

I will not use notes, other exams, or any source other than my own brain on this exam: _____

To students: To request a regrade or to otherwise appeal your score, submit a one paragraph explanation of why you think the grade was “in error” and submit your explanation/request together with your exam to Doug after class.

To graders: No question answer should receive less than 0, regardless of what rubric otherwise suggests. In general, be forgiving of minor syntax errors, but ask if there is any question

I will not use notes, other exams, or any source other than my own brain on this exam: _____

1. (2 pts) In the MoneyThink Mobile app “spend or save” challenge, choose one of the following groups of relations as most reasonable for inclusion in the database, with potentially still other relations, to support this app.

2 pts

(a) Decision(PersonId, ItemId, ItemImage, DateOfDecision, Buy?), Friend(Person1Id, Person2Id)

(b) Decision(PersonId, StockId, DateOfDecision, StockPrice, Buy?), Broker(BrokerId, ClientId)

1 pt

(c) Decision(PersonId, ItemId, ItemImage, DateOfDecision, Buy?), ~~Friend(Person1Id, Person2Id)~~
i.e., no social information recorded

(d) Decision(PersonId, StockId, Buy?), Broker(BrokerId, ClientId)

(e) None of the above are reasonable choices. Explain: _____

1 minute

2. Consider the following relational schema (underlined variables in each schema make up its key)

Supplier (sid: integer, sname: string, address: string, city: string)

Catalog (sid: integer, pid: integer, cost: real)

Part (pid: integer, pname: string, color: string)

a) (2 pts) Give a relational algebra expression that implements the query specified as

Find the pids and pnames of any green part.

1 pt for a proper selection (sigma), including the base table (Part) and condition

Project(pid, pname) (select(color = 'green') Part)

1 point for a proper projection (pi)

b) (3 pts) Give a relational algebra expression that implements the query specified as

Find the snames of Suppliers that supply any part that costs less than \$5 (or just '5'), and list the pid along with the supplier's name.

Project(sname, pid) (select(cost < 5) (Supplier njoin Catalog))

1 pt for a proper selection (sigma), including condition

njoin means natural join

1 point for a proper join, including both tables

1 point for proper projection

$$\pi_{\text{sname, pid}}(\sigma_{\text{cost} < 5}(\text{Supplier} \bowtie \text{Catalog}))$$

Answers can also write a natural join as a theta join (e.g., with the explicit condition of equality between all same named attributes (but to be precisely correct, this would require renaming operators, but be lenient if such an answer forgets renaming)

4 minutes

3. (3 pts) Consider the following relational schema (bold underlined variables in each schema make up a key)

Supplier (**sid: integer**, sname: string, address: string, city: string)

Catalog (**sid: integer, pid: integer**, cost: real)

Part (**pid: integer**, pname: string, color: string)

Give a relational algebra expression that implements the query specified as

*Find the **pairs of Sids** of suppliers that are in the **same city**. List each qualifying pair of Sids only once, regardless of order (i.e., if 123, 456) is listed, (456, 123) will not be.*

Possible answers

**Project(S1.sid, S2.sid) (Rename(Supplier = S1)
join(S1.sid < S2.sid and S1.city = S2.city)
Rename(Supplier = S2))**

Notice that a selection is not needed for this question; notice this is not a natural join, but a theta join

**Project(S1.sid, S2.sid) (Select(S1.sid < S2.sid and S1.city = S2.city)
(Rename(Supplier = S1)
X
Rename(Supplier = S2)
)
)**

Cross product

Or perhaps an answer that puts one condition in the join condition and one condition in an outer selection condition

The notation students will use is below, but they might also use assignment notation for renaming, such as S1 = Supplier and S2 = Supplier

-1 if there is a missing rename operator (should be at least two) but be flexible on syntax for the renaming – it will typically be through use of rho , but might also be through use of an “assignment” op

-1 for each of missing/botched selection, join, projection, etc

$\pi_{S1.sid, S2.sid} (\rho_{S1(sid,...)}(Supplier) \bowtie_{S1.sid < S2.sid \text{ AND } S1.city = S2.city} \rho_{S2(sid,...)}(Supplier))$

4. (3 pts) Consider the following relational schema (underlined variables in each schema make up its primary key)

Supplier (sid: integer, sname: string, address: string, city: string)

Catalog (sid: integer, pid: integer, cost: real)

Part (pid: integer, pname: string, color: string)

Give a **left-deep** expression that represents

Find the sids of Suppliers that supply any green part and are in the city of Nashville.

