9-15-GROUP 8 – PROJECT 1

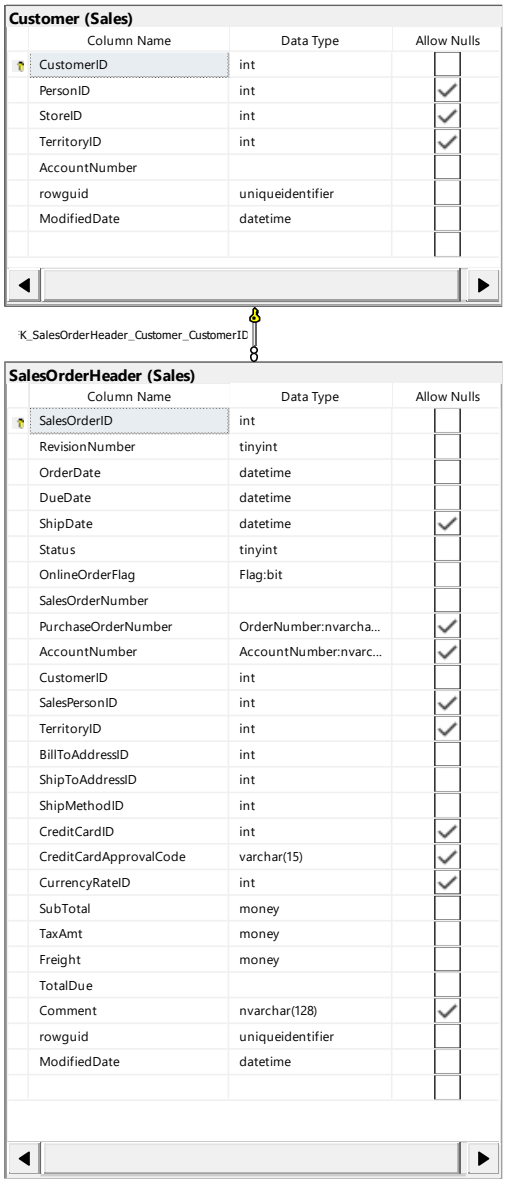
Prepared By: Humaira Qadeer

Contents

Proposition 01: Find by Customer, all orders placed with the date of order placed

Explanation: Find the salesorderid, customerid, accountnumber and orderdate for all orders placed in the year 2014

Columns From Standard View



Customer (Sales)

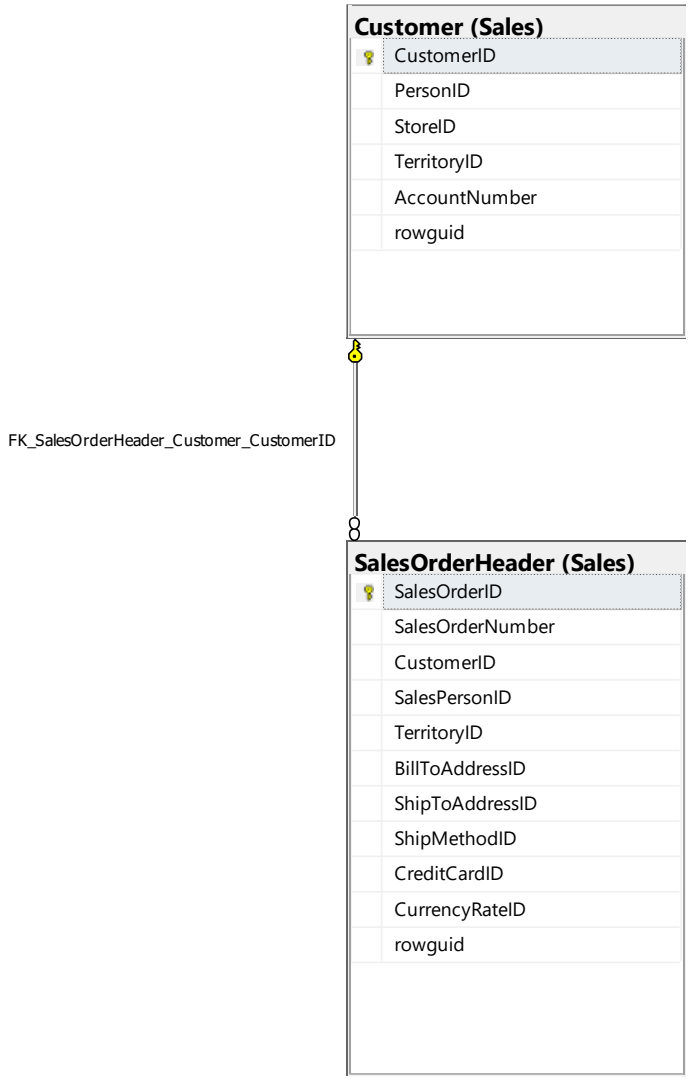
Column Name	Data Type	Allow Nulls
CustomerID	int	☐
PersonID	int	☑
StoreID	int	☑
TerritoryID	int	☑
AccountNumber		☐
rowguid	uniqueidentifier	☐
ModifiedDate	datetime	☐

K_SalesOrderHeader_Customer_CustomerID

SalesOrderHeader (Sales)

Column Name	Data Type	Allow Nulls
SalesOrderID	int	☐
RevisionNumber	tinyint	☐
OrderDate	datetime	☐
DueDate	datetime	☐
ShipDate	datetime	☑
Status	tinyint	☐
OnlineOrderFlag	Flag:bit	☐
SalesOrderNumber		☐
PurchaseOrderNumber	OrderNumber:nvarcha...	☑
AccountNumber	AccountNumber:nvarc...	☑
CustomerID	int	☐
SalesPersonID	int	☑
TerritoryID	int	☑
BillToAddressID	int	☐
ShipToAddressID	int	☐
ShipMethodID	int	☐
CreditCardID	int	☑
CreditCardApprovalCode	varchar(15)	☑
CurrencyRateID	int	☑
SubTotal	money	☐
TaxAmt	money	☐
Freight	money	☐
TotalDue		☐
Comment	nvarchar(128)	☑
rowguid	uniqueidentifier	☐
ModifiedDate	datetime	☐

Columns From Key View



Columns from their respective tables in the select clause

Table Name	Column Name
SalesOrderHeader	SalesOrderID OrderDate
Customer	CustomerId AccountNumber

Order By

Table Name	Column Name	Sort Order
Customer	CustomerId	
Order	OrderDate	

Problem solving query

```
USE AdventureWorks2017;
SELECT OH.SalesOrderID,
       C.CustomerID,
       C.AccountNumber,
       OH.OrderDate
FROM Sales.Customer AS C
    INNER JOIN Sales.SalesOrderHeader AS OH
        ON C.CustomerID = OH.CustomerID
WHERE YEAR(OH.OrderDate) = 2014
ORDER BY OH.CustomerID,
         OH.OrderDate;
--FOR JSON PATH, ROOT('CustomerOrders'), INCLUDE_NULL_VALUES;
```

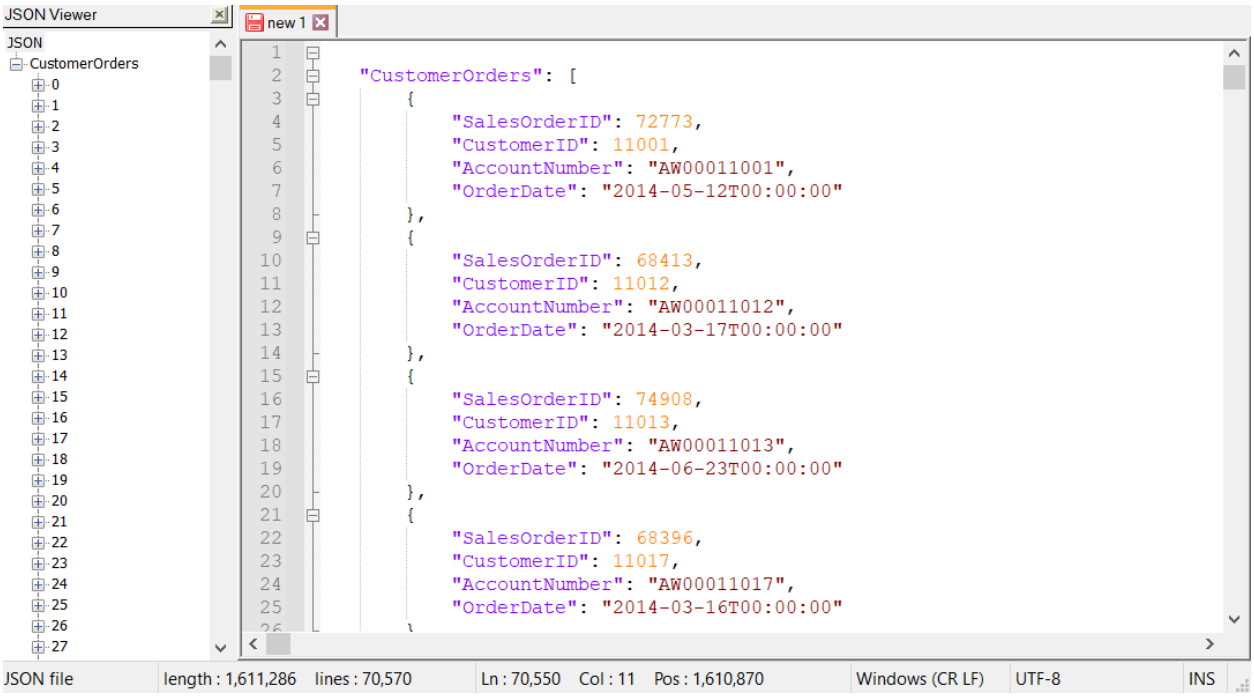
Sample Relational Output with total number of rows returned (11761)

	SalesOrderID	CustomerID	AccountNumber	OrderDate
1	72773	11001	AW00011001	2014-05-12 00:00:00.000
2	68413	11012	AW00011012	2014-03-17 00:00:00.000
3	74908	11013	AW00011013	2014-06-23 00:00:00.000
4	68396	11017	AW00011017	2014-03-16 00:00:00.000
5	69087	11018	AW00011018	2014-03-26 00:00:00.000
6	64737	11019	AW00011019	2014-01-22 00:00:00.000
7	65967	11019	AW00011019	2014-02-08 00:00:00.000
8	66658	11019	AW00011019	2014-02-19 00:00:00.000
9	68563	11019	AW00011019	2014-03-19 00:00:00.000
10	69177	11019	AW00011019	2014-03-28 00:00:00.000
11	71963	11019	AW00011019	2014-05-01 00:00:00.000
12	72015	11019	AW00011019	2014-05-02 00:00:00.000
13	72747	11019	AW00011019	2014-05-12 00:00:00.000
14	74561	11019	AW00011019	2014-06-13 00:00:00.000
15	74593	11019	AW00011019	2014-06-14 00:00:00.000
16	74658	11023	AW00011023	2014-06-16 00:00:00.000
17	69157	11025	AW00011025	2014-03-27 00:00:00.000
18	68468	11026	AW00011026	2014-03-17 00:00:00.000
19	69088	11027	AW00011027	2014-03-26 00:00:00.000
20	67961	11028	AW00011028	2014-03-09 00:00:00.000
21	70593	11029	AW00011029	2014-04-13 00:00:00.000
22	70449	11030	AW00011030	2014-04-11 00:00:00.000
23	70748	11031	AW00011031	2014-04-15 00:00:00.000
24	64042	11032	AW00011032	2014-01-11 00:00:00.000
25	70373	11032	AW00011032	2014-04-10 00:00:00.000
26	63409	11033	AW00011033	2014-01-01 00:00:00.000
27	70163	11033	AW00011033	2014-04-07 00:00:00.000
28	63737	11034	AW00011034	2014-01-06 00:00:00.000
29	70521	11034	AW00011034	2014-04-12 00:00:00.000
30	66140	11035	AW00011035	2014-02-10 00:00:00.000
31	70870	11035	AW00011035	2014-04-17 00:00:00.000
32	65549	11038	AW00011038	2014-02-01 00:00:00.000
33	70671	11038	AW00011038	2014-04-14 00:00:00.000
34	66747	11039	AW00011039	2014-02-20 00:00:00.000
35	70594	11039	AW00011039	2014-04-13 00:00:00.000
36	68609	11042	AW00011042	2014-03-19 00:00:00.000
37	64496	11043	AW00011043	2014-01-18 00:00:00.000
38	70074	11044	AW00011044	2014-04-06 00:00:00.000
39	70374	11044	AW00011044	2014-04-10 00:00:00.000
40	69572	11045	AW00011045	2014-03-31 00:00:00.000
41	69653	11046	AW00011046	2014-03-31 00:00:00.000
42	70651	11046	AW00011046	2014-04-14 00:00:00.000
43	70087	11047	AW00011047	2014-04-06 00:00:00.000
44	71132	11047	AW00011047	2014-04-21 00:00:00.000

Sample Json Output with total number of rows returned(11761)

```
USE AdventureWorks2017;

SELECT OH.SalesOrderID,
       C.CustomerID,
       C.AccountNumber,
       OH.OrderDate
```

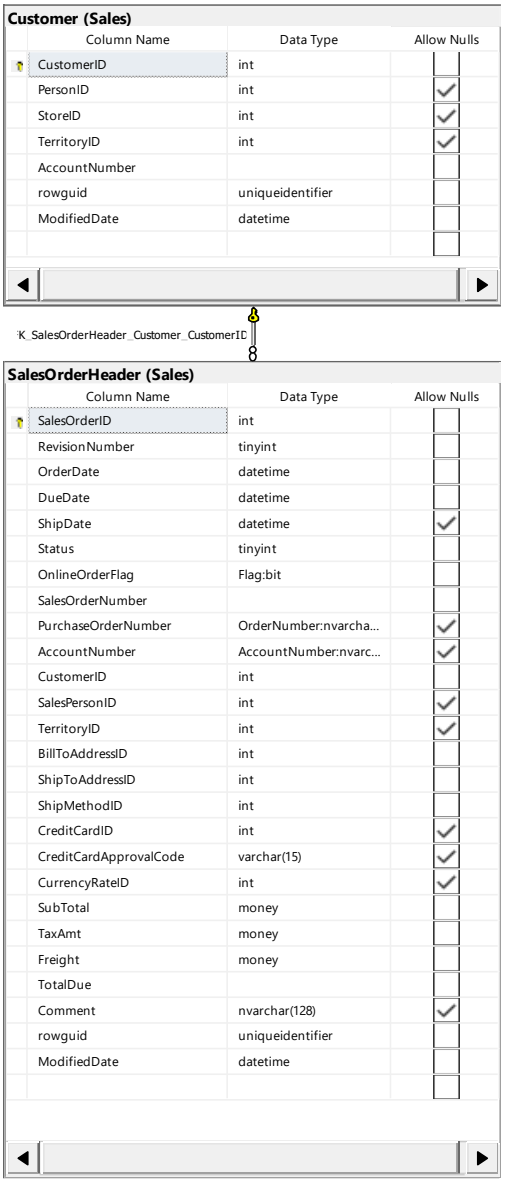
```
FROM Sales.Customer AS C
    INNER JOIN Sales.SalesOrderHeader AS OH
        ON C.CustomerID = OH.CustomerID
WHERE YEAR(OH.OrderDate) = 2014
ORDER BY OH.CustomerID,
         OH.OrderDate
FOR JSON PATH, ROOT('CustomerOrders'), INCLUDE_NULL_VALUES;
```



Proposition 02 – Find all customers who have not placed orders

Explanation: Using the salesorderid and customer id to find matching orders and include customers with no orders

Columns from Standard view



Customer (Sales)

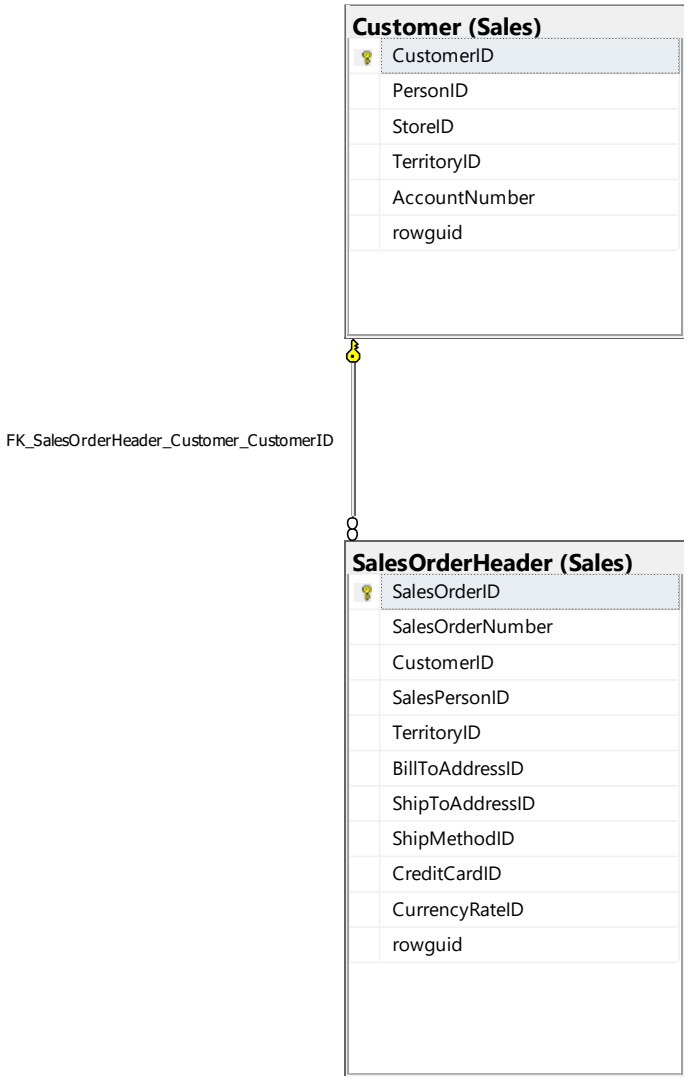
Column Name	Data Type	Allow Nulls
CustomerID	int	
PersonID	int	✓
StoreID	int	✓
TerritoryID	int	✓
AccountNumber		
rowguid	uniqueidentifier	
ModifiedDate	datetime	

FK_SalesOrderHeader_Customer_CustomerID

SalesOrderHeader (Sales)

Column Name	Data Type	Allow Nulls
SalesOrderID	int	
RevisionNumber	tinyint	
OrderDate	datetime	
DueDate	datetime	
ShipDate	datetime	✓
Status	tinyint	
OnlineOrderFlag	Flag:bit	
SalesOrderNumber		
PurchaseOrderNumber	OrderNumber:nvarcha...	✓
AccountNumber	AccountNumber:nvarc...	✓
CustomerID	int	
SalesPersonID	int	✓
TerritoryID	int	✓
BillToAddressID	int	
ShipToAddressID	int	
ShipMethodID	int	
CreditCardID	int	✓
CreditCardApprovalCode	varchar(15)	✓
CurrencyRateID	int	✓
SubTotal	money	
TaxAmt	money	
Freight	money	
TotalDue		
Comment	nvarchar(128)	✓
rowguid	uniqueidentifier	
ModifiedDate	datetime	

Columns from key view

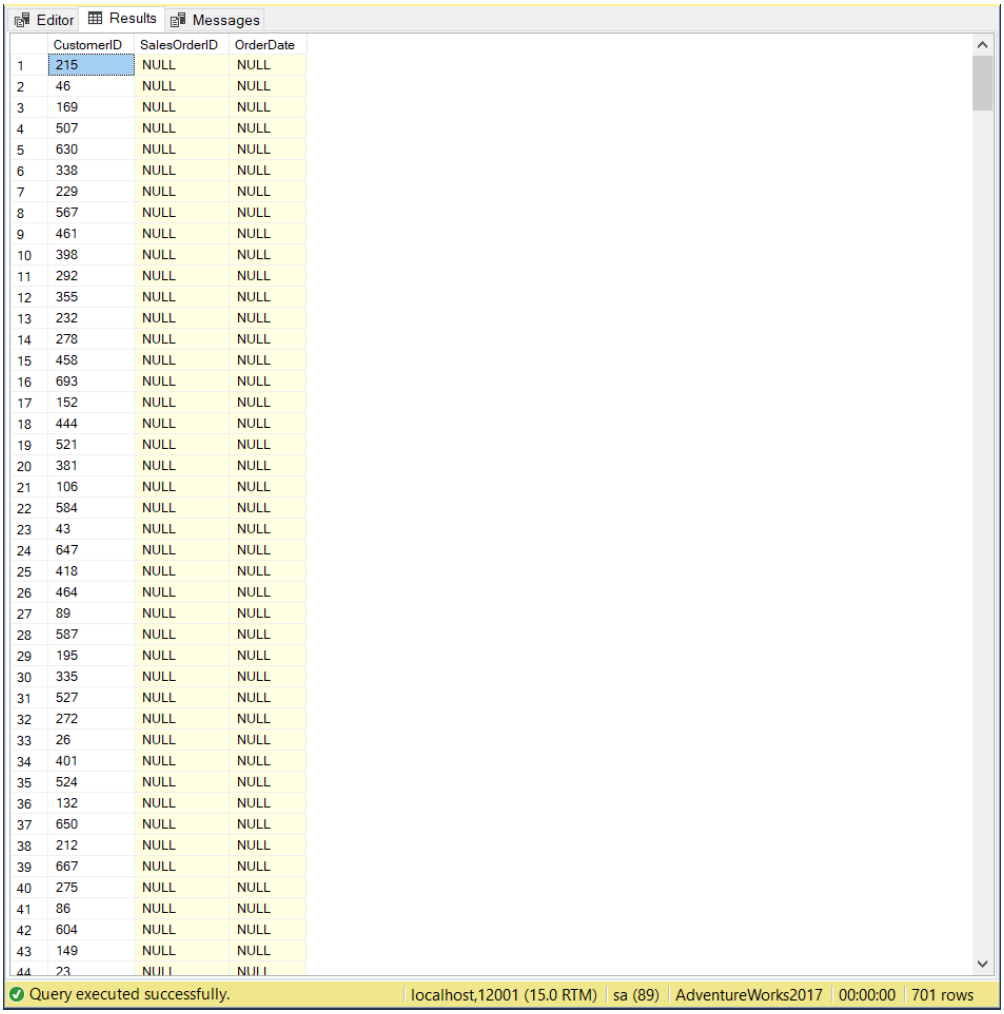


Columns from their respective tables in the select clause

Table Name	Column Name
Customer	OrderDate CustomerId
SalesOrderHeader	SalesOrderID

Problem solving Query

```
USE AdventureWorks2017;
SELECT C.CustomerID,
       S.SalesOrderID,
       S.OrderDate
FROM Sales.Customer AS C
    LEFT OUTER JOIN Sales.SalesOrderHeader AS S
        ON C.CustomerID = S.CustomerID
WHERE S.SalesOrderID IS NULL;
```



Editor | Results | Messages

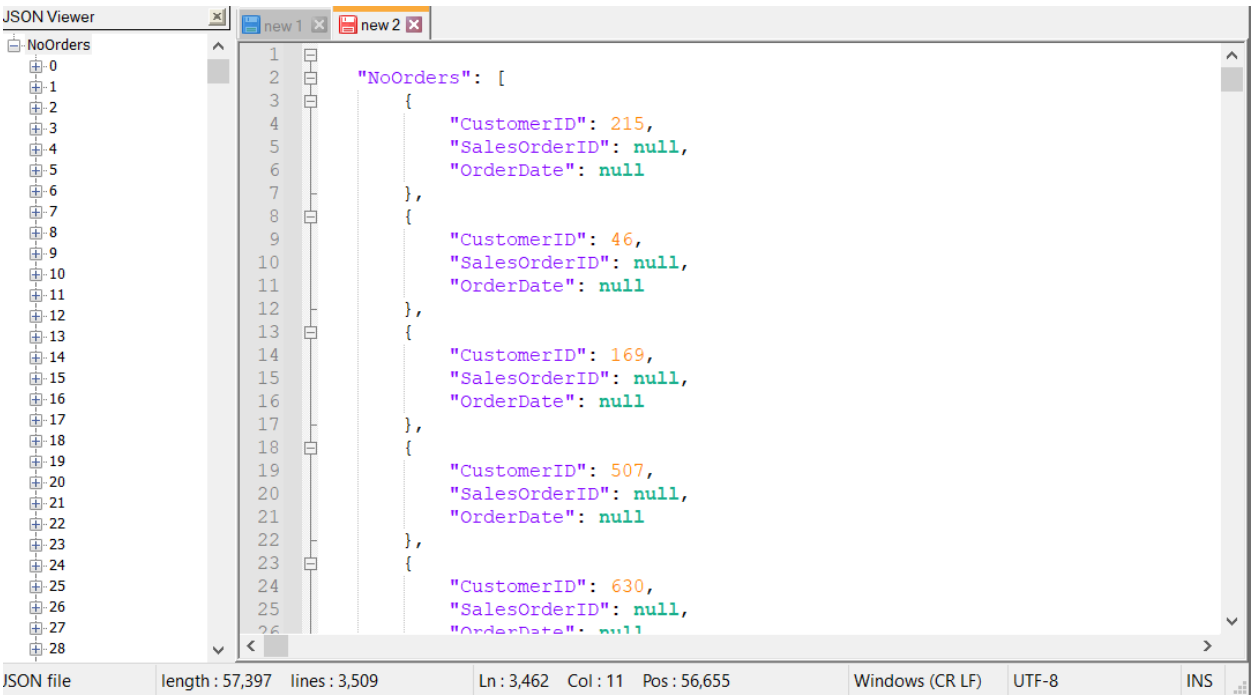
	CustomerID	SalesOrderID	OrderDate
1	215	NULL	NULL
2	46	NULL	NULL
3	169	NULL	NULL
4	507	NULL	NULL
5	630	NULL	NULL
6	338	NULL	NULL
7	229	NULL	NULL
8	567	NULL	NULL
9	461	NULL	NULL
10	398	NULL	NULL
11	292	NULL	NULL
12	355	NULL	NULL
13	232	NULL	NULL
14	278	NULL	NULL
15	458	NULL	NULL
16	693	NULL	NULL
17	152	NULL	NULL
18	444	NULL	NULL
19	521	NULL	NULL
20	381	NULL	NULL
21	106	NULL	NULL
22	584	NULL	NULL
23	43	NULL	NULL
24	647	NULL	NULL
25	418	NULL	NULL
26	464	NULL	NULL
27	89	NULL	NULL
28	587	NULL	NULL
29	195	NULL	NULL
30	335	NULL	NULL
31	527	NULL	NULL
32	272	NULL	NULL
33	26	NULL	NULL
34	401	NULL	NULL
35	524	NULL	NULL
36	132	NULL	NULL
37	650	NULL	NULL
38	212	NULL	NULL
39	667	NULL	NULL
40	275	NULL	NULL
41	86	NULL	NULL
42	604	NULL	NULL
43	149	NULL	NULL
44	23	NULL	NULL

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (89) | AdventureWorks2017 | 00:00:00 | 701 rows

Sample JSON Output with total number of rows returned(701)

```
USE AdventureWorks2017;
SELECT C.CustomerID,
       S.SalesOrderID,
```

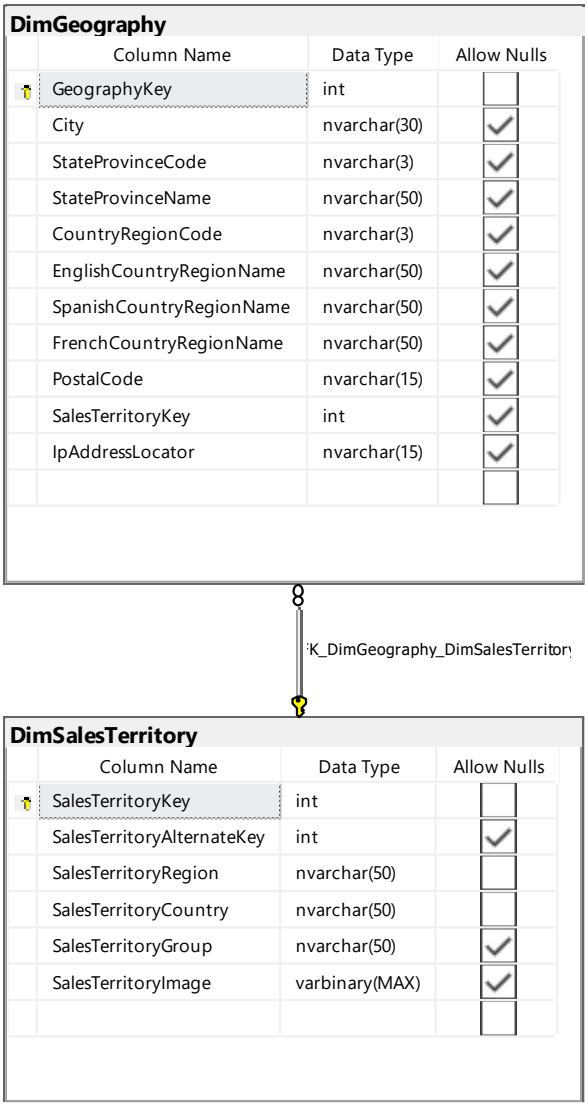
```
       S.OrderDate
FROM Sales.Customer AS C
    LEFT OUTER JOIN Sales.SalesOrderHeader AS S
        ON C.CustomerID = S.CustomerID
WHERE S.SalesOrderID IS NULL
FOR JSON PATH, ROOT('NoOrders'), INCLUDE_NULL_VALUES;
```



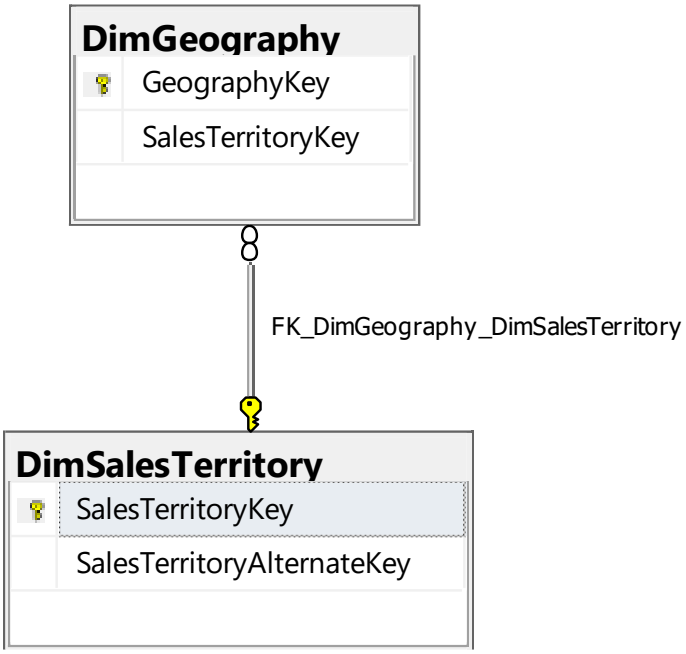
Proposition 03: Find all sales territories that have an IP Address beginning with “192”

