**Clase 04  
14.**SELECT fname, lname, state, company

FROM customer

WHERE state='CA'

ORDER BY company

**15. (Está bien? Porque es confuso lo que se pide, y esta solución me deja afuera algunos registros de compras de un producto único, pero que en esa única compra no supera los 1500, si se interpreta como que tiene que superar los 1500 en una única compra, está bien supongo, pero si se interpreta que la suma de todas las compras de un único producto tiene que superar 1500 ya no estaría bien. Hace falta un Distinct para que tome los productos como únicos? Creo que al agrupar stock\_num y manu\_name ya son únicos porque constituyen la clave del producto, estaría mal poner el distinct para asegurar que sean únicos siempre en esa query?)**SELECT stock\_num, manu\_code, Sum(quantity) AS cantComprada

FROM items

WHERE quantity\*unit\_price>1500

GROUP BY stock\_num, manu\_code

ORDER BY cantComprada DESC

**16.**SELECT manu\_code, stock\_num, quantity, quantity\*unit\_price AS total\_vendido

FROM items

WHERE manu\_code LIKE '\_R%'

ORDER BY manu\_code, stock\_num

**17.**SELECT customer\_num, Count(\*) AS cantOrdenes, Min(order\_date) AS fechaPrimeraCompra, Max(order\_date) AS fechaUltimaCompra

INTO #OrdenesTemp

FROM orders

GROUP BY customer\_num

SELECT \*

FROM #OrdenesTemp

WHERE fechaPrimeraCompra < '2015-05-23 00:00:00.000'

ORDER BY fechaUltimaCompra DESC

**18. (este fue el de la duda)**SELECT cantOrdenes, Count(\*) AS cantComprasIguales

FROM #OrdenesTemp

GROUP BY cantOrdenes

ORDER BY cantOrdenes DESC

**19.  
Invalid Object name #ordenesTemp  
  
20.**SELECT state, city, Count(\*) AS cantClientes

FROM customer

WHERE company LIKE '%ts%'

AND zipcode > 93000 AND zipcode < 94100

AND city != 'Mountain View'

GROUP BY state, city

ORDER BY city

**21.**SELECT state, Count(DISTINCT customer\_num) clientesReferidos

FROM customer

WHERE company BETWEEN 'A%' AND 'L%'

GROUP BY state

**22.**SELECT state, Avg(lead\_time) clientesReferidos

FROM manufact

WHERE manu\_name LIKE '%e%'

AND lead\_time BETWEEN 5 AND 20

GROUP BY state

**El merge se usa solo para hacer cosas entre 2 tablas preexistentes o tiene alguna utilidad clonar una tabla y mergearla contra la original? Dudas en el 15 y 18.  
  
Clase 05  
  
1.**

SELECT c.customer\_num, company, order\_num

FROM customer c JOIN orders o ON c.customer\_num = o.customer\_num

ORDER BY customer\_num

**2.**

SELECT order\_num, item\_num, description, manu\_code, quantity, unit\_price\*quantity AS precio\_total

FROM items i JOIN product\_types pt ON i.stock\_num = pt.stock\_num

WHERE order\_num = 1004

**3.**

SELECT order\_num, item\_num, description, m.manu\_code, quantity, manu\_name, unit\_price\*quantity AS precio\_total

FROM items i JOIN product\_types pt ON i.stock\_num = pt.stock\_num

JOIN manufact m ON i.manu\_code = m.manu\_code

WHERE order\_num = 1004

**4.**

SELECT order\_num, c.customer\_num, fname, lname, company

FROM orders o JOIN customer c ON c.customer\_num = o.customer\_num

**5.**

SELECT DISTINCT c.customer\_num, fname, lname, company

FROM customer c JOIN orders o ON c.customer\_num = o.customer\_num

**6.**

SELECT manu\_name, p.stock\_num, description, u.unit, p.unit\_price, p.unit\_price\*1.2 AS precio\_junio

FROM product\_types pt JOIN products p ON p.stock\_num = pt.stock\_num

JOIN units u ON u.unit\_code = p.unit\_code

JOIN manufact m ON m.manu\_code = p.manu\_code  
  
  
**7.**

SELECT i.item\_num, description, quantity, unit\_price\*quantity AS precio\_total

