LaunchCode T-SQL Workshop Day 2 Labs

Module 5

INFO for Lab 1

Inner Join

Returns the columns from two tables where there is a match on a key. This can be one or more columns.

```
--These tables join on the same column name
SELECT SOH.SalesOrderID, SOH.OrderDate,
       SOD.ProductID, SOD.OrderQty, SOD.LineTotal
FROM Sales.SalesOrderHeader AS SOH
INNER JOIN Sales.SalesOrderDetail AS SOD
ON SOD.SalesOrderID = SOH.SalesOrderID;

--These table join on a different column name
SELECT Cust.CustomerID, Cust.StoreID,
       Pers.FirstName, Pers.LastName
FROM Sales.Customer AS Cust
INNER JOIN Person.Person AS Pers
       ON Cust.PersonID = Pers.BusinessEntityID;

--You can continue to bring in tables
SELECT SOH.SalesOrderID, SOH.OrderDate,
       SOD.ProductID, Prod.Name, SOD.OrderQty, SOD.LineTotal
FROM Sales.SalesOrderHeader AS SOH
INNER JOIN Sales.SalesOrderDetail AS SOD
ON SOD.SalesOrderID = SOH.SalesOrderID
INNER JOIN Production.Product AS Prod
ON Prod.ProductID = SOD.ProductID;
```

INFO for LAB 2

LEFT Outer Join

Returns the rows from the first table (LEFT) and include columns from the second table (RIGHT) even if there is not a match. The table on the right will return NULL for rows that don't match.

```
--return customers even if they don't have sales
--this is filtered so you can easily see the NULL results
SELECT c.CustomerID, s.SalesOrderID, s.OrderDate
FROM Sales.Customer AS c
LEFT OUTER JOIN Sales.SalesOrderHeader AS s ON c.CustomerID = s.CustomerID
WHERE c.CustomerID IN (11028,11029,1,2,3,4);
```

Add more tables to LEFT join

Continue LEFT once you start down that path

```
SELECT C.CustomerID, SOH.SalesOrderID, SOD.SalesOrderDetailID,
    SOD.ProductID, T.Name
FROM Sales.Customer AS C
LEFT OUTER JOIN Sales.SalesOrderHeader AS SOH ON C.CustomerID = SOH.CustomerID
LEFT OUTER JOIN Sales.SalesOrderDetail AS SOD ON SOH.SalesOrderID = SOD.SalesOrderID
LEFT OUTER JOIN Sales.SalesTerritory AS T ON C.TerritoryID = T.TerritoryID
WHERE C.CustomerID IN (11028,11029,1,2,3,4);
```

RIGHT Outer Join (reference, not needed for lab)

Returns rows from the RIGHT table and include columns from the LEFT table even if there is not a match. I recommend LEFT JOIN.

```
SELECT c.CustomerID, s.SalesOrderID, s.OrderDate
FROM  Sales.SalesOrderHeader AS s
RIGHT OUTER JOIN Sales.Customer AS c ON c.CustomerID = s.CustomerID
WHERE c.CustomerID IN (11028,11029,1,2,3,4);
```

FULL Outer Join (reference, not needed for lab)

This is rarely used. Return all the rows from both sides even if there is not a match. There are no tables in AdventureWorks that work, so run this code to create a table.

```
DROP TABLE IF EXISTS Production.ProductColor;
CREATE table Production.ProductColor
    (Color nvarchar(15) NOT NULL PRIMARY KEY);
GO
--Insert most of the existing colors
INSERT INTO Production.ProductColor
SELECT DISTINCT Color
FROM Production.Product
WHERE Color IS NOT NULL and Color <> 'Silver';
--Insert some additional colors
INSERT INTO Production.ProductColor
VALUES ('Green'),('Orange'),('Purple');

--Here is the query:
SELECT c.Color AS "Color from list", p.Color, p.ProductID
FROM Production.Product AS p
FULL OUTER JOIN Production.ProductColor AS c ON p.Color = c.Color
ORDER BY p.ProductID;
```

INFO for Lab 3

LEFT outer join with NULL filter on RIGHT table

```
--Find rows in Customer that do not exist in SalesOrderHeader
SELECT c.CustomerID, s.SalesOrderID, s.OrderDate
FROM Sales.Customer AS c
LEFT OUTER JOIN Sales.SalesOrderHeader AS s ON c.CustomerID = s.CustomerID
WHERE s.SalesOrderID IS NULL;
```

