Marlon Louis

Prof. Heller

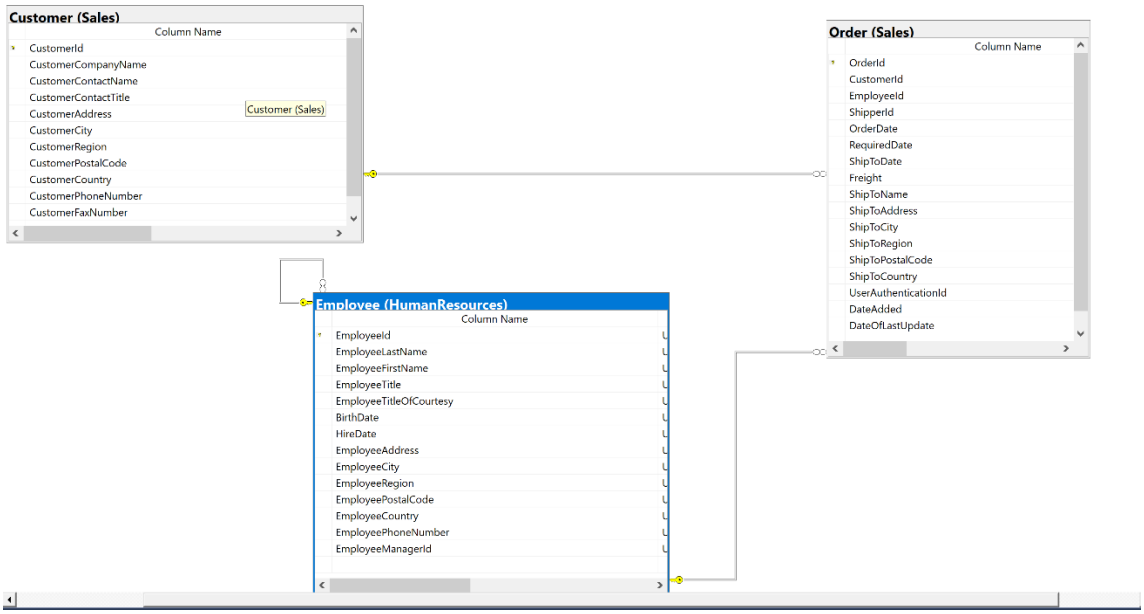
CSCI331

Group 4

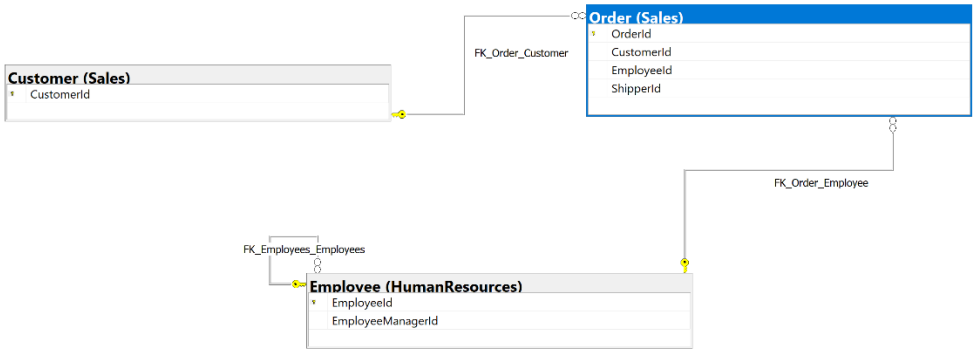
20 Individual Queries

SIMPLE 1: Find all customers who placed orders from US and the name and region of the employee who serviced them

Standard View:



Key View:



Columns from tables:

Table Name	Column Names
Sales.Customer	CustomerCountry CustomerId CustomerContactName
HumanResources.Employee	EmployeeFirstName EmployeeId

Order by

Table Name	Column Name	Sort Order
Sales.Customer	CustomerId	ASC
HumanResources.Employee	EmployeeId	ASC

Solution Without JSON:

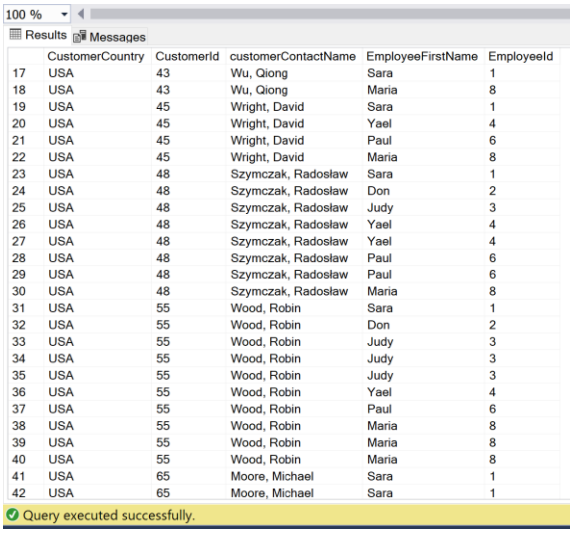
```
USE Northwinds2020TSQLV6;
SELECT c.CustomerCountry
      ,c.CustomerId
      ,c.customerContactName
      ,e.EmployeeFirstName
      ,e.EmployeeId
FROM Sales.Customer AS C
INNER JOIN Sales.[Order] AS O ON c.CustomerId = o.CustomerId
INNER JOIN HumanResources.Employee AS E ON e.EmployeeId = o.EmployeeId
WHERE c.CustomerCountry = N'USA'
ORDER BY c.CustomerId
      ,e.EmployeeId
```

Solution with JSON:

```
      USE Northwinds2020TSQLV6;
SELECT c.CustomerCountry
      ,c.CustomerId
      ,c.customerContactName
      ,e.EmployeeFirstName
```

```
        ,e.EmployeeId
FROM Sales.Customer AS C
INNER JOIN Sales.[Order] AS O ON c.CustomerId = o.CustomerId
INNER JOIN HumanResources.Employee AS E ON e.EmployeeId = o.EmployeeId
WHERE c.CustomerCountry = N'USA'
ORDER BY c.CustomerId
        ,e.EmployeeId
FOR JSON PATH, ROOT('Output'), INCLUDE_NULL_VALUES;
```

Sample Output (122 results returned):

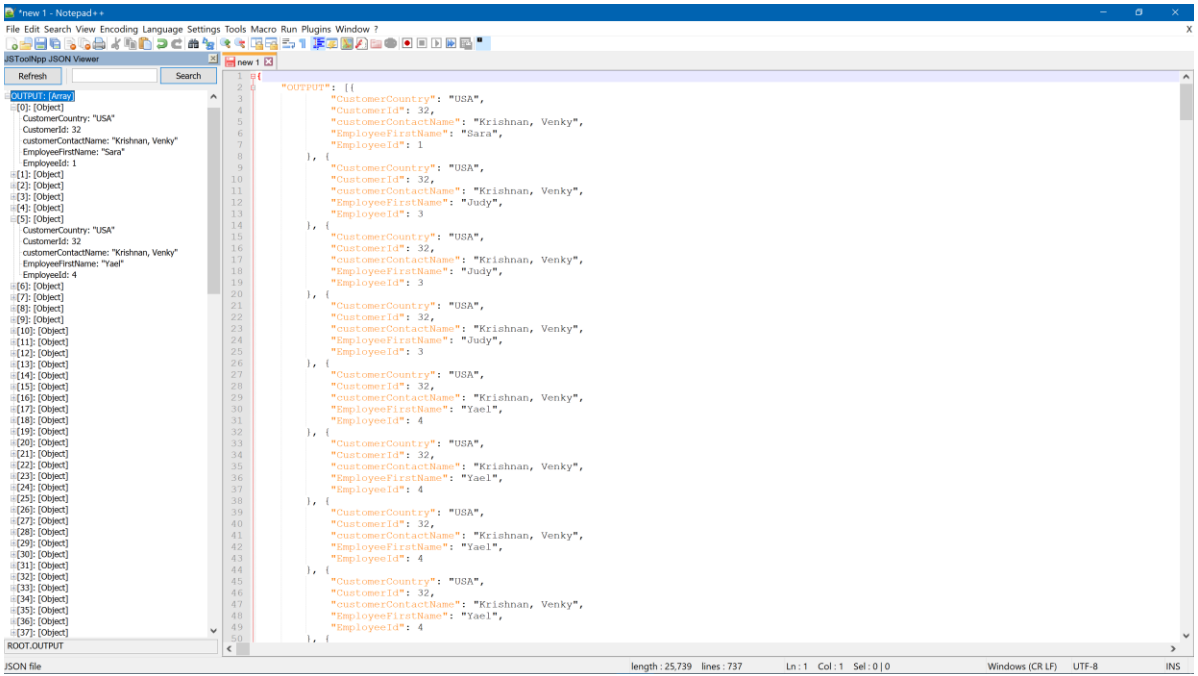


100 % Results Messages

	CustomerCountry	CustomerId	customerContactName	EmployeeFirstName	EmployeeId
17	USA	43	Wu, Qiong	Sara	1
18	USA	43	Wu, Qiong	Maria	8
19	USA	45	Wright, David	Sara	1
20	USA	45	Wright, David	Yael	4
21	USA	45	Wright, David	Paul	6
22	USA	45	Wright, David	Maria	8
23	USA	48	Szymczak, Radosław	Sara	1
24	USA	48	Szymczak, Radosław	Don	2
25	USA	48	Szymczak, Radosław	Judy	3
26	USA	48	Szymczak, Radosław	Yael	4
27	USA	48	Szymczak, Radosław	Yael	4
28	USA	48	Szymczak, Radosław	Paul	6
29	USA	48	Szymczak, Radosław	Paul	6
30	USA	48	Szymczak, Radosław	Maria	8
31	USA	55	Wood, Robin	Sara	1
32	USA	55	Wood, Robin	Don	2
33	USA	55	Wood, Robin	Judy	3
34	USA	55	Wood, Robin	Judy	3
35	USA	55	Wood, Robin	Judy	3
36	USA	55	Wood, Robin	Yael	4
37	USA	55	Wood, Robin	Paul	6
38	USA	55	Wood, Robin	Maria	8
39	USA	55	Wood, Robin	Maria	8
40	USA	55	Wood, Robin	Maria	8
41	USA	65	Moore, Michael	Sara	1
42	USA	65	Moore, Michael	Sara	1

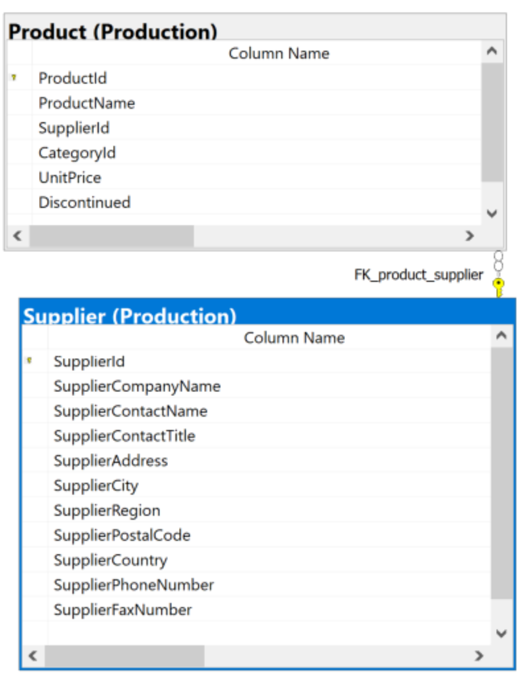
Query executed successfully.

Sample JSON Output (122 results returned):



SIMPLE 2: find all product names, price, and the name of the product supplier

Standard View:



Key View:

Columns from tables:

Table Name	Column Name
Production.Product	ProductName UnitPrice
Production.Supplier	SupplierContactName SupplierContactTitle SupplierCompanyName

Order By:

Table Name	Column Name	Sort Order
Production.Product	UnitPrice	DESC

Solution without JSON:

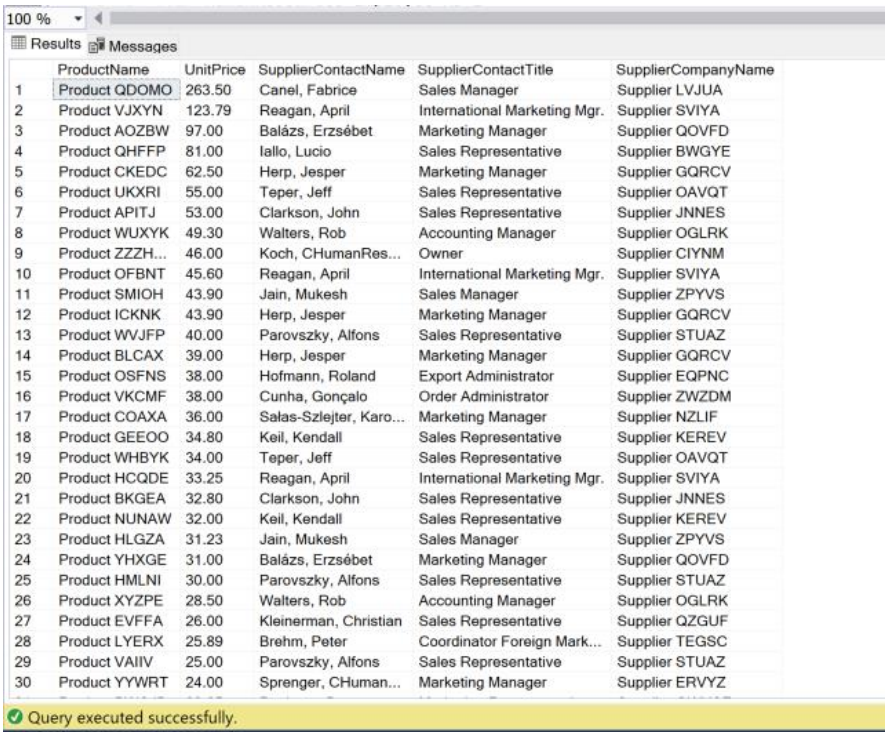
```
USE Northwinds2020TSQLV6;
SELECT p.ProductName
      ,p.UnitPrice
      ,s.SupplierContactName
      ,s.SupplierContactTitle
      ,s.SupplierCompanyName
FROM Production.Product AS P
INNER JOIN Production.Supplier AS S ON p.SupplierId = s.SupplierId
```

```
ORDER BY p.UnitPrice DESC
```

Solution with JSON:

```
USE Northwinds2020TSQLV6;
SELECT p.ProductName
      ,p.UnitPrice
      ,s.SupplierContactName
      ,s.SupplierContactTitle
      ,s.SupplierCompanyName
FROM Production.Product AS P
INNER JOIN Production.Supplier AS S ON p.SupplierId = s.SupplierId
ORDER BY p.UnitPrice DESC
FOR JSON PATH, ROOT('Simple 2 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (77 results returned):



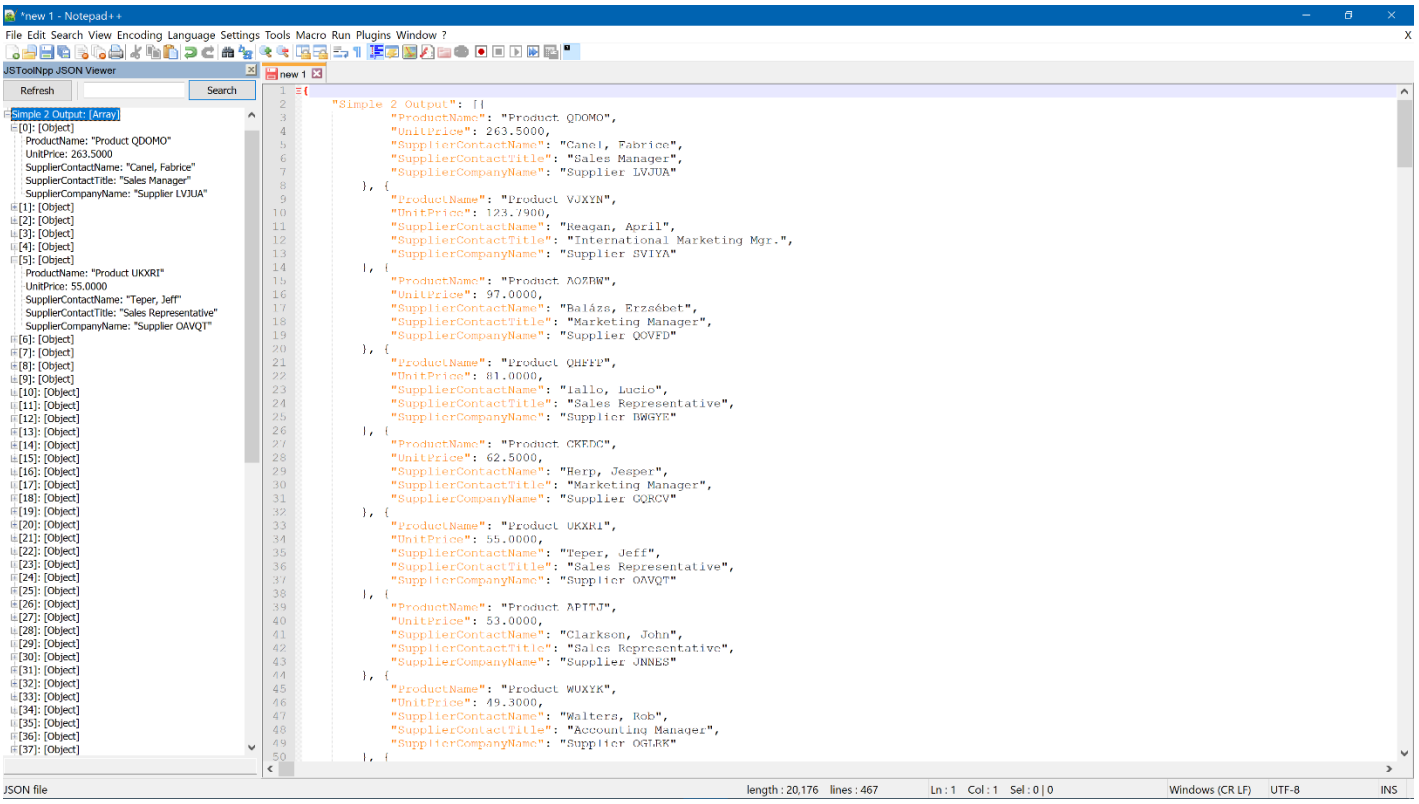
100 %

Results Messages

	ProductName	UnitPrice	SupplierContactName	SupplierContactTitle	SupplierCompanyName
1	Product QDOMO	263.50	Canel, Fabrice	Sales Manager	Supplier LVJUA
2	Product VJXYN	123.79	Reagan, April	International Marketing Mgr.	Supplier SVIYA
3	Product AOZBW	97.00	Balázs, Erzsébet	Marketing Manager	Supplier QOVFD
4	Product QHFFP	81.00	Iallo, Lucio	Sales Representative	Supplier BWGYE
5	Product CKEDC	62.50	Herp, Jesper	Marketing Manager	Supplier GQRCV
6	Product UKXRI	55.00	Teper, Jeff	Sales Representative	Supplier OAVQT
7	Product APITJ	53.00	Clarkson, John	Sales Representative	Supplier JNNES
8	Product WUXYK	49.30	Walters, Rob	Accounting Manager	Supplier OGLRK
9	Product ZZZH...	46.00	Koch, CHumanRes...	Owner	Supplier CIYNM
10	Product OFBNT	45.60	Reagan, April	International Marketing Mgr.	Supplier SVIYA
11	Product SMIOH	43.90	Jain, Mukesh	Sales Manager	Supplier ZPYVS
12	Product ICKNK	43.90	Herp, Jesper	Marketing Manager	Supplier GQRCV
13	Product WVJFP	40.00	Parovszky, Alfons	Sales Representative	Supplier STUAZ
14	Product BLCAX	39.00	Herp, Jesper	Marketing Manager	Supplier GQRCV
15	Product OSFNS	38.00	Hofmann, Roland	Export Administrator	Supplier EQPNC
16	Product VKCMF	38.00	Cunha, Gonçalo	Order Administrator	Supplier ZWZDM
17	Product COAXA	36.00	Sałas-Szlejter, Karo...	Marketing Manager	Supplier NZLIF
18	Product GEEOO	34.80	Keil, Kendall	Sales Representative	Supplier KEREV
19	Product WHBYK	34.00	Teper, Jeff	Sales Representative	Supplier OAVQT
20	Product HCQDE	33.25	Reagan, April	International Marketing Mgr.	Supplier SVIYA
21	Product BKGEA	32.80	Clarkson, John	Sales Representative	Supplier JNNES
22	Product NUNAW	32.00	Keil, Kendall	Sales Representative	Supplier KEREV
23	Product HLGZA	31.23	Jain, Mukesh	Sales Manager	Supplier ZPYVS
24	Product YHXGE	31.00	Balázs, Erzsébet	Marketing Manager	Supplier QOVFD
25	Product HMLNI	30.00	Parovszky, Alfons	Sales Representative	Supplier STUAZ
26	Product XYZPE	28.50	Walters, Rob	Accounting Manager	Supplier OGLRK
27	Product EVFFA	26.00	Kleinerman, Christian	Sales Representative	Supplier QZGUF
28	Product LYERX	25.89	Brehm, Peter	Coordinator Foreign Mark...	Supplier TEGSC
29	Product VAIIV	25.00	Parovszky, Alfons	Sales Representative	Supplier STUAZ
30	Product YYWRT	24.00	Sprenger, CHuman...	Marketing Manager	Supplier ERVYZ

Query executed successfully.

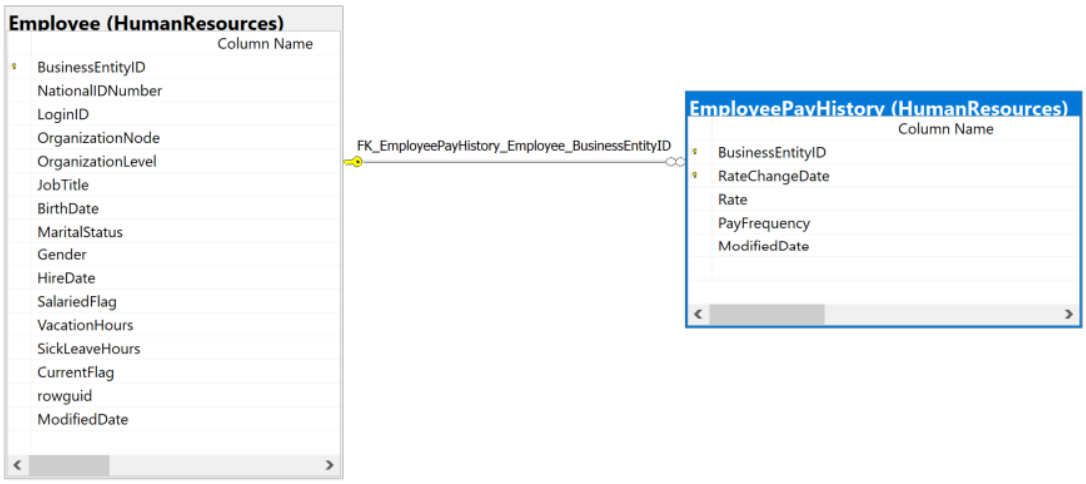
Sample JSON Output (77 results returned):



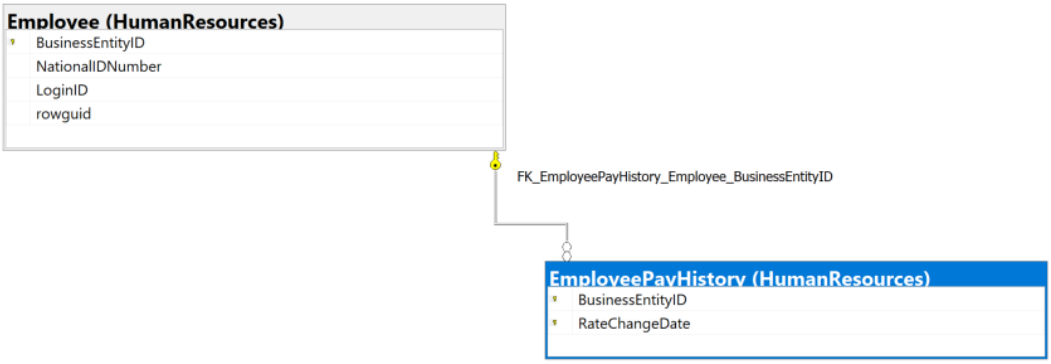

