

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
```

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (62))* - Microsoft SQL Server Management Studio

```

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

```

100 %

Results Messages

	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

Query executed successfully.

Ready 13°C Mostly cloudy Search 1:13 AM 28/04/2024

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (62))* - Microsoft SQL Server Management Studio

```

SELECT
    FirstName,
    LastName,
    JobTitle,
    Rate,
    AverageRate
FROM AdventureWorks2019.HumanResources.EmployeePayHistory
ORDER BY Rate DESC

```

100 %

Results Messages

	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	Rahem	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	Salavarria	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	Eminizer	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

Query executed successfully.

Ready 13°C Mostly cloudy Search 1:33 AM 28/04/2024

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()

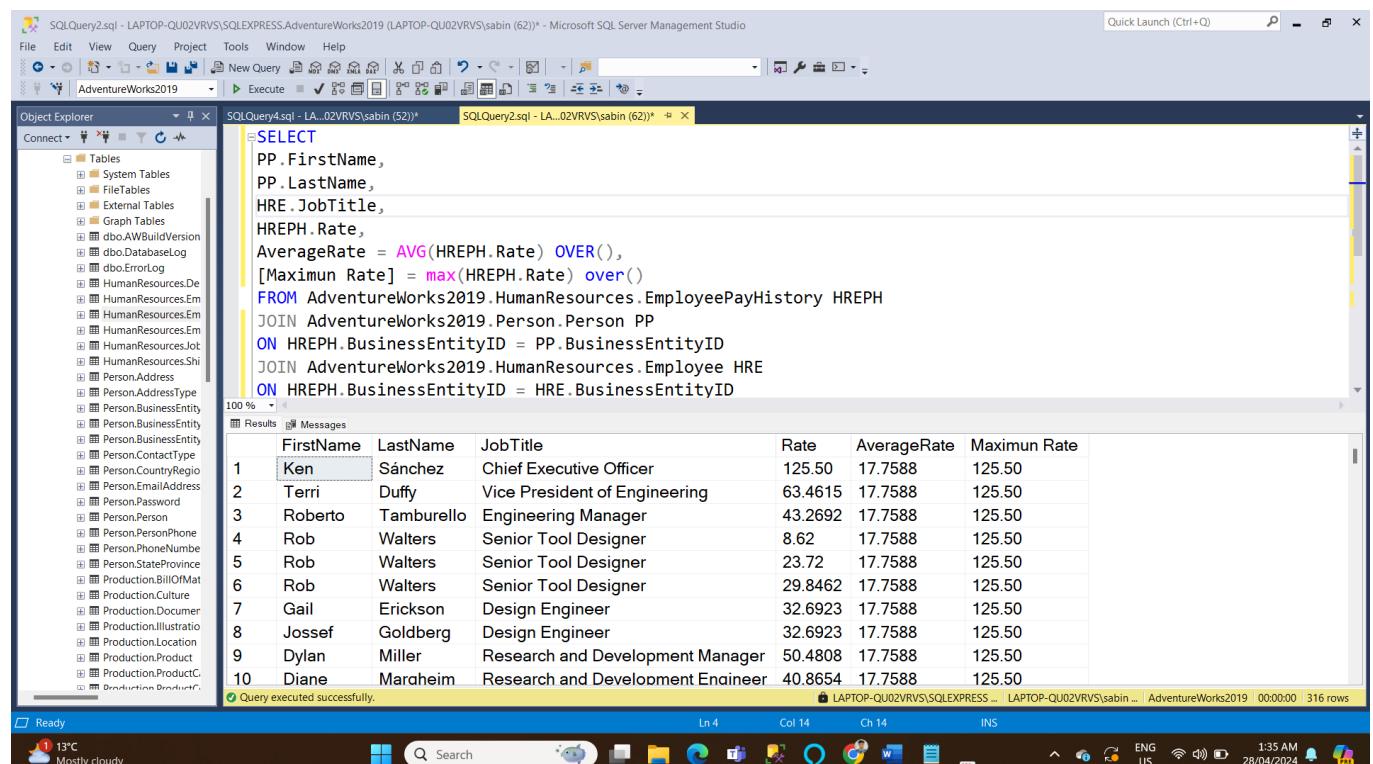
FROM AdventureWorks2019.HumanResources.EmployeePayHistory HREPH

JOIN AdventureWorks2019.Person.Person PP

ON HREPH.BusinessEntityID = PP.BusinessEntityID

JOIN AdventureWorks2019.HumanResources.Employee HRE

ON HREPH.BusinessEntityID = HRE.BusinessEntityID



The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the T-SQL code for Exercise 2. The results grid shows 10 rows of data with columns: FirstName, LastName, JobTitle, Rate, AverageRate, and Maximum Rate. The data is as follows:

	FirstName	LastName	JobTitle	Rate	AverageRate	Maximum 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	Marheim	Research and Development Engineer	40.8654	17.7588	125.50

At the bottom of the results grid, it says "Query executed successfully."

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'Tables' node is expanded, showing numerous tables such as 'HumanResources.Employee', 'Person.Address', and 'Production.Product'. In the center pane, a results grid displays the following data:

	FirstName	LastName	JobTitle	Rate	AverageRate	Maximum 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
11	Gigi	Matthew	Research and Development Engineer	40.8654	17.7588	125.50
12	Michael	Raheem	Research and Development Manager	42.4808	17.7588	125.50
13	Ovidiu	Craciun	Senior Tool Designer	28.8462	17.7588	125.50
14	Thierry	D'Hers	Tool Designer	25.00	17.7588	125.50
15	Janice	Galvin	Tool Designer	25.00	17.7588	125.50
16	Michael	Sullivan	Senior Design Engineer	36.0577	17.7588	125.50
17	Sharon	Salavarria	Design Engineer	32.6923	17.7588	125.50
18	David	Bradley	Marketing Manager	24.00	17.7588	125.50
19	David	Bradley	Marketing Manager	28.75	17.7588	125.50
20	David	Bradley	Marketing Manager	37.50	17.7588	125.50
21	Kevin	Brown	Marketing Assistant	13.4615	17.7588	125.50

Query executed successfully.

Exercise 3:

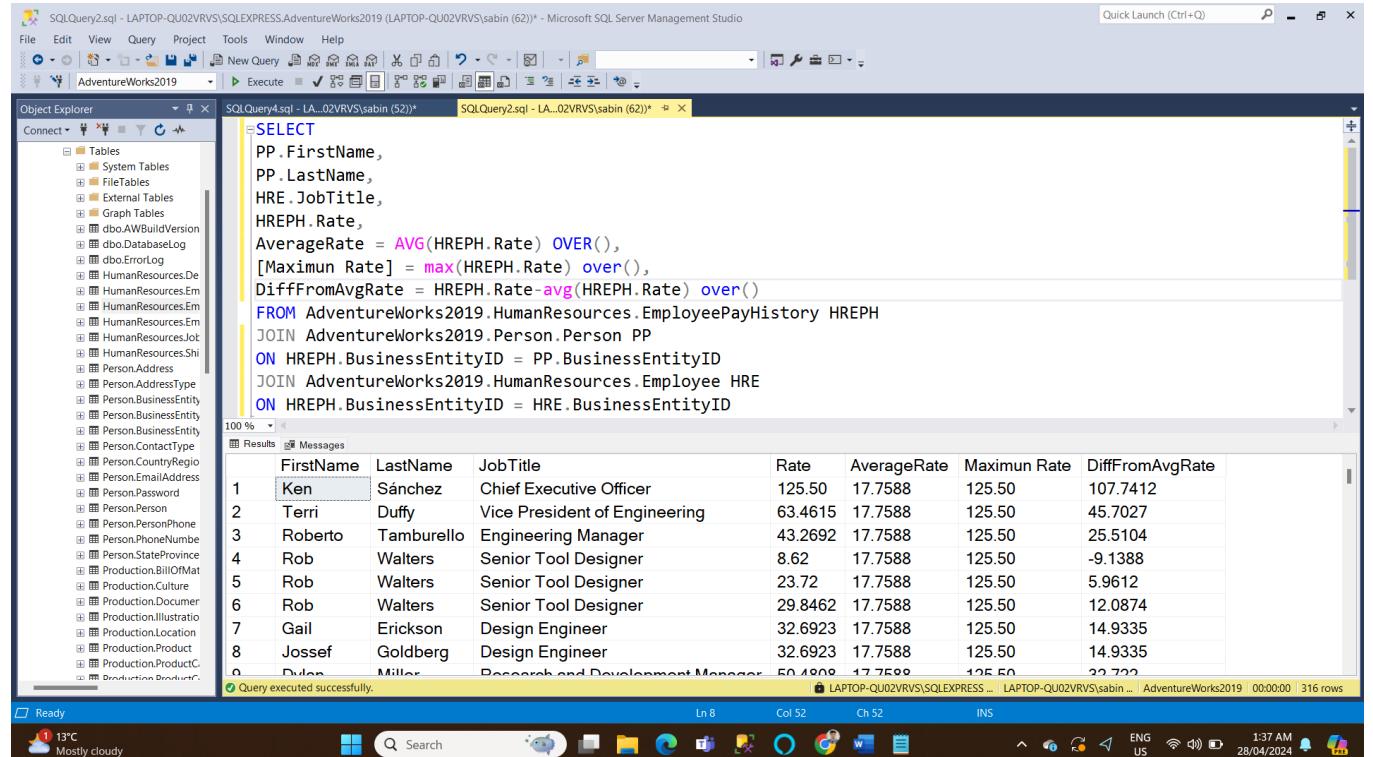
Enhance your query from Exercise 2 by adding a derived column called "DiffFromAvgRate" that returns the result of the following calculation: An employee's pay rate, MINUS the average of all values in the "Rate" column.

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
    
```

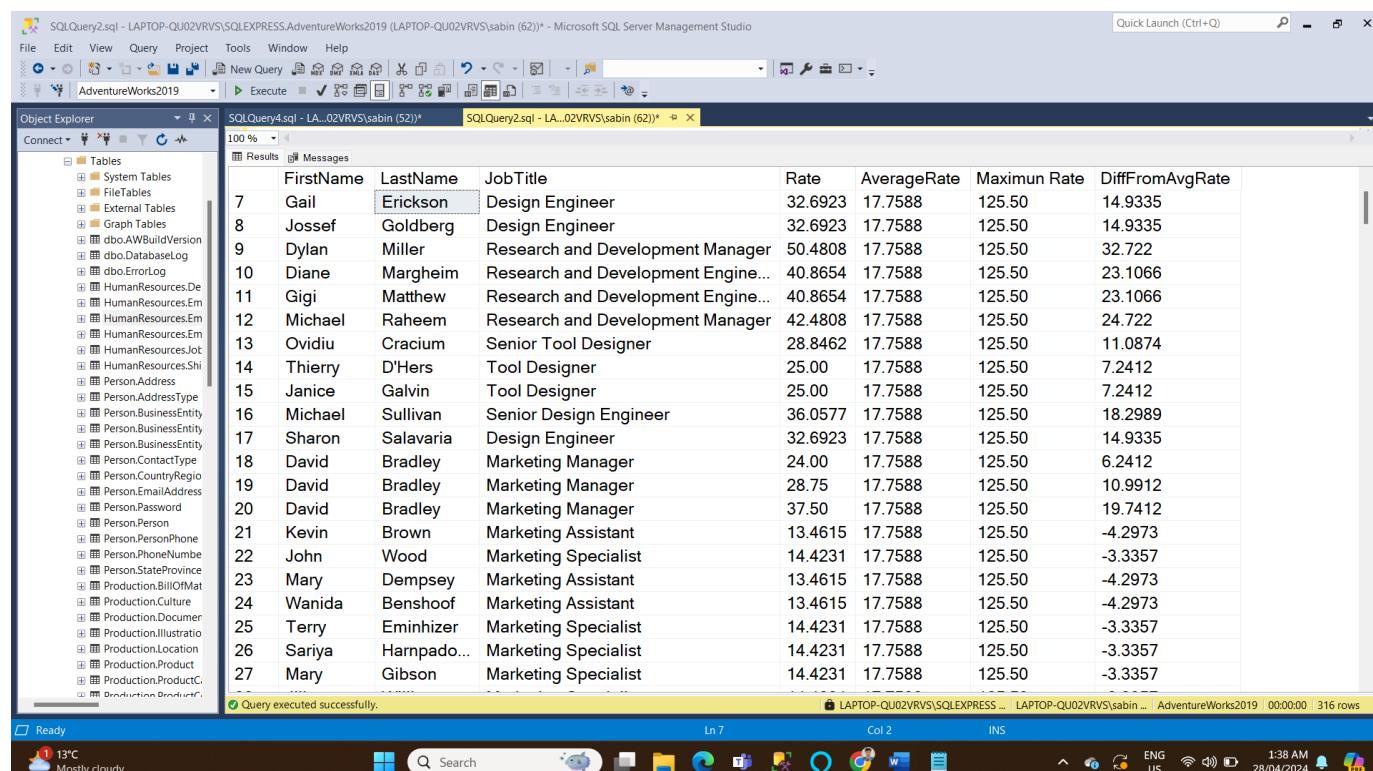
JOIN AdventureWorks2019.Person.Person PP
 ON HREPH.BusinessEntityID = PP.BusinessEntityID
 JOIN AdventureWorks2019.HumanResources.Employee HRE
 ON HREPH.BusinessEntityID = HRE.BusinessEntityID



```

SELECT
  PP.FirstName,
  PP.LastName,
  HRE.JobTitle,
  HREPH.Rate,
  AverageRate = AVG(HREPH.Rate) OVER(),
  [Maximum 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	Maximum 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
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	Rahemeh	Research and Development Manager	42.4808	17.7588	125.50	24.722
13	Ovidiu	Craciun	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	Salavarria	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



	FirstName	LastName	JobTitle	Rate	AverageRate	Maximum 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	Rahemeh	Research and Development Manager	42.4808	17.7588	125.50	24.722
13	Ovidiu	Craciun	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	Salavarria	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(),
    [Maximum 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
```

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (62)) - Microsoft SQL Server Management Studio

```

SELECT
PP.FirstName,
PP.LastName,
HRE.JobTitle,
HREPH.Rate,
AverageRate = AVG(HREPH.Rate) OVER(),
[Maximum 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	Maximum Rate	DiffFromAvgRate	PercentofMaxRate
Ken	Sánchez	Chief Executive Officer	125.50	17.7588	125.50	107.7412	100.00
Terri	Duffy	Vice President of Engineering	63.4615	17.7588	125.50	45.7027	50.56
Roberto	Tamburello	Engineering Manager	43.2692	17.7588	125.50	25.5104	34.47
Rob	Walters	Senior Tool Designer	8.62	17.7588	125.50	-9.1388	6.86
Rob	Walters	Senior Tool Designer	23.72	17.7588	125.50	5.9612	18.90
Rob	Walters	Senior Tool Designer	29.8462	17.7588	125.50	12.0874	23.78

Query executed successfully.

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (62)) - Microsoft SQL Server Management Studio

```

SELECT
FirstName, LastName, JobTitle, Rate, AverageRate, Maximum Rate, DiffFromAvgRate, PercentofMaxRate
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	Maximum Rate	DiffFromAvgRate	PercentofMaxRate	
Rob	Walters	Senior Tool Designer	29.8462	17.7588	125.50	12.0874	23.78	
Gail	Erickson	Design Engineer	32.6923	17.7588	125.50	14.9335	26.04	
Jossef	Goldberg	Design Engineer	32.6923	17.7588	125.50	14.9335	26.04	
Dylan	Miller	Research and Development...	50.4808	17.7588	125.50	32.722	40.22	
10	Diane	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	Salavarria	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	Bensoof	Marketing Assistant	13.4615	17.7588	125.50	-4.2973	10.72
25	Terry	Eminimizer	Marketing Specialist	14.4231	17.7588	125.50	-3.3357	11.49
26	Sariya	Hampado...	Marketing Specialist	14.4231	17.7588	125.50	-3.3357	11.49

Query executed successfully.

