Assignment Day3 –SQL: Comprehensive practice

Answer following questions

1. In SQL Server, assuming you can find the result by using both joins and subqueries, which one would you prefer to use and why?

   I would choose join. Join is shorter, which makes it much easier to read and maintain. Subqueries, like correlated subquery, will lower the performance. And in some scenarios, using subqueries following a from clause would give errors, because it can't guarantee to return a single value for from clause.

2. What is CTE and when to use it?
   CTE means common table expression. It is mainly used to create a recursive query. It can also be a substitute for a view and defined in user-defined routines like functions and triggers. Using cte can improve readability and lower the cost of maintenance.

3. What are Table Variables? What is their scope and where are they created in SQL Server?
   A table variable is similar to a table except that it has a strictly defined lifetime scope and foreign key constraints can't be used on the table variable.

   The scope is within a Transact-SQL batch, stored procedure, or function. Table variables are created inside the Transact-SQL batch, stored procedure, or function, which means we can't use the table variables outside these blocks.

4. What is the difference between DELETE and TRUNCATE? Which one will have better performance and why?

   Truncate will regenerate identity values, while delete does not. Truncate can remove all records without firing triggers. Truncate uses transaction log less which guarantees a faster performance. Truncate is not possible when a table is referenced by a Foreign Key or tables are used in replication or with indexed views. Delete is a DML command, truncate is a DDL command. Truncate can't be rolled back but delete can.

5. What is Identity column? How does DELETE and TRUNCATE affect it?

   Identity column is a special column whose value is created by the server and will increase automatically. Generally user can't insert a value into it. Identity column is used to identify the rows in the table.

   Truncate resets the identity value to the original value, while delete dose not affect the identity value so it keeps increasing.

6. What is difference between "delete from table_name" and "truncate table table_name"?

They all remove all records in the table. Truncate command resets the identity value to the original seed value, uses less log space, performs faster than delete command. It fails if this table is referenced by a Foreign Key. Delete command keeps the identity value, logs entry for every row in the transaction log, performs slower. Only delete command can be rolled back.

Write queries for following scenarios

All scenarios are based on Database NORTHWND.

1. List all cities that have both Employees and Customers.

```
select distinct c.City
from Customers c
where c.City in (select e.City from Employees e)
```

2. List all cities that have Customers but no Employee.
    a. Use sub-query

```
select distinct c.City
from Customers c
where not c.City in (select e.City from Employees e)
```

    b. Do not use sub-query

```
select distinct c.City from Customers c
left join Employees e
on c.City = e.City
where e.City is null
```

3. List all products and their total order quantities throughout all orders.

```
select (select p.ProductName from Products p where p.ProductID = od.ProductID)
Product_Name,
od.ProductID, sum(od.Quantity) Product_Quantity
from [Order Details] od
group by od.ProductID
```

4. List all Customer Cities and total products ordered by that city.

```
;with cte_customer_quantity as (select od.Quantity, (select o.CustomerID from
Orders o where o.OrderID = od.OrderID) cid from [Order Details] od)
select c.City, sum(cte.Quantity) from cte_customer_quantity cte
right join
Customers c
on c.CustomerID = cte.cid
group by c.City
order by c.city
```

5. List all Customer Cities that have at least two customers.

   a. Use union

```
select c.City from Customers c group by c.City having COUNT(c.City) = 2
union
select c1.City from Customers c1 group by c1.City having COUNT(c1.City) > 2
```

   b. Use sub-query and no union

```
select distinct c.City from Customers c
where c.City in
(select c1.City from Customers c1 group by c1.City having COUNT(c1.City) >= 2)
```

6. List all Customer Cities that have ordered at least two different kinds of products.

```
select c.City from Customers c
join (select o.CustomerID, od.ProductID from Orders o
join [Order Details] od
on o.OrderID = od.OrderID) t1
on c.CustomerID = t1.CustomerID
group by c.City
having count(distinct t1.ProductID) >= 2
```

