Fall
2020

CS331 Project 1

DATABASES - PROFESSOR HELLER - 10:45-12:00 TUES/THURS

PREPARED BY HARJIT LIYAL

Table of Contents

Preposition 1 - Simple Query 1 .....Page 2
Preposition 2 - Simple Query 2 .....Page 5
Preposition 3 - Simple Query 3 .....Page 7
Preposition 4 - Simple Query 4.....Page 10
Preposition 5 - Simple Query 5.....Page 13
Preposition 6 - Medium Query 1.....Page 16
Preposition 7 - Medium Query 2.....Page 19
Preposition 8 - Medium Query 3.....Page 22
Preposition 9 - Medium Query 4.....Page 25
Preposition 10 - Medium Query 5.....Page 28
Preposition 11 - Medium Query 6.....Page 31
Preposition 12 - Medium Query 7.....Page 34
Preposition 13 - Medium Query 8.....Page 37
Preposition 14 - Complex Query 1.....Page 40
Preposition 15 - Complex Query 2.....Page 43
Preposition 16 - Complex Query 3.....Page 47
Preposition 17 - Complex Query 4.....Page 51
Preposition 18 - Complex Query 5.....Page 55
Preposition 19 - Complex Query 6.....Page 59
Preposition 20 - Complex Query 7.....Page 63

All prepositions follow the format:

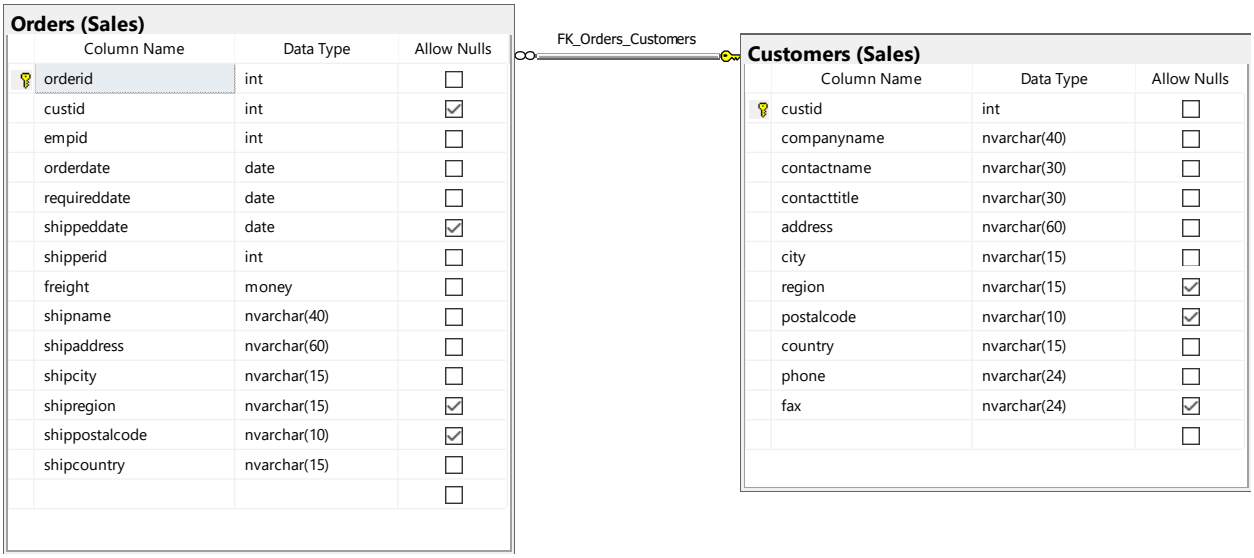
1. Preposition explaining what the query returns using given database and tables
2. Standard and key view of the tables used
3. Chart that states the table name, columns, and the order by
4. Example relational code solution with screenshot of results and number of rows returned
5. Example JSON code solution with screenshot of results and number of objects returned

Note: Follow the page numbers on the bottom of the document when navigating using Table of Contents.

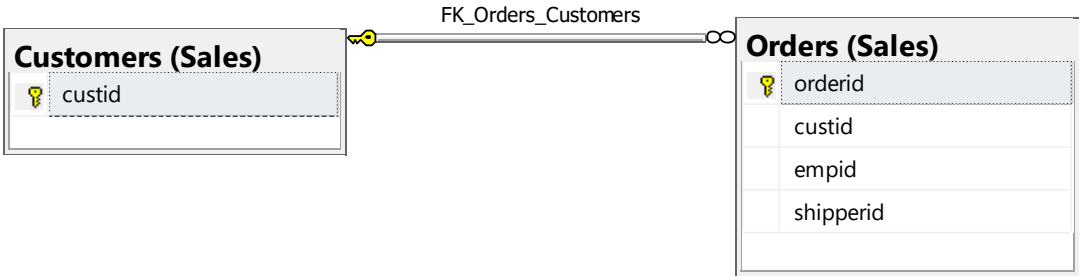
Simple Query 1

```
Return all customer ids, their country, and the date they ordered

Use TSQLV4 Database, Sales.Customer, Sales.Orders tables
```

Standard view



Key view



Columns from tables

Table Name	Column Names
Sales.Customers	Custid country
Sales.Orders	orderdate

Order By

Table Name	Column Name	Sort Order
Sales.Customers	custid	ASC
Sales.Customers	country	ASC
Sales.Orders	orderdate	ASC

Sample relational Solution Query Without JSON

```
USE TSQLV4;
SELECT E.custid,
       E.country,
       F.orderdate
FROM Sales.Customers AS E
    INNER JOIN Sales.Orders AS F
        ON F.custid = E.custid
ORDER BY E.custid
--FOR JSON PATH, ROOT ('custid'), INCLUDE_NULL_VALUES;
```

Solution Query With JSON

```
USE TSQLV4;
SELECT E.custid,
       E.country,
       F.orderdate
FROM Sales.Customers AS E
    INNER JOIN Sales.Orders AS F
        ON F.custid = E.custid
ORDER BY E.custid
FOR JSON PATH, ROOT('custid'), INCLUDE_NULL_VALUES;
```

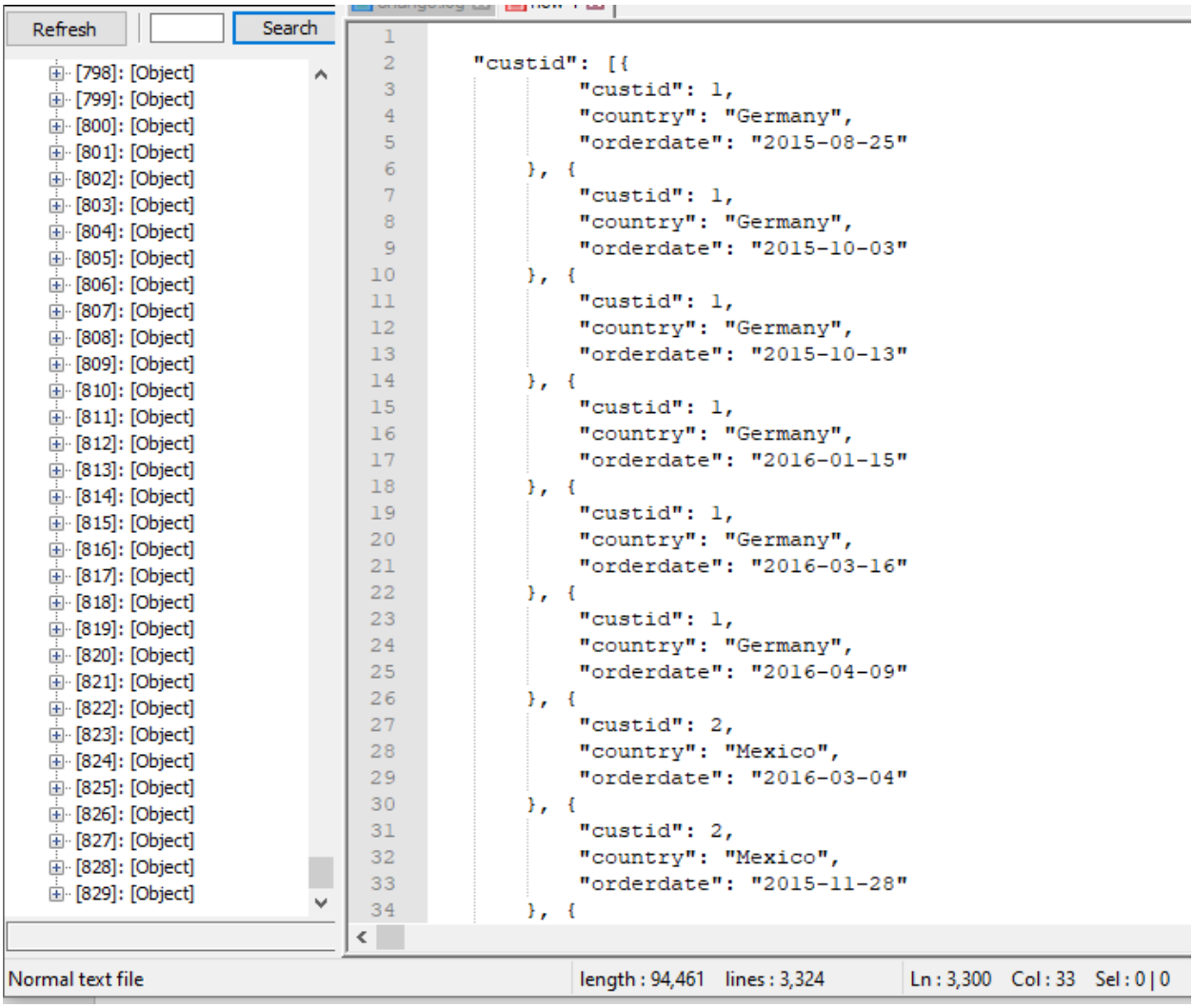
Sample Relational Output (830 Rows returned)

Results | Messages

	custid	country	orderdate
1	1	Germany	2015-08-25
2	1	Germany	2015-10-03
3	1	Germany	2015-10-13
4	1	Germany	2016-01-15
5	1	Germany	2016-03-16
6	1	Germany	2016-04-09
7	2	Mexico	2016-03-04
8	2	Mexico	2015-11-28
9	2	Mexico	2015-08-08
10	2	Mexico	2014-09-18
11	3	Mexico	2014-11-27
12	3	Mexico	2015-04-15
13	3	Mexico	2015-05-13
14	3	Mexico	2015-09-22
15	3	Mexico	2015-06-19
16	3	Mexico	2015-09-25

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (78) | TSQLV4 | 00:00:00 | 830 rows

Sample JSON Output (830 Objects returned)

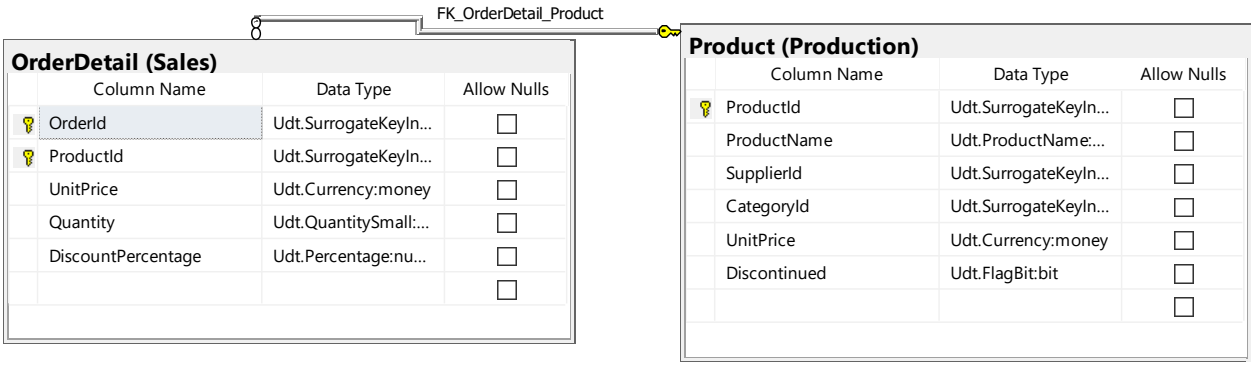

Refresh
Search
[798]: [Object]
[799]: [Object]
[800]: [Object]
[801]: [Object]
[802]: [Object]
[803]: [Object]
[804]: [Object]
[805]: [Object]
[806]: [Object]
[807]: [Object]
[808]: [Object]
[809]: [Object]
[810]: [Object]
[811]: [Object]
[812]: [Object]
[813]: [Object]
[814]: [Object]
[815]: [Object]
[816]: [Object]
[817]: [Object]
[818]: [Object]
[819]: [Object]
[820]: [Object]
[821]: [Object]
[822]: [Object]
[823]: [Object]
[824]: [Object]
[825]: [Object]
[826]: [Object]
[827]: [Object]
[828]: [Object]
[829]: [Object]
"custid": [{
"custid": 1,
"country": "Germany",
"orderdate": "2015-08-25"
}, {
"custid": 1,
"country": "Germany",
"orderdate": "2015-10-03"
}, {
"custid": 1,
"country": "Germany",
"orderdate": "2015-10-13"
}, {
"custid": 1,
"country": "Germany",
"orderdate": "2016-01-15"
}, {
"custid": 1,
"country": "Germany",
"orderdate": "2016-03-16"
}, {
"custid": 1,
"country": "Germany",
"orderdate": "2016-04-09"
}, {
"custid": 2,
"country": "Mexico",
"orderdate": "2016-03-04"
}, {
"custid": 2,
"country": "Mexico",
"orderdate": "2015-11-28"
}, {
Normal text file
length : 94,461 lines : 3,324
Ln : 3,300 Col : 33 Sel : 0 | 0

Simple Query 2

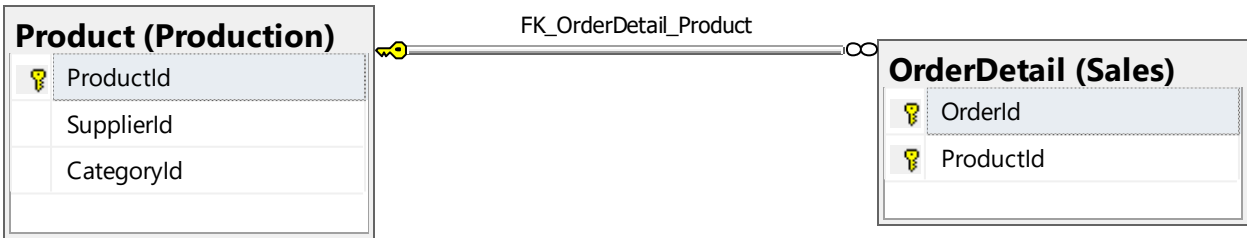
Returns a inner join of all product ids mapped to order id 42

Use Northwinds2020TSQLV6 Database and Production.Product and Sales.OrderDetail tables

Standard View



Key View



Columns from tables with order by

Table Name	Column Names	Order by
Production.Product	productid	ASC
Sales.OrderDetail	orderid	ASC

Solution Query Without JSON

```
USE Northwinds2020TSQLV6;
SELECT E.ProductId,
       F.OrderId
FROM Production.[Product] AS E
    INNER JOIN Sales.[OrderDetail] AS F
        ON F.ProductId = E.ProductId
WHERE E.ProductId = 42;
--FOR JSON PATH, ROOT ('custid'), INCLUDE_NULL_VALUES;
```

Solution Query With JSON

```
USE Northwinds2020TSQLV6;
SELECT E.ProductId,
       F.OrderId
FROM Production.[Product] AS E
    INNER JOIN Sales.[OrderDetail] AS F
        ON F.ProductId = E.ProductId
WHERE E.ProductId = 42
FOR JSON PATH, ROOT ('productid'), INCLUDE_NULL_VALUES;
```

Sample Relational Output (30 Rows returned)

Results | Messages

	ProductId	OrderId
1	42	10248
2	42	10309
3	42	10311
4	42	10332
5	42	10345
6	42	10404
7	42	10463
8	42	10492
9	42	10498
10	42	10516
11	42	10571
12	42	10588
13	42	10625
14	42	10652
15	42	10663
16	42	10673
17	42	10680
18	42	10746
19	42	10767
20	42	10776
21	42	10777
22	42	10845
23	42	10856
24	42	10882
25	42	10923
26	42	10996
27	42	11002
28	42	11007
29	42	11013

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (76) | Northwinds2020TSQLV6 | 00:00:00 | 30 rows

Sample JSON Output (30 Objects returned)

ROOT
- productid: [Array]
  - [0]: [Object]
  - [1]: [Object]
  - [2]: [Object]
  - [3]: [Object]
  - [4]: [Object]
  - [5]: [Object]
  - [6]: [Object]
  - [7]: [Object]
  - [8]: [Object]
  - [9]: [Object]
  - [10]: [Object]
  - [11]: [Object]
  - [12]: [Object]
  - [13]: [Object]
  - [14]: [Object]
  - [15]: [Object]
  - [16]: [Object]
  - [17]: [Object]
  - [18]: [Object]
  - [19]: [Object]
  - [20]: [Object]
  - [21]: [Object]
  - [22]: [Object]
  - [23]: [Object]
  - [24]: [Object]
  - [25]: [Object]
  - [26]: [Object]
  - [27]: [Object]
  - [28]: [Object]
  - [29]: [Object]

```
        }, {
            "ProductId": 42,
            "OrderId": 10777
        }, {
            "ProductId": 42,
            "OrderId": 10845
        }, {
            "ProductId": 42,
            "OrderId": 10856
        }, {
            "ProductId": 42,
            "OrderId": 10882
        }, {
            "ProductId": 42,
            "OrderId": 10923
        }, {
            "ProductId": 42,
            "OrderId": 10996
        }, {
            "ProductId": 42,
            "OrderId": 11002
        }, {
            "ProductId": 42,
            "OrderId": 11007
        }, {
            "ProductId": 42,
            "OrderId": 11013
        }, {
            "ProductId": 42,
            "OrderId": 11035
        }
    ]
```

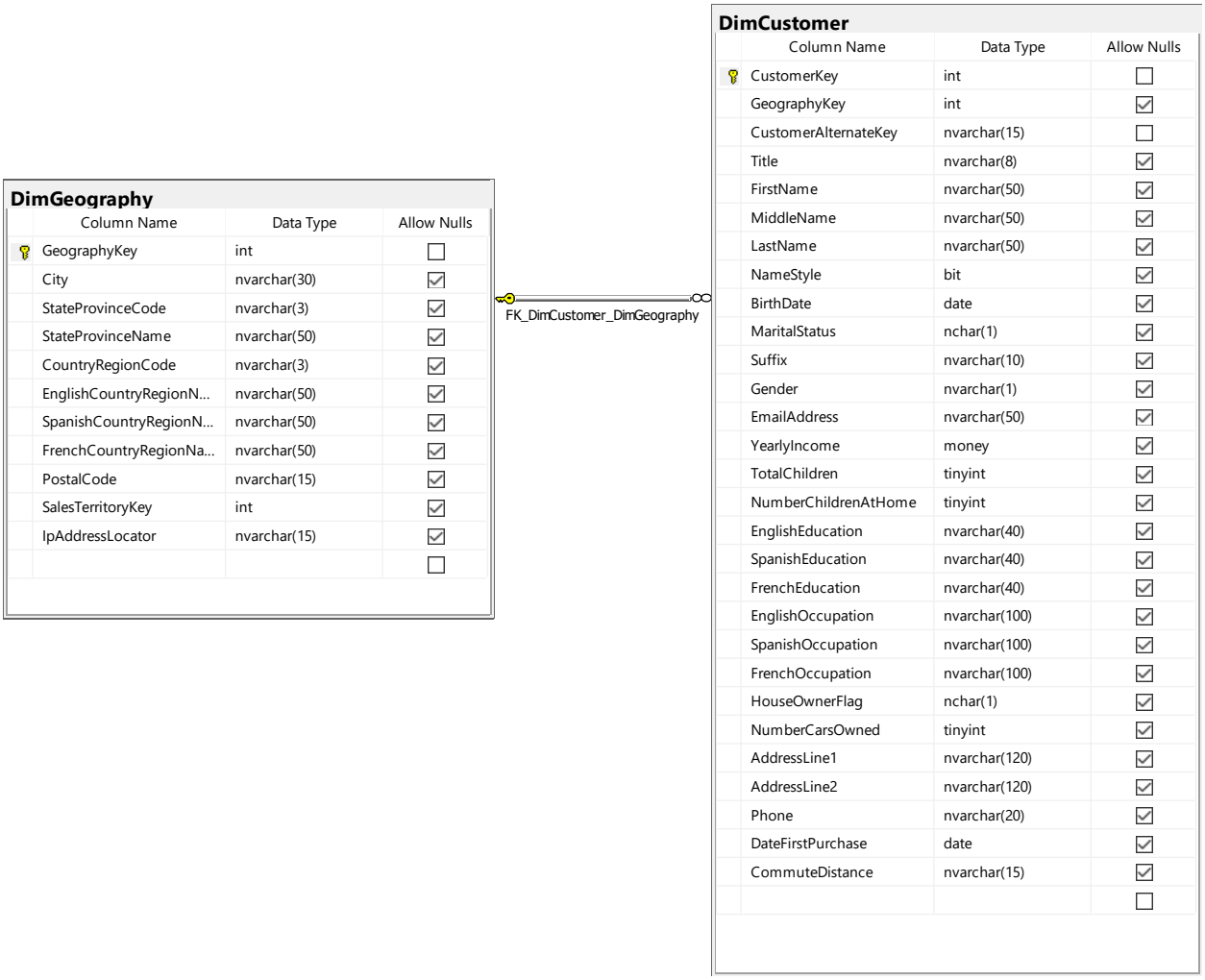
Normal text file | length : 2,249 lines : 94 | Ln