Explanation: Find locations based on sales territories that have ip addresses beginning with '192' and include the locations country, city, state/province code, state/province name, country/region code and postal code

Columns From Standard view



Columns From Key View



Columns from their respective table in the select clause

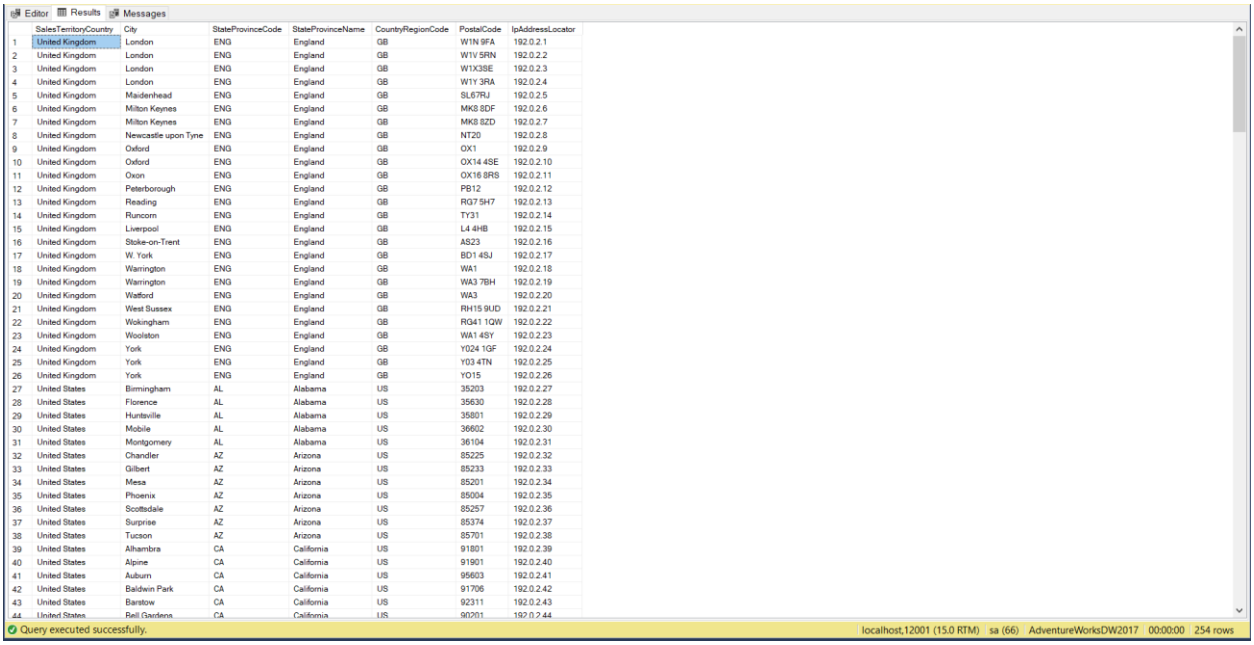
Table Name	Column Name
DimSalesTerritory	SalesTerritoryCountry
DimGeography	City StateProvinceCode StateProvinceName CountryRegionCode PostalCode IpAddressLocator

Problem Solving Query

```
USE AdventureWorksDW2017;
SELECT T.SalesTerritoryCountry,
       G.City,
```

```
       G.StateProvinceCode,
       G.StateProvinceName,
       G.CountryRegionCode,
       G.PostalCode,
       G.IpAddressLocator
FROM dbo.DimSalesTerritory AS T
    INNER JOIN dbo.DimGeography AS G
        ON G.SalesTerritoryKey = T.SalesTerritoryKey
WHERE G.IpAddressLocator LIKE '192%';
```

Sample Relational Output with total number of rows returned (254)



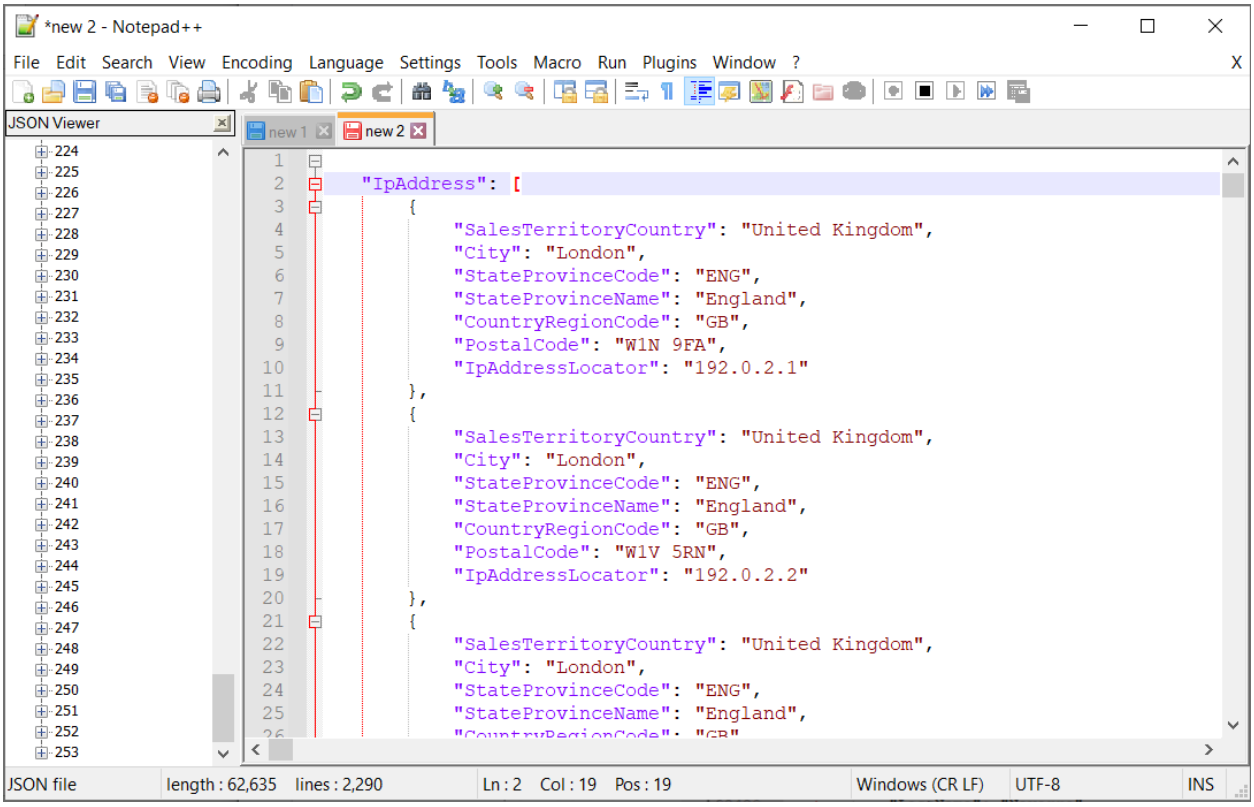
Editor | Results | Messages

	SalesTerritoryCountry	City	StateProvinceCode	StateProvinceName	CountryRegionCode	PostalCode	IpAddressLocator
1	United Kingdom	London	ENG	England	GB	W1N 9FA	192.0.2.1
2	United Kingdom	London	ENG	England	GB	W1V 5RN	192.0.2.2
3	United Kingdom	London	ENG	England	GB	W1X3SE	192.0.2.3
4	United Kingdom	London	ENG	England	GB	W1Y 3RA	192.0.2.4
5	United Kingdom	Maidenhead	ENG	England	GB	SL67RJ	192.0.2.5
6	United Kingdom	Milton Keynes	ENG	England	GB	MK8 8DF	192.0.2.6
7	United Kingdom	Milton Keynes	ENG	England	GB	MK8 8ZD	192.0.2.7
8	United Kingdom	Newcastle upon Tyne	ENG	England	GB	NT20	192.0.2.8
9	United Kingdom	Oxford	ENG	England	GB	OX1	192.0.2.9
10	United Kingdom	Oxford	ENG	England	GB	OX14 4SE	192.0.2.10
11	United Kingdom	Oxon	ENG	England	GB	OX16 8RS	192.0.2.11
12	United Kingdom	Peterborough	ENG	England	GB	PB12	192.0.2.12
13	United Kingdom	Reading	ENG	England	GB	RG7 5H7	192.0.2.13
14	United Kingdom	Runcorn	ENG	England	GB	TY31	192.0.2.14
15	United Kingdom	Liverpool	ENG	England	GB	L4 4HB	192.0.2.15
16	United Kingdom	Stoke-on-Trent	ENG	England	GB	AS23	192.0.2.16
17	United Kingdom	W. York	ENG	England	GB	BD1 4SJ	192.0.2.17
18	United Kingdom	Warrington	ENG	England	GB	WA1	192.0.2.18
19	United Kingdom	Warrington	ENG	England	GB	WA3 7BH	192.0.2.19
20	United Kingdom	Watford	ENG	England	GB	WA3	192.0.2.20
21	United Kingdom	West Sussex	ENG	England	GB	RH15 9UD	192.0.2.21
22	United Kingdom	Wokingham	ENG	England	GB	RG41 1QW	192.0.2.22
23	United Kingdom	Woolston	ENG	England	GB	WA1 4SY	192.0.2.23
24	United Kingdom	York	ENG	England	GB	Y024 1GF	192.0.2.24
25	United Kingdom	York	ENG	England	GB	Y03 4TN	192.0.2.25
26	United Kingdom	York	ENG	England	GB	YO15	192.0.2.26
27	United States	Birmingham	AL	Alabama	US	35203	192.0.2.27
28	United States	Florence	AL	Alabama	US	35630	192.0.2.28
29	United States	Huntsville	AL	Alabama	US	35801	192.0.2.29
30	United States	Mobile	AL	Alabama	US	36602	192.0.2.30
31	United States	Montgomery	AL	Alabama	US	36104	192.0.2.31
32	United States	Chandler	AZ	Arizona	US	85225	192.0.2.32
33	United States	Gilbert	AZ	Arizona	US	85233	192.0.2.33
34	United States	Mesa	AZ	Arizona	US	85201	192.0.2.34
35	United States	Phoenix	AZ	Arizona	US	85004	192.0.2.35
36	United States	Scottsdale	AZ	Arizona	US	85257	192.0.2.36
37	United States	Surprise	AZ	Arizona	US	85374	192.0.2.37
38	United States	Tucson	AZ	Arizona	US	85701	192.0.2.38
39	United States	Alhambra	CA	California	US	91801	192.0.2.39
40	United States	Alpine	CA	California	US	91901	192.0.2.40
41	United States	Auburn	CA	California	US	95603	192.0.2.41
42	United States	Baldwin Park	CA	California	US	91706	192.0.2.42
43	United States	Barstow	CA	California	US	92311	192.0.2.43
44	United States	Bell Gardens	CA	California	US	90201	192.0.2.44

Query executed successfully. localhost,12001 (15.0 RTM) | sa (66) | AdventureWorksDW2017 | 00:00:00 | 254 rows

Sample JSON Output with total number of rows (254)

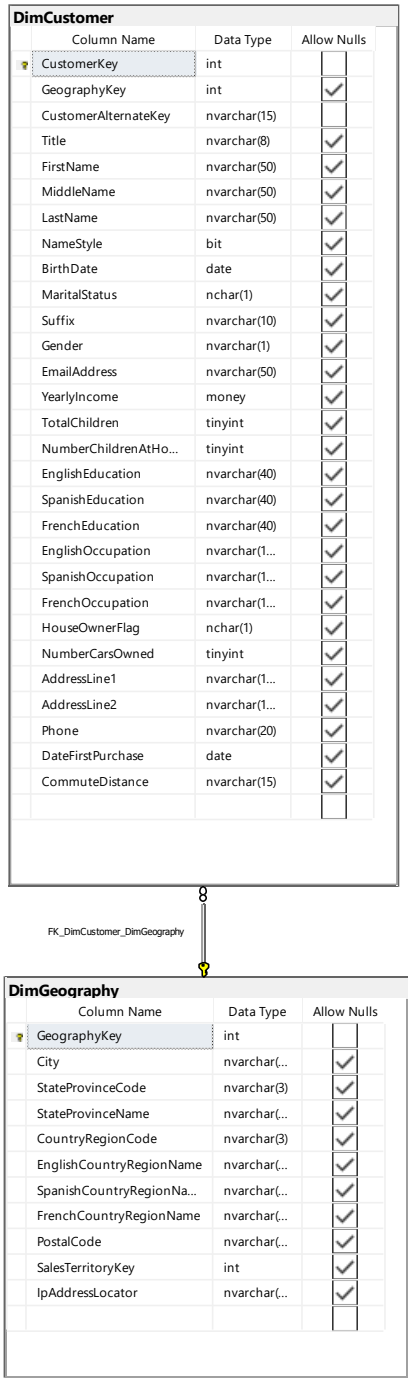
```
USE AdventureWorksDW2017;
SELECT T.SalesTerritoryCountry,
       G.City,
       G.StateProvinceCode,
       G.StateProvinceName,
       G.CountryRegionCode,
       G.PostalCode,
       G.IpAddressLocator
FROM dbo.DimSalesTerritory AS T
    INNER JOIN dbo.DimGeography AS G
        ON G.SalesTerritoryKey = T.SalesTerritoryKey
WHERE G.IpAddressLocator LIKE '192%'
FOR JSON PATH, ROOT('IpAddress'), INCLUDE_NULL_VALUES;
```



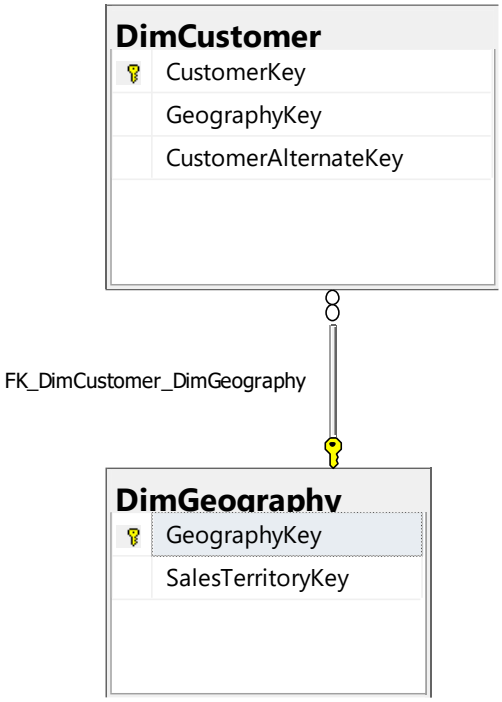
Proposition 04: Find all customers who placed their first purchase in 2014 and have a postal code that starts with a letter

Explanation: Find information about customers who placed their first purchase during the year of 2014, who have a postal code that starts with a letter and include the customerkey, geography key, date of first purchase, Email address, phone number, address, postal country and country

Columns from Standard View



Columns from Key View



Column from their respective table in the select clause

Table Name	Column Name
DimCustomer	CustomerKey GeographyKey FirstName LastName EmailAddress Phone AddressLine1
DimGeography	StateProvinceCode StateProvinceName CountryRegionCode PostalCode IpAddressLocator

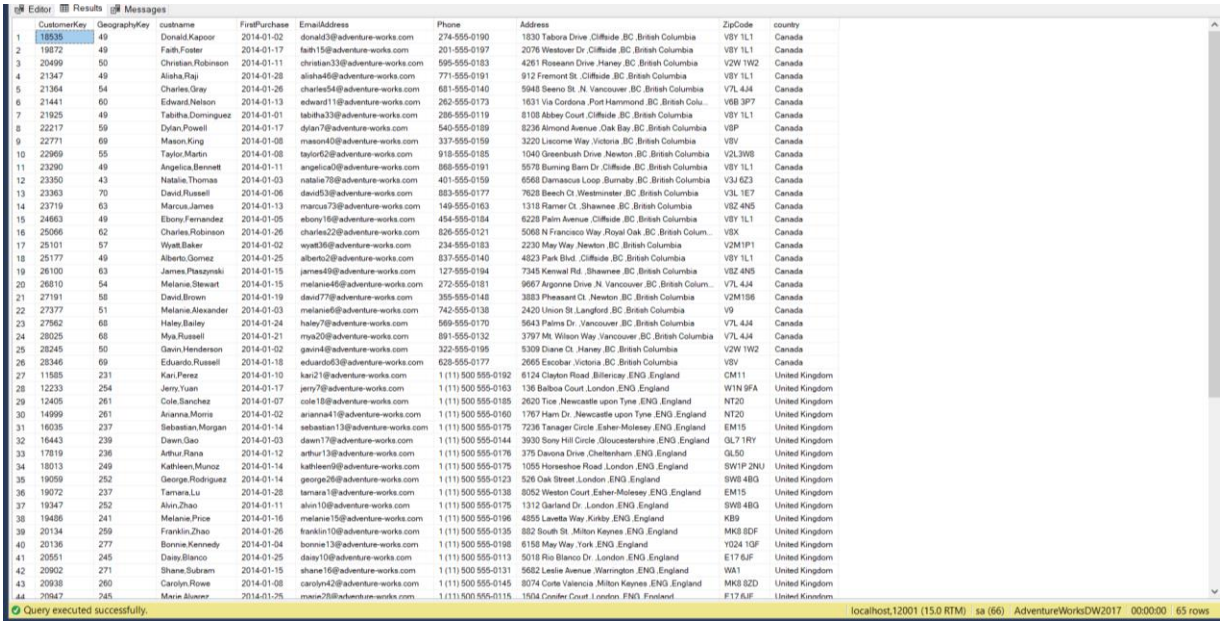
Order By

Table Name	Column Name	Sort
DimGeography	EnglishCountryRegionName	Default
DimCustomer	CustomerKey	Default

Problem Solving Query

```
USE AdventureWorksDW2017;
SELECT C.CustomerKey,
       C.GeographyKey,
       CONCAT_WS(',', C.FirstName, C.LastName) AS custname,
       (C.DateFirstPurchase) AS FirstPurchase,
       C.EmailAddress,
       C.Phone,
       CONCAT_WS(' ,', C.AddressLine1, G.City, G.StateProvinceCode, G.StateProvinceName)
AS Address,
       G.PostalCode AS ZipCode,
       G.EnglishCountryRegionName AS country
FROM dbo.DimCustomer AS C
    INNER JOIN dbo.DimGeography AS G
        ON G.GeographyKey = C.GeographyKey
WHERE YEAR(C.DateFirstPurchase) = 2014
      AND G.PostalCode LIKE '[A-Z]%'
ORDER BY G. EnglishCountryRegionName,
         C.CustomerKey;
```

Sample Relational output with total number of rows returned (65)



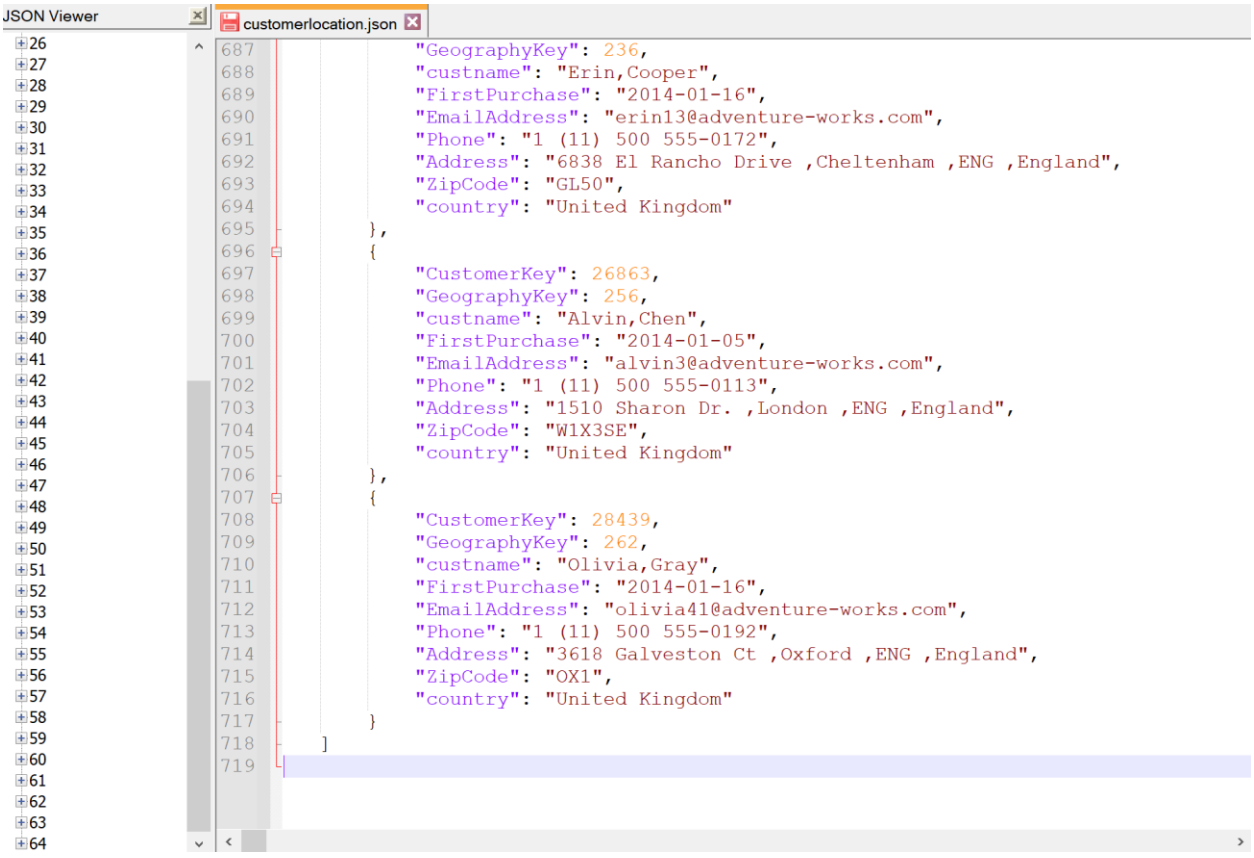
Editor Results Messages

	CustomerKey	GeographyKey	custname	FirstPurchase	EmailAddress	Phone	Address	ZipCode	country
1	18535	49	Donald,Kapoor	2014-01-02	donald3@adventure-works.com	274-555-0190	1830 Tabora Drive ,Cliffside ,BC ,British Columbia	V8Y 1L1	Canada
2	19872	49	Faith,Foster	2014-01-17	faith15@adventure-works.com	201-555-0197	2076 Westover Dr ,Cliffside ,BC ,British Columbia	V8Y 1L1	Canada
3	20499	50	Christian,Robinson	2014-01-11	christian33@adventure-works.com	595-555-0183	4261 Roseann Drive ,Haney ,BC ,British Columbia	V2W 1W2	Canada
4	21347	49	Alisha,Raji	2014-01-28	alisha46@adventure-works.com	771-555-0191	912 Fremont St ,Cliffside ,BC ,British Columbia	V8Y 1L1	Canada
5	21364	54	Charles,Gray	2014-01-26	charles54@adventure-works.com	681-555-0140	5948 Seeno St ,N. Vancouver ,BC ,British Columbia	V7L 4J4	Canada
6	21441	60	Edward,Nelson	2014-01-13	edward11@adventure-works.com	262-555-0173	1631 Via Cordona ,Port Hammond ,BC ,British Colu...	V6B 3P7	Canada
7	21925	49	Tabitha,Dominguez	2014-01-01	tabitha33@adventure-works.com	286-555-0119	8108 Abbey Court ,Cliffside ,BC ,British Columbia	V8Y 1L1	Canada
8	22217	59	Dylan,Powell	2014-01-17	dylan76@adventure-works.com	540-555-0189	8236 Almond Avenue ,Oak Bay ,BC ,British Columbia	V8P	Canada
9	22771	69	Mason,King	2014-01-08	mason40@adventure-works.com	337-555-0159	3220 Liscorne Way ,Victoria ,BC ,British Columbia	V8V	Canada
10	22969	55	Taylor,Martin	2014-01-08	taylor62@adventure-works.com	918-555-0185	1040 Greenbush Drive ,Newton ,BC ,British Columbia	V2L3W8	Canada
11	23290	49	Angelica,Bennett	2014-01-11	angelica0@adventure-works.com	868-555-0191	5578 Burning Barn Dr ,Cliffside ,BC ,British Columbia	V8Y 1L1	Canada
12	23350	43	Natalie,Thomas	2014-01-03	natalie78@adventure-works.com	401-555-0159	6568 Damascus Loop ,Burnaby ,BC ,British Columbia	V3J 6Z3	Canada
13	23363	70	David,Russell	2014-01-06	david53@adventure-works.com	883-555-0177	7628 Beech Ct ,Westminster ,BC ,British Columbia	V3L 1E7	Canada
14	23719	63	Marcus,James	2014-01-13	marcus73@adventure-works.com	149-555-0163	1318 Ramer Ct ,Shawnee ,BC ,British Columbia	V8Z 4N5	Canada
15	24663	49	Ebony,Fernandez	2014-01-05	ebony16@adventure-works.com	454-555-0184	6228 Palm Avenue ,Cliffside ,BC ,British Columbia	V8Y 1L1	Canada
16	25066	62	Charles,Robinson	2014-01-26	charles22@adventure-works.com	826-555-0121	5068 N Francisco Way ,Royal Oak ,BC ,British Colum...	V8X	Canada
17	25101	57	Wyatt,Baker	2014-01-02	wyatt36@adventure-works.com	234-555-0183	2230 May Way ,Newton ,BC ,British Columbia	V2M1P1	Canada
18	25177	49	Alberto,Gomez	2014-01-25	alberto2@adventure-works.com	837-555-0140	4823 Park Blvd. ,Cliffside ,BC ,British Columbia	V8Y 1L1	Canada
19	26100	63	James,Ptaszynski	2014-01-15	james49@adventure-works.com	127-555-0194	7345 Kenwal Rd. ,Shawnee ,BC ,British Columbia	V8Z 4N5	Canada
20	26810	54	Melanie,Stewart	2014-01-15	melanie46@adventure-works.com	272-555-0181	9667 Argonne Drive ,N. Vancouver ,BC ,British Colum...	V7L 4J4	Canada
21	27191	58	David,Brown	2014-01-19	david77@adventure-works.com	355-555-0148	3883 Pheasant Ct ,Newton ,BC ,British Columbia	V2M1S6	Canada
22	27377	51	Melanie,Alexander	2014-01-03	melanie6@adventure-works.com	742-555-0138	2420 Union St ,Langford ,BC ,British Columbia	V9	Canada
23	27562	68	Haley,Bailey	2014-01-24	haley76@adventure-works.com	569-555-0170	5643 Palma Dr. ,Vancouver ,BC ,British Columbia	V7L 4J4	Canada
24	28025	68	Mya,Russell	2014-01-21	mya20@adventure-works.com	891-555-0132	3797 Mt. Wilson Way ,Vancouver ,BC ,British Columbia	V7L 4J4	Canada
25	28245	50	Gavin,Henderson	2014-01-02	gavin4@adventure-works.com	322-555-0195	5309 Diane Ct. ,Haney ,BC ,British Columbia	V2W 1W2	Canada
26	28346	69	Eduardo,Russell	2014-01-18	eduardo63@adventure-works.com	628-555-0177	2665 Escobar ,Victoria ,BC ,British Columbia	V8V	Canada
27	11585	231	Kari,Perez	2014-01-10	kari21@adventure-works.com	1 (11) 500 555-0192	6124 Clayton Road ,Billericay ,ENG ,England	CM11	United Kingdom
28	12233	254	Jerry,Yuan	2014-01-17	jerry7@adventure-works.com	1 (11) 500 555-0163	136 Balboa Court ,London ,ENG ,England	W1N 9FA	United Kingdom
29	12405	261	Cole,Sanchez	2014-01-07	cole18@adventure-works.com	1 (11) 500 555-0185	2620 Tice ,Newcastle upon Tyne ,ENG ,England	NT20	United Kingdom
30	14999	261	Arianna,Morris	2014-01-02	arianna41@adventure-works.com	1 (11) 500 555-0160	1767 Ham Dr. ,Newcastle upon Tyne ,ENG ,England	NT20	United Kingdom
31	16035	237	Sebastian,Morgan	2014-01-14	sebastian13@adventure-works.com	1 (11) 500 555-0175	7236 Tanager Circle ,Esher-Molesey ,ENG ,England	EM15	United Kingdom
32	16443	239	Dawn,Gao	2014-01-03	dawn17@adventure-works.com	1 (11) 500 555-0144	3930 Sony Hill Circle ,Gloucestershire ,ENG ,England	GL7 1RY	United Kingdom
33	17819	236	Arthur,Rana	2014-01-12	arthur13@adventure-works.com	1 (11) 500 555-0176	375 Davona Drive ,Cheltenham ,ENG ,England	GL50	United Kingdom
34	18013	249	Kathleen,Munoz	2014-01-14	kathleen9@adventure-works.com	1 (11) 500 555-0175	1055 Horseshoe Road ,London ,ENG ,England	SW1P 2NU	United Kingdom
35	19059	252	George,Rodriguez	2014-01-14	george26@adventure-works.com	1 (11) 500 555-0123	526 Oak Street ,London ,ENG ,England	SW8 4BG	United Kingdom
36	19072	237	Tamara,Lu	2014-01-28	tamara1@adventure-works.com	1 (11) 500 555-0138	8052 Weston Court ,Esher-Molesey ,ENG ,England	EM15	United Kingdom
37	19347	252	Alvin,Zhao	2014-01-11	alvin10@adventure-works.com	1 (11) 500 555-0175	1312 Garland Dr. ,London ,ENG ,England	SW8 4BG	United Kingdom
38	19486	241	Melanie,Price	2014-01-16	melanie15@adventure-works.com	1 (11) 500 555-0196	4855 Lavetta Way ,Kirkby ,ENG ,England	KB9	United Kingdom
39	20134	259	Franklin,Zhao	2014-01-26	franklin10@adventure-works.com	1 (11) 500 555-0135	882 South St. ,Milton Keynes ,ENG ,England	MK8 8DF	United Kingdom
40	20136	277	Bonnie,Kennedy	2014-01-04	bonnie13@adventure-works.com	1 (11) 500 555-0198	6158 May Way ,York ,ENG ,England	Y024 1QF	United Kingdom
41	20551	245	Daisy,Blanco	2014-01-25	daisy10@adventure-works.com	1 (11) 500 555-0113	5018 Rio Blanco Dr. ,London ,ENG ,England	E17 6JF	United Kingdom
42	20902	271	Shane,Subram	2014-01-15	shane16@adventure-works.com	1 (11) 500 555-0131	5682 Leslie Avenue ,Warrington ,ENG ,England	WA1	United Kingdom
43	20938	260	Carolyn,Rowe	2014-01-08	carolyn42@adventure-works.com	1 (11) 500 555-0145	8074 Corte Valencia ,Milton Keynes ,ENG ,England	MK8 8ZD	United Kingdom
44	20947	245	Marie,Alvarez	2014-01-25	marie28@adventure-works.com	1 (11) 500 555-0115	1504 Conifer Court ,London ,ENG ,England	E17 6JF	United Kingdom