FROM orders o JOIN items i ON i.order\_num = o.order\_num

JOIN product\_types pt ON pt.stock\_num = i.stock\_num

WHERE o.order\_num = 1004

**8.**

SELECT DISTINCT manu\_name, lead\_time

FROM items i JOIN products p ON p.stock\_num = i.stock\_num

JOIN manufact m ON m.manu\_code = p.manu\_code

JOIN orders o ON o.order\_num = i.order\_num

WHERE o.customer\_num = 104

**9.**

SELECT o.order\_num, order\_date, item\_num, description, quantity, unit\_price\*quantity AS precio\_total

FROM orders o JOIN items i ON o.order\_num = i.order\_num

JOIN product\_types pt ON pt.stock\_num = i.stock\_num

**10.**SELECT lname + ', ' + fname AS Apellido\_Nombre, '('+SUBSTRING(phone, 1, 3)+') '+SUBSTRING(phone, 5, 8) AS Teléfono

FROM customer

ORDER BY lname, fname

**11.**SELECT ship\_date, lname + ', ' + fname AS apellidoYNombre, COUNT(order\_num)

FROM customer c JOIN orders o ON c.customer\_num = o.customer\_num

JOIN state s ON c.state = s.state

WHERE s.sname = 'California' AND c.zipcode BETWEEN 94000 AND 94100

GROUP BY ship\_date, lname, fname

ORDER BY ship\_date, lname, fname;

**12.**SELECT manu\_name, description, SUM(i.quantity) AS cantidad\_vendida, SUM(i.unit\_price\*quantity) AS total\_vendido

FROM manufact m JOIN items i on m.manu\_code = i.manu\_code

JOIN product\_types pt ON i.stock\_num = pt.stock\_num

JOIN orders o ON o.order\_num = i.order\_num

WHERE m.manu\_code IN ('ANZ', 'HRO', 'HSK', 'SMT')

AND MONTH(o.order\_date) IN (5, 6)

AND YEAR(o.order\_date) = 2015

GROUP BY manu\_name, description

ORDER BY total\_vendido DESC;

**13.**SELECT CAST(YEAR (o.order\_date) AS VARCHAR) +'/'+ CAST(MONTH(o.order\_date) AS VARCHAR) AS Año\_Mes , SUM(quantity) AS Cantidad, SUM (quantity\*unit\_price) AS Monto\_Total

FROM orders o JOIN items i ON i.order\_num = o.order\_num

GROUP BY YEAR(o.order\_date), MONTH(o.order\_date)

ORDER BY Monto\_Total DESC;

**Cosas que no vimos, CAST, SUBSTRING qué onda con eso? Y, hace diferencia en el 13 si pongo o no el cast en el group by? Y si lo uso como string? Es lo mismo, no?  
  
Clase 06  
  
1.**SELECT m.manu\_code, manu\_name, lead\_time, SUM(quantity\*unit\_price) AS monto\_total

FROM manufact m LEFT JOIN items i ON m.manu\_code = i.manu\_code

GROUP BY manu\_name, m.manu\_code, lead\_time

ORDER BY manu\_name;

**2.**SELECT s1.stock\_num, pt.description, s1.manu\_code, s2.manu\_code

FROM products s1 LEFT JOIN products s2 ON (s1.stock\_num=s2.stock\_num AND s1.manu\_code != s2.manu\_code)

JOIN product\_types pt ON (s1.stock\_num = pt.stock\_num)

WHERE s1.manu\_code < s2.manu\_code OR s2.manu\_code IS NULL

ORDER BY 1;

**3.  
a)**

SELECT c.customer\_num, c.fname, c.lname, s.cantidad

FROM customer c JOIN (SELECT customer\_num, Count(order\_num) cantidad FROM orders GROUP BY customer\_num) AS s

ON c.customer\_num = s.customer\_num

WHERE s.cantidad > 1;

**b)**SELECT c.customer\_num, c.fname, c.lname

FROM customer c JOIN (SELECT customer\_num FROM orders GROUP BY customer\_num HAVING Count(order\_num)>1) AS s

ON c.customer\_num = s.customer\_num;

4.  
SELECT i.order\_num, SUM(i.quantity\*i.unit\_price) montoTotal

FROM items i

GROUP BY i.order\_num

HAVING SUM(i.quantity\*i.unit\_price) < (SELECT AVG(unit\_price\*quantity) AS promedio FROM items);