Simple Query 3

Returns all Customer keys and Employee keys with their first and last name with the city they are from

Use AdventureWorksDW2017 Database and dbo.DimCustomer and dbo.DimGeography tables

Standard table view



Key view

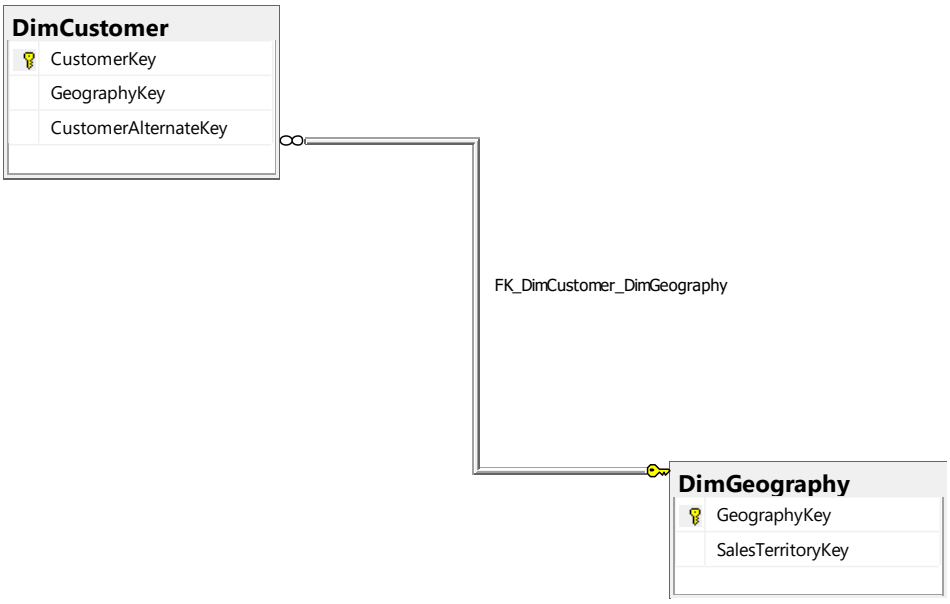


Table Names with column and order by

Table Name	Column Names	Order By
Dbo.Geography	CustomerKey FirstName LastName	ASC ASC ASC
Dbo.DimCustomer	City	ASC

Sample relational solution and output (18484 rows returned)

```
USE AdventureWorksDW2017;
SELECT E.CustomerKey,
       E.FirstName,
       E.LastName,
       F.EmployeeKey,
       F.FirstName,
       F.LastName
FROM dbo.[DimCustomer] AS E
    FULL OUTER JOIN dbo.[DimEmployee] AS F
        ON E.CustomerKey = F.EmployeeKey;
```

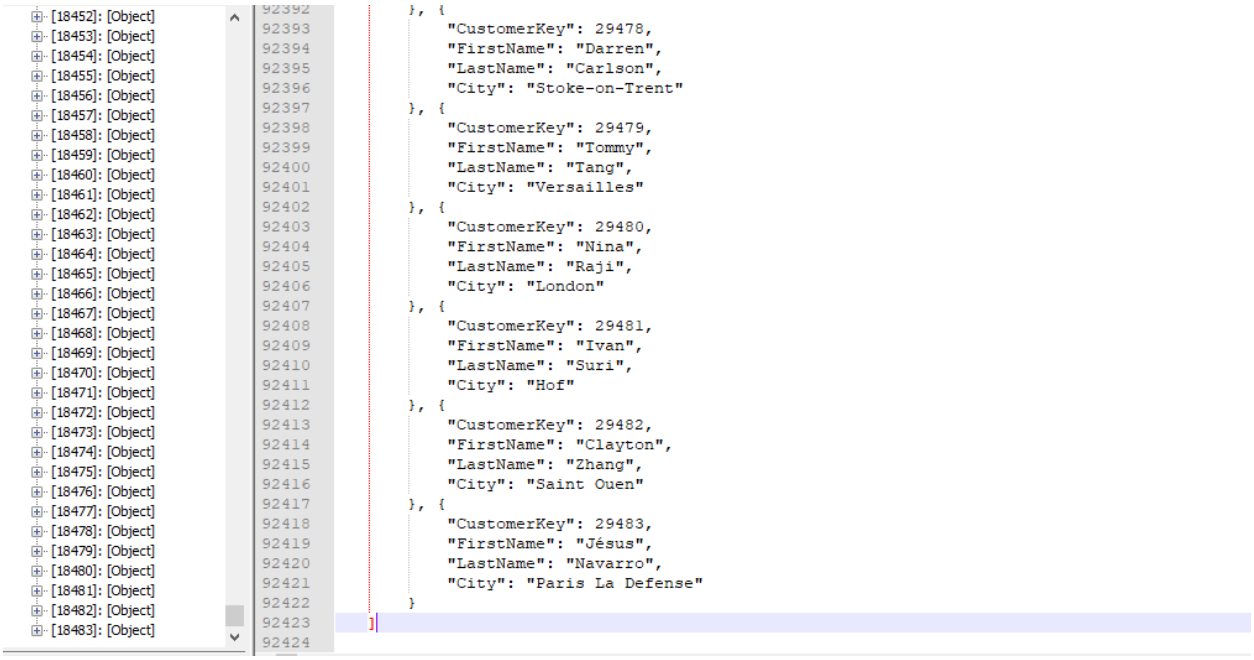
Results | Messages

	CustomerKey	FirstName	LastName	City
1	11000	Jon	Yang	Rockhampton
2	11001	Eugene	Huang	Seaford
3	11002	Ruben	Torres	Hobart
4	11003	Christy	Zhu	North Ryde
5	11004	Elizabeth	Johnson	Wollongong
6	11005	Julio	Ruiz	East Brisbane
7	11006	Janet	Alvarez	Matraville
8	11007	Marco	Mehta	Warrambool
9	11008	Rob	Verhoff	Bendigo
10	11009	Shannon	Carlson	Hervey Bay
11	11010	Jacquelyn	Suarez	East Brisbane
12	11011	Curtis	Lu	East Brisbane
13	11012	Lauren	Walker	Bremerton
14	11013	Ian	Jenkins	Lebanon
15	11014	Sydney	Bennett	Redmond
16	11015	Chloe	Young	Burbank
17	11016	Wyatt	Hill	Imperial Beach
18	11017	Shannon	Wang	Sunbury
19	11018	Clarence	Rai	Bendigo

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (76) | AdventureWorksDW2017 | 00:00:00 | 18,484 rows

Sample JSON solution and output (18780 Objects returned)

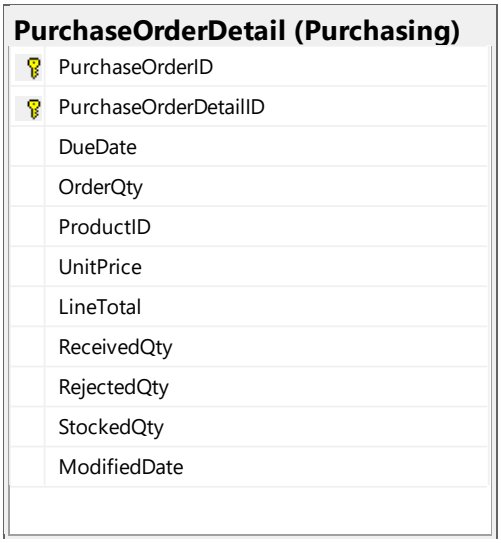
```sql
USE AdventureWorksDW2017;
SELECT E.CustomerKey,
       E.FirstName,
       E.LastName,
       F.City
FROM dbo.[DimCustomer] AS E
    INNER JOIN dbo.[DimGeography] AS F
        ON F.GeographyKey = E.GeographyKey
FOR JSON PATH, ROOT('E.CustomerKey'), INCLUDE_NULL_VALUES;
```

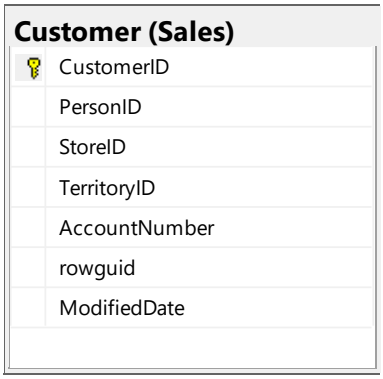


Simple Query 4

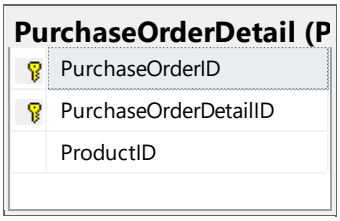
```
Performs a left join and returns the purchase order id with the amount of units the
customer ordered where the quantity is greater than 10 and not null

Use AdventureWorksDw database and PurchaseOrderDetail and Sales.Customer Tables
```

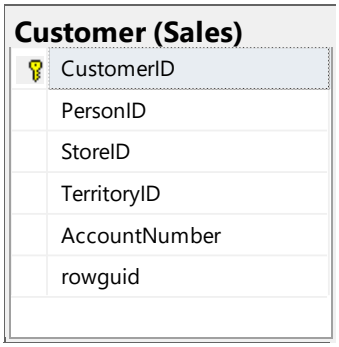
Standard view





Key view





Columns from tables with order by

Table Name	Column Names	Order By
PurchaseOrderDetail	PurchaseOrderID OrderQty	ASC ASC
Sales.Customer	CustomerID	ASC

Sample relational solution and output (809 rows returned)

```
USE AdventureWorks2017;
SELECT E.PurchaseOrderID,
       E.OrderQty,
       F.CustomerID
FROM Purchasing.[PurchaseOrderDetail] AS E
    LEFT JOIN Sales.[Customer] AS F
        ON E.PurchaseOrderID = F.CustomerID
WHERE E.OrderQty > 10
      AND F.CustomerID IS NOT NULL
ORDER BY E.OrderQty

FOR JSON PATH, ROOT('E.CustomerKey'), INCLUDE_NULL_VALUES;
```

	PurchaseOrderID	OrderQty	CustomerID
1	10	60	10
2	13	60	13
3	18	60	18
4	18	60	18
5	18	60	18
6	18	60	18
7	18	60	18
8	20	60	20
9	37	60	37
10	37	60	37
11	48	60	48
12	48	60	48
13	48	60	48
14	48	60	48
15	48	60	48
16	55	60	55
17	57	60	57
18	57	60	57
19	73	60	73

Query executed successfully. localhost,12001 (15.0 RTM) sa (76) AdventureWorks2017 00:00:00 809 rows

Sample JSON solution and output (809 objects returned)


```
[777]: [Object]
[778]: [Object]
[779]: [Object]
[780]: [Object]
[781]: [Object]
[782]: [Object]
[783]: [Object]
[784]: [Object]
[785]: [Object]
[786]: [Object]
[787]: [Object]
[788]: [Object]
[789]: [Object]
[790]: [Object]
[791]: [Object]
[792]: [Object]
[793]: [Object]
[794]: [Object]
[795]: [Object]
[796]: [Object]
[797]: [Object]
[798]: [Object]
[799]: [Object]
[800]: [Object]
[801]: [Object]
[802]: [Object]
[803]: [Object]
[804]: [Object]
[805]: [Object]
[806]: [Object]
[807]: [Object]
[808]: [Object]
```

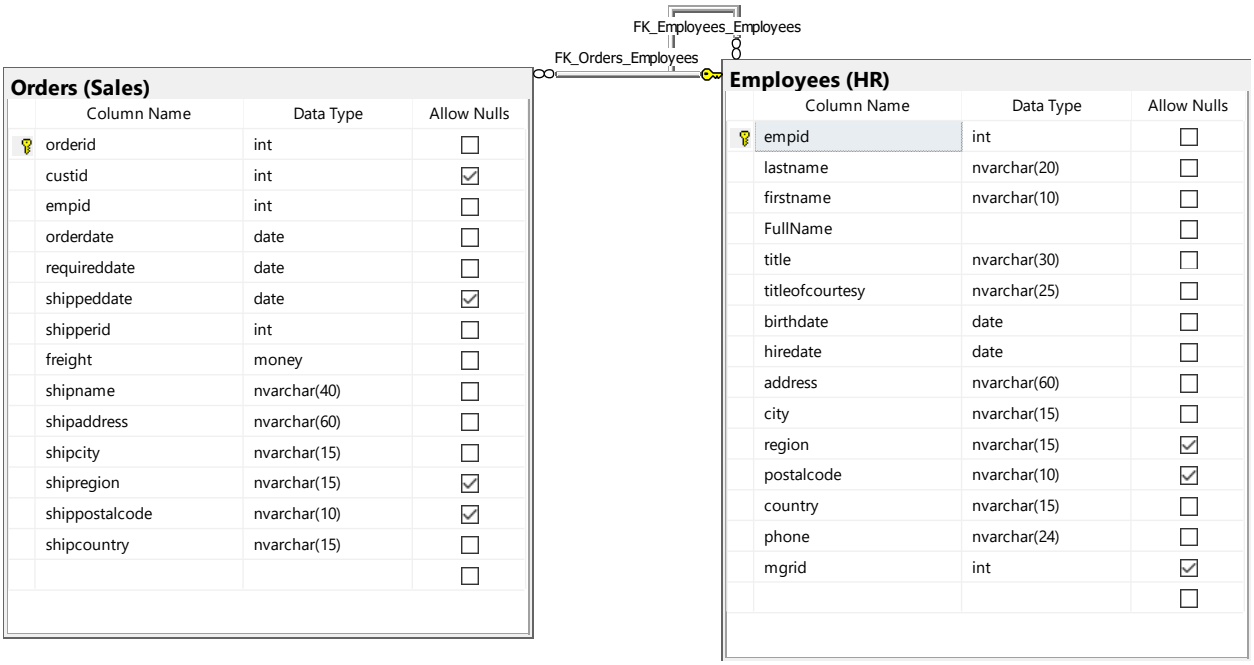
```
3208            "OrderQty": 1250,
3209            "CustomerID": 386
3210        }, {
3211            "PurchaseOrderID": 386,
3212            "OrderQty": 1250,
3213            "CustomerID": 386
3214        }, {
3215            "PurchaseOrderID": 495,
3216            "OrderQty": 1250,
3217            "CustomerID": 495
3218        }, {
3219            "PurchaseOrderID": 495,
3220            "OrderQty": 1250,
3221            "CustomerID": 495
3222        }, {
3223            "PurchaseOrderID": 578,
3224            "OrderQty": 1250,
3225            "CustomerID": 578
3226        }, {
3227            "PurchaseOrderID": 578,
3228            "OrderQty": 1250,
3229            "CustomerID": 578
3230        }, {
3231            "PurchaseOrderID": 665,
3232            "OrderQty": 1250,
3233            "CustomerID": 665
3234        }, {
3235            "PurchaseOrderID": 665,
3236            "OrderQty": 1250,
3237            "CustomerID": 665
3238        }
3239    ]
3240
```

Normal text file | length : 90,269 lines : 3,240 | Ln : 3,224 Col : 30 Sel : 0 | 0 | Windows (CR LF) | UTF-8 | INS

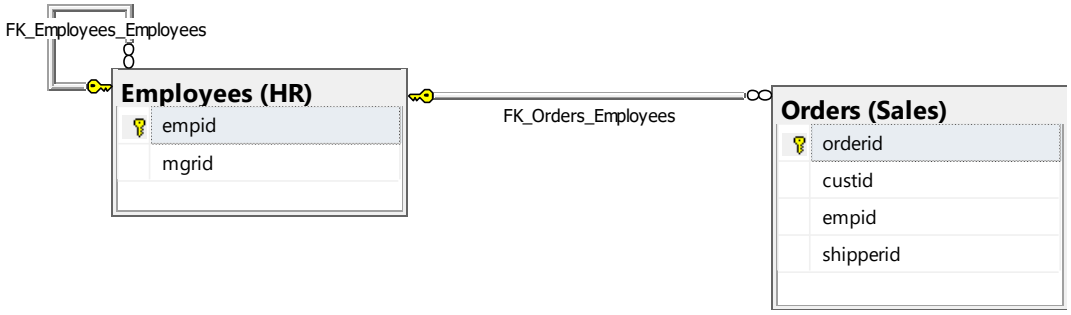
Simple Query 5

```
Performs a full outer join and returns the employee id and employee name with the order
id and date they handled the order
```

Standard view



Key view



Columns from tables with order by

Table Name	Column Names	Order By
HR.Employees	Empid Fullname	ASC ASC
Sales.Orders	OrderId OrderDate	ASC ASC

Sample Relation Solution with output (830 rows returned)

```
USE TSQLV4;
SELECT E.empid AS EmployeeID,
       E.FullName AS EmployeeName,
       F.orderid,
       F.orderdate
FROM HR.Employees AS E
    FULL OUTER JOIN Sales.Orders AS F
        ON F.empid = E.empid
WHERE E.empid IS NOT NULL
      AND F.orderid IS NOT NULL;
--FOR JSON PATH, ROOT('E.CustomerKey'), INCLUDE_NULL_VALUES;
```

Results | Messages

	EmployeeID	EmployeeName	orderid	orderdate
1	5	Sven Mortensen	10248	2014-07-04
2	6	Paul Suurs	10249	2014-07-05
3	4	Yael Peled	10250	2014-07-08
4	3	Judy Lew	10251	2014-07-08
5	4	Yael Peled	10252	2014-07-09
6	3	Judy Lew	10253	2014-07-10
7	5	Sven Mortensen	10254	2014-07-11
8	9	Patricia Doyle	10255	2014-07-12
9	3	Judy Lew	10256	2014-07-15
10	4	Yael Peled	10257	2014-07-16
11	1	Sara Davis	10258	2014-07-17
12	4	Yael Peled	10259	2014-07-18
13	4	Yael Peled	10260	2014-07-19
14	4	Yael Peled	10261	2014-07-19
15	8	Maria Cameron	10262	2014-07-22
16	9	Patricia Doyle	10263	2014-07-23
17	6	Paul Suurs	10264	2014-07-24
18	2	Don Funk	10265	2014-07-25
19	3	Judy Lew	10266	2014-07-26

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (61) | TSQLV4 | 00:00:00 | 830 rows

Sample JSON Solution with output (830 Objects returned)

```
USE TSQLV4;
SELECT E.empid AS EmployeeID,
       E.FullName AS EmployeeName,
       F.orderid,
       F.orderdate
FROM HR.Employees AS E
    FULL OUTER JOIN Sales.Orders AS F
        ON F.empid = E.empid
WHERE E.empid IS NOT NULL
      AND F.orderid IS NOT NULL
```

```sql
FOR JSON PATH, ROOT('E.CustomerKey'), INCLUDE_NULL_VALUES;
```


```
[798]: [Object]
[799]: [Object]
[800]: [Object]
[801]: [Object]
[802]: [Object]
[803]: [Object]
[804]: [Object]
[805]: [Object]
[806]: [Object]
[807]: [Object]
[808]: [Object]
[809]: [Object]
[810]: [Object]
[811]: [Object]
[812]: [Object]
[813]: [Object]
[814]: [Object]
[815]: [Object]
[816]: [Object]
[817]: [Object]
[818]: [Object]
[819]: [Object]
[820]: [Object]
[821]: [Object]
[822]: [Object]
[823]: [Object]
[824]: [Object]
[825]: [Object]
[826]: [Object]
[827]: [Object]
[828]: [Object]
[829]: [Object]
```

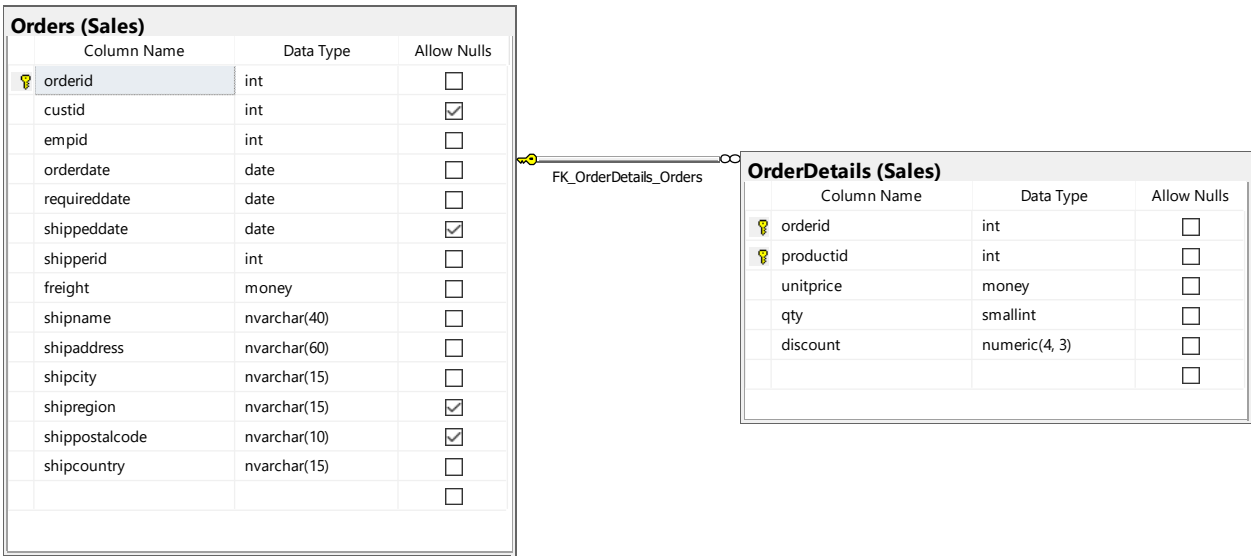
```
4122        }, {
4123            "EmployeeID": 4,
4124            "EmployeeName": "Yael Peled",
4125            "orderid": 11072,
4126            "orderdate": "2016-05-05"
4127        }, {
4128            "EmployeeID": 2,
4129            "EmployeeName": "Don Funk",
4130            "orderid": 11073,
4131            "orderdate": "2016-05-05"
4132        }, {
4133            "EmployeeID": 7,
4134            "EmployeeName": "Russell King",
4135            "orderid": 11074,
4136            "orderdate": "2016-05-06"
4137        }, {
4138            "EmployeeID": 8,
4139            "EmployeeName": "Maria Cameron",
4140            "orderid": 11075,
4141            "orderdate": "2016-05-06"
4142        }, {
4143            "EmployeeID": 4,
4144            "EmployeeName": "Yael Peled",
4145            "orderid": 11076,
4146            "orderdate": "2016-05-06"
4147        }, {
4148            "EmployeeID": 1,
4149            "EmployeeName": "Sara Davis",
4150            "orderid": 11077,
4151            "orderdate": "2016-05-06"
4152        }
4153    ]
4154
```

