# Project 1 - Team 3 - Damian Rozpedowski

## \===== Top 3 Best Problems =====

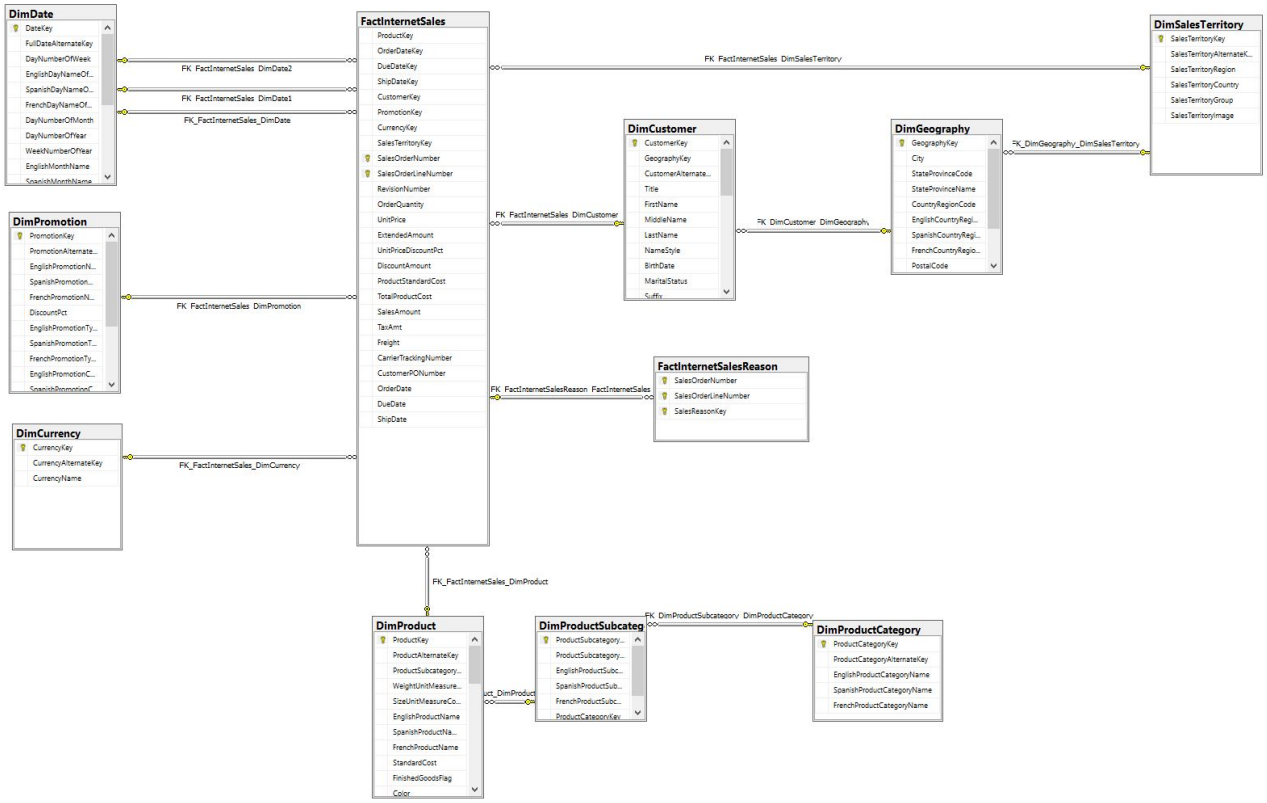
**Medium Proposition 1**: Determine the annual sales performance by region and country in the AdventureWorksDW2017 database, categorizing each territory based on total sales thresholds.

- **Tables Used:** Join `FactInternetSales` (`fis`) for sales data, `DimSalesTerritory` (`dst`) for region and country, and `DimDate` (`dd`) for year.
- **SQL Functions:** Utilize `SUM()`, `AVG()`, and `COUNT(DISTINCT)` to compute total sales, average sales per order, and order counts, respectively.
- **Aggregation:** Group by `dst.SalesTerritoryRegion`, `dst.SalesTerritoryCountry`, and `dd.CalendarYear`.
- **CTE Usage:** Create a `TerritorySales` CTE to aggregate sales metrics and classify performance as 'High' (>1,000,000), 'Medium' (500,000–1,000,000), or 'Low' (<500,000) sales.
- **Output:** Display year, region, country, sales metrics, and performance level, ordered by year and descending total sales.

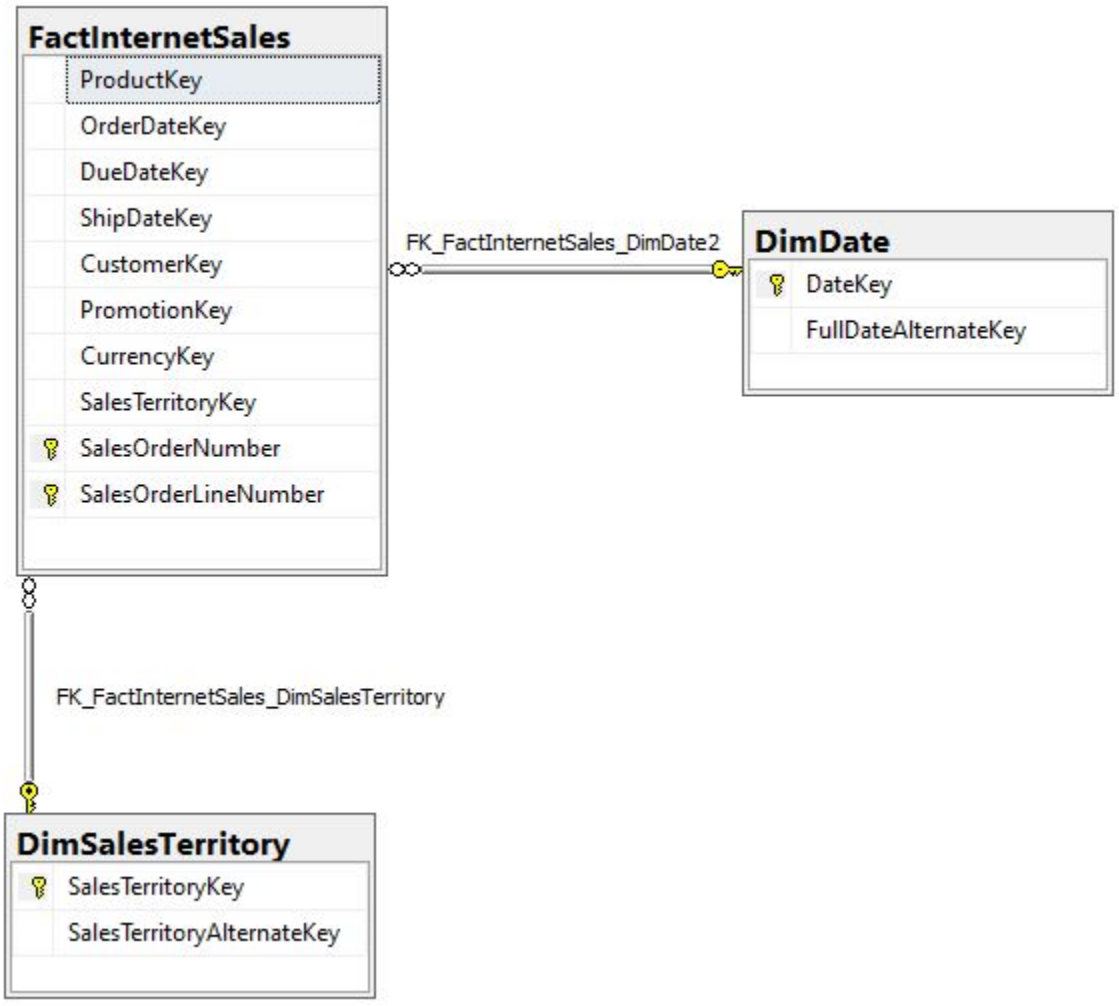
#### Why is this a top problem?

The query adheres to the criteria for a medium complexity SQL query by effectively joining 2 to 3 tables, leveraging built-in functions for summarization, and utilizing CTEs for complex data organization. The output, which reveals performance across territories, is particularly impressive for its practical insights and relevance, it provides meaningful information and is overall a very relevant proposition.

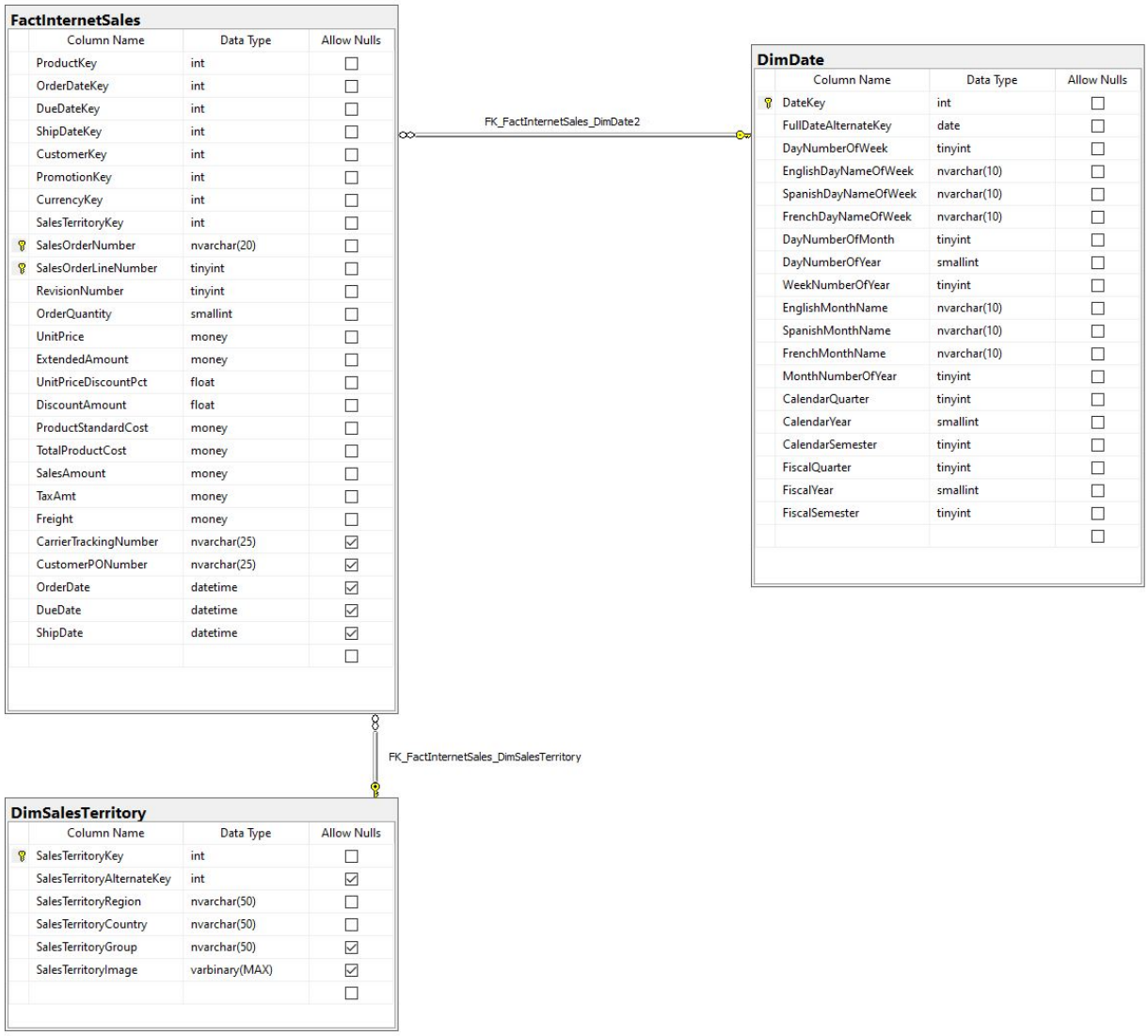
##### Subsystem in AdventureWorksDW2017




## Diagram of Tables



**Columns from Standard view**




# Table: Column Projection

| Table Name | Column Name | Derived Column |
|---|---|---|
| DimSalesTerritory | SalesTerritoryRegion | |
| DimSalesTerritory | SalesTerritoryCountry | |
| DimDate | CalendarYear | SaleYear |
| FactInternetSales | SalesAmount | TotalSales (SUM) |
| FactInternetSales | SalesAmount | AvgSalesPerOrder (AVG) |
| FactInternetSales | SalesOrderNumber | NumberOfOrders (COUNT DISTINCT) |

# Table: Order By

| Table Name | Column Name | Sort Order |
|---|---|---|
| DimDate | CalendarYear | ASC |
| (Derived) | TotalSales | DESC |

In [6]:
```
USE AdventureWorksDW2017;
GO

WITH TerritorySales AS (
    SELECT
        dst.SalesTerritoryRegion AS TerritoryRegion,
        dst.SalesTerritoryCountry AS TerritoryCountry,
        dd.CalendarYear AS SaleYear,
        SUM(fis.SalesAmount) AS TotalSales,
        AVG(fis.SalesAmount) AS AvgSalesPerOrder,
        COUNT(DISTINCT fis.SalesOrderNumber) AS NumberOfOrders
    FROM
        FactInternetSales fis
    JOIN DimSalesTerritory dst ON fis.SalesTerritoryKey = dst.SalesTerritoryKey
    JOIN DimDate dd ON fis.OrderDateKey = dd.DateKey
    GROUP BY
        dst.SalesTerritoryRegion,
        dst.SalesTerritoryCountry,
        dd.CalendarYear
)
SELECT
    SaleYear,
    TerritoryRegion,
    TerritoryCountry,
    TotalSales,
    AvgSalesPerOrder,
    NumberOfOrders,
    CASE
        WHEN TotalSales > 1000000 THEN 'High Performance'
        WHEN TotalSales BETWEEN 500000 AND 1000000 THEN 'Medium Performance'
        ELSE 'Low Performance'
    END AS PerformanceLevel
FROM
    TerritorySales
ORDER BY
    SaleYear,
    TotalSales DESC;
```

Commands completed successfully.

(41 rows affected)

Total execution time: 00:00:00.242

Out[6]:

| SaleYear | TerritoryRegion | TerritoryCountry | TotalSales | AvgSalesPerOrder | NumberOfOrders | PerformanceLeve |
|---|---|---|---|---|---|---|
| 2010 | Australia | Australia | 20909.78 | 3484.9633 | 6 | Low Performanc |
| 2010 | Southwest | United States | 11433.9082 | 2858.477 | 4 | Low Performanc |
| 2010 | Canada | Canada | 3578.27 | 3578.27 | 1 | Low Performanc |
| 2010 | France | France | 3399.99 | 3399.99 | 1 | Low Performanc |
| 2010 | Northwest | United States | 3399.99 | 3399.99 | 1 | Low Performanc |
| 2010 | United Kingdom | United Kingdom | 699.0982 | 699.0982 | 1 | Low Performanc |
| 2011 | Australia | Australia | 2563732.2493 | 3261.7458 | 786 | High Performanc |
| 2011 | Southwest | United States | 1531228.4113 | 3183.427 | 481 | High Performanc |
| 2011 | Northwest | United States | 927056.7613 | 3207.8088 | 289 | Mediur Performanc |
| 2011 | Canada | Canada | 571571.7984 | 3362.187 | 170 | Mediur Performanc |
| 2011 | United Kingdom | United Kingdom | 550591.2186 | 3146.2355 | 175 | Mediur Performanc |
| 2011 | Germany | Germany | 520500.1642 | 2974.2866 | 175 | Mediur |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | Performanc |
| 2011 | France | France | 410845.326 | 2934.6094 | 140 | Low Performanc |
| 2012 | Australia | Australia | 2128407.4551 | 1823.8281 | 1130 | High Performanc |
| 2012 | Southwest | United States | 810222.3327 | 1558.1198 | 497 | Mediur Performanc |
| 2012 | United Kingdom | United Kingdom | 712700.9641 | 1684.8722 | 405 | Mediur Performanc |
| 2012 | France | France | 648065.5383 | 1737.4411 | 359 | Mediur Performanc |
| 2012 | Northwest | United States | 619068.4799 | 1677.6923 | 365 | Mediur Performanc |
| 2012 | Germany | Germany | 608657.984 | 1653.9619 | 339 | Mediur Performanc |
| 2012 | Canada | Canada | 307604.5237 | 1788.3983 | 169 | Low Performanc |
| 2012 | Southeast | United States | 3637.3996 | 1212.4665 | 3 | Low Performanc |
| 2012 | Central | United States | 2071.4196 | 2071.4196 | 1 | Low Performanc |
| 2012 | Northeast | United States | 2049.0982 | 2049.0982 | 1 | Low Performanc |
| 2013 | Australia | Australia | 4339443.38 | 392.5677 | 4640 | High Performanc |
| 2013 | Southwest | United States | 3356890.10 | 308.2826 | 4319 | High Performanc |
| 2013 | United Kingdom | United Kingdom | 2124007.29 | 346.2114 | 2377 | High Performanc |
| 2013 | Northwest | United States | 2091249.51 | 262.5548 | 3242 | High Performanc |
| 2013 | Germany | Germany | 1761876.36 | 356.8718 | 1909 | High Performanc |
| 2013 | France | France | 1578511.80 | 323.4655 | 1917 | High Performanc |
| 2013 | Canada | Canada | 1085632.65 | 158.1863 | 2856 | High Performanc |
| 2013 | Southeast | United States | 8526.47 | 258.3778 | 12 | Low Performanc |
| 2013 | Northeast | United States | 4483.37 | 172.4373 | 9 | Low Performanc |
| 2013 | Central | United States | 929.41 | 48.9163 | 8 | Low Performanc |
| 2014 | Canada | Canada | 9457.62 | 22.8444 | 179 | Low Performanc |
| 2014 | Northwest | United States | 9091.81 | 24.639 | 161 | Low Performanc |
| 2014 | Australia | Australia | 8507.72 | 25.6256 | 156 | Low Performanc |
| 2014 | Southwest | United States | 8376.06 | 22.5769 | 172 | Low Performanc |
| 2014 | United Kingdom | United Kingdom | 3713.64 | 21.5909 | 73 | Low Performanc |
| 2014 | Germany | Germany | 3277.83 | 22.6057 | 61 | Low Performanc |
| 2014 | France | France | 3195.06 | 19.482 | 67 | Low Performanc |
| 2014 | Southeast | United States | 74.98 | 24.9933 | 2 | Low Performanc |

In [7]:

```
USE AdventureWorksDW2017;
GO

WITH TerritorySales AS (
    SELECT
        dst.SalesTerritoryRegion AS TerritoryRegion,
        dst.SalesTerritoryCountry AS TerritoryCountry,
        dd.CalendarYear AS SaleYear,
        SUM(fis.SalesAmount) AS TotalSales,
        AVG(fis.SalesAmount) AS AvgSalesPerOrder,
        COUNT(DISTINCT fis.SalesOrderNumber) AS NumberOfOrders
```

```
    FROM
        FactInternetSales fis
    JOIN DimSalesTerritory dst ON fis.SalesTerritoryKey = dst.SalesTerritoryKey
    JOIN DimDate dd ON fis.OrderDateKey = dd.DateKey
    GROUP BY
        dst.SalesTerritoryRegion,
        dst.SalesTerritoryCountry,
        dd.CalendarYear
)
SELECT
    SaleYear,
    TerritoryRegion,
    TerritoryCountry,
    TotalSales,
    AvgSalesPerOrder,
    NumberOfOrders,
    CASE
        WHEN TotalSales > 1000000 THEN 'High Performance'
        WHEN TotalSales BETWEEN 500000 AND 1000000 THEN 'Medium Performance'
        ELSE 'Low Performance'
    END AS PerformanceLevel
FROM
    TerritorySales
ORDER BY
    SaleYear,
    TotalSales DESC
FOR JSON PATH, ROOT('TerritoryPerformance');
```

Commands completed successfully.

(41 rows affected)

Total execution time: 00:00:00.308

Out[7]:


```
[{"SaleYear":2010,"TerritoryRegion":"Australia","TerritoryCountry":"Australia","TotalSales":20909.7800,"AvgSalesPerOrder"
Performance"},{"SaleYear":20
States","TotalSales":11433.9082,"AvgSalesPerOrder":2858.4770,"Nu
{"SaleYear":2010,"TerritoryRegion":"Canada","TerritoryCountry":"Canada","TotalSales":3578.2700,"AvgSalesPerOrder"

{"SaleYear":2010,"TerritoryRegion":"France","TerritoryCountry":"France","TotalSales":3399.9900,"AvgSalesPerOrder"
Performance"},{"SaleYear":20
States","TotalSales":3399.9900,"AvgSalesPerOrder":3399.9900,"NumberOfOrders":1,"PerformanceLevel":"Lov
Kingdom","TerritoryCountry":"United Kingdom","TotalSales":699.0982,"AvgSalesPerOrder":699.0982,"Nu
{"SaleYear":2011,"TerritoryRegion":"Australia","TerritoryCountry":"Australia","TotalSales":2563732.2493,"AvgSalesPerOrder":32
Performance"},{"SaleYear":20
States","TotalSales":1531228.4113,"AvgSalesPerOrder":3183.4270,"Numb
{"SaleYear":20
States","TotalSales":927056.7613,"AvgSalesPerOrder":3207.8088,"NumberC
{"SaleYear":2011,"TerritoryRegion":"Canada","TerritoryCountry":"Canada","TotalSales":571571.7984,"AvgSalesPerOrder":3362.
Performance"},{"SaleYear":2011,"T
Kingdom","TotalSales":550591.2186,"AvgSalesPerOrder":3146.2355,"NumberC
{"SaleYear":2011,"TerritoryRegion":"Germany","TerritoryCountry":"Germany","TotalSales":520500.1642,"AvgSalesPerOrder":2974.

{"SaleYear":2011,"TerritoryRegion":"France","TerritoryCountry":"France","TotalSales":410845.3260,"AvgSalesPerOrder":2

{"SaleYear":2012,"TerritoryRegion":"Australia","TerritoryCountry":"Australia","TotalSales":2128407.4551,"AvgSalesPerOrder":182
Performance"},{"SaleYear":20
States","TotalSales":810222.3327,"AvgSalesPerOrder":1558.1198,"NumberOfOrders":497,"PerformanceLevel":"Mediun
Kingdom","TerritoryCountry":"United Kingdom","TotalSales":712700.9641,"AvgSalesPerOrder":1684.8722,"NumberC
{"SaleYear":2012,"TerritoryRegion":"France","TerritoryCountry":"France","TotalSales":648065.5383,"AvgSalesPerOrder":1737.
Performance"},{"SaleYear":20
States","TotalSales":619068.4799,"AvgSalesPerOrder":1677.6923,"NumberC
{"SaleYear":2012,"TerritoryRegion":"Germany","TerritoryCountry":"Germany","TotalSales":608657.9840,"AvgSalesPerOrder":1653.

{"SaleYear":2012,"TerritoryRegion":"Canada","TerritoryCountry":"Canada","TotalSales":307604.5237,"AvgSalesPerOrder":1
Performance"},{"SaleYear":20
States","TotalSales":3637.3996,"AvgSalesPerOrder":1212.4665,"Nu
{"SaleYear":2012,"TerritoryRegion":"Central","TerritoryCountry":"United States","TotalSales":2071.4196,"AvgSalesPerOrder"
Performance"},{"SaleYear":2
```

States","TotalSales":2049.0982,"AvgSalesPerOrder":2049.0982,"Nu
{"SaleYear":2013,"TerritoryRegion":"Australia","TerritoryCountry":"Australia","TotalSales":4339443.3800,"AvgSalesPerOrder":39
Performance"},{"SaleYear":20
States","TotalSales":3356890.1000,"AvgSalesPerOrder":308.2826,"NumberOfOrders":4319,"PerformanceLevel":"High
Kingdom","TerritoryCountry":"United Kingdom","TotalSales":2124007.2900,"AvgSalesPerOrder":346.2114,"Numbe
{"SaleYear":20
States","TotalSales":2091249.5100,"AvgSalesPerOrder":262.5548,"Numbe
{"SaleYear":2013,"TerritoryRegion":"Germany","TerritoryCountry":"Germany","TotalSales":1761876.3600,"AvgSalesPerOrder":35

{"SaleYear":2013,"TerritoryRegion":"France","TerritoryCountry":"France","TotalSales":1578511.8000,"AvgSalesPerOrder":32

{"SaleYear":2013,"TerritoryRegion":"Canada","TerritoryCountry":"Canada","TotalSales":1085632.6500,"AvgSalesPerOrder":15
Performance"},{"SaleYear":20
States","TotalSales":8526.4700,"AvgSalesPerOrder":258.3778,"Num
{"SaleYear":2013,"TerritoryRegion":"Northeast","TerritoryCountry":"United States","TotalSales":4483.3700,"AvgSalesPerOrde
Performance"},{"SaleYear"
States","TotalSales":929.4100,"AvgSalesPerOrder":48.9163,"Nu
{"SaleYear":2014,"TerritoryRegion":"Canada","TerritoryCountry":"Canada","TotalSales":9457.6200,"AvgSalesPerOrder"
Performance"},{"SaleYear":20
States","TotalSales":9091.8100,"AvgSalesPerOrder":24.6390,"Numl
{"SaleYear":2014,"TerritoryRegion":"Australia","TerritoryCountry":"Australia","TotalSales":8507.7200,"AvgSalesPerOrder"
Performance"},{"SaleYear":20
States","TotalSales":8376.0600,"AvgSalesPerOrder":22.5769,"NumberOfOrders":172,"PerformanceLevel":"Lov
Kingdom","TerritoryCountry":"United Kingdom","TotalSales":3713.6400,"AvgSalesPerOrder":21.5909,"Num
{"SaleYear":2014,"TerritoryRegion":"Germany","TerritoryCountry":"Germany","TotalSales":3277.8300,"AvgSalesPerOrde

{"SaleYear":2014,"TerritoryRegion":"France","TerritoryCountry":"France","TotalSales":3195.0600,"AvgSalesPerOrde
Performance"},{"SaleYear":20
States","TotalSales":74.9800,"AvgSalesPerOrder":24.9933,"Nur

**Complex Proposition 1:** Perform a comprehensive annual sales analysis for each product in the AdventureWorks2017 database, accounting for variable discounts based on product ID and order date. The analysis will produce a detailed report showing sales volume, revenue, and the financial impact of discounts on each product over time.

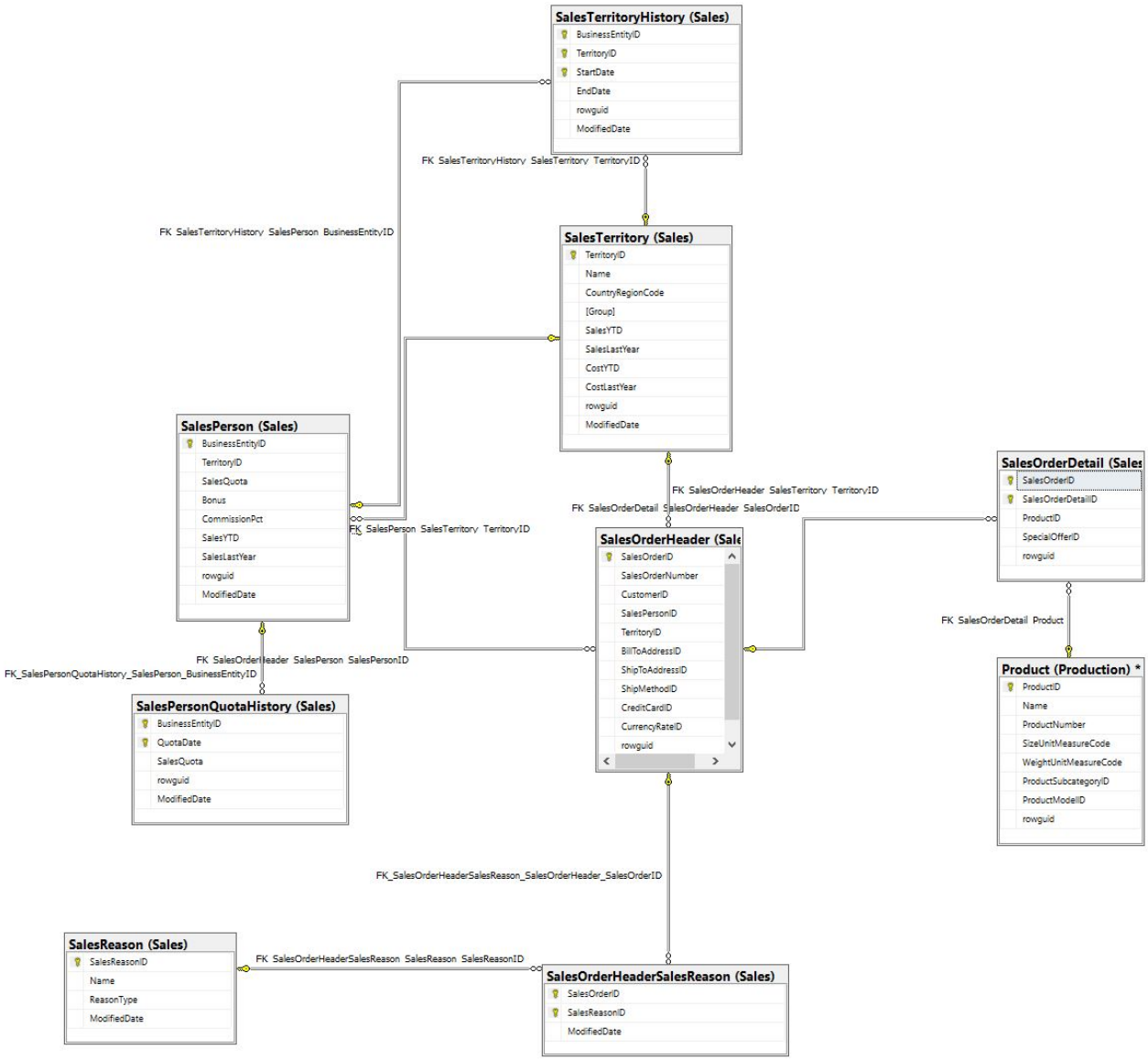
- **Custom Function:** Create `dbo.GetProductDiscount` to compute discounts based on product ID and order date, with 10% off for even product IDs from 2023 onwards, and 5% otherwise.
- **Tables Used:** Join `Sales.SalesOrderHeader` (soh), `Sales.SalesOrderDetail` (sod), and `Production.Product` (p) for sales and product details.
- **SQL Functions:** Leverage `COUNT()`, `SUM()`, `AVG()`, and the custom function to calculate total orders, quantity sold, average unit price, total sales, and total sales with discounts.
- **Aggregation:** Group by year of the order date and product name.
- **CTE Usage:** Employ a `SalesData` CTE to streamline data preparation before final aggregation.
- **Output:** Generate a report displaying the year, product name, total orders, total quantity, average unit price, total sales, and total sales after discounts, ordered by year and product name.

