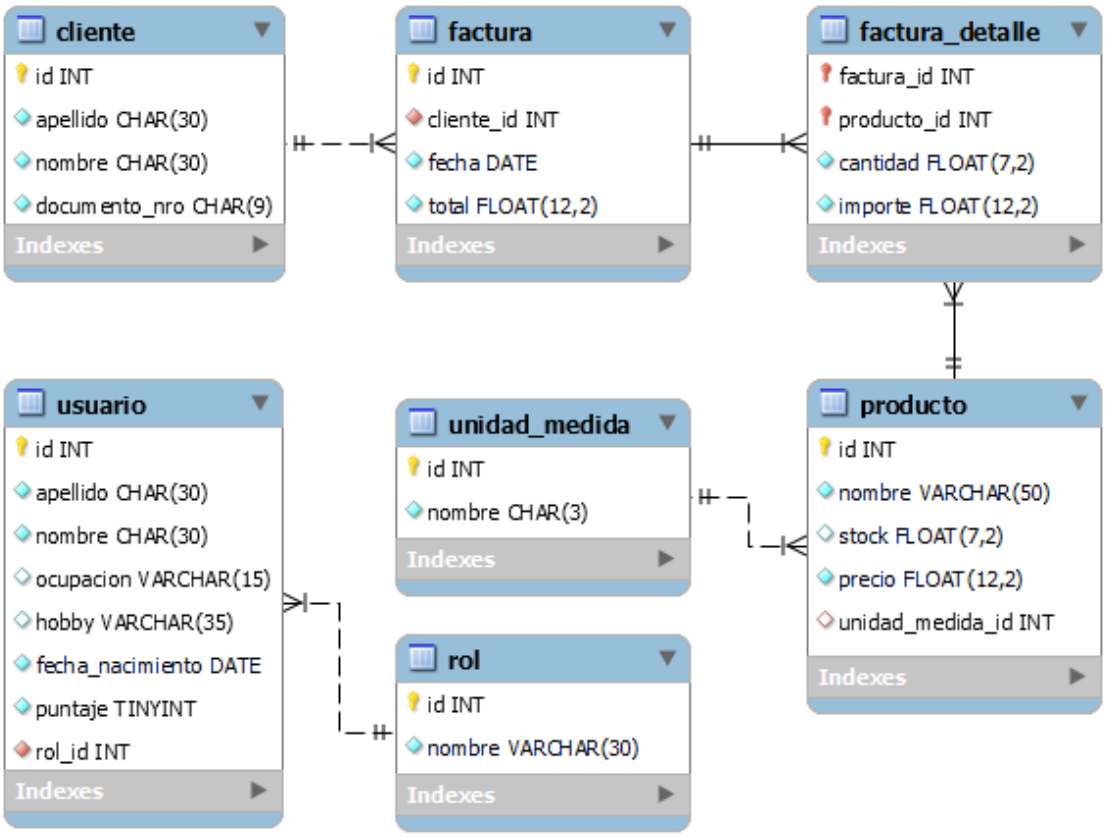
EJERCITACIÓN ADICIONAL (opcional)

TEMAS:

- Tipos de JOINS (Inner, Left, Left Excluding, Right, Right Excluding)
- Agrupamiento de datos (Group By)
- Funciones de alteración
- Funciones de agregación
- Ordenamiento de datos (Order By)
- Desde **aquí** puedes descargar la base de datos



1. Reportar los productos (nombre en mayúsculas), stock, unidad de medida y precio (agregar el signo de la moneda). Ordenarlos por nombre de producto.

```
SELECT UPPER(p.nombre) AS Producto, CONCAT(stock, " ", um.nombre) AS
Stock, CONCAT("$", precio) AS Precio FROM producto p
INNER JOIN unidad_medida um ON um.id = p.unidad_medida_id
ORDER BY Producto;
```

Explicación: Para mostrar los datos de los productos y unidades de medida, se requirió de INNER JOIN (producto/unidad_medida). Se utilizó UPPER para colocar el nombre del producto en mayúsculas; CONCAT para unir la columna stock y el nombre de la unidad de medida, lo mismo para colocar el signo peso al precio. Por último, se ordenó ascendentemente por el nombre del producto.