Normal text file | length : 130,693 lines : 4,154 | Ln : 4,142 Col : 13 Sel : 0 | 0 | Windows (CR LF) | UTF-8

Medium Query 1

```
Performs an inner join on the two tables and returns all order id, unit price, order id
and order date in 2016 in descending order

Use TSQLV4 database and Sales.Orders and Sales.OrderDetails tables
```

Standard view



Key view

Columns from tables with order by

Table Name	Column Names	Order By
Dbo.OrderDetails	OrderId Unitprice	ASC ASC
Dbo.Orders	OrderDate	DESC

Sample relation solution with output (686 rows returned)

```
USE TSQLV4;
SELECT O.orderid,
       O.unitprice,
       P.orderdate
FROM dbo.Orderdetails AS O
    INNER JOIN dbo.Orders AS P
        ON P.orderid = O.orderid
           AND YEAR(P.orderdate) = 2016
GROUP BY O.orderid,
         O.unitprice,
         P.orderid,
         P.orderdate
ORDER BY P.orderdate DESC;
--FOR JSON PATH, ROOT('orderid'), INCLUDE_NULL_VALUES;
```

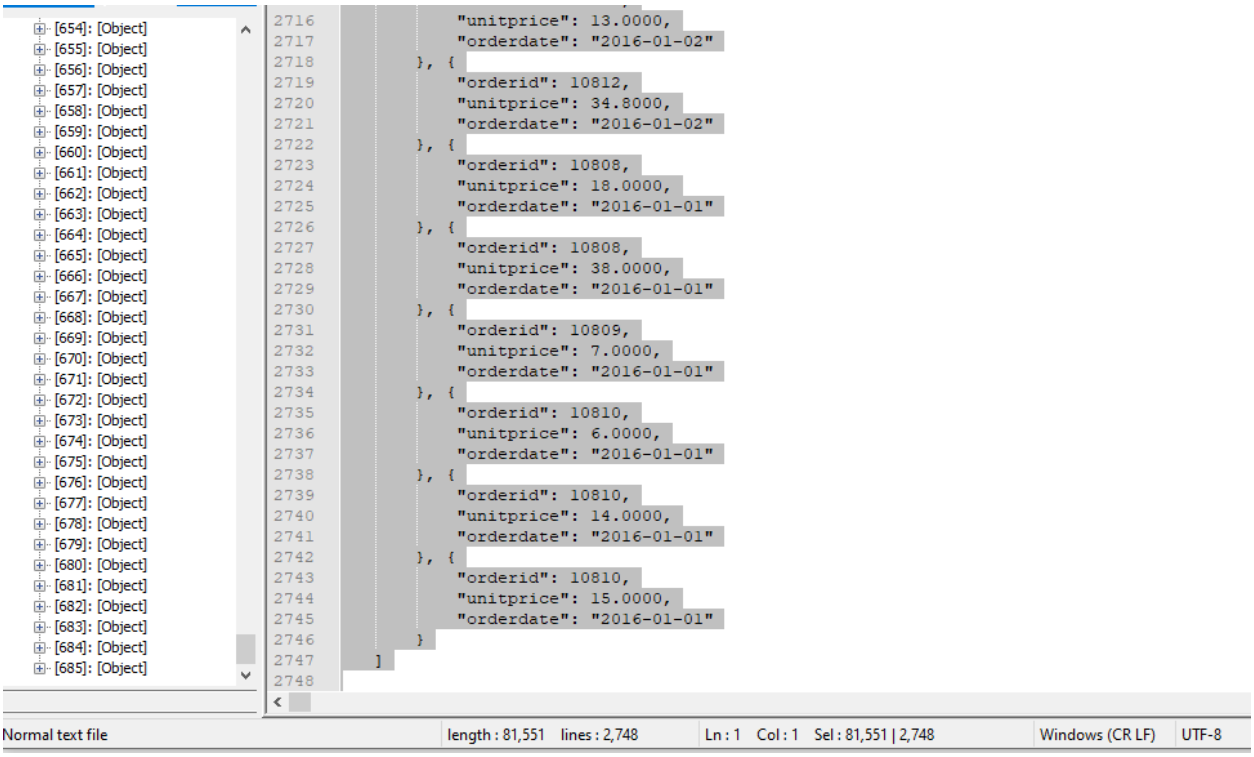
Results | Messages

	orderid	unitprice	orderdate
1	11074	17.45	2016-05-06
2	11075	12.00	2016-05-06
3	11075	18.00	2016-05-06
4	11075	19.00	2016-05-06
5	11076	9.20	2016-05-06
6	11076	23.25	2016-05-06
7	11076	25.00	2016-05-06
8	11077	6.00	2016-05-06
9	11077	7.00	2016-05-06
10	11077	7.75	2016-05-06
11	11077	9.00	2016-05-06
12	11077	9.65	2016-05-06
13	11077	10.00	2016-05-06
14	11077	12.00	2016-05-06
15	11077	13.00	2016-05-06
16	11077	15.00	2016-05-06
17	11077	17.00	2016-05-06
18	11077	17.45	2016-05-06
19	11077	18.00	2016-05-06

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (73) | TSQLV4 | 00:00:00 | 686 rows

Sample JSON solution with output (686 objects returned)

```
USE TSQLV4;
SELECT O.orderid,
       O.unitprice,
       P.orderdate
FROM dbo.Orderdetails AS O
    INNER JOIN dbo.Orders AS P
        ON P.orderid = O.orderid
           AND YEAR(P.orderdate) = 2016
GROUP BY O.orderid,
         O.unitprice,
         P.orderid,
         P.orderdate
ORDER BY P.orderdate DESC
FOR JSON PATH, ROOT('orderid'), INCLUDE_NULL_VALUES;
```




```
[654]: [Object]
[655]: [Object]
[656]: [Object]
[657]: [Object]
[658]: [Object]
[659]: [Object]
[660]: [Object]
[661]: [Object]
[662]: [Object]
[663]: [Object]
[664]: [Object]
[665]: [Object]
[666]: [Object]
[667]: [Object]
[668]: [Object]
[669]: [Object]
[670]: [Object]
[671]: [Object]
[672]: [Object]
[673]: [Object]
[674]: [Object]
[675]: [Object]
[676]: [Object]
[677]: [Object]
[678]: [Object]
[679]: [Object]
[680]: [Object]
[681]: [Object]
[682]: [Object]
[683]: [Object]
[684]: [Object]
[685]: [Object]
```

```
            "unitprice": 13.0000,
            "orderdate": "2016-01-02"
        }, {
            "orderid": 10812,
            "unitprice": 34.8000,
            "orderdate": "2016-01-02"
        }, {
            "orderid": 10808,
            "unitprice": 18.0000,
            "orderdate": "2016-01-01"
        }, {
            "orderid": 10808,
            "unitprice": 38.0000,
            "orderdate": "2016-01-01"
        }, {
            "orderid": 10809,
            "unitprice": 7.0000,
            "orderdate": "2016-01-01"
        }, {
            "orderid": 10810,
            "unitprice": 6.0000,
            "orderdate": "2016-01-01"
        }, {
            "orderid": 10810,
            "unitprice": 14.0000,
            "orderdate": "2016-01-01"
        }, {
            "orderid": 10810,
            "unitprice": 15.0000,
            "orderdate": "2016-01-01"
        }
    ]
```


Normal text file | length : 81,551 lines : 2,748 | Ln : 1 Col : 1 Sel : 81,551 | 2,748 | Windows (CR LF) | UTF-8

Medium Query 2

```
Performs an Inner join and returns all distinct customer/employee ids and fright where
the city is London

Use NorthwindsTSQLV6 database and Sales.Customer and Sales.Order tables
```

Standard View

Key view

Columns from tables with order by

Table Name	Column Names	Order By
Sales.Customer	CustomerId CustomerCity	ASC ASC
Sales.Order	EmployeeId Freight	ASC ASC

Sample Relational solution with output (1 row)

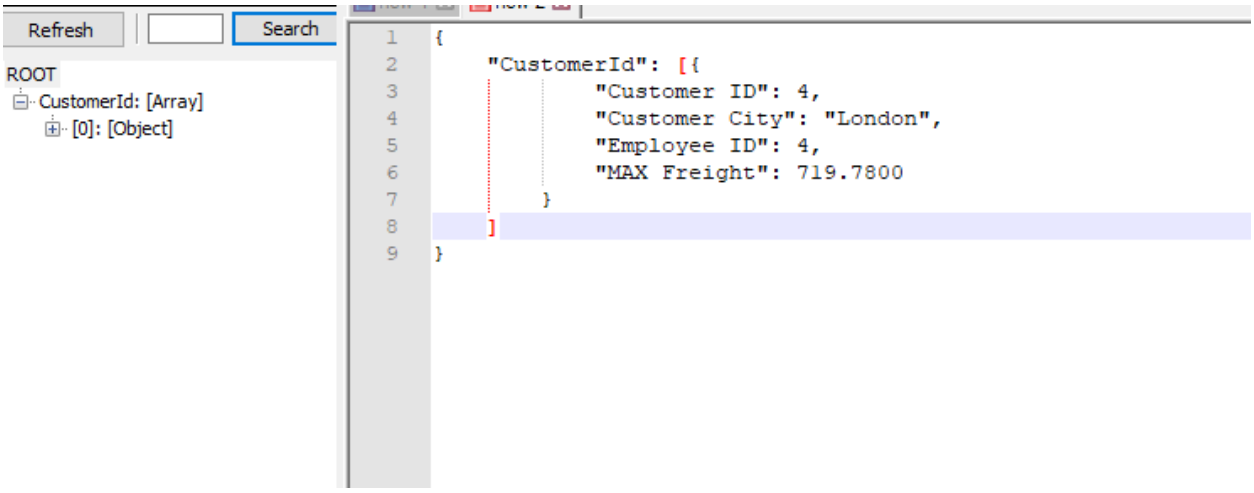
```
USE Northwinds2020TSQLV6;
SELECT DISTINCT
       A.CustomerId AS [Customer ID],
       A.CustomerCity AS [Customer City],
       B.EmployeeId AS [Employee ID],
       MAX(B.Freight) AS [MAX Freight]
FROM Sales.[Customer] AS A
    INNER JOIN Sales.[Order] AS B
        ON A.CustomerId = B.EmployeeId
WHERE B.Freight > 20
      AND A.CustomerCity = 'London'
GROUP BY A.CustomerId,
         A.CustomerCity,
         B.EmployeeId;
--FOR JSON PATH, ROOT('CustomerId'), INCLUDE_NULL_VALUES;
```

Results | Messages

	Customer ID	Customer City	Employee ID	MAX Freight
1	4	London	4	719.78

Sample Relational solution with output (1 Object)

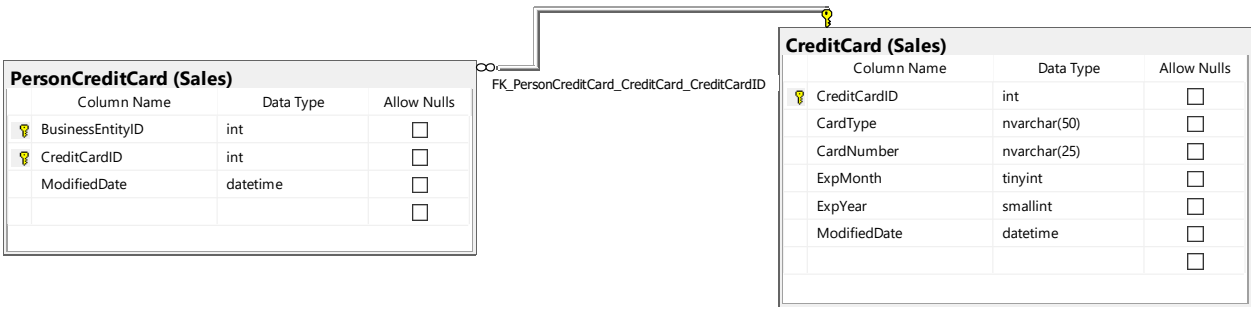
```
USE Northwinds2020TSQLV6;
SELECT DISTINCT
       A.CustomerId AS [Customer ID],
       A.CustomerCity AS [Customer City],
       B.EmployeeId AS [Employee ID],
       MAX(B.Freight) AS [MAX Freight]
FROM Sales.[Customer] AS A
    INNER JOIN Sales.[Order] AS B
        ON A.CustomerId = B.EmployeeId
WHERE B.Freight > 20
      AND A.CustomerCity = 'London'
GROUP BY A.CustomerId,
         A.CustomerCity,
         B.EmployeeId
FOR JSON PATH, ROOT('CustomerId'), INCLUDE_NULL_VALUES;
```


Refresh
Search
ROOT
CustomerId: [Array]
[0]: [Object]
{
"CustomerId": [{
"Customer ID": 4,
"Customer City": "London",
"Employee ID": 4,
"MAX Freight": 719.7800
}
]
}

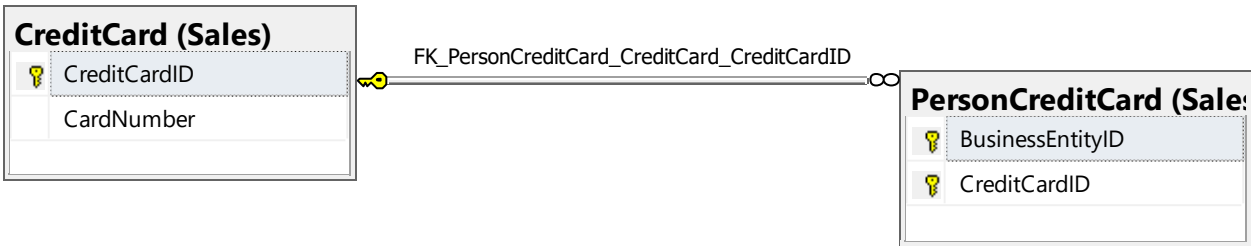
Medium Query 3

```
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

Use AdventureWorks2017 database and Sales.CreditCard and Sales.PersonCreditCard tables
```

Standard view



Key view



Columns from tables with order by

Table Name	Column Names	Order By
Sales.CreditCard	CreditCardID CardNumber	ASC
Sales.PersonCreditCard	BusinessEntityID	ASC

Sample Relational solution and output (1186 rows)

```
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('CreditCardID'), INCLUDE_NULL_VALUES;
```

Results | Messages

	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
13	18908	33335344128204	719
14	18407	33336686970946	723
15	18323	77774845469083	727
16	18049	33337924710118	737
17	18502	33334506790234	741
18	18496	55551787161363	787
19	18231	11113446412483	807

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (73) | AdventureWorks2017 | 00:00:00 | 1,186 rows

Sample JSON solution and output (1186 Objects)

```
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('CreditCardID'), INCLUDE_NULL_VALUES;
```

```
[1154]: [Object]
[1155]: [Object]
[1156]: [Object]
[1157]: [Object]
[1158]: [Object]
[1159]: [Object]
[1160]: [Object]
[1161]: [Object]
[1162]: [Object]
[1163]: [Object]
[1164]: [Object]
[1165]: [Object]
[1166]: [Object]
[1167]: [Object]
[1168]: [Object]
[1169]: [Object]
[1170]: [Object]
[1171]: [Object]
[1172]: [Object]
[1173]: [Object]
[1174]: [Object]
[1175]: [Object]
[1176]: [Object]
[1177]: [Object]
[1178]: [Object]
[1179]: [Object]
[1180]: [Object]
[1181]: [Object]
[1182]: [Object]
[1183]: [Object]
[1184]: [Object]
[1185]: [Object]
```

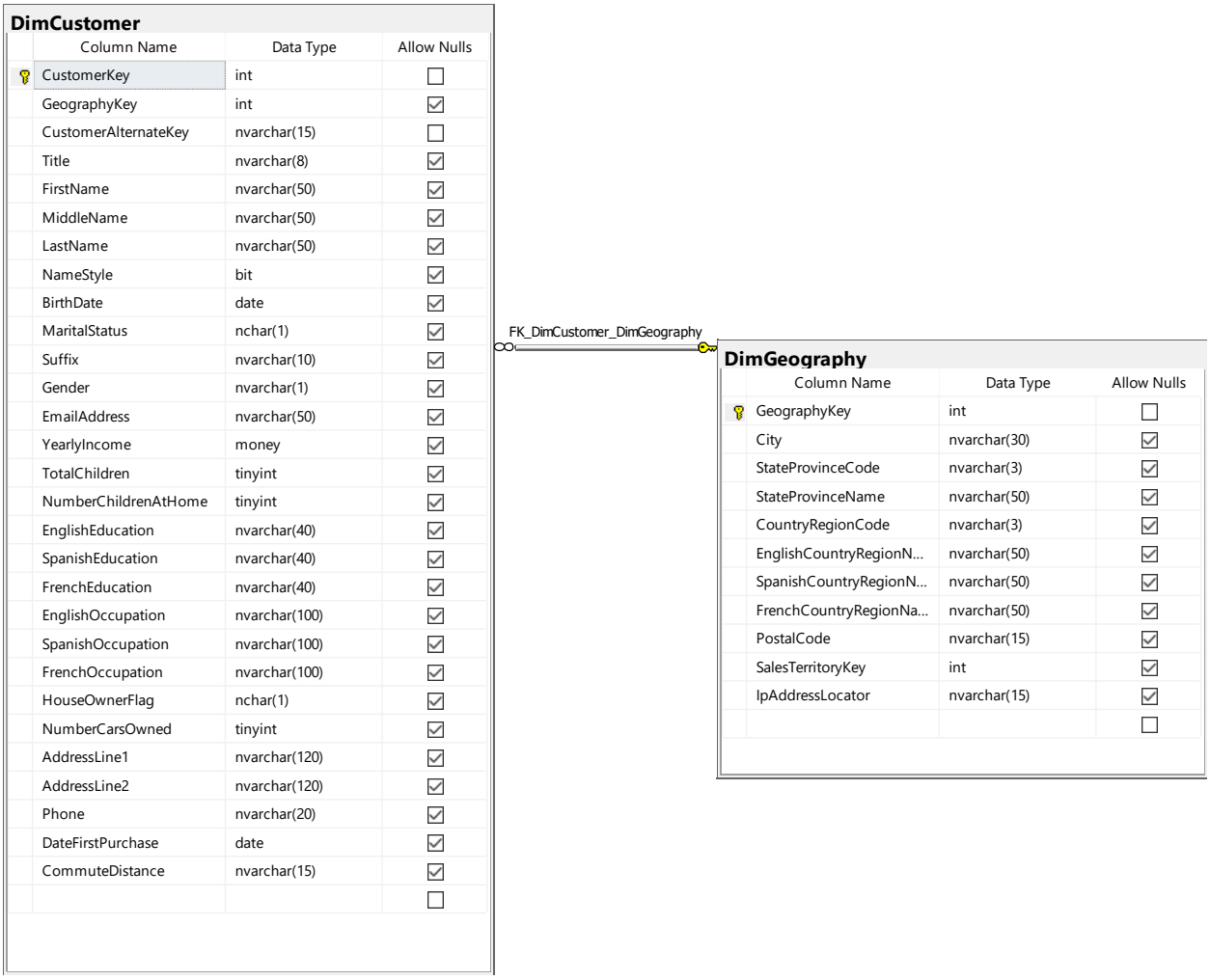
```
            "CardNumber": "11115363680521",
            "BusinessEntityID": 19818
        }, {
            "Min card": 19150,
            "CardNumber": "55558608871872",
            "BusinessEntityID": 19822
        }, {
            "Min card": 18817,
            "CardNumber": "33333782154517",
            "BusinessEntityID": 19846
        }, {
            "Min card": 18394,
            "CardNumber": "33333894853879",
            "BusinessEntityID": 19887
        }, {
            "Min card": 19149,
            "CardNumber": "77777760423102",
            "BusinessEntityID": 19919
        }, {
            "Min card": 18525,
            "CardNumber": "11117601945705",
            "BusinessEntityID": 19951
        }, {
            "Min card": 18353,
            "CardNumber": "55557594877073",
            "BusinessEntityID": 19984
        }, {
            "Min card": 18341,
            "CardNumber": "77779521438776",
            "BusinessEntityID": 20024
        }
    ]
```

lormal text file | length : 153,618 lines : 4,748 | Ln : 1 Col : 1 Sel : 153,618 | 4,748 | Windows (CR LF) | UTF-8

Medium Query 4

```
Performs an Inner join and returns the maximum income and max geo key from each customer
id between 15000 – 15500

Use AdventureWorksDW2017 database and dbo.DimGeography and dbo.DimCustomer tables
```

