

8. Orders shipping to France or Belgium

Looking at the Orders table, there's a field called ShipCountry. Write a query that shows the OrderID, CustomerID, and ShipCountry for the orders where the ShipCountry is either France or Belgium.

Expected Results

OrderID	CustomerID	ShipCountry
10248	VINET	France
10251	VICTE	France
10252	SUPRD	Belgium
10265	BLONP	France
10274	VINET	France
10295	VINET	France
10297	BLONP	France
10302	SUPRD	Belgium
10311	DUMON	France
10331	BONAP	France
10334	VICTE	France
10340	BONAP	France
10350	LAMAI	France
10358	LAMAI	France
10360	BLONP	France
10362	BONAP	France
10371	LAMAI	France

(Some rows were not included; the total should be 96)

Hint

In the where clause, instead of combining the filters with “and”, use “or”.

11. Showing only the Date with a DateTime field

In the output of the query above, showing the Employees in order of BirthDate, we see the time of the BirthDate field, which we don't want. Show only the date portion of the BirthDate field.

Expected Results

FirstName	LastName	Title	DateOnlyBirthDate
Margaret	Peacock	Sales Representative	1955-09-19
Nancy	Davolio	Sales Representative	1966-12-08
Andrew	Fuller	Vice President, Sales	1970-02-19
Steven	Buchanan	Sales Manager	1973-03-04
Laura	Callahan	Inside Sales Coordinator	1976-01-09
Robert	King	Sales Representative	1978-05-29
Michael	Suyama	Sales Representative	1981-07-02
Janet	Leverling	Sales Representative	1981-08-30
Anne	Dodsworth	Sales Representative	1984-01-27

Hint

Use the Convert function to convert the BirthDate column (originally a DateTime column) to a Date column.

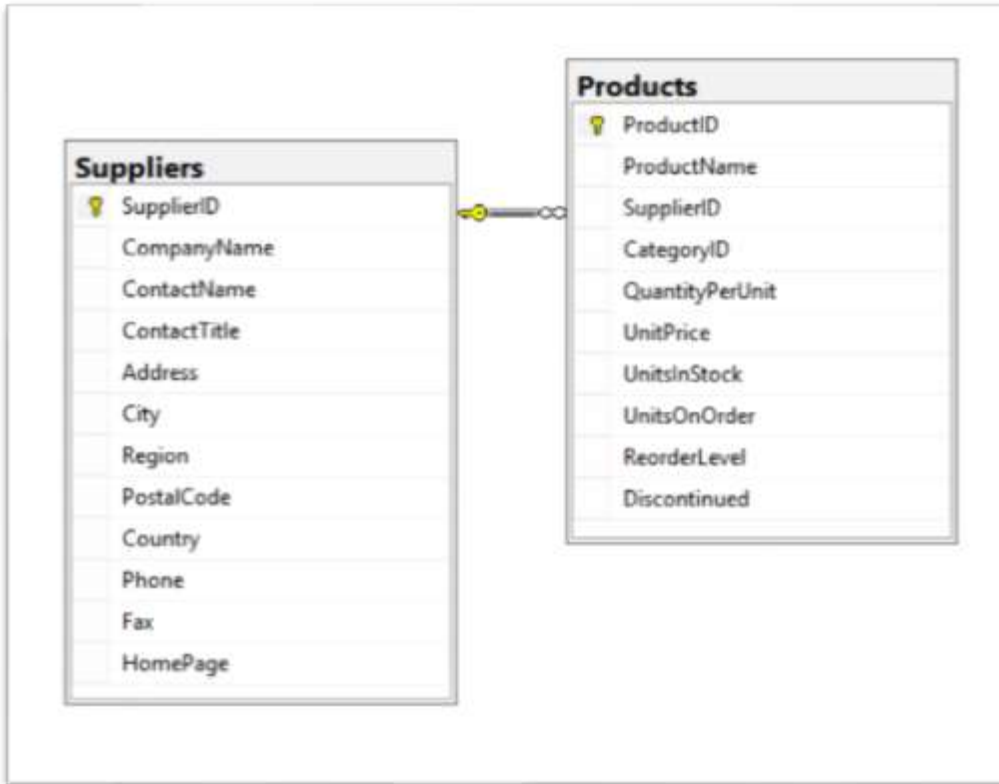
18. Products with associated supplier names

We'd like to show, for each product, the associated Supplier. Show the ProductID, ProductName, and the CompanyName of the Supplier.

