Haibo Liu

CS331 DATABASE SYSTEM

PROFESSOR HELLER

Group4\_project 1

Simple queries 1

Proposition: This query to find the customers who didi not place order

**WITHOUT JSON:**

use Northwinds2020TSQLV6;

SELECT C.CustomerId

,C.CustomerCompanyName

FROM Sales.[Customer] AS C

LEFT OUTER JOIN Sales.[Order] AS O ON C.CustomerId = O.CustomerId

WHERE O.orderid IS NULL;

--FOR JSON PATH, ROOT (“SIMPLE QUERY 1”), INCLUDE\_NULL\_VALUES

**WITH JSON:**

use Northwinds2020TSQLV6;

SELECT C.CustomerId

,C.CustomerCompanyName

FROM Sales.[Customer] AS C

LEFT OUTER JOIN Sales.[Order] AS O ON C.CustomerId = O.CustomerId

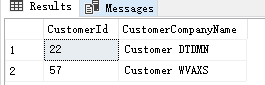
WHERE O.orderid IS NULL;

FOR JSON PATH, ROOT ('SIMPLE QUERY 1'), INCLUDE\_NULL\_VALUES

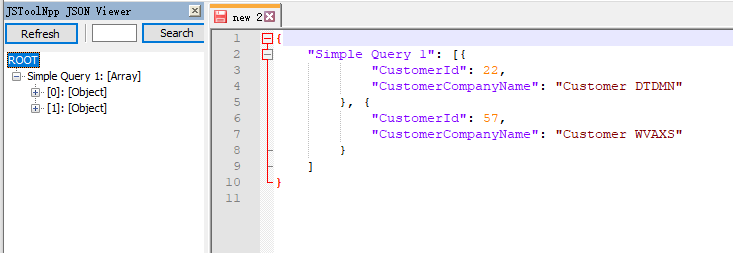
Table used :

|  |  |
| --- | --- |
| Table name | Column name |
| customer | customerId,customerCompanyName |
| Order | orderId |

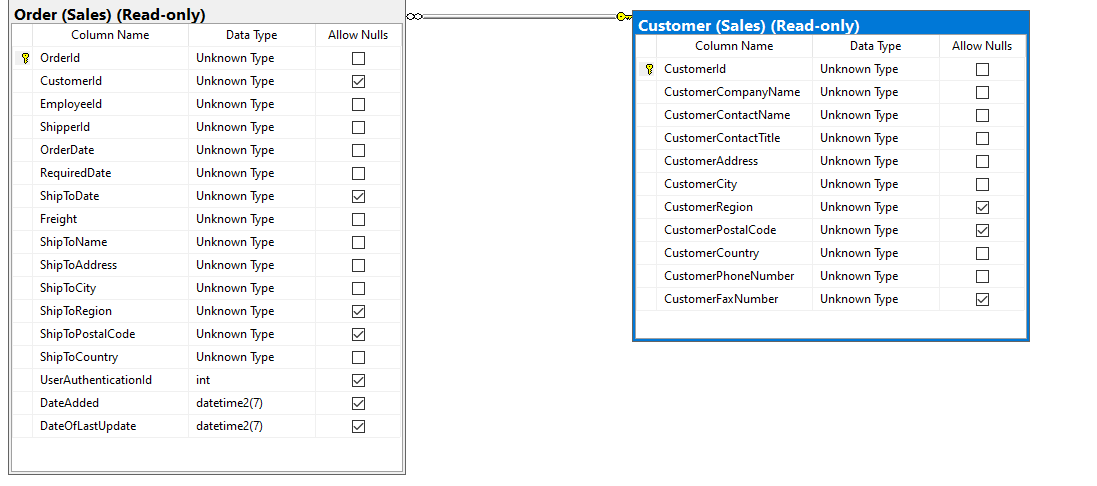
Sample output without JSON:



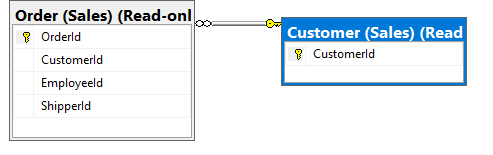
Sample output with JSON:



Standard View:



Key View:



Simple query 2:

Proposition: return the order between 20200101 and 20201009, order by orderdate

**WITHOUT JSON:**

USE Northwinds2020TSQLV6;

SELECT DATEADD(day, Nums.N - 1, CAST('20200101' AS DATE)) AS orderdate

,O.OrderId

,O.CustomerId

,O.EmployeeId

FROM dbo.Nums

LEFT OUTER JOIN Sales.[Order] AS O ON DATEADD(DAY, Nums.N - 1, CAST('20200101' AS DATE)) = O.orderdate

WHERE Nums.N <= DATEDIFF(DAY, '20200101', '20201009') + 1

ORDER BY orderdate;

--FOR JSON PATH, ROOT ('SIMPLE QUERY 2'), INCLUDE\_NULL\_VALUES

**WITH JSON:**

USE Northwinds2020TSQLV6;

SELECT DATEADD(day, Nums.N - 1, CAST('20200101' AS DATE)) AS orderdate

,O.OrderId

,O.CustomerId

,O.EmployeeId

FROM dbo.Nums

LEFT OUTER JOIN Sales.[Order] AS O ON DATEADD(DAY, Nums.N - 1, CAST('20200101' AS DATE)) = O.orderdate

WHERE Nums.N <= DATEDIFF(DAY, '20200101', '20201009') + 1

ORDER BY orderdate;

FOR JSON PATH, ROOT ('SIMPLE QUERY 2'), INCLUDE\_NULL\_VALUES

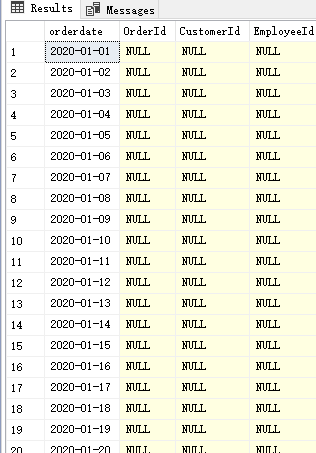
Table used :

|  |  |
| --- | --- |
| Table name | Column name |
| Nums | N |
| Order | orderId,customerId,EmployeeId,orderDate |

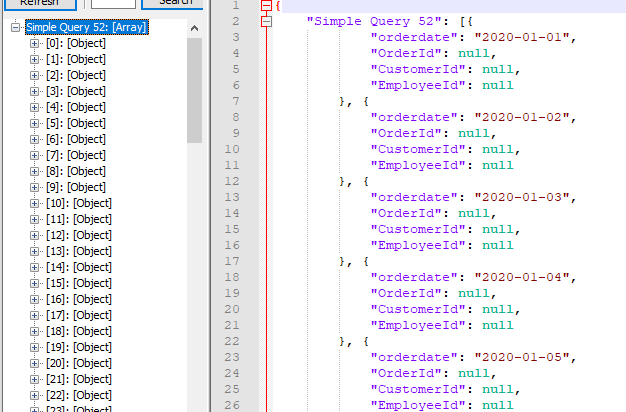
Sort by

|  |  |  |
| --- | --- | --- |
| Table name | Column name | Sort order |
| Order | Orderdate | ASC |

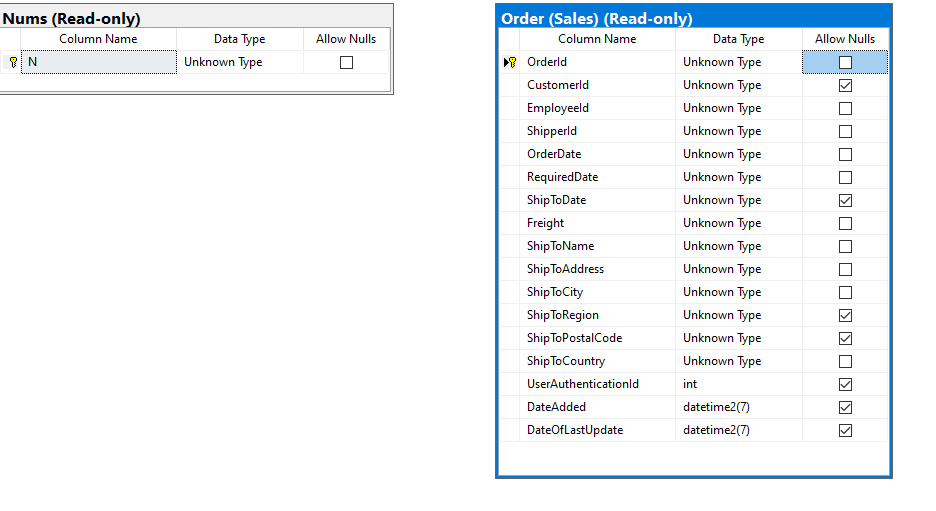
Sample output without JSON:



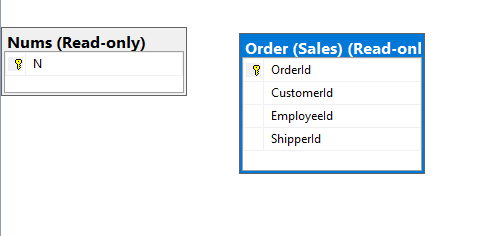
Sample output with JSON:



Standard View:



Key View:



Simple query 3:

Proposition: return mumber of orders of each customer placed

WITHOUT JSON:

SELECT C.CustomerId

,COUNT(\*) AS numorders

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O ON C.CustomerId = O.CustomerId

GROUP BY C.CustomerId;

--FOR JSON PATH, ROOT ('SIMPLE QUERY 3'), INCLUDE\_NULL\_VALUES

WITH JSON:

SELECT C.CustomerId

,COUNT(\*) AS numorders

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O ON C.CustomerId = O.CustomerId

GROUP BY C.CustomerId;

FOR JSON PATH, ROOT ('SIMPLE QUERY 3'), INCLUDE\_NULL\_VALUES

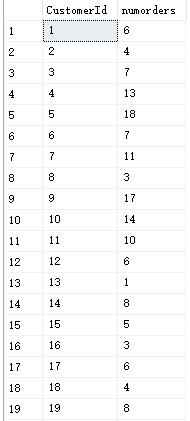
Table used :

|  |  |
| --- | --- |
| Table name | Column name |
| customer | customerId,numorders |
| Order | orderId |

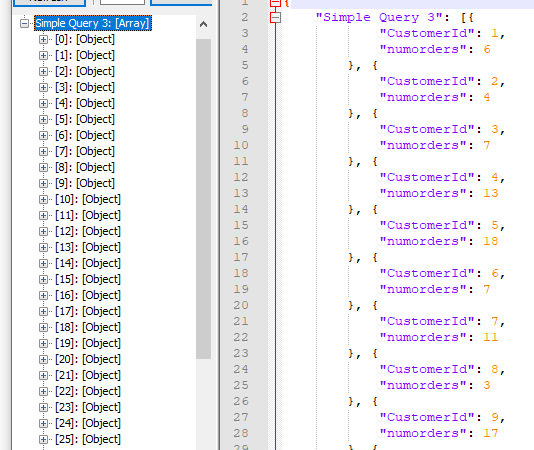
Sort by

|  |  |  |
| --- | --- | --- |
| Table name | Column name | Sort order |
| Customer | customerId | ASC |

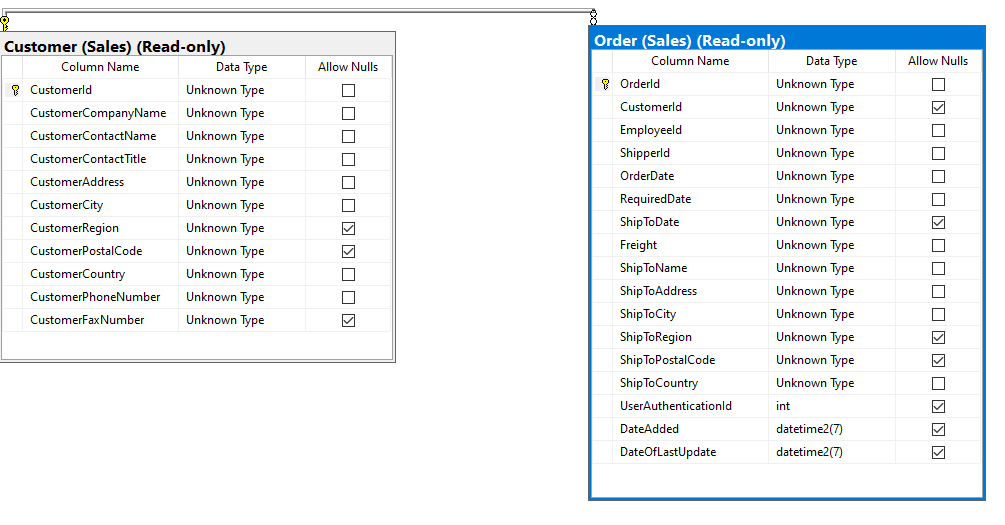
Sample output without JSON:



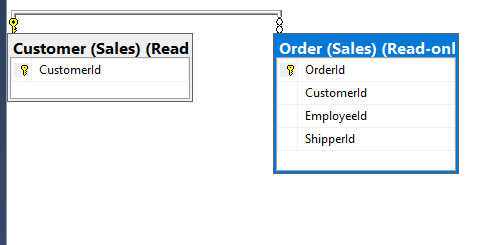
Sample output with JSON:



Standard View:



Key View:



Simple Query 4:

Proposition:return order between 20160212 and 20160312,without null order

WITHOUT JSON:

use Northwinds2020TSQLV6;

SELECT C.CustomerId,

C.CustomerCompanyName, ​

O.orderid, ​

O.orderdate​

FROM [Sales].[Customer] AS C​

LEFT OUTER JOIN ​

[Sales].[Order] AS O​

ON O.CustomerId = C.CustomerId​

AND O.orderdate between '20160212' and '20160312'​

where O.OrderId is not null

--FOR JSON PATH, ROOT ('SIMPLE QUERY 4'), INCLUDE\_NULL\_VALUES

WITH JSON:

use Northwinds2020TSQLV6;

SELECT C.CustomerId,

C.CustomerCompanyName, ​

O.orderid, ​

O.orderdate​

FROM [Sales].[Customer] AS C​

LEFT OUTER JOIN ​

[Sales].[Order] AS O​

ON O.CustomerId = C.CustomerId​

AND O.orderdate between '20160212' and '20160312'​

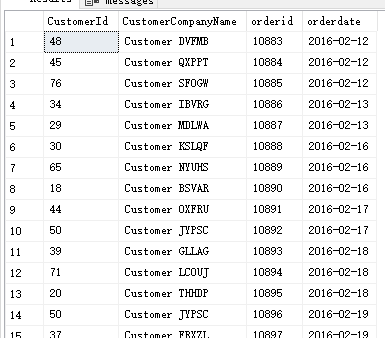
where O.OrderId is not null

FOR JSON PATH, ROOT ('SIMPLE QUERY 4'), INCLUDE\_NULL\_VALUES

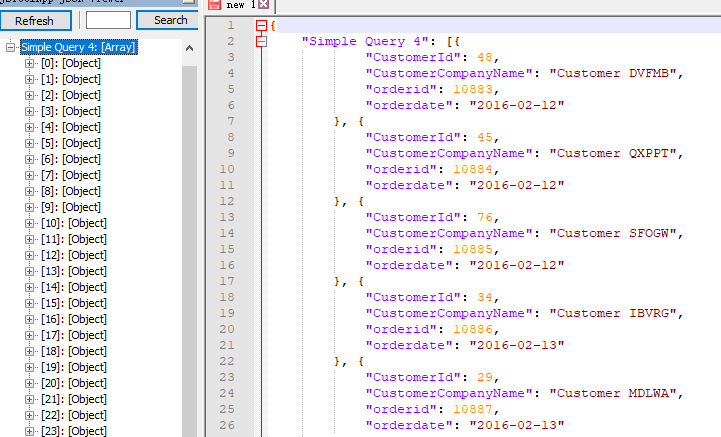
Table used :

|  |  |
| --- | --- |
| Table name | Column name |
| customer | customerId,customerCompanyName |
| Order | orderId,orderDate |

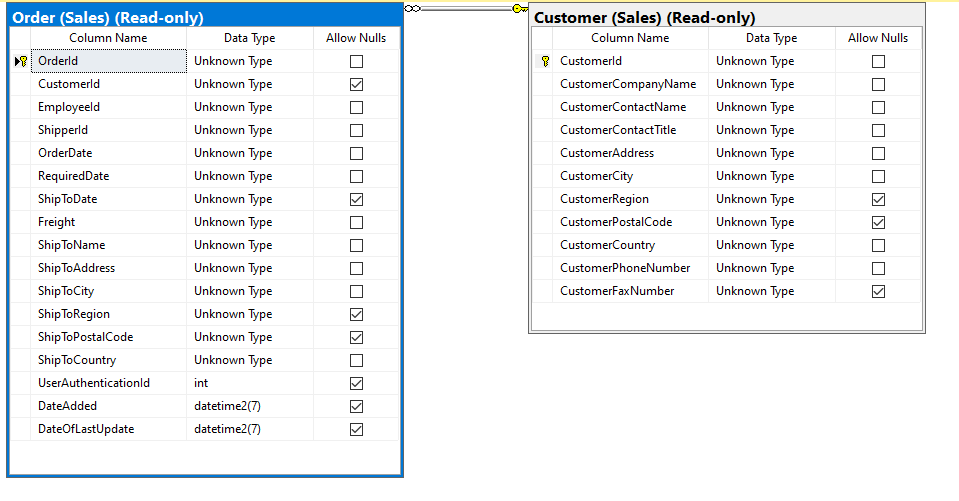
Sample output without JSON:



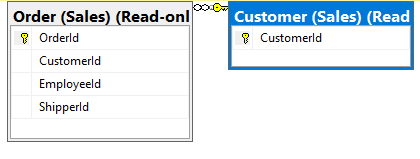
Sample output with JSON:



Standard View:



Key View:



Simple Query 5:

Proposition: return the spanish version of the date from DimDate.

WITHOUT JSON:

USE AdventureWorksDW2017;

SELECT fcr.DateKey

,CONCAT (

dd.SpanishMonthName

,' '

,dd.SpanishDayNameOfWeek

,' '

,dd.CalendarYear

) AS spanishdate

FROM dbo.DimDate AS dd

INNER JOIN dbo.FactCurrencyRate AS fcr ON dd.DateKey = fcr.DateKey

GROUP BY fcr.DateKey

,dd.SpanishMonthName

,dd.SpanishDayNameOfWeek

,dd.CalendarYear

ORDER BY fcr.DateKey;

--FOR JSON PATH, ROOT ('SpanishDate'), INCLUDE\_NULL\_VALUES

WITH JSON:

USE AdventureWorksDW2017;

SELECT fcr.DateKey

,CONCAT (

dd.SpanishMonthName

,' '

,dd.SpanishDayNameOfWeek

,' '

,dd.CalendarYear

) AS spanishdate

FROM dbo.DimDate AS dd

INNER JOIN dbo.FactCurrencyRate AS fcr ON dd.DateKey = fcr.DateKey

GROUP BY fcr.DateKey

,dd.SpanishMonthName

,dd.SpanishDayNameOfWeek

,dd.CalendarYear

ORDER BY fcr.DateKey;

FOR JSON PATH, ROOT ('SpanishDate'), INCLUDE\_NULL\_VALUES

Table used :

|  |  |
| --- | --- |
| Table name | Column name |
| Dimadate | Spanishmonthname,spanishdaynameofweek,calendaryear |
| Factcurrencyrate | Datekey |

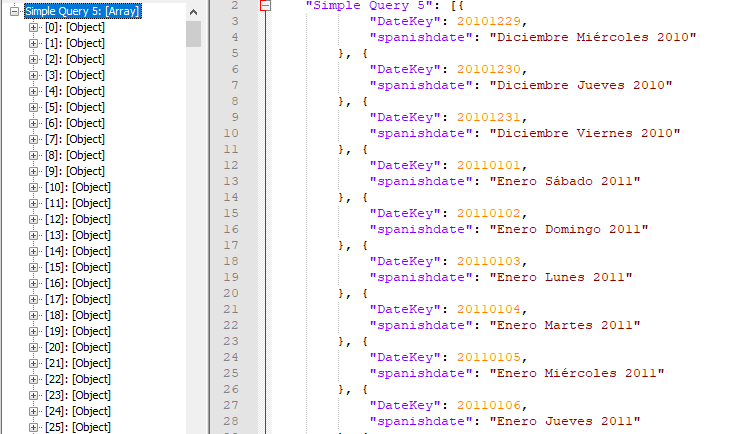
Sort by:

|  |  |  |
| --- | --- | --- |
| Table name | Column name | Sort order |
| Dimdate | Spanishmonthname,spanishdaynameofweek,calendaryear | ASC |
| FACTCURRENCYRATE | DATEKEY | ASC |

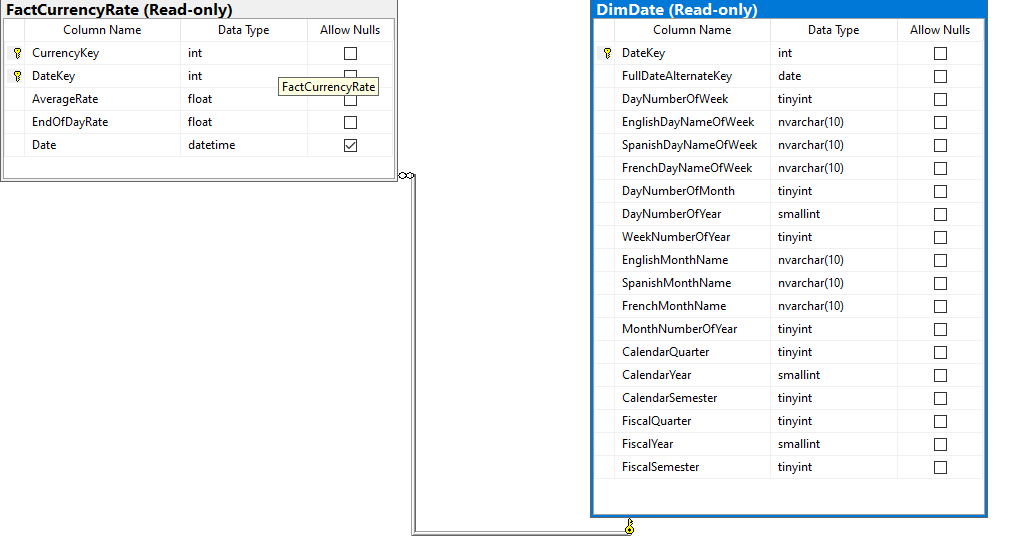
Sample output without JSON:



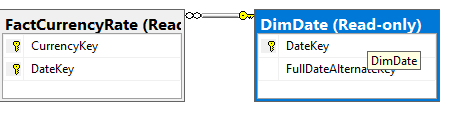
Sample output with JSON:



Standard View:



Key View:



Medium Queries:

Medium query 1:

Proposition: this query generates five copies of each employee row

**WITHOUT JSON:**

use Northwinds2020TSQLV6;

SELECT E.EmployeeId

,E.EmployeeFirstName

,E.EmployeeLastName

,N.N

FROM HumanResources.Employee AS E

CROSS JOIN dbo.Nums AS N

WHERE N.N <= 5

ORDER BY n

,EmployeeId;

--FOR JSON PATH, ROOT ('Medium Query 1'), INCLUDE\_NULL\_VALUES

**WITH JSON:**

use Northwinds2020TSQLV6;

SELECT E.EmployeeId

,E.EmployeeFirstName

,E.EmployeeLastName

,N.N

FROM HumanResources.Employee AS E

CROSS JOIN dbo.Nums AS N

WHERE N.N <= 5

ORDER BY n

,EmployeeId;

FOR JSON PATH, ROOT ('Medium Query 1'), INCLUDE\_NULL\_VALUES

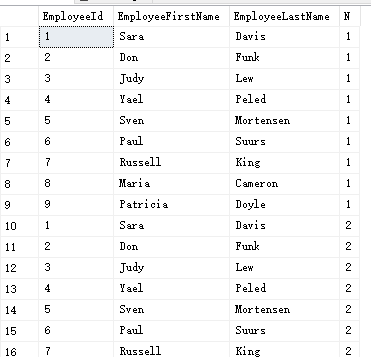
Table used :

|  |  |
| --- | --- |
| Table name | Column name |
| Employee | EmployeeId,EmployeeFirstName,EmployeeLastName |
| Nums | N |

Order BY

|  |  |  |
| --- | --- | --- |
| Table name | Column name | Sort by |
| Nums | N | ASC |
| Employee | EmployeeId | ASC |

