



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

FUNDAMENTOS DE BASES DE DATOS - 7094

T A R E A 4

EQUIPO:

DEL MONTE ORTEGA MARYAM MICHELLE - 320083527

SOSA ROMO JUAN MARIO - 320051926

CASTILLO HERNÁNDEZ ANTONIO - 320017438

ERIK EDUARDO GÓMEZ LÓPEZ - 320258211

JULIO CÉSAR ISLAS ESPINO - 320340594

FECHA DE ENTREGA:
14 DE OCTUBRE DE 2024

PROFESOR:
M. EN I. GERARDO AVILÉS ROSAS

AYUDANTES:
LUIS ENRIQUE GARCÍA GÓMEZ
KEVIN JAIR TORRES VALENCIA
RICARDO BADILLO MACÍAS
ROCÍO AYLIN HUERTA GONZÁLEZ



Tarea 4

Preguntas

1. Cardinalidad de la consulta

Considera las siguientes relaciones:

A	B
1	x
2	y
2	z
3	x
9	a

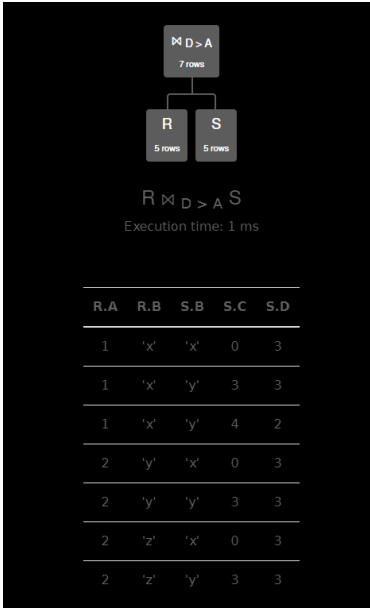
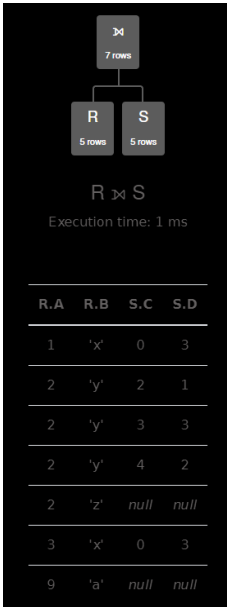
Tabla 1: R

B	C	D
x	0	3
y	2	1
y	3	3
w	3	0
y	4	2

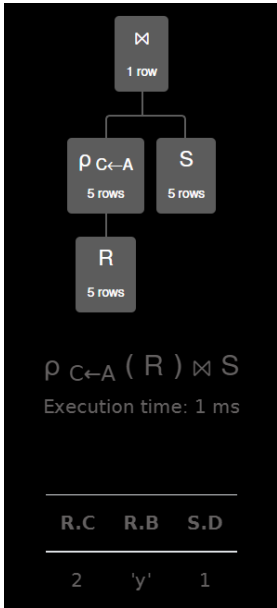
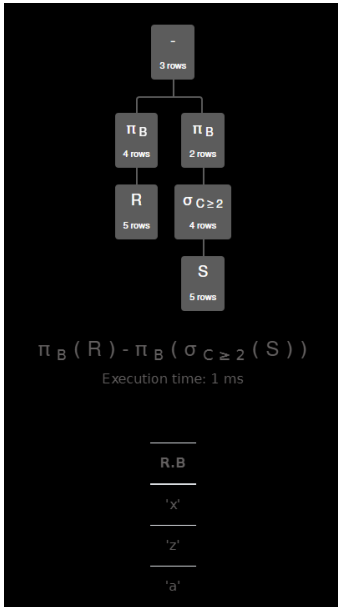
Tabla 2: S

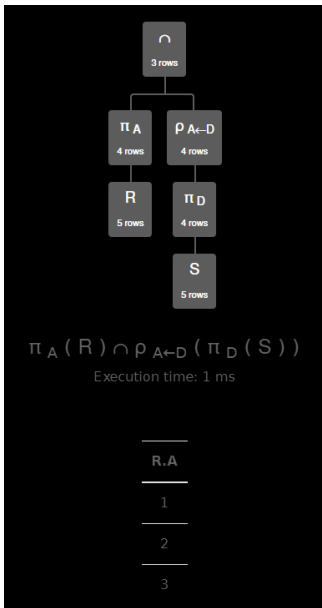
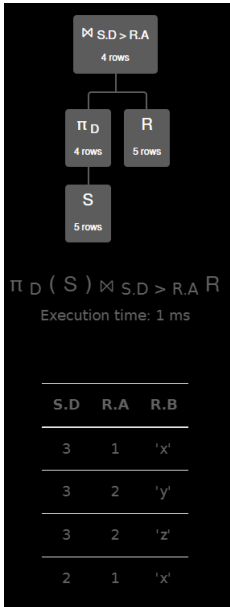
Para las siguientes expresiones de álgebra relacional, completa la tabla con el número de tuplas que cada una de ellas produce utilizando las relaciones R y S. Deberás indicar las tablas resultantes en cada caso.

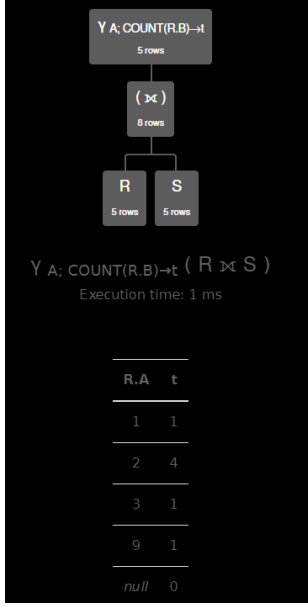
Expresión	Cardinalidad del resultado
$R \times S$	<p>Al ser un producto cartesiano se tienen $5 \times 5 = 25$ tuplas.</p> 

$R \bowtie_{D > A} S$	<p>Esta hace un join natural donde D es mayor que A, resulta en 7 tuplas :</p> 
$R \bowtie S$	<p>Se selecciona la relación R y se junta con S usando su columna en comun, si no hay coincidencia en S se añade un null, resultando en 7 tuplas:</p> 

