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
```

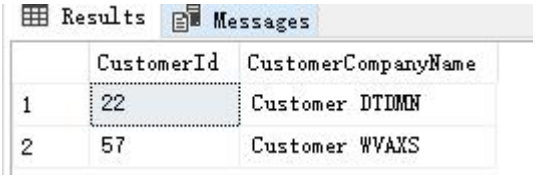
WITH JSON:

```
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
```

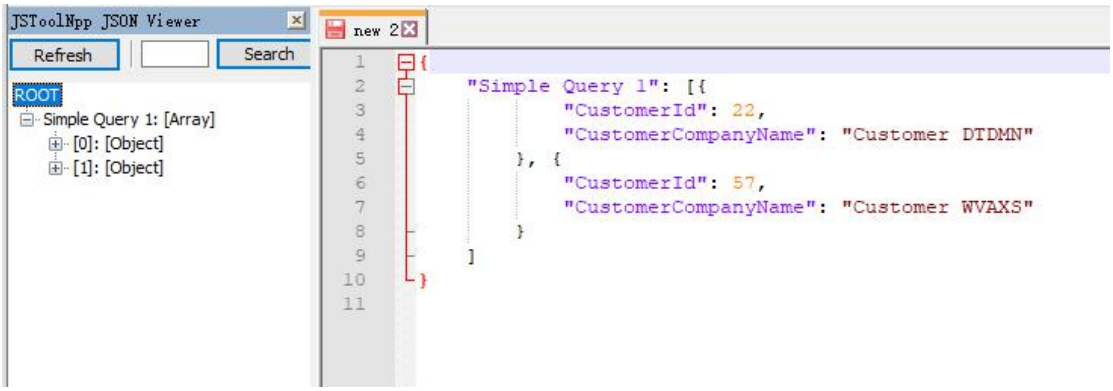
Table used :

Table name	Column name
customer	customerId,customerCompanyName
Order	orderId

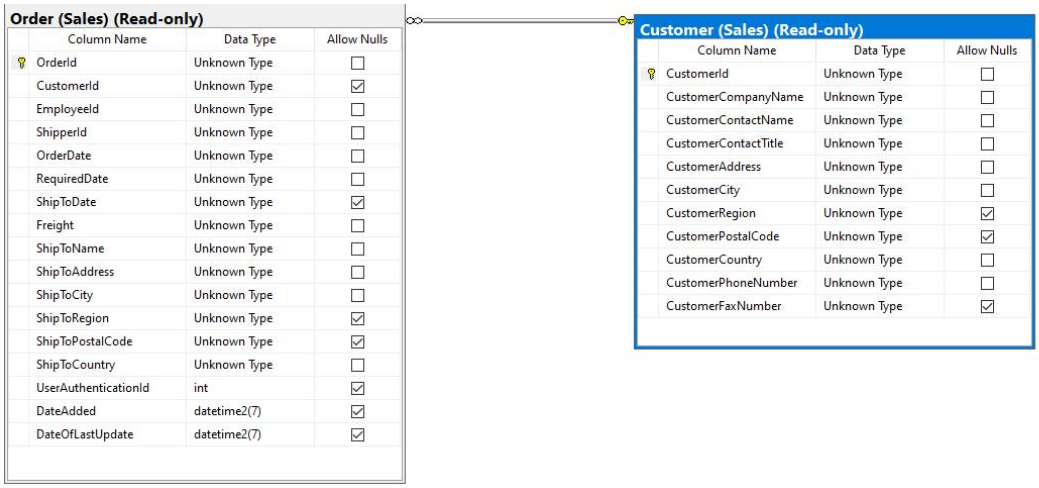
Sample output without JSON:



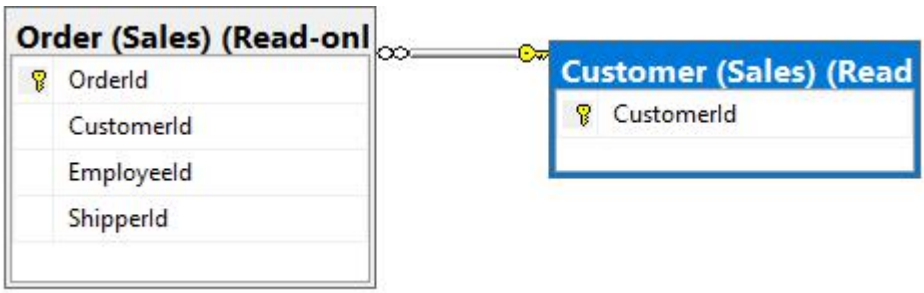
Sample output with JSON:



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

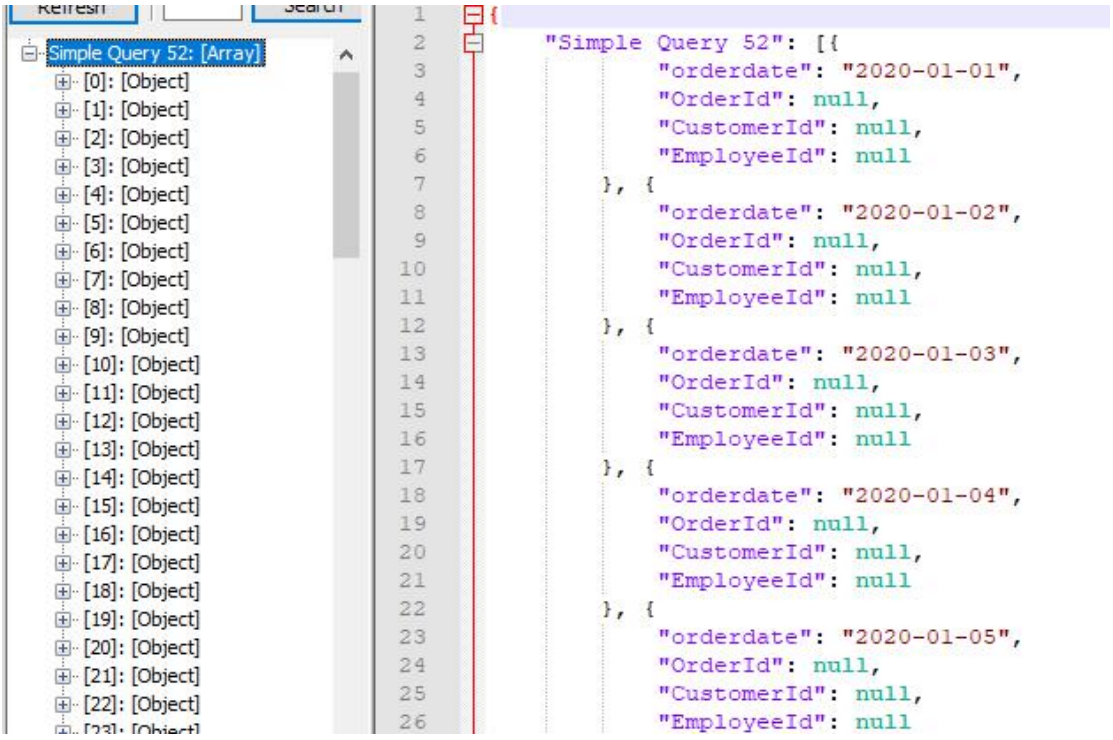
Table name	Column name	Sort order
Order	Orderdate	ASC

Sample output without JSON:

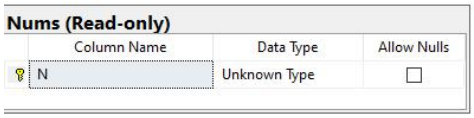
Results | Messages

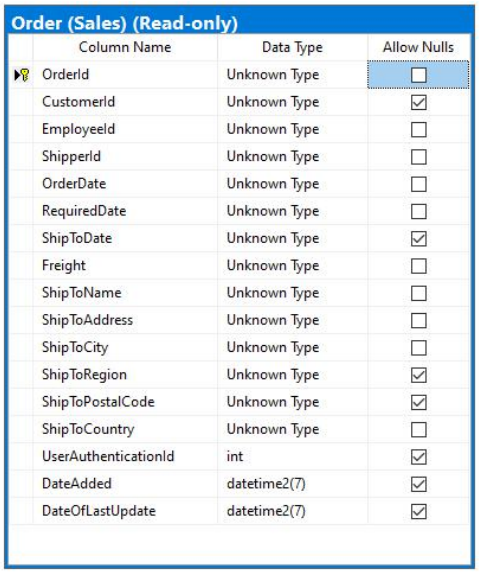
	orderdate	OrderId	CustomerId	EmployeeId
1	2020-01-01	NULL	NULL	NULL
2	2020-01-02	NULL	NULL	NULL
3	2020-01-03	NULL	NULL	NULL
4	2020-01-04	NULL	NULL	NULL
5	2020-01-05	NULL	NULL	NULL
6	2020-01-06	NULL	NULL	NULL
7	2020-01-07	NULL	NULL	NULL
8	2020-01-08	NULL	NULL	NULL
9	2020-01-09	NULL	NULL	NULL
10	2020-01-10	NULL	NULL	NULL
11	2020-01-11	NULL	NULL	NULL
12	2020-01-12	NULL	NULL	NULL
13	2020-01-13	NULL	NULL	NULL
14	2020-01-14	NULL	NULL	NULL
15	2020-01-15	NULL	NULL	NULL
16	2020-01-16	NULL	NULL	NULL
17	2020-01-17	NULL	NULL	NULL
18	2020-01-18	NULL	NULL	NULL
19	2020-01-19	NULL	NULL	NULL
20	2020-01-20	NULL	NULL	NULL

Sample output with JSON:

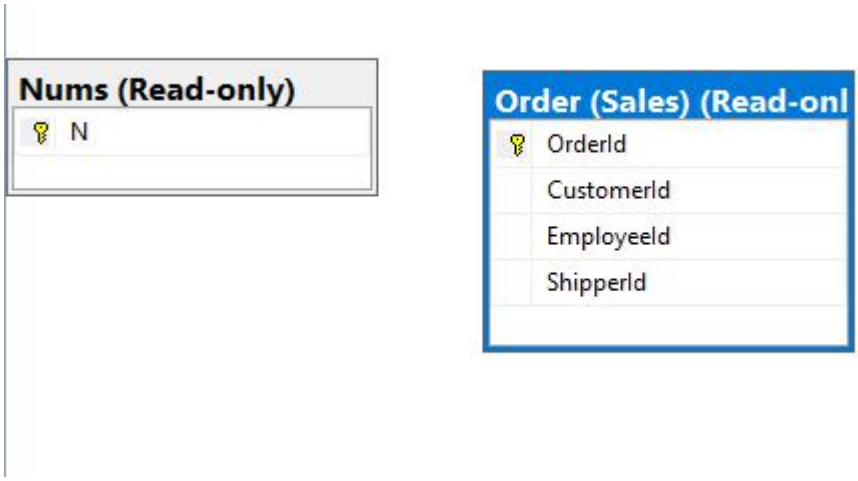


Standard View:





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

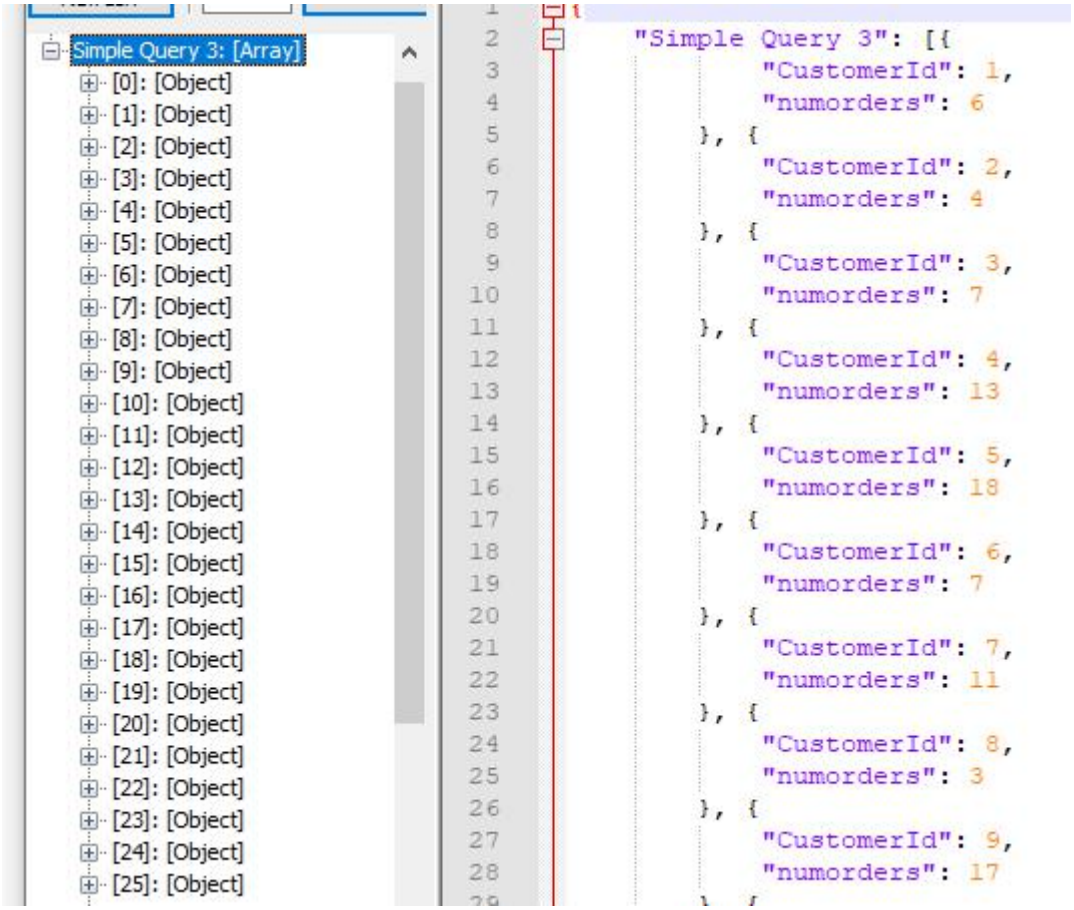
Sort by

Table name	Column name	Sort order
Customer	customerId	ASC

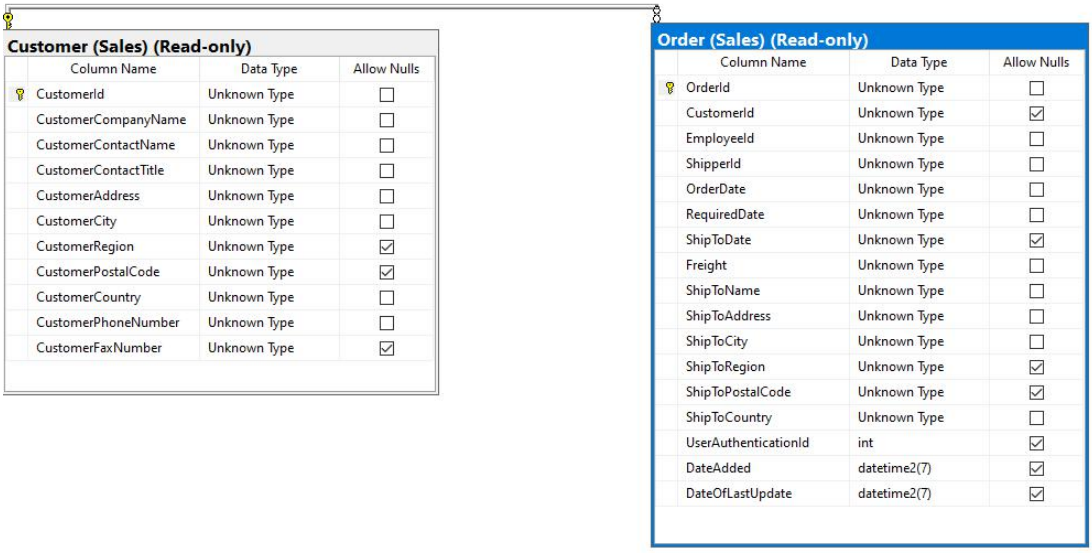
Sample output without JSON:

	CustomerId	numorders
1	1	6
2	2	4
3	3	7
4	4	13
5	5	18
6	6	7
7	7	11
8	8	3
9	9	17
10	10	14
11	11	10
12	12	6
13	13	1
14	14	8
15	15	5
16	16	3
17	17	6
18	18	4
19	19	8

Sample output with JSON:



Standard View:



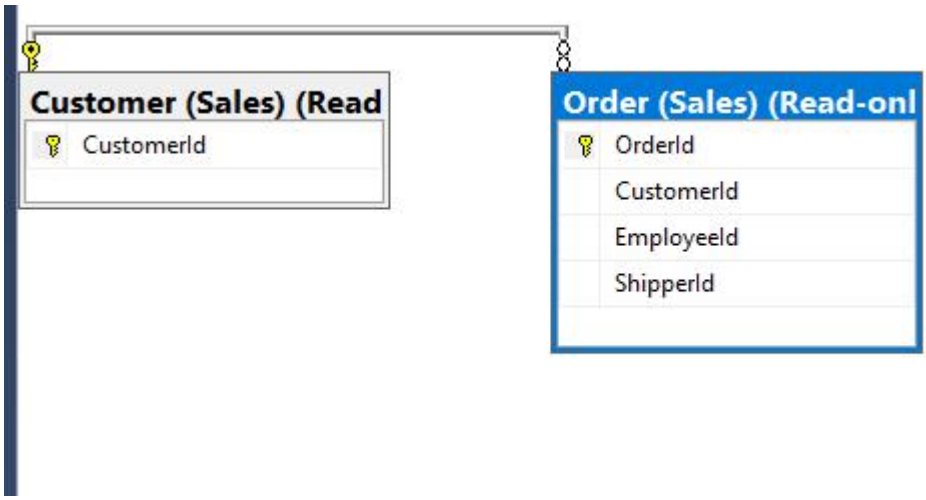
Customer (Sales) (Read-only)

Column Name	Data Type	Allow Nulls
CustomerId	Unknown Type	☐
CustomerCompanyName	Unknown Type	☐
CustomerContactName	Unknown Type	☐
CustomerContactTitle	Unknown Type	☐
CustomerAddress	Unknown Type	☐
CustomerCity	Unknown Type	☐
CustomerRegion	Unknown Type	☑
CustomerPostalCode	Unknown Type	☑
CustomerCountry	Unknown Type	☐
CustomerPhoneNumber	Unknown Type	☐
CustomerFaxNumber	Unknown Type	☑

Order (Sales) (Read-only)

Column Name	Data Type	Allow Nulls
OrderId	Unknown Type	☐
CustomerId	Unknown Type	☑
EmployeeId	Unknown Type	☐
ShipperId	Unknown Type	☐
OrderDate	Unknown Type	☐
RequiredDate	Unknown Type	☐
ShipToDate	Unknown Type	☑
Freight	Unknown Type	☐
ShipToName	Unknown Type	☐
ShipToAddress	Unknown Type	☐
ShipToCity	Unknown Type	☐
ShipToRegion	Unknown Type	☑
ShipToPostalCode	Unknown Type	☑
ShipToCountry	Unknown Type	☐
UserAuthenticationId	int	☑
DateAdded	datetime2(7)	☑
DateOfLastUpdate	datetime2(7)	☑

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
```

