Lab Title:

Transactions, Query Optimization, and Indexes

Submitted to:

Ma'am Darakhshan

Submitted by:

SABA

Course:

CS-363L Database Systems

Semester:

6th

Date:

10thApril, 2022

Department of Computer Engineering

University of Engineering and Technology, Lahore

Dummy Data

```
USE [BikeStores]
GO

--Insert 20000 rows in production.brands.
Declare @count int
SET @count = 1
While @count <= 20000
BEGIN
  INSERT INTO production.brands (brand_name) values ('brand - ' + CAST(@count as nvarchar(10)))
  SET @count = @count + 1
END

--Insert 20000 rows in production.categories.
Declare @count1 int
SET @count1 = 1
While @count1 <= 20000
BEGIN
  INSERT INTO production.categories (category_name) values ('category - ' + CAST(@count1 as
nvarchar(10)))
  SET @count1 = @count1 + 1
END

--Insert 20000 rows in production.products.
Declare @RandomBrandId int
Declare @RandomCategoryId int
Declare @RandomModelYear int
Declare @RandomListPrice int

Declare @LowerLimitForBrandId int
Declare @UpperLimitForBrandId int

Set @LowerLimitForBrandId = 1
Set @UpperLimitForBrandId = 20009

Declare @LowerLimitForCategoryId int
Declare @UpperLimitForCategoryId int

Set @LowerLimitForCategoryId = 1
Set @UpperLimitForCategoryId = 20007

Declare @LowerLimitForModelYear int
Declare @UpperLimitForModelYear int

Set @LowerLimitForModelYear = 2016
```

```
Set @UpperLimitForModelYear = 2020

Declare @LowerLimitForListPrice int
Declare @UpperLimitForListPrice int

Set @LowerLimitForListPrice = 200
Set @UpperLimitForListPrice = 5000

Declare @count int
Set @count = 1

While @count <= 20000
Begin

  Select @RandomBrandId = Round(((@UpperLimitForBrandId - @LowerLimitForBrandId) * Rand())
+ @LowerLimitForBrandId, 0)
  Select @RandomCategoryId = Round(((@UpperLimitForCategoryId - @LowerLimitForCategoryId) *
Rand()) + @LowerLimitForCategoryId, 0)
  Select @RandomModelYear = Round(((@UpperLimitForModelYear - @LowerLimitForModelYear) *
Rand()) + @LowerLimitForModelYear, 0)
  Select @RandomListPrice = Round(((@UpperLimitForListPrice - @LowerLimitForListPrice) *
Rand()) + @LowerLimitForListPrice, 0)

  INSERT INTO production.products(product_name,brand_id,category_id,model_year,list_price)
  values ('product - ' + CAST(@count as nvarchar(10)), @RandomBrandId, @RandomCategoryId,
@RandomModelYear, @RandomListPrice)
  Print @count
  Set @count = @count + 1
End

--Insert 20000 rows in sales.stores.
Declare @RandomZipCode int
Declare @LowerLimitForZipCode int
Declare @UpperLimitForZipCode int

Set @LowerLimitForZipCode = 10000
Set @UpperLimitForZipCode = 99999

Declare @count int
SET @count = 1

While @count <= 20000
BEGIN
  Select @RandomZipCode = Round(((@UpperLimitForZipCode - @LowerLimitForZipCode) * Rand())
+ @LowerLimitForZipCode, 0)
  INSERT INTO sales.stores (store_name,phone,email,street,city,state,zip_code)
  values ('store - ' + CAST(@count as nvarchar(10)),'phone-' + CAST(@count as nvarchar(10)),'email' +
CAST(@count as nvarchar(10)) + '@bikes.shop',
  'street-' + CAST(@count as nvarchar(10)),'city-' + CAST(@count as nvarchar(10)),'s-' + CAST(@count
