Project 1 Propositions

9 queries from each person:
(3) worst (3) best (3) improved

OCT 2021

ISSUED BY
10:45AM Group 4

REPRESENTATIVE
Michael Cao

Table of Contents

Proposition 1 (Best Simple)

Proposition 1: Return Tables with Employee First Name begin with A and Last Name begin with S along with Employee Key

Model Diagrams:

Figure 1A: Key View Model for Proposition 1

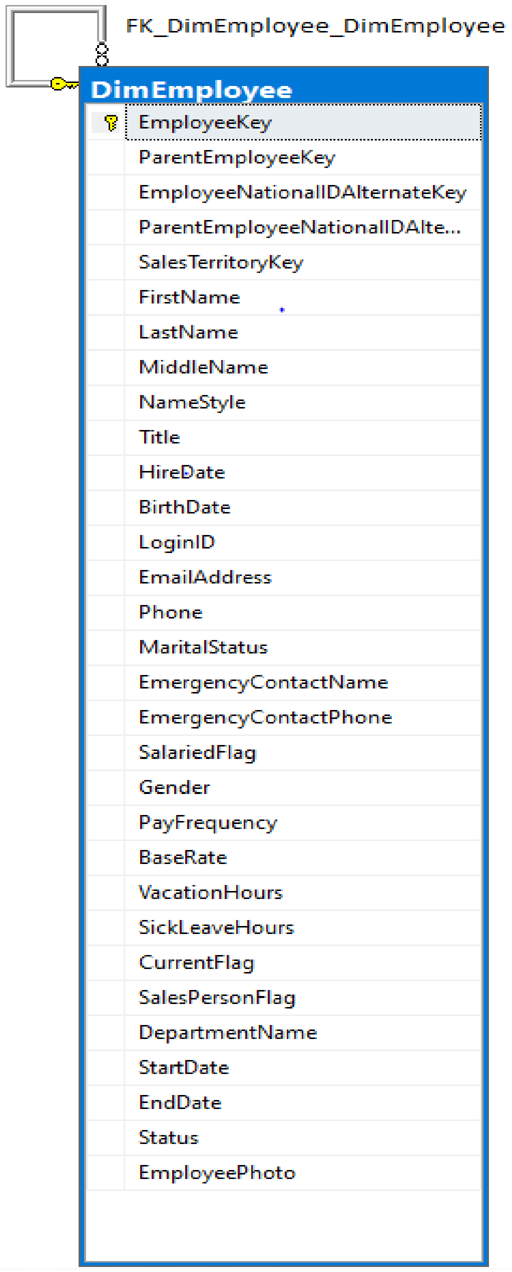
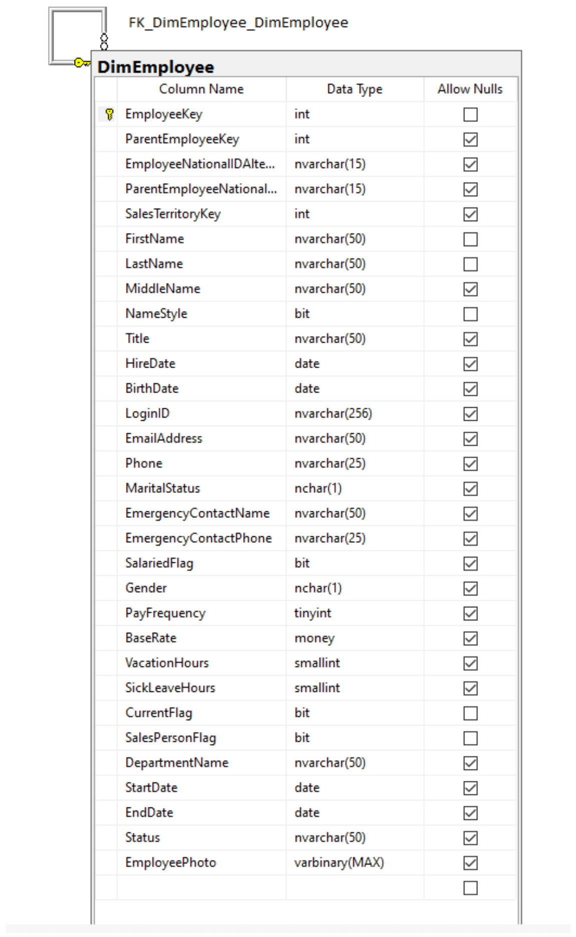


Figure 1B: Standard View Model for Proposition 1


FK_DimEmployee_DimEmployee

DimEmployee

Column Name	Data Type	Allow Nulls
EmployeeKey	int	☐
ParentEmployeeKey	int	☑
EmployeeNationalIDAlte...	nvarchar(15)	☑
ParentEmployeeNational...	nvarchar(15)	☑
SalesTerritoryKey	int	☑
FirstName	nvarchar(50)	☐
LastName	nvarchar(50)	☐
MiddleName	nvarchar(50)	☑
NameStyle	bit	☐
Title	nvarchar(50)	☑
HireDate	date	☑
BirthDate	date	☑
LoginID	nvarchar(256)	☑
EmailAddress	nvarchar(50)	☑
Phone	nvarchar(25)	☑
MaritalStatus	nchar(1)	☑
EmergencyContactName	nvarchar(50)	☑
EmergencyContactPhone	nvarchar(25)	☑
SalariedFlag	bit	☑
Gender	nchar(1)	☑
PayFrequency	tinyint	☑
BaseRate	money	☑
VacationHours	smallint	☑
SickLeaveHours	smallint	☑
CurrentFlag	bit	☐
SalesPersonFlag	bit	☐
DepartmentName	nvarchar(50)	☑
StartDate	date	☑
EndDate	date	☑
Status	nvarchar(50)	☑
EmployeePhoto	varbinary(MAX)	☑
		☐

Explanation:

Made a Function where it return only the key, first name and last name by making it focus on solely names that have a and s in first and last name respectively.

Figure 1C: Tables for SQL query components

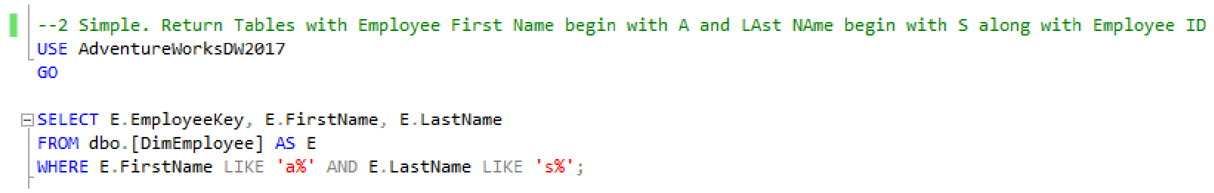
Select clause

Table name:	Column name:
DimEmployee	E.EmployeeKey, E.FirstName, E.LastName

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

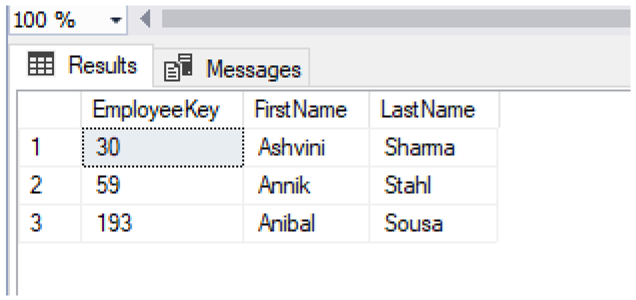
Figure 1D: Formatted SQL Query for Proposition 1



```
--2 Simple. Return Tables with Employee First Name begin with A and LAst NAme begin with S along with Employee ID
USE AdventureWorksDW2017
GO

SELECT E.EmployeeKey, E.FirstName, E.LastName
FROM dbo.[DimEmployee] AS E
WHERE E.FirstName LIKE 'a%' AND E.LastName LIKE 's%';
```

Figure 1E: Query Output for Proposition 1



100 %

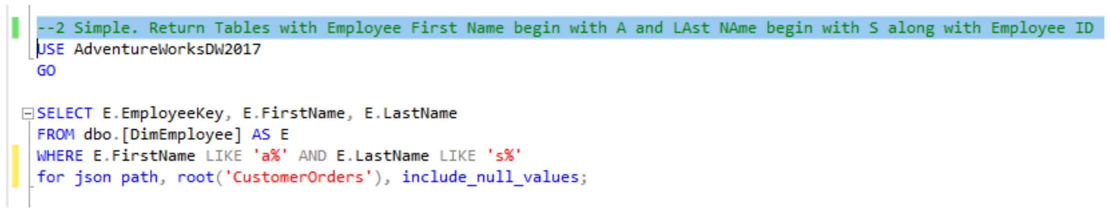
Results | Messages

	EmployeeKey	FirstName	LastName
1	30	Ashvini	Sharma
2	59	Annik	Stahl
3	193	Anibal	Sousa

JSON:

Sample JSON Output with total number of rows returned (3)

Figure 1F: Formatted SQL Query with JSON for Proposition 1



```
--2 Simple. Return Tables with Employee First Name begin with A and LAst NAme begin with S along with Employee ID
USE AdventureWorksDW2017
GO

SELECT E.EmployeeKey, E.FirstName, E.LastName
FROM dbo.[DimEmployee] AS E
WHERE E.FirstName LIKE 'a%' AND E.LastName LIKE 's%'
for json path, root('CustomerOrders'), include_null_values;
```

Figure 1G: Formatted JSON Output for Proposition 1

```
{
 "CustomerOrders":[
  {
   "BusinessEntityID":275,
   "TerritoryID":2,
      "SalesQuota":300000.0000,
      "Bonus":4100.0000,
      "CommissionPct":0.0120,
      "SalesYTD":3763178.1787,
      "SalesLastYear":1750406.4785,
      "rowguid":"1E0A7274-3064-4F58-88EE-4C6586C87169",
      "ModifiedDate":"2011-05-24T00:00:00"
    },
    {
      "BusinessEntityID":279,
      "TerritoryID":5,
      "SalesQuota":300000.0000,
      "Bonus":6700.0000,
      "CommissionPct":0.0100,
      "SalesYTD":2315185.6110,
      "SalesLastYear":1849640.9418,
      "rowguid":"52A5179D-3239-4157-AE29-17E868296DC0",
      "ModifiedDate":"2011-05-24T00:00:00"
    }
  ]
}
```

Proposition 2 (Best Medium)

Proposition 2: Return Customer ID, Order ID, Product ID, Quantity, and UnitPrice. Sorted by Customer ID

Model Diagrams:

Figure 2A: Key View Model for Proposition 2

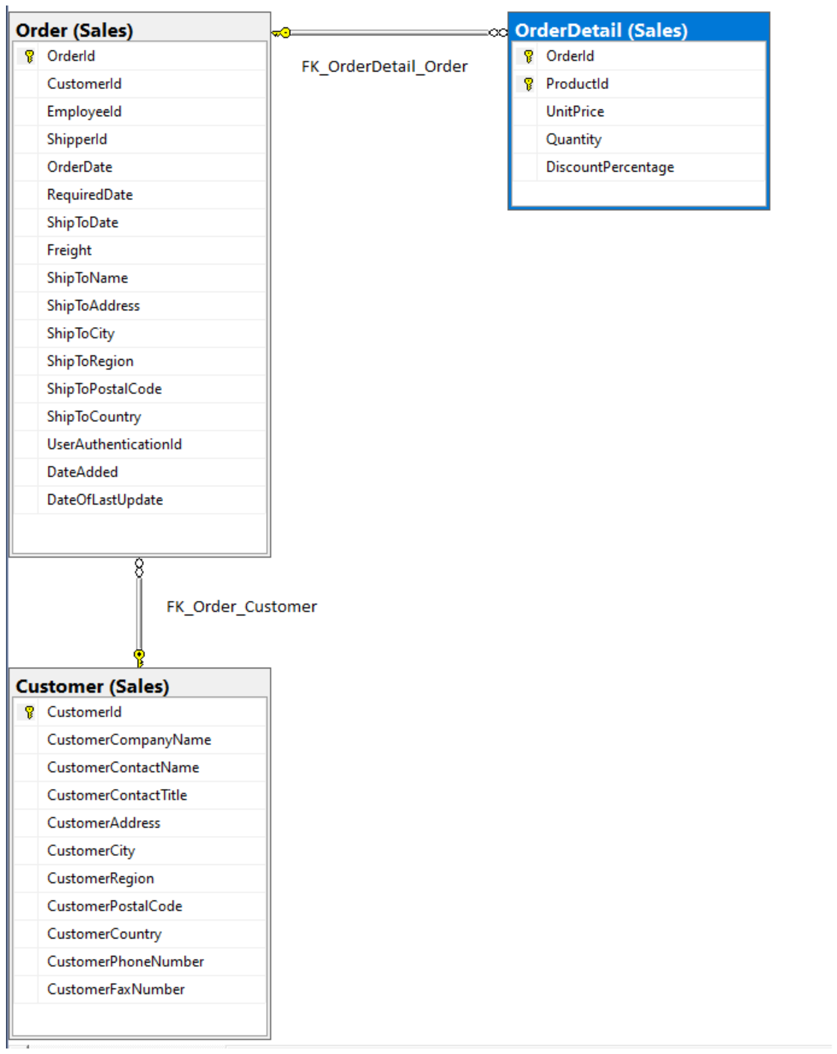
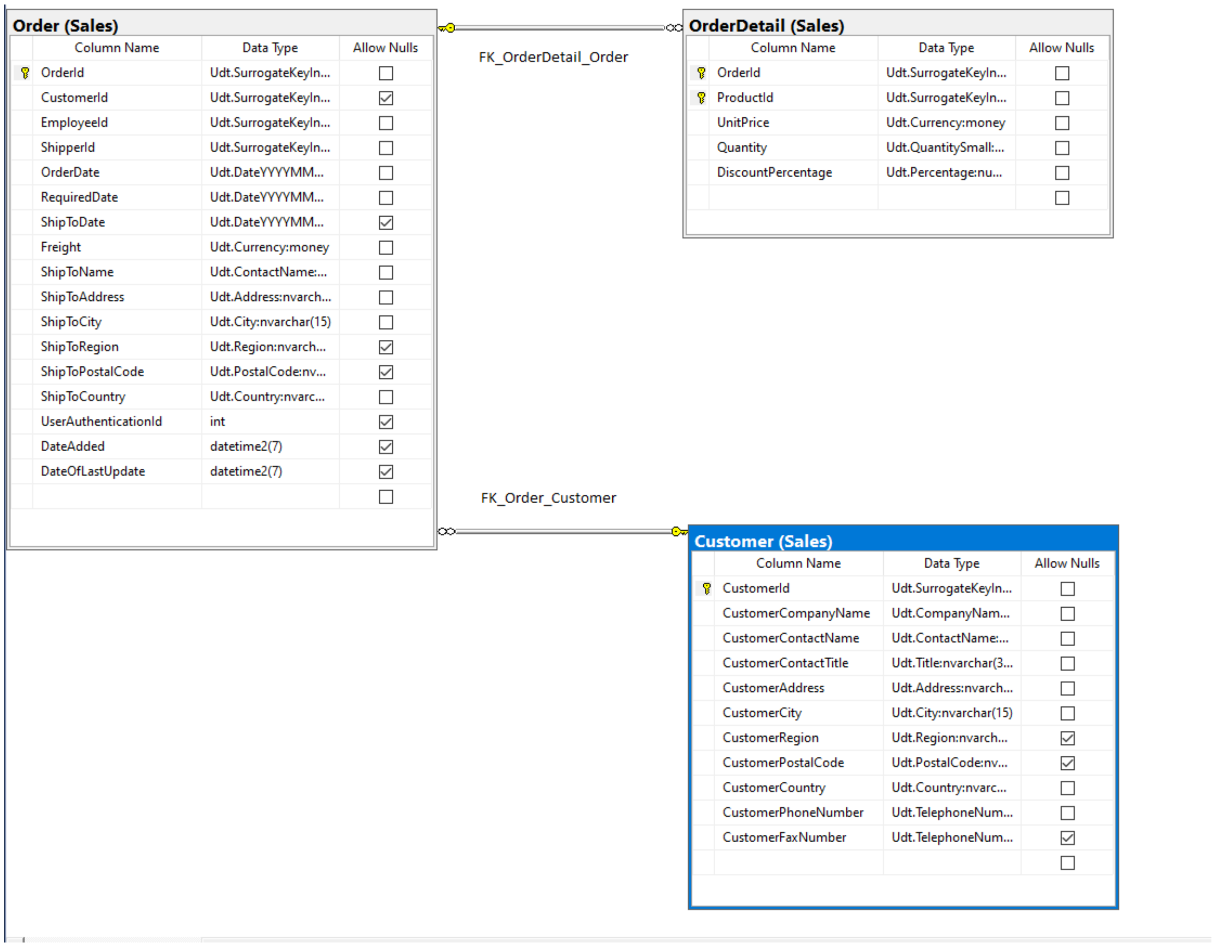