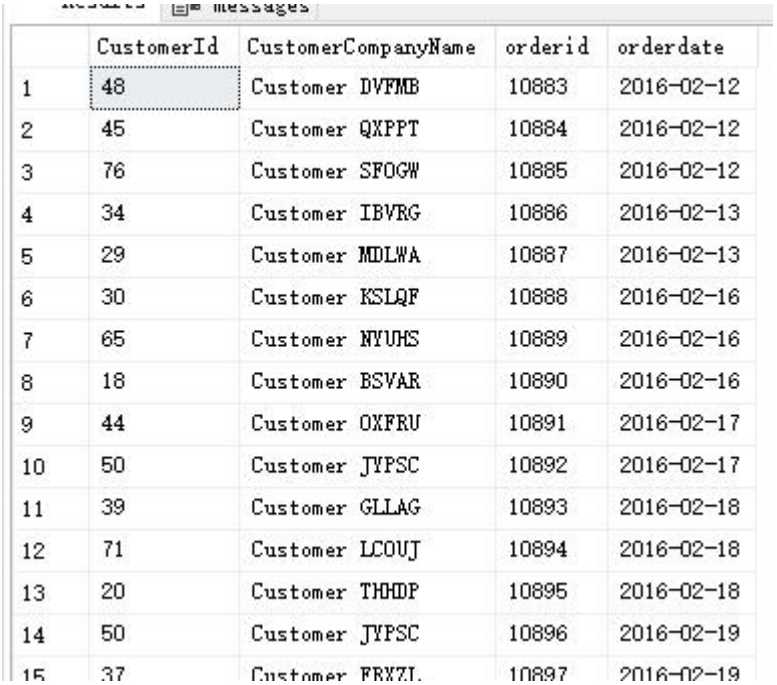
WITH JSON:

```
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 :

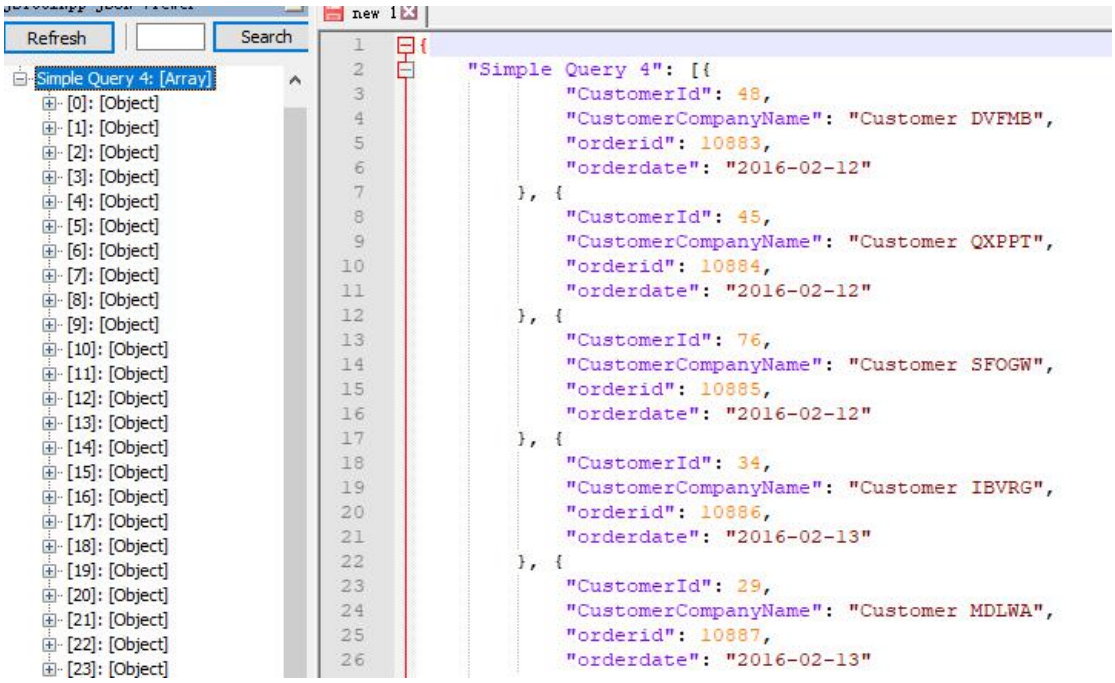
Table name	Column name
customer	customerId,customerCompanyName
Order	orderId,orderDate

Sample output without JSON:

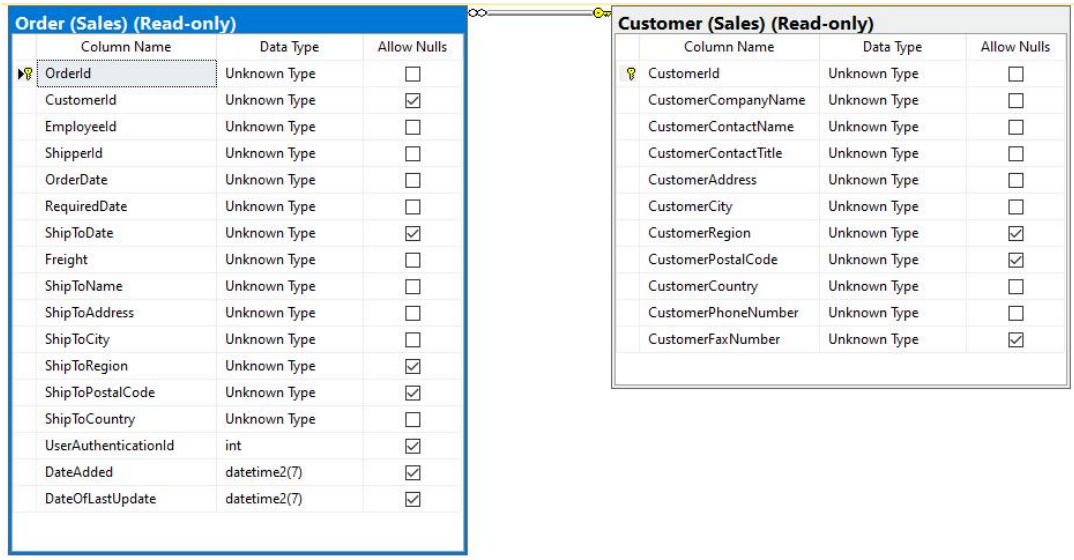


	CustomerId	CustomerCompanyName	orderid	orderdate
1	48	Customer DVFMB	10883	2016-02-12
2	45	Customer QXPPT	10884	2016-02-12
3	76	Customer SFOGW	10885	2016-02-12
4	34	Customer IBVRG	10886	2016-02-13
5	29	Customer MDLWA	10887	2016-02-13
6	30	Customer KSLQF	10888	2016-02-16
7	65	Customer NYUHS	10889	2016-02-16
8	18	Customer BSVAR	10890	2016-02-16
9	44	Customer OXFRU	10891	2016-02-17
10	50	Customer JYPSC	10892	2016-02-17
11	39	Customer GLLAG	10893	2016-02-18
12	71	Customer LCOUJ	10894	2016-02-18
13	20	Customer THHDP	10895	2016-02-18
14	50	Customer JYPSC	10896	2016-02-19
15	37	Customer FRXZL	10897	2016-02-19

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:

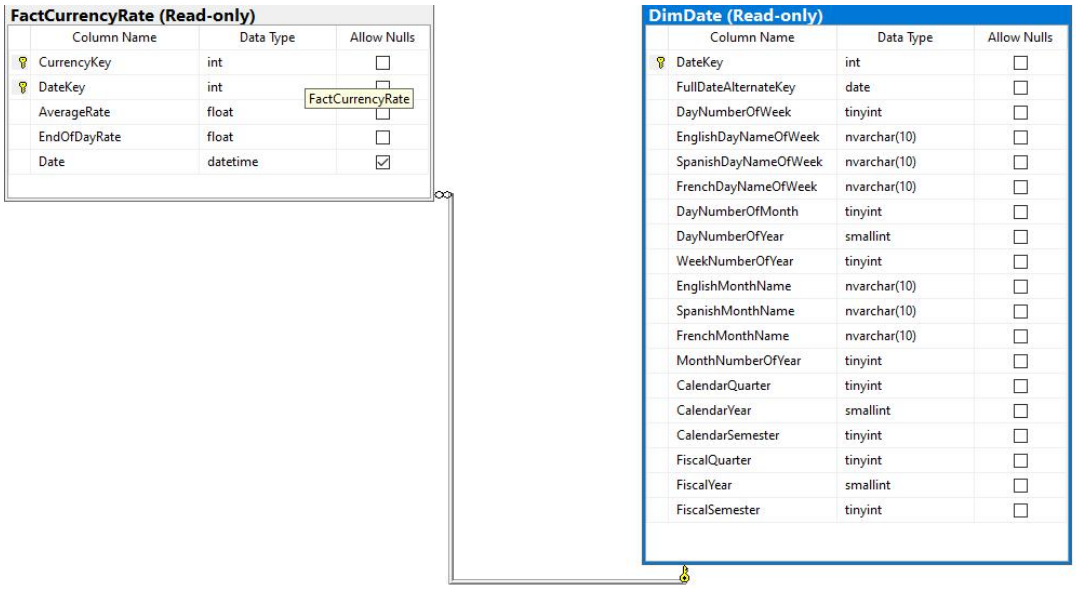
Table name	Column name	Sort order
Dimdate	Spanishmonthname,spanishdaynameofweek,calendar year	ASC
FACTCURRENCYRATE	DATEKEY	ASC

Sample output without JSON:

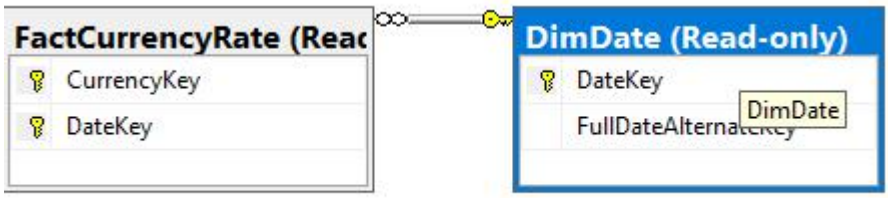
	DateKey	spanishdate
1	20101229	Diciembre Miércoles 2010
2	20101230	Diciembre Jueves 2010
3	20101231	Diciembre Viernes 2010
4	20110101	Enero Sábado 2011
5	20110102	Enero Domingo 2011
6	20110103	Enero Lunes 2011
7	20110104	Enero Martes 2011
8	20110105	Enero Miércoles 2011
9	20110106	Enero Jueves 2011
10	20110107	Enero Viernes 2011
11	20110108	Enero Sábado 2011
12	20110109	Enero Domingo 2011
13	20110110	Enero Lunes 2011
14	20110111	Enero Martes 2011
15	20110112	Enero Miércoles 2011
16	20110113	Enero Jueves 2011
17	20110114	Enero Viernes 2011
18	20110115	Enero Sábado 2011
19	20110116	Enero Domingo 2011
20	20110117	Enero Lunes 2011

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;
```

```sql
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

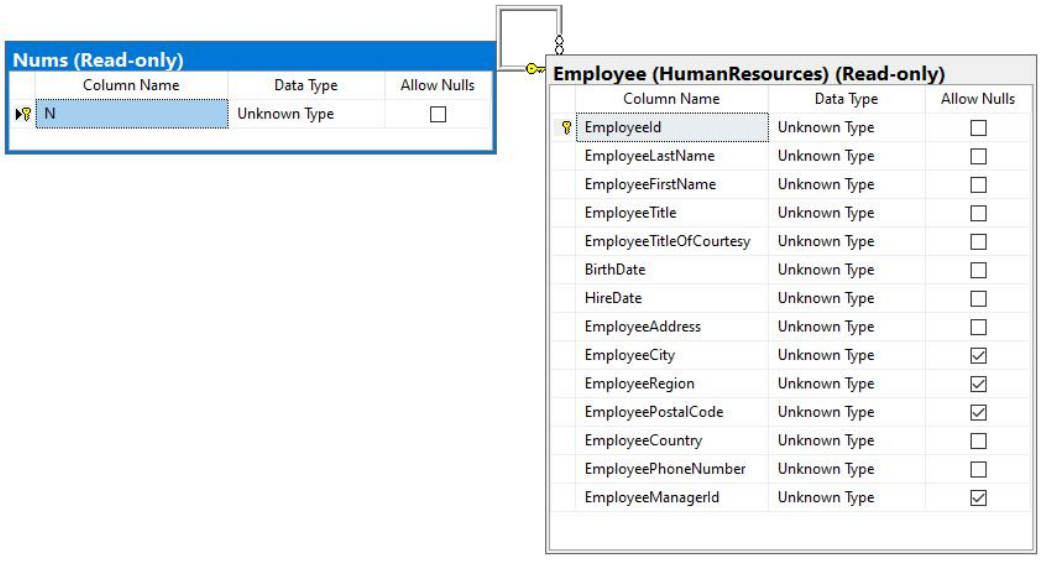
Table name	Column name	Sort by
Nums	N	ASC
Employee	EmployeeId	ASC

Sample output without JSON:

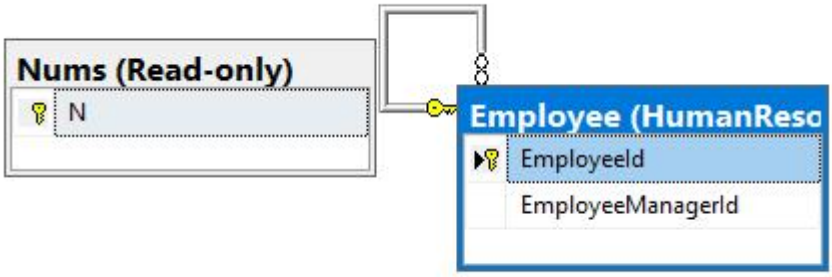
	EmployeeId	EmployeeFirstName	EmployeeLastName	N
1	1	Sara	Davis	1
2	2	Don	Funk	1
3	3	Judy	Lew	1
4	4	Yael	Peled	1
5	5	Sven	Mortensen	1
6	6	Paul	Suurs	1
7	7	Russell	King	1
8	8	Maria	Cameron	1
9	9	Patricia	Doyle	1
10	1	Sara	Davis	2
11	2	Don	Funk	2
12	3	Judy	Lew	2
13	4	Yael	Peled	2
14	5	Sven	Mortensen	2
15	6	Paul	Suurs	2
16	7	Russell	King	2

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
```

