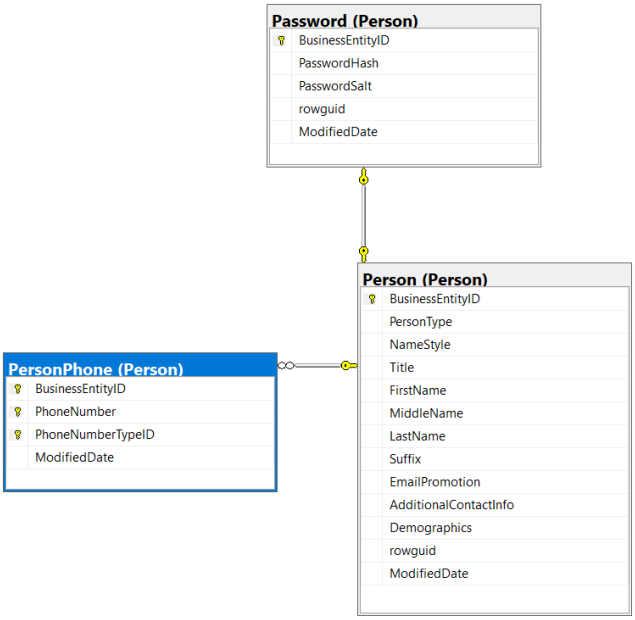
PROJECT ONE

This project aims to utilize database diagrams as key navigation and analysis tools, develop diagram views to separate subsystems for specific information queries, apply diagrammatic subject areas to address real-world business problems, document necessary data for accurate query resolution. Each of the five group member creates 20 queries across six selected databases. Use organized diagram views to facilitate the exploration and solving of complex issues.

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

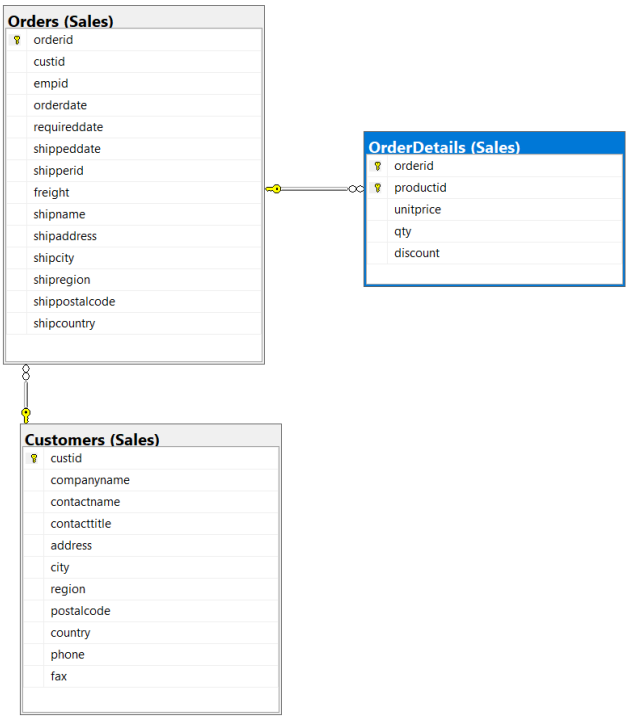
```
USE AdventureWorks2017;
SELECT TOP (10)
       P.BusinessEntityID,
       PS.PasswordHash
FROM Person.Person AS P
    JOIN Person.Password AS PS
        ON P.BusinessEntityID = PS.BusinessEntityID
    INNER JOIN Person.PersonPhone AS PP
        ON PP.BusinessEntityID = PS.BusinessEntityID
```

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

	BusinessEntityID	PasswordHash
1	16496	4hU0mcjBNWobcWZXa4HyMHO7hKbS/1GpPTCsNgWOjRU=
2	12506	keQ5vpcto4CLoVF4vcx0hB9dbx/bNBFJGzUNpCxHt08=
3	11390	cOlBWjtoRPt0VRx0VtLoLrZzVbqo4+DOb8rBRTegPSU=
4	10798	WsRoCsWWTm4y2SsDPGFF2uaMNQqcizj6bYQNVIQn3xo=
5	963	GkzDDuB6pTaET7MyM3Kpgh+sP0JyO6EvZnbS4ep3kEg=
6	12283	gf6FNgr5Poaohtjtgq+4xCW8rjtnI/7Zak1o84A7KyU=
7	3495	xScRCEEWoy6+9hygICfif7yn+xOVfg4V3oUP91YUYoI=
8	4944	F3HwanryATgUwEXvsptgEP946U3agfjapdXbvJ6vwyc=
9	11770	DXSCHCwNULDFXX9c3xrSAsrlC5SSAnCDKqa/cG1ZrPI=
10	17001	PqyzDBTlO5ExE/nId+NCsRtj/xnO/0SI6ryy65hettU=

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.

