Individual Queries

CSCI 331 Database

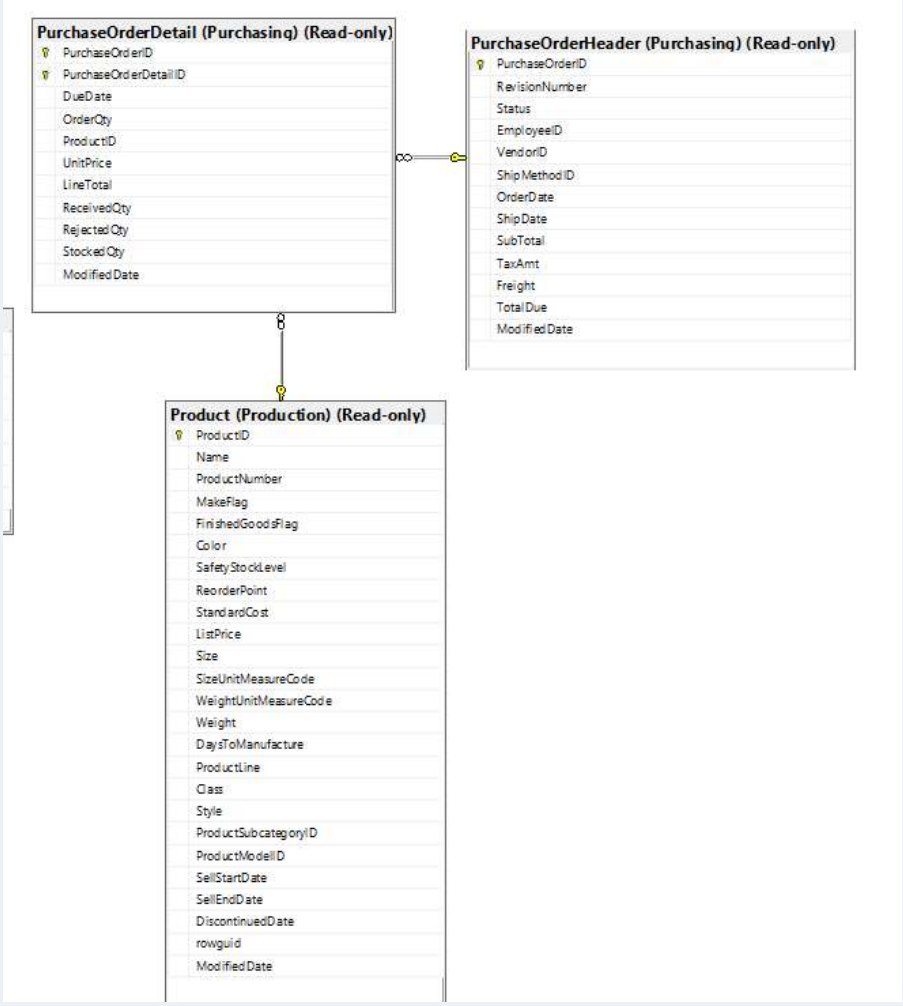
Student: JASMINE KIM
Teacher: PETER HELLER
Group: 2

Contents

BEST

SIMPLE

Example of PurchaseOrderDetail sub-system in AdventureWorks2017



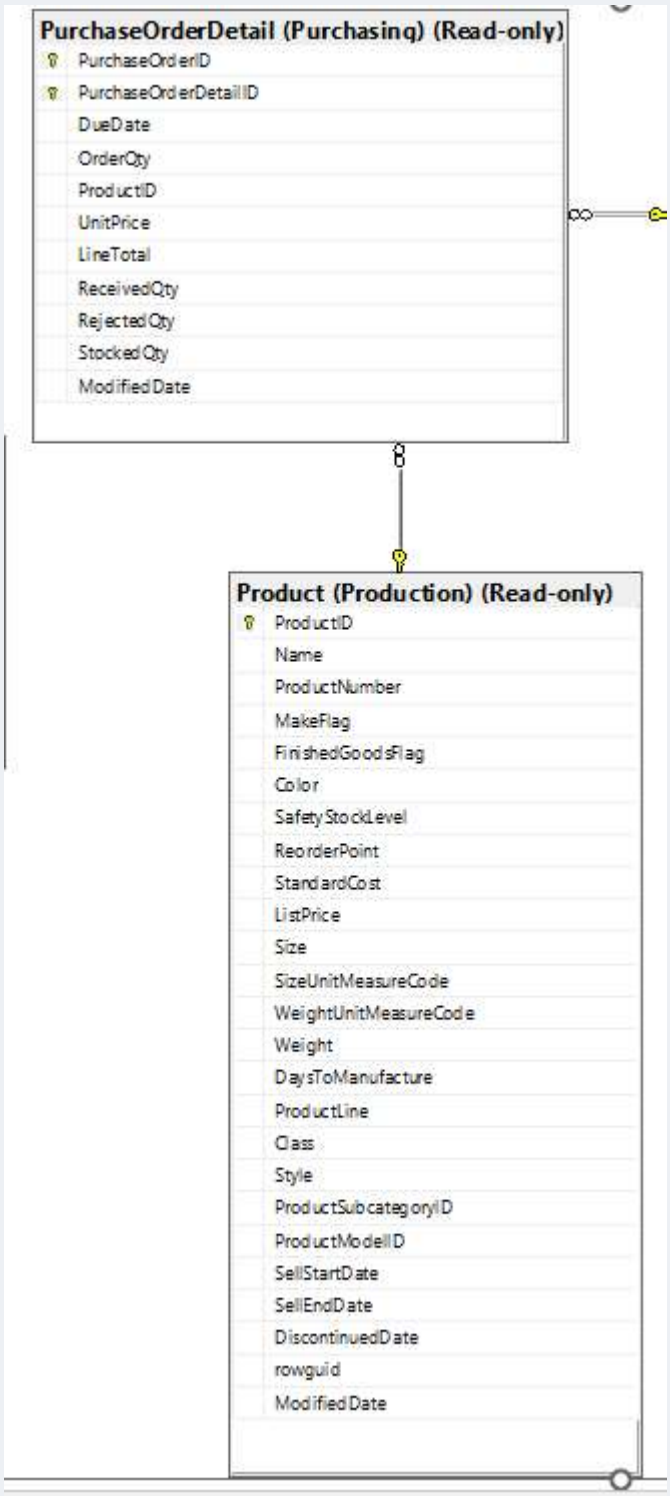
*Please note that this diagram is missing the constraint key information because I was unable to gain access to this database.

Proposition 01: Find out the list of products that have never been sold using AdventureWorks2017.

Detailed Explanation of the problem that will help the developer to write the query to resolve the issue.

To get the product information, we can use the Production.Product table. In order to identify products that have never been purchased, we need link to the Purchasing.PurchaseOrderDetail that shows the list of products that have been purchased. Products that have never been purchased will not have the PurchaseOrderID. Perform a LEFT OUTER JOIN and filter the rows by specifying in the WHERE clause: PurchaseOrderID is NULL.

Diagram of Tables



Columns from Standard View

*Please note that this diagram is missing because I no longer have access to this database half-way through the project.

Project following columns from their respective tables in the select clause

Table Name	Column Name
Product.Production	ProductId Name
Purchasing.PurchaseOrderDetail	PurchaseOrderID

Order By

Table Name	Column Name	Sort Order
Product.Production	ProductId	ASC

Problem Solving Query

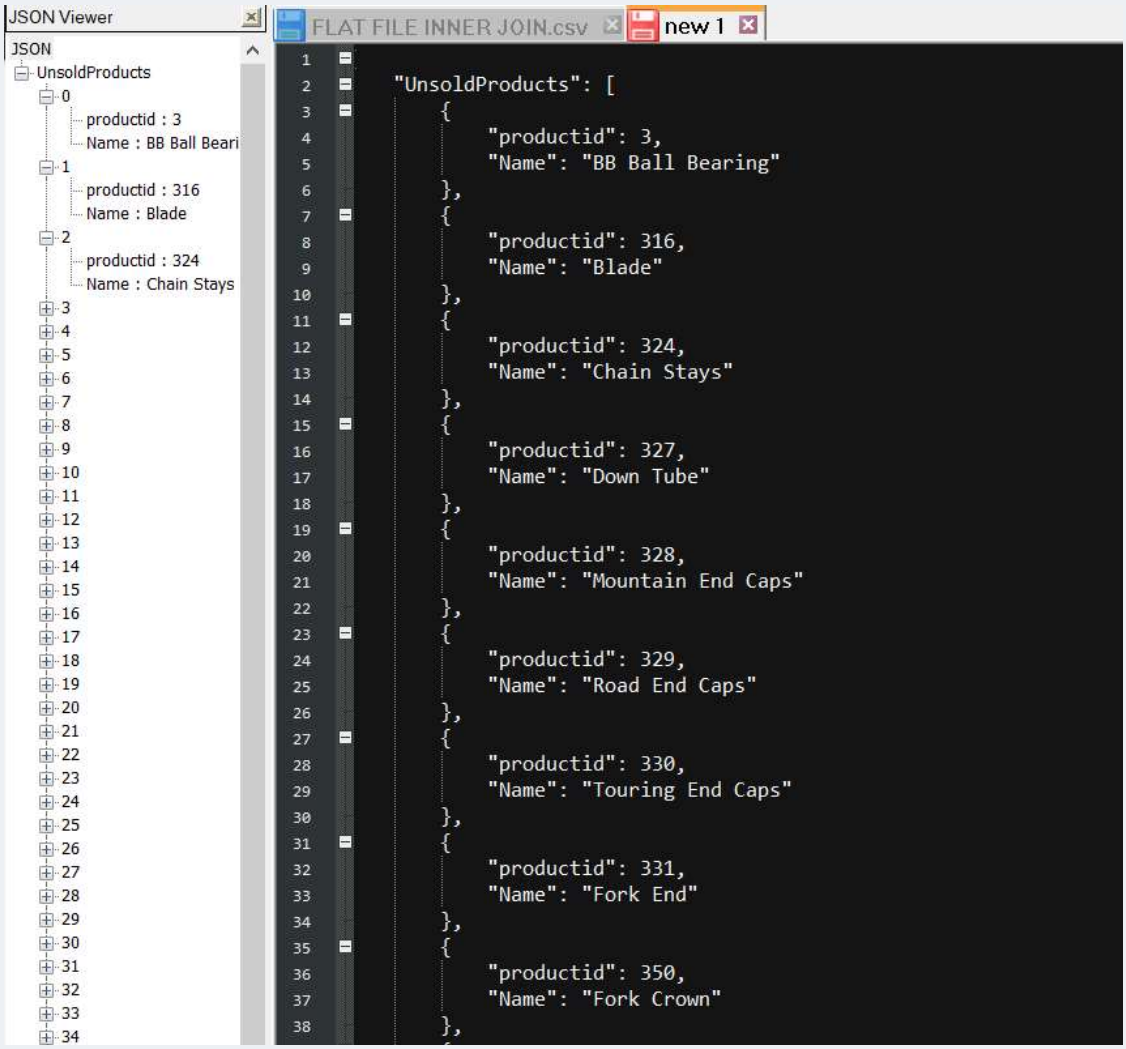
```
use AdventureWorks2017
select p.productid, p.Name
from production.product as p
    left outer join purchasing.purchaseOrderDetail as od
        on od.productid = p.productid
where od.PurchaseOrderID is null
ORDER BY p.productid;
```

Sample Relational Output with total number of rows returned (239)

	productid	Name
1	3	BB Ball Bearing
2	316	Blade
3	324	Chain Stays
4	327	Down Tube
5	328	Mountain End Caps
6	329	Road End Caps
7	330	Touring End Caps
8	331	Fork End
9	350	Fork Crown
10	398	Handlebar Tube
11	399	Head Tube
12	400	LL Hub
13	401	HL Hub
14	514	LL Mountain Seat Assembly
15	515	ML Mountain Seat Assembly
16	516	HL Mountain Seat Assembly
17	517	LL Road Seat Assembly

