Solutions to the Problems in SQL Practice Problems by Sylvia Moestl Vasilik

John Weatherwax

To my family.

Introduction

This is my solution manual to some of the problems in the excellent book:

SQL Practice Problems
by Sylvia Moestl Vasilik

While the original book has solution in SQL Server I found it easiest to work the problems in SQLite and thus these solutions are in that dialect of SQL. I hope you find them useful.

```
/* P 1: */
SELECT
    *
FROM
    Shippers;


/* P 2 (EPage 20): */
SELECT CategoryName, Description FROM Categories;


/* P 3: */
SELECT
    FirstName, LastName, HireDate
FROM
    Employees
WHERE
    Title = 'Sales Representative';


/* P 4 (EPage 25): */
SELECT
    FirstName, LastName, HireDate
FROM
    Employees
WHERE
    Title ='Sales Representative' AND Country ='USA';


/* P 5 (EPage 28): */
SELECT
    OrderID, OrderDate
FROM
    Orders
WHERE
    EmployeeID = 5;
```

```
/* P 6 (EPage 32): */
SELECT
    SupplierID, ContactName, ContactTitle
FROM
    Suppliers
WHERE
    ContactTitle <> 'Manager';


/* P 7 (EPage 35): */
SELECT
    ProductID, ProductName
FROM
    Products
WHERE
    ProductName LIKE '%queso%';


/* P 8 (EPage 38): */
SELECT
    OrderID, CustomerID, ShipCountry
FROM
    Orders
WHERE
    ShipCountry IN ('France', 'Belgium');


/* P 9 (EPage 41): */
SELECT
    OrderID, CustomerID, ShipCountry
FROM
    Orders
WHERE
    ShipCountry IN ('Brazil', 'Mexico', 'Argentina', 'Venezuela');


/* P 10 (EPage 44): */
SELECT
    FirstName
    , LastName
    , Title
    , BirthDate
FROM
    Employees
ORDER BY
    BirthDate ASC;
```

```
/* P 11 (EPage 44): */
SELECT
    FirstName
    , LastName
    , Title
    , strftime('%Y-%m-%d', BirthDate) AS DateOnlyBirthDate
FROM
    Employees
ORDER BY
    BirthDate ASC;


/* P 12 (EPage 50): */
SELECT
    FirstName
    , LastName
    , FirstName || ' ' || LastName AS FullName
FROM
    Employees;


/* P 13 (EPage 53): */
SELECT
    OrderID
    , ProductID
    , UnitPrice
    , Quantity
    , UnitPrice * Quantity AS TotalPrice
FROM
    [Order Details]
ORDER BY
    OrderID
    , ProductID;


/* P 14 (EPage 56): */
SELECT
    COUNT(*) AS TotalCustomers
FROM
    Customers;


/* P 15 (EPage 59): */
SELECT
    MIN(OrderDate) AS FirstOrder
FROM
    Orders;


/* P 16 (EPage 62): */
```

```
SELECT
    Country
FROM
    Customers
GROUP BY
    Country;


/* P 17 (EPage 65): */
SELECT
    ContactTitle
    , COUNT(*) AS TotalContractTitle
FROM
    Customers
GROUP BY
    ContactTitle
ORDER BY
    TotalContractTitle DESC;


/* P 18 (EPage): */
SELECT
    p.ProductID
    , p.ProductName
    , s.CompanyName
FROM
    Products p LEFT JOIN Suppliers s ON p.SupplierID = s.SupplierID;


/* P 19 (EPage 71): */
SELECT
    OrderID
    , date(OrderDate) AS OrderDate
    , CompanyName
FROM
    Orders LEFT JOIN Shippers ON Orders.ShipVia = Shippers.ShipperID
WHERE