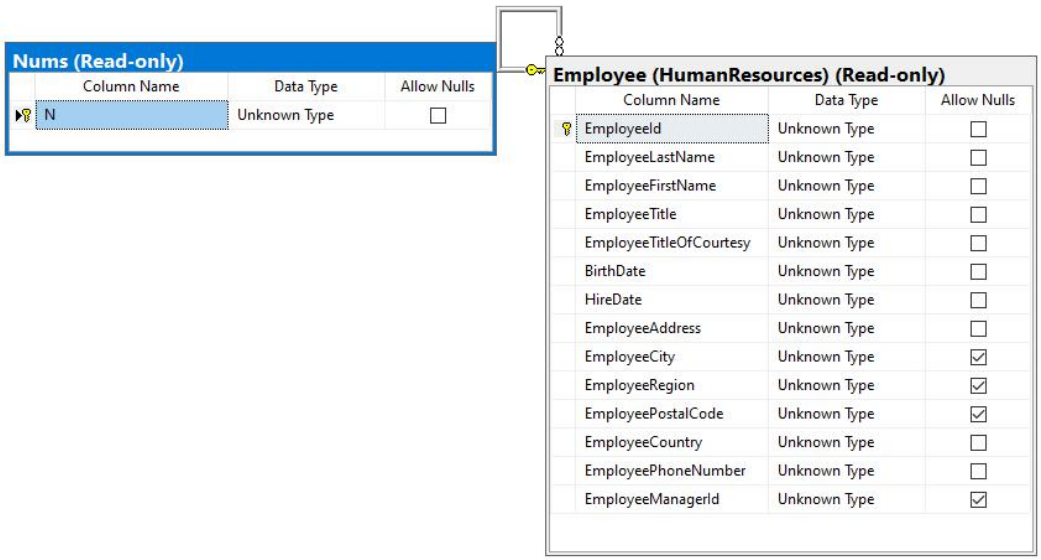
Table used :

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

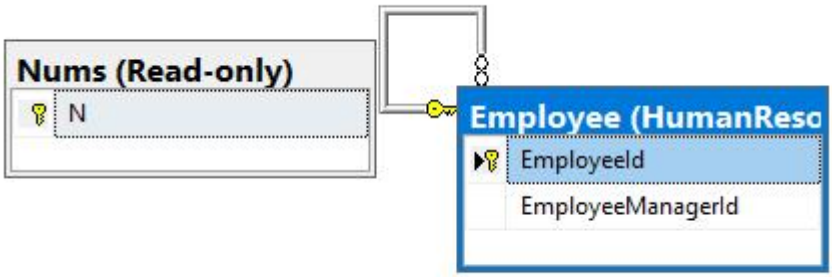
Order BY

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

Standard View:



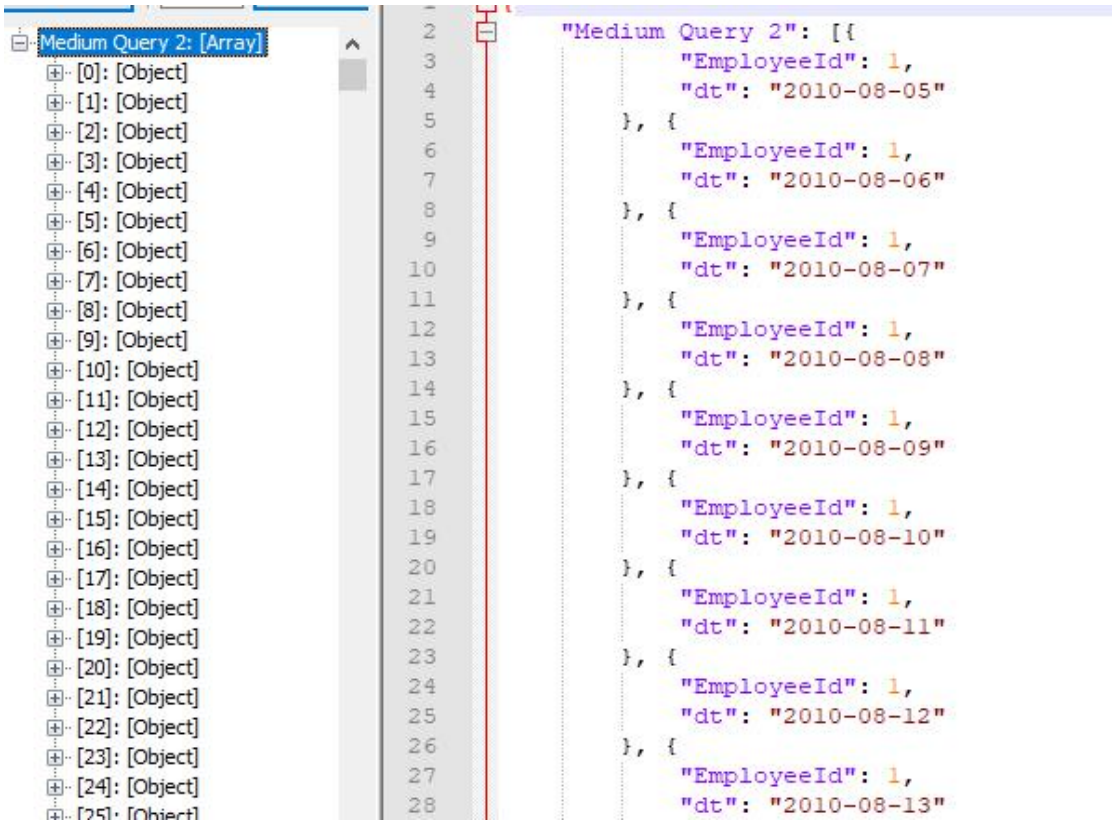
Key View:



Sample output without JSON:

	EmployeeId	dt
1	1	2010-08-05
2	1	2010-08-06
3	1	2010-08-07
4	1	2010-08-08
5	1	2010-08-09
6	1	2010-08-10
7	1	2010-08-11
8	1	2010-08-12
9	1	2010-08-13
10	1	2010-08-14

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

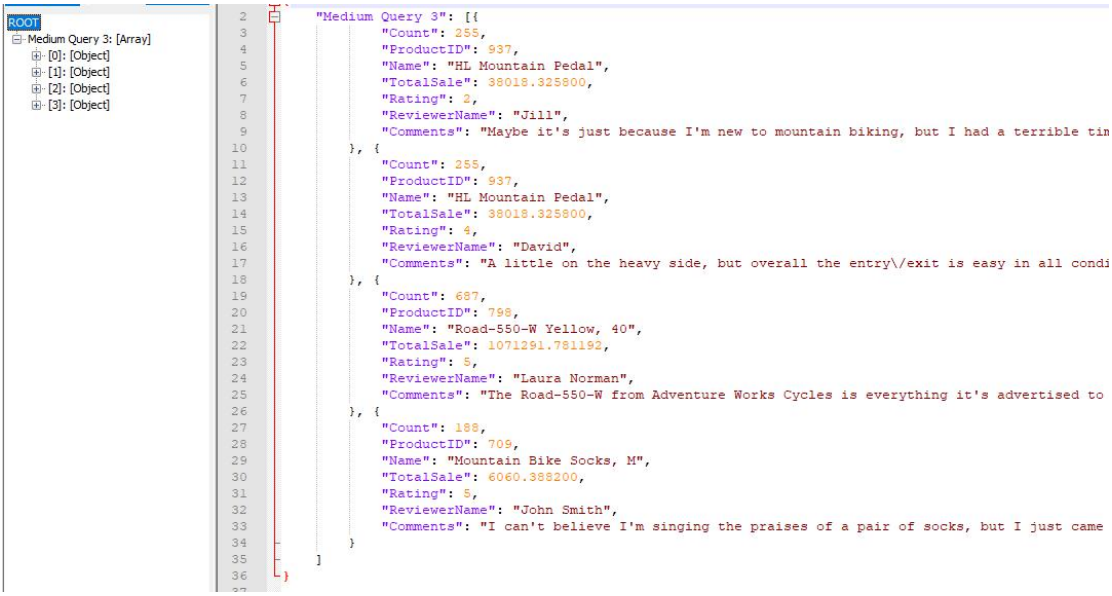
Sort by:

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

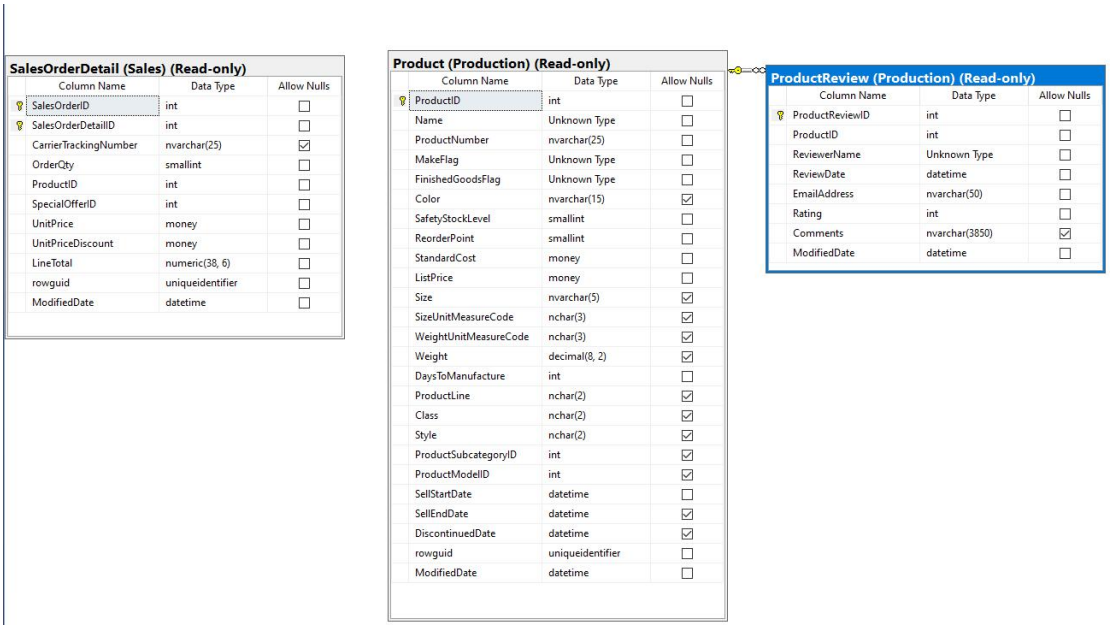
Sample output without JSON:

	Count	ProductID	Name	TotalSale	Rating	ReviewerName	Comments
1	255	937	HL Mountain Pedal	38018.325800	2	Jill	Maybe it's just because I'm new to mountain bik...
2	255	937	HL Mountain Pedal	38018.325800	4	David	A little on the heavy side, but overall the ent...
3	687	798	Road-550-W Yellow, 40	1071291.781192	5	Laura Norman	The Road-550-W from Adventure Works Cycles is e...
4	188	709	Mountain Bike Socks, M	6060.388200	5	John Smith	I can't believe I'm singing the praises of a pa...