Experimentemos:

a) *Modifiquemos el tipo de JOIN en la consulta, primero por LEFT JOIN y luego por RIGHT JOIN. En cada modificación, ejecutamos la consulta, observamos y analizamos detenidamente los resultados obtenidos.*
b) *Ahora, volvamos a modificar la consulta incluyendo la técnica de Excluding LEFT JOIN cómo RIGHT JOIN. En cada modificación, ejecutamos la consulta, observamos y analizamos detenidamente los resultados obtenidos.*

2.1 Reportar los productos (nombre en minúsculas), stock y las unidades de medidas sin importar si están o no asignadas a algún producto. Mostrar la leyenda –*Sin Asignación*- para los productos que figuren como nulos. Ordenarlos por el nombre de la unidad de medida.

```
SELECT COALESCE(LOWER(p.nombre), '-Sin Asignación-') AS Producto,
stock
        AS Stock, um.nombre AS 'Unidad_de_medida' FROM producto p
RIGHT JOIN unidad_medida um ON um.id = p.unidad_medida_id
ORDER BY um.nombre;
```

Explicación: Para mostrar los datos de los productos y las unidades de medida, incluyendo aquellas que no cumplen con la condición del ON, se requirió de RIGHT JOIN (producto/unidad_medida). Se utilizó COALESCE para evaluar los registros cuyo nombre del producto es nulo y, LOWER para colocar dicho nombre en minúsculas. Por último, se ordenó ascendentemente por el nombre del producto.

Experimentemos:

a) *Modifiquemos el tipo de JOIN en la consulta, primero por LEFT JOIN y luego por INNER JOIN. En cada modificación, ejecutamos la consulta, observamos y analizamos detenidamente los resultados obtenidos.*
b) *Ahora, volvemos a modificar la consulta incluyendo la técnica de Excluding LEFT JOIN cómo RIGHT JOIN. En cada modificación, ejecutamos la consulta, observamos y analizamos detenidamente los resultados obtenidos.*

2.2 Modificar la consulta (2.1) para que muestre solamente los registros donde las unidades de medida aún no fueron asignadas a algún producto.

```
SELECT COALESCE(LOWER(p.nombre), '-Sin Asignación-') AS Producto, stock
    AS Stock, um.nombre AS 'Unidad_de_medida' FROM producto p
RIGHT JOIN unidad_medida um ON um.id = p.unidad_medida_id
WHERE p.id IS NULL ORDER BY um.nombre;
```

Explicación: Para mostrar solamente los registros de los productos y las unidades de medida que no cumplen con la condición del ON, se requirió de RIGHT JOIN (producto/unidad_medida) y del WHERE p.id IS NULL (Técnica de Excluding) para excluir todos los productos que si tienen asignada una unidad de medida. Luego, se utilizó COALESCE para evaluar los registros cuyo nombre del producto es nulo y, LOWER para colocar dicho nombre en minúsculas. Por último, se ordenó ascendentemente por el nombre del producto.

3.1 Listar todas las ventas mostrando el número de factura, nombre del producto, cantidad, importe y total facturado. Ordenarlo por número de factura.

```
SELECT factura_id AS Factura, p.nombre AS Producto, cantidad, importe, total
FROM factura f
INNER JOIN factura_detalle fd ON fd.factura_id = f.id
INNER JOIN producto p ON p.id = fd.producto_id
ORDER BY Factura;
```

Explicación: Para mostrar los datos de las ventas y los productos que contienen en el detalle de cada factura, se requirió de dos INNER JOIN (factura/factura_detalle/producto). Por último, se ordenó ascendentemente por el número de factura.

3.2 Modificar la consulta (3.1) para que muestre todas las ventas incluyendo a aquellos productos que aún no han sido vendidos. Finalmente, ordenarlo por número de Factura.

```
SELECT factura_id AS Factura, p.nombre AS Producto, cantidad, importe, total
FROM factura f
INNER JOIN factura_detalle fd ON fd.factura_id = f.id
RIGHT JOIN producto p ON p.id = fd.producto_id
ORDER BY Factura;
```

Explicación: Para mostrar los datos de las ventas y los productos que contienen en el detalle de cada factura y, los productos que aún no han sido vendidos, se utilizó la combinación de INNER JOIN (factura/factura_detalle) y RIGHT JOIN (factura_detalle/producto). Esto último, muestra los productos que cumplen o no con la condición del ON. Por último, se ordenó ascendentemente por el número de factura.

