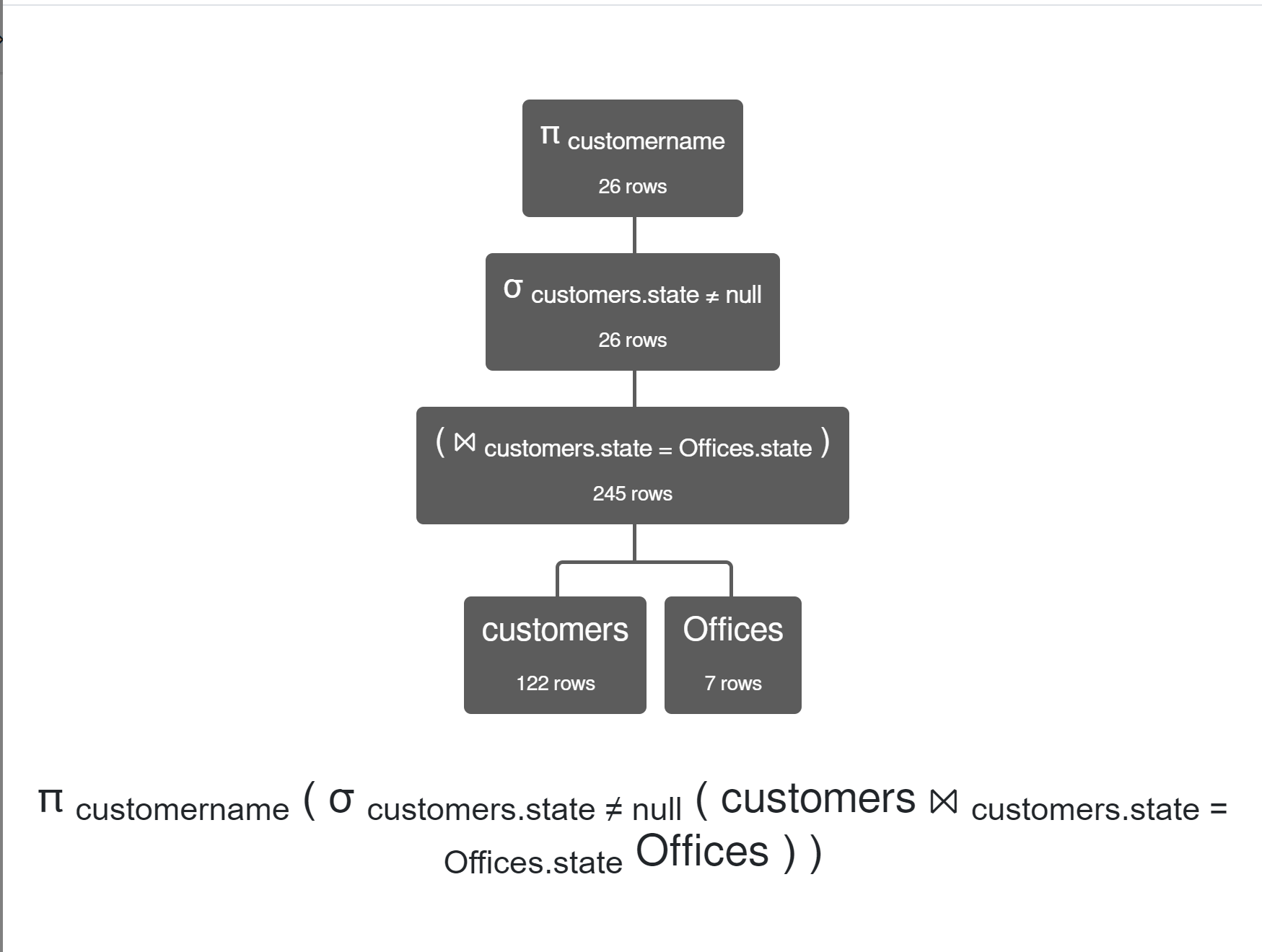
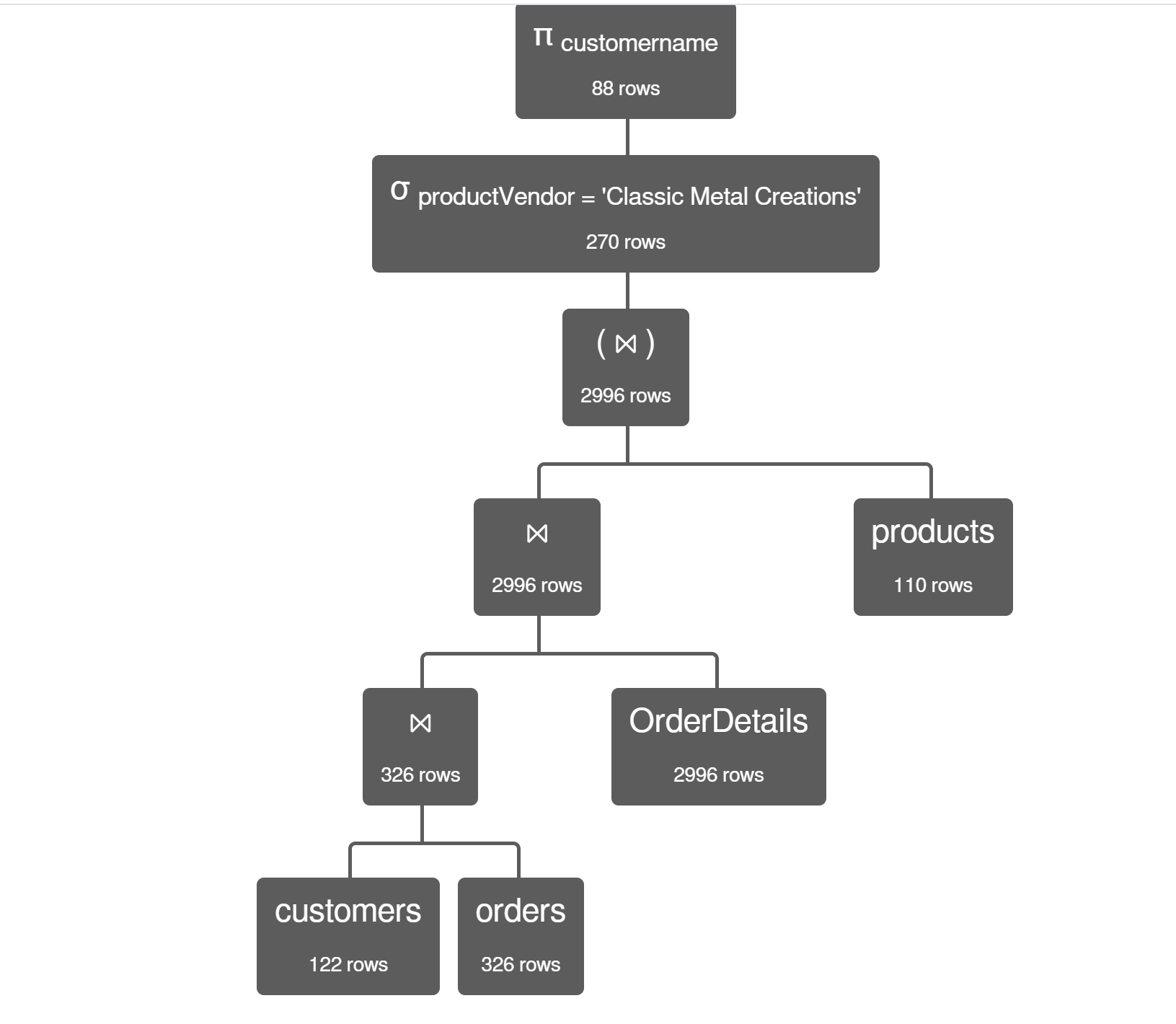
1. List the names of all Customers that are in the same state as one of our Offices. This will return 26 rows if you do not consider that some countries do not have states, so the state portion of their address will be null.

