```sql
-- Solutions to 'SQL Practice Problems' By Sylvia Moestl Vasilik

-- Not all 57 solutions appear here.  A few of the practice problems are built upon the problems before it and are designed
-- to fix certain problems or issues with the solution that the author might have came up with.  If my initial solution
-- did not have any problems, I did not do the next problem that was designed to fix the issue.

-- 1: Return all fields from all the shippers
SELECT * FROM Shippers;

-- 2: Select certain columns from Categories
SELECT CategoryName, Description
FROM Categories;

-- 3: Info about Sales Reps
SELECT FirstName, LastName, HireDate
FROM Employees
WHERE Title = 'Sales Representative';

-- 4: Info about Sales Reps inn the US
SELECT FirstName, LastName, HireDate
FROM Employees
WHERE Title = 'Sales Representative'
AND
COUNTRY = 'USA';

-- 5: Orders placed by secific EmployeeID
SELECT OrderID, OrderDate
FROM Orders
WHERE EmployeeID=5;

-- 6: Suppliers and ContactTitles
SELECT SupplierID, ContactName, ContactTitle
FROM Suppliers
WHERE ContactTitle!= 'Marketing Manager';

-- 7: Products with queso
SELECT ProductID, ProductName
FROM Products
WHERE ProductName LIKE '%queso%';

-- 8: Orders Shipping to France or Belgium
SELECT OrderID, CustomerID, ShipCountry
FROM Orders
WHERE ShipCountry IN ('France', 'Belgium');

-- 9: Orders shipping to any Latin American country
SELECT OrderID, CustomerID, ShipCountry
FROM Orders
WHERE ShipCountry IN ('Brazil', 'Mexico', 'Argentina','Venezuela');
```

```sql

-- 10: Employees ordered by age
SELECT FirstName, LastName, Title, BirthDate
FROM Employees
ORDER BY BirthDate ASC;

-- 11: Only show date instead of datetime
SELECT FirstName, LastName, Title, CAST(BirthDate AS Date) AS DateOnlyBirthDate
FROM Employees
ORDER BY BirthDate ASC;

-- 12: Employees' full name
SELECT FirstName, LastName, CONCAT(FirstName, ' ', LastName) AS FullName
FROM Employees;

-- 13: OrderDetails amount per line item
SELECT OrderID, ProductID, UnitPrice, Quantity, UnitPrice*Quantity AS TotalPrice
FROM OrderDetails
ORDER BY OrderID, ProductID;

-- 14: Number of Customers
SELECT COUNT(*)
FROM Customers;

-- 15: When was the first order?
SELECT MIN(OrderDate) AS FirstOrder
FROM Orders;

SELECT TOP 1 OrderDate as FirstOrder
FROM Orders
ORDER BY OrderDate ASC;

-- 16: Countries where there are customers
SELECT DISTINCT Country
FROM Customers;

SELECT Country
FROM Customers
GROUP BY Country;

-- 17: Contact titles for customers
SELECT ContactTitle, COUNT(ContactTitle) AS TotalContactTitle
FROM Customers
GROUP BY ContactTitle
ORDER BY TotalContactTitle DESC;

-- 18: Products with suppllier info
SELECT ProductID, ProductName, Suppliers.CompanyName AS Supplier
FROM Products
LEFT JOIN Suppliers ON Products.SupplierID = Suppliers.SupplierID;

-- 19: Orders and the shipper that was used
```

```sql
SELECT OrderID, CAST(OrderDate AS date) AS OrderDate, Shippers.CompanyName AS Shipper
FROM Orders LEFT JOIN Shippers on Orders.ShipVia = Shippers.ShipperID
ORDER BY OrderID;

-- 20: Categories and the total products in each category
SELECT Categories.CategoryName, COUNT(*) AS TotalProducts
FROM Products
JOIN Categories
ON Products.CategoryID = Categories.CategoryID
GROUP BY Categories.CategoryName
ORDER BY TotalProducts DESC;

-- 21: Total customers per country/city
SELECT Country, City, Count(*) AS TotalCustomer
FROM Customers
GROUP BY Country, City
ORDER BY TotalCustomer DESC;

-- 22: Products that need reordering
SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
FROM Products
WHERE UnitsInStock < ReorderLevel
ORDER BY ProductID;

-- 23: Product that need reordering, continued
SELECT ProductID, ProductName, UnitsInStock, ReorderLevel
FROM Products
WHERE (UnitsInStock + UnitsOnOrder) <= ReorderLevel
AND
Discontinued = 0
ORDER BY ProductID;

-- 24: Customer list by region
SELECT CustomerID, CompanyName, Region
FROM Customers
ORDER BY CASE
    WHEN Region IS NULL THEN 1 ELSE 0
    END,
Region ASC, CustomerID;

