Final Exam (CS 561-B) - Spring 2021 (Thu., 5/6/2021) 6:40-9:10 PM ET.

Due May 6 at 9:10pm **Points** 100 **Questions** 6
Available May 6 at 6:40pm - May 6 at 9:10pm about 3 hours
Time Limit 150 Minutes

Instructions

IMPORTANT: Please read the following instructions carefully before starting the exam.

This exam is worth 100 points (50% of your grade), and you have up to 150 minutes (2.5 hours) to finish the exam. This exam is closed book, closed notes and without looking up on-line. Write your answers "clearly."

Please be ABSOLUTELY sure you are the ONLY PERSON in the room where you're taking this exam--any indication of collaboration with anyone will result in the final grade of "F" for the course. No exceptions!

Please note/remember the following points while taking the exam:

- Although you're taking the exam in your own space, please keep all of the reference material closed, including the textbooks, notes, SQL HW answers, on-line access, etc. The exam is to test your knowledge and understanding of the topics, NOT your ability to look up and search!
- Again, please make sure you are alone in your space -- this is NOT a group exam.
- Questions (especially those involving key concepts) are not necessarily complex, however, I'm looking for answers "in your own words" (vs., taking answers from exercise questions, slides, etc.). For that reason, I expect everyone's answers to be "unique" (i.e., not resembling someone else's answers).
- When writing SQL queries:
  - Use appropriate indentations, etc. to make sure your queries are easy to read and understand. *Points will be deducted if you do not use proper indentation* (e.g., if you write a query or a part of a query in a long string such as "select A, B, C, D from table where . . . . ". You should write each clause (SELECT, FROM, WHERE, etc.) in a separate line.
  - Please use only the standard SQL features we covered in class -- do **NOT** use any other functions, etc. you might have used in your HW assignments. 50% of the total points for the question will be deducted if you use those non-standard syntactic features.
  - You are only allowed to use the simple syntax of func(x) for ALL aggregate functions (e.g., sum(quant)) -- i.e., do **NOT** use any other variations of syntax such as CASE . . . WHEN . . . THEN inside aggregate functions. These syntactic features are hiding implicit joins. 50% of the total points for the question will be deducted if you use those other variations of syntax.
  - Use the table names and column names provided in the schema(s) in the questions.
  - Use appropriate JOINS -- i.e., you may need to use OUTER JOINs in some cases.

This quiz was locked May 6 at 9:10pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	149 minutes	97 out of 100

Correct answers are hidden.

Score for this quiz: **97** out of 100
Submitted May 6 at 9:09pm
This attempt took 149 minutes.

Question 1

9 / 10 pts

For the year 2021 and for each customer, compute the average sales for each quarter as follows (some customers may not have any sales transactions for certain quarters, and obviously, for those, you don't need to compute the total sum).:

sales (prod, cust, yr, mo, day, state, quant)

```
CUST    QUATER   TOTAL_SUM
=====   ======   =========
Dan          1         143
Dan          3         243
Sam          3        2434
Claire       2        1592
Helen        4         982
. . .
```

Your Answer:

select cust ,(mo-1)/3+1 quater, sum(quant) total_sum

from sales

where yr=2021

group by cust, quater

you have to cast (mo-1)/3+1 as int;

Question 2

10 / 10 pts

For all customer, product, month, and year combinations, compute the average sales quantities for:

1. The year before
2. The current year/given year
3. The year after

E.g., the first of the example below shows the average sales quantity for (“Sam”, “Apple”, “January”) for the years, 2013, 2014 and 2015.

```
CUST   PROD   MONTH   YEAR   AVG_YR_BEF   AVG_CUR_YR   AVG_YR_AFT
====== ====== =====   ====   ==========   ==========   ==========
Sam    Apple      1   2014          523           90         2031
Mia    Bread     12   2020          189          382           90
Claire Eggs       8   2007           57          192          453
. . .
```

