

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
Deadline: 10/25/2020

Name: Erik Kim
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	Days	Number Of	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 Deadline: 10/25/2020				Name: Danny Kong 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		Thirteen		
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

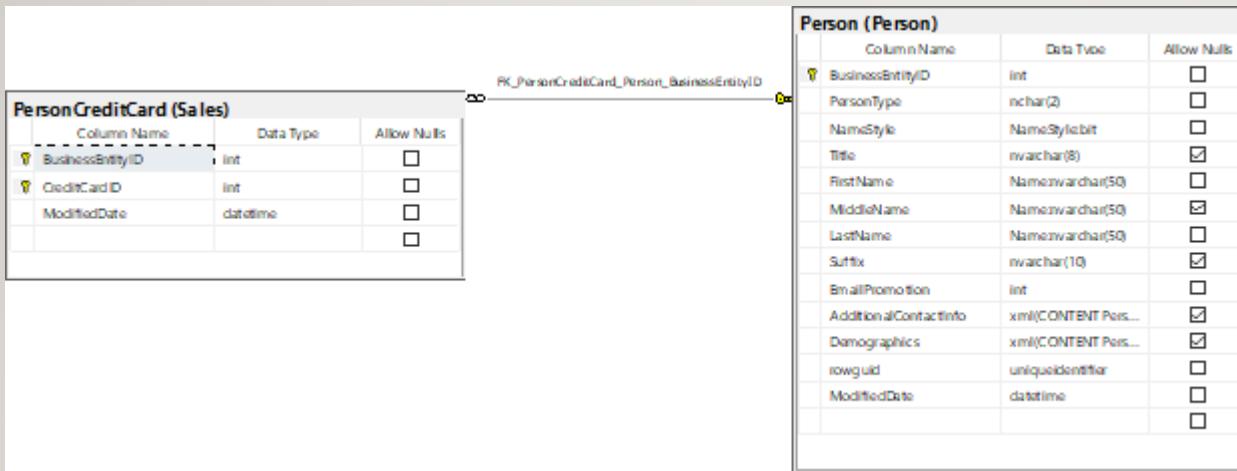
Select a period to highlight at right. A legend describing the charting follows.

Period Highlight: 1 Plan Duration Actual Start % Complete Actual (beyond plan) % Complete (beyond plan)

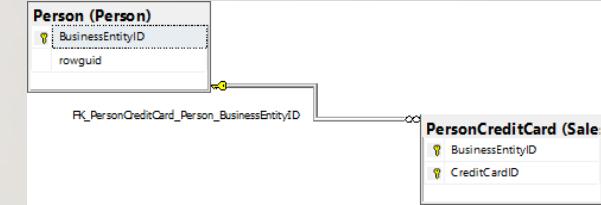


SIMPLE WORST#1 STANDARD/KEY VIEW

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:

Results Messages

	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

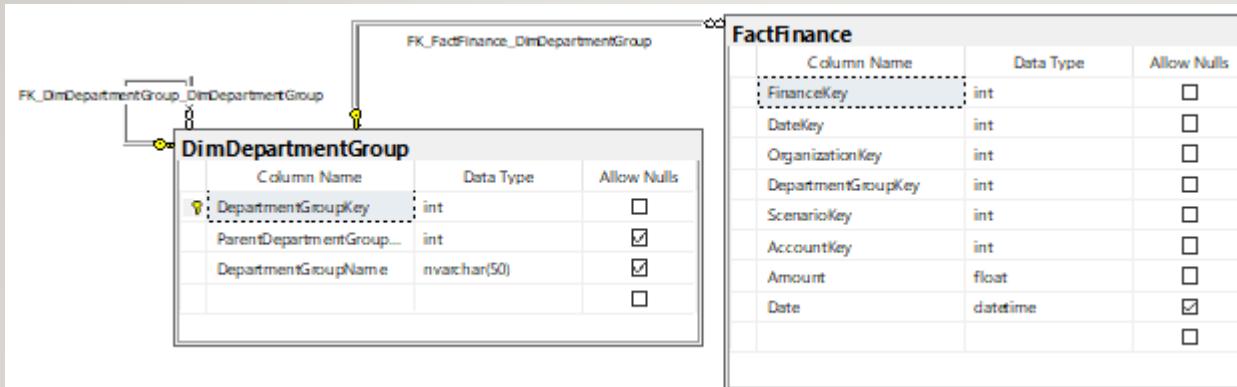
JSON Output:

```
BusinessEntityID<ID>
    CreditCardID<ID>
```

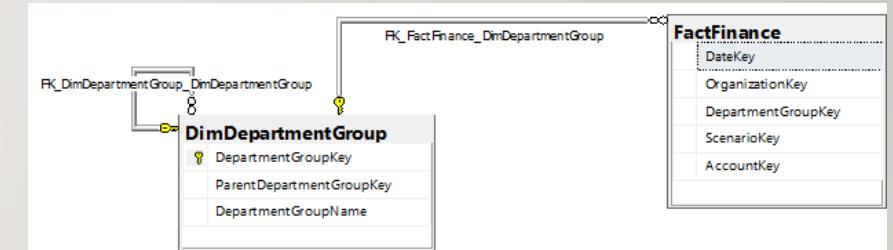


MEDIUM WORST#2 STANDARD/KEY VIEW

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:

DepartmentGroupKey	ParentDepartmentGroupKey
1	0
2	1
3	1
4	1
5	1
6	1
7	1

Query executed successfully.

localhost,12501 (15.0 RTM) : sa (SA) : AdventureWorksDW2017 : 00:00:00 7 rows

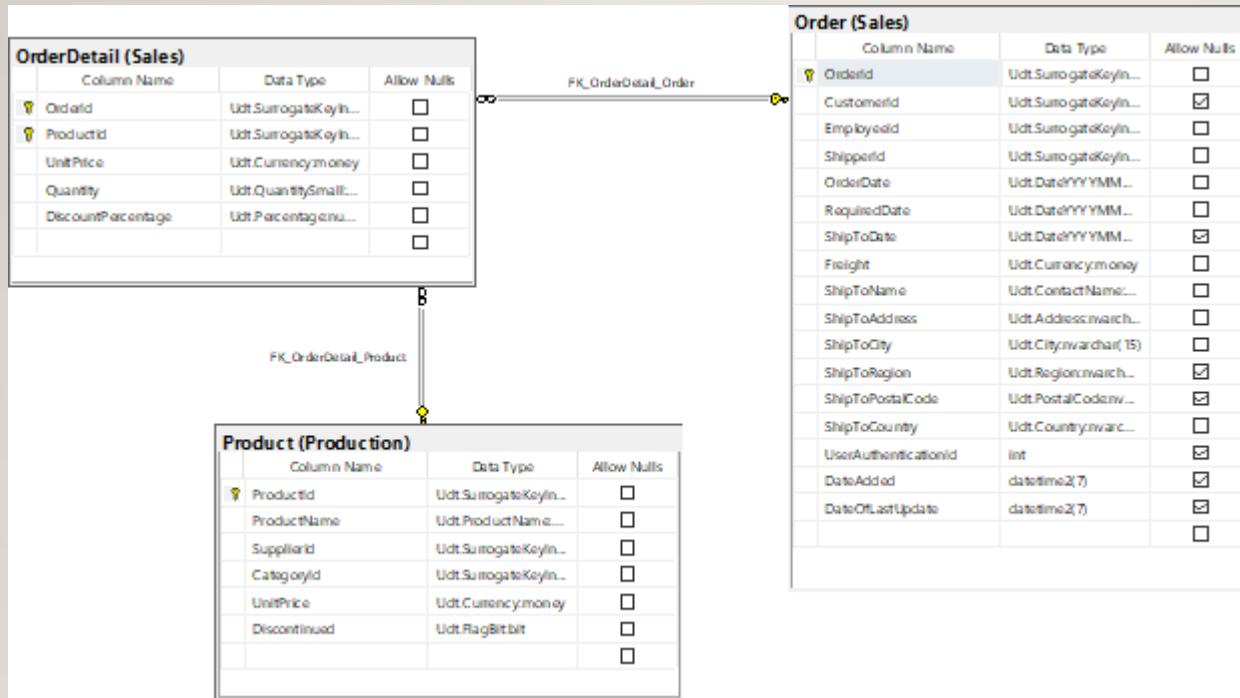
JSON Output:

```
1 "CheckKeyForNull": [i
2   "DepartmentGroupKey": 1,
3   "ParentDepartmentGroupKey": 0
4 ], {
5   "DepartmentGroupKey": 2,
6   "ParentDepartmentGroupKey": 1
7 }, {
8   "DepartmentGroupKey": 3,
9   "ParentDepartmentGroupKey": 1
10 }, {
11   "DepartmentGroupKey": 4,
12   "ParentDepartmentGroupKey": 1
13 }, {
14   "DepartmentGroupKey": 5,
15   "ParentDepartmentGroupKey": 1
16 }, {
17   "DepartmentGroupKey": 6,
18   "ParentDepartmentGroupKey": 1
19 }, {
20   "DepartmentGroupKey": 7,
21   "ParentDepartmentGroupKey": 1
22 }
23 ]
24 }
```

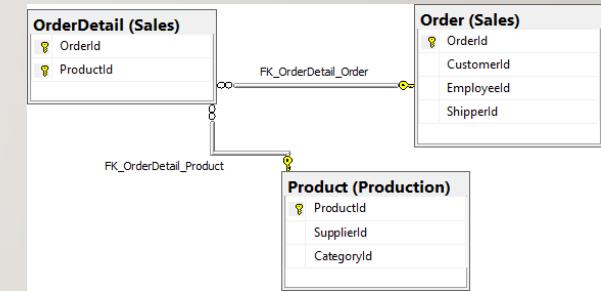


COMPLEX WORST#3 STANDARD/KEY VIEW

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 GMVNL
2	10249	10249	42	Product RUVNL
3	10249	10249	72	Product GEBCO
4	10249	10250	14	Product PVCIB
5	10249	10250	51	Product APIUJ
6	10250	10251	41	Product TTEEE
7	10250	10251	51	Product APIUJ
8	10250	10251	65	Product XYWBZ
9	10251	10252	22	Product CHFHY
10	10251	10252	37	Product CYLCI
11	10251	10252	65	Product XYWBZ
12	10252	10253	20	Product GHFFP
13	10252	10253	33	Product ASTMK
14	10252	10253	60	Product WHBYK
15	10253	10254	31	Product KWONC
16	10253	10254	39	Product LSOFI
17	10253	10254	49	Product FPYFL
18	10254	10255	24	Product GDDNU
19	10254	10255	58	Product YYWRT

JSON Output:

```
[{"Order": {"OrderID": 10724, "NextOrderID": 11078, "ProductID": 39, "ProductName": "Product LSOFL"}, {"Order": {"OrderID": 11077, "NextOrderID": 11078, "ProductID": 41, "ProductName": "Product TIEEX"}, {"Order": {"OrderID": 11077, "NextOrderID": 11078, "ProductID": 46, "ProductName": "Product CBRLB"}, {"Order": {"OrderID": 11077, "NextOrderID": 11078, "ProductID": 52, "ProductName": "Product QSRKF"}, {"Order": {"OrderID": 11077, "NextOrderID": 11078, "ProductID": 55, "ProductName": "Product YYWRT"}, {"Order": {"OrderID": 11077, "NextOrderID": 11078, "ProductID": 60, "ProductName": "Product WHBYK"}, {"Order": {"OrderID": 11077, "NextOrderID": 11078, "ProductID": 64, "ProductName": "Product HQDEK"}, {"Order": {"OrderID": 11077, "NextOrderID": 11078, "ProductID": 66, "ProductName": "Product LQMNH"}, {"Order": {"OrderID": 11077, "NextOrderID": 11078, "ProductID": 73, "ProductName": "Product WEUJZ"}, {"Order": {"OrderID": 11077, "NextOrderID": 11078, "ProductID": 75, "ProductName": "Product SNRLO"}, {"Order": {"OrderID": 11077, "NextOrderID": 11078, "ProductID": 77, "ProductName": "Product LUNZI"}]}
```



SIMPLE BEST#1 STANDARD/KEY VIEW

Standard View:

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

FK_SalesOrderHeader_CreditCard_CreditCardID

CreditCard (Sales)	Column Name	Data Type	Allow Nulls
CreditCardID	int		
CardType	nvarchar(50)		
CardNumber	nvarchar(25)		
ExpMonth	tinyint		
ExpYear	smallint		
ModifiedDate	datetime		

Key View:

SalesOrderHeader (Sale
SalesOrderID
SalesOrderNumber
CustomerID
SalesPersonID
TerritoryID
BillToAddressID
ShipToAddressID
ShipMethodID
CreditCardID
CurrencyRateID
rowguid

FK_SalesOrderHeader_CreditCard_CreditCardID

CreditCard (Sales)	CreditCardID	CardNumber
--------------------	--------------	------------



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	ASC
	OrderDate	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:

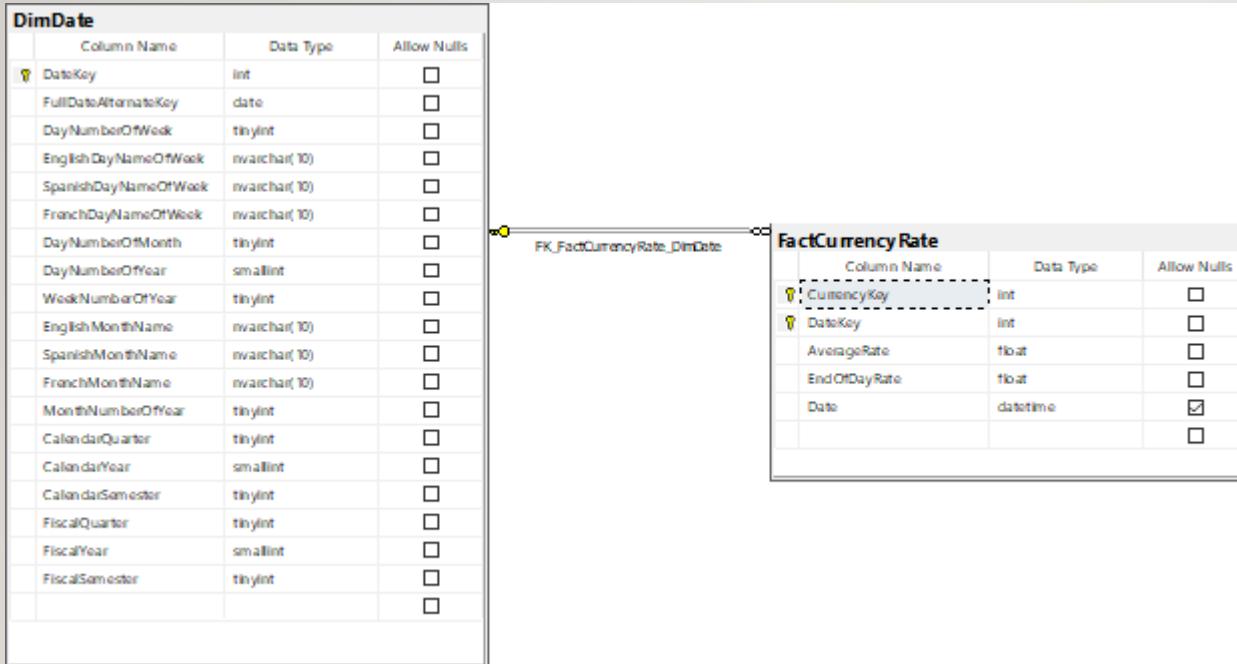
	CreditCardID	OrderDate	DueDate	ShipDate	Status
1	1	2013-07-29 00:00:00.000	2013-08-10 00:00:00.000	2013-08-05 00:00:00.000	5
2	1	2013-10-06 00:00:00.000	2013-10-13 00:00:00.000	2013-10-10 00:00:00.000	5
3	2	2013-12-05 00:00:00.000	2013-12-17 00:00:00.000	2013-12-12 00:00:00.000	5
4	2	2014-06-07 00:00:00.000	2014-06-17 00:00:00.000	2014-06-12 00:00:00.000	5
5	2	2014-06-23 00:00:00.000	2014-07-03 00:00:00.000	2014-06-30 00:00:00.000	5
6	3	2014-01-14 00:00:00.000	2014-01-26 00:00:00.000	2014-01-21 00:00:00.000	5
7	4	2013-05-20 00:00:00.000	2013-06-01 00:00:00.000	2013-05-27 00:00:00.000	5
8	4	2013-12-20 00:00:00.000	2014-01-09 00:00:00.000	2014-01-04 00:00:00.000	5
9	5	2013-02-01 00:00:00.000	2013-02-10 00:00:00.000	2013-02-08 00:00:00.000	5
10	5	2013-13-19 00:00:00.000	2013-13-21 00:00:00.000	2013-13-20 00:00:00.000	5
11	5	2014-02-17 00:00:00.000	2014-03-05 00:00:00.000	2014-02-28 00:00:00.000	5
12	6	2014-04-10 00:00:00.000	2014-04-22 00:00:00.000	2014-04-17 00:00:00.000	5
13	7	2013-02-01 00:00:00.000	2013-02-13 00:00:00.000	2013-02-08 00:00:00.000	5
14	7	2014-01-09 00:00:00.000	2014-01-21 00:00:00.000	2014-01-16 00:00:00.000	5
15	8	2013-06-30 00:00:00.000	2013-07-12 00:00:00.000	2013-07-07 00:00:00.000	5
16	8	2013-09-30 00:00:00.000	2013-10-12 00:00:00.000	2013-10-07 00:00:00.000	5
17	8	2015-12-31 00:00:00.000	2016-01-12 00:00:00.000	2014-07-07 00:00:00.000	5
18	8	2014-03-31 00:00:00.000	2014-04-12 00:00:00.000	2014-04-07 00:00:00.000	5
19	9	2013-09-23 00:00:00.000	2013-10-05 00:00:00.000	2013-09-30 00:00:00.000	5

JSON Output:

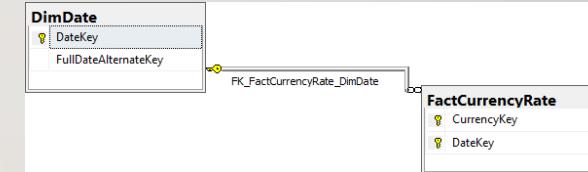


MEDIUM BEST#2 STANDARD/KEY VIEW

Standard View:



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

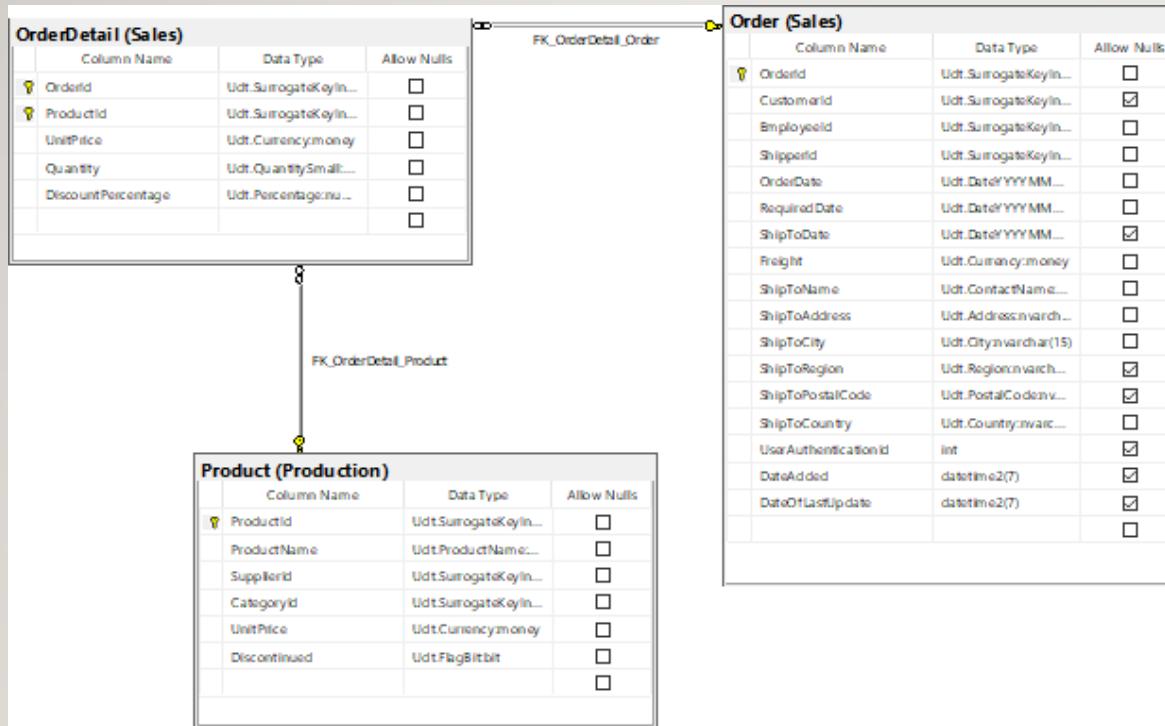
JSON Output:

DateKey	spanishdate
20140201	"DateKey": "20140201", "spanishdate": "Febrero Domingo 2014"
20140202	"DateKey": "20140202", "spanishdate": "Febrero Lunes 2014"
20140203	"DateKey": "20140203", "spanishdate": "Febrero Martes 2014"
20140204	"DateKey": "20140204", "spanishdate": "Febrero Miércoles 2014"
20140205	"DateKey": "20140205", "spanishdate": "Febrero Jueves 2014"
20140206	"DateKey": "20140206", "spanishdate": "Febrero Viernes 2014"
20140207	"DateKey": "20140207", "spanishdate": "Febrero Sábado 2014"
20140208	"DateKey": "20140208", "spanishdate": "Febrero Domingo 2014"
20140209	"DateKey": "20140209", "spanishdate": "Febrero Lunes 2014"
20140210	"DateKey": "20140210", "spanishdate": "Febrero Martes 2014"
20140211	"DateKey": "20140211", "spanishdate": "Febrero Miércoles 2014"
20140212	"DateKey": "20140212", "spanishdate": "Febrero Jueves 2014"
20140213	"DateKey": "20140213", "spanishdate": "Febrero Viernes 2014"
20140214	"DateKey": "20140214", "spanishdate": "Febrero Domingo 2014"
20140215	"DateKey": "20140215", "spanishdate": "Febrero Lunes 2014"
20140216	"DateKey": "20140216", "spanishdate": "Febrero Martes 2014"
20140217	"DateKey": "20140217", "spanishdate": "Febrero Miércoles 2014"
20140218	"DateKey": "20140218", "spanishdate": "Febrero Jueves 2014"
20140219	"DateKey": "20140219", "spanishdate": "Febrero Viernes 2014"
20140220	"DateKey": "20140220", "spanishdate": "Febrero Domingo 2014"
20140221	"DateKey": "20140221", "spanishdate": "Febrero Lunes 2014"
20140222	"DateKey": "20140222", "spanishdate": "Febrero Martes 2014"
20140223	"DateKey": "20140223", "spanishdate": "Febrero Miércoles 2014"
20140224	"DateKey": "20140224", "spanishdate": "Febrero Jueves 2014"
20140225	"DateKey": "20140225", "spanishdate": "Febrero Viernes 2014"
20140226	"DateKey": "20140226", "spanishdate": "Febrero Domingo 2014"
20140227	"DateKey": "20140227", "spanishdate": "Febrero Lunes 2014"
20140228	"DateKey": "20140228", "spanishdate": "Febrero Martes 2014"
20140229	"DateKey": "20140229", "spanishdate": "Febrero Miércoles 2014"
20140230	"DateKey": "20140230", "spanishdate": "Febrero Jueves 2014"
20140231	"DateKey": "20140231", "spanishdate": "Febrero Viernes 2014"

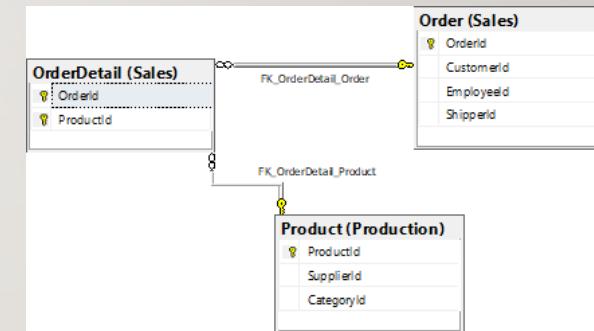


COMPLEX BEST#3 STANDARD KEY/VIEW

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 GMVUN	Expensive
2	10248	42	Product RUVNM	Expensive
3	10248	72	Product GEEDD	Expensive
4	10249	14	Product PWGJB	Expensive
5	10249	51	Product APITJ	Expensive
6	10250	41	Product TTEEX	Expensive
7	10250	51	Product APITJ	Expensive
8	10250	65	Product XWIBZ	Expensive
9	10251	22	Product CPHYF	Expensive
10	10251	57	Product OVLQJ	Expensive
11	10251	65	Product XWIBZ	Expensive
12	10252	20	Product GHFFP	Expensive
13	10252	33	Product ASTMN	Expensive
14	10252	60	Product WHHVK	Expensive
15	10253	31	Product KWICC	Expensive
16	10253	39	Product LSOFI	Expensive
17	10253	49	Product FPPFN	Expensive
18	10254	24	Product QODNJ	Expensive
19	10254	55	Product YWIRT	Expensive

JSON Output:

```
[{"OrderId": 10248, "ProductId": 11, "ProductName": "Product GMVUN", "PriceCheck": "Expensive"}, {"OrderId": 10248, "ProductId": 42, "ProductName": "Product RUVNM", "PriceCheck": "Expensive"}, {"OrderId": 10248, "ProductId": 72, "ProductName": "Product GEEDD", "PriceCheck": "Expensive"}, {"OrderId": 10249, "ProductId": 14, "ProductName": "Product PWGJB", "PriceCheck": "Expensive"}, {"OrderId": 10249, "ProductId": 51, "ProductName": "Product APITJ", "PriceCheck": "Expensive"}, {"OrderId": 10250, "ProductId": 41, "ProductName": "Product TTEEX", "PriceCheck": "Expensive"}, {"OrderId": 10250, "ProductId": 51, "ProductName": "Product APITJ", "PriceCheck": "Expensive"}, {"OrderId": 10250, "ProductId": 65, "ProductName": "Product XWIBZ", "PriceCheck": "Expensive"}, {"OrderId": 10251, "ProductId": 22, "ProductName": "Product CPHYF", "PriceCheck": "Expensive"}, {"OrderId": 10251, "ProductId": 57, "ProductName": "Product OVLQJ", "PriceCheck": "Expensive"}, {"OrderId": 10251, "ProductId": 65, "ProductName": "Product XWIBZ", "PriceCheck": "Expensive"}, {"OrderId": 10252, "ProductId": 20, "ProductName": "Product GHFFP", "PriceCheck": "Expensive"}, {"OrderId": 10252, "ProductId": 33, "ProductName": "Product ASTMN", "PriceCheck": "Expensive"}, {"OrderId": 10252, "ProductId": 60, "ProductName": "Product WHHVK", "PriceCheck": "Expensive"}, {"OrderId": 10253, "ProductId": 31, "ProductName": "Product KWICC", "PriceCheck": "Expensive"}, {"OrderId": 10253, "ProductId": 39, "ProductName": "Product LSOFI", "PriceCheck": "Expensive"}, {"OrderId": 10253, "ProductId": 49, "ProductName": "Product FPPFN", "PriceCheck": "Expensive"}, {"OrderId": 10254, "ProductId": 24, "ProductName": "Product QODNJ", "PriceCheck": "Expensive"}, {"OrderId": 10254, "ProductId": 55, "ProductName": "Product YWIRT", "PriceCheck": "Expensive"}]
```



