En esta guía continuaremos con la manipulación de datos desde mysql en este caso con el manejo de consultas un poco más complejas que las de la anterior guía, en esta trabajaremos con los comandos:

- COUNT
- MAX
- MIN
- SUM
- Group By
- Having
- Alias
- Concatenar Funcion
- Substring Funcion

Funciones

Ya que hemos comenzado trabajando con números, la siguiente pregunta natural a realizarse es si es posible hacer cálculos matemáticos con aquellos números, tales como sumas, o sacar un promedio. ¡La respuesta es sí! SQL tiene varias funciones aritméticas, y estas son:

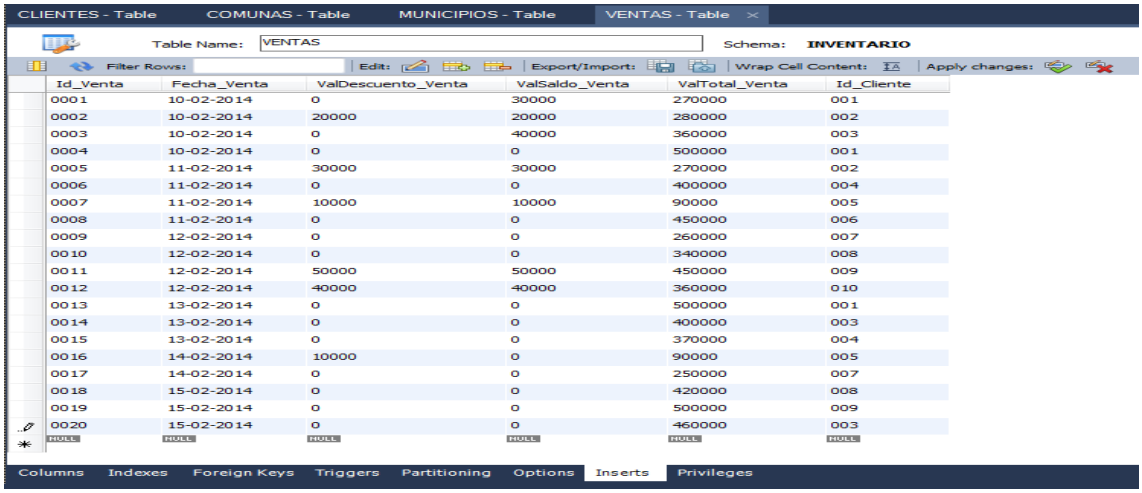
- **AVG**
- **COUNT**
- **MAX**
- **MIN**
- **SUM**

La sintaxis para el uso de funciones es,

```
SELECT "tipo de función"("nombre_columna")
FROM "nombre_tabla";
```

Por ejemplo, si deseamos obtener la sumatoria de todas las ventas totales de la tabla ventas,

Tabla ***VENTAS***

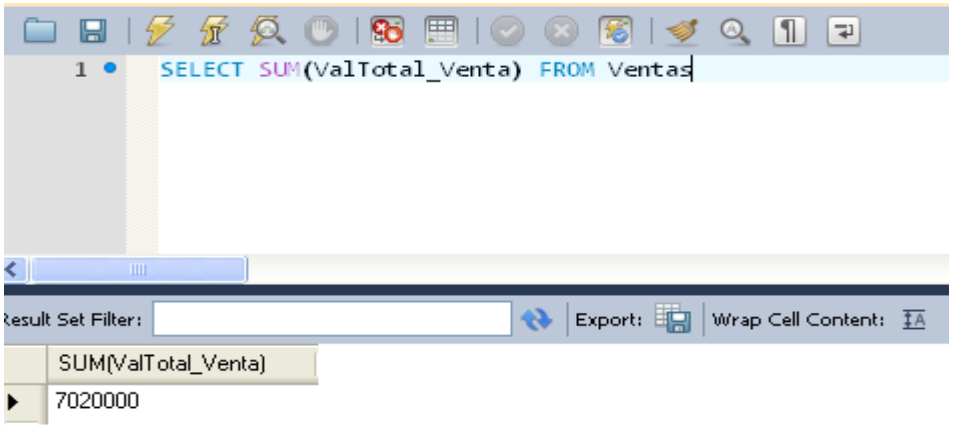


Id_Venta	Fecha_Venta	ValDescuento_Venta	ValSaldo_Venta	ValTotal_Venta	Id_Cliente
0001	10-02-2014	0	30000	270000	001
0002	10-02-2014	20000	20000	280000	002
0003	10-02-2014	0	40000	360000	003
0004	10-02-2014	0	0	500000	001
0005	11-02-2014	30000	30000	270000	002
0006	11-02-2014	0	0	400000	004
0007	11-02-2014	10000	10000	90000	005
0008	11-02-2014	0	0	450000	006
0009	12-02-2014	0	0	260000	007
0010	12-02-2014	0	0	340000	008
0011	12-02-2014	50000	50000	450000	009
0012	12-02-2014	40000	40000	360000	010
0013	13-02-2014	0	0	500000	001
0014	13-02-2014	0	0	400000	003
0015	13-02-2014	0	0	370000	004
0016	14-02-2014	10000	0	90000	005
0017	14-02-2014	0	0	250000	007
0018	15-02-2014	0	0	420000	008
0019	15-02-2014	0	0	500000	009
0020	15-02-2014	0	0	460000	003
NULL	NULL	NULL	NULL	NULL	NULL

Ingresaríamos

```
SELECT SUM(ValTotal_Venta) FROM Ventas;
```

Resultado:



Además de utilizar dichas funciones, también es posible utilizar SQL para realizar tareas simples como suma (+) y resta (-). Para ingresar datos del tipo caracter, hay también varias funciones de cadenas disponibles, tales como funciones de concatenación, reducción y subcadena. Los diferentes proveedores RDBMS tienen diferentes implementaciones de funciones de cadenas, y es mejor consultar las referencias para sus RDBMS a fin de ver cómo se utilizan estas funciones.

