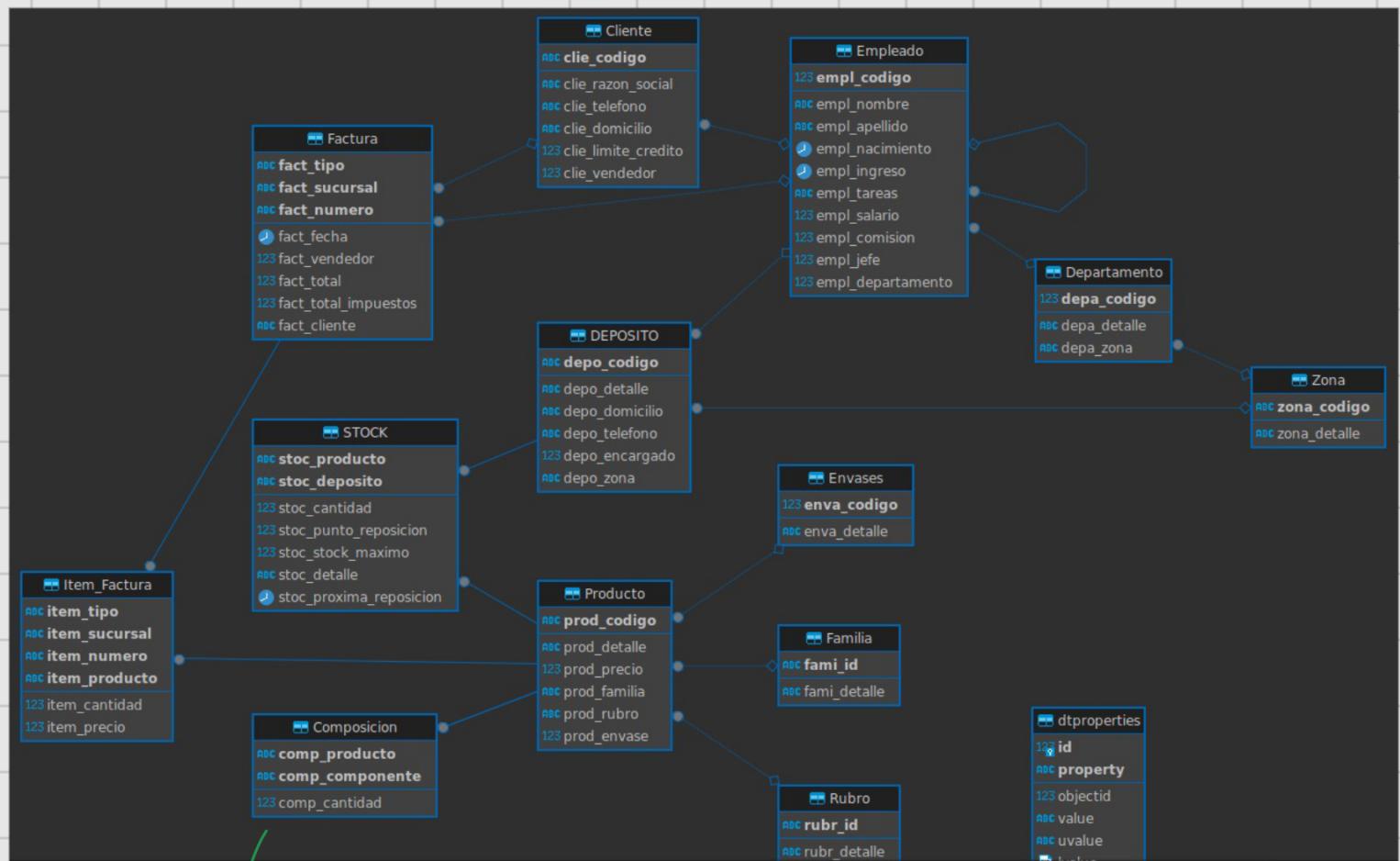


DER para ejercicios



Composición ←

- producto: código del producto que compone (padre)
- componente: código del producto que es componente (hijo)
- cantidad: cantidad de ese hijo que forma el padre

SUBSELECTS

①
tradicionales

WHERE
HAVING }
FROM

COMO condiciones

↓↓↓ con performance ?

② CORRELACIONADOS

1. WHERE }
2. HAVING } → COMO condiciones
3. SELECT }
4. ORDER BY } → Para traer datos

ORDEN DE Ejecución

Cómo se ve la consulta	Como se ejecuta	Por qué funciona de esta manera
SELECT	▶ FROM	▶ SQL comienza con la tabla de la que su consulta está tomando datos
FROM	▶ WHERE	▶ Así es como SQL filtra en filas
WHERE	▶ GROUP BY	▶ Aquí es donde su consulta SQL verifica si tiene una agregación
GROUP BY	▶ HAVING	▶ HAVING requiere una declaración GROUP BY Solo después de que se hayan realizado todos estos cálculos, SQL "SELECCIONARÁ" qué columnas desea que se devuelvan.
HAVING	▶ SELECT	
ORDER BY	▶ ORDER BY	▶ Esto ordena los datos devueltos
LIMIT	▶ LIMIT	▶ Por último, puede limitar el número de filas devueltas.

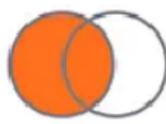
Ejemplo group by

title	genre	qty	genre	total
book 1	adventure	4		
book 2	fantasy	5		
book 3	romance	2		
book 4	adventure	3		
book 5	fantasy	3		
book 6	romance	1		
			adventure	7
			fantasy	8
			romance	3

SELECT genre, SUM(qty) AS total
FROM groups
GROUP BY genre;

JOINS

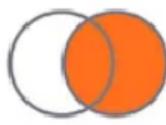
LEFT JOIN



Everything on the left
+
anything on the right that matches

```
SELECT *  
FROM TABLE_1  
LEFT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

RIGHT JOIN



Everything on the right
+
anything on the left that matches

```
SELECT *  
FROM TABLE_1  
RIGHT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

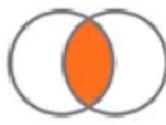
OUTER JOIN



Everything on the right
+
Everything on the left

```
SELECT *  
FROM TABLE_1  
OUTER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

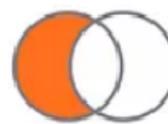
INNER JOIN



Only the things that match on the
left AND the right

```
SELECT *  
FROM TABLE_1  
INNER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

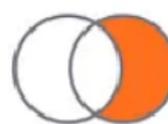
ANTI LEFT JOIN



Everything on the left
that is NOT on the right

```
SELECT *  
FROM TABLE_1  
LEFT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY  
WHERE TABLE_2.KEY IS NULL
```

