SQL Data Definition Language (DDL) Questions

1. Consider the following relational schema:

Reader(RDNR, Surname, Firstname, City, Birthdate)
Book(ISBN, Title, Author, NoPages, PubYear, PublisherName)
Publisher(PublisherName, PublisherCity)
Category(CategoryName, BelongsTo)
Copy(ISBN, CopyNumber, Shelf, Position)
Loan(ReaderNr, ISBN, Copy, ReturnDate)
BookCategory(ISBN, CategoryName)

(a) Enforce that a reader can only borrow up to 20 books. Give a solution using CHECK.

Solution:

```
CREATE ASSERTION MAX_NO_BOOKCHECK (
NOT EXISTS (
SELECT COUNT(*) FROM Loan
GROUP BY readerId
HAVING COUNT(*) > 20 ) );
```

(b) Formulate in SQL the following modifications. Change the return date of all the books in the category "Databases" that should be returned before 15.03.2013 so that they can be kept for 30 days longer (Assume that you can add days to dates in SQL).

Solution:

```
UPDATE Loan
SET ReturnDate = ReturnDate + 30
WHERE ReturnDate< '2013-03-15'
AND ISBN IN
(SELECT ISBN
FROM BookCategory
WHERE CategoryName = 'Databases');
```

(c) Formulate in SQL the following modifications:

The reader “Andy Goh” borrows the copy with CopyNumber “4” of the book with ISBN “123456”.

Solution:

```
INSERT INTO Loan (ReaderNr, ISBN, Copy)
SELECT RDNR, 123456, 4
FROM Reader
WHERE Firstname = ’Andy’ AND Surname = ’Goh’;
```

2. Consider the following schema of a database used by a hospital to record information about patients and wards.

Wards (number, numbed)
Patients (pid, name, year, gender)
PatientInWard(pid, wardNumber)
Tests (pid, testDate, testHour, temperature, heartrate)

The attribute numbed is the number of beds in that ward. The name, year of birth, and gender ('M' or 'F') of each patient are stored in the Patients relation. The ward to which each patient is assigned is stored in the relation PatientInWard. During their stay in hospital, patients will undergo routine tests. The date and hour of each occasion when these tests are performed on a patient are recorded, and for each of these tests the patient's temperature and heart rate are measured and recorded in the database. A patient will normally undergo these routine tests several times during his stay in the hospital.

The hospital has a policy that all patients of age at least 60 years, and all female patients must be given a bed. Write an assertion to check the following constraint.

The number of patients in a ward who satisfy at least one of the following constraints:

(i) year of birth 1957 or earlier, or
(ii) a female patient,

cannot exceed the number of beds in that ward.

Solution:

```
CREATE ASSERTION NotOverFullWard CHECK
( NOT EXISTS(
        SELECT      number
        FROM        Wards, Patients,    PatientInWard
        WHERE         number = wardNumber
        AND           PatientInWard.pid = Patients.pid
        AND           (year <= 1957
        OR            gender = 'F')
        GROUP BY    number
        HAVING      numbed < COUNT(pid)
));
```

3. Consider a "drinker" database with the following relations.

Drinker (drinker, age, address)
Like (drinker, beer)
Beer (beer, manufacturer)
Bar (bar, owner, address)
Frequent (drinker, bar)
Sell (bar, beer, price)

Table Frequent indicates which specific bar a drinker likes to visit.

Using SQL, create a view UnLuckyDrinker (drinker, bar), to record those drinkers who cannotfind all the beers that he likes in a bar that he frequents.

Solution:

```
CREATE VIEW UnLuckyDrinker(drinker, bar) AS
 SELECT *
 FROM Frequent
 WHERE EXISTS
 (
  SELECT beer
  FROM Like
  WHERE Frequent.drinker=Like.drinker
  EXCEPT
  SELECT beer
  FROM Sell
  WHERE Frequent.bar=Sell.bar );
```

4. Consider the following relation:

customer (cust_code, cust_name, cust_country, opening_amt, receive_amt, outstanding_amt)

Consider a view "myclient" as given below.

```
CREATE VIEW myclient (client_name, client_no, outspercent)
AS SELECT cust_name, cust_code,outstanding_amt*100/(opening_amt+receive_amt)
FROM customer
WHERE cust_country='USA'
AND  outstanding_amt*100/(opening_amt+receive_amt)>50;
```

What will be the output of the following SQL query? Justify your answer.

```
UPDATE myclient
SET outspercent=80
WHERE client_no=1;
```

Solution:

This view is a not an updatable view. Arithmetic expression has been used in the definition of the view, thus it won't work.

5. The following schema relates to holiday chalets that government employees may rent:

EMPLOYEE (emp-id, name, category, salary, age)

CHALET (chalet-id, location, price-per-day)

RENTAL (emp-id, chalet-id, start-date, no-of-days)

Note: Examples of category are Division I, Division II, etc.

Consider the view defined over the above schema:

```
CREATE VIEW X-VIEW
AS SELECT AVG(price-per-day)
FROM CHALET C, RENTAL R,
WHERE C.chalet-id = R.chalet-id
AND no-of-days> 7
GROUP BY location;
```

Describe what this view shows and give three reasons why it is generally considered not updatable.

Solution:

The view shows the average rental by location of long-term rentals (i.e. more than seven days in duration).

The view is not updatable for the following three reasons:

- It is defined on multiple tables (CHALET and RENTAL). The primary keys of both these tables are not available through this view.
- It is defined using the grouping function (GROUPBY LOCATION). The individual tuples are not accessible as the result groups the tuples based on the attribute LOCATION.
- It is defined using the aggregate functions (AVG). Modifying 'average' price does not make any sense since this value is based on a function of individual prices in the tuples.

6. Consider the following schema containing bank account information.

Primary Keys are in bold.

CUSTOMERS(**customer_name**, address)

ACCOUNTS(**account_number**, balance)

ACCOUNT_OWNERS(**customer_name**, **account_number**)

Write an assertion such that for every customer at least one of the following conditions holds true:

- He (or she) is owner of at most 5 accounts
- The sum of the balance of various accounts he (or she) owns is greater than $50,000.

Solution:

```
CREATE ASSERTION mid-term-assert    CHECK
 (NOT EXISTS (SELECT O.customer-name
              FROM Accounts A, Account-Owners O
              WHERE A.account-number = O.account-number
              GROUP BY O.customer-name
              HAVING COUNT(A.account-number) > 5
                      AND SUM(A.balance) < 50000))
```

Alternative solution:

```
CREATE ASSERTION mid-term-assert-2    CHECK
(NOT EXISTS
    (SELECT * FROM Customers C
     WHERE (SELECT COUNT(*)
            FROM Account-Owners O
            WHERE O.customer-name = C.customer-name)>5
      AND (SELECT SUM(balance)
           FROM Accounts A, Account-Owners O2
           WHERE A.account-number= O2.account-number
           AND A.customer-name = C.customer-name)<50000)
```