Project(sid) (select(color=Green and city=Nashville) (Supplier njoin Catalog njoin Part))

This is one possible relational algebra expression on which the tree can be based, and if they gave a Correct RA expression, ensure 1 point partial credit regardless of how incorrect the tree is. Other candidate expressions might have pushed a selection (or both selections – still partial credit, but something we will discuss in class)

Project(sid) (select(color=Green) (select(city=Nashville) (Supplier)) njoin Catalog njoin Part))

Project(sid) (select(city=Nashville) (Supplier njoin Catalog njoin (select(color=Green) Part)))

Any correct expression tree in which the right child of each join is a base table receives 3 points; and correct tree that is not left deep receives 1 point

5 minutes

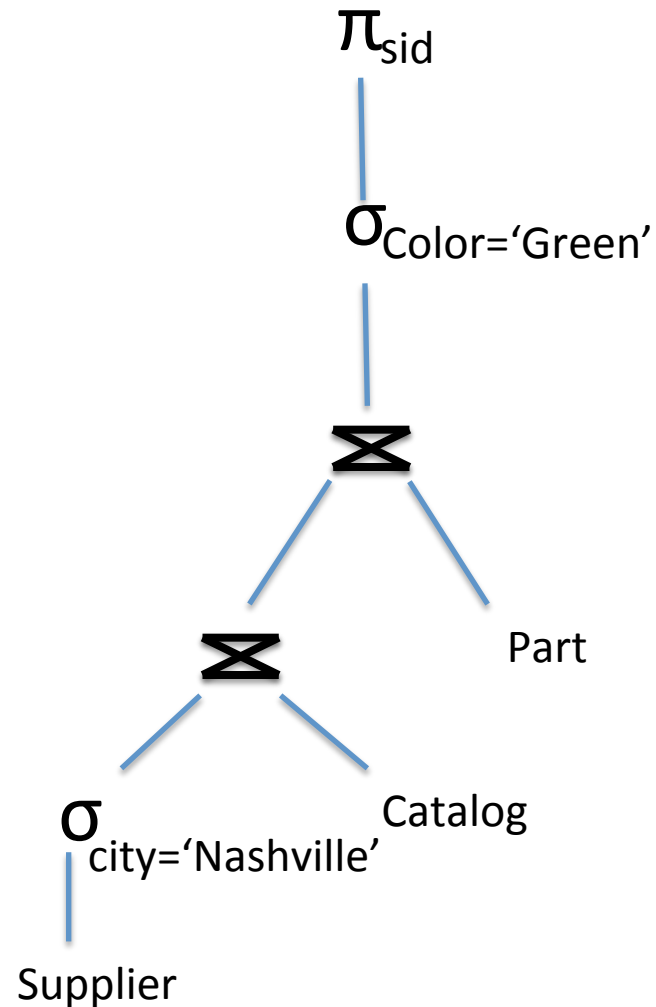
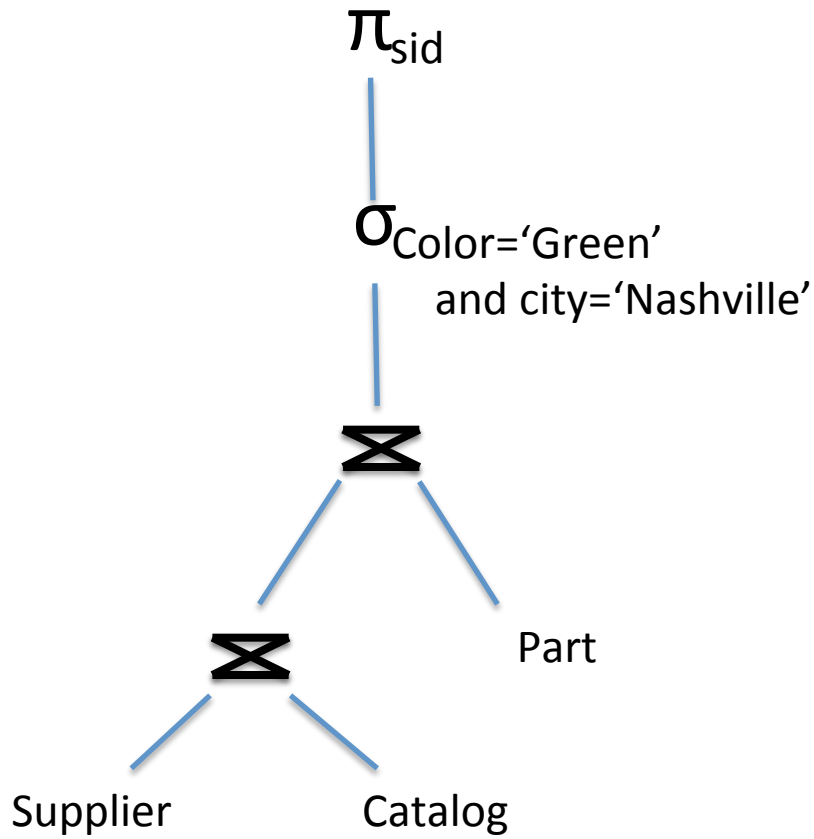
Give a **left-deep** expression that represents