7. List all Customers who have ordered products, but have the 'ship city' on the order different from their own customer cities.

```
select distinct o.CustomerID from Orders o
where o.ShipCity not in (select c.City from Customers c where o.CustomerID =
c.CustomerID)
```

8. List 5 most popular products, their average price, and the customer city that ordered most quantity of it.

```
;with product_sum as
(select od.ProductID,
(select p.ProductName from Products p where od.ProductID = p.ProductID)
product_name,
sum(od.Quantity) sum_qty,
avg(od.UnitPrice) avg_price
from [Order Details] od
group by od.ProductID),

product_popularity as
(select table2.pid, table2.qty, table2.shipcity,
rank() over(partition by table2.pid order by table2.qty desc) rank
from (select od.ProductID pid,o.ShipCity shipcity, sum(od.Quantity) qty
from Orders o
join [Order Details] od
on od.OrderID = o.OrderID
group by  od.ProductID, o.ShipCity) table2)

select top 5 table1.product_name,table1.ProductID, table1.avg_price,
table3.shipcity
from product_sum table1, product_popularity table3
where table1.ProductID = table3.pid and table3.rank = 1
order by table1.sum_qty desc
```

9. List all cities that have never ordered something but we have employees there.

    a. Use sub-query

```
select e.City from Employees e
where not e.City in (select o.ShipCity from Orders o)
```

    b. Do not use sub-query

```
select e.City from Employees e
left join Orders o
on e.City = o.ShipCity
where o.ShipCity is null
```

10. List one city, if exists, that is the city from where the employee sold most orders (not the product quantity) is, and also the city of most total quantity of products ordered from. (tip: join sub-query)

```
select e_city.City from
(select emp_sum.City, rank() over(order by emp_sum.emp_qty desc) emp_rank
from (select e.City, COUNT(o.OrderID) emp_qty from Employees e join Orders o on
e.EmployeeID = o.EmployeeID group by e.City) emp_sum
) e_city
join
(select city_sum.ShipCity, rank() over(order by city_sum.city_qty desc)
city_rank from
(select o1.ShipCity, sum(od.Quantity) city_qty from Orders o1 join [Order
Details] od on od.OrderID = o1.OrderID group by o1.ShipCity) city_sum) o_city
on e_city.City = o_city.ShipCity
where e_city.emp_rank = 1 and o_city.city_rank = 1
```

11. How do you remove the duplicates record of a table?

    Imagine ‘mytable’ has duplicate records and a field named ‘name’.

```
With cte_delete_duplicate as
(select name, row_number()
over(partition by name order by name ) rownumber  from mytable)
delete from cte_delete_duplicate where rownumber != 1
```

12. Sample table to be used for solutions below- Employee (empid integer, mgrid integer, deptid integer, salary integer) Dept (deptid integer, deptname text)
    Find employees who do not manage anybody.

```
select emp.empid from Employee emp where emp.empid not in (select emp2.mgrid
from Employee emp2)
```

13. Find departments that have maximum number of employees. (solution should consider scenario having more than 1 departments that have maximum number of employees). Result should only have - deptname, count of employees sorted by deptname.

```
;with emp_sum as (
select deptid did, count(distinct empid) e_qty
from Employees group by deptid
)
select table.did from (
select emp_sum.did, rank() over(order by emp_sum.e_qty desc) d_rank
from emp_sum
) table
where table.d_rank = 1
```

14. Find top 3 employees (salary based) in every department. Result should have deptname, empid, salary sorted by deptname and then employee with high to low salary.

```
;with sal_sum as (select empid eid, deptid did, salary sal, rank()
over(partition by deptid order by salary desc) e_rank from Employee)
select d.deptname, sal_sum.eid EmployeeId, sal_sum.sal EmployeeSalary from
sal_sum join dept d on d.deptid = sal_sum.did
where sal_sum.e_rank <= 3 order by d.deptname, sal_sum.e_rank
```

GOOD LUCK.