Project 1 Propositions

9 queries from each person:
(3) worst (3) best (3) improved

OCT 2021

ISSUED BY
10:45AM Group 4

REPRESENTATIVE
Saqib M

Table of Contents

Proposition 1 (Best Simple)

Proposition 1: This query's goal is to find out the cities that Employee 6 has packed to. This resulted in 40 different cities being delivered packages packed by Employee 6.

Model Diagrams:

Figure 1A: Key View Model for Proposition 1

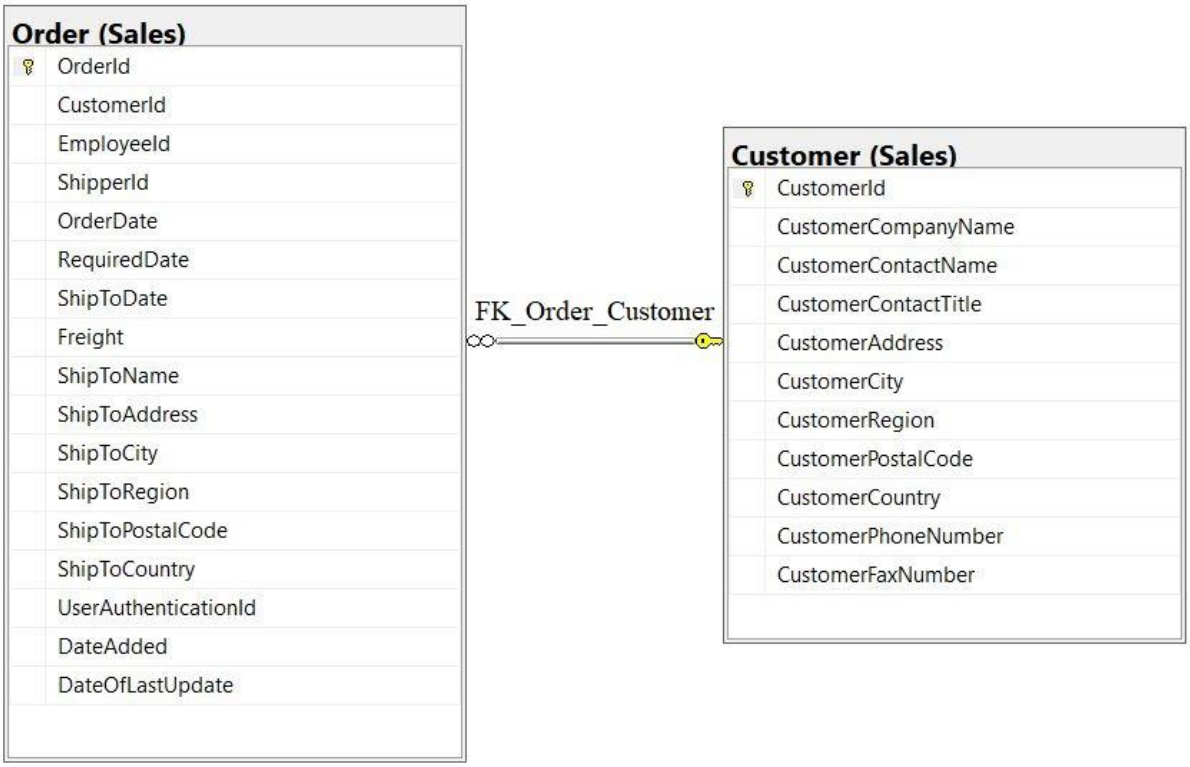
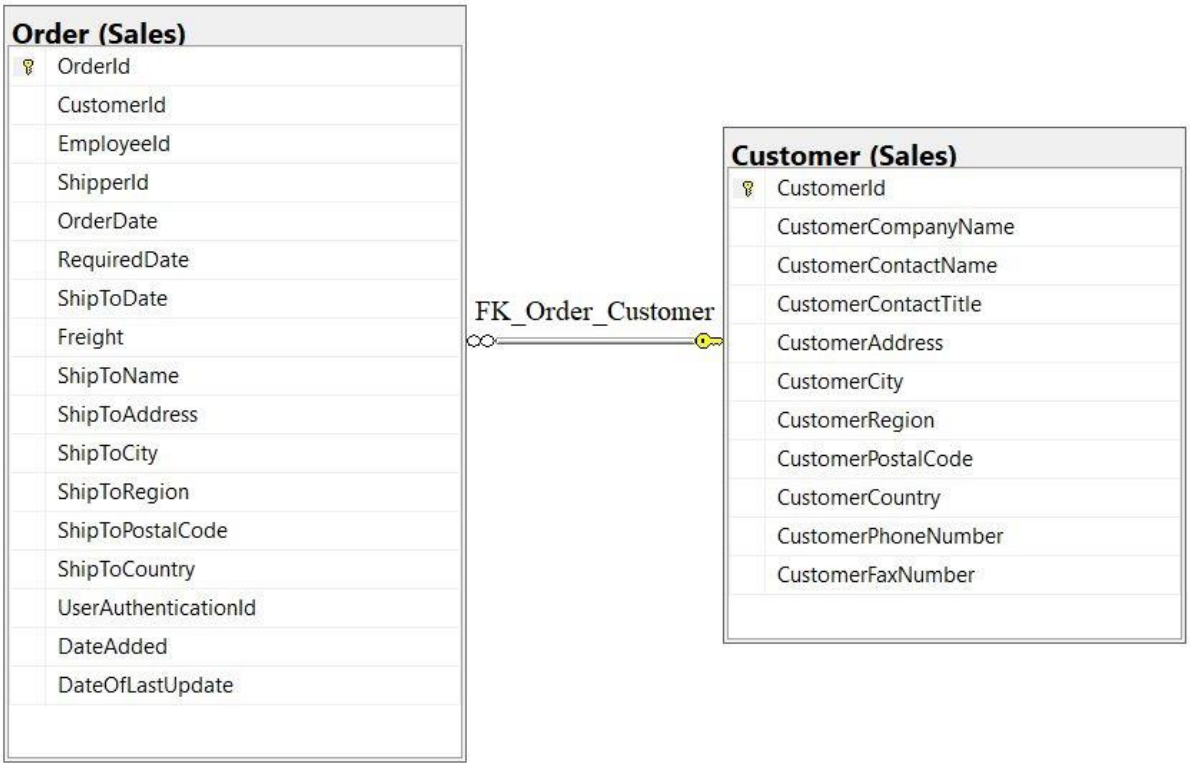


Figure 1B: Standard View Model for Proposition 1



Explanation:

Uses one inner join to combine the CustomerId columns from the tables Order.Sales and Customer.Sales, this was efficient and a straightforward approach to solving the proposition.

Figure 1C: Tables for SQL query components

Select clause

Table name:	Column name:
Sales.Order	CustomerId EmployeeId
Sales.Customer	CustomerId EmployeeId

Order by (optional, only if exist)

Table name	Column name	Sort order
Example table	Example column	asc/desc

Query:

All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

Figure 1D: Formatted SQL Query for Proposition 1

```
USE Northwinds2020TSQLV6

SELECT DISTINCT EmployeeId
        ,CustomerCity
FROM Sales.[Order]
INNER JOIN Sales.Customer ON [Order].CustomerId = Customer.CustomerId
WHERE EmployeeId = 6
ORDER BY CustomerCity
```

Figure 1E: Query Output for Proposition 1

Results | Messages

	EmployeeId	CustomerCity
1	6	Albuquerque
2	6	Anchorage
3	6	Århus
4	6	Barquisimeto
5	6	Bergamo
6	6	Berlin
7	6	Bern
8	6	Boise
9	6	Bräcke
10	6	Buenos Aires
11	6	Campinas
12	6	Charleroi
13	6	Cork
14	6	Cowes
15	6	Eugene
16	6	Graz
17	6	Kirkland
18	6	Köln

Query executed successfully. | 10.0.0.16,12001 (15.0 RTM) | sa (84) | Northwinds2020TSQLV6 | 00:00:00 | 40 rows

JSON:

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

Figure 1F: Formatted SQL Query with JSON for Proposition 1

```sql
USE Northwinds2020TSQLV6

SELECT EmployeeId
        ,CustomerCity
FROM Sales.[Order]
INNER JOIN Sales.Customer ON [Order].CustomerId = Customer.CustomerId
WHERE EmployeeId = 6
ORDER BY CustomerCity
FOR json path
        ,root('Employee6')
        ,include_null_values;
```

