CMPT 454 Assignment 4: Query Optimization 2

Question 1

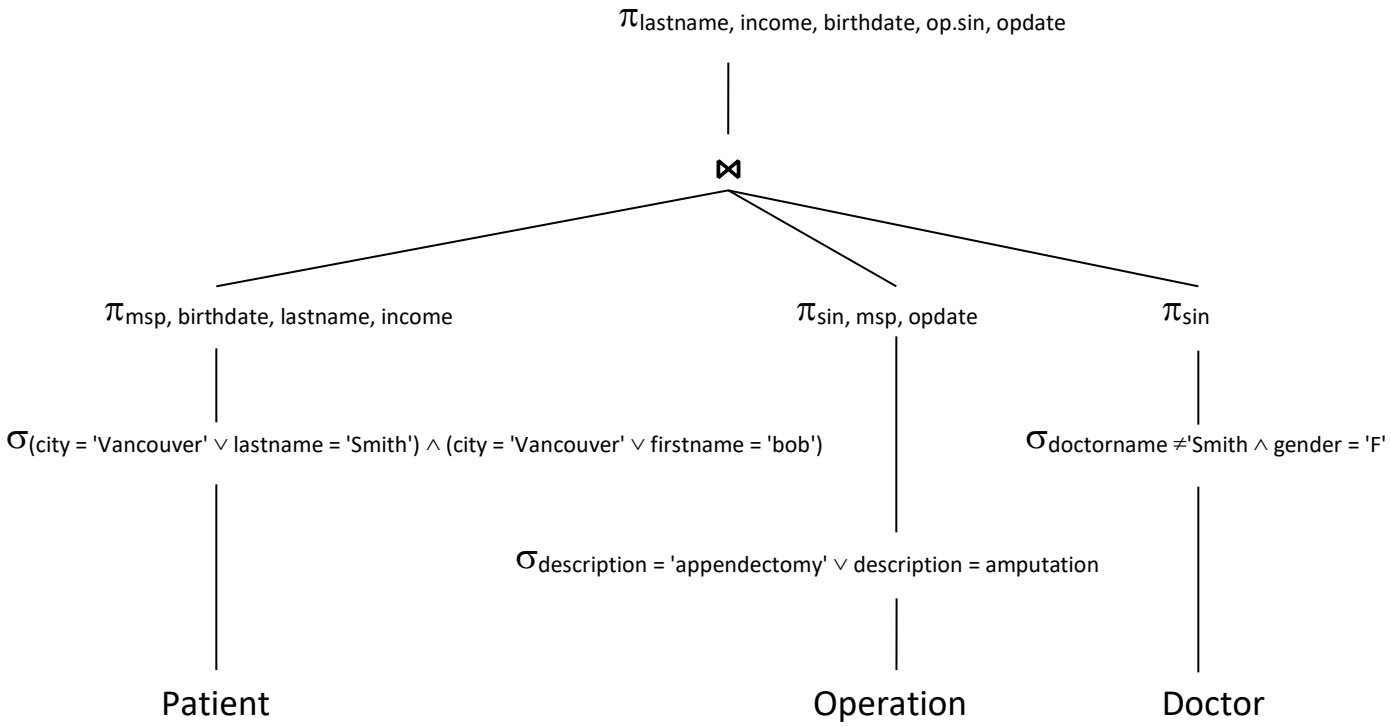
Answer the questions that follow about the SQL query shown below. [10]

```
SELECT p.lastname, p.income, p.birthdate, op.sin, op.opdate
FROM Patient p, Operation op, Doctor doc
WHERE (p.city = 'Vancouver' OR p.lastname = 'Smith')
AND (p.city = 'Vancouver' OR p.firstname = 'bob')
AND p.msp = op.msp AND op.sin = doc.sin
AND (op.description = 'appendectomy' OR op.description = 'amputation')
AND doc.doctorname <> 'Smith' AND doc.gender = 'F'
```

Sketch a relational algebra expression tree that represents an equivalent logical plan to your answer to the SQL query shown below. Your query should be derived by using the following heuristics:

- Replace Cartesian products with joins
- Push selections as far down the tree as possible
- Push projections as far down the tree as possible, except after selections, add attributes to the projections where necessary to achieve this
- Group join operators

selections before join: **3** (-1 for each error)
projections before join: **3** (-1 for each error)
join: position (above selections and projections): **1**, group (no order specified): **1**
final projection: **2**



Question 2

You are to estimate the cost of the relational algebra query shown below.

