Midterm Exam (CS 561-B) - Spring 2021 (Thu., 4/8/2021) 6:30-9 PM ET.

Due Apr 8 at 9:10pm **Points** 100 **Questions** 9
Available Apr 8 at 6:40pm - Apr 8 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 (30% 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 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!
- 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.
    - 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 assignment. 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 appropriate JOINS as needed.

This quiz was locked Apr 8 at 9:10pm.

Attempt History

	Attempt	Time	Score

	Attempt	Time	Score
LATEST	Attempt 1	150 minutes	76 out of 100

! Correct answers are hidden.

Score for this quiz: **76** out of 100
Submitted Apr 8 at 9:10pm
This attempt took 150 minutes.

Question 1

10 / 10 pts

Write an SQL query for the following:

For the (cust, prod) combinations, find the dates of most (MAX) and least (MIN) purchase quantities, and the corresponding states where the purchases were made.

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

cust	prod	max_date	max_state	min_date	min_state
Sam	Apple	2001-01-04	NJ	2002-05-23	NY
Boo	Cherry	2004-04-23	NY	2009-08-31	CT
. . .	. . .	. . .	. . .	. . .	. . .
Claire	Grapes	2002-07-31	CT	2005-04-23	NJ

Your Answer:

with Q1 as (

select cust, prod, sum(quant) as sum_q, yr,mo,day state

from sales

group by cust, prod, yr,mo,day),

Q2 as (

select cust, prod, max(sum_q) as max_q

```
from Q1

group by cust, prod),


max as (

select Q2.cust, Q2.prod, Q2.max_q, Q1.yr, Q1.mo,Q1.day, Q1.state

from Q2

inner join Q1 on Q1.cust = Q2.cust, Q1.prod = Q2.prod

group by Q2.cust, Q2.prod, Q2.max_q, Q1.yr, Q1.mo,Q1.day),


Q3 as (

select cust, prod, min(sum_q) as min_q

from Q1

group by cust, prod),


min as (

select Q3.cust, Q3.prod, Q3.min_q, Q1.yr, Q1.mo,Q1.day, Q1.state

from Q3

inner join Q1 on Q1.cust = Q3.cust, Q1.prod = Q3.prod

group by Q3.cust, Q3.prod, Q3.max_q, Q1.yr, Q1.mo,Q1.day),

select max.cust, max.prod, max.yr, max.mo, max.day, max.state,

min.yr, min.mo, min.day, min.state

from min

inner join max using(cust,prod);
```

for view max and min, you do not need group by

Question 2

10 / 10 pts

Write an SQL query for the following:

For each year, find the "busiest" and the "slowest" months (those months with the most and the least total sales quantities of products sold). For each of the "busiest" months, find the "maximum" sales quantity. Similarly, for each of the "slowest" months, find the "minimum" sales quantity.

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

year	busiest_month	max_quant	slowest_month	min_quant
2001	12	231,204	3	3,423
2005	7	9,082,023	1	15,320
. . .	. . .	. . .	. . .	. . .
2002	5	12,310,435	11	1,245

Your Answer:

```
with Q1 as (

select s.yr, s.mo,sum(quant) as sum_q

from sales as s

group by s.yr, s.mo

order by s.yr, s.mo),

max_table as (

select b.yr,  max(sum_q) as max_q

from Q1 as b

group by b.yr),


Q2 as (

select b.yr, Q1.mo,b.max_q

from max_table as b

inner join Q1 on b.max_q= Q1.sum_q
group by b.yr,Q1.mo,b.max_q),

min_table as(

select b.yr,min(sum_q) as min_q

from Q1 as b

group by b.yr),


Q3 as(
select b.yr,Q1.mo, b.min_q

from min_table as b

inner join Q1 on b.min_q = Q1.sum_q

group by b.yr,Q1.mo,b.min_q)


select Q2.yr "year", Q2.mo "busiest_month", Q2.max_q "max_quant",

Q3.mo "slowest_month", Q3.min_q "min_quant"

from Q2 inner join Q3 using(yr)

order by yr;
```

Question 3

10 / 10 pts

Write an SQL query for the following:

For each customer, find the year and month when the most amount of purchases (of all products) were made, and the year and month when the least amount of purchases (of all products) were made.

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

cust	max_year	max_month	min_year	min_month
Wally	2001	12	2008	5
Chae	2007	2	2006	4
. . .	. . .	. . .	. . .	. . .
Dan	2010	7	2009	10

Your Answer:

```
with Q1 as (
select cust, sum(quant) as amount, yr, mo
from sales
group by yr, mo),

max as (
select a.cust, max(s.amount), b.yr as max_year, b.mo as max_month
from Q1 as a
inner join Q1 as b on a.cust = b.cust
and a.amount = b.amount),

min as(
select a.cust, min(a.amount), b.yr as min_year, b.mo as min_month
from Q1 as a
inner join Q1 as b
on a.cust = b.cust
and a.amount = b.amount),

select a.cust, a.max_year, a.max_month, b.min_year,b.min_month
from max as a
full outer join min as b on a.cust = b.cust;
```

Question 4

10 / 10 pts

Write an SQL query for the following:

For each month (regardless of the year), find the most popular product purchased and the corresponding sales quantity. Similarly for each month, find the least popular product purchased and the corresponding sales quantity.

