Assignment -2

Exercise: SubQueries

Ques 1

Create a query that displays all rows and the following columns from the AdventureWorks2019.HumanResources.Employee table:

- BusinessEntityID
- JobTitle
- VacationHours

Also include a derived column called "MaxVacationHours" that returns the maximum amount of vacation hours for any one employee, in any given row.

```
select
    BusinessEntityID,
    JobTitle,
    VacationHours,
    (select max(VacationHours) from HumanResources.Employee)as MaxVacationHours
from
    HumanResources.Employee hre
```

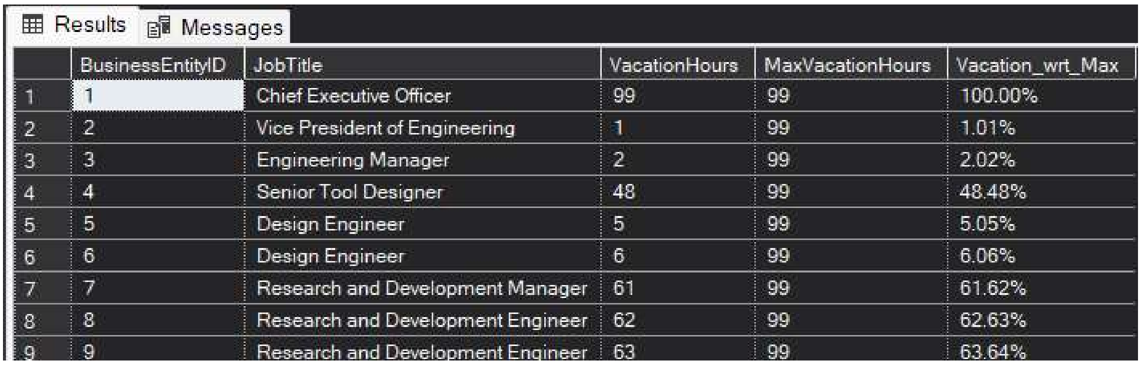
Ques 2

Add a new derived field to your query from Exercise 1, which returns the percent an individual employees' vacation hours are, *of the maximum vacation hours for any employee*. For example, the record for the employee with the most vacation hours should have a value of 1.00, or 100%, in this column.

Hints:

- You can repurpose your logic from "MaxVacationHours" for the denominator.
- Make sure you multiply at least one side of your equation by 1.0, to ensure the output will be a decimal.

```
select
    BusinessEntityID,
    JobTitle,
    VacationHours,
    (select max(VacationHours) from HumanResources.Employee)as MaxVacationHours,
    CONCAT(CAST(CAST(VacationHours * 100.0 / (SELECT MAX(VacationHours) FROM HumanResources.Employee) AS
DECIMAL(5, 2)) AS VARCHAR(10)), '%') AS Vacation_wrt_Max
from
    HumanResources.Employee hre
```



	BusinessEntityID	JobTitle	VacationHours	MaxVacationHours	Vacation_wrt_Max
1	1	Chief Executive Officer	99	99	100.00%
2	2	Vice President of Engineering	1	99	1.01%
3	3	Engineering Manager	2	99	2.02%
4	4	Senior Tool Designer	48	99	48.48%
5	5	Design Engineer	5	99	5.05%
6	6	Design Engineer	6	99	6.06%
7	7	Research and Development Manager	61	99	61.62%
8	8	Research and Development Engineer	62	99	62.63%
9	9	Research and Development Engineer	63	99	63.64%

