Assignment Day4 –SQL: Comprehensive practice

# Answer following questions

1. What is View? What are the benefits of using views?  
   - View is a virtual table based on the result-set of an SQL statement. Complex and reusable queries can be retrieved using view.
2. Can data be modified through views?  
   - Yes data can be modified using views but it is not recommended when view using more than one base table.
3. What is stored procedure and what are the benefits of using it?  
   - Store procedure is a collection of DML,DDL statements that can be executed together. Store procedures are helpful in maintaining clean scripts and easy testable and isolates business rules etc.
4. What is the difference between view and stored procedure?  
   - View has just select statements but store procedure has collection of DML and DDL statements.
5. What is the difference between stored procedure and functions?  
   - Store procedure can return any number of values or may not return any value but function must return a value. We can use transaction in SP but it is not possible in functions. We can have both input and output parameters in sp but we can have only input parameters in function.
6. Can stored procedure return multiple result sets?  
   - Yes, stored procedure can return multiple result sets.
7. Can stored procedure be executed as part of SELECT Statement? Why?  
   - No. Because store proc may or may not return a value.
8. What is Trigger? What types of Triggers are there?  
   - Trigger in SQL server is used for business logics to be executed. SQL server has after trigger and instead of trigger for insert, update and delete statements.
9. What are the scenarios to use Triggers?  
   - We can prevent creation of duplicate records. To create logs and so on.
10. What is the difference between Trigger and Stored Procedure?  
    - Triggers happen on DML statements occurrence whereas store procs should be executed manually.

# 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.
   1. A new region called “Middle Earth”;  
        
      - BEGIN TRAN  
      select \* from Region  
      select \* from Territories  
      select \* from Employees  
      select \* from EmployeeTerritories  
        
      INSERT INTO Region VALUES(6,'Middel Earth')

IF @@ERROR <>0

ROLLBACK  
ELSE BEGIN

* 1. A new territory called “Gondor”, belongs to region “Middle Earth”;  
       
     INSERT INTO Territories VALUES(98105,'Gondor',6)  
     DECLARE @error INT = @@ERROR  
     IF @error <>0
* BEGIN  
  PRINT @error  
  ROLLBACK  
  END  
  ELSE BEGIN  
  1. A new employee “Aragorn King” who's territory is “Gondor”.  
       
     - INSERT INTO Employees VALUES('Aragorn', 'King' ,'Sales Representative', 'Ms.' ,'1966-01-27 00:00:00.000','1994-11-15 00:00:00.000', 'Houndstooth Rd.', 'London', NULL ,'WG2 7LT', 'UK', '(71) 555-4444' ,452,NULL, 'Anne has a BA degree in English from St. Lawrence College. She is fluent in French and German.', 5, 'http://accweb/emmployees/davolio.bmp/')

INSERT INTO EmployeeTerritories VALUES(@@IDENTITY,98105)

DECLARE @error2 INT = @@ERROR

IF @error2 <>0

BEGIN

PRINT @error2

ROLLBACK

END

ELSE BEGIN

1. Change territory “Gondor” to “Arnor”.  
     
    - UPDATE Territories

SET TerritoryDescription = 'Arnor'

WHERE TerritoryDescription = 'Gondor'

IF @@ERROR<>0

ROLLBACK

ELSE BEGIN

1. 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 EmployeeTerritories

WHERE TerritoryID = (SELECT TerritoryID FROM Territories WHERE TerritoryDescription = 'Arnor')

DELETE FROM Territories

WHERE TerritoryDescription = 'Arnor'

DELETE FROM Region

WHERE RegionDescription = 'Middel Earth'

IF @@ERROR <>0

ROLLBACK

ELSE BEGIN

COMMIT

END

END

END

END

END

1. 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\_Gaddam

AS

SELECT ProductName,SUM(Quantity) As TotalOrderQty FROM [Order Details] OD JOIN Products P ON P.ProductID = OD.ProductID

GROUP BY ProductName

1. 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.

ALTER PROC sp\_Product\_Order\_Quantity\_Gaddam

@ProductID INT,

@TotalOrderQty INT OUT

AS

BEGIN

SELECT @TotalOrderQty = SUM(Quantity) FROM [Order Details] OD JOIN Products P ON P.ProductID = OD.ProductID

WHERE P.ProductID = @ProductID

GROUP BY ProductName

END

DECLARE @Tot INT

EXEC sp\_Product\_Order\_Quantity\_Gaddam 11,@Tot OUT

PRINT @Tot

1. 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.

ALTER PROC sp\_Product\_Order\_City\_Gaddam

@ProductName NVARCHAR(50)

AS

BEGIN

SELECT TOP 5 ShipCity,SUM(Quantity) FROM [Order Details] OD JOIN Products P ON P.ProductID = OD.ProductID JOIN Orders O ON O.OrderID = OD.OrderID

WHERE ProductName=@ProductName

GROUP BY ProductName,ShipCity

ORDER BY SUM(Quantity) DESC

END   
EXEC sp\_Product\_Order\_City\_Gaddam 'Queso Cabrales'

1. 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”.

