SQL Practice Problems

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

MySQL version

Sylvia Moestl Vasilik

Ordering Information:

Special discounts are available on quantity purchases by corporations, associations, and others.
For details, contact the publisher at info@SQLPracticeProblems.com.

Table of Contents

How to use this book

This book assumes that you have some basic background knowledge about relational databases. However, I've added some beginner level questions to gradually introduce the various parts of the SQL Select statement for those with less experience in SQL.

A note on the database used: it is *not* the standard Northwind database that was the sample included with several Microsoft database products. There have been many changes made to it, including additional tables, and modified data, to support the problems in this book. Do not try to use the standard Northwind sample database, many of the problems will not work.

Do you need to finish *all* the problems? Absolutely not. The introductory problems are fairly simple, so you may want to skip directly to the Intermediate Problems section. If you're not a beginner, but not sure where you should start, look at the problems and expected results in the Introductory Problems section, and make sure you completely understand the concepts and answers, then you could try skipping to the Intermediate problems.

If you're uncertain about how to start on a problem, the hints are designed to gradually walk you through how to approach the problem. Try hard to solve the problems first without the hints! It's best to start out using the "no hint" version of this document. Then, only open the main document (with hints) if you really need it. But if you're stuck, the hints will get you started.

If possible, don't look at the answers (in the back of the book), until you've solved the problem to the best of your ability on your own. You will learn and remember much more.

Should you search online for answers, examples, etc.? Absolutely. I expect you to do research online as you work through the problems. I do not include all the syntax in this book. In my day-to-day work as a data engineer, I would be lost without being able to do online research. Sometimes I search online for a reminder of a certain syntax, sometimes for examples of a particular type of code, and sometimes for approaches to specific problems. Learning to find answers online effectively can cut your problem-solving time dramatically.

Once you finish all the questions, you'll have some extremely useful skills in data analysis and data mining. The ability to use SQL is the foundation of data engineering, and (thankfully) does not change very frequently at all. This is useful because the SQL that you learned 10 years ago will probably be just as useful 10 years in the future! And it's a relatively rare skill.

I've interviewed many people who rated themselves a "9 or 10" on a 1-10 scale of SQL knowledge, who could not even do a simple Group by in a SQL statement.

Select statements aren't all there is to SQL, of course. There's also the syntax that lets you modify data (update, insert, delete), and create and modify database objects), as well as programming concepts such as stored procedures, and of course many other topics.

In this book, I'm only presenting problems involving retrieving data with Select statements, because that's by far the most common need, and it's also an area where it's very difficult for people to get solid practice with real life data problems, unless they're already working as a data engineer or programmer. And it's a critical first step for almost any other database topics.

Thank you for purchasing this book! Any feedback would be greatly appreciated. For any questions or issues, please send email to feedback@SQLPracticeProblems.com and I will be happy to respond.

Setup

This section will help you with the install of MySQL 8.0, and will walk you through the setup of the practice database.

MySQL 8.0 is a big leap in functionality from previous versions. It includes many very useful features that other DBMS systems (such as Microsoft SQL Server) have had for a long time, such as Window functions and CTEs (Common Table Expressions). The problems in this book are dependent on having MySQL 8.0 instead of previous versions.

Installing MySQL 8.0

To download and install MySQL 8.0, visit this website:

https://dev.mysql.com/downloads/

It's a straightforward setup. Make sure you install version 8.0 (MySQL 8.0 Generally Available (GA) Release) and not an earlier version. You can choose the operating system you wish to use, such as Microsoft Windows, macOS, Linux, etc.

Make sure you also install MySQL Workbench or a similar tool, in order to actually run SQL against the database. MySQL Workbench is included in the default installation, and that's the tool I use to walk through the setup of the practice database.

If you have any issues, the install documentation is online here:

https://dev.mysql.com/doc/refman/8.0/en/installing.html

Setting up the practice database

This video will walk you through the setup of the practice database used for the problems.

https://youtu.be/490vum3jyyU

It assumes that you have installed MySQL version 8.0.

Questions or problems with the setup?
Please email me at feedback@SQLPracticeProblems.com

Introductory Problems

1. Which shippers do we have?

We have a table called Shippers. Return all the fields from all the shippers

Expected Results

ShipperID	CompanyName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

2. Certain fields from Categories

In the Categories table, selecting all the fields using this SQL:

```
Select * from Categories;
```

…will return 4 columns. We only want to see two columns, CategoryName and Description.

Expected Results

CategoryName	Description
Beverages	Soft drinks, coffees, teas, beers, and ales
Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
Confections	Desserts, candies, and sweet breads
Dairy Products	Cheeses
Grains/Cereals	Breads, crackers, pasta, and cereal
Meat/Poultry	Prepared meats
Produce	Dried fruit and bean curd
Seafood	Seaweed and fish

3. Sales Representatives

We'd like to see just the FirstName, LastName, and HireDate of all the employees with the Title of Sales Representative. Write a SQL statement that returns only those employees.

Expected Results

FirstName	LastName	HireDate
Nancy	Davolio	2010-05-01 00:00:00.000
Janet	Leverling	2010-04-01 00:00:00.000
Margaret	Peacock	2011-05-03 00:00:00.000
Michael	Suyama	2011-10-17 00:00:00.000
Robert	King	2012-01-02 00:00:00.000
Anne	Dodsworth	2012-11-15 00:00:00.000

4. Sales Representatives in the United States

Now we'd like to see the same columns as above, but only for those employees that both have the title of Sales Representative, and also are in the United States.

Expected Results

FirstName	LastName	HireDate
Nancy	Davolio	2010-05-01 00:00:00.000
Janet	Leverling	2010-04-01 00:00:00.000
Margaret	Peacock	2011-05-03 00:00:00.000

5. Orders placed by specific EmployeeID

Show all the orders placed by a specific employee. The EmployeeID for this Employee (Steven Buchanan) is 5.

Expected Results

OrderID	OrderDate
10248	2014-07-04 08:00:00.000
10254	2014-07-11 02:00:00.000
10269	2014-07-31 00:00:00.000
10297	2014-09-04 21:00:00.000
10320	2014-10-03 12:00:00.000
10333	2014-10-18 18:00:00.000
10358	2014-11-20 05:00:00.000
10359	2014-11-21 14:00:00.000
10372	2014-12-04 10:00:00.000
10378	2014-12-10 00:00:00.000
10397	2014-12-27 17:00:00.000
10463	2015-03-04 13:00:00.000
10474	2015-03-13 16:00:00.000
10477	2015-03-17 02:00:00.000
10529	2015-05-07 01:00:00.000
10549	2015-05-27 03:00:00.000
10569	2015-06-16 15:00:00.000
10575	2015-06-20 22:00:00.000
10607	2015-07-22 09:00:00.000

(some result rows were not included, the total should be 42)

6. Suppliers and ContactTitles

In the Suppliers table, show the SupplierID, ContactName, and ContactTitle for those Suppliers whose ContactTitle is *not* Marketing Manager.

Expected Results

SupplierID	ContactName	ContactTitle
1	Charlotte Cooper	Purchasing Manager
2	Shelley Burke	Order Administrator
3	Regina Murphy	Sales Representative
5	Antonio del Valle Saavedra	Export Administrator

6	Mayumi Ohno	Marketing Representative
8	Peter Wilson	Sales Representative
9	Lars Peterson	Sales Agent
11	Petra Winkler	Sales Manager
12	Martin Bein	International Marketing Mgr.
13	Sven Petersen	Coordinator Foreign Markets
14	Elio Rossi	Sales Representative
16	Cheryl Saylor	Regional Account Rep.
17	Michael Björn	Sales Representative
18	Guylène Nodier	Sales Manager
19	Robb Merchant	Wholesale Account Agent
20	Chandra Leka	Owner
21	Niels Petersen	Sales Manager
22	Dirk Luchte	Accounting Manager
23	Anne Heikkonen	Product Manager
24	Wendy Mackenzie	Sales Representative
26	Giovanni Giudici	Order Administrator
27	Marie Delamare	Sales Manager
28	Eliane Noz	Sales Representative
29	Chantal Goulet	Accounting Manager

7. Products with “queso” in ProductName

In the products table, we’d like to see the ProductID and ProductName for those products where the ProductName includes the string “queso”.