Find the sids of Suppliers that supply any green part and are in the city of Nashville.

Possible answers

Project(sid) (select(color=Green and city=Nashville) (Supplier njoin Catalog njoin Part))

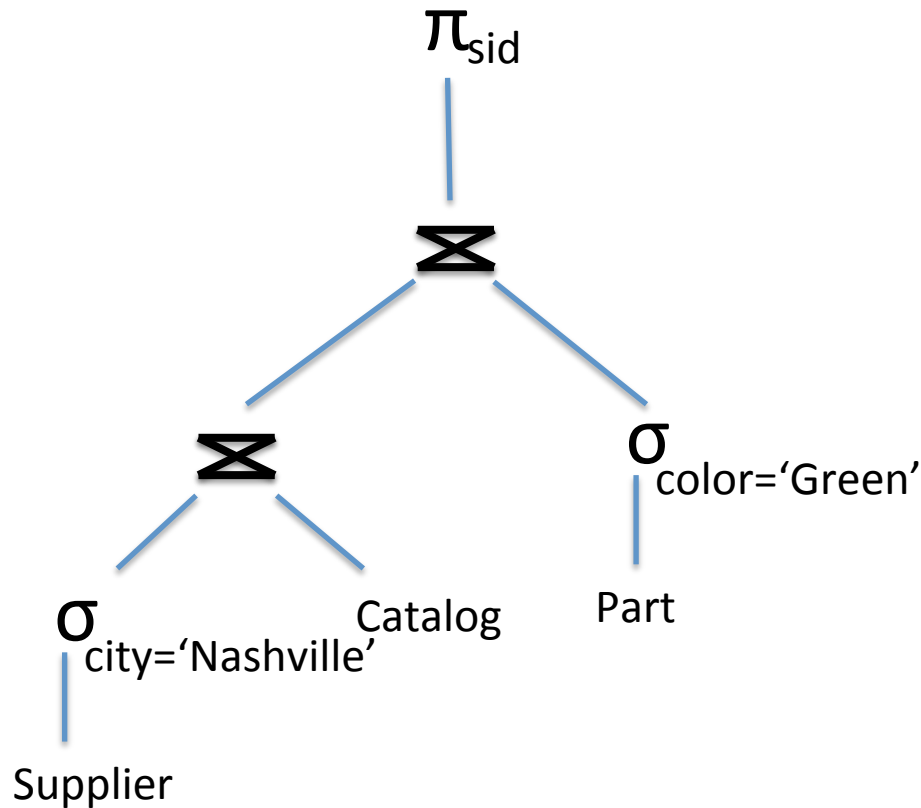
Project(sid) (select(color=Green) (select(city=Nashville) (Supplier)) njoin Catalog njoin Part))



Give a **left-deep** expression that represents

Find the sids of Suppliers that supply any green part and are in the city of Nashville.