### Why is this a top problem?

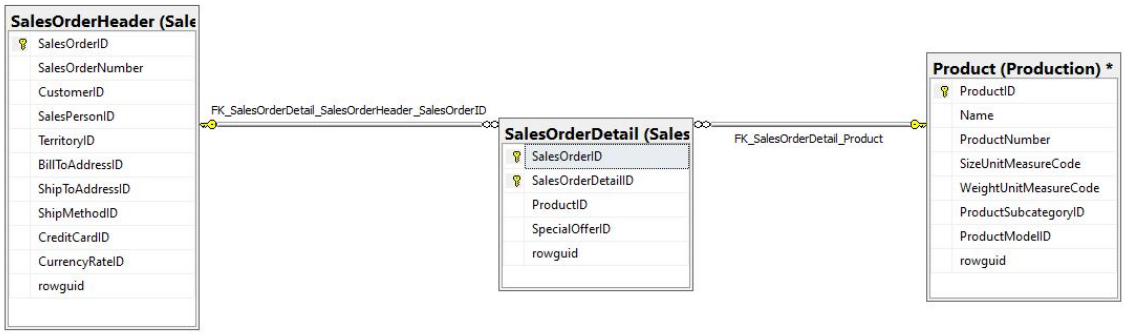
The provided SQL script exemplifies a complex query as per the defined guidelines, which necessitate joining three or more tables, creating a custom scalar function, and employing built-in SQL functions with `GROUP BY` summarization. Additionally, it integrates subqueries or CTEs for advanced data structuring. The script includes a custom scalar function `dbo.GetProductDiscount` to calculate discounts based on product ID and order date. The main query then joins multiple tables, utilizes this custom function within a CTE named `SalesData`, and performs a comprehensive analysis, including counting, summing, and averaging operations, which are then

grouped and ordered to provide a detailed view of sales data by year and product. This approach not only adheres to the complex query guidelines but also offers a great framework for analyzing sales performance and discount impacts.

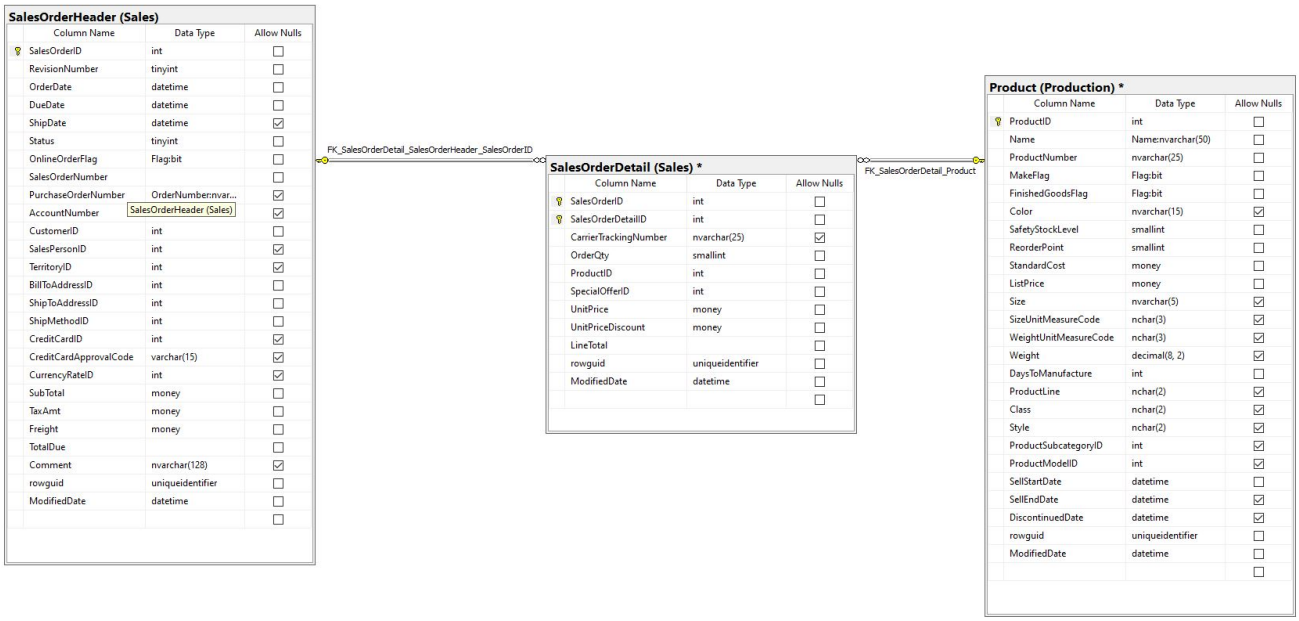
### Subsystem in AdventureWorks2017




### Diagram of Tables




### Columns from Standard view




# Table: Column Projection and Functions

| Table Name | Column Name | Derived Column | Function/Calculation |
|---|---|---|---|
| Sales.SalesOrderHeader | SalesOrderID | | |
| Sales.SalesOrderDetail | ProductID | | |
| Production.Product | Name | ProductName | |
| Sales.SalesOrderHeader | OrderDate | | |
| Sales.SalesOrderDetail | OrderQty | | |
| Sales.SalesOrderDetail | UnitPrice | | |
| Sales.SalesOrderDetail | LineTotal | | |
| | | Discount | dbo.GetProductDiscount() |

# Table: Group By and Order By

| Table Name | Column Name | Aggregation | Sort Order |
|---|---|---|---|
| (Derived) | OrderDate | YEAR(OrderDate) as OrderYear | ASC |
| Production.Product | Name | ProductName | ASC |

| | | |
|---|---|---|
| (Derived) | OrderQty | SUM(OrderQty) |
| (Derived) | UnitPrice | AVG(UnitPrice) |
| (Derived) | LineTotal | SUM(LineTotal) |
| (Derived) | Discount | SUM(LineTotal * (1 - Discount)) |

In [19]:

```
USE AdventureWorks2017;
go
-- Create a custom scalar function
CREATE OR ALTER FUNCTION dbo.GetProductDiscount(@productID INT, @orderDate DATE)
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE @discount DECIMAL(10, 2)

    -- Assume a simple discount calculation based on product ID and order date
    IF @productID % 2 = 0 AND @orderDate >= '2023-01-01'
        SET @discount = 0.1  -- 10% discount for even product IDs in 2023
    ELSE
        SET @discount = 0.05  -- 5% discount for all other cases

    RETURN @discount
end
go

-- Complex query with joins, custom function, built-in SQL functions, and group by summa
WITH SalesData AS (
    SELECT
        soh.SalesOrderID,
        sod.ProductID,
        p.Name AS ProductName,
        soh.OrderDate,
        sod.OrderQty,
        sod.UnitPrice,
        dbo.GetProductDiscount(sod.ProductID, soh.OrderDate) AS Discount,
        sod.LineTotal
    FROM
        Sales.SalesOrderHeader AS soh
    INNER JOIN
        Sales.SalesOrderDetail AS sod ON soh.SalesOrderID = sod.SalesOrderID
    INNER JOIN
        Production.Product AS p ON sod.ProductID = p.ProductID
)

SELECT
    YEAR(OrderDate) AS OrderYear,
    ProductName,
    COUNT(*) AS TotalOrders,
    SUM(OrderQty) AS TotalQuantity,
    AVG(UnitPrice) AS AvgUnitPrice,
    SUM(LineTotal) AS TotalSales,
    SUM(LineTotal * (1 - Discount)) AS TotalSalesWithDiscount
FROM
    SalesData
GROUP BY
    YEAR(OrderDate),
    ProductName
ORDER BY
    OrderYear,
    ProductName;
```

Commands completed successfully.

Commands completed successfully.

(610 rows affected)

Total execution time: 00:00:01.366

Out[19]:

| OrderYear | ProductName | TotalOrders | TotalQuantity | AvgUnitPrice | TotalSales | TotalSalesWithDiscount |
|---|---|---|---|---|---|---|
| 2011 | AWC Logo Cap | 159 | 545 | 5.1832 | 2816.535136 | 2675.708379 |
| 2011 | HL Mountain Frame - Black, 38 | 39 | 73 | 714.7043 | 52173.413900 | 49564.743205 |
| 2011 | HL Mountain Frame - Black, 42 | 47 | 88 | 714.7043 | 62893.978400 | 59749.279480 |
| 2011 | HL Mountain Frame - Black, 44 | 7 | 9 | 809.76 | 7287.840000 | 6923.448000 |
| 2011 | HL Mountain Frame - Black, 48 | 41 | 72 | 809.76 | 58302.720000 | 55387.584000 |
| 2011 | HL Mountain Frame - Silver, 38 | 50 | 91 | 722.5949 | 65756.135900 | 62468.329105 |
| 2011 | HL Mountain Frame - Silver, 42 | 8 | 13 | 722.5949 | 9393.733700 | 8924.047015 |
| 2011 | HL Mountain Frame - Silver, 46 | 41 | 74 | 722.5949 | 53472.022600 | 50798.421470 |
| 2011 | HL Mountain Frame - Silver, 48 | 49 | 90 | 818.70 | 73683.000000 | 69998.850000 |
| 2011 | HL Road Frame - Red, 44 | 7 | 11 | 758.0759 | 8338.834900 | 7921.893155 |
| 2011 | HL Road Frame - Red, 62 | 9 | 13 | 758.0759 | 9854.986700 | 9362.237365 |
| 2011 | LL Road Frame - Black, 44 | 10 | 13 | 178.5808 | 2321.550400 | 2205.472880 |
| 2011 | LL Road Frame - Black, 52 | 91 | 181 | 178.5808 | 32323.124800 | 30706.968560 |
| 2011 | LL Road Frame - Black, 58 | 54 | 112 | 178.5808 | 20001.049600 | 19000.997120 |
| 2011 | LL Road Frame - Black, 60 | 1 | 1 | 178.5808 | 178.580800 | 169.651760 |
| 2011 | LL Road Frame - Red, 44 | 97 | 185 | 183.9382 | 34028.567000 | 32327.138650 |
| 2011 | LL Road Frame - Red, 48 | 61 | 114 | 183.9382 | 20968.954800 | 19920.507060 |
| 2011 | LL Road Frame - Red, 52 | 4 | 5 | 183.9382 | 919.691000 | 873.706450 |
| 2011 | LL Road Frame - Red, | 92 | 181 | 183.9382 | 33292.814200 | 31628.173490 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2014 | Touring-2000 Blue, 54 | 151 | 345 | 870.3475 | 272433.028140 | 258811.376733 |
| 2014 | Touring-2000 Blue, 60 | 138 | 297 | 869.7621 | 235923.870000 | 224127.676500 |
| 2014 | Touring-3000 Blue, 44 | 64 | 114 | 547.4831 | 57309.420000 | 54443.949000 |
| 2014 | Touring-3000 Blue, 50 | 108 | 286 | 494.5563 | 131869.198125 | 125275.738219 |
| 2014 | Touring-3000 Blue, 54 | 110 | 241 | 515.5958 | 115064.250000 | 109311.037500 |
| 2014 | Touring-3000 Blue, 58 | 95 | 168 | 529.8034 | 82846.260000 | 78703.947000 |
| 2014 | Touring-3000 Blue, 62 | 77 | 127 | 599.6645 | 68444.670000 | 65022.436500 |
| 2014 | Touring-3000 Yellow, 44 | 129 | 305 | 525.7449 | 145679.951760 | 138395.954172 |
| 2014 | Touring-3000 Yellow, 50 | 108 | 235 | 530.6427 | 113876.490000 | 108182.665500 |
| 2014 | Touring-3000 Yellow, 54 | 92 | 167 | 516.4173 | 80916.150000 | 76870.342500 |
| 2014 | Touring-3000 Yellow, 58 | 64 | 111 | 561.4021 | 56864.010000 | 54020.809500 |
| 2014 | Touring-3000 Yellow, 62 | 126 | 308 | 511.161 | 144984.518280 | 137735.292366 |
| 2014 | Water Bottle - 30 oz. | 2273 | 2902 | 4.8669 | 12900.317660 | 12255.301775 |
| 2014 | Women's Mountain Shorts, L | 306 | 1133 | 58.3089 | 50164.884564 | 47656.640337 |
| 2014 | Women's Mountain Shorts, M | 266 | 393 | 59.4651 | 21150.978000 | 20093.429100 |
| 2014 | Women's Mountain Shorts, S | 271 | 1094 | 57.9341 | 47963.580081 | 45565.401080 |
| 2014 | Women's Tights, L | 1 | 4 | 48.7435 | 194.974000 | 185.225300 |
| 2014 | Women's Tights, S | 1 | 2 | 48.7435 | 97.487000 | 92.612650 |

In [8]:

```
USE AdventureWorks2017;
GO

-- Create a custom scalar function
CREATE OR ALTER FUNCTION dbo.GetProductDiscount(@productID INT, @orderDate DATE)
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE @discount DECIMAL(10, 2)

    -- Assume a simple discount calculation based on product ID and order date
    IF @productID % 2 = 0 AND @orderDate >= '2023-01-01'
        SET @discount = 0.1  -- 10% discount for even product IDs in 2023
    ELSE
        SET @discount = 0.05  -- 5% discount for all other cases

    RETURN @discount
```

```sql
END;
GO

-- Complex query with joins, custom function, built-in SQL functions, and group by summa
WITH SalesData AS (
    SELECT
        soh.SalesOrderID,
        sod.ProductID,
        p.Name AS ProductName,
        soh.OrderDate,
        sod.OrderQty,
        sod.UnitPrice,
        dbo.GetProductDiscount(sod.ProductID, soh.OrderDate) AS Discount,
        sod.LineTotal
    FROM
        Sales.SalesOrderHeader AS soh
    INNER JOIN
        Sales.SalesOrderDetail AS sod ON soh.SalesOrderID = sod.SalesOrderID
    INNER JOIN
        Production.Product AS p ON sod.ProductID = p.ProductID
)
SELECT
    YEAR(OrderDate) AS OrderYear,
    ProductName,
    COUNT(*) AS TotalOrders,
    SUM(OrderQty) AS TotalQuantity,
    AVG(UnitPrice) AS AvgUnitPrice,
    SUM(LineTotal) AS TotalSales,
    SUM(LineTotal * (1 - Discount)) AS TotalSalesWithDiscount
FROM
    SalesData
GROUP BY
    YEAR(OrderDate),
    ProductName
ORDER BY
    OrderYear,
    ProductName
FOR JSON PATH;  -- Converts the result set to JSON format
```

Commands completed successfully.

Commands completed successfully.

(610 rows affected)

Total execution time: 00:00:01.167

Out[8]:

**JSON**


```
[
Cap","TotalOrders":159,"TotalQuantity":545,"AvgUnitPrice":5.1832,"TotalSales":2816.535136,"TotalSalesWithDiscount":2675.7
Mountain Frame - Black, 38","TotalOrders":39,"TotalQuantity":73,"AvgUnitPrice":714.7043,"TotalSales":52173.413
{"OrderYear":201
42","TotalOrders":47,"TotalQuantity":88,"AvgUnitPrice":714.7043,"TotalSales":62893.978400,"TotalSalesWithDiscount":59749.2
Mountain Frame - Black, 44","TotalOrders":7,"TotalQuantity":9,"AvgUnitPrice":809.7600,"TotalSales":7287.84
{"OrderYear":201
48","TotalOrders":41,"TotalQuantity":72,"AvgUnitPrice":809.7600,"TotalSales":58302.720000,"TotalSalesWithDiscount":55387.5
Mountain Frame - Silver, 38","TotalOrders":50,"TotalQuantity":91,"AvgUnitPrice":722.5949,"TotalSales":65756.135
{"OrderYear":201
42","TotalOrders":8,"TotalQuantity":13,"AvgUnitPrice":722.5949,"TotalSales":9393.733700,"TotalSalesWithDiscount":8924.0
Mountain Frame - Silver, 46","TotalOrders":41,"TotalQuantity":74,"AvgUnitPrice":722.5949,"TotalSales":53472.022
{"OrderYear":201
48","TotalOrders":49,"TotalQuantity":90,"AvgUnitPrice":818.7000,"TotalSales":73683.000000,"TotalSalesWithDiscount":69998.8
Road Frame - Red, 44","TotalOrders":7,"TotalQuantity":11,"AvgUnitPrice":758.0759,"TotalSales":8338.83
{"OrderYear
62","TotalOrders":9,"TotalQuantity":13,"AvgUnitPrice":758.0759,"TotalSales":9854.986700,"TotalSalesWithDiscount":9362.23736
Frame - Black, 44","TotalOrders":10,"TotalQuantity":13,"AvgUnitPrice":178.5808,"TotalSales":2321.55
{"OrderYear"
52","TotalOrders":91,"TotalQuantity":181,"AvgUnitPrice":178.5808,"TotalSales":32323.124800,"TotalSalesWithDiscount":30706.9
Road Frame - Black, 58","TotalOrders":54,"TotalQuantity":112,"AvgUnitPrice":178.5808,"TotalSales":20001.049
{"OrderYear"
```

oz.","TotalOrders":2273,"TotalQuantity":2902,"AvgUnitPrice":4.8669,"TotalSales":12900.317
{"OrderYear":20
L","TotalOrders":306,"TotalQuantity":1133,"AvgUnitPrice":58.3089,"TotalSales":50164.884
{"OrderYear":20
M","TotalOrders":266,"TotalQuantity":393,"AvgUnitPrice":59.4651,"TotalSales":21150.978
{"OrderYear":20
S","TotalOrders":271,"TotalQuantity":1094,"AvgUnitPrice":57.9341,"TotalSales":47963.580
{"Ord
L","TotalOrders":1,"TotalQuantity":4,"AvgUnitPrice":48.7435,"TotalSales":194.974000,"TotalSalesWithDiscount":185.225300
Tights, S","TotalOrders":1,"TotalQuantity":2,"AvgUnitPrice":48.7435,"TotalSales":97

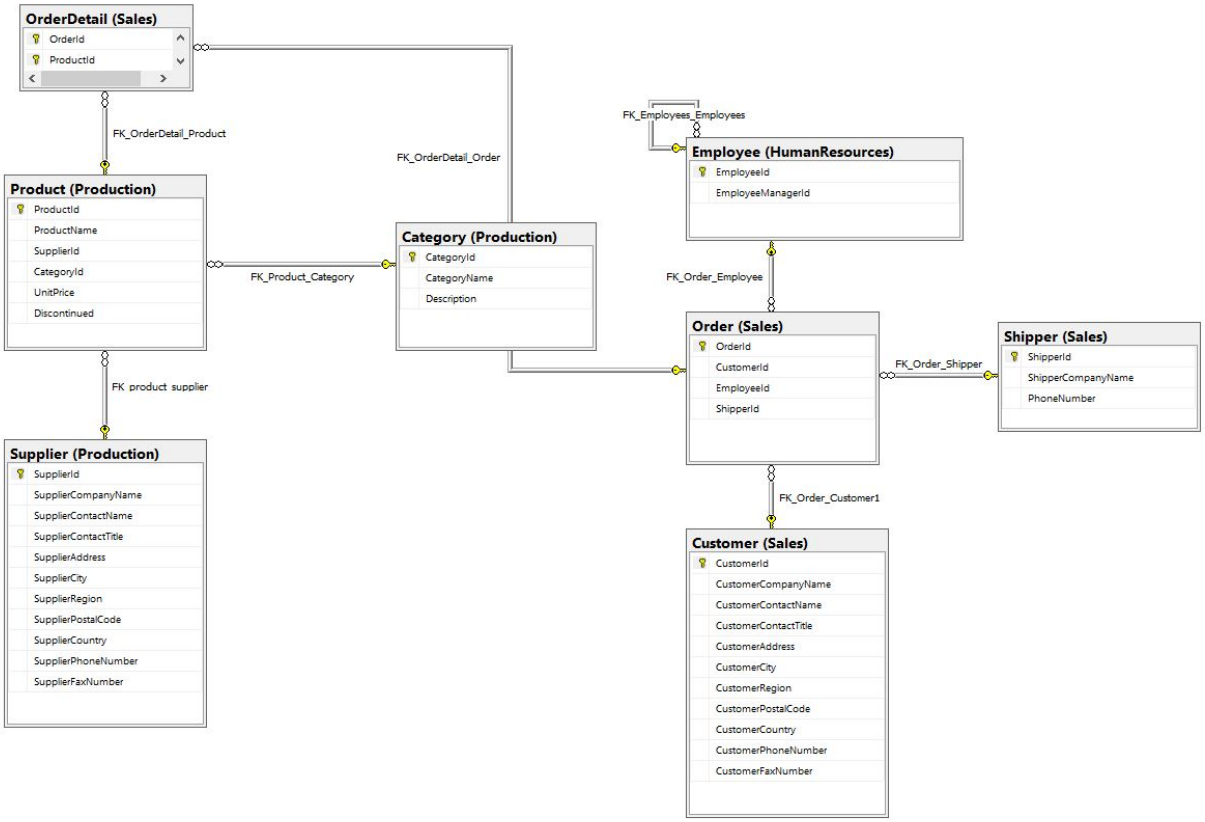
## **Complex Proposition 2:** Analyze sales commissions in the Northwinds2022TSQLV7 database by calculating total sales and commissions for each employee.

- **Custom Function:** Implement `dbo.CalculateCommission` to compute a 5% commission on sales, illustrating the direct financial benefit of sales activities for employees.
- **Tables Used:** The analysis will intersect data from `Sales.OrderDetail`, `HumanResources.Employee`, and `Sales.Order` to track sales transactions and employee contributions.
- **SQL Functions:** Employ `SUM()`, `COUNT()`, and `AVG()` in conjunction with `dbo.CalculateCommission` to derive total sales, order counts, average order value, and total commissions per employee.
- **Aggregation:** Data will be grouped by employee, summarizing sales and commission figures to highlight individual performance metrics.
- **CTE Usage:** Two CTEs, `TotalSalesPerOrder` and `TotalCommissionPerEmployee`, will streamline the calculation of sales totals and commission earnings, facilitating a comprehensive analysis.
- **Output:** The final report will detail each employee's ID, name, number of orders, total sales, average order value, and accumulated commissions, ranked by total sales to prioritize high performers.

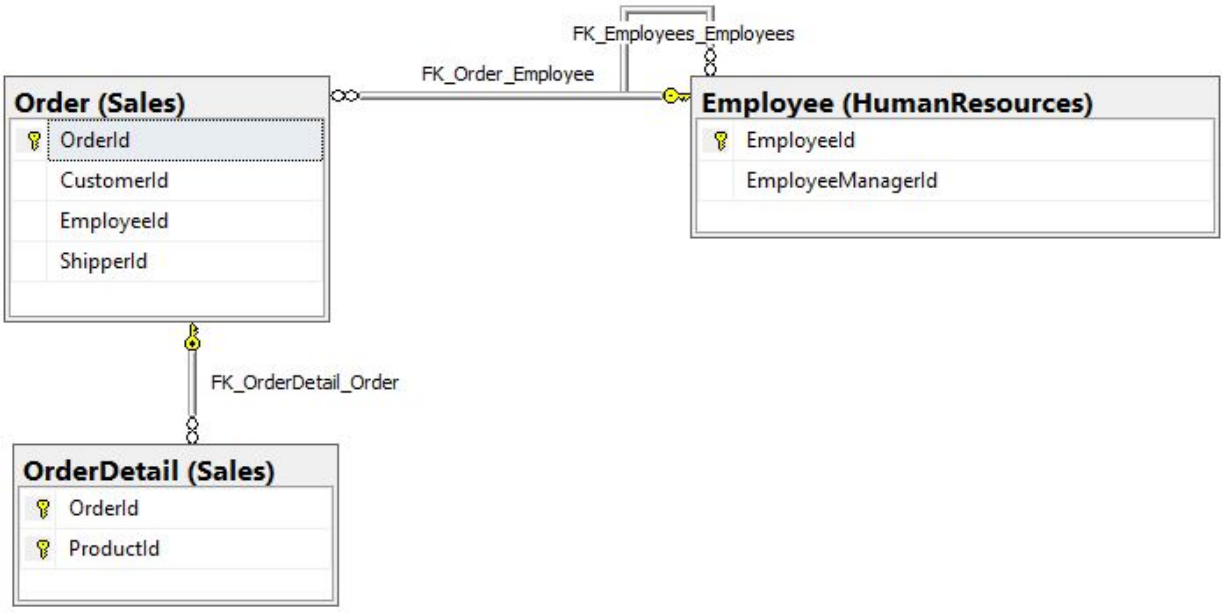
#### Why is this a top problem?

The SQL script provided is a well-structured example of a complex query, adhering to the specified guidelines for such queries. These guidelines include joining three or more tables, creating and using a custom scalar function, and utilizing built-in SQL functions with `GROUP BY` summarization, alongside integrating subqueries or CTEs for complex data organization. In this case, the query features a custom function to calculate commissions and employs two CTEs to manage sales and commission calculations. The final output, which presents a detailed aggregation of sales and commissions per employee, is highly relevant as it offers critical insights into employee performance and the effectiveness of the sales strategy, thereby providing a valuable tool for business decision-making and strategy optimization.

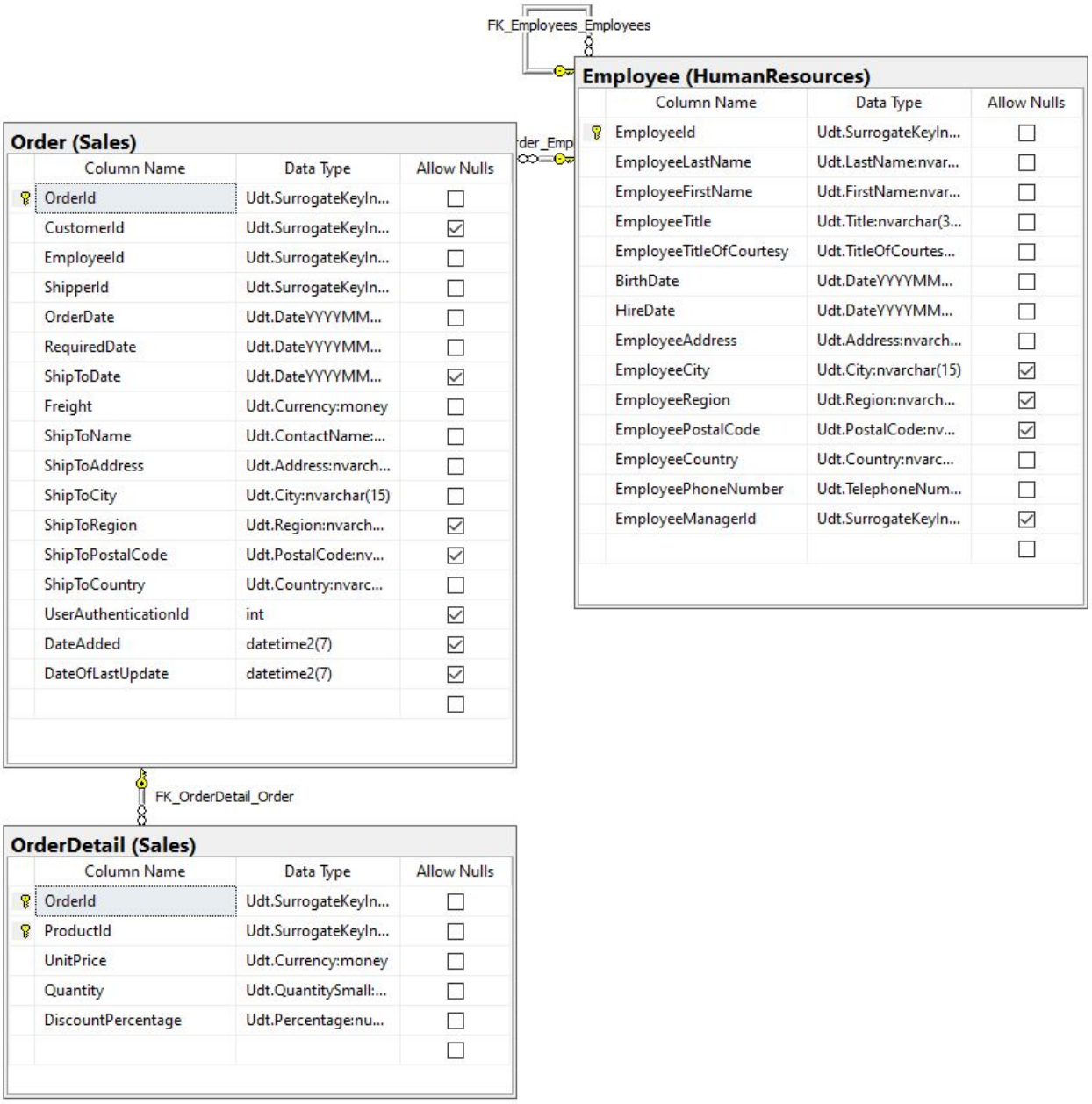
#### Subsystem in Northwinds2022TSQLV7




## Diagram of Tables



**Columns from Standard view**




# Table: Column Projection and Functions

| Table Name | Column Name | Derived Column | Function/Calculation |
|---|---|---|---|
| Sales.OrderDetail | OrderID | | |
| Sales.OrderDetail | UnitPrice, Quantity | TotalSaleAmount | SUM(UnitPrice * Quantity) in CTE |
| HumanResources.Employee | EmployeeID | | |
| HumanResources.Employee | EmployeeFirstName, EmployeeLastName | EmployeeName | CONCAT(EmployeeFirstName, ' ', EmployeeLastName) |
| Sales.Order | EmployeeID, OrderID | | |
| | | TotalCommission | dbo.CalculateCommission(TotalSaleAmount) |

# Table: Order By

| | Group By Column | Order By Column | Sort Order |
|---|---|---|---|

| EmployeeID, EmployeeName | TotalSales | DESC |
|---|---|---|

In [18]:
```
USE Northwinds2022TSQLV7;
GO

-- Create or alter a custom scalar function to calculate commission
CREATE OR ALTER FUNCTION dbo.CalculateCommission (@totalSaleAmount MONEY)
RETURNS MONEY
AS
BEGIN
    DECLARE @commissionRate FLOAT = 0.05;  -- 5% commission rate
    RETURN @totalSaleAmount * @commissionRate;
END;
GO

-- CTE for total sales per order
WITH TotalSalesPerOrder AS (
    SELECT
        od.OrderID,
        SUM(od.UnitPrice * od.Quantity) AS TotalSaleAmount
    FROM Sales.OrderDetail od
    GROUP BY od.OrderID
),
-- CTE for total commission per employee
TotalCommissionPerEmployee AS (
    SELECT
        e.EmployeeID,
        SUM(dbo.CalculateCommission(tspo.TotalSaleAmount)) AS TotalCommission
    FROM HumanResources.Employee e
    JOIN Sales.[Order] o ON e.EmployeeID = o.EmployeeID
    JOIN TotalSalesPerOrder tspo ON o.OrderID = tspo.OrderID
    GROUP BY e.EmployeeID
)
-- Main query
SELECT
    e.EmployeeID,
    e.EmployeeFirstName + ' ' + e.EmployeeLastName AS EmployeeName,
    COUNT(o.OrderID) AS NumberOfOrders,
    SUM(tspo.TotalSaleAmount) AS TotalSales,
    AVG(tspo.TotalSaleAmount) AS AverageOrderValue,
    tce.TotalCommission  -- Using the TotalCommission from the CTE
FROM HumanResources.Employee e
JOIN Sales.[Order] o ON e.EmployeeID = o.EmployeeID
JOIN TotalSalesPerOrder tspo ON o.OrderID = tspo.OrderID
JOIN TotalCommissionPerEmployee tce ON e.EmployeeID = tce.EmployeeID
GROUP BY e.EmployeeID, e.EmployeeFirstName, e.EmployeeLastName, tce.TotalCommission
ORDER BY TotalSales DESC;
```

Commands completed successfully.

Commands completed successfully.