Sample JSON Output with total number of rows returned (239)

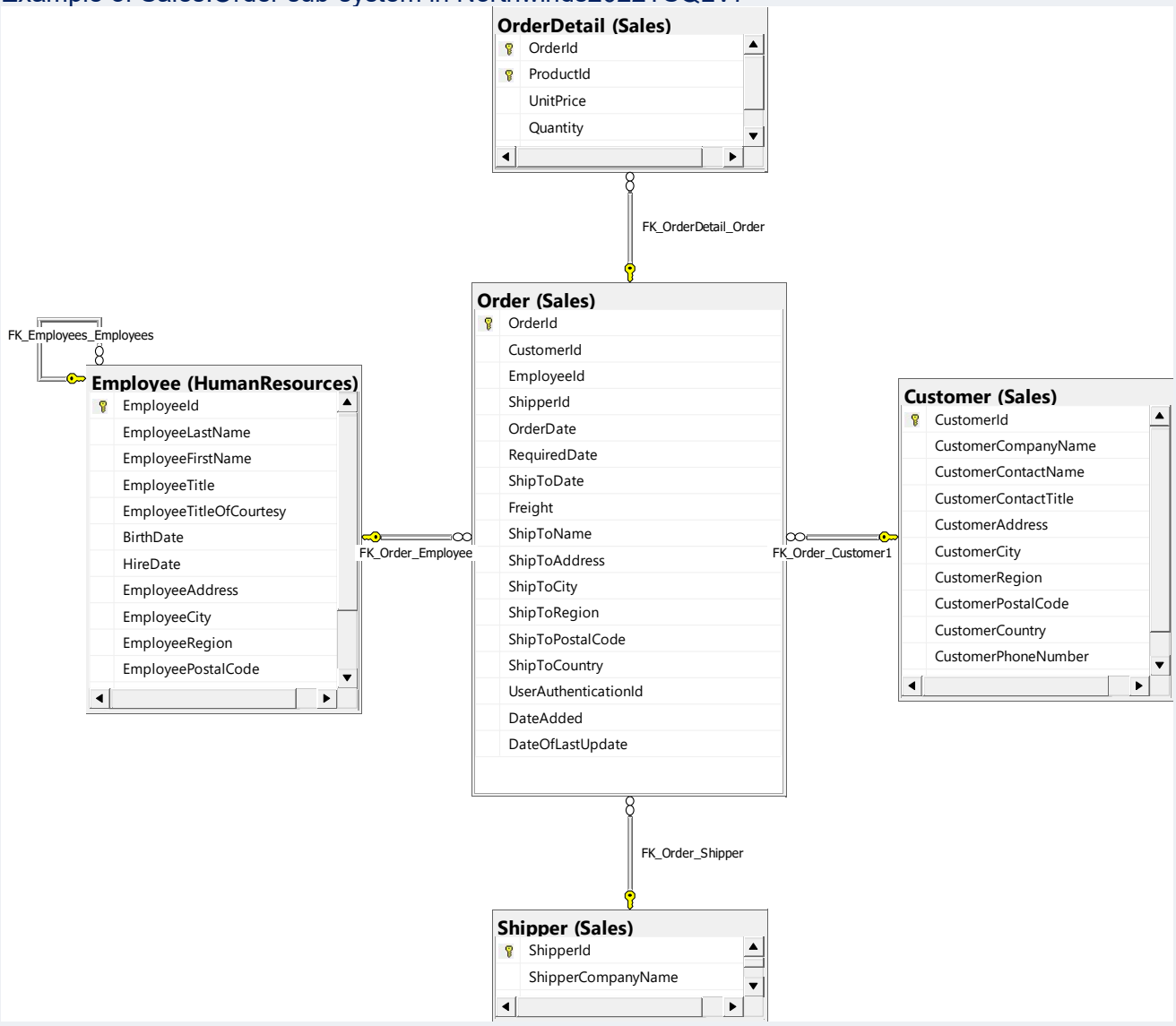
```sql
use AdventureWorks2017
select p.productid, p.Name
from production.product as p
    left outer join purchasing.purchaseOrderDetail as od
        on od.productid = p.productid
where od.PurchaseOrderID is null
ORDER BY p.productid
FOR JSON PATH, ROOT ('UnsoldProducts'), INCLUDE_NULL_VALUES;
```




```
{
    "UnsoldProducts": [
        {
            "productid": 3,
            "Name": "BB Ball Bearing"
        },
        {
            "productid": 316,
            "Name": "Blade"
        },
        {
            "productid": 324,
            "Name": "Chain Stays"
        },
        {
            "productid": 327,
            "Name": "Down Tube"
        },
        {
            "productid": 328,
            "Name": "Mountain End Caps"
        },
        {
            "productid": 329,
            "Name": "Road End Caps"
        },
        {
            "productid": 330,
            "Name": "Touring End Caps"
        },
        {
            "productid": 331,
            "Name": "Fork End"
        },
        {
            "productid": 350,
            "Name": "Fork Crown"
        },
```

MEDIUM

Example of Sales.Order sub-system in Northwinds2022TSQLV7



Proposition 02: Get the running total of the orders by year using NorthWinds2022TWQLV7.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue

To get the running total, first create a VIEW that summarizes the sales total by year. In this VIEW, use GROUP BY YEAR (OrderDate) and use the SUM function to get the total of the sales. In the outer query, use a subquery in the SELECT clause. The main query will join the VIEW to itself in the SELECT clause. The subquery will join the VIEW to itself using the variable, “year” because we want to see the total sales by *year*. If you are wondering which variable to use in the WHERE clause in the subquery, look at the proposition and identify what column/attribute comes after the word *by* or *per*. The operator in the WHERE clause of the running total queries will be <= to add current and all the

previous years' sales with the SUM function. Notice you can refer to the alias of the columns of the VIEW from the main query.

Diagram of Tables

Order (Sales)

- OrderId
- CustomerId
- EmployeeId
- ShipperId
- OrderDate
- RequiredDate
- ShipToDate
- Freight
- ShipToName
- ShipToAddress
- ShipToCity
- ShipToRegion
- ShipToPostalCode
- ShipToCountry
- UserAuthenticationId
- DateAdded
- DateOfLastUpdate

Columns from Standard View

Order (Sales)

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

Project following columns from their respective tables in the select clause

Table Name	Column Name
Sales[Order]	OrderDate Freight
Derived Column	[Year] TotalFreightCost runningTotal

Order By

Table Name	Column Name	Sort Order
Sales.OrderTotalByYear	[Year]	ASC

Problem Solving Query

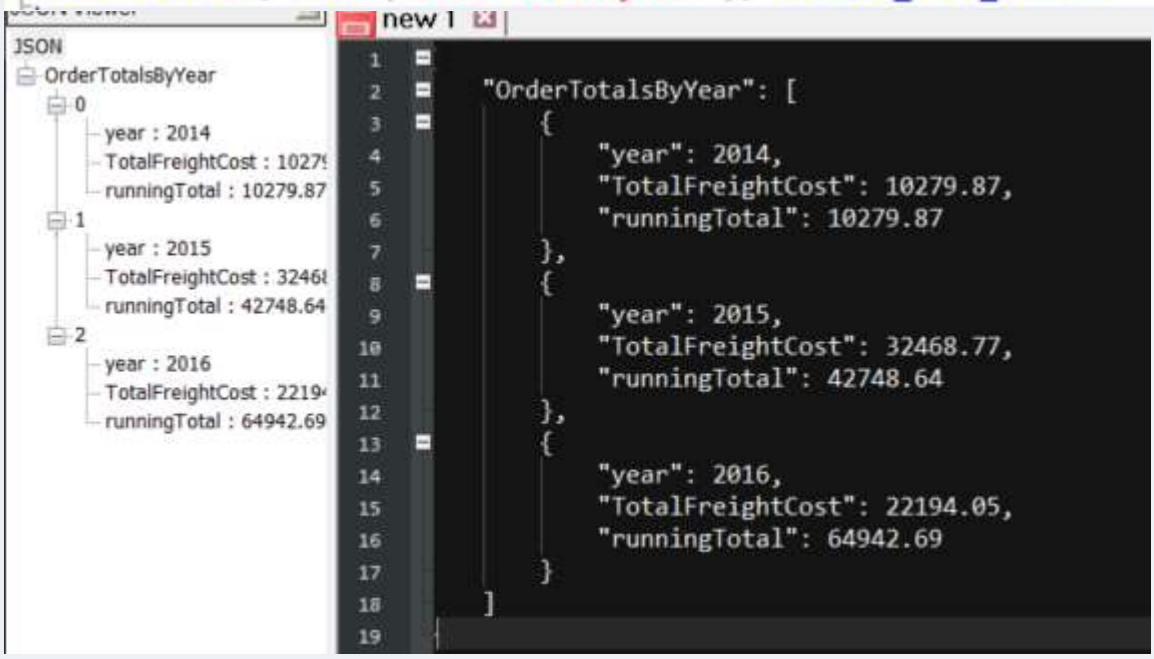
```
use Northwinds2022TSQLV7
drop VIEW if exists Sales.OrderTotalByYear;
go
create VIEW Sales.OrderTotalByYear
as

select year(orderDate) as [Year], sum(freight) as TotalFreightCost
from Sales.[Order]
group by year(orderDate)
go

select [year], TotalFreightCost
        , (select sum(O2.TotalFreightCost)
           from Sales.OrderTotalByYear as O2
           where O2.[year] <= O1.[year]
           )as runningTotal
from Sales.OrderTotalByYear as O1
order by [year]
```

Sample Relational Output with total number of rows returned (3)

	year	TotalFreightCost	runningTotal
1	2014	10279.87	10279.87
2	2015	32468.77	42748.64
3	2016	22194.05	64942.69

Sample JSON Output with total number of rows returned (3)

```sql
use Northwinds2022TSQLV7
select [year], TotalFreightCost
        , (select sum(O2.TotalFreightCost)
           from Sales.OrderTotalByYear as O2
           where O2.[year] <= O1.[year]
           )as runningTotal
from Sales.OrderTotalByYear as O1
order by [year]
FOR JSON PATH, ROOT ('OrderTotalsByYear'), INCLUDE_NULL_VALUES
```



COMPLEX

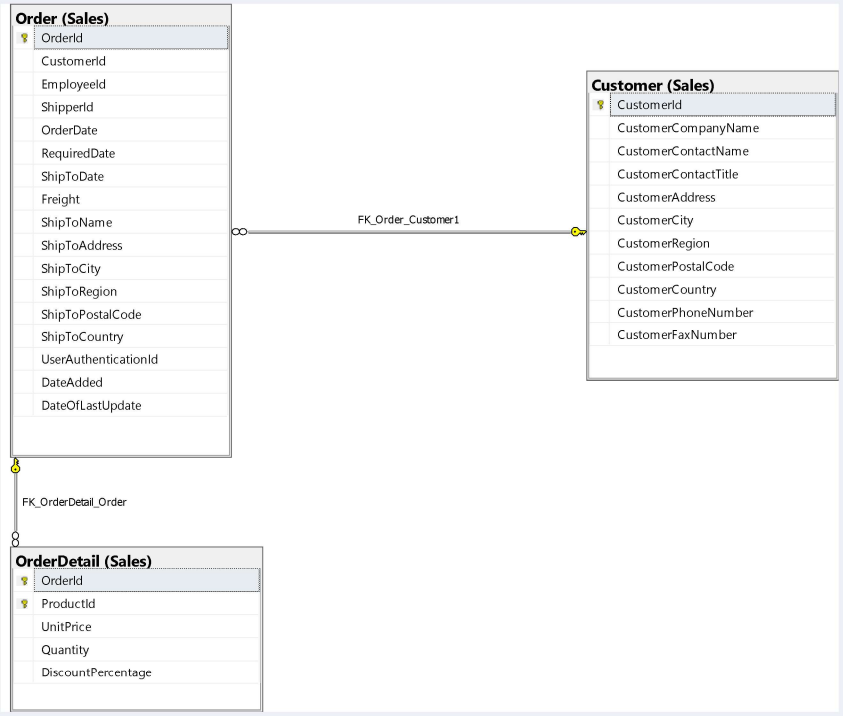
Example of OrderDetail sub-system in Northwinds2022TSQLV7

Proposition 03: Find the top 3 most expensive items each customer has ever purchased.

