

SQL Practice Problems

57 beginning, intermediate,
and advanced challenges for
you to solve using a "learn-
by-doing" approach

Sylvia Moestl Vasilik

Intermediate Problems

20. Categories, and the total products in each category

For this problem, we'd like to see the total number of products in each category. Sort the results by the total number of products, in descending order.

Expected Results

CategoryName	TotalProducts

Confections	13
Beverages	12
Condiments	12
Seafood	12
Dairy Products	10
Grains/Cereals	7
Meat/Poultry	6
Produce	5

(8 row(s) affected)

Hint

To solve this problem, you need to combine a join, and a group by.

A good way to start is by creating a query that shows the CategoryName and all ProductIDs associated with it, without grouping. Then, add the Group by

21. Total customers per country/city

In the Customers table, show the total number of customers per Country and City.

Expected Results

Country	City	TotalCustomer

UK	London	6
Mexico	México D.F.	5
Brazil	Sao Paulo	4
Brazil	Rio de Janeiro	3
Spain	Madrid	3
Argentina	Buenos Aires	3
France	Paris	2
USA	Portland	2
France	Nantes	2
Portugal	Lisboa	2
Finland	Oulu	1
Italy	Reggio Emilia	1
France	Reims	1
Brazil	Resende	1

... (skipping some rows)

Canada	Montréal	1
Germany	München	1
Germany	Münster	1
Germany	Aachen	1
USA	Albuquerque	1
USA	Anchorage	1
Denmark	Århus	1
Spain	Barcelona	1
Venezuela	Barquisimeto	1
Italy	Bergamo	1
Germany	Berlin	1
Switzerland	Bern	1
USA	Boise	1
Sweden	Bräcke	1
Germany	Brandenburg	1
Belgium	Bruxelles	1

(69 row(s) affected)

Hint

Just as you can have multiple fields in a Select clause, you can also have multiple fields in a Group By clause.

22. Products that need reordering

What products do we have in our inventory that should be reordered? For now, just use the fields `UnitsInStock` and `ReorderLevel`, where `UnitsInStock` is less than the `ReorderLevel`, ignoring the fields `UnitsOnOrder` and `Discontinued`.

Order the results by `ProductID`.

Expected Results

ProductID	ProductName	UnitsInStock	ReorderLevel
2	Chang	17	25
3	Aniseed Syrup	13	25
11	Queso Cabrales	22	30
21	Sir Rodney's Scones	3	5
30	Nord-Ost Matjeshering	10	15
31	Gorgonzola Telino	0	20
32	Mascarpone Fabioli	9	25
37	Gravad lax	11	25
43	Ipoh Coffee	17	25
45	Rogede sild	5	15
48	Chocolade	15	25
49	Maxilaku	10	15
56	Gnocchi di nonna Alice	21	30
64	Wimmers gute Semmelknödel	22	30
66	Louisiana Hot Spiced Okra	4	20
68	Scottish Longbreads	6	15
70	Outback Lager	15	30
74	Longlife Tofu	4	5

(18 row(s) affected)

Hint

We want to show all fields where the UnitsInStock is less than the ReorderLevel. So in the Where clause, use the following:

```
UnitsInStock < ReorderLevel
```

23. Products that need reordering, continued

Now we need to incorporate these fields—UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued—into our calculation. We'll define “products that need reordering” with the following:

- UnitsInStock plus UnitsOnOrder are less than or equal to ReorderLevel
- The Discontinued flag is false (0).

Expected Results

ProductID	ProductName	UnitsInStock	UnitsOnOrder	ReorderLevel	Discontinued
30	Nord-Ost Matjeshering	10	0	15	0
70	Outback Lager	15	10	30	0

(2 row(s) affected)

Hint

For the first part of the Where clause, you should have something like this:

```
UnitsInStock + UnitsOnOrder <= ReorderLevel
```

24. Customer list by region

A salesperson for Northwind is going on a business trip to visit customers, and would like to see a list of all customers, sorted by region, alphabetically.

However, he wants the customers with no region (null in the Region field) to be at the end, instead of at the top, where you'd normally find the null values. Within the same region, companies should be sorted by CustomerID.

Expected Results

CustomerID	CompanyName	Region
OLDWO	Old World Delicatessen	AK
BOTTM	Bottom-Dollar Markets	BC
LAUGB	Laughing Bacchus Wine Cellars	BC
LETSS	Let's Stop N Shop	CA
HUNGO	Hungry Owl All-Night Grocers	Co. Cork
GROSR	GROSELLA-Restaurante	DF
SAVEA	Save-a-lot Markets	ID
ISLAT	Island Trading	Isle of Wight
LILAS	LILA-Supermercado	Lara
THECR	The Cracker Box	MT
RATTC	Rattlesnake Canyon Grocery	NM

... (skipping some rows)

SANTG	Santé Gourmet	NULL
SEVES	Seven Seas Imports	NULL
SIMOB	Simons bistro	NULL
SPECD	Spécialités du monde	NULL
SUPRD	Suprêmes délices	NULL
TOMSP	Toms Spezialitäten	NULL
TORTU	Tortuga Restaurante	NULL
VAFFE	Vaffeljernet	NULL
VICTE	Victuailles en stock	NULL
VINET	Vins et alcools Chevalier	NULL
WANDK	Die Wandernde Kuh	NULL
WARTH	Wartian Herkku	NULL
WILMK	Wilman Kala	NULL
WOLZA	Wolski Zajazd	NULL

(91 row(s) affected)

Hint

You won't be able to sort directly on the Region field here. You'll need to sort on the Region field, and also on a computed field that you create, which will give you a secondary sort for when Region is null

First, without ordering, create a computed field that has a value which will sort the way you want. In this case, you can create a field with the Case statement, which allows you to do if/then logic. You want a field that is 1 when Region is null.

Take a look at the Examples section in the SQL Server documentation for Case (<https://msdn.microsoft.com/en-us/library/ms181765.aspx#examples>).

Note that when filtering for null values, you can't use "FieldName = Null". You must use "FieldName is null".

Hint

You should have something like this:

```
Select
  CustomerID
, CompanyName
, Region
, Case
    when Region is null then 1
    else 0
End
From Customers
```

When the Region contains a null, you will have a 1 in the final column.
Now, just add the fields for the Order By clause, in the right order.

25. High freight charges

Some of the countries we ship to have very high freight charges. We'd like to investigate some more shipping options for our customers, to be able to offer them lower freight charges. Return the three ship countries with the highest average freight overall, in descending order by average freight.

Expected Results

ShipCountry	AverageFreight
Austria	184.7875
Ireland	145.0126
USA	112.8794

(3 row(s) affected)

Hint

We'll be using the Orders table, and using the Freight and ShipCountry fields.