Carlo Ace Sagad - Project One

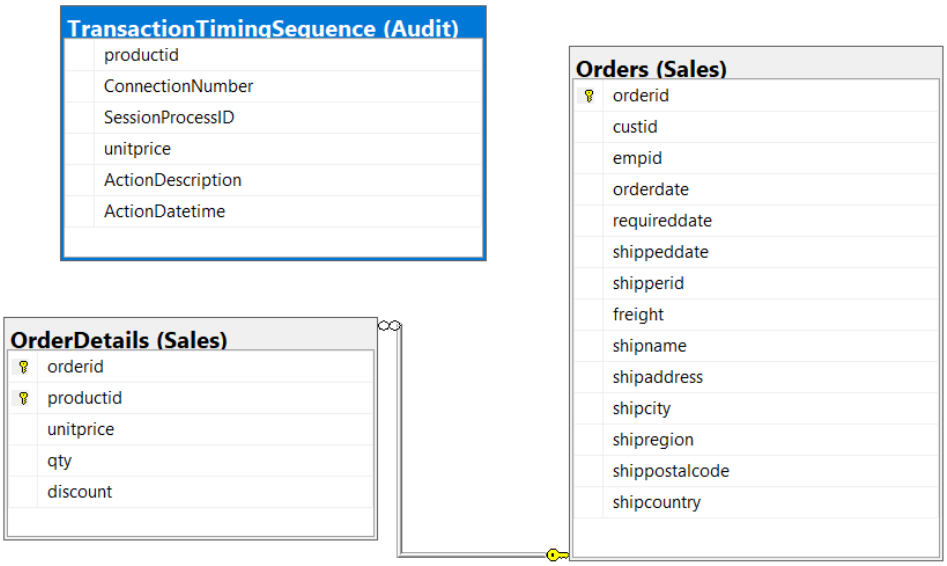
Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

```sql
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');
```

	custid	num_orders	total_quantity_ordered
1	51	13	966
2	10	14	956
3	42	3	62

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

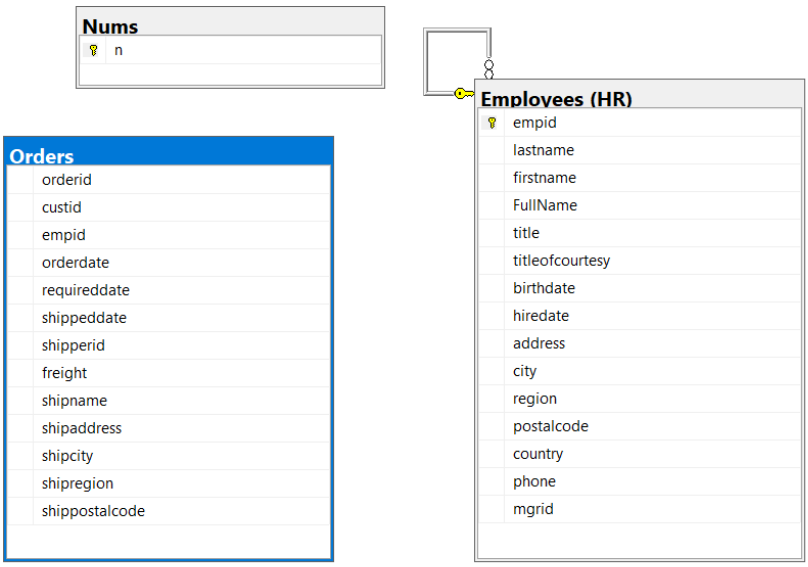
```
USE TSQLV4

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');
```

	custid	numorders	totalqty
1	6	0	980

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
```

Carlo Ace Sagad - Project One

```
        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');
```

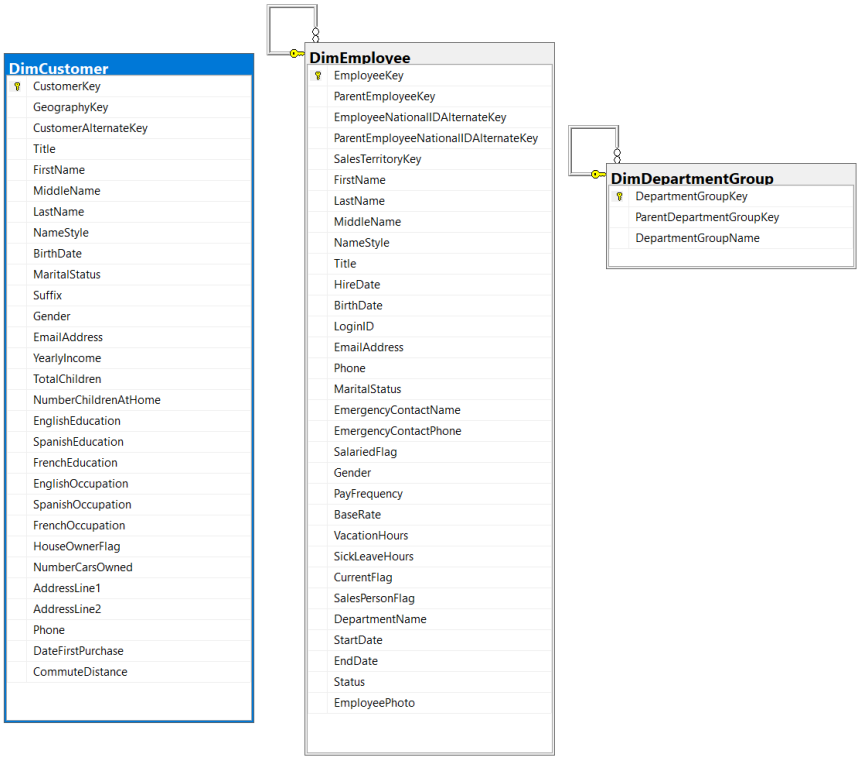
(4150 rows affected)

Total execution time: 00:00:00.025

	empid	firstname	lastname
1	1	Sara	Davis
2	1	Sara	Davis
3	1	Sara	Davis
4	1	Sara	Davis
5	1	Sara	Davis
6	1	Sara	Davis
7	1	Sara	Davis
8	1	Sara	Davis
9	1	Sara	Davis
10	1	Sara	Davis
11	1	Sara	Davis
12	1	Sara	Davis
13	1	Sara	Davis
14	1	Sara	Davis
15	1	Sara	Davis
16	1	Sara	Davis
17	1	Sara	Davis
18	1	Sara	Davis
19	1	Sara	Davis
20	1	Sara	Davis

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');
```

