2022

GROUP PROJECT 1

JAY MISTRY: INDIVIDUAL 9 QUERIES

DATE PREPARED: 03/10/2022

Problem 01 (Simple Worst): Using

Proposition:

Calculate the danger zones for orders that arrived later than the required date

Information:

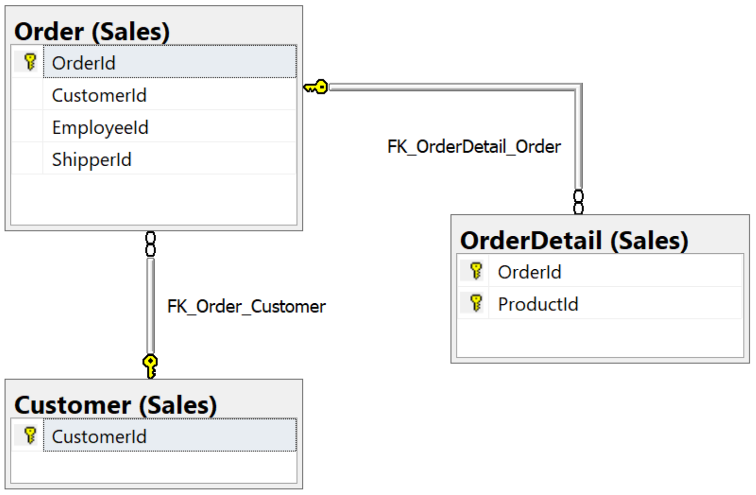
Display the OrderId, CustomerId, CustomerContactName, TotalPaid, RequiredDate, ShippedDate, and DangerZone

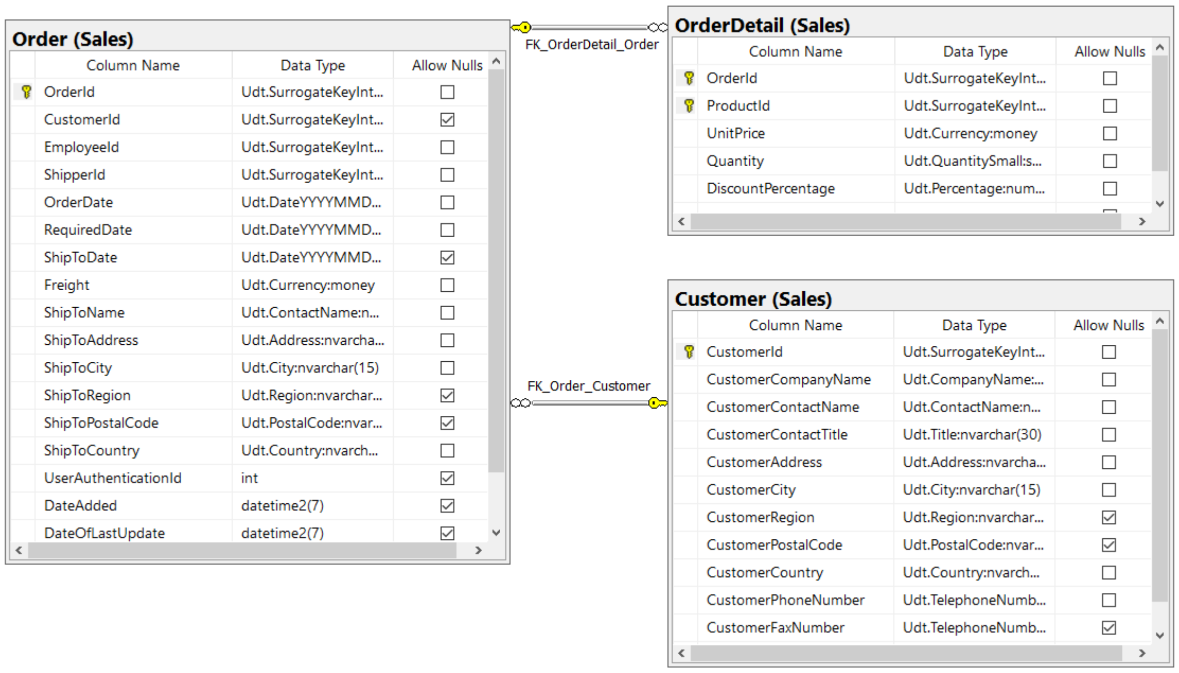
Tables Involved:

Order, Customer, OrderDetail

Diagrams:

Key View & Standard View





Order (Sales)

	Column Name	Data Type	Allow Nulls
🔑	OrderId	Udt.SurrogateKeyInt...	☐
	CustomerId	Udt.SurrogateKeyInt...	☑
	EmployeeId	Udt.SurrogateKeyInt...	☐
	ShipperId	Udt.SurrogateKeyInt...	☐
	OrderDate	Udt.DateYYYYMMD...	☐
	RequiredDate	Udt.DateYYYYMMD...	☐
	ShipToDate	Udt.DateYYYYMMD...	☑
	Freight	Udt.Currency:money	☐
	ShipToName	Udt.ContactName:n...	☐
	ShipToAddress	Udt.Address:nvarcha...	☐
	ShipToCity	Udt.City:nvarchar(15)	☐
	ShipToRegion	Udt.Region:nvarchar...	☑
	ShipToPostalCode	Udt.PostalCode:nvar...	☑
	ShipToCountry	Udt.Country:nvarch...	☐
	UserAuthenticationId	int	☑
	DateAdded	datetime2(7)	☑
	DateOfLastUpdate	datetime2(7)	☑

OrderDetail (Sales)

	Column Name	Data Type	Allow Nulls
🔑	OrderId	Udt.SurrogateKeyInt...	☐
🔑	ProductId	Udt.SurrogateKeyInt...	☐
	UnitPrice	Udt.Currency:money	☐
	Quantity	Udt.QuantitySmall:s...	☐
	DiscountPercentage	Udt.Percentage:num...	☐

Customer (Sales)

	Column Name	Data Type	Allow Nulls
🔑	CustomerId	Udt.SurrogateKeyInt...	☐
	CustomerCompanyName	Udt.CompanyName:...	☐
	CustomerContactName	Udt.ContactName:n...	☐
	CustomerContactTitle	Udt.Title:nvarchar(30)	☐
	CustomerAddress	Udt.Address:nvarcha...	☐
	CustomerCity	Udt.City:nvarchar(15)	☐
	CustomerRegion	Udt.Region:nvarchar...	☑
	CustomerPostalCode	Udt.PostalCode:nvar...	☑
	CustomerCountry	Udt.Country:nvarch...	☐
	CustomerPhoneNumber	Udt.TelephoneNumb...	☐
	CustomerFaxNumber	Udt.TelephoneNumb...	☑

SELECT CLASUE Chart:

Table Name	Column Name
Orders	OrderID CustomerID RequiredDate ShipToDate
Customer	CustomerContactName
OrderDetail	UnitPrice Quantity DiscountPercentage

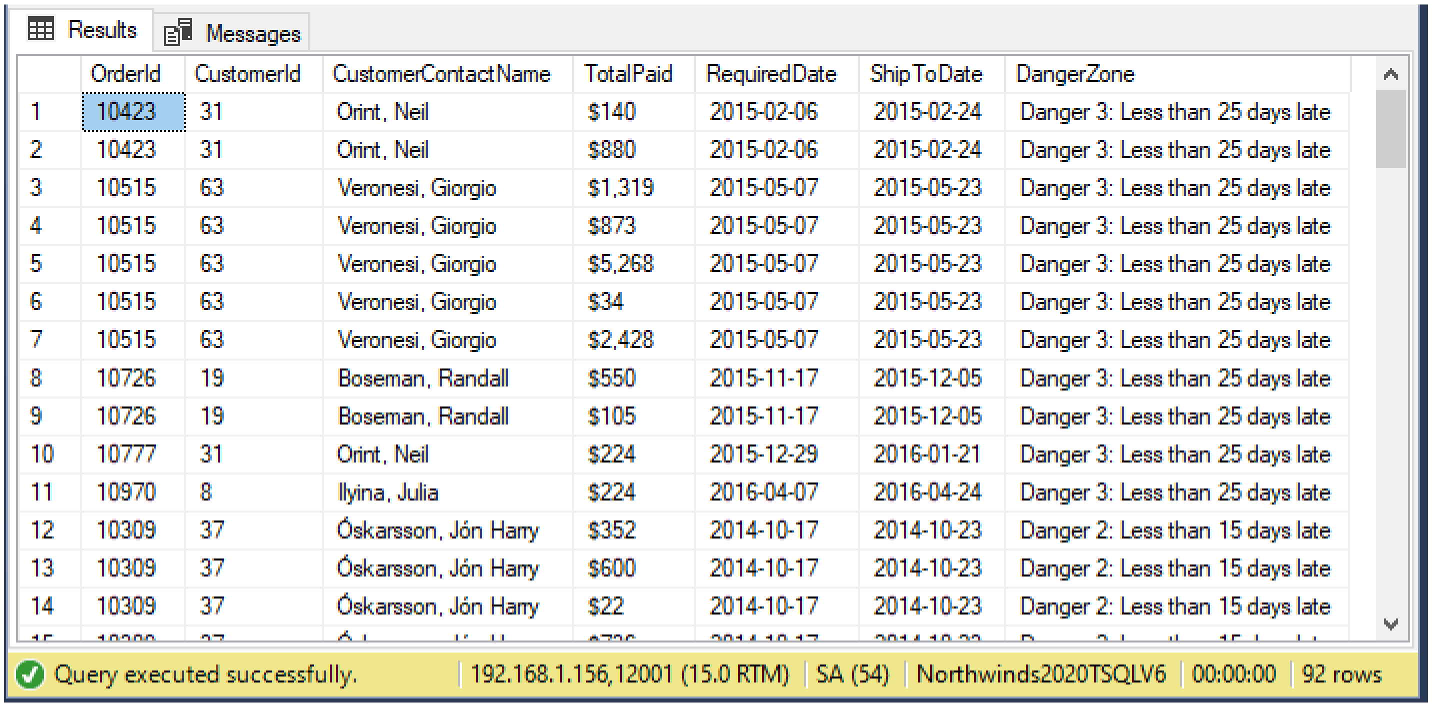
ORDER BY Chart:

Table Name	Column Name	Sort Order
Order	DangerZone	DESC
Order	OrderId	ASC

Problem solving using query:

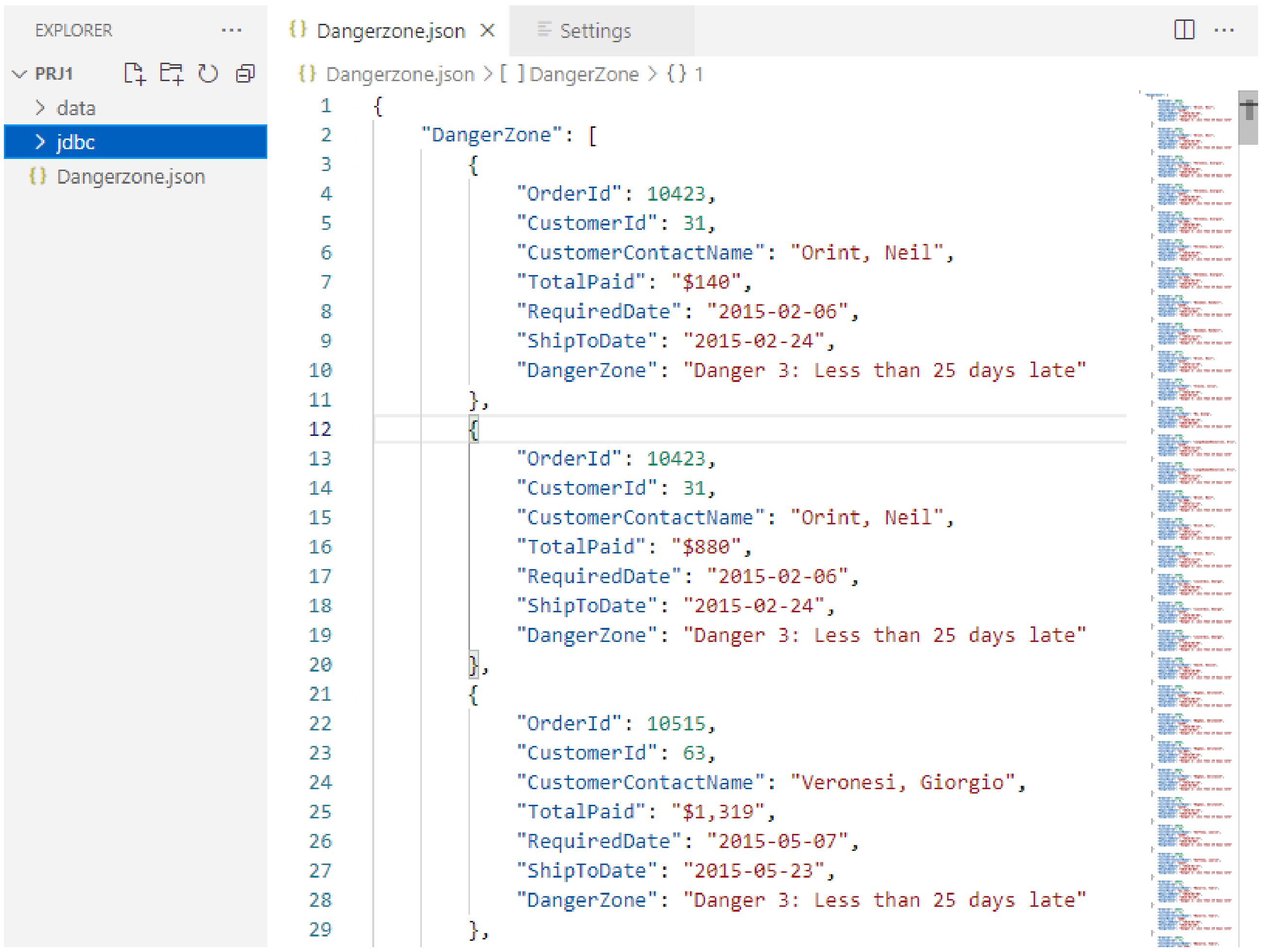
```
USE Northwinds2022TSQLV7; GO

DROP FUNCTION IF EXISTS dbo.dangerZone; GO

CREATE FUNCTION dbo.dangerZone
(
    @RequiredDate DATE,
    @ShippedDate DATE
)
RETURNS NVARCHAR(35)
AS
BEGIN
    DECLARE @DaysLate INT = DATEDIFF(DAY, @RequiredDate, @ShippedDate);
    IF (@DaysLate < 5)
        RETURN 'Danger 1: Less than 5 days late';
    IF (@DaysLate < 15)
        RETURN 'Danger 2: Less than 15 days late';
    IF (@DaysLate < 25)
        RETURN 'Danger 3: Less than 25 days late';
    RETURN 'Danger 4: More than 25 days late';
END; GO

SELECT O.OrderId,
       O.CustomerId,
       C.CustomerContactName,
       FORMAT(D.UnitPrice * D.Quantity * (1 - D.DiscountPercentage), 'c0') AS TotalPaid,
       RequiredDate,
       ShipToDate,
       dbo.dangerZone(RequiredDate, ShipToDate) AS DangerZone
FROM Sales.[Order] AS O
    INNER JOIN Sales.Customer AS C
        ON O.CustomerId = C.CustomerId
    INNER JOIN Sales.OrderDetail AS D
        ON O.OrderId = D.OrderId
WHERE requireddate < shiptodate
ORDER BY DangerZone DESC;
FOR JSON PATH, ROOT('DangerZone'), INCLUDE_NULL_VALUES;
```

```
(92 rows affected) -Completion time: 2021-10-10T20:54:23.3722301-04:00
```



	OrderId	CustomerId	CustomerContactName	TotalPaid	RequiredDate	ShipToDate	DangerZone
1	10423	31	Orint, Neil	$140	2015-02-06	2015-02-24	Danger 3: Less than 25 days late
2	10423	31	Orint, Neil	$880	2015-02-06	2015-02-24	Danger 3: Less than 25 days late
3	10515	63	Veronesi, Giorgio	$1,319	2015-05-07	2015-05-23	Danger 3: Less than 25 days late
4	10515	63	Veronesi, Giorgio	$873	2015-05-07	2015-05-23	Danger 3: Less than 25 days late
5	10515	63	Veronesi, Giorgio	$5,268	2015-05-07	2015-05-23	Danger 3: Less than 25 days late
6	10515	63	Veronesi, Giorgio	$34	2015-05-07	2015-05-23	Danger 3: Less than 25 days late
7	10515	63	Veronesi, Giorgio	$2,428	2015-05-07	2015-05-23	Danger 3: Less than 25 days late
8	10726	19	Boseman, Randall	$550	2015-11-17	2015-12-05	Danger 3: Less than 25 days late
9	10726	19	Boseman, Randall	$105	2015-11-17	2015-12-05	Danger 3: Less than 25 days late
10	10777	31	Orint, Neil	$224	2015-12-29	2016-01-21	Danger 3: Less than 25 days late
11	10970	8	Ilyina, Julia	$224	2016-04-07	2016-04-24	Danger 3: Less than 25 days late
12	10309	37	Óskarsson, Jón Harry	$352	2014-10-17	2014-10-23	Danger 2: Less than 15 days late
13	10309	37	Óskarsson, Jón Harry	$600	2014-10-17	2014-10-23	Danger 2: Less than 15 days late
14	10309	37	Óskarsson, Jón Harry	$22	2014-10-17	2014-10-23	Danger 2: Less than 15 days late

```
Query executed successfully. | 192.168.1.156,12001 (15.0 RTM) | SA (54) | Northwinds2020TSQLV6 | 00:00:00 | 92 rows
```



```
{
    "DangerZone": [
        {
            "OrderId": 10423,
            "CustomerId": 31,
            "CustomerContactName": "Orint, Neil",
            "TotalPaid": "$140",
            "RequiredDate": "2015-02-06",
            "ShipToDate": "2015-02-24",
            "DangerZone": "Danger 3: Less than 25 days late"
        },
        {
            "OrderId": 10423,
            "CustomerId": 31,
            "CustomerContactName": "Orint, Neil",
            "TotalPaid": "$880",
            "RequiredDate": "2015-02-06",
            "ShipToDate": "2015-02-24",
            "DangerZone": "Danger 3: Less than 25 days late"
        },
        {
            "OrderId": 10515,
            "CustomerId": 63,
            "CustomerContactName": "Veronesi, Giorgio",
            "TotalPaid": "$1,319",
            "RequiredDate": "2015-05-07",
            "ShipToDate": "2015-05-23",
            "DangerZone": "Danger 3: Less than 25 days late"
        },
```

Problem 01 (Simple Best): Using WideWorldImportersDW

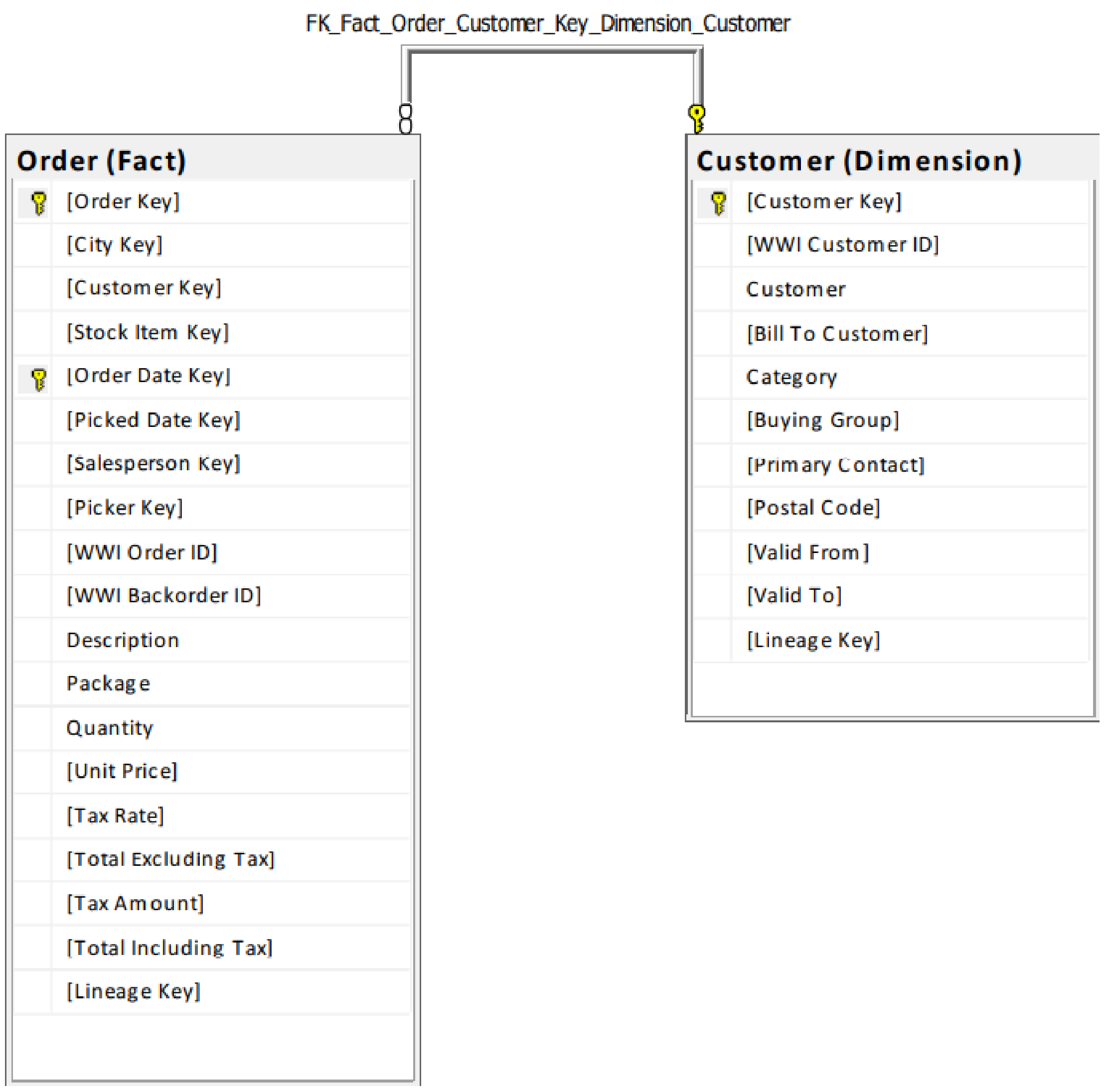
Proposition:

```
List the name of all the customers who have brought a 'DBA joke mug - mind if I join you? (Black)' mug
and the quantity they brought in descending order.
```