Experimentemos:

a) *Modifiquemos el tipo de JOIN para cada tabla de la consulta, primero por LEFT JOIN y luego por RIGHT JOIN. Experimentemos combinaciones. En cada modificación, ejecutamos la consulta, observamos y analizamos detenidamente los resultados obtenidos.*
b) *En la consulta original (3.1), modifiquemos lo necesario para agrupar por número factura, ejecute la consulta y observe los resultados. Luego, intentemos lo mismo, pero agrupemos por nombre del producto.*
c) *En la consulta original (3.1), modifiquemos lo necesario para agrupar por unidad de medida.*

4.1 Listar todas las ventas mostrando el número de factura, fecha, nombre completo del cliente, nombre del producto, cantidad, unidad de medida, importe y total facturado. Ordenarlo por el número de factura de mayor a menor.

```
SELECT factura_id AS Factura, fecha, CONCAT(c.apellido, " ", c.nombre)
        AS Cliente, p.nombre AS Producto, cantidad, um.nombre AS Medida,
        importe, total FROM cliente c
INNER JOIN factura f ON c.id = f.cliente_id
INNER JOIN factura_detalle fd ON fd.factura_id = f.id
INNER JOIN producto p ON p.id = fd.producto_id
```

```
INNER JOIN unidad_medida um ON um.id = p.unidad_medida_id
ORDER BY Factura DESC;
```

Explicación: Para mostrar los datos de las ventas a clientes y los productos que contienen en el detalle de cada factura y, las unidades de medida de cada producto, se utilizó la combinación de INNER JOIN (cliente/factura/factura_detalle/producto/unidad_medida). Por último, se ordenó descendentemente por el número de factura.

4.2 Modificar la consulta (4.1) para que muestre todas las ventas incluyendo a aquellos productos que aún no han sido vendidos. Finalmente, ordenarlo por número de factura.

```
SELECT factura_id AS Factura, fecha, CONCAT(c.apellido, " ", c.nombre)
        AS Cliente, p.nombre AS Producto, cantidad, um.nombre AS Medida,
        importe, total FROM cliente c
INNER JOIN factura f ON c.id = f.cliente_id
INNER JOIN factura_detalle fd ON fd.factura_id = f.id
RIGHT JOIN producto p ON p.id = fd.producto_id
INNER JOIN unidad_medida um ON um.id = p.unidad_medida_id
ORDER BY Factura DESC;
```

Explicación: Para mostrar los datos de las ventas a clientes y los productos que contienen en el detalle de cada factura y, las unidades de medida de cada producto, incluyendo aquellos productos que aún no se han incluido en el detalle de alguna factura. Se utilizó la combinación de INNER JOIN (cliente/factura/factura_detalle), RIGHT JOIN (factura_detalle/producto) e INNER JOIN (producto/unidad_medida). Por medio de RIGHT, se muestran los productos que cumplen o no con la condición del ON. Por último, se ordenó descendentemente por el número de factura.

Experimentemos:

a) *Intentemos conseguir los mismos resultados de esta consulta utilizando LEFT JOIN. (Es muy probable que tengamos que cambiar el orden de las tablas dentro del código de la consulta).*

4.3 Modificar la consulta (4.1) para que muestre todas las ventas incluyendo a aquellos clientes que aún no han realizado alguna compra. Finalmente, ordenarlo por número de factura.

```
SELECT factura_id AS Factura, fecha, CONCAT(c.apellido, " ", c.nombre)
```

```
AS Cliente, p.nombre AS Producto, cantidad, um.nombre AS Medida,
importe, total FROM cliente c
LEFT JOIN factura f ON c.id = f.cliente_id
LEFT JOIN factura_detalle fd ON fd.factura_id = f.id
LEFT JOIN producto p ON p.id = fd.producto_id
LEFT JOIN unidad_medida um ON um.id = p.unidad_medida_id
ORDER BY Factura DESC;
```

Explicación: Para mostrar los datos de las ventas a clientes y los productos que contienen en el detalle de cada factura y, las unidades de medida de cada producto, incluyendo aquellos clientes que aún no has realizado alguna compra. Se requirió solamente del uso de LEFT JOIN para todas las tablas (cliente/factura/factura_detalle/producto/unidad_medida). Nótese la importancia de la repetición de la cláusula LEFT en cada JOIN. Por medio de LEFT, es que se logra mostrar los clientes que cumplen o no con la condición del ON. Por último, se ordenó descendentemente por el número de factura.