WRONG answer



This is a valid expression tree that correctly implements the query specification, but it is **NOT** left deep (because there exists a child of a join that is **NOT** a base table)

For other examples of left deep trees, though not for the same query specification, see the key for Quiz 2 under week 3 of <https://my.vanderbilt.edu/cs265/schedule/>

For SQL questions, you may not use the JOIN keywords

5. Assume the following relational schemas (underlined variables in each schema make up its key).

Customer (SSN: integer, *name*: string, *address*: string, *city*: string)

Account (SSN: integer, AcctNo: integer)

Transaction (AcctNo: integer, ProductId: integer, *date*: string, *quantity*: integer)
/* *quantity* is the number of ProductId purchased on given transaction */
/* You can think of “transaction” as synonymous with a purchase */

Product (ProductId: integer, *ProductName*: string, *cost*: real)

Write SQL queries for the following. You may NOT use nested queries and you may NOT use any JOIN keyword constructs.

a) **(1 pt)** Write an SQL query that lists the names of all customers living in Nashville (i.e., city = ‘Nashville’).

```
SELECT C.Name FROM Customer C WHERE C.city = 'Nashville'
```

All or nothing

b) **(2 pt)** Write an SQL query that lists the names and account numbers of all customers (regardless of city).

```
SELECT C.Name, A.AcctNo  
FROM Customer C, Account A  
WHERE C.SSN = A.SSN
```

All or nothing

4 minutes

6. Using the same relational schemas, write SQL queries for the following

Customer (SSN: integer, name: string, address: string, city: string)

Account (SSN: integer, AccntNo: integer)

Transaction (AccntNo: integer, ProductId: integer, date: string, quantity: integer)

Product (ProductId: integer, ProductName: string, cost: real)

a) **(1 pts)** List the total number (sum) of all quantities for the product identified by “ProductId = X”, regardless of AccntNo, that were purchased on “date = Y”.

```
SELECT SUM(T.quantity)
FROM Transactions T
WHERE T.ProductId = X AND T.date = Y
```

All or nothing

b) **(2 pts)** For EACH transaction date and ProductId, list the date, ProductId, and the total number (sum) of all the product’s quantities purchased on that date.

```
SELECT T.ProductId, T.date, SUM(T.quantity)
FROM Transactions T
GROUP BY T.ProductId, T.date
```

-1 if didn’t Group By both ProductID and date
-2 if no Group By at all
-1 if didnt use aggregate operator
-1 for other errors

There were some who were confused between SUM and COUNT on this question, and questions 7 and 8

6 minutes

7. (5 points) A health facility wants to measure the **AVERAGE (AVG) spread in the weight of clients between the ages of 45 and 60**, inclusive, **who have at least 100 weight entries in the facility's DB**. The spread of a client's weight is the maximum weight on record in the facility's DB minus the minimum weight on record in the DB (regardless of age at the time the weight entries were made). The DB contains two tables representing these two relations (among others).

Client (**cid**, name, age, address, phone, ...) DailyRecord(**cid, date**, weight, ...)

The attributes making up the key of each table (relation) are bold-face and underlined. In the actual table definitions, the attribute of **age is declared as NOT NULL** (i.e., it will never be NULL), and **weight can be NULL**.

Complete the following skeletal query to compute the AVERAGE (AVG) spread in the weight of clients between the ages of 45 and 60, inclusive, with at least 100 weight entries, by filling in the blanks.