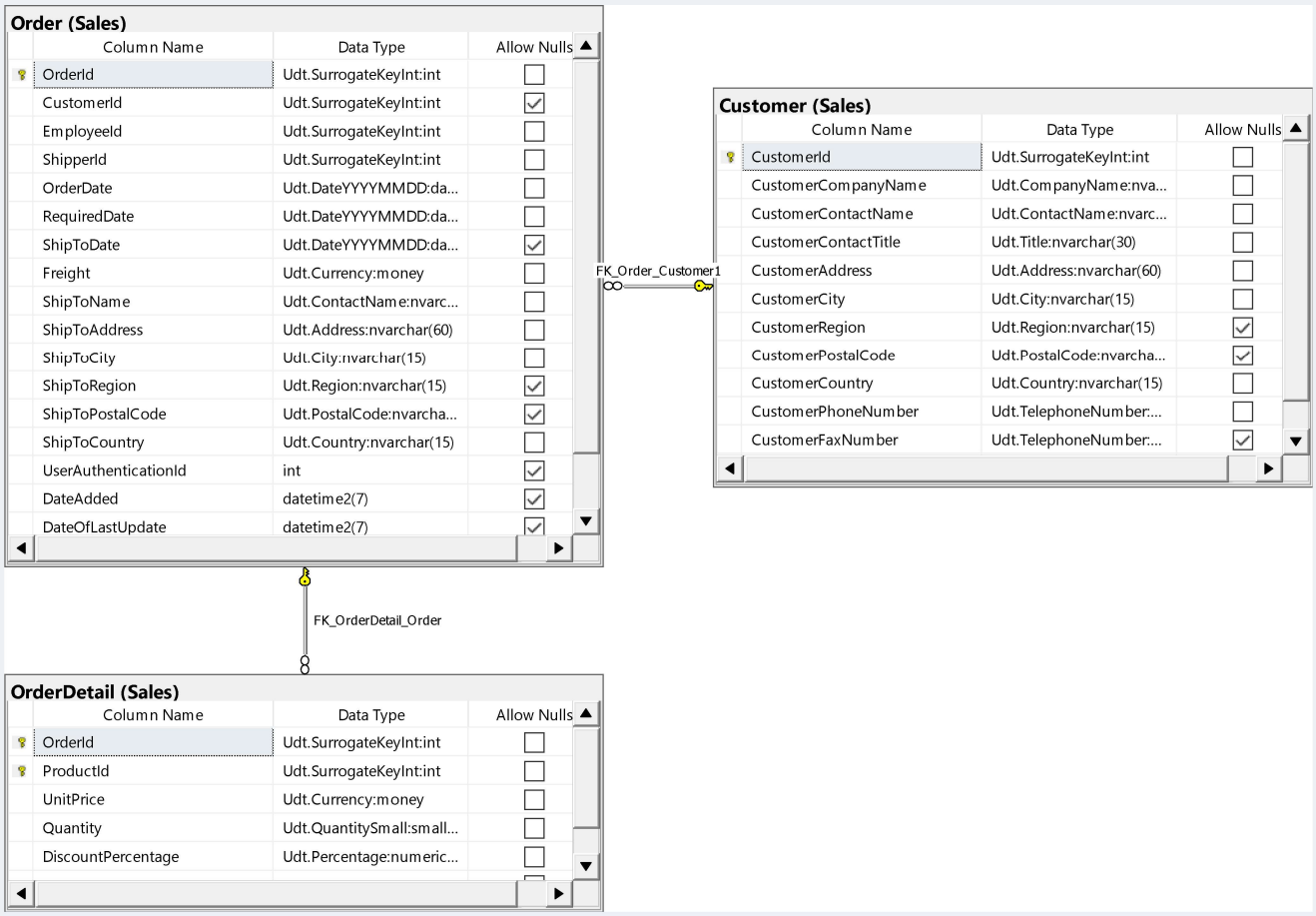
Detailed explanation of the problem that will help the developer to write the query to resolve the issue

Create an Inline Table-Valued Function that takes in the number of items to display and the CustomerID. This function performs a join with the Sales.OrderDetail and Sales.Order tables to output the ProductId data with the customers who ordered those products (OrderDetail table does not have the CustomerID). It does a filter/WHERE by an individual customer. If a constant number is entered as a parameter to this function, it will output products ordered by that individual customer. If a CustomerID attribute is entered as a parameter to this function, it will output the products ordered by all customers. The latter is what we are intending to do. ORDER BY is essential here in order to output the three most expensive products instead of *every* products the customer has ever purchased. The main/outer query, which uses the Inline Table-Valued Function, will perform an cross apply with the Customer table to output all customers (who placed an order) and their products. To obtain the customers will no orders, OUTER APPLY can be used. The ROW_NUMBER() function is used to specify that there are 3 products per each customer. In order to see that the correct query is being returned, use the ORDER BY CustomerID and UnitPrice in the main query. CROSS APPLY does not require ON clause to join the tables based on common columns, so the code may look simpler than INNER JOIN.

Diagram of Tables



Columns from Standard View



Project following columns from their respective tables in the select clause

Table Name	Column Name
Dbo.TopPriceItems	CustomerId OrderID ProductId UnitPrice
Sales.Customer	CustomerID
Derived Column	number

Order By

Table Name	Column Name	Sort Order
Sales.Customer	CustomerID	ASC
Dbo.TopPriceItems	UnitPrice	DESC

Problem Solving Query

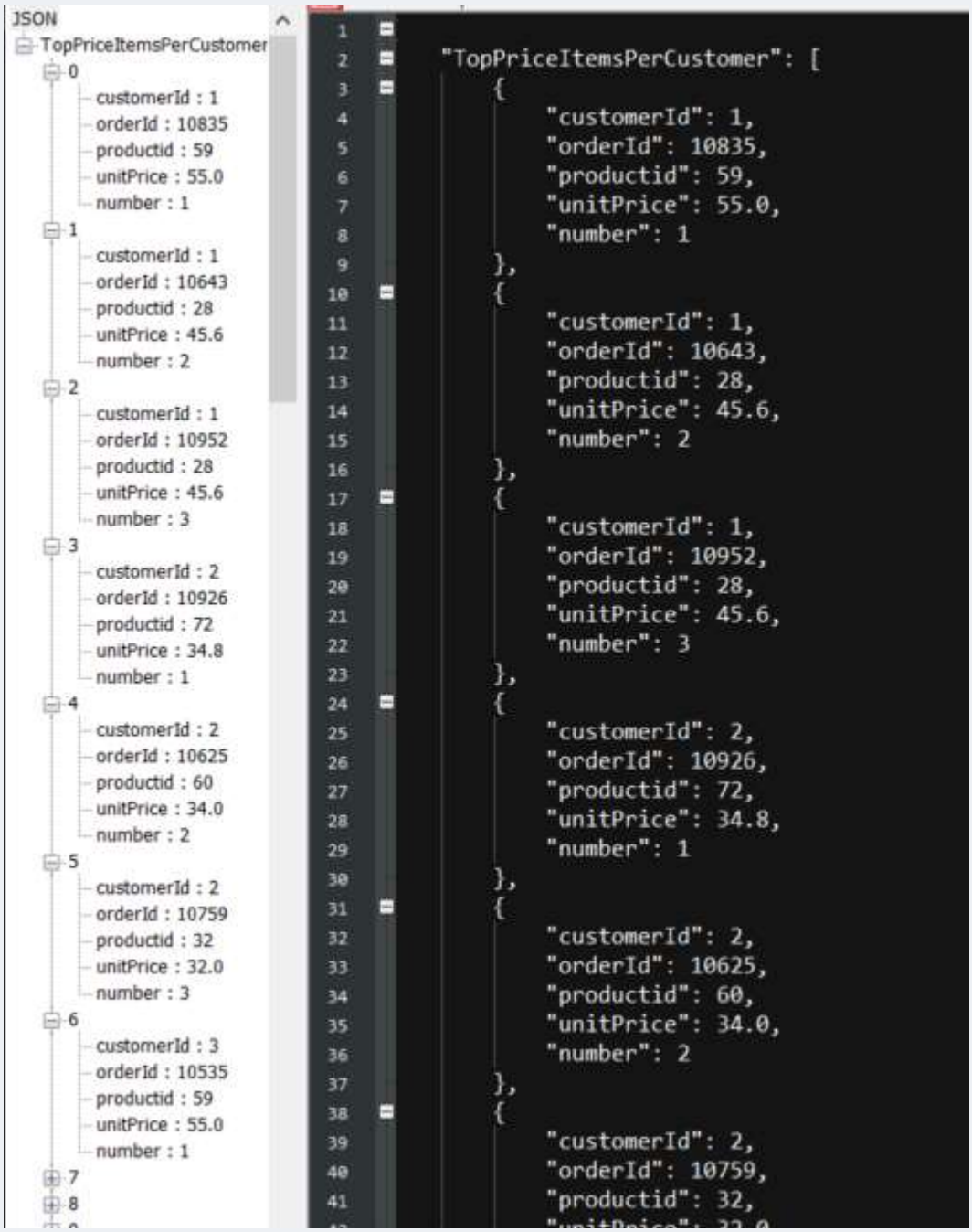
```sql
use Northwinds2022TSQLV7;
drop function if exists dbo.TopPriceItems;
go
create function dbo.TopPriceItems
(
    @howManyItems as int,
    @customerId as int
)
returns table
as
    return
        select top (@howmanyitems) o.customerId, od.OrderId, od.ProductId, od.UnitPrice
        from Sales.OrderDetail as od
            inner join Sales.[Order] as o
                on o.orderId = od.orderId
        where o.customerId = @customerId
        order by od.unitPrice desc
go
use Northwinds2022TSQLV7;
SELECT c.customerId
        , a.orderId, a.productid, a.unitPrice
        , row_number() over(partition by c.customerId order by a.unitPrice DESC) as number
from sales.Customer as c
    cross apply dbo.TopPriceItems (3, c.customerId) as a
order by c.customerId, a.UnitPrice DESC
```

Sample Relational Output with total number of rows returned (265)

Results | Messages

	customerId	orderId	productid	unitPrice	number
1	1	10835	59	55.00	1
2	1	10643	28	45.60	2
3	1	10952	28	45.60	3
4	2	10926	72	34.80	1
5	2	10625	60	34.00	2
6	2	10759	32	32.00	3
7	3	10535	59	55.00	1
8	3	10507	43	46.00	2
9	3	10573	17	39.00	3
10	4	10953	20	81.00	1
11	4	10558	51	53.00	2
12	4	10768	60	34.00	3
13	5	10672	38	263.50	1
14	5	10857	29	123.79	2
15	5	10384	20	64.80	3
16	6	10853	18	62.50	1
17	6	10956	51	53.00	2
18	6	10509	28	45.60	3
19	7	10360	38	210.80	1
20	7	10360	29	99.00	2
21	7	10566	18	62.50	3
22	8	10801	29	123.79	1
23	8	10801	17	39.00	2
24	8	10326	4	17.60	3
25	9	10663	51	53.00	1

