Assignment Day4 –SQL: Comprehensive practice

Amber Han

Answer following questions

1. What is View? What are the benefits of using views?
   View is the result set of a stored query on the data, which the database users can query just as they would in a persistent database collection object.
   A view can limit the degree of exposure of the underlying tables to the outer world: a given user may have permission to query the view, while denied access to the rest of the base table.

2. Can data be modified through views?

   Data cannot be modified through views.

3. What is stored procedure and what are the benefits of using it?
   A stored procedure is a prepared SQL code that you can save, so the code can be reused over and over again. So if you have an SQL query that you write over and over again, sav it as a store procedure, and then just call it to execute it.

4. What is the difference between view and stored procedure?

   View is simple showcasing data stored in the database tables whereas a stored procedure is a group of statements that can be executed. A view is faster as it displays data from the tables referenced whereas a store procedure executes sql statements.

5. What is the difference between stored procedure and functions?
   A function has a return type and returns a value. A procedure does not have a return type. But it returns values using the OUT parameters.

6. Can stored procedure return multiple result sets?

   It can. Stored procedure contains IN and OUT parameters or both. They may return result sets in case you use SELECT statements.

7. Can stored procedure be executed as part of SELECT Statement? Why?

Stored procedures are typically executed with an EXEC statement. However, you can execute a stored procedure implicitly from within a SELECT statement, provided that the stored procedure returns a result set.

8. What is Trigger? What types of Triggers are there?

   A trigger is a special type of stored procedure in database that automatically invokes/runs/fires when an event occurs in the database server.
   DDL/DML/Logon Trigger.

9. What are the scenarios to use Triggers?
   A trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

10. What is the difference between Trigger and Stored Procedure?

    Stored procedures are a pieces of the code in written in PL/SQL to do some specific task. Stored procedures can be invoked explicitly by the user. It's like a java program , it can take some input as a parameter then can do some processing and can return values.

    On the other hand, trigger is a stored procedure that runs automatically when various events happen (eg update, insert, delete). Triggers are more like an event handler they run at the specific event. Trigger can not take input and they can't return values.

Write queries for following scenarios

Use Northwind database. All questions are based on assumptions described by the Database Diagram sent to you yesterday. When inserting, make up info if necessary. Write query for each step. Do not use IDE. BE CAREFUL WHEN DELETING DATA OR DROPPING TABLE.

1. Lock tables Region, Territories, EmployeeTerritories and Employees. Insert following information into the database. In case of an error, no changes should be made to DB.
   a. A new region called "Middle Earth";
      INSERT INTO Region VALUES(5, 'Middle Earth')
   b. A new territory called "Gondor", belongs to region "Middle Earth";
      INSERT INTO Territories VALUES(11111, 'Gondor', 5)
   c. A new employee "Aragorn King" who's territory is "Gondor".
      INSERT INTO Employees(EmployeeID, LastName, FirstName, Territory)VALUES(10, 'King', 'Aragorn', 'Gondor')

2. Change territory “Gondor” to “Arnor”.
   UPDATE Territories SET TerritoryDescription = ‘Arnor’ WHERE TerritoryID = 11111

3. Delete Region “Middle Earth”. (tip: remove referenced data first) (Caution: do not forget WHERE or you will delete everything.) In case of an error, no changes should be made to DB. Unlock the tables mentioned in question 1.
   DELETE FROM Territories WHERE TerritoryID = 11111 AND TerritoryDescription = ‘Arnor’ AND RegionID = 5

4. Create a view named “view_product_order_[your_last_name]”, list all products and total ordered quantity for that product.

