CSCI-331 Group Project 1

Tristen Aguilar’s 20 Queries

Proposition 01: There are Customers that placed orders on Feb 2nd, 2016.

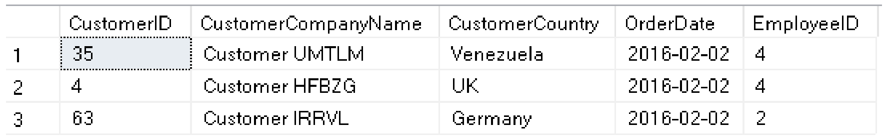
The Query above is asking you to find all of the Customers that have placed orders on the date Feb 2nd, 2016. First, you must identify which tables you need to use. For this question, you would need to use the Customer Table in order to return all of the customers information as well as the Orders table to receive date in which the customer has placed the order. Finally, make sure to check for results where the customer ID's match and make sure the order date is Feb 2nd, 2016.

Projected Columns

Table Name	**Column Name**
Sales.Order	OrderDate, EmployeeID
Sales.Customer	CustomerID, CustomerCompanyName, CustomerCountry

Table Results

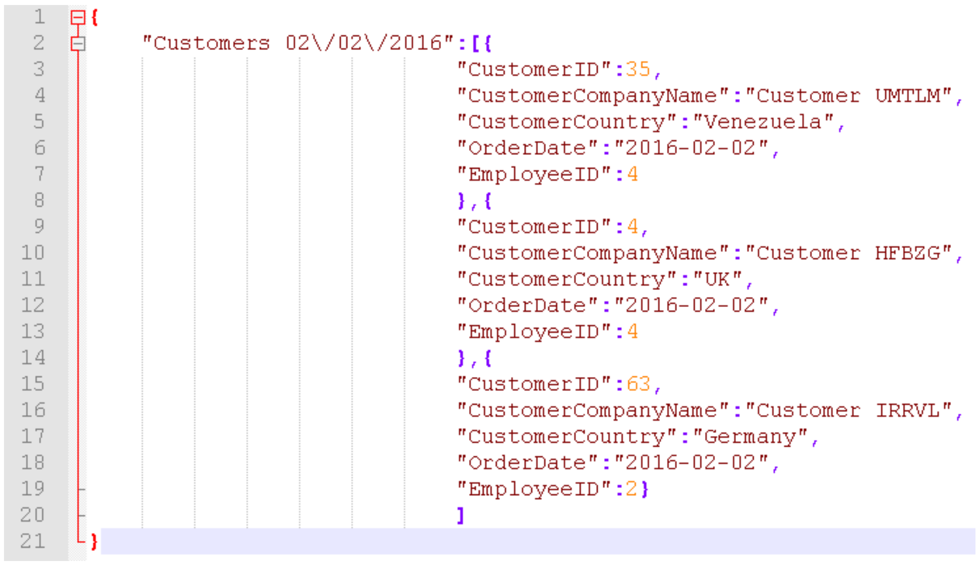
```sql
SELECT
	C.CustomerID,
	C.CustomerCompanyName,
	C.CustomerCountry,
	O.OrderDate,
	O.EmployeeID
FROM Sales.[Order] AS O
INNER JOIN Sales.Customer AS C
ON O.CustomerID = C.CustomerID
AND O.OrderDate = '20160202';
```



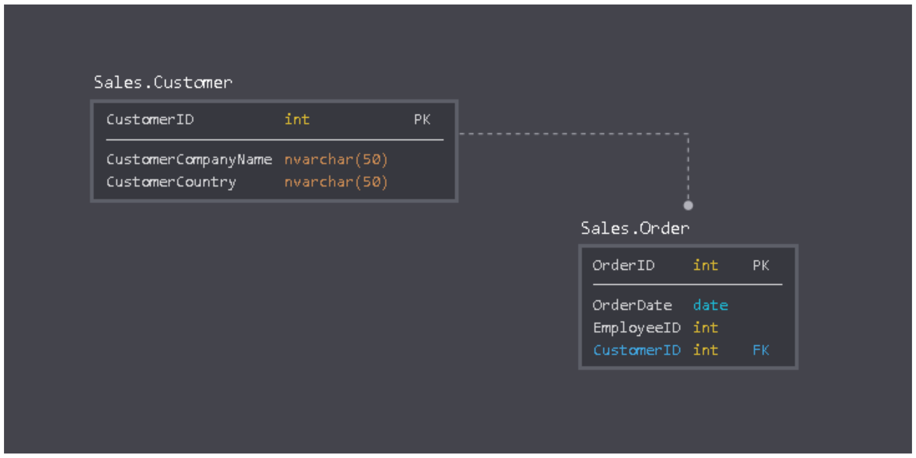
```
	CustomerID	CustomerCompanyName	CustomerCountry	OrderDate	EmployeeID
1	35	Customer UMTLM	Venezuela	2016-02-02	4
2	4	Customer HFBZG	UK	2016-02-02	4
3	63	Customer IRRVL	Germany	2016-02-02	2
```

JSON Results

```sql
SELECT
	C.CustomerID,
	C.CustomerCompanyName,
	C.CustomerCountry,
	O.OrderDate,
	O.EmployeeID
FROM Sales.[Order] AS O
INNER JOIN Sales.Customer AS C
ON O.CustomerID = C.CustomerID
AND O.OrderDate = '20160202'
FOR JSON PATH, ROOT('Customers 02/02/2016');
```



```
{
    "Customers 02\/02\/2016":[{
                        "CustomerID":35,
                        "CustomerCompanyName":"Customer UMTLM",
                        "CustomerCountry":"Venezuela",
                        "OrderDate":"2016-02-02",
                        "EmployeeID":4
                        },{
                        "CustomerID":4,
                        "CustomerCompanyName":"Customer HFBZG",
                        "CustomerCountry":"UK",
                        "OrderDate":"2016-02-02",
                        "EmployeeID":4
                        },{
                        "CustomerID":63,
                        "CustomerCompanyName":"Customer IRRVL",
                        "CustomerCountry":"Germany",
                        "OrderDate":"2016-02-02",
                        "EmployeeID":2}
                        ]
}
```



Proposition 02: Return all order details for orders placed between April and May of 2016 that shipped to Madrid.

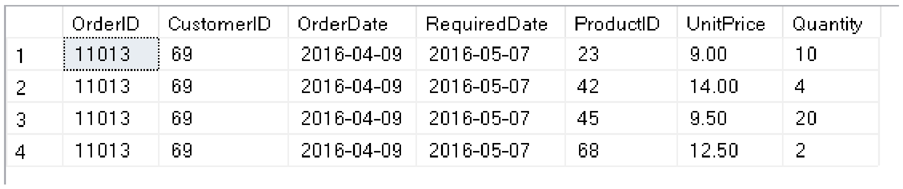
The Query above is asking you to find all order details placed Between April and May of 2016 that were shipped to Madrid, Spain. Since you are concerned with finding the Order Details, that will be one of the tables we use. In addition to that, we also need to use the Orders table since we need to look at all of the dates that orders were placed. Then, once we have linked the two tables together based on the OrderID, we must check for 2 conditions; First being that the order was placed between April and May, and secondly that the order was shipped to Madrid.

Projected Columns

Table Name	Column Name
Sales.Order	OrderID, CustomerID, OrderDate, RequiredDate
Sales.OrderDetails	ProductID, UnitPrice, Quantity

Table Results

```
SELECT
	O.OrderID,
	O.CustomerID,
	O.OrderDate,
	O.RequiredDate,
	OD.ProductID,
	OD.UnitPrice,
	OD.Quantity
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
ON OD.OrderID = O.OrderID
AND (O.OrderDate >= '20160401' AND O.OrderDate < '20160601')
AND ShipToCity = 'Madrid';
```

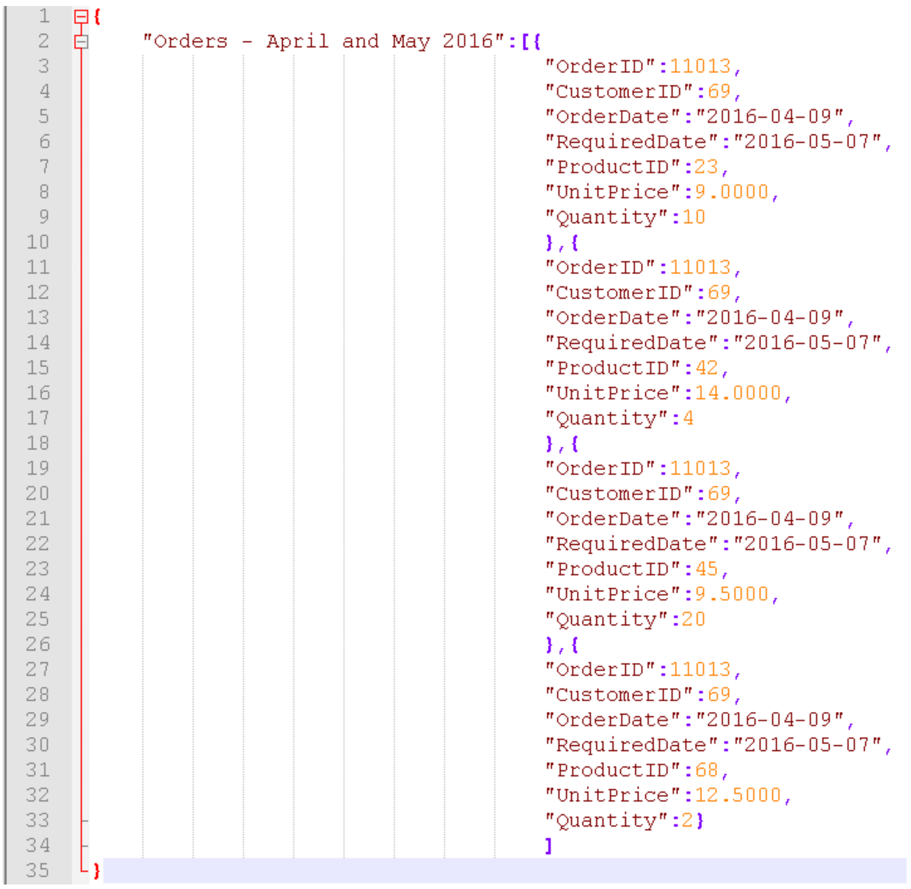


	OrderID	CustomerID	OrderDate	RequiredDate	ProductID	UnitPrice	Quantity
1	11013	69	2016-04-09	2016-05-07	23	9.00	10
2	11013	69	2016-04-09	2016-05-07	42	14.00	4
3	11013	69	2016-04-09	2016-05-07	45	9.50	20
4	11013	69	2016-04-09	2016-05-07	68	12.50	2

JSON Results

```
SELECT
	O.OrderID,
	O.CustomerID,
	O.OrderDate,
	O.RequiredDate,
	OD.ProductID,
	OD.UnitPrice,
	OD.Quantity
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
```

```sql
ON OD.OrderID = O.OrderID
AND (O.OrderDate >= '20160401' AND O.OrderDate < '20160601')
AND ShipToCity = 'Madrid'
FOR JSON PATH, ROOT('Orders - April and May 2016');
```



```json
{
    "Orders - April and May 2016":[{
                                "OrderID":11013,
                                "CustomerID":69,
                                "OrderDate":"2016-04-09",
                                "RequiredDate":"2016-05-07",
                                "ProductID":23,
                                "UnitPrice":9.0000,
                                "Quantity":10
                                },{
                                "OrderID":11013,
                                "CustomerID":69,
                                "OrderDate":"2016-04-09",
                                "RequiredDate":"2016-05-07",
                                "ProductID":42,
                                "UnitPrice":14.0000,
                                "Quantity":4
                                },{
                                "OrderID":11013,
                                "CustomerID":69,
                                "OrderDate":"2016-04-09",
                                "RequiredDate":"2016-05-07",
                                "ProductID":45,
                                "UnitPrice":9.5000,
                                "Quantity":20
                                },{
                                "OrderID":11013,
                                "CustomerID":69,
                                "OrderDate":"2016-04-09",
                                "RequiredDate":"2016-05-07",
                                "ProductID":68,
                                "UnitPrice":12.5000,
                                "Quantity":2}
                                ]
}
```

Proposition 03: Return the Shipper Company Name and Required Dates for all orders that were required to be shipped on Jan 1st, 2016

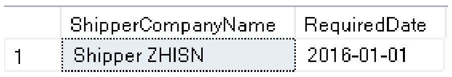
The Query above is asking you to find all Shipper Company Names that possessed Required Shipping Dates on January 1st, 2016. First, you must use both the Shipper table since it is the only table that contains the names of the shipping companies. Secondly, you must use the Order table which has the ShipperID attribute to link with the shipper table, and it also contains the Required Date. Then, you must return the Shipping Company Name where the Date Required matches Jan 1st, 2016.

Projected Columns

Table Name	Column Name
Sales.Shipper	ShipperCompanyName
Sales.Order	RequiredDate

Table Results

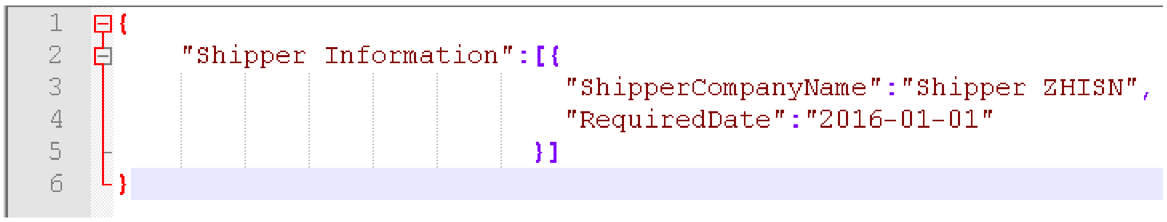
```sql
SELECT
	S.ShipperCompanyName,
	O.RequiredDate
FROM Sales.Shipper AS S
INNER JOIN Sales.[Order] AS O
ON S.ShipperID = O.ShipperID AND O.RequiredDate = '20160101';
```



	ShipperCompanyName	RequiredDate
1	Shipper ZHISN	2016-01-01

JSON Results

```sql
SELECT
	S.ShipperCompanyName,
	O.RequiredDate
FROM Sales.Shipper AS S
INNER JOIN Sales.[Order] AS O
ON S.ShipperID = O.ShipperID AND O.RequiredDate = '20160101'
FOR JSON PATH, ROOT('Shipper Information');
```



```json
{
    "Shipper Information":[{
                            "ShipperCompanyName":"Shipper ZHISN",
                            "RequiredDate":"2016-01-01"
                        }]
}
```

Proposition 04: Return the number of products that were discontinued with a Unit Price greater than 50 dollars.

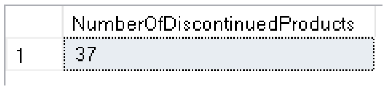
The Query above is asking you to return all products that were discontinued where the price of the item was greater than 50 dollars. For this query, is it necessary to first reference the Production.Product table since that table contains the information as to whether or not the item has been discontinued. Next, we need to also use the Order Detail table because that table will have the Unit Prices for all of the products. Once you have joined both tables using an inner join, you will have to do a quick check to see which products have been discontinued by using the boolean value of true, '1'. After that, you need to check if the Unit Price is greater than 50.