(265 rows affected)

Total execution time: 00:00:00.013

	CustomerKey	FirstName	MiddleName	LastName	BirthDate
1	11147	Ernest	L	Wu	1944-08-04
2	11165	Jocelyn	NULL	Alexander	1979-01-15
3	11214	Charles	O	Miller	1955-05-07
4	11248	Tristan	P	Alexander	1966-03-22
5	11333	Emily	R	Miller	1968-01-07
6	11368	Edward	NULL	Miller	1984-05-24
7	11479	Darryl	L	Wu	1978-10-20
8	11493	Dawn	T	Wu	1976-05-31
9	11503	Dennis	G	Wu	1936-04-11
10	11532	Lauren	NULL	Miller	1962-08-26
11	11647	Randy	J	Wu	1975-03-22
12	11721	Jennifer	A	Alexander	1971-01-18
13	11738	Elijah	NULL	Alexander	1970-10-05
14	11780	Jessica	K	Alexander	1977-05-19
15	11817	Morgan	C	Miller	1981-11-23
16	11837	Haley	NULL	Alexander	1976-08-05
17	11938	Seth	D	Alexander	1983-03-21
18	11940	Tyler	E	Miller	1983-01-09
19	12083	Kayla	NULL	Alexander	1951-08-28
20	12172	Eduardo	F	Alexander	1972-03-01

WORST QUERY (3)

