consultas avanzadas.
- Captura del modelo relacional.
- Datos del desarrollador (nombre, clan, correo).

Puntos extra *(máximo +4 pts sin superar la nota máxima)*

- Implementar carga masiva desde CSV mediante un **endpoint que reciba un archivo (Multer)**.

Criterios de aceptación

- **Modelo relacional:** bien estructurado, normalizado, entregado como imagen o PDF.
- **Base de datos:** creada correctamente con DDL completo y nombres en inglés.
- **Carga masiva desde CSV:** funcional y documentada.
- **CRUD:** implementado y con dashboard funcional.
- **Frontend:** sencillo y funcional.
- **Postman:** colección completa con CRUD + 3 consultas avanzadas.
- **README.md:** claro, completo y en inglés.

Entregables

- Carpeta del proyecto comprimida en **.zip** y subida a Moodle.
- **Repositorio público en GitHub** con:
  - Código fuente backend y frontend.

- DDL de la base de datos.
- Modelo relacional.
- Archivo CSV.
- Colección de Postman.
- README.md documentado.