Hint

You'll want to group by ShipCountry, and use the Avg function. Don't worry about showing only the top 3 rows until you have the grouping and average freight set up.

Hint

You should have something like this:

Select

ShipCountry

,AverageFreight = avg(freight)

From Orders

Group By ShipCountry

Order By AverageFreight desc;

Now you just need to show just the top 3 rows.

26. High freight charges - 2015

We're continuing on the question above on high freight charges. Now, instead of using *all* the orders we have, we only want to see orders from the year 2015.

Expected result

ShipCountry	AverageFreight
Austria	178.3642
Switzerland	117.1775
France	113.991

(3 row(s) affected)

Hint

You need to add a Where clause to the query from the previous problem. The field to filter on is OrderDate.

Hint

When filtering on dates, you need to know whether the date field is a DateTime, or a Date field. Is OrderDate a Datetime or a Date field?

27. High freight charges with between

Another (incorrect) answer to the problem above is this:

Select Top 3

ShipCountry

,AverageFreight = avg(freight)

From Orders

Where

OrderDate between '1/1/2015' and '12/31/2015'

Group By ShipCountry

Order By AverageFreight desc;

Notice when you run this, it gives Sweden as the ShipCountry with the third highest freight charges. However, this is wrong - it should be France.

What is the OrderID of the order that the (incorrect) answer above is missing?

Expected Result

(no expected results this time - we're looking for a specific OrderID)

Hint

The Between statement is inclusive. Why isn't it showing the orders made on December 31, 2015?

Hint

Run this query, and look at the rows around December 31, 2015. What do you notice? Look specifically at the Freight field.

```
select * from orders order by OrderDate
```

28. High freight charges - last year

We're continuing to work on high freight charges. We now want to get the three ship countries with the highest average freight charges. But instead of filtering for a particular year, we want to use the last 12 months of order data, using as the end date the last OrderDate in Orders.

Expected Results

ShipCountry	AverageFreight
Ireland	200.21
Austria	186.4596
USA	119.3032

(3 row(s) affected)

Hint

First, get the last OrderDate in Orders. Write a simple select statement to get the highest value in the OrderDate field using the Max aggregate function.

Hint

You should have something like this:

```
Select Max(OrderDate) from Orders
```

Now you need to get the date 1 year before the last order date. Create a simple select statement that subtracts 1 year from the last order date

You can use the DateAdd function for this. Note that within DateAdd, you can use the subquery you created above. Look online for some examples if you need to.

Hint

You should have something like this:

```
Select Dateadd(yy, -1, (Select Max(OrderDate) from Orders))
```

Now you just need to put it in the where clause.

29. Inventory list

We're doing inventory, and need to show information like the below, for all orders. Sort by OrderID and Product ID.

Expected Results

EmployeeID	LastName	OrderID	ProductName	Quantity
5	Buchanan	10248	Queso Cabrales	12
5	Buchanan	10248	Singaporean Hokkien Fried Mee	10
5	Buchanan	10248	Mozzarella di Giovanni	5
6	Suyama	10249	Tofu	9
6	Suyama	10249	Manjimup Dried Apples	40
4	Peacock	10250	Jack's New England Clam Chowder	10
4	Peacock	10250	Manjimup Dried Apples	35
4	Peacock	10250	Louisiana Fiery Hot Pepper Sauce	15
3	Leverling	10251	Gustaf's Knäckebröd	6
3	Leverling	10251	Ravioli Angelo	15
3	Leverling	10251	Louisiana Fiery Hot Pepper Sauce	20
4	Peacock	10252	Sir Rodney's Marmalade	40
4	Peacock	10252	Geitost	25
4	Peacock	10252	Camembert Pierrot	40
3	Leverling	10253	Gorgonzola Telino	20
3	Leverling	10253	Chartreuse verte	42
3	Leverling	10253	Maxilaku	40

...
(total 2155 rows)

Hint

You'll need to do a join between 4 tables, displaying only those fields that are necessary.

30. Customers with no orders

There are some customers who have never actually placed an order.
Show these customers.

Expected Results

Customers_CustomerID	Orders_CustomerID
FISSA	NULL
PARIS	NULL

(2 row(s) affected)

Hint

One way of doing this is to use a left join, also known as a left outer join.

Hint

Select

```
Customers_CustomerID = Customers.CustomerID  
,Orders_CustomerID = Orders.CustomerID
```

From Customers

```
left join Orders  
on Orders.CustomerID = Customers.CustomerID
```

This is a good start. It shows all records from the Customers table, and the matching records from the Orders table. However, we only want those records where the CustomerID in Orders is null. You still need a filter

31. Customers with no orders for EmployeeID 4

One employee (Margaret Peacock, EmployeeID 4) has placed the most orders. However, there are some customers who've never placed an order with her. Show only those customers who have never placed an order with her.

Expected Result

CustomerID	CustomerID
SEVES	NULL
THEBI	NULL
LAZYK	NULL
GROSR	NULL
PARIS	NULL
FISSA	NULL
SPECD	NULL
LAUGB	NULL
PRINI	NULL
VINET	NULL
FRANR	NULL
CONSH	NULL
NORTS	NULL
PERIC	NULL
DUMON	NULL
SANTG	NULL

(16 row(s) affected)

Hint

Building on the previous problem, you might think you need to do something like this:

Select

```
Customers.CustomerID  
,Orders.CustomerID
```

From

```
Customers
```

```
left join Orders  
on Orders.CustomerID = Customers.CustomerID
```

Where

```
Orders.CustomerID is null  
and Orders.EmployeeID = 4
```

...adding this filter in the where clause:

```
and Orders.EmployeeID = 4
```

However, this returns no records.

Note that with outer joins, the filters on the where clause are applied *after* the join.

Congratulations! You've completed the intermediate problems

Any questions or feedback on the problems, hints, or answers? I'd like to hear from you. Please email me at feedback@SQLPracticeProblems.com.

Advanced Problems

32. High-value customers

We want to send all of our high-value customers a special VIP gift. We're defining high-value customers as those who've made at least 1 order with a total value (not including the discount) equal to \$10,000 or more. We only want to consider orders made in the year 2016.

Expected Result

CustomerID	CompanyName	OrderID	TotalOrderAmount
QUICK	QUICK-Stop	10865	17250.00
SAVEA	Save-a-lot Markets	11030	16321.90
HANAR	Hanari Carnes	10981	15810.00
KOENE	Königlich Essen	10817	11490.70
RATTC	Rattlesnake Canyon Grocery	10889	11380.00
HUNGO	Hungry Owl All-Night Grocers	10897	10835.24

(6 row(s) affected)

Hint

First, let's get the necessary fields for all orders made in the year 2016. Don't bother grouping yet, just work on the Where clause. You'll need the CustomerID, CompanyName from Customers; OrderID from Orders; and Quantity and unit price from OrderDetails. Order by the total amount of the order, in descending order.