as nvarchar(10)),@RandomZipCode)
```

```
  SET @count = @count + 1END

--Insert 20000 rows into production.stocks
Declare @RandomStoreId int
Declare @RandomProductId int
Declare @RandomQuantity int

Declare @LowerLimitForStoreId int
Declare @UpperLimitForStoreId int

Set @LowerLimitForStoreId = 1
Set @UpperLimitForStoreId = 20003

Declare @LowerLimitForProductId int
Declare @UpperLimitForProductId int

Set @LowerLimitForProductId = 1
Set @UpperLimitForProductId = 20321

Declare @LowerLimitForQuantity int
Declare @UpperLimitForQuantity int

Set @LowerLimitForQuantity = 0
Set @UpperLimitForQuantity = 100

Declare @count int
Set @count = 1

While @count <= 20000
Begin

  Select @RandomStoreId = Round(((@UpperLimitForStoreId - @LowerLimitForStoreId) * Rand()) +
@LowerLimitForStoreId, 0)
  Select @RandomProductId = Round(((@UpperLimitForProductId - @LowerLimitForProductId) *
Rand()) + @LowerLimitForProductId, 0)
  Select @RandomQuantity = Round(((@UpperLimitForQuantity - @LowerLimitForQuantity) * Rand())
+ @LowerLimitForQuantity, 0)

  INSERT INTO production.stocks(store_id,product_id,quantity)
  values (@RandomStoreId, @RandomProductId, @RandomQuantity)
  Set @count = @count + 1
End

-- Insert 20000 rows into sales.customers.
Declare @RandomZipCode int

Declare @LowerLimitForZipCode int
Declare @UpperLimitForZipCode int

Set @LowerLimitForZipCode = 10000
```

```
Set @UpperLimitForZipCode = 99999
Declare @count int
SET @count = 1

While @count <= 20000
BEGIN
  Select @RandomZipCode = Round((((@UpperLimitForZipCode - @LowerLimitForZipCode) * Rand())
+ @LowerLimitForZipCode, 0)
  INSERT INTO sales.customers (first_name,last_name,phone,email,street,city,state,zip_code)
  values ('Fname-' + CAST(@count as nvarchar(10)),'Lname-' + CAST(@count as nvarchar(10)),'phone-'
+ CAST(@count as nvarchar(10)),'email' + CAST(@count as nvarchar(10)) + '@gmail.com',
  'street-' + CAST(@count as nvarchar(10)),'city-' + CAST(@count as nvarchar(10)),'s-' + CAST(@count
as nvarchar(10)),@RandomZipCode)
  SET @count = @count + 1
END

-- Insert 20000 rows into sales.staffs.
Declare @RandomStoreId int
Declare @RandomManagerId int

Declare @LowerLimitForStoreId int
Declare @UpperLimitForStoreId int

Set @LowerLimitForStoreId = 1
Set @UpperLimitForStoreId = 20003

Declare @LowerLimitForManagerId int
Declare @UpperLimitForManagerId int

Set @LowerLimitForManagerId = 1
Set @UpperLimitForManagerId= 10

Declare @count int
Set @count = 1

While @count <= 20000
Begin

  Select @RandomStoreId = Round((((@UpperLimitForStoreId - @LowerLimitForStoreId) * Rand()) +
@LowerLimitForStoreId, 0)
  Select @RandomManagerId = Round((((@UpperLimitForManagerId - @LowerLimitForManagerId) *
Rand()) + @LowerLimitForManagerId, 0)

INSERT INTO sales.staffs(first_name,last_name,email,phone,active,store_id,manager_id)
    VALUES ('Fname-' + CAST(@count as nvarchar(10)),'Lname-' + CAST(@count as
nvarchar(10)),'email' + CAST(@count as nvarchar(10)) + '@bikes.shop','phone-' + CAST(@count as
nvarchar(10)),1,
          @RandomStoreId, @RandomManagerId)
    Set @count = @count + 1
End
```