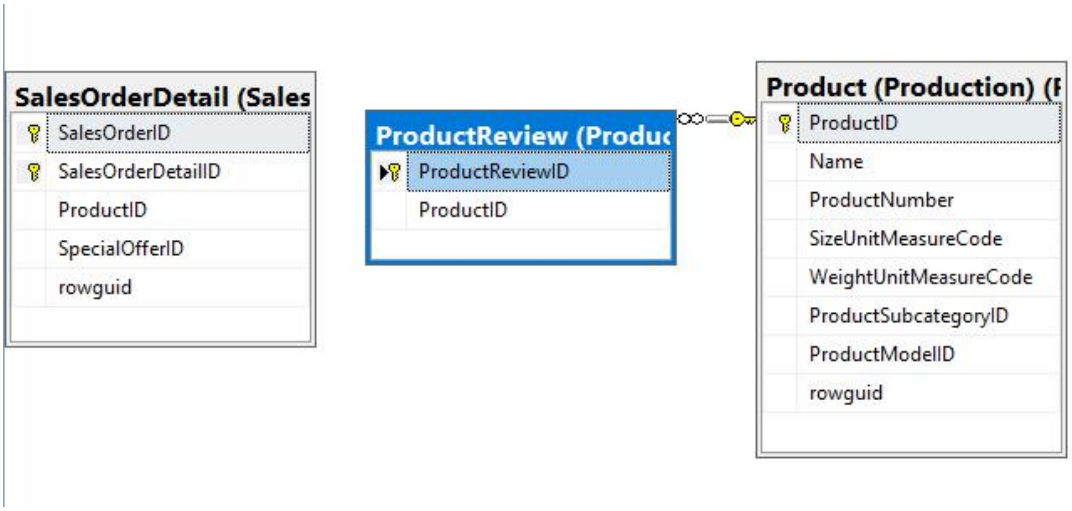
Sample output with JSON:



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:

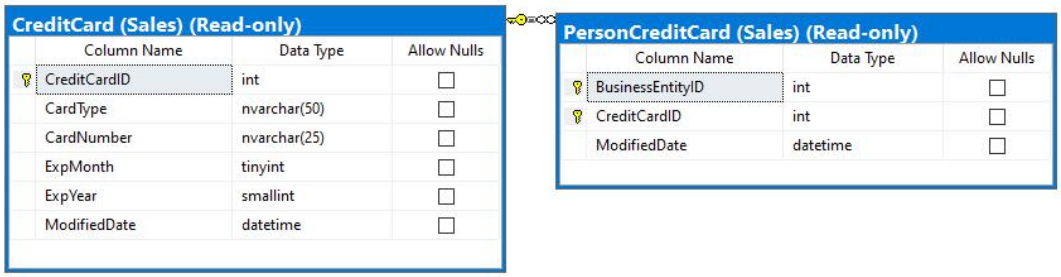
CreditCard	CardNumber	ASC
PersonCreditCard	BusinessEntityId	ASC

Sample output without JSON:

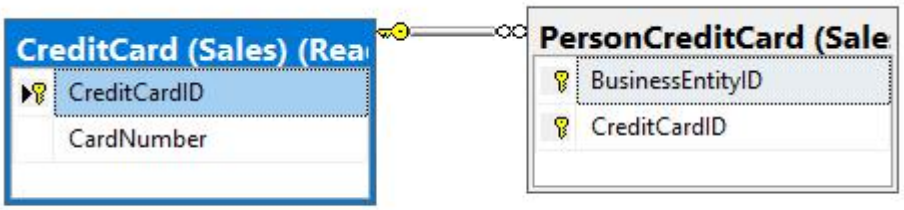
	Min card	CardNumber	BusinessEntityID
1	18100	77776101760732	333
2	18154	55553726568239	351
3	18257	33333080494431	353
4	18570	33334709876564	383
5	18295	55556966170951	421
6	18017	11119277394350	487
7	19204	33337125636756	551
8	18868	11118576036422	567
9	18717	33339764311137	589
10	18460	55553612948274	643
11	19154	55551043988788	671
12	18328	11119800451748	695

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,

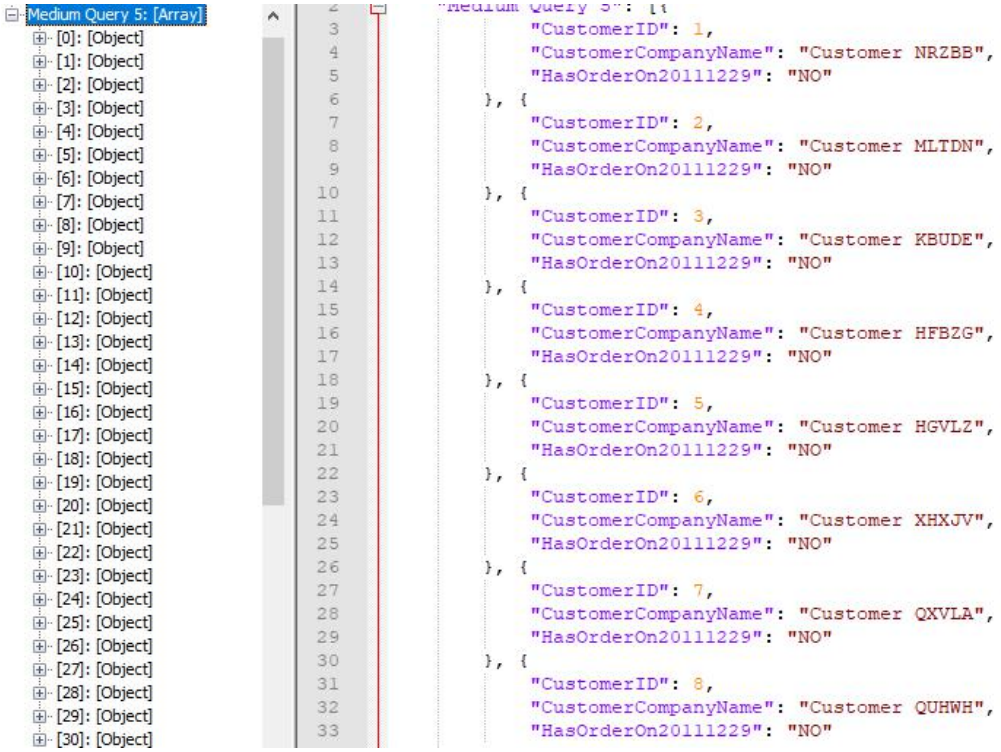
Sort by:

Orders	Orderdate	ASC

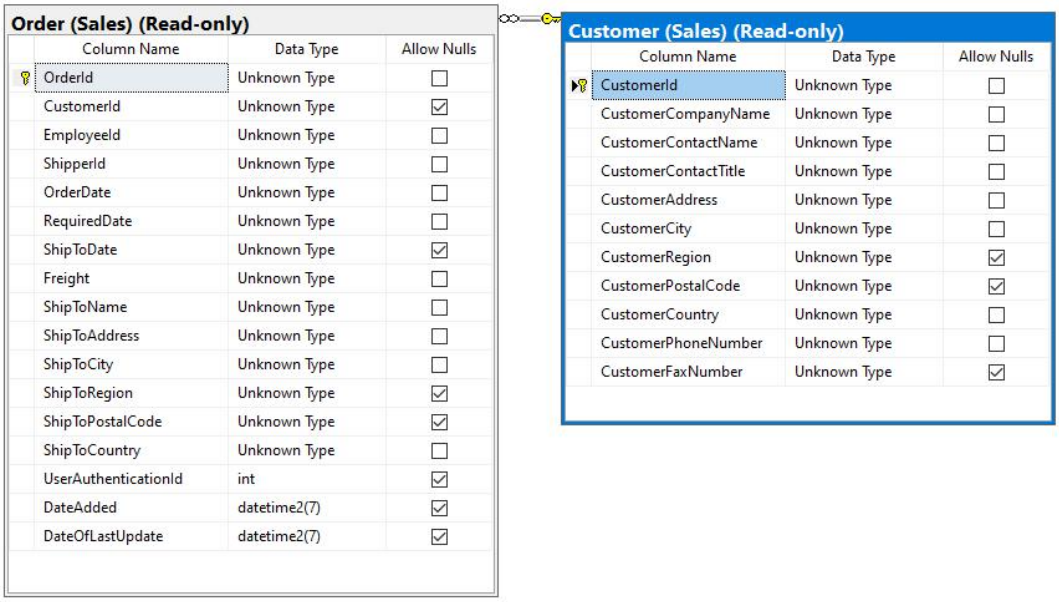
Sample output without JSON:

	CustomerID	CustomerCompanyName	HasOrderOn20111229
1	1	Customer NRZBB	NO
2	2	Customer MLTDN	NO
3	3	Customer KBUDE	NO
4	4	Customer HFBZG	NO
5	5	Customer HGVLZ	NO
6	6	Customer XHXJV	NO
7	7	Customer QXVLA	NO
8	8	Customer QUHWH	NO
9	9	Customer RTXGC	NO
10	10	Customer EEALV	NO
11	11	Customer UBHAU	NO

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

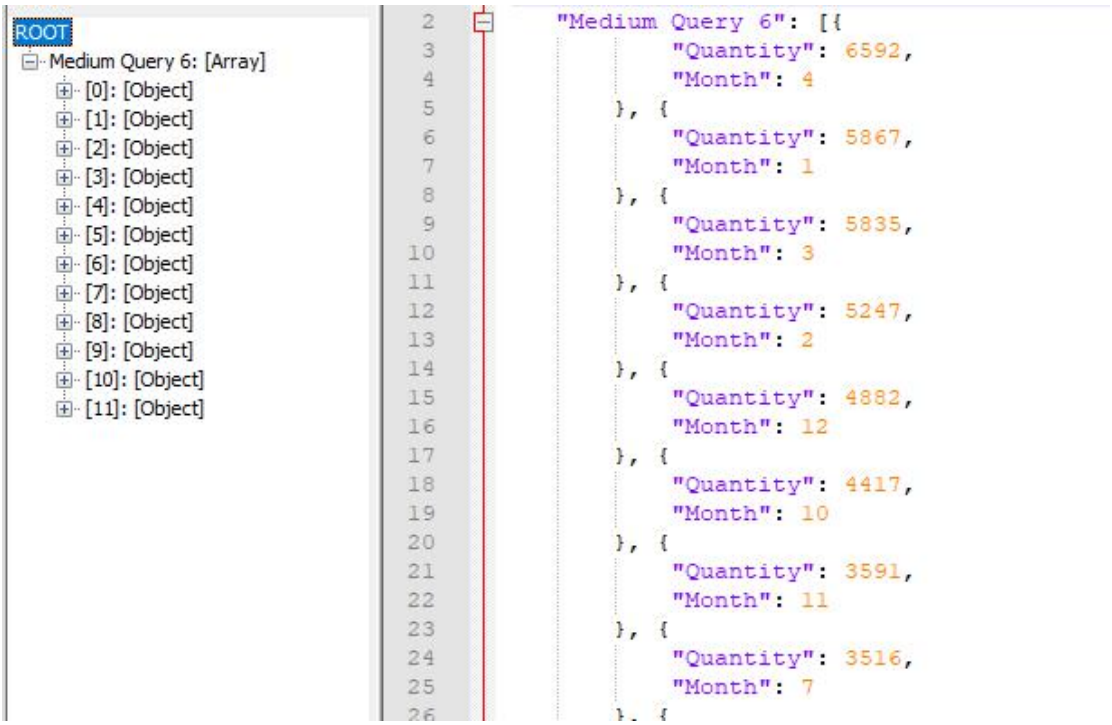
Sort by:

OrderDetail	Quantity	ASC

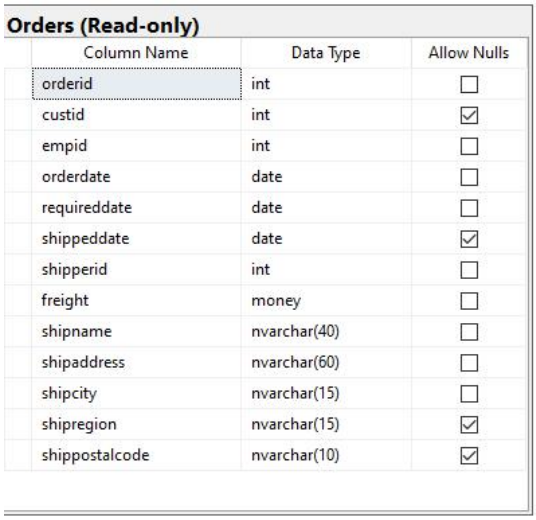
Sample output without JSON:

	Quantity	Month
1	6592	4
2	5867	1
3	5835	3
4	5247	2
5	4882	12
6	4417	10
7	3591	11
8	3516	7
9	3467	9
10	3183	8
11	3085	5
12	1635	6

Sample output with JSON:

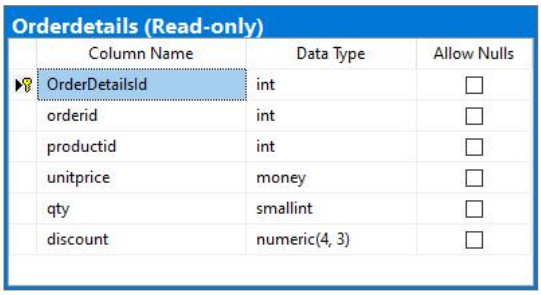


Standard View:



Orders (Read-only)

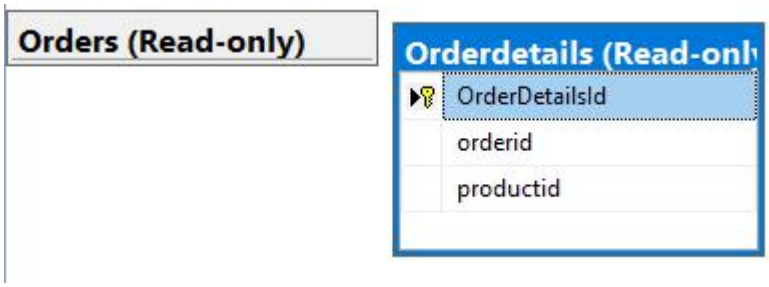
Column Name	Data Type	Allow Nulls
orderid	int	☐
custid	int	☑
empid	int	☐
orderdate	date	☐
requireddate	date	☐
shippeddate	date	☑
shipperid	int	☐
freight	money	☐
shipname	nvarchar(40)	☐
shipaddress	nvarchar(60)	☐
shipcity	nvarchar(15)	☐
shipregion	nvarchar(15)	☑
shippostalcode	nvarchar(10)	☑



Orderdetails (Read-only)

Column Name	Data Type	Allow Nulls
OrderDetailsId	int	☐
orderid	int	☐
productid	int	☐
unitprice	money	☐
qty	smallint	☐
discount	numeric(4, 3)	☐

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:

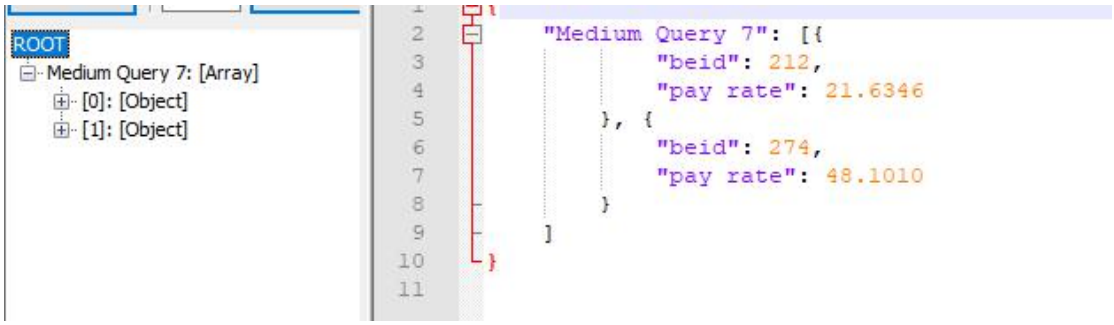
JobCandidate	businessEntityId,rate	ASC

Sample output without JSON:

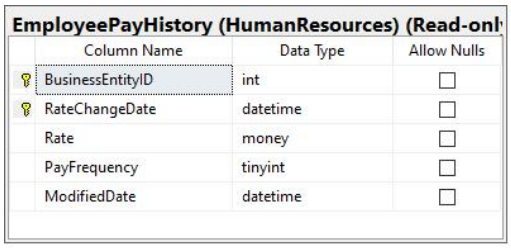
Results Messages

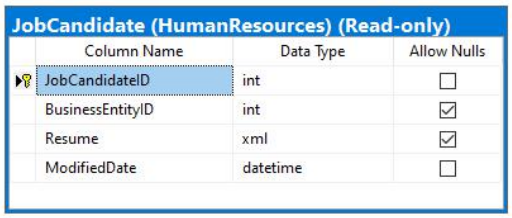
	beid	pay rate
1	212	21.6346
2	274	48.101

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
```

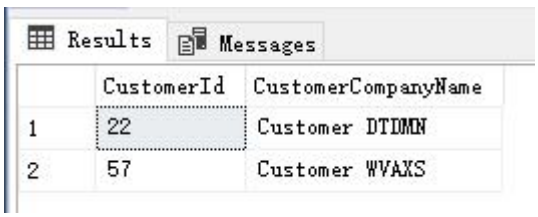
WITH JSON:

```
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
```

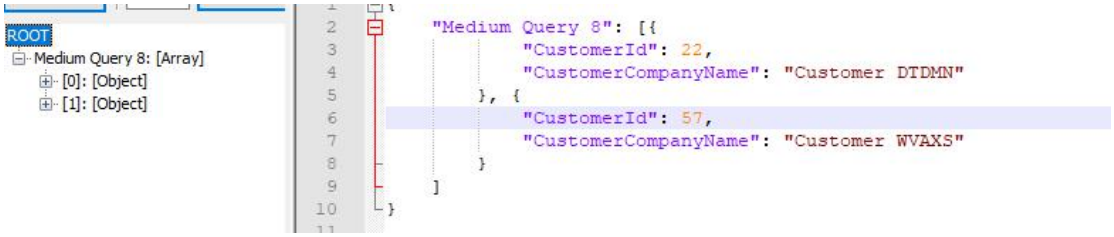
Table used :

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

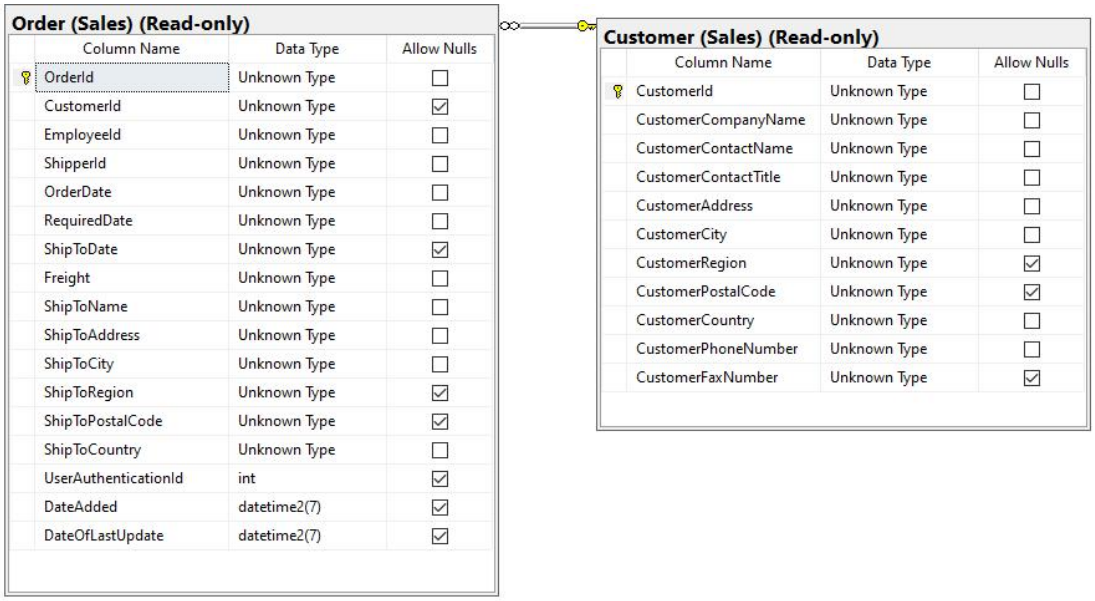
Sample output without JSON:



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

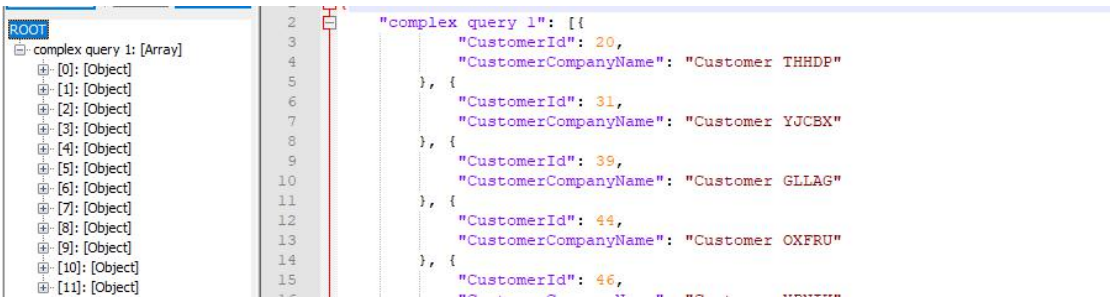
Sort By:

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

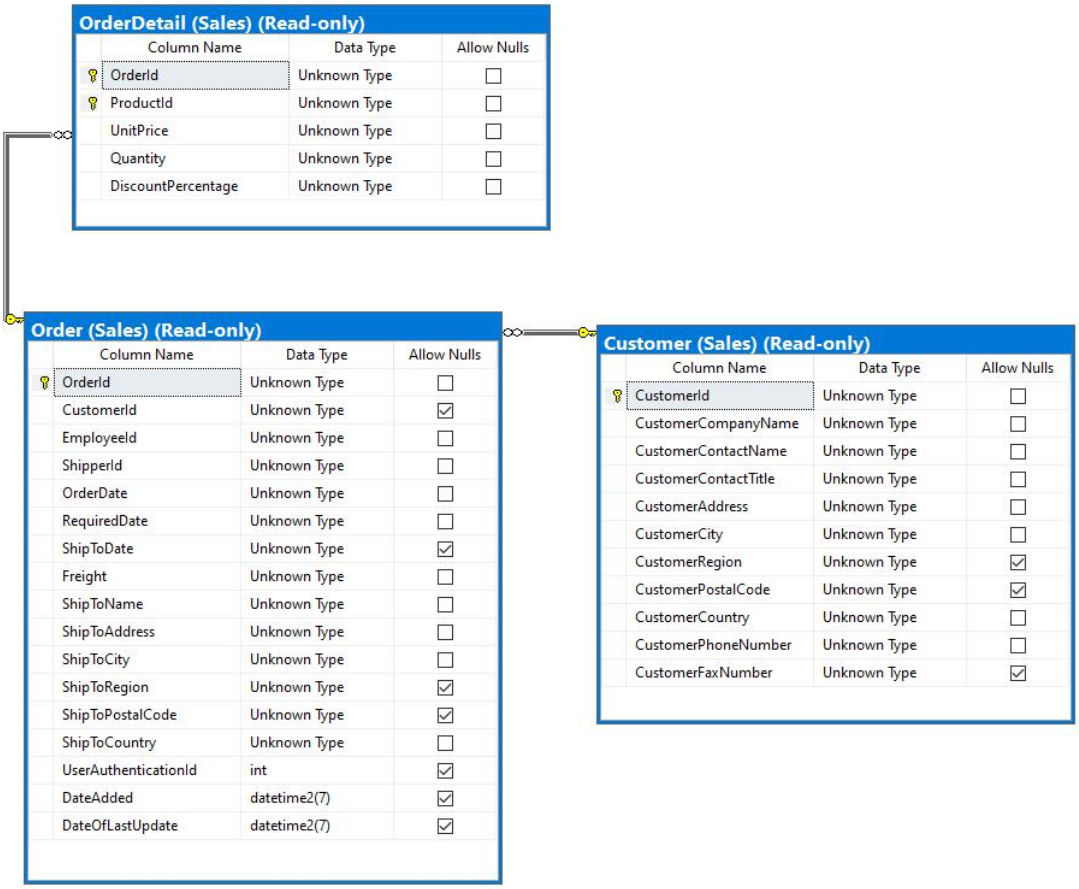
Sample output without JSON:

	CustomerId	CustomerCompanyName
1	20	Customer THHDP
2	31	Customer YJCBX
3	39	Customer GLLAG
4	44	Customer OXFRU
5	46	Customer XPNIK
6	48	Customer DVFMB
7	51	Customer PVDZC
8	65	Customer NYUHS
9	71	Customer LCOUJ
10	86	Customer SNXOJ
11	87	Customer ZHYOS
12	90	Customer XBBVR

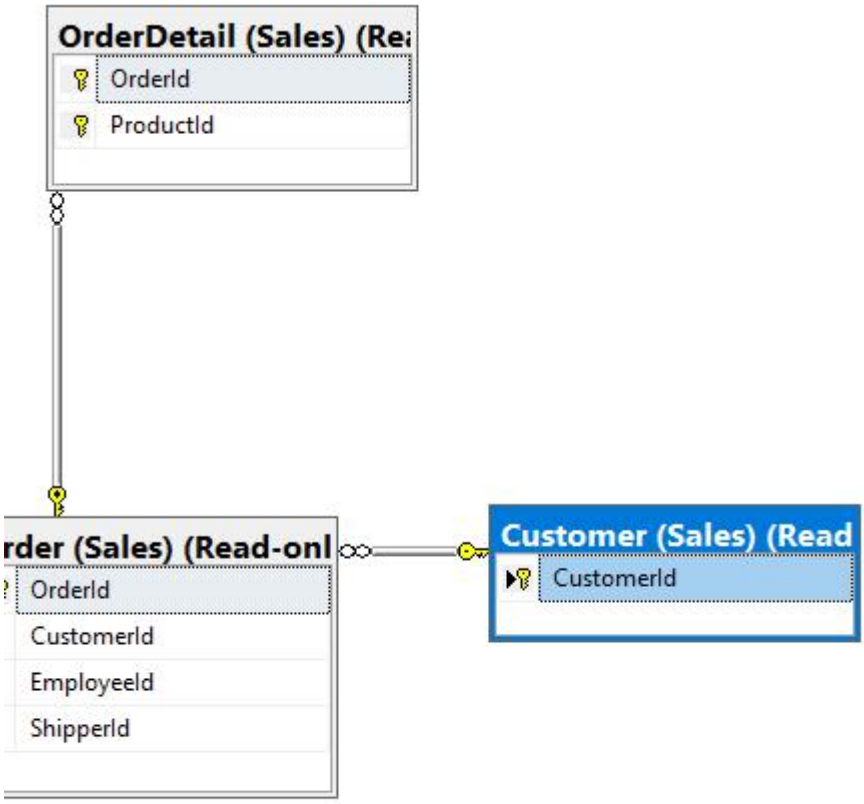
Sample output with JSON:



Standard View:


OrderDetail (Sales) (Read-only)
Column Name
Data Type
Allow Nulls
OrderId
Unknown Type
ProductId
Unknown Type
UnitPrice
Unknown Type
Quantity
Unknown Type
DiscountPercentage
Unknown Type
Order (Sales) (Read-only)
Column Name
Data Type
Allow Nulls
OrderId
Unknown Type
CustomerId
Unknown Type
EmployeeId
Unknown Type
ShipperId
Unknown Type
OrderDate
Unknown Type
RequiredDate
Unknown Type
ShipToDate
Unknown Type
Freight
Unknown Type
ShipToName
Unknown Type
ShipToAddress
Unknown Type
ShipToCity
Unknown Type
ShipToRegion
Unknown Type
ShipToPostalCode
Unknown Type
ShipToCountry
Unknown Type
UserAuthenticationId
int
DateAdded
datetime2(7)
DateOfLastUpdate
datetime2(7)
Customer (Sales) (Read-only)
Column Name
Data Type
Allow Nulls
CustomerId
Unknown Type
CustomerCompanyName
Unknown Type
CustomerContactName
Unknown Type
CustomerContactTitle
Unknown Type
CustomerAddress
Unknown Type
CustomerCity
Unknown Type
CustomerRegion
Unknown Type
CustomerPostalCode
Unknown Type
CustomerCountry
Unknown Type
CustomerPhoneNumber
Unknown Type
CustomerFaxNumber
Unknown Type

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

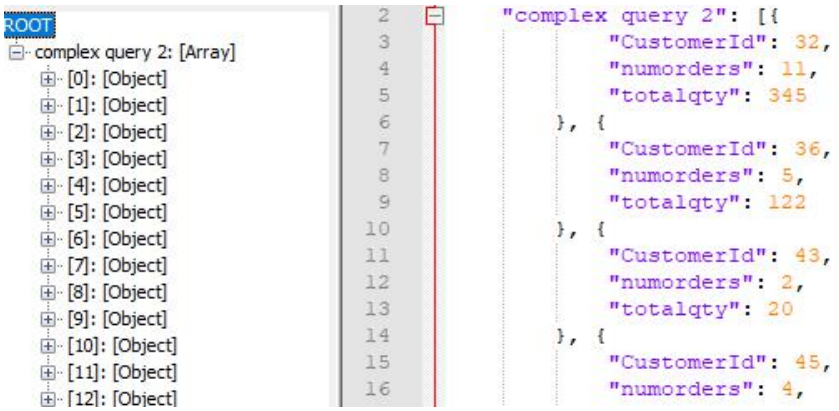
Order By

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

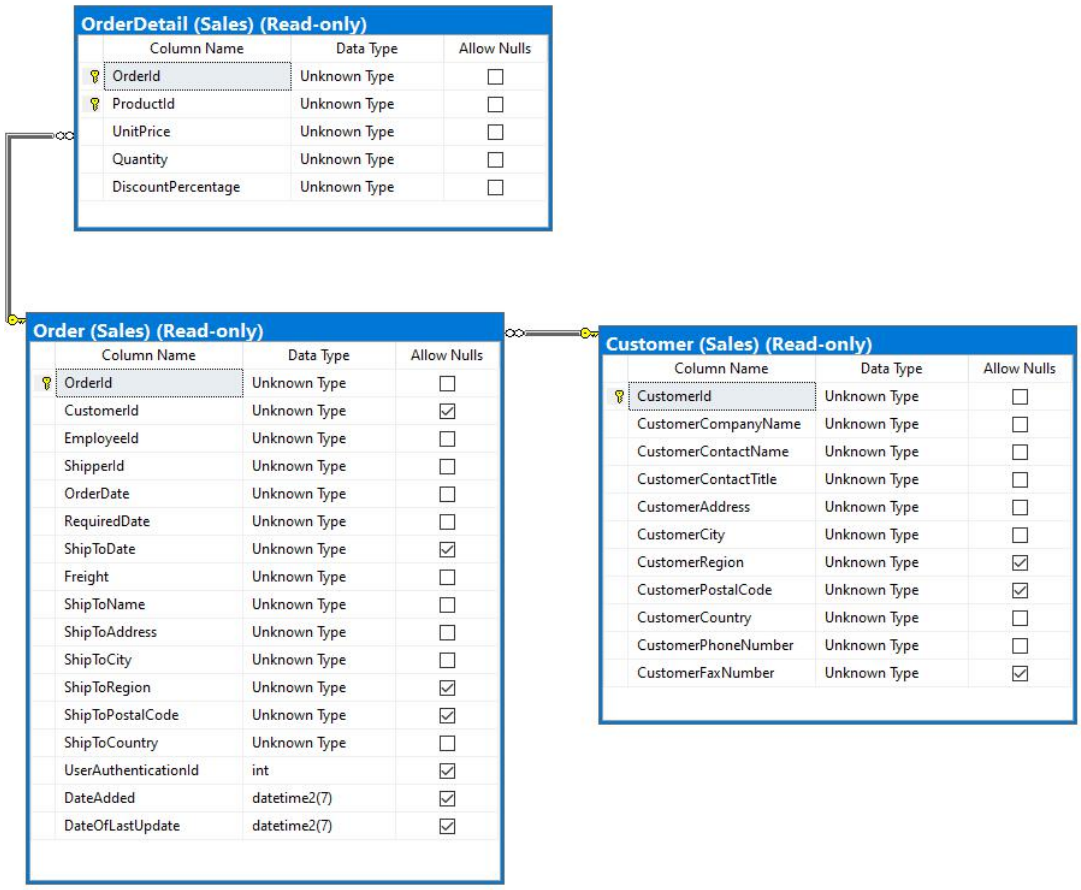
Sample output without JSON:

	CustomerId	numorders	totalqty
1	32	11	345
2	36	5	122
3	43	2	20
4	45	4	181
5	48	8	134
6	55	10	603
7	65	18	1383
8	71	31	4958
9	75	9	327
10	77	4	46

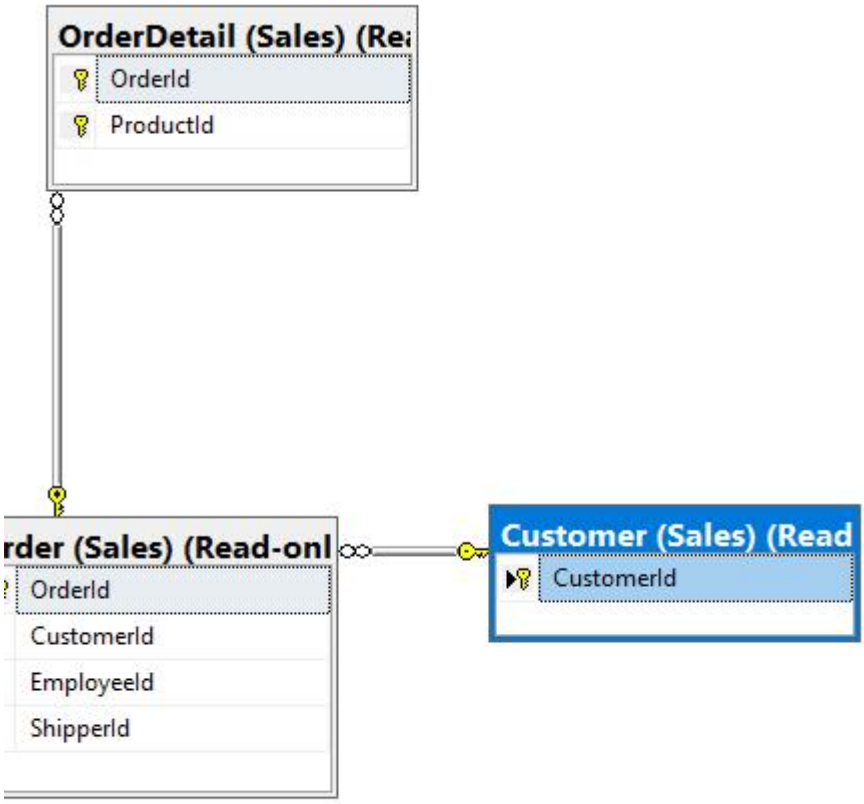
Sample output with JSON:



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:

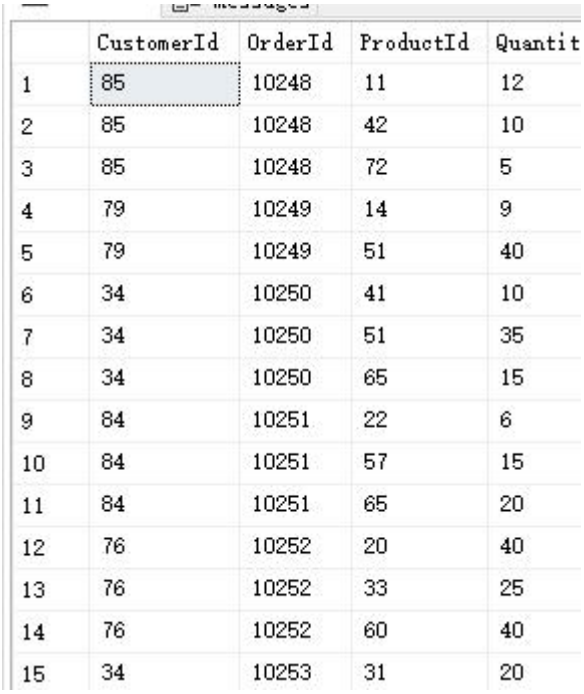
```
use Northwinds2020TSQLV6;
SELECT C.CustomerId
     ,O.OrderId
     ,OD.ProductId
     ,OD.Quantity
```

