Advance Analysis with Windows Function:

Exercise 1: Create a query with the following columns:

FirstName and LastName, from the Person.Person table

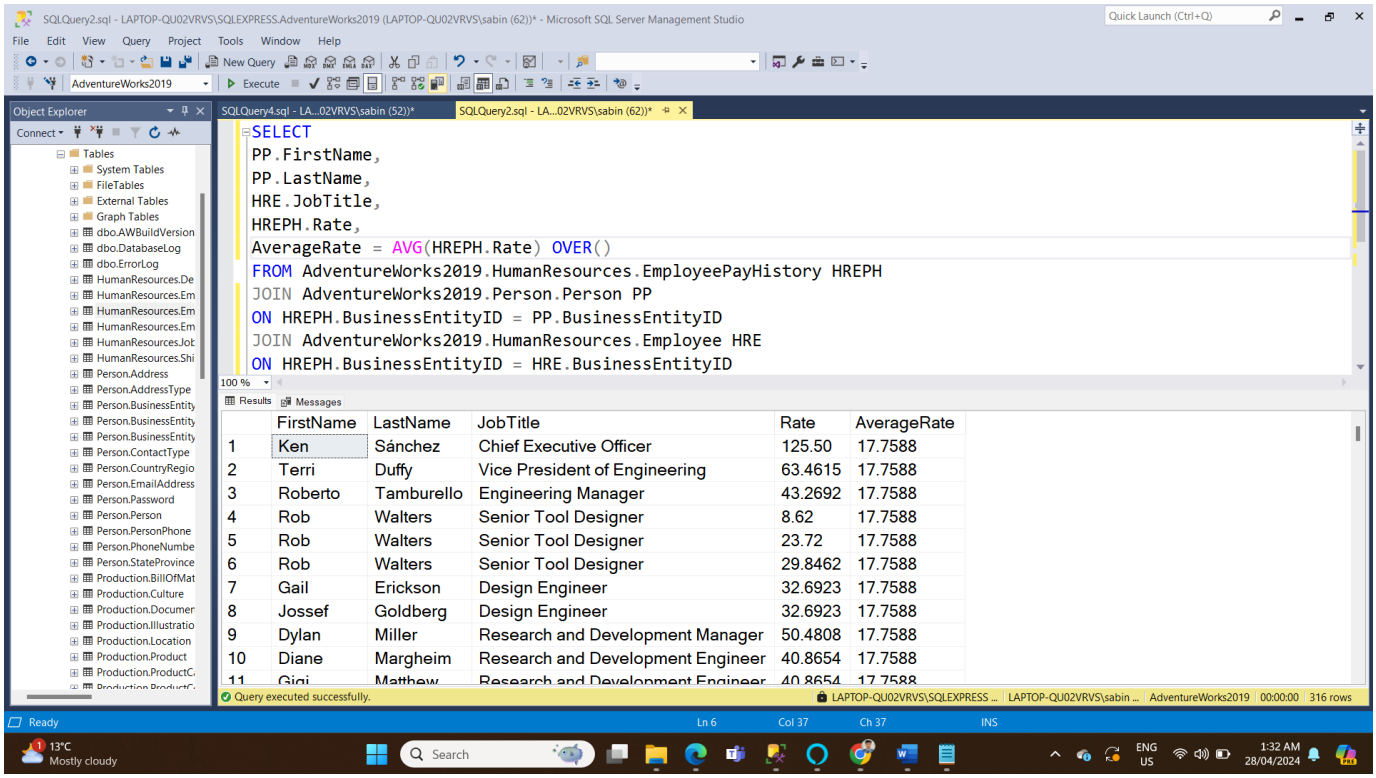
JobTitle, from the HumanResources.Employee table

Rate, from the HumanResources.EmployeePayHistory table

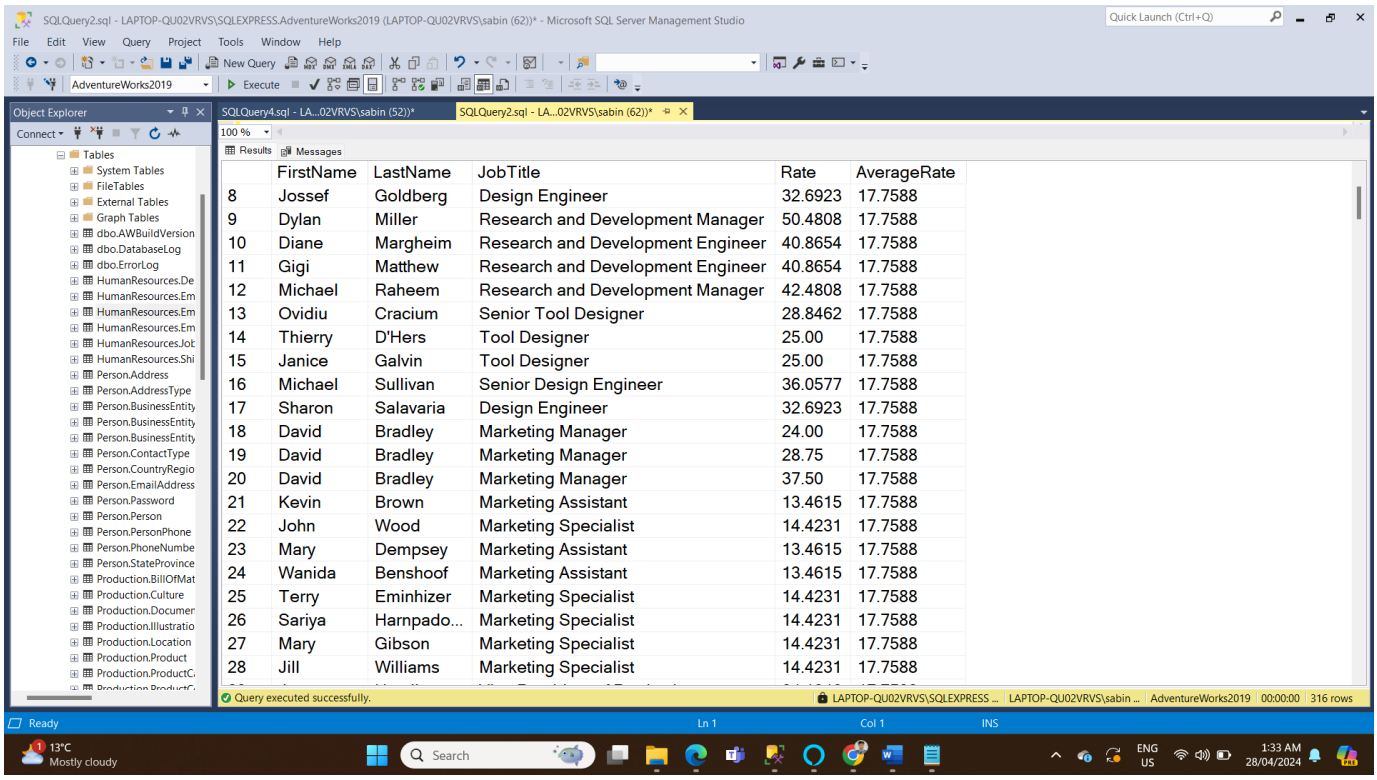
A derived column called "AverageRate" that returns the average of all values in the "Rate" column, in each row.

Answer:

```
SELECT
PP.FirstName,
PP.LastName,
HRE.JobTitle,
HREPH.Rate,
AverageRate = AVG(HREPH.Rate) OVER()
FROM AdventureWorks2019.HumanResources.EmployeePayHistory HREPH
JOIN AdventureWorks2019.Person.Person PP
ON HREPH.BusinessEntityID = PP.BusinessEntityID
JOIN AdventureWorks2019.HumanResources.Employee HRE
ON HREPH.BusinessEntityID = HRE.BusinessEntityID
```



	FirstName	LastName	JobTitle	Rate	AverageRate
1	Ken	Sánchez	Chief Executive Officer	125.50	17.7588
2	Terri	Duffy	Vice President of Engineering	63.4615	17.7588
3	Roberto	Tamburello	Engineering Manager	43.2692	17.7588
4	Rob	Walters	Senior Tool Designer	8.62	17.7588
5	Rob	Walters	Senior Tool Designer	23.72	17.7588
6	Rob	Walters	Senior Tool Designer	29.8462	17.7588
7	Gail	Erickson	Design Engineer	32.6923	17.7588
8	Jossef	Goldberg	Design Engineer	32.6923	17.7588
9	Dylan	Miller	Research and Development Manager	50.4808	17.7588
10	Diane	Margheim	Research and Development Engineer	40.8654	17.7588
11	Gigi	Matthew	Research and Development Engineer	40.8654	17.7588

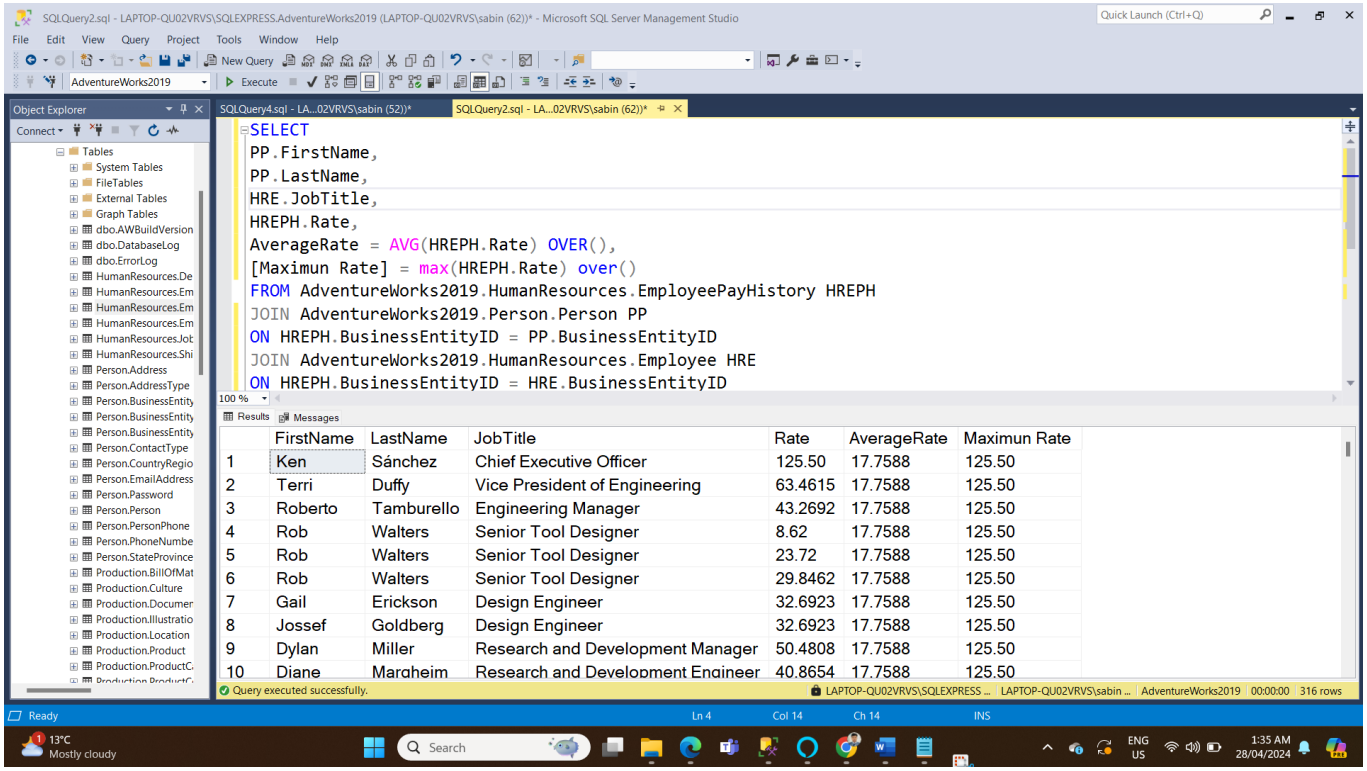


	FirstName	LastName	JobTitle	Rate	AverageRate
8	Jossef	Goldberg	Design Engineer	32.6923	17.7588
9	Dylan	Miller	Research and Development Manager	50.4808	17.7588
10	Diane	Margheim	Research and Development Engineer	40.8654	17.7588
11	Gigi	Matthew	Research and Development Engineer	40.8654	17.7588
12	Michael	Raheem	Research and Development Manager	42.4808	17.7588
13	Ovidiu	Cracium	Senior Tool Designer	28.8462	17.7588
14	Thierry	D'Hers	Tool Designer	25.00	17.7588
15	Janice	Galvin	Tool Designer	25.00	17.7588
16	Michael	Sullivan	Senior Design Engineer	36.0577	17.7588
17	Sharon	Salavaria	Design Engineer	32.6923	17.7588
18	David	Bradley	Marketing Manager	24.00	17.7588
19	David	Bradley	Marketing Manager	28.75	17.7588
20	David	Bradley	Marketing Manager	37.50	17.7588
21	Kevin	Brown	Marketing Assistant	13.4615	17.7588
22	John	Wood	Marketing Specialist	14.4231	17.7588
23	Mary	Dempsey	Marketing Assistant	13.4615	17.7588
24	Wanida	Benshoof	Marketing Assistant	13.4615	17.7588
25	Terry	Eminhizer	Marketing Specialist	14.4231	17.7588
26	Sariya	Harnpado...	Marketing Specialist	14.4231	17.7588
27	Mary	Gibson	Marketing Specialist	14.4231	17.7588
28	Jill	Williams	Marketing Specialist	14.4231	17.7588

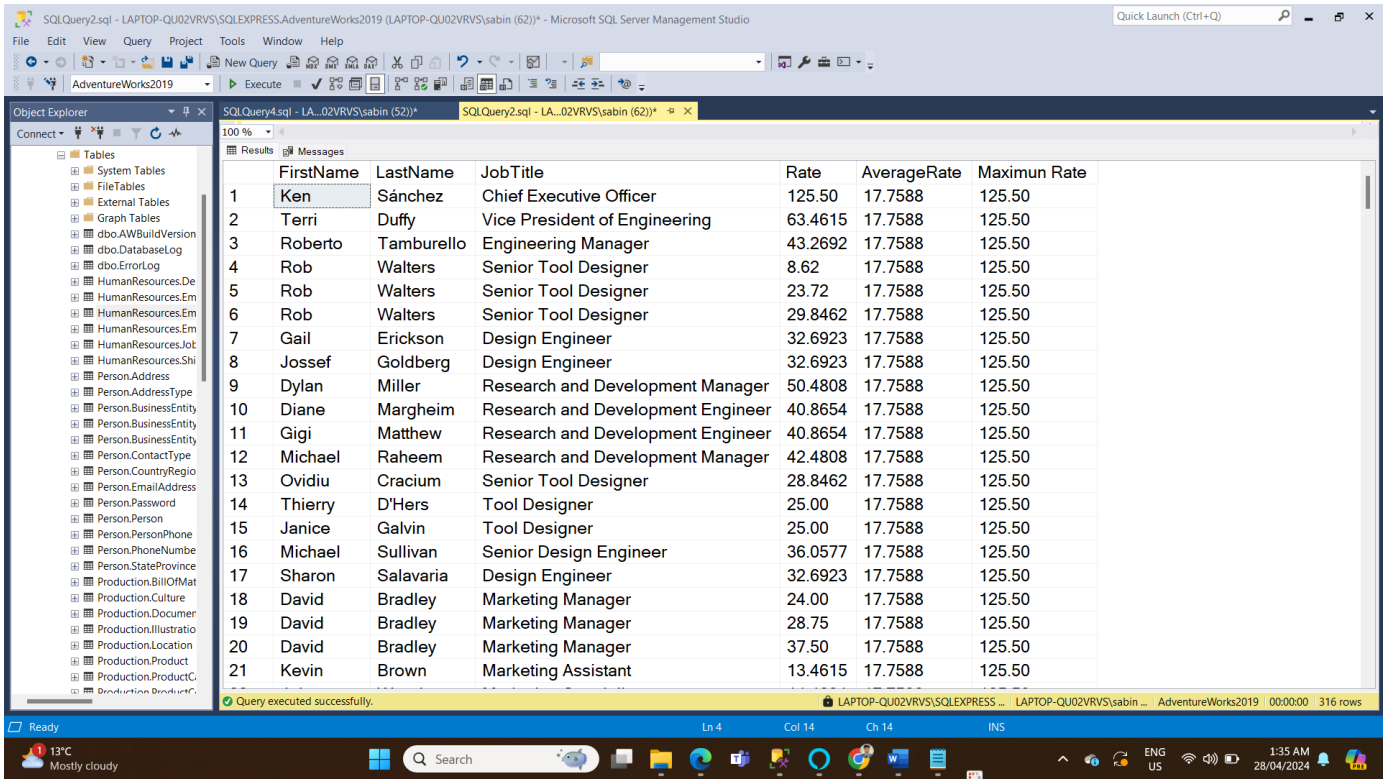
Exercise 2: Enhance your query from Exercise 1 by adding a derived column called "MaximumRate" that returns the largest of all values in the "Rate" column, in each row.

Answer:

```
SELECT
PP.FirstName,
PP.LastName,
HRE.JobTitle,
HREPH.Rate,
AverageRate = AVG(HREPH.Rate) OVER(),
[Maximun Rate] = max(HREPH.Rate) over()
FROM AdventureWorks2019.HumanResources.EmployeePayHistory HREPH
JOIN AdventureWorks2019.Person.Person PP
ON HREPH.BusinessEntityID = PP.BusinessEntityID
JOIN AdventureWorks2019.HumanResources.Employee HRE
ON HREPH.BusinessEntityID = HRE.BusinessEntityID
```



	FirstName	LastName	JobTitle	Rate	AverageRate	Maximun Rate
1	Ken	Sánchez	Chief Executive Officer	125.50	17.7588	125.50
2	Terri	Duffy	Vice President of Engineering	63.4615	17.7588	125.50
3	Roberto	Tamburello	Engineering Manager	43.2692	17.7588	125.50
4	Rob	Walters	Senior Tool Designer	8.62	17.7588	125.50
5	Rob	Walters	Senior Tool Designer	23.72	17.7588	125.50
6	Rob	Walters	Senior Tool Designer	29.8462	17.7588	125.50
7	Gail	Erickson	Design Engineer	32.6923	17.7588	125.50
8	Jossef	Goldberg	Design Engineer	32.6923	17.7588	125.50
9	Dylan	Miller	Research and Development Manager	50.4808	17.7588	125.50
10	Diane	Margheim	Research and Development Engineer	40.8654	17.7588	125.50



Exercise 3:

Enhance your query from Exercise 2 by adding a derived column called"DiffFromAvgRate" that returns the result of the following calculation: An employees's pay rate, MINUS the average of all values in the "Rate" column.

Answer:

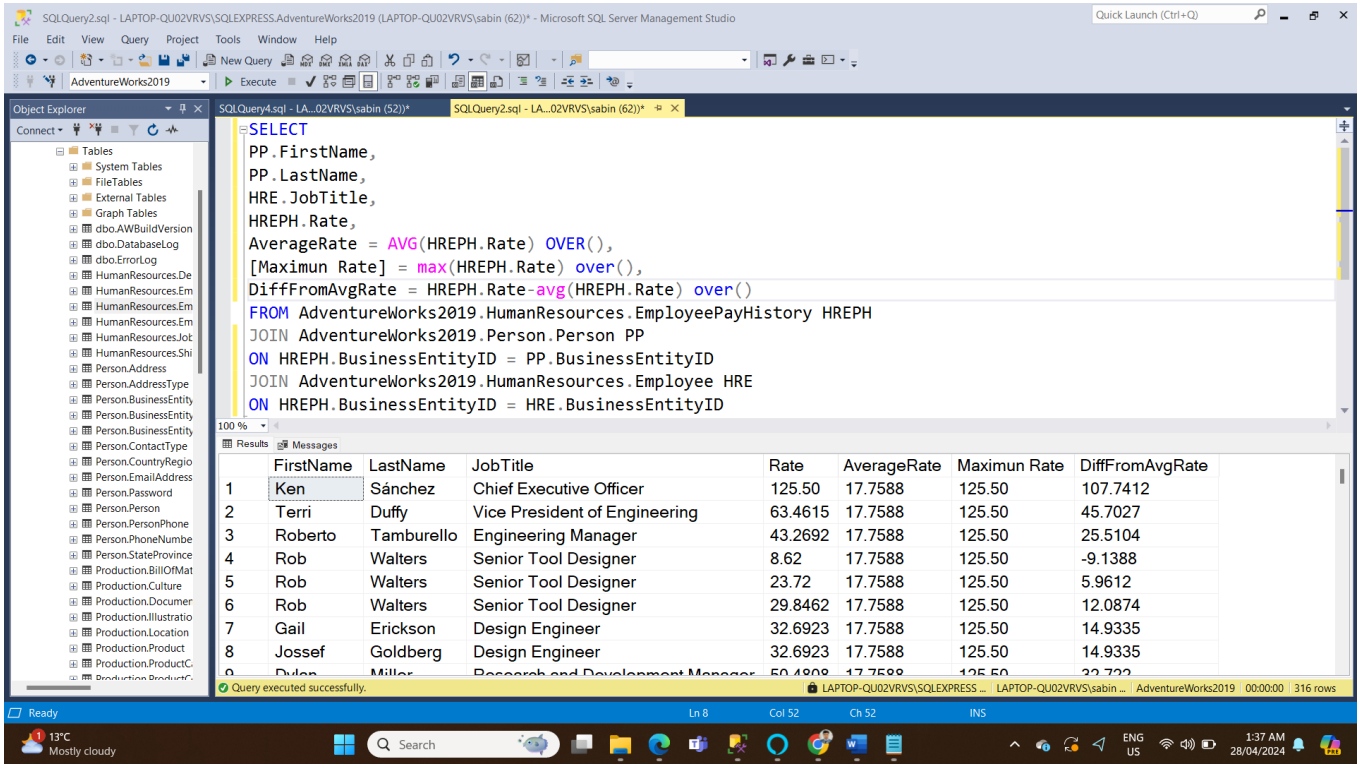
```
SELECT
PP.FirstName,
PP.LastName,
HRE.JobTitle,
HREPH.Rate,
AverageRate = AVG(HREPH.Rate) OVER(),
[Maximun Rate] = max(HREPH.Rate) over(),
DiffFromAvgRate = HREPH.Rate-avg(HREPH.Rate) over()
FROM AdventureWorks2019.HumanResources.EmployeePayHistory HREPH
```

JOIN AdventureWorks2019.Person.Person PP

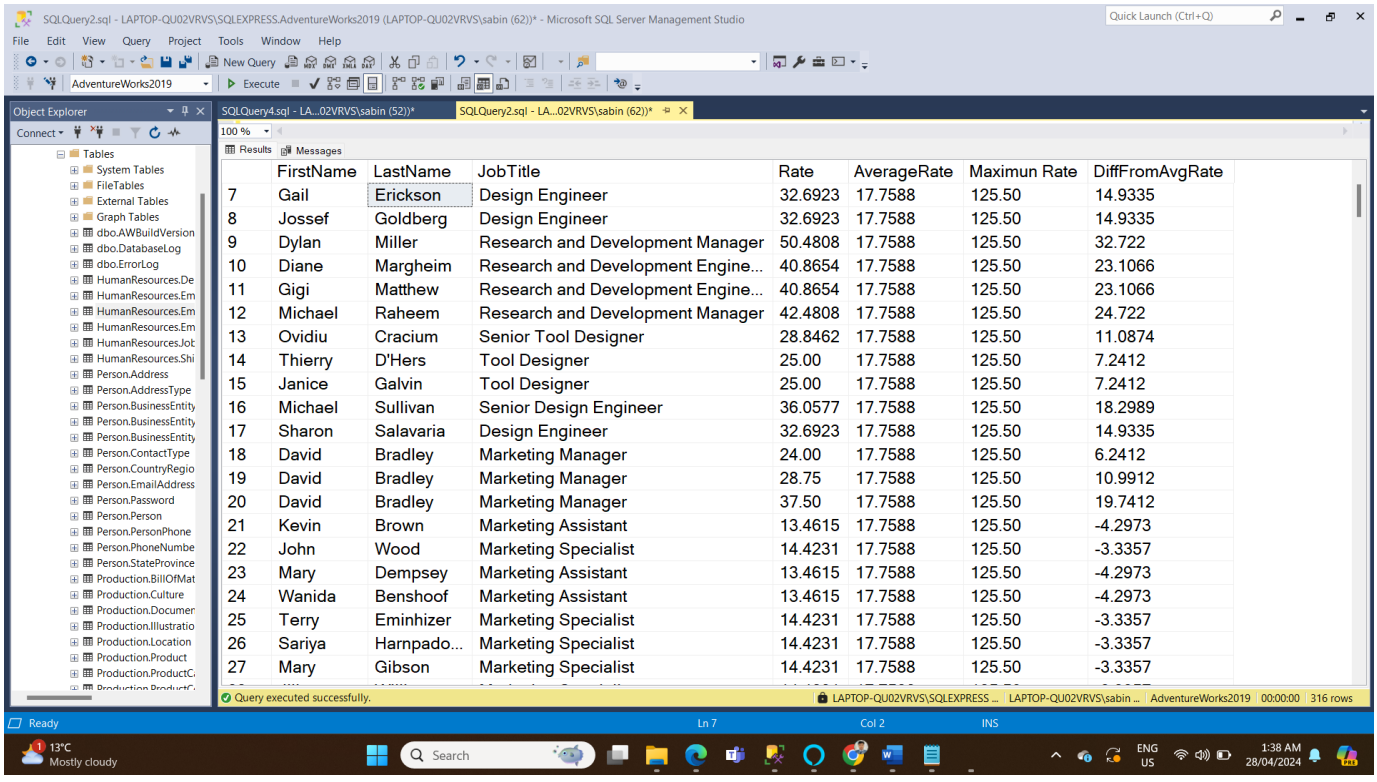
ON HREPH.BusinessEntityID = PP.BusinessEntityID

JOIN AdventureWorks2019.HumanResources.Employee HRE

ON HREPH.BusinessEntityID = HRE.BusinessEntityID



	FirstName	LastName	JobTitle	Rate	AverageRate	Maximun Rate	DiffFromAvgRate
1	Ken	Sánchez	Chief Executive Officer	125.50	17.7588	125.50	107.7412
2	Terri	Duffy	Vice President of Engineering	63.4615	17.7588	125.50	45.7027
3	Roberto	Tamburello	Engineering Manager	43.2692	17.7588	125.50	25.5104
4	Rob	Walters	Senior Tool Designer	8.62	17.7588	125.50	-9.1388
5	Rob	Walters	Senior Tool Designer	23.72	17.7588	125.50	5.9612
6	Rob	Walters	Senior Tool Designer	29.8462	17.7588	125.50	12.0874
7	Gail	Erickson	Design Engineer	32.6923	17.7588	125.50	14.9335
8	Jossef	Goldberg	Design Engineer	32.6923	17.7588	125.50	14.9335



	FirstName	LastName	JobTitle	Rate	AverageRate	Maximun Rate	DiffFromAvgRate
7	Gail	Erickson	Design Engineer	32.6923	17.7588	125.50	14.9335
8	Jossef	Goldberg	Design Engineer	32.6923	17.7588	125.50	14.9335
9	Dylan	Miller	Research and Development Manager	50.4808	17.7588	125.50	32.722
10	Diane	Margheim	Research and Development Engine...	40.8654	17.7588	125.50	23.1066
11	Gigi	Matthew	Research and Development Engine...	40.8654	17.7588	125.50	23.1066
12	Michael	Raheem	Research and Development Manager	42.4808	17.7588	125.50	24.722
13	Ovidiu	Cracium	Senior Tool Designer	28.8462	17.7588	125.50	11.0874
14	Thierry	D'Hers	Tool Designer	25.00	17.7588	125.50	7.2412
15	Janice	Galvin	Tool Designer	25.00	17.7588	125.50	7.2412
16	Michael	Sullivan	Senior Design Engineer	36.0577	17.7588	125.50	18.2989
17	Sharon	Salavaria	Design Engineer	32.6923	17.7588	125.50	14.9335
18	David	Bradley	Marketing Manager	24.00	17.7588	125.50	6.2412
19	David	Bradley	Marketing Manager	28.75	17.7588	125.50	10.9912
20	David	Bradley	Marketing Manager	37.50	17.7588	125.50	19.7412
21	Kevin	Brown	Marketing Assistant	13.4615	17.7588	125.50	-4.2973
22	John	Wood	Marketing Specialist	14.4231	17.7588	125.50	-3.3357
23	Mary	Dempsey	Marketing Assistant	13.4615	17.7588	125.50	-4.2973
24	Wanida	Benshoof	Marketing Assistant	13.4615	17.7588	125.50	-4.2973
25	Terry	Eminhizer	Marketing Specialist	14.4231	17.7588	125.50	-3.3357
26	Sariya	Harnpado...	Marketing Specialist	14.4231	17.7588	125.50	-3.3357
27	Mary	Gibson	Marketing Specialist	14.4231	17.7588	125.50	-3.3357

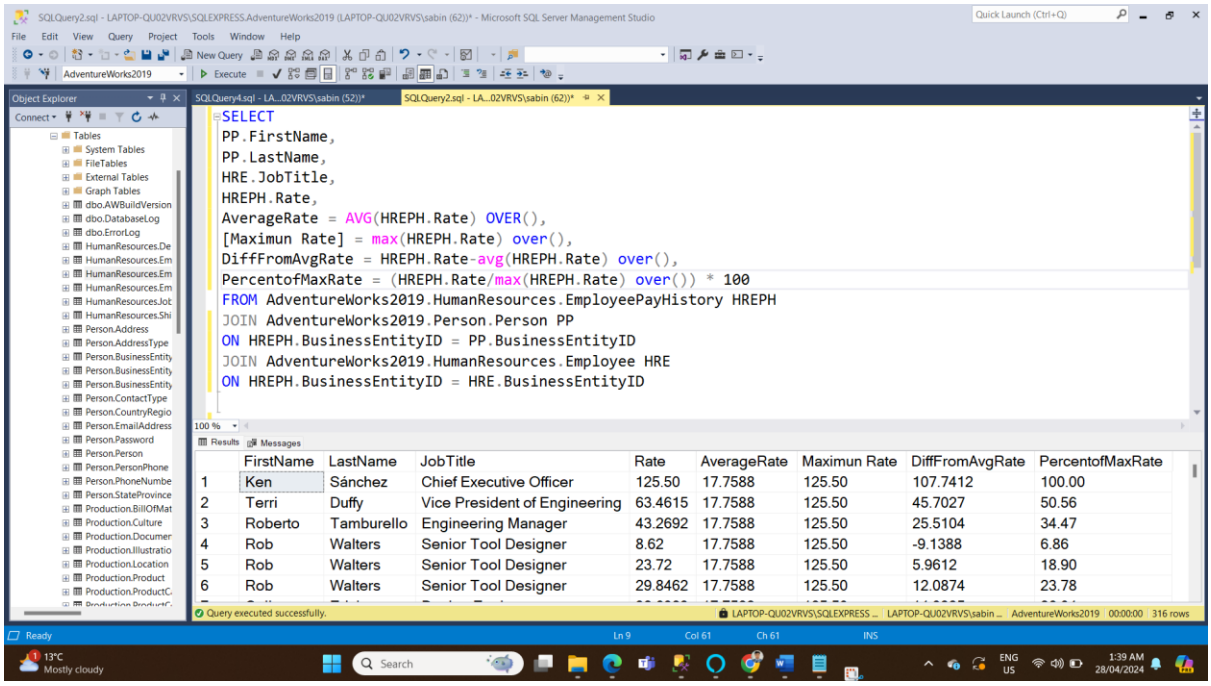
Exercise 4: Enhance your query from Exercise 3 by adding a derived column called

"PercentofMaxRate" that returns the result of the following calculation:

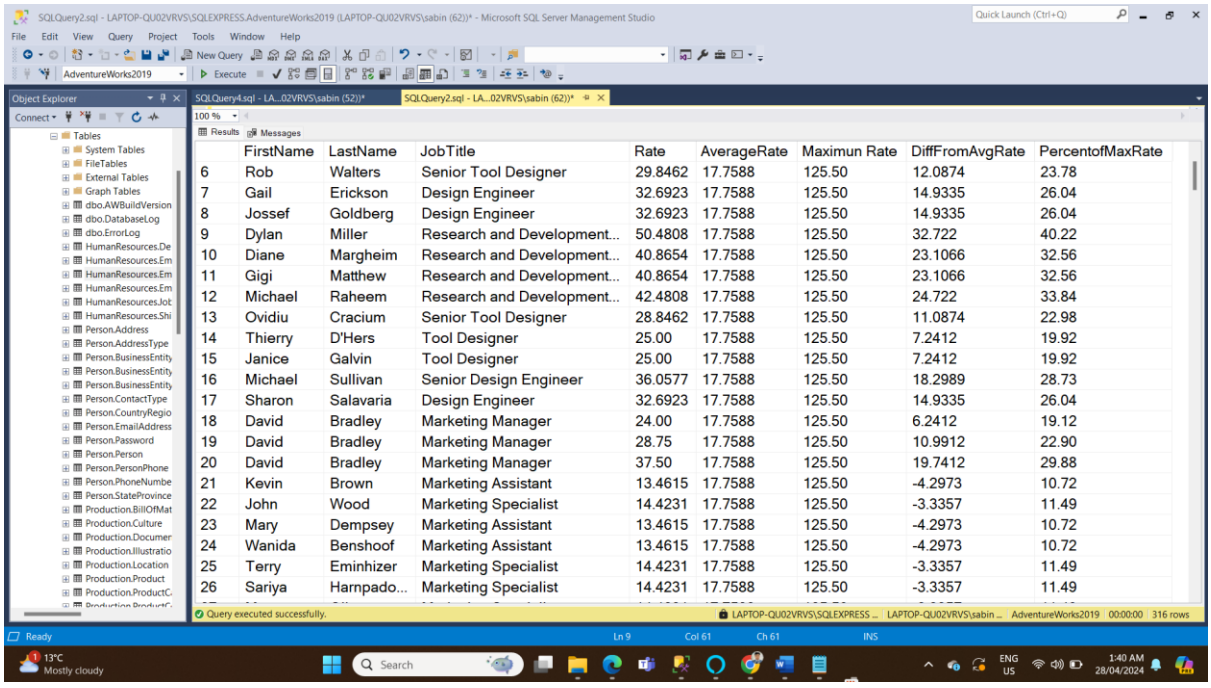
An employees's pay rate, DIVIDED BY the maximum of all values in the "Rate" column, times 100.