$\pi_{cname,cemail,vname,vcity,ceo,pname,ptype,description,sdate}($
$\pi_{vname,vcity,ceo}(\sigma_{vcountry='Canada' \wedge employees > 500}(Vendor)) \bowtie$
$\pi_{pid,vname,pname,description,ptype}(Product) \bowtie$
$\pi_{cid,cname,cemail}(\sigma_{ccity = 'Vancouver' \vee ccity = 'Edmonton'}(Customer)) \bowtie Sales)$

Indexes

- Secondary B+ tree on {*vcountry*, *vcity*}, of height 3 in Vendor
- Primary B+ tree on {*ccity*, *cstreet, cnumber*}, of height 4 in Customer
- Linear hash index on cid in Sales, no bucket consists of more than one block

Relation Statistics

	Customer	Product	Vendor	Sales
T(R)	200,000	100,000	50,000	10,000,000
B(R)	25,000	10,000	5,000	500,000
V(R, cid)	200,000			200,000
V(C, ccity)	100			
V(R, pid)		100,000		100,000
V(R, vname)		50,000	50,000	
V(V, vcountry)			5	
V(V, employees)			5,000	

- Primary B+ tree index of height 4 on pid in Sales
- For B+ tree indexes assume that the root node of the index is retained in main memory
- Records of all tables, including intermediate relations. should not span more than one block

Attribute, Block and Main Memory Sizes

- cid, pid, employees – 4 bytes
- ptype, sdate – 8 bytes
- cname, ccity, pname, vname, vcity, ceo, vcountry – 16 bytes
- cemail – 32 bytes
- description – 64 bytes
- Block (and main memory frame) – 4,096 bytes
- There are 600 main memory frames available for the query

a) Estimate the cost for performing the selection and projection on Vendor, do not include the cost for writing out the result (if any): $\pi_{vname,vcity,ceo}(\sigma_{vcountry='Canada' \wedge employees > 500}(Vendor))$. The minimum and maximum values for the *employees* attribute are 0 and 5,000. [2] No index on employees, the {vcountry, vcity} index is secondary and returns $1/5^{th}$ of the records so scan the file. selection cost = **5,000**, projection can be applied to the results of the selection, cost = **0**. **1** mark for **each** (or just **2** marks for stating cost is 5,000)

b) State the size of the Vendor selection and projection result in records and bytes. [2] Records: 50,000 * 1/5 * 9/10 = **9,000** records (**1** mark). Each record takes up 48 (16*3) bytes, bf = 85, so size = 9,000 / 85 = **106** *blocks* ± 6 or between **434,176** and **432,000** *bytes* (**1** mark)

c) Estimate the cost for performing the selection and projection on Customer, do not include the cost for writing out the result (if any): $\pi_{cid,cname,cemail}(\sigma_{ccity = 'Vancouver' \vee ccity = 'Edmonton'}(Customer))$. Histograms kept by the DB indicate that 25% of customers live in Vancouver and 15% live in Edmonton. [2] Use the primary address index. Cost = 3 (tree height – 1) * 2 (two index lookups) + the cost to read 40% of the file = **10,006**. **2** marks for ± 3, 1 mark for ± 6

d) State the size of the Customer selection and projection result in records and bytes. [2] Returns 40% of the records (from the question) so **80,000** records (**1** mark). Each record takes up 52 (4 + 16 + 32) bytes, bf = 78, so size = 80,000 / 78 = **1,026** blocks ± 20 or between **4,202,496** and **4,160,000** *bytes* (**1** mark)

e) State the size of the Product projection result in records and bytes. [2] Returns all the records so **100,000** records (**1** mark). Each record takes up 108 (4 + 16 +16 + 8 + 64) bytes, bf = 37, so size = 100,000 / 37 = **2,703** blocks ± 75 or between **11,071,488** and **10,800,000** *bytes* (**1** mark)

f) Determine the most efficient join order using the dynamic programming approach described in class*. The metric you use to determine join order should be the least number of *records*. You should not consider the actual cost of performing the join in your calculation, or the cost of producing input for the join. You should only consider left-deep join trees. [10]

*You can do this informally, and do not have to submit the tables shown in the class example (and Exercise 9). Don't forget that the size in records of some of the tables is affected by previous operations.

Your answer should consist of the following:

- The order in which relations are to be joined. *e.g. (((A ⋈ B) ⋈ C) ⋈ D)*
- The total cost estimate (the sum of the intermediate relation sizes) for your chosen join order, broken down by join. *e.g. Total cost: AB cost + ABC cost = total cost*
- The size in *records* of each of the three joined relations. *e.g. AB, ABC and ABCD size*
- The size in *blocks* of each of the three joined relations. *e.g. AB, ABC and ABCD size*

This is where it gets tricky as you need to refer to the sizes of the results of the previous operations in b, d and e which may not be correct. ***As much as possible do not deduct (more) marks for incorrect answers in the previous questions***. This may not be possible if the previous answers are incorrect by a large margin.