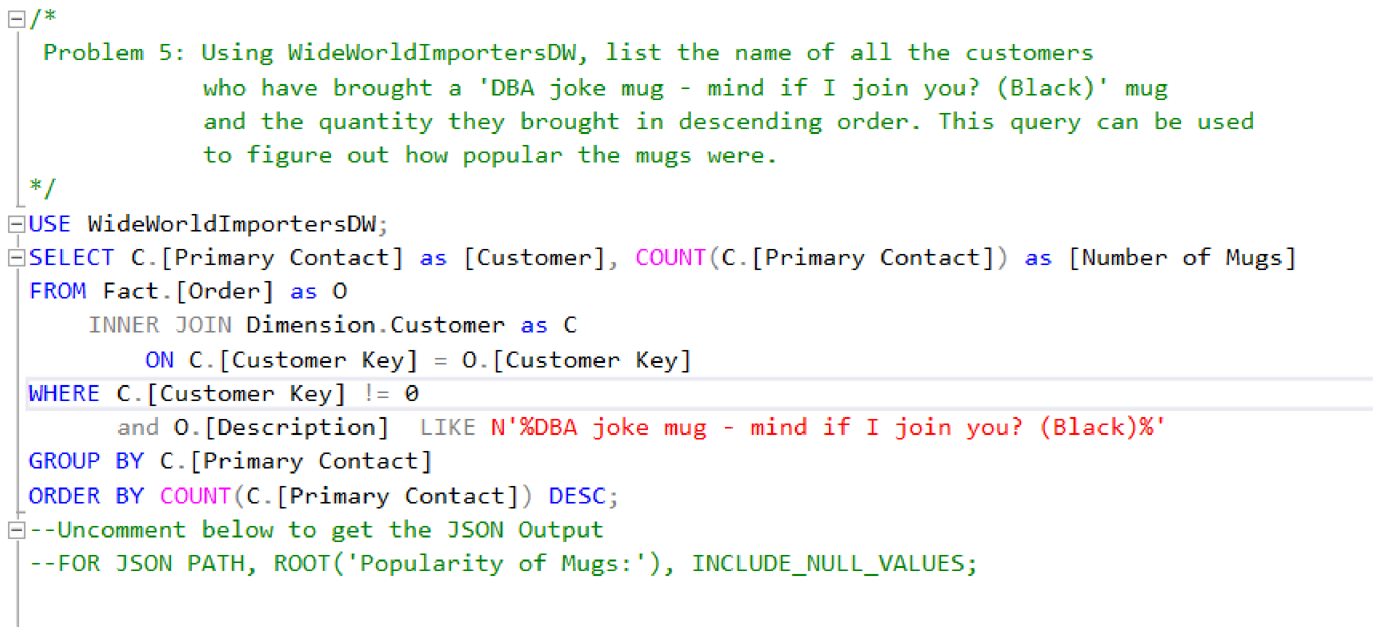
Diagrams:

Standard View

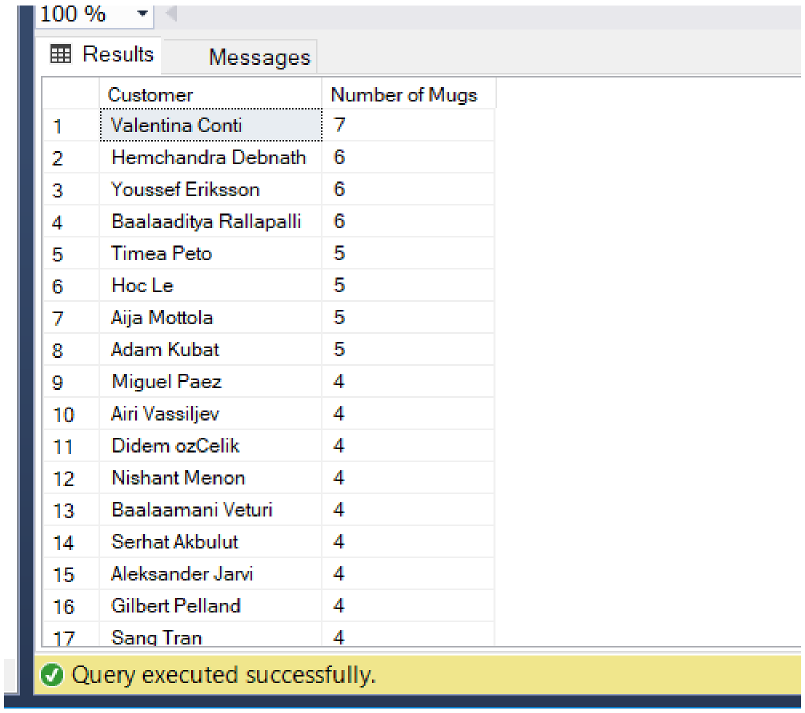
Database Diagram: The Order column and Customer column are joined the Customer key.



Problem solving using query:



```sql
/*
 Problem 5: Using WideWorldImportersDW, list the name of all the customers
            who have brought a 'DBA joke mug - mind if I join you? (Black)' mug
            and the quantity they brought in descending order. This query can be used
            to figure out how popular the mugs were.
*/
USE WideWorldImportersDW;
SELECT C.[Primary Contact] as [Customer], COUNT(C.[Primary Contact]) as [Number of Mugs]
FROM Fact.[Order] as O
    INNER JOIN Dimension.Customer as C
        ON C.[Customer Key] = O.[Customer Key]
WHERE C.[Customer Key] != 0
      and O.[Description]  LIKE N'%DBA joke mug - mind if I join you? (Black)%'
GROUP BY C.[Primary Contact]
ORDER BY COUNT(C.[Primary Contact]) DESC;
--Uncomment below to get the JSON Output
--FOR JSON PATH, ROOT('Popularity of Mugs:'), INCLUDE_NULL_VALUES;
```



100 %

Results Messages

	Customer	Number of Mugs
1	Valentina Conti	7
2	Hemchandra Debnath	6
3	Youssef Eriksson	6
4	Baalaaditya Rallapalli	6
5	Timea Peto	5
6	Hoc Le	5
7	Aija Mottola	5
8	Adam Kubat	5
9	Miguel Paez	4
10	Airi Vassiljev	4
11	Didem ozCelik	4
12	Nishant Menon	4
13	Baalaamani Veturi	4
14	Serhat Akbulut	4
15	Aleksander Jarvi	4
16	Gilbert Pelland	4
17	Sang Tran	4

Query executed successfully.

Conclusion: There are a total of 315 customers who brought at least one of the 'DBA joke mug - mind if I join you? (Black)' mug. Customer Valentina Conti was the one who brought the greatest quantity, 7. This query could be improved by simply listing how many customers brought x numbers of 'DBA joke mug - mind if I join you? (Black)' mugs.

Problem 02 (Medium Best): Using AdventureWorks2017

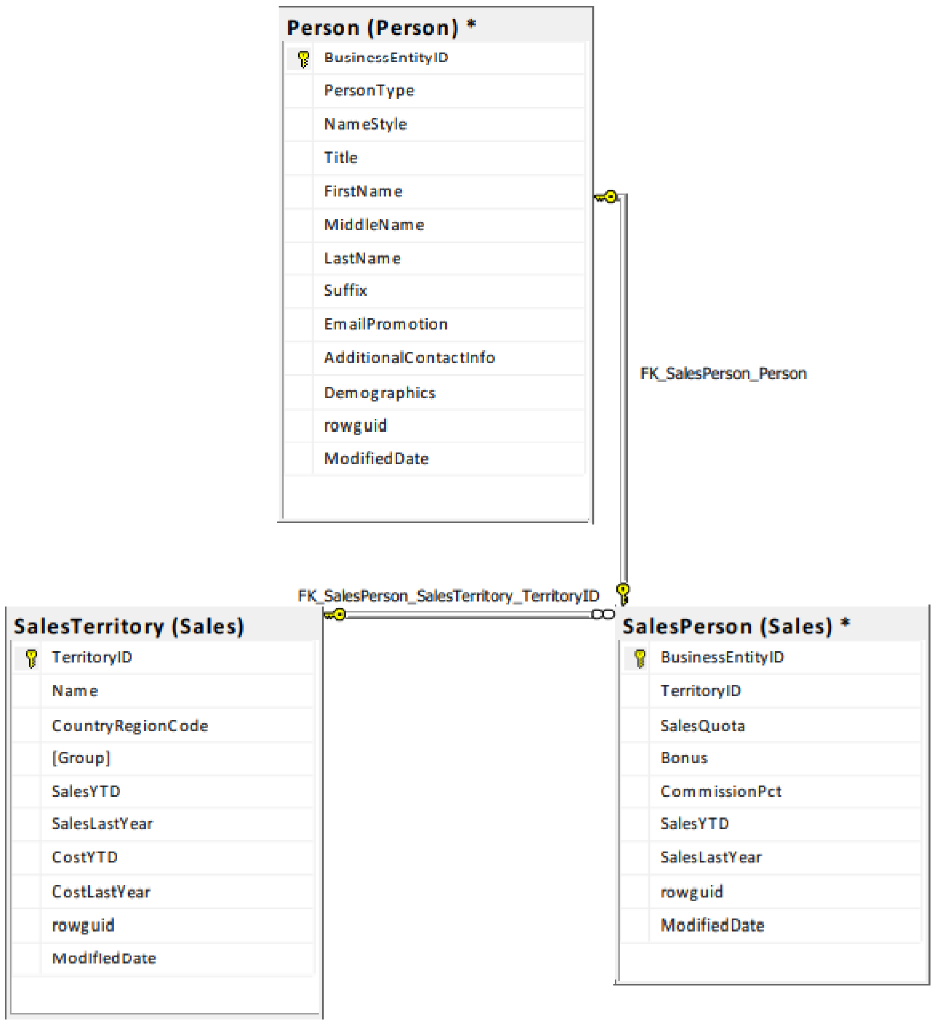
Proposition:

List the SalesYTD rounded to the thousandths for each of the salesperson, ordered by SalesYTD and last name. This query can be used to determine who is making the most sale.

Diagrams:

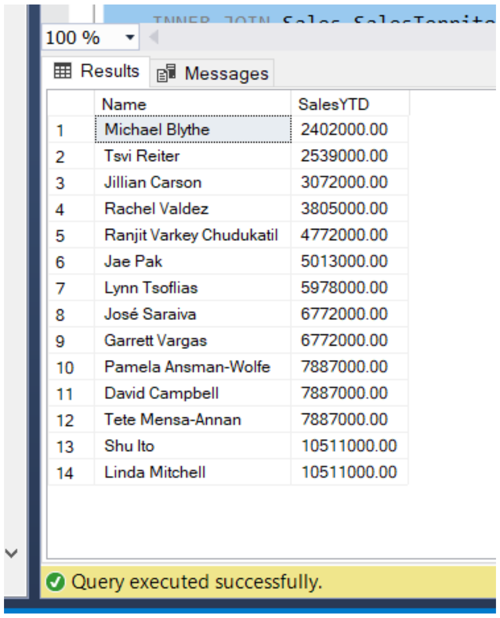
Standard View

Database Diagram: The SalesPerson column and Person column are joined by a BusinessEntityID key while the SalesPerson column and SalesTerritory column are joined by a TerritoryID key.