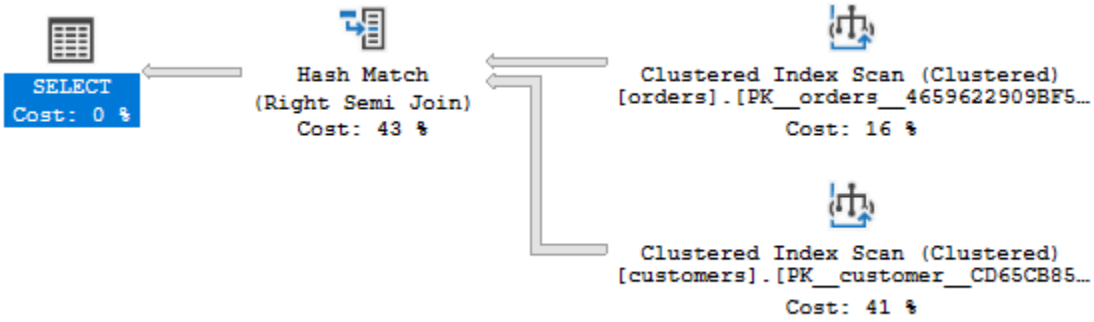
```
--Insert 20000 rows in sales.orders.
Declare @RandomCustomerId int
Declare @RandomOrderStatus int
Declare @RandomStoreId int
Declare @RandomStaffId int

Declare @LowerLimitForCustomerId int
Declare @UpperLimitForCustomerId int

Set @LowerLimitForCustomerId = 1
Set @UpperLimitForCustomerId = 21445


Declare @LowerLimitForOrderStatus int
Declare @UpperLimitForOrderStatus int

Set @LowerLimitForOrderStatus = 1
Set @UpperLimitForOrderStatus = 5

Declare @LowerLimitForStoreId int
Declare @UpperLimitForStoreId int

Set @LowerLimitForStoreId = 1
Set @UpperLimitForStoreId = 20003

Declare @LowerLimitForStaffId int
Declare @UpperLimitForStaffId int

Set @LowerLimitForStaffId = 200
Set @UpperLimitForStaffId = 5000

Declare @count int
Set @count = 1

While @count <= 20000
Begin

  Select @RandomCustomerId = Round(((@UpperLimitForCustomerId - @LowerLimitForCustomerId)
* Rand()) + @LowerLimitForCustomerId, 0)
  Select @RandomOrderStatus = Round(((@UpperLimitForOrderStatus - @LowerLimitForOrderStatus)
* Rand()) + @LowerLimitForOrderStatus, 0)
  Select @RandomStoreId = Round(((@UpperLimitForStoreId - @LowerLimitForStoreId) * Rand()) +
@LowerLimitForStoreId, 0)
  Select @RandomStaffId = Round(((@UpperLimitForStaffId - @LowerLimitForStaffId) * Rand()) +
@LowerLimitForStaffId, 0)
```

```
  INSERT INTO
sales.orders(customer_id,order_status,order_date,required_date,shipped_date,store_id,staff_id)
  values (@RandomCustomerId, @RandomOrderStatus, '2019-07-01', '2019-07-01',NULL,
@RandomStoreId, @RandomStaffId)
  Print @count
  Set @count = @count + 1
End

--Insert 20000 rows in sales.order_items.
Declare @RandomOrderId int
Declare @RandomItemId int
Declare @RandomProductId int
Declare @RandomQuantity int
Declare @RandomListPrice int
Declare @RandomDiscount int

Declare @LowerLimitForOrderId int
Declare @UpperLimitForOrderId int

Set @LowerLimitForOrderId = 1
Set @UpperLimitForOrderId = 21616

Declare @LowerLimitForItemId int
Declare @UpperLimitForItemId int

Set @LowerLimitForItemId = 1
Set @UpperLimitForItemId = 30

Declare @LowerLimitForProductId int
Declare @UpperLimitForProductId int

Set @LowerLimitForProductId = 1
Set @UpperLimitForProductId = 20321

Declare @LowerLimitForQuantity int
Declare @UpperLimitForQuantity int

Set @LowerLimitForQuantity = 0
Set @UpperLimitForQuantity = 10

Declare @LowerLimitForListPrice int
Declare @UpperLimitForListPrice int

Set @LowerLimitForListPrice = 200
Set @UpperLimitForListPrice = 4000

Declare @LowerLimitForDiscount int
Declare @UpperLimitForDiscount int

Set @LowerLimitForDiscount = 0.01
```