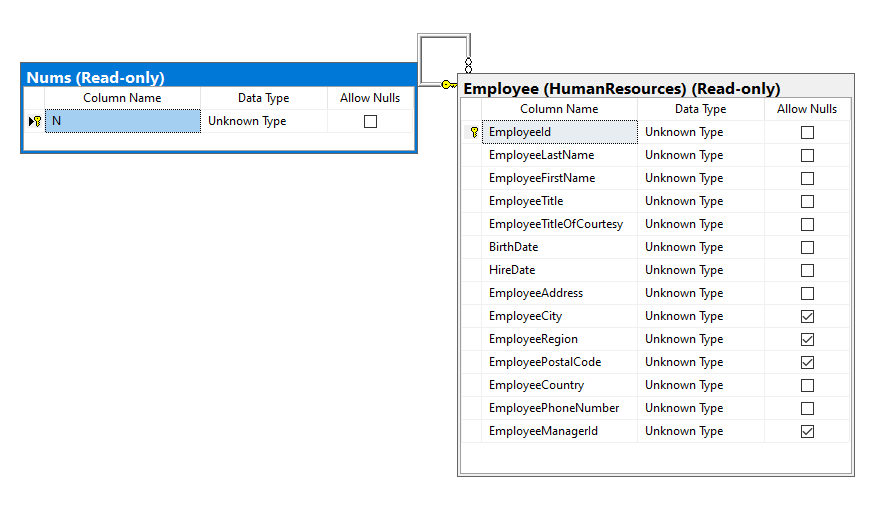
Sample output without JSON:



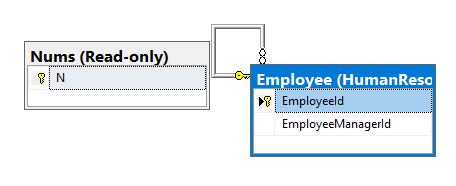
Sample output with JSON:



Standard View:



Key View:



Medium query 2:

Proposition: this query returns a row for each employee and day in the range August 5,2010 through December 29 2011

WITHOUT JSON:

use Northwinds2020TSQLV6;

SELECT E.EmployeeId

,DATEADD(day, D.n - 1, CAST('20100805' AS DATE)) AS dt

FROM HumanResources.Employee AS E

CROSS JOIN dbo.Nums AS D

WHERE D.n <= DATEDIFF(day, '20100805', '20111229') + 1

ORDER BY EmployeeId

,dt;

--FOR JSON PATH, ROOT ('Medium Query 2'), INCLUDE\_NULL\_VALUES

WITH JSON:

use Northwinds2020TSQLV6;

SELECT E.EmployeeId

,DATEADD(day, D.n - 1, CAST('20100805' AS DATE)) AS dt

FROM HumanResources.Employee AS E

CROSS JOIN dbo.Nums AS D

WHERE D.n <= DATEDIFF(day, '20100805', '20111229') + 1

ORDER BY EmployeeId

,dt;

FOR JSON PATH, ROOT ('Medium Query 2'), INCLUDE\_NULL\_VALUES

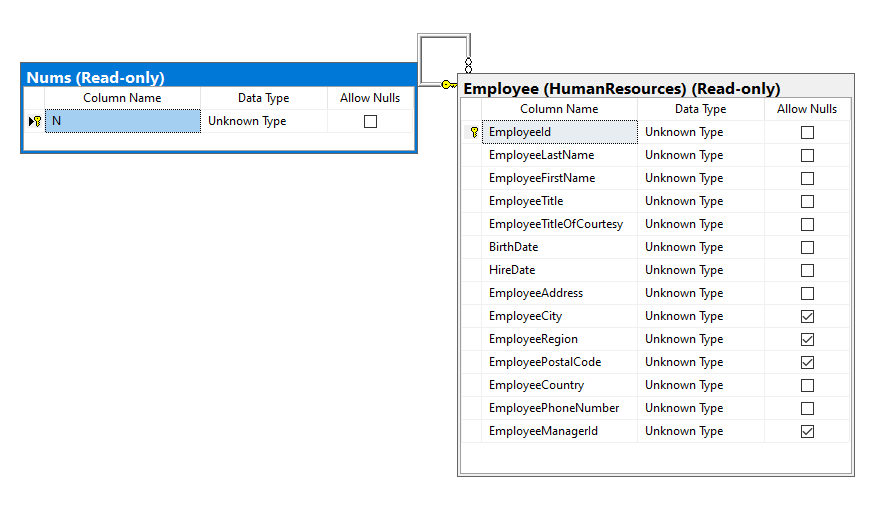
Table used :

|  |  |
| --- | --- |
| Table name | Column name |
| Employee | EmployeeId,EmployeeFirstName,EmployeeLastName |
| Nums | N |

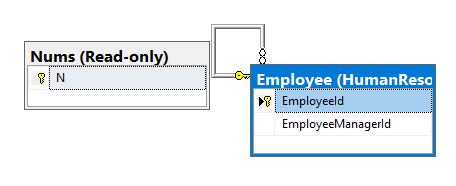
Order BY

|  |  |  |
| --- | --- | --- |
| Table name | Column name | Sort by |
| Nums | N | ASC |
| Employee | EmployeeId | ASC |

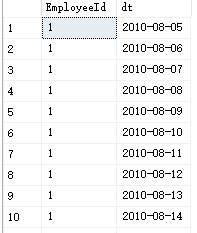
Standard View:



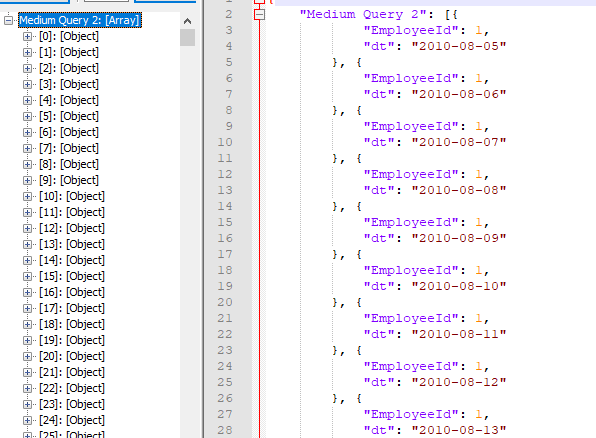
Key View:



Sample output without JSON:



Sample output with JSON:



Medium query 3:

Proposition: return the reviewers and comments for each order, and ratings

WITHOUT JSON:

Use AdventureWorks2017;​

SELECT COUNT(OD.ProductID) [Count], ​

OD.ProductID, PP.[Name], ​

SUM(LineTotal) TotalSale,PRP.Rating, PRP.ReviewerName, PRP.Comments ​

FROM Sales.SalesOrderDetail AS OD ​

JOIN Production.Product AS PP ON PP.ProductID = OD.ProductID ​

JOIN Production.ProductReview AS PRP ON PRP.ProductID = PP.ProductID ​

GROUP BY OD.ProductID, PP.[Name], PRP.Rating, PRP.ReviewerName, PRP.Comments

--FOR JSON PATH, ROOT ('Medium Query 3'), INCLUDE\_NULL\_VALUES

WITH JSON:

Use AdventureWorks2017;​

SELECT COUNT(OD.ProductID) [Count], ​

OD.ProductID, PP.[Name], ​

SUM(LineTotal) TotalSale,PRP.Rating, PRP.ReviewerName, PRP.Comments ​

FROM Sales.SalesOrderDetail AS OD ​

JOIN Production.Product AS PP ON PP.ProductID = OD.ProductID ​

JOIN Production.ProductReview AS PRP ON PRP.ProductID = PP.ProductID ​

GROUP BY OD.ProductID, PP.[Name], PRP.Rating, PRP.ReviewerName, PRP.Comments

FOR JSON PATH, ROOT ('Medium Query 3'), INCLUDE\_NULL\_VALUES

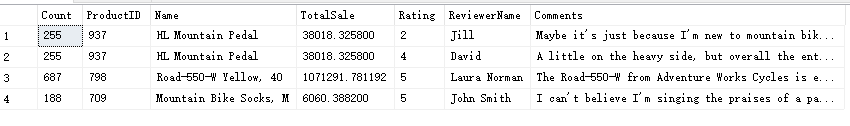
Table:

|  |  |
| --- | --- |
| SalesOrderDetail | ProductId,Quantity,OrerDate |
| Product | Name,ProductId |
| ProductReview | Rating,ReviewerName,Comments |

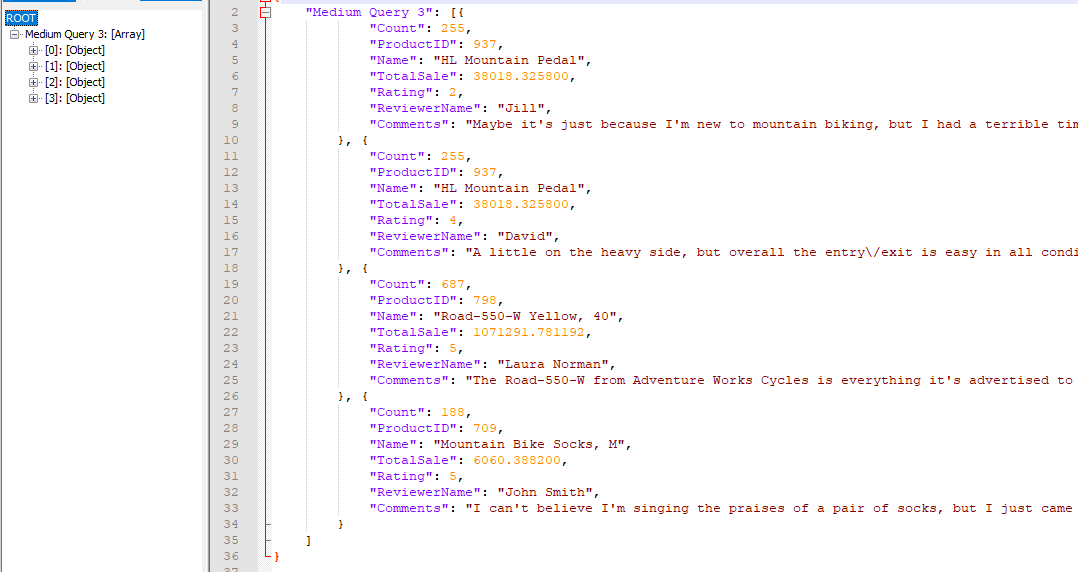
Sort by:

|  |  |  |
| --- | --- | --- |
| SalesOrderDetail | ProductId | ASC |
| Product | Name | ASC |
| ProductReview | Rating,ReviewerName,Comment | ASC |

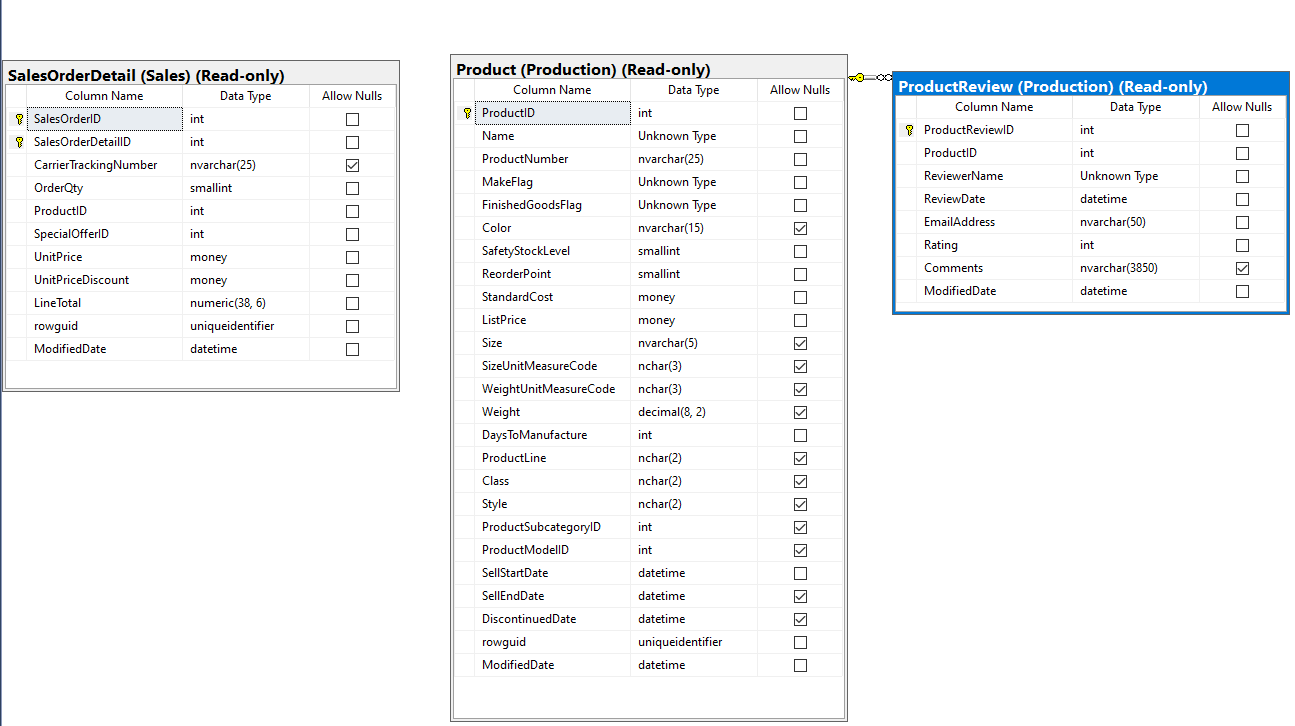
Sample output without JSON:



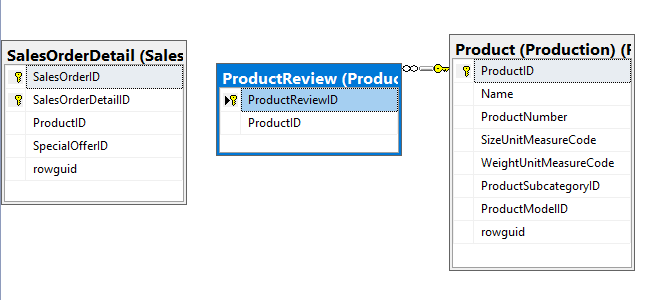
Sample output with JSON:



Standard View:



Key View:



Medium query 4:

Proposition: Performs an Inner join and returns credit card id, number and the business entity associated with it ordered by the credit card id which is greater than 18000

WITHOUT JSON:

USE AdventureWorks2017;

SELECT MIN(C.CreditCardID) AS [Min card]

,C.CardNumber

,B.BusinessEntityID

FROM Sales.CreditCard AS C

INNER JOIN Sales.PersonCreditCard AS B ON B.CreditCardID = C.CreditCardID

AND C.CreditCardID > 18000

GROUP BY C.CardNumber

,B.BusinessEntityID;

--FOR JSON PATH, ROOT ('Medium Query 4'), INCLUDE\_NULL\_VALUES

WITH JSON:

USE AdventureWorks2017;

SELECT MIN(C.CreditCardID) AS [Min card]

,C.CardNumber

,B.BusinessEntityID

FROM Sales.CreditCard AS C

INNER JOIN Sales.PersonCreditCard AS B ON B.CreditCardID = C.CreditCardID

AND C.CreditCardID > 18000

GROUP BY C.CardNumber

,B.BusinessEntityID;

FOR JSON PATH, ROOT ('Medium Query 4'), INCLUDE\_NULL\_VALUES

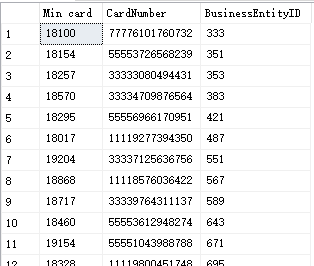
Table:

|  |  |
| --- | --- |
| CreditCard | CreditCardId,CardNumber |
| PersonCreditCard | BusinessEntityId,CreditCardId |

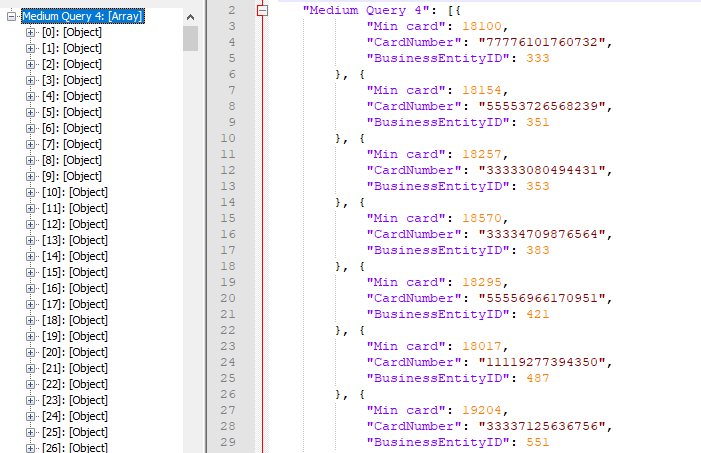
Sort by:

|  |  |  |
| --- | --- | --- |
| CreditCard | CardNumber | ASC |
| PersonCreditCard | BusinessEntityId | ASC |

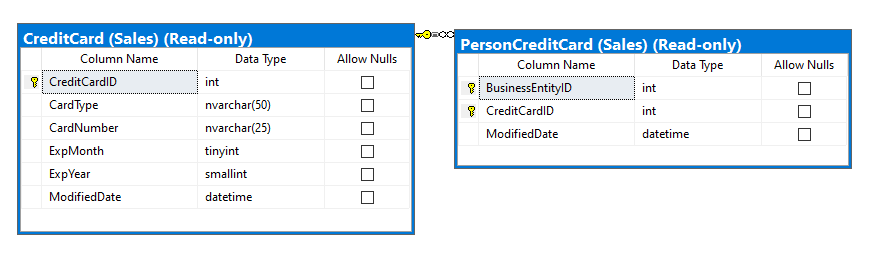
Sample output without JSON:



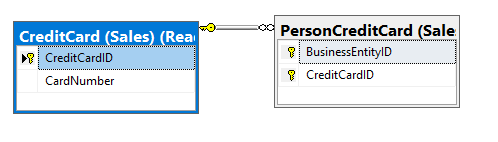
Sample output JSON:



Standard View:



Key view:



Medium query 5:

Proposition: this query return all customers, and for each return a Yes/No value depending on whether the customer placed orders on December 29, 2011(best)

WITHOUT JSON:

use Northwinds2020TSQLV6;

SELECT DISTINCT C.CustomerID

,C.CustomerCompanyName

,CASE

WHEN O.OrderId IS NOT NULL

THEN 'YES'

ELSE 'NO'

END AS HasOrderOn20111229

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O ON O.CustomerID = C.CustomerID

AND O.OrderDate = '20111229';

--FOR JSON PATH, ROOT ('Medium Query 5'), INCLUDE\_NULL\_VALUES

WITH JSON:

use Northwinds2020TSQLV6;

SELECT DISTINCT C.CustomerID

,C.CustomerCompanyName

,CASE

WHEN O.OrderId IS NOT NULL

THEN 'YES'

ELSE 'NO'

END AS HasOrderOn20111229

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O ON O.CustomerID = C.CustomerID

AND O.OrderDate = '20111229';

FOR JSON PATH, ROOT ('Medium Query 5'), INCLUDE\_NULL\_VALUES

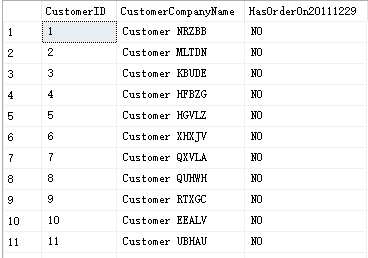
Table:

|  |  |
| --- | --- |
| Customer | CustomerCompanyName,CustomerId, |
| Order | OrderDate,CustomerId, |

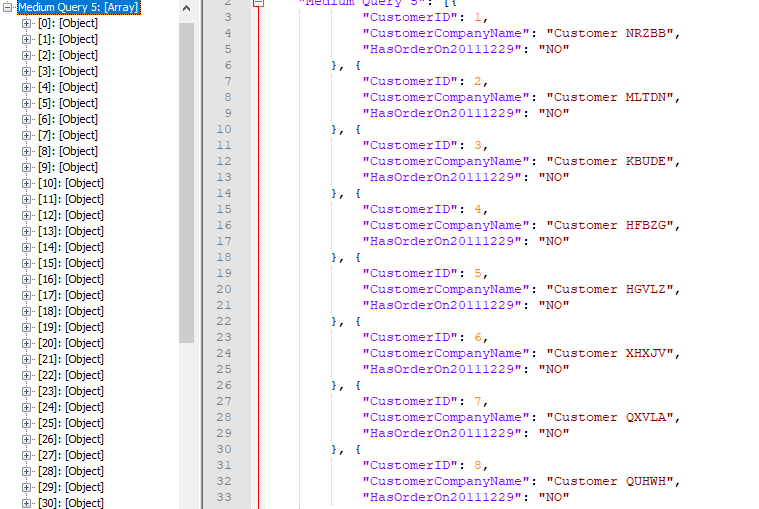
Sort by:

|  |  |  |
| --- | --- | --- |
| Orders | Orderdate | ASC |

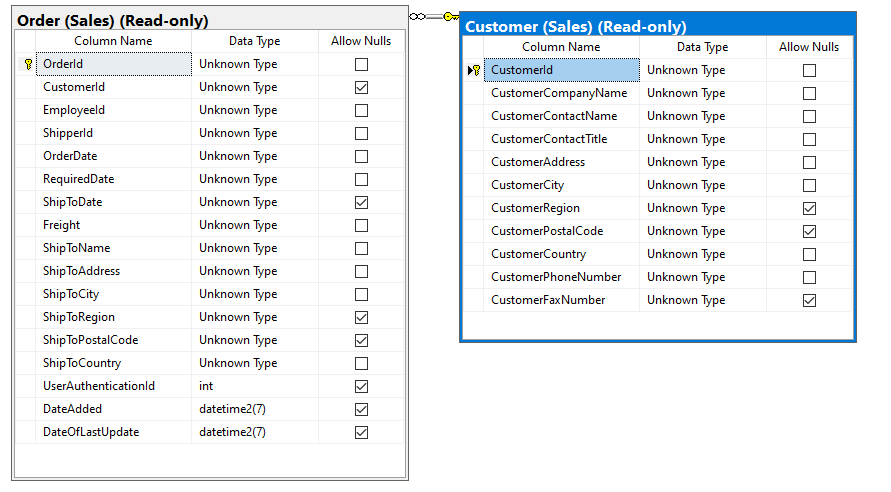
Sample output without JSON:



Sample output with JSON:



Standard View:



Key View:



Medium query 6:

Proposition:How many orders were there per month in descending order.Tables used: Sales.OrderDetails, Sales.Orders

WITHOUT JSON:

use TSQLV4;

select sum(od.qty) as Quantity, month(o.orderdate) as [Month]

from Sales.OrderDetails as od

inner join Sales.Orders as O

on od.orderid = O.orderid

group by month(o.orderdate)

Order by Quantity desc

--for json path, root('Medium Query 6'), INCLUDE\_NULL\_VALUES

WITH JSON:

use TSQLV4;

select sum(od.qty) as Quantity, month(o.orderdate) as [Month]

from Sales.OrderDetails as od

inner join Sales.Orders as O

on od.orderid = O.orderid

group by month(o.orderdate)

Order by Quantity desc

for json path, root('Medium Query 6'), INCLUDE\_NULL\_VALUES

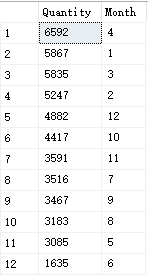
Tables:

|  |  |
| --- | --- |
| Table name | Column name |
| orderDetails | Qty,orderid |
| Orders | Orderid,orderdate |

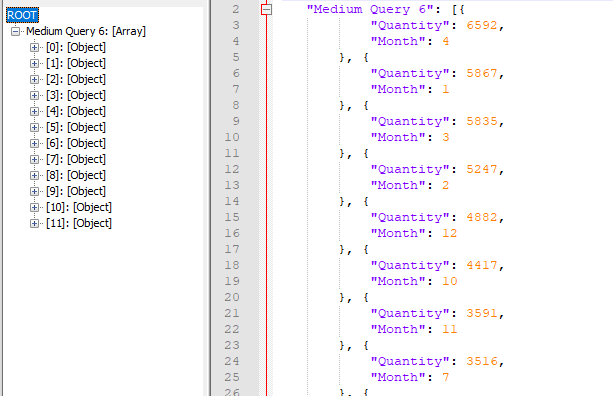
Sort by:

|  |  |  |
| --- | --- | --- |
| OrderDetail | Quantity | ASC |

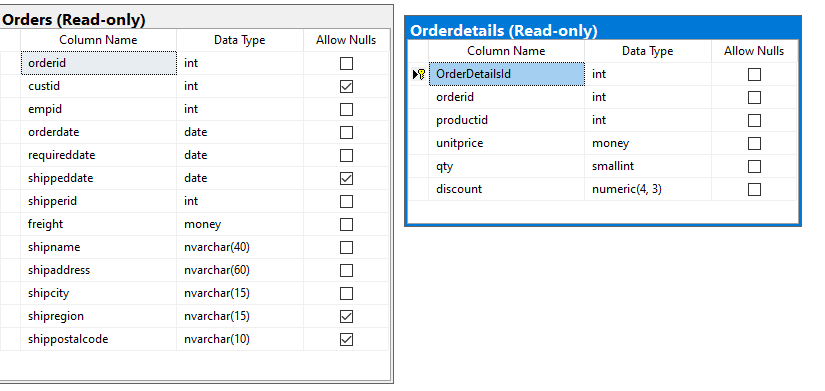
Sample output without JSON:



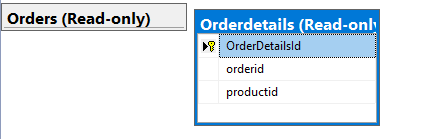
Sample output with JSON:



Standard View:



Key view:



Medium query 7:

Proposition: what are the job candidates pay history

use human resources.jobcandidate and humanresources.employeepayhistory tables

WITHOUT JSON:

use AdventureWorks2017

select jc.businessEntityID as beid, Max(eph.rate) as [pay rate]

from HumanResources.JobCandidate as jc

inner join

HumanResources.EmployeePayHistory as eph

on jc.BusinessEntityID = eph.BusinessEntityID

group by jc.BusinessEntityID

order by [pay rate]

--for json path, root('Medium Query 7'), INCLUDE\_NULL\_VALUES

WITH JSON:

use AdventureWorks2017

select jc.businessEntityID as beid, Max(eph.rate) as [pay rate]

from HumanResources.JobCandidate as jc

inner join

HumanResources.EmployeePayHistory as eph

on jc.BusinessEntityID = eph.BusinessEntityID

group by jc.BusinessEntityID

order by [pay rate]

for json path, root('Medium Query 7'), INCLUDE\_NULL\_VALUES

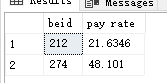
Table:

|  |  |
| --- | --- |
| JobCandidate | BusinessEntityId |
| EmployeePayHistory | Rate,BusinessEntityId, |

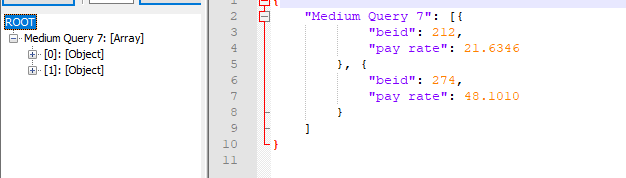
Sort by:

|  |  |  |
| --- | --- | --- |
| JobCandidate | businessEntityId,rate | ASC |

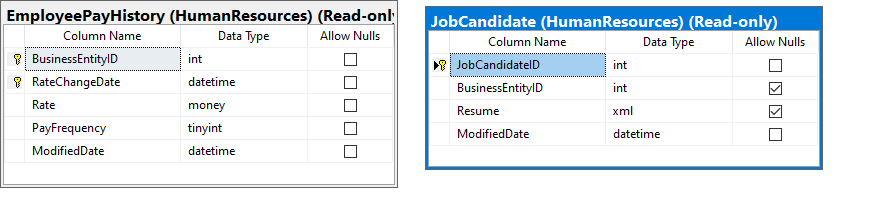
**Sample output without JSON:**



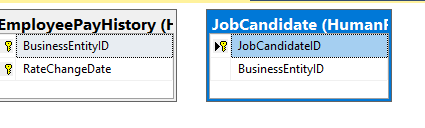
**Sample output with JSON:**



Standard View:



Key View:



**Medium query 8:**

**Proposition:** this query returnan empty set because of the comparison with the NULL.

WITHOUT JSON:

use Northwinds2020TSQLV6;

SELECT CustomerId

,CustomerCompanyName

FROM Sales.Customer AS C

WHERE NOT EXISTS (

SELECT \*

FROM Sales.[Order] AS O

WHERE O.CustomerId = C.CustomerId

);

--FOR JSON PATH, ROOT ('Medium Query 8'), INCLUDE\_NULL\_VALUES

WITH JSON:

use Northwinds2020TSQLV6;

SELECT CustomerId

,CustomerCompanyName

FROM Sales.Customer AS C

WHERE NOT EXISTS (

SELECT \*

FROM Sales.[Order] AS O

WHERE O.CustomerId = C.CustomerId

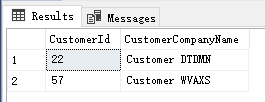
)

FOR JSON PATH, ROOT ('Medium Query 8'), INCLUDE\_NULL\_VALUES

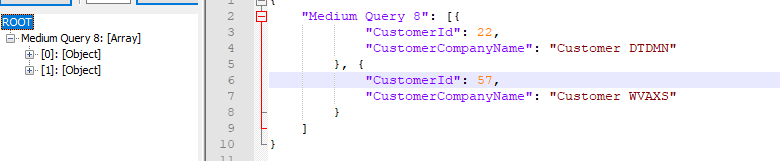
Table used :

|  |  |
| --- | --- |
| Table name | Column name |
| customer | customerId,customerCompanyName |
| Order | orderId,orderDate |

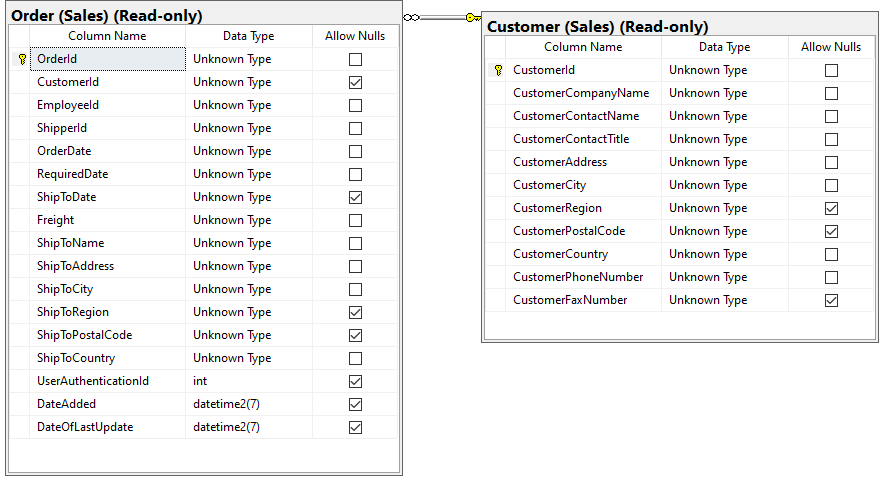
Sample output without JSON:



Sample output with JSON:



Standard View:



Key View:



Complex query:

Complex query 1:

Proposition: return the customers who place order that contain product 12

Without JSON:

use Northwinds2020TSQLV6;

SELECT CustomerId

,CustomerCompanyName

FROM Sales.Customer AS C

WHERE EXISTS (

SELECT \*

FROM Sales.[Order] AS O

WHERE O.CustomerId = C.CustomerId

AND EXISTS (

SELECT \*

FROM Sales.OrderDetail AS OD

WHERE OD.OrderId = O.OrderId

AND OD.ProductId = 12

)

);

--FOR JSON PATH, ROOT ('complex query 1'), INCLUDE\_NULL\_VALUES

With JSON:

use Northwinds2020TSQLV6;

SELECT CustomerId

,CustomerCompanyName

FROM Sales.Customer AS C

WHERE EXISTS (

SELECT \*

FROM Sales.[Order] AS O

WHERE O.CustomerId = C.CustomerId

AND EXISTS (

SELECT \*

FROM Sales.OrderDetail AS OD

WHERE OD.OrderId = O.OrderId

AND OD.ProductId = 12

)

)

FOR JSON PATH, ROOT ('complex query 1'), INCLUDE\_NULL\_VALUES

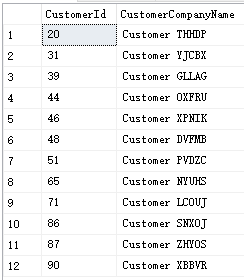
TABLE:

|  |  |
| --- | --- |
| Table name | Column name |
| Order | CustomerId,OrderId |
| OrderDetail | Orderid,ProductId,Quantity, |
| Customer | CustomerId, CustomerCompanyName,CustomerCountry |

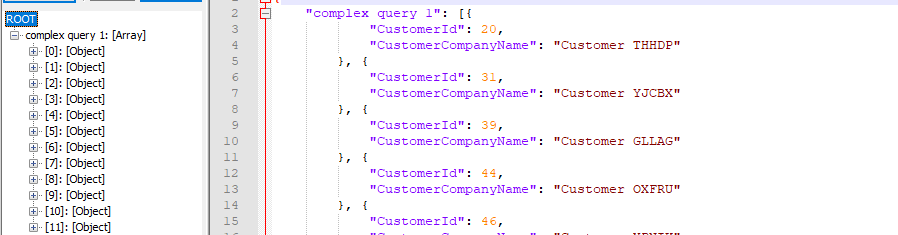
Sort By:

|  |  |  |
| --- | --- | --- |
| Table name | Column name | SORT BY |
| Customer | customerId,customercompanyname,customercity  customercountry | ASC |

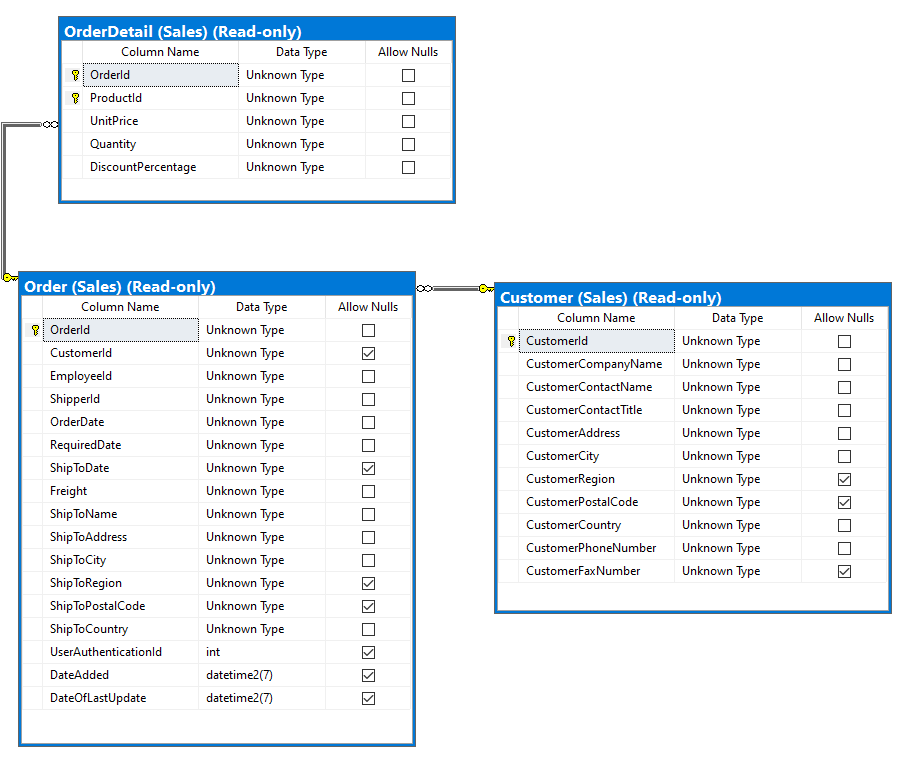
Sample output without JSON:



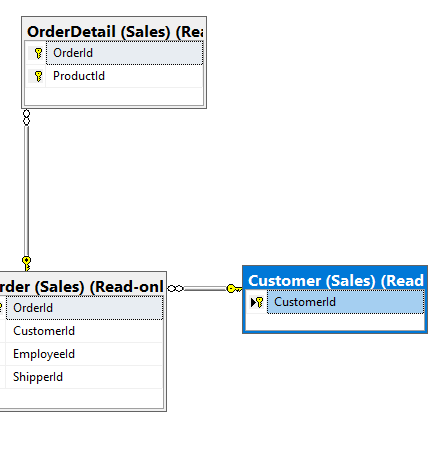
Sample output with JSON:



Standard View:



Key View:



Complex query 2:

Proposition: return US customers, and for each customer return the total number of order and total quantities

Without JSON:

use Northwinds2020TSQLV6;

SELECT C.CustomerId

,count(DISTINCT O.orderid) AS numorders

,SUM(OD.Quantity) AS totalqty

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O ON O.CustomerId = C.CustomerId

INNER JOIN Sales.OrderDetail AS OD ON OD.OrderId = O.OrderId

WHERE C.CustomerCountry = N'USA'