ANTI RIGHT JOIN



Everything on the right
that is NOT on the left

```
SELECT *  
FROM TABLE_1  
RIGHT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY  
WHERE TABLE_1.KEY IS NULL
```

ANTI OUTER JOIN



Everything on the left and right
that is unique to each side

```
SELECT *  
FROM TABLE_1  
OUTER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY  
WHERE TABLE_1.KEY IS NULL  
OR TABLE_2.KEY IS NULL
```

CROSS JOIN

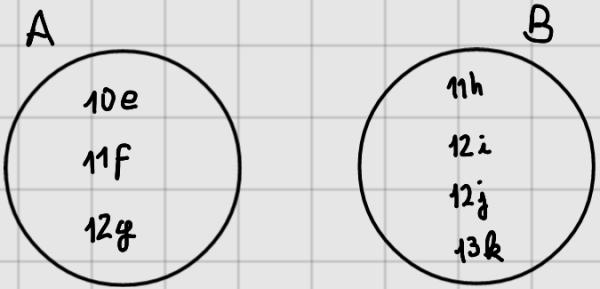


All combination of rows from the
right and the left (cartesian
product)

```
SELECT *  
FROM TABLE_1  
CROSS JOIN TABLE_2
```

EJEMPLOS JOIN

Tabla A		Tabla B	
Campo 1	Campo 2	Campo 3	Campo 4
10	e	11	h
11	f	12	i
12	g	12	j
		13	k



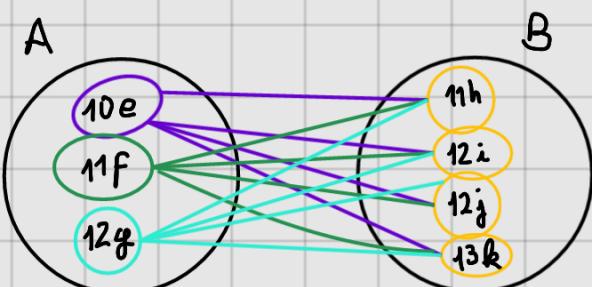
PRODUCTO CARTESIANO (cross join)

Se combinan todas las filas de A con todas las de B

Select *
From A, B

```
SELECT *
FROM TABLE_1
CROSS JOIN TABLE_2
```

en el ejemplo:



Campo 1	Campo 2	Campo 3	Campo 4
10	e	11	h
10	e	12	i
10	e	12	j
10	e	13	k
11	f	11	h
		

$$\text{cantidad de filas (cross join)} = \text{cant. filas A} \times \text{cant. filas B}$$

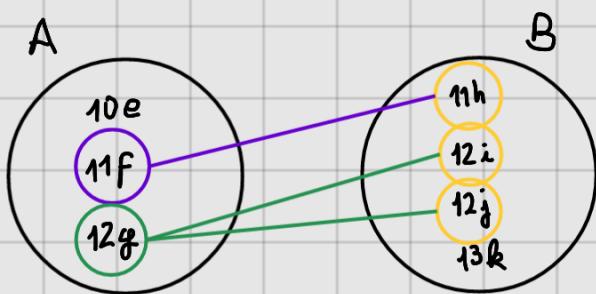
INNER JOIN (o simplemente Join)

Solo unirá aquellos con la misma clave de la condición (solo lo que tiene "match" en la otra tabla)

Select *
From A, B
Where A.id = B.id

```
SELECT *
FROM TABLE_1
INNER JOIN TABLE_2
ON TABLE_1.KEY = TABLE_2.KEY
```

En el ejemplo Si Joinearíamos : A.CAMPO1 = B.CAMPO3



el 10 y el 13 al no tener coincidencia en la otra tabla quedan fuera

Campo 1	Campo 2	Campo 3	Campo 4
11	f	11	h
12	g	12	i
12	g	12	j

• Si Joineo una FK con una PK

$$\underbrace{c.comp_prod}_{FK} = \underbrace{p.prod_cod}_{PK}$$

Voy a tener la misma CANT de filas,
porque cada FK deberá tener una única
equivalencia a la PK

SE GARANTIZA unicidad

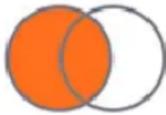
• Si Joineo una FK con otra FK

$$\underbrace{c.comp_prod}_{FK} = \underbrace{s.stock_prod}_{FK}$$

ya no garantizo unicidad porque por cada
comp._prod puedo tener muchos stock._prod
∴ se multiplican las filas

OUTER JOIN

LEFT JOIN (LEFT OUTER JOIN)



Everything on the left
+
anything on the right that matches

```
SELECT *  
FROM TABLE_1  
LEFT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

⇒ Si la Izquierda es la "dominante"

RIGHT JOIN (RIGHT OUTER JOIN)



Everything on the right
+
anything on the left that matches

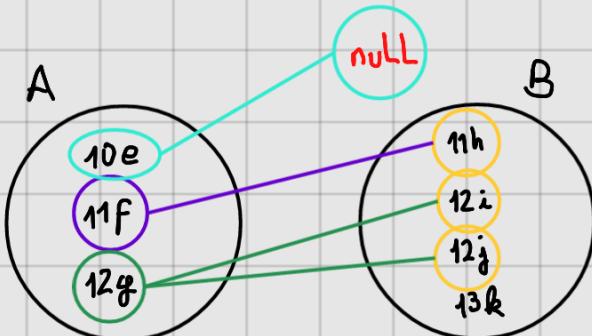
```
SELECT *  
FROM TABLE_1  
RIGHT JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

⇒ Si la Derecha es la "dominante"

ejemplo Left Join

en el ejemplo, si hiciésemos un

```
Select *  
From A  
left Join B  
On A.campo1 = B.campo3
```



Campo 1	Campo 2	Campo 3	Campo 4
11	f	11	h
12	g	12	i
12	g	12	j
10	e	null	null

los registros de la "dominante" que no tengan relación con ninguno de la otra van a null

OUTER JOIN (Full outer Join)



Everything on the right
+
Everything on the left

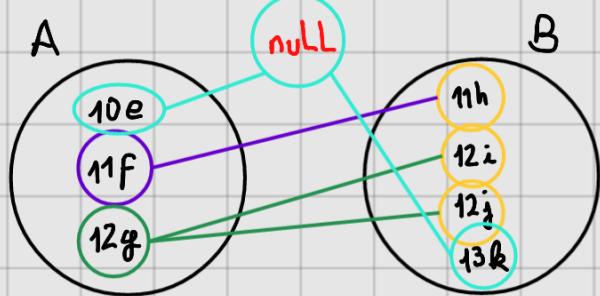
```
SELECT *  
FROM TABLE_1  
OUTER JOIN TABLE_2  
ON TABLE_1.KEY = TABLE_2.KEY
```

