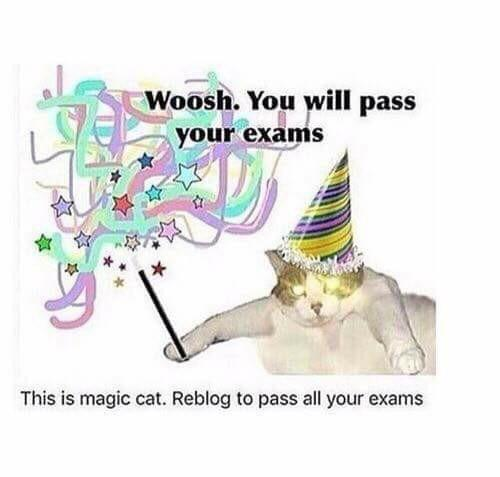
**-**

**Congratz everyone, enjoy your holidays**

**GOOD LUCK EVERYONE. +1+1+1**

**HERE YOU GO: **

**And everyone liked that**

**SHOUTOUT TO EVERYONE WHO CONTRIBUTED ANSWERS IN THE GOOGLE DOCS AND GROUP CHAT OVER THE SUMMER!**

**IF YOU AGREE WITH AN ANSWER, ADD A “+1” NEXT TO IT**

[**April 2018**](https://q.utoronto.ca/courses/95202/files/4073842/download?wrap=1)

**Q1**

a) 2, 4, 100 **+1 +1 +1, +1, +1, +1 +1 +1**

b) 0, 1, 2, 4, 100 **+1 +1 +1, +1, +1, +1 +1**

c) 2, 4, 100 **+1, +1, +1 +1+1 +1**

d) No. **+1 +1 +1, +1, +1+1 +1**

Three:

|  |  |  |
| --- | --- | --- |
| h | i | j |
| 1 | 3 | 5 |
| 1 | 6 | 6 |
| 1 | 9 | 7 |
| 1 | 2 | 8 |
| 2 | 9 | 10 |

Four:

|  |  |  |
| --- | --- | --- |
| m | n | o |
| 1 | 1 | 3 |
| 4 | 1 | 6 |
| 7 | 1 | 9 |
| 10 | 1 | 2 |

Last tuple in Three wont be included in second query.+1

We can find smaller relations which show that this is a no.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Three |  |  |  | Four |  |  |
| h | i | j |  | m | n | o |
| 0 | 0 | 0 |  | 0 | 0 | 0 |
| 1 | 1 | 1 |  | 1 | 0 | 1 |
| 2 | 2 | 2 |  | 2 | 1 | 0 |
|  |  |  |  | 3 | 1 | 1 |

e) No. I.3 is violated. **+1 +1 +1, +1+1**

f) No. I.1 is violated and key constraint of table Two. **+1 +1 +1, +1+1**

**Q2**

1. **+1, +1+1+1**



Question

Also, I think this solution would have worked without Message.to=Delivered.uid, since Message.mid implies Message.to +1

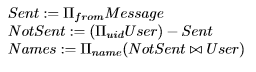
b) I don’t think the below constraint is correct since mid is a key (**+1**). Instead self-join Delivered, say D1 and D2, and select D1.mid = D2.mid and D1.uid < D2.uid. **(+1)**



（Edit: I think the only way to check if two messages are the same is to check its content, not its mid, because every message has its own mid even when they have the same content）

\select\_{ D.mid = D2.mid ^ D.uid <> D2.uid} (pD Delivered x pD2 Delivered)

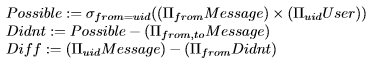
c)

+

Are we allowed to subtract Sent from User? Since they have different attribute names

U can always rename it by Sent(uid):=..., But i think it should work anyway+1

d)



(Unsure about this one)

**+1 +1 +1**

Everything else should work

We exclude pairs where user1.uid = user2.uid because then if a user didn’t send a message to herself it would count towards users who didn’t send at least one message to everyone else

**Q3**

a)

SELECT DISTINCT campus

FROM Student

WHERE campus NOT IN (SELECT campus

(I have sID instead of campus) FROM Took JOIN Student ON Took.sID = Student.sID);

Not sure if this query is equivalent to the one above

(SELECT DISTINCT campus

FROM Student)

EXCEPT

(SELECT DISTINCT campus

FROM Student NATURAL JOIN Took); +1**, +1+1**

The above two look equivalent to me. I thought of the second one as well.