Exercise 5: Create a query with the following columns:

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

ListPrice” from the Production.Product table

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

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

Answer:

select

ProductName = PP.Name,

PP.ListPrice,

ProductSubcategory = PPS.Name,

ProductCategory = PPC.Name

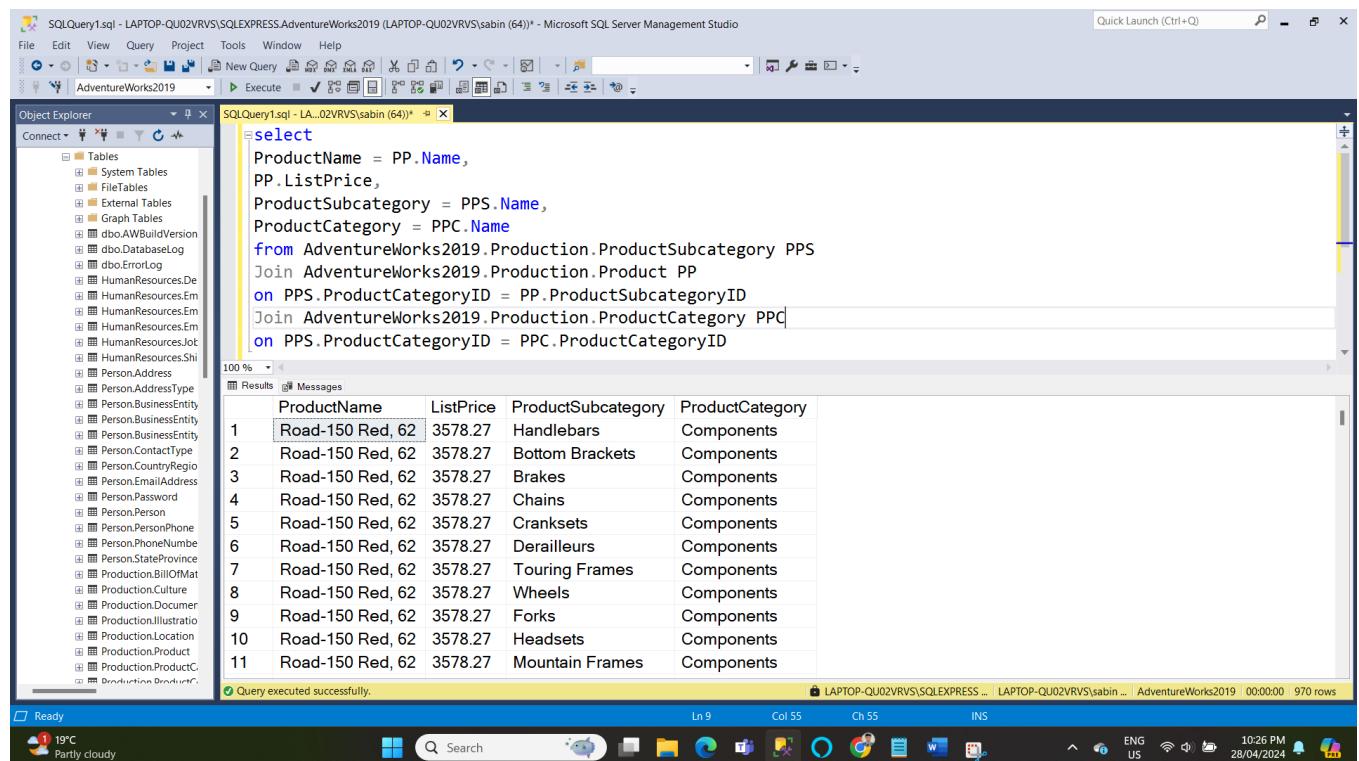
from AdventureWorks2019.Production.ProductSubcategory PPS

Join AdventureWorks2019.Production.Product PP

on PPS.ProductCategoryID = PP.ProductSubcategoryId

Join AdventureWorks2019.Production.ProductCategory PPC

on PPS.ProductCategoryID = PPC.ProductCategoryID



The screenshot shows the Microsoft SQL Server Management Studio interface. The query window contains the following T-SQL code:

```
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
```

The results grid displays 11 rows of data:

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

At the bottom of the screen, the taskbar shows the date and time as 28/04/2024 10:26 PM.

	ProductName	ListPrice	ProductSubcategory	ProductCategory
11	Road-150 Red, 62	3578.27	Mountain Frames	Components
12	Road-150 Red, 62	3578.27	Pedals	Components
13	Road-150 Red, 62	3578.27	Road Frames	Components
14	Road-150 Red, 62	3578.27	Saddles	Components
15	Road-150 Red, 44	3578.27	Handlebars	Components
16	Road-150 Red, 44	3578.27	Bottom Brackets	Components
17	Road-150 Red, 44	3578.27	Brakes	Components
18	Road-150 Red, 44	3578.27	Chains	Components
19	Road-150 Red, 44	3578.27	Cranksets	Components
20	Road-150 Red, 44	3578.27	Derailleurs	Components
21	Road-150 Red, 44	3578.27	Touring Frames	Components
22	Road-150 Red, 44	3578.27	Wheels	Components
23	Road-150 Red, 44	3578.27	Forks	Components
24	Road-150 Red, 44	3578.27	Headsets	Components
25	Road-150 Red, 44	3578.27	Mountain Frames	Components
26	Road-150 Red, 44	3578.27	Pedals	Components
27	Road-150 Red, 44	3578.27	Road Frames	Components
28	Road-150 Red, 44	3578.27	Saddles	Components
29	Road-150 Red, 48	3578.27	Handlebars	Components
30	Road-150 Red, 48	3578.27	Bottom Brackets	Components
31	Road-150 Red, 48	3578.27	Brakes	Components

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

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (64))* - Microsoft SQL Server Management Studio

```

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

```

100 %

Results Messages

	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

Query executed successfully.

LAPTOP-QU02VRVS\SQLEXPRESS ... LAPTOP-QU02VRVS\sabin ... AdventureWorks2019 | 00:00:00 | 970 rows

Ready 19°C Clear

LN 11 Col 49 Ch 49 INS

ENG US 10:42 PM 28/04/2024

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (64))* - Microsoft SQL Server Management Studio

```

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

```

100 %

Results Messages

	ProductName	ListPrice	ProductSubcategory	ProductCategory	AvgPriceByCategory
12	LL Mountain Handlebars	44.54	Tires and Tubes	Accessories	73.89
13	ML Mountain Handle...	61.92	Bike Racks	Accessories	73.89
14	ML Mountain Handle...	61.92	Bike Stands	Accessories	73.89
15	ML Mountain Handle...	61.92	Bottles and Cages	Accessories	73.89
16	ML Mountain Handle...	61.92	Cleaners	Accessories	73.89
17	ML Mountain Handle...	61.92	Fenders	Accessories	73.89
18	ML Mountain Handle...	61.92	Helmets	Accessories	73.89
19	ML Mountain Handle...	61.92	Hydration Packs	Accessories	73.89
20	ML Mountain Handle...	61.92	Lights	Accessories	73.89
21	ML Mountain Handle...	61.92	Locks	Accessories	73.89
22	ML Mountain Handle...	61.92	Panniers	Accessories	73.89
23	ML Mountain Handle...	61.92	Pumps	Accessories	73.89
24	ML Mountain Handle...	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

Query executed successfully.

LAPTOP-QU02VRVS\SQLEXPRESS ... LAPTOP-QU02VRVS\sabin ... AdventureWorks2019 | 00:00:00 | 970 rows

Ready 19°C Clear

LN 11 Col 49 Ch 49 INS

ENG US 10:43 PM 28/04/2024

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

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (63))* - Microsoft SQL Server Management Studio

```

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

```

Results

	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	Rib-Shorts	Clothing	1425.2481	1425.2481

Query executed successfully.

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (63))* - Microsoft SQL Server Management Studio

```

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

```

Results

	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

Query executed successfully.

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
```

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (63))* - Microsoft SQL Server Management Studio

```

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

```

100 %

Results Messages

	ProductName	ListPrice	ProductSubcategory	ProductCategory	AvgPriceByCategory	AvgPriceByCategoryAndSubcategory	ProductVsCategoryDelta
1	Touring-3000 Blue, 44	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481	-682.8981
2	Touring-3000 Blue, 50	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481	-682.8981
3	Touring-2000 Blue, 60	1214.85	Bib-Shorts	Clothing	1425.2481	1425.2481	-210.3981
4	Touring-1000 Yellow, 46	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481	958.8219
5	Touring-1000 Yellow, 50	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481	958.8219
6	Touring-1000 Yellow, 54	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481	958.8219
7	Touring-1000 Yellow, 60	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481	958.8219
8	Touring-3000 Blue, 54	742.35	Rib-Shorts	Clothing	1425.2481	1425.2481	-682.8981

Query executed successfully.

Ready 20°C Mostly cloudy Search Ch 55 INS ENG US 11:34 PM 29/04/2024

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (67))* - Microsoft SQL Server Management Studio

```

SELECT ProductName, ListPrice, ProductSubcategory, ProductCategory, AvgPriceByCategory, AvgPriceByCategoryAndSubcategory, ProductVsCategoryDelta
FROM AdventureWorks2019.Production.ProductSubcategory
JOIN AdventureWorks2019.Production.Product ON ProductSubcategoryID = ProductID
JOIN AdventureWorks2019.Production.ProductCategory ON ProductCategoryID = CategoryID

```

100 %

Results Messages

	ProductName	ListPrice	ProductSubcategory	ProductCategory	AvgPriceByCategory	AvgPriceByCategoryAndSubcategory	ProductVsCategoryDelta
7	Touring-1000 Yellow, 60	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481	958.8219
8	Touring-3000 Blue, 54	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481	-682.8981
9	Touring-3000 Blue, 58	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481	-682.8981
10	Touring-3000 Blue, 62	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481	-682.8981
11	Touring-3000 Yellow, 44	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481	-682.8981
12	Touring-3000 Yellow, 50	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481	-682.8981
13	Touring-3000 Yellow, 54	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481	-682.8981
14	Touring-3000 Yellow, 58	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481	-682.8981
15	Touring-3000 Yellow, 62	742.35	Bib-Shorts	Clothing	1425.2481	1425.2481	-682.8981
16	Touring-1000 Blue, 46	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481	958.8219
17	Touring-1000 Blue, 50	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481	958.8219
18	Touring-1000 Blue, 54	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481	958.8219
19	Touring-1000 Blue, 60	2384.07	Bib-Shorts	Clothing	1425.2481	1425.2481	958.8219
20	Touring-2000 Blue, 46	1214.85	Bib-Shorts	Clothing	1425.2481	1425.2481	-210.3981
21	Touring-2000 Blue, 50	1214.85	Bib-Shorts	Clothing	1425.2481	1425.2481	-210.3981
22	Touring-2000 Blue, 54	1214.85	Bib-Shorts	Clothing	1425.2481	1425.2481	-210.3981
23	LL Mountain Handlebars	44.54	Bike Racks	Accessories	73.89	73.89	-29.35
24	ML Mountain Handleb...	61.92	Bike Racks	Accessories	73.89	73.89	-11.97
25	HL Mountain Handlebars	120.27	Bike Racks	Accessories	73.89	73.89	46.38
26	LL Road Handlebars	44.54	Bike Racks	Accessories	73.89	73.89	-29.35
27	ML Road Handlebars	61.92	Bike Racks	Accessories	73.89	73.89	-11.97

Query executed successfully.

Ready 20°C Mostly cloudy Search Ch 49 INS ENG US 11:39 PM 29/04/2024

Exercise 9: Create a query with the following columns:

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

“ListPrice” from the Production.Product table.

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

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

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (67)) - Microsoft SQL Server Management Studio

```

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

```

100 %

Results Messages

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

Query executed successfully.

Ready

LAPTOP-QU02VRVS\SQLEXPRESS ... LAPTOP-QU02VRVS\sabin ... AdventureWorks2019 | 00:00:00 | 970 rows

LN 5 Col 27 Ch 27 INS

ENG US 11:45 PM 29/04/2024

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (67)) - Microsoft SQL Server Management Studio