(9 rows affected)

Total execution time: 00:00:00.121

Out[18]:

| EmployeeID | EmployeeName | NumberOfOrders | TotalSales | AverageOrderValue | TotalCommission |
|---|---|---|---|---|---|
| 4 | Yael Peled | 156 | 250187.45 | 1603.7657 | 12509.3725 |
| 3 | Judy Lew | 127 | 213051.30 | 1677.5692 | 10652.565 |
| 1 | Sara Davis | 123 | 202143.71 | 1643.4447 | 10107.1855 |
| 2 | Don Funk | 96 | 177749.26 | 1851.5547 | 8887.463 |
| 7 | Russell King | 72 | 141295.99 | 1962.4443 | 7064.7995 |
| 8 | Maria Cameron | 104 | 133301.03 | 1281.7406 | 6665.0515 |

| | | | | | |
|---|---|---|---|---|---|
| 9 | Patricia Doyle | 43 | 82964.00 | 1929.3953 | 4148.20 |
| 6 | Paul Suurs | 67 | 78198.10 | 1167.1358 | 3909.905 |
| 5 | Sven Mortensen | 42 | 75567.75 | 1799.2321 | 3778.3875 |

In [9]:

```
USE Northwinds2022TSQLV7;
GO

-- Create or alter a custom scalar function to calculate commission
CREATE OR ALTER FUNCTION dbo.CalculateCommission (@totalSaleAmount MONEY)
RETURNS MONEY
AS
BEGIN
    DECLARE @commissionRate FLOAT = 0.05;  -- 5% commission rate
    RETURN @totalSaleAmount * @commissionRate;
END;
GO

-- CTE for total sales per order
WITH TotalSalesPerOrder AS (
    SELECT
        od.OrderID,
        SUM(od.UnitPrice * od.Quantity) AS TotalSaleAmount
    FROM Sales.OrderDetail od
    GROUP BY od.OrderID
),
-- CTE for total commission per employee
TotalCommissionPerEmployee AS (
    SELECT
        e.EmployeeID,
        SUM(dbo.CalculateCommission(tspo.TotalSaleAmount)) AS TotalCommission
    FROM HumanResources.Employee e
    JOIN Sales.[Order] o ON e.EmployeeID = o.EmployeeID
    JOIN TotalSalesPerOrder tspo ON o.OrderID = tspo.OrderID
    GROUP BY e.EmployeeID
)
-- Main query
SELECT
    e.EmployeeID,
    e.EmployeeFirstName + ' ' + e.EmployeeLastName AS EmployeeName,
    COUNT(o.OrderID) AS NumberOfOrders,
    SUM(tspo.TotalSaleAmount) AS TotalSales,
    AVG(tspo.TotalSaleAmount) AS AverageOrderValue,
    tce.TotalCommission  -- Using the TotalCommission from the CTE
FROM HumanResources.Employee e
JOIN Sales.[Order] o ON e.EmployeeID = o.EmployeeID
JOIN TotalSalesPerOrder tspo ON o.OrderID = tspo.OrderID
JOIN TotalCommissionPerEmployee tce ON e.EmployeeID = tce.EmployeeID
GROUP BY e.EmployeeID, e.EmployeeFirstName, e.EmployeeLastName, tce.TotalCommission
ORDER BY TotalSales DESC
FOR JSON PATH;
```

Commands completed successfully.

Commands completed successfully.

(9 rows affected)

Total execution time: 00:00:00.041

Out[9]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**

[{"EmployeeID":4,"EmployeeName":"Yael Peled","NumberOfOrders":156,"TotalSales":250187.4500,"AverageOrderValue":1603.7657,"TotalCommission":12509.3725},
{"EmployeeID":3,"EmployeeName":"Judy Lew","NumberOfOrders":127,"TotalSales":213051.3000,"AverageOrderValue":1677.5692,"TotalCommission":10652.5650},
{"EmployeeID":1,"EmployeeName":"Sara Davis","NumberOfOrders":123,"TotalSales":202143.7100,"AverageOrderValue":1643.4447,"TotalCommission":10107.1855},

{"EmployeeID":2,"EmployeeName":"Don Funk","NumberOfOrders":96,"TotalSales":177749.2600,"AverageOrderValue":1851.5547,"TotalCommission":8887.4630},
{"EmployeeID":7,"EmployeeName":"Russell King","NumberOfOrders":72,"TotalSales":141295.9900,"AverageOrderValue":1962.4443,"TotalCommission":7064.7995},
{"EmployeeID":8,"EmployeeName":"Maria Cameron","NumberOfOrders":104,"TotalSales":133301.0300,"AverageOrderValue":1281.7406,"TotalCommission":6665.0515},
{"EmployeeID":9,"EmployeeName":"Patricia Doyle","NumberOfOrders":43,"TotalSales":82964.0000,"AverageOrderValue":1929.3953,"TotalCommission":4148.2000},
{"EmployeeID":6,"EmployeeName":"Paul Suurs","NumberOfOrders":67,"TotalSales":78198.1000,"AverageOrderValue":1167.1358,"TotalCommission":3909.9050},
{"EmployeeID":5,"EmployeeName":"Sven Mortensen","NumberOfOrders":42,"TotalSales":75567.7500,"AverageOrderValue":1799.2321,"TotalCommission":3778.3875}]

# \===== Top 3 Worst Problems =====

## **Medium Proposition 2:** Generate a detailed order summary report in the Northwinds2022TSQLV7 database, showcasing total sales amount, average quantity per order, and the number of orders per customer.

**CTE Creation:** `OrderSummary` will be established to compute the total amount and average quantity for each order in the `Sales.OrderDetail` table.

**Tables Used:** The primary focus will be on `Sales.OrderDetail` for detailed line items and `Sales.Order` for overarching order data, ensuring a comprehensive analysis of sales transactions.

**SQL Functions:** The query will integrate `SUM()` to calculate the total sales amount, `AVG()` to find the average quantity per order, and `COUNT()` to count the number of orders for each customer.

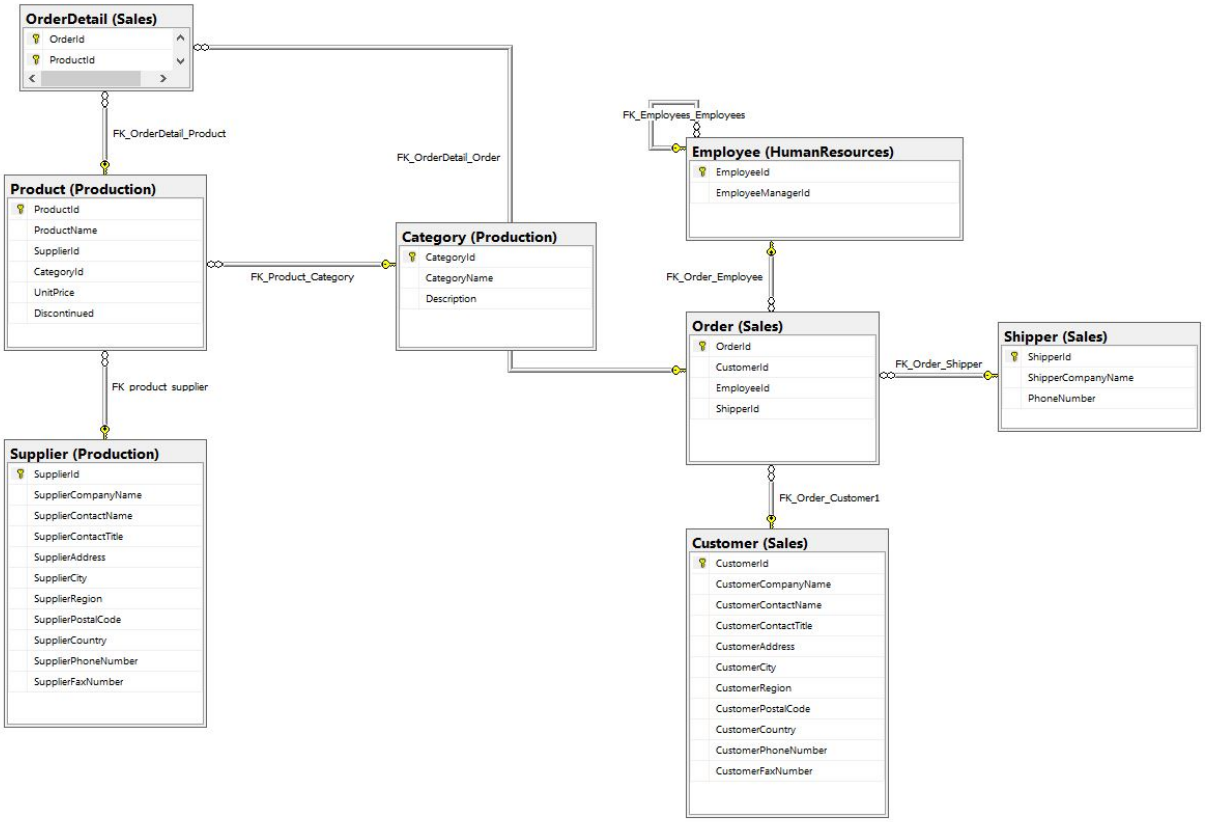
**Aggregation:** Grouping will occur at the order level within the CTE, and further aggregation in the main query will provide a per-customer summary, illustrating the total sales volume and average product quantity ordered.

**Output:** The final output will present the order ID, customer ID, total sales amount, average quantity per order, and total number of orders for each customer, sorted by order ID to facilitate easy tracking of sales activity and customer engagement.

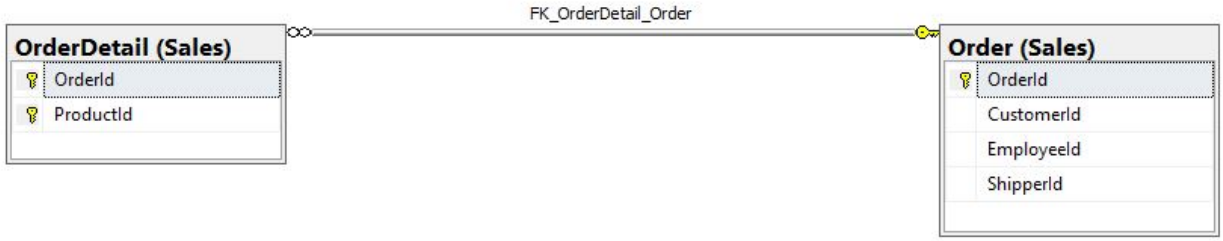
**Why is this a top problem?**

I believe the current query is too simplistic for its intended medium complexity because the `NumberOfOrders` column offers no additional insight, given the data is already grouped by `OrderID`. To improve this, I intend to remove that column and enrich the query with more complex analytics. A different approach could be to join the customer table and aggregate data to assess customer buying behavior, such as calculating the average purchase amount per customer or identifying trends in customer orders over time. This would not only increase the query's complexity but also provide more actionable insights for business strategies.

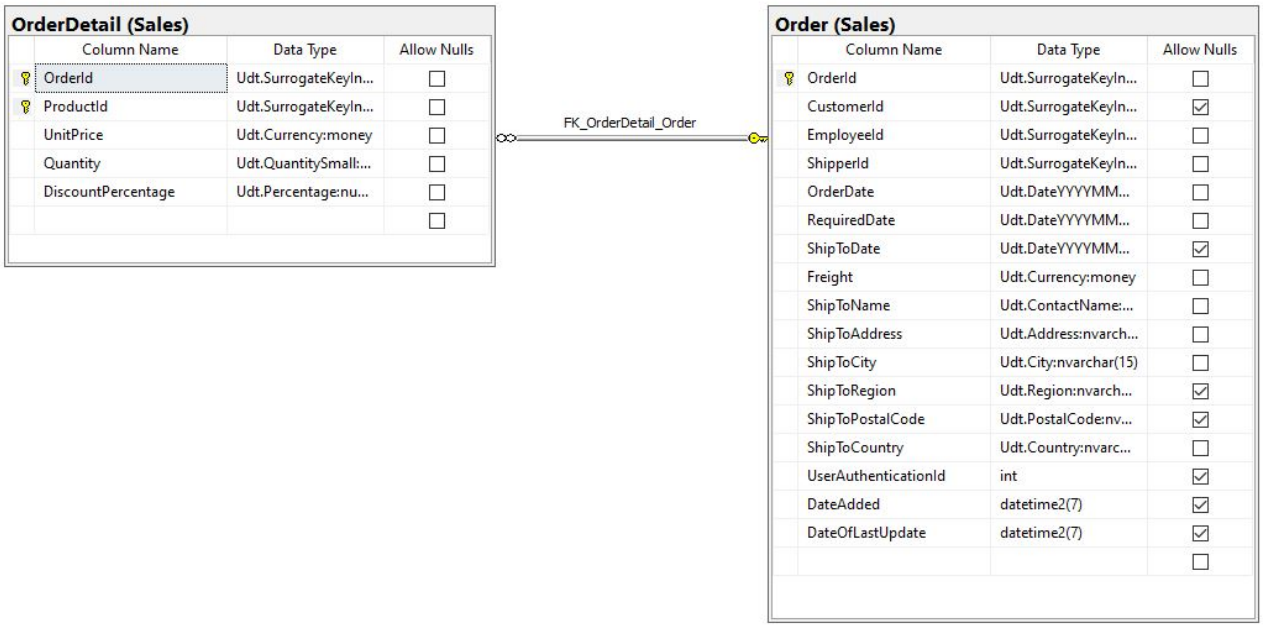
**Subsystem in Northwinds2022TSQLV7**




## Diagram of Tables



**Columns from Standard view**




## Table: Column Projection

| Table Name | Column Name | Derived Column | Function/Calculation |
|---|---|---|---|
| Sales.OrderDetail | OrderID | | |
| Sales.OrderDetail | UnitPrice, Quantity | TotalAmount | SUM(UnitPrice * Quantity) |
| Sales.OrderDetail | Quantity | AverageQuantity | AVG(Quantity) |
| Sales.Order | OrderID | | |
| Sales.Order | CustomerID | | |

## Table: Group By Order By

| Group By Column | Order By Column |
|---|---|
| OrderID, | |

CustomerID, TotalAmount, AverageQuantity | OrderID |

In [23]:
```
USE Northwinds2022TSQLV7;

WITH OrderSummary AS (
    SELECT
        OrderID,
        SUM(UnitPrice * Quantity) AS TotalAmount,
        AVG(Quantity) AS AverageQuantity
    FROM Sales.OrderDetail
    GROUP BY OrderID
)
SELECT
    o.OrderID,
    o.CustomerID,
    os.TotalAmount,
    os.AverageQuantity,
    COUNT(o.OrderID) AS NumberOfOrders
FROM Sales.[Order] o
```

```
JOIN OrderSummary os ON o.OrderID = os.OrderID
GROUP BY o.OrderID, o.CustomerID, os.TotalAmount, os.AverageQuantity
ORDER BY o.OrderID;
```

(830 rows affected)

Total execution time: 00:00:00.019

Out[23]:

| OrderID | CustomerID | TotalAmount | AverageQuantity | NumberOfOrders |
|---|---|---|---|---|
| 10248 | 85 | 440.00 | 9 | 1 |
| 10249 | 79 | 1863.40 | 24 | 1 |
| 10250 | 34 | 1813.00 | 20 | 1 |
| 10251 | 84 | 670.80 | 13 | 1 |
| 10252 | 76 | 3730.00 | 35 | 1 |
| 10253 | 34 | 1444.80 | 34 | 1 |
| 10254 | 14 | 625.20 | 19 | 1 |
| 10255 | 68 | 2490.50 | 27 | 1 |
| 10256 | 88 | 517.80 | 13 | 1 |
| 10257 | 35 | 1119.90 | 15 | 1 |
| 10258 | 20 | 2018.60 | 40 | 1 |
| 10259 | 13 | 100.80 | 5 | 1 |
| 10260 | 56 | 1746.20 | 25 | 1 |
| 10261 | 61 | 448.00 | 20 | 1 |
| 10262 | 65 | 624.80 | 9 | 1 |
| 10263 | 20 | 2464.80 | 46 | 1 |
| 10264 | 24 | 724.50 | 30 | 1 |
| 10265 | 7 | 1176.00 | 25 | 1 |
| 10266 | 87 | 364.80 | 12 | 1 |
| 10267 | 25 | 4031.00 | 45 | 1 |
| 10268 | 33 | 1101.20 | 7 | 1 |
| 10269 | 89 | 676.00 | 40 | 1 |
| 10270 | 87 | 1376.00 | 27 | 1 |
| 10271 | 75 | 48.00 | 24 | 1 |
| 10272 | 65 | 1456.00 | 23 | 1 |
| 10273 | 63 | 2142.40 | 30 | 1 |
| 10274 | 85 | 538.60 | 13 | 1 |
| 10275 | 49 | 307.20 | 9 | 1 |
| 10276 | 80 | 420.00 | 12 | 1 |
| 10277 | 52 | 1200.80 | 16 | 1 |
| 10278 | 5 | 1488.80 | 16 | 1 |
| 10279 | 44 | 468.00 | 15 | 1 |
| 10280 | 5 | 613.20 | 20 | 1 |
| 10281 | 69 | 86.50 | 3 | 1 |
| 10282 | 69 | 155.40 | 4 | 1 |

| | | | | |
|---|---|---|---|---|
| 11053 | 59 | 3658.75 | 26 | 1 |
| 11054 | 12 | 305.00 | 15 | 1 |
| 11055 | 35 | 1727.50 | 17 | 1 |
| 11056 | 19 | 3740.00 | 41 | 1 |
| 11057 | 53 | 45.00 | 3 | 1 |
| 11058 | 6 | 858.00 | 9 | 1 |
| 11059 | 67 | 1838.00 | 25 | 1 |
| 11060 | 27 | 266.00 | 7 | 1 |
| 11061 | 32 | 510.00 | 15 | 1 |
| 11062 | 66 | 508.00 | 11 | 1 |
| 11063 | 37 | 1445.50 | 33 | 1 |
| 11064 | 71 | 4722.30 | 34 | 1 |
| 11065 | 46 | 252.56 | 12 | 1 |
| 11066 | 89 | 928.75 | 26 | 1 |
| 11067 | 17 | 86.85 | 9 | 1 |
| 11068 | 62 | 2384.80 | 24 | 1 |
| 11069 | 80 | 360.00 | 20 | 1 |
| 11070 | 44 | 1873.50 | 27 | 1 |
| 11071 | 46 | 510.00 | 12 | 1 |
| 11072 | 20 | 5218.00 | 50 | 1 |
| 11073 | 58 | 300.00 | 15 | 1 |
| 11074 | 73 | 244.30 | 14 | 1 |
| 11075 | 68 | 586.00 | 14 | 1 |
| 11076 | 9 | 1057.00 | 16 | 1 |
| 11077 | 65 | 1374.60 | 2 | 1 |

In [10]:

```
USE Northwinds2022TSQLV7;

WITH OrderSummary AS (
    SELECT
        OrderID,
        SUM(UnitPrice * Quantity) AS TotalAmount,
        AVG(Quantity) AS AverageQuantity
    FROM Sales.OrderDetail
    GROUP BY OrderID
)
SELECT
    o.OrderID,
    o.CustomerID,
    os.TotalAmount,
    os.AverageQuantity
FROM Sales.[Order] o
JOIN OrderSummary os ON o.OrderID = os.OrderID
GROUP BY o.OrderID, o.CustomerID, os.TotalAmount, os.AverageQuantity
ORDER BY o.OrderID
FOR JSON PATH;
```

(830 rows affected)

Total execution time: 00:00:00.020

Out[10]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**


[{"OrderID":10248,"CustomerID":85,"TotalAmount":440.0000,"AverageQuantity":9},
{"OrderID":10249,"CustomerID":79,"TotalAmount":1863.4000,"AverageQuantity":24},
{"OrderID":10250,"CustomerID":34,"TotalAmount":1813.0000,"AverageQuantity":20},
{"OrderID":10251,"CustomerID":84,"TotalAmount":670.8000,"AverageQuantity":13},
{"OrderID":10252,"CustomerID":76,"TotalAmount":3730.0000,"AverageQuantity":35},
{"OrderID":10253,"CustomerID":34,"TotalAmount":1444.8000,"AverageQuantity":34},
{"OrderID":10254,"CustomerID":14,"TotalAmount":625.2000,"AverageQuantity":19},
{"OrderID":10255,"CustomerID":68,"TotalAmount":2490.5000,"AverageQuantity":27},
{"OrderID":10256,"CustomerID":88,"TotalAmount":517.8000,"AverageQuantity":13},
{"OrderID":10257,"CustomerID":35,"TotalAmount":1119.9000,"AverageQuantity":15},
{"OrderID":10258,"CustomerID":20,"TotalAmount":2018.6000,"AverageQuantity":40},
{"OrderID":10259,"CustomerID":13,"TotalAmount":100.8000,"AverageQuantity":5},
{"OrderID":10260,"CustomerID":56,"TotalAmount":1746.2000,"AverageQuantity":25},
{"OrderID":10261,"CustomerID":61,"TotalAmount":448.0000,"AverageQuantity":20},
{"OrderID":10262,"CustomerID":65,"TotalAmount":624.8000,"AverageQuantity":9},
{"OrderID":10263,"CustomerID":20,"TotalAmount":2464.8000,"AverageQuantity":46},
{"OrderID":10264,"CustomerID":24,"TotalAmount":724.5000,"AverageQuantity":30},
{"OrderID":10265,"CustomerID":7,"TotalAmount":1176.0000,"AverageQuantity":25},
{"OrderID":10266,"CustomerID":87,"TotalAmount":364.8000,"AverageQuantity":12},
{"OrderID":10267,"CustomerID":25,"TotalAmount":4031.0000,"AverageQuantity":45},
{"OrderID":10268,"CustomerID":33,"TotalAmount":1101.2000,"AverageQuantity":7},
{"OrderID":10269,"CustomerID":89,"TotalAmount":676.0000,"AverageQuantity":40},
{"OrderID":10270,"CustomerID":87,"TotalAmount":1376.0000,"AverageQuantity":27},
{"OrderID":10271,"CustomerID":75,"TotalAmount":48.0000,"AverageQuantity":24},
{"OrderID":10272,"CustomerID":65,"TotalAmount":1456.0000,"AverageQuantity":23},
{"OrderID":10273,"CustomerID":63,"TotalAmount":2142.4000,"AverageQuantity":30},
{"OrderID":10274,"CustomerID":85,"TotalAmount":538.6000,"AverageQuantity":13},
{"OrderID":10275,"CustomerID":49,"TotalAmount":307.2000,"AverageQuantity":9},
{"OrderID":10276,"CustomerID":80,"TotalAmount":420.0000,"AverageQuantity":12},
{"OrderID":10277,"CustomerID":52,"TotalAmount":1200.8000,"AverageQuantity":16},
{"OrderID":10278,"CustomerID":5,"TotalAmount":1488.8000,"AverageQuantity":16},
{"OrderID":10279,"CustomerID":44,"TotalAmount":468.0000,"AverageQuantity":15},
{"OrderID":10280,"CustomerID":5,"TotalAmount":613.2000,"AverageQuantity":20},
{"OrderID":10281,"CustomerID":69,"TotalAmount":86.5000,"AverageQuantity":3},
{"OrderID":10282,"CustomerID":69,"TotalAmount":155.4000,"AverageQuantity":4},
{"OrderID":10283,"CustomerID":46,"TotalAmount":1414.8000,"AverageQuantity":19},
{"OrderID":10284,"CustomerID":44,"TotalAmount":1452.0000,"AverageQuantity":15},
{"OrderID":10285,"CustomerID":63,"TotalAmount":2179.2000,"AverageQuantity":40},
{"OrderID":10286,"CustomerID":63,"TotalAmount":3016.0000,"AverageQuantity":70},
{"OrderID":10287,"CustomerID":67,"TotalAmount":924.0000,"AverageQuantity":25},
{"OrderID":10288,"CustomerID":66,"TotalAmount":89.0000,"AverageQuantity":6},
{"OrderID":10289,"CustomerID":11,"TotalAmount":479.4000,"AverageQuantity":19},
{"OrderID":10290,"CustomerID":15,"TotalAmount":2169.0000,"AverageQuantity":15},
{"OrderID":10291,"CustomerID":61,"TotalAmount":552.8000,"AverageQuantity":15},
{"OrderID":10292,"CustomerID":81,"TotalAmount":1296.0000,"AverageQuantity":20},
{"OrderID":10293,"CustomerID":80,"TotalAmount":848.7000,"AverageQuantity":8},
{"OrderID":10294,"CustomerID":65,"TotalAmount":1887.6000,"AverageQuantity":15},
{"OrderID":10295,"CustomerID":85,"TotalAmount":121.6000,"AverageQuantity":4},
{"OrderID":10296,"CustomerID":46,"TotalAmount":1050.6000,"AverageQuantity":19},
{"OrderID":10297,"CustomerID":7,"TotalAmount":1420.0000,"AverageQuantity":40},
{"OrderID":10298,"CustomerID":37,"TotalAmount":3127.0000,"AverageQuantity":31},
{"OrderID":10299,"CustomerID":67,"TotalAmount":349.5000,"AverageQuantity":17},
{"OrderID":10300,"CustomerID":49,"TotalAmount":608.0000,"AverageQuantity":25},
{"OrderID":10301,"CustomerID":86,"TotalAmount":755.0000,"AverageQuantity":15},
{"OrderID":10302,"CustomerID":76,"TotalAmount":2708.8000,"AverageQuantity":26},
{"OrderID":10303,"CustomerID":30,"TotalAmount":1242.0000,"AverageQuantity":28},
{"OrderID":10304,"CustomerID":80,"TotalAmount":954.4000,"AverageQuantity":14},
{"OrderID":10305,"CustomerID":55,"TotalAmount":4157.0000,"AverageQuantity":26},
{"OrderID":10306,"CustomerID":69,"TotalAmount":498.5000,"AverageQuantity":8},
{"OrderID":10307,"CustomerID":48,"TotalAmount":424.0000,"AverageQuantity":6},
{"OrderID":10308,"CustomerID":2,"TotalAmount":88.8000,"AverageQuantity":3},
{"OrderID":10309,"CustomerID":37,"TotalAmount":1762.0000,"AverageQuantity":15},
{"OrderID":10310,"CustomerID":77,"TotalAmount":336.0000,"AverageQuantity":7},
{"OrderID":10311,"CustomerID":18,"TotalAmount":268.8000,"AverageQuantity":6},
{"OrderID":10312,"CustomerID":86,"TotalAmount":1614.8000,"AverageQuantity":14},
{"OrderID":10313,"CustomerID":63,"TotalAmount":182.4000,"AverageQuantity":12},
{"OrderID":10314,"CustomerID":65,"TotalAmount":2327.0000,"AverageQuantity":31},
{"OrderID":10315,"CustomerID":38,"TotalAmount":516.8000,"AverageQuantity":22},
{"OrderID":10316,"CustomerID":65,"TotalAmount":2835.0000,"AverageQuantity":40},
{"OrderID":10317,"CustomerID":48,"TotalAmount":288.0000,"AverageQuantity":20},
{"OrderID":10318,"CustomerID":38,"TotalAmount":240.4000,"AverageQuantity":13},
{"OrderID":10319,"CustomerID":80,"TotalAmount":1191.2000,"AverageQuantity":17},
{"OrderID":10320,"CustomerID":87,"TotalAmount":516.0000,"AverageQuantity":30},

{"OrderID":11071,"CustomerID":46,"TotalAmount":510.0000,"AverageQuantity":12},
{"OrderID":11072,"CustomerID":20,"TotalAmount":5218.0000,"AverageQuantity":50},
{"OrderID":11073,"CustomerID":58,"TotalAmount":300.0000,"AverageQuantity":15},
{"OrderID":11074,"CustomerID":73,"TotalAmount":244.3000,"AverageQuantity":14},
{"OrderID":11075,"CustomerID":68,"TotalAmount":586.0000,"AverageQuantity":14},
{"OrderID":11076,"CustomerID":9,"TotalAmount":1057.0000,"AverageQuantity":16},
{"OrderID":11077,"CustomerID":65,"TotalAmount":1374.6000,"AverageQuantity":2}]

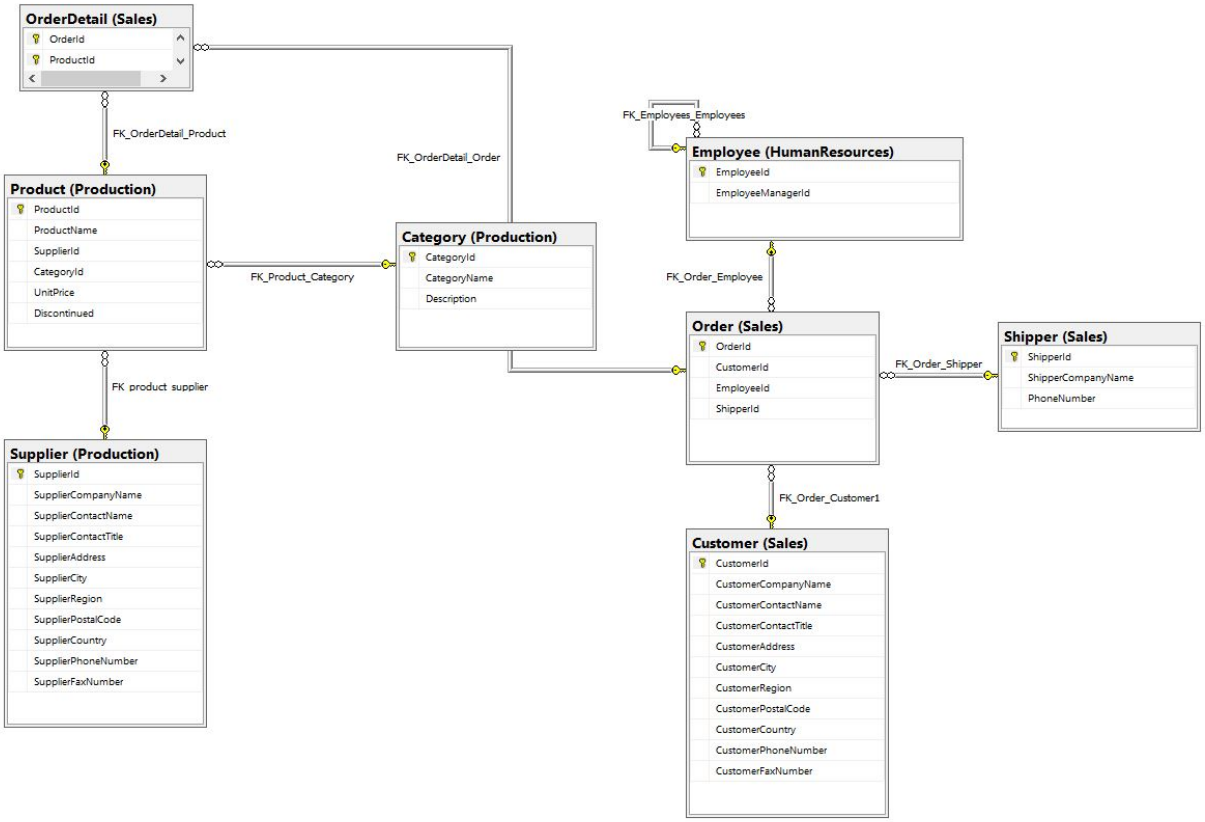
## Medium Proposition 3: Create a comprehensive report in the Northwinds2022TSQLV7 database that details the total number of orders and total revenue generated by each shipper.

- **CTE Creation:** Two CTEs, `OrderCounts` and `RevenueByShipper`, will be crafted. `OrderCounts` will calculate the total number of orders for each shipper, while `RevenueByShipper` will aggregate the total revenue each shipper generated.
- **Tables Used:** The data will be drawn from `Sales.[Order]` for order counts and shipping details, and `Sales.OrderDetail` for revenue calculations, ensuring a thorough analysis of shipping operations and financial contributions.
- **SQL Functions:** The query will apply `COUNT()` in `OrderCounts` to enumerate the orders per shipper, and `SUM()` in `RevenueByShipper` to total the revenue associated with each shipper's deliveries.
- **Aggregation:** Each CTE will group the results by `ShipperID` to organize the data for clear, actionable insights.
- **Output:** The final report will align the `Sales.Shipper` table with the CTEs to provide a cohesive view, listing each shipper's ID, company name, total orders, and total revenue, sorted by total revenue in descending order to highlight the most financially significant shippers.

