FUTURISTICLAB
Consultoría en Disruptividad Estratégica

Menu

Aprenda a realizar consultas en SQL en 2 horas. 2da Parte

October 6, 2019 by user

Segunda Hora

En la siguiente articulo (segunda hora) hablaremos de como producir consultas tomando y relacionado datos de distintas tablas.

> (Recuerde que estamos trabajado las consultas en esta página de internet: https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all)

Ahora vamos a trabajar con consultas que mezclan los datos de varias tablas. Por ejemplo necesitamos dos columnas, la primera con el nombre del cliente y la segunda con el OrderID de cada una de sus ordenes.

En este caso la tabla Orders me proporciona el id del cliente más no su nombre por lo que necesito cruzar la información de la tabla Orders con la de Customers.

Para tal fin tengo que hacer uso de la palabra **JOIN**.

Esta consulta se vería de la siguiente forma

```
SELECT C.CustomerName AS Cliente, O.OrderID AS Ordenes FROM Orders AS
O JOIN Customers AS C ON O.CustomerID = C.CustomerID;
```

Explicación: Recordemos que AS es para renombrar, en este caso renombramos las tablas Orders y Customer con la letra O y C respectivamente (esto para menor extensión de la consulta en caracteres y mayor legibilidad) .

A su vez llamamos en la consulta una columna de cada tabla, una columna de Customers y otra de Orders, para saber a que tabla estamos haciendo referencia anteponemos al nombre de la columna el nombre de la tabla y un punto, es decir: C.Customer o O.OrderID, posteriormente con la palabra JOIN juntamos o relacionamos estas dos tablas de tal manera que siempre haya una equivalencia entre las dos(datos en común o llaves foraneas), las tablas se juntan por lo general en donde haya una coincidencia en los datos por ejemplo las dos tablas tienen el campo CustomerID lo que asegura que al juntarles en ese campo va a ver consistencia en los datos.

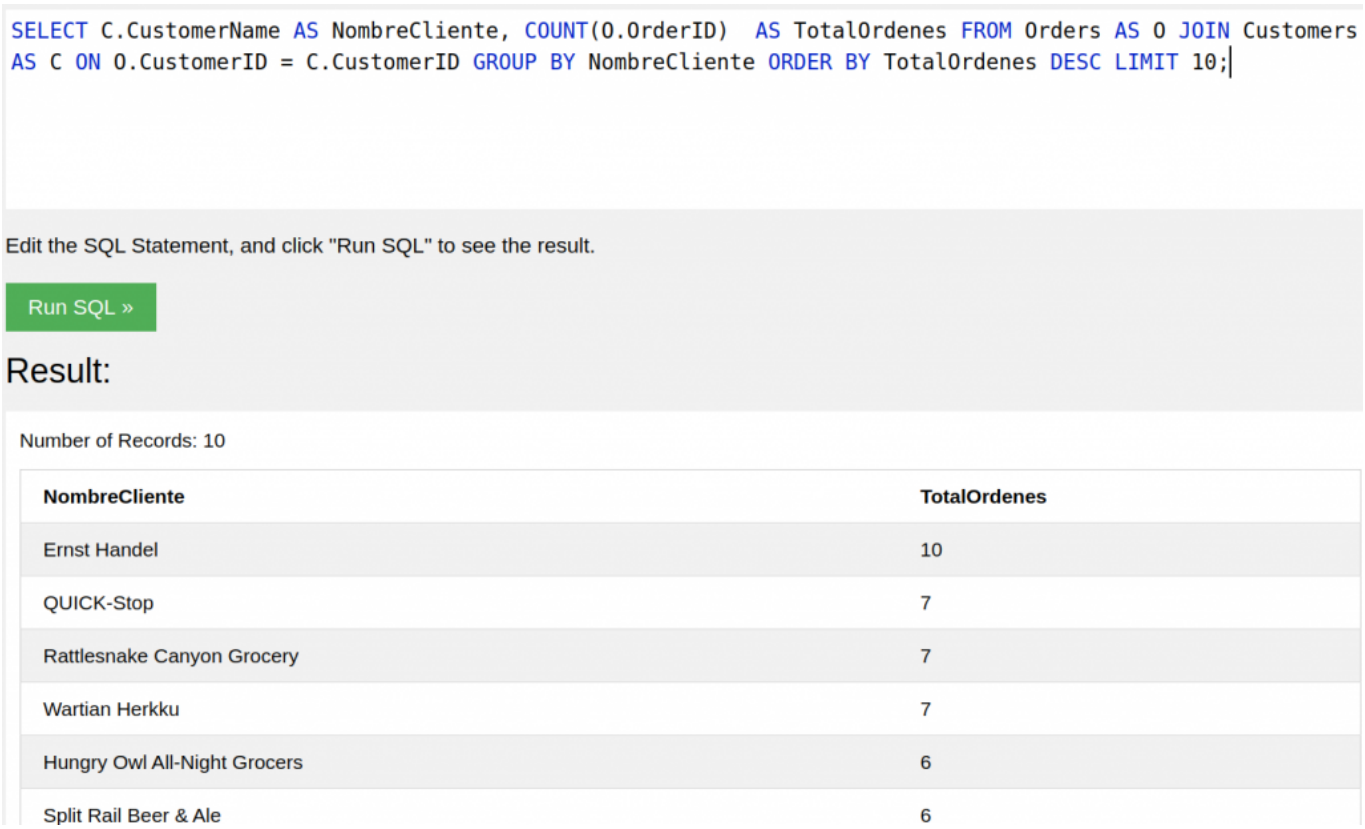
Ahora modifiquemos la consulta anterior con GROUP BY para que nos muestre el número de ordenes por cliente. Es decir la consulta debe tener dos columnas una NombreCliente y Otra TotalOrdenes. Este seria el resultado:

```
SELECT C.CustomerName AS NombreCliente, COUNT(O.OrderID) AS
TotalOrdenes FROM Orders AS O JOIN Customers AS C ON O.CustomerID =
C.CustomerID GROUP BY NombreCliente ;
```