Query executed successfully. localhost,12001 (15.0 RTM) sa (66) AdventureWorksDW2017 00:00:00 65 rows

Sample Json Output With Total number of rows returned (65)

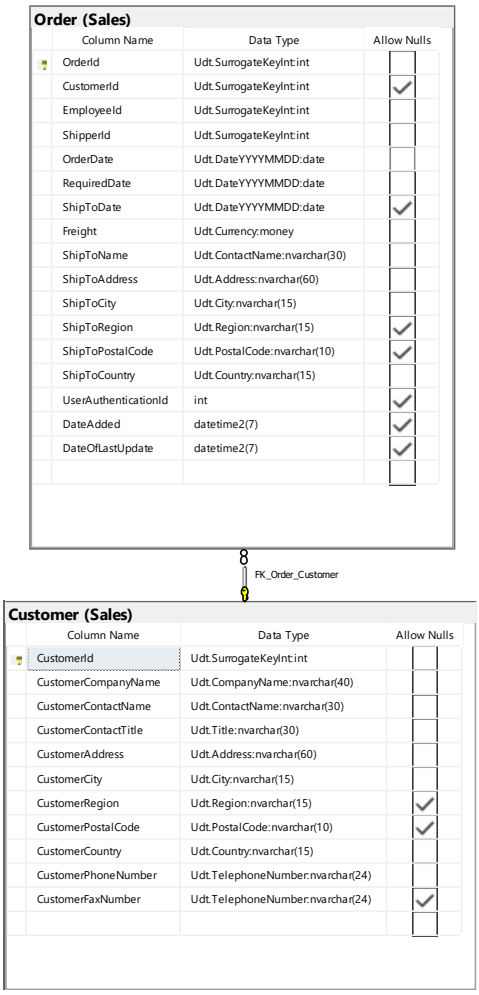
```
SELECT C.CustomerKey,
       C.GeographyKey,
       CONCAT_WS(',', C.FirstName, C.LastName) AS custname,
       (C.DateFirstPurchase) AS FirstPurchase,
       C.EmailAddress,
       C.Phone,
       CONCAT_WS(' ,', C.AddressLine1, G.City, G.StateProvinceCode, G.StateProvinceName)
AS Address,
       G.PostalCode AS ZipCode,
       G.EnglishCountryRegionName AS country
FROM dbo.DimCustomer AS C
    INNER JOIN dbo.DimGeography AS G
        ON G.GeographyKey = C.GeographyKey
WHERE YEAR(C.DateFirstPurchase) = 2014
      AND G.PostalCode LIKE '[A-Z]%'
ORDER BY G.EnglishCountryRegionName,
         C.CustomerKey
FOR JSON PATH,ROOT('CustomerLocation'),INCLUDE_NULL_VALUES;
```



Proposition 05: Figure out customer loyalty level

Explanation: Assign loyalty levels to customers based on the amount of orders they placed beginning with the most amount of orders being Platinum level then Gold, Silver and Bronzeas decreasing levels

Columns from Standard View



Order (Sales)

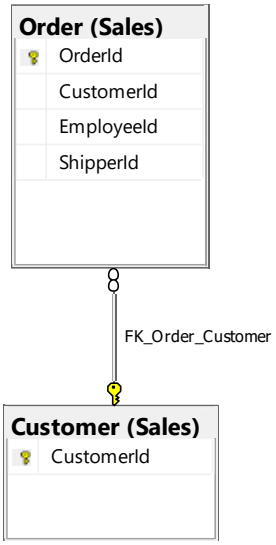
Column Name	Data Type	Allow Nulls
OrderId	Udt.SurrogateKeyInt:int	
CustomerId	Udt.SurrogateKeyInt:int	✓
EmployeeId	Udt.SurrogateKeyInt:int	
ShipperId	Udt.SurrogateKeyInt:int	
OrderDate	Udt.DateYYYYMMDD:date	
RequiredDate	Udt.DateYYYYMMDD:date	
ShipToDate	Udt.DateYYYYMMDD:date	✓
Freight	Udt.Currency:money	
ShipToName	Udt.ContactName:nvarchar(30)	
ShipToAddress	Udt.Address:nvarchar(60)	
ShipToCity	Udt.City:nvarchar(15)	
ShipToRegion	Udt.Region:nvarchar(15)	✓
ShipToPostalCode	Udt.PostalCode:nvarchar(10)	✓
ShipToCountry	Udt.Country:nvarchar(15)	
UserAuthenticationId	int	✓
DateAdded	datetime2(7)	✓
DateOfLastUpdate	datetime2(7)	✓

FK_Order_Customer

Customer (Sales)

Column Name	Data Type	Allow Nulls
CustomerId	Udt.SurrogateKeyInt:int	
CustomerCompanyName	Udt.CompanyName:nvarchar(40)	
CustomerContactName	Udt.ContactName:nvarchar(30)	
CustomerContactTitle	Udt.Title:nvarchar(30)	
CustomerAddress	Udt.Address:nvarchar(60)	
CustomerCity	Udt.City:nvarchar(15)	
CustomerRegion	Udt.Region:nvarchar(15)	✓
CustomerPostalCode	Udt.PostalCode:nvarchar(10)	✓
CustomerCountry	Udt.Country:nvarchar(15)	
CustomerPhoneNumber	Udt.TelephoneNumber:nvarchar(24)	
CustomerFaxNumber	Udt.TelephoneNumber:nvarchar(24)	✓

Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
Order	OrderId
Customer	CustomerId CustomerContactName
DerivedColumn	LoyaltyLevel

Order By

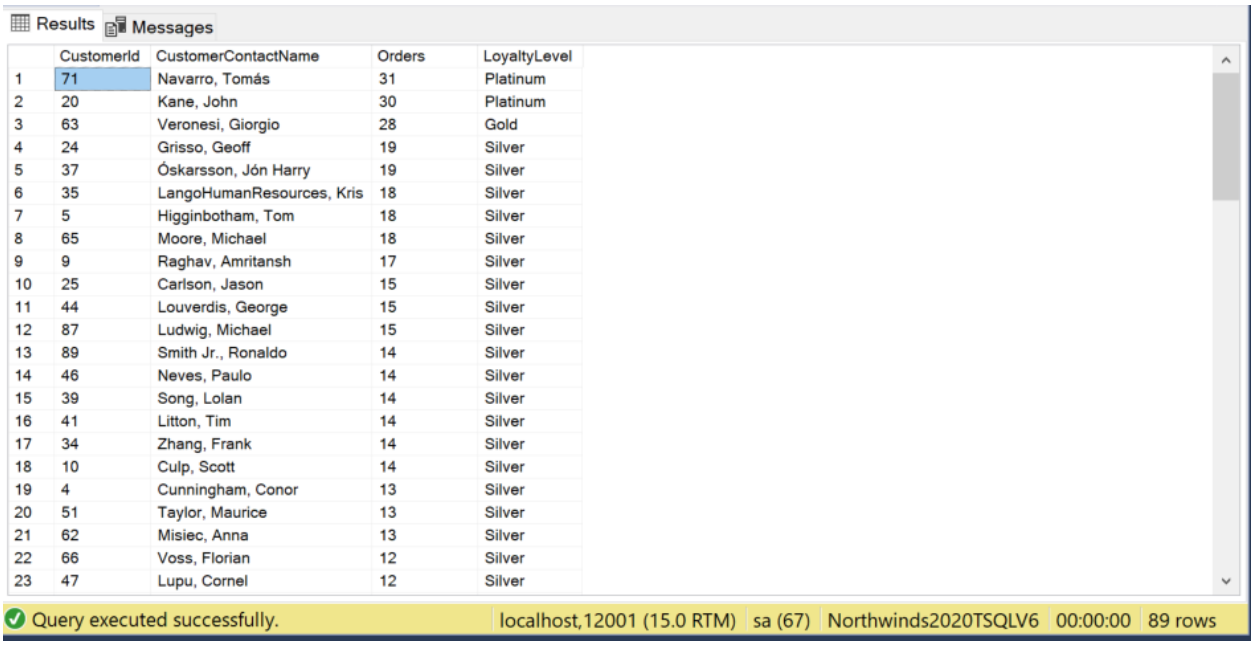
Table Name	Column Name	Sort Order
Order	OrderId	DESC

Problem Solving Query

```
USE Northwinds2020TSQLV6;
SELECT DISTINCT
       C.CustomerId,
       C.CustomerContactName,
       COUNT(O.OrderId) AS Orders,
       "LoyaltyLevel" = CASE
                        WHEN COUNT(O.OrderId) >= 30 THEN
                            'Platinum'
                        WHEN COUNT(O.OrderId) < 30
                             AND COUNT(O.OrderId) >= 20 THEN
                            'Gold'
                        WHEN COUNT(O.OrderId) < 20
                             AND COUNT(O.OrderId) >= 10 THEN
                            'Silver'
                        WHEN COUNT(O.OrderId) < 10 THEN
                            ' Bronze'
```

```
                        END
FROM Sales.[Order] AS O
    INNER JOIN Sales.Customer AS C
        ON C.CustomerId = O.CustomerId
GROUP BY C.CustomerId,
         C.CustomerContactName
ORDER BY Orders DESC;
--FOR JSON PATH, ROOT('Loyalty'), INCLUDE_NULL_VALUES;
```

Sample Relational output with total number of rows returned (89)



Results | Messages

	CustomerId	CustomerContactName	Orders	LoyaltyLevel
1	71	Navarro, Tomás	31	Platinum
2	20	Kane, John	30	Platinum
3	63	Veronesi, Giorgio	28	Gold
4	24	Grisso, Geoff	19	Silver
5	37	Óskarsson, Jón Harry	19	Silver
6	35	LangoHumanResources, Kris	18	Silver
7	5	Higginbotham, Tom	18	Silver
8	65	Moore, Michael	18	Silver
9	9	Raghav, Amritansh	17	Silver
10	25	Carlson, Jason	15	Silver
11	44	Louverdis, George	15	Silver
12	87	Ludwig, Michael	15	Silver
13	89	Smith Jr., Ronaldo	14	Silver
14	46	Neves, Paulo	14	Silver
15	39	Song, Lolan	14	Silver
16	41	Litton, Tim	14	Silver
17	34	Zhang, Frank	14	Silver
18	10	Culp, Scott	14	Silver
19	4	Cunningham, Conor	13	Silver
20	51	Taylor, Maurice	13	Silver
21	62	Misiec, Anna	13	Silver
22	66	Voss, Florian	12	Silver
23	47	Lupu, Cornel	12	Silver

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (67) | Northwinds2020TSQLV6 | 00:00:00 | 89 rows

Sample JSON output with total number of rows returned(89)

```
USE Northwinds2020TSQLV6;
SELECT DISTINCT
       C.CustomerId,
       C.CustomerContactName,
       COUNT(O.OrderId) AS Orders,
       "LoyaltyLevel" = CASE
                            WHEN COUNT(O.OrderId) >= 30 THEN
                                'Platinum'
                            WHEN COUNT(O.OrderId) < 30
                                 AND COUNT(O.OrderId) >= 20 THEN
                                'Gold'
                            WHEN COUNT(O.OrderId) < 20
                                 AND COUNT(O.OrderId) >= 10 THEN
                                'Silver'
                            WHEN COUNT(O.OrderId) < 10 THEN
                                ' Bronze'
                        END
FROM Sales.[Order] AS O
    INNER JOIN Sales.Customer AS C
        ON C.CustomerId = O.CustomerId
GROUP BY C.CustomerId,
         C.CustomerContactName
ORDER BY Orders DESC
```

```
FOR JSON PATH, ROOT('Loyalty'), INCLUDE_NULL_VALUES;
```

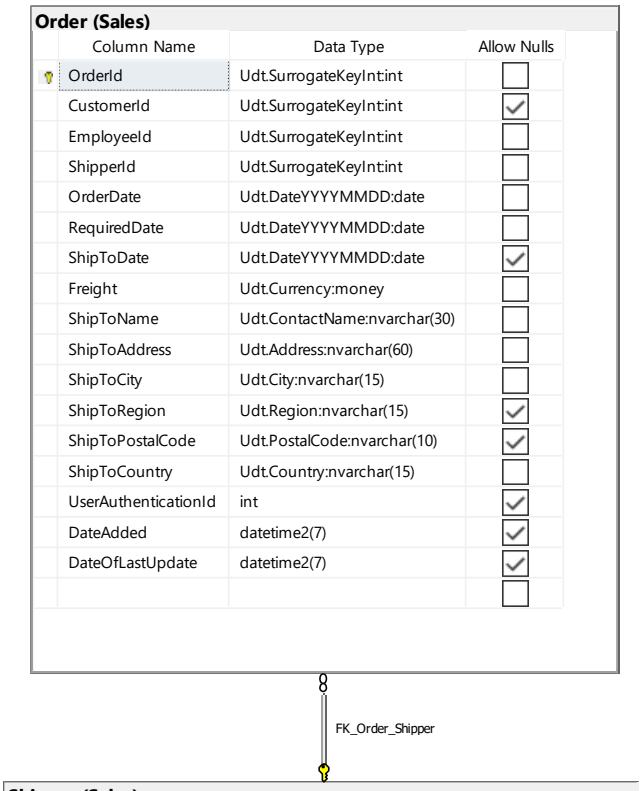
JSON Viewer

new 1

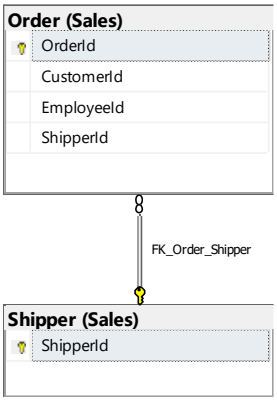

```
    "CustomerContactName": "Birkby, Dana",
    "Orders": 3,
    "LoyaltyLevel": " Bronze"
,
    "CustomerId": 26,
    "CustomerContactName": "Koch, Paul",
    "Orders": 3,
    "LoyaltyLevel": " Bronze"
,
    "CustomerId": 78,
    "CustomerContactName": "Ma, Andrew",
    "Orders": 3,
    "LoyaltyLevel": " Bronze"
,
    "CustomerId": 82,
    "CustomerContactName": "Veninga, Tjeerd",
    "Orders": 3,
    "LoyaltyLevel": " Bronze"
,
    "CustomerId": 33,
    "CustomerContactName": "Yuksel, Ayca",
    "Orders": 2,
    "LoyaltyLevel": " Bronze"
,
    "CustomerId": 43,
    "CustomerContactName": "Wu, Qiong",
    "Orders": 2,
    "LoyaltyLevel": " Bronze"
,
    "CustomerId": 13,
```

Proposition 06: Find the amount of orders shipped by each shipping company based on the country they shipped and limit the results to shipments of amounts greater than 10

Columns from Standard View



Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
Shipper	ShipperCompanyName
Order	ShipToCountry
DerivedColumn	AmtShippedToCountry

Order by

Table Name	Column Name	Sort
Order	ShipToCountry	DESC

Problem Solving Query

```
USE Northwinds2020TSQLV6;
SELECT S.ShipperCompanyName,
       COUNT(*) AS AmtShippedToCountry,
       O.ShipToCountry AS Countries
FROM Sales.Shipper AS S
    INNER JOIN Sales.[Order] AS O
        ON O.ShipperId = S.ShipperId
GROUP BY S.ShipperCompanyName,
         O.ShipToCountry
HAVING COUNT(*) > 10
ORDER BY AmtShippedToCountry DESC;
```

Sample Relational output with total number of rows returned (26)

Editor | Results | Messages

	ShipperCompanyName	AmtShippedToCountry	Countries
1	Shipper ETYNR	53	Germany
2	Shipper ETYNR	51	USA
3	Shipper GVSUA	41	Germany
4	Shipper ZHISN	40	USA
5	Shipper ETYNR	35	Brazil
6	Shipper GVSUA	31	Brazil
7	Shipper GVSUA	31	USA
8	Shipper ETYNR	29	France
9	Shipper ZHISN	28	Germany
10	Shipper GVSUA	27	France
11	Shipper ETYNR	23	UK
12	Shipper ZHISN	22	UK
13	Shipper ZHISN	21	France
14	Shipper ZHISN	17	Brazil
15	Shipper GVSUA	17	Venezuela
16	Shipper ETYNR	16	Venezuela
17	Shipper ZHISN	16	Canada
18	Shipper ETYNR	15	Austria
19	Shipper ETYNR	15	Sweden
20	Shipper GVSUA	14	Sweden
21	Shipper GVSUA	14	Italy
22	Shipper ZHISN	14	Mexico
23	Shipper ZHISN	13	Venezuela
24	Shipper ZHISN	13	Austria
25	Shipper GVSUA	12	Austria
26	Shipper GVSUA	11	UK

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (89) | Northwinds2020TSQLV6 | 00:00:00 | 26 rows

Sample JSON output with total number of rows returned(26)

```
USE Northwinds2020TSQLV6;
SELECT S.ShipperCompanyName,
       COUNT(*) AS AmtShippedToCountry,
       O.ShipToCountry AS Countries
```

```
FROM Sales.Shipper AS S
    INNER JOIN Sales.[Order] AS O
        ON O.ShipperId = S.ShipperId
GROUP BY S.ShipperCompanyName,
         O.ShipToCountry
HAVING COUNT(*) > 10
ORDER BY AmtShippedToCountry DESC
FOR JSON PATH, ROOT('TopShipperCountries'), INCLUDE_NULL_VALUES;
```

Proposition 07: Find all the top 10 percent products that had the longest deviation in time from the scheduled start date compared to the actual start date and the scheduled end date compared to the actual end date as well as the process in assembly where the deviation occurred.

Explanation: During the production phases of all products, which products were in the top 10 percent for having the longest difference in time between their scheduled start date and actual-start date and the longest difference in time between their scheduled end date and which sequence of production were they in

Columns from Standard View

Location (Production)

Column Name	Data Type	Allow Nulls
LocationID	smallint	
Name	Name:nvarchar(50)	
CostRate	smallmoney	
Availability	decimal(8, 2)	
ModifiedDate	datetime	

Product (Production)

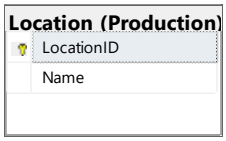
Column Name	Data Type	Allow Nul
ProductID	int	
Name	Name:nvarchar(50)	
ProductNumber	nvarchar(25)	
MakeFlag	Flag:bit	
FinishedGoodsFlag	Flag:bit	
Color	nvarchar(15)	✓
SafetyStockLevel	smallint	
ReorderPoint	smallint	
StandardCost	money	
ListPrice	money	
Size	nvarchar(5)	✓
SizeUnitMeasureCode	nchar(3)	✓
WeightUnitMeasureCode	nchar(3)	✓
Weight	decimal(8, 2)	✓
DaysToManufacture	int	
ProductLine	nchar(2)	✓
Class	nchar(2)	✓
Style	nchar(2)	✓
ProductSubcategoryID	int	✓
ProductModelID	int	✓
SellStartDate	datetime	
SellEndDate	datetime	✓
DiscontinuedDate	datetime	✓

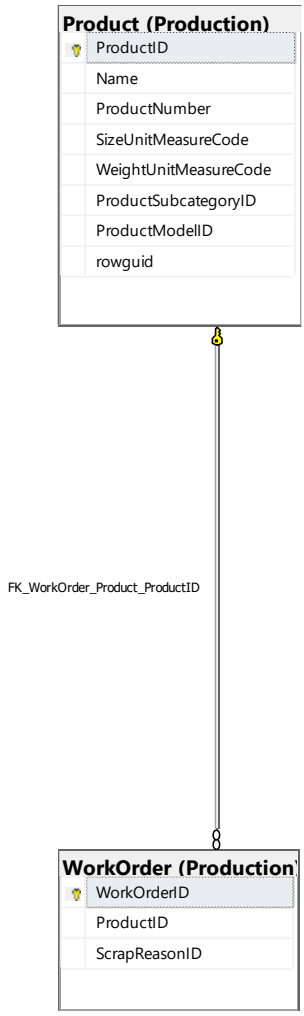
FK_WorkOrder_Product_ProductID

WorkOrder (Production)

Column Name	Data Type	Allow Nul
WorkOrderID	int	
ProductID	int	
OrderQty	int	
StockedQty		
ScrappedQty	smallint	
StartDate	datetime	
EndDate	datetime	✓
DueDate	datetime	
ScrapReasonID	smallint	✓

Columns from Key View





Columns from their respective table in the select clause

Table Name	Column Name
WorkOrderRouting	WorkOrderId OperationSequence ScheduledStartDate ActualStartDate

	ScheduledEndDate WOR.ActualEndDate)
Product	Name
Location	Name
DerivedColumn	ScheduledVsActualStartDate ScheduledVsActualEndDate

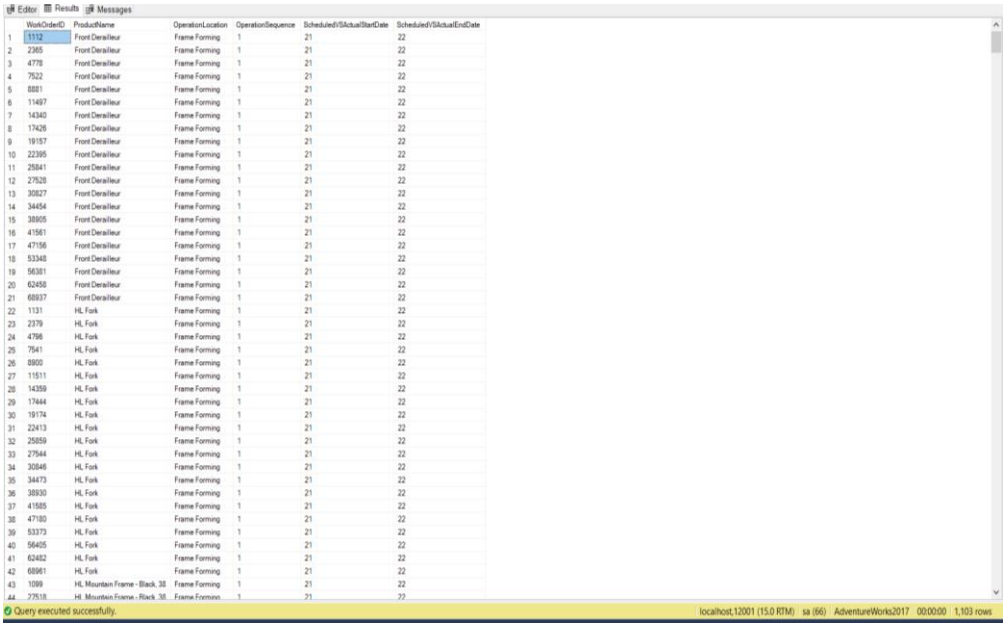
Order By

Table Name	Column Name	Sort
WorkOrderRouting	ScheduledVSActualEndDate OperationSequence	DESC
Product	Name	Default

Problem Solving Query

```
USE AdventureWorks2017;
SELECT TOP (10) PERCENT
       WOR.WorkOrderID,
       P.Name AS ProductName,
       L.Name AS OperationLocation,
       WOR.OperationSequence,
       DATEDIFF(dd, WOR.ScheduledStartDate, WOR.ActualStartDate) AS
ScheduledVSActualStartDate,
       DATEDIFF(dd, WOR.ScheduledEndDate, WOR.ActualEndDate) AS ScheduledVSActualEndDate
FROM Production.WorkOrderRouting AS WOR
    INNER JOIN Production.Location AS L
        ON L.LocationID = WOR.LocationID
    INNER JOIN Production.Product AS P
        ON P.ProductID = WOR.ProductID
WHERE DATEDIFF(dd, WOR.ScheduledStartDate, WOR.ActualStartDate) > 0
      AND DATEDIFF(dd, WOR.ScheduledEndDate, WOR.ActualEndDate) > 0
GROUP BY DATEDIFF(dd, WOR.ScheduledStartDate, WOR.ActualStartDate),
         DATEDIFF(dd, WOR.ScheduledEndDate, WOR.ActualEndDate),
         WOR.WorkOrderID,
         P.Name,
         L.Name,
         WOR.OperationSequence
ORDER BY ScheduledVSActualEndDate DESC,
         WOR.OperationSequence,
         P.Name;
```

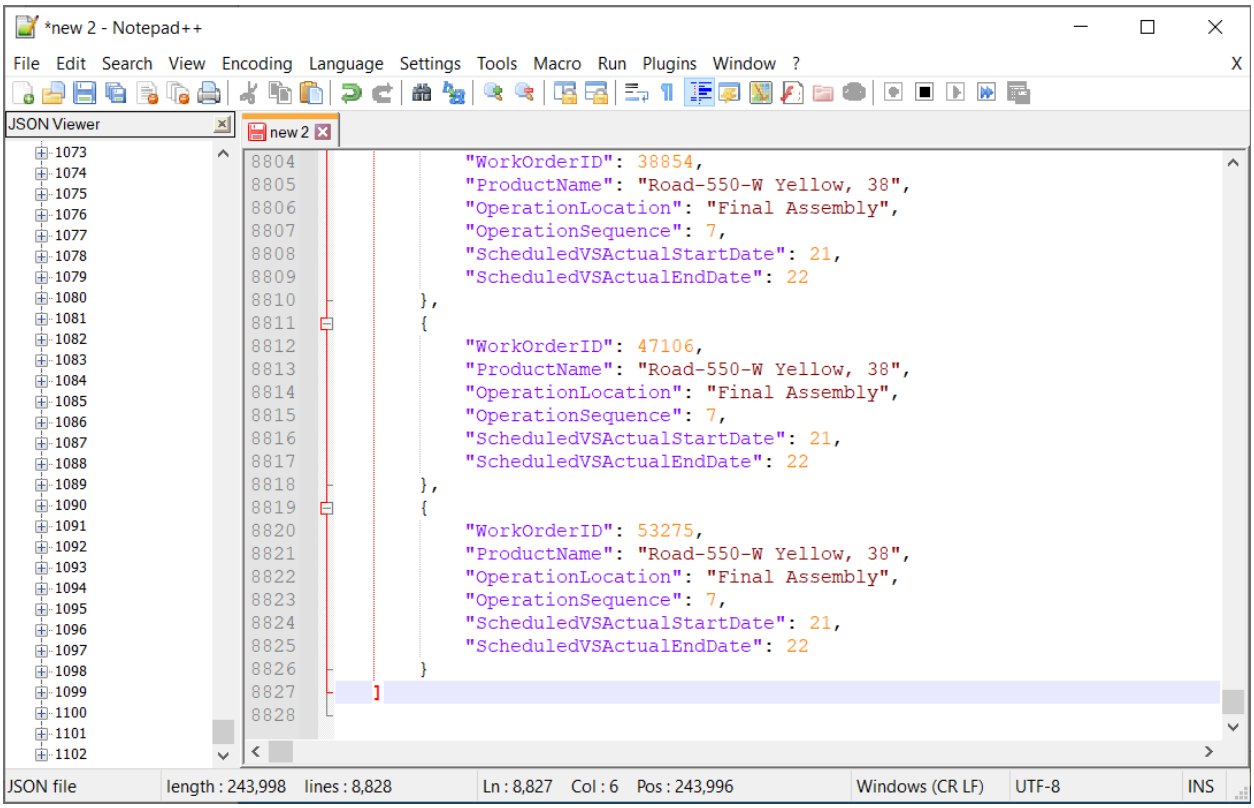
Sample Relational output with total number of rows returned(1103)



Sample JSON output with total number of rows returned(1103)

```
USE AdventureWorks2017;
SELECT TOP (10) PERCENT
       WOR.WorkOrderID,
       P.Name AS ProductName,
       L.Name AS OperationLocation,
       WOR.OperationSequence,
       DATEDIFF(dd, WOR.ScheduledStartDate, WOR.ActualStartDate) AS
ScheduledVSActualStartDate,
       DATEDIFF(dd, WOR.ScheduledEndDate, WOR.ActualEndDate) AS ScheduledVSActualEndDate
FROM Production.WorkOrderRouting AS WOR
    INNER JOIN Production.Location AS L
        ON L.LocationID = WOR.LocationID
    INNER JOIN Production.Product AS P
        ON P.ProductID = WOR.ProductID
WHERE DATEDIFF(dd, WOR.ScheduledStartDate, WOR.ActualStartDate) > 0
      AND DATEDIFF(dd, WOR.ScheduledEndDate, WOR.ActualEndDate) > 0
GROUP BY DATEDIFF(dd, WOR.ScheduledStartDate, WOR.ActualStartDate),
         DATEDIFF(dd, WOR.ScheduledEndDate, WOR.ActualEndDate),
         WOR.WorkOrderID,
         P.Name,
         L.Name,
         WOR.OperationSequence
ORDER BY ScheduledVSActualEndDate DESC,
         WOR.OperationSequence,
         P.Name
```

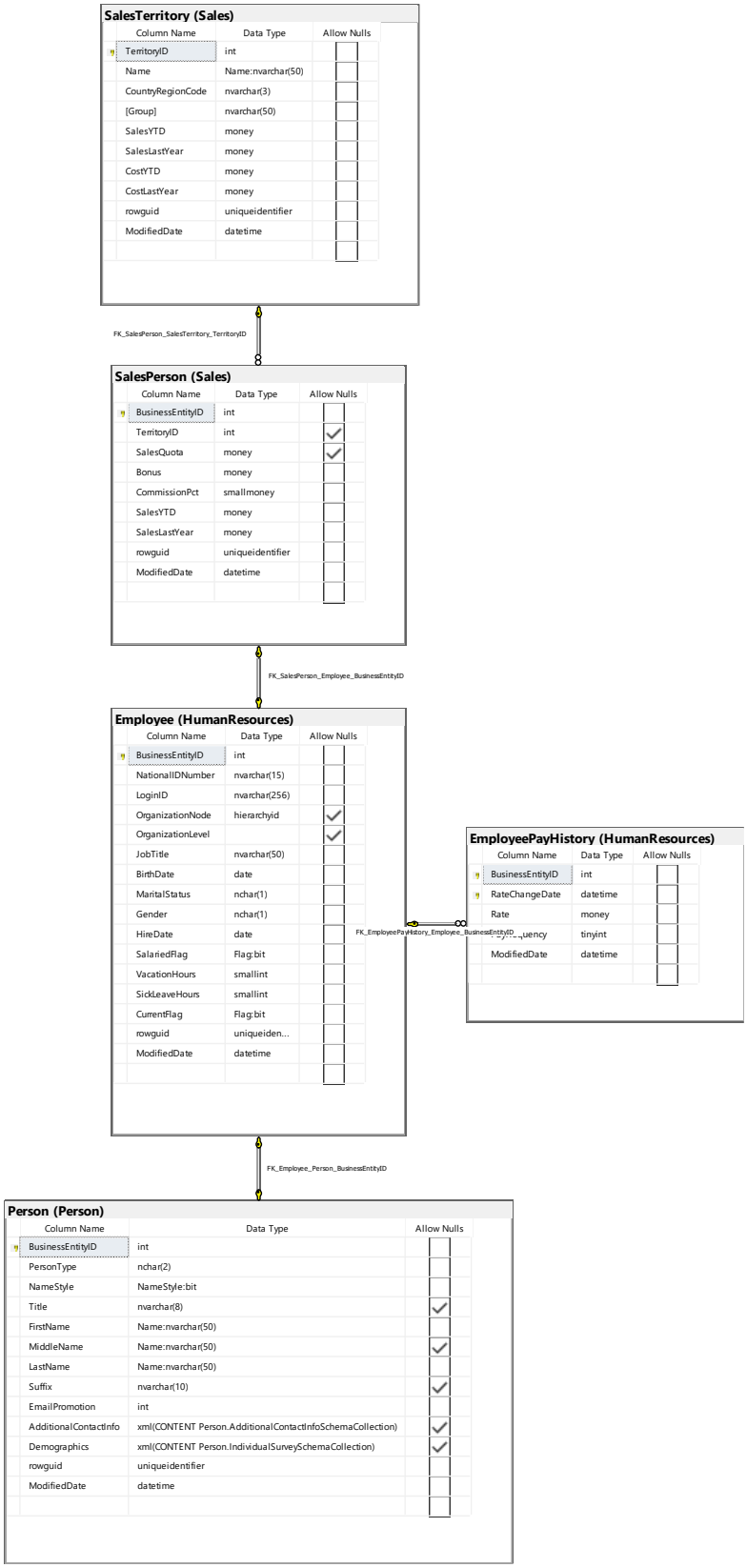
```
FOR JSON PATH, ROOT('ProductionTime'),INCLUDE_NULL_VALUES
```



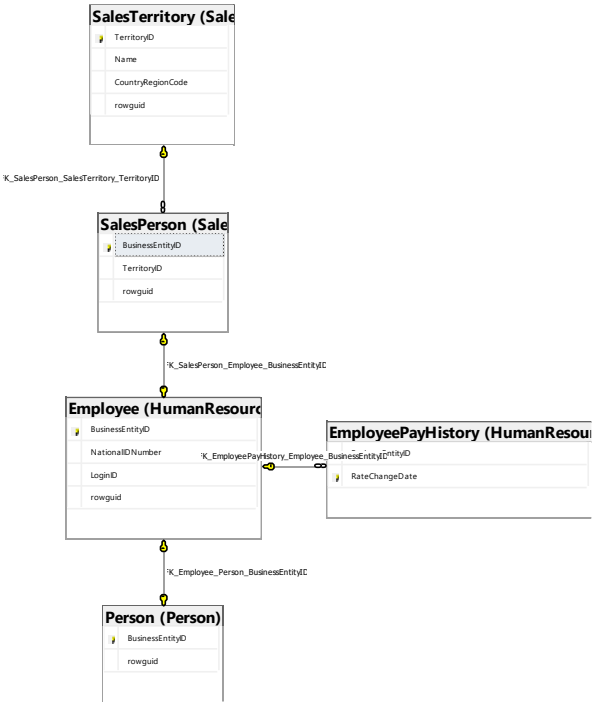
Proposition 08: Find the total salary for each employee based on their pay frequency, commission percentage, pay rate, and bonus as well as their sales location.