⇒ Si no hay dominante

ejemplo OuterJoin

en el ejemplo, si hiciéramos un

```
Select *  
From A  
Outer Join B  
On A.campo1 = B.campo3
```



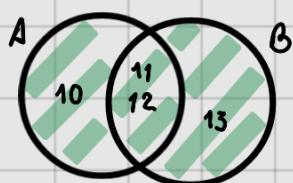
Campo 1	Campo 2	Campo 3	Campo 4
11	f	11	h
12	g	12	i
12	g	12	j
10	e	null	null
null	null	13	k

OPERADORES DE conjuntos

La cantidad de columnas no varían

Tabla A		Tabla B	
Campo 1	Campo2	Campo 3	Campo4
10	e	11	h
11	f	12	i
12	g	12	j
		13	k

UNIÓN



```
SELECT campo1
FROM TABLA A
UNION
SELECT campo3
FROM TABLA B
```

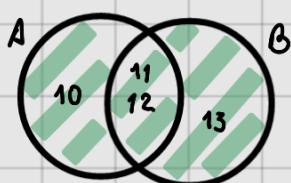
resultado :

res
10
11
12
12
13

→ Elimina duplicados

Mantiene los que aparecen más de una vez en su mínima expresión (no se puede "recorrer datos")

UNIÓN ALL



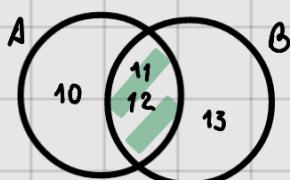
igual que el Union pero se pone todas las repeticiones

```
SELECT campo1
FROM TABLA A
UNION ALL
SELECT campo3
FROM TABLA B
```

resultado :

res
10
11
11
12
12
12
13

INTERSECT (intersección)

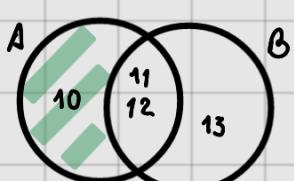


```
SELECT campo1
FROM TABLA A
INTERSECT
SELECT campo3
FROM TABLA B
```

resultado :

res
11
12

EXCEPT (resta)



```
SELECT campo1
FROM TABLA A
EXCEPT
SELECT campo3
FROM TABLA B
```

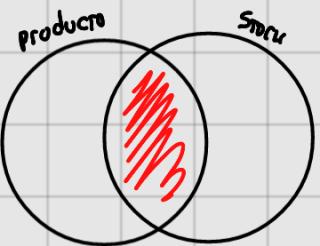
resultado :

res
10

3. Realizar una consulta que muestre código de producto, nombre de producto y el stock total, sin importar en que deposito se encuentre, los datos deben ser ordenados por nombre del artículo de menor a mayor.

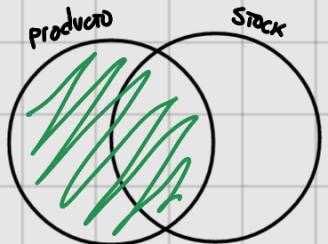
```
--mal: porque los productos sin stock no se muestran
SELECT P.prod_codigo,p.prod_detalle,s.stoc_cantidad
FROM Producto P INNER JOIN Stock S ON S.stoc_producto=P.prod_codigo
GROUP BY P.prod_codigo,p.prod_detalle,s.stoc_cantidad
ORDER BY p.prod_detalle ASC
```

X



```
--OK: se muestran los productos aunque no tengan stock
SELECT P.prod_codigo,p.prod_detalle,ISNULL(s.stoc_cantidad,0)
FROM Producto P
LEFT OUTER JOIN Stock S ON P.prod_codigo = S.stoc_producto
GROUP BY P.prod_codigo,p.prod_detalle,s.stoc_cantidad
ORDER BY p.prod_detalle ASC
```

✓



4. Realizar una consulta que muestre para todos los artículos código, detalle y cantidad de artículos que lo componen. Mostrar solo aquellos artículos para los cuales el stock promedio por depósito sea mayor a 100.

Mal Ojo aca, con el inner join a Stock lo que estoy haciendo es, los registros que ya tenia (que era la cantidad de productos) se multiplica por la cantidad de veces que lo tengo en stock. Es decir que la sumatoria no va a decir la cantidad de artículos que lo componen, sino la cantidad de artículos que componen ~~tocando~~ el stock que tenga (se multiplica la composición de un producto por el stock)

```
/*
SELECT P.prod_codigo, p.prod_detalle, SUM(ISNULL(C.comp_cantidad, 0))
FROM Producto P
LEFT OUTER JOIN Composicion C ON P.prod_codigo = C.comp_producto
INNER JOIN Stock S ON P.prod_codigo = S.stoc_producto --> maaaaaaal X
GROUP BY P.prod_codigo, p.prod_detalle
```

• Si Joineo una FK con una PK

C.comp_prod = P.prod_cod
 FK PK

Voy a tener la misma CANT de filas,
 porque cada FK deberá tener una única
 equivalencia a la PK
 SE GARANTIZA unicidad

• Si Joineo una FK con otra FK

C.comp_prod = S.stoc_prod
 FK PK

ya no garantizo unicidad porque por cada
 comp_prod puedo tener muchos stoc_prod
 .. se multiplican las filas

Lo ponemos en el having (desp.agrupar) y no en el where
 porque le tengo que pasar el código de producto.
 Si lo pongo en el where (antes de agrupar) lo voy a estar pasando
 muchas veces lo mismo

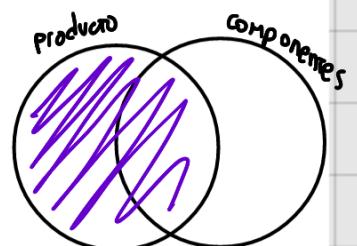
--Sigo pensando para ver el promedio de stock por depósito, llego a un subselect:

```
SELECT avg(s.stoc_cantidad) FROM Stock S WHERE S.stoc_producto = '00006404'
```



La inserto en la query correcta

```
SELECT P.prod_codigo, p.prod_detalle, SUM(ISNULL(C.comp_cantidad, 0)) AS componentes,
  (SELECT avg(s.stoc_cantidad) FROM Stock S WHERE S.stoc_producto = P.prod_codigo) AS stockProm
  FROM Producto P
  LEFT OUTER JOIN Composicion C ON P.prod_codigo = C.comp_producto
  GROUP BY P.prod_codigo, p.prod_detalle
  HAVING (SELECT avg(s.stoc_cantidad) FROM Stock S WHERE S.stoc_producto = P.prod_codigo) > 100
```



prod_codigo	prod_detalle	componentes	stockProm
1 00000000	Bolsas para despachar	0.00	125.000000
2 00000302	TARJETAS C.T.I. DE \$ 20	0.00	364.818181
3 00000303	TARJETAS C.T.I. DE \$ 15	0.00	336.416666
4 00000806	BAZOOKA X 120u. MENTA	0.00	114.000000
5 00000809	PALITOS SELVA FRU/VAIN X 220u.	0.00	140.555555
6 00000817	BELDENT X 20u. MENTA	0.00	144.000000