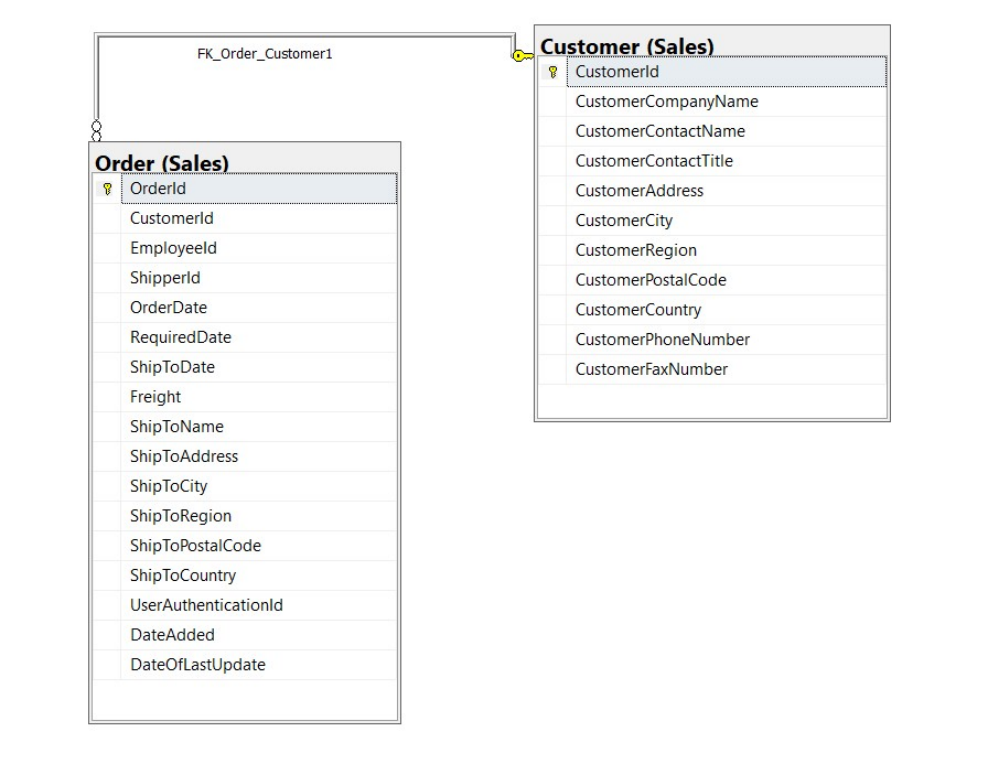
Sample JSON Output with total number of rows returned (265)



WORST

MEDIUM

Example of Customer sub-system in Northwinds2022TSQLV7



Proposition 04: Find the CustomerContactTitle, CustomerId, CustomerRegion, and CustomerPostalCode. Give the owners and sales agents their own number. Replace the empty/null customer region with more information.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue

First, create a VIEW that filters for the Owners and Sales Agents. In the main/outer query, use the ROW_NUMBER() function to generate a unique number for each Owners and Sales Agents. Use COALESCE function to display “No Region” instead of NULLs. Use CONCAT function to generate a new label for Owners and Sales Agents. Do an INNER JOIN with the Customer table to obtain the postal code. Even though this query essentially uses a single table, I categorized this as a MEDIUM because it uses a table expression which is later joined by the original table. So many instances of Customer table is used here. I wonder if I could just invoke the Customer table once without the need for the table expression to simply this query. This query unnecessarily uses additional table expression just to satisfy the requirements of the assignment.

Diagram of Tables

Customer (Sales)

CustomerId
CustomerCompanyName
CustomerContactName
CustomerContactTitle
CustomerAddress
CustomerCity
CustomerRegion
CustomerPostalCode
CustomerCountry
CustomerPhoneNumber
CustomerFaxNumber

Columns from Standard View

Customer (Sales)

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

Project following columns from their respective tables in the select clause

Table Name	Column Name
Utils.OwnersAndAgents (VIEW)	CustomerId CustomerContactTitle CustomerCity CustomerRegion CustomerCountry
Sales.Customer	CustomerPostalCode
Derived Column	CustomerRegion rowNum

	newLabel

Order By

Table Name	Column Name	Sort Order
Utils.OwnersAndAgents (VIEW)	CustomerContactTitle	ASC

Problem solving Query

```
use Northwinds2022TSQLV7
drop view if exists Utils.ownersAndAgents;
go
create VIEW Utils.ownersAndAgents
as

select CustomerId, CustomerContactTitle, CustomerCity, CustomerRegion, CustomerCountry
from sales.Customer
where CustomerContactTitle IN (N'Owner', N'Sales Agent');

go
```

```
use Northwinds2022TSQLV7
SELECT oa.CustomerContactTitle, oa.CustomerId
       , coalesce (oa.CustomerRegion, N'No Region')  as CustomerRegion
       ,Row_Number() over (partition by oa.CustomerContactTitle order by oa.customerID) as rowNum
       ,concat(oa.CustomerContactTitle,(Row_Number() over (partition by oa.CustomerContactTitle order by oa.customerID))) as newLabel
       ,c.CustomerPostalCode
from Utils.ownersAndAgents as oa
inner join Sales.Customer as c
    on c.customerId = oa.customerId
```

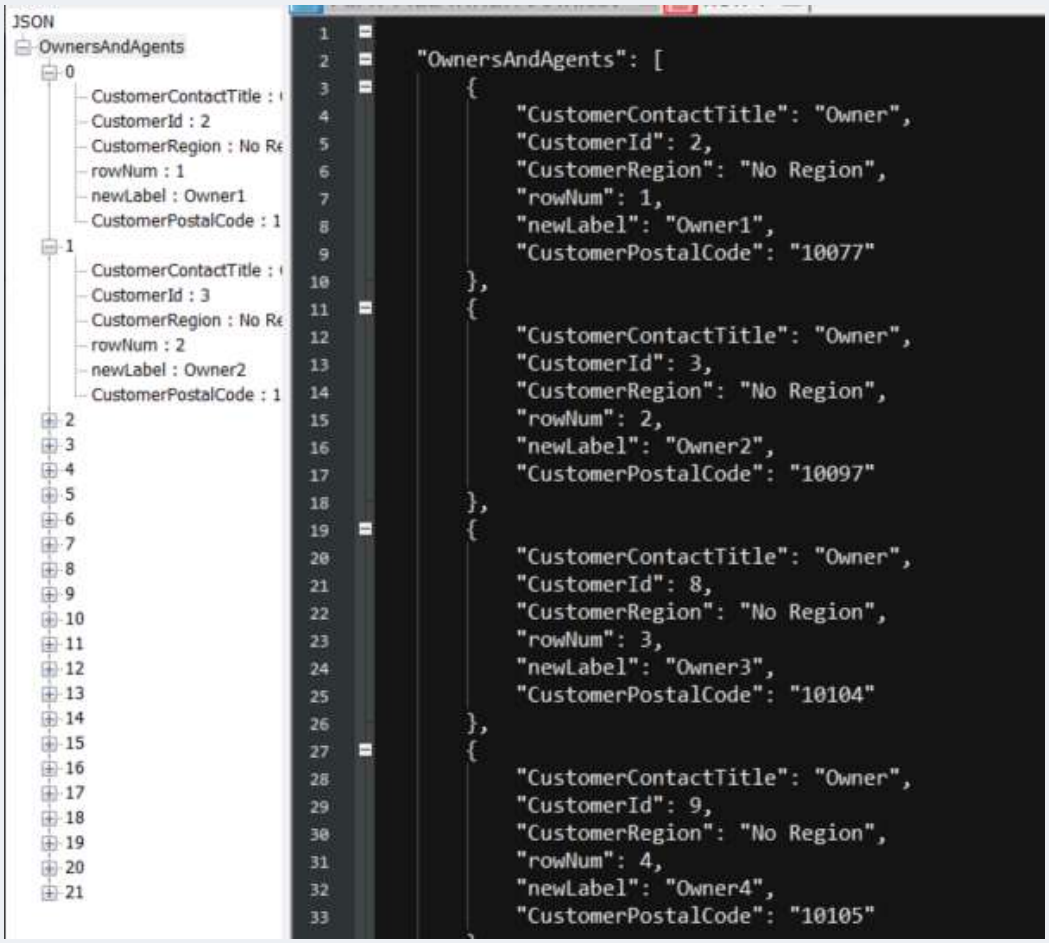
Sample Relational Output with total number of rows returned (22)

Results | Messages

	CustomerContactTitle	CustomerId	CustomerRegion	rowNum	newLabel	CustomerPostalCode
1	Owner	2	No Region	1	Owner1	10077
2	Owner	3	No Region	2	Owner2	10097
3	Owner	8	No Region	3	Owner3	10104
4	Owner	9	No Region	4	Owner4	10105
5	Owner	14	No Region	5	Owner5	10065
6	Owner	18	No Region	6	Owner6	10041
7	Owner	24	No Region	7	Owner7	10114
8	Owner	33	DF	8	Owner8	10043
9	Owner	45	CA	9	Owner9	10062
10	Owner	47	Nueva Esparta	10	Owner10	10121
11	Owner	56	No Region	11	Owner11	10047
12	Owner	57	No Region	12	Owner12	10085
13	Owner	70	No Region	13	Owner13	10123
14	Owner	73	No Region	14	Owner14	10079
15	Owner	80	No Region	15	Owner15	10044
16	Owner	89	WA	16	Owner16	10049
17	Owner	91	No Region	17	Owner17	10068
18	Sales Agent	12	No Region	1	Sales Agent1	10057
19	Sales Agent	19	No Region	2	Sales Agent2	10110
20	Sales Agent	50	No Region	3	Sales Agent3	10074
21	Sales Agent	54	No Region	4	Sales Agent4	10094
22	Sales Agent	84	No Region	5	Sales Agent5	10072

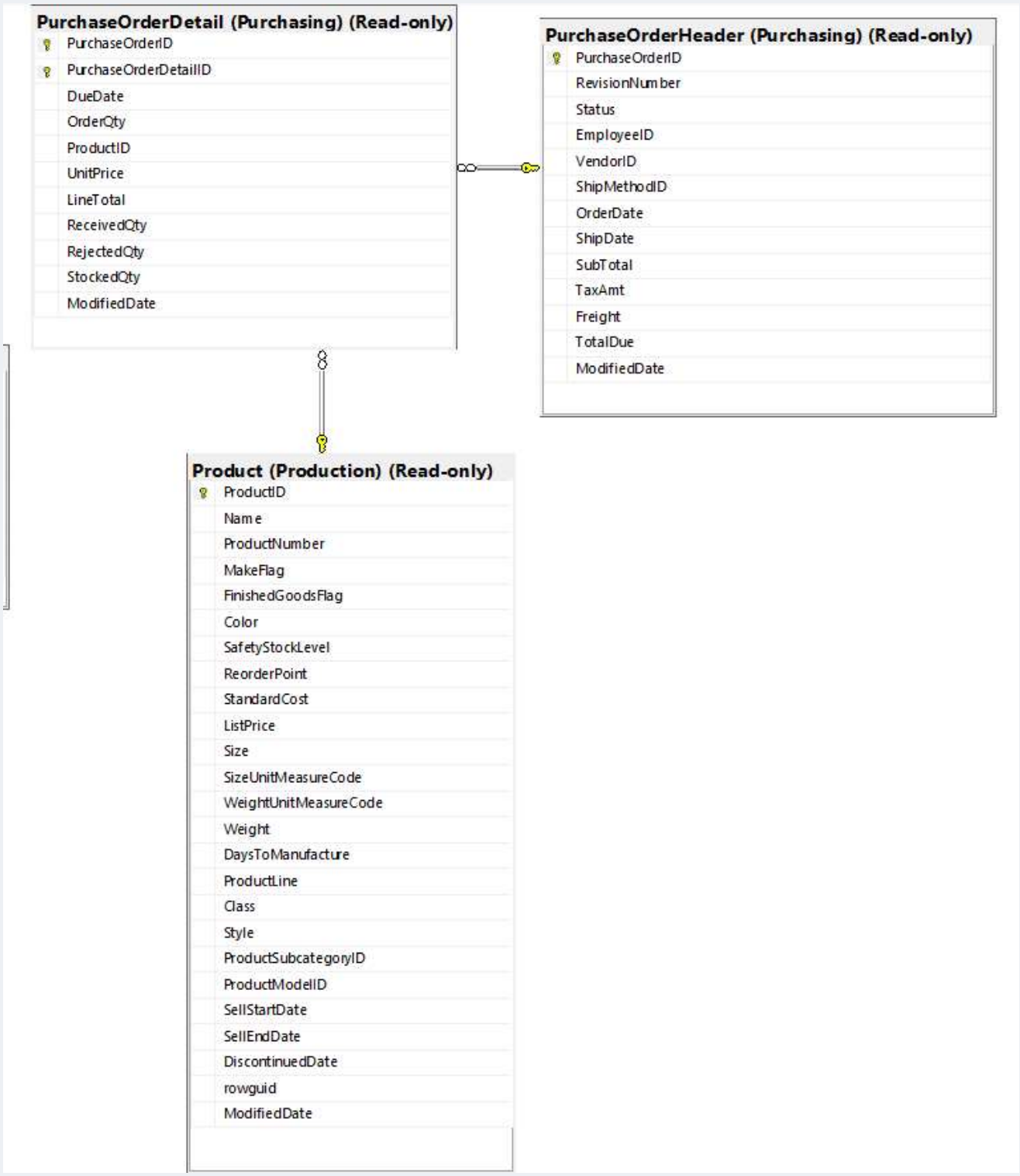
Sample JSON Output with total number of rows returned (22)

```sql
use Northwinds2022TSQLV7
SELECT oa.CustomerContactTitle, oa.CustomerId
       , coalesce (oa.CustomerRegion, N'No Region')  as CustomerRegion
       ,Row_Number() over (partition by oa.CustomerContactTitle order by o
       ,concat(oa.CustomerContactTitle,(Row_Number() over (partition by oa
       ,c.CustomerPostalCode
from Utils.ownersAndAgents as oa
inner join Sales.Customer as c
    on c.customerId = oa.customerId
ORDER BY oa.CustomerContactTitle
FOR JSON PATH, ROOT ('OwnersAndAgents'), INCLUDE_NULL_VALUES;
```



