GROUP PROJECT #1

DANNY KONG, HAIBO LIU, ERIK KIM, HARJIT LIYAL, JONATHAN ENG, JAMIL KOCACAL, MARLON LOUIS

EFFECTIVENESS OF WORKING IN A GROUP

- Delegation of work
- Extra minds to brainstorm with
- Peer review

TO-DO LIST -ERIK

To-do list

To be completed by: 10/25/2020	Name: Erik Kim
Deadline: 10/25/2020	Date: 10/2/2020

Project 1

% done	Phase	Start By	Original Due By	Revised Due By	Number Of Days	Revision Notes
100%	Planning	10/2/2020	10/2/2020		One	
100%	5 queries	10/2/2020	10/5/2020		Three	
100%	Medium queries	10/5/2020	10/9/2020		Four	
100%	Finish Microsoft Word Document	10/9/2020	10/22/2020		Thirteen	
100%	Finish all queries	10/9/2020	10/16/2020		Seven	
100%	JDBC Library	10/10/2020	10/14/2020		Four	
100%	Finish recording individual portion	10/16/2020	10/19/2020		Three	
100%	JDBC Recording	10/16/2020	10/21/2020	10/22/2020	Two	extended to meet Professor Heller's Standards in the previous meeting
100%	Check Everything is Complete	10/22/2020	10/23/2020		One	
0%						

TO-DO LIST -HARJIT

To-do list

To be completed by: 10/25/2020	Name: Harjit Liyal
Deadline: 10/25/2020	Date: 10/2/2020

Project 1

% done	Phase	Start By	Original Due By	Revised Due By	Number Of Days	Revision Notes
100%	Planning	10/2/2020	10/2/2020		One	
100%	Create meeting notes and take attendance for each meeting	10/2/2020	10/23/2020		Twenty-One	
100%	5 queries	10/2/2020	10/5/2020		Three	
100%	Medium queries	10/5/2020	10/9/2020		Four	
100%	Finish all queries	10/9/2020	10/16/2020		Seven	
100%	Finish Microsoft Word Document	10/9/2020	10/22/2020		Thirteen	
100%	Finish recording portions individual	10/16/2020	10/19/2020		Three	
100%	JDBC Recording	10/16/2020	10/21/2020	10/22/2020	Two	extended to meet Professor Heller's Standards in the previous meeting
0%						

TO-DO LIST -JAMIL

To-do list

To be completed by: 10/25/2020	Name: Jamil Kocacal
Deadline: 10/25/2020	Date: 10/2/2020

Project 1

% done	Phase	Start By	Original Due By	Revised Due By	Number Of Days	Revision Notes
100%	Planning	10/2/2020	10/2/2020		One	
100%	Create tasks within the to-do list	10/2/2020	10/23/2020		Twenty-One	
100%	5 queries	10/2/2020	10/5/2020		Three	
100%	Medium queries	10/5/2020	10/9/2020		Four	
100%	Finish all queries	10/9/2020	10/16/2020		Seven	
100%	Finish Microsoft Word Document	10/9/2020	10/22/2020		Thirteen	
100%	Finish recording Individual Portion	10/16/2020	10/19/2020		Three	
100%	JDBC Recording	10/16/2020	10/21/2020	10/22/2020	Two	extended to meet Professor Heller's Standards in the previous meeting
0%	Follow-up					

TO-DO LIST -DANNY

To-do list

To be completed by: 10/25/2020	Name: Danny Kong
Deadline: 10/25/2020	Date: 10/2/2020

Project 1

% done	Phase	Start By	Original Due By	Revised Due By	Number Of Days	Revision Notes
100%	Planning	10/2/2020	10/2/2020		One	
100%	5 queries	10/2/2020	10/5/2020		Three	
100%	Medium queries	10/5/2020	10/9/2020		Four	
100%	Finish all queries	10/9/2020	10/16/2020		Seven	
100%	Help set up slides for the power point	10/10/2020	10/23/2020		Thirteeen	
100%	Finish recording Individual Portion	10/16/2020	10/19/2020		Three	
100%	Finish Microsoft Word Document	10/9/2020	10/22/2020		Thirteen	
100%	JDBC Recording	10/16/2020	10/21/2020	10/22/2020	Two	extended to meet Professor Heller's Standards in the previous meeting
0%						

TO-DO LIST -JONATHAN

To-do list

To be completed by: 10/25/2020	Name: Jonathan Eng
Deadline: 10/25/2020	Date: 10/2/2020

Project 1

% done	Phase	Start By	Original Due By	Revised Due By	Number Of Days	Revision Notes
100%	Planning	10/2/2020	10/2/2020		One	
100%	Complete Project Planner	10/2/2020	10/23/2020		Twenty-One	
100%	5 queries	10/2/2020	10/5/2020		Three	
100%	Medium queries	10/5/2020	10/9/2020		Four	
100%	Finish all queries	10/9/2020	10/16/2020		Seven	
100%	Finish Microsoft Word Document	10/9/2020	10/22/2020		Thirteen	
100%	Finish recording Individual Portion	10/16/2020	10/19/2020		Three	
100%	JDBC Recording	10/16/2020	10/21/2020	10/22/2020	Five	extended to meet Professor Heller's standards in the previous meeting
100%	Edit JDBC Recording	10/21/2020	10/22/2020	10/23/2020	Three	Extended due to needing to redo the recording and thus need to reedit to meet Professor Heller's standard.
0%						

TO-DO LIST -HAIBO

To-do list

To be completed by:	10/25/2020	Name: Haibo Liu	
Deadline:	10/25/2020	Date: 10/2/2020	

Project 1

% done	Phase	Start By	Original Due By	Revised Due By	Number Of Days	Revision Notes
100%	5 queries	10/5/2020	10/5/2020	10/7/2020	Three	Joined group late required more time to finish five queries
100%	Medium queries	10/5/2020	10/9/2020		Four	
100%	Finish all queries	10/9/2020	10/16/2020		Seven	
100%	Finish Microsoft Word Document	10/9/2020	10/22/2020		Thirteen	
100%	Help set up slides for the power point	10/10/2020	10/23/2020		Thirteen	
100%	Finish recording Individual Portion	10/16/2020	10/19/2020		Three	
100%	JDBC Recording	10/16/2020	10/21/2020	10/22/2020	Five	extended to meet Professor Heller's standards in the previous meeting
0%						

TO-DO LIST -MARLON

To-do list

To be completed by: 10/25/2020	Name: Marlon Louis
Deadline: 10/25/2020	Date: 10/2/2020

Project 1

% done	Phase	Start By	Original Due	Revised Due	Number Of Days	Revision Notes
100%	Catch up on queries	10/13/2020	10/16/2020	10/18/2020	Five	Joined group late neede more time to finish all the queries
100%	Finish Microsoft Word Document	10/16/2020	10/22/2020		Thirteen	
100%	Record Beginning Segment of the power point	10/16/2020	10/23/2020		Seven	
100%	Finish recording Individual Portion	10/16/2020	10/19/2020		Three	
100%	JDBC Recording	10/16/2020	10/21/2020	10/22/2020	Five	extended to meet Professor Heller's standards in the previous meeting
0%						

PROJECT PLAN

Project Planner

Select a period to highlight at right. A legend describing the charting follows.

Period Highlight: 1

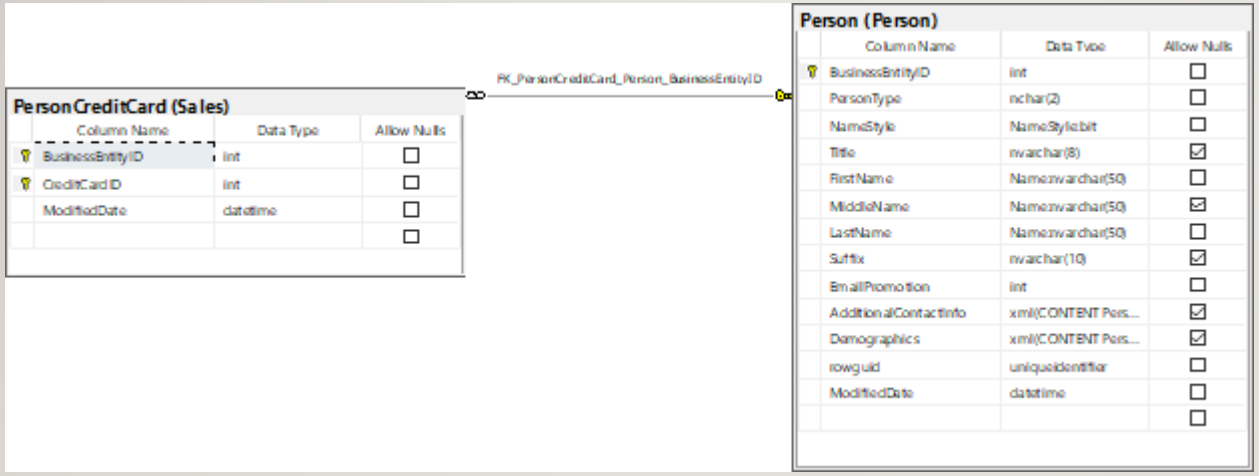
Plan Duration | Actual Start | % Complete | Actual (beyond plan) | % Complete (beyond plan)

Task	Day Start	Amount of Days Planned	Start Day	Days it took	PERCENT COMPLETE
First Meet. Discuss the Assignment & Nominated Leader. Assign jobs to group members (Missing 1 Member Unitl Otherwise Noted)*	1	1	1	1	100%
Assign Task 1 to Each Group Member Finalize Roles for each group member (Task 1: 5 Simple Queries By Next Meet)	1	4	1	4	100%
Check Each Others 5 Queries (Made sure no Queries are too Similar)	5	1	5	1	100%
Assign Task 2 to Each Group Member (Task 2: 7 Medium Queies by Next Meet)	5	5	5	5	100%
Check Each Other 7 Medium Queries (Made sure no Queries were too Similar) (Last Member Joined)*	10	1	10	1	100%
Assign Task 3 to Each Group Member (Task 3: Finish remaining Queries)	10	7	10	7	100%
Discuss and Work on JDBC	10	7	10	11	100%
Discuss Diagram Layout and JSON and assign Task 4 (Task 4: Write Word Document of Queries/Diagrams, Prepare Recordings	17	1	17	1	100%
Discuss JSON and SQL Diagrams; Assign Task 5 (Task 5: Work On Individual Word Submissions of Code Documentation)	18	3	18	3	100%
Continue Fixing JDBC Prepare Powerpoint Presentation Outline	21	1	21	1	100%
Gather All Word Documents Continue Powerpoints Finalize SQL Queries	22	1	22	1	100%
Collect SQL Queries Finalized JDBC Completed Powerpoint Assign Task 6: (Task 6: Record Powerpoint video, and JDBC Video) [Final Task]	23	3	23	3	100%

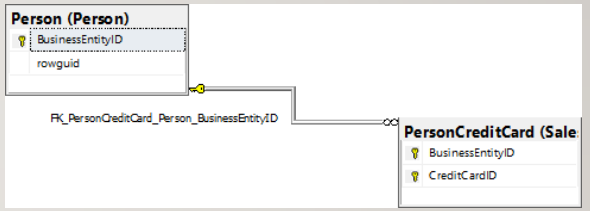
Days: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50

SIMPLE WORST#1 STANDARD/KEY VIEW

Standard View:



Key View:



SIMPLE WORST#1 TABLES

Columns From Tables:

Table Name	Column Names
PersonCreditCard	CreditCardID
Person	BusinessEntityID

Sort Order:

Table Name	Column Name	Sort Order
PersonCreditCard	CreditCardID	ASC

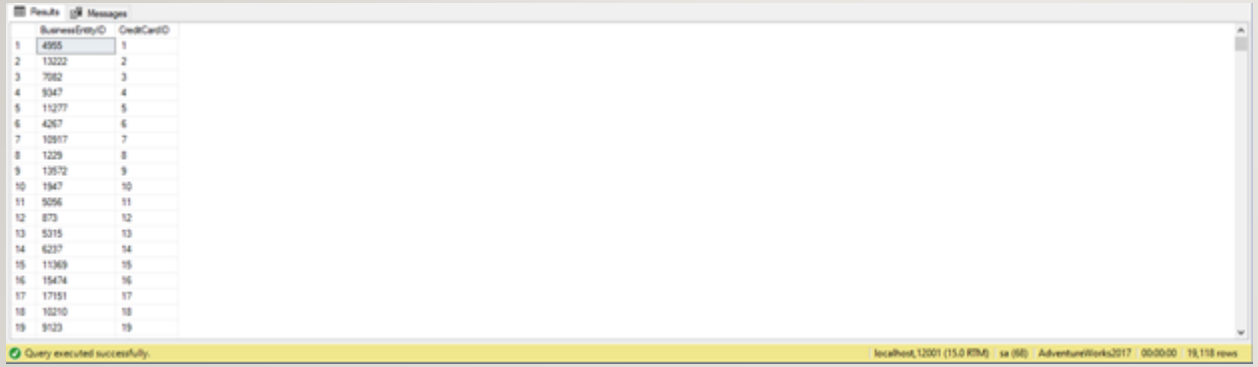
ERIK'S QUERY (SIMPLE) WORST #1

- --Find credit card IDs of all business entities that have used credit cards.

```
USE AdventureWorks2017;
SELECT DISTINCT
       p.BusinessEntityID,
       pc.CreditCardID
FROM Person.Person AS p
    INNER JOIN Sales.PersonCreditCard AS pc
        ON p.BusinessEntityID = pc.BusinessEntityID
ORDER BY pc.CreditCardID;
```

SIMPLE WORST#1 OUTPUT SAMPLE

Relational Output:

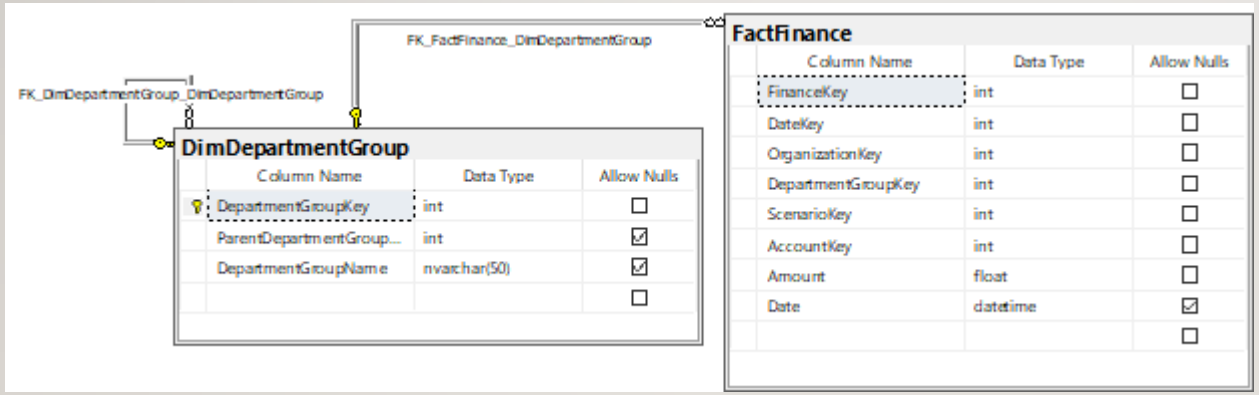


	BusinessEntityID	CreditCardID
1	4955	1
2	13222	2
3	7082	3
4	9347	4
5	11277	5
6	4267	6
7	10917	7
8	1229	8
9	13572	9
10	1947	10
11	5056	11
12	873	12
13	5315	13
14	6237	14
15	11369	15
16	15474	16
17	17151	17
18	10210	18
19	9123	19

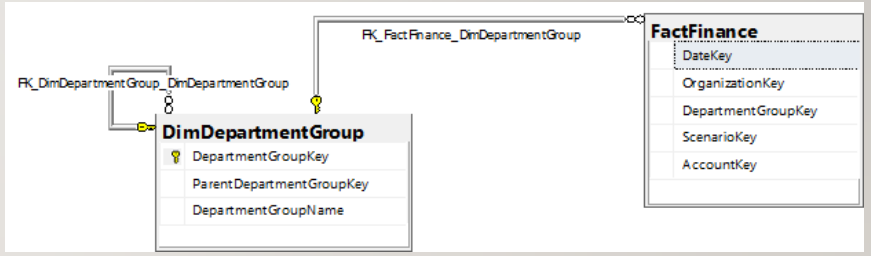
JSON Output:

MEDIUM WORST#2 STANDARD/KEY VIEW

Standard View:



Key View:



MEDIUM WORST#2 TABLES

Columns From Tables:

Table Name	Column Names
FactFinance	DepartmentGroupKey
DimDepartmentGroup	ParentDepartmentGroupKey

Sort Order:

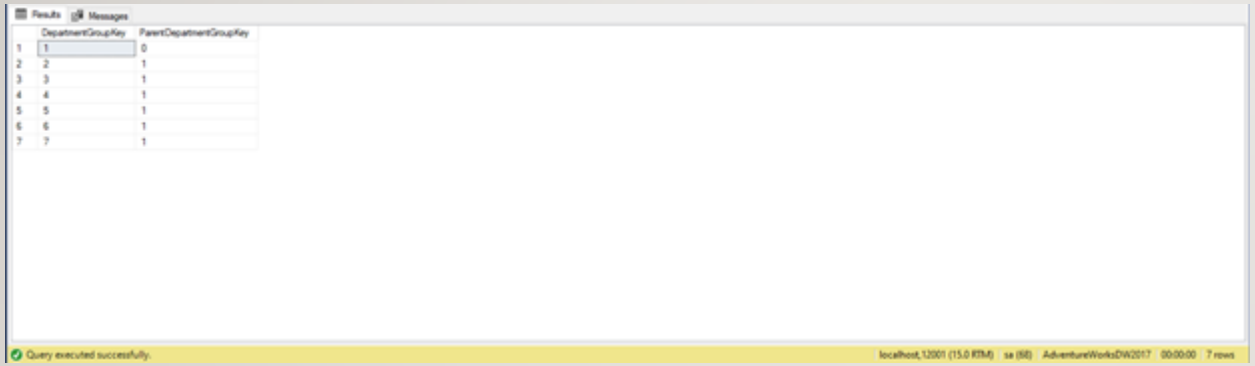
Table Name	Column Name	Sort Order
FactFinance	DepartmentGroupKey	ASC

ERIK'S QUERY (MEDIUM) WORST #2

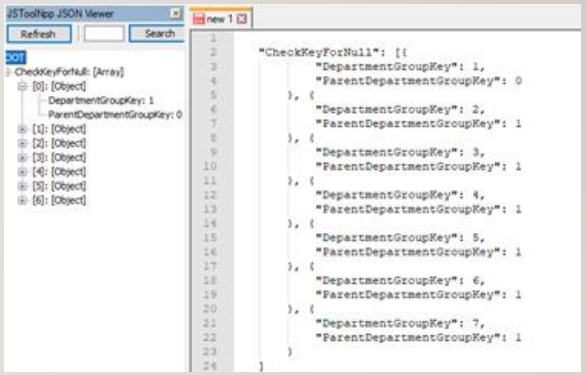
```
--Use the department key in FactFinance to figure out if any ParentDepartmentGroupKey in DimDepartmentGroup are NULL, then
coalesce them to '0'.


USE AdventureWorksDW2017;
SELECT ff.DepartmentGroupKey,
       COALESCE(ddg.ParentDepartmentGroupKey, 0) AS ParentDepartmentGroupKey
FROM dbo.FactFinance AS ff
    INNER JOIN dbo.DimDepartmentGroup AS ddg
        ON ff.DepartmentGroupKey = ddg.DepartmentGroupKey
GROUP BY ff.DepartmentGroupKey,
         ddg.ParentDepartmentGroupKey
ORDER BY ff.DepartmentGroupKey;
```

MEDIUM WORST#2 OUTPUT SAMPLE

Relational Output:

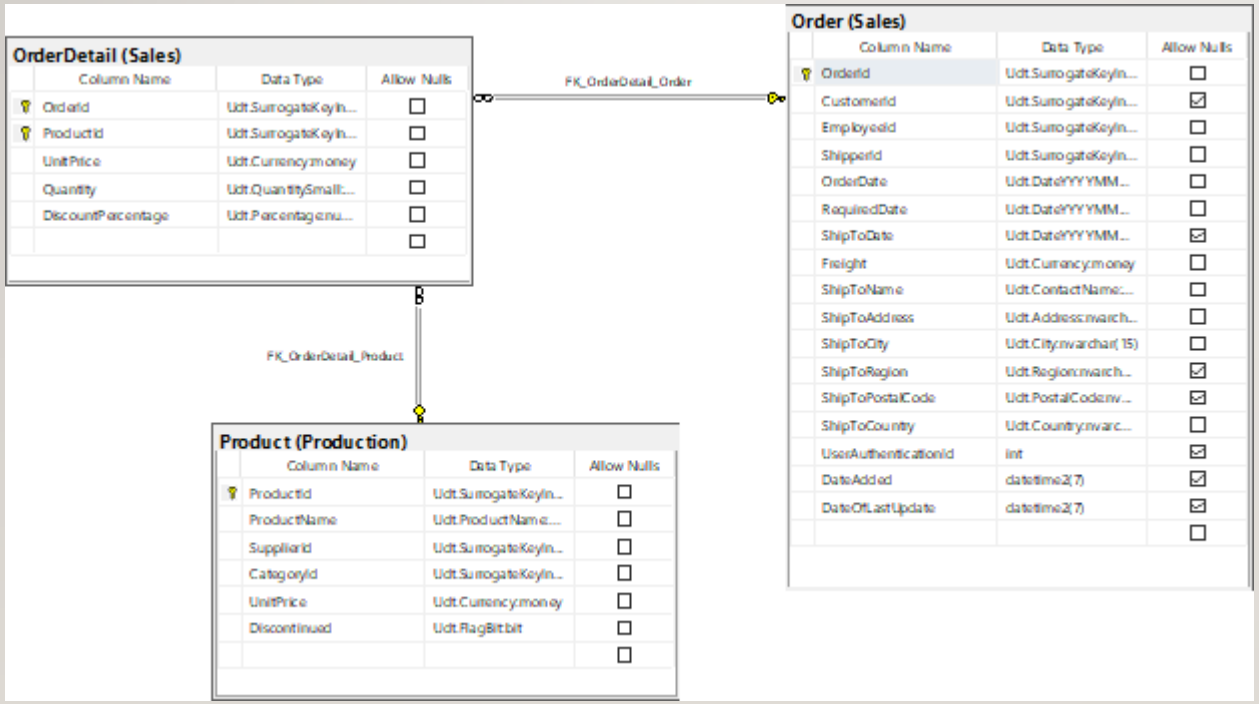


JSON Output:

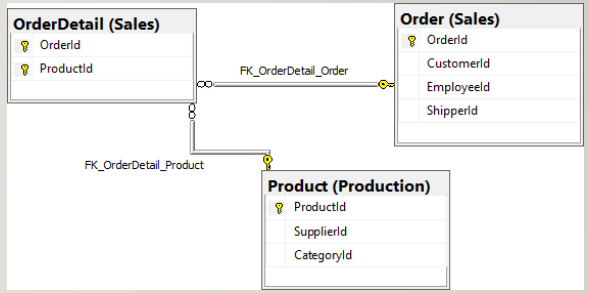


COMPLEX WORST#3 STANDARD/KEY VIEW

Standard View:



Key View:



COMPLEX WORST#3 TABLES

Columns From Tables:

Table Name	Column Names
Order	OrderId
OrderDetail	ProductId
Product	ProductName

Sort Order:

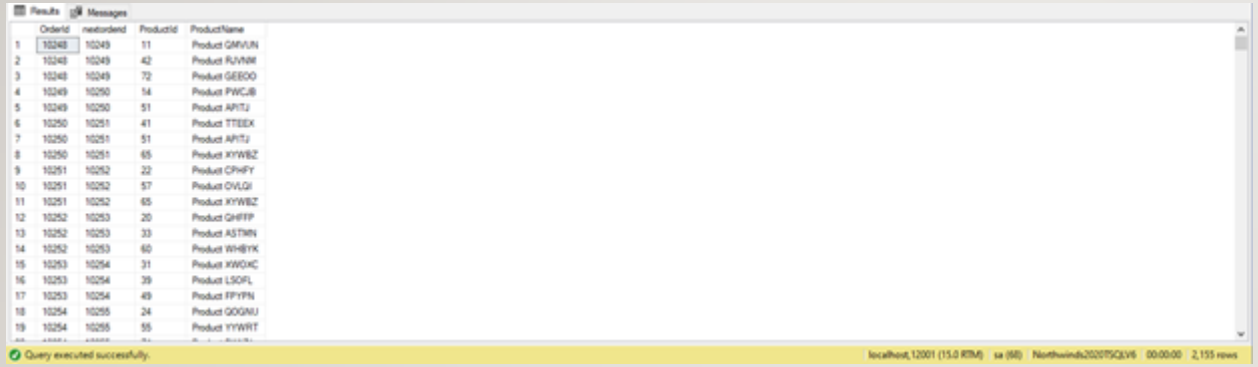
Table Name	Column Name	Sort Order
Order	OrderId	ASC

ERIK'S QUERY (COMPLEX) WORST #3

```
--Create a column for the nextorderid in Sales.[Order], display ProductId from Sales.OrderDetail and ProductName from Production.Product.

USE Northwinds2020TSQLV6;
GO

CREATE OR ALTER FUNCTION Sales.udf_NextOrder
(
   @currentorder INT
)
RETURNS INT
BEGIN
   DECLARE @nextorder INT;
   SELECT @nextorder = @currentorder + 1;
   RETURN @nextorder;
```

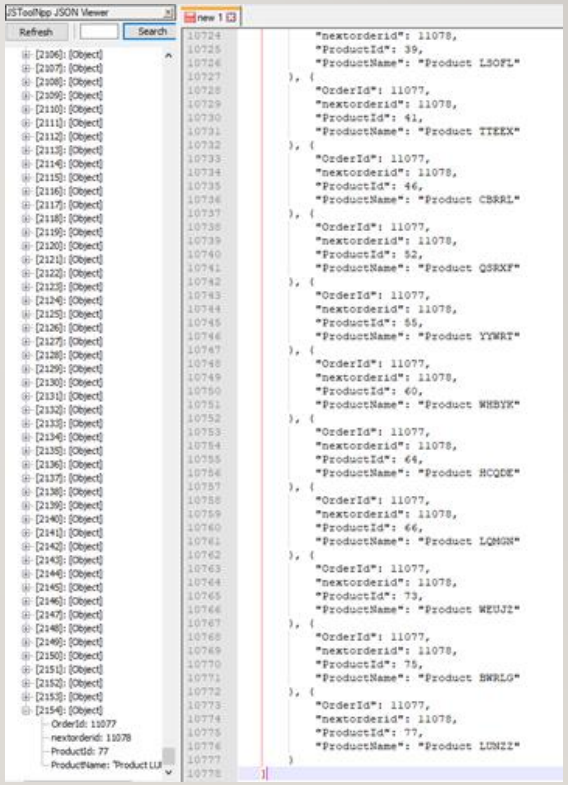
COMPLEX WORST#3 OUTPUT SAMPLE

Relational Output:



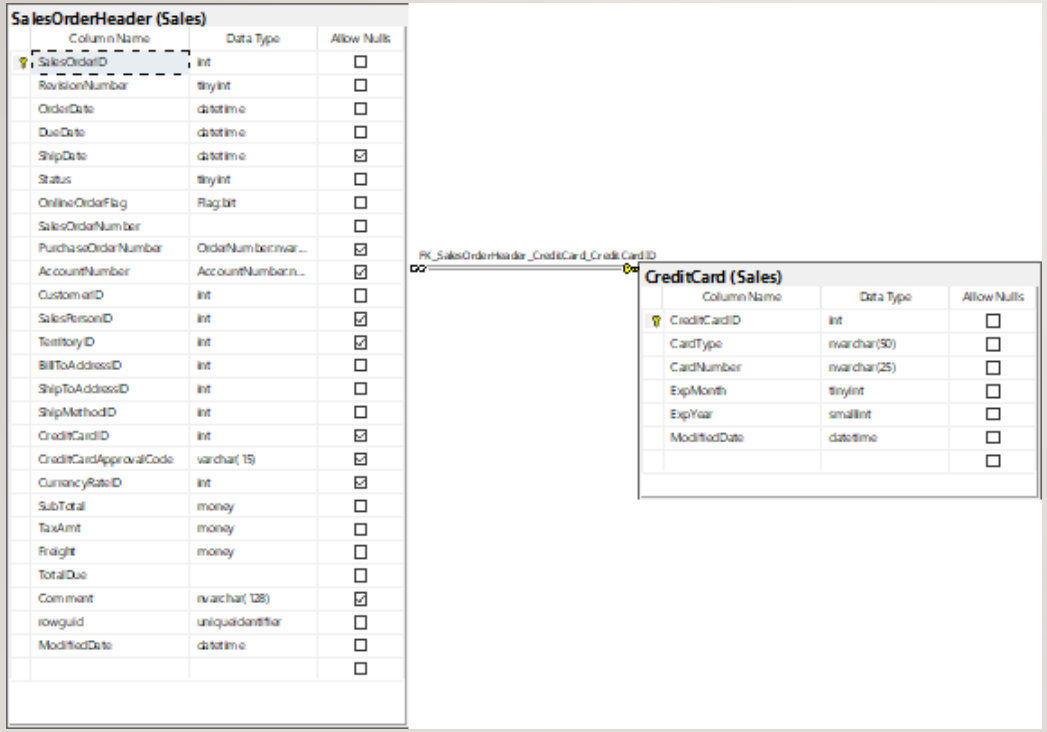
	OrderId	nextorderid	ProductId	ProductName
1	10248	10249	11	Product GMVUN
2	10248	10249	42	Product RJVNW
3	10248	10249	72	Product GEEOO
4	10249	10250	14	Product PWCJB
5	10249	10250	51	Product APITJ
6	10250	10251	41	Product TTEEX
7	10250	10251	51	Product APITJ
8	10250	10251	65	Product KYWBZ
9	10251	10252	22	Product CPHFY
10	10251	10252	57	Product OVLQI
11	10251	10252	65	Product KYWBZ
12	10252	10253	20	Product GHFFP
13	10252	10253	33	Product ASTMN
14	10252	10253	60	Product WHBYK
15	10253	10254	31	Product KWOXC
16	10253	10254	39	Product LSOFL
17	10253	10254	49	Product FPYPN
18	10254	10255	24	Product QOGNU
19	10254	10255	55	Product YYWRT

JSON Output:

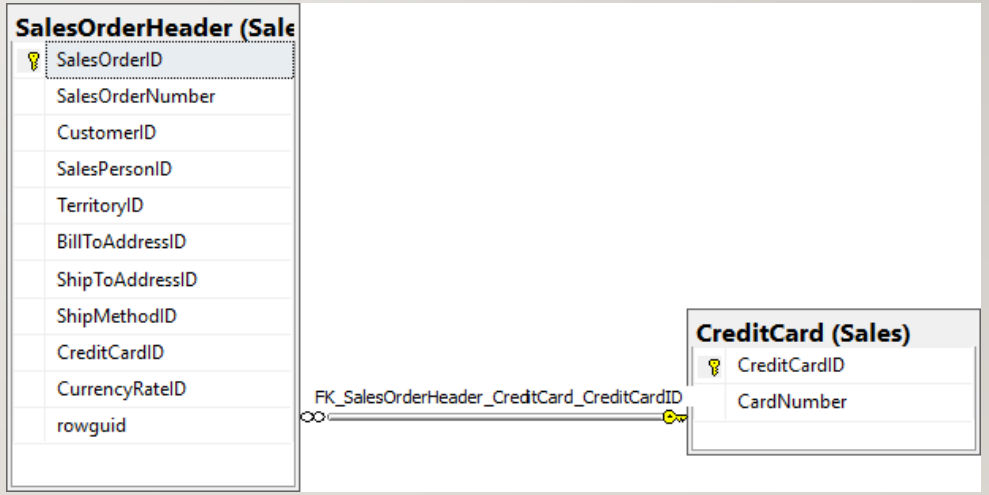


SIMPLE BEST#1 STANDARD/KEY VIEW

Standard View:



Key View:



SIMPLE BEST#1 TABLES

Columns From Tables:

Table Name	Column Names
CreditCard	CreditCardID
SalesOrderHeader	OrderDate DueDate ShipDate

Sort Order:

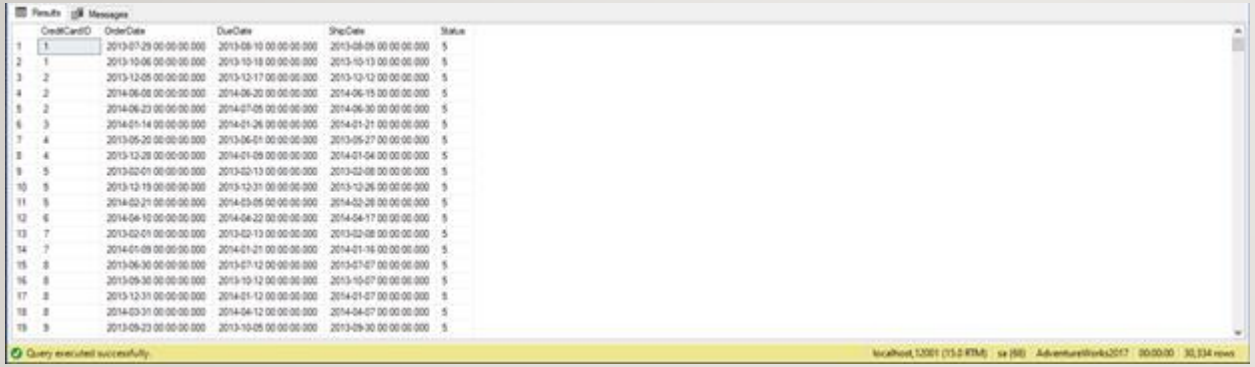
Table Name	Column Name	Sort Order
SalesOrderHeader	CreditCardID OrderDate	ASC ASC

ERIK'S QUERY (SIMPLE) BEST #1

```
--Using credit card ID, find SalesOrderID, OrderDate, DueDate, ShipDate and Status.


USE AdventureWorks2017;
SELECT cc.CreditCardID,
       oh.OrderDate,
       oh.DueDate,
       oh.ShipDate,
       oh.[Status]
FROM Sales.CreditCard AS cc
    INNER JOIN Sales.SalesOrderHeader AS oh
        ON cc.CreditCardID = oh.CreditCardID
ORDER BY oh.CreditCardID, oh.OrderDate;
```

SIMPLE BEST#1 OUTPUT SAMPLE

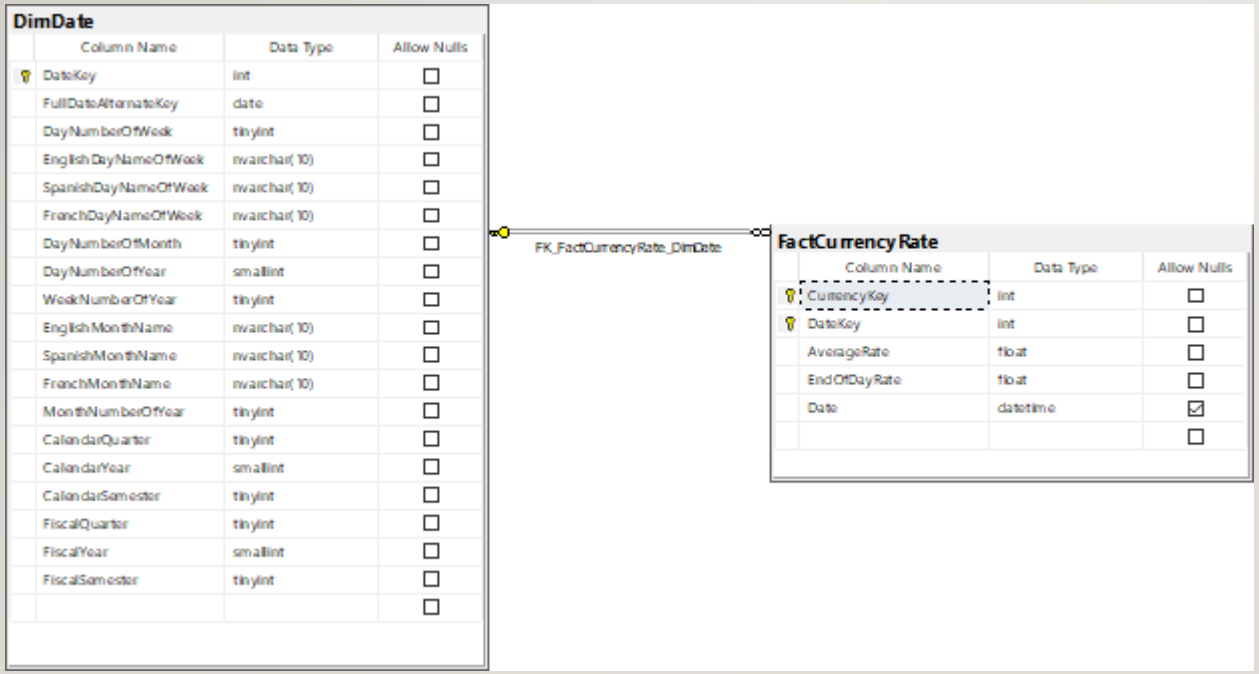
Relational Output:



JSON Output:

MEDIUM BEST#2 STANDARD/KEY VIEW

Standard View:



DimDate

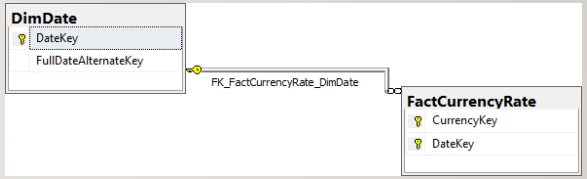
Column Name	Data Type	Allow Nulls
DateKey	int	☐
FullDateAlternateKey	date	☐
DayNumberOfWeek	tinyint	☐
EnglishDayNameOfWeek	nvarchar(10)	☐
SpanishDayNameOfWeek	nvarchar(10)	☐
FrenchDayNameOfWeek	nvarchar(10)	☐
DayNumberOfMonth	tinyint	☐
DayNumberOfYear	smallint	☐
WeekNumberOfYear	tinyint	☐
EnglishMonthName	nvarchar(10)	☐
SpanishMonthName	nvarchar(10)	☐
FrenchMonthName	nvarchar(10)	☐
MonthNumberOfYear	tinyint	☐
CalendarQuarter	tinyint	☐
CalendarYear	smallint	☐
CalendarSemester	tinyint	☐
FiscalQuarter	tinyint	☐
FiscalYear	smallint	☐
FiscalSemester	tinyint	☐
		☐

FK_FactCurrencyRate_DimDate

FactCurrencyRate

Column Name	Data Type	Allow Nulls
CurrencyKey	int	☐
DateKey	int	☐
AverageRate	float	☐
EndOfDayRate	float	☐
Date	datetime	☑
		☐

Key View:



MEDIUM BEST#2 TABLES

Columns From Tables:

Table Name	Column Names
DimDate	SpanishMonthName SpanishDayNameofWeek CalendarYear
FactCurrencyRate	DateKey

Sort Order:

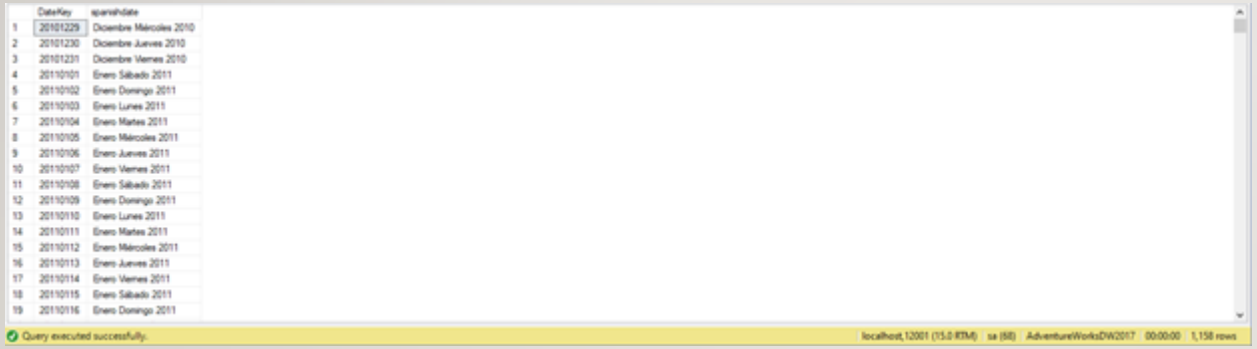
Table Name	Column Name	Sort Order
FactCurrencyRate	DateKey	ASC

ERIK'S QUERY (MEDIUM) BEST #2

```
--Using the datekey from FactCurrencyRate, find the spanish version of the date from DimDate.


USE AdventureWorksDW2017;
SELECT fcr.DateKey,
       CONCAT(dd.SpanishMonthName, ' ', dd.SpanishDayNameOfWeek, ' ', dd.CalendarYear) AS spanishdate
FROM dbo.DimDate AS dd
    INNER JOIN dbo.FactCurrencyRate AS fcr
        ON dd.DateKey = fcr.DateKey
GROUP BY fcr.DateKey,
         dd.SpanishMonthName,
         dd.SpanishDayNameOfWeek,
         dd.CalendarYear
ORDER BY fcr.DateKey;
```