Experimentemos:

b) *Intentemos conseguir los mismos resultados de esta consulta utilizando RIGHT JOIN. (Es muy probable que tengamos que cambiar el orden de las tablas dentro del código de la consulta).*

4.4 Modificar la consulta (4.1) para que muestre todas las ventas incluyendo a aquellos productos que aún no han sido vendidos y las unidades de medidas que aún no han sido asignadas a algún producto.

```
SELECT factura_id AS Factura, fecha, CONCAT(c.apellido, " ", c.nombre) AS
Cliente, p.nombre AS Producto, cantidad, um.nombre AS Medida, importe, total
FROM cliente c
INNER JOIN factura f ON c.id = f.cliente_id
INNER JOIN factura_detalle fd ON fd.factura_id = f.id
RIGHT JOIN producto p ON p.id = fd.producto_id
RIGHT JOIN unidad_medida um ON um.id = p.unidad_medida_id
ORDER BY Factura DESC;
```

Explicación: Para mostrar los datos de las ventas a clientes y los productos que contienen en el detalle de cada factura y, las unidades de medida de cada producto, incluyendo aquellos productos y unidades de medida que aún no tienen una asignación. Se requirió de la combinación de INNER JOIN (cliente/factura/factura_detalle), RIGTH JOIN (factura_detalle/producto/unidad_medida). Por medio de RIGHT, es que se logra mostrar tanto

los productos como las unidades de medida que cumplen o no con la condición del ON. Por último, se ordenó descendentemente por el número de factura.

Experimentemos:

c) *Intentemos conseguir los mismos resultados de esta consulta utilizando LEFT JOIN. (Es muy probable que tengamos que cambiar el orden de las tablas dentro del código de la consulta).*

5.1 Reportar los productos y cantidades que ha comprado cada cliente (id y nombre completo). Ordenar por cliente en forma descendente y por producto en forma ascendente.

```
SELECT c.id, CONCAT(c.apellido, " ", c.nombre) AS cliente, p.nombre AS
producto, SUM(fd.cantidad) AS 'Cantidad_de_productos' FROM cliente c
INNER JOIN factura f ON c.id = f.cliente_id
INNER JOIN factura_detalle fd ON fd.factura_id = f.id
INNER JOIN producto p ON p.id = fd.producto_id
GROUP BY c.id, cliente, producto
ORDER BY Cliente DESC, Producto;
```

Explicación: Para mostrar los datos de los productos comprados por clientes, se requirió del uso de INNER JOIN (cliente/factura/factura_detalle/producto). Nótese la importancia de la utilización de GROUP BY para el uso de columnas propias. Para lograr mostrar la cantidad de productos que cada cliente compró, se utilizó SUM en la columna cantidad de la tabla factura_detalle para sumar la cantidad en cada agrupamiento, el cual se logra por medio de la cláusula GROUP BY (cliente y producto). También, se utilizó CONCAT para unir el apellido y nombre del cliente. Por último, como primer criterio se ordenó descendentemente por el nombre completo del cliente y como segundo criterio, se ordenó por el nombre del producto de forma ascendente.

5.2 Modificar la consulta (5.1) para que incluya aquellos clientes (id y nombre completo) que aún no han generado compras.

```
SELECT c.id, CONCAT(c.apellido, " ", c.nombre) AS cliente, p.nombre AS
producto, SUM(fd.cantidad) AS 'Cantidad_de_productos' FROM cliente c
LEFT JOIN factura f ON c.id = f.cliente_id
LEFT JOIN factura_detalle fd ON fd.factura_id = f.id
LEFT JOIN producto p ON p.id = fd.producto_id
```

```
GROUP BY c.id, cliente, producto
ORDER BY Cliente DESC, Producto;
```

Explicación: Para mostrar los datos de los productos comprados por clientes incluyendo aquellos clientes que aún no han realizado compra alguna, se requirió solamente del uso de LEFT JOIN (cliente/factura/factura_detalle/producto). Nótese la importancia de la utilización de GROUP BY para el uso de columnas propias y la repetición de la cláusula LEFT en cada JOIN. Para lograr mostrar la cantidad de productos que cada cliente compró. Se utilizó SUM en la columna cantidad de la tabla factura_detalle para sumar la cantidad en cada agrupamiento, el cual se logra por medio de la cláusula GROUP BY (cliente y producto). En este caso, por medio de LEFT, se logra incluir los registros de aquellos clientes que aún no han adquirido algún producto. También, se utilizó CONCAT para unir el apellido y nombre del cliente. Por último, como primer criterio se ordenó descendentemente por el nombre completo del cliente y como segundo criterio, se ordenó por el nombre del producto de forma ascendente.