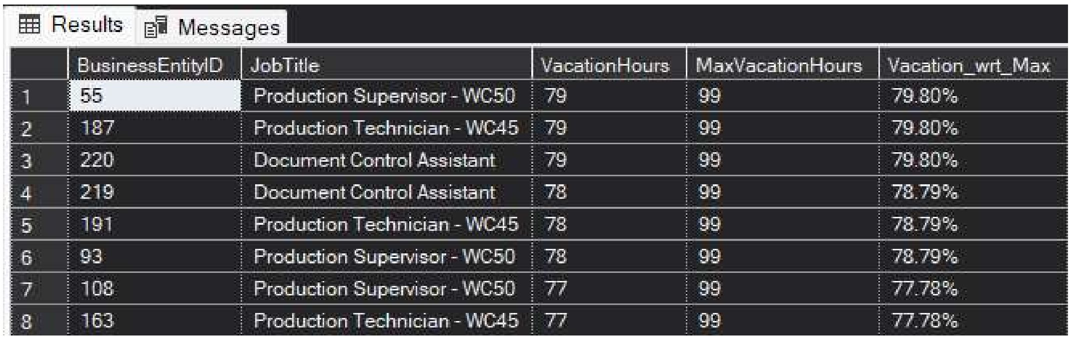
Ques 3

Refine your output with a criterion in the WHERE clause that filters out any employees whose vacation hours are less then 80% of the maximum amount of vacation hours for any one employee. In other words, return only employees who have at least 80% as much vacation time as the employee with the most vacation time.

Hint: The query should return 60 rows

```
select
    BusinessEntityID,
    JobTitle,
    VacationHours,
    (select max(VacationHours) from HumanResources.Employee)as MaxVacationHours,
    CONCAT(CAST(CAST(VacationHours * 100.0 / (SELECT MAX(VacationHours) FROM HumanResources.Employee) AS
DECIMAL(5, 2)) AS VARCHAR(10)), '%') AS Vacation_wrt_Max
from
    HumanResources.Employee hre
WHERE
    CAST(VacationHours * 100.0 / (SELECT MAX(VacationHours) FROM HumanResources.Employee) AS DECIMAL(5,
2)) < 80
ORDER BY
    VacationHours DESC
OFFSET 0 ROWS FETCH NEXT 60 ROWS ONLY;
```

- Note: SQL server dont have `LIMIT`
- When using `OFFSET` and `FETCH NEXT`, it's essential to have an `ORDER BY` clause. Without it, the result set may be non-deterministic, meaning the rows returned could change each time you run the query. The `ORDER BY` clause determines which rows are returned and in what order.



	BusinessEntityID	JobTitle	VacationHours	MaxVacationHours	Vacation_wrt_Max
1	55	Production Supervisor - WC50	79	99	79.80%
2	187	Production Technician - WC45	79	99	79.80%
3	220	Document Control Assistant	79	99	79.80%
4	219	Document Control Assistant	78	99	78.79%
5	191	Production Technician - WC45	78	99	78.79%
6	93	Production Supervisor - WC50	78	99	78.79%
7	108	Production Supervisor - WC50	77	99	77.78%
8	163	Production Technician - WC45	77	99	77.78%