Explanation: Find all the sales representatives by their sales territory and the country return their base rate pay, pay frequency(whether they were payed biweekly or had a salary), and calculate their total salary based on their base rate pay, pay frequency, commission percent, and bonus

Columns from Standard View



Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
Employee	JobTitle
Territory	Name CountryRegionCode
Person	FirstName MiddleName LastName
EmployeePayHistory	Rate
SalesPerson	CommissionPct SalesYTD Bonus
DerivedColumn	PayFrequency(Case statement) TotalSalary

Order By

Table Name	Column Name	Sort
DerivedColumn	TotalSalary	DESC

Problem solving Query

```
USE AdventureWorks2017;
SELECT Territory.Name,
```

```
       Territory.CountryRegionCode,
       E.JobTitle,
       CONCAT_WS(', ', P.FirstName, P.MiddleName, P.LastName) AS SalesPersonName,
       pay.Rate,
       "PayFrequency" = CASE
                            WHEN pay.PayFrequency = 1 THEN
                                'Salary'
                            ELSE
                                'BiWeekly'
                        END,
       ((SP.CommissionPct * SP.SalesYTD) + (pay.Rate * 40 * 14) * 26 + SP.Bonus) AS
TotalSalary
FROM HumanResources.Employee AS E
    RIGHT OUTER JOIN Sales.SalesPerson AS SP
        ON E.BusinessEntityID = SP.BusinessEntityID
    INNER JOIN Person.Person AS P
        ON P.BusinessEntityID = E.BusinessEntityID
    INNER JOIN Sales.SalesTerritory AS Territory
        ON Territory.TerritoryID = SP.TerritoryID
    INNER JOIN HumanResources.EmployeePayHistory AS pay
        ON pay.BusinessEntityID = E.BusinessEntityID
GROUP BY CONCAT_WS(', ', P.FirstName, P.MiddleName, P.LastName),
         CASE
             WHEN pay.PayFrequency = 1 THEN
                 'Salary'
             ELSE
                 'BiWeekly'
         END,
         ((SP.CommissionPct * SP.SalesYTD) + (pay.Rate * 40 * 14) * 26 + SP.Bonus),
         Territory.Name,
         Territory.CountryRegionCode,
         E.JobTitle,
         pay.Rate
ORDER BY TotalSalary DESC;
```

Sample Relational output with total number of rows returned(14)

Editor | Results | Messages

	Name	CountryRegionCode	JobTitle	SalesPersonName	Rate	PayFrequency	TotalSalary
1	United Kingdom	GB	Sales Representative	Jae, B, Pak	23.0769	BiWeekly	423487.0886
2	Southwest	US	Sales Representative	Linda, C, Mitchell	23.0769	BiWeekly	401770.1922
3	France	FR	Sales Representative	Ranjit, R, Varkey Chudukatil	23.0769	BiWeekly	386930.5251
4	Central	US	Sales Representative	Jillian, Carson	23.0769	BiWeekly	386340.9395
5	Northeast	US	Sales Representative	Michael, G, Blythe	23.0769	BiWeekly	385257.8021
6	Canada	CA	Sales Representative	José, Edvaldo, Saraiva	23.0769	BiWeekly	380067.7748
7	Northwest	US	Sales Representative	Tete, A, Mensa-Annan	23.0769	BiWeekly	369854.3457
8	Germany	DE	Sales Representative	Rachel, B, Valdez	23.0769	BiWeekly	368961.8648
9	Australia	AU	Sales Representative	Lynn, N, Tsoflias	23.0769	BiWeekly	367242.2606
10	Southeast	US	Sales Representative	Tsvi, Michael, Reiter	23.0769	BiWeekly	365851.5201
11	Southwest	US	Sales Representative	Shu, K, Ito	23.0769	BiWeekly	364135.0202
12	Northwest	US	Sales Representative	David, R, Campbell	23.0769	BiWeekly	358375.8193
13	Northwest	US	Sales Representative	Pamela, O, Ansman-Wolfe	23.0769	BiWeekly	354525.4353
14	Canada	CA	Sales Representative	Garrett, R, Vargas	23.0769	BiWeekly	351036.8587

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (66) | AdventureWorks2017 | 00:00:00 | 14 rows

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

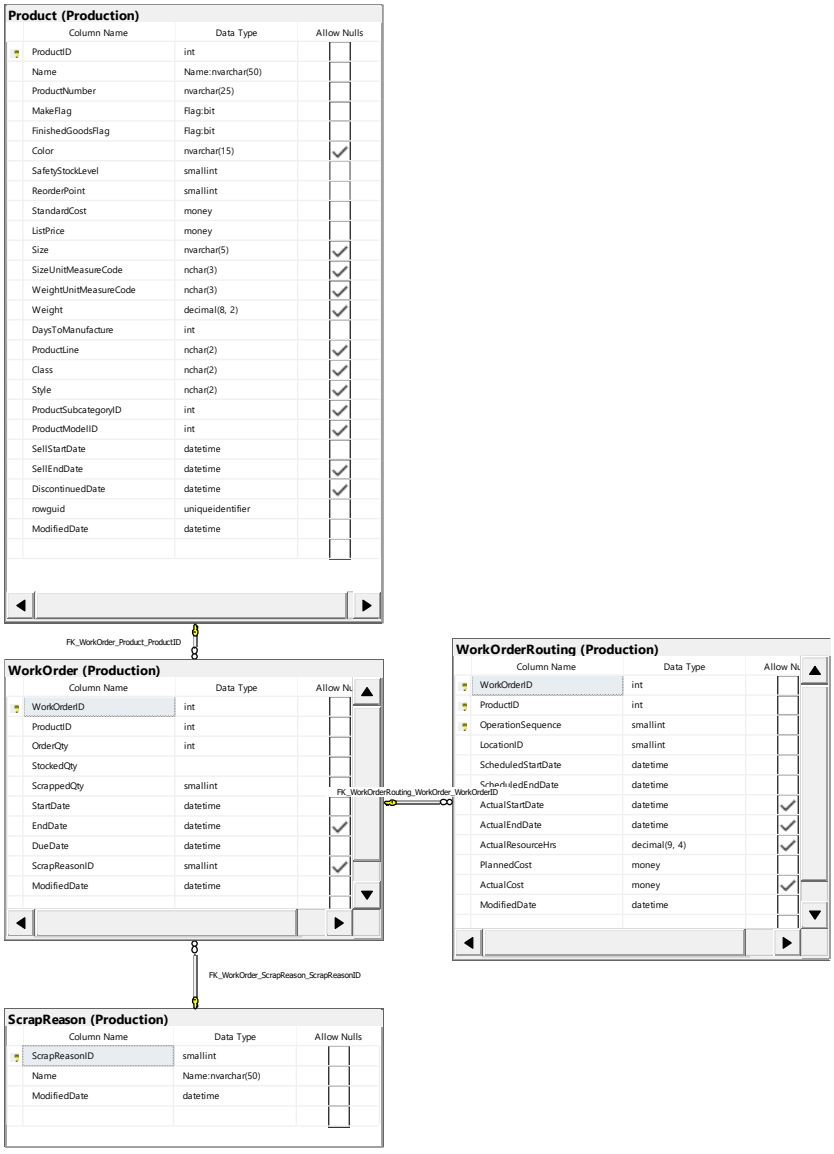
```
USE AdventureWorks2017;
SELECT Territory.Name,
       Territory.CountryRegionCode,
       E.JobTitle,
       CONCAT_WS(', ', P.FirstName, P.MiddleName, P.LastName) AS SalesPersonName,
       pay.Rate,
       "PayFrequency" = CASE
                            WHEN pay.PayFrequency = 1 THEN
```

```
                            'Salary'
                        ELSE
                            'BiWeekly'
                    END,
        ((SP.CommissionPct * SP.SalesYTD) + (pay.Rate * 40 * 14) * 26 + SP.Bonus) AS
TotalSalary
FROM HumanResources.Employee AS E
    RIGHT OUTER JOIN Sales.SalesPerson AS SP
        ON E.BusinessEntityID = SP.BusinessEntityID
    INNER JOIN Person.Person AS P
        ON P.BusinessEntityID = E.BusinessEntityID
    INNER JOIN Sales.SalesTerritory AS Territory
        ON Territory.TerritoryID = SP.TerritoryID
    INNER JOIN HumanResources.EmployeePayHistory AS pay
        ON pay.BusinessEntityID = E.BusinessEntityID
GROUP BY CONCAT_WS(', ', P.FirstName, P.MiddleName, P.LastName),
         CASE
             WHEN pay.PayFrequency = 1 THEN
                 'Salary'
             ELSE
                 'BiWeekly'
         END,
         ((SP.CommissionPct * SP.SalesYTD) + (pay.Rate * 40 * 14) * 26 + SP.Bonus),
         Territory.Name,
         Territory.CountryRegionCode,
         E.JobTitle,
         pay.Rate
ORDER BY TotalSalary DESC
FOR JSON PATH, ROOT('SalesRepSalary'),INCLUDE_NULL_VALUES;
```

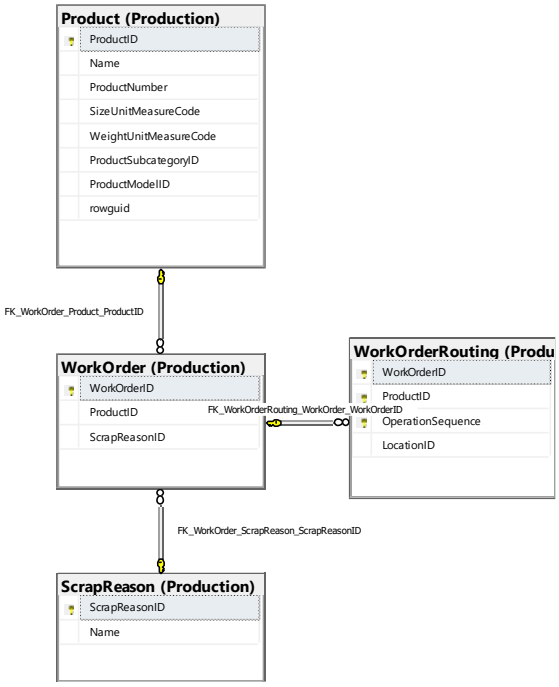
Proposition 09: Based on the year and month, which products were scrapped, what was the quantity of the products that were scrapped, how much was the total cost of scrapping those products and why were they scrapped?

Explanation: Based on the year and month, which products were scrapped, what was the quantity of the products that were scrapped, how much was the total cost of scrapping those products and why were they scrapped? Standard view

Columns from Standard View



Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
WorkOrder	EndDate ProductID WorkOrderID ScrappedQty
Product	Name ProductNumber
ScrapReason	Name
WorkOrderRouting	ActualCost
DerivedColumn	TotalScrapCost

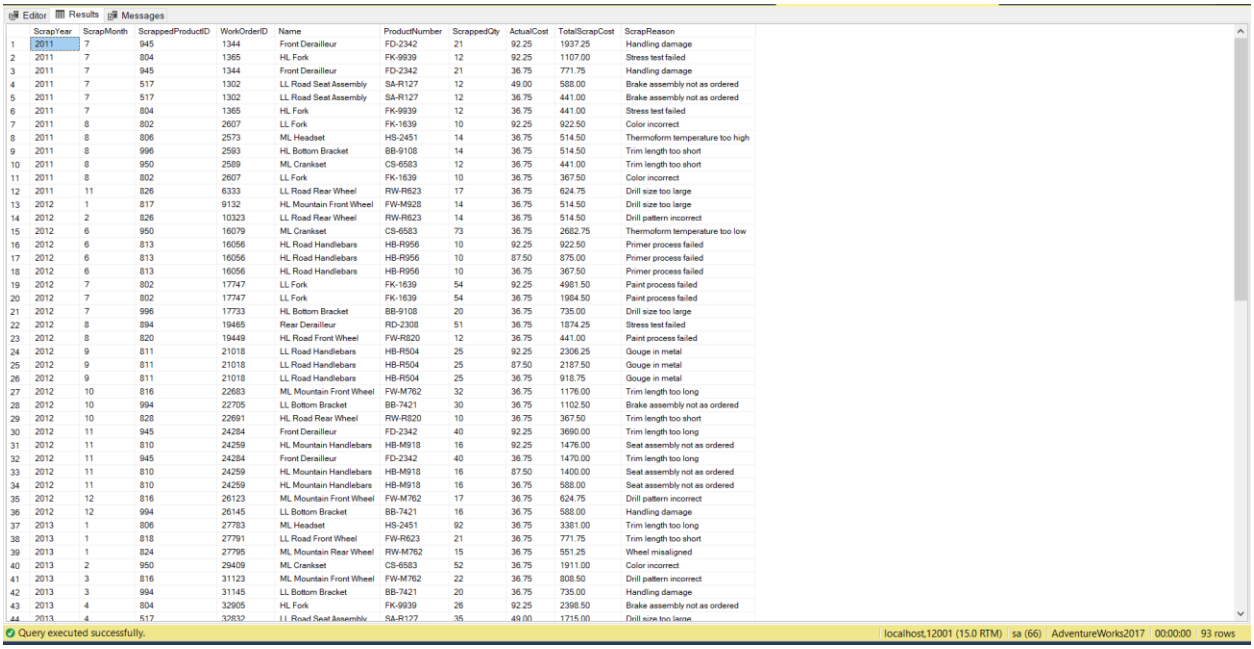
Order By

Table Name	Column Name	Order
WorkOrder	YEAR(EndDate) AS ScrapYear MONTH(EndDate) AS ScrapMonth	Default
WorkOrderRouting Workorder	TotalScrapCost	DESC

Problem Solving Query

```
USE AdventureWorks2017;
SELECT YEAR(WO.EndDate) AS ScrapYear,
       MONTH(WO.EndDate) AS ScrapMonth,
       WO.ProductID AS ScrappedProductID,
       WO.WorkOrderID,
       P.Name,
       P.ProductNumber,
       WO.ScrappedQty,
       WOR.ActualCost,
       (WO.ScrappedQty) * (WOR.ActualCost) AS TotalScrapCost,
       SR.Name AS ScrapReason
FROM Production.WorkOrder AS WO
    INNER JOIN Production.Product AS P
        ON P.ProductID = WO.ProductID
    INNER JOIN Production.ScrapReason AS SR
        ON SR.ScrapReasonID = WO.ScrapReasonID
    INNER JOIN Production.WorkOrderRouting AS WOR
        ON WOR.WorkOrderID = WO.WorkOrderID
WHERE WO.ScrappedQty >= 10
GROUP BY YEAR(WO.EndDate),
         MONTH(WO.EndDate),
         (WO.ScrappedQty) * (WOR.ActualCost),
         WO.ProductID,
         WO.WorkOrderID,
         P.Name,
         P.ProductNumber,
         WO.ScrappedQty,
         WOR.ActualCost,
         SR.Name
ORDER BY ScrapYear,
         ScrapMonth,
         TotalScrapCost DESC;
```

Sample relational out with rows returned (93)



Editor | Results | Messages

	ScrapYear	ScrapMonth	ScrappedProductID	WorkOrderID	Name	ProductNumber	ScrappedQty	ActualCost	TotalScrapCost	ScrapReason
1	2011	7	945	1344	Front Derailleur	FD-2342	21	92.25	1937.25	Handling damage
2	2011	7	804	1365	HL Fork	FK-9939	12	92.25	1107.00	Stress test failed
3	2011	7	945	1344	Front Derailleur	FD-2342	21	36.75	771.75	Handling damage
4	2011	7	517	1302	LL Road Seat Assembly	SA-R127	12	49.00	588.00	Brake assembly not as ordered
5	2011	7	517	1302	LL Road Seat Assembly	SA-R127	12	36.75	441.00	Brake assembly not as ordered
6	2011	7	804	1365	HL Fork	FK-9939	12	36.75	441.00	Stress test failed
7	2011	8	802	2607	LL Fork	FK-1639	10	92.25	922.50	Color incorrect
8	2011	8	806	2573	ML Headset	HS-2451	14	36.75	514.50	Thermoform temperature too high
9	2011	8	996	2593	HL Bottom Bracket	BB-9108	14	36.75	514.50	Trim length too short
10	2011	8	950	2589	ML Crankset	CS-6583	12	36.75	441.00	Trim length too short
11	2011	8	802	2607	LL Fork	FK-1639	10	36.75	367.50	Color incorrect
12	2011	11	826	6333	LL Road Rear Wheel	RW-R623	17	36.75	624.75	Drill size too large
13	2012	1	817	9132	HL Mountain Front Wheel	FW-M928	14	36.75	514.50	Drill size too large
14	2012	2	826	10323	LL Road Rear Wheel	RW-R623	14	36.75	514.50	Drill pattern incorrect
15	2012	6	950	16079	ML Crankset	CS-6583	73	36.75	2682.75	Thermoform temperature too low
16	2012	6	813	16056	HL Road Handlebars	HB-R956	10	92.25	922.50	Primer process failed
17	2012	6	813	16056	HL Road Handlebars	HB-R956	10	87.50	875.00	Primer process failed
18	2012	6	813	16056	HL Road Handlebars	HB-R956	10	36.75	367.50	Primer process failed
19	2012	7	802	17747	LL Fork	FK-1639	54	92.25	4981.50	Paint process failed
20	2012	7	802	17747	LL Fork	FK-1639	54	36.75	1984.50	Paint process failed
21	2012	7	996	17733	HL Bottom Bracket	BB-9108	20	36.75	735.00	Drill size too large
22	2012	8	894	19465	Rear Derailleur	RD-2308	51	36.75	1874.25	Stress test failed
23	2012	8	820	19449	HL Road Front Wheel	FW-R820	12	36.75	441.00	Paint process failed
24	2012	9	811	21018	LL Road Handlebars	HB-R504	25	92.25	2306.25	Gouge in metal
25	2012	9	811	21018	LL Road Handlebars	HB-R504	25	87.50	2187.50	Gouge in metal
26	2012	9	811	21018	LL Road Handlebars	HB-R504	25	36.75	918.75	Gouge in metal
27	2012	10	816	22683	ML Mountain Front Wheel	FW-M762	32	36.75	1176.00	Trim length too long
28	2012	10	994	22705	LL Bottom Bracket	BB-7421	30	36.75	1102.50	Brake assembly not as ordered
29	2012	10	828	22691	HL Road Rear Wheel	RW-R820	10	36.75	367.50	Trim length too short
30	2012	11	945	24284	Front Derailleur	FD-2342	40	92.25	3690.00	Trim length too long
31	2012	11	810	24259	HL Mountain Handlebars	HB-M918	16	92.25	1476.00	Seat assembly not as ordered
32	2012	11	945	24284	Front Derailleur	FD-2342	40	36.75	1470.00	Trim length too long
33	2012	11	810	24259	HL Mountain Handlebars	HB-M918	16	87.50	1400.00	Seat assembly not as ordered
34	2012	11	810	24259	HL Mountain Handlebars	HB-M918	16	36.75	588.00	Seat assembly not as ordered
35	2012	12	816	26123	ML Mountain Front Wheel	FW-M762	17	36.75	624.75	Drill pattern incorrect
36	2012	12	994	26145	LL Bottom Bracket	BB-7421	16	36.75	588.00	Handling damage
37	2013	1	806	27783	ML Headset	HS-2451	92	36.75	3381.00	Trim length too long
38	2013	1	818	27791	LL Road Front Wheel	FW-R623	21	36.75	771.75	Trim length too short
39	2013	1	824	27795	ML Mountain Rear Wheel	RW-M762	15	36.75	551.25	Wheel misaligned
40	2013	2	950	29409	ML Crankset	CS-6583	52	36.75	1911.00	Color incorrect
41	2013	3	816	31123	ML Mountain Front Wheel	FW-M762	22	36.75	808.50	Drill pattern incorrect
42	2013	3	994	31145	LL Bottom Bracket	BB-7421	20	36.75	735.00	Handling damage
43	2013	4	804	32905	HL Fork	FK-9939	26	92.25	2398.50	Brake assembly not as ordered
44	2013	4	517	32832	LL Road Seat Assembly	SA-R127	35	49.00	1715.00	Drill size too large

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (66) | AdventureWorks2017 | 00:00:00 | 93 rows

Sample JSON Output with total number of rows returned(93)

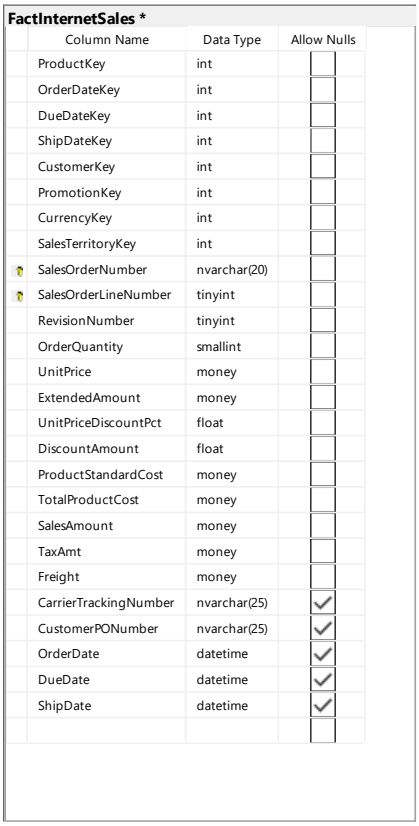
```
USE AdventureWorks2017;
SELECT YEAR(WO.EndDate) AS ScrapYear,
       MONTH(WO.EndDate) AS ScrapMonth,
       WO.ProductID AS ScrappedProductID,
       WO.WorkOrderID,
       P.Name,
       P.ProductNumber,
       WO.ScrappedQty,
       WOR.ActualCost,
       (WO.ScrappedQty) * (WOR.ActualCost) AS TotalScrapCost,
       SR.Name AS ScrapReason
FROM Production.WorkOrder AS WO
    INNER JOIN Production.Product AS P
        ON P.ProductID = WO.ProductID
    INNER JOIN Production.ScrapReason AS SR
        ON SR.ScrapReasonID = WO.ScrapReasonID
    INNER JOIN Production.WorkOrderRouting AS WOR
        ON WOR.WorkOrderID = WO.WorkOrderID
WHERE WO.ScrappedQty >= 10
GROUP BY YEAR(WO.EndDate),
         MONTH(WO.EndDate),
         (WO.ScrappedQty) * (WOR.ActualCost),
         WO.ProductID,
         WO.WorkOrderID,
         P.Name,
         P.ProductNumber,
         WO.ScrappedQty,
```

```
        WOR.ActualCost,
        SR.Name
ORDER BY ScrapYear,
        ScrapMonth,
        TotalScrapCost DESC
FOR JSON PATH, ROOT('TotalScrapCost'),INCLUDE_NULL_VALUES;
```

Proposition 10: What was the growth in territories for each consecutive year, beginning with the first order year, for internet sales?

Explanation: Determine the growth in sales territories for internet sales beginning with the first order year up until the last order year

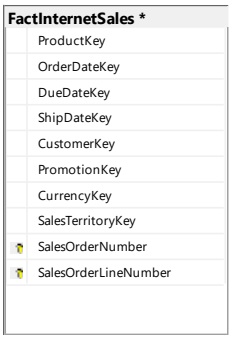
Columns from Standard View



FactInternetSales *

Column Name	Data Type	Allow Nulls
ProductKey	int	
OrderDateKey	int	
DueDateKey	int	
ShipDateKey	int	
CustomerKey	int	
PromotionKey	int	
CurrencyKey	int	
SalesTerritoryKey	int	
SalesOrderNumber	nvarchar(20)	
SalesOrderLineNumber	tinyint	
RevisionNumber	tinyint	
OrderQuantity	smallint	
UnitPrice	money	
ExtendedAmount	money	
UnitPriceDiscountPct	float	
DiscountAmount	float	
ProductStandardCost	money	
TotalProductCost	money	
SalesAmount	money	
TaxAmt	money	
Freight	money	
CarrierTrackingNumber	nvarchar(25)	✓
CustomerPONumber	nvarchar(25)	✓
OrderDate	datetime	✓
DueDate	datetime	✓
ShipDate	datetime	✓

Columns from Key View



FactInternetSales *

ProductKey
OrderDateKey
DueDateKey
ShipDateKey
CustomerKey
PromotionKey
CurrencyKey
SalesTerritoryKey
SalesOrderNumber
SalesOrderLineNumber

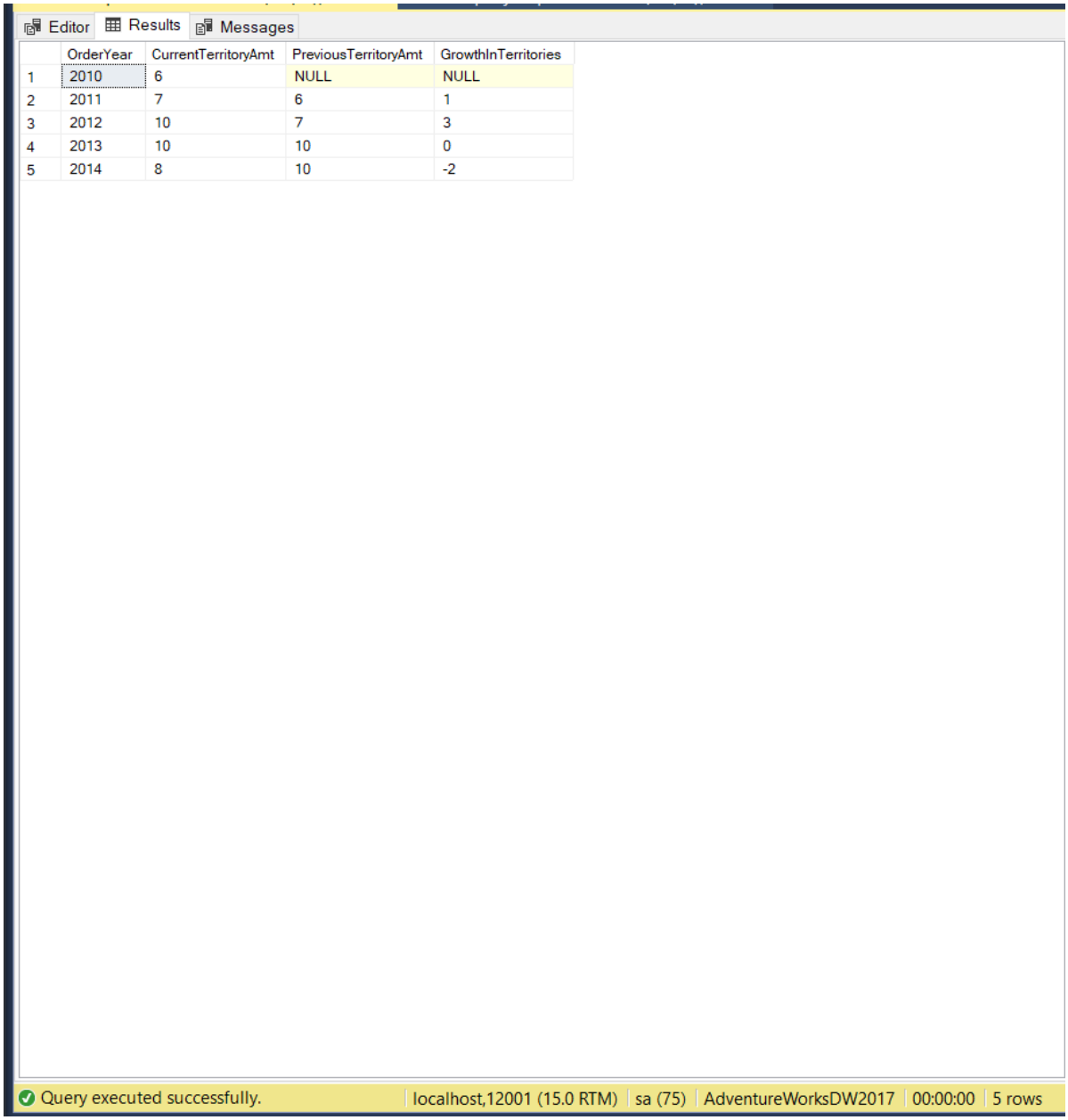
Columns from their respective table in the select clause