Sort the result by ProductID.

This question will introduce what may be a new concept—the Join clause in SQL. The Join clause is used to join two or more relational database tables together in a logical way.

Here's a data model of the relationship between Products and Suppliers.



Expected Results

ProductID	ProductName	Supplier
1	Chai	Exotic Liquids
2	Chang	Exotic Liquids
3	Aniseed Syrup	Exotic Liquids
4	Chef Anton's Cajun Seasoning	New Orleans Cajun Delights
5	Chef Anton's Gumbo Mix	New Orleans Cajun Delights
6	Grandma's Boysenberry Spread	Grandma Kelly's Homestead
7	Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead
8	Northwoods Cranberry Sauce	Grandma Kelly's Homestead
9	Mishi Kobe Niku	Tokyo Traders
10	Ikura	Tokyo Traders
11	Queso Cabrales	Cooperativa de Quesos 'Las Cabras'
12	Queso Manchego La Pastora	Cooperativa de Quesos 'Las Cabras'
13	Konbu	Mayumi's
14	Tofu	Mayumi's
15	Genen Shouyu	Mayumi's
16	Pavlova	Pavlova, Ltd.
17	Alice Mutton	Pavlova, Ltd.
18	Carnarvon Tigers	Pavlova, Ltd.

19	Teatime Chocolate Biscuits	Specialty Biscuits, Ltd.
20	Sir Rodney's Marmalade	Specialty Biscuits, Ltd.
21	Sir Rodney's Scones	Specialty Biscuits, Ltd.

(Some rows were not included, the total should be 77)

Hint

Just as a reference, here's an example of what the syntax for the Join looks like, using different tables from the Northwind database. It will show all the products, with the associated CategoryName.

```
Select
    ProductID
    ,ProductName
    ,CategoryName
From Products
    Join Categories
    on Products.CategoryID = Categories.CategoryID
```

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
LINOD	LINO-Delicatesses	Nueva Esparta
GREAL	Great Lakes Food Market	OR
HUNGC	Hungry Coyote Import Store	OR
	(skipping some rows in the middle, the total rows returned should be 91)	
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

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 do to 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.

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

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?

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. So, if the total dollar amount of the customer's purchases in that year were between 0 to 1,000, they would be in the "Low" group. A customer with purchase from 1,000 to 5,000 would be in the "Medium" group, and so on.

A good starting point for this query is the answer from the problem "High-value customers—total orders". Also, we only want to show customers who have ordered in 2016.

Order the results by CustomerID.

Expected Result

CustomerID	Company Name	Total Order Amount	Customer Group
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
CONSH	Consolidated Holdings	931.50	Low
DRACD	Drachenblut Delikatessen	2809.61	Medium
DUMON	Du monde entier	860.10	Low
EASTC	Eastern Connection	9569.31	High
ERNSH	Ernst Handel	42598.90	Very High
FOLKO	Folk och få HB	15973.85	Very High
FRANK	Frankenversand	5587.00	High

(Some rows were not included, the total should be 81)

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.

ANSWERS

8. Orders shipping to France or Belgium

```
Select
    OrderID
    ,CustomerID
    ,ShipCountry
From Orders
where
    ShipCountry = 'France'
    or ShipCountry = 'Belgium'
```

Discussion

This is a very simple example, but in many situations you will have multiple where clauses, with combined “Or” and “And” sections.

In this situation, an alternative would have been to use the “In” operator. We’ll do that in a future problem.

11. Showing only the Date with a DateTime field

```
Select
    FirstName
    ,LastName
    ,Title
    ,DateOnlyBirthDate = convert(date, BirthDate)
From Employees
Order By Birthdate
```

Discussion

What we're using here is called a computed column, also sometimes called a calculated column. Anytime you're doing something besides just returning the column, as it is stored in the database, you're using a computed column. In this case, we're applying a function to convert the datatype returned.

