Assignment Day4 –SQL: Comprehensive practice

# Answer following questions

1. What is View? What are the benefits of using views?
2. A view is a virtual table that is generated from a query and stored as a permanent object. Using a view can be helpful in limiting the degree of exposure a user has to the base tables.
3. Can data be modified through views?
4. Yes.
5. What is stored procedure and what are the benefits of using it?
6. Stored procedures are a type of code in SQL that can be stored for later use and can be used many times. The benefit of using stored procedures is that it has better performance, and it is easy to maintain.
7. What is the difference between view and stored procedure?
8. A view is a simple showcasing data stored in the database table whereas a stored procedure is a group of statements that can be executed.
9. What is the difference between stored procedure and functions?
10. Procedures can have output parameters as well as having return statement is optional in procedures, while a function must return a value and cannot have output variables.
11. Can stored procedure return multiple result sets?
12. Yes
13. Can stored procedure be executed as part of SELECT Statement? Why?
14. No, stored procedure cannot be executed as a part of SELECT statement because a stored procedure may return multiple resultsets which may cause problems.
15. What is Trigger? What types of Triggers are there?
16. Trigger is a special type of stored procedures that is automatically executed when an event occurs in a specific database server. Different types of triggers are: DML Triggers and DDL Triggers.
17. What are the scenarios to use Triggers?
18. Triggers can be used to enforce referential integrity or to derive additional data.
19. What is the difference between Trigger and Stored Procedure?
20. A trigger is a stored procedure which runs automatically when some event occurs, and it cannot take input values or return values like stored procedures.

# 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”;
   2. A new territory called “Gondor”, belongs to region “Middle Earth”;
   3. A new employee “Aragorn King” who's territory is “Gondor”.
2. begin tran

Insert into Region values (5,'Middle Earth')

Insert into Territories values (1212,'Gondor',5)

Insert into Employees values (10,'King','Aragorn')

Insert into EmployeeTerritories values (10,1212)

1. Change territory “Gondor” to “Arnor”.
2. Update Territories set TerritoryDescription = 'Arnor' where TerritoryDescription='Gondor'
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.

begin tran

Insert into Region values (5,'Middle Earth')

Insert into Territories values (1212,'Gondor',5)

Insert into Employees (LastName,FirstName) values ('King','Aragorn')

Insert into EmployeeTerritories values (10,1212)

Update Territories set TerritoryDescription = 'Arnor' where TerritoryDescription='Gondor'

Delete from EmployeeTerritories where EmployeeID = 10

Delete from Employees where EmployeeID = 10

Delete from Territories where RegionID = 5

Delete from Region where RegionID = 5

commit

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\_subhedar as select p.ProductID as "ProductID",sum(od.Quantity) as "Quantity" from Products p inner join [Order Details] od on od.ProductID = p.ProductID group by p.ProductID;

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.

create proc sp\_product\_order\_quantity\_subhedar

@product\_id int,

@qty int out

as

select @qty = Quantity from view\_product\_order\_subhedar where productID = @product\_id

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.

create proc sp\_product\_order\_city\_subhedar

@product\_name varchar(20)

as

select dt.city,dt.TotalQuantity from (select c.city,sum(od.quantity) as "TotalQuantity",dense\_rank() over(order by sum(od.quantity) desc ) as "rnk" from orders o

inner join [order details] od on o.orderId = od.orderID

inner join customers c on c.customerID = o.customerID where od.productid=(select p.productid from products p where p.productname=@product\_name) group by c.city)

dt where dt.rnk <=5

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”.

create proc sp\_move\_employees\_subhedar

@records int out

as

select @records = count(1) from employees e where e.employeeID in (select et.EmployeeID from EmployeeTerritories et inner join territories t on t.territoryID = et.territoryID and t.TerritoryDescription = 'Tory');

begin tran

declare @op int;

exec sp\_move\_employees\_subhedar @op out;

if @op<>0

begin

insert into territories values (1515,'Stevens Point',3);

update EmployeeTerritories set territoryid = 1515 where territoryId = (select t.territoryID from territories t where t.TerritoryDescription = 'Tory')

end

commit

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 testTrigger on EmployeeTerritories after insert

as

begin

declare @emp\_count int

select @emp\_count =count(et.employeeid) from EmployeeTerritories et where et.territoryID = (select t.territoryid from territories t where t.territoryDescription='Stevens Point')

if @emp\_count>100

update EmployeeTerritories set territoryid = (select t.territoryID from territories t where t.TerritoryDescription = 'Tory') where territoryId = (select t.territoryID from territories t where t.TerritoryDescription = 'Stevens Point')

end

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 view Packers\_chaitanya

as

select p.name from people\_subhedar p inner join city\_subhedar c on c.id = p.city where c.city='Green Bay'

begin tran

insert into city\_subhedar values (3,'Madison')

update people\_subhedar set city = 3 where city = (select c.id from city\_subhedar c where c.city='seattle')

delete from city\_subhedar where city='seattle'

select \* from Packers\_chaitanya

commit

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.

create proc sp\_birthday\_employees\_subhedar

as

begin

create table birthday\_employees\_subhedar (EmployeeId int,LastName varchar(20),FirstName varchar(20))

insert into birthday\_employees\_subhedar select EmployeeID,LastName,FirstName from employees where month(BirthDate) = 2

select \* from birthday\_employees\_subhedar

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\_subhedar\_1

as

select City from CUSTOMERS

group by City

having COUNT(customerid)>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(productid) BETWEEN 0 AND 1

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

If except of table returns 0 rows.

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 tb where MiddleName is null union select FirstName+ ' '+LastName+' '+MiddleName+'.' from tb 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 from students 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 group by sex,marks,student

GOOD LUCK.