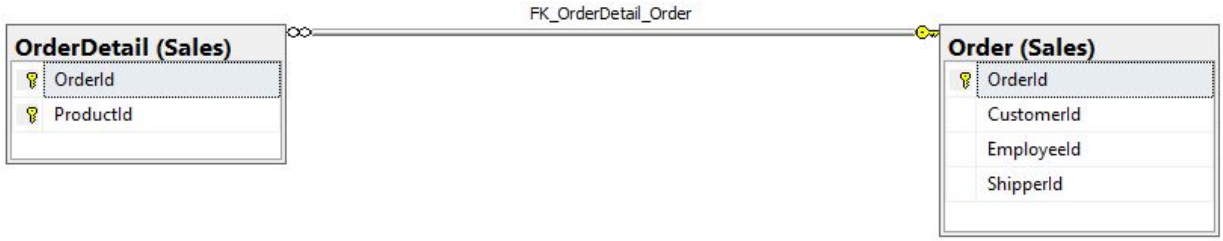
#### Why is this a top problem?

The current query, while using CTEs to analyze order counts and revenue by shipper, lacks depth in its analysis. To enrich it, I suggest adding a dimension of time to the existing CTEs, such as calculating monthly revenue growth for each shipper. This enhancement would allow for a more dynamic analysis, providing insights into how shipper performance trends over time, thereby increasing the query's complexity and yielding more actionable business intelligence in a single, cohesive analysis.

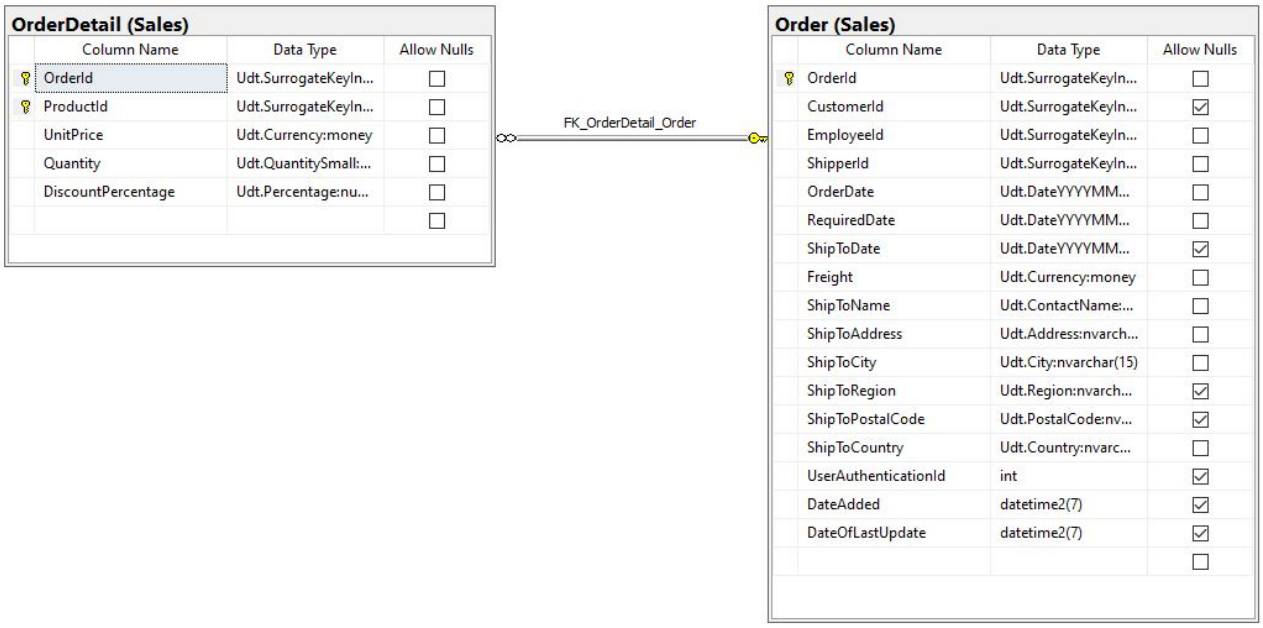
#### Subsystem in Northwinds2022TSQLV7




## Diagram of Tables



### Columns from Standard view




## Table: Column Projection

| Table Name | Column Name | Derived Column | Function/Calculation |
|---|---|---|---|
| Sales.[Order] | ShipperID, OrderID | | |
| Sales.OrderDetail | UnitPrice, Quantity | TotalRevenue | SUM(UnitPrice * Quantity) |
| Sales.Shipper | ShipperID | | |
| Sales.Shipper | ShipperCompanyName | ShipperName | |

## Table:Group By Order By

| Group By Column | Order By Column |
|---|---|
| ShipperID | TotalRevenue DESC |

In [24]:

```
USE Northwinds2022TSQLV7
go

WITH OrderCounts AS (
    SELECT
        ShipperID,
        COUNT(OrderID) AS TotalOrders
    FROM Sales.[Order]
    GROUP BY ShipperID
),
RevenueByShipper AS (
    SELECT
        o.ShipperId,
        SUM(od.UnitPrice * od.Quantity) AS TotalRevenue
    FROM Sales.[Order] o
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY o.ShipperId
)
SELECT
    s.ShipperID,
```

```
    s.ShipperCompanyName AS ShipperName,
    oc.TotalOrders,
    rs.TotalRevenue
FROM Sales.Shipper s
LEFT JOIN OrderCounts oc ON s.ShipperID = oc.ShipperID
LEFT JOIN RevenueByShipper rs ON s.ShipperID = rs.ShipperID
ORDER BY rs.TotalRevenue DESC;
```

Commands completed successfully.

(3 rows affected)

Total execution time: 00:00:00.010

Out[24]:

| ShipperID | ShipperName | TotalOrders | TotalRevenue |
|---|---|---|---|
| 2 | Shipper ETYNR | 326 | 572724.58 |
| 3 | Shipper ZHISN | 255 | 407750.82 |
| 1 | Shipper GVSUA | 249 | 373983.19 |

In [11]:

```
USE Northwinds2022TSQLV7;
GO

WITH OrderCounts AS (
    SELECT
        ShipperID,
        COUNT(OrderID) AS TotalOrders
    FROM Sales.[Order]
    GROUP BY ShipperID
),
RevenueByShipper AS (
    SELECT
        o.ShipperId,
        SUM(od.UnitPrice * od.Quantity) AS TotalRevenue
    FROM Sales.[Order] o
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY o.ShipperId
)
SELECT
    s.ShipperID,
    s.ShipperCompanyName AS ShipperName,
    oc.TotalOrders,
    rs.TotalRevenue
FROM Sales.Shipper s
LEFT JOIN OrderCounts oc ON s.ShipperID = oc.ShipperID
LEFT JOIN RevenueByShipper rs ON s.ShipperID = rs.ShipperID
ORDER BY rs.TotalRevenue DESC
FOR JSON PATH;
```

Commands completed successfully.

(3 rows affected)

Total execution time: 00:00:00.021

Out[11]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**

[{"ShipperID":2,"ShipperName":"Shipper ETYNR","TotalOrders":326,"TotalRevenue":572724.5800},
{"ShipperID":3,"ShipperName":"Shipper ZHISN","TotalOrders":255,"TotalRevenue":407750.8200},
{"ShipperID":1,"ShipperName":"Shipper GVSUA","TotalOrders":249,"TotalRevenue":373983.1900}]

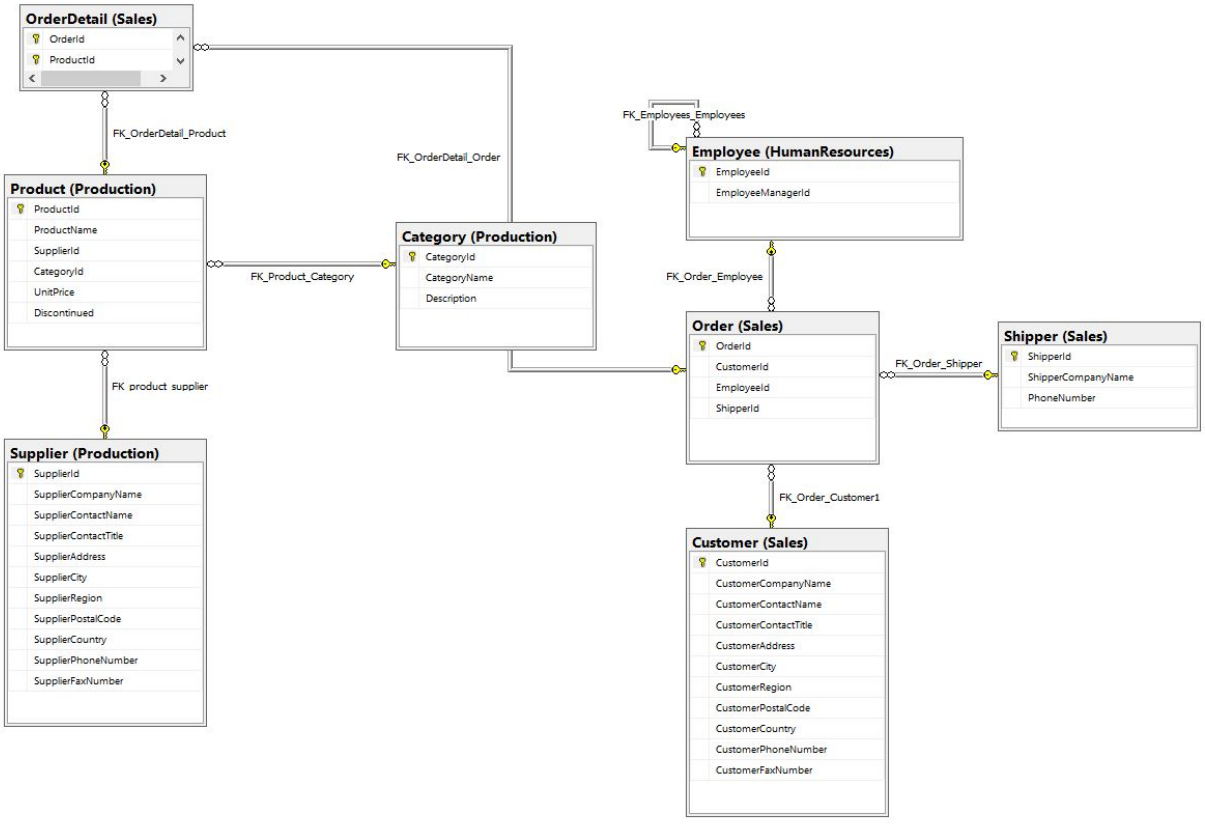
## Medium Proposition 4: Develop a report in the Northwinds2022TSQLV7 database that identifies the last order date for each customer along with their total number of orders, organized by the most frequent customers and recent orders.

- **CTE Creation:** `LastOrderDetails` CTE is defined to identify the most recent order for each customer by utilizing the `ROW_NUMBER()` function, partitioned by `CustomerID` and ordered by `OrderDate` in descending sequence.
- **Tables Used:** The data will be sourced from `Sales.Customer` for customer information, `Sales.[Order]` for order counts, and `Sales.OrderDetail` for detailed order data, ensuring a full spectrum analysis of customer engagement.
- **SQL Functions:** The `ROW_NUMBER()` function is used within the CTE to rank orders per customer based on recency. The `COUNT()` function in the main query calculates the total number of orders per customer.
- **Aggregation:** Grouping in the main query is by `CustomerID` and `CustomerCompanyName`, along with the last order date from the CTE, to summarize the total orders and identify the last order date per customer.
- **Output:** The resulting report will display each customer's ID, company name, total order count, and the date of their last order, sorted primarily by the total number of orders and then by the most recent order date, to prioritize customers with higher engagement and recent activity.

**Why is this a top problem?**

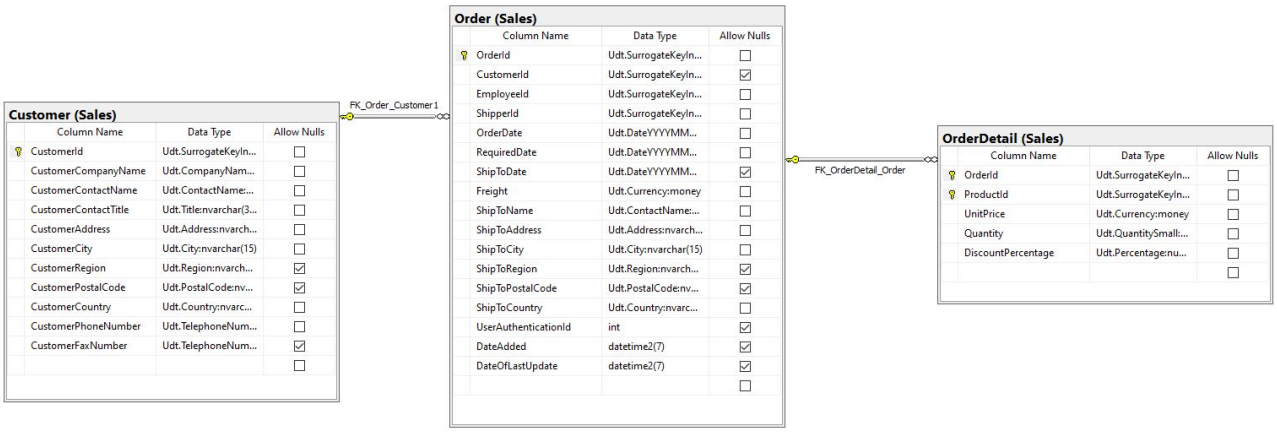
I find the query too simplistic as it only considers the number of orders and the last order date, which doesn't offer a comprehensive view of customer activity or value. To make the analysis more robust, I plan to include additional metrics like total spending per customer and average order value. This will involve either expanding the existing `LastOrderDetails` CTE or adding a new one to aggregate these financial metrics, providing a more nuanced and insightful perspective on customer engagement and their financial contribution to the business.

**Subsystem in Northwinds2022TSQLV7**



### Diagram of Tables


### Columns from Standard view




# Table: Column Projection

| Table Name | Column Name | Derived Column | Function/Calculation |
|---|---|---|---|
| Sales.[Order] | CustomerID, OrderID, OrderDate | | |
| Sales.OrderDetail | OrderID | | |
| Sales.Customer | CustomerID | | |
| Sales.Customer | CustomerCompanyName | | |

# Table: Group By Order By

| Group By Column | Order By Column |
|---|---|
| CustomerID, CustomerCompanyName, OrderDate | NumberOfOrders DESC, LastOrderDate DESC |

In [22]:
```
USE Northwinds2022TSQLV7
go

WITH LastOrderDetails AS (
    SELECT
        o.CustomerID,
        o.OrderID,
        o.OrderDate,
        ROW_NUMBER() OVER (PARTITION BY o.CustomerID ORDER BY o.OrderDate DESC) AS RowNu
    FROM Sales.[Order] o
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
)
SELECT
    c.CustomerID,
    c.CustomerCompanyName,
```

```
    COUNT(o.OrderID) AS NumberOfOrders,
    lod.OrderDate AS LastOrderDate
FROM Sales.Customer c
JOIN Sales.[Order] o ON c.CustomerID = o.CustomerID
JOIN LastOrderDetails lod ON c.CustomerID = lod.CustomerID AND lod.RowNum = 1
GROUP BY c.CustomerID, c.CustomerCompanyName, lod.OrderDate
ORDER BY NumberOfOrders DESC, LastOrderDate DESC;
```

Commands completed successfully.

(89 rows affected)

Total execution time: 00:00:00.014

Out[22]:

| CustomerID | CustomerCompanyName | NumberOfOrders | LastOrderDate |
|---|---|---|---|
| 71 | Customer LCOUJ | 31 | 2016-05-01 |
| 20 | Customer THHDP | 30 | 2016-05-05 |
| 63 | Customer IRRVL | 28 | 2016-04-14 |
| 37 | Customer FRXZL | 19 | 2016-04-30 |
| 24 | Customer CYZTN | 19 | 2016-04-27 |
| 65 | Customer NYUHS | 18 | 2016-05-06 |
| 35 | Customer UMTLM | 18 | 2016-04-28 |
| 5 | Customer HGVLZ | 18 | 2016-03-04 |
| 9 | Customer RTXGC | 17 | 2016-05-06 |
| 44 | Customer OXFRU | 15 | 2016-05-05 |
| 87 | Customer ZHYOS | 15 | 2016-04-15 |
| 25 | Customer AZJED | 15 | 2016-04-09 |
| 46 | Customer XPNIK | 14 | 2016-05-05 |
| 89 | Customer YBQTI | 14 | 2016-05-01 |
| 34 | Customer IBVRG | 14 | 2016-04-27 |
| 41 | Customer XIIWM | 14 | 2016-04-27 |
| 10 | Customer EEALV | 14 | 2016-04-24 |
| 39 | Customer GLLAG | 14 | 2016-04-16 |
| 62 | Customer WFIZJ | 13 | 2016-05-04 |
| 4 | Customer HFBZG | 13 | 2016-04-10 |
| 51 | Customer PVDZC | 13 | 2015-10-30 |
| 66 | Customer LHANT | 12 | 2016-04-30 |
| 76 | Customer SFOGW | 12 | 2016-04-21 |
| 47 | Customer PSQUZ | 12 | 2016-04-21 |
| 32 | Customer YSIQX | 11 | 2016-04-30 |
| 67 | Customer QVEPD | 11 | 2016-04-29 |
| 83 | Customer ZRNDE | 11 | 2016-04-02 |
| 7 | Customer QXVLA | 11 | 2016-01-12 |
| 68 | Customer CCKOT | 10 | 2016-05-06 |
| 80 | Customer VONTK | 10 | 2016-05-04 |
| 59 | Customer LOLJO | 10 | 2016-04-27 |
| 86 | Customer SNXOJ | 10 | 2016-04-23 |

| | | | |
|---|---|---|---|
| 74 | Customer YSHXL | 4 | 2016-04-22 |
| 77 | Customer LCYBZ | 4 | 2016-04-01 |
| 40 | Customer EFFTC | 4 | 2016-03-24 |
| 2 | Customer MLTDN | 4 | 2016-03-04 |
| 18 | Customer BSVAR | 4 | 2016-02-16 |
| 45 | Customer QXPPT | 4 | 2016-02-12 |
| 53 | Customer GCJSG | 3 | 2016-04-29 |
| 78 | Customer NLTYP | 3 | 2016-04-06 |
| 8 | Customer QUHWH | 3 | 2016-03-24 |
| 26 | Customer USDBG | 3 | 2016-03-24 |
| 16 | Customer GYBBY | 3 | 2016-01-23 |
| 82 | Customer EYHKM | 3 | 2016-01-08 |
| 42 | Customer IAIJK | 3 | 2016-01-01 |
| 33 | Customer FVXPQ | 2 | 2015-12-18 |
| 43 | Customer UISOJ | 2 | 2015-05-22 |
| 13 | Customer VMLOG | 1 | 2014-07-18 |

In [12]:
```
USE Northwinds2022TSQLV7;
GO

WITH LastOrderDetails AS (
    SELECT
        o.CustomerID,
        o.OrderID,
        o.OrderDate,
        ROW_NUMBER() OVER (PARTITION BY o.CustomerID ORDER BY o.OrderDate DESC) AS RowNu
    FROM Sales.[Order] o
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
)
SELECT
    c.CustomerID,
    c.CustomerCompanyName,
    COUNT(o.OrderID) AS NumberOfOrders,
    lod.OrderDate AS LastOrderDate
FROM Sales.Customer c
JOIN Sales.[Order] o ON c.CustomerID = o.CustomerID
JOIN LastOrderDetails lod ON c.CustomerID = lod.CustomerID AND lod.RowNum = 1
GROUP BY c.CustomerID, c.CustomerCompanyName, lod.OrderDate
ORDER BY NumberOfOrders DESC, LastOrderDate DESC
FOR JSON PATH;
```

Commands completed successfully.

(89 rows affected)

Total execution time: 00:00:00.022

Out[12]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**


```
[{"CustomerID":71,"CustomerCompanyName":"Customer LCOUJ","NumberOfOrders":31,"LastOrderDate":"2016-05-01"},
{"CustomerID":20,"CustomerCompanyName":"Customer THHDP","NumberOfOrders":30,"LastOrderDate":"2016-05-05"},
{"CustomerID":63,"CustomerCompanyName":"Customer IRRVL","NumberOfOrders":28,"LastOrderDate":"2016-04-14"},
{"CustomerID":37,"CustomerCompanyName":"Customer FRXZL","NumberOfOrders":19,"LastOrderDate":"2016-04-30"},
{"CustomerID":24,"CustomerCompanyName":"Customer CYZTN","NumberOfOrders":19,"LastOrderDate":"2016-04-27"},
{"CustomerID":65,"CustomerCompanyName":"Customer NYUHS","NumberOfOrders":18,"LastOrderDate":"2016-05-06"},
{"CustomerID":35,"CustomerCompanyName":"Customer UMTLM","NumberOfOrders":18,"LastOrderDate":"2016-04-28"},
{"CustomerID":5,"CustomerCompanyName":"Customer HGVLZ","NumberOfOrders":18,"LastOrderDate":"2016-03-04"},
{"CustomerID":9,"CustomerCompanyName":"Customer RTXGC","NumberOfOrders":17,"LastOrderDate":"2016-05-06"},
```

{"CustomerID":82,"CustomerCompanyName":"Customer EYHKM","NumberOfOrders":3,"LastOrderDate":"2016-01-08"},
{"CustomerID":42,"CustomerCompanyName":"Customer IAIJK","NumberOfOrders":3,"LastOrderDate":"2016-01-01"},
{"CustomerID":33,"CustomerCompanyName":"Customer FVXPQ","NumberOfOrders":2,"LastOrderDate":"2015-12-18"},
{"CustomerID":43,"CustomerCompanyName":"Customer UISOJ","NumberOfOrders":2,"LastOrderDate":"2015-05-22"},
{"CustomerID":13,"CustomerCompanyName":"Customer VMLOG","NumberOfOrders":1,"LastOrderDate":"2014-07-18"}]

# \===== Remaining 14 Problems =====

## **Complex Proposition 3:** Construct a detailed purchase analysis report in the Northwinds2022TSQLV7 database that calculates the total sales amount per customer, utilizing a custom function to compute individual purchase amounts.

- **Custom Function Creation:** `dbo.CalculateTotalPurchaseAmount` is introduced to multiply the unit price by quantity, providing a granular calculation of each order's total purchase amount.
- **CTE Usage:** `CustomerPurchaseSummary` CTE is established to aggregate the total purchase amounts per customer, calling the `dbo.CalculateTotalPurchaseAmount` function for each order detail record to ensure accurate total calculations.
- **Tables Engaged:** The `Sales.Customer` table is joined with `Sales.[Order]` and `Sales.OrderDetail` to collect comprehensive data on customer orders and their detailed line items, facilitating a full assessment of customer purchasing behavior.
- **Data Aggregation:** The main query groups the accumulated data by `CustomerID` and `CustomerCompanyName`, summarizing the total purchase amounts to reflect the overall sales contribution of each customer.
- **Output Goal:** The final output of the report showcases the `CustomerID`, `CustomerCompanyName`, and the `TotalSalesAmount`, representing the sum of purchase amounts calculated for each customer, enabling the business to identify key customers based on their total spending.

In [1]:
```
USE Northwinds2022TSQLV7;
go
CREATE OR ALTER FUNCTION dbo.CalculateTotalPurchaseAmount (@unitPrice MONEY, @quantity I
RETURNS MONEY
AS
BEGIN
    RETURN @unitPrice * @quantity;
END;
GO

WITH CustomerPurchaseSummary AS (
    SELECT
        c.CustomerID,
        c.CustomerCompanyName,
        SUM(dbo.CalculateTotalPurchaseAmount(od.UnitPrice, od.Quantity)) AS TotalPurchas
    FROM Sales.Customer c
    JOIN Sales.[Order] o ON c.CustomerID = o.CustomerID
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY c.CustomerID, c.CustomerCompanyName
)

SELECT
    CustomerID,
    CustomerCompanyName,
    SUM(TotalPurchaseAmount) AS TotalSalesAmount
FROM CustomerPurchaseSummary
GROUP BY CustomerID, CustomerCompanyName;
```

Commands completed successfully.
Commands completed successfully.
(89 rows affected)
Total execution time: 00:00:01.409

Out[1]:

| CustomerID | CustomerCompanyName | TotalSalesAmount |
|---|---|---|
| 23 | Customer WVFAF | 11666.90 |
| 46 | Customer XPNIK | 17825.06 |
| 69 | Customer SIUIH | 1467.29 |
| 29 | Customer MDLWA | 836.70 |
| 75 | Customer XOJYP | 12489.70 |
| 15 | Customer JUWXK | 3810.75 |
| 9 | Customer RTXGC | 23850.95 |
| 89 | Customer YBQTI | 29073.45 |
| 3 | Customer KBUDE | 7515.35 |
| 52 | Customer PZNLA | 5042.20 |
| 72 | Customer AHPOP | 17172.05 |
| 66 | Customer LHANT | 7555.60 |
| 78 | Customer NLTYP | 1947.24 |
| 32 | Customer YSIQX | 19711.13 |
| 26 | Customer USDBG | 3172.16 |
| 12 | Customer PSNMQ | 1814.80 |
| 35 | Customer UMTLM | 23611.58 |
| 86 | Customer SNXOJ | 10653.85 |
| 63 | Customer IRRVL | 117483.39 |
| 6 | Customer XHXJV | 3239.80 |
| 55 | Customer KZQZT | 16325.15 |
| 43 | Customer UISOJ | 357.00 |
| 49 | Customer CQRAA | 7603.85 |
| 67 | Customer QVEPD | 12924.40 |
| 21 | Customer KIDPX | 4438.90 |
| 27 | Customer WMFEA | 1545.70 |
| 58 | Customer AHXHT | 4242.20 |
| 81 | Customer YQQWW | 7310.62 |
| 64 | Customer LWGMD | 2844.10 |
| 87 | Customer ZHYOS | 16617.10 |
| 38 | Customer LJUCA | 6146.30 |
| 7 | Customer QXVLA | 19088.00 |
| 44 | Customer OXFRU | 21282.02 |
| 50 | Customer JYPSC | 10430.58 |
| 1 | Customer NRZBB | 4596.20 |
| 24 | Customer CYZTN | 32555.55 |

| | | |
|---|---|---|
| 11 | Customer UBHAU | 6089.90 |
| 54 | Customer TDKEG | 3460.20 |
| 91 | Customer CCFIZ | 3531.95 |
| 20 | Customer THHDP | 113236.68 |
| 68 | Customer CCKOT | 20033.20 |
| 80 | Customer VONTK | 10812.15 |
| 28 | Customer XYUFB | 7151.55 |
| 74 | Customer YSHXL | 2423.35 |
| 14 | Customer WNMAF | 12886.30 |
| 37 | Customer FRXZL | 57317.39 |
| 8 | Customer QUHWH | 5297.80 |
| 51 | Customer PVDZC | 32203.90 |

In [13]:

```
USE Northwinds2022TSQLV7;
GO
CREATE OR ALTER FUNCTION dbo.CalculateTotalPurchaseAmount (@unitPrice MONEY, @quantity I
RETURNS MONEY
AS
BEGIN
    RETURN @unitPrice * @quantity;
END;
GO

WITH CustomerPurchaseSummary AS (
    SELECT
        c.CustomerID,
        c.CustomerCompanyName,
        SUM(dbo.CalculateTotalPurchaseAmount(od.UnitPrice, od.Quantity)) AS TotalPurchas
    FROM Sales.Customer c
    JOIN Sales.[Order] o ON c.CustomerID = o.CustomerID
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY c.CustomerID, c.CustomerCompanyName
)

SELECT
    CustomerID,
    CustomerCompanyName,
    SUM(TotalPurchaseAmount) AS TotalSalesAmount
FROM CustomerPurchaseSummary
GROUP BY CustomerID, CustomerCompanyName
FOR JSON PATH;
```

Commands completed successfully.

Commands completed successfully.

(89 rows affected)

Total execution time: 00:00:00.102

Out[13]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**


```
[{"CustomerID":23,"CustomerCompanyName":"Customer WVFAF","TotalSalesAmount":11666.9000},
 {"CustomerID":46,"CustomerCompanyName":"Customer XPNIK","TotalSalesAmount":17825.0600},
  {"CustomerID":69,"CustomerCompanyName":"Customer SIUIH","TotalSalesAmount":1467.2900},
 {"CustomerID":29,"CustomerCompanyName":"Customer MDLWA","TotalSalesAmount":836.7000},
{"CustomerID":75,"CustomerCompanyName":"Customer XOJYP","TotalSalesAmount":12489.7000},
 {"CustomerID":15,"CustomerCompanyName":"Customer JUWXK","TotalSalesAmount":3810.7500},
 {"CustomerID":9,"CustomerCompanyName":"Customer RTXGC","TotalSalesAmount":23850.9500},
 {"CustomerID":89,"CustomerCompanyName":"Customer YBQTI","TotalSalesAmount":29073.4500},
  {"CustomerID":3,"CustomerCompanyName":"Customer KBUDE","TotalSalesAmount":7515.3500},
```

{"CustomerID":74,"CustomerCompanyName":"Customer YSHXL","TotalSalesAmount":2423.3500},
{"CustomerID":14,"CustomerCompanyName":"Customer WNMAF","TotalSalesAmount":12886.3000},
{"CustomerID":37,"CustomerCompanyName":"Customer FRXZL","TotalSalesAmount":57317.3900},
{"CustomerID":8,"CustomerCompanyName":"Customer QUHWH","TotalSalesAmount":5297.8000},
{"CustomerID":51,"CustomerCompanyName":"Customer PVDZC","TotalSalesAmount":32203.9000}]

## **Complex Proposition 4:** Implement a comprehensive sales and discount analysis in the Northwinds2022TSQLV7 database by calculating the total price after discount for each order and summarizing sales and discounts per customer and order.

- **Custom Function Creation:** `dbo.CalculateTotalPriceAfterDiscount` is crafted to compute the total price after applying a discount to each purchase, factoring in unit price, quantity, and discount rate.
- **CTE Usage:** `OrderDetailsWithTotal` CTE is established to apply the `dbo.CalculateTotalPriceAfterDiscount` function across the `Sales.OrderDetail` table, calculating the total price after discount for each order line item.
- **Tables Engaged:** The query integrates data from `Sales.Customer` and `Sales.[Order]`, linking customer information with their respective orders, and ties in the `OrderDetailsWithTotal` CTE to access the computed sales and discount figures.
- **Data Aggregation:** In the main query, data is aggregated by customer and order, summing up the total sales after discount and the total discount value for each order, providing a dual perspective on revenue and cost-saving through discounts.
- **Output Goal:** The final report will display each customer's ID, company name, order ID, total sales after discount, and the total discount amount, sorted by total sales in descending order to highlight the most significant transactions. This detailed view allows for an in-depth analysis of sales effectiveness and discount impact on the overall revenue.

In [25]:
```
use Northwinds2022TSQLV7
GO

CREATE OR ALTER FUNCTION dbo.CalculateTotalPriceAfterDiscount (@unitPrice MONEY, @quanti
RETURNS MONEY
AS
BEGIN
    RETURN @unitPrice * @quantity * (1 - @discount)
END
go
-- CTE for calculating total price after discount in order details
WITH OrderDetailsWithTotal AS (
    SELECT
        od.OrderID,
        od.UnitPrice,
        od.Quantity,
        od.DiscountPercentage,
        dbo.CalculateTotalPriceAfterDiscount(od.UnitPrice, od.Quantity, od.DiscountPerce
    FROM Sales.OrderDetail od
)
SELECT
    c.CustomerID,
    c.CustomerCompanyName,
    o.OrderID,
    SUM(odt.TotalPriceAfterDiscount) AS TotalSales,
    SUM(odt.UnitPrice * odt.Quantity - odt.TotalPriceAfterDiscount) AS TotalDiscount
FROM Sales.Customer c
JOIN Sales.[Order] o ON c.CustomerID = o.CustomerID
JOIN OrderDetailsWithTotal odt ON o.OrderID = odt.OrderID
```

```
GROUP BY c.CustomerID, c.CustomerCompanyName, o.OrderID
ORDER BY TotalSales DESC;
```

Commands completed successfully.