Figure 2B: Standard View Model for Proposition 2



Order (Sales)

Column Name	Data Type	Allow Nulls
OrderId	Udt.SurrogateKeyIn...	☐
CustomerId	Udt.SurrogateKeyIn...	✓
EmployeeId	Udt.SurrogateKeyIn...	☐
ShipperId	Udt.SurrogateKeyIn...	☐
OrderDate	Udt.DateYYYYMM...	☐
RequiredDate	Udt.DateYYYYMM...	☐
ShipToDate	Udt.DateYYYYMM...	✓
Freight	Udt.Currency:money	☐
ShipToName	Udt.ContactName:...	☐
ShipToAddress	Udt.Address:nvarch...	☐
ShipToCity	Udt.City:nvarchar(15)	☐
ShipToRegion	Udt.Region:nvarch...	✓
ShipToPostalCode	Udt.PostalCode:nv...	✓
ShipToCountry	Udt.Country:nvarc...	☐
UserAuthenticationId	int	✓
DateAdded	datetime2(7)	✓
DateOfLastUpdate	datetime2(7)	✓
		☐

OrderDetail (Sales)

Column Name	Data Type	Allow Nulls
OrderId	Udt.SurrogateKeyIn...	☐
ProductId	Udt.SurrogateKeyIn...	☐
UnitPrice	Udt.Currency:money	☐
Quantity	Udt.QuantitySmall:...	☐
DiscountPercentage	Udt.Percentage:nu...	☐
		☐

Customer (Sales)

Column Name	Data Type	Allow Nulls
CustomerId	Udt.SurrogateKeyIn...	☐
CustomerCompanyName	Udt.CompanyNam...	☐
CustomerContactName	Udt.ContactName:...	☐
CustomerContactTitle	Udt.Title:nvarchar(3...	☐
CustomerAddress	Udt.Address:nvarch...	☐
CustomerCity	Udt.City:nvarchar(15)	☐
CustomerRegion	Udt.Region:nvarch...	✓
CustomerPostalCode	Udt.PostalCode:nv...	✓
CustomerCountry	Udt.Country:nvarc...	☐
CustomerPhoneNumber	Udt.TelephoneNum...	☐
CustomerFaxNumber	Udt.TelephoneNum...	✓
		☐

Explanation:

Selected C.CustomerId, O.OrderId, OD.ProductId, OD.Quantity, OD.UnitPrice. Use Inner Join to combine the Sales.Order table and Sales.OrderDetails Table for the Quantity and UnitPrice. Also included in Sales.Customer for the Customer ID with the Right Outer Join

Figure 2C: Tables for SQL query components

Select clause

Table name:	Column name:
Sales.Order	O.OrderId,
Sales.OrderDetails	OD.ProductId, OD.Quantity OD.UnitPrice

Sales.Customer	C.CustomerId

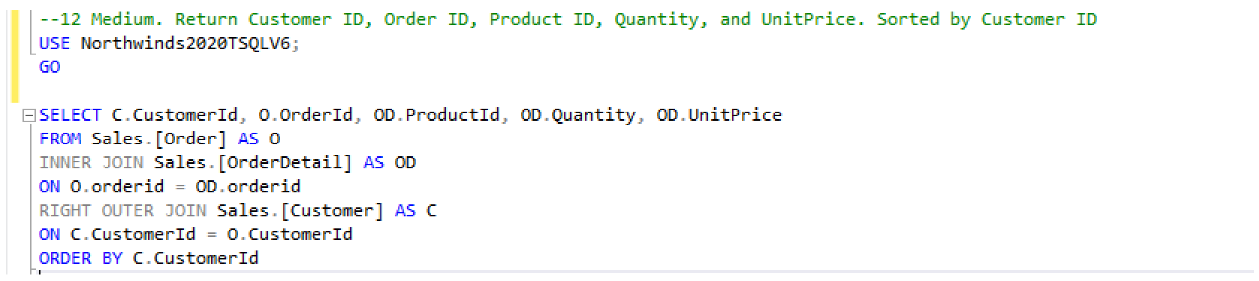
Order by (optional, only if exist)

Table name	Column name	Sort order
Sales.Customer	C.CustomerId	desc

Query:

All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

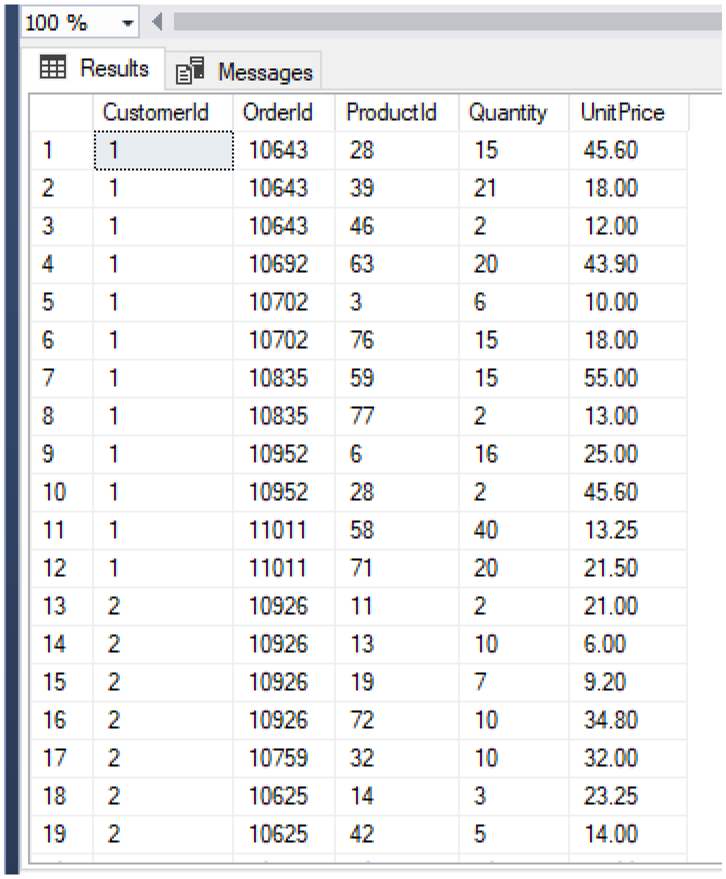
Figure 2D: Formatted SQL Query for Proposition 2



```sql
--12 Medium. Return Customer ID, Order ID, Product ID, Quantity, and UnitPrice. Sorted by Customer ID
USE Northwinds2020TSQLV6;
GO

SELECT C.CustomerId, O.OrderId, OD.ProductId, OD.Quantity, OD.UnitPrice
FROM Sales.[Order] AS O
INNER JOIN Sales.[OrderDetail] AS OD
ON O.orderid = OD.orderid
RIGHT OUTER JOIN Sales.[Customer] AS C
ON C.CustomerId = O.CustomerId
ORDER BY C.CustomerId
```

Figure 2E: Query Output for Proposition 2



100 %

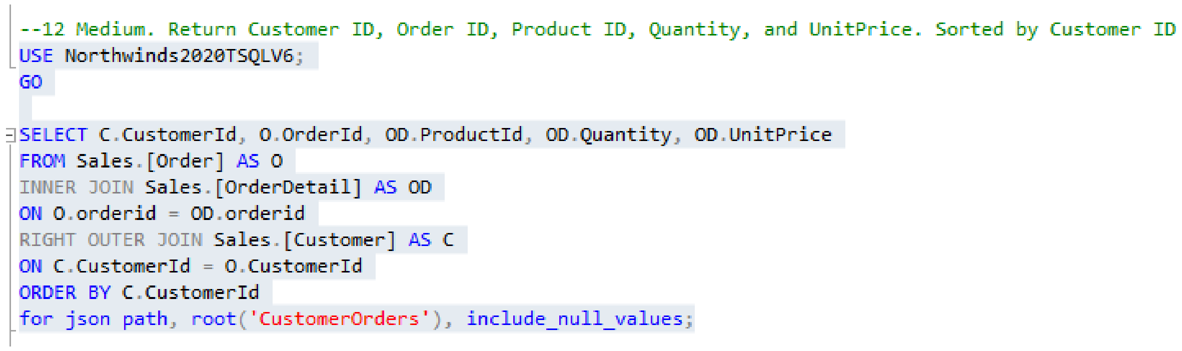
Results | Messages

	CustomerId	OrderId	ProductId	Quantity	UnitPrice
1	1	10643	28	15	45.60
2	1	10643	39	21	18.00
3	1	10643	46	2	12.00
4	1	10692	63	20	43.90
5	1	10702	3	6	10.00
6	1	10702	76	15	18.00
7	1	10835	59	15	55.00
8	1	10835	77	2	13.00
9	1	10952	6	16	25.00
10	1	10952	28	2	45.60
11	1	11011	58	40	13.25
12	1	11011	71	20	21.50
13	2	10926	11	2	21.00
14	2	10926	13	10	6.00
15	2	10926	19	7	9.20
16	2	10926	72	10	34.80
17	2	10759	32	10	32.00
18	2	10625	14	3	23.25
19	2	10625	42	5	14.00

JSON:

Sample JSON Output with total number of rows returned (2,157)

Figure 2F: Formatted SQL Query with JSON for Proposition 2



```sql
--12 Medium. Return Customer ID, Order ID, Product ID, Quantity, and UnitPrice. Sorted by Customer ID
USE Northwinds2020TSQLV6;
GO

SELECT C.CustomerId, O.OrderId, OD.ProductId, OD.Quantity, OD.UnitPrice
FROM Sales.[Order] AS O
INNER JOIN Sales.[OrderDetail] AS OD
ON O.orderid = OD.orderid
RIGHT OUTER JOIN Sales.[Customer] AS C
ON C.CustomerId = O.CustomerId
ORDER BY C.CustomerId
for json path, root('CustomerOrders'), include_null_values;
```

{Figure 2G: Formatted JSON Output for Proposition 2

{

"CustomerOrders":[

```
{
  "CustomerId":1,
  "OrderId":10643,
  "ProductId":28,
  "Quantity":15,
  "UnitPrice":45.6000
},
{
  "CustomerId":1,
  "OrderId":10643,
  "ProductId":39,
  "Quantity":21,
  "UnitPrice":18.0000
},
{
  "CustomerId":1,
  "OrderId":10643,
  "ProductId":46,
  "Quantity":2,
  "UnitPrice":12.0000
},
{
  "CustomerId":1,
  "OrderId":10692,
  "ProductId":63,
  "Quantity":20,
  "UnitPrice":43.9000
},
{
  "CustomerId":1,
  "OrderId":10702,
  "ProductId":3,
  "Quantity":6,
  "UnitPrice":10.0000
},........
```

Proposition 3 (Best Complex)

Proposition 3: Create a Function where the input returns the top 3 suppliers in USA along with Product ID and Order ID. Sorted by Supplier ID

Model Diagrams:

Figure 3A: Key View Model for Proposition 3

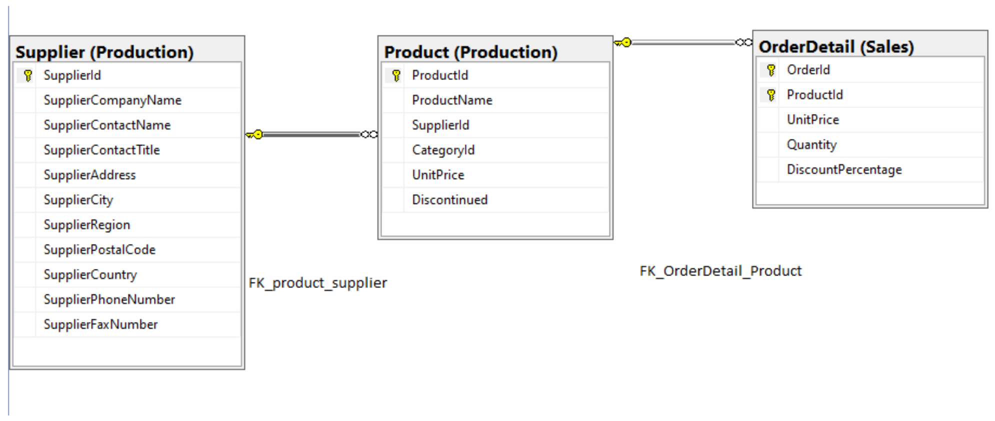
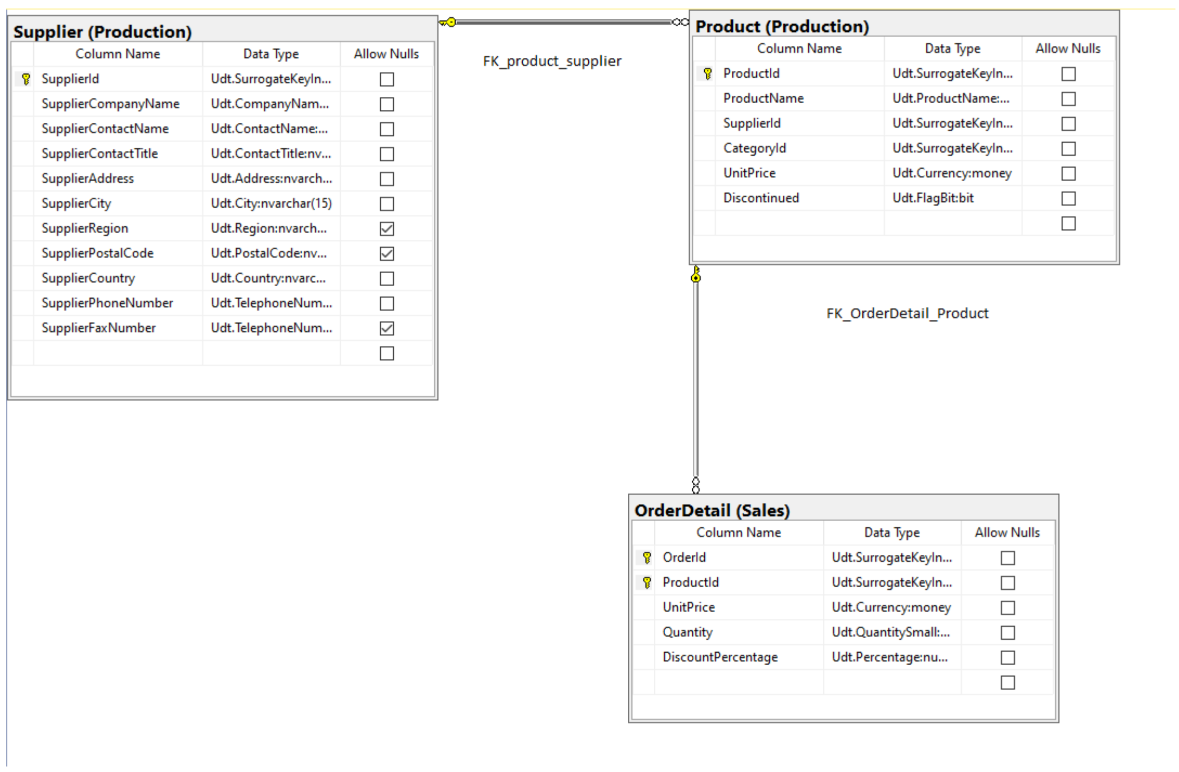