```
{
    "Simple 2 Output": [
        {
            "ProductName": "Product QDOMO",
            "UnitPrice": 263.5000,
            "SupplierContactName": "Canel, Fabrice",
            "SupplierContactTitle": "Sales Manager",
            "SupplierCompanyName": "Supplier LVJUA"
        }, {
            "ProductName": "Product VJXYN",
            "UnitPrice": 123.7900,
            "SupplierContactName": "Reagan, April",
            "SupplierContactTitle": "International Marketing Mgr.",
            "SupplierCompanyName": "Supplier SVIYA"
        }, {
            "ProductName": "Product AOZBW",
            "UnitPrice": 97.0000,
            "SupplierContactName": "Baláza, Erzsébet",
            "SupplierContactTitle": "Marketing Manager",
            "SupplierCompanyName": "Supplier QOVFD"
        }, {
            "ProductName": "Product QHFFP",
            "UnitPrice": 81.0000,
            "SupplierContactName": "Iallo, Lucio",
            "SupplierContactTitle": "Sales Representative",
            "SupplierCompanyName": "Supplier BWGYE"
        }, {
            "ProductName": "Product CKEDC",
            "UnitPrice": 62.5000,
            "SupplierContactName": "Herp, Jesper",
            "SupplierContactTitle": "Marketing Manager",
            "SupplierCompanyName": "Supplier QQRCV"
        }, {
            "ProductName": "Product UKXRI",
            "UnitPrice": 55.0000,
            "SupplierContactName": "Teper, Jeff",
            "SupplierContactTitle": "Sales Representative",
            "SupplierCompanyName": "Supplier OAVQT"
        }, {
            "ProductName": "Product APTTJ",
            "UnitPrice": 53.0000,
            "SupplierContactName": "Clarkson, John",
            "SupplierContactTitle": "Sales Representative",
            "SupplierCompanyName": "Supplier JNNES"
        }, {
            "ProductName": "Product WUXYK",
            "UnitPrice": 49.3000,
            "SupplierContactName": "Walters, Rob",
            "SupplierContactTitle": "Accounting Manager",
            "SupplierCompanyName": "Supplier OGLRK"
        }, {
```


SIMPLE 3: list all employee titles and their current pay

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
HumanResources.Employee	BusinessEntityId JobTitle
HumanResources.EmployeePayHistory	Rate

Order By:

Table Name	Column Name	Sort Order
HumanResources.EmployeePayHistory	HourlyWage	DESC

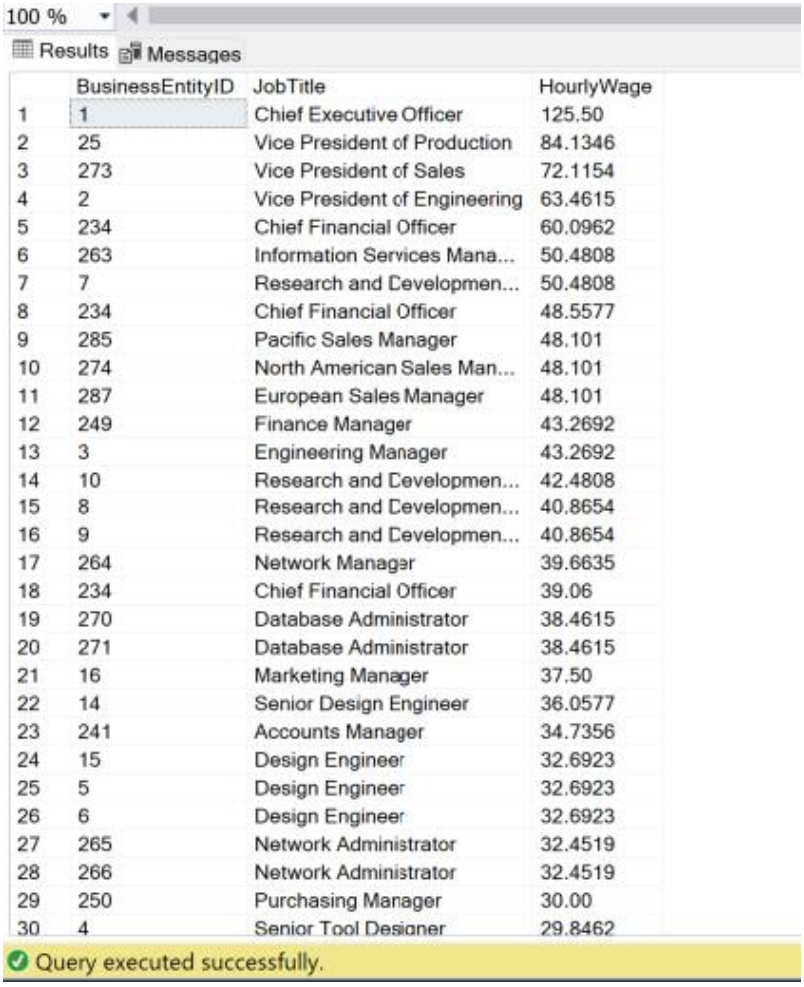
Solution without JSON:

```
USE AdventureWorks2017;

SELECT e.BusinessEntityID

    ,e.JobTitle

    ,p.Rate AS HourlyWage

FROM HumanResources.Employee AS E

INNER JOIN HumanResources.EmployeePayHistory AS P ON e.BusinessEntityID =
p.BusinessEntityID

ORDER BY HourlyWage DESC
```

Solution with JSON:

```
USE AdventureWorks2017;

SELECT e.BusinessEntityID

    ,e.JobTitle

    ,p.Rate AS HourlyWage

FROM HumanResources.Employee AS E

INNER JOIN HumanResources.EmployeePayHistory AS P ON e.BusinessEntityID = p.BusinessEntityID

ORDER BY HourlyWage DESC

FOR JSON PATH, ROOT('Simple 3 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (316 results returned):



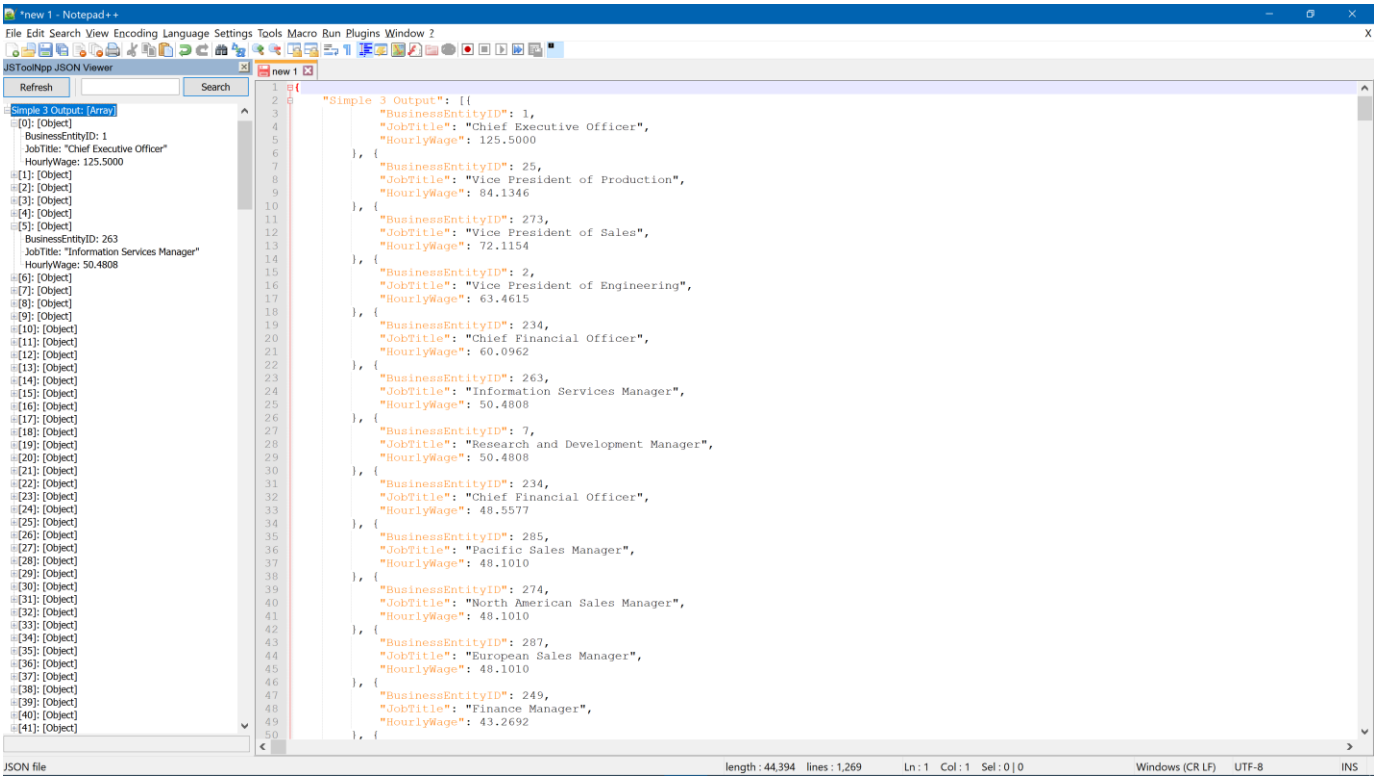
100 %

Results Messages

	BusinessEntityID	JobTitle	HourlyWage
1	1	Chief Executive Officer	125.50
2	25	Vice President of Production	84.1346
3	273	Vice President of Sales	72.1154
4	2	Vice President of Engineering	63.4615
5	234	Chief Financial Officer	60.0962
6	263	Information Services Mana...	50.4808
7	7	Research and Development...	50.4808
8	234	Chief Financial Officer	48.5577
9	285	Pacific Sales Manager	48.101
10	274	North American Sales Man...	48.101
11	287	European Sales Manager	48.101
12	249	Finance Manager	43.2692
13	3	Engineering Manager	43.2692
14	10	Research and Development...	42.4808
15	8	Research and Development...	40.8654
16	9	Research and Development...	40.8654
17	264	Network Manager	39.6635
18	234	Chief Financial Officer	39.06
19	270	Database Administrator	38.4615
20	271	Database Administrator	38.4615
21	16	Marketing Manager	37.50
22	14	Senior Design Engineer	36.0577
23	241	Accounts Manager	34.7356
24	15	Design Engineer	32.6923
25	5	Design Engineer	32.6923
26	6	Design Engineer	32.6923
27	265	Network Administrator	32.4519
28	266	Network Administrator	32.4519
29	250	Purchasing Manager	30.00
30	4	Senior Tool Designer	29.8462

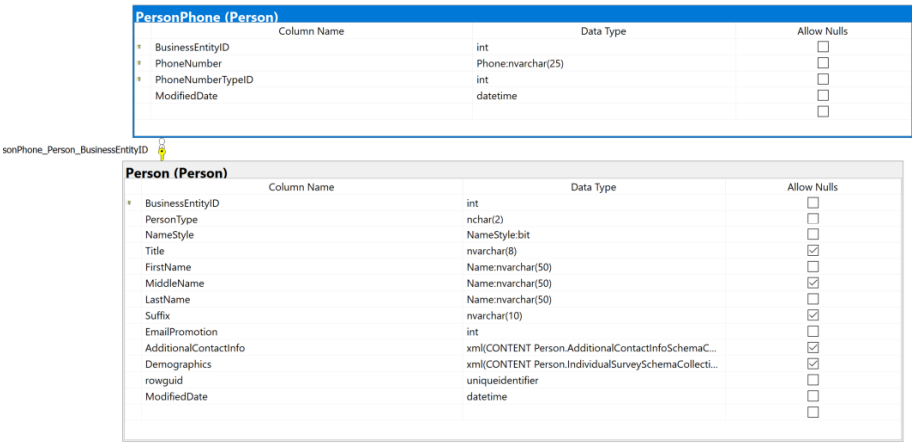
Query executed successfully.

Sample JSON Output (316 results returned):

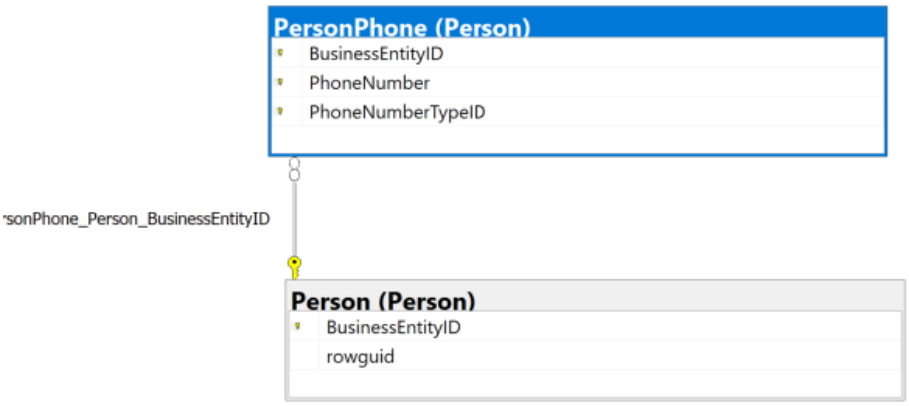


SIMPLE 4: List all employee names and their phone number

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Person.Person	FirstName LastName
Person.PersonPhone	PhoneNumber

Order By:

Table Name	Column Name	Sort Order
Person.PersonPhone	BusinessEntityId	ASC

Solution without JSON:

```
USE AdventureWorks2017;
SELECT n.FirstName
      ,n.LastName
      ,p.PhoneNumber
FROM Person.PersonPhone AS p
INNER JOIN Person.Person AS n ON p.BusinessEntityID = n.BusinessEntityID
ORDER BY p.BusinessEntityID
```

Solution with JSON:

```
USE AdventureWorks2017;
SELECT n.FirstName
      ,n.LastName
      ,p.PhoneNumber
FROM Person.PersonPhone AS p
INNER JOIN Person.Person AS n ON p.BusinessEntityID = n.BusinessEntityID
ORDER BY p.BusinessEntityID
FOR JSON PATH, ROOT('Simple 4 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (19,972 results returned):

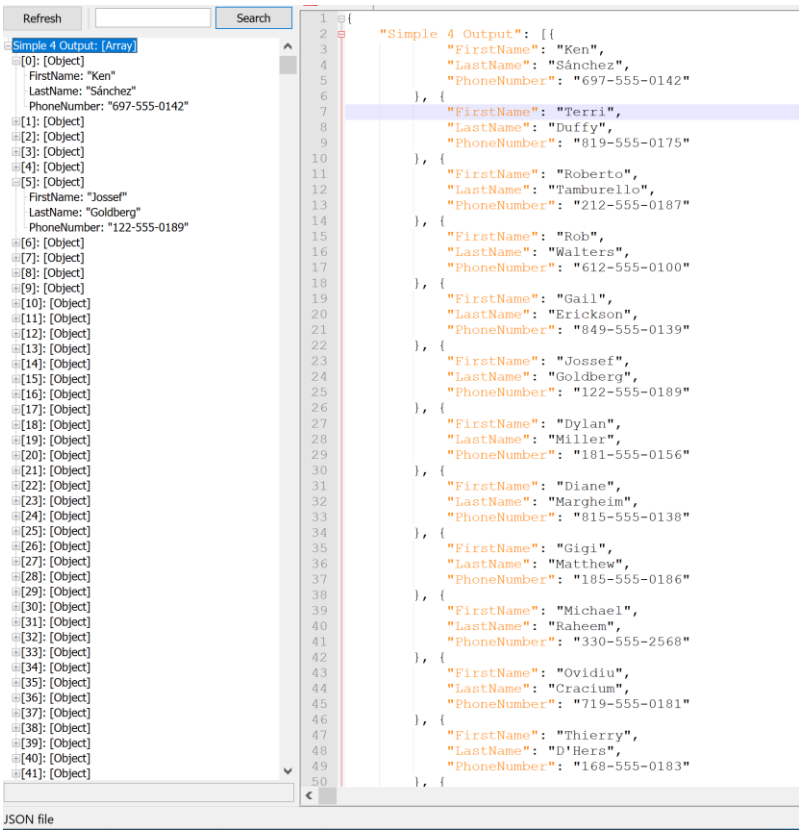
100 %

Results | Messages

	FirstName	LastName	PhoneNumber
1	Ken	Sánchez	697-555-0142
2	Terri	Duffy	819-555-0175
3	Roberto	Tamburello	212-555-0187
4	Rob	Walters	612-555-0100
5	Gail	Erickson	849-555-0139
6	Jossef	Goldberg	122-555-0189
7	Dylan	Miller	181-555-0156
8	Diane	Margheim	815-555-0138
9	Gigi	Matthew	185-555-0186
10	Michael	Raheem	330-555-2568
11	Ovidiu	Cracium	719-555-0181
12	Thierry	D'Hers	168-555-0183
13	Janice	Galvin	473-555-0117
14	Michael	Sullivan	465-555-0156
15	Sharon	Salavaria	970-555-0138
16	David	Bradley	913-555-0172
17	Kevin	Brown	150-555-0189
18	John	Wood	486-555-0150
19	Mary	Dempsey	124-555-0114
20	Wanida	Benshoof	708-555-0141
21	Terry	Eminhizer	138-555-0118
22	Sariya	Harnpad...	399-555-0176
23	Mary	Gibson	531-555-0183
24	Jill	Williams	510-555-0121
25	James	Hamilton	870-555-0122
26	Peter	Krebs	913-555-0196
27	Jo	Brown	632-555-0129
28	Guy	Gilbert	320-555-0195

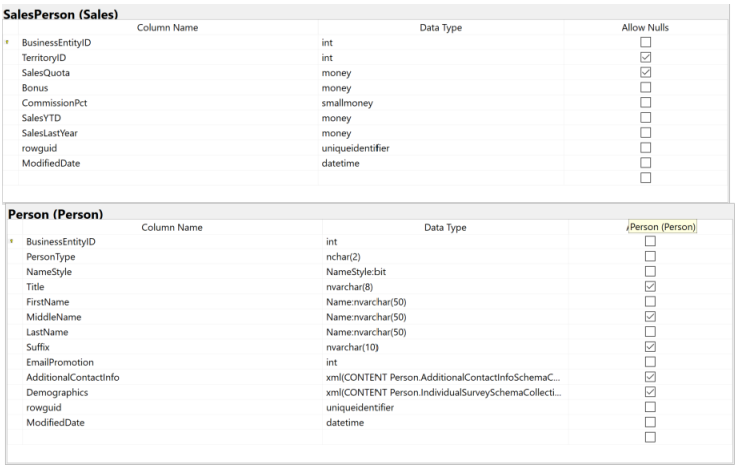
Query executed successfully.

Sample JSON Output (19,972 results returned):



SIMPLE 5: listing sales people in order from previous year earnings

Standard View:



SalesPerson (Sales)

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

Person (Person)

Column Name	Data Type	Person (Person)
BusinessEntityID	int	☐
PersonType	nchar(2)	☐
NameStyle	NameStyle:bit	☐
Title	nvarchar(8)	☑
FirstName	Name:nvarchar(50)	☐
MiddleName	Name:nvarchar(50)	☑
LastName	Name:nvarchar(50)	☐
Suffix	nvarchar(10)	☑
EmailPromotion	int	☐
AdditionalContactInfo	xml(CONTENT Person.AdditionalContactInfoSchemaC...	☑
Demographics	xml(CONTENT Person.IndividualSurveySchemaCollecti...	☑
rowguid	uniqueidentifier	☐
ModifiedDate	datetime	☐

Key View:

Columns from tables:

Table Name	Column Name
Person.Person	BusinessEntityId FirstName LastName
Sales.SalesPerson	SalesYTD SalesLastYear

Order By:

Table Name	Column Name	Sort Order
Sales.SalesPerson	SalesLastYear	DESC

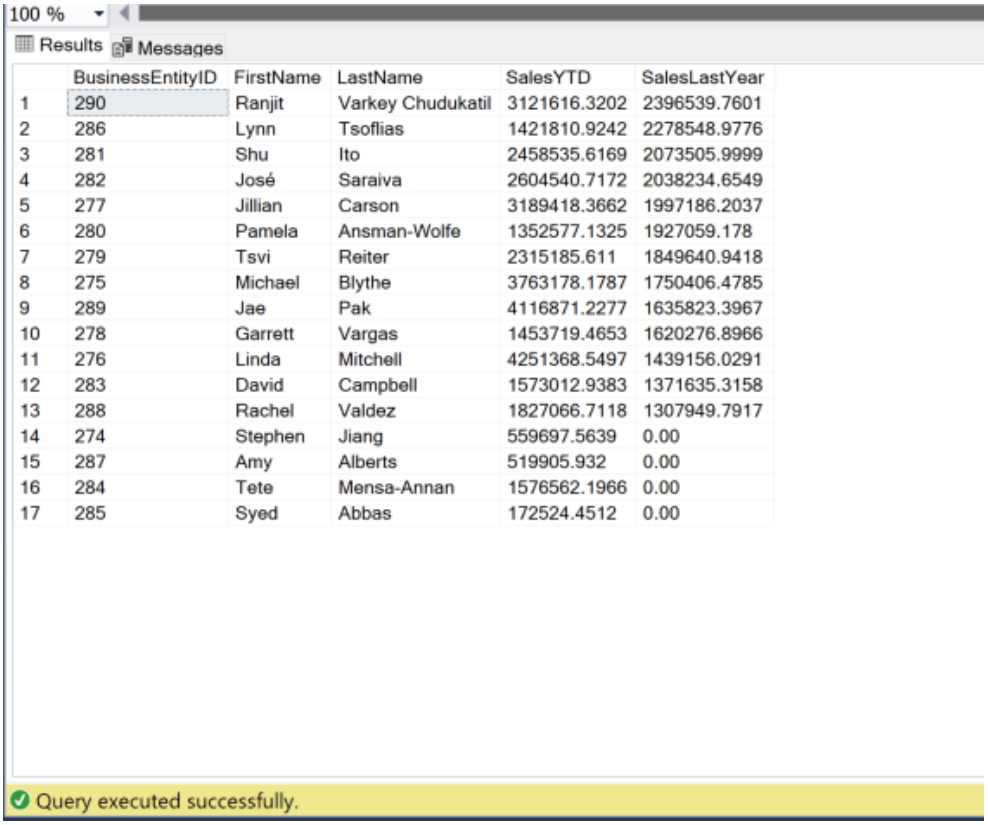
Solution without JSON:

```
USE AdventureWorks2017;
SELECT p.BusinessEntityID
	,p.FirstName
	,p.LastName
	,s.SalesYTD
	,s.SalesLastYear
FROM Sales.SalesPerson AS s
INNER JOIN Person.Person AS p ON s.BusinessEntityID = p.BusinessEntityID
ORDER BY SalesLastYear DESC
```

Solution with JSON:

```
USE AdventureWorks2017;
SELECT p.BusinessEntityID
	,p.FirstName
	,p.LastName
	,s.SalesYTD
	,s.SalesLastYear
FROM Sales.SalesPerson AS s
INNER JOIN Person.Person AS p ON s.BusinessEntityID = p.BusinessEntityID
ORDER BY SalesLastYear DESC
FOR JSON PATH, ROOT('Simple 5 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (17 results returned):



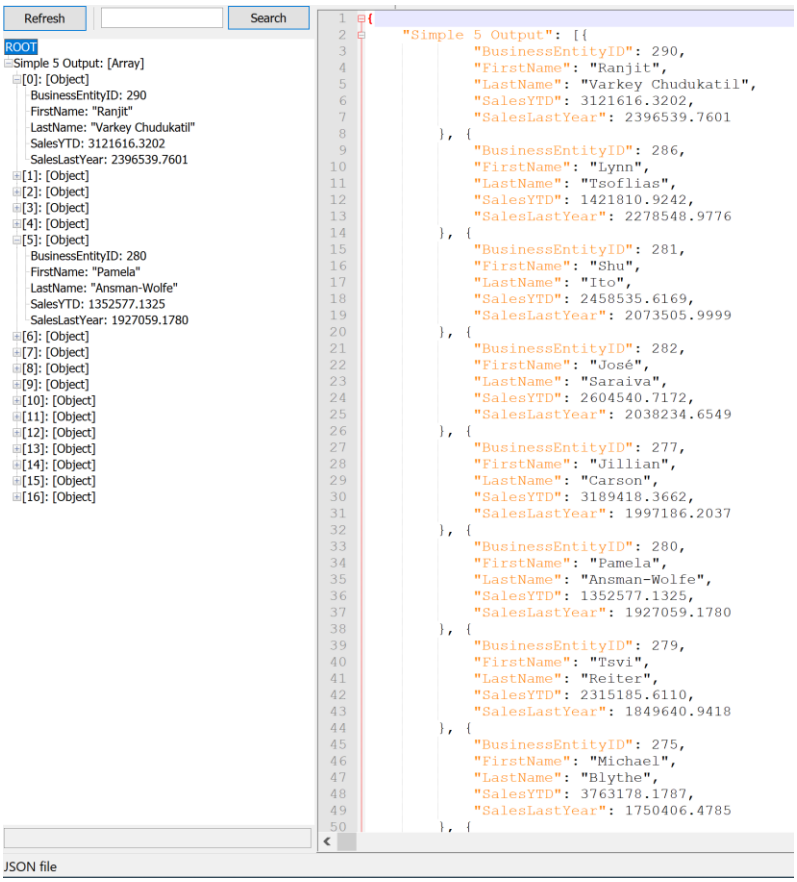
100 %

Results | Messages

	BusinessEntityID	FirstName	LastName	SalesYTD	SalesLastYear
1	290	Ranjit	Varkey Chudukatil	3121616.3202	2396539.7601
2	286	Lynn	Tsoflias	1421810.9242	2278548.9776
3	281	Shu	Ito	2458535.6169	2073505.9999
4	282	José	Saraiva	2604540.7172	2038234.6549
5	277	Jillian	Carson	3189418.3662	1997186.2037
6	280	Pamela	Ansman-Wolfe	1352577.1325	1927059.178
7	279	Tsvi	Reiter	2315185.611	1849640.9418
8	275	Michael	Blythe	3763178.1787	1750406.4785
9	289	Jae	Pak	4116871.2277	1635823.3967
10	278	Garrett	Vargas	1453719.4653	1620276.8966
11	276	Linda	Mitchell	4251368.5497	1439156.0291
12	283	David	Campbell	1573012.9383	1371635.3158
13	288	Rachel	Valdez	1827066.7118	1307949.7917
14	274	Stephen	Jiang	559697.5639	0.00
15	287	Amy	Alberts	519905.932	0.00
16	284	Tete	Mensa-Annan	1576562.1966	0.00
17	285	Syed	Abbas	172524.4512	0.00

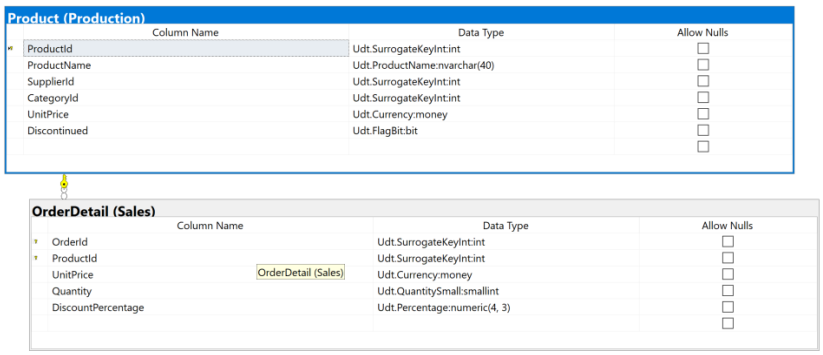
Query executed successfully.