Figure 1G: Formatted JSON Output for Proposition 1


```
{
   "Employee6":[
      {
         "EmployeeId":6,
         "CustomerCity":"Albuquerque"
      },
      {
         "EmployeeId":6,
         "CustomerCity":"Anchorage"
      },
      {
         "EmployeeId":6,
         "CustomerCity":"Århus"
      },
      {
         "EmployeeId":6,
         "CustomerCity":"Barquisimeto"
      },
      {
         "EmployeeId":6,
         "CustomerCity":"Bergamo"
      },
      {
         "EmployeeId":6,
         "CustomerCity":"Berlin"
      },
      {
         "EmployeeId":6,
         "CustomerCity":"Bern"
      },
      {
         "EmployeeId":6,
         "CustomerCity":"Boise"
      },
      {
         "EmployeeId":6,
         "CustomerCity":"Bräcke"
      },
      {
         "EmployeeId":6,
         "CustomerCity":"Buenos Aires"
      },
```

Proposition 2 (Best Medium)

Proposition 2: Show me the total order amount and amount spent from January 1 2012 to January 1 2013

Model Diagrams:

Figure 2A: Key View Model for Proposition 2

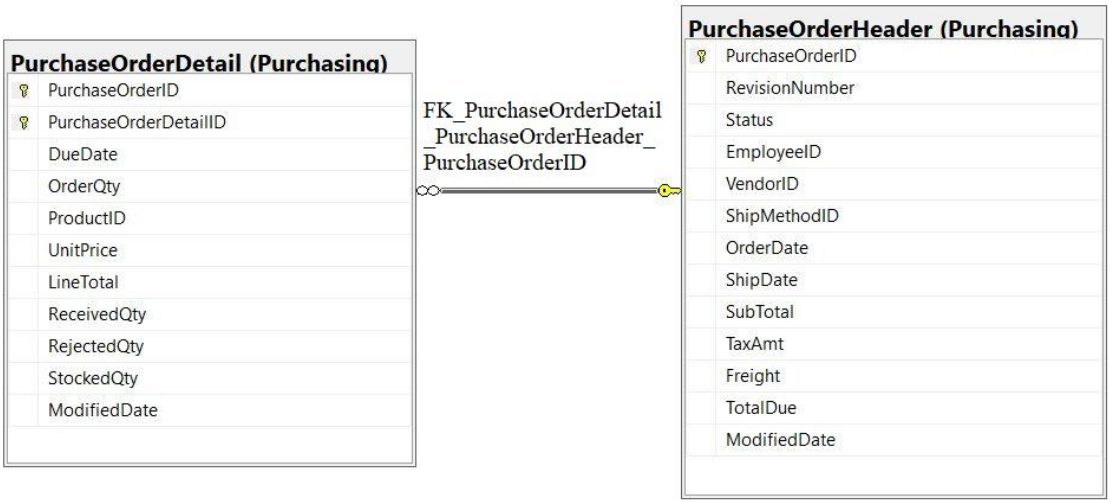
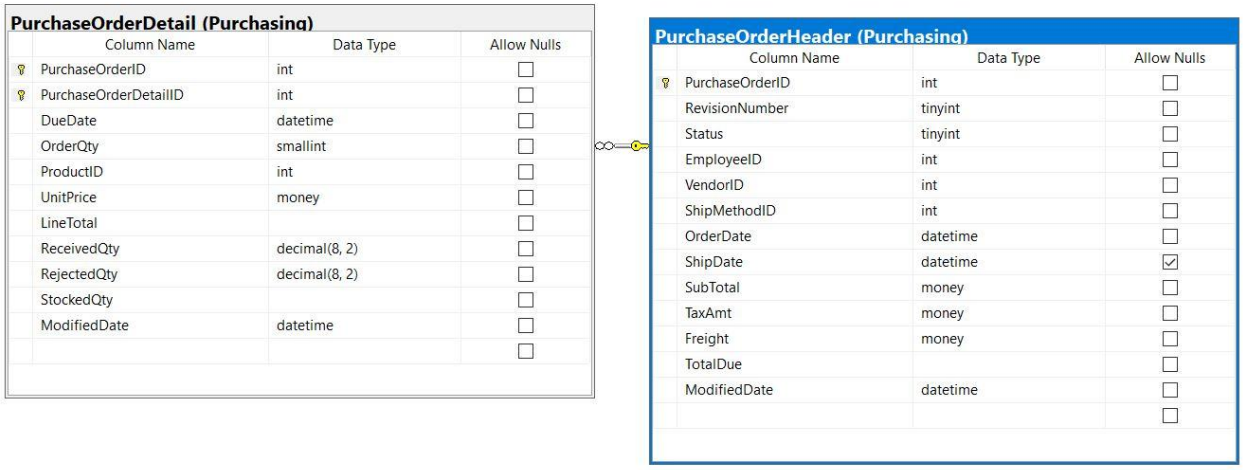


Figure 2B: Standard View Model for Proposition 2



PurchaseOrderDetail (Purchasing)

Column Name	Data Type	Allow Nulls
PurchaseOrderID	int	☐
PurchaseOrderDetailID	int	☐
DueDate	datetime	☐
OrderQty	smallint	☐
ProductID	int	☐
UnitPrice	money	☐
LineTotal		☐
ReceivedQty	decimal(8, 2)	☐
RejectedQty	decimal(8, 2)	☐
StockedQty		☐
ModifiedDate	datetime	☐
		☐

PurchaseOrderHeader (Purchasing)

Column Name	Data Type	Allow Nulls
PurchaseOrderID	int	☐
RevisionNumber	tinyint	☐
Status	tinyint	☐
EmployeeID	int	☐
VendorID	int	☐
ShipMethodID	int	☐
OrderDate	datetime	☐
ShipDate	datetime	☑
SubTotal	money	☐
TaxAmt	money	☐
Freight	money	☐
TotalDue		☐
ModifiedDate	datetime	☐
		☐

Explanation:
This table uses one inner join along with two CTEs to help create a total order amount along with the total spent between January 1st 2012 through to January 1st 2013. This is done through pinpointing columns PurchaseOrderID, LineTotal, and OrderQty from the table Purchasing.PurchaseOrderDetail. We also used PurchaseOrderID from the table Purchasing.PurchaseOrderHeader.

Figure 2C: Tables for SQL query components

Select clause

Table name:	Column name:
Purchasing.PurchaseOrderDetail	PurchaseOrderID LineTotal OrderQty
Purchasing.PurchaseOrderHeader	PurchaseOrderID

Order by (optional, only if exist)

Table name	Column name	Sort order
Example table	Example column	asc/desc

Query:
All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

Figure 2D: Formatted SQL Query for Proposition 2

```sql
USE AdventureWorks2017;

WITH OrderDetail (
        OrderID
        ,OrderDetailQty
        ,OrderDetailAmt
        )
AS (
        SELECT PurchaseOrderID
                ,OrderQty
                ,LineTotal
        FROM Purchasing.PurchaseOrderDetail
        )
        ,Orders (
        OrderID
        ,OrderDate
        )
AS (
        SELECT PurchaseOrderID
                ,OrderDate
        FROM Purchasing.PurchaseOrderHeader h
        WHERE OrderDate > '2012 - 1- 1'
                AND OrderDate < '2013 - 1 - 1'
        )
SELECT o.OrderDate
        ,SUM(od.OrderDetailQty) AS TotalOrderQty
        ,SUM(od.OrderDetailAmt) AS TotalOrderAmt
FROM Orders o
INNER JOIN OrderDetail od ON o.OrderID = od.OrderID
GROUP BY o.OrderDate
ORDER BY o.OrderDate
```

