```sql
 1  -- Solutions to 'SQL Practice Problems' By Sylvia Moestl Vasilik
 2
 3  -- Not all 57 solutions appear here.  A few of the practice problems are built
     upon the problems before it and are designed
 4  -- to fix certain problems or issues with the solution that the author might have
     came up with.  If my initial solution
 5  -- did not have any problems, I did not do the next problem that was designed to
    fix the issue.
 6
 7  -- 1: Return all fields from all the shippers
 8  SELECT * FROM Shippers;
 9
10  -- 2: Select certain columns from Categories
11  SELECT CategoryName, Description
12  FROM Categories;
13
14  -- 3: Info about Sales Reps
15  SELECT FirstName, LastName, HireDate
16  FROM Employees
17  WHERE Title = 'Sales Representative';
18
19  -- 4: Info about Sales Reps inn the US
20  SELECT FirstName, LastName, HireDate
21  FROM Employees
22  WHERE Title = 'Sales Representative'
23  AND
24  COUNTRY = 'USA';
25
26  -- 5: Orders placed by secific EmployeeID
27  SELECT OrderID, OrderDate
28  FROM Orders
29  WHERE EmployeeID=5;
30
31  -- 6: Suppliers and ContactTitles
32  SELECT SupplierID, ContactName, ContactTitle
33  FROM Suppliers
34  WHERE ContactTitle!= 'Marketing Manager';
35
36  -- 7: Products with queso
37  SELECT ProductID, ProductName
38  FROM Products
39  WHERE ProductName LIKE '%queso%';
40
41  -- 8: Orders Shipping to France or Belgium
42  SELECT OrderID, CustomerID, ShipCountry
43  FROM Orders
44  WHERE ShipCountry IN ('France', 'Belgium');
45
46  -- 9: Orders shipping to any Latin American country
47  SELECT OrderID, CustomerID, ShipCountry
48  FROM Orders
49  WHERE ShipCountry IN ('Brazil', 'Mexico', 'Argentina','Venezuela');
```

```sql
50
51  -- 10: Employees ordered by age
52  SELECT FirstName, LastName, Title, BirthDate
53  FROM Employees
54  ORDER BY BirthDate ASC;
55
56  -- 11: Only show date instead of datetime
57  SELECT FirstName, LastName, Title, CAST(BirthDate AS Date) AS DateOnlyBirthDate
58  FROM Employees
59  ORDER BY BirthDate ASC;
60
61  -- 12: Employees' full name
62  SELECT FirstName, LastName, CONCAT(FirstName, ' ', LastName) AS FullName
63  FROM Employees;
64
65  -- 13: OrderDetails amount per line item
66  SELECT OrderID, ProductID, UnitPrice, Quantity, UnitPrice*Quantity AS TotalPrice
67  FROM OrderDetails
68  ORDER BY OrderID, ProductID;
69
70  -- 14: Number of Customers
71  SELECT COUNT(*)
72  FROM Customers;
73
74  -- 15: When was the first order?
