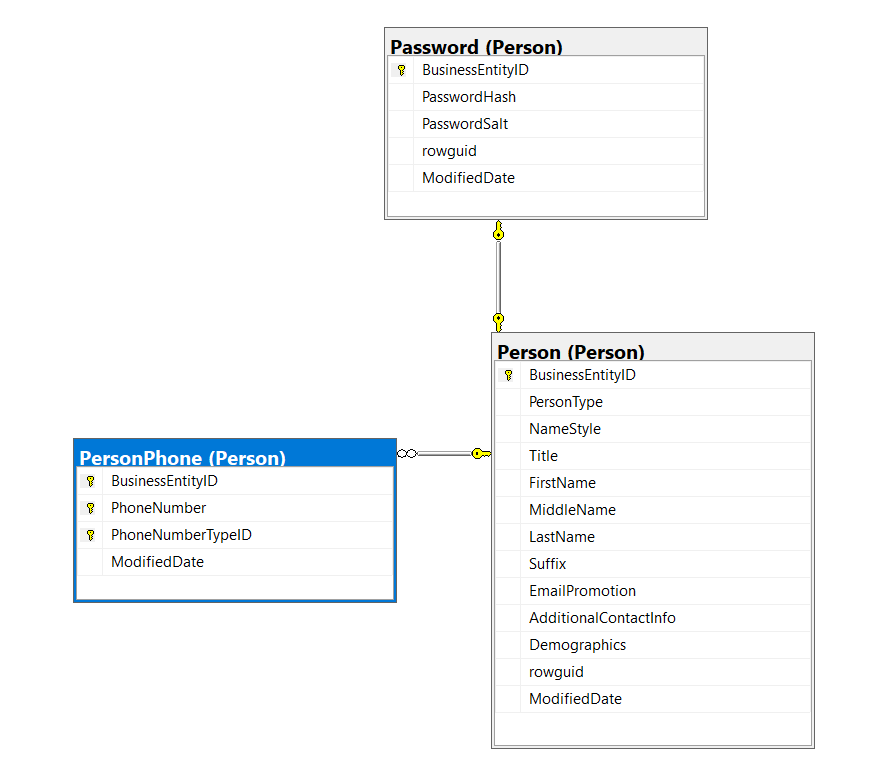


# TOP QUERY (1)

## USE: AdventureWorks2017 Database



* **Proposition:** Create a query that retrieves the business entity IDs of individuals along with their corresponding passwords.
* **Table:** AdventureWorks2017 database. Person.Person table, Person.Password and Person.PersonPhone table

| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

USE AdventureWorks2017;

SELECT TOP (10)

P.BusinessEntityID,

PS.PasswordHash

FROM Person.Person AS P

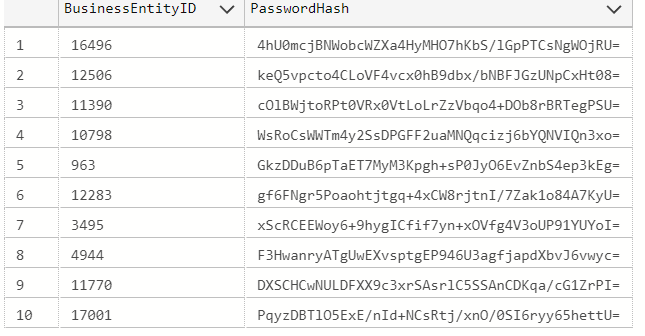
JOIN Person.Password AS PS

ON P.BusinessEntityID = PS.BusinessEntityID

INNER JOIN Person.PersonPhone AS PP

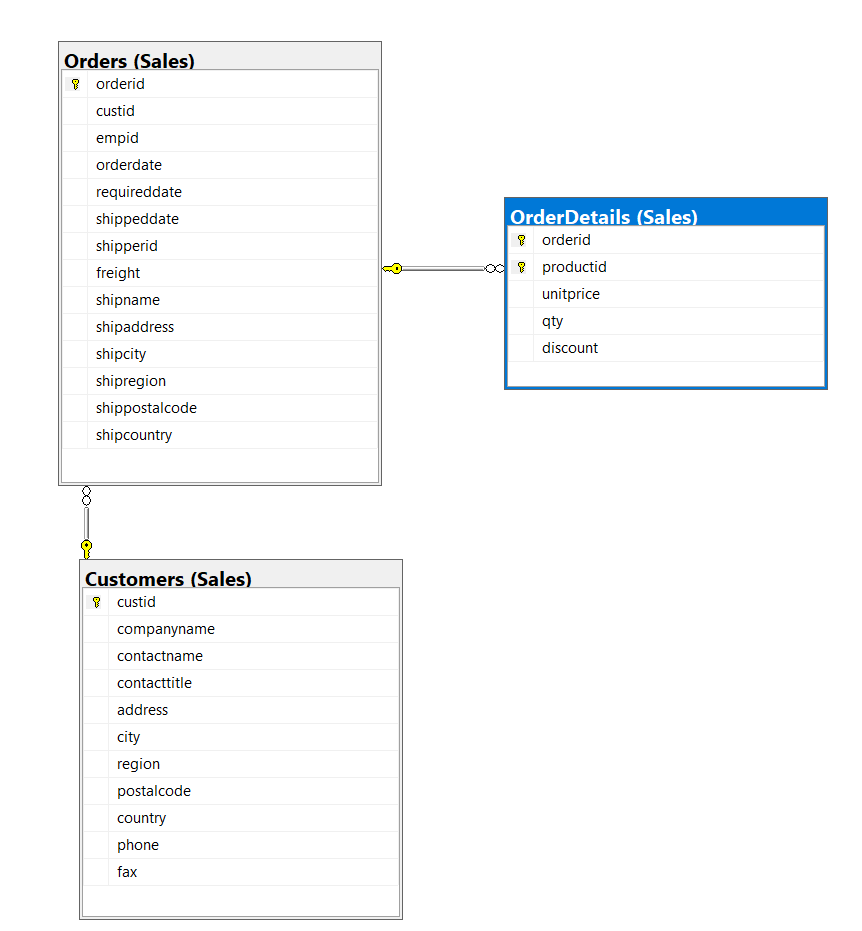
ON PP.BusinessEntityID = PS.BusinessEntityID

FOR JSON PATH, ROOT('person');



# TOP QUERY (2)

## USE: TSQV4 Database



* **Proposition:** Retrieve details of orders made by customers from the Canada, ordered by the total quantity of items ordered by each customer. This query aims to analyze the order behavior of Canadian customers within the Sales database, prioritizing customers based on the total quantity of items they've ordered.
* **Table:** Sales.Customers: Contains information about customers, including custid and country. Sales.Orders: Stores details about orders, including orderid and custid. Sales.OrderDetails: Holds information about order items, including qty and orderid.

| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

USE TSQLV4

SELECT C.custid,

COUNT(DISTINCT O.orderid) AS num\_orders,

SUM(OD.qty) AS total\_quantity\_ordered

FROM Sales.Customers AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

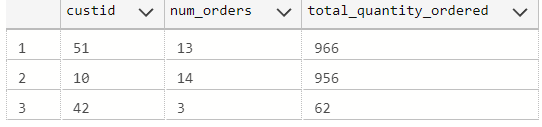
ON OD.orderid = O.orderid

WHERE C.country = N'Canada'

GROUP BY C.custid

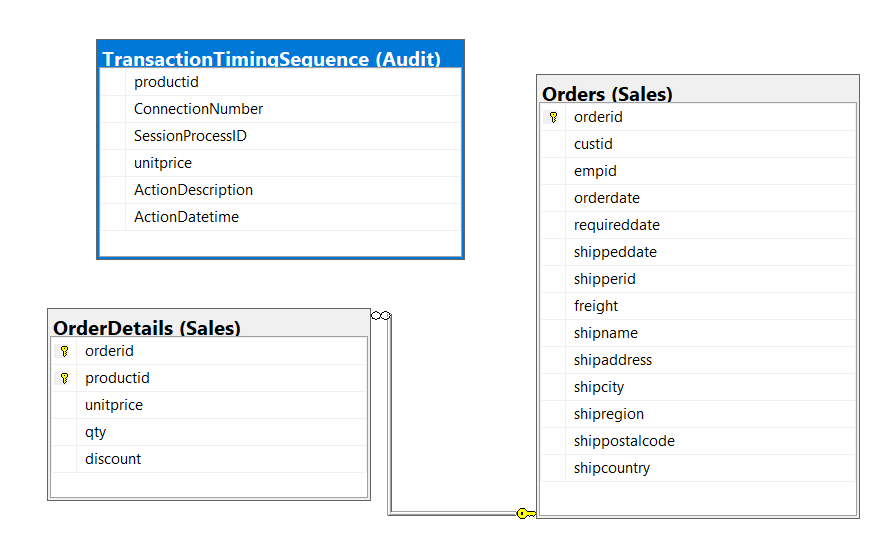
ORDER BY total\_quantity\_ordered DESC

FOR JSON PATH, ROOT('canada');