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

month	most_popular_prod	most_popular_quant	least_popular_pr
1	Apple	62,253	Cherry
11	Butter	198,324	Ice
. . .	. . .	. . .	. . .
4	Fish	23,459	Eggs

Your Answer:

with Q1 as (

select s.mo, s.prod, sum(quant) as sum_q

from sales as s

group by s.mo, s.prod

order by s.mo),

Q2 as (

select mo, max(sum_q) as max_q

from Q1

group by mo) ,

most_pop as (

select Q2.mo, Q1.prod,Q2.max_q

```
from Q2

inner join Q1 on Q1.mo =Q2.mo and Q1.sum_q=Q2.max_q),


Q3 as (

select mo, min(sum_q) as min_q

from Q1

group by mo) ,

least_pop as (

select Q3.mo, Q1.prod,Q3.min_q

from Q3

inner join Q1 on Q1.mo =Q3.mo and Q1.sum_q=Q3.min_q)


select s.mo "month", s.prod "most_popular_prod", s.max_q
"max_popular_quant",

l.prod "least_popular_prod", l.min_q "least_popular_quant"

from most_pop as s

inner join least_pop as l using(mo)

order by mo;
```

Question 5

1 / 10 pts

Write an SQL query for the following:

For each customer and year between 2001 and 2010, find the total "cumulative" sales quantities for 3 consecutive years -- i.e., for the current, previous, and the year before. For example, for ('Dan', 2001), find total sales quantities just for 2001. Similarly, for ('Dan', 2002), find the total sales quantities for 2002 and 2001. For the other years, e.g., ('Dan', 2003), find the total sales quantities for 2003, 2002 and 2001.

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

cust	year	cum_quantity
Claire	2001	3,214
Helen	2009	19,524
. . .	. . .	. . .
Emily	2005	21,920

Your Answer:

with Q1 as (

select s.cust, s.prod, s.mo, s.yr, sum(quant) as sum_q

from sales as s

group by s.cust, s.mo, s.prod

order by s.month),

this question has nothing to do with month

Question 6

10 / 10 pts

Rewrite the following query using the following syntax (2 separate queries -- one using WITH and the other using HAVING):

1. WITH
2. HAVING

```
select cust, prod, avg_quant

 from (

    select cust, prod, avg(quant) avg_quant

      from sales

    group by cust, prod

) as cust_prod_avg (cust, prod, avg_quant)

where avg_quant > 100
```

Your Answer:

1.

with a as

(select cust, prod, avg(quant) avg_quant

from sales

group by cust prod)

select cust, prod, avg_quant

from a

where avg_quant>100

2.

select cust, prod, avg(quant) as avg_quant

from sales

group by cust, prod

having avg_quant > 100;

Question 7

5 / 15 pts

Rewrite the following query using:

1. cartesian product & where clause
2. inner join & “on”, and
3. inner join & “using”:

```
select * from account natural join depositor na
tural join customer
```

Schemas for the 3 tables are as follows:

- account (account_number, branch_name, balance)
- customer (customer_name, customer_street, customer_city)
- depositor (customer_name, account_number)

Your Answer:

1.
select a.account_number, a.branch_name, a.balance,

c.customer_name, c.customer_street,
c.customer_city,d.customer_name, d.account_number
from account as a, customer as c, depositor as d
where c.customer_name=d_customer.name and a.account_number=
d.account_number

2.

1. select * is sufficient

Question 8

5 / 5 pts

Rewrite the following query using SET operations (e.g., union, intersect, except):

select distinct *customer_name*

from *borrower, loan*

where *borrower.loan_number* = *loan.loan_number*

and *branch_name* = 'Perryridge'

and (*branch_name, customer_name*) **not in**

(select *branch_name, customer_name*

from *depositor, account*

where *depositor.account_number* = *account.account_number*

and *branch_name* = *'Perryridge'*)

Your Answer:

(SELECT distinct customer_name

FROM borrower, loan

WHERE borrower.loan_number = loan.loan_number

AND branch_name = 'Perryridge')

EXCEPT

(SELECT customer_name

FROM depositor, account

WHERE depositor.account_number = account.account_number

AND branch = 'Perryridge');

Question 9

15 / 20 pts

Answer the following questions in your own words -- each question is worth 4 points:

1. Describe in your own words what “***atomic***” data means? And is the name of ‘Michael Jordan’ considered atomic? In either case of ‘YES’ or ‘NO’, justify your answer. And why is it important to have all of the data to be "atomic"?
2. What is the main difference between “***sets***” and “***multi-sets***”? And why is it necessary to have multi-sets (instead of sets)?
3. Define "***candidate key***" in your own words (i.e., do not say "a minimal super key", etc.). How would you test to see if a given set of attributes is a “***candidate key***” using SQL queries?
4. I discussed the logical equivalence between "***tables***" and "***queries***". Describe how they are equivalent in your own words.
5. Why is the ***NULL*** value problematic? What kinds of problems can it cause? Provide 2 examples.

Your Answer:

1) "Atomic" data means that the data value in attributes or columns need to be un-dividable depends on subjective option. In most cases, it means that values in the attribute can onlu contain a word.

I don't know whether ‘Michael Jordan’ is atomic, because it depends on business need(use case). For bank, whey will use full name to store the customer information . At that time, it is automic.

However, it is not atomic when we divide it into first name and last name.

2) multi_sets means the table can duplicate like SQL, however sets means that the table can not be duplicate.

because it may provide convenience for keeping original data during an operation process.

3) candidate key is a kind of attribute that if you remove any of these attributes, it is no longer a super key. The function is like an ID fro tuples, it is selected to uniquely identify tuples in a relation.

4) queries extract information from one or more tables without changing their forms to achieve database management goals. The result of queries should also in tables format.

5) The SQL **null** is the term used to represent a missing value. A field with a NULL value is the one that has been left blank during the record creation.

2. not the table can duplicate, but the rows can duplicate; 5. missing examples 3. (-2) you did not answer how you'd use SQL to check for a candidate key.

Quiz Score: **76** out of 100