75  SELECT MIN(OrderDate) AS FirstOrder
76  FROM Orders;
77
78  SELECT TOP 1 OrderDate as FirstOrder
79  FROM Orders
80  ORDER BY OrderDate ASC;
81
82  -- 16: Countries where there are customers
83  SELECT DISTINCT Country
84  FROM Customers;
85
86  SELECT Country
87  FROM Customers
88  GROUP BY Country;
89
90  -- 17: Contact titles for customers
91  SELECT ContactTitle, COUNT(ContactTitle) AS TotalContactTitle
92  FROM Customers
93  GROUP BY ContactTitle
94  ORDER BY TotalContactTitle DESC;
95
96  -- 18: Products with suppllier info
97  SELECT ProductID, ProductName, Suppliers.CompanyName AS Supplier
98  FROM Products
99  LEFT JOIN Suppliers ON Products.SupplierID = Suppliers.SupplierID;
100
101 -- 19: Orders and the shipper that was used
```

```sql
102  SELECT OrderID, CAST(OrderDate AS date) AS OrderDate, Shippers.CompanyName AS    ⇾
         Shipper
103  FROM Orders LEFT JOIN Shippers on Orders.ShipVia = Shippers.ShipperID
104  ORDER BY OrderID;
105
106  -- 20: Categories and the total products in each category
107  SELECT Categories.CategoryName, COUNT(*) AS TotalProducts
108  FROM Products
109  JOIN Categories
110  ON Products.CategoryID = Categories.CategoryID
111  GROUP BY Categories.CategoryName
112  ORDER BY TotalProducts DESC;
113
114  -- 21: Total customers per country/city
115  SELECT Country, City, Count(*) AS TotalCustomer
116  FROM Customers
117  GROUP BY Country, City
118  ORDER BY TotalCustomer DESC;
119
120  -- 22: Products that need reordering
121  SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
122  FROM Products
123  WHERE UnitsInStock < ReorderLevel
124  ORDER BY ProductID;
125
126  -- 23: Product that need reordering, continued
127  SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
128  FROM Products
129  WHERE (UnitsInStock + UnitsOnOrder) <= ReorderLevel
130  AND
131  Discontinued = 0
132  ORDER BY ProductID;
133
134  -- 24: Customer list by region
135  SELECT CustomerID, CompanyName, Region
136  FROM Customers
137  ORDER BY CASE
138      WHEN Region IS NULL THEN 1 ELSE 0
139      END,
140  Region ASC, CustomerID;
141
142  -- 25: High freight charges
143  SELECT TOP 3 ShipCountry, AVG(Freight) AS AverageFreight
144  FROM Orders
145  GROUP BY ShipCountry
146  ORDER BY AverageFreight DESC;
147
148  -- 26: High freight charges 2015
149  SELECT TOP 3 ShipCountry, AVG(Freight) AS AverageFreight
150  FROM Orders
151  WHERE DATEPART(yyyy,OrderDate)=2015
152  GROUP BY ShipCountry
```

```sql
153  ORDER BY AverageFreight DESC;
154
155  -- 27: High freight charges with between (Order 10806 isn't being tracked, the    ⮡
         OrderDate must be cast as date for the BETWEEN
156  --  string to work correctly)
157  SELECT TOP 3 ShipCountry, AVG(Freight) AS AverageFreight
158  FROM Orders
159  WHERE CAST(OrderDate AS DATE) BETWEEN '1/1/2015' AND '12/31/2015'
160  GROUP BY ShipCountry
161  ORDER BY AverageFreight DESC;
162
163  -- 28: High freight charges - Last year
164  SELECT TOP 3
165  ShipCountry, AVG(Freight) AS AverageFreight
166  FROM Orders
167  WHERE OrderDate > (
168      SELECT DATEADD(yy,-1,MAX(OrderDate))
169      FROM Orders
170      )
171  GROUP BY ShipCountry
172  ORDER BY AverageFreight DESC;
173
174  -- 29: Inventory List
175  SELECT Orders.EmployeeID, LastName, OD.OrderID, ProductName, Quantity
176  FROM OrderDetails AS OD
177  LEFT JOIN Products
178      ON OD.ProductID = Products.ProductID
179  LEFT JOIN Orders
180      ON OD.OrderID = Orders.OrderID