   CREATE VIEW view_product_order_Han AS SELECT p.ProductName, Count(o.Quantity)
   QuantityCount FROM Products p INNER JOIN [Order Details] o
   ON o.ProductID = p.ProductID
   GROUP BY p.ProductName

5. Create a stored procedure “sp_product_order_quantity_[your_last_name]” that accept product id as an input and total quantities of order as output parameter.
   CREATE PROC sp_product_order_quantity_Han (@ product_id INT, @ total_quatity
   FLOAT OUTPUT) AS
   BEGIN
   SELECT @ product_id = p.productid
   FROM Products p
   INNER JOIN [Order Details] od
   ON p.ProductID = od.ProductID
   WHERE SUM(od.Quantity = @total_quantity)
   GROUP BY p.ProductID
   END

6. Create a stored procedure “sp_product_order_city_[your_last_name]” that accept product name as an input and top 5 cities that ordered most that product combined with the total quantity of that product ordered from that city as output.

   CREATE PROC sp_product_order_city_Han (@product_name VARCHAR(20),
   @order_city VARCHAR(20) OUTPUT)
   AS
   BEGIN
   SELECT @product_name = ss.productname FROM (SELECT top 5 d.ProductID,
   d.ProductName)

FROM (SELECT p.ProductID, p.ProductName, SUM(od.quantity) t FROM Products p
INNER JOIN [Order details] od ON p.ProductID = od.ProductID GROUP BY p.PRODUCTID, p.ProductName)AS [d] ORDER BY d.t DESC) ss LEFT JOIN(
SELECT * FROM (SELECT dd.productid, dd.city, RANK() OVER(PARTITION BY productid ORDER BY q DESC) [rk] FROM (SELECT p.ProductID, c.city, SUM(od.Quantity) q FROM Customers c INNER JOIN orders o ON c.CustomerID = o.CustomerID
LEFT JOIN [Order Details] od ON o.OrderID = od.OrderID
LEFT JOIN Products p ON od.ProductID = p.ProductID)
LEFT JOIN Products p ON od.ProductID = p.ProductID
GROUP BY p.ProductID, c.City)dd
)cc WHERE cc.rk = 1 x
ON ss.productid = x.productid
WHERE x.city = @order_city
END

7. Lock tables Region, Territories, EmployeeTerritories and Employees. Create a stored procedure "sp_move_employees_[your_last_name]" that automatically find all employees in territory "Tory"; if more than 0 found, insert a new territory "Stevens Point" of region "North" to the database, and then move those employees to "Stevens Point".
CREATE PROC sp_move_employees_Han
@territory_name CHAR(20) = 'tory'
AS
IF EXISTS(SELECT e.EmployeeID, COUNT(t.TerritoryDescription) c FROM Territory t
INNER JOIN employeeterritories et ON t.territoryID = et.TerritoryID
INNER JOIN Employees e ON et.EmployeeID = e.EmployeeID
WHERE TerritoryDescription = @territory_name AND COUNT(t.TerritoryDescription) > 0)
BEGIN
INSERT INTO Territories(territoryID, TerritoryDescription, RegionID) VALUES (98432, 'STEVEN Point', 11)
INSERT INTO Region(RegionID, Regiondescription) VALUES(11, 'North')
END

8. Create a trigger that when there are more than 100 employees in territory "Stevens Point", move them back to Troy. (After test your code,) remove the trigger. Move those employees back to "Troy", if any. Unlock the tables.

```
CREATE TRIGGER trg_Han ON territories
FOR UPDATE AS
```

```
IF EXISTS(SELECT e.employeeid, COUNT(t.TerritoryDescription) FROM Territories t
INNER JOIN employeeterritories et ON t.TerritoryID = et.TerritoryID
INNER JOIN Employees e ON et.EmployeeID = e.EmployeeID
WHERE t.TerritoryDescription = 'steven point'
GROUP BY e.EmployeeID
HAVING COUNT(t.TerritoryDescription) > 100 )
BEGIN
UPDATE Territories SET Territorydescription = 'Tory'
WHERE TerritoryDescription = 'steven point'
END
DROP TRIGGER trg_Han
```