MEDIUM BEST#2 OUTPUT SAMPLE

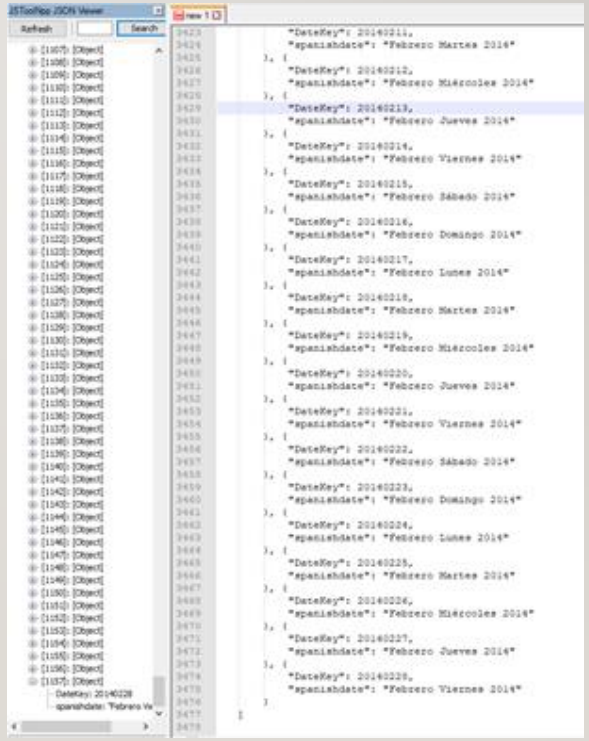
Relational Output:



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

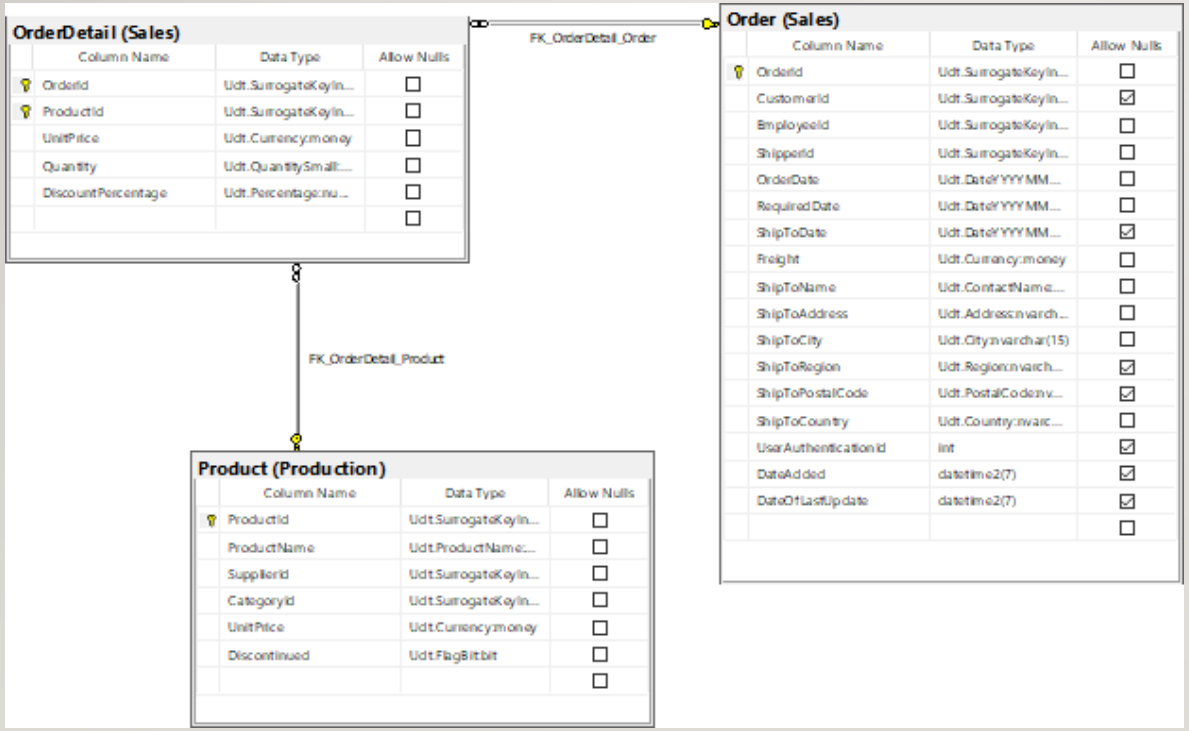
Query executed successfully. localhost,12001 (15.0 RTM) sa (58) AdventureWorksDW2017 00:00:00 1,158 rows

JSON Output:

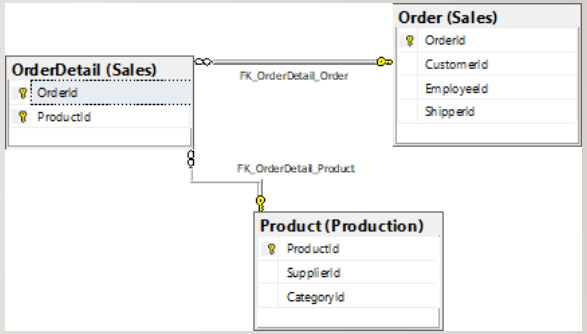


COMPLEX BEST#3 STANDARD KEY/VIEW

Standard View:



Key View:



COMPLEX BEST#3 TABLES

Columns From Tables:

Table Name	Column Names
Order	OrderId
OrderDetail	ProductId Quantity DiscountPercentage
Product	ProductName UnitPrice

Sort Order:

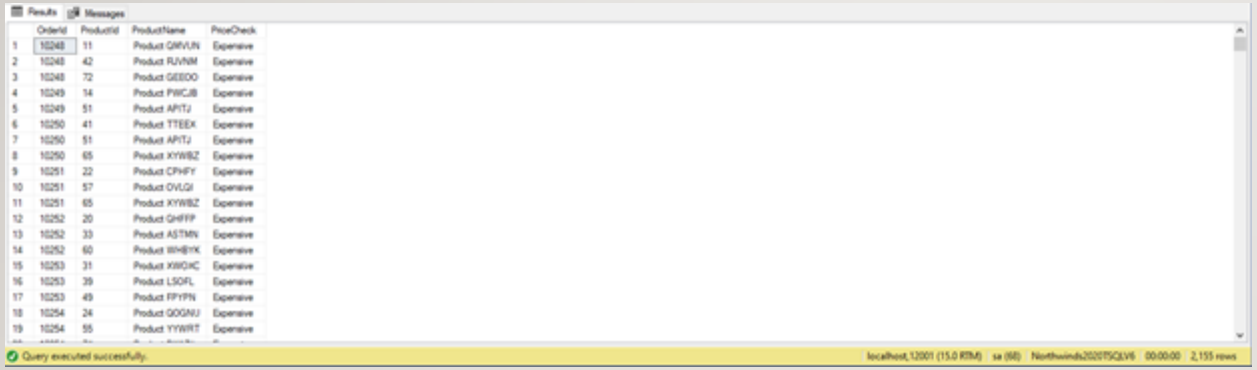
Table Name	Column Name	Sort Order
Order	OrderId	ASC

ERIK'S QUERY (COMPLEX) BEST #3

```
--Create a scalar function that categorizes totalunitprice as cheap or expensive from Production.Product, showing OrderID
from Sales.[Order] and ProductID from Sales.OrderDetail

USE Northwinds2020TSQLV6;

GO

CREATE OR ALTER FUNCTION Sales.udf_PriceCheck
(
   @totalunitprice MONEY
)
RETURNS NVARCHAR(20)
AS
BEGIN
   DECLARE @result NVARCHAR(20);
```

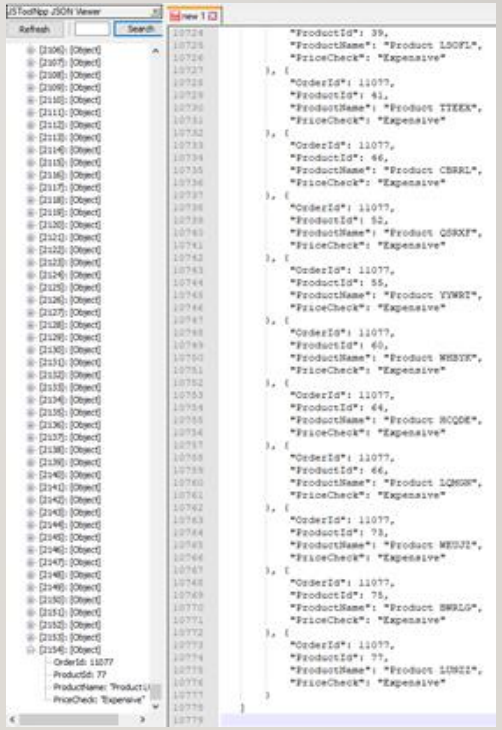
COMPLEX BEST#3 OUTPUT SAMPLE

Relational Output:



	OrderId	ProductId	ProductName	PriceCheck
1	10248	11	Product QMVUN	Expensive
2	10248	42	Product RUVNM	Expensive
3	10248	72	Product GEEOO	Expensive
4	10249	14	Product FWCJB	Expensive
5	10249	51	Product APITJ	Expensive
6	10250	41	Product TTEEX	Expensive
7	10250	51	Product APITJ	Expensive
8	10250	65	Product XYWBZ	Expensive
9	10251	22	Product CPHFY	Expensive
10	10251	57	Product OVLQI	Expensive
11	10251	65	Product XYWBZ	Expensive
12	10252	20	Product GHFFP	Expensive
13	10252	33	Product ASTMN	Expensive
14	10252	60	Product IRHBYK	Expensive
15	10253	31	Product XWOXC	Expensive
16	10253	39	Product LSOFL	Expensive
17	10253	49	Product FPYPN	Expensive
18	10254	24	Product QOGNU	Expensive
19	10254	55	Product YYWRT	Expensive

JSON Output:

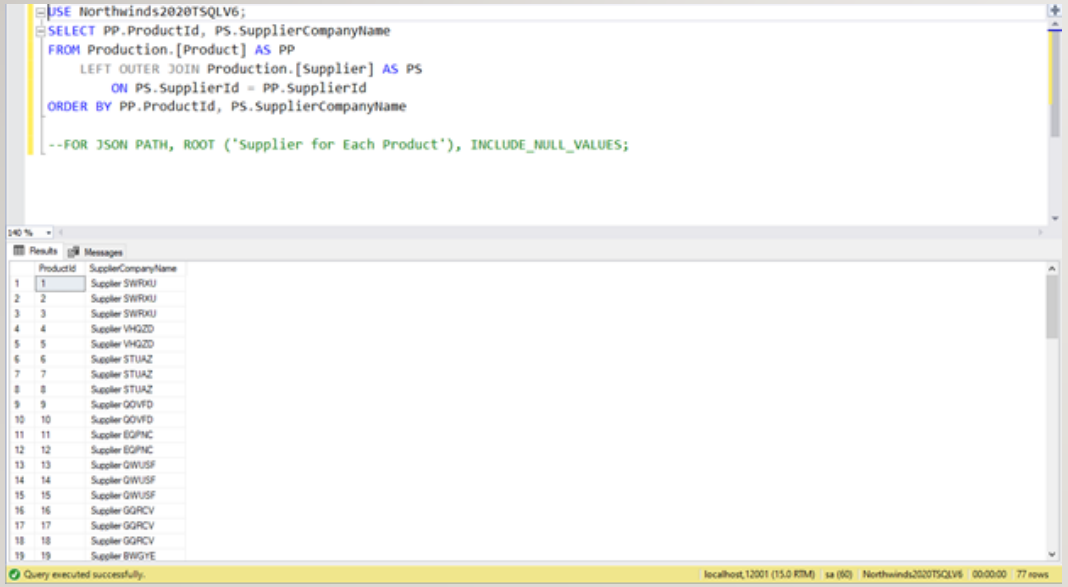


JONATHAN'S QUERY (SIMPLE) - BEST #1 QUERY

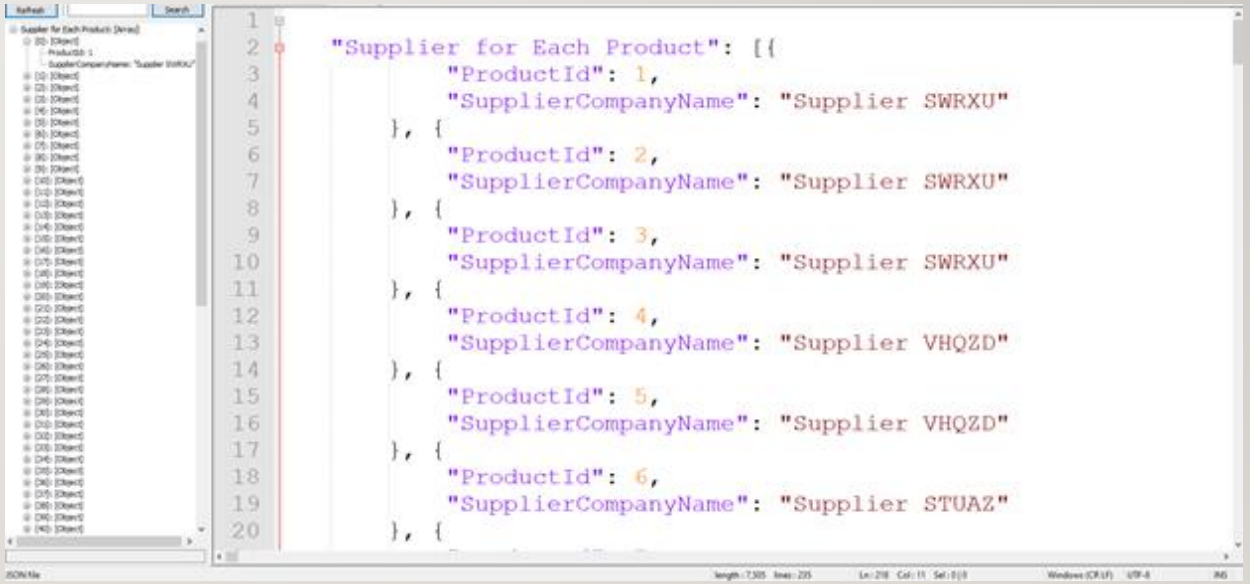
```
-- Show the Supplier for each Product

USE Northwinds2020TSQLV6;

SELECT PP.ProductId, PS.SupplierCompanyName

FROM Production.[Product] AS PP

        LEFT OUTER JOIN Production.[Supplier] AS PS

                ON PS.SupplierId = PP.SupplierId

ORDER BY PP.ProductId, PS.SupplierCompanyName
```

JONATHAN'S QUERY (SIMPLE) - BEST #1 OUTPUTS

Relational Output:

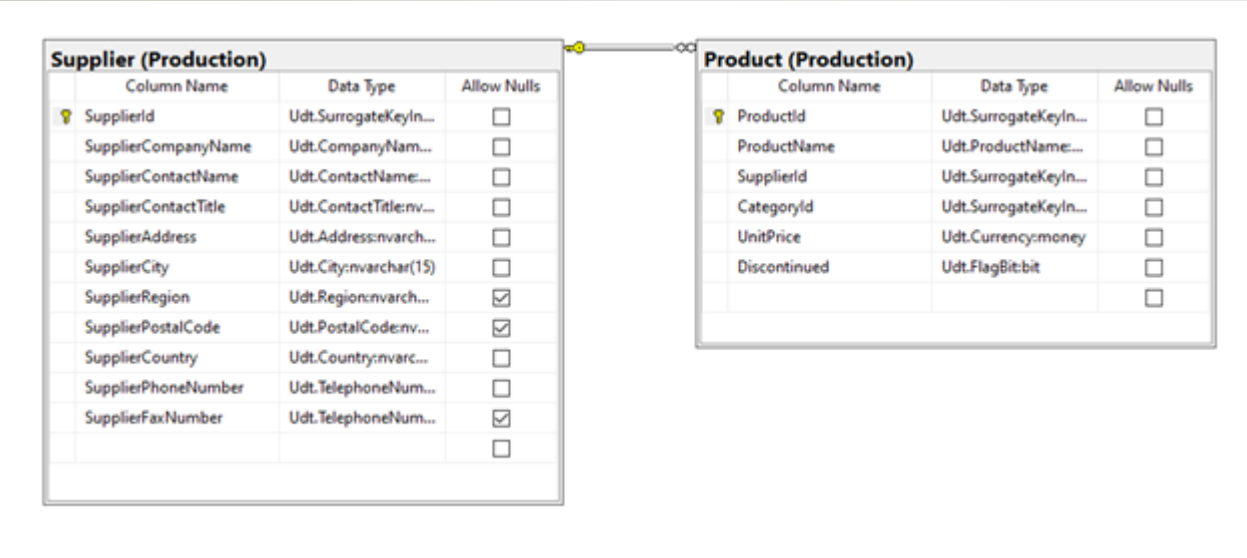


JSON Output:

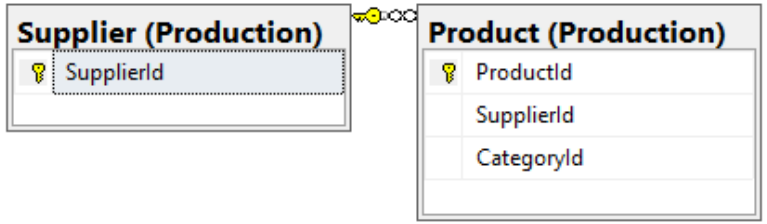


JONATHAN'S QUERY (SIMPLE) - BEST #1 VIEWS

Standard View



Key View



JONATHAN'S QUERY (SIMPLE) - BEST #1 TABLES

Table Name	Column Name
Product	ProductId
Supplier	SupplierCompanyName

Order By

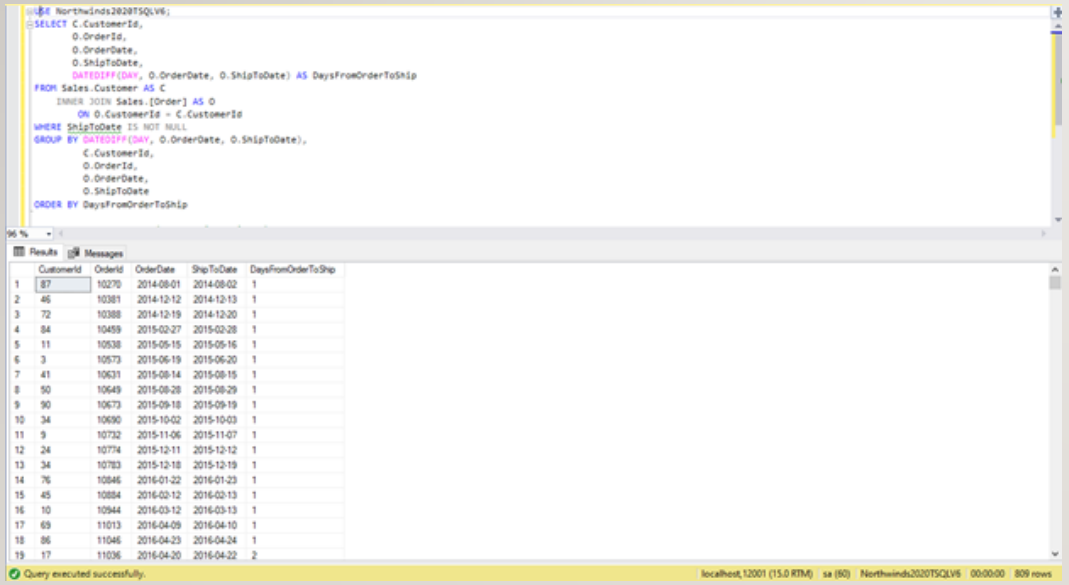
Table Name	Column Name	Sort Order
Product, Supplier	ProductId, SupplierCompanyName	ASC, ASC

JONATHAN'S QUERY (MEDIUM) - BEST #2 QUERY

```
-- Show the OrderDate and ShipToDate and Find
    the amount of days it will take until arrival

USE Northwinds2020TSQLV6;

SELECT C.CustomerId,

      O.OrderId,

      O.OrderDate,

      O.ShipToDate,

      DATEDIFF(DAY, O.OrderDate, O.ShipToDate)
    AS DaysFromOrderToShip

FROM Sales.Customer AS C

   INNER JOIN Sales.[Order] AS O

      ON O.CustomerId = C.CustomerId

WHERE ShipToDate IS NOT NULL

GROUP BY
    DATEDIFF(DAY, O.OrderDate, O.ShipToDate),

      C.CustomerId,

      O.OrderId,

      O.OrderDate,

      O.ShipToDate

ORDER BY DaysFromOrderToShip
```

JONATHAN'S QUERY (MEDIUM) - BEST #2 OUTPUTS

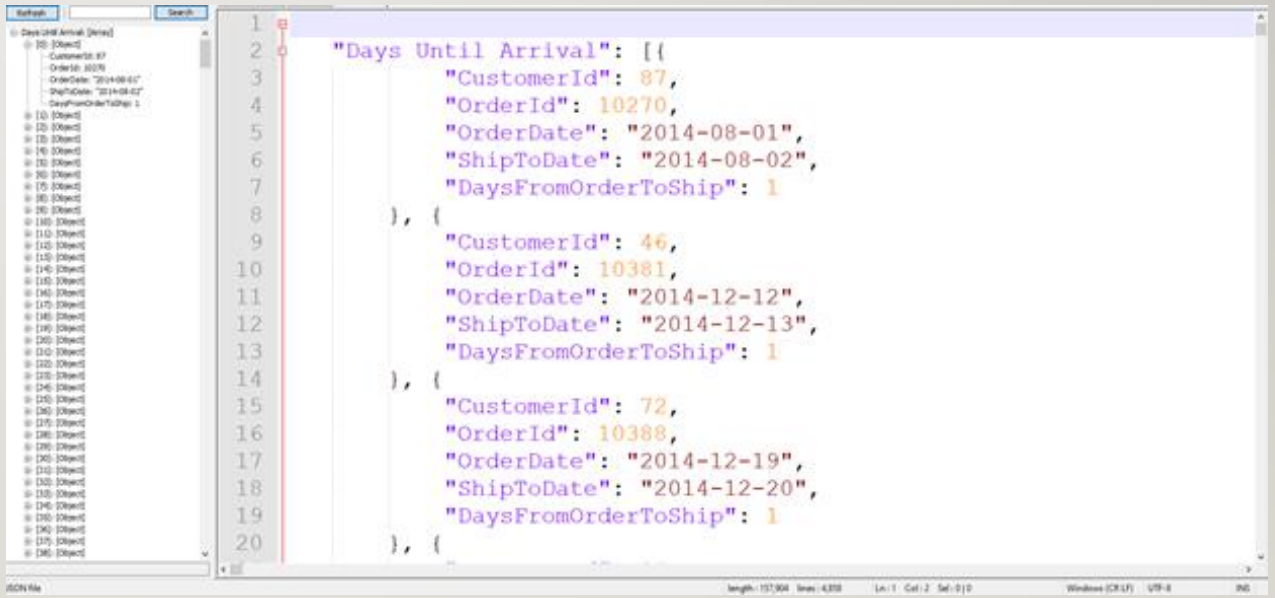
Relational Output:



```
USE Northwinds2020TSQLV6;
SELECT C.CustomerId,
       O.OrderId,
       O.OrderDate,
       O.ShipToDate,
       DATEDIFF(DAY, O.OrderDate, O.ShipToDate) AS DaysFromOrderToShip
FROM Sales.Customer AS C
    INNER JOIN Sales.[Order] AS O
        ON O.CustomerId = C.CustomerId
WHERE ShipToDate IS NOT NULL
GROUP BY DATEDIFF(DAY, O.OrderDate, O.ShipToDate),
         C.CustomerId,
         O.OrderId,
         O.OrderDate,
         O.ShipToDate
ORDER BY DaysFromOrderToShip
```

	CustomerId	OrderId	OrderDate	ShipToDate	DaysFromOrderToShip
1	87	10270	2014-08-01	2014-08-02	1
2	46	10381	2014-12-12	2014-12-13	1
3	72	10388	2014-12-19	2014-12-20	1
4	84	10459	2015-02-27	2015-02-28	1
5	11	10538	2015-05-15	2015-05-16	1
6	3	10573	2015-06-19	2015-06-20	1
7	41	10631	2015-08-14	2015-08-15	1
8	50	10649	2015-08-28	2015-08-29	1
9	90	10673	2015-09-18	2015-09-19	1
10	34	10690	2015-10-02	2015-10-03	1
11	9	10732	2015-11-06	2015-11-07	1
12	24	10774	2015-12-11	2015-12-12	1
13	34	10783	2015-12-18	2015-12-19	1
14	76	10846	2016-01-22	2016-01-23	1
15	45	10884	2016-02-12	2016-02-13	1
16	10	10944	2016-03-12	2016-03-13	1
17	69	11013	2016-04-09	2016-04-10	1
18	86	11046	2016-04-23	2016-04-24	1
19	17	11036	2016-04-20	2016-04-22	2

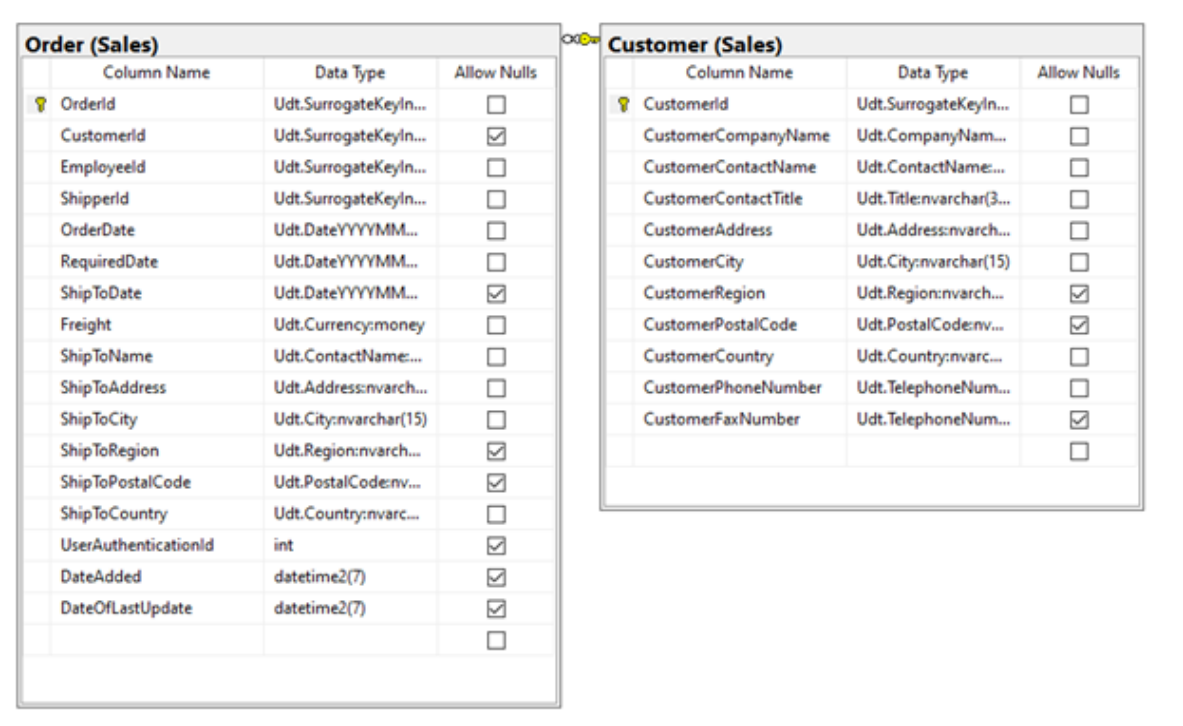
JSON Output:



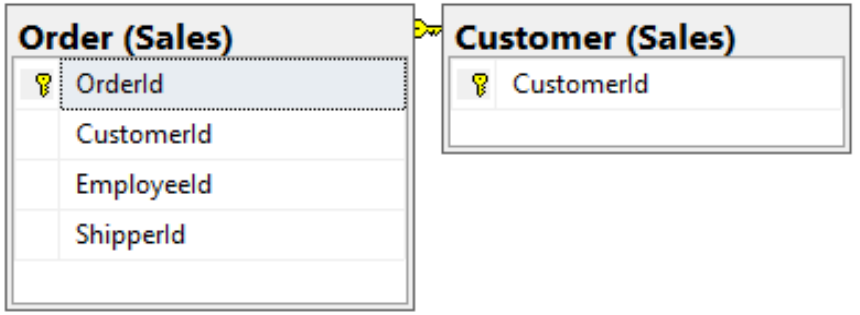

```
{
    "Days Until Arrival": [{
            "CustomerId": 87,
            "OrderId": 10270,
            "OrderDate": "2014-08-01",
            "ShipToDate": "2014-08-02",
            "DaysFromOrderToShip": 1
        }, {
            "CustomerId": 46,
            "OrderId": 10381,
            "OrderDate": "2014-12-12",
            "ShipToDate": "2014-12-13",
            "DaysFromOrderToShip": 1
        }, {
            "CustomerId": 72,
            "OrderId": 10388,
            "OrderDate": "2014-12-19",
            "ShipToDate": "2014-12-20",
            "DaysFromOrderToShip": 1
        }, {
```

JONATHAN'S QUERY (MEDIUM) - BEST #2 VIEWS

Standard View



Key View



JONATHAN'S QUERY (MEDIUM) - BEST #2 TABLES

Table Name	Column Name
Customer	CustomerId
Order	OrderId, EmployeeId
OrderDetail	Quantity

Order By

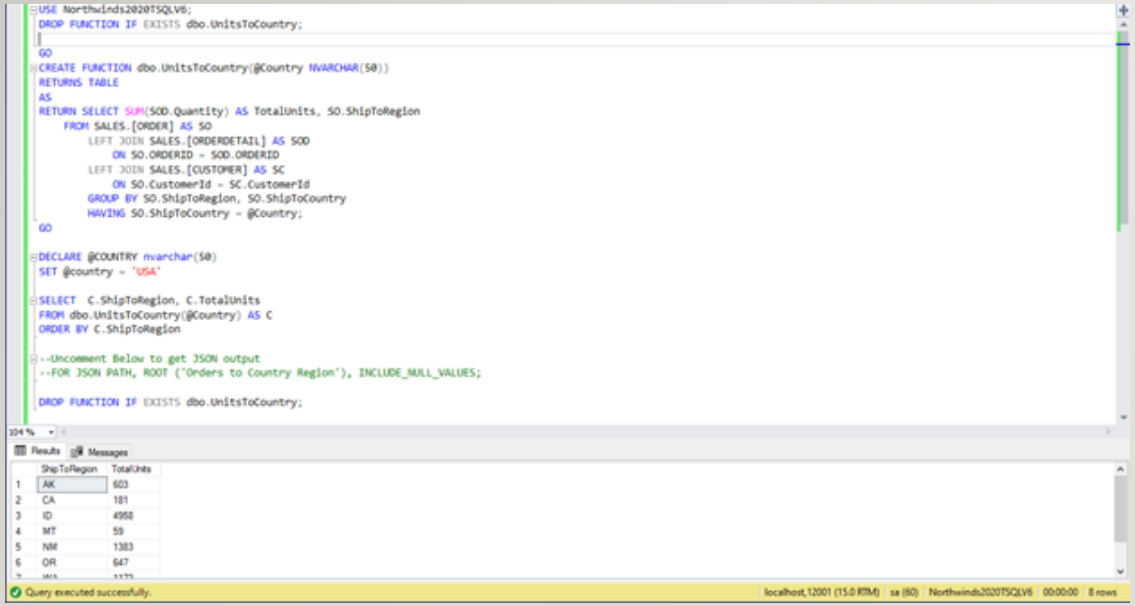
Table Name	Column Name	Sort Order
Order	EmployeeId	ASC

JONATHAN'S QUERY (COMPLEX) - BEST #3 QUERY

```
--Write a function that shows the total quantity going to each region of a specified country

USE Northwinds2020TSQLV6;

DROP FUNCTION IF EXISTS dbo.UnitsToCountry;

 GO

CREATE FUNCTION dbo.UnitsToCountry(@Country NVARCHAR(50))

RETURNS TABLE

AS

RETURN SELECT SUM(SOD.Quantity) AS TotalUnits, SO.ShipToRegion

 FROM SALES.[ORDER] AS SO

      LEFT JOIN SALES.[ORDERDETAIL] AS SOD

           ON SO.ORDERID = SOD.ORDERID

      LEFT JOIN SALES.[CUSTOMER] AS SC

           ON SO.CustomerId = SC.CustomerId

      GROUP BY SO.ShipToRegion, SO.ShipToCountry

      HAVING SO.ShipToCountry = @Country;

GO

DECLARE @COUNTRY nvarchar(50)

SET @country = 'USA'


SELECT  C.ShipToRegion, C.TotalUnits

FROM dbo.UnitsToCountry(@Country) AS C

ORDER BY C.ShipToRegion
```

JONATHAN'S QUERY (COMPLEX) - BEST #3 OUTPUTS

Relational Output:



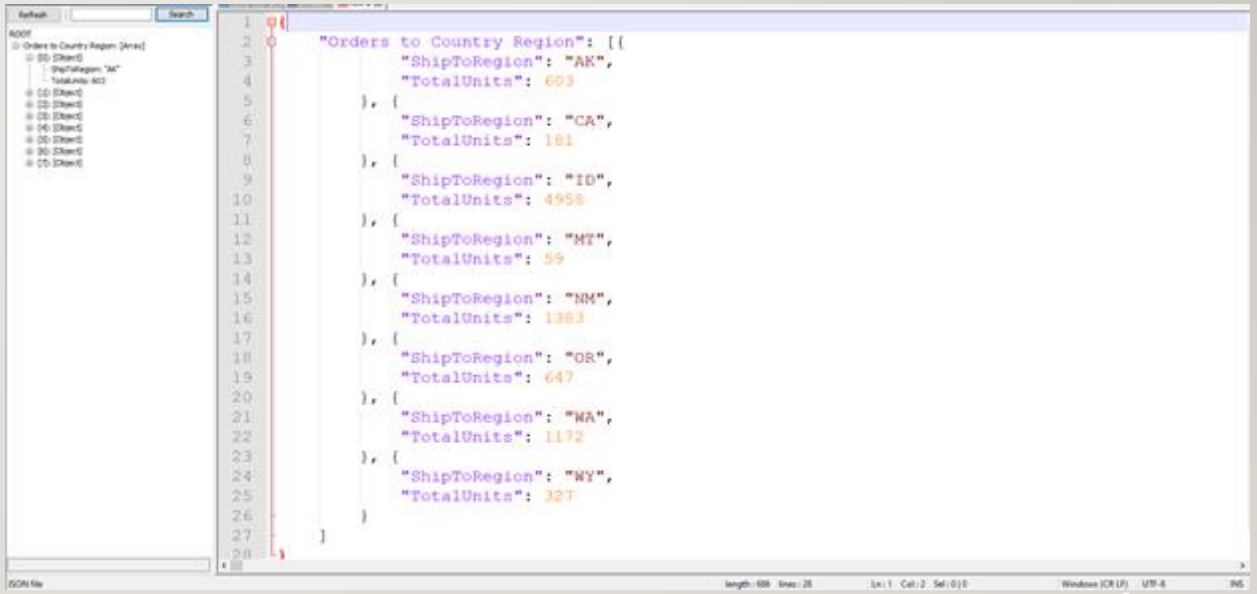
```
USE Northwinds2020TSQLV6;
DROP FUNCTION IF EXISTS dbo.UnitsToCountry;

GO
CREATE FUNCTION dbo.UnitsToCountry(@Country NVARCHAR(50))
RETURNS TABLE
AS
RETURN SELECT SUM(SOD.Quantity) AS TotalUnits, SO.ShipToRegion
    FROM SALES.[ORDER] AS SO
        LEFT JOIN SALES.[ORDERDETAIL] AS SOD
            ON SO.ORDERID = SOD.ORDERID
        LEFT JOIN SALES.[CUSTOMER] AS SC
            ON SO.CustomerId = SC.CustomerId
        GROUP BY SO.ShipToRegion, SO.ShipToCountry
        HAVING SO.ShipToCountry = @Country;
GO

DECLARE @COUNTRY nvarchar(50)
SET @country = 'USA'

SELECT  C.ShipToRegion, C.TotalUnits
FROM dbo.UnitsToCountry(@Country) AS C
ORDER BY C.ShipToRegion

--Uncomment Below to get JSON output
--FOR JSON PATH, ROOT ('Orders to Country Region'), INCLUDE_NULL_VALUES;

DROP FUNCTION IF EXISTS dbo.UnitsToCountry;
```

Results | Messages

	ShipToRegion	TotalUnits
1	AK	603
2	CA	181
3	ID	4958
4	MT	59
5	NM	1383
6	OR	647

Query executed successfully. localhost,12001 (15.0 RTM) | sa (60) | Northwinds2020TSQLV6 | 00:00:00 | 8 rows

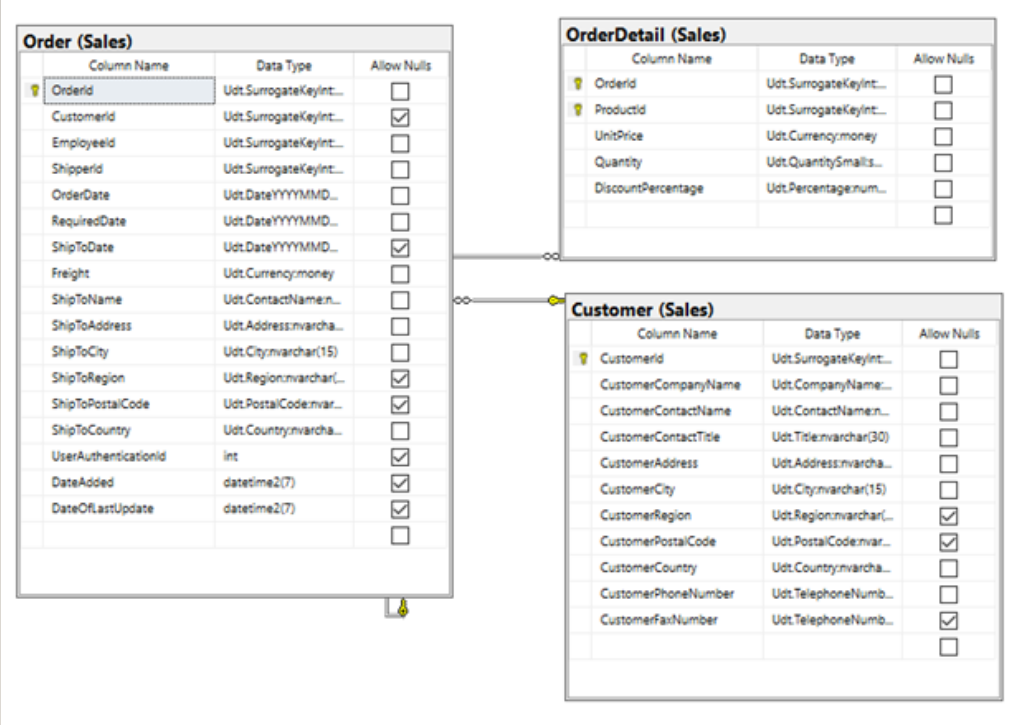
JSON Output:



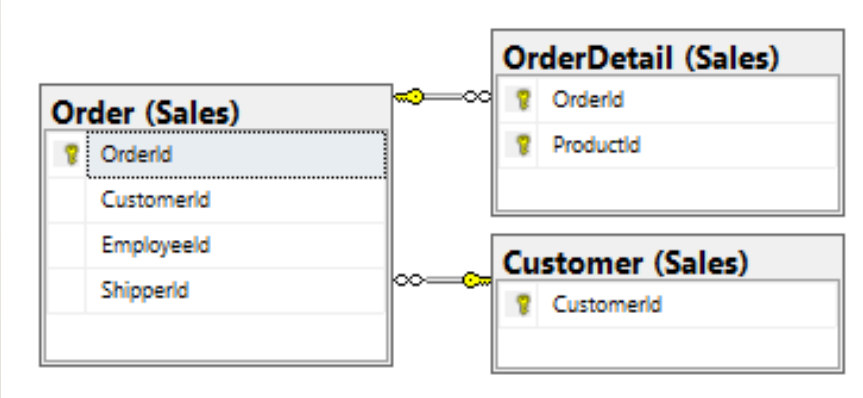
```
{
    "Orders to Country Region": [{
            "ShipToRegion": "AK",
            "TotalUnits": 603
        }, {
            "ShipToRegion": "CA",
            "TotalUnits": 181
        }, {
            "ShipToRegion": "ID",
            "TotalUnits": 4958
        }, {
            "ShipToRegion": "MT",
            "TotalUnits": 59
        }, {
            "ShipToRegion": "NM",
            "TotalUnits": 1383
        }, {
            "ShipToRegion": "OR",
            "TotalUnits": 647
        }, {
            "ShipToRegion": "WA",
            "TotalUnits": 1172
        }, {
            "ShipToRegion": "WY",
            "TotalUnits": 327
        }
    ]
}
```

JONATHAN'S QUERY (COMPLEX) - BEST #3 VIEWS

Standard View



Key View



JONATHAN'S QUERY (MEDIUM) - BEST #3 TABLES

Table Name	Column Name
Order	ShipToRegion
Customer	CustomerId
OrderDetail	Quantity

Order By

Table Name	Column Name	Sort Order
OrderDetail	ShipToRegion	ASC

JONATHAN'S QUERY (SIMPLE) - WORST #1 QUERY

```
-- Show Orders placed in 2016 without Ship Dates
USE Northwinds2020TSQLV6;


SELECT C.CustomerId, O.orderid, O.orderdate, O.ShipToDate
FROM Sales.Customer AS C
	INNER JOIN Sales.[Order] AS O
		ON O.CustomerId = C.CustomerId
WHERE O.ShipToDate is NULL AND YEAR(O.OrderDate) = '2016'
ORDER BY O.OrderDate, O.CustomerId
```

JONATHAN'S QUERY (SIMPLE) - BEST #1 OUTPUTS

Relational Output:

```
USE Northwinds2020TSQLV6;

SELECT C.CustomerId, O.orderid, O.orderdate, O.ShipToDate
FROM Sales.Customer AS C
    INNER JOIN Sales.[Order] AS O
        ON O.CustomerId = C.CustomerId
WHERE O.ShipToDate is NULL AND YEAR(O.OrderDate) = '2016'
ORDER BY O.OrderDate, O.CustomerId

--FOR JSON PATH, ROOT ('Orders From 2016 Without Ship Dates'), INCLUDE_NULL_VALUES;
```