Table Name	Column Name
FactInternetSales	SalesTerritoryKey OrderDate
DerivedColumn	OrderYear TerritoryAmt GrowthInTerritories

Problem Solving Query

```
USE AdventureWorksDW2017;
SELECT C.OrderYear,
       C.TerritoryAmt AS CurrentTerritoryAmt,
       P.TerritoryAmt AS PreviousTerritoryAmt,
       C.TerritoryAmt - P.TerritoryAmt AS GrowthInTerritories
FROM
(
    SELECT DATEPART(yyyy, OrderDate) AS OrderYear,
           COUNT(DISTINCT SalesTerritoryKey) AS TerritoryAmt
    FROM dbo.FactInternetSales
    GROUP BY DATEPART(yyyy, OrderDate)
) AS C
    LEFT OUTER JOIN
    (
        SELECT DATEPART(yyyy, OrderDate) AS OrderYear,
               COUNT(DISTINCT SalesTerritoryKey) AS TerritoryAmt
        FROM dbo.FactInternetSales
        GROUP BY DATEPART(yyyy, OrderDate)
    ) AS P
        ON C.OrderYear = P.OrderYear + 1;
```

Sample Relational output with total number of rows returned(4)



	OrderYear	CurrentTerritoryAmt	PreviousTerritoryAmt	GrowthInTerritories
1	2010	6	NULL	NULL
2	2011	7	6	1
3	2012	10	7	3
4	2013	10	10	0
5	2014	8	10	-2

Sample JSON output with total number of rows returned(4)

```
USE AdventureWorksDW2017;

SELECT C.OrderYear,
       C.TerritoryAmt AS CurrentTerritoryAmt,
       P.TerritoryAmt AS PreviousTerritoryAmt,
       C.TerritoryAmt - P.TerritoryAmt AS GrowthInTerritories
FROM
(
    SELECT DATEPART(yyyy, OrderDate) AS OrderYear,
```

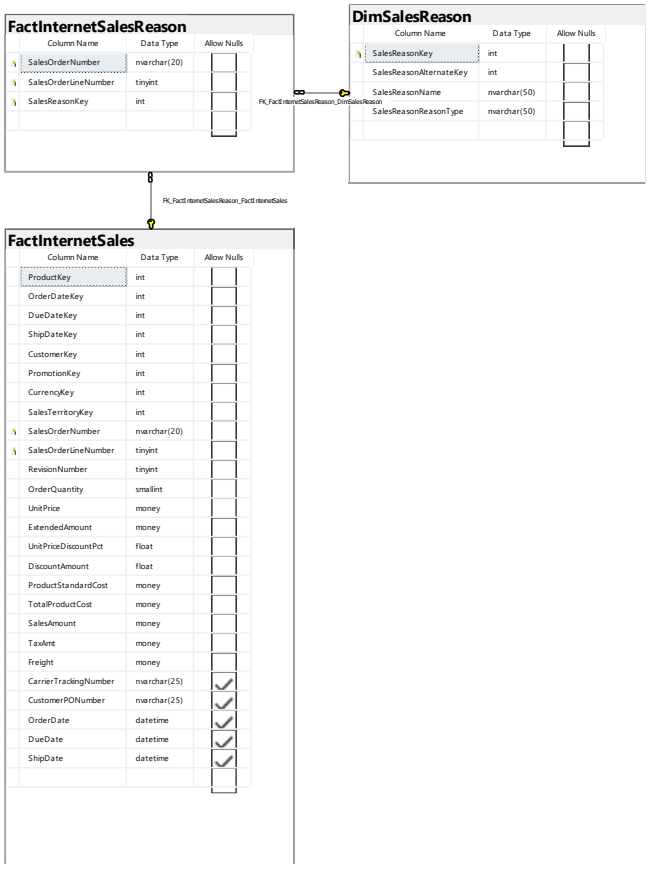
```
            COUNT(DISTINCT SalesTerritoryKey) AS TerritoryAmt
    FROM dbo.FactInternetSales
    GROUP BY DATEPART(yyyy, OrderDate)
) AS C
    LEFT OUTER JOIN
    (
        SELECT DATEPART(yyyy, OrderDate) AS OrderYear,
               COUNT(DISTINCT SalesTerritoryKey) AS TerritoryAmt
        FROM dbo.FactInternetSales
        GROUP BY DATEPART(yyyy, OrderDate)
    ) AS P
        ON C.OrderYear = P.OrderYear + 1
FOR JSON PATH, ROOT('TerritoryGrowth'),INCLUDE_NULL_VALUES;
```

JSON OUTPUT

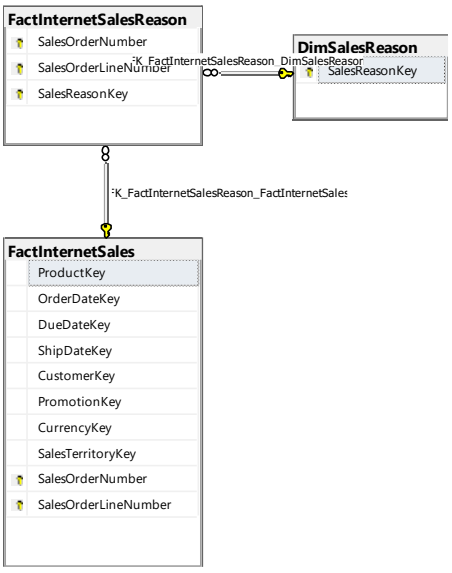
Proposition 11: Find the total amount of orders placed based upon the sales reason

Explanation: Explanation: Determine the total amount of orders placed based upon the reason of sale for internet sales with the results starting from the top reason

Columns from Standard View



Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
DimSalesReason	SalesReasonName

FactInternetSalesReason	SalesOrderLineNumber
DerivedColumn	CountOrders (SalesOrderLineNumber)Count

Order By

Table Name	Column Name	Sort
FactInternetSalesReason	Count(SalesOrderLineNumber) AS CountOrders	DESC

Problem Solving Query

```
USE AdventureWorksDW2017; SELECT DISTINCT
       SR.SalesReasonName,
       COUNT(FIS.SalesOrderLineNumber) AS CountOrders
FROM dbo.DimSalesReason AS SR
    INNER JOIN dbo.FactInternetSalesReason AS FISR
        ON FISR.SalesReasonKey = SR.SalesReasonKey
    INNER JOIN dbo.FactInternetSales AS FIS
        ON FIS.SalesOrderNumber = FISR.SalesOrderNumber
           AND FIS.SalesOrderLineNumber = FISR.SalesOrderLineNumber
WHERE FIS.SalesOrderLineNumber >= 1
GROUP BY SR.SalesReasonName
ORDER BY CountOrders DESC;
```

Sample relational output with total number of rows returned(7)

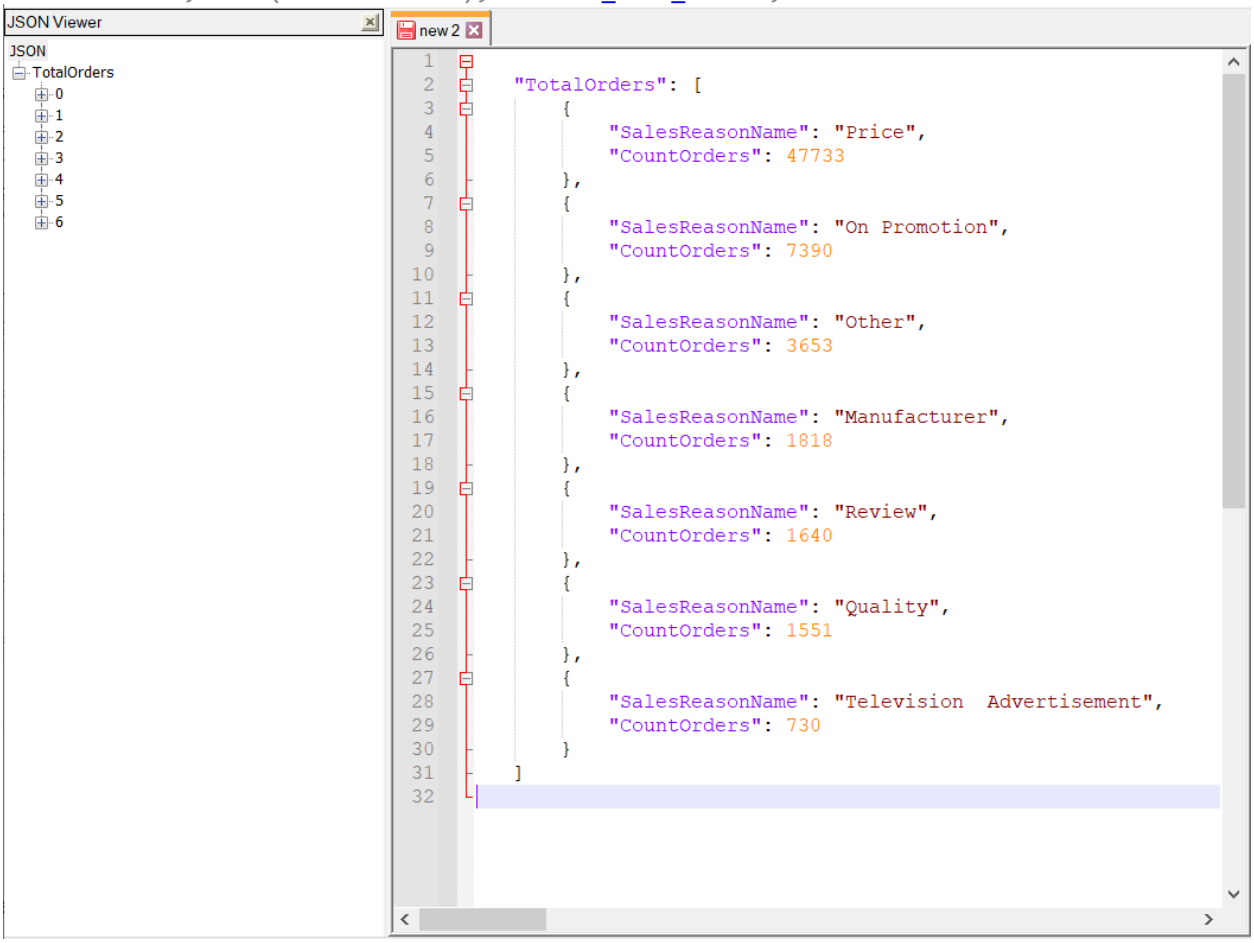
Editor | Results | Messages

	SalesReasonName	CountOrders
1	Price	47733
2	On Promotion	7390
3	Other	3653
4	Manufacturer	1818
5	Review	1640
6	Quality	1551
7	Television Advertisement	730

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (75) | AdventureWorksDW2017 | 00:00:00 | 7 rows

Sample JSON output with total number of rows returned(7)

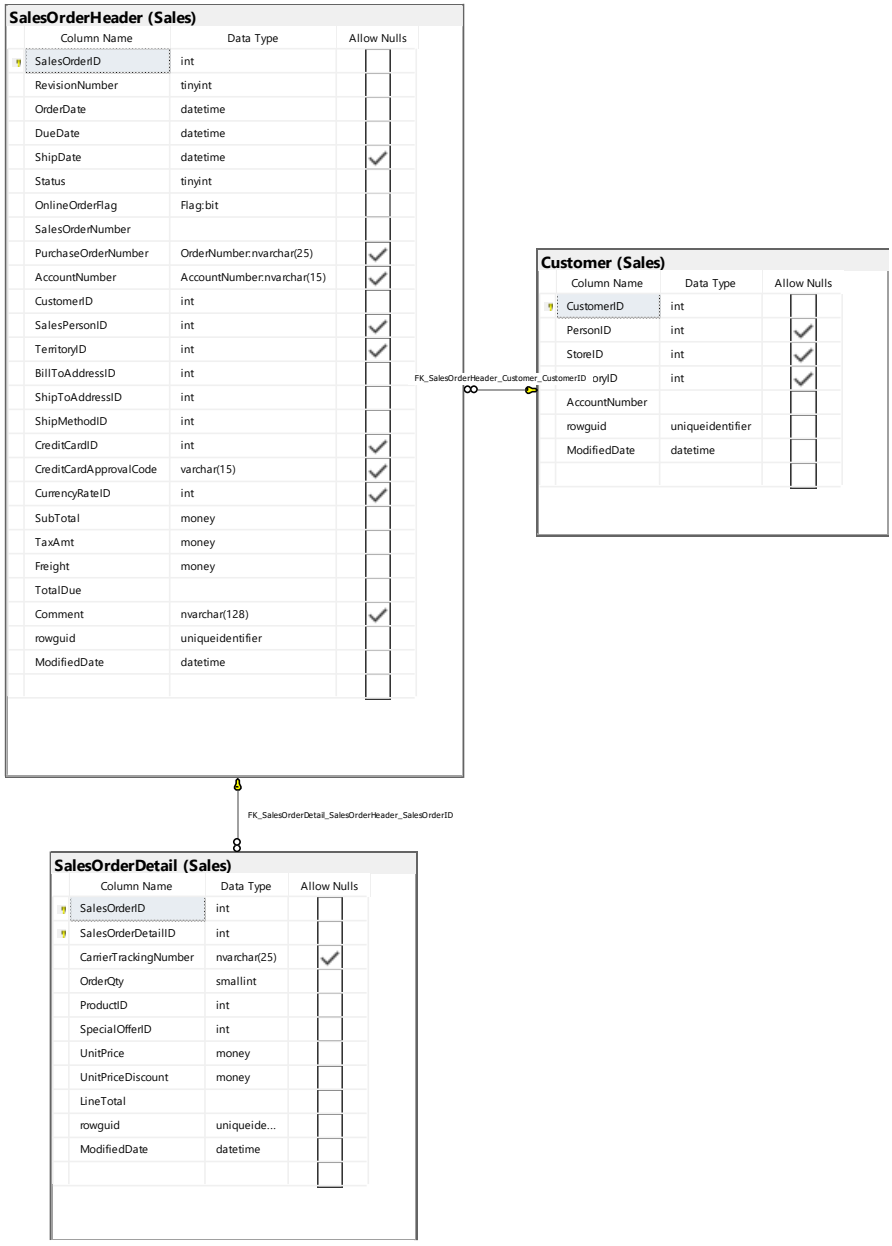
```
USE AdventureWorksDW2017;
SELECT DISTINCT
       SR.SalesReasonName,
       COUNT(FIS.SalesOrderLineNumber) AS CountOrders
FROM dbo.DimSalesReason AS SR
    INNER JOIN dbo.FactInternetSalesReason AS FISR
        ON FISR.SalesReasonKey = SR.SalesReasonKey
    INNER JOIN dbo.FactInternetSales AS FIS
        ON FIS.SalesOrderNumber = FISR.SalesOrderNumber
           AND FIS.SalesOrderLineNumber = FISR.SalesOrderLineNumber
WHERE FIS.SalesOrderLineNumber >= 1
GROUP BY SR.SalesReasonName
ORDER BY CountOrders DESC
FOR JSON PATH, ROOT('TotalOrders'), INCLUDE_NULL_VALUES;
```



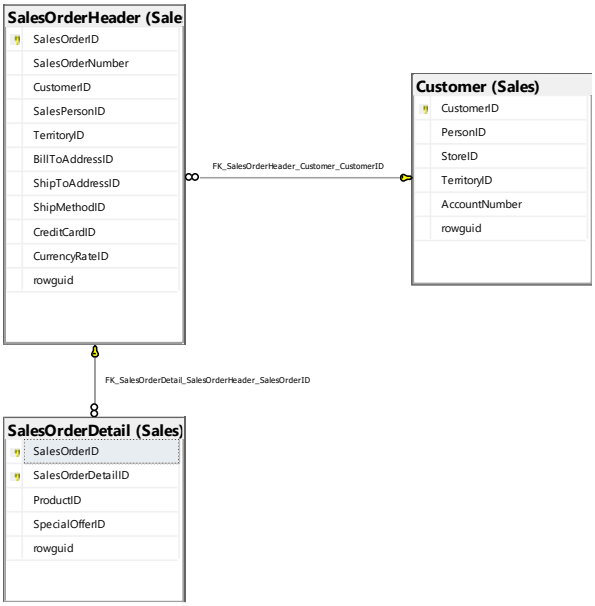
Proposition 12: Find the total value in sales for each territory

Explanation : Explanation: Calculate the total amount of revenue based on each territory according to the amount of orders placed, their unit price and discount if applicable

Columns from Standard View



Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
Customer	TerritoryID
SalesOrderDetail	OrderQty UnitPrice UnitPriceDiscount
DerivedColumn	Total

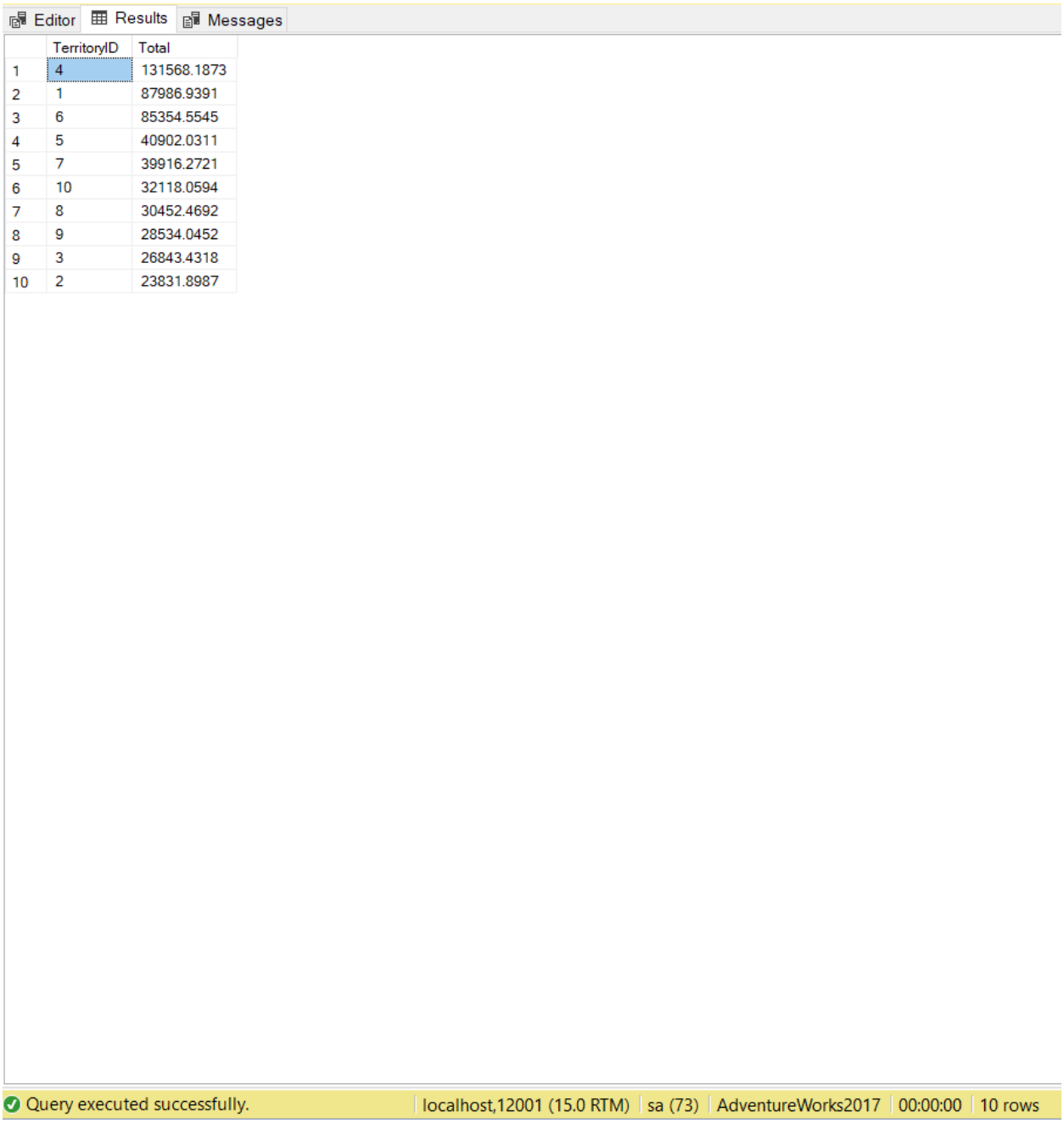
Order by

Table Name	Column Name	Sort
SalesOrderDetail	Total(OrderQty UnitPrice UnitPriceDiscount)	DESC

Problem Solving Query

```
USE AdventureWorks2017;
SELECT C.TerritoryID,
       SUM(SOD.OrderQty * SOD.UnitPrice) - SUM((SOD.OrderQty * SOD.UnitPrice) * (1 -
SOD.UnitPriceDiscount)) AS Total
FROM Sales.Customer AS C
    INNER JOIN Sales.SalesOrderHeader AS O
        ON C.CustomerID = O.CustomerID
    INNER JOIN Sales.SalesOrderDetail AS SOD
        ON O.SalesOrderID = SOD.SalesOrderID
GROUP BY C.TerritoryID
ORDER BY Total DESC;
```

Sample Relational Output with total number of rows returned(10)

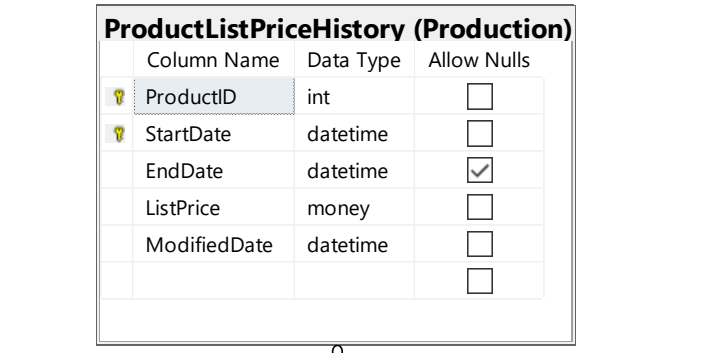


Sample Relational Output with total number of rows returned(10)

```
USE AdventureWorks2017;
SELECT C.TerritoryID,
       SUM(SOD.OrderQty * SOD.UnitPrice) - SUM((SOD.OrderQty * SOD.UnitPrice) * (1 -
SOD.UnitPriceDiscount)) AS Total
FROM Sales.Customer AS C
    INNER JOIN Sales.SalesOrderHeader AS O
        ON C.CustomerID = O.CustomerID
    INNER JOIN Sales.SalesOrderDetail AS SOD
        ON O.SalesOrderID = SOD.SalesOrderID
GROUP BY C.TerritoryID
ORDER BY Total DESC
FOR JSON PATH,ROOT('TotalSalesByTerritory'),INCLUDE_NULL_VALUES;
```

Proposition 13: Find the amount of orders that were placed based on the color of the product and the age of the customer to determine the preferences of product colors by age

Columns from Standard View



ProductListPriceHistory (Production)

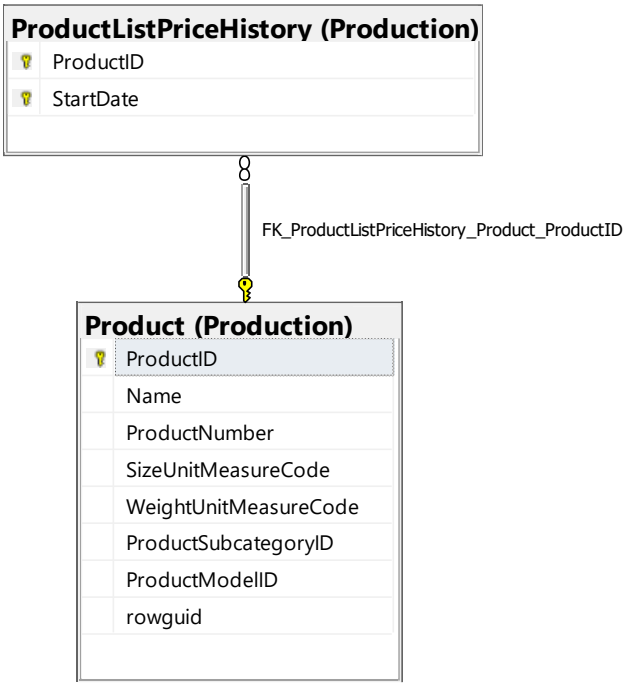
Column Name	Data Type	Allow Nulls
ProductID	int	☐
StartDate	datetime	☐
EndDate	datetime	☑
ListPrice	money	☐
ModifiedDate	datetime	☐
		☐

FK_ProductListPriceHistory_Product_ProductID

Product (Production)

Column Name	Data Type	Allow Nulls
ProductID	int	☐
Name	Name:nvarchar(50)	☐
ProductNumber	nvarchar(25)	☐
MakeFlag	Flag:bit	☐
FinishedGoodsFlag	Flag:bit	☐
Color	nvarchar(15)	☑
SafetyStockLevel	smallint	☐
ReorderPoint	smallint	☐
StandardCost	money	☐
ListPrice	money	☐
Size	nvarchar(5)	☑
SizeUnitMeasureCode	nchar(3)	☑
WeightUnitMeasureCode	nchar(3)	☑
Weight	decimal(8, 2)	☑
DaysToManufacture	int	☐
ProductLine	nchar(2)	☑
Class	nchar(2)	☑
Style	nchar(2)	☑
ProductSubcategoryID	int	☑
ProductModelID	int	☑
SellStartDate	datetime	☐
SellEndDate	datetime	☑
DiscontinuedDate	datetime	☑
rowguid	uniqueidentifier	☐
ModifiedDate	datetime	☐
		☐

Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
ProductionListPriceHistory	ListPrice
DerivedColumns	MinimumListPrice(MIN function) MaximumListPrice(MAX function) AverageListPrice(AVG function) MaxVsMinDiff(MAX – MIN)

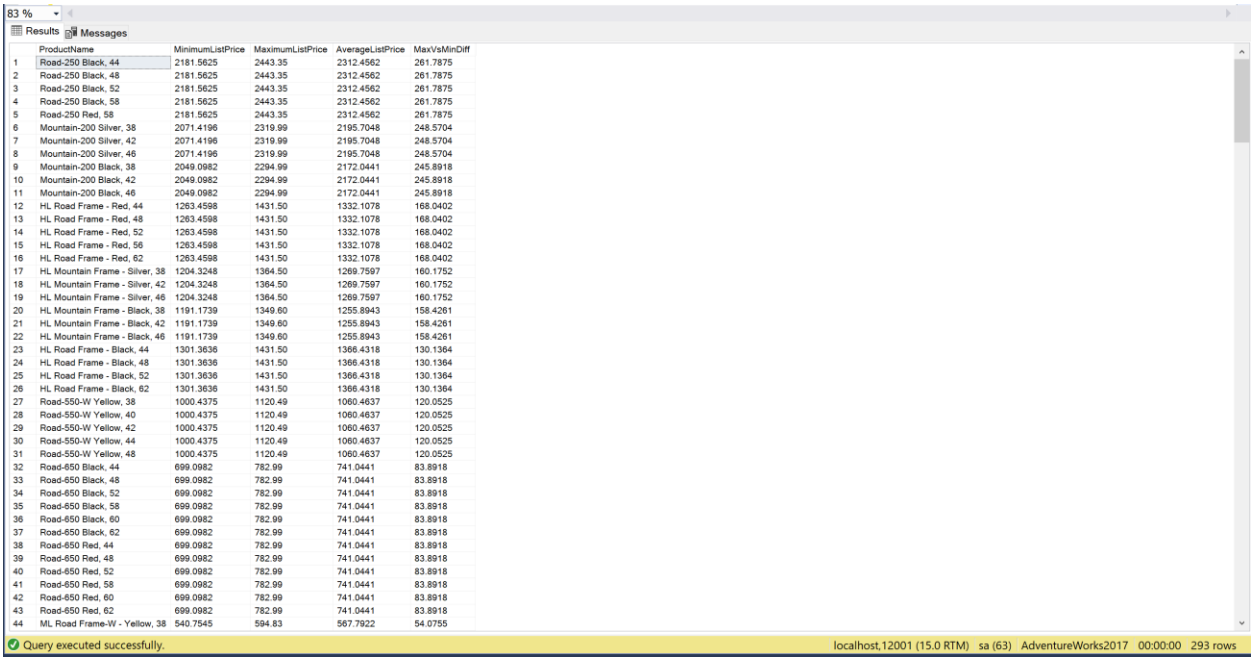
Order By

Table Name	Column Name	Sort
DerivedColumn	MaxVsMinDiff	DESC

Problem Solving Query

```
USE AdventureWorks2017;
SELECT DISTINCT
       P.Name AS ProductName,
       MIN(PLH.ListPrice) AS MinimumListPrice,
       MAX(PLH.ListPrice) AS MaximumListPrice,
       AVG(PLH.ListPrice) AS AverageListPrice,
       MAX(PLH.ListPrice) - MIN(PLH.ListPrice) AS MaxVsMinDiff
FROM Production.Product AS P
    INNER JOIN Production.ProductListPriceHistory AS PLH
        ON PLH.ProductID = P.ProductID
GROUP BY P.Name
ORDER BY MaxVsMinDiff DESC;
```

Sample Relational output with total number of rows returned(293)