Sample JSON Output (17 results returned):

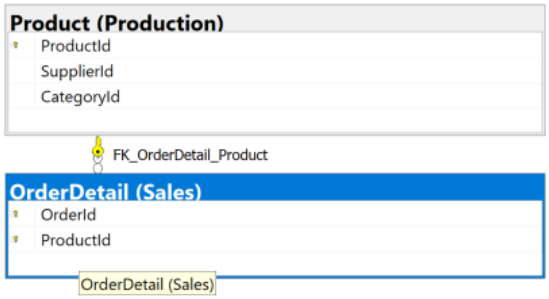


MEDIUM 1: find the product name and Id that was purchased the most amongst all customers

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Production.Product	ProductName ProductId
Sales.OrderDetails	ProductId

Order By:

Table Name	Column Name	Sort Order
Sales.OrderDetails	NumSales	DESC

Solution without JSON:

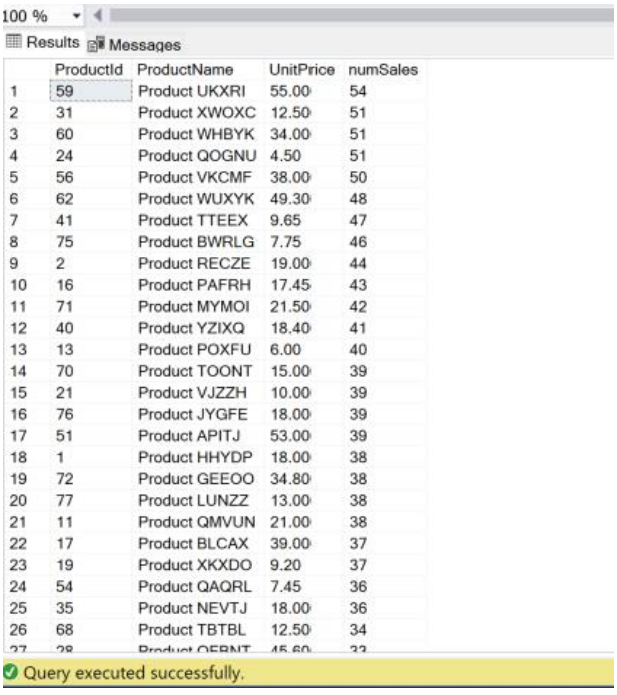
```
USE Northwinds2020TSQLV6;
SELECT o.ProductId
        ,p.ProductName
        ,p.UnitPrice
        ,COUNT(*) AS numSales
FROM Sales.[OrderDetail] AS O
INNER JOIN Production.Product AS P ON o.ProductId = p.ProductId
GROUP BY o.ProductId
        ,p.ProductName
        ,p.UnitPrice
```

```
ORDER BY numSales DESC
```

Solution with JSON:

```
USE Northwinds2020TSQLV6;
SELECT o.ProductId
	,p.ProductName
	,p.UnitPrice
	,COUNT(*) AS numSales
FROM Sales.[OrderDetail] AS O
INNER JOIN Production.Product AS P ON o.ProductId = p.ProductId
GROUP BY o.ProductId
	,p.ProductName
	,p.UnitPrice
ORDER BY numSales DESC
FOR JSON PATH, ROOT('Medium 1 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (77 results returned):



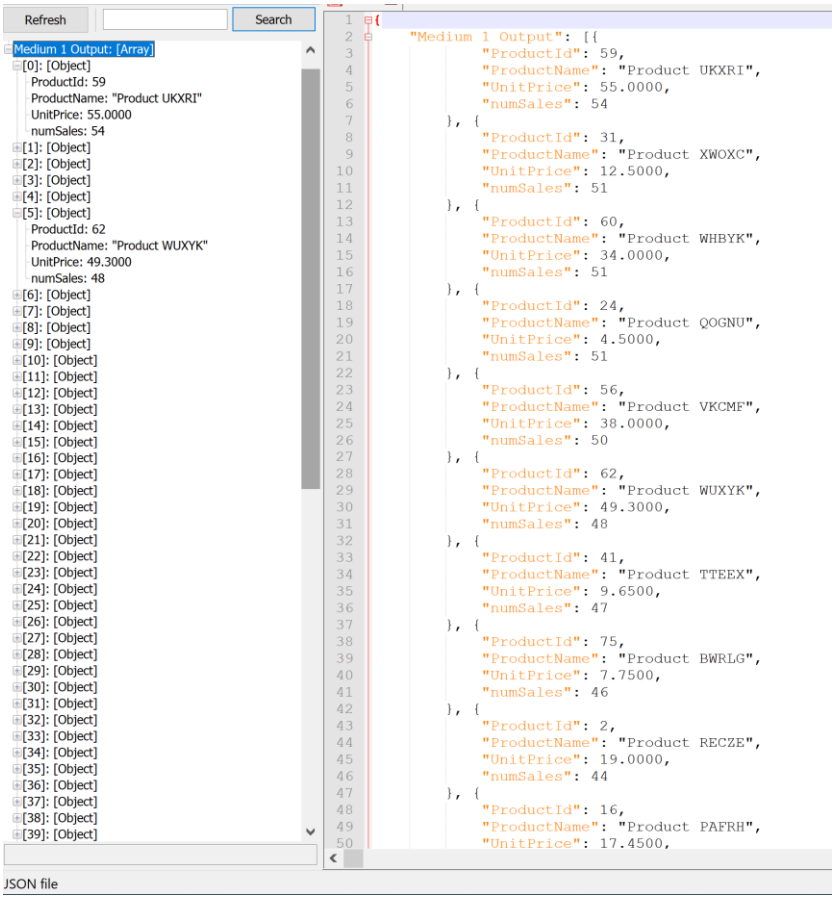
100 %

Results Messages

	ProductId	ProductName	UnitPrice	numSales
1	59	Product UKXRI	55.00	54
2	31	Product XWOXC	12.50	51
3	60	Product WHBYK	34.00	51
4	24	Product QOGNU	4.50	51
5	56	Product VKCMF	38.00	50
6	62	Product WUXYK	49.30	48
7	41	Product TTEEX	9.65	47
8	75	Product BWRLG	7.75	46
9	2	Product RECZE	19.00	44
10	16	Product PAFRH	17.45	43
11	71	Product MYMOI	21.50	42
12	40	Product YZIXQ	18.40	41
13	13	Product POXFU	6.00	40
14	70	Product TOONT	15.00	39
15	21	Product VJZZH	10.00	39
16	76	Product JYGFE	18.00	39
17	51	Product APITJ	53.00	39
18	1	Product HHYDP	18.00	38
19	72	Product GEEOO	34.80	38
20	77	Product LUNZZ	13.00	38
21	11	Product QMVUN	21.00	38
22	17	Product BLCAX	39.00	37
23	19	Product XKXDO	9.20	37
24	54	Product QAQRL	7.45	36
25	35	Product NEVTJ	18.00	36
26	68	Product TBTBL	12.50	34
27	28	Product QERNT	45.60	33

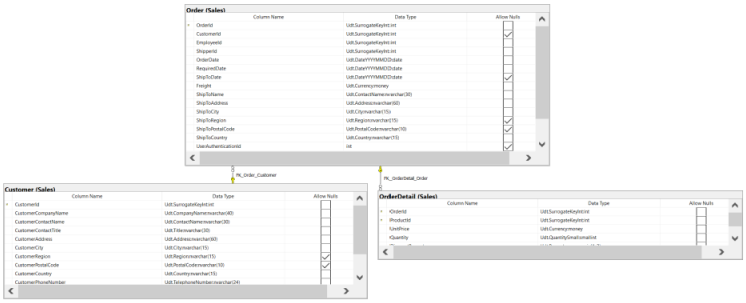
Query executed successfully.

Sample JSON Output (77 results returned):

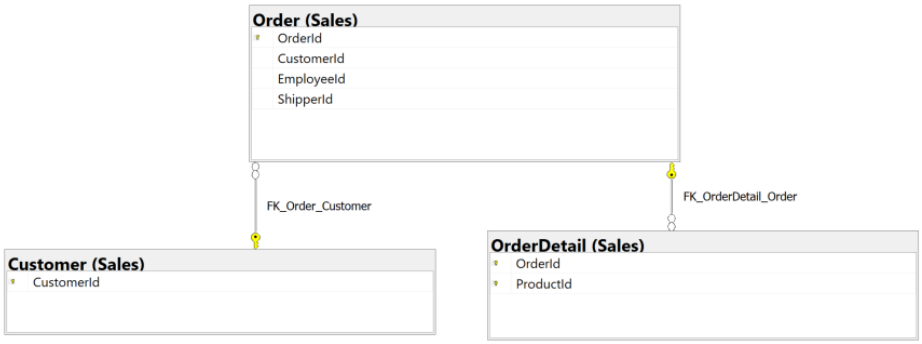


MEDIUM 2: This query lists the customers who spent the most on purchasing products.

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Sales.Order	CustomerId
Sales.Customer	CustomerContactName

Order By:

Table Name	Colum Name	Sort Order
Sales.OrderDetail	TotalSpent	DESC

Solution without JSON:

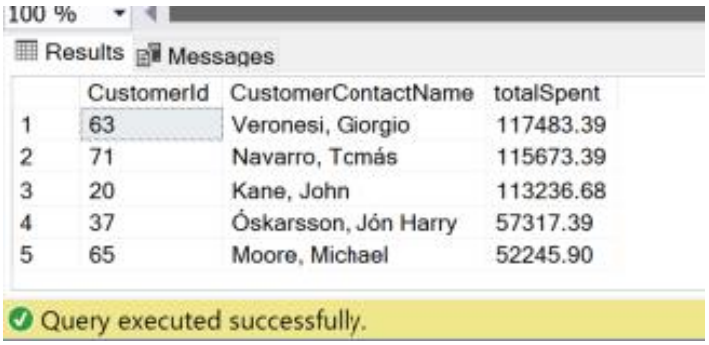
```
USE Northwinds2020TSQLV6;

SELECT TOP 5 o.CustomerId
```

```
        ,c.CustomerContactName
        ,sum(d.UnitPrice * d.Quantity) AS totalSpent
FROM Sales.OrderDetail AS d
INNER JOIN Sales.[Order] AS o ON d.OrderId = o.OrderId
INNER JOIN Sales.Customer AS c ON o.CustomerId = c.CustomerId
GROUP BY o.customerId
        ,c.customerContactName
ORDER BY totalSpent DESC;
```

Solution with JSON:

```
USE Northwinds2020TSQLV6;
SELECT TOP 5 o.CustomerId
        ,c.CustomerContactName
        ,sum(d.UnitPrice * d.Quantity) AS totalSpent
FROM Sales.OrderDetail AS d
INNER JOIN Sales.[Order] AS o ON d.OrderId = o.OrderId
INNER JOIN Sales.Customer AS c ON o.CustomerId = c.CustomerId
GROUP BY o.customerId
        ,c.customerContactName
ORDER BY totalSpent DESC;
FOR JSON PATH, ROOT('Medium 2 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (5 results returned):

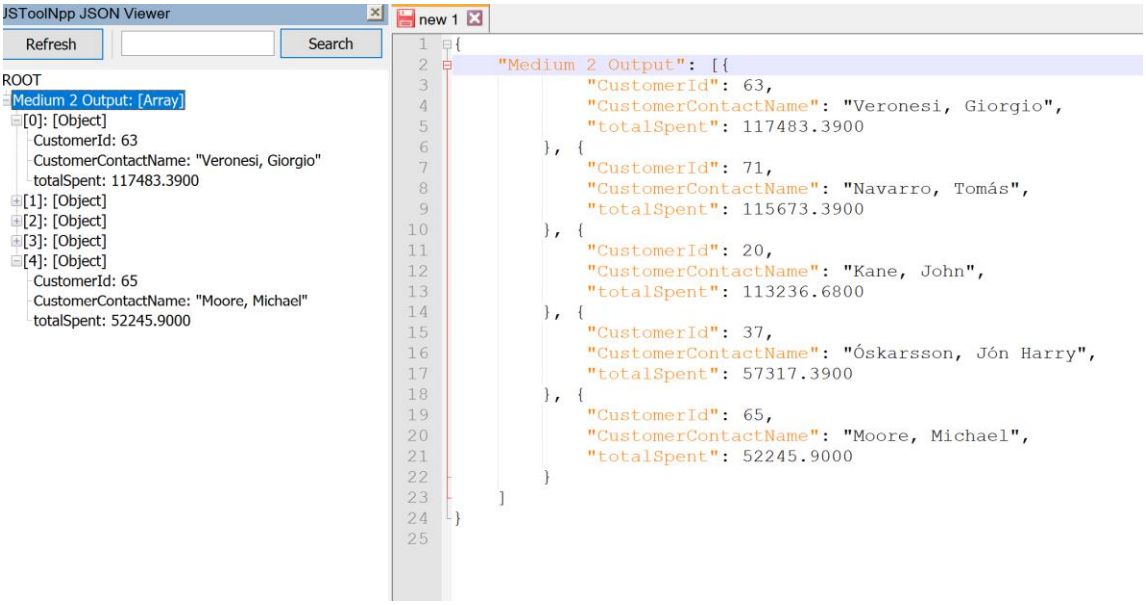

100 %

Results Messages

	CustomerId	CustomerContactName	totalSpent
1	63	Veronesi, Giorgio	117483.39
2	71	Navarro, Tomás	115673.39
3	20	Kane, John	113236.68
4	37	Óskarsson, Jón Harry	57317.39
5	65	Moore, Michael	52245.90

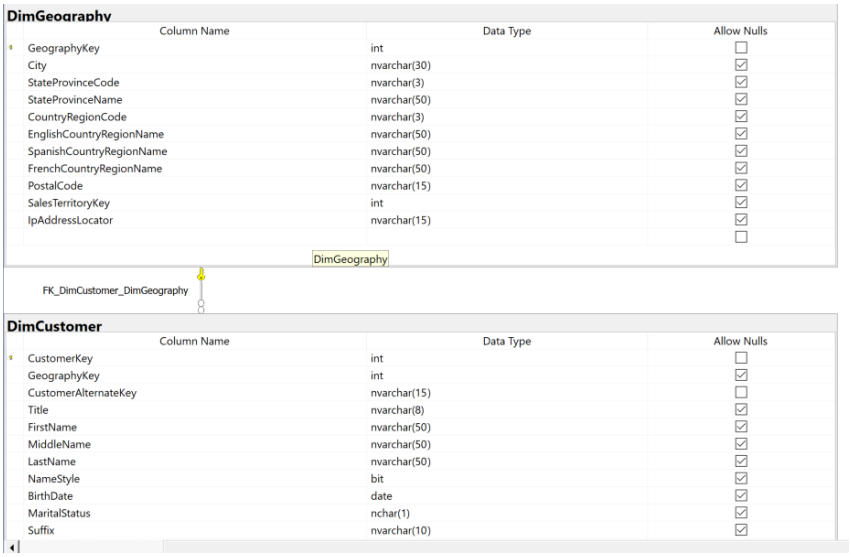
Query executed successfully.

Sample JSON Output (5 results returned):

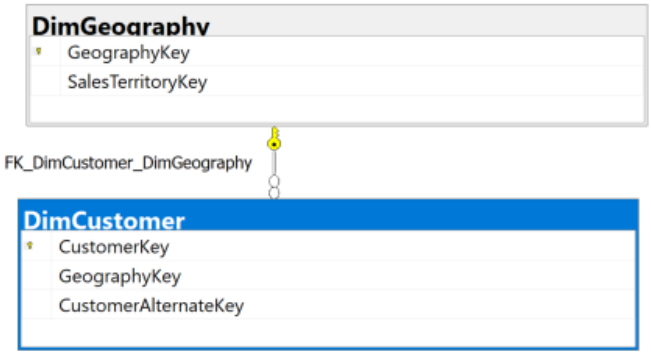


MEDIUM 3: This query finds the city where most customers reside and lists all customers who live there

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Dbo.DimCustomer	FirstName LastName EmailAddress BirthDate Gender YearlyIncome
Dbo.DimGeography	City

Order By:

Table Name	Column Name	Sort Order
Dbo.DimCustomer	YearlyIncome	DESC

Solution without JSON:

```
USE AdventureWorksDW2017;

SELECT g.city
        ,c.firstname
        ,c.lastname
        ,c.emailaddress
        ,c.birthdate
        ,c.gender
```

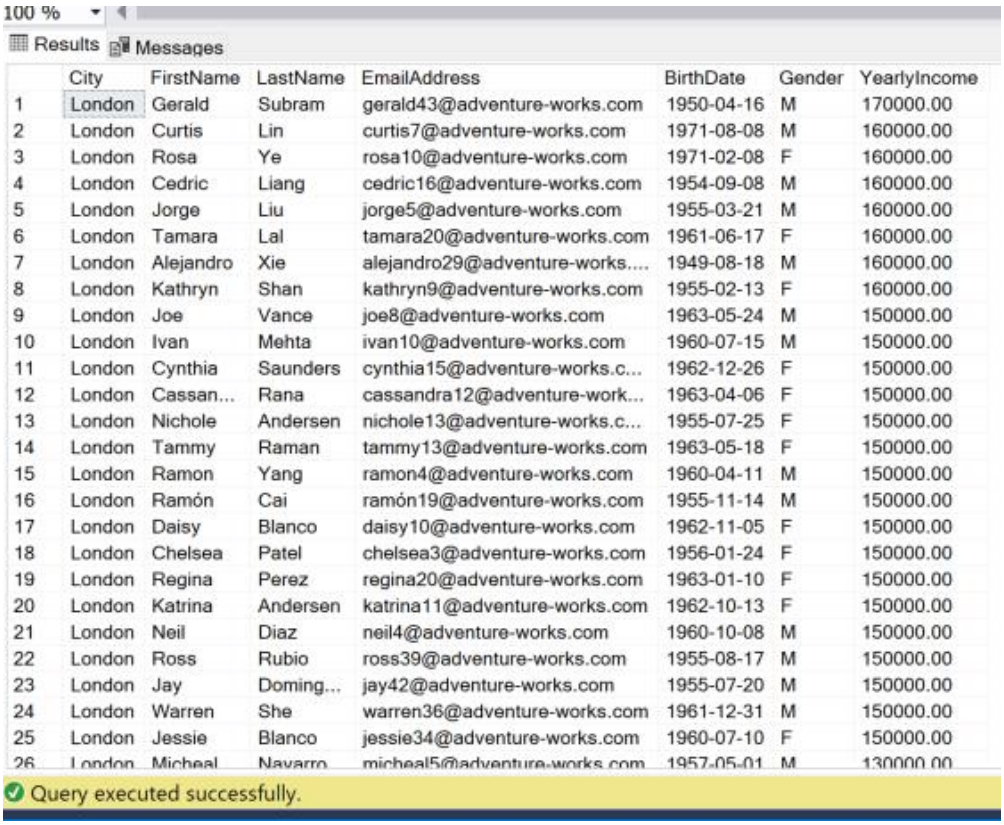
```
	,c.yearlyincome
FROM dbo.dimCustomer AS c
INNER JOIN dbo.dimGeography AS g ON c.geographyKey = g.GEOGRAPHYkey
WHERE g.city = (
		SELECT city
		FROM (
			SELECT TOP 1 g.city
				,COUNT(*) AS custLocal
			FROM dbo.dimCustomer AS c
			INNER JOIN dbo.dimGeography AS g ON g.geographyKey =
c.geographyKey
			GROUP BY g.city
			ORDER BY custLocal DESC
			) AS gu
		)
ORDER BY yearlyincome DESC
```

Solution with JSON:

```
USE AdventureWorksDW2017;
SELECT g.city
	,c.firstname
	,c.lastname
	,c.emailaddress
	,c.birthdate
	,c.gender
	,c.yearlyincome
FROM dbo.dimCustomer AS c
INNER JOIN dbo.dimGeography AS g ON c.geographyKey = g.GEOGRAPHYkey
WHERE g.city = (
```

```
		SELECT city
		FROM (
			SELECT TOP 1 g.city
				,COUNT(*) AS custLocal
			FROM dbo.dimCustomer AS c
			INNER JOIN dbo.dimGeography AS g ON g.geographyKey = c.geographyKey
			GROUP BY g.city
			ORDER BY custLocal DESC
			) AS gu
		)
ORDER BY yearlyincome DESC
FOR JSON PATH, ROOT('Medium 3 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (420 results returned):



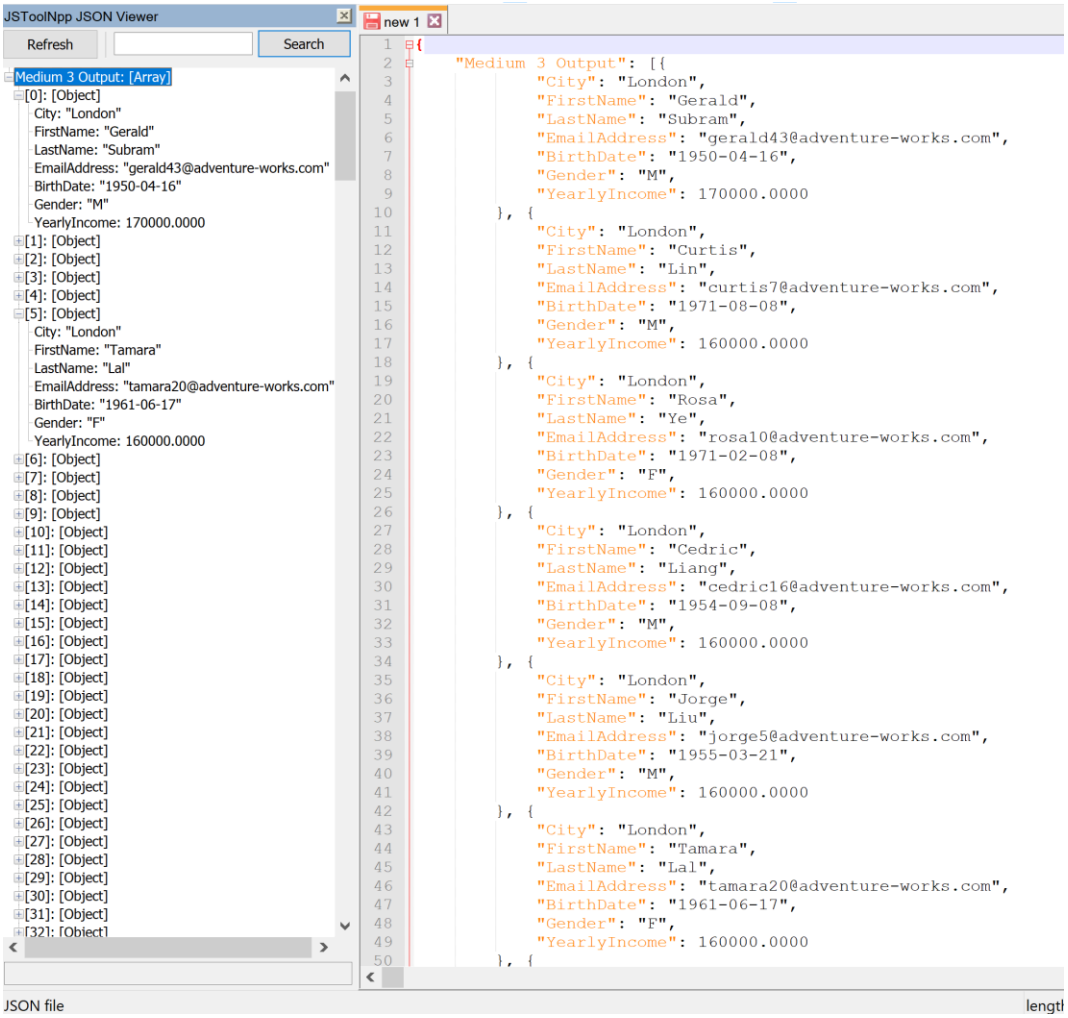
100 %

Results Messages

	City	FirstName	LastName	EmailAddress	BirthDate	Gender	YearlyIncome
1	London	Gerald	Subram	gerald43@adventure-works.com	1950-04-16	M	170000.00
2	London	Curtis	Lin	curtis7@adventure-works.com	1971-08-08	M	160000.00
3	London	Rosa	Ye	rosa10@adventure-works.com	1971-02-08	F	160000.00
4	London	Cedric	Liang	cedric16@adventure-works.com	1954-09-08	M	160000.00
5	London	Jorge	Liu	jorge5@adventure-works.com	1955-03-21	M	160000.00
6	London	Tamara	Lal	tamara20@adventure-works.com	1961-06-17	F	160000.00
7	London	Alejandro	Xie	alejandro29@adventure-works....	1949-08-18	M	160000.00
8	London	Kathryn	Shan	kathryn9@adventure-works.com	1955-02-13	F	160000.00
9	London	Joe	Vance	joe8@adventure-works.com	1963-05-24	M	150000.00
10	London	Ivan	Mehta	ivan10@adventure-works.com	1960-07-15	M	150000.00
11	London	Cynthia	Saunders	cynthia15@adventure-works.c...	1962-12-26	F	150000.00
12	London	Cassan...	Rana	cassandra12@adventure-work...	1963-04-06	F	150000.00
13	London	Nichole	Andersen	nichole13@adventure-works.c...	1955-07-25	F	150000.00
14	London	Tammy	Raman	tammy13@adventure-works.com	1963-05-18	F	150000.00
15	London	Ramon	Yang	ramon4@adventure-works.com	1960-04-11	M	150000.00
16	London	Ramón	Cai	ramón19@adventure-works.com	1955-11-14	M	150000.00
17	London	Daisy	Blanco	daisy10@adventure-works.com	1962-11-05	F	150000.00
18	London	Chelsea	Patel	chelsea3@adventure-works.com	1956-01-24	F	150000.00
19	London	Regina	Perez	regina20@adventure-works.com	1963-01-10	F	150000.00
20	London	Katrina	Andersen	katrina11@adventure-works.com	1962-10-13	F	150000.00
21	London	Neil	Diaz	neil4@adventure-works.com	1960-10-08	M	150000.00
22	London	Ross	Rubio	ross39@adventure-works.com	1955-08-17	M	150000.00
23	London	Jay	Doming...	jay42@adventure-works.com	1955-07-20	M	150000.00
24	London	Warren	She	warren36@adventure-works.com	1961-12-31	M	150000.00
25	London	Jessie	Blanco	jessie34@adventure-works.com	1960-07-10	F	150000.00
26	London	Micheal	Navarro	micheal5@adventure-works.com	1957-05-01	M	130000.00

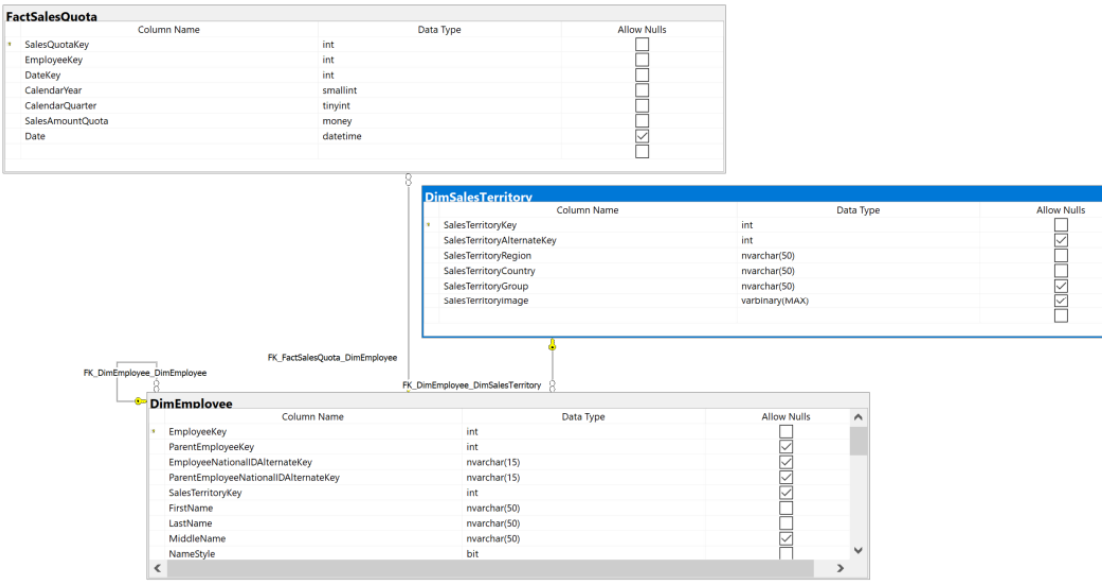
Query executed successfully.

Sample JSON Output (420 results returned):

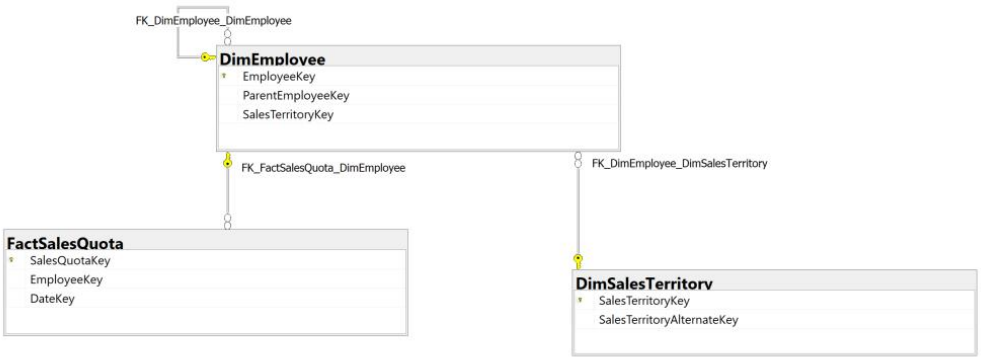


MEDIUM 4: This query finds the salaried employees who had the highest sales quotas to meet from 2010-2013

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Dbo.DimEmployee	Title EmployeeKey FirstName LastName
Dbo.FactSalesQuota	CalendarYear SalesAmountQuota
Dbo.DimSalesTerritory	SalesTerritoryRegion SalesTerritoryCountry

Order By:

Table Name	Column Name	Sort Order
Dbo.FactSalesQuota	SalesAmountQuota	DESC

Solution without JSON:

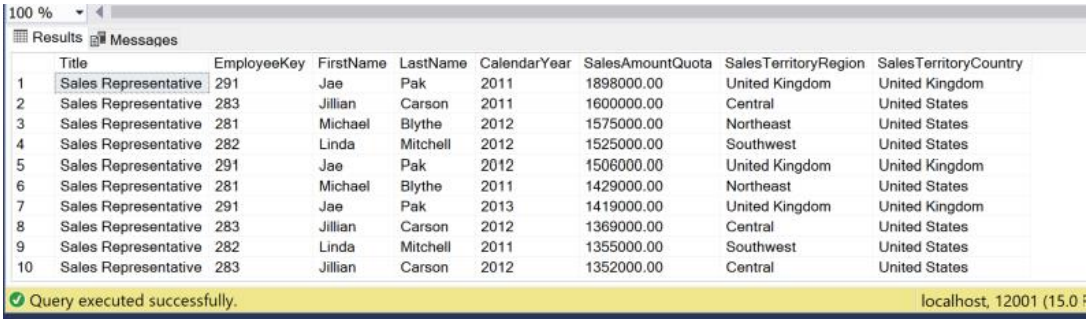
```
USE AdventureWorksDW2017;
SELECT TOP 10 e.title
	,e.employeekey
	,e.firstName
	,e.lastName
	,q.calendarYear
	,q.salesamountquota
	,t.salesterritoryregion
	,t.salesterritorycountry
FROM dbo.dimemployee AS e
INNER JOIN dbo.factsalesquota AS q ON e.employeekey = q.employeekey
INNER JOIN dbo.dimsalesterritory AS t ON e.salesterritorykey = t.salesterritorykey
WHERE e.salariedflag = 1
ORDER BY salesamountquota DESC
```

Solution with JSON:

```
USE AdventureWorksDW2017;
SELECT TOP 10 e.title
	,e.employeekey
	,e.firstName
	,e.lastName
	,q.calendarYear
```

```
        ,q.salesamountquota
        ,t.salesterritoryregion
        ,t.salesterritorycountry
FROM dbo.dimemployee AS e
INNER JOIN dbo.factsalesquota AS q ON e.employeekey = q.employeekey
INNER JOIN dbo.dimsalesterritory AS t ON e.salesterritorykey = t.salesterritorykey
WHERE e.salariedflag = 1
ORDER BY salesamountquota DESC
FOR JSON PATH, ROOT('Medium 4 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (10 results returned):

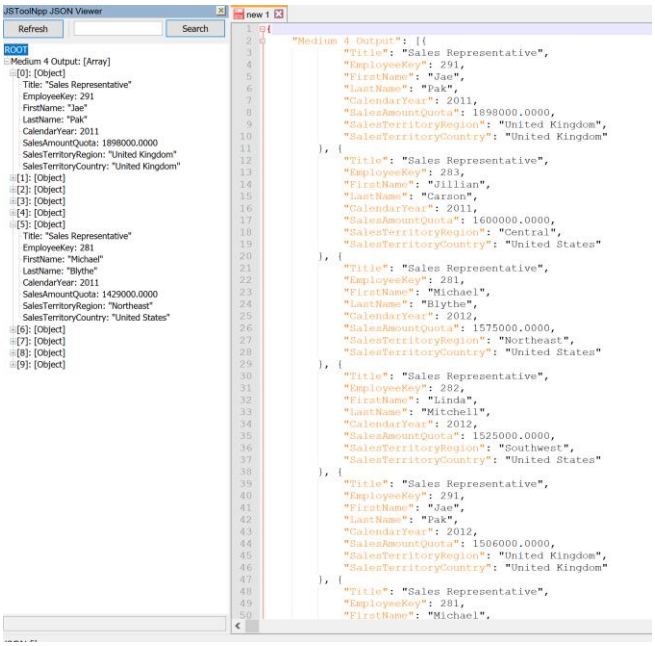


Results Messages

	Title	EmployeeKey	FirstName	LastName	CalendarYear	SalesAmountQuota	SalesTerritoryRegion	SalesTerritoryCountry
1	Sales Representative	291	Jae	Pak	2011	1898000.00	United Kingdom	United Kingdom
2	Sales Representative	283	Jillian	Carson	2011	1600000.00	Central	United States
3	Sales Representative	281	Michael	Blythe	2012	1575000.00	Northeast	United States
4	Sales Representative	282	Linda	Mitchell	2012	1525000.00	Southwest	United States
5	Sales Representative	291	Jae	Pak	2012	1506000.00	United Kingdom	United Kingdom
6	Sales Representative	281	Michael	Blythe	2011	1429000.00	Northeast	United States
7	Sales Representative	291	Jae	Pak	2013	1419000.00	United Kingdom	United Kingdom
8	Sales Representative	283	Jillian	Carson	2012	1369000.00	Central	United States
9	Sales Representative	282	Linda	Mitchell	2011	1355000.00	Southwest	United States
10	Sales Representative	283	Jillian	Carson	2012	1352000.00	Central	United States

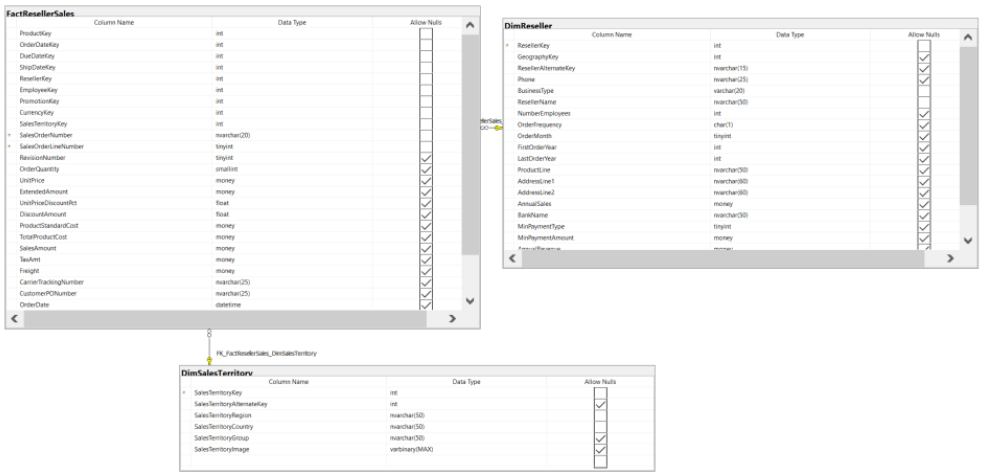
Query executed successfully. localhost, 12001 (15.0

Sample JSON Output (10 results returned):

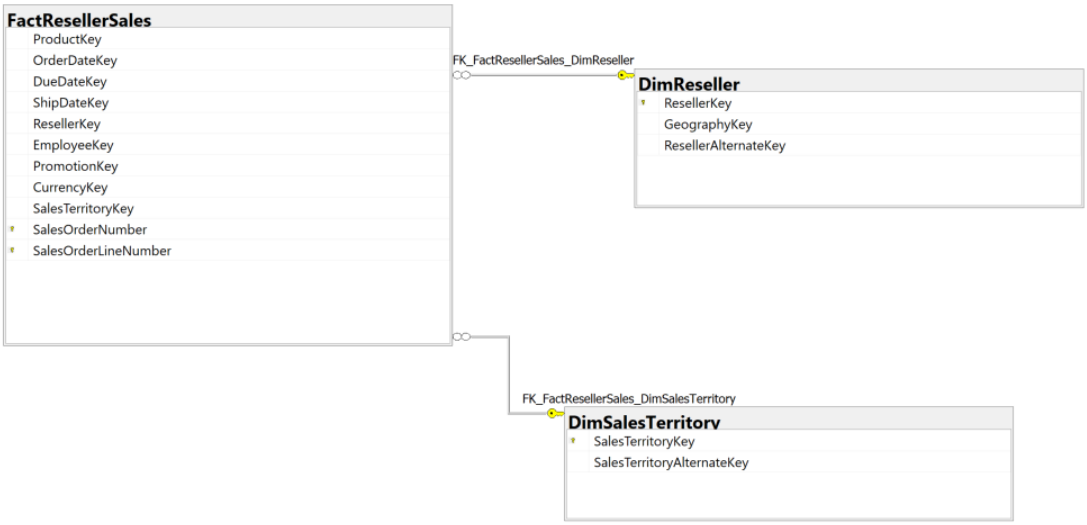


MEDIUM 5: This query finds the resellers who sold the most value and lists some information about each reseller

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Dbo.DimReseller	ResellerName AnnualSales NumberEmployees
Dbo.DimSalesTerritory	SalesTerritoryCountry SalesTerritoryRegion

Order By:

Table Name	Column Name	Sort Order
Dbo.DimReseller	AnnualSales	DESC

Solution without JSON:

```
USE AdventureWorksDW2017;
SELECT sales.resellerkey
	,rs.resellername
	,rs.annualsales
	,rs.numberemployees
	,terr.salesterritorycountry
	,terr.salesterritoryregion
FROM dbo.dimreseller AS rs
INNER JOIN dbo.factresellersales AS sales ON rs.resellerkey = sales.resellerkey
INNER JOIN dbo.dimsalesterritory AS terr ON sales.salesterritorykey = terr.salesterritorykey
WHERE rs.annualsales = (
		SELECT max(annualsales)
		FROM dbo.dimreseller
		)
GROUP BY sales.resellerkey
	,rs.resellername
	,rs.annualsales
	,rs.numberemployees
```

```
        ,terr.salesterritorycountry
        ,terr.salesterritoryregion
```

Solution with JSON:

```
USE AdventureWorksDW2017;
SELECT sales.resellerkey
        ,rs.resellername
        ,rs.annualsales
        ,rs.numberemployees
        ,terr.salesterritorycountry
        ,terr.salesterritoryregion
FROM dbo.dimreseller AS rs
INNER JOIN dbo.factresellersales AS sales ON rs.resellerkey = sales.resellerkey
INNER JOIN dbo.dimsalesterritory AS terr ON sales.salesterritorykey = terr.salesterritorykey
WHERE rs.annualsales = (
                SELECT max(annualsales)
                FROM dbo.dimreseller
                )
GROUP BY sales.resellerkey
        ,rs.resellername
        ,rs.annualsales
        ,rs.numberemployees
        ,terr.salesterritorycountry
        ,terr.salesterritoryregion
FOR JSON PATH, ROOT('Medium 5 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (193 results returned):

Results Messages

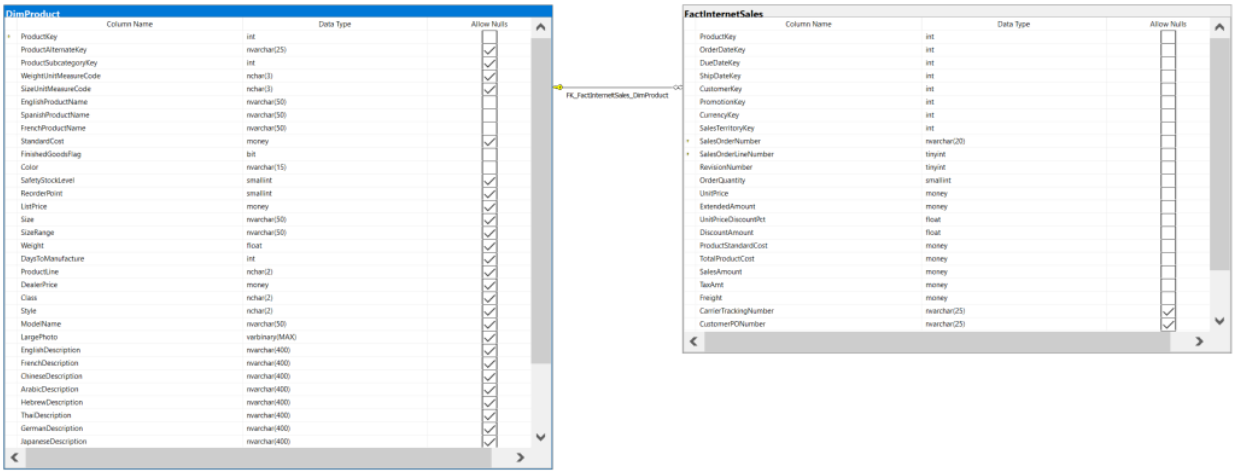
	ResellerKey	ResellerName	AnnualSales	NumberEmployees	SalesTerritoryCountry	SalesTerritoryRegion
1	15	Budget Toy Store	3000000.00	52	Australia	Australia
2	18	Catalog Store	3000000.00	55	United States	Central
3	21	Chic Department Stores	3000000.00	58	United States	Southwest
4	24	Eastside Department Store	3000000.00	61	United States	Southwest
5	27	Sports Sales and Rental	3000000.00	64	United States	Southeast
6	30	Cycle Merchants	3000000.00	67	Canada	Canada
7	33	Global Sports Outlet	3000000.00	70	Australia	Australia
8	36	Exotic Bikes	3000000.00	73	United States	Northeast
9	39	Fitness Hotel	3000000.00	76	United States	Central
10	45	Every Bike Shop	3000000.00	80	United States	Southeast
11	48	Grand Industries	3000000.00	81	Canada	Canada
12	51	Ideal Components	3000000.00	82	Australia	Australia
13	54	Larger Cycle Shop	3000000.00	83	United States	Northeast
14	57	Leading Sales & Repair	3000000.00	84	United States	Central
15	63	Metro Bike Mart	3000000.00	86	United States	Southeast
16	66	Neighborhood Store	3000000.00	87	Canada	Canada
17	69	Online Bike Catalog	3000000.00	88	Australia	Australia
18	72	Outdoor Equipment Store	3000000.00	89	United States	Northeast
19	75	Paint Supply	3000000.00	90	United States	Southwest
20	78	Preferred Bikes	3000000.00	91	United States	Southwest
21	81	Rally Day Mall	3000000.00	92	United States	Southeast
22	84	Rewarding Activities Co...	3000000.00	93	Canada	Canada
23	87	Rich Department Store	3000000.00	94	Australia	Australia
24	90	Sales and Supply Comp...	3000000.00	95	United States	Southeast
25	93	Stationary Bikes and Sta...	3000000.00	96	United States	Southwest
26	96	More Bikes!	3000000.00	97	United States	Southwest
27	99	Unified Sports Company	3000000.00	98	United States	Southeast
28	102	National Manufacturing	3000000.00	99	Canada	Canada

Query executed successfully.

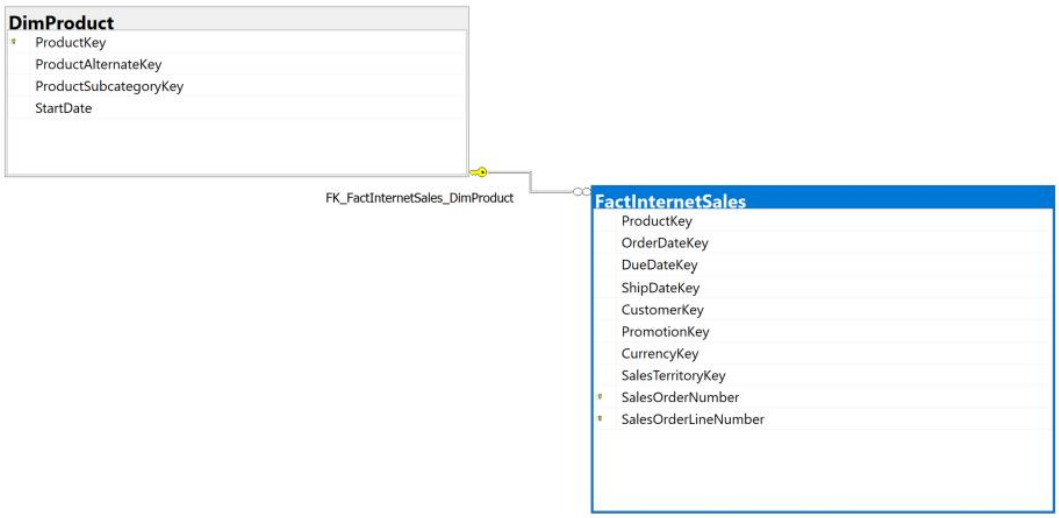
Sample JSON Output:

MEDIUM 6: This query List the most popular items from internet sales and give some details

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Dbo.FactInternetSales	ProductKey UnitPrice
Dbo.DimProduct	EnglishProductName

Order By:

Table Name	Column Name	Sort Order
Dbo.FactInternetSales	NumSales	Desc

Solution without JSON:

```
USE AdventureWorksDW2017;
SELECT sales.productkey
        ,prod.englishproductname
        ,sales.unitprice
        ,count(*) AS NumSales
        ,sales.unitprice * count(*) AS ValueSold
FROM dbo.factinternetsales AS sales
INNER JOIN dbo.dimproduct AS prod ON sales.productkey = prod.productkey
```

```
GROUP BY sales.productkey
        ,prod.englishproductname
        ,sales.unitprice
ORDER BY count(*) DESC
```

Solution with JSON:

```
USE AdventureWorksDW2017;
SELECT sales.productkey
        ,prod.englishproductname
        ,sales.unitprice
        ,count(*) AS NumSales
        ,sales.unitprice * count(*) AS ValueSold
FROM dbo.factinternetsales AS sales
INNER JOIN dbo.dimproduct AS prod ON sales.productkey = prod.productkey
GROUP BY sales.productkey
        ,prod.englishproductname
        ,sales.unitprice
ORDER BY count(*) DESC
FOR JSON PATH, ROOT('Medium 6 Output'), INCLUDE_NULL_VALUES;
```

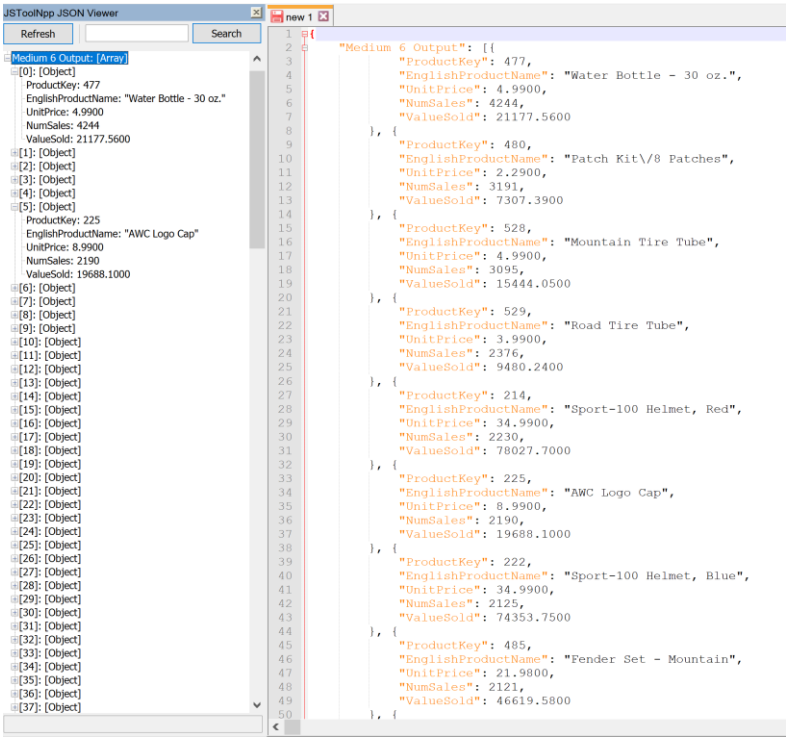
Sample Output (158 results returned):

Results | Messages

	ProductKey	EnglishProductName	UnitPrice	NumSales	ValueSold
1	477	Water Bottle - 30 oz.	4.99	4244	21177.56
2	480	Patch Kit/8 Patches	2.29	3191	7307.39
3	528	Mountain Tire Tube	4.99	3095	15444.05
4	529	Road Tire Tube	3.99	2376	9480.24
5	214	Sport-100 Helmet, Red	34.99	2230	78027.70
6	225	AWC Logo Cap	8.99	2190	19688.10
7	222	Sport-100 Helmet, Blue	34.99	2125	74353.75
8	485	Fender Set - Mountain	21.98	2121	46619.58
9	217	Sport-100 Helmet, Black	34.99	2085	72954.15
10	478	Mountain Bottle Cage	9.99	2025	20229.75
11	479	Road Bottle Cage	8.99	1712	15390.88
12	530	Touring Tire Tube	4.99	1488	7425.12
13	537	HL Mountain Tire	35.00	1396	48860.00
14	536	ML Mountain Tire	29.99	1161	34818.39
15	538	LL Road Tire	21.49	1044	22435.56
16	541	Touring Tire	28.99	935	27105.65
17	539	ML Road Tire	24.99	926	23140.74
18	484	Bike Wash - Dissolver	7.95	908	7218.60
19	535	LL Mountain Tire	24.99	862	21541.38

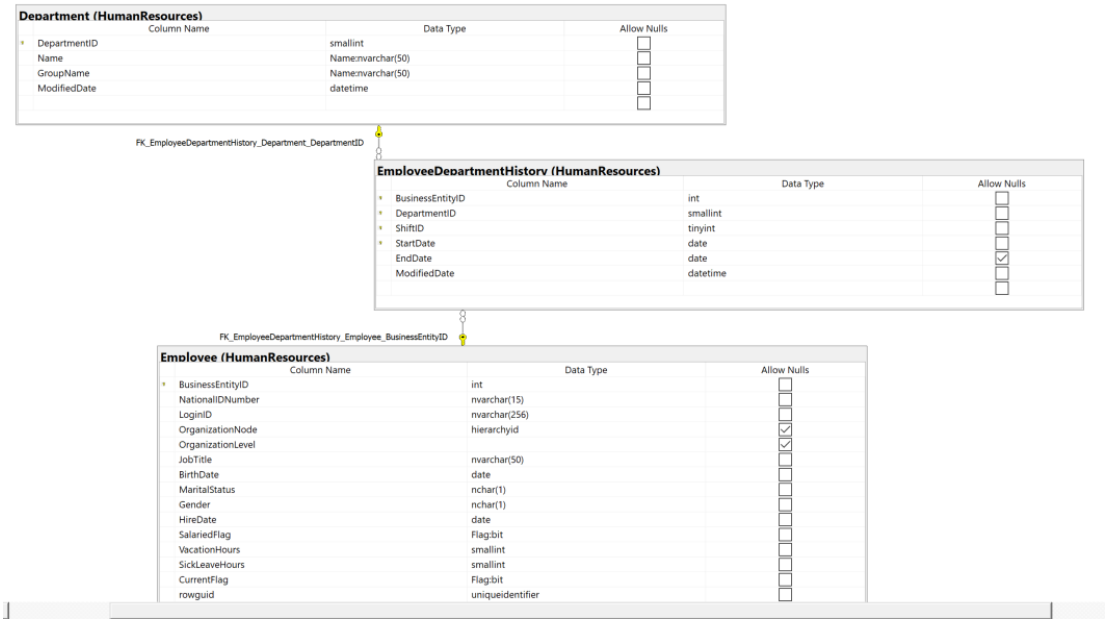
Query executed successfully.

Sample JSON Output (158 results returned):

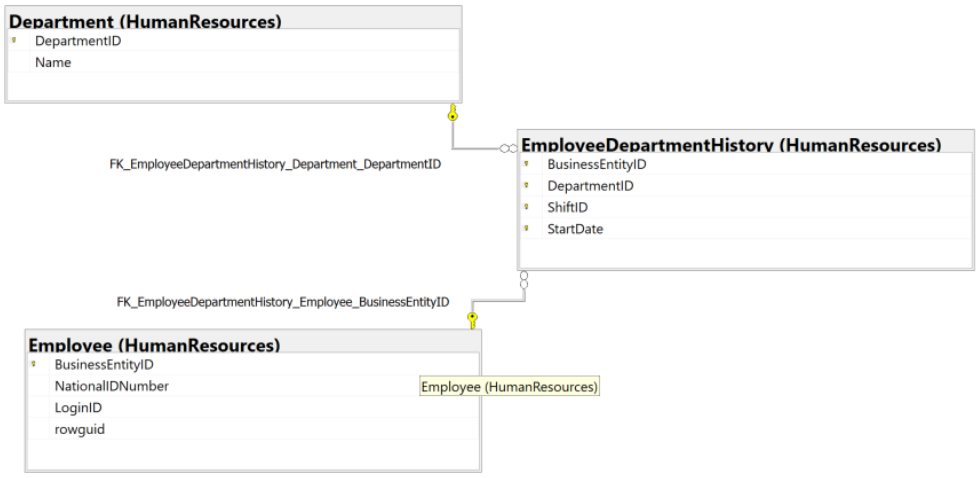


MEDIUM 7: This query orders departments based on the number of employees, possibly if layoffs are needed

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
HumanResources.EmployeeDepartementHistory	DepartementId Name
HumanResources.Employee	NumEmployees

Order By:

Table Name	Column Name	Sort Order
HumanResources.Employee	NumEmployees	DESC

Solution without JSON:

```
USE AdventureWorks2017;

SELECT depohist.departmentid

	,depo.name AS DepartmentName

	,count(e.BusinessEntityID) AS NumEmployees

FROM HumanResources.employee AS E

INNER JOIN HumanResources.employeedepartmenthistory AS depohist ON
e.BusinessEntityID = depohist.BusinessEntityID

INNER JOIN humanresources.department AS depo ON depohist.departmentid =
depo.departmentid

GROUP BY depohist.departmentid

	,depo.name

ORDER BY count(e.BusinessEntityID) DESC
```

Solution with JSON:

```
USE AdventureWorks2017;

SELECT depohist.departmentid

	,depo.name AS DepartmentName

	,count(e.BusinessEntityID) AS NumEmployees

FROM HumanResources.employee AS E

INNER JOIN HumanResources.employeedepartmenthistory AS depohist ON
e.BusinessEntityID = depohist.BusinessEntityID

INNER JOIN humanresources.department AS depo ON depohist.departmentid =
depo.departmentid

GROUP BY depohist.departmentid

	,depo.name

ORDER BY count(e.BusinessEntityID) DESC
```

```
FOR JSON PATH, ROOT('Medium 7 Output'), INCLUDE_NULL_VALUES;
```

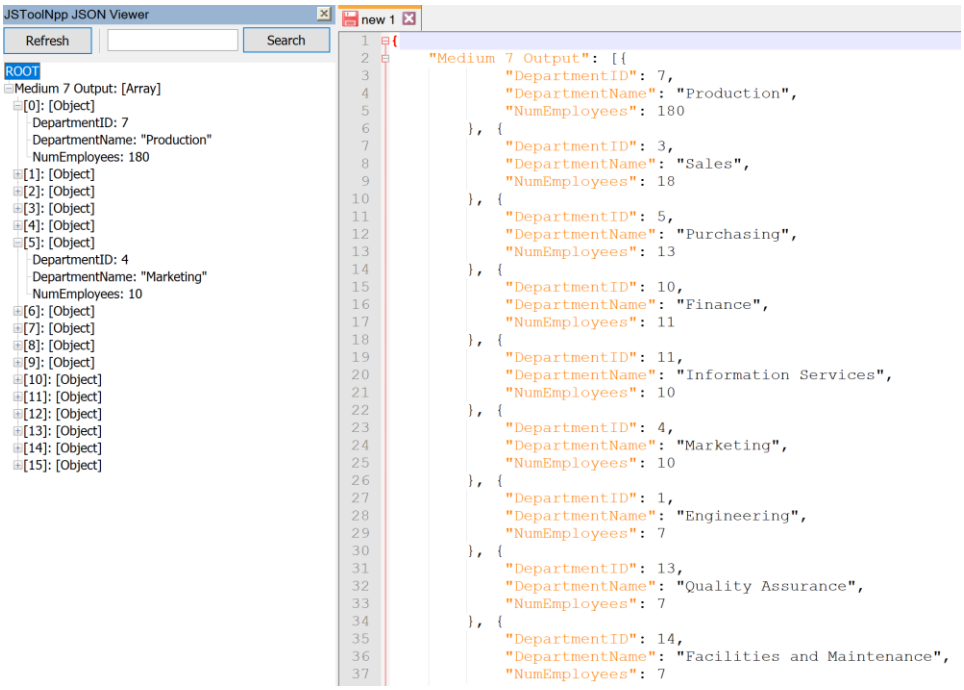
Sample Output (16 results returned):

Results | Messages

	DepartmentID	DepartmentName	NumEmployees
1	7	Production	180
2	3	Sales	18
3	5	Purchasing	13
4	10	Finance	11
5	11	Information Services	10
6	4	Marketing	10
7	1	Engineering	7
8	13	Quality Assurance	7
9	14	Facilities and Maintenance	7
10	15	Shipping and Receiving	6
11	8	Production Control	6
12	9	Human Resources	6
13	12	Document Control	5

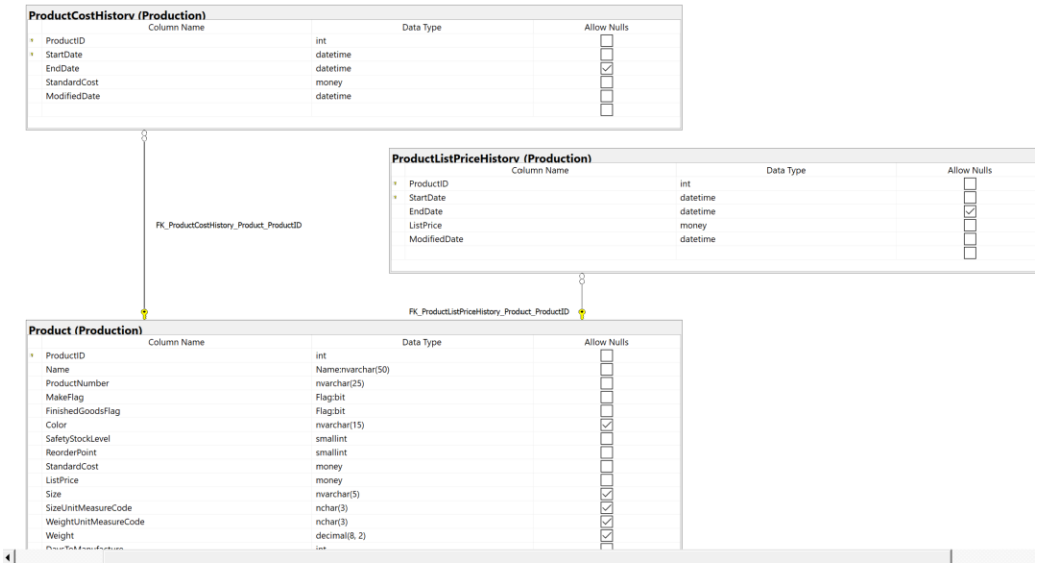
Query executed successfully.

Sample JSON Output (16 results returned):

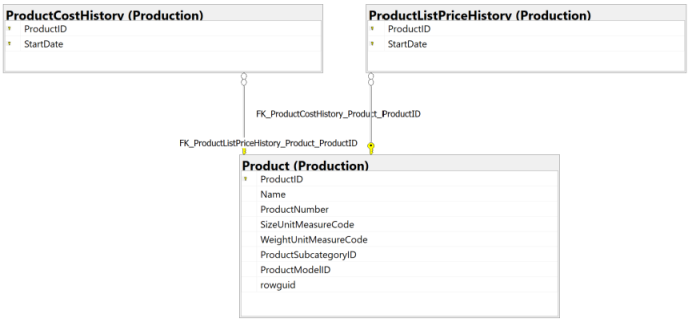


MEDIUM 8: This query lists the most profitable products

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Production.Product	ProductId Name
Production.ProductCostHistory	AvgCostToProd
Production.ListPriceHistory	AvgSellPrice

Order By:

Table Name	Column Name	Sort Order
Production.ProductListHistory	AvgProfit	DESC

Solution without JSON:

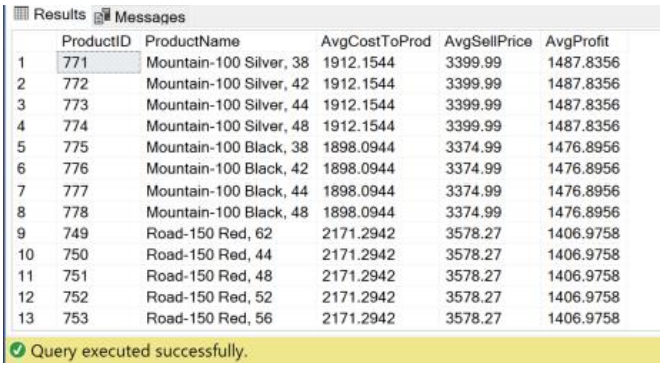
```
USE AdventureWorks2017;
SELECT prod.productid
	,prod.name AS ProductName
	,avg(cost.standardcost) AS AvgCostToProd
	,avg(list.listprice) AS AvgSellPrice
	,avg(list.listprice) - avg(cost.standardcost) AS AvgProfit
FROM production.product AS prod
INNER JOIN production.productcosthistory AS cost ON prod.ProductId = cost.ProductId
INNER JOIN production.productlistpricehistory AS list ON prod.productid = list.ProductId
GROUP BY prod.productid
	,prod.name
ORDER BY AvgProfit DESC
```

Solution with JSON:

```
USE AdventureWorks2017;
SELECT prod.productid
	,prod.name AS ProductName
	,avg(cost.standardcost) AS AvgCostToProd
	,avg(list.listprice) AS AvgSellPrice
	,avg(list.listprice) - avg(cost.standardcost) AS AvgProfit
FROM production.product AS prod
INNER JOIN production.productcosthistory AS cost ON prod.ProductId = cost.ProductId
INNER JOIN production.productlistpricehistory AS list ON prod.productid = list.ProductId
GROUP BY prod.productid
```

```
        ,prod.name
ORDER BY AvgProfit DESC
FOR JSON PATH, ROOT('Medium 8 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (293 results returned):

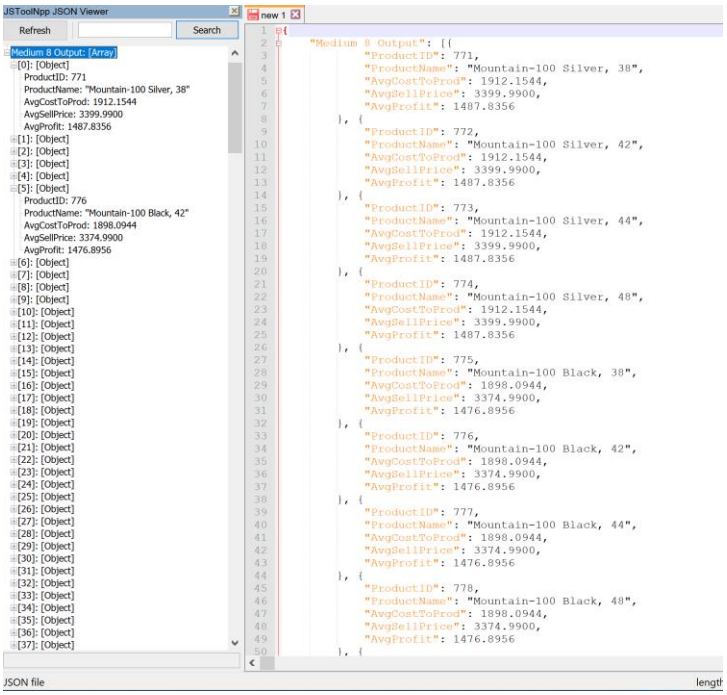


Results Messages

	ProductID	ProductName	AvgCostToProd	AvgSellPrice	AvgProfit
1	771	Mountain-100 Silver, 38	1912.1544	3399.99	1487.8356
2	772	Mountain-100 Silver, 42	1912.1544	3399.99	1487.8356
3	773	Mountain-100 Silver, 44	1912.1544	3399.99	1487.8356
4	774	Mountain-100 Silver, 48	1912.1544	3399.99	1487.8356
5	775	Mountain-100 Black, 38	1898.0944	3374.99	1476.8956
6	776	Mountain-100 Black, 42	1898.0944	3374.99	1476.8956
7	777	Mountain-100 Black, 44	1898.0944	3374.99	1476.8956
8	778	Mountain-100 Black, 48	1898.0944	3374.99	1476.8956
9	749	Road-150 Red, 62	2171.2942	3578.27	1406.9758
10	750	Road-150 Red, 44	2171.2942	3578.27	1406.9758
11	751	Road-150 Red, 48	2171.2942	3578.27	1406.9758
12	752	Road-150 Red, 52	2171.2942	3578.27	1406.9758
13	753	Road-150 Red, 56	2171.2942	3578.27	1406.9758

Query executed successfully.

Sample JSON Output (293 results returned):



JSTool Npp JSON Viewer

Refresh Search

Medium 8 Output: [Array]
- [0]: [Object]
  - ProductID: 771
  - ProductName: "Mountain-100 Silver, 38"
  - AvgCostToProd: 1912.1544
  - AvgSellPrice: 3399.9900
  - AvgProfit: 1487.8356
- [1]: [Object]
- [2]: [Object]
- [3]: [Object]
- [4]: [Object]
- [5]: [Object]
  - ProductID: 776
  - ProductName: "Mountain-100 Black, 42"
  - AvgCostToProd: 1898.0944
  - AvgSellPrice: 3374.9900
  - AvgProfit: 1476.8956
- [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]
- [30]: [Object]
- [31]: [Object]
- [32]: [Object]
- [33]: [Object]
- [34]: [Object]
- [35]: [Object]
- [36]: [Object]
- [37]: [Object]

new 1


```
{
    "Medium 8 Output": [{
            "ProductID": 771,
            "ProductName": "Mountain-100 Silver, 38",
            "AvgCostToProd": 1912.1544,
            "AvgSellPrice": 3399.9900,
            "AvgProfit": 1487.8356
        }, {
            "ProductID": 772,
            "ProductName": "Mountain-100 Silver, 42",
            "AvgCostToProd": 1912.1544,
            "AvgSellPrice": 3399.9900,
            "AvgProfit": 1487.8356
        }, {
            "ProductID": 773,
            "ProductName": "Mountain-100 Silver, 44",
            "AvgCostToProd": 1912.1544,
            "AvgSellPrice": 3399.9900,
            "AvgProfit": 1487.8356
        }, {
            "ProductID": 774,
            "ProductName": "Mountain-100 Silver, 48",
            "AvgCostToProd": 1912.1544,
            "AvgSellPrice": 3399.9900,
            "AvgProfit": 1487.8356
        }, {
            "ProductID": 775,
            "ProductName": "Mountain-100 Black, 38",
            "AvgCostToProd": 1898.0944,
            "AvgSellPrice": 3374.9900,
            "AvgProfit": 1476.8956
        }, {
            "ProductID": 776,
            "ProductName": "Mountain-100 Black, 42",
            "AvgCostToProd": 1898.0944,
            "AvgSellPrice": 3374.9900,
            "AvgProfit": 1476.8956
        }, {
            "ProductID": 777,
            "ProductName": "Mountain-100 Black, 44",
            "AvgCostToProd": 1898.0944,
            "AvgSellPrice": 3374.9900,
            "AvgProfit": 1476.8956
        }, {
            "ProductID": 778,
            "ProductName": "Mountain-100 Black, 48",
            "AvgCostToProd": 1898.0944,
            "AvgSellPrice": 3374.9900,
            "AvgProfit": 1476.8956
        }, {
```


JSON file

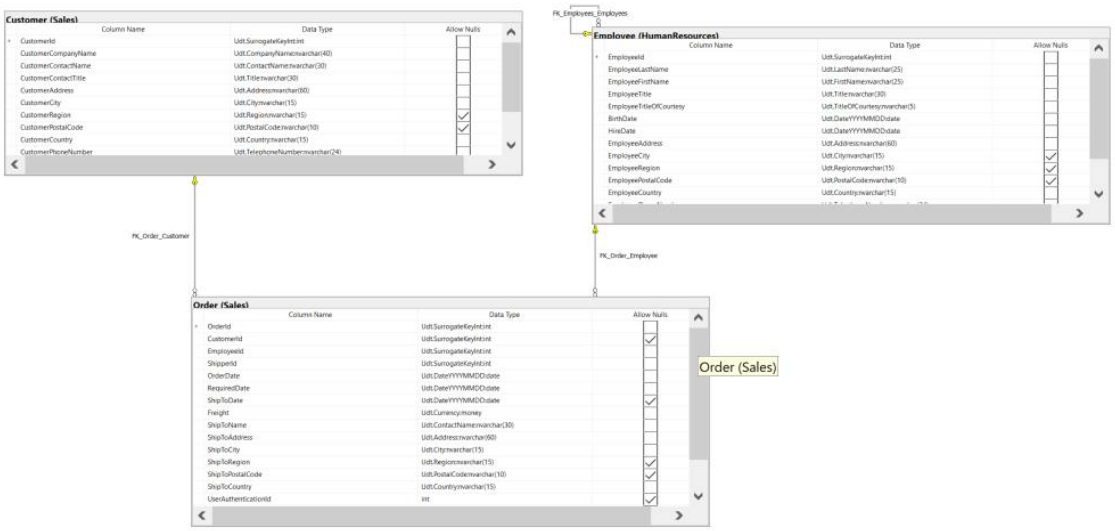
length

COMPLEX 1:

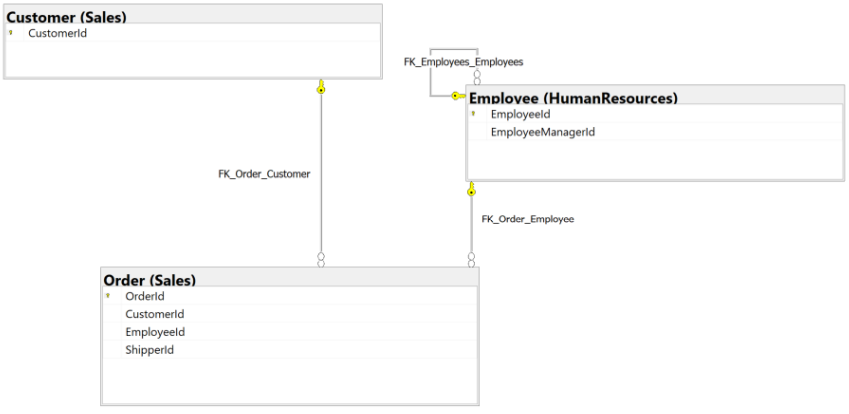
QUERY: This query finds the average amount of days it takes to process a customer's order, from placing the order to shipping it out. Additional information about the customer is returned to determine whether there may be any correlations in disparities. This query also finds the employee who services that customer the most.

FUNCTIONS: [calcDateAvg] calculates the average amount of days from the order being placed to it being shipped out across all orders for the specified customer. [favEmployee] returns the name of the employee that services the specified customer

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Sales.Customer	CustomerId CustomerContactName CustomerCity
Sales.Order	AvgResponseInDays
HumanResources.Employee	FreqHandledBy

Order By:

Table Name	Column Name	Sort Order
Sales.Order	AvgResponseInDays	ASC

Solution without JSON:

```
USE Northwinds2020TSQLV6;
GO
CREATE FUNCTION [sales].[calcDateAvg] (@id INT)
RETURNS INT
AS
BEGIN
	DECLARE @avg INT;

	SET @avg = (
			SELECT avg(DATEDIFF(day, orderdate, shiptodate))
			FROM sales.[order]
			WHERE customerid = @id
			GROUP BY customerid
			)

	RETURN @avg
END
GO
```

```
DROP FUNCTION

IF EXISTS [sales].[calcDateAvg]
	---------------------------------
	GO
	CREATE FUNCTION [sales].[favEmployee] (@custId INT)
	RETURNS NVARCHAR(100)
	AS
	BEGIN
		DECLARE @favEmp NVARCHAR(100);
		DECLARE @empl INT;

		SET @empl = (
				SELECT TOP 1 EmployeeId
				FROM sales.[order]
				WHERE CustomerId = @custId
				GROUP BY CustomerId
					,EmployeeId
				ORDER BY count(EmployeeId) DESC
				)
		SET @favEmp = (
				SELECT EmployeeFirstName
				FROM HumanResources.Employee
				WHERE EmployeeId = @empl
				) + ' ' + (
				SELECT EmployeeLastName
				FROM HumanResources.Employee
				WHERE EmployeeId = @empl
				)
```

```
        RETURN @favEmp
    END
GO

DROP FUNCTION

IF EXISTS [sales].[favEmployee]

    USE Northwinds2020TSQLV6;
    SELECT c.customerid
        ,c.CustomerContactName
        ,c.CustomerCity
        ,[sales].[calcDateAvg](c.customerid) AS AvgResponseInDays
        ,count(so.customerid) AS NumOrders
        ,[Sales].[favemployee](c.CustomerId) AS FreqHandBy
    FROM sales.[order] AS so
    INNER JOIN sales.customer AS c ON so.customerid = c.customerid
    INNER JOIN HumanResources.Employee AS e ON so.EmployeeId = e.EmployeeId
    GROUP BY c.customerid
        ,c.CustomerContactName
        ,c.CustomerCity
    ORDER BY AvgResponseInDays ASC
```

Solution with JSON:

```
USE Northwinds2020TSQLV6;
GO
CREATE FUNCTION [sales].[calcDateAvg] (@id INT)
RETURNS INT
```

```
AS
BEGIN
	DECLARE @avg INT;

	SET @avg = (
			SELECT avg(DATEDIFF(day, orderdate, shiptodate))
			FROM sales.[order]
			WHERE customerid = @id
			GROUP BY customerid
			)

	RETURN @avg
END
GO

DROP FUNCTION

IF EXISTS [sales].[calcDateAvg]
	---------------------------------
	GO
	CREATE FUNCTION [sales].[favEmployee] (@custId INT)
	RETURNS NVARCHAR(100)
	AS
	BEGIN
		DECLARE @favEmp NVARCHAR(100);
		DECLARE @empl INT;

		SET @empl = (
				SELECT TOP 1 EmployeeId
				FROM sales.[order]
```

```
                WHERE CustomerId = @custId
                GROUP BY CustomerId
                        ,EmployeeId
                ORDER BY count(EmployeeId) DESC
                )
        SET @favEmp = (
                SELECT EmployeeFirstName
                FROM HumanResources.Employee
                WHERE EmployeeId = @empl
                ) + ' ' + (
                SELECT EmployeeLastName
                FROM HumanResources.Employee
                WHERE EmployeeId = @empl
                )

        RETURN @favEmp
    END
GO

DROP FUNCTION

IF EXISTS [sales].[favEmployee]

    USE Northwinds2020TSQLV6;
    SELECT c.customerid
        ,c.CustomerContactName
        ,c.CustomerCity
        ,[sales].[calcDateAvg](c.customerid) AS AvgResponseInDays
        ,count(so.customerid) AS NumOrders
        ,[Sales].[favemployee](c.CustomerId) AS FreqHandBy
```

```
        FROM sales.[order] AS so
        INNER JOIN sales.customer AS c ON so.customerid = c.customerid
        INNER JOIN HumanResources.Employee AS e ON so.EmployeeId = e.EmployeeId
        GROUP BY c.customerid
                ,c.CustomerContactName
                ,c.CustomerCity
        ORDER BY AvgResponseInDays ASC
FOR JSON PATH, ROOT('Complex 1 Output'), INCLUDE_NULL_VALUES;
```

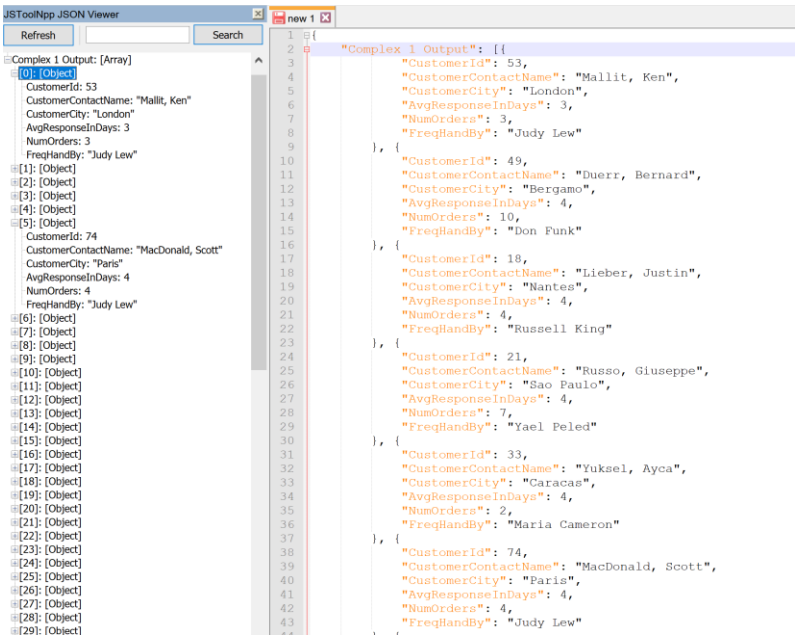
Sample Output (89 results returned):

Results | Messages

	CustomerId	CustomerContactName	CustomerCity	AvgResponseInDays	NumOrders	FreqHandBy
1	53	Mallit, Ken	London	3	3	Judy Lew
2	49	Duerr, Bernard	Bergamo	4	10	Don Funk
3	18	Lieber, Justin	Nantes	4	4	Russell King
4	21	Russo, Giuseppe	Sao Paulo	4	7	Yael Peled
5	33	Yuksel, Ayca	Caracas	4	2	Maria Cameron
6	74	MacDonald, Scott	Paris	4	4	Judy Lew
7	90	Larsson, Katarina	Helsinki	4	7	Don Funk
8	81	Edwards, Josh	Sao Paulo	5	6	Sara Davis
9	12	Ray, Mike	Buenos Aires	5	6	Maria Cameron
10	51	Taylor, Maurice	Montréal	5	13	Judy Lew
11	39	Song, Lolan	Brandenburg	6	14	Sara Davis
12	61	Meisels, Josh	Rio de Janeiro	6	9	Don Funk
13	69	Troup, Carol	Madrid	6	5	Yael Peled

Query executed successfully.