Expected Results

ProductID	**ProductName**
11	Queso Cabrales
12	Queso Manchego La Pastora

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)

9. Orders shipping to any country in Latin America

Now, instead of just wanting to return all the orders from France or Belgium, we want to show all the orders from any Latin American country. But we don't have a list of Latin American countries in a table in the Northwind database. So, we're going to just use this list of Latin American countries that happen to be in the Orders table:

Brazil
Mexico

Argentina
Venezuela

It doesn't make sense to use multiple Or statements anymore. Use the In statement.

Note:

By default, MySQL only returns 100 rows in the result window. In order to show all of the rows, click the drop-down at the top of your query tab that says "Limit to 100 rows". Choose a higher number, or choose "Don't Limit".

Expected Results

OrderID	CustomerID	ShipCountry
10250	HANAR	Brazil
10253	HANAR	Brazil
10256	WELLI	Brazil
10257	HILAA	Venezuela
10259	CENTC	Mexico
10261	QUEDE	Brazil
10268	GROSR	Venezuela
10276	TORTU	Mexico
10283	LILAS	Venezuela
10287	RICAR	Brazil
10290	COMMI	Brazil
10291	QUEDE	Brazil
10292	TRADH	Brazil
10293	TORTU	Mexico
10296	LILAS	Venezuela
10299	RICAR	Brazil
10304	TORTU	Mexico
10308	ANATR	Mexico
10319	TORTU	Mexico
10322	PERIC	Mexico
10330	LILAS	Venezuela
10347	FAMIA	Brazil
10354	PERIC	Mexico
10357	LILAS	Venezuela

(Some rows were not included, the total should be 173)

10. Employees, in order of age

For all the employees in the Employees table, show the FirstName, LastName, Title, and BirthDate. Order the results by BirthDate, so we have the oldest employees first.

Expected Results

FirstName	LastName	Title	BirthDate
Margaret	Peacock	Sales Representative	1955-09-19 00:00:00.000
Nancy	Davolio	Sales Representative	1966-12-08 00:00:00.000
Andrew	Fuller	Vice President, Sales	1970-02-19 00:00:00.000
Steven	Buchanan	Sales Manager	1973-03-04 00:00:00.000
Laura	Callahan	Inside Sales Coordinator	1976-01-09 00:00:00.000
Robert	King	Sales Representative	1978-05-29 00:00:00.000
Michael	Suyama	Sales Representative	1981-07-02 00:00:00.000
Janet	Leverling	Sales Representative	1981-08-30 00:00:00.000
Anne	Dodsworth	Sales Representative	1984-01-27 00:00:00.000

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

12. Employees full name

Show the FirstName and LastName columns from the Employees table, and then create a new column called FullName, showing FirstName and LastName joined together in one column, with a space in-between.

Expected Results

FirstName	LastName	FullName
Nancy	Davolio	Nancy Davolio
Andrew	Fuller	Andrew Fuller
Janet	Leverling	Janet Leverling
Margaret	Peacock	Margaret Peacock
Steven	Buchanan	Steven Buchanan
Michael	Suyama	Michael Suyama
Robert	King	Robert King
Laura	Callahan	Laura Callahan
Anne	Dodsworth	Anne Dodsworth

13. OrderDetails amount per line item

In the OrderDetails table, we have the fields UnitPrice and Quantity. Create a new field, TotalPrice, that multiplies these two together. We'll ignore the Discount field for now.

In addition, show the OrderID, ProductID, UnitPrice, and Quantity. Order by OrderID and ProductID.

Expected Results

OrderID	ProductID	UnitPrice	Quantity	TotalPrice
10248	11	14.00	12	168.00
10248	42	9.80	10	98.00
10248	72	34.80	5	174.00
10249	14	18.60	9	167.40
10249	51	42.40	40	1696.00
10250	41	7.70	10	77.00
10250	51	42.40	35	1484.00
10250	65	16.80	15	252.00

10251	22	16.80	6	100.80
10251	57	15.60	15	234.00
10251	65	16.80	20	336.00
10252	20	64.80	40	2592.00
10252	33	2.00	25	50.00
10252	60	27.20	40	1088.00
10253	31	10.00	20	200.00
10253	39	14.40	42	604.80
10253	49	16.00	40	640.00
10254	24	3.60	15	54.00
10254	55	19.20	21	403.20
10254	74	8.00	21	168.00
10255	2	15.20	20	304.00
10255	16	13.90	35	486.50

(Some rows were not included, the total should be 2155)

14. How many customers?

How many customers do we have in the Customers table? Show one value only, and don't rely on getting the record count at the end of a resultset.

Expected Results

TotalCustomers
91

15. When was the first order?

Show the date of the first order ever made in the Orders table.

Expected Results

FirstOrder
2014-07-04 08:00:00.000

16. Countries where there are customers

Show a list of countries where the Northwind company has customers. Sort the list by the name of the country

Expected Results

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

17. Contact titles for customers

Show a list of all the different values in the Customers table for ContactTitles. Also include a count for each ContactTitle.

This is similar in concept to the previous question “Countries where there are customers”, except we now want a count for each ContactTitle.

Expected Results

ContactTitle	TotalContactTitle
Owner	17
Sales Representative	17
Marketing Manager	12
Sales Manager	11
Accounting Manager	10
Sales Associate	7
Marketing Assistant	6
Sales Agent	5
Assistant Sales Agent	2
Order Administrator	2
Assistant Sales Representative	1
Owner/Marketing Assistant	1

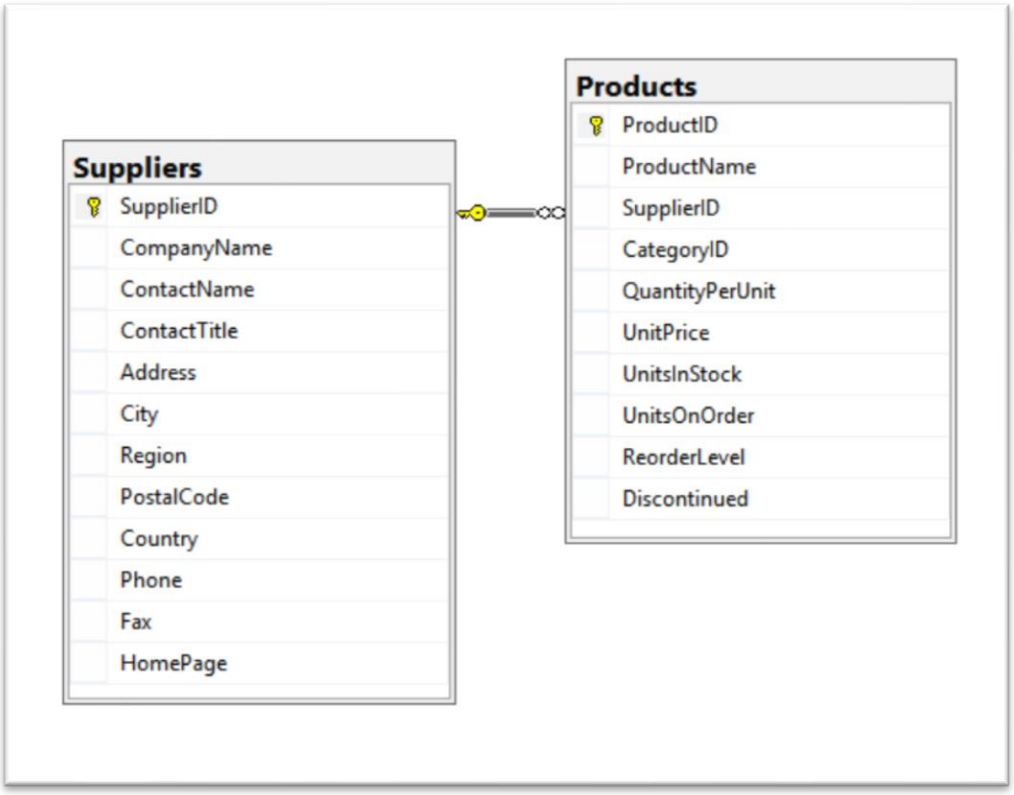
18. Products with associated supplier names

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)

19. Orders and the Shipper that was used

We'd like to show a list of the Orders that were made, including the Shipper that was used. Show the OrderID, OrderDate (date only), and CompanyName of the Shipper, and sort by OrderID.