Projected Columns

Table Name	Column Name
Production.Product	Discontinued
Sales.OrderDetail	N/A

Table Results

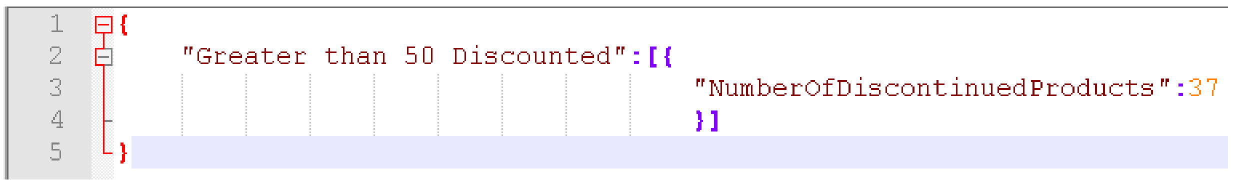
```
SELECT
       COUNT(Discontinued) AS NumberOfDiscontinuedProducts
FROM Production.Product AS P
INNER JOIN Sales.OrderDetail AS OD
ON P.ProductID = OD.ProductID
AND P.Discontinued = 1
AND OD.UnitPrice > 50;
```



	NumberOfDiscontinuedProducts
1	37

JSON Results

```
SELECT
       COUNT(Discontinued) AS NumberOfDiscontinuedProducts
FROM Production.Product AS P
INNER JOIN Sales.OrderDetail AS OD
ON P.ProductID = OD.ProductID
AND P.Discontinued = 1
AND OD.UnitPrice > 50
FOR JSON PATH, ROOT('Greater than 50 Discounted');
```



```
{
    "Greater than 50 Discounted":[{
                                        "NumberOfDiscontinuedProducts":37
                                        }]
}
```

Proposition 05: Return all the Design Engineers that are Females and their Pay Rates.

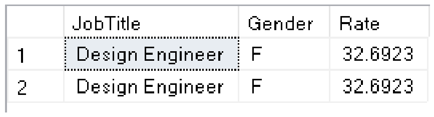
The Query above is asking you to return all Female Design Engineers and their pay rates. For this problem, you have to use two tables, first being the Human Resources Employee Table. You need to use this table in order to check to see which employees are Females. Also, you need this table to check to see if the employees job title is a Design Engineer. Next, you need to use the Employee Pay History Table to connect to the Employee table where the BusinessEntityID's are equal.

Projected Columns

Table Name	Column Name
HumanResources.Employee	JobTitle, Gender
HumanResources.EmployeePayHistory	Rate

Table Results

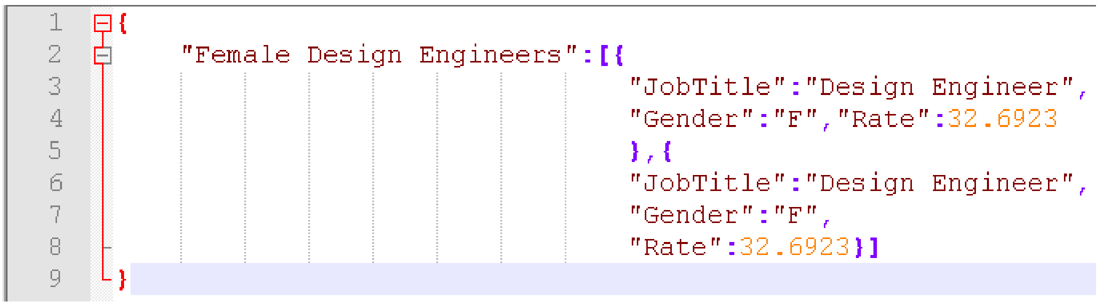
```
SELECT
        E.JobTitle,
        E.Gender,
        PH.Rate
FROM HumanResources.Employee AS E
INNER JOIN HumanResources.EmployeePayHistory AS PH
ON E.BusinessEntityID = PH.BusinessEntityID
AND E.Gender = 'F'
AND E.JobTitle = 'Design Engineer';
```



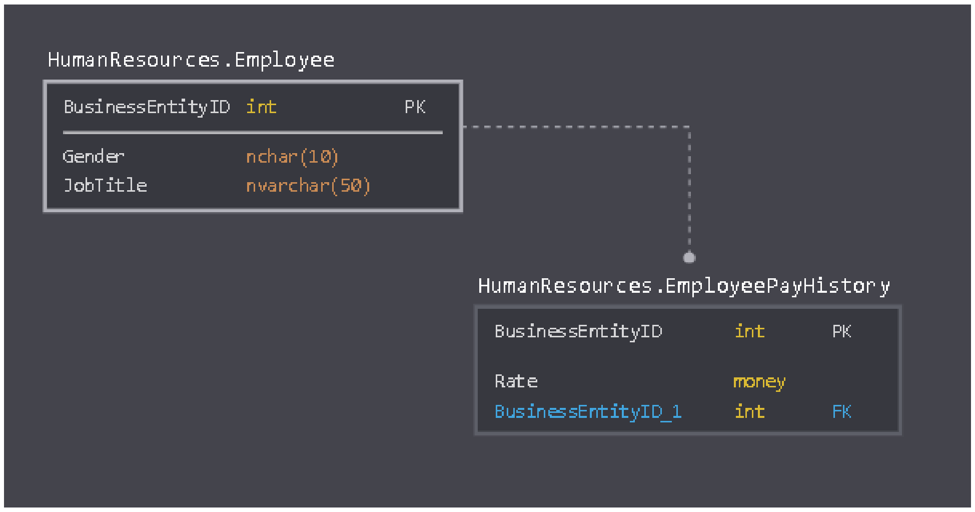
	JobTitle	Gender	Rate
1	Design Engineer	F	32.6923
2	Design Engineer	F	32.6923

JSON Results

```sql
SELECT
       E.JobTitle,
       E.Gender,
       PH.Rate
FROM HumanResources.Employee AS E
INNER JOIN HumanResources.EmployeePayHistory AS PH
ON E.BusinessEntityID = PH.BusinessEntityID
AND E.Gender = 'F'
AND E.JobTitle = 'Design Engineer'
FOR JSON PATH, ROOT('Female Design Engineers');
```



```
{
    "Female Design Engineers":[{
                                  "JobTitle":"Design Engineer",
                                  "Gender":"F","Rate":32.6923
                                  },{
                                  "JobTitle":"Design Engineer",
                                  "Gender":"F",
                                  "Rate":32.6923}]
}
```



Proposition 06: Return all of the Products that have a Unit Price greater than the average price of all products and their Category Description. In addition to that, Group them by their Unit Price and provide the number of Supplier Company Name's.

The Query above is asking you to return all queries where the average of all the products is less than the specific product price. Then, once you have retrieved all products, return their description. Also, you need to return the number of suppliers that provide that certain product and finally group them by their unit Price. To start, we must use the Product Category class in order to retrieve the Product Description.

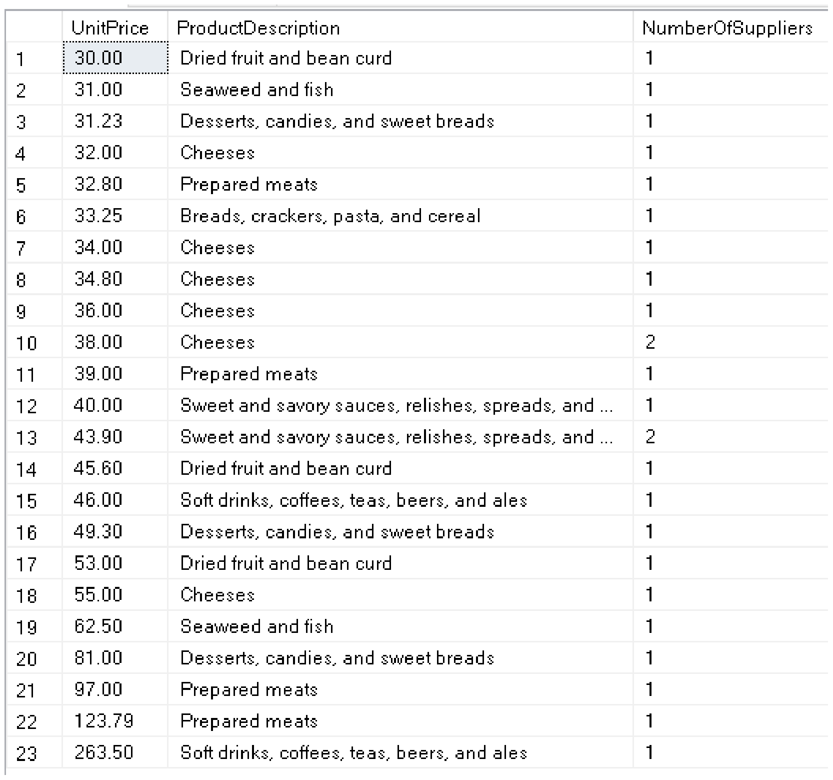
Then, we have to use the Production.Product table in order to get the price of each product. Lastly, we use the Production Supplier table since we know that table contains the details on the supplier company's name. After we link all of the tables, we check that all final queries include products with unit prices greater than the average price of all products. At last, group the information by the unit price.

Projected Columns

Table Name	Column Name
Production.Category	Description
Production.Product	UnitPrice
Production.Supplier	SupplierCompanyName

Table Results

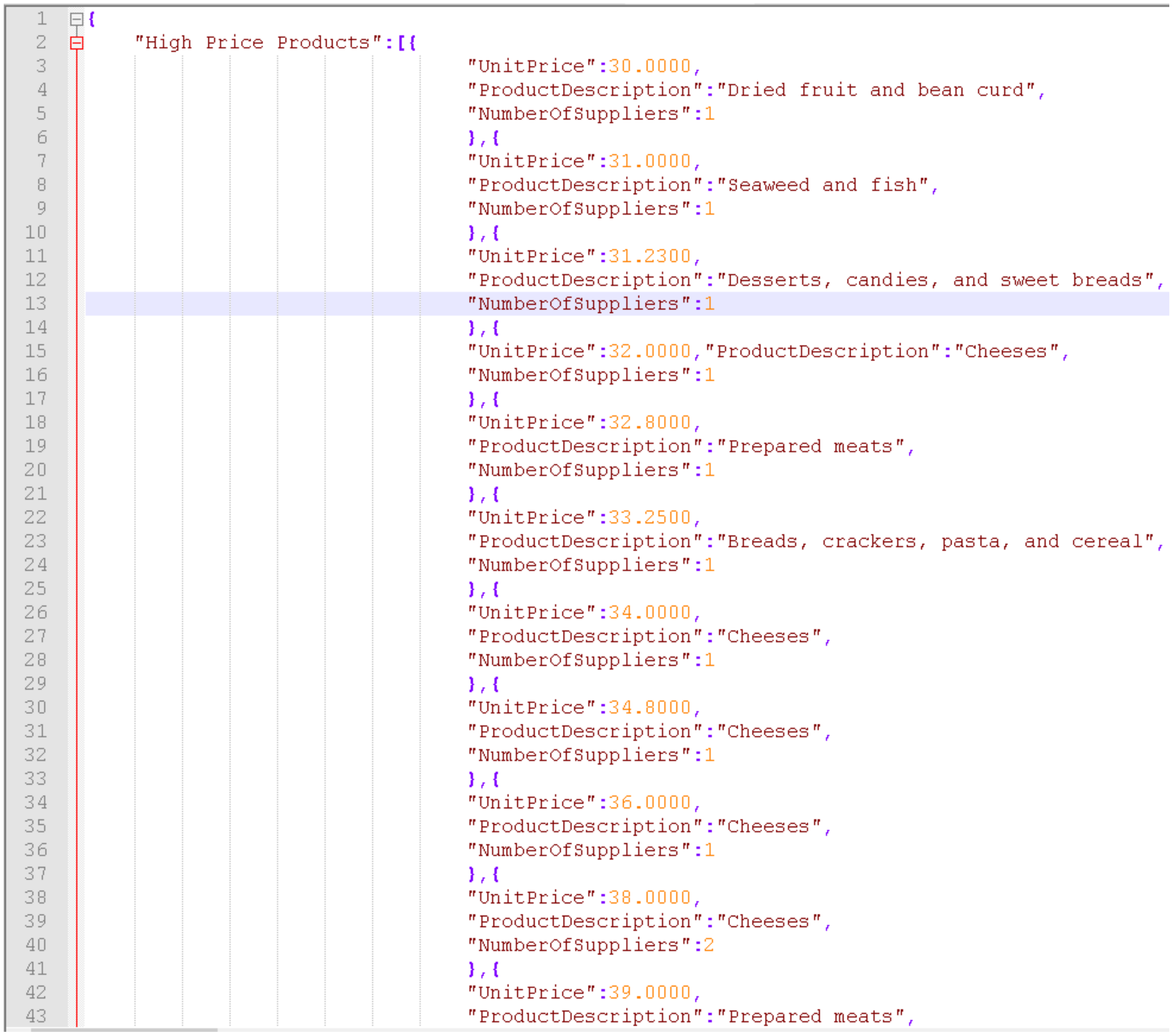
```
DECLARE @PriceAVG int;
SET @PriceAVG = (SELECT AVG(UnitPrice) FROM Production.Product);

SELECT
       P.UnitPrice,
       MAX(C.[Description]) AS ProductDescription,
       COUNT(S.SupplierCompanyName) AS NumberOfSuppliers
FROM Production.Category AS C
INNER JOIN Production.Product AS P ON C.CategoryID = P.CategoryID
INNER JOIN Production.Supplier AS S ON P.SupplierID = S.SupplierID
WHERE P.UnitPrice > @PriceAVG
GROUP BY P.UnitPrice;
```



	UnitPrice	ProductDescription	NumberOfSuppliers
1	30.00	Dried fruit and bean curd	1
2	31.00	Seaweed and fish	1
3	31.23	Desserts, candies, and sweet breads	1
4	32.00	Cheeses	1
5	32.80	Prepared meats	1
6	33.25	Breads, crackers, pasta, and cereal	1
7	34.00	Cheeses	1
8	34.80	Cheeses	1
9	36.00	Cheeses	1
10	38.00	Cheeses	2
11	39.00	Prepared meats	1
12	40.00	Sweet and savory sauces, relishes, spreads, and ...	1
13	43.90	Sweet and savory sauces, relishes, spreads, and ...	2
14	45.60	Dried fruit and bean curd	1
15	46.00	Soft drinks, coffees, teas, beers, and ales	1
16	49.30	Desserts, candies, and sweet breads	1
17	53.00	Dried fruit and bean curd	1
18	55.00	Cheeses	1
19	62.50	Seaweed and fish	1
20	81.00	Desserts, candies, and sweet breads	1
21	97.00	Prepared meats	1
22	123.79	Prepared meats	1
23	263.50	Soft drinks, coffees, teas, beers, and ales	1

JSON Results

```
DECLARE @PriceAVG int;
SET @PriceAVG = (SELECT AVG(UnitPrice) FROM Production.Product);

SELECT
       P.UnitPrice,
       MAX(C.[Description]) AS ProductDescription,
       COUNT(S.SupplierCompanyName) AS NumberOfSuppliers
FROM Production.Category AS C
INNER JOIN Production.Product AS P ON C.CategoryID = P.CategoryID
INNER JOIN Production.Supplier AS S ON P.SupplierID = S.SupplierID
WHERE P.UnitPrice > @PriceAVG
GROUP BY P.UnitPrice
FOR JSON PATH, ROOT('High Price Products');
```



```
{
    "High Price Products":[{
                            "UnitPrice":30.0000,
                            "ProductDescription":"Dried fruit and bean curd",
                            "NumberOfSuppliers":1
                            },{
                            "UnitPrice":31.0000,
                            "ProductDescription":"Seaweed and fish",
                            "NumberOfSuppliers":1
                            },{
                            "UnitPrice":31.2300,
                            "ProductDescription":"Desserts, candies, and sweet breads",
                            "NumberOfSuppliers":1
                            },{
                            "UnitPrice":32.0000,"ProductDescription":"Cheeses",
                            "NumberOfSuppliers":1
                            },{
                            "UnitPrice":32.8000,
                            "ProductDescription":"Prepared meats",
                            "NumberOfSuppliers":1
                            },{
                            "UnitPrice":33.2500,
                            "ProductDescription":"Breads, crackers, pasta, and cereal",
                            "NumberOfSuppliers":1
                            },{
                            "UnitPrice":34.0000,
                            "ProductDescription":"Cheeses",
                            "NumberOfSuppliers":1
                            },{
                            "UnitPrice":34.8000,
                            "ProductDescription":"Cheeses",
                            "NumberOfSuppliers":1
                            },{
                            "UnitPrice":36.0000,
                            "ProductDescription":"Cheeses",
                            "NumberOfSuppliers":1
                            },{
                            "UnitPrice":38.0000,
                            "ProductDescription":"Cheeses",
                            "NumberOfSuppliers":2
                            },{
                            "UnitPrice":39.0000,
                            "ProductDescription":"Prepared meats",
```

Proposition 07: Return the number of customers in each country that received an order between May and August of 2016 and order the results by country.