Hint

You should have something like this:

Select

```
Customers.CustomerID  
,Customers.CompanyName  
,Orders.OrderID  
,Amount = Quantity * UnitPrice
```

From Customers

```
join Orders  
  on Orders.CustomerID = Customers.CustomerID  
join OrderDetails  
  on Orders.OrderID = OrderDetails.OrderID
```

Where

```
OrderDate >= '20160101'  
and OrderDate < '20170101'
```

This gives you the total amount for each Order Detail item in 2016 orders, at the Order Detail level. Now, which fields do you need to group on, and which need to be summed?

Hint

Select

```
Customers.CustomerID  
,Customers.CompanyName  
,Orders.OrderID  
,TotalOrderAmount = sum(Quantity * UnitPrice)
```

From Customers

```
Join Orders  
    on Orders.CustomerID = Customers.CustomerID  
Join OrderDetails  
    on Orders.OrderID = OrderDetails.OrderID
```

Where

```
OrderDate >= '20160101'  
and OrderDate < '20170101'
```

Group By

```
Customers.CustomerID  
,Customers.CompanyName  
,Orders.OrderID
```

The fields at the Customer and Order level need to be grouped by, and the TotalOrderAmount needs to be summed.

How would you filter on the sum, in order to get orders of \$10,000 or more? Can you put it straight into the where clause?

33. High-value customers - total orders

The manager has changed his mind. Instead of requiring that customers have at least one individual orders totaling \$10,000 or more, he wants to define high-value customers as those who have orders totaling \$15,000 or more in 2016. How would you change the answer to the problem above?

Expected Result

CustomerID	CompanyName	TotalOrderAmount
SAVEA	Save-a-lot Markets	42806.25
ERNSH	Ernst Handel	42598.90
QUICK	QUICK-Stop	40526.99
HANAR	Hanari Carnes	24238.05
HUNGO	Hungry Owl All-Night Grocers	22796.34
RATTC	Rattlesnake Canyon Grocery	21725.60
KOENE	Königlich Essen	20204.95
FOLKO	Folk och få HB	15973.85
WHITC	White Clover Markets	15278.90

(9 row(s) affected)

Hint

This query is almost identical to the one above, but there's just a few lines you need to delete or comment out, to group at a different level.

34. High-value customers - with discount

Change the above query to use the discount when calculating high-value customers. Order by the total amount which includes the discount.

Expected Result

CustomerID	CompanyName	TotalsWithoutDiscount	TotalsWithDiscount
ERNSH	Ernst Handel	42598.90	41210.6500244141
QUICK	QUICK-Stop	40526.99	37217.3150024414
SAVEA	Save-a-lot Markets	42806.25	36310.1097793579
HANAR	Hanari Carnes	24238.05	23821.1999893188
RATTC	Rattlesnake Canyon Grocery	21725.60	21238.2704410553
HUNGO	Hungry Owl All-Night Grocers	22796.34	20402.119934082
KOENE	Königlich Essen	20204.95	19582.7739868164
WHITC	White Clover Markets	15278.90	15278.8999862671
FOLKO	Folk och få HB	15973.85	13644.0674972534
SUPRD	Suprêmes délices	11862.50	11644.5999984741
BOTTM	Bottom-Dollar Markets	12227.40	11338.5500488281

(11 row(s) affected)

Hint

To start out, just use the OrderDetails table. You'll need to figure out how the Discount field is structured.

Hint

You should have something like this:

Select

OrderID

,ProductID

,UnitPrice

,Quantity

,Discount

,TotalWithDisccount = UnitPrice * Quantity * (1 - Discount)

from OrderDetails

Note that Discount is applied as a percentage. So, if there's a 0.15 in the discount field, you need to multiply the UnitPrice * Quantity by .85 (1.00 - .15). You need parenthesis around (1 - Discount) to make sure that calculation is done first.

35. Month-end orders

At the end of the month, salespeople are likely to try much harder to get orders, to meet their month-end quotas. Show all orders made on the last day of the month. Order by EmployeeID and OrderID

Expected Result

EmployeeID	OrderID	OrderDate

1	10461	2015-02-28 00:00:00.000
1	10616	2015-07-31 00:00:00.000
2	10583	2015-06-30 00:00:00.000
2	10686	2015-09-30 00:00:00.000
2	10989	2016-03-31 00:00:00.000
2	11060	2016-04-30 00:00:00.000
3	10432	2015-01-31 00:00:00.000
3	10806	2015-12-31 00:00:00.000
3	10988	2016-03-31 00:00:00.000
3	11063	2016-04-30 00:00:00.000
4	10343	2014-10-31 00:00:00.000
4	10522	2015-04-30 00:00:00.000
4	10584	2015-06-30 00:00:00.000
4	10617	2015-07-31 00:00:00.000
4	10725	2015-10-31 00:00:00.000
4	10807	2015-12-31 00:00:00.000
4	11061	2016-04-30 00:00:00.000
4	11062	2016-04-30 00:00:00.000
5	10269	2014-07-31 00:00:00.000
6	10317	2014-09-30 00:00:00.000
7	10490	2015-03-31 00:00:00.000
8	10399	2014-12-31 00:00:00.000
8	10460	2015-02-28 00:00:00.000
8	10491	2015-03-31 00:00:00.000
8	10987	2016-03-31 00:00:00.000
9	10687	2015-09-30 00:00:00.000

(26 row(s) affected)

Hint

You can work on calculating this yourself, with a combination of date functions such as `DateAdd` and `DateDiff`. But feel free to shortcut the process by doing some research online.

36. Orders with many line items

The Northwind mobile app developers are testing an app that customers will use to show orders. In order to make sure that even the largest orders will show up correctly on the app, they'd like some samples of orders that have lots of individual line items. Show the 10 orders with the most line items, in order of total line items.

Expected Result

OrderID	TotalOrderDetails
11077	25
10979	6
10657	6
10847	6
10845	5
10836	5
10714	5
10670	5
10691	5
10698	5

(10 row(s) affected)

Hint

Using Orders and OrderDetails, you'll use Group by and count() functionality.

37. Orders - random assortment

The Northwind mobile app developers would now like to just get a random assortment of orders for beta testing on their app. Show a random set of 2% of all orders.