In order to not show all the orders (there's more than 800), show only those rows with an OrderID of less than 10270.

Expected Results

OrderID	OrderDate	Shipper
10248	2014-07-04	Federal Shipping
10249	2014-07-05	Speedy Express
10250	2014-07-08	United Package
10251	2014-07-08	Speedy Express
10252	2014-07-09	United Package
10253	2014-07-10	United Package
10254	2014-07-11	United Package
10255	2014-07-12	Federal Shipping
10256	2014-07-15	United Package
10257	2014-07-16	Federal Shipping
10258	2014-07-17	Speedy Express
10259	2014-07-18	Federal Shipping
10260	2014-07-19	Speedy Express
10261	2014-07-19	United Package
10262	2014-07-22	Federal Shipping
10263	2014-07-23	Federal Shipping
10264	2014-07-24	Federal Shipping
10265	2014-07-25	Speedy Express
10266	2014-07-26	Federal Shipping
10267	2014-07-29	Speedy Express

10268	2014-07-30	Federal Shipping
10269	2014-07-31	Speedy Express

Congratulations! You've completed the introductory 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.

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

21. Total customers per country/city

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

Expected Results

Country	City	TotalCustomers
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
Austria	Salzburg	1
Venezuela	San Cristóbal	1
USA	San Francisco	1
USA	Seattle	1

(Some rows were not included, the total should be 69)

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 or equal to the ReorderLevel, Ignore the fields UnitsOnOrder and Discontinued.

Sort the results by ProductID.

Expected Results

ProductID	ProductName	UnitsInStock	ReorderLevel
2	Chang	17	25
3	Aniseed Syrup	13	25
5	Chef Anton's Gumbo Mix	0	0
11	Queso Cabrales	22	30
17	Alice Mutton	0	0
21	Sir Rodney's Scones	3	5
29	Thüringer Rostbratwurst	0	0
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
53	Perth Pasties	0	0
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

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

Product ID	Product Name	Units In Stock	Units On Order	Reorder Level	Discontinued
30	Nord-Ost Matjeshering	10	0	15	0
70	Outback Lager	15	10	30	0

24. Customer list by region

A salesperson for Northwind is going on a business trip to visit customers. He 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-Delicateses	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

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

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

27. High freight charges with between

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

```
Select
    ShipCountry
    ,avg(freight) as AverageFreight
From Orders
Where
    OrderDate between '2015-01-01' and '2015-12-31'
Group By ShipCountry
Order By AverageFreight desc
Limit 3;
```

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.

Find the OrderID that is causing the SQL statement above to be incorrect.

Expected Result

(no expected results this time—we're looking for one specific OrderID)

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

29. Employee/Order Detail report

We're doing inventory, and need to show Employee and Order Detail 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
5	Buchanan	10254	Guaraná Fantástica	15
5	Buchanan	10254	Pâté chinois	21
5	Buchanan	10254	Longlife Tofu	21

(Some rows were not included, the total should be 2155)

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

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

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

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 fä HB	15973.85
WHITC	White Clover Markets	15278.90

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

Customer ID	Company Name	Totals Without Discount	Totals With Discount
ERNSH	Ernst Handel	42598.90	41210.65
QUICK	QUICK-Stop	40526.99	37217.315
SAVEA	Save-a-lot Markets	42806.25	36310.11
HANAR	Hanari Carnes	24238.05	23821.2
RATTC	Rattlesnake Canyon Grocery	21725.60	21238.2705
HUNGO	Hungry Owl All-Night Grocers	22796.34	20402.120000000003
KOENE	Königlich Essen	20204.95	19582.774
WHITC	White Clover Markets	15278.90	15278.9
FOLKO	Folk och fä HB	15973.85	13644.067500000001
SUPRD	Suprêmes délices	11862.50	11644.6
BOTTM	Bottom-Dollar Markets	12227.40	11338.550000000001

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	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	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

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
10847	6
10657	6
10979	6
10294	5
10382	5
10406	5
10558	5
10670	5
10607	5

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 10 orders.

Expected Result

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

OrderID
10715
11033
10655
10296
10658
10642
10682
10942
10458
10800

38. Orders—accidental double-entry

Janet Leverling, one of the salespeople, has come to you with a request. She thinks that she accidentally entered a line item twice on an order, each time 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
10658
10990
11030

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

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;
```

41. Late orders

Some customers are complaining about their orders arriving late. Which orders are late? Sort the results by OrderID.

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
10660	2015-09-08	2015-10-06	2015-10-15

(Some rows were not included, your total should be 39)

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
2	Fuller	4
7	King	4
1	Davolio	3
6	Suyama	3

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. We want 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

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

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

46. Late orders vs. total orders—percentage

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

Expected Result

Employee ID	Last Name	All Orders	Late Orders	Percent Late Orders
1	Davolio	123	3	0.0244
2	Fuller	96	4	0.0417
3	Leverling	127	5	0.0394
4	Peacock	156	10	0.0641
5	Buchanan	42	0	0.0000
6	Suyama	67	3	0.0448
7	King	72	4	0.0556
8	Callahan	104	5	0.0481
9	Dodsworth	43	5	0.1163

47. Late orders vs. total orders—fix decimal

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

Employee ID	Last Name	All Orders	Late Orders	Percent Late Orders
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

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 entire	860.10	Low
EASTC	Eastern Connection	9569.31	High
ERNSH	Ernst Handel	42598.90	Very High
FOLKO	Folk och fä HB	15973.85	Very High
FRANK	Frankenversand	5587.00	High

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

49. Customer grouping—fix null

There's a problem 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

CustomerID	Company Name	Total Order Amount	Customer Group
MAISD	Maison Dewey	5000.20	High

(The total output is still 81 rows, but here we're only showing the row which had a null CustomerGroup value in the answer to the previous problem.)

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	0.4321
Low	20	0.2469
High	13	0.1605
Very High	13	0.1605

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

Customer ID	Company Name	Total Order Amount	Customer Group Name
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 entire	860.10	Low
EASTC	Eastern Connection	9569.31	High
ERNSH	Ernst Handel	42598.90	Very High
FOLKO	Folk och fä HB	15973.85	Very High
FRANK	Frankenversand	5587.00	High

(The expected results are the same as for the original problem, it's just that we're getting the answer differently. The total rows returned will still be 81, we're just showing a subset here.)

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

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

54. Countries with suppliers or customers, version 3

The output in the above practice problem 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

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, for each country, we want one row. That row should contain the earliest order for that country, with the associated ShipCountry, CustomerID, OrderID, and OrderDate.

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

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 freight 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

Customer ID	Initial Order ID	Initial Order Date	Next Order ID	Next Order Date	Days Between Orders
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
EASTC	11047	2016-04-24	11056	2016-04-28	4
ERNSH	10402	2015-01-02	10403	2015-01-03	1
ERNSH	10771	2015-12-10	10773	2015-12-11	1

(Some rows were not included, the total should be 71)

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

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

Expected Results

CustomerID	InitialOrderDate	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
BSBEV	2015-05-15	2015-05-16	1
BSBEV	2016-03-11	2016-03-13	2
EASTC	2016-04-24	2016-04-28	4
ERNSH	2015-01-02	2015-01-03	1
ERNSH	2015-12-10	2015-12-11	1
ERNSH	2015-12-11	2015-12-15	4
ERNSH	2016-03-23	2016-03-26	3
ERNSH	2016-04-08	2016-04-13	5
FOLKO	2016-03-26	2016-03-27	1
FOLKO	2016-03-27	2016-04-01	5
FOLKO	2016-04-01	2016-04-06	5
FRANK	2015-09-16	2015-09-19	3

(Some rows were not included, the total should be 69)

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.

ANSWERS

Introductory Problems

1. Which shippers do we have?