Exercise: Correlated Subqueries

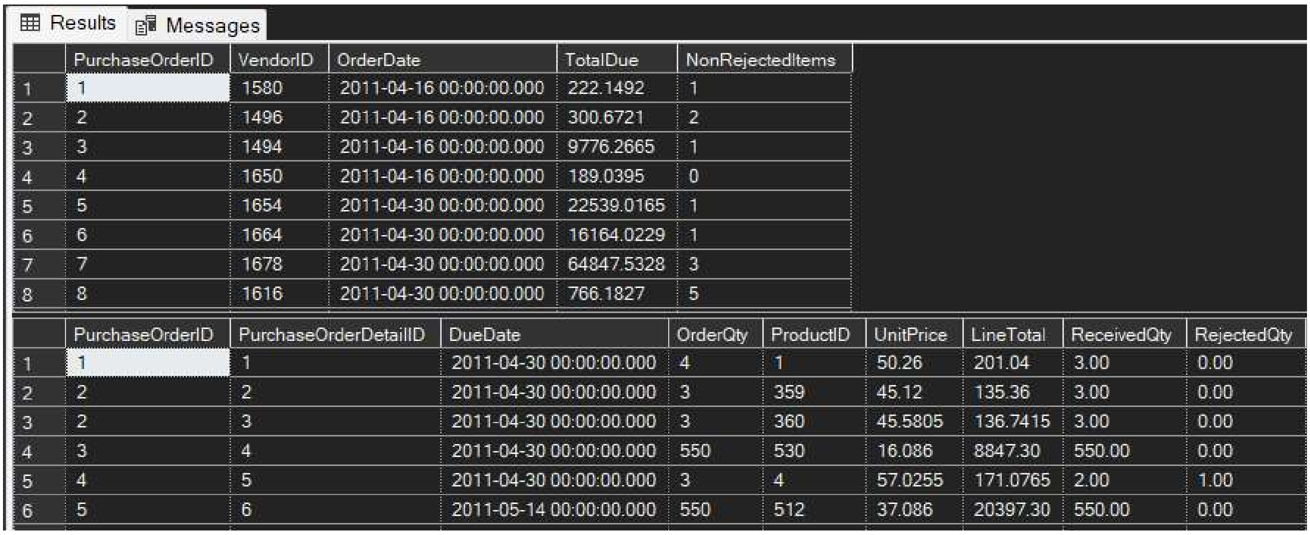
Ques 1

Write a query that outputs all records from the Purchasing.PurchaseOrderHeader table. Include the following columns from the table:

- PurchaseOrderID
- VendorID
- OrderDate
- TotalDue

Add a derived column called **NonRejectedItems** which returns, for each purchase order ID in the query output, the number of line items from the Purchasing.PurchaseOrderDetail table which did not have any rejections (i.e., RejectedQty = 0). Use a correlated subquery to do this.

```
select
    PurchaseOrderID,
    VendorID,
    OrderDate,
    TotalDue,
    (
    select count(*)
    from Purchasing.PurchaseOrderDetail ppod
    where ppod.PurchaseOrderID = ppoh.PurchaseOrderId
    and RejectedQty = 0
    )
    as NonRejectedItems
from Purchasing.PurchaseOrderHeader ppoh
```



	PurchaseOrderID	VendorID	OrderDate	TotalDue	NonRejectedItems
1	1	1580	2011-04-16 00:00:00.000	222.1492	1
2	2	1496	2011-04-16 00:00:00.000	300.6721	2
3	3	1494	2011-04-16 00:00:00.000	9776.2665	1
4	4	1650	2011-04-16 00:00:00.000	189.0395	0
5	5	1654	2011-04-30 00:00:00.000	22539.0165	1
6	6	1664	2011-04-30 00:00:00.000	16164.0229	1
7	7	1678	2011-04-30 00:00:00.000	64847.5328	3
8	8	1616	2011-04-30 00:00:00.000	766.1827	5

	PurchaseOrderID	PurchaseOrderDetailID	DueDate	OrderQty	ProductID	UnitPrice	LineTotal	ReceivedQty	RejectedQty
1	1	1	2011-04-30 00:00:00.000	4	1	50.26	201.04	3.00	0.00
2	2	2	2011-04-30 00:00:00.000	3	359	45.12	135.36	3.00	0.00
3	2	3	2011-04-30 00:00:00.000	3	360	45.5805	136.7415	3.00	0.00
4	3	4	2011-04-30 00:00:00.000	550	530	16.086	8847.30	550.00	0.00
5	4	5	2011-04-30 00:00:00.000	3	4	57.0255	171.0765	2.00	1.00
6	5	6	2011-05-14 00:00:00.000	550	512	37.086	20397.30	550.00	0.00