COMPLEX I

Example of PurchaseOrderDetail sub-system in AdventureWorks2017



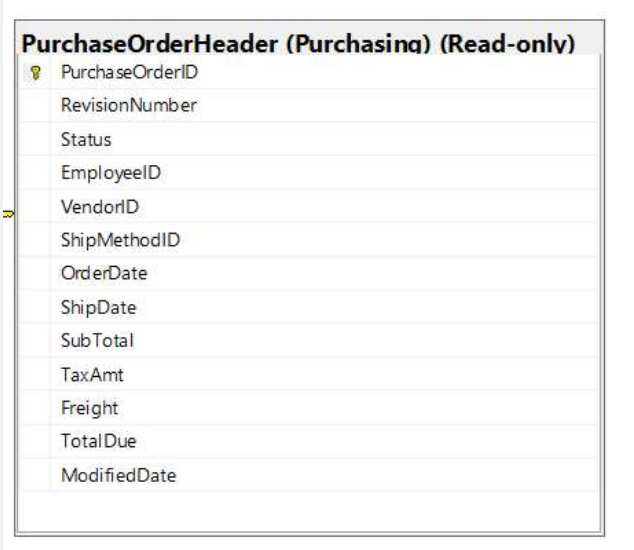
*Please note that this diagram is missing the constraint key information because I was not able to gain access to this database.

Proposition 05: Show percentage of each employee's order total as a ratio of the employee's total orders serviced. Also show percentage of the employee's order against all orders.

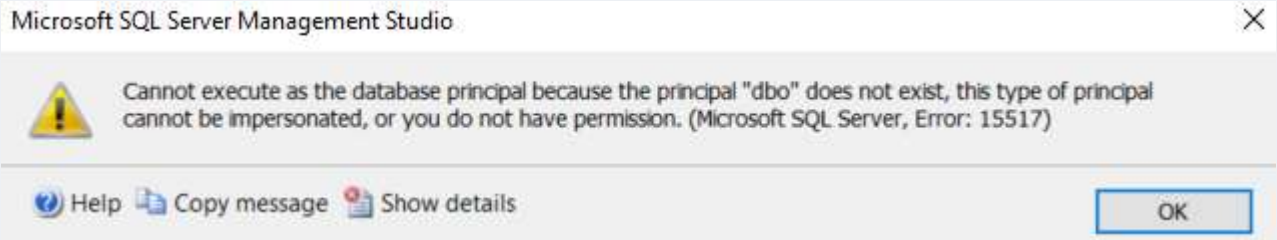
Detailed explanation of the problem that will help the developer to write the query to resolve the issue

There are four subqueries in the SELECT clause in this query. These are two subqueries that are invoked two times each. The first subquery obtains the total sales amount serviced by each employee. The second query obtains the total sales amount of the entire company. The amount returned by the first query divides each sales order then it is multiplied by 100. The amount returned by the second query divides each sales order then it is multiplied by 100. This query shows the impact each order and each employee has toward the company's overall profit. The number of times the subqueries invoked clouds the overall query.

Diagram of Tables



Columns from Standard View



*Please note that I was unable to obtain the data type of each column due to access issue.

Project following columns from their respective tables in the select clause

Table Name	Column Name
Purchasing.PurchaseOrderHeader	EmployeeId PurchaseOrderId TotalDue

Order By

Table Name	Column Name	Sort Order
Purchasing.PurchaseOrderHeader	EmployeeId	ASC
Purchasing.PurchaseOrderHeader	PurchaseOrderId	ASC

Problem solving Query

```
use AdventureWorks2017
select employeeId, PurchaseOrderId, TotalDue
       ,  FORMAT((100. * O1.TotalDue/(select sum(O2.TotalDue)
                                 from purchasing.PurchaseOrderHeader as O2
                                 where O2.EmployeeID = O1.EmployeeID)
          ),'P') as PtyOfTotalEmployeeOrder
        , FORMAT((select sum(O2.TotalDue)
           from purchasing.PurchaseOrderHeader as O2
           where O2.EmployeeID = O1.EmployeeID),'C')as employeeTotal
        , Format (100. * O1.TotalDue/ (SELECT SUM(O2.TotalDue) FROM purchasing.PurchaseOrderHeader as O2),'P') AS PtyOFAllTotal
        , FORMAT((SELECT SUM(O2.TotalDue) FROM purchasing.PurchaseOrderHeader as O2), 'C') AS allTotal
from purchasing.PurchaseOrderHeader as O1
Order BY EmployeeID,  PurchaseOrderID;
```

Sample Relational Output with total number of rows returned (4012)

	employeeId	PurchaseOrderId	TotalDue	PtyOfTotalEmployeeOrder	employeeTotal	PtyOFAllTotal	allTotal
1	250	10	1984.6192	7.93%	$2,501,613.04	0.28%	$70,479,332.64
2	250	21	7721.4638	30.87%	$2,501,613.04	1.10%	$70,479,332.64
3	250	45	31160.2541	124.56%	$2,501,613.04	4.42%	$70,479,332.64
4	250	92	284.6209	1.14%	$2,501,613.04	0.04%	$70,479,332.64
5	250	110	157.3647	0.63%	$2,501,613.04	0.02%	$70,479,332.64
6	250	121	38281.8686	153.03%	$2,501,613.04	5.43%	$70,479,332.64
7	250	145	731.6189	2.92%	$2,501,613.04	0.10%	$70,479,332.64
8	250	192	1410.2839	5.64%	$2,501,613.04	0.20%	$70,479,332.64
9	250	210	126.4905	0.51%	$2,501,613.04	0.02%	$70,479,332.64
10	250	221	590.9618	2.36%	$2,501,613.04	0.08%	$70,479,332.64
11	250	245	525.628	2.10%	$2,501,613.04	0.07%	$70,479,332.64
12	250	292	462.9398	1.85%	$2,501,613.04	0.07%	$70,479,332.64
13	250	310	1043.5289	4.17%	$2,501,613.04	0.15%	$70,479,332.64
14	250	321	22539.0165	90.10%	$2,501,613.04	3.20%	$70,479,332.64
15	250	345	458.4844	1.83%	$2,501,613.04	0.07%	$70,479,332.64
16	250	392	41817.1504	167.16%	$2,501,613.04	5.93%	$70,479,332.64
17	250	410	113.3332	0.45%	$2,501,613.04	0.02%	$70,479,332.64
18	250	420	3758.6299	15.02%	$2,501,613.04	0.53%	$70,479,332.64
19	250	438	2032.6535	8.13%	$2,501,613.04	0.29%	$70,479,332.64
20	250	449	8423.415	33.67%	$2,501,613.04	1.20%	$70,479,332.64
21	250	473	10828.6133	43.29%	$2,501,613.04	1.54%	$70,479,332.64
22	250	520	166.6235	0.67%	$2,501,613.04	0.02%	$70,479,332.64
23	250	542	1410.2839	5.64%	$2,501,613.04	0.20%	$70,479,332.64
24	250	553	31160.2541	124.56%	$2,501,613.04	4.42%	$70,479,332.64
25	250	577	100685.3348	402.48%	$2,501,613.04	14.29%	$70,479,332.64