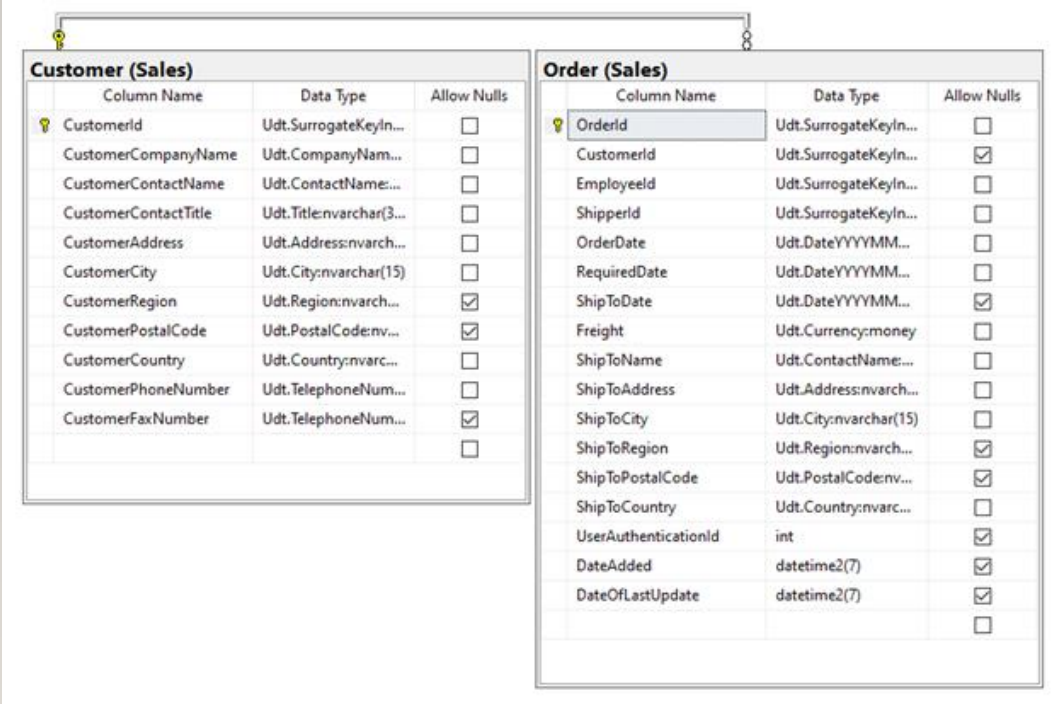
	CustomerId	orderid	orderdate	ShipToDate
1	20	11008	2016-04-08	NULL
2	64	11019	2016-04-13	NULL
3	47	11039	2016-04-21	NULL
4	32	11040	2016-04-22	NULL
5	10	11045	2016-04-23	NULL
6	41	11051	2016-04-27	NULL
7	12	11054	2016-04-28	NULL
8	6	11058	2016-04-29	NULL
9	67	11059	2016-04-29	NULL
10	32	11061	2016-04-30	NULL
11	66	11062	2016-04-30	NULL
12	46	11065	2016-05-01	NULL
13	62	11068	2016-05-04	NULL
14	20	11072	2016-05-05	NULL
15	44	11070	2016-05-05	NULL
16	46	11071	2016-05-05	NULL
17	58	11073	2016-05-05	NULL
18	9	11076	2016-05-06	NULL
19	65	11077	2016-05-06	NULL

JSON Output:

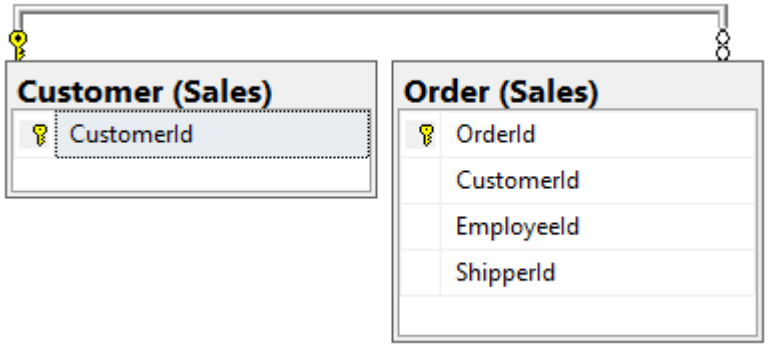

```
{
    "Orders From 2016 Without Ship Dates": [{
            "CustomerId": 20,
            "orderid": 11008,
            "orderdate": "2016-04-08",
            "ShipToDate": null
        }, {
            "CustomerId": 64,
            "orderid": 11019,
            "orderdate": "2016-04-13",
            "ShipToDate": null
        }, {
            "CustomerId": 47,
            "orderid": 11039,
            "orderdate": "2016-04-21",
            "ShipToDate": null
        }, {
            "CustomerId": 32,
            "orderid": 11040,
            "orderdate": "2016-04-22",
            "ShipToDate": null
        }, {
```

JONATHAN'S QUERY (SIMPLE) - BEST #1 VIEWS

Standard View



Key View



JONATHAN'S QUERY (SIMPLE) - BEST #1 TABLES

Table Name	Column Name
Customer	CustomerId
Order	OrderId; Orderdate; ShipToDate

Order By

Table Name	Column Name	Sort Order
Order	OrderDate, CustomerId	ASC, ASC

JONATHAN'S QUERY (MEDIUM) - WORST #2 QUERY

```
-- Show the Average Pay Rate of the different Job Titles
USE AdventureWorks2017;
SELECT DISTINCT
        HE.JobTitle,
       AVG(HP.Rate) AS AverageRate
FROM HumanResources.[Employee] AS HE
    LEFT OUTER JOIN HumanResources.EmployeePayHistory AS HP
         ON HE.BusinessEntityID = HP.BusinessEntityID
GROUP BY HE.JobTitle
ORDER BY AverageRate DESC
```

JONATHAN'S QUERY (MEDIUM) - WORST #2 OUTPUTS

Relational Output:

```
USE AdventureWorks2017;
SELECT DISTINCT
       HE.JobTitle,
       AVG(HP.Rate) AS AverageRate
FROM HumanResources.[Employee] AS HE
    LEFT OUTER JOIN HumanResources.EmployeePayHistory AS HP
        ON HE.BusinessEntityID = HP.BusinessEntityID
GROUP BY HE.JobTitle
ORDER BY AverageRate DESC;

--FOR JSON PATH, ROOT ('Average Salary Per Position'), INCLUDE_NULL_VALUES;
```

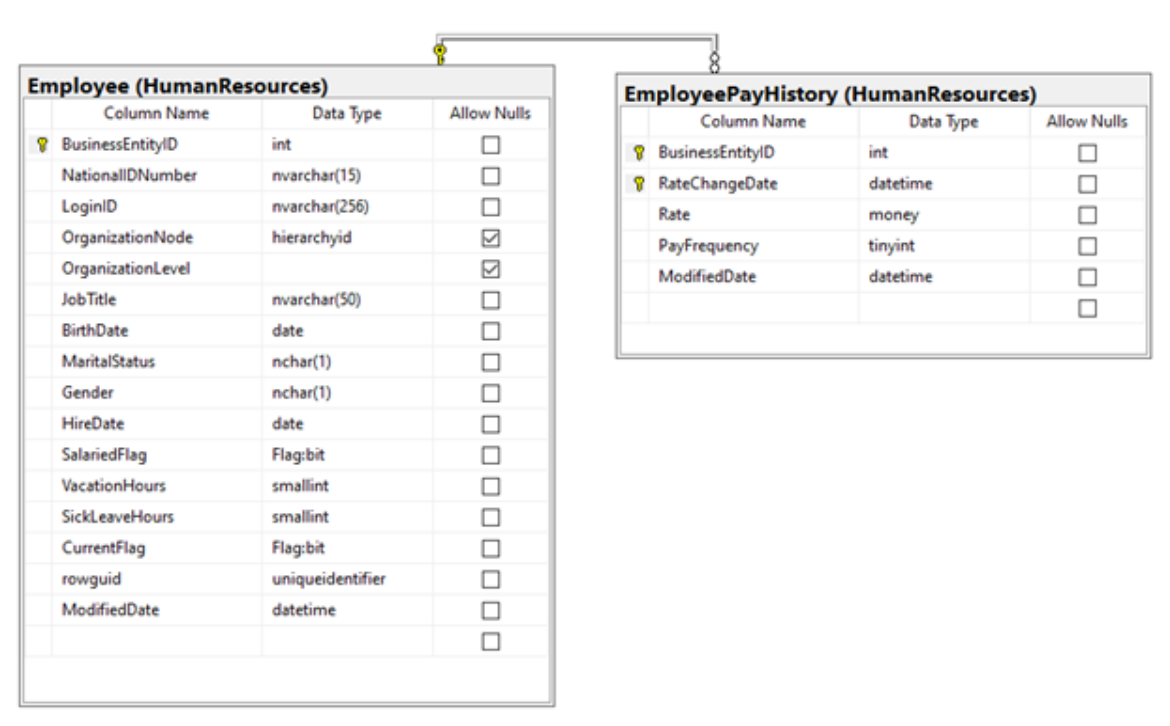
	JobTitle	AverageRate
1	Chief Executive Officer	125.50
2	Vice President of Production	84.1346
3	Vice President of Sales	72.1154
4	Vice President of Engineering	63.4615
5	Information Services Manager	50.4808
6	Chief Financial Officer	49.2379
7	European Sales Manager	48.101
8	North American Sales Manager	48.101
9	Pacific Sales Manager	48.101
10	Research and Development Manager	46.4808
11	Engineering Manager	43.2692
12	Finance Manager	43.2692
13	Research and Development Engineer	40.8654
14	Network Manager	39.6635
15	Database Administrator	38.4615
16	Senior Design Engineer	36.0577
17	Accounts Manager	34.7356
18	Design Engineer	32.6923
19	Network Administrator	32.4519

JSON Output:

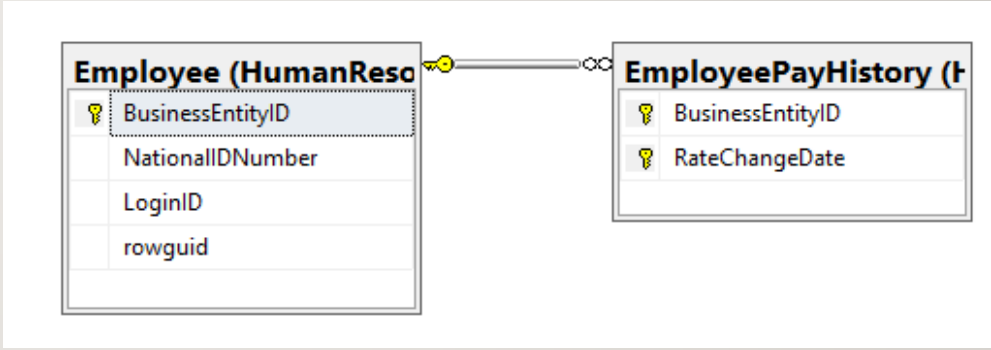
```
{
    "Average Salary Per Position": [{
            "JobTitle": "Chief Executive Officer",
            "AverageRate": 125.5000
        }, {
            "JobTitle": "Vice President of Production",
            "AverageRate": 84.1346
        }, {
            "JobTitle": "Vice President of Sales",
            "AverageRate": 72.1154
        }, {
            "JobTitle": "Vice President of Engineering",
            "AverageRate": 63.4615
        }, {
            "JobTitle": "Information Services Manager",
            "AverageRate": 50.4808
        }, {
            "JobTitle": "Chief Financial Officer",
            "AverageRate": 49.2379
        }, {
```

JONATHAN'S QUERY (MEDIUM) - WORST #2 VIEWS

Standard View



Key View



JONATHAN'S QUERY (MEDIUM) - WORST #2 TABLES

Table Name	Column Name
Employee	JobTitle
EmployeePayHistory	Rate

Order By

Table Name	Column Name	Sort Order
EmployeePayHistory	AverageRate	ASC

JONATHAN'S QUERY (COMPLEX) - WORST #3 QUERY

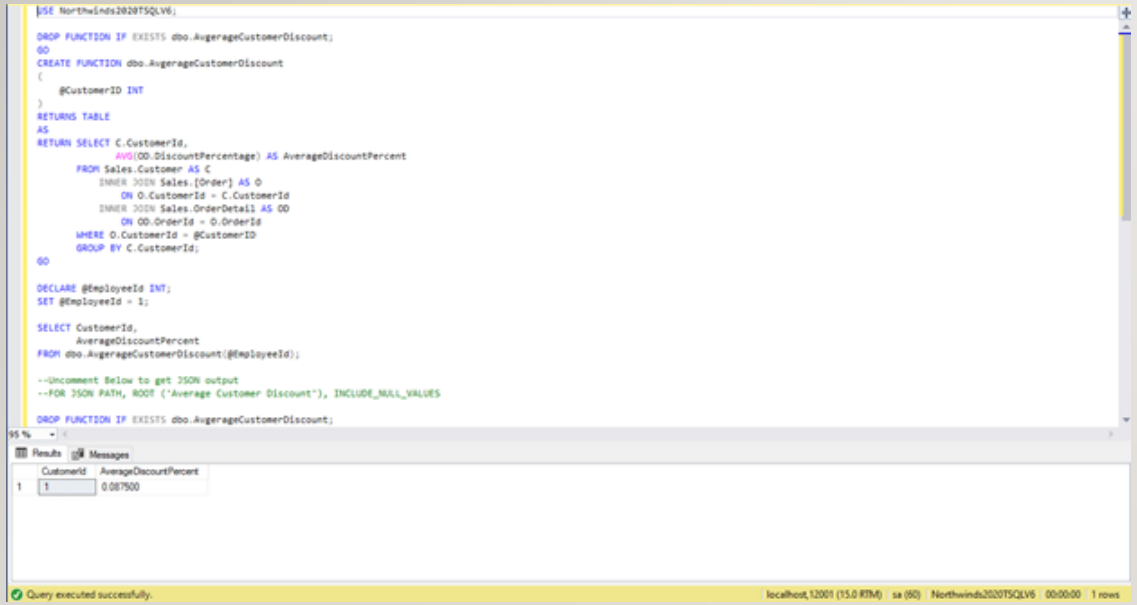
```
-- Show the Average Discount to a
    Specific Customer

USE Northwinds2020TSQLV6;



DROP FUNCTION IF EXISTS
    dbo.AvgerageCustomerDiscount;

GO

CREATE FUNCTION
    dbo.AvgerageCustomerDiscount

(

   @CustomerID INT

)

RETURNS TABLE

AS

RETURN SELECT C.CustomerId,

        AVG(OD.DiscountPercentag
e) AS AverageDiscountPercent

 FROM Sales.Customer AS C

    INNER JOIN Sales.[Order] AS
O

        ON O.CustomerId = C.Cust
omerId

    INNER
JOIN Sales.OrderDetail AS OD

        ON OD.OrderId = O.Order
Id

WHERE O.CustomerId = @Customer
    ID

 GROUP BY C.CustomerId;

GO
```

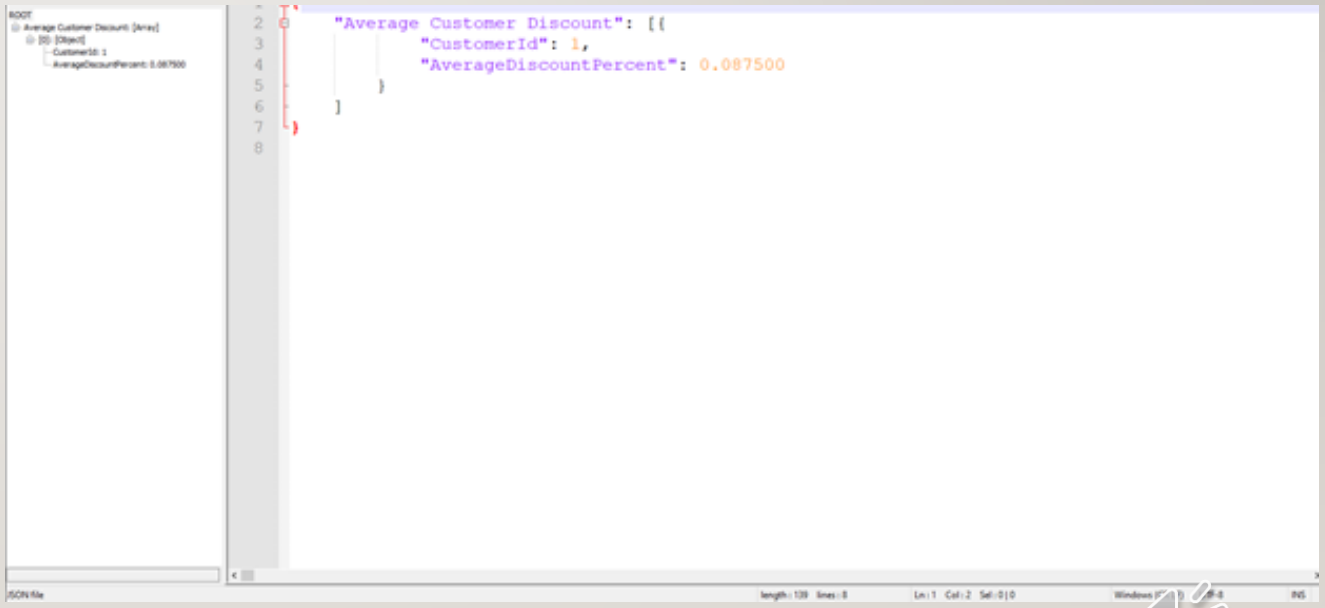
```
DECLARE @ CustomerId INT;

SET @CustomerId = 1;


SELECT CustomerId,

     AverageDiscountPercent

FROM dbo.AvgerageCustomerDiscoun
    t(@CustomerId)

DROP
    FUNCTION IF EXISTS dbo.Avgera
    geCustomerDiscount;
```

JONATHAN'S QUERY (COMPLEX) - WORST #3 OUTPUTS

Relational Output:

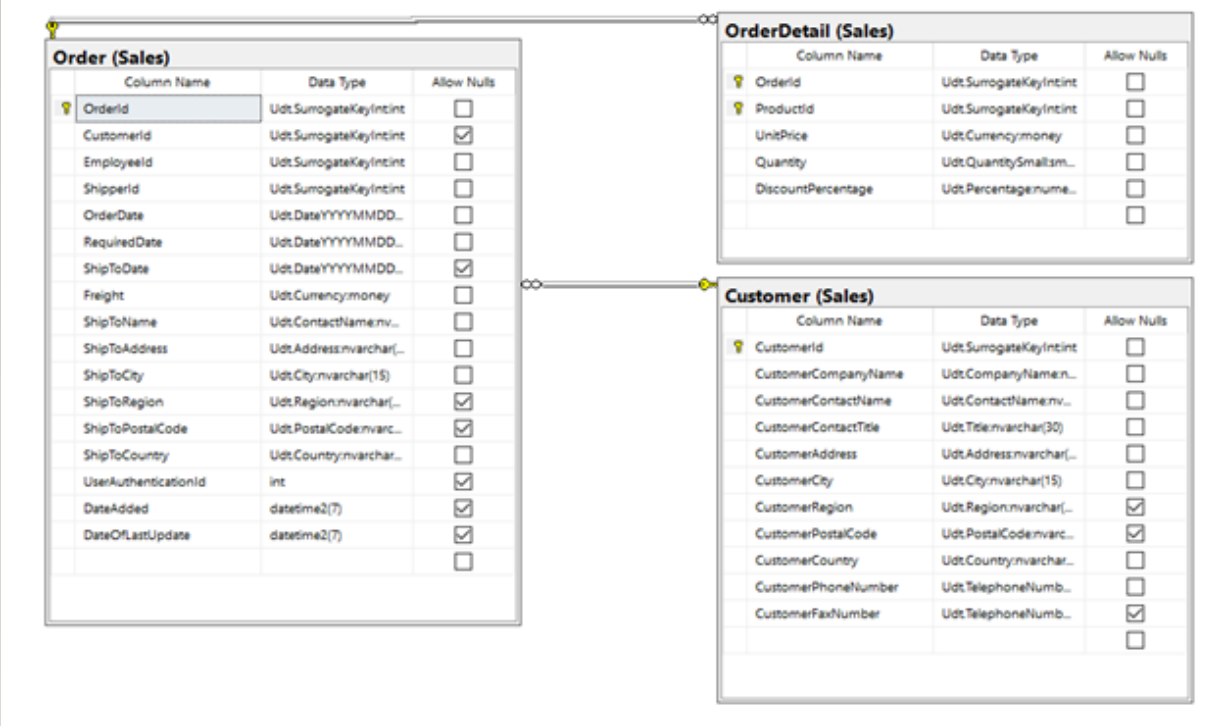
JSON Output:



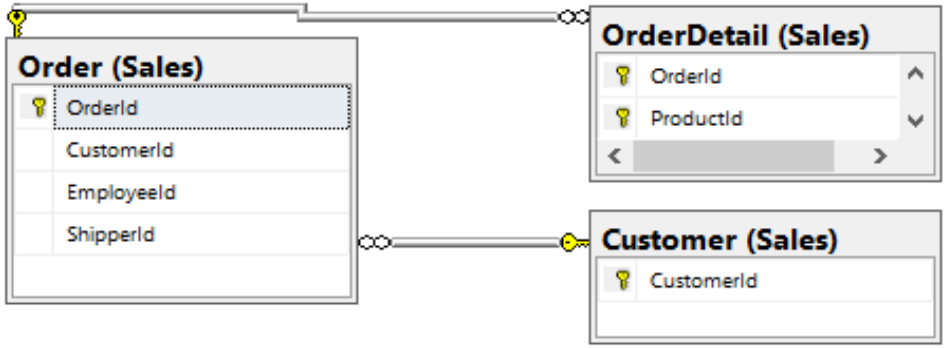


JONATHAN'S QUERY (COMPLEX) - WORST #3 VIEWS

Standard View



Key View



JONATHAN'S QUERY (COMPLEX) - WORST #3 TABLES

Table Name	Column Name
Order	OrderId
Customer	CustomerId
OrderDetail	DiscountPercentage

Order By

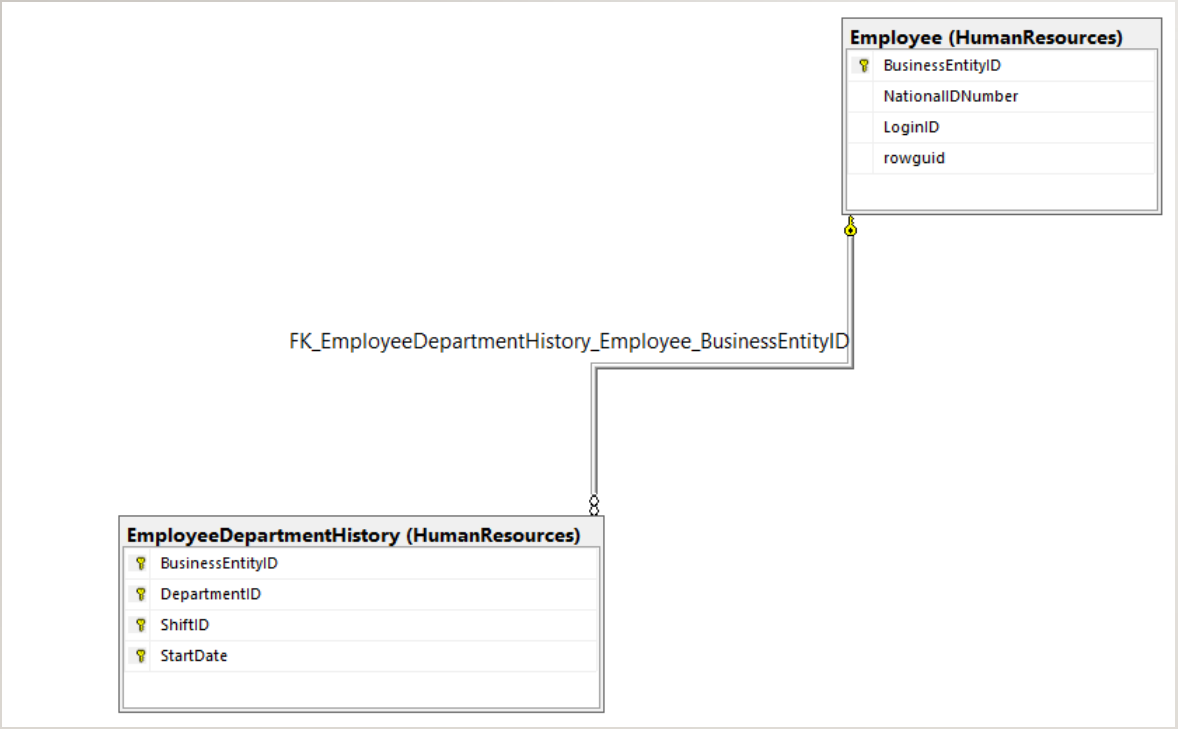
Table Name	Column Name	Sort Order
Customer	CustomerId	ASC

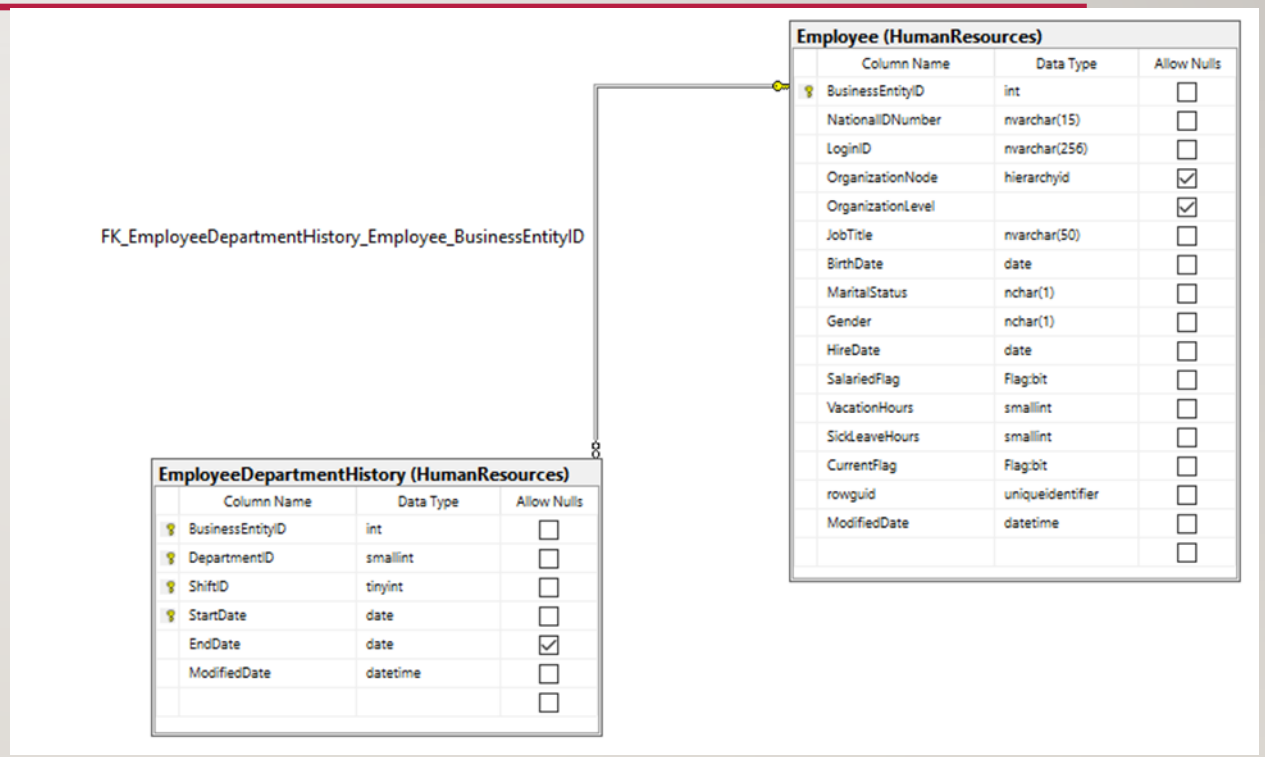
DANNY'S QUERY (SIMPLE) (WORST)

Display for each employee: their business entity ID, the ID of their department, shift ID, and job title

```
USE AdventureWorks2017;
SELECT HREDH.BusinessEntityID,
       HREDH.DepartmentID,
       HREDH.ShiftID,
       HRE.JobTitle
FROM HumanResources.EmployeeDepartmentHistory AS HREDH
INNER JOIN HumanResources.Employee AS HRE
ON HRE.BusinessEntityID = HREDH.BusinessEntityID
ORDER BY HREDH.DepartmentID, HREDH.ShiftID
```

DANNY'S SIMPLE QUERY WORST DIAGRAMS (KEY/STANDARD VIEWS)





DANNY'S SIMPLE QUERY WORST COLUMNS/ORDER BY TABLES

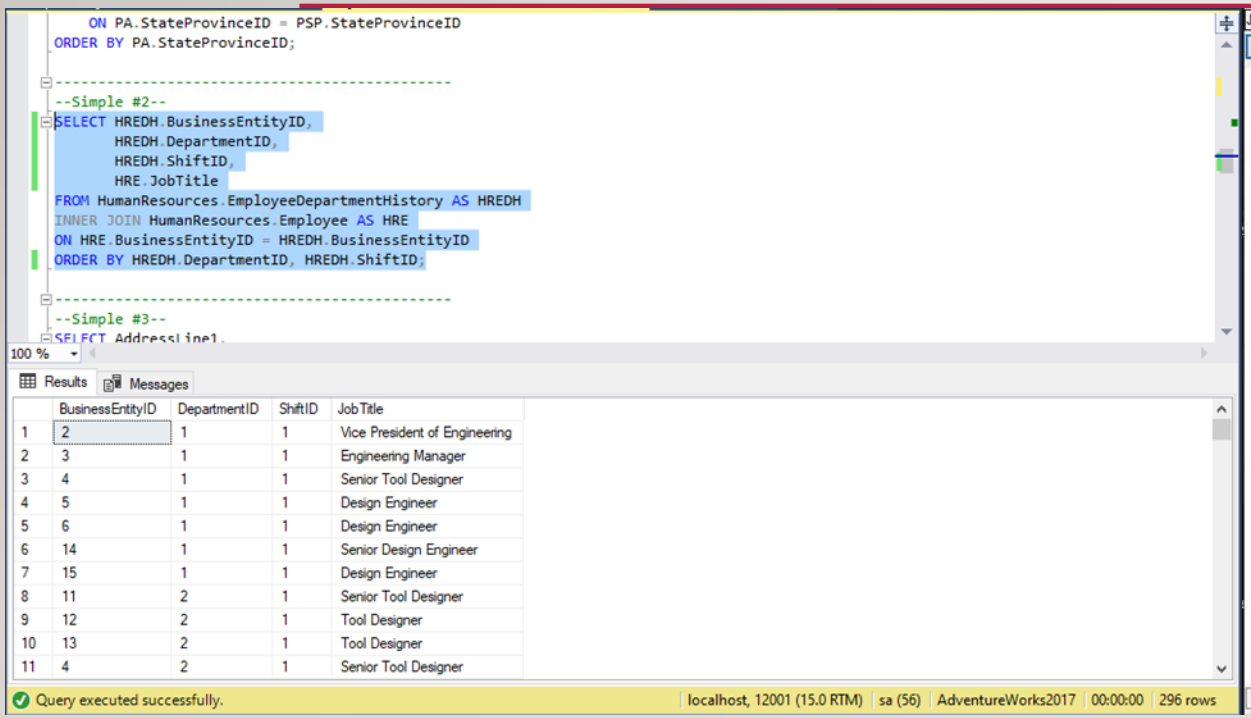
Columns from Tables

Table Name	Column Name
EmployeeDepartmentHistory	BusinessEntityID, DepartmentID, ShiftID
Employee	JobTitle

Order By

Table Name	Column Name	Sort Order
EmployeeDepartmentHistory	DepartmentID, ShiftID	ASC

DANNY'S SIMPLE QUERY WORST (SQL/JSON OUTPUT)



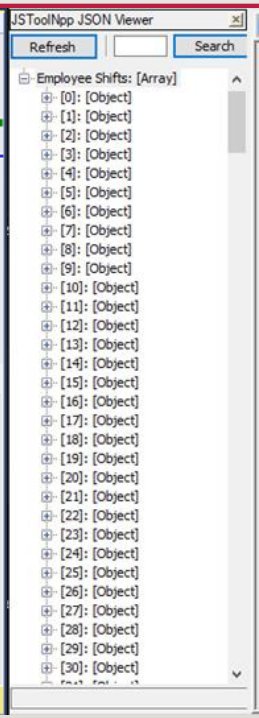
```
    ON PA.StateProvinceID = PSP.StateProvinceID
ORDER BY PA.StateProvinceID;

--------------------------------------------
--Simple #2--
SELECT HREDH.BusinessEntityID,
       HREDH.DepartmentID,
       HREDH.ShiftID,
       HRE.JobTitle
FROM HumanResources.EmployeeDepartmentHistory AS HREDH
INNER JOIN HumanResources.Employee AS HRE
ON HRE.BusinessEntityID = HREDH.BusinessEntityID
ORDER BY HREDH.DepartmentID, HREDH.ShiftID;

--------------------------------------------
--Simple #3--
SELECT AddressLine1,
```

100 %

Results | Messages

	BusinessEntityID	DepartmentID	ShiftID	JobTitle
1	2	1	1	Vice President of Engineering
2	3	1	1	Engineering Manager
3	4	1	1	Senior Tool Designer
4	5	1	1	Design Engineer
5	6	1	1	Design Engineer
6	14	1	1	Senior Design Engineer
7	15	1	1	Design Engineer
8	11	2	1	Senior Tool Designer
9	12	2	1	Tool Designer
10	13	2	1	Tool Designer
11	4	2	1	Senior Tool Designer

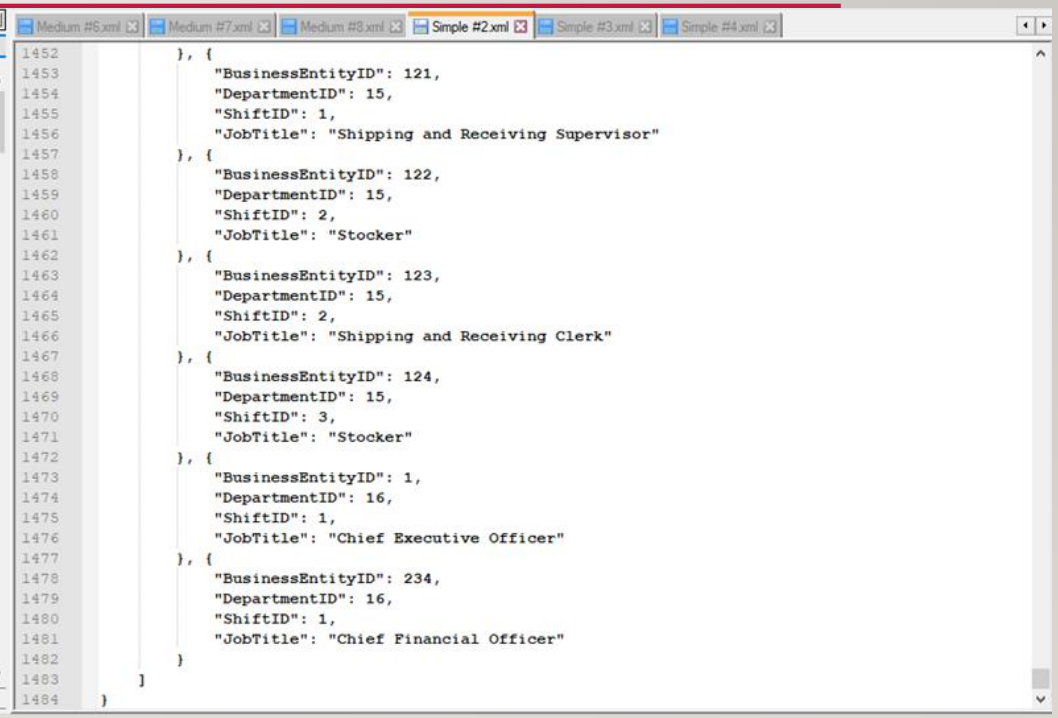
Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (56) | AdventureWorks2017 | 00:00:00 | 296 rows



JSToolNpp JSON Viewer

Refresh | Search

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



Medium #6.xml | Medium #7.xml | Medium #8.xml | Simple #2.xml | Simple #3.xml | Simple #4.xml

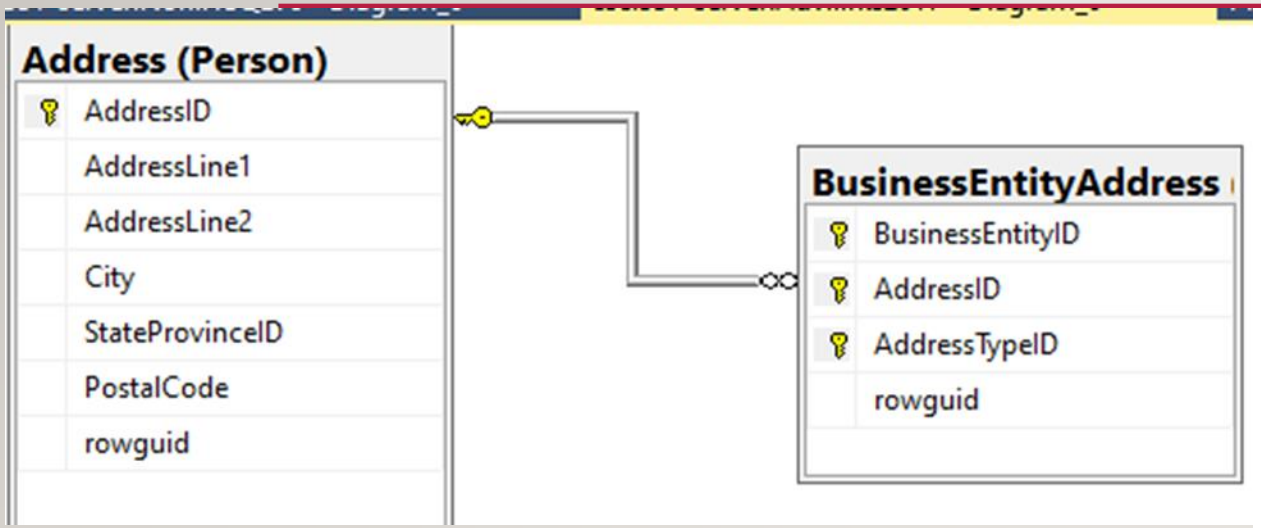
```
        }, {
            "BusinessEntityID": 121,
            "DepartmentID": 15,
            "ShiftID": 1,
            "JobTitle": "Shipping and Receiving Supervisor"
        }, {
            "BusinessEntityID": 122,
            "DepartmentID": 15,
            "ShiftID": 2,
            "JobTitle": "Stocker"
        }, {
            "BusinessEntityID": 123,
            "DepartmentID": 15,
            "ShiftID": 2,
            "JobTitle": "Shipping and Receiving Clerk"
        }, {
            "BusinessEntityID": 124,
            "DepartmentID": 15,
            "ShiftID": 3,
            "JobTitle": "Stocker"
        }, {
            "BusinessEntityID": 1,
            "DepartmentID": 16,
            "ShiftID": 1,
            "JobTitle": "Chief Executive Officer"
        }, {
            "BusinessEntityID": 234,
            "DepartmentID": 16,
            "ShiftID": 1,
            "JobTitle": "Chief Financial Officer"
        }
    ]
}
```

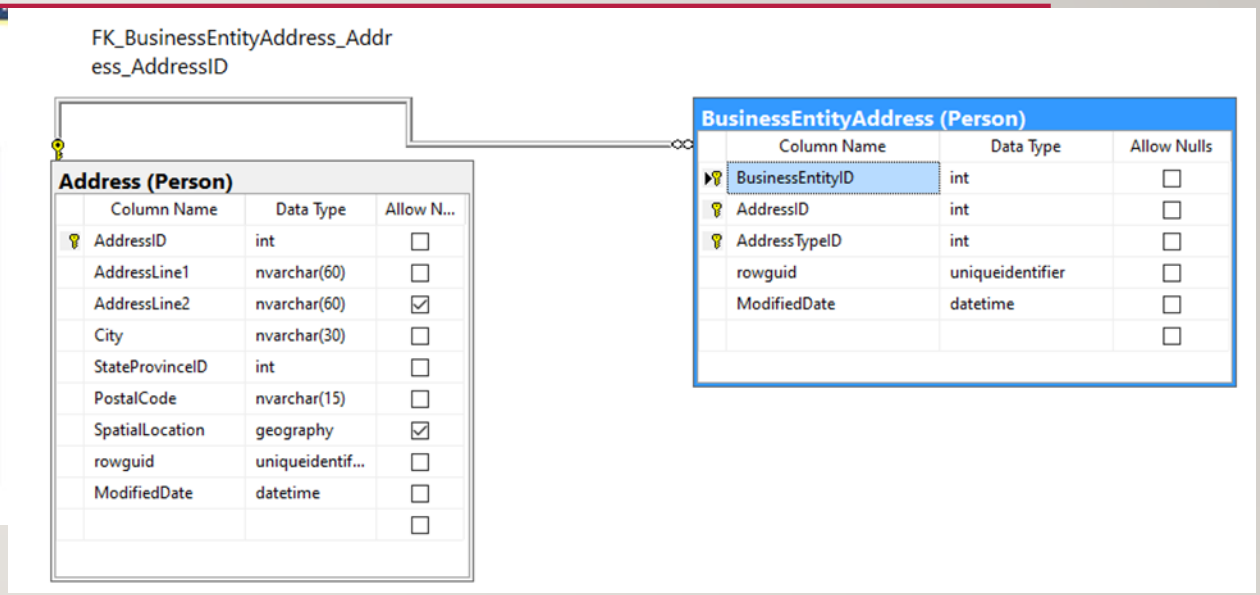
DANNY'S QUERY (SIMPLE) (TOP)