# TOP QUERY (3)

## USE: TSQV4 Database



* **Proposition:** Utilize the user-defined function dbo.GetCustOrders to extract order details made by a specified customer and analyze the product ordering behavior. This query aims to provide insights into the quantity of products ordered by the designated customer within the TSQLV4 database.
* **Table:** Sales.Orders: Contains comprehensive information regarding orders, including orderid, custid, empid, orderdate, requireddate, shipregion, shippostalcode, and shipcountry. Sales.OrderDetails: Stores detailed data about order items, encompassing orderid, productid, and qty. Audit.TransactionTimingSequence: Records transaction timing details, including productid.

| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

USE TSQLV4

DROP FUNCTION IF EXISTS dbo.GetCustOrders;

go

CREATE FUNCTION dbo.GetCustOrders

(

@cid AS INT

)

RETURNS TABLE

AS

RETURN SELECT orderid,

custid,

empid,

orderdate,

requireddate,

shipregion,

shippostalcode,

shipcountry

FROM Sales.Orders

WHERE custid = @cid

GO

SELECT C.custid,

COUNT(DISTINCT ODA.productid) AS numorders,

SUM(OD.qty) AS totalqty

FROM dbo.GetCustOrders(6) AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

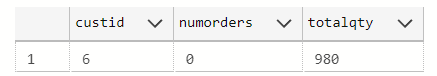
ON OD.orderid = O.orderid

LEFT OUTER JOIN Audit.TransactionTimingSequence AS ODA

ON O.orderid = ODA.productid

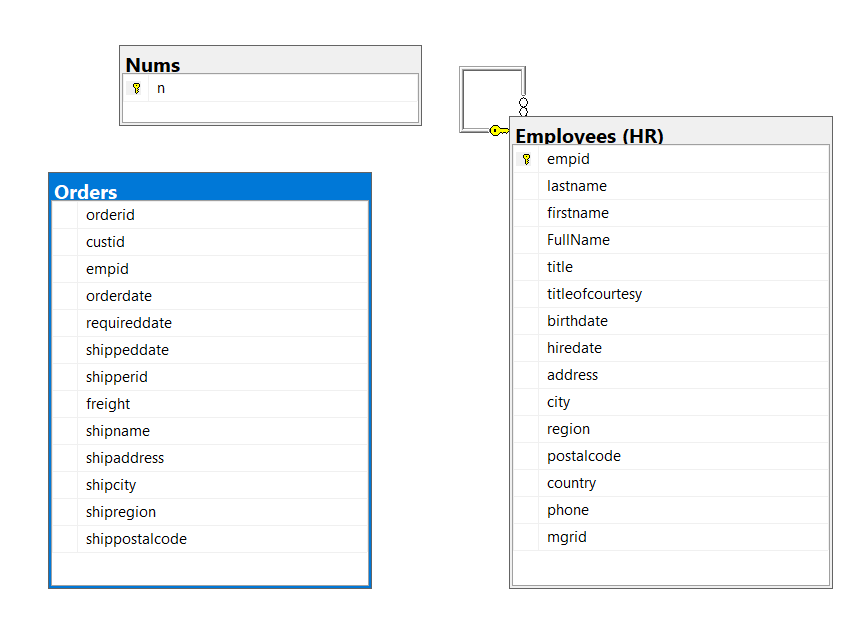
GROUP BY C.custid

FOR JSON PATH, ROOT('six');



# WORST QUERY (1)

## USE: TSQV4 Database



* **Proposition:** Retrieve employee information along with their associated orders by joining the Employees table with a number table and the Orders table. This query aims to analyze employee orders within the TSQLV4 database.
* **Table:** TSQLV4 database, HR.Employees and dbo.Nums tables

| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

USE TSQLV4

SELECT E.empid,

E.firstname,

E.lastname

FROM HR.Employees AS E

JOIN dbo.Nums AS N

ON N.n

BETWEEN 1 AND 5

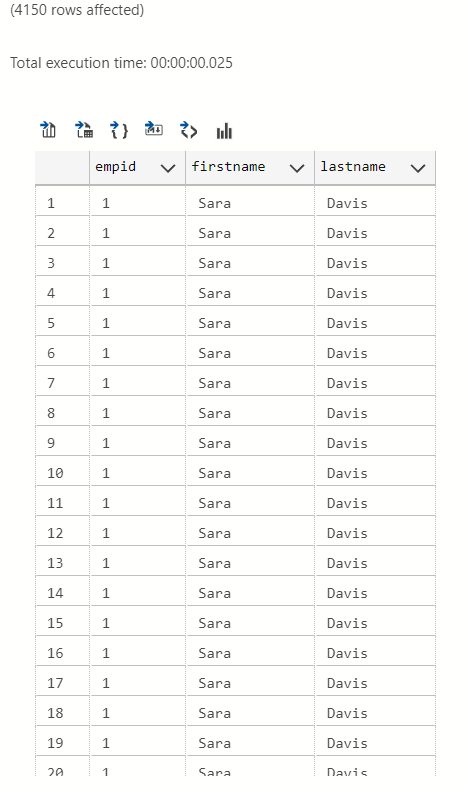
JOIN dbo.Orders AS O

ON E.empid = O.empid

ORDER BY E.empid,

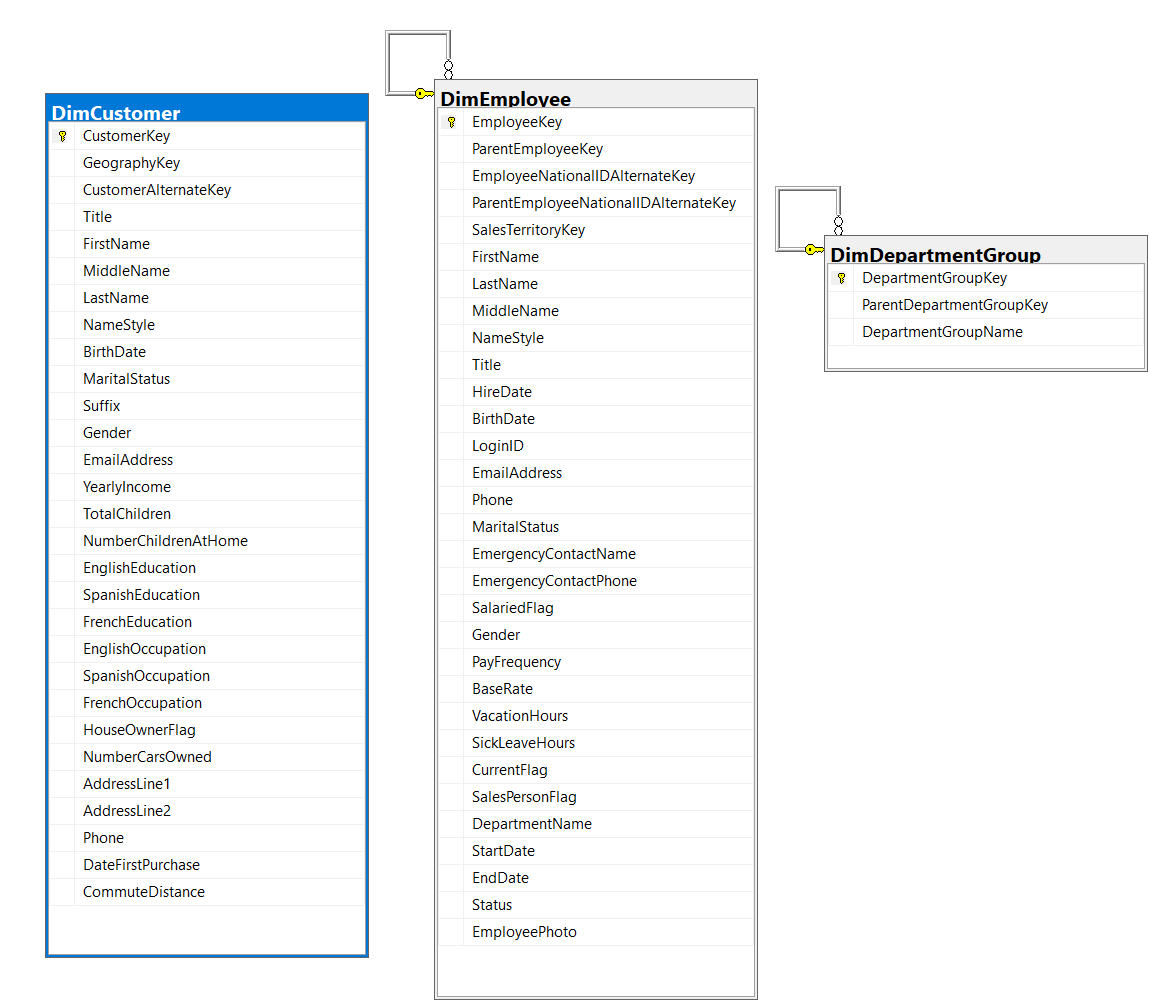
N.n

FOR JSON PATH, ROOT('employees');



# WORST QUERY (2)

## USE: AdventureWorksDW2017 Database



* **Proposition:** Retrieve a list of customers along with their personal information such as first name, middle name, last name, and birth date, who share the same last name as employees and belong to the department groups defined within the organization. This query aims to explore potential familial or organizational relationships between employees and customers within the database, offering insights into potential correlations or connections based on shared attributes.
* **Table:** AdventureWorksDW2017 database. dbo.DimCustomer table, dbo.DimEmployee and dbo.DimDepartmentGroup table

| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

USE AdventureWorksDW2017;

SELECT DC.CustomerKey,

DC.FirstName,

DC.MiddleName,

DC.LastName,

DC.BirthDate

FROM dbo.DimCustomer AS DC

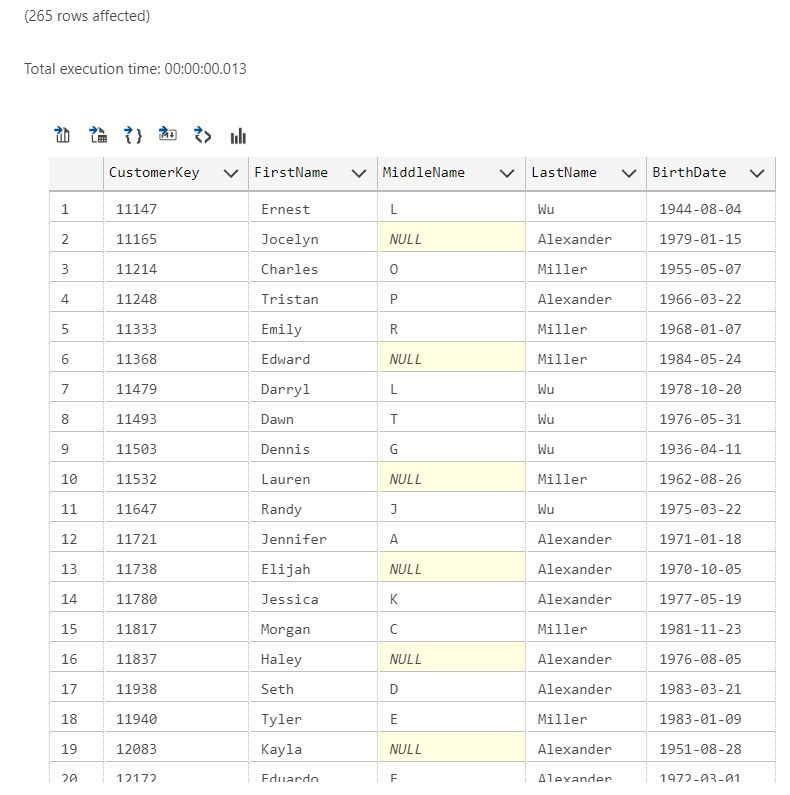
INNER JOIN dbo.DimEmployee AS DE

ON DE.LastName = DC.LastName

INNER JOIN dbo.DimDepartmentGroup AS G

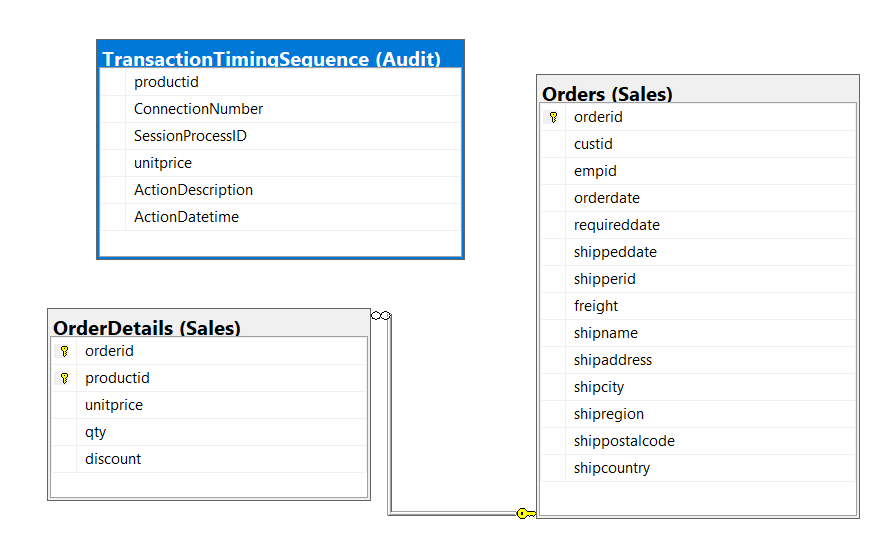
ON G.DepartmentGroupName = DE.DepartmentName

FOR JSON PATH, ROOT('dimcustomer');



# WORST QUERY (3)

## USE: TSQLV4 Database



| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Utilize a user-defined function named dbo.GetCustOrders to retrieve details of orders made by a specific customer, and then analyze the order behavior of that customer. This query aims to provide insights into the order patterns and quantities of items ordered by a particular customer within the Sales database.
* **Table:** Sales.Orders: Contains information about orders, including orderid, custid, empid, orderdate, requireddate, shipregion, shippostalcode, and shipcountry. Sales.OrderDetails: Holds details about order items, including orderid and qty.

USE TSQLV4

DROP FUNCTION IF EXISTS dbo.GetCustOrders;

go

CREATE FUNCTION dbo.GetCustOrders

(

@cid AS INT

)