Answer:

```
SELECT
PP.FirstName,
PP.LastName,
HRE.JobTitle,
HREPH.Rate,
AverageRate = AVG(HREPH.Rate) OVER(),
[Maximun Rate] = max(HREPH.Rate) over(),
DiffFromAvgRate = HREPH.Rate-avg(HREPH.Rate) over(),
PercentofMaxRate = (HREPH.Rate/max(HREPH.Rate) over()) * 100
FROM AdventureWorks2019.HumanResources.EmployeePayHistory HREPH
JOIN AdventureWorks2019.Person.Person PP
ON HREPH.BusinessEntityID = PP.BusinessEntityID
JOIN AdventureWorks2019.HumanResources.Employee HRE
ON HREPH.BusinessEntityID = HRE.BusinessEntityID
```



	FirstName	LastName	JobTitle	Rate	AverageRate	Maximun Rate	DiffFromAvgRate	PercentofMaxRate
1	Ken	Sánchez	Chief Executive Officer	125.50	17.7588	125.50	107.7412	100.00
2	Terri	Duffy	Vice President of Engineering	63.4615	17.7588	125.50	45.7027	50.56
3	Roberto	Tamburello	Engineering Manager	43.2692	17.7588	125.50	25.5104	34.47
4	Rob	Walters	Senior Tool Designer	8.62	17.7588	125.50	-9.1388	6.86
5	Rob	Walters	Senior Tool Designer	23.72	17.7588	125.50	5.9612	18.90
6	Rob	Walters	Senior Tool Designer	29.8462	17.7588	125.50	12.0874	23.78



	FirstName	LastName	JobTitle	Rate	AverageRate	Maximun Rate	DiffFromAvgRate	PercentofMaxRate
6	Rob	Walters	Senior Tool Designer	29.8462	17.7588	125.50	12.0874	23.78
7	Gail	Erickson	Design Engineer	32.6923	17.7588	125.50	14.9335	26.04
8	Jossef	Goldberg	Design Engineer	32.6923	17.7588	125.50	14.9335	26.04
9	Dylan	Miller	Research and Development...	50.4808	17.7588	125.50	32.722	40.22
10	Diane	Margheim	Research and Development...	40.8654	17.7588	125.50	23.1066	32.56
11	Gigi	Matthew	Research and Development...	40.8654	17.7588	125.50	23.1066	32.56
12	Michael	Raheem	Research and Development...	42.4808	17.7588	125.50	24.722	33.84
13	Ovidiu	Cracium	Senior Tool Designer	28.8462	17.7588	125.50	11.0874	22.98
14	Thierry	D'Hers	Tool Designer	25.00	17.7588	125.50	7.2412	19.92
15	Janice	Galvin	Tool Designer	25.00	17.7588	125.50	7.2412	19.92
16	Michael	Sullivan	Senior Design Engineer	36.0577	17.7588	125.50	18.2989	28.73
17	Sharon	Salavaria	Design Engineer	32.6923	17.7588	125.50	14.9335	26.04
18	David	Bradley	Marketing Manager	24.00	17.7588	125.50	6.2412	19.12
19	David	Bradley	Marketing Manager	28.75	17.7588	125.50	10.9912	22.90
20	David	Bradley	Marketing Manager	37.50	17.7588	125.50	19.7412	29.88
21	Kevin	Brown	Marketing Assistant	13.4615	17.7588	125.50	-4.2973	10.72
22	John	Wood	Marketing Specialist	14.4231	17.7588	125.50	-3.3357	11.49
23	Mary	Dempsey	Marketing Assistant	13.4615	17.7588	125.50	-4.2973	10.72
24	Wanida	Benshoof	Marketing Assistant	13.4615	17.7588	125.50	-4.2973	10.72
25	Terry	Eminhizer	Marketing Specialist	14.4231	17.7588	125.50	-3.3357	11.49
26	Sariya	Harnpado...	Marketing Specialist	14.4231	17.7588	125.50	-3.3357	11.49

Exercise 5: Create a query with the following columns:

“Name” from the Production.Product table, which can be alised as “ProductName”.

ListPrice” from the Production.Product table

“Name” from the Production. ProductSubcategory table, which can be alised as “ProductSubcategory”

“Name” from the Production.ProductCategory table, which can be alised as “ProductCategory”

Answer:

select

ProductName = PP.Name,

PP.ListPrice,

ProductSubcategory = PPS.Name,

ProductCategory = PPC.Name

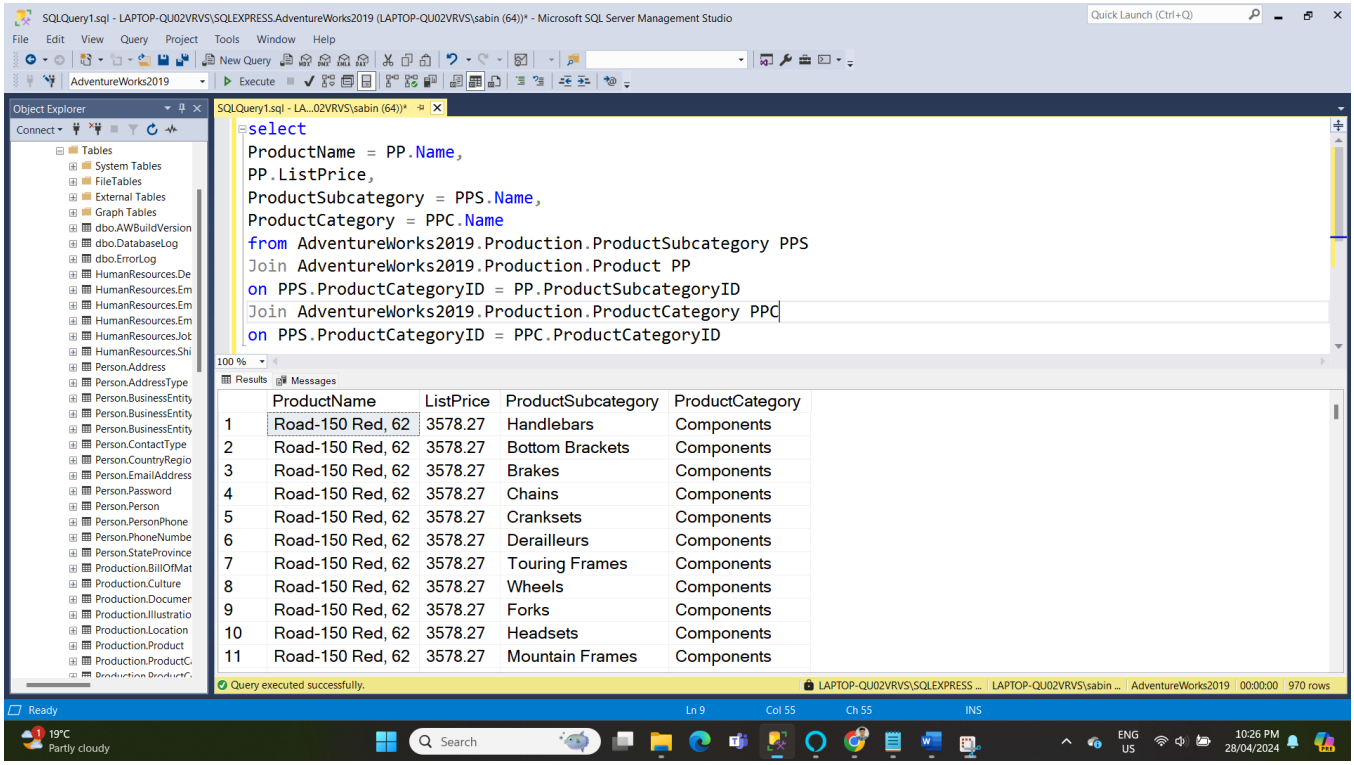
from AdventureWorks2019.Production.ProductSubcategory PPS

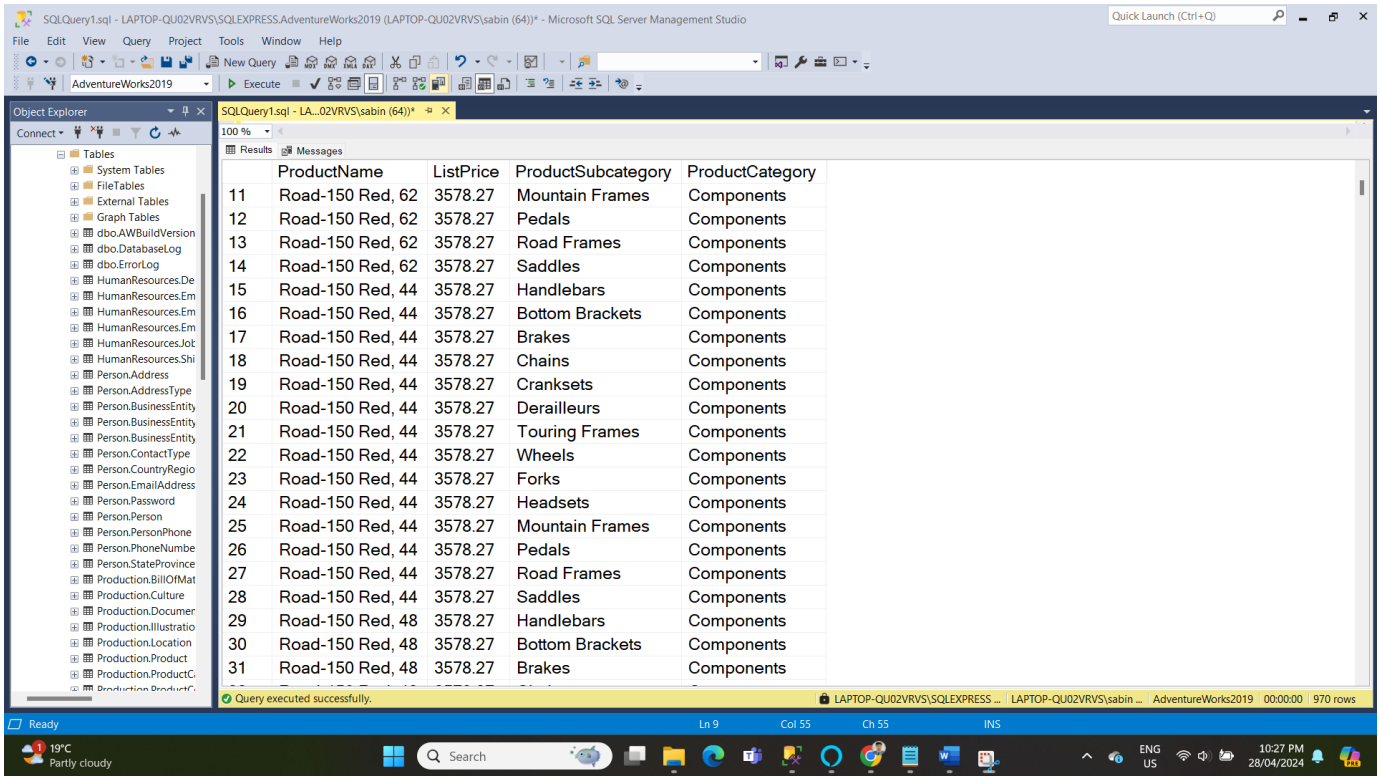
Join AdventureWorks2019.Production.Product PP

on PPS.ProductCategoryID = PP.ProductSubcategoryID

Join AdventureWorks2019.Production.ProductCategory PPC

on PPS.ProductCategoryID = PPC.ProductCategoryID

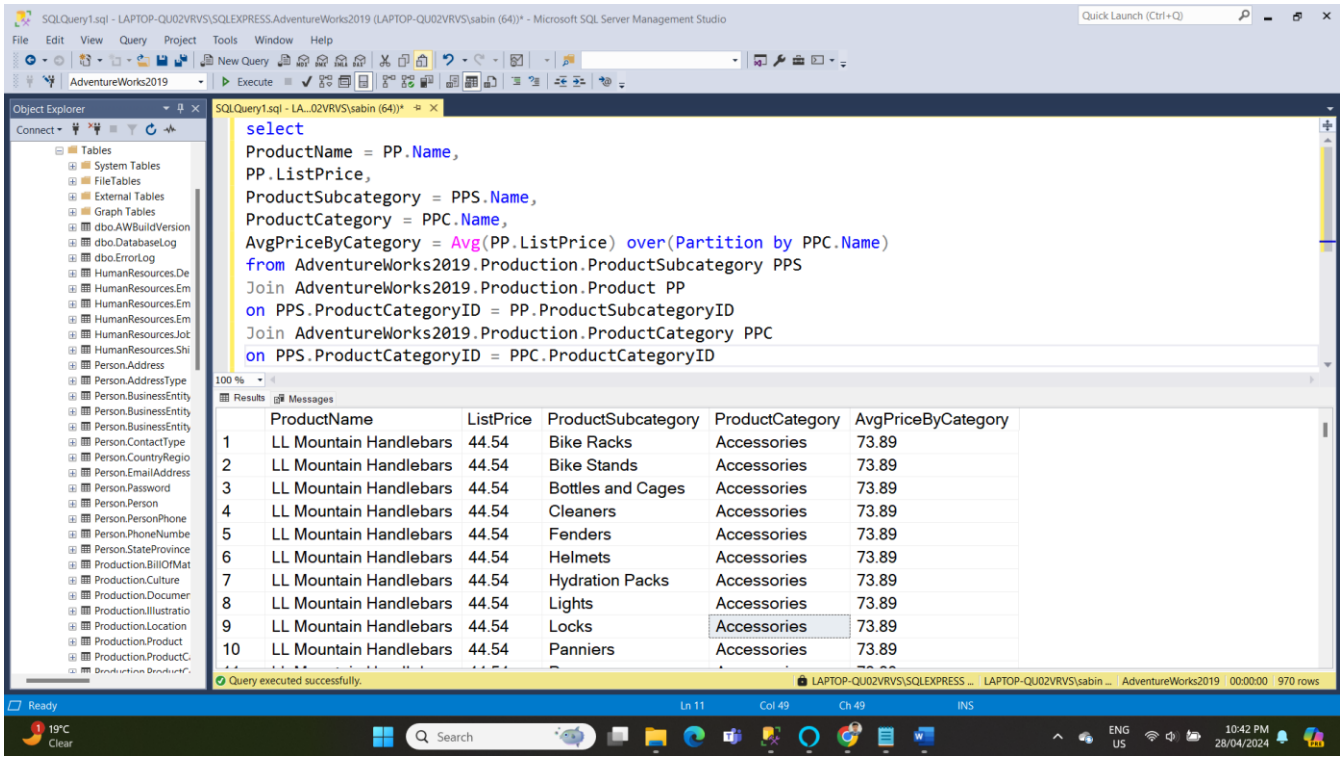




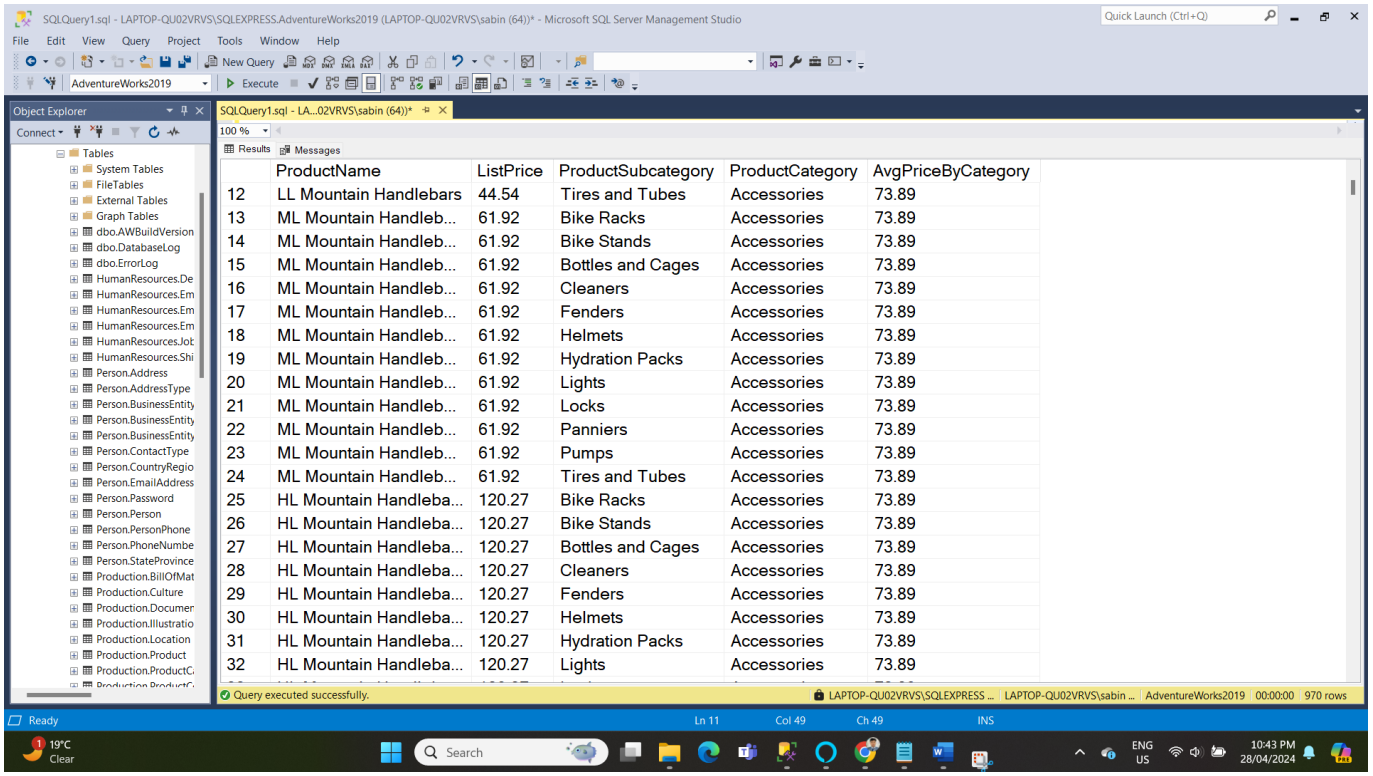
Exercise 6: Enhance your query from Exercise 5 by adding a derived column called "AvgPriceByCategory " that returns the average ListPrice for the product category in each given row.

Answer:

```
select
ProductName = PP.Name,
PP.ListPrice,
ProductSubcategory = PPS.Name,
ProductCategory = PPC.Name,
AvgPriceByCategory = Avg(PP.ListPrice) over(Partition by PPC.Name)
from AdventureWorks2019.Production.ProductSubcategory PPS
Join AdventureWorks2019.Production.Product PP
on PPS.ProductCategoryID = PP.ProductSubcategoryID
Join AdventureWorks2019.Production.ProductCategory PPC
on PPS.ProductCategoryID = PPC.ProductCategoryID
```



	ProductName	ListPrice	ProductSubcategory	ProductCategory	AvgPriceByCategory
1	LL Mountain Handlebars	44.54	Bike Racks	Accessories	73.89
2	LL Mountain Handlebars	44.54	Bike Stands	Accessories	73.89
3	LL Mountain Handlebars	44.54	Bottles and Cages	Accessories	73.89
4	LL Mountain Handlebars	44.54	Cleaners	Accessories	73.89
5	LL Mountain Handlebars	44.54	Fenders	Accessories	73.89
6	LL Mountain Handlebars	44.54	Helmets	Accessories	73.89
7	LL Mountain Handlebars	44.54	Hydration Packs	Accessories	73.89
8	LL Mountain Handlebars	44.54	Lights	Accessories	73.89
9	LL Mountain Handlebars	44.54	Locks	Accessories	73.89
10	LL Mountain Handlebars	44.54	Panniers	Accessories	73.89



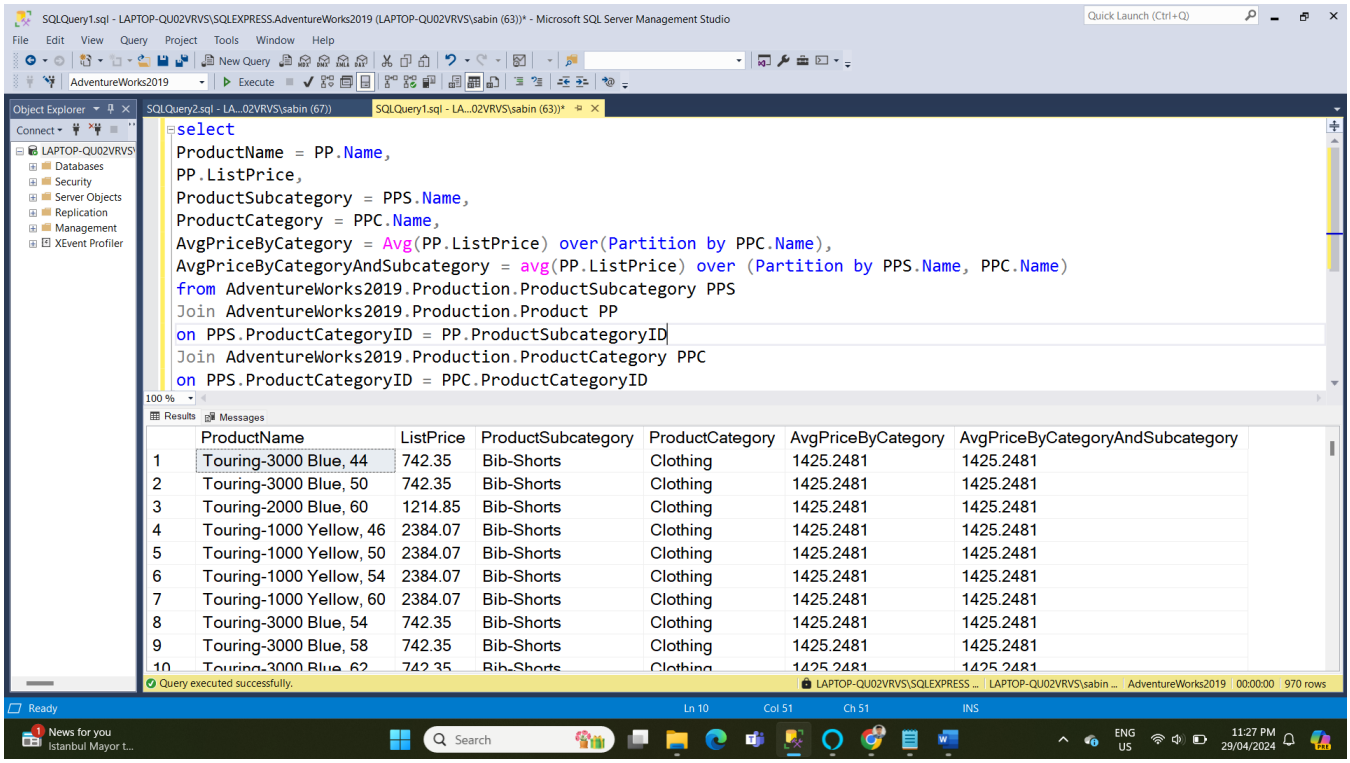
	ProductName	ListPrice	ProductSubcategory	ProductCategory	AvgPriceByCategory
12	LL Mountain Handlebars	44.54	Tires and Tubes	Accessories	73.89
13	ML Mountain Handleb...	61.92	Bike Racks	Accessories	73.89
14	ML Mountain Handleb...	61.92	Bike Stands	Accessories	73.89
15	ML Mountain Handleb...	61.92	Bottles and Cages	Accessories	73.89
16	ML Mountain Handleb...	61.92	Cleaners	Accessories	73.89
17	ML Mountain Handleb...	61.92	Fenders	Accessories	73.89
18	ML Mountain Handleb...	61.92	Helmets	Accessories	73.89
19	ML Mountain Handleb...	61.92	Hydration Packs	Accessories	73.89
20	ML Mountain Handleb...	61.92	Lights	Accessories	73.89
21	ML Mountain Handleb...	61.92	Locks	Accessories	73.89
22	ML Mountain Handleb...	61.92	Panniers	Accessories	73.89
23	ML Mountain Handleb...	61.92	Pumps	Accessories	73.89
24	ML Mountain Handleb...	61.92	Tires and Tubes	Accessories	73.89
25	HL Mountain Handleba...	120.27	Bike Racks	Accessories	73.89
26	HL Mountain Handleba...	120.27	Bike Stands	Accessories	73.89
27	HL Mountain Handleba...	120.27	Bottles and Cages	Accessories	73.89
28	HL Mountain Handleba...	120.27	Cleaners	Accessories	73.89
29	HL Mountain Handleba...	120.27	Fenders	Accessories	73.89
30	HL Mountain Handleba...	120.27	Helmets	Accessories	73.89
31	HL Mountain Handleba...	120.27	Hydration Packs	Accessories	73.89
32	HL Mountain Handleba...	120.27	Lights	Accessories	73.89

Exercise 7: Enhance your query from Exercise 6 by adding a derived column called

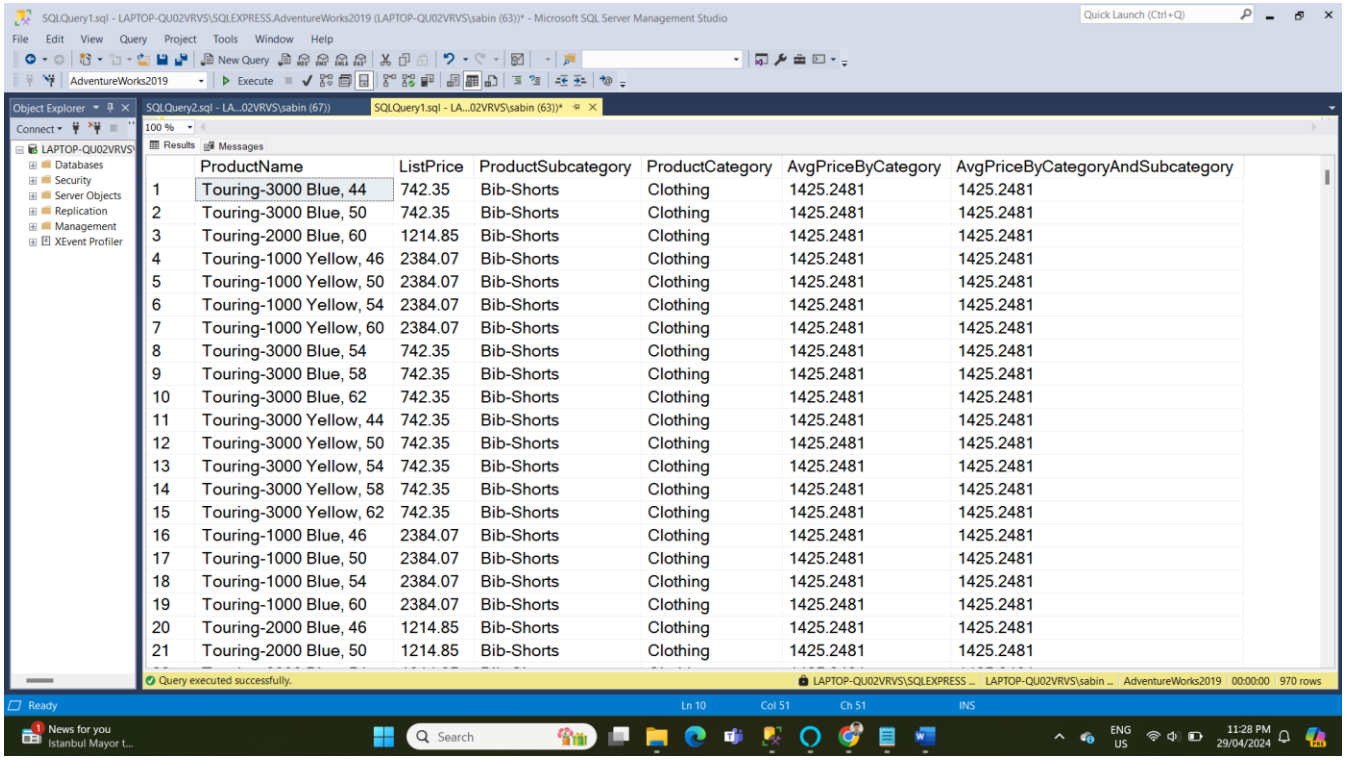
"AvgPriceByCategoryAndSubcategory" that returns the average ListPrice for the product category AND subcategory in each given row.