Problem solving using query:

```
/*
 Problem 11: Using AdventureWorks2017, list the SalesYTD rounded to the
             thousandths for each of the salesperson, ordered by SalesYTD
             and last name. This query can be used to determine who is making
             the most sale.
*/
USE AdventureWorks2017;
SELECT CONCAT(P.FirstName, ' ',P.LastName) as [Name], ROUND(T.SalesYTD,-3) as [SalesYTD]
FROM Sales.SalesPerson as S
    INNER JOIN Person.Person as P
        ON S.BusinessEntityID = P.BusinessEntityID
    INNER JOIN Sales.SalesTerritory as T
        ON S.TerritoryID = T.TerritoryID
GROUP BY P.LastName, P.FirstName, T.SalesYTD
ORDER BY T.SalesYTD, P.LastName;
--Uncomment below to get the JSON Output
--FOR JSON PATH, ROOT('SalesYTD:'), INCLUDE_NULL_VALUES;
```



	Name	SalesYTD
1	Michael Blythe	2402000.00
2	Tsvi Reiter	2539000.00
3	Jillian Carson	3072000.00
4	Rachel Valdez	3805000.00
5	Ranjit Varkey Chudukatil	4772000.00
6	Jae Pak	5013000.00
7	Lynn Tsoflias	5978000.00
8	José Saraiva	6772000.00
9	Garrett Vargas	6772000.00
10	Pamela Ansman-Wolfe	7887000.00
11	David Campbell	7887000.00
12	Tete Mensa-Annan	7887000.00
13	Shu Ito	10511000.00
14	Linda Mitchell	10511000.00

Query executed successfully.

Conclusion: There are a total of 14 salespersons who have made at least $2,402,000 in the current year up to the current date. If this was descending, we can determine who is making the most sale.

Problem 03 (Hard Best): Using AdventureWorks2017

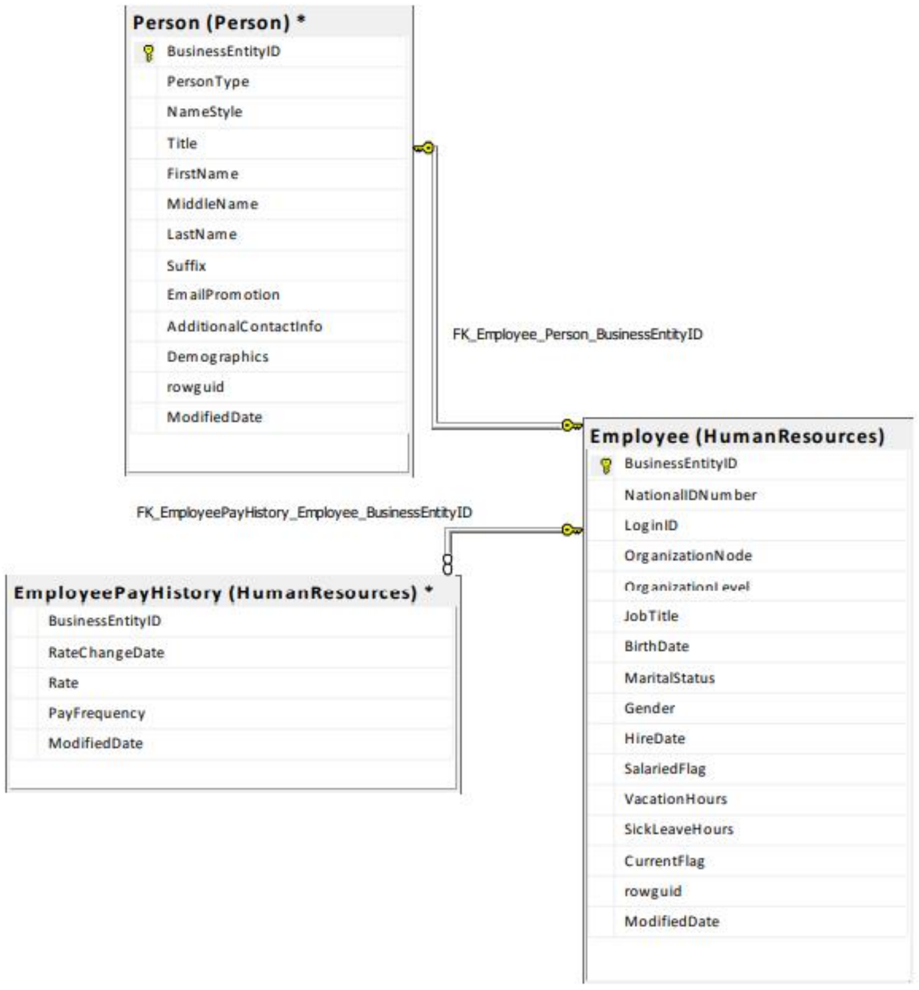
Proposition:

```
List the name, rate and number of years worked by each Human Resources employee up to the present day
in order of number of years.
```

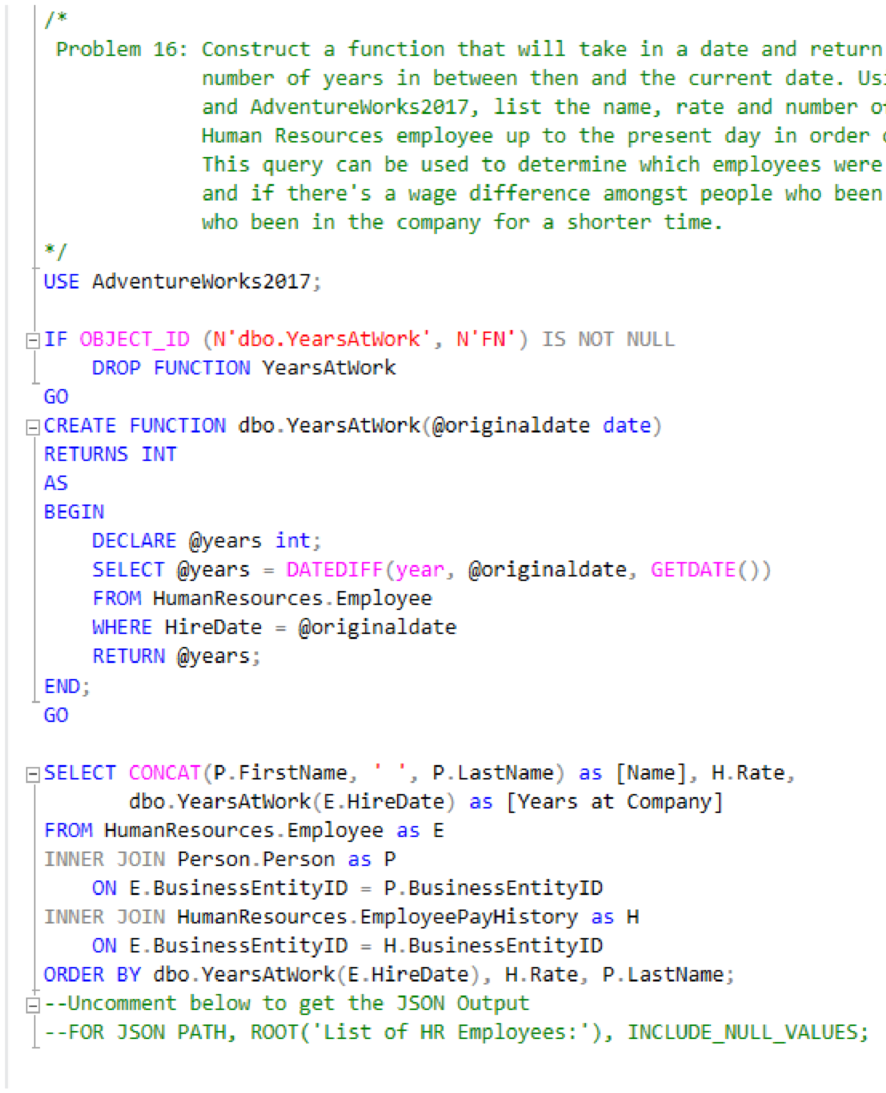
Diagrams:

Standard View

Database Diagram: The Employee column and Person column are joined by the BusinessEntityID key as well as the Employee column and EmployeePayHistory column.

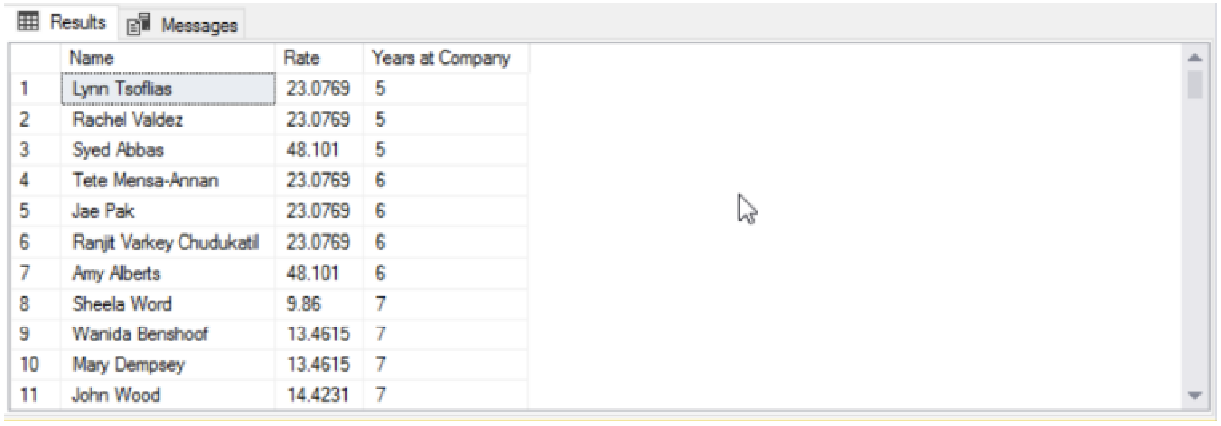


Problem solving using query:



```
/*
 Problem 16: Construct a function that will take in a date and return the
             number of years in between then and the current date. Using this function
             and AdventureWorks2017, list the name, rate and number of years worked by each
             Human Resources employee up to the present day in order of number of years.
             This query can be used to determine which employees were here for the longest time
             and if there's a wage difference amongst people who been in the company longer than those
             who been in the company for a shorter time.
*/
USE AdventureWorks2017;

IF OBJECT_ID (N'dbo.YearsAtWork', N'FN') IS NOT NULL
    DROP FUNCTION YearsAtWork
GO
CREATE FUNCTION dbo.YearsAtWork(@originaldate date)
RETURNS INT
AS
BEGIN
    DECLARE @years int;
    SELECT @years = DATEDIFF(year, @originaldate, GETDATE())
    FROM HumanResources.Employee
    WHERE HireDate = @originaldate
    RETURN @years;
END;
GO

SELECT CONCAT(P.FirstName, ' ', P.LastName) as [Name], H.Rate,
       dbo.YearsAtWork(E.HireDate) as [Years at Company]
FROM HumanResources.Employee as E
INNER JOIN Person.Person as P
    ON E.BusinessEntityID = P.BusinessEntityID
INNER JOIN HumanResources.EmployeePayHistory as H
    ON E.BusinessEntityID = H.BusinessEntityID
ORDER BY dbo.YearsAtWork(E.HireDate), H.Rate, P.LastName;
--Uncomment below to get the JSON Output
--FOR JSON PATH, ROOT('List of HR Employees:'), INCLUDE_NULL_VALUES;
```

Output Table:



Results | Messages

	Name	Rate	Years at Company
1	Lynn Tsoflias	23.0769	5
2	Rachel Valdez	23.0769	5
3	Syed Abbas	48.101	5
4	Tete Mensa-Annan	23.0769	6
5	Jae Pak	23.0769	6
6	Ranjit Varkey Chudukatil	23.0769	6
7	Amy Alberts	48.101	6
8	Sheela Word	9.86	7
9	Wanida Benshoof	13.4615	7
10	Mary Dempsey	13.4615	7
11	John Wood	14.4231	7

Conclusion: The minimum number of years worked by an HR employee is 5 years with a rate of $23.07. There is no discrepancy in years at the company and rate. Job title could be a better indicator of rate.