9. Create 2 new tables "people_your_last_name" "city_your_last_name". City table has two records: {Id:1, City: Seattle}, {Id:2, City: Green Bay}. People has three records: {id:1, Name: Aaron Rodgers, City: 2}, {id:2, Name: Russell Wilson, City:1}, {Id: 3, Name: Jody Nelson, City:2}. Remove city of Seattle. If there was anyone from Seattle, put them into a new city "Madison". Create a view "Packers_your_name" lists all people from Green Bay. If any error occurred, no changes should be made to DB. (after test) Drop both tables and view.

```
CREATE TABLE people_han (id INT, name VARCHAR(20), cityid INT)
CREATE TABLE city_han (cityid INT, city VARCHAR(20))
INSERT INTO people_han(id, name, cityid) VALUES(1, 'Aaron Rodgers', 2)
INSERT INTO people_han(id, name, cityid) VALUES(2, 'Russell Wilson', 1)
INSERT INTO people_han(id, name, cityid) VALUES(3, 'Jody Nelson', 2)
INSERT INTO city_han(cityid, city) VALUES(1, 'Seattle')
INSERT INTO city_han(cityid, city) VALUES(2, 'Green Bay')
CREATE VIEW Packers_Yi_Han AS
SELECT * FROM peolpe_han p INNER JOIN city_han c ON p.cityid = c.cityid WHERE
c.city = 'Green Bay'
BEGIN TRAN
ROLLBACK
DROP TABLE people_han
DROP TABLE city_han
DROP VIEW Packers_Yi_Han
```

10. Create a stored procedure "sp_birthday_employees_[you_last_name]" that creates a new table "birthday_employees_your_last_name" and fill it with all employees that have a birthday on Feb. (Make a screen shot) drop the table. Employee table should not be affected.

```
SELECT employeeid, LastName, FirstName, Title, TitleOfCourtesy, BirthDay,
Hiredate, Photo INTO birthday_employees_han FROM employees
WHERE MONTH(BirthDate) = 2
END
DROP TABLE birthday_employees_han
```

11. Create a stored procedure named "sp_your_last_name_1" that returns all cites that have at least 2 customers who have bought no or only one kind of product. Create a stored procedure named "sp_your_last_name_2" that returns the same but using a

different approach. (sub-query and no-sub-query).

```
CREATE PROC sp_han_1 AS
SELECT c.city, COUNT(c.CoustomerID) FROM customers c
INNER JOIN (
SELECT x.CustomerID, COUNT(x.CustomerID) xx FROM (SELECT DISTINCT c.CustomerID,
p.ProductID FROM Products p
INNER JOIN [Order Details] od ON p.ProductID = od.ProductID
INNER JOIN Orders o ON od.OrderID = o.OrderID
INNER JOIN Customers c ON o.CustomerID = c.CustomerID) x
GROUP BY x.CustomerID
HAVING COUNT(x.Customer) < 2 s
ON c.CustomerID = s.CUstomerIDGROUP BY city
HAVING COUNT(c.CustomerID) > 1
)
```

12. How do you make sure two tables have the same data?

Store the total data number for both table and do the union, and check id the number of data for 2 table is not the same number of the union data, then they do not have the same data.

14.

First Name	Last Name	Middle Name
John	Green	
Mike	White	M

Output should be

Full Name
John Green
Mike White M.

Note: There is a dot after M when you output.

```
DECLARE @fullname VARCHAR(20)
SELECT @fullname = firstneme + lastname + middlename + '.' FROM people
PRINT @fullname
SELECT firstname + '.' + lastname + middlename + '.' AS fullname FROM table1
```

15.

Student	Marks	Sex
Ci	70	F

Bob	80	M
Li	90	F
Mi	95	M

Find the top marks of Female students.

If there are to students have the max score, only output one.

```
DECLARE @student VARCHAR(20)
DECLARE @marks INT
SET @student
SET @marks = (SELECT MAX(marks) FROM student WHERE sex = 'F')
PRINT @student
```

16.

Student	Marks	Sex
Li	90	F
Ci	70	F
Mi	95	M
Bob	80	M

How do you out put this?

```
DECLARE @student VARCHAR(20)
DECLARE @marks INT
SET @student
SET @marks = (SELECT MAX(marks) FROM student ORDER BY sex)
PRINT @student
```

GOOD LUCK.