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:

```
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

--FOR JSON PATH, ROOT ('Supplier for Each Product'), INCLUDE_NULL_VALUES;
```

Results

ProductId	SupplierCompanyName
1	Supplier SWRXU
2	Supplier SWRXU
3	Supplier SWRXU
4	Supplier VHQZD
5	Supplier VHQZD
6	Supplier STUAZ
7	Supplier STUAZ
8	Supplier STUAZ
9	Supplier QVFD
10	Supplier QVFD
11	Supplier EPMC
12	Supplier EPMC
13	Supplier GWUSF
14	Supplier GWUSF
15	Supplier GWUSF
16	Supplier GORCV
17	Supplier GORCV
18	Supplier GORCV
19	Supplier BWYFE

Messages

```
Query executed successfully.
```

localhost,12001 (15.0 RTM) | sa (60) | Northwinds2020TSQLV6 | 00:00:00 | 77 rows

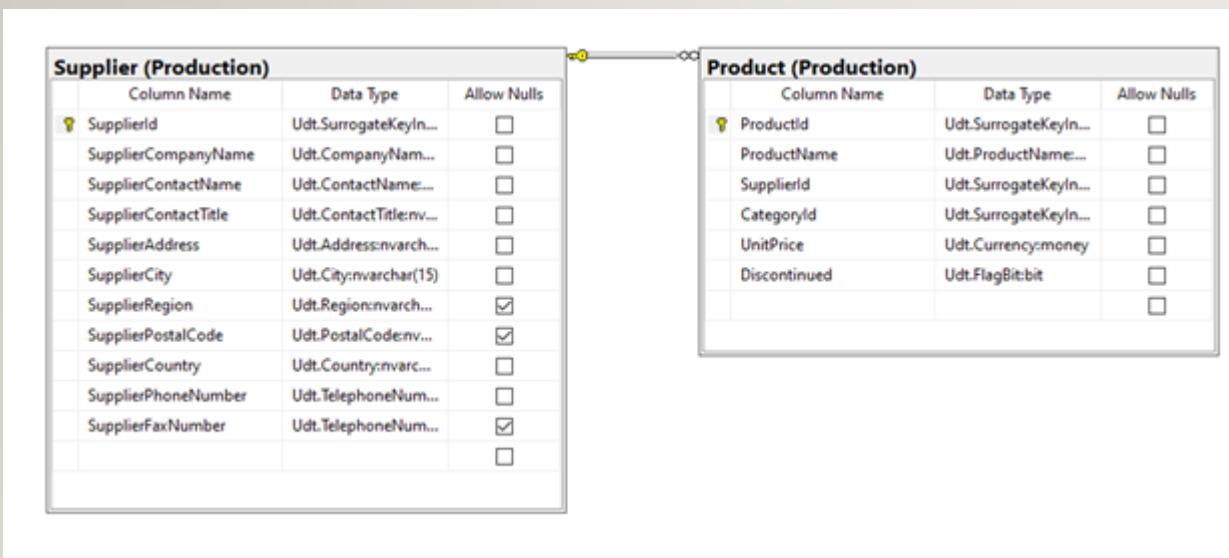
JSON Output:

```
"Supplier for Each Product": [
    {"ProductId": 1,
     "SupplierCompanyName": "Supplier SWRXU"
    },
    {"ProductId": 2,
     "SupplierCompanyName": "Supplier SWRXU"
    },
    {"ProductId": 3,
     "SupplierCompanyName": "Supplier SWRXU"
    },
    {"ProductId": 4,
     "SupplierCompanyName": "Supplier VHQZD"
    },
    {"ProductId": 5,
     "SupplierCompanyName": "Supplier VHQZD"
    },
    {"ProductId": 6,
     "SupplierCompanyName": "Supplier STUAZ"
    }
]
```

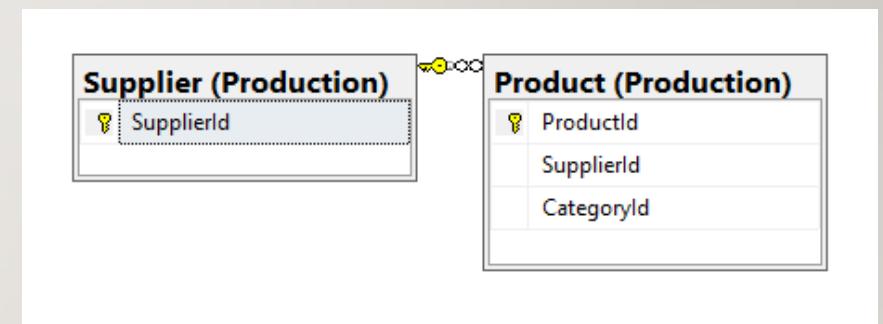


JONATHAN'S QUERY (SIMPLE) - BEST #1 VIEWS

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)

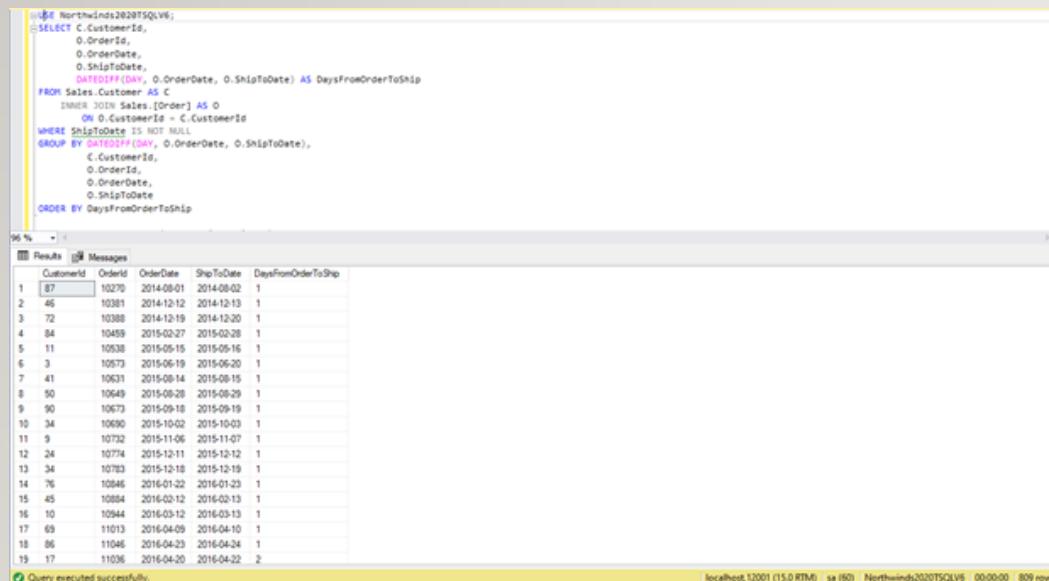
AS DaysFromOrderToShip

```
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

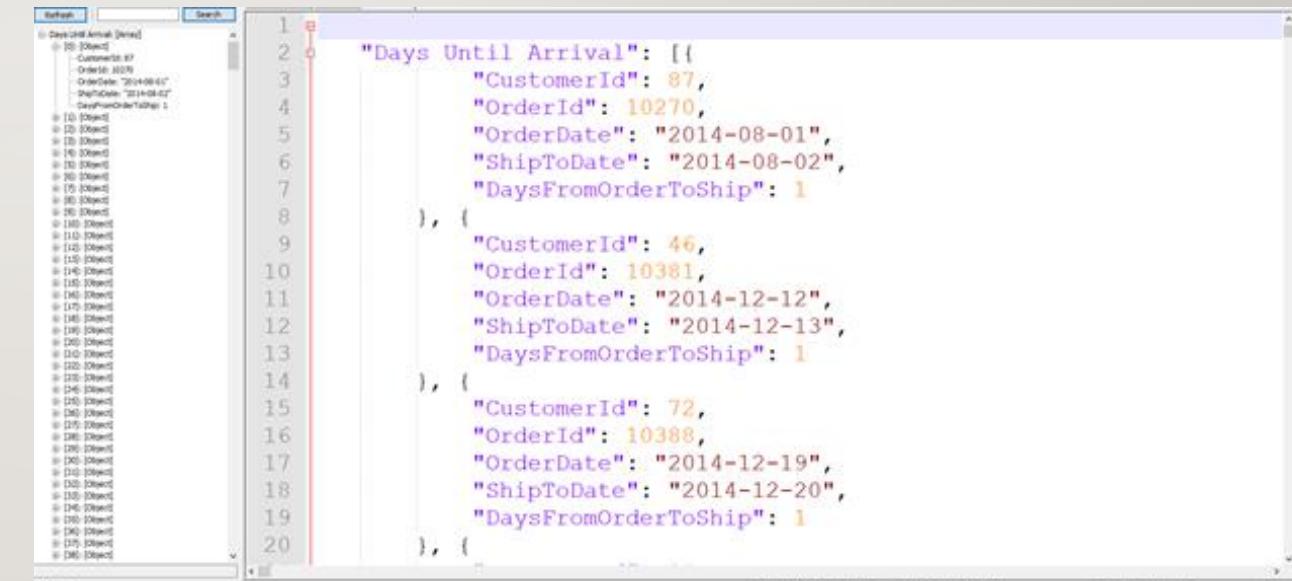
Relational Output:



A screenshot of SQL Server Management Studio showing a query results grid. The query retrieves data from the Northwind database, specifically from the Sales.Customer and Sales.Order tables. It calculates the number of days between the OrderDate and ShipToDate for each order, ordering the results by this calculated value. The results show various customer IDs, order IDs, dates, and the calculated days difference.

CustomerID	OrderId	OrderDate	ShipToDate	DaysFromOrderToShip
87	10270	2014-08-01	2014-08-02	1
46	10381	2014-12-12	2014-12-13	1
72	10388	2014-12-19	2014-12-20	1
84	10459	2015-02-27	2015-02-28	1
11	10538	2015-05-15	2015-05-16	1
3	10573	2015-06-19	2015-06-20	1
41	10631	2015-08-15	2015-08-15	0
50	10649	2015-08-28	2015-08-29	1
90	10673	2015-09-15	2015-09-19	1
34	10690	2015-10-01	2015-10-03	1
9	10732	2015-11-06	2015-11-07	1
24	10774	2015-12-11	2015-12-12	1
34	10783	2015-12-15	2015-12-19	1
76	10846	2016-01-22	2016-01-23	1
45	10884	2016-02-12	2016-02-13	1
10	10944	2016-03-12	2016-03-13	1
69	11013	2016-04-09	2016-04-10	1
86	11046	2016-04-23	2016-04-24	1
17	11056	2016-04-20	2016-04-22	2

JSON Output:



A screenshot of a JSON file viewer showing the output of the query in JSON format. The JSON structure is an array of objects, where each object represents an order. Each object contains the CustomerId, OrderId, OrderDate, ShipToDate, and DaysFromOrderToShip fields. The data is identical to the relational output shown above.

```
[{"Days Until Arrival": 1, "CustomerId": 87, "OrderId": 10270, "OrderDate": "2014-08-01", "ShipToDate": "2014-08-02", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 46, "OrderId": 10381, "OrderDate": "2014-12-12", "ShipToDate": "2014-12-13", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 72, "OrderId": 10388, "OrderDate": "2014-12-19", "ShipToDate": "2014-12-20", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 34, "OrderId": 10459, "OrderDate": "2015-02-27", "ShipToDate": "2015-02-28", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 0, "CustomerId": 11, "OrderId": 10538, "OrderDate": "2015-05-15", "ShipToDate": "2015-05-16", "DaysFromOrderToShip": 0}, {"Days Until Arrival": 1, "CustomerId": 3, "OrderId": 10573, "OrderDate": "2015-06-19", "ShipToDate": "2015-06-20", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 0, "CustomerId": 41, "OrderId": 10631, "OrderDate": "2015-08-15", "ShipToDate": "2015-08-15", "DaysFromOrderToShip": 0}, {"Days Until Arrival": 1, "CustomerId": 50, "OrderId": 10649, "OrderDate": "2015-08-28", "ShipToDate": "2015-08-29", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 90, "OrderId": 10673, "OrderDate": "2015-09-15", "ShipToDate": "2015-09-19", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 34, "OrderId": 10690, "OrderDate": "2015-10-01", "ShipToDate": "2015-10-03", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 9, "OrderId": 10732, "OrderDate": "2015-11-06", "ShipToDate": "2015-11-07", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 24, "OrderId": 10774, "OrderDate": "2015-12-11", "ShipToDate": "2015-12-12", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 34, "OrderId": 10783, "OrderDate": "2015-12-15", "ShipToDate": "2015-12-19", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 76, "OrderId": 10846, "OrderDate": "2016-01-22", "ShipToDate": "2016-01-23", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 45, "OrderId": 10884, "OrderDate": "2016-02-12", "ShipToDate": "2016-02-13", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 10, "OrderId": 10944, "OrderDate": "2016-03-12", "ShipToDate": "2016-03-13", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 69, "OrderId": 11013, "OrderDate": "2016-04-09", "ShipToDate": "2016-04-10", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 1, "CustomerId": 86, "OrderId": 11046, "OrderDate": "2016-04-23", "ShipToDate": "2016-04-24", "DaysFromOrderToShip": 1}, {"Days Until Arrival": 2, "CustomerId": 17, "OrderId": 11056, "OrderDate": "2016-04-20", "ShipToDate": "2016-04-22", "DaysFromOrderToShip": 2}]
```



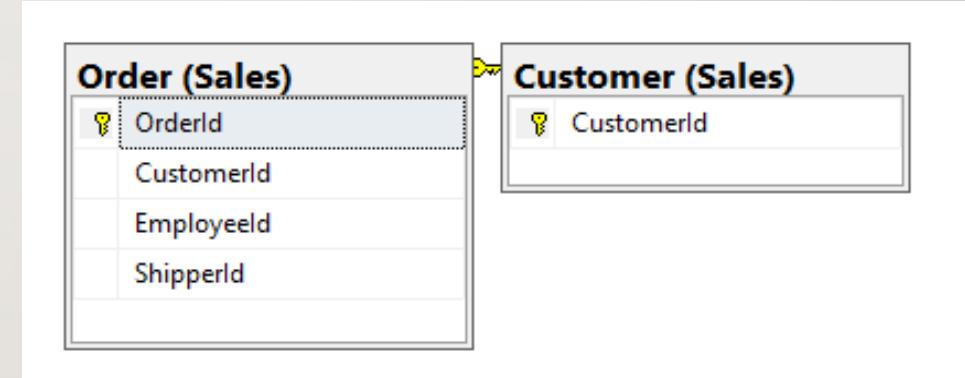
JONATHAN'S QUERY (MEDIUM) - BEST #2 VIEWS

Standard View

Order (Sales)		
Column Name	Data Type	Allow Nulls
OrderId	Udt.SurrogateKeyIn...	<input type="checkbox"/>
CustomerId	Udt.SurrogateKeyIn...	<input checked="" type="checkbox"/>
EmployeeId	Udt.SurrogateKeyIn...	<input type="checkbox"/>
ShipperId	Udt.SurrogateKeyIn...	<input type="checkbox"/>
OrderDate	Udt.DateYYYYMM...	<input type="checkbox"/>
RequiredDate	Udt.DateYYYYMM...	<input type="checkbox"/>
ShipToDate	Udt.DateYYYYMM...	<input checked="" type="checkbox"/>
Freight	Udt.Currency:money	<input type="checkbox"/>
ShipToName	Udt.ContactName:...	<input type="checkbox"/>
ShipToAddress	Udt.Address:nvarchar...	<input type="checkbox"/>
ShipToCity	Udt.City:nvarchar(15)	<input type="checkbox"/>
ShipToRegion	Udt.Region:nvarchar...	<input checked="" type="checkbox"/>
ShipToPostalCode	Udt.PostalCode:nv...	<input checked="" type="checkbox"/>
ShipToCountry	Udt.Country:nvarc...	<input type="checkbox"/>
UserAuthenticationId	int	<input checked="" type="checkbox"/>
DateAdded	datetime2(7)	<input checked="" type="checkbox"/>
DateOfLastUpdate	datetime2(7)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

Customer (Sales)		
Column Name	Data Type	Allow Nulls
CustomerId	Udt.SurrogateKeyIn...	<input type="checkbox"/>
CustomerCompanyName	Udt.CompanyNam...	<input type="checkbox"/>
CustomerContactName	Udt.ContactName:...	<input type="checkbox"/>
CustomerContactTitle	Udt.Title:nvarchar(3...	<input type="checkbox"/>
CustomerAddress	Udt.Address:nvarchar...	<input type="checkbox"/>
CustomerCity	Udt.City:nvarchar(15)	<input type="checkbox"/>
CustomerRegion	Udt.Region:nvarchar...	<input checked="" type="checkbox"/>
CustomerPostalCode	Udt.PostalCode:nv...	<input checked="" type="checkbox"/>
CustomerCountry	Udt.Country:nvarc...	<input type="checkbox"/>
CustomerPhoneNumber	Udt.TelephoneNum...	<input type="checkbox"/>
CustomerFaxNumber	Udt.TelephoneNum...	<input checked="" type="checkbox"/>

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 Northwind2020SQLV8;
DROP FUNCTION IF EXISTS dbo.UnitsToCountry;
GO
CREATE FUNCTION dbo.UnitsToCountry(@Country NVARCHAR(50))
RETURNS TABLE
AS
RETURN SELECT SUM(SO0.Quantity) AS TotalUnits, SO.ShipToRegion
FROM SALES..[ORDER] AS SO
LEFT JOIN SALES..[ORDERDETAIL] AS SO0
ON SO.ORDERID = SO0.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

ShipToRegion	TotalUnits
AK	603
CA	181
ID	4958
MT	59
NM	1383
OR	647
WA	1175

Query executed successfully.

JSON Output:

```
Refresh Search
ROOT
Orders to Country Region [Area]
  01:01:0001
    ShipToRegion: "AK"
      TotalUnits: 603
  01:02:0002
    ShipToRegion: "CA"
      TotalUnits: 181
  01:03:0003
    ShipToRegion: "ID"
      TotalUnits: 4958
  01:04:0004
    ShipToRegion: "MT"
      TotalUnits: 59
  01:05:0005
    ShipToRegion: "NM"
      TotalUnits: 1383
  01:06:0006
    ShipToRegion: "OR"
      TotalUnits: 647
  01:07:0007
    ShipToRegion: "WA"
      TotalUnits: 1175
  01:08:0008
    ShipToRegion: "NY"
      TotalUnits: 127
```

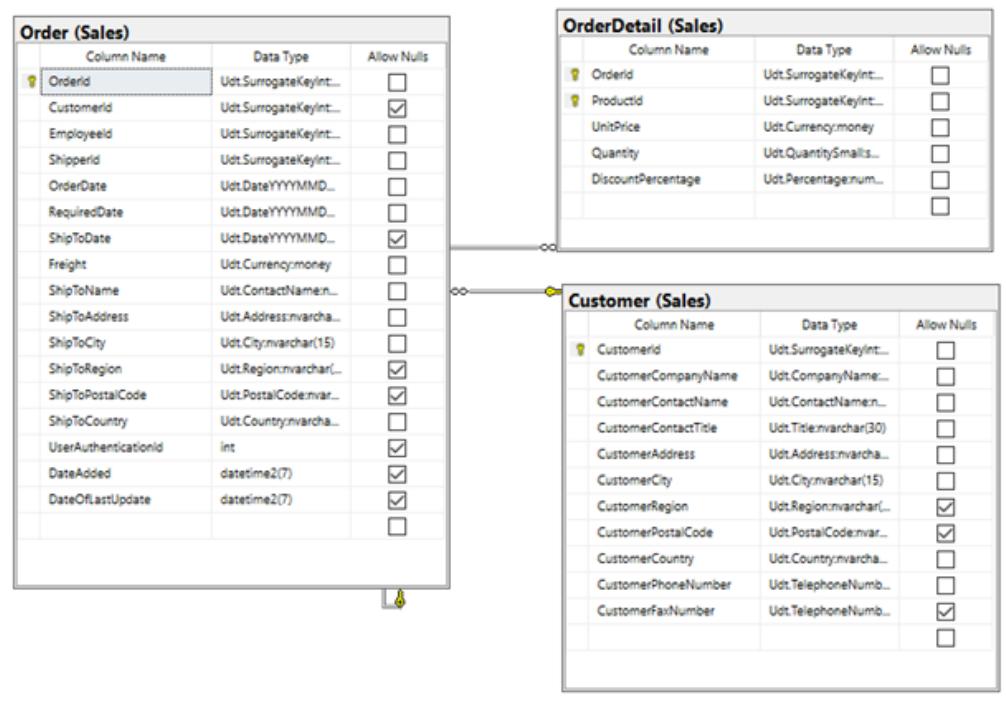
JSON File

Length: 688 Lines: 28 Ls: 1 Col: 2 Sel: 0 B Windows (OR F) VM-8 IN5

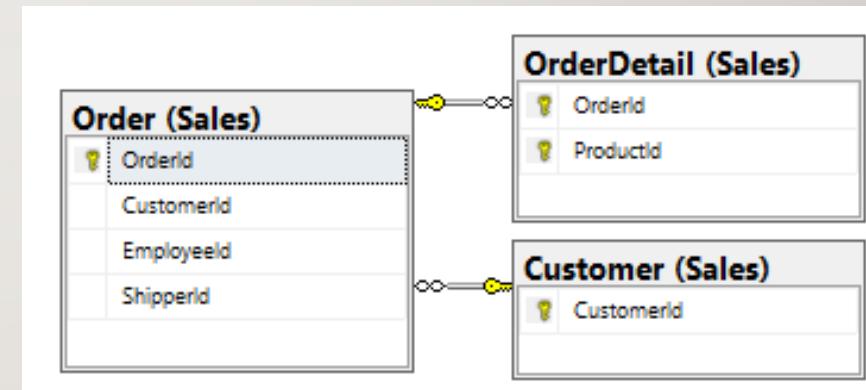


JONATHAN'S QUERY (COMPLEX) - BEST #3 VIEWS

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
```

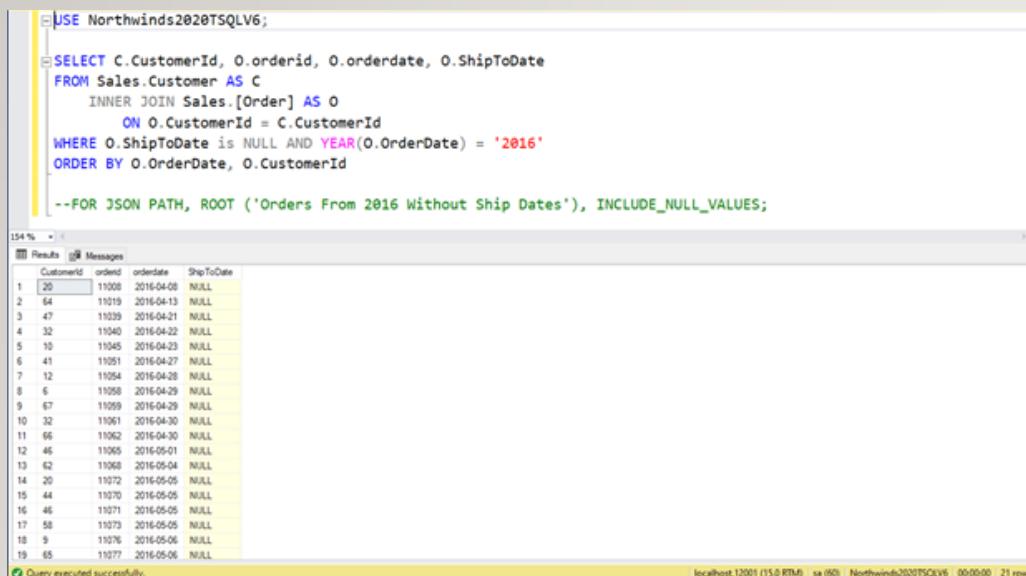
```
USE Northwinds2020TSQLV6;
```

```
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

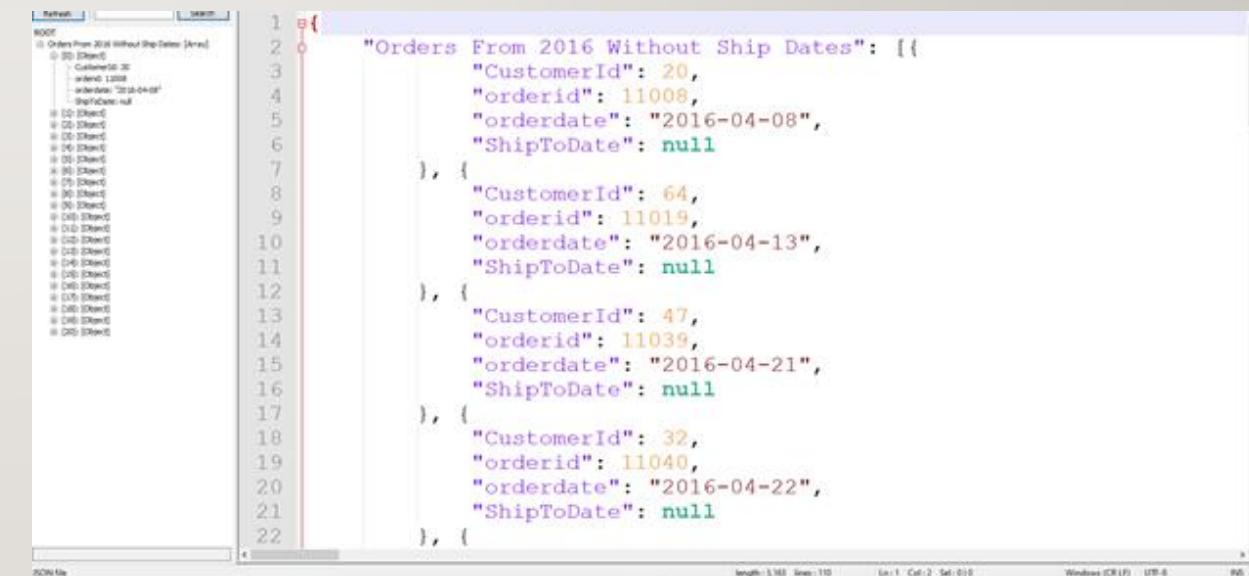
Relational Output:



```
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
20	11008	2016-04-08	NULL
64	11019	2016-04-13	NULL
47	11039	2016-04-21	NULL
32	11040	2016-04-22	NULL
10	11045	2016-04-23	NULL
41	11051	2016-04-27	NULL
12	11054	2016-04-28	NULL
6	11058	2016-04-29	NULL
67	11059	2016-04-29	NULL
32	11061	2016-04-30	NULL
66	11062	2016-04-30	NULL
46	11065	2016-05-01	NULL
52	11068	2016-05-04	NULL
20	11072	2016-05-05	NULL
44	11070	2016-05-05	NULL
46	11071	2016-05-05	NULL
58	11073	2016-05-05	NULL
9	11076	2016-05-06	NULL
65	11077	2016-05-06	NULL

Query executed successfully.



```
USE Northwinds2020TSQLV6;
SELECT * FROM [Order] WHERE ShipDate IS NULL AND YEAR(OrderDate) = '2016';
--FOR JSON PATH, ROOT ('Orders From 2016 Without Ship Dates'), INCLUDE_NULL_VALUES;
```

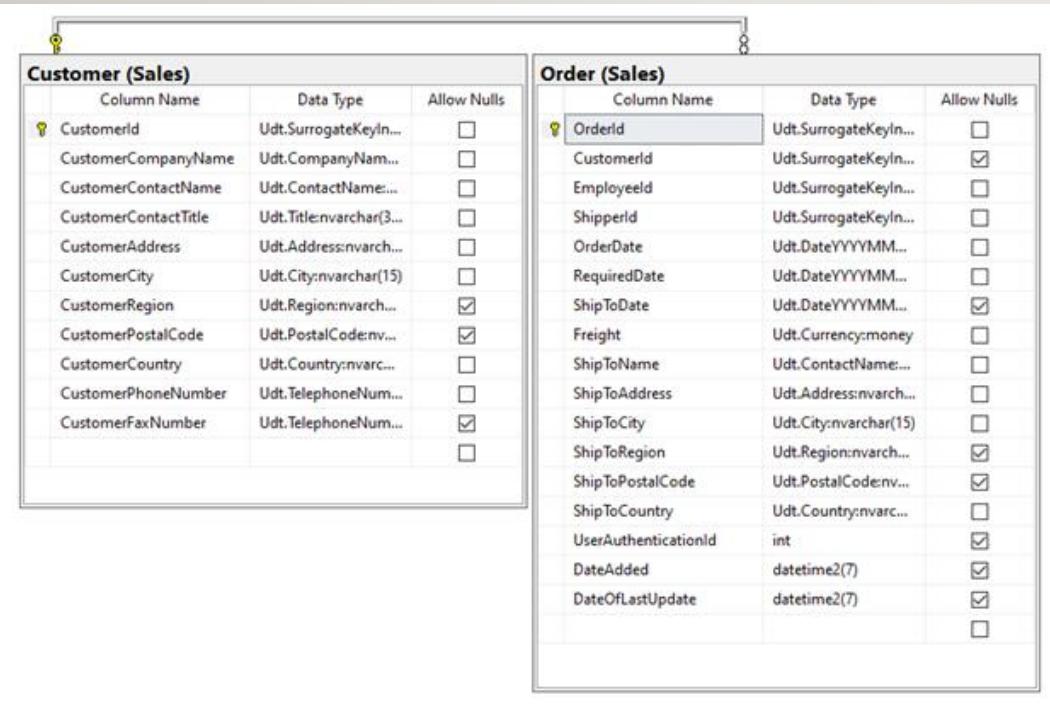
CustomerID	OrderID	OrderDate	ShipDate
20	11008	2016-04-08	NULL
64	11019	2016-04-13	NULL
47	11039	2016-04-21	NULL
32	11040	2016-04-22	NULL
10	11045	2016-04-23	NULL
41	11051	2016-04-27	NULL
12	11054	2016-04-28	NULL
6	11058	2016-04-29	NULL
67	11059	2016-04-29	NULL
32	11061	2016-04-30	NULL
66	11062	2016-04-30	NULL
46	11065	2016-05-01	NULL
52	11068	2016-05-04	NULL
20	11072	2016-05-05	NULL
44	11070	2016-05-05	NULL
46	11071	2016-05-05	NULL
58	11073	2016-05-05	NULL
9	11076	2016-05-06	NULL
65	11077	2016-05-06	NULL

```
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
    }
]
```

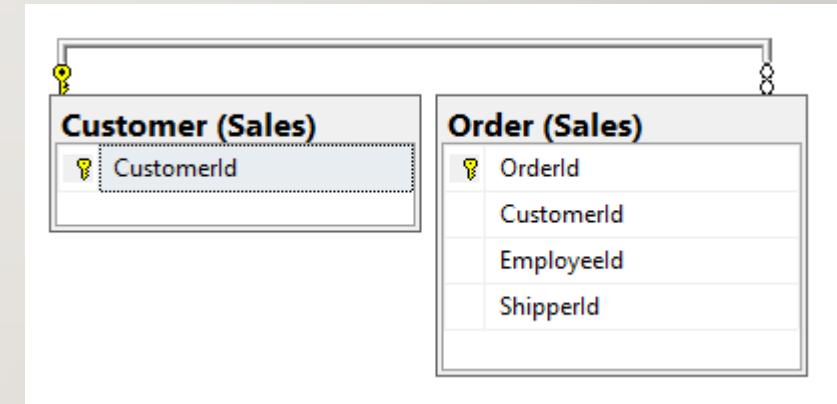


JONATHAN'S QUERY (SIMPLE) - BEST #1 VIEWS

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
Chief Executive Officer	125.50
Vice President of Production	84.1346
Vice President of Sales	72.1154
Vice President of Engineering	63.4615
Information Services Manager	50.4808
Chief Financial Officer	49.2379
European Sales Manager	48.101
North American Sales Manager	48.101
Pacific Sales Manager	48.101
Research and Development Manager	46.4308
Engineering Manager	43.2692
Finance Manager	43.2692
Research and Development Engineer	40.8554
Network Manager	39.6635
Database Administrator	38.4615
Senior Design Engineer	36.0577
Accounts Manager	34.7356
Design Engineer	32.6933
Network Administrator	32.4519

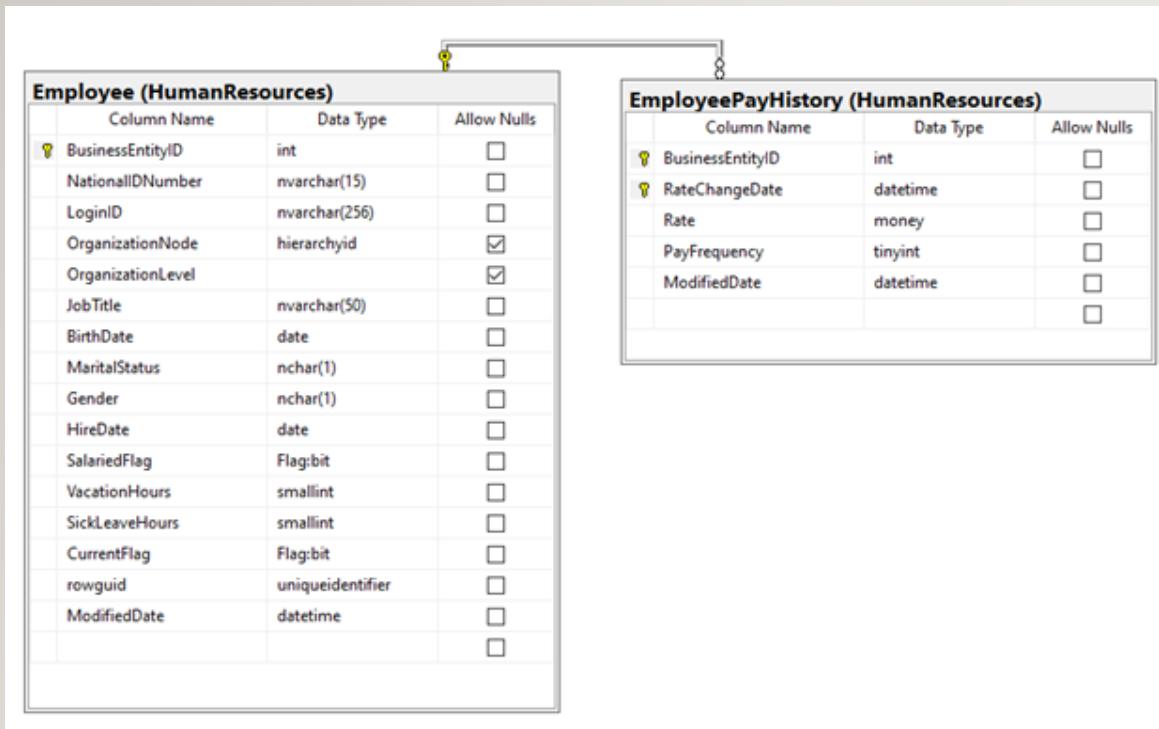
JSON Output:

```
1 {  
2     "Average Salary Per Position": [ {  
3         "JobTitle": "Chief Executive Officer",  
4         "AverageRate": 125.5000  
5     }, {  
6         "JobTitle": "Vice President of Production",  
7         "AverageRate": 84.1346  
8     }, {  
9         "JobTitle": "Vice President of Sales",  
10        "AverageRate": 72.1154  
11     }, {  
12         "JobTitle": "Vice President of Engineering",  
13         "AverageRate": 63.4615  
14     }, {  
15         "JobTitle": "Information Services Manager",  
16         "AverageRate": 50.4808  
17     }, {  
18         "JobTitle": "Chief Financial Officer",  
19         "AverageRate": 49.2379  
20     } ]  
}
```

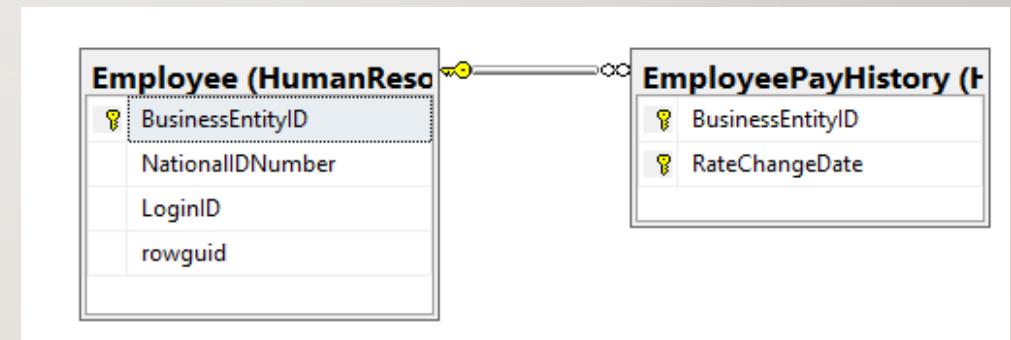


JONATHAN'S QUERY (MEDIUM) - WORST #2 VIEWS

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.AverageCustomerDiscount;
```

```
GO
```

```
CREATE FUNCTION  
dbo.AverageCustomerDiscount
```

```
(  
@CustomerID INT
```

```
)  
  
RETURNS TABLE
```

```
AS
```

```
    RETURN SELECT C.CustomerId,  
           AVG(OD.DiscountPercentage)  
           AS AverageDiscountPercent  
      FROM Sales.Customer AS C  
           INNER JOIN Sales.[Order] AS O  
             ON O.CustomerId = C.CustomerId  
           INNER  
          JOIN Sales.OrderDetail AS OD  
            ON OD.OrderId = O.Order  
           Id
```

```
      WHERE O.CustomerId = @Customer  
        ID  
      GROUP BY C.CustomerId;  
  
      GO
```

```
      DECLARE @CustomerId INT;  
      SET @CustomerId = I;  
  
      SELECT CustomerId,  
             AverageDiscountPercent  
        FROM dbo.AverageCustomerDiscount(@CustomerId)  
  
      DROP  
      FUNCTION IF EXISTS dbo.AverageCustomerDiscount;
```



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

Relational Output:

```
USE Northwind2020TSQLV6;
DROP FUNCTION IF EXISTS dbo.AverageCustomerDiscount;
GO
CREATE FUNCTION dbo.AverageCustomerDiscount
(
    @CustomerId INT
)
RETURNS TABLE
AS
RETURN SELECT C.CustomerId,
            AVG(OD.DiscountPercentage) AS AverageDiscountPercent
        FROM Sales.Customer AS C
        INNER JOIN Sales.[Order] AS O
            ON O.CustomerId = C.CustomerId
        INNER JOIN Sales.OrderDetail AS OD
            ON OD.OrderId = O.OrderId
        WHERE O.CustomerId = @CustomerId
        GROUP BY C.CustomerId;
GO

DECLARE @EmployeeId INT;
SET @EmployeeId = 1;

SELECT CustomerId,
        AverageDiscountPercent
FROM dbo.AverageCustomerDiscount(@EmployeeId);

--Uncomment Below to get JSON output
--FOR JSON PATH, ROOT ('Average Customer Discount'), INCLUDE_NULL_VALUES

DROP FUNCTION IF EXISTS dbo.AverageCustomerDiscount;
```

JSON Output:

```
ROOT
  "Average Customer Discount": [
    {
      "CustomerId": 1,
      "AverageDiscountPercent": 0.087500
    }
  ]
```



Query executed successfully.

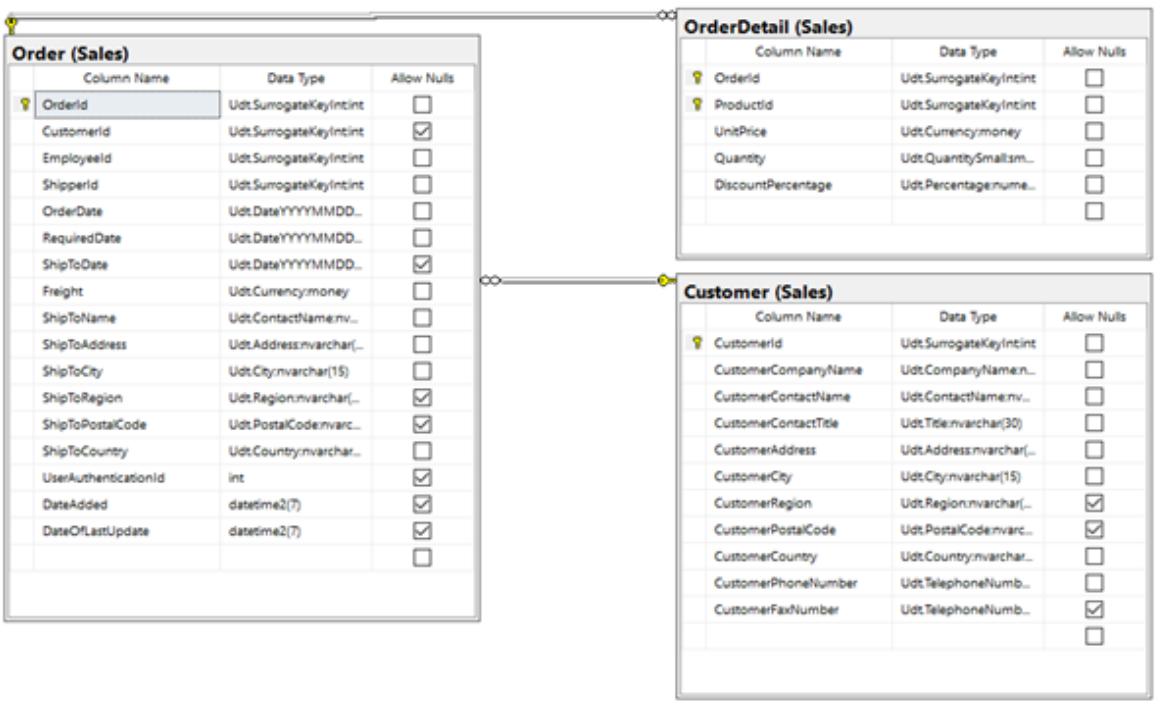
localhost,12001 (15.0 KtM) sa (60) Northwind2020TSQLV6 00:00:00 1 rows

length: 139 lines: 8 Ln:1 Col:2 Sel:0/0

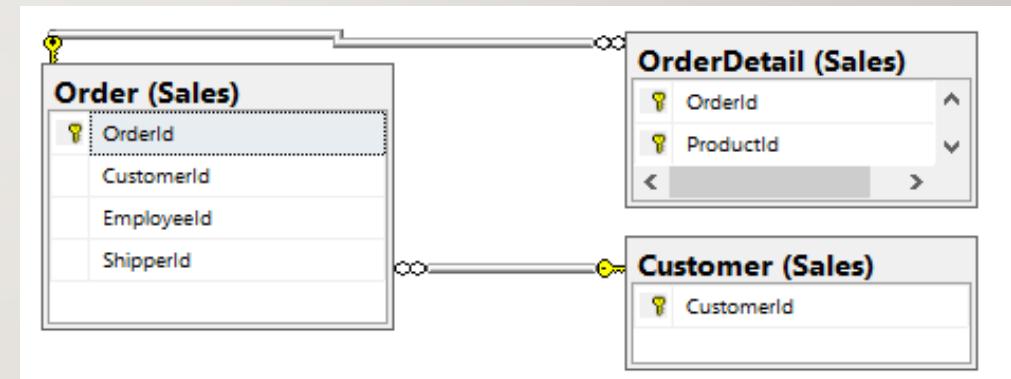
Windows 10 8.1 8 7 6 5 4 3 2 1

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



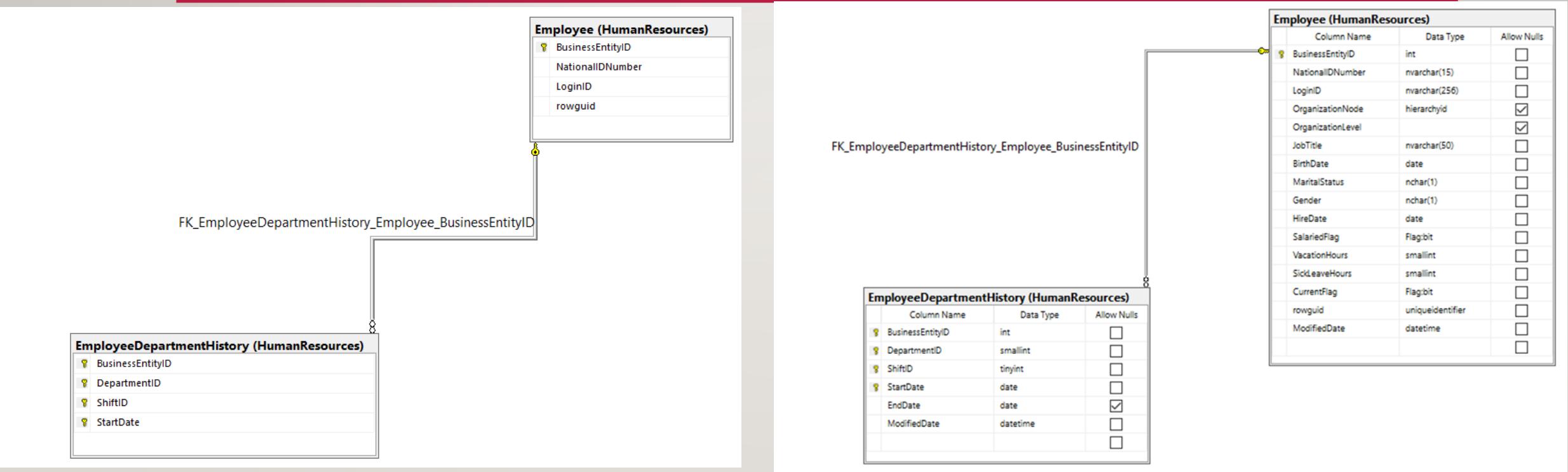
DANNY'S QUERY (SIMPLE) (WORST)

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)

The screenshot shows a SQL Server Management Studio (SSMS) interface with the following components:

- Query Editor:** Displays a T-SQL script with three numbered sections (Simple #1, Simple #2, Simple #3).

```
--Simple #1--  
SELECT PA.StateProvinceID = PSP.StateProvinceID  
FROM Person.StateProvince AS PA  
ORDER BY PA.StateProvinceID;  
  
--Simple #2--  
SELECT HREH.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.
```
- Results Grid:** Shows the output of the query for Simple #3, displaying 296 rows of address lines.
- Messages:** Shows a successful execution message: "Query executed successfully."
- JSToolNpp JSON Viewer:** A separate window or tab showing the JSON output of the query. It displays an array of objects representing employee shifts, with each object containing properties like BusinessEntityID, DepartmentID, ShiftID, and JobTitle.



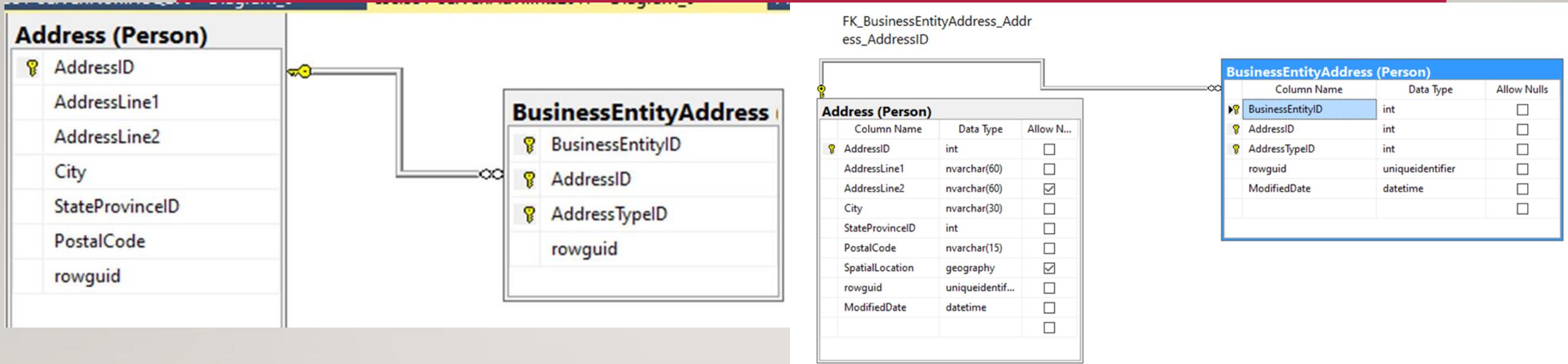
DANNY'S QUERY (SIMPLE) (TOP)

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)

The screenshot shows a comparison between a standard SQL query results table and a JSON output viewer.

Left Side (SQL Results):

```
--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;
```

Right Side (JSToolNpp JSON Viewer):

The JSON viewer displays the results of the query as an array of objects, each representing an address. The structure is as follows:

```
[  
  {  
    "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": "Dresden"  
  }]
```

The JSON viewer also includes a tree view on the left showing the structure of the JSON data, and a list of line numbers on the right corresponding to the JSON elements.



DANNY'S QUERY (MEDIUM) (TOP)

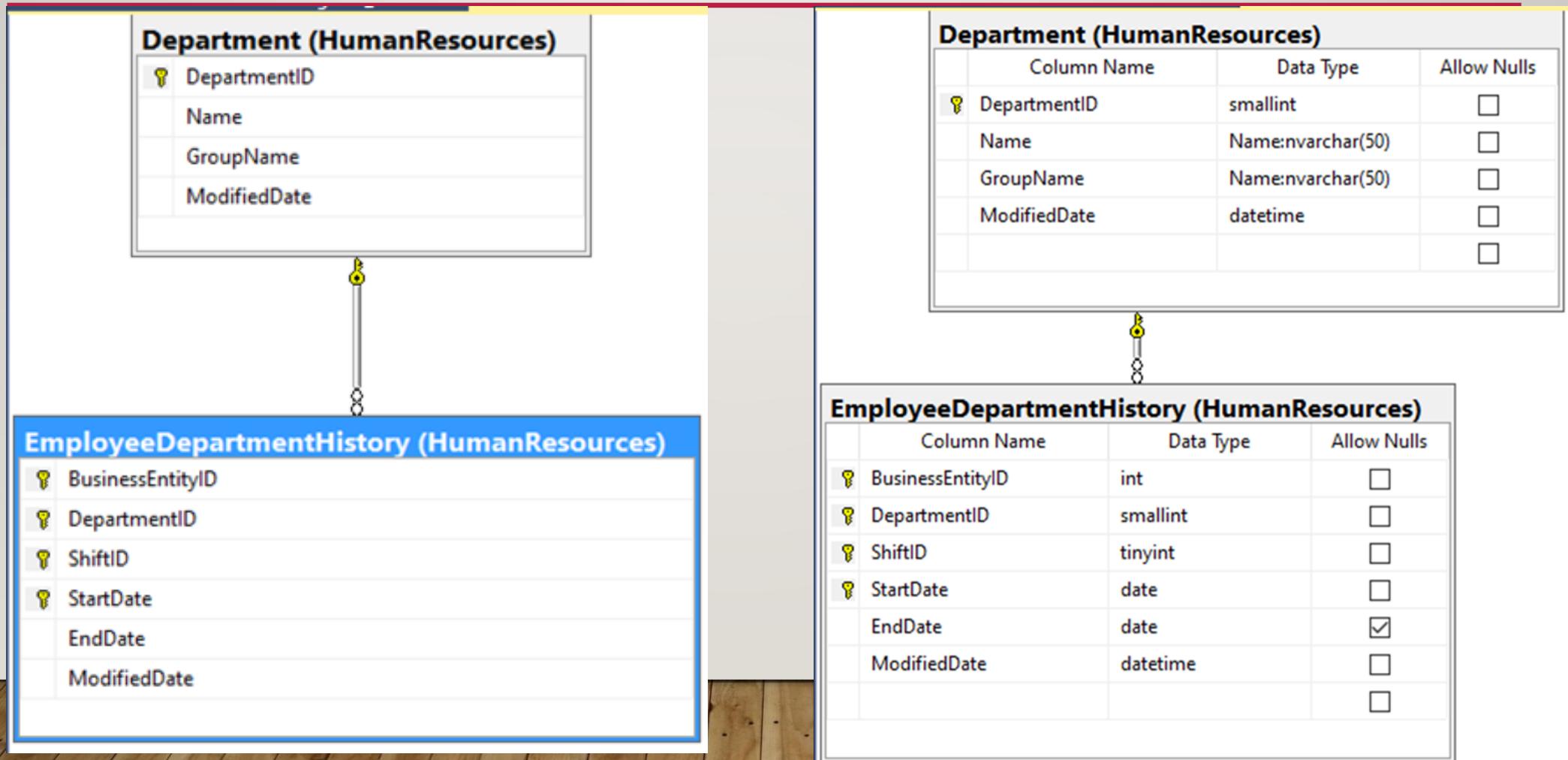
Display for each department: the department ID, the name of the department, the shift ID, and how many employees worked in that department during that shift

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

Query executed successfully.

localhost, 12001 (15.0 RTM) | sa (56) | AdventureWorks2017 | 00:00:00 | 29 rows

JSToolNpp JSON Viewer

Refresh Search

ROOT

NumberofEmployeesinDepartment: [

- [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]

117), {
118 "DepartmentID": 14,
119 "Name": "Facilities and Maintenance",
120 "Employee Number": 2,
121 "ShiftID": 2
}, {
122 "DepartmentID": 14,
123 "Name": "Facilities and Maintenance",
124 "Employee Number": 2,
125 "ShiftID": 3
}, {
126 "DepartmentID": 15,
127 "Name": "Shipping and Receiving",
128 "Employee Number": 3,
129 "ShiftID": 1
}, {
130 "DepartmentID": 15,
131 "Name": "Shipping and Receiving",
132 "Employee Number": 3,
133 "ShiftID": 1
}, {
134 "DepartmentID": 15,
135 "Name": "Shipping and Receiving",
136 "Employee Number": 2,
137 "ShiftID": 2
}, {
138 "DepartmentID": 15,
139 "Name": "Shipping and Receiving",
140 "Employee Number": 2,
141 "ShiftID": 3
}, {
142 "DepartmentID": 16,
143 "Name": "Executive",
144 "Employee Number": 2,
145 "ShiftID": 1
}, {
146 "DepartmentID": 16,
147 "Name": "Executive",
148 "Employee Number": 2,
149 "ShiftID": 1

Complex #7xml Medium #1xml Medium #2xml Medium #3xml Medium #4xml Medium #5xml Medium #6xml Medium



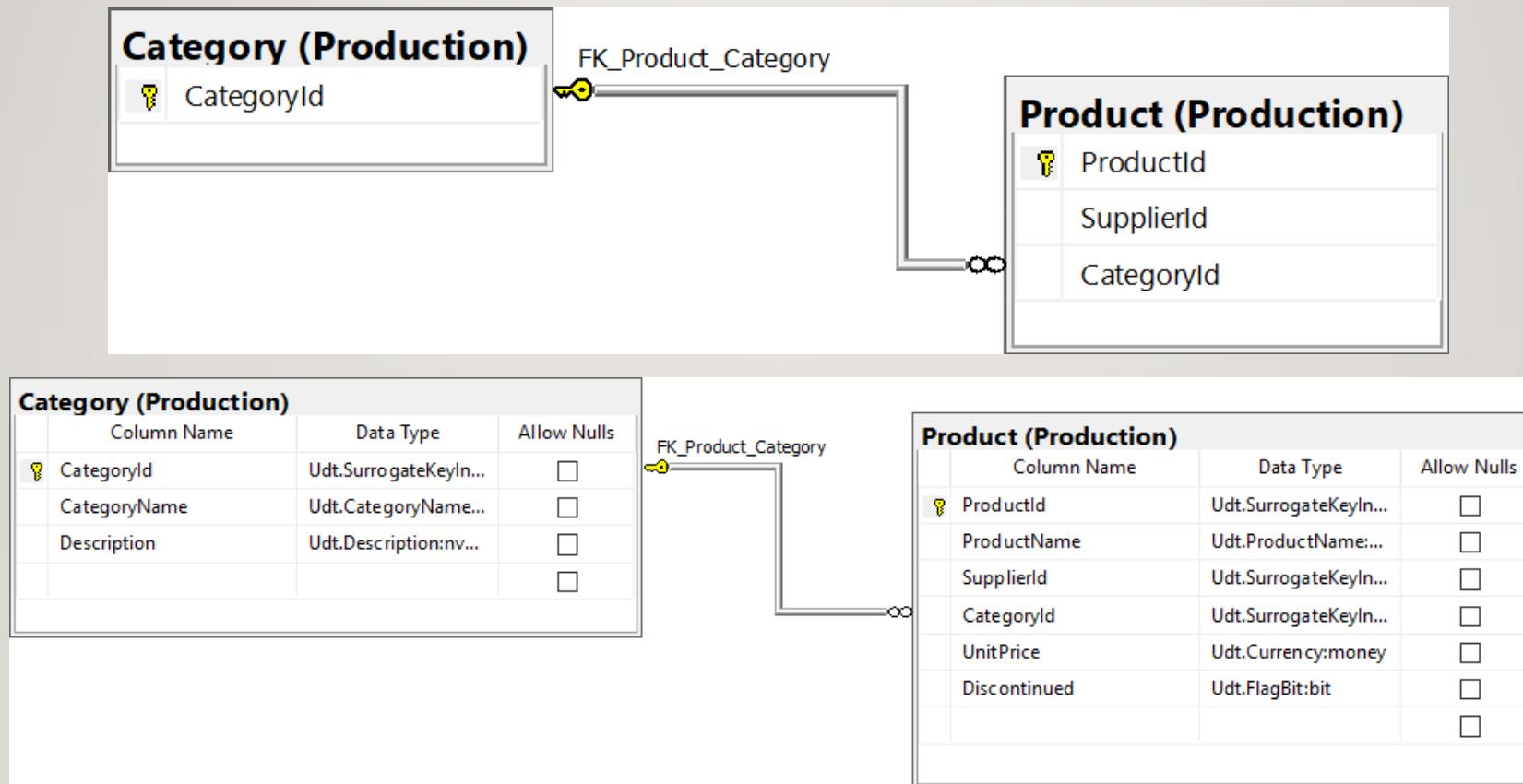
DANNY'S QUERY (MEDIUM) (WORST)

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)



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)

The screenshot shows a database query results window with two panes. The left pane displays a table of product categories with their counts and descriptions. The right pane shows the raw SQL query and its corresponding JSON output.

Table Data:

	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 Result Message: Query executed successfully.

Query Results Headers: localhost, 12001 (15.0 RTM) | sa (74) | Northwinds2020TSQLV6 | 00:00:00 | 8 rows

Raw SQL Query:

```
SELECT CategoryName, Description, COUNT(*) AS [Number of Products in this Category]
FROM Categories
ORDER BY CategoryName;
```

JSON Output:

```
1, {
  "CategoryName": "Beverages",
  "Description": "Soft drinks, coffees, teas, beers, and ales"
}, {
  "CategoryName": "Condiments",
  "Description": "Sweet and savory sauces, relishes, spreads, and seasonings"
}, {
  "CategoryName": "Confections",
  "Description": "Desserts, candies, and sweet breads"
}, {
  "CategoryName": "Dairy Products",
  "Description": "Cheeses"
}, {
  "CategoryName": "Grains\Cereals",
  "Description": "Breads, crackers, pasta, and cereal"
}, {
  "CategoryName": "Meat\Poultry",
  "Description": "Prepared meats"
}, {
  "CategoryName": "Produce",
  "Description": "Dried fruit and bean curd"
}, {
  "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;
```



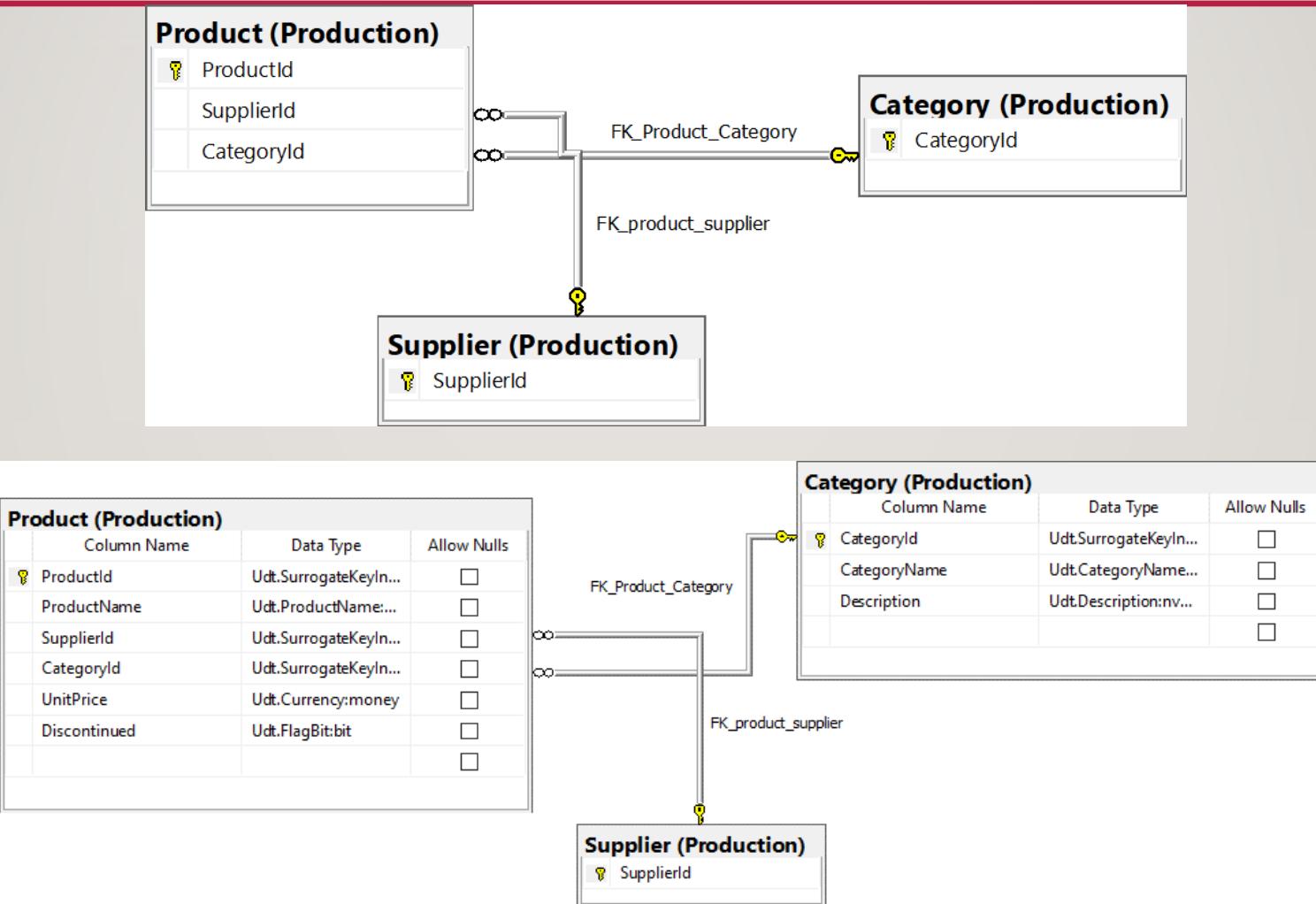
DANNY'S QUERY (COMPLEX) (TOP)

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)

Screenshot of a SQL Server Management Studio (SSMS) query results window.

The Results tab shows the following table:

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

The Messages tab is empty.

Status bar message: Query executed successfully.

Connection information: localhost, 12001 (15.0 RTM) | sa (52) | Northwinds2020TSQLV6 | 00:00:00 | 3 rows

Screenshot of JSToolNpp JSON Viewer showing the output of the complex query.

The left pane shows the JSON structure:

```
ROOT
  Product from this Supplier: [Array]
    [0]: [Object]
    [1]: [Object]
    [2]: [Object]
```

The right pane shows the generated JSON code:

```
1  "Product from this Supplier": [
2    {
3      "ProductId": 25,
4      "ProductName": "Product LYLNI",
5      "CategoryId": 3
6    },
7    {
8      "ProductId": 26,
9      "ProductName": "Product HLGZA",
10     "CategoryId": 3
11   },
12   {
13     "ProductId": 27,
14     "ProductName": "Product SMIOH",
15     "CategoryId": 3
16   }
]
```



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



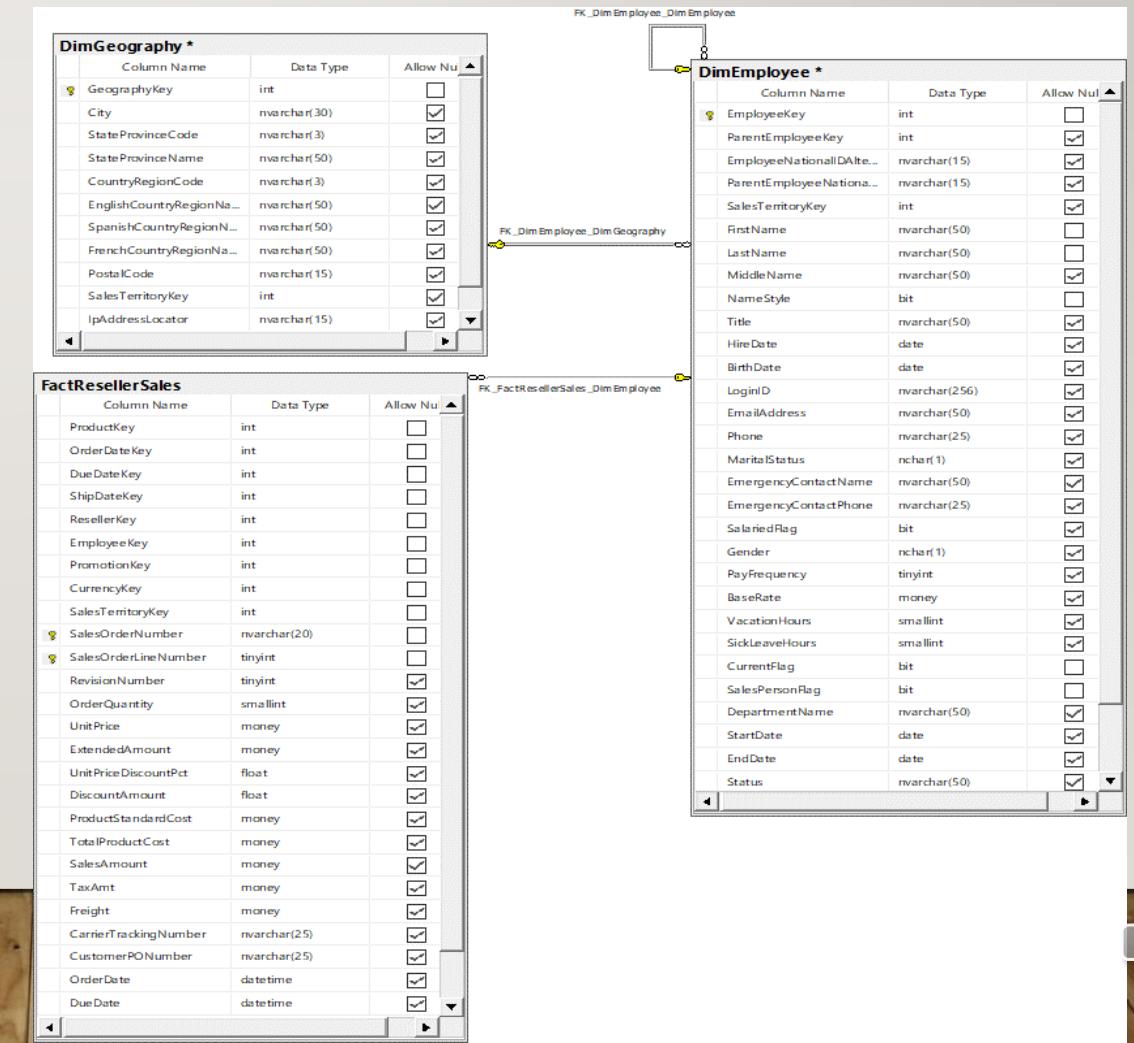
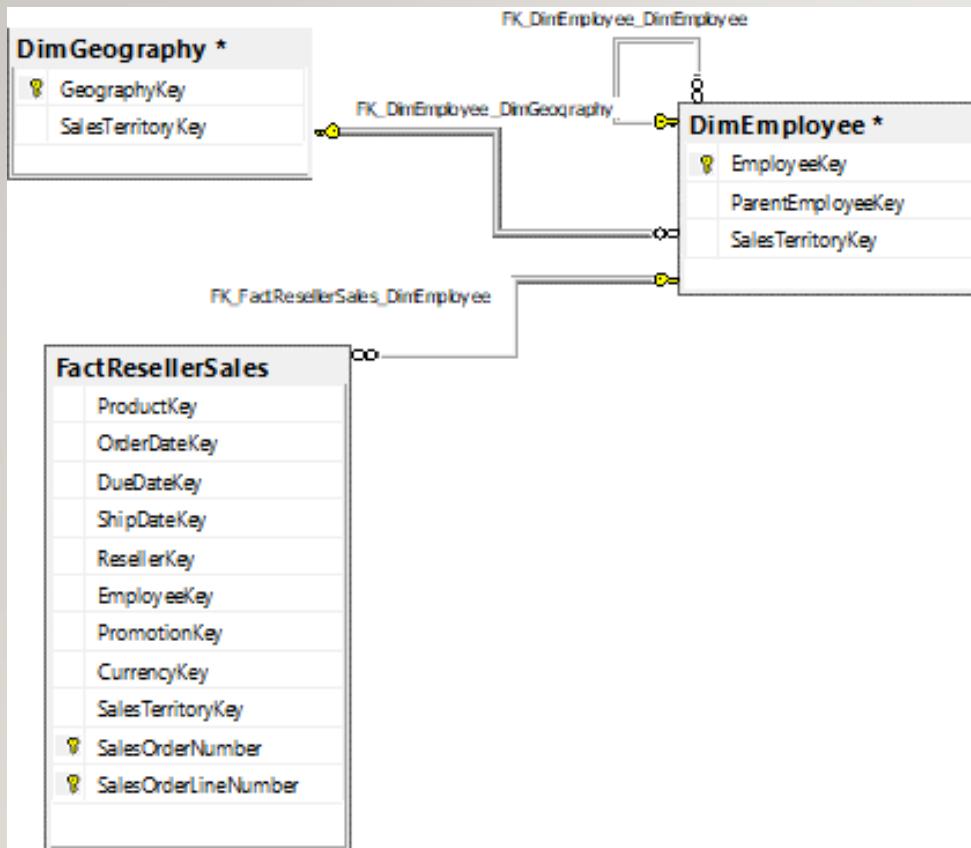
DANNY'S QUERY (COMPLEX) (WORST)

```
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)

The screenshot shows two windows side-by-side. On the left is a 'Results' window from SQL Server Management Studio (SSMS) displaying a table of order data. On the right is a 'JSToolNpp JSON Viewer' window showing the same data as JSON output.

Results Window (SSMS):

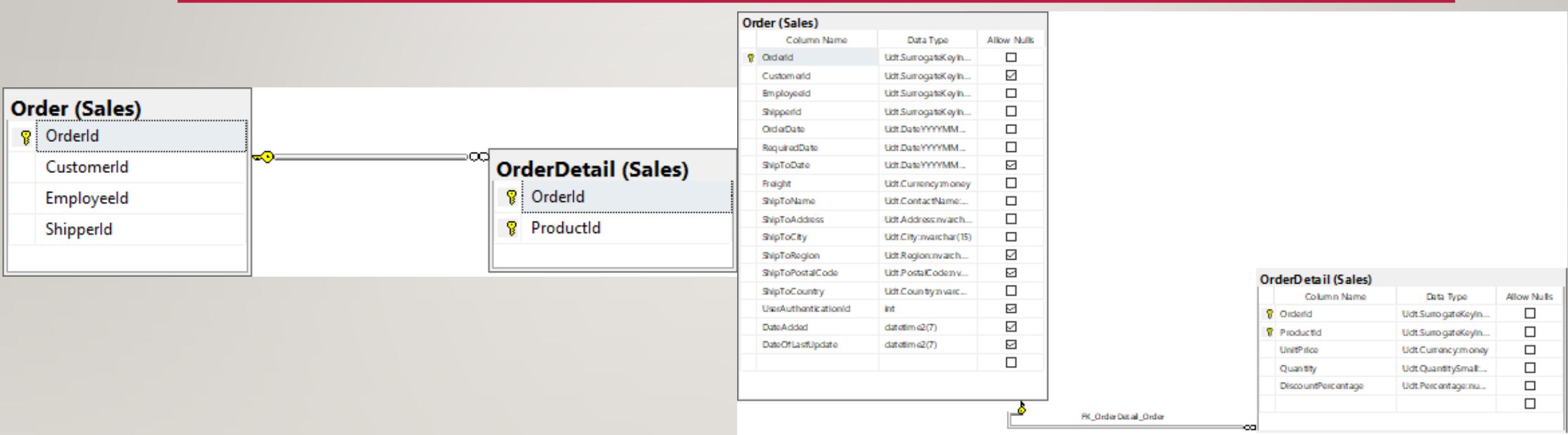
	City	Name	# of orders	OrderDate
1	Ascheim	Valdez Rachel B	178	2013-01-28 00:00:00.000
2	Ascheim	Valdez Rachel B	188	2013-02-28 00:00:00.000
3	Ascheim	Valdez Rachel B	123	2013-03-30 00:00:00.000
4	Ascheim	Valdez Rachel B	169	2013-04-30 00:00:00.000
5	Ascheim	Valdez Rachel B	131	2013-05-30 00:00:00.000
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	2013-10-29 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

JSToolNpp JSON Viewer:

```
1 "Handled Orders by this employee": [
2     {
3         "City": "Ascheim",
4         "Name": "Valdez Rachel B",
5         "# of orders": 178,
6         "OrderDate": "2013-01-28T00:00:00"
7     },
8     {
9         "City": "Ascheim",
10        "Name": "Valdez Rachel B",
11        "# of orders": 188,
12        "OrderDate": "2013-02-28T00:00:00"
13    },
14    {
15        "City": "Ascheim",
16        "Name": "Valdez Rachel B",
17        "# of orders": 123,
18        "OrderDate": "2013-03-30T00:00:00"
19    },
20    {
21        "City": "Ascheim",
22        "Name": "Valdez Rachel B",
23        "# of orders": 169,
24        "OrderDate": "2013-04-30T00:00:00"
25    },
26    {
27        "City": "Ascheim",
28        "Name": "Valdez Rachel B",
29        "# of orders": 131,
30        "OrderDate": "2013-05-30T00:00:00"
31    },
32    {
33        "City": "Ascheim",
34        "Name": "Valdez Rachel B",
35        "# of orders": 105,
36        "OrderDate": "2013-06-30T00:00:00"
37    }
]
```



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

The screenshot displays two windows side-by-side. On the left is a SQL Server Management Studio (SSMS) window showing a T-SQL query and its results. On the right is an IToolNpp JSON Viewer window showing the query's output as JSON.

SQL Query (SSMS):

```
USE Northwind;
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;
```

Query Results (SSMS):

	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

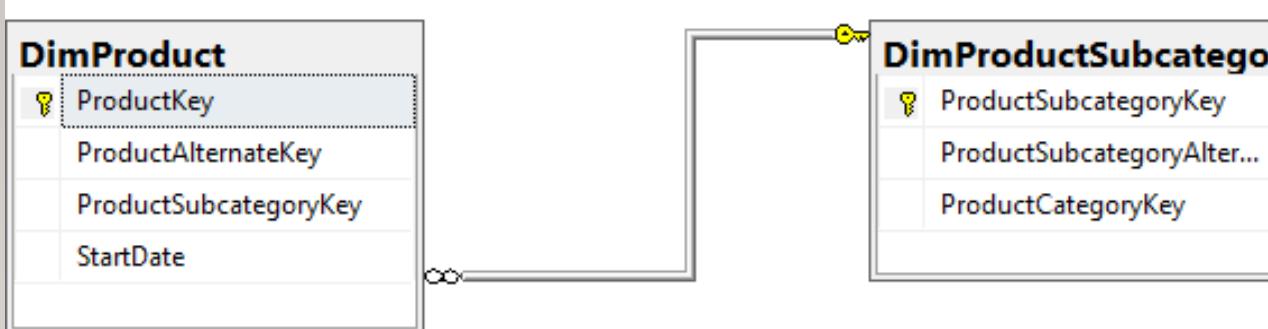
JSON Output (IToolNpp JSON Viewer):

```
15060
15061
15062
15063
15064
15065
15066
15067
15068
15069
15070
15071
15072
15073
15074
15075
15076
15077
15078
15079
15080
15081
15082
15083
15084
15085
15086
15087
15088
15089

"OrderId": 11077,
"CustomerId": 65,
"EmployeeId": 1,
"ShipToCountry": "USA",
"ProductId": 66,
"UnitPrice": 17.0000
}, {
"OrderId": 11077,
"CustomerId": 65,
"EmployeeId": 1,
"ShipToCountry": "USA",
"ProductId": 73,
"UnitPrice": 15.0000
}, {
"OrderId": 11077,
"CustomerId": 65,
"EmployeeId": 1,
"ShipToCountry": "USA",
"ProductId": 75,
"UnitPrice": 7.7500
}, {
"OrderId": 11077,
"CustomerId": 65,
"EmployeeId": 1,
"ShipToCountry": "USA",
"ProductId": 77,
"UnitPrice": 13.0000
}
```



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



DimProductSubcategory			
Column Name	Data Type	Allow Nulls	
ProductSubcategoryKey	Int	<input type="checkbox"/>	
ProductSubcategoryName	Int	<input checked="" type="checkbox"/>	
EnglishProductSubcategory	nvarchar(50)	<input type="checkbox"/>	
SpanishProductSubcategory	nvarchar(50)	<input type="checkbox"/>	
FrenchProductSubcategory	nvarchar(50)	<input type="checkbox"/>	
ProductCategoryKey	Int	<input checked="" type="checkbox"/>	

Dim Product	Column Name	Data Type	Allow Nulls
	ProductKey	int	<input checked="" type="checkbox"/>
	ProductAlternateKey	nvarchar(25)	<input checked="" type="checkbox"/>
	ProductSubcategoryKey	int	<input checked="" type="checkbox"/>
	WeightUnitMeasureCode	nchar(3)	<input checked="" type="checkbox"/>
	SizeUnitMeasureCode	nchar(3)	<input checked="" type="checkbox"/>
	EnglishProductName	nvarchar(50)	<input type="checkbox"/>
	SpanishProductName	nvarchar(50)	<input type="checkbox"/>
	FrenchProductName	nvarchar(50)	<input type="checkbox"/>
	StandardCost	money	<input checked="" type="checkbox"/>
	FinishedGoodsFlag	bit	<input type="checkbox"/>
	Color	nvarchar(15)	<input type="checkbox"/>
	SafetyStockLevel	smallint	<input checked="" type="checkbox"/>
	ReorderPoint	smallint	<input checked="" type="checkbox"/>
	ListPrice	money	<input checked="" type="checkbox"/>
	Size	nvarchar(50)	<input checked="" type="checkbox"/>
	SizeRange	nvarchar(50)	<input checked="" type="checkbox"/>
	Weight	float	<input checked="" type="checkbox"/>
	DaysToManufacture	int	<input checked="" type="checkbox"/>
	ProductLine	nchar(2)	<input checked="" type="checkbox"/>
	DealerPrice	money	<input checked="" type="checkbox"/>
	Class	nchar(2)	<input checked="" type="checkbox"/>
	Style	nchar(2)	<input checked="" type="checkbox"/>
	ModelName	nvarchar(50)	<input checked="" type="checkbox"/>
	LargePhoto	varbinary(MAX)	<input checked="" type="checkbox"/>
	EnglishDescription	nvarchar(400)	<input checked="" type="checkbox"/>
	FrenchDescription	nvarchar(400)	<input checked="" type="checkbox"/>
	ChineseDescription	nvarchar(400)	<input checked="" type="checkbox"/>
	ArabicDescription	nvarchar(400)	<input checked="" type="checkbox"/>
	HindiDescription	nvarchar(400)	<input checked="" type="checkbox"/>
	ThaiDescription	nvarchar(400)	<input checked="" type="checkbox"/>
	GermanDescription	nvarchar(400)	<input checked="" type="checkbox"/>
	JapaneseDescription	nvarchar(400)	<input checked="" type="checkbox"/>
	TurkishDescription	nvarchar(400)	<input checked="" type="checkbox"/>
	StartDate	datetime	<input checked="" type="checkbox"/>
	EndDate	datetime	<input checked="" type="checkbox"/>
	Status	nvarchar(7)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>