Display for all known addresses that do not use the 2nd address line: their address line 1, postal code, city, and address type ID

```
USE AdventureWorks2017
SELECT PA.AddressLine1,
   PA.PostalCode,
       PA.City,
   PBEA.AddressTypeID
FROM Person.Address AS PA
INNER JOIN Person.BusinessEntityAddress AS PBEA
ON PBEA.AddressID = PA.AddressID
WHERE PA.AddressLine2 IS NULL
ORDER BY PA.PostalCode, PA.City
```

DANNY'S TOP SIMPLE QUERY DIAGRAMS (KEY/STANDARD VIEWS)





DANNY'S TOP SIMPLE QUERY (COLUMNS/ORDER BY TABLES)

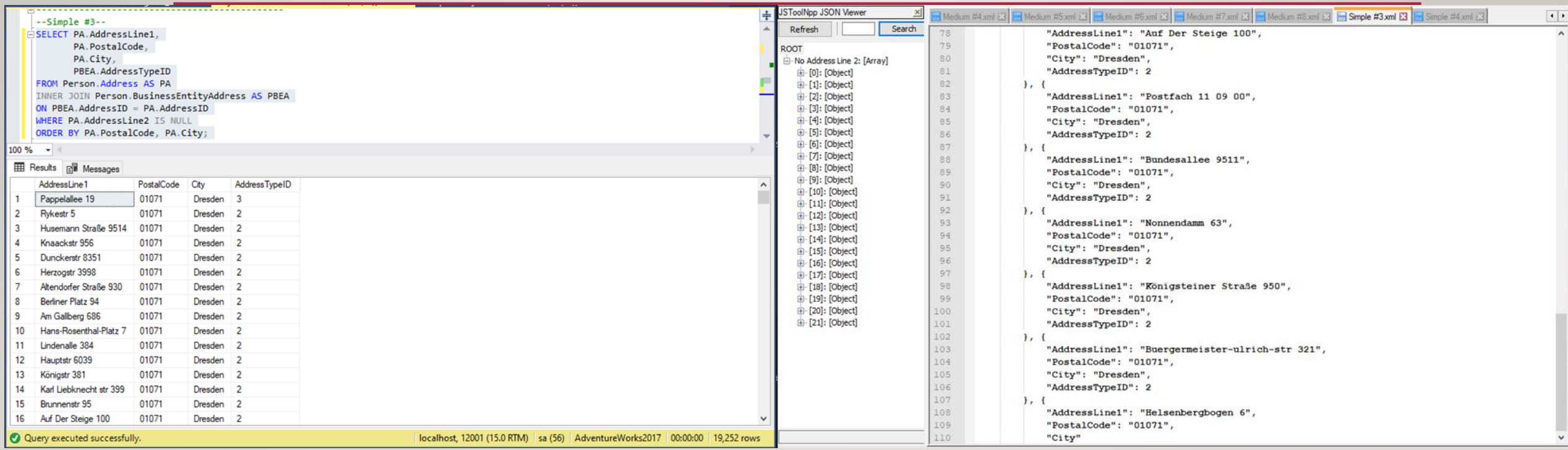
Columns from Tables

Table Name	Column Name
Address	AddressLine1, PostalCode, City
BusinessEntityAddress	AddressTypeID

Order By

Table Name	Column Name	Sort Order
PersonAddress	PostalCode, City	DESC

DANNY'S TOP SIMPLE QUERY (SQL/JSON OUTPUT)



```
--Simple #3--
SELECT PA.AddressLine1,
       PA.PostalCode,
       PA.City,
       PBEA.AddressTypeID
FROM Person.Address AS PA
INNER JOIN Person.BusinessEntityAddress AS PBEA
ON PBEA.AddressID = PA.AddressID
WHERE PA.AddressLine2 IS NULL
ORDER BY PA.PostalCode, PA.City;
```

100 %

Results | Messages

	AddressLine1	PostalCode	City	AddressTypeID
1	Pappelallee 19	01071	Dresden	3
2	Rykestr 5	01071	Dresden	2
3	Husemann Straße 9514	01071	Dresden	2
4	Knaackstr 956	01071	Dresden	2
5	Dunckerstr 8351	01071	Dresden	2
6	Herzogstr 3998	01071	Dresden	2
7	Altendorfer Straße 930	01071	Dresden	2
8	Berliner Platz 94	01071	Dresden	2
9	Am Gallberg 686	01071	Dresden	2
10	Hans-Rosenthal-Platz 7	01071	Dresden	2
11	Lindenalle 384	01071	Dresden	2
12	Hauptstr 6039	01071	Dresden	2
13	Königstr 381	01071	Dresden	2
14	Karl Liebknecht str 399	01071	Dresden	2
15	Brunnenstr 95	01071	Dresden	2
16	Auf Der Steige 100	01071	Dresden	2

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (56) | AdventureWorks2017 | 00:00:00 | 19,252 rows

JSToolNpp JSON Viewer

Refresh | Search

ROOT

- No Address Line 2: [Array]
  - [0]: [Object]
  - [1]: [Object]
  - [2]: [Object]
  - [3]: [Object]
  - [4]: [Object]
  - [5]: [Object]
  - [6]: [Object]
  - [7]: [Object]
  - [8]: [Object]
  - [9]: [Object]
  - [10]: [Object]
  - [11]: [Object]
  - [12]: [Object]
  - [13]: [Object]
  - [14]: [Object]
  - [15]: [Object]
  - [16]: [Object]
  - [17]: [Object]
  - [18]: [Object]
  - [19]: [Object]
  - [20]: [Object]
  - [21]: [Object]

Medium #4.xml | Medium #5.xml | Medium #6.xml | Medium #7.xml | Medium #8.xml | Simple #3.xml | Simple #4.xml

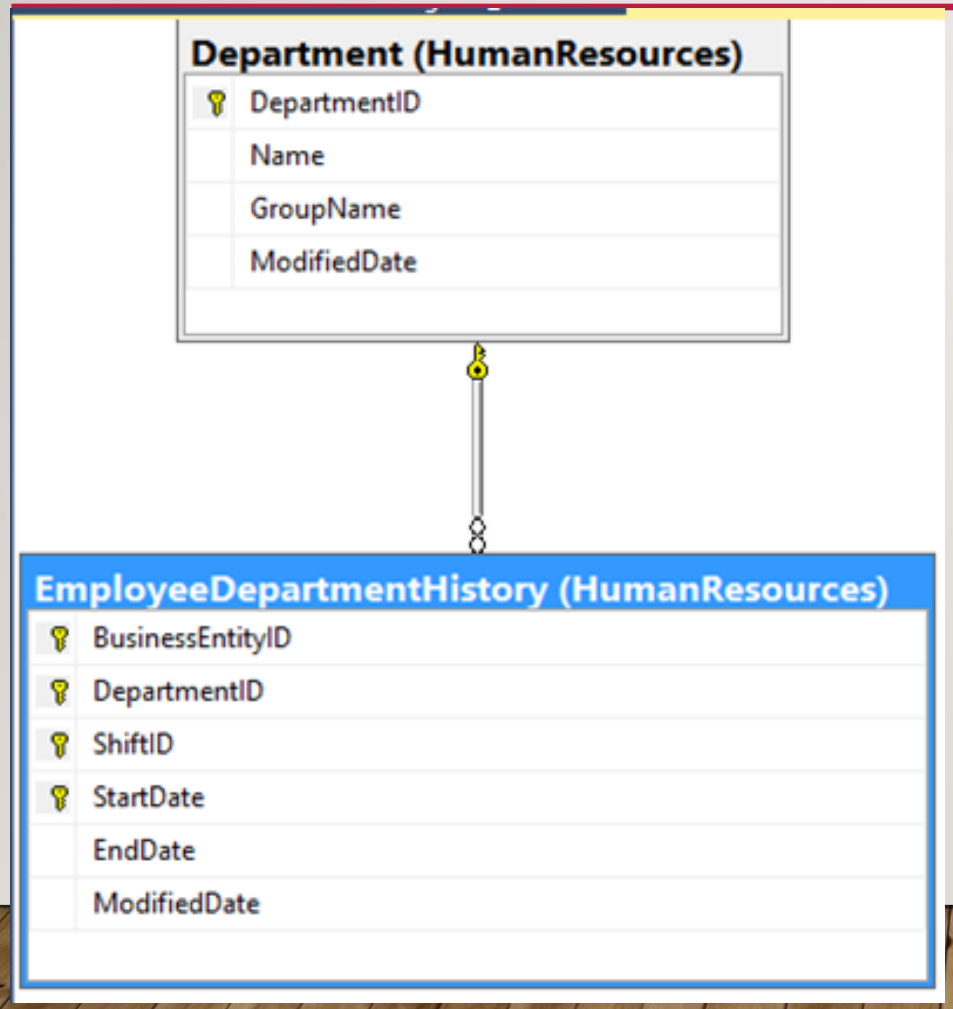

```
            "AddressLine1": "Auf Der Steige 100",
            "PostalCode": "01071",
            "City": "Dresden",
            "AddressTypeID": 2
        }, {
            "AddressLine1": "Postfach 11 09 00",
            "PostalCode": "01071",
            "City": "Dresden",
            "AddressTypeID": 2
        }, {
            "AddressLine1": "Bundesallee 9511",
            "PostalCode": "01071",
            "City": "Dresden",
            "AddressTypeID": 2
        }, {
            "AddressLine1": "Nonnendamm 63",
            "PostalCode": "01071",
            "City": "Dresden",
            "AddressTypeID": 2
        }, {
            "AddressLine1": "Königsteiner Straße 950",
            "PostalCode": "01071",
            "City": "Dresden",
            "AddressTypeID": 2
        }, {
            "AddressLine1": "Buergermeister-ulrich-str 321",
            "PostalCode": "01071",
            "City": "Dresden",
            "AddressTypeID": 2
        }, {
            "AddressLine1": "Helsenbergbogen 6",
            "PostalCode": "01071",
            "City"
```

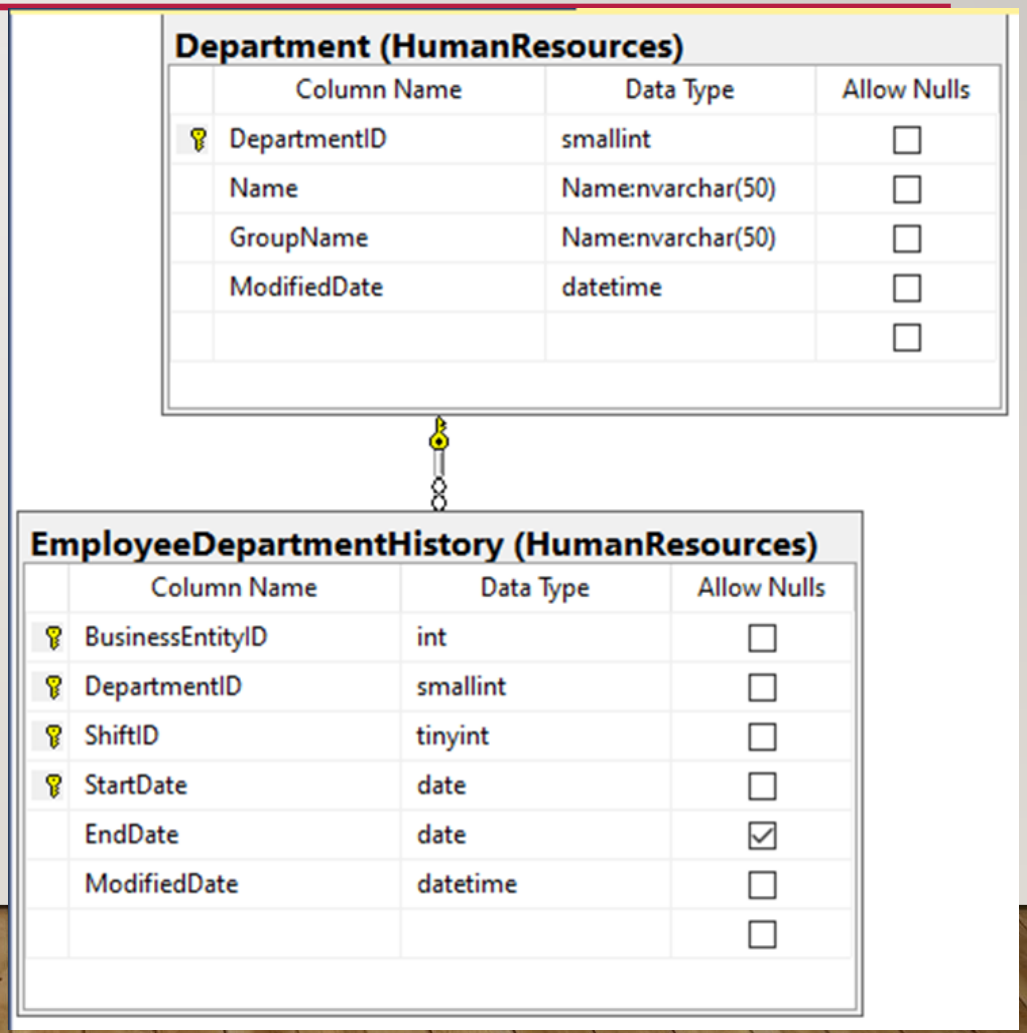
DANNY'S QUERY (MEDIUM) (TOP)

Display for each department: the department ID, the name of the department, the shift ID, and how many employees worked in that department during that shift

```
USE AdventureWorks2017

SELECT HREDH.DepartmentID,
       HRD.Name,
       COUNT(HREDH.DepartmentID) AS 'Employee Number',
       HREDH.ShiftID
FROM HumanResources.Department AS HRD
    INNER JOIN HumanResources.EmployeeDepartmentHistory AS HREDH
        ON HRD.DepartmentID = HREDH.DepartmentID
GROUP BY HREDH.DepartmentID,
         HREDH.ShiftID,
         HRD.Name
ORDER BY HREDH.DepartmentID
```

DANNY'S TOP MEDIUM QUERY DIAGRAMS (KEY/STANDARD VIEWS)





DANNY'S TOP MEDIUM QUERY (COLUMNS/ORDER BY TABLES)

Columns from Tables

Table Name	Column Name
Department	DepartmentName
EmployeeDepartmentHistory	DepartmentID, ShiftID

Order By

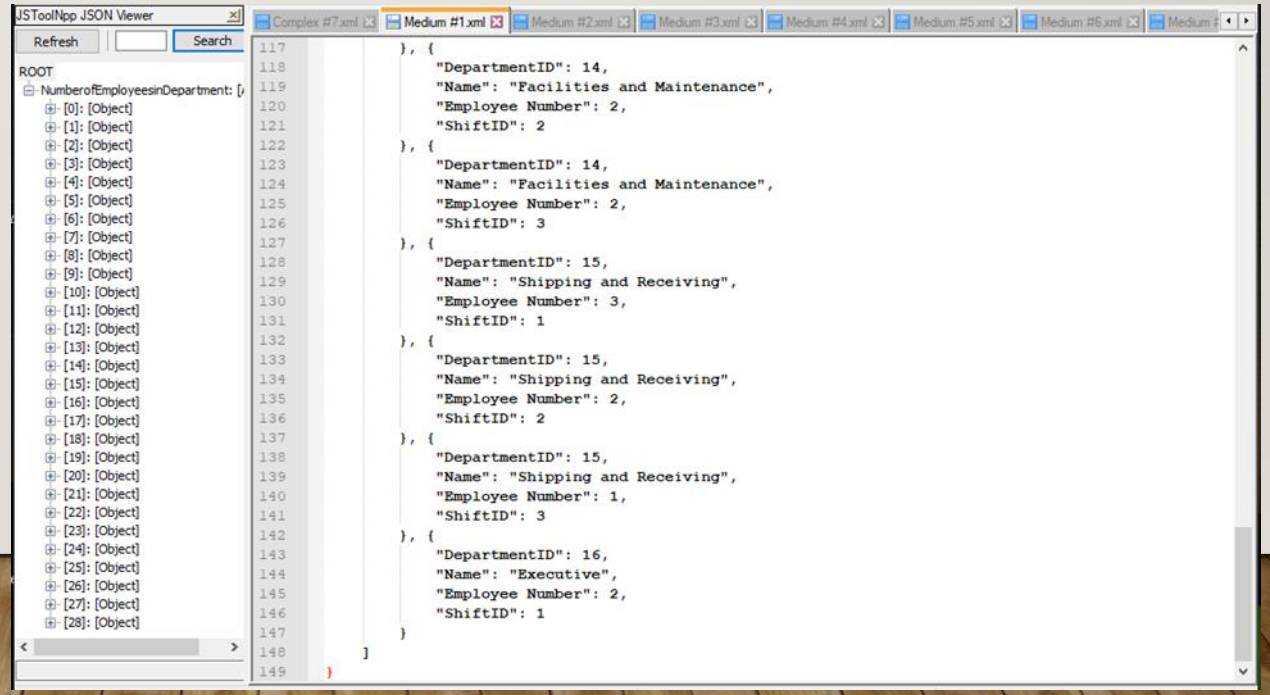
Table Name	Column Name	Sort Order
EmployeeDepartmentHistory	DepartmentID	ASC

DANNY'S TOP MEDIUM QUERY (SQL/JSON OUTPUT)

Results | Messages

	DepartmentID	Name	Employee Number	ShiftID
1	1	Engineering	7	1
2	2	Tool Design	4	1
3	3	Sales	18	1
4	4	Marketing	10	1
5	5	Purchasing	13	1
6	6	Research and Development	4	1
7	7	Production	80	1
8	7	Production	54	2
9	7	Production	46	3
10	8	Production Control	4	1
11	8	Production Control	1	2
12	8	Production Control	1	3
13	9	Human Resources	6	1
14	10	Finance	11	1
15	11	Information Services	9	1

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (56) | AdventureWorks2017 | 00:00:00 | 29 rows

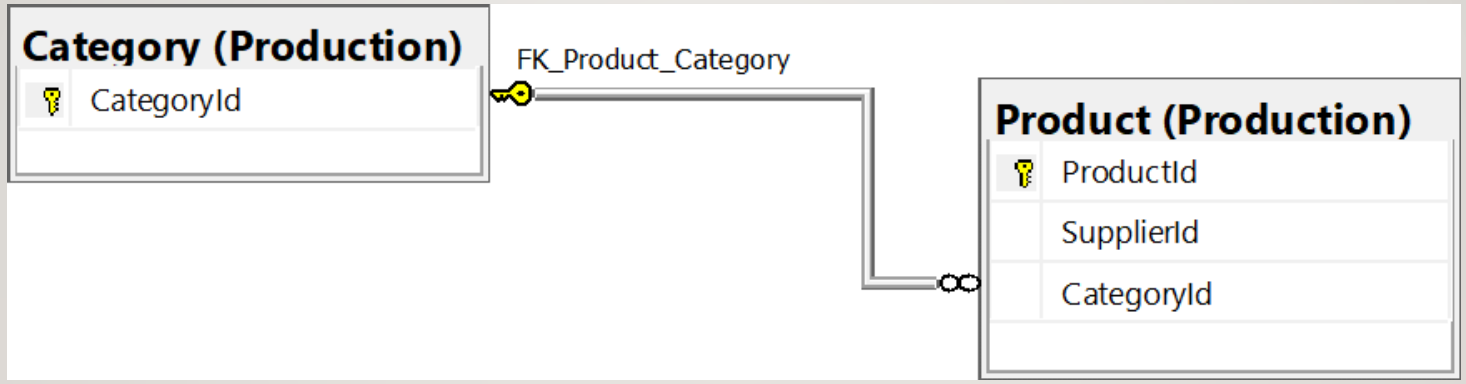


DANNY'S QUERY (MEDIUM) (WORST)

Display for each category: the number of products it contains, the category name, and the description of the category

```
USE Northwinds2020TSQLV6
SELECT COUNT(PP.CategoryId) AS 'Number of Products in this Category',
       PC.CategoryName,
       PC.Description
FROM Production.Product AS PP
    INNER JOIN Production.Category AS PC
        ON PP.CategoryId = PC.CategoryId
GROUP BY PC.CategoryName,
         PC.Description
ORDER BY PC.CategoryName
```

DANNY'S WORST MEDIUM QUERY DIAGRAMS (KEY/STANDARD VIEWS)



Category (Production)

	Column Name	Data Type	Allow Nulls
🔑	CategoryId	Udt.SurrogateKeyIn...	☐
	CategoryName	Udt.CategoryName...	☐
	Description	Udt.Description:nv...	☐
			☐

FK_Product_Category

Product (Production)

	Column Name	Data Type	Allow Nulls
🔑	ProductId	Udt.SurrogateKeyIn...	☐
	ProductName	Udt.ProductName:...	☐
	SupplierId	Udt.SurrogateKeyIn...	☐
	CategoryId	Udt.SurrogateKeyIn...	☐
	UnitPrice	Udt.Currency:money	☐
	Discontinued	Udt.FlagBit:bit	☐
			☐

DANNY'S WORST MEDIUM QUERY (COLUMNS/ORDER BY TABLES)

Columns from Tables

Table Name	Column Name
Category	CategoryName, Description
Product	COUNT(ProductId) AS Number of Products in this category

Order By

Table Name	Column Name	Sort Order
Category	CategoryName	ASC

DANNY'S WORST MEDIUM QUERY (SQL/JSON OUTPUT)

	Number of Products in this Category	CategoryName	Description
1	12	Beverages	Soft drinks, coffees, teas, beers, and ales
2	12	Condiments	Sweet and savory sauces, relishes, spreads, and ...
3	13	Confections	Desserts, candies, and sweet breads
4	10	Dairy Products	Cheeses
5	7	Grains/Cereals	Breads, crackers, pasta, and cereal
6	6	Meat/Poultry	Prepared meats
7	5	Produce	Dried fruit and bean curd
8	12	Seafood	Seaweed and fish

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (74) | Northwinds2020TSQLV6 | 00:00:00 | 8 rows

Refresh | Search

ROOT
- # of Products: [Array]
  - [0]: [Object]
  - [1]: [Object]
  - [2]: [Object]
  - [3]: [Object]
  - [4]: [Object]
  - [5]: [Object]
  - [6]: [Object]
  - [7]: [Object]


```
            "CategoryName": "Beverages",
            "Description": "Soft drinks, coffees, teas, beers, and ales"
        }, {
            "Number of Products in this Category": 12,
            "CategoryName": "Condiments",
            "Description": "Sweet and savory sauces, relishes, spreads, and seasonings"
        }, {
            "Number of Products in this Category": 13,
            "CategoryName": "Confections",
            "Description": "Desserts, candies, and sweet breads"
        }, {
            "Number of Products in this Category": 10,
            "CategoryName": "Dairy Products",
            "Description": "Cheeses"
        }, {
            "Number of Products in this Category": 7,
            "CategoryName": "Grains\/Cereals",
            "Description": "Breads, crackers, pasta, and cereal"
        }, {
            "Number of Products in this Category": 6,
            "CategoryName": "Meat\/Poultry",
            "Description": "Prepared meats"
        }, {
            "Number of Products in this Category": 5,
            "CategoryName": "Produce",
            "Description": "Dried fruit and bean curd"
        }, {
            "Number of Products in this Category": 12,
            "CategoryName": "Seafood",
            "Description": "Seaweed and fish"
        }
    ]
)
```

THIS FUNCTION WILL BE USED FOR THE FOLLOWING QUERY

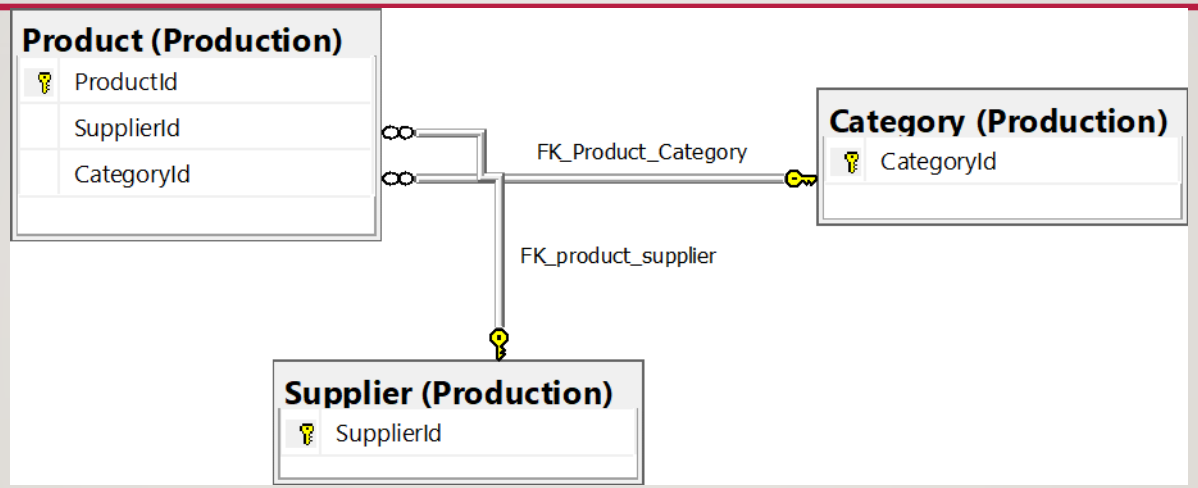
```
CREATE FUNCTION fnd_suppliernm
(@SupplierID int)
RETURNS varchar(15)
BEGIN
RETURN
(
SELECT SupplierCompanyName
FROM Production.Supplier
WHERE SupplierId = @SupplierID
);
END;
```

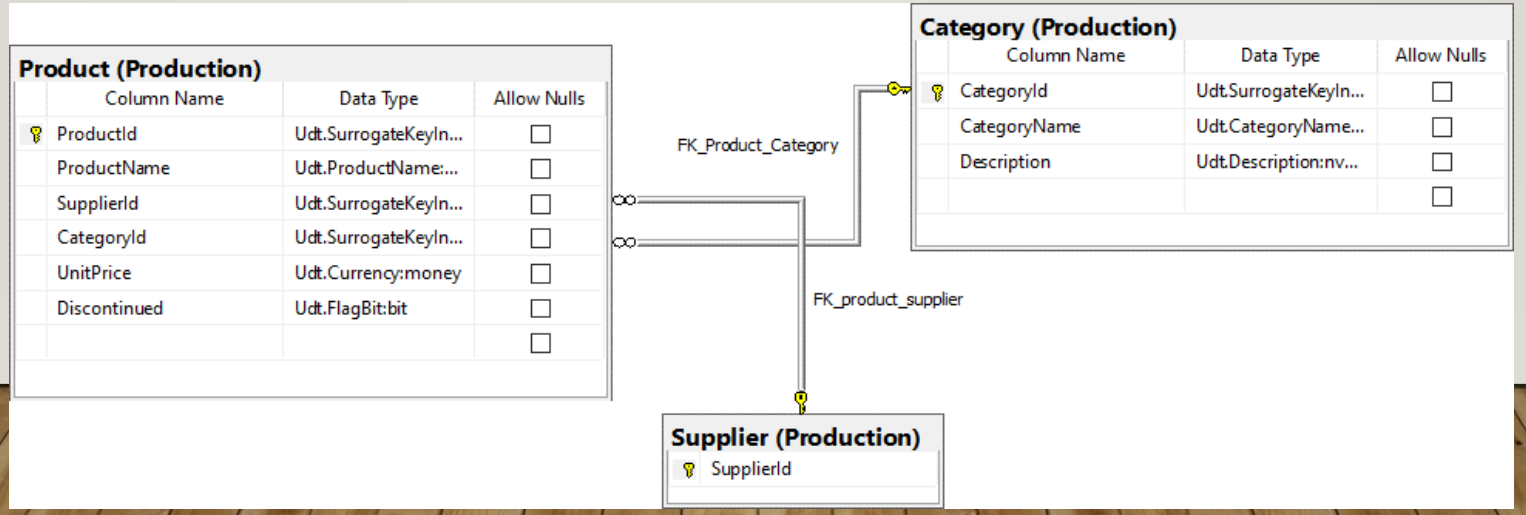
DANNY'S QUERY (COMPLEX) (TOP)

Display for all products that share the same supplier as the inputted ID: the product ID, name, and the ID of the category said product belongs to.

```
USE Northwinds2020TSQLV6

SELECT PrPr.ProductId, PrPr.ProductName, PrC.CategoryId

FROM Production.Product AS PrPr

    INNER JOIN Production.Category AS PrC

        ON PrPr.CategoryId = PrC.CategoryId

    INNER JOIN Production.Supplier AS PrS

        ON PrS.SupplierId = PrPr.SupplierId

WHERE PrS.SupplierCompanyName = dbo.fnd_suppliernm(11)

GROUP BY PrC.CategoryId, PrPr.ProductName, PrPr.ProductId

ORDER BY PrC.CategoryId
```

DANNY'S TOP COMPLEX QUERY DIAGRAMS (KEY/STANDARD VIEWS)





DANNY'S TOP COMPLEX QUERY (COLUMNS/ORDER BY TABLES)

Columns from Tables

Table Name	Column Name
Product	ProductId, ProductName
Category	CategoryId

Order By

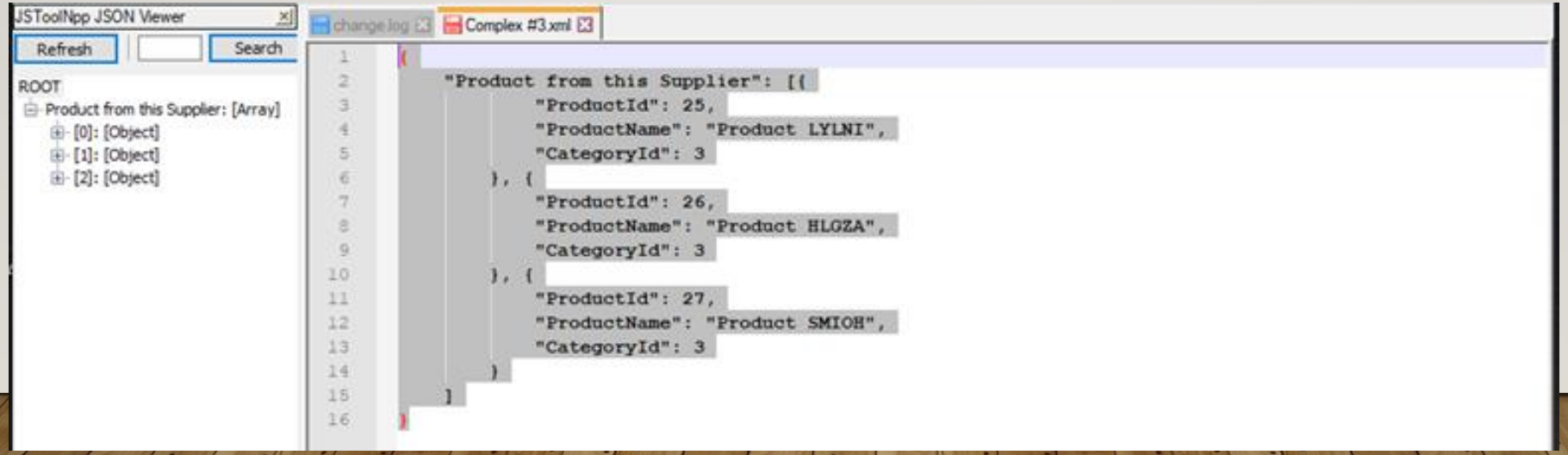
Table Name	Column Name	Sort Order
Category	CategoryID	ASC

DANNY'S TOP COMPLEX QUERY (SQL/JSON OUTPUT)

Results | Messages

	ProductId	ProductName	CategoryId
1	25	Product LYLNI	3
2	26	Product HLGZA	3
3	27	Product SMIOH	3

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (52) | Northwinds2020TSQLV6 | 00:00:00 | 3 rows



THIS FUNCTION WILL BE USED FOR THE FOLLOWING QUERY

```
CREATE FUNCTION fnd_city
(
    @GeoKey int
)
RETURNS varchar(15)
BEGIN
    RETURN
    (
        SELECT SalesTerritoryKey
        FROM dbo.DimGeography
        WHERE GeographyKey = @GeoKey
    );
END;
```

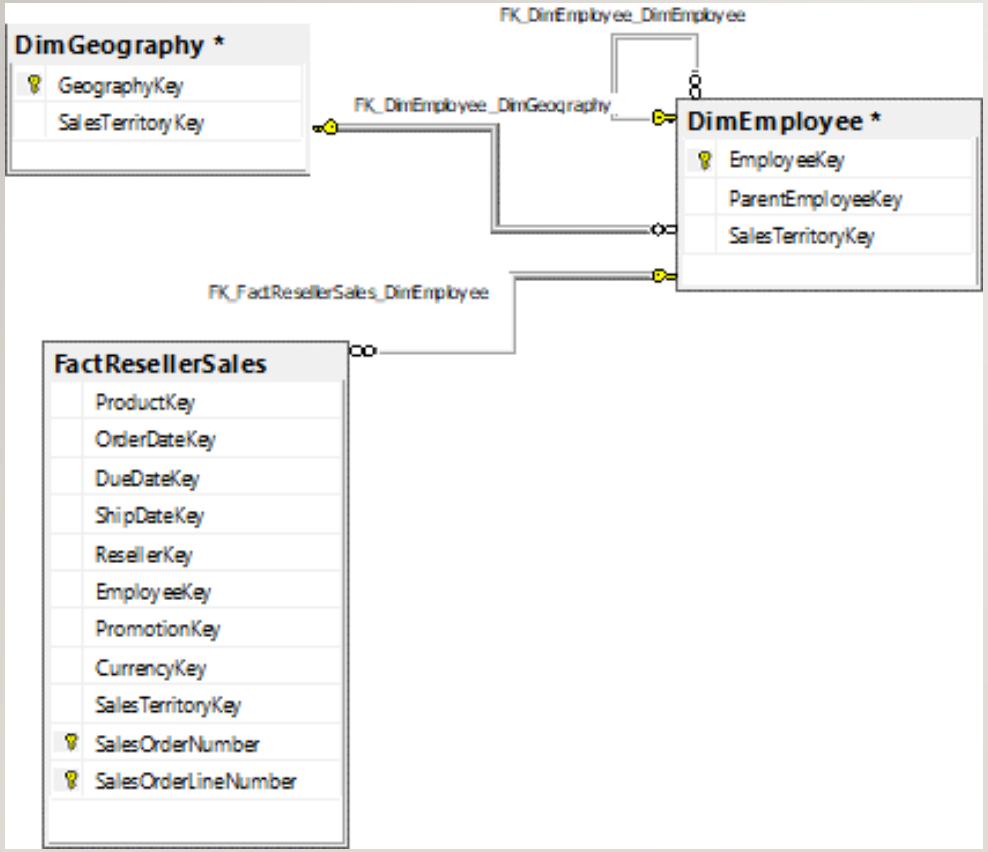
DANNY'S QUERY (COMPLEX) (WORST)

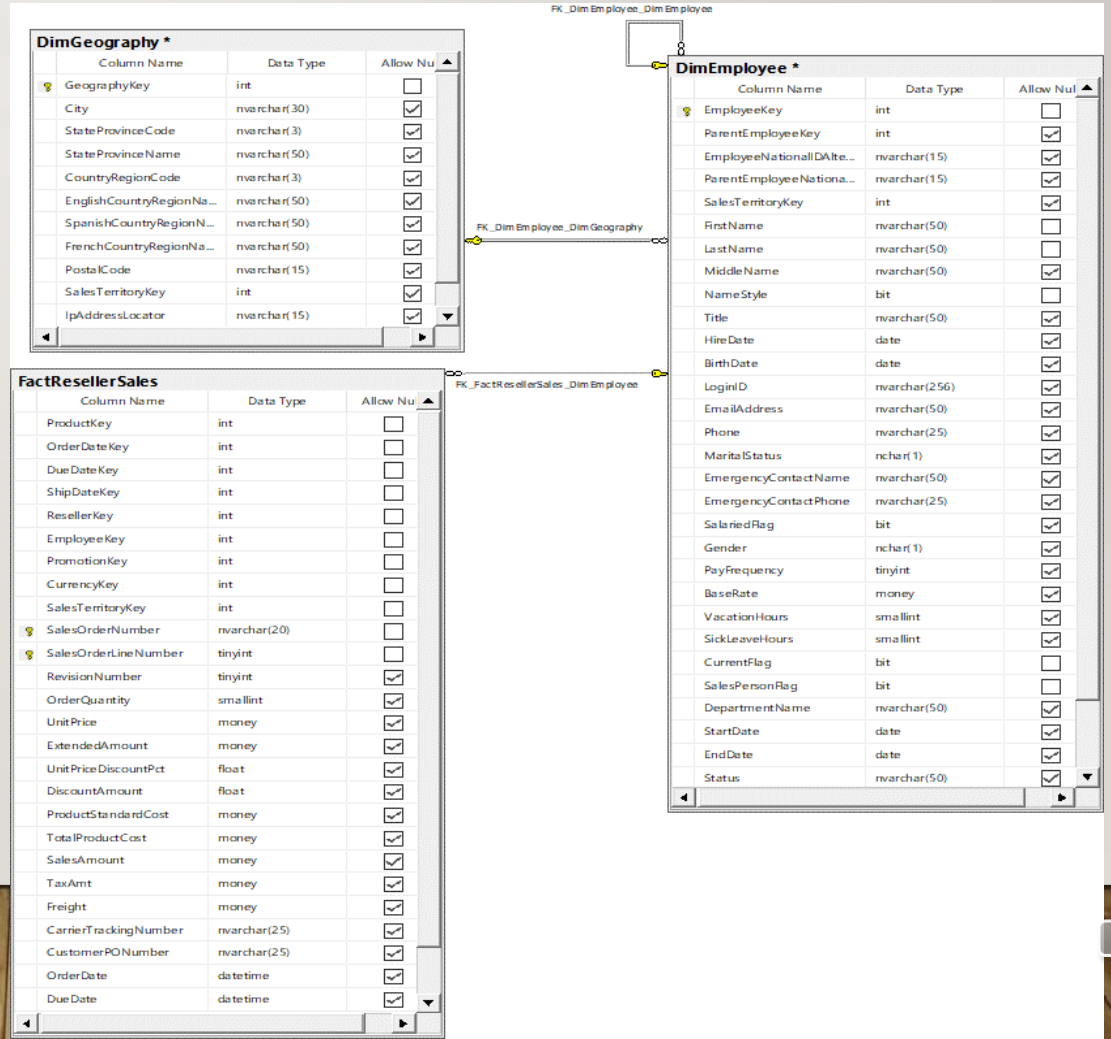
- Display for the sales territory that has Geography Key #155: the city, first, last, and middle name of the employee in charge of that territory, the number of orders they handled, and the order date of those orders made after the start of 2013

DANNY'S QUERY (COMPLEX) (WORST)

```
USE AdventureWorksDW2017

SELECT DG.City,
       CONCAT(DE.LastName, ' ', DE.FirstName, ' ', DE.MiddleName) AS Name,
       COUNT(FRS.SalesOrderNumber) AS '# of orders',
   FRS.OrderDate
FROM dbo.DimGeography AS DG
    INNER JOIN dbo.DimEmployee AS DE
        ON DE.SalesTerritoryKey = DG.SalesTerritoryKey
    INNER JOIN dbo.FactResellerSales AS FRS
        ON FRS.EmployeeKey = DE.EmployeeKey
WHERE FRS.SalesTerritoryKey = dbo.fnd_city(155) AND FRS.OrderDate > '20130101'
GROUP BY DG.City,
         FRS.OrderDate,
 CONCAT(DE.LastName, ' ', DE.FirstName, ' ', DE.MiddleName)
ORDER BY DG.City
```

DANNY'S WORST COMPLEX QUERY DIAGRAMS (KEY/STANDARD VIEWS)





DANNY'S WORST COMPLEX QUERY (COLUMNS/ORDER BY TABLES)

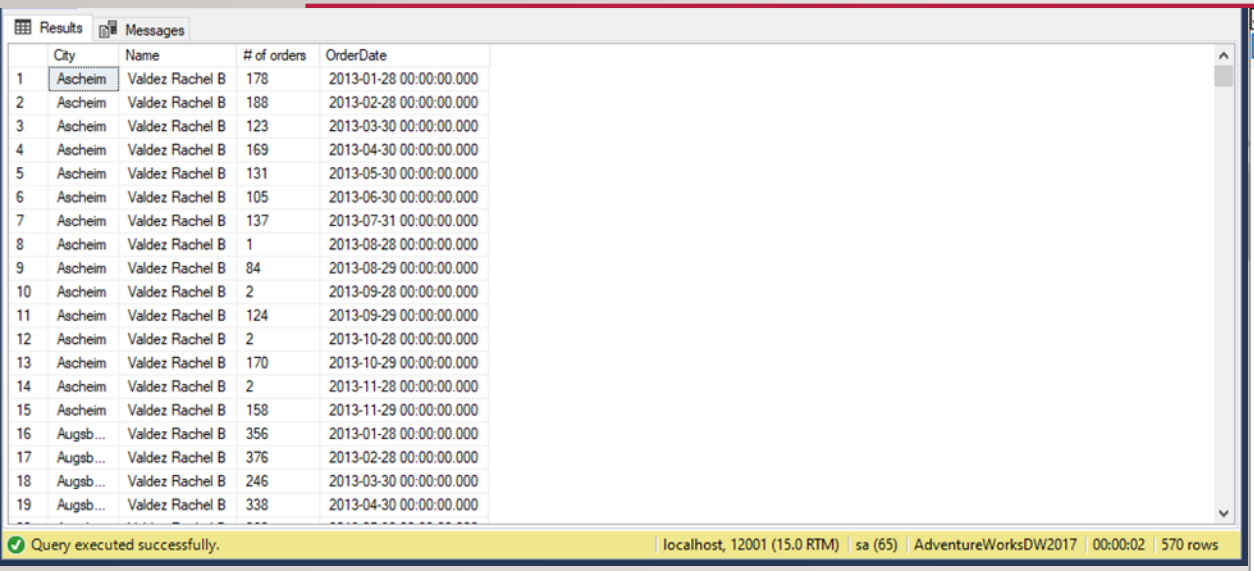
Columns from Tables

Table Name	Column Name
DimGeography	City
DimEmployee	CONCAT(LastName, FirstName, MiddleName) AS Name
FactsResellerSales	COUNT(SalesOrderNumber) AS # of Orders, OrderDate