```
Select
    *
From Shippers;
```

Discussion

This is a basic select statement, returning all rows, just to get you warmed up.

Most of the time, a simple select statement like this is written all on one line, like this:

```
Select * From Shippers
```

But because we'll be getting more complex quickly, we'll start out with formatting it with separate lines for each clause, which we'll be doing in future questions.

2. Certain fields from Categories

```
Select
    CategoryName
    ,Description
from Categories;
```

Discussion

Instead of doing a “Select *”, we specify the column names, and only get those columns returned.

3. Sales Representatives

```
Select
    FirstName
    ,LastName
    ,HireDate
From Employees
Where
    Title = 'Sales Representative';
```

Discussion

This is a simple filter against a string datatype. When comparing a value to a string datatype, you need to enclose the value in single quotes.

What happens when you don’t? Try running the following:

```
Select
    FirstName
    ,LastName
    ,HireDate
From Employees
Where
    Title = Sales Representative
```

Notice that you get an error.

What if you compare against a number? Try the following:

```
Select
    FirstName
    ,LastName
    ,HireDate
From Employees
Where
    Title = 1;
```

In some database systems you'll get an error (because a number does not match the character datatype in the table) but MySQL does not return an error.

4. Sales Representatives in the United States

```
Select
    FirstName
    ,LastName
    ,HireDate
From Employees
Where
    Title = 'Sales Representative'
    and Country = 'USA';
```

Discussion

You can have as many filters in the where clause as you need. I usually indent all the filters, and put them on new lines, in order to make it easier to read.

5. Orders placed by specific EmployeeID

```
Select
    OrderID
    ,OrderDate
From Orders
```

```
Where
    EmployeeID = 5;
```

Discussion

This simple query filters for one value in the EmployeeID field, using the "=" comparison operator.

Here's another set of very commonly used comparison operators that you're probably familiar with from math class:

>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

6. Suppliers and ContactTitles

```
Select
    SupplierID
    ,ContactName
    ,ContactTitle
From Suppliers
Where
    ContactTitle <> 'Marketing Manager';
```

Discussion

Another way of expressing the Not is by using the following

```
!=
```

So, the below is equivalent to the answer with "<>".

```
Select
    CompanyName
    ,ContactName
    ,ContactTitle
From Suppliers
Where
```

```
    ContactTitle != 'Marketing Manager';
```

7. Products with “queso” in ProductName

```
Select
    ProductID
    ,ProductName
From Products
Where
    ProductName like '%queso%';
```

Discussion

The “Like” operator is always used with wildcards, such as the percent symbol (%), which substitutes for any number of characters.

Note that even though the search string used a lowercase “q” with the Like clause

```
ProductName like '%queso%'
```

the resulting rows both had an uppercase Q.

```
Queso Cabrales
Queso Manchego La Pastora
```

This is because by default MySQL is case insensitive, although it is also possible to have a case-sensitive installation, or to specify that a certain column is always case-sensitiv.

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.

9. Orders shipping to any country in Latin America

```
Select
    OrderID
    ,CustomerID
    ,ShipCountry
From Orders
where
    ShipCountry in
        (
        'Brazil'
        ,'Mexico'
        ,'Argentina'
        ,'Venezuela'
        );
```

Discussion

Using the In statement like this is a very common scenario when writing SQL. Whenever there’s more than just a few—say 2 or 3—values that we’re filtering for, I will generally put them on separate lines. It’s easier to read, understand, and modify.

Also, many times the list of items you’re filtering for will be coming from somewhere else—for instance, a spreadsheet—and will already be on separate lines.

10. Employees, in order of age

```
Select
    FirstName
    ,LastName
    ,Title
    ,BirthDate
From Employees
Order By Birthdate;
```

Discussion

This is a simple example of an Order By clause.

By default, MySQL sorts by ascending order (first to last). To sort in descending order (last to first), run the following, with the **desc** keyword:

```
Select
    FirstName
    ,LastName
    ,Title
    ,BirthDate
From Employees
Order By Birthdate desc;  -- keyword desc for last to first search
```

11. Showing only the Date with a DateTime field

```
Select
    FirstName
    ,LastName
    ,Title
    ,Date(BirthDate) as DateOnlyBirthDate
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 to a Date.

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

```
Date(BirthDate) as DateOnlyBirthDate
```

Run the following SQL, which does not have an alias name. What shows up as the column headers?

```
Select
    FirstName
    ,LastName
    ,Title
    ,Date(BirthDate)
From Employees
Order By Birthdate;
```

12. Employees full name

```
Select
    FirstName
    ,LastName
    ,Concat(FirstName, ' ',LastName) as FullName
From Employees;
```

Discussion

This is another example of a computed column. In this case, instead of applying a function to a field, we're concatenating two fields.

13. OrderDetails amount per line item

```
Select
    OrderID
    ,ProductID
    ,UnitPrice
    ,Quantity
    , UnitPrice * Quantity as TotalPrice
From OrderDetails
Order by
    OrderID
    ,ProductID;
```

Discussion

Here we have another example of a computed column, this time using the arithmetic operator "*"for multiplication.

14. How many customers?

```
Select
    count(*) as TotalCustomers
from Customers;
```

Discussion

Aggregates functions and grouping are very important when retrieving data. In almost all cases, when doing data analysis, you'll be using multiple groupings and aggregates.

15. When was the first order?

```
Select
    min(OrderDate) as FirstOrder
From Orders;
```

Discussion

For the aggregate function Count, you don't need to specify a column name—just count(*) will work.

However, for other aggregate functions such as Min, Avg, Sum, etc., you will need to specify a column name since you're not just counting all rows.

16. Countries where there are customers

```
Select
    Country
From Customers
Group by
    Country
Order by
    Country;
```

Discussion

The Group By clause is a cornerstone of SQL. With most data analysis of any complexity at all, you'll be using multiple Group By clauses, so they're important to understand.

Another way of getting the same results is to use the Distinct keyword, as below:

```
Select distinct
    Country
From Customers
Order by
    Country;
```

It looks simpler, and works well for queries that are very straightforward. But in everyday use, you'll use Group By more often than of Distinct, because you'll need to use additional aggregate functions such as Count, and Sum.

17. Contact titles for customers

```
Select
    ContactTitle
    , count(*) as TotalContactTitle
From Customers
Group by
    ContactTitle
Order by
    count(*) desc;
```

Discussion

This particular construction, with a grouping, and then a count of the total in each group, is very common both on its own, and as a part of other queries.

