03. Consultas a Bases de Datos Estructuradas - Desafío 2

Se pide utilizar la consulta dada (Figura 1) que muestra los mejores médicos para cada procedimiento (los que tienen puntuación 1) para escribir una consulta que, dado un conjunto de productos de inventario registrados en la tabla Productos (product_id, product_name, product_cost, product_stock, ...) de un ERP, un conjunto de Clientes (customer_id, customer_name, etc) que compran estos productos periódicamente, quedando las ventas registradas en la tabla Invoice (invoice_id, invoice_date, invoice_product_id, invoice_mount, etc) devuelva:

- Productos más vendidos en el último mes y a qué clientes en orden a su volumen de compra.
- Monto de ventas por cliente de mayor ranking.

```sql
WITH cte AS (
  SELECT
    name,
    first_name,
    last_name,
    COUNT(*) c,
    RANK() OVER(PARTITION BY name ORDER BY COUNT(*) DESC) AS rank
  FROM procedure p
  JOIN doctor d
    ON p.doctor_id = d.id
  WHERE score >= (SELECT avg(score)
                  FROM procedure p1
                  WHERE p1.name = p.name)
  GROUP BY name, first_name, last_name
)

SELECT
  name,
  first_name,
  last_name
FROM cte
WHERE rank = 1;
```

Figura 1 - Consulta a adaptar.

Para lograr este objetivo, primero se debe analizar la consulta brindada para comprender su estructura y funcionalidad. A partir de este análisis, es posible identificar las partes clave de la consulta y adaptarlas para obtener los datos específicos que se necesitan en la nueva realidad descripta. Esto implica modificar los criterios de selección, condiciones, y agrupaciones en función de los

nuevos requisitos, asegurando que los resultados reflejen correctamente la nueva realidad planteada.

La primera consulta a resolver es la que brinda una lista de los productos más vendidos en el último mes y a qué clientes se les vendió en orden a su volumen de compra. Primero que nada para este fin se consideró que la tabla de Invoice también guarda información del id del cliente porque es necesario que estas tablas están relacionadas, tiene sentido que se guarde en la compra la cantidad pagada, el cliente, el producto.

Para resolver la consulta se adaptó la consulta brindada. Se realizó una primera consulta donde se obtuvo una tabla con el nombre del producto, nombre del cliente, y la suma de las compras agrupado por nombre de producto y por cliente. Para esto se combinaron las tres tablas ya que se requería información que estaba distribuida en las diferentes tablas. La tabla de Invoice se combinó con la de producto a través del id del producto, y con la de cliente a través del id del cliente. También se filtraron las compras hechas en el último mes, y se creó un ranking a partir de la cantidad de plata gastada por producto.

Luego se realiza una consulta utilizando esta tabla, donde se quiere nombre de producto, nombre de cliente que más compro y total de ventas del producto, mostrando solamente los productos más vendidos (línea 27), ordenados de mayor a menor recaudación.

```sql
WITH cte AS (
  SELECT
    p.product_name,
    c.customer_name,
    SUM(i.invoice_amount) AS total_sales,
    RANK() OVER(PARTITION BY p.product_name ORDER BY SUM(i.invoice_amount) DESC) AS rank
  FROM
    invoice AS i
  JOIN
    productos AS p
  ON
    p.product_id = i.invoice_product_id
  JOIN
    clientes AS c
  ON
    c.customer_id = i.customer_id
  WHERE i.invoice_date >= '2024-07-21' -- Sabiendo la fecha de hoy colocamos un mes antes
  GROUP BY p.product_name, c.customer_name
)
SELECT
  product_name,
  customer_name,
  total_sales
FROM
  cte
WHERE
  rank = 1
ORDER BY total_sales DESC;
```

Cuando conocemos la fecha actual y queremos retroceder un mes, es posible hacerlo manualmente si no tenemos acceso a funciones nativas que ajusten automáticamente la fecha en SQL. Haciéndolo manualmente hay que tener cuidado en casos como enero, al retroceder un mes, es necesario restar 1 al año y cambiar el mes de 01 a 12 (es decir, retrocedemos al diciembre del año anterior). Por ejemplo, si la fecha es 07/01/25, al restar un mes, el resultado debería ser 07/12/24. Es importante también considerar días como el 31, ya que no todos los meses tienen ese número de días. En tales casos, es necesario ajustar el día al último día del mes anterior. Es importante asegurarse también de que el formato de fecha sea consistente en las consultas y que las operaciones de manipulación de fechas sean compatibles con el motor SQL específico.

El segundo dato que se pide es el monto de ventas por cliente de mayor ranking.

Esta consulta es un poco más sencilla ya que en este caso no se necesita información de productos, solamente se debe de combinar la tabla de clientes con la de invoice. Puntualmente se quiere agrupar por cliente, realizar la suma de sus compras, y ordenar de mayor a menor, mostrando el cliente que tenga el monto más grande.

Esta consulta se puede realizar de diferentes formas. A mi se me ocurrieron dos formas para hacerla. Una opción es simplemente agrupar por nombre de cliente, guardar información del nombre y de la suma de los pagos, y luego filtrar que muestre la información ordenada por ventas totales del cliente de forma descendente, y quedándonos con el primer registro de la tabla (LIMIT 1). Esto tiene la desventaja de que en el caso de que varios clientes coincidan en las ventas totales solo va a aparecer un registro cuando deberían de aparecer los nombres de todos los clientes que hayan vendido esa cantidad.

Para corregir esto se puede nuevamente usar el mismo patrón de consulta utilizado anteriormente, generando un ranking, y en el caso de que coincidan todos estos clientes quedan con rank=1, y la condición para mostrarlo es esa, entonces mostraría a todos los clientes que vendieron esa cantidad.

```sql
WITH cte AS (
    SELECT
        c.customer_name,
        SUM(i.invoice_amount) AS total_sales,
        RANK() OVER(ORDER BY SUM(i.invoice_amount) DESC) AS rank
    FROM
        invoice AS i
    JOIN
        clientes AS c
    ON
        c.customer_id = i.customer_id
    GROUP BY c.customer_name
)
SELECT
    customer_name,
    total_sales
FROM
    cte
```

```sql
WITH cte AS (
    SELECT
        c.customer_name,
        SUM(i.invoice_amount) AS total_sales
    FROM
        invoice AS i
    JOIN
        clientes AS c
    ON
        c.customer_id = i.customer_id
    GROUP BY c.customer_name
)
SELECT
    customer_name,
    total_sales
FROM
    cte
ORDER BY total_sales DESC
```