Order By

Table Name	Column Name	Sort Order
DimGeography	City	ASC

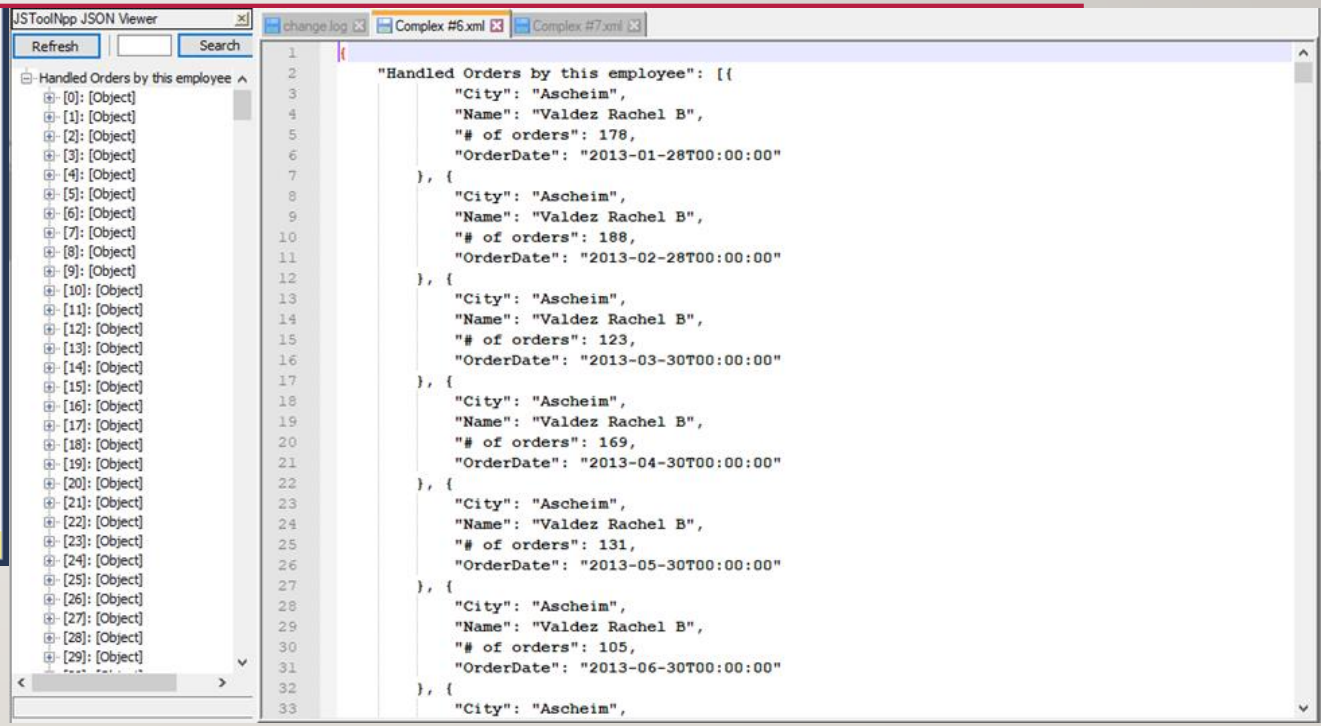
DANNY'S WORST COMPLEX QUERY (SQL/JSON OUTPUT)



Results | Messages

	City	Name	# of orders	OrderDate
1	Ascheim	Valdez Rachel B	178	2013-01-28 00:00:00.000
2	Ascheim	Valdez Rachel B	188	2013-02-28 00:00:00.000
3	Ascheim	Valdez Rachel B	123	2013-03-30 00:00:00.000
4	Ascheim	Valdez Rachel B	169	2013-04-30 00:00:00.000
5	Ascheim	Valdez Rachel B	131	2013-05-30 00:00:00.000
6	Ascheim	Valdez Rachel B	105	2013-06-30 00:00:00.000
7	Ascheim	Valdez Rachel B	137	2013-07-31 00:00:00.000
8	Ascheim	Valdez Rachel B	1	2013-08-28 00:00:00.000
9	Ascheim	Valdez Rachel B	84	2013-08-29 00:00:00.000
10	Ascheim	Valdez Rachel B	2	2013-09-28 00:00:00.000
11	Ascheim	Valdez Rachel B	124	2013-09-29 00:00:00.000
12	Ascheim	Valdez Rachel B	2	2013-10-28 00:00:00.000
13	Ascheim	Valdez Rachel B	170	2013-10-29 00:00:00.000
14	Ascheim	Valdez Rachel B	2	2013-11-28 00:00:00.000
15	Ascheim	Valdez Rachel B	158	2013-11-29 00:00:00.000
16	Augsb...	Valdez Rachel B	356	2013-01-28 00:00:00.000
17	Augsb...	Valdez Rachel B	376	2013-02-28 00:00:00.000
18	Augsb...	Valdez Rachel B	246	2013-03-30 00:00:00.000
19	Augsb...	Valdez Rachel B	338	2013-04-30 00:00:00.000

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (65) | AdventureWorksDW2017 | 00:00:02 | 570 rows



JSToolNpp JSON Viewer

Refresh | Search

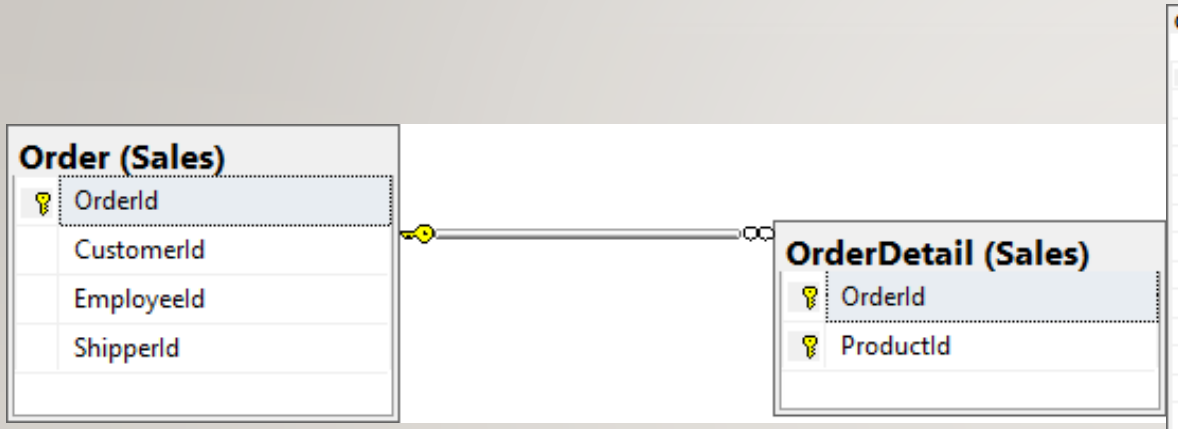
Handled Orders by this employee

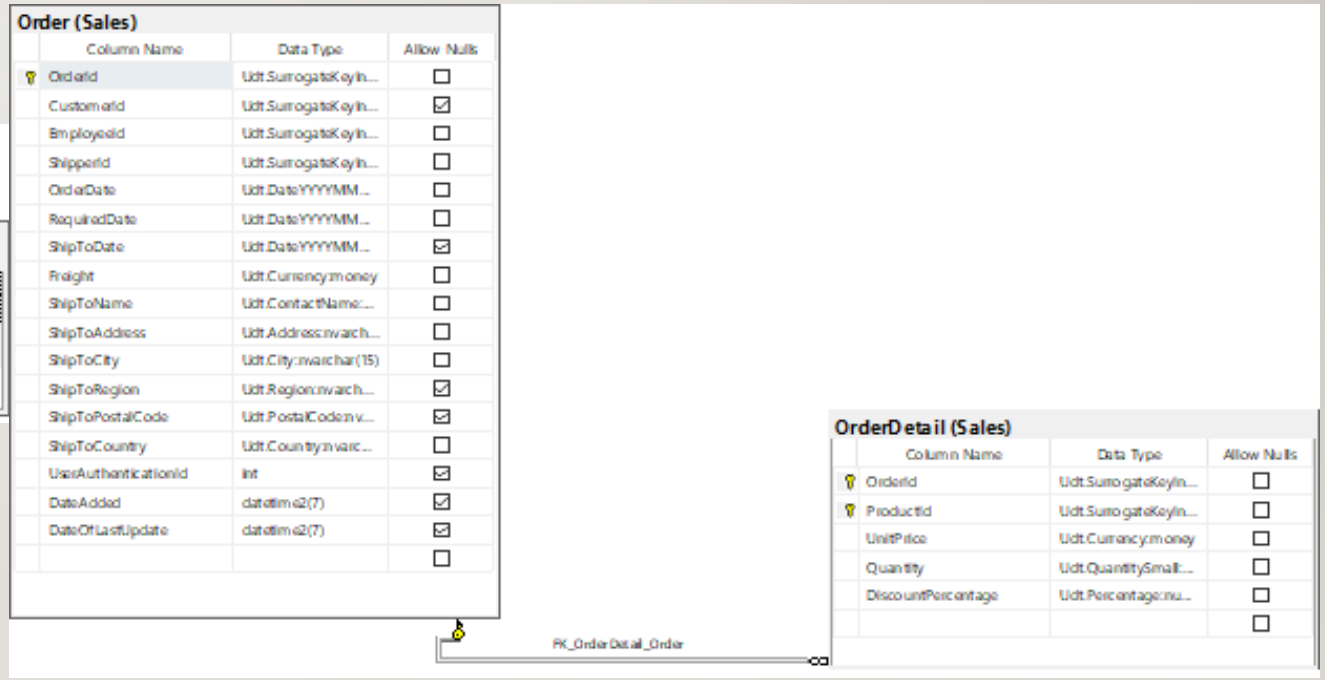
[0]: [Object]
[1]: [Object]
[2]: [Object]
[3]: [Object]
[4]: [Object]
[5]: [Object]
[6]: [Object]
[7]: [Object]
[8]: [Object]
[9]: [Object]
[10]: [Object]
[11]: [Object]
[12]: [Object]
[13]: [Object]
[14]: [Object]
[15]: [Object]
[16]: [Object]
[17]: [Object]
[18]: [Object]
[19]: [Object]
[20]: [Object]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

change.log | Complex #6.xml | Complex #7.xml


```
{
    "Handled Orders by this employee": [{
            "City": "Ascheim",
            "Name": "Valdez Rachel B",
            "# of orders": 178,
            "OrderDate": "2013-01-28T00:00:00"
        }, {
            "City": "Ascheim",
            "Name": "Valdez Rachel B",
            "# of orders": 188,
            "OrderDate": "2013-02-28T00:00:00"
        }, {
            "City": "Ascheim",
            "Name": "Valdez Rachel B",
            "# of orders": 123,
            "OrderDate": "2013-03-30T00:00:00"
        }, {
            "City": "Ascheim",
            "Name": "Valdez Rachel B",
            "# of orders": 169,
            "OrderDate": "2013-04-30T00:00:00"
        }, {
            "City": "Ascheim",
            "Name": "Valdez Rachel B",
            "# of orders": 131,
            "OrderDate": "2013-05-30T00:00:00"
        }, {
            "City": "Ascheim",
            "Name": "Valdez Rachel B",
            "# of orders": 105,
            "OrderDate": "2013-06-30T00:00:00"
        }, {
            "City": "Ascheim",
```

JAMIL'S QUERY (SIMPLE)(TOP) STANDARD/KEY VIEW





JAMIL'S QUERY (SIMPLE)(TOP) TABLES

Columns from Tables

Table Name	Column Name
Order	OrderId, CustomerId, EmployeeId, ShipToCountry
OrderDetail	UnitCost, UnitBalance

JAMIL'S QUERY (SIMPLE)(TOP) SQL CODE

- using Northwinds2020TSQLV6 make a table that joins the sales order table and Sales.OrderDetail table such that each row returns the OrderId, CustomerId, EmployeeId, ShipToCountry, ProductId, UnitPrice
- `USE Northwinds2020TSQLV6;`
-
- `SELECT E.OrderId`
- `,E.CustomerId`
- `,E.EmployeeId`
- `,E.ShipToCountry`
- `,O.ProductId`
- `,O.UnitPrice`
- `FROM Sales.[Order] AS E`
- `INNER JOIN Sales.OrderDetail AS O ON E.OrderId = O.OrderId`

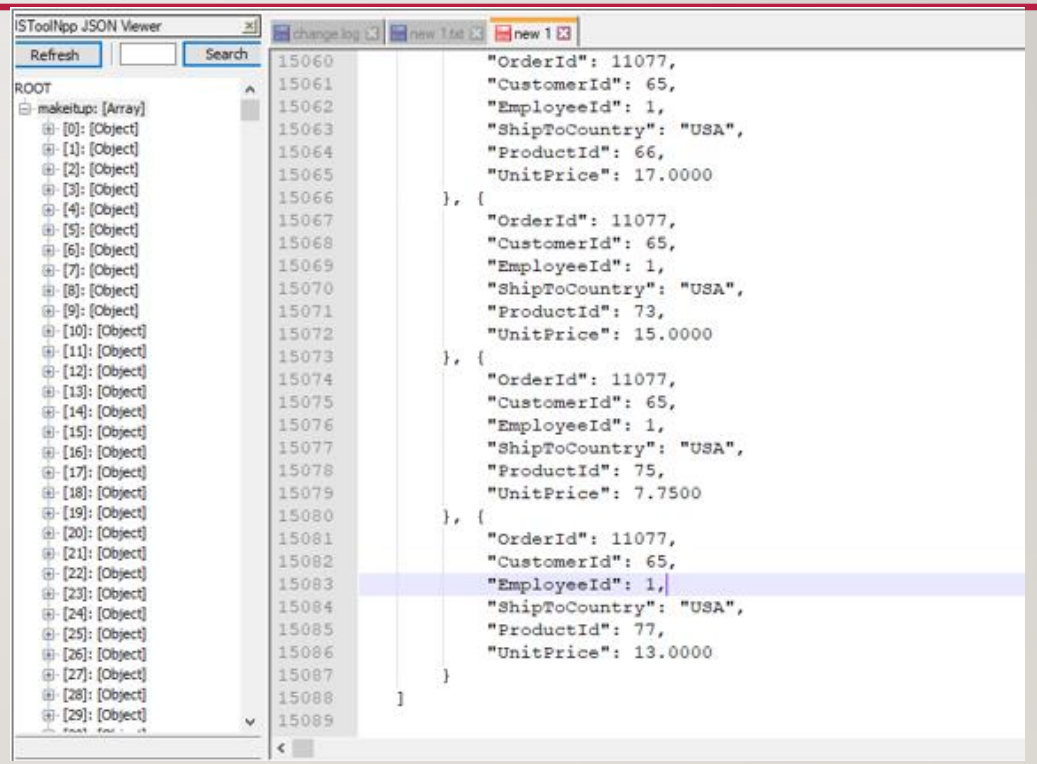
JAMIL'S QUERY (SIMPLE)(TOP) OUTPUTS

```
USE Northwinds2020TSQLV6;

SELECT E.OrderId
    ,E.CustomerId
    ,E.EmployeeId
    ,E.ShipToCountry
    ,O.ProductId
    ,O.UnitPrice
FROM Sales.[Order] AS E
INNER JOIN Sales.OrderDetail AS O ON E.OrderId = O.OrderId
--FOR JSON PATH, ROOT ('makeitup'), INCLUDE_NULL_VALUES;
```

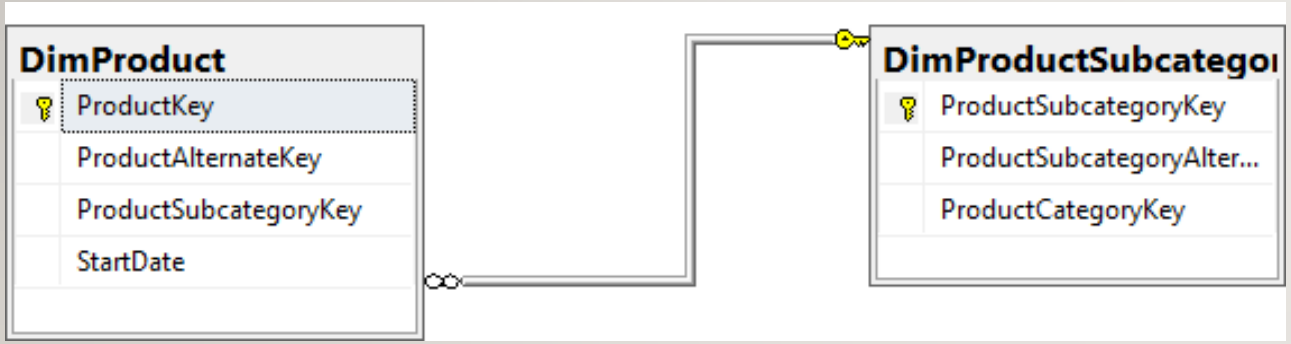
195 %

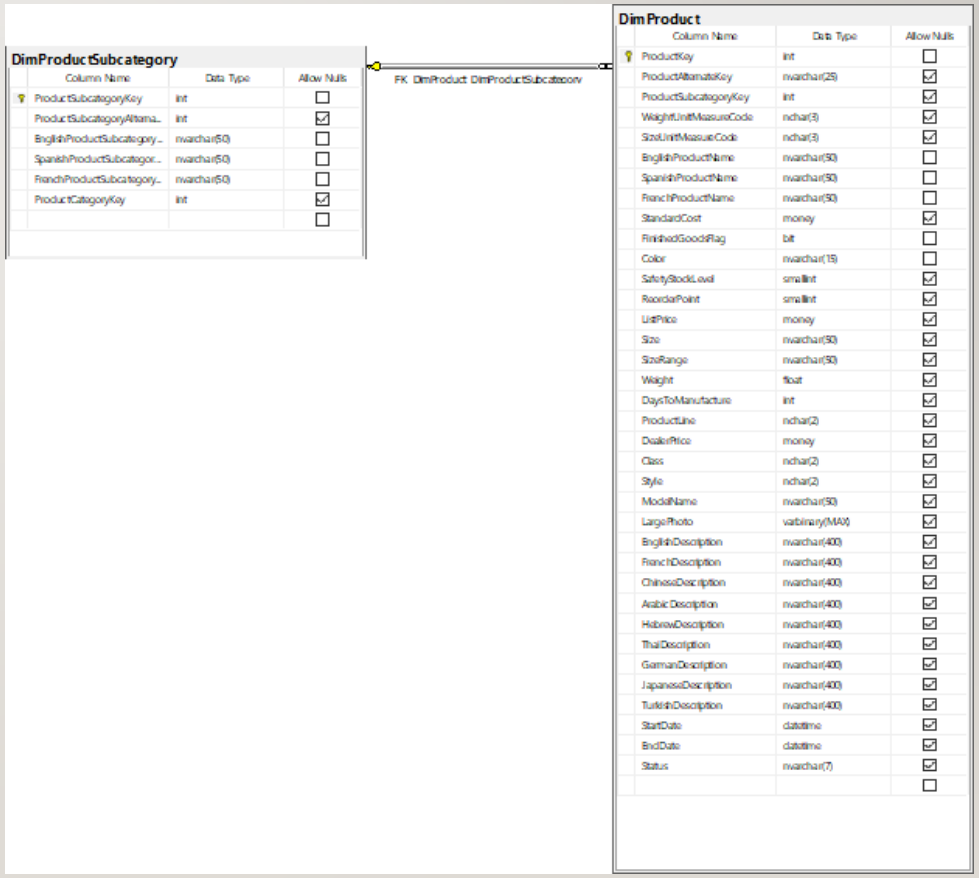
Results | Messages

	OrderId	CustomerId	EmployeeId	ShipToCountry	ProductId	UnitPrice
1	10248	85	5	France	11	14.00
2	10248	85	5	France	42	9.80
3	10248	85	5	France	72	34.80
4	10249	79	6	Germany	14	18.60
5	10249	79	6	Germany	51	42.40
6	10250	34	4	Brazil	41	7.70
7	10250	34	4	Brazil	51	42.40
8	10250	34	4	Brazil	65	16.80
9	10251	84	3	France	22	16.80
10	10251	84	3	France	57	15.60
11	10251	84	3	France	65	16.80
12	10252	76	4	Belgium	20	64.80
13	10252	76	4	Belgium	33	2.00
14	10252	76	4	Belgium	60	27.20
15	10253	34	3	Brazil	31	10.00
16	10253	34	3	Brazil	39	14.40
17	10253	34	3	Brazil	49	16.00
18	10254	14	5	Switzerland	24	3.60
19	10254	14	5	Switzerland	55	19.20



JAMIL'S QUERY (MEDIUM)(TOP) STANDARD/KEY VIEW





JAMIL'S QUERY (MEDIUM)(TOP) TABLES

Columns from Tables

Table Name	Column Name
DimProduct	EnglishProductName, SpanishProductName
DimProductSubcategory	SpanishProductSubcategoryName

Order By

Table Name	Column Name	Sort Order
DimProduct	EnglishProductName	ASC

JAMIL'S QUERY (MEDIUM)(TOP) SQL CODE

- Using AdventureWorksDW2017 make a table that outputs the english name of a product, the spanish name and its spanish subcategory name when spanish product name exists

```
USE AdventureWorksDW2017


SELECT E.EnglishProductName,
               E.SpanishProductName,
               O.SpanishProductSubcategoryName
FROM dbo.DimProduct AS E
INNER JOIN dbo.DimProductSubcategory AS O ON E.ProductSubcategoryKey = O.ProductSubcategoryKey
WHERE SpanishProductName IS NOT NULL
GROUP BY E.EnglishProductName,
               E.SpanishProductName,
               O.SpanishProductSubcategoryName
ORDER BY E.EnglishProductName
```

JAMIL'S QUERY (MEDIUM)(TOP) OUTPUTS

```sql
USE AdventureWorksDW2017

SELECT E.EnglishProductName,
    E.SpanishProductName,
    O.SpanishProductSubcategoryName
FROM dbo.DimProduct AS E
INNER JOIN dbo.DimProductSubcategory AS O ON E.ProductSubcategoryKey = O.ProductSubcategoryKey
WHERE SpanishProductName IS NOT NULL
GROUP BY E.EnglishProductName,
    E.SpanishProductName,
    O.SpanishProductSubcategoryName
ORDER BY E.EnglishProductName

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

Results | Messages

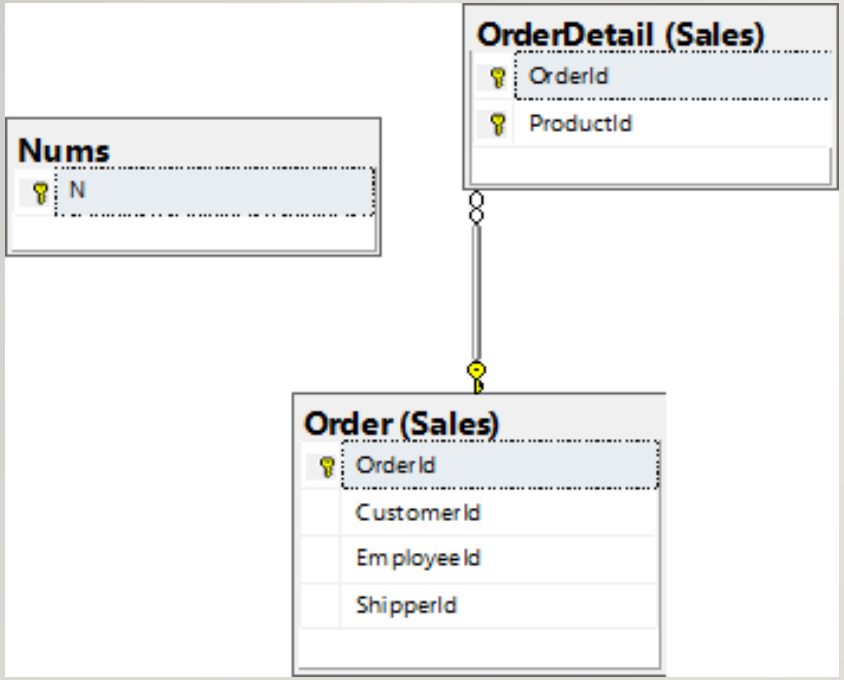
	EnglishProductName	SpanishProductName	SpanishProductSubcategoryName
1	All-Purpose Bike Stand	Soporte multiusos para bicicletas	Soporte para bicicletas
2	AWC Logo Cap		Gorra
3	Bike Wash - Dissolver	Lavado de bicicletas: disolvente	Limpiador
4	Cable Lock	Cable antirrobo	Candado
5	Chain	Cadena	Cadena
6	Classic Vest, L	Camiseta clásica, G	Camiseta
7	Classic Vest, M	Camiseta clásica, M	Camiseta
8	Classic Vest, S	Camiseta clásica, P	Camiseta
9	Fender Set - Mountain	Conjunto de guardabarros: montaña	Guardabarros
10	Front Brakes	Frenos delanteros	Frenos
11	Front Derailleur	Desviador delantero	Desviador
12	Full-Finger Gloves, L	Guantes completos, G	Guantes
13	Full-Finger Gloves, M	Guantes completos, M	Guantes
14	Full-Finger Gloves, S	Guantes completos, P	Guantes
15	Half-Finger Gloves, L		Guantes
16	Half-Finger Gloves, M		Guantes
17	Half-Finger Gloves, S		Guantes
18	Headlights - Dual-Beam	Luces: doble haz	Luz
19	Headlights - Weatherproof	Luces: resistentes al agua	Luz

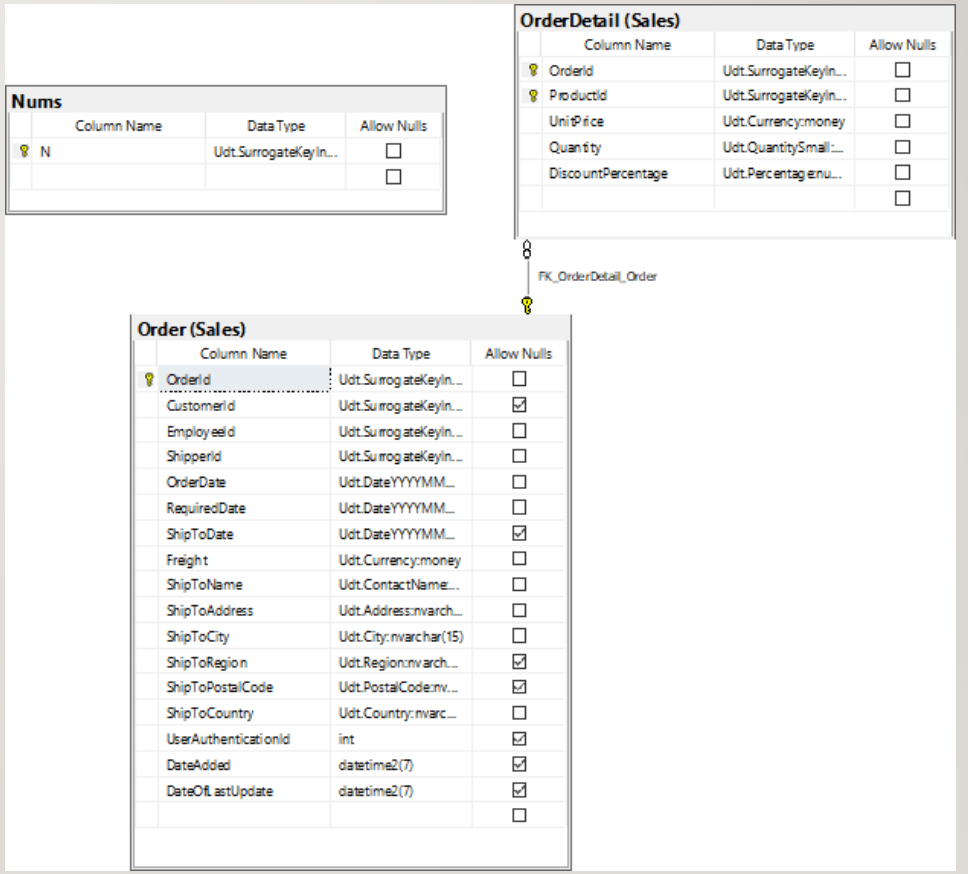
Refresh | Search

: [Array]


```
            "SpanishProductName": "Cesta de paseo, grande",
            "SpanishProductSubcategoryName": "Cesta"
        }, {
            "EnglishProductName": "Water Bottle - 30 oz.",
            "SpanishProductName": "",
            "SpanishProductSubcategoryName": "Portabotellas y botella"
        }, {
            "EnglishProductName": "Women's Mountain Shorts, L",
            "SpanishProductName": "",
            "SpanishProductSubcategoryName": "Pantalones cortos"
        }, {
            "EnglishProductName": "Women's Mountain Shorts, M",
            "SpanishProductName": "",
            "SpanishProductSubcategoryName": "Pantalones cortos"
        }, {
            "EnglishProductName": "Women's Mountain Shorts, S",
            "SpanishProductName": "",
            "SpanishProductSubcategoryName": "Pantalones cortos"
        }, {
            "EnglishProductName": "Women's Tights, L",
            "SpanishProductName": "Mallas para mujer, G",
            "SpanishProductSubcategoryName": "Mallas"
        }, {
            "EnglishProductName": "Women's Tights, M",
            "SpanishProductName": "Mallas para mujer, M",
            "SpanishProductSubcategoryName": "Mallas"
        }, {
            "EnglishProductName": "Women's Tights, S",
            "SpanishProductName": "Mallas para mujer, P",
            "SpanishProductSubcategoryName": "Mallas"
        }
    ]
```

JAMIL'S QUERY (COMPLEX)(TOP) STANDARD/KEY VIEW





JAMIL'S QUERY (COMPLEX)(TOP) TABLES

Columns from Tables

Table Name	Column Name
Order	ProductId
OrderDetail	SpanishProductSubcategoryName
Nums	n

Order By

Table Name	Column Name	Sort Order
Nums	n	ASC

JAMIL'S QUERY (COMPLEX)(TOP) SCALAR FUNCTION

```
USE Northwinds2020TSQLV6
GO
CREATE OR ALTER FUNCTION dbo.createDate (
                @numberOfMonths INT,
                @date DATE
                )
RETURNS DATE
AS
BEGIN
                DECLARE @month INT;
                DECLARE @year INT;
                SELECT @year = @numberOfMonths / 12;
                SELECT @month = (@numberOfMonths % 12);
                SELECT @date = DATEADD(year, @year, @date);
                SELECT @date = DATEADD(month, @month, @date)
                RETURN @date;
END;
```

JAMIL'S QUERY (COMPLEX)(TOP) SQL CODE

- Make a scalar function that will add n months to a enter date -- then use this function with the dbo.nums to make a table with sales.order and sales.orderdetail that creates -- new ship dates for each product where each shipment is delayed n number of months where N <= 24

```
SELECT D.n,
                O.ProductId,
                E.ShipToDate,
                dbo.createDate(D.n, E.ShipToDate) AS NewShipToDate
FROM dbo.Nums AS D
CROSS JOIN Sales.[Order] AS E
INNER JOIN Sales.OrderDetail AS O ON E.OrderId = O.OrderId
WHERE n <= 24
                AND ShipToDate IS NOT NULL
GROUP BY D.n,
                O.ProductId,
                E.ShipToDate,
                D.n,
                E.ShipToDate
ORDER BY D.n
```

JAMIL'S QUERY (COMPLEX)(TOP) OUTPUTS

```
USE Northwinds2020TSQLV6
GO

CREATE
    OR

ALTER FUNCTION dbo.createDate (
    @numberOfMonths INT
    ,@date DATE
    )
RETURNS DATE
AS
BEGIN
    DECLARE @month INT;
    DECLARE @year INT;

    SELECT @year = @numberOfMonths / 12;

    SELECT @month = (@numberOfMonths % 12);
```

132 %

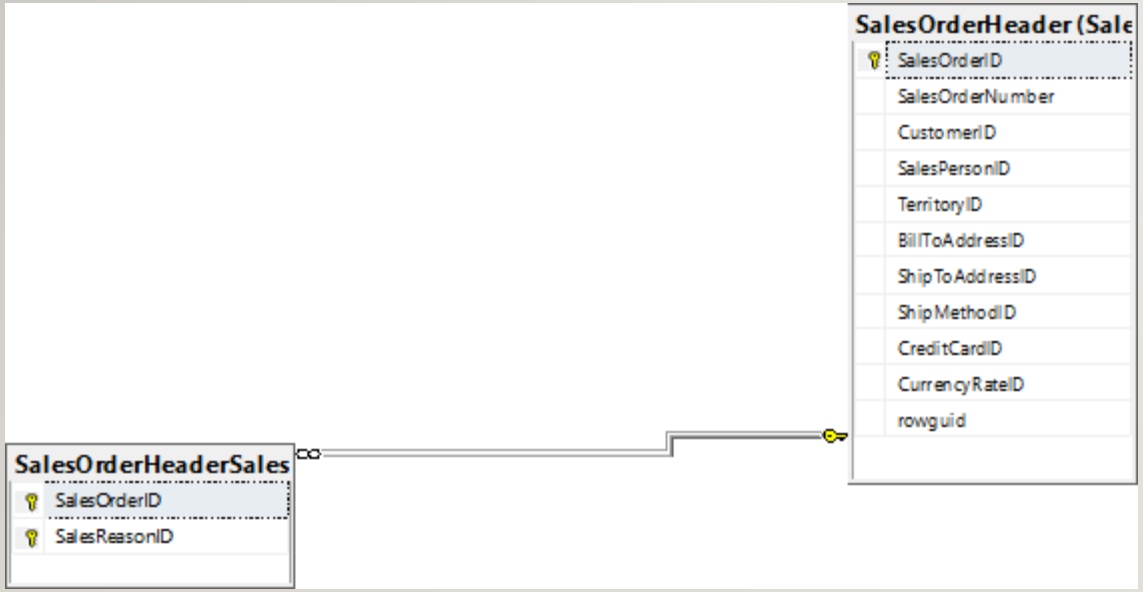
Results | Messages

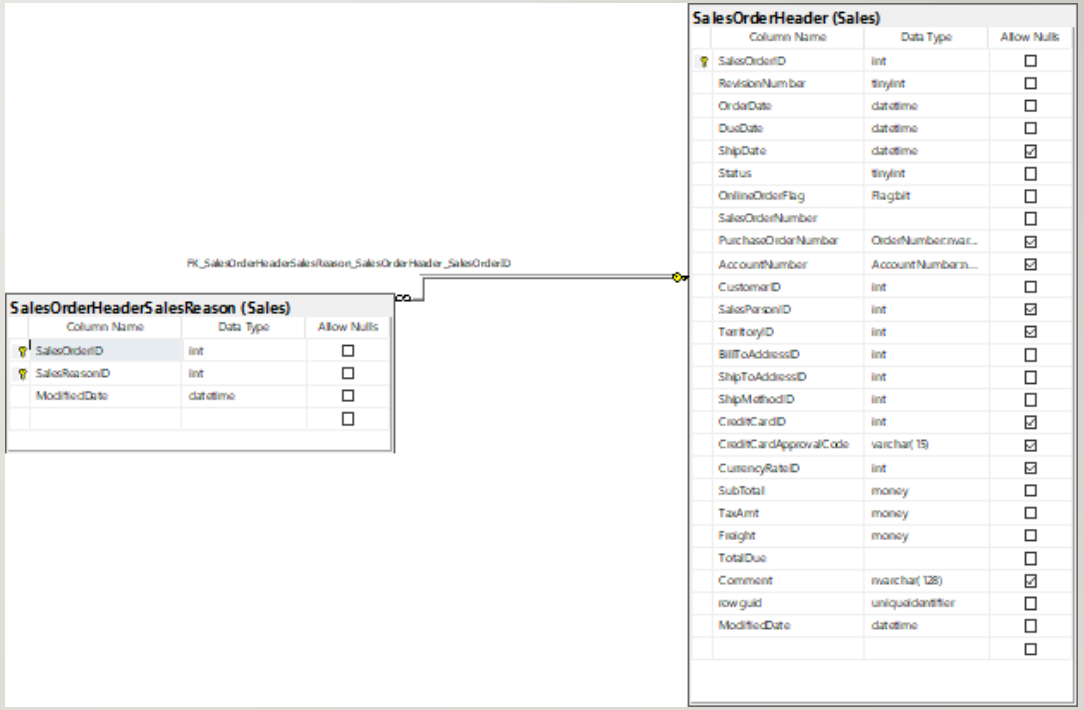
	n	ProductId	ShipToDate	NewShipToDate
1	1	28	2015-02-14	2015-03-14
2	1	55	2015-03-12	2015-04-12
3	1	29	2016-01-19	2016-02-19
4	1	56	2016-01-13	2016-02-13
5	1	11	2015-01-30	2015-02-28
6	1	7	2016-03-06	2016-04-06
7	1	31	2015-09-23	2015-10-23
8	1	77	2015-04-07	2015-05-07
9	1	48	2015-04-22	2015-05-22
10	1	40	2015-03-14	2015-04-14
11	1	60	2015-11-20	2015-12-20
12	1	17	2016-02-18	2016-03-18
13	1	31	2015-04-07	2015-05-07
14	1	44	2016-05-01	2016-06-01
15	1	40	2014-08-12	2014-09-12
16	1	11	2015-04-02	2015-05-02
17	1	5	2015-11-05	2015-12-05
18	1	73	2015-08-21	2015-09-21

Refresh | Search

```
243812          }, {
243813              "n": 24,
243814              "ProductId": 62,
243815              "ShipToDate": "2016-01-05",
243816              "NewShipToDate": "2018-01-05"
243817          }, {
243818              "n": 24,
243819              "ProductId": 16,
243820              "ShipToDate": "2016-03-10",
243821              "NewShipToDate": "2018-03-10"
243822          }, {
243823              "n": 24,
243824              "ProductId": 26,
243825              "ShipToDate": "2015-12-04",
243826              "NewShipToDate": "2017-12-04"
243827          }, {
243828              "n": 24,
243829              "ProductId": 2,
243830              "ShipToDate": "2014-07-15",
243831              "NewShipToDate": "2016-07-15"
243832          }, {
243833              "n": 24,
243834              "ProductId": 42,
243835              "ShipToDate": "2016-04-16",
243836              "NewShipToDate": "2018-04-16"
243837          }, {
243838              "n": 24,
243839              "ProductId": 24,
243840              "ShipToDate": "2016-03-31",
243841              "NewShipToDate": "2018-03-31"
243842          }
243843      ]
243844
```

JAMIL'S QUERY (SIMPLE)(WORST) STANDARD/KEY VIEW





JAMIL'S QUERY (SIMPLE)(WORST) TABLES

Columns from Tables

Table Name	Column Name
SalesOrderHeaderSalesReason	SalesOrderId, CustomerId, AccountNumber
SalesOrderHeader	ModifiedDate

JAMIL'S QUERY (SIMPLE)(WORST) SQL CODE

- Using AdventureWorks2017 make a table that joins the SalesOrderHeaderSalesReason table along with the SalesOrderHeader such that the table shows the SalesOrderID, SalesReasonID, CustomerID, ModifiedDate, AccountNumber

```
USE AdventureWorks2017


SELECT O.SalesOrderID,
   O.CustomerID,
   O.AccountNumber
FROM Sales.SalesOrderHeaderSalesReason AS E
LEFT JOIN sales.SalesOrderHeader AS O ON E.ModifiedDate = O.ModifiedDate
WHERE O.CustomerID < 12000
   AND E.SalesOrderID IS NOT NULL
```

JAMIL'S QUERY (SIMPLE)(WORST) OUTPUTS