Problem 01 (Simple Worst): Using Northwinds2022TSQLV7

Proposition:

`List the names and order id of customers who ordered from the UK in 2016`

Information:

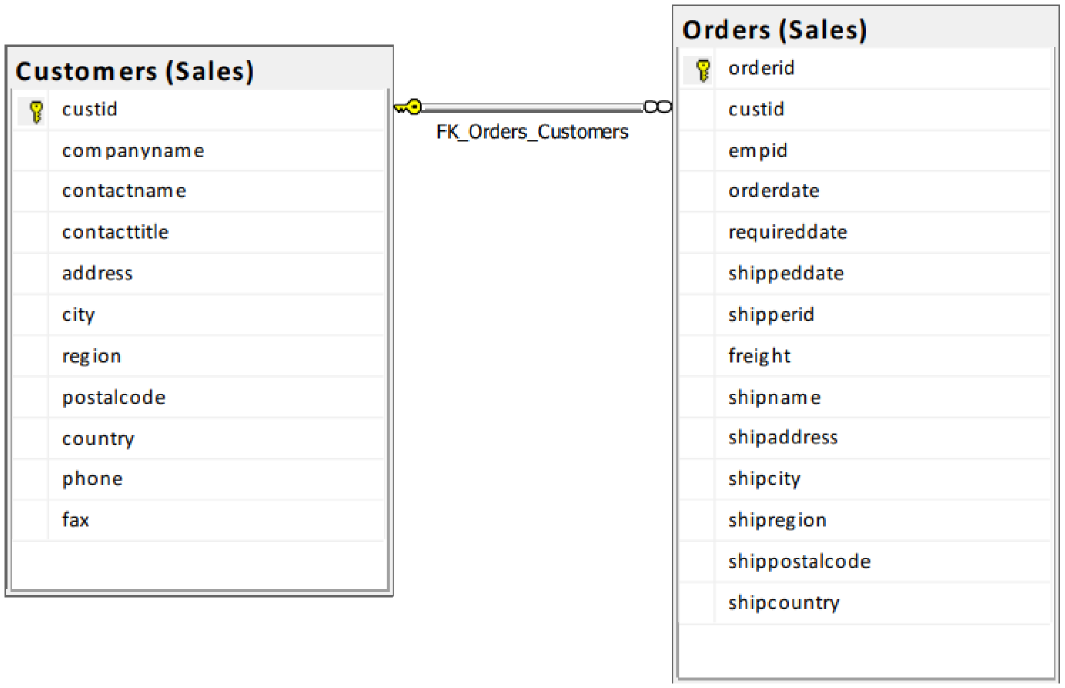
Display the OrderId, CustomerId, CustomerContactName, TotalPaid, RequiredDate, ShippedDate, and DangerZone

Tables Involved:

Order, Customer, OrderDetail

Diagrams:

Key View & Standard View



SELECT CLASUE Chart:

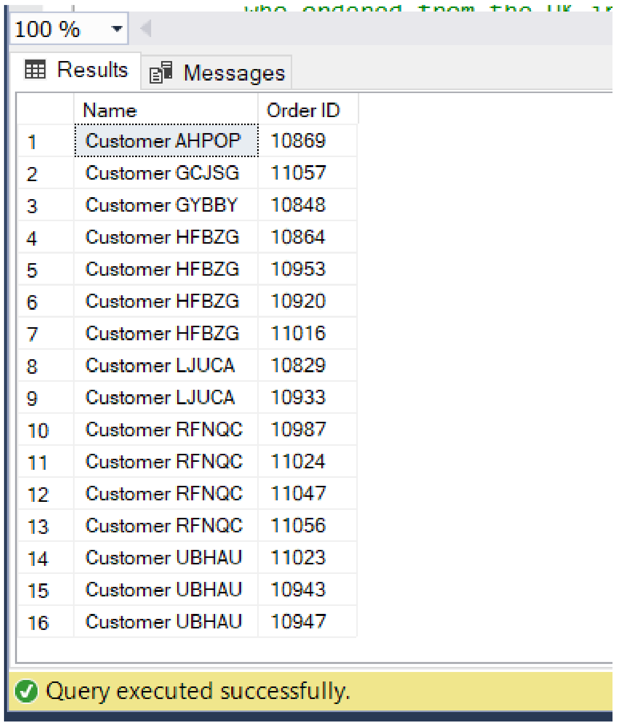
Table Name	Column Name
Orders	OrderID CustomerID RequiredDate ShipToDate
Customer	CustomerContactName
OrderDetail	UnitPrice Quantity DiscountPercentage

ORDER BY Chart:

Table Name	Column Name	Sort Order
Order	DangerZone	DESC
Order	OrderId	ASC

Problem solving using query:

```
/*
Problem 3: Using Northwinds2020TSQLV6, list the names and order id of customers
           who ordered from the UK in 2016. This query will be used to count how
           many orders were sent to the UK in 2016.
*/
USE Northwinds2020TSQLV6;
SELECT C.CustomerCompanyName as [Name], O.orderid as [Order ID]
FROM Sales.[Order] as O
    INNER JOIN Sales.[Customer] as C
        ON O.CustomerId = C.CustomerId
WHERE O.ShipToCountry = N'UK' AND O.orderdate >= '20160101' AND O.orderdate < '20170101'
ORDER BY C.CustomerCompanyName;
--Uncomment below to get the JSON Output
--FOR JSON PATH, ROOT('Orders sent to UK in 2016:'), INCLUDE_NULL_VALUES;
```



	Name	Order ID
1	Customer AHPOP	10869
2	Customer GCJSG	11057
3	Customer GYBBY	10848
4	Customer HFBZG	10864
5	Customer HFBZG	10953
6	Customer HFBZG	10920
7	Customer HFBZG	11016
8	Customer LJUCA	10829
9	Customer LJUCA	10933
10	Customer RFNQC	10987
11	Customer RFNQC	11024
12	Customer RFNQC	11047
13	Customer RFNQC	11056
14	Customer UBHAU	11023
15	Customer UBHAU	10943
16	Customer UBHAU	10947

Query executed successfully.

Problem 01 (Simple Worst Fixed): Using Northwinds2022TSQLV7

Proposition:

```
List the names and order id of customers who ordered from the UK in 2016
```

Information:

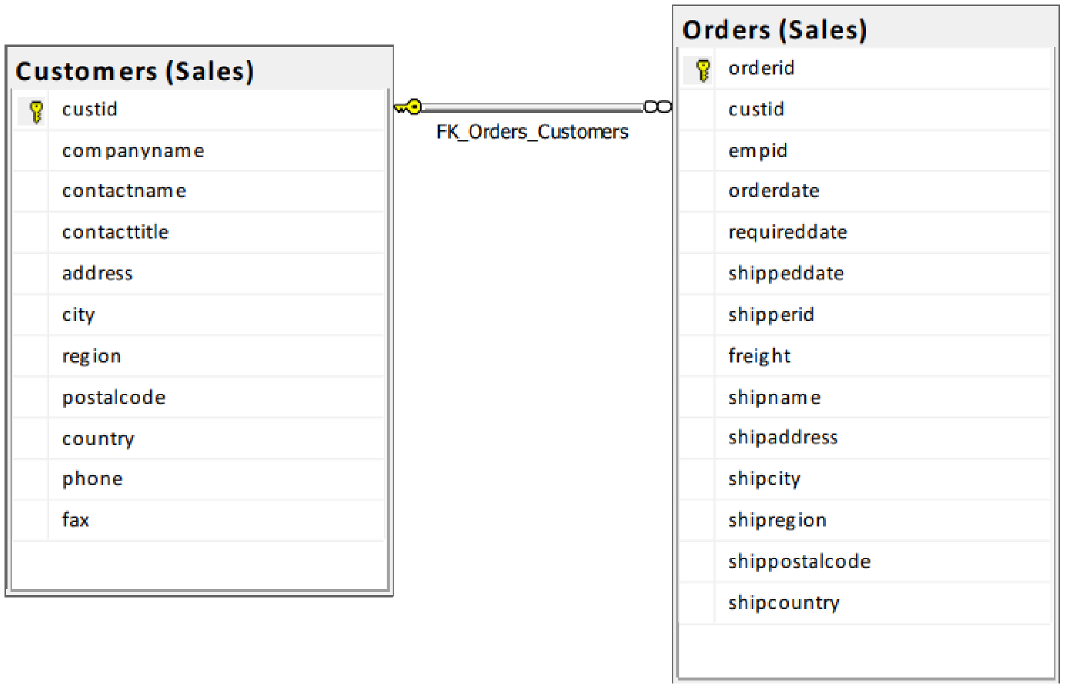
Display the OrderId, CustomerId, CustomerContactName, TotalPaid, RequiredDate, ShippedDate, and DangerZone

Tables Involved:

Order, Customer, OrderDetail

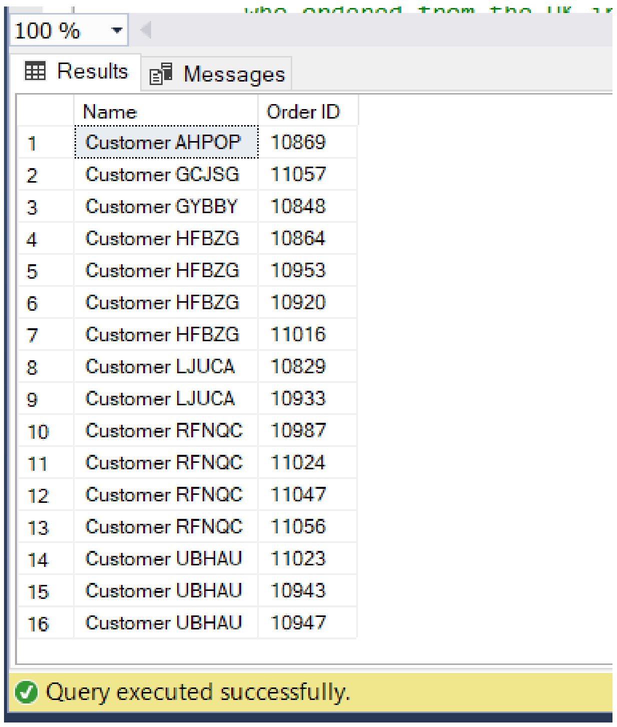
Diagrams:

Key View & Standard View



Problem solving using query:

```
--Modified
/*
Problem 3: Using Northwinds2020TSQLV6, list the names and order id of customers
           who ordered from the UK in 2016. This query will be used to count how
           many orders were sent to the UK in 2016.
*/
USE Northwinds2020TSQLV6;
SELECT C.CustomerCompanyName as [Name], O.orderid as [Order ID]
FROM Sales.[Order] as O
    INNER JOIN Sales.[Customer] as C
        ON O.CustomerId = C.CustomerId
WHERE O.ShipToCountry = N'UK' AND O.orderdate >= '20160101' AND O.orderdate < '20170101'
GROUP BY C.CustomerCompanyName, O.orderid
ORDER BY C.CustomerCompanyName;
--Uncomment below to get the JSON Output
--FOR JSON PATH, ROOT('Orders sent to UK in 2016:'), INCLUDE_NULL_VALUES;
```



	Name	Order ID
1	Customer AHPOP	10869
2	Customer GCJSG	11057
3	Customer GYBBY	10848
4	Customer HFBZG	10864
5	Customer HFBZG	10953
6	Customer HFBZG	10920
7	Customer HFBZG	11016
8	Customer LJUCA	10829
9	Customer LJUCA	10933
10	Customer RFNQC	10987
11	Customer RFNQC	11024
12	Customer RFNQC	11047
13	Customer RFNQC	11056
14	Customer UBHAU	11023
15	Customer UBHAU	10943
16	Customer UBHAU	10947

Query executed successfully.

Problem 02 (Medium Worst): Using AdventureworksDW2017

Proposition:

```
List the names and order id of customers who ordered from the UK in 2016
```

Information:

Display the OrderId, CustomerId, CustomerContactName, TotalPaid, RequiredDate, ShippedDate, and DangerZone

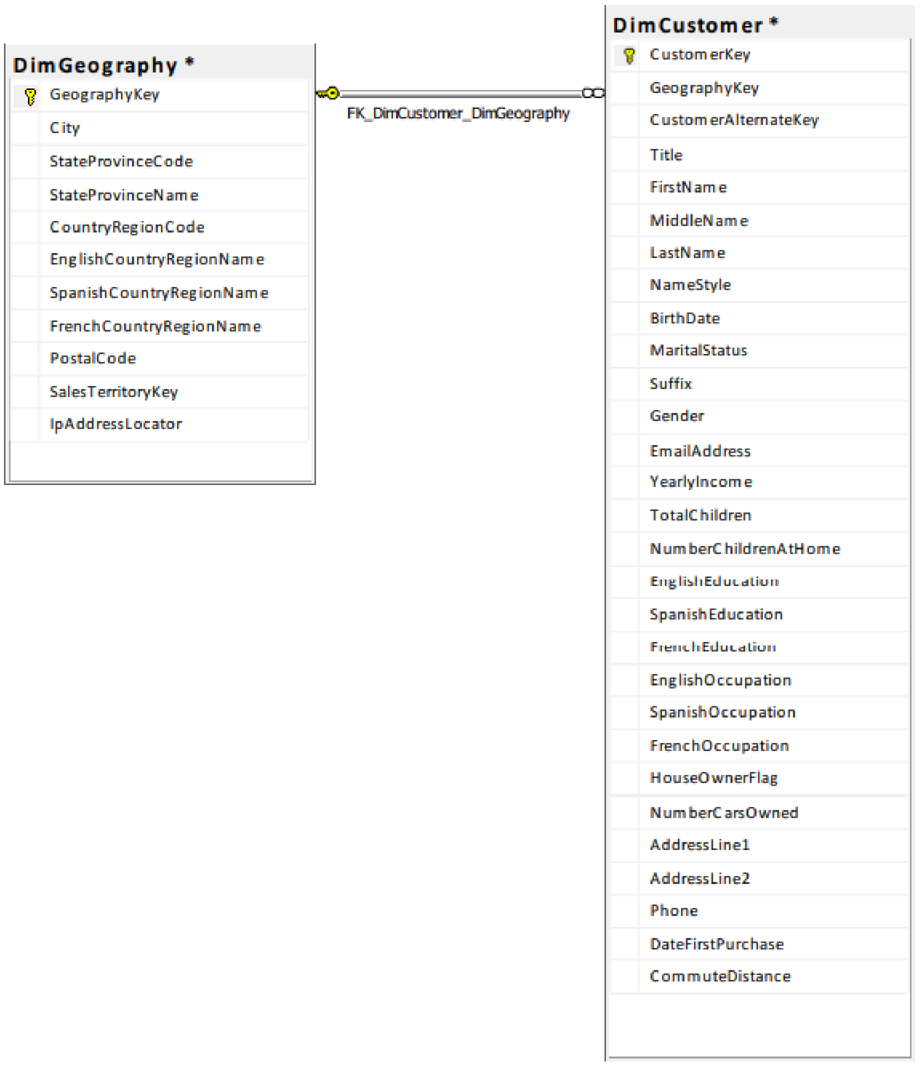
Tables Involved:

Order, Customer, OrderDetail

Diagrams:

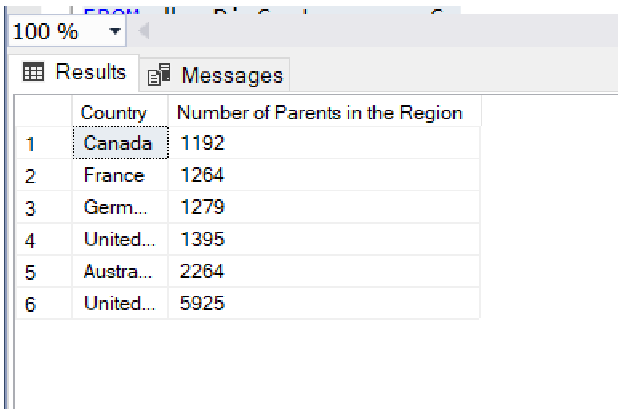
Key View & Standard View

Database Diagram: The Geography column and Customer column are joined using the Geography key.



Problem solving using query:

```
/*
 Problem 7: Using AdventureWorksDW2017, list the number of parents in each
            of the countries in ascending order of number of parents. This
            query can be used for census purposes.
*/
USE AdventureWorksDW2017;
SELECT G.EnglishCountryRegionName as [Country],
       COUNT(G.EnglishCountryRegionName) as [Number of Parents in the Region]
FROM dbo.DimCustomer as C
     INNER JOIN dbo.DimGeography as G
        ON C.GeographyKey = G.GeographyKey
GROUP BY G.EnglishCountryRegionName
ORDER BY COUNT(G.EnglishCountryRegionName);
--Uncomment below to get the JSON Output
--FOR JSON PATH, ROOT('Number of Parents:'), INCLUDE_NULL_VALUES;
```



	Country	Number of Parents in the Region
1	Canada	1192
2	France	1264
3	Germ...	1279
4	United...	1395
5	Austra...	2264
6	United...	5925

Problem 02 (Medium Worst Fixed): Using AdventureworksDW2017

Proposition:

List the names and order id of customers who ordered from the UK in 2016

Information:

Display the OrderId, CustomerId, CustomerContactName, TotalPaid, RequiredDate, ShippedDate, and DangerZone

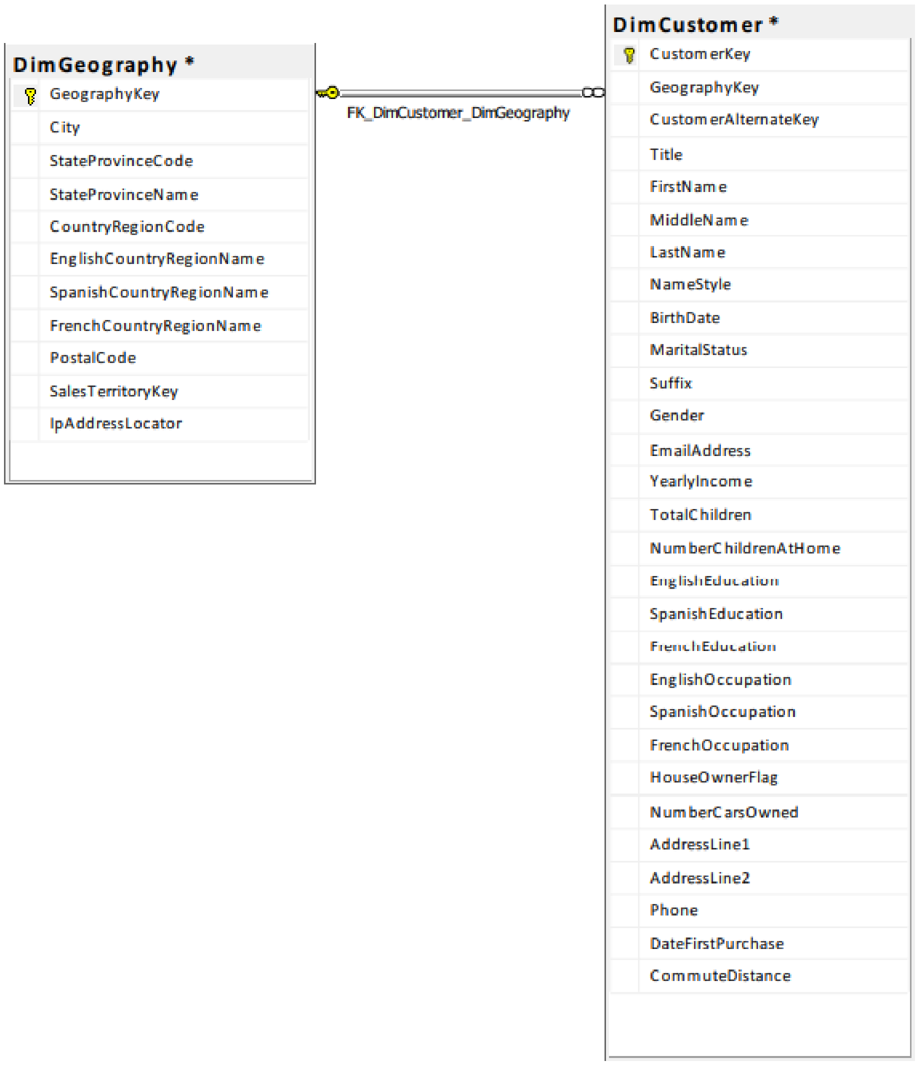
Tables Involved:

Order, Customer, OrderDetail

Diagrams:

Key View & Standard View

Database Diagram: The Geography column and Customer column are joined using the Geography key.