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

```
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

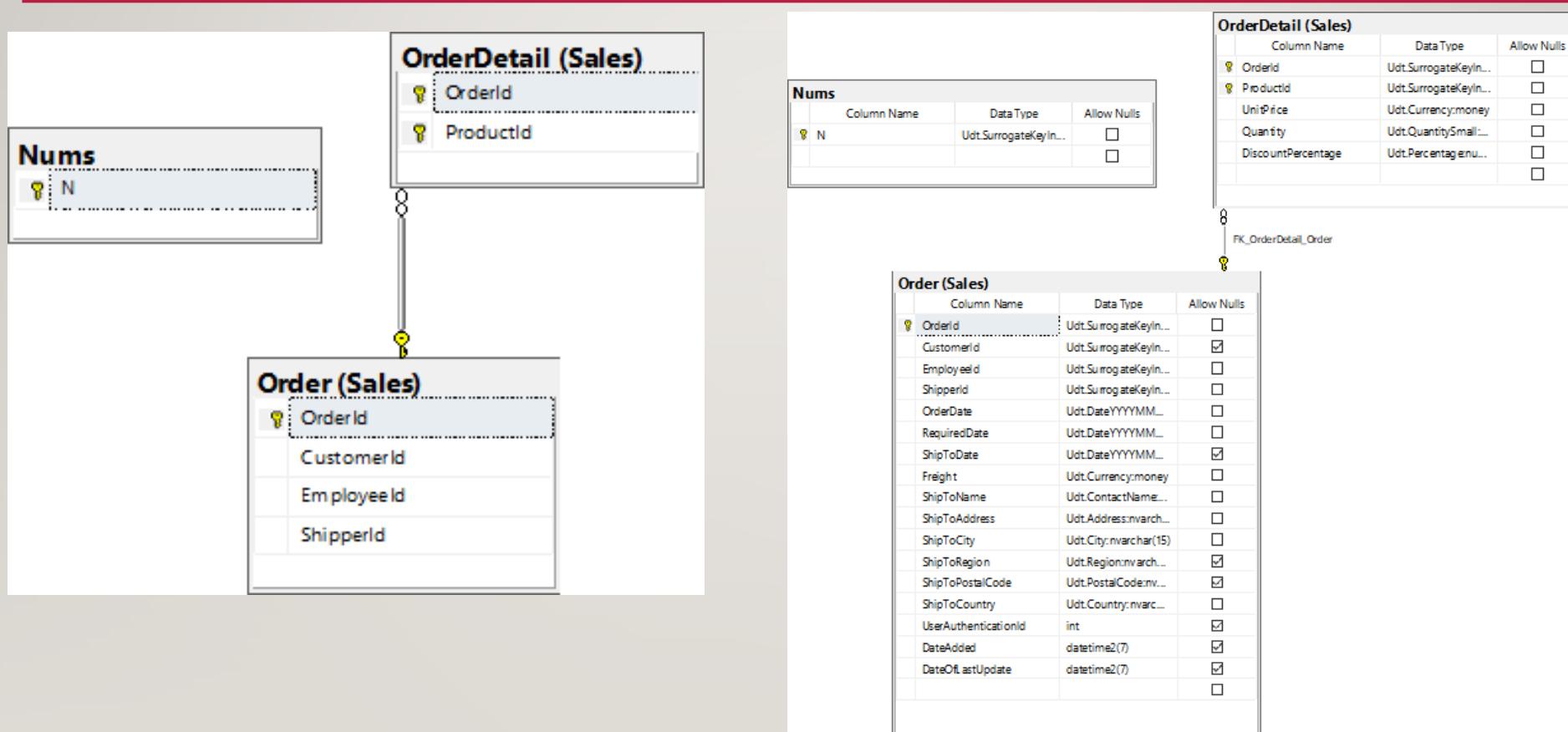
EnglishProductName	SpanishProductName	SpanishProductSubcategoryName
All-Purpose Bike Stand	Sopete multiusos para bicicletas	Sopete para bicicletas
AIVC Logo Cap	Gorra	
Bike Wash - Dissolver	Lavado de bicicletas: disolvente	Limpador
Cable Lock	Cable antirrobo	Candado
Chain	Cadena	Cadena
Classic Vest, L	Camiseta clásica, G	Camiseta
Classic Vest, M	Camiseta clásica, M	Camiseta
Classic Vest, S	Camiseta clásica, P	Camiseta
Fender Set - Mountain	Conjunto de guardabarros: montaña	Guardabarros
Front Brake	Frenos delanteros	Frenos
Front Derailleur	Derivador delantero	Derivador
Full-Finger Gloves, L	Guantes completos, G	Guantes
Full-Finger Gloves, M	Guantes completos, M	Guantes
Full-Finger Gloves, S	Guantes completos, P	Guantes
Half-Finger Gloves, L		Guantes
Half-Finger Gloves, M		Guantes
Half-Finger Gloves, S		Guantes
Headlights - Dual-Beam	Luces: doble haz	Luz
Headlights - Weatherproof	Luces: resistentes al agua	Luz

Refresh Search

[[{"id": 1152, "product": {"EnglishProductName": "All-Purpose Bike Stand", "SpanishProductName": "Sopete multiusos para bicicletas", "SpanishProductSubcategoryName": "Sopete para bicicletas"}, "id": 1153}, {"id": 1154, "product": {"EnglishProductName": "AIVC Logo Cap", "SpanishProductName": "Gorra", "SpanishProductSubcategoryName": null}, "id": 1155}, {"id": 1156, "product": {"EnglishProductName": "Bike Wash - Dissolver", "SpanishProductName": "Lavado de bicicletas: disolvente", "SpanishProductSubcategoryName": "Limpador"}, "id": 1157}, {"id": 1158, "product": {"EnglishProductName": "Cable Lock", "SpanishProductName": "Cable antirrobo", "SpanishProductSubcategoryName": "Candado"}, "id": 1159}, {"id": 1159, "product": {"EnglishProductName": "Chain", "SpanishProductName": "Cadena", "SpanishProductSubcategoryName": "Cadena"}, "id": 1160}, {"id": 1160, "product": {"EnglishProductName": "Classic Vest, L", "SpanishProductName": "Camiseta cl\u00e1sica, G", "SpanishProductSubcategoryName": "Camiseta"}, "id": 1161}, {"id": 1161, "product": {"EnglishProductName": "Classic Vest, M", "SpanishProductName": "Camiseta cl\u00e1sica, M", "SpanishProductSubcategoryName": "Camiseta"}, "id": 1162}, {"id": 1162, "product": {"EnglishProductName": "Classic Vest, S", "SpanishProductName": "Camiseta cl\u00e1sica, P", "SpanishProductSubcategoryName": "Camiseta"}, "id": 1163}, {"id": 1163, "product": {"EnglishProductName": "Fender Set - Mountain", "SpanishProductName": "Conjunto de guardabarros: montaña", "SpanishProductSubcategoryName": "Guardabarros"}, "id": 1164}, {"id": 1164, "product": {"EnglishProductName": "Front Brake", "SpanishProductName": "Frenos delanteros", "SpanishProductSubcategoryName": "Frenos"}, "id": 1165}, {"id": 1165, "product": {"EnglishProductName": "Front Derailleur", "SpanishProductName": "Derivador delantero", "SpanishProductSubcategoryName": "Derivador"}, "id": 1166}, {"id": 1166, "product": {"EnglishProductName": "Full-Finger Gloves, L", "SpanishProductName": "Guantes completos, G", "SpanishProductSubcategoryName": "Guantes"}, "id": 1167}, {"id": 1167, "product": {"EnglishProductName": "Full-Finger Gloves, M", "SpanishProductName": "Guantes completos, M", "SpanishProductSubcategoryName": "Guantes"}, "id": 1168}, {"id": 1168, "product": {"EnglishProductName": "Full-Finger Gloves, S", "SpanishProductName": "Guantes completos, P", "SpanishProductSubcategoryName": "Guantes"}, "id": 1169}, {"id": 1169, "product": {"EnglishProductName": "Half-Finger Gloves, L", "SpanishProductName": null, "SpanishProductSubcategoryName": "Guantes"}, "id": 1170}, {"id": 1170, "product": {"EnglishProductName": "Half-Finger Gloves, M", "SpanishProductName": null, "SpanishProductSubcategoryName": "Guantes"}, "id": 1171}, {"id": 1171, "product": {"EnglishProductName": "Half-Finger Gloves, S", "SpanishProductName": null, "SpanishProductSubcategoryName": "Guantes"}, "id": 1172}, {"id": 1172, "product": {"EnglishProductName": "Headlights - Dual-Beam", "SpanishProductName": "Luces: doble haz", "SpanishProductSubcategoryName": "Luz"}, "id": 1173}, {"id": 1173, "product": {"EnglishProductName": "Headlights - Weatherproof", "SpanishProductName": "Luces: resistentes al agua", "SpanishProductSubcategoryName": "Luz"}, "id": 1174}, {"id": 1174, "product": {"EnglishProductName": "Water Bottle - 30 oz.", "SpanishProductName": "", "SpanishProductSubcategoryName": "Portabotellas y botella"}, "id": 1175}, {"id": 1175, "product": {"EnglishProductName": "Women's Mountain Shorts, L", "SpanishProductName": "", "SpanishProductSubcategoryName": "Pantalones cortos"}, "id": 1176}, {"id": 1176, "product": {"EnglishProductName": "Women's Mountain Shorts, M", "SpanishProductName": "", "SpanishProductSubcategoryName": "Pantalones cortos"}, "id": 1177}, {"id": 1177, "product": {"EnglishProductName": "Women's Mountain Shorts, S", "SpanishProductName": "", "SpanishProductSubcategoryName": "Pantalones cortos"}, "id": 1178}, {"id": 1178, "product": {"EnglishProductName": "Women's Tights, L", "SpanishProductName": "Mallas para mujer, G", "SpanishProductSubcategoryName": "Mallas"}, "id": 1179}, {"id": 1179, "product": {"EnglishProductName": "Women's Tights, M", "SpanishProductName": "Mallas para mujer, M", "SpanishProductSubcategoryName": "Mallas"}, "id": 1180}, {"id": 1180, "product": {"EnglishProductName": "Women's Tights, S", "SpanishProductName": "Mallas para mujer, P", "SpanishProductSubcategoryName": "Mallas"}, "id": 1181}, {"id": 1181, "product": {"EnglishProductName": null, "SpanishProductName": null, "SpanishProductSubcategoryName": null}, "id": 1182}, {"id": 1182, "product": {"EnglishProductName": null, "SpanishProductName": null, "SpanishProductSubcategoryName": null}, "id": 1183}, {"id": 1183, "product": {"EnglishProductName": null, "SpanishProductName": null, "SpanishProductSubcategoryName": null}, "id": 1184}]]



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;

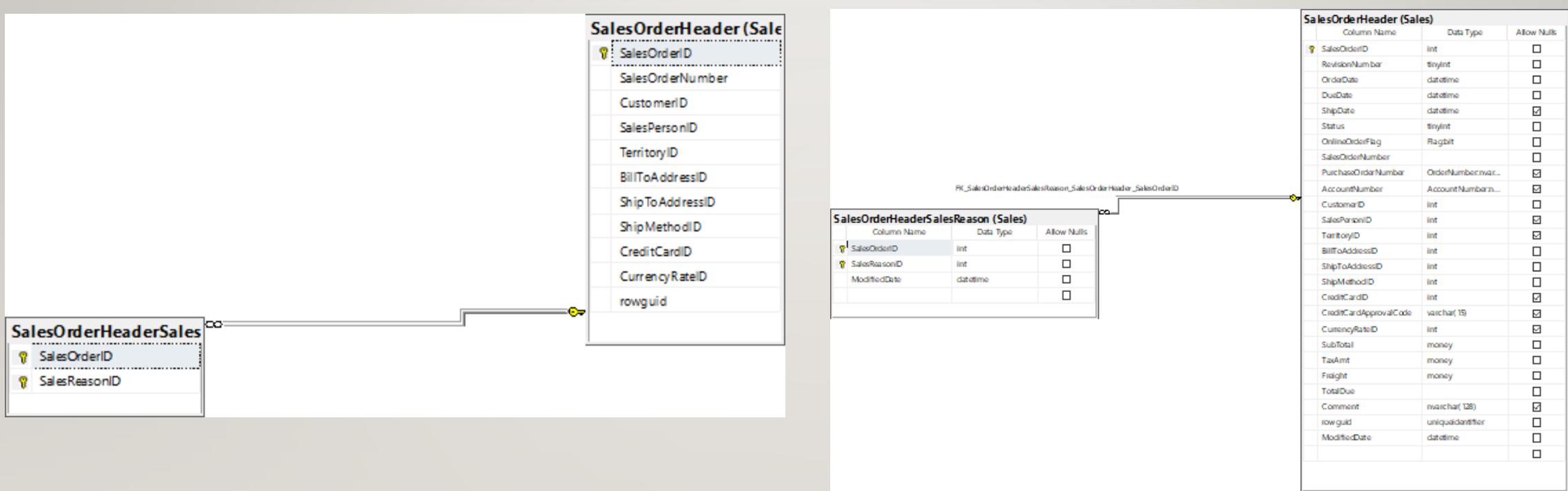
    SELECT @year = @numberOfMonths / 12;

    SELECT @month = (@numberOfMonths % 12);

    SELECT ProductId, ShipToDate, NewShipToDate
    FROM Products
    ORDER BY ProductId
```



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

The screenshot shows a SQL Server Management Studio (SSMS) session with two panes. The left pane displays a T-SQL query and its results. The right pane shows the JSON output of the query.

Query (Left Pane):

```
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;
```

Results (Left Pane):

	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

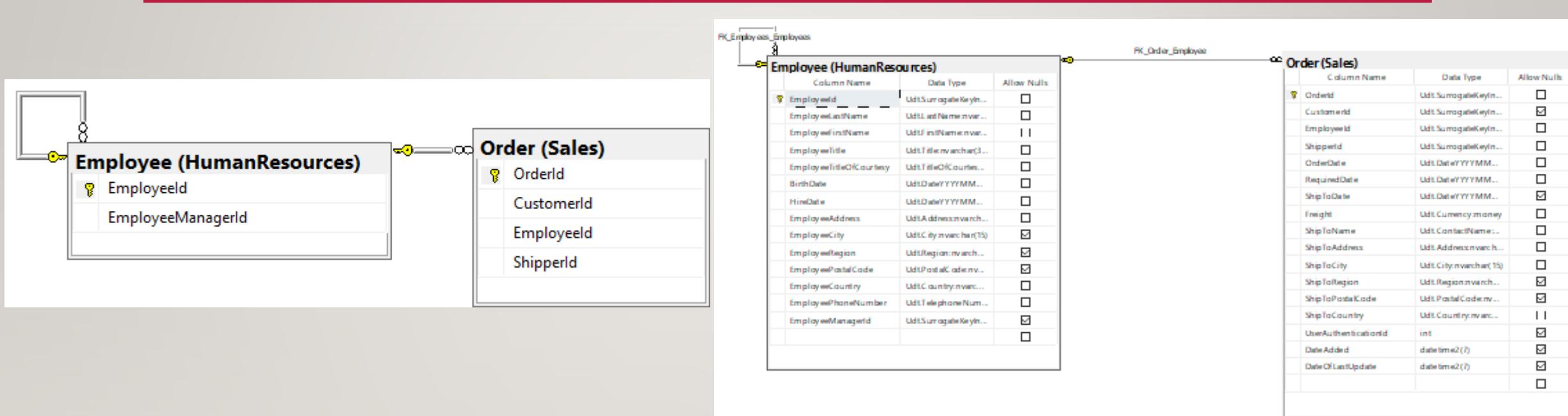
JSON Output (Right Pane):

```
Refresh | Search
ROOT
  \ makeupup: [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
  ] 615192
  ] 615193
  ] 615194
  ] 615195
  ] 615196
  ] 615197
  ] 615198
  ] 615199
  ] 615200
  ] 615201
  ] 615202
  ] 615203
  ] 615204
  ] 615205
  ] 615206
  ] 615207
  ] 615208
  ] 615209
  ] 615210
  ] 615211
  ] 615212
  ] 615213
  ] 615214
  ] 615215
  ] 615216
  ] 615217
  ] 615218
  ] 615219
  ] 615220 } }

  "SalesOrderID": 74887,
  "CustomerID": 11711,
  "AccountNumber": "10-4030-011711"
}, {
  "SalesOrderID": 74908,
  "CustomerID": 11013,
  "AccountNumber": "10-4030-011013"
}, {
  "SalesOrderID": 74909,
  "CustomerID": 11501,
  "AccountNumber": "10-4030-011501"
}, {
  "SalesOrderID": 74880,
  "CustomerID": 11188,
  "AccountNumber": "10-4030-011188"
}, {
  "SalesOrderID": 74887,
  "CustomerID": 11711,
  "AccountNumber": "10-4030-011711"
}, {
  "SalesOrderID": 74908,
  "CustomerID": 11013,
  "AccountNumber": "10-4030-011013"
}, {
  "SalesOrderID": 74909,
  "CustomerID": 11501,
  "AccountNumber": "10-4030-011501"
}
```



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

```
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;
```

Results

EmployeeCity	NumberOfEmployees
Kirkland	127
London	224
Redmond	156
Seattle	227
Tacoma	96

Refresh | Search

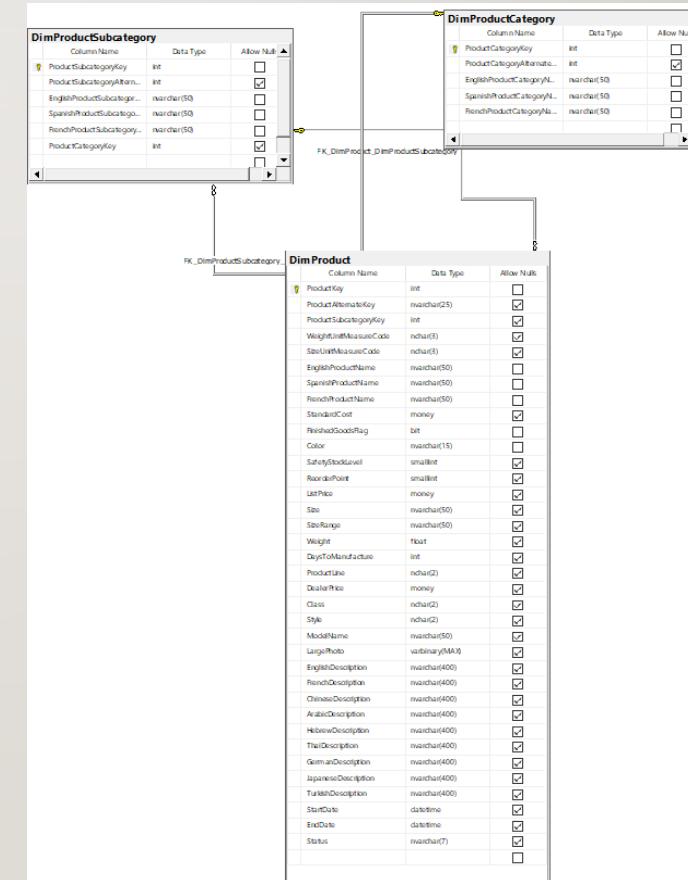
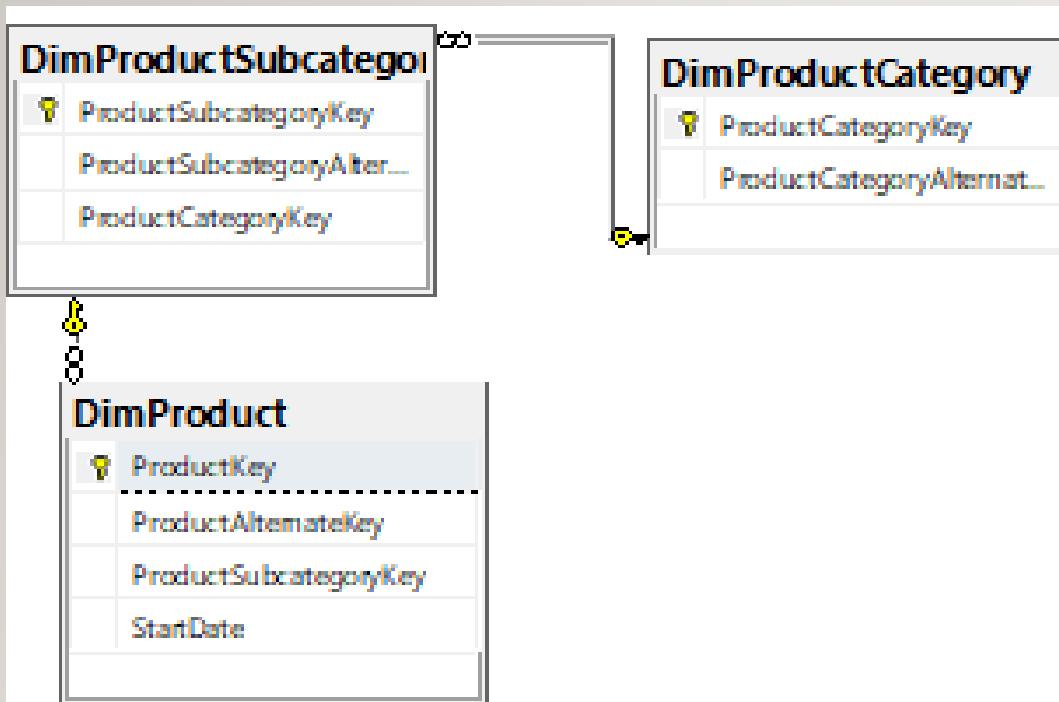
ROOT

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

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
"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	EnglishProductName	ASC



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

```
USE AdventureWorksDW2017;
GO

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

Results Messages

FullEnglishName	FullSpanishName
All-Purpose Bike Stand Bike Stands Accessories	Soporte mutusas para bicicletas Soporte para bicicletas
All Purpose Bike Stand Bike Stands Accessories	Soporte mutusas para bicicletas Soporte para bicicletas
All-Wheel Caps Clothing	Gorra Prenda
Bike Wash - Dissolver Cleaners Accessories	Lavado de bicicletas: disolviente Limpador Accesorio
Cable Lock Locks Accessories	Cable antirrobo Candado Accesorio
Chain Chians Components	Cadena Cadena Componente
Classic Vest, L Vests Clothing	Camiseta clásica, G Camiseta Prenda
Classic Vest, M Vests Clothing	Camiseta clásica, M Camiseta Prenda
Classic Vest, S Vests Clothing	Camiseta clásica, P Camiseta Prenda
Fender Set - Mountain Fenders Accessories	Conjunto de guardabarros: montaña Guardabarros Accesorios
Front Brakes Brakes Components	Freno delantero Frenos Componente
Front Derailleur Derailleurs Components	Desviador delantero Desviador Componente
Full-Finger Gloves, L Gloves Clothing	Guantes completos, G Guantes Prenda
Full-Finger Gloves, M Gloves Clothing	Guantes completos, M Guantes Prenda
Full-Finger Gloves, S Gloves Clothing	Guantes completos, P Guantes Prenda
Half-Finger Gloves, L Gloves Clothing	Guantes Prenda
Half-Finger Gloves, M Gloves Clothing	Guantes Prenda
Half-Finger Gloves, S Gloves Clothing	Guantes Prenda
Headlights - Dual-Beam Lights Accessories	Luces: doble haz Luz Accesorio
Headlights - Weatherproof Lights Accessories	Luces: resistentes al agua Luz Accesorio

Refresh Search

ObjectID	Object	FullEnglishName	FullSpanishName
857	Object	"FullEnglishName": "Touring-3000 Yellow, 58 Touring Bikes Bikes", "FullSpanishName": "Paseo: 3000, amarilla, 58 Bicicleta de paseo Bicic"	
858	Object	"FullEnglishName": "Touring-3000 Yellow, 62 Touring Bikes Bikes", "FullSpanishName": "Paseo: 3000, amarilla, 62 Bicicleta de paseo Bicic"	
859	Object	"FullEnglishName": "Touring-Panniers, Large Panniers Accessories", "FullSpanishName": "Cesta de paseo, grande Cesta Accesorio"	
860	Object	"FullEnglishName": "Water Bottle - 30 oz. Bottles and Cages Accessorie", "FullSpanishName": "Portabotellas y botella Accesorio"	
861	Object	"FullEnglishName": "Women's Mountain Shorts, L Shorts Clothing", "FullSpanishName": "Pantalones cortos Prenda"	
862	Object	"FullEnglishName": "Women's Mountain Shorts, M Shorts Clothing", "FullSpanishName": "Pantalones cortos Prenda"	
863	Object	"FullEnglishName": "Women's Mountain Shorts, S Shorts Clothing", "FullSpanishName": "Pantalones cortos Prenda"	
864	Object	"FullEnglishName": "Women's Tights, L Tights Clothing", "FullSpanishName": "Mallas para mujer, G Mallas Prenda"	
865	Object	"FullEnglishName": "Women's Tights, M Tights Clothing", "FullSpanishName": "Mallas para mujer, M Mallas Prenda"	
866	Object	"FullEnglishName": "Women's Tights, S Tights Clothing", "FullSpanishName": "Mallas para mujer, P Mallas Prenda"	
867	Object		
868	Object		
869	Object		
870	Object		
871	Object		
872	Object		
873	Object		
874	Object		
875	Object		
876	Object		
877	Object		
878	Object		
879	Object		
880	Object		
881	Object		
882	Object		
883	Object		
884	Object		
885	Object		
886	Object		
887	Object		
888	Object		
889	Object		



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

Preposition with databases and tables:

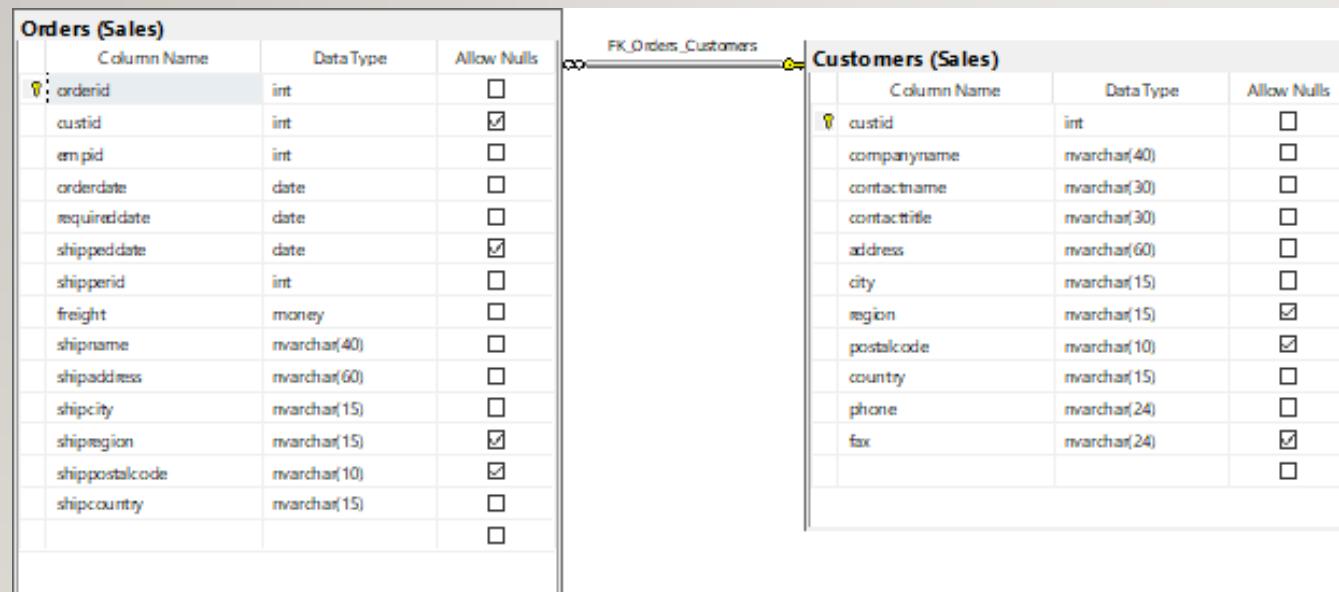
Return all customer ids, their country, and the date they ordered.

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

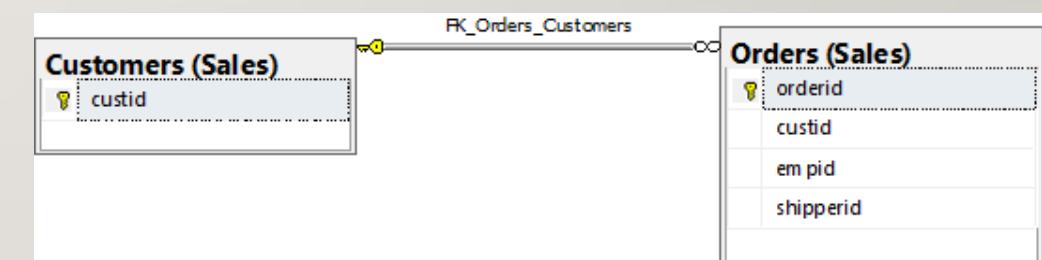


HARJIT'S WORST QUERY: KEYS (1/3)

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

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

Query executed successfully.

Sample JSON output

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

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

"custid": [
    "custid": 1,
    "country": "Germany",
    "orderdate": "2015-08-25"
],
{
    "custid": 1,
    "country": "Germany",
    "orderdate": "2015-10-03"
},
{
    "custid": 1,
    "country": "Germany",
    "orderdate": "2015-10-13"
},
{
    "custid": 1,
    "country": "Germany",
    "orderdate": "2016-01-15"
},
{
    "custid": 1,
    "country": "Germany",
    "orderdate": "2016-03-16"
},
{
    "custid": 1,
    "country": "Germany",
    "orderdate": "2016-04-09"
},
{
    "custid": 2,
    "country": "Mexico",
    "orderdate": "2016-03-04"
},
{
    "custid": 2,
    "country": "Mexico",
    "orderdate": "2016-11-28"
}

Normal text file
length: 94,461 lines: 3,324 Ln: 3,300 Col: 33 Sel: 0 | 0
```