The Query above is asking for you to return the number of customers that received an order that they made between the dates of May and August of the year 2016. It then also asks you to return them in order of the country in which of the customers live in. First, we must use both the Sales.Customer table along with the Sales.Order table since they both have information on when the order was placed and where the customer lives. Then, once you have joined the table based upon the equivalence of their CustomerID's, you will then check to see if the RequiredDate is between the respected range and finally you will order and group the information based on the customer's country.

Projected Columns

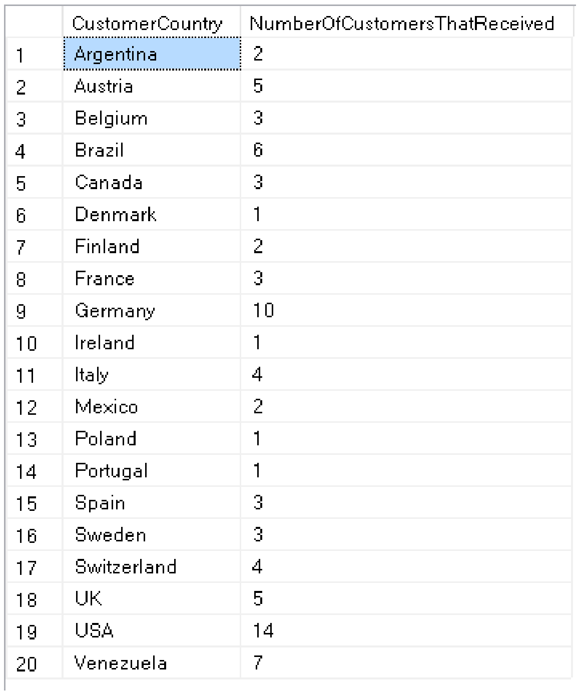
Table Name	Column Name
Sales.Customer	CustomerCountry
Sales.Order	RequiredDate

Order By

Table Name	Column Name
Sales.Customer	CustomerCountry

Table Results

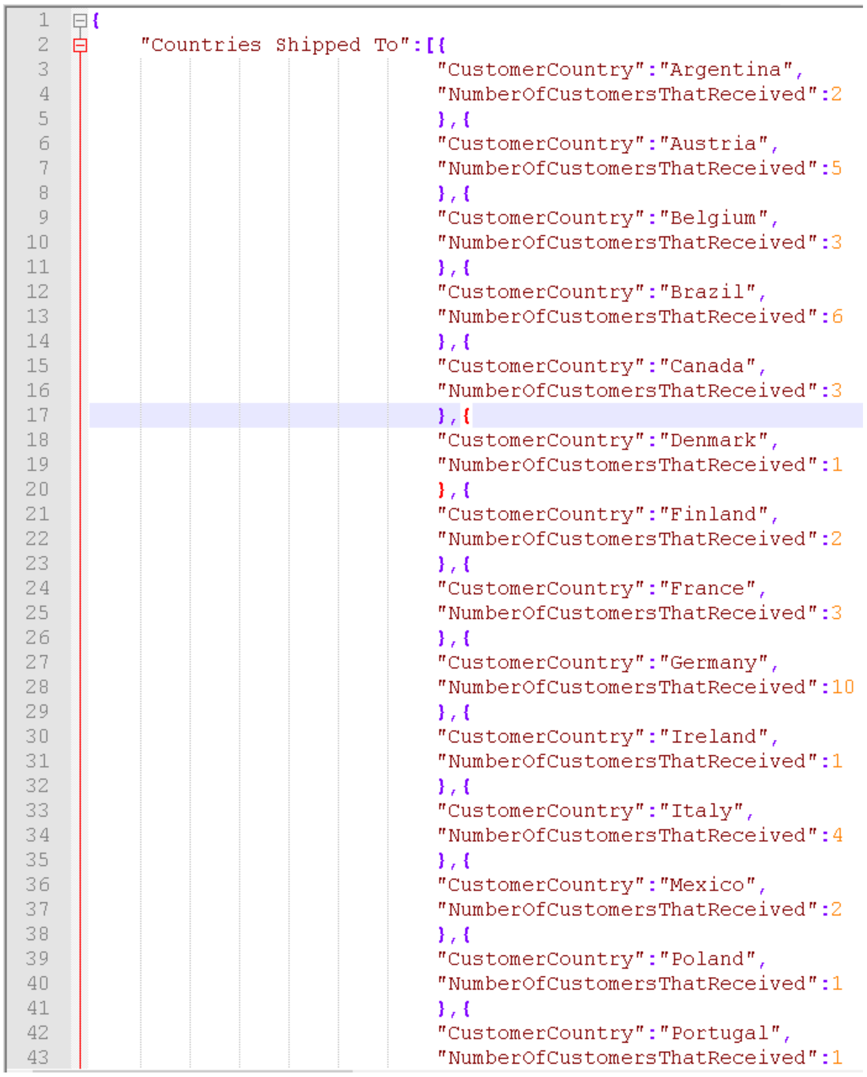
```sql
SELECT
       C.CustomerCountry,
       COUNT(O.RequiredDate) AS NumberOfCustomersThatReceived
FROM Sales.Customer AS C
INNER JOIN Sales.[Order] AS O
ON C.CustomerID = O.CustomerID
WHERE O.RequiredDate >= '20160501' AND O.RequiredDate < '20160901'
GROUP BY C.CustomerCountry
ORDER BY C.CustomerCountry;
```



	CustomerCountry	NumberOfCustomersThatReceived
1	Argentina	2
2	Austria	5
3	Belgium	3
4	Brazil	6
5	Canada	3
6	Denmark	1
7	Finland	2
8	France	3
9	Germany	10
10	Ireland	1
11	Italy	4
12	Mexico	2
13	Poland	1
14	Portugal	1
15	Spain	3
16	Sweden	3
17	Switzerland	4
18	UK	5
19	USA	14
20	Venezuela	7

JSON Results

```
SELECT
       C.CustomerCountry,
       COUNT(O.RequiredDate) AS NumberOfCustomersThatReceived
FROM Sales.Customer AS C
INNER JOIN Sales.[Order] AS O
ON C.CustomerID = O.CustomerID
WHERE O.RequiredDate >= '20160501' AND O.RequiredDate < '20160901'
GROUP BY C.CustomerCountry
ORDER BY C.CustomerCountry
FOR JSON PATH, ROOT('Countries Shipped To');
```



```
{
    "Countries Shipped To":[{
                            "CustomerCountry":"Argentina",
                            "NumberOfCustomersThatReceived":2
                            },{
                            "CustomerCountry":"Austria",
                            "NumberOfCustomersThatReceived":5
                            },{
                            "CustomerCountry":"Belgium",
                            "NumberOfCustomersThatReceived":3
                            },{
                            "CustomerCountry":"Brazil",
                            "NumberOfCustomersThatReceived":6
                            },{
                            "CustomerCountry":"Canada",
                            "NumberOfCustomersThatReceived":3
                            },{
                            "CustomerCountry":"Denmark",
                            "NumberOfCustomersThatReceived":1
                            },{
                            "CustomerCountry":"Finland",
                            "NumberOfCustomersThatReceived":2
                            },{
                            "CustomerCountry":"France",
                            "NumberOfCustomersThatReceived":3
                            },{
                            "CustomerCountry":"Germany",
                            "NumberOfCustomersThatReceived":10
                            },{
                            "CustomerCountry":"Ireland",
                            "NumberOfCustomersThatReceived":1
                            },{
                            "CustomerCountry":"Italy",
                            "NumberOfCustomersThatReceived":4
                            },{
                            "CustomerCountry":"Mexico",
                            "NumberOfCustomersThatReceived":2
                            },{
                            "CustomerCountry":"Poland",
                            "NumberOfCustomersThatReceived":1
                            },{
                            "CustomerCountry":"Portugal",
                            "NumberOfCustomersThatReceived":1
```

Proposition 08: Return all Orders that were placed on the same day as the employee's birthday that was in charge of the order.

The Query above is asking for you to return the number of orders that were placed on the specific employee's birthday in which matched the order date. To start, we will first call upon the Human Resources Employee table since we need to know the date of the employee's birth date. Then, once we have received that, we will call upon the Orders table where we will match the EmployeeID to the Employee table. Once the tables are linked, we will then check to see where the Month and Day of the Order Date and the Birth Date of the Employee and Orders match. If they do, that means that the customer placed the order on the same exact day of the Employee Birth Date. Then, we will simply Group and Sort the information according to the Employee's Birth Date.

Projected Columns

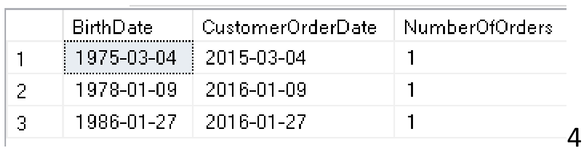
Table Name	**Column Name**
HumanResources.Employee	BirthDate
Sales.Order	OrderDate, OrderID

Order By

Table Name	**Column Name**
HumanResources.Employee	BirthDate

Table Results

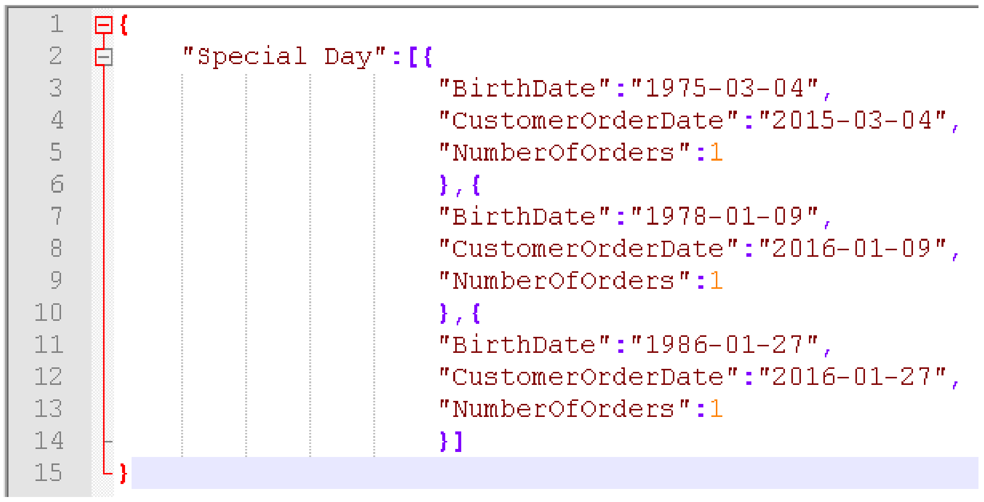
```
SELECT
       E.BirthDate,
       MAX(O.OrderDate) AS CustomerOrderDate,
       COUNT(O.OrderID) AS NumberOfOrders
FROM HumanResources.Employee AS E
INNER JOIN Sales.[Order] AS O
ON E.EmployeeID = O.EmployeeID
WHERE MONTH(E.BirthDate) = MONTH(O.OrderDate) AND DAY(E.BirthDate) = DAY(O.OrderDate)
GROUP BY E.BirthDate
ORDER BY E.BirthDate;
```



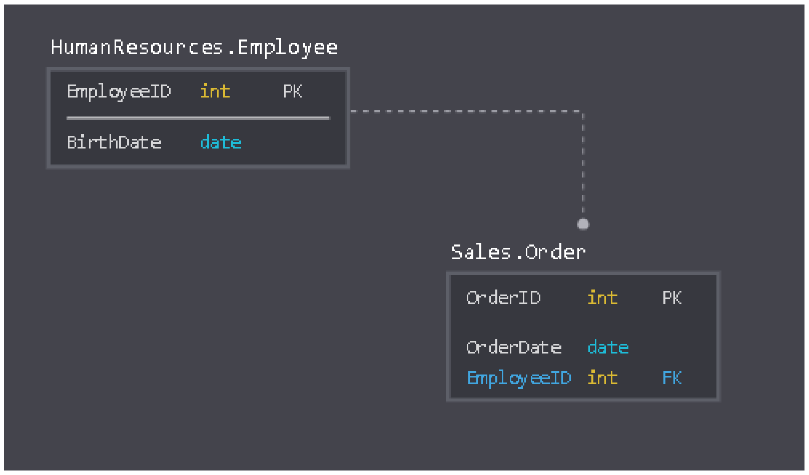
	BirthDate	CustomerOrderDate	NumberOfOrders
1	1975-03-04	2015-03-04	1
2	1978-01-09	2016-01-09	1
3	1986-01-27	2016-01-27	1

JSON Results

```sql
SELECT
       E.BirthDate,
       MAX(O.OrderDate) AS CustomerOrderDate,
       COUNT(O.OrderID) AS NumberOfOrders
FROM HumanResources.Employee AS E
INNER JOIN Sales.[Order] AS O
ON E.EmployeeID = O.EmployeeID
WHERE MONTH(E.BirthDate) = MONTH(O.OrderDate) AND DAY(E.BirthDate) = DAY(O.OrderDate)
GROUP BY E.BirthDate
ORDER BY E.BirthDate
FOR JSON PATH, ROOT('Special Day');
```



```json
{
    "Special Day":[{
                        "BirthDate":"1975-03-04",
                        "CustomerOrderDate":"2015-03-04",
                        "NumberOfOrders":1
                        },{
                        "BirthDate":"1978-01-09",
                        "CustomerOrderDate":"2016-01-09",
                        "NumberOfOrders":1
                        },{
                        "BirthDate":"1986-01-27",
                        "CustomerOrderDate":"2016-01-27",
                        "NumberOfOrders":1
                        }]
}
```



Proposition 09: Return the number of employees in groups that either started their shifts at 7am or 3pm. In addition to that, indicate whether the shift is a Morning, Day, or Evening Shift.

The Query above is asking for you to return the number of employees that work at a certain time of the day, in this case 7am and 3pm. Also, indicate whether the shift is a morning, day or evening shift. First, we will evaluate the EmployeeDepartmentHistory table since this table is responsible for linking the specific employees with their according shifts. Then, we must use the Shift table which is responsible for providing us with the name and start time of the shift. Once we link them based on their ShiftID's, we can then compare to see which queries will return to us the start time of either 7am or 3pm. Once we have found our results, we will group the results by the shift start time.