10 minutes

```
SELECT AVG(Temp.Flux)
FROM (SELECT MAX(D.weight) – MIN(D.weight) AS Flux
      FROM Client C, DailyRecord D
      WHERE C.id = D.id AND D.weight IS NOT NULL AND C.age >= 45 AND C.age <= 60
      GROUP BY C.id
      HAVING COUNT(*) >= 100 ) AS Temp
```

7. (5 points) A health facility wants to measure the **AVERAGE (AVG) spread in the weight of clients between the ages of 45 and 60**, inclusive, **who have at least 100 weight entries in the facility's DB**. The spread of a client's weight is the maximum weight on record in the facility's DB minus the minimum weight on record in the DB (regardless of age at the time the weight entries were made). The DB contains two tables representing these two relations (among others).

Client (**cid**, name, age, address, phone, ...) DailyRecord(**cid, date**, weight, ...)

The attributes making up the key of each table (relation) are bold-face and underlined. In the actual table definitions, the attribute of **age is declared as NOT NULL** (i.e., it will never be NULL), and **weight can be NULL**.

Complete the following skeletal query to compute the AVERAGE (AVG) spread in the weight of clients between the ages of 45 and 60, inclusive, with at least 100 weight entries, by filling in the blanks.

SELECT AVG(Temp.Flux) **1 pt** 10 minutes

FROM (SELECT MAX(D.weight) – MIN(D.weight) AS Flux **1 pt**

FROM Client C, DailyRecord D

1 pt for join condition **Ok if weight NOT NULL check is missing, but not next time**

WHERE C.id = D.id AND D.weight IS NOT NULL AND C.age >= 45 AND C.age <= 60

GROUP BY C.id **2 pts for rest, use discretion**

HAVING COUNT(*) >= 100) AS Temp

7. (5 points) A health facility wants to measure the **AVERAGE (AVG) spread in the weight of clients between the ages of 45 and 60**, inclusive, **who have at least 100 weight entries in the facility's DB**. The spread of a client's weight is the maximum weight on record in the facility's DB minus the minimum weight on record in the DB (regardless of age at the time the weight entries were made). The DB contains two tables representing these two relations (among others).

Client (**cid**, name, age, address, phone, ...) DailyRecord(**cid, date**, weight, ...)

The attributes making up the key of each table (relation) are bold-face and underlined. In the actual table definitions, the attribute of **age is declared as NOT NULL** (i.e., it will never be NULL), and **weight can be NULL**.

Complete the following skeletal query to compute the AVERAGE (AVG) spread in the weight of clients between the ages of 45 and 60, inclusive, with at least 100 weight entries, by filling in the blanks.

Alternate (to previous page)

```
SELECT AVG(Temp.MaxWeight) – AVG(Temp.MinWeight)(or AVG(Temp.MaxWeight-Temp.MinWeight))
FROM (SELECT MAX(D.weight) AS MaxWeight, MIN(D.weight) AS MinWeight
      FROM Client C, DailyRecord D
      WHERE C.id = D.id AND D.weight IS NOT NULL AND C.age >= 45 AND C.age <= 60
      GROUP BY C.id
      HAVING COUNT(*) >= 100 ) AS Temp
```

8. Using the same relational schemas, write SQL queries for the following

Customer (SSN: integer, name: string, address: string, city: string)

Account (SSN: integer, AcctNo: integer)

Transaction (AcctNo: integer, ProductId: integer, date: string, quantity: integer)

Product (ProductId: integer, ProductName: string, cost: real)

a) (3 pts) For each product, list the item's ProductId, ProductName, and total (sum) of all item quantities purchased by Nashville customers.

```
SELECT P.ProductId, P.ProductName, SUM(T.quantity)
FROM Customers C, Accounts A, Transactions T, Products P
WHERE C.city = 'Nashville' AND C.SSN = A.SSN AND A.AcctNo = T.AcctNo AND T.ProductId = P.ProductId
GROUP BY P.ProductId, P.ProductName
```

Don't take off this time for missing ProductName in Group By, but -1 for each omission from this standard

b) (2 pts) For EACH item, list the item's ProductId, ProductName, and total (Sum) of all item quantities purchased by Nashville customers, BUT ONLY for those product's with an individual cost > 50 and having a total purchased quantity of greater than 100. You may answer by stating the addition(s) to your query in (a) if you wish)

```
SELECT P.ProductId, P.ProductName, SUM(T.quantity)
FROM Customers C, Accounts A, Transactions T, Products P
WHERE C.city = 'Nashville' AND C.SSN = A.SSN AND A.AcctNo = T.AcctNo AND
      T.ProductId = P.ProductId AND P.cost > 50
GROUP BY P.ProductId, P.ProductName
HAVING SUM(T.quantity) > 100
```