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

Orders (Sales)	
!	orderid
	custid
	empid
	orderdate
	requireddate
	shippeddate
	shipperid
	freight
	shipname
	shipaddress
	shipcity
	shipregion
	shippostalcode
	shipcountry

Orders (Scratch)	
!	orderid
	shipname
	Timestamp

Key view

Orders (Sales)	
!	orderid
	custid
	empid
	shipperid

Orders (Scratch)	
!	orderid



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

```
Results Messages
orderd 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 7.76 2016-05-06
11 11077 9.00 2016-05-06
12 11077 9.69 2016-05-06
13 11077 10.00 2016-05-06
14 11077 12.00 2016-05-06
15 11077 13.00 2016-05-06
16 11077 15.00 2016-05-06
17 11077 17.00 2016-05-06
18 11077 17.45 2016-05-06
19 11077 18.00 2016-05-06

Query executed successfully.
```

JSON Output

```
[{"id": 654, "orderd": 11074, "unitprice": 17.45, "orderdate": "2016-05-06"}, {"id": 655, "orderd": 11075, "unitprice": 12.0, "orderdate": "2016-05-06"}, {"id": 656, "orderd": 11075, "unitprice": 18.0, "orderdate": "2016-05-06"}, {"id": 657, "orderd": 11075, "unitprice": 19.0, "orderdate": "2016-05-06"}, {"id": 658, "orderd": 11076, "unitprice": 9.2, "orderdate": "2016-05-06"}, {"id": 659, "orderd": 11076, "unitprice": 23.25, "orderdate": "2016-05-06"}, {"id": 660, "orderd": 11076, "unitprice": 25.0, "orderdate": "2016-05-06"}, {"id": 661, "orderd": 11077, "unitprice": 6.0, "orderdate": "2016-05-06"}, {"id": 662, "orderd": 11077, "unitprice": 7.0, "orderdate": "2016-05-06"}, {"id": 663, "orderd": 11077, "unitprice": 7.76, "orderdate": "2016-05-06"}, {"id": 664, "orderd": 11077, "unitprice": 9.0, "orderdate": "2016-05-06"}, {"id": 665, "orderd": 11077, "unitprice": 9.69, "orderdate": "2016-05-06"}, {"id": 666, "orderd": 11077, "unitprice": 10.0, "orderdate": "2016-05-06"}, {"id": 667, "orderd": 11077, "unitprice": 12.0, "orderdate": "2016-05-06"}, {"id": 668, "orderd": 11077, "unitprice": 13.0, "orderdate": "2016-05-06"}, {"id": 669, "orderd": 11077, "unitprice": 15.0, "orderdate": "2016-05-06"}, {"id": 670, "orderd": 11077, "unitprice": 17.0, "orderdate": "2016-05-06"}, {"id": 671, "orderd": 11077, "unitprice": 17.45, "orderdate": "2016-05-06"}, {"id": 672, "orderd": 11077, "unitprice": 18.0, "orderdate": "2016-05-06"}, {"id": 673, "orderd": 11078, "unitprice": 13.0, "orderdate": "2016-01-02"}, {"id": 674, "orderd": 11078, "unitprice": 34.8, "orderdate": "2016-01-02"}, {"id": 675, "orderd": 11079, "unitprice": 18.0, "orderdate": "2016-01-01"}, {"id": 676, "orderd": 11080, "unitprice": 7.0, "orderdate": "2016-01-01"}, {"id": 677, "orderd": 11081, "unitprice": 38.0, "orderdate": "2016-01-01"}, {"id": 678, "orderd": 11082, "unitprice": 6.0, "orderdate": "2016-01-01"}, {"id": 679, "orderd": 11083, "unitprice": 14.0, "orderdate": "2016-01-01"}, {"id": 680, "orderd": 11084, "unitprice": 15.0, "orderdate": "2016-01-01"}]
```



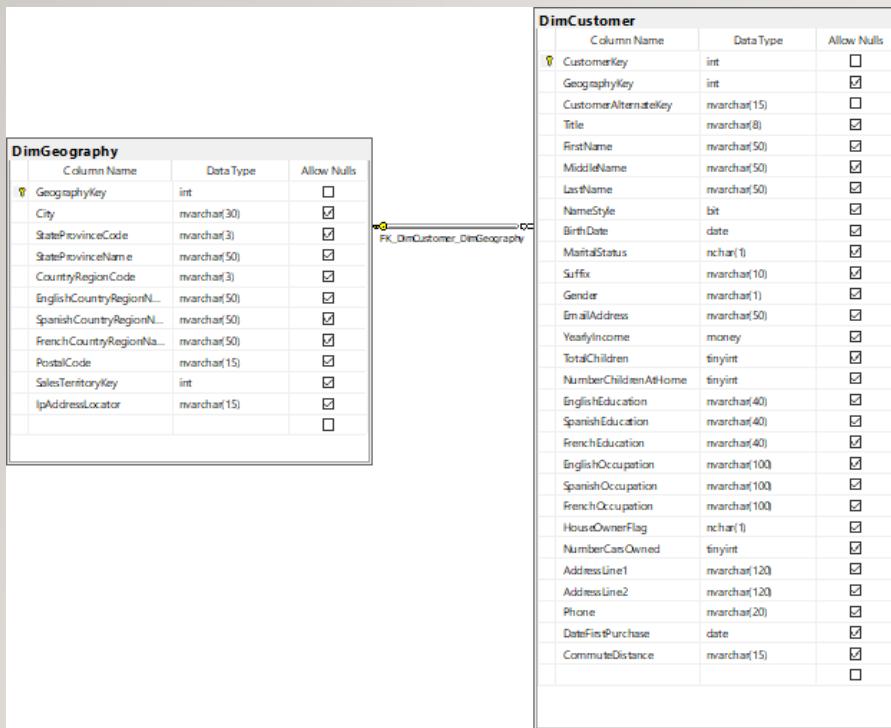
HARJIT'S WORST QUERY: PREPOSITION (3/3)

- 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

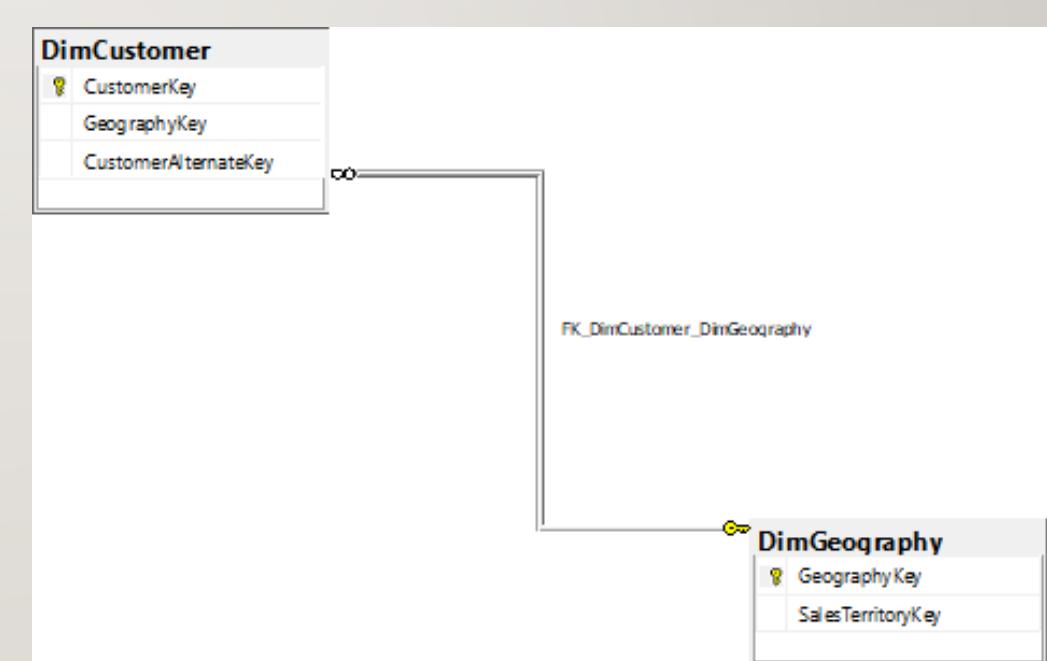


HARJIT'S WORST QUERY: KEYS (3/3)

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

	CustomerKey	FirstName	LastName	City
1	11000	Jon	Ying	Rockhampton
2	11001	Eugene	Huang	Sealed
3	11002	Ruben	Tuna	Hobart
4	11003	Orville	Zu	North Ryde
5	11004	Deborah	Johnson	Wollongong
6	11005	Jake	Pau	East Brisbane
7	11006	Janet	Averett	Melville
8	11007	Marcia	Mehta	Wamersdorf
9	11008	Rob	Wetoff	Bridge
10	11009	Shannon	Carlson	Henry Bay
11	11010	Jacquelyn	Sunee	East Brisbane
12	11011	Gute	Iz	East Brisbane
13	11012	Laura	Walker	Brenton
14	11013	Ian	Jenkins	Lebanon
15	11014	Sydney	Bennett	Redmond
16	11015	Olive	Young	Bulark
17	11016	Ryatt	Hil	Imperial Beach
18	11017	Sharon	Wang	Surbury
19	11018	Dawnie	Re	Bridge

Query executed successfully.

JSON Output

```
[{"CustomerKey": 29478, "FirstName": "Darren", "LastName": "Carlson", "City": "Stoke-on-Trent"}, {"CustomerKey": 29479, "FirstName": "Tommy", "LastName": "Tang", "City": "Versailles"}, {"CustomerKey": 29480, "FirstName": "Nina", "LastName": "Raji", "City": "London"}, {"CustomerKey": 29481, "FirstName": "Ivan", "LastName": "Suzi", "City": "Hof"}, {"CustomerKey": 29482, "FirstName": "Clayton", "LastName": "Zhang", "City": "Saint Ouen"}, {"CustomerKey": 29483, "FirstName": "Jesus", "LastName": "Navarro", "City": "Paris La Defense"}]
```



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

Orders (Sales)	
💡	orderid
	custid
	empid
	orderdate
	requireddate
	shippeddate
	shipperid
	freight
	shipname
	shipaddress
	shipcity
	shipregion
	shippostalcode
	shipcountry

Orders	
💡	orderid
	custid
	empid
	orderdate
	requireddate
	shippeddate
	shipperid
	freight
	shipname
	shipaddress
	shipcity
	shipregion
	shippostalcode

Key View

Orders (Sales)	
💡	orderid
	custid
	empid
	shipperid

Orders	



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

	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

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

```
/ 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 ]
```

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

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



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

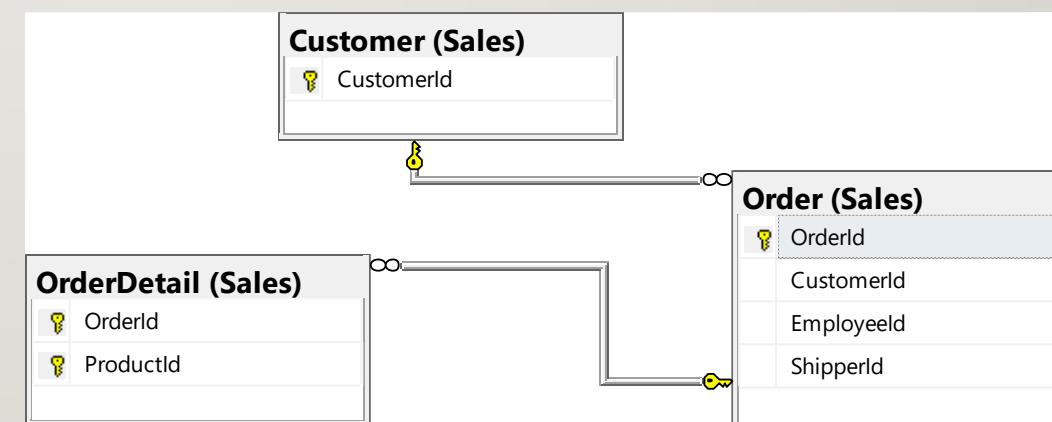


HARJIT'S BEST QUERY: KEYS (2/3)

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

JSON Output

JSToolNpp JSON Viewer

Refresh | Search

[117]: [Object]
[118]: [Object]
[119]: [Object]
[120]: [Object]
[121]: [Object]
[122]: [Object]
[123]: [Object]
[124]: [Object]
[125]: [Object]
[126]: [Object]
[127]: [Object]
[128]: [Object]
[129]: [Object]
[130]: [Object]
[131]: [Object]
[132]: [Object]
[133]: [Object]
[134]: [Object]
[135]: [Object]
[136]: [Object]
[137]: [Object]
[138]: [Object]
[139]: [Object]
[140]: [Object]
[141]: [Object]
[142]: [Object]
[143]: [Object]
[144]: [Object]
[145]: [Object]
[146]: [Object]
[147]: [Object]
[148]: [Object]

new 1 new 2 new 3 new 4 new 5 new 6 new 7 new 8

567 "CustomerId": 89,
568 "ShipToCountry": "USA",
569 "Maximum Quantity": 70
570 }, {
571 "CustomerId": 71,
572 "ShipToCountry": "USA",
573 "Maximum Quantity": 77
574 }, {
575 "CustomerId": 71,
576 "ShipToCountry": "USA",
577 "Maximum Quantity": 80
578 }, {
579 "CustomerId": 71,
580 "ShipToCountry": "USA",
581 "Maximum Quantity": 84
582 }, {
583 "CustomerId": 71,
584 "ShipToCountry": "USA",
585 "Maximum Quantity": 90
586 }, {
587 "CustomerId": 71,
588 "ShipToCountry": "USA",
589 "Maximum Quantity": 100
590 }, {
591 "CustomerId": 71,
592 "ShipToCountry": "USA",
593 "Maximum Quantity": 110
594 }, {
595 "CustomerId": 71,
596 "ShipToCountry": "USA",
597 "Maximum Quantity": 120
598 }
599]
600



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

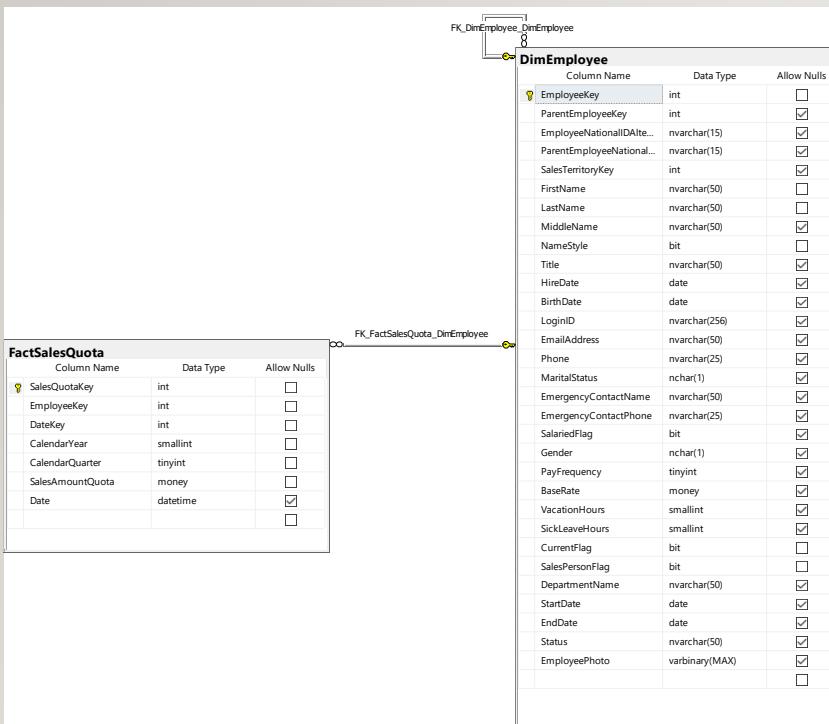
Performs an inner join and returns the employee key, quota, the maximum base pay where the employee was hired after 2011

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

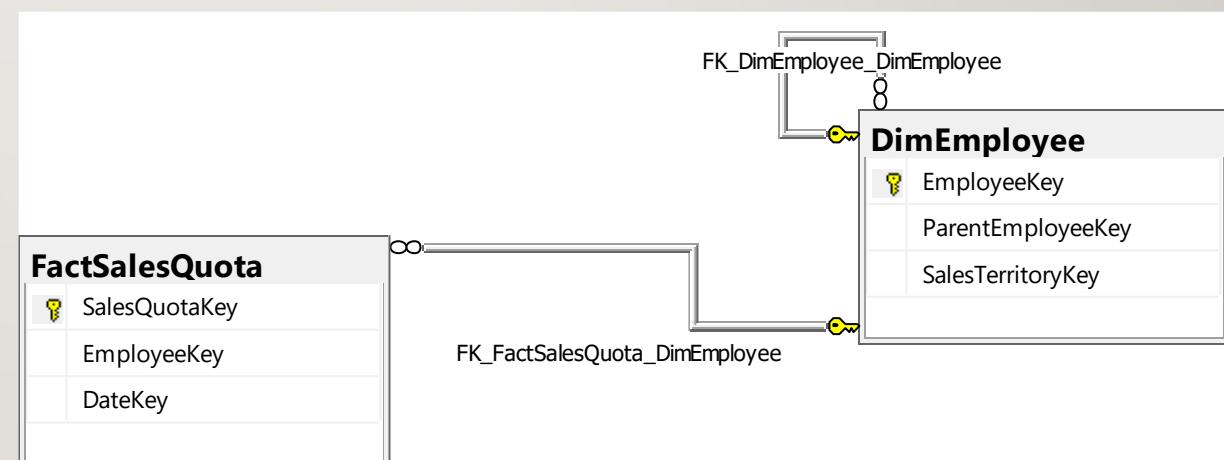


HARJIT'S BEST QUERY: KEYS (3/3)

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

	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	260000.00	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	568000.00	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 (7) | 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]
  ↳ [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]
  ↳ [31]: [Object]
  ↳ [32]: [Object]
  ↳ [33]: [Object]
  ↳ [34]: [Object]
  ↳ [35]: [Object]
  ↳ [36]: [Object]
  ↳ [37]: [Object]
  ↳ [38]: [Object]
  ↳ [39]: [Object]
  ↳ [40]: [Object]
  ↳ [41]: [Object]
  ↳ [42]: [Object]
  ↳ [43]: [Object]
  ↳ [44]: [Object]
  ↳ [45]: [Object]
  ↳ [46]: [Object]
  ↳ [47]: [Object]
  ↳ [48]: [Object]
  ↳ [49]: [Object]
  ↳ [50]: [Object]
  ↳ [51]: [Object]
  ↳ [52]: [Object]
  ↳ [53]: [Object]
  ↳ [54]: [Object]
  ↳ [55]: [Object]
  ↳ [56]: [Object]
  ↳ [57]: [Object]
  ↳ [58]: [Object]
  ↳ [59]: [Object]
  ↳ [60]: [Object]
  ↳ [61]: [Object]
  ↳ [62]: [Object]
  ↳ [63]: [Object]
  ↳ [64]: [Object]
  ↳ [65]: [Object]
  ↳ [66]: [Object]
  ↳ [67]: [Object]
  ↳ [68]: [Object]
  ↳ [69]: [Object]
  ↳ [70]: [Object]
  ↳ [71]: [Object]
  ↳ [72]: [Object]
  ↳ [73]: [Object]
  ↳ [74]: [Object]
  ↳ [75]: [Object]
  ↳ [76]: [Object]
  ↳ [77]: [Object]
  ↳ [78]: [Object]
  ↳ [79]: [Object]
  ↳ [80]: [Object]
  ↳ [81]: [Object]
  ↳ [82]: [Object]
  ↳ [83]: [Object]
  ↳ [84]: [Object]
  ↳ [85]: [Object]
  ↳ [86]: [Object]
  ↳ [87]: [Object]
  ↳ [88]: [Object]
  ↳ [89]: [Object]
  ↳ [90]: [Object]
  ↳ [91]: [Object]
  ↳ [92]: [Object]
  ↳ [93]: [Object]
  ↳ [94]: [Object]
  ↳ [95]: [Object]
  ↳ [96]: [Object]
  ↳ [97]: [Object]
  ↳ [98]: [Object]
  ↳ [99]: [Object]
```



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 DVFM	10883	2016-02-12
2	45	Customer QXPPT	10884	2016-02-12
3	76	Customer SFOWG	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: [Array]

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

"SIMPLE TOP 1": [
 {"CustomerId": 48,
 "CustomerCompanyName": "Customer DVFM",
 "orderid": 10883,
 "orderdate": "2016-02-12"
 },
 {"CustomerId": 45,
 "CustomerCompanyName": "Customer QXPPT",
 "orderid": 10884,
 "orderdate": "2016-02-12"
 },
 {"CustomerId": 76,
 "CustomerCompanyName": "Customer SFOWG",
 "orderid": 10885,
 "orderdate": "2016-02-12"
 },
 {"CustomerId": 34,
 "CustomerCompanyName": "Customer IBVRG",
 "orderid": 10886,
 "orderdate": "2016-02-13"
 },
 {"CustomerId": 29,
 "CustomerCompanyName": "Customer MDLWA",
 "orderid": 10887,
 "orderdate": "2016-02-13"}]



(SIMPLE) TOP #1 DIAGRAM

- Standard View

The image shows two separate tables side-by-side. The left table is titled "Order (Sales) (Read-only)" and the right table is titled "Customer (Sales) (Read-only)". Both tables have columns for OrderId, CustomerId, EmployeeId, ShipperId, OrderDate, RequiredDate, ShipToDate, Freight, ShipToName, ShipToAddress, ShipToCity, ShipToRegion, ShipToPostalCode, ShipToCountry, UserAuthenticationId, DateAdded, and DateOfLastUpdate. The Order table has a primary key icon next to OrderId, while the Customer table has a primary key icon next to CustomerId.

Column Name	Data Type	Allow Nulls
OrderId	Unknown Type	<input type="checkbox"/>
CustomerId	Unknown Type	<input checked="" type="checkbox"/>
EmployeeId	Unknown Type	<input type="checkbox"/>
ShipperId	Unknown Type	<input type="checkbox"/>
OrderDate	Unknown Type	<input type="checkbox"/>
RequiredDate	Unknown Type	<input type="checkbox"/>
ShipToDate	Unknown Type	<input checked="" type="checkbox"/>
Freight	Unknown Type	<input type="checkbox"/>
ShipToName	Unknown Type	<input type="checkbox"/>
ShipToAddress	Unknown Type	<input type="checkbox"/>
ShipToCity	Unknown Type	<input type="checkbox"/>
ShipToRegion	Unknown Type	<input checked="" type="checkbox"/>
ShipToPostalCode	Unknown Type	<input checked="" type="checkbox"/>
ShipToCountry	Unknown Type	<input type="checkbox"/>
UserAuthenticationId	int	<input checked="" type="checkbox"/>
DateAdded	datetime2(7)	<input checked="" type="checkbox"/>
DateOfLastUpdate	datetime2(7)	<input checked="" type="checkbox"/>

Column Name	Data Type	Allow Nulls
CustomerId	Unknown Type	<input type="checkbox"/>
CustomerCompanyName	Unknown Type	<input type="checkbox"/>
CustomerContactName	Unknown Type	<input type="checkbox"/>
CustomerContactTitle	Unknown Type	<input type="checkbox"/>
CustomerAddress	Unknown Type	<input type="checkbox"/>
CustomerCity	Unknown Type	<input type="checkbox"/>
CustomerRegion	Unknown Type	<input checked="" type="checkbox"/>
CustomerPostalCode	Unknown Type	<input checked="" type="checkbox"/>
CustomerCountry	Unknown Type	<input type="checkbox"/>
CustomerPhoneNumber	Unknown Type	<input type="checkbox"/>
CustomerFaxNumber	Unknown Type	<input checked="" type="checkbox"/>

Key View

The image shows two simplified tables. The left table is titled "Order (Sales) (Read-only)" and the right table is titled "Customer (Sales) (Read-only)". Both tables only contain the primary key columns: OrderId and CustomerId, respectively. Primary key icons are present next to both column names in each table.

OrderId
CustomerId
EmployeeId
ShipperId

CustomerId



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	Mt. Mountain Pedal	38018.325800	2	Jill	Maybe it's just because I'm new to mountain bik...
2	255	937	Mt. 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...

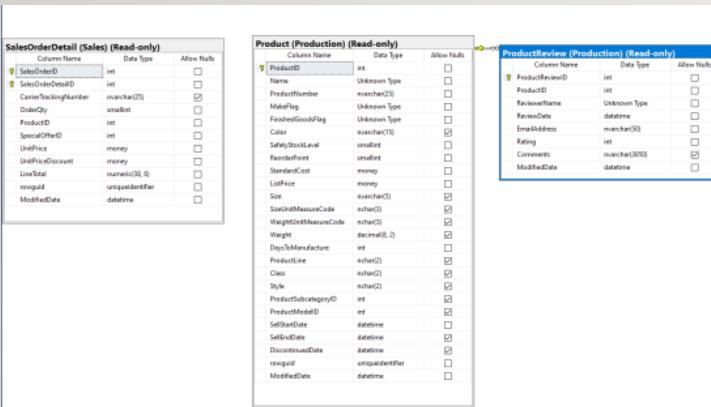
- WITH JSON:

```
ROOT
  MediumQuery3: [
    {
      "Count": 255,
      "ProductID": 937,
      "Name": "Mt. Mountain Pedal",
      "TotalSale": 38018.325800,
      "Rating": 2,
      "ReviewerName": "Jill",
      "Comments": "Maybe it's just because I'm new to mountain biking, but I had a terrible time getting it to work at first. Once I figured out how to assemble it, though, it's been great! The only downside is that it's a bit heavy, but overall it's a solid purchase."
    },
    {
      "Count": 255,
      "ProductID": 937,
      "Name": "Mt. Mountain Pedal",
      "TotalSale": 38018.325800,
      "Rating": 4,
      "ReviewerName": "David",
      "Comments": "A little on the heavy side, but overall the entry/exit is easy in all conditions. The build quality is good, but I would have preferred a lighter weight option for my needs." 
    },
    {
      "Count": 687,
      "ProductID": 798,
      "Name": "Road-550-W Yellow, 40",
      "TotalSale": 1071291.781192,
      "Rating": 5,
      "ReviewerName": "Laura Norman",
      "Comments": "The Road-550-W from Adventure Works Cycles is everything it's advertised to be! It's fast, reliable, and comfortable. I highly recommend this bike for anyone looking for a great ride." 
    },
    {
      "Count": 188,
      "ProductID": 709,
      "Name": "Mountain Bike Socks, M",
      "TotalSale": 6060.388200,
      "Rating": 5,
      "ReviewerName": "John Smith",
      "Comments": "I can't believe I'm singing the praises of a pair of socks, but I just came back from a long ride and these socks kept my feet nice and dry the entire time. Highly recommend!" 
    }
  ]
}
```

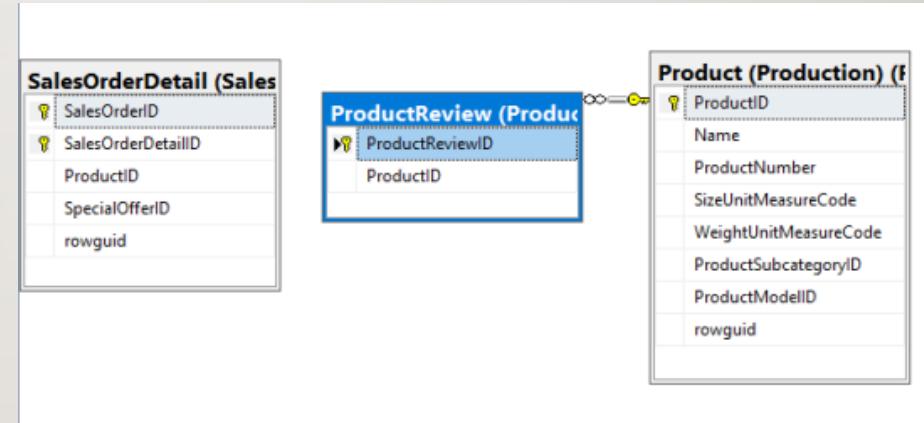


HAIBO'S QUERY (MEDIUM) TOP #2 DIAGRAM

- Standard View:



Key View:



HAIBO'S QUERY (COMPLEX) TOP #3

- return all the orders and the travel from production to customer in 2016
- use Northwinds2020TSQLV6;
- select S.SupplierId, S.SupplierCompanyName, LOCATION1 = 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,Location,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	LOCATION	ProductId	ProductName	OrderId	OrderDate	CustomerCompanyName	LOCATION
1	2	Supplier VHQZD	New Orleans, USA	4	Product KSBRM	11074	2016-05-06	Customer JMIKN
2	2	Supplier VHQZD	New Orleans, USA	5	Product EPEIM	11074	2016-05-06	Customer JMIKN
3	2	Supplier VHQZD	New Orleans, USA	65	Product XYWEK	11074	2016-05-06	Customer JMIKN
4	2	Supplier VHQZD	New Orleans, USA	66	Product LQWGW	11074	2016-05-06	Customer JMIKN
5	2	Supplier VHQZD	New Orleans, USA	4	Product KSBRM	11075	2016-05-06	Customer CCHOT
6	2	Supplier VHQZD	New Orleans, USA	5	Product EPEIM	11075	2016-05-06	Customer CCHOT
7	2	Supplier VHQZD	New Orleans, USA	65	Product XYWEK	11075	2016-05-06	Customer CCHOT
8	2	Supplier VHQZD	New Orleans, USA	66	Product LQWGW	11075	2016-05-06	Customer CCHOT
9	2	Supplier VHQZD	New Orleans, USA	4	Product KSBRM	11076	2016-05-06	Customer KIDOC
10	2	Supplier VHQZD	New Orleans, USA	5	Product EPEIM	11076	2016-05-06	Customer KIDOC
11	2	Supplier VHQZD	New Orleans, USA	65	Product XYWEK	11076	2016-05-06	Customer KIDOC
12	2	Supplier VHQZD	New Orleans, USA	66	Product LQWGW	11076	2016-05-06	Customer KIDOC

- WITH JSON:

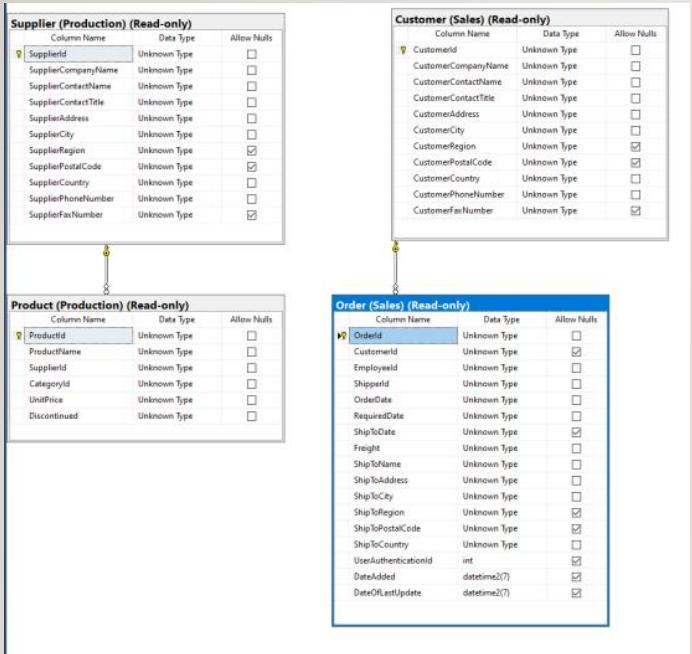
```
complex query 5: [Object]
[0]: [Object]
[1]: [Object]
[2]: [Object]
[3]: [Object]
[4]: [Object]
[5]: [Object]
[6]: [Object]
[7]: [Object]
[8]: [Object]
[9]: [Object]
[10]: [Object]
[11]: [Object]
[12]: [Object]
[13]: [Object]
[14]: [Object]
[15]: [Object]
[16]: [Object]
[17]: [Object]
[18]: [Object]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

```
"complex query 5": [
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 4,
    "ProductName": "Product KSBRM",
    "OrderId": 11074,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer JMIKN",
    "LOCATION": "Kopenhagen, Denmark"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 5,
    "ProductName": "Product EPEIM",
    "OrderId": 11074,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer JMIKN",
    "LOCATION": "Kopenhagen, Denmark"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 65,
    "ProductName": "Product XYWEK",
    "OrderId": 11074,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer JMIKN",
    "LOCATION": "Kopenhagen, Denmark"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 66,
    "ProductName": "Product LQWGW",
    "OrderId": 11074,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer JMIKN",
    "LOCATION": "Kopenhagen, Denmark"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 4,
    "ProductName": "Product KSBRM",
    "OrderId": 11075,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer CCHOT",
    "LOCATION": "Genfve, Switzerland"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 5,
    "ProductName": "Product EPEIM",
    "OrderId": 11075,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer CCHOT",
    "LOCATION": "Genfve, Switzerland"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 65,
    "ProductName": "Product XYWEK",
    "OrderId": 11075,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer CCHOT",
    "LOCATION": "Genfve, Switzerland"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 66,
    "ProductName": "Product LQWGW",
    "OrderId": 11075,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer CCHOT",
    "LOCATION": "Genfve, Switzerland"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 4,
    "ProductName": "Product KSBRM",
    "OrderId": 11076,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer KIDOC",
    "LOCATION": "Marseille, France"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 5,
    "ProductName": "Product EPEIM",
    "OrderId": 11076,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer KIDOC",
    "LOCATION": "Marseille, France"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 65,
    "ProductName": "Product XYWEK",
    "OrderId": 11076,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer KIDOC",
    "LOCATION": "Marseille, France"
  },
  {
    "SupplierId": 2,
    "SupplierCompanyName": "Supplier VHQZD",
    "LOCATION": "New Orleans, USA",
    "ProductId": 66,
    "ProductName": "Product LQWGW",
    "OrderId": 11076,
    "OrderDate": "2016-05-06",
    "CustomerCompanyName": "Customer KIDOC",
    "LOCATION": "Marseille, France"
  }
]
```

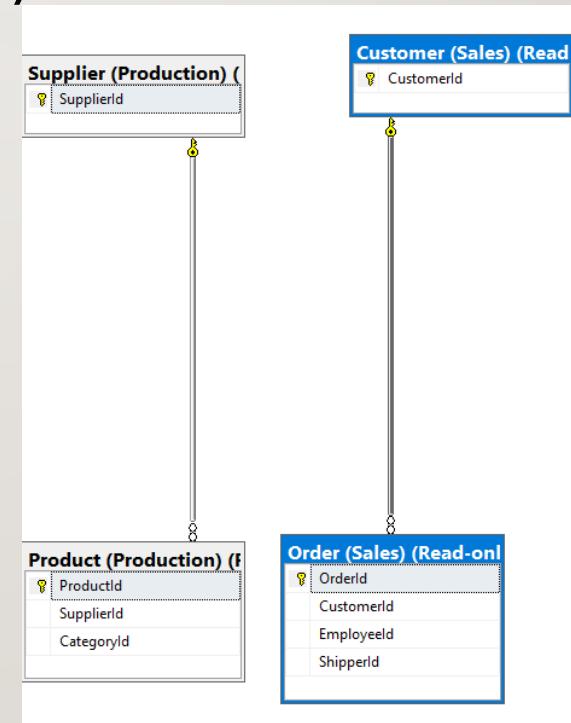


HAIBO'S QUERY (COMPLEX) TOP #3 DIAGRAM

- 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

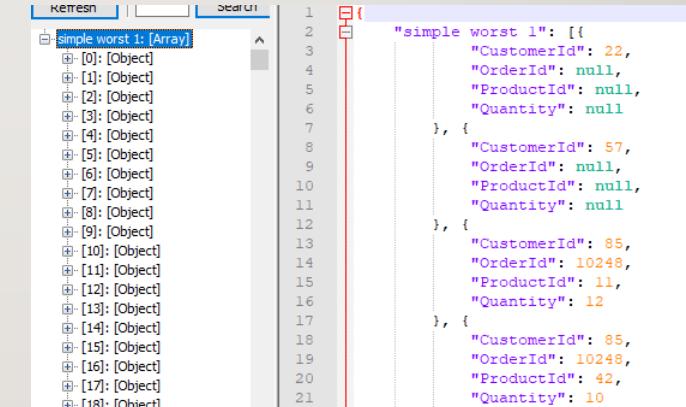


HAIBO'S QUERY (SIMPLE) WORST#1 OUTPUT WITH/WITHOUT JSON

- 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:

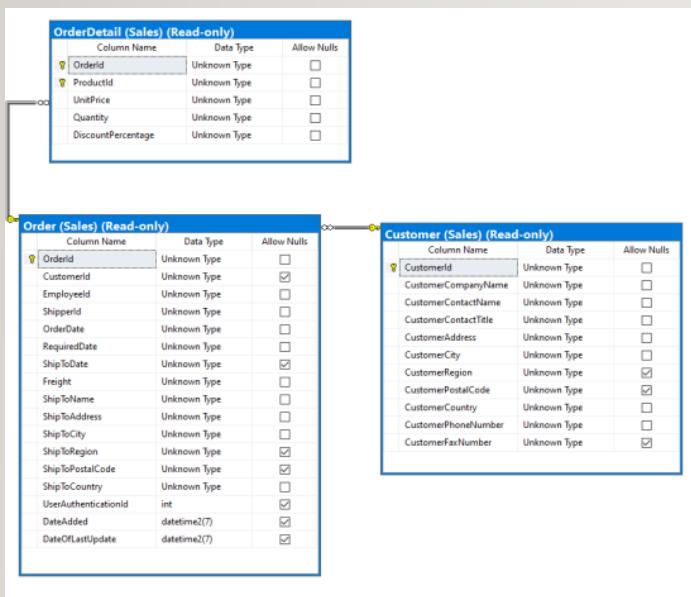


```
simple worst 1: [
  {
    "CustomerId": 22,
    "OrderId": null,
    "ProductId": null,
    "Quantity": null
  },
  {
    "CustomerId": 57,
    "OrderId": null,
    "ProductId": null,
    "Quantity": null
  },
  {
    "CustomerId": 85,
    "OrderId": 10248,
    "ProductId": 11,
    "Quantity": 12
  },
  {
    "CustomerId": 85,
    "OrderId": 10248,
    "ProductId": 42,
    "Quantity": 10
  }
]
```

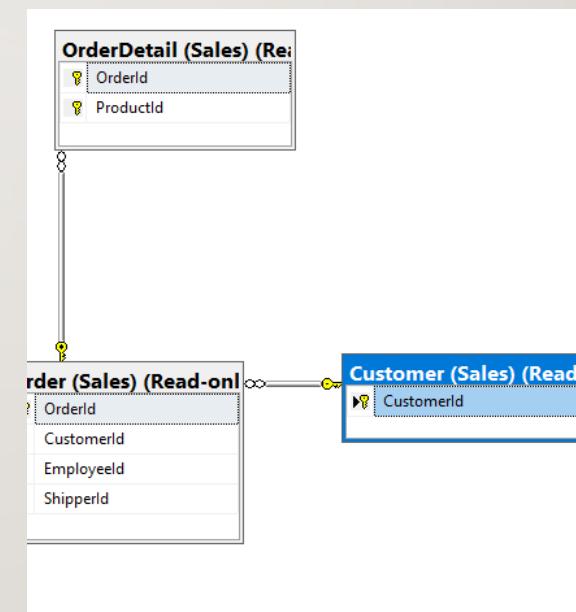


HAIBO'S QUERY (SIMPLE) WORST#1 DIAGRAM

- 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:

A screenshot of a database interface showing a 'Results' tab. The table has two columns: 'CustomerId' and 'CustomerCompanyName'. Row 1 contains CustomerId 22 and CustomerCompanyName Customer DTIDMN. Row 2 contains CustomerId 57 and CustomerCompanyName Customer WVAXS.

	CustomerId	CustomerCompanyName
1	22	Customer DTIDMN
2	57	Customer WVAXS

- WITH JSON:

A screenshot of a JSON editor showing the output of the query. The JSON structure is as follows:

```
ROOT
  - Medium Query 8: [Array]
    - [0]: [Object]
    - [1]: [Object]
```

The array contains two objects, indexed 0 and 1. Object at index 0 has properties CustomerId: 22 and CustomerCompanyName: "Customer DTIDMN". Object at index 1 has properties CustomerId: 57 and CustomerCompanyName: "Customer WVAXS".

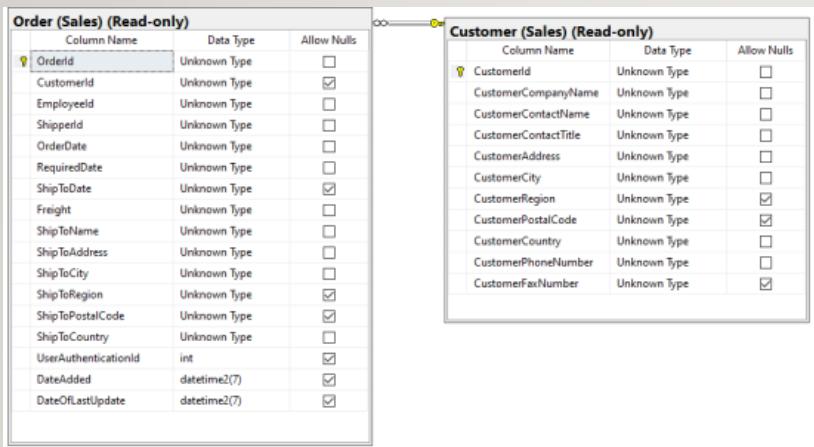
```
1
2
3
4
5
6
7
8
9
10
11
```

```
1
"Medium Query 8": [
  {
    "CustomerId": 22,
    "CustomerCompanyName": "Customer DTIDMN"
  },
  {
    "CustomerId": 57,
    "CustomerCompanyName": "Customer WVAXS"
  }
]
```



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	Quantity
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

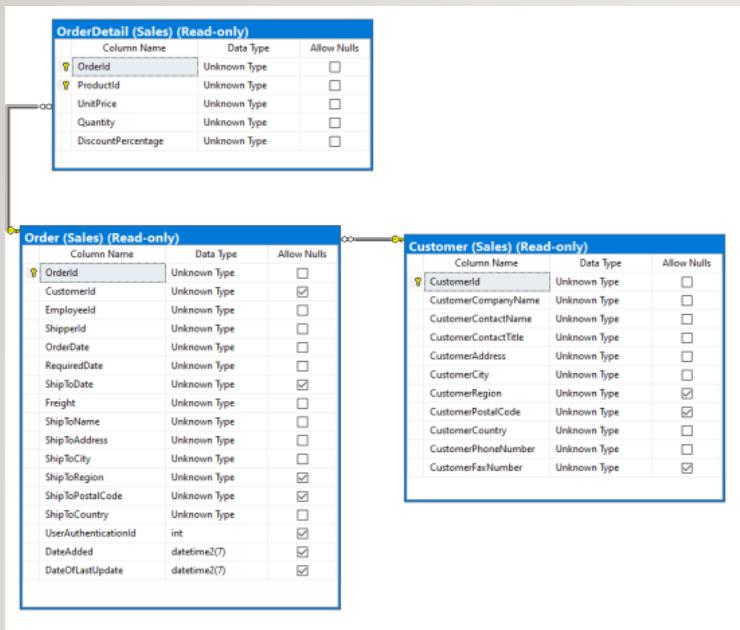
- WITH JSON:

```
complex query 4: [
  {
    "CustomerId": 85,
    "OrderId": 10248,
    "ProductId": 11,
    "Quantity": 12
  },
  {
    "CustomerId": 85,
    "OrderId": 10248,
    "ProductId": 42,
    "Quantity": 10
  },
  {
    "CustomerId": 85,
    "OrderId": 10248,
    "ProductId": 72,
    "Quantity": 5
  },
  {
    "CustomerId": 79,
    "OrderId": 10249,
    "ProductId": 14,
    "Quantity": 9
  },
  {
    "CustomerId": 79,
    "OrderId": 10249,
    "ProductId": 51,
    "Quantity": 40
  },
  {
    "CustomerId": 34,
    "OrderId": 10250,
    "ProductId": 41,
    "Quantity": 10
  },
  {
    "CustomerId": 34,
    "OrderId": 10250,
    "ProductId": 51,
    "Quantity": 35
  },
  {
    "CustomerId": 34,
    "OrderId": 10250,
    "ProductId": 65,
    "Quantity": 15
  },
  {
    "CustomerId": 84,
    "OrderId": 10251,
    "ProductId": 22,
    "Quantity": 6
  },
  {
    "CustomerId": 84,
    "OrderId": 10251,
    "ProductId": 57,
    "Quantity": 15
  },
  {
    "CustomerId": 84,
    "OrderId": 10251,
    "ProductId": 65,
    "Quantity": 20
  },
  {
    "CustomerId": 76,
    "OrderId": 10252,
    "ProductId": 20,
    "Quantity": 40
  },
  {
    "CustomerId": 76,
    "OrderId": 10252,
    "ProductId": 33,
    "Quantity": 25
  },
  {
    "CustomerId": 76,
    "OrderId": 10252,
    "ProductId": 60,
    "Quantity": 40
  },
  {
    "CustomerId": 34,
    "OrderId": 10253,
    "ProductId": 31,
    "Quantity": 20
  }
], {
```

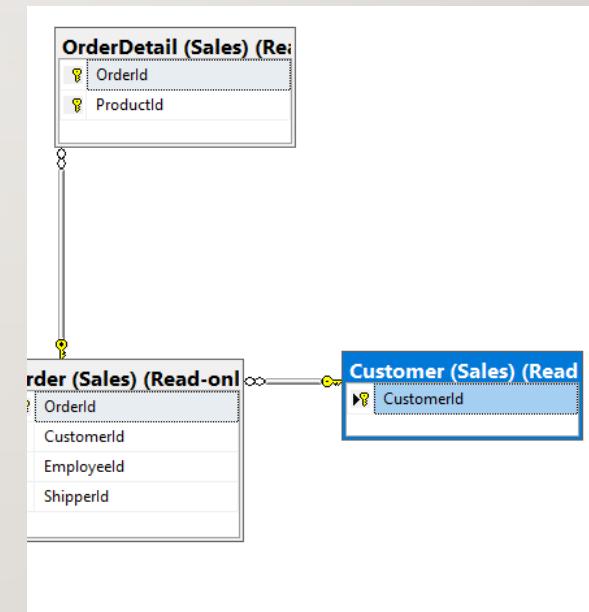


HAIBO'S QUERY (COMPLEX) WORST#3 DIAGRAM

- STANDARD VIEW:



Key View:

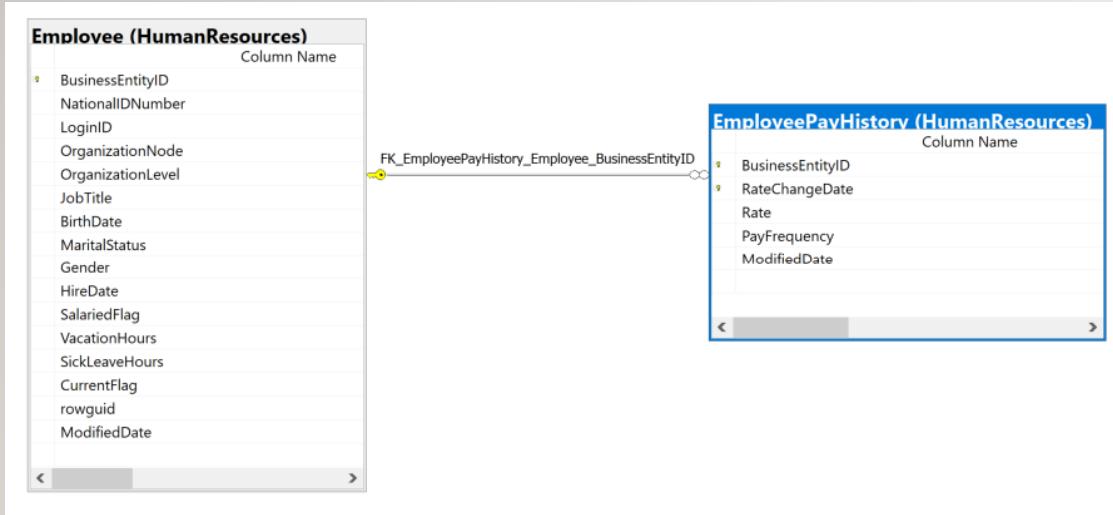


MARLON'S WORST SIMPLE QUERY

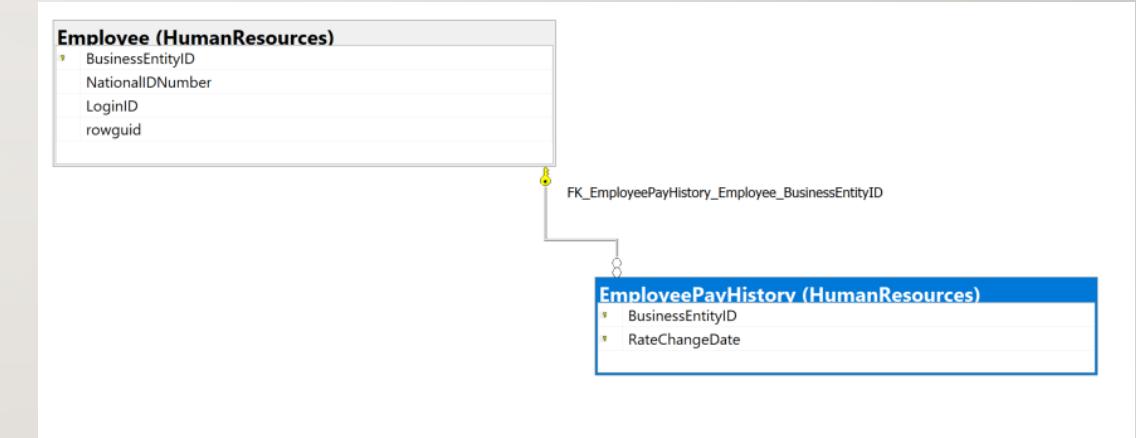
```
--(Simple 3) List all employee titles and their current pay
USE AdventureWorks2017;
SELECT E.BusinessEntityID,
       E.JobTitle,
       P.Rate AS HourlyWage
FROM HumanResources.Employee AS E
     INNER JOIN HumanResources.EmployeePayHistory AS P
              ON E.BusinessEntityID = P.BusinessEntityID
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.Employee PayHistory	HourlyWage	DESC

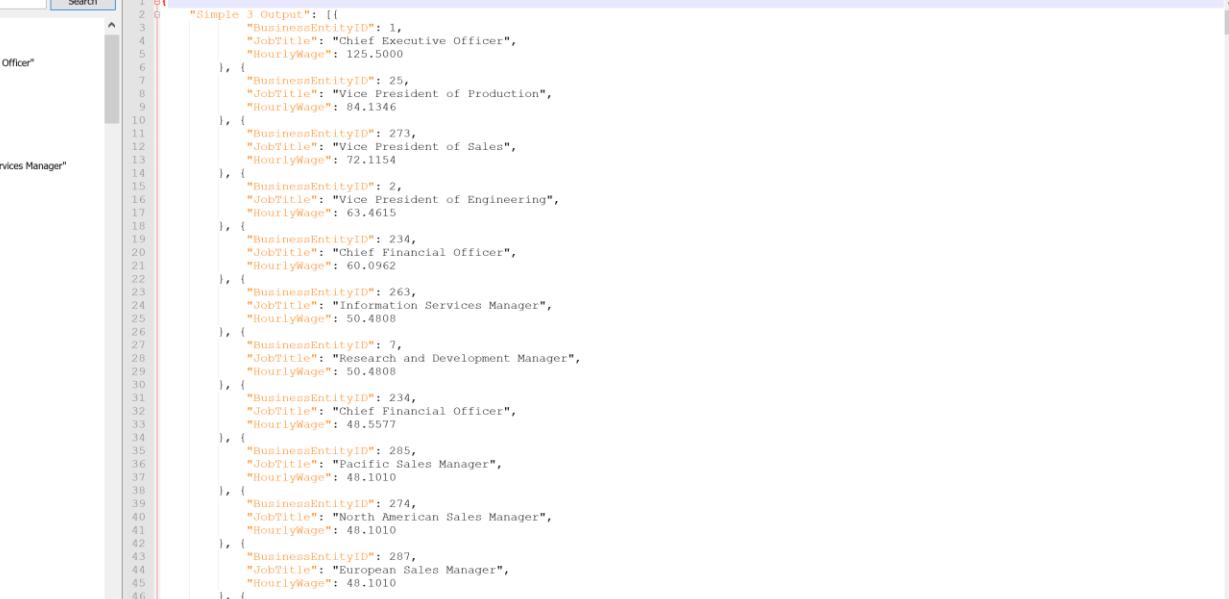


MARLON'S WORST SIMPLE QUERY: OUTPUT

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 Manager	50.4808
7	7	Research and Development Manager	50.4808
8	234	Chief Financial Officer	48.5577
9	285	Pacific Sales Manager	48.101
10	274	North American Sales Manager	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 Manager	42.4808
15	8	Research and Development Manager	40.8654
16	9	Research and Development Manager	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



The screenshot shows the Notepad++ interface with the title bar "new 1 - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language Settings, Tools, Macro, Run, Plugins, and Window. The toolbar has icons for Open, Save, Find, Replace, and others. The status bar at the bottom shows "length: 44,394 lines: 1,269 Ln: 1 Col: 1 Sel: 0 | 0 Windows (CR LF) UTF-8 INS". The main window displays a JSON viewer titled "JSToolPP JSON Viewer" with a "Search" field. The JSON data is a large array of objects representing business entities. Each object contains fields: BusinessEntityID, JobTitle, and HourlyWage. The array has 41 elements, indexed from 0 to 40. The data includes various job titles like "Chief Executive Officer", "Vice President of Production", "Information Services Manager", etc., with corresponding hourly wages ranging from 125.5000 to 43.2692.