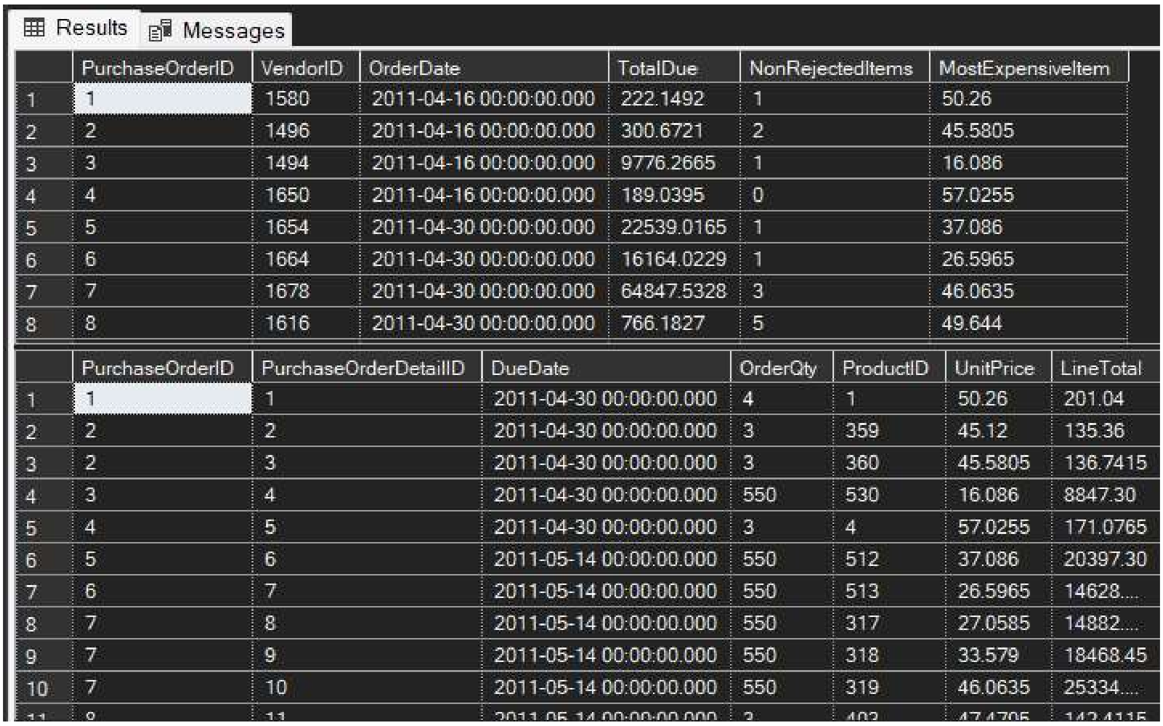
Ques 2

Modify your query to include a second derived field called **MostExpensiveItem**.

This field should return, for each purchase order ID, the UnitPrice of the most expensive item for that order in the Purchasing.PurchaseOrderDetail table.

Use a correlated subquery to do this as well.

```
select
    PurchaseOrderID,
    VendorID,
    OrderDate,
    TotalDue,
    (
    select count(*)
    from Purchasing.PurchaseOrderDetail ppod
    where ppod.PurchaseOrderID = ppoh.PurchaseOrderId
    and RejectedQty = 0
    )
    as NonRejectedItems,
    (
    select max(unitprice)
    from Purchasing.PurchaseOrderDetail ppod
    where ppod.PurchaseOrderID = ppoh.PurchaseOrderId
    )as MostExpensiveItem
from Purchasing.PurchaseOrderHeader ppoh

select *  from Purchasing.PurchaseOrderDetail
```



	PurchaseOrderID	VendorID	OrderDate	TotalDue	NonRejectedItems	MostExpensiveItem
1	1	1580	2011-04-16 00:00:00.000	222.1492	1	50.26
2	2	1496	2011-04-16 00:00:00.000	300.6721	2	45.5805
3	3	1494	2011-04-16 00:00:00.000	9776.2665	1	16.086
4	4	1650	2011-04-16 00:00:00.000	189.0395	0	57.0255
5	5	1654	2011-04-30 00:00:00.000	22539.0165	1	37.086
6	6	1664	2011-04-30 00:00:00.000	16164.0229	1	26.5965
7	7	1678	2011-04-30 00:00:00.000	64847.5328	3	46.0635
8	8	1616	2011-04-30 00:00:00.000	766.1827	5	49.644

	PurchaseOrderID	PurchaseOrderDetailID	DueDate	OrderQty	ProductID	UnitPrice	LineTotal
1	1	1	2011-04-30 00:00:00.000	4	1	50.26	201.04
2	2	2	2011-04-30 00:00:00.000	3	359	45.12	135.36
3	2	3	2011-04-30 00:00:00.000	3	360	45.5805	136.7415
4	3	4	2011-04-30 00:00:00.000	550	530	16.086	8847.30
5	4	5	2011-04-30 00:00:00.000	3	4	57.0255	171.0765
6	5	6	2011-05-14 00:00:00.000	550	512	37.086	20397.30
7	6	7	2011-05-14 00:00:00.000	550	513	26.5965	14628....
8	7	8	2011-05-14 00:00:00.000	550	317	27.0585	14882....
9	7	9	2011-05-14 00:00:00.000	550	318	33.579	18468.45
10	7	10	2011-05-14 00:00:00.000	550	319	46.0635	25334....
11	8	11	2011-05-14 00:00:00.000	3	403	47.4705	142.4115