```
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

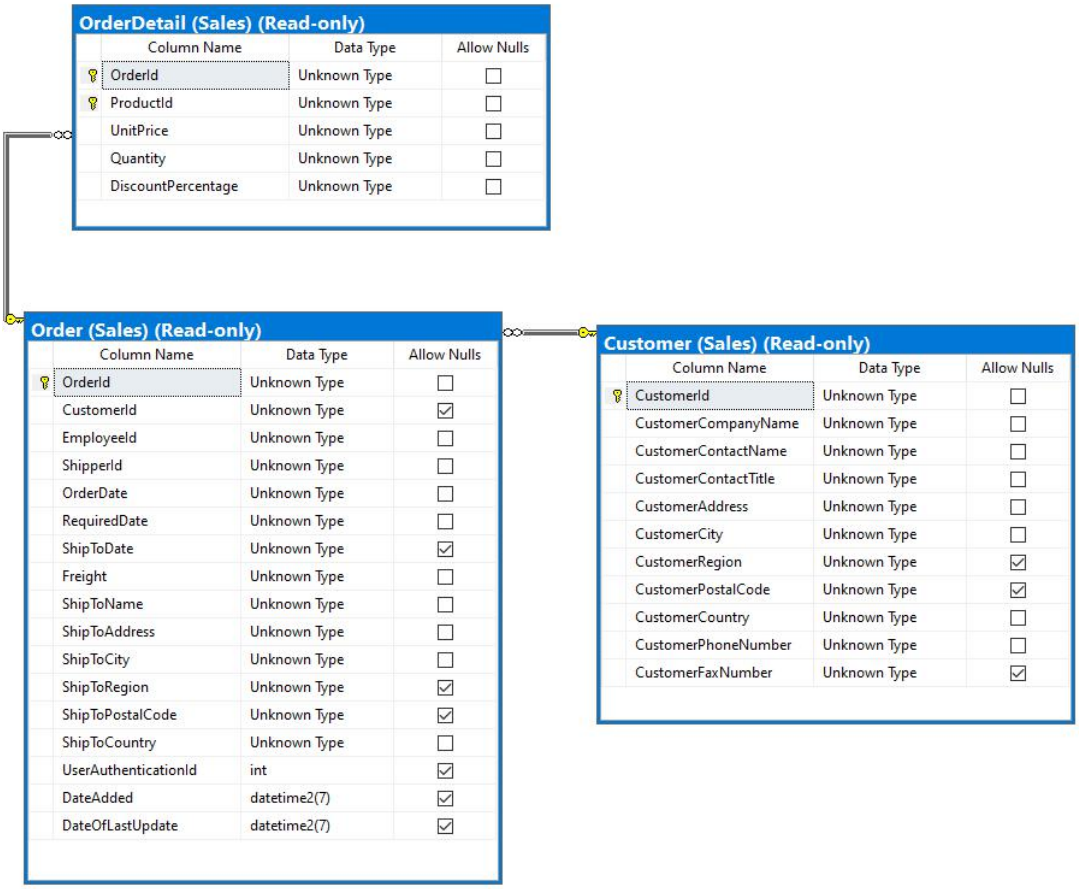
Sample output without JSON:



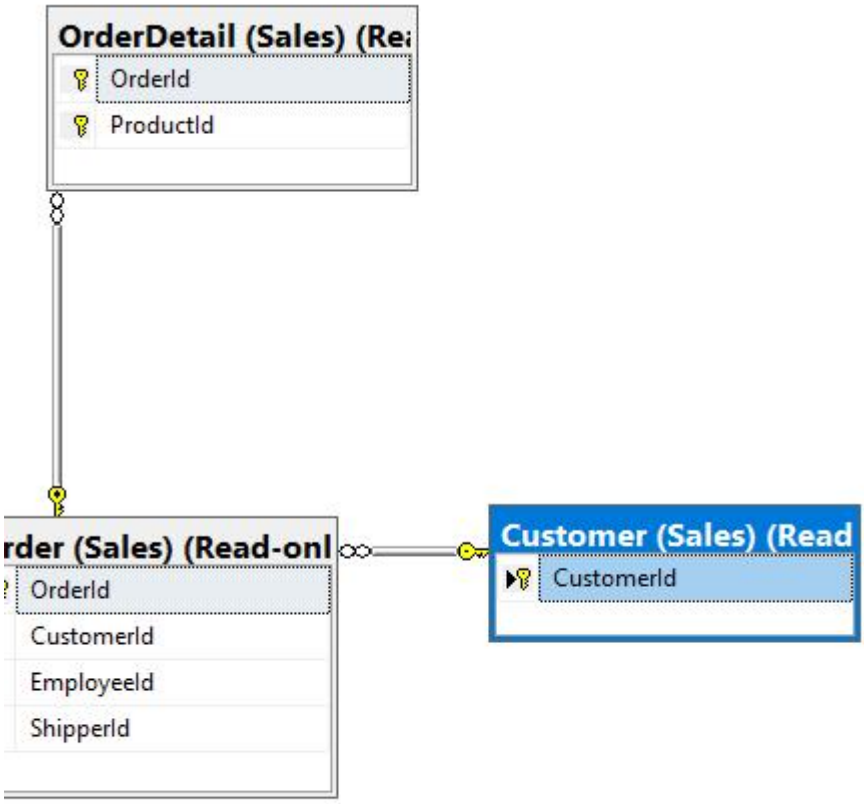
	CustomerId	OrderId	ProductId	Quantit
1	85	10248	11	12
2	85	10248	42	10
3	85	10248	72	5
4	79	10249	14	9
5	79	10249	51	40
6	34	10250	41	10
7	34	10250	51	35
8	34	10250	65	15
9	84	10251	22	6
10	84	10251	57	15
11	84	10251	65	20
12	76	10252	20	40
13	76	10252	33	25
14	76	10252	60	40
15	34	10253	31	20

Sample output with JSON:

Standard View:


OrderDetail (Sales) (Read-only)
Column Name
Data Type
Allow Nulls
OrderId
Unknown Type
ProductId
Unknown Type
UnitPrice
Unknown Type
Quantity
Unknown Type
DiscountPercentage
Unknown Type
Order (Sales) (Read-only)
Column Name
Data Type
Allow Nulls
OrderId
Unknown Type
CustomerId
Unknown Type
EmployeeId
Unknown Type
ShipperId
Unknown Type
OrderDate
Unknown Type
RequiredDate
Unknown Type
ShipToDate
Unknown Type
Freight
Unknown Type
ShipToName
Unknown Type
ShipToAddress
Unknown Type
ShipToCity
Unknown Type
ShipToRegion
Unknown Type
ShipToPostalCode
Unknown Type
ShipToCountry
Unknown Type
UserAuthenticationId
int
DateAdded
datetime2(7)
DateOfLastUpdate
datetime2(7)
Customer (Sales) (Read-only)
Column Name
Data Type
Allow Nulls
CustomerId
Unknown Type
CustomerCompanyName
Unknown Type
CustomerContactName
Unknown Type
CustomerContactTitle
Unknown Type
CustomerAddress
Unknown Type
CustomerCity
Unknown Type
CustomerRegion
Unknown Type
CustomerPostalCode
Unknown Type
CustomerCountry
Unknown Type
CustomerPhoneNumber
Unknown Type
CustomerFaxNumber
Unknown Type

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:

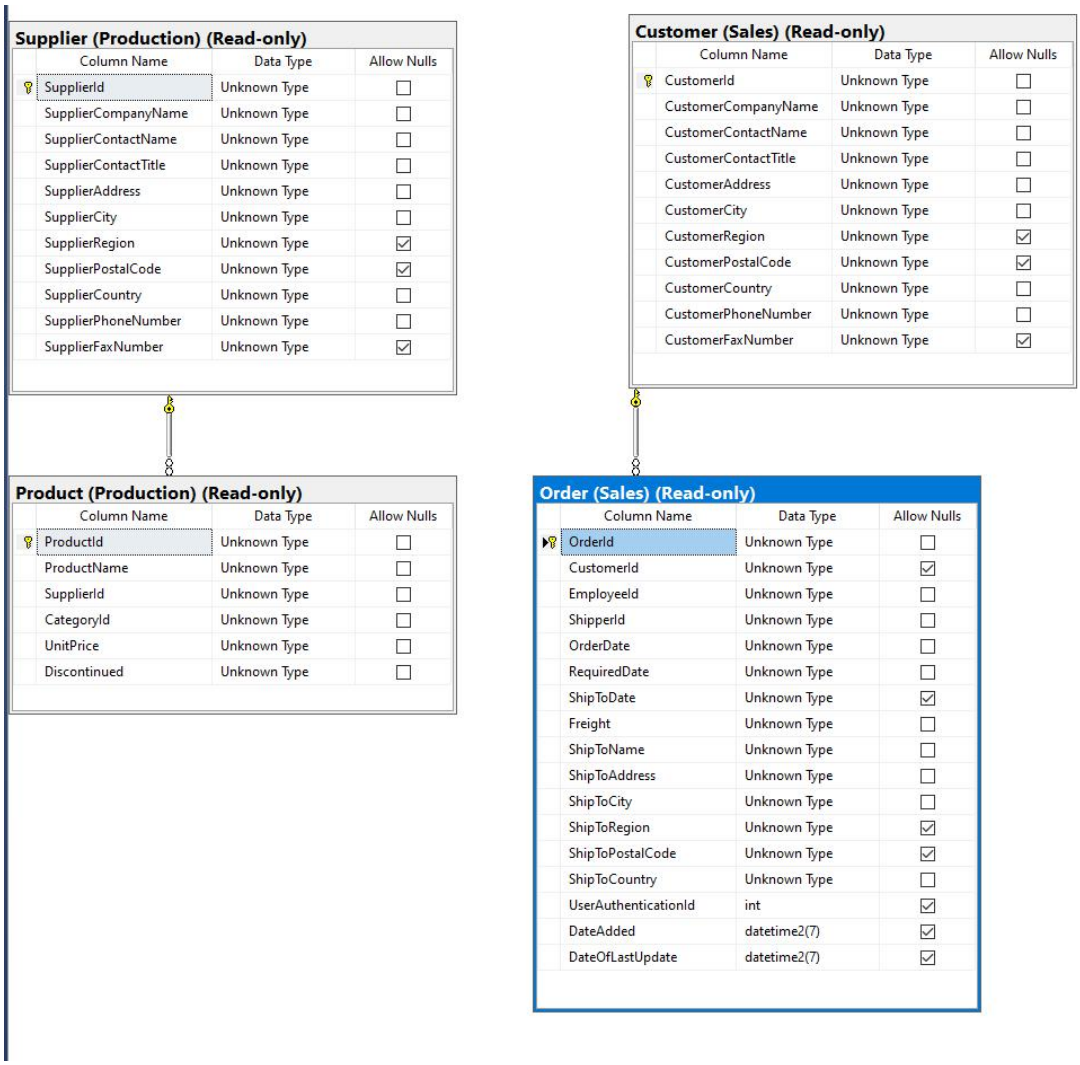
Table name	Column name	Sort order
Order	Orderdate	ASC

Sample output without JSON:

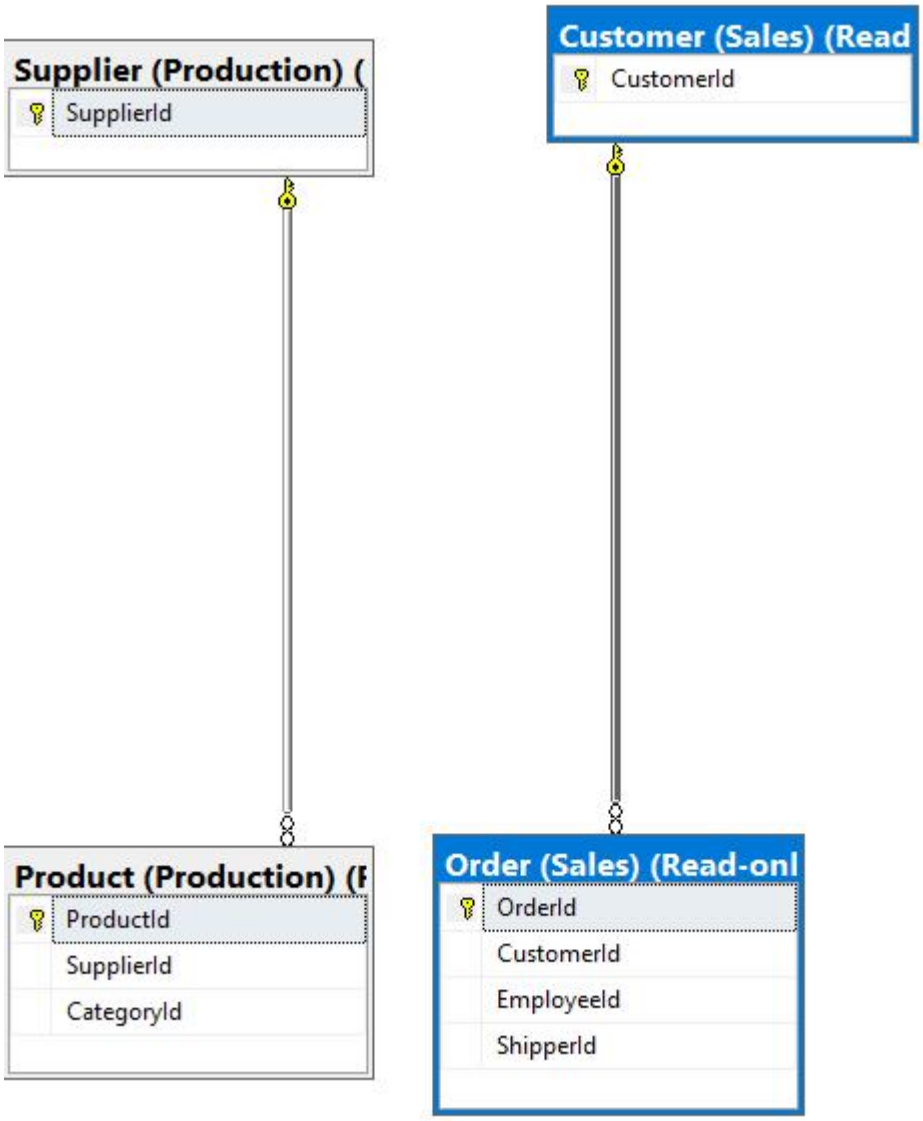
	SupplierId	SupplierCompanyName	LOCATIONE	ProductId	ProductName	OrderId	OrderDate	CustomerCompanyName	LOCATION
1	2	Supplier VHQZD	New Orleans, USA	4	Product KSBRM	11074	2016-05-06	Customer JMIKW	Kobenhavn, Denmark
2	2	Supplier VHQZD	New Orleans, USA	5	Product EPEIM	11074	2016-05-06	Customer JMIKW	Kobenhavn, Denmark
3	2	Supplier VHQZD	New Orleans, USA	65	Product XYWBZ	11074	2016-05-06	Customer JMIKW	Kobenhavn, Denmark
4	2	Supplier VHQZD	New Orleans, USA	66	Product LQMGN	11074	2016-05-06	Customer JMIKW	Kobenhavn, Denmark
5	2	Supplier VHQZD	New Orleans, USA	4	Product KSBRM	11075	2016-05-06	Customer CCKOT	Genève, Switzerland
6	2	Supplier VHQZD	New Orleans, USA	5	Product EPEIM	11075	2016-05-06	Customer CCKOT	Genève, Switzerland
7	2	Supplier VHQZD	New Orleans, USA	65	Product XYWBZ	11075	2016-05-06	Customer CCKOT	Genève, Switzerland
8	2	Supplier VHQZD	New Orleans, USA	66	Product LQMGN	11075	2016-05-06	Customer CCKOT	Genève, Switzerland
9	2	Supplier VHQZD	New Orleans, USA	4	Product KSBRM	11076	2016-05-06	Customer RTXGC	Marseille, France
10	2	Supplier VHQZD	New Orleans, USA	5	Product EPEIM	11076	2016-05-06	Customer RTXGC	Marseille, France
11	2	Supplier VHQZD	New Orleans, USA	65	Product XYWBZ	11076	2016-05-06	Customer RTXGC	Marseille, France
12	2	Supplier VHQZD	New Orleans, USA	66	Product LQMGN	11076	2016-05-06	Customer RTXGC	Marseille, France

Sample output with JSON:

Standard View:


Supplier (Production) (Read-only)
Column Name
Data Type
Allow Nulls
SupplierId
Unknown Type
SupplierCompanyName
SupplierContactName
SupplierContactTitle
SupplierAddress
SupplierCity
SupplierRegion
SupplierPostalCode
SupplierCountry
SupplierPhoneNumber
SupplierFaxNumber
Customer (Sales) (Read-only)
CustomerId
CustomerCompanyName
CustomerContactName
CustomerContactTitle
CustomerAddress
CustomerCity
CustomerRegion
CustomerPostalCode
CustomerCountry
CustomerPhoneNumber
CustomerFaxNumber
Product (Production) (Read-only)
ProductId
ProductName
SupplierId
CategoryId
UnitPrice
Discontinued
Order (Sales) (Read-only)
OrderId
CustomerId
EmployeeId
ShipperId
OrderDate
RequiredDate
ShipToDate
Freight
ShipToName
ShipToAddress
ShipToCity
ShipToRegion
ShipToPostalCode
ShipToCountry
UserAuthenticationId
int
DateAdded
datetime2(7)
DateOfLastUpdate