5. Realizar una consulta que muestre código de artículo, detalle y cantidad de egresos de stock que se realizaron para ese artículo en el año 2012 (egresan los productos que fueron vendidos). Mostrar solo aquellos que hayan tenido más egresos que en el 2011.

```
-- ventas 2012 para los productos mostrando lo que me pide
SELECT p.prod_codigo, p.prod_detalle, SUM(i.item_cantidad)
FROM Producto P
INNER JOIN Item_Factura I ON P.prod_codigo=i.item_producto -- PK=FK
INNER JOIN Factura F ON F.fact_tipo = i.item_tipo AND f.fact_sucursal = i.item_sucursal AND f.fact_numero=i.item_numero
WHERE YEAR(f.fact_fecha) = 2012
GROUP BY p.prod_codigo, p.prod_detalle
Todas PK=FK/
```



Me quedo la misma cant. de filas que Item_Factura por relacionar FK con PK

```
-- armo la query de ventas de 2011 para un el producto '00010220'
SELECT SUM(i.item_cantidad) as VENTAS_2011
FROM Item_Factura I
INNER JOIN Factura F ON F.fact_tipo = i.item_tipo AND f.fact_sucursal = i.item_sucursal AND f.fact_numero=i.item_numero
WHERE YEAR(f.fact_fecha) = 2011 AND I.item_producto = '00010220'
```

```
-- combino ambas querys
SELECT p.prod_codigo, p.prod_detalle, SUM(i.item_cantidad)
FROM Producto P
INNER JOIN Item_Factura I ON P.prod_codigo=i.item_producto -- PK=FK
INNER JOIN Factura F ON F.fact_tipo = i.item_tipo AND f.fact_sucursal = i.item_sucursal AND f.fact_numero=i.item_numero -- todas
WHERE YEAR(f.fact_fecha) = 2012
GROUP BY p.prod_codigo, p.prod_detalle
HAVING sum(i.item_cantidad) > ISNULL((), 0)

SELECT SUM(i2.item_cantidad) as VENTAS_2011
FROM Item_Factura I2
INNER JOIN Factura F2 ON F2.fact_tipo = i2.item_tipo AND f2.fact_sucursal = i2.item_sucursal AND f2.fact_numero=i2.item_numero
WHERE YEAR(f2.fact_fecha) = 2011 AND I2.item_producto = P.prod_codigo -- EL PRODUCTO DE LA QUERY PADRE
),0) -- Le ponemos el isnull porque no puede comparar > a nulo
```

8. Mostrar para el o los artículos que tengan stock en todos los depósitos, nombre del artículo, stock del depósito que más stock tiene.

```

SELECT P.prod_detalle,
MAX(S.stoc_cantidad) as [Stock maximo] ,
count(*) as [Cantidad de depositos en los que esta]
FROM Producto P
INNER JOIN Stock S ON P.prod_codigo = S.stoc_producto
WHERE S.stoc_cantidad > 0
GROUP BY P.prod_codigo, p.prod_detalle
HAVING COUNT(*) = (SELECT COUNT(*) FROM DEPOSITO)

```

1) FROM

PRODUCTO	STOCK
01 prod1	dep01 80 u.
01 prod1	dep02 40 u.
02 prod2	dep01 15 u.

me quedan tantas filas como tabla STOCK

2) WHERE

filtro aquellos productos con stock cero

3) GROUP By

Agrupo por código y detalle

clave del grupo	
01 prod1	dep01, dep02 80,40
02 prod2	dep01 15

Count(*) sin group by: cuenta cantidad de filas de un resultado de una query
 Count(*) con group by: cuenta el número de filas de cada grupo

4) HAVING

Si hago el count de esta tabla, me dará la cantidad de veces que se repite cada producto, y como se repite una vez por cada producto en stock en un almacén, el count(*) será la cantidad de depósitos en los que está cada producto

clave del grupo	cantidad de filas donde aparece count(*)
01 prod1	2
02 prod2	1

entonces puedo filtrar diciendo que la cantidad de depósitos debe ser igual a la total

5) SELECT

Muestro lo que me pide la consigna

a) detalle

b) MAX STOCK (para esto busco la fila del grupo con más cantidad de stock)

9. Mostrar el código del jefe, código del empleado que lo tiene como jefe, nombre del mismo y la cantidad de depósitos que ambos tienen asignados.

Antes que nada vemos LA TABLA empleado, Tiene un campo Jefe que se relaciona con otro registro de la misma tabla

	Results	Messages								
	empl_codigo	empl_nombre	empl_apellido	empl_nacimiento	empl_ingreso	empl_areas	empl_salario	empl_comision	empl_jefe	empl_departamento
1	1	Juan	Perez	1978-01-01 00:00:00	2000-01-01 00:00:00	Gerente	25000.00	0.00	NULL	1
2	2	Pedro	Gonzalez	1979-01-05 00:00:00	2005-01-05 00:00:00	Jefe Compras	15000.00	0.00	1	3
3	3	Horacio	Quiroga	1975-08-05 00:00:00	2003-12-12 00:00:00	Jefe Ventas	10000.00	0.20	1	2
4	4	Gabriel	Trap	1965-05-05 00:00:00	2002-01-05 00:00:00	Vendedor	8000.00	0.13	3	2
5	5	Patricia	Borr	1946-08-03 00:00:00	2001-05-05 00:00:00	Vendedor	8000.00	0.15	3	2
6	6	Belen	Trap	1980-01-14 00:00:00	2010-05-05 00:00:00	Administrativa	3500.00	NULL	2	1
7	7	Fernando	Trisolí	1978-05-22 00:00:00	2010-03-02 00:00:00	Vendedor	9000.00	0.12	3	2
8	8	Santiago	Jand	1962-08-01 00:00:00	2009-02-05 00:00:00	Vendedor	9500.00	0.18	3	2
9	9	Matias	Doind	1974-11-12 00:00:00	2009-11-05 00:00:00	Vendedor	9200.00	0.17	3	2

--depositos que el jefe 1 y el empleado 2 tienen asignados:

```
SELECT * FROM deposito d WHERE d.depo_encargado in(1,2)
```