EJERCICIO 8

CONSULTA 22: de la Tabla **VENTAS**, sumar todas las ventas totales

CONSULTA 23: de la Tabla **VENTAS**, sumar todos los descuentos de ventas

CONSULTA 24: de la Tabla **VENTAS**, sumar todos los saldos de ventas

CONSULTA 25: de la Tabla **VENTAS**, sumar todos los saldos de ventas totales realizadas al cliente con Id_Cliente 001

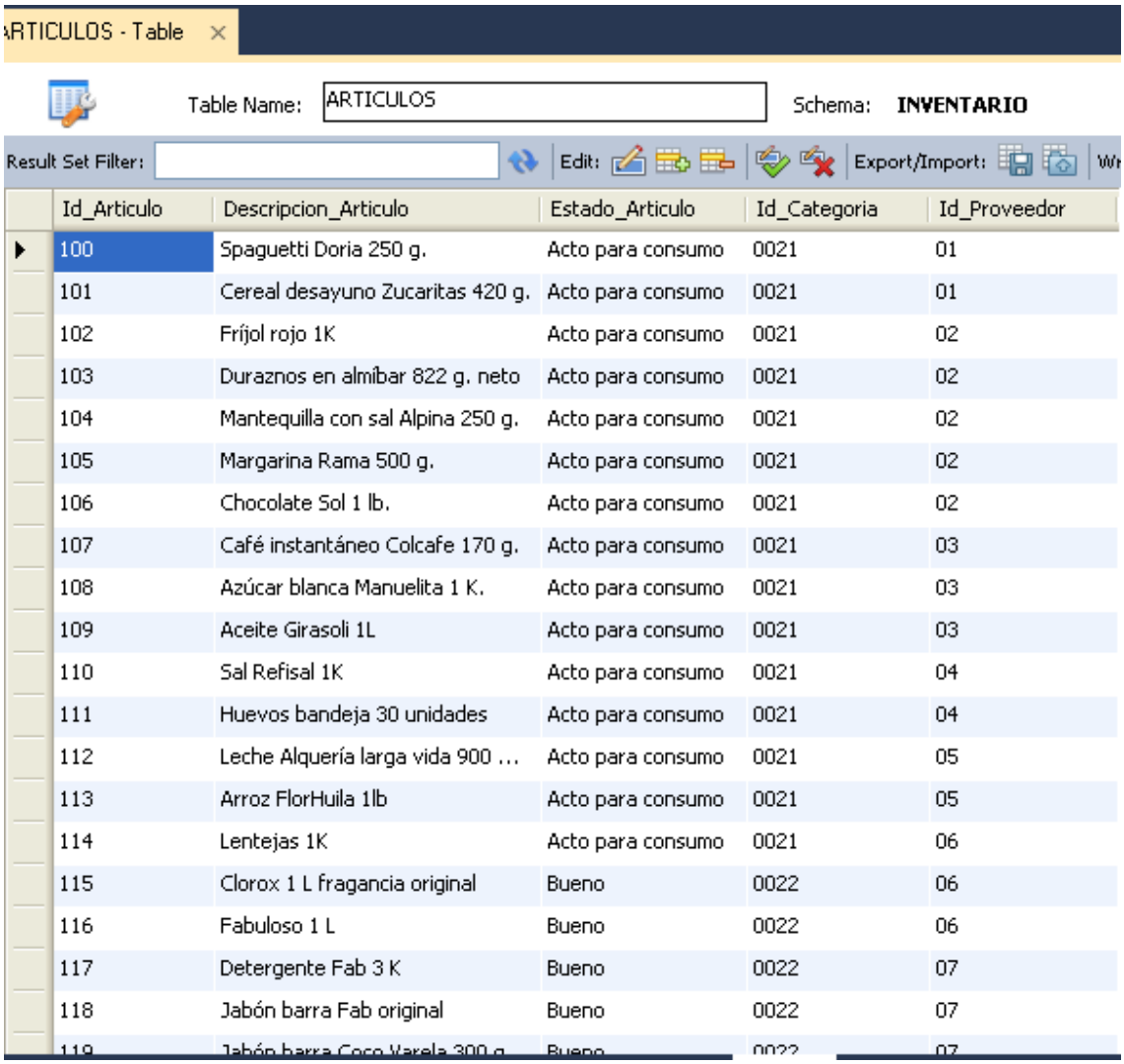
Count

Otra función aritmética es **COUNT**. Esto nos permite **COUNT** el número de filas en una tabla determinada. La sintaxis es,

```
SELECT COUNT("nombre_columna")
FROM "nombre_columna";
```

Por ejemplo, si deseamos conocer el número de artículos en nuestra tabla Articulos,