Sample JSON Output (89 results returned):



JSToolNpp JSON Viewer — Refresh — Search

```
Complex 1 Output: [Array]
  [0]: [Object]
    CustomerId: 53
    CustomerContactName: "Mallit, Ken"
    CustomerCity: "London"
    AvgResponseInDays: 3
    NumOrders: 3
    FreqHandBy: "Judy Lew"
  [1]: [Object]
  [2]: [Object]
  [3]: [Object]
  [4]: [Object]
  [5]: [Object]
    CustomerId: 74
    CustomerContactName: "MacDonald, Scott"
    CustomerCity: "Paris"
    AvgResponseInDays: 4
    NumOrders: 4
    FreqHandBy: "Judy Lew"
  [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]
```

new 1

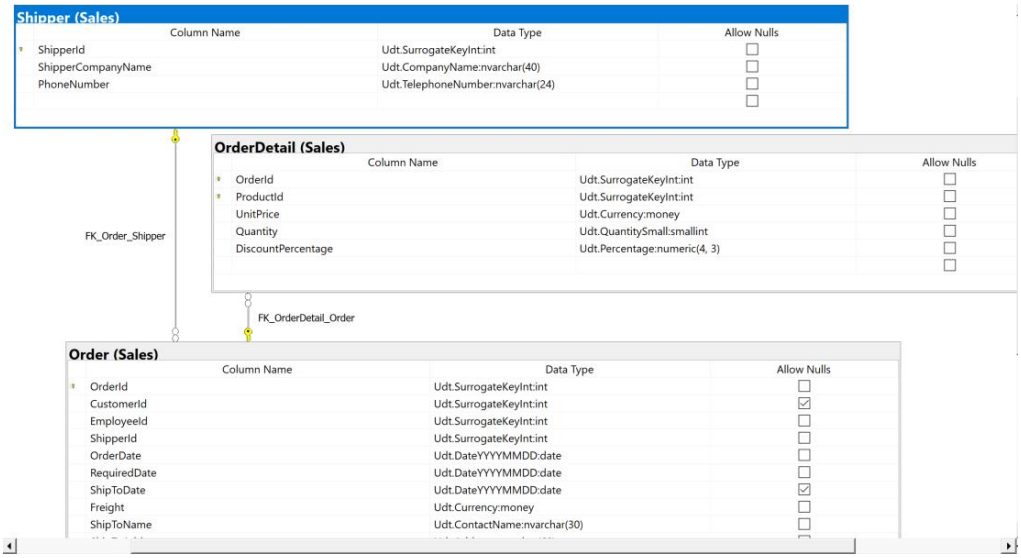

```
{
    "Complex 1 Output": [{
            "CustomerId": 53,
            "CustomerContactName": "Mallit, Ken",
            "CustomerCity": "London",
            "AvgResponseInDays": 3,
            "NumOrders": 3,
            "FreqHandBy": "Judy Lew"
        }, {
            "CustomerId": 49,
            "CustomerContactName": "Duerr, Bernard",
            "CustomerCity": "Bergamo",
            "AvgResponseInDays": 4,
            "NumOrders": 10,
            "FreqHandBy": "Don Funk"
        }, {
            "CustomerId": 18,
            "CustomerContactName": "Lieber, Justin",
            "CustomerCity": "Nantes",
            "AvgResponseInDays": 4,
            "NumOrders": 4,
            "FreqHandBy": "Russell King"
        }, {
            "CustomerId": 21,
            "CustomerContactName": "Russo, Giuseppe",
            "CustomerCity": "Sao Paulo",
            "AvgResponseInDays": 4,
            "NumOrders": 7,
            "FreqHandBy": "Yael Peled"
        }, {
            "CustomerId": 33,
            "CustomerContactName": "Yuksel, Ayca",
            "CustomerCity": "Caracas",
            "AvgResponseInDays": 4,
            "NumOrders": 2,
            "FreqHandBy": "Maria Cameron"
        }, {
            "CustomerId": 74,
            "CustomerContactName": "MacDonald, Scott",
            "CustomerCity": "Paris",
            "AvgResponseInDays": 4,
            "NumOrders": 4,
            "FreqHandBy": "Judy Lew"
        }, {
```

COMPLEX 2:

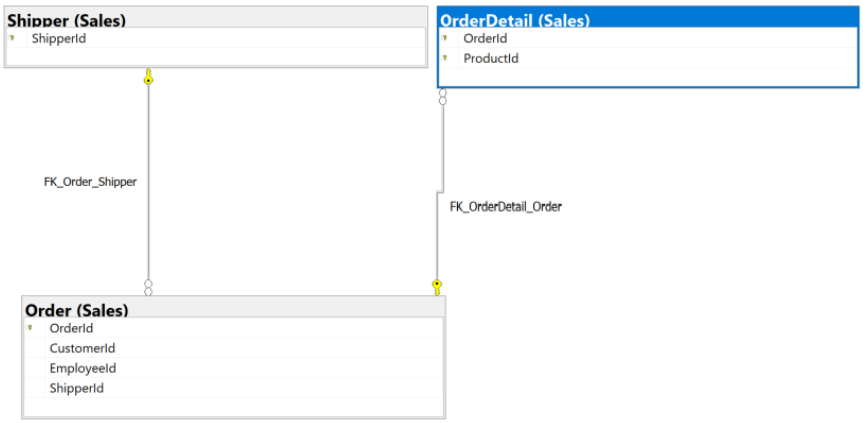
QUERY: This query ranks all shippers based on the number of deliveries their company have completed. This query also list the customer that each shipper delivers most frequently to.

FUNCTIONS: [freqCust] finds the customer that the specified shipper has delivered to most.

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Sales.Shipper	ShipperId

	ShipperCompanyName NumDeliveries FreqCust

Order By:

Table Name	Column Name	Sort Order
Sales.Shipper	NumDeliveries	DESC

Solution without JSON:

```
USE Northwinds2020TSQLV6;
GO
CREATE FUNCTION sales.[freqCust] (@supid INT)
RETURNS NVARCHAR(100)
AS
BEGIN
	DECLARE @custid INT;
	DECLARE @custName NVARCHAR(100);

	SET @custid = (
			SELECT TOP 1 CustomerId
			FROM Sales.[Order]
			WHERE ShipperId = @supid
			GROUP BY ShipperId
				,CustomerId
			ORDER BY COUNT(CustomerId) DESC
			);
	SET @custName = (
			SELECT CustomerContactName
			FROM Sales.Customer
			WHERE CustomerId = @custid
```

```
                    )

        RETURN @custname
END
GO

SELECT s.ShipperId
        ,s.ShipperCompanyName
        ,count(o.ShipperId) AS NumDeliveries
        ,sales.[freqCust](s.ShipperId) AS FreqCust
FROM Sales.[Order] AS o
INNER JOIN Sales.OrderDetail AS od ON o.OrderId = od.OrderId
INNER JOIN sales.Shipper AS s ON s.shipperid = o.ShipperId
GROUP BY s.ShipperId
        ,s.ShipperCompanyName
ORDER BY NumDeliveries DESC
```

Solution with JSON:

```
USE Northwinds2020TSQLV6;
GO
CREATE FUNCTION sales.[freqCust] (@supid INT)
RETURNS NVARCHAR(100)
AS
BEGIN
        DECLARE @custid INT;
        DECLARE @custName NVARCHAR(100);
        SET @custid = (
                        SELECT TOP 1 CustomerId
                        FROM Sales.[Order]
```

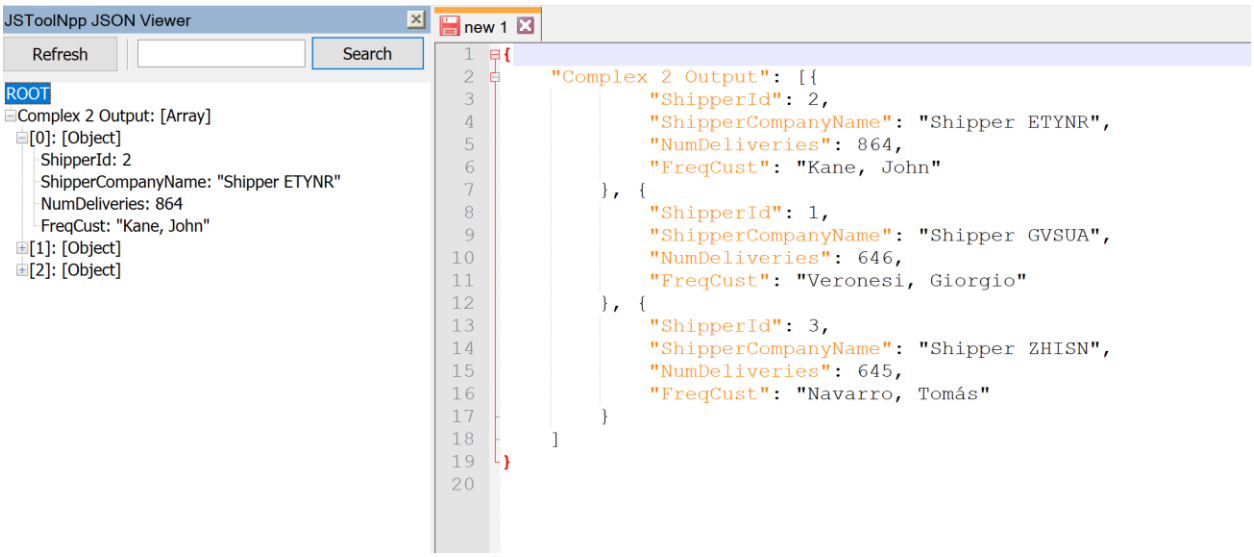
```
                    WHERE ShipperId = @supid
                    GROUP BY ShipperId
                            ,CustomerId
                    ORDER BY COUNT(CustomerId) DESC
                    );
        SET @custName = (
                    SELECT CustomerContactName
                    FROM Sales.Customer
                    WHERE CustomerId = @custid
                    )

        RETURN @custname
END
GO

USE Northwinds2020TSQLV6;
SELECT s.ShipperId
        ,s.ShipperCompanyName
        ,count(o.ShipperId) AS NumDeliveries
        ,sales.[freqCust](s.ShipperId) AS FreqCust
FROM Sales.[Order] AS o
INNER JOIN Sales.OrderDetail AS od ON o.OrderId = od.OrderId
INNER JOIN sales.Shipper AS s ON s.shipperid = o.ShipperId
GROUP BY s.ShipperId
        ,s.ShipperCompanyName
ORDER BY NumDeliveries DESC
FOR JSON PATH, ROOT('Complex 2 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (3 results returned):

Results Messages

	ShipperId	ShipperCompanyName	NumDeliveries	FreqCust
1	2	Shipper ETYNR	864	Kane, John
2	1	Shipper GVSUA	646	Veronesi, Giorgio
3	3	Shipper ZHISN	645	Navarro, Tomás

Sample JSON Output (3 results returned):

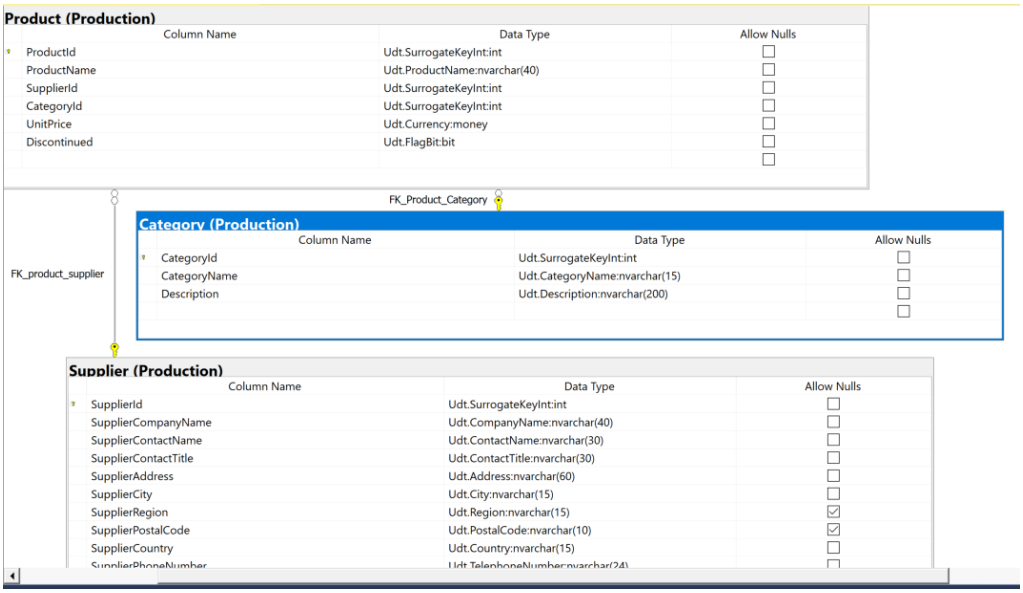


COMPLEX 3:

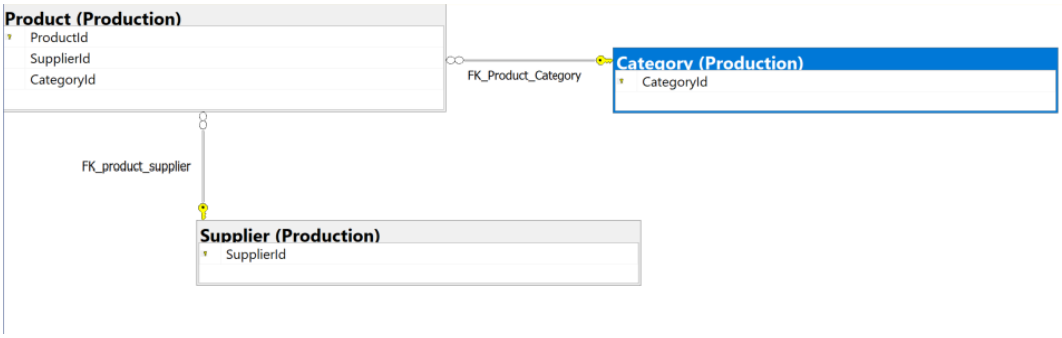
QUERY: This query ranks store's catalogs based on the average cost of all their goods. This query also finds each store's most expensive item along with some information about that product.

FUNCTIONS: [avgIntValue] finds the average price of all the specified supplier's goods. [mostExpProd] finds the most expensive item of the specified supplier.

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Production.Supplier	SupplierId SupplierCompanyName
Production.Product	MostValProdPrice ProdName
Production.Category	Category Description AvgCatelogValue

Order By:

Table Name	Column Name	Sort Order
Production.Supplier	AvgCatelogValue	DESC

Solution without JSON:

```
USE Northwinds2020TSQLV6;
GO
CREATE FUNCTION Production.[avgInvtValue] (@supid INT)
RETURNS FLOAT
AS
BEGIN
	DECLARE @avg FLOAT;


	SET @avg = (
			SELECT avg(UnitPrice)
			FROM Production.Product
			WHERE SupplierId = @supid
			)

	RETURN @avg
END
GO

--cleanup
DROP FUNCTION

IF EXISTS Production.avgInvtValue;

	GO
	CREATE FUNCTION Production.[mostExpProd] (@supid INT)
	RETURNS NVARCHAR(100)
```

```
	AS
	BEGIN
		DECLARE @prodNam NVARCHAR(100);

		SET @prodNam = (
				SELECT TOP 1 ProductName
				FROM Production.Product
				WHERE SupplierId = @supid
				ORDER BY UnitPrice DESC
				)

		RETURN @prodNam
	END
GO

--cleanup
DROP FUNCTION

IF EXISTS Production.mostExpProd;
	-----------------------------------------------------------------------------------------------------------
	GO
	CREATE FUNCTION Production.[mostExpProdCat] (@supid INT)
	RETURNS NVARCHAR(100)
	AS
	BEGIN
		DECLARE @catNam NVARCHAR(100);
		DECLARE @catid INT;

		SET @catid = (
				SELECT TOP 1 CategoryId
```

```
                    FROM Production.Product
                    WHERE SupplierId = @supid
                    ORDER BY UnitPrice DESC
                    )
        SET @catNam = (
                    SELECT CategoryName
                    FROM Production.Category
                    WHERE CategoryId = @catid
                    )

        RETURN @catNam
    END
GO

--cleanup
DROP FUNCTION

IF EXISTS Production.mostExpProdCat;
    USE Northwinds2020TSQLV6;
    SELECT sup.SupplierId
        ,sup.SupplierCompanyName
        ,Production.avgInvtValue(sup.SupplierId) AS AvgCatelogValue
        ,MAX(prod.UnitPrice) AS MostValProdPrice
        ,Production.mostExpProd(sup.SupplierId) AS ProdName
        ,Production.mostExpProdCat(sup.SupplierId) AS Category
        ,cat.[Description]
    FROM Production.Supplier AS sup
    INNER JOIN Production.Product AS prod ON sup.SupplierId = prod.SupplierId
    INNER JOIN Production.Category AS cat ON cat.CategoryName =
Production.mostExpProdCat(sup.SupplierId)
```

```
	GROUP BY sup.SupplierId
		,sup.SupplierCompanyName
		,cat.[Description]
	ORDER BY AvgCatelogValue DESC
```

Solution with JSON:

```
USE Northwinds2020TSQLV6;
GO
CREATE FUNCTION Production.[avgInvtValue] (@supid INT)
RETURNS FLOAT
AS
BEGIN
	DECLARE @avg FLOAT;


	SET @avg = (
			SELECT avg(UnitPrice)
			FROM Production.Product
			WHERE SupplierId = @supid
			)

	RETURN @avg
END
GO

--cleanup
DROP FUNCTION
```

```
IF EXISTS Production.avgInvtValue;

	GO
	CREATE FUNCTION Production.[mostExpProd] (@supid INT)
	RETURNS NVARCHAR(100)
	AS
	BEGIN
		DECLARE @prodNam NVARCHAR(100);

		SET @prodNam = (
				SELECT TOP 1 ProductName
				FROM Production.Product
				WHERE SupplierId = @supid
				ORDER BY UnitPrice DESC
				)

		RETURN @prodNam
	END
GO

--cleanup
DROP FUNCTION

IF EXISTS Production.mostExpProd;
	-----------------------------------------------------------------------------------------------------------------
	GO
	CREATE FUNCTION Production.[mostExpProdCat] (@supid INT)
	RETURNS NVARCHAR(100)
	AS
	BEGIN
```

```
		DECLARE @catNam NVARCHAR(100);
		DECLARE @catid INT;

		SET @catid = (
				SELECT TOP 1 CategoryId
				FROM Production.Product
				WHERE SupplierId = @supid
				ORDER BY UnitPrice DESC
				)
		SET @catNam = (
				SELECT CategoryName
				FROM Production.Category
				WHERE CategoryId = @catid
				)

		RETURN @catNam
	END
GO

--cleanup
DROP FUNCTION

IF EXISTS Production.mostExpProdCat;
	USE Northwinds2020TSQLV6;
	SELECT sup.SupplierId
		,sup.SupplierCompanyName
		,Production.avgInvtValue(sup.SupplierId) AS AvgCatelogValue
		,MAX(prod.UnitPrice) AS MostValProdPrice
		,Production.mostExpProd(sup.SupplierId) AS ProdName
		,Production.mostExpProdCat(sup.SupplierId) AS Category
```

```
        ,cat.[Description]
    FROM Production.Supplier AS sup
    INNER JOIN Production.Product AS prod ON sup.SupplierId = prod.SupplierId
    INNER JOIN Production.Category AS cat ON cat.CategoryName =
Production.mostExpProdCat(sup.SupplierId)
    GROUP BY sup.SupplierId
        ,sup.SupplierCompanyName
        ,cat.[Description]
    ORDER BY AvgCatelogValue DESC
```

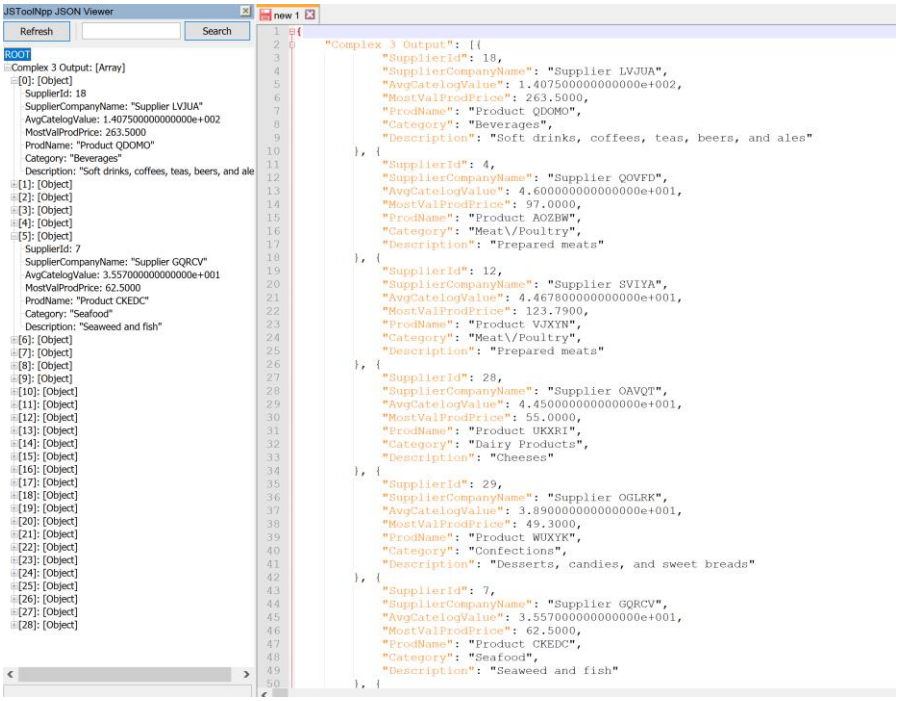
Sample Output (29 results returned):

Results | Messages

	SupplierId	SupplierCompanyName	AvgCatelogValue	MostValProdPrice	ProdName	Category	Description
1	18	Supplier LVJUA	140.75	263.50	Product QDOMO	Beverages	Soft drinks, coffees, teas, beers, and ales
2	4	Supplier QOVFD	46	97.00	Product AOZBW	Meat/Poultry	Prepared meats
3	12	Supplier SVIYA	44.678	123.79	Product VJXYN	Meat/Poultry	Prepared meats
4	28	Supplier OAVQT	44.5	55.00	Product UKXRI	Dairy Products	Cheeses
5	29	Supplier OGLRK	38.9	49.30	Product WUXYK	Confections	Desserts, candies, and sweet breads
6	7	Supplier GQRCV	35.57	62.50	Product CKEDC	Seafood	Seaweed and fish
7	3	Supplier STUAZ	31.6666	40.00	Product WVJFP	Condiments	Sweet and savory sauces, relishes, spreads, and ...
8	24	Supplier JNNES	30.9333	53.00	Product APITJ	Produce	Dried fruit and bean curd
9	11	Supplier ZPYVS	29.71	43.90	Product SMIOH	Confections	Desserts, candies, and sweet breads

Query executed successfully. localhost, 12001 (15.0 RTM) sa (56)

Sample JSON Output (29 results returned):

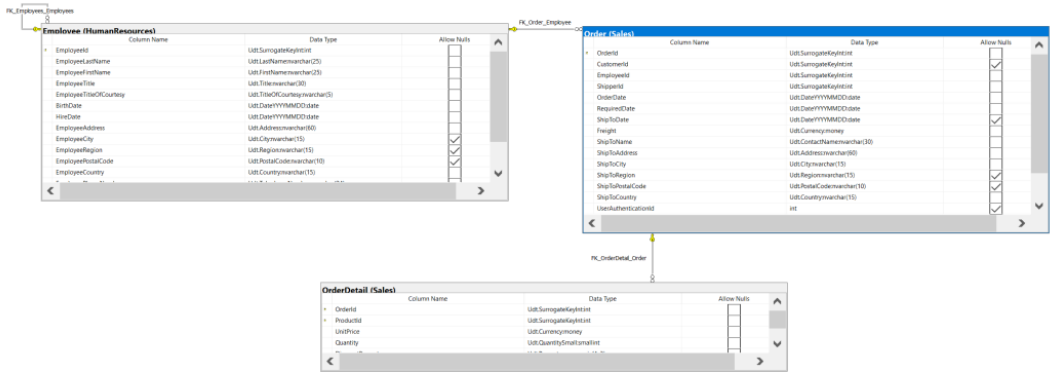


COMPLEX 4:

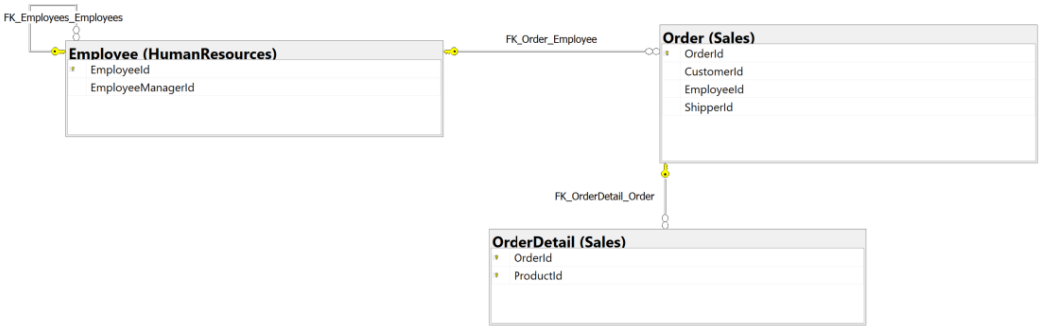
QUERY: This query finds the most productive quarter for each employee based on data from their transactions since the beginning of their employment. This query also compares their total sales to the total sales of the employee specifically from that quarter.

FUNCTIONS: [yearQtr] converts the date to quarter of the year. More specifically, determines what quarter the date is in. [mostProdQtr] returns the quarter in which the employee completed the most orders. [numOrdersProd] returns the number of orders from the employee's most productive quarter. [totEmpSales] finds the total number of transactions the specified employee has completed.

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
HumanResources.Employee	EmployeeId EmployeeFirstName EmployeeLastName EmployeeTitle MostProdQtr
Sales.Order	MPQ_Sales TotalSales

Order By:

Table Name	Column Name	Sort Order
Sales.Order	TotalSales	DESC

Solution without JSON:

```
USE Northwinds2020TSQLV6;
GO
CREATE FUNCTION sales.[yearQtr] (@date DATE)
RETURNS NVARCHAR(10)
AS
BEGIN
	DECLARE @val NVARCHAR(10);
	DECLARE @month INT;

	SET @month = MONTH(@date);

	SELECT @val = CASE
			WHEN @month BETWEEN 1
					AND 3
				THEN 'QTR 1'
			WHEN @month BETWEEN 4
					AND 6
				THEN 'QTR 2'
			WHEN @month BETWEEN 7
					AND 9
				THEN 'QTR 3'
			WHEN @month BETWEEN 10
					AND 12
				THEN 'QTR 4'
			ELSE 'Unknown QTR'
			END

	RETURN @val
```

```
END
GO

--cleanup
DROP FUNCTION

IF EXISTS Sales.yearQtr;
	-----------------------------------------------------------------------------------------------------------------------

	GO
	CREATE FUNCTION sales.[mostProdQtr] (@empl INT)
	RETURNS NVARCHAR(10)
	AS
	BEGIN
		DECLARE @val NVARCHAR(10);

		SET @val = (
				SELECT TOP 1 Sales.yearQtr(ShipToDate)
				FROM Sales.[Order]
				WHERE employeeid = @empl
				GROUP BY Sales.yearQtr(ShipToDate)
				ORDER BY count(*) DESC
				)

		RETURN @val
	END
GO

--cleanup
DROP FUNCTION
```

```
IF EXISTS Sales.mostProdQtr;
	-----------------------------------------------------------------------------------------------------------------

	GO
	CREATE FUNCTION sales.[numOrdersProd] (@empl INT)
	RETURNS INT
	AS
	BEGIN
		DECLARE @val INT;

		SET @val = (
				SELECT TOP 1 count(*)
				FROM Sales.[Order]
				WHERE EmployeeId = @empl
				GROUP BY Sales.yearqtr(ShipToDate)
				ORDER BY COUNT(*) DESC
				)

		RETURN @val
	END
GO

--cleanup
DROP FUNCTION

IF EXISTS Sales.numOrdersProd;

	GO
	CREATE FUNCTION sales.[totempsales] (@empl INT)
```

```
	RETURNS INT
	AS
	BEGIN
		DECLARE @val INT;

		SET @val = (
				SELECT count(*)
				FROM Sales.[Order]
				WHERE EmployeeId = @empl
				)

		RETURN @val
	END
GO

--cleanup
DROP FUNCTION

IF EXISTS Sales.totempsales;
	-----------------------------------------------------------------------------------------------------------
	USE Northwinds2020TSQLV6;
	SELECT e.EmployeeId
		,e.EmployeeFirstName
		,e.EmployeeLastName
		,e.EmployeeTitle
		,Sales.mostprodqtr(e.EmployeeId) AS MostProdQtr
		,Sales.numordersprod(e.EmployeeId) AS MPQ_Sales
		,Sales.totempsales(e.EmployeeId) AS TotalSales
	FROM Sales.[Order] AS o
	INNER JOIN HumanResources.Employee AS e ON o.EmployeeId = e.EmployeeId
```

```
    GROUP BY e.EmployeeId
        ,e.EmployeeFirstName
        ,e.EmployeeLastName
        ,e.EmployeeTitle
    ORDER BY TotalSales DESC
```

Solution with JSON:

```
USE Northwinds2020TSQLV6;
GO
CREATE FUNCTION sales.[yearQtr] (@date DATE)
RETURNS NVARCHAR(10)
AS
BEGIN
    DECLARE @val NVARCHAR(10);
    DECLARE @month INT;

    SET @month = MONTH(@date);

    SELECT @val = CASE
            WHEN @month BETWEEN 1
                    AND 3
                THEN 'QTR 1'
            WHEN @month BETWEEN 4
                    AND 6
                THEN 'QTR 2'
            WHEN @month BETWEEN 7
                    AND 9
                THEN 'QTR 3'
            WHEN @month BETWEEN 10
```

```
                                AND 12
                            THEN 'QTR 4'
                        ELSE 'Unknown QTR'
                        END

    RETURN @val
END
GO

--cleanup
DROP FUNCTION

IF EXISTS Sales.yearQtr;
    ----------------------------------------------------------------------------------------------------------------

    GO
    CREATE FUNCTION sales.[mostProdQtr] (@empl INT)
    RETURNS NVARCHAR(10)
    AS
    BEGIN
        DECLARE @val NVARCHAR(10);

        SET @val = (
                    SELECT TOP 1 Sales.yearQtr(ShipToDate)
                    FROM Sales.[Order]
                    WHERE employeeid = @empl
                    GROUP BY Sales.yearQtr(ShipToDate)
                    ORDER BY count(*) DESC
                    )
```

```
        RETURN @val
    END
GO

--cleanup
DROP FUNCTION

IF EXISTS Sales.mostProdQtr;
    ----------------------------------------------------------------------------------------------------------------

    GO
    CREATE FUNCTION sales.[numOrdersProd] (@empl INT)
    RETURNS INT
    AS
    BEGIN
        DECLARE @val INT;

        SET @val = (
                SELECT TOP 1 count(*)
                FROM Sales.[Order]
                WHERE EmployeeId = @empl
                GROUP BY Sales.yearqtr(ShipToDate)
                ORDER BY COUNT(*) DESC
                )

        RETURN @val
    END
GO

--cleanup
```

```
DROP FUNCTION

IF EXISTS Sales.numOrdersProd;

	GO
	CREATE FUNCTION sales.[totempsales] (@empl INT)
	RETURNS INT
	AS
	BEGIN
		DECLARE @val INT;

		SET @val = (
				SELECT count(*)
				FROM Sales.[Order]
				WHERE EmployeeId = @empl
				)

		RETURN @val
	END
GO

--cleanup
DROP FUNCTION

IF EXISTS Sales.totempsales;
	-----------------------------------------------------------------------------------------------------------------
	USE Northwinds2020TSQLV6;
	SELECT e.EmployeeId
		,e.EmployeeFirstName
		,e.EmployeeLastName
```

```
        ,e.EmployeeTitle
        ,Sales.mostprodqtr(e.EmployeeId) AS MostProdQtr
        ,Sales.numordersprod(e.EmployeeId) AS MPQ_Sales
        ,Sales.totempsales(e.EmployeeId) AS TotalSales
FROM Sales.[Order] AS o
INNER JOIN HumanResources.Employee AS e ON o.EmployeeId = e.EmployeeId
GROUP BY e.EmployeeId
        ,e.EmployeeFirstName
        ,e.EmployeeLastName
        ,e.EmployeeTitle
ORDER BY TotalSales DESC
```

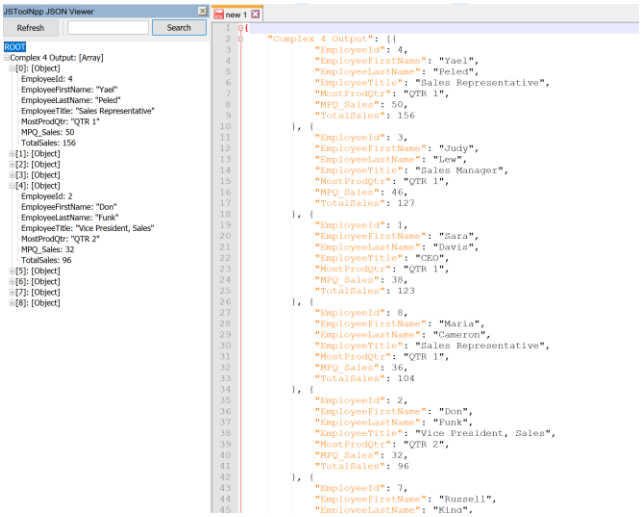
Sample Output (9 results returned):

Results | Messages

	EmployeeId	EmployeeFirstName	EmployeeLastName	EmployeeTitle	MostProdQtr	MPQ_Sales	TotalSales
1	4	Yael	Peled	Sales Representative	QTR 1	50	156
2	3	Judy	Lew	Sales Manager	QTR 1	46	127
3	1	Sara	Davis	CEO	QTR 1	38	123
4	8	Maria	Cameron	Sales Representative	QTR 1	36	104
5	2	Don	Funk	Vice President, Sales	QTR 2	32	96
6	7	Russell	King	Sales Representative	QTR 2	22	72
7	6	Paul	Suurs	Sales Representative	QTR 1	23	67
8	9	Patricia	Doyle	Sales Representative	QTR 1	15	43
9	5	Sven	Mortensen	Sales Manager	QTR 1	16	42

Query executed successfully.

Sample JSON Output (9 results returned):



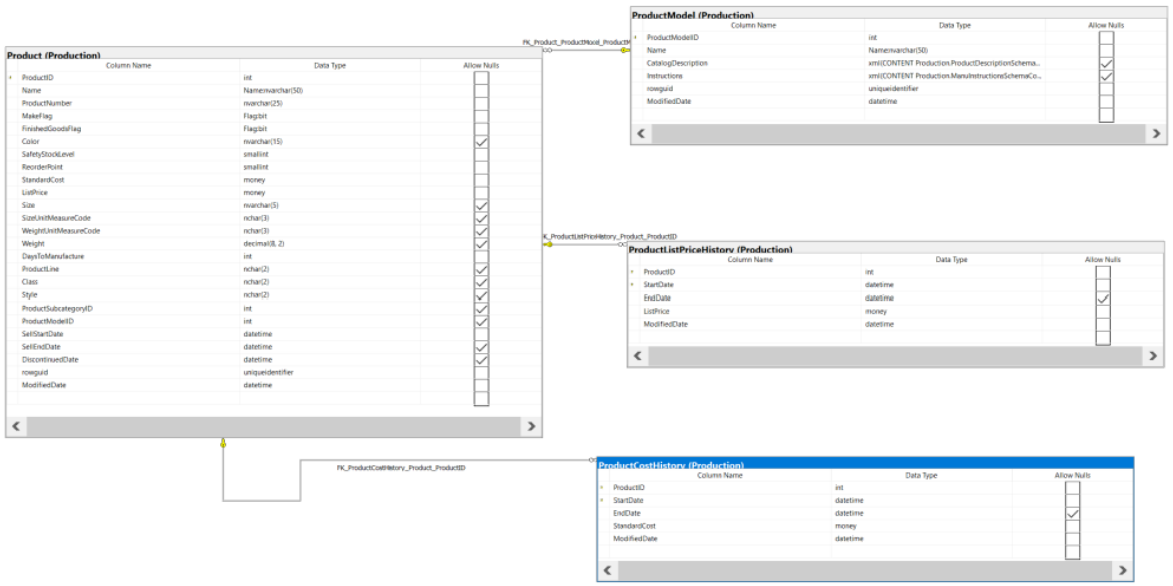

```
{
    "Complex 4 Output": [{
            "EmployeeId": 4,
            "EmployeeFirstName": "Yael",
            "EmployeeLastName": "Peled",
            "EmployeeTitle": "Sales Representative",
            "MostProdQtr": "QTR 1",
            "MPQ_Sales": 50,
            "TotalSales": 156
        }, {
            "EmployeeId": 3,
            "EmployeeFirstName": "Judy",
            "EmployeeLastName": "Lew",
            "EmployeeTitle": "Sales Manager",
            "MostProdQtr": "QTR 1",
            "MPQ_Sales": 46,
            "TotalSales": 127
        }, {
            "EmployeeId": 1,
            "EmployeeFirstName": "Sara",
            "EmployeeLastName": "Davis",
            "EmployeeTitle": "CEO",
            "MostProdQtr": "QTR 1",
            "MPQ_Sales": 38,
            "TotalSales": 123
        }, {
            "EmployeeId": 8,
            "EmployeeFirstName": "Maria",
            "EmployeeLastName": "Cameron",
            "EmployeeTitle": "Sales Representative",
            "MostProdQtr": "QTR 1",
            "MPQ_Sales": 36,
            "TotalSales": 104
        }, {
            "EmployeeId": 2,
            "EmployeeFirstName": "Don",
            "EmployeeLastName": "Funk",
            "EmployeeTitle": "Vice President, Sales",
            "MostProdQtr": "QTR 2",
            "MPQ_Sales": 32,
            "TotalSales": 96
        }, {
            "EmployeeId": 7,
            "EmployeeFirstName": "Russell",
            "EmployeeLastName": "King",
```

COMPLEX 5:

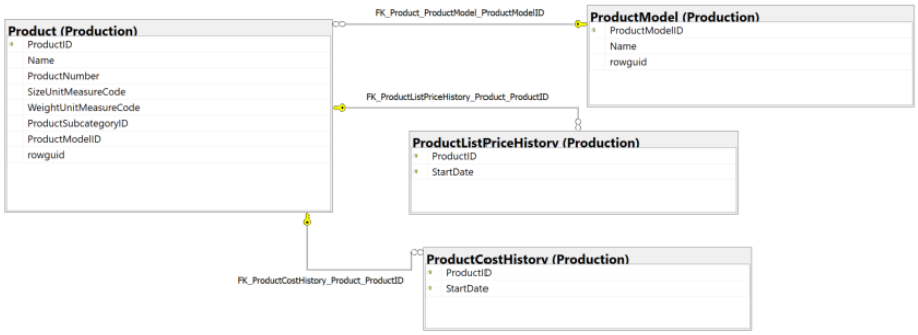
QUERY: This query ranks the best performing products based their total profits and provides some information about each product

FUNCTIONS: [profMarg] returns the profit margins of the specified product for the specified year. [prodProfits] returns the total net profit amount from all transactions of the specified product. [totSales] returns the gross sales amount across all transactions of the specified product.

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Production.Product	ProductId Name
Production.ProductCostHistory	AvgProductionCost
Production.ProductListPriceHistory	AvgListPrice
Sales.Order	TotalSalesAmt TotalProfits

Order By:

Table Name	Column Name	Sort Order
Sales.Order	TotalProfits	DESC

Solution without JSON:

```
USE AdventureWorks2017;
GO
CREATE FUNCTION production.[profMarg] (
	@prodId INT
	,@year INT
	)
RETURNS FLOAT
AS
BEGIN
	DECLARE @cost FLOAT;
	DECLARE @price FLOAT;
	DECLARE @res FLOAT

	SET @cost = (
			SELECT StandardCost
			FROM Production.ProductCostHistory
			WHERE ProductID = @prodId
```

```
                        AND year(StartDate) = @year
                )
        SET @price = (
                SELECT ListPrice
                FROM Production.ProductListPriceHistory
                WHERE ProductID = @prodId
                        AND year(StartDate) = @year
                )
        SET @res = @price - @cost

        RETURN @res
END
GO

GO

CREATE FUNCTION production.[prodProfits] (@prodId INT)
RETURNS FLOAT
AS
BEGIN
        DECLARE @sum FLOAT;

        SET @sum = (
                SELECT sum(Production.profMarg(@prodId, year(TransactionDate)) *
Quantity)
                FROM Production.TransactionHistory
                WHERE ProductID = @prodId
                        AND TransactionType = 'S'
                )
```

```
	RETURN @sum
END
GO

GO
CREATE FUNCTION production.[totSales] (@prodId INT)
RETURNS FLOAT
AS
BEGIN
	DECLARE @sum FLOAT;

	SET @sum = (
			SELECT sum(ActualCost)
			FROM Production.TransactionHistory
			WHERE ProductID = @prodId
				AND TransactionType = 'S'
			)

	RETURN @sum
END
GO

USE AdventureWorks2017;
SELECT prod.ProductID
	,prod.Name
	,mod.Name AS Model
	,AVG(cost.StandardCost) AS AvgProductionCost
	,AVG(price.ListPrice) AS AvgListPrice
	,AVG(price.ListPrice) - AVG(cost.StandardCost) AS AvgProfitMargs
	,Production.prodProfits(prod.ProductID) AS TotalProfits
```

```
        ,Production.totSales(prod.ProductID) AS TotalSalesAmt
FROM Production.Product AS prod
INNER JOIN Production.ProductModel AS mod ON prod.ProductModelID =
mod.ProductModelID
INNER JOIN Production.ProductListPriceHistory AS price ON prod.ProductID = price.ProductID
INNER JOIN Production.ProductCostHistory AS cost ON prod.ProductID = cost.ProductID
GROUP BY prod.ProductID
        ,prod.Name
        ,mod.Name
ORDER BY TotalProfits DESC
```

Solution with JSON:

```
USE AdventureWorks2017;
GO
CREATE FUNCTION production.[profMarg] (
        @prodId INT
        ,@year INT
        )
RETURNS FLOAT
AS
BEGIN
        DECLARE @cost FLOAT;
        DECLARE @price FLOAT;
        DECLARE @res FLOAT

        SET @cost = (
                        SELECT StandardCost
                        FROM Production.ProductCostHistory
                        WHERE ProductID = @prodId
```

```
                        AND year(StartDate) = @year
                )
        SET @price = (
                SELECT ListPrice
                FROM Production.ProductListPriceHistory
                WHERE ProductID = @prodId
                        AND year(StartDate) = @year
                )
        SET @res = @price - @cost

        RETURN @res
END
GO

GO

CREATE FUNCTION production.[prodProfits] (@prodId INT)
RETURNS FLOAT
AS
BEGIN
        DECLARE @sum FLOAT;

        SET @sum = (
                SELECT sum(Production.profMarg(@prodId, year(TransactionDate)) *
Quantity)
                FROM Production.TransactionHistory
                WHERE ProductID = @prodId
                        AND TransactionType = 'S'
                )
```

```
	RETURN @sum
END
GO

GO
CREATE FUNCTION production.[totSales] (@prodId INT)
RETURNS FLOAT
AS
BEGIN
	DECLARE @sum FLOAT;

	SET @sum = (
			SELECT sum(ActualCost)
			FROM Production.TransactionHistory
			WHERE ProductID = @prodId
				AND TransactionType = 'S'
			)

	RETURN @sum
END
GO

USE AdventureWorks2017;
SELECT prod.ProductID
	,prod.Name
	,mod.Name AS Model
	,AVG(cost.StandardCost) AS AvgProductionCost
	,AVG(price.ListPrice) AS AvgListPrice
	,AVG(price.ListPrice) - AVG(cost.StandardCost) AS AvgProfitMargs
	,Production.prodProfits(prod.ProductID) AS TotalProfits
```

```
        ,Production.totSales(prod.ProductID) AS TotalSalesAmt
FROM Production.Product AS prod
INNER JOIN Production.ProductModel AS mod ON prod.ProductModelID =
mod.ProductModelID
INNER JOIN Production.ProductListPriceHistory AS price ON prod.ProductID = price.ProductID
INNER JOIN Production.ProductCostHistory AS cost ON prod.ProductID = cost.ProductID
GROUP BY prod.ProductID
        ,prod.Name
        ,mod.Name
ORDER BY TotalProfits DESC
FOR JSON PATH, ROOT('Complex 5 Output'), INCLUDE_NULL_VALUES;
```

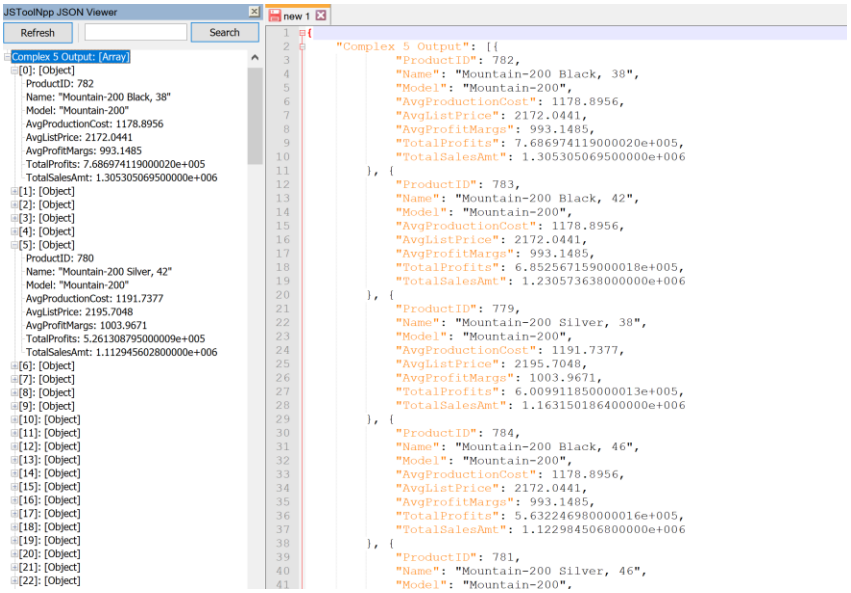
Sample Output (293 results returned):

Results | Messages

	ProductID	Name	Model	AvgProductionCost	AvgListPrice	AvgProfitMargs	TotalProfits	TotalSalesAmt
1	782	Mountain-200 Black, 38	Mountain-200	1178.8956	2172.0441	993.1485	768697.411900002	1305305.0695
2	783	Mountain-200 Black, 42	Mountain-200	1178.8956	2172.0441	993.1485	685256.715900002	1230573.638
3	779	Mountain-200 Silver, 38	Mountain-200	1191.7377	2195.7048	1003.9671	600991.185000001	1163150.1864
4	784	Mountain-200 Black, 46	Mountain-200	1178.8956	2172.0441	993.1485	563224.698000002	1122984.5068
5	781	Mountain-200 Silver, 46	Mountain-200	1191.7377	2195.7048	1003.9671	534565.843500001	1140692.6832
6	780	Mountain-200 Silver, 42	Mountain-200	1191.7377	2195.7048	1003.9671	526130.879500001	1112945.6028
7	969	Touring-1000 Blue, 60	Touring-1000	1481.9379	2384.07	902.1321	476325.748799999	765000.3816
8	976	Road-350-W Yellow, 48	Road-350-W	1082.51	1700.99	618.48	465096.96	705434.5728
9	957	Touring-1000 Yellow, 60	Touring-1000	1481.9379	2384.07	902.1321	460989.503099999	736677.63

Query executed successfully. localhost, 12001 (15

Sample JSON Output (293 results returned):



JSToolNpp JSON Viewer

Refresh | Search

Complex 5 Output: [Array]
- [0]: [Object]
  - ProductID: 782
  - Name: "Mountain-200 Black, 38"
  - Model: "Mountain-200"
  - AvgProductionCost: 1178.8956
  - AvgListPrice: 2172.0441
  - AvgProfitMargs: 993.1485
  - TotalProfits: 7.686974119000020e+005
  - TotalSalesAmt: 1.305305069500000e+006
- [1]: [Object]
- [2]: [Object]
- [3]: [Object]
- [4]: [Object]
- [5]: [Object]
  - ProductID: 780
  - Name: "Mountain-200 Silver, 42"
  - Model: "Mountain-200"
  - AvgProductionCost: 1191.7377
  - AvgListPrice: 2195.7048
  - AvgProfitMargs: 1003.9671
  - TotalProfits: 5.261308795000009e+005
  - TotalSalesAmt: 1.112945602800000e+006
- [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]

new 1

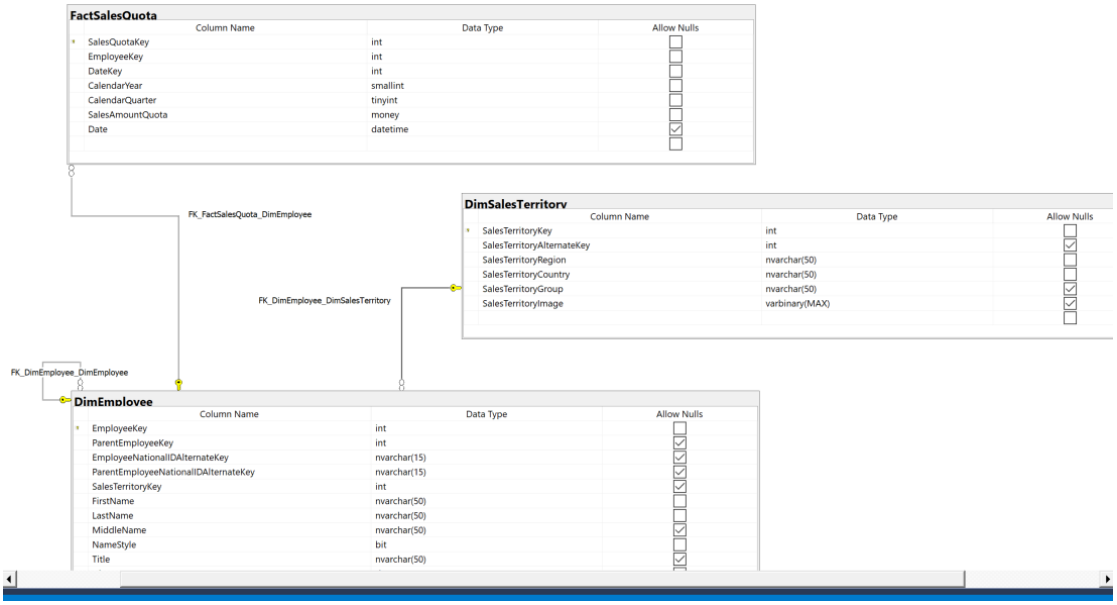