RETURNS TABLE

AS

RETURN SELECT orderid,

custid,

empid,

orderdate,

requireddate,

shipregion,

shippostalcode,

shipcountry

FROM Sales.Orders

WHERE custid = @cid

GO

SELECT C.custid,

COUNT(DISTINCT ODA.orderid) AS numorders,

SUM(OD.qty) AS totalqty

FROM dbo.GetCustOrders(0) AS C

INNER JOIN Sales.Orders AS O

ON O.custid = C.custid

INNER JOIN Sales.OrderDetails AS OD

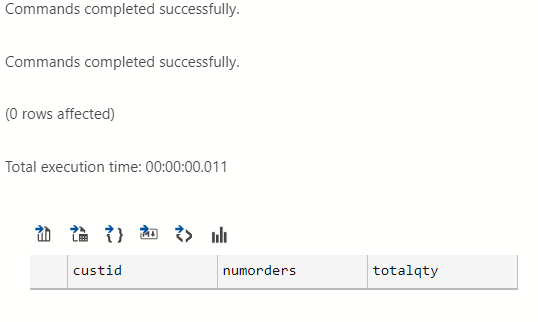
ON OD.orderid = O.orderid

LEFT OUTER JOIN Sales.OrderDetails AS ODA

ON O.orderid = ODA.orderid

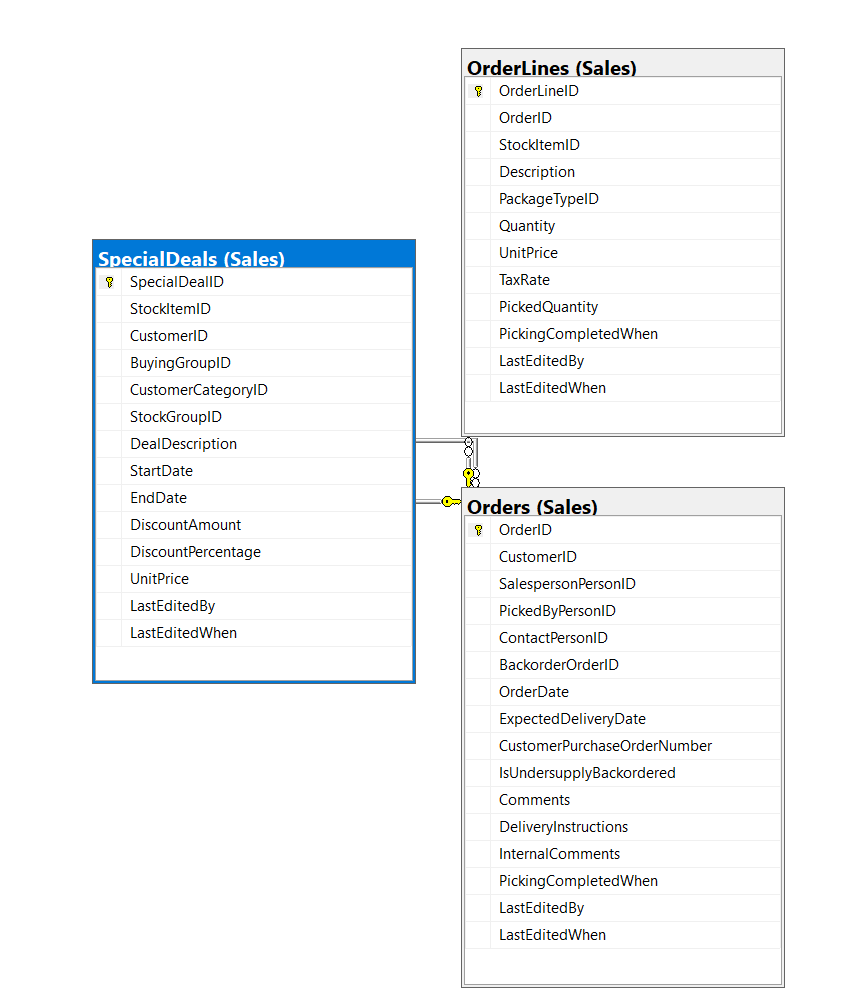
GROUP BY C.custid

FOR JSON PATH, ROOT('zero');



# Medium Queries

## USE: WideWorldImporters Database



| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Retrieve the count of distinct orders placed by each customer, ordered by the ascending CustomerID. This query aims to provide insights into customer order behavior within the Sales database, specifically focusing on the number of orders placed by each customer.
* **Table:** WideWorldImporters database. Sales.Orders: Contains information about orders, including CustomerID and OrderID. Sales.SpecialDeals: Stores special deals information related to customers. Sales.OrderLines: Holds details about order lines, including the order ID.

USE WideWorldImporters

SELECT TOP (10)

O.CustomerID,

COUNT(DISTINCT O.OrderID) AS num\_orders

FROM Sales.Orders AS O

LEFT OUTER JOIN Sales.SpecialDeals AS SP

ON SP.CustomerID = O.CustomerID

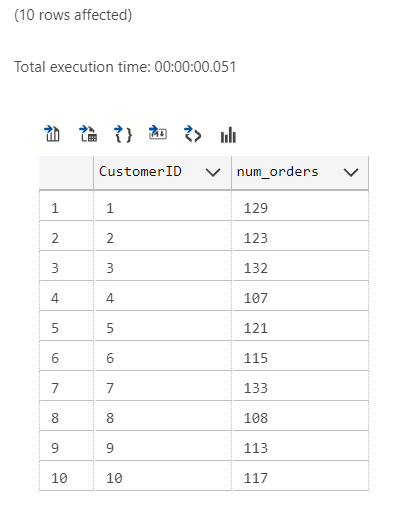
INNER JOIN Sales.OrderLines AS OD

ON O.orderid = OD.orderid

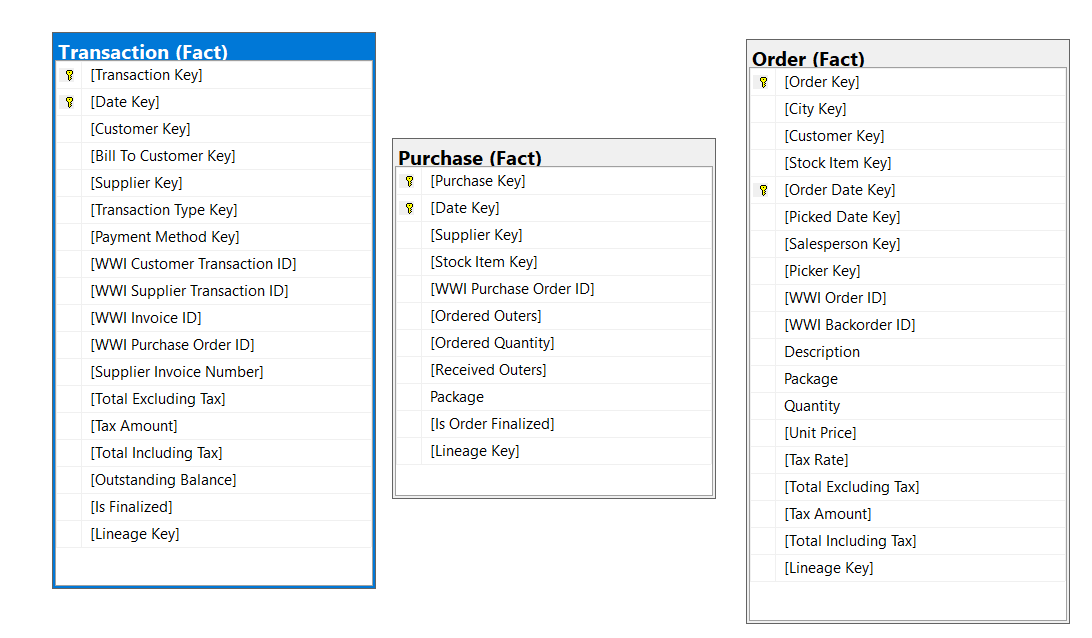
GROUP BY O.CustomerID

ORDER BY O.CustomerID ASC

FOR JSON PATH, ROOT('orders');

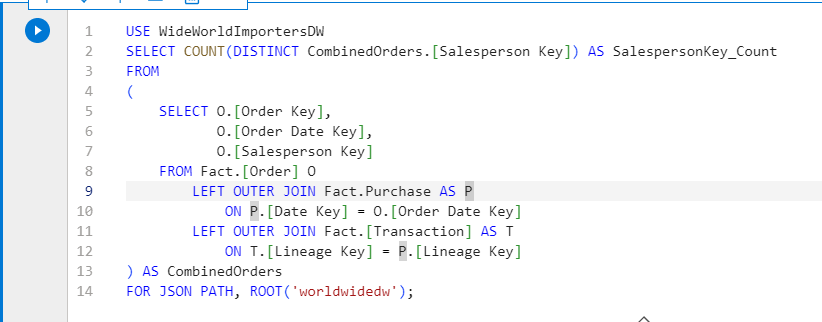


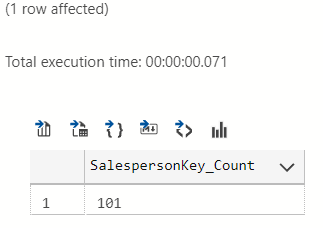
## USE: WideWorldImportersDW Database



| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Count the distinct occurrences of Salesperson Key across orders within the Fact schema. This query aims to provide insights into the diversity of sales personnel involved in transactions recorded in the Fact schema, considering the relationships between Order, Purchase, and Transaction data.
* **Table:** WideWorldImportersDW database. Fact.Order: Contains information about orders, including Order Key, Order Date Key, and Salesperson Key. Fact.Purchase: Stores details about purchases, including Date Key and Lineage Key. Fact.Transaction: Holds information about transactions, including Lineage Key.