```

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

```

100 %

Results Messages

	ProductName	ListPrice	ProductSubcategory	ProductCategory
9	Road-150 Red, 62	3578.27	Forks	Components
10	Road-150 Red, 62	3578.27	Headsets	Components
11	Road-150 Red, 62	3578.27	Mountain Frames	Components
12	Road-150 Red, 62	3578.27	Pedals	Components
13	Road-150 Red, 62	3578.27	Road Frames	Components
14	Road-150 Red, 62	3578.27	Saddles	Components
15	Road-150 Red, 44	3578.27	Handlebars	Components
16	Road-150 Red, 44	3578.27	Bottom Brackets	Components
17	Road-150 Red, 44	3578.27	Brakes	Components
18	Road-150 Red, 44	3578.27	Chains	Components
19	Road-150 Red, 44	3578.27	Cranksets	Components
20	Road-150 Red, 44	3578.27	Derailleurs	Components
21	Road-150 Red, 44	3578.27	Touring Frames	Components
22	Road-150 Red, 44	3578.27	Wheels	Components
23	Road-150 Red, 44	3578.27	Forks	Components
24	Road-150 Red, 44	3578.27	Headsets	Components
25	Road-150 Red, 44	3578.27	Mountain Frames	Components
26	Road-150 Red, 44	3578.27	Pedals	Components
27	Road-150 Red, 44	3578.27	Road Frames	Components
28	Road-150 Red, 44	3578.27	Saddles	Components
29	Road-150 Red, 48	3578.27	Handlebars	Components

Query executed successfully.

Ready

LAPTOP-QU02VRVS\SQLEXPRESS ... LAPTOP-QU02VRVS\sabin ... AdventureWorks2019 | 00:00:00 | 970 rows

LN 5 Col 27 Ch 27 INS

ENG US 11:45 PM 29/04/2024

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)

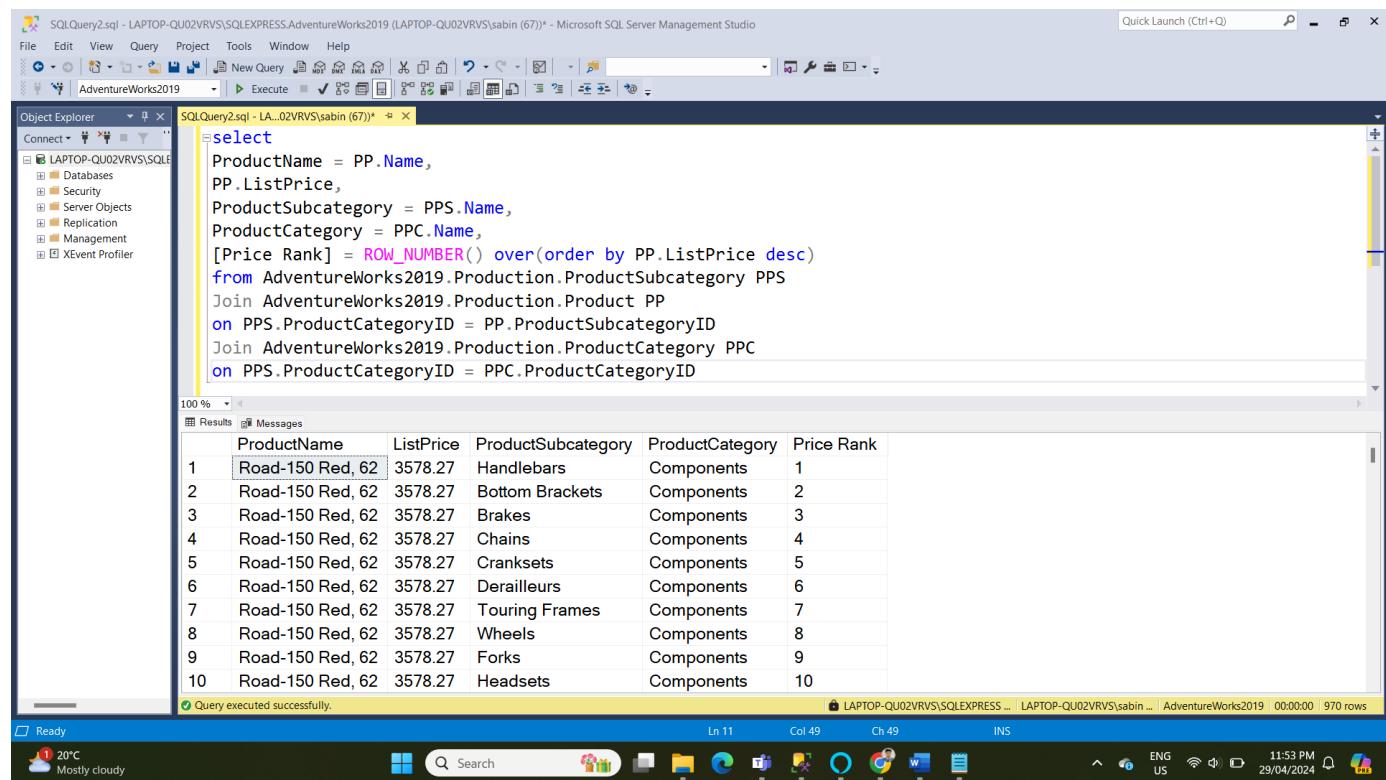
from AdventureWorks2019.Production.ProductSubcategory PPS

Join AdventureWorks2019.Production.Product PP

on PPS.ProductCategoryID = PP.ProductSubcategoryID

Join AdventureWorks2019.Production.ProductCategory PPC

on PPS.ProductCategoryID = PPC.ProductCategoryID



The screenshot shows a Microsoft SQL Server Management Studio window with the following details:

- Query Editor:** The main pane displays the T-SQL query provided above.
- Object Explorer:** On the left, it shows the database structure for AdventureWorks2019, including Databases, Security, Server Objects, Replication, Management, and XEvent Profiler.
- Status Bar:** At the bottom, it shows "Query executed successfully." and the system status bar with information like "LN 11 Col 49 Ch 49 INS", "LAPTOP-QU02VRVS\SQLEXPRESS ...", "AdventureWorks2019", "00:00:00", and "970 rows".
- Taskbar:** The bottom of the screen shows the Windows taskbar with icons for weather (20°C), search, file explorer, internet explorer, and other system tools.

The query results grid displays the following data:

	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

The screenshot shows a Microsoft SQL Server Management Studio window. The title bar reads "SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (67)) - Microsoft SQL Server Management Studio". The main area displays a results grid titled "SQLQuery2.sql - LA...02VRVS\sabin (67)*". The grid has columns: ProductName, ListPrice, ProductSubcategory, ProductCategory, and Price Rank. The data consists of 970 rows of products from the AdventureWorks2019 database. The bottom status bar shows "Query executed successfully." and other system information like the date and time.

	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,

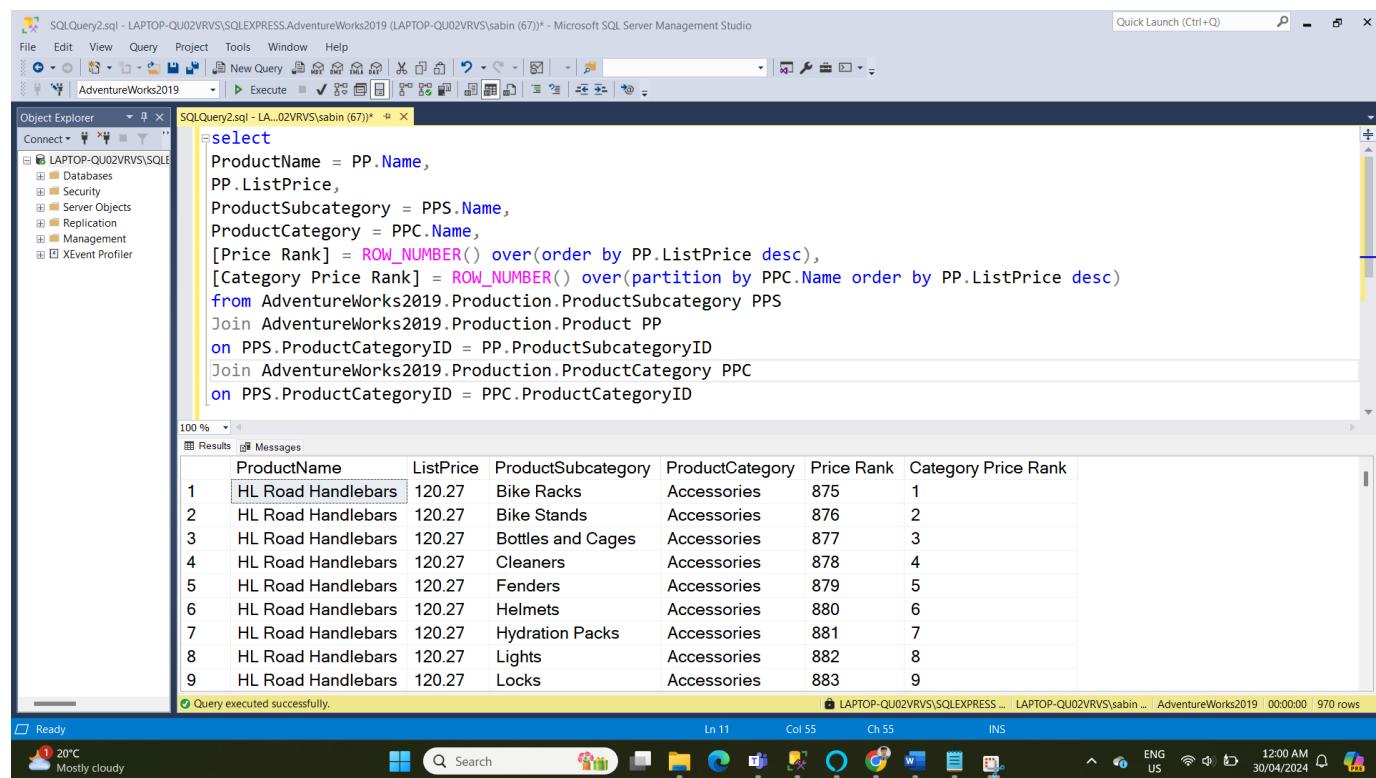
ProductCategory = PPC.Name,

[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

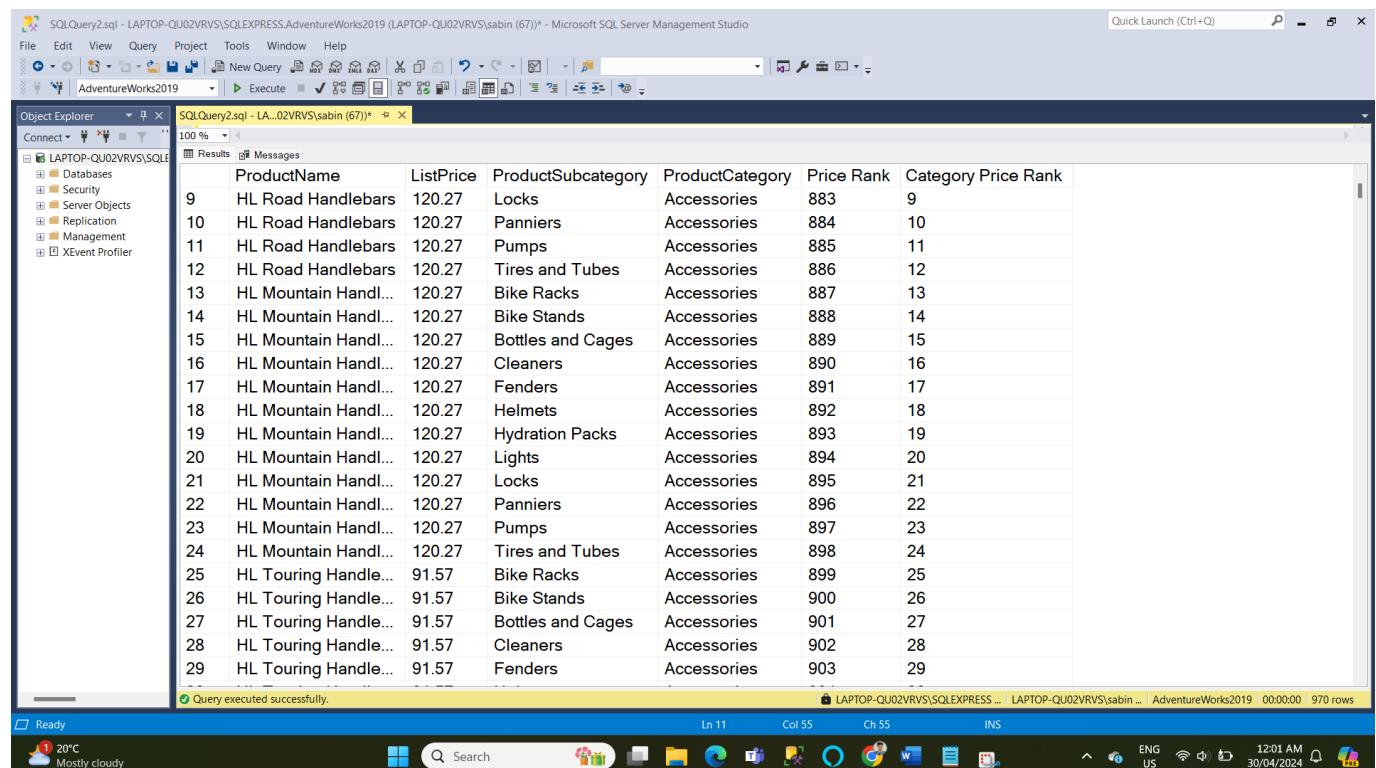


```

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (67)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
AdventureWorks2019 Execute
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

Query executed successfully.



	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

Query executed successfully.

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
```

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (67)) - Microsoft SQL Server Management Studio

```

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

```

Results Messages

Product Name	List Price	Product Subcategory	Product Category	Price Rank	Category Price Rank	Top 5 Price in category
HL Road Handlebars	120.27	Bike Racks	Accessories	875	1	Yes
HL Road Handlebars	120.27	Bike Stands	Accessories	876	2	Yes
HL Road Handlebars	120.27	Bottles and Cages	Accessories	877	3	Yes
HL Road Handlebars	120.27	Cleaners	Accessories	878	4	Yes

Query executed successfully.

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (67)) - Microsoft SQL Server Management Studio

```

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

```

Results Messages

Product Name	List Price	Product Subcategory	Product Category	Price Rank	Category Price Rank	Top 5 Price in category
HL Road Handlebars	120.27	Cleaners	Accessories	878	4	Yes
HL Road Handlebars	120.27	Fenders	Accessories	879	5	Yes
HL Road Handlebars	120.27	Helmets	Accessories	880	6	No
HL Road Handlebars	120.27	Hydration Packs	Accessories	881	7	No
HL Road Handlebars	120.27	Lights	Accessories	882	8	No
HL Road Handlebars	120.27	Locks	Accessories	883	9	No
HL Road Handlebars	120.27	Panniers	Accessories	884	10	No
HL Road Handlebars	120.27	Pumps	Accessories	885	11	No
HL Road Handlebars	120.27	Tires and Tubes	Accessories	886	12	No
HL Mountain Handl...	120.27	Bike Racks	Accessories	887	13	No
HL Mountain Handl...	120.27	Bike Stands	Accessories	888	14	No
HL Mountain Handl...	120.27	Bottles and Cages	Accessories	889	15	No
HL Mountain Handl...	120.27	Cleaners	Accessories	890	16	No
HL Mountain Handl...	120.27	Fenders	Accessories	891	17	No
HL Mountain Handl...	120.27	Helmets	Accessories	892	18	No
HL Mountain Handl...	120.27	Hydration Packs	Accessories	893	19	No
HL Mountain Handl...	120.27	Lights	Accessories	894	20	No
HL Mountain Handl...	120.27	Locks	Accessories	895	21	No
HL Mountain Handl...	120.27	Panniers	Accessories	896	22	No
HL Mountain Handl...	120.27	Pumps	Accessories	897	23	No
HL Mountain Handl...	120.27	Tires and Tubes	Accessories	898	24	No

Query executed successfully.

Exercise 13: Using your solution query to Exercise 11 from the ROW_NUMBER exercises as a starting 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
```

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.master (LAPTOP-QU02VRVS\sabin (73)) - Microsoft SQL Server Management Studio

```

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

```

100 %

Results Messages

	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

Query executed successfully.

LN 13 Col 49 Ch 49 INS

Ready 18°C Clear Search ENG US 11:49 PM 30/04/2024

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.master (LAPTOP-QU02VRVS\sabin (73)) - Microsoft SQL Server Management Studio

```

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

```

100 %

Results Messages

	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
31	HL Touring Handle...	91.57	Hydration Packs	Accessories	905	21	25

Query executed successfully.

LN 13 Col 49 Ch 49 INS

Ready 18°C Clear Search ENG US 11:49 PM 30/04/2024

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
```

```

SQLQuery2.sql - LAPTOP-QU02VRV\SQLEXPRESS.master (LAPTOP-QU02VRV\sa\bin (73)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query New Query Results Messages
master
Object Explorer
Connect Databases System Databases Database Snapshots AdventureWorks2019 Security Server Objects Replication Management XEvent Profiler
SQLQuery2.sql - LA..02VRV\sa\bin (73)*
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

```

100 %

Results Messages

	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
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

Query executed successfully.

100 %

Results Messages

	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

Query executed successfully.

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

```

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

```

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
HL Road Handlebars	120.27	Bike Racks	Accessories	875	1	1	1	Yes
HL Road Handlebars	120.27	Bike Stands	Accessories	876	2	1	1	Yes
HL Road Handlebars	120.27	Bottles and Cages	Accessories	877	3	1	1	Yes
HL Road Handlebars	120.27	Cleaners	Accessories	878	4	1	1	Yes
HL Road Handlebars	120.27	Fenders	Accessories	879	5	1	1	Yes
HL Road Handlebars	120.27	Helmets	Accessories	880	6	1	1	Yes

Query executed successfully.

SQLQuery2.sql - LAPTOP-QU02VRVS\SQLEXPRESS.master (LAPTOP-QU02VRVS\sabin (55)) - Microsoft SQL Server Management Studio

```

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

```

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
HL Road Handlebars	120.27	Bike Stands	Accessories	876	2	1	1	Yes
HL Road Handlebars	120.27	Bottles and Cages	Accessories	877	3	1	1	Yes
HL Road Handlebars	120.27	Cleaners	Accessories	878	4	1	1	Yes
HL Road Handlebars	120.27	Fenders	Accessories	879	5	1	1	Yes
HL Road Handlebars	120.27	Helmets	Accessories	880	6	1	1	Yes
HL Road Handlebars	120.27	Hydration Packs	Accessories	881	7	1	1	Yes
HL Road Handlebars	120.27	Lights	Accessories	882	8	1	1	Yes
HL Road Handlebars	120.27	Locks	Accessories	883	9	1	1	Yes
HL Road Handlebars	120.27	Panniers	Accessories	884	10	1	1	Yes
HL Road Handlebars	120.27	Pumps	Accessories	885	11	1	1	Yes
HL Road Handlebars	120.27	Tires and Tubes	Accessories	886	12	1	1	Yes
HL Mountain Handl...	120.27	Bike Racks	Accessories	887	13	1	1	Yes
HL Mountain Handl...	120.27	Bike Stands	Accessories	888	14	1	1	Yes
HL Mountain Handl...	120.27	Bottles and Cages	Accessories	889	15	1	1	Yes
HL Mountain Handl...	120.27	Cleaners	Accessories	890	16	1	1	Yes
HL Mountain Handl...	120.27	Fenders	Accessories	891	17	1	1	Yes
HL Mountain Handl...	120.27	Helmets	Accessories	892	18	1	1	Yes
HL Mountain Handl...	120.27	Hydration Packs	Accessories	893	19	1	1	Yes
HL Mountain Handl...	120.27	Lights	Accessories	894	20	1	1	Yes
HL Mountain Handl...	120.27	Locks	Accessories	895	21	1	1	Yes
HL Mountain Handl...	120.27	Panniers	Accessories	896	22	1	1	Yes
HL Mountain Handl...	120.27	Pumps	Accessories	897	23	1	1	Yes
HL Mountain Handl...	120.27	Tires and Tubes	Accessories	898	24	1	1	Yes
HL Touring Handl...	91.57	Bike Rocks	Accessories	899	25	25	2	Yes
HL Touring Handl...	91.57	Bike Stands	Accessories	900	26	25	2	Yes
HL Touring Handl...	91.57	Bottles and Cages	Accessories	901	27	25	2	Yes
HL Touring Handl...	91.57	Cleaners	Accessories	902	28	25	2	Yes
HL Touring Handl...	91.57	Fenders	Accessories	903	29	25	2	Yes
HL Touring Handl...	91.57	Helmets	Accessories	904	30	25	2	Yes
HL Touring Handl...	91.57	Hydration Packs	Accessories	905	31	25	2	Yes
HL Touring Handl...	91.57	Lights	Accessories	906	32	25	2	Yes
HL Touring Handl...	91.57	Locks	Accessories	907	33	25	2	Yes

Query executed successfully.

The screenshot shows a Microsoft SQL Server Management Studio window with the following details:

- File Bar:** File, Edit, View, Query, Project, Tools, Window, Help.
- Toolbar:** Standard toolbar with icons for New Query, Save, Print, etc.
- Object Explorer:** Shows the database structure under "LAPTOP-QU02VRVS\SQLEXPRESS\master".
- Results Grid:** The main area displays a table with 970 rows. The columns are:

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
Mountain-300 Blac...	1079.99	Touring Bikes	Bikes	519	45	43	5	Yes
Mountain-300 Blac...	1079.99	Mountain Bikes	Bikes	520	46	43	5	Yes
Mountain-300 Blac...	1079.99	Road Bikes	Bikes	521	47	43	5	Yes
Mountain-300 Blac...	1079.99	Touring Bikes	Bikes	522	48	43	5	Yes
Mountain-300 Blac...	1079.99	Mountain Bikes	Bikes	523	49	43	5	Yes
Mountain-300 Blac...	1079.99	Road Bikes	Bikes	524	50	43	5	Yes
Mountain-300 Blac...	1079.99	Touring Bikes	Bikes	525	51	43	5	Yes
Mountain-300 Blac...	1079.99	Mountain Bikes	Bikes	526	52	43	5	Yes
Mountain-300 Blac...	1079.99	Road Bikes	Bikes	527	53	43	5	Yes
Mountain-300 Blac...	1079.99	Touring Bikes	Bikes	528	54	43	5	Yes
Mountain-400-W Si...	769.49	Mountain Bikes	Bikes	697	55	55	6	NO
Mountain-400-W Si...	769.49	Road Bikes	Bikes	698	56	55	6	NO
Mountain-400-W Si...	769.49	Touring Bikes	Bikes	699	57	55	6	NO
Mountain-400-W Si...	769.49	Mountain Bikes	Bikes	700	58	55	6	NO
Mountain-400-W Si...	769.49	Road Bikes	Bikes	701	59	55	6	NO
Mountain-400-W Si...	769.49	Touring Bikes	Bikes	702	60	55	6	NO
Mountain-400-W Si...	769.49	Mountain Bikes	Bikes	703	61	55	6	NO
Mountain-400-W Si...	769.49	Road Bikes	Bikes	704	62	55	6	NO
Mountain-400-W Si...	769.49	Touring Bikes	Bikes	705	63	55	6	NO
Mountain-400-W Si...	769.49	Mountain Bikes	Bikes	706	64	55	6	NO
Mountain-400-W Si...	769.49	Road Bikes	Bikes	707	65	55	6	NO
Mountain-400-W Si...	769.49	Touring Bikes	Bikes	708	66	55	6	NO
Mountain-500 Silve...	564.99	Mountain Bikes	Bikes	789	67	67	7	NO
Mountain-500 Silve...	564.99	Road Bikes	Bikes	790	68	67	7	NO
Mountain-500 Silve...	564.99	Touring Bikes	Bikes	791	69	67	7	NO
Mountain-500 Silve...	564.99	Mountain Bikes	Bikes	792	70	67	7	NO
Mountain-500 Silve...	564.99	Road Bikes	Bikes	793	71	67	7	NO
Mountain-500 Silve...	564.99	Touring Bikes	Bikes	794	72	67	7	NO
Mountain-500 Silve...	564.99	Mountain Bikes	Bikes	795	73	67	7	NO
Mountain-500 Silve...	564.99	Road Bikes	Bikes	796	74	67	7	NO
Mountain-500 Silve...	564.99	Touring Bikes	Bikes	797	75	67	7	NO
Mountain-500 Silve...	564.99	Mountain Bikes	Bikes	798	76	67	7	NO
- Messages:** A green message bar at the bottom left says "Query executed successfully."
- System Bar:** Shows the date and time (1/05/2024, 12:17 AM), system status (Ready), and various system icons.

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
```

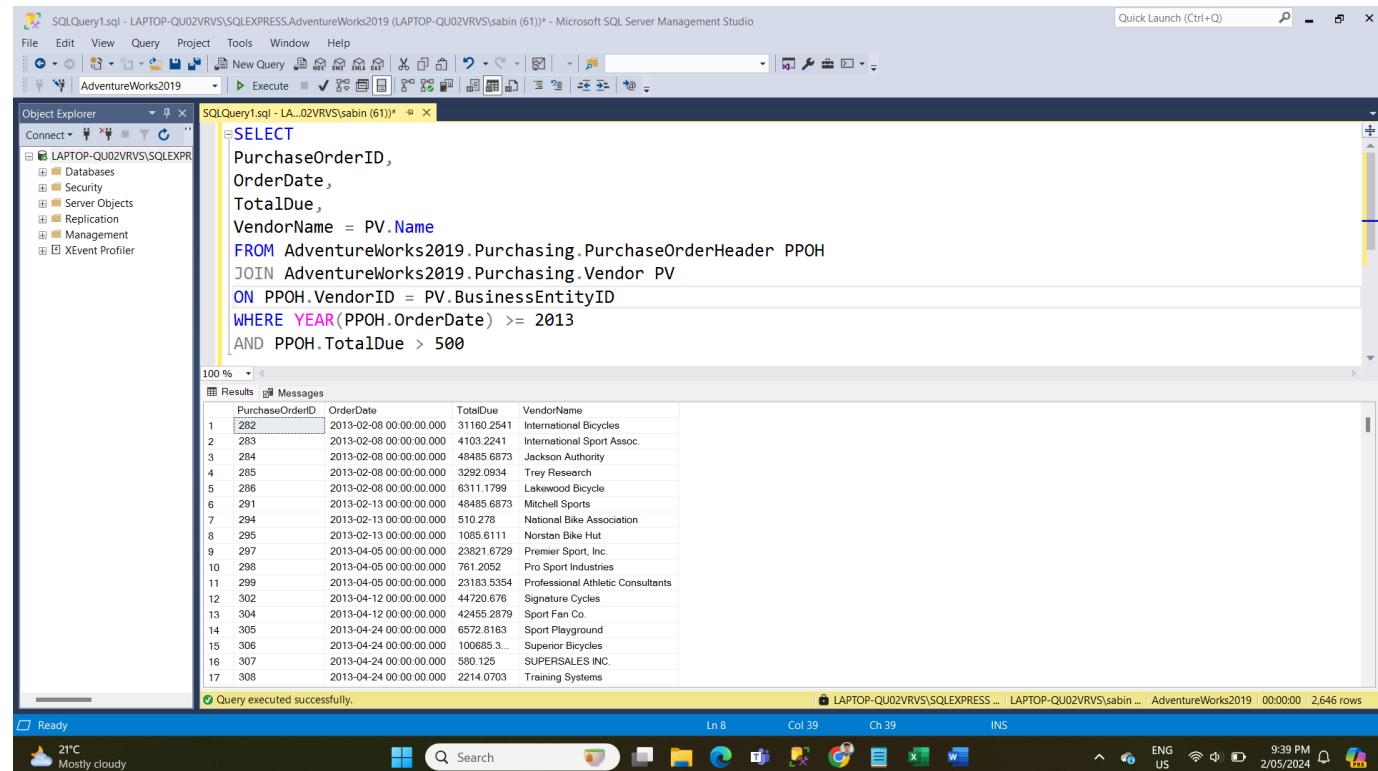
```
FROM AdventureWorks2019.Purchasing.PurchaseOrderHeader PPOH
```

```
JOIN AdventureWorks2019.Purchasing.Vendor PV
```

```
ON PPOH.VendorID = PV.BusinessEntityID
```

```
WHERE YEAR(PPOH.OrderDate) >= 2013
```

```
AND PPOH.TotalDue > 500
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the T-SQL code for selecting purchase order details from the AdventureWorks2019 database. The results pane shows a table with 2,646 rows, listing purchase orders with IDs 282 through 308, their dates, total due amounts, and vendor names. The vendor names include International Bicycles, International Sport Assoc., Jackson Authority, Trey Research, Lakewood Bicycle, Mitchell Sports, National Bike Association, Norstan Bike Hut, Premier Sport, Inc., Pro Sport Industries, Professional Athletic Consultants, Signature Cycles, Sport Fan Co., Sport Playground, Superior Bicycles, SUPERSALES INC., and Training Systems.

PurchaseOrderID	OrderDate	TotalDue	VendorName
282	2013-02-08 00:00:00.000	31160.2541	International Bicycles
283	2013-02-08 00:00:00.000	4103.2241	International Sport Assoc.
284	2013-02-08 00:00:00.000	48485.6873	Jackson Authority
285	2013-02-08 00:00:00.000	3292.0934	Trey Research
286	2013-02-08 00:00:00.000	6311.7799	Lakewood Bicycle
291	2013-02-13 00:00:00.000	48485.6873	Mitchell Sports
294	2013-02-13 00:00:00.000	510.278	National Bike Association
295	2013-02-13 00:00:00.000	1085.6111	Norstan Bike Hut
297	2013-04-05 00:00:00.000	23821.6729	Premier Sport, Inc.
298	2013-04-05 00:00:00.000	761.2052	Pro Sport Industries
299	2013-04-05 00:00:00.000	23183.5354	Professional Athletic Consultants
302	2013-04-12 00:00:00.000	44720.676	Signature Cycles
304	2013-04-12 00:00:00.000	42465.2879	Sport Fan Co.
305	2013-04-24 00:00:00.000	6572.8163	Sport Playground
306	2013-04-24 00:00:00.000	100685.3...	Superior Bicycles
307	2013-04-24 00:00:00.000	580.125	SUPERSALES INC.
308	2013-04-24 00:00:00.000	2214.0703	Training Systems

PurchaseOrderID	OrderDate	TotalDue	VendorName
9	2013-04-05 00:00:00.000	23821.6729	Premier Sport, Inc.
10	2013-04-05 00:00:00.000	761.2052	Pro Sport Industries
11	2013-04-05 00:00:00.000	23183.5354	Professional Athletic Consultants
12	2013-04-12 00:00:00.000	44720.676	Signature Cycles
13	2013-04-12 00:00:00.000	42455.2879	Sport Fan Co.
14	2013-04-24 00:00:00.000	6572.8163	Sport Playground
15	2013-04-24 00:00:00.000	100689.3...	Superior Bicycles
16	2013-04-24 00:00:00.000	580.125	SUPERSALES INC.
17	2013-04-24 00:00:00.000	2214.0703	Training Systems
18	2013-04-24 00:00:00.000	29233.0789	Trikes, Inc.
19	2013-04-24 00:00:00.000	1043.5289	Varsity Sport Co.
20	2013-04-24 00:00:00.000	7727.8451	Victory Bikes
21	2013-04-24 00:00:00.000	39915.5006	Vision Cycles, Inc.
22	2013-04-24 00:00:00.000	41817.1504	Vista Road Bikes
23	2013-04-24 00:00:00.000	26212.0569	West Junction Cycles
24	2013-04-24 00:00:00.000	9776.2665	Allenson Cycles
25	2013-04-25 00:00:00.000	22539.0165	American Bikes
26	2013-04-25 00:00:00.000	16164.0229	Anderson's Custom Bikes
27	2013-04-25 00:00:00.000	27995.0921	Proswear, Inc.
28	2013-04-25 00:00:00.000	1654.7486	Australia Bike Retailer
29	2013-04-25 00:00:00.000	1984.6192	Beaumont Bikes
30	2013-04-25 00:00:00.000	816.584	Bergeron Off-Roads
31	2013-04-25 00:00:00.000	38281.8886	Bicycle Specialists
32	2013-04-25 00:00:00.000	2032.6535	Bike Satellite, Inc.
33	2013-04-25 00:00:00.000	15104.7146	Capital Road Cycles
34	2013-04-25 00:00:00.000	7121.6145	Carlson Specialties
35	2013-04-25 00:00:00.000	168602.3...	Chicago City Saddles
36	2013-04-25 00:00:00.000	609.8274	Chicago Rent-All
37	2013-05-09 00:00:00.000	7721.4638	Circuit Cycles
38	2013-05-09 00:00:00.000	31019.8639	Comfort Road Bicycles
39	2013-05-09 00:00:00.000	41230.0639	Compete Enterprises, Inc
40	2013-05-09 00:00:00.000	8423.415	Compete, Inc.

Query executed successfully.

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,

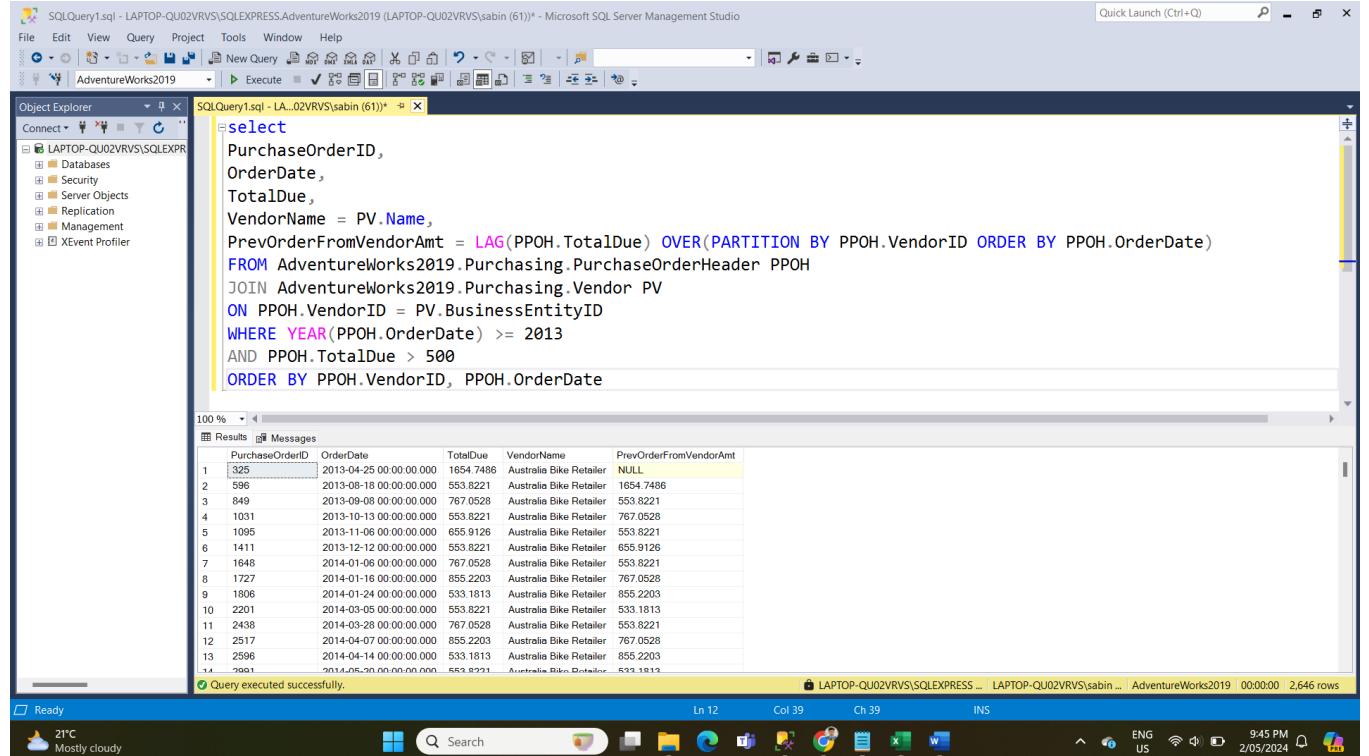
PrevOrderFromVendorAmt = LAG(PPOH.TotalDue) OVER(PARTITION BY PPOH.VendorID ORDER BY PPOH.OrderDate)

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



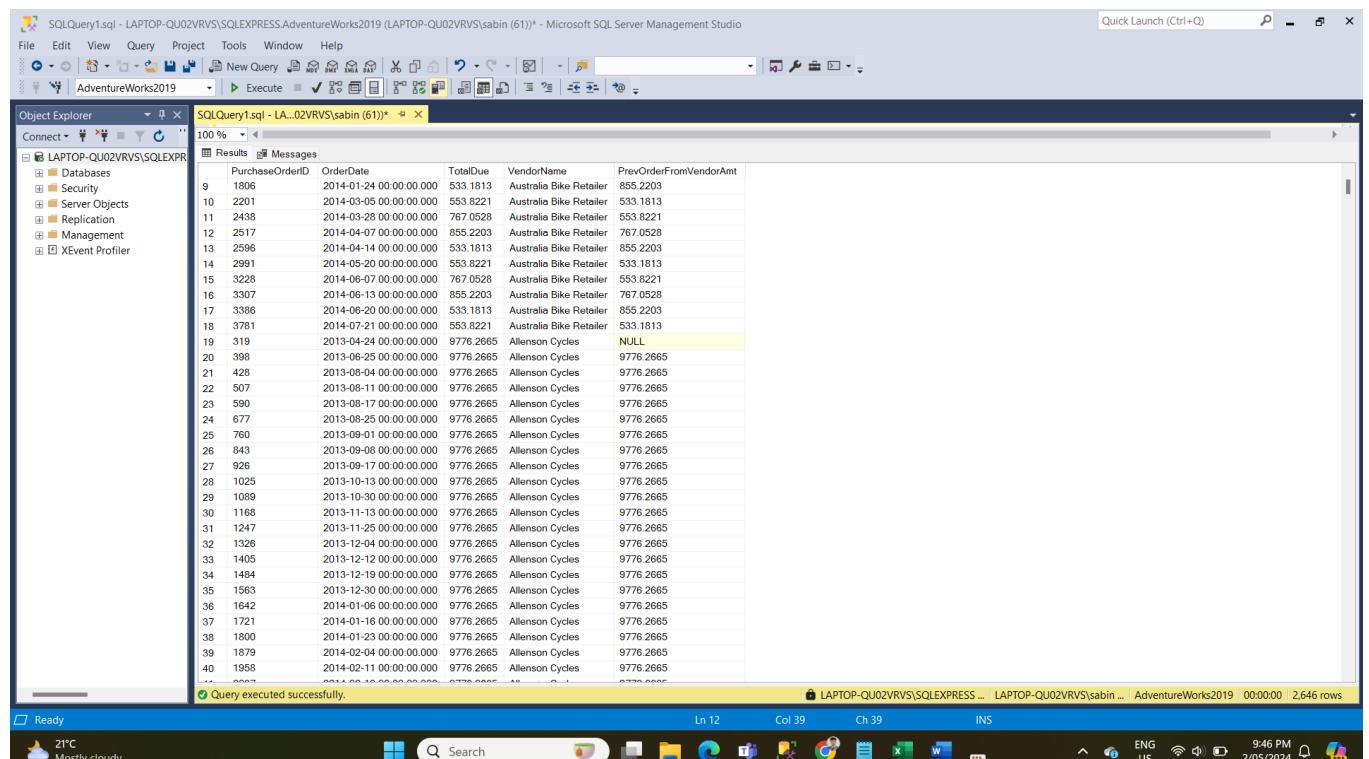
```

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
AdventureWorks2019 New Query Execute
Object Explorer
Connect
SQLQuery1.sql - LA_02VRVS\sabin (61)*
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
  
```

Results Messages

PurchaseOrderID	OrderDate	TotalDue	VendorName	PrevOrderFromVendorAmt
1 325	2013-04-25 00:00:00.000	1654.7486	Australia Bike Retailer	NULL
2 596	2013-08-18 00:00:00.000	553.8221	Australia Bike Retailer	1654.7486
3 849	2013-09-08 00:00:00.000	767.0528	Australia Bike Retailer	553.8221
4 1031	2013-10-13 00:00:00.000	553.8221	Australia Bike Retailer	767.0528
5 1095	2013-11-06 00:00:00.000	655.9126	Australia Bike Retailer	553.8221
6 1411	2013-12-12 00:00:00.000	553.8221	Australia Bike Retailer	655.9126
7 1648	2014-01-06 00:00:00.000	767.0528	Australia Bike Retailer	553.8221
8 1727	2014-01-16 00:00:00.000	855.2203	Australia Bike Retailer	767.0528
9 1806	2014-01-24 00:00:00.000	533.1813	Australia Bike Retailer	855.2203
10 2201	2014-03-05 00:00:00.000	553.8221	Australia Bike Retailer	533.1813
11 2438	2014-03-28 00:00:00.000	767.0528	Australia Bike Retailer	553.8221
12 2517	2014-04-07 00:00:00.000	855.2203	Australia Bike Retailer	767.0528
13 2596	2014-04-14 00:00:00.000	533.1813	Australia Bike Retailer	855.2203
14 2991	2014-05-20 00:00:00.000	553.8221	Australia Bike Retailer	533.1813

Query executed successfully.



```

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
AdventureWorks2019 New Query Execute
Object Explorer
Connect
SQLQuery1.sql - LA_02VRVS\sabin (61)*
select
PurchaseOrderID,
OrderDate,
TotalDue,
VendorName,
PrevOrderFromVendorAmt
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
  
```

Results Messages

PurchaseOrderID	OrderDate	TotalDue	VendorName	PrevOrderFromVendorAmt
9 1806	2014-01-24 00:00:00.000	533.1813	Australia Bike Retailer	855.2203
10 2201	2014-03-05 00:00:00.000	553.8221	Australia Bike Retailer	533.1813
11 2438	2014-03-28 00:00:00.000	767.0528	Australia Bike Retailer	553.8221
12 2517	2014-04-07 00:00:00.000	855.2203	Australia Bike Retailer	767.0528
13 2596	2014-04-14 00:00:00.000	533.1813	Australia Bike Retailer	855.2203
14 2991	2014-05-20 00:00:00.000	553.8221	Australia Bike Retailer	533.1813
15 3228	2014-06-07 00:00:00.000	767.0528	Australia Bike Retailer	553.8221
16 3307	2014-06-13 00:00:00.000	855.2203	Australia Bike Retailer	767.0528
17 3386	2014-06-20 00:00:00.000	533.1813	Australia Bike Retailer	855.2203
18 3781	2014-07-21 00:00:00.000	553.8221	Australia Bike Retailer	533.1813
19 319	2013-04-24 00:00:00.000	9776.2665	Allenson Cycles	NULL
20 398	2013-06-25 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
21 428	2013-08-04 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
22 507	2013-08-11 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
23 590	2013-08-17 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
24 677	2013-08-25 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
25 760	2013-09-01 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
26 843	2013-09-08 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
27 926	2013-09-17 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
28 1025	2013-10-13 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
29 1089	2013-10-30 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
30 1168	2013-11-13 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
31 1247	2013-11-25 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
32 1326	2013-12-04 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
33 1405	2013-12-12 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
34 1484	2013-12-19 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
35 1563	2013-12-30 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
36 1642	2014-01-06 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
37 1721	2014-01-16 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
38 1800	2014-01-23 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
39 1879	2014-02-04 00:00:00.000	9776.2665	Allenson Cycles	9776.2665
40 1958	2014-02-11 00:00:00.000	9776.2665	Allenson Cycles	9776.2665

Query executed successfully.

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
```

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 Object Explorer SQL Server Object Explorer Task List Results Messages

AdventureWorks2019 Execute

```

Object Explorer
Connect
LAPTOP-QU02VRVS\SQLEXPRESS
  Databases
  Security
  Server Objects
  Replication
  Management
  XEvent Profiler

SQLQuery1.sql - LA...02VRVS\sabin (61)*
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
  
```

Results Messages

PurchaseOrderID	OrderDate	TotalDue	VendorName	PrevOrderFromVendorAmt	NextOrderByEmployeeVendor
310	2013-04-24 00:00:00.000	1043.5289	Varsity Sport Co.	NULL	American Bikes
321	2013-04-25 00:00:00.000	22539.0165	American Bikes	NULL	Vista Road Bikes
392	2013-05-29 00:00:00.000	41817.1504	Vista Road Bikes	41817.1504	Victory Bikes
420	2013-08-04 00:00:00.000	3758.6299	Victory Bikes	53246.193	Bike Satellite Inc.
438	2013-08-05 00:00:00.000	2032.6535	Bike Satellite Inc.	2032.6535	Compete, Inc.
449	2013-08-06 00:00:00.000	8423.4115	Compete, Inc.	8423.4115	Trey Research
473	2013-08-07 00:00:00.000	10826.6133	Trey Research	7227.4233	Federal Sport
542	2013-08-13 00:00:00.000	1410.2839	Federal Sport	1410.2839	International Bicycles
553	2013-08-13 00:00:00.000	31160.2541	International Bicycles	31160.2541	Superior Bicycles
577	2013-08-15 00:00:00.000	100685.3348	Superior Bicycles	100685.3348	Gardner Touring Cycles
628	2013-08-19 00:00:00.000	502.6203	Gardner Touring Cycles	502.6203	Speed Corporation
631	2013-08-20 00:00:00.000	1020.6000	Speed Corporation	1020.6000	Superior Bicycles

Query executed successfully.

Ready Temps to drop Tomorrow Search L11 Col 35 Ch 35 INS

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 Object Explorer SQL Server Object Explorer Task List Results Messages

AdventureWorks2019 Execute

```

Object Explorer
Connect
LAPTOP-QU02VRVS\SQLEXPRESS
  Databases
  Security
  Server Objects
  Replication
  Management
  XEvent Profiler

SQLQuery1.sql - LA...02VRVS\sabin (61)*
select
    PurchaseOrderID,
    OrderDate,
    TotalDue,
    VendorName,
    PrevOrderFromVendorAmt,
    NextOrderByEmployeeVendor
  
```

Results Messages

PurchaseOrderID	OrderDate	TotalDue	VendorName	PrevOrderFromVendorAmt	NextOrderByEmployeeVendor
9	2013-08-13 00:00:00.000	31160.2541	International Bicycles	31160.2541	Superior Bicycles
10	2013-08-15 00:00:00.000	100685.3348	Superior Bicycles	100685.3348	Gardner Touring Cycles
11	2013-08-19 00:00:00.000	502.6203	Gardner Touring Cycles	502.6203	Speed Corporation
12	2013-08-22 00:00:00.000	616.0232	Speed Corporation	NULL	Bergerson Off-Roads
13	2013-08-25 00:00:00.000	2875.3392	Bergerson Off-Roads	697.6816	Vista Road Bikes
14	2013-08-29 00:00:00.000	41817.1504	Vista Road Bikes	41817.1504	Aurora Bike Center
15	2013-09-01 00:00:00.000	2370.704	Aurora Bike Center	1187.6319	Electronic Bike Repair & Su...
16	2013-09-03 00:00:00.000	53073.8959	Electronic Bike Repair...	53073.8959	Cruger Bike Company
17	2013-09-09 00:00:00.000	703.4248	Cruger Bike Company	525.28	Carlson Specialties
18	2013-09-09 00:00:00.000	18114.5192	Carlson Specialties	10992.9047	Circuit Cycles
19	2013-09-18 00:00:00.000	4211.7075	Circuit Cycles	7721.4638	Hill Bicycle Center
20	2013-09-22 00:00:00.000	7880.9981	Hill Bicycle Center	7880.9981	Victory Bikes
21	2013-10-06 00:00:00.000	57004.8229	Victory Bikes	57004.8229	Inline Accessories
22	2013-10-20 00:00:00.000	38281.8686	Inline Accessories	38281.8686	Carlson Specialties
23	2013-11-07 00:00:00.000	7121.6145	Carlson Specialties	3242.6667	First Rate Bicycles
24	2013-11-08 00:00:00.000	31319.7885	First Rate Bicycles	56200.7696	Jackson Authority
25	2013-11-11 00:00:00.000	70533.3379	Jackson Authority	70533.3379	Training Systems
26	2013-11-12 00:00:00.000	2214.0703	Training Systems	1902.578	Hill Bicycle Center
27	2013-11-19 00:00:00.000	7880.9981	Hill Bicycle Center	7880.9981	National Bike Association
28	2013-11-21 00:00:00.000	510.278	National Bike Associa...	510.278	Sport Playground
29	2013-11-22 00:00:00.000	6572.8163	Sport Playground	6572.8163	Bike Satellite Inc.
30	2013-11-26 00:00:00.000	2032.6535	Bike Satellite Inc.	2032.6535	Premier Sport, Inc.
31	2013-11-29 00:00:00.000	23821.6729	Premier Sport, Inc.	23821.6729	Beaumont Bikes
32	2013-12-04 00:00:00.000	1685.3792	Beaumont Bikes	1984.6192	Federal Sport
33	2013-12-06 00:00:00.000	1410.2839	Federal Sport	1410.2839	Chicago Rent-All
34	2013-12-13 00:00:00.000	609.8274	Chicago Rent-All	609.8274	Compete Enterprises, Inc
35	2013-12-23 00:00:00.000	39111.4474	Compete Enterprises...	41230.0639	Hybrid Bicycle Center
36	2013-12-24 00:00:00.000	24880.9811	Hybrid Bicycle Center	24880.9811	Vista Road Bikes
37	2013-12-30 00:00:00.000	41817.1504	Vista Road Bikes	41817.1504	Professional Athletic Consu...
38	2014-01-03 00:00:00.000	45358.8135	Professional Athletic ...	49915.1153	Varsity Sport Co.
39	2014-01-06 00:00:00.000	1043.5289	Varsity Sport Co.	1043.5289	Carlson Specialties
40	2014-01-08 00:00:00.000	18114.5192	Carlson Specialties	18114.5192	Signature Cycles

Query executed successfully.

Ready Temps to drop Tomorrow Search L13 Col 35 Ch 35 INS

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
```

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61)) - Microsoft SQL Server Management Studio

```

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.PurchaseOrderHeader PV
ON PPOH.VendorID = PV.BusinessEntityID
WHERE YEAR(PPOH.OrderDate) >= 2013
AND PPOH.TotalDue > 500
ORDER BY PPOH.EmployeeID, PPOH.OrderDate

```

Results

PurchaseOrderID	OrderDate	TotalDue	VendorName	PrevOrderFromVendorAmt	NextOrderByEmployeeVendor	Next2OrderByEmployeeVendor
310	2013-04-24 00:00:00.000	1043.5289	Varsity Sport Co.	NULL	American Bikes	Vista Road Bikes
321	2013-04-25 00:00:00.000	22559.0165	American Bikes	NULL	Vista Road Bikes	Victory Bikes
392	2013-05-29 00:00:00.000	41817.1504	Vista Road Bikes	41817.1504	Victory Bikes	Bike Satellite Inc.
420	2013-08-04 00:00:00.000	3758.6299	Victory Bikes	53246.193	Compete, Inc.	Compete, Inc.
438	2013-08-05 00:00:00.000	2032.6535	Bike Satellite Inc.	2032.6535	Trey Research	Trey Research
449	2013-08-06 00:00:00.000	8423.415	Compete, Inc.	8423.415	Federal Sport	Federal Sport
473	2013-08-07 00:00:00.000	10828.133	Trey Research	7227.4293	International Bicycles	International Bicycles
542	2013-08-13 00:00:00.000	1410.2839	Federal Sport	1410.2839	Superior Bicycles	Superior Bicycles
553	2013-08-13 00:00:00.000	31160.2541	International Bi...	31160.2541	Gardner Touring Cycles	Gardner Touring Cycles
577	2013-08-15 00:00:00.000	100685.3...	Superior Bicycl...	100685.3348	Speed Corporation	Speed Corporation
628	2013-08-19 00:00:00.000	502.6203	Gardner Touri...	502.6203	Bergeron Off-Roads	Bergeron Off-Roads

Query executed successfully.

LN 13 Col 24 Ch 24 INS

Ready 5 / Mounts Bay Rd Construction Search ENG US 9:55 PM 2/05/2024

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61)) - Microsoft SQL Server Management Studio

```

select
PurchaseOrderID OrderDate TotalDue VendorName PrevOrderFromVendorAmt NextOrderByEmployeeVendor Next2OrderByEmployeeVendor
9 553 2013-08-13 00:00:00.000 31160.2541 International Bi... 31160.2541 Superior Bicycles Gardner Touring Cycles
10 577 2013-08-15 00:00:00.000 100685.3... Superior Bicycl... 100685.3348 Gardner Touring Cycles Speed Corporation
11 628 2013-08-19 00:00:00.000 502.6203 Gardner Touri... 502.6203 Bergeron Off-Roads Vista Road Bikes
12 661 2013-08-22 00:00:00.000 616.0232 Speed Corpora... NULL Bergeron Off-Roads Vista Road Bikes
13 685 2013-08-25 00:00:00.000 2675.3392 Bergeron Off-... 697.6816 Vista Road Bikes Aurora Bike Center
14 750 2013-08-29 00:00:00.000 41817.1504 Vista Road Bikes 41817.1504 Aurora Bike Center
15 765 2013-09-01 00:00:00.000 2370.704 Aurora Bike Ce... 1187.6319 Electronic Bike Repair & Su...
16 789 2013-09-03 00:00:00.000 53073.8959 Electronic Bike... 53073.8959 Berger Bike Company Carlson Specialties
17 869 2013-09-09 00:00:00.000 703.4248 Berger Bike Co... 525.28 Carlson Specialties
18 858 2013-09-09 00:00:00.000 18114.5192 Carlson Specia... 10992.9047 Circuit Cycles
19 944 2013-09-18 00:00:00.000 4211.7075 Circuit Cycles 7721.4638 Hill Bicycle Center
20 962 2013-09-22 00:00:00.000 7880.9981 Hill Bicycle Cen... 7880.9981 Victory Bikes
21 1005 2013-10-06 00:00:00.000 57004.8229 Victory Bikes 57004.8229 Inline Accessories
22 1072 2013-10-20 00:00:00.000 38281.8686 Inline Accessori... 38281.8686 Carlson Specialties
23 1104 2013-11-07 00:00:00.000 7121.6145 Carlson Specia... 3242.6667 First Rate Bicycles
24 1122 2013-11-08 00:00:00.000 31319.7885 First Rate Bicy... 56200.7696 Jackson Authority
25 1133 2013-11-11 00:00:00.000 70533.3379 Jackson Autho... 70533.3379 Training Systems
26 1157 2013-11-12 00:00:00.000 2214.0703 Training Syste... 1902.578 Hill Bicycle Center
27 1204 2013-11-19 00:00:00.000 7880.9981 Hill Bicycle Cen... 7880.9981 National Bike Association
28 1222 2013-11-21 00:00:00.000 510.278 National Bike A... 510.278 Sport Playground
29 1233 2013-11-22 00:00:00.000 6572.8163 Sport Playgroun... 6572.8163 Bike Satellite Inc.
30 1257 2013-11-26 00:00:00.000 2032.6535 Bike Satellite Inc. 2032.6535 Premier Sport, Inc.
31 1304 2013-11-29 00:00:00.000 23821.6729 Premier Sport, ...
32 1333 2013-12-04 00:00:00.000 1685.3792 Beaumont Bikes 1984.6192 Federal Sport
33 1357 2013-12-06 00:00:00.000 1410.2839 Federal Sport 1410.2839 Chicago Rent-All
34 1422 2013-12-13 00:00:00.000 609.8274 Chicago Rent-All 609.8274 Compete Enterprises, Inc.
35 1504 2013-12-23 00:00:00.000 39111.4474 Compete Enter... 41230.0639 Hybrid Bicycle Center
36 1522 2013-12-24 00:00:00.000 24880.9811 Hybrid Bicycle ... 24880.9811 Vista Road Bikes
37 1557 2013-12-30 00:00:00.000 41817.1504 Vista Road Bikes 41817.1504 Professional Athletic Consult...
38 1622 2014-01-03 00:00:00.000 45358.8135 Professional At... 49915.1153 Varsity Sport Co.
39 1633 2014-01-06 00:00:00.000 1043.5289 Varsity Sport Co. 1043.5289 Carlson Specialties
40 1657 2014-01-08 00:00:00.000 18114.5192 Carlson Specia... 18114.5192 Signature Cycles

```

Query executed successfully.

LN 13 Col 24 Ch 24 INS

Ready 5 / Mounts Bay Rd Construction Search ENG US 9:56 PM 2/05/2024

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
1	Accountant	2009-02-18	58	58
2	Accountant	2009-03-08	59	58
3	Accounts Manager	2009-01-30	57	57
4	Accounts Payable Specialist	2009-02-11	63	63
5	Accounts Payable Specialist	2009-03-01	64	63
6	Accounts Receivable Specialist	2008-12-18	60	60
7	Accounts Receivable Specialist	2009-01-06	61	60
8	Accounts Receivable Specialist	2009-01-24	62	60
9	Application Specialist	2008-12-23	71	71
10	Application Specialist	2009-01-11	72	71
11	Application Specialist	2009-02-03	73	71
12	Application Specialist	2009-02-16	74	71
13	Assistant to the Chief Financial Officer	2009-01-12	56	56
14	Benefits Specialist	2008-12-25	51	51
15	Buyer	2009-02-10	59	59
16	Buyer	2009-02-28	60	59
17	Buyer	2009-12-17	56	59
18	Buyer	2010-01-04	57	59
19	Buyer	2010-01-11	52	59

The screenshot shows the Microsoft SQL Server Management Studio interface. A query named 'SQLQuery1.sql' is running against the 'AdventureWorks2019' database. The results pane displays a table with columns: EmployeeID, JobTitle, HireDate, VacationHours, and FirstHireVacationHours. The data consists of 290 rows of employee information. The status bar at the bottom indicates the query was executed successfully at 00:00:00 on 2/05/2024.

EmployeeID	JobTitle	HireDate	VacationHours	FirstHireVacationHours
18	Buyer	2010-01-04	57	59
19	Buyer	2010-01-11	52	59
20	Buyer	2010-01-23	58	59
21	Buyer	2010-01-27	54	59
22	Buyer	2010-01-31	53	59
23	Buyer	2010-03-09	55	59
24	Chief Executive Officer	2009-01-14	99	99
25	Chief Financial Officer	2009-01-31	0	0
26	Control Specialist	2008-12-16	76	76
27	Control Specialist	2009-03-06	75	76
28	Database Administrator	2009-01-17	67	67
29	Database Administrator	2009-01-22	66	67
30	Design Engineer	2008-01-06	5	5
31	Design Engineer	2008-01-24	6	5
32	Design Engineer	2011-01-18	4	5
33	Document Control Assistant	2009-01-22	78	78
34	Document Control Assistant	2009-02-09	79	78
35	Document Control Manager	2009-01-04	77	77
36	Engineering Manager	2007-11-11	2	2
37	European Sales Manager	2012-04-16	21	21
38	Facilities Administrative Assistant	2009-12-21	87	87
39	Facilities Manager	2009-12-02	86	86
40	Finance Manager	2008-12-25	55	55
41	Human Resources Administrative A...	2009-02-06	52	52
42	Human Resources Administrative A...	2009-02-25	53	52
43	Human Resources Manager	2008-12-06	54	54
44	Information Services Manager	2008-12-11	65	65
45	Janitor	2010-01-27	89	89
46	Janitor	2010-02-16	90	89
47	Janitor	2010-03-05	88	89
48	Janitor	2010-03-07	91	89
49	Maintenance Supervisor	2008-12-14	92	92

Exercise 2: Create a query with the following columns:

- “ProductID” from the Production.Product table
- “Name” from the Production.Product table (alias this as “ProductName”)
- “ListPrice” from the Production.ProductListPriceHistory table
- “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
```

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61)) - Microsoft SQL Server Management Studio

```

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

```

100 %

Results Messages

ProductID	ProductName	ListPrice	ModifiedDate	HighestPrice	LowestPrice	PriceRange
1	Sport-100 Helmet, Red	33.6442	2012-05-29 00:00:00.000	34.99	33.6442	1.3458
2	Sport-100 Helmet, Red	34.99	2013-05-09 00:00:00.000	34.99	33.6442	1.3458
3	Sport-100 Helmet, Red	33.6442	2013-05-29 00:00:00.000	34.99	33.6442	1.3458
4	Sport-100 Helmet, Black	33.6442	2012-05-29 00:00:00.000	34.99	33.6442	1.3458
5	Sport-100 Helmet, Black	34.99	2013-05-09 00:00:00.000	34.99	33.6442	1.3458
6	Sport-100 Helmet, Black	33.6442	2013-05-29 00:00:00.000	34.99	33.6442	1.3458
7	Mountain Bike Socks, M	9.50	2012-05-29 00:00:00.000	9.50	9.50	0.00
8	Mountain Bike Socks, L	9.50	2012-05-29 00:00:00.000	9.50	9.50	0.00
9	Sport-100 Helmet, Blue	33.6442	2012-05-29 00:00:00.000	34.99	33.6442	1.3458
10	Sport-100 Helmet, Blue	34.99	2013-05-09 00:00:00.000	34.99	33.6442	1.3458
11	Sport-100 Helmet, Blue	33.6442	2013-05-29 00:00:00.000	34.99	33.6442	1.3458
12	AWC Logo Cap	8.6442	2012-05-29 00:00:00.000	8.99	8.6442	0.3458
13	AWC Logo Cap	8.99	2013-05-09 00:00:00.000	8.99	8.6442	0.3458

Query executed successfully.

LN 12 Col 34 Ch 34 INS

Ready 20°C Mostly cloudy Search

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61)) - Microsoft SQL Server Management Studio

```

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

```

100 %

Results Messages

ProductID	ProductName	ListPrice	ModifiedDate	HighestPrice	LowestPrice	PriceRange
11	Sport-100 Helmet, Blu...	33.6442	2013-05-29 00:00:00.000	34.99	33.6442	1.3458
12	AWC Logo Cap	8.6442	2012-05-29 00:00:00.000	8.99	8.6442	0.3458
13	AWC Logo Cap	8.99	2013-05-09 00:00:00.000	8.99	8.6442	0.3458
14	AWC Logo Cap	8.6442	2013-05-29 00:00:00.000	8.99	8.6442	0.3458
15	Long-Sleeve Logo Jer...	48.0673	2012-05-29 00:00:00.000	49.99	48.0673	1.9227
16	Long-Sleeve Logo Jer...	49.99	2013-05-09 00:00:00.000	49.99	48.0673	1.9227
17	Long-Sleeve Logo Jer...	48.0673	2013-05-29 00:00:00.000	49.99	48.0673	1.9227
18	Long-Sleeve Logo Jer...	48.0673	2012-05-29 00:00:00.000	49.99	48.0673	1.9227
19	Long-Sleeve Logo Jer...	49.99	2013-05-09 00:00:00.000	49.99	48.0673	1.9227
20	Long-Sleeve Logo Jer...	48.0673	2013-05-29 00:00:00.000	49.99	48.0673	1.9227
21	Long-Sleeve Logo Jer...	48.0673	2012-05-29 00:00:00.000	49.99	48.0673	1.9227
22	Long-Sleeve Logo Jer...	49.99	2013-05-09 00:00:00.000	49.99	48.0673	1.9227
23	Long-Sleeve Logo Jer...	48.0673	2013-05-29 00:00:00.000	49.99	48.0673	1.9227
24	Long-Sleeve Logo Jer...	48.0673	2012-05-29 00:00:00.000	49.99	48.0673	1.9227
25	Long-Sleeve Logo Jer...	49.99	2013-05-09 00:00:00.000	49.99	48.0673	1.9227
26	Long-Sleeve Logo Jer...	48.0673	2013-05-29 00:00:00.000	49.99	48.0673	1.9227
27	HL Road Frame - Red...	1263...	2012-05-29 00:00:00.000	1431.50	1263.4598	168.0402
28	HL Road Frame - Red...	1431.50	2013-05-09 00:00:00.000	1431.50	1263.4598	168.0402
29	HL Road Frame - Red...	1301...	2013-05-29 00:00:00.000	1431.50	1263.4598	168.0402
30	HL Road Frame - Red...	1263...	2012-05-29 00:00:00.000	1431.50	1263.4598	168.0402
31	HL Road Frame - Red...	1431.50	2013-05-09 00:00:00.000	1431.50	1263.4598	168.0402
32	HL Road Frame - Red...	1301...	2013-05-29 00:00:00.000	1431.50	1263.4598	168.0402
33	HL Road Frame - Red...	1263...	2012-05-29 00:00:00.000	1431.50	1263.4598	168.0402
34	HL Road Frame - Red...	1431.50	2013-05-09 00:00:00.000	1431.50	1263.4598	168.0402
35	HL Road Frame - Red...	1301...	2013-05-29 00:00:00.000	1431.50	1263.4598	168.0402
36	HL Road Frame - Red...	1263...	2012-05-29 00:00:00.000	1431.50	1263.4598	168.0402
37	HL Road Frame - Red...	1431.50	2013-05-09 00:00:00.000	1431.50	1263.4598	168.0402
38	HL Road Frame - Red...	1301...	2013-05-29 00:00:00.000	1431.50	1263.4598	168.0402
39	HL Road Frame - Red...	1263...	2012-05-29 00:00:00.000	1431.50	1263.4598	168.0402
40	HL Road Frame - Red...	1431.50	2013-05-09 00:00:00.000	1431.50	1263.4598	168.0402
41	HL Road Frame - Red...	1301...	2013-05-29 00:00:00.000	1431.50	1263.4598	168.0402
42	LL Road Frame - Blac...	297.6...	2012-05-29 00:00:00.000	337.22	297.6346	39.5854

Query executed successfully.

LN 12 Col 34 Ch 34 INS

Ready 20°C Mostly cloudy Search

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 Project Tools Window Help

New Query Object Explorer AdventureWorks2019 Execute

```

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
2	1727	1492	2014-01-16 00:00:00.000	61.9164	19.3489
3	2517	1492	2014-04-07 00:00:00.000	61.9164	19.3489
4	1879	1494	2014-02-04 00:00:00.000	707.784	221.1825

Query executed successfully.

Ln 18 Col 7 Ch 7 INS

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 Project Tools Window Help

New Query Object Explorer AdventureWorks2019 Execute

```

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
2	1727	1492	2014-01-16 00:00:00.000	61.9164	19.3489
3	2517	1492	2014-04-07 00:00:00.000	61.9164	19.3489
4	1879	1494	2014-02-04 00:00:00.000	707.784	221.1825
5	1958	1494	2014-02-11 00:00:00.000	707.784	221.1825
6	1800	1494	2014-01-23 00:00:00.000	707.784	221.1825
7	925	1496	2013-09-17 00:00:00.000	165.5413	51.7317
8	1325	1496	2013-12-04 00:00:00.000	165.5413	51.7317
9	397	1496	2013-06-25 00:00:00.000	67.1832	20.9948
10	2185	1498	2014-03-04 00:00:00.000	2116.422	661.3819
11	2106	1498	2014-02-26 00:00:00.000	2116.422	661.3819
12	2027	1498	2014-02-19 00:00:00.000	2116.422	661.3819
13	3590	1500	2014-07-07 00:00:00.000	33.516	10.4738
14	3511	1500	2014-06-30 00:00:00.000	33.516	10.4738
15	3432	1500	2014-06-24 00:00:00.000	33.516	10.4738
16	2133	1504	2014-02-27 00:00:00.000	44.1504	13.797
17	2212	1504	2014-03-05 00:00:00.000	44.1504	13.797
18	2054	1504	2014-02-20 00:00:00.000	44.1504	13.797
19	3573	1506	2014-07-04 00:00:00.000	3510.276	1096.9...
20	3652	1506	2014-07-11 00:00:00.000	3510.276	1096.9...
21	3494	1506	2014-06-28 00:00:00.000	3510.276	1096.9...
22	2136	1508	2014-02-27 00:00:00.000	2984.982	932.8069
23	2215	1508	2014-03-05 00:00:00.000	2984.982	932.8069
24	2057	1508	2014-02-21 00:00:00.000	2984.982	932.8069
25	3974	1510	2014-08-02 00:00:00.000	11.4433	3.576
26	3895	1510	2014-07-28 00:00:00.000	11.4433	3.576
27	3816	1510	2014-07-23 00:00:00.000	11.4433	3.576
28	1236	1514	2013-11-22 00:00:00.000	160.2947	50.0921
29	1157	1514	2013-11-12 00:00:00.000	160.2947	50.0921
30	1315	1514	2013-12-02 00:00:00.000	160.2947	50.0921
31	3730	1516	2014-07-17 00:00:00.000	36.3888	11.3715
32	3888	1516	2014-07-27 00:00:00.000	36.3888	11.3715
33	3809	1516	2014-07-23 00:00:00.000	36.3888	11.3715

Query executed successfully.

Ln 1 Col 1 INS

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
```

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61)) - Microsoft SQL Server Management Studio

```

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)
) PPOH

```

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 3307	1492	2014-06-13 00:00:00.000	61.9164	19.3489	855.2203
5 167	1492	2012-05-30 00:00:00.000	56.8764	17.7739	785.6053
6 1000	1494	2014-02-28 00:00:00.000	707.784	221.1825	9776.2665
7 1958	1494	2014-02-11 00:00:00.000	707.784	221.1825	9776.2665
8 1800	1494	2014-01-23 00:00:00.000	707.784	221.1825	9776.2665
9 1721	1494	2014-01-16 00:00:00.000	707.784	221.1825	9776.2665
10 2116	1494	2014-02-26 00:00:00.000	707.784	221.1825	9776.2665
11 2195	1494	2014-03-04 00:00:00.000	707.784	221.1825	9776.2665
12 2037	1494	2014-02-19 00:00:00.000	707.784	221.1825	9776.2665
13 1563	1494	2013-12-30 00:00:00.000	707.784	221.1825	9776.2665
14 1642	1494	2014-01-06 00:00:00.000	707.784	221.1825	9776.2665
15 1484	1494	2013-12-19 00:00:00.000	707.784	221.1825	9776.2665
16 1326	1494	2013-12-04 00:00:00.000	707.784	221.1825	9776.2665
17 1405	1494	2013-12-12 00:00:00.000	707.784	221.1825	9776.2665
18 1168	1494	2013-11-13 00:00:00.000	707.784	221.1825	9776.2665
19 1247	1494	2013-11-25 00:00:00.000	707.784	221.1825	9776.2665
20 760	1494	2013-09-01 00:00:00.000	707.784	221.1825	9776.2665
21 677	1494	2013-08-25 00:00:00.000	707.784	221.1825	9776.2665
22 590	1494	2013-08-17 00:00:00.000	707.784	221.1825	9776.2665
23 926	1494	2013-09-17 00:00:00.000	707.784	221.1825	9776.2665
24 843	1494	2013-09-08 00:00:00.000	707.784	221.1825	9776.2665
25 1025	1494	2013-10-13 00:00:00.000	707.784	221.1825	9776.2665
26 1089	1494	2013-10-30 00:00:00.000	707.784	221.1825	9776.2665
27 398	1494	2013-06-25 00:00:00.000	707.784	221.1825	9776.2665
28 319	1494	2013-04-24 00:00:00.000	707.784	221.1825	9776.2665
29 426	1494	2013-08-04 00:00:00.000	707.784	221.1825	9776.2665
30 507	1494	2013-08-11 00:00:00.000	707.784	221.1825	9776.2665
31 240	1494	2012-09-05 00:00:00.000	707.784	221.1825	9776.2665
32 161	1494	2012-05-30 00:00:00.000	707.784	221.1825	9776.2665

Query executed successfully.

LAPTOP-QU02VRVS\SQLEXPRESS ... | LAPTOP-QU02VRVS\sabin ... | AdventureWorks2019 | 00:00:00 | 3,142 rows

Ready AUD/GBP +0.38% Search Ln 18 Col 7 Ch 7 INS ENG US 10:23 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

AdventureWorks2019 - Execute Results Messages

Object Explorer Connect ...

Databases Security Server Objects Replication Management XEvent Profiler

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 3307	1492	2014-06-13 00:00:00.000	61.9164	19.3489	855.2203
5 167	1492	2012-05-30 00:00:00.000	56.8764	17.7739	785.6053
6 1000	1494	2014-02-28 00:00:00.000	707.784	221.1825	9776.2665
7 1958	1494	2014-02-11 00:00:00.000	707.784	221.1825	9776.2665
8 1800	1494	2014-01-23 00:00:00.000	707.784	221.1825	9776.2665
9 1721	1494	2014-01-16 00:00:00.000	707.784	221.1825	9776.2665
10 2116	1494	2014-02-26 00:00:00.000	707.784	221.1825	9776.2665
11 2195	1494	2014-03-04 00:00:00.000	707.784	221.1825	9776.2665
12 2037	1494	2014-02-19 00:00:00.000	707.784	221.1825	9776.2665
13 1563	1494	2013-12-30 00:00:00.000	707.784	221.1825	9776.2665
14 1642	1494	2014-01-06 00:00:00.000	707.784	221.1825	9776.2665
15 1484	1494	2013-12-19 00:00:00.000	707.784	221.1825	9776.2665
16 1326	1494	2013-12-04 00:00:00.000	707.784	221.1825	9776.2665
17 1405	1494	2013-12-12 00:00:00.000	707.784	221.1825	9776.2665
18 1168	1494	2013-11-13 00:00:00.000	707.784	221.1825	9776.2665
19 1247	1494	2013-11-25 00:00:00.000	707.784	221.1825	9776.2665
20 760	1494	2013-09-01 00:00:00.000	707.784	221.1825	9776.2665
21 677	1494	2013-08-25 00:00:00.000	707.784	221.1825	9776.2665
22 590	1494	2013-08-17 00:00:00.000	707.784	221.1825	9776.2665
23 926	1494	2013-09-17 00:00:00.000	707.784	221.1825	9776.2665
24 843	1494	2013-09-08 00:00:00.000	707.784	221.1825	9776.2665
25 1025	1494	2013-10-13 00:00:00.000	707.784	221.1825	9776.2665
26 1089	1494	2013-10-30 00:00:00.000	707.784	221.1825	9776.2665
27 398	1494	2013-06-25 00:00:00.000	707.784	221.1825	9776.2665
28 319	1494	2013-04-24 00:00:00.000	707.784	221.1825	9776.2665
29 426	1494	2013-08-04 00:00:00.000	707.784	221.1825	9776.2665
30 507	1494	2013-08-11 00:00:00.000	707.784	221.1825	9776.2665
31 240	1494	2012-09-05 00:00:00.000	707.784	221.1825	9776.2665
32 161	1494	2012-05-30 00:00:00.000	707.784	221.1825	9776.2665

Query executed successfully.

LAPTOP-QU02VRVS\SQLEXPRESS ... | LAPTOP-QU02VRVS\sabin ... | AdventureWorks2019 | 00:00:00 | 3,142 rows

Ready Tomorrow's low Near record Search Ln 18 Col 7 Ch 7 INS ENG US 10:23 PM 2/05/2024

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.

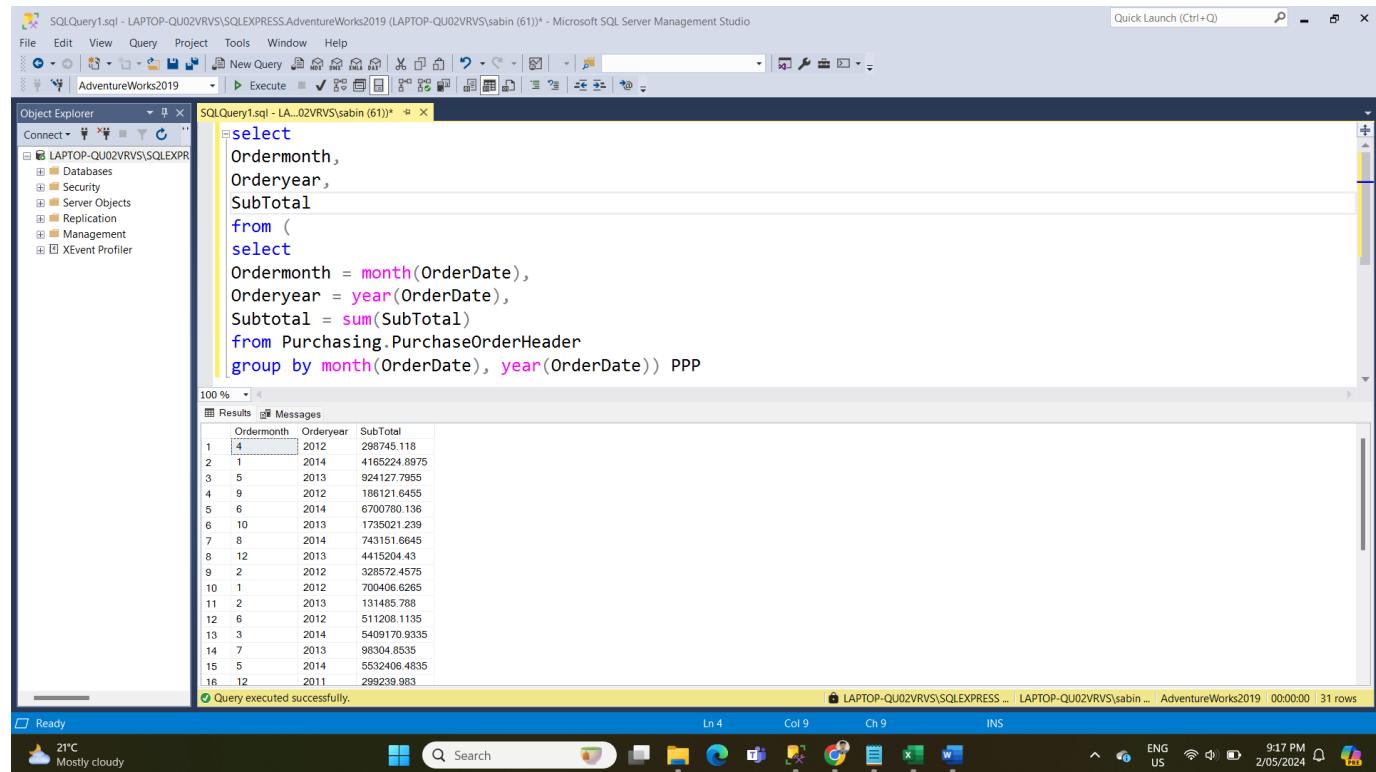
"OrderYear", a derived column featuring the year corresponding with the Order Date in a given row.

"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
```