Your Answer:

with current as (

select cust, prod, mo, yr, round(avg(quant)) avg_q

from sales

group by cust,prod,mo,yr)

select a.cust, a.prod, a.mo, a.yr, b.avg_q avg_yr_bef, a.avg_q avg_cur_yr, c.avg_q avg_yr_aft

from current a

left join current b

on a.cust=b.cust and a.prod=b.prod and a.mo=b.mo and b.yr=a.yr-1

left join current c

on a.cust=c.cust and a.prod=c.prod and a.mo=c.mo and c.yr=a.yr+1

Question 3

10 / 10 pts

For each customer, product and month, compute:

1. The product's average sales for the given customer for the given month.
2. The average sales for the given customer and product but for all other months.
3. The average sales for the given customer and the month but for all other products.

sales (prod, cust, yr, mo, day, state, quant)

```
CUST    PROD    MONTH   THE_AVG   OTHER_MONTHS_AVG   OTHER_PROD_AVG
======= ======= =====   =======   ================   ==============
Bloom   Banana     12       465                982              273
Dan     Eggs        3       193                248              732
Sam     Apple       6       987                243              592
. . .
```

Your Answer:

with Q1 as(

select cust, prod, mo, round(avg(quant)) avg_q

from sales

group by cust,prod,mo

),

Q2 as(

select s1.cust, s1.prod, s1.mo, round(avg(s2.quant)) as other_mo_avg

from sales as s1

left join sales as s2

on s1.cust = s2.cust and s1.prod = s2.prod and s1.mo != s2.mo

group by s1.cust, s1.prod, s1.mo

),

Q3 as(

select s1.cust, s1.prod, s1.mo, round(avg(s2.quant)) as other_prod_avg

from sales as s1

left join sales as s2

on s1.cust = s2.cust and s1.prod != s2.prod and s1.mo = s2.mo

group by s1.cust, s1.prod, s1.mo

)

select s1.cust, s1.prod,s1.mo, avg_q as "THE_AVG", other_mo_avg "OTHER_MOS_AVG",

other_prod_avg "OTHER_PROD_AVG"

from Q1 as s1

inner join Q2 on s1.cust=Q2.cust and s1.prod= Q2.prod and s1.mo=Q2.mo

inner join Q3 on s1.cust=Q3.cust and s1.prod= Q3.prod and s1.mo=Q3.mo

Question 4

10 / 10 pts

For each product, find:

1. the median quantity
2. avg sales quantity for the first half of the sales quantities, excluding the median quantity
3. avg sales quantity for the second half of the sales quantities, excluding the median quantity

For example, given the following sales transactions for Bread...

1. the median quantity is 3
2. the avg sales quantity for 1st half is (1+1+1+2+2)/5 = 1.4
3. the avg sales quantity for 2nd half is (4+5+6+7+8)/5 = 6.0

sales (prod, cust, yr, mo, day, state, quant)

PRODUCT	QUANT
Bread	1
Bread	1
Bread	1
Bread	2
Bread	2
Bread	3
Bread	4
Bread	5
Bread	6

```
Bread        7
Bread        8


PROD   MEDIAN  AVG_1ST_HF  AVG_2ND_HF
=====  ======  ==========  ==========
Bread       3         1.4         6.0
. . .
```

Your Answer:

```
with Q0 as (
select prod, quant
from sales
),
Q1 as (
select prod, count(quant) as count_q
from sales
group by prod),
Q2 as (
select prod,quant,
(select count(s2.quant)
from sales s2
where s1.prod=s2.prod and s1.quant>=s2.quant ) as order
from sales s1
group by prod, quant
),
median as (
select Q2.prod , Q2.quant
from Q2, Q1
where Q1.prod = Q2.prod and Q2.order = Q1.count_q/2+1
group by Q2.prod, Q2.quant),
AVG_1ST as(
select Q0.prod, round(avg(Q0.quant))
from Q0,median
```

where Q0.prod=median.prod and Q0.quant<median.quant

group by Q0.prod),

AVG_2ND as(

select Q0.prod, round(avg(Q0.quant))

from Q0,median

where Q0.prod=median.prod and Q0.quant>median.quant

group by Q0.prod)

select m.prod, m.quant "MEDIAN", L.round "AVG_1ST_HF", R.round "AVG_2ND_HF"

from median m

left join AVG_1ST L on m.prod = L.prod

left join AVG_2ND R on m.prod = R.prod

Question 5

5 / 5 pts

As discussed in class, “the design” process (for relational tables) has 2 main steps. List what they are and provide a brief description of each step in your own words:

Your Answer:

1. To examine whether schema is good? If yes, do nothing.

A good table:

1) table should only contain one entity

2) all attributes should functionally dependent on super keys.