Commands completed successfully.

(830 rows affected)

Total execution time: 00:00:00.108

Out[25]:

| CustomerID | CustomerCompanyName | OrderID | TotalSales | TotalDiscount |
|---|---|---|---|---|
| 63 | Customer IRRVL | 10865 | 16387.50 | 862.50 |
| 34 | Customer IBVRG | 10981 | 15810.00 | 0.00 |
| 71 | Customer LCOUJ | 11030 | 12615.05 | 3706.85 |
| 65 | Customer NYUHS | 10889 | 11380.00 | 0.00 |
| 73 | Customer JMIKW | 10417 | 11188.40 | 94.80 |
| 39 | Customer GLLAG | 10817 | 10952.845 | 537.855 |
| 37 | Customer FRXZL | 10897 | 10835.24 | 0.00 |
| 65 | Customer NYUHS | 10479 | 10495.60 | 0.00 |
| 63 | Customer IRRVL | 10540 | 10191.70 | 0.00 |
| 63 | Customer IRRVL | 10691 | 10164.80 | 0.00 |
| 63 | Customer IRRVL | 10515 | 9921.30 | 667.20 |
| 62 | Customer WFIZJ | 10372 | 9210.90 | 3070.30 |
| 51 | Customer PVDZC | 10424 | 9194.56 | 2298.64 |
| 89 | Customer YBQTI | 11032 | 8902.50 | 0.00 |
| 20 | Customer THHDP | 10514 | 8623.45 | 0.00 |
| 59 | Customer LOLJO | 10353 | 8593.28 | 2148.32 |
| 32 | Customer YSIQX | 10816 | 8446.45 | 444.55 |
| 7 | Customer QXVLA | 10360 | 7390.20 | 0.00 |
| 20 | Customer THHDP | 11017 | 6750.00 | 0.00 |
| 20 | Customer THHDP | 10776 | 6635.275 | 349.225 |
| 71 | Customer LCOUJ | 10607 | 6475.40 | 0.00 |
| 20 | Customer THHDP | 10895 | 6379.40 | 0.00 |
| 71 | Customer LCOUJ | 10612 | 6375.00 | 0.00 |
| 63 | Customer IRRVL | 11021 | 6306.24 | 635.25 |
| 37 | Customer FRXZL | 10912 | 6200.55 | 2066.85 |
| 20 | Customer THHDP | 10633 | 5510.5925 | 972.4575 |
| 39 | Customer GLLAG | 10893 | 5502.11 | 0.00 |
| 20 | Customer THHDP | 10351 | 5398.725 | 278.875 |
| 71 | Customer LCOUJ | 10324 | 5275.715 | 880.185 |
| 71 | Customer LCOUJ | 10678 | 5256.50 | 0.00 |
| 20 | Customer THHDP | 11072 | 5218.00 | 0.00 |
| 23 | Customer WVFAF | 10634 | 4985.50 | 0.00 |
| 37 | Customer FRXZL | 10687 | 4960.90 | 1241.00 |
| 71 | Customer LCOUJ | 10847 | 4931.92 | 1232.98 |

| | | | | |
|---|---|---|---|---|
| 24 | Customer CYZTN | 10955 | 74.40 | 18.60 |
| 41 | Customer XIIWM | 10371 | 72.96 | 18.24 |
| 32 | Customer YSIQX | 10589 | 72.00 | 0.00 |
| 29 | Customer MDLWA | 10887 | 70.00 | 0.00 |
| 77 | Customer LCYBZ | 10992 | 69.60 | 0.00 |
| 18 | Customer BSVAR | 10683 | 63.00 | 0.00 |
| 30 | Customer KSLQF | 11037 | 60.00 | 0.00 |
| 28 | Customer XYUFB | 10963 | 57.80 | 10.20 |
| 42 | Customer IAIJK | 10620 | 57.50 | 0.00 |
| 41 | Customer XIIWM | 10631 | 55.80 | 6.20 |
| 49 | Customer CQRAA | 10754 | 55.20 | 0.00 |
| 74 | Customer YSHXL | 10738 | 52.35 | 0.00 |
| 27 | Customer WMFEA | 10422 | 49.80 | 0.00 |
| 83 | Customer ZRNDE | 10602 | 48.75 | 16.25 |
| 75 | Customer XOJYP | 10271 | 48.00 | 0.00 |
| 38 | Customer LJUCA | 10674 | 45.00 | 0.00 |
| 53 | Customer GCJSG | 11057 | 45.00 | 0.00 |
| 71 | Customer LCOUJ | 10815 | 40.00 | 0.00 |
| 41 | Customer XIIWM | 11051 | 36.00 | 9.00 |
| 48 | Customer DVFMB | 10883 | 36.00 | 0.00 |
| 88 | Customer SRQVM | 10900 | 33.75 | 11.25 |
| 54 | Customer TDKEG | 10898 | 30.00 | 0.00 |
| 76 | Customer SFOGW | 10767 | 28.00 | 0.00 |
| 66 | Customer LHANT | 10586 | 23.80 | 4.20 |
| 27 | Customer WMFEA | 10807 | 18.40 | 0.00 |
| 12 | Customer PSNMQ | 10782 | 12.50 | 0.00 |

In [14]:

```
USE Northwinds2022TSQLV7;
GO

CREATE OR ALTER FUNCTION dbo.CalculateTotalPriceAfterDiscount (@unitPrice MONEY, @quanti
RETURNS MONEY
AS
BEGIN
    RETURN @unitPrice * @quantity * (1 - @discount);
END;
GO

-- CTE for calculating total price after discount in order details
WITH OrderDetailsWithTotal AS (
    SELECT
        od.OrderID,
        od.UnitPrice,
        od.Quantity,
        od.DiscountPercentage,
        dbo.CalculateTotalPriceAfterDiscount(od.UnitPrice, od.Quantity, od.DiscountPerce
    FROM Sales.OrderDetail od
)
SELECT
```

```
    c.CustomerID,
    c.CustomerCompanyName,
    o.OrderID,
    SUM(odt.TotalPriceAfterDiscount) AS TotalSales,
    SUM(odt.UnitPrice * odt.Quantity - odt.TotalPriceAfterDiscount) AS TotalDiscount
FROM Sales.Customer c
JOIN Sales.[Order] o ON c.CustomerID = o.CustomerID
JOIN OrderDetailsWithTotal odt ON o.OrderID = odt.OrderID
GROUP BY c.CustomerID, c.CustomerCompanyName, o.OrderID
ORDER BY TotalSales DESC
FOR JSON PATH;
```

Commands completed successfully.

Commands completed successfully.

(830 rows affected)

Total execution time: 00:00:00.094

Out[14]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**


```
[{"CustomerID":63,"CustomerCompanyName":"Customer IRRVL","OrderID":10865,"TotalSales":16387.5000,"TotalDiscount":862.5000},
{"CustomerID":34,"CustomerCompanyName":"Customer IBVRG","OrderID":10981,"TotalSales":15810.0000,"TotalDiscount":0.0000},
{"CustomerID":71,"CustomerCompanyName":"Customer LCOUJ","OrderID":11030,"TotalSales":12615.0500,"TotalDiscount":3706.8500},
{"CustomerID":65,"CustomerCompanyName":"Customer NYUHS","OrderID":10889,"TotalSales":11380.0000,"TotalDiscount":0.0000},
{"CustomerID":73,"CustomerCompanyName":"Customer JMIKW","OrderID":10417,"TotalSales":11188.4000,"TotalDiscount":94.8000},
{"CustomerID":39,"CustomerCompanyName":"Customer GLLAG","OrderID":10817,"TotalSales":10952.8450,"TotalDiscount":537.8550},
{"CustomerID":37,"CustomerCompanyName":"Customer FRXZL","OrderID":10897,"TotalSales":10835.2400,"TotalDiscount":0.0000},
{"CustomerID":65,"CustomerCompanyName":"Customer NYUHS","OrderID":10479,"TotalSales":10495.6000,"TotalDiscount":0.0000},
{"CustomerID":63,"CustomerCompanyName":"Customer IRRVL","OrderID":10540,"TotalSales":10191.7000,"TotalDiscount":0.0000},
{"CustomerID":63,"CustomerCompanyName":"Customer IRRVL","OrderID":10691,"TotalSales":10164.8000,"TotalDiscount":0.0000},
{"CustomerID":63,"CustomerCompanyName":"Customer IRRVL","OrderID":10515,"TotalSales":9921.3000,"TotalDiscount":667.2000},
{"CustomerID":62,"CustomerCompanyName":"Customer WFIZJ","OrderID":10372,"TotalSales":9210.9000,"TotalDiscount":3070.3000},
{"CustomerID":51,"CustomerCompanyName":"Customer PVDZC","OrderID":10424,"TotalSales":9194.5600,"TotalDiscount":2298.6400},
{"CustomerID":89,"CustomerCompanyName":"Customer YBQTI","OrderID":11032,"TotalSales":8902.5000,"TotalDiscount":0.0000},
{"CustomerID":20,"CustomerCompanyName":"Customer THHDP","OrderID":10514,"TotalSales":8623.4500,"TotalDiscount":0.0000},
{"CustomerID":59,"CustomerCompanyName":"Customer LOLJO","OrderID":10353,"TotalSales":8593.2800,"TotalDiscount":2148.3200},
{"CustomerID":32,"CustomerCompanyName":"Customer YSIQX","OrderID":10816,"TotalSales":8446.4500,"TotalDiscount":444.5500},
{"CustomerID":7,"CustomerCompanyName":"Customer QXVLA","OrderID":10360,"TotalSales":7390.2000,"TotalDiscount":0.0000},
{"CustomerID":20,"CustomerCompanyName":"Customer THHDP","OrderID":11017,"TotalSales":6750.0000,"TotalDiscount":0.0000},
{"CustomerID":20,"CustomerCompanyName":"Customer THHDP","OrderID":10776,"TotalSales":6635.2750,"TotalDiscount":349.2250},
{"CustomerID":71,"CustomerCompanyName":"Customer LCOUJ","OrderID":10607,"TotalSales":6475.4000,"TotalDiscount":0.0000},
{"CustomerID":20,"CustomerCompanyName":"Customer THHDP","OrderID":10895,"TotalSales":6379.4000,"TotalDiscount":0.0000},
{"CustomerID":71,"CustomerCompanyName":"Customer LCOUJ","OrderID":10612,"TotalSales":6375.0000,"TotalDiscount":0.0000},
{"CustomerID":63,"CustomerCompanyName":"Customer IRRVL","OrderID":11021,"TotalSales":6306.2400,"TotalDiscount":635.2500},
{"CustomerID":37,"CustomerCompanyName":"Customer FRXZL","OrderID":10912,"TotalSales":6200.5500,"TotalDiscount":2066.8500},
{"CustomerID":20,"CustomerCompanyName":"Customer THHDP","OrderID":10633,"TotalSales":5510.5925,"TotalDiscount":972.4575},
{"CustomerID":39,"CustomerCompanyName":"Customer
```

{"CustomerID":49,"CustomerCompanyName":"Customer CQRAA","OrderID":10754,"TotalSales":55.2000,"TotalDiscount":0.0000},
{"CustomerID":74,"CustomerCompanyName":"Customer YSHXL","OrderID":10738,"TotalSales":52.3500,"TotalDiscount":0.0000},
{"CustomerID":27,"CustomerCompanyName":"Customer WMFEA","OrderID":10422,"TotalSales":49.8000,"TotalDiscount":0.0000},
{"CustomerID":83,"CustomerCompanyName":"Customer ZRNDE","OrderID":10602,"TotalSales":48.7500,"TotalDiscount":16.2500},
{"CustomerID":75,"CustomerCompanyName":"Customer XOJYP","OrderID":10271,"TotalSales":48.0000,"TotalDiscount":0.0000},
{"CustomerID":38,"CustomerCompanyName":"Customer LJUCA","OrderID":10674,"TotalSales":45.0000,"TotalDiscount":0.0000},
{"CustomerID":53,"CustomerCompanyName":"Customer GCJSG","OrderID":11057,"TotalSales":45.0000,"TotalDiscount":0.0000},
{"CustomerID":71,"CustomerCompanyName":"Customer LCOUJ","OrderID":10815,"TotalSales":40.0000,"TotalDiscount":0.0000},
{"CustomerID":41,"CustomerCompanyName":"Customer XIIWM","OrderID":11051,"TotalSales":36.0000,"TotalDiscount":9.0000},
{"CustomerID":48,"CustomerCompanyName":"Customer DVFMB","OrderID":10883,"TotalSales":36.0000,"TotalDiscount":0.0000},
{"CustomerID":88,"CustomerCompanyName":"Customer SRQVM","OrderID":10900,"TotalSales":33.7500,"TotalDiscount":11.2500},
{"CustomerID":54,"CustomerCompanyName":"Customer TDKEG","OrderID":10898,"TotalSales":30.0000,"TotalDiscount":0.0000},
{"CustomerID":76,"CustomerCompanyName":"Customer SFOGW","OrderID":10767,"TotalSales":28.0000,"TotalDiscount":0.0000},
{"CustomerID":66,"CustomerCompanyName":"Customer LHANT","OrderID":10586,"TotalSales":23.8000,"TotalDiscount":4.2000},
{"CustomerID":27,"CustomerCompanyName":"Customer WMFEA","OrderID":10807,"TotalSales":18.4000,"TotalDiscount":0.0000},
{"CustomerID":12,"CustomerCompanyName":"Customer PSNMQ","OrderID":10782,"TotalSales":12.5000,"TotalDiscount":0.0000}]

## **Complex Proposition 5:** Conduct a detailed analysis of product category sales performance in the AdventureWorksDW2017 database, specifically focusing on calculating total sales for each product category using a custom scalar function named `dbo.fn_GetTotalSales`.

- **Custom Function Creation:** The custom function `dbo.fn_GetTotalSales` calculates total sales for a given product category based on the product category key provided as input.
- **CTE Usage:** Utilize a Common Table Expression (CTE) named `CategorySales` to aggregate sales data for each product category. The CTE uses the custom function `dbo.fn_GetTotalSales` to compute total sales per category.
- **Tables Engaged:** The analysis integrates data from the DimProductCategory, FactInternetSales, DimProduct, and DimProductSubcategory tables. These tables are joined to calculate total sales for each product category accurately.
- **Data Aggregation:** The main query uses the CTE `CategorySales` to compile metrics by product category, including the product category key, name, and total sales. This offers insights into the sales performance of different product categories.
- **Output Goal:** The outcome is a report listing product category key, name, and total sales, sorted by total sales in descending order to identify the top-performing product categories in terms of sales.

In [3]:
```
-- Create a custom scalar function to calculate total sales for a given product category
USE AdventureWorksDW2017;
go
CREATE OR ALTER FUNCTION dbo.fn_GetTotalSales (@ProductCategoryKey INT)
RETURNS DECIMAL(18,2)
AS
BEGIN
    DECLARE @TotalSales DECIMAL(18,2);
    SELECT @TotalSales = SUM(FactInternetSales.SalesAmount)
    FROM FactInternetSales
```

```
    JOIN DimProduct ON FactInternetSales.ProductKey = DimProduct.ProductKey
    JOIN DimProductSubcategory ON DimProduct.ProductSubcategoryKey = DimProductSubcatego
    WHERE DimProductSubcategory.ProductCategoryKey = @ProductCategoryKey;
    RETURN @TotalSales;
END;
GO

-- Query to retrieve product categories with their total sales
WITH CategorySales AS (
    SELECT
        DimProductCategory.ProductCategoryKey,
        DimProductCategory.EnglishProductCategoryName AS ProductCategory,
        dbo.fn_GetTotalSales(DimProductCategory.ProductCategoryKey) AS TotalSales
    FROM
        DimProductCategory
)
SELECT
    cs.ProductCategoryKey,
    cs.ProductCategory,
    cs.TotalSales
FROM
    CategorySales cs
ORDER BY
    cs.TotalSales DESC;
```

Commands completed successfully.

Commands completed successfully.

(4 rows affected)

Total execution time: 00:00:00.083

Out[3]:

| ProductCategoryKey | ProductCategory | TotalSales |
|---|---|---|
| 1 | Bikes | 28318144.65 |
| 4 | Accessories | 700759.96 |
| 3 | Clothing | 339772.61 |
| 2 | Components | NULL |

In [15]:

```
USE AdventureWorksDW2017;
GO

CREATE OR ALTER FUNCTION dbo.fn_GetTotalSales (@ProductCategoryKey INT)
RETURNS DECIMAL(18,2)
AS
BEGIN
    DECLARE @TotalSales DECIMAL(18,2);
    SELECT @TotalSales = SUM(FactInternetSales.SalesAmount)
    FROM FactInternetSales
    JOIN DimProduct ON FactInternetSales.ProductKey = DimProduct.ProductKey
    JOIN DimProductSubcategory ON DimProduct.ProductSubcategoryKey = DimProductSubcatego
    WHERE DimProductSubcategory.ProductCategoryKey = @ProductCategoryKey;
    RETURN @TotalSales;
END;
GO

-- Query to retrieve product categories with their total sales
WITH CategorySales AS (
    SELECT
        DimProductCategory.ProductCategoryKey,
        DimProductCategory.EnglishProductCategoryName AS ProductCategory,
        dbo.fn_GetTotalSales(DimProductCategory.ProductCategoryKey) AS TotalSales
    FROM
        DimProductCategory
)
```

```
SELECT
    cs.ProductCategoryKey,
    cs.ProductCategory,
    cs.TotalSales
FROM
    CategorySales cs
ORDER BY
    cs.TotalSales DESC
FOR JSON PATH;
```

Commands completed successfully.

Commands completed successfully.

(4 rows affected)

Total execution time: 00:00:00.086

Out[15]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**

[{"ProductCategoryKey":1,"ProductCategory":"Bikes","TotalSales":28318144.65},
{"ProductCategoryKey":4,"ProductCategory":"Accessories","TotalSales":700759.96},
{"ProductCategoryKey":3,"ProductCategory":"Clothing","TotalSales":339772.61},
{"ProductCategoryKey":2,"ProductCategory":"Components"}]

## Medium Proposition 5: Perform a comprehensive analysis of product categories in Northwinds2022TSQLV7, focusing on the number of products and suppliers, and total sales per category, to evaluate market performance and category health.

- **CTE Usage:** The `CategoryProductCounts` CTE calculates the number of distinct products and suppliers per category within the `Production.Product` table. Simultaneously, the `CategorySales` CTE aggregates the total sales for each category, utilizing the `Production.Product` and `Sales.OrderDetail` tables.
- **Tables Engaged:** This analysis merges insights from `Production.Category`, `Production.Product`, and `Sales.OrderDetail` to form a comprehensive view of sales and supply chain metrics across product categories.
- **Data Aggregation:** By grouping data at the category level, the approach enables a detailed assessment of the number of products and suppliers, alongside sales totals, facilitating a granular understanding of category performance.
- **Output Goal:** The resulting output, structured around Category ID, Category Name, Number of Products, Number of Suppliers, and Total Sales, is sorted by total sales in descending order to highlight the most successful categories, offering strategic insights into product distribution and sales efficacy.

In [1]:

```
USE Northwinds2022TSQLV7;
GO

WITH CategoryProductCounts AS (
    SELECT
        CategoryID,
        COUNT(DISTINCT ProductID) AS NumberOfProducts,
        COUNT(DISTINCT SupplierID) AS NumberOfSuppliers
    FROM Production.Product
    GROUP BY CategoryID
),
CategorySales AS (
    SELECT
        prod.CategoryID,
        SUM(od.Quantity * od.UnitPrice) AS TotalSales
    FROM Production.Product prod
    JOIN Sales.OrderDetail od ON prod.ProductID = od.ProductID
```

```
    GROUP BY prod.CategoryID
)
SELECT
    cat.CategoryID,
    cat.CategoryName,
    pc.NumberOfProducts,
    pc.NumberOfSuppliers,
    cs.TotalSales
FROM Production.Category cat
JOIN CategoryProductCounts pc ON cat.CategoryID = pc.CategoryID
JOIN CategorySales cs ON cat.CategoryID = cs.CategoryID
ORDER BY cs.TotalSales DESC;
```

Commands completed successfully.

(8 rows affected)

Total execution time: 00:00:00.025

Out[1]:

| CategoryID | CategoryName | NumberOfProducts | NumberOfSuppliers | TotalSales |
|---|---|---|---|---|
| 1 | Beverages | 12 | 8 | 286526.95 |
| 4 | Dairy Products | 10 | 4 | 251330.50 |
| 6 | Meat/Poultry | 6 | 5 | 178188.80 |
| 3 | Confections | 13 | 6 | 177099.10 |
| 8 | Seafood | 12 | 8 | 141623.09 |
| 2 | Condiments | 12 | 8 | 113694.75 |
| 7 | Produce | 5 | 5 | 105268.60 |
| 5 | Grains/Cereals | 7 | 5 | 100726.80 |

In [16]:

```
USE Northwinds2022TSQLV7;
GO

WITH CategoryProductCounts AS (
    SELECT
        CategoryID,
        COUNT(DISTINCT ProductID) AS NumberOfProducts,
        COUNT(DISTINCT SupplierID) AS NumberOfSuppliers
    FROM Production.Product
    GROUP BY CategoryID
),
CategorySales AS (
    SELECT
        prod.CategoryID,
        SUM(od.Quantity * od.UnitPrice) AS TotalSales
    FROM Production.Product prod
    JOIN Sales.OrderDetail od ON prod.ProductID = od.ProductID
    GROUP BY prod.CategoryID
)
SELECT
    cat.CategoryID,
    cat.CategoryName,
    pc.NumberOfProducts,
    pc.NumberOfSuppliers,
    cs.TotalSales
FROM Production.Category cat
JOIN CategoryProductCounts pc ON cat.CategoryID = pc.CategoryID
JOIN CategorySales cs ON cat.CategoryID = cs.CategoryID
ORDER BY cs.TotalSales DESC
FOR JSON PATH;
```

Commands completed successfully.

(8 rows affected)

Total execution time: 00:00:00.057

Out[16]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**

[{"CategoryID":1,"CategoryName":"Beverages","NumberOfProducts":12,"NumberOfSuppliers":8,"TotalSales":286526.9500},
{"CategoryID":4,"CategoryName":"Dairy Products","NumberOfProducts":10,"NumberOfSuppliers":4,"TotalSales":251330.5000},
{"CategoryID":6,"CategoryName":"Meat\/Poultry","NumberOfProducts":6,"NumberOfSuppliers":5,"TotalSales":178188.8000},
{"CategoryID":3,"CategoryName":"Confections","NumberOfProducts":13,"NumberOfSuppliers":6,"TotalSales":177099.1000},
{"CategoryID":8,"CategoryName":"Seafood","NumberOfProducts":12,"NumberOfSuppliers":8,"TotalSales":141623.0900},
{"CategoryID":2,"CategoryName":"Condiments","NumberOfProducts":12,"NumberOfSuppliers":8,"TotalSales":113694.7500},
{"CategoryID":7,"CategoryName":"Produce","NumberOfProducts":5,"NumberOfSuppliers":5,"TotalSales":105268.6000},
{"CategoryID":5,"CategoryName":"Grains\/Cereals","NumberOfProducts":7,"NumberOfSuppliers":5,"TotalSales":100726.8000}]

## Medium Proposition 6: Conduct an in-depth analysis of customer purchasing behavior in Northwinds2022TSQLV7 by calculating the total number of orders and average order size per customer, to discern spending patterns and assess customer value.

- **CTE Usage:** The `OrderSize` CTE is established to compute the total price for each order by aggregating product quantities and unit prices from `Sales.OrderDetail`, grouped by `CustomerID` and `OrderID`.
- **Tables Engaged:** This analysis utilizes `Sales.Customer` for customer details, `Sales.[Order]` for order information, and `Sales.OrderDetail` for line item data, providing a comprehensive view of customer transactions.
- **Data Aggregation:** In the primary query, data is collated by customer, summing the total number of orders and calculating the average order size, thereby quantifying each customer's purchasing volume and average spending.
- **Output Goal:** The end report will present the CustomerID, Total Orders, and Average Order Size, offering insights into how frequently customers place orders and how much they spend on average, enabling targeted customer relationship and sales strategies.

In [5]:

```
USE Northwinds2022TSQLV7
go

WITH OrderSize AS (
    SELECT
        o.CustomerID,
        o.OrderID,
        SUM(od.Quantity * od.UnitPrice) AS TotalPrice
    FROM Sales.[Order] o
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY o.CustomerID, o.OrderID
)
SELECT
    c.CustomerID,
    COUNT(o.OrderID) AS TotalOrders,
    AVG(os.TotalPrice) AS AverageOrderSize
FROM Sales.Customer c
JOIN Sales.[Order] o ON c.CustomerID = o.CustomerID
JOIN OrderSize os ON o.OrderID = os.OrderID
GROUP BY c.CustomerID;
```

Commands completed successfully.

(89 rows affected)

Total execution time: 00:00:00.022

Out[5]:

| CustomerID | TotalOrders | AverageOrderSize |
|---|---|---|

| | | |
|---|---|---|
| 23 | 5 | 2333.38 |
| 46 | 14 | 1273.2185 |
| 69 | 5 | 293.458 |
| 29 | 5 | 167.34 |
| 75 | 9 | 1387.7444 |
| 9 | 17 | 1402.997 |
| 15 | 5 | 762.15 |
| 89 | 14 | 2076.675 |
| 3 | 7 | 1073.6214 |
| 52 | 5 | 1008.44 |
| 72 | 9 | 1908.0055 |
| 66 | 12 | 629.6333 |
| 78 | 3 | 649.08 |
| 32 | 11 | 1791.9209 |
| 26 | 3 | 1057.3866 |
| 12 | 6 | 302.4666 |
| 35 | 18 | 1311.7544 |
| 86 | 10 | 1065.385 |
| 63 | 28 | 4195.8353 |
| 6 | 7 | 462.8285 |
| 55 | 10 | 1632.515 |
| 43 | 2 | 178.50 |
| 49 | 10 | 760.385 |
| 67 | 11 | 1174.9454 |
| 21 | 7 | 634.1285 |
| 27 | 6 | 257.6166 |
| 58 | 6 | 707.0333 |
| 81 | 6 | 1218.4366 |
| 64 | 5 | 568.82 |
| 38 | 10 | 614.63 |
| 87 | 15 | 1107.8066 |
| 7 | 11 | 1735.2727 |
| 44 | 15 | 1418.8013 |
| 1 | 6 | 766.0333 |
| 50 | 7 | 1490.0828 |
| 24 | 19 | 1713.45 |
| 47 | 12 | 1490.7958 |
| 70 | 6 | 955.8583 |
| 18 | 4 | 403.975 |
| 30 | 10 | 1183.01 |
| 84 | 10 | 993.71 |

| | | |
|---|---|---|
| 10 | 14 | 1614.8357 |
| 61 | 9 | 774.8477 |
| 41 | 14 | 733.7392 |
| 90 | 7 | 451.6214 |
| 4 | 13 | 1062.0384 |
| 65 | 18 | 2902.55 |
| 79 | 6 | 825.6666 |
| 19 | 8 | 1879.2075 |
| 73 | 7 | 2591.2071 |
| 25 | 15 | 1914.8473 |
| 36 | 5 | 612.64 |
| 85 | 5 | 296.00 |
| 62 | 13 | 2325.0846 |
| 13 | 1 | 100.80 |
| 42 | 3 | 174.1666 |
| 5 | 18 | 1498.2305 |
| 56 | 10 | 1315.75 |
| 76 | 12 | 2058.70 |
| 59 | 10 | 2625.995 |
| 82 | 3 | 523.7333 |
| 39 | 14 | 2267.5535 |
| 33 | 2 | 744.35 |
| 16 | 3 | 573.0333 |
| 88 | 9 | 720.0777 |
| 53 | 3 | 216.3333 |
| 45 | 4 | 872.505 |
| 2 | 4 | 350.7375 |
| 48 | 8 | 532.325 |
| 71 | 31 | 3731.3996 |
| 77 | 4 | 840.25 |
| 17 | 6 | 627.2016 |
| 31 | 9 | 966.9144 |
| 60 | 5 | 1063.42 |
| 83 | 11 | 1513.0727 |
| 34 | 14 | 2435.7964 |
| 40 | 4 | 498.0125 |
| 11 | 10 | 608.99 |
| 54 | 5 | 692.04 |
| 91 | 7 | 504.5642 |
| 20 | 30 | 3774.556 |

| | | |
|---|---|---|
| 68 | 10 | 2003.32 |
| 80 | 10 | 1081.215 |
| 28 | 8 | 893.9437 |
| 74 | 4 | 605.8375 |
| 14 | 8 | 1610.7875 |
| 37 | 19 | 3016.7047 |
| 8 | 3 | 1765.9333 |
| 51 | 13 | 2477.223 |

In [18]:

```
USE Northwinds2022TSQLV7;
GO

WITH OrderSize AS (
    SELECT
        o.CustomerID,
        o.OrderID,
        SUM(od.Quantity * od.UnitPrice) AS TotalPrice
    FROM Sales.[Order] o
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY o.CustomerID, o.OrderID
)
SELECT
    c.CustomerID,
    COUNT(o.OrderID) AS TotalOrders,
    AVG(os.TotalPrice) AS AverageOrderSize
FROM Sales.Customer c
JOIN Sales.[Order] o ON c.CustomerID = o.CustomerID
JOIN OrderSize os ON o.OrderID = os.OrderID
GROUP BY c.CustomerID
FOR JSON PATH;
```

Commands completed successfully.