	Results	Messages				
	depo_codigo	depo_detalle	depo_domicilio	depo_telefono	depo_encargado	depo_zona
1	00	DEPOSITO CURUZU CUATIA	PEDRO NOLASCO SILVA S/N	NULL	1	003
2	04	VARGAS TINO	AVDA SAN MARTIN	NULL	2	010
3	06	DEBONA LEANDRO	ECUADOR	NULL	1	005
4	07	JONES CARLOS	SAN MARTIN	NULL	1	011
5	11	DEPOSITO LIBRES	SARMIENTO	NULL	2	003
6	13	MAQUIAVELO CRISTIAN	TRISTAN FERNANDEZ S/N	NULL	2	005
7	14	ORTIZ JULIO	AVDA ALEM	NULL	2	007
8	16	BERTINI CARLOS	SARMIENTO & LARREA	NULL	1	008
9	17	ROMANI PABLO	FALUCHO & P. FERRE	NULL	1	003
10	19	UDIATI UECTOP	CUNICULICO AL DD. DIAZ	NULL	2	010

Si quiero contar la cantidad de depósitos que tienen como encargado Al Jefe 1 o Empleado 2 (depósitos para cada combinación)

--cantidad de depósitos que el jefe 1 y el empleado 2 tienen asignados:

```
SELECT COUNT(*) FROM deposito d WHERE d.depo_encargado in(1,2) --que el empleado sea el 1 o el 2
```

y Así queda la query final :

```
SELECT s.empl_codigo AS [codigo jefe], e.empl_codigo AS [codigo empleado],
concat(rtrim(e.empl_apellido),',',rtrim(e.empl_nombre)) AS nombre,
(
SELECT COUNT(*) FROM deposito d WHERE d.depo_encargado = e.empl_codigo OR d.depo_encargado = s.empl_codigo
) AS [depositos asignados]
FROM empleado e INNER JOIN empleado s ON e.empl_jefe = s.empl_codigo
```

NOTAR que se les pone
ALIAS

Una es el "padre" y la otra el "hijo"

	Results	Messages		
	codigo jefe	codigo empleado	nombre	depositos asignados
1	1	2	Gonzalez,Pedro	16
2	1	3	Quiroga,Horacio	13
3	3	4	Trap,Gabriel	11
4	3	5	Borr,Patricia	13
5	2	6	Trap,Belen	10
6	3	7	Trisolí,Fernando	7
7	3	8	Jand,Santiago	7
8	3	9	Doind,Matias	7

15. Escriba una consulta que retorne los pares de productos que hayan sido vendidos juntos (en la misma factura) más de 500 veces. El resultado debe mostrar el código y descripción de cada uno de los productos y la cantidad de veces que fueron vendidos juntos. El resultado debe estar ordenado por la cantidad de veces que se vendieron juntos dichos productos. Los distintos pares no deben retornarse más de una vez.

Ejemplo de lo que retornaría la consulta:

PROD1	DETALLE1	PROD2	DETALLE2	VECES
1731	MARLBORO KS	1718	PHILIPS MORRIS KS	507
1718	PHILIPS MORRIS KS	1705	PHILIPS MORRIS BOX	10562

Empiezo Joinando la tabla de items con si misma, solo Joino por las claves de "Factura" porque quiero que cada par de productos Esté en la misma factura, pero no necesariamente sea el mismo

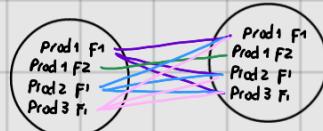
```
SELECT *
FROM Item_Factura I1
-- joino productos de la misma factura, pero pueden ser diferentes
INNER JOIN Item_Factura I2 ON
I1.item_sucursal=I2.item_sucursal AND I2.item_numero=I1.item_numero AND I1.item_tipo=I2.item_tipo
```

Le AGREGO los campos pedidos de producto a %

```
SELECT *
FROM Item_Factura I1
-- joino productos de la misma factura, pero pueden ser diferentes
INNER JOIN Item_Factura I2 ON
I1.item_sucursal=I2.item_sucursal AND I2.item_numero=I1.item_numero AND I1.item_tipo=I2.item_tipo
-- traigo ademas los datos de codigo y detalle de cada producto
INNER JOIN Producto P1 ON p1.prod_codigo=I1.item_producto
INNER JOIN Producto P2 ON p2.prod_codigo=I2.item_producto
```

Como debes retornar cada par solo una vez, y los pares deben ser de dif. productos:

```
SELECT *
FROM Item_Factura I1
-- joino productos de la misma factura, pero pueden ser diferentes
INNER JOIN Item_Factura I2 ON
I1.item_sucursal=I2.item_sucursal AND I2.item_numero=I1.item_numero AND I1.item_tipo=I2.item_tipo
-- traigo ademas los datos de codigo y detalle de cada producto
INNER JOIN Producto P1 ON p1.prod_codigo=I1.item_producto
INNER JOIN Producto P2 ON p2.prod_codigo=I2.item_producto
-- con esto saco la relacion de cada producto con si mismo y todos los inversos (puede ser < o >)
where i1.item_producto<i2.item_producto
```



I1	I2
P1F1	P4F1
P1F1	P2F1
P1F1	P3F1
P1F2	P1F2
P2F1	P4F1
P2F1	P2F1
P2F1	P3F1
P3F1	P4F1
P3F1	P2F1
P3F1	P3F1

↓
Se deben sacar
Lo hago con
 $> \theta <$

I1	I2
P1F1	P2F1
P1F1	P3F1
P2F1	P3F1

RESOLUCIÓN FINAL

```
--pares de productos que se vendieron juntos mas de 500 veces
SELECT i1.item_producto,p1.prod_detalle, i2.item_producto,p2.prod_detalle,COUNT(*) AS VECES
FROM Item_Factura I1
-- joino productos de la misma factura, pero pueden ser diferentes
INNER JOIN Item_Factura I2 ON
I1.item_sucursal=I2.item_sucursal AND I2.item_numero=I1.item_numero AND I1.item_tipo=I2.item_tipo
-- traigo ademas los datos de codigo y detalle de cada producto
INNER JOIN Producto P1 ON p1.prod_codigo=I1.item_producto
INNER JOIN Producto P2 ON p2.prod_codigo=I2.item_producto
-- con esto saco la relacion de cada producto con si mismo y todos los inversos (puede ser < o >)
where i1.item_producto<i2.item_producto
group by i1.item_producto,p1.prod_detalle, i2.item_producto,p2.prod_detalle
--mas de 500 veces
having COUNT(*) > 500
order by COUNT(*)
```

Ahora solo Agrupo por par de productos, Muestro los datos pedidos en el select y ..

Count(*) sera' La cant. de veces que ese par Aparecio en una factura Juntos.

Aplico el having > 500 y lo ordeno

el i1.item_producto<i2.item_producto se podria poner en el having y tendría el mismo resultado, pero no seria performante porque se tomo el trabajo de agrupar y despues filtrar, cuando se podria haber ahorrado agrupaciones si se lo ponia en el where

10. Mostrar los 10 productos más vendidos en la historia y también los 10 productos menos vendidos en la historia. Además mostrar de esos productos, quien fue el cliente que mayor compra realizó.