Answer:

```
select

ProductName = PP.Name,

PP.ListPrice,

ProductSubcategory = PPS.Name,

ProductCategory = PPC.Name,

AvgPriceByCategory = Avg(PP.ListPrice) over(Partition by PPC.Name),

AvgPriceByCategoryAndSubcategory = avg(PP.ListPrice) over (Partition by
PPS.Name, PPC.Name)

from AdventureWorks2019.Production.ProductSubcategory PPS

Join AdventureWorks2019.Production.Product PP

on PPS.ProductCategoryID = PP.ProductSubcategoryID

Join AdventureWorks2019.Production.ProductCategory PPC

on PPS.ProductCategoryID = PPC.ProductCategoryID
```



	ProductName	ListPrice	ProductSubcategory	ProductCategory	AvgPriceByCategory	AvgPriceByCategoryAndSubcategory
1	Touring-3000 Blue, 44	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
2	Touring-3000 Blue, 50	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
3	Touring-2000 Blue, 60	1214.85	Bib-Shorts	Clothing	1425.2481	1425.2481
4	Touring-1000 Yellow, 46	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
5	Touring-1000 Yellow, 50	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
6	Touring-1000 Yellow, 54	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
7	Touring-1000 Yellow, 60	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
8	Touring-3000 Blue, 54	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
9	Touring-3000 Blue, 58	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
10	Touring-3000 Blue, 62	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481



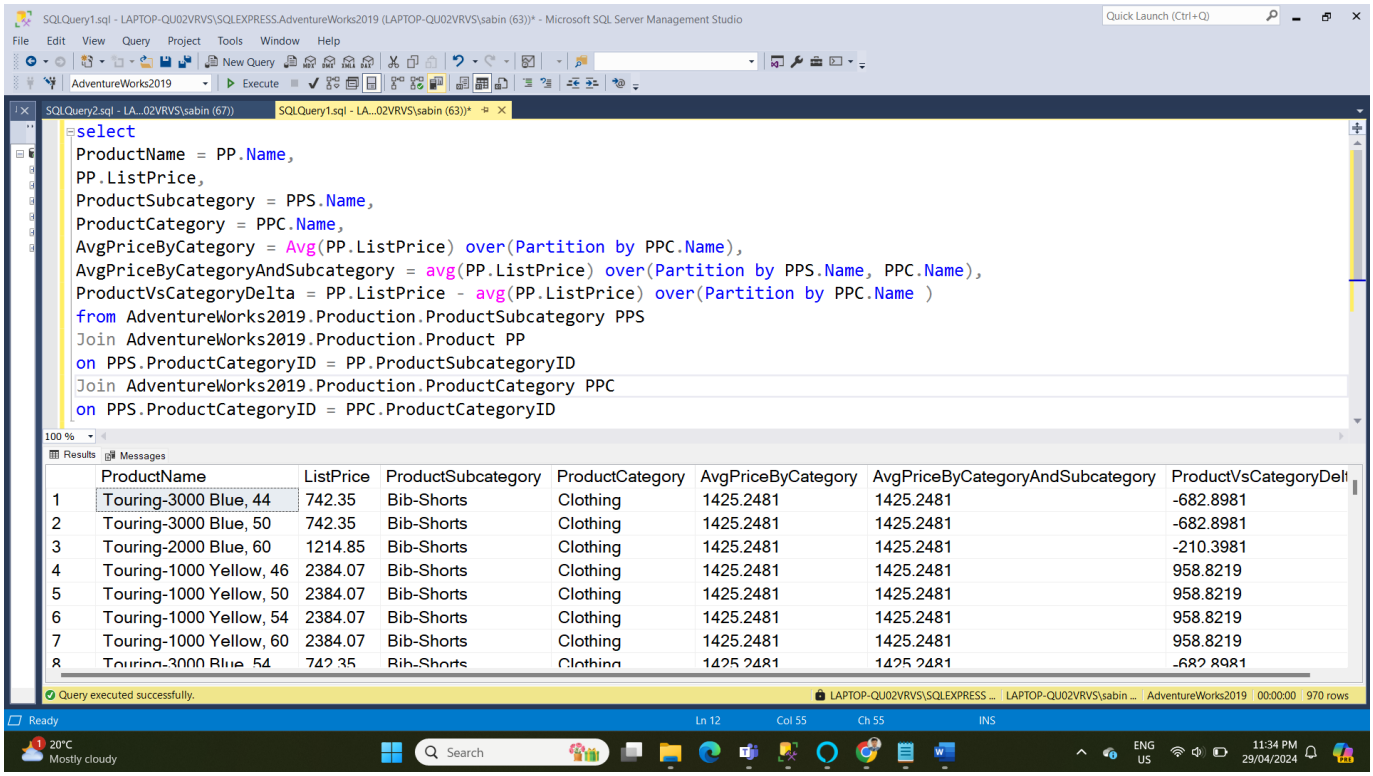
	ProductName	ListPrice	ProductSubcategory	ProductCategory	AvgPriceByCategory	AvgPriceByCategoryAndSubcategory
1	Touring-3000 Blue, 44	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
2	Touring-3000 Blue, 50	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
3	Touring-2000 Blue, 60	1214.85	Bib-Shorts	Clothing	1425.2481	1425.2481
4	Touring-1000 Yellow, 46	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
5	Touring-1000 Yellow, 50	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
6	Touring-1000 Yellow, 54	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
7	Touring-1000 Yellow, 60	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
8	Touring-3000 Blue, 54	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
9	Touring-3000 Blue, 58	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
10	Touring-3000 Blue, 62	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
11	Touring-3000 Yellow, 44	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
12	Touring-3000 Yellow, 50	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
13	Touring-3000 Yellow, 54	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
14	Touring-3000 Yellow, 58	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
15	Touring-3000 Yellow, 62	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481
16	Touring-1000 Blue, 46	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
17	Touring-1000 Blue, 50	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
18	Touring-1000 Blue, 54	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
19	Touring-1000 Blue, 60	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481
20	Touring-2000 Blue, 46	1214.85	Bib-Shorts	Clothing	1425.2481	1425.2481
21	Touring-2000 Blue, 50	1214.85	Bib-Shorts	Clothing	1425.2481	1425.2481

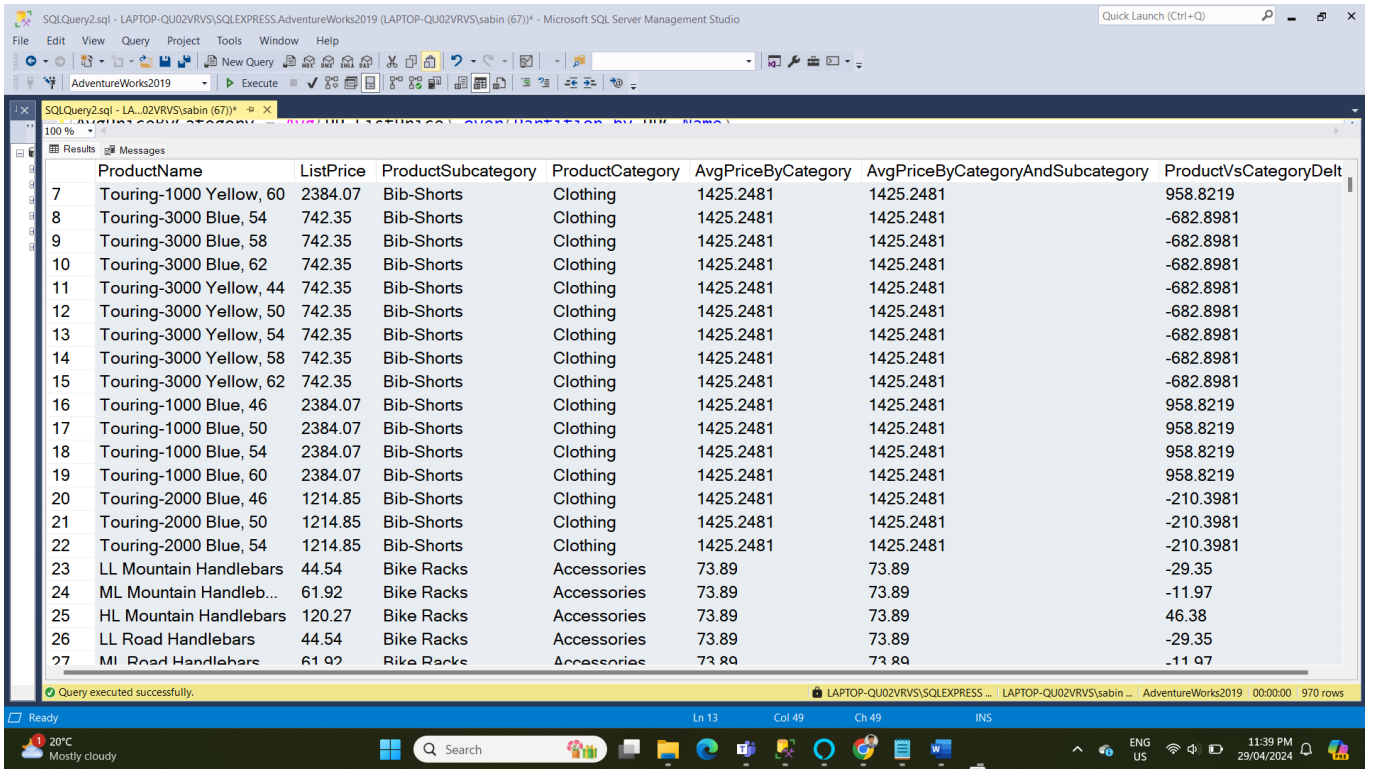
Exercise 8: Enhance your query from Exercise 7 by adding a derived column called" ProductVsCategoryDelta" that returns the result of the following calculation:

A product's list price, MINUS the average ListPrice for that product's category.

Answer:

```
select
ProductName = PP.Name,
PP.ListPrice,
ProductSubcategory = PPS.Name,
ProductCategory = PPC.Name,
AvgPriceByCategory = Avg(PP.ListPrice) over(Partition by PPC.Name),
AvgPriceByCategoryAndSubcategory = avg(PP.ListPrice) over(Partition by PPS.Name, PPC.Name),
ProductVsCategoryDelta = PP.ListPrice - avg(PP.ListPrice) over(Partition by PPC.Name )
from AdventureWorks2019.Production.ProductSubcategory PPS
Join AdventureWorks2019.Production.Product PP
on PPS.ProductCategoryID = PP.ProductSubcategoryID
Join AdventureWorks2019.Production.ProductCategory PPC
on PPS.ProductCategoryID = PPC.ProductCategoryID
```





Exercise 9: Create a query with the following columns:

“Name” from the Production.Product table, which can be alised as “ProductName”.

“ListPrice” from the Production.Product table.

“Name” from the Production. ProductSubcategory table, which can be alised as “ProductSubcategory”.

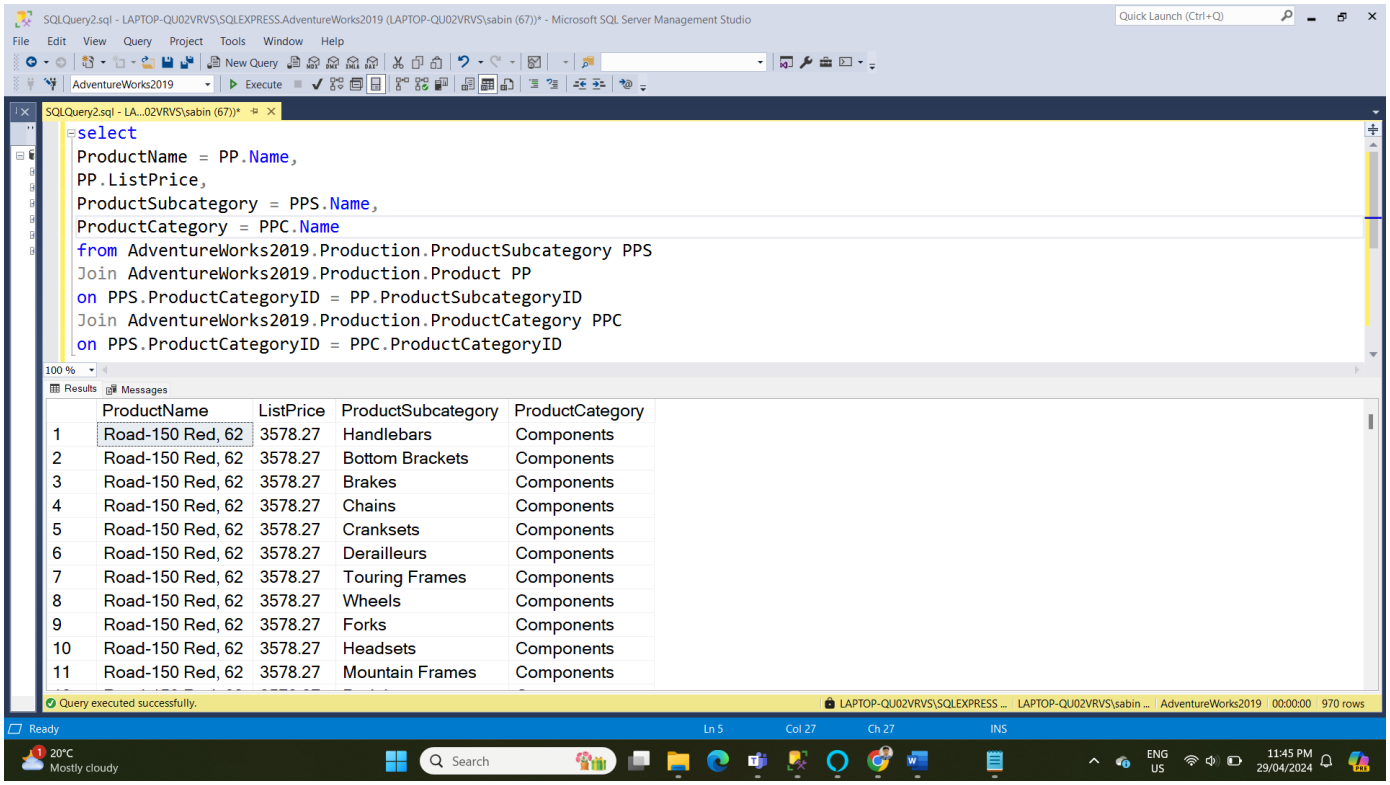
“Name” from the Production.ProductCategory table, which can be alised as “ProductCategory”.

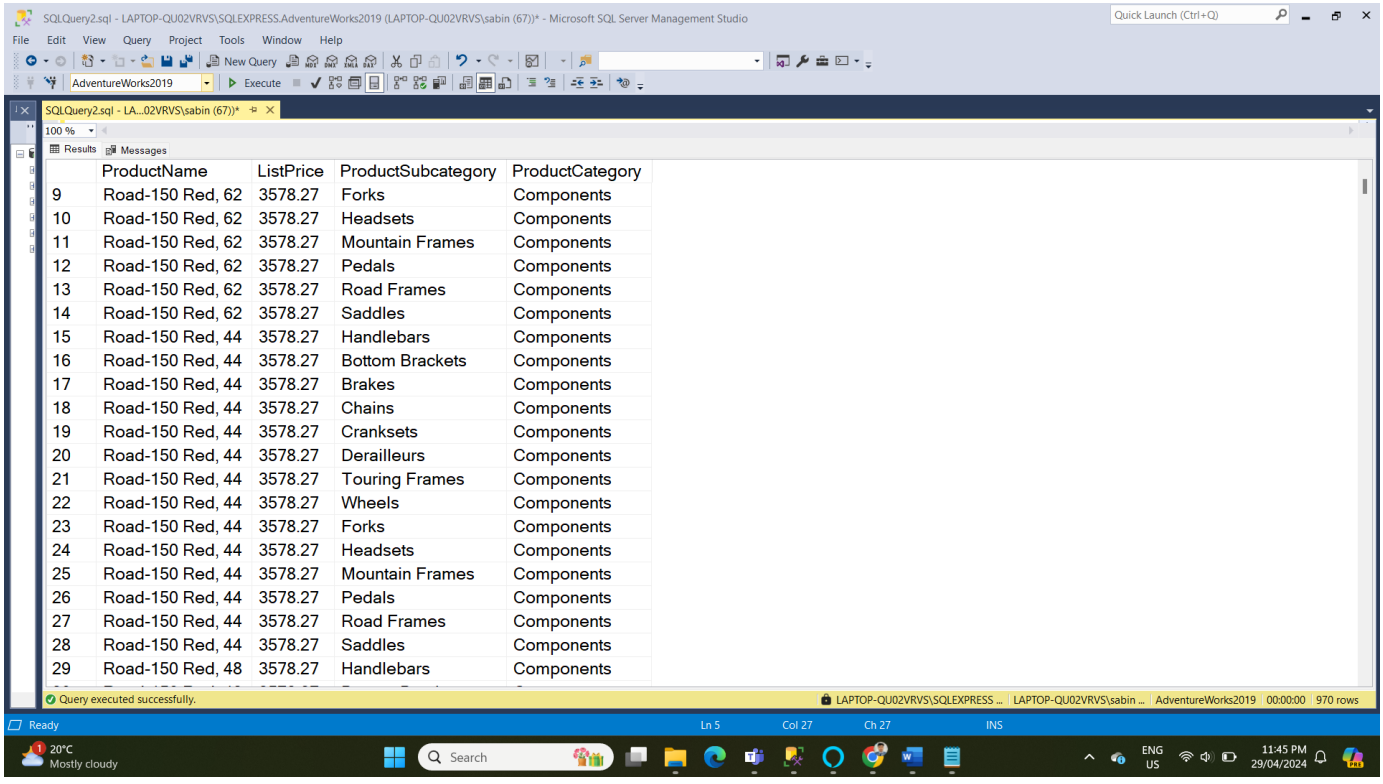
Join Production.ProductSubcategory to Production.Product on “ProductSubcategoryID”.

Join Production.ProductCategory to ProductSubcategory on “ProductCategoryID”.

Answer:

```
select

ProductName = PP.Name,

PP.ListPrice,

ProductSubcategory = PPS.Name,

ProductCategory = PPC.Name

from AdventureWorks2019.Production.ProductSubcategory PPS

Join AdventureWorks2019.Production.Product PP

on PPS.ProductCategoryID = PP.ProductSubcategoryID

Join AdventureWorks2019.Production.ProductCategory PPC

on PPS.ProductCategoryID = PPC.ProductCategoryID
```



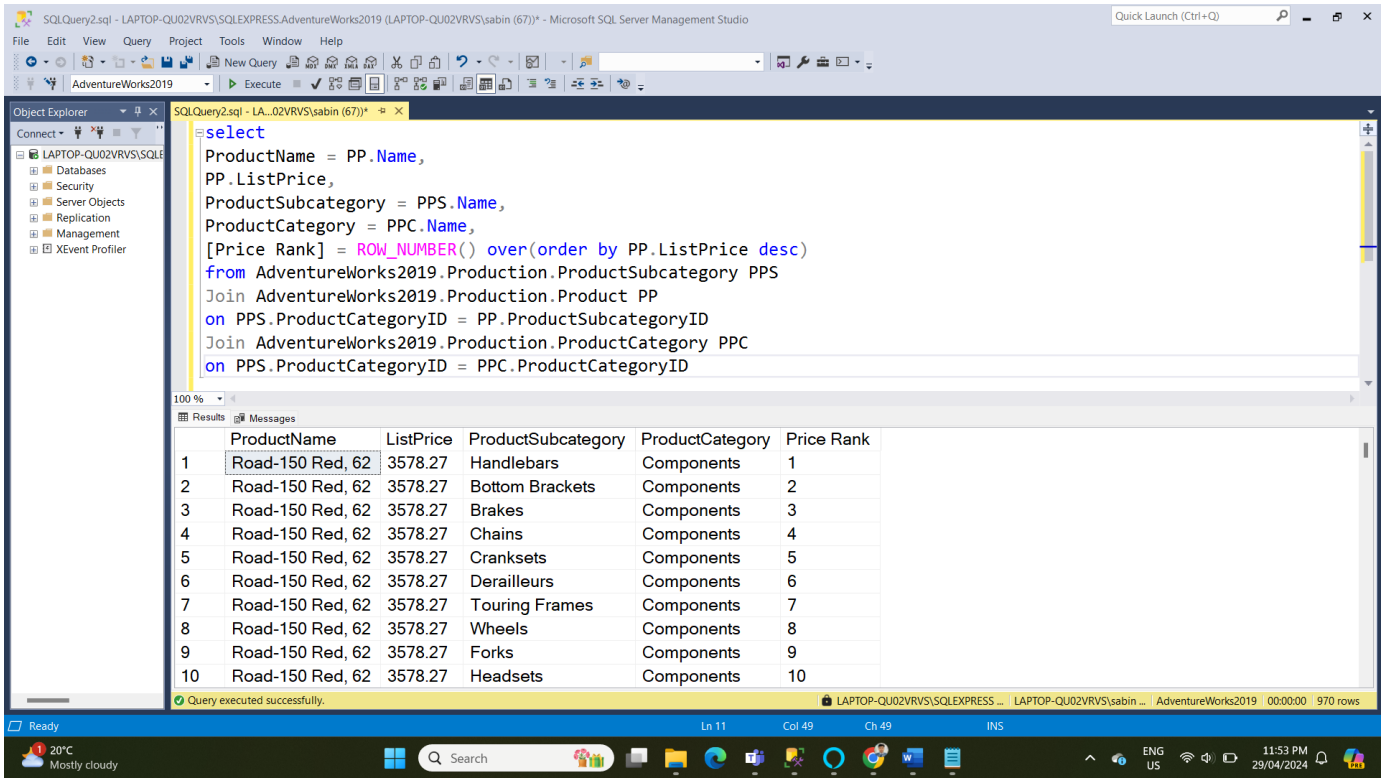


Exercise 10: Enhance your query from Exercise 9 by adding a derived column called "Price Rank " that ranks all records in the dataset by ListPrice, in descending order. That is to say, the product with the most expensive price

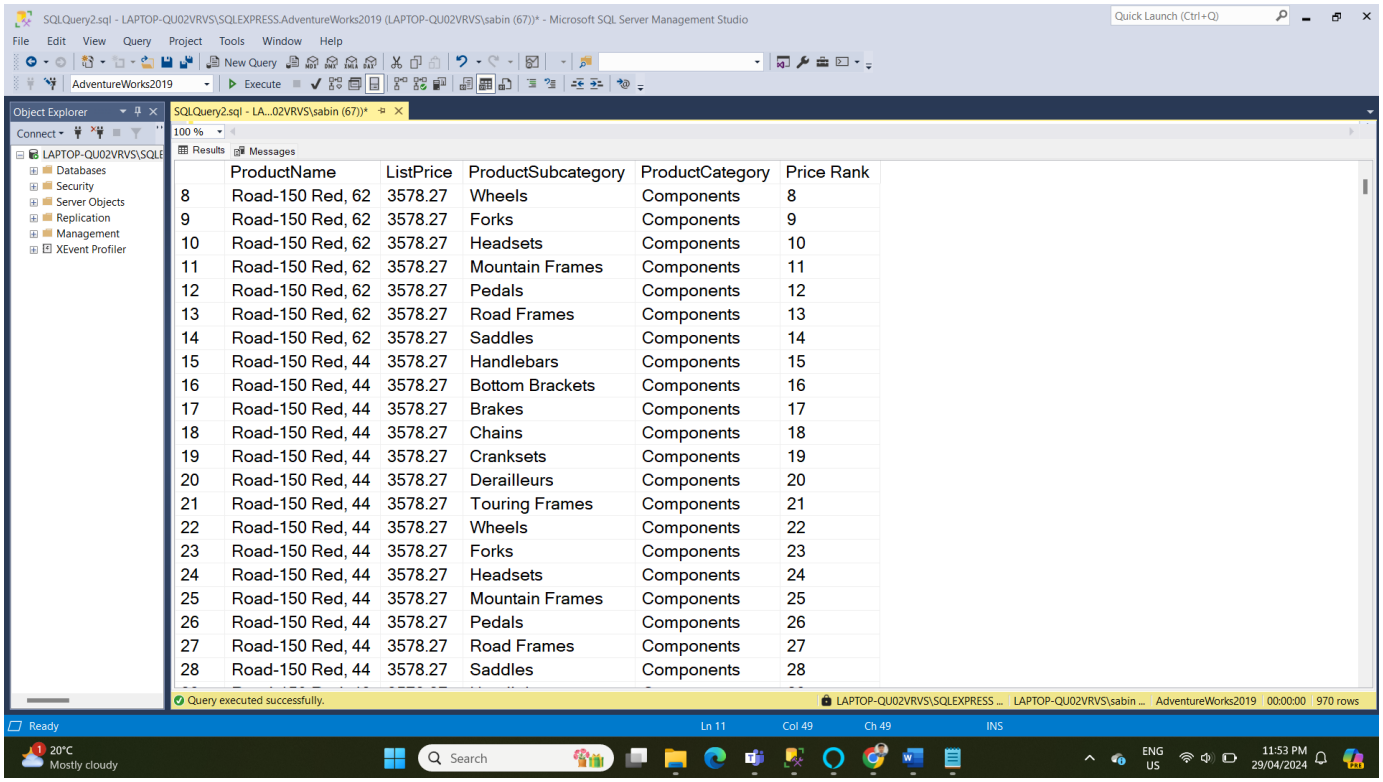
should have a rank of 1, and the product with the least expensive price should have a rank equal to the number of records in the dataset.

Answer:

```
select
ProductName = PP.Name,
PP.ListPrice,
ProductSubcategory = PPS.Name,
ProductCategory = PPC.Name,
[Price Rank] = ROW_NUMBER() over(order by PP.ListPrice desc)
from AdventureWorks2019.Production.ProductSubcategory PPS
Join AdventureWorks2019.Production.Product PP
on PPS.ProductCategoryID = PP.ProductSubcategoryID
Join AdventureWorks2019.Production.ProductCategory PPC
on PPS.ProductCategoryID = PPC.ProductCategoryID
```



	ProductName	ListPrice	ProductSubcategory	ProductCategory	Price Rank
1	Road-150 Red, 62	3578.27	Handlebars	Components	1
2	Road-150 Red, 62	3578.27	Bottom Brackets	Components	2
3	Road-150 Red, 62	3578.27	Brakes	Components	3
4	Road-150 Red, 62	3578.27	Chains	Components	4
5	Road-150 Red, 62	3578.27	Cranksets	Components	5
6	Road-150 Red, 62	3578.27	Derailleurs	Components	6
7	Road-150 Red, 62	3578.27	Touring Frames	Components	7
8	Road-150 Red, 62	3578.27	Wheels	Components	8
9	Road-150 Red, 62	3578.27	Forks	Components	9
10	Road-150 Red, 62	3578.27	Headsets	Components	10



	ProductName	ListPrice	ProductSubcategory	ProductCategory	Price Rank
8	Road-150 Red, 62	3578.27	Wheels	Components	8
9	Road-150 Red, 62	3578.27	Forks	Components	9
10	Road-150 Red, 62	3578.27	Headsets	Components	10
11	Road-150 Red, 62	3578.27	Mountain Frames	Components	11
12	Road-150 Red, 62	3578.27	Pedals	Components	12
13	Road-150 Red, 62	3578.27	Road Frames	Components	13
14	Road-150 Red, 62	3578.27	Saddles	Components	14
15	Road-150 Red, 44	3578.27	Handlebars	Components	15
16	Road-150 Red, 44	3578.27	Bottom Brackets	Components	16
17	Road-150 Red, 44	3578.27	Brakes	Components	17
18	Road-150 Red, 44	3578.27	Chains	Components	18
19	Road-150 Red, 44	3578.27	Cranksets	Components	19
20	Road-150 Red, 44	3578.27	Derailleurs	Components	20
21	Road-150 Red, 44	3578.27	Touring Frames	Components	21
22	Road-150 Red, 44	3578.27	Wheels	Components	22
23	Road-150 Red, 44	3578.27	Forks	Components	23
24	Road-150 Red, 44	3578.27	Headsets	Components	24
25	Road-150 Red, 44	3578.27	Mountain Frames	Components	25
26	Road-150 Red, 44	3578.27	Pedals	Components	26
27	Road-150 Red, 44	3578.27	Road Frames	Components	27
28	Road-150 Red, 44	3578.27	Saddles	Components	28

Exercise 11: Enhance your query from Exercise 10 by adding a derived column called "Category Price Rank" that ranks all products by ListPrice – within each category - in descending order. In other words, every product within a given category should be ranked relative to other products in the same category.

