Desafio KANBAN- Maria Emilia Caraballo. 03.Consultas a Bases de Datos Estructuradas

1er Desafío SQL

Analice la siguiente consulta SQL y explique lo que se obtiene con la misma:

```
SELECT
 fecha_renta,
 titulo,
 genero,
 cantidad_pagada,
 RANK() OVER(PARTITION BY genero ORDER BY cantidad_pagada
DESC)
FROM pelicula
JOIN alquiler
 ON alquiler.pelicula_id = pelicula.id;
```

Hecho lo anterior, diseñe tabla de clientes con al menos 3 campos y arme una nueva consulta que incorpore datos de estos, ordenados por nombre de película y cliente.

La consulta SQL anterior genera una lista de todas las películas alquiladas por tanto que el id de la película se encuentra en la tabla de alquileres. La lista tiene las columnas de la fecha de renta, el título, el género, la cantidad pagada y un ranking que ordena las películas por género según la cantidad pagada en orden descendente (mayor a menor).
La consulta realiza un JOIN entre las tablas pelicula y alquiler usando las columnas de id de pelicula y pelicula_id en alquiler, esto permite acceder a la información proporcionada por ambas tablas.
Sería una buena práctica nombrar a la columna del ranking para hacer la tabla más organizada, modificando la línea por:
RANK() OVER (PARTITION BY genero ORDER BY cantidad_pagada DESC) AS ranking

Creación tabla clientes con campos id (identificador único por cliente), el nombre del cliente, el email del cliente y la fecha en la que el mismo se registró al sistema:

```
CREATE TABLE cliente (
    id INT PRIMARY KEY,
    nombre VARCHAR(100),
    email VARCHAR(100),
    fecha_registro DATE
);
```

Consulta que incorpora los datos de la tabla cliente, info de la película, y los ordena por nombre de pelicula y de cliente:

```
SELECT
    p.titulo AS nombre_pelicula,
    c.nombre AS nombre_cliente,
    c.email,
    c.fecha_registro,
    p.genero,
    a.fecha_renta,
    a.cantidad_pagada
FROM
    pelicula p
JOIN
    alquiler a ON a.pelicula_id = p.id
JOIN
    cliente c ON a.cliente_id = c.id
ORDER BY
    p.titulo ASC,
    c.nombre ASC;
```

2do Desafío SQL

Utilice la siguiente consulta:

```
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 pl
                  WHERE pl.name = p.name)
  GROUP BY name, first_name, last_name
)

SELECT
  name,
  first_name,
  last_name
FROM cte
WHERE rank = 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.

```
WITH last_month_sales AS (
    SELECT
        p.product_id,
        p.product_name,
        c.customer_id,
        c.customer_name,
        SUM(i.invoice_mount) AS total_sales,
        COUNT(i.invoice_product_id) AS quantity_sold
    FROM
        invoice i
    JOIN
        Productos p ON i.invoice_product_id = p.product_id
    JOIN
        Clientes c ON i.customer_id = c.customer_id
    WHERE
        i.invoice_date >= DATEADD(MONTH, -1, GETDATE())
    GROUP BY
        p.product_id, p.product_name, c.customer_id, c.customer_name
),
sales_ranking AS (
    SELECT
        v.product_id,
        v.product_name,
        v.customer_id,
        v.customer_name,
        v.total_sales,
        v.quantity_sold,
        RANK() OVER (PARTITION BY v.product_id ORDER BY v.quantity_sold
DESC) AS product_rank,
        RANK() OVER (PARTITION BY v.customer_id ORDER BY v.total_sales DESC)
AS customer_rank
    FROM
        last_month_sales v
)
SELECT
    r.product_name AS top_selling_product,
    r.customer_name AS customer,
    r.total_sales AS total_sales_amount
FROM
    sales_ranking r
WHERE
    r.product_rank = 1
ORDER BY
    r.customer_rank ASC;
```

La consulta lo que hace es al inicio definir una CTE que se llama last_month_sales, esto es un conjunto de resultados temporales para usarse en la consulta principal. Dentro de last_month_sales se selecciona el id y el nombre del producto, el id y el nombre del cliente, se calcula el monto total de ventas para cada combinación de producto y cliente en el último mes y se cuenta cuántas veces se vende cada producto a cada cliente. Luego se realiza un Join de las tablas invoice, productos y clientes, donde se vincula a cada invoice con el producto y con el cliente correspondiente, filtrando únicamente las facturas del último mes. Por último se agrupan los resultados por producto y cliente para sumar el total de ventas y la cantidad vendida para cada combinación.

Luego se define otra CTE que la llame sales_ranking, construida a partir de los resultados de la CTE anterior.
Se seleccionan todos los campos de last_month_sales, incluidas la suma y el conteo. A partir de esto se calcula un ranking para cada producto dentro de su grupo, basado en la cantidad total vendida en orden descendente. Y por último en este CTE se calcula un ranking para cada cliente dentro de su grupo según id, basada en el monto total de ventas en orden descendente.

Por último se seleccionan los datos de la CTE sales_ranking, el nombre del producto, el nombre del cliente y el monto total de ventas para la combinación de producto-cliente. Se filtran los resultados para incluir solo el producto más vendido para cada Id de producto, o sea el producto con mayor cantidad vendida para cada cliente. Y por último se ordenan los resultados por el ranking del cliente en orden ascendente por lo que el cliente con mayor cantidad de ventas aparece primero.