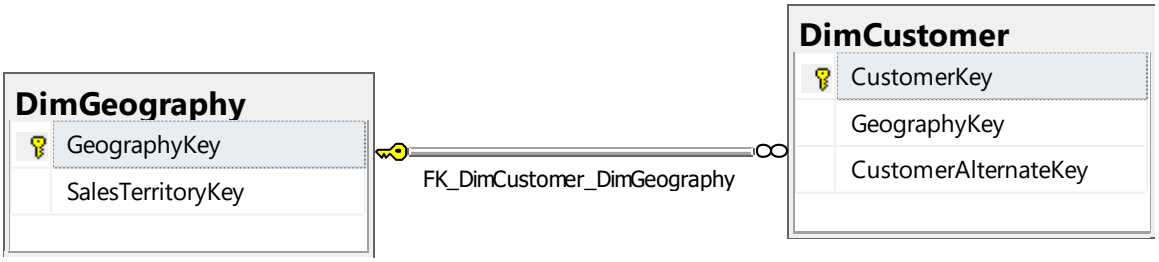
Standard view



Column Name	Data Type	Allow Nulls
CustomerKey	int	☐
GeographyKey	int	☑
CustomerAlternateKey	nvarchar(15)	☐
Title	nvarchar(8)	☑
FirstName	nvarchar(50)	☑
MiddleName	nvarchar(50)	☑
LastName	nvarchar(50)	☑
NameStyle	bit	☑
BirthDate	date	☑
MaritalStatus	nchar(1)	☑
Suffix	nvarchar(10)	☑
Gender	nvarchar(1)	☑
EmailAddress	nvarchar(50)	☑
YearlyIncome	money	☑
TotalChildren	tinyint	☑
NumberChildrenAtHome	tinyint	☑
EnglishEducation	nvarchar(40)	☑
SpanishEducation	nvarchar(40)	☑
FrenchEducation	nvarchar(40)	☑
EnglishOccupation	nvarchar(100)	☑
SpanishOccupation	nvarchar(100)	☑
FrenchOccupation	nvarchar(100)	☑
HouseOwnerFlag	nchar(1)	☑
NumberCarsOwned	tinyint	☑
AddressLine1	nvarchar(120)	☑
AddressLine2	nvarchar(120)	☑
Phone	nvarchar(20)	☑
DateFirstPurchase	date	☑
CommuteDistance	nvarchar(15)	☑
		☐

Column Name	Data Type	Allow Nulls
GeographyKey	int	☐
City	nvarchar(30)	☑
StateProvinceCode	nvarchar(3)	☑
StateProvinceName	nvarchar(50)	☑
CountryRegionCode	nvarchar(3)	☑
EnglishCountryRegionN...	nvarchar(50)	☑
SpanishCountryRegionN...	nvarchar(50)	☑
FrenchCountryRegionNa...	nvarchar(50)	☑
PostalCode	nvarchar(15)	☑
SalesTerritoryKey	int	☑
IpAddressLocator	nvarchar(15)	☑
		☐

Key view



Columns from tables with order by

Table Name	Column Names	Order By
Dbo.DimCustomer	GeographyKey CustomerKey YearlyIncome	ASC ASC ASC
Dbo.DimGeography	City CountryRegionCode	ASC ASC

Sample relational solution with output (499 rows returned)

```
USE AdventureWorksDW2017;
SELECT MAX(C.GeographyKey) AS [Max Geo],
       C.CustomerKey,
       C.YearlyIncome AS [Max Income],
       D.City,
       D.CountryRegionCode
FROM dbo.DimCustomer AS C
    INNER JOIN dbo.DimGeography AS D
        ON C.GeographyKey = D.GeographyKey
WHERE C.CustomerKey > 15000
      AND C.CustomerKey < 15500
GROUP BY C.CustomerKey,
         C.YearlyIncome,
         D.City,
         D.CountryRegionCode;
--FOR JSON PATH, ROOT('CreditCardID'), INCLUDE_NULL_VALUES;
```

Results | Messages

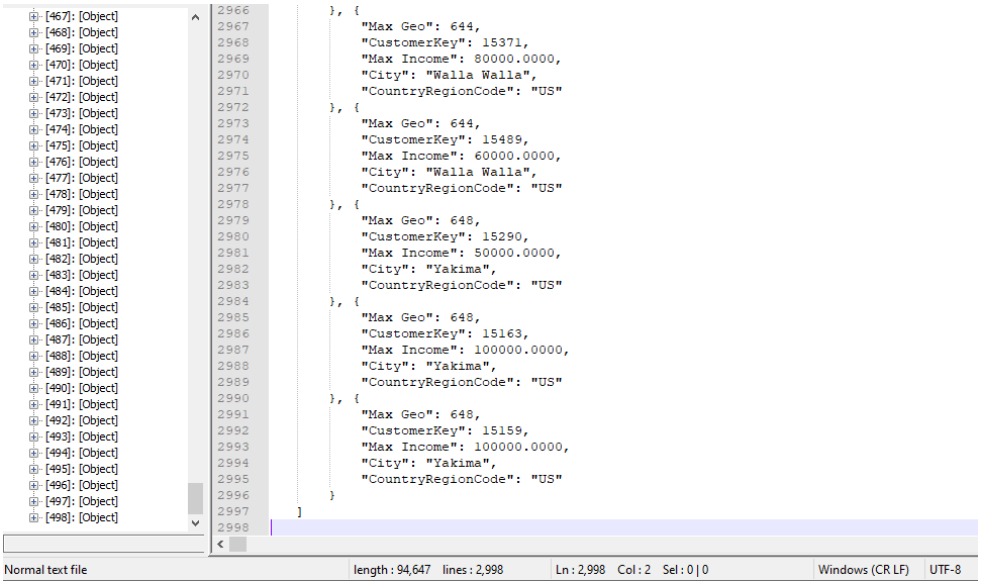
	Max Geo	CustomerKey	Max Income	City	CountryRegionCode
1	2	15019	30000.00	Coffs Harbour	AU
2	2	15042	20000.00	Coffs Harbour	AU
3	2	15153	110000.00	Coffs Harbour	AU
4	2	15262	30000.00	Coffs Harbour	AU
5	3	15220	70000.00	Darlinghurst	AU
6	3	15238	70000.00	Darlinghurst	AU
7	4	15212	60000.00	Goulburn	AU
8	4	15021	30000.00	Goulburn	AU
9	4	15425	10000.00	Goulburn	AU
10	5	15436	30000.00	Lane Cove	AU
11	5	15127	90000.00	Lane Cove	AU
12	5	15146	100000.00	Lane Cove	AU
13	5	15208	60000.00	Lane Cove	AU
14	5	15226	70000.00	Lane Cove	AU
15	6	15151	110000.00	Lavender Bay	AU
16	7	15031	10000.00	Malabar	AU
17	7	15426	10000.00	Malabar	AU
18	8	15131	80000.00	Matraville	AU
19	8	15204	90000.00	Matraville	AU

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (74) | AdventureWorksDW2017 | 00:00:00 | 499 rows

Sample JSON solution with output (499 Objects returned)

```
USE AdventureWorksDW2017;
SELECT MAX(C.GeographyKey) AS [Max Geo],
       C.CustomerKey,
       C.YearlyIncome AS [Max Income],
       D.City,
       D.CountryRegionCode
FROM dbo.DimCustomer AS C
    INNER JOIN dbo.DimGeography AS D
        ON C.GeographyKey = D.GeographyKey
WHERE C.CustomerKey > 15000
      AND C.CustomerKey < 15500
GROUP BY C.CustomerKey,
```

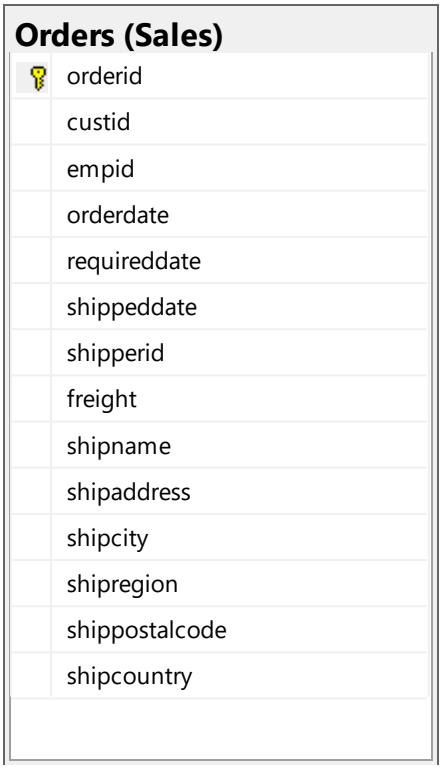
```
        C.YearlyIncome,
        D.City,
        D.CountryRegionCode
FOR JSON PATH, ROOT('CreditCardID'), INCLUDE_NULL_VALUES;
```

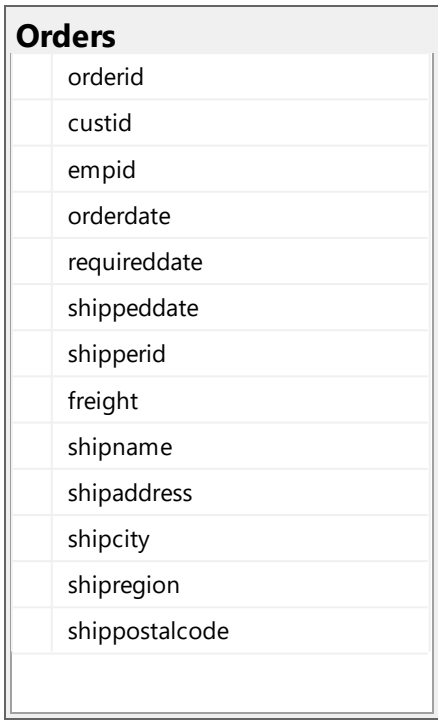


Medium Query 5

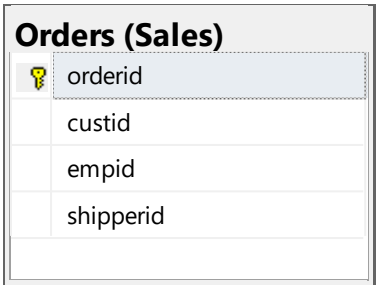
```
Performs an left join and returns every order made by customer 90

Use TSQLV6 database and Sales.Orders and dbo.Orders
```

Standard view





Key view



Columns from tables with order by

Table Name	Column Names	Order By
Dbo.Order	Orderid custid	ASC ASC

	freight	ASC
Sales.Orders	shippeddate	ASC

Sample Relational solution and output (Returns 7 rows)

```
USE TSQLV4;
SELECT O.orderid,
       O.custid,
       SUM(O.freight) AS [Summed Freight],
       S.shippeddate
FROM dbo.Orders AS O
    LEFT JOIN Sales.Orders AS S
        ON S.custid = O.custid
           AND S.orderid = O.orderid
WHERE O.custid = 90
GROUP BY O.orderid,
         O.custid,
         S.shippeddate
ORDER BY [Summed Freight];
--FOR JSON PATH, ROOT('custid'), INCLUDE_NULL_VALUES;
```

Results | Messages

	orderid	custid	Summed Freight	shippeddate
1	10615	90	0.75	2015-08-06
2	11005	90	0.75	2016-04-10
3	10873	90	0.82	2016-02-09
4	10879	90	8.50	2016-02-12
5	10695	90	16.72	2015-10-14
6	10673	90	22.76	2015-09-19
7	10910	90	38.11	2016-03-04

Sample JSON solution and output (Returns 7 objects)

```
USE TSQLV4;
SELECT O.orderid,
       O.custid,
       SUM(O.freight) AS [Summed Freight],
       S.shippeddate
FROM dbo.Orders AS O
    LEFT JOIN Sales.Orders AS S
        ON S.custid = O.custid
           AND S.orderid = O.orderid
WHERE O.custid = 90
GROUP BY O.orderid,
         O.custid,
         S.shippeddate
ORDER BY [Summed Freight]
FOR JSON PATH, ROOT('custid'), INCLUDE_NULL_VALUES;
```

ROOT
- custid: [Array]
  - [0]: [Object]
  - [1]: [Object]
  - [2]: [Object]
  - [3]: [Object]
  - [4]: [Object]
  - [5]: [Object]
  - [6]: [Object]


```
        }, {
            "orderid": 11005,
            "custid": 90,
            "Summed Freight": 0.7500,
            "shippeddate": "2016-04-10"
        }, {
            "orderid": 10873,
            "custid": 90,
            "Summed Freight": 0.8200,
            "shippeddate": "2016-02-09"
        }, {
            "orderid": 10879,
            "custid": 90,
            "Summed Freight": 8.5000,
            "shippeddate": "2016-02-12"
        }, {
            "orderid": 10695,
            "custid": 90,
            "Summed Freight": 16.7200,
            "shippeddate": "2015-10-14"
        }, {
            "orderid": 10673,
            "custid": 90,
            "Summed Freight": 22.7600,
            "shippeddate": "2015-09-19"
        }, {
            "orderid": 10910,
            "custid": 90,
            "Summed Freight": 38.1100,
            "shippeddate": "2016-03-04"
        }
    ]
```

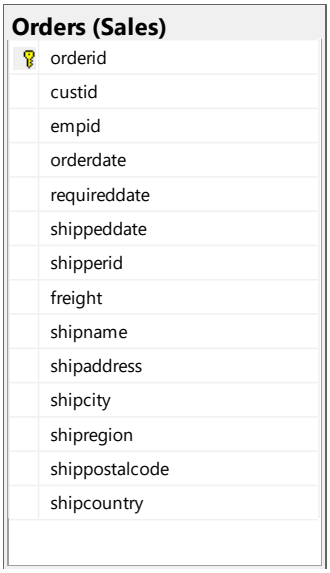

Normal text file | length : 1,093 lines : 39 | Ln : 19 Col : 26 Sel : 0 | 0 | Windows (CR LF) | UTF-8

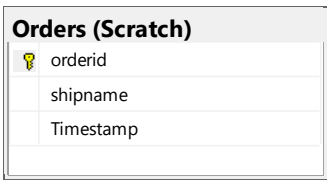
Medium Query 6

Performs an inner join and returns order id, customer id, the maximum freight for that order with its destination

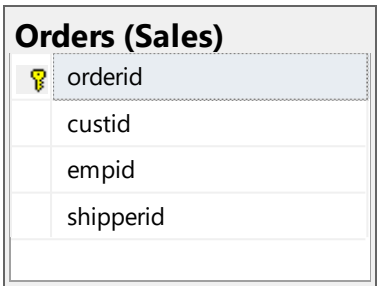
Use TSQLV6 database with Sales.Orders and Scratch.Orders tables

Standard view





Key view



Columns from tables with order by

Table Name	Column Names	Order By
Sales.Orders	Orderid custid freight	ASC ASC ASC
Scratch.Orders	shipto	ASC

Sample relational solution and output (Returns 48 Rows)

```
USE TSQLV4;
SELECT O.orderid,
       O.custid,
```

```
       MAX(O.freight) AS [MAX Freight],
       S.shipname AS [Ship To]
FROM Sales.Orders AS O
    INNER JOIN Scratch.Orders AS S
        ON O.orderid = S.orderid
WHERE O.orderid IS NOT NULL
GROUP BY O.orderid,
         O.custid,
         S.shipname
ORDER BY O.orderid;
--FOR JSON PATH, ROOT('orderid'), INCLUDE_NULL_VALUES;
```

	orderid	custid	MAX Freight	Ship To
1	10256	88	13.97	Ship to 71-C
2	10269	89	4.56	Ship to 71-C
3	10282	69	12.69	Ship to 89-B
4	10295	85	1.15	Ship to 68-A
5	10308	2	1.61	Ship to 55-B
6	10321	38	3.43	Ship to 76-B
7	10334	84	8.56	Destination YPUYI
8	10347	21	3.10	Destination GGQIR
9	10360	7	131.70	Ship to 76-A
10	10373	37	124.12	Ship to 47-C
11	10386	21	13.99	Destination VYOBK
12	10399	83	27.36	Destination YUJRD
13	10412	87	3.77	Destination EVHYA
14	10425	41	7.93	Ship to 74-B
15	10438	79	8.24	Ship to 91-B
16	10451	63	189.09	Destination LPHSI
17	10464	28	89.00	Ship to 86-C
18	10477	60	13.02	Destination FRCGJ
19	10490	35	210.19	Destination XJIBQ

Query executed successfully. localhost,12001 (15.0 RTM) sa (77) TSQLV4 00:00:00 48 rows

Sample JSON solution and output (Returns 48 Objects)

```
USE TSQLV4;
SELECT O.orderid,
       O.custid,
       MAX(O.freight) AS [MAX Freight],
       S.shipname AS [Ship To]
FROM Sales.Orders AS O
    INNER JOIN Scratch.Orders AS S
        ON O.orderid = S.orderid
WHERE O.orderid IS NOT NULL
GROUP BY O.orderid,
         O.custid,
         S.shipname
ORDER BY O.orderid
FOR JSON PATH, ROOT('custid'), INCLUDE_NULL_VALUES;
```

```
[16]: [Object]
[17]: [Object]
[18]: [Object]
[19]: [Object]
[20]: [Object]
[21]: [Object]
[22]: [Object]
[23]: [Object]
[24]: [Object]
[25]: [Object]
[26]: [Object]
[27]: [Object]
[28]: [Object]
[29]: [Object]
[30]: [Object]
[31]: [Object]
[32]: [Object]
[33]: [Object]
[34]: [Object]
[35]: [Object]
[36]: [Object]
[37]: [Object]
[38]: [Object]
[39]: [Object]
[40]: [Object]
[41]: [Object]
[42]: [Object]
[43]: [Object]
[44]: [Object]
[45]: [Object]
[46]: [Object]
[47]: [Object]
```

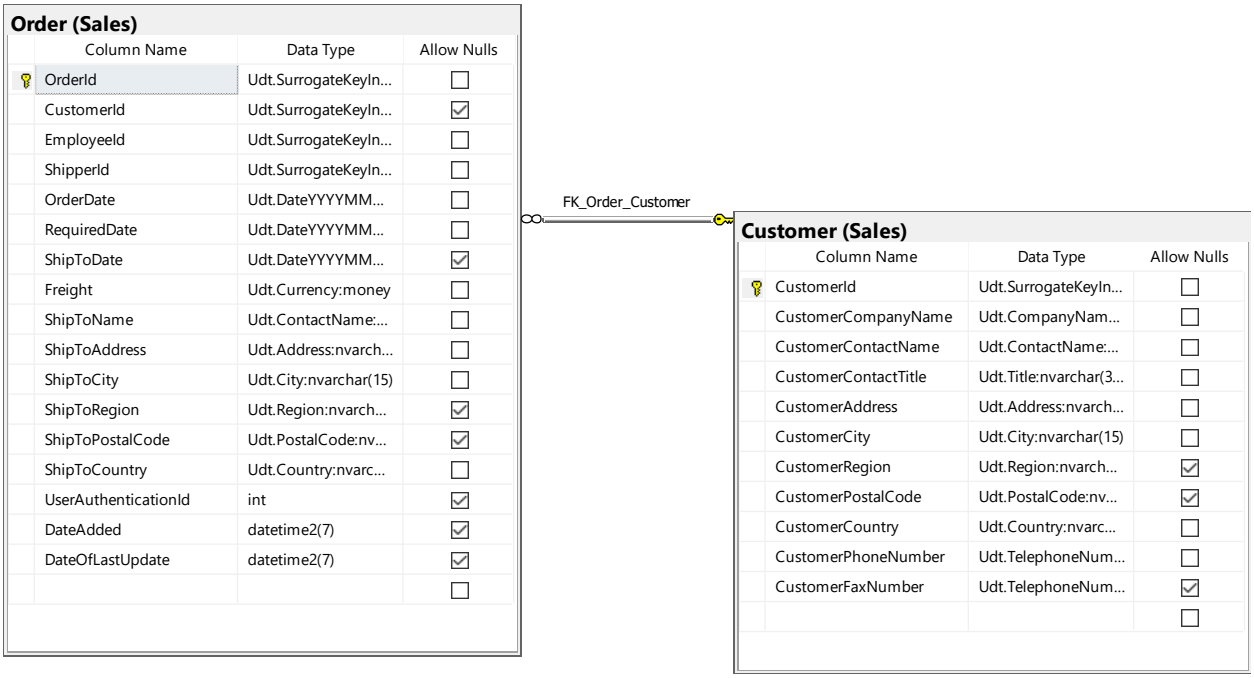
```
        }, {
            "orderid": 10802,
            "custid": 73,
            "MAX Freight": 257.2600,
            "Ship To": "Destination RVDMF"
        }, {
            "orderid": 10815,
            "custid": 71,
            "MAX Freight": 14.6200,
            "Ship To": "Ship to 58-B"
        }, {
            "orderid": 10828,
            "custid": 64,
            "MAX Freight": 90.8500,
            "Ship To": "Ship to 73-A"
        }, {
            "orderid": 10841,
            "custid": 76,
            "MAX Freight": 424.3000,
            "Ship To": "Ship to 68-A"
        }, {
            "orderid": 10854,
            "custid": 20,
            "MAX Freight": 100.2200,
            "Ship To": "Ship to 9-A"
        }, {
            "orderid": 10867,
            "custid": 48,
            "MAX Freight": 1.9300,
            "Ship To": "Ship to 65-A"
        }
    ]
```

Normal text file | length : 7,226 lines : 244 | Ln : 230 Col : 37 Sel : 0 | 0 | Winc