GROUP BY C.CustomerId;

--FOR JSON PATH, ROOT ('complex query 2'), INCLUDE\_NULL\_VALUES

WITH JSON:

use Northwinds2020TSQLV6;

SELECT C.CustomerId

,count(DISTINCT O.orderid) AS numorders

,SUM(OD.Quantity) AS totalqty

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O ON O.CustomerId = C.CustomerId

INNER JOIN Sales.OrderDetail AS OD ON OD.OrderId = O.OrderId

WHERE C.CustomerCountry = N'USA'

GROUP BY C.CustomerId

FOR JSON PATH, ROOT ('complex query 2'), INCLUDE\_NULL\_VALUES

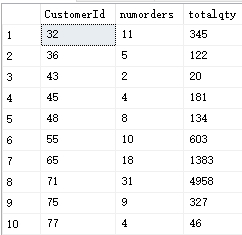
TABLE:

|  |  |
| --- | --- |
| Table name | Column name |
| Order | CustomerId,OrderId |
| OrderDetail | Orderid,ProductId,Quantity, |
| Customer | CustomerId, CustomerCompanyName,CustomerCountry |

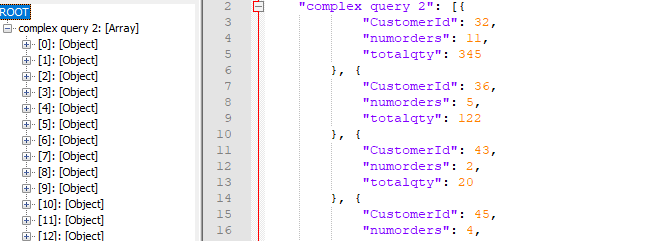
Order By

|  |  |  |
| --- | --- | --- |
| Table name | Column name | SORT BY |
| Customer | customerId,customercompanyname,customercity  customercountry | ASC |

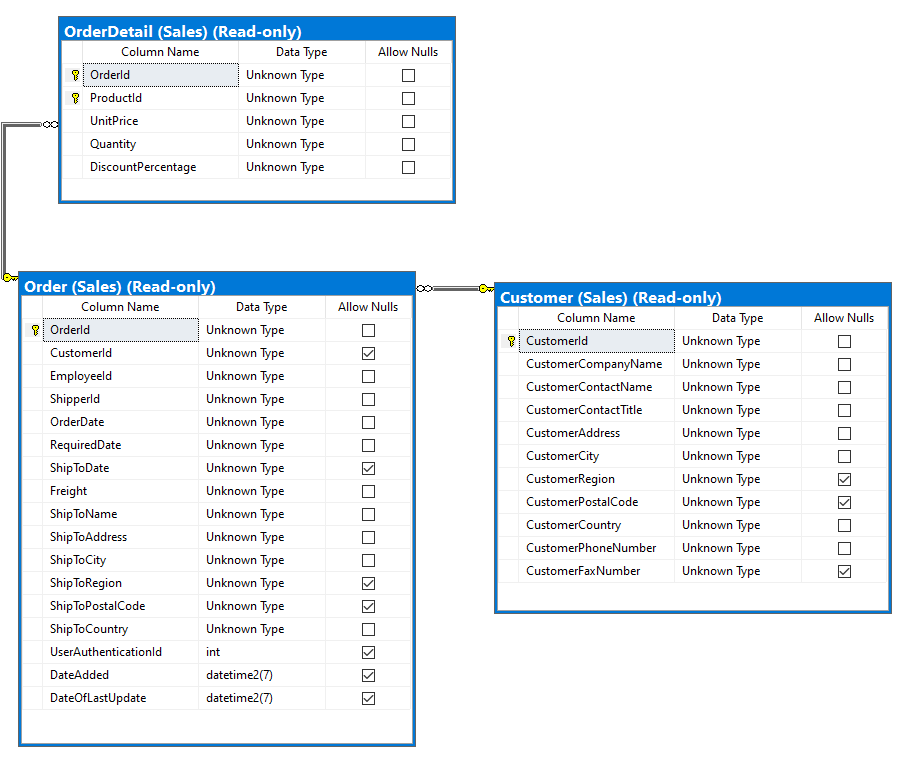
Sample output without JSON:



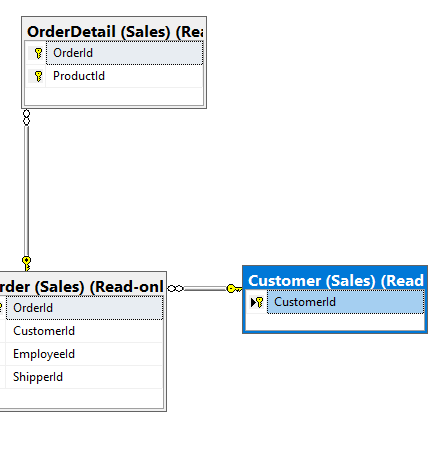
Sample output with JSON:



Standard View:



Key View:



Complex query 3:

Proposition: this code return customers with no orders in the output with left outer join in the second join

Without JSON:

use Northwinds2020TSQLV6;

SELECT C.CustomerId

,O.OrderId

,OD.ProductId

,OD.Quantity

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O ON C.CustomerId = O.CustomerId

LEFT OUTER JOIN Sales.OrderDetail AS OD ON O.OrderId = OD.OrderId;

--FOR JSON PATH, ROOT ('complex query 3'), INCLUDE\_NULL\_VALUES

WITH JSON:

use Northwinds2020TSQLV6;

SELECT C.CustomerId

,O.OrderId

,OD.ProductId

,OD.Quantity

FROM Sales.Customer AS C

LEFT OUTER JOIN Sales.[Order] AS O ON C.CustomerId = O.CustomerId

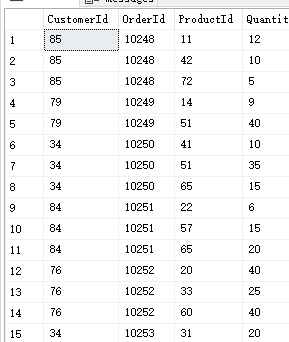
LEFT OUTER JOIN Sales.OrderDetail AS OD ON O.OrderId = OD.OrderId

FOR JSON PATH, ROOT ('complex query 3'), INCLUDE\_NULL\_VALUES

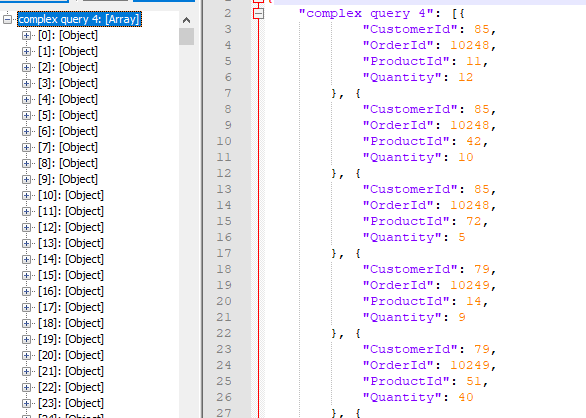
TABLE:

|  |  |
| --- | --- |
| Table name | Column name |
| Order | CustomerId,OrderId |
| OrderDetail | Orderid,ProductId,Quantity, |
| Customer | CustomerId, CustomerCompanyName,CustomerCountry |

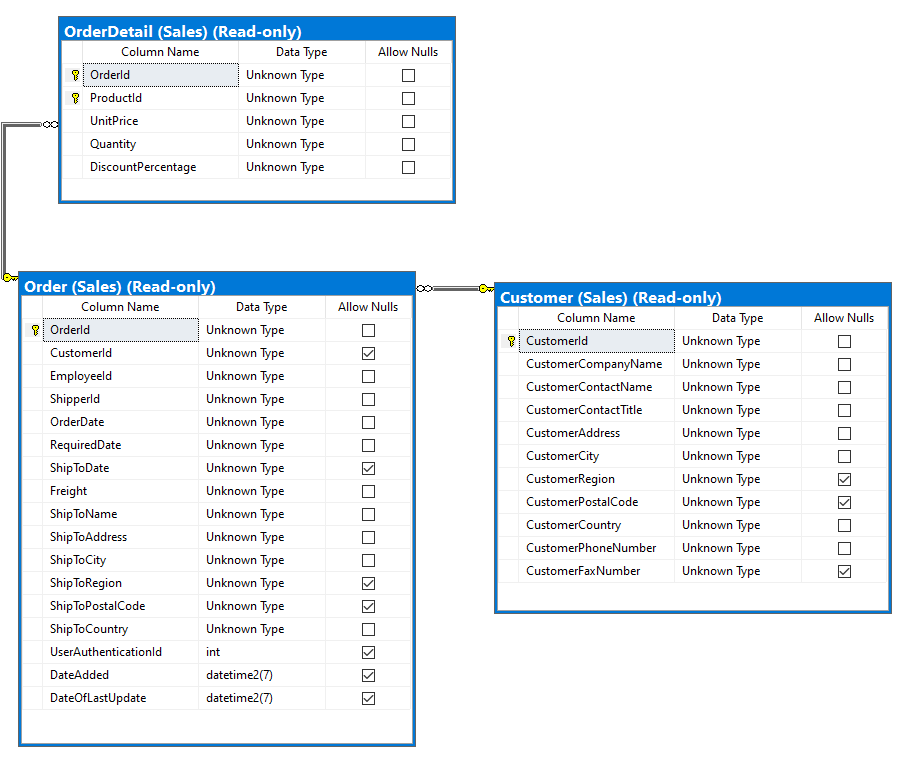
Sample output without JSON:



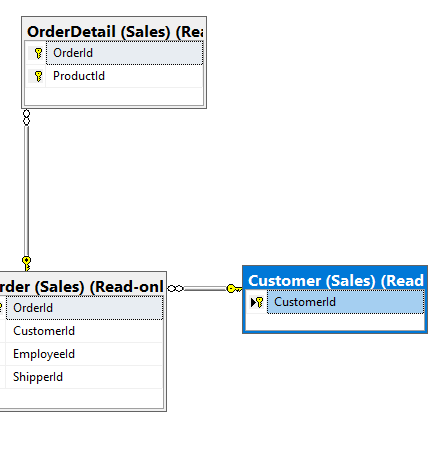
Sample output with JSON:



Standard View:



Key View:



Complex query 4:

Proposition:return all the orders abd tgeur travel from production to cutomer in 2016

Without JSON:

use Northwinds2020TSQLV6;

select S.SupplierId, S.SupplierCompanyName, [LocationE] = concat(S.SupplierCity,', ', S.SupplierCountry),

P.ProductId,P.ProductName,

O.OrderId, O.OrderDate, C.CustomerCompanyName,

[Location] = concat(C.CustomerCity,', ',CustomerCountry)

from [Production].[Supplier] as S

inner join [Production].[Product] as P

on S.SupplierId = P.SupplierId

inner join [Sales].[Order] as O

on P.SupplierId = O.ShipperId

inner join [Sales].[Customer] as C

on C.CustomerId = O.CustomerId

where year(O.orderdate) = 2016

group by S.SupplierId, S.SupplierCompanyName,concat(S.SupplierCity,', ', S.SupplierCountry),

P.ProductId,P.ProductName,

O.OrderId, O.OrderDate, C.CustomerCompanyName,concat(C.CustomerCity,', ',CustomerCountry)

order by O.orderdate desc

--FOR JSON PATH, ROOT ('complex query 4'), INCLUDE\_NULL\_VALUES

WITH JSON:

use Northwinds2020TSQLV6;

select S.SupplierId, S.SupplierCompanyName, LOCATIONE = concat(S.SupplierCity,', ', S.SupplierCountry),

P.ProductId,P.ProductName,

O.OrderId, O.OrderDate, C.CustomerCompanyName,

LOCATION = concat(C.CustomerCity,', ',CustomerCountry)

from Production.Supplier as S

inner join Production.Product as P

on S.SupplierId = P.SupplierId

inner join Sales.[Order] as O

on P.SupplierId = O.ShipperId

inner join Sales.[Customer] as C

on C.CustomerId = O.CustomerId

where year(O.orderdate) = 2016

group by S.SupplierId, S.SupplierCompanyName,concat(S.SupplierCity,', ', S.SupplierCountry),