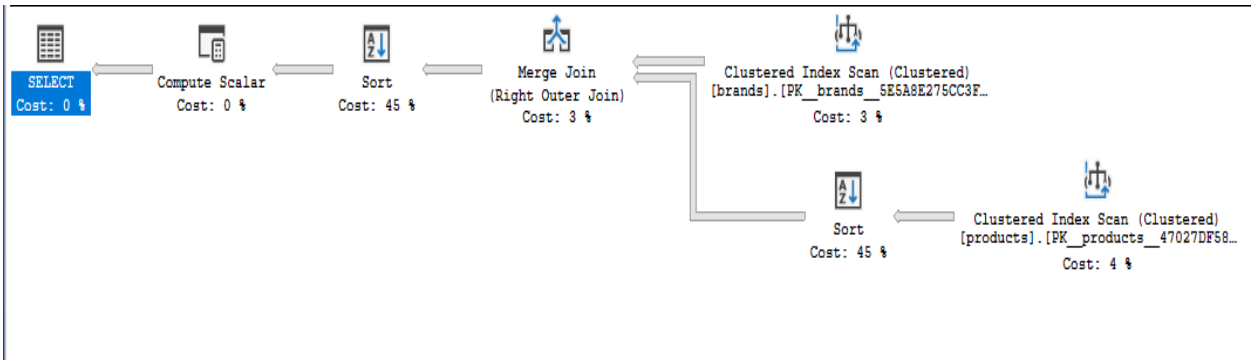
```
Set @UpperLimitForDiscount = 0.30

Declare @count int
Set @count = 1

While @count <= 20000
Begin

  Select @RandomOrderId = Round(((@UpperLimitForOrderId - @LowerLimitForOrderId) * Rand()) +
@LowerLimitForOrderId, 0)
  Select @RandomItemId = Round(((@UpperLimitForItemId - @LowerLimitForItemId) * Rand()) +
@LowerLimitForItemId, 0)
  Select @RandomProductId = Round(((@UpperLimitForProductId - @LowerLimitForProductId) *
Rand()) + @LowerLimitForProductId, 0)
  Select @RandomQuantity = Round(((@UpperLimitForQuantity - @LowerLimitForQuantity) * Rand())
+ @LowerLimitForQuantity, 0)
  Select @RandomListPrice = Round(((@UpperLimitForListPrice - @LowerLimitForListPrice) *
Rand()) + @LowerLimitForListPrice, 0)
  Select @RandomDiscount = Round(((@UpperLimitForDiscount - @LowerLimitForDiscount) *
Rand()) + @LowerLimitForDiscount, 0)

  INSERT INTO sales.order_items(order_id,item_id,product_id,quantity,list_price,discount)
  values (@RandomOrderId, @RandomItemId,
@RandomProductId,@RandomQuantity,@RandomListPrice,@RandomDiscount)
  Set @count = @count + 1
End
```

Execution Plans

```
--1. Give the names of customers whose orders were delayed. Your answer should have the following schema.
--Customers(CustomerId, CustomerName)
BEGIN TRANSACTION
SELECT DISTINCT customer_id,first_name AS CustomerName
FROM sales.customers WHERE customer_id IN
(SELECT DISTINCT customer_id from sales.orders WHERE shipped_date>required_date)
COMMIT
GO
```