Expected Result

(note - your results will be different, because we're returning a random set)

OrderID

11034
10400
10948
10931
10942
10604
10350
10499
10927
10896
10774
10932
10592
10706
10479
10782
10898

(17 row(s) affected)

Hint

Note that in the below SQL, the RandomValue field returns the *same* random value for each row. Do some research online to figure out how to get a *new* random value for each row.

```
Select  
    OrderID  
    , RandomValue = Rand()  
From Orders
```

38. Orders - accidental double-entry

Janet Leverling, one of the salespeople, has come to you with a request. She thinks that she accidentally double-entered a line item on an order, with a different ProductID, but the same quantity. She remembers that the quantity was 60 or more. Show all the OrderIDs with line items that match this, in order of OrderID.

Expected Result

OrderID

10263

10263

10990

10658

11030

(5 row(s) affected)

Hint

You might start out with something like this:

```
Select
  OrderID
, ProductID
, Quantity
From OrderDetails
Where Quantity >= 60
```

However, this will only give us the orders where at least one order detail has a quantity of 60 or more. We need to show orders with *more* than one order detail with a quantity of 60 or more. Also, the same value for quantity needs to be there more than once.

Hint

In addition to grouping on the OrderID, we also need to group by the Quantity, since we need to show the order details that have the same quantity, within an order. So, we need to group by both OrderID, and Quantity.

39. Orders - accidental double-entry details

Based on the previous question, we now want to show details of the order, for orders that match the above criteria.

Expected Result

OrderID	ProductID	UnitPrice	Quantity	Discount
10263	16	13.90	60	0.25
10263	30	20.70	60	0.25
10263	24	3.60	65	0
10263	74	8.00	65	0.25
10658	60	34.00	55	0.05
10658	21	10.00	60	0
10658	40	18.40	70	0.05
10658	77	13.00	70	0.05
10990	34	14.00	60	0.15
10990	21	10.00	65	0
10990	55	24.00	65	0.15
10990	61	28.50	66	0.15
11030	29	123.79	60	0.25
11030	5	21.35	70	0
11030	2	19.00	100	0.25
11030	59	55.00	100	0.25

(16 row(s) affected)

Hint

There are many ways of doing this, including CTE (common table expression) and derived tables. I suggest using a CTE and a subquery. Here's a good article on CTEs (<https://technet.microsoft.com/en-us/library/ms175972.aspx>).

This is an example of a simple CTE in Northwind. It returns orders made by the oldest employee:

```
,with OldestEmployee as (  
Select top 1  
    EmployeeID  
from Employees  
order by BirthDate  
)  
Select  
    OrderID  
    ,OrderDate  
from Orders  
where  
    EmployeeID in (Select EmployeeID from OldestEmployee)
```

40. Orders - accidental double-entry details, derived table

Here's another way of getting the same results as in the previous problem, using a derived table instead of a CTE. However, there's a bug in this SQL. It returns 20 rows instead of 16. Correct the SQL.

Problem SQL:

```
Select
  OrderDetails.OrderID
,ProductID
,UnitPrice
,Quantity
,Discount
From OrderDetails
Join (
  Select
    OrderID
  From OrderDetails
  Where Quantity >= 60
  Group By OrderID, Quantity
  Having Count(*) > 1
) PotentialProblemOrders
on PotentialProblemOrders.OrderID = OrderDetails.OrderID
Order by OrderID, ProductID
```

Hint

Your first step should be to run the SQL in the derived table

```
Select  
  OrderID  
From OrderDetails  
Where Quantity >= 60  
Group By OrderID, Quantity  
Having Count(*) > 1
```

What do you notice about the results?

Hint

There are 2 rows for OrderID 10263, because there are 2 sets of rows that have the same, identical quantity, that's 60 or above.

When you do a join to a table that has duplicates, you will get duplicates in the output as well, unless you take steps to avoid it.

Find a single keyword that you can easily add to avoid duplicates in SQL.

41. Late orders

Some customers are complaining about their orders arriving late. Which orders are late?

Expected Result

OrderID	OrderDate	RequiredDate	ShippedDate
10264	2014-07-24	2014-08-21	2014-08-23
10271	2014-08-01	2014-08-29	2014-08-30
10280	2014-08-14	2014-09-11	2014-09-12
10302	2014-09-10	2014-10-08	2014-10-09
10309	2014-09-19	2014-10-17	2014-10-23
10380	2014-12-12	2015-01-09	2015-01-16
10423	2015-01-23	2015-02-06	2015-02-24
10427	2015-01-27	2015-02-24	2015-03-03
10433	2015-02-03	2015-03-03	2015-03-04
10451	2015-02-19	2015-03-05	2015-03-12
10483	2015-03-24	2015-04-21	2015-04-25
10515	2015-04-23	2015-05-07	2015-05-23
10523	2015-05-01	2015-05-29	2015-05-30
10545	2015-05-22	2015-06-19	2015-06-26
10578	2015-06-24	2015-07-22	2015-07-25
10593	2015-07-09	2015-08-06	2015-08-13
10596	2015-07-11	2015-08-08	2015-08-12
10663	2015-09-10	2015-09-24	2015-10-03
10687	2015-09-30	2015-10-28	2015-10-30
10660	2015-09-08	2015-10-06	2015-10-15
10705	2015-10-15	2015-11-12	2015-11-18
10709	2015-10-17	2015-11-14	2015-11-20
10726	2015-11-03	2015-11-17	2015-12-05
10727	2015-11-03	2015-12-01	2015-12-05
10749	2015-11-20	2015-12-18	2015-12-19
10777	2015-12-15	2015-12-29	2016-01-21
10779	2015-12-16	2016-01-13	2016-01-14
10788	2015-12-22	2016-01-19	2016-01-19
10807	2015-12-31	2016-01-28	2016-01-30
10816	2016-01-06	2016-02-03	2016-02-04
10827	2016-01-12	2016-01-26	2016-02-06
10828	2016-01-13	2016-01-27	2016-02-04
10847	2016-01-22	2016-02-05	2016-02-10
10924	2016-03-04	2016-04-01	2016-04-08
10927	2016-03-05	2016-04-02	2016-04-08
10960	2016-03-19	2016-04-02	2016-04-08
10970	2016-03-24	2016-04-07	2016-04-24
10978	2016-03-26	2016-04-23	2016-04-23
10998	2016-04-03	2016-04-17	2016-04-17

(39 row(s) affected)

Hint

To determine which orders are late, you can use a combination of the `RequiredDate` and `ShippedDate`. It's not exact, but if `ShippedDate` is actually AFTER `RequiredDate`, you can be sure it's late.

42. Late orders - which employees?

Some salespeople have more orders arriving late than others. Maybe they're not following up on the order process, and need more training. Which salespeople have the most orders arriving late?

