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

Due May 10 at 9:10pm **Points** 100 **Questions** 6
Available May 10 at 6:40pm - May 10 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 10 at 9:10pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	128 minutes	83 out of 100

ⓘ Correct answers are hidden.

Score for this quiz: **83** out of 100
Submitted May 10 at 8:48pm
This attempt took 128 minutes.

Question 1

5 / 10 pts

For January through June (first 6 months), for each customer, product and month, compute running sum of 3 following months' sales quantities, including the current month. For example, for (Dan, Apple, 6), find sum of sales quantities from June, July and August.

The "YEAR" attribute is not considered for this query – for example, both January of 2007 and January of 2008 are considered January regardless of the year.

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

```
CUST      PROD      MO     RUNNING_3_MONTH_SUM
========  ========  ==     ===================
Dan       Apple      6                    1492
Helen     Eggs       3                     927
Claire    Banana     1                    4238
. . .
```

Your Answer:

```
(select prod, 1 as mo, sum(quant) RUNNING_3_MONTH_SUM
from sales
where month between 1 and 3
group by prod)
union
(select prod, 2 as mo, sum(quant) RUNNING_3_MONTH_SUM
from sales
where month between 2 and 4
group by prod)
union
(select prod, 3 as mo, sum(quant) RUNNING_3_MONTH_SUM
from sales
where month between 3 and 5
group by prod)
union
(select prod, 4 as mo, sum(quant) RUNNING_3_MONTH_SUM
from sales
where month between 4 and 6
group by prod)
union
(select prod, 5 as mo, sum(quant) RUNNING_3_MONTH_SUM
from sales
where month between 5 and 6
group by prod)
union
(select prod, 6 as mo, sum(quant) RUNNING_3_MONTH_SUM
from sales
where month = 6
group by prod)
order by mo
```

Did not include customer in combination

Question 2

10 / 10 pts

For customer and product, show the average sales before, during and after each quarter (e.g., for Quarter 2 (April, May & June), show average sales quantities for Quarter 1 (January, February & March) for BEFORE and average sales quantities for Quarter 3 (July, August & September) for AFTER. For “BEFORE” Quarter 1 and “AFTER” Quarter 4, display <NULL>.

The “YEAR” attribute is not considered for this query – for example, both January of 2007 and January of 2008 are considered January regardless of the year.

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

```
CUST      PROD      QTR     BEFORE_QTR_AVG   DURING_QTR_AVG   FOLLO
WING_QTR_AVG
========  ========  ===     ==============   ==============   =====
=============
Bloom     Cheese      1               NULL              298
125
Sam       Cherry      3                502              319
204
Joe       Apple       4                238              192
NULL
. . .
```

Your Answer:

```
with base as(
select cust, prod
from sales
group by cust, prod
),
q1 as(
select cust, prod, 1 as q1, round(avg(quant)) q1_avg
from sales
where month between 1 and 3
group by cust, prod
```

```
),
q2 as(
select cust, prod, 2 as q1, round(avg(quant)) q2_avg
from sales
where month between 4 and 6
group by cust, prod
),
q3 as(
select cust, prod, 3 as q1, round(avg(quant)) q3_avg
from sales
where month between 7 and 9
group by cust, prod
),
q4 as(
select cust, prod, 4 as q1, round(avg(quant)) q4_avg
from sales
where month between 10 and 12
group by cust, prod
),
final as(
select b.cust, b.prod, 1 as q1, null as BEFORE_AVG, q1_avg
DURING_AVG,
q2_avg AFTER_AVG
from base as b
left join q1 using(cust,prod)
left join q2 using(cust,prod)
union
select b.cust, b.prod, 2 as q1, q1_avg BEFORE_AVG, q2_avg
DURING_AVG,
q3_avg AFTER_AVG
from base as b
left join q2 using(cust,prod)
left join q1 using(cust,prod)
left join q3 using(cust,prod)
union
select b.cust, b.prod, 3 as q1, q2_avg BEFORE_AVG, q3_avg
```

```
DURING_AVG,
q4_avg AFTER_AVG
from base as b
left join q3 using(cust,prod)
left join q2 using(cust,prod)
left join q4 using(cust,prod)
union
select b.cust, b.prod, 4 as q1, q3_avg BEFORE_AVG, q4_avg
DURING_AVG,
null as AFTER_AVG
from base as b
left join q4 using(cust,prod)
left join q3 using(cust,prod)
)
select cust, prod, q1 as QTR, BEFORE_AVG as before_qtr_avg,
DURING_AVG as during_qtr_avg,
AFTER_AVG as following_qtr_avg
from final
order by q1
```