Figure 3B: Standard View Model for Proposition 3



Supplier (Production)

Column Name	Data Type	Allow Nulls
SupplierId	Udt.SurrogateKeyIn...	☐
SupplierCompanyName	Udt.CompanyNam...	☐
SupplierContactName	Udt.ContactName:...	☐
SupplierContactTitle	Udt.ContactTitle:nv...	☐
SupplierAddress	Udt.Address:nvarch...	☐
SupplierCity	Udt.City:nvarchar(15)	☐
SupplierRegion	Udt.Region:nvarch...	✓
SupplierPostalCode	Udt.PostalCode:nv...	✓
SupplierCountry	Udt.Country:nvarc...	☐
SupplierPhoneNumber	Udt.TelephoneNum...	☐
SupplierFaxNumber	Udt.TelephoneNum...	✓
		☐

Product (Production)

Column Name	Data Type	Allow Nulls
ProductId	Udt.SurrogateKeyIn...	☐
ProductName	Udt.ProductName:...	☐
SupplierId	Udt.SurrogateKeyIn...	☐
CategoryId	Udt.SurrogateKeyIn...	☐
UnitPrice	Udt.Currency:money	☐
Discontinued	Udt.FlagBit:bit	☐
		☐

OrderDetail (Sales)

Column Name	Data Type	Allow Nulls
OrderId	Udt.SurrogateKeyIn...	☐
ProductId	Udt.SurrogateKeyIn...	☐
UnitPrice	Udt.Currency:money	☐
Quantity	Udt.QuantitySmall:...	☐
DiscountPercentage	Udt.Percentage:nu...	☐
		☐

Explanation:

summary explanation that will help the developer with the proposition.

Figure 3C: Tables for SQL query components

Select clause

Table name:	Column name:
Supplier	S.SupplierId S.SupplierCompanyName, S.SupplierContactName, S.SupplierContactTitle
Production.Product	P.ProductId
OrderDetail	OD.OrderId

Order by (optional, only if exist)

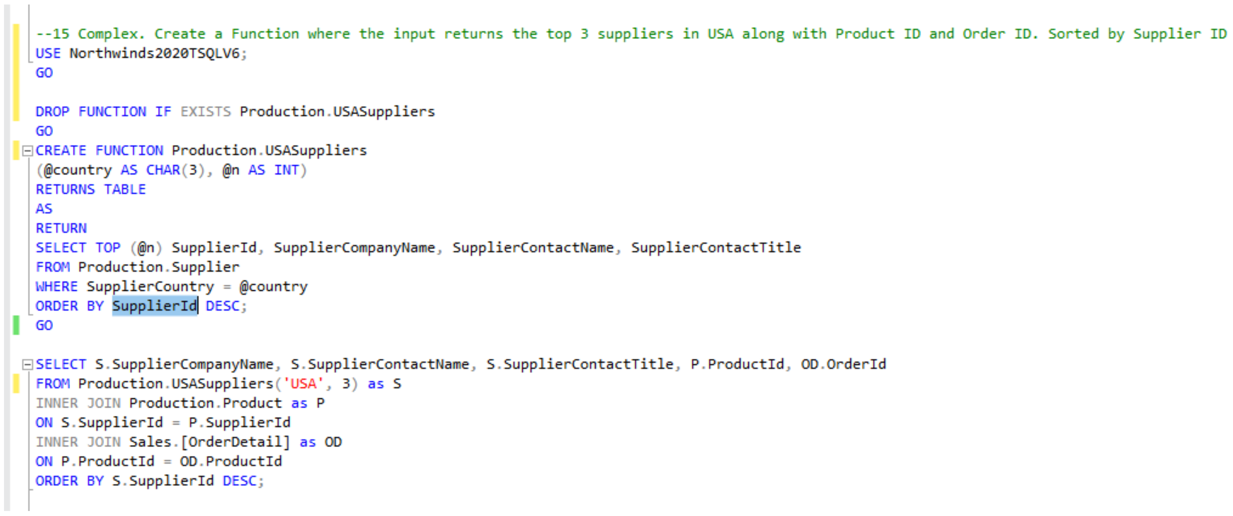
Table name	Column name	Sort order

Supplier	SupplierId	desc

Query:

All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

Figure 3D: Formatted SQL Query for Proposition 3

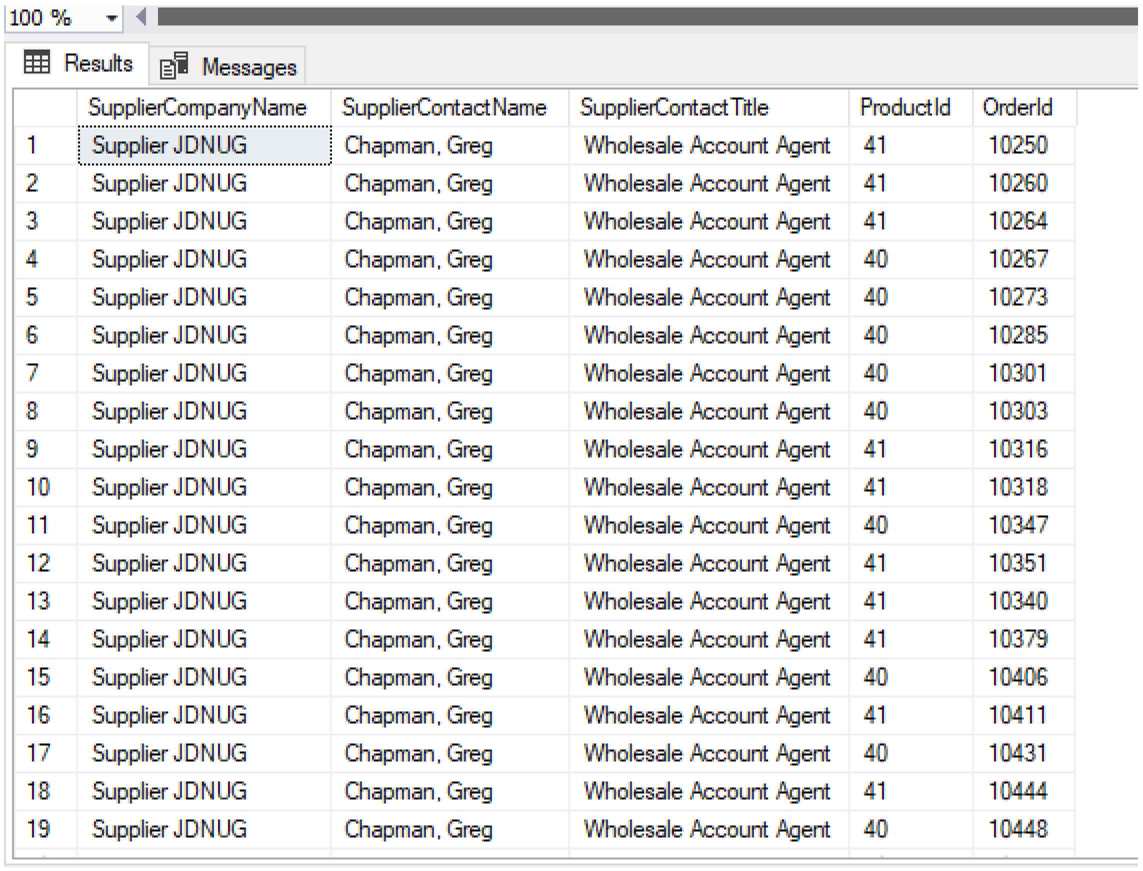


```sql
--15 Complex. Create a Function where the input returns the top 3 suppliers in USA along with Product ID and Order ID. Sorted by Supplier ID
USE Northwinds2020TSQLV6;
GO

DROP FUNCTION IF EXISTS Production.USASuppliers
GO
CREATE FUNCTION Production.USASuppliers
(@country AS CHAR(3), @n AS INT)
RETURNS TABLE
AS
RETURN
SELECT TOP (@n) SupplierId, SupplierCompanyName, SupplierContactName, SupplierContactTitle
FROM Production.Supplier
WHERE SupplierCountry = @country
ORDER BY SupplierId DESC;
GO

SELECT S.SupplierCompanyName, S.SupplierContactName, S.SupplierContactTitle, P.ProductId, OD.OrderId
FROM Production.USASuppliers('USA', 3) as S
INNER JOIN Production.Product as P
ON S.SupplierId = P.SupplierId
INNER JOIN Sales.[OrderDetail] as OD
ON P.ProductId = OD.ProductId
ORDER BY S.SupplierId DESC;
```

Figure 3E: Query Output for Proposition 3



100 %

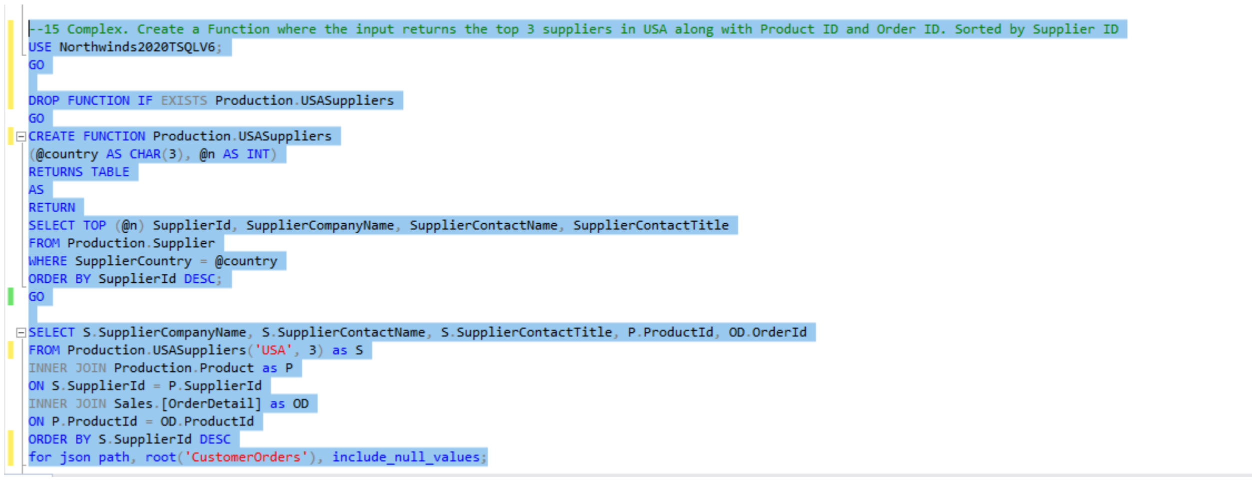
Results | Messages

	SupplierCompanyName	SupplierContactName	SupplierContactTitle	ProductId	OrderId
1	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	41	10250
2	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	41	10260
3	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	41	10264
4	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	40	10267
5	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	40	10273
6	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	40	10285
7	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	40	10301
8	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	40	10303
9	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	41	10316
10	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	41	10318
11	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	40	10347
12	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	41	10351
13	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	41	10340
14	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	41	10379
15	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	40	10406
16	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	41	10411
17	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	40	10431
18	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	41	10444
19	Supplier JDNUG	Chapman, Greg	Wholesale Account Agent	40	10448

JSON:

Sample JSON Output with total number of rows returned (207)

Figure 3F: Formatted SQL Query with JSON for Proposition 3



```sql
--15 Complex. Create a Function where the input returns the top 3 suppliers in USA along with Product ID and Order ID. Sorted by Supplier ID
USE Northwinds2020TSQLV6;
GO

DROP FUNCTION IF EXISTS Production.USASuppliers
GO
CREATE FUNCTION Production.USASuppliers
(@country AS CHAR(3), @n AS INT)
RETURNS TABLE
AS
RETURN
SELECT TOP (@n) SupplierId, SupplierCompanyName, SupplierContactName, SupplierContactTitle
FROM Production.Supplier
WHERE SupplierCountry = @country
ORDER BY SupplierId DESC;
GO

SELECT S.SupplierCompanyName, S.SupplierContactName, S.SupplierContactTitle, P.ProductId, OD.OrderId
FROM Production.USASuppliers('USA', 3) as S
INNER JOIN Production.Product as P
ON S.SupplierId = P.SupplierId
INNER JOIN Sales.[OrderDetail] as OD
ON P.ProductId = OD.ProductId
ORDER BY S.SupplierId DESC
for json path, root('CustomerOrders'), include_null_values;
```

Figure 3G: Formatted JSON Output for Proposition 3

```
    "SupplierCompanyName":"Supplier JDNUG",
    "SupplierContactName":"Chapman, Greg",
    "SupplierContactTitle":"Wholesale Account Agent",
    "ProductId":41,
    "OrderId":10340
  },
  {
    "SupplierCompanyName":"Supplier JDNUG",
    "SupplierContactName":"Chapman, Greg",
    "SupplierContactTitle":"Wholesale Account Agent",
    "ProductId":41,
    "OrderId":10379
  },
  {
    "SupplierCompanyName":"Supplier JDNUG",
    "SupplierContactName":"Chapman, Greg",
    "SupplierContactTitle":"Wholesale Account Agent",
    "ProductId":40,
    "OrderId":10406
  },
  {
    "SupplierCompanyName":"Supplier JDNUG",
    "SupplierContactName":"Chapman, Greg",
    "SupplierContactTitle":"Wholesale Account Agent",
    "ProductId":41,
    "OrderId":10411
  },..........
```