$R \bowtie S$	<p>Se selecciona la relación S y se junta con R usando su columna en comun, si no hay coincidencia en R se añade un null, resultando en 6 tuplas:</p> <div><div><div><div>⋈</div><div>6 rows</div></div><div><div>R</div><div>5 rows</div></div><div><div>S</div><div>5 rows</div></div></div><div><div>$R \bowtie S$</div><div>Execution time: 1 ms</div></div><table><thead><tr><th>R.A</th><th>S.B</th><th>S.C</th><th>S.D</th></tr></thead><tbody><tr><td>1</td><td>'x'</td><td>0</td><td>3</td></tr><tr><td>3</td><td>'x'</td><td>0</td><td>3</td></tr><tr><td>2</td><td>'y'</td><td>2</td><td>1</td></tr><tr><td>2</td><td>'y'</td><td>3</td><td>3</td></tr><tr><td>null</td><td>'w'</td><td>3</td><td>0</td></tr><tr><td>2</td><td>'y'</td><td>4</td><td>2</td></tr></tbody></table></div>	R.A	S.B	S.C	S.D	1	'x'	0	3	3	'x'	0	3	2	'y'	2	1	2	'y'	3	3	null	'w'	3	0	2	'y'	4	2		
R.A	S.B	S.C	S.D																												
1	'x'	0	3																												
3	'x'	0	3																												
2	'y'	2	1																												
2	'y'	3	3																												
null	'w'	3	0																												
2	'y'	4	2																												
$R \bowtie_{A=D} S$	<p>Al ser un theta join, se seleccionan las tuplas que cumplan con la condición, en este caso que A sea igual a D (por cada de A buscas cuantos D son iguales), se obtienen 5 tuplas:</p> <div><div><div><div>⋈_{A=D}</div><div>5 rows</div></div><div><div>R</div><div>5 rows</div></div><div><div>S</div><div>5 rows</div></div></div><div><div>$R \bowtie_{A=D} S$</div><div>Execution time: 1 ms</div></div><table><thead><tr><th>R.A</th><th>R.B</th><th>S.B</th><th>S.C</th><th>S.D</th></tr></thead><tbody><tr><td>1</td><td>'x'</td><td>'y'</td><td>2</td><td>1</td></tr><tr><td>2</td><td>'y'</td><td>'y'</td><td>4</td><td>2</td></tr><tr><td>2</td><td>'z'</td><td>'y'</td><td>4</td><td>2</td></tr><tr><td>3</td><td>'x'</td><td>'x'</td><td>0</td><td>3</td></tr><tr><td>3</td><td>'x'</td><td>'y'</td><td>3</td><td>3</td></tr></tbody></table></div>	R.A	R.B	S.B	S.C	S.D	1	'x'	'y'	2	1	2	'y'	'y'	4	2	2	'z'	'y'	4	2	3	'x'	'x'	0	3	3	'x'	'y'	3	3
R.A	R.B	S.B	S.C	S.D																											
1	'x'	'y'	2	1																											
2	'y'	'y'	4	2																											
2	'z'	'y'	4	2																											
3	'x'	'x'	0	3																											
3	'x'	'y'	3	3																											

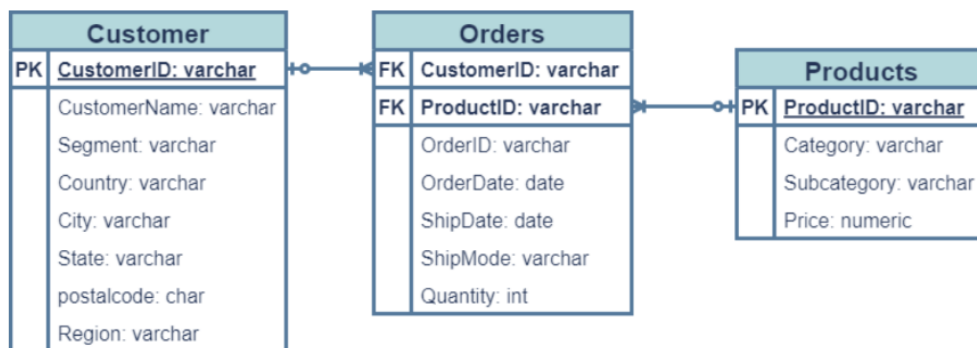
$\rho_{C \leftarrow A} R \bowtie S$	<p>Se renombra la columna A de R a C y se hace un join natural con S, se regresa donde C sea igual a A y como comparten B tambien debe ser igual, resultando en 1 tupla:</p> <div><pre>graph TD Join["⋈ 1 row"] --- Rename["ρ_{C ← A} 5 rows"] Join --- S["S 5 rows"] Rename --- R["R 5 rows"]</pre><p>$\rho_{C \leftarrow A} (R) \bowtie S$ Execution time: 1 ms</p><table><tr><th>R.C</th><th>R.B</th><th>S.D</th></tr><tr><td>2</td><td>'y'</td><td>1</td></tr></table></div>	R.C	R.B	S.D	2	'y'	1
R.C	R.B	S.D					
2	'y'	1					
$\pi_B(R) - \pi_B(\sigma_{C \geq 2}(S))$	<p>Se selecciona de S las tuplas donde C es mayor o igual a 2, se seleccionan las diferentes B de la consulta anterior; se toman las diferentes B de R y se restan la primera consulta, resultando en 3 tuplas:</p> <div><pre>graph TD Minus["- 3 rows"] --- PiB1["π_B 4 rows"] Minus --- PiB2["π_B 2 rows"] PiB1 --- R["R 5 rows"] PiB2 --- Sigma["σ_{C ≥ 2} 4 rows"] Sigma --- S["S 5 rows"]</pre><p>$\pi_B (R) - \pi_B (\sigma_{C \geq 2} (S))$ Execution time: 1 ms</p><table><tr><th>R.B</th></tr><tr><td>'x'</td></tr><tr><td>'z'</td></tr><tr><td>'a'</td></tr></table></div>	R.B	'x'	'z'	'a'		
R.B							
'x'							
'z'							
'a'							

$\pi_A(R) \cap \rho_{A \leftarrow D} (\pi_D(S))$	<p>Selecciona las diferentes A de R y se intersecan con las diferentes D de S ahora renombradas a A, resultando en 3 tuplas:</p> <div><p>Execution time: 1 ms</p><table><tr><th>R.A</th></tr><tr><td>1</td></tr><tr><td>2</td></tr><tr><td>3</td></tr></table></div>	R.A	1	2	3											
R.A																
1																
2																
3																
$\pi_D(S) \bowtie_{S.D > R.A} R$	<p>Selecciona las diferentes D de S y se hace un join natural con S, se seleccionan las tuplas donde D es mayor que A de R, resultando en 4 tuplas:</p> <div><p>Execution time: 1 ms</p><table><tr><th>S.D</th><th>R.A</th><th>R.B</th></tr><tr><td>3</td><td>1</td><td>'x'</td></tr><tr><td>3</td><td>2</td><td>'y'</td></tr><tr><td>3</td><td>2</td><td>'z'</td></tr><tr><td>2</td><td>1</td><td>'x'</td></tr></table></div>	S.D	R.A	R.B	3	1	'x'	3	2	'y'	3	2	'z'	2	1	'x'
S.D	R.A	R.B														
3	1	'x'														
3	2	'y'														
3	2	'z'														
2	1	'x'														