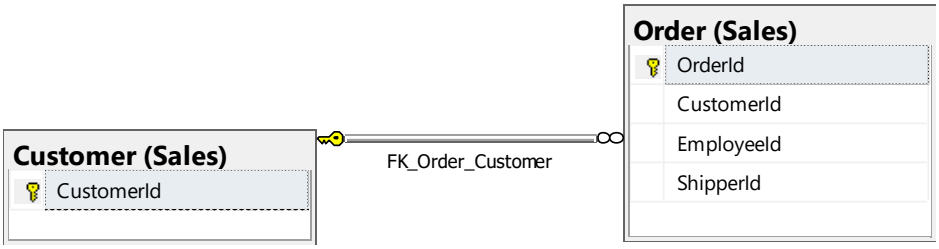
Medium Query 7

```
Performs an inner join and returns customer's data with minimum freight where the year
2014 and customer id > 60 which is ordered by the customer id

Use NorthwindsTSQLV6 Database with Sales.Customer and Sales.Order tables
```

Standard view



Key view



Columns from tables with order by

Table Name	Column Names	Order By
Sales.Orders	CustomerID CustomerCompanyName CustomerPostalCode	ASC ASC ASC
Sales.[Order]	EmployeeID	ASC

Sample relational solution and output (Returns 57 rows)

```
USE Northwinds2020TSQLV6;
SELECT C.CustomerId,
       C.CustomerCompanyName,
       C.CustomerPostalCode,
       E.EmployeeId,
       MIN(E.Freight) AS [MIN Freight]
FROM Sales.Customer AS C
    INNER JOIN Sales.[Order] AS E
        ON E.CustomerId = C.CustomerId
WHERE YEAR(E.OrderDate) = '2014'
      AND E.CustomerId > 60
GROUP BY C.CustomerId,
         C.CustomerCompanyName,
         C.CustomerPostalCode,
         E.EmployeeId
ORDER BY C.CustomerId;
--FOR JSON PATH, ROOT('customerid'), INCLUDE_NULL_VALUES;
```

Results | Messages

	CustomerId	CustomerCompanyName	CustomerPostalCode	EmployeeId	MIN Freight
1	61	Customer WULWD	10115	2	45.03
2	61	Customer WULWD	10115	4	3.05
3	61	Customer WULWD	10115	6	6.40
4	62	Customer WFIZJ	10102	5	890.78
5	63	Customer IRRVL	10126	1	76.83
6	63	Customer IRRVL	10126	2	1.96
7	63	Customer IRRVL	10126	3	76.07
8	63	Customer IRRVL	10126	8	229.24
9	65	Customer NYUHS	10109	1	74.16
10	65	Customer NYUHS	10109	3	142.08
11	65	Customer NYUHS	10109	4	147.26
12	65	Customer NYUHS	10109	6	98.03
13	65	Customer NYUHS	10109	8	48.29
14	66	Customer LHANT	10038	4	7.45
15	67	Customer QVEPD	10052	4	29.76
16	67	Customer QVEPD	10052	8	12.76
17	68	Customer CCKOT	10122	9	148.33
18	69	Customer SIUIH	10071	1	7.56
19	69	Customer SIUIH	10071	4	2.94

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (76) | Northwinds2020TSQLV6 | 00:00:00 | 57 rows

Sample JSON solution and output (Returns 57 Objects)

```
USE Northwinds2020TSQLV6;
SELECT C.CustomerId,
       C.CustomerCompanyName,
       C.CustomerPostalCode,
       E.EmployeeId,
       MIN(E.Freight) AS [MIN Freight]
FROM Sales.Customer AS C
    INNER JOIN Sales.[Order] AS E
        ON E.CustomerId = C.CustomerId
WHERE YEAR(E.OrderDate) = '2014'
      AND E.CustomerId > 60
GROUP BY C.CustomerId,
         C.CustomerCompanyName,
         C.CustomerPostalCode,
         E.EmployeeId
ORDER BY C.CustomerId
FOR JSON PATH, ROOT('custid'), INCLUDE_NULL_VALUES;
```

```
[25]: [Object]
[26]: [Object]
[27]: [Object]
[28]: [Object]
[29]: [Object]
[30]: [Object]
[31]: [Object]
[32]: [Object]
[33]: [Object]
[34]: [Object]
[35]: [Object]
[36]: [Object]
[37]: [Object]
[38]: [Object]
[39]: [Object]
[40]: [Object]
[41]: [Object]
[42]: [Object]
[43]: [Object]
[44]: [Object]
[45]: [Object]
[46]: [Object]
[47]: [Object]
[48]: [Object]
[49]: [Object]
[50]: [Object]
[51]: [Object]
[52]: [Object]
[53]: [Object]
[54]: [Object]
[55]: [Object]
[56]: [Object]
```

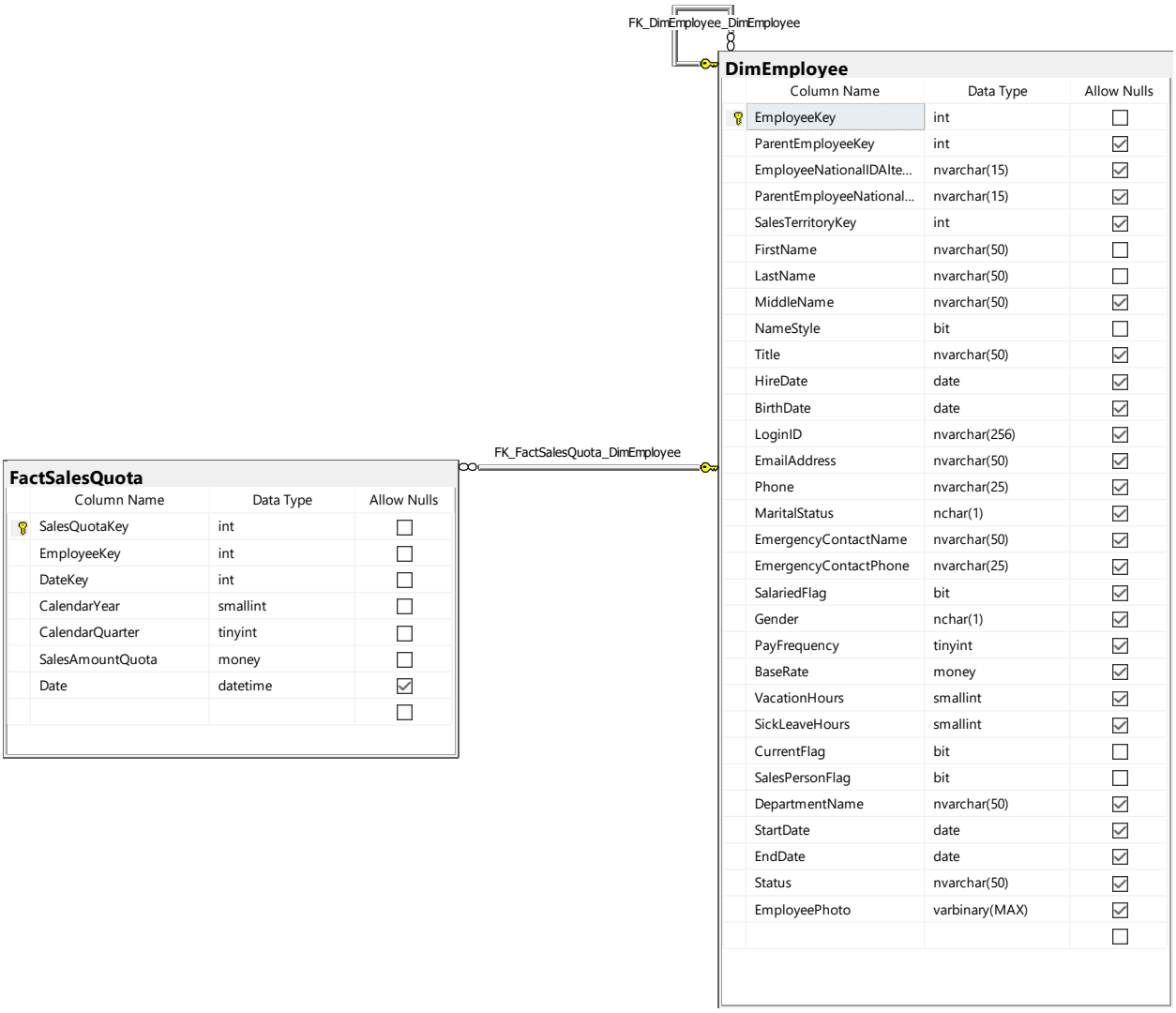
```
    }, {
        "CustomerId": 87,
        "CustomerCompanyName": "Customer ZHYOS",
        "CustomerPostalCode": "10045",
        "EmployeeId": 5,
        "MIN Freight": 0.5900
    }, {
        "CustomerId": 88,
        "CustomerCompanyName": "Customer SRQVM",
        "CustomerPostalCode": "10084",
        "EmployeeId": 3,
        "MIN Freight": 13.9700
    }, {
        "CustomerId": 89,
        "CustomerCompanyName": "Customer YBQTI",
        "CustomerPostalCode": "10049",
        "EmployeeId": 4,
        "MIN Freight": 23.2900
    }, {
        "CustomerId": 89,
        "CustomerCompanyName": "Customer YBQTI",
        "CustomerPostalCode": "10049",
        "EmployeeId": 5,
        "MIN Freight": 4.5600
    }, {
        "CustomerId": 91,
        "CustomerCompanyName": "Customer CCFIZ",
        "CustomerPostalCode": "10068",
        "EmployeeId": 1,
        "MIN Freight": 3.9400
    }
]
```

Medium Query 8

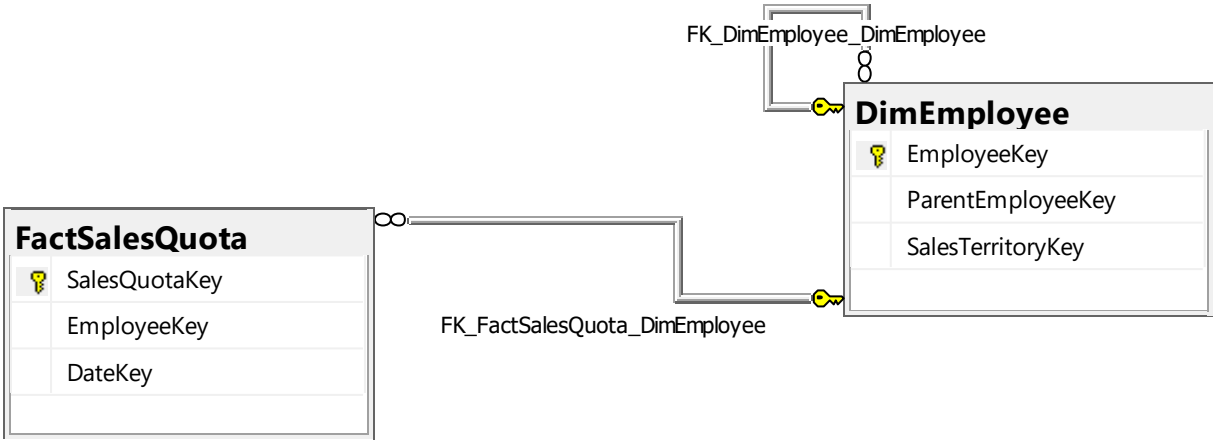
Performs an inner join and returns the employee key, quota, the maximum base pay where the employee was hired after 2011

Use AdventureWorksDW2017 database with dbo.DimEmployee and dbo.FactSalesQuota tables

Standard view



Key view



Columns from tables with order by

Table Name	Column Names	Order By
Dbo.FactSalesQuota	EmployeeKey SalesAmountQuota	ASC ASC
Dbo.DimEmployee	HireDate BaseRate	ASC ASC

Sample relational solution and output (Returns 19 rows)

```
USE AdventureWorksDW2017;
SELECT A.EmployeeKey,
       A.SalesAmountQuota,
       B.HireDate,
       MAX(B.BaseRate) AS [Maximum Rate]
FROM dbo.FactSalesQuota AS A
    INNER JOIN dbo.DimEmployee AS B
        ON B.EmployeeKey = A.EmployeeKey
WHERE YEAR(B.HireDate) > 2011
GROUP BY A.EmployeeKey,
         A.SalesAmountQuota,
         B.HireDate
ORDER BY B.HireDate;
--FOR JSON PATH, ROOT('EmployeeKey'), INCLUDE_NULL_VALUES;
```

Results | Messages

	EmployeeKey	SalesAmountQuota	HireDate	Maximum Rate
1	293	281000.00	2012-04-30	23.0769
2	293	304000.00	2012-04-30	23.0769
3	293	321000.00	2012-04-30	23.0769
4	293	380000.00	2012-04-30	23.0769
5	293	497000.00	2012-04-30	23.0769
6	293	516000.00	2012-04-30	23.0769
7	293	454000.00	2012-04-30	23.0769
8	294	7000.00	2012-10-12	48.101
9	294	26000.00	2012-10-12	48.101
10	294	40000.00	2012-10-12	48.101
11	294	132000.00	2012-10-12	48.101
12	295	366000.00	2012-12-28	23.0769
13	295	566000.00	2012-12-28	23.0769
14	295	627000.00	2012-12-28	23.0769
15	295	728000.00	2012-12-28	23.0769
16	296	389000.00	2012-12-28	23.0769
17	296	399000.00	2012-12-28	23.0769
18	296	421000.00	2012-12-28	23.0769
19	296	478000.00	2012-12-28	23.0769

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (71) | AdventureWorksDW2017 | 00:00:00 | 19 rows

Sample JSON solution and output (Returns 19 Objects)

```
USE AdventureWorksDW2017;
SELECT A.EmployeeKey,
       A.SalesAmountQuota,
       B.HireDate,
       MAX(B.BaseRate) AS [Maximum Rate]
FROM dbo.FactSalesQuota AS A
    INNER JOIN dbo.DimEmployee AS B
        ON B.EmployeeKey = A.EmployeeKey
WHERE YEAR(B.HireDate) > 2011
GROUP BY A.EmployeeKey,
         A.SalesAmountQuota,
         B.HireDate
ORDER BY B.HireDate
FOR JSON PATH, ROOT('EmployeeKey'), INCLUDE_NULL_VALUES;
```

ROOT
- EmployeeKey: [Array]
  - [0]: [Object]
  - [1]: [Object]
  - [2]: [Object]
  - [3]: [Object]
  - [4]: [Object]
  - [5]: [Object]
  - [6]: [Object]
  - [7]: [Object]
  - [8]: [Object]
  - [9]: [Object]
  - [10]: [Object]
  - [11]: [Object]
  - [12]: [Object]
  - [13]: [Object]
  - [14]: [Object]
  - [15]: [Object]
  - [16]: [Object]
  - [17]: [Object]
  - [18]: [Object]

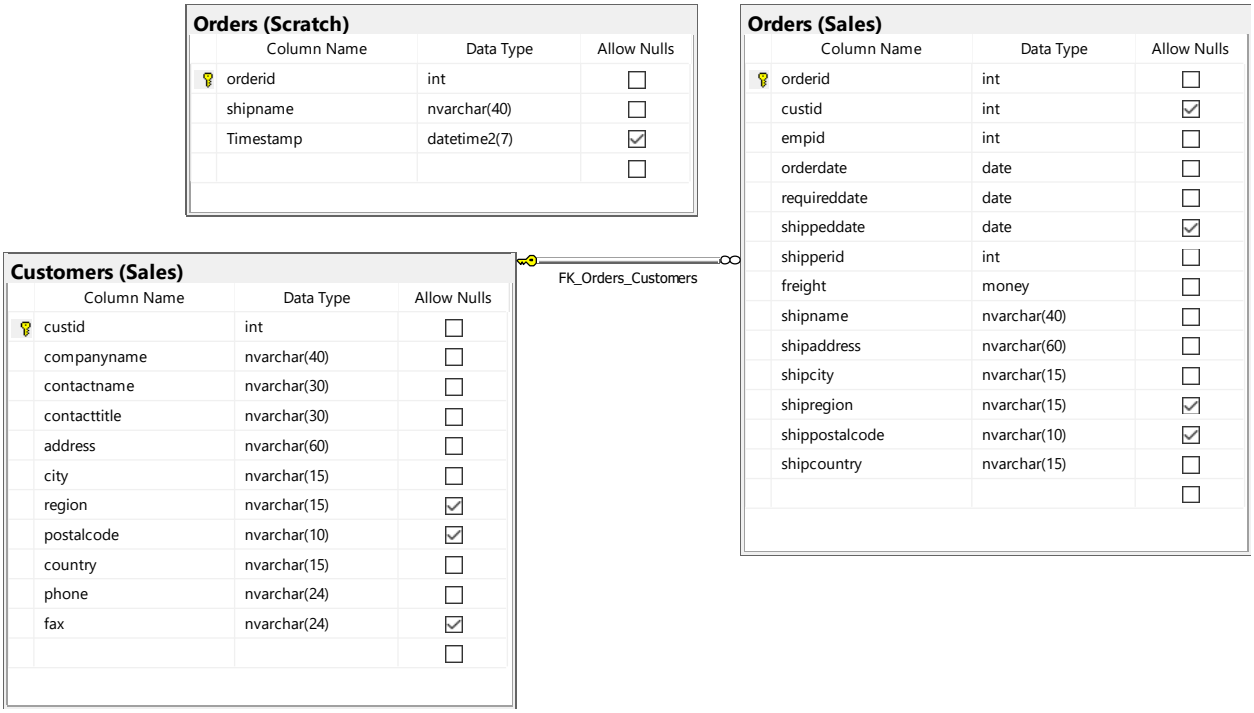

```
        }, {
            "EmployeeKey": 295,
            "SalesAmountQuota": 627000.0000,
            "HireDate": "2012-12-28",
            "Maximum Rate": 23.0769
        }, {
            "EmployeeKey": 295,
            "SalesAmountQuota": 728000.0000,
            "HireDate": "2012-12-28",
            "Maximum Rate": 23.0769
        }, {
            "EmployeeKey": 296,
            "SalesAmountQuota": 389000.0000,
            "HireDate": "2012-12-28",
            "Maximum Rate": 23.0769
        }, {
            "EmployeeKey": 296,
            "SalesAmountQuota": 399000.0000,
            "HireDate": "2012-12-28",
            "Maximum Rate": 23.0769
        }, {
            "EmployeeKey": 296,
            "SalesAmountQuota": 421000.0000,
            "HireDate": "2012-12-28",
            "Maximum Rate": 23.0769
        }, {
            "EmployeeKey": 296,
            "SalesAmountQuota": 478000.0000,
            "HireDate": "2012-12-28",
            "Maximum Rate": 23.0769
        }
    ]
```

Complex Query 1

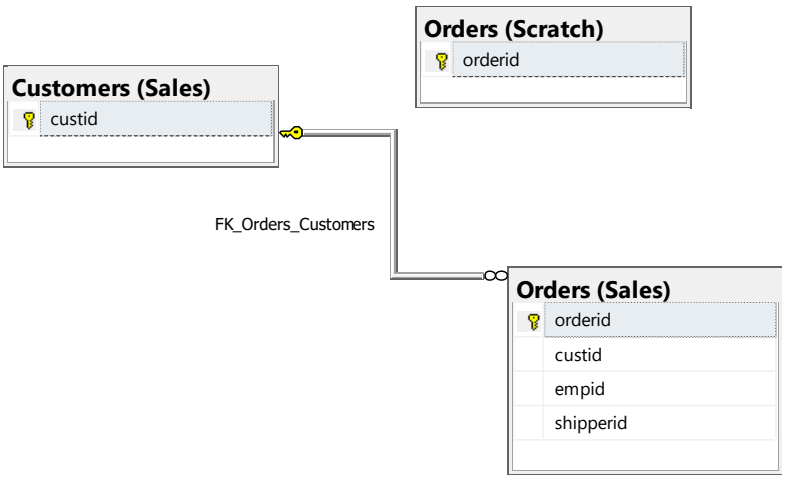
Scalar function that returns the maximum customer id. Query performs a inner join twice and returns the max customer id, customer's company name and the orderdate

Use TSQLV4 database with Sales.Customers, Scratch.Orders, and Sales.Orders tables

Standard view



Key view



Columns from tables with order by

Table Name	Column Names	Order By
Sales.Customers	CompanyName Custid	ASC ASC
Sales.Orders	orderdate	ASC
Scratch.Orders	Custid	ASC

Sample relational solution with output (Returns 48 rows)

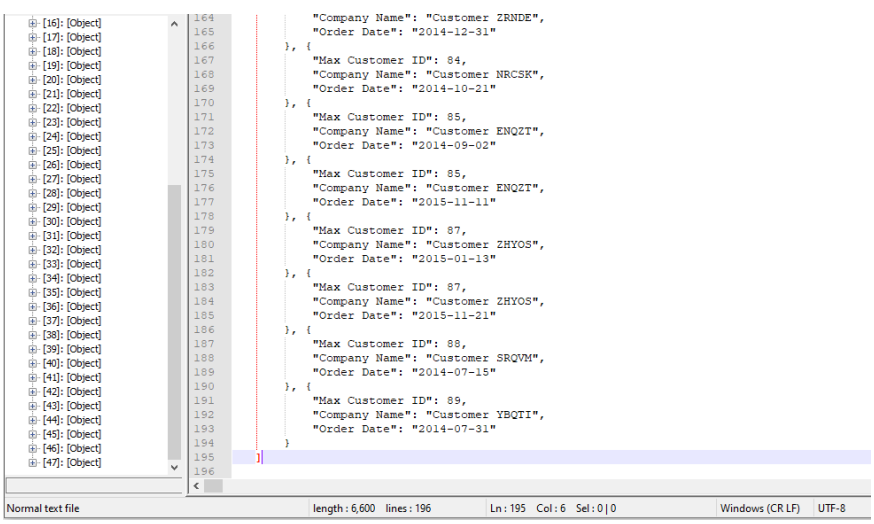
```
USE TSQLV4;
DROP FUNCTION IF EXISTS dbo.CustomerInfo;
GO
CREATE FUNCTION dbo.CustomerInfo
(
    @Custid AS INT
)
RETURNS INT
AS
BEGIN

    RETURN
    (
        SELECT MAX(@Custid) FROM Sales.Customers
    );

END;
GO

USE TSQLV4;
SELECT dbo.CustomerInfo(A.custid) AS [Max Customer ID],
       A.companyname AS [Company Name],
       B.orderdate AS [Order Date]
FROM Sales.Customers AS A
    INNER JOIN Sales.Orders AS B
        ON A.custid = B.custid
    INNER JOIN Scratch.Orders AS C
        ON B.orderid = C.orderid
GROUP BY dbo.CustomerInfo(A.custid),
         A.companyname,
         B.orderdate,
         A.custid
ORDER BY A.custid;
```