Exercise: EXISTS()

Ques 1

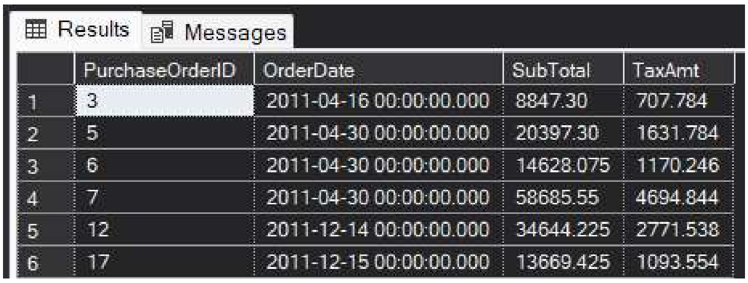
Select all records from the Purchasing.PurchaseOrderHeader table such that there is at least one item in the order with an order quantity greater than 500. The individual items tied to an order can be found in the Purchasing.PurchaseOrderDetail table.

Select the following columns:

- PurchaseOrderID
- OrderDate
- SubTotal
- TaxAmt

Sort by purchase order ID.

```
select
    ppoh.PurchaseOrderID,
    ppoh.OrderDate,
    ppoh.SubTotal,
    ppoh.TaxAmt
from
    Purchasing.PurchaseOrderHeader ppoh
where exists(
            select 1
            from Purchasing.PurchaseOrderDetail ppod
            where ppod.PurchaseOrderID = ppoh.PurchaseOrderID
            AND OrderQty >500
)
order by ppoh.PurchaseOrderID
```



	PurchaseOrderID	OrderDate	SubTotal	TaxAmt
1	3	2011-04-16 00:00:00.000	8847.30	707.784
2	5	2011-04-30 00:00:00.000	20397.30	1631.784
3	6	2011-04-30 00:00:00.000	14628.075	1170.246
4	7	2011-04-30 00:00:00.000	58685.55	4694.844
5	12	2011-12-14 00:00:00.000	34644.225	2771.538
6	17	2011-12-15 00:00:00.000	13669.425	1093.554

Ques 2

Modify your query from Exercise 1 as follows:

Select all records from the Purchasing.PurchaseOrderHeader table such that there is at least one item in the order with an order quantity greater than 500, AND a unit price greater than $50.00.

Select ALL columns from the Purchasing.PurchaseOrderHeader table for display in your output.

Even if you have aliased this table to enable the use of a JOIN or EXISTS, you can still use the SELECT * shortcut to do this. Assuming you have aliased your table "A", simply use "SELECT A.*" to select all columns from that table.

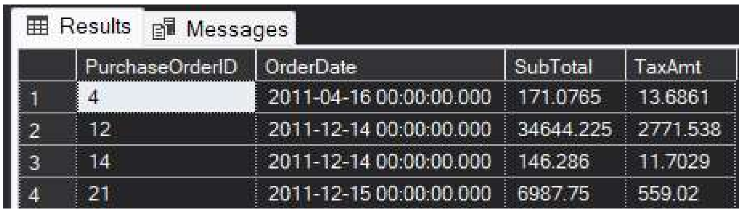
```
select
    ppoh.PurchaseOrderID,
    ppoh.OrderDate,
    ppoh.SubTotal,
    ppoh.TaxAmt
from
    Purchasing.PurchaseOrderHeader ppoh
where exists(
            select 1
            from Purchasing.PurchaseOrderDetail ppod
            where ppod.PurchaseOrderID = ppoh.PurchaseOrderID
            AND ppod.OrderQty >500 and ppod.UnitPrice >50
)
order by ppoh.PurchaseOrderID
```

Ques 3

Select all records from the Purchasing.PurchaseOrderHeader table such that NONE of the items within the order have a rejected quantity greater than 0.

Select ALL columns from the Purchasing.PurchaseOrderHeader table using the "SELECT *" shortcut.

```
select
    ppoh.PurchaseOrderID,
    ppoh.OrderDate,
    ppoh.SubTotal,
    ppoh.TaxAmt
from
    Purchasing.PurchaseOrderHeader ppoh
where exists(
            select 1
            from Purchasing.PurchaseOrderDetail ppod
            where ppod.PurchaseOrderID = ppoh.PurchaseOrderID
            AND ppod.RejectedQty > 0
)
order by ppoh.PurchaseOrderID
```



	PurchaseOrderID	OrderDate	SubTotal	TaxAmt
1	4	2011-04-16 00:00:00.000	171.0765	13.6861
2	12	2011-12-14 00:00:00.000	34644.225	2771.538
3	14	2011-12-14 00:00:00.000	146.286	11.7029
4	21	2011-12-15 00:00:00.000	6987.75	559.02

Exercise: Flattening Rows

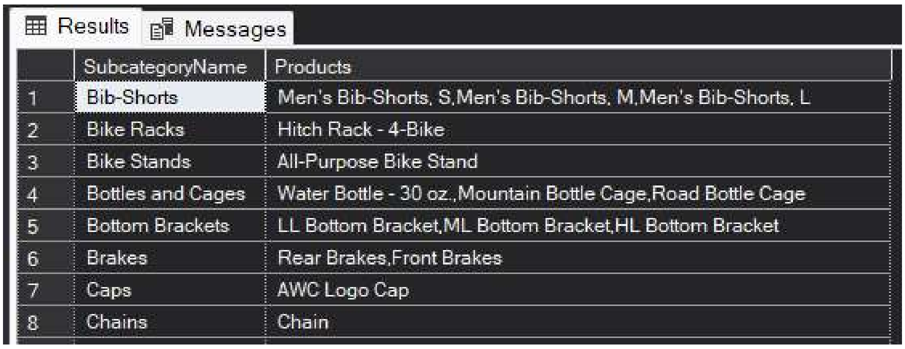
Ques 1

Create a query that displays all rows from the Production.ProductSubcategory table, and includes the following fields:

- The "Name" field from Production.ProductSubcategory, which should be aliased as "SubcategoryName"
- A derived field called "Products" which displays, for each Subcategory in Production.ProductSubcategory, a semicolon-separated list of all products from Production.Product contained within the given subcategory

Hint: Production.ProductSubcategory and Production.Product are related by the "ProductSubcategoryID" field.

```
SELECT
    pps.name as SubcategoryName,
    -- Concatenate product names
    STUFF(
        (
            SELECT ',' + pp.name
            FROM Production.Product pp
            WHERE pps.ProductsubCategoryID = pp.ProductSubcategoryID
            FOR XML PATH('')
        ), 1, 1, ''
    ) AS Products
FROM
    Production.ProductSubcategory pps
```



	SubcategoryName	Products
1	Bib-Shorts	Men's Bib-Shorts, S,Men's Bib-Shorts, M,Men's Bib-Shorts, L
2	Bike Racks	Hitch Rack - 4-Bike
3	Bike Stands	All-Purpose Bike Stand
4	Bottles and Cages	Water Bottle - 30 oz.,Mountain Bottle Cage,Road Bottle Cage
5	Bottom Brackets	LL Bottom Bracket,ML Bottom Bracket,HL Bottom Bracket
6	Brakes	Rear Brakes,Front Brakes
7	Caps	AWC Logo Cap
8	Chains	Chain

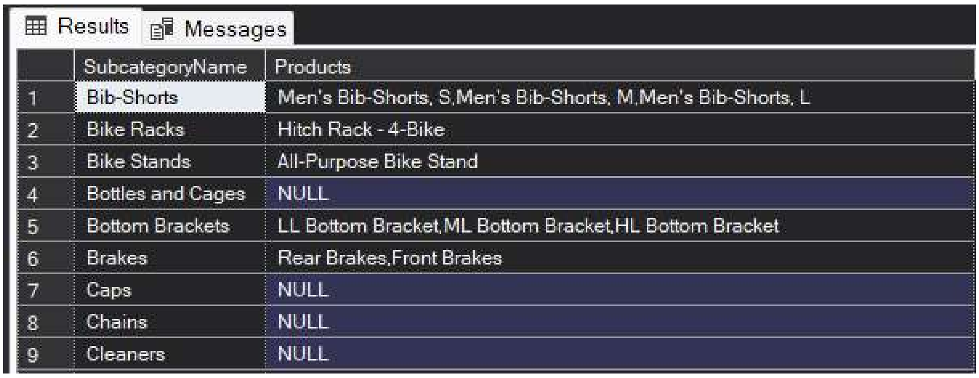
Ques 2

Modify the query from Exercise 1 such that only products with a ListPrice value greater than $50 are listed in the "Products" field.