```
{
    "Complex 5 Output": [{
            "ProductID": 782,
            "Name": "Mountain-200 Black, 38",
            "Model": "Mountain-200",
            "AvgProductionCost": 1178.8956,
            "AvgListPrice": 2172.0441,
            "AvgProfitMargs": 993.1485,
            "TotalProfits": 7.686974119000020e+005,
            "TotalSalesAmt": 1.305305069500000e+006
        }, {
            "ProductID": 783,
            "Name": "Mountain-200 Black, 42",
            "Model": "Mountain-200",
            "AvgProductionCost": 1178.8956,
            "AvgListPrice": 2172.0441,
            "AvgProfitMargs": 993.1485,
            "TotalProfits": 6.852567159000018e+005,
            "TotalSalesAmt": 1.230573638000000e+006
        }, {
            "ProductID": 779,
            "Name": "Mountain-200 Silver, 38",
            "Model": "Mountain-200",
            "AvgProductionCost": 1191.7377,
            "AvgListPrice": 2195.7048,
            "AvgProfitMargs": 1003.9671,
            "TotalProfits": 6.009911850000013e+005,
            "TotalSalesAmt": 1.163150186400000e+006
        }, {
            "ProductID": 784,
            "Name": "Mountain-200 Black, 46",
            "Model": "Mountain-200",
            "AvgProductionCost": 1178.8956,
            "AvgListPrice": 2172.0441,
            "AvgProfitMargs": 993.1485,
            "TotalProfits": 5.632246980000016e+005,
            "TotalSalesAmt": 1.122984506800000e+006
        }, {
            "ProductID": 781,
            "Name": "Mountain-200 Silver, 46",
            "Model": "Mountain-200",
```

COMPLEX 6:

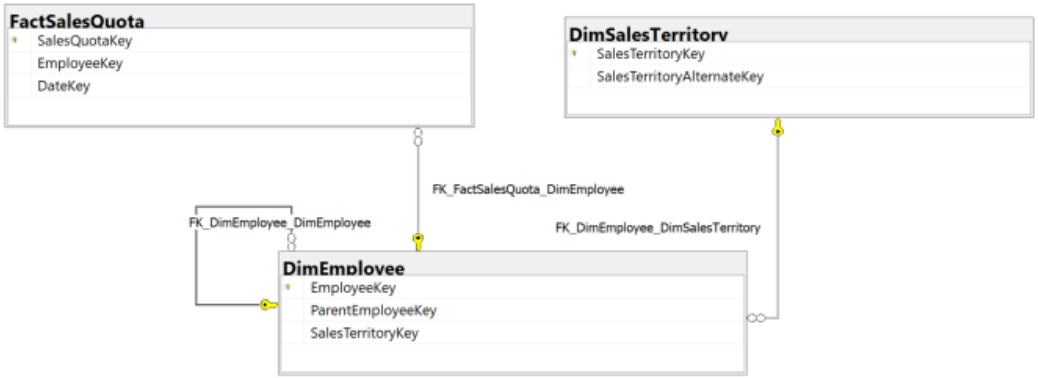
QUERY: This query lists the sales quota for each quarter over 3 years for employee and compares that amount to their actual sales amount during the corresponding time periods.

FUNCTIONS: [dateToQtr] converts the date to quarter. [avgQtrQuota] finds the average quota amount for the specified year and qtr. [actQtrSales] finds the actual sales amount for the specified qtr and year.

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Dbo.DimEmployee	EmployeeKey FirstName LastName Title
Dbo.DimSalesTerritory	SalesTerritoryCountry
Dbo.FactSalesQuota	CalendarYear CalendarQuarter QtrQuota

Order By:

Table Name	Column Name	Sort Order
Dbo.DimEMployee	EmployeeKey	ASC
Dbo.FactSalesQuota	CalendarYear CalendarQuota	ASC

Solution without JSON:

```
USE AdventureWorksDW2017
GO
CREATE FUNCTION dbo.[dateToQtr] (@date DATE)
RETURNS INT
AS
BEGIN
	DECLARE @val INT;
	DECLARE @month INT;

	SET @month = MONTH(@date);

	SELECT @val = CASE
			WHEN @month BETWEEN 1
				AND 3
			THEN 1
			WHEN @month BETWEEN 4
```

```
                                AND 6
                        THEN 2
                WHEN @month BETWEEN 7
                                AND 9
                        THEN 3
                WHEN @month BETWEEN 10
                                AND 12
                        THEN 4
                ELSE 00000 --unknown qtr
                END

        RETURN @val
END
GO

GO
CREATE FUNCTION dbo.[avgQtrQuota] (
        @empkey INT
        ,@calyear INT
        ,@calqtr INT
        )
RETURNS FLOAT
AS
BEGIN
        DECLARE @val FLOAT;

        SET @val = (
                        SELECT avg(SalesAmountQuota)
                        FROM dbo.FactSalesQuota
                        WHERE EmployeeKey = @empkey
```

```
                        AND CalendarYear = @calyear
                        AND CalendarQuarter = @calqtr
                )

        RETURN @val;
END
GO

GO
CREATE FUNCTION dbo.[actQtrSales] (
        @empkey INT
        ,@calyear INT
        ,@calqtr INT
        )
RETURNS FLOAT
AS
BEGIN
        DECLARE @val FLOAT;

        SET @val = (
                        SELECT sum(SalesAmount)
                        FROM dbo.FactResellerSales
                        WHERE EmployeeKey = @empkey
                                AND year(ShipDate) = @calyear
                                AND dbo.dateToQtr(ShipDate) = @calqtr
                        )

        RETURN @val;
END
GO
```

```
USE AdventureWorksDW2017
SELECT emp.EmployeeKey
	,emp.FirstName
	,emp.LastName
	,emp.Title
	,terr.SalesTerritoryCountry
	,quota.CalendarYear
	,quota.CalendarQuarter
	,avg(quota.SalesAmountQuota) AS QtrQuota
	,dbo.actQtrSales(emp.EmployeeKey, quota.CalendarYear, quota.CalendarQuarter) AS QuarterSales
FROM dbo.FactSalesQuota AS quota
INNER JOIN dbo.DimEmployee AS emp ON emp.EmployeeKey = quota.EmployeeKey
INNER JOIN dbo.DimSalesTerritory AS terr ON terr.SalesTerritoryKey = emp.SalesTerritoryKey
WHERE quota.CalendarYear != 2010
GROUP BY quota.CalendarYear
	,quota.CalendarQuarter
	,emp.EmployeeKey
	,emp.FirstName
	,emp.LastName
	,emp.Title
	,terr.SalesTerritoryCountry
ORDER BY emp.EmployeeKey
	,quota.CalendarYear
	,quota.CalendarQuarter
```

Solution with JSON:

```
USE AdventureWorksDW2017
GO
```

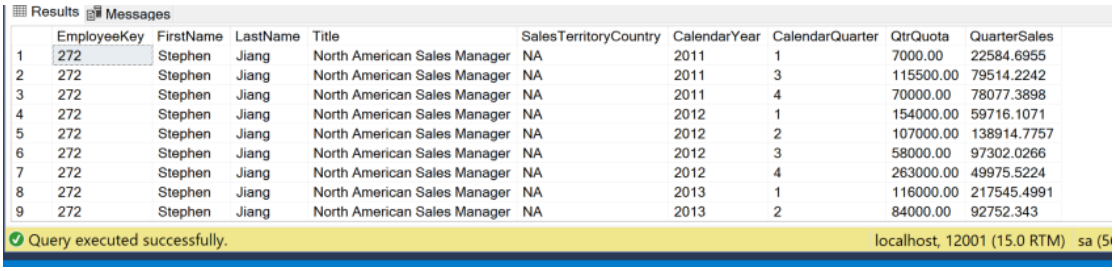
```
CREATE FUNCTION dbo.[dateToQtr] (@date DATE)
RETURNS INT
AS
BEGIN
	DECLARE @val INT;
	DECLARE @month INT;

	SET @month = MONTH(@date);

	SELECT @val = CASE
			WHEN @month BETWEEN 1
					AND 3
				THEN 1
			WHEN @month BETWEEN 4
					AND 6
				THEN 2
			WHEN @month BETWEEN 7
					AND 9
				THEN 3
			WHEN @month BETWEEN 10
					AND 12
				THEN 4
			ELSE 00000 --unknown qtr
			END

	RETURN @val
END
GO

GO
```

```
CREATE FUNCTION dbo.[avgQtrQuota] (
	@empkey INT
	,@calyear INT
	,@calqtr INT
	)
RETURNS FLOAT
AS
BEGIN
	DECLARE @val FLOAT;

	SET @val = (
			SELECT avg(SalesAmountQuota)
			FROM dbo.FactSalesQuota
			WHERE EmployeeKey = @empkey
				AND CalendarYear = @calyear
				AND CalendarQuarter = @calqtr
			)

	RETURN @val;
END
GO

GO
CREATE FUNCTION dbo.[actQtrSales] (
	@empkey INT
	,@calyear INT
	,@calqtr INT
	)
RETURNS FLOAT
AS
```

```
BEGIN
	DECLARE @val FLOAT;

	SET @val = (
			SELECT sum(SalesAmount)
			FROM dbo.FactResellerSales
			WHERE EmployeeKey = @empkey
				AND year(ShipDate) = @calyear
				AND dbo.dateToQtr(ShipDate) = @calqtr
			)

	RETURN @val;
END
GO

USE AdventureWorksDW2017
SELECT emp.EmployeeKey
	,emp.FirstName
	,emp.LastName
	,emp.Title
	,terr.SalesTerritoryCountry
	,quota.CalendarYear
	,quota.CalendarQuarter
	,avg(quota.SalesAmountQuota) AS QtrQuota
	,dbo.actQtrSales(emp.EmployeeKey, quota.CalendarYear, quota.CalendarQuarter) AS
QuarterSales
FROM dbo.FactSalesQuota AS quota
INNER JOIN dbo.DimEmployee AS emp ON emp.EmployeeKey = quota.EmployeeKey
INNER JOIN dbo.DimSalesTerritory AS terr ON terr.SalesTerritoryKey = emp.SalesTerritoryKey
WHERE quota.CalendarYear != 2010
```

```
GROUP BY quota.CalendarYear
        ,quota.CalendarQuarter
        ,emp.EmployeeKey
        ,emp.FirstName
        ,emp.LastName
        ,emp.Title
        ,terr.SalesTerritoryCountry
ORDER BY emp.EmployeeKey
        ,quota.CalendarYear
        ,quota.CalendarQuarter
```

Sample Output (143 results returned):

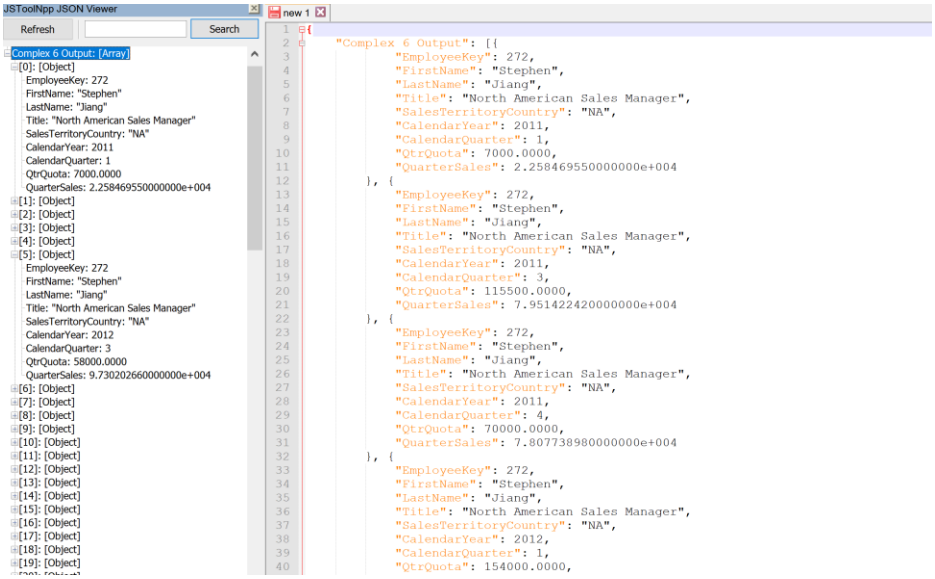


Results | Messages

	EmployeeKey	FirstName	LastName	Title	SalesTerritoryCountry	CalendarYear	CalendarQuarter	QtrQuota	QuarterSales
1	272	Stephen	Jiang	North American Sales Manager	NA	2011	1	7000.00	22584.6955
2	272	Stephen	Jiang	North American Sales Manager	NA	2011	3	115500.00	79514.2242
3	272	Stephen	Jiang	North American Sales Manager	NA	2011	4	70000.00	78077.3898
4	272	Stephen	Jiang	North American Sales Manager	NA	2012	1	154000.00	59716.1071
5	272	Stephen	Jiang	North American Sales Manager	NA	2012	2	107000.00	138914.7757
6	272	Stephen	Jiang	North American Sales Manager	NA	2012	3	58000.00	97302.0266
7	272	Stephen	Jiang	North American Sales Manager	NA	2012	4	263000.00	49975.5224
8	272	Stephen	Jiang	North American Sales Manager	NA	2013	1	116000.00	217545.4991
9	272	Stephen	Jiang	North American Sales Manager	NA	2013	2	84000.00	92752.343

Query executed successfully. localhost, 12001 (15.0 RTM) sa (5

Sample JSON Output (143 results returned):



JSTOolNpp JSON Viewer — Refresh — Search

```
Complex 6 Output: [Array]
  [0]: [Object]
    EmployeeKey: 272
    FirstName: "Stephen"
    LastName: "Jiang"
    Title: "North American Sales Manager"
    SalesTerritoryCountry: "NA"
    CalendarYear: 2011
    CalendarQuarter: 1
    QtrQuota: 7000.0000
    QuarterSales: 2.258469550000000e+004
  [1]: [Object]
  [2]: [Object]
  [3]: [Object]
  [4]: [Object]
  [5]: [Object]
    EmployeeKey: 272
    FirstName: "Stephen"
    LastName: "Jiang"
    Title: "North American Sales Manager"
    SalesTerritoryCountry: "NA"
    CalendarYear: 2012
    CalendarQuarter: 3
    QtrQuota: 58000.0000
    QuarterSales: 9.730202660000000e+004
  [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]
```

new 1

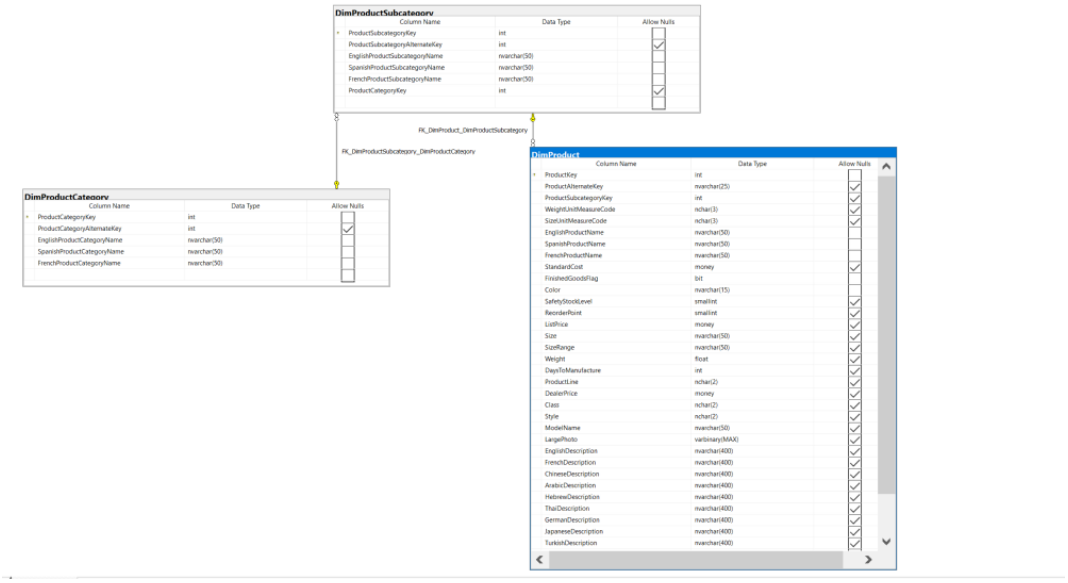
```
{
    "Complex 6 Output": [{
            "EmployeeKey": 272,
            "FirstName": "Stephen",
            "LastName": "Jiang",
            "Title": "North American Sales Manager",
            "SalesTerritoryCountry": "NA",
            "CalendarYear": 2011,
            "CalendarQuarter": 1,
            "QtrQuota": 7000.0000,
            "QuarterSales": 2.258469550000000e+004
        }, {
            "EmployeeKey": 272,
            "FirstName": "Stephen",
            "LastName": "Jiang",
            "Title": "North American Sales Manager",
            "SalesTerritoryCountry": "NA",
            "CalendarYear": 2011,
            "CalendarQuarter": 3,
            "QtrQuota": 115500.0000,
            "QuarterSales": 7.951422420000000e+004
        }, {
            "EmployeeKey": 272,
            "FirstName": "Stephen",
            "LastName": "Jiang",
            "Title": "North American Sales Manager",
            "SalesTerritoryCountry": "NA",
            "CalendarYear": 2011,
            "CalendarQuarter": 4,
            "QtrQuota": 70000.0000,
            "QuarterSales": 7.807738980000000e+004
        }, {
            "EmployeeKey": 272,
            "FirstName": "Stephen",
            "LastName": "Jiang",
            "Title": "North American Sales Manager",
            "SalesTerritoryCountry": "NA",
            "CalendarYear": 2012,
            "CalendarQuarter": 1,
            "QtrQuota": 154000.0000,
```

COMPLEX 7:

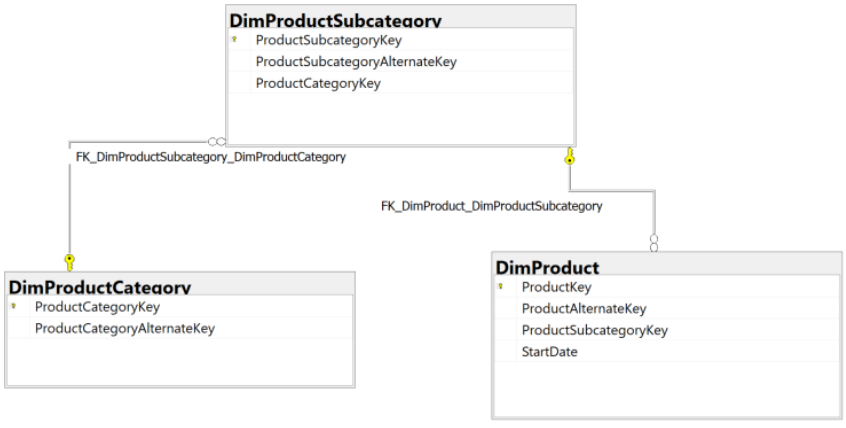
QUERY: This query ranks products based on the amount sold by reseller and sold online and gives some information about each product.

FUNCTIONS: [numProdSold] finds the total amount of sales for the specified product online and by resellers

Standard View:



Key View:



Columns from tables:

Table Name	Column Name
Dbo.DimProduct	ProductKey EnglishProductName ListPrice TotalUnitsSold
Dbo.DimProductCategory	EnglishProductCategoryName
Dbo.DimProductSubcategory	EnglishProductSubcategoryName

Order By:

Table Name	Column Name	Sort Order
Dbo.DimProduct	TotalUnitsSold	DESC

Solution without JSON:

```
USE AdventureWorksDW2017;
GO
CREATE FUNCTION dbo.[numProdSold] (@prodId INT)
RETURNS INT
AS
BEGIN
	DECLARE @val INT;
	DECLARE @sum1 INT;
	DECLARE @sum2 INT;

	SET @sum1 = (
			SELECT sum(OrderQuantity)
			FROM dbo.FactInternetSales
			WHERE ProductKey = @prodId
			)
	SET @sum2 = (
			SELECT sum(OrderQuantity)
```

```
                    FROM dbo.FactResellerSales
                    WHERE ProductKey = @prodId
                    )
    SET @val = @sum1 + @sum2

    RETURN @val;
END
GO

USE AdventureWorksDW2017;
SELECT prod.ProductKey
    ,prod.EnglishProductName
    ,subcat.EnglishProductSubcategoryName
    ,cat.EnglishProductCategoryName
    ,prod.ListPrice
    ,dbo.numProdSold(prod.ProductKey) AS TotalUnitsSold
FROM dbo.DimProductSubcategory AS subcat
INNER JOIN dbo.DimProduct AS prod ON prod.ProductSubcategoryKey =
subcat.ProductSubcategoryKey
INNER JOIN dbo.DimProductCategory AS cat ON subcat.ProductCategoryKey =
cat.ProductCategoryKey
GROUP BY prod.ProductKey
    ,prod.EnglishProductName
    ,subcat.EnglishProductSubcategoryName
    ,cat.EnglishProductCategoryName
    ,prod.ListPrice
ORDER BY TotalUnitsSold DESC
```

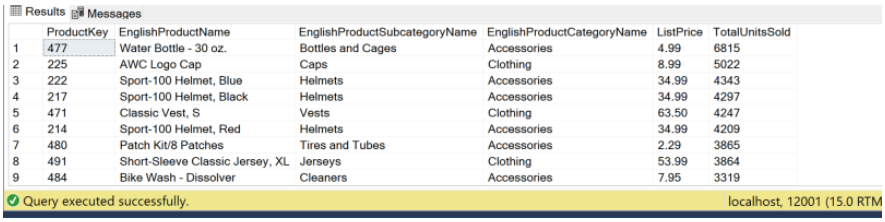
Solution with JSON:

```
USE AdventureWorksDW2017;
```

```
GO
CREATE FUNCTION dbo.[numProdSold] (@prodId INT)
RETURNS INT
AS
BEGIN
	DECLARE @val INT;
	DECLARE @sum1 INT;
	DECLARE @sum2 INT;

	SET @sum1 = (
			SELECT sum(OrderQuantity)
			FROM dbo.FactInternetSales
			WHERE ProductKey = @prodId
			)
	SET @sum2 = (
			SELECT sum(OrderQuantity)
			FROM dbo.FactResellerSales
			WHERE ProductKey = @prodId
			)
	SET @val = @sum1 + @sum2

	RETURN @val;
END
GO

USE AdventureWorksDW2017;
SELECT prod.ProductKey
	,prod.EnglishProductName
	,subcat.EnglishProductSubcategoryName
	,cat.EnglishProductCategoryName
```

```
        ,prod.ListPrice
        ,dbo.numProdSold(prod.ProductKey) AS TotalUnitsSold
FROM dbo.DimProductSubcategory AS subcat
INNER JOIN dbo.DimProduct AS prod ON prod.ProductSubcategoryKey =
subcat.ProductSubcategoryKey
INNER JOIN dbo.DimProductCategory AS cat ON subcat.ProductCategoryKey =
cat.ProductCategoryKey
GROUP BY prod.ProductKey
        ,prod.EnglishProductName
        ,subcat.EnglishProductSubcategoryName
        ,cat.EnglishProductCategoryName
        ,prod.ListPrice
ORDER BY TotalUnitsSold DESC
FOR JSON PATH, ROOT('Complex 7 Output'), INCLUDE_NULL_VALUES;
```

Sample Output (397 results returned):

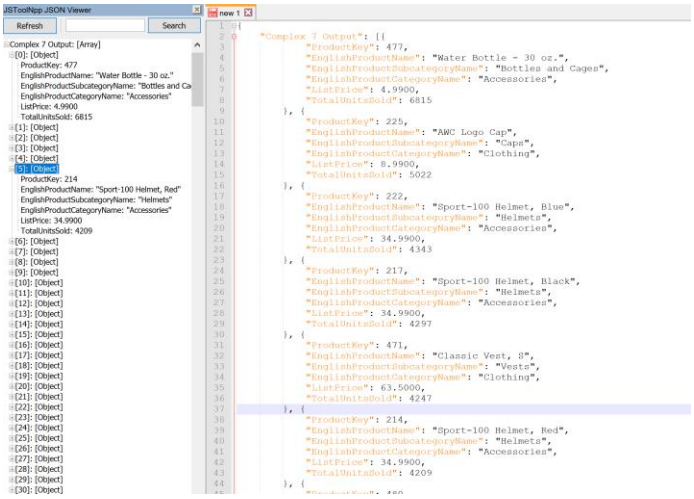


Results | Messages

	ProductKey	EnglishProductName	EnglishProductSubcategoryName	EnglishProductCategoryName	ListPrice	TotalUnitsSold
1	477	Water Bottle - 30 oz.	Bottles and Cages	Accessories	4.99	6815
2	225	AWC Logo Cap	Caps	Clothing	8.99	5022
3	222	Sport-100 Helmet, Blue	Helmets	Accessories	34.99	4343
4	217	Sport-100 Helmet, Black	Helmets	Accessories	34.99	4297
5	471	Classic Vest, S	Vests	Clothing	63.50	4247
6	214	Sport-100 Helmet, Red	Helmets	Accessories	34.99	4209
7	480	Patch Kit/8 Patches	Tires and Tubes	Accessories	2.29	3865
8	491	Short-Sleeve Classic Jersey, XL	Jerseys	Clothing	53.99	3864
9	484	Bike Wash - Dissolver	Cleaners	Accessories	7.95	3319

Query executed successfully. localhost, 12001 (15.0 RTM

Sample JSON Output (397 results returned):



JSToolNpp JSON Viewer — Refresh — Search — new 1


```
{
  "Complex 7 Output": [{
        "ProductKey": 477,
        "EnglishProductName": "Water Bottle - 30 oz.",
        "EnglishProductSubcategoryName": "Bottles and Cages",
        "EnglishProductCategoryName": "Accessories",
        "ListPrice": 4.9900,
        "TotalUnitsSold": 6815
    }, {
        "ProductKey": 225,
        "EnglishProductName": "AWC Logo Cap",
        "EnglishProductSubcategoryName": "Caps",
        "EnglishProductCategoryName": "Clothing",
        "ListPrice": 8.9900,
        "TotalUnitsSold": 5022
    }, {
        "ProductKey": 222,
        "EnglishProductName": "Sport-100 Helmet, Blue",
        "EnglishProductSubcategoryName": "Helmets",
        "EnglishProductCategoryName": "Accessories",
        "ListPrice": 34.9900,
        "TotalUnitsSold": 4343
    }, {
        "ProductKey": 217,
        "EnglishProductName": "Sport-100 Helmet, Black",
        "EnglishProductSubcategoryName": "Helmets",
        "EnglishProductCategoryName": "Accessories",
        "ListPrice": 34.9900,
        "TotalUnitsSold": 4297
    }, {
        "ProductKey": 471,
        "EnglishProductName": "Classic Vest, S",
        "EnglishProductSubcategoryName": "Vests",
        "EnglishProductCategoryName": "Clothing",
        "ListPrice": 63.5000,
        "TotalUnitsSold": 4247
    }, {
        "ProductKey": 214,
        "EnglishProductName": "Sport-100 Helmet, Red",
        "EnglishProductSubcategoryName": "Helmets",
        "EnglishProductCategoryName": "Accessories",
        "ListPrice": 34.9900,
        "TotalUnitsSold": 4209
    }, {
```