Hicimos que nos cuente todas las ordenes de la tabla las cuales serán agrupadas de acuerdo al nombre del cliente y que nos las agrupe por cliente.

Si queremos los primeros 10 los clientes con mayor número de ordenes seria:

```
SELECT C.CustomerName AS NombreCliente, COUNT(O.OrderID) AS TotalOrdenes FROM Orders AS O JOIN Customers AS C ON O.CustomerID = C.CustomerID GROUP BY NombreCliente ORDER BY TotalOrdenes DESC LIMIT 10;
```



En este caso al agregar organizamos las ordenes con ORDER BY de mayor a menor y limitamos los resultados a 10 con la palabra LIMIT.

LIMIT también puede recibir parámetros, por ejemplo si quiero que me muestre el quinto y el sexto mayor cliente puedo limitar la consulta de esta manera.

```
SELECT C.CustomerName AS NombreCliente, COUNT(O.OrderID) AS TotalOrdenes FROM Orders AS O JOIN Customers AS C ON O.CustomerID =
```

```
C.CustomerID GROUP BY NombreCliente ORDER BY TotalOrdenes DESC
LIMIT 4,2;
```

El cuatro representa el quinto resultado, ya que se empieza contando desde la posición cero (cosas de la gente que diseño las bases de datos) y a partir de ahí se entregan dos filas.

> **Ejercicio:** Haga una consulta para obtener una consulta que tenga el nombre del producto y otra columna con el total de unidades vendidas.

La palabra JOIN de las consultas anteriores también puede ser remplazada y es equivalente a INNER JOIN. Este tipo de unión solamente traerá una nueva tabla con los datos en común de las dos tablas. Como se ve en la imagen:

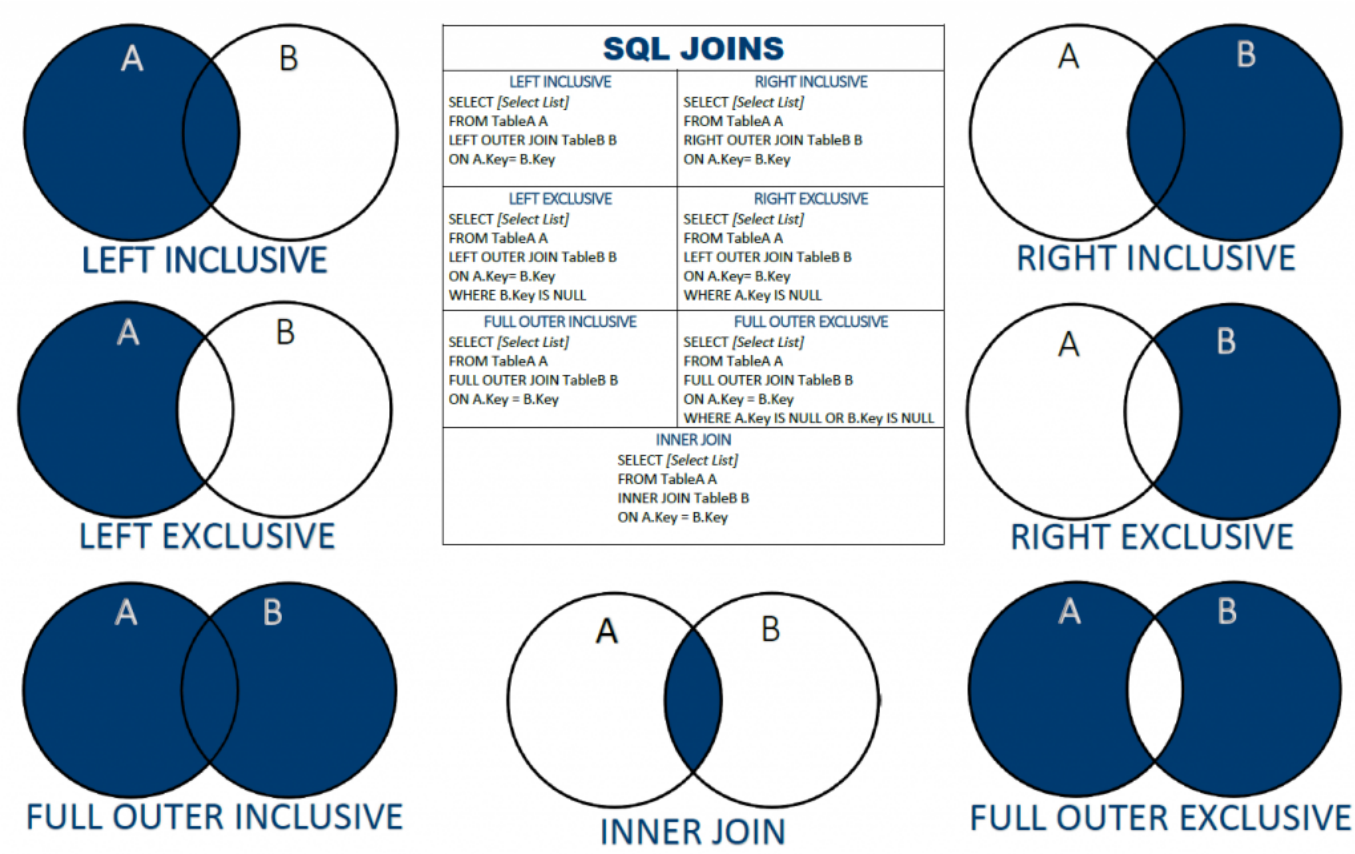


Imagen toma de:
https://www.reddit.com/r/SQL/comments/aysflk/sql_join_chart_custom_poster_size/

Vamos a hacer una consulta LEFT y su utilidad. Por ejemplo si queremos cruzar Orders con Customers nos va a traer datos de los clientes que realizaron ordenes (datos en común) pero no nos va traer datos de los clientes que nunca han realizado una orden (ya que esto no es un dato en común) en este caso utilizariamos una LEFT JOIN en la siguiente consulta vamos a obtener los nombres de los clientes que nunca han realizado una orden.

```
SELECT Customers.CustomerName, Orders.OrderID FROM Customers LEFT JOIN Orders ON Customers.CustomerID=Orders.CustomerID WHERE Orders.OrderID is NULL ORDER BY Customers.CustomerName;
```

Ejercicio: Realice una consulta para verificar que todos los productos se han ordenado. Si todos los productos han sido ordenados al menos una vez la consulta no arroja resultados.