3) schema is in 3NF(or BCNF)

2. If not, decompose the table based on the closure of functional dependencies. And then, examine whether the child schemas are good, if not, split again. Repeat this process again until all child schemas are good.

3. Finally, join all child schemas, and then we can get the orginal table back

two steps

Question 6

53 / 55 pts

1. (3 pts.) Describe 3 ways of describing what a "**bad**" table is/looks like (based on our discussions of good tables).

2. (4 pts.) What does "Loss Less Join Decomposition" (LLJD) mean and how do we ensure that we don't have LLJD when doing a design? Explain **in your own words** (i.e., do NOT just repeat what you read from the textbook and/or slides say -- you will get '0' point if you simply repeat the definition from the text/slides).

3. (4 pts.) Sometimes, even a "good" table is decomposed. Why would that be necessary? And how would you decompose such a table (e.g., using which columns, etc.)?

4. Closure of Attribute Sets:

a. (6 pts.) Which 2 Armstrong's axioms are used in "Closure of Attribute Sets" algorithm, and describe how the algorithm utilizes the 2 Armstrong's axioms.

b. Describe how the "Closure of Attribute Sets" algorithm is used in the following -- describe in your own words:

1) (3 pts.) To test super key

2) (4 pts.) To test candidate key

3) (3 pts.) To test FD

5. (3 pts.) Given that R is a table that is in 3NF, do the following queries produce the same results? If so, why? Or if not, why not? And explain what the differences are. And how would you change the queries to produce the same results?

Q1: select * from R;
Q2: (select * from R) intersect (select * from R);

6. (5 pts.) Given R (A, B, C) and the following functional dependencies, which of the 3 left hand sides (AB, BC, AC) are super keys? Are they also candidate keys? Justify your answers.

AB -> C

BC -> A

AC -> B

7. (3 pts.) Is the 'sales' table (used for the course) in BCNF? First, answer YES or NO, and justify your answer. Identify some functional dependencies in the 'sales' table.

8. (3 pts.) Given the following,

R = (A, B, C)

F = {A -> B, B -> C}

. . . is it possible to have a decomposition as follows based on BCNF? Justify your answer.

R1 = (A, B), R2 = (A, C)

9. (4 pts.) Given the following,

R = (J, K, L)

F = {JK -> L

L -> K }

Find at least 2 candidate keys and briefly justify your answer (i.e., why they are candidate keys).

10. (10 pts.) Describe in detail how the "Closure of Attribute Sets" algorithm is used to produce F+ (closure of F, functional dependencies). Describe in English, without using mathematical symbols -- as discussed in class, pretend you're explaining the idea to a non-DB expert.

Your Answer:

1.

The concept of “bad tables” talking about here are specifically in OLTP environment:

(1). A bad table is table about more than one entity.

(2). If in one table all other columns don't structurally depend on key columns, the table is a bad one.

(3). If a schema R is not in 3NF or BCNF or even more strict form, the relation R is always in “bad form”.

2.

(1). LLJD requires that if a schema decomposes into two separate schemas, the intersect of the two decomposed schemas can be used to describe all functional dependencies of either the left-hand side or right-hand side schema. In another word, to perform LLJD, the intersect of the two decomposed schemas must contain the super key for either

the left-hand side or right-hand side schema. The LLJD can be either dependency preserving or not dependency preserving.

(2). If we join all child shemas and then find it is different from the orginal table. So, we can ensure that we don't have LLJD when doing a design

3.

Sometimes a good table is decomposed in order to set the functional dependencies of the table is in BCNF or 3NF. It is necessary because we want a table not only contently about one thing but also structurally or functionally depend on one thing.