Proposition 4 (Worst Simple)

Proposition 4: Return Num of Order on each Customer ID that has more than 1 dry items. Sorted by Customer ID

Model Diagrams:

Figure 4A: Key View Model for Proposition 4

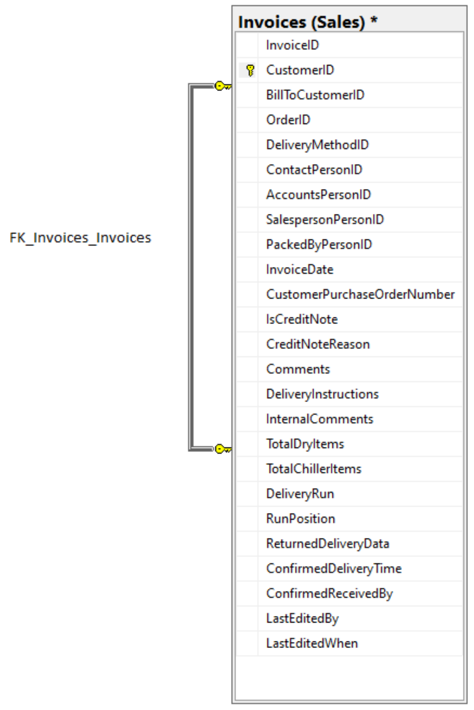
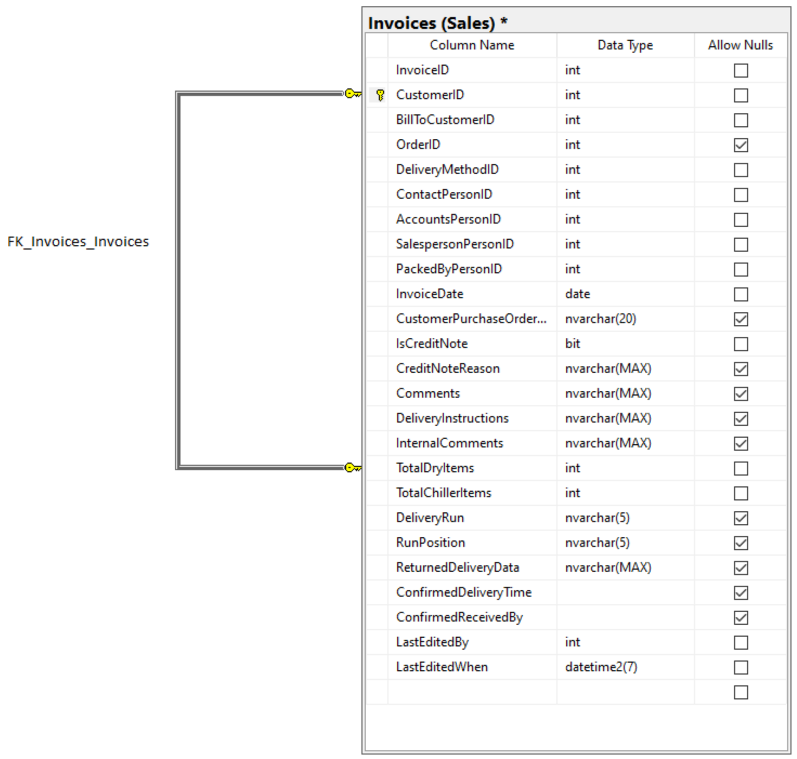


Figure 4B: Standard View Model for Proposition 4



Invoices (Sales) *

Column Name	Data Type	Allow Nulls
InvoiceID	int	☐
CustomerID	int	☐
BillToCustomerID	int	☐
OrderID	int	☑
DeliveryMethodID	int	☐
ContactPersonID	int	☐
AccountsPersonID	int	☐
SalespersonPersonID	int	☐
PackedByPersonID	int	☐
InvoiceDate	date	☐
CustomerPurchaseOrder...	nvarchar(20)	☑
IsCreditNote	bit	☐
CreditNoteReason	nvarchar(MAX)	☑
Comments	nvarchar(MAX)	☑
DeliveryInstructions	nvarchar(MAX)	☑
InternalComments	nvarchar(MAX)	☑
TotalDryItems	int	☐
TotalChillerItems	int	☐
DeliveryRun	nvarchar(5)	☑
RunPosition	nvarchar(5)	☑
ReturnedDeliveryData	nvarchar(MAX)	☑
ConfirmedDeliveryTime		☑
ConfirmedReceivedBy		☑
LastEditedBy	int	☐
LastEditedWhen	datetime2(7)	☐
		☐

Explanation:

Selected CustomerID and NumOrder which is made from Total Count. Had it returned only tables that have TotalDryItems greater than 1. This query took longer than expected since I made it greater than 1 rather than greater '1'.

Figure 4C: Tables for SQL query components

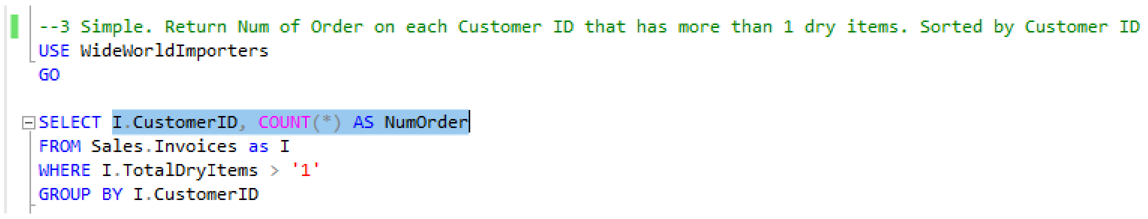
Select clause

Table name:	Column name:
Sales.Invoices	I.CustomerID COUNT(*) AS NumOrder

Query:

All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

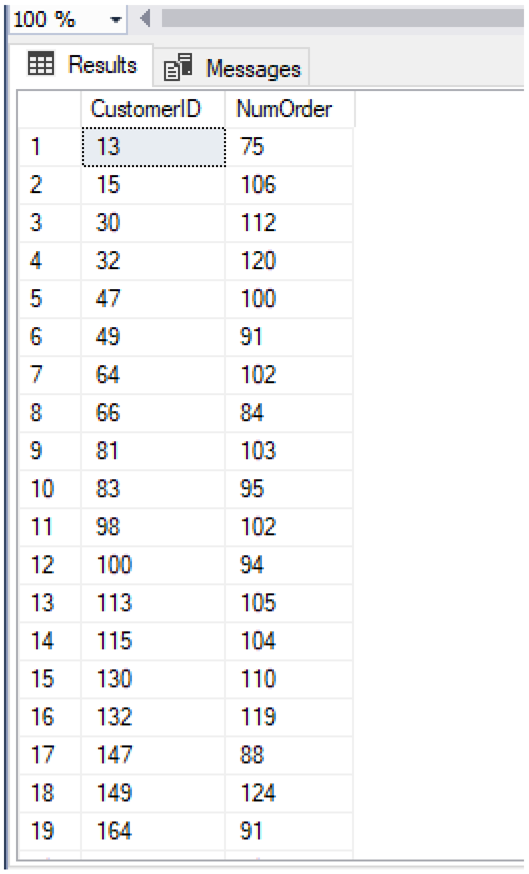
Figure 4D: Formatted SQL Query for Proposition 4



```
--3 Simple. Return Num of Order on each Customer ID that has more than 1 dry items. Sorted by Customer ID
USE WideWorldImporters
GO

SELECT I.CustomerID, COUNT(*) AS NumOrder
FROM Sales.Invoices as I
WHERE I.TotalDryItems > '1'
GROUP BY I.CustomerID
```

Figure 4E: Query Output for Proposition 4

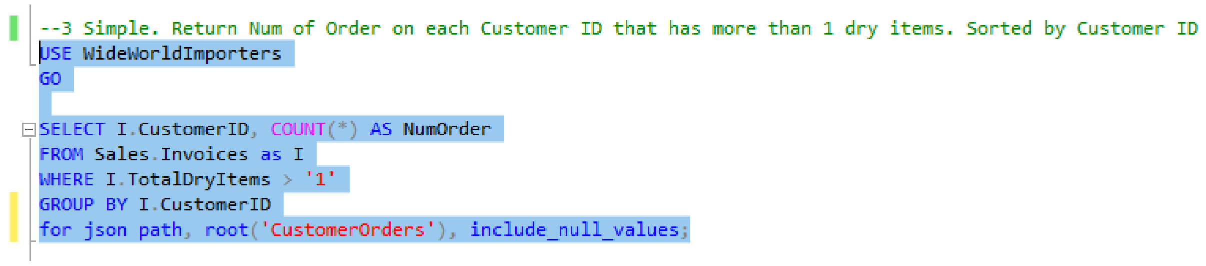


	CustomerID	NumOrder
1	13	75
2	15	106
3	30	112
4	32	120
5	47	100
6	49	91
7	64	102
8	66	84
9	81	103
10	83	95
11	98	102
12	100	94
13	113	105
14	115	104
15	130	110
16	132	119
17	147	88
18	149	124
19	164	91

JSON:

Sample JSON Output with total number of rows returned (663)

Figure 4F: Formatted SQL Query with JSON for Proposition 4



```
--3 Simple. Return Num of Order on each Customer ID that has more than 1 dry items. Sorted by Customer ID
USE WideWorldImporters
GO

SELECT I.CustomerID, COUNT(*) AS NumOrder
FROM Sales.Invoices as I
WHERE I.TotalDryItems > '1'
GROUP BY I.CustomerID
for json path, root('CustomerOrders'), include_null_values;
```

Figure 4G: Formatted JSON Output for Proposition 4

```
{
 "CustomerOrders":[
   {
     "CustomerID":13,
     "NumOrder":75
   },
   {
     "CustomerID":15,
     "NumOrder":106
   },
   {
     "CustomerID":30,
     "NumOrder":112
   },
   {
     "CustomerID":32,
     "NumOrder":120
   },
   {
     "CustomerID":47,
     "NumOrder":100
   },
   {
     "CustomerID":49,
     "NumOrder":91
   },
```

Proposition 5 (Worst Medium)

Proposition 5: Return all customers, and for each return a Yes/No value depending on if the order is from Brazil. Sorted by OrderId

Model Diagrams:

Figure 5A: Key View Model for Proposition 5

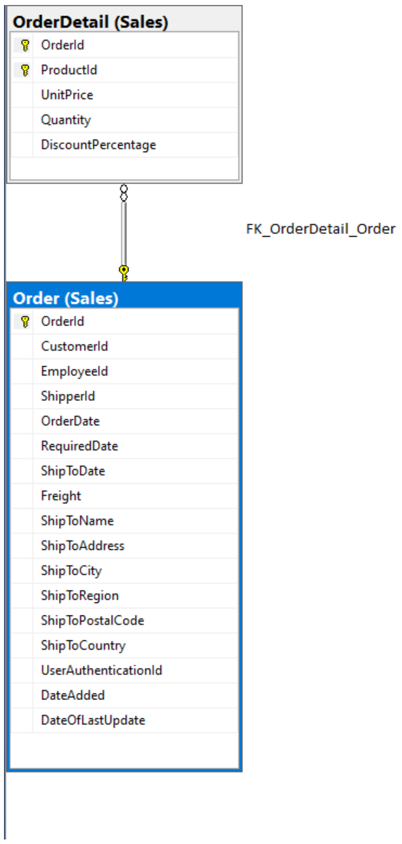
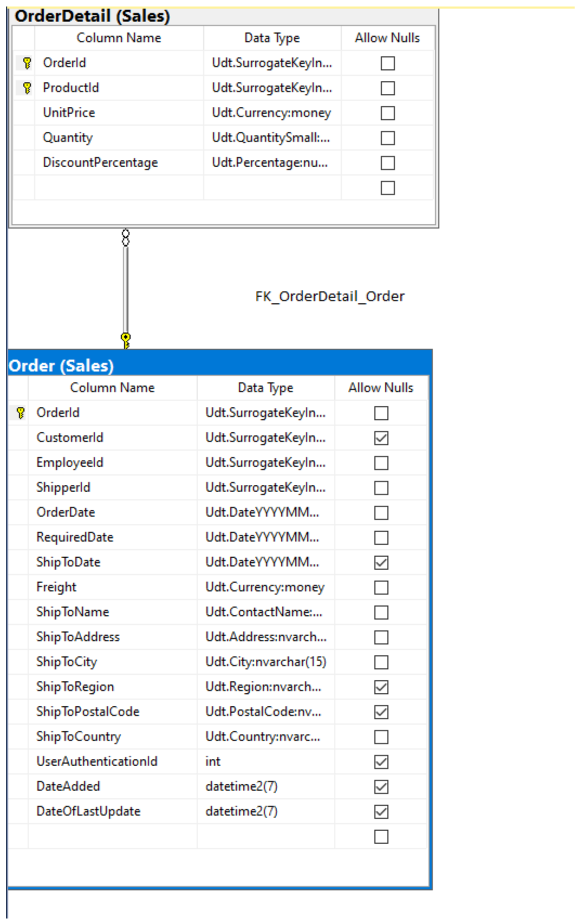


Figure 5B: Standard View Model for Proposition 5


OrderDetail (Sales)

Column Name	Data Type	Allow Nulls
OrderId	Udt.SurrogateKeyIn...	☐
ProductId	Udt.SurrogateKeyIn...	☐
UnitPrice	Udt.Currency:money	☐
Quantity	Udt.QuantitySmall:...	☐
DiscountPercentage	Udt.Percentage:nu...	☐
		☐

Order (Sales)

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

Explanation:

Selected OrderId and ProductId. Create a new Column that takes in a Yes/No Value if OrderId is Null. Had trouble with this part since I forgot to apply the null value. Then create a Left Outer Join on a Country that specifically has Brazil. That way there will create null values that I can take advantage of as a No value while Non-Null Values are Yes.