**5.**

SELECT p.manu\_code, manu\_name, p.stock\_num, description, unit\_price

FROM products p JOIN manufact m ON p.manu\_code = m.manu\_code

JOIN product\_types pt ON pt.stock\_num = p.stock\_num

WHERE unit\_price > (SELECT AVG(unit\_price) FROM products GROUP BY manu\_code HAVING manu\_code = p.manu\_code);

**6.**SELECT o.customer\_num, company, o.order\_num, order\_date

FROM orders o JOIN customer c ON o.customer\_num = c.customer\_num

WHERE NOT EXISTS (SELECT description

FROM product\_types pt JOIN items i ON i.stock\_num = pt.stock\_num

WHERE description LIKE '%baseball gloves%' AND o.order\_num = i.order\_num)

ORDER BY company ASC, o.order\_num DESC

**7.**SELECT c.customer\_num, fname, lname

FROM customer c

WHERE NOT EXISTS (SELECT 1 FROM orders o JOIN items i ON o.order\_num = i.order\_num

WHERE manu\_code = 'HSK' AND c.customer\_num = o.customer\_num)  
  
**8.**SELECT c.customer\_num, fname, lname

FROM customer c

WHERE NOT EXISTS(SELECT stock\_num FROM products WHERE manu\_code = 'HSK'

EXCEPT SELECT i.stock\_num FROM items i JOIN orders o ON o.order\_num = i.order\_num WHERE c.customer\_num = o.customer\_num)

**9.**  
SELECT \* FROM products WHERE manu\_code = 'HRO'

UNION

SELECT \* FROM products WHERE stock\_num = 1

**10.**SELECT 1 sortkey, city ciudad, Company 'Compañía'

FROM customer

WHERE city ='Redwood City'

UNION

SELECT 2 sortkey, city ciudad, Company 'Compañía'

FROM customer

WHERE city !='Redwood City'

ORDER BY sortkey, city

**11.**SELECT i.stock\_num, sum(i.quantity) Total

FROM items i

WHERE i.stock\_num in (SELECT TOP 2 i2.stock\_num

FROM items i2

GROUP BY i2.stock\_num

ORDER BY sum(i2.quantity) DESC)

GROUP BY i.stock\_num

UNION

SELECT i.stock\_num, sum(i.quantity) Total

FROM items i

WHERE i.stock\_num in (SELECT TOP 2 i2.stock\_num

FROM items i2

GROUP BY i2.stock\_num

ORDER BY sum(i2.quantity) ASC)

GROUP BY i.stock\_num

ORDER BY 2 DESC

**Respuestas de los pibitos:**-- Ej. 5 SELECT m.manu\_code, manu\_name, p.stock\_num, description, unit\_price FROM manufact m JOIN products p ON p.manu\_code=m.manu\_code JOIN product\_types pt ON p.stock\_num=pt.stock\_num WHERE p.unit\_price > (SELECT AVG(unit\_price) FROM products WHERE manu\_code=m.manu\_code)   
--6  
SELECT o.customer\_num, company, o.order\_num, order\_date FROM orders o JOIN customer c ON c.customer\_num = o.customer\_num WHERE NOT EXISTS ( SELECT i.stock\_num FROM items i join product\_types t on i.stock\_num = t.stock\_num WHERE description LIKE '%baseball gloves%' AND o.order\_num = i.order\_num) order by company, order\_num desc;  
  
--7  
SELECT c.customer\_num, fname, lname FROM customer c where not exists (select 1 from orders o join items i on o.order\_num = i.order\_num where i.manu\_code = 'HSK' and o.customer\_num = c.customer\_num) order by 1;  
---------OTRA OPCION-----------  
SELECT c.customer\_num, fname, lname FROM customer c where c.customer\_num not in (select o.customer\_num from orders o join items i on o.order\_num = i.order\_num where i.manu\_code = 'HSK') order by 1;  
  
--8  
SELECT c.customer\_num, c.fname, C.lname FROM customer C where not exists ( SELECT p.stock\_num from products p where manu\_code = 'HSK' AND NOT EXISTS (Select 1 from orders o join items i on o.order\_num = i.order\_num where P.stock\_num = i.stock\_num and p.manu\_code = i.manu\_code and o.customer\_num = c.customer\_num))  
  