-- 25: High freight charges
SELECT TOP 3 ShipCountry, AVG(Freight) AS AverageFreight
FROM Orders
GROUP BY ShipCountry
ORDER BY AverageFreight DESC;

-- 26: High freight charges 2015
SELECT TOP 3 ShipCountry, AVG(Freight) AS AverageFreight
FROM Orders
WHERE DATEPART(yyyy,OrderDate)=2015
GROUP BY ShipCountry
```

```sql
ORDER BY AverageFreight DESC;

-- 27: High freight charges with between (Order 10806 isn't being tracked, the OrderDate must be cast as date for the BETWEEN
--  string to work correctly)
SELECT TOP 3 ShipCountry, AVG(Freight) AS AverageFreight
FROM Orders
WHERE CAST(OrderDate AS DATE) BETWEEN '1/1/2015' AND '12/31/2015'
GROUP BY ShipCountry
ORDER BY AverageFreight DESC;

-- 28: High freight charges - Last year
SELECT TOP 3
ShipCountry, AVG(Freight) AS AverageFreight
FROM Orders
WHERE OrderDate > (
    SELECT DATEADD(yy,-1,MAX(OrderDate))
    FROM Orders
    )
GROUP BY ShipCountry
ORDER BY AverageFreight DESC;

-- 29: Inventory List
SELECT Orders.EmployeeID, LastName, OD.OrderID, ProductName, Quantity
FROM OrderDetails AS OD
LEFT JOIN Products
    ON OD.ProductID = Products.ProductID
LEFT JOIN Orders
    ON OD.OrderID = Orders.OrderID
LEFT JOIN Employees
    ON Orders.EmployeeID = Employees.EmployeeID
ORDER BY OD.OrderID, OD.ProductID
;

-- 30: Customers with no orders
SELECT Customers.CustomerID AS Customers_CustomerID, Orders.CustomerID AS Orders_CustomerID
FROM Customers
LEFT JOIN Orders
    ON Customers.CustomerID = Orders.CustomerID
WHERE OrderID IS NULL;

-- 31: Customers with no orders for EmployeeID 4
SELECT DISTINCT Customers.CustomerID, Orders.CustomerID
FROM Customers
LEFT JOIN Orders
    ON Customers.CustomerID = Orders.CustomerID
WHERE
    Customers.CustomerID NOT IN (
        SELECT DISTINCT Customers.CustomerID
        FROM Customers
        LEFT JOIN Orders
```

```
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
SELECT EmployeeID, OrderID, OrderDate
FROM Orders
WHERE DATEPART(dd,(SELECT EOMONTH(OrderDate))) = DATEPART(dd, OrderDate)
ORDER BY EmployeeID, OrderID;

-- 36: Orders with many line items
SELECT TOP 10 OrderID, COUNT(*)
FROM OrderDetails
GROUP BY OrderID
ORDER BY COUNT(*) DESC;

-- 37: Orders - Random assortment
SELECT TOP 2 PERCENT OrderID
FROM Orders
ORDER BY NEWID();


-- 38: Accidental double entry (returns order info for orders that had at least two line items with
-- matching quantities, both over 60 items)

SELECT OrderID, Quantity, COUNT(*) AS NumOrders
FROM OrderDetails
WHERE Quantity >= 60
GROUP BY OrderID, Quantity
HAVING Count(*) >= 2;

-- 39: Accidental double-entry details
SELECT OrderID, ProductID, UnitPrice, Quantity, Discount
FROM OrderDetails
WHERE OrderID IN
 (
    SELECT OrderID
    FROM OrderDetails
    WHERE Quantity >= 60
    GROUP BY OrderID, Quantity
    HAVING Count(*) >= 2
  )
ORDER BY OrderID, Quantity;

-- 40: Orders- accidental double entry details, derived table (Fix the code)
SELECT
    OrderDetails.OrderID,
    ProductID,
    UnitPrice,
    Quantity,
    Discount
FROM OrderDetails
JOIN (
    SELECT OrderID
    FROM OrderDetails
    WHERE Quantity >= 60
```

```
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
-- 45: Late orders - fix null (does not apply: my solution did not have a NULL)

-- 46: Late orders vs. total orders - percentage
SELECT Orders.EmployeeID, LastName, Count(*) AS TotalOrders,
SUM(CASE WHEN RequiredDate <= ShippedDate THEN 1 ELSE 0 END) AS TotalLateOrders,
CAST(SUM(CASE WHEN RequiredDate <= ShippedDate THEN 1 ELSE 0 END) AS FLOAT)/CAST
  (Count(*) AS FLOAT) AS PercentageLateOrders
FROM Orders
JOIN Employees
    ON Orders.EmployeeID = Employees.EmployeeID
GROUP BY Orders.EmployeeID, LastName
ORDER BY EmployeeID;

-- 47: Late orders vs. total orders - fix decimal
SELECT Orders.EmployeeID, LastName, Count(*) AS TotalOrders,
SUM(CASE WHEN RequiredDate <= ShippedDate THEN 1 ELSE 0 END) AS TotalLateOrders,
    CAST(
        CAST(SUM(CASE WHEN RequiredDate <= ShippedDate THEN 1 ELSE 0 END) AS
          FLOAT)
        /