1 point for each addition in red (to whatever query they had from part (a), even if part (a) was incorrect)

Could also use the answer to (a) as a nested query in the FROM clause, creating a temporary Table, with SUM(T.quantity) AS Total, for B, and checking whether Temp.Total > 100 in the WHERE clause of the outer query of B.

9. (5 pts) (Inspired by a Widom practice exercise) Consider a database of researchers and their works (like ResearchGate.com), with relational schema

Researcher (ID, Name, Institution)

Collaborator (ID1, ID2) /* ID1 is a collaborator with ID2. Collaboration is symmetric, so if (abc, wxy) is in the Collaborator table, so is (wxy, abc) */

Follows (ID1, ID2) /* ID1 follows the posts of ID2, where Follows is not symmetric, so if (abc, wxy) is in Follows table, there is no guarantee that (wxy, abc) is also present. */

Consider finding all those researchers for whom all of those they Follow are at different institutions than themselves. Return the names and institutions of all such researchers. One way to write this query is

```
SELECT R.Name, R.Institution
FROM Researcher R
WHERE NOT EXISTS (SELECT * FROM Follows F, Researcher R2
                  WHERE R.ID = F.ID1 AND F.ID2 = R2.ID AND R2.Institution = R.Institution)
```

2 points if literal replacement of NOT EXISTS by NOT IN

Write the query that satisfies the same English specification using the “NOT IN” phrase rather than “NOT EXISTS”:

All researchers who don't follow anyone from the same institution

SELECT R.Name, R.Institution
FROM Researcher R

WHERE R.ID **NOT IN** (SELECT F.ID1
FROM Follows F, Researcher R2
F.ID2 = R2.ID AND R.Institution = R2.Institution)

The set of all researchers following someone who is at the same institution of R

*-1 point if Name used instead of ID; 2 points only if they have * in inner SELECT*

9. (5 pts) (Inspired by a Widom practice exercise) Consider a database of researchers and their works (like ResearchGate.com), with relational schema

Researcher (ID, Name, Institution)

Collaborator (ID1, ID2) /* ID1 is a collaborator with ID2. Collaboration is symmetric, so if (abc, wxy) is in the Collaborator table, so is (wxy, abc) */

Follows (ID1, ID2) /* ID1 follows the posts of ID2, where Follows is not symmetric, so if (abc, wxy) is in Follows table, there is no guarantee that (wxy, abc) is also present. */

Consider finding all those researchers for whom all of those they Follow are at different institutions than themselves. Return the names and institutions of all such researchers. One way to write this query is

```
SELECT R.Name, R.Institution
FROM Researcher R
WHERE NOT EXISTS (SELECT * FROM Follows F, Researcher R2
                  WHERE R.ID = F.ID1 AND F.ID2 = R2.ID AND R2.Institution = R.Institution)
```

Write the query that satisfies the same English specification using the “NOT IN” phrase rather than “NOT EXISTS”:

```
SELECT R.Name, R.Institution
FROM Researcher R
WHERE R.ID NOT IN (SELECT F.ID1 /* could be R.ID here */
                  FROM Follows F, Researcher R2
                  WHERE R.ID = F.ID1 AND F.ID2 = R2.ID AND R.Institution = R2.Institution)
```

Basically same as previous, but for each researcher being considered in turn; will return empty set if R.ID isn't following anyone at same institution

*This addition to previouspage query is what causes nested query to evaluate to NULL for Rs who arent following anyone at same institution. **This isn't needed, but full credit.***

9. (5 pts) (Inspired by a Widom practice exercise) Consider a database of researchers and their works (like ResearchGate.com), with relational schema

Researcher (ID, Name, Institution)