Tabla ***ARTICULOS***

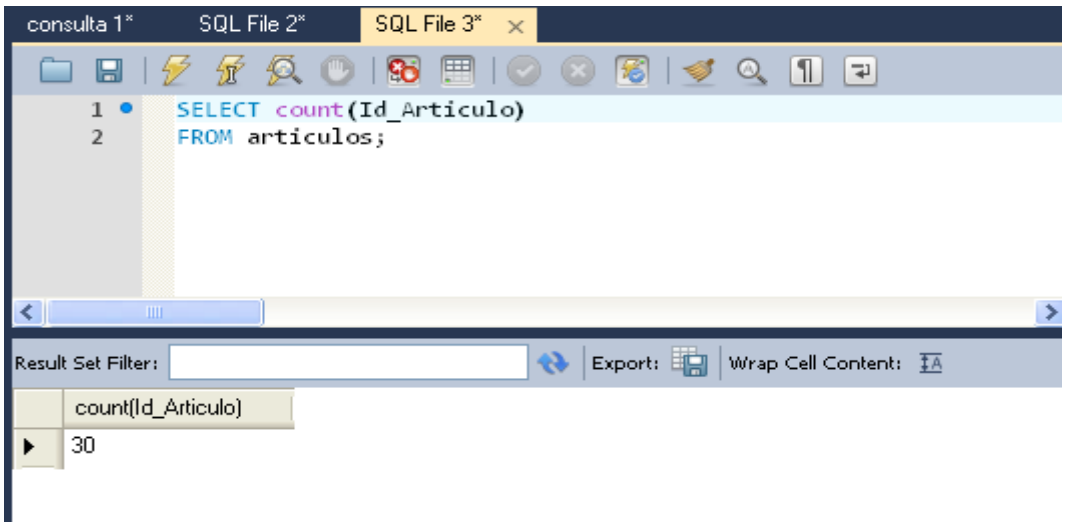


Id_Articulo	Descripcion_Articulo	Estado_Articulo	Id_Categoria	Id_Proveedor
100	Spaguetti Doria 250 g.	Acto para consumo	0021	01
101	Cereal desayuno Zucaritas 420 g.	Acto para consumo	0021	01
102	Fríjol rojo 1K	Acto para consumo	0021	02
103	Duraznos en almíbar 822 g. neto	Acto para consumo	0021	02
104	Mantequilla con sal Alpina 250 g.	Acto para consumo	0021	02
105	Margarina Rama 500 g.	Acto para consumo	0021	02
106	Chocolate Sol 1 lb.	Acto para consumo	0021	02
107	Café instantáneo Colcafe 170 g.	Acto para consumo	0021	03
108	Azúcar blanca Manuelita 1 K.	Acto para consumo	0021	03
109	Aceite Girasoli 1L	Acto para consumo	0021	03
110	Sal Refisal 1K	Acto para consumo	0021	04
111	Huevos bandeja 30 unidades	Acto para consumo	0021	04
112	Leche Alquería larga vida 900 ...	Acto para consumo	0021	05
113	Arroz FlorHuila 1lb	Acto para consumo	0021	05
114	Lentejas 1K	Acto para consumo	0021	06
115	Clorox 1 L fragancia original	Bueno	0022	06
116	Fabuloso 1 L	Bueno	0022	06
117	Detergente Fab 3 K	Bueno	0022	07
118	Jabón barra Fab original	Bueno	0022	07
119	Jabón barra Coco Varela 300 g.	Bueno	0022	07

Ingresamos,

```
SELECT COUNT(Id_Articulo)
FROM Articulos;
```

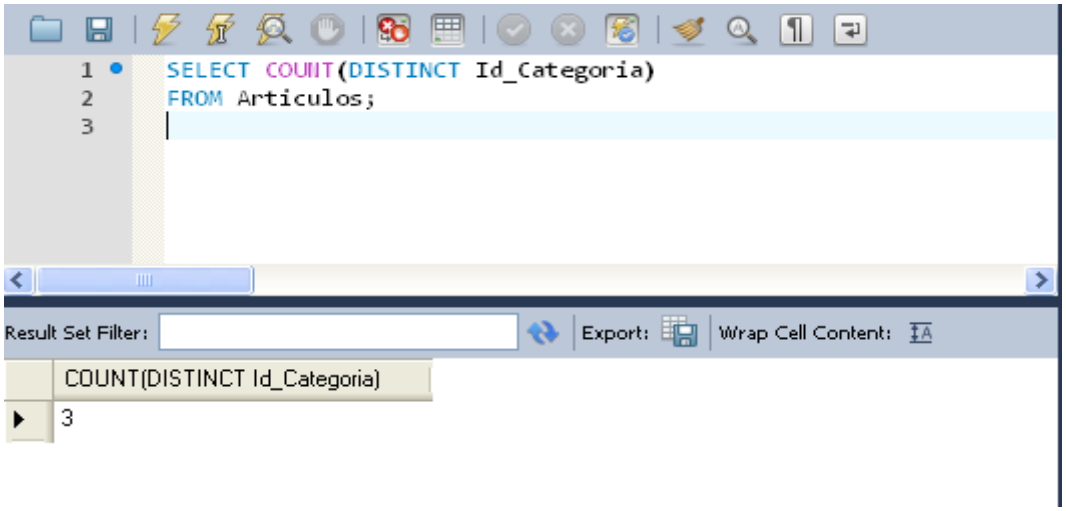
Resultado:



COUNT y **DISTINCT** pueden utilizarse juntos en una instrucción para determinar el número de las distintas entradas en una tabla. Por ejemplo, si deseamos saber el número de las distintas categoría de los articulos, ingresaríamos,

```
SELECT COUNT(DISTINCT Id_Categoria)
FROM Articulos;
```

Resultado:



EJERCICIO 9

CONSULTA 26: de la Tabla **ARTICULOS**, contar los artículos existentes
CONSULTA 27: de la Tabla **ARTICULOS**, contar las distintas categorías de los artículos.
CONSULTA 28: de la Tabla **VENTAS**, contar todos los Id_clientes distintos
CONSULTA 29: de la Tabla **DETALLE_VENTAS**, contar todos los Id_Venta distintos