$\gamma_A; \text{count}(B) \rightarrow t(R \bowtie S)$	<p>Empezamos haciendo el natural join de R y S donde B es igual, si falta alguno ponemos null, agrupamos las filas resultantes por el atributo A de R usando la funcion de agregacion que cuenta el numero de ocurrencias de cada valor B en la relacion R (B era ambiguo pues ambos tienen B) por cada grupo de A, el resultado se guarda en una columna llamada t, al final salen 5 tuplas: (una por cada tipo de A)</p>  <p>Execution time: 1 ms</p> <table border="1"> <thead> <tr> <th>R.A</th> <th>t</th> </tr> </thead> <tbody> <tr><td>1</td><td>1</td></tr> <tr><td>2</td><td>4</td></tr> <tr><td>3</td><td>1</td></tr> <tr><td>9</td><td>1</td></tr> <tr><td>null</td><td>0</td></tr> </tbody> </table>	R.A	t	1	1	2	4	3	1	9	1	null	0
R.A	t												
1	1												
2	4												
3	1												
9	1												
null	0												

2. Tienda de productos en línea.

Tienes el siguiente esquema de una base de datos para una tienda en línea (ID gist: 31074567738afef8c497f6ca89335782)



Escribe una expresión de álgebra relacional para responder las siguientes consultas. Deberás comprobar cada una ellas en la calculadora Relax y agregar para cada inciso la expresión en álgebra relacional y una captura de pantalla con el resultado obtenido (no es necesario mostrar todas las tuplas):

- a. Obtener toda la información de los clientes que viven en Seattle o en San Francisco, que pertenezcan al segmento corporate que hayan solicitado una orden en el segundo trimestre de 2014. Mostrar la información ordenada por la cantidad solicitada.

La verdad es que la sintaxis de la pregunta deja poco claro lo que se necesita pero yo lo interprete de la siguiente manera:

```
1 C = σ segment='Corporate' AND (city='Seattle' OR city='San Francisco') (customer)
2 D = σ orderdate >= date('2014-04-01') and orderdate <= date('2014-06-30') orders
3 A = (C ⋈ D)
4 O = τ quantity A
5 π customerid, customername, segment, country, city, state, postalcode, region O
6
```

La idea es la siguiente, comenzamos por seleccionar a los clientes que viven en Seattle o en San Francisco, que pertenezcan al segmento corporate, eso es una mini consulta. Luego seleccionamos las ordenes que se hicieron en el segundo trimestre de 2014, eso es otra mini consulta. Luego unimos ambas consultas, finalmente ordenamos por la cantidad solicitada, aqui es donde no estoy seguro de si ahi acaba nuestra consulta o, como dice que solo quiere la información de los clientes tenemos que solo regresarle las columnas relevantes (si no hacemos el siguiente paso se van a repetir clientes porque tienen varios pedidos, igual otra opcion es agrupar por cliente), entonces decidi incluir esta ultima aunque si no es necesario podriamos parar ahi, entonces solo proyectamos sobre las columnas de customer sobre la consulta anterior.

Esto nos regresa la siguiente tabla:

customerid	customername	customersegment	customercountry	customercity	customerstate	customerpostalcode	customerregion
'KL-16555'	'Kelly Lampkin'	'Corporate'	'United States'	'San Francisco'	'California'	'94110'	'West'
'JM-15655'	'Jim Mitchum'	'Corporate'	'United States'	'Seattle'	'Washington'	'98103'	'West'
'ML-17395'	'Marina Lichtenstein'	'Corporate'	'United States'	'San Francisco'	'California'	'94109'	'West'
'MH-17785'	'Maja Herman'	'Corporate'	'United States'	'Seattle'	'Washington'	'98105'	'West'
'GP-14740'	'Guy Phoney'	'Corporate'	'United States'	'Seattle'	'Washington'	'98103'	'West'

Como se ve esto regresa toda la información de los clientes en la forma que se solicita con el detalle de no incluir detalle de la orden.

- b. Obtener una relación de los productos que pertenecen a la categoría Office Supplies con precio mayor de \$300 y menor de \$600, pero que no hayan sido solicitados en ninguna orden.

```

1 r = σ category = 'Office Supplies' ∧ price > 300 ∧ price < 600
  (products)
2 s = (π productid (orders))
3 r ⋈ s productid, category, subcategory, price (r⋈s)

```

Esta consulta resulta de la abstracción de algunos conceptos:

- **r** tabla de products con las condiciones solicitadas de precio mayor de \$300 y menor de \$600
- **s** es la tabla de los ids de productos que han sido solicitados en alguna orden
- $r \bowtie s$ es la tabla de productos de la categoría Office Supplies con precio mayor de 300 y menor de 600 y que no coinciden o no están dentro de la tabla de ids de productos solicitados en alguna orden.

El resultado es:

products.productid	products.category	products.subcategory	products.price
'OFF-PA-10001593'	'Office Supplies'	'Paper'	563.4
'OFF-PA-10002109'	'Office Supplies'	'Paper'	505.18
'OFF-PA-10003205'	'Office Supplies'	'Paper'	478.48
'OFF-AP-10003278'	'Office Supplies'	'Appliances'	597.13
'OFF-AR-10003896'	'Office Supplies'	'Art'	462.56



- c. Obtener el nombre de todos los clientes que vivan en la región West y hayan solicitado productos de las categorías Technology o Furniture. El pedido debió de solicitarse en 2106 y el modo de envío debe ser Standard Class.