Collaborator (ID1, ID2) /* ID1 is a collaborator with ID2. Collaboration is symmetric, so if (abc, wxy) is in the Collaborator table, so is (wxy, abc) */

Follows (ID1, ID2) /* ID1 follows the posts of ID2, where Follows is not symmetric, so if (abc, wxy) is in Follows table, there is no guarantee that (wxy, abc) is also present. */


Consider finding all those researchers for whom all of those they Follow are at different institutions than themselves. Return the names and institutions of all such researchers. One way to write this query is

```
SELECT R.Name, R.Institution
FROM Researcher R
WHERE NOT EXISTS (SELECT * FROM Follows F, Researcher R2
                  WHERE R.ID = F.ID1 AND F.ID2 = R2.ID AND R2.Institution = R.Institution)
```

Write the query that satisfies the same English specification using the “NOT IN” phrase rather than “NOT EXISTS”:

IDs of all researchers who follow anyone from the same institution

```
SELECT R.Name, R.Institution
FROM Researcher R
WHERE R.ID NOT IN (SELECT F.ID1 /* could be R1.ID here */
                  FROM Follows F, Researcher R1, Researcher R2
                  WHERE R1.ID = F.ID1 AND F.ID2 = R2.ID AND R1.Institution = R2.Institution)
```



A version that doesn't use a correlated query – full credit

9. (5 pts) (Inspired by a Widom practice exercise) Consider a database of researchers and their works (like ResearchGate.com), with relational schema

Researcher (ID, Name, Institution)

Collaborator (ID1, ID2) /* ID1 is a collaborator with ID2. Collaboration is symmetric, so if (abc, wxy) is in the Collaborator table, so is (wxy, abc) */

Follows (ID1, ID2) /* ID1 follows the posts of ID2, where Follows is not symmetric, so if (abc, wxy) is in Follows table, there is no guarantee that (wxy, abc) is also present. */

Consider finding all those researchers for whom all of those they Follow are at different institutions than themselves. Return the names and institutions of all such researchers. One way to write this query is

```
SELECT R.Name, R.Institution
FROM Researcher R
WHERE NOT EXISTS (SELECT * FROM Follows F, Researcher R2
                  WHERE R.ID = F.ID1 AND F.ID2 = R2.ID AND R2.Institution = R.Institution)
```

Write the query that satisfies the same English specification using the “NOT IN” phrase rather than “NOT EXISTS”:

Institutions of all researchers who are followed by a given R

5 pts {
SELECT R.Name, R.Institution
FROM Researcher R
WHERE R.Institution **NOT IN** (SELECT R2.Institution
FROM Follows F, Researcher R2
WHERE R.ID = F.ID1 AND R2.ID = F.ID2)

Correct answers also stem from variants on a different perspective, exemplified at left

All researchers who don't follow anyone from the same institution

10. (5 pts) This question uses the same relational schema as the last problem.

Researcher (ID, Name, Institution)
Collaborator (ID1, ID2)
Follows (ID1, ID2)

What is the average number of Collaborators per Researcher? Importantly, compute this average over only those Researchers who have more than 1 Collaborator. (Your result should be just one number.) .

```
SELECT AVG(Temp.Cnt)  
FROM (SELECT COUNT(*) AS Cnt  
      FROM Collaborator C  
      GROUP BY C.ID1 /* or C.ID2 because Collaborator is symmetric */  
      HAVING COUNT(*) > 1) AS Temp
```

Most will have something very much like this (5 pts)
-2 for each main missing element

Don't need rename as Temp in any of these

```
SELECT AVG(Temp.Cnt)  
FROM (SELECT COUNT(*) AS Cnt  
      FROM Researcher R, Collaborator C  
      WHERE R.ID = C.ID1 /* or C.ID2 because Collaborator is symmetric */  
      GROUP BY C.ID1 /* or R.ID */  
      HAVING COUNT(*) > 1) AS Temp
```