18. Products with associated supplier names

```
Select
    ProductID
    ,ProductName
    ,CompanyName as Supplier
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
    ,CompanyName as Supplier
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.

19. Orders and the Shipper that was used

```
Select
    OrderID
    ,Date(OrderDate) as OrderDate
    ,CompanyName as Shipper
From Orders
    join Shippers
        on Shippers.ShipperID = Orders.ShipVia
Where
    OrderID < 10270
Order by
    OrderID;
```

Discussion

As the SQL you write gets more complex, it's even more important to format it so that it's easily readable.

20. Categories, and the total products in each category

```
Select
    CategoryName
    ,count(*) as TotalProducts
From Products
    Join Categories
        on Products.CategoryID = Categories.CategoryID
Group by
    CategoryName
Order by
    count(*) desc;
```

Discussion

We're expanding our knowledge of grouping here with a very common scenario—grouping across two joined tables. In this case, the tables have what's called a parent-child relationship. The parent table is Categories, and the child table is Products.

21. Total customers per country/city

```
Select
    Country
    ,City
    ,Count(*) as TotalCustomers
From Customers
Group by
    Country
    ,City
Order by
    count(*) desc;
```

Discussion

Note that once you have a Group by clause in a SQL statement, every field that appears in the Select statement needs to either appear in the Group by clause, or needs to have some kind of aggregate function applied to it.

If you don't do this, you will get incorrect results. For instance, try running the following SQL, with the City commented out in the Group by clause:

```
Select
    Country
    ,City
    ,Count(*)  as TotalCustomers
From Customers
Group by
    Country
    -- ,City
Order by
    count(*) desc;
```

Most database systems would give you an error, but in MySQL, it will just give you the incorrect results.

SQL Server will give the following error message:

```
Msg 8120, Level 16, State 1, Line 3
Column 'Customers.City' is invalid in the select list because it is not contained in
either an aggregate function or the GROUP BY clause.
```

22. Products that need reordering

```
Select
    ProductID
    ,ProductName
    ,UnitsInStock
    ,ReorderLevel
From Products
Where
    UnitsInStock <= ReorderLevel
Order by ProductID;
```

Discussion

This is a straightforward query on one table. Instead of using a specific string or numeric value to filter, we're using another field.

23. Products that need reordering, continued

```
Select
    ProductID
    ,ProductName
    ,UnitsInStock
    ,UnitsOnOrder
    ,ReorderLevel
    ,Discontinued
From Products
Where
    UnitsInStock + UnitsOnOrder <= ReorderLevel
    and Discontinued = 0
Order by ProductID;
```

Discussion

Instead of writing

```
and Discontinued = 0
```

…you can also write the following if you find it easier to read:

```
and Discontinued = 'false'
```

MySQL will automatically convert the 'false' to 0.

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
    ,Case
        when Region is null then 1
        else 0
    End
        as RegionOrder
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.

25. High freight charges

```
Select
    ShipCountry
    , Avg(freight) as AverageFreight
From Orders
Group By ShipCountry
Order By AverageFreight desc
Limit 3
```

Discussion

In MySQL, the keyword Limit is the easiest and most commonly used method of showing only a certain number of records.

SQL Server uses the keyword Top to do the same thing.

26. High freight charges—2015

```
Select
    ShipCountry
    ,avg(freight) as AverageFreight
From Orders
Where
    OrderDate >= '2015-01-01'
    and OrderDate  < '2016-01-01'
Group By ShipCountry
Order By AverageFreight desc
Limit 3;
```

Discussion

You could also use the following:

```
Select
    ShipCountry
    ,avg(freight) as AverageFreight
From Orders
Where
```

```
    year(OrderDate) = 2015          -- using Year function
Group By ShipCountry
Order By AverageFreight desc
Limit 3;
```

This looks straightforward and is easy to read. However, when you put a function such as Year on the OrderDate field, any indexes that exist won't be used anymore, so it will potentially be much slower.

Also, you can only filter for specific calendar years instead of a date range, so it's not very flexible.

27. High freight charges with between

The OrderID that's causing the different results is 10806.

Discussion

There's an order made on December 31, 2015 with a really high value in the Freight field. This would have skewed the results, and put France in third place for highest freight charges, but only if it were included in the Where clause.

This SQL would have worked fine if OrderDate were a Date field, instead of DateTime.

```
    OrderDate between '2015-01-01' and '2015-12-31'
```

However, since it's a DateTime field, it gives an incorrect answer because it's not taking into account records where the OrderDate is *during the day* on December 31, 2015.

Note that for a DateTime field, the value

2015-12-31'

is equivalent *only* to

2015-12-31 00:00:00.000

…and *not* to values that have a time component.

28. High freight charges—last year

```
Select
    ShipCountry
    ,Avg(freight) as AverageFreight
From Orders
Where
    OrderDate >=
        DATE_ADD((Select max(OrderDate) from Orders) , INTERVAL -1 year)
Group by ShipCountry
Order by AverageFreight desc
Limit 3;
```

Discussion

Using SQL like this, that can generate a dynamic date range, is critical for most data analysis work. Most reports and queries will need to be flexible, without hard-coded date values.

29. Employee/Order Detail report

```
Select
    Employees.EmployeeID
    ,Employees.LastName
    ,Orders.OrderID
    ,Products.ProductName
    ,OrderDetails.Quantity
From Employees
    join Orders
        on Orders.EmployeeID = Employees.EmployeeID
    join OrderDetails
        on Orders.OrderID = OrderDetails.OrderID
    join Products
        on Products.ProductID = OrderDetails.ProductID
Order by
    Orders.OrderID
    ,Products.ProductID;
```

Discussion

This problem is more practice with basic joins and multiple tables.

You can replace Join with Inner Join, but mostly people just use Join.

30. Customers with no orders

```
Select
    Customers.CustomerID as Customers_CustomerID
    ,Orders.CustomerID as Orders_CustomerID
From Customers
    left join Orders
        on Orders.CustomerID = Customers.CustomerID
Where
    Orders.CustomerID is null;
```

Discussion

There are many ways of getting the same results. The main options are the Left Join with Is Null, Not In, and Not Exists.

Above, we used the Left Join option. When performance is equivalent, I prefer the Not In method, shown below.

```
Select CustomerID
From Customers
Where
    CustomerID not in (select CustomerID from Orders);
```

I believe this is the easiest to read and understand.

Another option is to use Not Exists. This requires a correlated subquery.

```
Select CustomerID
From Customers
Where Not Exists
    (
    Select CustomerID
    from Orders
    where
        Orders.CustomerID = Customers.CustomerID
```

```
    );
```

Performance for the different options can be affected by whether or not the fields are indexed or nullable. For additional reading on the pros and cons of each method, do some research online.

31. Customers with no orders for EmployeeID 4

```
Select
    Customers.CustomerID
    ,Orders.CustomerID
From Customers
    left join Orders
        on Orders.CustomerID = Customers.CustomerID
        and Orders.EmployeeID = 4
Where
    Orders.CustomerID is null;
```

Discussion

Because the filters in the Where clause are applied *after* the results of the Join, we need the EmployeeID = 4 filter in the Join clause, instead of the Where clause.

Run the below query and review the results. It should give you a better sense of how the left join with "is null" works. Note that the Where clause is commented out.

```
Select
    Customers.CustomerID
    ,Orders.CustomerID
    ,Orders.EmployeeID
From Customers
    left join Orders
        on Orders.CustomerID = Customers.CustomerID
        and Orders.EmployeeID = 4
-- Where
--     Orders.CustomerID is null;
```

The most common way to solve this kind of problem is as above, with a left join. However, here are some alternatives using Not In and Not Exists.