Answer:

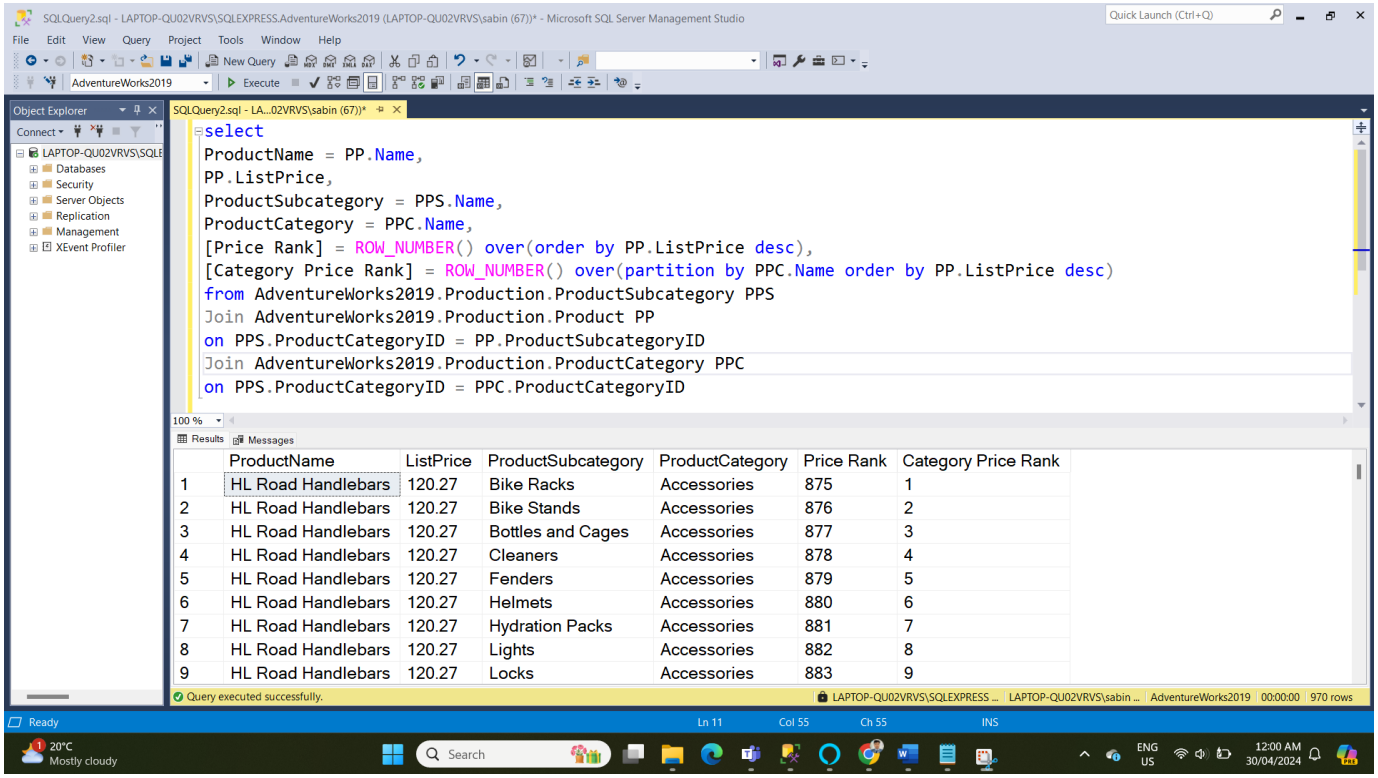
```
select

ProductName = PP.Name,

PP.ListPrice,

ProductSubcategory = PPS.Name,

ProductCategory = PPC.Name,

[Price Rank] = ROW_NUMBER() over(order by PP.ListPrice desc),

[Category Price Rank] = ROW_NUMBER() over(partition by PPC.Name order by PP.ListPrice desc)

from AdventureWorks2019.Production.ProductSubcategory PPS
```

Join AdventureWorks2019.Production.Product PP

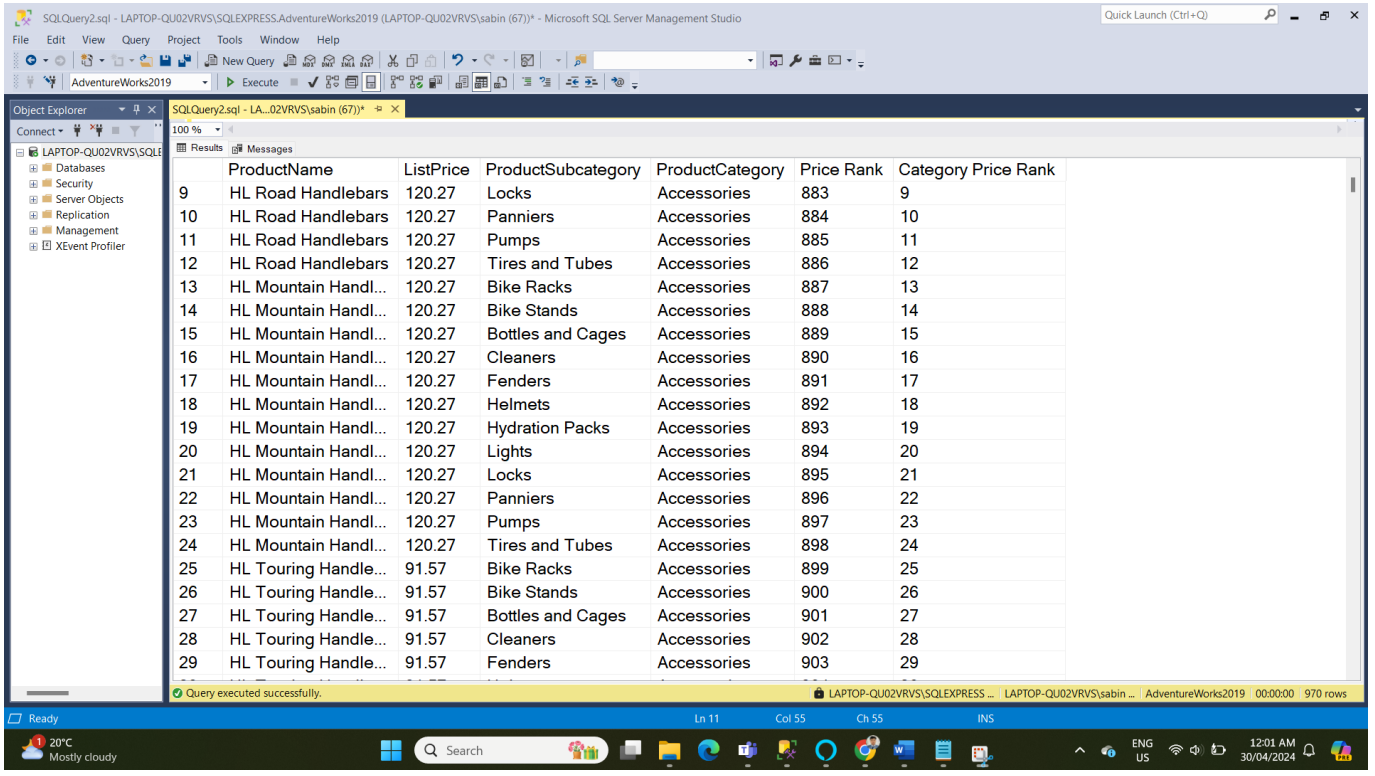
on PPS.ProductCategoryID = PP.ProductSubcategoryID

Join AdventureWorks2019.Production.ProductCategory PPC

on PPS.ProductCategoryID = PPC.ProductCategoryID



	ProductName	ListPrice	ProductSubcategory	ProductCategory	Price Rank	Category Price Rank
1	HL Road Handlebars	120.27	Bike Racks	Accessories	875	1
2	HL Road Handlebars	120.27	Bike Stands	Accessories	876	2
3	HL Road Handlebars	120.27	Bottles and Cages	Accessories	877	3
4	HL Road Handlebars	120.27	Cleaners	Accessories	878	4
5	HL Road Handlebars	120.27	Fenders	Accessories	879	5
6	HL Road Handlebars	120.27	Helmets	Accessories	880	6
7	HL Road Handlebars	120.27	Hydration Packs	Accessories	881	7
8	HL Road Handlebars	120.27	Lights	Accessories	882	8
9	HL Road Handlebars	120.27	Locks	Accessories	883	9

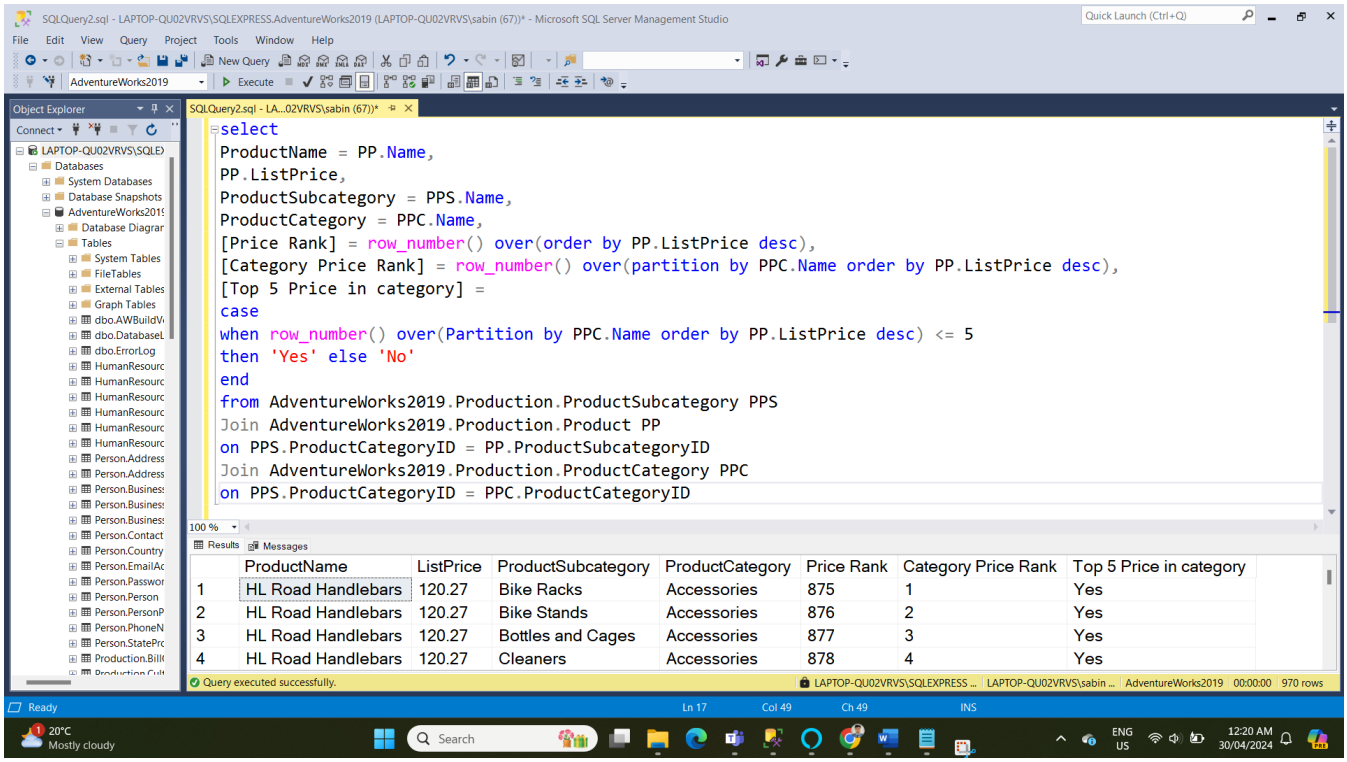


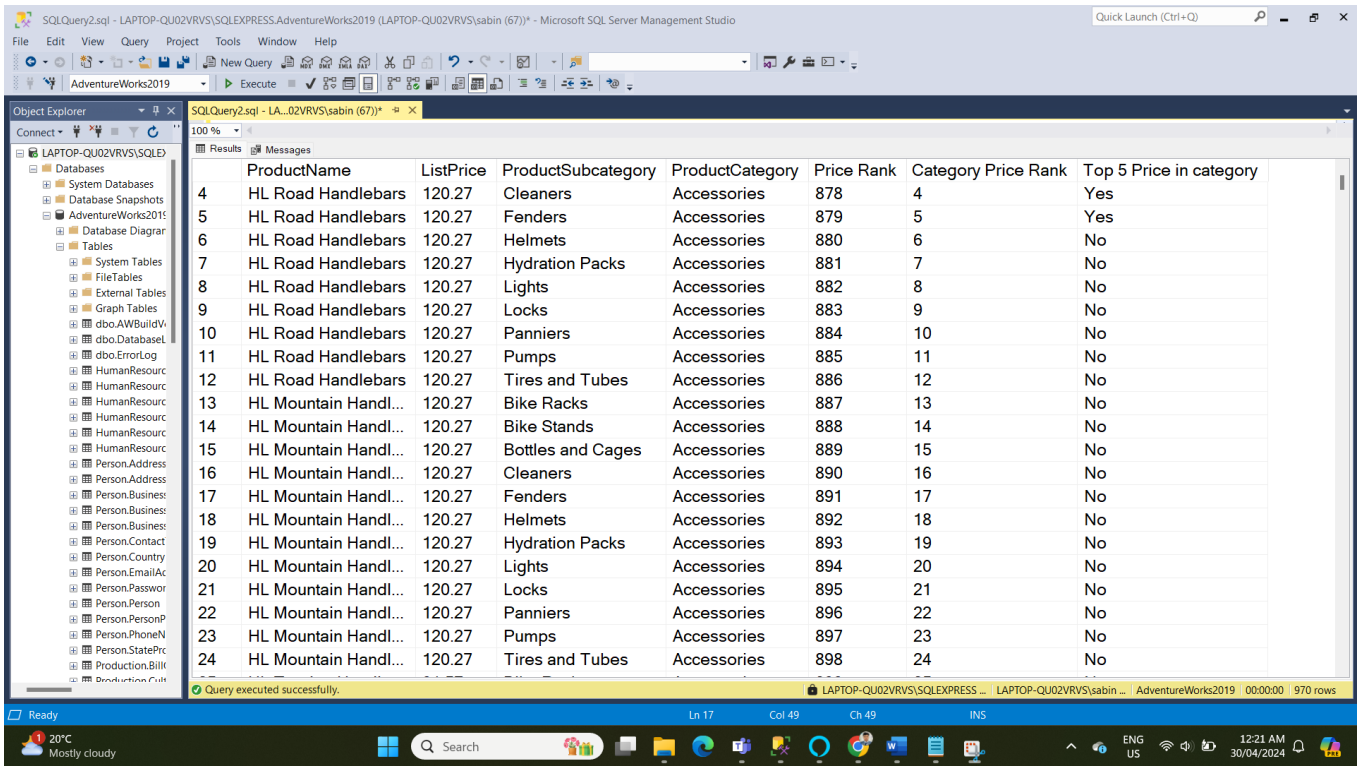
	ProductName	ListPrice	ProductSubcategory	ProductCategory	Price Rank	Category Price Rank
9	HL Road Handlebars	120.27	Locks	Accessories	883	9
10	HL Road Handlebars	120.27	Panniers	Accessories	884	10
11	HL Road Handlebars	120.27	Pumps	Accessories	885	11
12	HL Road Handlebars	120.27	Tires and Tubes	Accessories	886	12
13	HL Mountain Handl...	120.27	Bike Racks	Accessories	887	13
14	HL Mountain Handl...	120.27	Bike Stands	Accessories	888	14
15	HL Mountain Handl...	120.27	Bottles and Cages	Accessories	889	15
16	HL Mountain Handl...	120.27	Cleaners	Accessories	890	16
17	HL Mountain Handl...	120.27	Fenders	Accessories	891	17
18	HL Mountain Handl...	120.27	Helmets	Accessories	892	18
19	HL Mountain Handl...	120.27	Hydration Packs	Accessories	893	19
20	HL Mountain Handl...	120.27	Lights	Accessories	894	20
21	HL Mountain Handl...	120.27	Locks	Accessories	895	21
22	HL Mountain Handl...	120.27	Panniers	Accessories	896	22
23	HL Mountain Handl...	120.27	Pumps	Accessories	897	23
24	HL Mountain Handl...	120.27	Tires and Tubes	Accessories	898	24
25	HL Touring Handle...	91.57	Bike Racks	Accessories	899	25
26	HL Touring Handle...	91.57	Bike Stands	Accessories	900	26
27	HL Touring Handle...	91.57	Bottles and Cages	Accessories	901	27
28	HL Touring Handle...	91.57	Cleaners	Accessories	902	28
29	HL Touring Handle...	91.57	Fenders	Accessories	903	29

Exercise 12: Enhance your query from Exercise 11 by adding a derived column called "Top 5 Price In Category" that returns the string “Yes” if a product has one of the top 5 list prices in its product category, and “No” if it does not. You can try incorporating your logic from Exercise 3 into a CASE statement to make this work.

Answer:

```
select
ProductName = PP.Name,
PP.ListPrice,
ProductSubcategory = PPS.Name,
ProductCategory = PPC.Name,
[Price Rank] = row_number() over(order by PP.ListPrice desc),
[Category Price Rank] = row_number() over(partition by PPC.Name order by PP.ListPrice desc),
[Top 5 Price in category] =
case
when row_number() over(Partition by PPC.Name order by PP.ListPrice desc) <= 5
then 'Yes' else 'No'
end
from AdventureWorks2019.Production.ProductSubcategory PPS
Join AdventureWorks2019.Production.Product PP
on PPS.ProductCategoryID = PP.ProductSubcategoryID
Join AdventureWorks2019.Production.ProductCategory PPC
on PPS.ProductCategoryID = PPC.ProductCategoryID
```



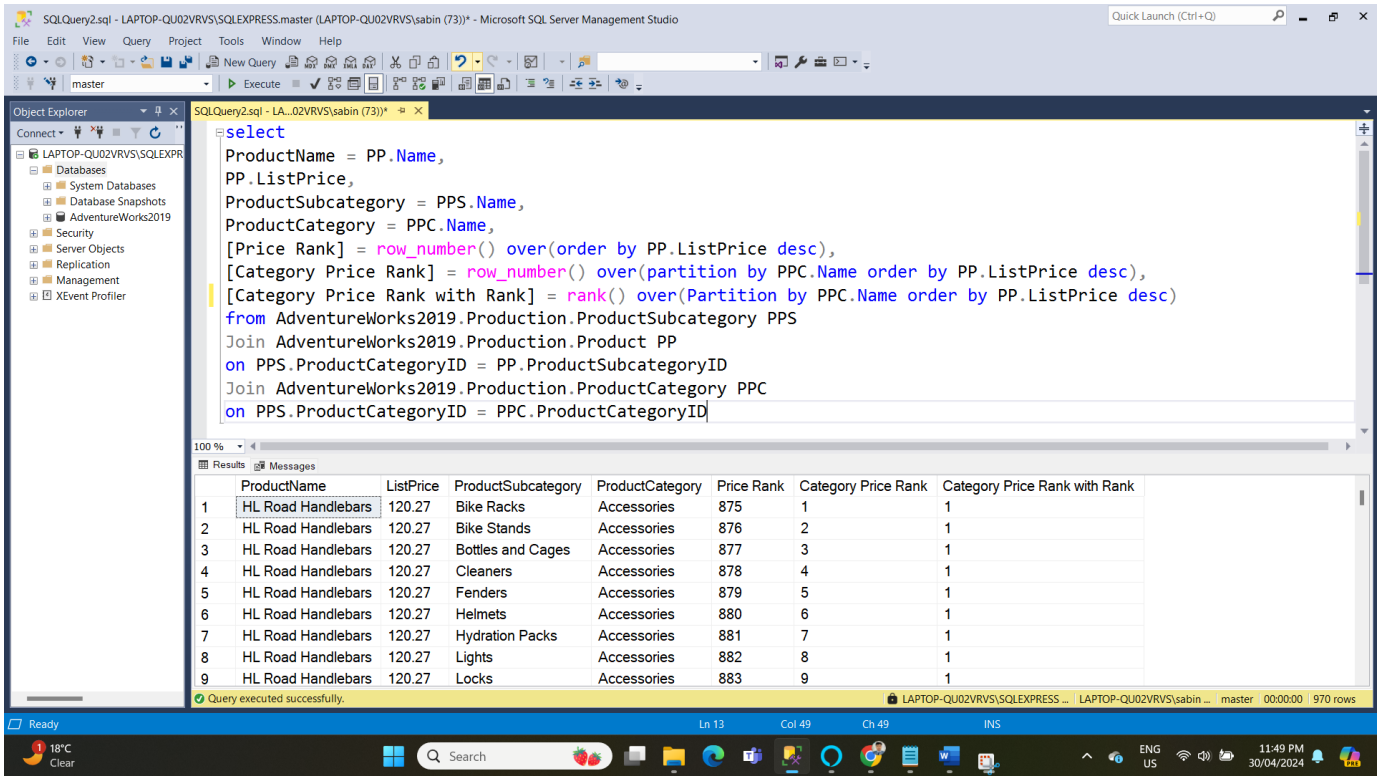


Exercise 13: Using your solution query to Exercise 11 from the ROW_NUMBER exercises as a staring point, add a derived column called "Category Price Rank With Rank" that uses the RANK function to rank all products by ListPrice – within each category - in descending order. Observe

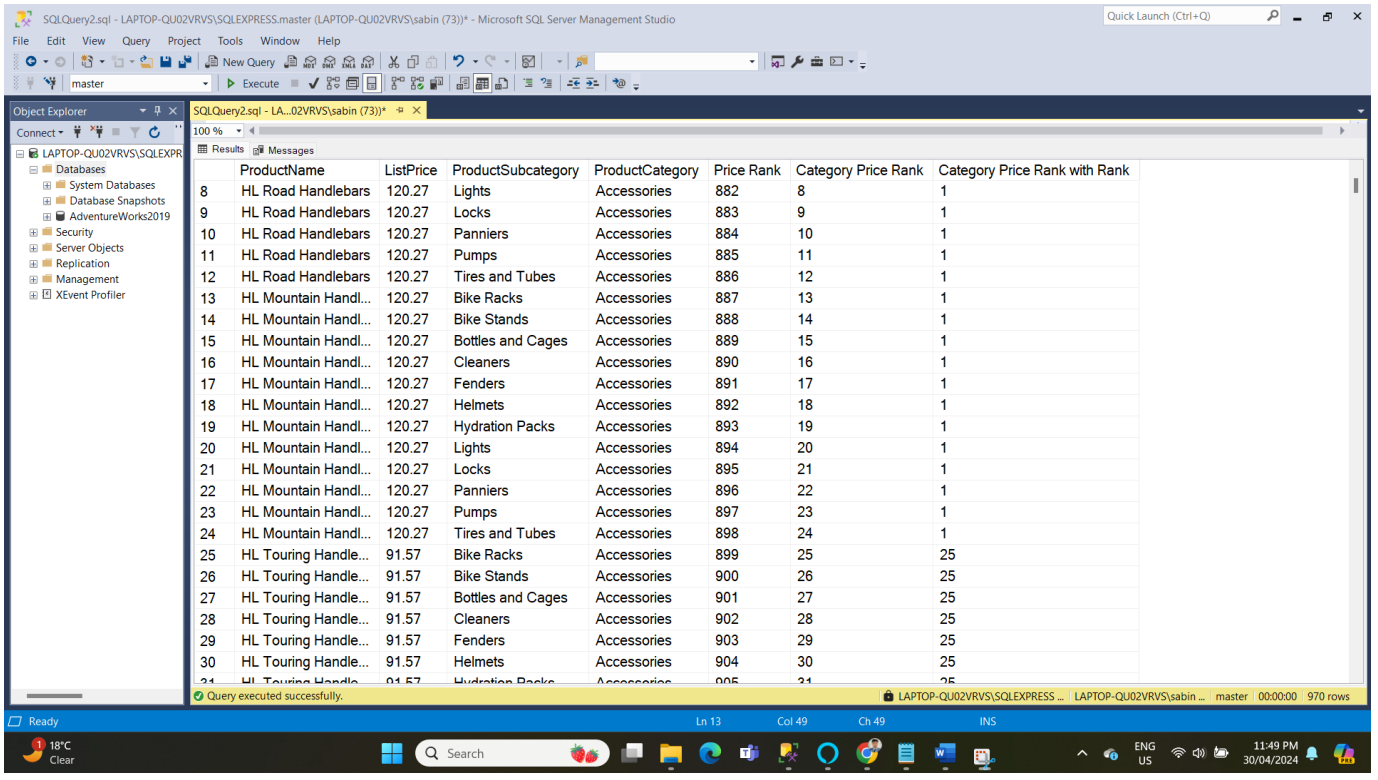
the differences between the “Category Price Rank” and “Category Price Rank With Rank” fields.

Answer:

```
select
ProductName = PP.Name,
PP.ListPrice,
ProductSubcategory = PPS.Name,
ProductCategory = PPC.Name,
[Price Rank] = row_number() over(order by PP.ListPrice desc),
[Category Price Rank] = row_number() over(partition by PPC.Name order by PP.ListPrice desc),
[Category Price Rank with Rank] = rank() over(Partition by PPC.Name order by PP.ListPrice desc)
from AdventureWorks2019.Production.ProductSubcategory PPS
Join AdventureWorks2019.Production.Product PP
on PPS.ProductCategoryID = PP.ProductSubcategoryID
Join AdventureWorks2019.Production.ProductCategory PPC
on PPS.ProductCategoryID = PPC.ProductCategoryID
```



	ProductName	ListPrice	ProductSubcategory	ProductCategory	Price Rank	Category Price Rank	Category Price Rank with Rank
1	HL Road Handlebars	120.27	Bike Racks	Accessories	875	1	1
2	HL Road Handlebars	120.27	Bike Stands	Accessories	876	2	1
3	HL Road Handlebars	120.27	Bottles and Cages	Accessories	877	3	1
4	HL Road Handlebars	120.27	Cleaners	Accessories	878	4	1
5	HL Road Handlebars	120.27	Fenders	Accessories	879	5	1
6	HL Road Handlebars	120.27	Helmets	Accessories	880	6	1
7	HL Road Handlebars	120.27	Hydration Packs	Accessories	881	7	1
8	HL Road Handlebars	120.27	Lights	Accessories	882	8	1
9	HL Road Handlebars	120.27	Locks	Accessories	883	9	1



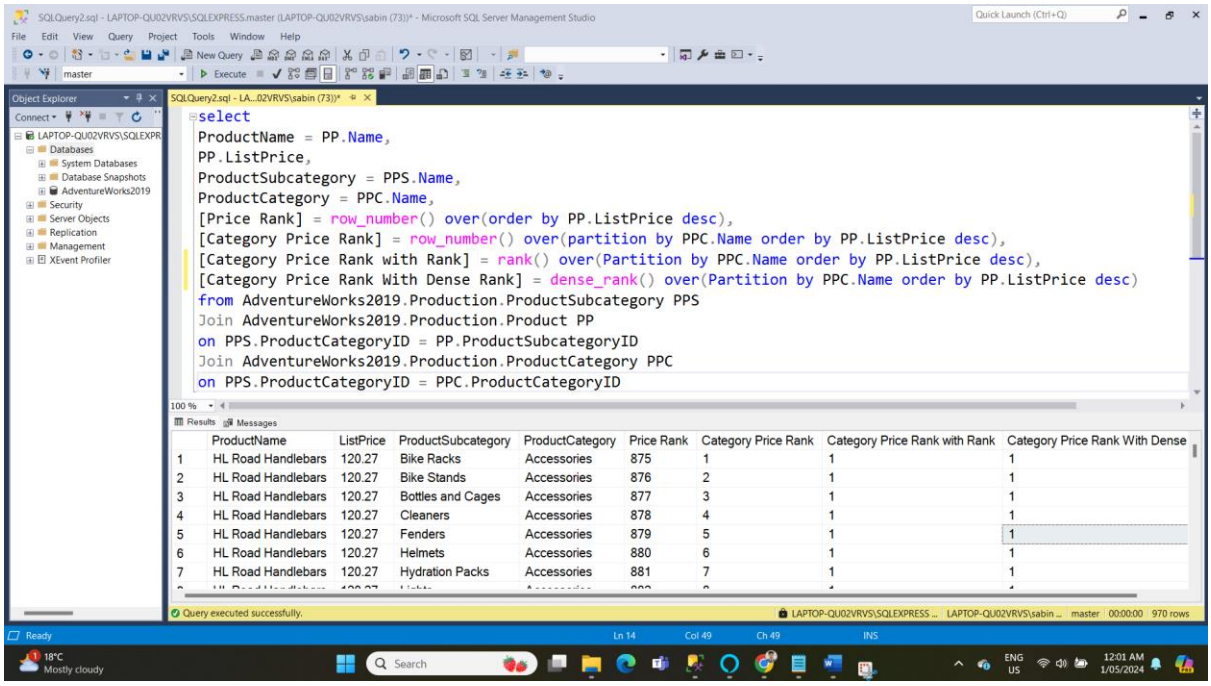
	ProductName	ListPrice	ProductSubcategory	ProductCategory	Price Rank	Category Price Rank	Category Price Rank with Rank
8	HL Road Handlebars	120.27	Lights	Accessories	882	8	1
9	HL Road Handlebars	120.27	Locks	Accessories	883	9	1
10	HL Road Handlebars	120.27	Panniers	Accessories	884	10	1
11	HL Road Handlebars	120.27	Pumps	Accessories	885	11	1
12	HL Road Handlebars	120.27	Tires and Tubes	Accessories	886	12	1
13	HL Mountain Handl...	120.27	Bike Racks	Accessories	887	13	1
14	HL Mountain Handl...	120.27	Bike Stands	Accessories	888	14	1
15	HL Mountain Handl...	120.27	Bottles and Cages	Accessories	889	15	1
16	HL Mountain Handl...	120.27	Cleaners	Accessories	890	16	1
17	HL Mountain Handl...	120.27	Fenders	Accessories	891	17	1
18	HL Mountain Handl...	120.27	Helmets	Accessories	892	18	1
19	HL Mountain Handl...	120.27	Hydration Packs	Accessories	893	19	1
20	HL Mountain Handl...	120.27	Lights	Accessories	894	20	1
21	HL Mountain Handl...	120.27	Locks	Accessories	895	21	1
22	HL Mountain Handl...	120.27	Panniers	Accessories	896	22	1
23	HL Mountain Handl...	120.27	Pumps	Accessories	897	23	1
24	HL Mountain Handl...	120.27	Tires and Tubes	Accessories	898	24	1
25	HL Touring Handle...	91.57	Bike Racks	Accessories	899	25	25
26	HL Touring Handle...	91.57	Bike Stands	Accessories	900	26	25
27	HL Touring Handle...	91.57	Bottles and Cages	Accessories	901	27	25
28	HL Touring Handle...	91.57	Cleaners	Accessories	902	28	25
29	HL Touring Handle...	91.57	Fenders	Accessories	903	29	25
30	HL Touring Handle...	91.57	Helmets	Accessories	904	30	25

Exercise 14: Modify your query from Exercise 13 by adding a derived column called "Category Price Rank With Dense Rank" that that uses the DENSE_RANK function to rank all products by ListPrice – within each category - in descending order. Observe the differences among the "Category Price

Rank", "Category Price Rank With Rank", and "Category Price Rank With Dense Rank" fields.