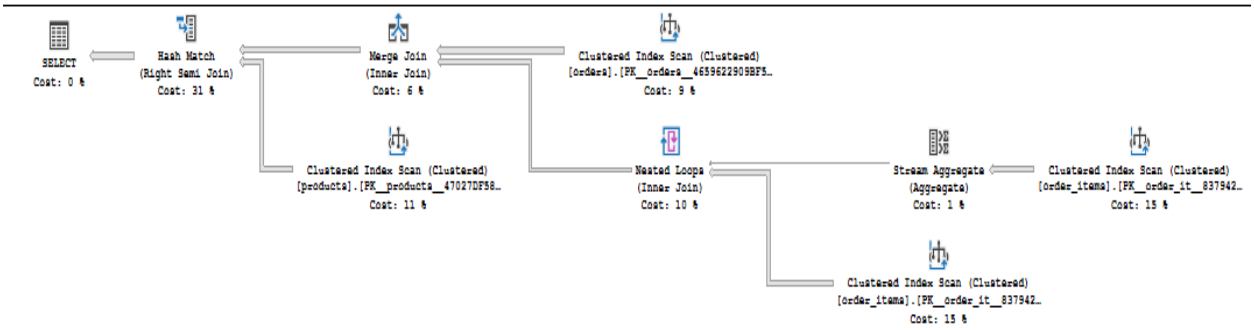



```
--2. Give the products details with its brand name. Products(ProductName, SupplierName)
BEGIN TRANSACTION
SELECT product_name,
(Select brand_name from production.brands where production.brands.brand_id =
production.products.brand_id) AS BrandName
FROM production.products
ORDER BY BrandName
COMMIT
GO
```

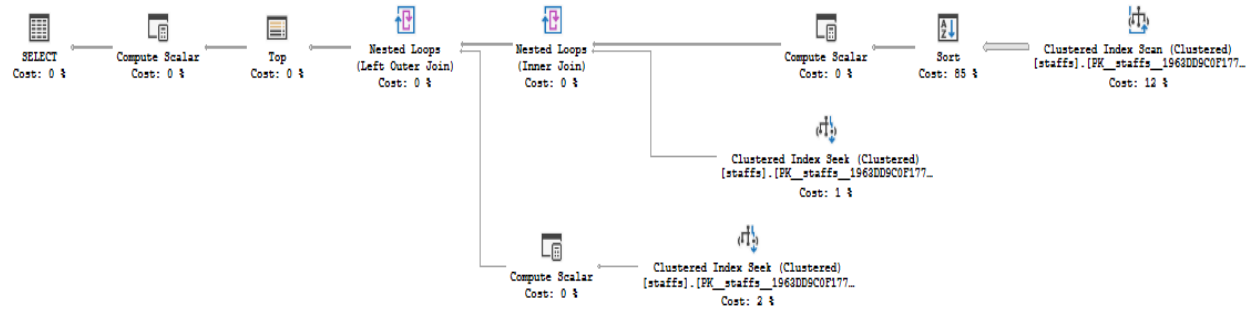



```
--3. Give the name of top products which have highest sale in the year 2019.
BEGIN TRANSACTION
SELECT product_name AS ProductsHighestSaleIn2019
FROM production.products
```

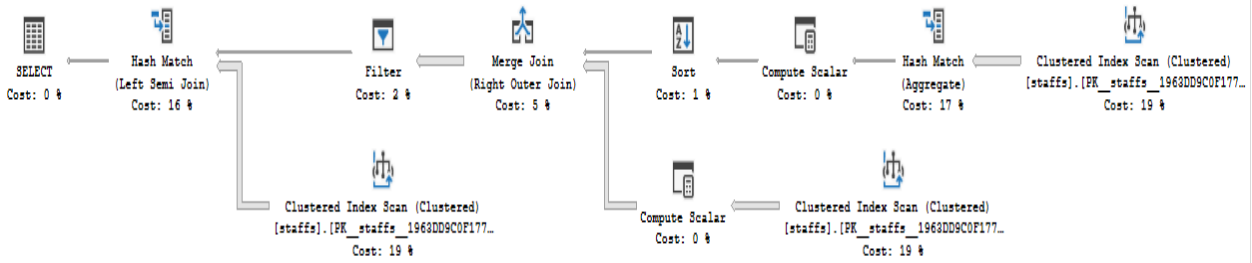
```
Where product_id IN
(Select product_id from sales.order_items where order_id IN
(select order_iD from sales.orders where sales.order_items.quantity= (SELECT
MAX(sales.order_items.quantity) FROM sales.order_items WHERE YEAR(sales.orders.order_date) =
2019)))
COMMIT
GO
```