I think this one is the simplest:

SELECT campus

FROM Student

WHERE sID NOT IN (SELECT sID

FROM Took);

Pretty sure this is equivalent to the one using set semantics though.

b) **+1 +1+1, +1**

SELECT Student.sID, cgpa, avg(grade) AS avgGrade, count(DISTINCT dept) AS numDept

FROM (Student **LEFT** JOIN Took ON Student.sID = Took.sID) a **LEFT** JOIN Offering ON a.oID = Offering.oID

WHERE campus IS NOT NULL

GROUP BY Student.sID, cgpa;

We need to use an outer join because we must include students who have not taken any course. **+1, +1**

c) **+1 +1, +1**

-- Find courses that had less than 100 people enrolled

CREATE VIEW Enrollment AS

SELECT dept, cnum

FROM Took JOIN Offering ON Took.oID = Offering.oID

WHERE year > 2000

GROUP BY dept, cnum

HAVING count(sID) <= 100;

(Edit: I think in the Group by clause there should also be oID, because in the question it says “if one of those offerings had an enrollment of only 90, don’t include that”, so we’re actually counting the number of enrollment of each offering, not the entire course)

-- Find courses that were offered more than 10 times since 2000

CREATE VIEW Ten AS

SELECT Course.dept, Course.cnum, name

FROM Course, Offering

WHERE Course.cnum = Offering.cnum AND Course.dept = Offering.dept AND year > 2000

GROUP BY Course.dept, Course.cnum, name

HAVING count(oID) >= 10;

-- Choose tuples that are in Ten and not in Enrollment i.e. have been offered 10 times without having enrollment less than 100

SELECT \*

FROM Ten

WHERE (dept, cnum) NOT IN (SELECT \*

FROM Enrollment);

**Q4 +1+1, +1**

There’s probably an easier way to check if the table is empty. (According to Mark, we’ll need to remember the syntax for everything so it’s worth looking over the examples we had)

I think your first qString could have been “SELECT count(\*) from myCourses;”. Then you can omit the while loop and just write

rs.next();

if (rs.getInt(**1**) > 0 ) //rest as above

That’s maybe simpler?

**Good point, noted. +1, +1**

**Q5**

1. **+1 +1 +1 +1 +1**

One:

|  |  |
| --- | --- |
| c | d |
| 3 | 1 |
| 2 | 2 |

Two:

|  |  |
| --- | --- |
| e | f |
| 2 | 1 |
| 3 | 2 |
| 2 | 2 |
| 3 | 1 |

Three:

|  |  |  |  |
| --- | --- | --- | --- |
| g | h | i | j |
| 1 | 1 |  |  |
| 2 | 2 |  |  |

This is correct after testing on teach.cs. 👍

b) Yes. The update would not occur as when it cascades from Two to Three, the reaction policy would restrict it, causing an error. **+1 +1 +1 +1 +1**

c) Last statement causes an error since (3, 4) is not present in table Two (violates reference constraint). **+1 +1 +1 +1, +1+1**

**Q6**

a) {LMP}+ = {L, M, P, N, J, Q} **+1 +1 +1 +1 +1**

b) No. The closure of MN is MNLJQ which does not include all attributes. **+1 +1 +1+1 +1**

c) MPK **+1 +1 +1+1, +1**

d) The closure of MPK is MPKNLJQ which includes all attributes. Furthermore, we cannot reduce MPK and produce an equivalent FD. We know that K must be included in any key since it does not appear in any dependency. The same is true for P, because it is only included in the LHS of any dependency. So any key must include KP. The closure of KP is KP so we know we must include at least one more attribute to produce a key. **+1 +1 +1+1**

**Q7 +1 +1+1, +1**

|  |  |  |  |
| --- | --- | --- | --- |
| Dependency | Violates BCNF? | R1 Attributes | R2 Attributes |
| E -> F | Yes | EF | ABCDE |
| A -> CE | Yes | ACEF | ABD |
| BD -> A | No |  |  |
| DF -> E | Yes | DFE | ABCDF |

**Q8**

a) Min. Basis = {IK -> H, H -> L, L -> K, I -> J} **+1 +1 +1 +1, +1**

b) New Schema: **+1, +1**

R1(I, K, H)

R2(H, L)

R3(L, K)

R4(I, J)