83 %

Results | Messages

	ProductName	MinimumListPrice	MaximumListPrice	AverageListPrice	MaxVsMinDiff
1	Road-250 Black, 44	2181.5625	2443.35	2312.4562	261.7875
2	Road-250 Black, 48	2181.5625	2443.35	2312.4562	261.7875
3	Road-250 Black, 52	2181.5625	2443.35	2312.4562	261.7875
4	Road-250 Black, 58	2181.5625	2443.35	2312.4562	261.7875
5	Road-250 Red, 58	2181.5625	2443.35	2312.4562	261.7875
6	Mountain-200 Silver, 38	2071.4196	2319.99	2195.7048	248.5704
7	Mountain-200 Silver, 42	2071.4196	2319.99	2195.7048	248.5704
8	Mountain-200 Silver, 46	2071.4196	2319.99	2195.7048	248.5704
9	Mountain-200 Black, 38	2049.0982	2294.99	2172.0441	245.8918
10	Mountain-200 Black, 42	2049.0982	2294.99	2172.0441	245.8918
11	Mountain-200 Black, 46	2049.0982	2294.99	2172.0441	245.8918
12	HL Road Frame - Red, 44	1263.4598	1431.50	1332.1078	168.0402
13	HL Road Frame - Red, 48	1263.4598	1431.50	1332.1078	168.0402
14	HL Road Frame - Red, 52	1263.4598	1431.50	1332.1078	168.0402
15	HL Road Frame - Red, 56	1263.4598	1431.50	1332.1078	168.0402
16	HL Road Frame - Red, 62	1263.4598	1431.50	1332.1078	168.0402
17	HL Mountain Frame - Silver, 38	1204.3248	1364.50	1269.7597	160.1752
18	HL Mountain Frame - Silver, 42	1204.3248	1364.50	1269.7597	160.1752
19	HL Mountain Frame - Silver, 46	1204.3248	1364.50	1269.7597	160.1752
20	HL Mountain Frame - Black, 38	1191.1739	1349.60	1255.8943	158.4261
21	HL Mountain Frame - Black, 42	1191.1739	1349.60	1255.8943	158.4261
22	HL Mountain Frame - Black, 46	1191.1739	1349.60	1255.8943	158.4261
23	HL Road Frame - Black, 44	1301.3636	1431.50	1366.4318	130.1364
24	HL Road Frame - Black, 48	1301.3636	1431.50	1366.4318	130.1364
25	HL Road Frame - Black, 52	1301.3636	1431.50	1366.4318	130.1364
26	HL Road Frame - Black, 62	1301.3636	1431.50	1366.4318	130.1364
27	Road-550-W Yellow, 38	1000.4375	1120.49	1060.4637	120.0525
28	Road-550-W Yellow, 40	1000.4375	1120.49	1060.4637	120.0525
29	Road-550-W Yellow, 42	1000.4375	1120.49	1060.4637	120.0525
30	Road-550-W Yellow, 44	1000.4375	1120.49	1060.4637	120.0525
31	Road-550-W Yellow, 48	1000.4375	1120.49	1060.4637	120.0525
32	Road-650 Black, 44	699.0982	782.99	741.0441	83.8918
33	Road-650 Black, 48	699.0982	782.99	741.0441	83.8918
34	Road-650 Black, 52	699.0982	782.99	741.0441	83.8918
35	Road-650 Black, 58	699.0982	782.99	741.0441	83.8918
36	Road-650 Black, 60	699.0982	782.99	741.0441	83.8918
37	Road-650 Black, 62	699.0982	782.99	741.0441	83.8918
38	Road-650 Red, 44	699.0982	782.99	741.0441	83.8918
39	Road-650 Red, 48	699.0982	782.99	741.0441	83.8918
40	Road-650 Red, 52	699.0982	782.99	741.0441	83.8918
41	Road-650 Red, 58	699.0982	782.99	741.0441	83.8918
42	Road-650 Red, 60	699.0982	782.99	741.0441	83.8918
43	Road-650 Red, 62	699.0982	782.99	741.0441	83.8918
44	ML Road Frame-W - Yellow, 38	540.7545	594.83	567.7922	54.0755

Query executed successfully. localhost,12001 (15.0 RTM) sa (63) AdventureWorks2017 00:00:00 293 rows

Sample JSON Output with total number of rows returned(411)

```
USE AdventureWorks2017;


SELECT DISTINCT
       P.Name AS ProductName,
       MIN(PLH.ListPrice) AS MinimumListPrice,
       MAX(PLH.ListPrice) AS MaximumListPrice,
       AVG(PLH.ListPrice) AS AverageListPrice,
       MAX(PLH.ListPrice) - MIN(PLH.ListPrice) AS MaxVsMinDiff
FROM Production.Product AS P
    INNER JOIN Production.ProductListPriceHistory AS PLH
        ON PLH.ProductID = P.ProductID
GROUP BY P.Name
ORDER BY MaxVsMinDiff DESC
FOR JSON PATH,ROOT('ListPriceInfo'),INCLUDE_NULL_VALUES;
```

Proposition 14:

Columns from Standard View

Columns from Key View

Columns from their respective table in the select clause

Table Name	Column Name
FactInternetSales	OrderDate
DimProduct	Color
DimCustomer	BirthDate
DerivedColumn	Age OverallTotalColorsOrdered

Order By

Table Name	Column Name	Sort
Dimproduct	Color	Default
DerivedColumn	Age OverallTotalColorsOrdered	 DESC

Problem Solving Query

Scalar function --Create a function that returns the age of the customer on the day they placed an order

```
USE [AdventureWorksDW2017]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO


DECLARE FUNCTION [dbo].[GetCustomerAgeForOrder]
(

	@BirthDate AS DATE,
	@OrderDate AS DATETIME
)
RETURNS INT
AS
BEGIN

	RETURN DATEDIFF(YEAR, @birthdate, @OrderDate) -
	CASE WHEN 100 * MONTH(@OrderDate) + DAY(@OrderDate)
	< 100 * MONTH(@birthdate) + DAY(@birthdate)
	THEN 1 ELSE 0
	END;

END;

USE AdventureWorksDW2017;
SELECT DISTINCT
	dbo.GetCustomerAgeForOrder(C.BirthDate, FIS.OrderDate) AS Age,
	P.Color,
	COUNT(*) OVER (PARTITION BY dbo.GetCustomerAgeForOrder(C.BirthDate,
FIS.OrderDate),
                                  P.Color
                    ) OverallTotalOfColorsOrdered
FROM dbo.FactInternetSales AS FIS
    INNER JOIN dbo.DimCustomer AS C
        ON C.CustomerKey = FIS.CustomerKey
    INNER JOIN dbo.DimProduct AS P
        ON P.ProductKey = FIS.ProductKey
WHERE P.Color IS NOT NULL
      AND P.Color <> 'NA'
GROUP BY dbo.GetCustomerAgeForOrder(C.BirthDate, FIS.OrderDate),
         C.CustomerKey,
         P.EnglishProductName,
         P.Color
ORDER BY Age,
         OverallTotalOfColorsOrdered DESC,
         P.Color;
```

Sample Relational Output with total number of rows returned(411)

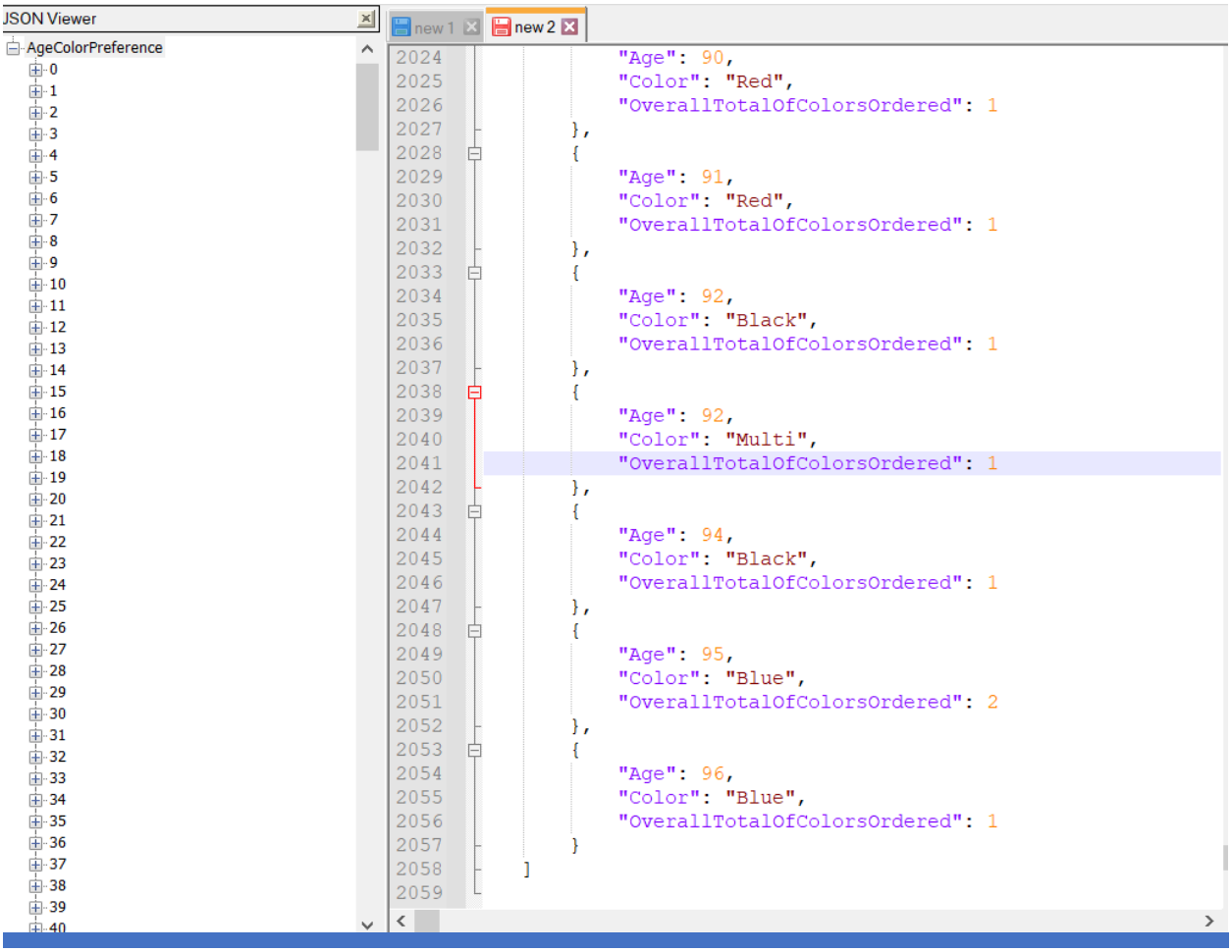
Editor | Results | Messages

	Age	Color	OverallTotalOfColorsOrdered
1	24	Red	8
2	25	Red	34
3	25	Black	12
4	25	Yellow	3
5	25	Silver	2
6	26	Red	68
7	26	Black	53
8	26	Yellow	22
9	26	Silver	19
10	26	Multi	6
11	26	Blue	5
12	26	White	2
13	27	Black	195
14	27	Red	142
15	27	Yellow	83
16	27	Multi	59
17	27	Blue	58
18	27	Silver	58
19	27	White	9
20	28	Black	267
21	28	Red	125
22	28	Yellow	105
23	28	Silver	102
24	28	Multi	98
25	28	Blue	89
26	28	White	19
27	29	Black	305
28	29	Yellow	169
29	29	Red	147
30	29	Multi	112
31	29	Blue	108
32	29	Silver	101
33	29	White	16
34	30	Black	295
35	30	Red	168
36	30	Yellow	140
37	30	Multi	125
38	30	Blue	121
39	30	Silver	84
40	30	White	25
41	31	Black	350
42	31	Red	171
43	31	Yellow	139
44	31	Blue	137

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (64) | AdventureWorksDW2017 | 00:00:00 | 411 rows

Sample JSON Output with total number of rows returned(411)

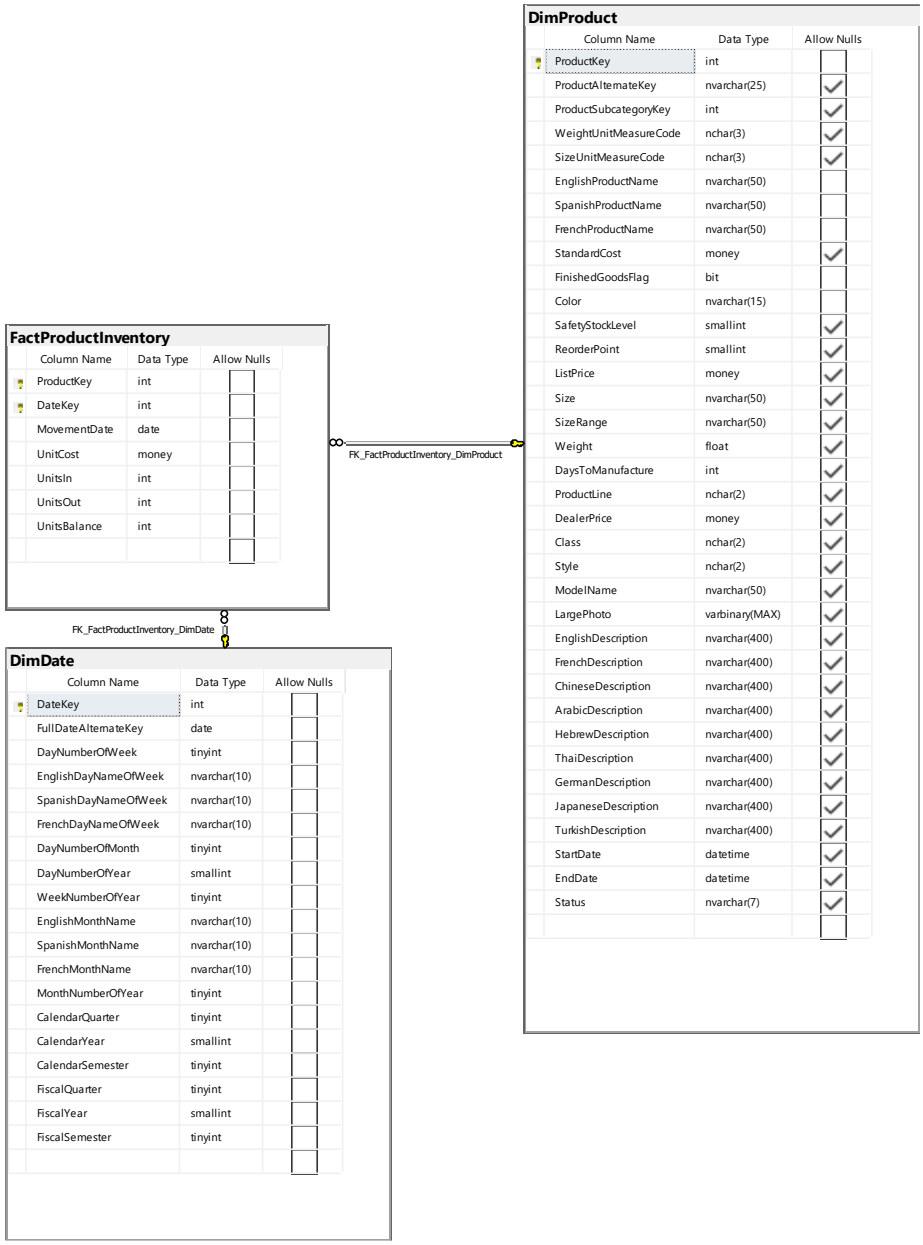
```
USE AdventureWorksDW2017;
SELECT DISTINCT
       dbo.GetCustomerAgeForOrder(C.BirthDate, FIS.OrderDate) AS Age,
       P.Color,
       COUNT(*) OVER (PARTITION BY dbo.GetCustomerAgeForOrder(C.BirthDate,
FIS.OrderDate),
                                   P.Color
                     ) OverallTotalOfColorsOrdered
FROM dbo.FactInternetSales AS FIS
    INNER JOIN dbo.DimCustomer AS C
        ON C.CustomerKey = FIS.CustomerKey
    INNER JOIN dbo.DimProduct AS P
        ON P.ProductKey = FIS.ProductKey
WHERE P.Color IS NOT NULL
      AND P.Color <> 'NA'
GROUP BY dbo.GetCustomerAgeForOrder(C.BirthDate, FIS.OrderDate),
         C.CustomerKey,
         P.EnglishProductName,
         P.Color
ORDER BY Age,
         OverallTotalOfColorsOrdered DESC,
         P.Color
FOR JSON PATH, ROOT('AgeColorPreference'),INCLUDE_NULL_VALUES;
```



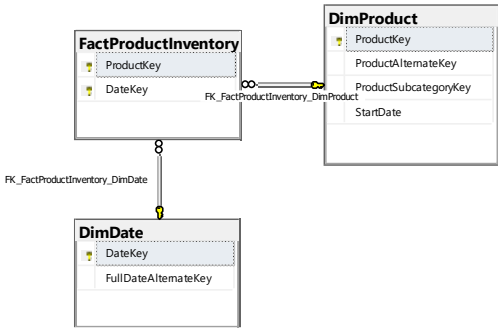
Proposition 15: Determine whether or not products that have a unit price greater than 1000 are in stock during the month of june and december for each year

Explanation: create function to calculate the difference between unitsin and unitsout if the difference is greater than 0 then output the status of the product as in stock if the difference is less than 0 then output the status of the product as not in stock

Columns from Standard View



Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
FactProductInventory	UnitCost ProductKey MovementDate UnitsIn UnitsOut
DimProduct	ProductKey EnglishProductName ModelName
DerivedColumn	StockLevel(dbo.UnitsStock function)

Order By

Table Name	Column Name	Sort
FactProductInventory	Month Year UnitsCost	Default Default DESC
DerivedColumn	StockLevel	Default

Problem Solving Query

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE FUNCTION dbo.UnitStock
(

	@UnitsIn INT, @UnitsOut INT
)
RETURNS NVARCHAR(30)
AS
BEGIN

	DECLARE @Stock NVARCHAR(30)
```

```
	SELECT @Stock = CASE
			WHEN (@UnitsIn - @UnitsOut) <= 0
			THEN  'NOT IN STOCK'
			WHEN  (@UnitsIn - @UnitsOut) > 0
			THEN  'IN STOCK'
			ELSE 'Unknown'
			END;
	RETURN @Stock;

END;
GO
USE AdventureWorksDW2017;
SELECT F.UnitCost,
	F.ProductKey,
	P.EnglishProductName,
	P.ModelName,
	DATEPART(MM, F.MovementDate) AS Month,
	DATEPART(YYYY, F.MovementDate) AS Year,
	dbo.UnitStock(F.UnitsIn, F.UnitsOut) AS Stocklevel
FROM dbo.FactProductInventory AS F
    INNER JOIN dbo.DimDate AS D
        ON D.DateKey = F.DateKey
    INNER JOIN dbo.DimProduct AS P
        ON P.ProductKey = F.ProductKey
WHERE (
          DATEPART(MM, F.MovementDate) = 6
          OR DATEPART(MM, F.MovementDate) = 12
      )
      AND (F.UnitCost > 1000)
GROUP BY DATEPART(MM, F.MovementDate),
         DATEPART(YYYY, F.MovementDate),
         dbo.UnitStock(F.UnitsIn, F.UnitsOut),
         F.UnitCost,
         F.ProductKey,
         P.EnglishProductName,
         P.ModelName
ORDER BY F.UnitCost DESC,
         Month,
         Year,
         Stocklevel;
```

Sample Relational output with total number of rows returned(350)

Editor | Results | Messages

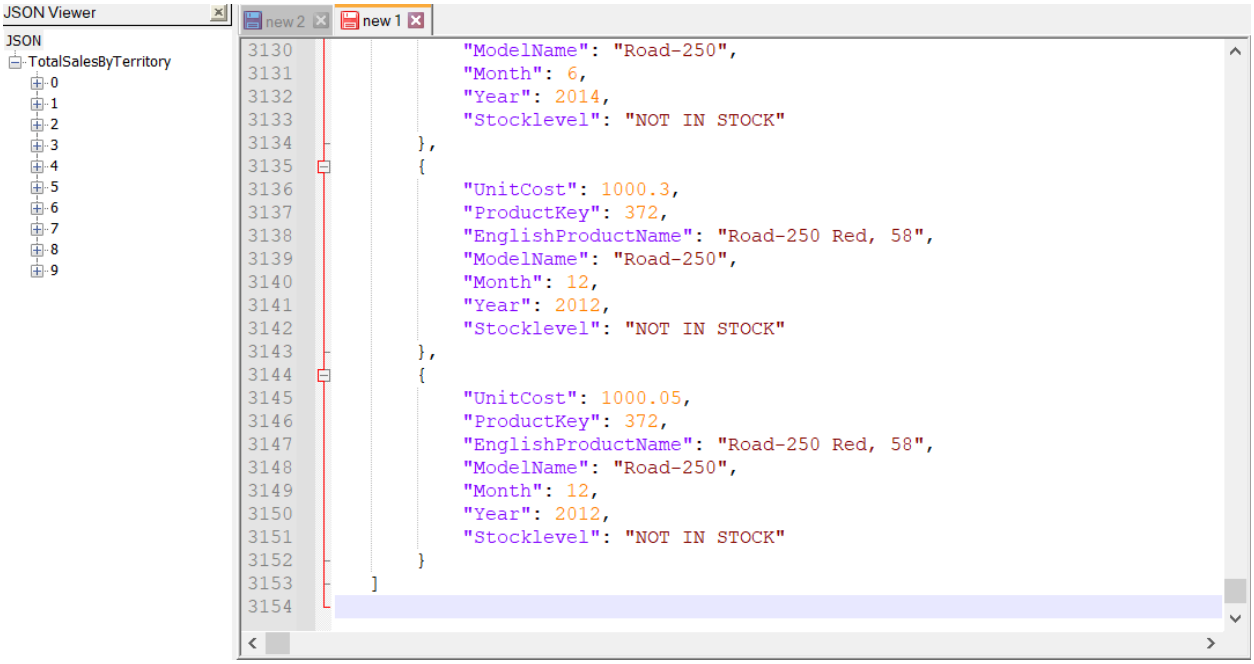
	UnitCost	ProductKey	EnglishProductName	ModelName	Month	Year	Stocklevel
1	1131.25	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
2	1129.00	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
3	1127.08	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
4	1126.74	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
5	1125.73	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
6	1124.38	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
7	1123.04	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
8	1122.59	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
9	1122.03	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
10	1118.56	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
11	1118.22	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
12	1115.77	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
13	1114.77	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
14	1114.44	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
15	1112.21	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
16	1109.00	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
17	1108.78	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
18	1107.89	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
19	1104.03	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
20	1103.30	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
21	1102.28	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
22	1102.05	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
23	1101.74	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
24	1101.42	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
25	1101.18	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
26	1100.87	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
27	1099.41	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
28	1098.97	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
29	1098.01	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
30	1097.55	371	Road-250 Red, 58	Road-250	6	2014	NOT IN STOCK
31	1095.78	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
32	1094.14	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
33	1093.26	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
34	1091.19	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
35	1088.90	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
36	1088.56	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
37	1087.99	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
38	1086.73	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
39	1085.75	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
40	1085.07	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
41	1083.12	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
42	1082.94	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
43	1080.08	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK
44	1079.81	371	Road-250 Red, 58	Road-250	12	2013	NOT IN STOCK

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (58) | AdventureWorksDW2017 | 00:00:00 | 350 rows

Sample JSON output with total number of rows returned(350)

```
USE AdventureWorksDW2017;
SELECT F.UnitCost,
       F.ProductKey,
       P.EnglishProductName,
       P.ModelName,
       DATEPART(MM, F.MovementDate) AS Month,
       DATEPART(YYYY, F.MovementDate) AS Year,
       dbo.UnitStock(F.UnitsIn, F.UnitsOut) AS Stocklevel
FROM dbo.FactProductInventory AS F
    INNER JOIN dbo.DimDate AS D
        ON D.DateKey = F.DateKey
    INNER JOIN dbo.DimProduct AS P
        ON P.ProductKey = F.ProductKey
WHERE (
```

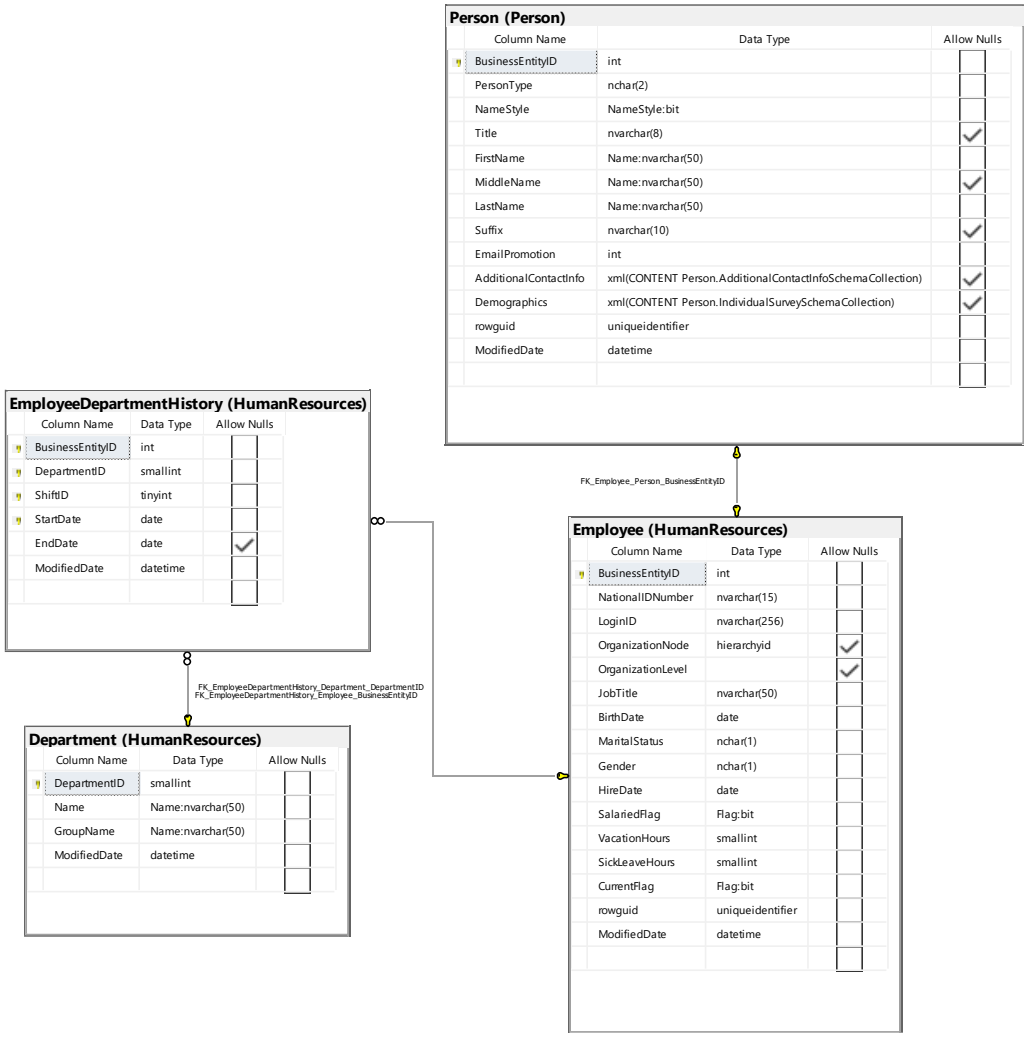
```
            DATEPART(MM, F.MovementDate) = 6
            OR DATEPART(MM, F.MovementDate) = 12
        )
        AND (F.UnitCost > 1000)
GROUP BY DATEPART(MM, F.MovementDate),
         DATEPART(YYYY, F.MovementDate),
         dbo.UnitStock(F.UnitsIn, F.UnitsOut),
         F.UnitCost,
         F.ProductKey,
         P.EnglishProductName,
         P.ModelName
ORDER BY F.UnitCost DESC,
         Month,
         Year,
         Stocklevel
FOR JSON PATH,ROOT('StockLevel'),INCLUDE_NULL_VALUES;
```



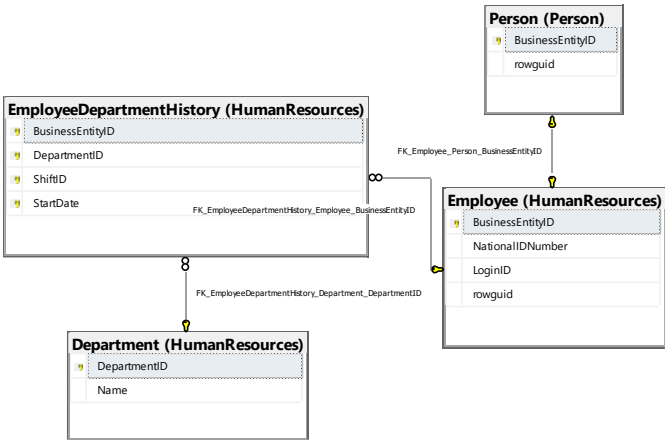
Proposition 16: How many years did each employee work up to the current date?

How many years did each employee work up to the current date? For every employee, return the organization and department they work in, their job title and full name and calculate the amount of years they worked up until the current year take into account employees that have switched department or quit

Columns from Standard View



Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name

Employee	JobTitle BusinessEntityID
Department	Name GroupName
Person	FirstName MiddleName LastName
DerivedColumn	FullName(From Person) YearsWorked(From dbo.YearsInDept Function)

Order By

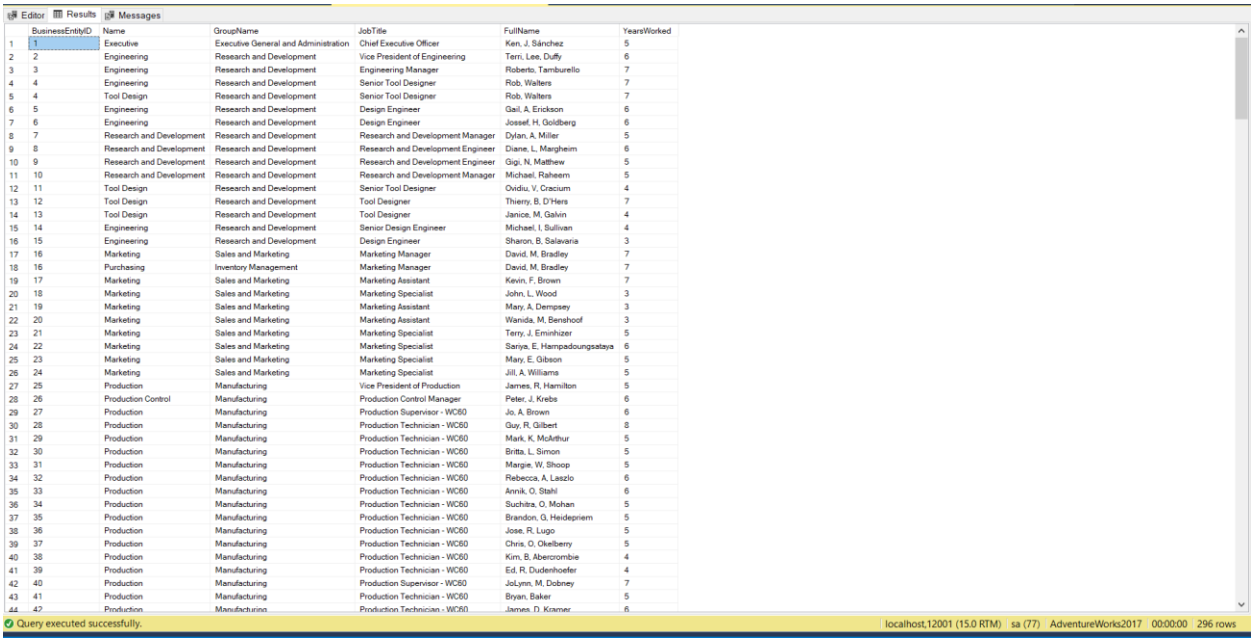
Table Name	ColumnName	Sort
Employee	BusinessEntityID	ASC
Employee	YearsWorked(function)	DESC

Problem Solving Query

```
USE AdventureWorks2017;