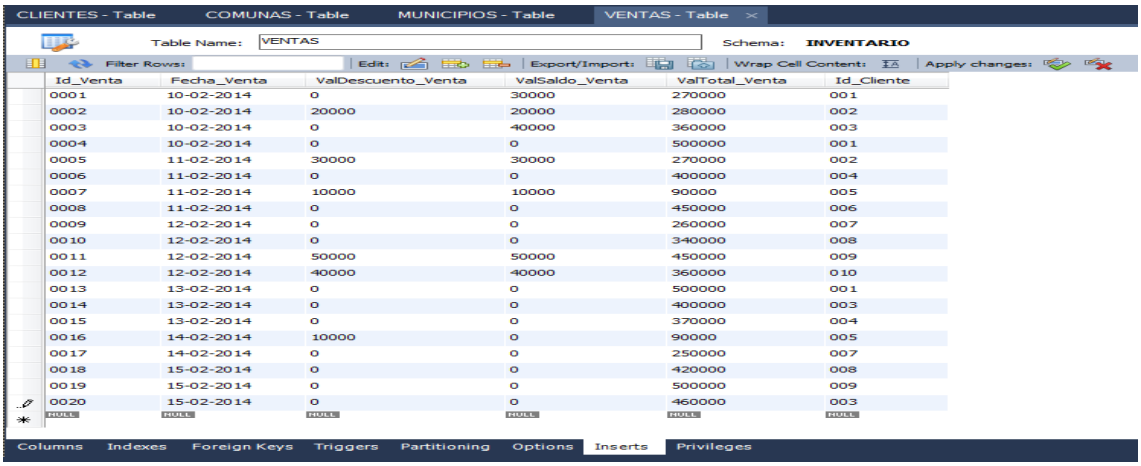
Group By

Ahora regresamos a las funciones de agregados. ¿Recuerda que utilizamos la palabra clave **SUM** para calcular las ventas totales? ¿Y si quisiéramos calcular el total de ventas para *cada* cliente? Entonces, necesitamos hacer dos cosas: Primero, necesitamos asegurarnos de que hayamos **seleccionado** el Id Cliente así como también las ventas totales. Segundo, debemos asegurarnos de que todas las sumas de las ventas totales estén **GROUP BY** ValTotal_Venta. La sintaxis SQL correspondiente es,

```
SELECT "nombre1_columna", SUM("nombre2_columna")
FROM "nombre_tabla"
GROUP BY "nombre1-columna";
```

Ilustremos utilizando la siguiente tabla,

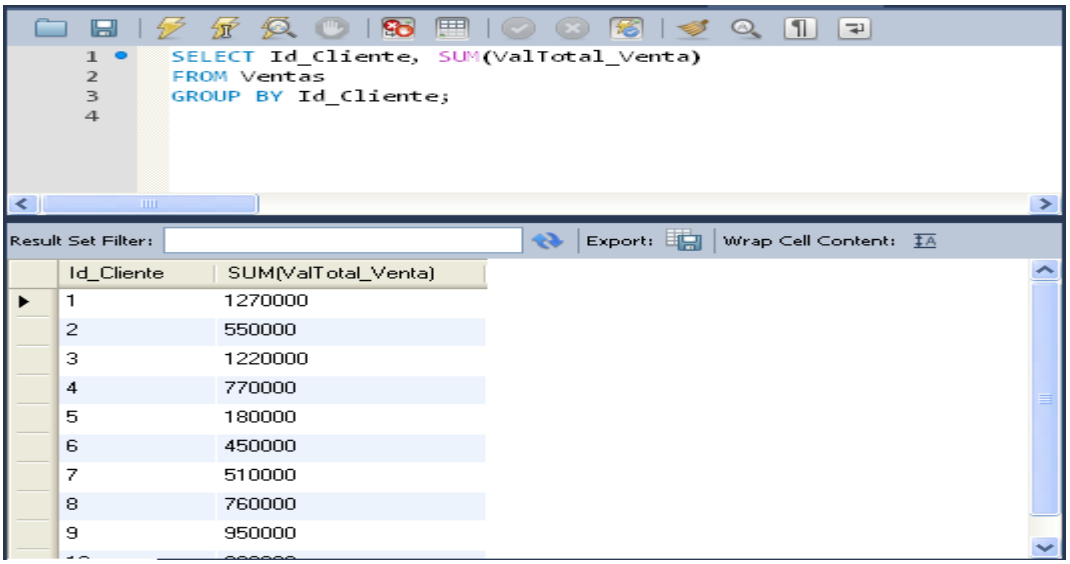
Tabla ***VENTAS***



Deseamos saber las ventas totales para cada cliente. Para hacerlo, ingresaríamos,

```
SELECT Id_Cliente, SUM(ValTotal_Venta)
FROM Ventas
GROUP BY Id_Cliente;
```

Resultado:



La palabra clave **GROUP BY** se utiliza cuando estamos seleccionado columnas múltiples desde una tabla (o tablas) y aparece al menos un operador aritmético en la instrucción **SELECT**. Cuando esto sucede, necesitamos**GROUP BY** todas las otras columnas seleccionadas, *es decir*, todas las columnas excepto aquella(s) que se operan por un operador aritmético.

EJERCICIO 10

CONSULTA 30: de la Tabla **VENTAS**, sumar todas las ventas totales por cada cliente
CONSULTA 31: de la Tabla **VENTAS**, sumar todos los descuentos de ventas por cada cliente
CONSULTA 32: de la Tabla **VENTAS**, sumar todos los saldos de ventas por cada cliente