```
USE AdventureWorks2017

SELECT O.SalesOrderID
    ,O.CustomerID
    ,O.AccountNumber
FROM Sales.SalesOrderHeaderSalesReason AS E
LEFT JOIN sales.SalesOrderHeader AS O ON E.ModifiedDate = O.ModifiedDate
where O.CustomerID < 15000 and E.SalesOrderID IS NOT NULL
--FOR JSON PATH, ROOT ('temp'), INCLUDE_NULL_VALUES;
```

76 %

Results | Messages

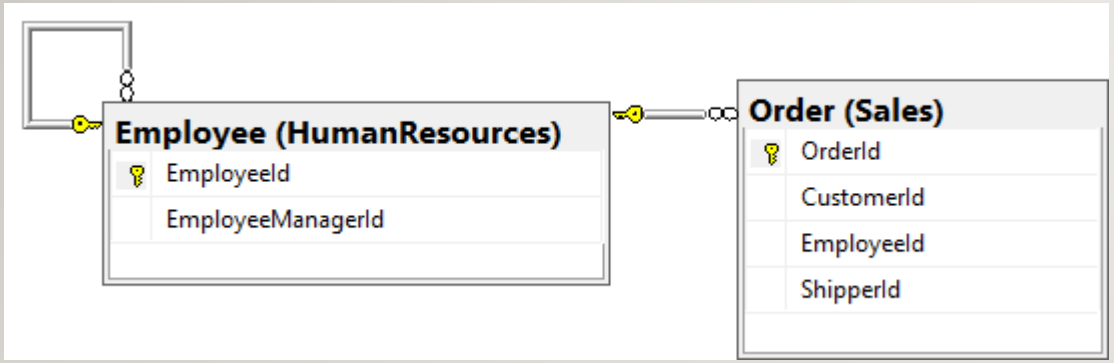
	SalesOrderID	CustomerID	AccountNumber
1	43700	14501	10-4030-014501
2	43701	11003	10-4030-011003
3	43700	14501	10-4030-014501
4	43701	11003	10-4030-011003
5	43700	14501	10-4030-014501
6	43701	11003	10-4030-011003
7	43700	14501	10-4030-014501
8	43701	11003	10-4030-011003
9	43704	11005	10-4030-011005
10	43705	11011	10-4030-011011
11	43704	11005	10-4030-011005
12	43705	11011	10-4030-011011
13	43704	11005	10-4030-011005
14	43705	11011	10-4030-011011
15	43704	11005	10-4030-011005
16	43705	11011	10-4030-011011
17	43704	11005	10-4030-011005
18	43705	11011	10-4030-011011
19	43704	11005	10-4030-011005

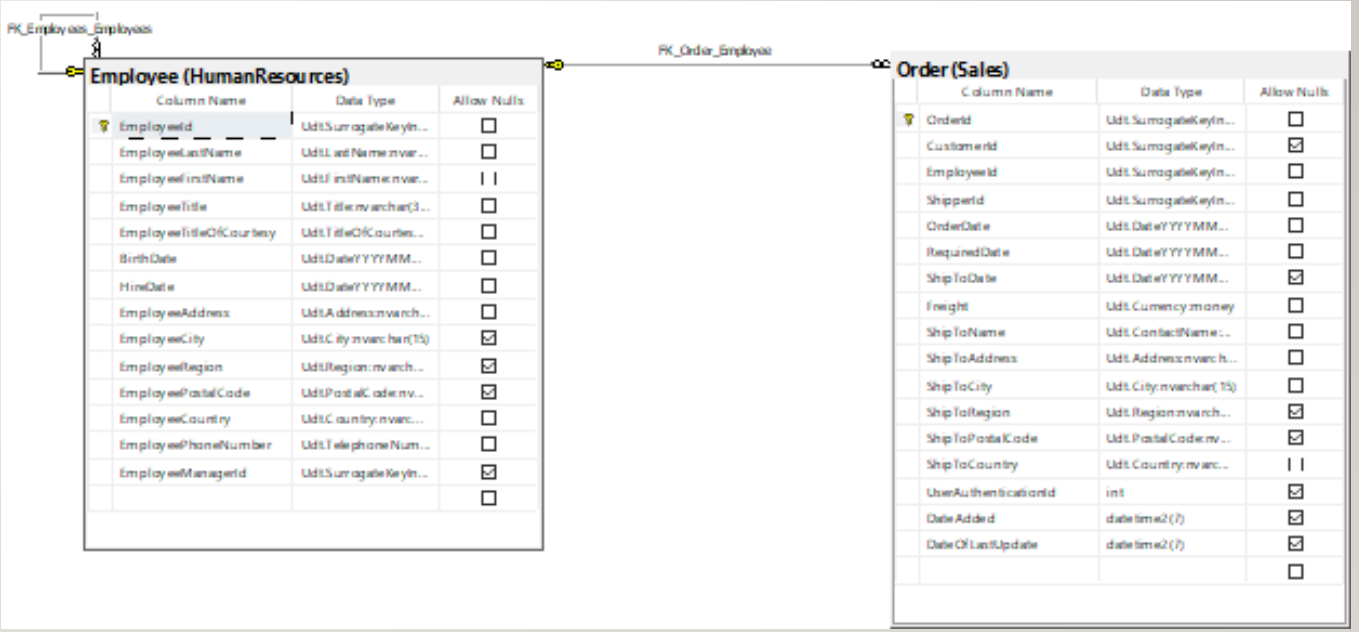
Refresh | Search

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

```
615191            "SalesOrderID": 74887,
615192            "CustomerID": 11711,
615193            "AccountNumber": "10-4030-011711"
615194        }, {
615195            "SalesOrderID": 74908,
615196            "CustomerID": 11013,
615197            "AccountNumber": "10-4030-011013"
615198        }, {
615199            "SalesOrderID": 74909,
615200            "CustomerID": 11501,
615201            "AccountNumber": "10-4030-011501"
615202        }, {
615203            "SalesOrderID": 74880,
615204            "CustomerID": 11188,
615205            "AccountNumber": "10-4030-011188"
615206        }, {
615207            "SalesOrderID": 74887,
615208            "CustomerID": 11711,
615209            "AccountNumber": "10-4030-011711"
615210        }, {
615211            "SalesOrderID": 74908,
615212            "CustomerID": 11013,
615213            "AccountNumber": "10-4030-011013"
615214        }, {
615215            "SalesOrderID": 74909,
615216            "CustomerID": 11501,
615217            "AccountNumber": "10-4030-011501"
615218        }
615219    ]
615220 }
```

JAMIL'S QUERY (MEDIUM)(WORST) STANDARD/KEY VIEW





JAMIL'S QUERY (MEDIUM)(WORST) TABLES

Columns from Tables

Table Name	Column Name
Order	EmployeeId
Employee	EmployeeCity

JAMIL'S QUERY (MEDIUM)(WORST)SQL CODE

Using northwinds create a table that shows the number of employees in each of their respective city

```
USE Northwinds2020TSQLV6;


SELECT O.EmployeeCity
        ,count(E.EmployeeId) AS NumberOfEmployees
FROM Sales.[Order] AS E
INNER JOIN HumanResources.Employee AS O ON E.EmployeeId = O.EmployeeId
GROUP BY O.EmployeeCity
```

JAMIL'S QUERY (MEDIUM)(WORST) OUTPUTS

```sql
USE Northwinds2020TSQLV6;

SELECT O.EmployeeCity
    ,count(E.EmployeeId) AS NumberOfEmployees
FROM Sales.[Order] AS E
INNER JOIN HumanResources.Employee AS O ON E.EmployeeId = O.EmployeeId
GROUP BY O.EmployeeCity
--FOR JSON PATH, ROOT('temp2'), INCLUDE_NULL_VALUES;
```

160 %

Results | Messages

	EmployeeCity	NumberOfEmployees
1	Kirkland	127
2	London	224
3	Redmond	156
4	Seattle	227
5	Tacoma	96

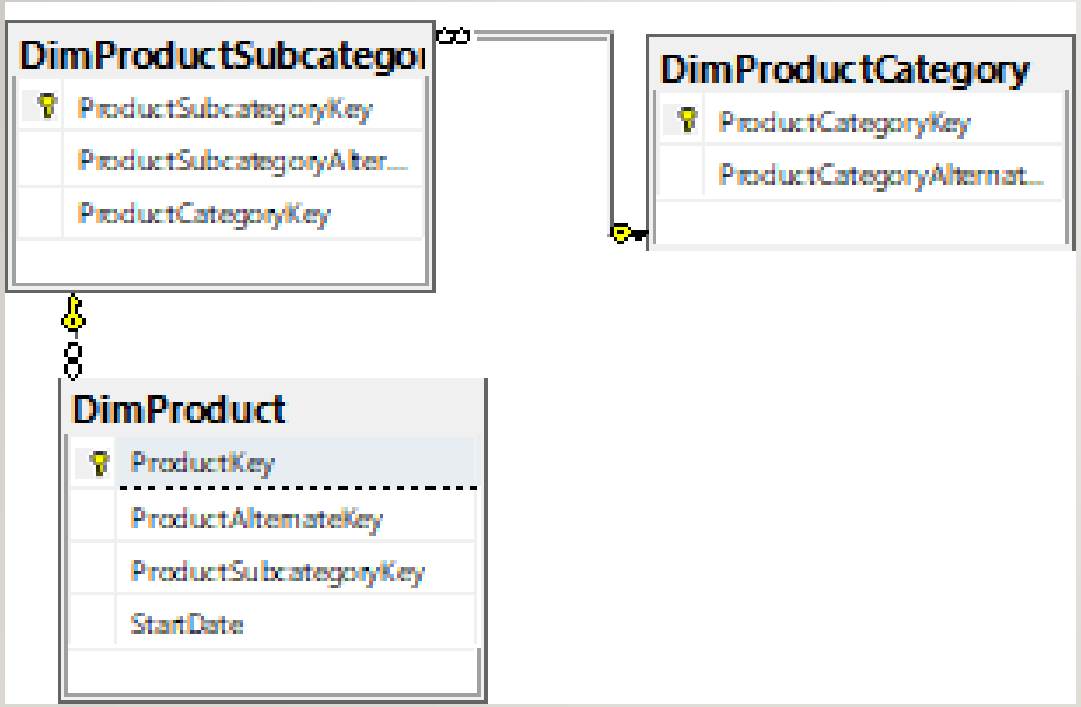
Refresh | Search

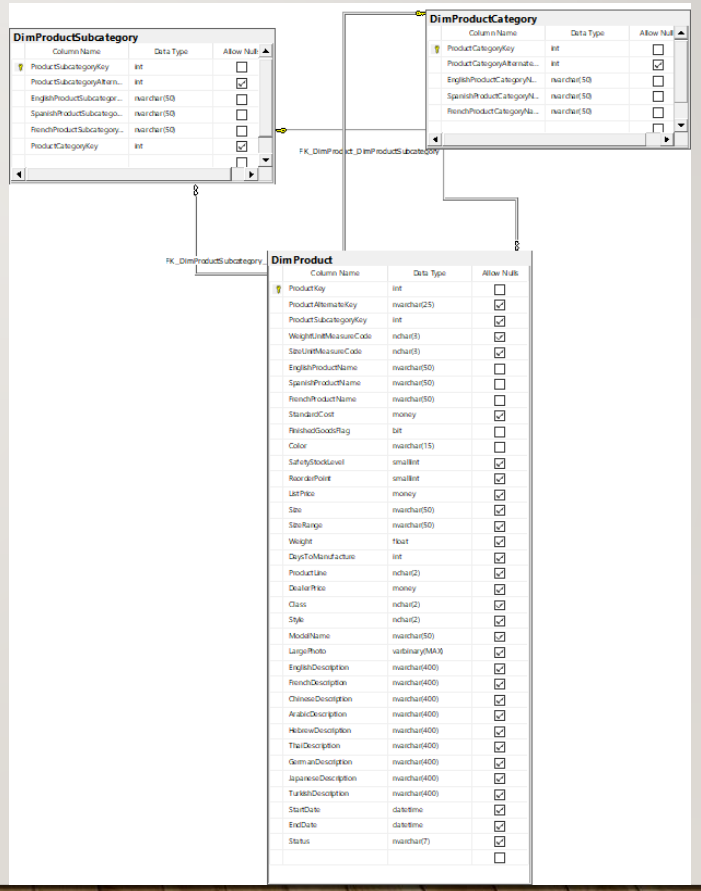
ROOT

- temp2: [Array]
  - [0]: [Object]
  - [1]: [Object]
  - [2]: [Object]
  - [3]: [Object]
  - [4]: [Object]

```
"temp2": [{
        "EmployeeCity": "Kirkland",
        "NumberOfEmployees": 127
    }, {
        "EmployeeCity": "London",
        "NumberOfEmployees": 224
    }, {
        "EmployeeCity": "Redmond",
        "NumberOfEmployees": 156
    }, {
        "EmployeeCity": "Seattle",
        "NumberOfEmployees": 227
    }, {
        "EmployeeCity": "Tacoma",
        "NumberOfEmployees": 96
    }
]
```

JAMIL'S QUERY (COMPLEX)(WORST) STANDARD/KEY VIEW





JAMIL'S QUERY (COMPLEX)(WORST) TABLES

Columns from Tables

Table Name	Column Name
DimProduct	EnglishProductName, SpanishProductName
DimProductSubcategory	SpanishProductSubcategoryName, EnglishProductSubcategoryName
DimProductCategory	EnglishProductCategoryName, SpanishProductCategoryName

Order By

Table Name	Column Name	Sort Order
DimProduct	EnglishProductNam e	ASC

JAMIL'S QUERY (COMPLEX)(WORST) SCALAR FUNCTION

```
USE AdventureWorksDW2017;
GO


CREATE OR ALTER FUNCTION dbo.compProductName (
                @ProductName NVARCHAR(50),
                @productcategoryname NVARCHAR(50),
                @productsubcategoryname NVARCHAR(50)
                )
RETURNS NVARCHAR(50)
AS
BEGIN
                RETURN @ProductName + ' ' + @productcategoryname + ' ' + @productsubcategoryname;
END;
GO
```

JAMIL'S QUERY (COMPLEX)(WORST) SQL CODE

- Create a custom scalar function that combines the products name, category name, and subcategory name and returns them all together and then make a table with two columns that outputs the english and spanish full name

```
USE AdventureWorksDW2017


SELECT dbo.compProductName(E.EnglishProductName, O.EnglishProductSubcategoryName, D.EnglishProductCategoryName) AS FullEnglishName,
                dbo.compProductName(E.SpanishProductName, O.SpanishProductSubcategoryName, D.SpanishProductCategoryName) AS FullSpanishName
FROM dbo.DimProduct AS E
INNER JOIN dbo.DimProductSubcategory AS O ON E.ProductSubcategoryKey = O.ProductSubcategoryKey
INNER JOIN dbo.DimProductCategory AS D ON D.ProductCategoryKey = O.ProductCategoryKey
WHERE SpanishProductName IS NOT NULL
GROUP BY E.EnglishProductName,
                E.SpanishProductName,
                O.SpanishProductSubcategoryName,
                D.SpanishProductCategoryName,
                O.EnglishProductSubcategoryName,
```

JAMIL'S QUERY (COMPLEX)(WORST) OUTPUTS

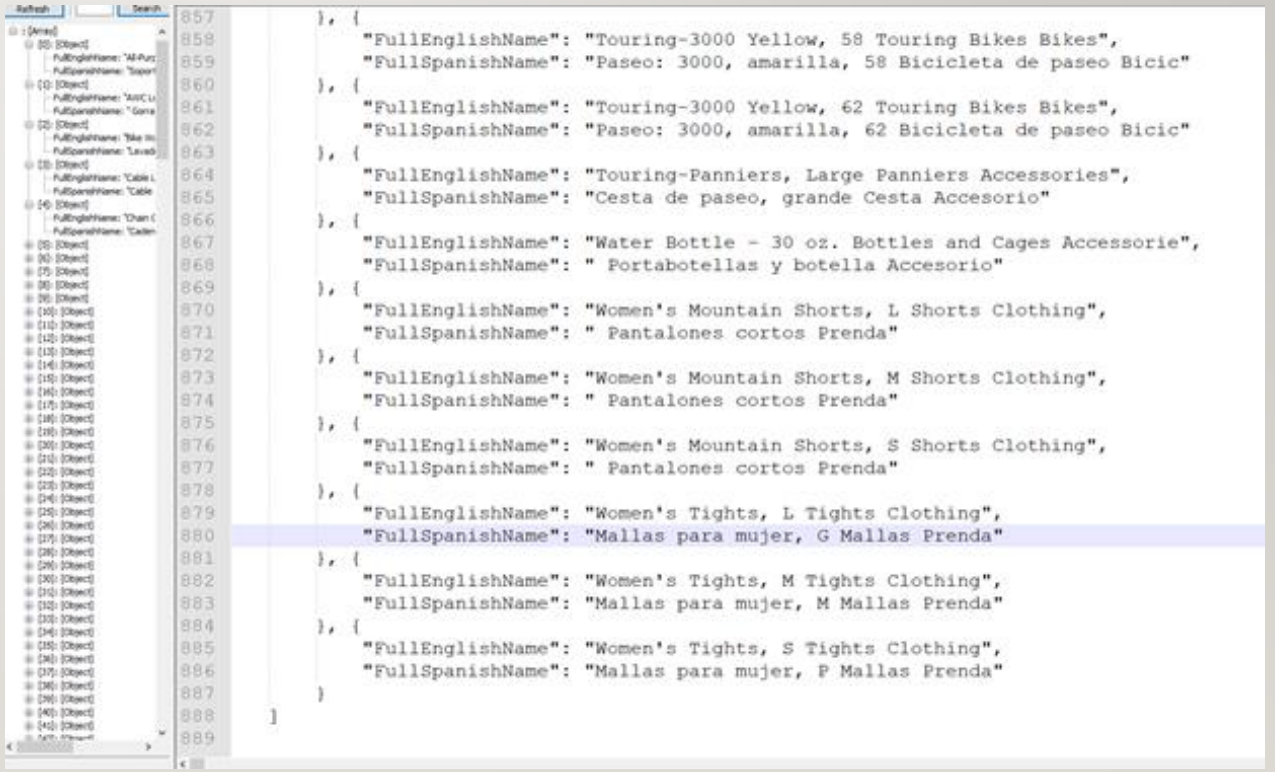
```
USE AdventureWorksDW2017;
GO

CREATE
    OR

ALTER FUNCTION dbo.compProductName (
    @ProductName NVARCHAR(50)
    ,@productcategoryname NVARCHAR(50)
    ,@productsubcategoryname NVARCHAR(50)
    )
RETURNS NVARCHAR(50)
AS
BEGIN
    RETURN @ProductName + ' ' + @productcategoryname + ' ' + @productsubcategoryname;
END;
```

160 %

Results | Messages

	FullEnglishName	FullSpanishName
1	All-Purpose Bike Stand Bike Stands Accessories	Soporte multiusos para bicicletas Soporte para bic
2	AWC Logo Cap Caps Clothing	Gorra Prenda
3	Bike Wash - Dissolver Cleaners Accessories	Lavado de bicicletas: disolvente Limpiador Acesor
4	Cable Lock Locks Accessories	Cable antirrobo Candado Accesorio
5	Chain Chains Components	Cadena Cadena Componente
6	Classic Vest, L Vests Clothing	Camiseta clásica, G Camiseta Prenda
7	Classic Vest, M Vests Clothing	Camiseta clásica, M Camiseta Prenda
8	Classic Vest, S Vests Clothing	Camiseta clásica, P Camiseta Prenda
9	Fender Set - Mountain Fenders Accessories	Conjunto de guardabarros: montaña Guardabarros Acc
10	Front Brakes Brakes Components	Frenos delanteros Frenos Componente
11	Front Derailleur Derailleurs Components	Desviador delantero Desviador Componente
12	Full-Finger Gloves, L Gloves Clothing	Guantes completos, G Guantes Prenda
13	Full-Finger Gloves, M Gloves Clothing	Guantes completos, M Guantes Prenda
14	Full-Finger Gloves, S Gloves Clothing	Guantes completos, P Guantes Prenda
15	Half-Finger Gloves, L Gloves Clothing	Guantes Prenda
16	Half-Finger Gloves, M Gloves Clothing	Guantes Prenda
17	Half-Finger Gloves, S Gloves Clothing	Guantes Prenda
18	Headlights - Dual-Beam Lights Accessories	Luces: doble haz Luz Accesorio
19	Headlights - Weatherproof Lights Accessories	Luces: resistentes al agua Luz Accesorio

Query executed successfully.



```
        }, {
            "FullEnglishName": "Touring-3000 Yellow, 58 Touring Bikes Bikes",
            "FullSpanishName": "Paseo: 3000, amarilla, 58 Bicicleta de paseo Bicic"
        }, {
            "FullEnglishName": "Touring-3000 Yellow, 62 Touring Bikes Bikes",
            "FullSpanishName": "Paseo: 3000, amarilla, 62 Bicicleta de paseo Bicic"
        }, {
            "FullEnglishName": "Touring-Panniers, Large Panniers Accessories",
            "FullSpanishName": "Cesta de paseo, grande Cesta Accesorio"
        }, {
            "FullEnglishName": "Water Bottle - 30 oz. Bottles and Cages Accessorie",
            "FullSpanishName": " Portabotellas y botella Accesorio"
        }, {
            "FullEnglishName": "Women's Mountain Shorts, L Shorts Clothing",
            "FullSpanishName": " Pantalones cortos Prenda"
        }, {
            "FullEnglishName": "Women's Mountain Shorts, M Shorts Clothing",
            "FullSpanishName": " Pantalones cortos Prenda"
        }, {
            "FullEnglishName": "Women's Mountain Shorts, S Shorts Clothing",
            "FullSpanishName": " Pantalones cortos Prenda"
        }, {
            "FullEnglishName": "Women's Tights, L Tights Clothing",
            "FullSpanishName": "Mallas para mujer, G Mallas Prenda"
        }, {
            "FullEnglishName": "Women's Tights, M Tights Clothing",
            "FullSpanishName": "Mallas para mujer, M Mallas Prenda"
        }, {
            "FullEnglishName": "Women's Tights, S Tights Clothing",
            "FullSpanishName": "Mallas para mujer, P Mallas Prenda"
        }
    ]
```

HARJIT'S WORST QUERY: PREPOSITION (1/3)

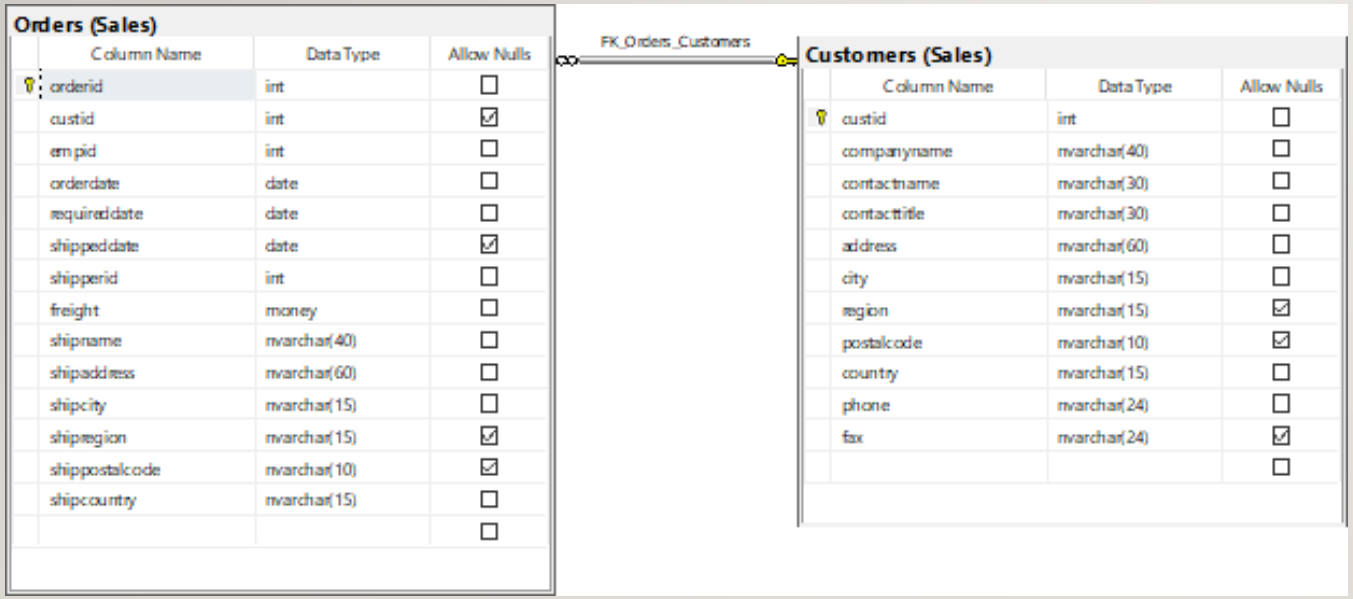
Preposition with databases and tables:

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

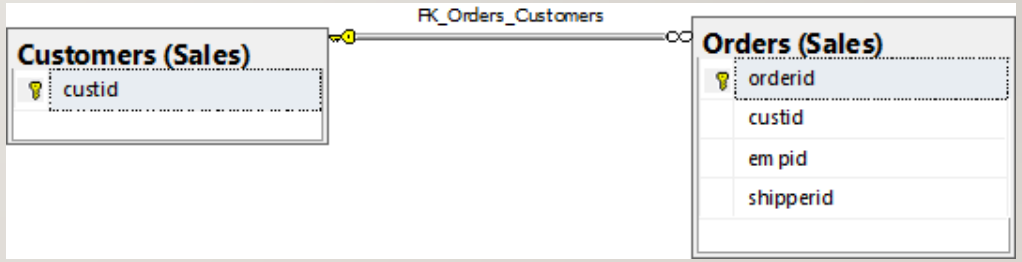
Use TSQLV4 Database, Sales.Customer, Sales.Orders tables

HARJIT'S WORST QUERY: KEYS (1/3)

Standard View



Key View



HARJIT'S WORST QUERY: TABLES (1/3)

Column Name

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

Order By

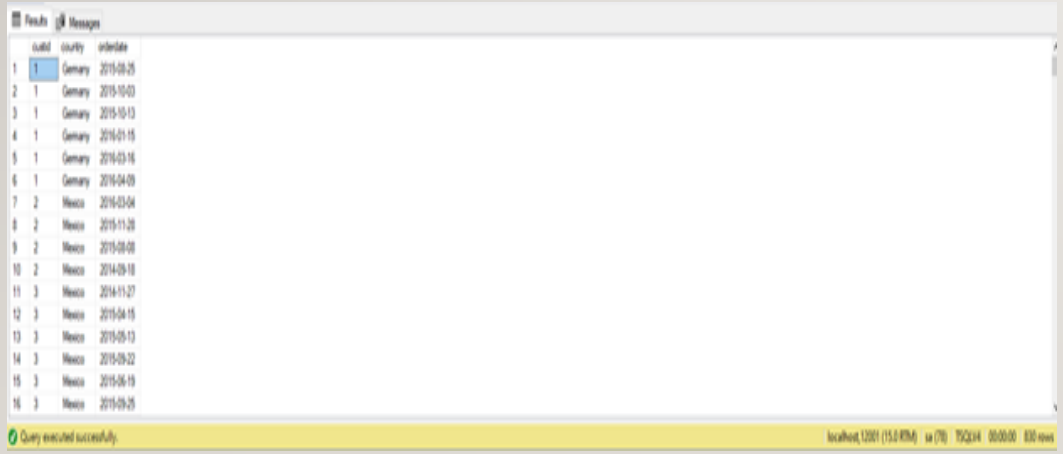
Table Name	Column Name	Sort Order
Sales.Customers	custid	ASC
Sales.Customers	country	ASC
Sales.Orders	orderdate	ASC

HARJIT'S WORST QUERY: SAMPLE CODE (1/3)

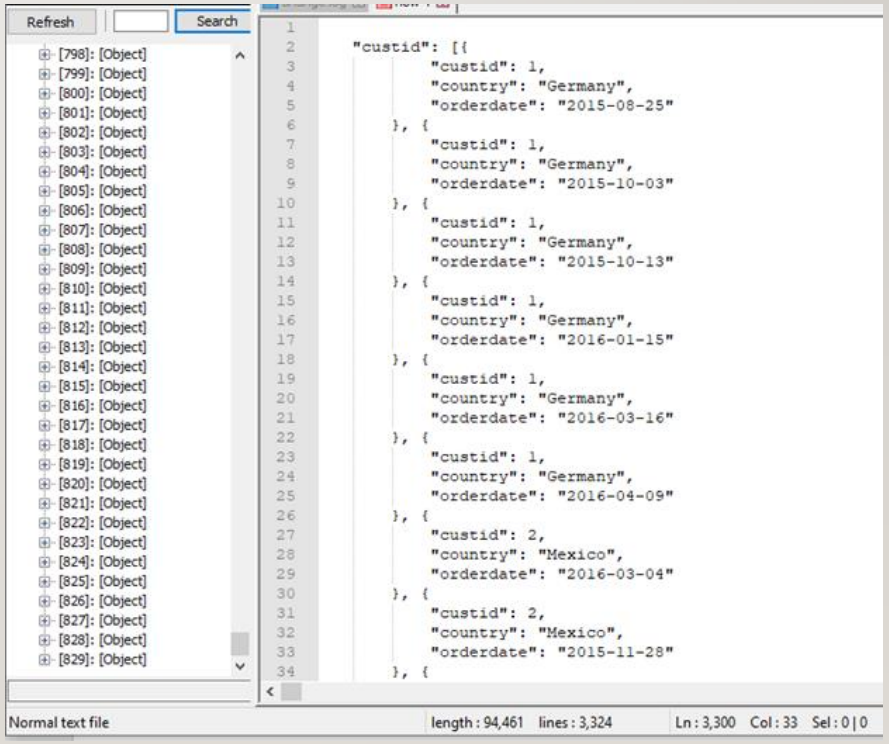
```
USE TSQLV4;
SELECT E.custid,
       E.country,
       F.orderdate
FROM Sales.Customers AS E
    INNER JOIN Sales.Orders AS F
        ON F.custid = E.custid
ORDER BY E.custid
--FOR JSON PATH, ROOT ('custid'), INCLUDE_NULL_VALUES;
```

HARJIT'S WORST QUERY: SAMPLE OUTPUT (1/3)

Sample relational output



Sample JSON output

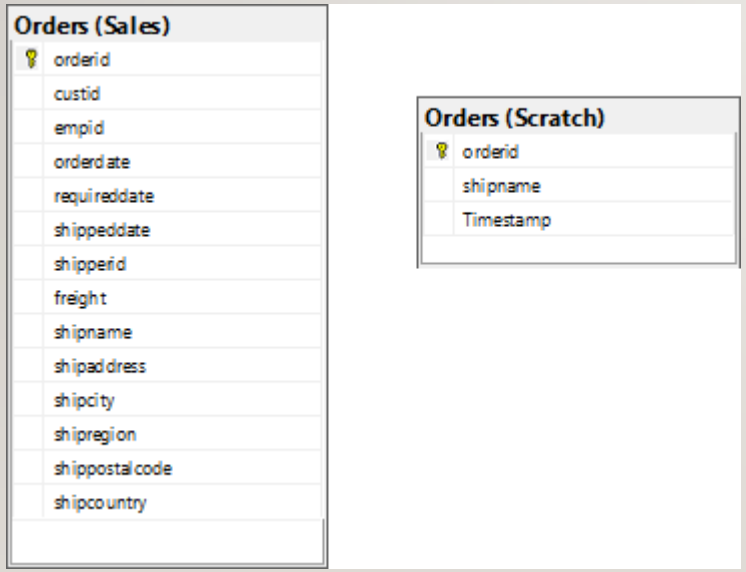


HARJIT'S WORST QUERY: PREPOSITION (2/3)

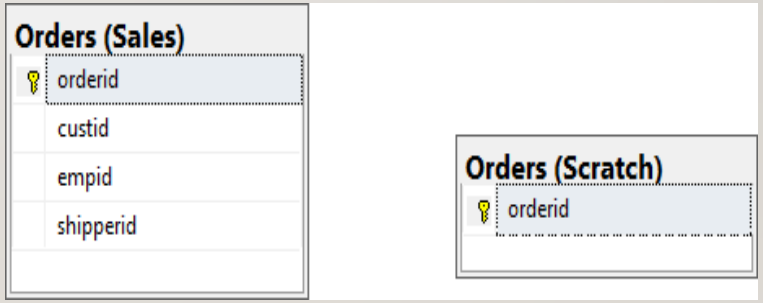
- Performs an inner join and returns order id, customer id, the maximum freight for that order with its destination
- Use TSQLV6 database with Sales.Orders and Scratch.Orders tables

HARJIT'S WORST QUERY: KEYS (2/3)

Standard View



Key view



HARJIT'S WORST QUERY: TABLES (2/3)

Columns from tables with order by

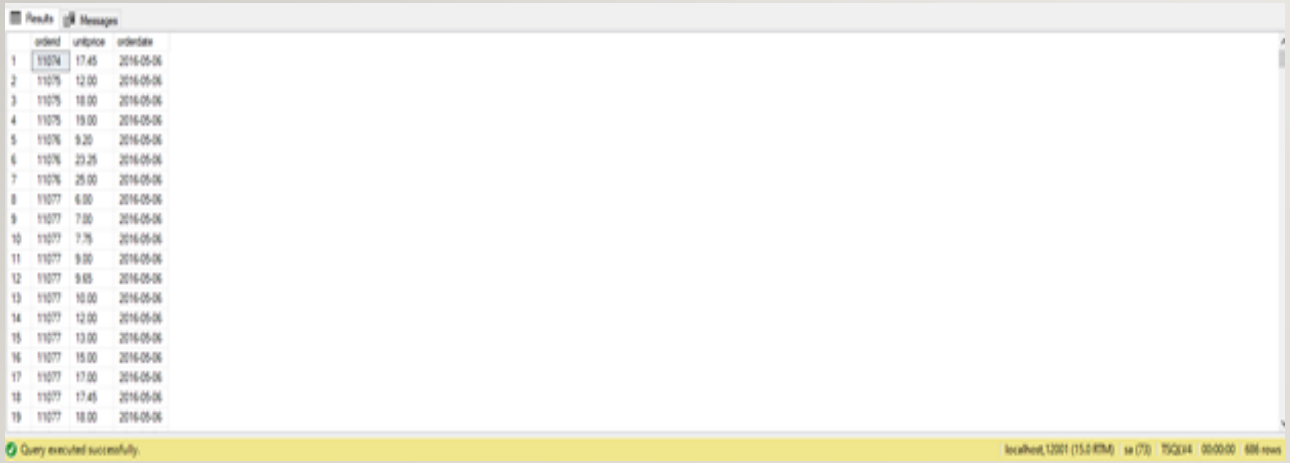
Table Name	Column Names	Order By
Sales.Orders	Orderid	ASC
	custid	ASC
	freight	ASC
Scratch.Orders	shipto	ASC

HARJIT'S WORST QUERY: CODE (2/3)

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

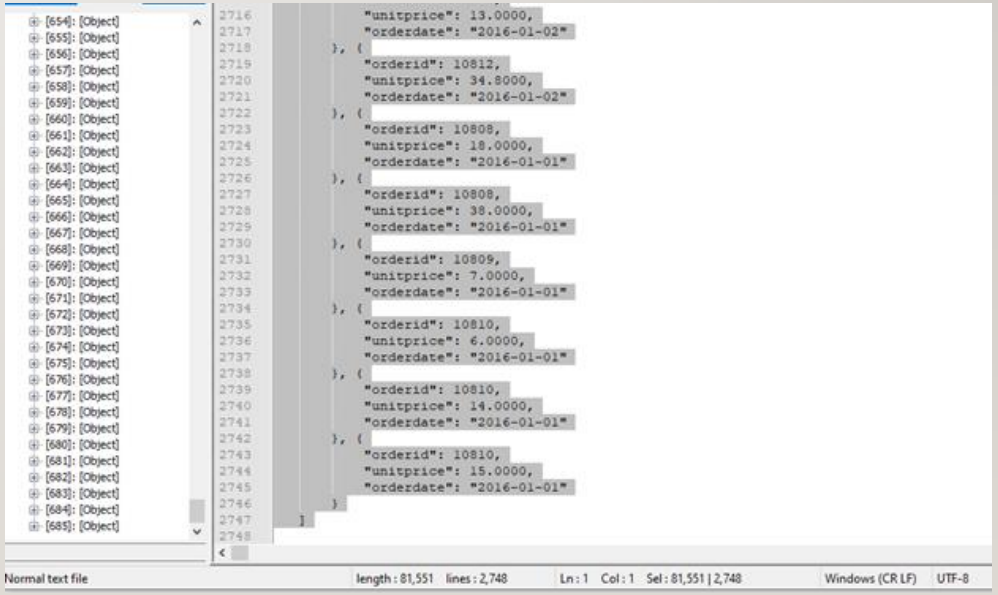
HARJIT'S WORST QUERY: SAMPLE OUTPUT(2/3)

Relational output



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

JSON Output



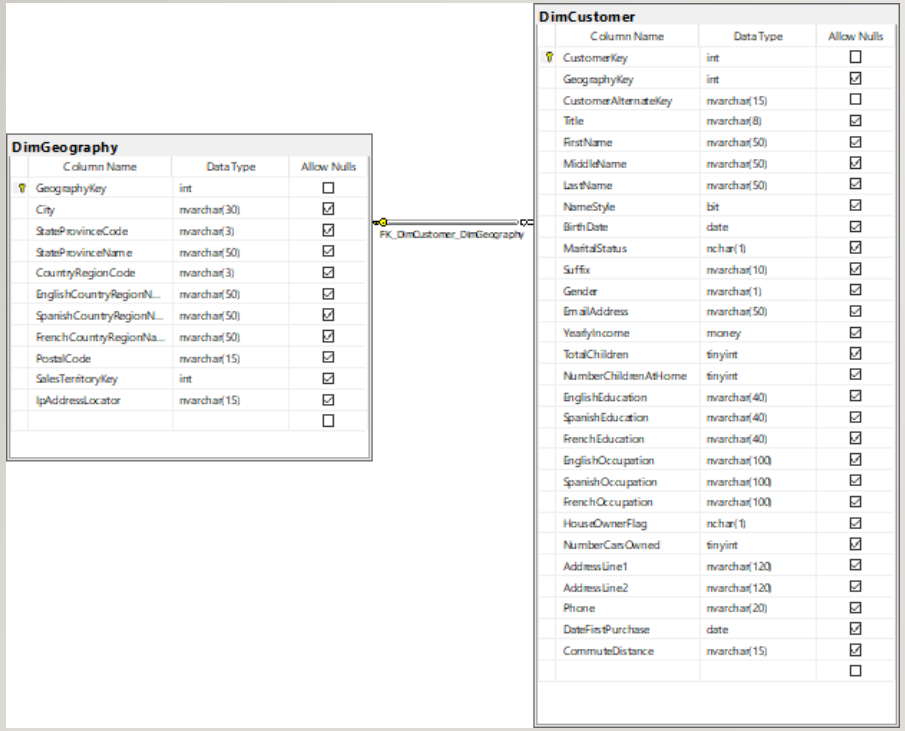
```
        "unitprice": 13.0000,
        "orderdate": "2016-01-02"
    }, {
        "orderid": 10812,
        "unitprice": 34.8000,
        "orderdate": "2016-01-02"
    }, {
        "orderid": 10808,
        "unitprice": 18.0000,
        "orderdate": "2016-01-01"
    }, {
        "orderid": 10808,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

HARJIT'S WORST QUERY: PREPOSITION (3/3)

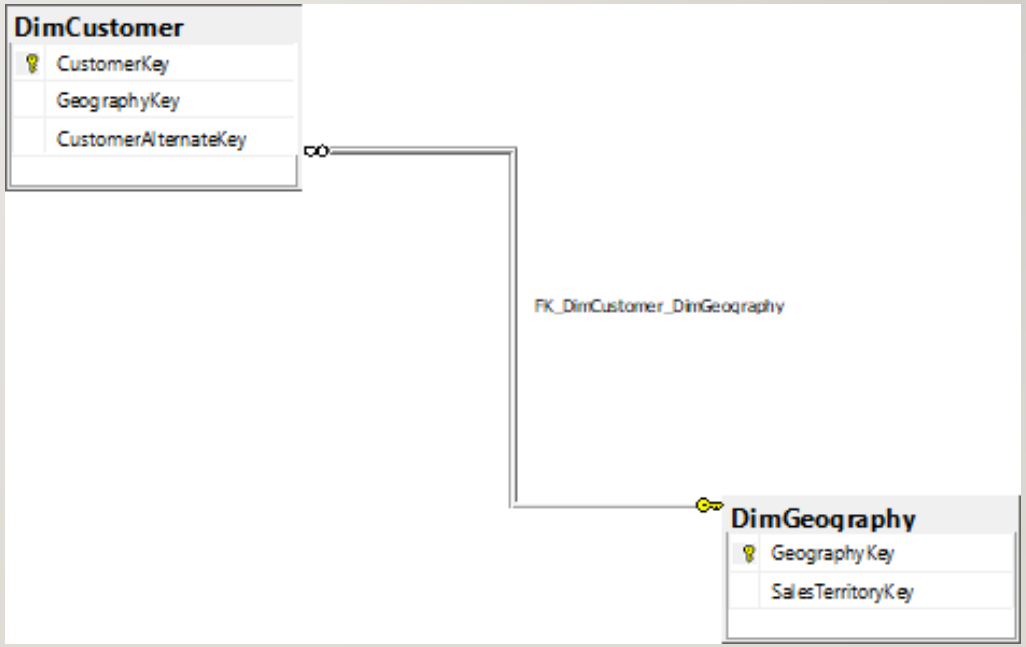
- Returns all Customer keys and Employee keys with their first and last name with the city they are from
- Use AdventureWorksDW2017 Database with dbo.DimCustomer and dbo.DimGeography tables

HARJIT'S WORST QUERY: KEYS (3/3)

Standard view



Key view



HARJIT'S WORST QUERY: TABLES (3/3)

Table Names with column and order by

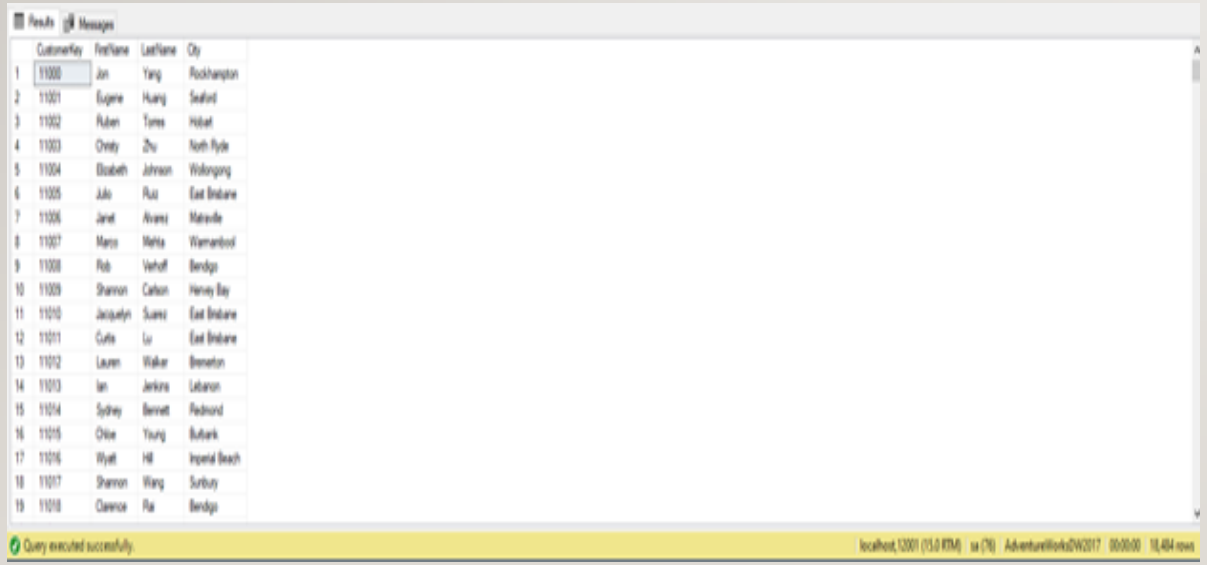
Table Name	Column Names	Order By
Dbo.Geography	CustomerKey	ASC
	FirstName	ASC
	LastName	ASC
Dbo.DimCustomer	City	ASC

HARJIT'S WORST QUERY: CODE (3/3)

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

HARJIT'S WORST QUERY: SAMPLE OUTPUT (3/3)