  
**CLASE 08  
  
1.**

CREATE PROCEDURE actualizaEstadisticas

@customer\_numDES INT , @customer\_numHAS INT

AS

BEGIN

DECLARE CustomerCursor CURSOR FOR

SELECT customer\_num from customer WHERE customer\_num

BETWEEN @customer\_numDES AND @customer\_numHAS

DECLARE @customer\_num INT, @ordersqty INT, @maxdate DATETIME,

@uniqueManufact INT;

OPEN CustomerCursor;

FETCH NEXT FROM CustomerCursor INTO @customer\_num

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT @ordersqty=count(\*) , @maxDate=max(order\_date)

FROM orders

WHERE customer\_num = @customer\_num;

SELECT @uniqueManufact=count(distinct stock\_num)

FROM items i, orders o

WHERE o.customer\_num = @customer\_num

AND o.order\_num = i.order\_num;

IF NOT EXISTS( SELECT 1 FROM CustomerStatistics

WHERE customer\_num = @customer\_num)

insert into customerStatistics

values (@customer\_num,@ordersQty, @maxDate,@uniqueManufact);

ELSE

update customerStatistics

SET ordersQty=@ordersQty,maxDate=@maxDate,

uniqueManufact= @uniqueManufact

WHERE customer\_num = @customer\_num;

FETCH NEXT FROM CustomerCursor INTO @customer\_num

END;

CLOSE CustomerCursor;

DEALLOCATE CustomerCursor;

END

**Clase 09.  
  
a.**CREATE TABLE CustomerStatistics(

customer\_num int PRIMARY KEY,

ordersQty int,

maxDate date,

productsQty int

)

GO

CREATE PROCEDURE CustomerStatisticsUpdate (@fecha\_DES date)

AS

DECLARE CustomerCursor CURSOR FOR

SELECT customer\_num from customer;

DECLARE @customer\_num int, @ordersQty int, @productsQty int, @maxDate date

OPEN CustomerCursor;

FETCH NEXT INTO @customer\_num

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT @ordersQty = Count(\*), @maxDate = MAX(order\_date)

FROM orders WHERE customer\_num = @customer\_num AND order\_date >= @fecha\_DES GROUP BY customer\_num

SELECT @productsQty = Count(\*)

FROM (SELECT DISTINCT stock\_num, manu\_code FROM items i JOIN orders o ON i.order\_num = o.order\_num

WHERE customer\_num = @customer\_num) productos\_unicos;

IF NOT EXISTS( SELECT 1 FROM CustomerStatistics

WHERE customer\_num = @customer\_num)

INSERT into CustomerStatistics (customer\_num, ordersqty,

maxdate, productsQty)

VALUES (@customer\_num, @ordersQty, @maxDate, @productsQty);

ELSE

UPDATE CustomerStatistics

SET ordersQty =ordersQty+@ordersQty,

maxDate=@maxDate,

productsQty= @productsQty

WHERE customer\_num = @customer\_num;

END

**b. (hasta donde llegué)**

CREATE PROCEDURE generarInformeGerencial @fechaInforme date

AS

BEGIN

DECLARE CursorProductos CURSOR FOR SELECT stock\_num, manu\_code FROM products

DECLARE @stock\_num int, @manu\_code char(3), @cantOrdenes int, @UltCompra date, @cantClientes int,

@totalVentas float

OPEN CursorProductos;

FETCH FROM CursorProductos INTO @stock\_num, @manu\_code

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT @cantOrdenes=Count(\*) FROM items i JOIN orders o ON i.order\_num = o.order\_num WHERE i.stock\_num = @stock\_num AND i.manu\_code = @manu\_code

IF NOT EXISTS(SELECT 1 FROM informeStock WHERE fechaInforme = @fechaInforme)

INSERT INTO informeStock (fechaInforme, stock\_num, manu\_code, cantOrdenes, UltCompra, cantClientes, totalVentas)

VALUES (@fechaInforme, @stock\_num, @manu\_code, )

CLOSE CursorProductos;  
DEALLOCATE CursorProductos;

END

END

Select \* from informeStock

**c.**

create table informeVentas

(fechaInforme datetime,

cod\_Estado char(2),

customer\_num smallint,

cantOrdenes integer,

UltCompra datetime,

cantProductos integer,

totalVentas integer,

PRIMARY KEY (fechaInforme, cod\_Estado, customer\_num));