Having

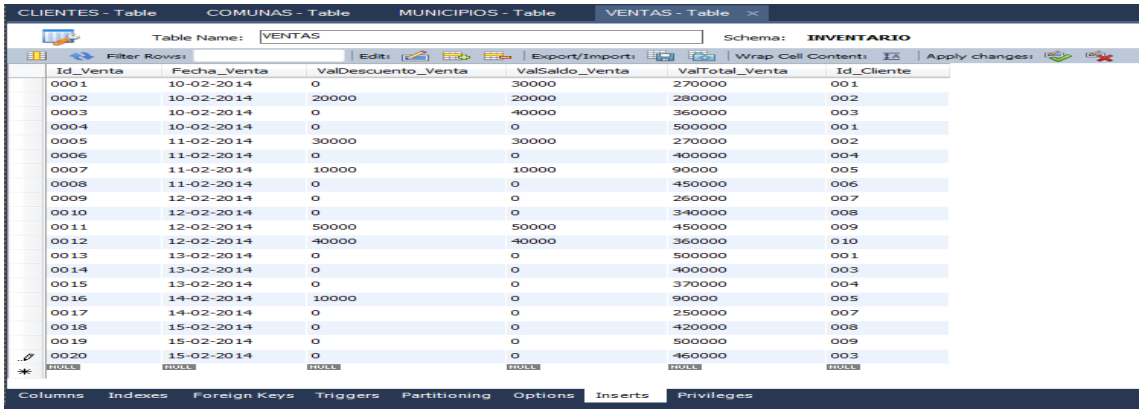
Otra cosa que la gente puede querer hacer es limitar el resultado según la suma correspondiente (o cualquier otra función de agregado). Por ejemplo, podríamos desear ver sólo las ventas totales mayores a 700000, pesos. En vez de utilizar la cláusula **WHERE** en la instrucción SQL, a pesar de que necesitemos utilizar la cláusula **HAVING**, que se reserva para funciones de agregados. La cláusula **HAVING** se coloca generalmente cerca del fin de la instrucción SQL, y la instrucción SQL con la cláusula **HAVING**. puede o no incluir la cláusula**GROUP BY** sintaxis para **HAVING** es,

```
SELECT "nombre1_columna", SUM("nombre2_columna")
FROM "nombre_tabla"
GROUP BY "nombre1_columna"
HAVING (condición de función aritmética);
```

Nota: La cláusula **GROUP BY** es opcional.

En nuestro ejemplo, tabla ***Ventas***,

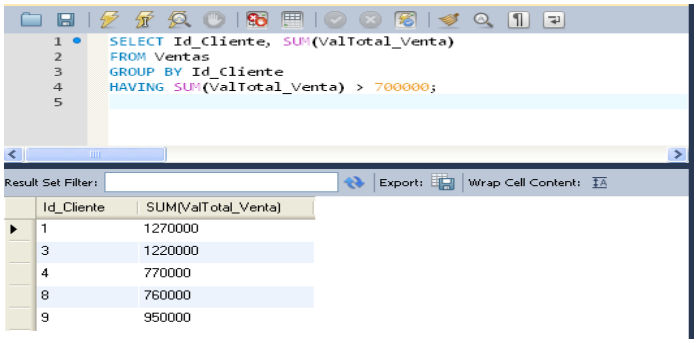
Tabla ***Ventas***



Ingresaríamos,

```
SELECT Id_Cliente, SUM(ValTotal_Venta)
FROM Ventas
GROUP BY Id_Cliente
HAVING SUM(ValTotal_Venta) > 700000;
```

Resultado:



EJERCICIO 11

CONSULTA 33: de la Tabla **VENTAS**, mostrar la suma de todas las ventas totales mayores a 700000 por cada cliente
CONSULTA 34: de la Tabla **VENTAS**, mostrar la suma de todos los descuentos de ventas por cada cliente mayores a 10000
CONSULTA 35: de la Tabla **VENTAS**, mostrar la suma de todos los saldos de ventas por cada cliente mayores a 20000

Alias

Nos concentraremos ahora en el uso de alias. Hay dos tipos de alias que se utilizan con mayor frecuencia. Alias de columna y alias de tabla.

Resumiendo, los alias de columna existen para ayudar en la organización del resultado. En el ejemplo anterior, cualquiera sea el momento en que vemos las ventas totales, se enumeran como SUM(ValTotal_Venta). Mientras esto es comprensible, podemos ver casos donde el título de la columna pueden complicarse (especialmente si incluye varias operaciones aritméticas). El uso de un alias de columna haría el resultado mucho más legible.

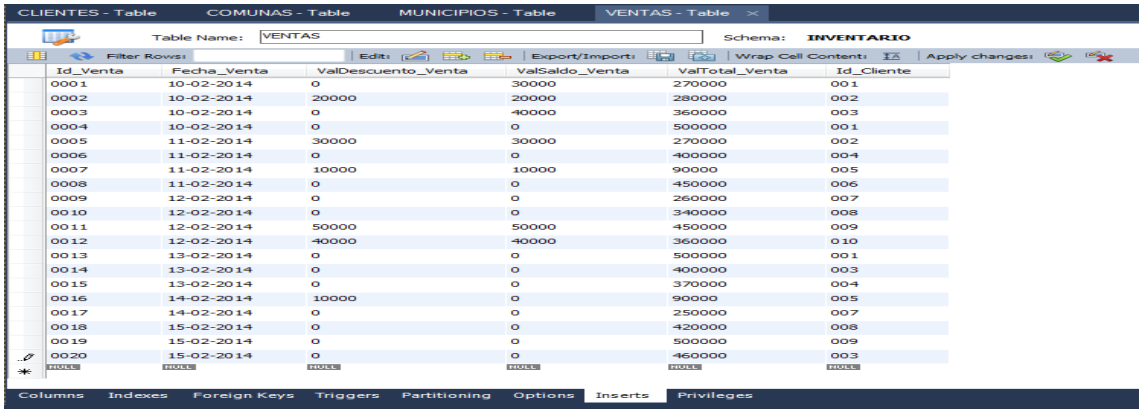
El segundo tipo de alias es el alias de tabla. Esto se alcanza al colocar un alias directamente luego del nombre de tabla en la cláusula **FROM**. Esto es conveniente cuando desea obtener información de dos tablas separadas (el término técnico es 'realizar uniones'). La ventaja de utiliza un alias de tablas cuando realizamos uniones es rápidamente aparente cuando hablamos de uniones.