To decompose such a table, we need to first test the set of functional dependencies of a relation schema is whether in BCNF or 3NF or not. If it is not satisfied with the BCNF or 3NF rules, we need to decompose the relation into two relations. The left-hand side relation should include the super key of the schema and an attribute that is not satisfied with the BCNF or 3NF rules. The right-hand side relation should include the rest of the schema except for the attribute within the left-hand side relation, plus the super key of the schema.

Then, we continue to test the functional dependencies of the right-hand side relation is whether in BCNF or 3NF or not. We repeat the process again and again until the functional dependencies of all right-hand side relations are satisfied with BCNF or 3NF rules.

4.

A:

Reflexivity and transitivity are used in the "Closure of Attribute Sets" algorithm.

The first step of the algorithm is to set the initial output equals to the given attributes. The idea is that let the given attributes determine themselves. It is called reflexivity in Armstrong's axioms.

The second step is to look for the given functional dependencies, if there are some other attributes that are determined by the given attributes, then include these attributes to the outputs. Then, looking for the given functional dependencies again to check if there are other attributes that are determined by newly included attributes, then include them in the outputs. It is called transitivity in Armstrong's axioms.

Then, repeat the process until there are no more attributes functionally dependent on the final output. Thus, the final output of the algorithm is the closure of attribute sets

B:

(1). When the process described above is finished, we take a look at the final output. If the final output contains all attributes or columns, which equals the entire schema of a relation, we call the final output is a super key.

(2). Based on the above process, once we determine that the final output is a super key, then we take a look at any subset of the final output. If any subset of the final out is already a super key, then the final output is not a super key. If no such subset is a super key, then we call the final output is the minimal super key, which is also called a candidate key.

(3). To check FD, just check if the target attribute or attributes are contained within the final output. If it contained, we called the target attributes have FD. If the final output does not contain, we called that the target attributes do not have FD.

5.

The following queries produce the same results.

Because the dataset returned by the INTERSECT statement will be the intersection of the data-sets of the two SELECT statements. In another words, Q2 returns R itself, which is equal to Q1.

6.

AB, BC and AC are all super keys; AB, BC and AC are all candidate keys also.

AB -> C , so (AB)+ = {ABC}

BC -> A , so (BC)+ = {ABC}

AC -> B , so (AC)+ = {ABC} so they are all super keys;

However, if we divide AB, BC, or AC

(A)+={A}, (B)+={B} and (C)+= {C}

A, B ,C are all not super keys, which mean AB, BC and AC are the minimal super keys therefore they are also candidate keys.

7.

The sales table is not in BCNF because some attributes are not functionally dependent on the super key.

For example, cust, prod, day, month, year, state determines quant; year, month determines day; year determines month; cust determines

prod.

8.

It is not possible to have a decomposition as follows based on BCNF. Since (A)+ ={ABC} and (B)+={BC} and (C)+={C}

A is the super key in R, However, B is not the super key. So, R is not in BCNF.

Based on BCNF, the decomposition should be R1=(B,C), R2=(A,B) (which means original schema minus right hand-side)

9.

JK is a candidate key because (JK)+ = {JKL}. So, JK is a super key. In addition, J or K individually is not a super key, so, JK is also a candidate key.
JL is also a candidate key because (JL)+ = {JKL}. So, JL is a super key. J or L individually is not a super key, so, JL is a candidate key.

10.

(1). Take schema R, and produce all possible combinations of attributes, if schema R is ABC, then find A,B,C, AB,AC,BC and ABC

(2). Then we compute the closure of attribute sets for all those combinations, the closure of A, the closure of B, ..., the closure of ABC.

(3). If the closure of A determines AB, then we can generate A determines A, A determines B, A determines AB. And we get all possible combinations of these right hand-side. Then we can derive all of these new FD

(4). For each combination, we compute all the corresponding new FD, then we can generate all possible dependencies based on the given set of FD

4.b.3. test the closure of the left hand side [-1]; 5. explanation is wrong [-1];

Quiz Score: **97** out of 100