Primero arranqué planeando la query para los mas vendidos, pero falta unir con los 10 menos vendidos

```
--LOS 10 MAS VENDIDOS (pero falta unir con 10 menos vendidos)
SELECT TOP 10 p.prod_codigo, p.prod_detalle, sum(i.item_cantidad) AS cantidad_vendida,
    -- cliente que mas unidades compro del producto
    SELECT top 1 C2.clie_razon_social
    FROM Item_Factura I2
    INNER JOIN Factura F2 ON f2.fact_tipo=i2.item_tipo AND f2.fact_sucursal=I2.item_sucursal AND f2.fact_numero=i2.item_numero
    INNER JOIN Cliente C2 ON F2.fact_cliente=C2.clie_codigo
    WHERE I2.item_producto = p.prod_codigo
    GROUP BY C2.clie_codigo,C2.clie_razon_social
    ORDER BY sum(i2.item_cantidad) DESC
) AS cliente_que_mas_compro
FROM Producto P
INNER JOIN Item_Factura I ON i.item_producto=p.prod_codigo
GROUP BY p.prod_codigo, p.prod_detalle
ORDER BY sum(i.item_cantidad) DESC
```

Solución uniendo Los 10 mas vendidos y los 10 menos vendidos

```
-- SOLUCION CON LOS 10 MAS VENDIDOS Y LOS 10 MENOS VENDIDOS
SELECT p.prod_codigo, p.prod_detalle, (
    -- cliente que mas unidades compro del producto
    SELECT top 1 F2.fact_cliente
    FROM Item_Factura I2
    INNER JOIN Factura F2 ON f2.fact_tipo=i2.item_tipo AND f2.fact_sucursal=I2.item_sucursal AND f2.fact_numero=i2.item_numero
    WHERE I2.item_producto = p.prod_codigo
    GROUP BY F2.fact_cliente
    ORDER BY sum(i2.item_cantidad) DESC
) AS cliente_que_mas_compro
FROM Producto P
WHERE P.prod_codigo IN (
    -- Código de 10 productos mas vendidos
    SELECT TOP 10 p.prod_codigo
    FROM Producto P
    INNER JOIN Item_Factura I ON i.item_producto=p.prod_codigo
    GROUP BY p.prod_codigo, p.prod_detalle
    ORDER BY sum(i.item_cantidad) DESC
) OR P.prod_codigo IN (
    -- Código de 10 productos menos vendidos
    SELECT TOP 10 p.prod_codigo
    FROM Producto P
    INNER JOIN Item_Factura I ON i.item_producto=p.prod_codigo
    GROUP BY p.prod_codigo, p.prod_detalle
    ORDER BY sum(i.item_cantidad) ASC
)
```

Solución con UNION

```
-- Solución del profe con UNION
SELECT p.prod_codigo, p.prod_detalle
,(SELECT TOP 1 f.fact_cliente FROM Factura f JOIN item_factura it ON it.item_numero = f.fact_numero
AND it.item_sucursal = f.fact_sucursal AND it.item_tipo = f.fact_tipo WHERE it.item_producto = p.prod_codigo
GROUP BY f.fact_cliente ORDER BY SUM(item_cantidad) DESC) AS cliente,
'Menos Vendidos' AS Rango
FROM Producto P
WHERE p.prod_codigo IN (SELECT TOP 10 item_producto FROM item_factura GROUP BY item_producto ORDER BY SUM(item_cantidad))
UNION ALL
SELECT p.prod_codigo, p.prod_detalle
,(SELECT TOP 1 f.fact_cliente FROM Factura f JOIN item_factura it ON it.item_numero = f.fact_numero
AND it.item_sucursal = f.fact_sucursal AND it.item_tipo = f.fact_tipo WHERE it.item_producto = p.prod_codigo
GROUP BY f.fact_cliente ORDER BY SUM(item_cantidad) DESC) AS cliente,
'Mas Vendidos' AS Rango
FROM Producto P
WHERE p.prod_codigo IN (SELECT TOP 10 item_producto FROM item_factura GROUP BY item_producto ORDER BY SUM(item_cantidad) DESC)
```

11. Realizar una consulta que retorne el detalle de la familia, la cantidad diferentes de productos vendidos y el monto de dichas ventas sin impuestos. Los datos se deberán ordenar de mayor a menor, por la familia que más productos diferentes vendidos tenga, solo se deberán mostrar las familias que tengan una venta superior a 20000 pesos para el año 2012.

Mi Solución

```
SELECT tam.fami_detalle,
       COUNT(DISTINCT p.prod_codigo) AS Cantidad_dif_de_productos,
    -- Suponiendo que el monto sin impuestos es fact_total - fact_total_impuestos
    --SUM(fac.fact_total - fac.fact_total_impuestos) AS Monto_sin_impuestos
    --Suponiendo que el monto sin impuestos es cant*precio
    --SUM(i.item_cantidad * i.item_precio) AS Monto_sin_impuestos
  FROM TItem Factura_T
 INNER JOIN Producto P ON i.item_producto=p.prod_codigo
 INNER JOIN Familia Fam ON Fam.fami_id=p.prod_familia
 INNER JOIN Factura Fac ON fac.fact_lipo=i.item_lipo AND fac.fact_sucursal=i.item_sucursal AND fac.fact_numero=i.item_numero
 WHERE Fam.fami_id IN (
    --familias que tuvieron una venta superior a 20000 pesos para el año 2012
    SELECT tam2.fami_id
    FROM item_lactura i2
    INNER JOIN Producto P2 ON i2.item_producto=p2.prod_codigo
    INNER JOIN Familia Fam2 ON Fam2.fami_id=p2.prod_familia
    INNER JOIN Factura Fac2 ON fac2.fact_lipo=i2.item_lipo AND fac2.fact_sucursal=i2.item_sucursal AND fac2.fact_numero=i2.item_numero
    WHERE YEAR(fac2.fact_fecha) = '2012'
    GROUP BY tam2.fami_id, tam2.fami_detalle
    HAVING SUM(i2.item_cantidad * i2.item_precio) > '20000'
)
GROUP BY tam.fami_id, tam.fami_detalle
ORDER BY COUNT(DISTINCT p.prod_codigo) DESC
```