Working

Relation sizes: V = 9,000; C = 80,000; P = 100,000; S = 10,000,000.
Join attributes: CS on cid, PS on pid, PV on vname. Rule out CP, CV and VS (Cartesian products)

2 Relation Joins
CS = 10,000,000 * 80,000 / 200,000 = 4,000,000
PV = 9,000 * 100,000 / 50,000 = 18,000
PS = 100,000 * 10,000,000 / 100,000 = 10,000,000

3 Relation Joins
(PV)S = 18,000 * 10,000,000 / 100,000 = 1,800,000 – note this size + PV size (18,000) is less than the size of either CS or PS. (PV)C is a Cartesian product so can also be ruled out.

Answer
Join Order: (((*Product* ⋈ *Vendor*) ⋈ *Sales*) ⋈ *Customer*) – note that *Vendor* ⋈ *Product* for the first join is fine. 2 marks – 1 for each error.
Cost estimate: PV = **18,000** + PVS = **1,800,000** = 1,818,000. **2** marks, **1** for each component.
Record size: PV = **18,000**; PVS = **1,800,000**; PVSC = (1,800,000 * 80,000 / 200,000) = **720,000**. 3 marks, **½** for each of PV and PVS, **2** for PVSC.

1. Blocks of PV: record size (48 + 92 only one *vname*) = 140. So, 29 per block, so **621**.
2. Blocks of PVS – this is a bit trickier because we don't know the attributes in Sales. But we do know its blocking factor is 20. Assume each record is approx. size 204 (4,096 / 20). That gives a joined record size of (204 + 140 – 4) = 340. So, 12 per block, so **150,000**.
3. Blocks of PVSC – Joined record size of (340 + 52 – 4) = 388. So, 10 per block, so **72,000**.

1 mark for each relation. Answers within 10% are acceptable.

g) For the first of the three joins specified in part (f) you are to calculate the cost in reads and writes of performing the join. [10 divided between g, h and i] ***For each calculation give students a mark if the basis of their calculation is shown and it appears correct.***

Note that:

- Your cost estimate should only include the cost of performing the join and should not include the cost of previous operations or the cost of writing out the result.
- Your cost estimate should not be the same as the result of part (f) – the value computed in (f) is used only to determine the join order.
- You should only consider three join algorithms: *block nested loop join*, *hash join* and *index nested loop join* ***and must state the cost for each of these algorithms***. If an index nested loop join is not possible (because there is no relevant index) state this.

First join: *Product* ⋈ *Vendor*, block sizes: P = 2,703; V = 106; M = 600
Block nested loop join – can fit V in main memory, cost = 106 + 2,703 = **2.809** – **1** mark, students should get a mark if the cost is equal to the combined sizes of the relations as long as one is less than 600 (M)
Hash join – can fit V in main memory, cost = 106 + 2,703 = **2.809** – **1** mark, comment as above
Index nested loop join – not possible as P and V are not base tables and have no indexes – **1** mark

h) For the second of the three joins specified in part (f) you are to calculate the cost in reads and writes of performing the join. Read the description in part (g) for more information.

Second join: PV ⋈ *Sales*, block sizes: PV = 621; S = 500,000; M = 600
Block nested loop join – Can't fit PV in main memory (quite) so have to scan S twice = 621 + 500,000 *2 = **1,000,621** – **1** mark, arguably don't need initial read of PV, so 1,000,000 is OK
Hash join – 3(PV + S) = **1,501,863** – **1** mark, 1,501,242 is OK

Index nested loop join – use the primary index on pid in Sales. Each look-up requires 3 reads to get to the leaf and each PV record matches to 100 records on 6 blocks. Cost is 621 + 18,000 * 9 = **162,621** – 2 marks, answers for full marks: in range of 140,000 to 181,000

i) For the third of the three joins specified in part (f) you are to calculate the cost in reads and writes of performing the join. Read the description in part (g) for more information.

Third join: PVS ⋈ *Customer*, block sizes: PVS = 150,000; C = 1,026; M = 600
Block nested loop join – Can't fit C in main memory (quite) so have to scan PVS twice = 1,026 + 300,000 *2 = **301,026** – 1 mark, arguably don't need initial read of PV, so 300,000 is OK
Hash join – 3(PV + S) = **453,078** – 1 mark
Index nested loop join – not possible as P and V are not base tables and have no indexes – 1 mark

j) Estimate the total cost of performing the query. This is mostly adding up the costs from previous parts of the question, but you should note whether or not operations can be pipelined. If not, you should include the cost of materializing the results of intermediate operations. [6] ***I'm more concerned with the pipeline / materialization costs, so give marks as shown below if the decision to pipeline / materialize appears correct.***