181  LEFT JOIN Employees
182      ON Orders.EmployeeID = Employees.EmployeeID
183  ORDER BY OD.OrderID, OD.ProductID
184  ;
185
186  -- 30: Customers with no orders
187  SELECT Customers.CustomerID AS Customers_CustomerID, Orders.CustomerID AS       ⮡
         Orders_CustomerID
188  FROM Customers
189  LEFT JOIN Orders
190      ON Customers.CustomerID = Orders.CustomerID
191  WHERE OrderID IS NULL;
192
193  -- 31: Customers with no orders for EmployeeID 4
194  SELECT DISTINCT Customers.CustomerID, Orders.CustomerID
195  FROM Customers
196  LEFT JOIN Orders
197      ON Customers.CustomerID = Orders.CustomerID
198  WHERE
199      Customers.CustomerID NOT IN (
200          SELECT DISTINCT Customers.CustomerID
201          FROM Customers
202          LEFT JOIN Orders
```

```sql
203            ON Customers.CustomerID = Orders.CustomerID
204        WHERE
205            EmployeeID = 4)
206 ;
207
208 -- 32: High-value customers (returns info for customers that spent over $10,000  ⮡
       on a single order in year 2016)
209 SELECT Customers.CustomerID, Customers.CompanyName, OrderDetails.OrderID, SUM  ⮡
       (UnitPrice*Quantity) AS TotalOrderAmount
210 FROM OrderDetails
211 JOIN Orders
212     ON Orders.OrderID = OrderDetails.OrderID
213 JOIN Customers
214     ON Orders.CustomerID = Customers.CustomerID
215 WHERE DATEPART(yyyy,Orders.OrderDate) = 2016
216 GROUP BY OrderDetails.OrderID, Customers.CustomerID, Customers.CompanyName
217 HAVING SUM(UnitPrice*Quantity) >= 10000
218 ORDER BY TotalOrderAmount DESC;
219
220 -- 33: High value customers - total orders (returns info for customers that spent ⮡
       over $15,000 total in year 2016)
221 SELECT
222 Customers.CustomerID,
223 Customers.CompanyName,
224 SUM(UnitPrice*Quantity) AS TotalOrderAmount
225 FROM OrderDetails
226 JOIN Orders
227     ON Orders.OrderID = OrderDetails.OrderID
228 JOIN Customers
229     ON Orders.CustomerID = Customers.CustomerID
230 WHERE DATEPART(yyyy,Orders.OrderDate) = 2016
231 GROUP BY Customers.CustomerID, Customers.CompanyName
232 HAVING SUM(UnitPrice*Quantity) >= 15000
233 ORDER BY TotalOrderAmount DESC;
234
235 -- 34: High value customers - with discount
236 SELECT
237 Customers.CustomerID,
238 Customers.CompanyName,
239 SUM(UnitPrice*Quantity) AS TotalsWithoutDiscount,
240 SUM(UnitPrice*Quantity*(1-Discount)) AS TotalsWithDiscount
241 FROM OrderDetails
242 JOIN Orders
243     ON Orders.OrderID = OrderDetails.OrderID
244 JOIN Customers
245     ON Orders.CustomerID = Customers.CustomerID
246 WHERE DATEPART(yyyy,Orders.OrderDate) = 2016
247 GROUP BY Customers.CustomerID, Customers.CompanyName
248 HAVING SUM(UnitPrice*Quantity*(1-Discount)) >= 10000
249 ORDER BY TotalsWithDiscount DESC;
250
251 -- 35: Month-end orders
```

```sql
252  SELECT EmployeeID, OrderID, OrderDate
253  FROM Orders
254  WHERE DATEPART(dd,(SELECT EOMONTH(OrderDate))) = DATEPART(dd, OrderDate)
255  ORDER BY EmployeeID, OrderID;
256
257  -- 36: Orders with many line items
258  SELECT TOP 10 OrderID, COUNT(*)
259  FROM OrderDetails
260  GROUP BY OrderID
261  ORDER BY COUNT(*) DESC;
262
263  -- 37: Orders - Random assortment
264  SELECT TOP 2 PERCENT OrderID
265  FROM Orders
266  ORDER BY NEWID();
267
268
269  -- 38: Accidental double entry (returns order info for orders that had at least  ⮡
        two line items with
270  -- matching quantities, both over 60 items)
271
272  SELECT OrderID, Quantity, COUNT(*) AS NumOrders
