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 prefer to use join the subquery because it is more efficient and better performance.

1. What is CTE and when to use it?

CTE is common table expression and it is temporary name of result set. It references the result table multiple times in the same statement.

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

Table variables are local variable and created in batch that is only declared. It helps to store data temporarily.

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

DELETE removes rows one at a time and TRUNCATE removes all the rows from a table.

TRUNCATE is faster and fewer resources to be executed because it always locks the table and pages not each row compared DELETE.

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

Identity column is an integer or bignit column which values are automatically generated from a system-defined sequence.

After DELETE statement is executed to remove the record, but it doesn’t reset the identity value and manually set a new identity value for the identity column.

After TRUNCATE statement is executed to remove the record, when we insert a new record in a table, we can see the identity value again from 1 as defined in the table properties.

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

TRUNCATE always remove all rows in a table, DELET removes rows conditionally if the where clause is used.

# Write queries for following scenarios

All scenarios are based on Database NORTHWND.

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

SELECT City

FROM Customers

WHERE city IN(

SELECT DISTINCT city FROM Employees

)

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

SELECT City

FROM Customers

WHERE city not IN(

SELECT city FROM Employees

)

* 1. Do not use sub-query

SELECT city

FROM customers

EXCEPT

SELECT DISTINCT e.city

FROM Employees E INNER JOIN Customers C

ON E.city = C.city

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

SELECT p.productname, COUNT(od.OrderId) AS NumOfOrders

FROM Products p LEFT JOIN [Order Details] od ON p.ProductID = od.ProductID

GROUP BY p.ProductName

ORDER BY NumOfOrders DESC

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

SELECT c.City, COUNT(o.OrderId) AS NumOfOrders

FROM customers c LEFT JOIN orders o ON o.CustomerID=o.CustomerID

GROUP BY c.city

ORDER BY NumOfOrders DESC

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

select c1.City

from Customers c1

group by c1.City

having COUNT(c1.City) > 2

union

select c2.City

from Customers c2

group by c2.City

having COUNT(c2.City) = 2

* 1. Use sub-query and no union

select distinct c1.City

from Customers c1

where c1.City in

(select c2.City

from Customers

c2 group by c2.City

having COUNT(c2.City) >=2 )

1. 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

1. 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)

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

WITH cteorder

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 orders o left join [Order Details] od on o.orderID = od.orderID

GROUP BY o.shipCity, od.productID

ORDER BY number DESC

) oc

)

select \* from cteorder where rnk=1

1. List all cities that have never ordered something but we have employees there.
   1. 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)

* 1. Do 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

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 \* 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(od.Quantity) countQuantity

from [Order Details] od inner join Orders o on od.OrderID = o.OrderID

inner join Customers c on c.CustomerID = o.CustomerID

group by c.City) T2

on T1.City = T2.City;

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

I used distinct.

12. Sample table to be used for solutions below- Employee (empid integer, mgrid integer, deptid integer, salary money) Dept (deptid integer, deptname varchar(20))

Find employees who do not manage anybody.

select empid

from Employyees

where depid not in (select deptid from dept where deptid is not null)

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 deptcount

from Employee

group by deptid

order by deptcount desc) as maxcount

inner join

(select dept.id, dept.`name`, count(\*) as employeecount

from dept

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.