Lab 1: 15 minutes

1. The HumanResources.Employee table does not contain the employee name. Join that table to the Person.Person table on the BusinessEntityID column. Display the BusinessEntityID, first and last names, job title, and birth date.
2. Join the Production.Product table to the Production.ProductSubCategory table. Then join to the Production.ProductCategory. Return the ProductID. Return the Name of the product, subcategory and category. Be sure to use aliases. You'll use ProductSubCategoryID and ProductCategoryID as keys in this example.

Lab 2: 15 minutes

1. Write a query that displays all the products along with the SalesOrderID even if an order has never been placed for that product. Join to the Sales.SalesOrderDetail table using the ProductID column. Include the ProductID, name, SalesOrderID, and OrderQty.
2. Write a query that returns all the rows from the Sales.SalesOrderHeader table joined to the Sales.SalesPerson table (SalesPersonID to BusinessEntityID) even if no orders match. Include the SalesPersonID, SalesYTD and SalesOrderID columns in the results.

Lab 3: 15 minutes

1. Change the query written in Lab 2 Question 1 so that only products that have not been ordered show up in the results of the query.
2. Change the query in Lab 2 Question 2 so that the orders with no salesperson are returned.

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.

Module 7: Subqueries and Common Table Expressions (CTE)

INFO for Lab 1

IN Subquery

Used in the WHERE clause. Must return one column.

```
--Returns customers who have orders
SELECT CustomerID, AccountNumber
FROM Sales.Customer
WHERE CustomerID IN (SELECT CustomerID FROM Sales.SalesOrderHeader);

--Returns customers who don't have orders
SELECT CustomerID, AccountNumber
FROM Sales.Customer
WHERE CustomerID NOT IN (SELECT CustomerID FROM Sales.SalesOrderHeader);
```

INFO for Lab 2

Correlated subquery

A subquery used in the SELECT list to include an aggregate or other single value. The subquery can see the main query.

```
SELECT CustomerID, SalesOrderID, OrderDate, TotalDue,
       (SELECT SUM(TotalDue)
       FROM Sales.SalesOrderHeader
       WHERE CustomerID = SOD.CustomerID) AS TotalCustSales,
       (SELECT SUM(TotalDue)
       FROM Sales.SalesOrderHeader) AS TotalSales
FROM Sales.SalesOrderHeader AS SOD
ORDER BY CustomerID;
```

INFO for Lab 2

Derived table (reference, not needed for lab)

A subquery used in place of a table in FROM. It can't see the main query. These are often nested.

```
SELECT SOH.CustomerID, SOH.SalesOrderID, SOH.OrderDate, SOH.TotalDue, TS.TotalSales
FROM Sales.SalesOrderHeader AS SOH
INNER JOIN (SELECT SUM(TotalDue) AS TotalSales, CustomerID
            FROM Sales.SalesOrderHeader
                    GROUP BY CustomerID) AS TS ON SOH.CustomerID = TS.CustomerID
ORDER BY SOH.CustomerID;
```

Common Table Expressions

Like a derived table, but is defined at the top of the query. Nesting not allowed, but you can base one CTE on a previous one.

```
WITH TS AS
      (SELECT SUM(TotalDue) AS TotalSales, CustomerID
       FROM Sales.SalesOrderHeader
       GROUP BY CustomerID)
SELECT SOH.CustomerID, SOH.SalesOrderID, SOH.OrderDate, SOH.TotalDue, TS.TotalSales
```

```
FROM Sales.SalesOrderHeader AS SOH
INNER JOIN TS ON SOH.CustomerID = TS.CustomerID
ORDER BY SOH.CustomerID;


WITH MonthTotal AS (
	SELECT SUM(TotalDue) AS MonthTotal, YEAR(OrderDate) AS OrderYear,
		MONTH(OrderDate) AS OrderMonth
	FROM Sales.SalesOrderHeader
	GROUP BY YEAR(OrderDate), MONTH(OrderDate)
	),
YearTotal AS (
	SELECT SUM(MonthTotal) AS YearTotal, OrderYear
	FROM MonthTotal
	GROUP BY OrderYear
	),
Sales AS (
	SELECT SalesOrderID, OrderDate, TotalDue,
		YEAR(OrderDate) AS OrderYear, MONTH(OrderDate) AS OrderMonth
	FROM Sales.SalesOrderHeader)
SELECT Sales.SalesOrderID, Sales.OrderDate, Sales.TotalDue,
	YearTotal.YearTotal, MonthTotal.MonthTotal
FROM Sales
INNER JOIN YearTotal ON YearTotal.OrderYear = Sales.OrderYear
INNER JOIN MonthTotal ON MonthTotal.OrderMonth = Sales.OrderMonth
	AND MonthTotal.OrderMonth = Sales.OrderMonth
ORDER BY MonthTotal.OrderYear, MonthTotal.OrderMonth;
```