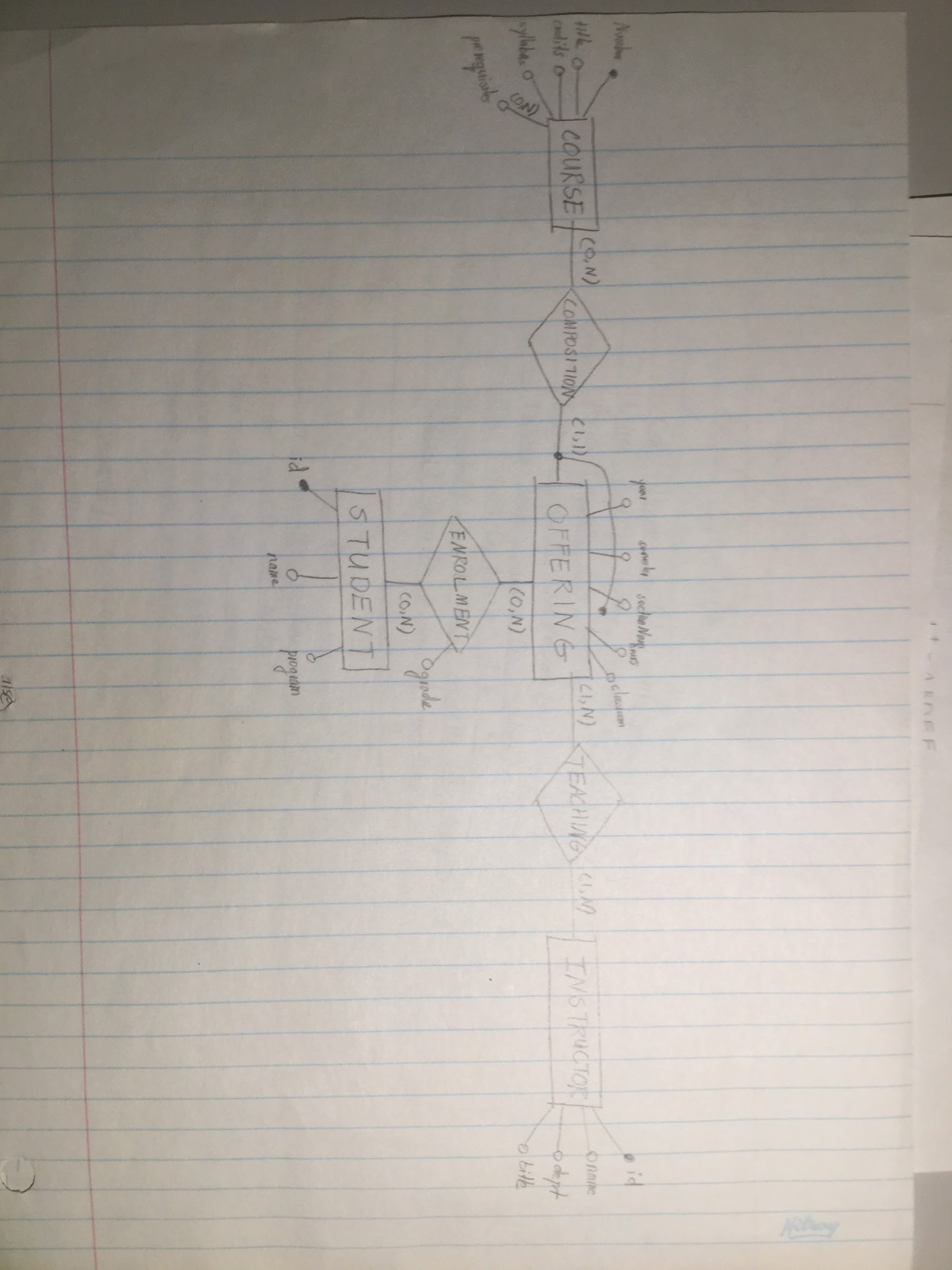
c) Yes. R5(H, I, K, M). I got R5(H,I,M) since HIM\* = HIMLKJ. +1 (+1**, +1** to R5(H,I,M))

**Q9**

{E -> BDF, F -> BDE, BD-> EF} **+1 +1 +1+1 +1**

|  |  |
| --- | --- |
| B+ = B | Nothing |
| `D+ = D | Nothing |
| E+ = DBEAF | E → BDF (1) |
| F+ = FEDBA | F → BDE (2) |
| BD+ = ABDFE | BD → EF (3) |