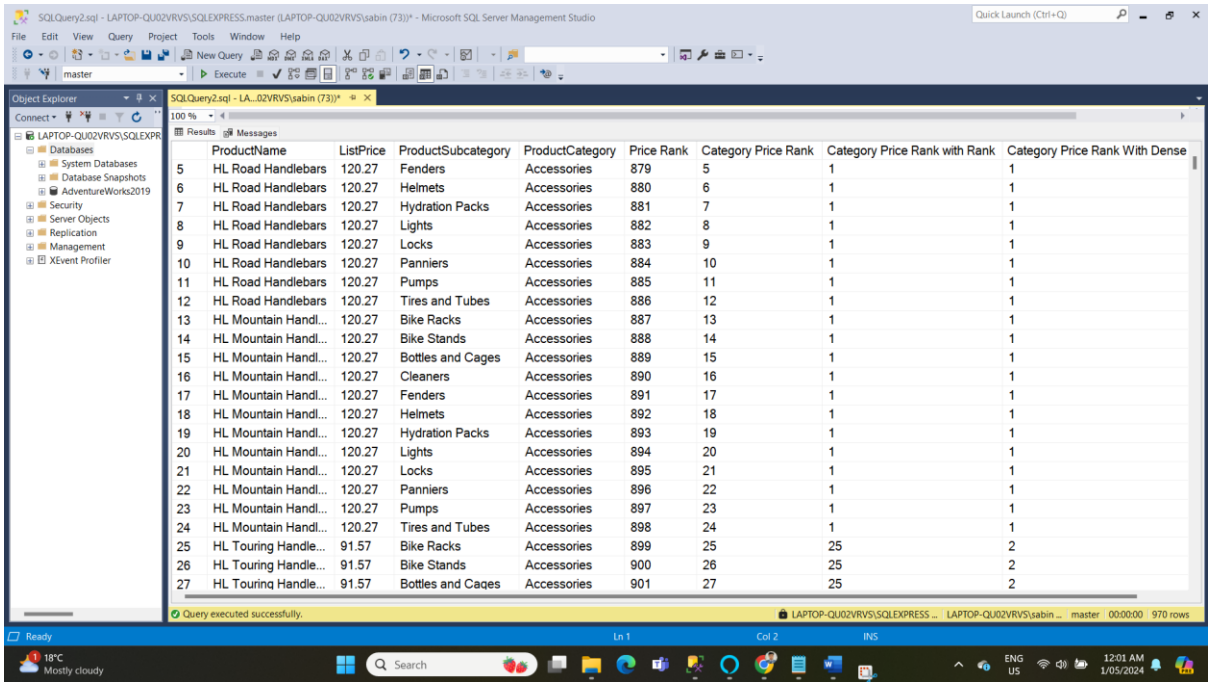
Answer:

```
select

ProductName = PP.Name,

PP.ListPrice,

ProductSubcategory = PPS.Name,

ProductCategory = PPC.Name,

[Price Rank] = row_number() over(order by PP.ListPrice desc),

[Category Price Rank] = row_number() over(partition by PPC.Name order by PP.ListPrice desc),

[Category Price Rank with Rank] = rank() over(Partition by PPC.Name order by PP.ListPrice desc),

[Category Price Rank With Dense Rank] = dense_rank() over(Partition by PPC.Name order by PP.ListPrice desc)

from AdventureWorks2019.Production.ProductSubcategory PPS

Join AdventureWorks2019.Production.Product PP

on PPS.ProductCategoryID = PP.ProductSubcategoryID

Join AdventureWorks2019.Production.ProductCategory PPC

on PPS.ProductCategoryID = PPC.ProductCategoryID
```



```
select
ProductName = PP.Name,
PP.ListPrice,
ProductSubcategory = PPS.Name,
ProductCategory = PPC.Name,
[Price Rank] = row_number() over(order by PP.ListPrice desc),
[Category Price Rank] = row_number() over(partition by PPC.Name order by PP.ListPrice desc),
[Category Price Rank with Rank] = rank() over(Partition by PPC.Name order by PP.ListPrice desc),
[Category Price Rank With Dense Rank] = dense_rank() over(Partition by PPC.Name order by PP.ListPrice desc)
from AdventureWorks2019.Production.ProductSubcategory PPS
Join AdventureWorks2019.Production.Product PP
on PPS.ProductCategoryID = PP.ProductSubcategoryID
Join AdventureWorks2019.Production.ProductCategory PPC
on PPS.ProductCategoryID = PPC.ProductCategoryID
```

	ProductName	ListPrice	ProductSubcategory	ProductCategory	Price Rank	Category Price Rank	Category Price Rank with Rank	Category Price Rank With Dense
1	HL Road Handlebars	120.27	Bike Racks	Accessories	875	1	1	1
2	HL Road Handlebars	120.27	Bike Stands	Accessories	876	2	1	1
3	HL Road Handlebars	120.27	Bottles and Cages	Accessories	877	3	1	1
4	HL Road Handlebars	120.27	Cleaners	Accessories	878	4	1	1
5	HL Road Handlebars	120.27	Fenders	Accessories	879	5	1	1
6	HL Road Handlebars	120.27	Helmets	Accessories	880	6	1	1
7	HL Road Handlebars	120.27	Hydration Packs	Accessories	881	7	1	1

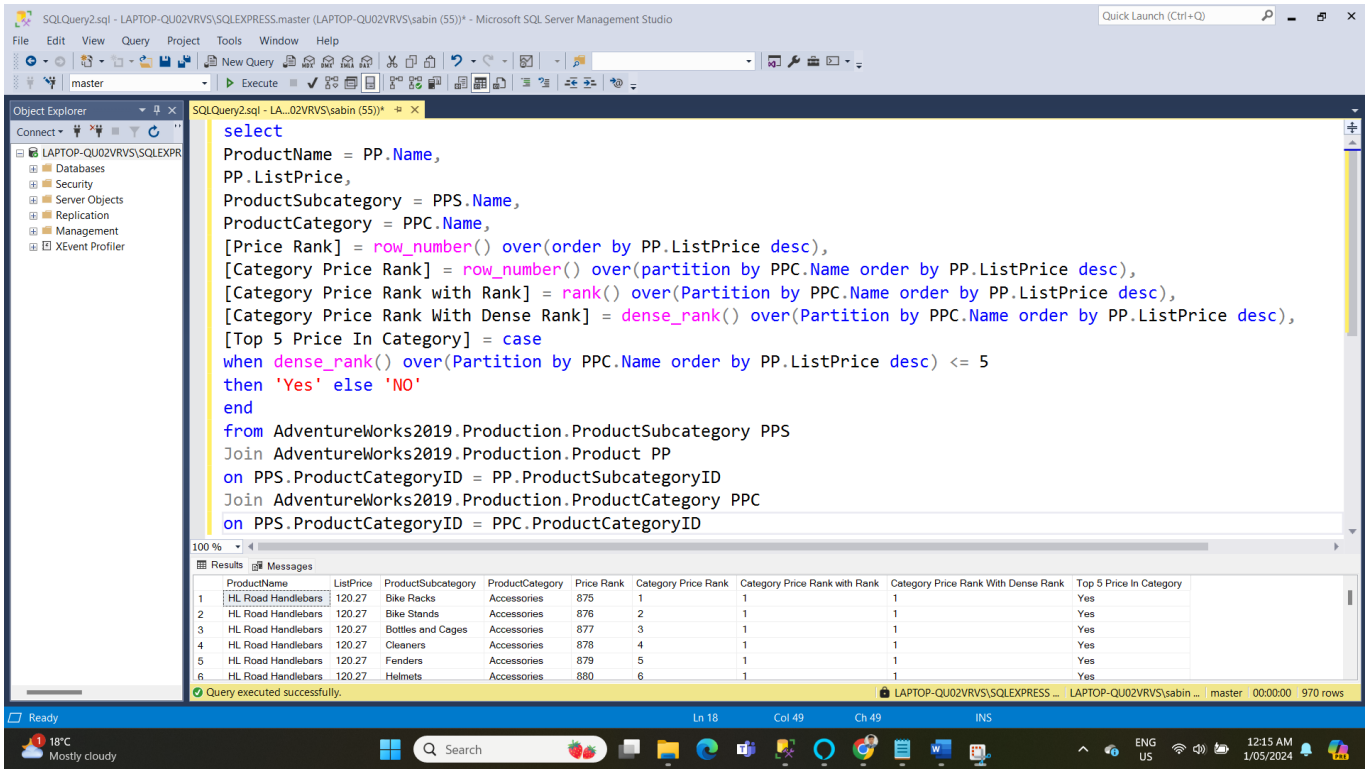


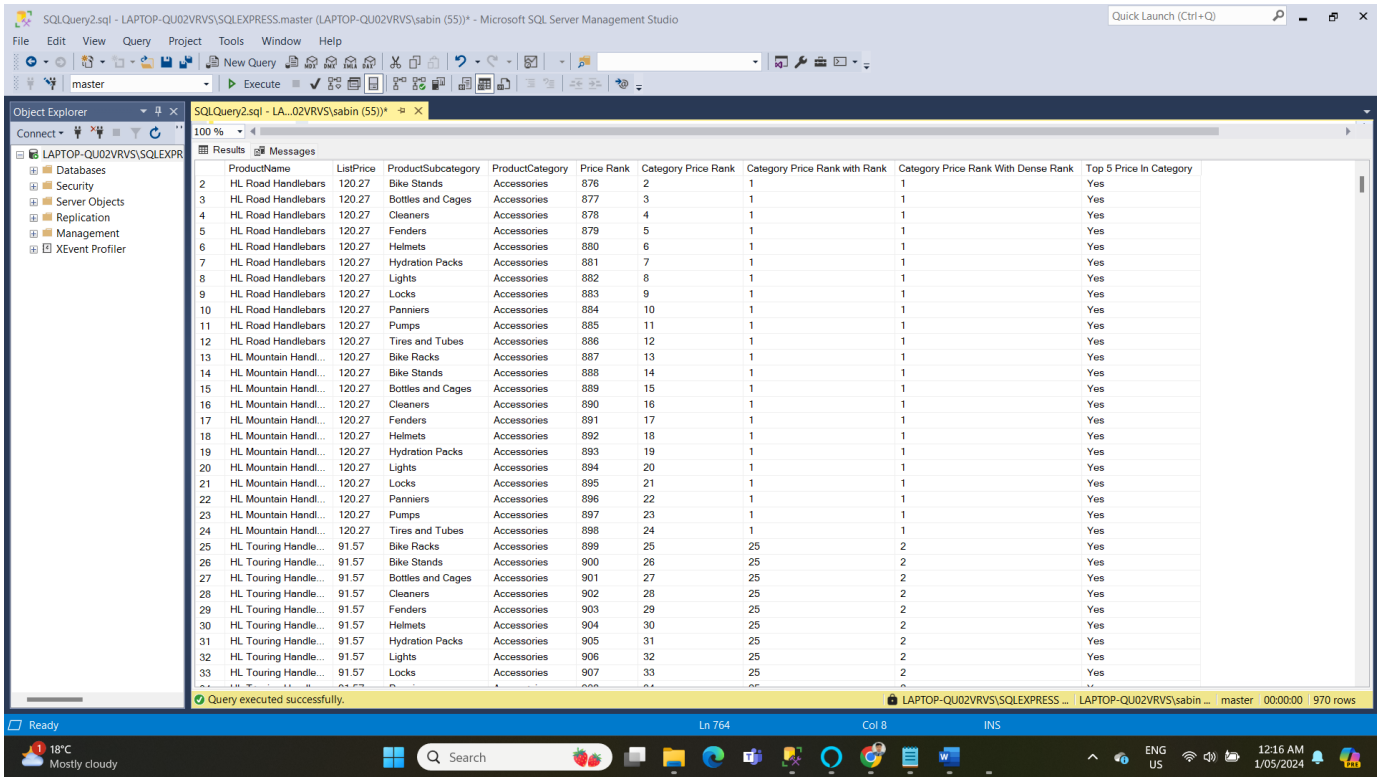
	ProductName	ListPrice	ProductSubcategory	ProductCategory	Price Rank	Category Price Rank	Category Price Rank with Rank	Category Price Rank With Dense
5	HL Road Handlebars	120.27	Fenders	Accessories	879	5	1	1
6	HL Road Handlebars	120.27	Helmets	Accessories	880	6	1	1
7	HL Road Handlebars	120.27	Hydration Packs	Accessories	881	7	1	1
8	HL Road Handlebars	120.27	Lights	Accessories	882	8	1	1
9	HL Road Handlebars	120.27	Locks	Accessories	883	9	1	1
10	HL Road Handlebars	120.27	Panniers	Accessories	884	10	1	1
11	HL Road Handlebars	120.27	Pumps	Accessories	885	11	1	1
12	HL Road Handlebars	120.27	Tires and Tubes	Accessories	886	12	1	1
13	HL Mountain Handl...	120.27	Bike Racks	Accessories	887	13	1	1
14	HL Mountain Handl...	120.27	Bike Stands	Accessories	888	14	1	1
15	HL Mountain Handl...	120.27	Bottles and Cages	Accessories	889	15	1	1
16	HL Mountain Handl...	120.27	Cleaners	Accessories	890	16	1	1
17	HL Mountain Handl...	120.27	Fenders	Accessories	891	17	1	1
18	HL Mountain Handl...	120.27	Helmets	Accessories	892	18	1	1
19	HL Mountain Handl...	120.27	Hydration Packs	Accessories	893	19	1	1
20	HL Mountain Handl...	120.27	Lights	Accessories	894	20	1	1
21	HL Mountain Handl...	120.27	Locks	Accessories	895	21	1	1
22	HL Mountain Handl...	120.27	Panniers	Accessories	896	22	1	1
23	HL Mountain Handl...	120.27	Pumps	Accessories	897	23	1	1
24	HL Mountain Handl...	120.27	Tires and Tubes	Accessories	898	24	1	1
25	HL Touring Handle...	91.57	Bike Racks	Accessories	899	25	25	2
26	HL Touring Handle...	91.57	Bike Stands	Accessories	900	26	25	2
27	HL Touring Handle...	91.57	Bottles and Cages	Accessories	901	27	25	2

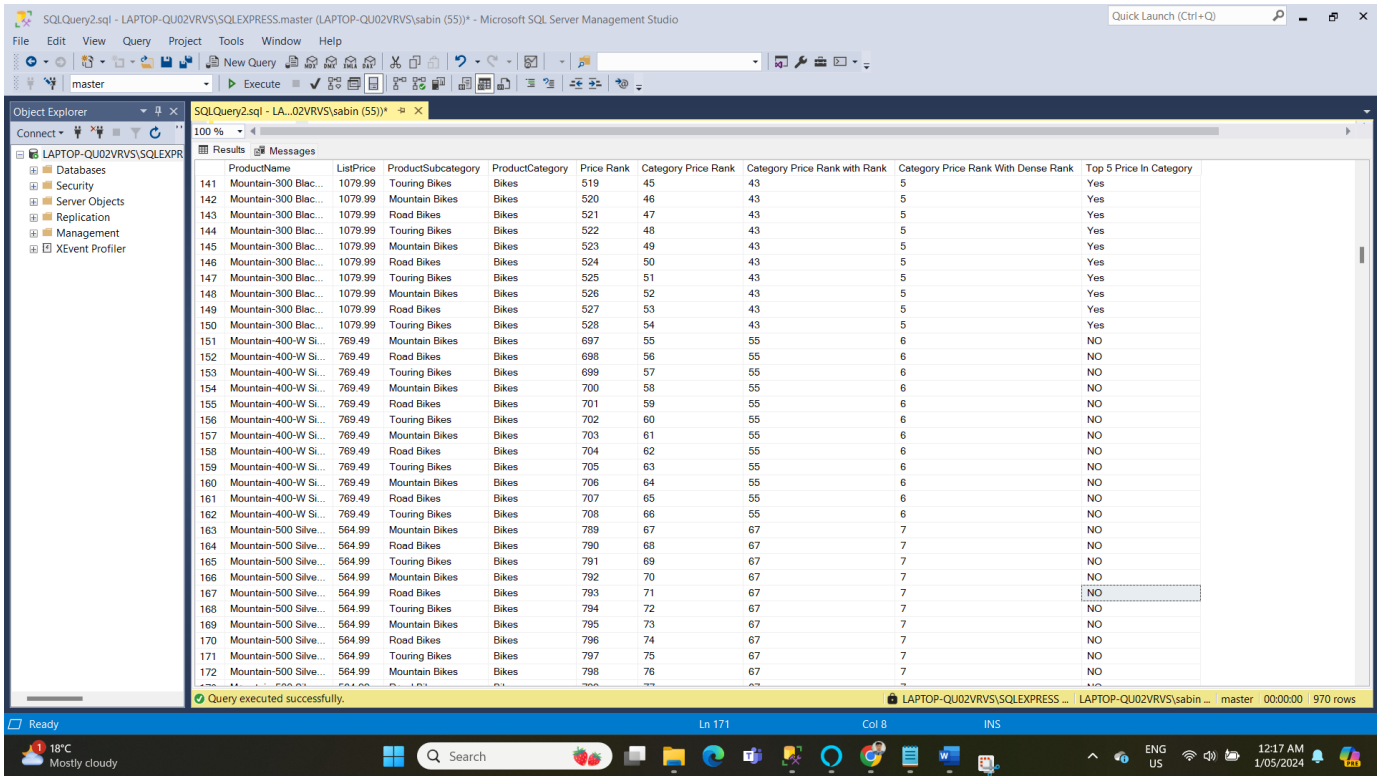
Exercise 15: Examine the code you wrote to define the "Top 5 Price In Category" field back in the ROW_NUMBER exercises. Now that you understand the differences among ROW_NUMBER, RANK, and DENSE_RANK, consider which of these functions would be most appropriate to return a true top 5 products by price, assuming we want to see the top 5 distinct prices AND we want "ties" (by price) to all share the same rank.

Answer:

```
select
ProductName = PP.Name,
PP.ListPrice,
ProductSubcategory = PPS.Name,
ProductCategory = PPC.Name,
[Price Rank] = row_number() over(order by PP.ListPrice desc),
[Category Price Rank] = row_number() over(partition by PPC.Name order by PP.ListPrice desc),
[Category Price Rank with Rank] = rank() over(Partition by PPC.Name order by PP.ListPrice desc),
[Category Price Rank With Dense Rank] = dense_rank() over(Partition by PPC.Name order by PP.ListPrice desc),
[Top 5 Price In Category] = case
when dense_rank() over(Partition by PPC.Name order by PP.ListPrice desc) <= 5
then 'Yes' else 'NO'
end
from AdventureWorks2019.Production.ProductSubcategory PPS
Join AdventureWorks2019.Production.Product PP
on PPS.ProductCategoryID = PP.ProductSubcategoryID
Join AdventureWorks2019.Production.ProductCategory PPC
on PPS.ProductCategoryID = PPC.ProductCategoryID
```


SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.master (LAPTOP-QU02VRVS\sabin (55))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
master
Object Explorer
Connect
LAPTOP-QU02VRVS\SQLEXPR
Databases
Security
Server Objects
Replication
Management
XEvent Profiler
SQLQuery2.sql - LA...02VRVS\sabin (55))*
select
ProductName = PP.Name,
PP.ListPrice,
ProductSubcategory = PPS.Name,
ProductCategory = PPC.Name,
[Price Rank] = row_number() over(order by PP.ListPrice desc),
[Category Price Rank] = row_number() over(partition by PPC.Name order by PP.ListPrice desc),
[Category Price Rank with Rank] = rank() over(Partition by PPC.Name order by PP.ListPrice desc),
[Category Price Rank With Dense Rank] = dense_rank() over(Partition by PPC.Name order by PP.ListPrice desc),
[Top 5 Price In Category] = case
when dense_rank() over(Partition by PPC.Name order by PP.ListPrice desc) <= 5
then 'Yes' else 'NO'
end
from AdventureWorks2019.Production.ProductSubcategory PPS
Join AdventureWorks2019.Production.Product PP
on PPS.ProductCategoryID = PP.ProductSubcategoryID
Join AdventureWorks2019.Production.ProductCategory PPC
on PPS.ProductCategoryID = PPC.ProductCategoryID
100 %
Results
Messages
ProductName ListPrice ProductSubcategory ProductCategory Price Rank Category Price Rank Category Price Rank with Rank Category Price Rank With Dense Rank Top 5 Price In Category
1 HL Road Handlebars 120.27 Bike Racks Accessories 875 1 1 1 Yes
2 HL Road Handlebars 120.27 Bike Stands Accessories 876 2 1 1 Yes
3 HL Road Handlebars 120.27 Bottles and Cages Accessories 877 3 1 1 Yes
4 HL Road Handlebars 120.27 Cleaners Accessories 878 4 1 1 Yes
5 HL Road Handlebars 120.27 Fenders Accessories 879 5 1 1 Yes
6 HL Road Handlebars 120.27 Helmets Accessories 880 6 1 1 Yes
Query executed successfully.
LAPTOP-QU02VRVS\SQLEXPRESS ... LAPTOP-QU02VRVS\sabin ... master 00:00:00 970 rows
Ready
Ln 18 Col 49 Ch 49 INS
18°C Mostly cloudy
Search
ENG US
12:15 AM 1/05/2024


SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.master (LAPTOP-QU02VRVS\sabin (55))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
master
Object Explorer
Connect
LAPTOP-QU02VRVS\SQLEXPR
Databases
Security
Server Objects
Replication
Management
XEvent Profiler
SQLQuery2.sql - LA...02VRVS\sabin (55))*
100 %
Results
Messages
ProductName ListPrice ProductSubcategory ProductCategory Price Rank Category Price Rank Category Price Rank with Rank Category Price Rank With Dense Rank Top 5 Price In Category
2 HL Road Handlebars 120.27 Bike Stands Accessories 876 2 1 1 Yes
3 HL Road Handlebars 120.27 Bottles and Cages Accessories 877 3 1 1 Yes
4 HL Road Handlebars 120.27 Cleaners Accessories 878 4 1 1 Yes
5 HL Road Handlebars 120.27 Fenders Accessories 879 5 1 1 Yes
6 HL Road Handlebars 120.27 Helmets Accessories 880 6 1 1 Yes
7 HL Road Handlebars 120.27 Hydration Packs Accessories 881 7 1 1 Yes
8 HL Road Handlebars 120.27 Lights Accessories 882 8 1 1 Yes
9 HL Road Handlebars 120.27 Locks Accessories 883 9 1 1 Yes
10 HL Road Handlebars 120.27 Panniers Accessories 884 10 1 1 Yes
11 HL Road Handlebars 120.27 Pumps Accessories 885 11 1 1 Yes
12 HL Road Handlebars 120.27 Tires and Tubes Accessories 886 12 1 1 Yes
13 HL Mountain Handl... 120.27 Bike Racks Accessories 887 13 1 1 Yes
14 HL Mountain Handl... 120.27 Bike Stands Accessories 888 14 1 1 Yes
15 HL Mountain Handl... 120.27 Bottles and Cages Accessories 889 15 1 1 Yes
16 HL Mountain Handl... 120.27 Cleaners Accessories 890 16 1 1 Yes
17 HL Mountain Handl... 120.27 Fenders Accessories 891 17 1 1 Yes
18 HL Mountain Handl... 120.27 Helmets Accessories 892 18 1 1 Yes
19 HL Mountain Handl... 120.27 Hydration Packs Accessories 893 19 1 1 Yes
20 HL Mountain Handl... 120.27 Lights Accessories 894 20 1 1 Yes
21 HL Mountain Handl... 120.27 Locks Accessories 895 21 1 1 Yes
22 HL Mountain Handl... 120.27 Panniers Accessories 896 22 1 1 Yes
23 HL Mountain Handl... 120.27 Pumps Accessories 897 23 1 1 Yes
24 HL Mountain Handl... 120.27 Tires and Tubes Accessories 898 24 1 1 Yes
25 HL Touring Handle... 91.57 Bike Racks Accessories 899 25 25 2 Yes
26 HL Touring Handle... 91.57 Bike Stands Accessories 900 26 25 2 Yes
27 HL Touring Handle... 91.57 Bottles and Cages Accessories 901 27 25 2 Yes
28 HL Touring Handle... 91.57 Cleaners Accessories 902 28 25 2 Yes
29 HL Touring Handle... 91.57 Fenders Accessories 903 29 25 2 Yes
30 HL Touring Handle... 91.57 Helmets Accessories 904 30 25 2 Yes
31 HL Touring Handle... 91.57 Hydration Packs Accessories 905 31 25 2 Yes
32 HL Touring Handle... 91.57 Lights Accessories 906 32 25 2 Yes
33 HL Touring Handle... 91.57 Locks Accessories 907 33 25 2 Yes
Query executed successfully.
LAPTOP-QU02VRVS\SQLEXPRESS ... LAPTOP-QU02VRVS\sabin ... master 00:00:00 970 rows
Ready
Ln 764 Col 8 INS
18°C Mostly cloudy
Search
ENG US
12:16 AM 1/05/2024



	ProductName	ListPrice	ProductSubcategory	ProductCategory	Price Rank	Category Price Rank	Category Price Rank with Rank	Category Price Rank With Dense Rank	Top 5 Price In Category
141	Mountain-300 Blac...	1079.99	Touring Bikes	Bikes	519	45	43	5	Yes
142	Mountain-300 Blac...	1079.99	Mountain Bikes	Bikes	520	46	43	5	Yes
143	Mountain-300 Blac...	1079.99	Road Bikes	Bikes	521	47	43	5	Yes
144	Mountain-300 Blac...	1079.99	Touring Bikes	Bikes	522	48	43	5	Yes
145	Mountain-300 Blac...	1079.99	Mountain Bikes	Bikes	523	49	43	5	Yes
146	Mountain-300 Blac...	1079.99	Road Bikes	Bikes	524	50	43	5	Yes
147	Mountain-300 Blac...	1079.99	Touring Bikes	Bikes	525	51	43	5	Yes
148	Mountain-300 Blac...	1079.99	Mountain Bikes	Bikes	526	52	43	5	Yes
149	Mountain-300 Blac...	1079.99	Road Bikes	Bikes	527	53	43	5	Yes
150	Mountain-300 Blac...	1079.99	Touring Bikes	Bikes	528	54	43	5	Yes
151	Mountain-400-W Si...	769.49	Mountain Bikes	Bikes	697	55	55	6	NO
152	Mountain-400-W Si...	769.49	Road Bikes	Bikes	698	56	55	6	NO
153	Mountain-400-W Si...	769.49	Touring Bikes	Bikes	699	57	55	6	NO
154	Mountain-400-W Si...	769.49	Mountain Bikes	Bikes	700	58	55	6	NO
155	Mountain-400-W Si...	769.49	Road Bikes	Bikes	701	59	55	6	NO
156	Mountain-400-W Si...	769.49	Touring Bikes	Bikes	702	60	55	6	NO
157	Mountain-400-W Si...	769.49	Mountain Bikes	Bikes	703	61	55	6	NO
158	Mountain-400-W Si...	769.49	Road Bikes	Bikes	704	62	55	6	NO
159	Mountain-400-W Si...	769.49	Touring Bikes	Bikes	705	63	55	6	NO
160	Mountain-400-W Si...	769.49	Mountain Bikes	Bikes	706	64	55	6	NO
161	Mountain-400-W Si...	769.49	Road Bikes	Bikes	707	65	55	6	NO
162	Mountain-400-W Si...	769.49	Touring Bikes	Bikes	708	66	55	6	NO
163	Mountain-500 Silve...	564.99	Mountain Bikes	Bikes	789	67	67	7	NO
164	Mountain-500 Silve...	564.99	Road Bikes	Bikes	790	68	67	7	NO
165	Mountain-500 Silve...	564.99	Touring Bikes	Bikes	791	69	67	7	NO
166	Mountain-500 Silve...	564.99	Mountain Bikes	Bikes	792	70	67	7	NO
167	Mountain-500 Silve...	564.99	Road Bikes	Bikes	793	71	67	7	NO
168	Mountain-500 Silve...	564.99	Touring Bikes	Bikes	794	72	67	7	NO
169	Mountain-500 Silve...	564.99	Mountain Bikes	Bikes	795	73	67	7	NO
170	Mountain-500 Silve...	564.99	Road Bikes	Bikes	796	74	67	7	NO
171	Mountain-500 Silve...	564.99	Touring Bikes	Bikes	797	75	67	7	NO
172	Mountain-500 Silve...	564.99	Mountain Bikes	Bikes	798	76	67	7	NO

LEAD and LAG - Exercises

Exercise 1: Create a query with the following columns:

“PurchaseOrderID” from the Purchasing.PurchaseOrderHeader table

“OrderDate” from the Purchasing.PurchaseOrderHeader table

“TotalDue” from the Purchasing.PurchaseOrderHeader table

“Name” from the Purchasing.Vendor table, which can be aliased as “VendorName”*

***Join Purchasing.Vendor to Purchasing.PurchaseOrderHeader on BusinessEntityID = VendorID**

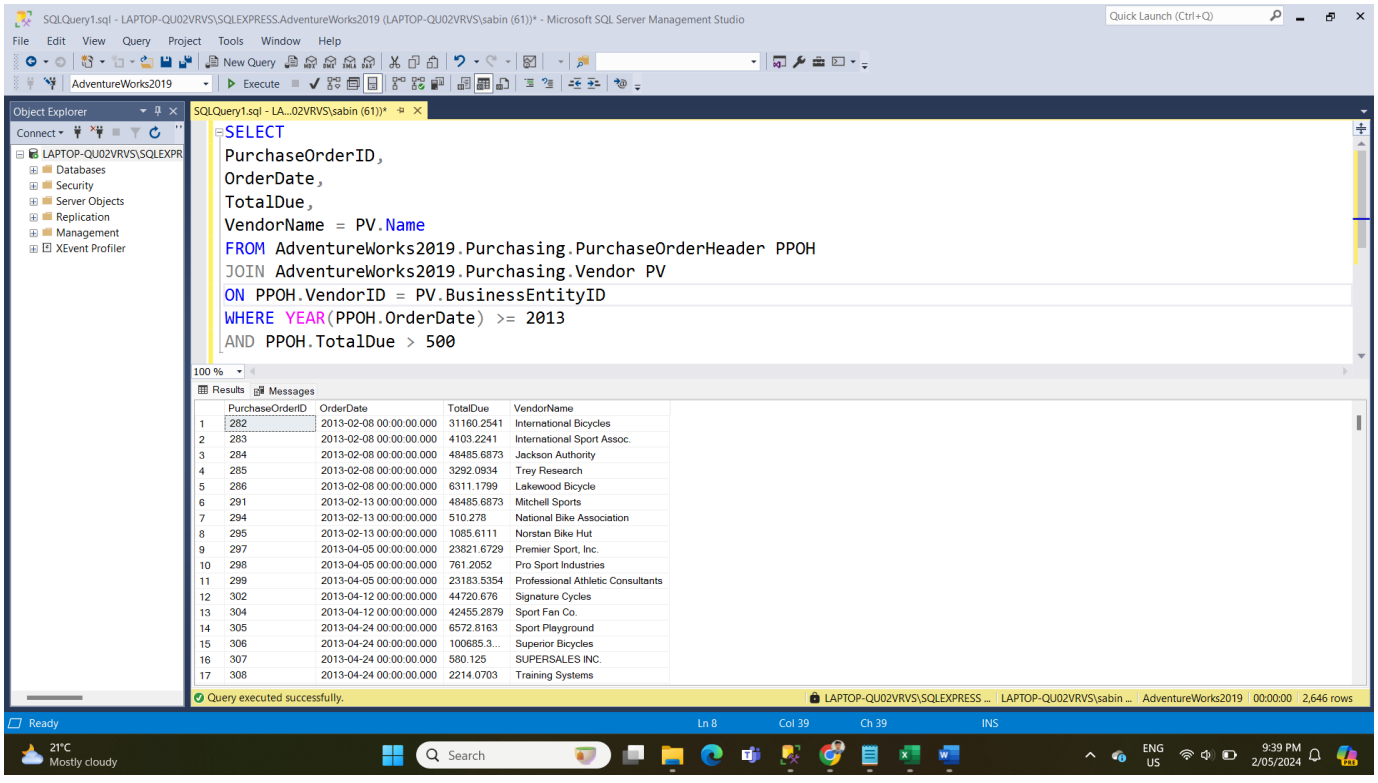
Apply the following criteria to the query:

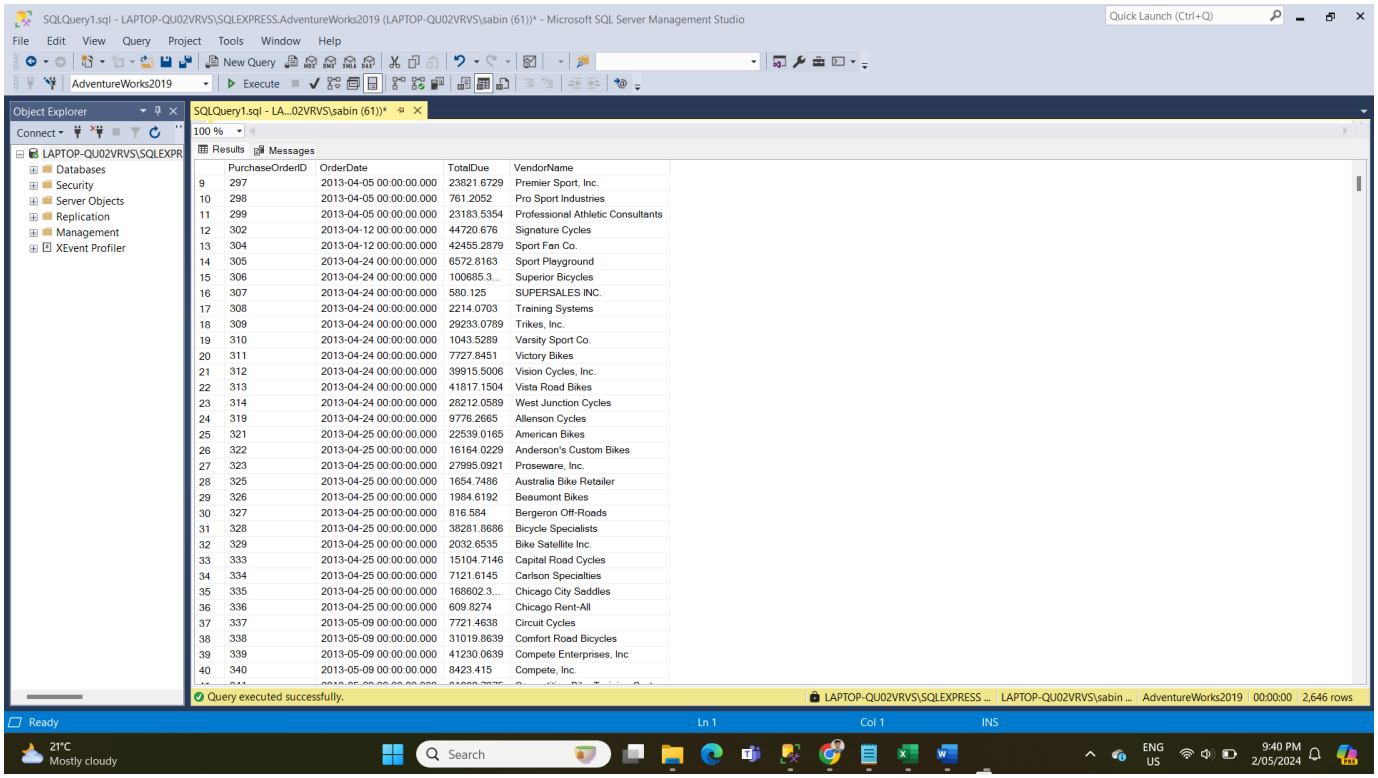
Order must have taken place on or after 2013

TotalDue must be greater than $500

Answer:

```
SELECT
PurchaseOrderID,
OrderDate,
TotalDue,
VendorName = PV.Name
FROM AdventureWorks2019.Purchasing.PurchaseOrderHeader PPOH
JOIN AdventureWorks2019.Purchasing.Vendor PV
ON PPOH.VendorID = PV.BusinessEntityID
WHERE YEAR(PPOH.OrderDate) >= 2013
AND PPOH.TotalDue > 500
```





Exercise 2: Modify your query from Exercise 1 by adding a derived column called "PrevOrderFromVendorAmt", that returns the “previous” TotalDue value (relative to the current row) within the group of all orders with the same vendor ID. We are defining “previous” based on order date.