Relational output



JSON Output

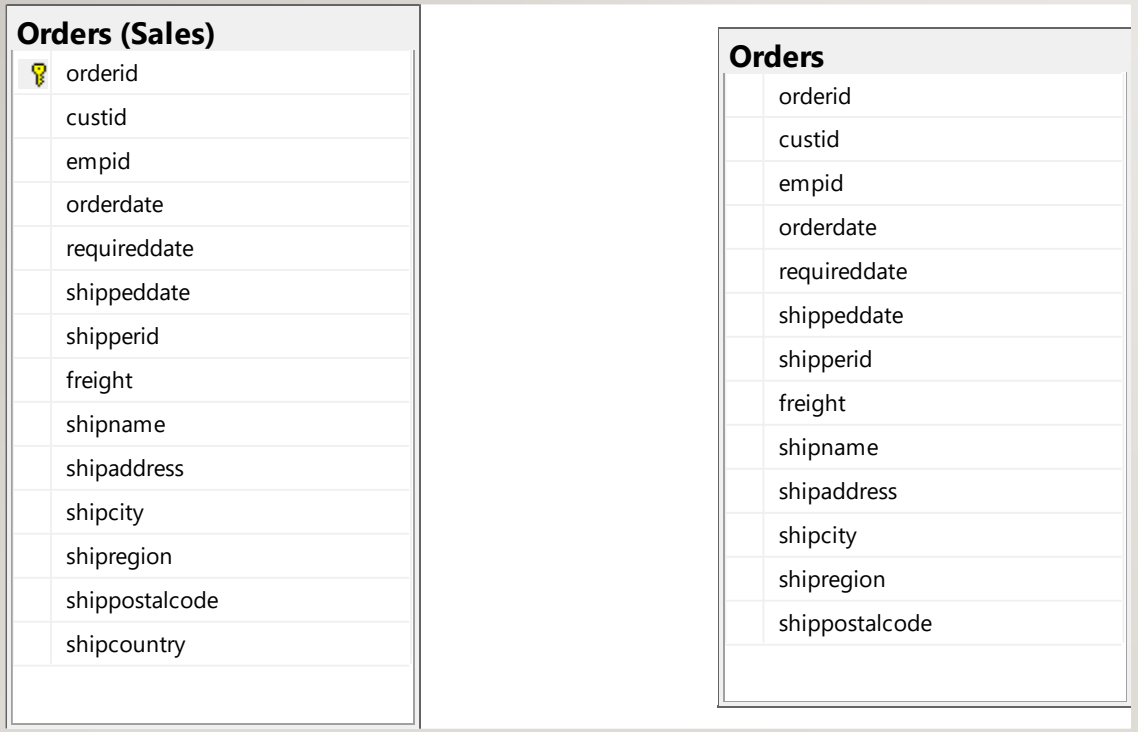
HARJIT'S BEST QUERY: PREPOSITION (1/3)

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

Use TSQLV6 database and Sales.Orders and dbo.Orders

HARJIT'S BEST QUERY: KEYS (1/3)

Standard view



Key View

HARJIT'S BEST QUERY: TABLES (1/3)

Columns from tables with order by

Table Name	Column Names	Order By
Dbo.Order	Orderid custid freight	ASC ASC ASC
Sales.Orders	shippeddate	ASC

HARJIT'S BEST QUERY: CODE (1/3)

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

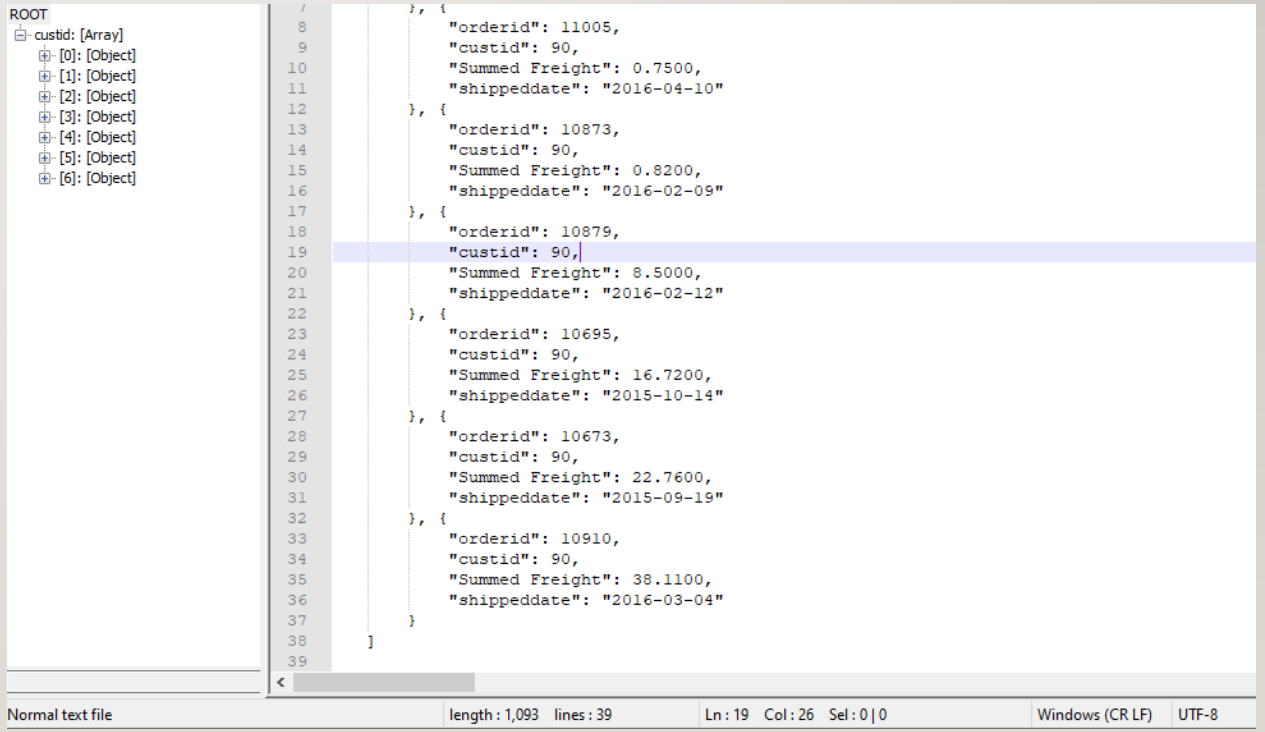
HARJIT'S BEST QUERY: SAMPLE OUTPUT (1/3)

Relational output

Results | Messages

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

JSON Output



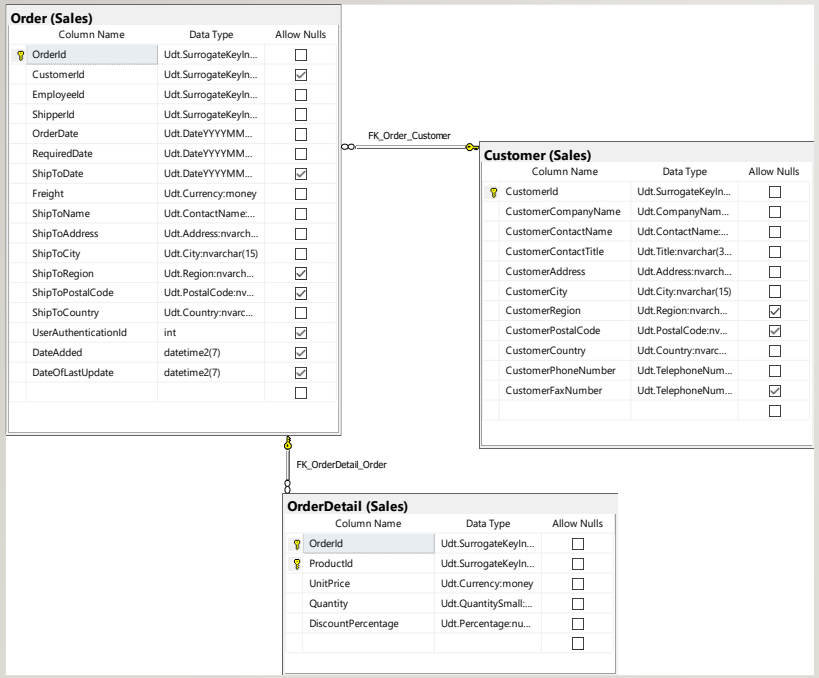
HARJIT'S BEST QUERY: PREPOSITIONS (2/3)

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

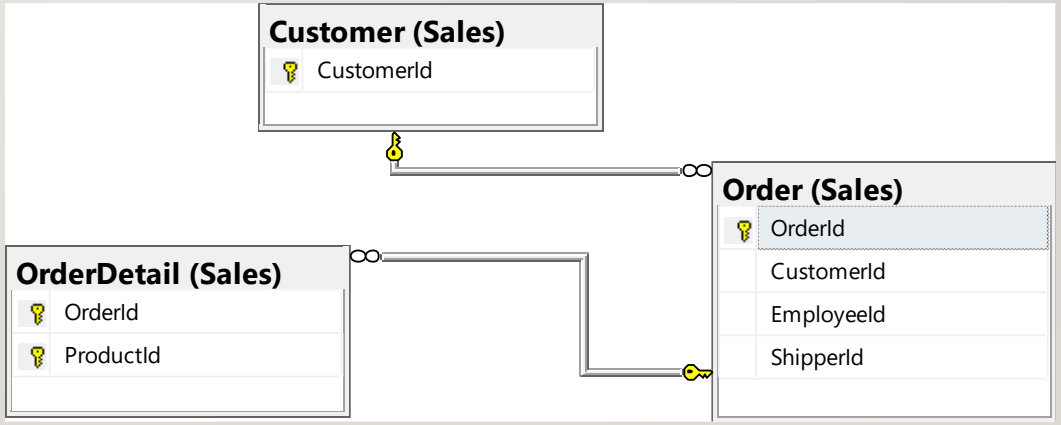
Use Northwinds2020TSQLV6 database with Sales.Customer, Sales.Order, Sales.OrderDetail tables

HARJIT'S BEST QUERY: KEYS (2/3)

Standard view



Key view



HARJIT'S BEST QUERY:TABLES (2/3)

Columns from tables with order by

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

HARJIT'S BEST QUERY: CODE (2/3)

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

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

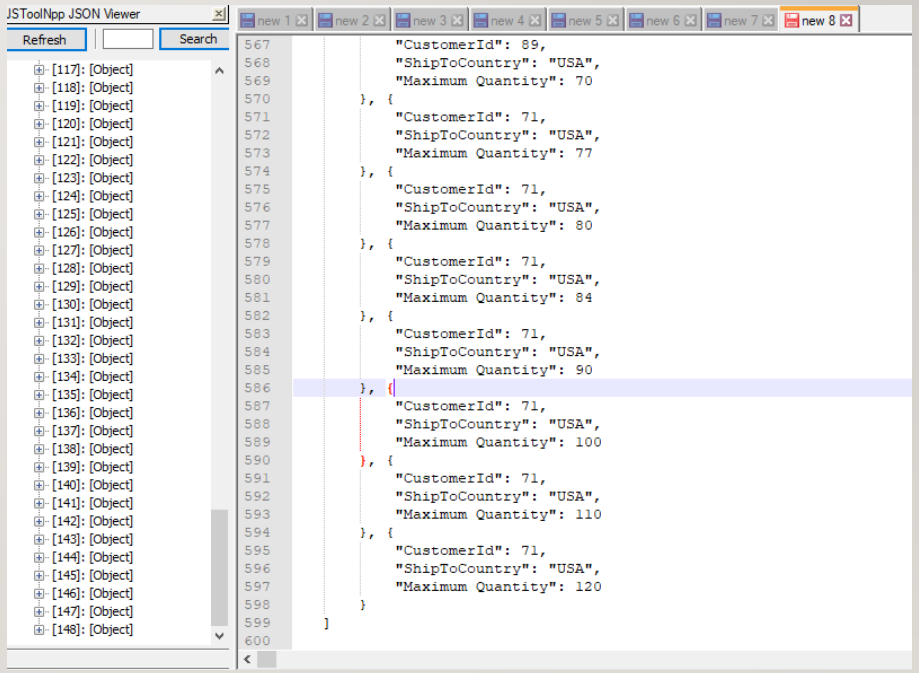
HARJIT'S BEST QUERY: SAMPLE OUTPUT (2/3)

Relational output

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

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

JSON Output



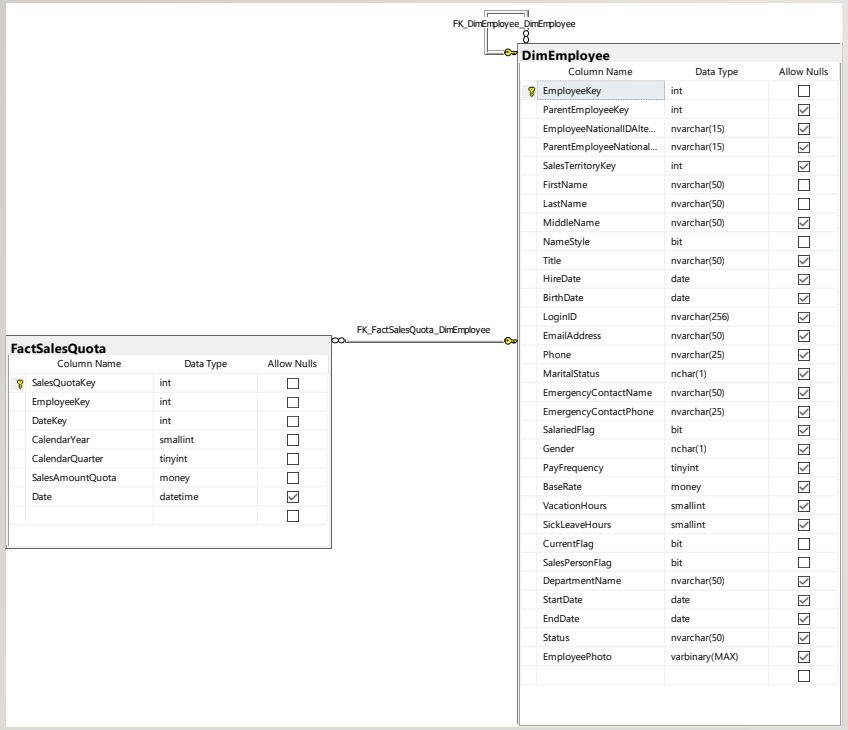
HARJIT'S BEST QUERY: PREPOSITIONS (3/3)

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

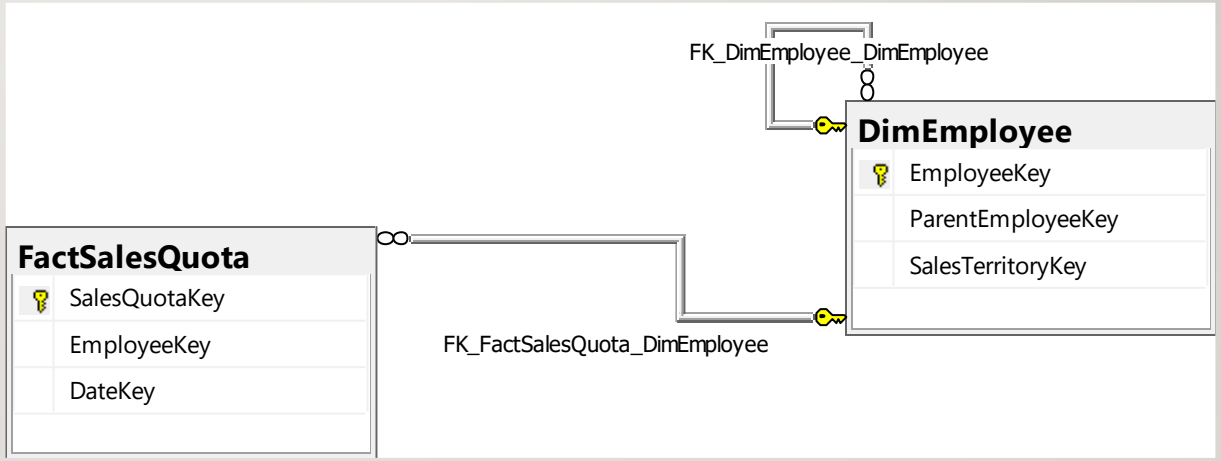
Use AdventureWorksDW2017 database with dbo.DimEmployee and dbo.FactSalesQuota tables

HARJIT'S BEST QUERY: KEYS (3/3)

Standard view



Key view



HARJIT'S BEST QUERY:TABLES (3/3)

Columns from tables with order by

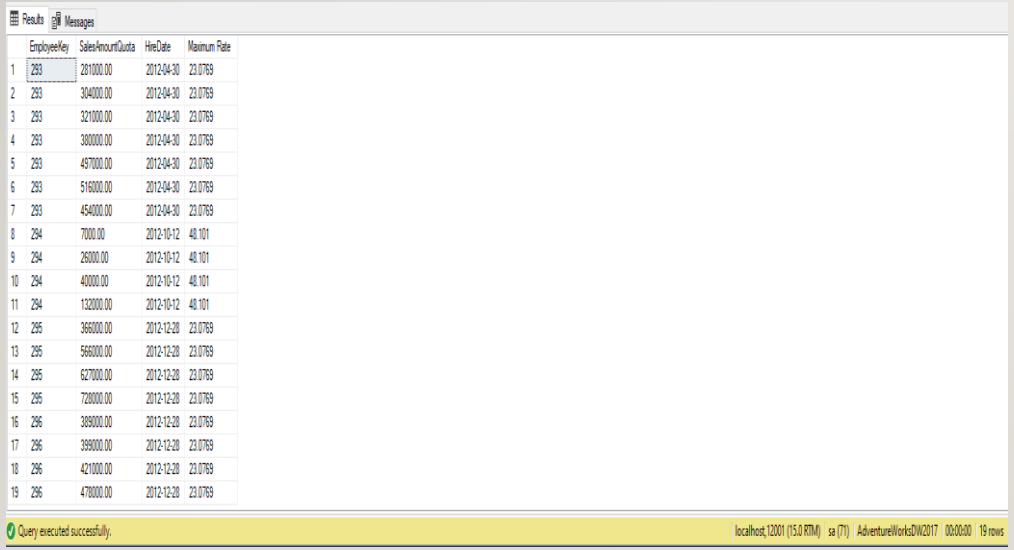
Table Name	Column Names	Order By
Dbo.FactSalesQuota	EmployeeKey SalesAmountQuota	ASC ASC
Dbo.DimEmployee	HireDate BaseRate	ASC ASC

HARJIT'S BEST QUERY: CODE (3/3)

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

HARJIT'S BEST QUERY: SAMPLE OUTPUT (3/3)

Relational Solution

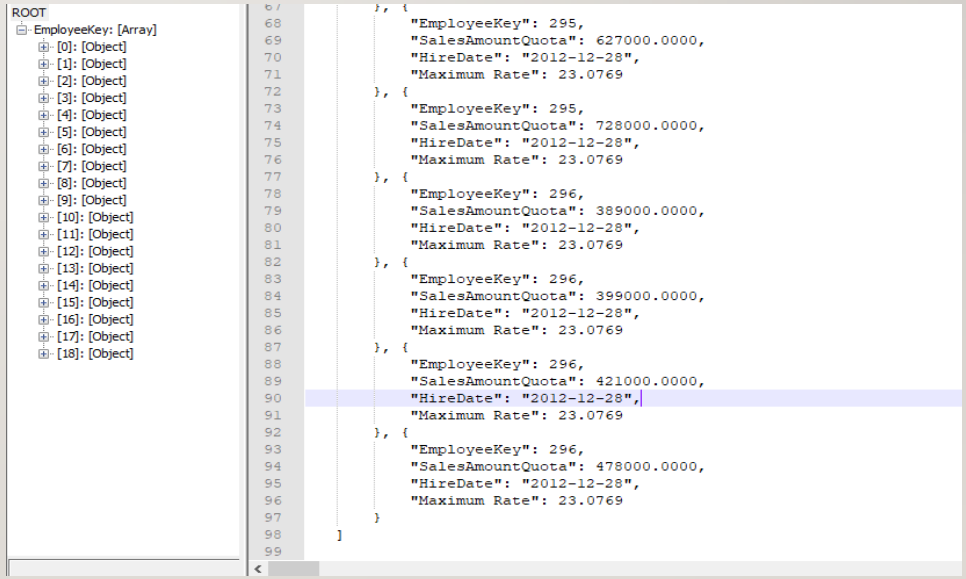


Results | Messages

	EmployeeKey	SalesAmountQuota	HireDate	Maximum Rate
1	293	281000.00	2012-04-30	23.0769
2	293	304000.00	2012-04-30	23.0769
3	293	321000.00	2012-04-30	23.0769
4	293	380000.00	2012-04-30	23.0769
5	293	457000.00	2012-04-30	23.0769
6	293	516000.00	2012-04-30	23.0769
7	293	454000.00	2012-04-30	23.0769
8	294	7000.00	2012-10-12	48.101
9	294	26000.00	2012-10-12	48.101
10	294	40000.00	2012-10-12	48.101
11	294	132000.00	2012-10-12	48.101
12	295	366000.00	2012-12-28	23.0769
13	295	566000.00	2012-12-28	23.0769
14	295	627000.00	2012-12-28	23.0769
15	295	728000.00	2012-12-28	23.0769
16	296	389000.00	2012-12-28	23.0769
17	296	399000.00	2012-12-28	23.0769
18	296	421000.00	2012-12-28	23.0769
19	296	478000.00	2012-12-28	23.0769

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

JSON solution



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

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

HAIBO'S QUERY (SIMPLE) TOP #1

```
return order between 20160212 and 20160312,without null order
--simple query
use Northwinds2020TSQLV6
SELECT C.CustomerId,
       C.CustomerCompanyName,
       O.orderid,
       O.orderdate
FROM   [Sales].[Customer] AS C
       LEFT OUTER JOIN
       [Sales].[Order] AS O
       ON O.CustomerId = C.CustomerId
       AND O.orderdate between '20160212' and '20160312'
where O.OrderId is not null
Order By O.OrderId
```

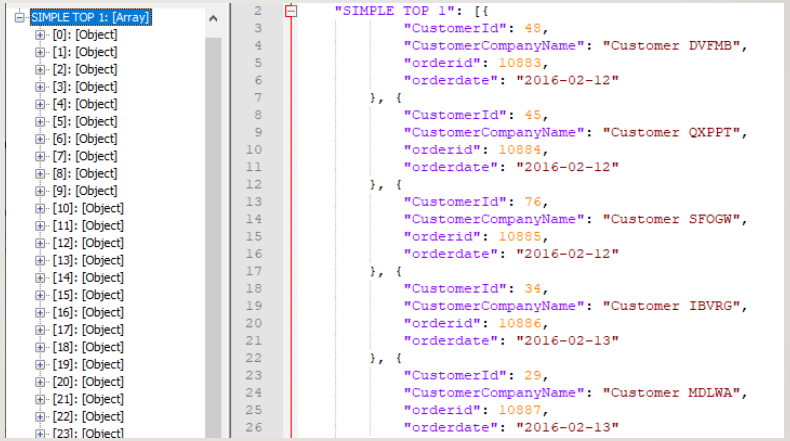
(SIMPLE) TOP #1 TABLE:

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

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

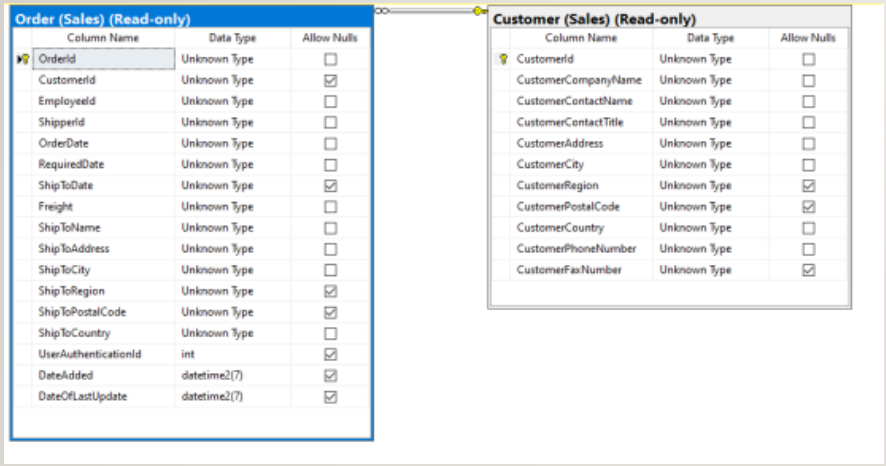
(SIMPLE) TOP #1 OUTPUT WITH/WITHOUT JSON

	CustomerId	CustomerCompanyName	orderid	orderdate
1	48	Customer DVFMB	10883	2016-02-12
2	45	Customer QXPPT	10884	2016-02-12
3	76	Customer SFOGW	10885	2016-02-12
4	34	Customer IBVRG	10886	2016-02-13
5	29	Customer MDLWA	10887	2016-02-13
6	30	Customer KSLQF	10888	2016-02-16
7	65	Customer NYUHS	10889	2016-02-16
8	18	Customer BSVAR	10890	2016-02-16
9	44	Customer OXFRU	10891	2016-02-17
10	50	Customer JYPSC	10892	2016-02-17

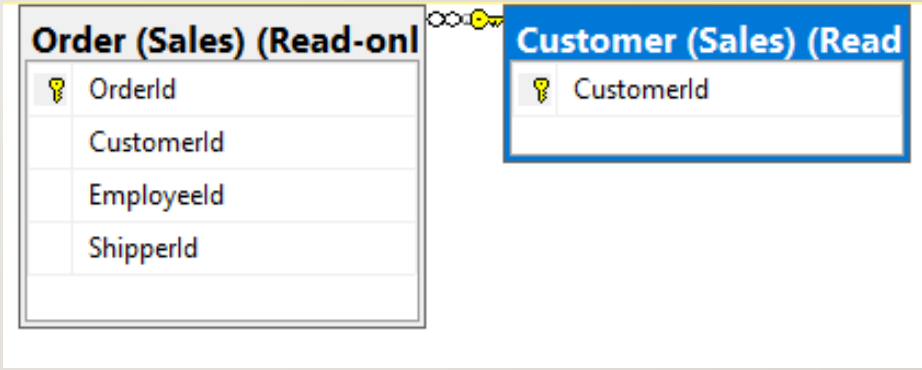


(SIMPLE) TOP #1 DIAGRAM

- Standard View



Key View



HAIBO'S QUERY (MEDIUM) TOP #2

```
return the reviewers and comments for each order, and ratings
Use AdventureWorks2017;

  SELECT COUNT(OD.ProductID) [Count],

  OD.ProductID, PP.[Name],

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

  FROM Sales.SalesOrderDetail  AS OD

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

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

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

HAIBO'S QUERY (MEDIUM) TOP #2 TABLE

- Tables:

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

- Order By:

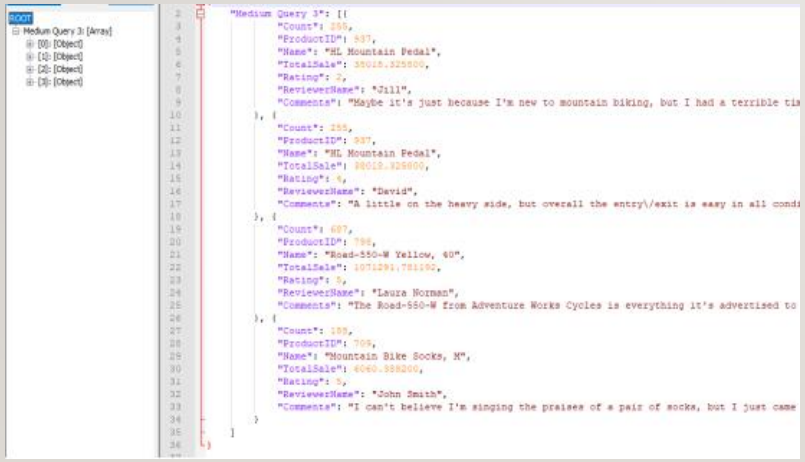
SalesOrderDetail	ProductId	ASC
Product	Name	ASC
ProductReview	Rating,ReviewerName,Comment	ASC

HAIBO'S QUERY (MEDIUM) TOP #2 OUTPUT WITH/WITHOUT JSON

- WITHOUT JSON:

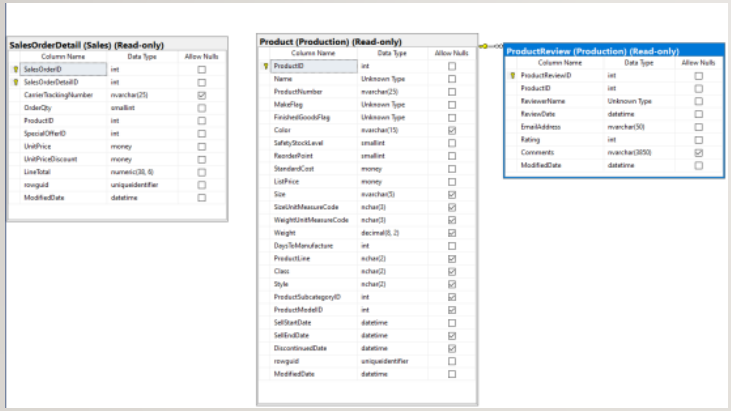
	Count	ProductID	Name	TotalSale	Rating	ReviewerName	Comments
1	255	937	HL Mountain Pedal	38018.325800	2	Jill	Maybe it's just because I'm new to mountain bik...
2	255	937	HL Mountain Pedal	38018.325800	4	David	A little on the heavy side, but overall the ent...
3	687	798	Road-550-W Yellow, 40	1071291.781192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

- WITH JSON:

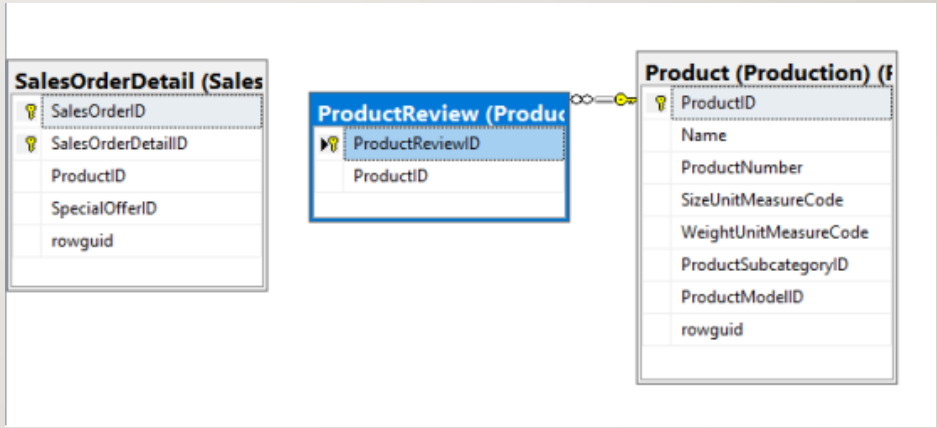


HAIBO'S QUERY (MEDIUM) TOP #2 DIAGRAM

- Standard View:



Key View:



HAIBO'S QUERY (COMPLEX) TOP #3

```
return all the orders and the travel from production to cutomer in 2016
use Northwinds2020TSQLV6;
select S.SupplierId, S.SupplierCompanyName, LOCATIONE = concat(S.SupplierCity,', ', S.SupplierCountry),
        P.ProductId,P.ProductName,
        O.OrderId, O.OrderDate, C.CustomerCompanyName,
        LOCATION = concat(C.CustomerCity,', ',CustomerCountry)
from Production.Supplier as S
        inner join Production.Product as P
            on S.SupplierId = P.SupplierId
        inner join Sales.[Order] as O
            on P.SupplierId = O.ShipperId
        inner join Sales.[Customer] as C
            on C.CustomerId = O.CustomerId
where year(O.orderdate) = 2016
group by S.SupplierId, S.SupplierCompanyName,concat(S.SupplierCity,', ', S.SupplierCountry),
         P.ProductId,P.ProductName,
         O.OrderId, O.OrderDate, C.CustomerCompanyName,concat(C.CustomerCity,', ',CustomerCountry)
order by O.orderdate desc
```

HAIBO'S QUERY (COMPLEX) TOP #3 TABLE

- Table:

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

- Order By:

Table name	Column name	Sort order
Order	Orderdate	ASC

HAIBO'S QUERY (COMPLEX) TOP #3 OUTPUT WITH/WITHOUT JSON

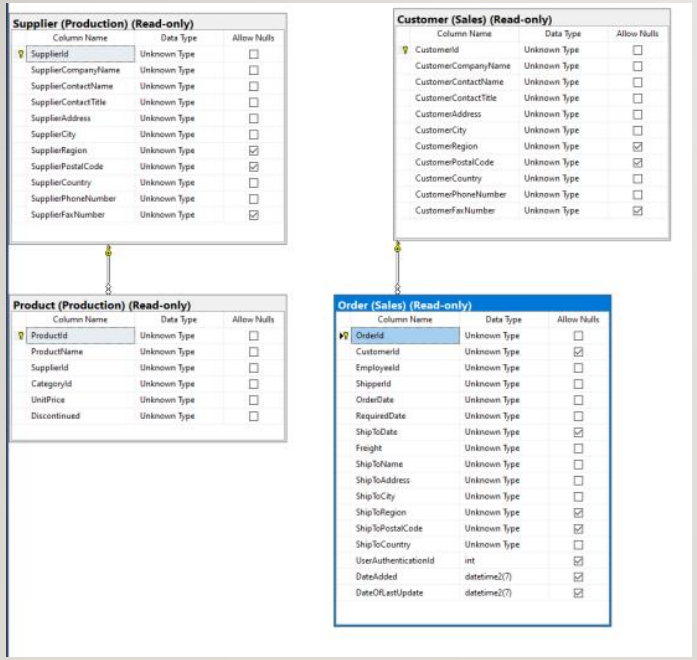
- WITHOUT JSON:

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

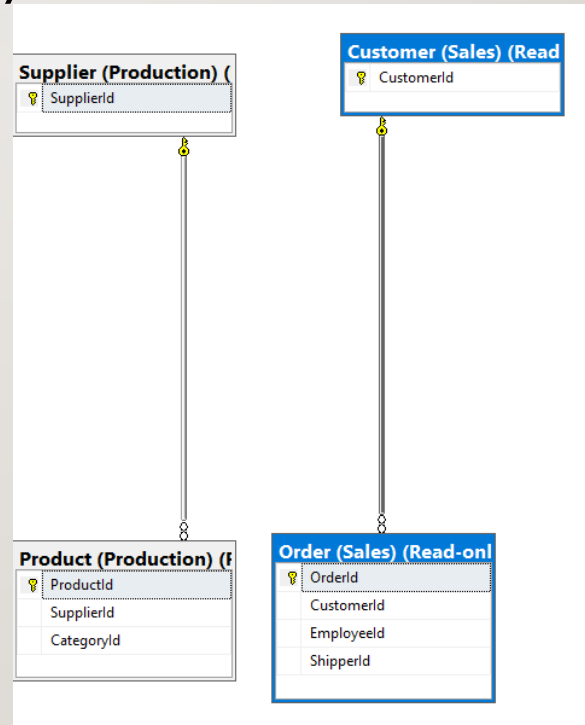
- WITH JSON:

HAIBO'S QUERY (COMPLEX) TOP #3 DIAGRAM

- Standard View:



Key View:



HAIBO'S QUERY (SIMPLE) WORST#1

```
SELECT C.CustomerId, O.OrderId, OD.ProductId, OD.Quantity
FROM Sales.Customer AS C
    LEFT OUTER JOIN Sales.[Order] AS O
        ON C.CustomerId = O.CustomerId
    LEFT OUTER JOIN Sales.OrderDetail AS OD
        ON O.OrderId = OD.OrderId;
```

HAIBO'S QUERY (SIMPLE) WORST#1 TABLE

- Table:

Table name	Column name
Order	OrderId,customerId
Customer	CustomerId
orderDetail	Productid,quantity,OrderId

- Order By:

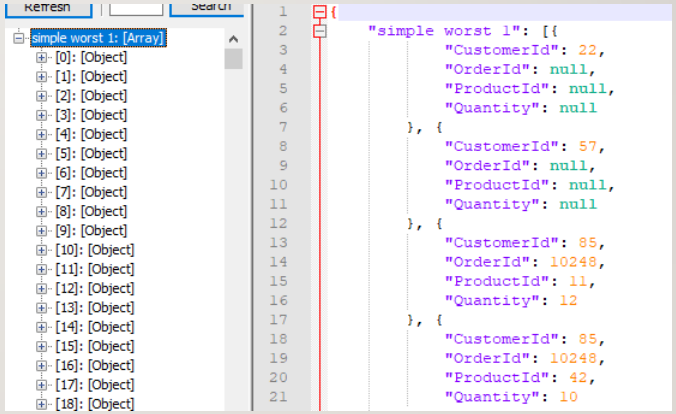
Table Name	Column name	SORT ORDER
Order	OrderId	ASC

HAIBO'S QUERY (SIMPLE) WORST#1 OUTPUT WITH/WITHOUT JSON

- WithOUT JSON:

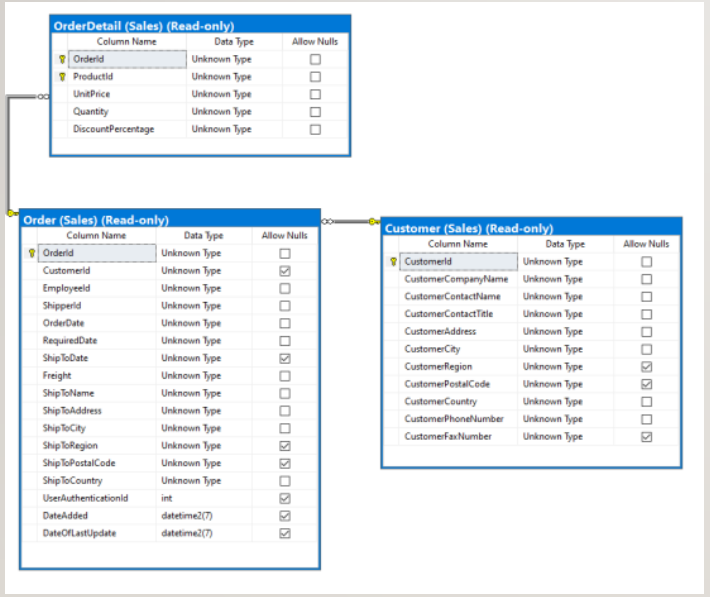
	CustomerId	OrderId	ProductId	Quantity
1	22	NULL	NULL	NULL
2	57	NULL	NULL	NULL
3	85	10248	11	12
4	85	10248	42	10
5	85	10248	72	5
6	79	10249	14	9
7	79	10249	51	40
8	34	10250	41	10
9	34	10250	51	35

WITH JSON:

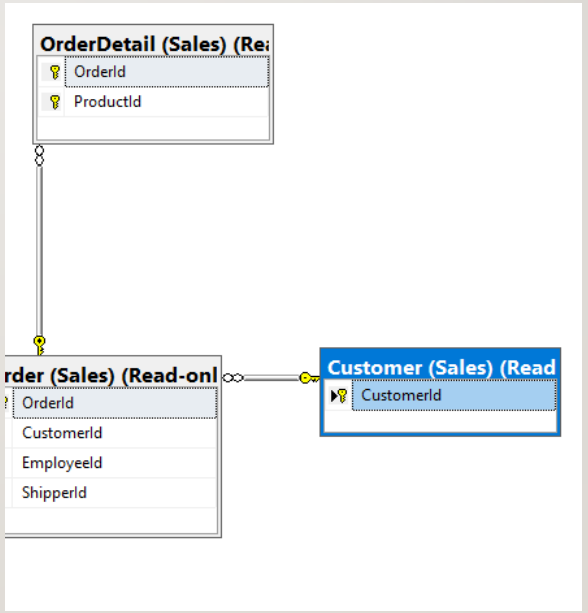


HAIBO'S QUERY (SIMPLE) WORST#1 DIAGRAM

- Standard View:



Key View:



HAIBO'S QUERY (MEDIUM) WORST#2

```
SELECT Customer ,CustomerCompanyName
FROM Sales.Customer AS C
WHERE NOT EXISTS
    (SELECT*
    FROM Sales.[Order] AS O
    WHERE O.CustomerId = C.CustomerId)
Order by CustomerId
```

HAIBO'S QUERY (MEDIUM) WORST#2 TABLE

- Table:

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

- Order By:

Table Name	Column name	SORT ORDER
Customer	CustomerId	ASC

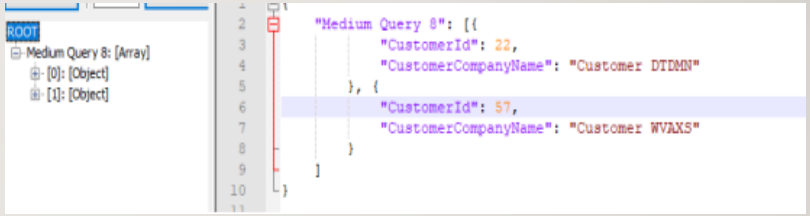
HAIBO'S QUERY (MEDIUM) WORST#2 OUTPUT WITH/WITHOUT JSON

- Without JSON:

Results | Messages

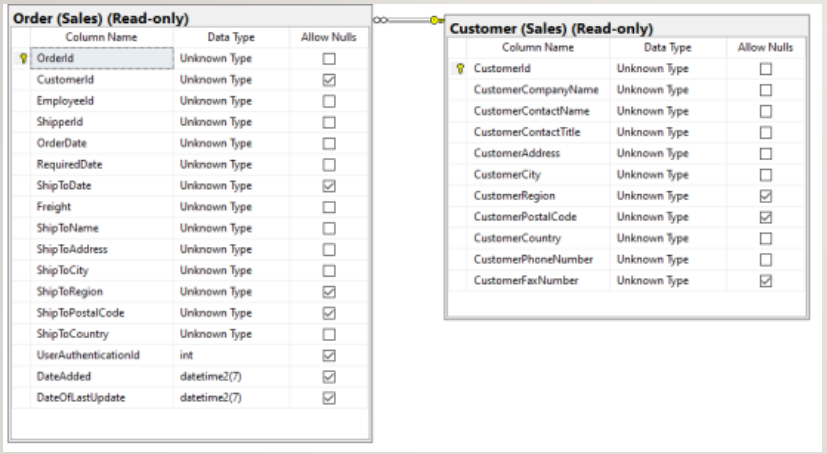
	CustomerId	CustomerCompanyName
1	22	Customer DTDMN
2	57	Customer WVAXS

WITH JSON:



HAIBO'S QUERY (MEDIUM) WORST#2 DIAGRAM

- Standard View:



Key View:

HAIBO'S QUERY (COMPLEX) WORST#3