Note that we've added a name, DateOnlyBirthDate, for our computed column. This is called an "alias".

```
DateOnlyBirthDate = convert(date, BirthDate)
```

If you don't actually specify the column alias, you get an empty column header, which is not unhelpful.

18. Products with associated supplier names

```
Select
    ProductID
    ,ProductName
    ,Supplier = CompanyName
From Products
Join Suppliers
    on Products.SupplierID = Suppliers.SupplierID
```

Discussion

Joins can range from the very simple, which we have here, to the very complex. You need to understand them thoroughly, as they're critical in writing anything but the simplest SQL.

One thing you'll see when reading SQL code is, instead of something like the answer above, something like this:

```
Select
    ProductID
    ,ProductName
    ,Supplier = CompanyName
From Products P      -- Aliased table
    Join Suppliers S  -- Aliased table
    on P.SupplierID = S.SupplierID
```

Notice that the Products table and Suppliers table is aliased, or renamed, with one letter aliases—P and S. If this is done, the P and S need to be used in the On clause as well.

I'm not a fan of this type of aliasing, although it's common. The only benefit is avoiding some typing, which is trivial. But the downside is that the code is harder to read and understand.

It's not so much a problem in small chunks of SQL like this one. However, in long, convoluted SQL, you'll find yourself wondering what the one-letter aliases mean, always needing to refer back to the From clause, and translate in your head.

The only time I use tables aliases is if the table name is extremely long. And then, I use table alias names that are understandable, just shortened.

24. Customer list by region

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

Discussion

Once we have the Case expression set up correctly, you just need to create an Order By clause for it, and add the additional fields for sorting (Region and CustomerID).

If we had wanted to include the sorting field in the output , you could write this:

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

You would not need to repeat the case statement in the Order By, you can just refer to the alias - RegionOrder.

Advanced Problems

32. High-value customers

```
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
Having Sum(Quantity * UnitPrice) > 10000
Order by TotalOrderAmount DESC
```

Discussion

If you tried putting this filter

```
and sum(Quantity * UnitPrice) >= 10000
```

... in the where clause, you got an error. Aggregate functions can only be used to filter (with some exceptions) in the Having clause, not the Where clause.

48. Customer grouping

```
;with Orders2016 as (
    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
)
Select
    CustomerID
    ,CompanyName
    ,TotalOrderAmount
    ,CustomerGroup =
        Case
            when TotalOrderAmount between 0 and 1000 then 'Low'
```

```

        when TotalOrderAmount between 1001 and 5000 then 'Medium'
        when TotalOrderAmount between 5001 and 10000 then 'High'
        when TotalOrderAmount > 10000 then 'Very High'
    End
from Orders2016
Order by CustomerID

```

Discussion

(Note—there's a small bug in the above SQL, which we'll review in the next problem.)

The CTE works well for this problem, but it's not strictly necessary. You could also use SQL like this:

```

Select
    Customers.CustomerID
    ,Customers.CompanyName
    ,TotalOrderAmount = SUM(Quantity * UnitPrice)
    ,CustomerGroup =
        Case
            when SUM(Quantity * UnitPrice) between 0 and 1000 then 'Low'
            when SUM(Quantity * UnitPrice) between 1001 and 5000 then 'Medium'
            when SUM(Quantity * UnitPrice) between 5001 and 10000 then 'High'
            when SUM(Quantity * UnitPrice) > 10000 then 'Very High'
        End
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

```

This gives the same result, but notice that the calculation for getting the TotalOrderAmount was repeated 5 times, including the 4 times in the Case statement.

It's best to avoid repeating calculations like this. The calculations will usually be quite complex and difficult to read, and you want to have them only in one place. In something simple, like Quantity * UnitPrice, it's not necessarily a problem. But most of the time, you should avoid repeating any calculations and code. An easy way to remember this is with the acronym DRY, which stands for “Don’t Repeat Yourself”. Here’s an article on the topic:

https://en.wikipedia.org/wiki/Don%27t_repeat_yourself