LaunchCode T-SQL Workshop Class 3 Labs

Module 5: See Class 2 for labs

Module 6: Grouping

INFO for Lab 1

Popular aggregate functions:

- COUNT
- SUM
- AVG
- MIN
- MAX

```
--One summary row returned
SELECT COUNT(*) AS CountOfRows,
    MAX(TotalDue) AS MaxTotal,
    MIN(TotalDue) AS MinTotal,
    SUM(TotalDue) AS SumOfTotal,
    AVG(TotalDue) AS AvgTotal
FROM Sales.SalesOrderHeader;
```

Group by

Any expression in the SELECT or ORDER BY clauses not in an aggregate function must be included in the GROUP BY clause

```
--Returns one row per customer
SELECT CustomerID,SUM(TotalDue) AS TotalPerCustomer
FROM Sales.SalesOrderHeader
GROUP BY CustomerID;

--Be sure to include the expression
SELECT COUNT(*) AS CountOfOrders, YEAR(OrderDate) AS OrderYear
FROM Sales.SalesOrderHeader
GROUP BY YEAR(OrderDate);
```

INFO for Lab 2

Having clause

Use to filter groups on summary values. The WHERE clause filters rows before the grouping is done.

```
SELECT CustomerID,SUM(TotalDue) AS TotalPerCustomer
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
HAVING SUM(TotalDue) > 5000;

--Eliminate rows with WHERE and groups with HAVING
SELECT CustomerID,SUM(TotalDue) AS TotalPerCustomer
FROM Sales.SalesOrderHeader
WHERE CustomerID < 20000
GROUP BY CustomerID
HAVING SUM(TotalDue) > 5000;
```

```
--The aggregate in HAVING doesn't have to appear in the
--rest of the query
SELECT CustomerID,SUM(TotalDue) AS TotalPerCustomer
FROM Sales.SalesOrderHeader
GROUP BY CustomerID
HAVING COUNT(*)= 3;
```

Lab 1: 15 minutes

1. Write a query to determine the number of customers in the Sales.Customer table.
2. Write a query that returns the overall total number of products ordered. Use the OrderQty column of the Sales.SalesOrderDetail table and the SUM function.
3. Write a query that shows the total number of items ordered *for each product*. Use the Sales.SalesOrderDetail table to write the query.
4. Write a query that provides a count of the first initial of the LastName column of the Person.Person table.

Lab 2: 15 minutes

1. Write a query that returns a count of detail lines in the Sales.SalesOrderDetail table by SalesOrderID. Include only those sales that have more than three detail lines.
2. Write a query that creates a sum of the LineTotal in the Sales.SalesOrderDetail table grouped by the SalesOrderID. Include only those rows where the sum exceeds 1,000.
3. Write a query that lists the ProductModelID from Production.Product along with a count. Display the rows that have a count that equals 1.

Solutions

Module 5: See Class 2 for solutions

Module 6

Lab 1

```
--1
SELECT COUNT(*) AS CustomerCount
FROM Sales.Customer;

--2
SELECT SUM(OrderQty) AS TotalOrdered
FROM Sales.SalesOrderDetail;

--3
SELECT ProductID, SUM(OrderQty) AS TotalOrdered
FROM Sales.SalesOrderDetail
GROUP BY ProductID;

--4
SELECT LEFT(LastName,1) AS FirstInitial, COUNT(*) AS ItemCount
FROM Person.Person
GROUP BY LEFT(LastName,1);
```

Lab 2

```
--1
SELECT SalesOrderID, COUNT(*) AS LineCount
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
HAVING COUNT(*) > 3;

--2
SELECT SalesOrderID, SUM(LineTotal) AS Total
FROM Sales.SalesOrderDetail
GROUP BY SalesOrderID
HAVING SUM(LineTotal) > 1000;

--3
SELECT ProductModelID, COUNT(*) AS ProductCount
FROM Production.Product
GROUP BY ProductModelID
HAVING COUNT(*) = 1;
```