Some will have something more complicated, like this, but still (5 pts)

```
SELECT AVG(Temp.Cnt)  
FROM (SELECT COUNT(*) AS Cnt  
      FROM Researcher R, Collaborator C  
      WHERE R.ID = C.ID1  
      GROUP BY C.ID1) AS Temp  
WHERE Temp.Cnt > 1
```

11. (5 pts) Consider the relational schema

HRel (ID, name)

+2 for one of correct answers selected, and 1 point each additional option

10 minutes

Circle all queries below that return the tuples of HRel with the top 5 values of ID

(a) SELECT H.ID, H.name
FROM HRel H
WHERE NOT EXISTS
 (SELECT *
 FROM HRel H1, HRel H2
 WHERE H.ID = H1.ID AND H1.ID < H2.ID
 GROUP BY H1.ID
 HAVING COUNT(*) >= 5)
ORDER BY ID DESC

(b) SELECT H.ID, H.name
FROM HRel H
WHERE (SELECT COUNT (*)
 FROM HRel H2
 WHERE H.ID < H2.ID) < 5
ORDER BY H.ID DESC

} There are less than 5 IDs greater than H.ID

(c) SELECT H.ID, H.name
FROM HRel H
EXCEPT
SELECT H.ID, H.name
FROM HRel H
WHERE (SELECT COUNT (*)
 FROM HRel H2
 WHERE H.ID < H2.ID) >= 5
ORDER BY H.ID DESC

} There are at least 5 IDs greater than H.ID

(d) SELECT H.ID, H.name
FROM HRel H
EXCEPT
SELECT H1.ID, H1.name
FROM HRel H1, HRel H2
WHERE H1.ID < H2.ID
GROUP BY H1.ID, H1.name
HAVING COUNT(*)+1 > 5
ORDER BY ID DESC

(e) None of the above

0 pts if option (e) selected, whether alone or in conjunction with any other answer (though unlikely anyone did that)

Question 11 comments.

I had intended that ID was the key of HRel, and I think most everyone assumed this, even though I didn't bold face and underline it. So, here appear to be almost no one affected by that omission, but the following comments are worth making and understanding.

Even without specifying the primary key, these answers will be right using what I think is a reasonable interpretation (but not the only one).

Lets say that we have repeats of ID, supposing its not the key. Consider this

ID,name	larger IDs	Place aka COUNT(*)+1
11,a	<	1
8,b	< 11	2
8,c	< 11	2
8,d	< 11	2
6,e	< 8,8,8,11	5
6,f	< 8,8,8,11	5
3,g	< 6,6,8,8,8,11	7
3,h	< 6,6,8,8,8,11	7
3,i	< 6,6,8,8,8,11	7
2,j	< 3,3,3,6,6,8,8,8,11	10
1,k	< 2,3,3,3,6,6,8,8,8,11	11

You've probably seen rankings (e.g., of colleges), in which ties receive that same number (e.g., ties for 2nd), but then the ordering picks up after the last tied value.

Look at the simplest of the queries (b)

```
SELECT H.ID, H.name
FROM HRel H
WHERE (SELECT COUNT (*)
      FROM HRel H2
      WHERE H.ID < H2.ID) < 5
ORDER BY H.ID DESC
```

In query (b), rows of HRel that would pass the WHERE test are 11, 8, 6, and all the corresponding rows would be returned (6 of them).

Consider (d), which I think is the next most intuitive

```
SELECT H.ID, H.name
FROM HRel H
EXCEPT
SELECT H1.ID, H1.name
FROM HRel H1, HRel H2
WHERE H1.ID < H2.ID
GROUP BY H1.ID, H1.name
HAVING COUNT(*)+1 > 5      (will include groups (3,g), (3,h)...(1,k))

ORDER BY ID DESC
```

Again, the right answer is returned under a reasonable interpretation that is often used.

I intended ID to be the key, which would have resulted in much less potential for misunderstanding, but if you were one of the very few who didn't assume that ID was key (and you didn't assume the interpretation I just laid out), then see me. In particular, I think that a very few might have interpreted the question as asking to either (i) return exactly 5 rows with highest Ids, or (ii) return rows with the highest ID values regardless of ties, which would have been rows with values of 11 down to 2 in the example above.