P.ProductId,P.ProductName,

O.OrderId, O.OrderDate, C.CustomerCompanyName,concat(C.CustomerCity,', ',CustomerCountry)

order by O.orderdate desc

FOR JSON PATH, ROOT ('complex query 5'), INCLUDE\_NULL\_VALUES

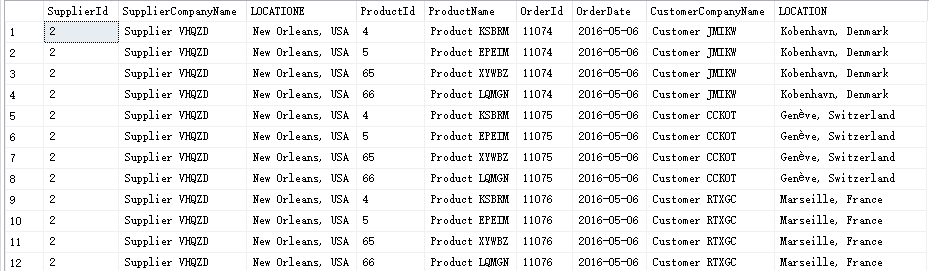
Table:

|  |  |
| --- | --- |
| Table name | Column name |
| Supplier | SupplierId,SupplierCompanyName,Locatione,SupplierCountry |
| Product | ProductId,ProductName |
| Order | OrderId,OrderDate |
| Customer | customerCompanyName,CustomerCity,CustomerCountry |

Order By:

|  |  |  |
| --- | --- | --- |
| Table name | Column name | Sort order |
| Order | Orderdate | ASC |

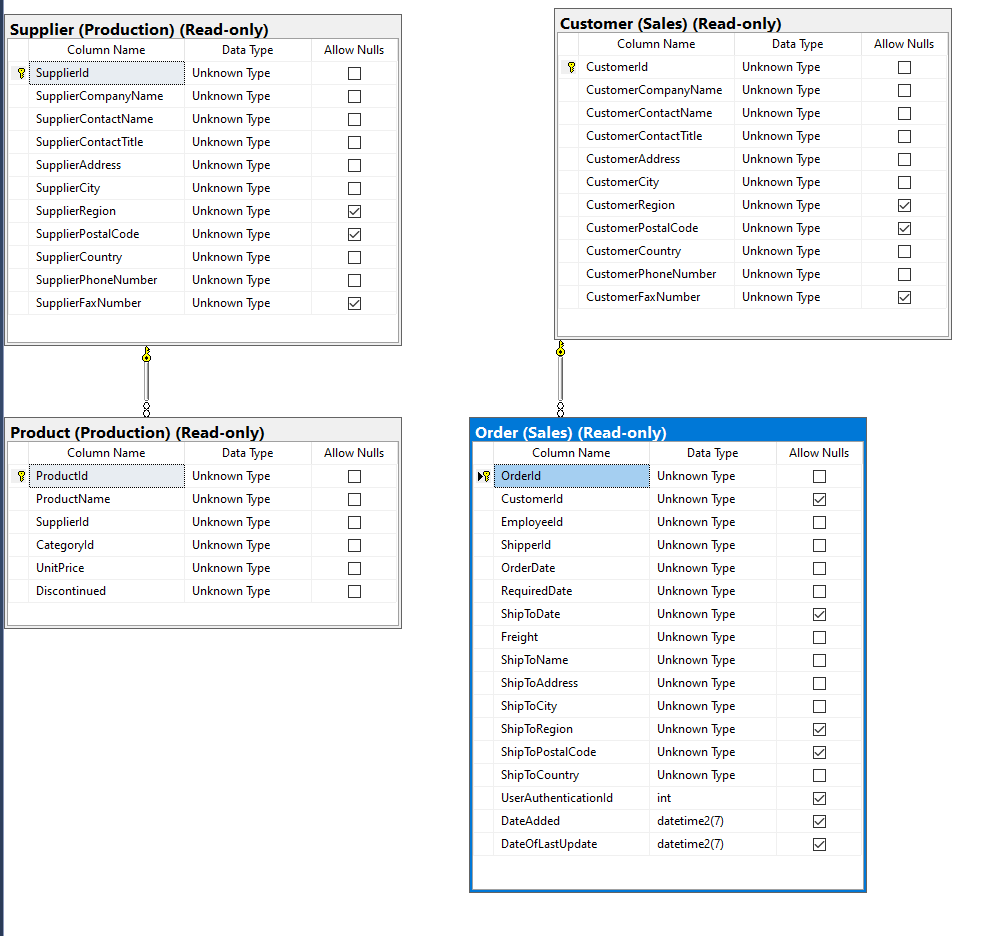
Sample output without JSON:



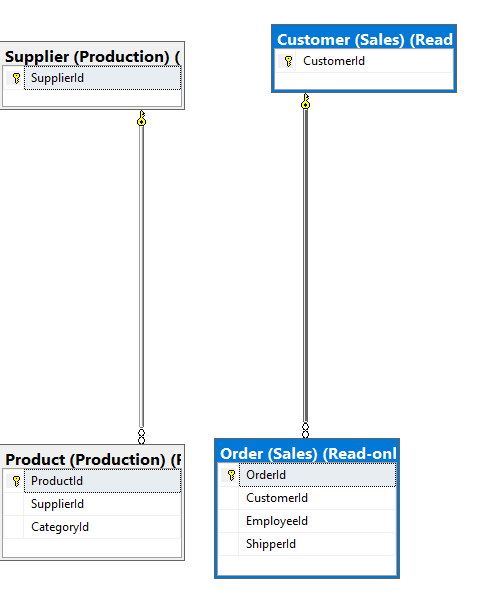
Sample output with JSON:



Standard View:



Key view:



Complex query 5:

Proposition: this query returns customers who ordered product 20 with nesting EXISTS predicates with correlated subqueries.

Without JSON:

use Northwinds2020TSQLV6;

SELECT CustomerId

,CustomerCompanyName

FROM Sales.Customer AS C

WHERE EXISTS (

SELECT \*

FROM Sales.[Order] AS O

WHERE O.CustomerId = C.CustomerId

AND EXISTS (

SELECT \*

FROM Sales.OrderDetail AS OD

WHERE OD.OrderId = O.OrderId

AND OD.ProductId = 20

)

);

--FOR JSON PATH, ROOT ('complex query 5'), INCLUDE\_NULL\_VALUES

WITH JSON:

use Northwinds2020TSQLV6;

SELECT CustomerId

,CustomerCompanyName

FROM Sales.Customer AS C

WHERE EXISTS (

SELECT \*

FROM Sales.[Order] AS O

WHERE O.CustomerId = C.CustomerId

AND EXISTS (

SELECT \*

FROM Sales.OrderDetail AS OD

WHERE OD.OrderId = O.OrderId

AND OD.ProductId = 20

)

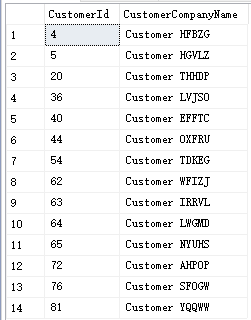
)

FOR JSON PATH, ROOT ('complex query 5'), INCLUDE\_NULL\_VALUES

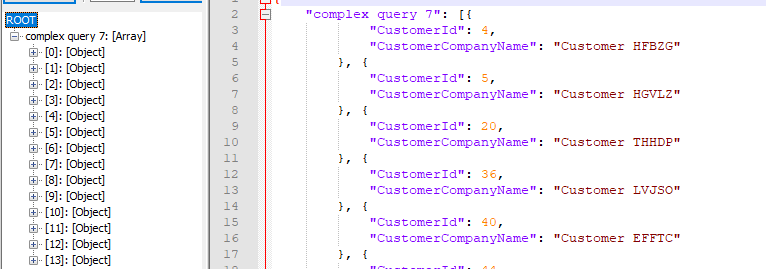
Table:

|  |  |
| --- | --- |
| Table name | Column name |
| Order | OrderId,OrderDate |
| Customer | customerCompanyName,CustomerCity,CustomerCountry |
| OrderDetail | OrderId |

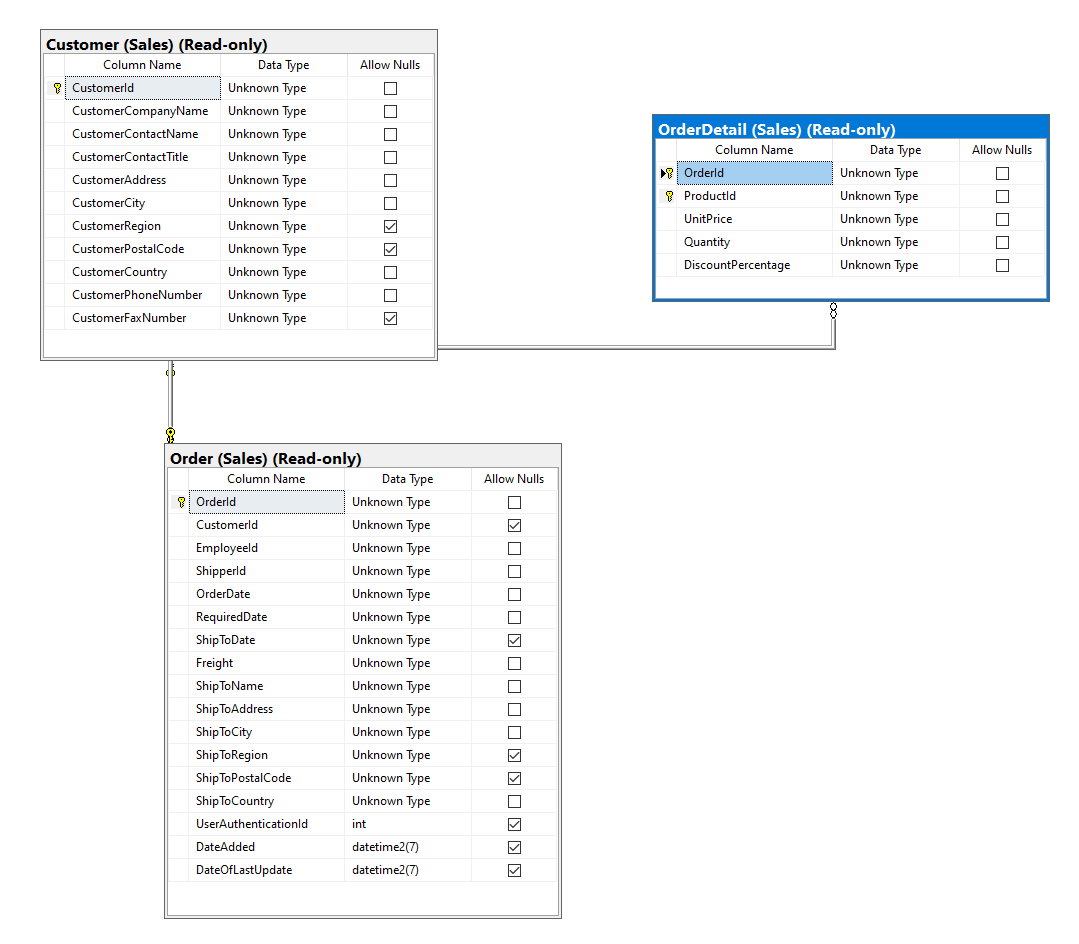
Sample output without JSON:



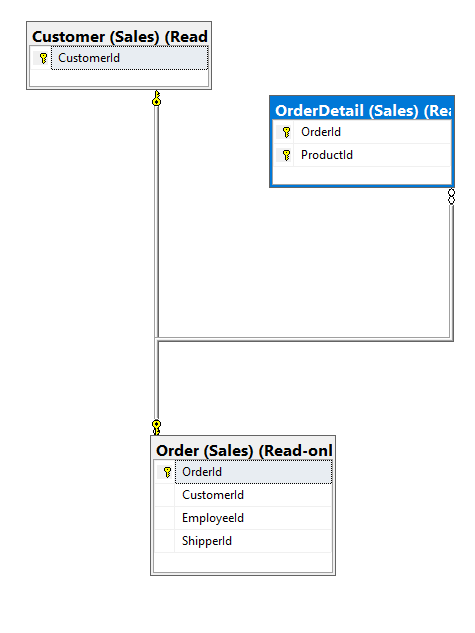
Sample output with JSON:



Standard View:



Key View:



Complex query 6:

Proposition:return the totalprice for each customer placed

Without JSON:

use DB1045\_LiuHaibo413;

DROP FUNCTION IF exists sales.udf\_GetTotalPrice

GO

CREATE FUNCTION Sales.udf\_GetTotalPrice(

@Quantity INT,

@UnitPrice DEC(10,2),

@DiscountPercentage DEC(4,2)

)

RETURNS DEC(10,2)

AS

BEGIN

RETURN @quantity \* @UnitPrice \* (1 - @DiscountPercentage);

END;