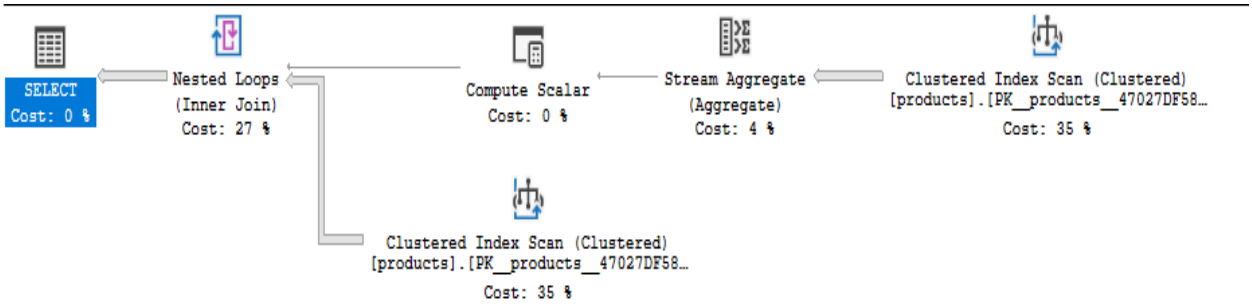

```
--4. Give the name of staff with its manager name. Schema should have the following schema.
--(EmployeeName, ManagerName)
BEGIN TRANSACTION
SELECT TOP 50 CONCAT(emp.first_name,' ',emp.last_name) AS StaffName,
(SELECT CONCAT(mng.first_name,' ',mng.last_name) from sales.staffs mng where emp.manager_id=
mng.staff_id) AS ManagerName
FROM sales.staffs emp,sales.staffs mng where emp.manager_id= mng.staff_id
ORDER BY emp.first_Name,emp.last_name
COMMIT
GO
```



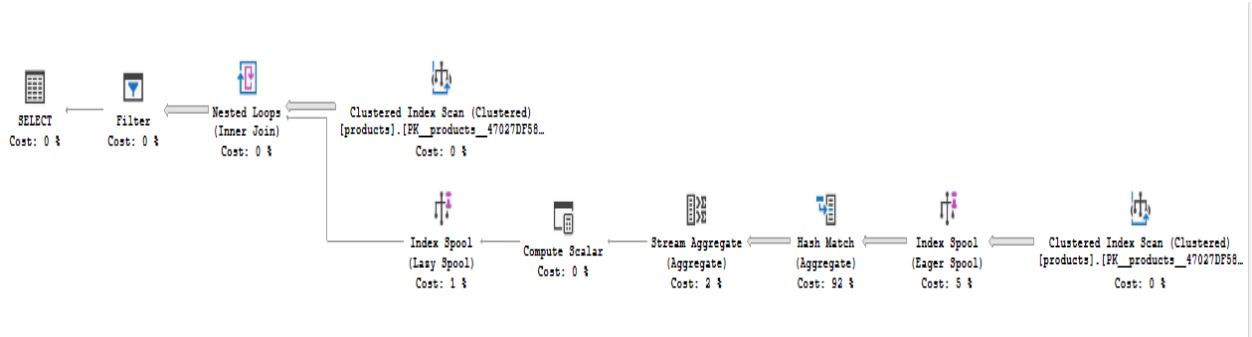

```
--5. Give the full names of managers who have greater than two employees.
BEGIN TRANSACTION
SELECT CONCAT(mng.first_name,' ',mng.last_name) AS ManagerNameLessThan2Employee
FROM sales.staffs mng Where staff_id IN
(Select manager_id from sales.staffs emp WHERE (SELECT COUNT(sales.staffs.manager_id) FROM
sales.staffs WHERE sales.staffs.manager_id=mng.staff_id) > 2)
COMMIT
GO
```