Asumiré que cuando dice "El pedido debió de solicitarse en 2106" se refiere al 2016.

```

1 r = σ region = 'West' (customer)
2 s = σ category = 'Technology' ∨ category = 'Furniture' (products)
3 q = σ orderdate >= date('2016-01-01') ∧ orderdate <= date('2016-12-31')
  ∧ shipmode = 'Standard Class' (orders)
4 t = r ⋈ q
5 π customername (t⋈s)

```

Para esta consulta

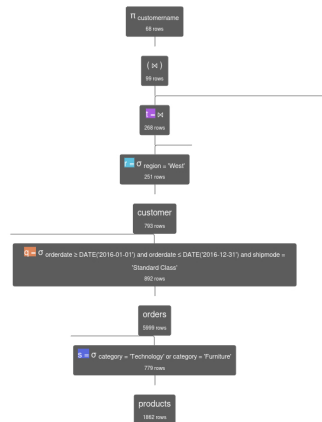
- **r** es la tabla de clientes de la región West
- **s** es la tabla de productos de categoría Technology o Furniture
- **q** es la tabla de pedidos enviados en modo Standard Class en el 2016
- **t** es la relación de clientes que viven en West que hicieron un pedido en modo Standard Class en el 2016
- **t join s** es la relación de clientes que viven en West que hicieron un pedido en modo Standard Class en el 2016, y además su producto es de categoría Technology o Furniture

El resultado es:

customer.customername
'Brosina Hoffman'
'Zuschuss Donatelli'
'Emily Burns'
'Eric Hoffmann'
'Lena Creighton'
'Jonathan Doherty'
'Nora Paige'
'Chad Sievert'
'Jennifer Braxton'
'Jonathan Howell'

< 1 2 3 >

Nótese que primero hacemos join de r con q , ya que el atributo en común es `customerid`, de otra forma obtendríamos el producto cartesiano, lo cuál es demasiado costoso en términos de recursos.



- d. Toda la información de los clientes del segmento **Corporate** que realizaron una orden con modo de envío **First Class** y que no viven en **California**.

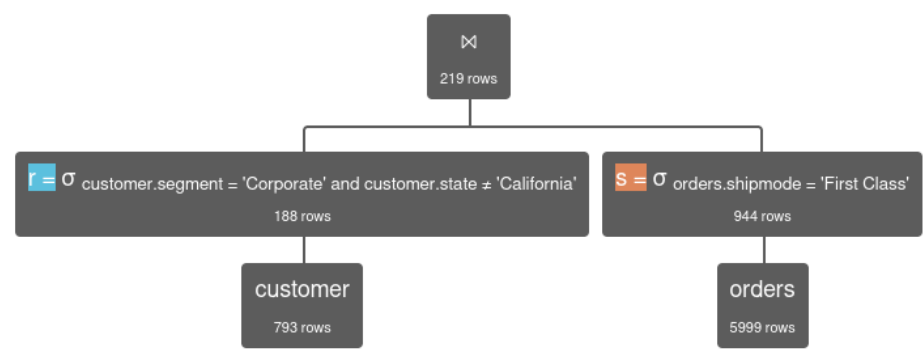
En primera instancia, *seleccionamos* los clientes que pertenecen al segmento **Corporate** y que no viven en **California** de la relación *customer*, almacenando dichas tuplas en una relación temporal r . Luego, seleccionamos las órdenes cuyo modo de envío es **First Class** de la relación *orders*. Finalmente, realizamos un natural join entre las dos relaciones resultantes para obtener lo que nos pide el inciso.

```

1 r = sigma customer.segment = 'Corporate' ^ customer.state != 'California' (customer)
2 s = sigma orders.shipmode = 'First Class' (orders)
3
4 r ⋈ s

```

Ahora el arbol de la consulta se ve de la siguiente manera:



y parte de la tabla resultante es la siguiente:

customer.customerid	customer.customername	customer.segment	customer.country	customer.city	customer.state
'KB-16585'	'Ken Black'	'Corporate'	'United States'	'Fremont'	'Nebraska'
'KB-16585'	'Ken Black'	'Corporate'	'United States'	'Fremont'	'Nebraska'
'KB-16585'	'Ken Black'	'Corporate'	'United States'	'Fremont'	'Nebraska'
'GH-14485'	'Gene Hale'	'Corporate'	'United States'	'Richardson'	'Texas'
'GH-14485'	'Gene Hale'	'Corporate'	'United States'	'Richardson'	'Texas'
'LC-16930'	'Linda Cazamias'	'Corporate'	'United States'	'Naperville'	'Illinois'
'LC-16930'	'Linda Cazamias'	'Corporate'	'United States'	'Naperville'	'Illinois'
'ES-14080'	'Erin Smith'	'Corporate'	'United States'	'Melbourne'	'Florida'
'ES-14080'	'Erin Smith'	'Corporate'	'United States'	'Melbourne'	'Florida'
'ES-14080'	'Erin Smith'	'Corporate'	'United States'	'Melbourne'	'Florida'

customer.state	customer.postalcode	customer.region	orders.orderid	orders.orderdate	orders.shipdate	orders
Nebraska	'68025'	'Central'	'CA-2015-138674'	2015-11-14	2015-11-17	'Firm'
Nebraska	'68025'	'Central'	'CA-2015-138674'	2015-11-14	2015-11-17	'Firm'
Nebraska	'68025'	'Central'	'CA-2015-138674'	2015-11-14	2015-11-17	'Firm'
Texas	'75080'	'Central'	'CA-2016-117590'	2016-12-08	2016-12-10	'Firm'
Texas	'75080'	'Central'	'CA-2016-117590'	2016-12-08	2016-12-10	'Firm'
Illinois	'60540'	'Central'	'CA-2016-152289'	2016-08-26	2016-08-28	'Firm'
Illinois	'60540'	'Central'	'CA-2016-152289'	2016-08-26	2016-08-28	'Firm'
Florida	'32935'	'South'	'CA-2016-136924'	2016-07-14	2016-07-17	'Firm'
Florida	'32935'	'South'	'CA-2014-108189'	2014-10-02	2014-10-05	'Firm'
Florida	'32935'	'South'	'CA-2014-108189'	2014-10-02	2014-10-05	'Firm'

- e. Obtener el **estado**, **segmento** y el **total de clientes** que no han solicitado **ninguna orden**.

Dado esto, *seleccionamos* todos los clientes de la relación *customer*, almacenando dichas tuplas en una relación temporal *clientes*. Luego, seleccionamos los IDs de los clientes que han realizado órdenes de la relación *orders*, almacenando dichas tuplas en una relación temporal *clientes_con_ordenes*. A continuación, obtenemos los IDs de los clientes que no han realizado ninguna orden mediante la operación de diferencia de conjuntos entre *clientes* y *clientes_con_ordenes*, almacenando el resultado en relación temporal *clientes_sin_ordenes*. Finalmente, utilizamos el operador de proyección para obtener el **estado**, **segmento** y el **total de clientes** que no han solicitado **ninguna orden**.