| BE+ = BEDAF | BE → DF weaker than (1) |
| BF+ = BFEDA | BF → ED weaker than (2) |
| DE+ = DEBAF | DE → BF weaker than (1) |

Since E, F and BD are superkey, any superset of E OR F OR DB can only yield FDs that are weaker than the one we have.

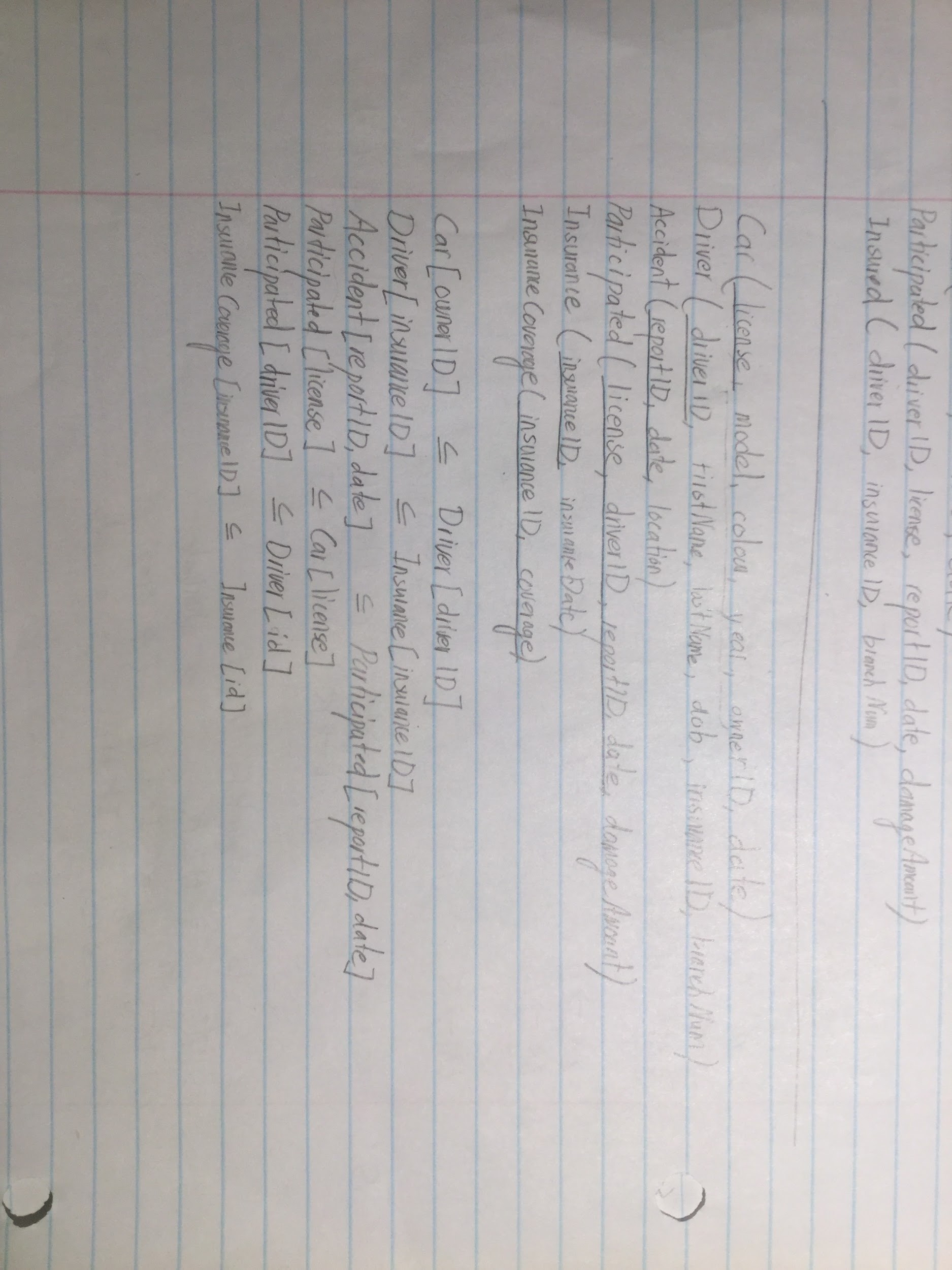
**Q10 (If anyone could do these last two I’ll love you 3000)**