No solamente puedo enlazar dos tablas, pueden ser todas las que yo quiera por ejemplo hagamos una consulta que me muestre el nombre del cliente en una columna, en otra el número de productos (SKU) que ha comprado históricamente. Todo esto organizado por el cliente con mayor penetración de nuestro portafolio de mayor a menor. La información tiene que obtenerse de tres tablas, la de Customers, la de Orders que tiene el identificador de orden por cliente, y la de OrderDetails donde esta el detalle por producto. La consulta seria algo como lo siguiente:

```
SELECT C.CustomerName AS Cliente,COUNT (DISTINCT OD.ProductID) AS SKUporCliente, FROM OrderDetails AS OD INNER JOIN Orders ON OD.OrderID = Orders.OrderID JOIN Customers AS C ON Orders.CustomerID = C.CustomerID GROUP BY Cliente ORDER BY SKUporCliente DESC ;
```

La palabra **DISTINCT tiene como objetivo asegurar que el conteo de los productos se haga sobre productos distintos, es decir que cada ProducID sea contado una única vez.**

En SQL incluso podemos hacer algoritmos de recomendación, un algoritmo de recomendación responde la pregunta, para un producto en particular por ejemplo el ProductID =31, cual es el producto que más frecuentemente lo acompaña en una orden de ventas, veamos la consulta:

```
SELECT ProductID, COUNT(OrderID) AS EnTantasOrdenes FROM OrderDetails
WHERE OrderID IN(SELECT OrderID FROM OrderDetails WHERE ProductID =
31) AND ProductID IS NOT 31 GROUP BY ProductID ORDER BY
EnTantasOrdenes DESC ;
```

(SELECT OrderID FROM OrderDetails WHERE ProductID = 31) La consulta interna nos arroja el OrdenID de las ordenes que contienen el producto principal sobre el cual queremos que nos muestre recomendaciones (Cuando una consulta esta dentro de otra se llaman consultas anidadas). En la consulta exterior decimos que nos muestre todos los productos que estén asociados a las ordenes de la consulta interna con WHERE IN, excluyendo el producto sobre el cual estamos buscando las recomendaciones (IS NOT), sobre esos productos le vamos a pedir que nos agrupe el conteo de ordenes que tiene cada producto, y que nos ordene de mayor a menor. Los primeros resultados serán los productos que más se venden con el producto 31.

SQL Statement:

```
SELECT ProductID, COUNT(OrderID) AS EnTantasOrdenes FROM OrderDetails WHERE OrderID IN(SELECT OrderID
FROM OrderDetails WHERE ProductID = 31) AND ProductID IS NOT 31 GROUP BY ProductID ORDER BY
 EnTantasOrdenes DESC ;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 26

ProductID	EnTantasOrdenes
2	3
72	3
14	2
55	2

Algoritmo de recomendación en SQL

Ejercicio: Reemplazar el ProductID por el ProductName en la consulta. Para eso habrá que relacionar tablas.

Espero que este pequeño tutorial haya sido de su agrado y si tienen dudas por favor déjenlas en los comentarios.

Inteligencia de Negocios, IT

sql

‹ Aprenda a realizar consultas en SQL en 2 horas. 1ra parte.

› Materiales Curso de Power BI

Leave a Comment

Name *

Email *

Website

Post Comment

Search …

Recent Posts

Materiales Curso de Power BI

Aprenda a realizar consultas en SQL en 2 horas. 2da Parte

Aprenda a realizar consultas en SQL en 2 horas. 1ra parte.

Aplicación de análisis de sentimientos a encuestas de satisfacción

Análisis histórico de eventos de minas antipersonas y desminado utilizando POWER BI

Recent Comments

erotik izle on Materiales Curso de Power BI

Edward on Tendencias para INNOVAR en la industria de BEBIDAS en 2017 y 2018

Antonella - clima laboral on 2 CASOS EXITOSOS DE ESTRATEGIAS DE FELICIDAD CORPORATIVA.

Archives

February 2020

October 2019

March 2019

November 2018

June 2018

May 2018

March 2018

February 2018

January 2018

December 2017

November 2017

October 2017

August 2017

July 2017

June 2017

May 2017

March 2017

February 2017

January 2017

November 2016

August 2016

July 2016

June 2016

May 2016

April 2016

February 2016

January 2016

Categories

Creatividad

Cultura Organizacional

Estrategia

Innovación

Inteligencia de Negocios

IT

Líderes

Prospectiva

Tendecias

Uncategorized

Meta

Log in

Entries feed

Comments feed

WordPress.org

Estrategias innovadoras

Genere innovación y diferenciación tanto en el negocio tradicional de su compañía, como para nuevas unidades de negocios.