Solución del profe:

```
-- SOLUCION DEL PROFE
SELECT f.fami_detalle, COUNT(DISTINCT p.prod_codigo) AS [productos_vendidos], SUM(it.item_precio * it.item_cantidad) AS [venta]
FROM Familia F
INNER JOIN producto p ON p.prod_familia = f.fami_id
INNER JOIN item_factura it ON it.item_producto = p.prod_codigo
GROUP BY f.fami_id, f.fami_detalle
HAVING f.fami_id IN (
    SELECT p.prod_familia FROM producto p
    INNER JOIN item_factura it ON it.item_producto = p.prod_codigo
    INNER JOIN factura f ON it.item_numero = f.fact_numero AND it.item_lipo = f.fact_lipo
    AND it.item_sucursal = f.fact_sucursal
    WHERE YEAR(f.fact_fecha) = 2012
    GROUP BY p.prod_familia
    HAVING SUM(it.item_precio * it.item_cantidad) > 20000
)
ORDER BY 2 DESC
```

⚠ A tener en cuenta :

Monto de Ventas sin impuestos → $\sum(\text{item_cantidad} * \text{item_precio})$

Monto Vendido del producto

12. Mostrar nombre de producto, cantidad de clientes distintos que lo compraron importe promedio pagado por el producto, cantidad de depósitos en los cuales hay stock del producto y stock actual del producto en todos los depósitos. Se deberán mostrar aquellos productos que hayan tenido operaciones en el año 2012 y los datos deberán ordenarse de mayor a menor por monto vendido del producto.

Mi solución:

```
SELECT p.prod_detalle,
       COUNT(DISTINCT fac.fact_cliente) AS cant_compradores,
       --MI MANERA:
       AVG(i.item_precio) AS precio_prom,
       MANERA DEL PROFESOR:
       SUM(item_cantidad * item_precio) / SUM(item_cantidad) AS precio_prom,
       (
           -- cantidad de depósitos en los que hay stock de p.prod_codigo
           SELECT COUNT(DISTINCT d.depo_codigo)
           FROM Stock s
           INNER JOIN DEPOSITO D ON s.stoc_deposito = d.depo_codigo
           WHERE s.stoc_cantidad > 0 AND s.stoc_producto = p.prod_codigo
       ) AS cantidad_de_depositos_stock,
       (
           --stock actual del producto en todos los depósitos
           SELECT ISNULL(SUM(s2.stoc_cantidad), 0)
           FROM Stock s2
           INNER JOIN DEPOSITO D2 ON s2.stoc_deposito = d2.depo_codigo
           WHERE s2.stoc_cantidad > 0 AND s2.stoc_producto = p.prod_codigo
       ) AS stock_todos_depositos
   FROM Item_Factura I
   INNER JOIN Producto P ON i.item_producto = p.prod_codigo
   INNER JOIN Factura Fac ON fac.fact_tipo = i.item_tipo AND fac.fact_sucursal = i.item_sucursal AND fac.fact_numero = i.item_numero
   WHERE YEAR(fac.fact_fecha) = '2012'
   GROUP BY p.prod_codigo, p.prod_detalle
   ORDER BY SUM(i.item_cantidad * i.item_precio) DESC
   OJO, No confundir cantidad vendida (i.item_cantidad) con monto total (i.item_cantidad * i.item_precio)
```

Solución del Profesor:

```
SOLUCIÓN DEL PROFESOR
SELECT p.prod_detalle, SUM(item_cantidad * item_precio) / SUM(item_cantidad) AS [importe promedio],
       COUNT(*) AS [cantidad de clientes],
       (
           SELECT COUNT(DISTINCT s.stoc_deposito)
           FROM Stock s
           WHERE s.stoc_cantidad > 0 AND s.stoc_producto = p.prod_codigo
       ) AS [cantidad de depósitos],
       (
           SELECT ISNULL(SUM(s.stoc_cantidad), 0)
           FROM Stock s
           WHERE s.stoc_cantidad > 0 AND s.stoc_producto = p.prod_codigo
       ) AS [stock actual]
   FROM Producto p
   INNER JOIN Item_Factura i ON i.item_producto = p.prod_codigo
   INNER JOIN Factura f ON f.fact_tipo = i.item_tipo AND f.fact_sucursal = i.item_sucursal AND f.fact_numero = i.item_numero
   WHERE YEAR(f.fact_fecha) = 2012
   GROUP BY p.prod_codigo, p.prod_detalle
   ORDER BY SUM(i.item_precio * i.item_cantidad) DESC
```

13. Realizar una consulta que retorne para cada producto que posea composición nombre del producto, precio del producto, precio de la sumatoria de los precios por la cantidad de los productos que lo componen. Solo se deberán mostrar los productos que estén compuestos por más de 2 productos y deben ser ordenados de mayor a menor por cantidad de productos que lo componen.

Sumatoria de precios por : $\text{precioUnitComp1} * \text{cantComp1} + \text{precioUnitComp2} * \text{cantComp2}$
cant de productos que lo componen

```
--SELECT P.prod_detalle,
--P.prod_precio,
--SUM(P2.prod_precio * C.comp_cantidad) AS sumatoria_precios_cant
--FROM Composicion C
--INNER JOIN Producto P ON C.comp_producto = P.prod_codigo --el producto
--INNER JOIN Producto P2 ON C.comp_componente = P2.prod_codigo --el componente
--GROUP BY P.prod_codigo, P.prod_detalle, P.prod_precio
---- HAVING COUNT(*) > 2 -- MAAAAl porque los componentes pueden tener mas de una unidad
---- ORDER BY COUNT(*) DESC
---- un producto con dos componentes pero tienen cant 1u y 2u son 3 productos que lo componen
---- CONCLUSION: Es con el SUM porque dice productos, NO productos distintos.
--HAVING SUM(C.comp_cantidad) > 2 --OK
--ORDER BY SUM(C.comp_cantidad) DESC
```

14. Escriba una consulta que retorne una estadística de ventas por cliente. Los campos que debe retornar son:

Código del cliente

Cantidad de veces que compro en el último año

Promedio por compra en el último año

Cantidad de productos diferentes que compro en el último año

Monto de la mayor compra que realizo en el último año

Se deberán retornar todos los clientes ordenados por la cantidad de veces que compro en el último año.