Figure 2E: Query Output for Proposition 2

70 %

Results | Messages

	OrderDate	TotalOrderQty	TotalOrderAmt
1	2012-01-08 00:00:00.000	2768	50781.36
2	2012-01-16 00:00:00.000	10341	304186.869
3	2012-01-20 00:00:00.000	568	6353.256
4	2012-01-24 00:00:00.000	6302	218983.233
5	2012-01-25 00:00:00.000	4766	120101.9085
6	2012-02-09 00:00:00.000	7829	277446.9285
7	2012-02-23 00:00:00.000	631	37056.4635
8	2012-02-27 00:00:00.000	1109	14069.0655
9	2012-03-08 00:00:00.000	12496	288477.798
10	2012-03-09 00:00:00.000	10879	309315.4995
11	2012-03-14 00:00:00.000	27	1010.1105
12	2012-03-28 00:00:00.000	1358	48172.4355
13	2012-04-11 00:00:00.000	9836	298745.118
14	2012-05-02 00:00:00.000	3850	106491.00
15	2012-05-30 00:00:00.000	4060	144196.962
16	2012-06-11 00:00:00.000	1346	27381.4695
17	2012-06-22 00:00:00.000	16067	316508.934
18	2012-06-25 00:00:00.000	5620	167317.71
19	2012-07-27 00:00:00.000	4120	116119.647
20	2012-07-31 00:00:00.000	1112	44223.5325
21	2012-08-15 00:00:00.000	2416	89145.3465
22	2012-08-16 00:00:00.000	1124	41557.6035

Query executed successfully. | 10.0.0.16,12001 (15.0 RTM) | sa (72) | AdventureWorks2017 | 00:00:00 | 29 rows

JSON:

Sample JSON Output with total number of rows returned (29 rows)

Figure 2F: Formatted SQL Query with JSON for Proposition 2

```
USE AdventureWorks2017;

WITH OrderDetail (
        OrderID
        ,OrderDetailQty
        ,OrderDetailAmt
        )
AS (
        SELECT PurchaseOrderID
                ,OrderQty
                ,LineTotal
        FROM Purchasing.PurchaseOrderDetail
        )
        ,Orders (
        OrderID
        ,OrderDate
        )
AS (
        SELECT PurchaseOrderID
                ,OrderDate
        FROM Purchasing.PurchaseOrderHeader h
        WHERE OrderDate > '2012 - 1- 1'
                AND OrderDate < '2013 - 1 - 1'
        )
SELECT o.OrderDate
        ,SUM(od.OrderDetailQty) AS TotalOrderQty
        ,SUM(od.OrderDetailAmt) AS TotalOrderAmt
FROM Orders o
INNER JOIN OrderDetail od ON o.OrderID = od.OrderID
GROUP BY o.OrderDate
ORDER BY o.OrderDate
FOR json path
        ,root('OrderDeets2012')
        ,include_null_values;
```

Figure 2G: Formatted JSON Output for Proposition 2


```
{
   "OrderDeets2012":[
      {
         "OrderDate":"2012-01-08T00:00:00",
         "TotalOrderQty":2768,
         "TotalOrderAmt":50781.3600
      },
      {
         "OrderDate":"2012-01-16T00:00:00",
         "TotalOrderQty":10341,
         "TotalOrderAmt":304186.8690
      },
      {
         "OrderDate":"2012-01-20T00:00:00",
         "TotalOrderQty":568,
         "TotalOrderAmt":6353.2560
      },
      {
         "OrderDate":"2012-01-24T00:00:00",
         "TotalOrderQty":6302,
         "TotalOrderAmt":218983.2330
      },
      {
         "OrderDate":"2012-01-25T00:00:00",
         "TotalOrderQty":4766,
         "TotalOrderAmt":120101.9085
      },
      {
         "OrderDate":"2012-02-09T00:00:00",
         "TotalOrderQty":7829,
         "TotalOrderAmt":277446.9285
      },
      {
         "OrderDate":"2012-02-23T00:00:00",
         "TotalOrderQty":631,
         "TotalOrderAmt":37056.4635
      },
      {
         "OrderDate":"2012-02-27T00:00:00",
         "TotalOrderQty":1109,
         "TotalOrderAmt":14069.0655
      },
      {
```

Proposition 3 (Best Complex)

Proposition 3: Show me who ordered the shark slippers, get the customer and their phone number. Only get customers who aren't businesses.

Model Diagrams:

Figure 3A: Key View Model for Proposition 3

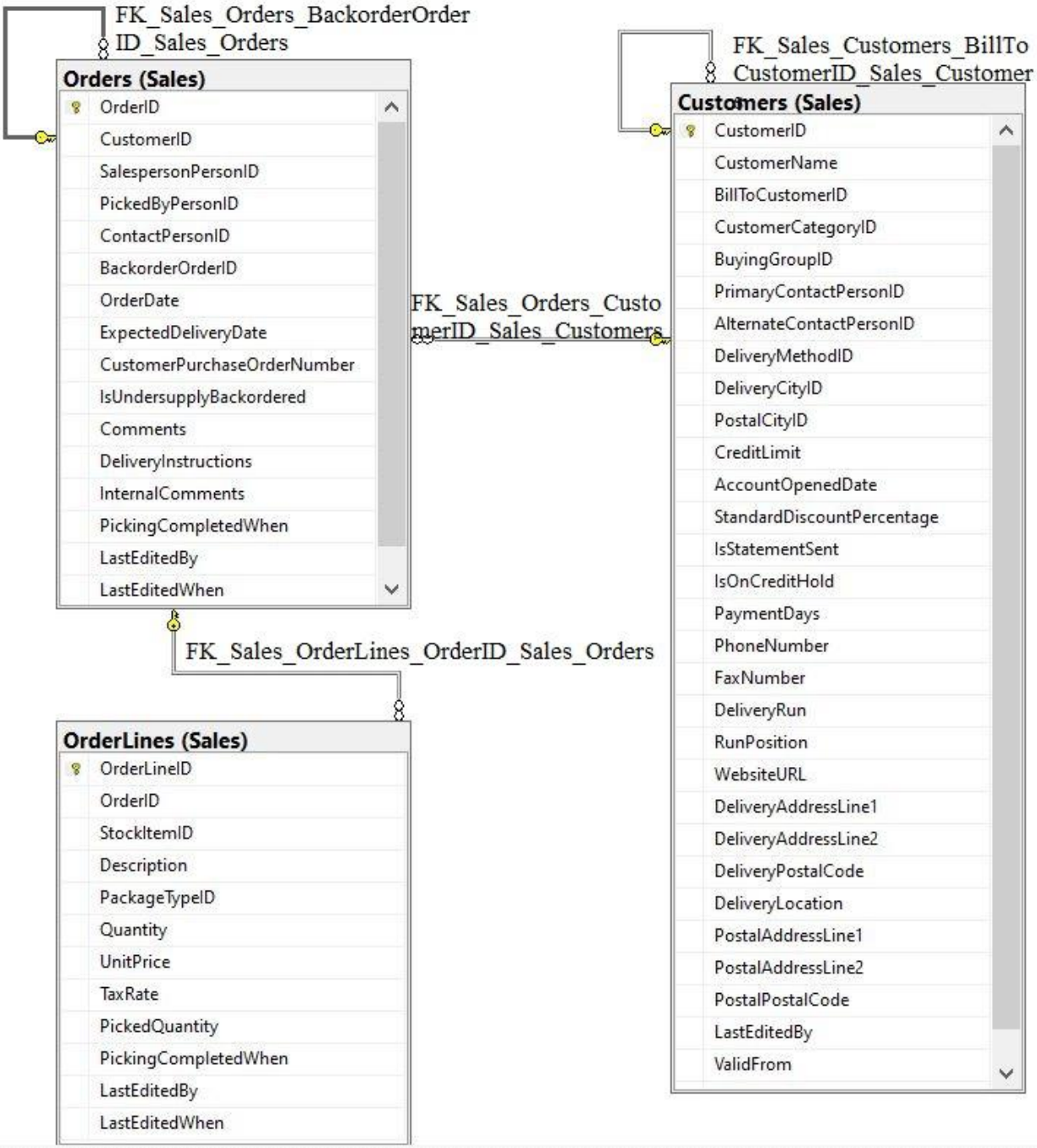
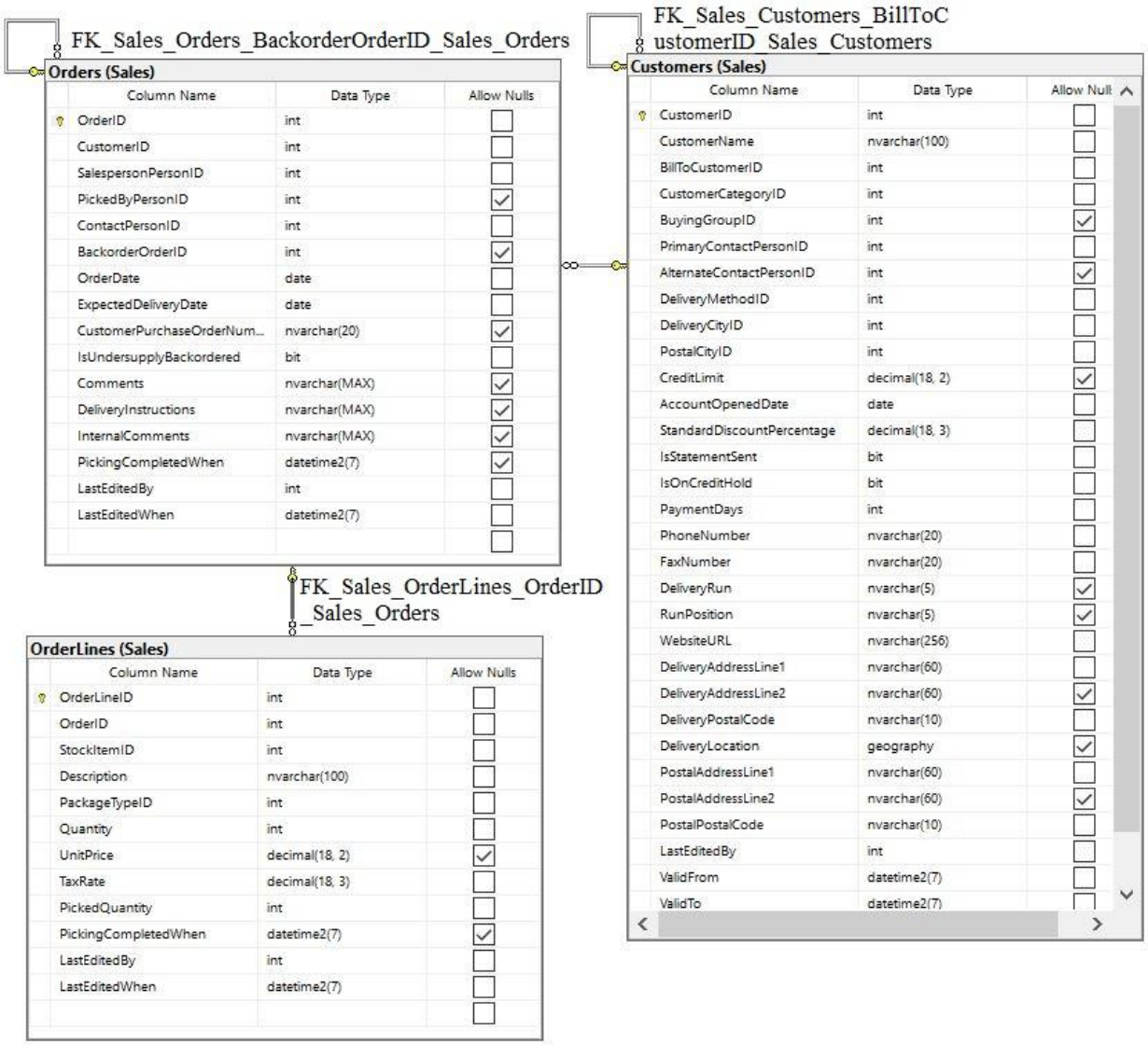