SELECT CLASUE Chart:

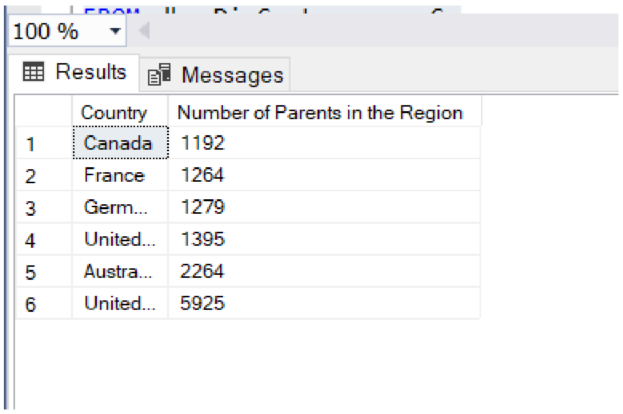
Table Name	Column Name
Orders	OrderID CustomerID RequiredDate ShipToDate
Customer	CustomerContactName
OrderDetail	UnitPrice Quantity DiscountPercentage

ORDER BY Chart:

Table Name	Column Name	Sort Order
Order	DangerZone	DESC
Order	OrderId	ASC

Problem solving using query:

```
--Modified
/*
 Problem 7: Using AdventureWorksDW2017, list the number of parents in each
            of the countries in ascending order of number of parents. This
            query can be used for census purposes.
*/
USE AdventureWorksDW2017;
SELECT G.EnglishCountryRegionName as [Country],
       COUNT(G.EnglishCountryRegionName) as [Number of Parents in the Region]
FROM dbo.DimCustomer as C
      INNER JOIN dbo.DimGeography as G
         ON C.GeographyKey = G.GeographyKey
WHERE C.TotalChildren > 0
GROUP BY G.EnglishCountryRegionName
ORDER BY COUNT(G.EnglishCountryRegionName);
--Uncomment below to get the JSON Output
--FOR JSON PATH, ROOT('Number of Parents:'), INCLUDE_NULL_VALUES;
```



```
   Country   Number of Parents in the Region
1  Canada    1192
2  France    1264
3  Germ...   1279
4  United... 1395
5  Austra... 2264
6  United... 5925
```

Problem 03 (Hard Worst): Using AdventureworksDW2017

Proposition:

List the names and order id of customers who ordered from the UK in 2016

Information:

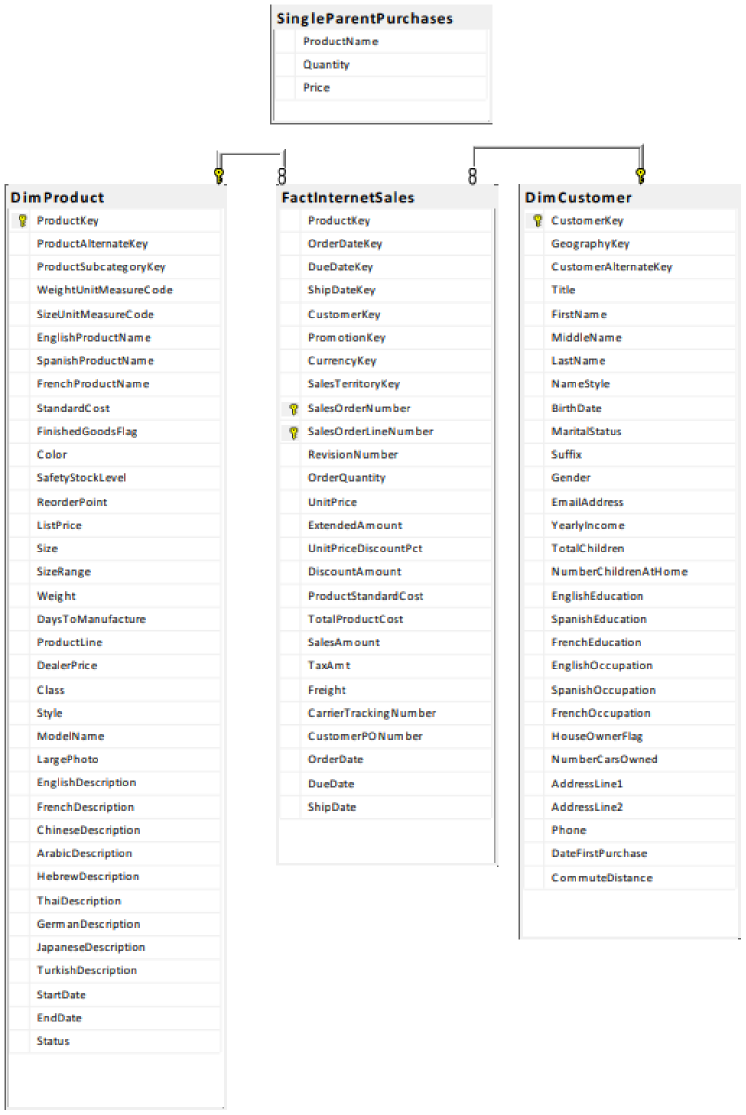
Display the OrderId, CustomerId, CustomerContactName, TotalPaid, RequiredDate, ShippedDate, and DangerZone

Tables Involved:

Order, Customer, OrderDetail

Diagrams:
Key View & Standard View

Database Diagram: The Product column and InternetSales column are joined by the Product key while the InternetSales column and Customer column are joined by the Customer key.



SELECT CLASUE Chart:

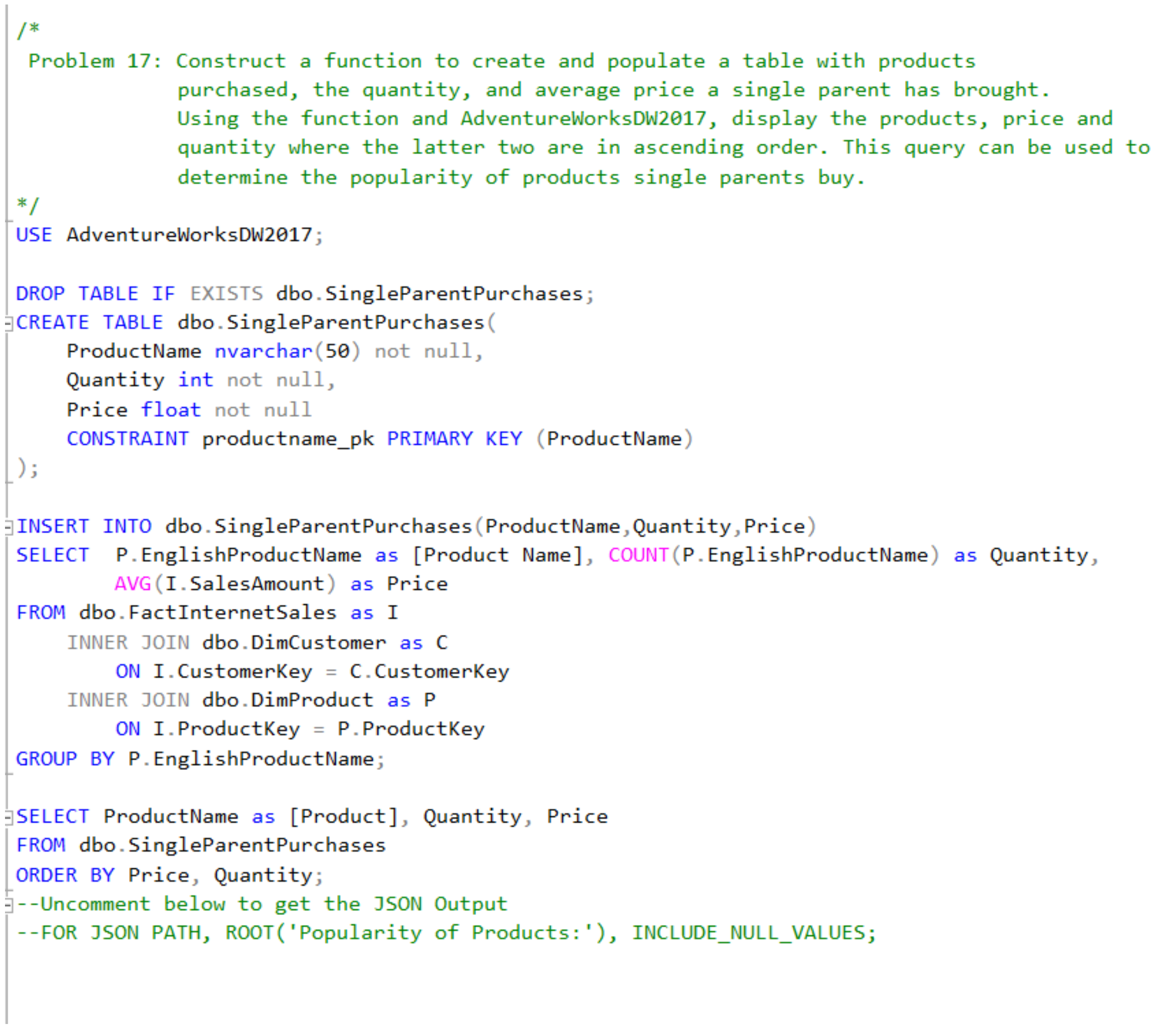
Table Name	Column Name
Orders	OrderID CustomerID RequiredDate ShipToDate
Customer	CustomerContactName
OrderDetail	UnitPrice

	Quantity DiscountPercentage

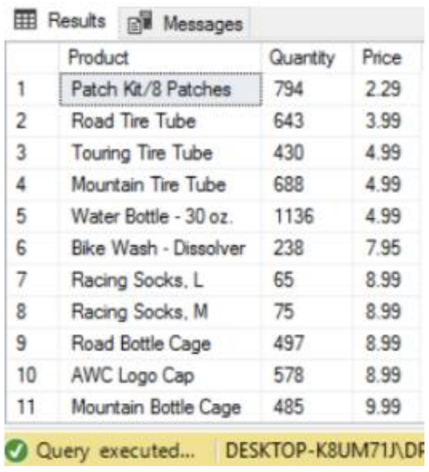
ORDER BY Chart:

Table Name	Column Name	Sort Order
Order	DangerZone	DESC
Order	OrderId	ASC

Problem solving using query:



```
/*
 Problem 17: Construct a function to create and populate a table with products
             purchased, the quantity, and average price a single parent has brought.
             Using the function and AdventureWorksDW2017, display the products, price and
             quantity where the latter two are in ascending order. This query can be used to
             determine the popularity of products single parents buy.
*/
USE AdventureWorksDW2017;

DROP TABLE IF EXISTS dbo.SingleParentPurchases;
CREATE TABLE dbo.SingleParentPurchases(
    ProductName nvarchar(50) not null,
    Quantity int not null,
    Price float not null
    CONSTRAINT productname_pk PRIMARY KEY (ProductName)
);

INSERT INTO dbo.SingleParentPurchases(ProductName,Quantity,Price)
SELECT  P.EnglishProductName as [Product Name], COUNT(P.EnglishProductName) as Quantity,
        AVG(I.SalesAmount) as Price
FROM dbo.FactInternetSales as I
    INNER JOIN dbo.DimCustomer as C
        ON I.CustomerKey = C.CustomerKey
    INNER JOIN dbo.DimProduct as P
        ON I.ProductKey = P.ProductKey
GROUP BY P.EnglishProductName;

SELECT ProductName as [Product], Quantity, Price
FROM dbo.SingleParentPurchases
ORDER BY Price, Quantity;
--Uncomment below to get the JSON Output
--FOR JSON PATH, ROOT('Popularity of Products:'), INCLUDE_NULL_VALUES;
```