* π customername (σ customers.state ≠ NULL (customers ⨝ customers.state = Offices.state Offices))



1. List the names of all Customers who have ordered Products where the vendor is “Classic Metal Creations”. This returns 88 rows if you remember to use the “distinct” keyword, which isn’t necessary in Relational Algebra of course.

* πcustomername σ productVendor= 'Classic Metal Creations' (customers⨝orders⨝OrderDetails⨝products )



1. List the names of all Customers whose Order was shipped within three days of being ordered. RelaX has an adddate( ) and a subdate( ) function. Both accept a date, and an integer as arguments, and return another date. RelaX also allows you to use Boolean comparison operators on dates. Sadly, to do this in Derby requires that we resort to a Java escape syntax to perform the necessary date arithmetic. While this looks like voodoo, trust me when I say that it works:

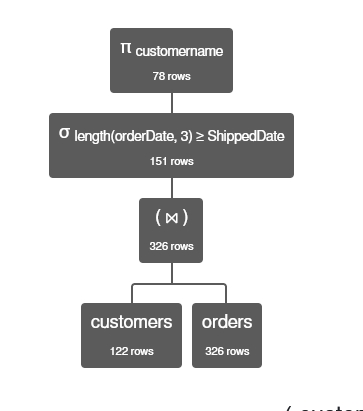
select distinct cust.customerName

from customers cust inner join orders ord using (customerNumber)

where abs({fn TIMESTAMPDIFF (SQL\_TSI\_DAY, ord.ORDERDATE,

ord.SHIPPEDDATE)}) <= 3;

And it returns 78 rows.

* π customername ( σ adddate(orderDate,3) ≥ ShippedDate ( customers ⨝ orders ) )
* 

1. List the names of all Customers, their service rep and the Office that the service rep for that customer works in. Returns 122 rows. Take into account that I said that I want to see all customers.

* πcustomers.customername, Employees.lastName, Employees.firstName, Offices.officeCode, Offices.city (

customers ⟕ (

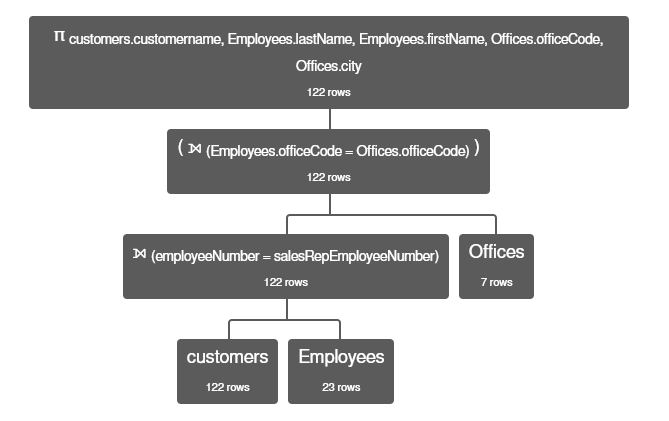
employeeNumber = salesRepEmployeeNumber

) Employees ⟕ (

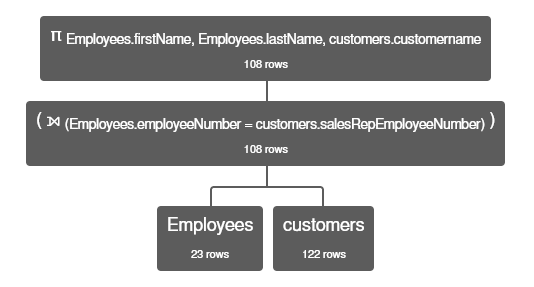
Employees.officeCode = Offices.officeCode

) Offices

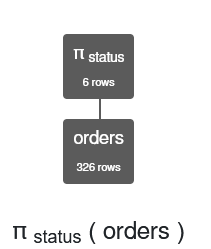
)

* 

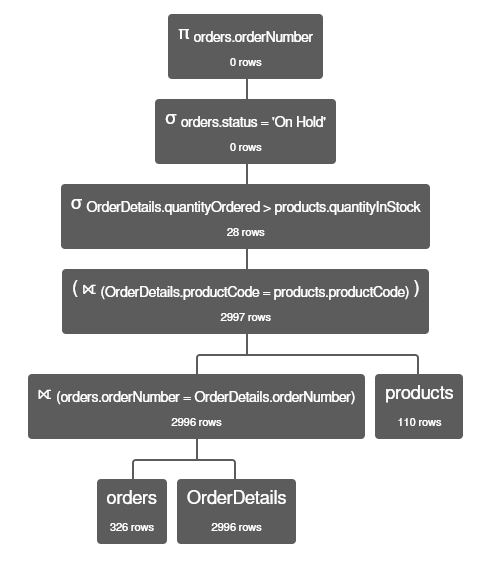
1. List the Employee first and last name, and their Customer’s name even if the Employee is not working with a Customer. Returns 108 rows.