Figure 3B: Standard View Model for Proposition 3



Explanation:

This query uses a function to help find the customers who ordered shark slippers. The information used is from the tables Sales.Customers, Sales.Orders, and Sales.OrderLines.

Figure 3C: Tables for SQL query components

Select clause

Table name:	Column name:
Sales.Customers	CustomerID
Sales.Orders Sales.OrderLines	Descripton StockItemID OrderID

Order by (optional, only if exist)

Table name	Column name	Sort order
Sales.Customers	CustomerId	asc/desc

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 3D: Formatted SQL Query for Proposition 3

```
USE WideWorldImporters;

WITH sharkslipper
AS (
        SELECT Description
                ,StockItemID
                ,OrderID
        FROM Sales.OrderLines
        WHERE Description LIKE '%shark slippers%'
        )
SELECT DISTINCT c.CustomerID
        ,c.CustomerName
        ,c.PhoneNumber
FROM Sales.Orders AS o
INNER JOIN sharkslipper AS s ON s.OrderID = o.OrderID
INNER JOIN sales.Customers AS c ON c.CustomerID = o.CustomerID
WHERE c.BuyingGroupID IS NULL
ORDER BY c.CustomerID;
```

Figure 3E: Query Output for Proposition 3

70 %

Results | Messages

	CustomerID	CustomerName	PhoneNumber
1	801	Eric Torres	(307) 555-0100
2	802	Cosmina Vlad	(505) 555-0100
3	803	Bala Dixit	(209) 555-0100
4	804	Aleksandrs Riekstins	(605) 555-0100
5	805	Ratan Poddar	(907) 555-0100
6	806	Shi Tu	(307) 555-0100
7	807	Gunnar Lohmus	(201) 555-0100
8	808	Jackson Kolios	(209) 555-0100
9	809	Madhu Dwivedi	(802) 555-0100
10	810	Alena Kellnerova	(303) 555-0100
11	811	Surendra Sahu	(210) 555-0100
12	812	Celica Barajas	(270) 555-0100
13	813	Shyam Poddar	(218) 555-0100
14	814	Johanna Hoornstra	(210) 555-0100
15	815	Libuse Valentova	(270) 555-0100
16	816	Harsha Huq	(314) 555-0100
17	817	Agrita Kanepa	(201) 555-0100

Query executed successfully. | 10.0.0.16,12001 (15.0 RTM) | sa (71) | WideWorldImporters | 00:00:00 | 252 rows

JSON:

Sample JSON Output with total number of rows returned (252 rows)

Figure 3F: Formatted SQL Query with JSON for Proposition 3

```
USE WideWorldImporters;

WITH sharkslipper
AS (
        SELECT Description
                ,StockItemID
                ,OrderID
        FROM Sales.OrderLines
        WHERE Description LIKE '%shark slippers%'
        )
SELECT DISTINCT c.CustomerID
        ,c.CustomerName
        ,c.PhoneNumber
FROM Sales.Orders AS o
INNER JOIN sharkslipper AS s ON s.OrderID = o.OrderID
INNER JOIN sales.Customers AS c ON c.CustomerID = o.CustomerID
WHERE c.BuyingGroupID IS NULL
ORDER BY c.CustomerID;
FOR

json path
        ,root('Slippers')
        ,include_null_values;
```

Figure 3G: Formatted JSON Output for Proposition 3


```
{
   "Slippers":[
      {
         "CustomerID":801,
         "CustomerName":"Eric Torres",
         "PhoneNumber":"(307) 555-0100"
      },
      {
         "CustomerID":802,
         "CustomerName":"Cosmina Vlad",
         "PhoneNumber":"(505) 555-0100"
      },
      {
         "CustomerID":803,
         "CustomerName":"Bala Dixit",
         "PhoneNumber":"(209) 555-0100"
      },
      {
         "CustomerID":804,
         "CustomerName":"Aleksandrs Riekstins'
```

Proposition 4 (Worst Simple)

Proposition 4: Show me the BusinessEntityID of people living in Ballard

Model Diagrams:

Figure 4A: Key View Model for Proposition 4

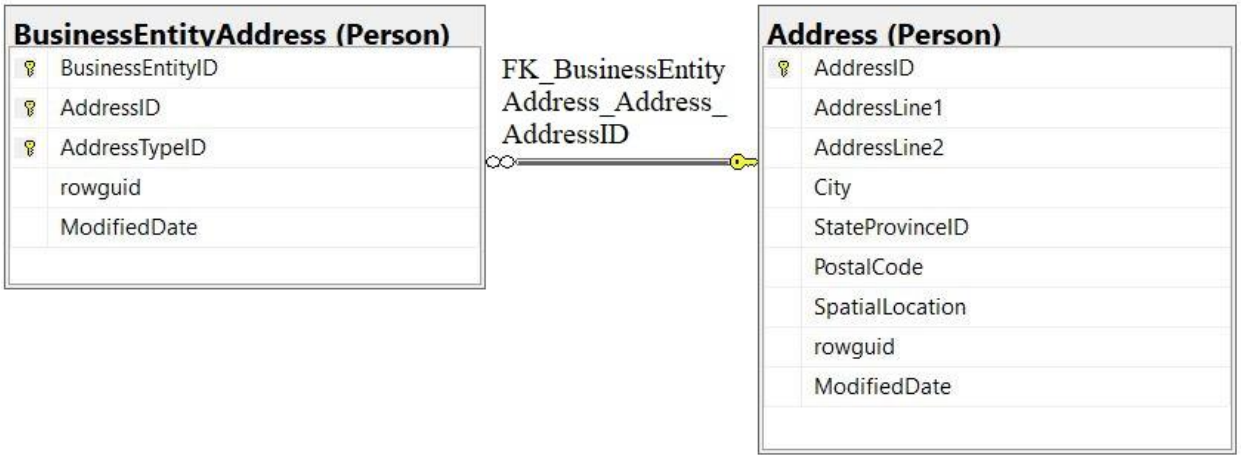
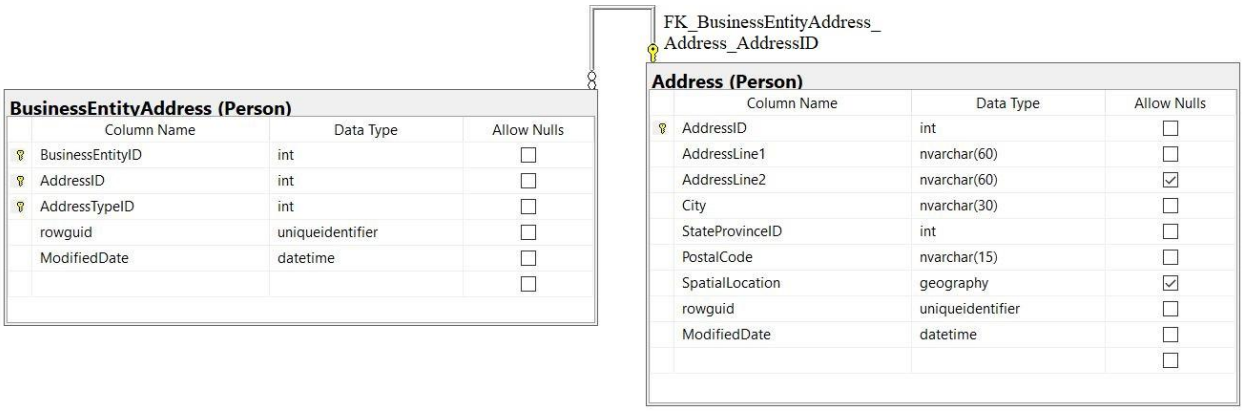


Figure 4B: Standard View Model for Proposition 4



Explanation:

This query uses one inner join to reveal the BusinessEntityId of people living in Ballard. It combines the AddressId from the tables BusinessEntityAddress and Person.Address.

Figure 4C: Tables for SQL query components

Select clause

Table name:	Column name:
BusinessEntityAddress	AddressID BusinessEntityID
Person.Address	City AddressID

Order by (optional, only if exist)

Table name	Column name	Sort order
Person.Address	BusinessEntityID	asc

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 4D: Formatted SQL Query for Proposition 4

```
USE AdventureWorks2017

SELECT City
        ,BusinessEntityID
FROM Person.Address
INNER JOIN Person.BusinessEntityAddress ON Address.AddressID = BusinessEntityAddress.AddressID
WHERE City = 'Ballard'
ORDER BY City
```

Figure 4E: Query Output for Proposition 4

Results | Messages

	City	BusinessEntityID
1	Ballard	1614
2	Ballard	2397
3	Ballard	2573
4	Ballard	3083
5	Ballard	3270
6	Ballard	3942
7	Ballard	4003
8	Ballard	4268
9	Ballard	4498
10	Ballard	4820
11	Ballard	4959
12	Ballard	5124
13	Ballard	5653
14	Ballard	5668
15	Ballard	6062
16	Ballard	6721
17	Ballard	7375
18	Ballard	7660
19	Ballard	7988

Query executed successfully. | 10.0.0.16,12001 (15.0 RTM) | sa (72) | AdventureWorks2017 | 00:00:00 | 69 rows

JSON:

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

Figure 4F: Formatted SQL Query with JSON for Proposition 4

```
USE AdventureWorks2017

SELECT City
        ,BusinessEntityID
FROM Person.Address
INNER JOIN Person.BusinessEntityAddress ON Address.AddressID = BusinessEntityAddress.AddressID
WHERE City = 'Ballard'
ORDER BY City
FOR json path
        ,root('Ballard')
        ,include_null_values;
```

Figure 4G: Formatted JSON Output for Proposition 4


```
{
   "Ballard":[
      {
         "City":"Ballard",
         "BusinessEntityID":1614
      },
      {
         "City":"Ballard",
         "BusinessEntityID":2397
      },
      {
         "City":"Ballard",
         "BusinessEntityID":2573
      },
      {
         "City":"Ballard",
         "BusinessEntityID":3083
      },
      {
         "City":"Ballard",
         "BusinessEntityID":3270
      },
      {
```

Proposition 5 (Worst Medium)

Proposition 5: Show me Full Name of customers whose credit cards expire after February 2007

Model Diagrams:

Figure 5A: Key View Model for Proposition 5

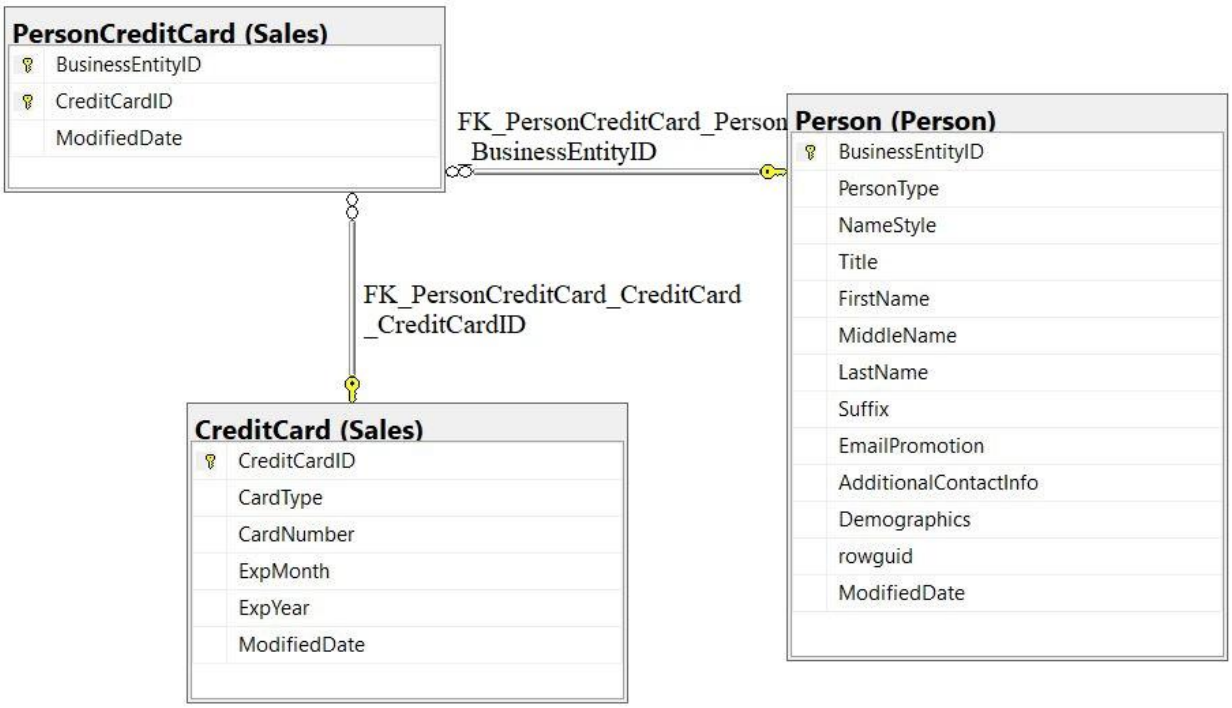
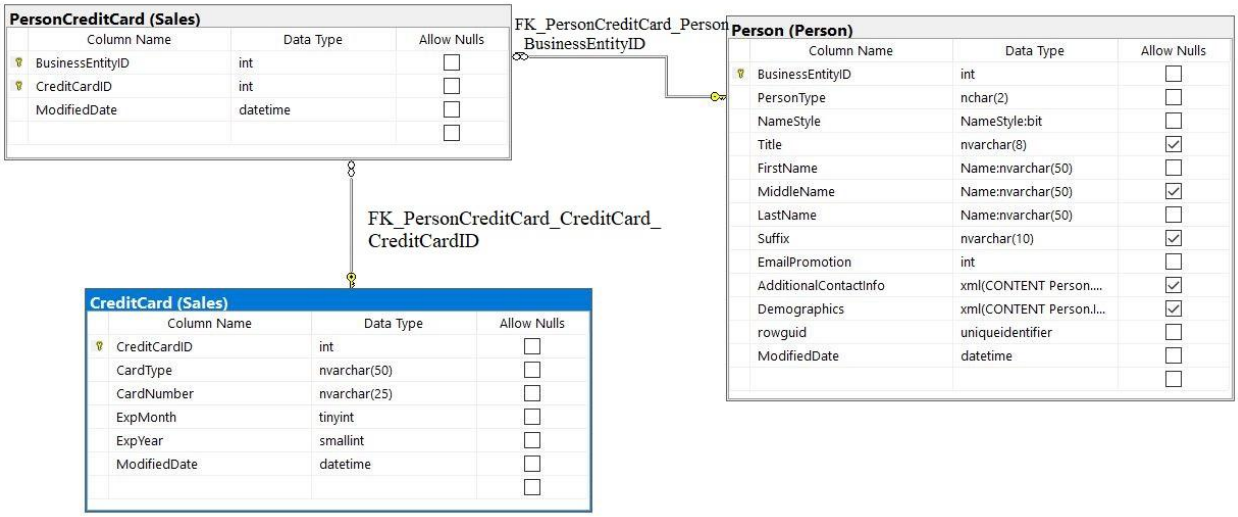