**+1 +1 i got the same**

This looks correct. Just to confirm --- the relationships Composition and Teaching (and their multiplicities) are implied based on the description of the database and baseline intuition?

I might have put course into a **recursive** relationship (HasPrerequisites) instead. That way when you make this into a database, you don’t end up with a course which exists as a prerequisite but doesn’t exist as a course. 👍

**Q11**

****

**+1**

Driver (driverID, firstName, lastName, dob, insuranceID)

Car (license, model, color, year, ownDate, driverID)

Accident (reportID, date, license, location, damageAmount)

Insurance (insuranceID, coverage, insuranceDate, branchNum)

Accident[license] < car[license]

insurance[driverID] < driver[driverID]

car[driverID] < driver[driverID]

Driver[insuranceID] < Insurance[insuranceID]

In this case coverage cannot be null?

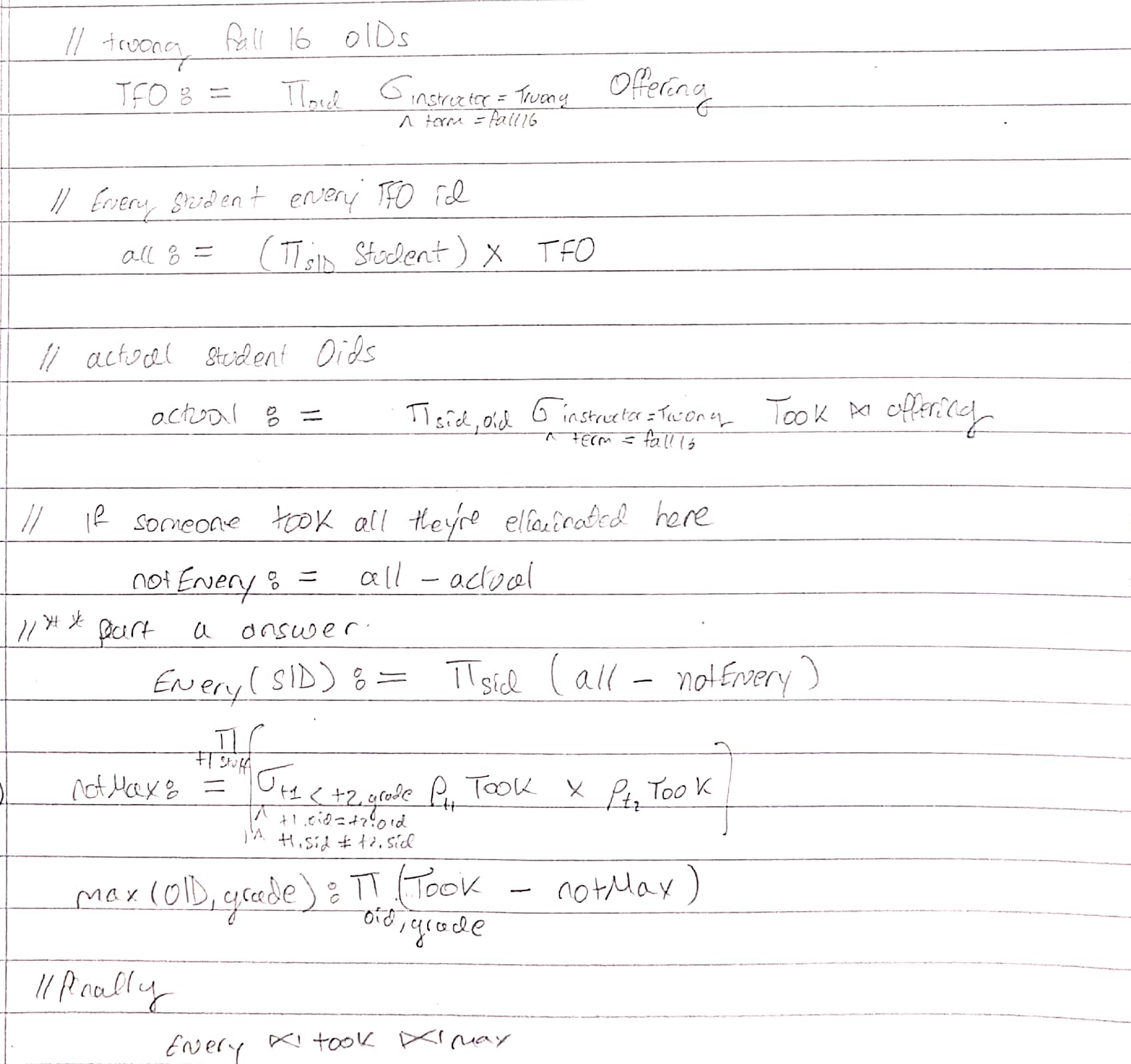
**April 2017**

**Q1**

a) An instructor can’t teach for two different departments unless neither of the courses is first-year. **+1+1**

b) A -B instead of B - A ? **+1 +1+1+1**

c)

**+1**

(slightly different than mine but overall the workflow looks right)**+1**

c) Create a new relation with code (assuming Diagnoses.code is a unique identifier for the ill ness), shortName, longName.

d) all patients must be admitted before staying in ICU and must be discharged from ICU before being dishcharged from the whole hospital

e)

TwoStays := \select\_{A1.pid<A2.pid}(\rename\_A1 Admissions ) \cross (\rename\_A2 Admissions)

AtLeastOneCritical := \select\_{A1.admitID = ICUStays.admitID OR A2.admitID = ICUStays.admitID } TwoStays \cross ICUStays

AtLeastTwoNonCritical := \project\_(A1.admitID, A2.admitID) TwoStays -- \project\_(A1.admitID, A2.admitID) AtLeastOneCritical

Answer := \project\_pid (\select\_{Admissions.admitID = A1.admitID} Admissions \cross AtLeastTwoNonCritical )

**Q2**

a) count(DISTINCT stuff) <= count(stuff) <= count(\*) **+1+1, +1**

b)\forall t1, t2: t1.P=t2.P AND t1.Q=t2.Q \implies t1.R = t2.R**, +1 +1**

c) Yes. **+1, +1**

d)

|  |  |  |
| --- | --- | --- |
| SQL Join | Min # of rows | Max # of rows |
| Hansel JOIN Gretel ON one = four | 0 **+1+1** | 150 **+1+1** |
| Hansel NATURAL JOIN Gretel | 150**+1, +1** (0) | 150**+1, +1** (0) |
| Hansel LEFT JOIN Gretel ON one = four | 10 **+1+1** | 150 **+1+1** |
| Hansel FULL JOIN Gretel ON one = four | 15 **+1, +1 (25) + 1 +1** | 150 **+1+1, +1** |

e) True, False, False, False

**Q3**

a) **+1+1+1**

CREATE VIEW ExtendedType AS

SELECT iid, type, b AS supertype

FROM Item LEFT JOIN Subcategory ON a = type;

b) **+1+1**

CREATE VIEW OrderCount AS

SELECT EXTRACT (year FROM owhen) AS year, iid, sum(quantity)

FROM LineItem JOIN Order ON LineItem.oID = Order.oID

GROUP BY EXTRACT (year FROM owhen), iid;

CREATE VIEW Popular AS

SELECT year, iid, max(sum) as totalquantity

FROM OrderCount

GROUP BY year, iid;

c)-1

SELECT DISTINCT iid, type, supertype

FROM Popular

WHERE year < 2010

GROUP BY iid

HAVING count(\*) >= 10;

（I think this should join EntendedType from part(a) to get the type and supertype）

d)

CREATE VIEW OrderPrice AS

SELECT oid, quantity\*price AS amount

FROM LineItem JOIN Item ON LineItem.iid = Item.iid;

INSERT INTO Charge

(SELECT cid, Orderr.oid, sum(amount) AS amount

FROM OrderPrice JOIN Orderr ON OrderPrice.oid = Orderr.oid

GROUP BY cid, oid);

**Q4 +1 +1 +1, +1**

X:

|  |  |  |
| --- | --- | --- |
| a | b | c |
| 5 | 2 | 9 |
| 2 | 8 | 9 |
| 1 | 0 | 6 |

Y:

|  |  |  |
| --- | --- | --- |
| d | e | f |
| 2 | 1 | 6 |
| 6 | 1 | 8 |
| 4 | 4 | 6 |
| 9 | 3 | 8 |