## 

## 

## 

## 

## 

## 

## 

## 

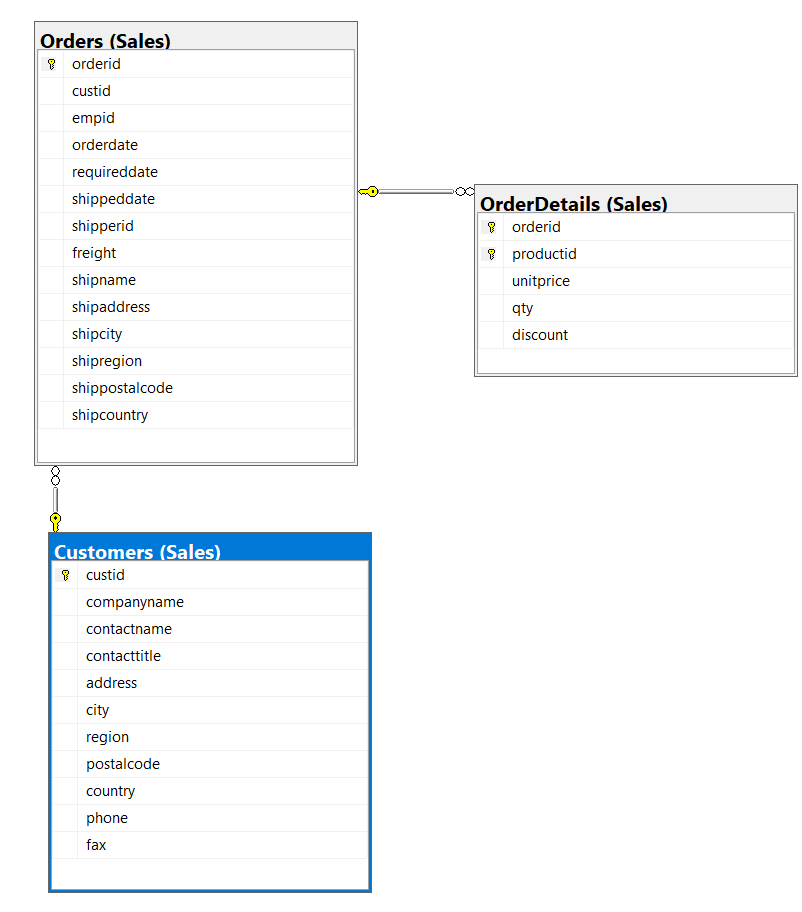
## 

## 

## 

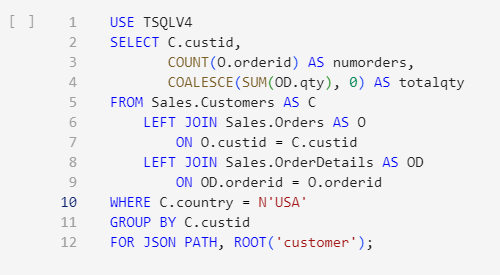
## 

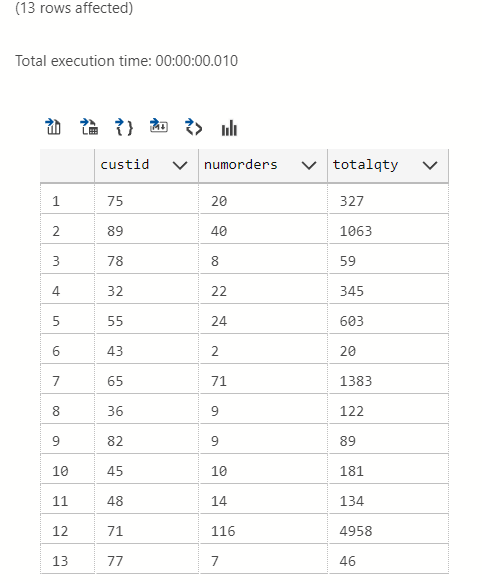
## USE: TSQLV4 Database



| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Retrieve information about customers and their orders, including customers who haven't placed any orders. This query aims to provide insights into customer order behavior within the context of the TSQLV4 database, focusing on the Customers and Orders tables.
* **Table:** Sales.Customers: Contains data about customers, including custid and country. Sales.Orders: Stores information about orders, including orderid and custid. Sales.OrderDetails: Holds details about order items, including qty and orderid.





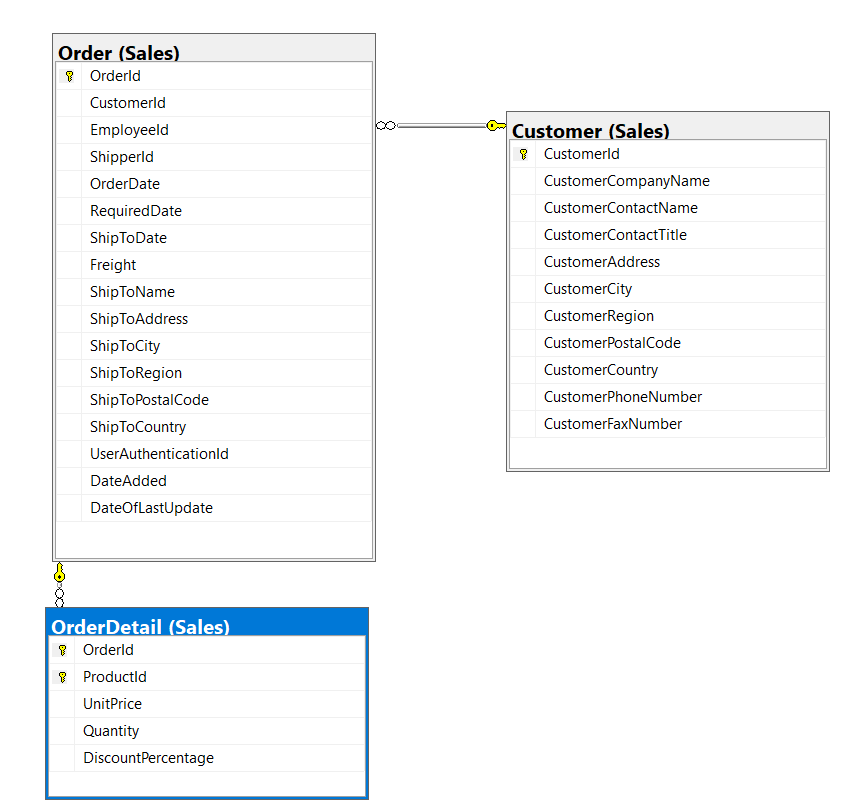
## 

## 

## 

## USE: TSQLV4 Database

## USE: Northwinds2022TSQLV7 Database



| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Retrieve information about customers and their orders, focusing on the Customers and Orders tables. This query aims to provide insights into customer order behavior by calculating the number of orders placed by each customer and the total quantity of items ordered.
* **Table:** Sales.Customer: Contains data about customers, including custid. Sales.Order: Stores information about orders, including orderid and custid. Sales.OrderDetail: Holds details about order items, including qty and orderid