Projected Columns

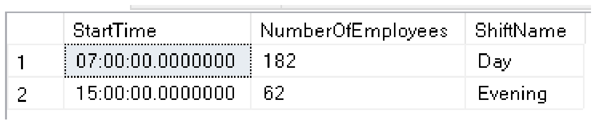
Table Name	Column Name
HumanResources.EmployeeDepartmentHistory	BusinessEntityID
HumanResources.[Shift]	StartTime, Name

Order By

Table Name	Column Name
HumanResources.EmployeeDepartmentHistory	StartTime

Table Results

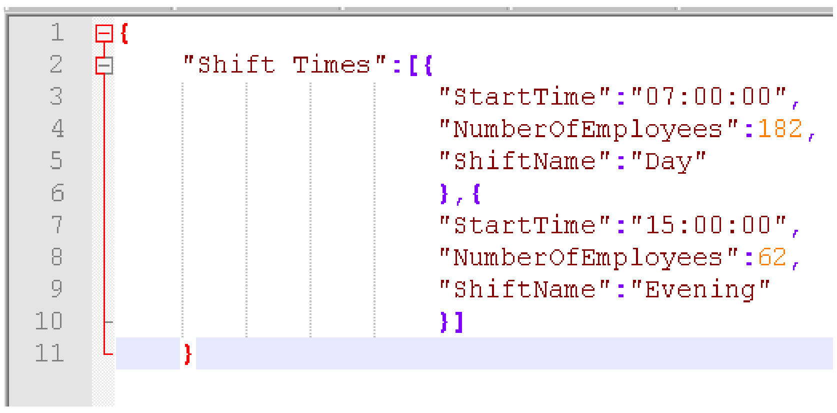
```
SELECT
       S.StartTime,
       COUNT(EDH.BusinessEntityID) AS NumberOfEmployees,
       MAX(S.Name) AS ShiftName
FROM HumanResources.EmployeeDepartmentHistory AS EDH
INNER JOIN HumanResources.[Shift] AS S
ON EDH.ShiftID = S.ShiftID
WHERE S.StartTime = '07:00:00' OR S.StartTime = '15:00:00'
GROUP BY S.StartTime
ORDER BY S.StartTime;
```



	StartTime	NumberOfEmployees	ShiftName
1	07:00:00.0000000	182	Day
2	15:00:00.0000000	62	Evening

JSON Results

```
SELECT
       S.StartTime,
       COUNT(EDH.BusinessEntityID) AS NumberOfEmployees,
       MAX(S.Name) AS ShiftName
FROM HumanResources.EmployeeDepartmentHistory AS EDH
INNER JOIN HumanResources.[Shift] AS S
ON EDH.ShiftID = S.ShiftID
WHERE S.StartTime = '07:00:00' OR S.StartTime = '15:00:00'
GROUP BY S.StartTime
ORDER BY S.StartTime
FOR JSON PATH, ROOT('Shift Times');
```



```
{
    "Shift Times":[{
                    "StartTime":"07:00:00",
                    "NumberOfEmployees":182,
                    "ShiftName":"Day"
                    },{
                    "StartTime":"15:00:00",
                    "NumberOfEmployees":62,
                    "ShiftName":"Evening"
                    }]
}
```

Proposition 10: Return all products with their name, ID, price and cost that are above 100 dollars and have been sold for more than 6 months

The Query above is asking for you to find all products that cost more than 100 dollars and to return the name of the product, ID, price and cost. First, you have to use the Production Product table in order to pull the List Price and Standard Cost. Once you have done that, in order to project these attributes, we will use the MAX function on the Product Model ID since the same product has the same ID and we will use an AVG function for the List Price so that we get the price that it has been given along all its models. In addition to that, we use the MAX function to get the highest cost that it may take to produce a certain product. Once you have done all of that, you will need to join the Product Model table as well in order to retrieve the Model ID as well as the Model Name. At this point, the only thing we need to check for 2 conditions: First, we check if the list price of the item is greater than 100 dollars. Then, we check to see if the period between the time the product started selling and stopped selling is over 6 months. Lastly, order them by the Product Models Name.

Projected Columns

Table Name	Column Name
Production.Product	ListPrice, StandardCost
Production.ProductModel	ProductModelID

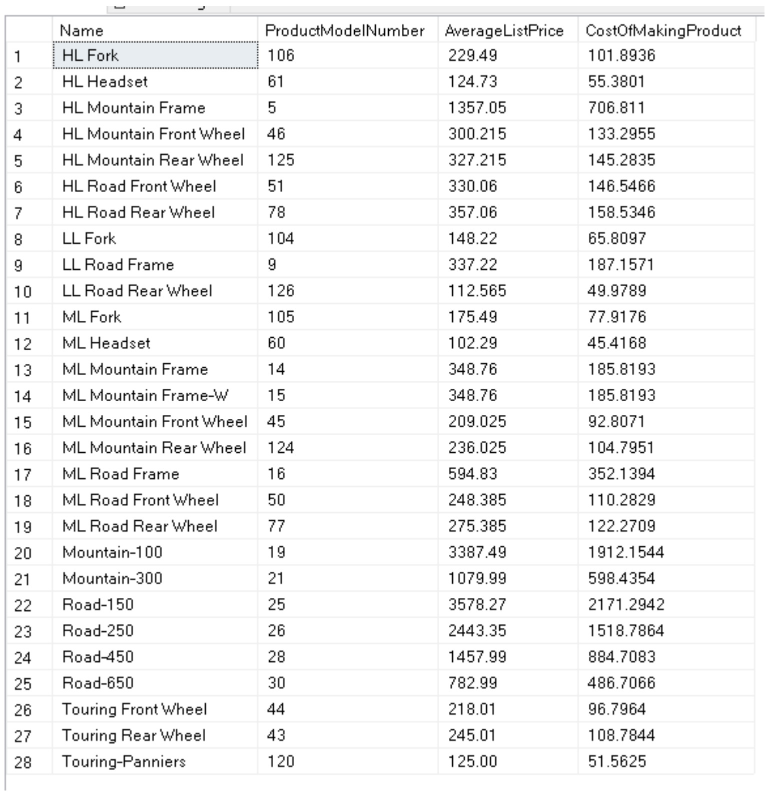
Order By

Table Name	Column Name
Production.Product	Name

Table Results

```
USE AdventureWorks2014;
GO
```

```
SELECT
	PM.Name,
	MAX(PM.ProductModelID) AS ProductModelNumber,
	AVG(P.ListPrice) AS AverageListPrice,
	MAX(P.StandardCost) AS CostOfMakingProduct
FROM Production.Product AS P
INNER JOIN Production.ProductModel AS PM
ON P.ProductModelID = PM.ProductModelID
WHERE P.ListPrice > 100 AND DATEDIFF(Month, P.SellStartDate, P.SellEndDate) > 6
GROUP BY PM.Name
ORDER BY PM.Name;
```

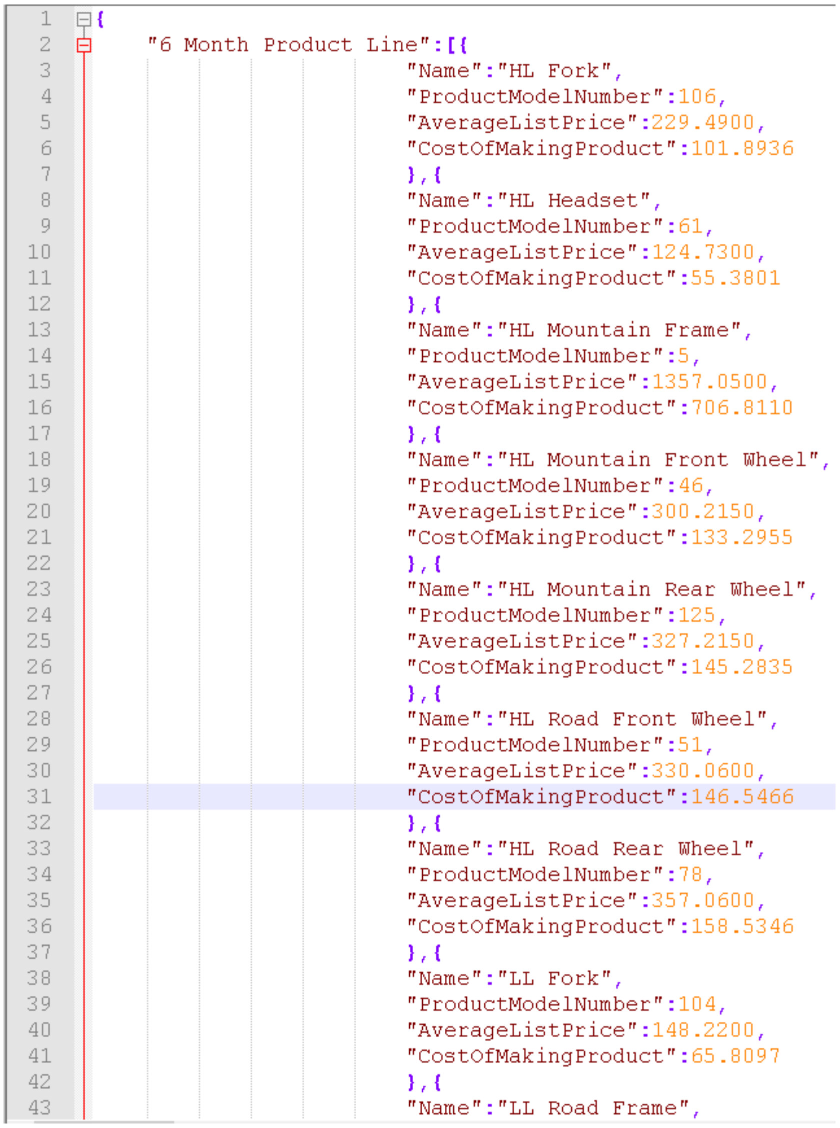


	Name	ProductModelNumber	AverageListPrice	CostOfMakingProduct
1	HL Fork	106	229.49	101.8936
2	HL Headset	61	124.73	55.3801
3	HL Mountain Frame	5	1357.05	706.811
4	HL Mountain Front Wheel	46	300.215	133.2955
5	HL Mountain Rear Wheel	125	327.215	145.2835
6	HL Road Front Wheel	51	330.06	146.5466
7	HL Road Rear Wheel	78	357.06	158.5346
8	LL Fork	104	148.22	65.8097
9	LL Road Frame	9	337.22	187.1571
10	LL Road Rear Wheel	126	112.565	49.9789
11	ML Fork	105	175.49	77.9176
12	ML Headset	60	102.29	45.4168
13	ML Mountain Frame	14	348.76	185.8193
14	ML Mountain Frame-W	15	348.76	185.8193
15	ML Mountain Front Wheel	45	209.025	92.8071
16	ML Mountain Rear Wheel	124	236.025	104.7951
17	ML Road Frame	16	594.83	352.1394
18	ML Road Front Wheel	50	248.385	110.2829
19	ML Road Rear Wheel	77	275.385	122.2709
20	Mountain-100	19	3387.49	1912.1544
21	Mountain-300	21	1079.99	598.4354
22	Road-150	25	3578.27	2171.2942
23	Road-250	26	2443.35	1518.7864
24	Road-450	28	1457.99	884.7083
25	Road-650	30	782.99	486.7066
26	Touring Front Wheel	44	218.01	96.7964
27	Touring Rear Wheel	43	245.01	108.7844
28	Touring-Panniers	120	125.00	51.5625

JSON Results

```
USE AdventureWorks2014;
GO

SELECT
	PM.Name,
	MAX(PM.ProductModelID) AS ProductModelNumber,
	AVG(P.ListPrice) AS AverageListPrice,
	MAX(P.StandardCost) AS CostOfMakingProduct
FROM Production.Product AS P
INNER JOIN Production.ProductModel AS PM
ON P.ProductModelID = PM.ProductModelID
WHERE P.ListPrice > 100 AND DATEDIFF(Month, P.SellStartDate, P.SellEndDate) > 6
GROUP BY PM.Name
ORDER BY PM.Name
```

```
FOR JSON PATH, ROOT('6 Month Product Line');
```



```
{
    "6 Month Product Line":[{
                    "Name":"HL Fork",
                    "ProductModelNumber":106,
                    "AverageListPrice":229.4900,
                    "CostOfMakingProduct":101.8936
                    },{
                    "Name":"HL Headset",
                    "ProductModelNumber":61,
                    "AverageListPrice":124.7300,
                    "CostOfMakingProduct":55.3801
                    },{
                    "Name":"HL Mountain Frame",
                    "ProductModelNumber":5,
                    "AverageListPrice":1357.0500,
                    "CostOfMakingProduct":706.8110
                    },{
                    "Name":"HL Mountain Front Wheel",
                    "ProductModelNumber":46,
                    "AverageListPrice":300.2150,
                    "CostOfMakingProduct":133.2955
                    },{
                    "Name":"HL Mountain Rear Wheel",
                    "ProductModelNumber":125,
                    "AverageListPrice":327.2150,
                    "CostOfMakingProduct":145.2835
                    },{
                    "Name":"HL Road Front Wheel",
                    "ProductModelNumber":51,
                    "AverageListPrice":330.0600,
                    "CostOfMakingProduct":146.5466
                    },{
                    "Name":"HL Road Rear Wheel",
                    "ProductModelNumber":78,
                    "AverageListPrice":357.0600,
                    "CostOfMakingProduct":158.5346
                    },{
                    "Name":"LL Fork",
                    "ProductModelNumber":104,
                    "AverageListPrice":148.2200,
                    "CostOfMakingProduct":65.8097
                    },{
                    "Name":"LL Road Frame",
```

Proposition 11: Return the amount of businesses that use a vista credit card type

The Query above is asking for you to return the number of businesses that use each type of Credit Card. In order to do this, we need to first pull information from the Credit Card Table since this will provide us with the Credit Card Type. In addition to that, we will also need to use the Person Credit Credit table to get the BusinessEntityID which will provide us with the businesses that have allowed the person to purchase from them. Therefore, once we have joined the two tables based on their CreditCardID's, we will now just need to check and see which CreditCards had a type of 'Vista', which will lead to our desired results.

Projected Columns

Table Name	Column Name
Sales.PersonCreditCard	BusinessEntityID

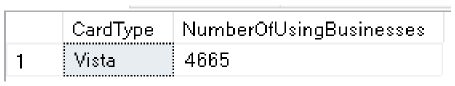
Sales.CreditCard	CardType

Order By

Table Name	Column Name
Sales.CreditCard	CardType

Table Results

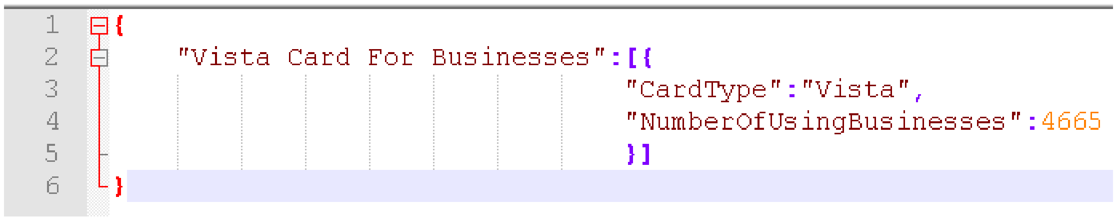
```
USE AdventureWorks2014;
GO
SELECT
	C.CardType,
	COUNT(PCC.BusinessEntityID) AS NumberOfUsingBusinesses
FROM Sales.PersonCreditCard AS PCC
INNER JOIN Sales.CreditCard AS C
ON C.CreditCardID = PCC.CreditCardID
WHERE C.CardType = 'Vista'
GROUP BY C.CardType
ORDER BY C.CardType;
```



	CardType	NumberOfUsingBusinesses
1	Vista	4665

Table Results

```
USE AdventureWorks2014;
GO
SELECT
	C.CardType,
	COUNT(PCC.BusinessEntityID) AS NumberOfUsingBusinesses
FROM Sales.PersonCreditCard AS PCC
INNER JOIN Sales.CreditCard AS C
ON C.CreditCardID = PCC.CreditCardID
WHERE C.CardType = 'Vista'
GROUP BY C.CardType
ORDER BY C.CardType
FOR JSON PATH, ROOT('Vista Card For Businesses');
```



```
{
    "Vista Card For Businesses":[{
                                  "CardType":"Vista",
                                  "NumberOfUsingBusinesses":4665
                                  }]
}
```

Proposition 12: Proposition: Return all products that were on a discount of more than 10% that did no end their special offer until before 2013 and were more expensive than 200 dollars.