Hint: Assuming you used a correlated subquery in Exercise 1, keep in mind that you can apply additional criteria to it, just as with any other correlated subquery.

NOTE: Your query should still include ALL product subcategories, but only list associated products greater than 50. But since there are certain product subcategories that don't have *any* associated products greater than $50, some rows in your query output may have a NULL value in the product field.

```
SELECT
    pps.name as SubcategoryName,
    -- Concatenate product names
    STUFF(
        (
            SELECT ',' + pp.name
            FROM Production.Product pp
            WHERE pps.ProductsubCategoryID = pp.ProductSubcategoryID
            AND pp.ListPrice > 50
            FOR XML PATH('')
        ), 1, 1, ''
    ) AS Products
FROM
    Production.ProductSubcategory pps
```



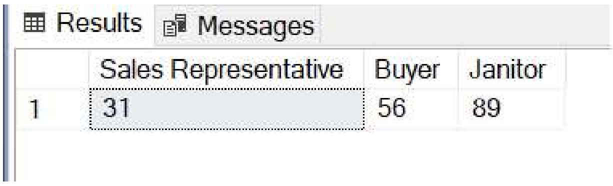
	SubcategoryName	Products
1	Bib-Shorts	Men's Bib-Shorts, S,Men's Bib-Shorts, M,Men's Bib-Shorts, L
2	Bike Racks	Hitch Rack - 4-Bike
3	Bike Stands	All-Purpose Bike Stand
4	Bottles and Cages	NULL
5	Bottom Brackets	LL Bottom Bracket,ML Bottom Bracket,HL Bottom Bracket
6	Brakes	Rear Brakes,Front Brakes
7	Caps	NULL
8	Chains	NULL
9	Cleaners	NULL

Exercise: Pivot()

Ques 1

that summarizes the average amount of vacation time for Sales Representatives, Buyers, and Janitors.

Your output should look like the image below.



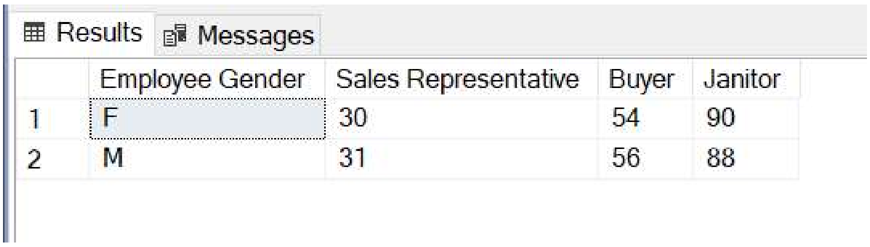
	Sales Representative	Buyer	Janitor
1	31	56	89

```
SELECT
*
FROM
(
SELECT
JobTitle,
VacationHours

FROM HumanResources.Employee
) A

PIVOT(
AVG(VacationHours)
FOR JobTitle IN([Sales Representative],[Buyer],[Janitor])
) B
```

Ques 2

Modify your query from Exercise 1 such that the results are broken out by Gender. Alias the Gender field as "Employee Gender" in your output.

Your output should look like the image below:



	Employee Gender	Sales Representative	Buyer	Janitor
1	F	30	54	90
2	M	31	56	88

```
SELECT
Gender as 'Employee Gender',
[Sales Representative],
[Buyer],
[Janitor]
FROM
(
SELECT
JobTitle,
Gender, -- no pivot or aggregate is applied to it means it will be a row
VacationHours

FROM HumanResources.Employee
) A

PIVOT(
AVG(VacationHours)
FOR JobTitle IN([Sales Representative],[Buyer],[Janitor])
) B
```