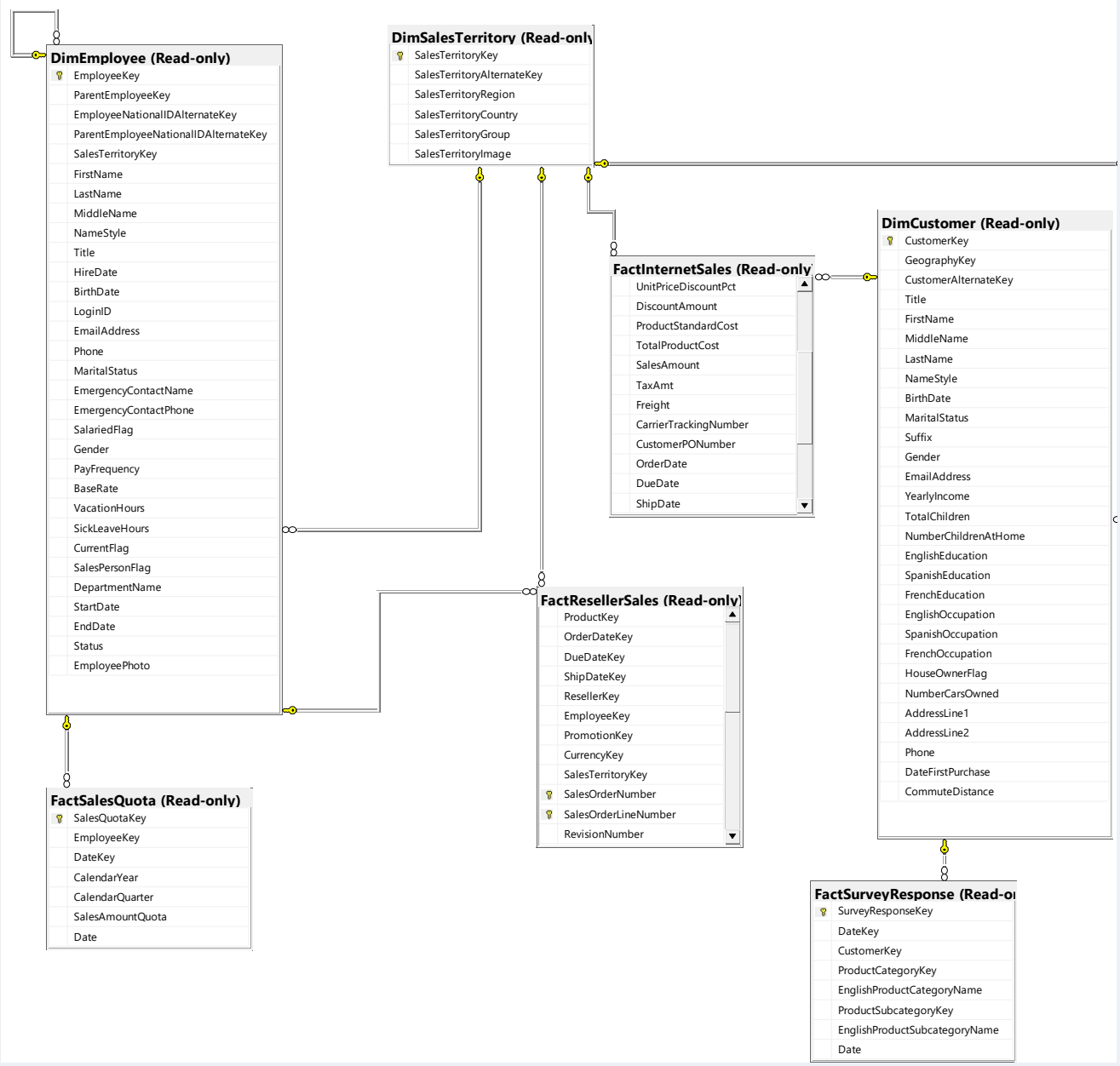
Sample JSON Output with total number of rows returned (4012)

```sql
use AdventureWorks2017
select employeeId, PurchaseOrderId, TotalDue
        ,  FORMAT((100. * O1.TotalDue/(select sum(O2.TotalDue)
                                 from purchasing.PurchaseOrderHeader as O2
                                 where O2.EmployeeID = O1.EmployeeID)
          ),'P') as PtyOfTotalEmployeeOrder
         , FORMAT((select sum(O2.TotalDue)
           from purchasing.PurchaseOrderHeader as O2
           where O2.EmployeeID = O1.EmployeeID),'C')as employeeTotal
         , Format (100. * O1.TotalDue/ (SELECT SUM(O2.TotalDue) FROM purchasing.PurchaseOrderHeader as O2),'P') AS PtyOFAllTotal
         , FORMAT((SELECT SUM(O2.TotalDue) FROM purchasing.PurchaseOrderHeader as O2), 'C') AS allTotal
from purchasing.PurchaseOrderHeader as O1
Order BY EmployeeID,  PurchaseOrderID
FOR JSON PATH, ROOT ('EmployeeTotalSalesPTY'), INCLUDE_NULL_VALUES
```


```
1  {
2      "EmployeeTotalSalesPTY": [
3          {
4              "employeeId": 250,
5              "PurchaseOrderId": 10,
6              "TotalDue": 1984.6192,
7              "PtyOfTotalEmployeeOrder": "7.93%",
8              "employeeTotal": "$2,501,613.04",
9              "PtyOFAllTotal": "0.28%",
10             "allTotal": "$70,479,332.64"
11         },
12         {
13             "employeeId": 250,
14             "PurchaseOrderId": 21,
15             "TotalDue": 7721.4638,
16             "PtyOfTotalEmployeeOrder": "30.87%",
17             "employeeTotal": "$2,501,613.04",
18             "PtyOFAllTotal": "1.10%",
19             "allTotal": "$70,479,332.64"
20         },
21         {
22             "employeeId": 250,
23             "PurchaseOrderId": 45,
24             "TotalDue": 31160.2541,
25             "PtyOfTotalEmployeeOrder": "124.56%",
26             "employeeTotal": "$2,501,613.04",
27             "PtyOFAllTotal": "4.42%",
28             "allTotal": "$70,479,332.64"
29         },
30         {
31             "employeeId": 250,
32             "PurchaseOrderId": 92,
33             "TotalDue": 284.6209,
34             "PtyOfTotalEmployeeOrder": "1.14%",
35             "employeeTotal": "$2,501,613.04",
36             "PtyOFAllTotal": "0.04%",
37             "allTotal": "$70,479,332.64"
38         },
39         {
```

COMPLEX II

Example of FactInternetSales sub-system in AdventureWorksDW2017

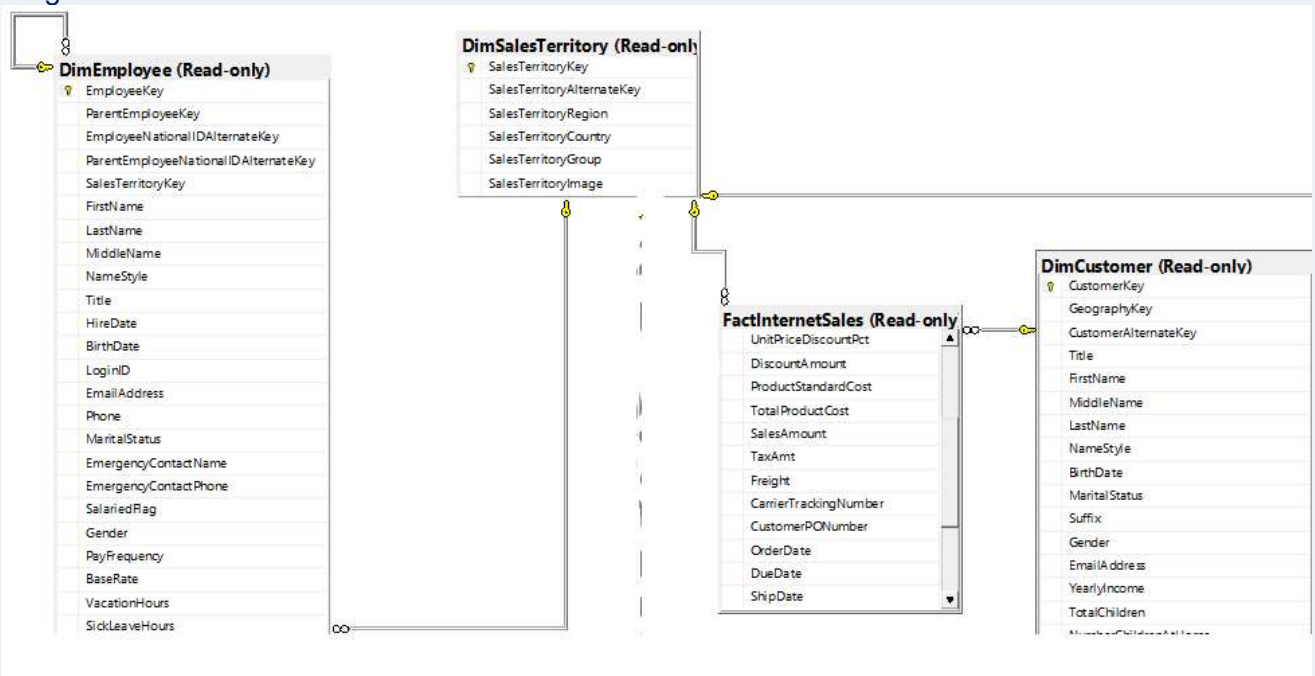


Proposition 06: Our company is having a special discount for families with children. The discount varies by the number of children. Show how much discount each family will receive.

Detailed explanation of the problem that will help the developer to write the query to resolve the issue

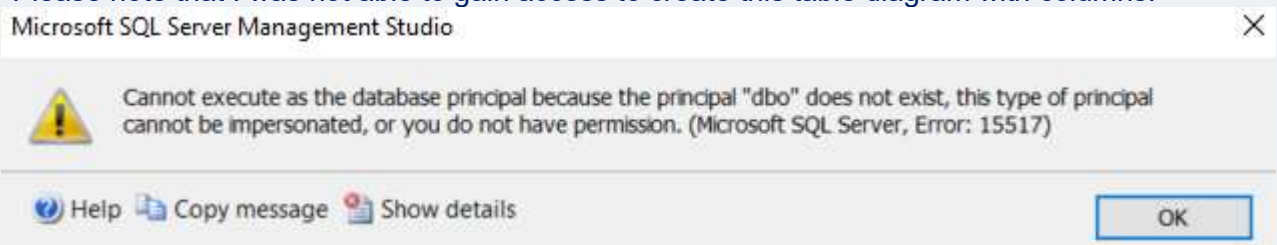
This query will use a scalar function that returns the appropriate discount amount that varies by the customer's number of children and common table expression (CTE) that calculates the total sales amount ordered by each customer. It performs two INNER JOINs to obtain EmployeeKey from FactInternetSales.

Diagram of Tables



Columns from Standard View

*Please note that I was not able to gain access to create this table diagram with columns.



Project following columns from their respective tables in the select clause

Table Name	Column Name
SalesTotal	CustomerKey SalesOrderNumber
SalesTerritory	EmployeeKey
Dbo.dimCustomer	TotalChidlren CustomerKey

Problem Solving Query

```sql
use AdventureWorksDW2017;
GO
CREATE FUNCTION dbo.ChildrenDiscount
(
    @TotalChildren as int
)
RETURNS INT
AS
BEGIN
    DECLARE @Result INT;

    SELECT @Result = CASE
                        WHEN @TotalChildren > 5 THEN 25
                        WHEN @totalChildren BETWEEN 2 AND 4 THEN 15
                        WHEN @TotalChildren > 0 THEN 10
                        ELSE 0

                    END;

    RETURN @Result;
END;
```

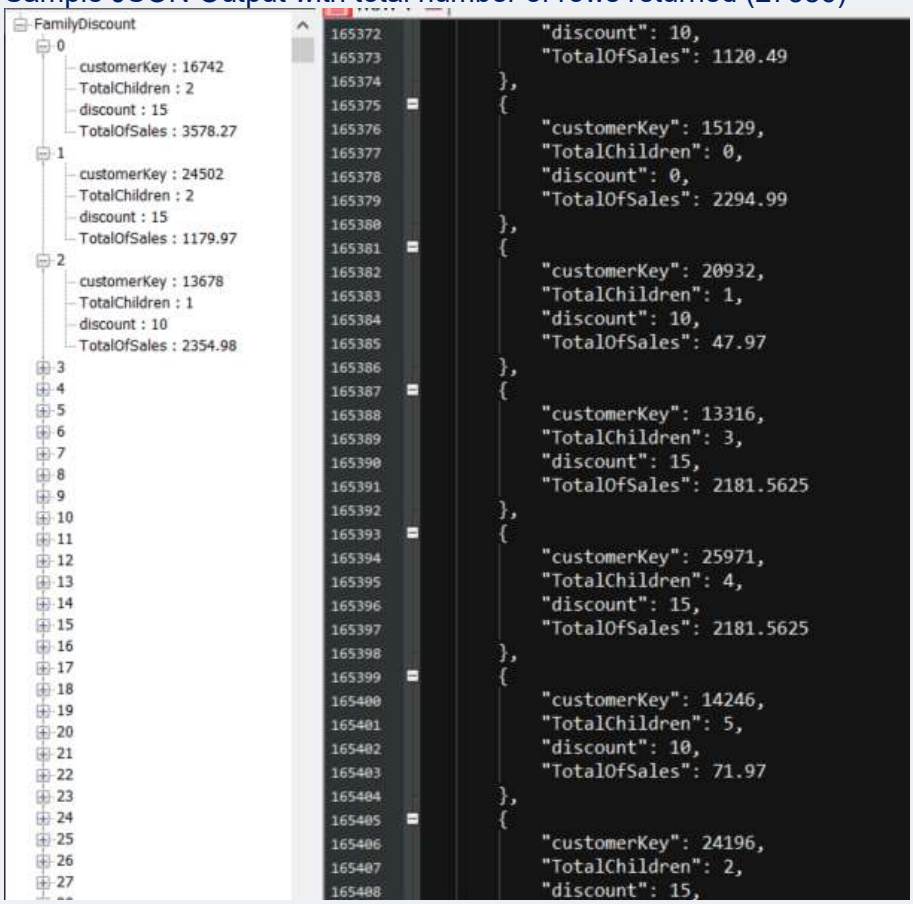
```sql
use AdventureWorksDW2017;
WITH SalesTotal AS
(   SELECT  S1.CustomerKey, S1.SalesOrderNumber,
           (SELECT SUM(S2.SalesAmount)
            FROM dbo.FactInternetSales AS S2
            WHERE S2.SalesOrderNumber = S1.SalesOrderNumber) AS TotalOfSales
            , E.EmployeeKey
    FROM dbo.FactInternetSales AS S1
        INNER JOIN dbo.DimSalesTerritory AS T
            ON T.SalesTerritoryKey = S1.SalesTerritoryKey
        INNER JOIN dbo.DimEmployee AS E
            ON E.SalesTerritoryKey = T.SalesTerritoryKey
)

SELECT DISTINCT C.customerKey, C.TotalChildren,
        dbo.ChildrenDiscount (C.TotalChildren) AS discount
        , S.TotalOfSales
FROM SalesTotal AS S
    INNER JOIN dbo.dimCustomer AS C
        ON C.CustomerKey = S.CustomerKey
```