USE: TSQLV4 Database

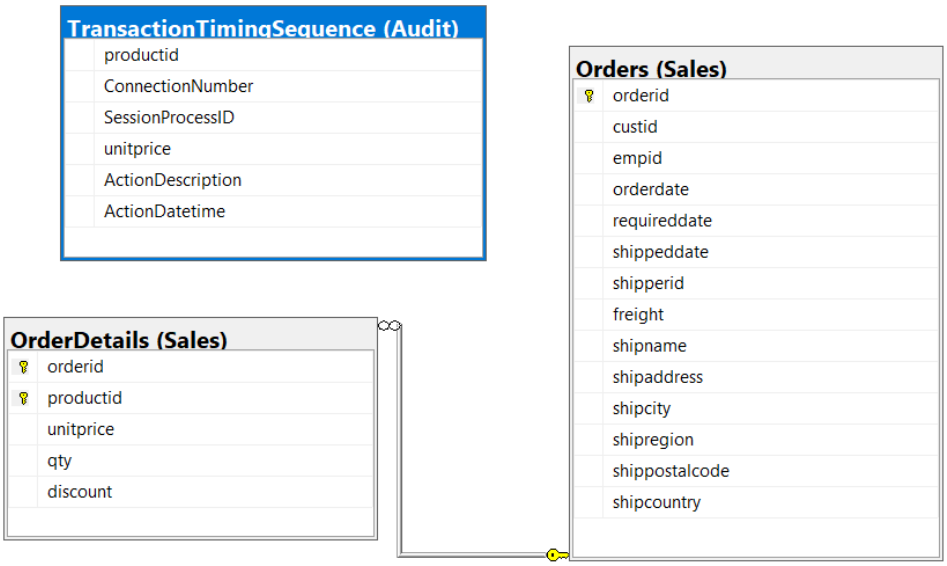


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

- **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

)
```

Carlo Ace Sagad - Project One

```
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');
```

Commands completed successfully.

Commands completed successfully.

(0 rows affected)

Total execution time: 00:00:00.011

	custid	numorders	totalqty

Medium Queries

USE: WideWorldImporters Database

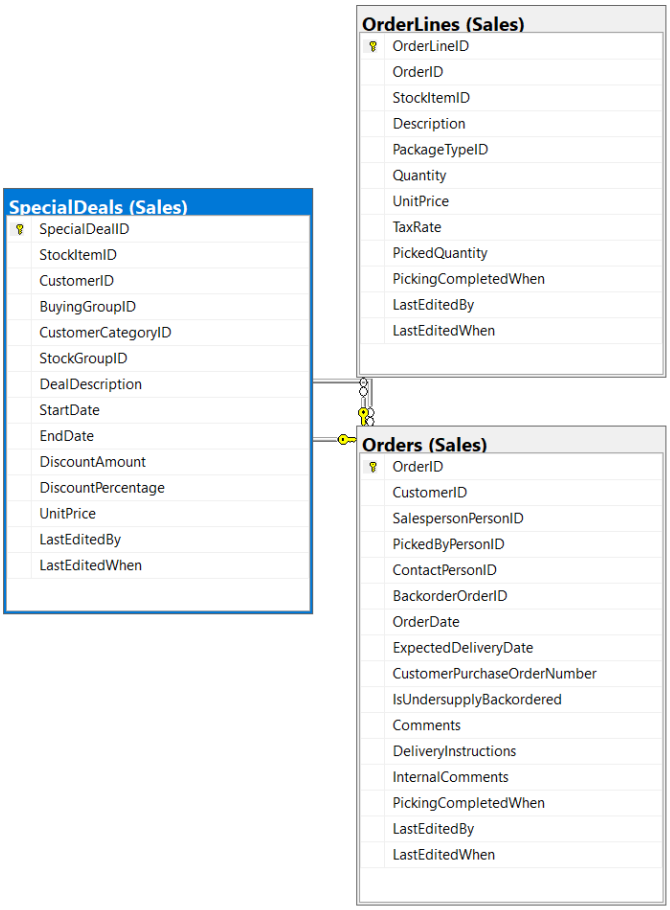


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

- **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');
```

(10 rows affected)

Total execution time: 00:00:00.051

	CustomerID	num_orders
1	1	129
2	2	123
3	3	132
4	4	107
5	5	121
6	6	115
7	7	133
8	8	108
9	9	113
10	10	117

Carlo Ace Sagad - Project One

USE: WideWorldImportersDW Database

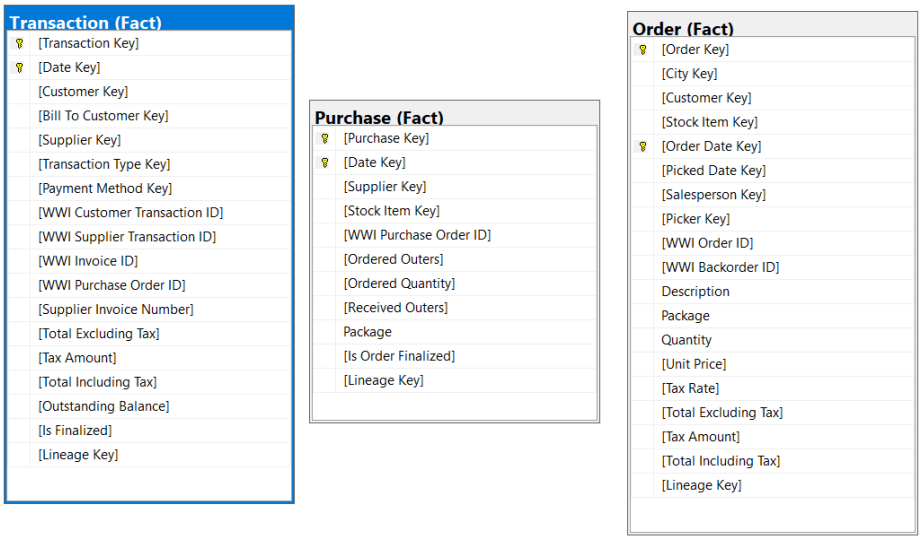


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

- **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.

Carlo Ace Sagad - Project One

```sql
USE WideWorldImportersDW
SELECT COUNT(DISTINCT CombinedOrders.[Salesperson Key]) AS SalespersonKey_Count
FROM
(
    SELECT O.[Order Key],
           O.[Order Date Key],
           O.[Salesperson Key]
    FROM Fact.[Order] O
        LEFT OUTER JOIN Fact.Purchase AS P
            ON P.[Date Key] = O.[Order Date Key]
        LEFT OUTER JOIN Fact.[Transaction] AS T
            ON T.[Lineage Key] = P.[Lineage Key]
) AS CombinedOrders
FOR JSON PATH, ROOT('worldwidedw');
```

(1 row affected)