(89 rows affected)

Total execution time: 00:00:00.023

Out[18]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**


[{"CustomerID":23,"TotalOrders":5,"AverageOrderSize":2333.3800},
{"CustomerID":46,"TotalOrders":14,"AverageOrderSize":1273.2185},
{"CustomerID":69,"TotalOrders":5,"AverageOrderSize":293.4580},
{"CustomerID":29,"TotalOrders":5,"AverageOrderSize":167.3400},
{"CustomerID":75,"TotalOrders":9,"AverageOrderSize":1387.7444},
{"CustomerID":9,"TotalOrders":17,"AverageOrderSize":1402.9970},
{"CustomerID":15,"TotalOrders":5,"AverageOrderSize":762.1500},
{"CustomerID":89,"TotalOrders":14,"AverageOrderSize":2076.6750},
{"CustomerID":3,"TotalOrders":7,"AverageOrderSize":1073.6214},
{"CustomerID":52,"TotalOrders":5,"AverageOrderSize":1008.4400},
{"CustomerID":72,"TotalOrders":9,"AverageOrderSize":1908.0055},
{"CustomerID":66,"TotalOrders":12,"AverageOrderSize":629.6333},
{"CustomerID":78,"TotalOrders":3,"AverageOrderSize":649.0800},
{"CustomerID":32,"TotalOrders":11,"AverageOrderSize":1791.9209},
{"CustomerID":26,"TotalOrders":3,"AverageOrderSize":1057.3866},
{"CustomerID":12,"TotalOrders":6,"AverageOrderSize":302.4666},
{"CustomerID":35,"TotalOrders":18,"AverageOrderSize":1311.7544},
{"CustomerID":86,"TotalOrders":10,"AverageOrderSize":1065.3850},
{"CustomerID":63,"TotalOrders":28,"AverageOrderSize":4195.8353},
{"CustomerID":6,"TotalOrders":7,"AverageOrderSize":462.8285},
{"CustomerID":55,"TotalOrders":10,"AverageOrderSize":1632.5150},
{"CustomerID":43,"TotalOrders":2,"AverageOrderSize":178.5000},
{"CustomerID":49,"TotalOrders":10,"AverageOrderSize":760.3850},
{"CustomerID":67,"TotalOrders":11,"AverageOrderSize":1174.9454},
{"CustomerID":21,"TotalOrders":7,"AverageOrderSize":634.1285},
{"CustomerID":27,"TotalOrders":6,"AverageOrderSize":257.6166},
{"CustomerID":58,"TotalOrders":6,"AverageOrderSize":707.0333},

{"CustomerID":81,"TotalOrders":6,"AverageOrderSize":1218.4366},
{"CustomerID":64,"TotalOrders":5,"AverageOrderSize":568.8200},
{"CustomerID":38,"TotalOrders":10,"AverageOrderSize":614.6300},
{"CustomerID":87,"TotalOrders":15,"AverageOrderSize":1107.8066},
{"CustomerID":7,"TotalOrders":11,"AverageOrderSize":1735.2727},
{"CustomerID":44,"TotalOrders":15,"AverageOrderSize":1418.8013},
{"CustomerID":1,"TotalOrders":6,"AverageOrderSize":766.0333},
{"CustomerID":50,"TotalOrders":7,"AverageOrderSize":1490.0828},
{"CustomerID":24,"TotalOrders":19,"AverageOrderSize":1713.4500},
{"CustomerID":47,"TotalOrders":12,"AverageOrderSize":1490.7958},
{"CustomerID":70,"TotalOrders":6,"AverageOrderSize":955.8583},
{"CustomerID":18,"TotalOrders":4,"AverageOrderSize":403.9750},
{"CustomerID":30,"TotalOrders":10,"AverageOrderSize":1183.0100},
{"CustomerID":84,"TotalOrders":10,"AverageOrderSize":993.7100},
{"CustomerID":10,"TotalOrders":14,"AverageOrderSize":1614.8357},
{"CustomerID":61,"TotalOrders":9,"AverageOrderSize":774.8477},
{"CustomerID":41,"TotalOrders":14,"AverageOrderSize":733.7392},
{"CustomerID":90,"TotalOrders":7,"AverageOrderSize":451.6214},
{"CustomerID":4,"TotalOrders":13,"AverageOrderSize":1062.0384},
{"CustomerID":65,"TotalOrders":18,"AverageOrderSize":2902.5500},
{"CustomerID":79,"TotalOrders":6,"AverageOrderSize":825.6666},
{"CustomerID":19,"TotalOrders":8,"AverageOrderSize":1879.2075},
{"CustomerID":73,"TotalOrders":7,"AverageOrderSize":2591.2071},
{"CustomerID":25,"TotalOrders":15,"AverageOrderSize":1914.8473},
{"CustomerID":36,"TotalOrders":5,"AverageOrderSize":612.6400},
{"CustomerID":85,"TotalOrders":5,"AverageOrderSize":296.0000},
{"CustomerID":62,"TotalOrders":13,"AverageOrderSize":2325.0846},
{"CustomerID":13,"TotalOrders":1,"AverageOrderSize":100.8000},
{"CustomerID":42,"TotalOrders":3,"AverageOrderSize":174.1666},
{"CustomerID":5,"TotalOrders":18,"AverageOrderSize":1498.2305},
{"CustomerID":56,"TotalOrders":10,"AverageOrderSize":1315.7500},
{"CustomerID":76,"TotalOrders":12,"AverageOrderSize":2058.7000},
{"CustomerID":59,"TotalOrders":10,"AverageOrderSize":2625.9950},
{"CustomerID":82,"TotalOrders":3,"AverageOrderSize":523.7333},
{"CustomerID":39,"TotalOrders":14,"AverageOrderSize":2267.5535},
{"CustomerID":33,"TotalOrders":2,"AverageOrderSize":744.3500},
{"CustomerID":16,"TotalOrders":3,"AverageOrderSize":573.0333},
{"CustomerID":88,"TotalOrders":9,"AverageOrderSize":720.0777},
{"CustomerID":53,"TotalOrders":3,"AverageOrderSize":216.3333},
{"CustomerID":45,"TotalOrders":4,"AverageOrderSize":872.5050},
{"CustomerID":2,"TotalOrders":4,"AverageOrderSize":350.7375},
{"CustomerID":48,"TotalOrders":8,"AverageOrderSize":532.3250},
{"CustomerID":71,"TotalOrders":31,"AverageOrderSize":3731.3996},
{"CustomerID":77,"TotalOrders":4,"AverageOrderSize":840.2500},
{"CustomerID":17,"TotalOrders":6,"AverageOrderSize":627.2016},
{"CustomerID":31,"TotalOrders":9,"AverageOrderSize":966.9144},
{"CustomerID":60,"TotalOrders":5,"AverageOrderSize":1063.4200},
{"CustomerID":83,"TotalOrders":11,"AverageOrderSize":1513.0727},
{"CustomerID":34,"TotalOrders":14,"AverageOrderSize":2435.7964},
{"CustomerID":40,"TotalOrders":4,"AverageOrderSize":498.0125},
{"CustomerID":11,"TotalOrders":10,"AverageOrderSize":608.9900},
{"CustomerID":54,"TotalOrders":5,"AverageOrderSize":692.0400},
{"CustomerID":91,"TotalOrders":7,"AverageOrderSize":504.5642},
{"CustomerID":20,"TotalOrders":30,"AverageOrderSize":3774.5560},
{"CustomerID":68,"TotalOrders":10,"AverageOrderSize":2003.3200},
{"CustomerID":80,"TotalOrders":10,"AverageOrderSize":1081.2150},
{"CustomerID":28,"TotalOrders":8,"AverageOrderSize":893.9437},
{"CustomerID":74,"TotalOrders":4,"AverageOrderSize":605.8375},
{"CustomerID":14,"TotalOrders":8,"AverageOrderSize":1610.7875},
{"CustomerID":37,"TotalOrders":19,"AverageOrderSize":3016.7047},
{"CustomerID":8,"TotalOrders":3,"AverageOrderSize":1765.9333},
{"CustomerID":51,"TotalOrders":13,"AverageOrderSize":2477.2230}]

**Corrected: Added MonthlySalesGrowth to add complexity and usability.**

**Complex Proposition 6:** Implement a detailed monthly sales growth analysis for each product category in Northwinds2022TSQLV7, utilizing a custom function to calculate growth rates and identify sales trends over time.

- **Custom Function Creation:** `dbo.CalculateMonthlySalesGrowth` is developed to compute the monthly sales growth percentage, accounting for cases where previous sales are zero to avoid division errors, thus ensuring accurate growth calculation.
- **CTE Usage:** `CategoryMonthlySales` CTE is formed to aggregate total sales per category per month, providing a foundation for calculating monthly sales growth.
- **Tables Engaged:** The analysis incorporates data from `Production.Category`, `Production.Product`, and `Sales.OrderDetail`, along with `Sales.[Order]`, to track sales figures and growth across different time periods and product categories.
- **Data Aggregation:** In the primary query, sales data is summed and grouped by category, month, and year to facilitate the growth calculation, with the `dbo.CalculateMonthlySalesGrowth` function applied to compare current and previous month sales figures.
- **Output Goal:** The final output will list Category ID, Category Name, Order Month, Order Year, Total Sales, and Monthly Sales Growth, sorted by category and date to showcase sales performance and growth trends, aiding in strategic business planning and category management.

In [21]:
```
USE Northwinds2022TSQLV7;
go
CREATE OR ALTER FUNCTION dbo.CalculateMonthlySalesGrowth (@currentSales MONEY, @previous
RETURNS FLOAT
AS
BEGIN
    DECLARE @growth FLOAT;
    IF @previousSales = 0
        SET @growth = NULL;
    ELSE
        SET @growth = (@currentSales - @previousSales) / @previousSales;
    RETURN @growth;
END;
GO
WITH CategoryMonthlySales AS (
    SELECT
        c.CategoryID,
        c.CategoryName,
        MONTH(o.OrderDate) AS OrderMonth,
        YEAR(o.OrderDate) AS OrderYear,
        SUM(od.UnitPrice * od.Quantity) AS TotalSales
    FROM Production.Category c
    JOIN Production.Product p ON c.CategoryID = p.CategoryID
    JOIN Sales.OrderDetail od ON p.ProductID = od.ProductID
    JOIN Sales.[Order] o ON od.OrderID = o.OrderID
    GROUP BY c.CategoryID, c.CategoryName, YEAR(o.OrderDate), MONTH(o.OrderDate)
)
SELECT
    c.CategoryID,
    c.CategoryName,
    MONTH(o1.OrderDate) AS OrderMonth,
    YEAR(o1.OrderDate) AS OrderYear,
    SUM(od1.UnitPrice * od1.Quantity) AS TotalSales,
    dbo.CalculateMonthlySalesGrowth(SUM(od1.UnitPrice * od1.Quantity), SUM(od2.UnitPrice
FROM Production.Category c
JOIN Production.Product p ON c.CategoryID = p.CategoryID
JOIN Sales.OrderDetail od1 ON p.ProductID = od1.ProductID
JOIN Sales.[Order] o1 ON od1.OrderID = o1.OrderID
JOIN Sales.OrderDetail od2 ON p.ProductID = od2.ProductID
JOIN Sales.[Order] o2 ON od2.OrderID = o2.OrderID AND MONTH(o2.OrderDate) = MONTH(o1.Ord
GROUP BY c.CategoryID, c.CategoryName, YEAR(o1.OrderDate), MONTH(o1.OrderDate)
ORDER BY c.CategoryID, YEAR(o1.OrderDate), MONTH(o1.OrderDate);
```

Commands completed successfully.

Commands completed successfully.

(157 rows affected)

Total execution time: 00:00:00.085

Out[21]:

| CategoryID | CategoryName | OrderMonth | OrderYear | TotalSales | MonthlySalesGrowth |
|---|---|---|---|---|---|
| 1 | Beverages | 8 | 2014 | 2260.80 | 0.602 |
| 1 | Beverages | 9 | 2014 | 5040.40 | -0.1321 |
| 1 | Beverages | 10 | 2014 | 5692.40 | 0.0857 |
| 1 | Beverages | 11 | 2014 | 18991.60 | 0.3452 |
| 1 | Beverages | 12 | 2014 | 27356.00 | 0.3842 |
| 1 | Beverages | 2 | 2015 | 1884.80 | -0.501 |
| 1 | Beverages | 3 | 2015 | 5972.40 | 0.1563 |
| 1 | Beverages | 4 | 2015 | 9121.00 | -0.277 |
| 1 | Beverages | 5 | 2015 | 15424.00 | 0.316 |
| 1 | Beverages | 6 | 2015 | 4142.00 | -0.3497 |
| 1 | Beverages | 7 | 2015 | 11379.50 | 0.5372 |
| 1 | Beverages | 8 | 2015 | 9091.50 | 0.4061 |
| 1 | Beverages | 9 | 2015 | 3336.50 | -0.0997 |
| 1 | Beverages | 10 | 2015 | 6235.00 | 0.6678 |
| 1 | Beverages | 11 | 2015 | 10442.00 | 0.1717 |
| 1 | Beverages | 12 | 2015 | 3456.75 | -0.0303 |
| 1 | Beverages | 2 | 2016 | 135231.50 | 1.2534 |
| 1 | Beverages | 3 | 2016 | 57414.50 | -0.2444 |
| 1 | Beverages | 4 | 2016 | 58082.50 | 0.0654 |
| 1 | Beverages | 5 | 2016 | 16571.00 | -0.3606 |
| 2 | Condiments | 8 | 2014 | 888.00 | -0.4414 |
| 2 | Condiments | 10 | 2014 | 1707.20 | -0.2096 |
| 2 | Condiments | 11 | 2014 | 2352.00 | -0.1975 |
| 2 | Condiments | 12 | 2014 | 1299.20 | 1.0661 |
| 2 | Condiments | 2 | 2015 | 6374.40 | 1.5604 |
| 2 | Condiments | 3 | 2015 | 1464.30 | -0.7106 |
| 2 | Condiments | 4 | 2015 | 2210.10 | 0.0372 |
| 2 | Condiments | 5 | 2015 | 7629.40 | 0.1564 |
| 2 | Condiments | 6 | 2015 | 1468.45 | -0.3991 |
| 2 | Condiments | 7 | 2015 | 2798.70 | 0.5975 |
| 2 | Condiments | 8 | 2015 | 3147.05 | -0.4437 |
| 2 | Condiments | 9 | 2015 | 3169.90 | 0.0781 |
| 2 | Condiments | 10 | 2015 | 7505.35 | 0.9449 |
| 2 | Condiments | 11 | 2015 | 3986.00 | -0.3426 |
| 2 | Condiments | 12 | 2015 | 5928.70 | 0.6435 |
| 2 | Condiments | 2 | 2016 | 9262.20 | 0.405 |
| 2 | Condiments | 3 | 2016 | 8135.00 | -0.3232 |

In [19]:
```
USE Northwinds2022TSQLV7;
go
CREATE OR ALTER FUNCTION dbo.CalculateMonthlySalesGrowth (@currentSales MONEY, @previous
RETURNS FLOAT
AS
BEGIN
    DECLARE @growth FLOAT;
    IF @previousSales = 0
        SET @growth = NULL;
    ELSE
        SET @growth = (@currentSales - @previousSales) / @previousSales;
    RETURN @growth;
END;
GO
WITH CategoryMonthlySales AS (
    SELECT
        c.CategoryID,
        c.CategoryName,
        MONTH(o.OrderDate) AS OrderMonth,
        YEAR(o.OrderDate) AS OrderYear,
        SUM(od.UnitPrice * od.Quantity) AS TotalSales
    FROM Production.Category c
    JOIN Production.Product p ON c.CategoryID = p.CategoryID
    JOIN Sales.OrderDetail od ON p.ProductID = od.ProductID
    JOIN Sales.[Order] o ON od.OrderID = o.OrderID
    GROUP BY c.CategoryID, c.CategoryName, YEAR(o.OrderDate), MONTH(o.OrderDate)
)
SELECT
    c.CategoryID,
    c.CategoryName,
    MONTH(o1.OrderDate) AS OrderMonth,
    YEAR(o1.OrderDate) AS OrderYear,
    SUM(od1.UnitPrice * od1.Quantity) AS TotalSales,
    dbo.CalculateMonthlySalesGrowth(SUM(od1.UnitPrice * od1.Quantity), SUM(od2.UnitPrice
FROM Production.Category c
JOIN Production.Product p ON c.CategoryID = p.CategoryID
JOIN Sales.OrderDetail od1 ON p.ProductID = od1.ProductID
JOIN Sales.[Order] o1 ON od1.OrderID = o1.OrderID
JOIN Sales.OrderDetail od2 ON p.ProductID = od2.ProductID
JOIN Sales.[Order] o2 ON od2.OrderID = o2.OrderID AND MONTH(o2.OrderDate) = MONTH(o1.Ord
GROUP BY c.CategoryID, c.CategoryName, YEAR(o1.OrderDate), MONTH(o1.OrderDate)
ORDER BY c.CategoryID, YEAR(o1.OrderDate), MONTH(o1.OrderDate)
FOR JSON PATH;
```

Commands completed successfully.

Commands completed successfully.

(157 rows affected)

Total execution time: 00:00:00.057

Out[19]:

**JSON_F52E2B61-18**


```
[{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":8,"OrderYear":2014,"TotalSales":2260.8000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":9,"OrderYear":2014,"TotalSales":5040.4000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":10,"OrderYear":2014,"TotalSales":5692.4000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":11,"OrderYear":2014,"TotalSales":18991.6000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":12,"OrderYear":2014,"TotalSales":27356.0000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":2,"OrderYear":2015,"TotalSales":1884.8000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":3,"OrderYear":2015,"TotalSales":5972.4000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":4,"OrderYear":2015,"TotalSales":9121.0000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":5,"OrderYear":2015,"TotalSales":15424.0000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":6,"OrderYear":2015,"TotalSales":4142.0000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":7,"OrderYear":2015,"TotalSales":11379.5000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":8,"OrderYear":2015,"TotalSales":9091.5000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":9,"OrderYear":2015,"TotalSales":3336.5000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":10,"OrderYear":2015,"TotalSales":6235.0000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":11,"OrderYear":2015,"TotalSales":10442.0000,"MonthlySalesGrow
{"CategoryID":1,"CategoryName":"Beverages","OrderMonth":12,"OrderYear":2015,"TotalSales":3456.7500,"MonthlySalesGrow
```

{"CategoryID":8,"CategoryName":"Seafood","OrderMonth":3,"OrderYear":2016,"TotalSales":20001.8600,"MonthlySalesGrow
{"CategoryID":8,"CategoryName":"Seafood","OrderMonth":4,"OrderYear":2016,"TotalSales":34401.0000,"MonthlySalesGrov
{"CategoryID":8,"CategoryName":"Seafood","OrderMonth":5,"OrderYear":2016,"TotalSales":4212.9600,"MonthlySalesGrow

## Medium Proposition 7: Conduct an analysis in Northwinds2022TSQLV7 to calculate the average purchase amount for each customer, offering insights into customer spending behavior and aiding in the identification of key customer segments.

- **CTE Usage:** The `CustomerOrderTotal` CTE aggregates the total purchase amount for each customer by summing the product of unit price and quantity for all orders in `Sales.OrderDetail`, grouped by `CustomerID`.
- **Tables Engaged:** The query leverages data from `Sales.Customer` for customer details and `Sales.OrderDetail` for transaction information, ensuring a comprehensive evaluation of purchasing patterns.
- **Data Aggregation:** By joining the `CustomerOrderTotal` CTE with `Sales.Customer`, the main query calculates the average purchase amount per customer, providing a metric that reflects the typical spending per customer within the database.
- **Output Goal:** The final report displays the CustomerID, CustomerCompanyName, and AveragePurchaseAmount, giving a clear picture of how much each customer spends on average, which can be used to tailor marketing strategies and improve customer relationship management.

In [7]:
```
USE Northwinds2022TSQLV7
go

WITH CustomerOrderTotal AS (
    SELECT
        o.CustomerID,
        SUM(od.UnitPrice * od.Quantity) AS TotalAmount
    FROM Sales.[Order] o
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY o.CustomerID
)
SELECT
    c.CustomerID,
    c.CustomerCompanyName,
    AVG(cot.TotalAmount) AS AveragePurchaseAmount
FROM Sales.Customer c
JOIN CustomerOrderTotal cot ON c.CustomerID = cot.CustomerID
GROUP BY c.CustomerID, c.CustomerCompanyName;
```

Commands completed successfully.

(89 rows affected)

Total execution time: 00:00:00.023

Out[7]:

| CustomerID | CustomerCompanyName | AveragePurchaseAmount |
|---|---|---|
| 23 | Customer WVFAF | 11666.90 |
| 46 | Customer XPNIK | 17825.06 |
| 69 | Customer SIUIH | 1467.29 |
| 29 | Customer MDLWA | 836.70 |
| 75 | Customer XOJYP | 12489.70 |
| 15 | Customer JUWXK | 3810.75 |
| 9 | Customer RTXGC | 23850.95 |
| 89 | Customer YBQTI | 29073.45 |

In [20]:
```
USE Northwinds2022TSQLV7
go

WITH CustomerOrderTotal AS (
    SELECT
        o.CustomerID,
        SUM(od.UnitPrice * od.Quantity) AS TotalAmount
    FROM Sales.[Order] o
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY o.CustomerID
)
SELECT
    c.CustomerID,
    c.CustomerCompanyName,
    AVG(cot.TotalAmount) AS AveragePurchaseAmount
FROM Sales.Customer c
JOIN CustomerOrderTotal cot ON c.CustomerID = cot.CustomerID
GROUP BY c.CustomerID, c.CustomerCompanyName
FOR JSON PATH;
```

Commands completed successfully.

(89 rows affected)

Total execution time: 00:00:00.024

Out[20]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**


```
[{"CustomerID":23,"CustomerCompanyName":"Customer WVFAF","AveragePurchaseAmount":11666.9000},
{"CustomerID":46,"CustomerCompanyName":"Customer XPNIK","AveragePurchaseAmount":17825.0600},
{"CustomerID":69,"CustomerCompanyName":"Customer SIUIH","AveragePurchaseAmount":1467.2900},
{"CustomerID":29,"CustomerCompanyName":"Customer MDLWA","AveragePurchaseAmount":836.7000},
{"CustomerID":75,"CustomerCompanyName":"Customer XOJYP","AveragePurchaseAmount":12489.7000},
{"CustomerID":15,"CustomerCompanyName":"Customer JUWXK","AveragePurchaseAmount":3810.7500},
{"CustomerID":9,"CustomerCompanyName":"Customer RTXGC","AveragePurchaseAmount":23850.9500},
{"CustomerID":89,"CustomerCompanyName":"Customer YBQTI","AveragePurchaseAmount":29073.4500},
{"CustomerID":3,"CustomerCompanyName":"Customer KBUDE","AveragePurchaseAmount":7515.3500},
{"CustomerID":52,"CustomerCompanyName":"Customer PZNLA","AveragePurchaseAmount":5042.2000},
{"CustomerID":72,"CustomerCompanyName":"Customer AHPOP","AveragePurchaseAmount":17172.0500},
{"CustomerID":66,"CustomerCompanyName":"Customer LHANT","AveragePurchaseAmount":7555.6000},
{"CustomerID":78,"CustomerCompanyName":"Customer NLTYP","AveragePurchaseAmount":1947.2400},
{"CustomerID":32,"CustomerCompanyName":"Customer YSIQX","AveragePurchaseAmount":19711.1300},
{"CustomerID":26,"CustomerCompanyName":"Customer USDBG","AveragePurchaseAmount":3172.1600},
{"CustomerID":12,"CustomerCompanyName":"Customer PSNMQ","AveragePurchaseAmount":1814.8000},
{"CustomerID":35,"CustomerCompanyName":"Customer UMTLM","AveragePurchaseAmount":23611.5800},
{"CustomerID":86,"CustomerCompanyName":"Customer SNXOJ","AveragePurchaseAmount":10653.8500},
{"CustomerID":63,"CustomerCompanyName":"Customer IRRVL","AveragePurchaseAmount":117483.3900},
{"CustomerID":6,"CustomerCompanyName":"Customer XHXJV","AveragePurchaseAmount":3239.8000},
{"CustomerID":55,"CustomerCompanyName":"Customer KZQZT","AveragePurchaseAmount":16325.1500},
{"CustomerID":43,"CustomerCompanyName":"Customer UISOJ","AveragePurchaseAmount":357.0000},
{"CustomerID":49,"CustomerCompanyName":"Customer CQRAA","AveragePurchaseAmount":7603.8500},
{"CustomerID":67,"CustomerCompanyName":"Customer QVEPD","AveragePurchaseAmount":12924.4000},
{"CustomerID":21,"CustomerCompanyName":"Customer KIDPX","AveragePurchaseAmount":4438.9000},
{"CustomerID":27,"CustomerCompanyName":"Customer WMFEA","AveragePurchaseAmount":1545.7000},
{"CustomerID":58,"CustomerCompanyName":"Customer AHXHT","AveragePurchaseAmount":4242.2000},
{"CustomerID":81,"CustomerCompanyName":"Customer YQQWW","AveragePurchaseAmount":7310.6200},
{"CustomerID":64,"CustomerCompanyName":"Customer LWGMD","AveragePurchaseAmount":2844.1000},
{"CustomerID":87,"CustomerCompanyName":"Customer ZHYOS","AveragePurchaseAmount":16617.1000},
{"CustomerID":38,"CustomerCompanyName":"Customer LJUCA","AveragePurchaseAmount":6146.3000},
{"CustomerID":7,"CustomerCompanyName":"Customer QXVLA","AveragePurchaseAmount":19088.0000},
{"CustomerID":44,"CustomerCompanyName":"Customer OXFRU","AveragePurchaseAmount":21282.0200},
{"CustomerID":50,"CustomerCompanyName":"Customer JYPSC","AveragePurchaseAmount":10430.5800},
{"CustomerID":1,"CustomerCompanyName":"Customer NRZBB","AveragePurchaseAmount":4596.2000},
{"CustomerID":24,"CustomerCompanyName":"Customer CYZTN","AveragePurchaseAmount":32555.5500},
{"CustomerID":47,"CustomerCompanyName":"Customer PSQUZ","AveragePurchaseAmount":17889.5500},
{"CustomerID":70,"CustomerCompanyName":"Customer TMXGN","AveragePurchaseAmount":5735.1500},
{"CustomerID":30,"CustomerCompanyName":"Customer KSLQF","AveragePurchaseAmount":11830.1000},
{"CustomerID":18,"CustomerCompanyName":"Customer BSVAR","AveragePurchaseAmount":1615.9000},
{"CustomerID":84,"CustomerCompanyName":"Customer NRCSK","AveragePurchaseAmount":9937.1000},
{"CustomerID":10,"CustomerCompanyName":"Customer EEALV","AveragePurchaseAmount":22607.7000},
{"CustomerID":61,"CustomerCompanyName":"Customer WULWD","AveragePurchaseAmount":6973.6300},
{"CustomerID":41,"CustomerCompanyName":"Customer XIIWM","AveragePurchaseAmount":10272.3500},
```

{"CustomerID":90,"CustomerCompanyName":"Customer XBBVR","AveragePurchaseAmount":3161.3500},
{"CustomerID":4,"CustomerCompanyName":"Customer HFBZG","AveragePurchaseAmount":13806.5000},
{"CustomerID":65,"CustomerCompanyName":"Customer NYUHS","AveragePurchaseAmount":52245.9000},
{"CustomerID":79,"CustomerCompanyName":"Customer FAPSM","AveragePurchaseAmount":4954.0000},
{"CustomerID":19,"CustomerCompanyName":"Customer RFNQC","AveragePurchaseAmount":15033.6600},
{"CustomerID":73,"CustomerCompanyName":"Customer JMIKW","AveragePurchaseAmount":18138.4500},
{"CustomerID":25,"CustomerCompanyName":"Customer AZJED","AveragePurchaseAmount":28722.7100},
{"CustomerID":36,"CustomerCompanyName":"Customer LVJSO","AveragePurchaseAmount":3063.2000},
{"CustomerID":85,"CustomerCompanyName":"Customer ENQZT","AveragePurchaseAmount":1480.0000},
{"CustomerID":13,"CustomerCompanyName":"Customer VMLOG","AveragePurchaseAmount":100.8000},
{"CustomerID":62,"CustomerCompanyName":"Customer WFIZJ","AveragePurchaseAmount":30226.1000},
{"CustomerID":42,"CustomerCompanyName":"Customer IAIJK","AveragePurchaseAmount":522.5000},
{"CustomerID":5,"CustomerCompanyName":"Customer HGVLZ","AveragePurchaseAmount":26968.1500},
{"CustomerID":56,"CustomerCompanyName":"Customer QNIVZ","AveragePurchaseAmount":13157.5000},
{"CustomerID":76,"CustomerCompanyName":"Customer SFOGW","AveragePurchaseAmount":24704.4000},
{"CustomerID":59,"CustomerCompanyName":"Customer LOLJO","AveragePurchaseAmount":26259.9500},
{"CustomerID":82,"CustomerCompanyName":"Customer EYHKM","AveragePurchaseAmount":1571.2000},
{"CustomerID":33,"CustomerCompanyName":"Customer FVXPQ","AveragePurchaseAmount":1488.7000},
{"CustomerID":39,"CustomerCompanyName":"Customer GLLAG","AveragePurchaseAmount":31745.7500},
{"CustomerID":16,"CustomerCompanyName":"Customer GYBBY","AveragePurchaseAmount":1719.1000},
{"CustomerID":88,"CustomerCompanyName":"Customer SRQVM","AveragePurchaseAmount":6480.7000},
{"CustomerID":53,"CustomerCompanyName":"Customer GCJSG","AveragePurchaseAmount":649.0000},
{"CustomerID":45,"CustomerCompanyName":"Customer QXPPT","AveragePurchaseAmount":3490.0200},
{"CustomerID":2,"CustomerCompanyName":"Customer MLTDN","AveragePurchaseAmount":1402.9500},
{"CustomerID":48,"CustomerCompanyName":"Customer DVFMB","AveragePurchaseAmount":4258.6000},
{"CustomerID":71,"CustomerCompanyName":"Customer LCOUJ","AveragePurchaseAmount":115673.3900},
{"CustomerID":77,"CustomerCompanyName":"Customer LCYBZ","AveragePurchaseAmount":3361.0000},
{"CustomerID":17,"CustomerCompanyName":"Customer FEVNN","AveragePurchaseAmount":3763.2100},
{"CustomerID":31,"CustomerCompanyName":"Customer YJCBX","AveragePurchaseAmount":8702.2300},
{"CustomerID":60,"CustomerCompanyName":"Customer QZURI","AveragePurchaseAmount":5317.1000},
{"CustomerID":83,"CustomerCompanyName":"Customer ZRNDE","AveragePurchaseAmount":16643.8000},
{"CustomerID":34,"CustomerCompanyName":"Customer IBVRG","AveragePurchaseAmount":34101.1500},
{"CustomerID":40,"CustomerCompanyName":"Customer EFFTC","AveragePurchaseAmount":1992.0500},
{"CustomerID":11,"CustomerCompanyName":"Customer UBHAU","AveragePurchaseAmount":6089.9000},
{"CustomerID":54,"CustomerCompanyName":"Customer TDKEG","AveragePurchaseAmount":3460.2000},
{"CustomerID":91,"CustomerCompanyName":"Customer CCFIZ","AveragePurchaseAmount":3531.9500},
{"CustomerID":20,"CustomerCompanyName":"Customer THHDP","AveragePurchaseAmount":113236.6800},
{"CustomerID":68,"CustomerCompanyName":"Customer CCKOT","AveragePurchaseAmount":20033.2000},
{"CustomerID":80,"CustomerCompanyName":"Customer VONTK","AveragePurchaseAmount":10812.1500},
{"CustomerID":28,"CustomerCompanyName":"Customer XYUFB","AveragePurchaseAmount":7151.5500},
{"CustomerID":74,"CustomerCompanyName":"Customer YSHXL","AveragePurchaseAmount":2423.3500},
{"CustomerID":14,"CustomerCompanyName":"Customer WNMAF","AveragePurchaseAmount":12886.3000},
{"CustomerID":37,"CustomerCompanyName":"Customer FRXZL","AveragePurchaseAmount":57317.3900},
{"CustomerID":8,"CustomerCompanyName":"Customer QUHWH","AveragePurchaseAmount":5297.8000},
{"CustomerID":51,"CustomerCompanyName":"Customer PVDZC","AveragePurchaseAmount":32203.9000}]

## Corrected: Added Sales Quarters to add in complexity and usability. 

## Medium Proposition 8: Perform a quarterly sales analysis in the Northwinds2022TSQLV7 database to track sales trends over time, providing a structured view of revenue changes across different quarters of each year.

- **CTE Usage:** `QuarterlySales` aggregates sales data by year and quarter, summing up the total sales for each period by multiplying unit price and quantity from the `Sales.OrderDetail` table, ensuring a segmented view of sales performance over time.
- **Tables Engaged:** The analysis utilizes `Sales.[Order]` for order dates and `Sales.OrderDetail` for detailed transaction data, enabling a comprehensive evaluation of sales across different time frames.
- **Data Aggregation:** The data is grouped by year and quarter within the `QuarterlySales` CTE, facilitating an organized summary of total sales that captures seasonal and quarterly sales trends.
- **Output Goal:** The final output, sorted by year and quarter, presents SaleYear, SaleQuarter, and TotalSales, offering a clear, chronological view of sales performance, which is crucial for assessing

seasonal impacts, planning inventory, and making strategic business decisions.