273  FROM OrderDetails
274  WHERE Quantity >= 60
275  GROUP BY OrderID, Quantity
276  HAVING Count(*) >= 2;
277
278  -- 39: Accidental double-entry details
279  SELECT OrderID, ProductID, UnitPrice, Quantity, Discount
280  FROM OrderDetails
281  WHERE OrderID IN
282   (
283      SELECT OrderID
284      FROM OrderDetails
285      WHERE Quantity >= 60
286      GROUP BY OrderID, Quantity
287      HAVING Count(*) >= 2
288   )
289  ORDER BY OrderID, Quantity;
290
291  -- 40: Orders- accidental double entry details, derived table (Fix the code)
292  SELECT
293      OrderDetails.OrderID,
294      ProductID,
295      UnitPrice,
296      Quantity,
297      Discount
298  FROM OrderDetails
299  JOIN (
300      SELECT OrderID
301      FROM OrderDetails
302      WHERE Quantity >= 60
```

```sql
303        GROUP BY OrderID, Quantity
304        HAVING Count(*) > 1)
305        PotentialProblemOrders ON
306        PotentialProblemOrders.OrderID = OrderDetails.OrderID
307 ORDER BY OrderID, ProductID;
308
309 -- 40 FIXED (Add DISTINCT on subquery)
310
311 SELECT
312     OrderDetails.OrderID,
313     ProductID,
314     UnitPrice,
315     Quantity,
316     Discount
317 FROM OrderDetails
318 JOIN (
319     SELECT DISTINCT OrderID
320     FROM OrderDetails
321     WHERE Quantity >= 60
322     GROUP BY OrderID, Quantity
323     HAVING Count(*) > 1)
324     PotentialProblemOrders ON
325     PotentialProblemOrders.OrderID = OrderDetails.OrderID
326 ORDER BY OrderID, ProductID;
327
328 -- 41: Late Orders
329 SELECT OrderID, OrderDate, RequiredDate, ShippedDate
330 FROM Orders
331 WHERE RequiredDate <= ShippedDate
332 ORDER BY OrderID;
333
334 -- 42: Late orders - Which employees?
335 SELECT Orders.EmployeeID, LastName, Count(*) AS TotalLateOrders
336 FROM Orders
337 JOIN Employees
338     ON Orders.EmployeeID = Employees.EmployeeID
339 WHERE RequiredDate <= ShippedDate
340 GROUP BY Orders.EmployeeID, LastName
341 ORDER BY TotalLateOrders DESC;
342
343 -- 43: Late orders vs. total orders
344 SELECT Orders.EmployeeID, LastName, Count(*) AS TotalOrders,
345 SUM(CASE WHEN RequiredDate <= ShippedDate THEN 1 ELSE 0 END) AS TotalLateOrders
346 FROM Orders
347 JOIN Employees
348     ON Orders.EmployeeID = Employees.EmployeeID
349 GROUP BY Orders.EmployeeID, LastName
350 ORDER BY EmployeeID;
351
352 -- 44: Late orders vs. total orders - missing employee (does not apply: my
        solution had all 9 employees)
353
```

```sql
354  -- 45: Late orders - fix null (does not apply: my solution did not have a NULL)
355
356  -- 46: Late orders vs. total orders - percentage
357  SELECT Orders.EmployeeID, LastName, Count(*) AS TotalOrders,
358  SUM(CASE WHEN RequiredDate <= ShippedDate THEN 1 ELSE 0 END) AS TotalLateOrders,
359  CAST(SUM(CASE WHEN RequiredDate <= ShippedDate THEN 1 ELSE 0 END) AS FLOAT)/CAST  ⇁
        (Count(*) AS FLOAT) AS PercentageLateOrders
360  FROM Orders
361  JOIN Employees
362      ON Orders.EmployeeID = Employees.EmployeeID
363  GROUP BY Orders.EmployeeID, LastName
364  ORDER BY EmployeeID;
365
366  -- 47: Late orders vs. total orders - fix decimal
367  SELECT Orders.EmployeeID, LastName, Count(*) AS TotalOrders,
368  SUM(CASE WHEN RequiredDate <= ShippedDate THEN 1 ELSE 0 END) AS TotalLateOrders,
369      CAST(
370          CAST(SUM(CASE WHEN RequiredDate <= ShippedDate THEN 1 ELSE 0 END) AS     ⇁
              FLOAT)
371          /
372          CAST(Count(*) AS FLOAT)
373      AS decimal(4,2))