Sample Relational Output with total number of rows returned (27586)

Results | Messages

	customerKey	TotalChildren	discount	TotalOfSales
1	16742	2	15	3578.27
2	24502	2	15	1179.97
3	13678	1	10	2354.98
4	28974	3	15	24.99
5	26267	2	15	21.49
6	12431	1	10	32.28
7	16820	5	10	876.33
8	27343	0	0	132.97
9	18407	2	15	32.28
10	15908	2	15	630.94
11	16367	5	10	39.98
12	11050	3	15	777.34
13	27829	0	0	39.98
14	13829	0	0	3578.27
15	12552	1	10	8.99
16	11166	0	0	47.97
17	28039	2	15	3578.27
18	27596	5	10	64.47
19	18975	5	10	89.97
20	20824	1	10	782.99
21	26638	1	10	2049.0982
22	25869	2	15	79.97
23	14838	0	0	836.45
24	14434	1	10	1000.4375
25	15826	0	0	120.48

Sample JSON Output with total number of rows returned (27586)



CORRECTION

SIMPLE

Problem Query (Proposition 04 Above)

```
use Northwinds2022TSQLV7
drop view if exists Utils.ownersAndAgents;
go
create VIEW Utils.ownersAndAgents
as

select CustomerId, CustomerContactTitle, CustomerCity, CustomerRegion, CustomerCountry
from sales.Customer
where CustomerContactTitle IN (N'Owner', N'Sales Agent');

go
```

```
use Northwinds2022TSQLV7
SELECT oa.CustomerContactTitle, oa.CustomerId
       , coalesce (oa.CustomerRegion, N'No Region')  as CustomerRegion
       ,Row_Number() over (partition by oa.CustomerContactTitle order by oa.customerID) as rowNum
       ,concat(oa.CustomerContactTitle,(Row_Number() over (partition by oa.CustomerContactTitle order by oa.customerID))) as newLabel
       ,c.CustomerPostalCode
from Utils.ownersAndAgents as oa
inner join Sales.Customer as c
    on c.customerId = oa.customerId
```

Correction of the Problem Query

Instead of invoking the same table inside the table expression then joining the original table to the table expression, the only the original table is invoked once in the query. The MEDIUM complexity is now reduced to a SIMPLE query.

Project following columns from their respective tables in the select clause

Table Name	Column Name
Sales.Customer	CustomerContactTitle CustomerId CustomerRegion CustomerPostalCode
Derived Column	CustomerRegion rowNum newLabel

Order By

Table Name	Column Name	Sort Order
Utils.OwnersAndAgents (VIEW)	CustomerContactTitle	ASC

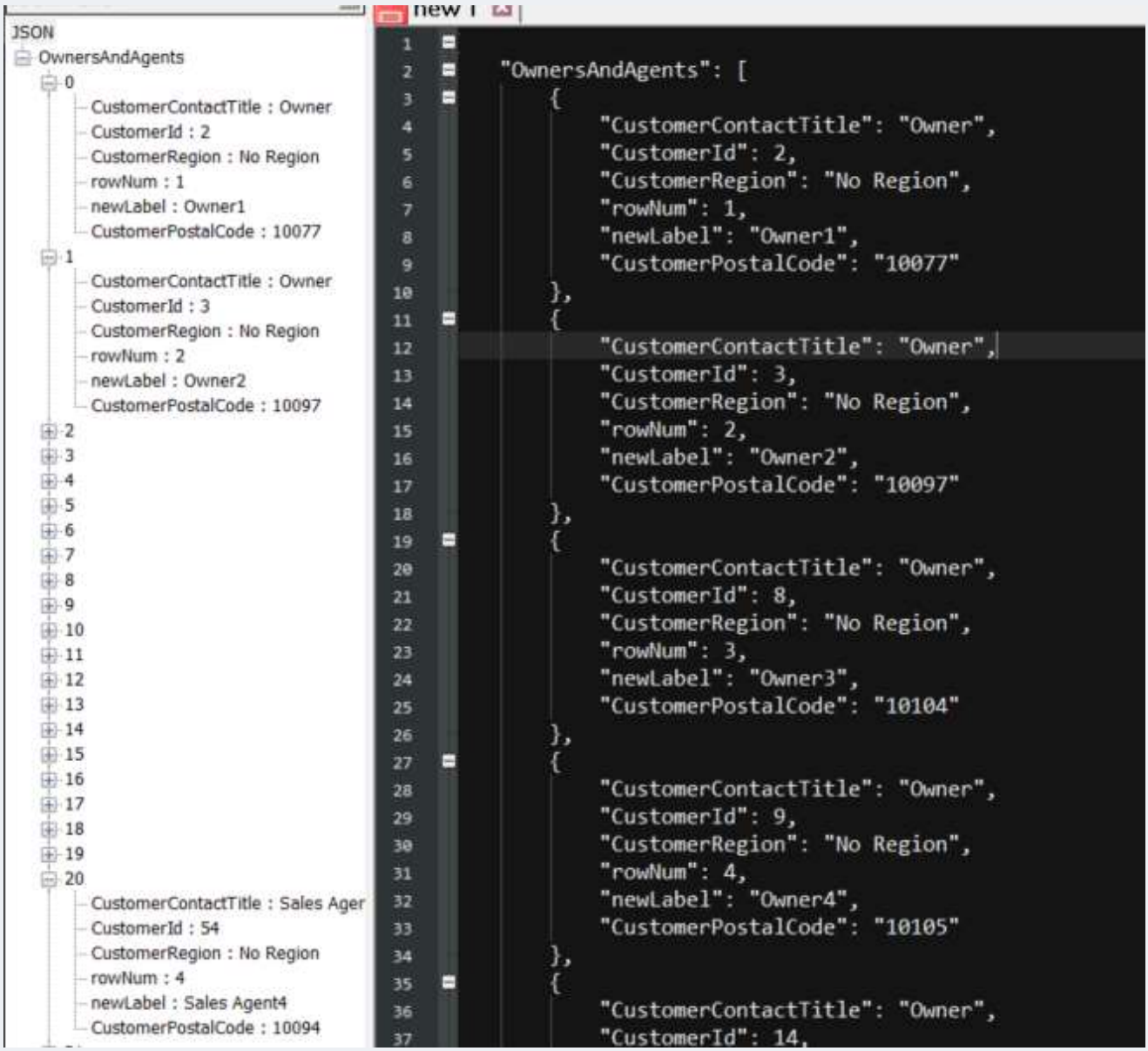
Problem Solving Query

```
use Northwinds2022TSQLV7
SELECT CustomerContactTitle, CustomerId
        , coalesce (CustomerRegion, N'No Region') as CustomerRegion
        ,Row_Number() over (partition by CustomerContactTitle order by customerID) as rowNum
        ,concat(CustomerContactTitle,(Row_Number() over (partition by CustomerContactTitle order by customerID))) as newLabel
        ,CustomerPostalCode
FROM Sales.Customer
WHERE CustomerContactTitle IN (N'Owner', N'Sales Agent')
ORDER BY CustomerContactTitle
```

Sample Relational Output with total number of rows returned (22)

	CustomerContactTitle	CustomerId	CustomerRegion	rowNum	newLabel	CustomerPostalCode
1	Owner	2	No Region	1	Owner1	10077
2	Owner	3	No Region	2	Owner2	10097
3	Owner	8	No Region	3	Owner3	10104
4	Owner	9	No Region	4	Owner4	10105
5	Owner	14	No Region	5	Owner5	10065
6	Owner	18	No Region	6	Owner6	10041
7	Owner	24	No Region	7	Owner7	10114
8	Owner	33	DF	8	Owner8	10043
9	Owner	45	CA	9	Owner9	10062
10	Owner	47	Nueva Esparta	10	Owner10	10121
11	Owner	56	No Region	11	Owner11	10047
12	Owner	57	No Region	12	Owner12	10085
13	Owner	70	No Region	13	Owner13	10123
14	Owner	73	No Region	14	Owner14	10079
15	Owner	80	No Region	15	Owner15	10044
16	Owner	89	WA	16	Owner16	10049
17	Owner	91	No Region	17	Owner17	10068
18	Sales Agent	12	No Region	1	Sales Agent1	10057
19	Sales Agent	19	No Region	2	Sales Agent2	10110
20	Sales Agent	50	No Region	3	Sales Agent3	10074
21	Sales Agent	54	No Region	4	Sales Agent4	10094
22	Sales Agent	84	No Region	5	Sales Agent5	10072

Sample JSON Output with total number of rows returned (22)



COMPLEX I

Problem Query (Proposition 05 Above)

```
use AdventureWorks2017
select employeeId, PurchaseOrderId, TotalDue
       ,  FORMAT((100. * O1.TotalDue/(select sum(O2.TotalDue)
                              from purchasing.PurchaseOrderHeader as O2
                              where O2.EmployeeID = O1.EmployeeID)
          ),'P') as PtyOfTotalEmployeeOrder
        , FORMAT((select sum(O2.TotalDue)
           from purchasing.PurchaseOrderHeader as O2
           where O2.EmployeeID = O1.EmployeeID),'C')as employeeTotal
        , Format (100. * O1.TotalDue/ (SELECT SUM(O2.TotalDue) FROM purchasing.PurchaseOrderHeader as O2),'P') AS PtyOFAllTotal
        , FORMAT((SELECT SUM(O2.TotalDue) FROM purchasing.PurchaseOrderHeader as O2), 'C') AS allTotal
from purchasing.PurchaseOrderHeader as O1
Order BY EmployeeID,  PurchaseOrderID;
```

Correction of the Problem Query- Detailed Explanation

The subquery that returns the total sales serviced by each employee makes the code difficult to read because it takes up multiple lines. Moreover, the same query is used twice. So I decided to make a scalar function that returns the totalSales amount for employee specified. By passing in the EmployeeId column variable instead of a constant, the function outputs all total sales for all employes. The use of the scalar function reduced several lines from the original query.