Total execution time: 00:00:00.071

	SalespersonKey_Count
1	101

USE: TSQLV4 Database

Carlo Ace Sagad - Project One

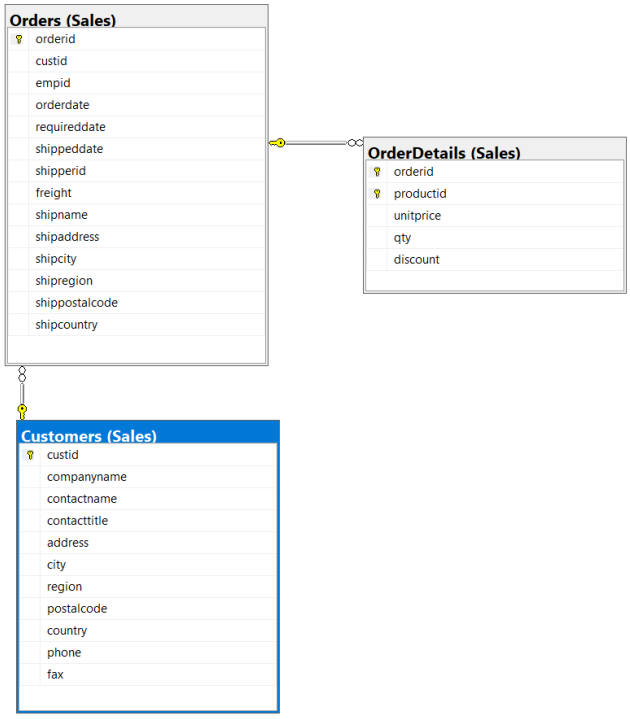


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

- **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.

Carlo Ace Sagad - Project One

```sql
USE TSQLV4
SELECT C.custid,
       COUNT(O.orderid) AS numorders,
       COALESCE(SUM(OD.qty), 0) AS totalqty
FROM Sales.Customers AS C
    LEFT JOIN Sales.Orders AS O
        ON O.custid = C.custid
    LEFT JOIN Sales.OrderDetails AS OD
        ON OD.orderid = O.orderid
WHERE C.country = N'USA'
GROUP BY C.custid
FOR JSON PATH, ROOT('customer');
```

(13 rows affected)

Total execution time: 00:00:00.010

	custid	numorders	totalqty
1	75	20	327
2	89	40	1063
3	78	8	59
4	32	22	345
5	55	24	603
6	43	2	20
7	65	71	1383
8	36	9	122
9	82	9	89
10	45	10	181
11	48	14	134
12	71	116	4958
13	77	7	46

USE: TSQLV4 Database

Carlo Ace Sagad - Project One

USE: Northwinds2022TSQLV7 Database

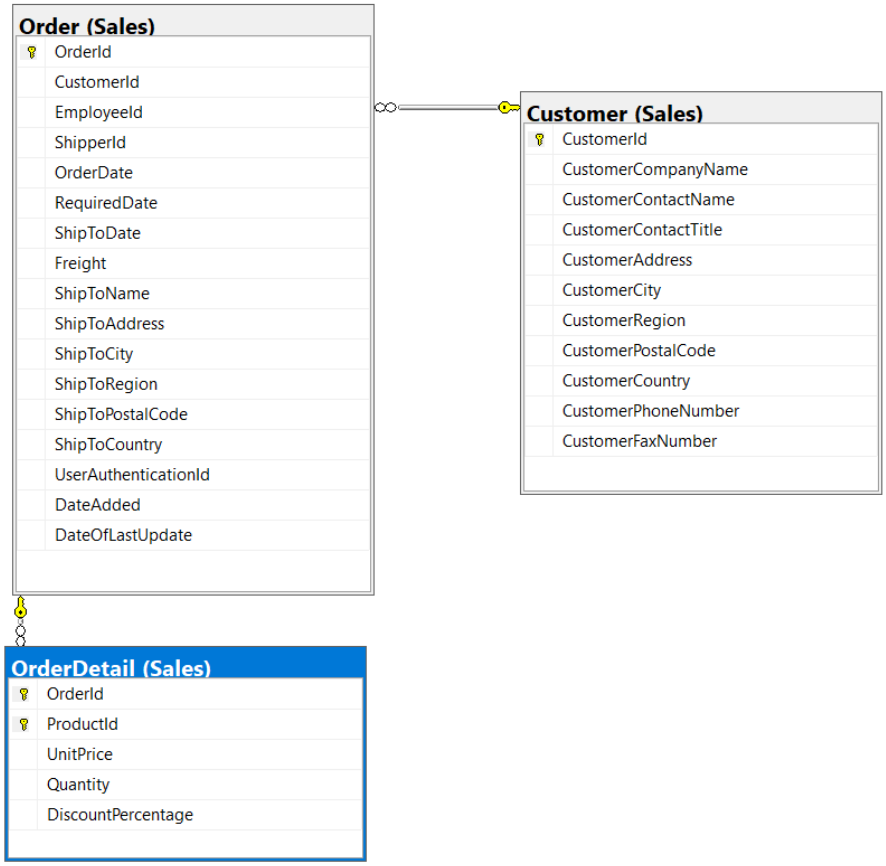


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

- **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
```

Carlo Ace Sagad - Project One