USE Northwinds2022TSQLV7

SELECT C.CustomerId,

COUNT(O.orderid) AS numorders,

SUM(OD.Quantity) AS totalqty

FROM Sales.Customer AS C

INNER JOIN Sales.[Order] AS O

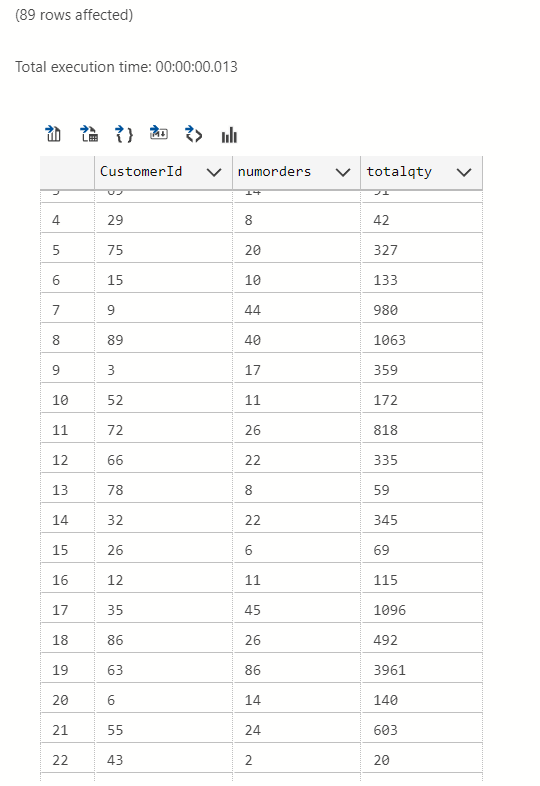
ON O.CustomerId = C.CustomerId

INNER JOIN Sales.OrderDetail AS OD

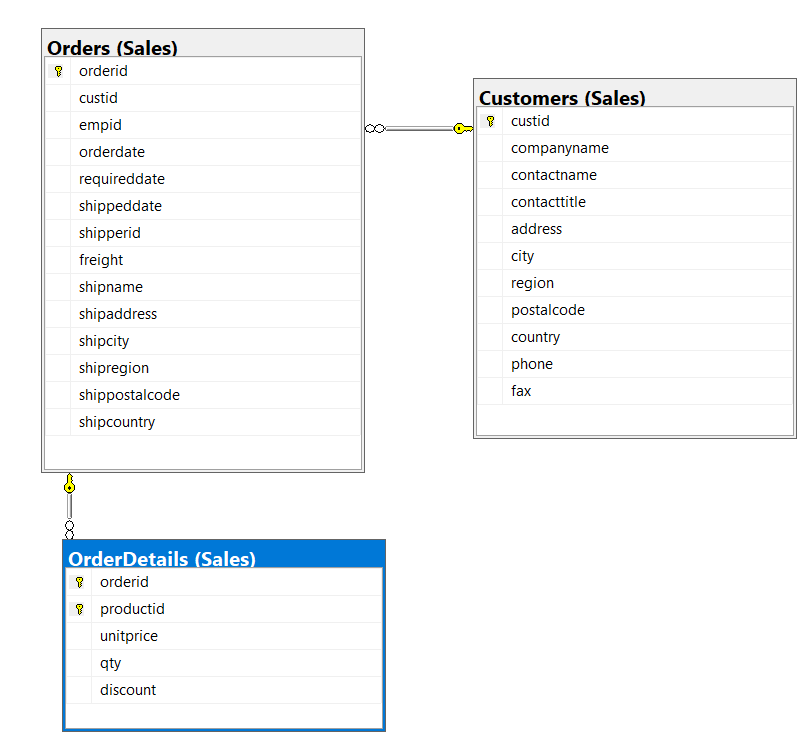
ON OD.orderid = O.orderid

GROUP BY C.CustomerId

FOR JSON PATH, ROOT('customer');

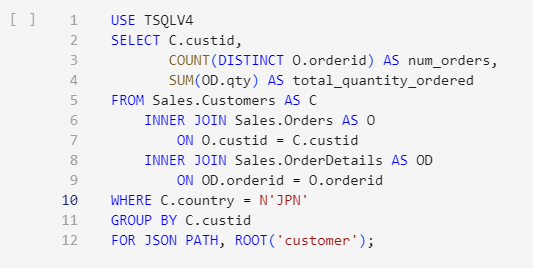


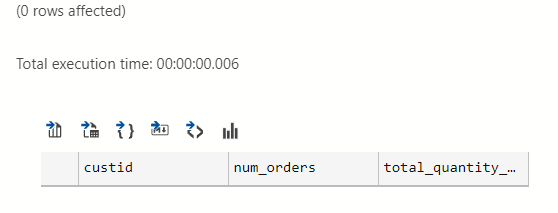
## USE: TSQLV4 Database



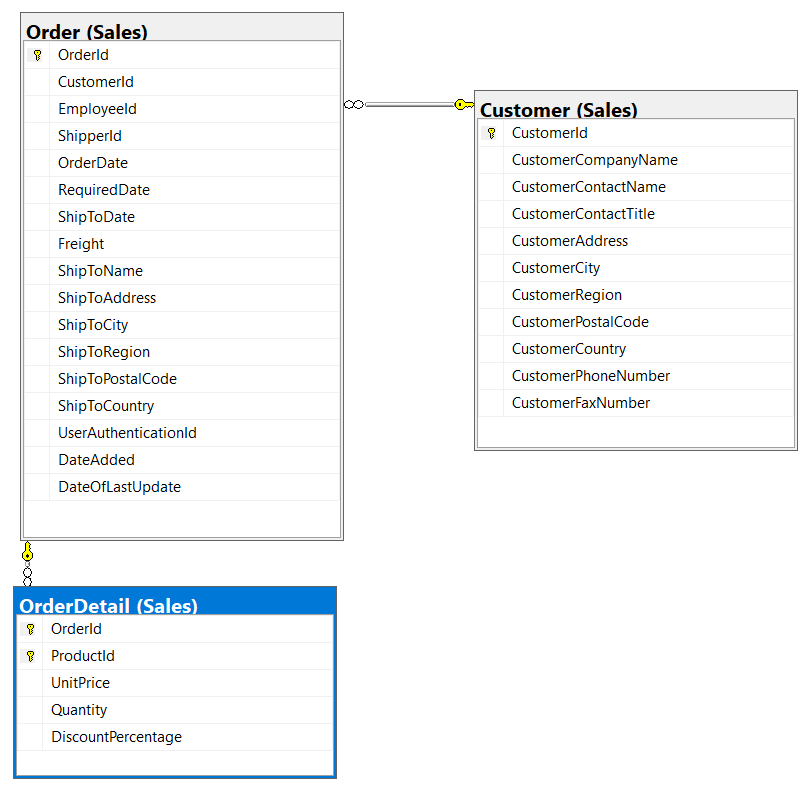
| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Retrieve details of orders made by customers from Japan.
* **Table:** TSQLV4 database. Sales.Customers, Sales.Orders and Sales.Orderdetails table