```
Select CustomerID
From Customers
Where
    CustomerID not in (select CustomerID from Orders where EmployeeID = 4);

Select CustomerID
From Customers
Where Not Exists
    (
    Select CustomerID
    from Orders
        where Orders.CustomerID = Customers.CustomerID
        and EmployeeID = 4
    );
```

Advanced Problems

32. High-value customers

```
Select
    Customers.CustomerID
    ,Customers.CompanyName
    ,Orders.OrderID
    ,SUM(Quantity * UnitPrice) as TotalOrderAmount
From Customers
    Join Orders
        on Orders.CustomerID = Customers.CustomerID
    Join OrderDetails
        on Orders.OrderID = OrderDetails.OrderID
Where
    OrderDate >= '2016-01-01'
    and OrderDate  < '2017-01-01'
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 this error:

```
Error Code: 1111. Invalid use of group function
```

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

33. High-value customers—total orders

```
Select
    Customers.CustomerID
    ,Customers.CompanyName
    -- ,Orders.OrderID
    ,SUM(Quantity * UnitPrice) as TotalOrderAmount
From Customers
    Join Orders
        on Orders.CustomerID = Customers.CustomerID
    Join OrderDetails
        on Orders.OrderID = OrderDetails.OrderID
Where
    OrderDate >= '2016-01-01'
    and OrderDate  < '2017-01-01'
Group by
    Customers.CustomerID
    ,Customers.CompanyName
    -- ,Orders.Orderid
Having sum(Quantity * UnitPrice) > 15000
Order by TotalOrderAmount desc;
```

Discussion

All that was necessary here was to comment out references in the Select clause and the Group By clause to OrderID. By doing that, we're grouping at the Customer level, and not at the Order level.

34. High-value customers—with discount

```
Select
    Customers.CustomerID
    ,Customers.CompanyName
    ,SUM(Quantity * UnitPrice) as TotalsWithoutDiscount
    ,SUM(Quantity * UnitPrice * (1- Discount)) as TotalsWithDiscount
From Customers
    Join Orders
        on Orders.CustomerID = Customers.CustomerID
    Join OrderDetails
        on Orders.OrderID = OrderDetails.OrderID
Where
    OrderDate >= '2016-01-01'
    and OrderDate  < '2017-01-01'
Group by
    Customers.CustomerID
    ,Customers.CompanyName
Having TotalsWithDiscount  > 10000
Order by TotalsWithDiscount DESC;
```

Discussion

Note that you need to use the new calculation for order totals with discounts in the Select clause, the Having clause, and also the Order by clause.

In MySQL, you can just re-use the field via the column alias in the Having clause, like this:

```
Having TotalsWithDiscount  > 10000
```

However, in other database systems (such as SQL Server), you must repeat the calculation in the Having clause, and can't just re-use it.

35. Month-end orders

```
Select
    EmployeeID
    ,OrderID
    ,OrderDate
From Orders
Where OrderDate = Last_Day(OrderDate )
Order by
    EmployeeID
    ,OrderID;
```

Discussion

Very frequently the end of the month will be needed in queries and reports. The Last_Day function is very useful in these cases.

Here's a bonus question for you. There are actually 2 more orders where the OrderDate is the end of the month. They are OrderID 10806 and 10807.

Why are they not showing up in the results? Can you modify the SQL to show these 2 orders as well?

36. Orders with many line items

```
Select
    OrderID
    ,count(*) as TotalOrderDetails
From OrderDetails
Group By OrderID
Order By Count(*) desc
Limit 10;
```

Discussion

Try limiting the rows returned to 50 instead of 10. What happens?

You'll notice that there are many orders that have 5 total line items. But since we originally only showed the top 10, SQL Server eliminated most of the orders with 5 line items.

37. Orders—random assortment

```
Select
    OrderID
From Orders
Order By Rand()
Limit 10;
```

Discussion

Using the Rand() function, and then ordering by it and limiting the result, is a reasonable way of getting a random set of records from a table. However, it may be slow if the table is large.

There are other techniques that can be used if performance is an issue. You can do research online to find some better, more complex techniques.

38. Orders—accidental double-entry

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

Discussion

This SQL shows orders that have at least 1 order detail with a quantity of 60 or more (the Where clause), *and* the quantity is duplicated within the order (the Group by and Having clause). This occurs because we're grouping on *both* OrderID and Quantity.

39. Orders—accidental double-entry details

```
with PotentialDuplicates as (
    Select
        OrderID
    From OrderDetails
    Where Quantity >= 60
    Group By OrderID, Quantity
    Having Count(*) > 1
    )
Select
    OrderID
    ,ProductID
    ,UnitPrice
    ,Quantity
    ,Discount
From OrderDetails
Where
    OrderID in (Select OrderID from PotentialDuplicates)
Order by
    OrderID
    ,Quantity;
```

Discussion

There are quite a few different ways of getting the same results for this problem. Based on years of painful troubleshooting caused by poorly-written, tangled SQL, I suggest that writing easily understandable, straightforward code is one of the most important things to strive for. Using a well thought-out CTE is one way of doing this.

In the next problem, we'll look at another way of getting the same result.

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

```
Select
    OrderDetails.OrderID
    ,ProductID
    ,UnitPrice
    ,Quantity
    ,Discount
From OrderDetails
```

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

Discussion

Note the Distinct keyword, added after the Select in the derived table. This gives us only distinct rows in the output, which avoids the problem with duplicate OrderIDs.

41. Late orders

```
Select
    OrderID
    ,Date(OrderDate) as OrderDate
    ,Date(RequiredDate) as RequiredDate
    ,Date(ShippedDate) as ShippedDate
From Orders
Where
    RequiredDate <= ShippedDate
Order by
    OrderID;
```

Discussion

This is a straight-forward query that we'll use as a base for future problems.

42. Late orders—which employees?

```
Select
    Employees.EmployeeID
    ,LastName
    ,Count(*) as TotalLateOrders
From Orders
    Join Employees
        on Employees.EmployeeID = Orders.EmployeeID
Where
    RequiredDate <= ShippedDate
Group By
    Employees.EmployeeID
    ,Employees.LastName
Order by TotalLateOrders desc;
```

Discussion

In many database systems (such as SQL Server), when a query has a Group By clause, each column in the Select statement needs to either also be grouped by, or have an aggregate (sum, count) applied to it.

MySQL does not require this. A column in a query that has a Group By clause does *not* have to be in the Group by clause, or have an aggregate.

In SQL Server, you would get the following error:

```
Msg 8120, Level 16, State 1, Line 3
Column 'Employees.LastName' is invalid in the select list because it is not contained
in either an aggregate function or the GROUP BY clause.
```

In MySQL, you would not get an error. However, you may not get the results you're looking for, since it will just by default apply a "Min" to the field.

43. Late orders vs. total orders

```
With LateOrders as (
    Select
        EmployeeID
        ,Count(*) as TotalOrders
```

```
    From Orders
    Where
        RequiredDate <= ShippedDate
    Group By
        EmployeeID
)
, AllOrders as (
    Select
        EmployeeID
        ,Count(*) as TotalOrders
    From Orders
    Group By
        EmployeeID
)
Select
    Employees.EmployeeID
    ,LastName
    ,AllOrders.TotalOrders as AllOrders
    ,LateOrders.TotalOrders as LateOrders
From Employees
    Join AllOrders
        on AllOrders.EmployeeID = Employees.EmployeeID
    Join LateOrders
        on LateOrders.EmployeeID = Employees.EmployeeID
Order by Employees.EmployeeID;
```

Discussion

The above query is almost correct, but if you're paying careful attention, you'll realize it has a slight problem. We'll learn more in the next problem.