```
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');
```

(89 rows affected)

Total execution time: 00:00:00.013

	CustomerId	numorders	totalqty
3	89	14	91
4	29	8	42
5	75	20	327
6	15	10	133
7	9	44	980
8	89	40	1063
9	3	17	359
10	52	11	172
11	72	26	818
12	66	22	335
13	78	8	59
14	32	22	345
15	26	6	69
16	12	11	115
17	35	45	1096
18	86	26	492
19	63	86	3961
20	6	14	140
21	55	24	603
22	43	2	20

Carlo Ace Sagad - Project One

USE: TSQLV4 Database

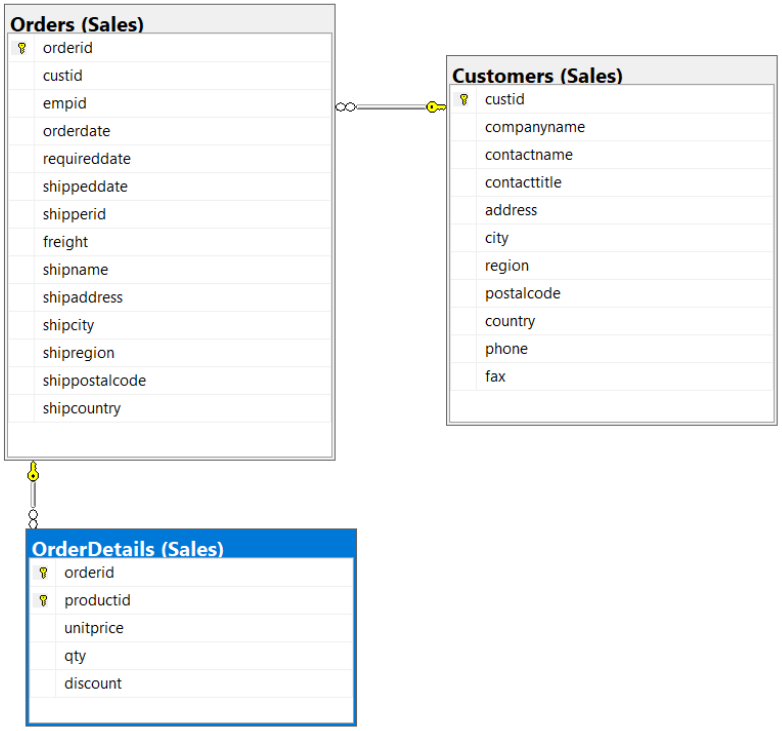


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

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

Carlo Ace Sagad - Project One

```sql
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'JPN'
GROUP BY C.custid
FOR JSON PATH, ROOT('customer');
```

(0 rows affected)

Total execution time: 00:00:00.006

	custid	num_orders	total_quantity_…

USE: Northwinds2022TSQLV7 Database

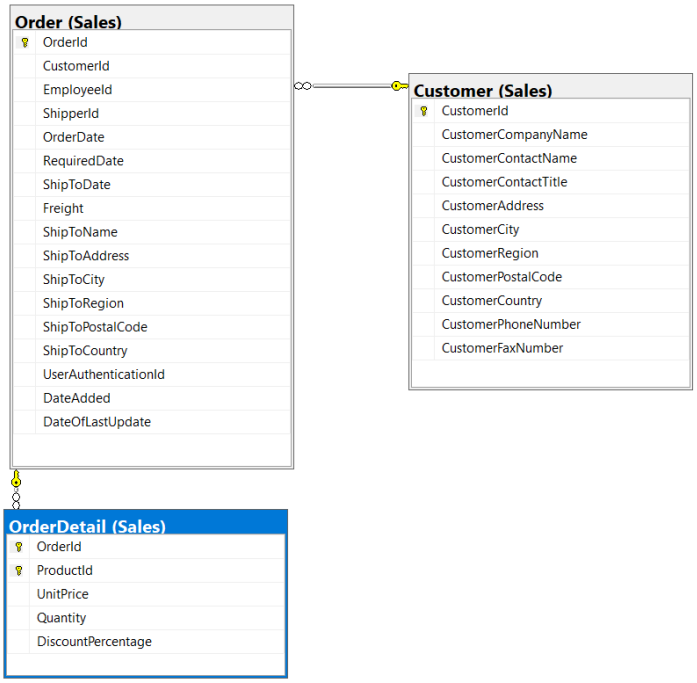


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

- **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.

