Assignment #4
Instructor: Richard Frank (rfrank@sfu.ca)
TA: Ankit Gupta (aga53@sfu.ca)

Total Marks: 50 (5% of the Individual Assignments)
Due Date: Oct 7, 14:30

For all questions, use the AdventureWorksLT database.

Question 1) [12 marks] Find the names of customers who have ordered all products in category 'Helmets'. First, write the relational algebra using the basic operations (selection, projection, Cartesian product, set-difference, union), then write the SQL code to perform the division. Hint, both are covered in the slides.

Step 1 Project *Product* onto *ProductID*

$\pi_{ProductID}$(Product)

Step 2 Perform Cartesian product with ProductCategory

$\pi_{ProductID}$(Product) × ProductCategory

Step 3 Subtract *Product*

$\pi_{ProductID}$(Product) × ProductCategory – Product

Step 4 Project onto the *ProductID* of *Product*

$\pi_{ProductID}$ ($\pi_{ProductID}$(Product) × ProductCategory – Product)

Step 5 Subtract from Product, projected onto *ProductID*

$\pi_{ProductID}$ (Product) - $\pi_{ProductID}$ ($\pi_{ProductID}$(Product) × ProductCategory – Product)

```
SELECT FirstName, LastName
FROM SalesLT.Customer C
WHERE NOT EXISTS
      (SELECT *
      FROM SalesLT.Product P, SalesLT.ProductCategory Cat
      WHERE P.ProductCategoryID = Cat.ProductCategoryID
      AND Cat.Name = 'Helmets'
      AND NOT EXISTS
            (SELECT *
            FROM SalesLT.SalesOrderHeader H , SalesLT.SalesOrderDetail D
            WHERE H.CustomerID = C.CustomerID
            AND H.SalesOrderID = D.SalesOrderID
            AND D.ProductID = P.ProductID));
```

Question 2) For each relational algebra expression below, answer the following questions:

i) What is the meaning of the query in English? [1 mark each]

ii) Can it be modified to be more efficient? If so, what is the most optimal solution? Rationalize each step in terms of the number of tuples at each step, number of cells, etc. Show work/logic! It is not necessary to go into the database and use exact numbers, just assume/estimate. [3 marks for algebra manipulation, 3 for rationalization]

For the below calculations/estimations, I'm using the following assumptions:

- All relations contain 10 tuples
- All joins of relation *n* and *m* result in the creation of $^{nm}/_{2}$ tuples.
- All conditions/selections on relation *n* halve the number of tuples to $^{n}/_{2}$.

a) $\pi_{Product.Name,Color}[\sigma_{weight<10}(ProductCategory \rhd\lhd Product)]$

i) Display name and colour of all products.

ii) $\pi_{Name,Color}[\sigma_{weight<10}(Product)]$

ProductCategory is not needed. All conditions and features exist in Product. Instead of creating 50 tuples, we need to analyze only 10.

b) $\sigma_{Address1.City='Burnaby' AND\ SubTotal>1000}(Customer \rhd\lhd SalesOrderHeader$
$\rhd\lhd_{SalesOrderHeader.ShipToAddressID=AddressID}\ (\rho_{Address1}Address)$
$\rhd\lhd_{SalesOrderHeader.BillToAddressID=AddressID}\ Address)$

i) All details of Customers and their orders for those customers who had an order, with subtotal more than 1000 shipped to Burnaby.

ii) $(Customer \rhd\lhd \sigma_{SubTotal>1000}SalesOrderHeader$
$\rhd\lhd_{SalesOrderHeader.ShipToAddressID=AddressID}\ \sigma_{Address1.City='Burnaby'}(\rho_{Address1}Address)$
$\rhd\lhd_{SalesOrderHeader.BillToAddressID=AddressID}\ Address)$

Original Query:

Assume the join between Customer and SalesOrderHeader creates originally 50 tuples. Joining further to Address (renamed Address1) will result in 50*10/5=250 tuples. Finally, joining to Address will result in 250*10/2 = 1250 tuples. Selection is applied via *City='Burnaby'*, decreasing our dataset to 625 tuples. Then the selection *SubTotal > 1000* is applied, giving a final result of 312 tuples.