44. Late orders vs. total orders—missing employee

```
With LateOrders as (
    Select
        EmployeeID
        ,Count(*) as TotalOrders
    From Orders
    Where
        RequiredDate <= ShippedDate
```

```
    Group By
        EmployeeID
)
, AllOrders as (
    Select
        EmployeeID
        ,Count(*) as TotalOrders
    From Orders
    Group By
        EmployeeID
)
Select
    Employees.EmployeeID
    ,LastName
    ,AllOrders.TotalOrders as AllOrders
    ,LateOrders.TotalOrders as LateOrders
From Employees
    Join AllOrders
        on AllOrders.EmployeeID = Employees.EmployeeID
    Left Join LateOrders
        on LateOrders.EmployeeID = Employees.EmployeeID;
```

Discussion

The above SQL shows all employees who have made orders, even if they have no late orders.

What would we need to do if we wanted to show all employees, even if they have not been the sales person (EmployeeID) for an Order?

45. Late orders vs. total orders—fix null

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

```
    Select
        EmployeeID
        ,Count(*) as TotalOrders
    From Orders
    Group By
        EmployeeID
)
Select
    Employees.EmployeeID
    ,LastName
    ,AllOrders.TotalOrders as AllOrders
    ,IfNull(LateOrders.TotalOrders, 0) as LateOrders
From Employees
    Join AllOrders
        on AllOrders.EmployeeID = Employees.EmployeeID
    Left Join LateOrders
        on LateOrders.EmployeeID = Employees.EmployeeID;
```

Discussion

Using a straightforward IfNull on LateOrder is the best way to solve this problem.

Another way to write it would be using a Case statement

```
    ,Case
        When LateOrders.TotalOrders is null Then 0
        Else LateOrders.TotalOrders
    End as LateOrders
```

But when you don't need any other logic besides a test for null, IfNull is the way to go.

46. Late orders vs. total orders—percentage

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

```
        EmployeeID
)
, AllOrders as (
    Select
        EmployeeID
        ,Count(*) as TotalOrders
    From Orders
    Group By
        EmployeeID
)
Select
    Employees.EmployeeID
    ,LastName
    ,AllOrders.TotalOrders as AllOrders
    ,IfNull(LateOrders.TotalOrders, 0) as LateOrders
    ,IfNull(LateOrders.TotalOrders, 0)/ AllOrders.TotalOrders
        as PercentLateOrders
From Employees
    Join AllOrders
        on AllOrders.EmployeeID = Employees.EmployeeID
    Left Join LateOrders
        on LateOrders.EmployeeID = Employees.EmployeeID;
```

Discussion

In MySQL, you can just divide LateOrders by AllOrders without a problem.

In some other database systems (such as SQL Server), you would need to explicitly convert one of the fields to a decimal datatype, or implicitly convert them by multiplying by 1.0 in order to get a decimal output.

47. Late orders vs. total orders—fix decimal

```
With LateOrders as (
    Select
        EmployeeID
        ,Count(*) as TotalOrders
    From Orders
    Where
        RequiredDate <= ShippedDate
```

```
    Group By
        EmployeeID
)
, AllOrders as (
    Select
        EmployeeID
        ,Count(*) as TotalOrders
    From Orders
    Group By
        EmployeeID
)
Select
    Employees.EmployeeID
    ,LastName
    ,AllOrders.TotalOrders as AllOrders
    ,IfNull(LateOrders.TotalOrders, 0) as LateOrders
    ,Cast(
        IfNull(LateOrders.TotalOrders, 0)/ AllOrders.TotalOrders
        as Decimal (4,2)
        )
        as PercentLateOrders
From Employees
    Join AllOrders
        on AllOrders.EmployeeID = Employees.EmployeeID
    Left Join LateOrders
        on LateOrders.EmployeeID = Employees.EmployeeID;
```

Discussion

Rounding, truncating, and converting data types can get complicated, and there are many ways that you could get unexpected results. Always check your results carefully, and know whether you want rounding, or truncation.

Frequently, when creating this kind of query, you'll put the output into a tool like Excel, and do any additional formatting such as setting the decimal precision there. However, it's good to at least know how to do it in SQL.

You may have noticed that I added multiple lines in the calculation to make it easier to read. This isn't necessary—you could put everything on one line—but it's good programming practice, and easier to read and troubleshoot.

48. Customer grouping

```
with Orders2016 as (
    Select
        Customers.CustomerID
        ,Customers.CompanyName
        ,SUM(Quantity * UnitPrice) as TotalOrderAmount
    From Customers
        Join Orders
            on Orders.CustomerID = Customers.CustomerID
        Join OrderDetails
            on Orders.OrderID = OrderDetails.OrderID
    Where
        OrderDate >= '2016-01-01'
        and OrderDate  < '2017-01-01'
    Group by
        Customers.CustomerID
        ,Customers.CompanyName
)
Select
    CustomerID
    ,CompanyName
    ,TotalOrderAmount
    ,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
        as CustomerGroup
from Orders2016
Order by CustomerID;
```

Discussion

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

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

```
Select
    Customers.CustomerID
    ,Customers.CompanyName
    ,SUM(Quantity * UnitPrice) as TotalOrderAmount
    ,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 as CustomerGroup
From Customers
    Join Orders
        on Orders.CustomerID = Customers.CustomerID
    Join OrderDetails
        on Orders.OrderID = OrderDetails.OrderID
Where
    OrderDate >= '2016-01-01'
    and OrderDate  < '2017-01-01'
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 far better 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

49. Customer grouping—fix null

```
with Orders2016 as (
    Select
        Customers.CustomerID
        ,Customers.CompanyName
        ,SUM(Quantity * UnitPrice) as TotalOrderAmount
    From Customers
        Join Orders
            on Orders.CustomerID = Customers.CustomerID
        Join OrderDetails
            on Orders.OrderID = OrderDetails.OrderID
    Where
        OrderDate >= '2016-01-01'
        and OrderDate  < '2017-01-01'
    Group by
        Customers.CustomerID
        ,Customers.CompanyName
)
```

```
Select
    CustomerID
    ,CompanyName
    ,TotalOrderAmount
    ,Case
        when TotalOrderAmount >= 0 and TotalOrderAmount  < 1000
            then 'Low'
        when TotalOrderAmount >= 1000 and TotalOrderAmount  < 5000
            then 'Medium'
        when TotalOrderAmount >= 5000 and TotalOrderAmount  <10000
            then 'High'
        when TotalOrderAmount >= 10000
            then 'Very High'
    End as CustomerGroup
from Orders2016
Order by CustomerID;
```

Discussion

As you've been seeing in the above problems, knowing the data types you're working with and understanding the differences between them is important to get the right results. Using "between" would have been fine for integer values, but not for decimal.

50. Customer grouping with percentage

```
with Orders2016 as (
    Select
        Customers.CustomerID
        ,Customers.CompanyName
        ,SUM(Quantity * UnitPrice) as TotalOrderAmount
    From Customers
        join Orders
            on Orders.CustomerID = Customers.CustomerID
        join OrderDetails
            on Orders.OrderID = OrderDetails.OrderID
    Where
        OrderDate >= '2016-01-01'
        and OrderDate  < '2017-01-01'
    Group By
        Customers.CustomerID
        ,Customers.CompanyName
```

```
)
,CustomerGrouping as (
    Select
        CustomerID
        ,CompanyName
        ,TotalOrderAmount
        ,Case
            when TotalOrderAmount >= 0 and TotalOrderAmount  < 1000
                then 'Low'
            when TotalOrderAmount >= 1000 and TotalOrderAmount  < 5000
                then 'Medium'
            when TotalOrderAmount >= 5000 and TotalOrderAmount  <10000
                then 'High'
            when TotalOrderAmount >= 10000
                then 'Very High'
        End
            as CustomerGroup
    from Orders2016
)
Select
    CustomerGroup
    ,Count(*) as TotalInGroup
    ,Count(*)/(select count(*) from CustomerGrouping)
        as PercentageInGroup
from CustomerGrouping
group by CustomerGroup
order by TotalInGroup  desc;
```

Discussion

In the answer we added an intermediate CTE called CustomerGrouping. CustomerGrouping is referenced twice—once to get the total number of customers in the specific group, and once to get the total, as the denominator for the percentage.

51. Customer grouping—flexible