The Query above is asking for you to find and return all products that were on a discount of more than 10% and lasted longer than the year 2013. Also, you were to make sure that the product cost more than 200 dollars. First, we start with projecting the specific attributes(columns) that we desire: DiscountPct, EndDate, ProductName, and list price of the product itself. In order to do this, we must start by analyzing the specific tables that we are going to need to use. First, we can see that we are going to need the Special Offer table to find the discount percentage on each of the products as well as the day the discount ends. Secondly, we must use the Special Offer Product table in order to link the Product table to the Special Offer table. Just mentioned, we must also use the Product table in order to find the number of products under the specific discount as well as the List Price. Once you have joined all tables based on their according keys, we must make sure to check for 3 different conditions: First being that the discount percentage is more than 10%, the end date is after 2013 and that the list price is greater than 200 dollars. Finally, organized the query by grouping and ordering by the discount percentage.

Projected Columns

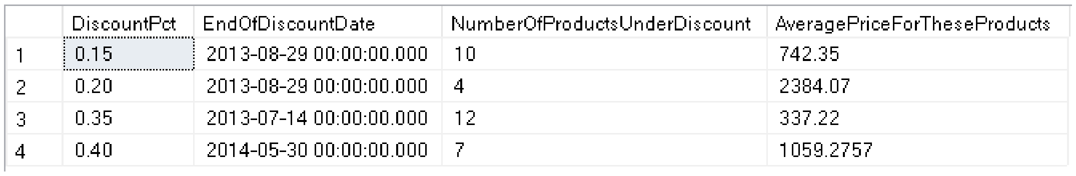
Table Name	**Column Name**
Sales.SpecialOfferProduct	N/A
Sales.SpecialOffer	EndDate, DiscountPct
Production.Product	Name, ListPrice

Order By

Table Name	**Column Name**
Sales.SpecialOffer	DiscountPct

Table Results

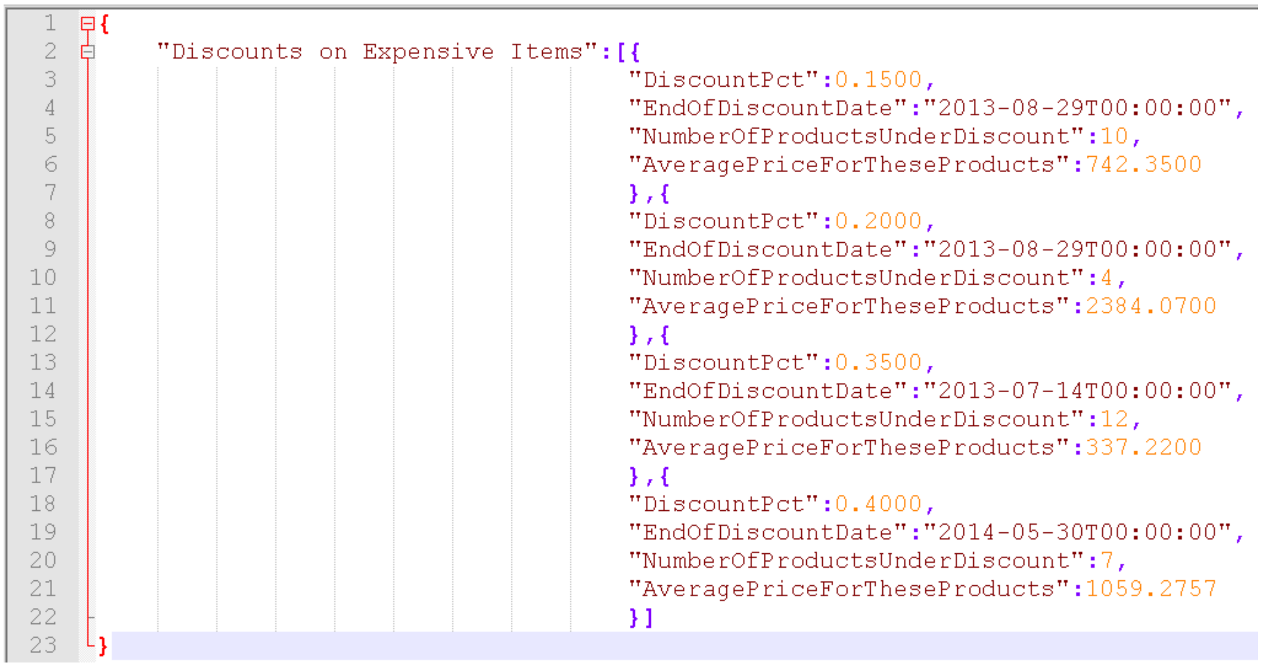
```
SELECT
       SO.DiscountPct,
       MAX(SO.EndDate) EndOfDiscountDate,
       COUNT(P.[Name]) AS NumberOfProductsUnderDiscount,
       AVG(P.ListPrice) AS AveragePriceForTheseProducts
FROM Sales.SpecialOfferProduct AS SOP
INNER JOIN Sales.SpecialOffer AS SO
       ON SOP.SpecialOfferID = SO.SpecialOfferID
INNER JOIN Production.Product AS P
       ON SOP.ProductID = P.ProductID
WHERE SO.DiscountPct > 0.10
AND SO.EndDate >= '20130101' AND P.ListPrice > 200
GROUP BY SO.DiscountPct
ORDER BY SO.DiscountPct;
```



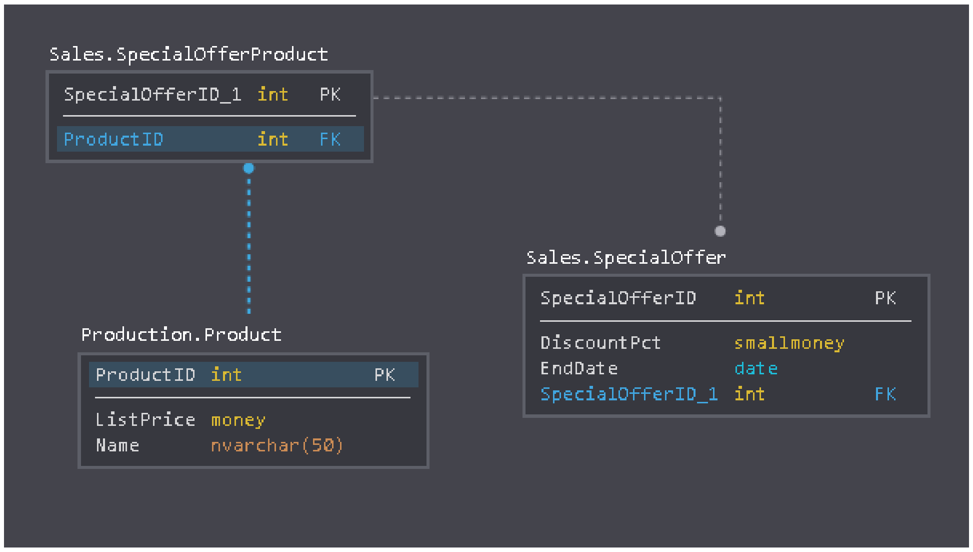
	DiscountPct	EndOfDiscountDate	NumberOfProductsUnderDiscount	AveragePriceForTheseProducts
1	0.15	2013-08-29 00:00:00.000	10	742.35
2	0.20	2013-08-29 00:00:00.000	4	2384.07
3	0.35	2013-07-14 00:00:00.000	12	337.22
4	0.40	2014-05-30 00:00:00.000	7	1059.2757

JSON Results

```sql
USE AdventureWorks2014;
GO
SELECT
	SO.DiscountPct,
	MAX(SO.EndDate) EndOfDiscountDate,
	COUNT(P.[Name]) AS NumberOfProductsUnderDiscount,
	AVG(P.ListPrice) AS AveragePriceForTheseProducts
FROM Sales.SpecialOfferProduct AS SOP
INNER JOIN Sales.SpecialOffer AS SO
	ON SOP.SpecialOfferID = SO.SpecialOfferID
INNER JOIN Production.Product AS P
	ON SOP.ProductID = P.ProductID
WHERE SO.DiscountPct > 0.10
AND SO.EndDate >= '20130101' AND P.ListPrice > 200
GROUP BY SO.DiscountPct
ORDER BY SO.DiscountPct
FOR JSON PATH, ROOT('Discounts on Expensive Items');
```



```json
{
    "Discounts on Expensive Items":[{
                                    "DiscountPct":0.1500,
                                    "EndOfDiscountDate":"2013-08-29T00:00:00",
                                    "NumberOfProductsUnderDiscount":10,
                                    "AveragePriceForTheseProducts":742.3500
                                    },{
                                    "DiscountPct":0.2000,
                                    "EndOfDiscountDate":"2013-08-29T00:00:00",
                                    "NumberOfProductsUnderDiscount":4,
                                    "AveragePriceForTheseProducts":2384.0700
                                    },{
                                    "DiscountPct":0.3500,
                                    "EndOfDiscountDate":"2013-07-14T00:00:00",
                                    "NumberOfProductsUnderDiscount":12,
                                    "AveragePriceForTheseProducts":337.2200
                                    },{
                                    "DiscountPct":0.4000,
                                    "EndOfDiscountDate":"2014-05-30T00:00:00",
                                    "NumberOfProductsUnderDiscount":7,
                                    "AveragePriceForTheseProducts":1059.2757
                                    }]
}
```



Proposition 13: Return the total number of orders that were shipped to each of the Territories where the subtotal of the order was over 20,000.

The Query above is asking for you to return the number of orders that are to be shipped to each territory where the subtotal of the order is greater than 20000. Like always, we start by identifying which tables are going to be necessary in order to solve the query. For this case, we are definitely going to need to use the SalesOrderHeader table since that table holds information such as the SalesOrderId used to find total orders as well as the Subtotal of the order. Now, link the tables together by comparing their territoryID's. After that, check to see if the subtotal that you have retrieved from the SalesTerritory table is greater than 20,000 dollars. Once that is done, you may now order and group the information by the Territory name.

Projected Columns

Table Name	Column Name
Sales.SalesOrderHeader	SalesOrderID, SubTotal
Sales.SalesTerritory	Name

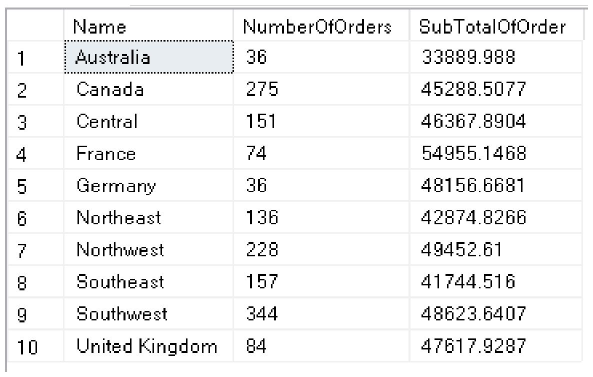
Order By

Table Name	Column Name
Sales.SalesTerritory	Name

Table Results

```
SELECT
       ST.[Name],
       COUNT(SOH.SalesOrderID) AS NumberOfOrders,
       AVG(SOH.SubTotal) AS SubTotalOfOrder
FROM Sales.SalesOrderHeader AS SOH
```

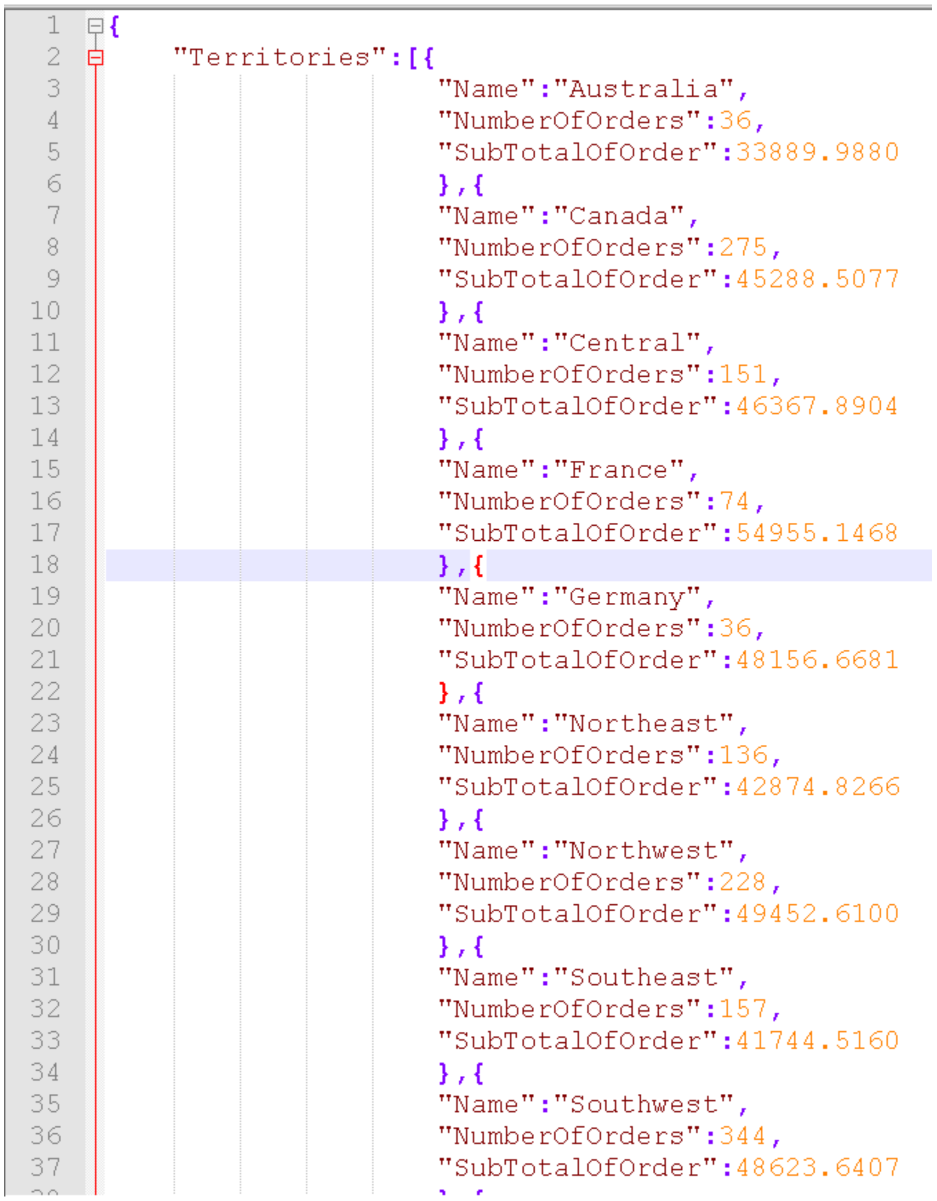
```
INNER JOIN Sales.SalesTerritory AS ST
ON SOH.TerritoryID = ST.TerritoryID
WHERE SOH.SubTotal > 20000
GROUP BY ST.[Name]
ORDER BY ST.[Name];
```



	Name	NumberOfOrders	SubTotalOfOrder
1	Australia	36	33889.988
2	Canada	275	45288.5077
3	Central	151	46367.8904
4	France	74	54955.1468
5	Germany	36	48156.6681
6	Northeast	136	42874.8266
7	Northwest	228	49452.61
8	Southeast	157	41744.516
9	Southwest	344	48623.6407
10	United Kingdom	84	47617.9287

JSON Results

```
SELECT
       ST.[Name],
       COUNT(SOH.SalesOrderID) AS NumberOfOrders,
       AVG(SOH.SubTotal) AS SubTotalOfOrder
FROM Sales.SalesOrderHeader AS SOH
INNER JOIN Sales.SalesTerritory AS ST
ON SOH.TerritoryID = ST.TerritoryID
WHERE SOH.SubTotal > 20000
GROUP BY ST.[Name]
ORDER BY ST.[Name]
FOR JSON PATH, ROOT('Territories');
```



```
{
    "Territories":[{
                        "Name":"Australia",
                        "NumberOfOrders":36,
                        "SubTotalOfOrder":33889.9880
                        },{
                        "Name":"Canada",
                        "NumberOfOrders":275,
                        "SubTotalOfOrder":45288.5077
                        },{
                        "Name":"Central",
                        "NumberOfOrders":151,
                        "SubTotalOfOrder":46367.8904
                        },{
                        "Name":"France",
                        "NumberOfOrders":74,
                        "SubTotalOfOrder":54955.1468
                        },{
                        "Name":"Germany",
                        "NumberOfOrders":36,
                        "SubTotalOfOrder":48156.6681
                        },{
                        "Name":"Northeast",
                        "NumberOfOrders":136,
                        "SubTotalOfOrder":42874.8266
                        },{
                        "Name":"Northwest",
                        "NumberOfOrders":228,
                        "SubTotalOfOrder":49452.6100
                        },{
                        "Name":"Southeast",
                        "NumberOfOrders":157,
                        "SubTotalOfOrder":41744.5160
                        },{
                        "Name":"Southwest",
                        "NumberOfOrders":344,
                        "SubTotalOfOrder":48623.6407
```

Proposition 14: Return the lowest spending 5 customers that either live in area 10038 or London that have purchased the lowest amount of product and placed their order before April 22nd, 2016.