$\sigma_{Address1.City='Burnaby' AND\ SubTotal>1000}(10 \rhd\lhd 10$
$\rhd\lhd_{SalesOrderHeader.ShipToAddressID=AddressID}\ (10) \rhd\lhd_{SalesOrderHeader.BillToAddressID=AddressID}\ 10)$
$= \sigma_{Address1.City='Burnaby' AND\ SubTotal>1000}(1250)$
$= \sigma_{SubTotal>1000}(625)$
=312 tuples

Note: This notation for calculating the complexity does not exist. I am just using it instead of writing things out. The number indicates the number of tuples.

Optimal Query:

$(10 \triangleright\!\triangleleft \sigma_{SubTotal>1000}(10) \triangleright\!\triangleleft \sigma_{Address1.City='Burnaby'}(10) \triangleright\!\triangleleft 10)$

$=(10 \triangleright\!\triangleleft 5 \triangleright\!\triangleleft 5 \triangleright\!\triangleleft 10)$

$=(25 \triangleright\!\triangleleft 5 \triangleright\!\triangleleft 10)$

$= (62.5 \triangleright\!\triangleleft 10)$

$= (312)$

The maximum number of tuples is 312, whereas in the original is it 1250.

c) $\pi_{Firstname}(Customer \triangleright\!\triangleleft CustomerAddress \triangleright\!\triangleleft \sigma_{City='Burnaby'}Address)$

$\cup$

$\pi_{Firstname}(\sigma_{LastName='Smith'}Customer \triangleright\!\triangleleft CustomerAddress \triangleright\!\triangleleft \sigma_{City='Vancouver'}Address)$

i) The first-name of all customers who live in Burnaby and all customers with last name Smith who live in Vancouver.

ii) $\pi_{Firstname}\{(\pi_{Firstname,CustomerID}Customer \triangleright\!\triangleleft \pi_{CustomerID,AddressID}CustomerAddress \triangleright\!\triangleleft \sigma_{City='Burnaby'}Address) \cup (\pi_{Firstname,CustomerID}\sigma_{LastName='Smith'}Customer \triangleright\!\triangleleft \pi_{CustomerID,AddressID}CustomerAddress \triangleright\!\triangleleft \sigma_{City='Vancouver'}Address)\}$

≪ 10,9 ≫ represents 10 tuples and 9 features, total of 90 cells in the relation.

Original Query:

$= \pi_{Firstname}0(Customer \triangleright\!\triangleleft CustomerAddress \triangleright\!\triangleleft \sigma_{Address1.City='Burnaby'}Address) \cup$
$\pi_{Firstname}(\sigma_{LastName='Smith'}Customer \triangleright\!\triangleleft CustomerAddress \triangleright\!\triangleleft \sigma_{City='Vancouver'}Address)$

$= \pi_{Firstname}(\ll 10,16 \gg \triangleright\!\triangleleft \ll 10,5 \gg \triangleright\!\triangleleft \sigma_{Address1.City='Burnaby'} \ll 10,9 \gg) \cup$
$\pi_{Firstname}(\sigma_{LastName='Smith'} \ll 10,16 \gg \triangleright\!\triangleleft \ll 10,5 \gg \triangleright\!\triangleleft \sigma_{City='Vancouver'} \ll 10,9 \gg)$

$= \pi_{Firstname}(\ll 10,16 \gg \triangleright\!\triangleleft \ll 10,5 \gg \triangleright\!\triangleleft \ll 5,9 \gg) \cup \pi_{Firstname}(\ll 5,16 \gg \triangleright\!\triangleleft \ll 10,5 \gg \triangleright\!\triangleleft \ll 5,9 \gg)$