	Max Customer ID	Company Name	Order Date
1	2	Customer MLTDN	2014-09-18
2	5	Customer HGVLZ	2015-09-17
3	7	Customer QXVLA	2014-11-22
4	20	Customer THHDP	2015-08-15
5	20	Customer THHDP	2015-10-09
6	20	Customer THHDP	2015-12-15
7	20	Customer THHDP	2016-01-27
8	21	Customer KIDPX	2014-11-06
9	21	Customer KIDPX	2014-12-18
10	21	Customer KIDPX	2015-06-26
11	23	Customer WVFAF	2015-12-03
12	23	Customer WVFAF	2015-12-22
13	28	Customer XYUFB	2015-03-04
14	29	Customer MDLWA	2015-06-13
15	31	Customer YJCBX	2015-09-29

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (73) | TSQLV4 | 00:00:00 | 48 rows

Sample JSON solution with output (Returns 48 Objects)

```sql
USE TSQLV4;
DROP FUNCTION IF EXISTS dbo.CustomerInfo;
GO
CREATE FUNCTION dbo.CustomerInfo
(
    @Custid AS INT
)
RETURNS INT
AS
BEGIN

    RETURN
    (
        SELECT MAX(@Custid) FROM Sales.Customers
    );

END;
GO

USE TSQLV4;
SELECT dbo.CustomerInfo(A.custid) AS [Max Customer ID],
       A.companyname AS [Company Name],
       B.orderdate AS [Order Date]
FROM Sales.Customers AS A
    INNER JOIN Sales.Orders AS B
        ON A.custid = B.custid
    INNER JOIN Scratch.Orders AS C
        ON B.orderid = C.orderid
GROUP BY dbo.CustomerInfo(A.custid),
         A.companyname,
         B.orderdate,
         A.custid
ORDER BY A.custid
FOR JSON PATH, ROOT('orderid'), INCLUDE_NULL_VALUES;
```

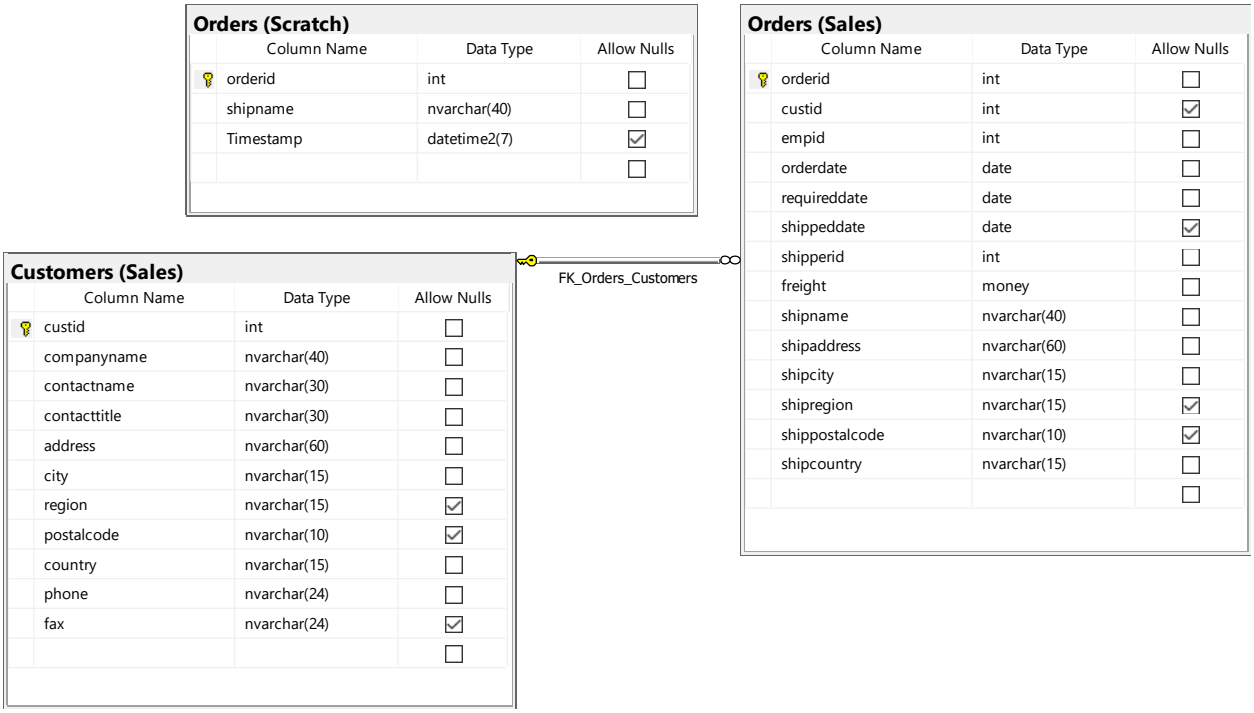


Complex Query 2

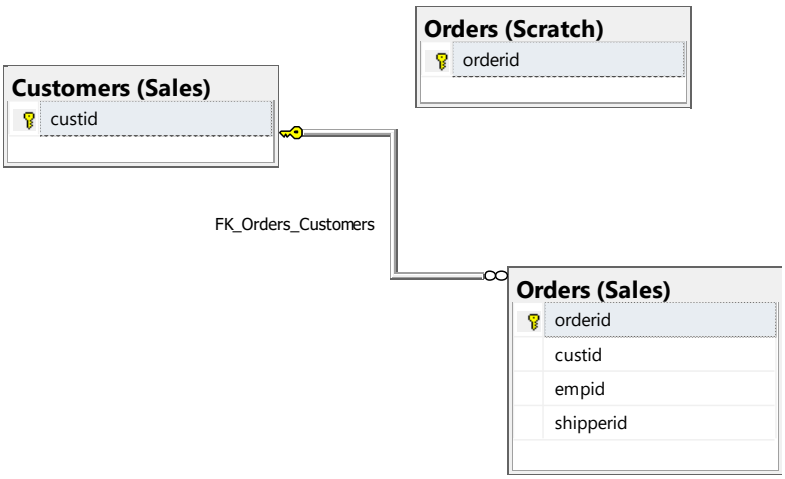
Scalar function that returns the minimum customer id. Query performs a inner join twice and returns the min customer id, customer's company name and the orderdate

Use TSQLV4 database with Sales.Customers, Scratch.Orders, and Sales.Orders tables

Standard view



Key view



Columns from tables with order by

Table Name	Column Names	Order By
Sales.Customers	CompanyName Custid	ASC ASC
Sales.Orders	orderdate	ASC
Scratch.Orders	Custid	ASC

Sample relational solution with output (Returns 48 rows)

```
USE TSQLV4;
DROP FUNCTION IF EXISTS dbo.MinCustomerInfo;
GO
CREATE FUNCTION dbo.MinCustomerInfo
(
    @Custid AS INT
)
RETURNS INT
AS
BEGIN

    RETURN
    (
        SELECT MIN(@Custid) FROM Sales.Customers
    );

END;
GO

USE TSQLV4;
SELECT dbo.MinCustomerInfo(A.custid) AS [Min Customer ID],
       A.companyname AS [Company Name],
       B.orderdate AS [Order Date]
FROM Sales.Customers AS A
    INNER JOIN Sales.Orders AS B
        ON A.custid = B.custid
    INNER JOIN Scratch.Orders AS C
        ON B.orderid = C.orderid
GROUP BY dbo.MinCustomerInfo(A.custid),
         A.companyname,
         B.orderdate,
         A.custid
ORDER BY A.custid;
--FOR JSON PATH, ROOT('orderid'), INCLUDE_NULL_VALUES;
```

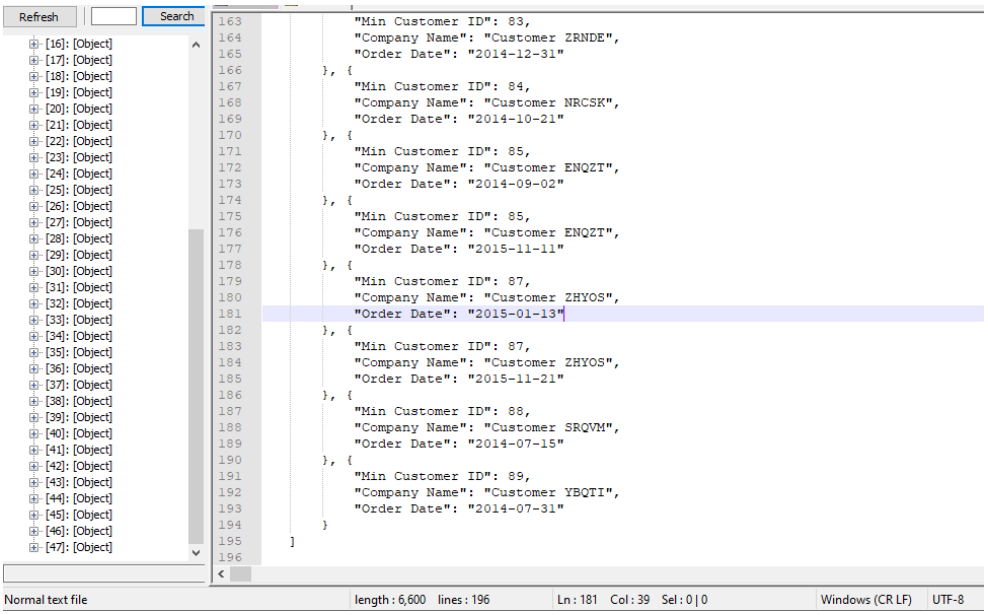
Results | Messages

	Min Customer ID	Company Name	Order Date
1	2	Customer MLTDN	2014-09-18
2	5	Customer HGVLZ	2015-09-17
3	7	Customer QXVLA	2014-11-22
4	20	Customer THHDP	2015-08-15
5	20	Customer THHDP	2015-10-09
6	20	Customer THHDP	2015-12-15
7	20	Customer THHDP	2016-01-27
8	21	Customer KIDPX	2014-11-06
9	21	Customer KIDPX	2014-12-18
10	21	Customer KIDPX	2015-06-26
11	23	Customer WVFAF	2015-12-03
12	23	Customer WVFAF	2015-12-22
13	28	Customer XYUFB	2015-03-04
14	29	Customer MDLWA	2015-06-13
15	31	Customer YJCBX	2015-09-29
16	35	Customer UMTLM	2015-03-31
17	37	Customer FRXZL	2014-12-05
18	37	Customer FRXZL	2015-04-11
19	37	Customer FRXZL	2015-04-24

Query executed successfully. localhost,12001 (15.0 RTM) | sa (59) | TSQLV4 | 00:00:00 | 48 rows

Sample JSON solution with output (Returns 48 Objects)

```
USE TSQLV4;
DROP FUNCTION IF EXISTS dbo.MinCustomerInfo;
GO
CREATE FUNCTION dbo.MinCustomerInfo
(
    @Custid AS INT
)
RETURNS INT
AS
BEGIN

    RETURN
    (
        SELECT MIN(@Custid) FROM Sales.Customers
    );

END;
GO

USE TSQLV4;
SELECT dbo.MinCustomerInfo(A.custid) AS [Min Customer ID],
       A.companyname AS [Company Name],
       B.orderdate AS [Order Date]
FROM Sales.Customers AS A
    INNER JOIN Sales.Orders AS B
        ON A.custid = B.custid
    INNER JOIN Scratch.Orders AS C
        ON B.orderid = C.orderid
GROUP BY dbo.MinCustomerInfo(A.custid),
         A.companyname,
         B.orderdate,
         A.custid
ORDER BY A.custid
FOR JSON PATH, ROOT('orderid'), INCLUDE_NULL_VALUES;
```



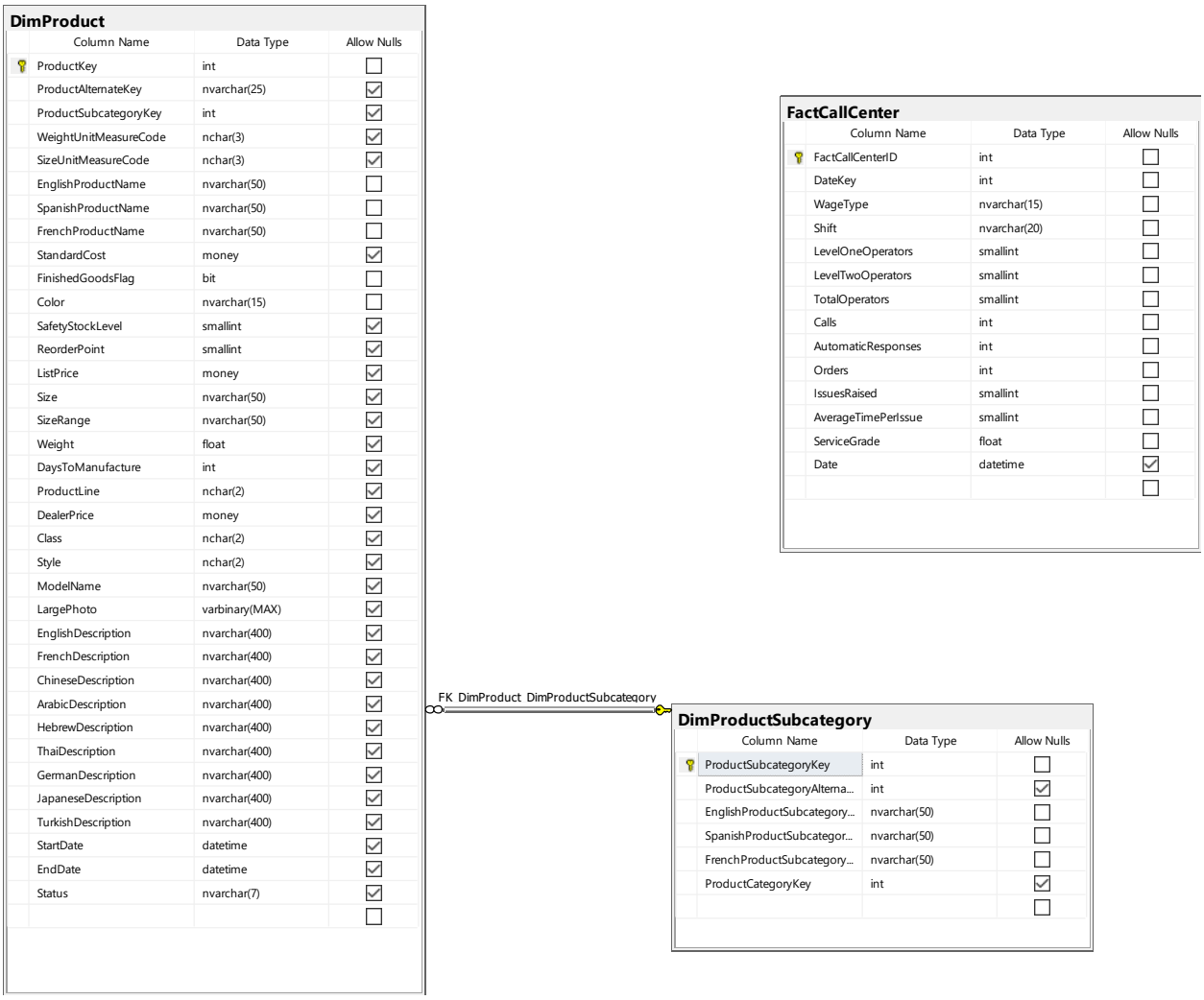

```
            "Min Customer ID": 83,
            "Company Name": "Customer ZRNDE",
            "Order Date": "2014-12-31"
        }, {
            "Min Customer ID": 84,
            "Company Name": "Customer NRCSK",
            "Order Date": "2014-10-21"
        }, {
            "Min Customer ID": 85,
            "Company Name": "Customer ENQZT",
            "Order Date": "2014-09-02"
        }, {
            "Min Customer ID": 85,
            "Company Name": "Customer ENQZT",
            "Order Date": "2015-11-11"
        }, {
            "Min Customer ID": 87,
            "Company Name": "Customer ZHYOS",
            "Order Date": "2015-01-13"
        }, {
            "Min Customer ID": 87,
            "Company Name": "Customer ZHYOS",
            "Order Date": "2015-11-21"
        }, {
            "Min Customer ID": 88,
            "Company Name": "Customer SRQVM",
            "Order Date": "2014-07-15"
        }, {
            "Min Customer ID": 89,
            "Company Name": "Customer YBQTI",
            "Order Date": "2014-07-31"
        }
    ]
```

Complex Query 3

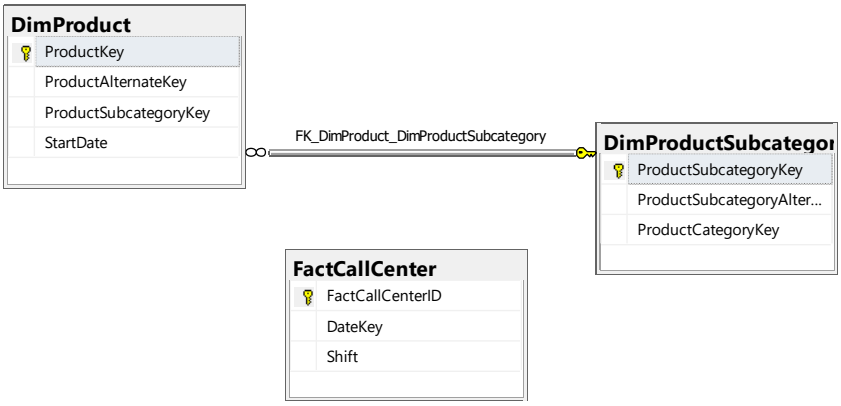
Scalar function returns the minimum number of issues from the company. The query returns the English product name and its category with its size, the average number of calls they get where issues are less than 3

Use AdventureWorksDW2017 database and dbo.DimProduct, dbo.DimProductSubcategory, dbo.FactCallCenter tables

Standard view



Key view



Columns from tables with order by

Table Name	Column Names	Order By
Dbo.DimProduct	EnglishProductName Size	ASC ASC
Dbo.DimProductSubcategory	EnglishProductSubcategoryName	ASC ASC
Dbo.FactCallCenter	IssuesRaised	ASC

Sample relational solution with output (Returns 633 rows)

```
USE AdventureWorksDW2017;
DROP FUNCTION IF EXISTS dbo.MinProductIssueFunction;
GO
CREATE FUNCTION dbo.MinProductIssueFunction
(
    @issues AS SMALLINT
)
RETURNS SMALLINT
AS
BEGIN
    RETURN
    (
        SELECT MIN(@issues) FROM dbo.FactCallCenter
    );
END;
GO

USE AdventureWorksDW2017;
SELECT A.EnglishProductName,
       A.Size,
       B.EnglishProductSubcategoryName,
       dbo.MinProductIssueFunction(C.IssuesRaised) AS [Number of Issues],
       AVG(C.Calls) AS [Average Calls]
FROM dbo.DimProduct AS A
    INNER JOIN dbo.DimProductSubcategory AS B
```

```
        ON B.ProductSubcategoryKey = A.ProductSubcategoryKey
    INNER JOIN dbo.FactCallCenter AS C
        ON A.Size IS NOT NULL
           AND dbo.MinProductIssueFunction(C.IssuesRaised) < 3
GROUP BY dbo.MinProductIssueFunction(C.IssuesRaised),
         A.EnglishProductName,
         A.Size,
         B.EnglishProductSubcategoryName
ORDER BY [Number of Issues];
--FOR JSON PATH, ROOT('orderid'), INCLUDE_NULL_VALUES;
```

	EnglishProductName	Size	EnglishProductSubcategoryName	Number of Issues	Average Calls
1	Touring-2000 Blue, 60	60	Touring Bikes	0	359
2	Mountain-500 Silver, 40	40	Mountain Bikes	0	359
3	Road-150 Red, 52	52	Road Bikes	0	359
4	Men's Sports Shorts, XL	XL	Shorts	0	359
5	Men's Bib-Shorts, M	M	Bib-Shorts	0	359
6	Classic Vest, M	M	Vests	0	359
7	LL Touring Frame - Yellow, 54	54	Touring Frames	0	359
8	ML Road Frame-W - Yellow, 44	44	Road Frames	0	359
9	LL Mountain Frame - Silver, 44	44	Mountain Frames	0	359
10	LL Road Frame - Red, 58	58	Road Frames	0	359
11	Mountain-400-W Silver, 40	40	Mountain Bikes	0	359
12	Half-Finger Gloves, S	S	Gloves	0	359
13	Mountain-200 Black, 38	38	Mountain Bikes	0	359
14	Men's Sports Shorts, S	S	Shorts	0	359
15	Mountain-100 Black, 44	44	Mountain Bikes	0	359
16	Mountain-200 Black, 46	46	Mountain Bikes	0	359
17	ML Mountain Frame-W - Silver, 46	46	Mountain Frames	0	359
18	Road-250 Black, 48	48	Road Bikes	0	359
19	LL Road Frame - Black, 48	48	Road Frames	0	359

