CS327E Lecture 3 - Monday 2/1/2016

Reading Quiz:
Question 1: Ans = A
Question 2: Ans = A
Question 3: Ans = C
Question 4: Ans = D

Concept Questions:

1. Recall the retail store that keeps information abouts its products in a table called SKU_Data. How can we look up all the products that are sold by the camping department or climbing department?
    a. SELECT * FROM SKU_Data
    WHERE DEPARTMENT = 'Camping'
    OR 'Climbing'
    There's a syntax error in this statement

    b. SELECT * FROM SKU_Data
    WHERE DEPARTMENT IN
    ('Camping', 'Climbing')

    c. SELECT * FROM SKU_Data
    WHERE Department = 'Camping'
    OR Department = 'Climbing'

    d. All of the above

    e. Only B and C

2. We have extended the retail store schema to allow tracking the vendors who supply products to the store. We want to obtain a list of the vendors, but we are only interested in those who are in Austin. What SQL query can we use to retrieve all vendors that have a presence in Austin?
    a. select vendName
    from vendors
    where vendCity = 'AUSTIN'
    Iff vendCity was 'austin' this would not work

    b. select vendName

from vendors
where vendCity =’Austin’
If vendCity was ‘austin’ this would not work

c. select *
from vendors
where UPPER(vendCity) =
‘AUSTIN’
We are trying to get all the vendors that are in Austin, and we don’t know if there’s typos in the values. So we transform everything to uppercase, and thus compare it to ‘AUSTIN’.

d. Any of the above

e. Not enough information

3. Continuing with the same example database, we now want to see a list of all vendors who are **not** based in Austin. Which SQL query will give us the answer?
   a. select vendName
   from vendors where UPPER(vendCity)
   != ‘AUSTIN’
   This does not take care of adding nulls

   b. select vendName
   from vendors
   where UPPER(vendCity) <>
   ‘AUSTIN’
   This does not take care of adding nulls

   c. select vendName
   from vendors where
   UPPER(vendCity) <>
   ‘AUSTIN’ or vendCity is null
   We have to explicitly specify null, and thus we take care of selecting nulls also

   d. Any of the above

   e. None of the above

4. Suppose we have a pool of printers and a set of registered users who have been given access to a printer. We now want to allow a guest user who is not in the table to use one of the common printers. How can we come up with a table definition that lets us assign common printers to guest users without losing existing functionality?
    a. (printer_name,
       printer_description,
       printer_type, userid)

    b. (printer_name,
       printer_description,
       userid_start,
       userid_end)
       Review the slide for an explanation of this, it handles load balancing

    c. (printer_name,
       printer_description,
       registered_userid,
       guest_userid)

    d. None of the above

5. Suppose we have a database that tracks software bugs. What is the relationship between the Bugs entity and the other entities according to the conceptual diagram?
    a. Bugs has a many-to-one relationship with Accounts

    b. Bugs has a one-to-many relationship with Comments

    c. Bugs has a many-to-many relationship with Products

    d. Bug has a one-to-many relationship with BugsProducts

    e. All of the above

6. How can we find all the bugs that are both **unassigned** and **active?** Assume that the assigned_to field identifies if a bug has been assigned and an **active** bug equals status of not ‘CLOSED’.
    a. select * from Bugs
       where assigned_to IS NULL
       and (status <> ‘CLOSED’

or status IS NULL)
We need to handle the case where the bug is null also, indicating that it is not active.

b. select * from Bugs
where assigned_to IS NULL
and status <> 'CLOSED'
This does not handle if the bug's status is null, indicating that it is inactive.

c. select * from Bugs
where assigned_to = NULL
and (status <> 'CLOSED' or
status = NULL)
We cannot use the = sign when checking for null equality.

d. select * from Bugs
where assigned_to IS NULL
and status NOT IN
('CLOSED')
This does not handle the null case.

e. None of the above