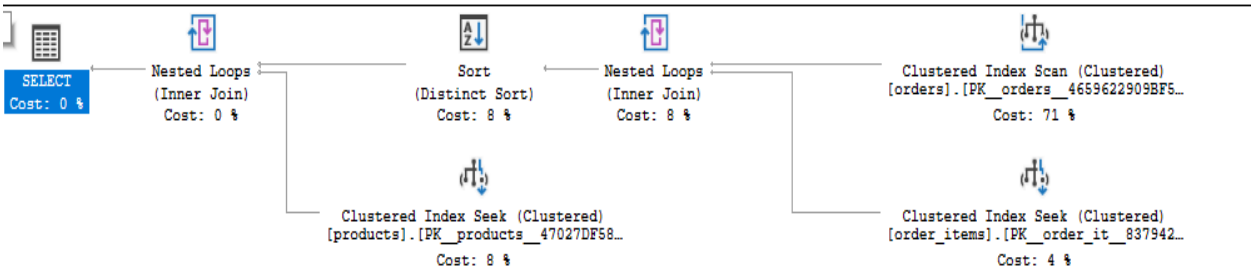

```
--6. List all the products whose price is more than average price.
BEGIN TRANSACTION
SELECT product_id, product_name AS ProductPriceMoreThanAvgPrice
FROM production.products
WHERE list_price > (SELECT AVG(list_price) FROM production.products)
COMMIT
GO
```



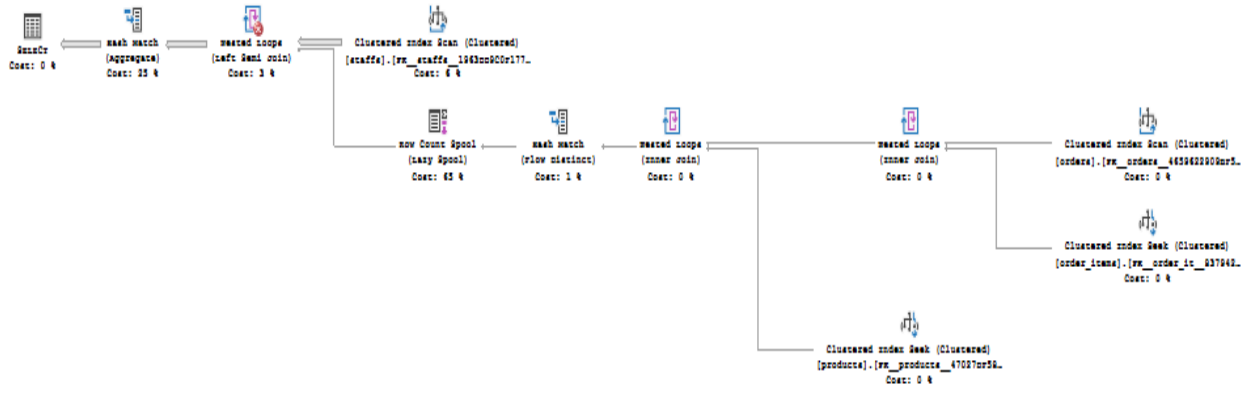

```
--7. Find second highest priced product without using TOP statement
BEGIN TRANSACTION
SELECT a.product_id, a.product_name AS SecondHighestPricedProduct
FROM production.products a
WHERE (SELECT COUNT(DISTINCT list_price) FROM production.products b WHERE
a.list_price<=b.list_price) = 2
COMMIT
GO
```