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (61) | AdventureWorksDW2017 | 00:00:05 | 633 rows

Sample JSON solution with output (Returns 633 Objects)

```
USE AdventureWorksDW2017;
DROP FUNCTION IF EXISTS dbo.MinProductIssueFunction;
GO
CREATE FUNCTION dbo.MinProductIssueFunction
(
    @issues AS SMALLINT
)
RETURNS SMALLINT
AS
BEGIN
    RETURN
    (
        SELECT MIN(@issues) FROM dbo.FactCallCenter
    );
END;
GO

USE AdventureWorksDW2017;
SELECT A.EnglishProductName,
       A.Size,
       B.EnglishProductSubcategoryName,
       dbo.MinProductIssueFunction(C.IssuesRaised) AS [Number of Issues],
       AVG(C.Calls) AS [Average Calls]
FROM dbo.DimProduct AS A
    INNER JOIN dbo.DimProductSubcategory AS B
        ON B.ProductSubcategoryKey = A.ProductSubcategoryKey
    INNER JOIN dbo.FactCallCenter AS C
        ON A.Size IS NOT NULL
           AND dbo.MinProductIssueFunction(C.IssuesRaised) < 3
GROUP BY dbo.MinProductIssueFunction(C.IssuesRaised),
         A.EnglishProductName,
         A.Size,
```

```sql
        B.EnglishProductSubcategoryName
ORDER BY [Number of Issues]
FOR JSON PATH, ROOT('orderid'), INCLUDE_NULL_VALUES;
```


```
[601]: [Object]
[602]: [Object]
[603]: [Object]
[604]: [Object]
[605]: [Object]
[606]: [Object]
[607]: [Object]
[608]: [Object]
[609]: [Object]
[610]: [Object]
[611]: [Object]
[612]: [Object]
[613]: [Object]
[614]: [Object]
[615]: [Object]
[616]: [Object]
[617]: [Object]
[618]: [Object]
[619]: [Object]
[620]: [Object]
[621]: [Object]
[622]: [Object]
[623]: [Object]
[624]: [Object]
[625]: [Object]
[626]: [Object]
[627]: [Object]
[628]: [Object]
[629]: [Object]
[630]: [Object]
[631]: [Object]
[632]: [Object]
```

```
        }, {
            "EnglishProductName": "Touring-1000 Yellow, 54",
            "Size": "54",
            "EnglishProductSubcategoryName": "Touring Bikes",
            "Number of Issues": 2,
            "Average Calls": 341
        }, {
            "EnglishProductName": "Men's Bib-Shorts, M",
            "Size": "M",
            "EnglishProductSubcategoryName": "Bib-Shorts",
            "Number of Issues": 2,
            "Average Calls": 341
        }, {
            "EnglishProductName": "Mountain Bike Socks, M",
            "Size": "M",
            "EnglishProductSubcategoryName": "Socks",
            "Number of Issues": 2,
            "Average Calls": 341
        }, {
            "EnglishProductName": "Road-450 Red, 58",
            "Size": "58",
            "EnglishProductSubcategoryName": "Road Bikes",
            "Number of Issues": 2,
            "Average Calls": 341
        }, {
            "EnglishProductName": "HL Mountain Frame - Silver, 48",
            "Size": "48",
            "EnglishProductSubcategoryName": "Mountain Frames",
            "Number of Issues": 2,
            "Average Calls": 341
        }
    ]
```

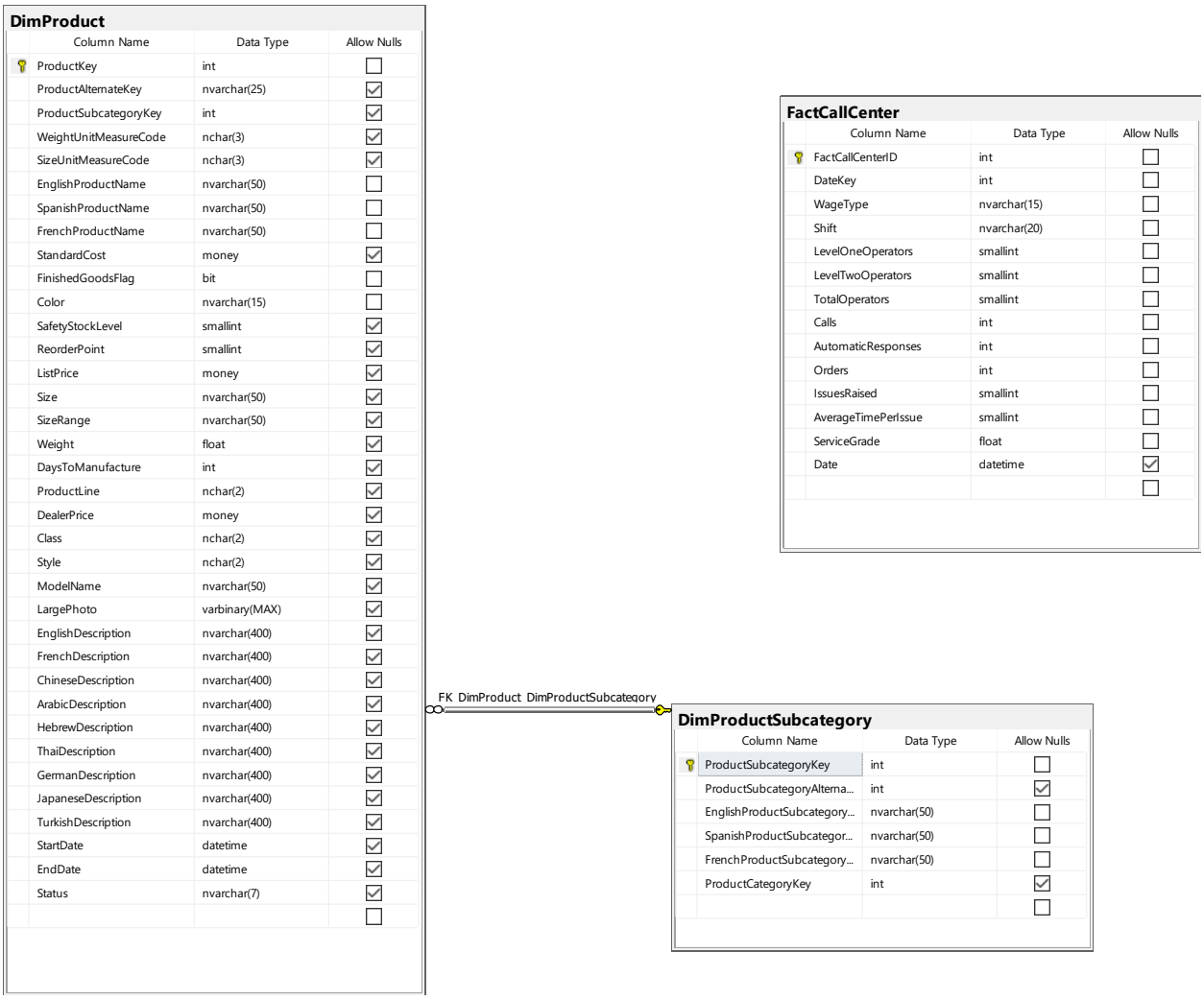
Normal text file | length : 148,473 lines : 3,802 | Ln : 3,798 Col : 35 Sel : 0 | 0 | Windows (CR LF) | UTF-8

Complex Query 4

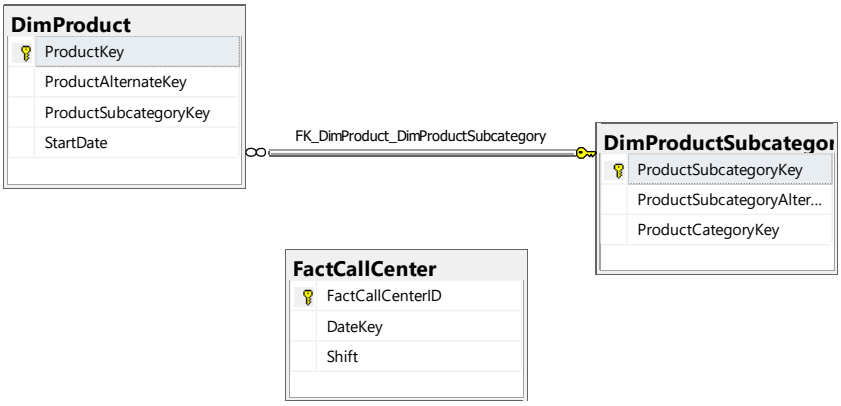
Scalar function returns the maximum number of issues from the company. The query returns the English product name and its category with its size, the average number of calls they get where issues are greater than 2

Use AdventureWorksDW2017 database and dbo.DimProduct, dbo.DimProductSubcategory, dbo.FactCallCenter tables

Standard view



Key view



Columns from tables with order by

Table Name	Column Names	Order By
Dbo.DimProduct	EnglishProductName Size	ASC ASC
Dbo.DimProductSubcategory	EnglishProductSubcategoryName	ASC
Dbo.FactCallCenter	IssuesRaised	ASC

Sample relational solution with output (Returns 210 rows)

```
USE AdventureWorksDW2017;
DROP FUNCTION IF EXISTS dbo.MaxProductIssueFunction;
GO
CREATE FUNCTION dbo.MaxProductIssueFunction
(
    @issues AS SMALLINT
)
RETURNS SMALLINT
AS
BEGIN
    RETURN
    (
        SELECT MAX(@issues) FROM dbo.FactCallCenter
    );
END;
GO

USE AdventureWorksDW2017;
SELECT A.EnglishProductName,
       A.Size,
       B.EnglishProductSubcategoryName,
       dbo.MaxProductIssueFunction(C.IssuesRaised) AS [Number of Issues],
       AVG(C.Calls) AS [Average Calls]
FROM dbo.DimProduct AS A
    INNER JOIN dbo.DimProductSubcategory AS B
        ON B.ProductSubcategoryKey = A.ProductSubcategoryKey
```

```
    INNER JOIN dbo.FactCallCenter AS C
        ON A.Size IS NOT NULL
           AND dbo.MaxProductIssueFunction(C.IssuesRaised) > 2
GROUP BY dbo.MaxProductIssueFunction(C.IssuesRaised),
         A.EnglishProductName,
         A.Size,
         B.EnglishProductSubcategoryName
ORDER BY [Number of Issues];
--FOR JSON PATH, ROOT('orderid'), INCLUDE_NULL_VALUES;
```

100 %

Results | Messages

	EnglishProductName	Size	EnglishProductSubcategoryName	Number of Issues	Average Calls
1	Touring-2000 Blue, 50	50	Touring Bikes	3	283
2	Road-450 Red, 48	48	Road Bikes	3	283
3	Long-Sleeve Logo Jersey, S	S	Jerseys	3	283
4	HL Road Frame - Red, 44	44	Road Frames	3	283
5	Road-550-W Yellow, 42	42	Road Bikes	3	283
6	HL Road Frame - Black, 62	62	Road Frames	3	283
7	LL Touring Frame - Blue, 62	62	Touring Frames	3	283
8	HL Mountain Frame - Silver, 38	38	Mountain Frames	3	283
9	HL Touring Frame - Blue, 46	46	Touring Frames	3	283
10	Mountain-100 Silver, 42	42	Mountain Bikes	3	283
11	Road-250 Black, 58	58	Road Bikes	3	283
12	Mountain-500 Black, 40	40	Mountain Bikes	3	283
13	ML Road Frame - Red, 60	60	Road Frames	3	283
14	Road-350-W Yellow, 42	42	Road Bikes	3	283
15	HL Road Frame - Black, 52	52	Road Frames	3	283
16	Mountain-200 Silver, 38	38	Mountain Bikes	3	283
17	Mountain-200 Silver, 46	46	Mountain Bikes	3	283
18	Women's Tights, L	L	Tights	3	283
19	Mountain-400-W Silver, 38	38	Mountain Bikes	3	283

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (76) | AdventureWorksDW2017 | 00:00:03 | 211 rows

Sample JSON solution with output (Returns 210 objects)

```
USE AdventureWorksDW2017;
DROP FUNCTION IF EXISTS dbo.MaxProductIssueFunction;
GO
CREATE FUNCTION dbo.MaxProductIssueFunction
(
    @issues AS SMALLINT
)
RETURNS SMALLINT
AS
BEGIN
    RETURN
    (
        SELECT MAX(@issues) FROM dbo.FactCallCenter
    );
END;
GO

USE AdventureWorksDW2017;
SELECT A.EnglishProductName,
       A.Size,
       B.EnglishProductSubcategoryName,
       dbo.MaxProductIssueFunction(C.IssuesRaised) AS [Number of Issues],
       AVG(C.Calls) AS [Average Calls]
FROM dbo.DimProduct AS A
    INNER JOIN dbo.DimProductSubcategory AS B
        ON B.ProductSubcategoryKey = A.ProductSubcategoryKey
    INNER JOIN dbo.FactCallCenter AS C
        ON A.Size IS NOT NULL
           AND dbo.MaxProductIssueFunction(C.IssuesRaised) > 2
GROUP BY dbo.MaxProductIssueFunction(C.IssuesRaised),
         A.EnglishProductName,
         A.Size,
```

```sql
        B.EnglishProductSubcategoryName
ORDER BY [Number of Issues]
FOR JSON PATH, ROOT('orderid'), INCLUDE_NULL_VALUES;
```


```
[179]: [Object]
[180]: [Object]
[181]: [Object]
[182]: [Object]
[183]: [Object]
[184]: [Object]
[185]: [Object]
[186]: [Object]
[187]: [Object]
[188]: [Object]
[189]: [Object]
[190]: [Object]
[191]: [Object]
[192]: [Object]
[193]: [Object]
[194]: [Object]
[195]: [Object]
[196]: [Object]
[197]: [Object]
[198]: [Object]
[199]: [Object]
[200]: [Object]
[201]: [Object]
[202]: [Object]
[203]: [Object]
[204]: [Object]
[205]: [Object]
[206]: [Object]
[207]: [Object]
[208]: [Object]
[209]: [Object]
[210]: [Object]
```

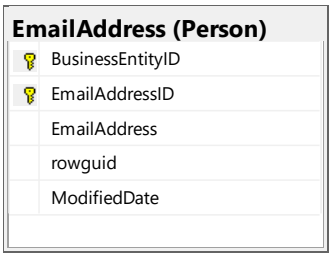
```
1238        }, {
1239            "EnglishProductName": "Classic Vest, L",
1240            "Size": "L",
1241            "EnglishProductSubcategoryName": "Vests",
1242            "Number of Issues": 3,
1243            "Average Calls": 283
1244        }, {
1245            "EnglishProductName": "Road-150 Red, 62",
1246            "Size": "62",
1247            "EnglishProductSubcategoryName": "Road Bikes",
1248            "Number of Issues": 3,
1249            "Average Calls": 283
1250        }, {
1251            "EnglishProductName": "Road-650 Red, 48",
1252            "Size": "48",
1253            "EnglishProductSubcategoryName": "Road Bikes",
1254            "Number of Issues": 3,
1255            "Average Calls": 283
1256        }, {
1257            "EnglishProductName": "LL Touring Frame - Yellow, 44",
1258            "Size": "44",
1259            "EnglishProductSubcategoryName": "Touring Frames",
1260            "Number of Issues": 3,
1261            "Average Calls": 283
1262        }, {
1263            "EnglishProductName": "Mountain-400-W Silver, 40",
1264            "Size": "40",
1265            "EnglishProductSubcategoryName": "Mountain Bikes",
1266            "Number of Issues": 3,
1267            "Average Calls": 283
1268        }
1269    ]
1270
```

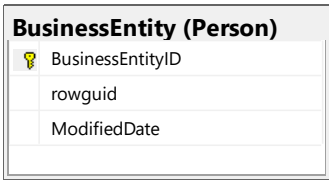
Normal text file | length : 49,509 lines : 1,270 | Ln : 1,269 Col : 6 Sel : 0 | 0 | Windows (CR LF) | UTF-8

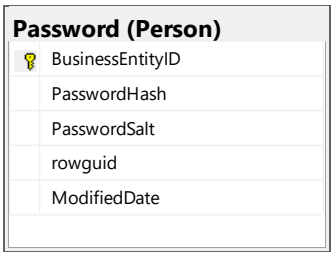
Complex Query 5

```
Scalar function returns the minimum Email promotions. Query returns top 50 email address,
password hash, first and last name where minimum promotions are less than 2

Use AdventureWorks2017 database with Person.EmailAddress, Person.BusinessEntity, and
Person.Password tables
```

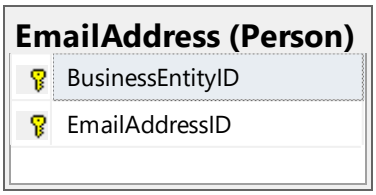
Standard view

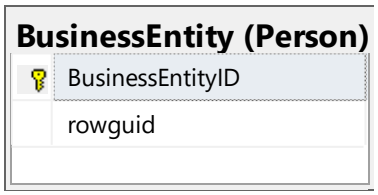






Key view





Columns from tables with order by

Table Name	Column Names	Order By
Person.EmailAddress	EmailAddress	ASC
Person.Password	PasswordHash	ASC
Person.Person	FirstName LastName	ASC ASC

Sample Relational Solution with output (returns 50 rows)

```
USE AdventureWorks2017;
DROP FUNCTION IF EXISTS dbo.MinPromotion;
GO
CREATE FUNCTION dbo.MinPromotion
(
    @Promotion AS INT
)
RETURNS SMALLINT
AS
BEGIN
    RETURN
    (
        SELECT MIN(@Promotion) FROM Person.Person
    );
END;
GO

USE AdventureWorks2017;
SELECT TOP 50
       A.EmailAddress,
       B.PasswordHash,
       C.FirstName,
       C.LastName,
       dbo.MinPromotion(C.EmailPromotion) AS [Minimum Promotions]
FROM Person.EmailAddress AS A
    INNER JOIN Person.Password AS B
        ON B.BusinessEntityID = A.BusinessEntityID
    INNER JOIN Person.Person AS C
        ON C.BusinessEntityID = A.BusinessEntityID
WHERE dbo.MinPromotion(C.EmailPromotion) < 2
GROUP BY dbo.MinPromotion(C.EmailPromotion),
         A.EmailAddress,
         B.PasswordHash,
         C.FirstName,
         C.LastName;
--FOR JSON PATH, ROOT('BusinessEntityID'), INCLUDE_NULL_VALUES;
```

Results | Messages

	EmailAddress	PasswordHash	FirstName	LastName	Minimum Promotions
1	ken0@adventure-works.com	pbFwXWE99vobT6g+vPWFy93NtUU/omIWafF01hccfM=	Ken	Sánchez	0
2	terri0@adventure-works.com	bawRVNrZQYQ05qF05Gz6VLlnviZmrqBReTTAGAudm0=	Terri	Duffy	1
3	roberto0@adventure-works.com	8BUXrZfDqO1IyHCWOYzYmqN1IhTUn3CJMpdx/UCQ3iY=	Roberto	Tamburello	0
4	rob0@adventure-works.com	SjLXpiarHSlz+6AG+H+4QpB/IPRzras/+9q/5Wr7tf8=	Rob	Walters	0
5	gail0@adventure-works.com	8FYdAiY5gWuBsgjCFdg0UlbtsqOcWHf9TyaHIP7+paA=	Gail	Erickson	0
6	jossef0@adventure-works.com	u5kbN5n84NRE1h/a+ktdRrXucjgmrfF6wZC4g82jrHM=	Jossef	Goldberg	0
7	diane1@adventure-works.com	s+FUWADIZzXBKpcbxe4OwL2uiJmjLogJNYXXHvc1X/k=	Diane	Margheim	0
8	gigi0@adventure-works.com	fCvCTy3RwzA2LNhhhYUbT7erkb9Au5wyM2q7ReHroV0=	Gigi	Matthew	0
9	ovidiu0@adventure-works.com	iaZ6ky76dbOG+0Y069v4bm78UffrfGX5eYtxp4Vgd15o=	Ovidiu	Cracium	0
10	david0@adventure-works.com	oaeJoTn5hbyNfemp2qzlpGTP5uNle8NRPki9Ur3Znl8=	David	Bradley	1
11	mary2@adventure-works.com	+NeFaMBZDuBHjMjVI5RQmWChbKw5O6H7LR8DAboLXc=	Mary	Dempsey	1
12	sariya0@adventure-works.com	NrVTt2LIAOOd/kB3YfCaKzLbeOgefZaGTeg2z7TKzhw=	Sariya	Hampadoungsataya	0
13	mary0@adventure-works.com	TlZs9R2WoVp1jfCyxE8Dtaw4OhlcKgCPbqXaOxeIW4=	Mary	Gibson	0
14	jill0@adventure-works.com	wI5v4SNv5Mg5ea0Ufy1xy966PrXWCBp8gLlhC1QTqWw=	Jill	Williams	0
15	james1@adventure-works.com	iYoywqstKDfRLnJwKmI/ObzZl6KQ4eX5hUgBa1Qgz0=	James	Hamilton	0
16	peter0@adventure-works.com	GCZ//7zDxPTsO9cuY1Rk0uUUlc6t2jEGZwjGqXVl6Ws=	Peter	Krebs	0
17	jo0@adventure-works.com	uQaHsaafTmDMRjX9nNhAFjHaYjqeTZhlcZp4azb5Hgs=	Jo	Brown	0
18	guy1@adventure-works.com	Li2+pm3yhRgX8v0v3JUZWL8NPgdBse1zAG1ph25eFAk=	Guy	Gilbert	0
19	mark1@adventure-works.com	2sudtPUZMsjMIVXKcfx3NPWhTws5ZSHJa+pzKYnyxtU=	Mark	McArthur	1