In [9]:
```
USE Northwinds2022TSQLV7
GO


WITH QuarterlySales AS (
    SELECT
        YEAR(o.OrderDate) AS SaleYear,
        DATEPART(QUARTER, o.OrderDate) AS SaleQuarter,
        SUM(od.UnitPrice * od.Quantity) AS TotalSales
    FROM Sales.[Order] o
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY YEAR(o.OrderDate), DATEPART(QUARTER, o.OrderDate)
)
SELECT
    SaleYear,
    SaleQuarter,
    TotalSales
FROM QuarterlySales
ORDER BY SaleYear, SaleQuarter;
```

Commands completed successfully.

(8 rows affected)

Total execution time: 00:00:00.024

Out[9]:

| SaleYear | SaleQuarter | TotalSales |
|---|---|---|
| 2014 | 3 | 84437.50 |
| 2014 | 4 | 141861.00 |
| 2015 | 1 | 147879.90 |
| 2015 | 2 | 151611.09 |
| 2015 | 3 | 165179.64 |
| 2015 | 4 | 193718.12 |
| 2016 | 1 | 315242.12 |
| 2016 | 2 | 154529.22 |

In [21]:
```
USE Northwinds2022TSQLV7
GO


WITH QuarterlySales AS (
    SELECT
        YEAR(o.OrderDate) AS SaleYear,
        DATEPART(QUARTER, o.OrderDate) AS SaleQuarter,
        SUM(od.UnitPrice * od.Quantity) AS TotalSales
    FROM Sales.[Order] o
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY YEAR(o.OrderDate), DATEPART(QUARTER, o.OrderDate)
)
SELECT
    SaleYear,
    SaleQuarter,
    TotalSales
FROM QuarterlySales
ORDER BY SaleYear, SaleQuarter
FOR JSON PATH;
```

Commands completed successfully.

(8 rows affected)

Total execution time: 00:00:00.019

Out[21]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**


```
[{"SaleYear":2014,"SaleQuarter":3,"TotalSales":84437.5000},{"SaleYear":2014,"SaleQuarter":4,"TotalSales":141861.0000},
{"SaleYear":2015,"SaleQuarter":1,"TotalSales":147879.9000},{"SaleYear":2015,"SaleQuarter":2,"TotalSales":151611.0900},
{"SaleYear":2015,"SaleQuarter":3,"TotalSales":165179.6400},{"SaleYear":2015,"SaleQuarter":4,"TotalSales":193718.1200},
{"SaleYear":2016,"SaleQuarter":1,"TotalSales":315242.1200},{"SaleYear":2016,"SaleQuarter":2,"TotalSales":154529.2200}]
```


## Medium Proposition 9: Execute a comprehensive sales performance review for employees in Northwinds2022TSQLV7, determining total sales generated by each employee to highlight top performers and assess overall sales productivity.

- **CTE Usage:** The `EmployeeSales` CTE calculates the total sales for each employee by summing the product of quantity and unit price from `Sales.OrderDetail`, grouped by employee details.
- **Tables Engaged:** This analysis integrates `HumanResources.Employee` for employee information, `Sales.[Order]` for connecting employees to sales transactions, and `Sales.OrderDetail` for the financial details of each sale.
- **Data Aggregation:** In the CTE, sales data is aggregated by employee, allowing for a clear representation of each individual's contribution to the company's sales efforts.
- **Output Goal:** The final output presents the EmployeeID, EmployeeName, and TotalSales, sorted in descending order of TotalSales to easily identify the highest-grossing employees, providing valuable insights for performance evaluation and incentive planning.

In [8]:

```
use Northwinds2022TSQLV7
go

WITH EmployeeSales AS (
    SELECT
        e.EmployeeID,
        e.EmployeeFirstName + ' ' + e.EmployeeLastName AS EmployeeName,
        SUM(od.Quantity * od.UnitPrice) AS TotalSales
    FROM HumanResources.Employee e
    JOIN Sales.[Order] o ON e.EmployeeID = o.EmployeeID
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY e.EmployeeID, e.EmployeeFirstName, e.EmployeeLastName
)
SELECT
    EmployeeID,
    EmployeeName,
    TotalSales
FROM EmployeeSales
ORDER BY TotalSales DESC;
```

Commands completed successfully.

(9 rows affected)

Total execution time: 00:00:00.022

Out[8]:

| EmployeeID | EmployeeName | TotalSales |
|---|---|---|
| 4 | Yael Peled | 250187.45 |
| 3 | Judy Lew | 213051.30 |
| 1 | Sara Davis | 202143.71 |
| 2 | Don Funk | 177749.26 |
| 7 | Russell King | 141295.99 |

| | | |
|---|---|---|
| 8 | Maria Cameron | 133301.03 |
| 9 | Patricia Doyle | 82964.00 |
| 6 | Paul Suurs | 78198.10 |
| 5 | Sven Mortensen | 75567.75 |

In [22]:
```
use Northwinds2022TSQLV7
go

WITH EmployeeSales AS (
    SELECT
        e.EmployeeID,
        e.EmployeeFirstName + ' ' + e.EmployeeLastName AS EmployeeName,
        SUM(od.Quantity * od.UnitPrice) AS TotalSales
    FROM HumanResources.Employee e
    JOIN Sales.[Order] o ON e.EmployeeID = o.EmployeeID
    JOIN Sales.OrderDetail od ON o.OrderID = od.OrderID
    GROUP BY e.EmployeeID, e.EmployeeFirstName, e.EmployeeLastName
)
SELECT
    EmployeeID,
    EmployeeName,
    TotalSales
FROM EmployeeSales
ORDER BY TotalSales DESC
FOR JSON PATH;
```

Commands completed successfully.

(9 rows affected)

Total execution time: 00:00:00.038

Out[22]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**

[{"EmployeeID":4,"EmployeeName":"Yael Peled","TotalSales":250187.4500},{"EmployeeID":3,"EmployeeName":"Judy Lew","TotalSales":213051.3000},{"EmployeeID":1,"EmployeeName":"Sara Davis","TotalSales":202143.7100},{"EmployeeID":2,"EmployeeName":"Don Funk","TotalSales":177749.2600},{"EmployeeID":7,"EmployeeName":"Russell King","TotalSales":141295.9900},{"EmployeeID":8,"EmployeeName":"Maria Cameron","TotalSales":133301.0300},{"EmployeeID":9,"EmployeeName":"Patricia Doyle","TotalSales":82964.0000},{"EmployeeID":6,"EmployeeName":"Paul Suurs","TotalSales":78198.1000},{"EmployeeID":5,"EmployeeName":"Sven Mortensen","TotalSales":75567.7500}]

## Medium Proposition 10: Analyze manufacturing costs and pricing in AdventureWorks2017 by category to inform pricing strategies and manufacturing efficiency.

- **CTE Usage:** `CategoryCosts` calculates average manufacturing costs and list prices, along with product counts, per category from `Production.Product`, `ProductSubcategory`, and `ProductCategory`.
- **Tables Engaged:** The analysis merges product cost and price data with category information to assess financial metrics across product categories.
- **Data Aggregation:** Aggregates financial data and product counts at the category level within the CTE to provide insights into cost and pricing trends.
- **Output Goal:** Outputs CategoryName, formatted average manufacturing costs, list prices, and product counts, sorted by category, aiding in financial and strategic product category analysis.

In [26]:
```
USE AdventureWorks2017

;WITH CategoryCosts AS (
    SELECT
        pc.Name AS CategoryName,
```

```
        AVG(p.StandardCost) AS AvgManufacturingCost,
        AVG(p.ListPrice) AS AvgListPrice,
        COUNT(p.ProductID) AS ProductCount
    FROM Production.Product p
    JOIN Production.ProductSubcategory psc ON p.ProductSubcategoryID = psc.ProductSubcat
    JOIN Production.ProductCategory pc ON psc.ProductCategoryID = pc.ProductCategoryID
    GROUP BY pc.Name
)
SELECT
    CategoryName,
    FORMAT(AvgManufacturingCost, 'C', 'en-us') AS AvgManufacturingCostFormatted,
    FORMAT(AvgListPrice, 'C', 'en-us') AS AvgListPriceFormatted,
    ProductCount
FROM CategoryCosts
ORDER BY CategoryName;
```

(4 rows affected)

Total execution time: 00:00:00.125

Out[26]:

| CategoryName | AvgManufacturingCostFormatted | AvgListPriceFormatted | ProductCount |
|---|---|---|---|
| Accessories | $13.23 | $34.35 | 29 |
| Bikes | $949.41 | $1,586.74 | 97 |
| Clothing | $24.80 | $50.99 | 35 |
| Components | $268.14 | $469.86 | 134 |

In [23]:

```
USE AdventureWorks2017

;WITH CategoryCosts AS (
    SELECT
        pc.Name AS CategoryName,
        AVG(p.StandardCost) AS AvgManufacturingCost,
        AVG(p.ListPrice) AS AvgListPrice,
        COUNT(p.ProductID) AS ProductCount
    FROM Production.Product p
    JOIN Production.ProductSubcategory psc ON p.ProductSubcategoryID = psc.ProductSubcat
    JOIN Production.ProductCategory pc ON psc.ProductCategoryID = pc.ProductCategoryID
    GROUP BY pc.Name
)
SELECT
    CategoryName,
    FORMAT(AvgManufacturingCost, 'C', 'en-us') AS AvgManufacturingCostFormatted,
    FORMAT(AvgListPrice, 'C', 'en-us') AS AvgListPriceFormatted,
    ProductCount
FROM CategoryCosts
ORDER BY CategoryName
FOR JSON PATH;
```

(4 rows affected)

Total execution time: 00:00:00.051

Out[23]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**

[{"CategoryName":"Accessories","AvgManufacturingCostFormatted":"13.23 ”,” AvgListPriceFormatted ”:” 34.35","ProductCount":29},{"CategoryName":"Bikes","AvgManufacturingCostFormatted":" 949.41 ”,” AvgListPriceFormatted ”:”1,586.74","ProductCount":97}, {"CategoryName":"Clothing","AvgManufacturingCostFormatted":"24.80 ”,” AvgListPriceFormatted ”:” 50.99","ProductCount":35},{"CategoryName":"Components","AvgManufacturingCostFormatted":" 268.14 ”,” AvgListPriceFormatted ”:”469.86","ProductCount":134}]

## Medium Proposition 11: Analyze 2011 product sales in AdventureWorks2017, focusing on quantity and sales amounts to

## pinpoint top products and trends.

- **CTE Usage:** `SalesData` compiles sales quantities and amounts for each product in 2011, joining `Production.Product` with `Sales.SalesOrderDetail` and filtering by `Sales.SalesOrderHeader` date.
- **Data Aggregation:** Groups by product and year, summing quantities and sales in the CTE for detailed yearly performance metrics.
- **Output Goal:** Produces a list of products with total sales and quantities for 2011, ordered by sales amount to highlight leading products and sales patterns.

In [27]:
```
USE AdventureWorks2017;

-- Medium complexity query with 2 tables joined, built-in SQL functions, and group by su
WITH SalesData AS (
    SELECT
        p.ProductID,
        p.Name AS ProductName,
        YEAR(soh.OrderDate) AS OrderYear,
        SUM(sod.OrderQty) AS TotalQuantity,
        SUM(sod.LineTotal) AS TotalSalesAmount
    FROM
        Production.Product p
    JOIN
        Sales.SalesOrderDetail sod ON p.ProductID = sod.ProductID
    JOIN
        Sales.SalesOrderHeader soh ON sod.SalesOrderID = soh.SalesOrderID
    WHERE
        soh.OrderDate >= '2011-01-01' AND soh.OrderDate < '2012-01-01'  -- Adjust the da
    GROUP BY
        p.ProductID,
        p.Name,
        YEAR(soh.OrderDate)
)

SELECT
    ProductID,
    ProductName,
    OrderYear,
    SUM(TotalQuantity) AS TotalQuantity,
    SUM(TotalSalesAmount) AS TotalSalesAmount
FROM
    SalesData
GROUP BY
    ProductID,
    ProductName,
    OrderYear
ORDER BY
    OrderYear DESC,
    TotalSalesAmount DESC;
```

(60 rows affected)

Total execution time: 00:00:00.161

Out[27]:

| ProductID | ProductName | OrderYear | TotalQuantity | TotalSalesAmount |
|---|---|---|---|---|
| 753 | Road-150 Red, 56 | 2011 | 363 | 1018375.642000 |
| 749 | Road-150 Red, 62 | 2011 | 320 | 968995.516000 |
| 777 | Mountain-100 Black, 44 | 2011 | 373 | 800119.679268 |
| 771 | Mountain-100 Silver, 38 | 2011 | 351 | 769077.738000 |
| 751 | Road-150 Red, 48 | 2011 | 237 | 765034.126000 |

| | | | | |
|---|---|---|---|---|
| 739 | HL Mountain Frame - Silver, 42 | 2011 | 13 | 9393.733700 |
| 718 | HL Road Frame - Red, 44 | 2011 | 11 | 8338.834900 |
| 716 | Long-Sleeve Logo Jersey, XL | 2011 | 255 | 7354.302000 |
| 744 | HL Mountain Frame - Black, 44 | 2011 | 9 | 7287.840000 |
| 711 | Sport-100 Helmet, Blue | 2011 | 360 | 7255.445108 |
| 708 | Sport-100 Helmet, Black | 2011 | 341 | 6883.596500 |
| 707 | Sport-100 Helmet, Red | 2011 | 331 | 6681.731500 |
| 714 | Long-Sleeve Logo Jersey, M | 2011 | 228 | 6575.611200 |
| 709 | Mountain Bike Socks, M | 2011 | 608 | 3314.189950 |
| 712 | AWC Logo Cap | 2011 | 545 | 2816.535136 |
| 736 | LL Road Frame - Black, 44 | 2011 | 13 | 2321.550400 |
| 727 | LL Road Frame - Red, 52 | 2011 | 5 | 919.691000 |
| 710 | Mountain Bike Socks, L | 2011 | 66 | 376.200000 |
| 723 | LL Road Frame - Black, 60 | 2011 | 1 | 178.580800 |

In [24]:

```
USE AdventureWorks2017;

-- Medium complexity query with 2 tables joined, built-in SQL functions, and group by su
WITH SalesData AS (
    SELECT
        p.ProductID,
        p.Name AS ProductName,
        YEAR(soh.OrderDate) AS OrderYear,
        SUM(sod.OrderQty) AS TotalQuantity,
        SUM(sod.LineTotal) AS TotalSalesAmount
    FROM
        Production.Product p
    JOIN
        Sales.SalesOrderDetail sod ON p.ProductID = sod.ProductID
    JOIN
        Sales.SalesOrderHeader soh ON sod.SalesOrderID = soh.SalesOrderID
    WHERE
        soh.OrderDate >= '2011-01-01' AND soh.OrderDate < '2012-01-01'  -- Adjust the da
    GROUP BY
        p.ProductID,
        p.Name,
        YEAR(soh.OrderDate)
)

SELECT
    ProductID,
    ProductName,
    OrderYear,
    SUM(TotalQuantity) AS TotalQuantity,
    SUM(TotalSalesAmount) AS TotalSalesAmount
FROM
    SalesData
GROUP BY
    ProductID,
    ProductName,
    OrderYear
ORDER BY
    OrderYear DESC,
    TotalSalesAmount DESC
    FOR JSON PATH;
```

(60 rows affected)

Total execution time: 00:00:00.047

Out[24]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**


[{"ProductID":753,"ProductName":"Road-150 Red, 56","OrderYear":2011,"TotalQuantity":363,"TotalSalesAmount":1018375.642000},{"ProductID":749,"ProductName":"Road-150 Red, 62","OrderYear":2011,"TotalQuantity":320,"TotalSalesAmount":968995.516000},{"ProductID":777,"ProductName":"Mountain-100 Black, 44","OrderYear":2011,"TotalQuantity":373,"TotalSalesAmount":800119.679268},{"ProductID":771,"ProductName":"Mountain-100 Silver, 38","OrderYear":2011,"TotalQuantity":351,"TotalSalesAmount":769077.738000},{"ProductID":751,"ProductName":"Road-150 Red, 48","OrderYear":2011,"TotalQuantity":237,"TotalSalesAmount":765034.126000},{"ProductID":775,"ProductName":"Mountain-100 Black, 38","OrderYear":2011,"TotalQuantity":356,"TotalSalesAmount":749388.179584},{"ProductID":776,"ProductName":"Mountain-100 Black, 42","OrderYear":2011,"TotalQuantity":339,"TotalSalesAmount":718943.069792},{"ProductID":773,"ProductName":"Mountain-100 Silver, 44","OrderYear":2011,"TotalQuantity":325,"TotalSalesAmount":698499.385584},{"ProductID":750,"ProductName":"Road-150 Red, 44","OrderYear":2011,"TotalQuantity":217,"TotalSalesAmount":690606.110000},{"ProductID":752,"ProductName":"Road-150 Red, 52","OrderYear":2011,"TotalQuantity":216,"TotalSalesAmount":685596.532000},{"ProductID":778,"ProductName":"Mountain-100 Black, 48","OrderYear":2011,"TotalQuantity":319,"TotalSalesAmount":678372.990000},{"ProductID":772,"ProductName":"Mountain-100 Silver, 42","OrderYear":2011,"TotalQuantity":298,"TotalSalesAmount":634973.972424},{"ProductID":774,"ProductName":"Mountain-100 Silver, 48","OrderYear":2011,"TotalQuantity":260,"TotalSalesAmount":549438.384000},{"ProductID":758,"ProductName":"Road-450 Red, 52","OrderYear":2011,"TotalQuantity":389,"TotalSalesAmount":340294.866000},{"ProductID":754,"ProductName":"Road-450 Red, 58","OrderYear":2011,"TotalQuantity":310,"TotalSalesAmount":270587.197708},{"ProductID":770,"ProductName":"Road-650 Black, 52","OrderYear":2011,"TotalQuantity":415,"TotalSalesAmount":178806.663340},{"ProductID":762,"ProductName":"Road-650 Red, 44","OrderYear":2011,"TotalQuantity":395,"TotalSalesAmount":168203.019200},{"ProductID":760,"ProductName":"Road-650 Red, 60","OrderYear":2011,"TotalQuantity":391,"TotalSalesAmount":167643.740800},{"ProductID":756,"ProductName":"Road-450 Red, 44","OrderYear":2011,"TotalQuantity":180,"TotalSalesAmount":157462.920000},{"ProductID":755,"ProductName":"Road-450 Red, 60","OrderYear":2011,"TotalQuantity":179,"TotalSalesAmount":156588.126000},{"ProductID":765,"ProductName":"Road-650 Black, 58","OrderYear":2011,"TotalQuantity":312,"TotalSalesAmount":133947.209100},{"ProductID":763,"ProductName":"Road-650 Red, 48","OrderYear":2011,"TotalQuantity":306,"TotalSalesAmount":131989.734300},{"ProductID":761,"ProductName":"Road-650 Red, 62","OrderYear":2011,"TotalQuantity":312,"TotalSalesAmount":131989.734000},{"ProductID":766,"ProductName":"Road-650 Black, 60","OrderYear":2011,"TotalQuantity":170,"TotalSalesAmount":75502.602500},{"ProductID":764,"ProductName":"Road-650 Red, 52","OrderYear":2011,"TotalQuantity":170,"TotalSalesAmount":74104.406000},{"ProductID":768,"ProductName":"Road-650 Black, 44","OrderYear":2011,"TotalQuantity":167,"TotalSalesAmount":73964.586500},{"ProductID":741,"ProductName":"HL Mountain Frame - Silver, 48","OrderYear":2011,"TotalQuantity":90,"TotalSalesAmount":73683.000000},{"ProductID":748,"ProductName":"HL Mountain Frame - Silver, 38","OrderYear":2011,"TotalQuantity":91,"TotalSalesAmount":65756.135900},{"ProductID":743,"ProductName":"HL Mountain Frame - Black, 42","OrderYear":2011,"TotalQuantity":88,"TotalSalesAmount":62893.978400},{"ProductID":745,"ProductName":"HL Mountain Frame - Black, 48","OrderYear":2011,"TotalQuantity":72,"TotalSalesAmount":58302.720000},{"ProductID":742,"ProductName":"HL Mountain Frame - Silver, 46","OrderYear":2011,"TotalQuantity":74,"TotalSalesAmount":53472.022600},{"ProductID":732,"ProductName":"ML Road Frame - Red, 48","OrderYear":2011,"TotalQuantity":147,"TotalSalesAmount":52464.006000},{"ProductID":747,"ProductName":"HL Mountain Frame - Black, 38","OrderYear":2011,"TotalQuantity":73,"TotalSalesAmount":52173.413900},{"ProductID":757,"ProductName":"Road-450 Red, 48","OrderYear":2011,"TotalQuantity":58,"TotalSalesAmount":50738.052000},{"ProductID":767,"ProductName":"Road-650 Black, 62","OrderYear":2011,"TotalQuantity":88,"TotalSalesAmount":41106.972700},{"ProductID":725,"ProductName":"LL Road Frame - Red, 44","OrderYear":2011,"TotalQuantity":185,"TotalSalesAmount":34028.567000},{"ProductID":729,"ProductName":"LL Road Frame - Red, 60","OrderYear":2011,"TotalQuantity":181,"TotalSalesAmount":33292.814200},{"ProductID":759,"ProductName":"Road-650 Red, 58","OrderYear":2011,"TotalQuantity":73,"TotalSalesAmount":33137.253400},{"ProductID":738,"ProductName":"LL Road Frame - Black, 52","OrderYear":2011,"TotalQuantity":181,"TotalSalesAmount":32323.124800},{"ProductID":769,"ProductName":"Road-650 Black, 48","OrderYear":2011,"TotalQuantity":74,"TotalSalesAmount":32158.515800},{"ProductID":730,"ProductName":"LL Road Frame - Red, 62","OrderYear":2011,"TotalQuantity":119,"TotalSalesAmount":21888.645800},{"ProductID":726,"ProductName":"LL Road Frame - Red, 48","OrderYear":2011,"TotalQuantity":114,"TotalSalesAmount":20968.954800},{"ProductID":722,"ProductName":"LL Road Frame - Black, 58","OrderYear":2011,"TotalQuantity":112,"TotalSalesAmount":20001.049600},{"ProductID":733,"ProductName":"ML Road Frame - Red, 52","OrderYear":2011,"TotalQuantity":55,"TotalSalesAmount":19629.390000},{"ProductID":715,"ProductName":"Long-Sleeve Logo Jersey, L","OrderYear":2011,"TotalQuantity":544,"TotalSalesAmount":15594.637060},{"ProductID":717,"ProductName":"HL Road Frame - Red, 62","OrderYear":2011,"TotalQuantity":13,"TotalSalesAmount":9854.986700},{"ProductID":739,"ProductName":"HL Mountain Frame - Silver, 42","OrderYear":2011,"TotalQuantity":13,"TotalSalesAmount":9393.733700},{"ProductID":718,"ProductName":"HL Road Frame - Red,

44","OrderYear":2011,"TotalQuantity":11,"TotalSalesAmount":8338.834900},{"ProductID":716,"ProductName":"Long-Sleeve Logo Jersey, XL","OrderYear":2011,"TotalQuantity":255,"TotalSalesAmount":7354.302000},{"ProductID":744,"ProductName":"HL Mountain Frame - Black, 44","OrderYear":2011,"TotalQuantity":9,"TotalSalesAmount":7287.840000},{"ProductID":711,"ProductName":"Sport-100 Helmet, Blue","OrderYear":2011,"TotalQuantity":360,"TotalSalesAmount":7255.445108},{"ProductID":708,"ProductName":"Sport-100 Helmet, Black","OrderYear":2011,"TotalQuantity":341,"TotalSalesAmount":6883.596500},{"ProductID":707,"ProductName":"Sport-100 Helmet, Red","OrderYear":2011,"TotalQuantity":331,"TotalSalesAmount":6681.731500},{"ProductID":714,"ProductName":"Long-Sleeve Logo Jersey, M","OrderYear":2011,"TotalQuantity":228,"TotalSalesAmount":6575.611200},{"ProductID":709,"ProductName":"Mountain Bike Socks, M","OrderYear":2011,"TotalQuantity":608,"TotalSalesAmount":3314.189950},{"ProductID":712,"ProductName":"AWC Logo Cap","OrderYear":2011,"TotalQuantity":545,"TotalSalesAmount":2816.535136},{"ProductID":736,"ProductName":"LL Road Frame - Black, 44","OrderYear":2011,"TotalQuantity":13,"TotalSalesAmount":2321.550400},{"ProductID":727,"ProductName":"LL Road Frame - Red, 52","OrderYear":2011,"TotalQuantity":5,"TotalSalesAmount":919.691000},{"ProductID":710,"ProductName":"Mountain Bike Socks, L","OrderYear":2011,"TotalQuantity":66,"TotalSalesAmount":376.200000},{"ProductID":723,"ProductName":"LL Road Frame - Black, 60","OrderYear":2011,"TotalQuantity":1,"TotalSalesAmount":178.580800}]

## Corrected: Added Average Quantity Per Order to add more complexity and usability.

## **Medium Proposition 12**: Conduct a detailed analysis of product sales performance and order metrics in the WideWorldImporters database to evaluate product popularity and customer purchasing patterns.

- **CTE Usage**: Utilize a Common Table Expression (CTE) named ProductSalesAnalysis to aggregate sales data by stock item, calculating total sales, the number of orders, and average quantity per order.
- **Tables Engaged**: The analysis integrates Sales.InvoiceLines, Sales.Invoices, and Warehouse.StockItems tables to gather sales and stock information required for the analysis.
- **Data Aggregation**: The CTE aggregates data by stock item name and ID, providing insights into total sales, order counts, and average quantity per order, aiding in understanding product performance.
- **Output Goal**: The output will include StockItemName, TotalSales, NumberOfOrders, and AvgQuantityPerOrder, ordered by total sales in descending order to identify top-selling products and customer purchase trends.

In [17]:
```
USE WideWorldImporters;
GO

SELECT
    si.StockItemName,
    SUM(il.LineProfit) AS TotalSales,
    COUNT(DISTINCT i.InvoiceID) AS NumberOfOrders,
    (SELECT AVG(Quantity)
     FROM Sales.InvoiceLines subIL
     JOIN Sales.Invoices subI ON subIL.InvoiceID = subI.InvoiceID
     WHERE subIL.StockItemID = il.StockItemID) AS AvgQuantityPerOrder
FROM
    Sales.InvoiceLines il
JOIN Sales.Invoices i ON il.InvoiceID = i.InvoiceID
JOIN Warehouse.StockItems si ON il.StockItemID = si.StockItemID
GROUP BY
    si.StockItemName, il.StockItemID
ORDER BY
    TotalSales DESC;
```

Commands completed successfully.

(227 rows affected)

Total execution time: 00:00:00.176

Out[17]:

| StockItemName | TotalSales | NumberOfOrders | AvgQuantityPerOrder |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 20 mm Double sided bubble wrap 50m | 5293680.00 | 1059 | 54 |
| Air cushion machine (Blue) | 4439391.00 | 1061 | 5 |
| 32 mm Anti static bubble wrap (Blue) 50m | 3526400.00 | 1085 | 56 |
| 10 mm Anti static bubble wrap (Blue) 50m | 3452220.00 | 1119 | 57 |
| 32 mm Double sided bubble wrap 50m | 2929310.00 | 1004 | 55 |
| 10 mm Double sided bubble wrap 50m | 2773400.00 | 1035 | 54 |
| 20 mm Anti static bubble wrap (Blue) 50m | 2670540.00 | 1033 | 55 |
| 32 mm Anti static bubble wrap (Blue) 20m | 1510500.00 | 1088 | 55 |
| Void fill 400 L bag (White) 400L | 1378320.00 | 1039 | 55 |
| 32 mm Double sided bubble wrap 20m | 1323190.00 | 1048 | 54 |
| 20 mm Anti static bubble wrap (Blue) 20m | 1316640.00 | 994 | 55 |
| Void fill 300 L bag (White) 300L | 1116960.00 | 1036 | 55 |
| 10 mm Anti static bubble wrap (Blue) 20m | 1088130.00 | 1061 | 53 |
| 20 mm Double sided bubble wrap 20m | 1064880.00 | 1110 | 53 |
| 32 mm Anti static bubble wrap (Blue) 10m | 929920.00 | 1048 | 55 |
| 20 mm Anti static bubble wrap (Blue) 10m | 894200.00 | 972 | 54 |
| 10 mm Double sided bubble wrap 20m | 892960.00 | 1012 | 55 |
| Tape dispenser (Blue) | 868800.00 | 1038 | 55 |
| Tape dispenser (Black) | 865500.00 | 1039 | 55 |
| USB food flash drive - dessert 10 drive variety pack | 837630.00 | 1065 | 5 |
| "The Gu" red shirt XML tag t-shirt (White) S | 817080.00 | 1099 | 67 |
| Ride on big wheel monster truck (Black) 1/12 scale | 807800.00 | 1043 | 5 |
| "The Gu" red shirt XML tag t-shirt (Black) XXS | 804144.00 | 1085 | 67 |
| "The Gu" red shirt XML tag t-shirt (Black) XS | 796224.00 | 1095 | 66 |
| Ride on vintage American toy coupe (Black) 1/12 scale | 794300.00 | 1103 | 5 |
| Bubblewrap dispenser (Red) 1.5m | 794040.00 | 1100 | 5 |
| USB food flash drive - dim sum 10 drive variety pack | 792775.50 | 1015 | 5 |
| "The Gu" red shirt XML tag t-shirt (Black) M | 761112.00 | 1060 | 65 |
| "The Gu" red shirt XML tag t-shirt (White) 3XS | 757812.00 | 1040 | 66 |
| "The Gu" red shirt XML tag t-shirt (White) XXL | 754992.00 | 1080 | 66 |
| "The Gu" red shirt XML tag t-shirt (Black) 3XS | 741708.00 | 1024 | 65 |
| 32 mm Double sided bubble wrap 10m | 741488.00 | 1089 | 56 |
| "The Gu" red shirt XML tag t-shirt (White) L | 739200.00 | 1026 | 65 |
| "The Gu" red shirt XML tag t-shirt (Black) S | 735636.00 | 1014 | 65 |
| Ride on vintage American toy coupe (Red) 1/12 scale | 735540.00 | 1020 | 5 |
| "The Gu" red shirt XML tag t-shirt (Black) L | 730752.00 | 983 | 67 |
| "The Gu" red shirt XML tag t-shirt (Black) XXL | 720468.00 | 1036 | 66 |
| "The Gu" red shirt XML tag t-shirt (White) 7XL | 715692.00 | 1110 | 67 |
| Bubblewrap dispenser (Black) 1.5m | 715130.00 | 1019 | 5 |
| Bubblewrap dispenser (Blue) 1.5m | 713180.00 | 1019 | 5 |

| | | | |
|---|---|---|---|
| DBA joke mug - daaaaaa-ta (Black) | 48654.00 | 1057 | 5 |
| Developer joke mug - Oct 31 = Dec 25 (White) | 48603.00 | 1028 | 5 |
| IT joke mug - keyboard not found … press F1 to continue (Black) | 48501.00 | 1035 | 5 |
| IT joke mug - keyboard not found … press F1 to continue (White) | 48365.00 | 1042 | 5 |
| Superhero action jacket (Blue) M | 48112.00 | 1071 | 5 |
| IT joke mug - that behavior is by design (White) | 47999.50 | 1029 | 5 |
| Developer joke mug - a foo walks into a bar (White) | 47872.00 | 1034 | 5 |
| Developer joke mug - this code was generated by a tool (Black) | 47744.50 | 1027 | 5 |
| Developer joke mug - there are 10 types of people in the world (Black) | 47702.00 | 1049 | 5 |
| Developer joke mug - inheritance is the OO way to become wealthy (White) | 47702.00 | 1042 | 5 |
| DBA joke mug - I will get you in order (White) | 47472.50 | 1045 | 5 |
| Developer joke mug - old C developers never die (Black) | 47362.00 | 1034 | 5 |
| Developer joke mug - understanding recursion requires understanding recursion (Black) | 47260.00 | 1055 | 5 |
| Developer joke mug - when your hammer is C++ (Black) | 47251.50 | 1019 | 5 |
| DBA joke mug - SELECT caffeine FROM mug (Black) | 46503.50 | 1035 | 5 |
| DBA joke mug - mind if I join you? (Black) | 45917.00 | 1012 | 5 |
| 3 kg Courier post bag (White) 300x190x95mm | 45712.50 | 1077 | 141 |
| Superhero action jacket (Blue) L | 43712.00 | 1027 | 5 |
| Superhero action jacket (Blue) XXL | 43232.00 | 1037 | 5 |
| Superhero action jacket (Blue) XL | 42984.00 | 995 | 5 |
| Superhero action jacket (Blue) XS | 36510.00 | 1080 | 5 |
| Superhero action jacket (Blue) XXS | 36234.00 | 1083 | 5 |
| Superhero action jacket (Blue) 3XS | 34188.00 | 1061 | 5 |
| Superhero action jacket (Blue) S | 32556.00 | 1005 | 5 |
| Packing knife with metal insert blade (Yellow) 18mm | 32436.00 | 995 | 27 |
| Packing knife with metal insert blade (Yellow) 9mm | 30046.50 | 1101 | 27 |
| Halloween zombie mask (Light Brown) L | -64104.00 | 982 | 65 |
| Halloween zombie mask (Light Brown) M | -67008.00 | 1017 | 65 |
| Halloween zombie mask (Light Brown) S | -71016.00 | 1044 | 68 |
| Halloween zombie mask (Light Brown) XL | -72372.00 | 1071 | 67 |

In [25]:

```
USE WideWorldImporters;
GO

SELECT
    si.StockItemName,
    SUM(il.LineProfit) AS TotalSales,
    COUNT(DISTINCT i.InvoiceID) AS NumberOfOrders,
    (SELECT AVG(Quantity)
     FROM Sales.InvoiceLines subIL
     JOIN Sales.Invoices subI ON subIL.InvoiceID = subI.InvoiceID
     WHERE subIL.StockItemID = il.StockItemID) AS AvgQuantityPerOrder
```

```
FROM
    Sales.InvoiceLines il
JOIN Sales.Invoices i ON il.InvoiceID = i.InvoiceID
JOIN Warehouse.StockItems si ON il.StockItemID = si.StockItemID
GROUP BY
    si.StockItemName, il.StockItemID
ORDER BY
    TotalSales DESC
FOR JSON PATH;
```

Commands completed successfully.