Z:

|  |  |  |
| --- | --- | --- |
| g | h | i |
| 6 | 5 | 3 |
| 8 | 1 | 4 |

**Q5**

try {

String qString = “SELECT max(guess) FROM Guesses;”;

ps = conn.prepareStatement(qString);

rs = ps.executeQuery();

while (rs.next()) {

biggest = rs.getInt(1);

}

qString = “UPDATE Guesses SET guess = ? WHERE name = ?”;

ps = conn.prepareStatement(qString);

ps.setInt(1, biggest+1);

ps.setString(2, who);

ps.executeUpdate();

} catch (SQLException se) {System.out.println(“Exception”);}

**Q6**

**NOT APPLICABLE +1, +1, +1**

**Q7**

1. No **+1**

Yes **+1**

Yes (this may be “No” **, +1**but it depends on XML syntax) **+1 to No**

No **+1**

Yes **+1**

2. <!ATTLIST d real （true|false|“unsure”） #REQUIRED> **+1**

3. <!ELEMENT junk (c, c, c+, (c\*, b+, c+, b\*))>

**I got this for 7.3 -** <!ELEMENT junk (c,c,c+, (b|c), (b|c)+) > **+1 got the same +1 +1 +1**

**Q8, +1, +1, +1**

Use final chase test tableau as a counter example:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | C | D | E |
| a | b1 | c | d | e1 |
| a2 | b | c2 | d2 | e |

**Q9 +1+1 +1 +1 +1**

Final Relations:

R1(A, C, D, F) {A -> CDF}

R2(E, B, C, D) {E -> BCD}

R3(E, F) {}

**Q10**

Min Basis: {D -> A, D -> E, E-> B, E-> C} **+1+1+1 +1**

**Q11**

a) An FD satisfies 3NF if the LHS is a superkey or the RHS is prime, i.e., a member of any superkey. **+1, +1**

b) Yes **+1, +1**

c) Final 3NF Schema:

R1(R, P, T)

R2(Q, S, U)

R3(P, Q, T)

R4(Pw, Q, R) + 1

d) **+1+1+1+1 +1 to all**

No **+1**

No **+1**

Yes **+1**

Yes **+1**

No **+1**

**Q12 +1+1+1 to all**

1. True
2. False
3. False
4. True
5. True
6. False
7. True
8. True

**Q13**

MP (ID, firstName, lastName,dob)

Party (name, colour, abbreviation)

Leads (ID, name)

Member (ID, name, dateJoined)

speech(ID, date, time, number, text)

Topic (topicName, date, time, ID, number)

Session (number, sittings)

leads[ID] < MP[ID]

member[ID] < MP[ID]

leads[name] < party[name]

member[name] < party[name]

speech[number] < session[number]

**Different Answer +1 +1**

speech(date, time, mp, text, session)

mp (ID, firstName, lastName,dob)

party (name, colour, abbreviation, mpLeader)

on(topicName, date, time, mp)

Member (mpID, pName, dateJoined)

Sessions (number, sittings)

speech(mp) mp(id)

speech (session) session(number)

party(mpLeader) mp(ID)

on(date, time, mp) speech(date, time, mp)

member(mpID) mp(id)

member(pName) party(name)