$= \pi_{Firstname}(\ll 125,28 \gg) \cup \pi_{Firstname}(\ll 62,28 \gg)$

$= \pi_{Firstname}(\ll 125,28 \gg + \ll 62,28 \gg)$

$= \pi_{Firstname}(\ll 187,28 \gg)$

$=\ll 187,1 \gg$ that is, 187 tuples, and 1 feature

Until the final stage, the database is also carrying around 16 features from *Customer*, 3 from *CustomerAddress* (since both CustomerID and AddressID are part of the JOIN, they are projected out and do not count), and 9 from Address. So the largest intermediary relation contains 125*28=3500 cells

Optimal Query:

$\pi_{Firstname}\{(\pi_{Firstname,CustomerID}Customer \triangleright\!\triangleleft \pi_{CustomerID,AddressID}CustomerAddress \triangleright\!\triangleleft \pi_{AddressID}\sigma_{City='Burnaby'}Address) \cup (\pi_{Firstname,CustomerID}\sigma_{LastName='Smith'}Customer \triangleright\!\triangleleft \pi_{CustomerID,AddressID}CustomerAddress \triangleright\!\triangleleft \pi_{AddressID}\sigma_{City='Vancouver'}Address)\}$

$= \pi_{Firstname}\{(\pi_{Firstname,CustomerID} \ll 10,16 \gg \triangleright\triangleleft \pi_{CustomerID,AddressID} \ll 10,5 \gg \triangleright\triangleleft \pi_{AddressID}\sigma_{City='Burnaby'} \ll 10,9 \gg) \cup (\pi_{Firstname,CustomerID}\sigma_{LastName='Smith'} \ll 10,16 \gg \triangleright\triangleleft \pi_{CustomerID,AddressID} \ll 10,5 \gg \triangleright\triangleleft \pi_{AddressID}\sigma_{City='Vancouver'} \ll 10,9 \gg)\}$
$= \pi_{Firstname}\{(\ll 10,2 \gg \triangleright\triangleleft \ll 10,2 \gg \triangleright\triangleleft \pi_{AddressID} \ll 5,9 \gg) \cup (\pi_{Firstname,CustomerID} \ll 5,16 \gg \triangleright\triangleleft \pi_{CustomerID,AddressID} \ll 10,5 \gg \triangleright\triangleleft \pi_{AddressID} \ll 5,9 \gg)\}$
$= \pi_{Firstname}\{(\ll 10,2 \gg \triangleright\triangleleft \ll 10,2 \gg \triangleright\triangleleft \ll 5,1 \gg) \cup (\ll 5,2 \gg \triangleright\triangleleft \ll 10,2 \gg \triangleright\triangleleft \ll 5,1 \gg)\}$
$= \pi_{Firstname}\{(\ll 50,3 \gg \triangleright\triangleleft \ll 5,1 \gg) \cup (\ll 25,3 \gg \triangleright\triangleleft \ll 5,1 \gg)\}$
$= \pi_{Firstname}\{(\ll 125,3 \gg) \cup (\ll 62,3 \gg)\}$
$= \pi_{Firstname}\{(\ll 125,3 \gg + \ll 62,3 \gg)\}$
$= \pi_{Firstname}\{(\ll 187,3 \gg)\}$
$= \ll 187,1 \gg$

The maximum amount of information we need to keep in memory is 187 tuples and 3 features.

d) $\rho_{Temp}(Customer \triangleright\triangleleft SalesOrderHeader \triangleright\triangleleft SalesOrderDetail \triangleright\triangleleft Product)$
$\pi_{Firstname,LastName,Product.Name,ListPrice}(Temp)$

i) The first name, last name of each customer, along with the products and the price at which they ordered it.