```
*new 1 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
JSToolPP JSON Viewer
Refresh Search
new 1
1 [
2   {
3     "Simple_3_Output": [
4       {
5         "BusinessEntityID": 1,
6         "JobTitle": "Chief Executive Officer",
7         "HourlyWage": 125.5000
8       },
9       {
10         "BusinessEntityID": 25,
11         "JobTitle": "Vice President of Production",
12         "HourlyWage": 84.1346
13       },
14       {
15         "BusinessEntityID": 23,
16         "JobTitle": "Vice President of Sales",
17         "HourlyWage": 72.1154
18       },
19       {
20         "BusinessEntityID": 2,
21         "JobTitle": "Vice President of Engineering",
22         "HourlyWage": 63.4615
23       },
24       {
25         "BusinessEntityID": 234,
26         "JobTitle": "Chief Financial Officer",
27         "HourlyWage": 60.0962
28       },
29       {
30         "BusinessEntityID": 263,
31         "JobTitle": "Information Services Manager",
32         "HourlyWage": 50.4808
33       },
34       {
35         "BusinessEntityID": 7,
36         "JobTitle": "Research and Development Manager",
37         "HourlyWage": 50.4808
38       },
39       {
40         "BusinessEntityID": 234,
41         "JobTitle": "Chief Financial Officer",
42         "HourlyWage": 49.5577
43       },
44       {
45         "BusinessEntityID": 285,
46         "JobTitle": "Pacific Sales Manager",
47         "HourlyWage": 48.1010
48       },
49       {
50         "BusinessEntityID": 274,
51         "JobTitle": "North American Sales Manager",
52         "HourlyWage": 48.1010
53       },
54       {
55         "BusinessEntityID": 287,
56         "JobTitle": "European Sales Manager",
57         "HourlyWage": 48.1010
58       },
59       {
60         "BusinessEntityID": 249,
61         "JobTitle": "Finance Manager",
62         "HourlyWage": 43.2692
63     }
64   ]
65 ]
```

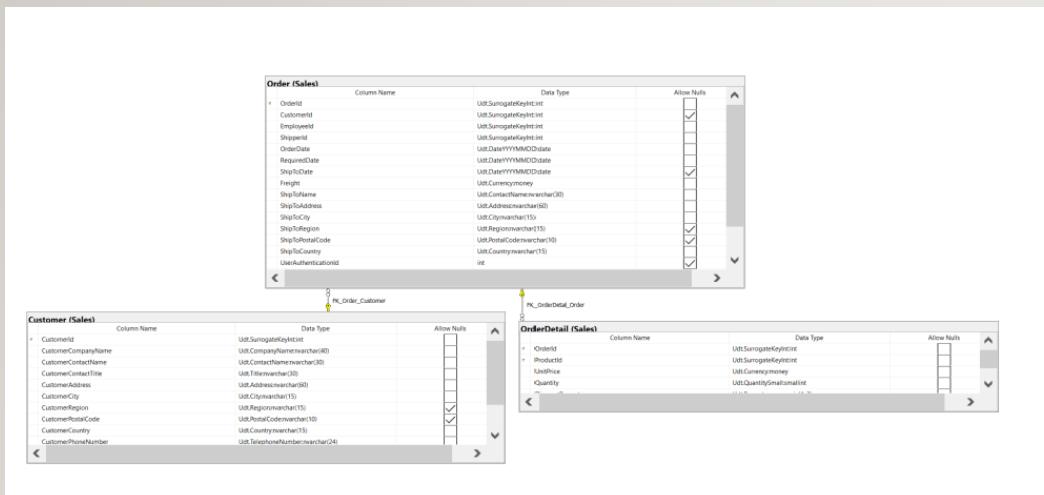
MARLON'S WORST MEDIUM QUERY

```
--(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;
```

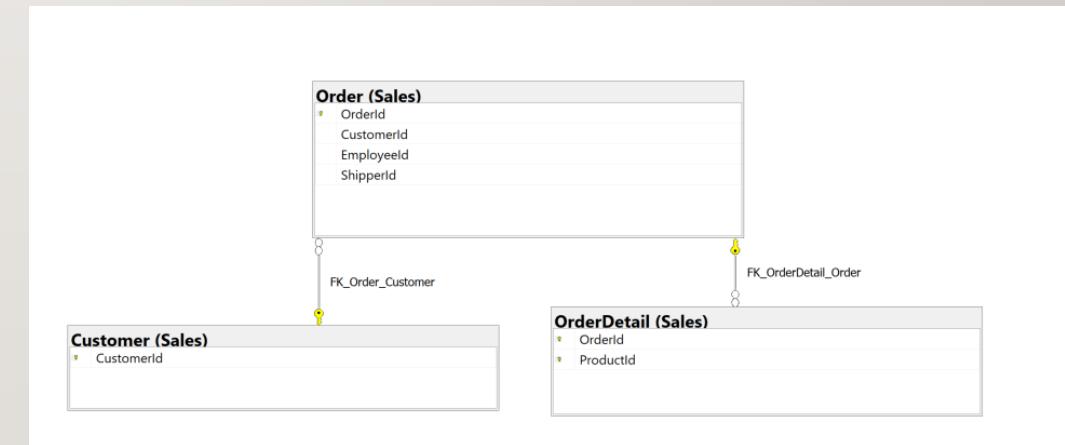


MARLON'S WORST MEDIUM QUERY: TABLE DIAGRAMS

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.OrderDeta il	TotalSpent	DESC



MARLON'S WORST MEDIUM QUERY: OUTPUT

100 %

Results Messages

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

Query executed successfully.

jSToolNpp JSON Viewer

ROOT

Medium 2 Output: [Array]

[0]: [Object]

CustomerId: 63
CustomerContactName: "Veronesi, Giorgio"
totalSpent: 117483.3900

[1]: [Object]

[2]: [Object]

[3]: [Object]

[4]: [Object]

CustomerId: 65
CustomerContactName: "Moore, Michael"
totalSpent: 52245.9000

"Medium 2 Output": [{
"CustomerId": 63,
"CustomerContactName": "Veronesi, Giorgio",
"totalSpent": 117483.3900
}, {
"CustomerId": 71,
"CustomerContactName": "Navarro, Tomás",
"totalSpent": 115673.3900
}, {
"CustomerId": 20,
"CustomerContactName": "Kane, John",
"totalSpent": 113236.6800
}, {
"CustomerId": 37,
"CustomerContactName": "Óskarsson, Jón Harry",
"totalSpent": 57317.3900
}, {
"CustomerId": 65,
"CustomerContactName": "Moore, Michael",
"totalSpent": 52245.9000
}]



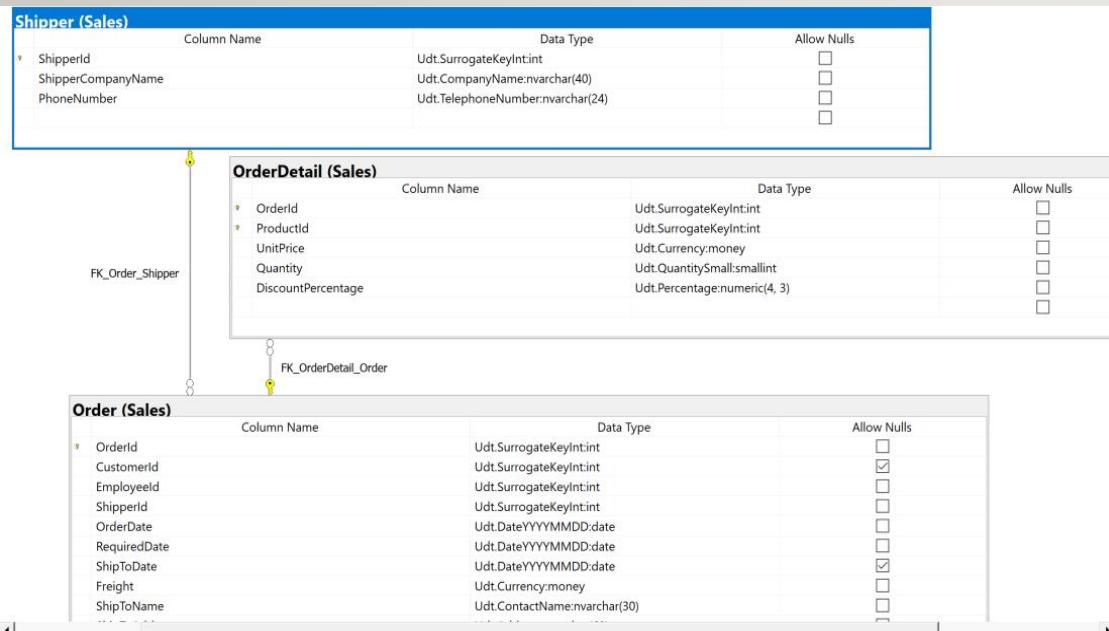
MARLON'S WORST COMPLEX QUERY

```
--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;
```

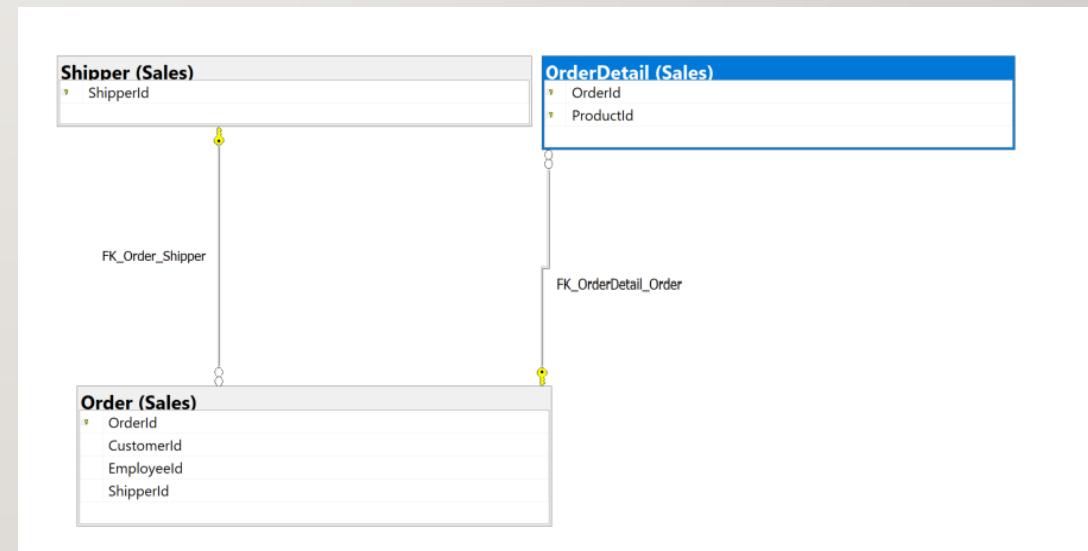


MARLON'S WORST COMPLEX QUERY: TABLE DIAGRAMS

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

JSToolNpp JSON Viewer

Refresh Search new 1

ROOT

Complex 2 Output: [Array]

[0]: [Object]

ShipperId: 2
ShipperCompanyName: "Shipper ETYNR"
NumDeliveries: 864
FreqCust: "Kane, John"

[1]: [Object]

[2]: [Object]

```
1  {
2    "Complex 2 Output": [
3      {
4        "ShipperId": 2,
5        "ShipperCompanyName": "Shipper ETYNR",
6        "NumDeliveries": 864,
7        "FreqCust": "Kane, John"
8      },
9      {
10        "ShipperId": 1,
11        "ShipperCompanyName": "Shipper GVSUA",
12        "NumDeliveries": 646,
13        "FreqCust": "Veronesi, Giorgio"
14      },
15      {
16        "ShipperId": 3,
17        "ShipperCompanyName": "Shipper ZHISN",
18        "NumDeliveries": 645,
19        "FreqCust": "Navarro, Tomás"
20      }
21    ]
22  }
```



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	<input type="checkbox"/>	
TerritoryID	int	<input checked="" type="checkbox"/>	
SalesQuota	money	<input checked="" type="checkbox"/>	
Bonus	money	<input checked="" type="checkbox"/>	
CommissionPct	smallmoney	<input type="checkbox"/>	
SalesYTD	money	<input type="checkbox"/>	
SalesLastYear	money	<input type="checkbox"/>	
rowguid	uniqueidentifier	<input type="checkbox"/>	
ModifiedDate	datetime	<input type="checkbox"/>	

Person (Person)			
Column Name	Data Type	Allow Nulls	iPerson (Person)
BusinessEntityID	int	<input type="checkbox"/>	
PersonType	nchar(2)	<input type="checkbox"/>	
NameStyle	NameStyle bit	<input type="checkbox"/>	
Title	nvarchar(8)	<input type="checkbox"/>	
FirstName	Name:nvarchar(50)	<input type="checkbox"/>	
MiddleName	Name:nvarchar(50)	<input type="checkbox"/>	
LastName	Name:nvarchar(50)	<input type="checkbox"/>	
Suffix	nvarchar(10)	<input type="checkbox"/>	
EmailPromotion	int	<input type="checkbox"/>	
AdditionalContactInfo	xml[CONTENT Person.AdditionalContactInfoSchemaC...]	<input type="checkbox"/>	
Demographics	xml[CONTENT Person.IndividualSurveySchemaCollecti...]	<input type="checkbox"/>	
rowguid	uniqueidentifier	<input type="checkbox"/>	
ModifiedDate	datetime	<input type="checkbox"/>	

Key View



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	Tsolfias	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	Anzman-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: "Anzman-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]

BusinessEntityID: 281
FirstName: "Shu"
LastName: "Ito"
SalesYTD: 2458535.6169
SalesLastYear: 2073505.9999

[17]: [Object]

BusinessEntityID: 282
FirstName: "José"
LastName: "Saraiva"
SalesYTD: 2604540.7172
SalesLastYear: 2038234.6549

[18]: [Object]

BusinessEntityID: 277
FirstName: "Jillian"
LastName: "Carson"
SalesYTD: 3189418.3662
SalesLastYear: 1997186.2037

[19]: [Object]

BusinessEntityID: 280
FirstName: "Pamela"
LastName: "Anzman-Wolfe"
SalesYTD: 1352577.1325
SalesLastYear: 1927059.1780

[20]: [Object]

BusinessEntityID: 279
FirstName: "Tsvi"
LastName: "Reiter"
SalesYTD: 2315185.6110
SalesLastYear: 1849640.9418

[21]: [Object]

BusinessEntityID: 275
FirstName: "Michael"
LastName: "Blythe"
SalesYTD: 3763178.1787
SalesLastYear: 1750406.4785

[22]: [Object]

[23]: [Object]

[24]: [Object]

[25]: [Object]

[26]: [Object]

[27]: [Object]

[28]: [Object]

[29]: [Object]

[30]: [Object]

[31]: [Object]

[32]: [Object]

[33]: [Object]

[34]: [Object]

[35]: [Object]

[36]: [Object]

[37]: [Object]

[38]: [Object]

[39]: [Object]

[40]: [Object]

[41]: [Object]

[42]: [Object]

[43]: [Object]

[44]: [Object]

[45]: [Object]

[46]: [Object]

[47]: [Object]

[48]: [Object]

[49]: [Object]

[50]: [Object]

JSON file



MARLON'S TOP MEDIUM QUERY

```
--(Medium 5) Find the resellers who sold the most value. List some information about 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	98	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.

JSToolNpp JSON Viewer

```

1  {
2    "Medium 5 Output": [
3      {
4        "ResellerKey": 15,
5        "ResellerName": "Budget Toy Store",
6        "AnnualSales": 3000000.0000,
7        "NumberEmployees": 52,
8        "SalesTerritoryCountry": "Australia",
9        "SalesTerritoryRegion": "Australia"
10       },
11      {
12        "ResellerKey": 18,
13        "ResellerName": "Catalog Store",
14        "AnnualSales": 3000000.0000,
15        "NumberEmployees": 55,
16        "SalesTerritoryCountry": "United States",
17        "SalesTerritoryRegion": "Central"
18       },
19      {
20        "ResellerKey": 21,
21        "ResellerName": "Chic Department Stores",
22        "AnnualSales": 3000000.0000,
23        "NumberEmployees": 58,
24        "SalesTerritoryCountry": "United States",
25        "SalesTerritoryRegion": "Southwest"
26       },
27      {
28        "ResellerKey": 24,
29        "ResellerName": "Eastside Department Store",
30        "AnnualSales": 3000000.0000,
31        "NumberEmployees": 61,
32        "SalesTerritoryCountry": "United States",
33        "SalesTerritoryRegion": "Southwest"
34       },
35      {
36        "ResellerKey": 27,
37        "ResellerName": "Sports Sales and Rental",
38        "AnnualSales": 3000000.0000,
39        "NumberEmployees": 64,
40        "SalesTerritoryCountry": "United States",
41        "SalesTerritoryRegion": "Southeast"
42       },
43      {
44        "ResellerKey": 30,
45        "ResellerName": "Cycle Merchants",
46        "AnnualSales": 3000000.0000,
47        "NumberEmployees": 67,
48        "SalesTerritoryCountry": "Canada",
49        "SalesTerritoryRegion": "Canada"
50       }
51    ]
52  }

```



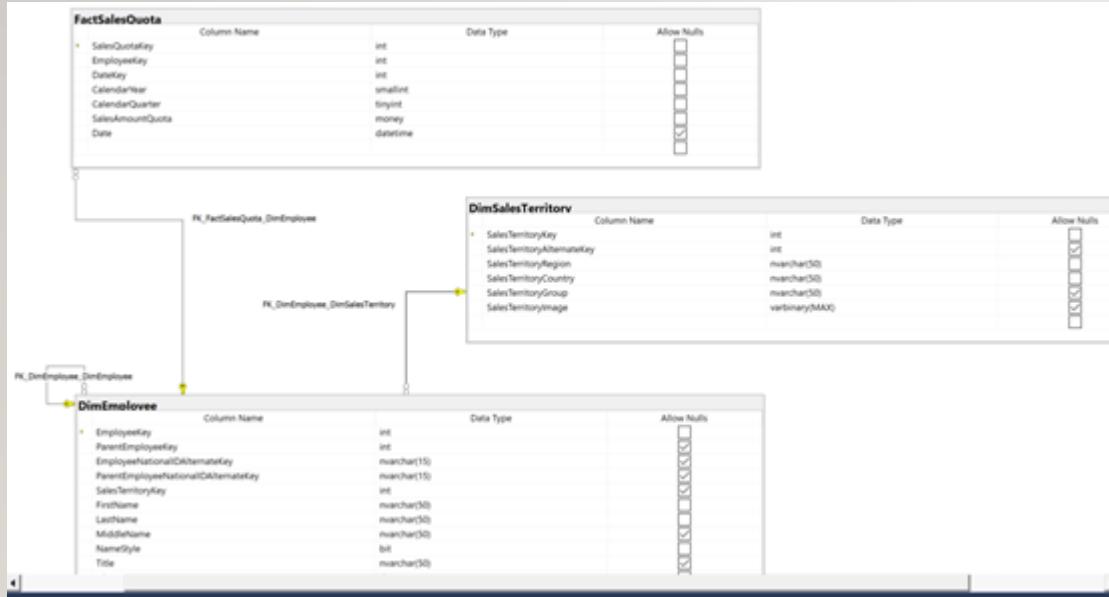
MARLON'S TOP COMPLEX QUERY

```
--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;
```

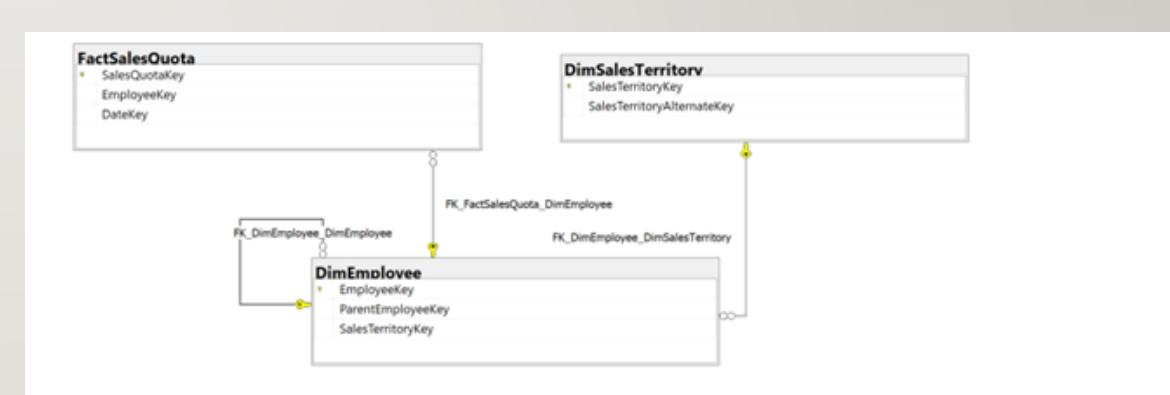


MARLON'S TOP COMPLEX QUERY: TABLE DIAGRAMS

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	ASC
	CalendarQuota	



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)

JSONToolbox JSON Viewer

```
Complex & Output: [Array]
[0]: [Object]
  EmployeeKey: 272
  FirstName: "Stephen"
  LastName: "Jiang"
  Title: "North American Sales Manager"
  SalesTerritoryCountry: "NA"
  CalendarYear: 2011
  CalendarQuarter: 1
  QtrQuota: 7000.0000
  QuarterSales: 2.25846955000000e+004
[1]: [Object]
  EmployeeKey: 272
  FirstName: "Stephen"
  LastName: "Jiang"
  Title: "North American Sales Manager"
  SalesTerritoryCountry: "NA"
  CalendarYear: 2011
  CalendarQuarter: 1
  QtrQuota: 115500.0000
  QuarterSales: 7.95142240000000e+004
[2]: [Object]
  EmployeeKey: 272
  FirstName: "Stephen"
  LastName: "Jiang"
  Title: "North American Sales Manager"
  SalesTerritoryCountry: "NA"
  CalendarYear: 2011
  CalendarQuarter: 3
  QtrQuota: 58000.0000
  QuarterSales: 9.73026600000000e+004
[3]: [Object]
  EmployeeKey: 272
  FirstName: "Stephen"
  LastName: "Jiang"
  Title: "North American Sales Manager"
  SalesTerritoryCountry: "NA"
  CalendarYear: 2012
  CalendarQuarter: 1
  QtrQuota: 154000.0000
  QuarterSales: 5.97161071000000e+004
[4]: [Object]
  EmployeeKey: 272
  FirstName: "Stephen"
  LastName: "Jiang"
  Title: "North American Sales Manager"
  SalesTerritoryCountry: "NA"
  CalendarYear: 2012
  CalendarQuarter: 2
  QtrQuota: 107000.0000
  QuarterSales: 1.38914775700000e+005
[5]: [Object]
  EmployeeKey: 272
  FirstName: "Stephen"
  LastName: "Jiang"
  Title: "North American Sales Manager"
  SalesTerritoryCountry: "NA"
  CalendarYear: 2012
  CalendarQuarter: 3
  QtrQuota: 263000.0000
  QuarterSales: 4.99755224000000e+004
[6]: [Object]
  EmployeeKey: 272
  FirstName: "Stephen"
  LastName: "Jiang"
  Title: "North American Sales Manager"
  SalesTerritoryCountry: "NA"
  CalendarYear: 2012
  CalendarQuarter: 4
  QtrQuota: 116000.0000
  QuarterSales: 2.17545499100000e+005
[7]: [Object]
  EmployeeKey: 272
  FirstName: "Stephen"
  LastName: "Jiang"
  Title: "North American Sales Manager"
  SalesTerritoryCountry: "NA"
  CalendarYear: 2013
  CalendarQuarter: 1
  QtrQuota: 84000.0000
  QuarterSales: 9.27523430000000e+004
```