Figure 5C: Tables for SQL query components

Select clause

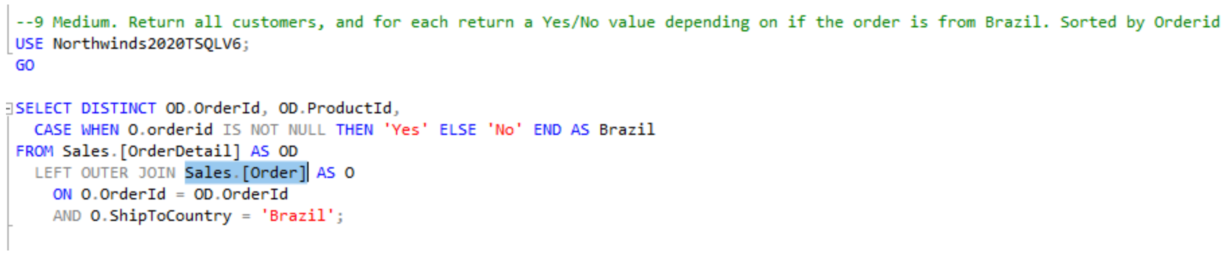
Table name:	Column name:
Sales.[OrderDetail]	OD.OrderId OD.ProductId Brazil

Sales.[Order]	Brazil

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

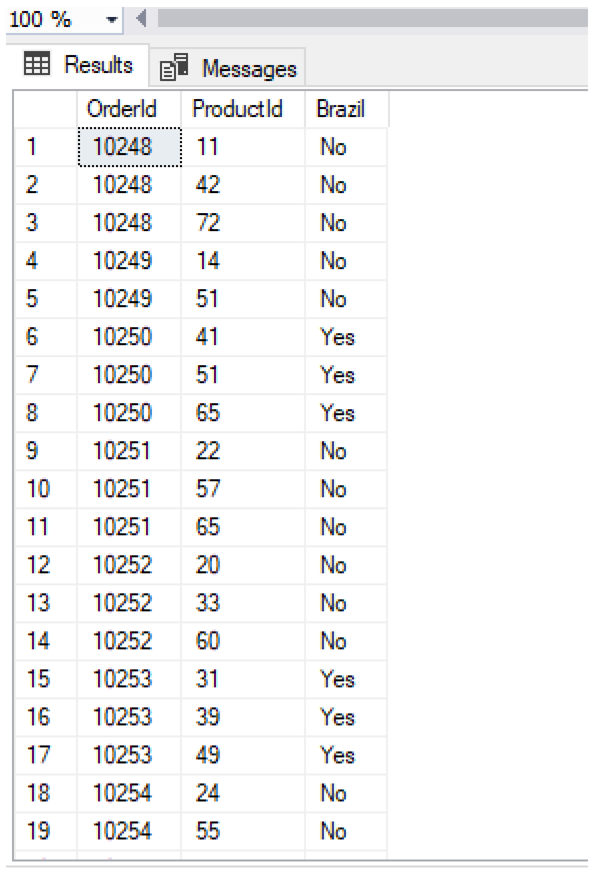
Figure 5D: Formatted SQL Query for Proposition 5



```
--9 Medium. Return all customers, and for each return a Yes/No value depending on if the order is from Brazil. Sorted by Orderid
USE Northwinds2020TSQLV6;
GO

SELECT DISTINCT OD.OrderId, OD.ProductId,
  CASE WHEN O.orderid IS NOT NULL THEN 'Yes' ELSE 'No' END AS Brazil
FROM Sales.[OrderDetail] AS OD
  LEFT OUTER JOIN Sales.[Order] AS O
    ON O.OrderId = OD.OrderId
    AND O.ShipToCountry = 'Brazil';
```

Figure 5E: Query Output for Proposition 5

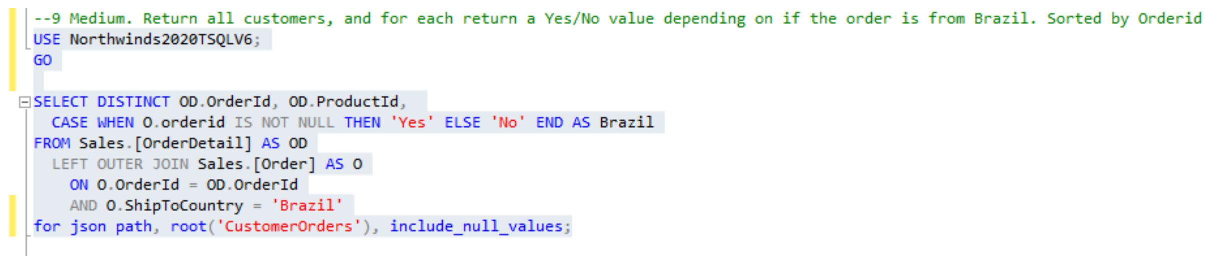


	OrderId	ProductId	Brazil
1	10248	11	No
2	10248	42	No
3	10248	72	No
4	10249	14	No
5	10249	51	No
6	10250	41	Yes
7	10250	51	Yes
8	10250	65	Yes
9	10251	22	No
10	10251	57	No
11	10251	65	No
12	10252	20	No
13	10252	33	No
14	10252	60	No
15	10253	31	Yes
16	10253	39	Yes
17	10253	49	Yes
18	10254	24	No
19	10254	55	No

JSON:

Sample JSON Output with total number of rows returned (2155)

Figure 5F: Formatted SQL Query with JSON for Proposition 5



```
--9 Medium. Return all customers, and for each return a Yes/No value depending on if the order is from Brazil. Sorted by Orderid
USE Northwinds2020TSQLV6;
GO

SELECT DISTINCT OD.OrderId, OD.ProductId,
  CASE WHEN O.orderid IS NOT NULL THEN 'Yes' ELSE 'No' END AS Brazil
FROM Sales.[OrderDetail] AS OD
  LEFT OUTER JOIN Sales.[Order] AS O
    ON O.OrderId = OD.OrderId
    AND O.ShipToCountry = 'Brazil'
for json path, root('CustomerOrders'), include_null_values;
```

Figure 5G: Formatted JSON Output for Proposition 5

```
{
 "CustomerOrders":[
   {
     "OrderId":10248,
     "ProductId":11,
     "Brazil":"No"
   },
   {
     "OrderId":10248,
     "ProductId":42,
     "Brazil":"No"
   },
   {
     "OrderId":10248,
     "ProductId":72,
     "Brazil":"No"
   },
   {
     "OrderId":10249,
     "ProductId":14,
     "Brazil":"No"
   },
   {
     "OrderId":10249,
     "ProductId":51,
     "Brazil":"No"
   },.......
```

Proposition 6 (Worst Complex)

Proposition 6: Create Query That Returns both the Total Sum of Distinct Stock Item Key and Sum of Purchase Key

Model Diagrams:

Figure 6A: Key View Model for Proposition 6

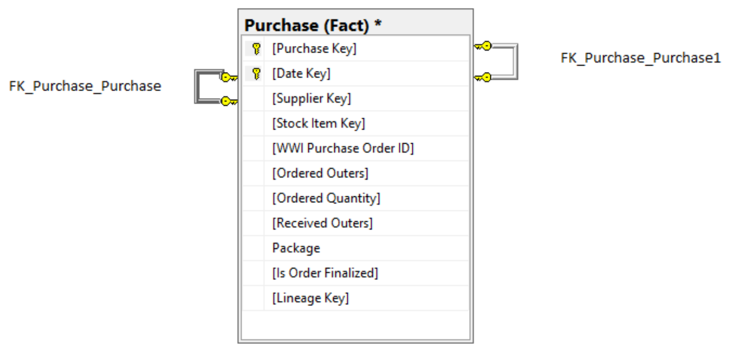
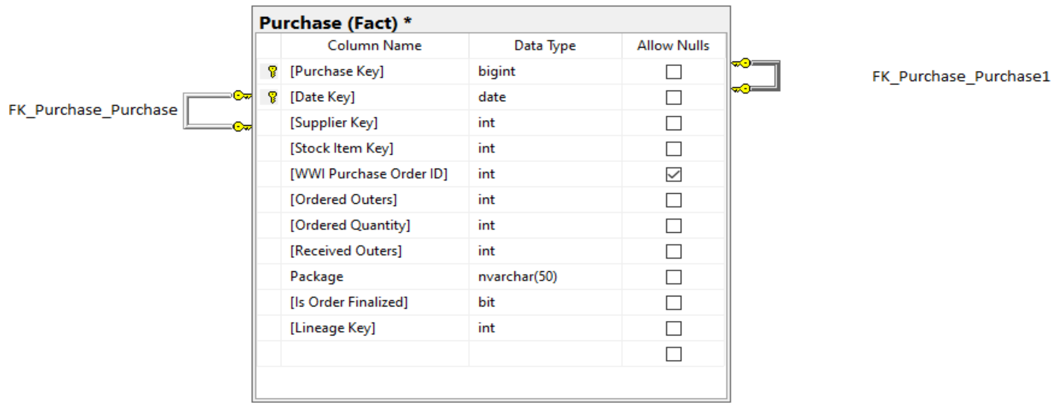


Figure 6B: Standard View Model for Proposition 6



Purchase (Fact) *

	Column Name	Data Type	Allow Nulls
🔑	[Purchase Key]	bigint	☐
🔑	[Date Key]	date	☐
	[Supplier Key]	int	☐
	[Stock Item Key]	int	☐
	[WWI Purchase Order ID]	int	☑
	[Ordered Outers]	int	☐
	[Ordered Quantity]	int	☐
	[Received Outers]	int	☐
	Package	nvarchar(50)	☐
	[Is Order Finalized]	bit	☐
	[Lineage Key]	int	☐
			☐

Explanation:

Create Multiple Queries in an Attempt to create a simple table. First Query makes a table. Second Query creates a Distinct Count on each Stock Key. Third Query Takes in Total Count of each Product Key. Then finally made an attempt to make a table that returns year along with the amount of distinct Stock Key and total Product Key. My result is not what I expected hence making this my worst Proposition.

Figure 6C: Tables for SQL query components

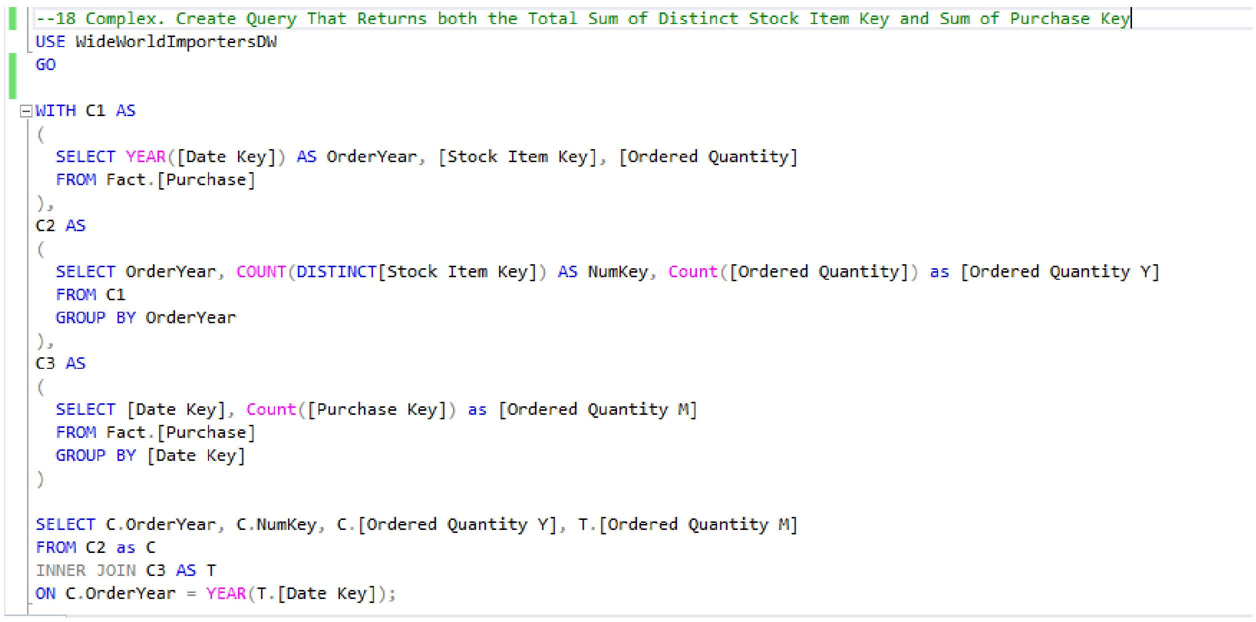
Select clause

Table name:	Column name:
Fact.[Purchase]	OrderYear, [Stock Item Key] [Ordered Quantity] [Ordered Quantity Y] [Ordered Quantity M]

Query:

All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

Figure 6D: Formatted SQL Query for Proposition 6

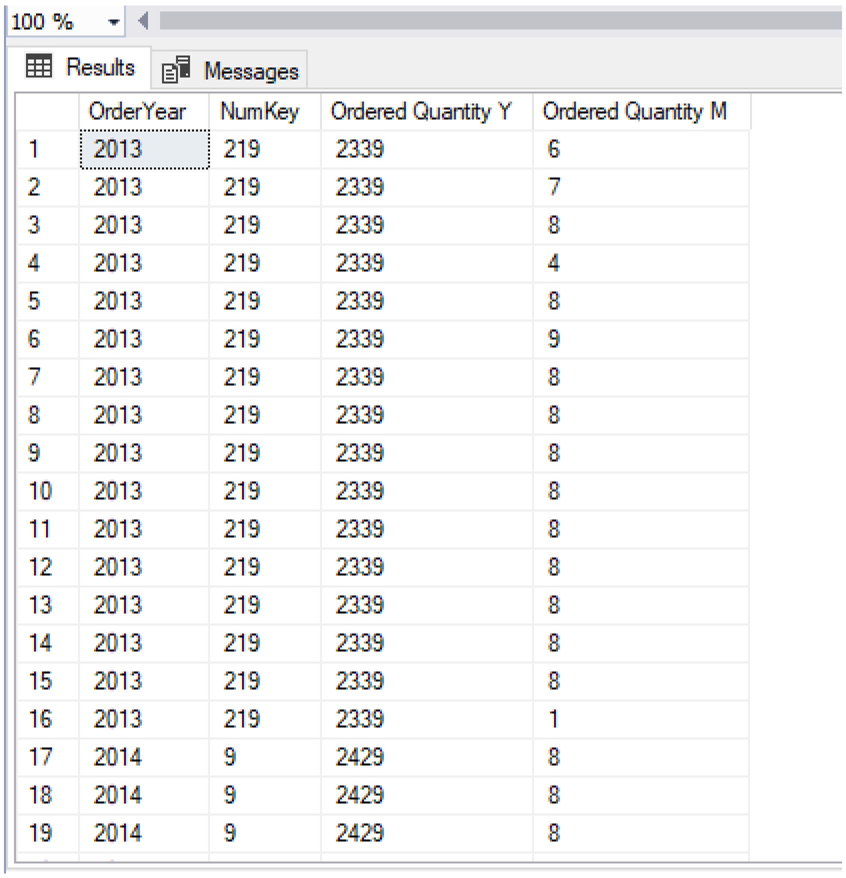


```
--18 Complex. Create Query That Returns both the Total Sum of Distinct Stock Item Key and Sum of Purchase Key
USE WideWorldImportersDW
GO

WITH C1 AS
(
  SELECT YEAR([Date Key]) AS OrderYear, [Stock Item Key], [Ordered Quantity]
  FROM Fact.[Purchase]
),
C2 AS
(
  SELECT OrderYear, COUNT(DISTINCT[Stock Item Key]) AS NumKey, Count([Ordered Quantity]) as [Ordered Quantity Y]
  FROM C1
  GROUP BY OrderYear
),
C3 AS
(
  SELECT [Date Key], Count([Purchase Key]) as [Ordered Quantity M]
  FROM Fact.[Purchase]
  GROUP BY [Date Key]
)

SELECT C.OrderYear, C.NumKey, C.[Ordered Quantity Y], T.[Ordered Quantity M]
FROM C2 as C
INNER JOIN C3 AS T
ON C.OrderYear = YEAR(T.[Date Key]);
```