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
```

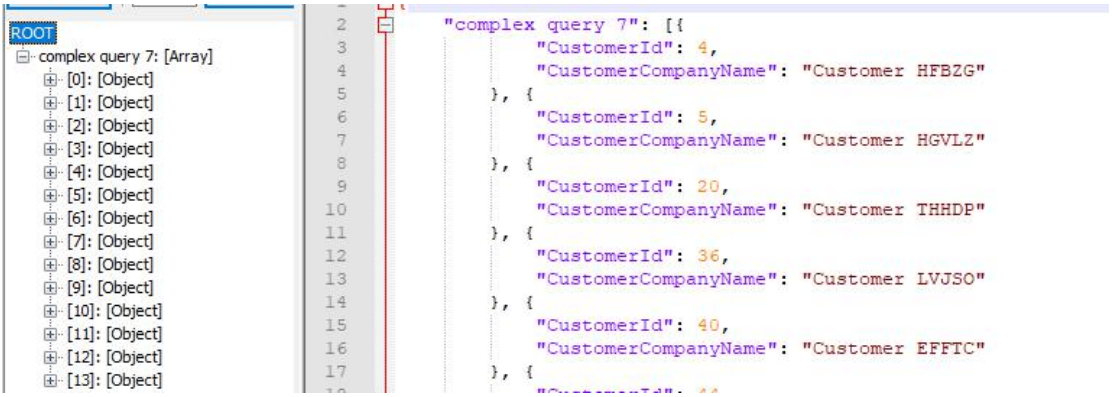
Table:

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

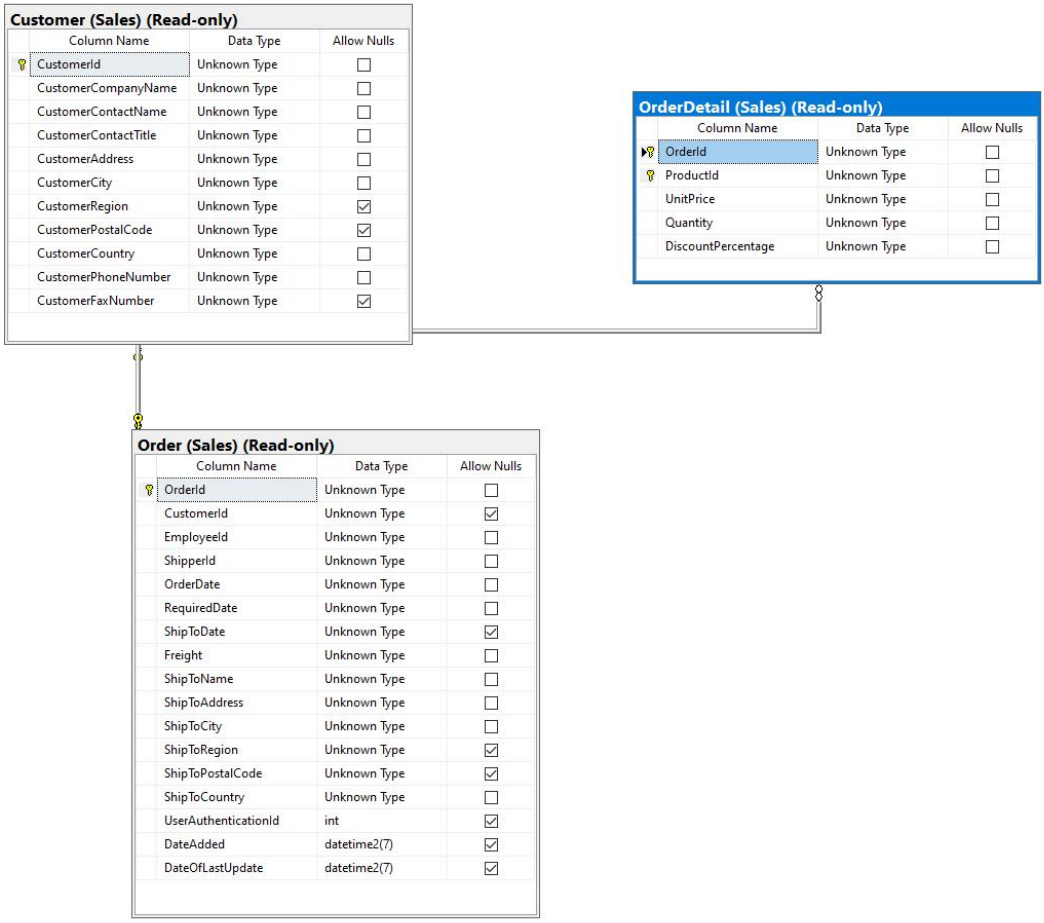
Sample output without JSON:

	CustomerId	CustomerCompanyName
1	4	Customer HFBZG
2	5	Customer HGVLZ
3	20	Customer THHDP
4	36	Customer LVJSO
5	40	Customer EFFTC
6	44	Customer OXFRU
7	54	Customer TDKEG
8	62	Customer WFIZJ
9	63	Customer IRRVL
10	64	Customer LWGMD
11	65	Customer NYUHS
12	72	Customer AHPOP
13	76	Customer SFOGW
14	81	Customer YQQWW

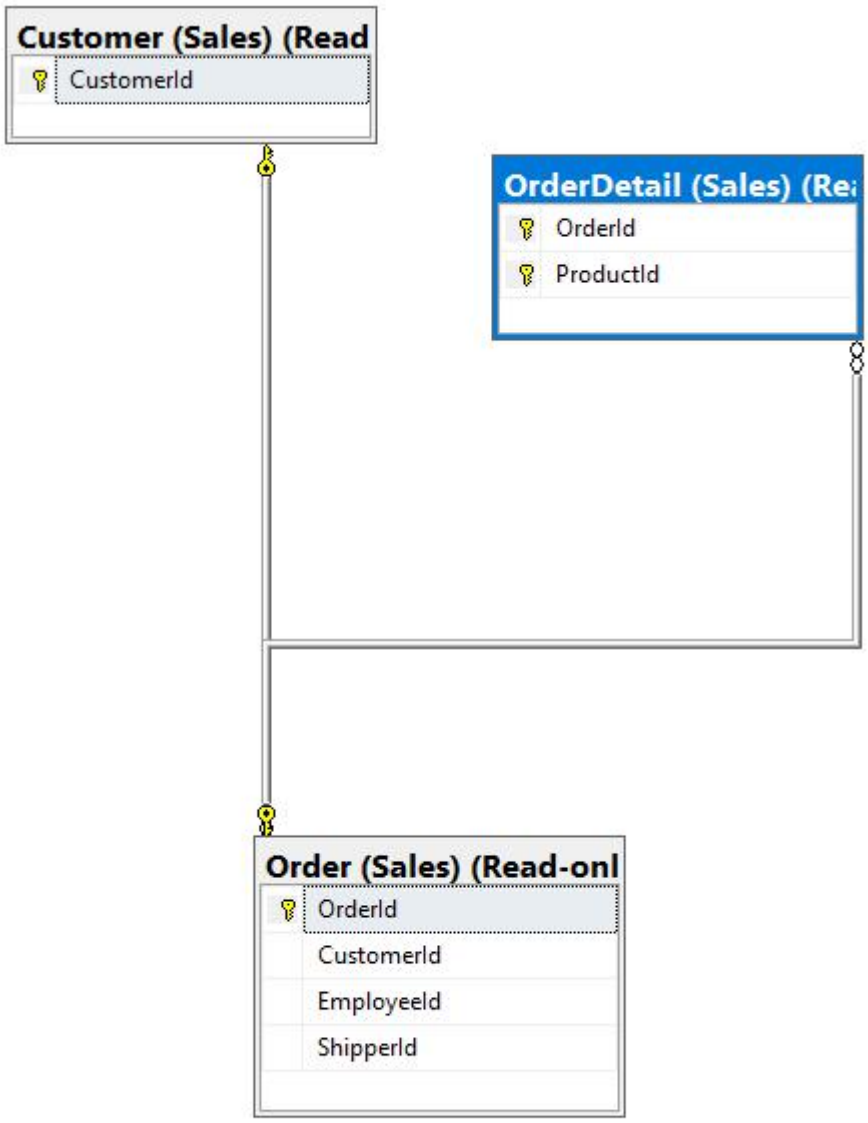
Sample output with JSON:



Standard View:


Customer (Sales) (Read-only)
Column Name
Data Type
Allow Nulls
CustomerId
Unknown Type
CustomerCompanyName
Unknown Type
CustomerContactName
Unknown Type
CustomerContactTitle
Unknown Type
CustomerAddress
Unknown Type
CustomerCity
Unknown Type
CustomerRegion
Unknown Type
CustomerPostalCode
Unknown Type
CustomerCountry
Unknown Type
CustomerPhoneNumber
Unknown Type
CustomerFaxNumber
Unknown Type
OrderDetail (Sales) (Read-only)
Column Name
Data Type
Allow Nulls
OrderId
Unknown Type
ProductId
Unknown Type
UnitPrice
Unknown Type
Quantity
Unknown Type
DiscountPercentage
Unknown Type
Order (Sales) (Read-only)
Column Name
Data Type
Allow Nulls
OrderId
Unknown Type
CustomerId
Unknown Type
EmployeeId
Unknown Type
ShipperId
Unknown Type
OrderDate
Unknown Type
RequiredDate
Unknown Type
ShipToDate
Unknown Type
Freight
Unknown Type
ShipToName
Unknown Type
ShipToAddress
Unknown Type
ShipToCity
Unknown Type
ShipToRegion
Unknown Type
ShipToPostalCode
Unknown Type
ShipToCountry
Unknown Type
UserAuthenticationId
int
DateAdded
datetime2(7)
DateOfLastUpdate
datetime2(7)

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:

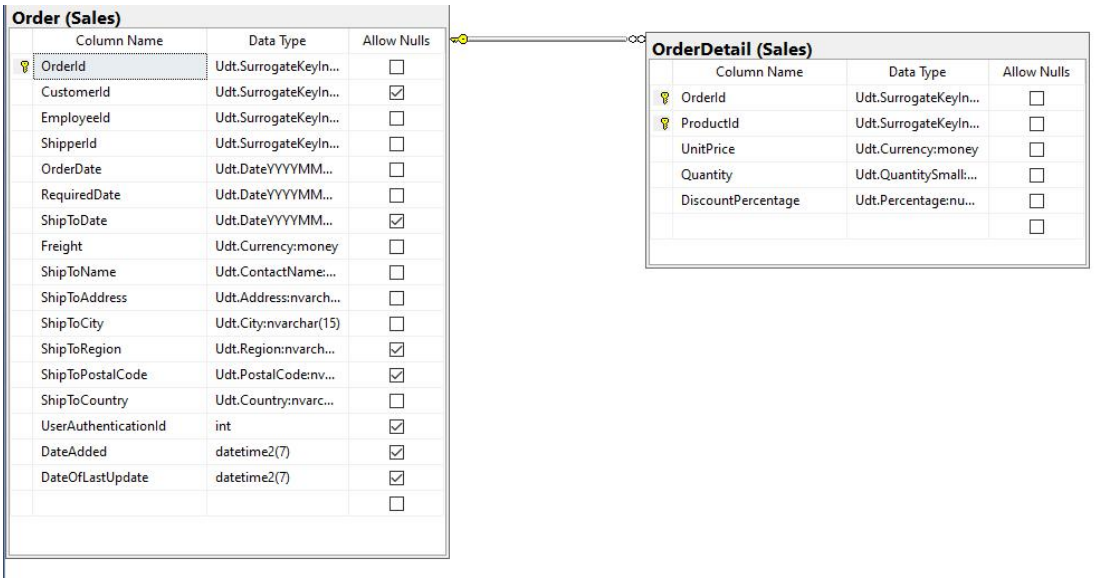
Table name	Column Name	SORT ORDER
orderDetail	orderId	ASC

Sample output without JSON:

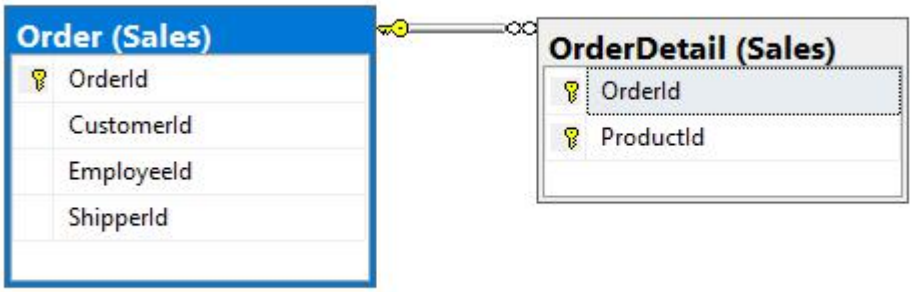
	OrderId	net_amount
1	10865	16387.50
2	10981	15810.00
3	11030	12615.05
4	10889	11380.00
5	10417	11188.40
6	10817	10952.85
7	10897	10835.24
8	10479	10495.60
9	10540	10191.70
10	10691	10164.80
11	10515	9921.30
12	10372	9210.90
13	10424	9194.56

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

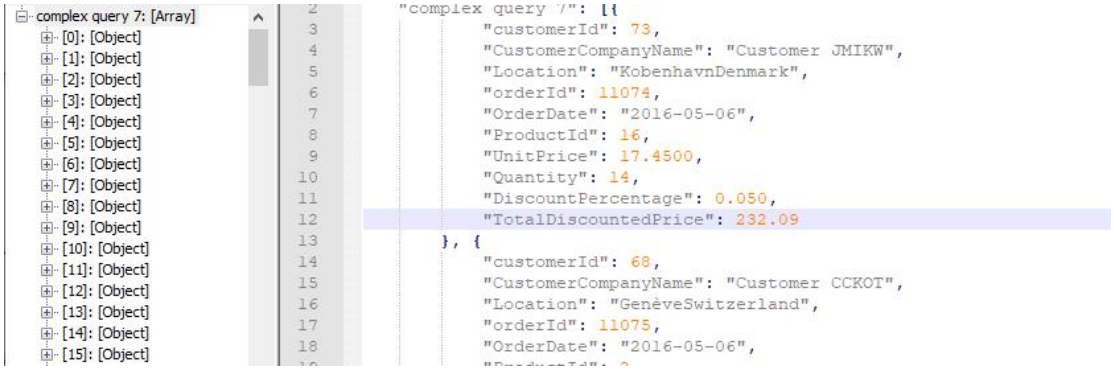
Order By:

Table name	Column name	Sort order
Order	Orderdate	ASC

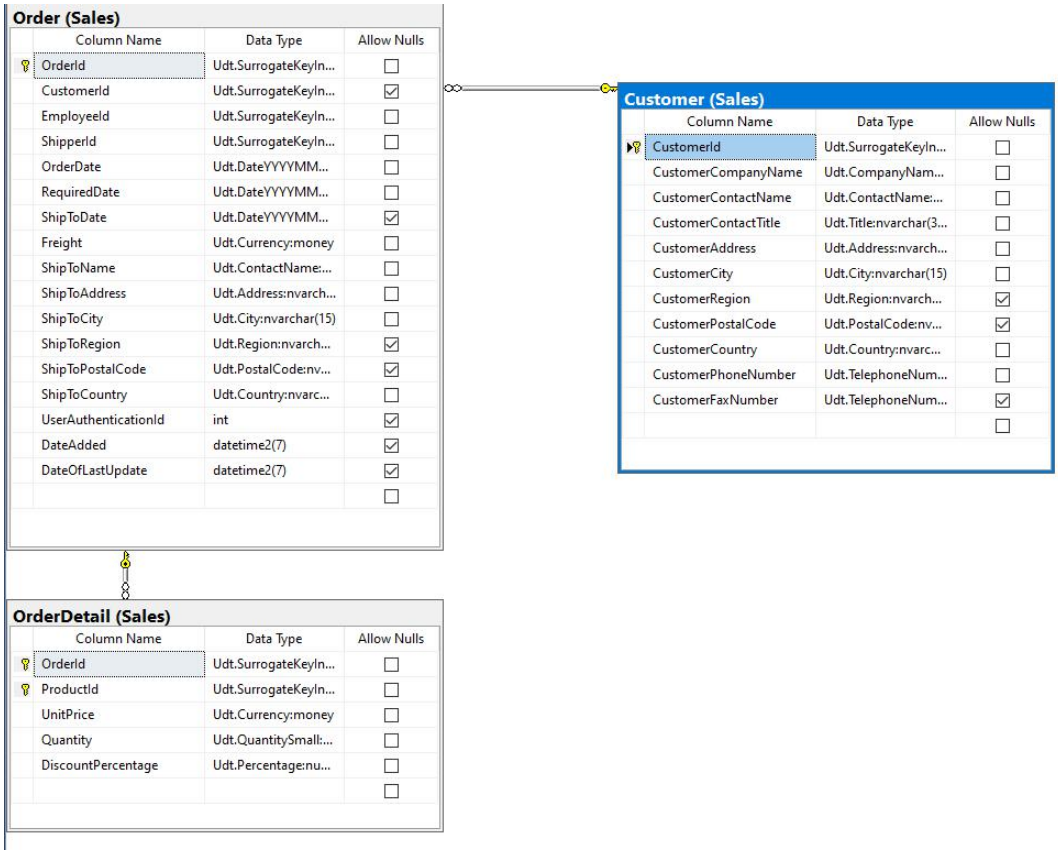
Sample output without JSON:

	customerId	CustomerCompanyName	Location	orderId	OrderDate	ProductId	UnitPrice	Quantity	DiscountPercentage	TotalDiscountedPrice
1	73	Customer JMIKW	KobenhavnDenmark	11074	2016-05-06	16	17.45	14	0.050	232.09
2	68	Customer CCKOT	GenèveSwitzerland	11075	2016-05-06	2	19.00	10	0.150	161.50
3	68	Customer CCKOT	GenèveSwitzerland	11075	2016-05-06	46	12.00	30	0.150	306.00
4	68	Customer CCKOT	GenèveSwitzerland	11075	2016-05-06	76	18.00	2	0.150	30.60
5	9	Customer RTXGC	MarseilleFrance	11076	2016-05-06	6	25.00	20	0.250	375.00
6	9	Customer RTXGC	MarseilleFrance	11076	2016-05-06	14	23.25	20	0.250	348.75
7	9	Customer RTXGC	MarseilleFrance	11076	2016-05-06	19	9.20	10	0.250	69.00
8	65	Customer NYUHS	AlbuquerqueUSA	11077	2016-05-06	2	19.00	24	0.200	364.80
9	65	Customer NYUHS	AlbuquerqueUSA	11077	2016-05-06	3	10.00	4	0.000	40.00
10	65	Customer NYUHS	AlbuquerqueUSA	11077	2016-05-06	4	22.00	1	0.000	22.00
11	65	Customer NYUHS	AlbuquerqueUSA	11077	2016-05-06	6	25.00	1	0.020	24.50
12	65	Customer NYUHS	AlbuquerqueUSA	11077	2016-05-06	7	30.00	1	0.050	28.50
13	65	Customer NYUHS	AlbuquerqueUSA	11077	2016-05-06	8	40.00	2	0.100	72.00
14	65	Customer NYUHS	AlbuquerqueUSA	11077	2016-05-06	10	31.00	1	0.000	31.00
15	65	Customer NYUHS	AlbuquerqueUSA	11077	2016-05-06	12	38.00	2	0.050	72.20
16	65	Customer NYUHS	AlbuquerqueUSA	11077	2016-05-06	13	6.00	4	0.000	24.00

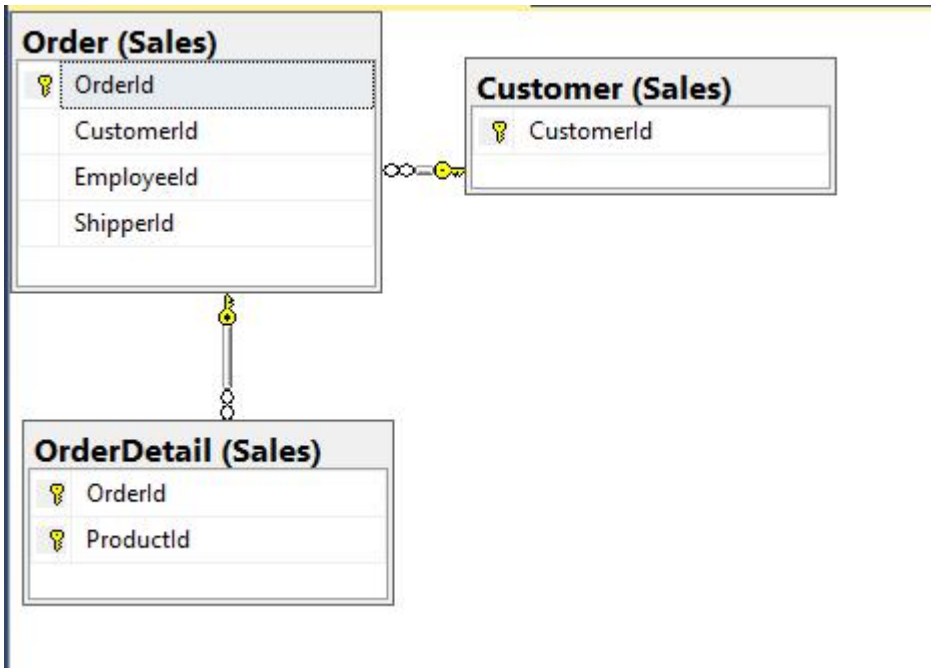
Sample output with JSON:



Standard View:



Key View:


Order (Sales)
OrderId
CustomerId
EmployeeId
ShipperId
Customer (Sales)
CustomerId
OrderDetail (Sales)
OrderId
ProductId