Expected Result

EmployeeID	LastName	TotalLateOrders
4	Peacock	10
3	Leverling	5
8	Callahan	5
9	Dodsworth	5
7	King	4
2	Fuller	4
1	Davolio	3
6	Suyama	3

(8 row(s) affected)

Hint

The answer from the problem above is a good starting point. You'll need to join to the Employee table to get the last name, and also add Count to show the total late orders.

43. Late orders vs. total orders

Andrew, the VP of sales, has been doing some more thinking some more about the problem of late orders. He realizes that just looking at the number of orders arriving late for each salesperson isn't a good idea. It needs to be compared against the *total* number of orders per salesperson. Return results like the following:

Expected Result

EmployeeID	LastName	AllOrders	LateOrders
1	Davolio	123	3
2	Fuller	96	4
3	Leverling	127	5
4	Peacock	156	10
6	Suyama	67	3
7	King	72	4
8	Callahan	104	5
9	Dodsworth	43	5

(8 row(s) affected)

Hint

You can use more than one CTE in a query. That would be a straightforward way of solving this problem.

Hint

Here are 2 SQL statements that could be put into CTEs and put together into a final SQL statement.

-- Late orders

```
Select
    EmployeeID
    ,TotalOrders = Count(*)
From Orders
Where
    RequiredDate <= ShippedDate
Group By
    EmployeeID
```

-- Total orders

```
Select
    EmployeeID
    ,TotalOrders = Count(*)
From Orders
Group By
    EmployeeID
```

44. Late orders vs. total orders - missing employee

There's an employee missing in the answer from the problem above. Fix the SQL to show all employees who have taken orders.

Expected Result

EmployeeID	LastName	AllOrders	LateOrders
1	Davolio	123	3
2	Fuller	96	4
3	Leverling	127	5
4	Peacock	156	10
5	Buchanan	42	NULL
6	Suyama	67	3
7	King	72	4
8	Callahan	104	5
9	Dodsworth	43	5

(9 row(s) affected)

Hint

How many rows are returned when you run just the AllOrders CTE?

How about when you run just the LateOrders CTE?

Hint

You'll want to add a left join (also known as a left outer join), to make sure that we show a row, even if there are no late orders.

45. Late orders vs. total orders - fix null

Continuing on the answer for above query, let's fix the results for row 5 - Buchanan. He should have a 0 instead of a Null in LateOrders.

Expected Result

EmployeeID	LastName	AllOrders	LateOrders
1	Davolio	123	3
2	Fuller	96	4
3	Leverling	127	5
4	Peacock	156	10
5	Buchanan	42	0
6	Suyama	67	3
7	King	72	4
8	Callahan	104	5
9	Dodsworth	43	5

(9 row(s) affected)

Hint

Find a function to test if a value is null, and return a different value when it is.

46. Late orders vs. total orders - percentage

Now we want to get the percentage of late orders over total orders.

Expected Result

EmployeeID	LastName	AllOrders	LateOrders	PercentLateOrders
1	Davolio	123	3	0.0243902439024
2	Fuller	96	4	0.041666666666666
3	Leverling	127	5	0.0393700787401
4	Peacock	156	10	0.0641025641025
5	Buchanan	42	0	0.000000000000000
6	Suyama	67	3	0.0447761194029
7	King	72	4	0.055555555555555
8	Callahan	104	5	0.0480769230769
9	Dodsworth	43	5	0.1162790697674

(9 row(s) affected)

Hint

By dividing late orders by total orders, you should be able to get the percentage of orders that are late. However, there's a common problem people run into, which is that an integer divided by an integer returns an integer. For instance, if you run the following SQL to divide 3 by 2:

```
select 3/2
```

You'll get 1 instead of 1.5, because it will return the closest integer.

Do some research online to find the answer to this issue.

47. Late orders vs. total orders - fix decimal

So now for the `PercentageLateOrders`, we get a decimal value like we should. But to make the output easier to read, let's cut the `PercentLateOrders` off at 2 digits to the right of the decimal point.

Expected Result

EmployeeID	LastName	AllOrders	LateOrders	PercentLateOrders
1	Davolio	123	3	0.02
2	Fuller	96	4	0.04
3	Leverling	127	5	0.04
4	Peacock	156	10	0.06
5	Buchanan	42	0	0.00
6	Suyama	67	3	0.04
7	King	72	4	0.06
8	Callahan	104	5	0.05
9	Dodsworth	43	5	0.12

(9 row(s) affected)

Hint

One straightforward way of doing this would be to explicitly convert PercentageLateOrders to a specific Decimal data type. With the Decimal datatype, you can specify how many digits you want to the right of the decimal point

Hint

The calculation PercentLateOrders is getting a little long and complicated, and it can be tricky to get all the commas and parenthesis correct.

One way to simplify it is to break it down with an actual value instead of a calculation.

For instance:

```
Select convert(decimal(10,2), 0.0243902439024)
```

48. Customer grouping

Andrew Fuller, the VP of sales at Northwind, would like to do a sales campaign for existing customers. He'd like to categorize customers into groups, based on how much they ordered in 2016. Then, depending on which group the customer is in, he will target the customer with different sales materials.

The customer grouping categories are 0 to 1,000, 1,000 to 5,000, 5,000 to 10,000, and over 10,000.

A good starting point for this query is the answer from the problem “High-value customers - total orders. We don’t want to show customers who don’t have any orders in 2016.

Order the results by CustomerID.

Expected Result

CustomerID	CompanyName	TotalOrderAmount	CustomerGroup
ALFKI	Alfreds Futterkiste	2302.20	Medium
ANATR	Ana Trujillo Emparedados y helados	514.40	Low
ANTON	Antonio Moreno Taquería	660.00	Low
AROUT	Around the Horn	5838.50	High
BERGS	Berglunds snabbköp	8110.55	High
BLAUS	Blauer See Delikatessen	2160.00	Medium
BLONP	Blondesddsl père et fils	730.00	Low
BOLID	Bólido Comidas preparadas	280.00	Low
BONAP	Bon app'	7185.90	High
BOTTM	Bottom-Dollar Markets	12227.40	Very High
BSBEV	B's Beverages	2431.00	Medium
CACTU	Cactus Comidas para llevar	1576.80	Medium
CHOPS	Chop-suey Chinese	4429.40	Medium
... (skipping some rows)			
SPLIR	Split Rail Beer & Ale	1117.00	Medium
SUPRD	Suprêmes délices	11862.50	Very High
THEBI	The Big Cheese	69.60	Low
THECR	The Cracker Box	326.00	Low
TOMSP	Toms Spezialitäten	910.40	Low
TORTU	Tortuga Restaurante	1874.50	Medium
TRADH	Tradição Hipermercados	4401.62	Medium
TRAIH	Trail's Head Gourmet Provisioners	237.90	Low
VAFFE	Vaffeljernet	4333.50	Medium
VICTE	Victuailles en stock	3022.00	Medium
WANDK	Die Wandernde Kuh	1564.00	Medium
WARTH	Wartian Herkku	300.00	Low
WELLI	Wellington Importadora	1245.00	Medium
WHITC	White Clover Markets	15278.90	Very High
WILMK	Wilman Kala	1987.00	Medium
WOLZA	Wolski Zajazd	1865.10	Medium

(81 row(s) affected)

Hint

This is the SQL from the problem “High-value customers - total orders”, but without the filter for order totals over 10,000.