374  AS PercentageLateOrders
375  FROM Orders
376  JOIN Employees
377      ON Orders.EmployeeID = Employees.EmployeeID
378  GROUP BY Orders.EmployeeID, LastName
379  ORDER BY EmployeeID;
380
381  -- 48: Customer grouping
382  SELECT CustomerID, SUM(UnitPrice*Quantity) AS TotalSales,
383  CASE WHEN SUM(UnitPrice*Quantity) BETWEEN 0 AND 1000 THEN 'Low '
384      WHEN SUM(UnitPrice*Quantity) BETWEEN 1000.01 AND 5000 THEN 'Medium'
385      WHEN SUM(UnitPrice*Quantity) BETWEEN 5000.01 AND 10000 THEN 'High'
386      ELSE 'Very High' END AS CustomerGroup
387  FROM OrderDetails
388  JOIN Orders
389      ON OrderDetails.OrderID = Orders.OrderID
390  WHERE DATEPART(yyyy, OrderDate)= 2016
391  GROUP BY CustomerID
392  ORDER BY CustomerID;
393
394  SELECT CustomerID, SUM(UnitPrice*Quantity) AS TotalSales,
395  CASE WHEN SUM(UnitPrice*Quantity) > 10000 THEN 'Very High'
396      WHEN SUM(UnitPrice*Quantity) > 5000 THEN 'High'
397      WHEN SUM(UnitPrice*Quantity) > 1000 THEN 'Medium'
398      ELSE 'Low' END AS CustomerGroup
399  FROM OrderDetails
400  JOIN Orders
401      ON OrderDetails.OrderID = Orders.OrderID
402  WHERE DATEPART(yyyy, OrderDate)= 2016
403  GROUP BY CustomerID
```

```sql
404  ORDER BY CustomerID;
405
406  -- 49: Customer grouping - fix NULL (Does not apply, my solution does not contain ⮑
         NULL)
407
408  -- 50: Customer grouping with percentage
409  WITH CustomerGroup_CTE(CustomerID, TotalSales, CustomerGroup) AS (
410      SELECT CustomerID, SUM(UnitPrice*Quantity) AS TotalSales,
411      CASE WHEN SUM(UnitPrice*Quantity) > 10000 THEN 'Very High'
412           WHEN SUM(UnitPrice*Quantity) > 5000 THEN 'High'
413           WHEN SUM(UnitPrice*Quantity) > 1000 THEN 'Medium'
414           ELSE 'Low' END AS CustomerGroup
415      FROM OrderDetails
416      JOIN Orders
417          ON OrderDetails.OrderID = Orders.OrderID
418      WHERE DATEPART(yyyy, OrderDate)= 2016
419      GROUP BY CustomerID
420      )
421  SELECT CustomerGroup,
422         Count(*) AS TotalInGroup,
423         CAST(Count(*) AS FLOAT)/CAST((SELECT Count(*) FROM CustomerGroup_CTE) AS ⮑
             FLOAT) AS PercentageInGroup
424  FROM CustomerGroup_CTE
425  GROUP BY CustomerGroup
426  ORDER BY TotalInGroup DESC;
427
428  -- 51: Customer grouping - flexible
429  SELECT CustomerID, SUM(UnitPrice*Quantity) AS TotalSales,
430  CASE WHEN SUM(UnitPrice*Quantity) BETWEEN (SELECT RangeBottom FROM                ⮑
       CustomerGroupThresholds WHERE CustomerGroupName='Low')   AND
431                        (SELECT RangeTop FROM CustomerGroupThresholds WHERE         ⮑
                          CustomerGroupName='Low') THEN 'Low'
432      WHEN SUM(UnitPrice*Quantity) BETWEEN (SELECT RangeBottom FROM                 ⮑
          CustomerGroupThresholds WHERE CustomerGroupName='Medium')   AND
433                        (SELECT RangeTop FROM CustomerGroupThresholds WHERE         ⮑
                          CustomerGroupName='Medium') THEN 'Medium'
434      WHEN SUM(UnitPrice*Quantity) BETWEEN (SELECT RangeBottom FROM                 ⮑
          CustomerGroupThresholds WHERE CustomerGroupName='High')   AND
435                        (SELECT RangeTop FROM CustomerGroupThresholds WHERE         ⮑
                          CustomerGroupName='High') THEN 'High'
436      ELSE 'Very High' END AS CustomerGroup
437  FROM OrderDetails
438  JOIN Orders
439      ON OrderDetails.OrderID = Orders.OrderID