Process	Cost	Mark
Perform Vendor selection and projection, output becomes first "clump" for input for the block nested loop join, i.e. pipeline into join.	5,000	
Read in Product records to join to Vendor, perform projection as they are read in.	10,000	
Join PV. Note that V is in main memory. P is read in to perform projections (requires 2 frames) and joined.	0	**1** for pipelining V **1** for pipelining P
PV does not quite fit in main memory, and 1/5th of M is taken up with P. For simplicity materialize all of P. In fact, most of it can be retained in memory and pipelined into next join.	621	**1** for materializing PV – note that a correct answer is that only some of it needs to be materialized.
Join PV and S – read in PV and use S index.	162,621	
Since C is to be outer relation of next join PVS must be materialized	150,000	**1** for materializing PVS
Join PVS and C	301,026	
Final projection, perform with PVS C join	0	**1**
	629,268	**1** ± 10%

Marks Summary

Question 1

- selections before join: **3** (-1 for each error)
- projections before join: **3** (-1 for each error)
- join: position (above selections and projections): **1**, group (no order specified): **1**
- final projection: **2**

Question 2

a) selection cost = **5,000** = **0**. **1** mark for each (or just **2** marks for stating cost is 5,000)

b) records = **9,000** (**1** mark). *blocks* = **106** ± 6 or between **434,176** and **432,000** *bytes*. (**1** mark)

c) cost = **10,006**. **2** marks for ± 3, **1** mark for ± 6

d) records = **80,000** records (**1** mark). Each record takes up 52 (4 + 16 + 32) bytes, bf = 78, so size = 80,000 / 78 = **1,026** blocks ± 20 or between **4,202,496** and **4,160,000** *bytes.* (**1** mark)

e) records = **100,000** (**1** mark). *blocks* = **2,703** *blocks* ± 75 or **11,071,488** and **10,800,000** *bytes.* (**1** mark)

f) ***As much as possible do not deduct (more) marks for incorrect answers in the previous questions***. This may not be possible if the previous answers are incorrect by a large margin.

Join Order: (((*Product* ⋈ *Vendor*) ⋈ *Sales*) ⋈ *Customer*) – note that *Vendor* ⋈ *Product* is OK for tr the first. **2** marks – **1** for each error.
Cost estimate: PV = **18,000** + PVS = **1,800,000** = 1,818,000. **2** marks, **1** for each component.
Record size: PV = **18,000**; PVS = **1,800,000**; PVSC = **720,000**. **3** marks, **½** for each of PV and PVS, **2** for PVSC.
Block Size

1. PV: **621**.
2. PVS: **150,000**.
3. PVSC: **72,000**.

1 mark for each relation. ***Answers within 10% are acceptable***.

g) [10 divided between g, h and i] ***For each calculation give students a mark if the basis of their calculation is shown and it appears correct.***

First join: *Product* ⋈ *Vendor*, block sizes: P = 2,703; V = 106; M = 600
Block nested loop join = **2.809** – **1** mark, students should get a mark if the cost is equal to the combined sizes of the relations if one is less than 600 (M)
Hash join = **2.809** – **1** mark, comment as above
Index nested loop join – not possible as P and V are not base tables and have no indexes – **1** mark

h) For the second of the three joins specified in part (f) you are to calculate the cost in reads and writes of performing the join. Read the description in part (g) for more information.

Second join: PV ⋈ *Sales*, block sizes: PV = 621; S = 500,000; M = 600
Block nested loop join = **1,000,621** – **1** mark, 1,000,000 is OK
Hash join = **1,501,863** – **1** mark, 1,501,242 is OK
Index nested loop join = **162,621** – **2** marks, answers for full marks: in range of 140,000 to 181,000

i) For the third of the three joins specified in part (f) you are to calculate the cost in reads and writes of performing the join. Read the description in part (g) for more information.

Third join: PVS ⋈ Customer, block sizes: PVS = 150,000; C = 1,026; M = 600
Block nested loop join = **301,026** – **1** mark, 300,000 is OK
Hash join – 3(PV + S) = **453,078** – **1** mark
Index nested loop join – not possible as P and V are not base tables and have no indexes – **1** mark

j) ***I'm more concerned with the pipeline / materialization costs, so give marks as shown below if the decision to pipeline / materialize appears correct.***

Process	Cost	Mark
Perform Vendor selection and projection	5,000	
Read in and project Product records to join to V	10,000	
Join PV. Note that V is in main memory. P is read in to perform projections and joined.	0	**1** for pipelining V **1** for pipelining P
PV does not quite fit in main memory, and 1/5th of M is taken up with P. For simplicity materialize all of P. In fact, most of it can be retained in memory and pipelined into next join.	621	**1** for materializing PV – note that a correct answer is that only some of it needs to be materialized.
Join PV and S – read in PV and use S index.	162,621	
C is outer relation so materialize PVS	150,000	**1** for materializing PVS
Join PVS and C	301,026	
Final projection, perform with PVS C join	0	**1**
	629,268	**1** ± 10%