Project following columns from their respective tables in the select clause

Table Name	Column Name
Dbo.EmployeeTotalSales (Scalar Function)	TotalDue
Purchasing.PurchaseOrderHeader	EmployeeId PurchaseOrderId TotalDue
Derived Column	PtyOfTotalEmployeeOrder employeeTotal PtyOfAllTotal allTotal

Order By

Table Name	Column Name	Sort Order
Purchasing.PurchaseOrderHeader	EmployeeId	ASC
Purchasing.PurchaseOrderHeader	PurchaseOrderId	ASC

Problem Solving Query

```sql
DROP FUNCTION IF EXISTS dbo.EmployeeTotalSales;
GO
CREATE FUNCTION dbo.EmployeeTotalSales
(
    @EmployeeId AS INT
)
RETURNS MONEY
AS
BEGIN

        RETURN (SELECT SUM (TotalDue)
                    FROM Purchasing.PurchaseOrderHeader
                   WHERE EmployeeId = @EmployeeId)
END
GO
```

```sql
use AdventureWorks2017
select employeeId, PurchaseOrderId, TotalDue
       ,  FORMAT((100. * O1.TotalDue/dbo.EmployeeTotalSales(O1.EmployeeId)),'P') as PtyOfTotalEmployeeOrder
       , FORMAT(dbo.EmployeeTotalSales(O1.EmployeeId),'C')as employeeTotal
       , Format (100. * O1.TotalDue/ (SELECT SUM(O2.TotalDue) FROM purchasing.PurchaseOrderHeader as O2),'P') AS PtyOFAllTotal
       , FORMAT((SELECT SUM(O2.TotalDue) FROM purchasing.PurchaseOrderHeader as O2), 'C') AS allTotal
from purchasing.PurchaseOrderHeader as O1
Order BY EmployeeID,  PurchaseOrderID;
```

Sample Relational Output with total number of rows returned (4012)

	employeeId	PurchaseOrderId	TotalDue	PtyOfTotalEmployeeOrder	employeeTotal	PtyOFAllTotal	allTotal
1	250	10	1984.6192	7.93%	$2,501,613.04	0.28%	$70,479,332.64
2	250	21	7721.4638	30.87%	$2,501,613.04	1.10%	$70,479,332.64
3	250	45	31160.2541	124.56%	$2,501,613.04	4.42%	$70,479,332.64
4	250	92	284.6209	1.14%	$2,501,613.04	0.04%	$70,479,332.64
5	250	110	157.3647	0.63%	$2,501,613.04	0.02%	$70,479,332.64
6	250	121	38281.8686	153.03%	$2,501,613.04	5.43%	$70,479,332.64
7	250	145	731.6189	2.92%	$2,501,613.04	0.10%	$70,479,332.64
8	250	192	1410.2839	5.64%	$2,501,613.04	0.20%	$70,479,332.64
9	250	210	126.4905	0.51%	$2,501,613.04	0.02%	$70,479,332.64
10	250	221	590.9618	2.36%	$2,501,613.04	0.08%	$70,479,332.64
11	250	245	525.628	2.10%	$2,501,613.04	0.07%	$70,479,332.64
12	250	292	462.9398	1.85%	$2,501,613.04	0.07%	$70,479,332.64
13	250	310	1043.5289	4.17%	$2,501,613.04	0.15%	$70,479,332.64
14	250	321	22539.0165	90.10%	$2,501,613.04	3.20%	$70,479,332.64
15	250	345	458.4844	1.83%	$2,501,613.04	0.07%	$70,479,332.64
16	250	392	41817.1504	167.16%	$2,501,613.04	5.93%	$70,479,332.64
17	250	410	113.3332	0.45%	$2,501,613.04	0.02%	$70,479,332.64
18	250	420	3758.6299	15.02%	$2,501,613.04	0.53%	$70,479,332.64
19	250	438	2032.6535	8.13%	$2,501,613.04	0.29%	$70,479,332.64
20	250	449	8423.415	33.67%	$2,501,613.04	1.20%	$70,479,332.64
21	250	473	10828.6133	43.29%	$2,501,613.04	1.54%	$70,479,332.64
22	250	520	166.6235	0.67%	$2,501,613.04	0.02%	$70,479,332.64
23	250	542	1410.2839	5.64%	$2,501,613.04	0.20%	$70,479,332.64
24	250	553	31160.2541	124.56%	$2,501,613.04	4.42%	$70,479,332.64
25	250	577	100685.3348	402.48%	$2,501,613.04	14.29%	$70,479,332.64

Sample JSON Output with total number of rows returned (4012)

```sql
use AdventureWorks2017
select employeeId, PurchaseOrderId, TotalDue
        ,  FORMAT((100. * O1.TotalDue/dbo.EmployeeTotalSales(O1.EmployeeId)),'P') as PtyOfTotalEmployeeOrder
        , FORMAT(dbo.EmployeeTotalSales(O1.EmployeeId),'C')as employeeTotal
        , Format (100. * O1.TotalDue/ (SELECT SUM(O2.TotalDue) FROM purchasing.PurchaseOrderHeader as O2),'P') AS PtyOFAllTotal
        , FORMAT((SELECT SUM(O2.TotalDue) FROM purchasing.PurchaseOrderHeader as O2), 'C') AS allTotal
from purchasing.PurchaseOrderHeader as O1
Order BY EmployeeID,  PurchaseOrderID
FOR JSON PATH, ROOT ('EmployeeTotalSalesPTY'), INCLUDE_NULL_VALUES;
```


```
1
2   "EmployeeTotalSalesPTY": [
3       {
4           "employeeId": 250,
5           "PurchaseOrderId": 10,
6           "TotalDue": 1984.6192,
7           "PtyOfTotalEmployeeOrder": "7.93%",
8           "employeeTotal": "$2,501,613.04",
9           "PtyOFAllTotal": "0.28%",
10          "allTotal": "$70,479,332.64"
11      },
12      {
13          "employeeId": 250,
14          "PurchaseOrderId": 21,
15          "TotalDue": 7721.4638,
16          "PtyOfTotalEmployeeOrder": "30.87%",
17          "employeeTotal": "$2,501,613.04",
18          "PtyOFAllTotal": "1.10%",
19          "allTotal": "$70,479,332.64"
20      },
21      {
22          "employeeId": 250,
23          "PurchaseOrderId": 45,
24          "TotalDue": 31160.2541,
25          "PtyOfTotalEmployeeOrder": "124.56%",
26          "employeeTotal": "$2,501,613.04",
27          "PtyOFAllTotal": "4.42%",
28          "allTotal": "$70,479,332.64"
29      },
30      {
31          "employeeId": 250,
32          "PurchaseOrderId": 92,
33          "TotalDue": 284.6209,
34          "PtyOfTotalEmployeeOrder": "1.14%",
35          "employeeTotal": "$2,501,613.04",
36          "PtyOFAllTotal": "0.04%",
37          "allTotal": "$70,479,332.64"
38      },
39      {
```

COMPLEX II

Problem Query (Proposition 06 Above)

```
use AdventureWorksDW2017;
GO
CREATE FUNCTION dbo.ChildrenDiscount
(
    @TotalChildren as int
)
RETURNS INT
AS
BEGIN
    DECLARE @Result INT;

    SELECT @Result = CASE
                        WHEN @TotalChildren > 5 THEN 25
                        WHEN @totalChildren BETWEEN 2 AND 4 THEN 15
                        WHEN @TotalChildren > 0 THEN 10
                        ELSE 0

                    END;

    RETURN @Result;
END;
```

```
use AdventureWorksDW2017;
WITH SalesTotal AS
(   SELECT  S1.CustomerKey, S1.SalesOrderNumber,
           (SELECT SUM(S2.SalesAmount)
            FROM dbo.FactInternetSales AS S2
            WHERE S2.SalesOrderNumber = S1.SalesOrderNumber) AS TotalOfSales
            , E.EmployeeKey
    FROM dbo.FactInternetSales AS S1
        INNER JOIN dbo.DimSalesTerritory AS T
            ON T.SalesTerritoryKey = S1.SalesTerritoryKey
        INNER JOIN dbo.DimEmployee AS E
            ON E.SalesTerritoryKey = T.SalesTerritoryKey
)

SELECT DISTINCT C.customerKey, C.TotalChildren,
        dbo.ChildrenDiscount (C.TotalChildren) AS discount
        , S.TotalOfSales
FROM SalesTotal AS S
    INNER JOIN dbo.dimCustomer AS C
        ON C.CustomerKey = S.CustomerKey
```

Revised Proposition: Apply family discount of varying amount depending on the number of children for orders placed on May 3rd, 2013.

Correction of the Problem Query- Detailed Explanation

First of all, the proposition of this query was not practical because it attempted to give discounts to families with children for all orders they have ever purchased. The company will lose so much money like that. So I had to specify the proposition to apply the discount on a single day: May 3rd, 2015. The two INNER JOINs inside the Common Table Expression was not necessary for EmployeeKey is not a necessary information to figure out how much discount the family will receive on the orders placed on 05/03/2015.

There was a special challenge working with this database because it only had Sales Details table without the Sales summary table like Sales.Order in NorthWinds. When trying to calculate the total Orders per each customer, duplicates printed because there are multiple product details for each order. I had to use the DISTINCT clause to rule these out.

I was also making a grave mistake in the subquery's WHERE clause. I was incorrectly obtaining total sales for each order instead of each customer! Using the SalesOrderID in the WHERE clause is a guaranteed way to obtain duplicate information even though I still had to use the DINTINCT clause in the outer query.

While trying to specify the query to only obtain the total for only 05/03/2015, I first tried to specify that in the outer query of SalesTotalCTE only. Even though the date was specified on the outer query, the inner query was calculating the total sum of all orders the customer purchased, not just the ones on 05/03/2015. So I had to add a date filter within the subquery of the CTE as well.

The original query simply output the discount percentage and did not calculate the actual amount of money the customers will receive discount on. So it did not fulfill the proposition. So I added that column. This corrected query seems more succinct and gathers accurate data.

Project following columns from their respective tables in the select clause

Table Name	Column Name
SalesTotalCTE	CustomerKey SalesAmount OrderDate
Dbo.dimCustomer	CustomerKey TotalChildren
Derived Column	TotalOfSalesPerCustomer discountPty totalSalesOn20130503 discountedAmount

Order By

Table Name	Column Name	Sort Order
Dbo.dimCustomer	CustomerKey	ASC

Problem Solving Query

```
use AdventureWorksDW2017;
GO
DROP FUNCTION IF EXISTS dbo.ChildrenDiscount;
GO
use AdventureWorksDW2017;
go
CREATE FUNCTION dbo.ChildrenDiscount
(
    @TotalChildren as int
)
RETURNS DECIMAL
AS
BEGIN
    DECLARE @Result DECIMAL;

    SELECT @Result = CASE
                        WHEN @TotalChildren > 5 THEN 25
                        WHEN @totalChildren BETWEEN 2 AND 4 THEN 15
                        WHEN @TotalChildren > 0 THEN 10
                        ELSE 0

                    END;

    RETURN @Result;
END;
```

```
use AdventureWorksDW2017;
WITH SalesTotalCTE AS
(   SELECT  DISTINCT S1.CustomerKey,
           (SELECT SUM(S2.SalesAmount)
            FROM dbo.FactInternetSales AS S2
            WHERE S2.CustomerKey = S1.CustomerKey
            AND S2.OrderDate = '20130503') AS TotalOfSalesPerCustomer
    FROM dbo.FactInternetSales AS S1
    WHERE S1.OrderDate = '20130503'

)

SELECT DISTINCT C.customerKey, C.TotalChildren,
        dbo.ChildrenDiscount (C.TotalChildren) AS discountPty
        , FORMAT(S.TotalOfSalesPerCustomer, 'C') AS totalSalesOn20130503
        , FORMAT ((dbo.ChildrenDiscount (C.TotalChildren)/100 )* S.TotalOfSalesPerCustomer, 'C') AS discountedAmount
FROM SalesTotalCTE AS S
    INNER JOIN dbo.dimCustomer AS C
        ON C.CustomerKey = S.CustomerKey
ORDER BY C.customerKey
```

Sample Relational Output with total number of rows returned (57)

	customerKey	TotalChildren	discountPty	totalSalesOn20130503	discountedAmount
1	11000	2	15	$2,507.03	$376.05
2	11172	1	10	$38.98	$3.90
3	11212	2	15	$52.28	$7.84
4	11432	4	15	$1,283.82	$192.57
5	11504	1	10	$42.28	$4.23
6	11792	1	10	$35.00	$3.50
7	11868	0	0	$14.98	$0.00
8	12247	0	0	$27.28	$0.00
9	12447	1	10	$2,331.95	$233.20
10	12689	0	0	$2,331.95	$0.00
11	13099	3	15	$120.00	$18.00
12	13231	2	15	$74.98	$11.25
13	13280	5	10	$2,376.96	$237.70
14	13303	4	15	$42.28	$6.34
15	13367	4	15	$2,334.97	$350.25
16	13941	1	10	$38.88	$3.89
17	13974	4	15	$796.34	$119.45
18	14076	0	0	$71.97	$0.00
19	14134	0	0	$1,785.97	$0.00
20	14928	1	10	$751.34	$75.13
21	15013	0	0	$26.97	$0.00
22	15137	0	0	$63.97	$0.00
23	15829	2	15	$24.99	$3.75
24	15833	2	15	$539.99	$81.00

Sample JSON Output with total number of rows returned (57)