Práctica Calificada: Integración de Conceptos de PostgreSQL

Profesor: Yoseph Ayala Valencia

Fecha de entrega: 14/03/25

Instrucciones Generales

- **Objetivo:** Repasar y aplicar conceptos de bases de datos relacionales, manipulación de datos, filtrado, joins, agregaciones, y creación de funciones/procedimientos en PostgreSQL, integrando la carga de información desde archivos CSV a través de Python.
- **Requisitos:**
  - Tener instalado y configurado PostgreSQL.
  - Contar con Python y la librería **psycopg2** instalada.
  - Se utilizarán scripts SQL para la manipulación y consulta de datos (excepto la carga de datos, que se realizará en Python).

Parte 1: Creación y Manipulación de la Base de Datos (3 puntos)

- **Tarea:**
  1. **Crear la base de datos:**
     - Una base de datos llamada `tienda_db`.
  2. **Crear las tablas:**
     - **clientes:**
       - `id_cliente` (PK, serial),
       - `nombre` (VARCHAR),
       - `email` (VARCHAR),
       - `fecha_registro` (DATE).
     - **productos:**
       - `id_producto` (PK, serial),
       - `nombre` (VARCHAR),
       - `precio` (NUMERIC),
       - `stock` (INTEGER).
     - **pedidos:**
       - `id_pedido` (PK, serial),
       - `id_cliente` (FK que referencia a clientes),
       - `fecha_pedido` (DATE),
       - `total` (NUMERIC).

     PK: primary key, FK: foreign key

  3. **Manipulación de datos:**
     - Inserta algunos registros de prueba para verificar la estructura (estos serán complementados luego con los datos del CSV).
     - Realiza al menos un `UPDATE` y un `DELETE` en alguna de las tablas.

- **Formato:** Script SQL.

Parte 2: Consultas Básicas y Filtros (2 puntos)

- **Tarea:**
    1. Desarrolla consultas **SELECT** que incluyan:
        - Uso de cláusula **WHERE** con operadores de comparación.
        - Operadores matemáticos (por ejemplo, calcular descuentos o nuevos precios).
        - Operadores de texto como **LIKE** o **ILIKE** para filtrar nombres o emails.
- **Formato:** Script SQL con comentarios que expliquen cada consulta.

Parte 3: Carga de Datos desde CSV con Python (2 puntos)

- **Tarea:**
    1. Se proveerán tres archivos **.csv** (uno para cada tabla: **clientes.csv**, **productos.csv** y **pedidos.csv**)
    2. Desarrolla un script Jupyter Notebook que:
        - Se conecte a la base de datos `tienda_db` utilizando **psycopg2**.
        - Lea cada uno de los archivos CSV.
        - Inserte los datos leídos en la tabla correspondiente.
        - Incorpore manejo básico de errores en la conexión y la ejecución de comandos.
- **Formato:** Jupyter Notebook con comentarios.

Parte 4: Filtros Avanzados y Condicionales (3 puntos)

- **Tarea:**
    1. Escribe una consulta que utilice **CASE WHEN** para categorizar los productos según su precio, por ejemplo:
        - *Económico* si el precio es menor a un determinado umbral,
        - *Moderado* si el precio se encuentra en un rango,
        - *Caro* si supera un valor establecido.
    2. En la misma o en consultas adicionales, incorpora:
        - Operadores lógicos (**IN**, **NOT IN**, **BETWEEN**, **IS NULL**) para filtrar registros.
- **Formato:** Script SQL con comentarios explicativos.

Parte 5: Joins, Subconsultas y Optimización (4 puntos)

- **Tarea:**

1. **Joins:**
    - Realiza una consulta que combine las tablas **clientes** y **pedidos** (por ejemplo, mediante un **INNER JOIN**) para listar cada cliente con sus respectivos pedidos.
2. **Subconsultas:**
    - Crea una subconsulta que calcule el total de ventas por cliente. Utiliza este resultado para filtrar y mostrar solo aquellos clientes con ventas superiores a un umbral determinado.

- **Formato:** Script SQL.

Parte 6: Funciones de Agregación, Agrupamiento y Funciones de Ventana (4 puntos)

- **Tarea:**
    1. Escribe una consulta que:
        - Utilice funciones de agregación (**SUM**, **COUNT**, **AVG**, **MIN**, **MAX**) para resumir los datos de la tabla **pedidos**.
        - Emplee **GROUP BY** para agrupar datos (por ejemplo, por cliente o por fecha).
        - Use **HAVING** para filtrar los resultados agregados (por ejemplo, mostrar solo clientes con más de cierta cantidad de pedidos).
    2. Añade una función de ventana (por ejemplo, **ROW_NUMBER() OVER (ORDER BY total DESC)**) para asignar un ranking a los clientes según el total de sus pedidos.
- **Formato:** Script SQL.

Parte 7: Variables, Procedimientos Almacenados y Funciones (2 puntos)

- **Tarea:**
    1. Crea una función o procedimiento almacenado en PostgreSQL que:
        - Reciba como parámetro un `id_cliente`.
        - Retorne (o imprima) el total de ventas realizadas por ese cliente (sumando el campo `total` de la tabla **pedidos**).
        - Utilice variables internas según sea necesario.
- **Formato:** Script SQL.

Criterios de Evaluación

- **Implementación Correcta:** Cada parte debe cumplir los requerimientos solicitados.
- **Claridad del Código:** Uso adecuado de comentarios y estructura clara en los scripts.

Entrega:

- El formato de entrega es el script de SQL (bien comentado) y el script Jupyter notebook (bien comentado).