```sql
USE Northwinds2022TSQLV7;
SELECT TOP (50)
       C.CustomerId,
       O.OrderId,
       OD.ProductId,
       OD.Quantity
FROM Sales.Customer AS C
    LEFT OUTER JOIN(Sales.[Order] AS O
    INNER JOIN Sales.OrderDetail AS OD
        ON O.OrderId = OD.OrderId)
        ON C.CustomerId = O.CustomerId
FOR JSON PATH, ROOT('customer');
```

Carlo Ace Sagad - Project One

(50 rows affected)

Total execution time: 00:00:00.009

	CustomerId	OrderId	ProductId	Quantity
1	1	10643	28	15
2	1	10643	39	21
3	1	10643	46	2
4	1	10692	63	20
5	1	10702	3	6
6	1	10702	76	15
7	1	10835	59	15
8	1	10835	77	2
9	1	10952	6	16
10	1	10952	28	2
11	1	11011	58	40
12	1	11011	71	20
13	2	10308	69	1
14	2	10308	70	5
15	2	10625	14	3
16	2	10625	42	5
17	2	10625	60	10
18	2	10759	32	10
19	2	10926	11	2
20	2	10926	13	10

Carlo Ace Sagad - Project One

USE: TSQLV4 Database

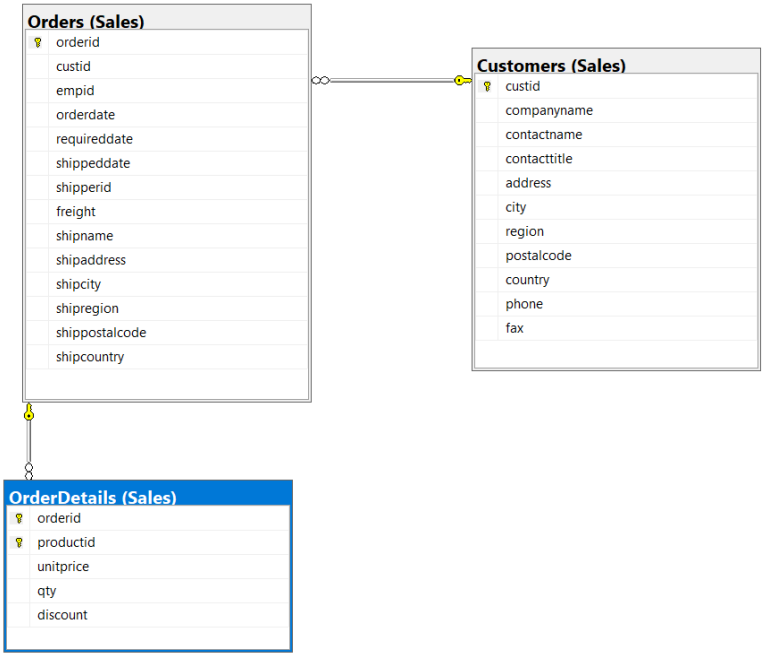


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

- **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

Carlo Ace Sagad - Project One

```sql
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'USA'
GROUP BY C.custid
ORDER BY num_orders ASC
FOR JSON PATH, ROOT('orderdetails');
```

(13 rows affected)

Total execution time: 00:00:00.008

	custid	num_orders	total_quantity_ordered
1	43	2	20
2	78	3	59
3	82	3	89
4	45	4	181
5	77	4	46
6	36	5	122
7	48	8	134
8	75	9	327
9	55	10	603
10	32	11	345
11	89	14	1063
12	65	18	1383
13	71	31	4958

Carlo Ace Sagad - Project One

USE: PrestigeCars Database

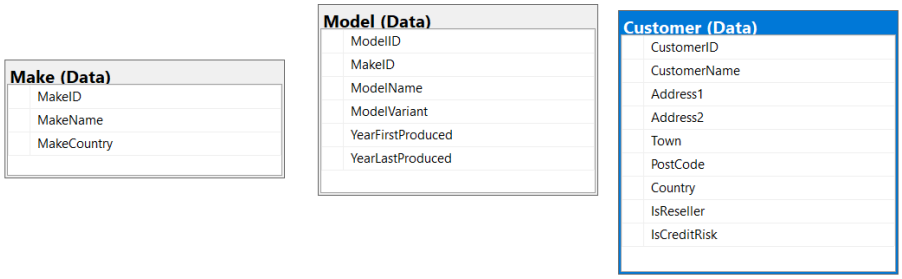


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

- **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