SET ANSI_NULLS ON;
GO
SET QUOTED_IDENTIFIER ON;
GO
DROP FUNCTION IF EXISTS dbo.YearsInDept;
GO
CREATE FUNCTION dbo.YearsInDept
(
    @EmployeeKey INT
)
RETURNS INT
AS
BEGIN

    DECLARE @Years AS INT;

    SELECT @Years = CASE
                        WHEN DH.EndDate IS NULL THEN
                            DATEDIFF(YEAR, E.HireDate, E.ModifiedDate)
                        ELSE
                            DATEDIFF(YEAR, DH.StartDate, DH.EndDate)
                    END
    FROM HumanResources.EmployeeDepartmentHistory AS DH
        INNER JOIN HumanResources.Employee AS E
            ON E.BusinessEntityID = E.BusinessEntityID
    WHERE E.BusinessEntityID = @EmployeeKey;

    RETURN @Years;
```

```
END;
GO

SELECT E.BusinessEntityID,
       D.Name,
       D.GroupName,
       E.JobTitle,
       CONCAT_WS(', ', P.FirstName, P.MiddleName, P.LastName) AS FullName,
       dbo.YearsInDept(E.BusinessEntityID) AS YearsWorked
FROM HumanResources.Employee AS E
    INNER JOIN HumanResources.EmployeeDepartmentHistory AS DH
        ON DH.BusinessEntityID = E.BusinessEntityID
    INNER JOIN Person.Person AS P
        ON P.BusinessEntityID = E.BusinessEntityID
    INNER JOIN HumanResources.Department AS D
        ON D.DepartmentID = DH.DepartmentID
GROUP BY CONCAT_WS(', ', P.FirstName, P.MiddleName, P.LastName),
         dbo.YearsInDept(E.BusinessEntityID),
         E.BusinessEntityID,
         E.JobTitle,
         DH.DepartmentID,
         D.Name,
         D.GroupName
ORDER BY E.BusinessEntityID ASC,
         YearsWorked DESC;
```

Sample Relational output with total number of rows returned(296)

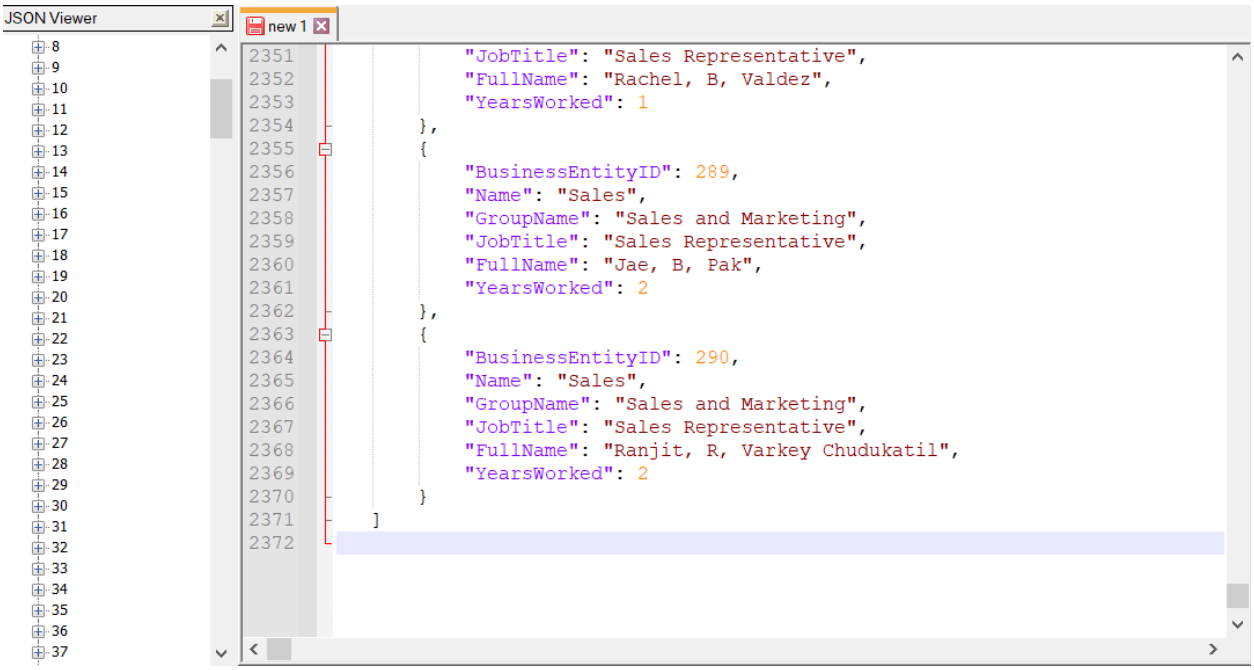


	BusinessEntityID	Name	GroupName	JobTitle	FullName	YearsWorked
1	1	Executive	Executive General and Administration	Chief Executive Officer	Ken, J. Sánchez	5
2	2	Engineering	Research and Development	Vice President of Engineering	Terri, Lee, Duffy	6
3	3	Engineering	Research and Development	Engineering Manager	Roberto, Tamburello	7
4	4	Engineering	Research and Development	Senior Tool Designer	Rob, Walters	7
5	4	Tool Design	Research and Development	Senior Tool Designer	Rob, Walters	7
6	5	Engineering	Research and Development	Design Engineer	Gail, A, Erickson	6
7	6	Engineering	Research and Development	Design Engineer	Jossef, H, Goldberg	6
8	7	Research and Development	Research and Development	Research and Development Manager	Dylan, A, Miller	5
9	8	Research and Development	Research and Development	Research and Development Engineer	Diane, L, Margheim	6
10	9	Research and Development	Research and Development	Research and Development Engineer	Gigi, N, Matthew	5
11	10	Research and Development	Research and Development	Research and Development Manager	Michael, Raheem	5
12	11	Tool Design	Research and Development	Senior Tool Designer	Ovidiu, V, Cracium	4
13	12	Tool Design	Research and Development	Tool Designer	Thierry, B, D'Hers	7
14	13	Tool Design	Research and Development	Tool Designer	Janice, M, Galvin	4
15	14	Engineering	Research and Development	Senior Design Engineer	Michael, I, Sullivan	4
16	15	Engineering	Research and Development	Design Engineer	Sharon, B, Salavaria	3
17	16	Marketing	Sales and Marketing	Marketing Manager	David, M, Bradley	7
18	16	Purchasing	Inventory Management	Marketing Manager	David, M, Bradley	7
19	17	Marketing	Sales and Marketing	Marketing Assistant	Kevin, F, Brown	7
20	18	Marketing	Sales and Marketing	Marketing Specialist	John, L, Wood	3
21	19	Marketing	Sales and Marketing	Marketing Assistant	Mary, A, Dempsey	3
22	20	Marketing	Sales and Marketing	Marketing Assistant	Wanida, M, Benshoof	3
23	21	Marketing	Sales and Marketing	Marketing Specialist	Terry, J, Eminhizer	5
24	22	Marketing	Sales and Marketing	Marketing Specialist	Sariya, E, Harnpadoungsataya	6
25	23	Marketing	Sales and Marketing	Marketing Specialist	Mary, E, Gibson	5
26	24	Marketing	Sales and Marketing	Marketing Specialist	Jill, A, Williams	5
27	25	Production	Manufacturing	Vice President of Production	James, R, Hamilton	5
28	26	Production Control	Manufacturing	Production Control Manager	Peter, J, Krebs	6
29	27	Production	Manufacturing	Production Supervisor - WC60	Jo, A, Brown	6
30	28	Production	Manufacturing	Production Technician - WC60	Guy, R, Gilbert	8
31	29	Production	Manufacturing	Production Technician - WC60	Mark, K, McArthur	5
32	30	Production	Manufacturing	Production Technician - WC60	Britta, L, Simon	5
33	31	Production	Manufacturing	Production Technician - WC60	Margie, W, Shoop	5
34	32	Production	Manufacturing	Production Technician - WC60	Rebecca, A, Laszlo	6
35	33	Production	Manufacturing	Production Technician - WC60	Annik, O, Stahl	6
36	34	Production	Manufacturing	Production Technician - WC60	Suchitra, O, Mohan	5
37	35	Production	Manufacturing	Production Technician - WC60	Brandon, G, Heidepriem	5
38	36	Production	Manufacturing	Production Technician - WC60	Jose, R, Lugo	5
39	37	Production	Manufacturing	Production Technician - WC60	Chris, O, Okelberry	5
40	38	Production	Manufacturing	Production Technician - WC60	Kim, B, Abercrombie	4
41	39	Production	Manufacturing	Production Technician - WC60	Ed, R, Dudenhoefer	4
42	40	Production	Manufacturing	Production Supervisor - WC60	JoLynn, M, Dobney	7
43	41	Production	Manufacturing	Production Technician - WC60	Bryan, Baker	5
44	42	Production	Manufacturing	Production Technician - WC60	James, D, Kramer	6

Sample JSON output with total number of rows returned(296)

```
SELECT E.BusinessEntityID,
       D.Name,
```

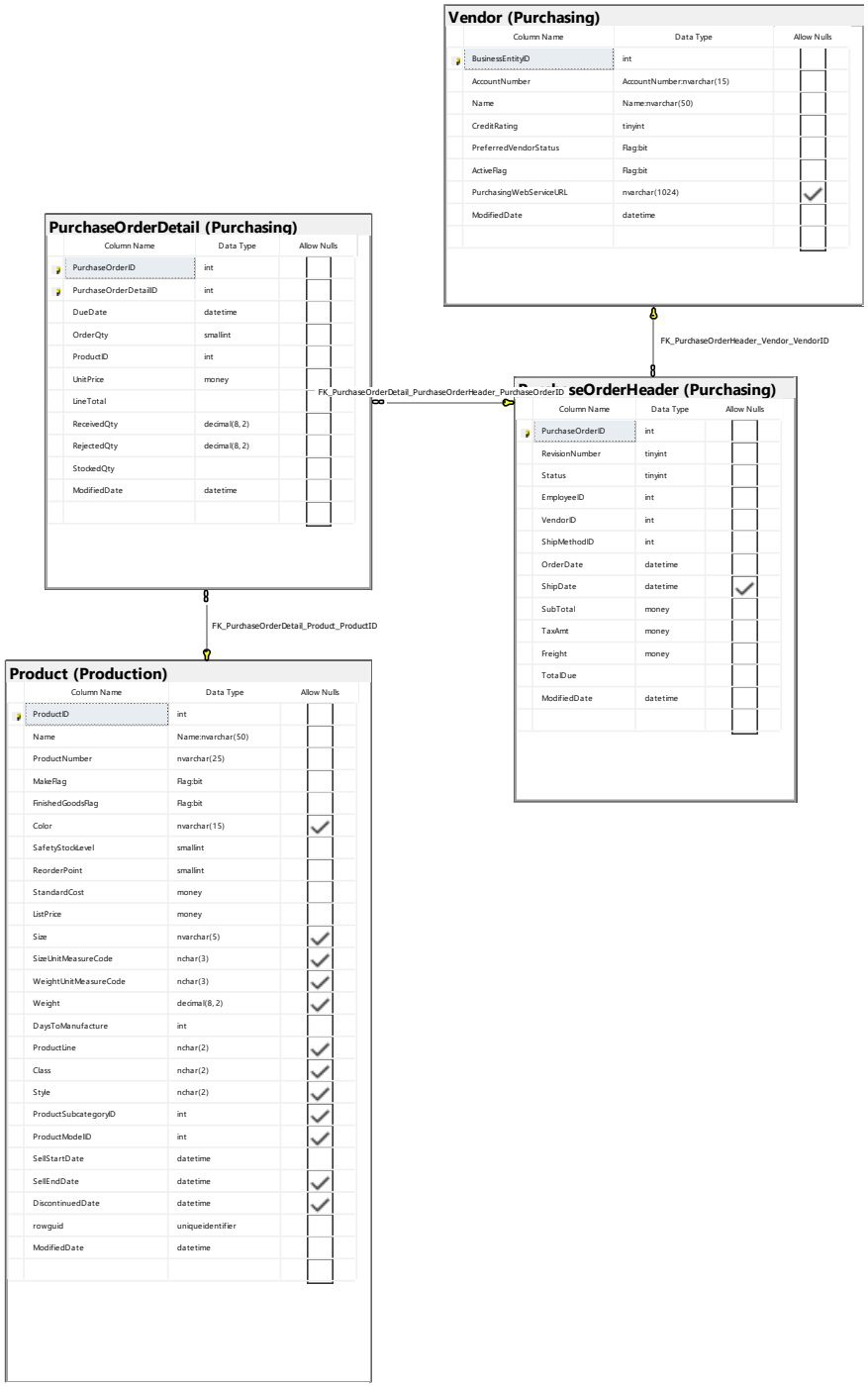
```
        D.GroupName,
        E.JobTitle,
        CONCAT_WS(', ', P.FirstName, P.MiddleName, P.LastName) AS FullName,
        dbo.YearsInDept(E.BusinessEntityID) AS YearsWorked
FROM HumanResources.Employee AS E
    INNER JOIN HumanResources.EmployeeDepartmentHistory AS DH
        ON DH.BusinessEntityID = E.BusinessEntityID
    INNER JOIN Person.Person AS P
        ON P.BusinessEntityID = E.BusinessEntityID
    INNER JOIN HumanResources.Department AS D
        ON D.DepartmentID = DH.DepartmentID
GROUP BY CONCAT_WS(', ', P.FirstName, P.MiddleName, P.LastName),
         dbo.YearsInDept(E.BusinessEntityID),
         E.BusinessEntityID,
         E.JobTitle,
         DH.DepartmentID,
         D.Name,
         D.GroupName
ORDER BY E.BusinessEntityID ASC,
         YearsWorked DESC
FOR JSON PATH, ROOT('EmployeeYearsWorked'),INCLUDE_NULL_VALUES;
```



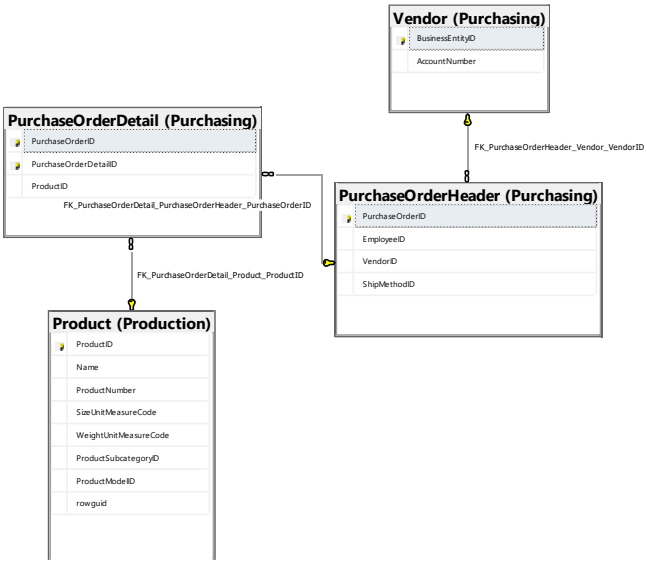
Proposition 17: Find all the products that were rejected

Explanation: Find all products that were rejected and include the name of the product, vendorid of the vendor who dealt with the product and the amount of items rejected

Columns from Standard View



Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
PurchaseOrderDetail	ProductID
PurchaseOrderHeader	VendorID
Product	Name
DerivedColumn	RejectedItems(dbo.RejectedProducts function)

Order By

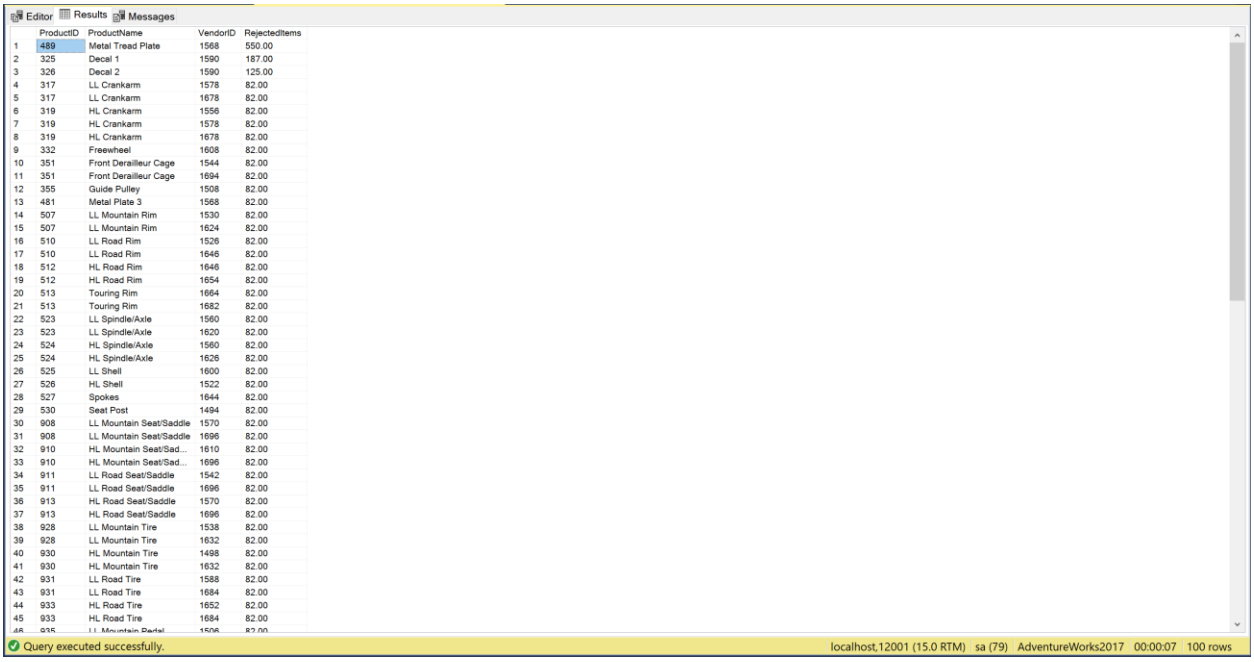
Table Name	Column Name	Sort
DerivedColumn	RejectedItems	DESC

Problem Solving Query

```
USE AdventureWorks2017;
SET ANSI_NULLS ON;
GO
SET QUOTED_IDENTIFIER ON;
GO

DROP FUNCTION IF EXISTS dbo.RejectedProducts;
GO

CREATE FUNCTION dbo.RejectedProducts
(
    @productId INT
)
RETURNS DECIMAL(8, 2)
AS
```

```
BEGIN

    DECLARE @RejectedQty INT;



    SELECT @RejectedQty = POD.RejectedQty
    FROM Purchasing.PurchaseOrderDetail AS POD
    WHERE POD.ProductID = @productId
          AND POD.RejectedQty > 0;
    IF (@RejectedQty IS NULL)
        SET @RejectedQty = 0;
    RETURN @RejectedQty;


END;
GO

SELECT PO.ProductID,
       P.Name AS ProductName,
       POH.VendorID,
       dbo.RejectedProducts(PO.ProductID) AS RejectedItems
FROM Purchasing.PurchaseOrderDetail AS PO
    INNER JOIN Purchasing.PurchaseOrderHeader AS POH
        ON POH.PurchaseOrderID = PO.PurchaseOrderID
    INNER JOIN Purchasing.Vendor AS V
        ON V.BusinessEntityID = POH.VendorID
    INNER JOIN Production.Product AS P
        ON P.ProductID = PO.ProductID
GROUP BY dbo.RejectedProducts(PO.ProductID),
         PO.ProductID,
         P.Name,
         POH.VendorID,
         V.Name
ORDER BY RejectedItems DESC OFFSET 0 ROWS FETCH FIRST 100 ROWS ONLY;
```

Sample Relational output with total number of rows returned(100)



Editor | Results | Messages

	ProductID	ProductName	VendorID	RejectedItems
1	489	Metal Tread Plate	1568	550.00
2	325	Decal 1	1590	187.00
3	326	Decal 2	1590	125.00
4	317	LL Crankarm	1578	82.00
5	317	LL Crankarm	1678	82.00
6	319	HL Crankarm	1556	82.00
7	319	HL Crankarm	1578	82.00
8	319	HL Crankarm	1678	82.00
9	332	Freewheel	1608	82.00
10	351	Front Derailleur Cage	1544	82.00
11	351	Front Derailleur Cage	1694	82.00
12	355	Guide Pulley	1508	82.00
13	481	Metal Plate 3	1568	82.00
14	507	LL Mountain Rim	1530	82.00
15	507	LL Mountain Rim	1624	82.00
16	510	LL Road Rim	1526	82.00
17	510	LL Road Rim	1646	82.00
18	512	HL Road Rim	1646	82.00
19	512	HL Road Rim	1654	82.00
20	513	Touring Rim	1664	82.00
21	513	Touring Rim	1682	82.00
22	523	LL Spindle/Axle	1560	82.00
23	523	LL Spindle/Axle	1620	82.00
24	524	HL Spindle/Axle	1560	82.00
25	524	HL Spindle/Axle	1626	82.00
26	525	LL Shell	1600	82.00
27	526	HL Shell	1522	82.00
28	527	Spokes	1644	82.00
29	530	Seat Post	1494	82.00
30	908	LL Mountain Seat/Saddle	1570	82.00
31	908	LL Mountain Seat/Saddle	1696	82.00
32	910	HL Mountain Seat/Sad...	1610	82.00
33	910	HL Mountain Seat/Sad...	1696	82.00
34	911	LL Road Seat/Saddle	1542	82.00
35	911	LL Road Seat/Saddle	1696	82.00
36	913	HL Road Seat/Saddle	1570	82.00
37	913	HL Road Seat/Saddle	1696	82.00
38	928	LL Mountain Tire	1538	82.00
39	928	LL Mountain Tire	1632	82.00
40	930	HL Mountain Tire	1498	82.00
41	930	HL Mountain Tire	1632	82.00
42	931	LL Road Tire	1588	82.00
43	931	LL Road Tire	1684	82.00
44	933	HL Road Tire	1652	82.00
45	933	HL Road Tire	1684	82.00

Query executed successfully. localhost,12001 (15.0 RTM) sa (79) AdventureWorks2017 00:00:07 100 rows

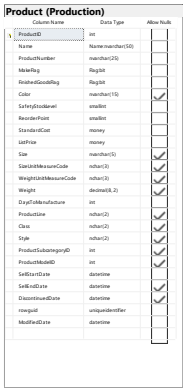
Sample JSON output with total number of rows returned(100)

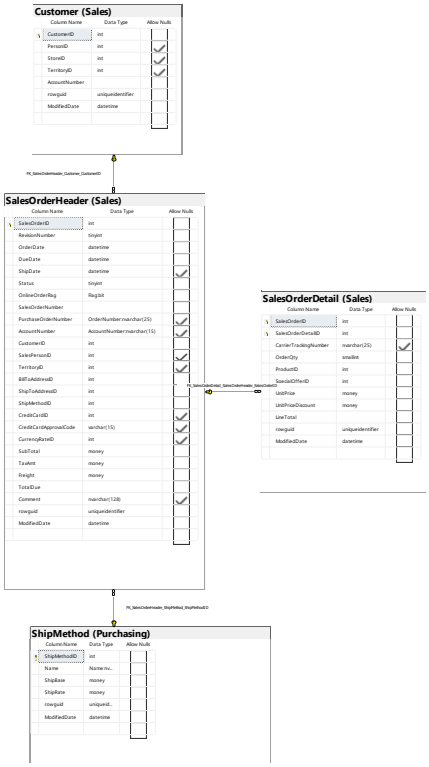
```
USE AdventureWorks2017;
SELECT PO.ProductID,
       P.Name AS ProductName,
       POH.VendorID,
       dbo.RejectedProducts(PO.ProductID) AS RejectedItems
FROM Purchasing.PurchaseOrderDetail AS PO
    INNER JOIN Purchasing.PurchaseOrderHeader AS POH
        ON POH.PurchaseOrderID = PO.PurchaseOrderID
    INNER JOIN Purchasing.Vendor AS V
        ON V.BusinessEntityID = POH.VendorID
    INNER JOIN Production.Product AS P
        ON P.ProductID = PO.ProductID
GROUP BY dbo.RejectedProducts(PO.ProductID),
         PO.ProductID,
         P.Name,
         POH.VendorID

ORDER BY RejectedItems DESC OFFSET 0 ROWS FETCH FIRST 100 ROWS ONLY
FOR JSON PATH,ROOT('Reject_Products'),INCLUDE_NULL_VALUES;
```

Proposition 18:Find the top 5 most expensive products to ship

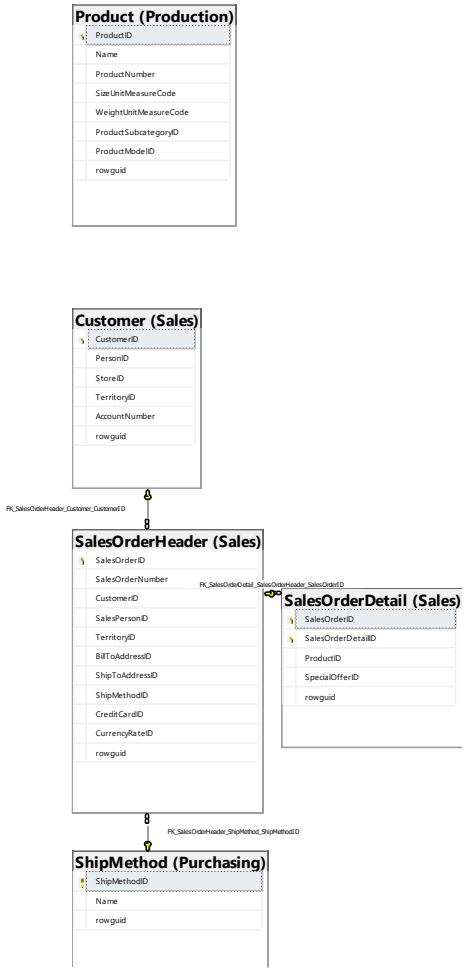
Explanation: find the top 5 most expensive products to ship based on the ship rate by the weight of the product in pounds(LB) plus the shippers base ship rate

Columns from Standard View





Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
SalesOrderDetail	ProductID
ShipMethod	ShipMethodID ShipBase ShipRate Name
Derived Column	ShipPrice (from dbo.ProductWeightLb)

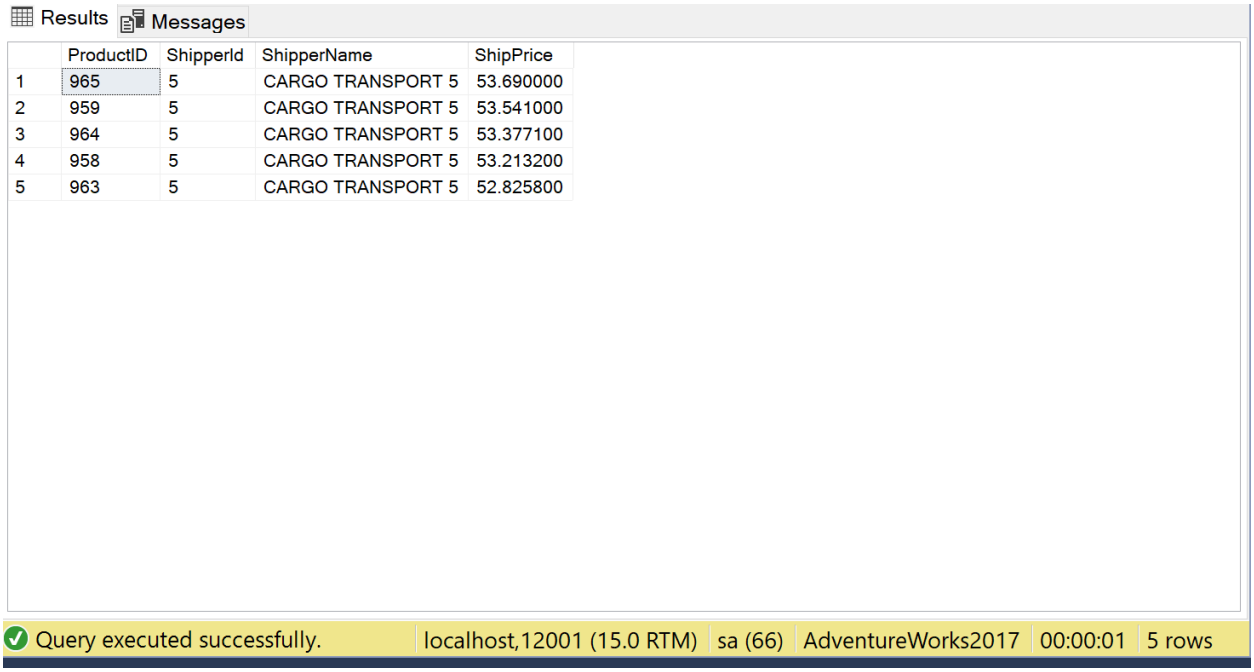
Order by

Table Name	Column Name	Sort
DerivedColumn	ShipPrice	DESC

Problem Solving Query

```
USE AdventureWorks2017;
GO
SET ANSI_NULLS ON;
GO
SET QUOTED_IDENTIFIER ON;
GO
DROP FUNCTION IF EXISTS dbo.ProductWeightLB;
GO
CREATE FUNCTION dbo.ProductWeightLB
(
    @productId INT
)
RETURNS DECIMAL(8, 2)
AS
BEGIN
    DECLARE @Weight DECIMAL(8, 2);

    SELECT @Weight = P.Weight
    FROM Production.Product AS P
    WHERE P.ProductID = @productId;

    RETURN @Weight;