    OrderID < 10300
ORDER BY
    OrderID;


/* P 20 (EPage 76) */
SELECT
    CategoryName
    , COUNT(*) AS Count
FROM
    Categories JOIN Products ON Categories.CategoryID = Products.CategoryID
GROUP BY
    CategoryName
```

```
ORDER BY
    Count DESC;


/* P 21 (EPage 79) */
SELECT
    Country
    , City
    , COUNT(*) AS TotalCustomers
FROM
    Customers
GROUP BY
    Country, City
ORDER BY
    TotalCustomers DESC;


/* P 22 (EPage 82) */
SELECT
    ProductID
    , ProductName
    , UnitsInStock
    , ReorderLevel
FROM
    Products
WHERE
    UnitsInStock <= ReorderLevel
ORDER BY
    ProductID;


/* P 23 (EPage 85) */
SELECT
    ProductID
    , ProductName
    , UnitsInStock
    , UnitsOnOrder
    , ReorderLevel
    , Discontinued
FROM
    Products
WHERE
    (UnitsInStock + UnitsOnOrder) <= ReorderLevel
    AND Discontinued='0'
ORDER BY
    ProductID;


/* P 24 (EPage 88) */
SELECT
```

```
    CustomerID
    , CompanyName
    , Region
FROM
    Customers
ORDER BY
    (CASE WHEN Region IS NULL THEN 1 ELSE 0 END), Region, CustomerID;


/* P 25 (EPage 92) */
SELECT
    ShipCountry
    , AVG(Freight) AS AverageFreight
FROM
    Orders
GROUP BY
    ShipCountry
ORDER BY
    AverageFreight DESC
LIMIT
    3;


/* P 26 (EPage 97) */
SELECT
    ShipCountry
    , AVG(Freight) AS AverageFreight
FROM
    Orders
WHERE
    strftime('%Y', OrderDate) = '1997'
GROUP BY
    ShipCountry
ORDER BY
    AverageFreight DESC
LIMIT
    3;


/* P 28 (EPage 105) */
SELECT
    ShipCountry
    , AVG(Freight) AS AverageFreight
FROM
    Orders
WHERE
    OrderDate >= (SELECT datetime(julianday(MAX(OrderDate)) - 365) FROM Orders)
GROUP BY
    ShipCountry
ORDER BY
```

```
    AverageFreight DESC
LIMIT
    3;


/* P 29 (EPage 110) */
SELECT
Orders.EmployeeID
    , Employees.LastName
    , Orders.OrderID
    , Products.ProductName
    , [Order Details].Quantity
FROM
    Orders
        LEFT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
        LEFT JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID
        LEFT JOIN Products ON [Order Details].ProductID = Products.ProductID
ORDER BY
    Orders.OrderID, [Order Details].ProductID
LIMIT
    20;


/* P 30 (EPage 113) */
SELECT
    Customers.CustomerID AS Customers_CustomerID
    , Orders.CustomerID AS Orders_CustomerID
FROM
    Customers
    LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
WHERE
    Orders.CustomerID IS NULL
ORDER BY
    Orders.CustomerID;


/* P 31 (EPage 117) */
SELECT
    Customers.CustomerID AS Customers_CustomerID
    , Orders.CustomerID AS Orders_CustomerID
FROM
    Customers
    LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    AND Orders.EmployeeID = 4
WHERE
    Orders.CustomerID IS NULL
ORDER BY
    Orders.CustomerID;
```

```
/* P 32 (EPage 121) */
SELECT
    Customers.CustomerID
    , Customers.CompanyName
    , Orders.OrderID