Question 3

10 / 10 pts

For each customer, product and month combination, compute

1. the product's average sale of this customer for the given month (i.e., the simple AVG for the group-by attributes–this is the easy part)
2. the average sale of the customer and month but for all of the other products
3. the customer's average sale for the given product, but for all of the other months.

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

CUST	PROD	MO	THE_AVG	OTHER_PROD_AVG	OTHER_MO_AVG
Vinny	Apple	1	42	928	29
Joe	Banana	12	183	762	132
. . .					

Your Answer:

```
with the_avg as(
select cust, prod, month, round(avg(quant)) THE_AVG
from sales
group by cust, prod, month
),
other_prod_avg as(
select q1.cust, q1.prod, q1.month, round(avg(quant))
OTHER_PROD_AVG
from the_avg as q1, sales s
where s.cust = q1.cust and s.month = q1.month and s.prod != q1.prod
group by q1.cust, q1.prod, q1.month
),
other_month_avg as(
select q1.cust, q1.prod, q1.month, round(avg(quant))
OTHER_MO_AVG
from the_avg as q1, sales s
where s.cust = q1.cust and s.prod = q1.prod and q1.month != s.month
group by q1.cust, q1.prod, q1.month
)
select * from the_avg as q1
natural full join other_prod_avg
natural full join other_month_avg
order by q1.cust, q1.prod, q1.month
order by q1.cust, q1.prod, q1.month
```

Question 4

7 / 10 pts

Using the "sales" table, compute the following:

For each product, find the smallest sales quantity that is greater than the smallest 30% of the quantities (of the product).

For example, given the following sales transactions for Bread, the smallest quant for Bread which is greater than the smallest 30% of the quantities is 21.

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

PRODUCT	QUANT
Bread	11
Bread	12
Bread	13
Bread	21
Bread	22
Bread	33
Bread	41
Bread	51
Bread	61
Bread	71

Your Answer:

```
with q1 as(
select prod, quant
from sales s
group by s.prod, s.quant
),
q2 as(
select q1.prod, q1.quant, sum(s.quant) CUM_QUANT
from q1, sales s
where s.prod = q1.prod and s.quant <= q1.quant
group by q1.prod, q1.quant
),
q3 as(
select prod, sum(quant) Q3_QUANT
from sales
group by prod
)
select q2.prod PRODUCT, min(q2.quant) as QUANT
from q2, q3
where q2.prod=q3.prod and q2.CUM_QUANT > 0.3 * q3.q3_quant
group by q2.prod
```

-Calculates 30% of the sum of quantity, not 30% of all quantities

Question 5

5 / 5 pts

Describe in your words what “functional dependency” is, and how it is used in table design.

Your Answer:

Functional dependency means relationship between sets of attribute or columns. It is a relationship that exists when one attribute uniquely determines another attribute i.e X->Y

- It is rule on legal relation/column.

- value of left hand side is always determined the value of right hand side.

FD is used to decompose given relation R into R1 and R2 and and form a good or bad table in database design.

Question 6

46 / 55 pts

1. (3 pts.) Describe the 3 ways of describing what a "good" table is/looks like, which I discussed at the beginning of Ch. 7.
2. (4 pts.) Is combining 2 tables (schemas) into a single table (schema) good or bad? Justify your answers.
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 (functional dependencies)

5. (3 pts.) Do the following queries produce the same results? If not,

what are the differences? And how would you change the queries to produce the same results?

Q1: select * from R;

Q2: (select * from R) intersect all (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 candidates 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

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 experts.

Your Answer:

Q1.

1. Table is about one thing.
2. Every attribute in the relation should be determined by a super key.
3. Table should be in 3NF or BCNF.

Q2.