* πEmployees.firstName, Employees.lastName, customers.customername (Employees ⟕ (Employees.employeeNumber = customers.salesRepEmployeeNumber) customers)
* 

1. List all the possible statuses for an order. Returns 6 rows. But this is a little suspect since it really just represents the statuses that we have in use in the data. There might be other statuses possible, but without a table of valid statuses to govern what goes into the status column, it’s impossible to say.

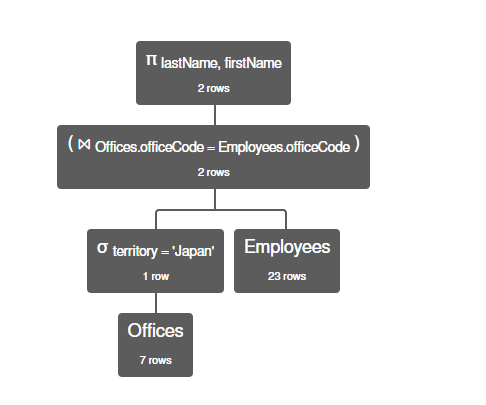
* π status (orders)
* 

1. List the Order Number for all Orders in the ‘On Hold’ status where the quantity of a product ordered is greater than the quantity of that product on hand. This returns 0 rows. Apparently, the ‘On Hold’ requirement removes any rows that we might have gotten otherwise.

* πorders.orderNumber σ orders.status = 'On Hold' σ OrderDetails.quantityOrdered > products.quantityInStock (orders ⟖ (orders.orderNumber = OrderDetails.orderNumber) OrderDetails ⟖ (OrderDetails.productCode = products.productCode) products)
* 

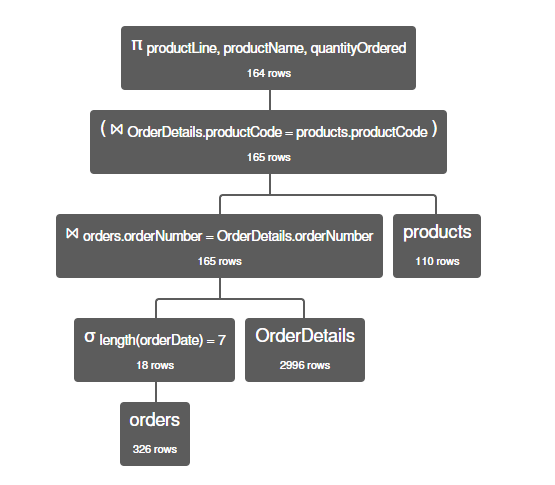
1. List the Employee LastName and FirstName that work in Japan. This returns 2 rows.

π lastName, firstName (σ territory = 'Japan' Offices ⨝ Offices.officeCode = Employees.officeCode Employees)



1. List the productLine, the ProductName and the quantityOrdered for all products ordered during the month of July. You will need a sigma that checks for month(orderDate) = ‘July’. This gets you 164.

π productLine, productName, quantityOrdered (σ month(orderDate) = 7 orders ⨝ orders.orderNumber = OrderDetails.orderNumber OrderDetails ⨝ OrderDetails.productCode = products.productCode products)



1. List the customerName, the paymentDate, and the amount on all payments that exceeded $1000. This returns 272 rows.

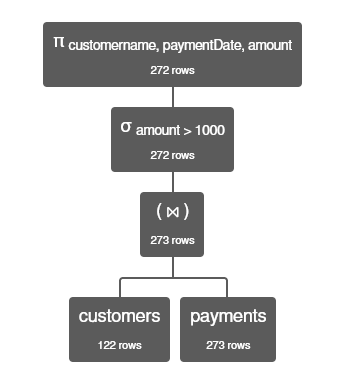
* π customername, paymentDate, amount (

σ amount > 1000 (

customers ⨝ payments

)

)

* 

1. List the productLine for all products ordered by customers from the State of ‘Louisiana’. Sadly, there are no customers from this fine state. Plus, the state in the Customers table is the two-character state code, not the state name. Sorry about that.

π productLine (σ state = 'LA' customers ⨝ customers.customerNumber = orders.customerNumber (orders) ⨝ orders.orderNumber = OrderDetails.orderNumber (OrderDetails) ⨝ OrderDetails.productCode = products.productCode (products) )