```
with Orders2016 as (
    Select
        Customers.CustomerID
```

```
        ,Customers.CompanyName
        ,SUM(Quantity * UnitPrice) as TotalOrderAmount
    From Customers
        Join Orders
            on Orders.CustomerID = Customers.CustomerID
        Join OrderDetails
            on Orders.OrderID = OrderDetails.OrderID
    Where
        OrderDate >= '2016-01-01'
        and OrderDate  < '2017-01-01'
    Group by
        Customers.CustomerID
        ,Customers.CompanyName
)
Select
    CustomerID
    ,CompanyName
    ,TotalOrderAmount
    ,CustomerGroupName
from Orders2016
    Join CustomerGroupThresholds
        on Orders2016.TotalOrderAmount between
            CustomerGroupThresholds.RangeBottom
            and CustomerGroupThresholds.RangeTop
Order by CustomerID;
```

Discussion

Note that this gives the same results as the original problem. However, instead of using hard-coded values in the Case statement to define the boundaries of the CustomerGroups, you have them in a table.

The benefit of this is that you can just reference the table, instead of duplicating a complex Case statement, in every query where you want to group customers the same way.

Also, take a look at the values in CustomerGroupThresholds.

```
select * From CustomerGroupThresholds
```

Note that there's no overlap between the rows when you look at RangeBottom and RangeTop, so you can use “between”. You need to be careful with this, and pay attention to how many digits there are to the right of the decimal, to make sure there are no potential values that could be missed.

52. Countries with suppliers or customers

```
Select Country From Customers
Union
Select Country From Suppliers
Order by Country;
```

Discussion

There are 2 ways of using the Union statement. One is a simple Union as in the answer here. Using a simple Union statement eliminates all the duplicates in the resultset.

You can also use Union All. Try it and take a look at the resultset:

```
Select distinct Country From Customers
Union All
Select distinct Country From Suppliers
Order by Country;
```

Notice that within the individual SQL statements, I've put a Distinct. However, there are still duplicates in the final output, because we have Union All, which does not eliminate duplicates.

53. Countries with suppliers or customers, version 2

```
With AllCountries as
    (Select Country from Suppliers
    Union
    Select Country from Customers
    )
,SupplierCountries as
    (Select Distinct Country from Suppliers)
,CustomerCountries as
    (Select Distinct Country from Customers)
Select
    SupplierCountries.Country as SupplierCountry
    ,CustomerCountries.Country as CustomerCountry
From AllCountries
    Left Join CustomerCountries
        on AllCountries.Country = CustomerCountries.Country
```

```
    Left Join SupplierCountries
        on AllCountries.Country = SupplierCountries.Country
Order by AllCountries.Country;
```

Discussion

If you just joined from the AllCountries CTE directly to the Suppliers table and the Customers table (without doing an intermediate CTE that does a Distinct), what would happen?

Here's an alternate way of writing the query, which returns the same results. It uses derived tables instead of CTEs.

```
Select
    SupplierCountries.Country as SupplierCountry
    ,CustomerCountries.Country as CustomerCountry
From
    (Select Country from Suppliers
    Union
    Select Country from Customers)
        AllCountries
    Left Join (Select Distinct Country from Suppliers)
        CustomerCountries
        on AllCountries.Country = CustomerCountries.Country
    Left Join (Select Distinct Country from Suppliers)
        SupplierCountries
        on AllCountries.Country = SupplierCountries.Country
Order by AllCountries.Country;
```

54. Countries with suppliers or customers, version 3

```
With AllCountries as
    (Select Country from Suppliers
    Union
    Select Country from Customers
    )
,SupplierCountries as
    (Select Country, count(*) as Total from Suppliers Group by Country)
,CustomerCountries as
    (Select Country, count(*) as Total from Customers Group by Country)
Select
    AllCountries.Country
```

```
    ,IfNull(SupplierCountries.Total,0) as TotalSuppliers
    ,IfNull(CustomerCountries.Total,0) as TotalCustomers
From AllCountries
    Left Join CustomerCountries
        on AllCountries.Country = CustomerCountries.Country
    Left Join SupplierCountries
        on AllCountries.Country = SupplierCountries.Country
Order by AllCountries.Country;
```

Discussion

Note that we had to switch from Distinct to Group By in the CTE, because we needed to get the total with Count(*). You can't use Distinct in this situation.

55. First order in each country

```
with OrdersByCountry as (
    Select
        ShipCountry
        ,CustomerID
        ,OrderID
        ,Date(OrderDate) as OrderDate
        ,Row_Number()
            over (Partition by ShipCountry Order by ShipCountry, OrderID)
            as RowNumberPerCountry
    From Orders
)
Select
    ShipCountry
    ,CustomerID
    ,OrderID
    ,OrderDate
From OrdersByCountry
Where
    RowNumberPerCountry = 1
Order by
    ShipCountry;
```

Discussion

In previous versions of MySQL, before Window functions were available, there were other options to get the same results. However, they were much more difficult to write.

Having the Window functions (Row_Number() plus many others) makes it much easier to get some very useful information.

56. Customers with multiple orders in 5 day period

```
Select
    InitialOrder.CustomerID
    ,InitialOrder.OrderID as InitialOrderID
    ,Date(InitialOrder.OrderDate) as InitialOrderDate
    ,NextOrder.OrderID as NextOrderID
    ,Date(NextOrder.OrderDate) as NextOrderDate
    ,DateDiff(NextOrder.OrderDate, InitialOrder.OrderDate)
        as DaysBetweenOrders
from Orders InitialOrder
    join Orders NextOrder
        on InitialOrder.CustomerID = NextOrder.CustomerID
where
    InitialOrder.OrderID < NextOrder.OrderID
    and DateDiff(NextOrder.OrderDate, InitialOrder.OrderDate) <= 5
Order by
    InitialOrder.CustomerID
    ,InitialOrder.OrderID;
```

Discussion

Including multiple instances of a table is one way of finding the answer we need.

When aliasing tables and columns, be careful to name them something meaningful, so you can read and understand your SQL.

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

```
With NextOrderDate as (
    Select
        CustomerID
        ,Date(OrderDate) as InitialOrderDate
        ,Date(Lead(OrderDate,1)
            OVER (Partition by CustomerID order by CustomerID, OrderDate)
            ) as NextOrderDate
    From Orders
)
Select
    CustomerID
    ,InitialOrderDate
    ,NextOrderDate
    ,DateDiff (NextOrderDate, InitialOrderDate) as DaysBetweenOrders
From NextOrderDate
Where
    DateDiff (NextOrderDate, InitialOrderDate )  <= 5;
```

Discussion

There's two main ways of solving this problem, the first using multiple instances of the table (which we did in the first version of the problem), and the other using Window functions.

Which is better? If we're okay with getting a narrower resultset, I'd prefer this version, using the Lead window function, instead of the previous solution.

But if we need multiple columns from the following order, then it's best to use the first version. Otherwise, you'd need multiple calculated columns with the same Partition and Order by.

Notice that the row count between the 2 answers are slightly different, 71 and 69. One of the customers that causes this discrepancy is CustomerID ERNSH. Look at the results of the answer SQL from problem # 56. Why would one OrderID show up twice?

Congratulations!

You're finished! Now that you've completed the practice problems, you've improved your SQL skills tremendously, and increased your ability in a skill that's in enormous demand.

If you have a moment, I would really appreciate a review of this book on Amazon. Your honest opinion can help people decide between the many SQL learning options available.

Any comments and suggestions are most welcome! Please email me at:
feedback@SQLPracticeProblems.com.

Thank you!

Sylvia Moestl Vasilik