/*

PROG 140 **Module 8 Assignment**

POINTS 50 DUE DATE : Consult module

Angelique Refahiyat

06/10/2018

Develop a SQL statement for each task listed. This exercise uses the Northwind database.

Please type your SQL statement under each task below. Ensure your statement works by testing it prior to turning it in

When writing answers for your select statements please keep in mind that there is more than one way to write a query.

Please do your best to write a query that not only returns the information the questions asks for but to the best of your ability,

in the best way possible to suit their needs. For example, consider the best way to sort, to name columns, etc. These are things you must think about on the job!

Turn In:

For this exercise you will submit one WORD documents (instead of a .sql file) in which you have copied and pasted

your entire work from your .sql file including all assignment questions and your SQL queries.

The document should contain your name at the top, assignment title, the date and any difficulties encountered.

Submit your file to the instructor using the Canvas Assignment tool.

*/

/*

12 pts

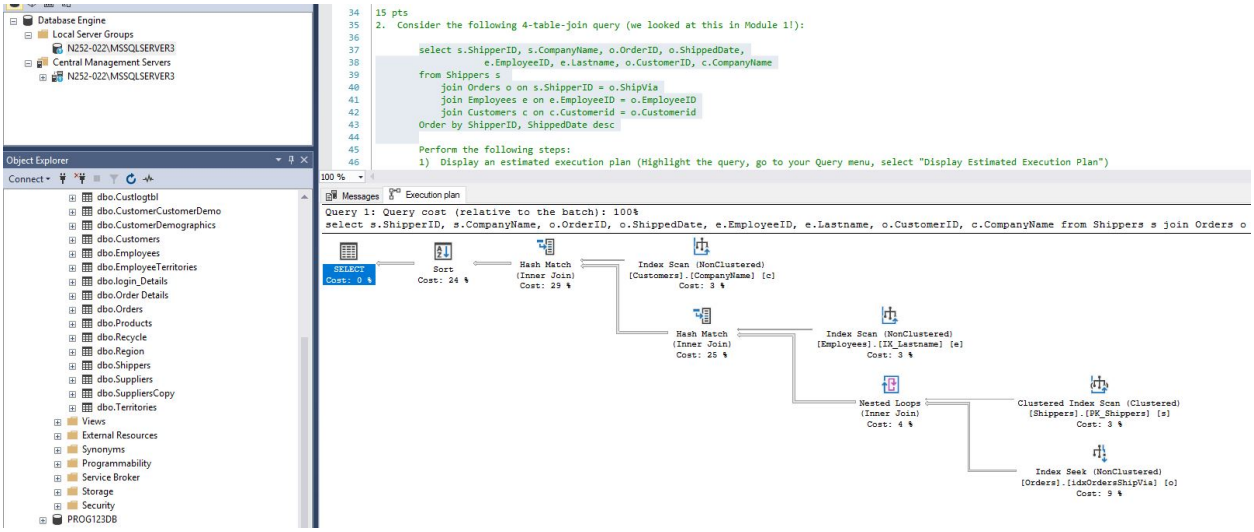
1. From our video (and PPT) on Performance, list 3 QUERY Performance tips and give query examples and explanations

of how/why they will help performance:

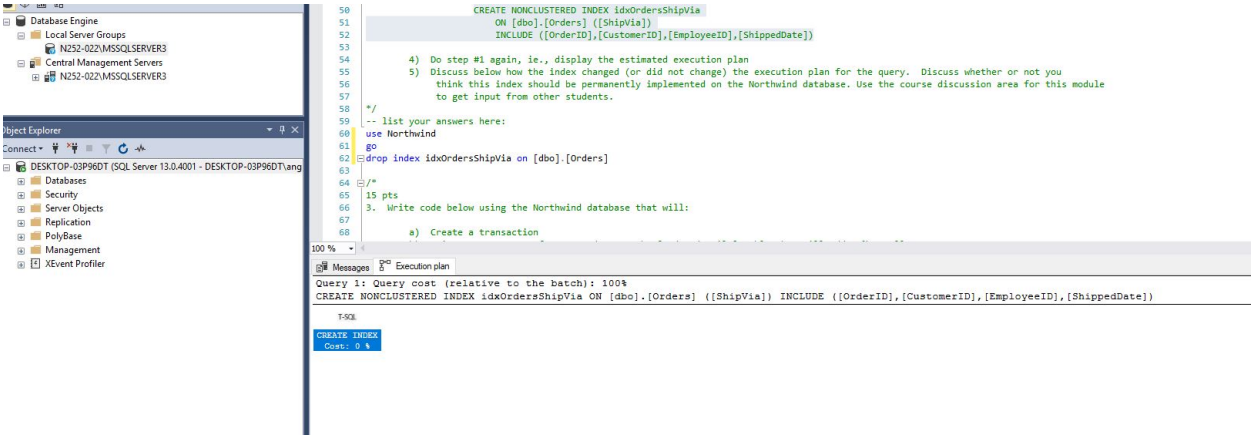
*/

-- list your answers here:

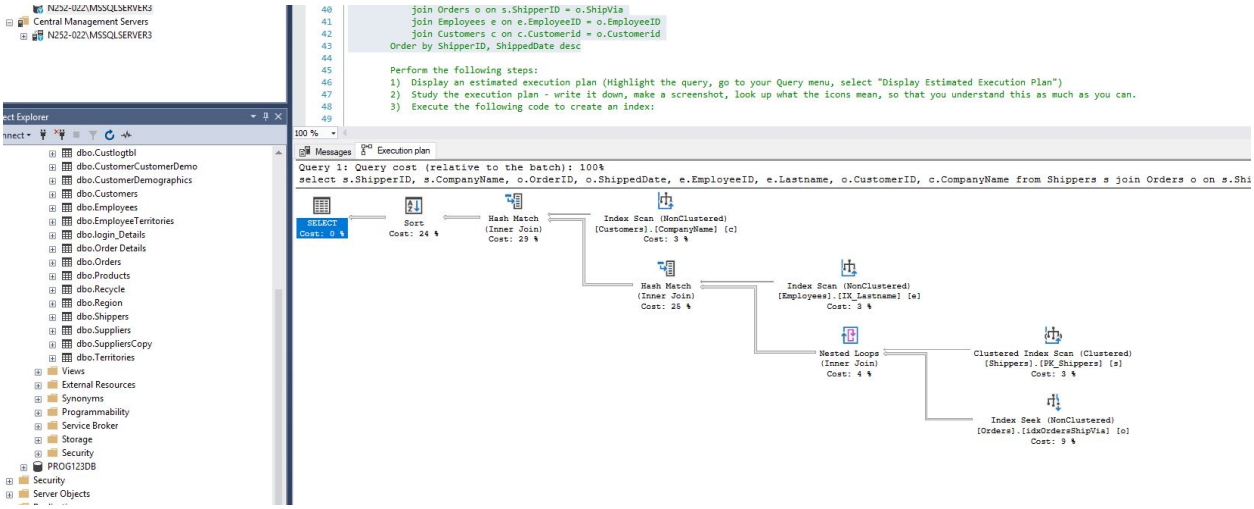
```
/*
15 pts
2.  Consider the following 4-table-join query (we looked at this in Module 1!):

                select s.ShipperID, s.CompanyName, o.OrderID, o.ShippedDate,
                                        e.EmployeeID, e.Lastname, o.CustomerID, c.CompanyName
                from Shippers s
                        join Orders o on s.ShipperID = o.ShipVia
                        join Employees e on e.EmployeeID = o.EmployeeID
                        join Customers c on c.Customerid = o.Customerid
                Order by ShipperID, ShippedDate desc

                Perform the following steps:
                1)  Display an estimated execution plan (Highlight the query, go to your Query menu, select
"Display Estimated Execution Plan")
                2)  Study the execution plan - write it down, make a screenshot, look up what the icons mean,
so that you understand this as much as you can.
                3)  Execute the following code to create an index:

                                        CREATE NONCLUSTERED INDEX idxOrdersShipVia
                                                ON [dbo].[Orders] ([ShipVia])
                                                INCLUDE
([OrderID],[CustomerID],[EmployeeID],[ShippedDate])

                4)  Do step #1 again, ie., display the estimated execution plan
                5)  Discuss below how the index changed (or did not change) the execution plan for the query.
Discuss whether or not you
                    think this index should be permanently implemented on the Northwind database. Use the
course discussion area for this module
                    to get input from other students.
*/
-- list your answers here:
use Northwind
go
drop index idxOrdersShipVia on [dbo].[Orders]
```



--Creating the index itself execution plan:



_Running the query after creating the index:



--Once I compared both Estimated Execution Plans from when the index were not exist with when I created an index, I saw there is no differenceence