Select

```
Customers.CustomerID  
,Customers.CompanyName  
,TotalOrderAmount = SUM(Quantity * UnitPrice)
```

From Customers

```
Join Orders  
    on Orders.CustomerID = Customers.CustomerID  
Join OrderDetails  
    on Orders.OrderID = OrderDetails.OrderID
```

Where

```
OrderDate >= '20160101'  
and OrderDate < '20170101'
```

Group By

```
Customers.CustomerID  
,Customers.CompanyName
```

Order By TotalOrderAmount Desc;

Hint

You can use the above SQL in a CTE (common table expression), and then build on it, using a Case statement on the TotalOrderAmount.

49. Customer grouping - fix null

There's a bug with the answer for the previous question. The CustomerGroup value for one of the rows is null.

Fix the SQL so that there are no nulls in the CustomerGroup field.

Expected Result

(Including only a subset of the output)

CustomerID	CompanyName	TotalOrderAmount	CustomerGroup
LILAS	LILA-Supermercado	5994.06	High
LINOD	LINO-Delicateses	10085.60	Very High
LONEP	Lonesome Pine Restaurant	1709.40	Medium
MAGAA	Magazzini Alimentari Riuniti	1693.00	Medium
MAISD	Maison Dewey	5000.20	High
MORGK	Morgenstern Gesundkost	245.00	Low
NORTS	North/South	45.00	Low
OCEAN	Océano Atlántico Ltda.	3031.00	Medium
OLDWO	Old World Delicatessen	5337.65	High
OTTIK	Ottilies Käseladen	3012.70	Medium
PERIC	Pericles Comidas clásicas	1496.00	Medium
PICCO	Piccolo und mehr	4393.75	Medium
PRINI	Princesa Isabel Vinhos	2633.90	Medium
QUEDE	Que Delícia	1353.60	Medium
QUEEN	Queen Cozinha	7007.65	High
QUICK	QUICK-Stop	40526.99	Very High
RANCH	Rancho grande	1694.70	Medium
RATTC	Rattlesnake Canyon Grocery	21725.60	Very High
REGGC	Reggiani Caseifici	4263.00	Medium
RICAR	Ricardo Adocicados	7312.00	High

Hint

What is the total order amount for CustomerID MAISD? How does that relate to our CustomerGroup boundaries?

Hint

Using “between” works well for integer values. However, the value we're working with is Money, which has decimals. Instead of something like:

```
when TotalOrderAmount between 0 and 1000 then 'Low'
```

You'll need to something like this:

```
when TotalOrderAmount >= 0 and TotalOrderAmount < 1000 then 'Low'
```

50. Customer grouping with percentage

Based on the above query, show all the defined CustomerGroups, and the percentage in each. Sort by the total in each group, in descending order.

Expected Result

CustomerGroup TotalInGroup PercentageInGroup

Medium	35	0.432098765432
Low	20	0.246913580246
High	13	0.160493827160
Very High	13	0.160493827160

(4 row(s) affected)

Hint

As a starting point, you can use the answer from the problem “Customer grouping - fix null”.

Hint

We no longer need to show the CustomerID and CompanyName in the final output. However, we need to count how many customers are in each CustomerGrouping. You can create another CTE level in order to get the counts in each CustomerGrouping for the final output.

51. Customer grouping - flexible

Andrew, the VP of Sales is still thinking about how best to group customers, and define low, medium, high, and very high value customers. He now wants complete flexibility in grouping the customers, based on the dollar amount they've ordered. He doesn't want to have to edit SQL in order to change the boundaries of the customer groups.

How would you write the SQL?

There's a table called CustomerGroupThreshold that you will need to use. Use only orders from 2016.

Expected Result

(The expected results are the same as for the original problem, it's just that we're getting the answer differently.)

CustomerID	CompanyName	TotalOrderAmount	CustomerGroupName
ALFKI	Alfreds Futterkiste	2302.20	Medium
ANATR	Ana Trujillo Emparedados y helados	514.40	Low
ANTON	Antonio Moreno Taquería	660.00	Low
AROUT	Around the Horn	5838.50	High
BERGS	Berglunds snabbköp	8110.55	High
BLAUS	Blauer See Delikatessen	2160.00	Medium
BLONP	Blondesddsl père et fils	730.00	Low
BOLID	Bólido Comidas preparadas	280.00	Low
BONAP	Bon app'	7185.90	High
BOTTM	Bottom-Dollar Markets	12227.40	Very High
BSBEV	B's Beverages	2431.00	Medium
CACTU	Cactus Comidas para llevar	1576.80	Medium
CHOPS	Chop-suey Chinese	4429.40	Medium
COMMI	Comércio Mineiro	513.75	Low
... (skipping some rows)			
SPLIR	Split Rail Beer & Ale	1117.00	Medium
SUPRD	Suprêmes délices	11862.50	Very High
THEBI	The Big Cheese	69.60	Low
THECR	The Cracker Box	326.00	Low
TOMSP	Toms Spezialitäten	910.40	Low
TORTU	Tortuga Restaurante	1874.50	Medium
TRADH	Tradição Hipermercados	4401.62	Medium
TRAIH	Trail's Head Gourmet Provisioners	237.90	Low
VAFFE	Vaffeljernet	4333.50	Medium
VICTE	Victuailles en stock	3022.00	Medium
WANDK	Die Wandernde Kuh	1564.00	Medium
WARTH	Wartian Herkku	300.00	Low
WELLI	Wellington Importadora	1245.00	Medium
WHITC	White Clover Markets	15278.90	Very High
WILMK	Wilman Kala	1987.00	Medium
WOLZA	Wolski Zajazd	1865.10	Medium

(81 row(s) affected)

Hint

As a starting point, use the SQL of the first CTE from the problem
“Customer grouping with percentage”

Select

```
Customers.CustomerID  
,Customers.CompanyName  
,TotalOrderAmount = SUM(Quantity * UnitPrice)
```

From Customers

```
join Orders  
on Orders.CustomerID = Customers.CustomerID  
join OrderDetails  
on Orders.OrderID = OrderDetails.OrderID
```

Where

```
OrderDate >= '20160101'  
and OrderDate < '20170101'
```

Group By

```
Customers.CustomerID  
,Customers.CompanyName
```

Hint

When thinking about how to use the table CustomerGroupThreshold, note that when joining to a table, you don't need to only use an equi-join (i.e., “=” in the join). You can also use other operators, such as between, and greater than/less than (> and <).