    , SUM([Order Details].UnitPrice * [Order Details].Quantity) AS TotalOrderAmount
FROM
    Customers
    JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID
WHERE
    strftime('%Y', Orders.OrderDate) ='1998'
GROUP BY
    Customers.CustomerID, Orders.OrderID
HAVING
    TotalOrderAmount > 10000
ORDER BY
    TotalOrderAmount DESC;


/* P 33 (EPage 126) */
SELECT
    Customers.CustomerID
    , Customers.CompanyName
    , SUM([Order Details].UnitPrice * [Order Details].Quantity) AS TotalOrderAmount
FROM
    Customers
    JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID
WHERE
    strftime('%Y', Orders.OrderDate) = '1998'
GROUP BY
    Customers.CustomerID
HAVING
    TotalOrderAmount > 15000
ORDER BY
    TotalOrderAmount DESC;


/* P 34 (EPage 129) */
SELECT
    Customers.CustomerID
    , Customers.CompanyName
    , SUM([Order Details].UnitPrice
      * [Order Details].Quantity) AS TotalWithoutDiscount
    , SUM([Order Details].UnitPrice
      * [Order Details].Quantity * (1-[Order Details].Discount)) AS TotalWithDiscount
FROM
    Customers
    JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID
WHERE
    strftime('%Y', Orders.OrderDate) = '1998'
GROUP BY
    Customers.CustomerID
HAVING
    TotalWithDiscount > 15000
ORDER BY
    TotalWithDiscount DESC;


/* P 35 (EPage 133) */
SELECT
    Orders.EmployeeID
    , Orders.OrderID
    , date(Orders.OrderDate)
FROM
    Orders
WHERE
    date(Orders.OrderDate, 'start of month', '+1 month', '-1 day') = date(Orders.OrderDate)
ORDER BY
    Orders.EmployeeID
    , Orders.OrderID;


/* P 36 (EPage 136) */
SELECT
    Orders.OrderID
    , COUNT(Orders.OrderID) AS TotalOrderDetails
FROM
    Orders
    JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID
GROUP BY
    Orders.OrderID
ORDER BY
    TotalOrderDetails
LIMIT
    10;


/* P 37 (EPage 139) */
/* To select a fixed number of random rows we can use */
SELECT
    OrderID
FROM
    Orders
ORDER BY
    RANDOM()
LIMIT
    10;
```

```
/* To select a fixed percentage of random rows we can use */
SELECT
    OrderID
FROM
    Orders
ORDER BY
    RANDOM()
LIMIT
    CAST(0.02 * (SELECT COUNT(*) FROM Orders) AS INTEGER);


/* P 38 (EPage 142) */
SELECT
    OrderID
    , Quantity
    , COUNT(*) AS Number
FROM
    [Order Details]
WHERE
    Quantity >= 60
GROUP BY
    OrderID, Quantity
HAVING
    Number > 1;


/* P 39 (EPage 146) */
/* Using a CTE (common table expression) */
WITH PossibleOrderIDs AS (
SELECT
    OrderID
FROM
    [Order Details]
WHERE
    Quantity >= 60
GROUP BY
    OrderID, Quantity
HAVING
    COUNT(*) > 1
)
SELECT
    OrderID
    , ProductID
    , UnitPrice
    , Quantity
    , Discount
FROM
    [Order Details]
WHERE
```

```
    OrderID IN PossibleOrderIDs
ORDER BY
    OrderID
    , Quantity;


/* P 40 (EPage 149) */
For this problem we add the keyword DISTINCT in the SELECT statement in the subquery
in the provided SQL.