The Query above is asking you to return the lowest 5 customers that either live in are 10038 or London that have purchased the lowest amount of product. First, we must connect the appropriate tables starting with the Sales.Order table. Also, we need to use the Sales.OrderDetails table since we can link it to the Order table based on the OrderID. Next we are also going to also need the Sales.Customer table since that table will provide use with the CustomerID. Once we have connected all tables, we then need to check conditions to make sure that the information provides us where the CustomerCity is either = London or the Postal Code 10038. Once the information is in ascending order, all you have to do is select the top 5.

Projected Columns

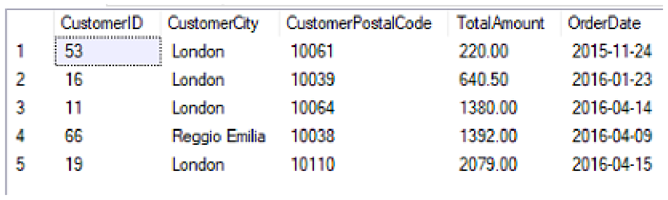
Table Name	Column Name
Sales.OrderDetail	OD.Quantity, OD.UnitPrice
Sales.Order	N/A
Sales.Customer	CustomerID, CustomerCity, CustomerPostalCode

Order By

Table Name	Column Name
Sales.OrderDetail	TotalAmount

Table Results

```
--Function
CREATE FUNCTION [Sales].[getTotalPriceOfOrder](@Quantity INT, @UnitPrice FLOAT)
RETURNS DECIMAL(10,2)
BEGIN
	DECLARE @numberOfOrders DECIMAL(10,2)
	SET @numberOfOrders = (@Quantity * @UnitPrice)
	RETURN @numberOfOrders
END
GO
--Query
SELECT TOP 5
		C.CustomerID,
		MAX(C.CustomerCity) AS CustomerCity,
		C.CustomerPostalCode,
		MAX([Sales].[getTotalPriceOfOrder](OD.Quantity, OD.UnitPrice)) AS
TotalAmount,
		MAX(O.OrderDate) AS OrderDate
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
	ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
	ON O.CustomerId = C.CustomerId
WHERE (C.CustomerPostalCode = '10038' OR C.CustomerCity = 'London') AND O.OrderDate <
'20160422'
GROUP BY C.CustomerID, C.CustomerPostalCode
ORDER BY TotalAmount ASC;
```



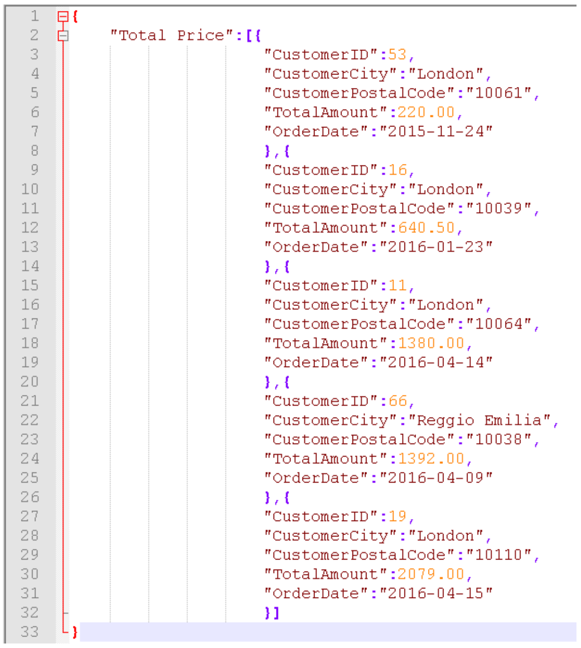
	CustomerID	CustomerCity	CustomerPostalCode	TotalAmount	OrderDate
1	53	London	10061	220.00	2015-11-24
2	16	London	10039	640.50	2016-01-23
3	11	London	10064	1380.00	2016-04-14
4	66	Reggio Emilia	10038	1392.00	2016-04-09
5	19	London	10110	2079.00	2016-04-15

JSON Results

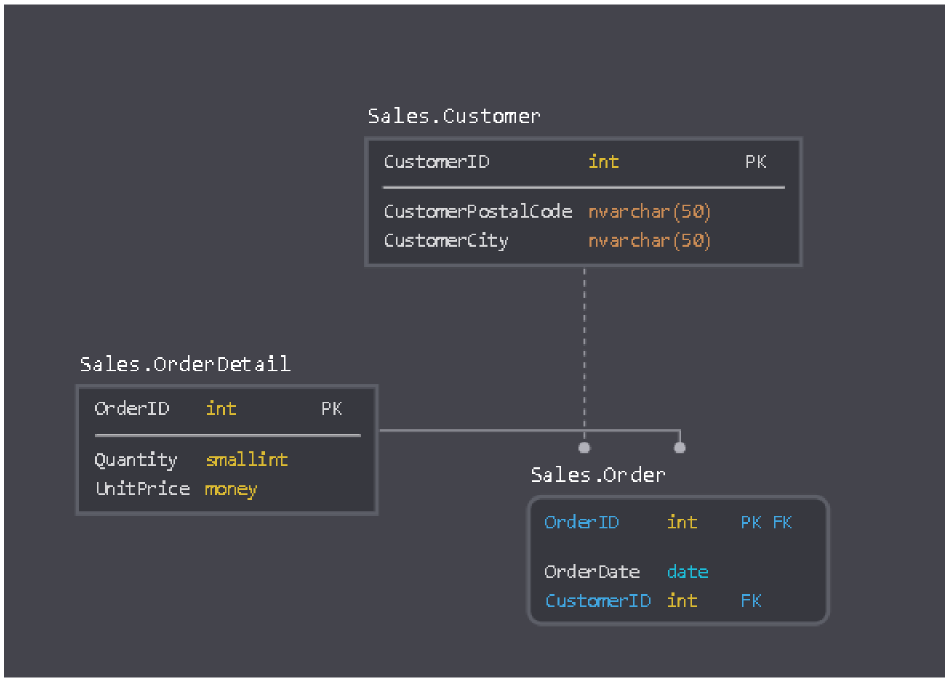
Prepared by: Tristen Aguilar

Date Prepared: 3/19/2020

```
--Query with JSON
SELECT TOP 5
              C.CustomerID,
              MAX(C.CustomerCity) AS CustomerCity,
              C.CustomerPostalCode,
              MAX([Sales].[getTotalPriceOfOrder](OD.Quantity, OD.UnitPrice)) AS
TotalAmount, --Use function
              MAX(O.OrderDate) AS OrderDate
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
WHERE (C.CustomerPostalCode = '10038' OR C.CustomerCity = 'London') AND O.OrderDate <
'20160422'
GROUP BY C.CustomerID, C.CustomerPostalCode
ORDER BY TotalAmount ASC
FOR JSON PATH, ROOT('Total Price');
```



```
{
    "Total Price":[{
                "CustomerID":53,
                "CustomerCity":"London",
                "CustomerPostalCode":"10061",
                "TotalAmount":220.00,
                "OrderDate":"2015-11-24"
                },{
                "CustomerID":16,
                "CustomerCity":"London",
                "CustomerPostalCode":"10039",
                "TotalAmount":640.50,
                "OrderDate":"2016-01-23"
                },{
                "CustomerID":11,
                "CustomerCity":"London",
                "CustomerPostalCode":"10064",
                "TotalAmount":1380.00,
                "OrderDate":"2016-04-14"
                },{
                "CustomerID":66,
                "CustomerCity":"Reggio Emilia",
                "CustomerPostalCode":"10038",
                "TotalAmount":1392.00,
                "OrderDate":"2016-04-09"
                },{
                "CustomerID":19,
                "CustomerCity":"London",
                "CustomerPostalCode":"10110",
                "TotalAmount":2079.00,
                "OrderDate":"2016-04-15"
                }]
}
```



Proposition 15: Return the highest spending 15 customers that either live in the USA or have a phone number that starts with 5 and placed their order before the new year of 2016.

The Query above is asking you to return the highest spending 15 customers that either live in the USA or have a phone number that starts with 5 and placed their order before the new year of 2016. We know that the tables necessary to solve this query are the Sales.OrderDetails, Sales.Order and Sales.Customer tables. We know that because we need the CustomerID, Country and CustomerPhoneNumber from the Customer table, the Total amount which comes about from the OrderDetails Table. Lastly, we need the Order table in order to track the OrderDate. Once all of these tables have been connected, we then check for the specific conditions which are: The Customer Country is the USA or the first 2 characters of the Phone number starts witha '(5'. Finally, we also check to see if the OrderDate is before the 2016 new year.

Projected Columns

Table Name	**Column Name**
Sales.OrderDetail	OD.Quantity, OD.UnitPrice
Sales.Order	N/A
Sales.Customer	CustomerID, CustomerCountry, CustomerPhoneNumber

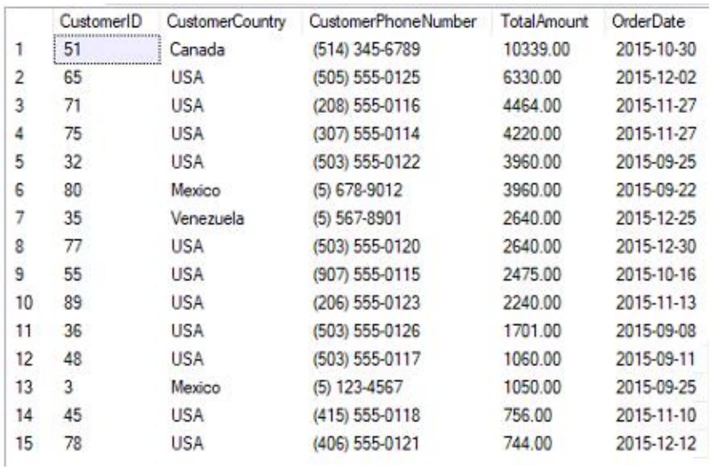
Order By

Table Name	**Column Name**
Sales.OrderDetail	TotalAmount

Prepared by: Tristen Aguilar Date Prepared: 3/19/2020

Table Results

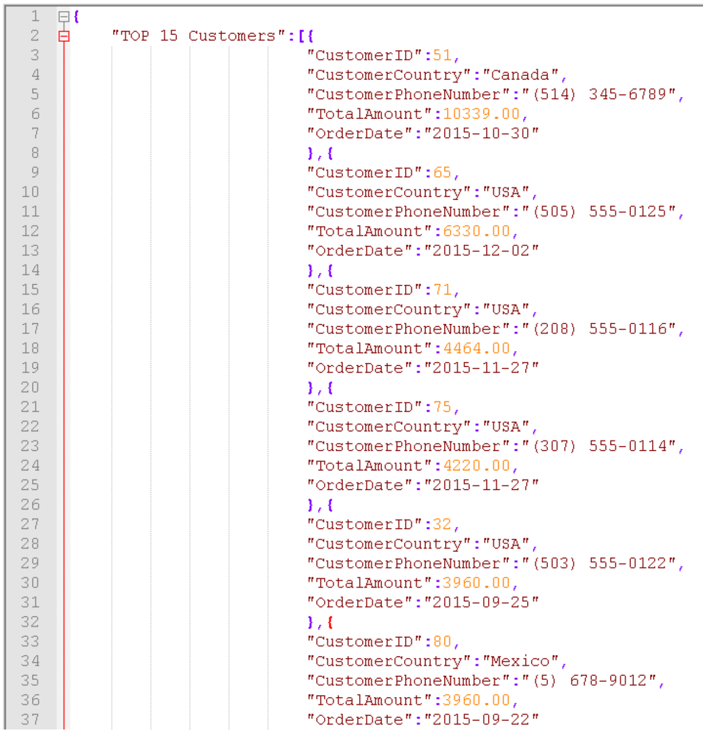
```
--Function
CREATE FUNCTION [Sales].[getTotalPriceOfOrder](@Quantity INT, @UnitPrice DECIMAL)
RETURNS DECIMAL(10,2)
BEGIN
       DECLARE @numberOfOrders DECIMAL(10,2)
       SET @numberOfOrders = (@Quantity * @UnitPrice)
       RETURN @numberOfOrders
END
GO

--Query
SELECT TOP 15
              C.CustomerID,
              MAX(C.CustomerCountry) AS CustomerCountry,
              MAX(C.CustomerPhoneNumber) AS CustomerPhoneNumber,
              MAX([Sales].[getTotalPriceOfOrder](OD.Quantity, OD.UnitPrice)) AS
TotalAmount,
              MAX(O.OrderDate) AS OrderDate
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
WHERE (C.CustomerCountry = 'USA' OR SUBSTRING(C.CustomerPhoneNumber, 1, 2) = '(5') AND
O.OrderDate < '20160101'
GROUP BY C.CustomerID, C.CustomerPostalCode
ORDER BY TotalAmount DESC;
```



	CustomerID	CustomerCountry	CustomerPhoneNumber	TotalAmount	OrderDate
1	51	Canada	(514) 345-6789	10339.00	2015-10-30
2	65	USA	(505) 555-0125	6330.00	2015-12-02
3	71	USA	(208) 555-0116	4464.00	2015-11-27
4	75	USA	(307) 555-0114	4220.00	2015-11-27
5	32	USA	(503) 555-0122	3960.00	2015-09-25
6	80	Mexico	(5) 678-9012	3960.00	2015-09-22
7	35	Venezuela	(5) 567-8901	2640.00	2015-12-25
8	77	USA	(503) 555-0120	2640.00	2015-12-30
9	55	USA	(907) 555-0115	2475.00	2015-10-16
10	89	USA	(206) 555-0123	2240.00	2015-11-13
11	36	USA	(503) 555-0126	1701.00	2015-09-08
12	48	USA	(503) 555-0117	1060.00	2015-09-11
13	3	Mexico	(5) 123-4567	1050.00	2015-09-25
14	45	USA	(415) 555-0118	756.00	2015-11-10
15	78	USA	(406) 555-0121	744.00	2015-12-12

JSON Results

```
--Query
SELECT TOP 15
              C.CustomerID,
              MAX(C.CustomerCountry) AS CustomerCountry,
              MAX(C.CustomerPhoneNumber) AS CustomerPhoneNumber,
              MAX([Sales].[getTotalPriceOfOrder](OD.Quantity, OD.UnitPrice)) AS
TotalAmount,
              MAX(O.OrderDate) AS OrderDate
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
WHERE (C.CustomerCountry = 'USA' OR SUBSTRING(C.CustomerPhoneNumber, 1, 2) = '(5') AND
O.OrderDate < '20160101'
GROUP BY C.CustomerID, C.CustomerPostalCode
ORDER BY TotalAmount DESC
FOR JSON PATH, ROOT('TOP 15 Spending');
```



```
{
    "TOP 15 Customers":[{
                    "CustomerID":51,
                    "CustomerCountry":"Canada",
                    "CustomerPhoneNumber":"(514) 345-6789",
                    "TotalAmount":10339.00,
                    "OrderDate":"2015-10-30"
                    },{
                    "CustomerID":65,
                    "CustomerCountry":"USA",
                    "CustomerPhoneNumber":"(505) 555-0125",
                    "TotalAmount":6330.00,
                    "OrderDate":"2015-12-02"
                    },{
                    "CustomerID":71,
                    "CustomerCountry":"USA",
                    "CustomerPhoneNumber":"(208) 555-0116",
                    "TotalAmount":4464.00,
                    "OrderDate":"2015-11-27"
                    },{
                    "CustomerID":75,
                    "CustomerCountry":"USA",
                    "CustomerPhoneNumber":"(307) 555-0114",
                    "TotalAmount":4220.00,
                    "OrderDate":"2015-11-27"
                    },{
                    "CustomerID":32,
                    "CustomerCountry":"USA",
                    "CustomerPhoneNumber":"(503) 555-0122",
                    "TotalAmount":3960.00,
                    "OrderDate":"2015-09-25"
                    },{
                    "CustomerID":80,
                    "CustomerCountry":"Mexico",
                    "CustomerPhoneNumber":"(5) 678-9012",
                    "TotalAmount":3960.00,
                    "OrderDate":"2015-09-22"
```

Proposition 16: Return the top 30 number of customers that placed an order with 20 or less days to be delivered. Also, project the discount rate at which these customers purchased at.

