Assignment Day3 –SQL: Comprehensive practice

Amber Han

Answer following questions

CTE Benefits:

- CTE improves the code readability.
- CTE provides recursive programming.
- CTE makes code maintainability easier.

CTE VS VIEW:

Views can be indexed but CTE cannot. Consider views when dealing with complex queries. Views being a physical object on database (but does not store data physically) and can be used on multiple queries, thus provide flexibility and centralized approach. CTE, on the other hand are temporary and will be created when they are used;

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?

   It depends on the result I want. Subqueries are used to return a scalar (single) value or a row set, whereas, joins are used to return rows.

2. What is CTE and when to use it?

   A Common Table Expression, also called as CTE in short form, is a temporary named result set that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement. You can use CTE when you need to refer/join the same data set multiple times.

3. What are Table Variables? What is their scope and where are they created in SQL Server?

   Table variables are local variables that have some similarities to temp tables. Table variables are created via a declaration statement like other local variables. The table variable scope is within the batch. We can define a table variable inside a stored procedure and function as well. In this case, the table variable scope is within the stored procedure and function. We cannot use it outside the scope of the batch, stored procedure or function.

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

Truncate removes all records and doesn't fire triggers. Truncate is faster compared to delete as it makes less use of the transaction log. Truncate is not possible when a table is referenced by a foreign key or tables are used in replication or with indexed views. Truncate is faster performance than delete.

5. What is Identity column? How does DELETE and TRUNCATE affect it?
   An identity column is a column in a database table that is made up of values generated by the database.
   Delete does not reset identity value but keep on increasing.
   Truncate resets the identity value to the original seed value of the table.

6. What is difference between "delete from table_name" and "truncate table table_name"?
   Both of them delete all rows from table. But truncate cannot rollback the changes.

Write queries for following scenarios

All scenarios are based on Database NORTHWND.

1. List all cities that have both Employees and Customers.
```
SELECT DISTINCT city FROM Customers WHERE city IN (SELECT city FROM Employees)
```

2. List all cities that have Customers but no Employee.
   a. Use sub-query
   b. Do not use sub-query
```
/* 2a */
SELECT DISTINCT City FROM Customers WHERE City NOT IN (SELECT City FROM
Employees)
GO
/*2b*/
SELECT DISTINCT c.City FROM Customers c
LEFT JOIN Employees e
ON c.City = e.City
```

3. List all products and their total order quantities throughout all orders.
```
SELECT c.CustomerID, c.CompanyName, c.ContactName, SUM(od.quantity) AS Quantity
FROM  Customers c
LEFT JOIN Orders o
ON c.CustomerID = o.CustomerID
LEFT JOIN
[Order Details] od
ON o.OrderID = od.OrderID
GROUP BY c.CustomerID, c.CompanyName, c.ContactName
ORDER BY Quantity DESC
```

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

```
SELECT c.City, SUM(od.Quantity) AS Quantity
FROM Customers c
LEFT JOIN
Orders o
ON c.CustomerID = o.CustomerID
LEFT JOIN
[Order Details] od
ON o.OrderID = od.OrderID
GROUP BY c.City
ORDER BY Quantity DESC
```

5. List all Customer Cities that have at least two customers.
   a. Use union
   b. Use sub-query and no union

```
/* 5a Use Union*/
SELECT c.City FROM Customers c GROUP BY c.City HAVING COUNT(c.City) > 2
UNION
SELECT u.City FROM Customers u GROUP BY u.City HAVING COUNT(u.City) = 2
GO
/* 5b sub-query*/
SELECT DISTINCT c.City
FROM Customers c
WHERE c.City IN (SELECT u.City FROM Customers u GROUP BY u.City HAVING
COUNT(u.City) >= 2 )
```

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

```
SELECT DISTINCT c.City
FROM Orders o INNER JOIN Customers c
ON o.CustomerID = c.CustomerID
INNER JOIN [Order Details] od
ON od.OrderID = o.OrderID
GROUP BY c.City, od.ProductID
HAVING COUNT(od.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 * FROM Customers c
WHERE c.City NOT IN (SELECT o.ShipCity FROM Orders o INNER JOIN Customers c ON
o.ShipCity = c.City)
```

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

```
WITH orderCTE
AS
(
SELECT oc.ShipCity, oc.ProductID, oc.average, DENSE_RANK() OVER (PARTITION BY
oc.ProductID Order BY
oc.number) rnk FROM (
SELECT TOP(5) od.ProductID, o.ShipCity, SUM(Quantity) number, AVG(od.UnitPrice)
average FROM
dbo.Orders o LEFT JOIN dbo.[Order Details] od ON o.OrderID = od.OrderID
GROUP BY o.ShipCity, od.ProductID
ORDER BY number DESC
)oc
)
SELECT * FROM orderCTE WHERE rnk = 1
```

9. List all cities that have never ordered something but we have employees there.
    a. Use sub-query
    b. Do not use sub-query

```
/* 9a use sub-query*/
SELECT e.City FROM Employees e
WHERE e.City NOT IN(
SELECT c.City FROM Orders o INNER JOIN Customers c
On c.CustomerID = o.CustomerID)
GO
/* 9b not use sub-query*/
SELECT DISTINCT e.City FROM Employees e
LEFT JOIN Customers c
ON e.City = c.City
WHERE c.City 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 * FROM
        (SELECT TOP 1 e.City, COUNT(o.OrderID) countOrder FROM Employees e INNER JOIN
Orders o
        ON e.EmployeeID = o.EmployeeID
        GROUP BY e.City)T1
        INNER JOIN(
        SELECT Top 1 c.City, COUNT(r.Quantity) countQuantity FROM [Order Details] r INNER
JOIN Orders d ON r.OrderID = d.OrderID
        INNER JOIN Customers c ON c.CustomerID = d.CustomerID GROUP BY c.City)T2
        ON T1.City = T2.city
```

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

Find Duplicate rows using GROUP BY clause or ROW NUMBER() function. Use DELETE statement to remove the duplicate rows.

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 deptname, salary
FROM(
SELECT d.deptname, e.empid, e.salary, RANK() OVER(PARTITION BY e.deptid ORDER BY
e.salary DESC)AS rnk
FROM dept d, employee e
WHERE d.deptid = e.deptid
)
WHERE rnk <= 3
ORDER BY deptname, rnk
```

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.

```
SELECT countbydept.*
FROM(
SELECT deptid, COUNT(*) AS departCount
FROM employee
GROUP BY deptid
ORDER BY departCount DESC
LIMIT 1
)AS maxcount
INNER JOIN
(SELECT dept.id, dept.'name', COUNT(*) AS employeeCount
FROM Dept
INNER JOIN
Employee ON Employee.deptid = deptid
GROUP BY deptid, deptname)
countbydept
```

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.

```
SELECT deptname, empid, salary
FROM(
SELECT d.deptname, e.empid, e.salary, RANK() OVER ( PARTITION BY e.deptid ORDER BY
e.salary DESC)AS rnk
FROM dept d, employee e
WHERE d.deptid = e.deptid
)
WHERE rnk <= 3
ORDER BY deptname, rnk
```

GOOD LUCK.