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (73) | AdventureWorks2017 | 00:00:00 | 50 rows

Sample JSON Solution with output (returns 50 objects)

```
USE AdventureWorks2017;
DROP FUNCTION IF EXISTS dbo.MinPromotion;
GO
CREATE FUNCTION dbo.MinPromotion
(
    @Promotion AS INT
)
RETURNS SMALLINT
AS
BEGIN
    RETURN
    (
        SELECT MIN(@Promotion) FROM Person.Person
    );
END;
GO

USE AdventureWorks2017;
SELECT TOP 50
       A.EmailAddress,
       B.PasswordHash,
       C.FirstName,
       C.LastName,
       dbo.MinPromotion(C.EmailPromotion) AS [Minimum Promotions]
FROM Person.EmailAddress AS A
    INNER JOIN Person.Password AS B
        ON B.BusinessEntityID = A.BusinessEntityID
    INNER JOIN Person.Person AS C
        ON C.BusinessEntityID = A.BusinessEntityID
WHERE dbo.MinPromotion(C.EmailPromotion) < 2
GROUP BY dbo.MinPromotion(C.EmailPromotion),
         A.EmailAddress,
         B.PasswordHash,
         C.FirstName,
         C.LastName
FOR JSON PATH, ROOT('BusinessEntityID'), INCLUDE_NULL_VALUES;
```

```
[18]: [Object]
[19]: [Object]
[20]: [Object]
[21]: [Object]
[22]: [Object]
[23]: [Object]
[24]: [Object]
[25]: [Object]
[26]: [Object]
[27]: [Object]
[28]: [Object]
[29]: [Object]
[30]: [Object]
[31]: [Object]
[32]: [Object]
[33]: [Object]
[34]: [Object]
[35]: [Object]
[36]: [Object]
[37]: [Object]
[38]: [Object]
[39]: [Object]
[40]: [Object]
[41]: [Object]
[42]: [Object]
[43]: [Object]
[44]: [Object]
[45]: [Object]
[46]: [Object]
[47]: [Object]
[48]: [Object]
[49]: [Object]
```

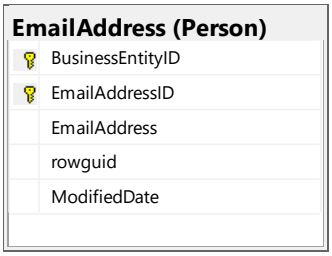
```
    }, {
        "EmailAddress": "michael7@adventure-works.com",
        "PasswordHash": "+jRUrihbwVEXmyQGmKz4a\/wALG2jzFJ616X\/tBQMrPk=",
        "FirstName": "Michael",
        "LastName": "Zwilling",
        "Minimum Promotions": 1
    }, {
        "EmailAddress": "randy0@adventure-works.com",
        "PasswordHash": "Vh6leuy0nDocJST\/CrxONfgv\/AHQebcBWDPk0RW4f\/0=",
        "FirstName": "Randy",
        "LastName": "Reeves",
        "Minimum Promotions": 0
    }, {
        "EmailAddress": "karan0@adventure-works.com",
        "PasswordHash": "RSJWB96i8LhoGXvbT1Wc9SPtwGh+EsrYVowBTzFDBSk=",
        "FirstName": "Karan",
        "LastName": "Khanna",
        "Minimum Promotions": 1
    }, {
        "EmailAddress": "jay0@adventure-works.com",
        "PasswordHash": "btAotCpGbd8u21XFa9jRomwxM9snEhk2j+T7FFf1QTs=",
        "FirstName": "Jay",
        "LastName": "Adams",
        "Minimum Promotions": 0
    }, {
        "EmailAddress": "steve0@adventure-works.com",
        "PasswordHash": "U4JRa3QwDqShrc\/sWzTbQyps\/i2Hsu\/qWDme7xF0+Rc=",
        "FirstName": "Steve",
        "LastName": "Masters",
        "Minimum Promotions": 1
    }
]
```

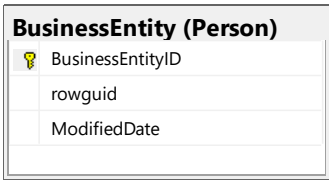
Normal text file | length : 12,970 lines : 304 | Ln : 303 Col : 6 Sel : 0 | 0 | Windows (CR LF) | UTF-8

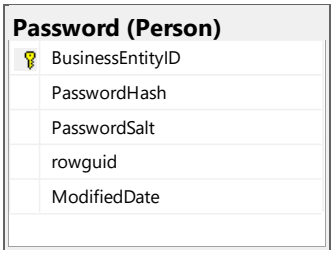
Complex Query 6

```
Scalar function returns the maximum Email promotions. Query returns email address,
password hash, first and last name where minimum promotions are greater than or equal to
2

Use AdventureWorks2017 database with Person.EmailAddress, Person.BusinessEntity, and
Person.Password tables
```

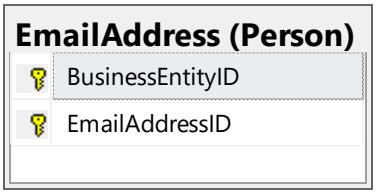
Standard view

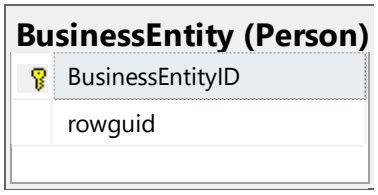






Key view





Columns from tables with order by

Table Name	Column Names	Order By
Person.EmailAddress	EmailAddress	ASC
Person.Password	PasswordHash	ASC
Person.Person	FirstName LastName	ASC ASC

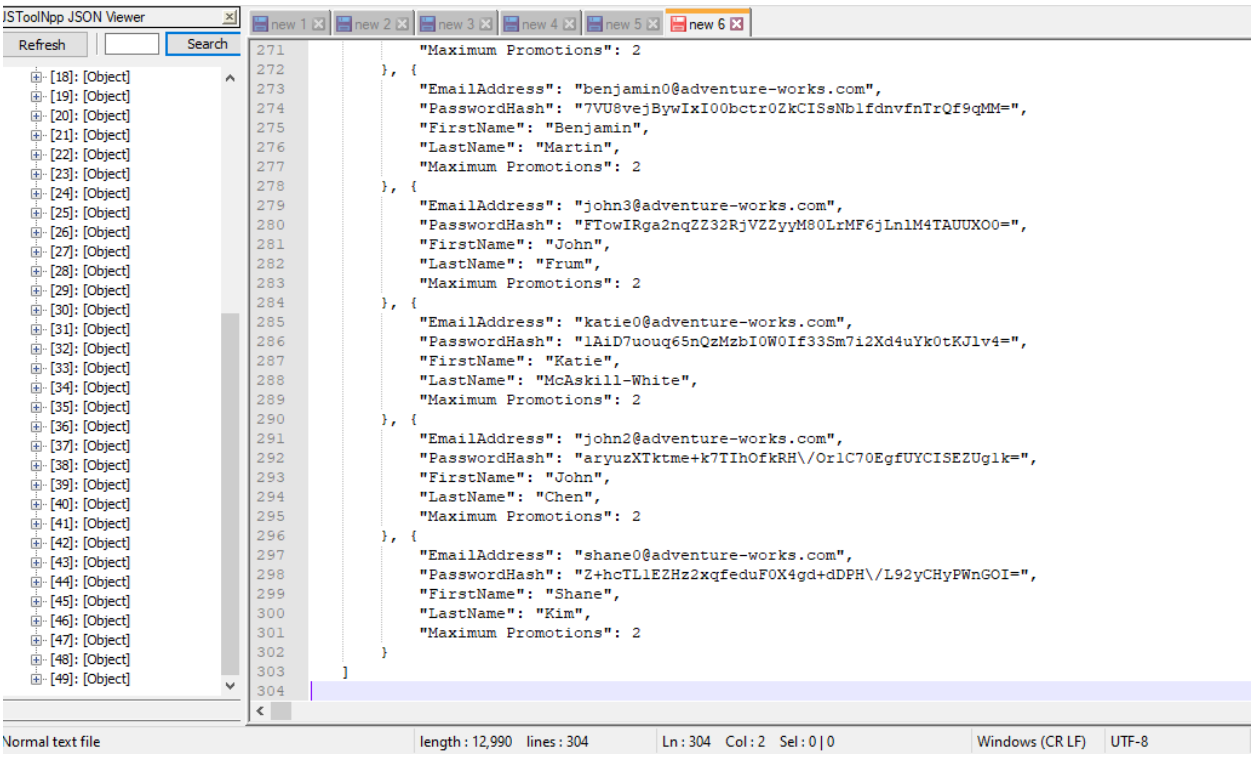
Sample relational solution with output (Returns 50 rows)

```
USE AdventureWorks2017;
DROP FUNCTION IF EXISTS dbo.MaxPromotion;
GO
CREATE FUNCTION dbo.MaxPromotion
(
    @Promotion AS INT
)
RETURNS SMALLINT
AS
BEGIN
    RETURN
    (
        SELECT MAX(@Promotion) FROM Person.Person
    );
END;
GO

USE AdventureWorks2017;
SELECT TOP 50
       A.EmailAddress,
       B.PasswordHash,
       C.FirstName,
       C.LastName,
       dbo.MaxPromotion(C.EmailPromotion) AS [Maximum Promotions]
FROM Person.EmailAddress AS A
    INNER JOIN Person.Password AS B
        ON B.BusinessEntityID = A.BusinessEntityID
    INNER JOIN Person.Person AS C
        ON C.BusinessEntityID = A.BusinessEntityID
WHERE dbo.MaxPromotion(C.EmailPromotion) >= 2
GROUP BY dbo.MaxPromotion(C.EmailPromotion),
         A.EmailAddress,
         B.PasswordHash,
         C.FirstName,
         C.LastName;
--FOR JSON PATH, ROOT('BusinessEntityID'), INCLUDE_NULL_VALUES;
```

Sample JSON solution with output (Returns 50 objects)

```
USE AdventureWorks2017;
DROP FUNCTION IF EXISTS dbo.MaxPromotion;
GO
```

```sql
CREATE FUNCTION dbo.MaxPromotion
(
    @Promotion AS INT
)
RETURNS SMALLINT
AS
BEGIN
    RETURN
    (
        SELECT MAX(@Promotion) FROM Person.Person
    );
END;
GO

USE AdventureWorks2017;
SELECT TOP 50
       A.EmailAddress,
       B.PasswordHash,
       C.FirstName,
       C.LastName,
       dbo.MaxPromotion(C.EmailPromotion) AS [Maximum Promotions]
FROM Person.EmailAddress AS A
    INNER JOIN Person.Password AS B
        ON B.BusinessEntityID = A.BusinessEntityID
    INNER JOIN Person.Person AS C
        ON C.BusinessEntityID = A.BusinessEntityID
WHERE dbo.MaxPromotion(C.EmailPromotion) >= 2
GROUP BY dbo.MaxPromotion(C.EmailPromotion),
         A.EmailAddress,
         B.PasswordHash,
         C.FirstName,
         C.LastName
FOR JSON PATH, ROOT('BusinessEntityID'), INCLUDE_NULL_VALUES;
```



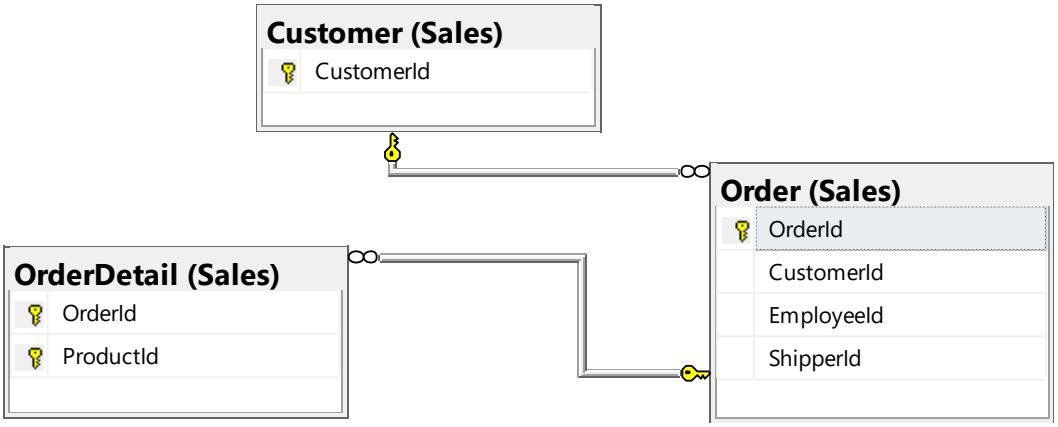

```
271                 "Maximum Promotions": 2
272             }, {
273                 "EmailAddress": "benjamin0@adventure-works.com",
274                 "PasswordHash": "7VU8vejBywIxI00bctr0ZkCISsNblfdnvfnTrQf9qMM=",
275                 "FirstName": "Benjamin",
276                 "LastName": "Martin",
277                 "Maximum Promotions": 2
278             }, {
279                 "EmailAddress": "john3@adventure-works.com",
280                 "PasswordHash": "FTowIRga2nqZZ32RjVZZyyM80LrMF6jLnlM4TAUUXO0=",
281                 "FirstName": "John",
282                 "LastName": "Frum",
283                 "Maximum Promotions": 2
284             }, {
285                 "EmailAddress": "katie0@adventure-works.com",
286                 "PasswordHash": "lAiD7uouq65nQzMzbI0W0If33Sm7i2Xd4uYk0tKJlv4=",
287                 "FirstName": "Katie",
288                 "LastName": "McAskill-White",
289                 "Maximum Promotions": 2
290             }, {
291                 "EmailAddress": "john2@adventure-works.com",
292                 "PasswordHash": "aryuzXTktme+k7TIhOfkRH\/OrlC70EgfUYCISEZUglk=",
293                 "FirstName": "John",
294                 "LastName": "Chen",
295                 "Maximum Promotions": 2
296             }, {
297                 "EmailAddress": "shane0@adventure-works.com",
298                 "PasswordHash": "Z+hcTLlEZHz2xqfeduF0X4gd+dDPH\/L92yCHyPWnGOI=",
299                 "FirstName": "Shane",
300                 "LastName": "Kim",
301                 "Maximum Promotions": 2
302             }
303         ]
304
```

Complex Query 7

```
Scalar function returns the customer id where it was shipped to the USA and their max
quantity

Use Northwinds2020TSQLV6 database with Sales.Customer, Sales.Order, Sales.OrderDetail
tables
```

Standard view

Key view



Columns from tables with order by

Table Name	Column Names	Order By
Sales.Customer	CustomerID	ASC
Sales.[Order]	ShipToCountry	ASC
Sales.OrderDetail	Quantity	ASC

Sample Relational solution with output (returns 149 rows)

```
USE Northwinds2020TSQLV6
DROP FUNCTION IF EXISTS dbo.MAXQuantity;
GO
CREATE FUNCTION dbo.MAXQuantity
(
    @quantity AS SMALLINT
)
RETURNS SMALLINT
AS
BEGIN
    RETURN
    (
        SELECT MAX(@quantity) FROM Sales.[ORDERDetail]
    );
END;
GO

USE Northwinds2020TSQLV6;
SELECT A.CustomerId,
       B.ShipToCountry,
       dbo.MAXQuantity(C.Quantity) AS [Maximum Quantity]
FROM Sales.Customer AS A
```

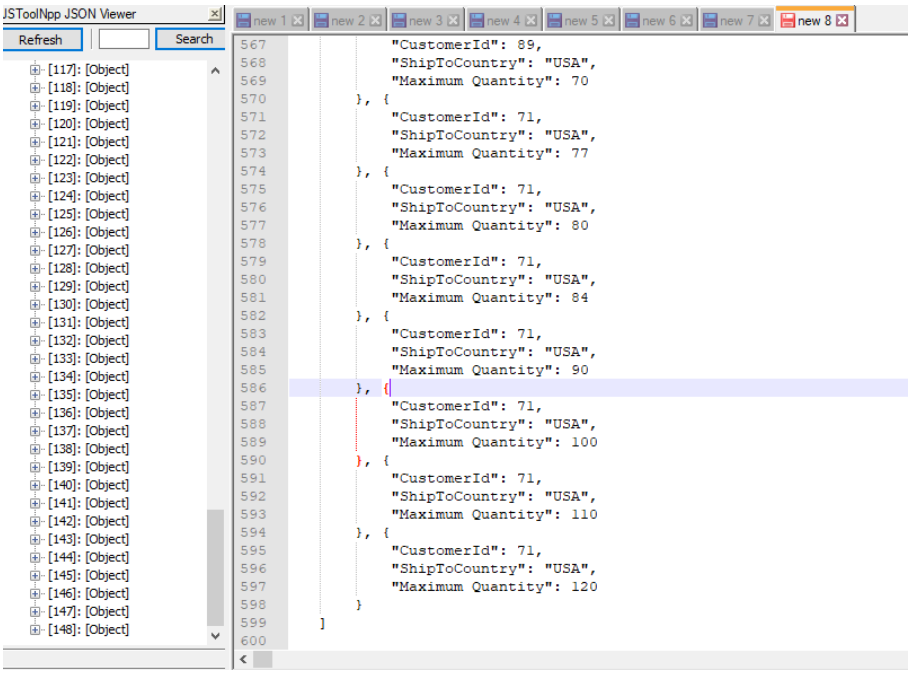
```
    INNER JOIN Sales.[Order] AS B
        ON B.CustomerId = A.CustomerId
    INNER JOIN Sales.OrderDetail AS C
        ON C.OrderId = B.OrderId
WHERE B.ShipToCountry LIKE 'USA'
GROUP BY dbo.MAXQuantity(C.Quantity),
         A.CustomerId,
         B.ShipToCountry;
--FOR JSON PATH, ROOT('CustomerId'), INCLUDE_NULL_VALUES;
```

	CustomerId	ShipToCountry	Maximum Quantity
1	48	USA	1
2	65	USA	1
3	32	USA	2
4	36	USA	2
5	45	USA	2
6	65	USA	2
7	77	USA	2
8	82	USA	2
9	89	USA	2
10	32	USA	3
11	45	USA	3
12	48	USA	3
13	65	USA	3
14	71	USA	3
15	78	USA	3
16	82	USA	3
17	89	USA	3
18	32	USA	4
19	36	USA	4

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (75) | Northwinds2020TSQLV6 | 00:00:01 | 149 rows

Sample JSON solution with output (Returns Objects 149)

```
USE Northwinds2020TSQLV6
DROP FUNCTION IF EXISTS dbo.MAXQuantity;
GO
CREATE FUNCTION dbo.MAXQuantity
(
    @quantity AS SMALLINT
)
RETURNS SMALLINT
AS
BEGIN
    RETURN
    (
        SELECT MAX(@quantity) FROM Sales.[ORDERDetail]
    );
END;
GO

USE Northwinds2020TSQLV6;
SELECT A.CustomerId,
       B.ShipToCountry,
       dbo.MAXQuantity(C.Quantity) AS [Maximum Quantity]
FROM Sales.Customer AS A
    INNER JOIN Sales.[Order] AS B
        ON B.CustomerId = A.CustomerId
    INNER JOIN Sales.OrderDetail AS C
        ON C.OrderId = B.OrderId
WHERE B.ShipToCountry LIKE 'USA'
GROUP BY dbo.MAXQuantity(C.Quantity),
         A.CustomerId,
         B.ShipToCountry
FOR JSON PATH, ROOT('CustomerId'), INCLUDE_NULL_VALUES;
```


JSToolNpp JSON Viewer
Refresh
Search
[117]: [Object]
[118]: [Object]
[119]: [Object]
[120]: [Object]
[121]: [Object]
[122]: [Object]
[123]: [Object]
[124]: [Object]
[125]: [Object]
[126]: [Object]
[127]: [Object]
[128]: [Object]
[129]: [Object]
[130]: [Object]
[131]: [Object]
[132]: [Object]
[133]: [Object]
[134]: [Object]
[135]: [Object]
[136]: [Object]
[137]: [Object]
[138]: [Object]
[139]: [Object]
[140]: [Object]
[141]: [Object]
[142]: [Object]
[143]: [Object]
[144]: [Object]
[145]: [Object]
[146]: [Object]
[147]: [Object]
[148]: [Object]
new 1
new 2
new 3
new 4
new 5
new 6
new 7
new 8
"CustomerId": 89,
"ShipToCountry": "USA",
"Maximum Quantity": 70
}, {
"CustomerId": 71,
"ShipToCountry": "USA",
"Maximum Quantity": 77
}, {
"CustomerId": 71,
"ShipToCountry": "USA",
"Maximum Quantity": 80
}, {
"CustomerId": 71,
"ShipToCountry": "USA",
"Maximum Quantity": 84
}, {
"CustomerId": 71,
"ShipToCountry": "USA",
"Maximum Quantity": 90
}, {
"CustomerId": 71,
"ShipToCountry": "USA",
"Maximum Quantity": 100
}, {
"CustomerId": 71,
"ShipToCountry": "USA",
"Maximum Quantity": 110
}, {
"CustomerId": 71,
"ShipToCountry": "USA",
"Maximum Quantity": 120
}
]