The Query above is asking you to return the top 30 number of customers that placed an order with 20 or less days to be delivered. Also, project the discount rate at which these customers purchased at. To begin, we first have to see what tables we need to join. Since it is asking for the numebr of customers, we know we are going to need to utilize the CustomerID. Then, we also see that we need to know how many days are in between the day the order was placed it was required to ship. Lastly, we need to know the discount percentage that were held under the order. Once we have linked all the tables based on their CustomerID's and OrderID's, we can know we need to use the customer function created. When I created the function, I used the DATEDIFF Functions in order to find the number of days between both the Order date and the Required Date. After that, we must use the functions to see if it returns a number less than 20.

Projected Columns

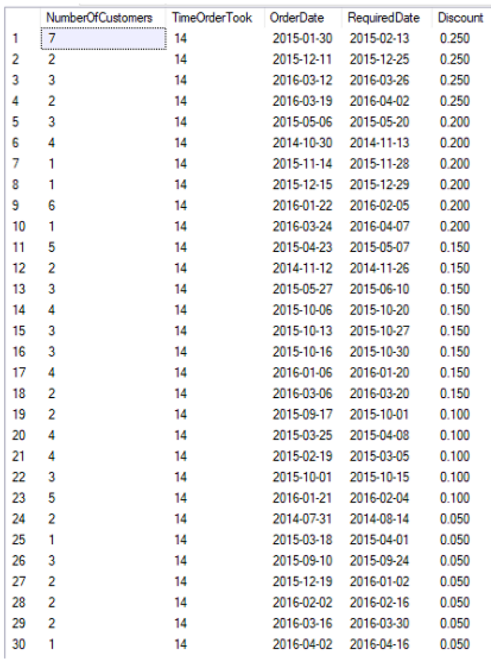
Table Name	Column Name
Sales.OrderDetail	DiscountPercentage
Sales.Order	OrderDate, RequiredDate
Sales.Customer	CustomerID

Order By

Table Name	Column Name
Sales.OrderDetail	DiscountPercentage

Table Results

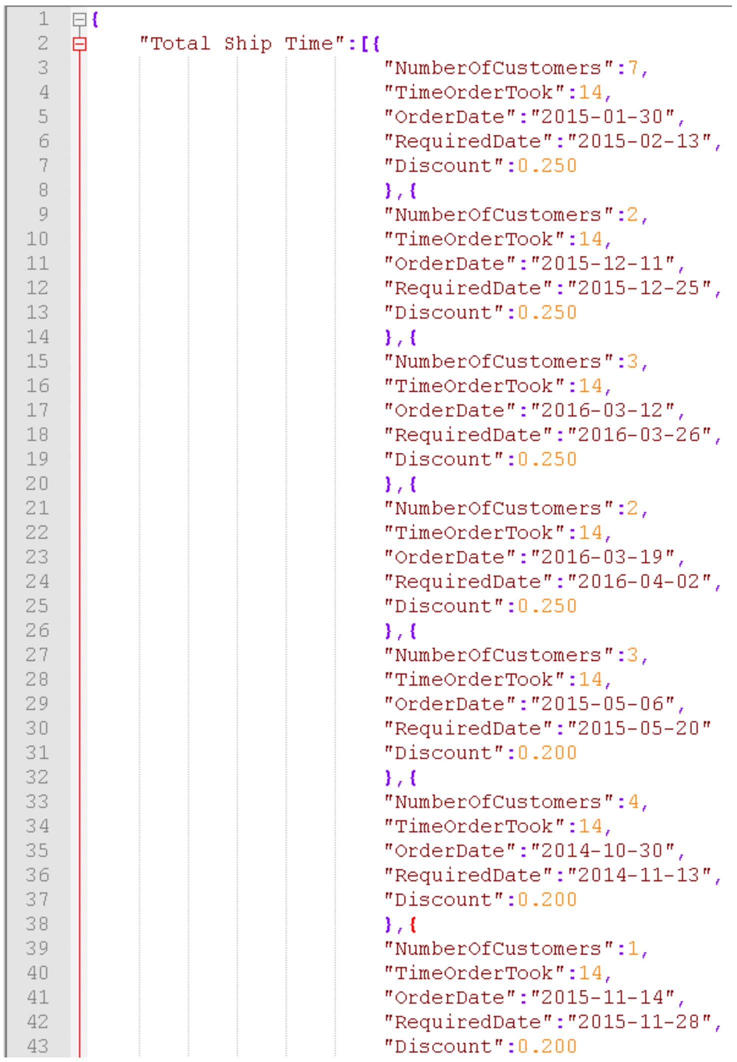
```
--Function
CREATE FUNCTION [Sales].[getTotalShipTime](@OrderDate DATE, @RequiredDate DATE)
RETURNS INT
BEGIN
       DECLARE @numberOfDays INT
       SET @numberOfDays = DATEDIFF(day, @OrderDate, @RequiredDate)
       RETURN @numberOfDays
END
GO

--Query
SELECT TOP 30
              COUNT(C.CustomerID) NumberOfCustomers,
              ([Sales].[getTotalShipTime](O.OrderDate, O.RequiredDate)) AS TimeOrderTook,
              MAX(O.OrderDate) AS OrderDate,
              MAX(O.RequiredDate) AS RequiredDate,
              MAX(OD.DiscountPercentage) AS Discount
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
WHERE DATEDIFF(day, O.OrderDate, O.RequiredDate) < 20
GROUP BY O.OrderDate, O.RequiredDate
ORDER BY Discount DESC;
```



	NumberOfCustomers	TimeOrderTook	OrderDate	RequiredDate	Discount
1	7	14	2015-01-30	2015-02-13	0.250
2	2	14	2015-12-11	2015-12-25	0.250
3	3	14	2016-03-12	2016-03-26	0.250
4	2	14	2016-03-19	2016-04-02	0.250
5	3	14	2015-05-06	2015-05-20	0.200
6	4	14	2014-10-30	2014-11-13	0.200
7	1	14	2015-11-14	2015-11-28	0.200
8	1	14	2015-12-15	2015-12-29	0.200
9	6	14	2016-01-22	2016-02-05	0.200
10	1	14	2016-03-24	2016-04-07	0.200
11	5	14	2015-04-23	2015-05-07	0.150
12	2	14	2014-11-12	2014-11-26	0.150
13	3	14	2015-05-27	2015-06-10	0.150
14	4	14	2015-10-06	2015-10-20	0.150
15	3	14	2015-10-13	2015-10-27	0.150
16	3	14	2015-10-16	2015-10-30	0.150
17	4	14	2016-01-06	2016-01-20	0.150
18	2	14	2016-03-06	2016-03-20	0.150
19	2	14	2015-09-17	2015-10-01	0.100
20	4	14	2015-03-25	2015-04-08	0.100
21	4	14	2015-02-19	2015-03-05	0.100
22	3	14	2015-10-01	2015-10-15	0.100
23	5	14	2016-01-21	2016-02-04	0.100
24	2	14	2014-07-31	2014-08-14	0.050
25	1	14	2015-03-18	2015-04-01	0.050
26	3	14	2015-09-10	2015-09-24	0.050
27	2	14	2015-12-19	2016-01-02	0.050
28	2	14	2016-02-02	2016-02-16	0.050
29	2	14	2016-03-16	2016-03-30	0.050
30	1	14	2016-04-02	2016-04-16	0.050

JSON Results

```sql
--Query
SELECT TOP 30
              COUNT(C.CustomerID) NumberOfCustomers,
              ([Sales].[getTotalShipTime](O.OrderDate, O.RequiredDate)) AS TimeOrderTook,
              MAX(O.OrderDate) AS OrderDate,
              MAX(O.RequiredDate) AS RequiredDate,
              MAX(OD.DiscountPercentage) AS Discount
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
WHERE DATEDIFF(day, O.OrderDate, O.RequiredDate) < 20
GROUP BY O.OrderDate, O.RequiredDate
ORDER BY Discount DESC
FOR JSON PATH, ROOT('Top 30 Customers');
```



```
{
    "Total Ship Time":[{
                        "NumberOfCustomers":7,
                        "TimeOrderTook":14,
                        "OrderDate":"2015-01-30",
                        "RequiredDate":"2015-02-13",
                        "Discount":0.250
                        },{
                        "NumberOfCustomers":2,
                        "TimeOrderTook":14,
                        "OrderDate":"2015-12-11",
                        "RequiredDate":"2015-12-25",
                        "Discount":0.250
                        },{
                        "NumberOfCustomers":3,
                        "TimeOrderTook":14,
                        "OrderDate":"2016-03-12",
                        "RequiredDate":"2016-03-26",
                        "Discount":0.250
                        },{
                        "NumberOfCustomers":2,
                        "TimeOrderTook":14,
                        "OrderDate":"2016-03-19",
                        "RequiredDate":"2016-04-02",
                        "Discount":0.250
                        },{
                        "NumberOfCustomers":3,
                        "TimeOrderTook":14,
                        "OrderDate":"2015-05-06",
                        "RequiredDate":"2015-05-20"
                        "Discount":0.200
                        },{
                        "NumberOfCustomers":4,
                        "TimeOrderTook":14,
                        "OrderDate":"2014-10-30",
                        "RequiredDate":"2014-11-13",
                        "Discount":0.200
                        },{
                        "NumberOfCustomers":1,
                        "TimeOrderTook":14,
                        "OrderDate":"2015-11-14",
                        "RequiredDate":"2015-11-28",
                        "Discount":0.200
```

Proposition 17: Return the number of customers that either placed orders with high amounts of product, or the ones that placed orders with a low quantity. Make sure their order time was less than 2 months.

The Query above is asking for you to return the number of customers that either placed high order amounts or low order amounts. In addition to that, make sure the order time for the given order was under 2 months. To start, we know we need to use the Sales.Customer table, Sales.Order table and finally then Sales.OrderDetails table. First, you must use the SQL COUNT function in order to receive the number of customers under each category. Then, we must use the function created for the specific problem whether to determine if the order quantity was large or small best on the quantity number. Lastly, once all the tables have been joined, you must use the function to check whether or not the two dates between ordered and required is at most 2 months.

Projected Columns

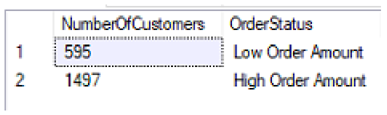
Table Name	Column Name
Sales.OrderDetail	Quantity
Sales.Customer	CustomerID

Order By

Table Name	Column Name
Sales.Customer	CustomerID

Table Results

```
--Function
CREATE FUNCTION [Sales].[isOrderAmountHigh](@Quantity INT)
RETURNS nvarchar(10)
AS
BEGIN

    DECLARE @answer nvarchar(10)
    SET @answer= CASE
                    WHEN @Quantity > 10 THEN 'High Order Amount'
                    ELSE 'Low Order Amount'
    END
    RETURN @answer
END;
GO

--Query
SELECT
              COUNT(C.CustomerID) AS NumberOfCustomers,
              OrderStatus = ([Sales].[isOrderAmountHigh](OD.Quantity))
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
WHERE DATEDIFF(month, O.OrderDate, O.RequiredDate) < 2 -- Use function
GROUP BY ([Sales].[isOrderAmountHigh](OD.Quantity))
ORDER BY NumberOfCustomers;
```

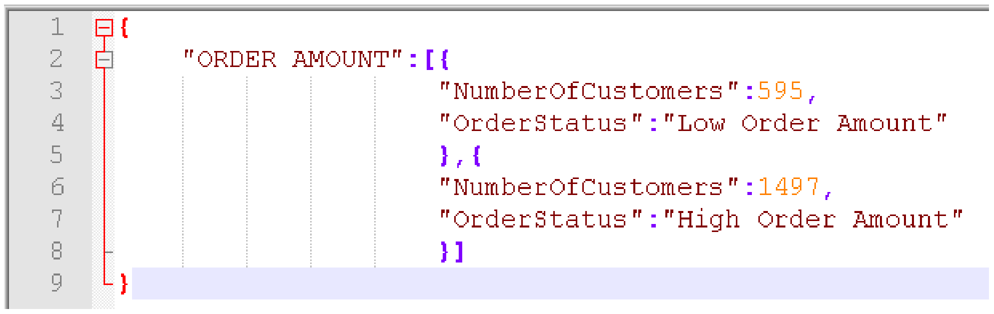


	NumberOfCustomers	OrderStatus
1	595	Low Order Amount
2	1497	High Order Amount

JSON Results

```
--Query with JSON
SELECT
              COUNT(C.CustomerID) AS NumberOfCustomers,
              OrderStatus = ([Sales].[isOrderAmountHigh](OD.Quantity)) -- Use function
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
```

```
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
WHERE DATEDIFF(month, O.OrderDate, O.RequiredDate) < 2 -- Use function
GROUP BY ([Sales].[isOrderAmountHigh](OD.Quantity))
ORDER BY NumberOfCustomers
FOR JSON PATH, ROOT('ORDER AMOUNT');
```



```
{
    "ORDER AMOUNT":[{
                "NumberOfCustomers":595,
                "OrderStatus":"Low Order Amount"
                },{
                "NumberOfCustomers":1497,
                "OrderStatus":"High Order Amount"
                }]
}
```

Proposition 18: Return all customers that have placed orders on now discontinued items and non- discontinued items, show the average discount for the products and make sure they were shipped to the USA.

The Query above is asking you to return all customers that have placed orders on now discontinued items and non-discontinued items, show the average discount for the products and make sure they were shipped to the USA. First step in doing this is deciding what tables we need to use in order to solve the queries. For this problem we need to use 4 tables: Sales.Order, Sales.OrderDetail, Sales.Customer and the Production.Product table. Once you have linked them upon their according keys, you will then need to use the function created in order to tell if the country that the user has decided to look for (in this case the USA) matches the data. Once you have used the function, group the data in respect to what items have been discounted and which haven't and use the AVG SQL function in order to provide the average discount for all products within the 2 categories.