**December 2017**

**Q1**

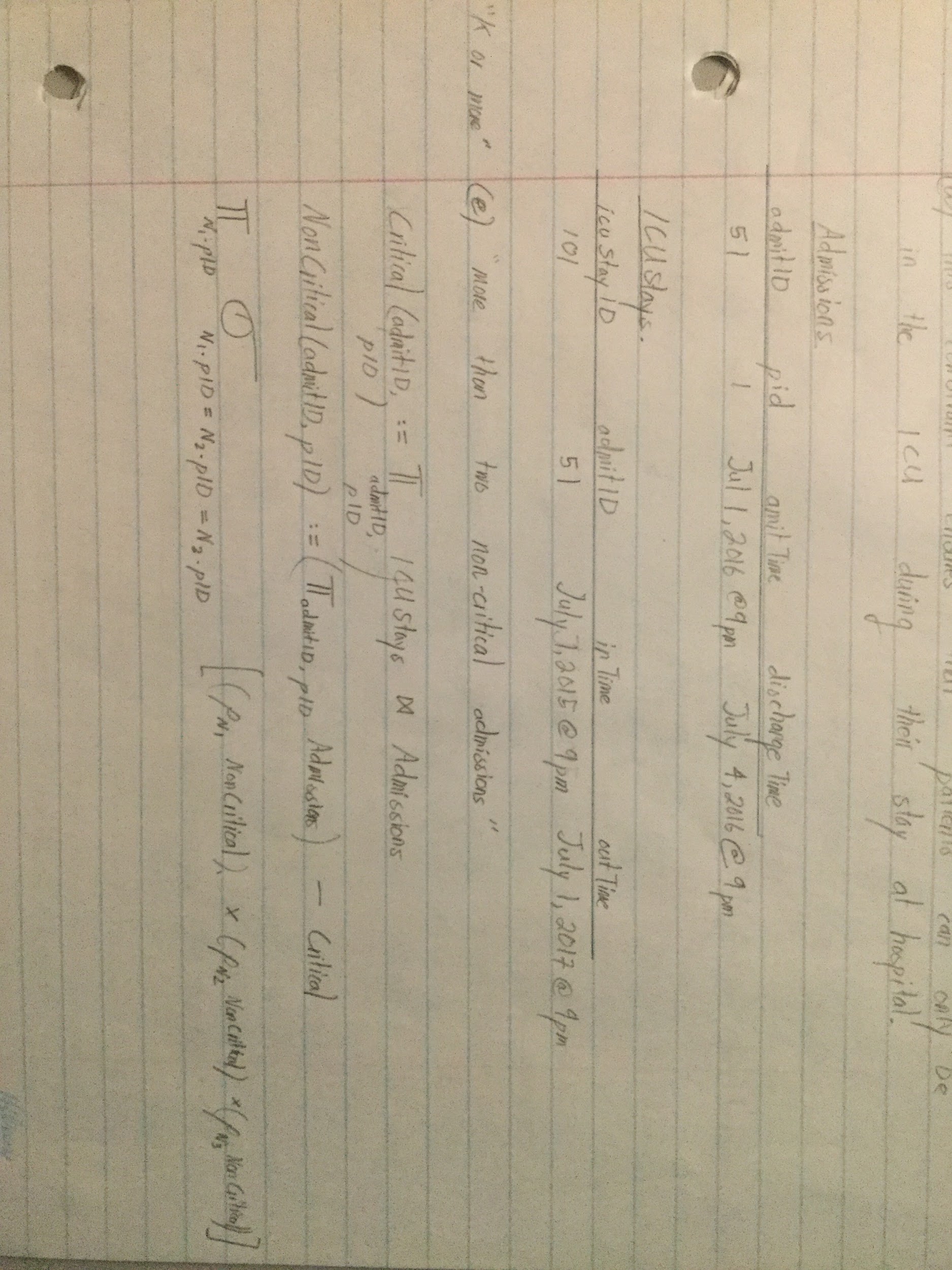
1. 0,1

0,1,2,5.**+1 +1 +1**

b) YES; the LHS of every FD is a superkey. **+1+1**

c) Add a new relation DiagnosisName(code, shortName, longName) and integrity constraint Diagnosis[code] subset of DiagnosisName[code]. **+1+1**

d) This constraint ensures that patients can only be in the ICU during their stay at the hospital. **+1+1+1**



(I believe *N1.admitID < N2.admitID < N3.admitID* is required in condition under select)

**Q2**

a) Think I got it. The query doesn’t work if a person had a diagnosis when they were less than one year old and then was admitted and diagnosed when they were older than one year old. This is because the Neonates relation is joined back with Admissions meaning any pid that matches a pid in Neonates will be in the resulting relation. So if a person has a diagnosis as an adult, then thats the diagnosis that would be reported instead of their neonatal one. +1

b) We are unable to calculate this value in RA because we can’t express constants in RA.

c) YES; Natural join ICUStays and Admissions and use the constant value of per-admission ICU percentage (outTime - inTime) / (dischargeTime - admitTime) \* 100 in a column, say ICU\_Percentage.

**Q3**

a)**+1**

CREATE VIEW Many AS

SELECT manager,name

FROM Manages JOIN Employee ON manager = eID

GROUP BY manager,name

HAVING count(junior) > 4;

SELECT manager, name, avg(amount)

FROM Many NATURAL JOIN Manages JOIN Sales ON junior = eID

WHERE EXTRACT(year from day) = 2009

GROUP BY manager, name;

b) **+1+1 +1+1**

CREATE VIEW DepartmentSize AS

SELECT Department.name, dID, count(eID) AS numEmp

FROM Department LEFT JOIN Employee ON Department.dID = Employee.dept

GROUP BY Department.name, dID;

c) **+1+1+1**

(SELECT name, dID, numEmp, ‘small’ AS size

FROM DepartmentSize

WHERE numEmp < 3)

UNION

(SELECT name, dID, numEmp, ‘medium’ AS size

FROM DepartmentSize

WHERE numEmp >= 3 AND numEmp <= 6);

d) **+1+1**

CREATE VIEW MonthlySales AS

SELECT eID, EXTRACT (month FROM day) as month, sum(amount)

FROM Sales

WHERE EXTRACT (year FROM day) = 2017 AND amount > 0

GROUP BY eID, EXTRACT (month FROM day);

SELECT eID

FROM MonthlySales

WHERE eID NOT IN (SELECT a.eID

FROM MonthlySales a, MonthlySales b

WHERE a.eID = b.eID AND a.month > b.month AND a.sales < b.sales);

**Q4 +1+1**

I had trouble with this one. How did you come up with counterexamples?

My strategy is to look at how the two queries differ and design the table around that difference.