CREATE PROCEDURE generarInformeVentas (@fechaInforme date, @codEstado char(2))

AS

BEGIN

IF EXISTS (SELECT 1 FROM informeVentas WHERE cod\_Estado = @codEstado AND fechaInforme = @fechaInforme )

THROW 50000, 'Ya existe una venta para la fecha y estado indicados', 1;

INSERT INTO informeVentas (fechainforme, cod\_Estado, customer\_num, cantOrdenes, UltCompra, cantProductos, totalVentas)

SELECT @fechaInforme, @codEstado, c.customer\_num, Count(DISTINCT o.order\_num), MAX(order\_date), Count(DISTINCT i.stock\_num), SUM(quantity\*unit\_price)

FROM customer c LEFT JOIN orders o ON c.customer\_num = o.customer\_num

LEFT JOIN items i ON i.order\_num = o.order\_num

WHERE c.state = @codEstado

GROUP BY c.customer\_num

END

**Clase 10  
  
1.**

**a.**

CREATE VIEW fabricantes\_de\_mas\_2\_prod AS

SELECT m.manu\_code, m.manu\_name, Count(DISTINCT p.stock\_num) as cant\_productos, Max(o.order\_date) as ult\_fecha\_orden

FROM manufact m LEFT JOIN products p on m.manu\_code = p.manu\_code

LEFT JOIN items i on m.manu\_code = i.manu\_code AND p.stock\_num = i.stock\_num

LEFT JOIN orders o on i.order\_num = o.order\_num

GROUP BY m.manu\_code, m.manu\_name

HAVING Count(p.stock\_num) > 2 OR Count(p.stock\_num) = 0

**b.**

SELECT manu\_code, manu\_name, cant\_productos, COALESCE (CONVERT(varchar, ult\_fecha\_orden), 'No Posee

Órdenes')

FROM fabricantes\_de\_mas\_2\_prod

**2.**

SELECT m.manu\_code, manu\_name, Count(DISTINCT i.order\_num) cant\_ord, Sum(quantity\*i.unit\_price) AS monto\_total

FROM manufact m JOIN items i ON m.manu\_code = i.manu\_code

JOIN product\_types pt ON i.stock\_num = pt.stock\_num

WHERE m.manu\_code LIKE '[AN]\_\_' AND (pt.description LIKE '%tennis%' OR pt.description LIKE '%ball%')

GROUP BY m.manu\_code, manu\_name

HAVING Sum(quantity\*unit\_price) > (SELECT Sum(quantity\*unit\_price)/Count(DISTINCT i.manu\_code) FROM items i)

**3. (Solución de Hernán en clase)**

select 2, c.customer\_num, c.lname, c.company, 0 cantidad\_ordenes, null ultima\_compra, 0 montoTotal, (select sum(unit\_price\*quantity) FROM items) total\_general from customer c where customer\_num not in (select customer\_num from orders) UNION

select 1, c.customer\_num, c.lname, c.company, count(distinct o.order\_num), MAX(order\_date), sum(i.unit\_price\*quantity), (select sum(unit\_price\*quantity) FROM items ) total\_general from customer c join orders o on c.customer\_num = o.customer\_num join items i on o.order\_num = i.order\_num where c.customer\_num in (select DISTINCT o2.customer\_num from orders o2 JOIN items i2 ON o2.order\_num=i2.order\_num

WHERE i2.stock\_num IN (SELECT stock\_num FROM products GROUP BY stock\_num HAVING count(\*) >2)) group by c.customer\_num,c.lname,c.company having count(distinct o.order\_num) >= 3

ORDER BY 1  
  
**4. (Solución de Hernán en Clase)**SELECT top 5 t.description, c.state, SUM(i.quantity) FROM items i JOIN product\_types t ON i.stock\_num=t.stock\_num JOIN orders o ON i.order\_num = o.order\_num JOIN customer c ON o.customer\_num = c.customer\_num where i.stock\_num = (SELECT TOP 1 i1.stock\_num FROM product\_types t1 JOIN items i1 ON i1.stock\_num = t1.stock\_num JOIN orders o1 ON i1.order\_num = o1.order\_num JOIN customer c1 ON o1.customer\_num = c1.custome

WHERE c.state = c1.state GROUP BY i1.stock\_num --, c1.state ORDER BY SUM(i1.quantity) DESC) GROUP BY t.description, c.state ORDER BY SUM(i.quantity) desc