The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled 'SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61))* - Microsoft SQL Server Management Studio' is open. The query itself is the one provided above, selecting Ordermonth, Orderyear, and SubTotal from the Purchasing.PurchaseOrderHeader table, grouped by month and year. Below the query, the results pane displays a table with 31 rows of data. The columns are Ordermonth, Orderyear, and SubTotal. The data shows various months and years with their corresponding SubTotal values. At the bottom of the results pane, a message indicates 'Query executed successfully.'

Ordermonth	Orderyear	SubTotal
1	2012	298745.118
2	2014	4165224.8975
3	2013	924127.7955
4	2012	1861216.0455
5	2014	6700780.136
6	2013	1735021.239
7	2014	743151.6645
8	2013	4415204.43
9	2012	328572.4575
10	2012	700406.6265
11	2013	131485.788
12	2012	511208.1135
13	2014	5409170.9335
14	2013	98304.8535
15	2014	5532406.4835
16	2011	298239.983

The screenshot shows the Microsoft SQL Server Management Studio interface. A query window titled "SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61))" is open, displaying a table of data. The table has three columns: Ordermonth, Orderyear, and SubTotal. The data is sorted by Ordermonth and Orderyear. The results grid contains 31 rows of data.

	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	388668.9805
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

Query executed successfully.

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

```

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	616042.598
9	4	2012	298745.118	5811811.8725
10	4	2013	641097.3975	1043738.3365
11	4	2014	5493161.691	6433004.2065
12	5	2012	250687.982	6384947.0505
13	5	2013	924127.7955	6667977.4485
14	5	2014	5532406.4835	6707222.241

Query executed successfully.

LN 11 Col 36 Ch 36 INS

Ready Temps to drop Tomorrow

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61)) - Microsoft SQL Server Management Studio

```

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	616042.598
9	4	2012	298745.118	5811811.8725
10	4	2013	641097.3975	1043738.3365
11	4	2014	5493161.691	6433004.2065
12	5	2012	250687.982	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	160343.179	6966137.8265
19	7	2013	98304.8535	6959428.169
20	7	2014	6824989.454	7083637.491
21	8	2012	454963.1205	7378257.432
22	8	2013	5155274.2185	12435226.797
23	8	2014	743151.6845	6353389.0035
24	9	2012	186121.6455	6084547.5285
25	9	2013	3481085.8635	4410399.1735
26	9	2014	1020.00	3668227.509
27	10	2012	388688.9805	3870974.844
28	10	2013	1735021.238	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.

LN 1 Col 1 INS

Ready Temps to drop Tomorrow

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
```

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61))* - Microsoft SQL Server Management Studio

```

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

```

Results

Ordermonth	Orderyear	SubTotal	Rolling3MonthTotal	MovingAvg6Month
1	2012	700406.6265	700406.6265	NULL
2	2014	4165224.8975	4865631.524	700406.6265
3	2012	3265724.4575	5194203.9815	2432815.762
4	2013	131485.788	4625283.143	1731401.3271
5	2014	4527263.0445	4987321.29	1331422.4423
6	2012	646975.8435	5305724.676	1970590.5628
7	2014	5409170.9335	10583409.8215	1749988.1095
8	2011	103895.821	6160042.598	2534782.1607
9	2012	296745.118	5811811.8725	1857893.9813
10	2013	641097.3975	1043738.3365	1852922.758
11	2014	5493161.691	6433004.2065	1937858.0263
12	2012	250687.962	6384947.0505	2098841.134
13	2013	924127.7955	6667977.4485	2032793.1538

Query executed successfully.

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61))* - Microsoft SQL Server Management Studio

```

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

```

Results

Ordermonth	Orderyear	SubTotal	Rolling3MonthTotal	MovingAvg6Month
1	2012	700406.6265	700406.6265	NULL
2	2014	4165224.8975	4865631.524	700406.6265
3	2012	3265724.4575	5194203.9815	2432815.762
4	2013	131485.788	4625283.143	1731401.3271
5	2014	4527263.0445	4987321.29	1331422.4423
6	2012	646975.8435	5305724.676	1970590.5628
7	2014	5409170.9335	10583409.8215	1749988.1095
8	2011	103895.821	6160042.598	2534782.1607
9	2012	296745.118	5811811.8725	1857893.9813
10	2013	641097.3975	1043738.3365	1852922.758
11	2014	5493161.691	6433004.2065	1937858.0263
12	2012	250687.962	6384947.0505	2098841.134
13	2013	924127.7955	6667977.4485	2032793.1538
14	2014	5532406.4835	6707222.241	1285285.9641
15	6	511201.1135	6967742.3925	2190037.7412
16	2013	105014.511	6148629.108	2225448.2405
17	6	6700780.136	7317002.7605	2136101.0927
18	7	160343.1795	6966137.8265	2337370.8335
19	7	98304.8535	6959428.169	2322313.3698
20	7	6824989.458	7083637.491	2184676.2128
21	8	454963.1205	7378257.432	2400106.7085
22	8	5155274.2185	12435226.797	2390732.543
23	8	743151.6645	6353368.0005	3232442.4943
24	9	186121.6455	6084547.5285	2239504.4157
25	9	3481085.8635	4410359.175	2243800.8267
26	9	1020.00	3668227.509	2807597.6617
27	10	388686.8905	3870974.844	1670269.4187
28	10	1735021.239	2124910.2195	1659253.7287
29	11	3378181.582	5502071.8015	1089211.5655
30	12	299239.983	5412442.804	1528383.2184
31	12	4415204.43	8092625.995	1547236.2746

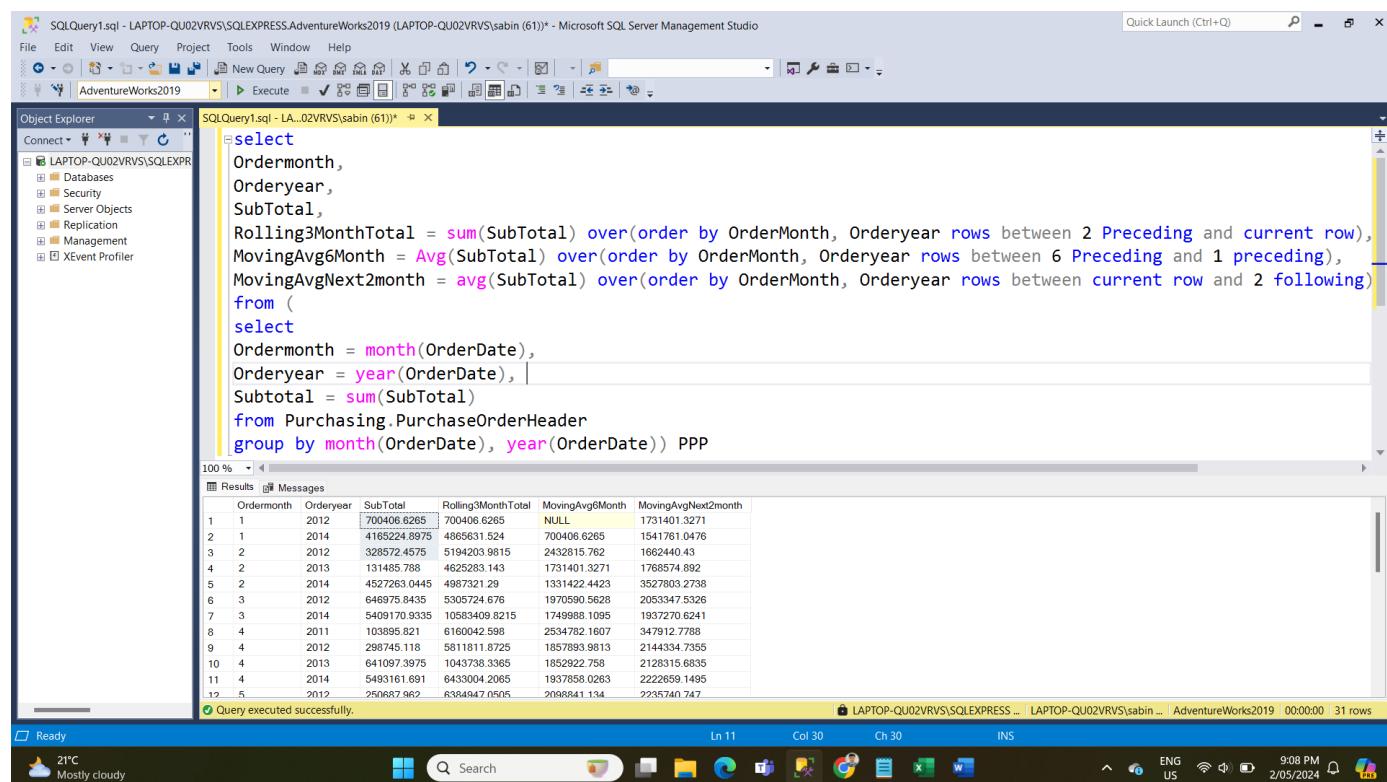
Query executed successfully.

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
```



The screenshot shows the Microsoft SQL Server Management Studio interface. The query window displays the T-SQL code for calculating moving averages. Below the code, the results pane shows a table with 12 rows of data. The columns are Ordermonth, Orderyear, SubTotal, Rolling3MonthTotal, MovingAvg6Month, and MovingAvgNext2month. The data includes various months from 2012 to 2014, with corresponding subtotal values and the calculated moving averages.

	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	646975.9435	5305724.676	1970590.5628	2053347.5326
7	3	2014	5409170.9335	10583409.8215	1749888.1095	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	641087.3975	1043738.3365	1852922.758	2128315.6835
11	4	2014	5493161.691	6433004.2065	1937858.0263	2222659.1495
12	5	2012	250827.962	6364947.0505	2098841.134	2235740.747

SQLQuery1.sql - LAPTOP-QU02VRVS\SQLEXPRESS.AdventureWorks2019 (LAPTOP-QU02VRVS\sabin (61)) - Microsoft SQL Server Management Studio

File Edit View Query Project Tools Window Help

AdventureWorks2019 Execute

Object Explorer Connect Object Explorer Details

SQLQuery1.sql - LA...02VRVS\sabin (61)*

Results Messages

	Ordermonth	Orderyear	SubTotal	Rolling3MonthTotal	MovingAvg6Month	MovingAvgNext2month
1	1	2012	700406.6265	700406.6265	NULL	1731401.3271
2	1	2014	416524.8975	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	452763.0445	4987321.29	1331422.4423	3527803.7738
6	3	2012	646975.8435	5305724.678	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	641097.397	1043738.3365	1852922.758	2128315.6833
11	4	2014	54893161.691	6433004.2063	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	511206.1135	6967742.3925	2190037.7412	2439000.9201
16	6	2013	105014.511	6148629.108	2225448.2405	2322045.9421
17	6	2014	6700780.136	7317002.760	2136101.0927	2319809.3896
18	7	2012	1660343.795	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	1470118.7245
24	9	2012	186121.645	6084547.5285	2239504.4157	1222742.503
25	9	2013	3481085.8635	4110359.173	2243800.8267	1290324.948
26	9	2014	1020.00	3668227.509	2807597.6617	708303.4065
27	10	2012	388868.9803	3870974.844	1670269.4187	1834023.9333
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

Query executed successfully.

Ln 31 Col 1 INS

Ready 21°C Mostly cloudy Search ENG US 9:09 PM 2/05/2024