It is not always a good option to combining tables.

- If you have same candidate key or primary key in both schemas then we can say that combining two table may create a good table.

- If you are combining two tables it may create unnecessary duplicates data which may lead to anomalies.

Q3.

- If we have a good table in BCNF but we can convert it to 3NF for dependency preserving, since it is not required to have dependencies preserved for a good table but with decomposition we can get a good table with all functional dependencies. So, if schema is dependency preserving then it won't need any costly join to perform on relation.

Q4.

(A). 2 Armstrong's axioms are used in "Closure of Attribute Sets" algorithm are **Reflexivity and Transitivity**.

Reflexivity - Closure of X+ determined {X} itself. (using reflexivity) and

Transitivity- (X->Y & Y->Z) then X -> Z thus result = {XYZ}

So, using this axioms LHS of functional dependency is in result then RHS should also be in the result.

(B).

1.Test Super Key(SK)- If the attribute is SK we'll need to find it's closure using Armstrong's axioms, check if it can contained all the attributes of relation R. EX- R(X,Y,Z), X+={X,Y,Z} i.e X is the SK.

2. To Test candidate Key(CK)- First find the SK, then if it contains any minimal then any minimal should not be a SK or whose proper subset is not a SK, then only it can be defined as CK. EX- if XY -> Z, it is super key and XY not determined R then XY is CK.

3. To test FD- The given dependency we'll check if the closure determinant contains dependent attribute as well then we can say that FD holds true. EX- R(X,Y,Z) , FD:{X->Y} then X+ = {XY} thus closure of X contains Y so FD holds true.

Q5.

Yes, both queries will produce the same result.

proof Q1- If R = (X, Y, Z), then select * from R = {XYZ}

Q2: {XYZ} intersectall {XYZ} thus Q1=Q2.

Q6.

1.

AB-> C: AB^+ = {A,B} , result = {A,B,C} (because AB->C)

so **AB^+ is a super key** because result = R

A^+={A}, B^+ = {B}, both not the minimal CK which proves **AB is Candidate Key.**

2.

BC-> A: BC^+ = {B,C} , result = {B,C,A} (because BC->A)

so **BC^+ is a super key** because result = R

B^+={B}, C^+ = {C}, both not the minimal CK which proves **BC is Candidate Key.**

3.

AC-> B: AC^+ = {A,C} , result = {A,C,B} (because AC->B)

so **AC^+ is a super key** because result = R

A^+={A}, C^+ = {C}, both not the minimal CK which proves **AC is Candidate Key**.

Q7.

Yes, There is no FD which does not have a super key on the left side.

It does not have any transitivity property like X->Y and Y->Z then X->Z so it is in 3NF.

All are atomic attributes.

Q8.

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

R1 intersect R2 = R1 and R1 intersect R2 = R2

R1 intersect R2 = {A} and A->{A,B,C}. Thus, A is a superkey.

In BCNF LHS must be a super key so this two relations are in BCNF form.

Q9.

- JK^+ = {JKL} It contains all the attribute of R so it is SK.

J^+={J}, K^+={K} so none of this have proper subset. so it is not a SK. so **JK^+ is one of the candidate key.**

and {J,K} are also prime attribute. So now check if J,K present in RHS then there should be more CK in given relation.

So (L->K) so replace K with L in first CK.

- JL^+ = {JLK} It contains all the attribute of R so it is SK.

J^+={J}, L^+={L} so none of this have proper subset. so it is not a SK. **so JL^+ is one of the candidate key.**

Thus, two candidate keys are JK and JL.

Q10.

The set of all attributes which can be functionally determined from an attribute set is called closure of attribute set. Add the attributes contained in the attribute set for which closure is being calculated to the result set. repeat steps to add more attributes to result set which can be functionally determined from the attributes already present in result set.

If the closure of attribute set contain all the attributes present in original relation R then that attribute set is called Super key.

Using super key we can find candidate key which is super keys whose proper subset is not a super key.

3)Needs to mention decomposing along super key 7) Sales table is not good because there is nothing to guarantee uniqueness (-3) 8) Did not fully answer question (-1) 10) The right idea but does not include all the steps (-3)

Quiz Score: **83** out of 100