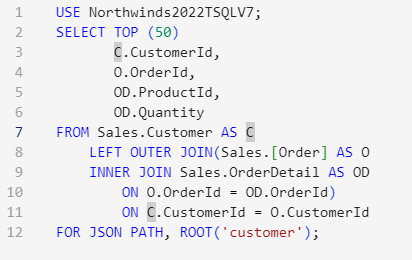


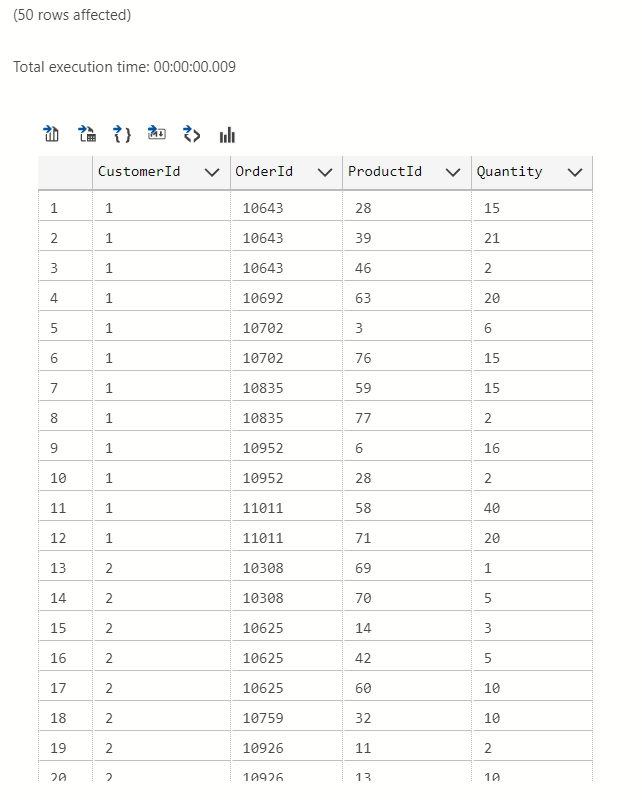
## USE: Northwinds2022TSQLV7 Database



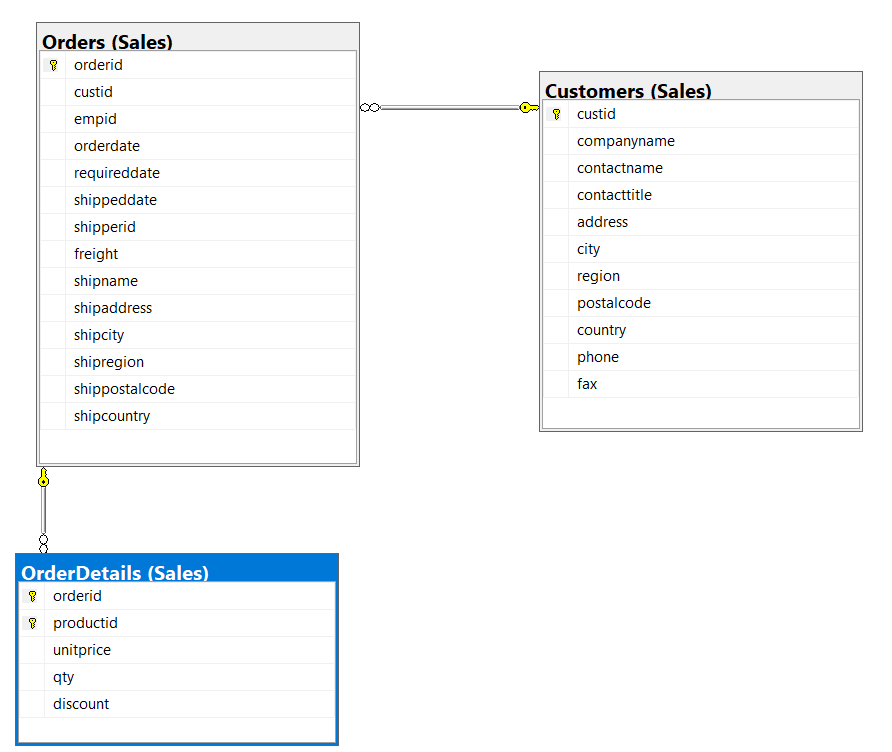
| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Constructs a comprehensive list of customer details along with associated order and order detail information by utilizing a LEFT OUTER JOIN between the Sales.Customer, Sales.[Order], and Sales.OrderDetail tables.
* **Table:** The query involves the Sales.Customer, Sales.[Order], and Sales.OrderDetail tables. Columns: The selected columns include CustomerId from the Sales.Customer table, OrderId from the Sales.[Order] table, and ProductId along with Quantity from the Sales.OrderDetail table.



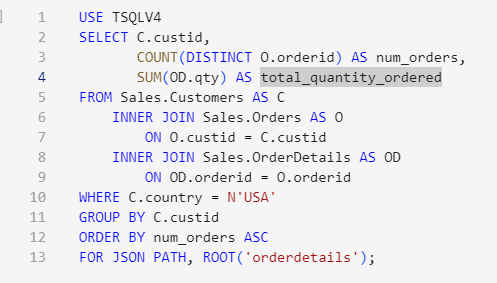


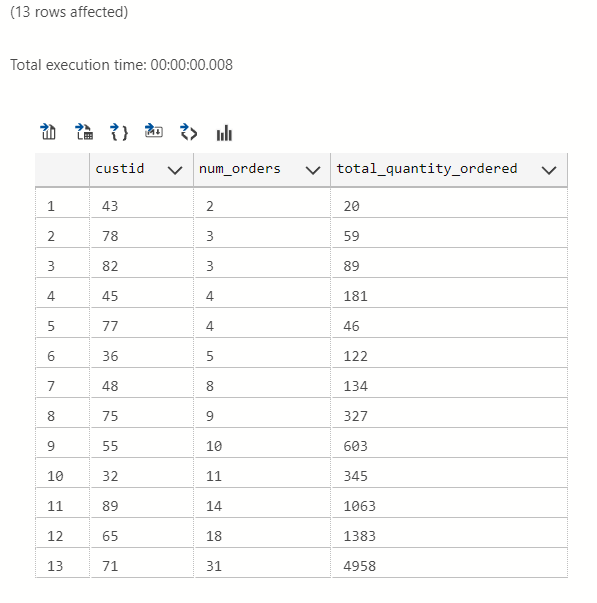
## USE: TSQLV4 Database



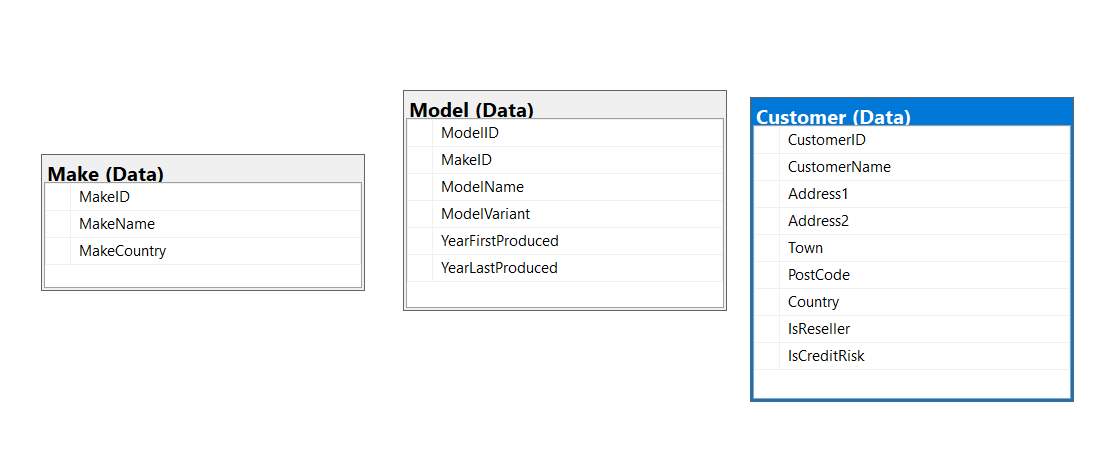
| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Retrieve details of orders made by customers from the USA, ordered by the ascending number of distinct orders placed by each customer. This query aims to analyze the order behavior of US customers within the Sales database.
* **Table:** Sales.Customers: Contains information about customers, including custid and country. Sales.Orders: Stores details about orders, including orderid and custid. Sales.OrderDetails: Holds information about order items, including qty and orderid