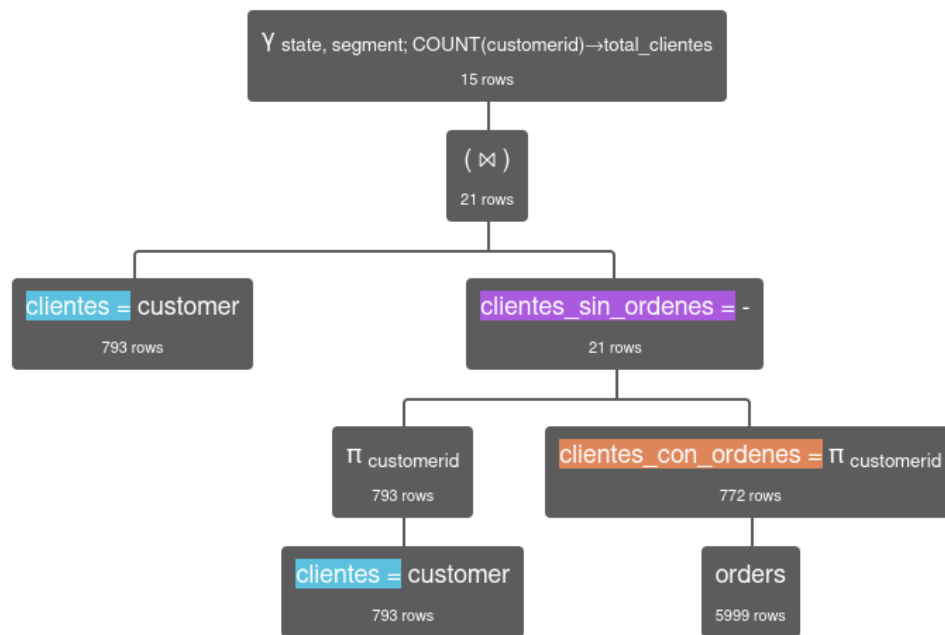
La consulta en álgebra relacional se vería mas o menos de la siguiente manera:

```

1 clientes = customer
2 clientes_con_ordenes =  $\pi$  customerid (orders)
3
4 clientes_sin_ordenes =  $\pi$  customerid (clientes) - clientes_con_ordenes
5
6  $\gamma$  state, segmento; COUNT(customerid)  $\rightarrow$  total_clientes (clientes  $\bowtie$  clientes_sin_ordenes)

```

El árbol de la consulta:



y la tabla resultante:

customer.state	customer.segment	total_clientes
'California'	'Consumer'	3
'Illinois'	'Home Office'	1
'Ohio'	'Corporate'	1
'New York'	'Corporate'	3
'Texas'	'Home Office'	1
'Delaware'	'Corporate'	2
'Tennessee'	'Corporate'	1
'California'	'Corporate'	1
'Pennsylvania'	'Corporate'	1
'Minnesota'	'Home Office'	1

- f. Una lista que muestre la región, el estado y el total de clientes que se tienen, considerando que los clientes deben haber realizado órdenes con al menos 6 productos durante 2014 o 2015. Ordenar la información por región y estado..

Primero el Álgebra Relacional es:

Select DB (BDVentas) ▾

Álgebra Relacional
SQL
Editor de Grupo

orders

- orderid string
- orderdate date
- shipdate date
- shipmode string
- customerid string
- productid string
- quantity number

products

- productid string
- category string
- subcategory string
- price number

customer

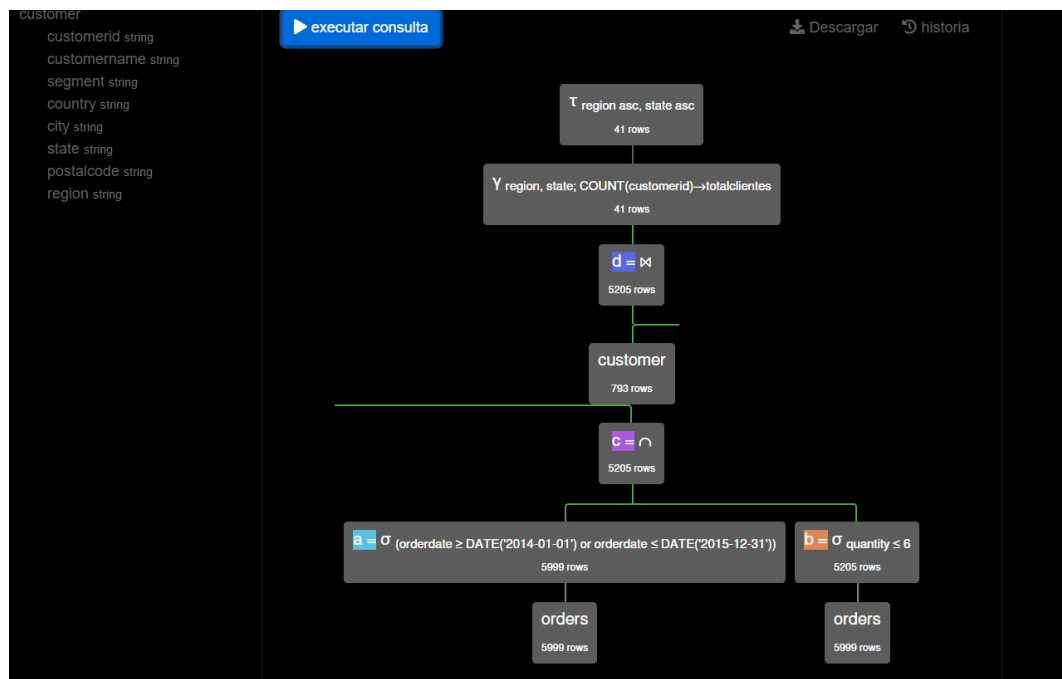
- customerid string
- customername string

```

1 a = σ (orderdate ≥ date('2014-01-01') ∨ orderdate ≤ date('2015-12-31'))
   (orders)
2
3 b = σ quantity ≤ 6(orders)
4
5 c = a ∩ b
6
7 d = customer ⋈ c
8
9 τ region, state(γ region, state; count(customerid) → totalclientes(d))
10
        
```

⬇ Descargar
🕒 historia

El resultado de las tablas es:



$\tau_{\text{region asc, state asc}} (\gamma_{\text{region, state; COUNT(customerid)} \rightarrow \text{totalclientes}} (\text{customer} \bowtie (\sigma_{\text{orderdate} \geq \text{DATE('2014-01-01')} \text{ or } \text{orderdate} \leq \text{DATE('2015-12-31')}}) (\text{orders}) \cap \sigma_{\text{quantity} \leq 6} (\text{orders}))))$		
Tiempo de consulta 2 ms		
customer.region	customer.state	totalclientes
'Central'	'Illinois'	282
'Central'	'Indiana'	58
'Central'	'Iowa'	44
'Central'	'Michigan'	179
'Central'	'Minnesota'	112
'Central'	'Missouri'	35
'Central'	'Nebraska'	28
'Central'	'Oklahoma'	11
'Central'	'Texas'	482
'Central'	'Wisconsin'	47