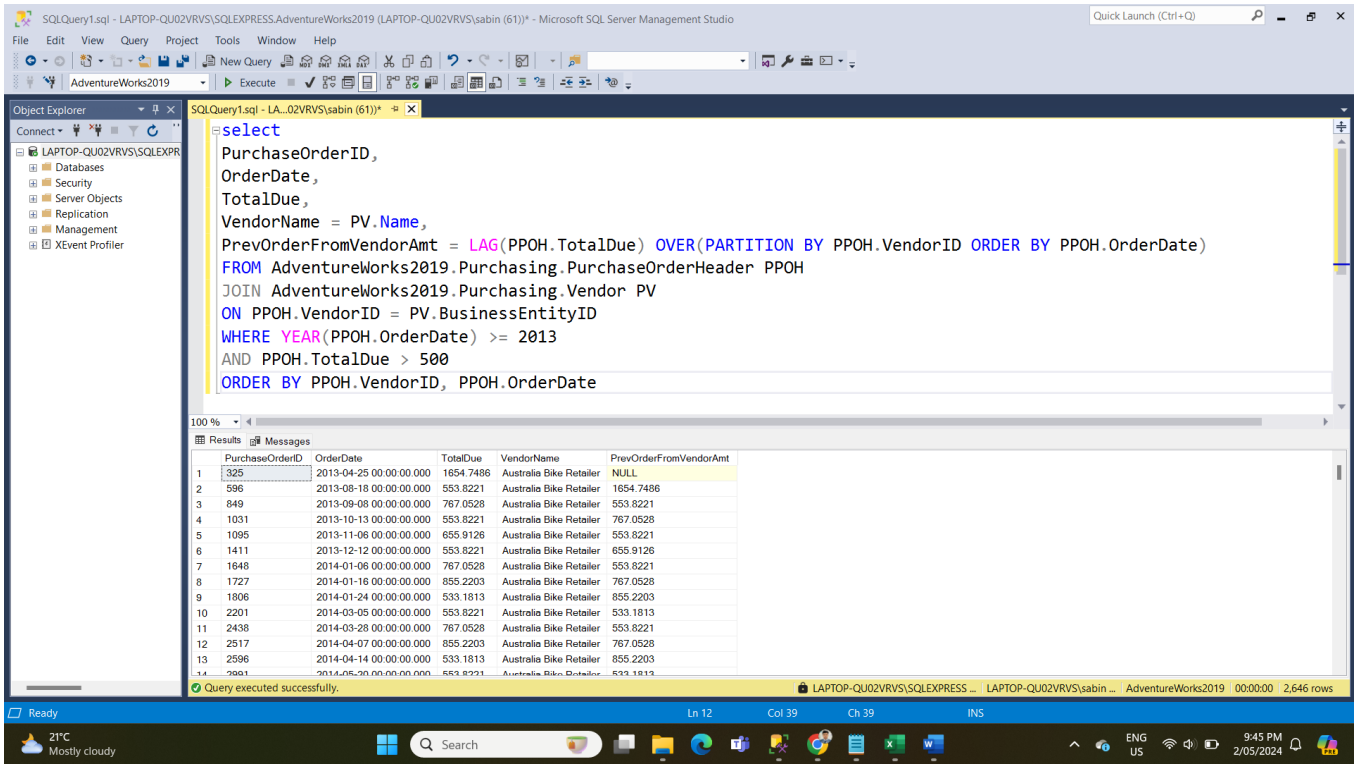
Answer:

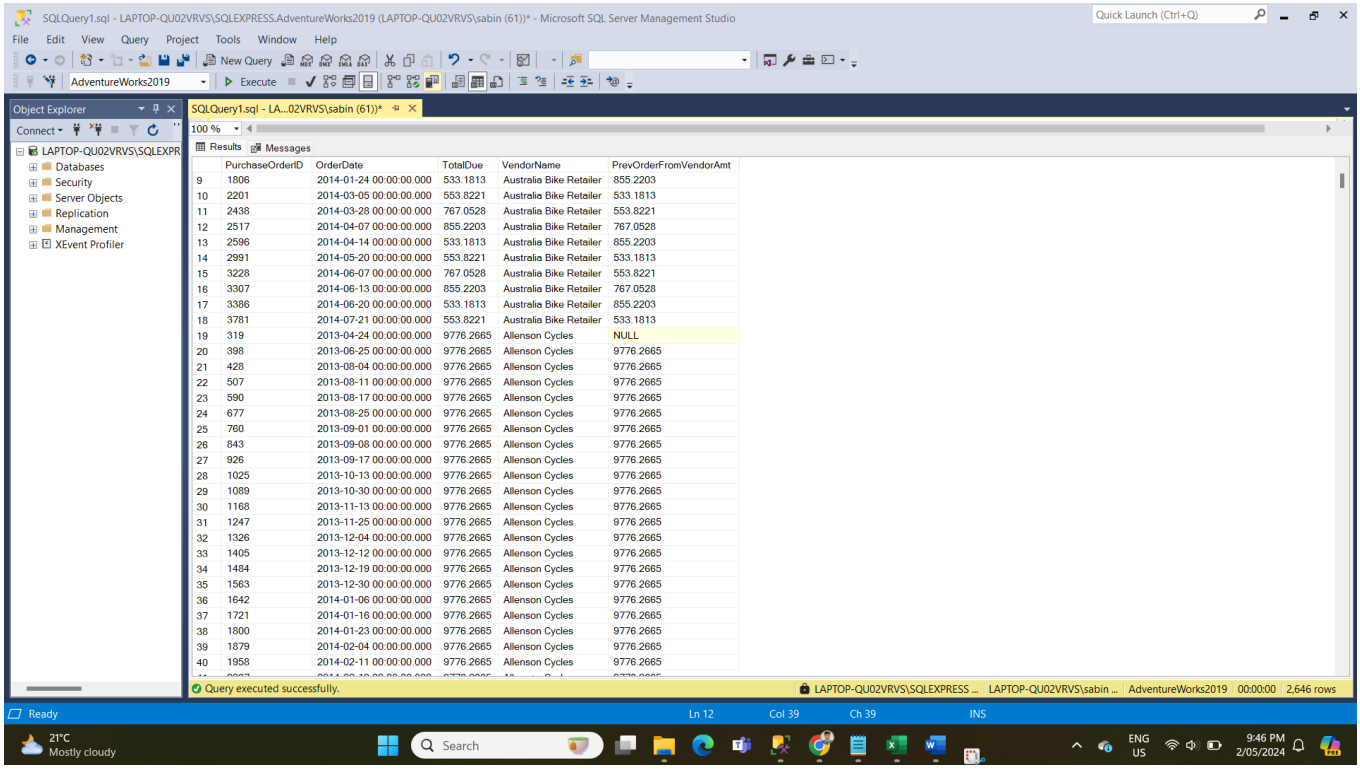
```
select

PurchaseOrderID,

OrderDate,

TotalDue,

VendorName = PV.Name,

PrevOrderFromVendorAmt = LAG(PPOH.TotalDue) OVER(PARTITION BY PPOH.VendorID ORDER BY PPOH.OrderDate)

FROM AdventureWorks2019.Purchasing.PurchaseOrderHeader PPOH

JOIN AdventureWorks2019.Purchasing.Vendor PV

ON PPOH.VendorID = PV.BusinessEntityID
```

WHERE YEAR(PPOH.OrderDate) >= 2013

AND PPOH.TotalDue > 500

ORDER BY PPOH.VendorID, PPOH.OrderDate

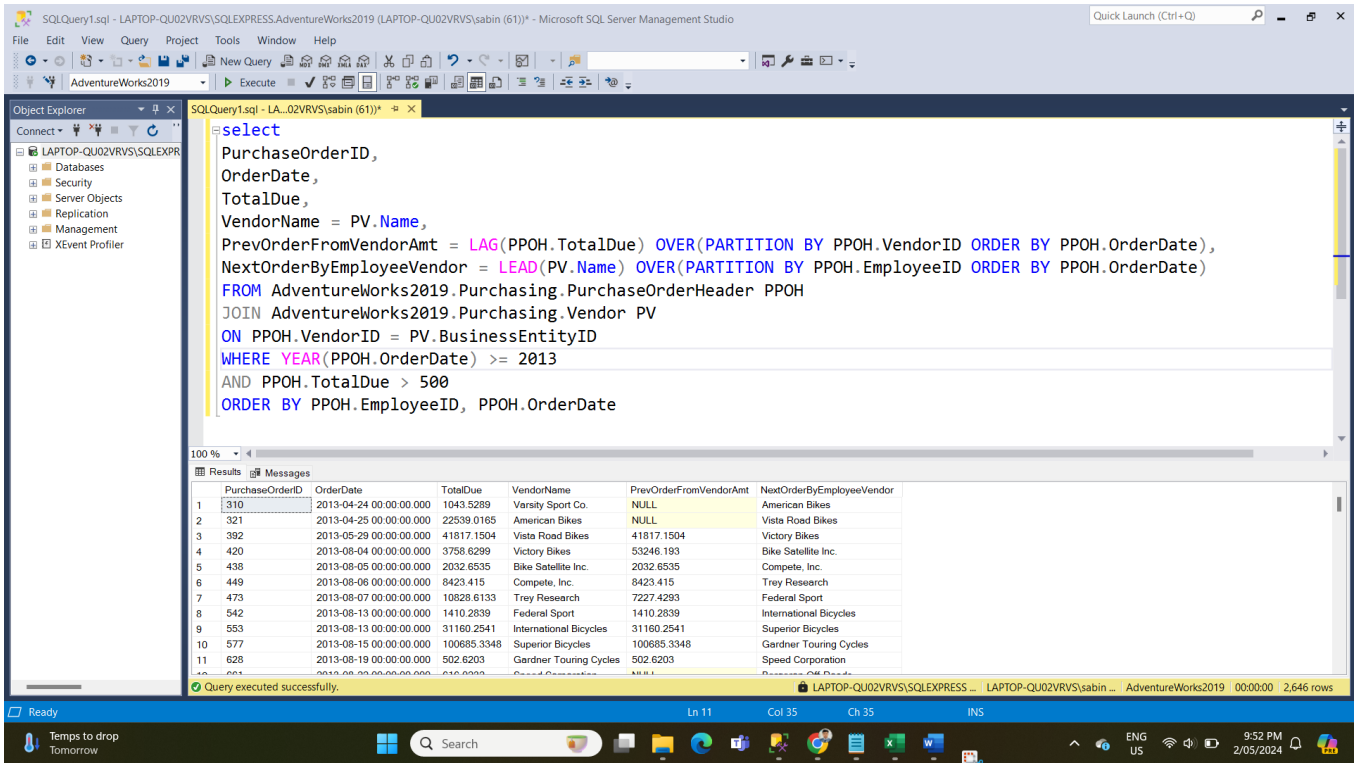


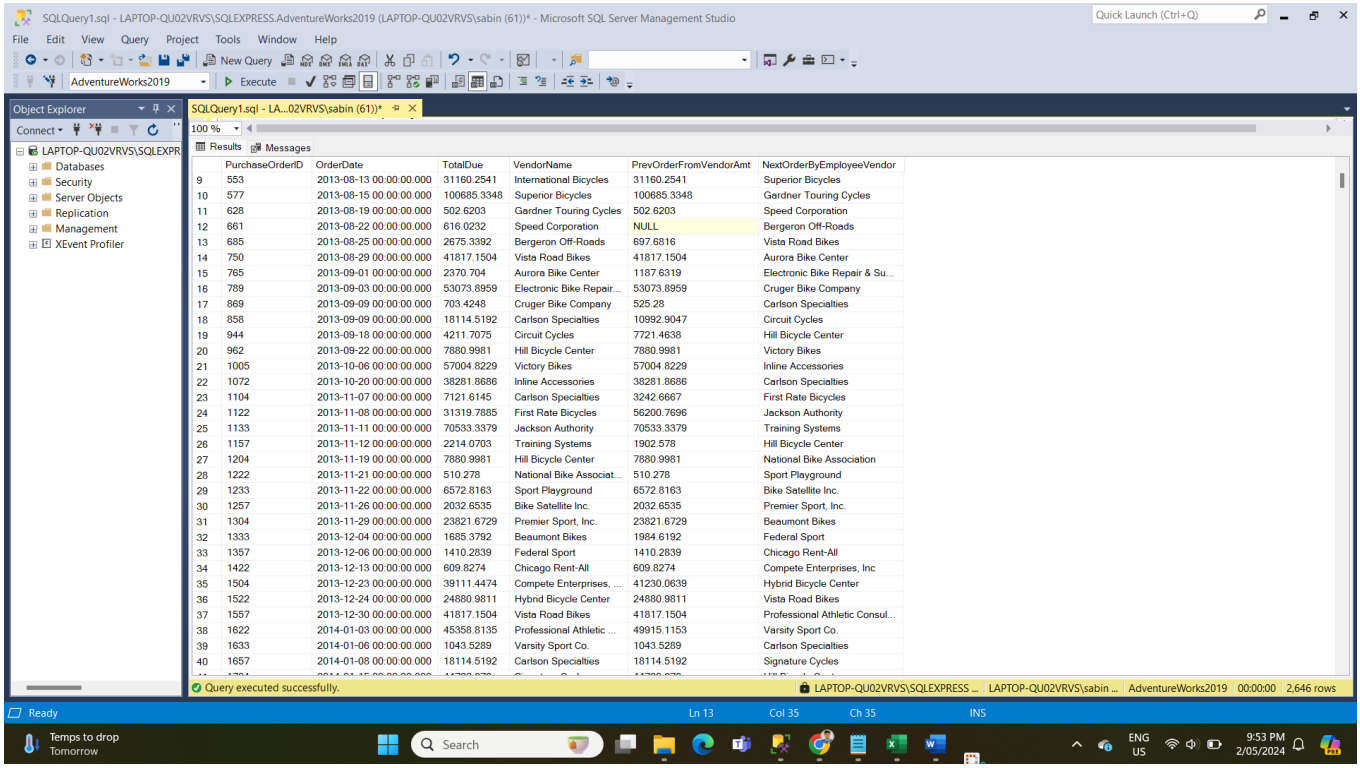


Exercise 3: Modify your query from Exercise 2 by adding a derived column called "NextOrderByEmployeeVendor", that returns the “next” vendor name (the “name” field from Purchasing.Vendor) within the group of all orders that have the same EmployeeID value in Purchasing.PurchaseOrderHeader. Similar to the last exercise, we are defining “next” based on order date.

Answer:

```
select

PurchaseOrderID,

OrderDate,

TotalDue,

VendorName = PV.Name,

PrevOrderFromVendorAmt = LAG(PPOH.TotalDue) OVER(PARTITION BY
PPOH.VendorID ORDER BY PPOH.OrderDate),

NextOrderByEmployeeVendor = LEAD(PV.Name) OVER(PARTITION BY
PPOH.EmployeeID ORDER BY PPOH.OrderDate)

FROM AdventureWorks2019.Purchasing.PurchaseOrderHeader PPOH

JOIN AdventureWorks2019.Purchasing.Vendor PV

ON PPOH.VendorID = PV.BusinessEntityID

WHERE YEAR(PPOH.OrderDate) >= 2013

AND PPOH.TotalDue > 500

ORDER BY PPOH.EmployeeID, PPOH.OrderDate
```



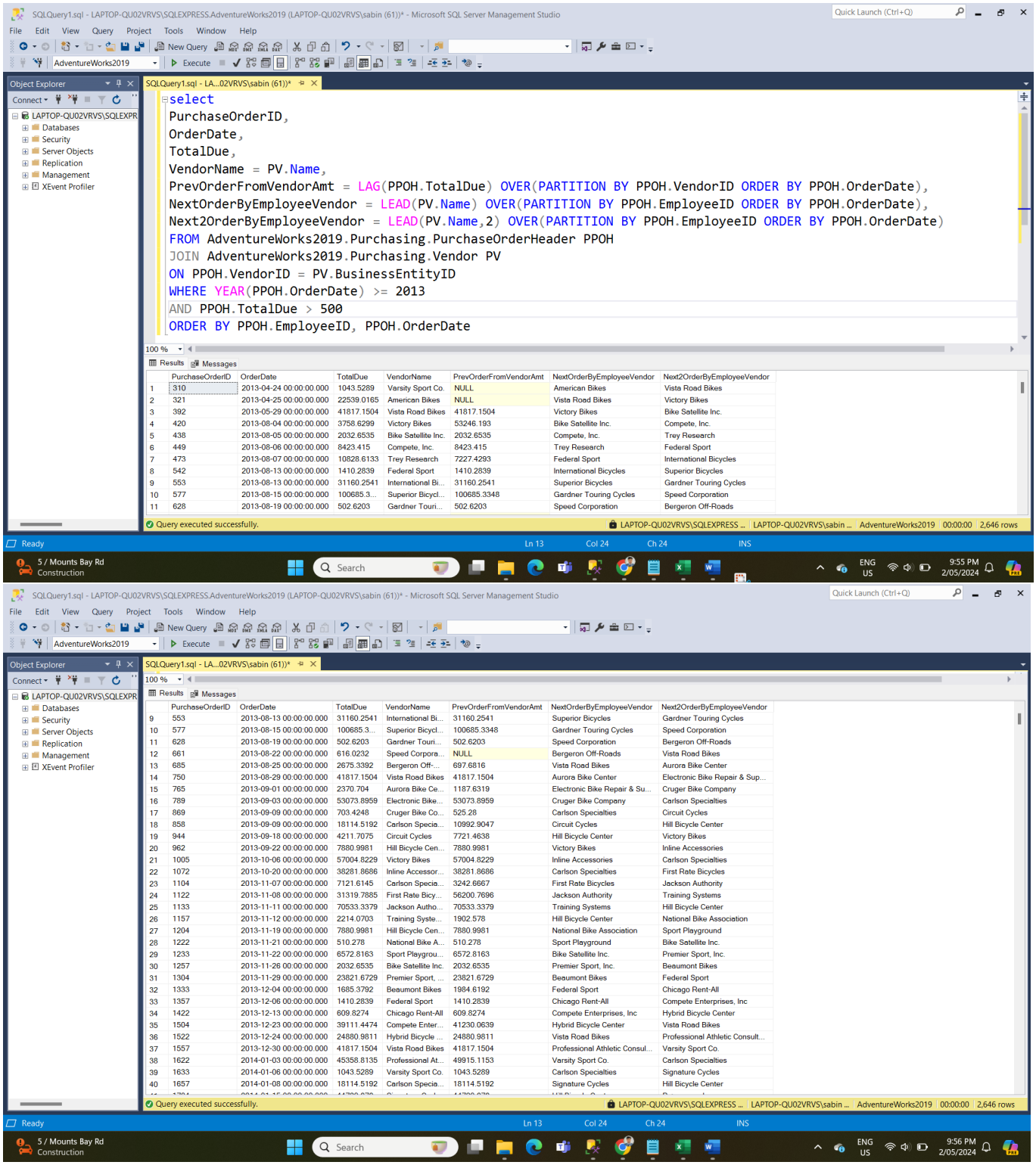


Exercise 4: Modify your query from Exercise 3 by adding a derived column called "Next2OrderByEmployeeVendor" that returns, within the group of all orders that have the same EmployeeID, the vendor name offset TWO orders into the “future” relative to the order in the current row. The code should be

very similar to Exercise 3, but with an extra argument passed to the Window Function used.

Answer:

```
select

PurchaseOrderID,

OrderDate,

TotalDue,

VendorName = PV.Name,

PrevOrderFromVendorAmt = LAG(PPOH.TotalDue) OVER(PARTITION BY
PPOH.VendorID ORDER BY PPOH.OrderDate),

NextOrderByEmployeeVendor = LEAD(PV.Name) OVER(PARTITION BY
PPOH.EmployeeID ORDER BY PPOH.OrderDate),

Next2OrderByEmployeeVendor = LEAD(PV.Name,2) OVER(PARTITION BY
PPOH.EmployeeID ORDER BY PPOH.OrderDate)

FROM AdventureWorks2019.Purchasing.PurchaseOrderHeader PPOH

JOIN AdventureWorks2019.Purchasing.Vendor PV

ON PPOH.VendorID = PV.BusinessEntityID

WHERE YEAR(PPOH.OrderDate) >= 2013

AND PPOH.TotalDue > 500

ORDER BY PPOH.EmployeeID, PPOH.OrderDate
```

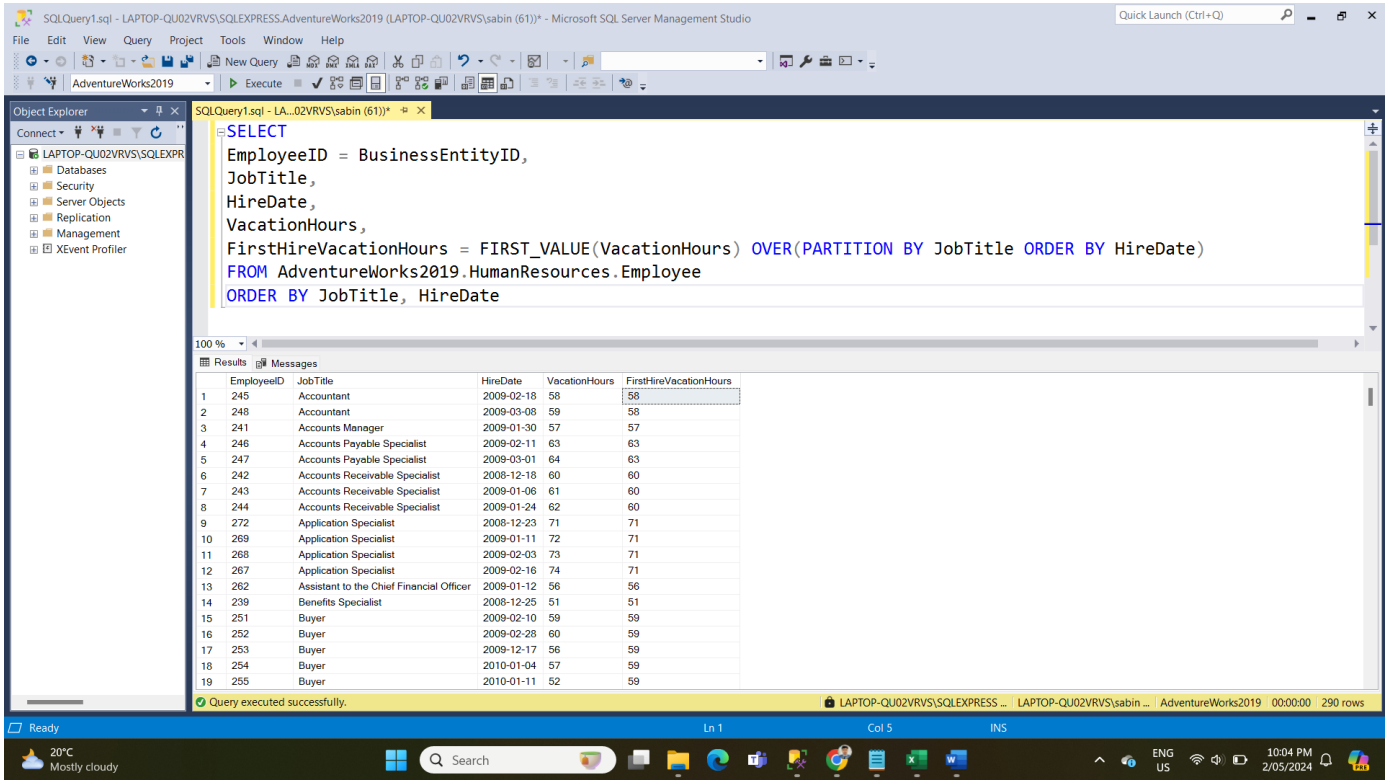


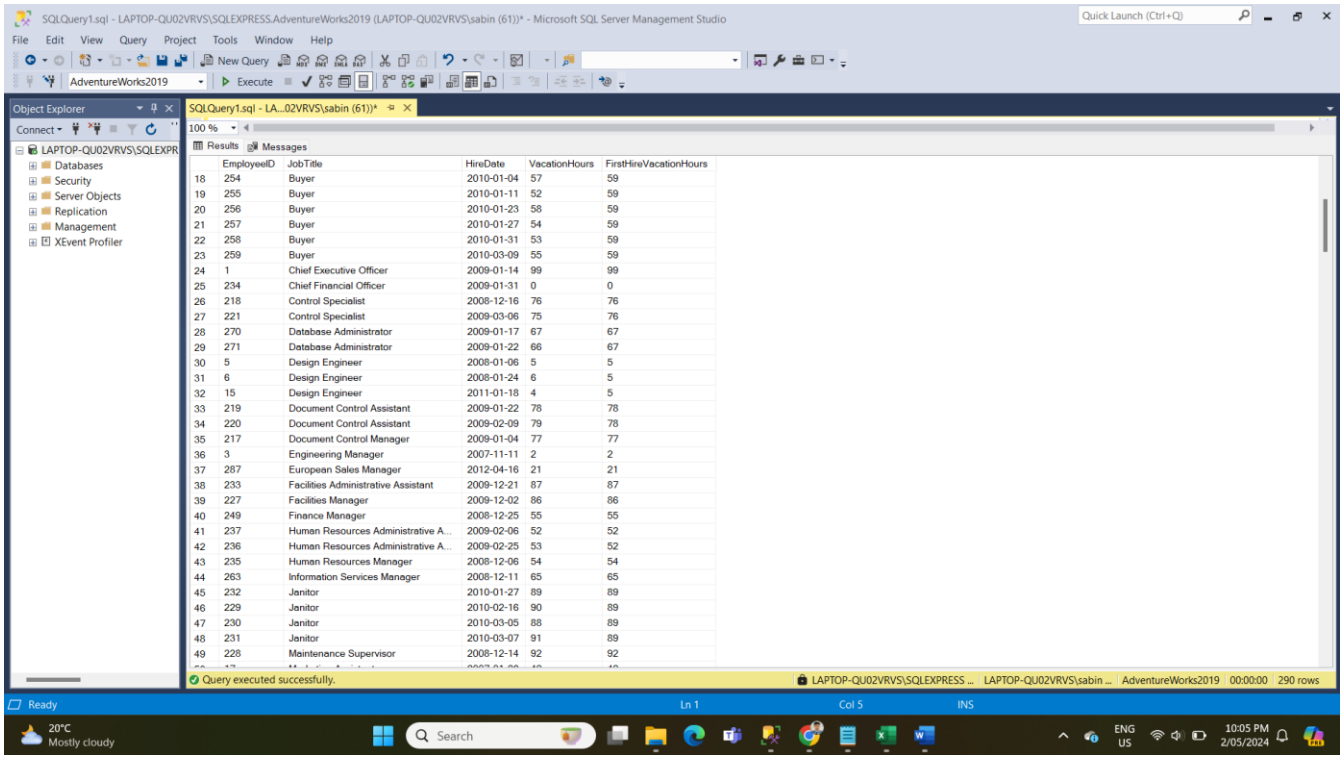
FIRST_VALUE - Exercises

Exercise 1: Create a query that returns all records - and the following columns - from the HumanResources.Employee table:

a. BusinessEntityID (alias this as “EmployeeID”)
b. JobTitle
c. HireDate
d. VacationHours
To make the effect of subsequent steps clearer, also sort the query output by "JobTitle" and HireDate, both in ascending order.
Now add a derived column called “FirstHireVacationHours” that displays – for a given job title – the amount of vacation hours possessed by the first employee hired who has that same job title. For example, if 5 employees have the title “Data Guru”, and the one of those 5 with the oldest hire date has 99 vacation hours, “FirstHireVacationHours” should display “99” for all 5 of those employees’ corresponding records in the query.

```
SELECT
EmployeeID = BusinessEntityID,
JobTitle,
HireDate,
VacationHours,
FirstHireVacationHours = FIRST_VALUE(VacationHours) OVER(PARTITION BY
JobTitle ORDER BY HireDate)
FROM AdventureWorks2019.HumanResources.Employee
ORDER BY JobTitle, HireDate
```





	EmployeeID	JobTitle	HireDate	VacationHours	FirstHireVacationHours
18	254	Buyer	2010-01-04	57	59
19	255	Buyer	2010-01-11	52	59
20	256	Buyer	2010-01-23	58	59
21	257	Buyer	2010-01-27	54	59
22	258	Buyer	2010-01-31	53	59
23	259	Buyer	2010-03-09	55	59
24	1	Chief Executive Officer	2009-01-14	99	99
25	234	Chief Financial Officer	2009-01-31	0	0
26	218	Control Specialist	2008-12-16	76	76
27	221	Control Specialist	2009-03-06	75	76
28	270	Database Administrator	2009-01-17	67	67
29	271	Database Administrator	2009-01-22	66	67
30	5	Design Engineer	2008-01-06	5	5
31	6	Design Engineer	2008-01-24	6	5
32	15	Design Engineer	2011-01-18	4	5
33	219	Document Control Assistant	2009-01-22	78	78
34	220	Document Control Assistant	2009-02-09	79	78
35	217	Document Control Manager	2009-01-04	77	77
36	3	Engineering Manager	2007-11-11	2	2
37	287	European Sales Manager	2012-04-16	21	21
38	233	Facilities Administrative Assistant	2009-12-21	87	87
39	227	Facilities Manager	2009-12-02	86	86
40	249	Finance Manager	2008-12-25	55	55
41	237	Human Resources Administrative A...	2009-02-06	52	52
42	236	Human Resources Administrative A...	2009-02-25	53	52
43	235	Human Resources Manager	2008-12-06	54	54
44	263	Information Services Manager	2008-12-11	65	65
45	232	Janitor	2010-01-27	89	89
46	229	Janitor	2010-02-16	90	89
47	230	Janitor	2010-03-05	88	89
48	231	Janitor	2010-03-07	91	89
49	228	Maintenance Supervisor	2008-12-14	92	92

Exercise 2: Create a query with the following columns:
a. “ProductID” from the Production.Product table
b. “Name” from the Production.Product table (alias this as “ProductName”)
c. “ListPrice” from the Production.ProductListPriceHistory table
d. “ModifiedDate” from the Production.ProductListPriceHistor
You can join the Production.Product and Production.ProductListPriceHistory tables on "ProductID".
Note that the Production.ProductListPriceHistory table contains a distinct record for every different price a product has been listed at. This means that a single product ID may have several records in this table – one for every list price it has had.
Also note that the “ModifiedDate” field in this table displays the effective date of each of these prices. So if there are 3 rows in the table for product ID 12345, the row with the oldest modified date also contains the first price in the associated product’s history. Conversely, the row with the most recent modified date also contains the current price of the product.

