* Parte 1: Usando INNER JOIN crear vistas que generen:
  1. Clientes que no tienen pedido facturado

La resolución puede observarlo en la imagen 1. Para ello, se utilizó el RIGHT OUTER JOIN, pues permite que conserve una unión entre dos tablas con una cláusula de unión explícita, preservando las filas no coincidentes de la segunda tabla.

Para ello agregamos el on pedidos.COD\_CLIE = clientes.COD\_CLIE con la finalidad que sea coincidente los códigos de Clientes y los Pedidos, dando como NULL en COD\_CLIE aquellos que han hecho pedidos, pero no son clientes, por ello fue que en la condición WHERE se utilizo el NOT NULL, adicionando que el VAL\_ESTA\_PEDI no debe ser igual a

FACTURADO, razón que se usa el NOT LIKE ‘%FACTURADO%’. Además, piden crear VIEW, por lo que se le añade todo a la sentencia CREATE VIEW CLIENTESINPEDIDOFACT as (query). Tal como se observa en la imagen 1 y el resultado:

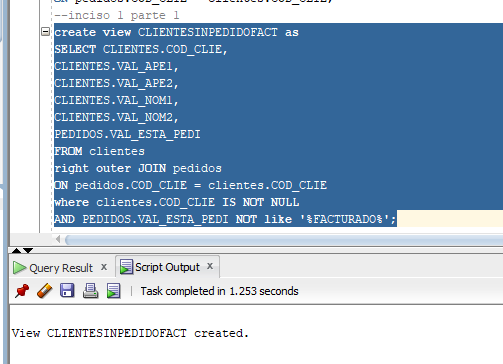


IMAGEN 1: Creación de la vista de la query solicita en el inciso A del reto. Elaboración Propia

Luego, para corroborarlo, se puede ver en la parte izquierda, en el apartado de VIEWS, del cual puede observarse la creación del VIEW CLIENTESINPEDIDOFACT:

A screenshot of a computer

Description automatically generated

IMAGEN 2: VIEW CLIENTESINPEDIDOFACT. Elaboración Propia

Además, para corroborar que hay datos, se hace la siguiente consulta, tal como se aprecia en la IMAGEN 3:

A screenshot of a computer

Description automatically generated

IMAGEN 3: Tabla CLIENTESINPEDIDOFACT. Elaboración Propia

* Usando INNER JOIN crear vistas que generen:

1. Pedidos cuyo cliente no existe en la tabla Clientes

Para ello, se toma la parte de los clientes que no existen en la tabla Clientes, para ello, basándonos en la imagen 4, tomara como nulo los valores de COD\_CLIE que no coincida el on pedidos.COD\_CLIE = clientes.COD\_CLIE de la query de la IMAGEN 5, para ello se debe considerar que clientes.COD\_CLIE sea NULL. Además, en la última imagen mencionada, piden que sea de tipo VIEW, por lo que se hace VIEW PEDIDOCLIE AS