```
use Northwinds2020TSQLV6;
SELECT C.CustomerId
,O.OrderId
,OD.ProductId
,OD.Quantity
FROM Sales.Customer AS C
LEFT OUTER JOIN Sales.[Order] AS O ON C.CustomerId = O.CustomerId
LEFT OUTER JOIN Sales.OrderDetail AS OD ON O.OrderId = OD.OrderId;
```

HAIBO'S QUERY (COMPLEX) WORST#3 TABLE

- Table:

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

- Order By:

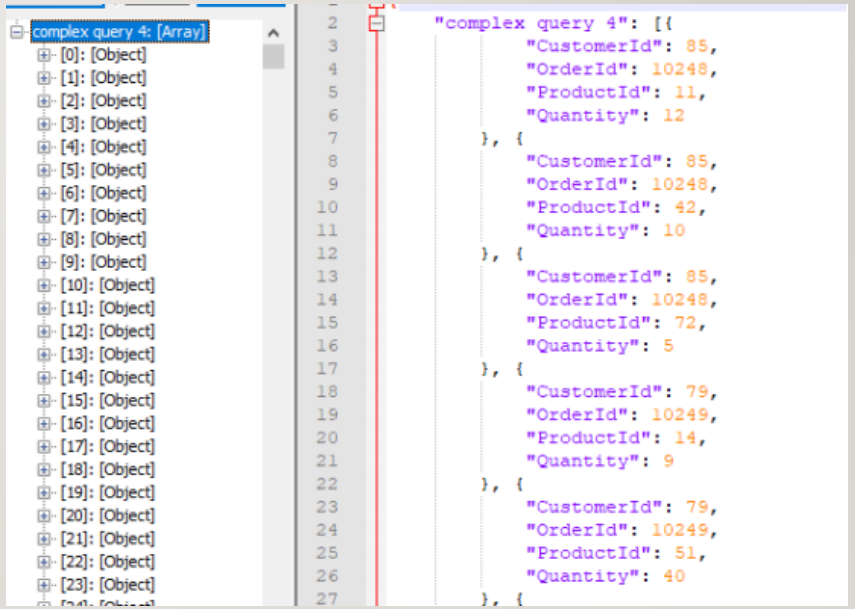
Table Name	Column name	SORT ORDER
Customer	CustomerId	ASC

HAIBO'S QUERY (COMPLEX) WORST#3 OUTPUT WITH/WITHOUT JSON

- WithOUT JSON:

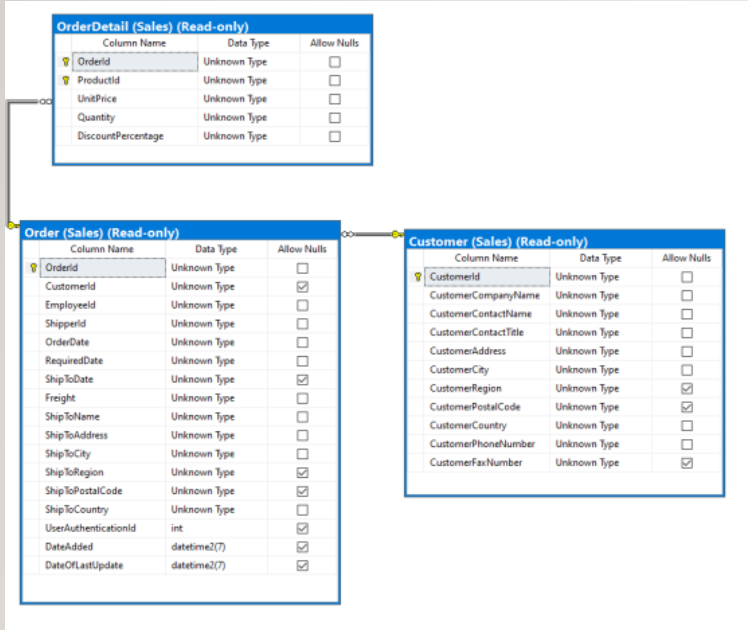
	CustomerId	OrderId	ProductId	Quantit
1	85	10248	11	12
2	85	10248	42	10
3	85	10248	72	5
4	79	10249	14	9
5	79	10249	51	40
6	34	10250	41	10
7	34	10250	51	35
8	34	10250	65	15
9	84	10251	22	6
10	84	10251	57	15
11	84	10251	65	20
12	76	10252	20	40
13	76	10252	33	25
14	76	10252	60	40
15	34	10253	31	20

WITH JSON:

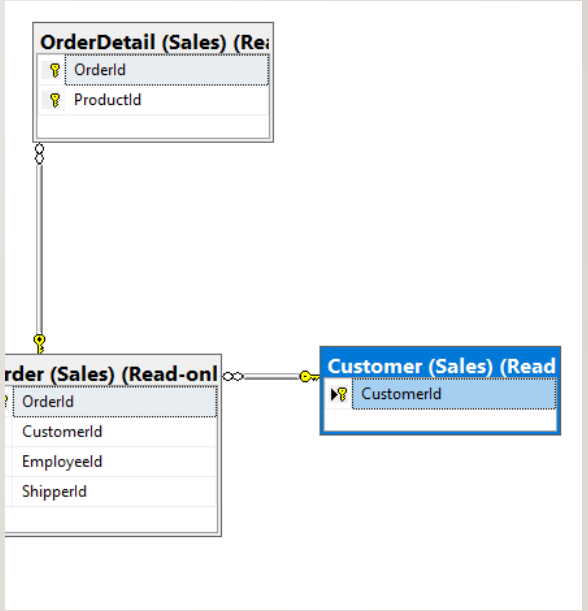


HAIBO'S QUERY (COMPLEX) WORST#3 DIAGRAM

- STANDARD VIEW:



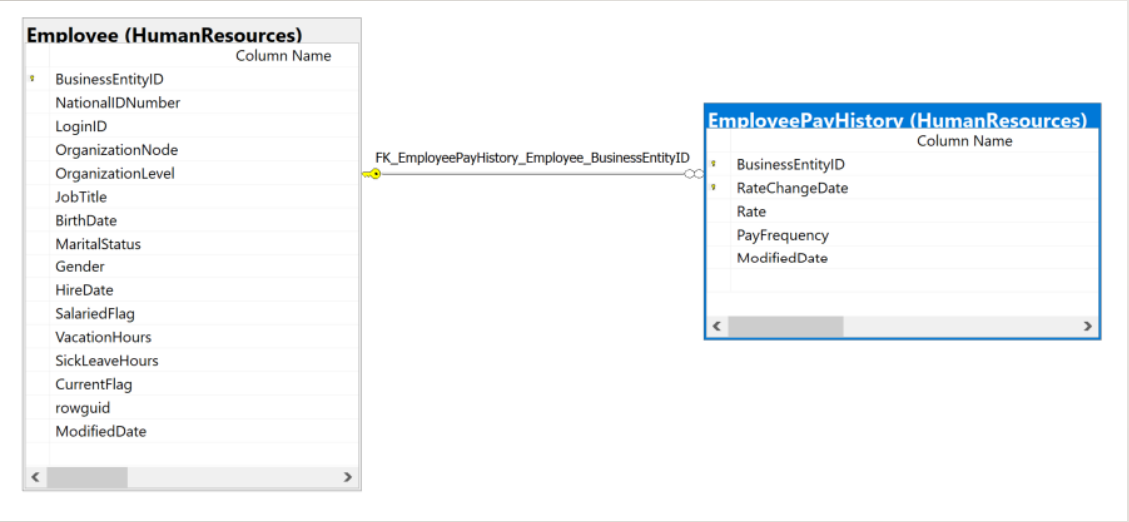
Key View:



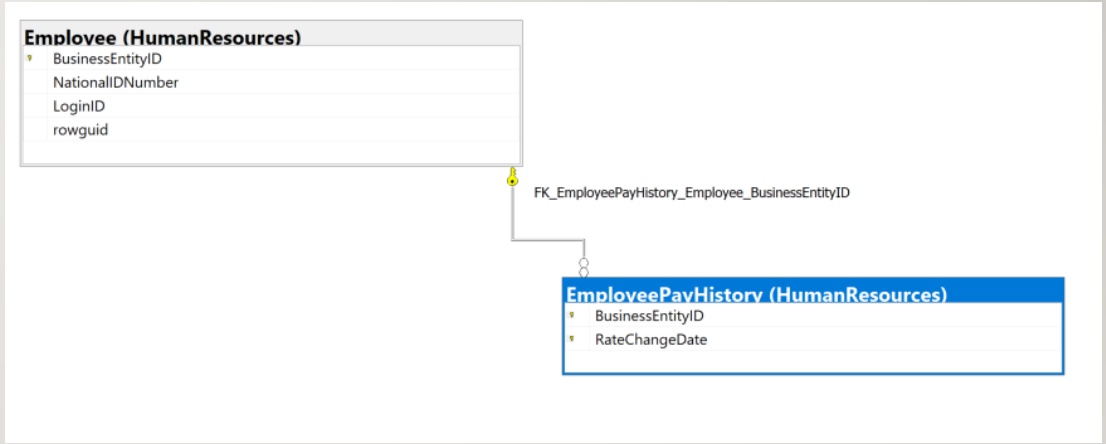
MARLON'S WORST SIMPLE QUERY

```
--(Simple 3) List all employee titles and their current pay
USE AdventureWorks2017;
SELECT E.BusinessEntityID,
       E.JobTitle,
       P.Rate AS HourlyWage
FROM HumanResources.Employee AS E
    INNER JOIN HumanResources.EmployeePayHistory AS P
        ON E.BusinessEntityID = P.BusinessEntityID
ORDER BY HourlyWage DESC
FOR JSON PATH, ROOT('Simple 3 Output'), INCLUDE_NULL_VALUES;
```

MARLON'S WORST SIMPLE QUERY: TABLE DIAGRAMS



Standard View



Key View

MARLON'S WORST SIMPLE QUERY: COLUMN AND ORDER BY TABLES

- **Columns from tables:**

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

Order By

Table Name	Column Name	Sort Order
HumanResources.EmployeePayHistory	HourlyWage	DESC

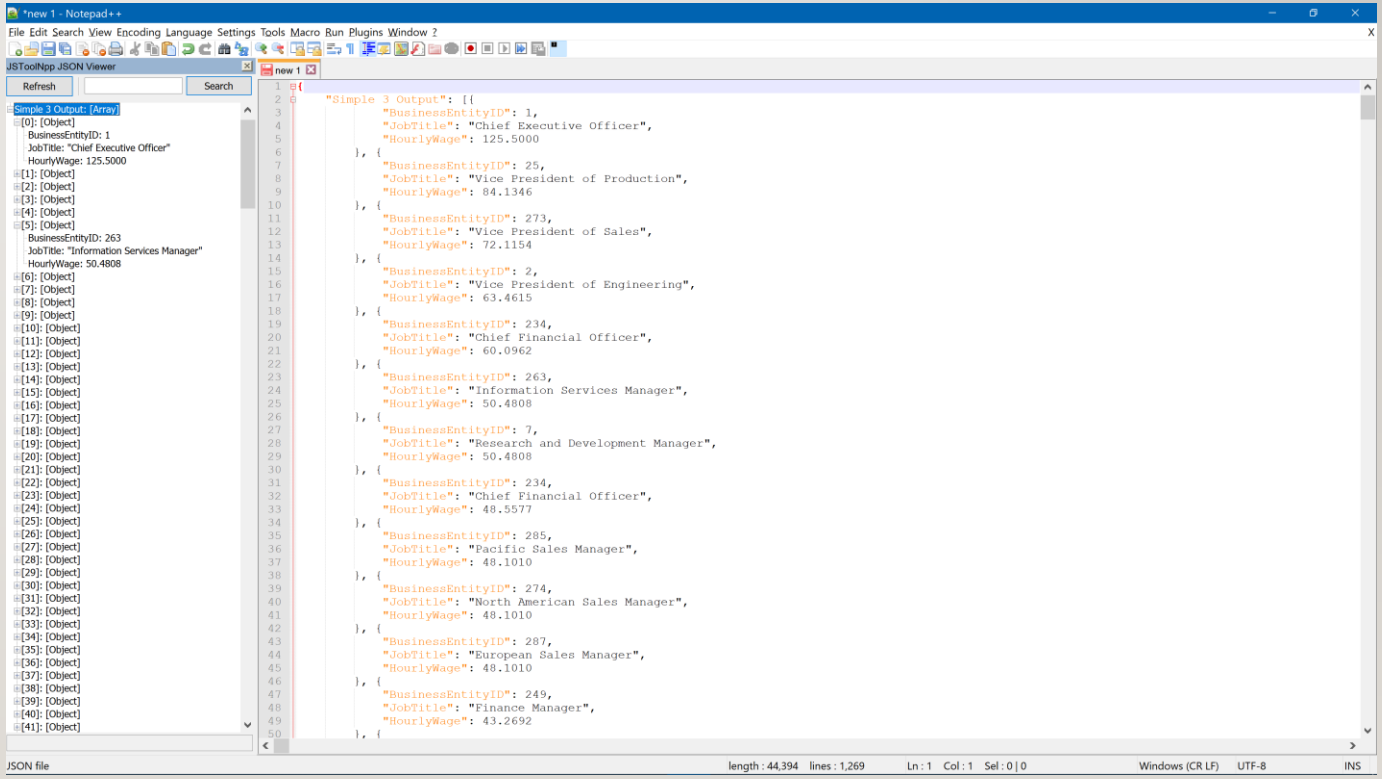
MARLON'S WORST SIMPLE QUERY: OUTPUT

100 %

Results Messages

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

Query executed successfully.

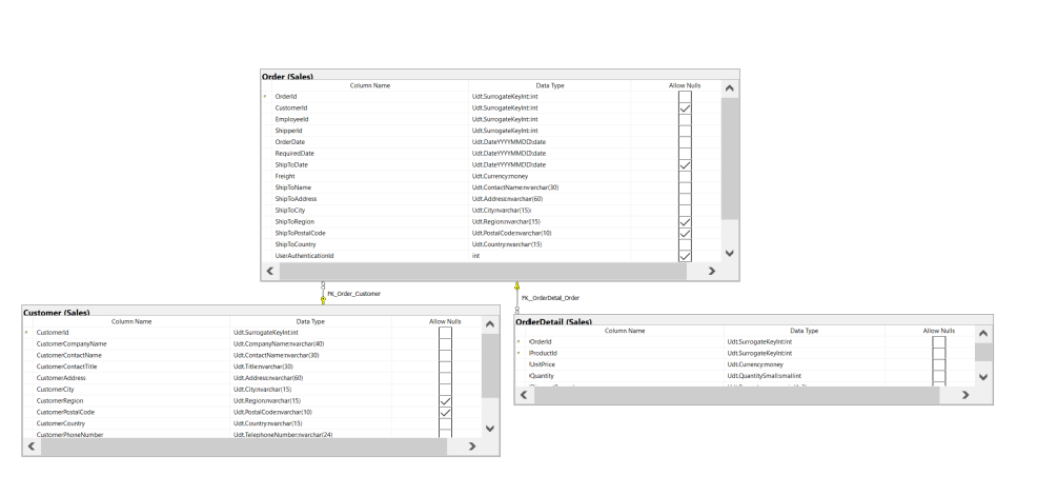


MARLON'S WORST MEDIUM QUERY

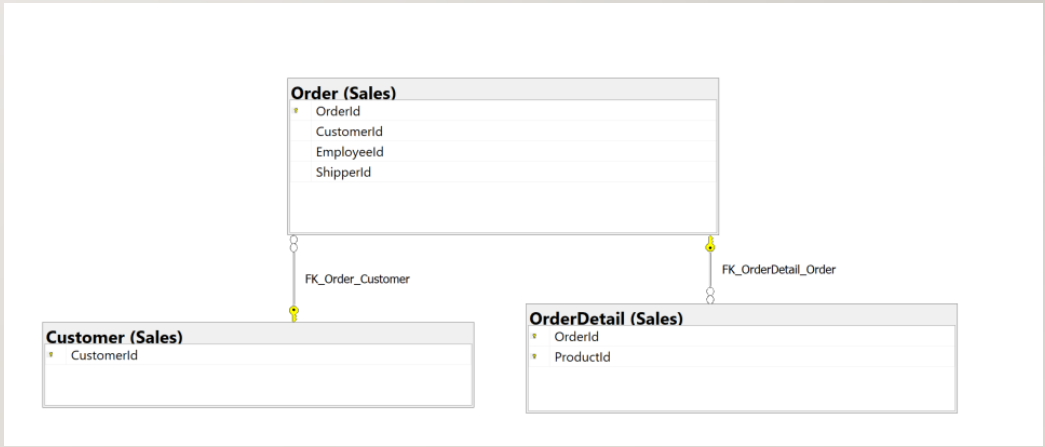
```
--(Medium 2) Find the customers who spent the most on purchasing products.
USE Northwinds2020TSQLV6;
SELECT TOP 5
       o.CustomerId,
       c.CustomerContactName,
       SUM(d.UnitPrice * d.Quantity) AS totalSpent
FROM Sales.OrderDetail AS d
    INNER JOIN Sales.[Order] AS o
        ON d.OrderId = o.OrderId
    INNER JOIN Sales.Customer AS c
        ON o.CustomerId = c.CustomerId
GROUP BY o.CustomerId,
         c.CustomerContactName
ORDER BY totalSpent DESC
FOR JSON PATH, ROOT('Medium 2 Output'), INCLUDE_NULL_VALUES;
```

MARLON'S WORST MEDIUM QUERY: TABLE DIAGRAMS

Standard View



Key View



MARLON'S WORST MEDIUM QUERY: COLUMN AND ORDER BY TABLE

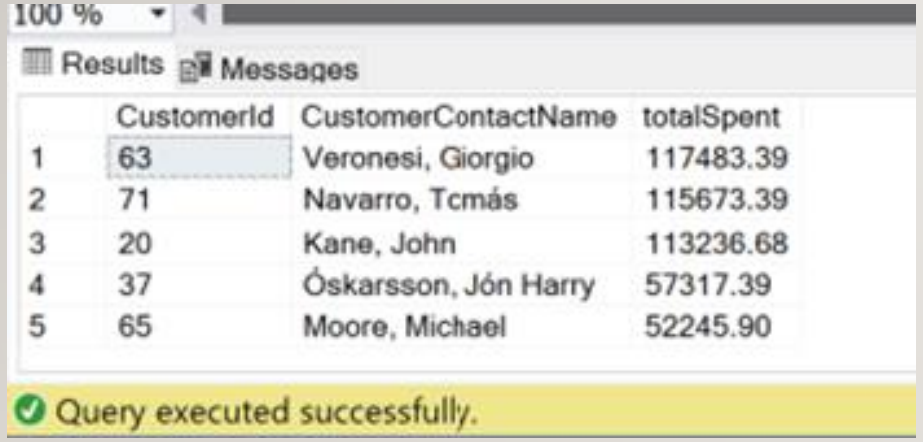
Columns from tables

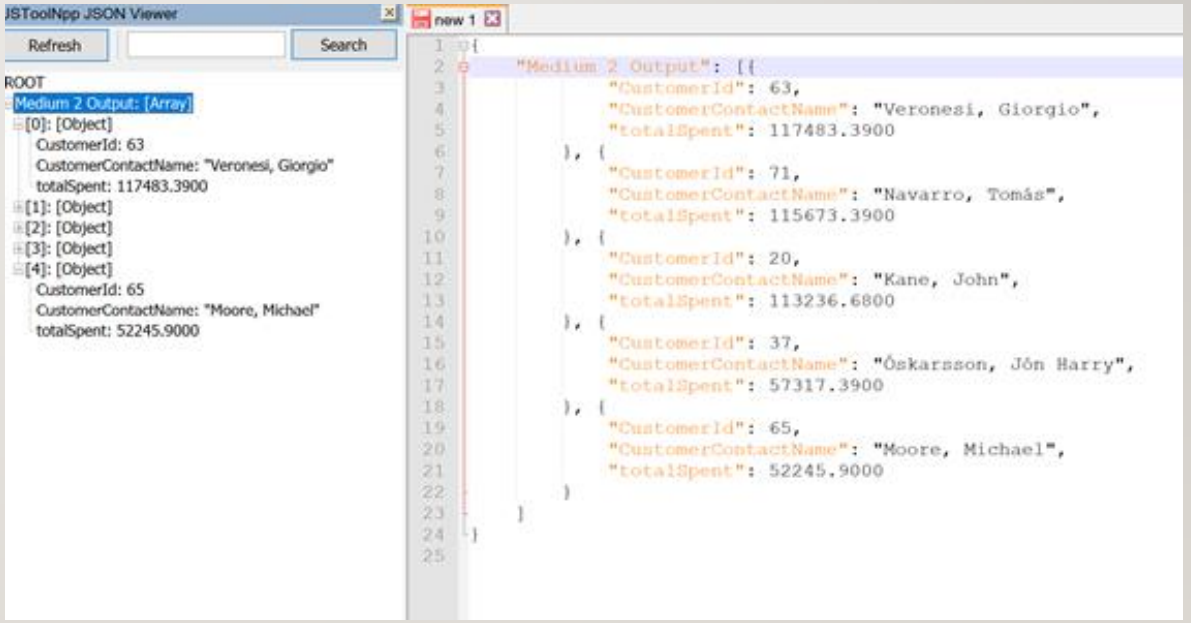
Table Name	Column Name
Sales.Order	CustomerId
Sales.Customer	CustomerContactName

Order By

Table Name	Colum Name	Sort Order
Sales.OrderDetail	TotalSpent	DESC

MARLON'S WORST MEDIUM QUERY: OUTPUT



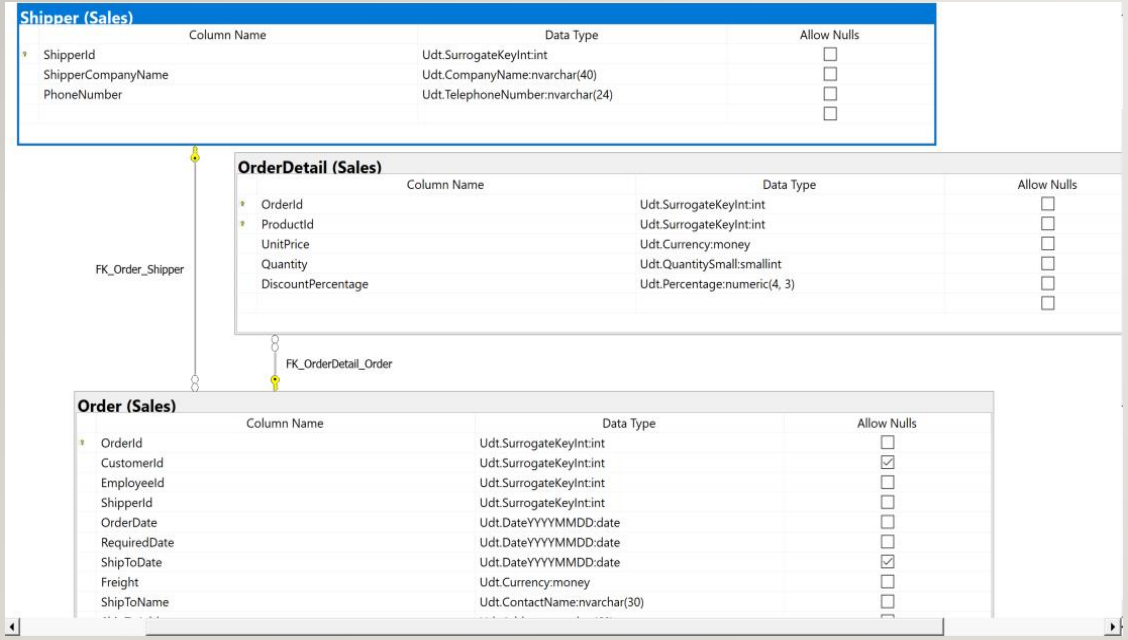


MARLON'S WORST COMPLEX QUERY

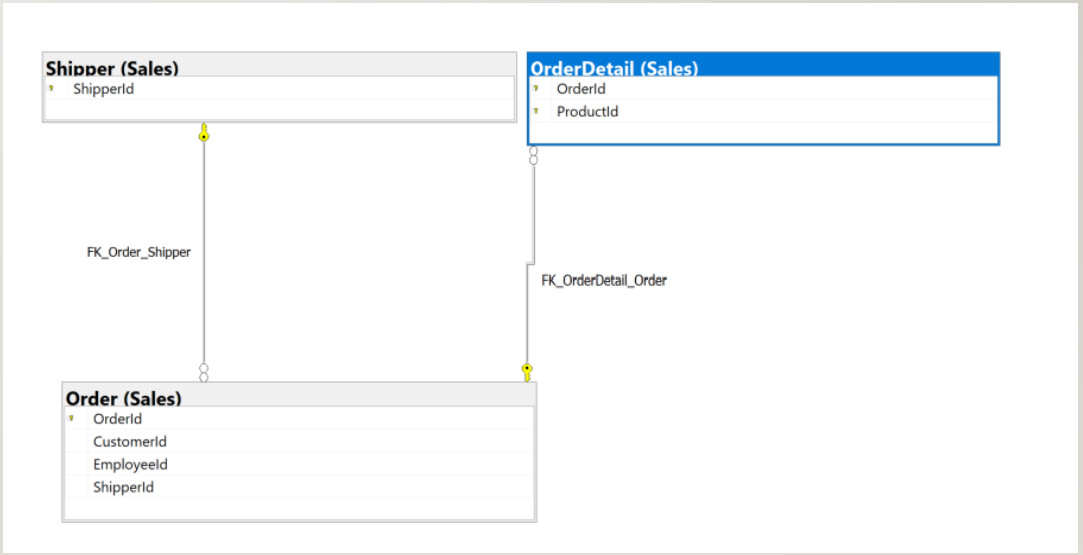
```sql
--Complex 2 -Rank Suppliers based on the number of deliveries they completed and find their most frequent customer
USE Northwinds2020TSQLV6;
SELECT s.ShipperId,
       s.ShipperCompanyName,
       COUNT(o.ShipperId) AS NumDeliveries,
       Sales.[freqCust](s.ShipperId) AS FreqCust
FROM Sales.[Order] AS o
    INNER JOIN Sales.OrderDetail AS od
        ON o.OrderId = od.OrderId
    INNER JOIN Sales.Shipper AS s
        ON s.ShipperId = o.ShipperId
GROUP BY s.ShipperId,
         s.ShipperCompanyName
ORDER BY NumDeliveries DESC
FOR JSON PATH, ROOT('Complex 2 Output'), INCLUDE_NULL_VALUES;
```

MARLON'S WORST COMPLEX QUERY: TABLE DIAGRAMS

Standard View



Key View



MARLON'S WORST COMPLEX QUERY: COLUMN AND ORDER BY TABLES

Columns from tables

Table Name	Column Name
Sales.Shipper	ShipperId ShipperCompanyName NumDeliveries FreqCust

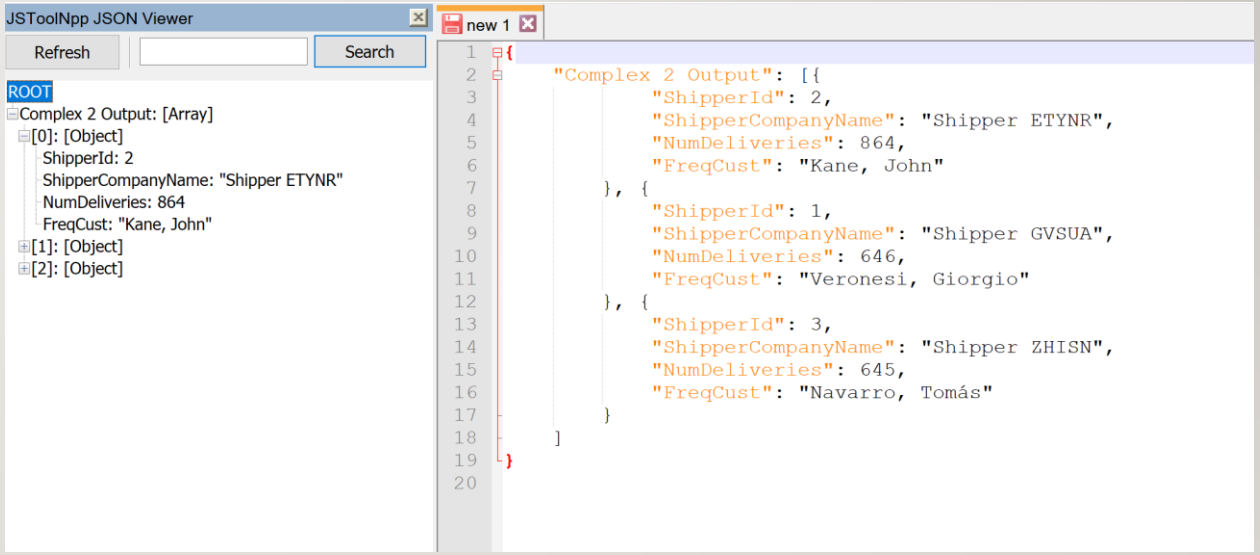
Order By

Table Name	Column Name	Sort Order
Sales.Shipper	NumDeliveries	DESC

MARLON'S WORST COMPLEX QUERY: OUTPUT

Results | Messages

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



MARLON'S TOP SIMPLE QUERY

```
--(Simple 5) List sales people in order from previous year earnings
USE AdventureWorks2017;
SELECT p.BusinessEntityID,
       p.FirstName,
       p.LastName,
       s.SalesYTD,
       s.SalesLastYear
FROM Sales.SalesPerson AS s
    INNER JOIN Person.Person AS p
        ON s.BusinessEntityID = p.BusinessEntityID
ORDER BY SalesLastYear DESC
FOR JSON PATH, ROOT('Simple 5 Output'), INCLUDE_NULL_VALUES;
```

MARLON'S TOP SIMPLE QUERY: TABLE DIAGRAMS

Standard View

SalesPerson (Sales)

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

Person (Person)

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

Key View

SalesPerson (Sales)

- BusinessEntityID
- TerritoryID
- rowguid

Person (Person)

- BusinessEntityID
- rowguid

MARLON'S TOP SIMPLE QUERY: COLUMN AND ORDER BY TABLES

Columns from tables

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

Order By

Table Name	Column Name	Sort Order
Sales.SalesPerson	SalesLastYear	DESC

MARLON'S TOP SIMPLE QUERY: OUTPUT

100 %

Results | Messages

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

Query executed successfully.

Refresh | Search

ROOT
Simple 5 Output: [Array]
[0]: [Object]
BusinessEntityID: 290
FirstName: "Ranjit"
LastName: "Varkey Chudukatil"
SalesYTD: 3121616.3202
SalesLastYear: 2396539.7601
[1]: [Object]
[2]: [Object]
[3]: [Object]
[4]: [Object]
[5]: [Object]
BusinessEntityID: 280
FirstName: "Pamela"
LastName: "Ansman-Wolfe"
SalesYTD: 1352577.1325
SalesLastYear: 1927059.1780
[6]: [Object]
[7]: [Object]
[8]: [Object]
[9]: [Object]
[10]: [Object]
[11]: [Object]
[12]: [Object]
[13]: [Object]
[14]: [Object]
[15]: [Object]
[16]: [Object]


```
{
    "Simple 5 Output": [{
            "BusinessEntityID": 290,
            "FirstName": "Ranjit",
            "LastName": "Varkey Chudukatil",
            "SalesYTD": 3121616.3202,
            "SalesLastYear": 2396539.7601
        }, {
            "BusinessEntityID": 286,
            "FirstName": "Lynn",
            "LastName": "Tsoflias",
            "SalesYTD": 1421810.9242,
            "SalesLastYear": 2278548.9776
        }, {
            "BusinessEntityID": 281,
            "FirstName": "Shu",
            "LastName": "Ito",
            "SalesYTD": 2458535.6169,
            "SalesLastYear": 2073505.9999
        }, {
            "BusinessEntityID": 282,
            "FirstName": "José",
            "LastName": "Saraiva",
            "SalesYTD": 2604540.7172,
            "SalesLastYear": 2038234.6549
        }, {
            "BusinessEntityID": 277,
            "FirstName": "Jillian",
            "LastName": "Carson",
            "SalesYTD": 3189418.3662,
            "SalesLastYear": 1997186.2037
        }, {
            "BusinessEntityID": 280,
            "FirstName": "Pamela",
            "LastName": "Ansman-Wolfe",
            "SalesYTD": 1352577.1325,
            "SalesLastYear": 1927059.1780
        }, {
            "BusinessEntityID": 279,
            "FirstName": "Tsvi",
            "LastName": "Reiter",
            "SalesYTD": 2315185.6110,
            "SalesLastYear": 1849640.9418
        }, {
            "BusinessEntityID": 275,
            "FirstName": "Michael",
            "LastName": "Blythe",
            "SalesYTD": 3763178.1787,
            "SalesLastYear": 1750406.4785
        }, {
```


JSON file

MARLON'S TOP MEDIUM QUERY

```
--(Medium 5) Find the resellers who sold the most value. List some information abot their operations
USE AdventureWorksDW2017;
SELECT sales.ResellerKey,
       rs.ResellerName,
       rs.AnnualSales,
       rs.NumberEmployees,
       terr.SalesTerritoryCountry,
       terr.SalesTerritoryRegion
FROM dbo.DimReseller AS rs
    INNER JOIN dbo.FactResellerSales AS sales
        ON rs.ResellerKey = sales.ResellerKey
    INNER JOIN dbo.DimSalesTerritory AS terr
        ON sales.SalesTerritoryKey = terr.SalesTerritoryKey
WHERE rs.AnnualSales =
(
    SELECT MAX(AnnualSales) FROM dbo.DimReseller
)
GROUP BY sales.ResellerKey,
         rs.ResellerName,
         rs.AnnualSales,
         rs.NumberEmployees,
         terr.SalesTerritoryCountry,
         terr.SalesTerritoryRegion
FOR JSON PATH, ROOT('Medium 5 Output'), INCLUDE_NULL_VALUES;
```

MARLON'S TOP MEDIUM QUERY: TABLE DIAGRAMS

Standard View

Key View

MARLON'S TOP MEDIUM QUERY: COLUMN AND ORDER BY TABLE

Columns from tables

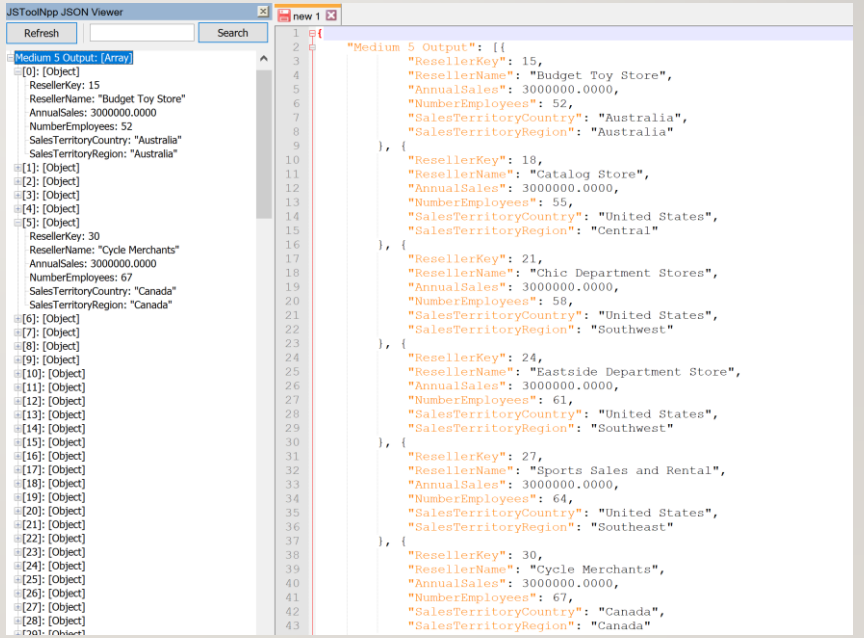
Table Name	Column Name
Dbo.DimReseller	ResellerName AnnualSales NumberEmployees
Dbo.DimSalesTerritory	SalesTerritoryCountry SalesTerritoryRegion

MARLON'S TOP MEDIUM QUERY: OUTPUT

Results Messages

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

Query executed successfully.

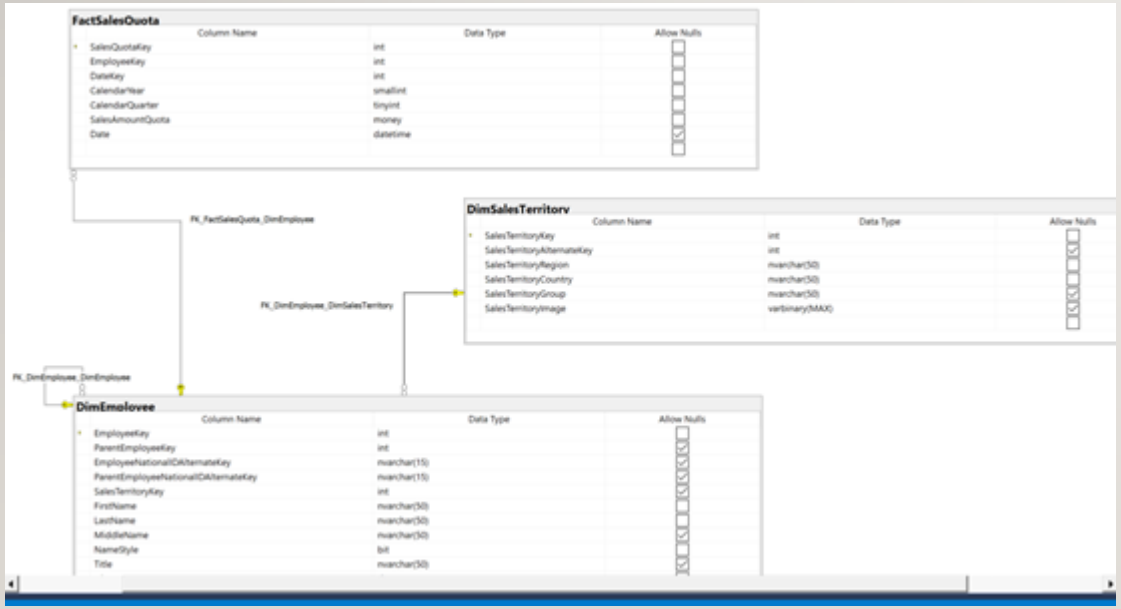


MARLON'S TOP COMPLEX QUERY

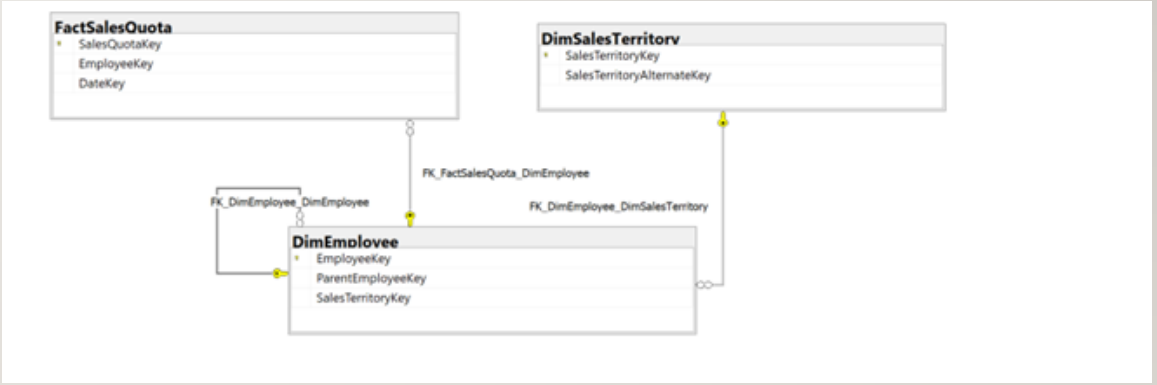
```sql
--Complex 6 -Find where employee sales were in relation to their quarter sales quota
USE AdventureWorksDW2017;
SELECT emp.EmployeeKey,
       emp.FirstName,
       emp.LastName,
       emp.Title,
       terr.SalesTerritoryCountry,
       quota.CalendarYear,
       quota.CalendarQuarter,
       AVG(quota.SalesAmountQuota) AS QtrQuota,
       dbo.actQtrSales(emp.EmployeeKey, quota.CalendarYear, quota.CalendarQuarter) AS QuarterSales
FROM dbo.FactSalesQuota AS quota
    INNER JOIN dbo.DimEmployee AS emp
        ON emp.EmployeeKey = quota.EmployeeKey
    INNER JOIN dbo.DimSalesTerritory AS terr
        ON terr.SalesTerritoryKey = emp.SalesTerritoryKey
WHERE quota.CalendarYear != 2010
GROUP BY quota.CalendarYear,
         quota.CalendarQuarter,
         emp.EmployeeKey,
         emp.FirstName,
         emp.LastName,
         emp.Title,
         terr.SalesTerritoryCountry
ORDER BY emp.EmployeeKey,
         quota.CalendarYear,
         quota.CalendarQuarter
FOR JSON PATH, ROOT('Complex 6 Output'), INCLUDE_NULL_VALUES;
```

MARLON'S TOP COMPLEX QUERY: TABLE DIAGRAMS

Standard View



Key View



MARLON'S TOP COMPLEX QUERY: COLUMN AND ORDER BY TABLES

Columns from tables

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

Order By

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

MARLON'S TOP COMPLEX QUERY: OUTPUT

Results Messages

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

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