/* P 41 (EPage 152) */
SELECT
    OrderID
    , OrderDate
    , RequiredDate
    , ShippedDate
FROM
    Orders
WHERE
    ShippedDate >= RequiredDate
ORDER BY
    OrderDate ASC;


/* P 42 (EPage 156) */
SELECT
    Orders.EmployeeID
    , Employees.LastName
    , COUNT(*) As TotalLateOrders
FROM
    Orders
    JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
WHERE
    ShippedDate >= RequiredDate
GROUP BY
    Orders.EmployeeID
    , Employees.LastName
ORDER BY
    TotalLateOrders DESC;


/* P 43-47 (EPage 159) */
/* Using two CTE (common table expressions) */
WITH
-- Total orders
TotalNumberOfOrders AS (
SELECT
    EmployeeID
    , COUNT(*) AS TotalOrders
FROM
    Orders
GROUP BY
    EmployeeID
),
-- Late orders
TotalLateOrders AS (
SELECT
    EmployeeID
    , COUNT(*) AS TotalLateOrders
FROM
    Orders
WHERE
    ShippedDate >= RequiredDate
GROUP BY
    EmployeeID
)

SELECT DISTINCT
    Orders.EmployeeID
    , Employees.LastName
    , TotalOrders
    , TotalLateOrders
    , ROUND(CAST(TotalLateOrders AS FLOAT)/TotalOrders, 2) AS PercentLateOrders
FROM
    Orders
    JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
    JOIN TotalNumberOfOrders ON Orders.EmployeeID = TotalNumberOfOrders.EmployeeID
    LEFT JOIN TotalLateOrders ON Orders.EmployeeID = TotalLateOrders.EmployeeID
ORDER BY
    PercentLateOrders DESC;


/* P 48-49 (EPage 177) */
WITH CustomerOrderSizes AS (
SELECT
    Customers.CustomerID
    , Customers.CompanyName
    , SUM([Order Details].UnitPrice * [Order Details].Quantity) AS TotalOrderAmount
FROM
    Customers
    JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID
WHERE
    strftime('%Y', Orders.OrderDate) = '1998'
GROUP BY
    Customers.CustomerID
ORDER BY
    Customers.CustomerID
)
```

```
SELECT
    CustomerID
    , 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'
        ELSE 'very high'
      END AS CustomerGroup
FROM
    CustomerOrderSizes
ORDER BY
    CustomerID;


/* P 50 (EPage 185) */
WITH CustomerOrderSizes AS (
SELECT
    Customers.CustomerID
    , Customers.CompanyName
    , SUM([Order Details].UnitPrice * [Order Details].Quantity) AS TotalOrderAmount
FROM
    Customers
    JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID
WHERE
    strftime('%Y', Orders.OrderDate) = '1998'
GROUP BY
    Customers.CustomerID
ORDER BY
    Customers.CustomerID
),
CustomerGroups AS (
SELECT
    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'
        ELSE 'very high'
      END AS CustomerGroup
FROM
    CustomerOrderSizes
)

SELECT
    CustomerGroup
    , COUNT(*) AS TotalInGroup
    , ROUND(CAST(COUNT(*) AS DOUBLE)/(SELECT COUNT(*) FROM CustomerGroups), 2)
```

```
      AS PercentageInGroup
FROM
    CustomerGroups
GROUP BY
    CustomerGroup
ORDER BY
    PercentageInGroup DESC;


/* P 51 (EPage 189) */
WITH CustomerOrderSizes AS (
SELECT
    Customers.CustomerID
    , Customers.CompanyName
    , SUM([Order Details].UnitPrice * [Order Details].Quantity) AS TotalOrderAmount
FROM
    Customers
    JOIN Orders ON Customers.CustomerID = Orders.CustomerID
    JOIN [Order Details] ON Orders.OrderID = [Order Details].OrderID
WHERE
    strftime('%Y', Orders.OrderDate) = '1998'
GROUP BY
    Customers.CustomerID
ORDER BY
    Customers.CustomerID
),
CustomerGroups AS (
SELECT
    CustomerGroupName AS CustomerGroup
    , TotalOrderAmount