Lab 1: 10 minutes

1. Using a subquery with the Sales.SalesOrderDetail table, display the product names and product ID numbers from the Production.Product table that have been ordered.

2. Change the query written in question 1 to display the products that have not been ordered.

Lab 2: 15 minutes

1. Write a query returning the list of products from Production.Product. Include the ProductID, Name, and Color. Using a correlated subquery, include the sum of OrderQty for each product from the Sales.SalesOrderDetail table. Filter the query to only return rows that have a FinishedGoodsFlag of 1.
2. Change the query you wrote in Lab 2 Question 1 so that the Average of OrderQty for each product is also included.

Lab 3: 15 minutes

1. Change the query you wrote in Lab 2 Question 1 to use a Common Table Expression (CTE) instead of the correlated subquery.
2. Change the query so that it also includes the Average calculation.

Module 8: UNION and related operators

Use the UNION operator to combine the results of two queries. The queries must have the same number of columns and compatible data types.

UNION

```
--Union eliminates duplicates
SELECT BusinessEntityID AS ID
FROM HumanResources.Employee
UNION
SELECT BusinessEntityID
FROM Person.Person
UNION
SELECT SalesOrderID
FROM Sales.SalesOrderHeader
ORDER BY ID;
```

UNION ALL

```
--Union ALL includes duplicates
SELECT BusinessEntityID AS ID
FROM HumanResources.Employee
UNION ALL
SELECT BusinessEntityID
FROM Person.Person
UNION ALL
SELECT SalesOrderID
FROM Sales.SalesOrderHeader
ORDER BY ID;
```

EXCEPT

```
--Rows in first query but not in second
SELECT BusinessEntityID AS ID
FROM HumanResources.Employee
EXCEPT
SELECT BusinessEntityID
FROM Person.Person;
```

INTERSECT

```
--Rows in both queries
SELECT BusinessEntityID AS ID
FROM HumanResources.Employee
INTERSECT
SELECT BusinessEntityID
FROM Person.Person;
```

Lab 1

1. The Person.Person table contains names for several tables. Write the following queries and then use UNION ALL operator to return one result set. Each query should return the ID, role (a literal string), first name and last name.

Table	Key joining to BusinessEntityID	Role
HumanResources.Employee	BusinessEntityID	Employee
HumanResources.JobCandidate	BusinessEntityID	Job Candidate
Sales.Customer	PersonID	Customer
Sales.SalesPerson	BusinessEntityID	Salesperson

Solutions

Module 5

Lab 1

```
--1
SELECT Emp.BusinessEntityID, P.FirstName, P.LastName,
       Emp.JobTitle, Emp.BirthDate
FROM HumanResources.Employee AS EMP
INNER JOIN Person.Person AS P ON  P.BusinessEntityID = EMP.BusinessEntityID;

--2
SELECT Prod.ProductID, Prod.Name AS ProductName,
       sub.Name AS SubCategoryName, Prod.Name AS CategoryName
FROM Production.Product AS Prod
INNER JOIN Production.ProductSubcategory AS Sub ON Sub.ProductSubcategoryID =
Prod.ProductSubcategoryID
INNER JOIN Production.ProductCategory AS Cat ON Cat.ProductCategoryID =
Sub.ProductCategoryID;
```

