Coder Girls Problem Set 5 Solution

Summarizing data by calculating sums, averages, and counts is a very important skill to learn, but also one that often trips up beginners.

Remember the rules:

1) Any column appearing in the SELECT list or the ORDER BY clause must be part of an Aggregate expression or added to the GROUP BY clause.
2) To filter out rows, add an expression to the WHERE clause. To filter out groups, add a filter to the HAVING clause.

Before completing this problem set, be sure to view the video on aggregate queries.

Exercise 1

1.1 Write a query returning the count of orders placed. Use the Sales.SalesOrderHeader table.

```
SELECT COUNT(*) AS CountOfOrders
FROM Sales.SalesOrderHeader;
```

1.2 Write a query returning the average TotalDue amount from all the orders.

```
SELECT AVG(TotalDue) AS AvgTotalDue
FROM Sales.SalesOrderHeader;
```

1.3 What is the most expensive UnitPrice found in the Sales.SalesOrderDetail table?

```
SELECT MAX(UnitPrice) AS MaxPrice
FROM Sales.SalesOrderDetail;
```

1.4 Write a query listing the minimum, maximum, and average ListPrice from the Production.Product table.

```
SELECT MIN(ListPrice) AS Minimum,
       MAX(ListPrice) AS Maximum,
       AVG(ListPrice) AS Average
FROM Production.Product;
```

Exercise 2

2.1 Write a query returning the average TotalDue for each order year.

```
SELECT AVG(TotalDue) AS AvgTotalDue, YEAR(OrderDate) AS OrderYear
FROM Sales.SalesOrderHeader
GROUP BY YEAR(OrderDate);
```

2.2 Write a query that adds up the total OrderQty for each SalesOrderID. Use the Sales.SalesOrderDetail table.

```
SELECT SUM(OrderQty) AS SumOfOrderQty, SalesOrderID
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID;
```

2.3 Write a query that returns the count of products by ProductLine found in the Production.Product table.

```
SELECT COUNT(*) AS CountOfProducts, ProductLine
FROM Production.Product
GROUP BY ProductLine;
```

2.4 Write a query that returns the sum of TotalDue by CusrtomerID and order year.

```
SELECT CustomerID, YEAR(OrderDate) AS OrderYear,
        SUM(TotalDue) AS SumOfTotalDue
FROM Sales.SalesOrderHeader
GROUP BY CustomerID, YEAR(OrderDate);
```

Exercise 3

3.1 Write a query that returns the count of orders by customer, but only return the list of customers who have placed less than 4 orders.

```
SELECT CustomerID, COUNT(*) AS OrderCount
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
HAVING COUNT(*) < 4;
```

3.2 Return the list of orders where the sum of the LIneTotal column is over $2000 for a SalesOrderID. Use the Sales.SalesOrderDetail table.

```
SELECT SalesOrderID
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
HAVING SUM(LineTotal) > 2000;
```

3.3 Return the count of products where the color is filled in, but there is only 1 product returned per color and ProductModelID.

```
SELECT COUNT(*) AS CountOfProducts,
```

```
        Color, ProductModelID
FROM Production.Product
WHERE Color IS NOT NULL
GROUP BY Color, ProductModelID
HAVING COUNT(*) = 1;
```

3.4 Return a count of distinct ProductID values from the Sales.SalesOrderDetail table.

```
SELECT COUNT(DISTINCT ProductID) AS CountOfDistinctProducts
FROM Sales.SalesOrderDetail;
```