- g. **Obtener el modo de envío y categoría que más productos ha vendido.**

El enunciado es bastante ambiguo. No hay muchos detalles así que realizamos lo siguiente:

Obtendremos las categorías y modos de envío que mas productos han vendido combinaciones diferentes

```

1 agrupado =  $\gamma$  shipmode, category; sum(quantity)  $\rightarrow$  total_vendido
  (orders $\bowtie$ products)
2
3  $\pi$  shipmode, category, total_vendido ( $\tau$  total_vendido desc (agrupado))
4 |

```

El resultado:

orders.shipmode	products.category	total_vendido
'Standard Class'	'Office Supplies'	8278
'Standard Class'	'Furniture'	2799
'Second Class'	'Office Supplies'	2706
'Standard Class'	'Technology'	2345
'First Class'	'Office Supplies'	2114
'Second Class'	'Furniture'	948
'Second Class'	'Technology'	891
'Same Day'	'Office Supplies'	708
'First Class'	'Technology'	702
'First Class'	'Furniture'	693

2

- h. Una tabla con la venta promedio, venta total, mayor venta, menor venta, y total de órdenes, por región, estado y ciudad. La venta promedio debe estar entre \$900 y \$1,500.

```
1. π customer.region, customer.state, customer.city, venta_promedio, venta_total,
   mayor_venta, menor_venta, total_ordenes σ venta_promedio ≥ 900 and venta_promedio ≤
   1500 γ customer.region, customer.state, customer.city;
   AVG(products.price)→venta_promedio, SUM(products.price)→venta_total,
   MAX(products.price)→mayor_venta, MIN(products.price)→menor_venta,
   COUNT(*)→total_ordenes ( ( customer ⋈ orders ) ⋈ products )
```



```

Π customer.region, customer.state, customer.city, venta_promedio, venta_total, mayor_venta,
  menor_venta, total_ordenes
σ venta_promedio ≥ 900 and venta_promedio ≤ 1500
Y
customer.region, customer.state, customer.city; AVG(products.price)→venta_promedio,
  SUM(products.price)→venta_total, MAX(products.price)→mayor_venta,
  MIN(products.price)→menor_venta, COUNT(*)→total_ordenes ( ( customer ⋈ orders ) ⋈
    products )

```

Tiempo de consulta 5 ms

customer.region	customer.state	customer.city	venta_promedio	venta_total	mayor_venta	menor_venta	total_ordenes
'East'	'New Jersey'	'Morristown'	971.3562499999999	7770.849999999999	3083.43	5.96	8

```

Π customer.region, customer.state, customer.city, venta_promedio, venta_total, mayor_venta,
  menor_venta, total_ordenes
σ venta_promedio ≥ 900 and venta_promedio ≤ 1500
Y
customer.region, customer.state, customer.city; AVG(products.price)→venta_promedio,
  SUM(products.price)→venta_total, MAX(products.price)→mayor_venta,
  MIN(products.price)→menor_venta, COUNT(*)→total_ordenes ( ( customer ⋈ orders ) ⋈
    products )

```

Tiempo de consulta 5 ms

customer.city	venta_promedio	venta_total	mayor_venta	menor_venta	total_ordenes
'Morristown'	971.3562499999999	7770.849999999999	3083.43	5.96	8

- i. El estado que ha realizado la mayor cantidad de órdenes. Se debe mostrar también el total de órdenes que haya entregado.

```

1 r = γ customer.state; COUNT(*)→total_ordenes (customer ⋈ orders)
2 s = γ MAX(total_ordenes) → maximo r ⋈ maximo = total_ordenes r
3 π total_ordenes, customer.state s

```

```

π total_ordenes, customer.state ( Y ; MAX(total_ordenes)→maximo Y customer.state;
COUNT(*)→total_ordenes ( customer ⋈ orders ) ⋈ maximo = total_ordenes Y
customer.state; COUNT(*)→total_ordenes ( customer ⋈ orders ) )

```

Tiempo de consulta 4 ms

total_ordenes	customer.state
1200	'California'

< 1 >

- j. La información del cliente que menos órdenes haya efectuado. Mostrar el número de órdenes que ha realizado.

```

1 r = γ customer.customerid; COUNT(*)→total_ordenes (customer ⋈ orders)
2 s = γ MIN(total_ordenes) → minimo r ⋈ minimo = total_ordenes r
3 π total_ordenes, customer.customerid s ⋈ customer

```

total_ordenes	customer.customerid	customer.customername	customer.segment	customer.country	customer.city
1	'HM-14980'	'Henry MacAllister'	'Consumer'	'United States'	'New York City'
1	'KH-16360'	'Katherine Hughes'	'Consumer'	'United States'	'San Francisco'
1	'TB-21190'	'Thomas Brumley'	'Home Office'	'United States'	'Los Angeles'
1	'PM-18940'	'Paul MacIntyre'	'Consumer'	'United States'	'New Brunswick'
1	'JR-15700'	'Jocasta Rupert'	'Consumer'	'United States'	'Detroit'
1	'GZ-14545'	'George Zrebassa'	'Corporate'	'United States'	'Norwich'
1	'PW-19030'	'Pauline Webber'	'Corporate'	'United States'	'New York City'
1	'BO-11425'	'Bobby Odegard'	'Consumer'	'United States'	'Philadelphia'
1	'PS-18760'	'Pamela Stobb'	'Consumer'	'United States'	'Seattle'
1	'BT-11440'	'Bobby Trafton'	'Consumer'	'United States'	'Long Beach'

< 1 2 3 >

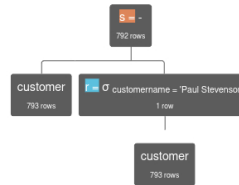
3. Operaciones de mantenimiento de datos: borrado, inserción y actualización

a. Borrar toda la información del cliente Paul Stevenson.

Para esta consulta en algebra relacional solo deberemos descartar cualquier fila que coincida con Paul Stevenson

```
1 r = σ customername = 'Paul Stevenson' (customer)
2 s = customer - r
3 s
4
5
```

Así obtenemos el siguiente árbol de consulta



Y el resultado:

customer.customerid	customer.customername	customer.segment	customer.country	customer.city
'CG-12520'	'Claire Gute'	'Consumer'	'United States'	'San Francisco'
'DV-13045'	'Darrin Van Huff'	'Corporate'	'United States'	'San Francisco'
'SO-20335'	'Sean ODonnell'	'Consumer'	'United States'	'San Francisco'
'BH-11710'	'Brosina Hoffman'	'Consumer'	'United States'	'San Francisco'
'AA-10480'	'Andrew Allen'	'Consumer'	'United States'	'San Francisco'
'IM-15070'	'Irene Maddox'	'Consumer'	'United States'	'San Francisco'
'HP-14815'	'Harold Pawlan'	'Home Office'	'United States'	'San Francisco'
'PK-19075'	'Pete Kriz'	'Consumer'	'United States'	'San Francisco'
'AG-10270'	'Alejandro Grove'	'Consumer'	'United States'	'San Francisco'
'ZD-21925'	'Zuschuss Donatelli'	'Consumer'	'United States'	'San Francisco'

Si lo hacemos con sql obtenemos los mismos resultados

```
select from where group having order limit
1 SELECT * FROM customer
2 WHERE customername <> 'Paul Stevenson';
```

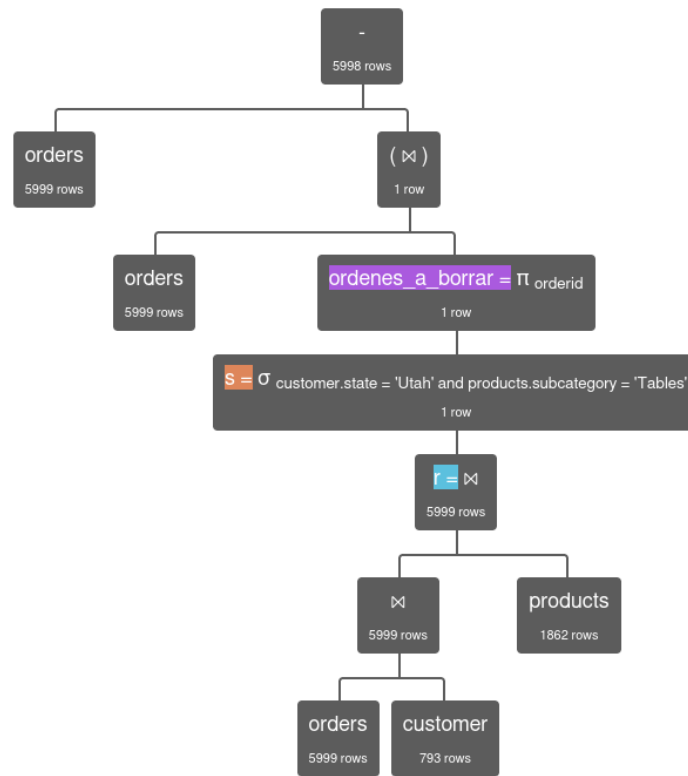
b. Borrar todas las órdenes de la ciudad Utah que tengan artículos de la subcategoría **Tables**.

En este caso, lo primero que hacemos es realizar un natural join entre *orders*, *customer* y *products*, almacenando dicho join en una relación temporal *r*. Luego, seleccionamos las tuplas que sean de la ciudad **Utah** y que tengan artículos de la subcategoría **Tables**, guardando estas tuplas en otra relación temporal *s*. A continuación, hacemos una proyección de los IDs de la relación *s* creada anteriormente (llamamos a esto *ordenes_a_borrar*). Finalmente, hacemos una diferencia entre *orders* y el join natural derivado de *orders* y *ordenes_a_borrar*.

La consulta en algebra relacional se veria de la siguiente manera:

```
1 r = orders ⋈ customer ⋈ products
2 s = σ customer.state = 'Utah' ∧ products.subcategory = 'Tables' (r)
3 ordenes_a_borrar = π orderid (s)
4 orders - (orders ⋈ ordenes_a_borrar)
```

Ahora el árbol de la consulta:

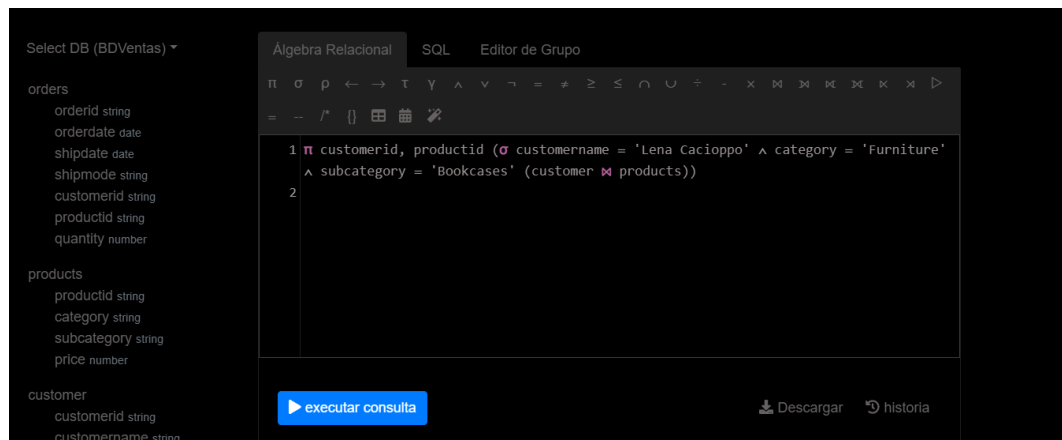


y forma de la tabla resultante:

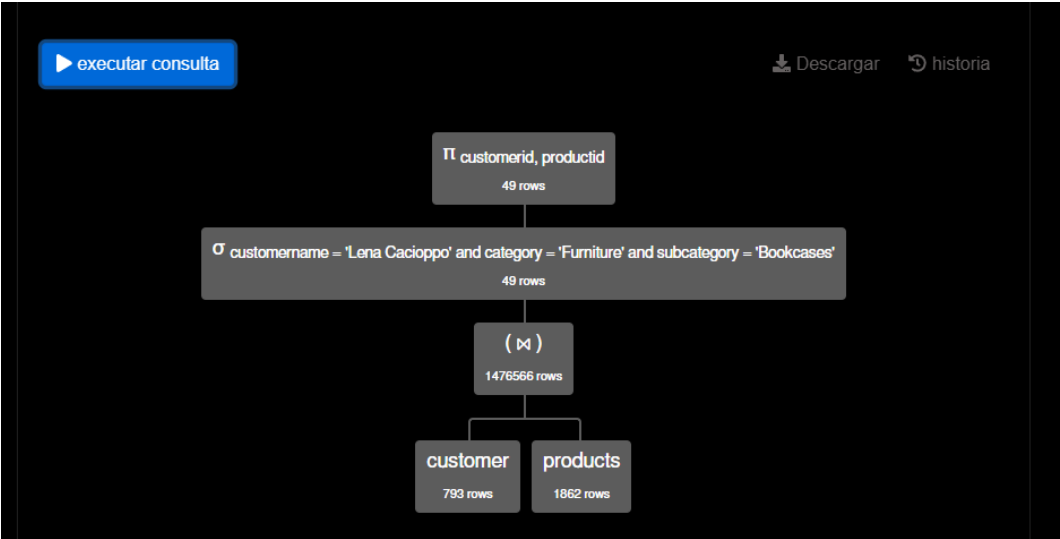
orders.orderid	orders.orderdate	orders.shipdate	orders.shipmode	orders.customerid	orders.productid
'CA-2016-152156'	2016-11-08	2016-11-11	'Second Class'	'CG-12520'	'FUR-BO-10001798'
'CA-2016-152156'	2016-11-08	2016-11-11	'Second Class'	'CG-12520'	'FUR-CH-10000454'
'CA-2016-138688'	2016-06-12	2016-06-16	'Second Class'	'DV-13045'	'OFF-LA-10000240'
'US-2015-108966'	2015-10-11	2015-10-18	'Standard Class'	'SO-20335'	'FUR-TA-10000577'
'US-2015-108966'	2015-10-11	2015-10-18	'Standard Class'	'SO-20335'	'OFF-ST-10000760'
'CA-2014-115812'	2014-06-09	2014-06-14	'Standard Class'	'BH-11710'	'FUR-FU-10001487'
'CA-2014-115812'	2014-06-09	2014-06-14	'Standard Class'	'BH-11710'	'OFF-AR-10002833'
'CA-2014-115812'	2014-06-09	2014-06-14	'Standard Class'	'BH-11710'	'TEC-PH-10002275'
'CA-2014-115812'	2014-06-09	2014-06-14	'Standard Class'	'BH-11710'	'OFF-BI-10003910'
'CA-2014-115812'	2014-06-09	2014-06-14	'Standard Class'	'BH-11710'	'OFF-AP-10002892'

- c. La clienta Lena Cacioppo compró un producto de cada subcategoría de Furniture. Deberás elegir los productos que desees e indicar como parte de esta consulta, la información que se agregará en cada caso.

Primero el Álgebra Relacional es:



El resultado de las tablas es:



Π customerid, productid (σ customername = 'Lena Cacioppo' and category = 'Furniture' and subcategory = 'Bookcases' (customer \Join products))

Tiempo de consulta 1 ms

customer.customerid	products.productid
'LC-16870'	'FUR-BO-10001798'
'LC-16870'	'FUR-BO-10004834'
'LC-16870'	'FUR-BO-10002545'
'LC-16870'	'FUR-BO-10002613'
'LC-16870'	'FUR-BO-10004695'
'LC-16870'	'FUR-BO-10004709'
'LC-16870'	'FUR-BO-10002268'
'LC-16870'	'FUR-BO-10001972'
'LC-16870'	'FUR-BO-10002824'
'LC-16870'	'FUR-BO-10001601'

d.

Aumentar los precios de productos de la subcategoría Phones en un 8%.

```

1 s = σ subcategory = 'Phones' (products)
2 s2 = π productid, category, subcategory, precionuevo ← price*1.08 (s)
3 s2

```

products.productid	products.category	products.subcategory	precionuevo
'TEC-PH-10002275'	'Technology'	'Phones'	979.7220000000001
'TEC-PH-10002033'	'Technology'	'Phones'	984.3336
'TEC-PH-10001949'	'Technology'	'Phones'	230.5584
'TEC-PH-10004977'	'Technology'	'Phones'	31.8276
'TEC-PH-10000486'	'Technology'	'Phones'	229.02480000000003
'TEC-PH-10004093'	'Technology'	'Phones'	400.8636
'TEC-PH-10003988'	'Technology'	'Phones'	48.6
'TEC-PH-10002447'	'Technology'	'Phones'	16.480800000000002
'TEC-PH-10002726'	'Technology'	'Phones'	230.16960000000003
'TEC-PH-10002844'	'Technology'	'Phones'	110.5488

- e. Disminuir 8% los precios de los productos de la categoría Furniture cuyo precio sea de \$600 a \$900. Aumentar en un 5% los precios de los productos de la categoría Technology y subcategoría Machines.

Para esta consulta use el video '07 Álgebra Relacional | Actualización | Operaciones de mantenimiento de datos', al final use esta consulta:

```

1 -- Vamos a ubicar a los que van a ser actualizados
2 F =  $\sigma$  category='Furniture' and price $\geq$ 600 and price $\leq$ 900 products
3 M =  $\sigma$  category='Technology' and subcategory='Machines' products
4
5 -- Actualizamos cadda uno
6 A =  $\pi$  productid, category,subcategory,newprice $\leftarrow$ price*0.92 (F)
7 B =  $\pi$  productid, category,subcategory,newprice $\leftarrow$ price*1.05 (M)
8
9 -- Unimos y renombramos
10 products1 = A $\cup$ B
11  $\pi$  productid, category, subcategory, price $\leftarrow$ newprice products1

```

De manera que me quedo la siguiente Actualización:

products.productid	products.category	products.subcategory	price
'FUR-CH-10000454'	'Furniture'	'Chairs'	673.3848
'FUR-FU-10003773'	'Furniture'	'Furnishings'	568.2840000000001
'FUR-TA-10004289'	'Furniture'	'Tables'	827.2088
'FUR-FU-10002960'	'Furniture'	'Furnishings'	576.2420000000001
'FUR-CH-10001891'	'Furniture'	'Chairs'	667.7728000000001
'FUR-BO-10001601'	'Furniture'	'Bookcases'	797.088
'FUR-FU-10000221'	'Furniture'	'Furnishings'	595.0376
'FUR-FU-10001095'	'Furniture'	'Furnishings'	595.0008
'FUR-FU-10004270'	'Furniture'	'Furnishings'	664.562
'FUR-FU-10002088'	'Furniture'	'Furnishings'	592.5536000000001

< 1 2 3 >

Ojo que esta solo contiene la tabla de los productos con los precios actualizados como se ve en el video, no contiene toda la lista de productos pero seria tan facil como eliminar las tuplas que se estan seleccionando, e insertar estas nuevas, no lo hice pues no lo hizo el profe :v.