Manages:

|  |  |
| --- | --- |
| manager | junior |
| 1 | 2 |
| 2 | 2 |
| 1 | 3 |

Sales:

|  |  |  |
| --- | --- | --- |
| eID | day | amount |
| 2 | 10 | 100 |
| 2 | 11 | 5 |

First Query Produces:

|  |
| --- |
| manager |
| 1 |
| 2 |
| 1 |

Second Query Produces:

|  |
| --- |
| manager |
| 1 |
| 2 |

**Q5 (if anyone could confirm this one, thatd be helpful)**

**+1 +1 to all**

|  |  |  |
| --- | --- | --- |
| RA | SQL | JDBC |
| yes | yes | Yes **+1+1** |
| yes | yes | Yes **+1** |
| no | no | Yes **+1** |
| no | yes | Yes **+1+1** |
| yes | yes | Yes **+1+1** |
| yes | yes | Yes **+1+1** |
| no | yes | Yes **+1+1** |
| no | yes | Yes **+1+1** |

**Q6**

a) Remove tuple (1, 5) from One. Remove tuple (1, 5, 7) from Two. Remove tuples (1, 2), (5, 5) and (1, 0) from Three. Four stays the same. **+1 +1+1 +1**

b) Yes. After “i INT” in table Four, add REFERENCES One(b) ON UPDATE SET NULL.**+1**

I got FOREIGN KEY i REFERENCES Three(g) ON UPDATE SET NULL

**Q7 +1 +1**

|  |  |  |  |
| --- | --- | --- | --- |
| D -> AC | No |  |  |
| A -> B | Yes | A, B | A, C, D, E |
| C -> ED | No |  |  |
| E -> AB | Yes | A, B, E | C, D, E |

**Q8**

a) The projection of S onto R1 is {JN -> L}. **+1 +1**

b) Original table: **+1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| J | K | L | M | N |
| j | k1 | l | m1 | n |
| j | k | l2 | m2 | n |
| j | k | l3 | m | n3 |

After chase test:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| J | K | L | M | N |
| j | k1 | l | m1 | n |
| j | k | l | m1 | n |
| j | k | l3 | m | n3 |

The decomposition is lossy. Use the final table as a counter example.

(Edit: I don’t think this is lossy, the second row by R2 should all be letters without subscripts, because the FD N->M first and then JM->L. )

**Q9**

a) The given FD’s are already a minimal basis. **+1+1+1**

b) Relations: **+1+1+1**

R1(E, D, B)

R2(B, C, A)

R3(F, E)

R4(D, C)

c) No. FE is a superkey for the whole schema since {FE}+ = FEDBCA. **+1+1+1**

**Q10**

a) {KL}+ = KLNM **+1+1+1**

b) No. The closure of KL does not encompass all attributes so it cannot be a key. **+1+1+1**

c) JL. **+1+1+1**

d) The closure of JL captures all attributes of the relation. Also, we know that both J and L need to be in any key since they appear only on the left side of any dependency. **+1+1+1**

**Q11 (Can someone double check?)+1**

Just to confirm, if |beside| >= |nonsense| then it’s possible? Also what does |beside| even mean? I know that |stuff| = number of entities in stuff and |nonsense| = number of entities in nonsense.

Edit: |beside| means how many different relationships are there.

**+1+1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |stuff| | |beside| | |nonsense| | Possible? | Explanation. |
| 15 | 0 | 15 | No | Membership in beside is not optional. |
| 2 | 3 | 5 | No | Every element of Nonsense needs to participate in beside. |
| 9 | 5 | 2 | Yes |  |
| 8 | 8 | 8 | Yes |  |
| 4 | 16 | 4 | Yes |  |
| 11 | 11 | 5 | Yes |  |

**Q12**

True, False, False, False, False, True, False, True(FALSE**+1+1 +1 +1**)

**Q13 +1 +1**

Franchise(ID, owner)

Job(address, date, price, fid)

Assigned(ID, address, date, crewID)

Crew(crewID)

Employee(eNum, name, address)

Member(eNum, crewID)

TrainingModule(name, level)

Trained(eNum,name,date,score)

Assigned[ID] ⊆ Franchise[ID]

Assigned[address, date] ⊆ Job[address, date]

Assigned[crewID] ⊆ Crew[crewID]

Member[eNum] ⊆ Employee[eNum]

Member[crewID] ⊆ Crew[crewID]

Trained[name] ⊆ TrainingModule[name]

Trained[eNum] ⊆ Employee[eNum]

**Q14**

a)

1. Not Valid. Doesnt include attribute for year in book element. Also needs at least one author. **+1+1**
2. Not Valid. ID cannot begin with an int. **+1 +1**

*(Question: but here ID is a string tho, I think its type is actaully valid??? )*

*(No it’s not, check the DTD and XML files posted on lecture website)*

1. Valid. **+1+1**
2. Not Valid. Author needs to appear before rating.**+1** I think the first rating is also required to have PCDATA **+1+1**

b) **+1 +1+1**

(Changes from the DTD given in the question are in bold)

**<!ELEMENT booklist (book+)>**

<!ELEMENT book (author+, rating\*)>

… ATTLIST stuff

<!ELEMENT author (#PCDATA)>

<!ELEMENT rating (#PCDATA)>

… ATTLIST stuff

<?xml version=“1.0” standalone=“no”?>

<!DOCTYPE **booklist** SYSTEM “bookstore.dtd”>

**<booklist>**

<book>

…

</book>

**</booklist>**

**Q15 +1 +1**

1. Valid JSON that Validates**+1**
2. Valid JSON but does not validate. Reason: missing “num” and “students”.**+1**
3. Valid JSON but does not validate. Reason: additional properties is false for “students”.**+1**
4. Valid JSON that validates. **+1**
5. Invalid JSON. Keys must be strings.**+1**