FROM
    CustomerOrderSizes
    JOIN CustomerGroupThresholds ON (RangeBottom < TotalOrderAmount)
    AND (TotalOrderAmount <= RangeTop)
)

SELECT
    CustomerGroup
    , COUNT(*) AS TotalInGroup
    , ROUND(CAST(COUNT(*) AS DOUBLE)/(SELECT COUNT(*) FROM CustomerGroups), 2)
      AS PercentageInGroup
FROM
    CustomerGroups
GROUP BY
    CustomerGroup
ORDER BY
    PercentageInGroup DESC;


/* P 52 (EPage 193) */
```

```
SELECT
    Country
FROM
    Suppliers
UNION
SELECT
    Country
FROM
    Customers
ORDER BY
    Country;


/* P 53 (EPage 196) */
WITH
SupplierCountries AS (
SELECT DISTINCT
    Country
FROM
    Suppliers
),
CustomerCountries AS (
SELECT DISTINCT
    Country
FROM
    Customers
)
-- Using ideas from: http://www.sqlitetutorial.net/sqlite-full-outer-join/
SELECT
    SupplierCountries.Country AS SupplierCountry
    , CustomerCountries.Country AS CustomerCountry
FROM
    SupplierCountries
    LEFT JOIN CustomerCountries USING(Country)
UNION
SELECT
    SupplierCountries.Country AS SupplierCountry
    , CustomerCountries.Country AS CustomerCountry
FROM
    CustomerCountries
    LEFT JOIN SupplierCountries USING(Country)
WHERE
    NOT ((SupplierCountry IS NULL) AND (CustomerCountry IS NULL));


/* P 54 (EPage 200) */
WITH
-- get suppliers countries (and count)
SupplierCountries AS (
SELECT
```

```
    Country
    , COUNT(*) AS TotalSuppliers
FROM
    Suppliers
WHERE
    Country IS NOT NULL
GROUP BY
    Country
) ,
-- get customer countries (and count)
CustomerCountries AS (
SELECT
    Country
    , COUNT(*) AS TotalCustomers
FROM
    Customers
WHERE
    Country IS NOT NULL
GROUP BY
    Country
) ,
-- get a union of all countries
AllCountries AS (
SELECT
    Country
FROM
    Suppliers
WHERE
    Country IS NOT NULL
UNION
SELECT
    Country
FROM
    Customers
WHERE
    Country IS NOT NULL
)
SELECT
    ac.Country
    , IFNULL(sc.TotalSuppliers, 0) AS TotalSuppliers
    , IFNULL(cc.TotalCustomers, 0) AS TotalCustomers
FROM
    AllCountries ac
    LEFT JOIN SupplierCountries sc ON ac.Country = sc.Country
    LEFT JOIN CustomerCountries cc ON ac.Country = cc.Country
ORDER BY
    ac.Country;


/* P 55 (EPage 204) */
```

```
--
-- Using ideas from:
-- https://www.xaprb.com/blog/2006/12/07/how-to-select-the-firstleastmax-row-per-group-in-sql/
--
WITH CountriesFirstOrder AS
(
-- get the first order date in each country
SELECT
    ShipCountry
    , MIN(OrderDate) AS FirstOrderDate
FROM
    Orders
GROUP BY
    ShipCountry
)
-- select the other information for this first order
SELECT
    cfo.ShipCountry
    , CustomerID
    , OrderID
    , date(OrderDate) AS OrderDate
FROM
    CountriesFirstOrder cfo
    LEFT JOIN Orders o ON (cfo.FirstOrderDate = o.OrderDate)
    AND (cfo.ShipCountry = o.ShipCountry)
ORDER BY
    cfo.ShipCountry;


/* P 56 (EPage 209) */
SELECT
    io.CustomerID
    , io.OrderID AS InitialOrderID
    , date(io.OrderDate) AS InitialOrderDate
    , so.OrderID AS NextOrderID
    , date(so.OrderDate) AS NextOrderDate
    , CAST(julianday(so.OrderDate) - julianday(io.OrderDate) AS INTEGER) AS DaysBetween
FROM
    -- io = initial order
    Orders io
    -- so = second order
    JOIN Orders so ON (io.CustomerID = so.CustomerID) AND (io.OrderID < so.OrderID)
WHERE
    DaysBetween <= 5;


/* P 57 (EPage 217) SQLite does not currently support windowing functions */
```