(227 rows affected)

Total execution time: 00:00:00.136

Out[25]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**


[{"StockItemName":"20 mm Double sided bubble wrap 50m","TotalSales":5293680.00,"NumberOfOrders":1059,"AvgQuantityPerOrder":54},{"StockItemName":"Air cushion machine (Blue)","TotalSales":4439391.00,"NumberOfOrders":1061,"AvgQuantityPerOrder":5},{"StockItemName":"32 mm Anti static bubble wrap (Blue) 50m","TotalSales":3526400.00,"NumberOfOrders":1085,"AvgQuantityPerOrder":56},{"StockItemName":"10 mm Anti static bubble wrap (Blue) 50m","TotalSales":3452220.00,"NumberOfOrders":1119,"AvgQuantityPerOrder":57},{"StockItemName":"32 mm Double sided bubble wrap 50m","TotalSales":2929310.00,"NumberOfOrders":1004,"AvgQuantityPerOrder":55},{"StockItemName":"10 mm Double sided bubble wrap 50m","TotalSales":2773400.00,"NumberOfOrders":1035,"AvgQuantityPerOrder":54},{"StockItemName":"20 mm Anti static bubble wrap (Blue) 50m","TotalSales":2670540.00,"NumberOfOrders":1033,"AvgQuantityPerOrder":55},{"StockItemName":"32 mm Anti static bubble wrap (Blue) 20m","TotalSales":1510500.00,"NumberOfOrders":1088,"AvgQuantityPerOrder":55},{"StockItemName":"Void fill 400 L bag (White) 400L","TotalSales":1378320.00,"NumberOfOrders":1039,"AvgQuantityPerOrder":55},{"StockItemName":"32 mm Double sided bubble wrap 20m","TotalSales":1323190.00,"NumberOfOrders":1048,"AvgQuantityPerOrder":54},{"StockItemName":"20 mm Anti static bubble wrap (Blue) 20m","TotalSales":1316640.00,"NumberOfOrders":994,"AvgQuantityPerOrder":55},{"StockItemName":"Void fill 300 L bag (White) 300L","TotalSales":1116960.00,"NumberOfOrders":1036,"AvgQuantityPerOrder":55},{"StockItemName":"10 mm Anti static bubble wrap (Blue) 20m","TotalSales":1088130.00,"NumberOfOrders":1061,"AvgQuantityPerOrder":53},{"StockItemName":"20 mm Double sided bubble wrap 20m","TotalSales":1064880.00,"NumberOfOrders":1110,"AvgQuantityPerOrder":53},{"StockItemName":"32 mm Anti static bubble wrap (Blue) 10m","TotalSales":929920.00,"NumberOfOrders":1048,"AvgQuantityPerOrder":55},{"StockItemName":"20 mm Anti static bubble wrap (Blue) 10m","TotalSales":894200.00,"NumberOfOrders":972,"AvgQuantityPerOrder":54},{"StockItemName":"10 mm Double sided bubble wrap 20m","TotalSales":892960.00,"NumberOfOrders":1012,"AvgQuantityPerOrder":55},{"StockItemName":"Tape dispenser (Blue)","TotalSales":868800.00,"NumberOfOrders":1038,"AvgQuantityPerOrder":55},{"StockItemName":"Tape dispenser (Black)","TotalSales":865500.00,"NumberOfOrders":1039,"AvgQuantityPerOrder":55},{"StockItemName":"USB food flash drive - dessert 10 drive variety pack","TotalSales":837630.00,"NumberOfOrders":1065,"AvgQuantityPerOrder":5},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (White) S","TotalSales":817080.00,"NumberOfOrders":1099,"AvgQuantityPerOrder":67},{"StockItemName":"Ride on big wheel monster truck (Black) 1\/12 scale","TotalSales":807800.00,"NumberOfOrders":1043,"AvgQuantityPerOrder":5},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (Black) XXS","TotalSales":804144.00,"NumberOfOrders":1085,"AvgQuantityPerOrder":67},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (Black) XS","TotalSales":796224.00,"NumberOfOrders":1095,"AvgQuantityPerOrder":66},{"StockItemName":"Ride on vintage American toy coupe (Black) 1\/12 scale","TotalSales":794300.00,"NumberOfOrders":1103,"AvgQuantityPerOrder":5},{"StockItemName":"Bubblewrap dispenser (Red) 1.5m","TotalSales":794040.00,"NumberOfOrders":1100,"AvgQuantityPerOrder":5},{"StockItemName":"USB food flash drive - dim sum 10 drive variety pack","TotalSales":792775.50,"NumberOfOrders":1015,"AvgQuantityPerOrder":5},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (Black) M","TotalSales":761112.00,"NumberOfOrders":1060,"AvgQuantityPerOrder":65},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (White) 3XS","TotalSales":757812.00,"NumberOfOrders":1040,"AvgQuantityPerOrder":66},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (White) XXL","TotalSales":754992.00,"NumberOfOrders":1080,"AvgQuantityPerOrder":66},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (Black) 3XS","TotalSales":741708.00,"NumberOfOrders":1024,"AvgQuantityPerOrder":65},{"StockItemName":"32 mm Double sided bubble wrap 10m","TotalSales":741488.00,"NumberOfOrders":1089,"AvgQuantityPerOrder":56},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (White) L","TotalSales":739200.00,"NumberOfOrders":1026,"AvgQuantityPerOrder":65},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (Black) S","TotalSales":735636.00,"NumberOfOrders":1014,"AvgQuantityPerOrder":65},{"StockItemName":"Ride on vintage American toy coupe (Red) 1\/12 scale","TotalSales":735540.00,"NumberOfOrders":1020,"AvgQuantityPerOrder":5},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (Black) L","TotalSales":730752.00,"NumberOfOrders":983,"AvgQuantityPerOrder":67},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (Black) XXL","TotalSales":720468.00,"NumberOfOrders":1036,"AvgQuantityPerOrder":66},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (White) 7XL","TotalSales":715692.00,"NumberOfOrders":1110,"AvgQuantityPerOrder":67},{"StockItemName":"Bubblewrap dispenser (Black) 1.5m","TotalSales":715130.00,"NumberOfOrders":1019,"AvgQuantityPerOrder":5},{"StockItemName":"Bubblewrap dispenser (Blue) 1.5m","TotalSales":713180.00,"NumberOfOrders":1019,"AvgQuantityPerOrder":5},{"StockItemName":"\"The Gu\" red shirt XML tag t-shirt (White) XL","TotalSales":710640.00,"NumberOfOrders":1016,"AvgQuantityPerOrder":66},

(Black)","TotalSales":48501.00,"NumberOfOrders":1035,"AvgQuantityPerOrder":5},{"StockItemName":"IT joke mug - keyboard not found … press F1 to continue (White)","TotalSales":48365.00,"NumberOfOrders":1042,"AvgQuantityPerOrder":5},{"StockItemName":"Superhero action jacket (Blue) M","TotalSales":48112.00,"NumberOfOrders":1071,"AvgQuantityPerOrder":5},{"StockItemName":"IT joke mug - that behavior is by design (White)","TotalSales":47999.50,"NumberOfOrders":1029,"AvgQuantityPerOrder":5},{"StockItemName":"Developer joke mug - a foo walks into a bar (White)","TotalSales":47872.00,"NumberOfOrders":1034,"AvgQuantityPerOrder":5},{"StockItemName":"Developer joke mug - this code was generated by a tool (Black)","TotalSales":47744.50,"NumberOfOrders":1027,"AvgQuantityPerOrder":5},{"StockItemName":"Developer joke mug - there are 10 types of people in the world (Black)","TotalSales":47702.00,"NumberOfOrders":1049,"AvgQuantityPerOrder":5},{"StockItemName":"Developer joke mug - inheritance is the OO way to become wealthy (White)","TotalSales":47702.00,"NumberOfOrders":1042,"AvgQuantityPerOrder":5},{"StockItemName":"DBA joke mug - I will get you in order (White)","TotalSales":47472.50,"NumberOfOrders":1045,"AvgQuantityPerOrder":5},{"StockItemName":"Developer joke mug - old C developers never die (Black)","TotalSales":47362.00,"NumberOfOrders":1034,"AvgQuantityPerOrder":5},{"StockItemName":"Developer joke mug - understanding recursion requires understanding recursion (Black)","TotalSales":47260.00,"NumberOfOrders":1055,"AvgQuantityPerOrder":5},{"StockItemName":"Developer joke mug - when your hammer is C++ (Black)","TotalSales":47251.50,"NumberOfOrders":1019,"AvgQuantityPerOrder":5},{"StockItemName":"DBA joke mug - SELECT caffeine FROM mug (Black)","TotalSales":46503.50,"NumberOfOrders":1035,"AvgQuantityPerOrder":5},{"StockItemName":"DBA joke mug - mind if I join you? (Black)","TotalSales":45917.00,"NumberOfOrders":1012,"AvgQuantityPerOrder":5},{"StockItemName":"3 kg Courier post bag (White) 300x190x95mm","TotalSales":45712.50,"NumberOfOrders":1077,"AvgQuantityPerOrder":141},{"StockItemName":"Superhero action jacket (Blue) L","TotalSales":43712.00,"NumberOfOrders":1027,"AvgQuantityPerOrder":5},{"StockItemName":"Superhero action jacket (Blue) XXL","TotalSales":43232.00,"NumberOfOrders":1037,"AvgQuantityPerOrder":5},{"StockItemName":"Superhero action jacket (Blue) XL","TotalSales":42984.00,"NumberOfOrders":995,"AvgQuantityPerOrder":5},{"StockItemName":"Superhero action jacket (Blue) XS","TotalSales":36510.00,"NumberOfOrders":1080,"AvgQuantityPerOrder":5},{"StockItemName":"Superhero action jacket (Blue) XXS","TotalSales":36234.00,"NumberOfOrders":1083,"AvgQuantityPerOrder":5},{"StockItemName":"Superhero action jacket (Blue) 3XS","TotalSales":34188.00,"NumberOfOrders":1061,"AvgQuantityPerOrder":5},{"StockItemName":"Superhero action jacket (Blue) S","TotalSales":32556.00,"NumberOfOrders":1005,"AvgQuantityPerOrder":5},{"StockItemName":"Packing knife with metal insert blade (Yellow) 18mm","TotalSales":32436.00,"NumberOfOrders":995,"AvgQuantityPerOrder":27},{"StockItemName":"Packing knife with metal insert blade (Yellow) 9mm","TotalSales":30046.50,"NumberOfOrders":1101,"AvgQuantityPerOrder":27},{"StockItemName":"Halloween zombie mask (Light Brown) L","TotalSales":-64104.00,"NumberOfOrders":982,"AvgQuantityPerOrder":65},{"StockItemName":"Halloween zombie mask (Light Brown) M","TotalSales":-67008.00,"NumberOfOrders":1017,"AvgQuantityPerOrder":65},{"StockItemName":"Halloween zombie mask (Light Brown) S","TotalSales":-71016.00,"NumberOfOrders":1044,"AvgQuantityPerOrder":68},{"StockItemName":"Halloween zombie mask (Light Brown) XL","TotalSales":-72372.00,"NumberOfOrders":1071,"AvgQuantityPerOrder":67}]

## **Complex Proposition 7**: Evaluate top customers based on their total revenue in the WideWorldImporters database to identify key revenue contributors.

- **Custom Function**: The fn_GetTotalRevenue function computes the total revenue for a given CustomerID by summing the product of quantity and unit price from Sales.InvoiceLines, filtered by CustomerID.
- **CTE Usage**: The CustomerRevenue CTE calculates the total revenue for each customer using the custom function fn_GetTotalRevenue, grouping by CustomerID and CustomerName.
- **Tables Engaged**: The analysis involves Sales.Customers, Sales.Invoices, and Sales.InvoiceLines to access customer information, invoices, and invoice line details.
- **Data Aggregation**: Aggregates total revenue per customer within the CTE to determine top revenue-generating customers.
- **Output Goal**: The output includes CustomerID, CustomerName, and TotalRevenue, sorted in descending order by TotalRevenue to highlight the top customers in terms of revenue contribution.

In [13]:
```
USE WideWorldImporters;
GO
CREATE OR ALTER FUNCTION dbo.fn_GetTotalRevenue (@CustomerID INT)
RETURNS DECIMAL(18,2)
AS
BEGIN
    DECLARE @TotalRevenue DECIMAL(18,2);
    SELECT @TotalRevenue = SUM(InvoiceLines.Quantity * InvoiceLines.UnitPrice)
```

```
    FROM Sales.InvoiceLines
    JOIN Sales.Invoices ON InvoiceLines.InvoiceID = Invoices.InvoiceID
    WHERE Invoices.CustomerID = @CustomerID;
    RETURN @TotalRevenue;
END;
GO

-- Query to retrieve top customers based on total revenue
WITH CustomerRevenue AS (
    SELECT
        Customers.CustomerID,
        Customers.CustomerName,
        dbo.fn_GetTotalRevenue(Customers.CustomerID) AS TotalRevenue
    FROM
        Sales.Customers
    JOIN
        Sales.Invoices ON Customers.CustomerID = Invoices.CustomerID
    JOIN
        Sales.InvoiceLines ON Invoices.InvoiceID = InvoiceLines.InvoiceID
    GROUP BY
        Customers.CustomerID,
        Customers.CustomerName
)
SELECT
    CR.CustomerID,
    CR.CustomerName,
    CR.TotalRevenue
FROM
    CustomerRevenue CR
ORDER BY
    CR.TotalRevenue DESC;
```

Commands completed successfully.

Commands completed successfully.

(663 rows affected)

Total execution time: 00:00:02.608

Out[13]:

| CustomerID | CustomerName | TotalRevenue |
|---|---|---|
| 149 | Tailspin Toys (Inguadona, MN) | 381585.35 |
| 132 | Tailspin Toys (Minidoka, ID) | 371822.30 |
| 977 | Mauno Laurila | 369058.30 |
| 580 | Wingtip Toys (Sarversville, PA) | 365427.00 |
| 954 | Nasrin Omidzadeh | 361939.75 |
| 14 | Tailspin Toys (Long Meadow, MD) | 360901.50 |
| 964 | Ingrida Zeltina | 359859.45 |
| 472 | Wingtip Toys (San Jacinto, CA) | 355293.35 |
| 996 | Laszlo Gardenier | 354680.80 |
| 841 | Camille Authier | 353499.15 |
| 1001 | Dinh Mai | 353177.75 |
| 593 | Wingtip Toys (Cuyamungue, NM) | 353046.95 |
| 550 | Wingtip Toys (Morrison Bluff, AR) | 352287.20 |
| 510 | Wingtip Toys (Grabill, IN) | 351138.65 |
| 569 | Wingtip Toys (West Frostproof, FL) | 346621.45 |
| 874 | Daniel Martensson | 346062.20 |

| | | |
|---|---|---|
| 1033 | Cuneyt Arslan | 173816.50 |
| 573 | Wingtip Toys (Marin City, CA) | 169931.60 |
| 1022 | Nadir Seddigh | 169357.35 |
| 1032 | Som Mukherjee | 166189.70 |
| 1034 | Aishwarya Dantuluri | 165909.10 |
| 1027 | Serdar ozden | 162660.00 |
| 1036 | Erik Malk | 160249.00 |
| 577 | Wingtip Toys (Cherryplain, NY) | 154555.45 |
| 1021 | Fabrice Cloutier | 153963.15 |
| 869 | Abel Tatarescu | 153664.05 |
| 1026 | Daniella Cavalcante | 146813.35 |
| 1043 | Raj Verma | 145724.40 |
| 1045 | Matteo Cattaneo | 144951.40 |
| 1031 | Dipti Shah | 140446.35 |
| 1030 | Chompoo Atitarn | 138137.40 |
| 1035 | Manjunatha Karnik | 136118.00 |
| 1028 | Emma Van Zant | 134261.00 |
| 1040 | Damodar Shenoy | 131871.90 |
| 1042 | Nguyen Banh | 111939.50 |
| 1041 | Tomo Vidovic | 108959.40 |
| 1038 | Damodara Trivedi | 106879.45 |
| 1046 | Christian Couet | 97806.65 |
| 1039 | Bhaamini Palagummi | 93465.40 |
| 1044 | Hanita Nookala | 92791.65 |
| 1052 | Ian Olofsson | 66913.85 |
| 1048 | Abhra Ganguly | 58243.35 |
| 1055 | Adriana Pena | 55629.05 |
| 1058 | Jaroslav Fisar | 54552.40 |
| 1054 | Emma Salpa | 51439.35 |
| 1047 | Ivana Hadrabova | 49336.95 |
| 1049 | Amet Shergill | 48685.80 |
| 1057 | Ganesh Majumdar | 46772.25 |
| 1056 | Kalyani Benjaree | 44808.45 |
| 1050 | Amrita Ganguly | 42138.15 |
| 1053 | Luis Saucedo | 40036.80 |
| 1051 | Sylvie Laramee | 38734.50 |
| 1061 | Agrita Abele | 22829.65 |
| 1059 | Jibek Juniskyzy | 13208.10 |
| 1060 | Anand Mudaliyar | 7240.20 |

In [26]:
```
USE WideWorldImporters;
GO
CREATE OR ALTER FUNCTION dbo.fn_GetTotalRevenue (@CustomerID INT)
RETURNS DECIMAL(18,2)
AS
BEGIN
    DECLARE @TotalRevenue DECIMAL(18,2);
    SELECT @TotalRevenue = SUM(InvoiceLines.Quantity * InvoiceLines.UnitPrice)
    FROM Sales.InvoiceLines
    JOIN Sales.Invoices ON InvoiceLines.InvoiceID = Invoices.InvoiceID
    WHERE Invoices.CustomerID = @CustomerID;
    RETURN @TotalRevenue;
END;
GO

-- Query to retrieve top customers based on total revenue
WITH CustomerRevenue AS (
    SELECT
        Customers.CustomerID,
        Customers.CustomerName,
        dbo.fn_GetTotalRevenue(Customers.CustomerID) AS TotalRevenue
    FROM
        Sales.Customers
    JOIN
        Sales.Invoices ON Customers.CustomerID = Invoices.CustomerID
    JOIN
        Sales.InvoiceLines ON Invoices.InvoiceID = InvoiceLines.InvoiceID
    GROUP BY
        Customers.CustomerID,
        Customers.CustomerName
)
SELECT
    CR.CustomerID,
    CR.CustomerName,
    CR.TotalRevenue
FROM
    CustomerRevenue CR
ORDER BY
    CR.TotalRevenue DESC
FOR JSON PATH;
```

Commands completed successfully.

Commands completed successfully.

(663 rows affected)

Total execution time: 00:00:01.931

Out[26]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**


[{"CustomerID":149,"CustomerName":"Tailspin Toys (Inguadona, MN)","TotalRevenue":381585.35},
{"CustomerID":132,"CustomerName":"Tailspin Toys (Minidoka, ID)","TotalRevenue":371822.30},
{"CustomerID":977,"CustomerName":"Mauno Laurila","TotalRevenue":369058.30},
{"CustomerID":580,"CustomerName":"Wingtip Toys (Sarversville, PA)","TotalRevenue":365427.00},
{"CustomerID":954,"CustomerName":"Nasrin Omidzadeh","TotalRevenue":361939.75},
{"CustomerID":14,"CustomerName":"Tailspin Toys (Long Meadow, MD)","TotalRevenue":360901.50},
{"CustomerID":964,"CustomerName":"Ingrida Zeltina","TotalRevenue":359859.45},
{"CustomerID":472,"CustomerName":"Wingtip Toys (San Jacinto, CA)","TotalRevenue":355293.35},
{"CustomerID":996,"CustomerName":"Laszlo Gardenier","TotalRevenue":354680.80},
{"CustomerID":841,"CustomerName":"Camille Authier","TotalRevenue":353499.15},
{"CustomerID":1001,"CustomerName":"Dinh Mai","TotalRevenue":353177.75},{"CustomerID":593,"CustomerName":"Wingtip Toys (Cuyamungue, NM)","TotalRevenue":353046.95},{"CustomerID":550,"CustomerName":"Wingtip Toys (Morrison Bluff, AR)","TotalRevenue":352287.20},{"CustomerID":510,"CustomerName":"Wingtip Toys (Grabill, IN)","TotalRevenue":351138.65},{"CustomerID":569,"CustomerName":"Wingtip Toys (West Frostproof, FL)","TotalRevenue":346621.45},{"CustomerID":874,"CustomerName":"Daniel Martensson","TotalRevenue":346062.20},
{"CustomerID":856,"CustomerName":"Satish Mittal","TotalRevenue":345839.10},
{"CustomerID":861,"CustomerName":"Amarasimha Vinjamuri","TotalRevenue":344208.65},
{"CustomerID":102,"CustomerName":"Tailspin Toys (Fieldbrook, CA)","TotalRevenue":344005.05},
{"CustomerID":480,"CustomerName":"Wingtip Toys (Wapinitia, OR)","TotalRevenue":343236.10},
{"CustomerID":949,"CustomerName":"Seo-yun Paik","TotalRevenue":342138.85},

{"CustomerID":1052,"CustomerName":"Ian Olofsson","TotalRevenue":66913.85},
{"CustomerID":1048,"CustomerName":"Abhra Ganguly","TotalRevenue":58243.35},
{"CustomerID":1055,"CustomerName":"Adriana Pena","TotalRevenue":55629.05},
{"CustomerID":1058,"CustomerName":"Jaroslav Fisar","TotalRevenue":54552.40},
{"CustomerID":1054,"CustomerName":"Emma Salpa","TotalRevenue":51439.35},
{"CustomerID":1047,"CustomerName":"Ivana Hadrabova","TotalRevenue":49336.95},
{"CustomerID":1049,"CustomerName":"Amet Shergill","TotalRevenue":48685.80},
{"CustomerID":1057,"CustomerName":"Ganesh Majumdar","TotalRevenue":46772.25},
{"CustomerID":1056,"CustomerName":"Kalyani Benjaree","TotalRevenue":44808.45},
{"CustomerID":1050,"CustomerName":"Amrita Ganguly","TotalRevenue":42138.15},
{"CustomerID":1053,"CustomerName":"Luis Saucedo","TotalRevenue":40036.80},
{"CustomerID":1051,"CustomerName":"Sylvie Laramee","TotalRevenue":38734.50},
{"CustomerID":1061,"CustomerName":"Agrita Abele","TotalRevenue":22829.65},
{"CustomerID":1059,"CustomerName":"Jibek Juniskyzy","TotalRevenue":13208.10},
{"CustomerID":1060,"CustomerName":"Anand Mudaliyar","TotalRevenue":7240.20}]

## **Medium Proposition 13**: Evaluate customer performance based on total orders, total order amount, and average order amount in the WideWorldImportersDW database to identify customer behavior trends.

- **CTE Usage**: The CustomerPerformance CTE calculates the total orders, total order amount, and average order amount for each customer by joining the Dimension.Customer table with the Fact.[Order] table.
- **Tables Engaged**: The analysis involves Dimension.Customer for customer details and Fact.[Order] for order-related information, connecting them using Customer Key and Order Key.
- **Data Aggregation**: Aggregates order-related metrics per customer within the CTE to provide insights into customer behavior and purchasing patterns.
- **Output Goal**: The output lists the top 15 customers based on their average order amount in ascending order, showcasing customers with lower average order amounts at the top to highlight potential areas for improvement or targeting.

In [12]:
```
USE WideWorldImportersDW;
GO

WITH CustomerPerformance AS (
    SELECT
        c.Customer,
        COUNT(o.[Order Key]) AS TotalOrders,
        SUM(o.[Total Including Tax]) AS TotalOrderAmount,
        SUM(o.[Total Including Tax]) / COUNT(o.[Order Key]) AS AvgOrderAmount
    FROM
        Dimension.Customer c
    LEFT JOIN
        Fact.[Order] o ON o.[Customer Key] = c.[Customer Key]
    GROUP BY
        c.Customer
)

SELECT TOP 15
    Customer,
    TotalOrders,
    TotalOrderAmount,
    AvgOrderAmount
FROM
    CustomerPerformance
ORDER BY
    AvgOrderAmount ASC;
```

Commands completed successfully.

(15 rows affected)

Total execution time: 00:00:00.066

Out[12]:

| Customer | TotalOrders | TotalOrderAmount | AvgOrderAmount |
|---|---|---|---|
| Wingtip Toys (Marin City, CA) | 302 | 200285.44 | 663.196821 |
| Wingtip Toys (Cape Neddick, ME) | 341 | 228440.40 | 669.913196 |
| Wingtip Toys (Portales, NM) | 345 | 232314.89 | 673.376492 |
| Wingtip Toys (Lucasville, OH) | 413 | 282858.28 | 684.886876 |
| Wingtip Toys (Willow Valley, AZ) | 315 | 216450.79 | 687.145365 |
| Wingtip Toys (Compass Lake, FL) | 369 | 256217.82 | 694.357235 |
| Wingtip Toys (Nuangola, PA) | 313 | 218635.81 | 698.516964 |
| Wingtip Toys (Miesville, MN) | 362 | 254697.24 | 703.583535 |
| Wingtip Toys (Federalsburg, MD) | 423 | 301510.13 | 712.789905 |
| Tailspin Toys (Fairfield Glade, TN) | 348 | 248145.79 | 713.062614 |
| Tailspin Toys (Ekron, KY) | 347 | 249589.33 | 719.277608 |
| Tailspin Toys (Arrow Rock, MO) | 384 | 277556.21 | 722.802630 |
| Tailspin Toys (South Euclid, OH) | 380 | 277404.42 | 730.011631 |
| Tailspin Toys (Stallion Springs, CA) | 386 | 282514.61 | 731.903134 |
| Wingtip Toys (Ovilla, TX) | 320 | 236779.77 | 739.936781 |

In [27]:

```
USE WideWorldImportersDW;
GO

WITH CustomerPerformance AS (
    SELECT
        c.Customer,
        COUNT(o.[Order Key]) AS TotalOrders,
        SUM(o.[Total Including Tax]) AS TotalOrderAmount,
        SUM(o.[Total Including Tax]) / COUNT(o.[Order Key]) AS AvgOrderAmount
    FROM
        Dimension.Customer c
    LEFT JOIN
        Fact.[Order] o ON o.[Customer Key] = c.[Customer Key]
    GROUP BY
        c.Customer
)

SELECT TOP 15
    Customer,
    TotalOrders,
    TotalOrderAmount,
    AvgOrderAmount
FROM
    CustomerPerformance
ORDER BY
    AvgOrderAmount ASC
FOR JSON PATH;
```

Commands completed successfully.

(15 rows affected)

Total execution time: 00:00:00.054

Out[27]:

**JSON_F52E2B61-18A1-11d1-B105-00805F49916B**

[{"Customer":"Wingtip Toys (Marin City, CA)","TotalOrders":302,"TotalOrderAmount":200285.44,"AvgOrderAmount":663.196821},{"Customer":"Wingtip Toys (Cape Neddick, ME)","TotalOrders":341,"TotalOrderAmount":228440.40,"AvgOrderAmount":669.913196},{"Customer":"Wingtip Toys

(Portales, NM)","TotalOrders":345,"TotalOrderAmount":232314.89,"AvgOrderAmount":673.376492},{"Customer":"Wingtip Toys (Lucasville, OH)","TotalOrders":413,"TotalOrderAmount":282858.28,"AvgOrderAmount":684.886876},{"Customer":"Wingtip Toys (Willow Valley, AZ)","TotalOrders":315,"TotalOrderAmount":216450.79,"AvgOrderAmount":687.145365},{"Customer":"Wingtip Toys (Compass Lake, FL)","TotalOrders":369,"TotalOrderAmount":256217.82,"AvgOrderAmount":694.357235},{"Customer":"Wingtip Toys (Nuangola, PA)","TotalOrders":313,"TotalOrderAmount":218635.81,"AvgOrderAmount":698.516964},{"Customer":"Wingtip Toys (Miesville, MN)","TotalOrders":362,"TotalOrderAmount":254697.24,"AvgOrderAmount":703.583535},{"Customer":"Wingtip Toys (Federalsburg, MD)","TotalOrders":423,"TotalOrderAmount":301510.13,"AvgOrderAmount":712.789905},{"Customer":"Tailspin Toys (Fairfield Glade, TN)","TotalOrders":348,"TotalOrderAmount":248145.79,"AvgOrderAmount":713.062614},{"Customer":"Tailspin Toys (Ekron, KY)","TotalOrders":347,"TotalOrderAmount":249589.33,"AvgOrderAmount":719.277608},{"Customer":"Tailspin Toys (Arrow Rock, MO)","TotalOrders":384,"TotalOrderAmount":277556.21,"AvgOrderAmount":722.802630},{"Customer":"Tailspin Toys (South Euclid, OH)","TotalOrders":380,"TotalOrderAmount":277404.42,"AvgOrderAmount":730.011631},{"Customer":"Tailspin Toys (Stallion Springs, CA)","TotalOrders":386,"TotalOrderAmount":282514.61,"AvgOrderAmount":731.903134},{"Customer":"Wingtip Toys (Ovilla, TX)","TotalOrders":320,"TotalOrderAmount":236779.77,"AvgOrderAmount":739.936781}]

*This project was written in collaboration with ChatGPT from OpenAI to improve understanding and assist with the explanation of the queries.*