ii) $\pi_{Firstname,LastName,Product.Name,ListPrice}(\pi_{Firstname,LastName,CustomerID}Customer \triangleright\triangleleft \pi_{SalesOrderID,CustomerID}SalesOrderHeader \triangleright\triangleleft \pi_{SalesOrderID,ProductID}SalesOrderDetail \triangleright\triangleleft \pi_{ProductID,Name,Price}Product)$

Original Query

$\pi_{Firstname,LastName,Product.Name,ListPrice}(Customer \triangleright\triangleleft SalesOrderHeader \triangleright\triangleleft SalesOrderDetail \triangleright\triangleleft Product)$
$= \pi_{Firstname,LastName,Product.Name,ListPrice}(\ll 10,16 \gg \triangleright\triangleleft \ll 10,22 \gg \triangleright\triangleleft \ll 10,9 \gg \triangleright\triangleleft \ll 10,17 \gg)$
$= \pi_{Firstname,LastName,Product.Name,ListPrice}(\ll 50,37 \gg \triangleright\triangleleft \ll 10,9 \gg \triangleright\triangleleft \ll 10,17 \gg)$
$= \pi_{Firstname,LastName,Product.Name,ListPrice}(\ll 250,46 \gg \triangleright\triangleleft \ll 10,17 \gg)$
$= \pi_{Firstname,LastName,Product.Name,ListPrice}(\ll 1250,62 \gg)$ ← that is 77500 cells!
$= \pi_{Firstname,LastName,Product.Name,ListPrice}(\ll 1250,4 \gg)$

Optimal Query

$= \pi_{Firstname,LastName,Product.Name,ListPrice}(\pi_{Firstname,LastName,CustomerID} \ll 10,16 \gg \triangleright\triangleleft \pi_{SalesOrderID,CustomerID} \ll 10,22 \gg \triangleright\triangleleft \pi_{SalesOrderID,ProductID} \ll 10,9 \gg \triangleright\triangleleft \pi_{ProductID,Name,Price} \ll 10,17 \gg)$
$= \pi_{Firstname,LastName,Product.Name,ListPrice}(\ll 10,3 \gg \triangleright\triangleleft \ll 10,2 \gg \triangleright\triangleleft \ll 10,2 \gg \triangleright\triangleleft \ll 10,3 \gg)$
$= \pi_{Firstname,LastName,Product.Name,ListPrice}(\ll 50,4 \gg \triangleright\triangleleft \ll 10,2 \gg \triangleright\triangleleft \ll 10,3 \gg)$
$= \pi_{Firstname,LastName,Product.Name,ListPrice}(\ll 250,5 \gg \triangleright\triangleleft \ll 10,3 \gg)$
$= \pi_{Firstname,LastName,Product.Name,ListPrice}(\ll 1250,7 \gg)$
$= (\ll 1250,4 \gg)$

Question 3) Write the *optimal* relational algebra statement for the following. [5 marks each]

a) Display the product descriptions for Mountain Tires (a product).

$=\pi_{Description}(\pi_{ProductModelID}\sigma_{Name='Mountain\ Tires'}Product \rhd\lhd \pi_{ProductModelID,ProductDescriptionID}ProductModelProductDescription \rhd\lhd \pi_{ProductDescriptionID,Description}ProductDescription$

b) Assume the ProductCategory has 3 levels. Write the query to display the product names at all levels within the hierarchy.

For example, assume ProductCategory looks like:

ProductCategoryID	ParentProductCategoryID	Name
1		Car
2	1	Sports
3	2	Mustang
4	2	Ferrari

Required result:

Car	Sports	Mustang
Car	Sports	Ferrari

$=\pi_{P1.Name,P2.Name,P3.Name}(\rho_{P1}ProductCategory \rhd\lhd_{P1.ProductCategoryID = P2.ParentProductCategoryID} \rho_{P2}ProductCategory \rhd\lhd_{P2.ProductCategoryID = P3.ParentProductCategoryID} \rho_{P3}ProductCategory)$