52. Countries with suppliers or customers

Some Northwind employees are planning a business trip, and would like to visit as many suppliers and customers as possible. For their planning, they'd like to see a list of all countries where suppliers and/or customers are based.

Expected Results

Country

Argentina
Australia
Austria
Belgium
Brazil
Canada
Denmark
Finland
France
Germany
Ireland
Italy
Japan
Mexico
Netherlands
Norway
Poland
Portugal
Singapore
Spain
Sweden
Switzerland
UK
USA
Venezuela

(25 row(s) affected)

Hint

Use the Union statement for this. It's a good way of putting together a simple resultset from multiple SQL statements.

53. Countries with suppliers or customers, version 2

The employees going on the business trip don't want just a raw list of countries, they want more details. We'd like to see output like the below, in the Expected Results.

Expected Result

SupplierCountry CustomerCountry

NULL	Argentina
Australia	NULL
NULL	Austria
NULL	Belgium
Brazil	Brazil
Canada	Canada
Denmark	Denmark
Finland	Finland
France	France
Germany	Germany
NULL	Ireland
Italy	Italy
Japan	NULL
NULL	Mexico
Netherlands	NULL
Norway	Norway
NULL	Poland
NULL	Portugal
Singapore	NULL
Spain	Spain
Sweden	Sweden
NULL	Switzerland
UK	UK
USA	USA
NULL	Venezuela

(25 row(s) affected)

Hint

A good way to start would be with a list of countries from the Suppliers table, and a list of countries from the Customers table. Use either Distinct or Group by to avoid duplicating countries. Sort by country name

Hint

You should have something like this:

```
Select Distinct Country from Customers  
Select Distinct Country from Suppliers
```

You can combine these with a CTEs or derived tables.

Note that there's a specific type of outer join you'll need, designed to return rows from *either* resultset. What is it? Look online for the different types of outer join available.

54. Countries with suppliers or customers - version 3

The output of the above is improved, but it's still not ideal

What we'd really like to see is the country name, the total suppliers, and the total customers.

Expected Result

Country	TotalSuppliers	TotalCustomers
Argentina	0	3
Australia	2	0
Austria	0	2
Belgium	0	2
Brazil	1	9
Canada	2	3
Denmark	1	2
Finland	1	2
France	3	11
Germany	3	11
Ireland	0	1
Italy	2	3
Japan	2	0
Mexico	0	5
Netherlands	1	0
Norway	1	1
Poland	0	1
Portugal	0	2
Singapore	1	0
Spain	1	5
Sweden	2	2
Switzerland	0	2
UK	2	7
USA	4	13
Venezuela	0	4

(25 row(s) affected)

Hint

You should be able to use the above query, and make a few changes to the CTE source queries to show the total number of Supplier countries and Customer countries. You won't be able to use the Distinct keyword anymore.

Hint

When joining the 2 CTEs together, you can use a computed column, with the IsNull function to show a non-null Country field, instead of the Supplier country or the Customer country.

55. First order in each country

Looking at the Orders table—we'd like to show details for each order that was the first in that particular country, ordered by OrderID.

So, we need one row per ShipCountry, and CustomerID, OrderID, and OrderDate should be of the first order from that country.

Expected Results

ShipCountry	CustomerID	OrderID	OrderDate

Argentina	OCEAN	10409	2015-01-09
Austria	ERNSH	10258	2014-07-17
Belgium	SUPRD	10252	2014-07-09
Brazil	HANAR	10250	2014-07-08
Canada	MEREP	10332	2014-10-17
Denmark	SIMOB	10341	2014-10-29
Finland	WARTH	10266	2014-07-26
France	VINET	10248	2014-07-04
Germany	TOMSP	10249	2014-07-05
Ireland	HUNGO	10298	2014-09-05
Italy	MAGAA	10275	2014-08-07
Mexico	CENTC	10259	2014-07-18
Norway	SANTG	10387	2014-12-18
Poland	WOLZA	10374	2014-12-05
Portugal	FURIB	10328	2014-10-14
Spain	ROMEY	10281	2014-08-14
Sweden	FOLKO	10264	2014-07-24
Switzerland	CHOPS	10254	2014-07-11
UK	BSBEV	10289	2014-08-26
USA	RATTC	10262	2014-07-22
Venezuela	HILAA	10257	2014-07-16

(21 row(s) affected)

Hint

Your first step will probably be to create a query like this:

Select

```
ShipCountry  
, CustomerID  
, OrderID  
, OrderDate = convert(date, OrderDate)
```

From orders

Order by

```
ShipCountry  
, OrderID
```

...which shows all the rows in the Order table, sorted first by Country and then by OrderID.

Hint

Your next step is to create a computed column that shows the row number for each order, partitioned appropriately.

There's a class of functions called Window functions or Ranking functions that you can use for this problem. Specifically, use the `Row_Number()` function, with the `Over` and `Partition` clause, to get the number, per country, of a particular order.

Hint

You'll have something like this:

```
Select
  ShipCountry
, CustomerID
, OrderID
, OrderDate = convert(date, OrderDate)
, RowNumberPerCountry =
    Row_Number()
      over (Partition by ShipCountry Order by ShipCountry, OrderID)
From Orders
```

Because of some limitations with Window functions, you can't directly filter the computed column created above. Use a CTE to solve the problem.

56. Customers with multiple orders in 5 day period

There are some customers for whom freight is a major expense when ordering from Northwind.

However, by batching up their orders, and making one larger order instead of multiple smaller orders in a short period of time, they could reduce their freight costs significantly.

Show those customers who have made more than 1 order in a 5 day period. The sales people will use this to help customers reduce their costs.

Note: There are more than one way of solving this kind of problem. For this problem, we will *not* be using Window functions.