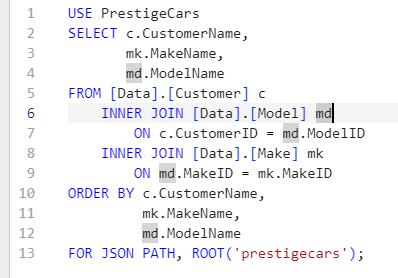


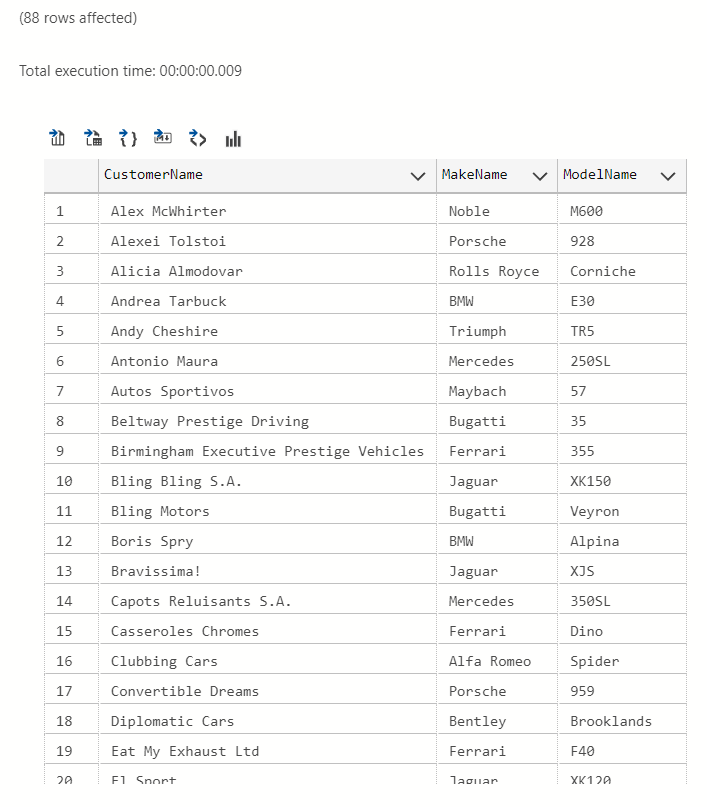
## USE: PrestigeCars Database



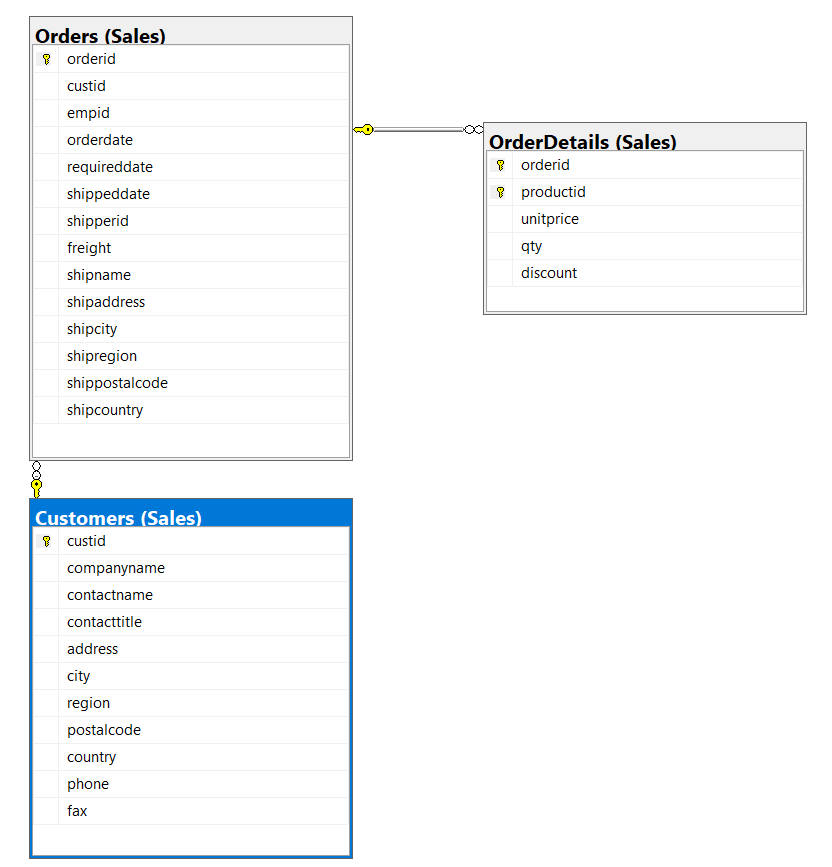
| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Retrieve the details of customers along with the make and model of the cars they own. The results should be sorted by the customer's name, followed by the make of the car, and then the model of the car. This query aims to provide a comprehensive overview of customers and the prestigious cars they own.
* **Table:** Prestige Cars database. Data.Customer table, Data.Model table and Data.Make table



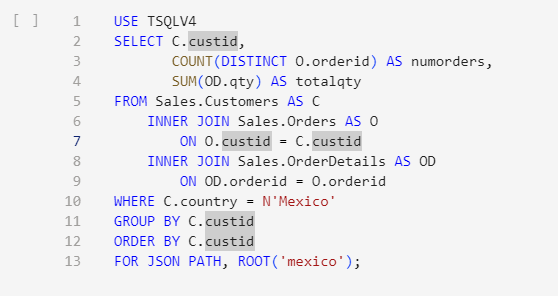


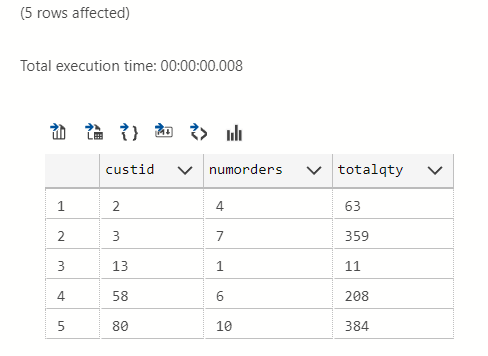
## USE: TSQLV4 Database



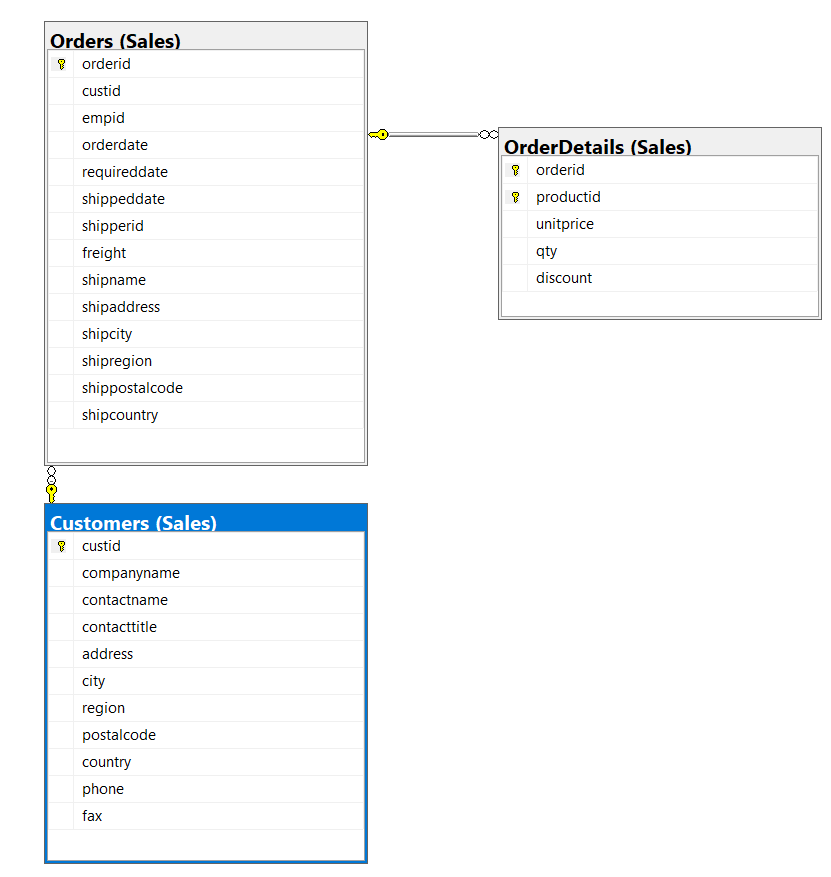
| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Retrieve details of orders made by customers from Mexico and organize the results by customer ID. This query aims to provide insights into the order behavior of Mexican customers within the Sales database.
* **Table:** Sales.Customers: Contains information about customers, including custid and country. Sales.Orders: Stores details about orders, including orderid and custid. Sales.OrderDetails: Holds information about order items, including qty and orderid





## USE: TSQLV4 Database



| Table Name | Column Name |
| --- | --- |
| Person.Person | BusinessEntityID |
| Person.Password | PasswordHash  BusinessEntityID |
| Person.PersonPhone | BusinessEntityID |

* **Proposition:** Retrieve details of orders made by customers from the Canada, ordered by the total quantity of items ordered by each customer. This query aims to analyze the order behavior of Canadian customers within the Sales database, prioritizing customers based on the total quantity of items they've ordered.
* **Table:** Sales.Customers: Contains information about customers, including custid and country. Sales.Orders: Stores details about orders, including orderid and custid. Sales.OrderDetails: Holds information about order items, including qty and orderid