Lab 2

```
--1
SELECT Prod.ProductID, Prod.Name, SOD.SalesOrderID, SOD.OrderQty
FROM Production.Product AS Prod
LEFT OUTER JOIN Sales.SalesOrderDetail AS SOD
       ON SOD.ProductID = Prod.ProductID;

--2
SELECT SOH.SalesPersonID, SP.SalesYTD, SOH.SalesOrderID
FROM Sales.SalesOrderHeader AS SOH
LEFT OUTER JOIN Sales.SalesPerson AS SP
       ON SOH.SalesPersonID = SP.BusinessEntityID;
```

Lab 3

```
--1
SELECT Prod.ProductID, Prod.Name, SOD.SalesOrderID, SOD.OrderQty
FROM Production.Product AS Prod
LEFT OUTER JOIN Sales.SalesOrderDetail AS SOD
       ON SOD.ProductID = Prod.ProductID
WHERE SOD.SalesOrderID IS NULL;

--2
SELECT SOH.SalesPersonID, SP.SalesYTD, SOH.SalesOrderID
FROM Sales.SalesOrderHeader AS SOH
LEFT OUTER JOIN Sales.SalesPerson AS SP
       ON SOH.SalesPersonID = SP.BusinessEntityID
WHERE SP.BusinessEntityID IS NULL;
```

Module 6

Lab 1

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

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

```sql
--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

```sql
--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;
```

Module 7

Lab 1

```sql
--1
SELECT Prod.ProductID, Prod.Name
FROM Production.Product AS Prod
WHERE Prod.ProductID IN (
       SELECT ProductID
       FROM Sales.SalesOrderDetail);

--2
SELECT Prod.ProductID, Prod.Name
FROM Production.Product AS Prod
WHERE Prod.ProductID NOT IN (
       SELECT ProductID
       FROM Sales.SalesOrderDetail);
```

Lab 2

```sql
--1
SELECT ProductID, Name, Color,
       (SELECT SUM(OrderQty) FROM
       Sales.SalesOrderDetail
```

```
       WHERE ProductID = Prod.ProductID) AS TotalQty
FROM Production.Product AS Prod
WHERE Prod.FinishedGoodsFlag =1;

--2
SELECT ProductID, Name, Color,
       (SELECT SUM(OrderQty) FROM
       Sales.SalesOrderDetail
       WHERE ProductID = Prod.ProductID) AS TotalQty,
       (SELECT AVG(OrderQty) FROM
       Sales.SalesOrderDetail
       WHERE ProductID = Prod.ProductID) AS AvgQty
FROM Production.Product AS Prod
WHERE Prod.FinishedGoodsFlag =1;
```

Lab 3

```
--1
WITH Sales AS (
       SELECT ProductID, SUM(OrderQty) AS TotalQty
       GROUP BY ProductID)
SELECT Prod.ProductID, Name, Color, TotalQty
FROM Production.Product AS Prod
INNER JOIN Sales ON Sales.ProductID = Prod.ProductID;


--2
WITH Sales AS (
       SELECT ProductID, SUM(OrderQty) AS TotalQty,
              AVG(OrderQty) AS AvgQty
       GROUP BY ProductID)
SELECT Prod.ProductID, Name, Color, TotalQty, AvgQty
FROM Production.Product AS Prod
INNER JOIN Sales ON Sales.ProductID = Prod.ProductID;
```

Module 8

Lab 1

```
SELECT Emp.BusinessEntityID, Pers.FirstName, Pers.LastName, 'Employee' AS Role
FROM Person.Person AS Pers
INNER JOIN HumanResources.Employee AS Emp
       ON Emp.BusinessEntityID = Pers.BusinessEntityID
UNION ALL
SELECT JC.BusinessEntityID, Pers.FirstName, Pers.LastName, 'Job Candidate'
FROM Person.Person AS Pers
INNER JOIN HumanResources.JobCandidate AS JC
       ON JC.BusinessEntityID = Pers.BusinessEntityID
UNION ALL
SELECT Cust.PersonID, Pers.FirstName, Pers.LastName, 'Customer'
FROM Person.Person AS Pers
INNER JOIN Sales.Customer AS Cust
       ON Cust.PersonID = Pers.BusinessEntityID
UNION ALL
```

```sql
SELECT SP.BusinessEntityID, Pers.FirstName, Pers.LastName, 'Salesperson'
FROM Person.Person AS Pers
INNER JOIN Sales.SalesPerson AS SP
       ON SP.BusinessEntityID = Pers.BusinessEntityID;
```