BEGIN TRAN

select \* from Region

select \* from Territories

select \* from Employees

select \* from EmployeeTerritories

GO

ALTER PROC sp\_move\_employees\_gaddam

AS

BEGIN

IF EXISTS(SELECT EmployeeID FROM EmployeeTerritories WHERE TerritoryID = (SELECT TerritoryID FROM Territories WHERE TerritoryDescription ='Troy'))

BEGIN

DECLARE @TerritotyID INT

SELECT @TerritotyID = MAX(TerritoryID) FROM Territories

BEGIN TRAN

INSERT INTO Territories VALUES(@TerritotyID+1 ,'Stevens Point',3)

UPDATE EmployeeTerritories

SET TerritoryID = @TerritotyID+1

WHERE EmployeeID IN (SELECT EmployeeID FROM EmployeeTerritories WHERE TerritoryID = (SELECT TerritoryID FROM Territories WHERE TerritoryDescription ='Troy'))

IF @@ERROR <> 0

BEGIN

ROLLBACK

END

ELSE

COMMIT

END

END

EXEC sp\_move\_employees\_gaddam

1. 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 tr\_move\_emp\_gaddam

ON EmployeeTerritories

AFTER INSERT

AS

DECLARE @EmpCount INT

SELECT @EmpCount = COUNT(\*) FROM EmployeeTerritories WHERE TerritoryID = (SELECT TerritoryID FROM Territories WHERE TerritoryDescription = 'Stevens Point' AND RegionID=3) GROUP BY EmployeeID

IF (@EmpCount>100)

BEGIN

UPDATE EmployeeTerritories

SET TerritoryID = (SELECT TerritoryID FROM Territories WHERE TerritoryDescription ='Troy')

WHERE EmployeeID IN (SELECT EmployeeID FROM EmployeeTerritories WHERE TerritoryID = (SELECT TerritoryID FROM Territories WHERE TerritoryDescription ='Stevens Point' AND RegionID=3))

END

DROP TRIGGER tr\_move\_emp\_gaddam

COMMIT

1. 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\_Gaddam

(

id int ,

name nvarchar(100),

city int

)

create table City\_Gaddam

(

id int,

city nvarchar(100)

)

BEGIN TRAN

insert into City\_Gaddam values(1,'Seattle')

insert into City\_Gaddam values(2,'Green Bay')

insert into People\_Gaddam values(1,'Aaron Rodgers',1)

insert into People\_Gaddam values(2,'Russell Wilson',2)

insert into People\_Gaddam values(3,'Jody Nelson',2)

if exists(select id from People\_Gaddam where city = (select id from City\_Gaddam where city = 'Seatle'))

begin

insert into City\_Gaddam values(3,'Madison')

update People\_Gaddam

set city = 'Madison'

where id in (select id from People\_Gaddam where city = (select id from City\_Gaddam where city = 'Seatle'))

end

delete from City\_Gaddam where city = 'Seattle'

CREATE VIEW Packers\_Gaddam

AS

SELECT name FROM People\_Gaddam WHERE city = 'Green Bay'

select \* from Packers\_Gaddam

commit

drop table People\_Gaddam

drop table City\_Gaddam

drop view Packers\_Gaddam

1. 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.

ALTER PROC sp\_birthday\_employee\_gaddam

AS

BEGIN

SELECT \* INTO #EmployeeTemp

FROM Employees WHERE DATEPART(MM,BirthDate) = 02

SELECT \* FROM #EmployeeTemp

END

1. 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\_gaddam\_1

AS

BEGIN

SELECT City FROM CUSTOMERS

GROUP BY City

HAVING COUNT(\*)>2

INTERSECT

SELECT City FROM Customers C JOIN Orders O ON O.CustomerID=C.CustomerID JOIN [Order Details] OD ON O.OrderID = OD.OrderID

GROUP BY OD.ProductID,C.CustomerID,City

HAVING COUNT(\*) BETWEEN 0 AND 1

END

GO

EXEC sp\_gaddam\_1

GO

CREATE PROC sp\_gaddam\_2

AS

BEGIN

SELECT City FROM CUSTOMERS

WHERE CITY IN (SELECT City FROM Customers C JOIN Orders O ON O.CustomerID=C.CustomerID JOIN [Order Details] OD ON O.OrderID = OD.OrderID

GROUP BY OD.ProductID,C.CustomerID,City

HAVING COUNT(\*) BETWEEN 0 AND 1)

GROUP BY City

HAVING COUNT(\*)>2

END

GO

EXEC sp\_gaddam\_2

GO

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

SELECT \* FROM Customers

EXCEPT

SELECT \* FROM Customers

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.

SELECT firstName+' '+lastName from Person where middleName is null UNION SELECT firstName+' '+lastName+' '+middelName+'.' from Person where middleName is not null

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.  
  
- select top 1 marks from student where sex = 'F' order by marks desc

16.

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

How do you out put this?

* select \* from students order by sex,marks