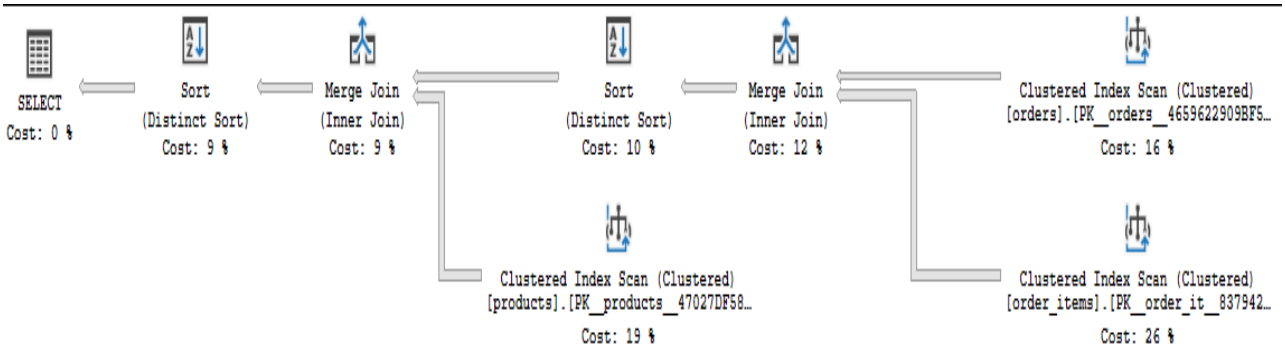
```
--8. List the names of products which were ordered on 1 March 2016.
BEGIN TRANSACTION
SELECT product_name AS OrderedOn1March2016
FROM production.products
Where product_id IN
(Select product_id from sales.order_items where order_id IN
(select order_id from sales.orders where sales.orders.order_date = '20160103'))
COMMIT
GO
```




```
--9. List the names of suppliers whose supplied products were ordered in 2017.
BEGIN TRANSACTION
SELECT DISTINCT sales.staffs.first_name AS SuppliersWhoseSuppliedProductsOrderedIn2017
from sales.staffs where staff_id IN
(Select staff_id from production.products Where product_id IN
(Select product_id from sales.order_items Where order_id IN
(Select order_id From sales.orders WHERE YEAR(sales.Orders.Order_Date) = 2017)))
COMMIT
GO
```



```
--10. Give the name of products which were not ordered in 2016.
BEGIN TRANSACTION
SELECT DISTINCT production.products.product_name AS ProductsNOTOrderedIn1996
from production.products Where product_id IN
(Select product_id From sales.order_items Where order_id IN
(Select order_id from sales.orders Where YEAR(order_date) != 2019 ))
ORDER BY product_name
COMMIT
GO
```



Views

```
USE [BikeStores]
GO

-- Name of Products whose Price is greater than avg price.
CREATE VIEW productsHavingPriceGreaterThanAvgPrices AS
SELECT product_name, list_price
FROM production.products
WHERE list_price >(SELECT AVG(list_price) FROM production.products)
GO

-- Products quantity greater than 20.
CREATE VIEW AvailableStock AS
SELECT product_name, quantity
FROM production.products,production.stocks
WHERE production.stocks.quantity >20
GO

-- Customer name and city and phone
CREATE VIEW CustomerInfo AS
SELECT first_name,phone,city
FROM sales.customers
GO
```

```
-- Discount greater than 0.20
CREATE VIEW discount AS
SELECT product_id,list_price
FROM sales.order_items
where discount > 0.20
GO


-- Staff name, email and phone
CREATE VIEW StaffInfo AS
SELECT first_name,email,phone
FROM sales.staffs
GO
```

Indexes

```
USE BikeStores
GO

CREATE NONCLUSTERED INDEX [brand_name] ON production.brands(brand_name)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON)

CREATE NONCLUSTERED INDEX [category_name] ON production.categories(category_name)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON)

CREATE NONCLUSTERED INDEX [product_name] ON production.products(product_name)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON)

CREATE NONCLUSTERED INDEX [customer_city] ON sales.customers(city)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON)
```

```
CREATE NONCLUSTERED INDEX [staff_city] ON sales.staffs(email)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON)

CREATE NONCLUSTERED INDEX [store_name] ON sales.stores(store_name)
WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS = ON,
ALLOW_PAGE_LOCKS = ON)
```

Conclusion:

Index is considered as one of the most important factors in the SQL Server performance. It helps in speeding up the queries by providing swift access to the requested data, instead of scanning the whole table to retrieve a few records. As we can also see in the above execution plans that indexes reduces the execution time of queries.