Antes de comenzar con las uniones, miremos la sintaxis tanto para el alias de columna como de tabla:

```
SELECT "alias_tabla"."nombre1_columna" "alias_columna"
FROM "nombre_tabla" "alias_tabla";
```

Brevemente, ambos tipos de alias se colocan directamente después del elemento por el cual generan el alias, separados por un espacio en blanco. Nuevamente utilizamos nuestra tabla, ***VENTAS***,

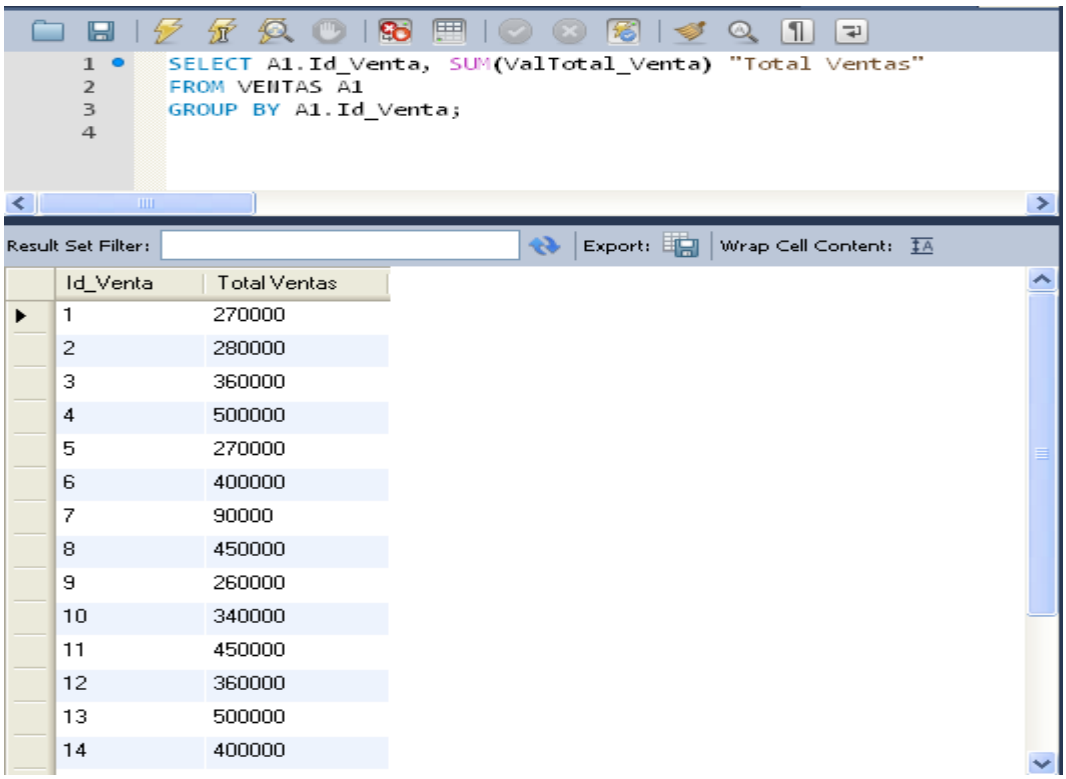
Tabla ***VENTAS***



Utilizamos el mismo ejemplo que en la sección **SQL GROUP BY**, salvo que hemos colocado tanto el alias de columna como el alias de tabla:

```
SELECT A1.Id_Venta, SUM(ValTotal_Venta) "Total Ventas"
FROM VENTAS A1
GROUP BY A1.Id_Venta;
```

Resultado:



Id_Venta	Total Ventas
1	270000
2	280000
3	360000
4	500000
5	270000
6	400000
7	90000
8	450000
9	260000
10	340000
11	450000
12	360000
13	500000
14	400000

Note la diferencia en el resultado: los títulos de las columnas ahora son diferentes. Ese es el resultado de utilizar el alias de columna. Note que en vez de "Sum(ValTotal_Ventas)" de algún modo enigmático, ahora tenemos "Total Ventas", que es más comprensible, como título de columna. La ventaja de utilizar un alias de tablas no es fácil de ver en este ejemplo.

EJERCICIO 12

CONSULTA 36: de la Tabla **VENTAS**, mostrar la suma de todas las ventas totales mayores a 700000 por cada cliente. Aplicar el alias: "Ventas superiores a 700000" a ventas Totales

CONSULTA 37: de la Tabla **VENTAS**, mostrar la suma de todos los descuentos de ventas por cada cliente mayores a 10000. Aplicar el alias: "Descuentos superiores a 10000" a Descuentos de ventas

CONSULTA 38: de la Tabla **VENTAS**, mostrar la suma de todos los saldos de ventas por cada cliente mayores a 20000. Aplicar el alias: "Saldos de ventas mayores a 20000" a Saldos de ventas.

Concatenar Funcion

Algunas veces es necesario combinar en forma conjunta (concatenar) los resultados de varios campos diferentes. Cada base de datos brinda una forma para realizar esto:

- MySQL: CONCAT( )
- Oracle: CONCAT( ), " "
- SQL Server: +

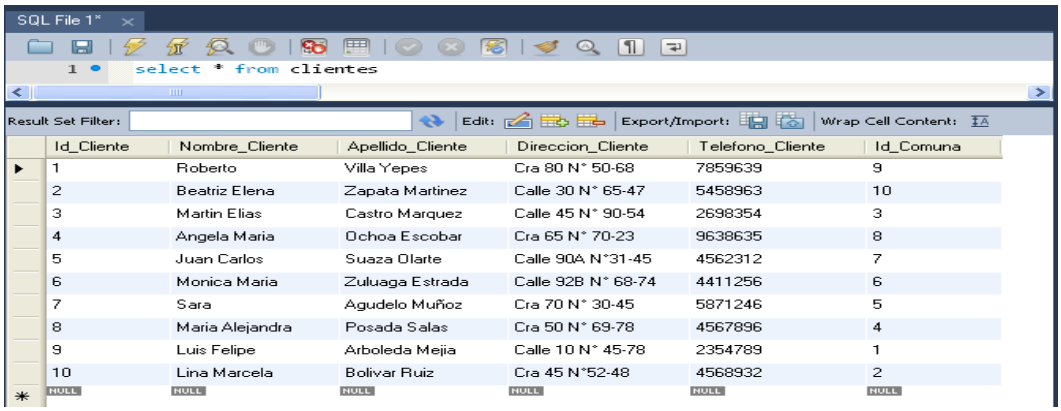
La sintaxis para CONCAT( ) es la siguiente:

CONCAT (cad1, cad2, cad3, ...)