--between them and the running time show the same as before and after creating an index. Means creating an index using to includ the columns doesn't have
-- any affects on the running and executing the join table.
--Each icon appearing on the screen represents an object to show how cuch the running time for each step in executing a query will take.
--Step one would be the reading part od an staement starting with the select statement, as it showed it did happen immidately and then moved on to
--the next step that would be the sorting part to see how long does it take to sort out the entire join in asecending or descending order.
--The next part would be scan all the indexes if there was any including clustered and non-clustered to verify if there was any of them exists.
--Mouving on to the next step which was going through the join to make the relationship within each table through reading the join statement.
--After creating the join statement system goes to implement the clustered or non-clustered index to start applying them in order.
--For the next step system went through a loop that made a connection within the tables through the join statement to pull out the data required.
--After searching through the loop for each table's primary keys then the system go head and confirm the clustered index for the one of the tables
--which is in this case is the Shipper table.
--For the last and the final step, the sytem also searched to see if there was any non-clustered index demanded from the query to apply it and scan it as what it done
-- from the previous step.--
--As it showed on both of my running time execution plans, there is no differences appeared to show whether ther was a running time
--differences that'll appects on running our query. In other word, creating this kind of index with include statement, did not affect on executing that specific query.
--and it did not accelarate the execution running time for the query.

/*
15 pts
3. Write code below using the Northwind database that will:

a) Create a transaction
b) Write a query to perform an update on the [order details] table that will add 10% to all unit prices.
c) Create a save point after the query in b)

d) Write a query to to perform an update on the Products table that will add $2.00 to all unit prices

e) Write the statement to rollback to the save point

f) Right under the statement in e), Write a statement that rolls back to the beginning of the Transaction.

*/

```
-- list your answers here:
use Northwind
go

UPDATE [dbo].[Order Details]
SET [UnitPrice]=[UnitPrice]+ 0.10
go
BEGIN TRANSACTION
select @@Trancount
SAVE TRANSACTION upd_untOd
--To get back at the saved point
ROLLBACK TRANSACTION upd_untOd
--To get back the very beginning of the transaction
COMMIT TRANSACTION upd_untPr
--Test it
select @@Trancount --numbers of transaction may vary depending on what method we used to delete the transaction
-- to the product table
UPDATE [dbo].[Products]
SET [UnitPrice]=[UnitPrice]+ 2.00
go
BEGIN TRANSACTION
select @@Trancount
SAVE TRANSACTION upd_untPr
ROLLBACK TRANSACTION upd_untPr
--check the number of transactions
select @@Trancount
COMMIT TRANSACTION upd_untPr
--checke the number of transaction dropped down one
select @@Trancount
/*
8 pts
```

4.
You have a colleague that is executing some work in a transaction on the
Customers table. She is running her transaction with an isolation level of Serializable.
You need to run a quick query for your boss that groups and counts all customers by their
Country. Write the query below that will execute (and not be blocked):
*/
-- list your answers here:
--Example of what she's done

```
use Northwind
go
select count(*) from [dbo].[Customers]--92
where [Country] ='FRANCE' --11
group by [Country]--22 rows affected

go
drop TRIGGER trgCustLog

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
DBCC USEROPTIONS--to verify if the system is sset up on serializable
select @@spid--54
SELECT * FROM sys.dm_tran_locks
where resource_type ='OBJECT' --0
DELETE FROM [dbo].[Customers]
where [CustomerID] ='RICHD'

Begin transaction

INSERT [dbo].[Customers] ([CustomerID],[CompanyName],[ContactName],[Country])
values('RICHD','ComapnySeller','Richard','FRANCE')


SAVE TRANSACTION upd_Custs
--End of her activities

--Check what she's done
SELECT * FROM sys.dm_tran_locks
where resource_type ='OBJECT' --1

ROLLBACK TRANSACTION upd_Custs --It did not get back to the beginning
```

```
--test to count the customers from the customer's table
select count(*) from [dbo].[Customers] --11 becomes 12 means one data added
where [Country] ='FRANCE'
group by [Country] --22 rows afftected

select * from sysObjects where ID = 693577509
select @@Trancount

COMMIT transaction upd_Custs
select count(*) from [dbo].[Customers]
where [Country] ='FRANCE'
group by [Country]

select @@Trancount  --dropped down one if the Commit statement used
SELECT * FROM sys.dm_tran_locks
where resource_type ='OBJECT' --0 again

EXECUTE sp_lock
select*from[dbo].[Customers]
WHERE CustomerID = 'RICHD'
--Make sure there is no blocked oject exist in the database
USE Master
GO
SELECT session_id, wait_duration_ms, wait_type, blocking_session_id
FROM sys.dm_os_waiting_tasks
WHERE blocking_session_id <> 0
GO --0

USE Master
GO
SELECT *
FROM sys.dm_exec_requests
WHERE blocking_session_id <> 0;
GO --0
use Northwind
go
select name, id from sysObjects where name = 'Customers' –To find the id for customer’s table
SELECT [CompanyName]FROM [dbo].[Customers] WITH (NOLOCK) --93 before commit and 92 after commit
transaction
```