Results | Messages

	Product	Quantity	Price
1	Patch Kit/8 Patches	794	2.29
2	Road Tire Tube	643	3.99
3	Touring Tire Tube	430	4.99
4	Mountain Tire Tube	688	4.99
5	Water Bottle - 30 oz.	1136	4.99
6	Bike Wash - Dissolver	238	7.95
7	Racing Socks, L	65	8.99
8	Racing Socks, M	75	8.99
9	Road Bottle Cage	497	8.99
10	AWC Logo Cap	578	8.99
11	Mountain Bottle Cage	485	9.99

Query executed... | DESKTOP-K8UM71J\DF

Problem 03 (Hard Worst Fixed): Using AdventureworksDW2017

Proposition:

`List the names and order id of customers who ordered from the UK in 2016`

Information:

Display the OrderId, CustomerId, CustomerContactName, TotalPaid, RequiredDate, ShippedDate, and DangerZone

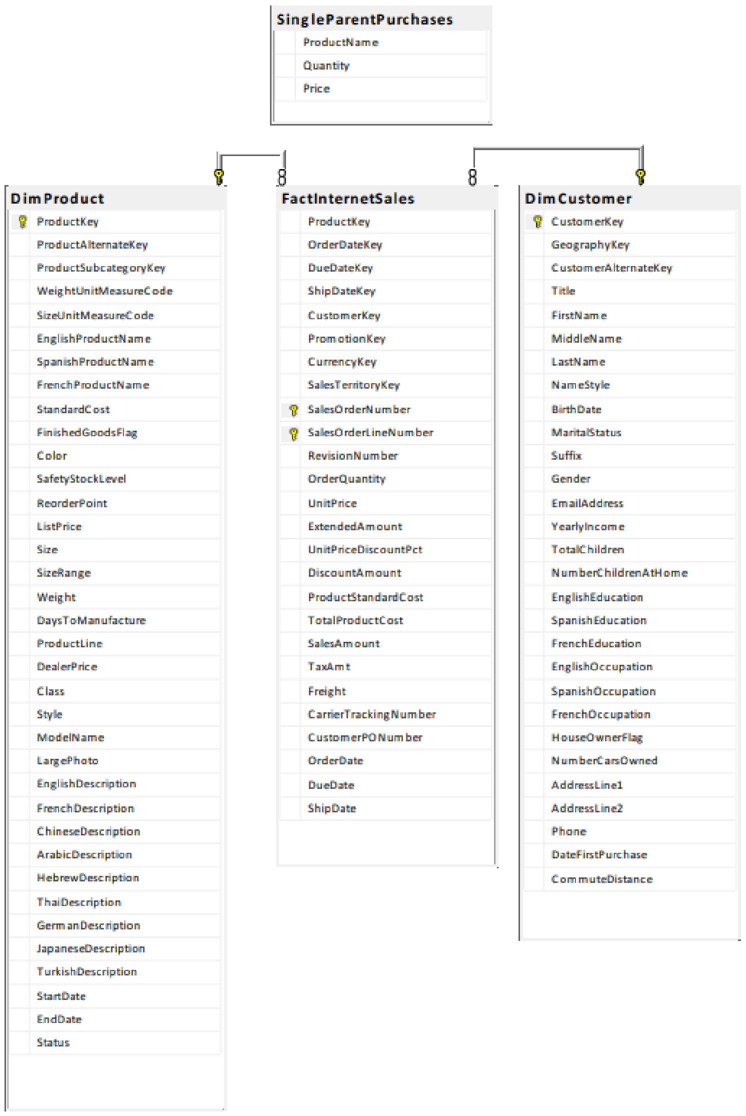
Tables Involved:

Order, Customer, OrderDetail

Diagrams:

Key View & Standard View

Database Diagram: The Product column and InternetSales column are joined by the Product key while the InternetSales column and Customer column are joined by the Customer key.



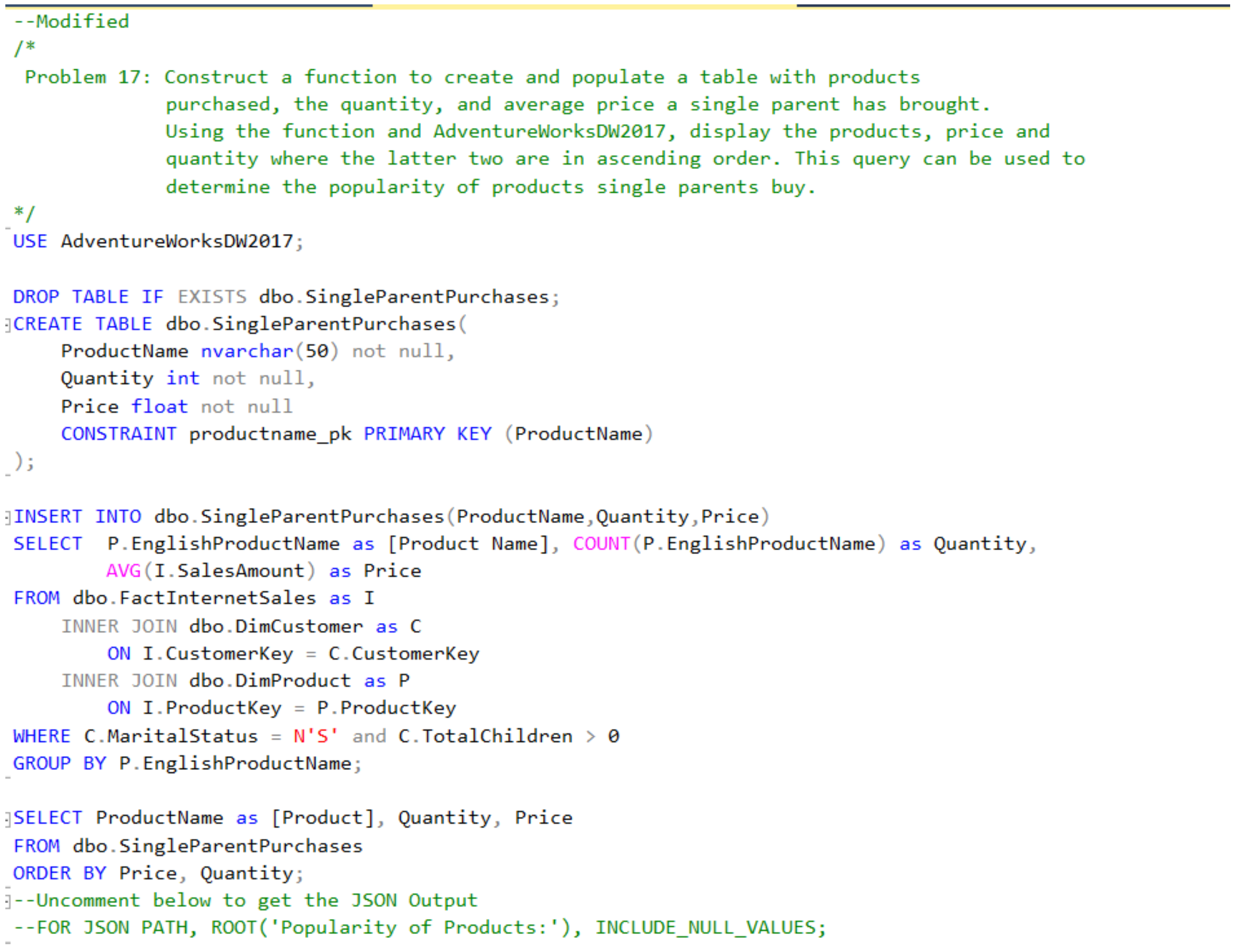
SELECT CLASUE Chart:

Table Name	Column Name
Orders	OrderID CustomerID RequiredDate ShipToDate
Customer	CustomerContactName
OrderDetail	UnitPrice Quantity DiscountPercentage

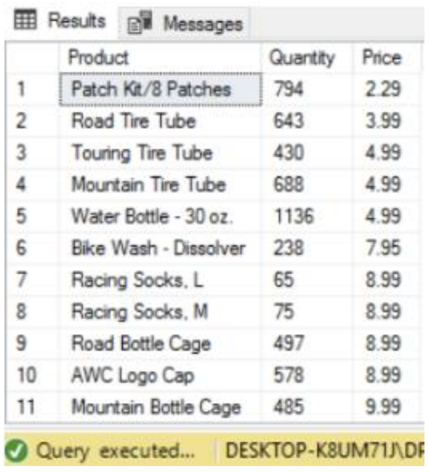
ORDER BY Chart:

Table Name	Column Name	Sort Order
Order	DangerZone	DESC
Order	OrderId	ASC

Problem solving using query:



```sql
--Modified
/*
 Problem 17: Construct a function to create and populate a table with products
             purchased, the quantity, and average price a single parent has brought.
             Using the function and AdventureWorksDW2017, display the products, price and
             quantity where the latter two are in ascending order. This query can be used to
             determine the popularity of products single parents buy.
*/
USE AdventureWorksDW2017;

DROP TABLE IF EXISTS dbo.SinglePurentPurchases;
CREATE TABLE dbo.SingleParentPurchases(
    ProductName nvarchar(50) not null,
    Quantity int not null,
    Price float not null
    CONSTRAINT productname_pk PRIMARY KEY (ProductName)
);

INSERT INTO dbo.SingleParentPurchases(ProductName,Quantity,Price)
SELECT  P.EnglishProductName as [Product Name], COUNT(P.EnglishProductName) as Quantity,
        AVG(I.SalesAmount) as Price
FROM dbo.FactInternetSales as I
    INNER JOIN dbo.DimCustomer as C
        ON I.CustomerKey = C.CustomerKey
    INNER JOIN dbo.DimProduct as P
        ON I.ProductKey = P.ProductKey
WHERE C.MaritalStatus = N'S' and C.TotalChildren > 0
GROUP BY P.EnglishProductName;

SELECT ProductName as [Product], Quantity, Price
FROM dbo.SingleParentPurchases
ORDER BY Price, Quantity;
--Uncomment below to get the JSON Output
--FOR JSON PATH, ROOT('Popularity of Products:'), INCLUDE_NULL_VALUES;
```



	Product	Quantity	Price
1	Patch Kit/8 Patches	794	2.29
2	Road Tire Tube	643	3.99
3	Touring Tire Tube	430	4.99
4	Mountain Tire Tube	688	4.99
5	Water Bottle - 30 oz.	1136	4.99
6	Bike Wash - Dissolver	238	7.95
7	Racing Socks, L	65	8.99
8	Racing Socks, M	75	8.99
9	Road Bottle Cage	497	8.99
10	AWC Logo Cap	578	8.99
11	Mountain Bottle Cage	485	9.99

```
Query executed...    DESKTOP-K8UM71J\DF
```