GO

SELECT

OD.OrderId,

SUM(Sales.udf\_GetTotalPrice(Quantity, UnitPrice, DiscountPercentage)) net\_amount

FROM

Sales.OrderDetail AS OD

GROUP BY

OD.OrderId

ORDER BY

net\_amount DESC

--FOR JSON PATH, ROOT ('complex query 6'), INCLUDE\_NULL\_VALUES

WITH JSON:

use DB1045\_LiuHaibo413;

DROP FUNCTION IF exists sales.udf\_GetTotalPrice

GO

CREATE FUNCTION Sales.udf\_GetTotalPrice(

@Quantity INT,

@UnitPrice DEC(10,2),

@DiscountPercentage DEC(4,2)

)

RETURNS DEC(10,2)

AS

BEGIN

RETURN @quantity \* @UnitPrice \* (1 - @DiscountPercentage);

END;

GO

SELECT

OD.OrderId,

SUM(Sales.udf\_GetTotalPrice(Quantity, UnitPrice, DiscountPercentage)) net\_amount

FROM

Sales.OrderDetail AS OD

GROUP BY

OD.OrderId

ORDER BY

net\_amount DESC

FOR JSON PATH, ROOT ('complex query 7'), INCLUDE\_NULL\_VALUES

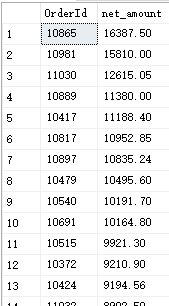
Table:

|  |  |
| --- | --- |
| Table name | Column name |
| Order | orderId |
| OrderDetail | OrderId,quantity,unitprice,discountpercentage |

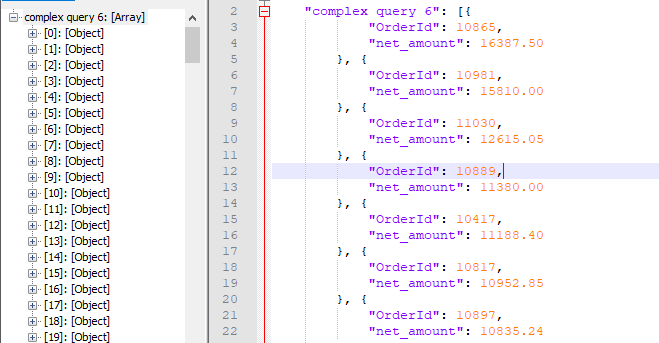
Order By:

|  |  |  |
| --- | --- | --- |
| Table name | Column Name | SORT ORDER |
| orderDetail | orderId | ASC |

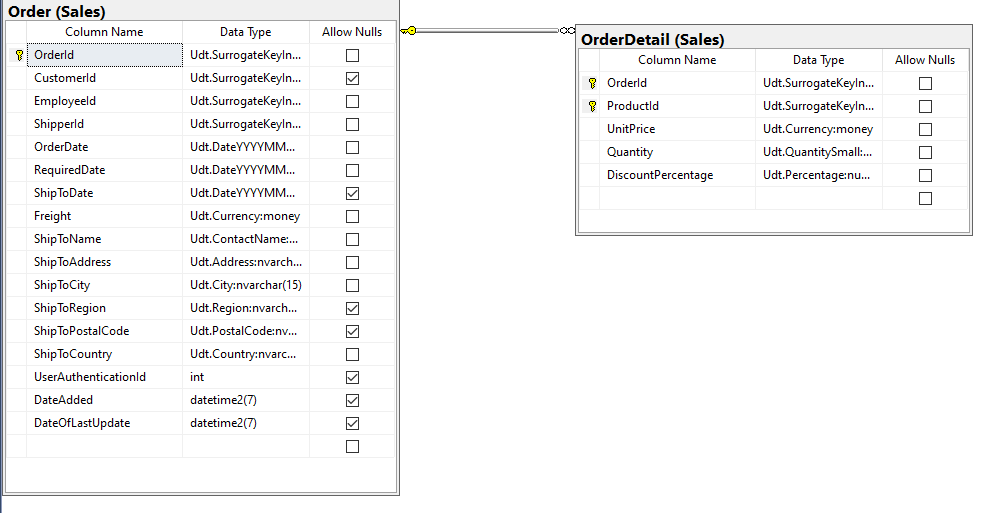
Sample output without JSON:



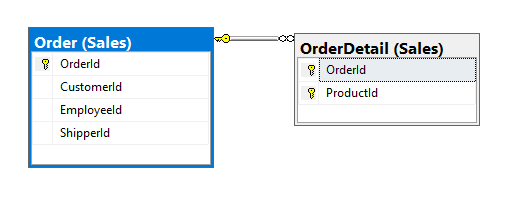
Sample output with JSON:



Standard View:



Key View:



Complex query 7:

Proposition: return all customers with their other orders made in the year 2016 and total price they are paying per order

Without JSON:

use DB1045\_LiuHaibo413;

drop function if exists [dbo].totalDiscoutPrice

go

create function [dbo].totalDiscountPrice(@unitPrice decimal(7,2), @qty int, @discount decimal(4,3))

returns decimal(7,2)

as

begin

declare @result decimal(7,2)

set @result = (@unitPrice \* @qty) - (@unitPrice \* @qty\*@discount)

return @result

end;

go

-------

select C.customerId, C.CustomerCompanyName,concat(C.CustomerCity,C.CustomerCountry) as Location,

O.orderId, O.OrderDate, OD.ProductId,OD.UnitPrice,OD.Quantity, OD.DiscountPercentage,

[dbo].totalDiscountPrice(OD.UnitPrice,OD.Quantity, OD.DiscountPercentage) as TotalDiscountedPrice

from [sales].[customer] as C

inner join [sales].[order] as O

on C.customerId = O.customerId

inner join [sales].[OrderDetail] as OD

on O.orderId = OD.orderId

where year(O.orderdate) = 2016

group by C.customerId, C.CustomerCompanyName,concat(C.CustomerCity,C.CustomerCountry),

O.orderId, O.OrderDate, OD.ProductId,OD.UnitPrice,OD.Quantity, OD.[DiscountPercentage],

[dbo].totalDiscountPrice(OD.UnitPrice,OD.Quantity, OD.DiscountPercentage)

order by orderdate desc

--FOR JSON PATH, ROOT ('complex query 7'), INCLUDE\_NULL\_VALUES

WITH JSON:

use DB1045\_LiuHaibo413;

drop function if exists [dbo].totalDiscoutPrice

go

create function [dbo].totalDiscountPrice(@unitPrice decimal(7,2), @qty int, @discount decimal(4,3))

returns decimal(7,2)

as

begin

declare @result decimal(7,2)

set @result = (@unitPrice \* @qty) - (@unitPrice \* @qty\*@discount)

return @result

end;

go

-------

select C.customerId, C.CustomerCompanyName,concat(C.CustomerCity,C.CustomerCountry) as Location,

O.orderId, O.OrderDate, OD.ProductId,OD.UnitPrice,OD.Quantity, OD.DiscountPercentage,

[dbo].totalDiscountPrice(OD.UnitPrice,OD.Quantity, OD.DiscountPercentage) as TotalDiscountedPrice

from [sales].[customer] as C

inner join [sales].[order] as O

on C.customerId = O.customerId

inner join [sales].[OrderDetail] as OD

on O.orderId = OD.orderId

where year(O.orderdate) = 2016

group by C.customerId, C.CustomerCompanyName,concat(C.CustomerCity,C.CustomerCountry),

O.orderId, O.OrderDate, OD.ProductId,OD.UnitPrice,OD.Quantity, OD.[DiscountPercentage],

[dbo].totalDiscountPrice(OD.UnitPrice,OD.Quantity, OD.DiscountPercentage)

order by orderdate desc

FOR JSON PATH, ROOT ('complex query 7'), INCLUDE\_NULL\_VALUES

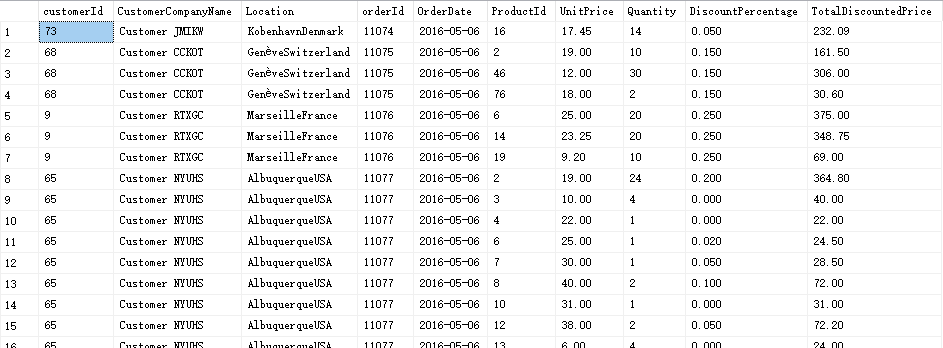
Table:

|  |  |
| --- | --- |
| Table name | Column Name |
| Order | customerId,orderid,orderdate, |
| OrderDetail | Orderid,discountpercentage,Productid,unitprice,quantity |
| customer | customerCompanyName,customerCity,customerCountry  customerId |

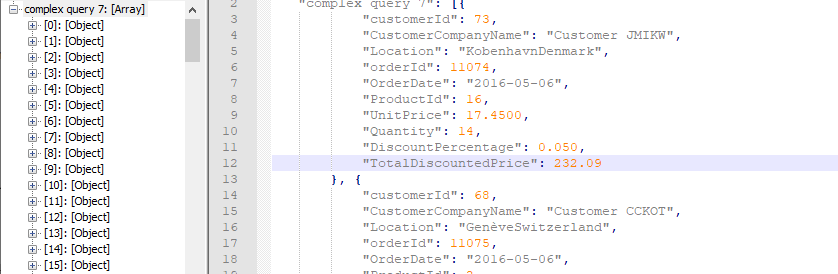
Order By:

|  |  |  |
| --- | --- | --- |
| Table name | Column name | Sort order |
| Order | Orderdate | ASC |

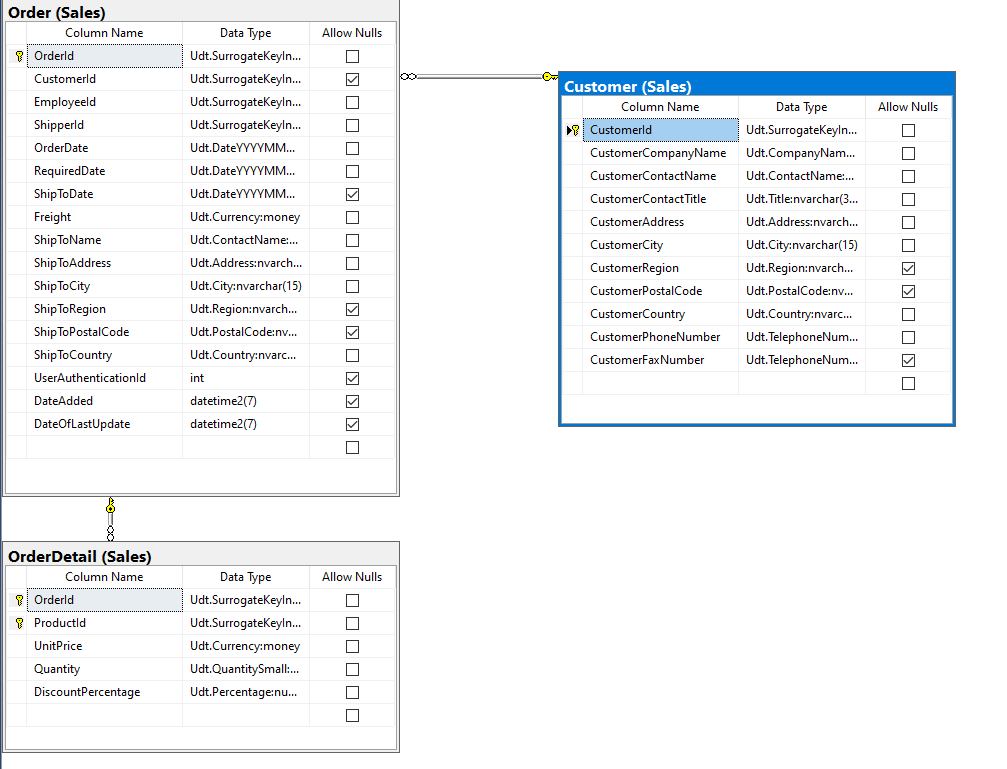
Sample output without JSON:



Sample output with JSON:



Standard View:



Key View:

