Coder Girls Problem Set 4

Most of the time when on the job, you will not write queries for just one table; your queries will join two or more tables together. Two tables can be joined together by using the INNER JOIN or OUTER JOIN syntax and a join condition. The join condition is an expression that returns true or false, and it's usually a column from one table equal to a column from the other table. It is often the foreign key column from one table and the primary key column from the other table. SQL Server does not enforce this, however. The join condition can be any expression that returns true or false as long as the data types are compatible.

You may be wondering how you will know which columns to include in the join condition. It depends on the situation. If you are lucky, primary and foreign keys will be defined on the tables. Often, you just have to understand the data to figure it out. Since SQL Server does not keep you from making a mistake, you will probably want to use the TOP keyword to restrict the number of rows returned while you are developing.

To prepare for this problem set, be sure to watch this video on INNER JOIN and this video on OUTER JOIN.

Exercise 1

1.1 Write a query that returns the JobTitle, Birtdate, first and last names. Join the HumanResources.Employee table to the Person.Person table. In this case, you can join on the BusinessEntityID column from both tables.

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

1.2 Customers can be joined to the Person table by the PersonID from the Sales.Customer table to the BuisinessEntityID from the Person.Person table. Join the tables and return the CustomerID, TerritoryID, first, last and middle names.

```
SELECT C.CustomerID, C.StoreID, C.TerritoryID,
        P.FirstName, P.MiddleName, P.LastName
FROM Sales.Customer AS C
INNER JOIN Person.Person AS P
        ON C.PersonID = P.BusinessEntityID;
```

1.3 Join the Sales.SalesOrderDetail table to the Production.Product table on ProductID. Return a list containing the SalesOrderID, ProductID, OrderQty, and product names.

```
SELECT SOD.SalesOrderID, P.ProductID,
        SOD.OrderQty, P.Name
```

```
FROM Sales.SalesOrderDetail AS SOD
INNER JOIN Production.Product AS P
        ON P.ProductID = SOD.ProductID;
```

1.4 Join the HumanResources.JobCandidate table to the Person.Person table by BusinessEntityID. Return the first, middle and last names along with the JobCandidateID column.

```
SELECT P.FirstName, P.MiddleName, P.LastName,
        JC.JobCandidateID
FROM HumanResources.JobCandidate AS JC
INNER JOIN Person.Person AS P
        ON P.BusinessEntityID = JC.BusinessEntityID;
```

Exercise 2

2.1 Change the query found in question 1.3 so that it includes the order date. HINT: You will have to join to one more table.

```
SELECT SOD.SalesOrderID, SOH.OrderDate, P.ProductID,
        SOD.OrderQty, P.Name
FROM Sales.SalesOrderDetail AS SOD
INNER JOIN Production.Product AS P
        ON P.ProductID = SOD.ProductID
INNER JOIN Sales.SalesOrderHeader AS SOH
        ON SOH.SalesOrderID = SOD.SalesOrderID;
```

2.2 Write a query that produces a list of the customer names and their SalesOrderID numbers. You will have to join three tables: Person.Person, Sales.Customer, and Sales.SalesOrderHeader.

```
SELECT P.FirstName, P.LastName,
        SOH.SalesOrderID
FROM Sales.SalesOrderHeader AS SOH
JOIN Sales.Customer AS C
        ON SOH.CustomerID = C.CustomerID
JOIN Person.Person AS P
        ON P.BusinessEntityID = C.PersonID;
```

2.3 Write a query that returns the list of employees and their current department. To do this, you will have to join the HumanResources.Employee table to the HumanResources.EmployeeDepartmentHistory table and the HumanResources.Department table. You must also return rows where the EndDate from the history table is NULL to get the latest data. Include BusinessEntityID, job title and department name in the results.

```
SELECT E.BusinessEntityID, E.JobTitle,
```

```
        D.Name
FROM HumanResources.Employee AS E
JOIN HumanResources.EmployeeDepartmentHistory AS H
        ON H.BusinessEntityID = E.BusinessEntityID
JOIN HumanResources.Department AS D
        ON D.DepartmentID = H.DepartmentID
WHERE H.EndDate IS NULL;
```

Exercise 3

3.1 Return a list of the ProductID and names along with the SalesOrderID. Include all products even if they have not been ordered.

```
SELECT P.ProductID, P.Name, SOD.SalesOrderID
FROM Production.Product AS P
LEFT JOIN Sales.SalesOrderDetail AS SOD
        ON SOD.ProductID = P.ProductID;
```

3.2 Write a query that returns a list of the names in the Person.Person table. If the individual also happens to be a customer, return the CustomerID. Join the PersonID to the BusinessEntityID.

```
SELECT P.FirstName, P.LastName, C.CustomerID
FROM Person.Person AS P
LEFT JOIN Sales.Customer AS C
        ON C.PersonID = P.BusinessEntityID;
```

3.3 Modify the query from 3.2 to include the list of orders placed as well. Include SalesOrderID and OrderDate. Make sure that none of the names drop out of the results.

```
SELECT P.FirstName, P.LastName, C.CustomerID, SalesOrderID,
        SOH.OrderDate
FROM Person.Person AS P
LEFT JOIN Sales.Customer AS C
        ON C.PersonID = P.BusinessEntityID
LEFT JOIN Sales.SalesOrderHeader AS SOH
ON SOH.CustomerID = C.CustomerID;
```