440  WHERE DATEPART(yyyy, OrderDate)= 2016
441  GROUP BY CustomerID
442  ORDER BY CustomerID;
443
444  -- 52: Countries with suppliers or customers
445  SELECT DISTINCT Country
446  FROM Suppliers
447  UNION
```

```sql
448  SELECT DISTINCT Country
449  FROM Customers;
450
451  -- 53: Countries with suppliers or customers, version 2
452  SELECT Suppliers.Country AS SupplierCountry, Customers.Country AS CustomerCountry
453      FROM Suppliers
454      FULL OUTER JOIN Customers
455          ON Suppliers.Country = Customers.Country
456  GROUP BY Suppliers.Country, Customers.Country
457  ORDER BY CASE WHEN Suppliers.Country IS NULL THEN Customers.Country
458   WHEN Customers.Country IS NULL THEN Suppliers.Country ELSE Suppliers.Country
        END, Suppliers.Country;
459
460  WITH CountryList(SupplierCountry, CustomerCountry) AS
461      (SELECT DISTINCT Suppliers.Country AS SupplierCountry, Customers.Country AS
            CustomerCountry
462      FROM Suppliers
463      FULL OUTER JOIN Customers
464          ON Suppliers.Country = Customers.Country)
465  SELECT SupplierCountry, CustomerCountry
466  FROM CountryList
467  ORDER BY CASE WHEN SupplierCountry IS NULL THEN CustomerCountry
468   WHEN CustomerCountry IS NULL THEN SupplierCountry ELSE SupplierCountry END,
        SupplierCountry;
469
470   WITH SupplierCountries AS (SELECT DISTINCT Country FROM Suppliers),
        CustomersCountries AS  (SELECT DISTINCT Country FROM Customers)
471  SELECT SupplierCountries.Country AS SupplierCountry, CustomersCountries.Country
        as CustomerCountry
472  FROM SupplierCountries
473  FULL OUTER JOIN CustomersCountries
474      ON SupplierCountries.Country = CustomersCountries.Country;
475
476  -- 54: Customers with suppliers or customers, version 3
477  WITH TS AS(
478  SELECT Country, Count(*) AS TotalSuppliers
479  FROM Suppliers
480  GROUP BY Country),
481
482  TC AS(
483  SELECT Country, Count(*) AS TotalCustomers
484  FROM Customers
485  GROUP BY Country),
486
487  AllCountries AS (
488  SELECT Country FROM Customers
489  UNION
490  SELECT Country FROM Suppliers
491  )
492  SELECT AllCountries.Country,
493      CASE WHEN TotalSuppliers IS NULL THEN 0 ELSE TotalSuppliers END AS
            TotalSuppliers,
```

```sql
494        CASE WHEN TotalCustomers IS NULL THEN 0 ELSE TotalCustomers END AS
           TotalCustomers
495  FROM AllCountries
496  LEFT OUTER JOIN TC
497      ON AllCountries.Country = TC.Country
498  LEFT OUTER JOIN TS
499      ON AllCountries.Country = TS.Country;
500
501  -- 55: First order in each country
502  SELECT Orders.ShipCountry, Orders.CustomerID, CAST(Orders.OrderDate AS DATE) AS
       OrderDate, OrderID
503  FROM(
504      SELECT ShipCountry, Min(OrderDate) AS minOrderDate
505      FROM Orders
506      GROUP BY ShipCountry)
507      AS o
508  INNER JOIN Orders
509      ON o.minOrderDate = Orders.OrderDate and Orders.ShipCountry = o.ShipCountry
510  ORDER BY Orders.ShipCountry;
511
512  -- 56: Customers with multiple orders in 5 day period
513  WITH FrequentShipments AS (SELECT
514      OrderID,
515      OrderDate,
516      LEAD(OrderDate, 1, 0) OVER (PARTITION BY CustomerID ORDER BY OrderDate) AS
         NextOrder,
517      DATEDIFF(dd, CAST(OrderDate AS DATE), LEAD(OrderDate, 1, 0) OVER (PARTITION
         BY CustomerID ORDER BY OrderDate)) AS DaysApart
518      FROM Orders)
519  SELECT *
520  FROM FrequentShipments
521  WHERE DaysApart BETWEEN 0 AND 5;
522
523
524
```