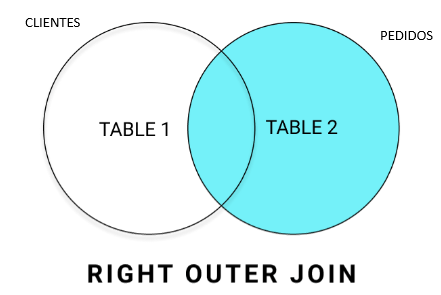


IMAGEN 4: Observación del RIGHT OUTER JOIN. Elaboración Propia

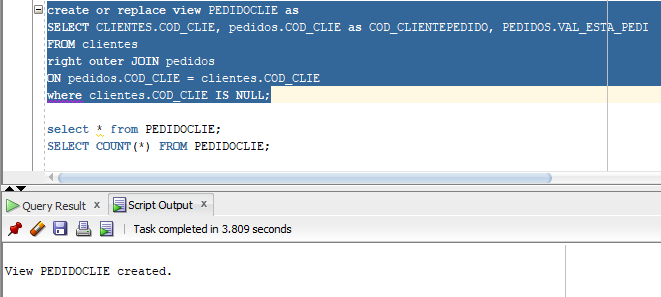


IMAGEN 5: VIEW PEDIDOCLIE. Elaboración Propia

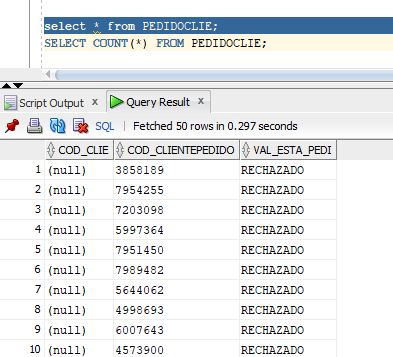


IMAGEN 6: Tabla PEDIDOCLIE. Elaboración Propia

* Parte 2: Crear vistas para mostrar:
  1. Acumulado de atributo VAL\_MONT\_SOLI agrupado por estado de Pedido, Región de aquellos pedidos facturados en junio, considerar para ello que el código de cliente exista en la tabla Cliente

Paso1: Para ello, previamente, debemos realizar la query, considerando el estado de Pedido, Región de aquellos pedidos facturados en junio, considerar para ello que el código de cliente exista en la tabla Cliente y VAL\_MONT\_SOLI. Para el campo MES se transformó el FEC\_FACT usando TO\_CHAR(pedidos.FEC\_FACT,’Month’) con la finalidad de obtener el mes del date FEC\_FACT. Si consideramos que el código de cliente exista en la tabla Cliente, debemos considerar que no sea de tipo NULL. La query es la siguiente:

A screenshot of a computer

Description automatically generated

IMAGEN 7: Query y resultado, considerando que no sea NULL, sea de Junio y que exista el COD\_CLIE

Paso 2: Luego, considerando los campos COD\_REGI, VAL\_ESTA\_PEDI, MES y VAL\_MONT\_SOLI, se realiza la agrupación de estos campos usando group by, considerando que para acumular se usa SUM (VAL\_MONT\_SOLI), tal como se detalla en la imagen 8:

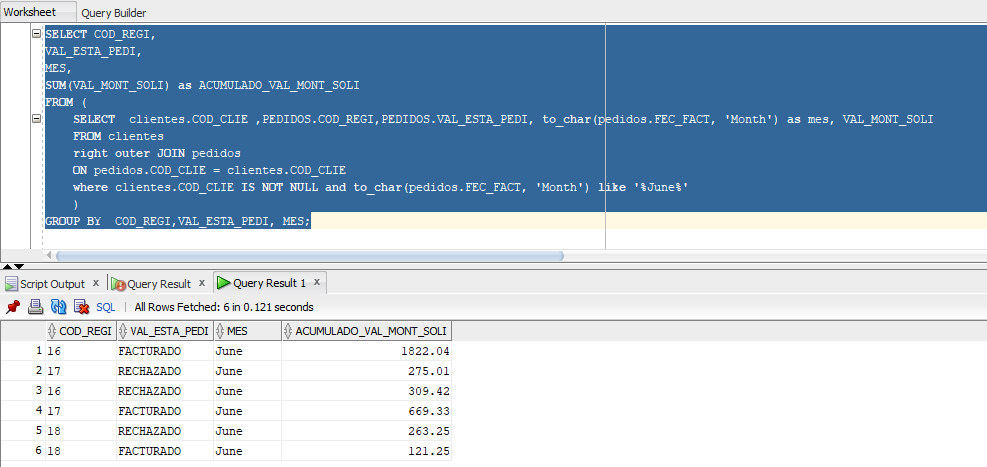


IMAGEN 8: Acumulado de atributo VAL\_MONT\_SOLI agrupado por estado de Pedido, Región de aquellos pedidos facturados en junio, considerar para ello que el código de cliente exista en la tabla Cliente

Paso 3: De acuerdo con la query del paso 1, se realiza el CREATE VIEW LISTACLIENTESJUNIO para reducir el tamaño del código, tal como se observa en la imagen:

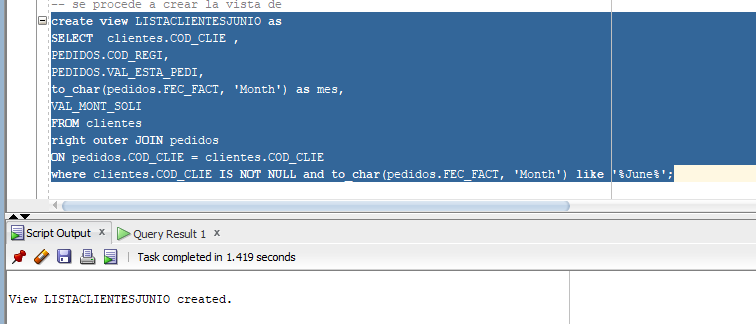


IMAGEN 9: Creación de la VIEW LISTACLIENTESJUNIO basado en la query del paso 1

Paso 4: De acuerdo con la query del paso 2, se realiza el CREATE VIEW AGRUPACIONFINALCLIENTESJUNIO para reducir el tamaño del código, tal como se observa en la imagen:

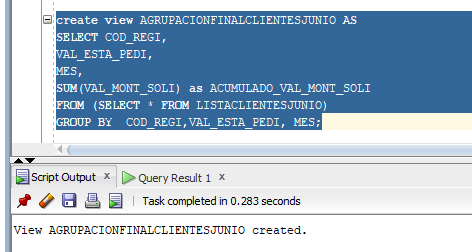


IMAGEN 10: Creación de la VIEW AGRUPACIONFINALCLIENTESJUNIO basado en la query del paso 2

De acuerdo con el problema, piden la VIEW. Para corroborar el VIEW AGRUPACIONFINALCLIENTESJUNIO, se realiza la petición:

A screenshot of a computer

Description automatically generated

IMAGEN 11: Comprobacion de la VIEW pedida AGRUPACIONFINALCLIENTESJUNIO

* 1. En base a la consulta anterior, mostrar una columna adicional que contenga el total de registros por cada agrupación y condicionar a que se muestre solo aquellos que tengan más de 500 registros agrupados

Paso 1: De acuerdo con la query del paso 2 del inciso a, se añade otro campo de conteo de registros, el COUNT(\*), pero adicionamos el HAVING GROUP COUNT(\*) > 500, pues piden que se vea si y solo si tengan mas de 500 registros agrupados. Tal como se observa en la imagen 12:

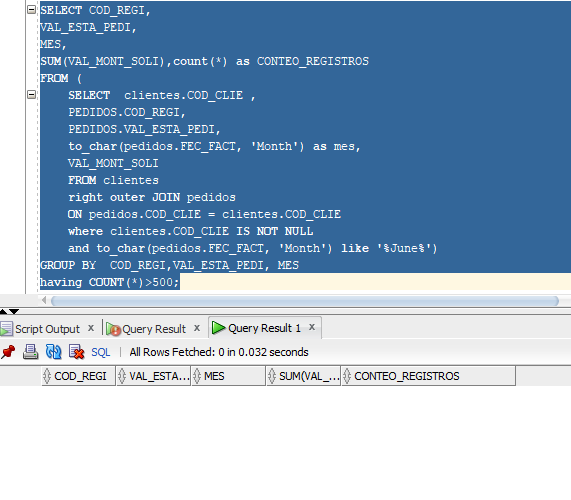


IMAGEN 12

Paso 2: De acuerdo con la query del paso 3 del inciso a, se usa el VIEW LISTACLIENTESJUNIO para reducir el tamaño del código dentro del primer FROM y se hace un CREATE VIEW AGRUPACIONESCLIENTESSUPERIORES, pues piden la vista:

A screenshot of a computer program

Description automatically generated

IMAGEN 12: Creación de la VIEW pedida en el Inciso B

Paso 3: Para comprobar, se realiza el select \* from AGRUPACIONESCLIENTESSUPERIORES.

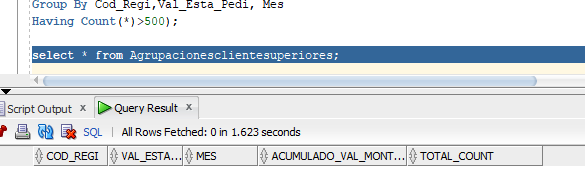


IMAGEN 13: Comprobación de la VIEW pedida en el Inciso B