No se deberán visualizar NULLs en ninguna columna

Mi SOLUCIÓN

```
SELECT c.clie_codigo,
-- Aca usamos el concat porque la clave es triple y el distinct solo me deja un campo
count(distinct concat(f.fact_tipo,f.fact_sucursal,f.fact_numero)) as Cant_facturas,
avg(f.fact_total) as Prom_compra,
count(distinct i.item_producto) as Productos_dif,
max(f.fact_total) as Mayor_compra
FROM Item_Factura I
INNER JOIN Factura F on f.fact_tipo = i.item_tipo AND f.fact_sucursal = i.item_sucursal AND f.fact_numero = i.item_numero
INNER JOIN Cliente C on f.fact_cliente=C.clie_codigo
WHERE YEAR(F.fact_fecha) = YEAR(GETDATE())
group by c.clie_codigo
order by count(distinct concat(f.fact_tipo,f.fact_sucursal,f.fact_numero)) desc
```

ACA USO CONCAT porque la clave de factura es triple y count(distinct...) solo permite un campo

Solución del profesor (teniendo en cuenta todos los clientes)

```
--SOLUCION DEL PROFESOR TENIENDO EN CUENTA TODOS LOS CLIENTES (Habra campos nulos)
SELECT c.clie_codigo,
count(distinct concat(f.fact_tipo,f.fact_sucursal,f.fact_numero)) as Cant_facturas,
avg( ISNULL(f.fact_total,0)/*para cuando es nulo*/ ) as Prom_compra,
count(distinct i.item_producto) as Productos_dif,
max( ISNULL(f.fact_total,0)/*para cuando es nulo*/ ) as Mayor_compra
FROM Cliente C
LEFT OUTER JOIN Factura F on f.fact_cliente=C.clie_codigo AND YEAR(F.fact_fecha) = YEAR(GETDATE())
LEFT OUTER JOIN Item_Factura I on f.fact_tipo = i.item_tipo AND f.fact_sucursal = i.item_sucursal AND f.fact_numero = i.item_numero
group by c.clie_codigo
order by count(distinct concat(f.fact_tipo,f.fact_sucursal,f.fact_numero)) desc
```

USA Left Join para tener en cuenta todos los clientes, incluso los que nunca compraron

↓ Pone el filtro en el Join porque ...

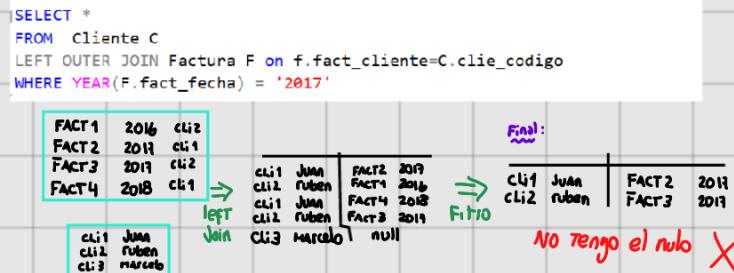
CASO①

```
-- CASO 1:
-- Primero filtra las facturas de 2017
-- Despues me hace el left join con la tabla dominante cliente

SELECT *
FROM Cliente C
LEFT OUTER JOIN Factura F on f.fact_cliente=C.clie_codigo AND YEAR(F.fact_fecha) = '2017'
```



```
-- CASO 2:
-- Primero hace el left join de cliente dominante con factura dominada
-- Despues filtra por año 2017
```



18. Escriba una consulta que retorne una estadística de ventas para todos los rubros.

La consulta debe retornar:

DETALLE_RUBRO: Detalle del rubro

VENTAS: Suma de las ventas en pesos de productos vendidos de dicho rubro

PROD1: Código del producto más vendido de dicho rubro

PROD2: Código del segundo producto más vendido de dicho rubro

CLIENTE: Código del cliente que compró más productos del rubro en los últimos 30 días

La consulta no puede mostrar NULL en ninguna de sus columnas y debe estar ordenada por cantidad de productos diferentes vendidos del rubro.

```
SELECT r.rubr_detalle,
       sum(i.item_cantidad*i.item_precio),
       (
           SELECT TOP 1 P2.prod_codigo
           FROM Item_Factura I2
           INNER JOIN Producto P2 ON P2.prod_codigo = I2.item_producto
           WHERE P2.prod_rubro = R.rubr_id
           GROUP BY P2.prod_codigo
           ORDER BY SUM(I2.item_cantidad*I2.item_precio) DESC
       )
   AS masVendido,
```

→ MAS VENDIDO

SEGUNDO MAS VENDIDO ALTERNATIVA 1

ISNULL((--LE PONGO ISNULL porque el segundo puede ser nulo

```
SELECT TOP 1 AliasSubselect.prod_codigo
  from (
    SELECT TOP 2 P4.prod_codigo, SUM(I4.item_cantidad*I4.item_precio) as SUMA
      FROM Item_Factura I4
      INNER JOIN Producto P4 ON P4.prod_codigo = I4.item_producto
      WHERE P4.prod_rubro = R.rubr_id
      GROUP BY P4.prod_codigo
      ORDER BY SUM(I4.item_cantidad*I4.item_precio) DESC
  ) AliasSubselect -- TODOS LOS SUBSELECT EN TI FROM TITENEN QUE TENER UN NOMBRE
  order by SUMA
), '000000'
AS segundoMasVendido,
```

Es mejor la Alt 2 porque en el caso que haya solo un producto vendido

Alt① Cliente | 12 | 23 | 34 | 45 | 56 | 67 | 78 | 89 | 90 | 000000
Alt② Cliente | 12 | 23 | 34 | 45 | 56 | 67 | 78 | 89 | 90 | 000000

SEGUNDO MAS VENDIDO ALTERNATIVA 3

ISNULL((--LE PONGO ISNULL porque el segundo puede ser nulo

```
SELECT AliasSubselect.prod_codigo
  from (
    SELECT ROW_NUMBER() OVER(ORDER BY SUM(I5.item_cantidad*I5.item_precio) DESC) AS orden, p5.prod_codigo
      FROM Item_Factura I5
      INNER JOIN Producto P5 ON P5.prod_codigo = I5.item_producto
      WHERE P5.prod_rubro = R.rubr_id
      GROUP BY P5.prod_codigo
  ) AliasSubselect -- TODOS LOS SUBSELECT EN TI FROM TITENEN QUE TENER UN NOMBRE
  WHERE orden = 2
), '000000'
AS segundoMasVendidoForma2,
```

rownumber le
pone numero a las filas

```
ISNULL(
(
    SELECT TOP 1 F3.fact_cliente
      FROM Item_Factura I3
      INNER JOIN Producto P3 ON P3.prod_codigo = I3.item_producto
      INNER JOIN Factura F3 ON f3.fact_tipo = i3.item_tipo
      AND f3.fact_sucursal = i3.item_sucursal AND f3.fact_numero = i3.item_numero
      WHERE DATEDIFF(DAY,F3.fact_fecha,GETDATE()) < 31 AND P3.prod_rubro=R.rubr_id
      GROUP BY F3.fact_cliente
      ORDER BY SUM(I3.item_cantidad) DESC
)
), '0')
AS clienteMasComprador

FROM Item_Factura I
INNER JOIN Producto P ON P.prod_codigo = I.item_producto
INNER JOIN Rubro R ON R.rubr_id=P.prod_rubro
GROUP BY r.rubr_id,r.rubr_detalle
ORDER BY COUNT(DISTINCT P.prod_codigo)
```

Para los últimos
30 días