Projected Columns

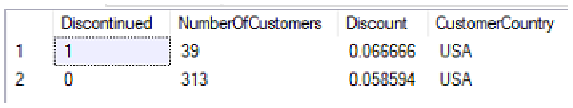
Table Name	Column Name
Sales.OrderDetail	DiscountPercentage
Sales.Customer	CustomerID, CustomerCountry
Production.Product	Discontinued

Order By

Table Name	Column Name
Sales.OrderDetail	DiscountPercentage

Table Results

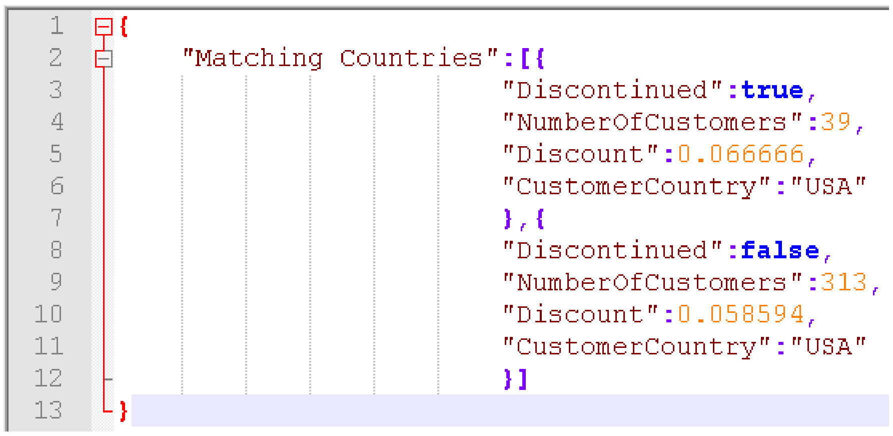
```sql
--Function
CREATE FUNCTION [Sales].[livesInMatchingCountry](@CustomerCountry NVARCHAR(20),
@RequestedCountry NVARCHAR(20))
RETURNS INT
       AS
       BEGIN
              DECLARE @livesInTheCountry INT=
              CASE
                            WHEN @CustomerCountry = @RequestedCountry THEN 1 ELSE 0
              END
              RETURN @livesInTheCountry
END
GO

--Query
SELECT
       P.Discontinued,
       COUNT(C.CustomerID) AS NumberOfCustomers,
       AVG(OD.DiscountPercentage) AS Discount,
       MAX(C.CustomerCountry) AS CustomerCountry
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
INNER JOIN Production.Product AS P
       ON OD.ProductId = P.ProductId
WHERE [Sales].[livesInMatchingCountry](C.CustomerCountry,'USA') = 1
GROUP BY P.Discontinued
ORDER BY Discount DESC;
```



```
   Discontinued  NumberOfCustomers  Discount  CustomerCountry
1  1             39                 0.066666  USA
2  0             313                0.058594  USA
```

JSON Results

```sql
--Query with JSON
SELECT
       P.Discontinued,
       COUNT(C.CustomerID) AS NumberOfCustomers,
       AVG(OD.DiscountPercentage) AS Discount,
       MAX(C.CustomerCountry) AS CustomerCountry
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
INNER JOIN Production.Product AS P
       ON OD.ProductId = P.ProductId
WHERE [Sales].[livesInMatchingCountry](C.CustomerCountry,'USA') = 1
GROUP BY P.Discontinued
ORDER BY Discount DESC
FOR JSON PATH, ROOT('Matching Countries');
```



```
{
    "Matching Countries":[{
                    "Discontinued":true,
                    "NumberOfCustomers":39,
                    "Discount":0.066666,
                    "CustomerCountry":"USA"
                    },{
                    "Discontinued":false,
                    "NumberOfCustomers":313,
                    "Discount":0.058594,
                    "CustomerCountry":"USA"
                    }]
}
```

Proposition 19: Return all customers placed orders on products that had prices greater than the average price of a product. On top of that, make sure the product was not yet discontinued and return the average price and order date is greater than 2015.

The Query above is asking you to return all customers that placed orders on products that had prices greater than the average price of a product. On top of that, make sure the product was not yet discontinued and return the average price and order date is greater than 2015. To start, we first have to join 4 tables: Sales.CustomerID, Sales.Order, Sales.OrderDetail and Production.Product. Then, you must create a variable that will keep track of what the average price for all products is. Once you have linked all tables according to their keys, you then check to see if the average price is less than the current price and after that you check to see if it has not been discontinued (value is 0). Lastly, check to see that the date is any time after the start of 2015.

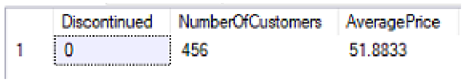
Projected Columns

Table Name	Column Name
Sales.OrderDetail	N/A
Sales.Customer	CustomerID
Production.Product	Discontinued, UnitPrice

Table Results

```
--Function
CREATE FUNCTION [Sales].[productAboveAverage](@UnitPrice FLOAT, @AvgPrice FLOAT)
RETURNS FLOAT
       AS
       BEGIN
              DECLARE @AboveAverage INT
              SET @AboveAverage =
                     CASE
```

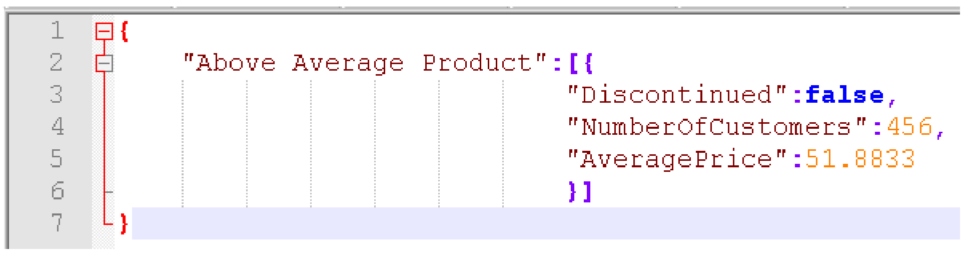
```
                                    WHEN @UnitPrice > @AvgPrice THEN 1 ELSE 0
       END
RETURN @AboveAverage
END
GO

--Query
DECLARE @AvgPrice FLOAT;
SET @AvgPrice = (SELECT AVG(UnitPrice) FROM Production.Product);

SELECT
       P.Discontinued,
       COUNT(C.CustomerID) AS NumberOfCustomers,
       AVG(P.UnitPrice) AS AveragePrice
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
INNER JOIN Production.Product AS P
       ON OD.ProductId = P.ProductId
WHERE [Sales].[productAboveAverage](OD.UnitPrice, @AvgPrice) = 1 -- Use function
AND P.Discontinued = 0
AND O.OrderDate > '20150101'
GROUP BY P.Discontinued;
```



	Discontinued	NumberOfCustomers	AveragePrice
1	0	456	51.8833

JSON Results

```
--Query with JSON
DECLARE @AvgPrice FLOAT;
SET @AvgPrice = (SELECT AVG(UnitPrice) FROM Production.Product);

SELECT
       P.Discontinued,
       COUNT(C.CustomerID) AS NumberOfCustomers,
       AVG(P.UnitPrice) AS AveragePrice
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
       ON O.CustomerId = C.CustomerId
INNER JOIN Production.Product AS P
       ON OD.ProductId = P.ProductId
WHERE [Sales].[productAboveAverage](OD.UnitPrice, @AvgPrice) = 1 -- Use function
AND P.Discontinued = 0
AND O.OrderDate > '20150101'
GROUP BY P.Discontinued
FOR JSON PATH, ROOT('Above Average Product');
```



```
{
    "Above Average Product":[{
                        "Discontinued":false,
                        "NumberOfCustomers":456,
                        "AveragePrice":51.8833
                        }]
}
```

Proposition 20: Return all customers that ordered products that where under the average price of all products that live in Mexico and ordered their product any time after the year 2014 that came within 2 months.

The Query above is asking you to return all customers that ordered products that where under the average price of all products that live in Mexico and ordered their product any time after the year 2014 that came within 2 months. Like some of the previous, this query will require 4 tables: Sales.Customer, Sales.Order, Sales.OrderDetail and Production.Product. With this query, we will need to use multiple functions I have created in order to solve it. The functions we need to use are listed: [Sales].[productAboveAverage], [Sales].[livesInMatchingCountry], [Sales].[getTotalShipTime]. From the conditions provided, using all of these functions will give you the answer.

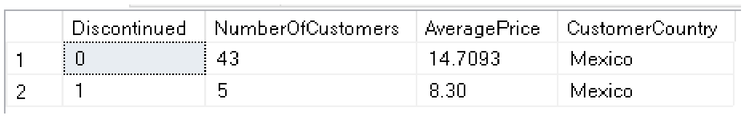
Projected Columns

Table Name	Column Name
Sales.OrderDetail	N/A
Sales.Customer	CustomerID, CustomerCountry
Production.Product	Discontinued, UnitPrice

Table Results

```
--Functions:
[Sales].[productAboveAverage], [Sales].[livesInMatchingCountry],

--Query
DECLARE @AvgPrice FLOAT;
SET @AvgPrice = (SELECT AVG(UnitPrice) FROM Production.Product);

SELECT
       P.Discontinued,
       COUNT(C.CustomerID) AS NumberOfCustomers,
       AVG(P.UnitPrice) AS AveragePrice,
       MAX(C.CustomerCountry) AS CustomerCountry
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
       ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
```

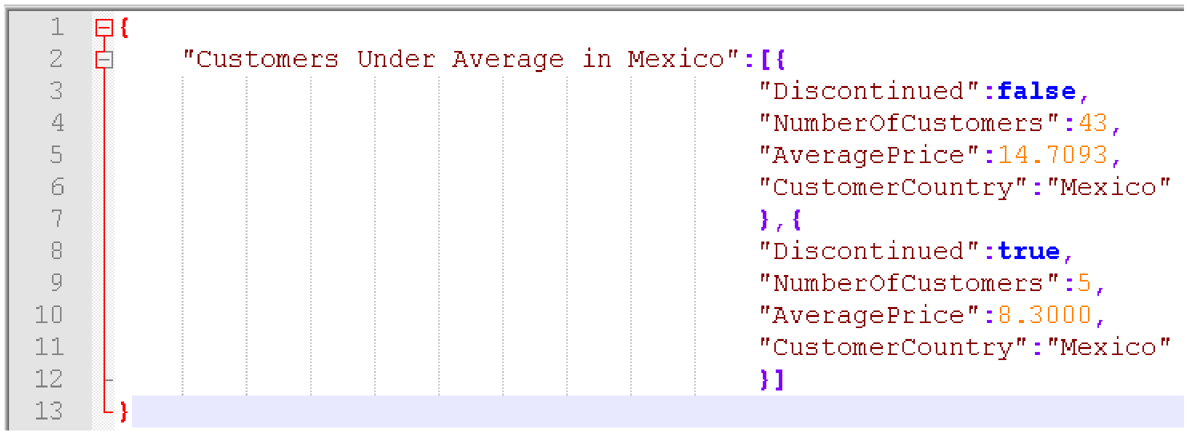
```sql
	ON O.CustomerId = C.CustomerId
INNER JOIN Production.Product AS P
	ON OD.ProductId = P.ProductId
WHERE [Sales].[productAboveAverage](OD.UnitPrice, @AvgPrice) = 0
AND [Sales].[livesInMatchingCountry](C.CustomerCountry,'Mexico') = 1
AND O.OrderDate > '20140101'
AND DATEDIFF(month, O.OrderDate, O.RequiredDate) < 2
GROUP BY P.Discontinued;
```



	Discontinued	NumberOfCustomers	AveragePrice	CustomerCountry
1	0	43	14.7093	Mexico
2	1	5	8.30	Mexico

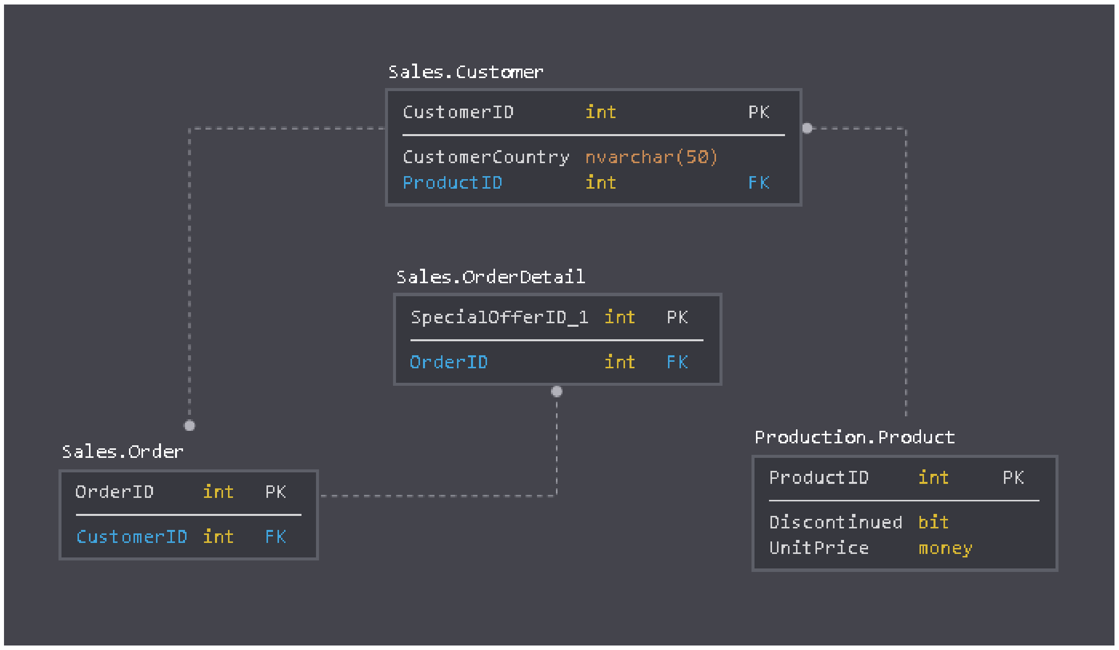
JSON Results

```sql
--Query
DECLARE @AvgPrice FLOAT;
SET @AvgPrice = (SELECT AVG(UnitPrice) FROM Production.Product);

SELECT
	P.Discontinued,
	COUNT(C.CustomerID) AS NumberOfCustomers,
	AVG(P.UnitPrice) AS AveragePrice,
	MAX(C.CustomerCountry) AS CustomerCountry
FROM Sales.OrderDetail AS OD
INNER JOIN Sales.[Order] AS O
	ON OD.OrderID = O.OrderID
INNER JOIN Sales.Customer AS C
	ON O.CustomerId = C.CustomerId
INNER JOIN Production.Product AS P
	ON OD.ProductId = P.ProductId
WHERE P.UnitPrice < @AvgPrice -- Use function
AND C.CustomerCountry = 'Mexico' -- Use function
AND O.OrderDate > '20140101'
AND DATEDIFF(month, O.OrderDate, O.RequiredDate) < 2
GROUP BY P.Discontinued
FOR JSON PATH, ROOT('Customers Under Average in Mexico');
```



```json
{
    "Customers Under Average in Mexico":[{
                                          "Discontinued":false,
                                          "NumberOfCustomers":43,
                                          "AveragePrice":14.7093,
                                          "CustomerCountry":"Mexico"
                                          },{
                                          "Discontinued":true,
                                          "NumberOfCustomers":5,
                                          "AveragePrice":8.3000,
                                          "CustomerCountry":"Mexico"
                                          }]
}
```

Tristen Aguilar Project 1



Prepared by: Tristen Aguilar

Date Prepared: 3/19/2020