END;
GO

USE AdventureWorks2017;
SELECT MAX(DISTINCT SOD.ProductID) AS ProductID,
       Ship.ShipMethodID AS ShipperId,
       Ship.Name AS ShipperName,
       (Ship.ShipBase + Ship.ShipRate * dbo.ProductWeightLb(P.ProductID)) AS ShipPrice
FROM Sales.SalesOrderHeader AS SOH
    INNER JOIN Purchasing.ShipMethod AS Ship
        ON Ship.ShipMethodID = SOH.ShipMethodID
    INNER JOIN Sales.Customer AS C
        ON C.CustomerID = SOH.CustomerID
    INNER JOIN Sales.SalesOrderDetail AS SOD
        ON SOD.SalesOrderID = SOH.SalesOrderID
    INNER JOIN Production.Product AS P
        ON P.ProductID = SOD.ProductID
WHERE P.WeightUnitMeasureCode = 'LB'
GROUP BY (Ship.ShipBase + Ship.ShipRate * dbo.ProductWeightLb(P.ProductID)),
         Ship.ShipMethodID,
         Ship.Name
ORDER BY ShipPrice DESC OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY
--FOR JSON PATH,ROOT('MostExpensiveShipments'),INCLUDE_NULL_VALUES;
```

Sample Relational output with total number of rows returned(5)



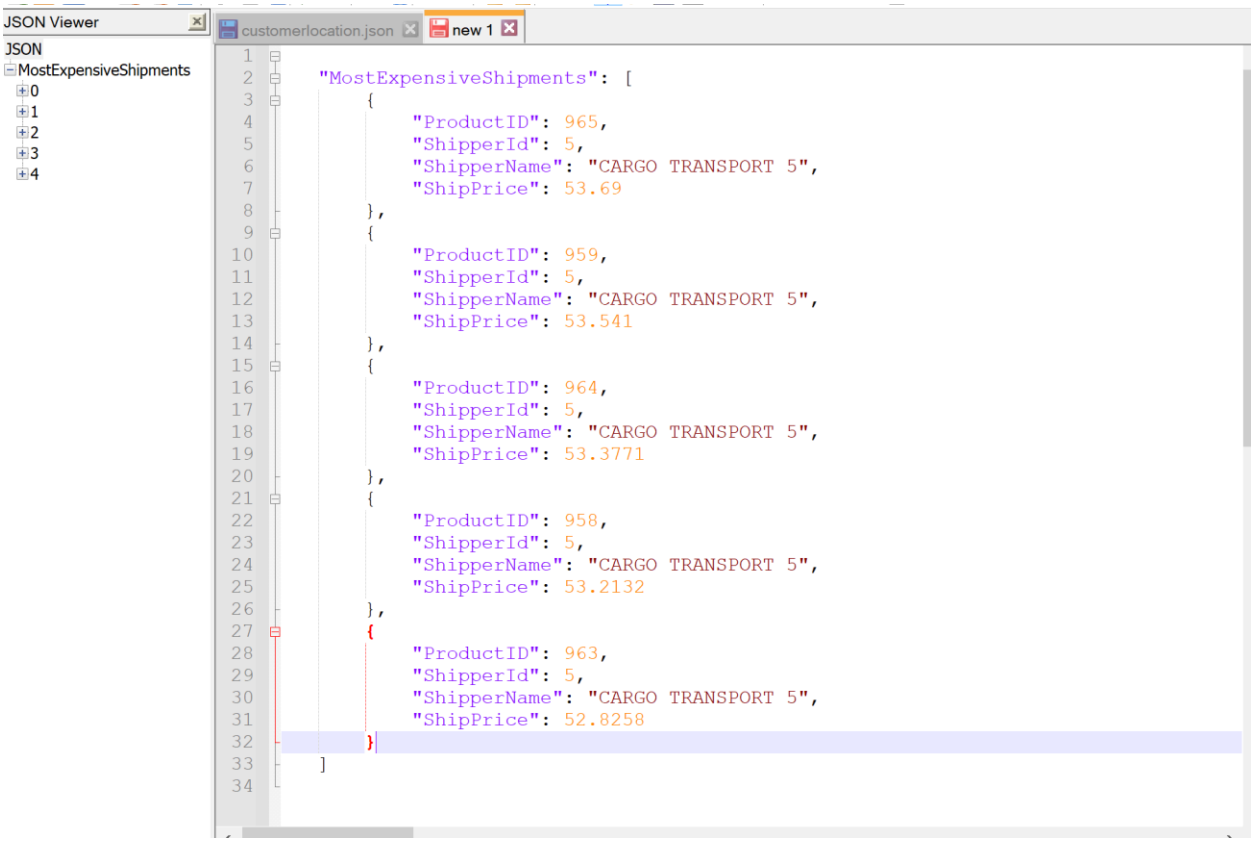
Results | Messages

	ProductID	ShipperId	ShipperName	ShipPrice
1	965	5	CARGO TRANSPORT 5	53.690000
2	959	5	CARGO TRANSPORT 5	53.541000
3	964	5	CARGO TRANSPORT 5	53.377100
4	958	5	CARGO TRANSPORT 5	53.213200
5	963	5	CARGO TRANSPORT 5	52.825800

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (66) | AdventureWorks2017 | 00:00:01 | 5 rows

Sample JSON output with total number of rows returned(5)

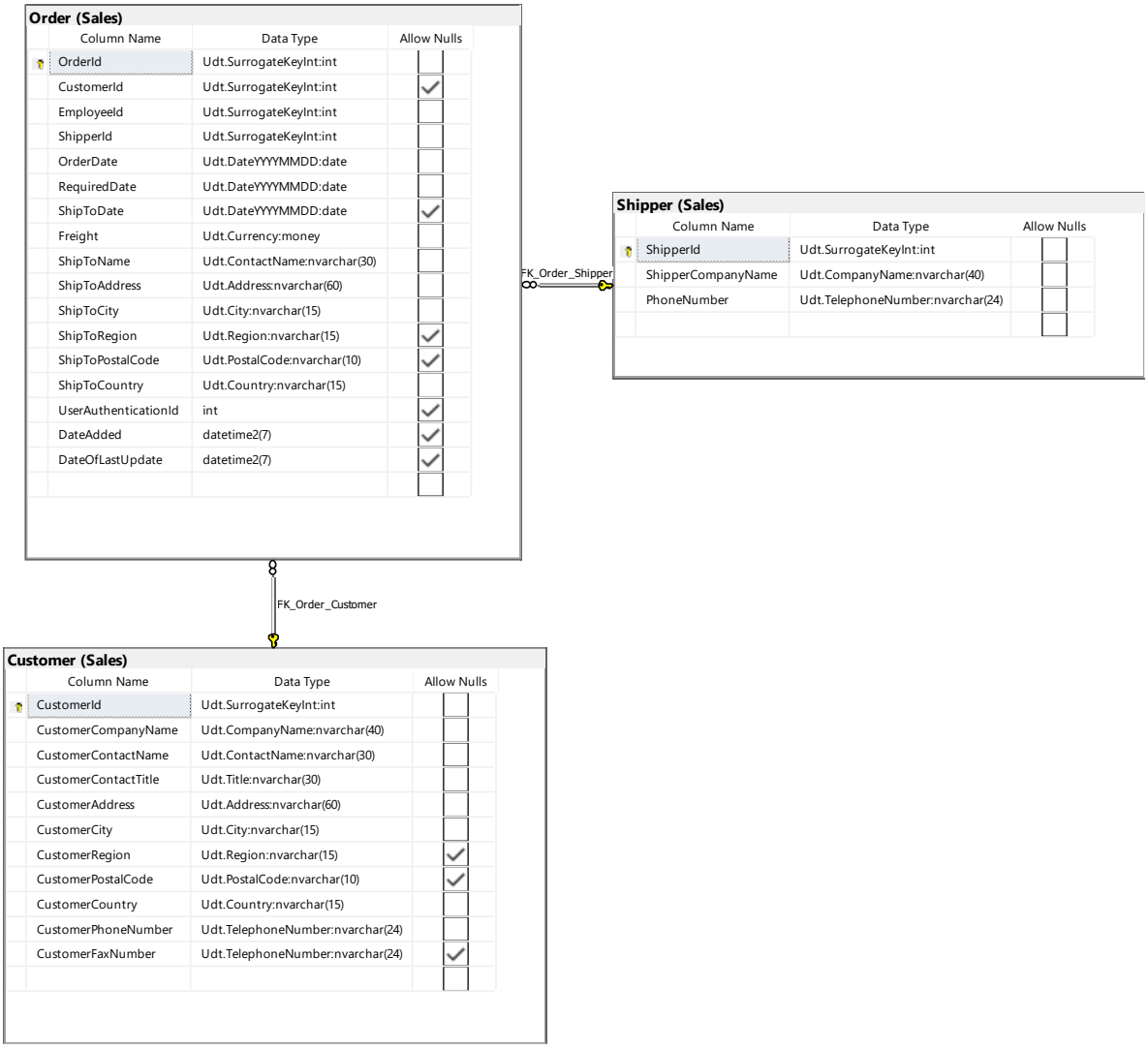
```
USE AdventureWorks2017;
SELECT MAX(DISTINCT SOD.ProductID) AS ProductID,
       Ship.ShipMethodID AS ShipperId,
       Ship.Name AS ShipperName,
       (Ship.ShipBase + Ship.ShipRate * dbo.ProductWeightLB(P.ProductID)) AS ShipPrice
FROM Sales.SalesOrderHeader AS SOH
    INNER JOIN Purchasing.ShipMethod AS Ship
        ON Ship.ShipMethodID = SOH.ShipMethodID
    INNER JOIN Sales.Customer AS C
        ON C.CustomerID = SOH.CustomerID
    INNER JOIN Sales.SalesOrderDetail AS SOD
        ON SOD.SalesOrderID = SOH.SalesOrderID
    INNER JOIN Production.Product AS P
        ON P.ProductID = SOD.ProductID
WHERE P.WeightUnitMeasureCode = 'LB'
GROUP BY (Ship.ShipBase + Ship.ShipRate * dbo.ProductWeightLB(P.ProductID)),
         Ship.ShipMethodID,
         Ship.Name
ORDER BY ShipPrice DESC OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY
FOR JSON PATH, ROOT('MostExpensiveShipments'), INCLUDE_NULL_VALUES;
```



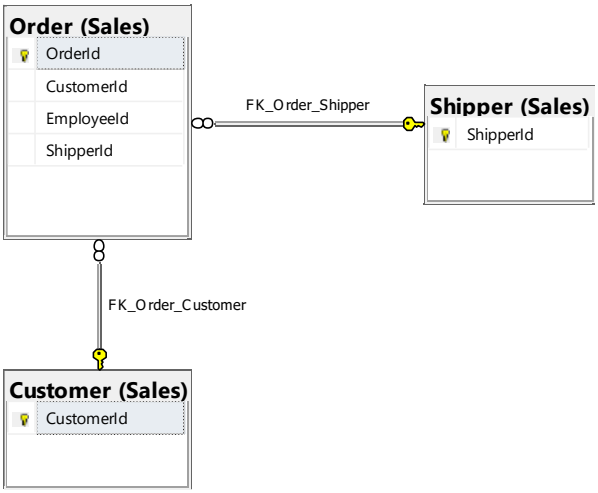
Proposition 19 Find out if shipments were shipped on time

Explanation: Determine whether a shipment was late, early or on time for each customer based on the difference between the requireddate of the order and the ship date and include the customerid, amount of orders shipped to them, the shippercompany that shipped those orders and the customers country

Columns from Standard View



Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
Order	OrderId RequiredDate ShipToDate
Shipper	ShipperId
Customer	CustomerId CustomerCountry
DerivedColumn	ShippingTime(dbo.ShipTime function)

Order By

Table Name	Column Name	Sort
Customer	CustomerId	DESC
DerivedColumn	Shipments	DESC

Problem Solving Query

```
USE Northwinds2020TSQLV6;
SET ANSI_NULLS ON;
GO
SET QUOTED_IDENTIFIER ON;
GO

CREATE FUNCTION dbo.shipTime
(
    @orderid INT,
    @requireddate DATE,
    @shiptodate DATE
)
RETURNS NVARCHAR(30)
AS
BEGIN

    DECLARE @shiptime NVARCHAR(30);


    SELECT @shiptime = CASE
                           WHEN DATEDIFF(DAY, O.RequiredDate, O.ShipToDate) < 0 THEN
                               'Late Shipment'
                           WHEN DATEDIFF(DAY, O.RequiredDate, O.ShipToDate) > 0 THEN
                               'Shipped Early'
                           ELSE
                               'ON TIME'
                       END
    FROM Sales.[Order] AS O
    WHERE O.OrderId = @orderid
          AND O.RequiredDate = @requireddate
          AND O.ShipToDate = @shiptodate;


    RETURN @shiptime;
```

```
END;
GO
SELECT DISTINCT
       c.CustomerId,
       SUM(S.ShipperId) AS Shipments,
       S.ShipperCompanyName AS ShipperCompany,
       c.CustomerCountry,
       dbo.shipTime(O.OrderId, O.RequiredDate, O.ShipToDate) AS ShippingTime
FROM Sales.[Order] AS O
    INNER JOIN Sales.Shipper AS S
        ON S.ShipperId = O.ShipperId
    INNER JOIN Sales.Customer AS c
        ON c.CustomerId = O.CustomerId
WHERE dbo.shipTime(O.OrderId, O.RequiredDate, O.ShipToDate) IS NOT NULL
GROUP BY dbo.shipTime(O.OrderId, O.RequiredDate, O.ShipToDate),
         c.CustomerId,
         S.ShipperCompanyName,
         c.CustomerCountry
ORDER BY c.CustomerId DESC,
         Shipments DESC;
```

Sample Relational output with total number of rows returned(271)

Editor | Results | Messages

	CustomerId	Shipments	ShipperCompany	CustomerCountry	ShippingTime
1	91	12	Shipper ZHISN	Poland	Late Shipment
2	91	2	Shipper ETYNR	Poland	Late Shipment
3	91	2	Shipper ETYNR	Poland	ON TIME
4	91	1	Shipper GVSUA	Poland	Late Shipment
5	90	9	Shipper ZHISN	Finland	Late Shipment
6	90	4	Shipper GVSUA	Finland	Late Shipment
7	89	15	Shipper ZHISN	USA	Late Shipment
8	89	8	Shipper ETYNR	USA	Late Shipment
9	89	3	Shipper GVSUA	USA	Late Shipment
10	89	2	Shipper ETYNR	USA	Shipped Early
11	89	1	Shipper GVSUA	USA	Shipped Early
12	88	8	Shipper ETYNR	Brazil	Late Shipment
13	88	4	Shipper GVSUA	Brazil	Late Shipment
14	88	3	Shipper ZHISN	Brazil	Late Shipment
15	87	15	Shipper ZHISN	Finland	Late Shipment
16	87	10	Shipper ETYNR	Finland	Late Shipment
17	87	4	Shipper GVSUA	Finland	Late Shipment
18	87	3	Shipper ZHISN	Finland	Shipped Early
19	86	14	Shipper ETYNR	Germany	Late Shipment
20	86	3	Shipper GVSUA	Germany	Late Shipment
21	85	6	Shipper ZHISN	France	Late Shipment
22	85	4	Shipper ETYNR	France	Late Shipment
23	85	1	Shipper GVSUA	France	Late Shipment
24	84	10	Shipper ETYNR	France	Late Shipment
25	84	9	Shipper ZHISN	France	Late Shipment
26	84	2	Shipper GVSUA	France	Late Shipment
27	83	12	Shipper ZHISN	Denmark	Late Shipment
28	83	6	Shipper ETYNR	Denmark	Late Shipment
29	83	4	Shipper GVSUA	Denmark	Late Shipment
30	82	4	Shipper ETYNR	USA	Late Shipment
31	82	3	Shipper ZHISN	USA	Late Shipment
32	81	9	Shipper ZHISN	Brazil	Late Shipment
33	81	6	Shipper ETYNR	Brazil	Late Shipment
34	80	15	Shipper ZHISN	Mexico	Late Shipment
35	80	10	Shipper ETYNR	Mexico	Late Shipment
36	79	8	Shipper ETYNR	Germany	Late Shipment
37	79	2	Shipper GVSUA	Germany	Late Shipment
38	78	3	Shipper ZHISN	USA	Late Shipment
39	78	2	Shipper ETYNR	USA	Late Shipment
40	78	1	Shipper GVSUA	USA	Late Shipment
41	77	6	Shipper ZHISN	USA	Late Shipment
42	77	4	Shipper ETYNR	USA	Late Shipment
43	76	18	Shipper ZHISN	Belgium	Late Shipment
44	76	8	Shipper ETYNR	Belgium	Late Shipment
45	76	2	Shipper ETYNR	Belgium	Shipped Early
46	76	1	Shipper GVSUA	Belgium	Late Shipment

Query executed successfully. | localhost,12001 (15.0 RTM) | sa (58) | Northwinds2020TSQLV6 | 00:00:00 | 271 rows

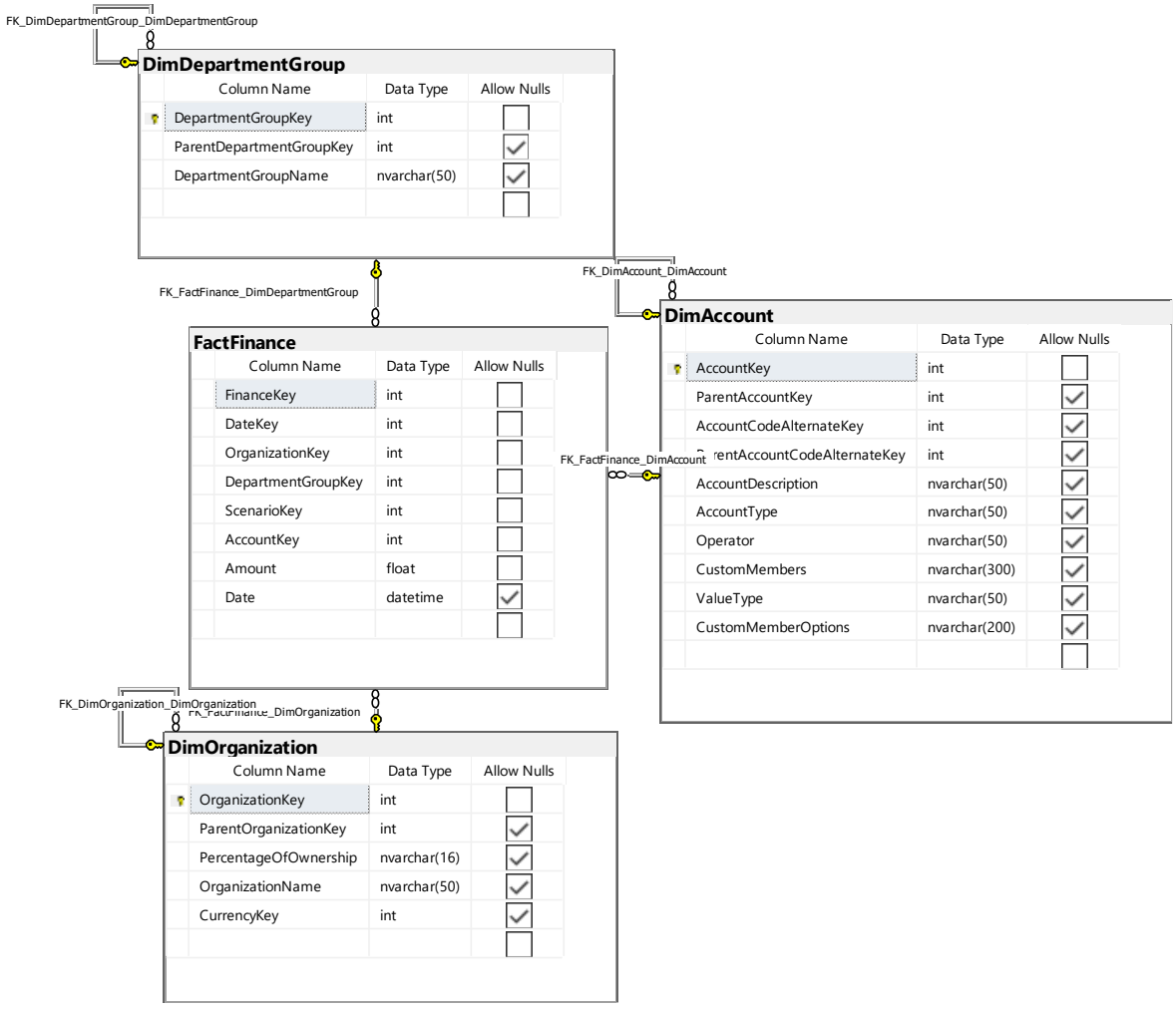
Sample JSON output with total number of rows returned(271)

```
SELECT DISTINCT
       c.CustomerId,
       SUM(S.ShipperId) AS Shipments,
       S.ShipperCompanyName AS ShipperCompany,
       c.CustomerCountry,
       dbo.shipTime(O.OrderId, O.RequiredDate, O.ShipToDate) AS ShippingTime
FROM Sales.[Order] AS O
    INNER JOIN Sales.Shipper AS S
        ON S.ShipperId = O.ShipperId
    INNER JOIN Sales.Customer AS c
        ON c.CustomerId = O.CustomerId
WHERE dbo.shipTime(O.OrderId, O.RequiredDate, O.ShipToDate) IS NOT NULL
GROUP BY dbo.shipTime(O.OrderId, O.RequiredDate, O.ShipToDate),
         c.CustomerId,
         S.ShipperCompanyName,
         c.CustomerCountry
ORDER BY c.CustomerId DESC,
         Shipments DESC
FOR JSON PATH, ROOT('ShippingTimeStatus'), INCLUDE_NULL_VALUES;
```

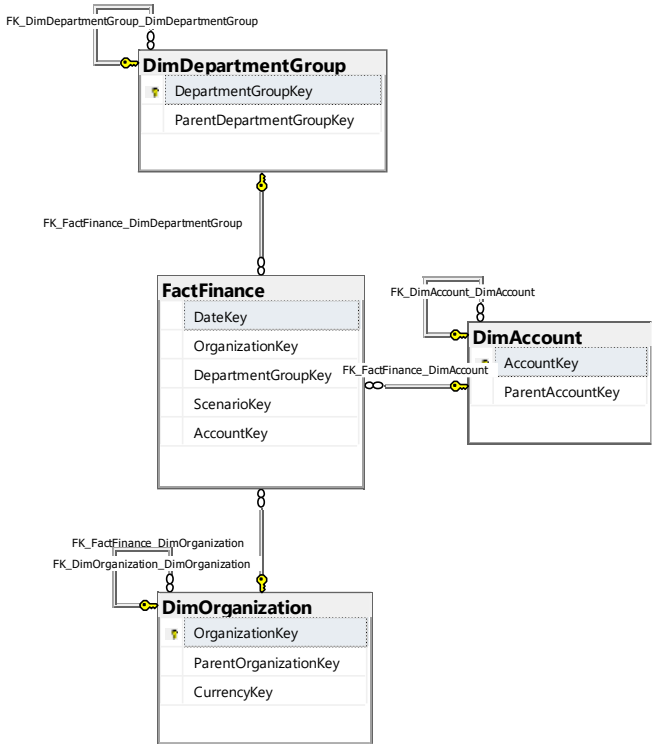
Proposition 20: Find the top 10 with ties results for the average amount of actual expenses on 12/29/2010 and include the organization, department, account description and type of account

Explanation: Find the top 10 with ties results for the average amount of actual expenses on 12/29/2010 and include the organization, department, account description and type of account

Columns from Standard View



Columns from Key View



Columns from their respective table in the select clause

Table Name	Column Name
FactFinance	AccountKey FinanceKey DateKey Date
DimAccount	AccountDescription AccountType
DimOrganization	OrganizationName
DepartmentGroup	DepartmentGroupName
DerivedColumn	AvgAmount(Dbo.SumActualAmount function)

Order By

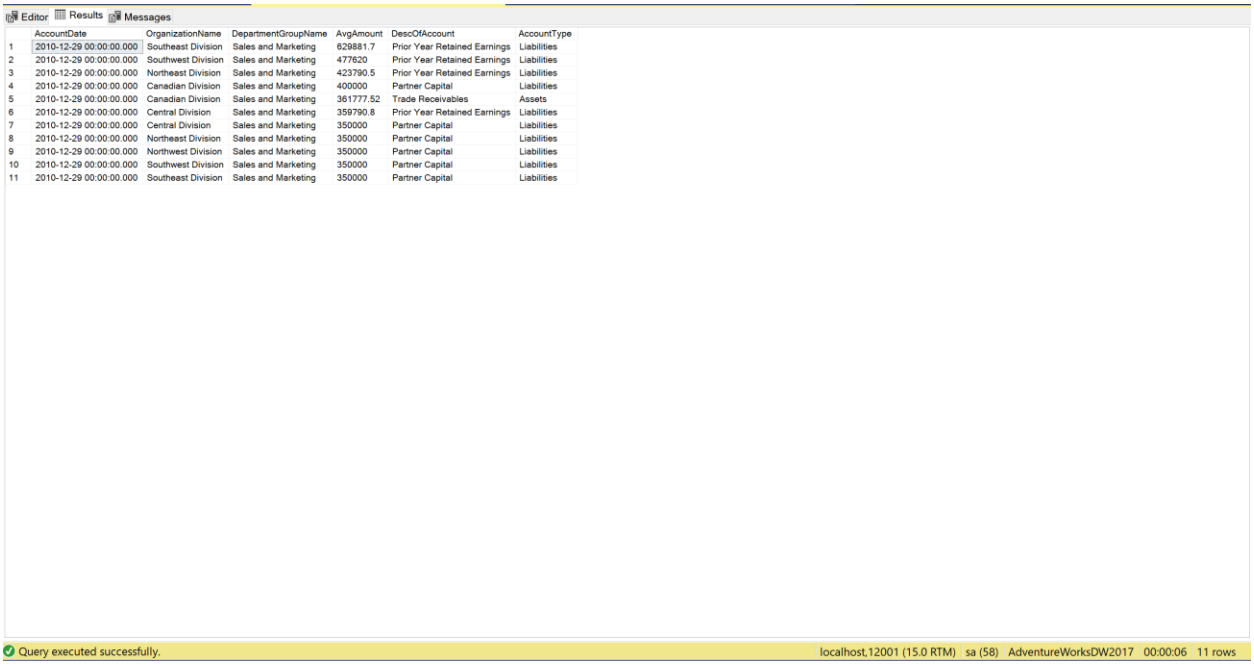
Table Name	Column Name	Sort
DerivedColumn	AvgAmount	DESC

Problem Solving Query

```
USE AdventureWorksDW2017;
SET ANSI_NULLS ON;
GO
SET QUOTED_IDENTIFIER ON;
```

```
GO
CREATE FUNCTION [dbo].SumActualAmount
(
    @accountkey INT,
    @financekey INT,
    @date INT
)
RETURNS FLOAT
AS
BEGIN
    DECLARE @Amount FLOAT;

    SELECT @Amount = SUM(F.Amount)
    FROM dbo.FactFinance AS F
    WHERE F.AccountKey = @accountkey
          AND F.FinanceKey = @financekey
          AND F.ScenarioKey = 1
          AND F.DateKey = @date;
    IF (@Amount IS NULL)
        SET @Amount = 0;


    RETURN @Amount;
END;
GO


USE AdventureWorksDW2017;

SELECT TOP 10 WITH TIES F.Date AS AccountDate,
       O.OrganizationName,
       DG.DepartmentGroupName,
       AVg(dbo.SumActualAmount(F.AccountKey, F.FinanceKey, F.DateKey)) AS AvgAmount,
       A.AccountDescription AS DescOfAccount,
       A.AccountType AS AccountType
FROM dbo.FactFinance AS F
    INNER JOIN dbo.DimAccount AS A
        ON A.AccountKey = F.AccountKey
    INNER JOIN dbo.DimOrganization AS O
        ON O.OrganizationKey = F.OrganizationKey
    INNER JOIN dbo.DimDepartmentGroup AS DG
        ON DG.DepartmentGroupKey = F.DepartmentGroupKey
WHERE dbo.AvgActualAmount(F.AccountKey, F.FinanceKey, F.DateKey) > 0
      AND F.DateKey = 20101229
GROUP BY F.Date,
         O.OrganizationName,
         DG.DepartmentGroupName,
         F.FinanceKey,
         A.AccountDescription,
         A.AccountType
ORDER BY AvgAmount DESC;
```

Sample Relational output with total number of rows returned(11)



Editor | Results | Messages

	AccountDate	OrganizationName	DepartmentGroupName	AvgAmount	DescOfAccount	AccountType
1	2010-12-29 00:00:00.000	Southeast Division	Sales and Marketing	629881.7	Prior Year Retained Earnings	Liabilities
2	2010-12-29 00:00:00.000	Southwest Division	Sales and Marketing	477620	Prior Year Retained Earnings	Liabilities
3	2010-12-29 00:00:00.000	Northeast Division	Sales and Marketing	423790.5	Prior Year Retained Earnings	Liabilities
4	2010-12-29 00:00:00.000	Canadian Division	Sales and Marketing	400000	Partner Capital	Liabilities
5	2010-12-29 00:00:00.000	Canadian Division	Sales and Marketing	361777.52	Trade Receivables	Assets
6	2010-12-29 00:00:00.000	Central Division	Sales and Marketing	359790.8	Prior Year Retained Earnings	Liabilities
7	2010-12-29 00:00:00.000	Central Division	Sales and Marketing	350000	Partner Capital	Liabilities
8	2010-12-29 00:00:00.000	Northeast Division	Sales and Marketing	350000	Partner Capital	Liabilities
9	2010-12-29 00:00:00.000	Northwest Division	Sales and Marketing	350000	Partner Capital	Liabilities
10	2010-12-29 00:00:00.000	Southwest Division	Sales and Marketing	350000	Partner Capital	Liabilities
11	2010-12-29 00:00:00.000	Southeast Division	Sales and Marketing	350000	Partner Capital	Liabilities

Query executed successfully. localhost,12001 (15.0 RTM) sa (58) AdventureWorksDW2017 00:00:06 11 rows

Sample JSON output with total number of rows returned(11)

```
USE AdventureWorksDW2017;

SELECT TOP 10 WITH TIES
       F.Date AS AccountDate,
       O.OrganizationName,
       DG.DepartmentGroupName,
       AVG(dbo.SumActualAmount(F.AccountKey, F.FinanceKey, F.DateKey)) AS AvgAmount,
       A.AccountDescription AS DescOfAccount,
       A.AccountType AS AccountType
FROM dbo.FactFinance AS F
    INNER JOIN dbo.DimAccount AS A
        ON A.AccountKey = F.AccountKey
    INNER JOIN dbo.DimOrganization AS O
        ON O.OrganizationKey = F.OrganizationKey
    INNER JOIN dbo.DimDepartmentGroup AS DG
        ON DG.DepartmentGroupKey = F.DepartmentGroupKey
WHERE dbo.AvgActualAmount(F.AccountKey, F.FinanceKey, F.DateKey) > 0
      AND F.DateKey = 20101229
GROUP BY F.Date,
         O.OrganizationName,
         DG.DepartmentGroupName,
         F.FinanceKey,
         A.AccountDescription,
         A.AccountType
ORDER BY AvgAmount DESC
FOR JSON PATH, ROOT('Top10HighestAmounts'), INCLUDE_NULL_VALUES;
```


JSON Viewer
JSON
Top10HighestAmounts
0
1
2
3
4
5
6
7
8
9
10
new 1
"AccountDate": "2010-12-29T00:00:00",
"OrganizationName": "Northwest Division",
"DepartmentGroupName": "Sales and Marketing",
"AvgAmount": 350000.0,
"DescOfAccount": "Partner Capital",
"AccountType": "Liabilities"
},
{
"AccountDate": "2010-12-29T00:00:00",
"OrganizationName": "Southwest Division",
"DepartmentGroupName": "Sales and Marketing",
"AvgAmount": 350000.0,
"DescOfAccount": "Partner Capital",
"AccountType": "Liabilities"
},
{
"AccountDate": "2010-12-29T00:00:00",
"OrganizationName": "Southeast Division",
"DepartmentGroupName": "Sales and Marketing",
"AvgAmount": 350000.0,
"DescOfAccount": "Partner Capital",
"AccountType": "Liabilities"
}
]