Expected Result

CustomerID InitialOrderID InitialOrderDate NextOrderID NextOrderDate DaysBetween

ANTON	10677	2015-09-22	10682	2015-09-25	3
AROUT	10741	2015-11-14	10743	2015-11-17	3
BERGS	10278	2014-08-12	10280	2014-08-14	2
BERGS	10444	2015-02-12	10445	2015-02-13	1
BERGS	10866	2016-02-03	10875	2016-02-06	3
BONAP	10730	2015-11-05	10732	2015-11-06	1
BONAP	10871	2016-02-05	10876	2016-02-09	4
BONAP	10932	2016-03-06	10940	2016-03-11	5
BOTTM	10410	2015-01-10	10411	2015-01-10	0
BOTTM	10944	2016-03-12	10949	2016-03-13	1
BOTTM	10975	2016-03-25	10982	2016-03-27	2
BOTTM	11045	2016-04-23	11048	2016-04-24	1
BSBEV	10538	2015-05-15	10539	2015-05-16	1
BSBEV	10943	2016-03-11	10947	2016-03-13	2

... (skipping some rows)

SEVES	10800	2015-12-26	10804	2015-12-30	4
SUPRD	10841	2016-01-20	10846	2016-01-22	2
SUPRD	11035	2016-04-20	11038	2016-04-21	1
TRADH	10830	2016-01-13	10834	2016-01-15	2
TRADH	10834	2016-01-15	10839	2016-01-19	4
TRAIH	10574	2015-06-19	10577	2015-06-23	4
VICTE	10806	2015-12-31	10814	2016-01-05	5
VICTE	10843	2016-01-21	10850	2016-01-23	2
VINET	10737	2015-11-11	10739	2015-11-12	1
WARTH	10412	2015-01-13	10416	2015-01-16	3
WELLI	10803	2015-12-30	10809	2016-01-01	2
WELLI	10900	2016-02-20	10905	2016-02-24	4
WHITC	10693	2015-10-06	10696	2015-10-08	2
WILMK	10873	2016-02-06	10879	2016-02-10	4

(71 row(s) affected)

Hint

You can use a self-join, with 2 instances of the Orders table, joined by CustomerID. Good naming for the table aliases (table instances) are important for readability. Don't name them Order1 and Order2.

Hint

Select

```
InitialOrder.CustomerID  
,InitialOrderID = InitialOrder.OrderID  
,InitialOrderDate = InitialOrder.OrderDate  
,NextOrderID = NextOrder.OrderID  
,NextOrderDate = NextOrder.OrderDate
```

from Orders InitialOrder

```
join Orders NextOrder
```

```
on InitialOrder.CustomerID = NextOrder.CustomerID
```

Order by

```
InitialOrder.CustomerID  
,InitialOrder.OrderID
```

This is a good start. You will need to filter on additional fields in the join clause between InitialOrder and NextOrder, because as it is, this returns far too many orders. It has what's called a cartesian product between the 2 instances of the Orders table. This means that for the total number of orders for a particular customer in Orders, you'll have that number, squared, in the output.

Look at some of the OrderID and OrderDate values in InitialOrder and NextOrder. Some of them definitely disqualify a row based on our criteria.

Hint

Should the OrderID of the NextOrder ever be less than or equal to the OrderID of the NextOrder?

Hint

Based on the hint above, we added a where clause.

Select

```
InitialOrder.CustomerID
,InitialOrderID = InitialOrder.OrderID
,InitialOrderDate = InitialOrder.OrderDate
,NextOrderID = NextOrder.OrderID
,NextOrderDate = NextOrder.OrderDate
from Orders InitialOrder
  join Orders NextOrder
    on InitialOrder.CustomerID = NextOrder.CustomerID
where
  InitialOrder.OrderID < NextOrder.OrderID
Order by
  InitialOrder.CustomerID
,InitialOrder.OrderID
```

Adding this filter:

```
and InitialOrder.OrderID < NextOrder.OrderID
```

...has cut down the output a lot. However, we still need to filter for the 5 day period.

Create a new field called DaysBetween that calculates the number of days between the InitialOrder OrderDate and the NextOrder OrderDate. Use the DateDiff function.

Hint

You should now have a line like this:

```
DaysBetween = datediff(dd, InitialOrder.OrderDate, NextOrder.OrderDate)
```

Use this calculation in the Where clause to filter for 5 days or less between orders.

57. Customers with multiple orders in 5 day period, version 2

There's another way of solving the problem above, using Window functions. We would like to see the following results.

Expected Results

CustomerID OrderDate NextOrderDate DaysBetweenOrders

ANTON	2015-09-22	2015-09-25	3
AROUT	2015-11-14	2015-11-17	3
BERGS	2014-08-12	2014-08-14	2
BERGS	2015-02-12	2015-02-13	1
BERGS	2016-02-03	2016-02-06	3
BONAP	2015-11-05	2015-11-06	1
BONAP	2016-02-05	2016-02-09	4
BONAP	2016-03-06	2016-03-11	5
BOTTM	2015-01-10	2015-01-10	0
BOTTM	2016-03-12	2016-03-13	1
BOTTM	2016-03-25	2016-03-27	2
BOTTM	2016-04-23	2016-04-24	1

... (skipping some rows)

SAVEA	2016-03-27	2016-03-30	3
SAVEA	2016-04-17	2016-04-17	0
SEVES	2015-12-26	2015-12-30	4
SUPRD	2016-01-20	2016-01-22	2
SUPRD	2016-04-20	2016-04-21	1
TRADH	2016-01-13	2016-01-15	2
TRADH	2016-01-15	2016-01-19	4
TRAIH	2015-06-19	2015-06-23	4
VICTE	2015-12-31	2016-01-05	5
VICTE	2016-01-21	2016-01-23	2
VINET	2015-11-11	2015-11-12	1
WARTH	2015-01-13	2015-01-16	3
WELLI	2015-12-30	2016-01-01	2
WELLI	2016-02-20	2016-02-24	4
WHITC	2015-10-06	2015-10-08	2
WILMK	2016-02-06	2016-02-10	4

(69 row(s) affected)

Hint

The window function to use here is the Lead function.

Look up some examples of the Lead function online.

As a first step, write SQL using the Lead function to return results like the following. The NextOrderDate is a computed column that uses the Lead function.

CustomerID	OrderDate	NextOrderDate
------------	-----------	---------------

ALFKI	2015-08-25	2015-10-03
ALFKI	2015-10-03	2015-10-13
ALFKI	2015-10-13	2016-01-15
ALFKI	2016-01-15	2016-03-16
ALFKI	2016-03-16	2016-04-09
ALFKI	2016-04-09	NULL
ANATR	2014-09-18	2015-08-08
ANATR	2015-08-08	2015-11-28
ANATR	2015-11-28	2016-03-04
ANATR	2016-03-04	NULL

Hint

You should have something like this:

```
Select
CustomerID
,OrderDate = convert(date, OrderDate)
,NextOrderDate =
    convert(
        date
        ,Lead(OrderDate,1)
        OVER (Partition by CustomerID order by CustomerID, OrderDate)
    )
From Orders
Order by
    CustomerID
    ,OrderID
```

Now, take the output of this, and using a CTE and the DateDiff function, filter for rows which match our criteria.

Congratulations! You've completed the advanced problems

Any questions or feedback on the problems, hints, or answers? I'd like to hear from you. Please email me at feedback@SQLPracticeProblems.com.