Figure 6E: Query Output for Proposition 6

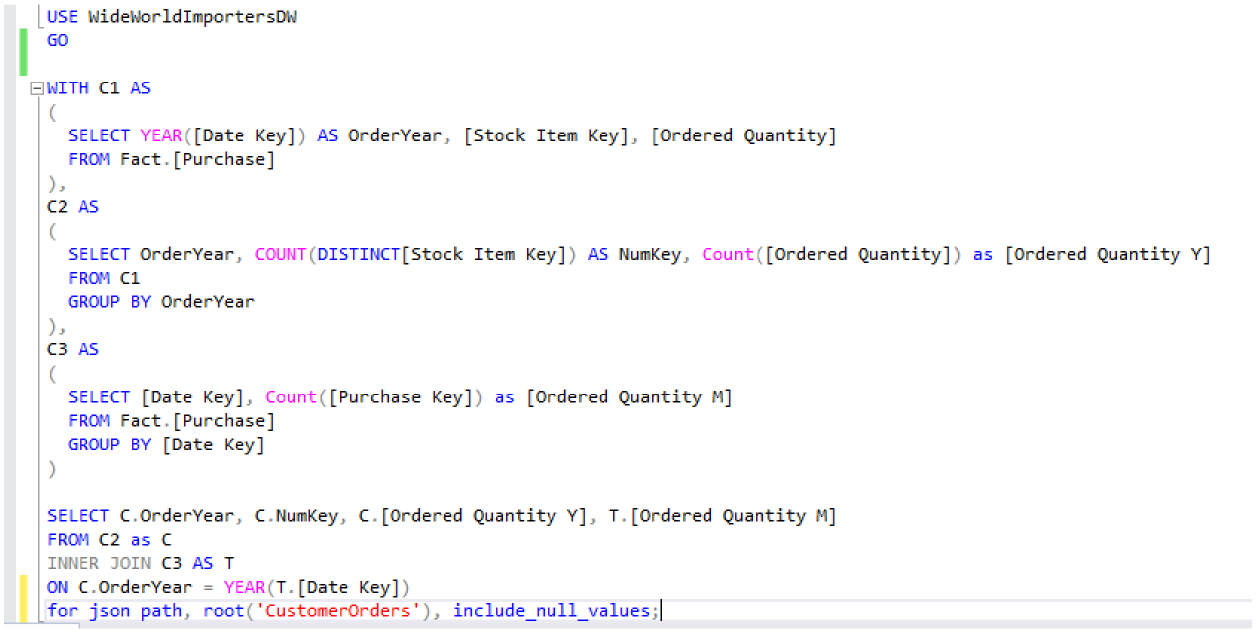


	OrderYear	NumKey	Ordered Quantity Y	Ordered Quantity M
1	2013	219	2339	6
2	2013	219	2339	7
3	2013	219	2339	8
4	2013	219	2339	4
5	2013	219	2339	8
6	2013	219	2339	9
7	2013	219	2339	8
8	2013	219	2339	8
9	2013	219	2339	8
10	2013	219	2339	8
11	2013	219	2339	8
12	2013	219	2339	8
13	2013	219	2339	8
14	2013	219	2339	8
15	2013	219	2339	8
16	2013	219	2339	1
17	2014	9	2429	8
18	2014	9	2429	8
19	2014	9	2429	8

JSON:

Sample JSON Output with total number of rows returned (1064)

Figure 6F: Formatted SQL Query with JSON for Proposition 6


```
USE WideWorldImportersDW
GO

WITH C1 AS
(
  SELECT YEAR([Date Key]) AS OrderYear, [Stock Item Key], [Ordered Quantity]
  FROM Fact.[Purchase]
),
C2 AS
(
  SELECT OrderYear, COUNT(DISTINCT[Stock Item Key]) AS NumKey, Count([Ordered Quantity]) as [Ordered Quantity Y]
  FROM C1
  GROUP BY OrderYear
),
C3 AS
(
  SELECT [Date Key], Count([Purchase Key]) as [Ordered Quantity M]
  FROM Fact.[Purchase]
  GROUP BY [Date Key]
)

SELECT C.OrderYear, C.NumKey, C.[Ordered Quantity Y], T.[Ordered Quantity M]
FROM C2 as C
INNER JOIN C3 AS T
ON C.OrderYear = YEAR(T.[Date Key])
for json path, root('CustomerOrders'), include_null_values;
```

Figure 6G: Formatted JSON Output for Proposition 6

```
},
{
  "OrderYear":2013,
  "NumKey":219,
  "Ordered Quantity Y":2339,
  "Ordered Quantity M":8
},
{
  "OrderYear":2013,
  "NumKey":219,
  "Ordered Quantity Y":2339,
  "Ordered Quantity M":8
},
{
  "OrderYear":2013,
  "NumKey":219,
  "Ordered Quantity Y":2339,
  "Ordered Quantity M":8
},....
```

Proposition 7 (Improved Simple)

Proposition 7: Create a Query that takes in SalesQuota that is Greater Than 200000 and Less Than 300000.

Model Diagrams:

Figure 7A: Key View Model for Proposition 7

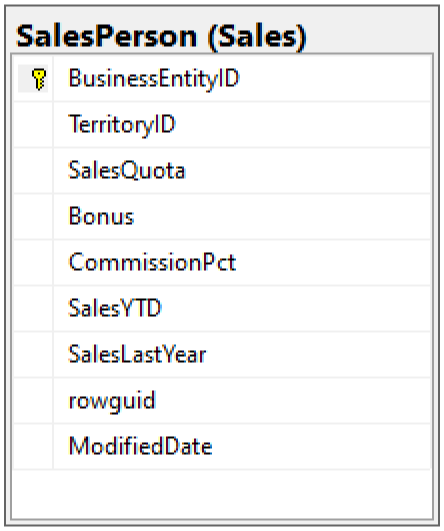
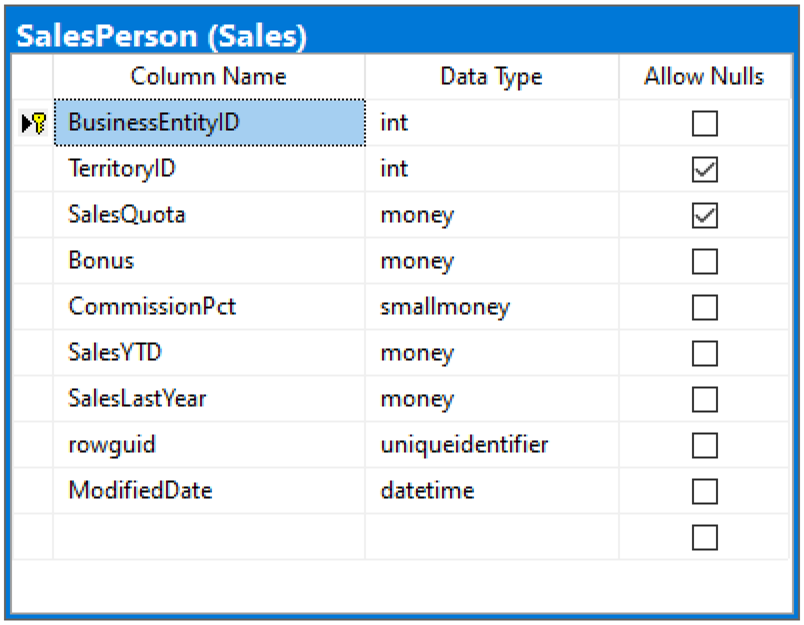


Figure 7B: Standard View Model for Proposition 7



SalesPerson (Sales)

Column Name	Data Type	Allow Nulls
BusinessEntityID	int	☐
TerritoryID	int	✓
SalesQuota	money	✓
Bonus	money	☐
CommissionPct	smallmoney	☐
SalesYTD	money	☐
SalesLastYear	money	☐
rowguid	uniqueidentifier	☐
ModifiedDate	datetime	☐
		☐

Explanation:

summary explanation that will help the developer with the proposition.

Figure 7C: Tables for SQL query components

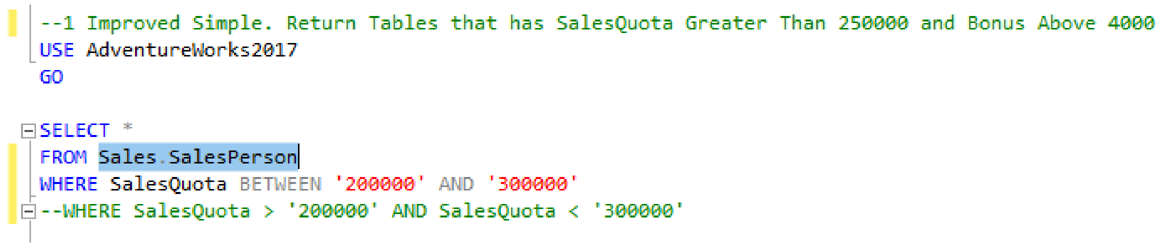
Select clause

Table name:	Column name:
Sales.SalesPerson	*

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

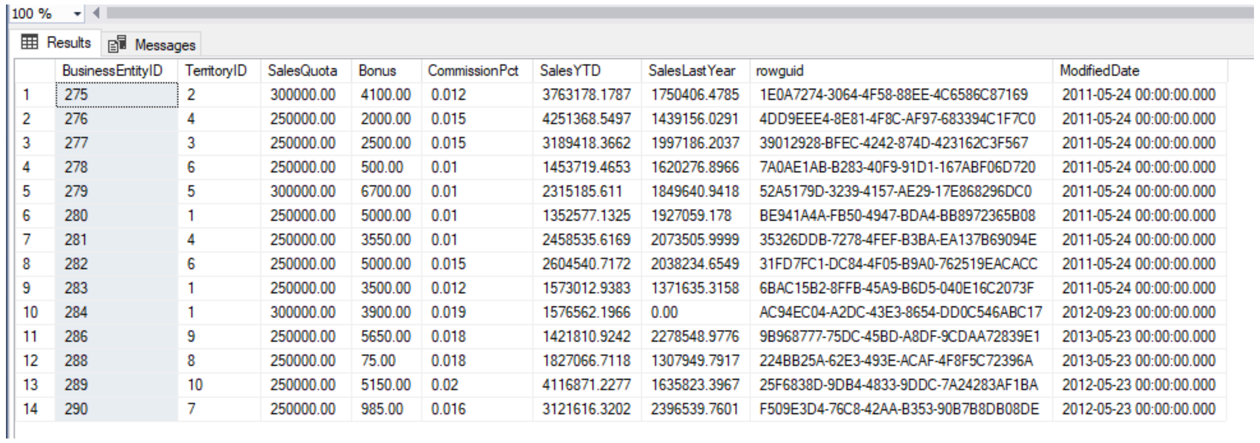
Figure 7D: Formatted SQL Query for Proposition 7



```
--1 Improved Simple. Return Tables that has SalesQuota Greater Than 250000 and Bonus Above 4000
USE AdventureWorks2017
GO

SELECT *
FROM Sales.SalesPerson
WHERE SalesQuota BETWEEN '200000' AND '300000'
--WHERE SalesQuota > '200000' AND SalesQuota < '300000'
```

Figure 7E: Query Output for Proposition 7

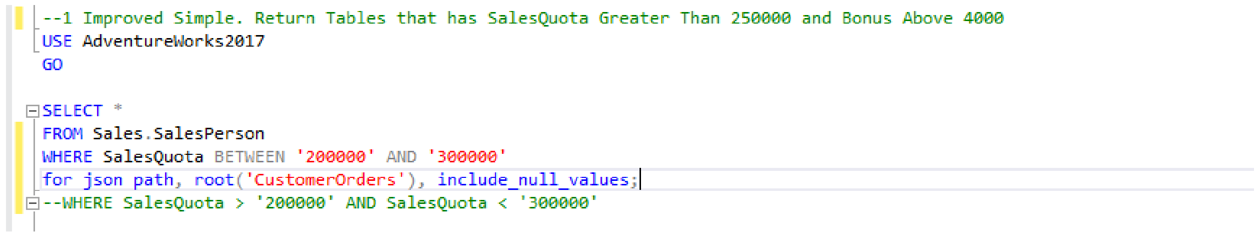


	BusinessEntityID	TerritoryID	SalesQuota	Bonus	CommissionPct	SalesYTD	SalesLastYear	rowguid	ModifiedDate
1	275	2	300000.00	4100.00	0.012	3763178.1787	1750406.4785	1E0A7274-3064-4F58-88EE-4C6586C87169	2011-05-24 00:00:00.000
2	276	4	250000.00	2000.00	0.015	4251368.5497	1439156.0291	4DD9EEE4-8E81-4F8C-AF97-683394C1F7C0	2011-05-24 00:00:00.000
3	277	3	250000.00	2500.00	0.015	3189418.3662	1997186.2037	39012928-BFEC-4242-874D-423162C3F567	2011-05-24 00:00:00.000
4	278	6	250000.00	500.00	0.01	1453719.4653	1620276.8966	7A0AE1AB-B283-40F9-91D1-167ABF06D720	2011-05-24 00:00:00.000
5	279	5	300000.00	6700.00	0.01	2315185.611	1849640.9418	52A5179D-3239-4157-AE29-17E868296DC0	2011-05-24 00:00:00.000
6	280	1	250000.00	5000.00	0.01	1352577.1325	1927059.178	BE941A4A-FB50-4947-BDA4-BB8972365B08	2011-05-24 00:00:00.000
7	281	4	250000.00	3550.00	0.01	2458535.6169	2073505.9999	35326DDB-7278-4FEF-B3BA-EA137B69094E	2011-05-24 00:00:00.000
8	282	6	250000.00	5000.00	0.015	2604540.7172	2038234.6549	31FD7FC1-DC84-4F05-B9A0-762519EACACC	2011-05-24 00:00:00.000
9	283	1	250000.00	3500.00	0.012	1573012.9383	1371635.3158	6BAC15B2-8FFB-45A9-B6D5-040E16C2073F	2011-05-24 00:00:00.000
10	284	1	300000.00	3900.00	0.019	1576562.1966	0.00	AC94EC04-A2DC-43E3-8654-DD0C546ABC17	2012-09-23 00:00:00.000
11	286	9	250000.00	5650.00	0.018	1421810.9242	2278548.9776	9B968777-75DC-45BD-A8DF-9CDAA72839E1	2013-05-23 00:00:00.000
12	288	8	250000.00	75.00	0.018	1827066.7118	1307949.7917	224BB25A-62E3-493E-ACAF-4F8F5C72396A	2013-05-23 00:00:00.000
13	289	10	250000.00	5150.00	0.02	4116871.2277	1635823.3967	25F6838D-9DB4-4833-9DDC-7A24283AF1BA	2012-05-23 00:00:00.000
14	290	7	250000.00	985.00	0.016	3121616.3202	2396539.7601	F509E3D4-76C8-42AA-B353-90B7B8DB08DE	2012-05-23 00:00:00.000