To make the effect of subsequent steps clearer, also sort the query output by ProductID and ModifiedDate, both in ascending order.

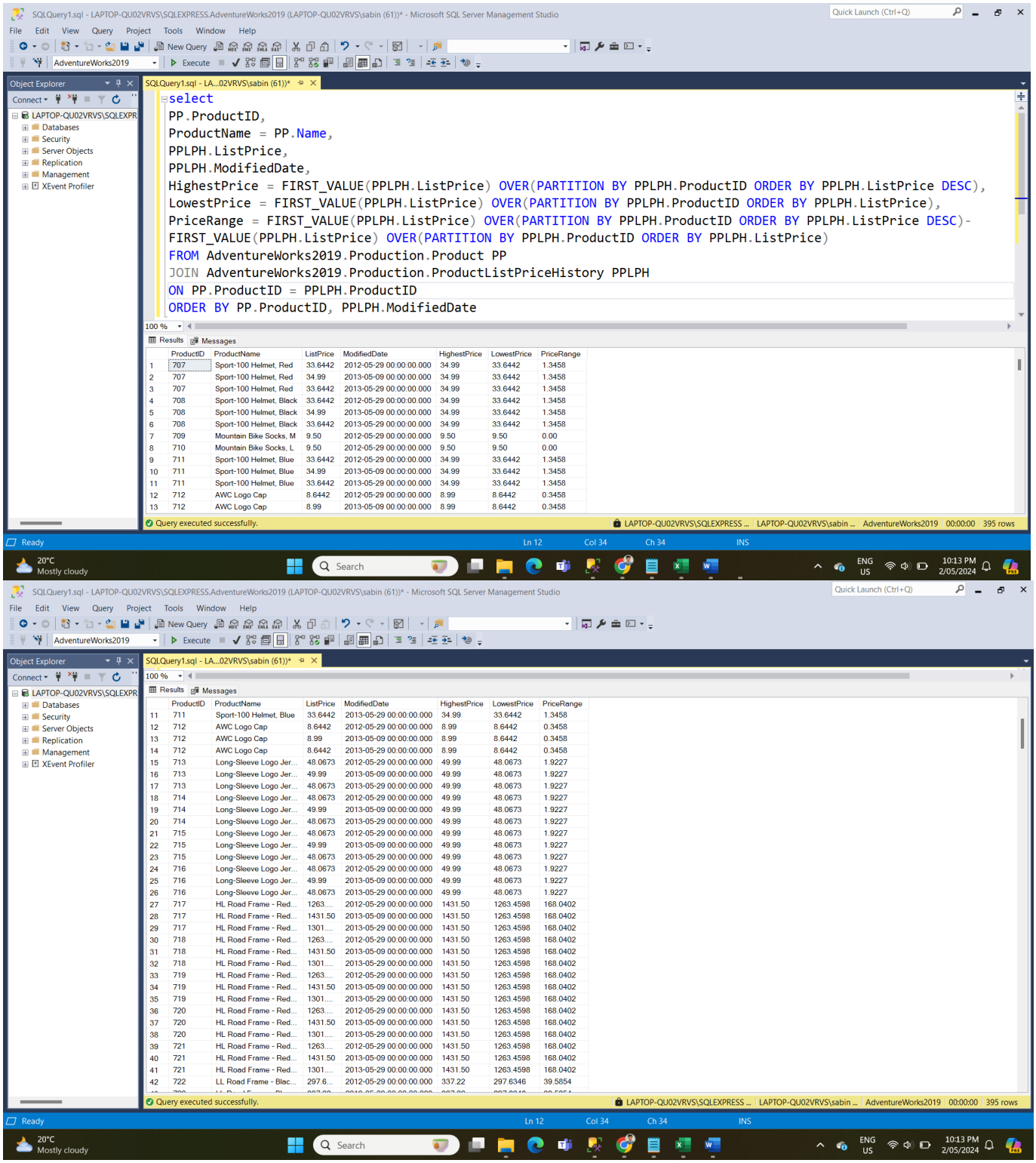
Now add a derived column called “HighestPrice” that displays – for a given product – the highest price that product has been listed at. So even if there

are 4 records for a given product, this column should only display the all-time highest list price for that product in each of those 4 rows.

Similarly, create another derived column called "LowestCost" that displays the all-time lowest price for a given product.

Finally, create a third derived column called "PriceRange" that reflects, for a given product, the difference between its highest and lowest ever list prices.
Answer:

```
select
PP.ProductID,
ProductName = PP.Name,
PPLPH.ListPrice,
PPLPH.ModifiedDate,
HighestPrice = FIRST_VALUE(PPLPH.ListPrice) OVER(PARTITION BY
PPLPH.ProductID ORDER BY PPLPH.ListPrice DESC),
LowestPrice = FIRST_VALUE(PPLPH.ListPrice) OVER(PARTITION BY
PPLPH.ProductID ORDER BY PPLPH.ListPrice),
PriceRange = FIRST_VALUE(PPLPH.ListPrice) OVER(PARTITION BY
PPLPH.ProductID ORDER BY PPLPH.ListPrice DESC)-
FIRST_VALUE(PPLPH.ListPrice) OVER(PARTITION BY PPLPH.ProductID ORDER BY
PPLPH.ListPrice)
FROM AdventureWorks2019.Production.Product PP
JOIN AdventureWorks2019.Production.ProductListPriceHistory PPLPH
ON PP.ProductID = PPLPH.ProductID
ORDER BY PP.ProductID, PPLPH.ModifiedDate
```



Introducing Subqueries - Exercises

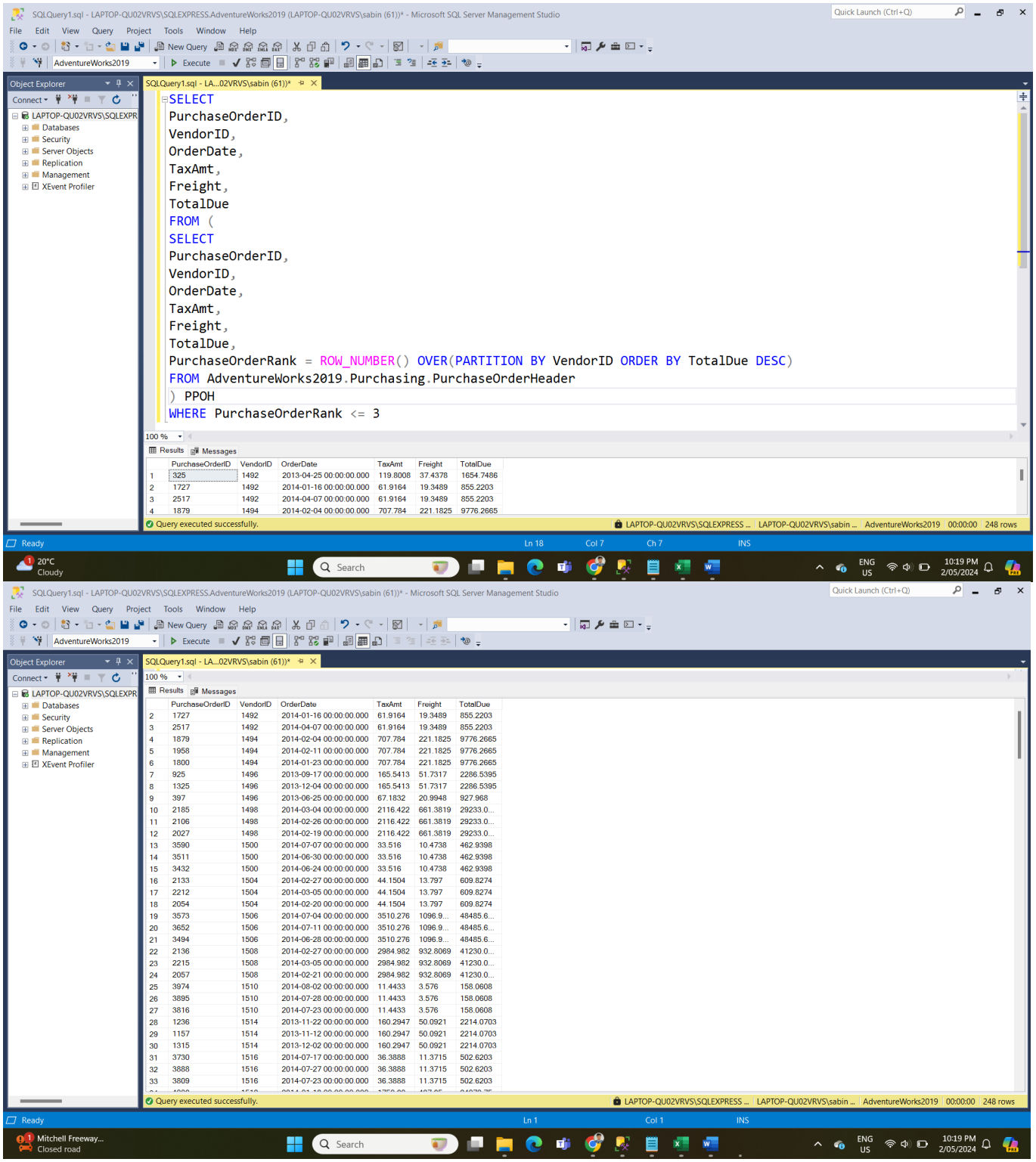
Exercise 1: Write a query that displays the three most expensive orders, per vendor ID, from the Purchasing.PurchaseOrderHeader table. There should

ONLY be three records per Vendor ID, even if some of the total amounts due are identical. "Most expensive" is defined by the amount in the "TotalDue" field.
Include the following fields in your output:
PurchaseOrderID
VendorID
OrderDate
TaxAmt
Freight
TotalDue

Hints:

You will first need to define a field that assigns a unique rank to every purchase order, within each group of like vendor IDs.
You'll probably want to use a Window Function with PARTITION BY and ORDER BY to do this.
The last step will be to apply the appropriate criteria to the field you created with your Window Function.
Answer:

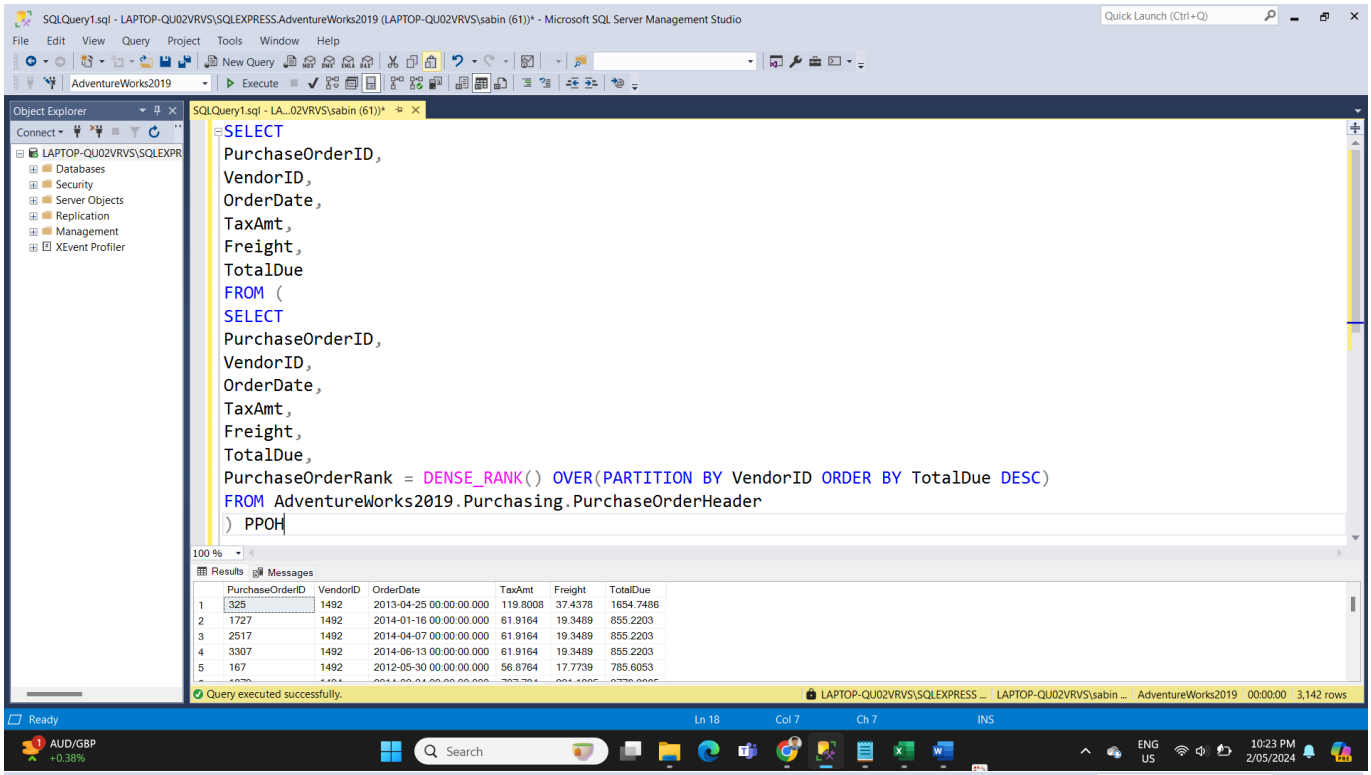
```
SELECT
PurchaseOrderID,
VendorID,
OrderDate,
TaxAmt,
Freight,
TotalDue
FROM (
SELECT
PurchaseOrderID,
VendorID,
OrderDate,
TaxAmt,
Freight,
TotalDue,
PurchaseOrderRank = ROW_NUMBER() OVER(PARTITION BY VendorID ORDER BY TotalDue DESC)
FROM AdventureWorks2019.Purchasing.PurchaseOrderHeader
) PPOH
WHERE PurchaseOrderRank <= 3
```

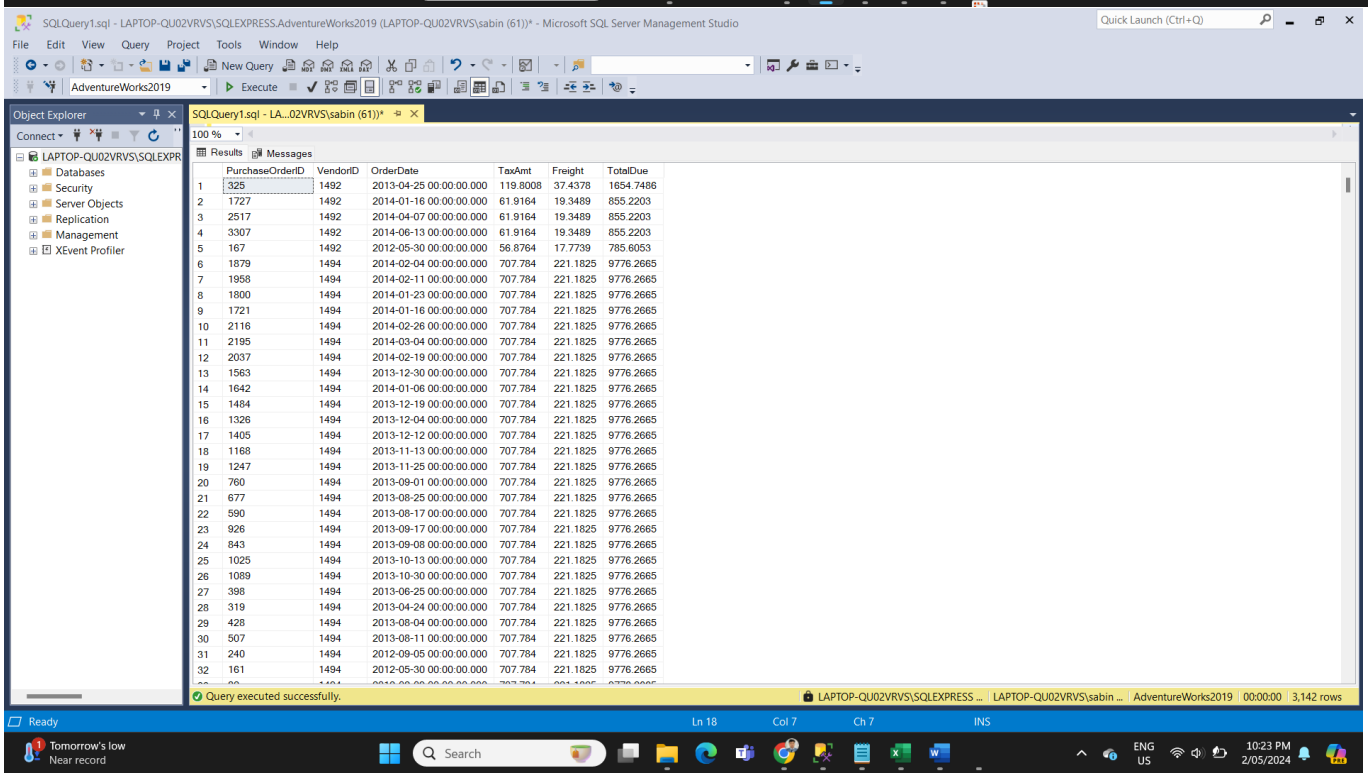

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
AdventureWorks2019
Execute
Object Explorer
Connect
LAPTOP-QU02VRVS\SQLEXPR
Databases
Security
Server Objects
Replication
Management
XEvent Profiler
SQLQuery1.sql - LA...02VRVS\sabin (61))*
SELECT
PurchaseOrderID,
VendorID,
OrderDate,
TaxAmt,
Freight,
TotalDue
FROM (
SELECT
PurchaseOrderID,
VendorID,
OrderDate,
TaxAmt,
Freight,
TotalDue,
PurchaseOrderRank = ROW_NUMBER() OVER(PARTITION BY VendorID ORDER BY TotalDue DESC)
FROM AdventureWorks2019.Purchasing.PurchaseOrderHeader
) PPOH
WHERE PurchaseOrderRank <= 3
100 %
Results
Messages
PurchaseOrderID VendorID OrderDate TaxAmt Freight TotalDue
1 325 1492 2013-04-25 00:00:00.000 119.8008 37.4378 1654.7486
2 1727 1492 2014-01-16 00:00:00.000 61.9164 19.3489 855.2203
3 2517 1492 2014-04-07 00:00:00.000 61.9164 19.3489 855.2203
4 1879 1494 2014-02-04 00:00:00.000 707.784 221.1825 9776.2665
Query executed successfully.
LAPTOP-QU02VRVS\SQLEXPRESS ...
LAPTOP-QU02VRVS\sabin ...
AdventureWorks2019
00:00:00
248 rows
Ready
Ln 18
Col 7
Ch 7
INS
20°C
Cloudy
Search
ENG
US
10:19 PM
2/05/2024

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
AdventureWorks2019
Execute
Object Explorer
Connect
LAPTOP-QU02VRVS\SQLEXPR
Databases
Security
Server Objects
Replication
Management
XEvent Profiler
SQLQuery1.sql - LA...02VRVS\sabin (61))*
100 %
Results
Messages
PurchaseOrderID VendorID OrderDate TaxAmt Freight TotalDue
2 1727 1492 2014-01-16 00:00:00.000 61.9164 19.3489 855.2203
3 2517 1492 2014-04-07 00:00:00.000 61.9164 19.3489 855.2203
4 1879 1494 2014-02-04 00:00:00.000 707.784 221.1825 9776.2665
5 1958 1494 2014-02-11 00:00:00.000 707.784 221.1825 9776.2665
6 1800 1494 2014-01-23 00:00:00.000 707.784 221.1825 9776.2665
7 925 1496 2013-09-17 00:00:00.000 165.5413 51.7317 2286.5395
8 1325 1496 2013-12-04 00:00:00.000 165.5413 51.7317 2286.5395
9 397 1496 2013-06-25 00:00:00.000 67.1832 20.9948 927.968
10 2185 1498 2014-03-04 00:00:00.000 2116.422 661.3819 29233.0...
11 2106 1498 2014-02-26 00:00:00.000 2116.422 661.3819 29233.0...
12 2027 1498 2014-02-19 00:00:00.000 2116.422 661.3819 29233.0...
13 3590 1500 2014-07-07 00:00:00.000 33.516 10.4738 462.9398
14 3511 1500 2014-06-30 00:00:00.000 33.516 10.4738 462.9398
15 3432 1500 2014-06-24 00:00:00.000 33.516 10.4738 462.9398
16 2133 1504 2014-02-27 00:00:00.000 44.1504 13.797 609.8274
17 2212 1504 2014-03-05 00:00:00.000 44.1504 13.797 609.8274
18 2054 1504 2014-02-20 00:00:00.000 44.1504 13.797 609.8274
19 3573 1506 2014-07-04 00:00:00.000 3510.276 1096.9... 48485.6...
20 3652 1506 2014-07-11 00:00:00.000 3510.276 1096.9... 48485.6...
21 3494 1506 2014-06-28 00:00:00.000 3510.276 1096.9... 48485.6...
22 2136 1508 2014-02-27 00:00:00.000 2984.982 932.8069 41230.0...
23 2215 1508 2014-03-05 00:00:00.000 2984.982 932.8069 41230.0...
24 2057 1508 2014-02-21 00:00:00.000 2984.982 932.8069 41230.0...
25 3974 1510 2014-08-02 00:00:00.000 11.4433 3.576 158.0608
26 3895 1510 2014-07-28 00:00:00.000 11.4433 3.576 158.0608
27 3816 1510 2014-07-23 00:00:00.000 11.4433 3.576 158.0608
28 1236 1514 2013-11-22 00:00:00.000 160.2947 50.0921 2214.0703
29 1157 1514 2013-11-12 00:00:00.000 160.2947 50.0921 2214.0703
30 1315 1514 2013-12-02 00:00:00.000 160.2947 50.0921 2214.0703
31 3730 1516 2014-07-17 00:00:00.000 36.3888 11.3715 502.6203
32 3888 1516 2014-07-27 00:00:00.000 36.3888 11.3715 502.6203
33 3809 1516 2014-07-23 00:00:00.000 36.3888 11.3715 502.6203
Query executed successfully.
LAPTOP-QU02VRVS\SQLEXPRESS ...
LAPTOP-QU02VRVS\sabin ...
AdventureWorks2019
00:00:00
248 rows
Ready
Ln 1
Col 1
INS
Mitchell Freeway...
Closed road
Search
ENG
US
10:19 PM
2/05/2024

Exercise 2: Modify your query from the first problem, such that the top three purchase order amounts are returned, regardless of how many records are returned per Vendor Id.
In other words, if there are multiple orders with the same total due amount, all should be returned as long as the total due amount for these orders is one of the top three.
Ultimately, you should see three distinct total due amounts (i.e., the top three) for each group of like Vendor Ids. However, there could be multiple records for each of these amounts.
Hint: Think carefully about how the different ranking functions (ROW_NUMBER, RANK, and DENSE_RANK) work, and which one might be best suited to help you here.
Answer:

```
SELECT
PurchaseOrderID,
VendorID,
OrderDate,
TaxAmt,
Freight,
TotalDue
FROM (
SELECT
PurchaseOrderID,
VendorID,
OrderDate,
TaxAmt,
Freight,
TotalDue,
PurchaseOrderRank = DENSE_RANK() OVER(PARTITION BY VendorID ORDER BY
TotalDue DESC)
FROM AdventureWorks2019.Purchasing.PurchaseOrderHeader
) PPOH
```



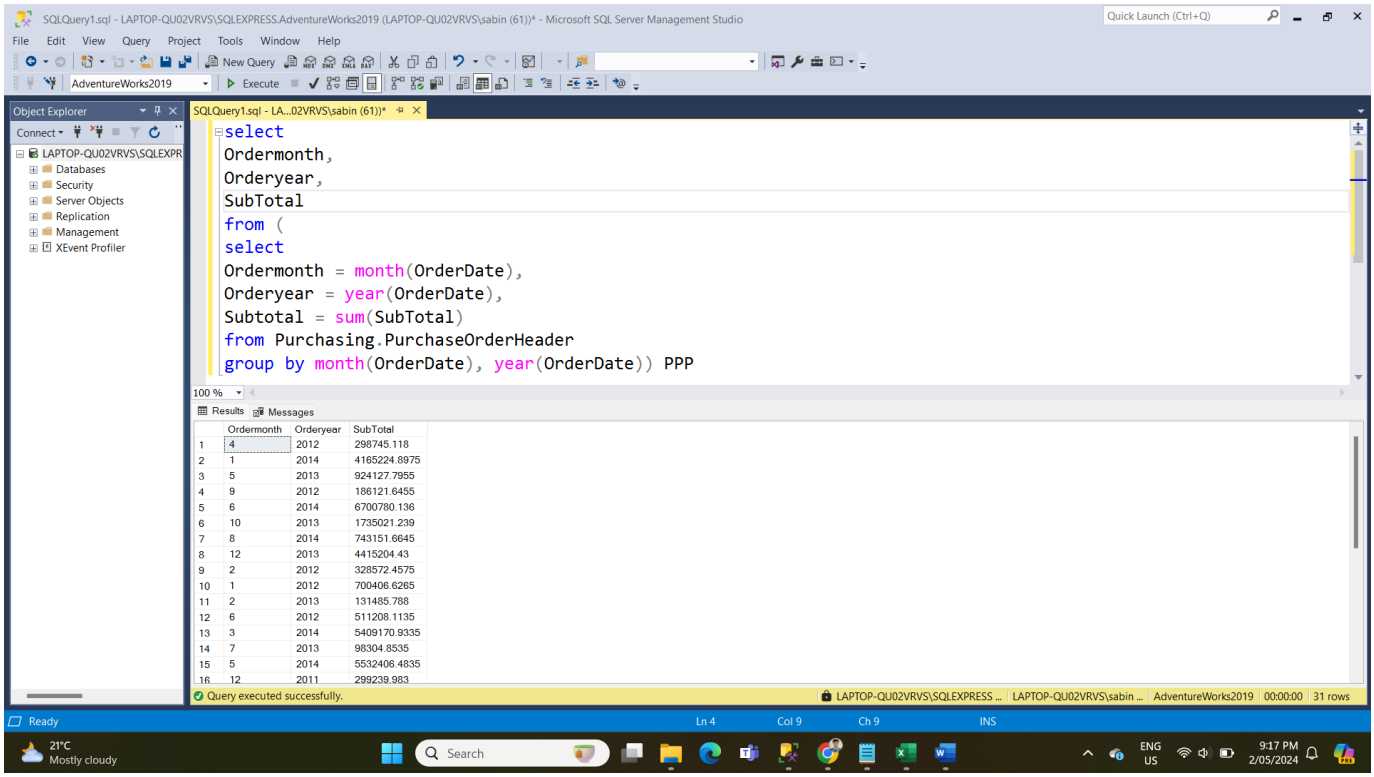


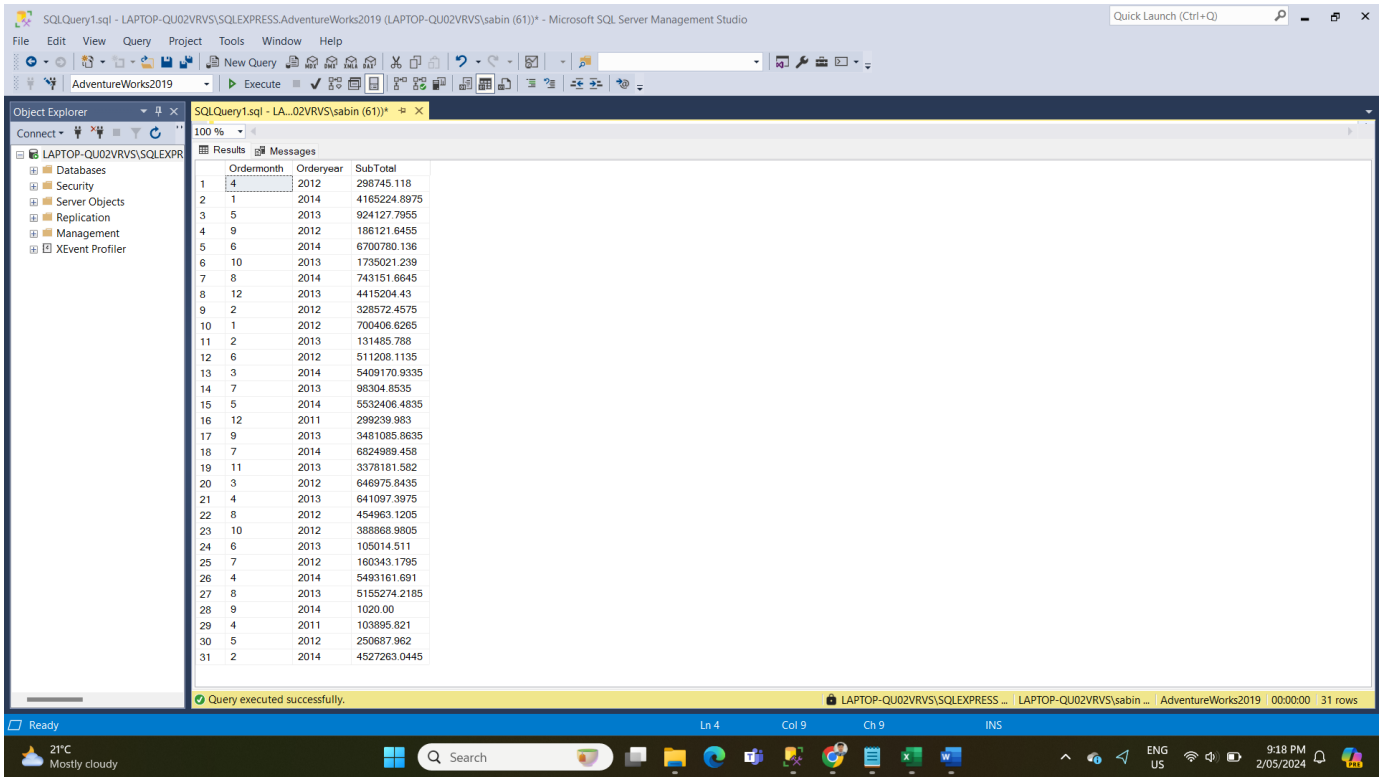
ROWS BETWEEN - Exercises

Exercise 1: Create a query with the following columns:

“OrderMonth”, a derived column (you’ll have to create this one yourself) featuring the month number corresponding with the Order Date in a given row.
“OrderYear”, a derived column featuring the year corresponding with the Order Date in a given row.
“SubTotal” from the Purchasing.PurchaseOrderHeader table
Your query should be an aggregate query – specifically, it should sum “SubTotal”, and group by the remaining fields.
Answer:

```
select
Ordermonth,
Orderyear,
SubTotal
from (
select
Ordermonth = month(OrderDate),
Orderyear = year(OrderDate),
Subtotal = sum(SubTotal)
from Purchasing.PurchaseOrderHeader
group by month(OrderDate), year(OrderDate)) PPP
```





	Ordermonth	Orderyear	SubTotal
1	4	2012	298745.118
2	1	2014	4165224.8975
3	5	2013	924127.7955
4	9	2012	186121.6455
5	6	2014	6700780.136
6	10	2013	1735021.239
7	8	2014	743151.6645
8	12	2013	4415204.43
9	2	2012	328572.4575
10	1	2012	700406.6265
11	2	2013	131485.788
12	6	2012	511208.1135
13	3	2014	5409170.9335
14	7	2013	98304.8535
15	5	2014	5532406.4835
16	12	2011	299239.983
17	9	2013	3481085.8635
18	7	2014	6824989.458
19	11	2013	3378181.582
20	3	2012	646975.8435
21	4	2013	641097.3975
22	8	2012	454963.1205
23	10	2012	388868.9805
24	6	2013	105014.511
25	7	2012	160343.1795
26	4	2014	5493161.691
27	8	2013	5155274.2185
28	9	2014	1020.00
29	4	2011	103895.821
30	5	2012	250687.962
31	2	2014	4527263.0445

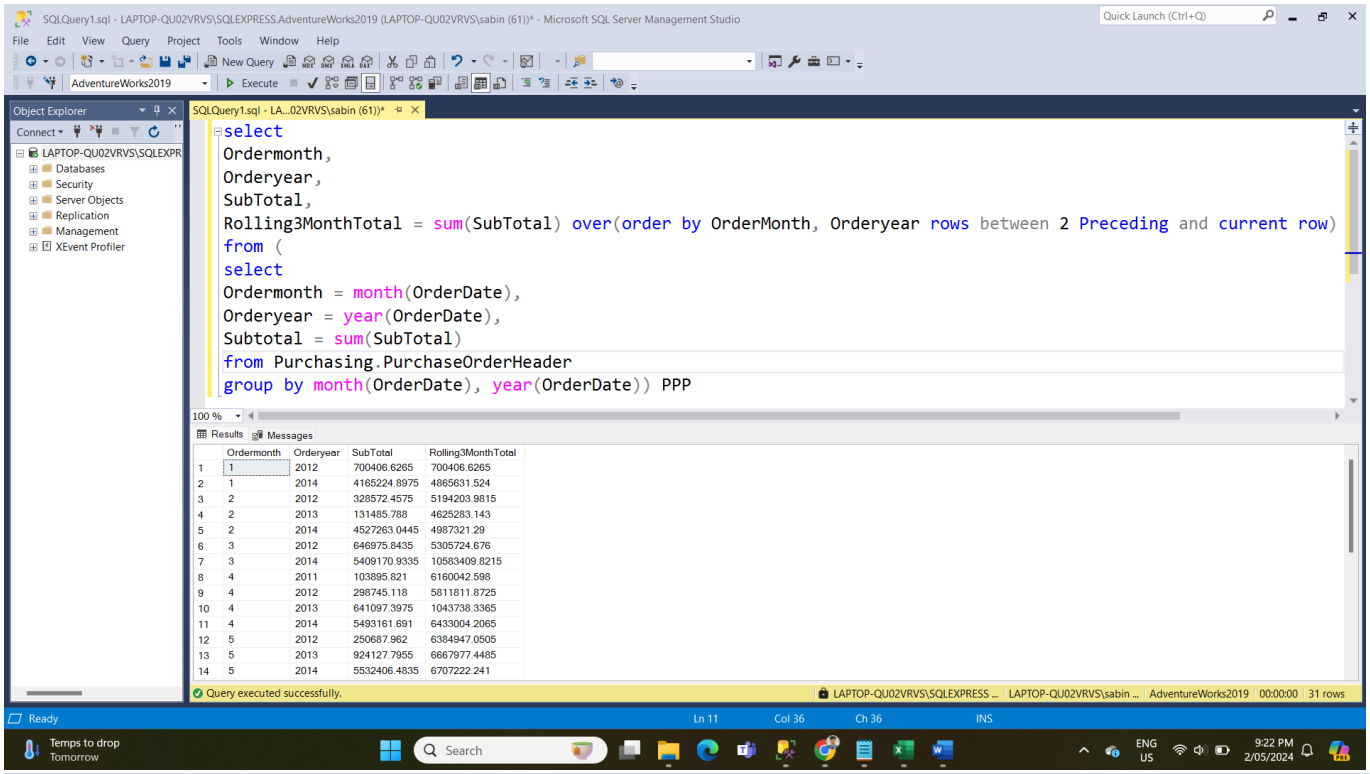
Exercise 2: Modify your query from Exercise 1 by adding a derived column called "Rolling3MonthTotal", that displays - for a given row - a running total of "SubTotal" for the prior three months (including the current row).

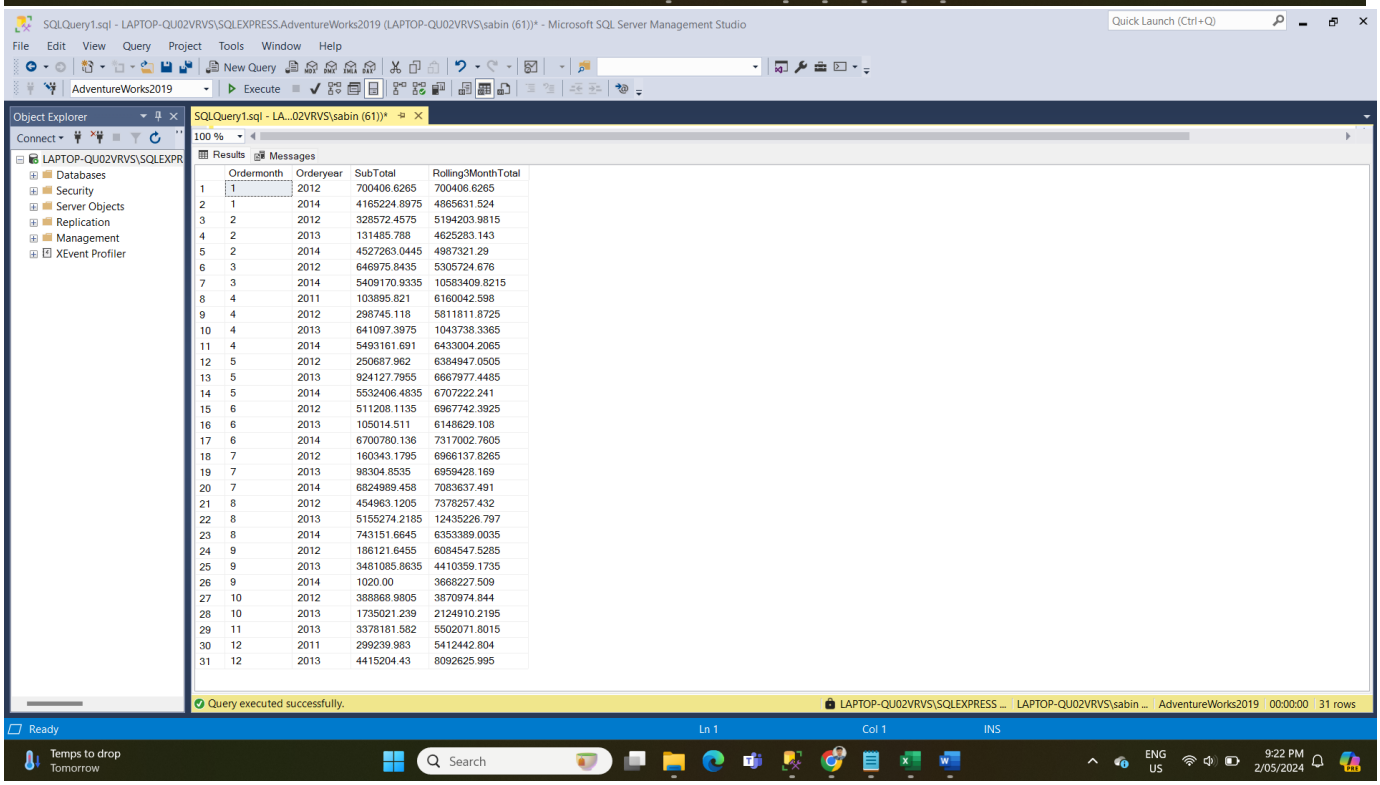
HINT: You will need to include multiple fields in your ORDER BY to get this to work!

Answer:

```
select

Ordermonth,

Orderyear,

SubTotal,

Rolling3MonthTotal = sum(SubTotal) over(order by OrderMonth, Orderyear
rows between 2 Preceding and current row)

from (

select
```

```
Ordermonth = month(OrderDate),
Orderyear = year(OrderDate),
Subtotal = sum(SubTotal)
from Purchasing.PurchaseOrderHeader
group by month(OrderDate), year(OrderDate)) PPP
```

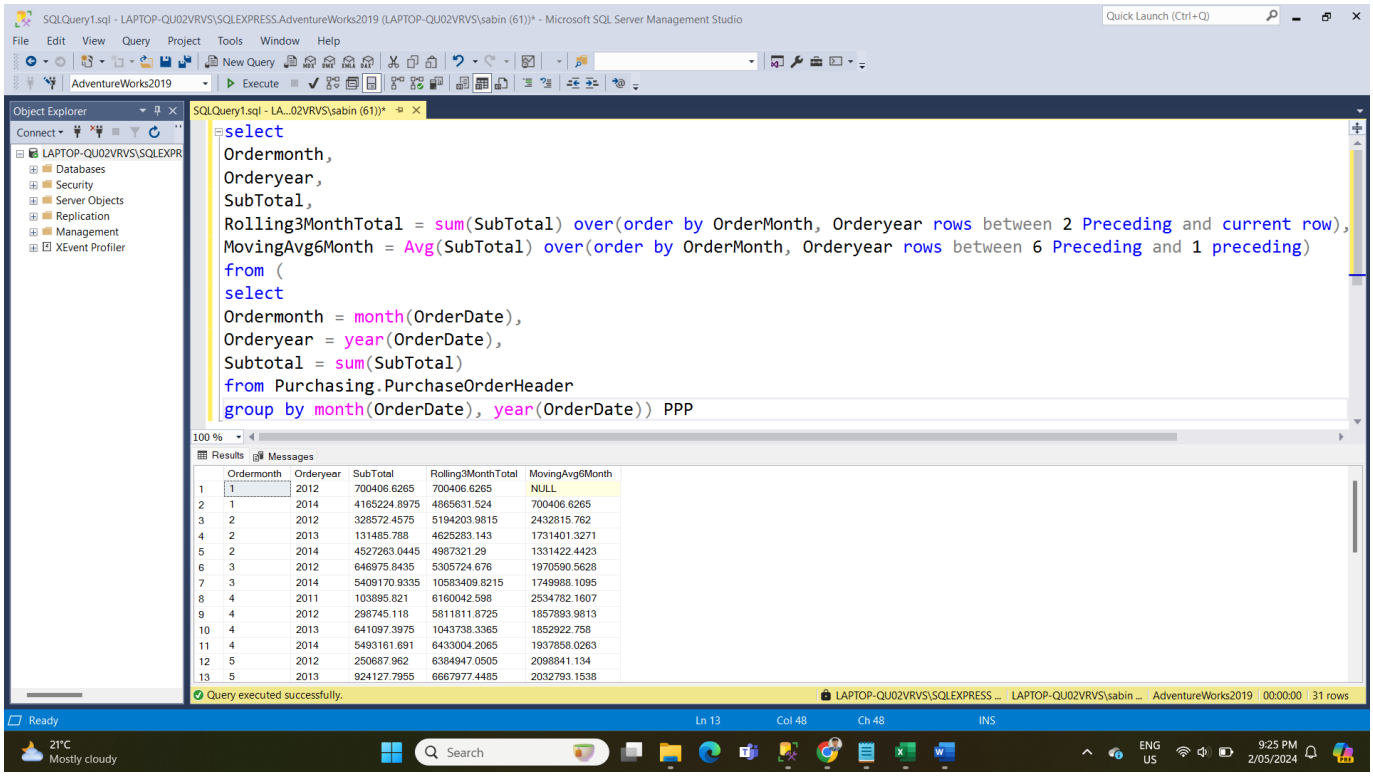

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
AdventureWorks2019
Object Explorer
Connect
LAPTOP-QU02VRVS\SQLEXPR
Databases
Security
Server Objects
Replication
Management
XEvent Profiler
SQLQuery1.sql - LA...02VRVS\sabin (61))*
select
Ordermonth,
Orderyear,
SubTotal,
Rolling3MonthTotal = sum(SubTotal) over(order by OrderMonth, Orderyear rows between 2 Preceding and current row)
from (
select
Ordermonth = month(OrderDate),
Orderyear = year(OrderDate),
Subtotal = sum(SubTotal)
from Purchasing.PurchaseOrderHeader
group by month(OrderDate), year(OrderDate)) PPP
100 %
Results
Messages
Ordermonth Orderyear SubTotal Rolling3MonthTotal
1 1 2012 700406.6265 700406.6265
2 1 2014 4165224.8975 4865631.524
3 2 2012 328572.4575 5194203.9815
4 2 2013 131485.788 4625283.143
5 2 2014 4527263.0445 4987321.29
6 3 2012 646975.8435 5305724.676
7 3 2014 5409170.9335 10583409.8215
8 4 2011 103895.821 6160042.598
9 4 2012 298745.118 5811811.8725
10 4 2013 641097.3975 1043738.3365
11 4 2014 5493161.691 6433004.2065
12 5 2012 250687.962 6384947.0505
13 5 2013 924127.7955 6667977.4485
14 5 2014 5532406.4835 6707222.241
Query executed successfully.
LAPTOP-QU02VRVS\SQLEXPRESS ...
LAPTOP-QU02VRVS\sabin ...
AdventureWorks2019
00:00:00
31 rows
Ready
Ln 11
Col 36
Ch 36
INS
Temps to drop
Tomorrow
Search
ENG
US
9:22 PM
2/05/2024

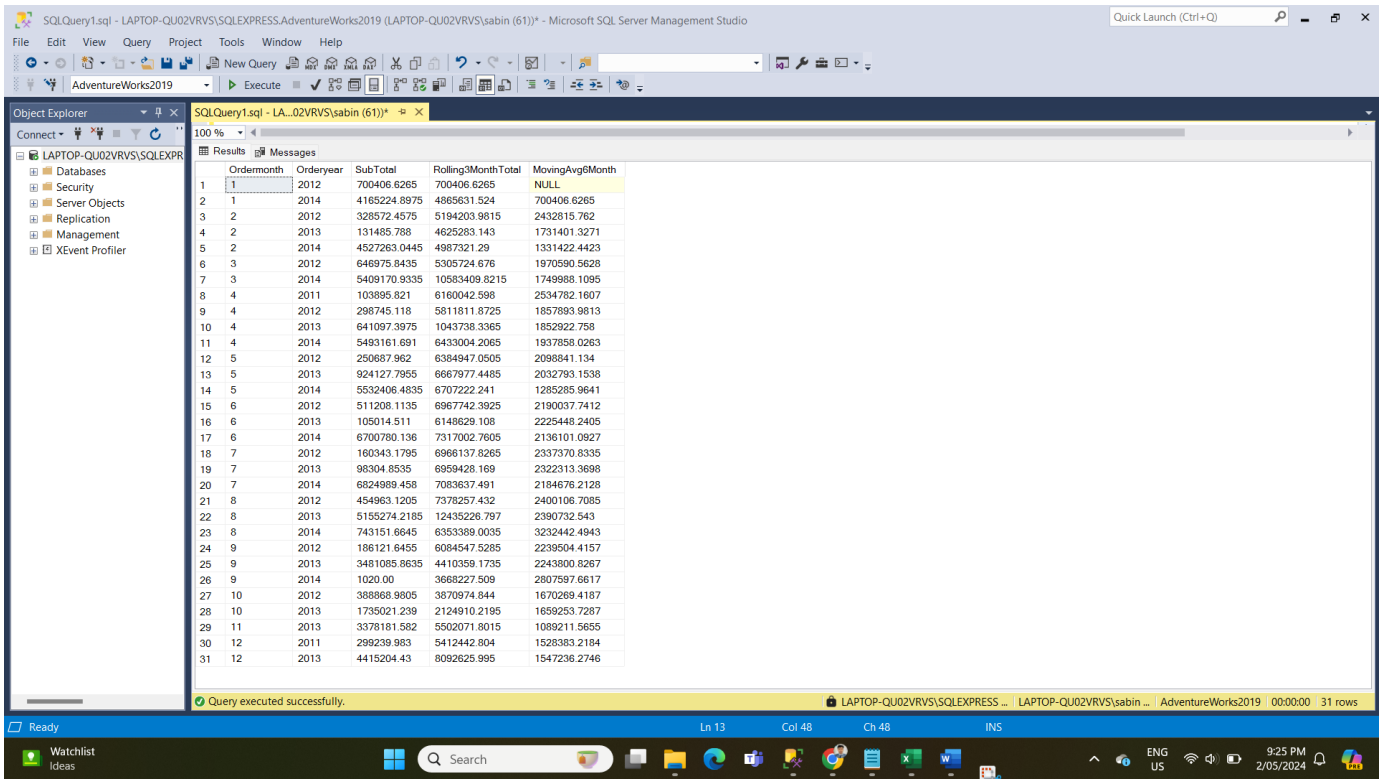

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
AdventureWorks2019
Object Explorer
Connect
LAPTOP-QU02VRVS\SQLEXPR
Databases
Security
Server Objects
Replication
Management
XEvent Profiler
SQLQuery1.sql - LA...02VRVS\sabin (61))*
100 %
Results
Messages
Ordermonth Orderyear SubTotal Rolling3MonthTotal
1 1 2012 700406.6265 700406.6265
2 1 2014 4165224.8975 4865631.524
3 2 2012 328572.4575 5194203.9815
4 2 2013 131485.788 4625283.143
5 2 2014 4527263.0445 4987321.29
6 3 2012 646975.8435 5305724.676
7 3 2014 5409170.9335 10583409.8215
8 4 2011 103895.821 6160042.598
9 4 2012 298745.118 5811811.8725
10 4 2013 641097.3975 1043738.3365
11 4 2014 5493161.691 6433004.2065
12 5 2012 250687.962 6384947.0505
13 5 2013 924127.7955 6667977.4485
14 5 2014 5532406.4835 6707222.241
15 6 2012 511208.1135 6967742.3925
16 6 2013 105014.511 6148629.108
17 6 2014 6700780.136 7317002.7605
18 7 2012 160343.1795 6966137.8265
19 7 2013 98304.8535 6959428.169
20 7 2014 6824989.458 7083637.491
21 8 2012 454963.1205 7378257.432
22 8 2013 5155274.2185 12435226.797
23 8 2014 743151.6645 6353389.0035
24 9 2012 186121.6455 6084547.5285
25 9 2013 3481085.8635 4410359.1735
26 9 2014 1020.00 3668227.509
27 10 2012 388868.9805 3870974.844
28 10 2013 1735021.239 2124910.2195
29 11 2013 3378181.582 5502071.8015
30 12 2011 299239.983 5412442.804
31 12 2013 4415204.43 8092625.995
Query executed successfully.
LAPTOP-QU02VRVS\SQLEXPRESS ...
LAPTOP-QU02VRVS\sabin ...
AdventureWorks2019
00:00:00
31 rows
Ready
Ln 1
Col 1
INS
Temps to drop
Tomorrow
Search
ENG
US
9:22 PM
2/05/2024

Exercise 3: Modify your query from Exercise 3 by adding another derived column called "MovingAvg6Month", that calculates a rolling average of "SubTotal" for the previous 6 months, relative to the month in the "current" row. Note that this average should NOT include the current row.

Answer:

```
select

Ordermonth,

Orderyear,

SubTotal,

Rolling3MonthTotal = sum(SubTotal) over(order by OrderMonth, Orderyear rows between 2 Preceding and current row),

MovingAvg6Month = Avg(SubTotal) over(order by OrderMonth, Orderyear rows between 6 Preceding and 1 preceding)

from (

select

Ordermonth = month(OrderDate),

Orderyear = year(OrderDate),

Subtotal = sum(SubTotal)

from Purchasing.PurchaseOrderHeader

group by month(OrderDate), year(OrderDate)) PPP
```



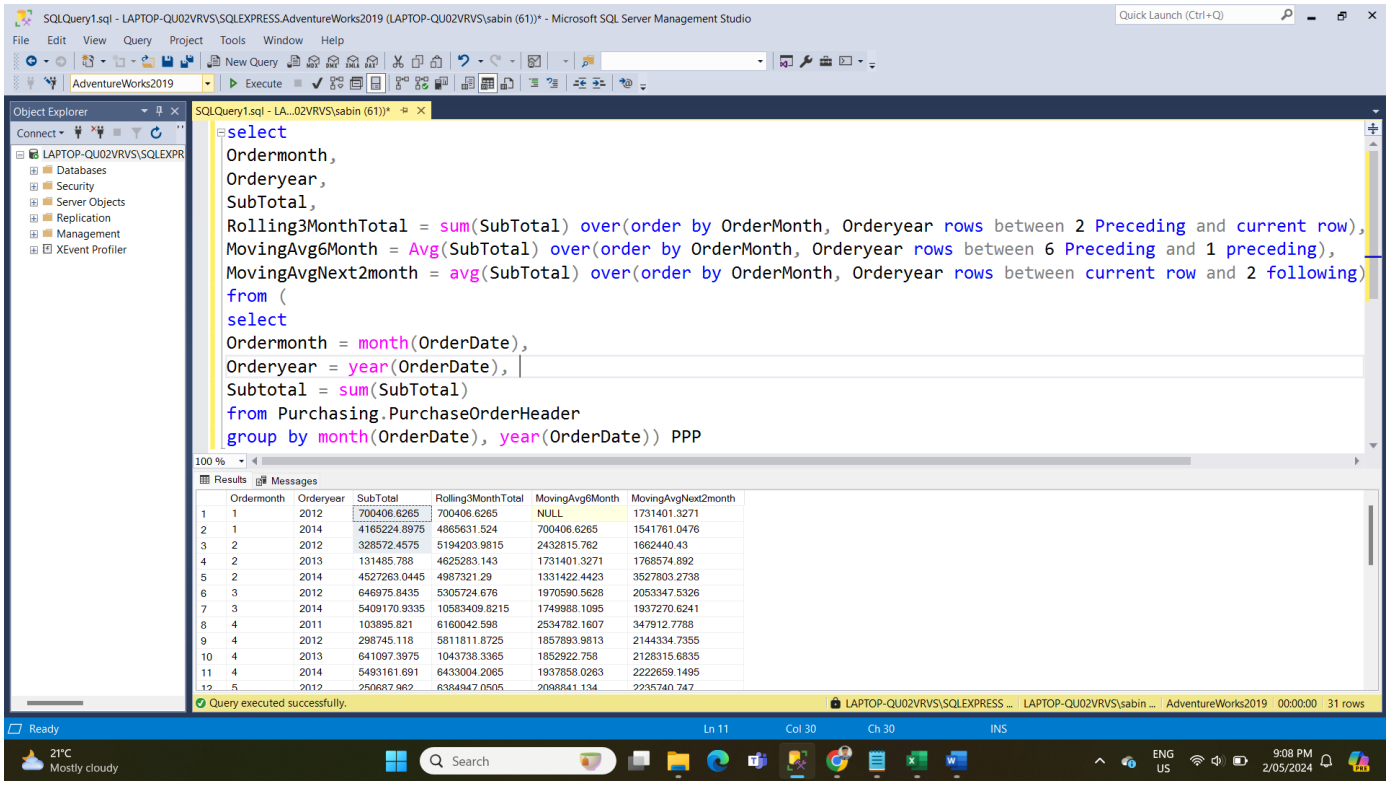


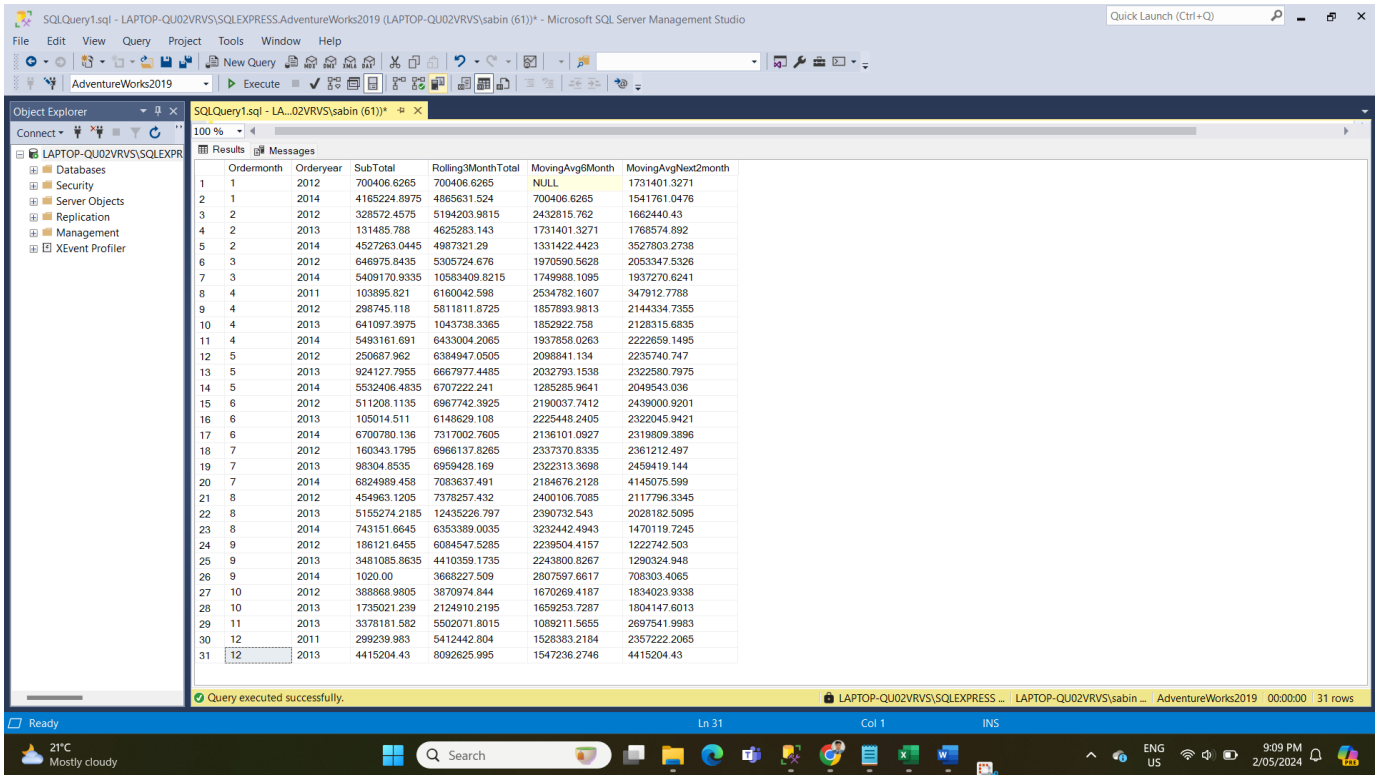
Exercise 4: Modify your query from Exercise 3 by adding (yet) another derived column called “MovingAvgNext2Months” , that calculates a rolling average of “SubTotal” for the month in the current row and the next two

months after that. This moving average will provide a kind of "forecast" for Subtotal by month.

Answer:

```
select
Ordermonth,
Orderyear,
SubTotal,
Rolling3MonthTotal = sum(SubTotal) over(order by OrderMonth, Orderyear
rows between 2 Preceding and current row),
MovingAvg6Month = Avg(SubTotal) over(order by OrderMonth, Orderyear
rows between 6 Preceding and 1 preceding),
MovingAvgNext2month = avg(SubTotal) over(order by OrderMonth, Orderyear
rows between current row and 2 following)
from (
select
Ordermonth = month(OrderDate),
Orderyear = year(OrderDate),
Subtotal = sum(SubTotal)
from Purchasing.PurchaseOrderHeader
group by month(OrderDate), year(OrderDate)) PPP
```





	Ordermonth	Orderyear	SubTotal	Rolling3MonthTotal	MovingAvg6Month	MovingAvgNext2month
1	1	2012	700406.6265	700406.6265	NULL	1731401.3271
2	1	2014	4165224.8975	4865631.524	700406.6265	1541761.0476
3	2	2012	328572.4575	5194203.9815	2432815.762	1662440.43
4	2	2013	131485.788	4625283.143	1731401.3271	1768574.892
5	2	2014	4527263.0445	4987321.29	1331422.4423	3527803.2738
6	3	2012	646975.8435	5305724.676	1970590.5628	2053347.5326
7	3	2014	5409170.9335	10583409.8215	1749988.1095	1937270.6241
8	4	2011	103895.821	6160042.598	2534782.1607	347912.7788
9	4	2012	298745.118	5811811.8725	1857893.9813	2144334.7355
10	4	2013	641097.3975	1043738.3365	1852922.758	2128315.6835
11	4	2014	5493161.691	6433004.2065	1937858.0263	2222659.1495
12	5	2012	250687.962	6384947.0505	2098841.134	2235740.747
13	5	2013	924127.7955	6667977.4485	2032793.1538	2322580.7975
14	5	2014	5532406.4835	6707222.241	1285285.9641	2049543.036
15	6	2012	511208.1135	6967742.3925	2190037.7412	2439000.9201
16	6	2013	105014.511	6148629.108	2225448.2405	2322045.9421
17	6	2014	6700780.136	7317002.7605	2136101.0927	2319809.3896
18	7	2012	160343.1795	6966137.8265	2337370.8335	2361212.497
19	7	2013	98304.8535	6959428.169	2322313.3698	2459419.144
20	7	2014	6824989.458	7083637.491	2184676.2128	4145075.599
21	8	2012	454963.1205	7378257.432	2400106.7085	2117796.3345
22	8	2013	5155274.2185	12435226.797	2390732.543	2028182.5095
23	8	2014	743151.6645	6353389.0035	3232442.4943	1470119.7245
24	9	2012	186121.6455	6084547.5285	2239504.4157	1222742.503
25	9	2013	3481085.8635	4410359.1735	2243800.8267	1290324.948
26	9	2014	1020.00	3668227.509	2807597.6617	708303.4065
27	10	2012	388868.9805	3870974.844	1670269.4187	1834023.9338
28	10	2013	1735021.239	2124910.2195	1659253.7287	1804147.6013
29	11	2013	3378181.582	5502071.8015	1089211.5655	2697541.9983
30	12	2011	299239.983	5412442.804	1528383.2184	2357222.2065
31	12	2013	4415204.43	8092625.995	1547236.2746	4415204.43