Concatenar cad1, cad2, cad3, y cualquier otra cadena juntas. Por favor note que la función CONCAT( ) de Oracle sólo permite dos argumentos – sólo dos cadenas pueden colocarse juntas al mismo tiempo utilizando esta función. Sin embargo, es posible concatenar más de dos cadenas al mismo tiempo en Oracle utilizando " ".

Observemos algunos ejemplos. Supongamos que tenemos la siguiente tabla:

Tabla ***CLIENTES***



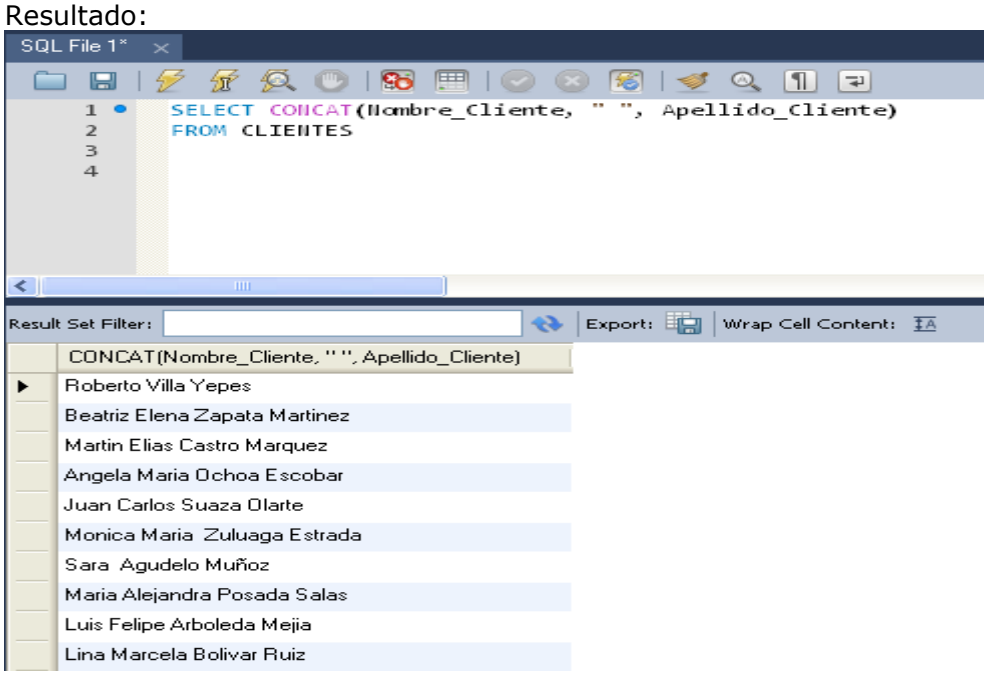
Id_Cliente	Nombre_Cliente	Apellido_Cliente	Direccion_Cliente	Telefono_Cliente	Id_Comuna
1	Roberto	Villa Yepes	Cra 80 N° 50-68	7859639	9
2	Beatriz Elena	Zapata Martinez	Calle 30 N° 65-47	5458963	10
3	Martin Elias	Castro Marquez	Calle 45 N° 90-54	2698354	3
4	Angela Maria	Ochoa Escobar	Cra 65 N° 70-23	9638635	8
5	Juan Carlos	Suaza Olarte	Calle 90A N°31-45	4562312	7
6	Monica Maria	Zuluaga Estrada	Calle 92B N° 68-74	4411256	6
7	Sara	Agudelo Muñoz	Cra 70 N° 30-45	5871246	5
8	Maria Alejandra	Posada Salas	Cra 50 N° 69-78	4567896	4
9	Luis Felipe	Arboleda Mejia	Calle 10 N° 45-78	2354789	1
10	Lina Marcela	Bolivar Ruiz	Cra 45 N°52-48	4568932	2
NULL	NULL	NULL	NULL	NULL	NULL

Ejemplo

MySQL/Oracle:

```
SELECT CONCAT(Nombre_Cliente, " ", Apellido_Cliente)
FROM CLIENTES
```

Resultado:



CONCAT(Nombre_Cliente, " ", Apellido_Cliente)
Roberto Villa Yepes
Beatriz Elena Zapata Martinez
Martin Elias Castro Marquez
Angela Maria Ochoa Escobar
Juan Carlos Suaza Olarte
Monica Maria Zuluaga Estrada
Sara Agudelo Muñoz
Maria Alejandra Posada Salas
Luis Felipe Arboleda Mejia
Lina Marcela Bolivar Ruiz

EJERCICIO 13

CONSULTA 39: de la Tabla **CLIENTES**, Concatenar los campos correspondientes a: Dirección y Teléfono.
CONSULTA 40: de la Tabla **MUNICIPIOS**, Concatenar los campos correspondientes a: el Id_Municipio y el nombre del municipio.
CONSULTA 41: de la Tabla **PROVEEDORES**, Concatenar los campos correspondientes a: el nombre comercial del proveedor, Nombre del proveedor y apellidos.
CONSULTA 42: de la Tabla **INVENTARIO**, Concatenar los campos correspondientes a: el STOCK y el valor de venta

Substring Funcion

La función de subcadena en SQL se utiliza para tomar una parte de los datos almacenados. Esta función tiene diferentes nombres según las diferentes bases de datos:

- MySQL: SUBSTR( ), SUBSTRING( )
- Oracle: SUBSTR( )
- SQL Server: SUBSTRING( )

Los usos más frecuentes son los siguientes (utilizaremos SUBSTR( ) aquí):

SUBSTR (str, pos)

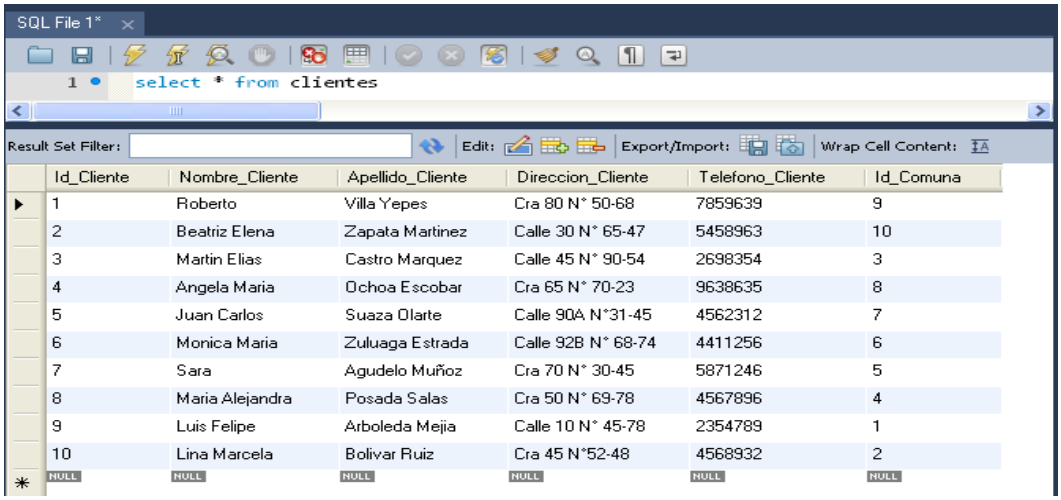
Selecciona todos los caracteres de <str> comenzando con posición <pos>.

SUBSTR (str, pos, len)

Comienza con el carácter <pos> en la cadena <str> y selecciona los siguientes caracteres <len>.

Supongamos que tenemos la siguiente tabla:

Tabla ***CLIENTES***

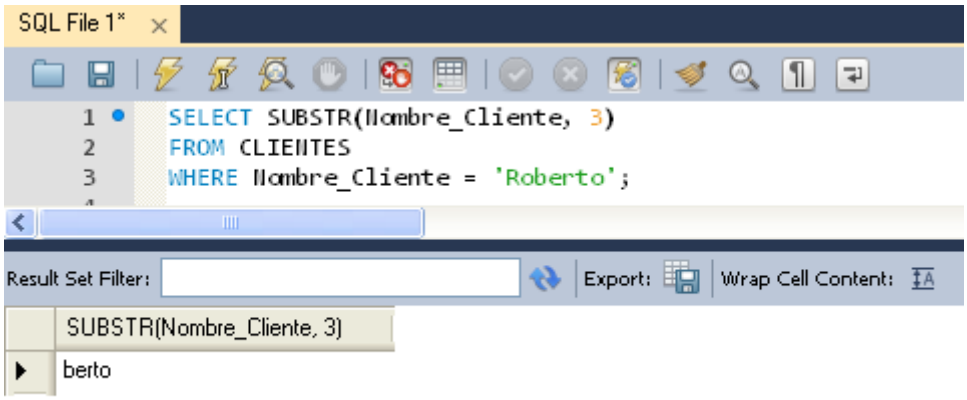


Id_Cliente	Nombre_Cliente	Apellido_Cliente	Direccion_Cliente	Telefono_Cliente	Id_Comuna
1	Roberto	Villa Yepes	Cra 80 N° 50-68	7859639	9
2	Beatriz Elena	Zapata Martinez	Calle 30 N° 65-47	5458963	10
3	Martin Elias	Castro Marquez	Calle 45 N° 90-54	2698354	3
4	Angela Maria	Ochoa Escobar	Cra 65 N° 70-23	9638635	8
5	Juan Carlos	Suaza Olarte	Calle 90A N°31-45	4562312	7
6	Monica Maria	Zuluaga Estrada	Calle 92B N° 68-74	4411256	6
7	Sara	Agudelo Muñoz	Cra 70 N° 30-45	5871246	5
8	Maria Alejandra	Posada Salas	Cra 50 N° 69-78	4567896	4
9	Luis Felipe	Arboleda Mejia	Calle 10 N° 45-78	2354789	1
10	Lina Marcela	Bolivar Ruiz	Cra 45 N°52-48	4568932	2
NULL	NULL	NULL	NULL	NULL	NULL

Ejemplo

```
SELECT SUBSTR(Nombre_Cliente, 3)
FROM CLIENTES
WHERE Nombre_Cliente = 'Roberto';
```

Resultado:



EJERCICIO 13

CONSULTA 43: de la Tabla **CLIENTES**, hacer un Substring con los tres primeros caracteres del nombre de cliente **Beatriz**
CONSULTA 44: de la Tabla **MUNICIPIOS**, hacer un Substring con los cuatro primeros caracteres del nombre de municipio Medellín.
CONSULTA 45: de la Tabla **PROVEEDORES**, hacer un Substring con los dos primeros caracteres del nombre comercial del proveedor SurtiMerca S.A.

ACTIVDAD COMPLEMENTARIA

1. En forma individual realizar un mapa conceptual a través del aplicativo Cmap Tools. Sobre el concepto de MATRIZ CRUD.

2. Consultar el procedimiento que se debe seguir para conectar una base de datos en MySQL Workbench con el lenguaje de programación CSHARP o C#.(En forma individual hacer una descripción del procedimiento o los pasos a seguir. Esta se puede hacer con el uso de imágenes y graficas).

Enviar dicha actividad al correo yadiralexanderdurango@gmail.com