**Clase 11  
  
1.**SELECT c.customer\_num, lname, fname, SUM(i.unit\_price\*i.quantity) TotalDelCliente, Count(DISTINCT o.order\_num)OC\_Del\_Cliente, (SELECT Count(\*) FROM orders) CantidadTotalOC

FROM customer c JOIN orders o ON c.customer\_num = o.customer\_num

JOIN items i ON i.order\_num = o.order\_num

WHERE c.zipcode LIKE '94%'

GROUP BY c.customer\_num, lname, fname

HAVING Count(DISTINCT o.order\_num) > 1 AND (SUM(i.unit\_price\*i.quantity))/Count(DISTINCT o.order\_num) > (SELECT SUM(i1.unit\_price\*i1.quantity)/Count(DISTINCT i1.order\_num) FROM items i1)

**2a.**SELECT pt.stock\_num, m.manu\_code, pt.description, manu\_name, Sum(i.unit\_price\*i.quantity)'u$ por Producto', SUM(i.quantity)'Unid. por Producto'

FROM manufact m JOIN product\_types pt ON pt.stock\_num = pt.stock\_num

JOIN items i ON i.stock\_num = pt.stock\_num AND m.manu\_code = i.manu\_code

WHERE m.manu\_code IN (SELECT manu\_code FROM products p1 GROUP BY p1.manu\_code HAVING Count(\*) >= 10)

GROUP BY pt.stock\_num, m.manu\_code, pt.description, manu\_name

ORDER BY 5

**Clase 12**Listar el numero, nombre, apellido, aestado, cant de ordenes, monto total comprado

por cliente durante el año 2015 que no sean del estado de florida

Mostrar solo aquellos clientes cuyo monto total comprado sea mayor

que el promedio del monto total comprado por cliente que no sean

del estado de florida. Ordenado por total comprado en forma descendente\*/

SELECT c.customer\_num, fname, lname, c.state, Count(DISTINCT o.order\_num) AS 'Cantidad de Ordenes', SUM(i.quantity\*unit\_price) AS 'Monto Total'

FROM customer c JOIN orders o ON c.customer\_num = o.customer\_num

JOIN items i ON i.order\_num = o.order\_num

WHERE state != 'FL' AND YEAR(o.order\_date) = 2015

GROUP BY c.customer\_num, c.fname, c.lname, c.state

HAVING SUM(i.quantity\*unit\_price) > (SELECT SUM(quantity\*unit\_price)/Count(DISTINCT c1.customer\_num)

FROM customer c1 JOIN orders o1 ON c1.customer\_num = o1.customer\_num

JOIN items i1 ON i1.order\_num = o1.order\_num WHERE state != 'FL')

ORDER BY 6 DESC

10. Seleccionar todos los clientes cuyo monto total comprado sea mayor al de su refererente durante el

año 2015. Mostrar número, nombre, apellido y los montos totales comprados de ambos durante ese

año. Tener en cuenta que un cliente puede no tener referente y que el referente pudo no haber

comprado nada durante el año 2015, mostrarlo igual.\*/

SELECT c1.customer\_num, c1.fname, c1.lname, SUM(i1.quantity\*i1.unit\_price) AS montoTotalCli, c3.customer\_num, c3.fname, c3.lname, c3.montoTotalRef

FROM customer c1 JOIN orders o1 ON c1.customer\_num = o1.customer\_num

JOIN items i1 ON i1.order\_num = o1.order\_num

LEFT JOIN

(SELECT c2.customer\_num, fname, lname, SUM(i2.quantity\*i2.unit\_price) AS montoTotalRef

FROM customer c2 LEFT JOIN orders o2 ON c2.customer\_num = o2.customer\_num

LEFT JOIN items i2 ON i2.order\_num = o2.order\_num

WHERE YEAR(o2.order\_date) = 2015

GROUP BY c2.customer\_num, fname, lname ) c3 ON c1.customer\_num\_referedBy = c3.customer\_num

WHERE YEAR(o1.order\_date) = 2015

GROUP BY c3.customer\_num, c3.fname, c3.lname, c1.customer\_num, c1.fname, c1.lname, c3.montoTotalRef

HAVING SUM(i1.quantity\*i1.unit\_price) > Coalesce(c3.montoTotalRef,0)