JSON:

Sample JSON Output with total number of rows returned (14)

Figure 7F: Formatted SQL Query with JSON for Proposition 7



```
--1 Improved Simple. Return Tables that has SalesQuota Greater Than 250000 and Bonus Above 4000
USE AdventureWorks2017
GO

SELECT *
FROM Sales.SalesPerson
WHERE SalesQuota BETWEEN '200000' AND '300000'
for json path, root('CustomerOrders'), include_null_values;
--WHERE SalesQuota > '200000' AND SalesQuota < '300000'
```

Figure 7G: Formatted JSON Output for Proposition 7

```
"BusinessEntityID":277,
"TerritoryID":3,
"SalesQuota":250000.0000,
"Bonus":2500.0000,
  "CommissionPct":0.0150,
  "SalesYTD":3189418.3662,
  "SalesLastYear":1997186.2037,
  "rowguid":"39012928-BFEC-4242-874D-423162C3F567",
  "ModifiedDate":"2011-05-24T00:00:00"
},
{
  "BusinessEntityID":278,
  "TerritoryID":6,
  "SalesQuota":250000.0000,
  "Bonus":500.0000,
  "CommissionPct":0.0100,
  "SalesYTD":1453719.4653,
  "SalesLastYear":1620276.8966,
  "rowguid":"7A0AE1AB-B283-40F9-91D1-167ABF06D720",
  "ModifiedDate":"2011-05-24T00:00:00"
},
{
  "BusinessEntityID":279,
  "TerritoryID":5,
  "SalesQuota":300000.0000,
  "Bonus":6700.0000,
  "CommissionPct":0.0100,
  "SalesYTD":2315185.6110,
  "SalesLastYear":1849640.9418,
  "rowguid":"52A5179D-3239-4157-AE29-17E868296DC0",
  "ModifiedDate":"2011-05-24T00:00:00"
},......
```

Proposition 8 (Improved Medium)

Proposition 8: Return orders on Argentina along with Total Quantity. Sorted by OrderId

Model Diagrams:

Figure 8A: Key View Model for Proposition 8

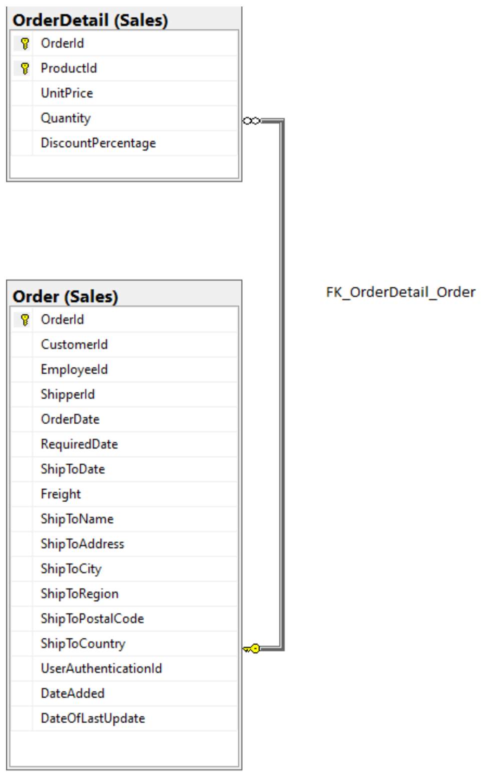
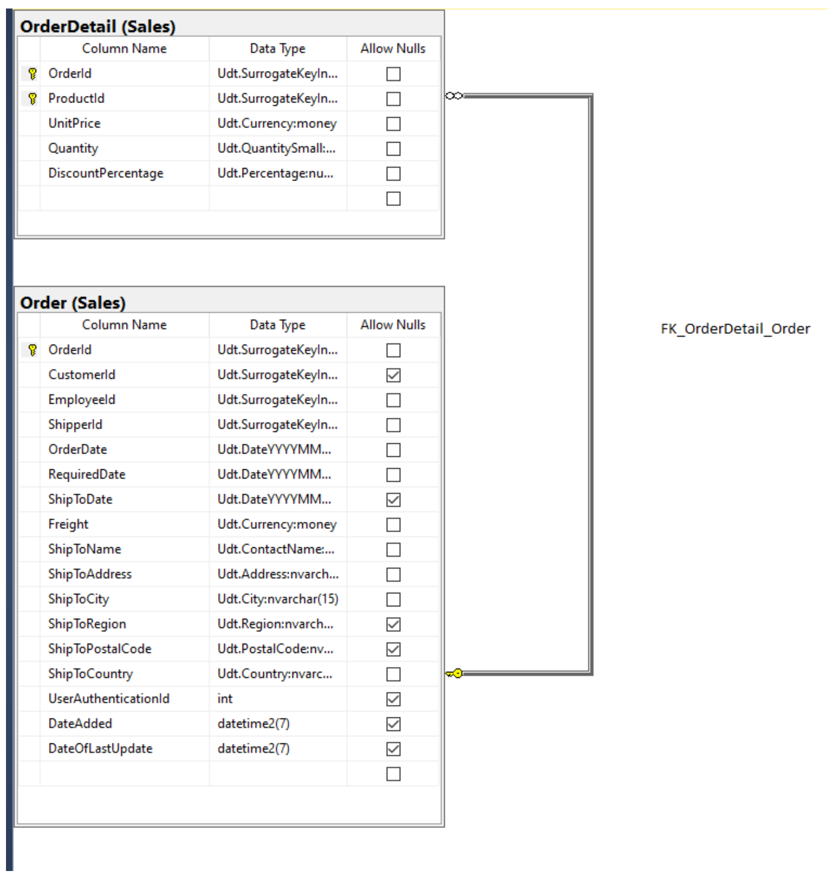


Figure 8B: Standard View Model for Proposition 8



OrderDetail (Sales)

Column Name	Data Type	Allow Nulls
OrderId	Udt.SurrogateKeyIn...	☐
ProductId	Udt.SurrogateKeyIn...	☐
UnitPrice	Udt.Currency:money	☐
Quantity	Udt.QuantitySmall:...	☐
DiscountPercentage	Udt.Percentage:nu...	☐
		☐

Order (Sales)

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

Explanation:
Select OrderId and Rather than return Total Orders, I return Total Distinct Orders. I Inner Join with Sales.Order on OrderId. Restricted the query to only output tables that have Argentina as the ShipToCountry.

Figure 8C: Tables for SQL query components

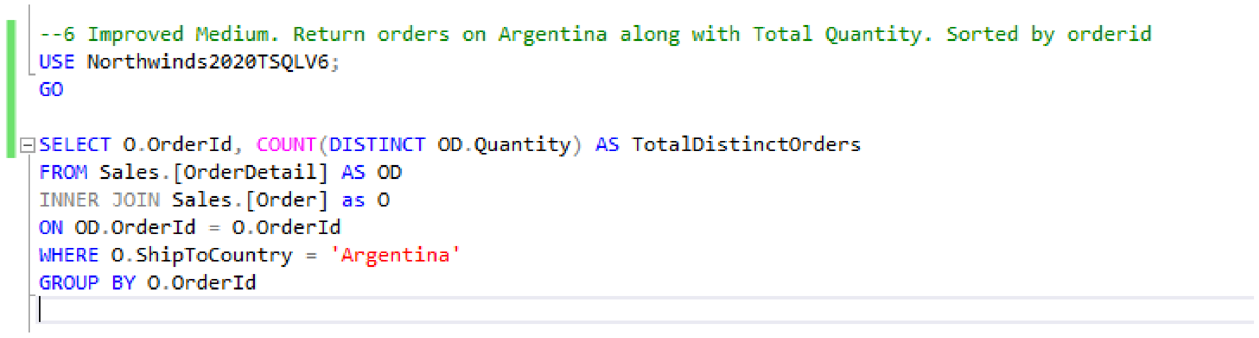
Select clause

Table name:	Column name:
Sales.[OrderDetails]	TotalDistinctOrders
Sales.[Order]	O.ShipToCountry O.OrderId

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

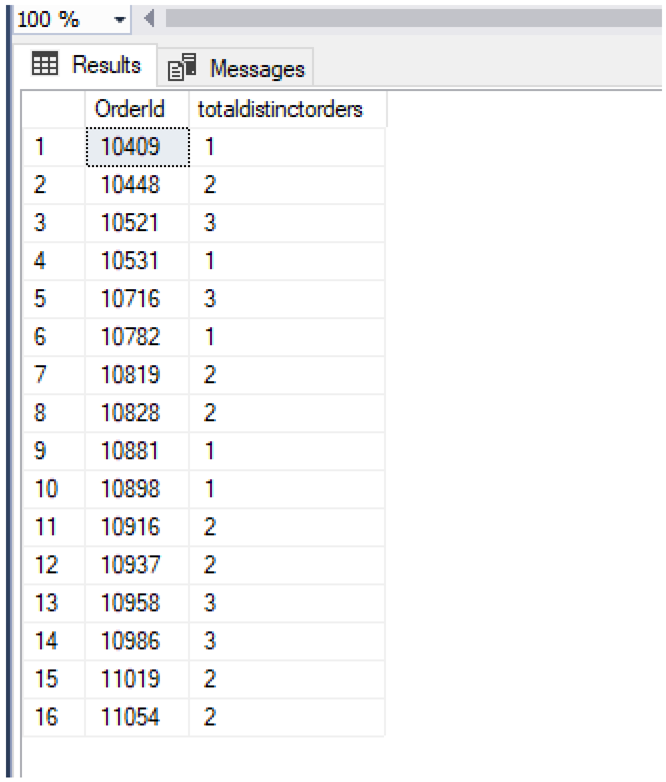
Figure 8D: Formatted SQL Query for Proposition 8



```
--6 Improved Medium. Return orders on Argentina along with Total Quantity. Sorted by orderid
USE Northwinds2020TSQLV6;
GO

SELECT O.OrderId, COUNT(DISTINCT OD.Quantity) AS TotalDistinctOrders
FROM Sales.[OrderDetail] AS OD
INNER JOIN Sales.[Order] as O
ON OD.OrderId = O.OrderId
WHERE O.ShipToCountry = 'Argentina'
GROUP BY O.OrderId
```

Figure 8E: Query Output for Proposition 8

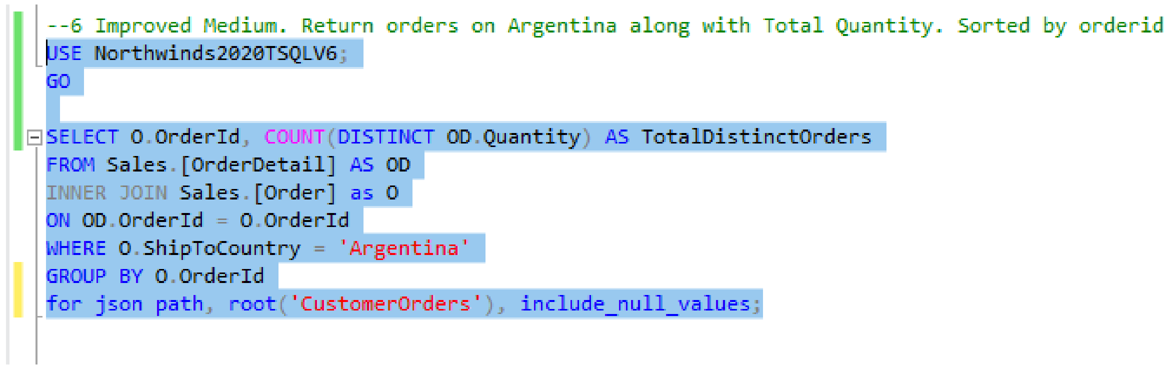


```
    OrderId  totaldistinctorders
1   10409    1
2   10448    2
3   10521    3
4   10531    1
5   10716    3
6   10782    1
7   10819    2
8   10828    2
9   10881    1
10  10898    1
11  10916    2
12  10937    2
13  10958    3
14  10986    3
15  11019    2
16  11054    2
```

JSON:

Sample JSON Output with total number of rows returned (16)

Figure 8F: Formatted SQL Query with JSON for Proposition 8



```
--6 Improved Medium. Return orders on Argentina along with Total Quantity. Sorted by orderid
USE Northwinds2020TSQLV6;
GO

SELECT O.OrderId, COUNT(DISTINCT OD.Quantity) AS TotalDistinctOrders
FROM Sales.[OrderDetail] AS OD
INNER JOIN Sales.[Order] as O
ON OD.OrderId = O.OrderId
WHERE O.ShipToCountry = 'Argentina'
GROUP BY O.OrderId
for json path, root('CustomerOrders'), include_null_values;
```

Figure 8G: Formatted JSON Output for Proposition 8

```
{
  "CustomerOrders":[
    {
      "OrderId":10409,
      "TotalDistinctOrders":1
    },
    {
      "OrderId":10448,
      "TotalDistinctOrders":2
    },
    {
      "OrderId":10521,
      "TotalDistinctOrders":3
    },
    {
      "OrderId":10531,
      "TotalDistinctOrders":1
    },
    {
      "OrderId":10716,
      "TotalDistinctOrders":3
    },
{
  "OrderId":10782,
  "TotalDistinctOrders":1
},
{
  "OrderId":10819,
  "TotalDistinctOrders":2
},
{
  "OrderId":10828,
  "TotalDistinctOrders":2
},
{
  "OrderId":10881,
  "TotalDistinctOrders":1
},
{
  "OrderId":10898,
  "TotalDistinctOrders":1
},
{
  "OrderId":10916,
  "TotalDistinctOrders":2
},
{
  "OrderId":10937,
  "TotalDistinctOrders":2
},
{
  "OrderId":10958,
  "TotalDistinctOrders":3
},........
```

Proposition 9 (Improved Complex)

Proposition 9: Create Function where you input first and last name and it returns the title. birth date, address, postal code, country, phone number, orderid and discount percentage. Sorted by OrderID

Model Diagrams:

Figure 9A: Key View Model for Proposition 9

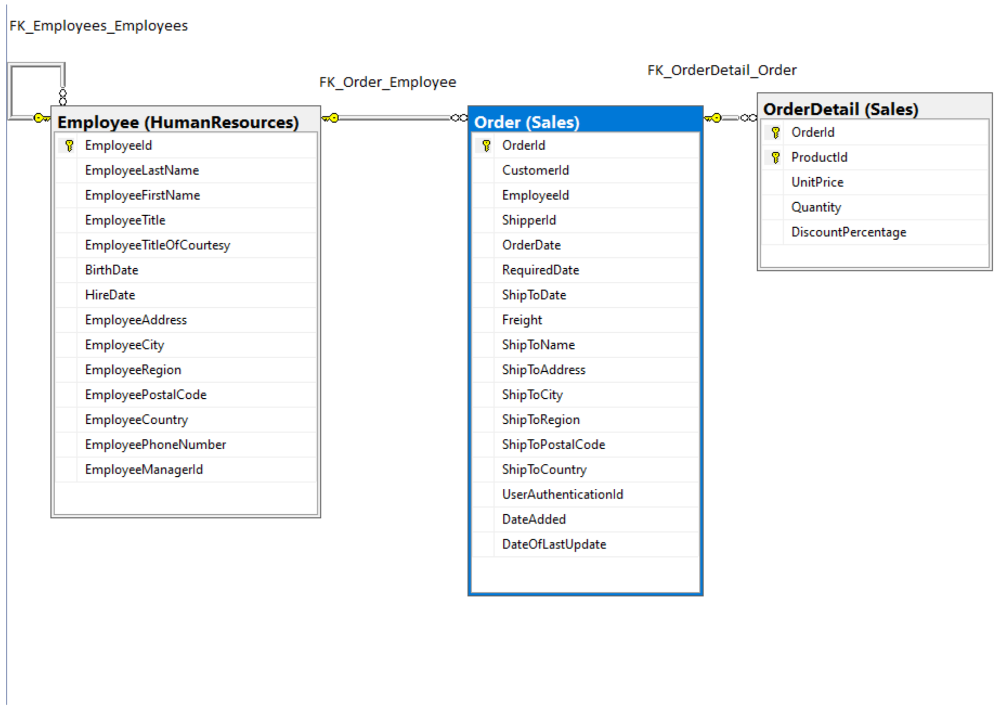
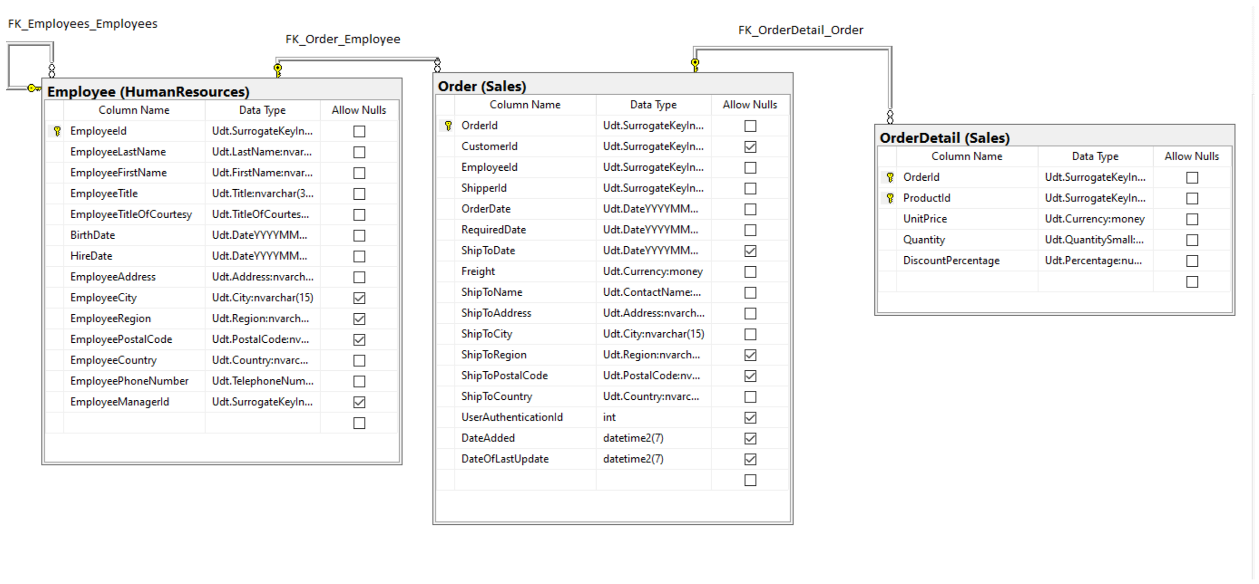


Figure 9B: Standard View Model for Proposition 9



Employee (HumanResources)

	Column Name	Data Type	Allow Nulls
🔑	EmployeeId	Udt.SurrogateKeyIn...	☐
	EmployeeLastName	Udt.LastName:nvar...	☐
	EmployeeFirstName	Udt.FirstName:nvar...	☐
	EmployeeTitle	Udt.Title:nvarchar(3...	☐
	EmployeeTitleOfCourtesy	Udt.TitleOfCourtes...	☐
	BirthDate	Udt.DateYYYYMM...	☐
	HireDate	Udt.DateYYYYMM...	☐
	EmployeeAddress	Udt.Address:nvarch...	☐
	EmployeeCity	Udt.City:nvarchar(15)	☑
	EmployeeRegion	Udt.Region:nvarch...	☑
	EmployeePostalCode	Udt.PostalCode:nv...	☑
	EmployeeCountry	Udt.Country:nvarc...	☐
	EmployeePhoneNumber	Udt.TelephoneNum...	☐
	EmployeeManagerId	Udt.SurrogateKeyIn...	☑
			☐

Order (Sales)

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

OrderDetail (Sales)

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

Explanation:

Selected E.EmployeeId, E.EmployeeTitle, E.BirthDate, E.HireDate, E.EmployeeAddress, E.EmployeePostalCode, E.EmployeeCountry, E.EmployeePhoneNumber on HumanResources.Employees. Then I Inner Join Sales.Order for OrderId and Inner Join on OrderDetail for DiscountPercentages. Had the Function take in First and Last Name so the User can Input and Find out the Information that the Function outputs.

Figure 9C: Tables for SQL query components

Select clause

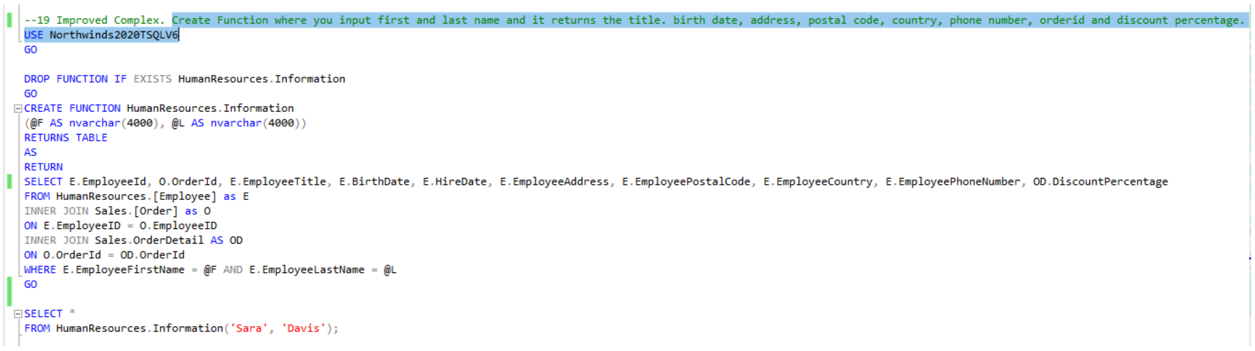
Table name:	Column name:
HumanResources.[Employee]	E.EmployeeId E.EmployeeTitle E.BirthDate E.HireDate E.EmployeeAddress, E.EmployeePostalCode E.EmployeeCountry, E.EmployeePhoneNumber
Sales.[Order]	OD.DiscountPercentage

Sales.OrderDetail	O.OrderId

Query:

All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

Figure 9D: Formatted SQL Query for Proposition 9

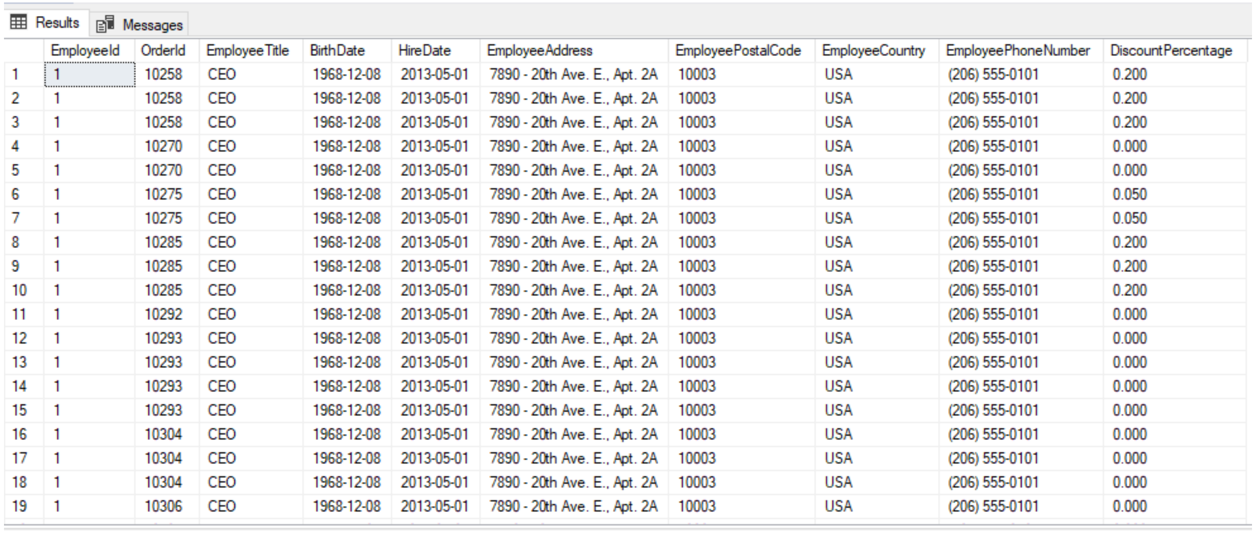


```sql
--19 Improved Complex. Create Function where you input first and last name and it returns the title. birth date, address, postal code, country, phone number, orderid and discount percentage.
USE Northwinds2020TSQLV6
GO

DROP FUNCTION IF EXISTS HumanResources.Information
GO
CREATE FUNCTION HumanResources.Information
(@F AS nvarchar(4000), @L AS nvarchar(4000))
RETURNS TABLE
AS
RETURN
SELECT E.EmployeeId, O.OrderId, E.EmployeeTitle, E.BirthDate, E.HireDate, E.EmployeeAddress, E.EmployeePostalCode, E.EmployeeCountry, E.EmployeePhoneNumber, OD.DiscountPercentage
FROM HumanResources.[Employee] as E
INNER JOIN Sales.[Order] as O
ON E.EmployeeID = O.EmployeeID
INNER JOIN Sales.OrderDetail AS OD
ON O.OrderId = OD.OrderId
WHERE E.EmployeeFirstName = @F AND E.EmployeeLastName = @L
GO

SELECT *
FROM HumanResources.Information('Sara', 'Davis');
```

Figure 9E: Query Output for Proposition 9



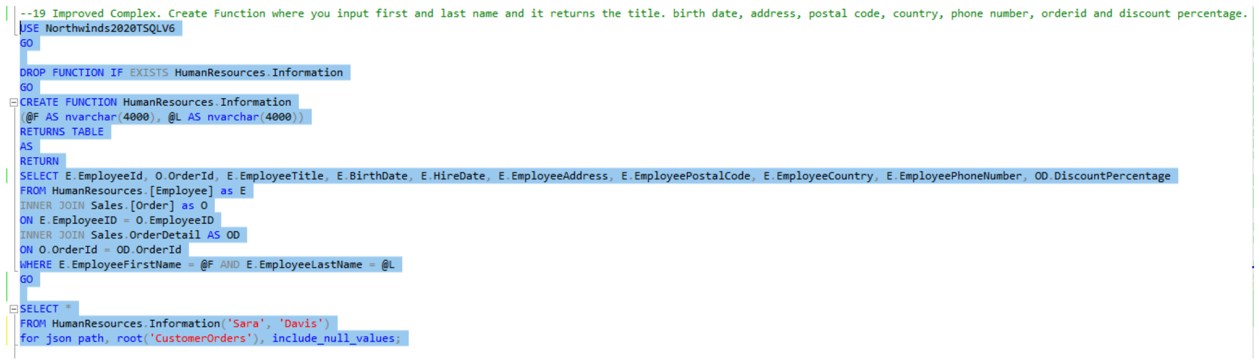
Results Messages

	EmployeeId	OrderId	EmployeeTitle	BirthDate	HireDate	EmployeeAddress	EmployeePostalCode	EmployeeCountry	EmployeePhoneNumber	DiscountPercentage
1	1	10258	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.200
2	1	10258	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.200
3	1	10258	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.200
4	1	10270	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000
5	1	10270	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000
6	1	10275	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.050
7	1	10275	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.050
8	1	10285	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.200
9	1	10285	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.200
10	1	10285	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.200
11	1	10292	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000
12	1	10293	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000
13	1	10293	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000
14	1	10293	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000
15	1	10293	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000
16	1	10304	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000
17	1	10304	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000
18	1	10304	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000
19	1	10306	CEO	1968-12-08	2013-05-01	7890 - 20th Ave. E., Apt. 2A	10003	USA	(206) 555-0101	0.000

JSON:

Sample JSON Output with total number of rows returned (345)

Figure 9F: Formatted SQL Query with JSON for Proposition 9



```sql
--19 Improved Complex. Create Function where you input first and last name and it returns the title. birth date, address, postal code, country, phone number, orderid and discount percentage.
USE Northwinds2020TSQLV6
GO

DROP FUNCTION IF EXISTS HumanResources.Information
GO
CREATE FUNCTION HumanResources.Information
(@F AS nvarchar(4000), @L AS nvarchar(4000))
RETURNS TABLE
AS
RETURN
SELECT E.EmployeeId, O.OrderId, E.EmployeeTitle, E.BirthDate, E.HireDate, E.EmployeeAddress, E.EmployeePostalCode, E.EmployeeCountry, E.EmployeePhoneNumber, OD.DiscountPercentage
FROM HumanResources.[Employee] as E
INNER JOIN Sales.[Order] as O
ON E.EmployeeID = O.EmployeeID
INNER JOIN Sales.OrderDetail AS OD
ON O.OrderId = OD.OrderId
WHERE E.EmployeeFirstName = @F AND E.EmployeeLastName = @L
GO

SELECT *
FROM HumanResources.Information('Sara', 'Davis')
for json path, root('CustomerOrders'), include_null_values;
```

Figure 9G: Formatted JSON Output for Proposition 9

```json
{
 "CustomerOrders":[
   {
     "EmployeeId":1,
     "OrderId":10258,
     "EmployeeTitle":"CEO",
     "BirthDate":"1968-12-08",
     "HireDate":"2013-05-01",
     "EmployeeAddress":"7890 - 20th Ave. E., Apt. 2A",
     "EmployeePostalCode":"10003",
     "EmployeeCountry":"USA",
     "EmployeePhoneNumber":"(206) 555-0101",
     "DiscountPercentage":0.200
   },
   {
     "EmployeeId":1,
     "OrderId":10258,
     "EmployeeTitle":"CEO",
     "BirthDate":"1968-12-08",
     "HireDate":"2013-05-01",
     "EmployeeAddress":"7890 - 20th Ave. E., Apt. 2A",
     "EmployeePostalCode":"10003",
     "EmployeeCountry":"USA",
     "EmployeePhoneNumber":"(206) 555-0101",
     "DiscountPercentage":0.200
},
```