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 prefer joins, because it is more efficient and easier to read when the queries become more complex.

1. What is CTE and when to use it?

CTE is called Common Table Expressions, which simplify various classes of SQL Queries for which a derived table was just unsuitable. It can be used as a reference within a select statement.

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

Table variables are a special type of local variable that helps to store data temporarily. Table variables are scoped to the stored procedure, batch, or function. They are created in the tempdb database.

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

TRUNCATE reset the identity to its seed value, whereas DELETE doesn't. Also, TRUNCATE always removes all the rows from a table, whereas DELETE may remove conditionally if the where clause is used. TRUNCATE is faster compared to delete as it makes less use of the transaction log.

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

Identity column of a table is a column that is used to automatically generate key values based on a provided seed and increment. TRUNCATE reset the identity to its seed value, whereas DELETE doesn't.

1. What is difference between “delete from table\_name” and “truncate table table\_name”?

DELETE statement deletes records one by one and makes an entry for each and every deletion in the transaction log, whereas TRUNCATE statement deallocates pages and makes an entry for deallocation of pages in the transaction log. Also, TRUNCATE reset the identity to its seed value, whereas DELETE doesn't.

# 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, e.city

FROM customers c, Employees e

WHERE e.city = c.City

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

SELECT DISTINCT City FROM customers c WHERE NOT EXISTS

(SELECT City FROM Employees e WHERE c.city = e.city)

* 1. Do not use sub-query

SELECT DISTINCT c.city

FROM customers c LEFT JOIN Employees e

ON e.city = c.City

WHERE e.city is NULL

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

SELECT p.ProductName,

(SELECT COUNT(od.OrderID)

FROM [Order Details] od

WHERE od.ProductID = p.ProductID)

AS "totalorder" FROM Products p

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

SELECT o.ShipCity, COUNT(o.OrderID) AS "totalorder"

FROM Orders o

GROUP BY o.ShipCity

1. List all Customer Cities that have at least two customers.
   1. Use union

SELECT City, COUNT(CustomerID)

FROM Customers

GROUP BY City

HAVING COUNT(CustomerID) >= 2

* 1. Use sub-query and no union

SELECT c.City,

(SELECT COUNT(o.CustomerID) FROM Orders o

WHERE o.CustomerID = c.CustomerID

HAVING COUNT(o.CustomerID) >= 2)

AS "totalcustomers" FROM Customers c

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

SELECT o.ShipCity, COUNT(od.ProductID) "countproducts"

FROM [Order Details] od INNER JOIN Orders o

ON od.OrderID = o.OrderID

GROUP BY o.ShipCity

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

SELECT distinct c.ContactName

FROM Customers c INNER JOIN

(SELECT o.CustomerID, o.ShipCity

FROM Orders o) dt

ON c.CustomerID = dt.CustomerID

WHERE c.City != dt.ShipCity

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

select top 5 p.ProductName, COUNT(od.ProductID) as productcount, AVG(od.UnitPrice\*od.Quantity\*od.Discount) as avgprice

from Products p join [Order Details] od

ON p.ProductID = od.ProductID

join Orders o

ON od.OrderID = o.OrderID

group by p.ProductName

order by productcount

desc

1. List all cities that have never ordered something but we have employees there.
   1. Use sub-query

select e.City

from Employees e left join

(select o.ShipCity

from Orders o) dt

ON e.City = dt.ShipCity

WHERE dt.ShipCity is NULL

* 1. 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

1. 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 top 1 dt2.City, dt2.totalorders, dt3.totalquantity from

(select e.City, dt.totalorders from Employees e left join

(select count(orderid) totalorders, EmployeeID from orders group by EmployeeID) dt

on e.EmployeeID = dt.EmployeeID) dt2

inner join

(select o.ShipCity, SUM(quantity) as totalquantity

from [Order Details] od join Orders o

on od.OrderID = o.OrderID

group by o.ShipCity) dt3

on dt2.City = dt3.ShipCity

order by totalorders

desc

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

Use the DISTINCT keyword in the SELECT statement.

1. 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 empid from Employee WHERE mgrid is NULL

1. 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 deptname, dt.countempl

FROM Dept d JOIN

(SELECT deptid, COUNT(empid) AS “countempl”

FROM Employee

GROUP BY deptid

ORDER BY countempl) dt

ON d.deptid = dt.deptid

1. 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, e.empid, e.salary from

(select e.empid, e.deptid, e.salary,

dense\_rank() over(partition by e.deptid order by e.salary desc) rnk from Employee e

inner join

Dept d

ON d.deptid = e.deptid) dt

where dt.rnk <=3

order by deptname, e.salary

GOOD LUCK.