Experimentemos:

a) *Intentemos conseguir los mismos resultados de esta consulta utilizando RIGHT JOIN. (Es muy probable que tengamos que cambiar el orden de las tablas dentro del código de la consulta).*

6.1 Reportar las cantidades vendidas y las recaudaciones totales discriminadas por productos y ordenar los resultados por el nombre del producto.

```
SELECT p.nombre AS Producto, SUM(fd.cantidad) AS 'Cantidad_vendida ',
        SUM(fd.importe) AS 'Total_recaudado' FROM cliente c
INNER JOIN factura f ON c.id = f.cliente_id
INNER JOIN factura_detalle fd ON fd.factura_id = f.id
INNER JOIN producto p ON p.id = fd.producto_id
GROUP BY Producto;
```

Explicación: Para mostrar los datos de las cantidades vendidas y las recaudaciones totales discriminadas por productos, se requirió del uso de INNER JOIN (cliente/factura/factura_detalle/producto). Para lograr mostrar la cantidad de productos vendidos, se utilizó SUM en la columna importe de la tabla factura_detalle para que sume dichas cantidades en cada agrupamiento. Para calcular la recaudación obtenida para cada producto, se utilizó la función SUM para sumar los importes en dichos agrupamientos. Para lograr agrupar por producto, se empleó la cláusula GROUP BY (producto).

6.2 Modificar la consulta (6.1) para que incluya aquellos productos que aún no han generado ventas.

```
SELECT p.nombre AS Producto, SUM(fd.cantidad) AS 'Cantidad_vendida ',
   SUM(fd.importe) AS 'Total_recaudado' FROM cliente c
INNER JOIN factura f ON c.id = f.cliente_id
INNER JOIN factura_detalle fd ON fd.factura_id = f.id
RIGHT JOIN producto p ON p.id = fd.producto_id
GROUP BY Producto;
```

Explicación: Para mostrar los datos de las cantidades vendidas y las recaudaciones totales discriminadas por productos y, aquellos registros de productos que aún no han sido vendidos. Se requirió de la combinación de INNER JOIN (cliente/factura/factura_detalle) y de RIGHT JOIN (factura_detalle/producto). Dicho RIGHT, muestra aquellos productos que aún no han sido incluidos en algún detalle de factura. Para lograr mostrar la cantidad de productos vendidos, se utilizó COUNT en la tabla factura_detalle que cuenta la cantidad de registros (productos) en cada agrupamiento. Para calcular la recaudación obtenida para cada producto, se utilizó la función SUM para sumar los importes en dichos agrupamientos. Finalmente, para lograr agrupar por producto, se empleó la cláusula GROUP BY (producto).

Experimentemos:

a) *Intentemos conseguir los mismos resultados de esta consulta utilizando LEFT JOIN. (Es muy probable que tengamos que cambiar el orden de las tablas dentro del código de la consulta).*

7.1 Reportar las ventas discriminando por fecha y cliente. Mostrar la fecha, nombre completo del cliente, cantidad de compras realizadas por día, recaudación total y la recaudación promedio. Finalmente, ordenar por la fecha de facturación y cliente.

```
SELECT fecha, CONCAT(c.apellido, " ", c.nombre) AS Cliente, COUNT(f.id) AS
      'Cantidad_de_compras', SUM(total) AS 'Recaudacion_total', AVG(total)
      AS 'Recaudacion_promedio' FROM cliente c
INNER JOIN factura f ON c.id = f.cliente_id
GROUP BY f.fecha, Cliente
ORDER BY f.fecha, Cliente;
```

Explicación: Para mostrar los datos de las ventas discriminando por fecha y cliente. Se requirió del uso de INNER JOIN (cliente/factura). Para lograr mostrar la cantidad de ventas realizadas, se utilizó COUNT en la tabla factura que cuenta la cantidad de registros (facturas) en cada

agrupamiento. Para calcular la recaudación obtenida para cada venta, se utilizó la función SUM para sumar los totales en dichos agrupamientos y, para calcular el promedio por agrupamiento, se usó la función AVG en la columna total de la tabla factura. Finalmente, para lograr agrupar por fecha y cliente, se empleó la cláusula GROUP BY (fecha, cliente). Por último, se ordenó ascendentemente por fecha como primer criterio y, como segundo criterio el nombre completo del cliente.