        CAST(Count(*) AS FLOAT)
    AS decimal(4,2))
AS PercentageLateOrders
FROM Orders
JOIN Employees
    ON Orders.EmployeeID = Employees.EmployeeID
GROUP BY Orders.EmployeeID, LastName
ORDER BY EmployeeID;

-- 48: Customer grouping
SELECT CustomerID, SUM(UnitPrice*Quantity) AS TotalSales,
CASE WHEN SUM(UnitPrice*Quantity) BETWEEN 0 AND 1000 THEN 'Low '
     WHEN SUM(UnitPrice*Quantity) BETWEEN 1000.01 AND 5000 THEN 'Medium'
     WHEN SUM(UnitPrice*Quantity) BETWEEN 5000.01 AND 10000 THEN 'High'
     ELSE 'Very High' END AS CustomerGroup
FROM OrderDetails
JOIN Orders
    ON OrderDetails.OrderID = Orders.OrderID
WHERE DATEPART(yyyy, OrderDate)= 2016
GROUP BY CustomerID
ORDER BY CustomerID;

SELECT CustomerID, SUM(UnitPrice*Quantity) AS TotalSales,
CASE WHEN SUM(UnitPrice*Quantity) > 10000 THEN 'Very High'
     WHEN SUM(UnitPrice*Quantity) > 5000 THEN 'High'
     WHEN SUM(UnitPrice*Quantity) > 1000 THEN 'Medium'
     ELSE 'Low' END AS CustomerGroup
FROM OrderDetails
JOIN Orders
    ON OrderDetails.OrderID = Orders.OrderID
WHERE DATEPART(yyyy, OrderDate)= 2016
GROUP BY CustomerID
```

```
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
SELECT DISTINCT Country
FROM Customers;

-- 53: Countries with suppliers or customers, version 2
SELECT Suppliers.Country AS SupplierCountry, Customers.Country AS CustomerCountry
    FROM Suppliers
    FULL OUTER JOIN Customers
        ON Suppliers.Country = Customers.Country
GROUP BY Suppliers.Country, Customers.Country
ORDER BY CASE WHEN Suppliers.Country IS NULL THEN Customers.Country
 WHEN Customers.Country IS NULL THEN Suppliers.Country ELSE Suppliers.Country
   END, Suppliers.Country;

WITH CountryList(SupplierCountry, CustomerCountry) AS
    (SELECT DISTINCT Suppliers.Country AS SupplierCountry, Customers.Country AS
      CustomerCountry
    FROM Suppliers
    FULL OUTER JOIN Customers
        ON Suppliers.Country = Customers.Country)
SELECT SupplierCountry, CustomerCountry
FROM CountryList
ORDER BY CASE WHEN SupplierCountry IS NULL THEN CustomerCountry
 WHEN CustomerCountry IS NULL THEN SupplierCountry ELSE SupplierCountry END,
   SupplierCountry;

 WITH SupplierCountries AS (SELECT DISTINCT Country FROM Suppliers),
   CustomersCountries AS  (SELECT DISTINCT Country FROM Customers)
SELECT SupplierCountries.Country AS SupplierCountry, CustomersCountries.Country
  as CustomerCountry
FROM SupplierCountries
FULL OUTER JOIN CustomersCountries
    ON SupplierCountries.Country = CustomersCountries.Country;

-- 54: Customers with suppliers or customers, version 3
WITH TS AS(
SELECT Country, Count(*) AS TotalSuppliers
FROM Suppliers
GROUP BY Country),

TC AS(
SELECT Country, Count(*) AS TotalCustomers
FROM Customers
GROUP BY Country),

AllCountries AS (
SELECT Country FROM Customers
UNION
SELECT Country FROM Suppliers
)
SELECT AllCountries.Country,
    CASE WHEN TotalSuppliers IS NULL THEN 0 ELSE TotalSuppliers END AS
      TotalSuppliers,
```

```sql
    CASE WHEN TotalCustomers IS NULL THEN 0 ELSE TotalCustomers END AS
      TotalCustomers
FROM AllCountries
LEFT OUTER JOIN TC
    ON AllCountries.Country = TC.Country
LEFT OUTER JOIN TS
    ON AllCountries.Country = TS.Country;

-- 55: First order in each country
SELECT Orders.ShipCountry, Orders.CustomerID, CAST(Orders.OrderDate AS DATE) AS
  OrderDate, OrderID
FROM(
    SELECT ShipCountry, Min(OrderDate) AS minOrderDate
    FROM Orders
    GROUP BY ShipCountry)
    AS o
INNER JOIN Orders
    ON o.minOrderDate = Orders.OrderDate and Orders.ShipCountry = o.ShipCountry
ORDER BY Orders.ShipCountry;

-- 56: Customers with multiple orders in 5 day period
WITH FrequentShipments AS (SELECT
    OrderID,
    OrderDate,
    LEAD(OrderDate, 1, 0) OVER (PARTITION BY CustomerID ORDER BY OrderDate) AS
      NextOrder,
    DATEDIFF(dd, CAST(OrderDate AS DATE), LEAD(OrderDate, 1, 0) OVER (PARTITION
      BY CustomerID ORDER BY OrderDate)) AS DaysApart
    FROM Orders)
SELECT *
FROM FrequentShipments
WHERE DaysApart BETWEEN 0 AND 5;


```