```sql
USE PrestigeCars
SELECT c.CustomerName,
       mk.MakeName,
       md.ModelName
FROM [Data].[Customer] c
    INNER JOIN [Data].[Model] md
        ON c.CustomerID = md.ModelID
    INNER JOIN [Data].[Make] mk
        ON md.MakeID = mk.MakeID
ORDER BY c.CustomerName,
         mk.MakeName,
         md.ModelName
FOR JSON PATH, ROOT('prestigecars');
```

Carlo Ace Sagad - Project One

(88 rows affected)

Total execution time: 00:00:00.009

	CustomerName	MakeName	ModelName
1	Alex McWhirter	Noble	M600
2	Alexei Tolstoi	Porsche	928
3	Alicia Almodovar	Rolls Royce	Corniche
4	Andrea Tarbuck	BMW	E30
5	Andy Cheshire	Triumph	TR5
6	Antonio Maura	Mercedes	250SL
7	Autos Sportivos	Maybach	57
8	Beltway Prestige Driving	Bugatti	35
9	Birmingham Executive Prestige Vehicles	Ferrari	355
10	Bling Bling S.A.	Jaguar	XK150
11	Bling Motors	Bugatti	Veyron
12	Boris Spry	BMW	Alpina
13	Bravissima!	Jaguar	XJS
14	Capots Reluisants S.A.	Mercedes	350SL
15	Casseroles Chromes	Ferrari	Dino
16	Clubbing Cars	Alfa Romeo	Spider
17	Convertible Dreams	Porsche	959
18	Diplomatic Cars	Bentley	Brooklands
19	Eat My Exhaust Ltd	Ferrari	F40
20	El Sport	Jaguar	XK120

Carlo Ace Sagad - Project One

USE: TSQLV4 Database

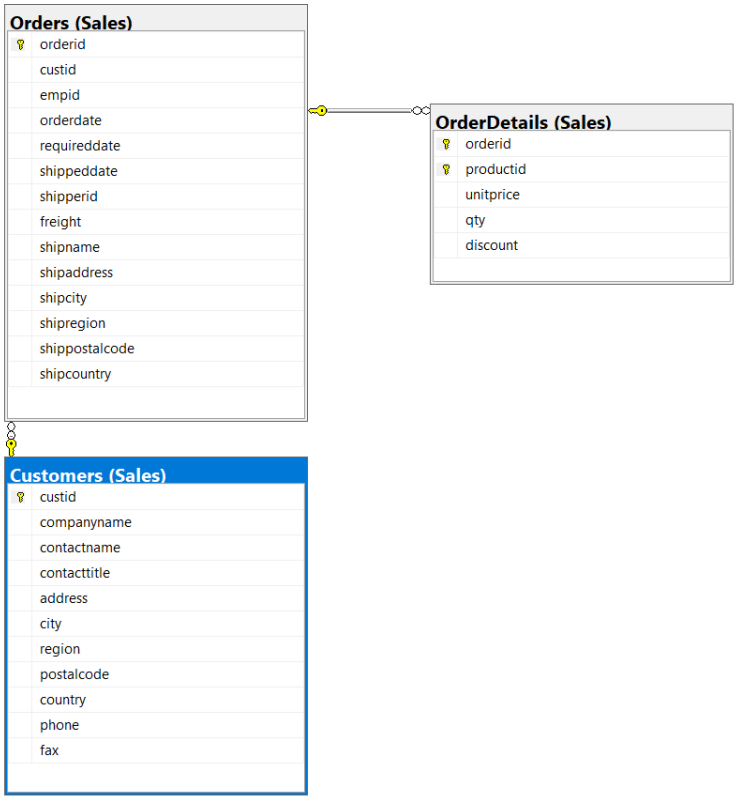


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

- **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

Carlo Ace Sagad - Project One

```sql
USE TSQLV4
SELECT C.custid,
       COUNT(DISTINCT O.orderid) AS numorders,
       SUM(OD.qty) AS totalqty
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'Mexico'
GROUP BY C.custid
ORDER BY C.custid
FOR JSON PATH, ROOT('mexico');
```

(5 rows affected)

Total execution time: 00:00:00.008

	custid	numorders	totalqty
1	2	4	63
2	3	7	359
3	13	1	11
4	58	6	208
5	80	10	384

Carlo Ace Sagad - Project One

USE: TSQLV4 Database

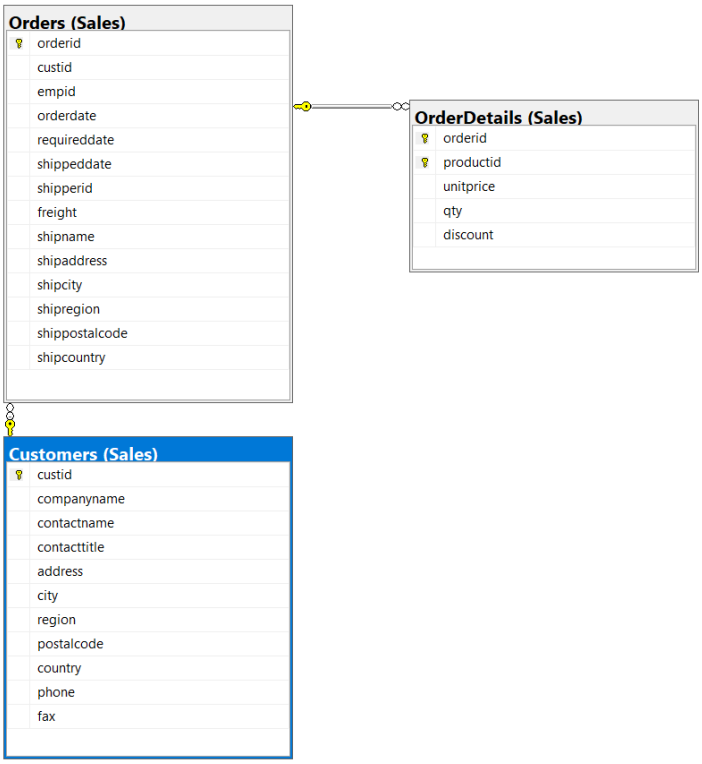


Table Name	Column Name
Person.Person	BusinessEntityID
Person.Password	PasswordHash BusinessEntityID
Person.PersonPhone	BusinessEntityID

- **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

Carlo Ace Sagad - Project One

```sql
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');
```