8.1 Listar la cantidad de productos comprados por cada cliente. Se debe mostrar el número de documento y el nombre completo del cliente y, la cantidad, el menor y mayor precio de los productos.

```
SELECT documento_nro AS numero_documento, CONCAT(c.apellido, " ", c.nombre) AS cliente, COUNT(p.precio) AS cantidad, MIN(p.precio) AS mayor_precio, MAX(p.precio) AS mayor_precio FROM cliente c
INNER JOIN factura f ON f.cliente_id = c.id
INNER JOIN factura_detalle fd ON fd.factura_id = f.id
INNER JOIN producto p ON p.id = fd.producto_id
GROUP BY numero_documento, cliente;
```

Explicación: Además de los JOIN necesarios. Para mostrar los datos correctos de las columnas 'numero_documento' y 'cliente' en conjunto con una o más funciones de agregación, se debe agregar la cláusula GROUP BY y en su cuerpo, incluir todas las columnas declaradas en el cuerpo del SELECT. Se utilizó COUNT para contar los productos comprados por cada cliente, MIN para calcular cuál fué el menor precio y MAX para el mayor precio de tales productos.

Experimentemos: En primer lugar, ejecutar la instrucción:

```
SET sql_mode = 'ONLY_FULL_GROUP_BY';
```

a) *Luego, quitamos una de las columnas declaradas en el cuerpo del GROUP BY y ejecutamos la consulta¿funciona?*

8.2 Listar la cantidad de productos comprados por cada cliente (incluir aquellos que aún no han hayan realizado una compra). Se debe mostrar el número de documento y el nombre completo del cliente y, la cantidad, el menor y mayor precio de los productos.

```
SELECT documento_nro AS numero_documento, CONCAT(c.apellido, " ", c.nombre) AS cliente, COUNT(p.precio) AS cantidad, MIN(p.precio) AS mayor_precio, MAX(p.precio) AS mayor_precio FROM cliente c
LEFT JOIN factura f ON f.cliente_id = c.id
```

```
LEFT JOIN factura_detalle fd ON fd.factura_id = f.id
LEFT JOIN producto p ON p.id = fd.producto_id
GROUP BY numero_documento, cliente;
```

Explicación: En esta variante, se reemplazan los INNER por LEFT para incluir todos los clientes que hayan realizado una compra o no. Nótese que en este caso, se debe reemplazar todos los JOIN subsecuentes por LEFT para evitar pisar el primer LEFT. Se utilizó COUNT para contar los productos comprados por cada cliente, MIN para calcular cuál fué el menor precio y MAX para el mayor precio de tales productos.

Experimentemos:

a) *Intentemos conseguir los mismos resultados de esta consulta utilizando RIGHT JOIN. (Es muy probable que tengamos que cambiar el orden de las tablas dentro del código de la consulta).*
b) *Intentemos aplicar la técnica de RIGHT EXCLUDING para mostrar aquellos clientes que aún no han realizado compras.*

9. Listar los productos y clasificar su stock. Mostrar el id y nombre del producto, el stock y su clasificación. Se debe clasificar como 'Bajo' cuando el stock es menor o igual a 5, 'Medio' cuando se encuentra entre 6 y 25, 'Alto' cuando es mayor a 25, y, 'Sin Clasificación' para algún otro rango.

```
SELECT id, nombre AS producto,
 CASE
 WHEN stock <= 5 THEN 'Baja'
 WHEN stock BETWEEN 6 AND 25 THEN 'Media'
 WHEN stock > 25 THEN 'Alta'
 ELSE 'Sin Clasificación'
 END AS clasificacion FROM producto;
```

Explicación: Para poder crear una clasificación de del stock, se debe utilizar CASE y se tiene que estar atento a los límites definidos en el WHEN. .

Experimentemos:

a) *Intentemos modificar los límites en la definición de los WHEN, como también, agregar o quitar un WHEN.*