Figure 5B: Standard View Model for Proposition 5



Explanation:

summary explanation that will help the developer with the proposition.

Figure 5C: Tables for SQL query components

Select clause

Table name:	Column name:
Person.Person	FirstName,LastName
Sales.PersonCreditCard Sales.CreditCard	ExpMonth,ExpYear BusinessEntityID

Order by (optional, only if exist)

Table name	Column name	Sort order
Example table	Example column	asc/desc

Query:

All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

Figure 5D: Formatted SQL Query for Proposition 5

```
USE AdventureWorks2017

SELECT FirstName
        ,LastName
        ,ExpMonth
        ,ExpYear
FROM Person.[Person]
INNER JOIN (
        Sales.PersonCreditCard INNER JOIN sales.CreditCard ON CreditCard.CreditCardID = PersonCreditCard.CreditCardID
        ) ON PersonCreditCard.BusinessEntityID = person.BusinessEntityID
WHERE ExpMonth > '2'
        AND ExpYear >= '2007'
ORDER BY ExpYear
        ,ExpMonth
```

Figure 5E: Query Output for Proposition 5

70 %

Results | Messages

	FirstName	LastName	ExpMonth	ExpYear
1	Connor	Adams	3	2007
2	Logan	Adams	3	2007
3	Jordan	Alexander	3	2007
4	Robyn	Alvarez	3	2007
5	Ruben	Alvarez	3	2007
6	Alvin	Andersen	3	2007
7	Mitchell	Andersen	3	2007
8	Jose	Anderson	3	2007
9	Kayla	Anderson	3	2007
10	Brandi	Ashe	3	2007
11	Chloe	Bailey	3	2007
12	Jacqueline	Bailey	3	2007
13	Riley	Bailey	3	2007
14	Xavier	Bailey	3	2007
15	Gabriella	Baker	3	2007
16	Noah	Baker	3	2007
17	Sean	Baker	3	2007
18	Melissa	Barnes	3	2007
19	Bryan	Bell	3	2007
20	Cody	Bell	3	2007
21	Shane	Belli	3	2007

Query executed successfully. | 10.0.0.16,12001 (15.0 RTM) | sa (72) | AdventureWorks2017 | 00:00:00 | 8,002 rows

JSON:

Sample JSON Output with total number of rows returned (8,002 rows)

Figure 5F: Formatted SQL Query with JSON for Proposition 5

```sql
USE AdventureWorks2017

SELECT FirstName
        ,LastName
        ,ExpMonth
        ,ExpYear
FROM Person.[Person]
INNER JOIN (
        Sales.PersonCreditCard INNER JOIN sales.CreditCard ON CreditCard.CreditCardID = PersonCreditCard.CreditCardID
        ) ON PersonCreditCard.BusinessEntityID = person.BusinessEntityID
WHERE ExpMonth > '2'
        AND ExpYear >= '2007'
ORDER BY ExpYear
        ,ExpMonth
FOR json path
        ,root('Total Sales')
        ,include_null_values;
```

Figure 5G: Formatted JSON Output for Proposition 5


```
{
   "Exp2007":[
      {
         "FirstName":"Connor",
         "LastName":"Adams",
         "ExpMonth":3,
         "ExpYear":2007
      },
      {
         "FirstName":"Logan",
         "LastName":"Adams",
         "ExpMonth":3,
         "ExpYear":2007
      },
      {
         "FirstName":"Jordan",
         "LastName":"Alexander",
         "ExpMonth":3,
         "ExpYear":2007
      },
      {
         "FirstName":"Robyn",
         "LastName":"Alvarez",
         "ExpMonth":3,
         "ExpYear":2007
      },
      {
         "FirstName":"Ruben",
         "LastName":"Alvarez",
         "ExpMonth":3,
         "ExpYear":2007
      },
      {
         "FirstName":"Alvin",
         "LastName":"Andersen",
         "ExpMonth":3,
         "ExpYear":2007
      },
      {
         "FirstName":"Mitchell",
         "LastName":"Andersen",
         "ExpMonth":3,
         "ExpYear":2007
```

Proposition 6 (Worst Complex)

Proposition 6: Show me which customers ordered a Front Derailleur for their bikes

Model Diagrams:

Figure 6A: Key View Model for Proposition 6

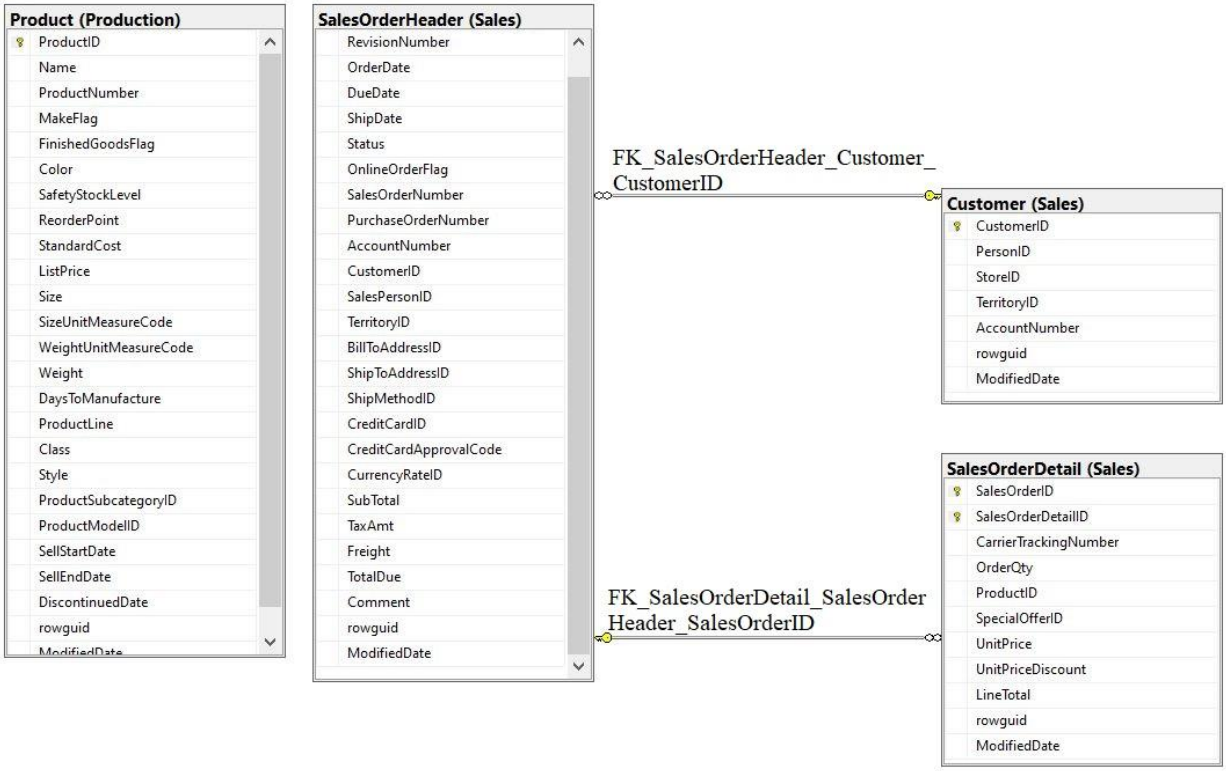
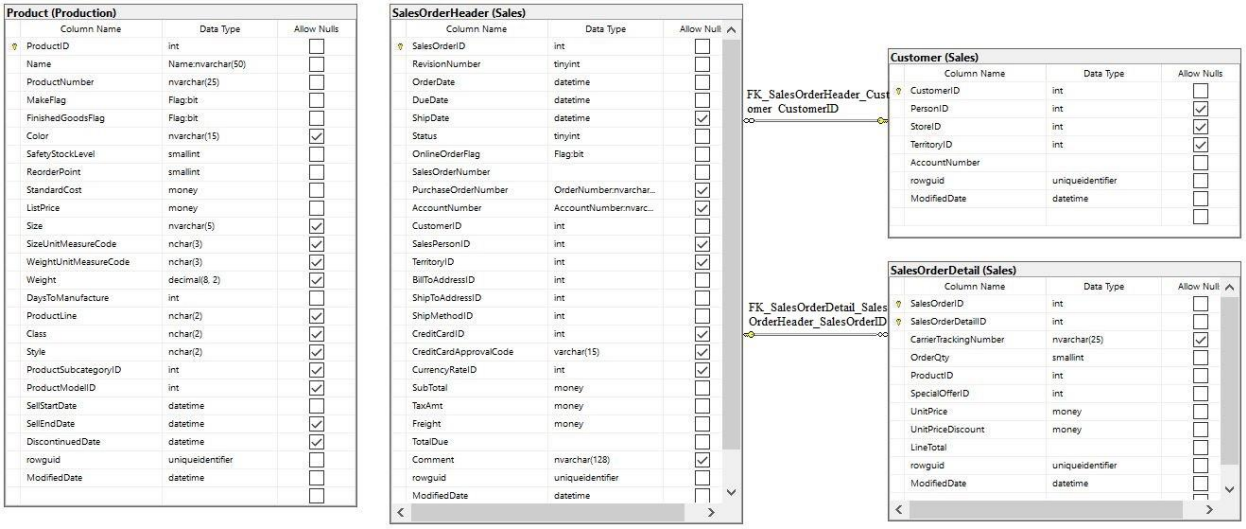


Figure 6B: Standard View Model for Proposition 6



Explanation:

This uses three join statements to pinpoint which customers bought a front derailleur for their bikes. This uses the tables Sales.SalesOrderDetail, Production.Product, Sales.Customer, and Sales.SalesOrderHeader.

Figure 6C: Tables for SQL query components

Select clause

Table name:	Column name:
Production.Product	ProductID
Sales.Customer Sales.SalesOrderHeader Sales.SalesOrderDetail	ProductID SalesOrderID CustomerID

Order by (optional, only if exist)

Table name	Column name	Sort order
Sales.SalesOrderDetail	OrderQty	DESC

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 6D: Formatted SQL Query for Proposition 6

```
USE AdventureWorks2017

SELECT Product.ProductModelID
        ,Product.Name
        ,Customer.PersonID
        ,SalesOrderDetail.OrderQty
FROM Production.Product
JOIN Sales.SalesOrderDetail ON Product.ProductID = SalesOrderDetail.ProductID
JOIN Sales.SalesOrderHeader ON SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID
JOIN Sales.Customer ON SalesOrderHeader.CustomerID = Customer.CustomerID
WHERE Product.ProductModelID = (
                SELECT ProductModelID
                FROM Production.ProductModel
                WHERE Name = 'Front Derailleur'
                )
ORDER BY OrderQty DESC
```

Figure 6E: Query Output for Proposition 6

Results | Messages

	ProductModelID	Name	PersonID	OrderQty
1	103	Front Derailleur	827	14
2	103	Front Derailleur	807	13
3	103	Front Derailleur	1261	12
4	103	Front Derailleur	851	11
5	103	Front Derailleur	1283	10
6	103	Front Derailleur	833	10
7	103	Front Derailleur	787	10
8	103	Front Derailleur	1835	10
9	103	Front Derailleur	1839	8
10	103	Front Derailleur	1835	8
11	103	Front Derailleur	1835	8
12	103	Front Derailleur	1321	8
13	103	Front Derailleur	1323	8
14	103	Front Derailleur	1429	8
15	103	Front Derailleur	1429	8
16	103	Front Derailleur	1969	8
17	103	Front Derailleur	1969	8
18	103	Front Derailleur	611	8
19	103	Front Derailleur	873	8
20	103	Front Derailleur	827	7

Query executed successfully. | 10.0.0.16,12001 (15.0 RTM) | sa (72) | AdventureWorks2017 | 00:00:00 | 257 rows

JSON:

Sample JSON Output with total number of rows returned (257 rows)

Figure 6F: Formatted SQL Query with JSON for Proposition 6

```
USE AdventureWorks2017

SELECT Product.ProductModelID
        ,Product.Name
        ,Customer.PersonID
        ,SalesOrderDetail.OrderQty
FROM Production.Product
JOIN Sales.SalesOrderDetail ON Product.ProductID = SalesOrderDetail.ProductID
JOIN Sales.SalesOrderHeader ON SalesOrderDetail.SalesOrderID = SalesOrderHeader.SalesOrderID
JOIN Sales.Customer ON SalesOrderHeader.CustomerID = Customer.CustomerID
WHERE Product.ProductModelID = (
                SELECT ProductModelID
                FROM Production.ProductModel
                WHERE Name = 'Front Derailleur'
                )
ORDER BY OrderQty DESC
FOR json path
        ,root('Exp2007')
        ,include_null_values;
```

Figure 6G: Formatted JSON Output for Proposition 6


```
{
   "Front Derailleur":[
      {
         "ProductModelID":103,
         "Name":"Front Derailleur",
         "PersonID":827,
         "OrderQty":14
      },
      {
         "ProductModelID":103,
         "Name":"Front Derailleur",
         "PersonID":807,
         "OrderQty":13
      },
      {
         "ProductModelID":103,
         "Name":"Front Derailleur",
         "PersonID":1261,
         "OrderQty":12
      },
      {
         "ProductModelID":103,
         "Name":"Front Derailleur",
         "PersonID":851,
         "OrderQty":11
      },
      {
         "ProductModelID":103,
         "Name":"Front Derailleur",
         "PersonID":1283,
         "OrderQty":10
      },
      {
         "ProductModelID":103,
         "Name":"Front Derailleur",
         "PersonID":833,
         "OrderQty":10
      },
      {
         "ProductModelID":103,
         "Name":"Front Derailleur",
         "PersonID":787,
         "OrderQty":10
```

Proposition 7 (Improved Simple)

Proposition 7: Show me the name of the products with the most quantity

Model Diagrams:

Figure 7A: Key View Model for Proposition 7

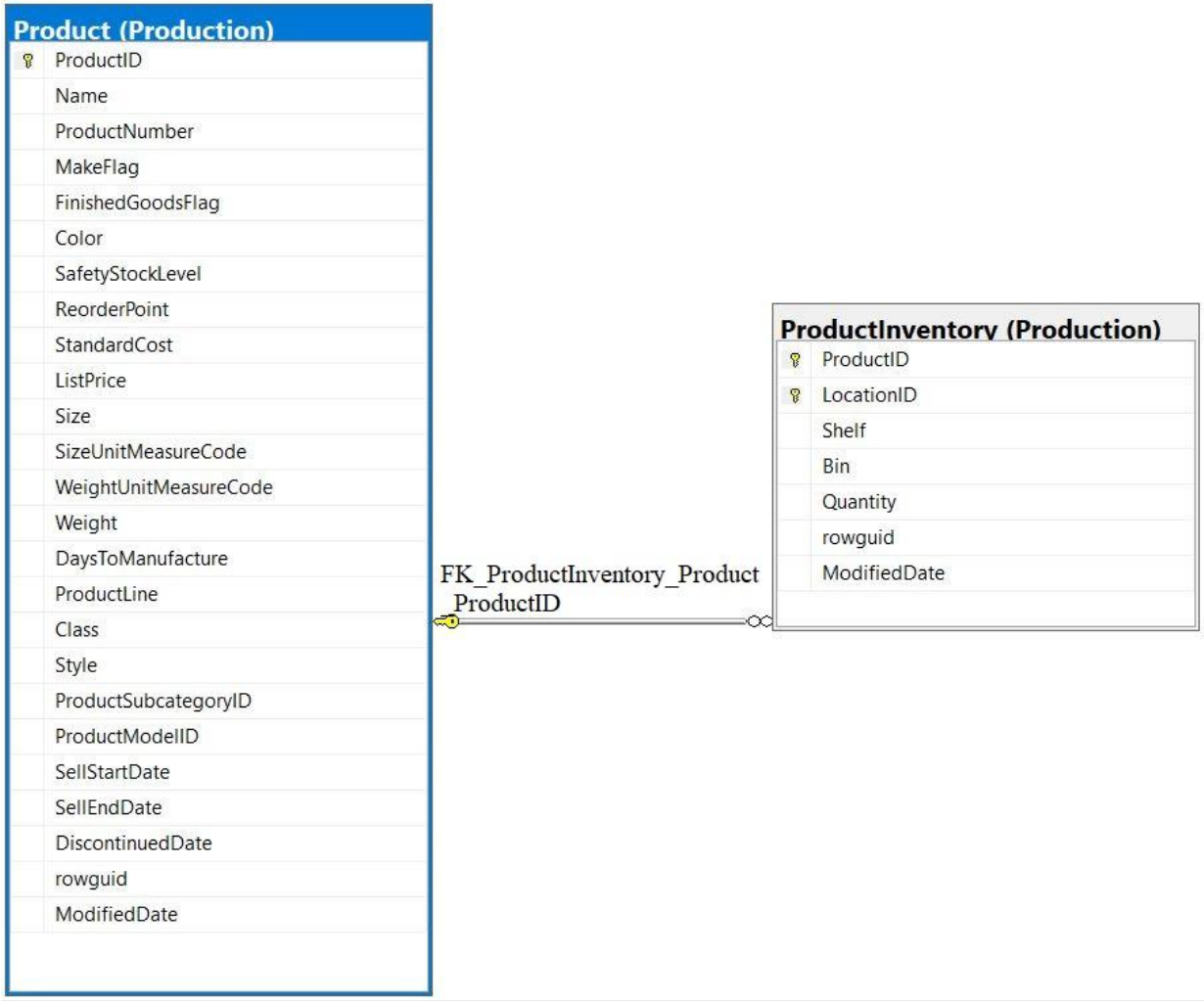
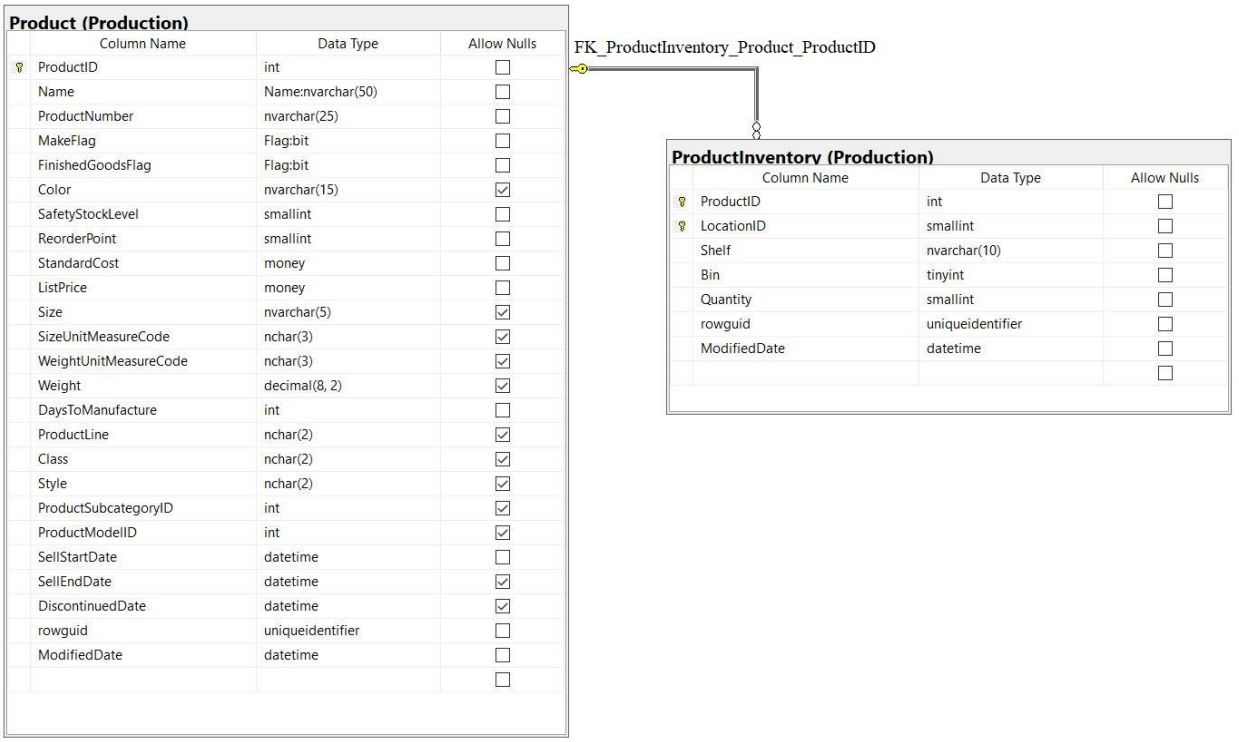


Figure 7B: Standard View Model for Proposition 7



Explanation:

This query uses an inner join to combine tables Production.Product and Production.ProductInventory. The columns that are combined are the ProductID. This is done to show the products with the most quantity. The results only show the top 10 through the TOP clause.

Figure 7C: Tables for SQL query components

Select clause

Table name:	Column name:
Production.Product	Name ProductID
Production.ProductInventory	Quantity ProductID

Order by (optional, only if exist)

Table name	Column name	Sort order
Production.ProductInventory	Quantity	DESC

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 7D: Formatted SQL Query for Proposition 7

```
USE AdventureWorks2017

SELECT TOP 10 Name
        ,Quantity
FROM Production.Product
INNER JOIN Production.ProductInventory ON Product.ProductID = ProductInventory.ProductID
ORDER BY Quantity DESC
```

Figure 7E: Query Output for Proposition 7

70 %

Results | Messages

	Name	Quantity
1	Seat Lug	924
2	Hex Nut 7	897
3	Spokes	888
4	Hex Nut 14	780
5	Hex Nut 19	763
6	Seat Lug	729
7	Touring Rim	724
8	Seat Stays	715
9	Hex Nut 10	710
10	Spokes	702

Query executed successfully. | 10.0.0.16,12001 (15.0 RTM) | sa (72) | AdventureWorks2017 | 00:00:00 | 10 rows

JSON:

Sample JSON Output with total number of rows returned (10 rows)

Figure 7F: Formatted SQL Query with JSON for Proposition 7

```
USE AdventureWorks2017

SELECT TOP 10 Name
        ,Quantity
FROM Production.Product
INNER JOIN Production.ProductInventory ON Product.ProductID = ProductInventory.ProductID
ORDER BY Quantity DESC
FOR json path
        ,root('top')
        ,include_null_values;
```

Figure 7G: Formatted JSON Output for Proposition 7


```
{
   "top":[
      {
         "Name":"Seat Lug",
         "Quantity":924
      },
      {
         "Name":"Hex Nut 7",
         "Quantity":897
      },
      {
         "Name":"Spokes",
         "Quantity":888
      },
      {
         "Name":"Hex Nut 14",
         "Quantity":780
      },
      {
         "Name":"Hex Nut 19",
         "Quantity":763
      },
      {
         "Name":"Seat Lug",
         "Quantity":729
      },
      {
         "Name":"Touring Rim",
         "Quantity":724
      },
      {
         "Name":"Seat Stays",
         "Quantity":715
      },
      {
         "Name":"Hex Nut 10",
         "Quantity":710
      },
      {
         "Name":"Spokes",
         "Quantity":702
      }
   ]
}
```

Proposition 8 (Improved Medium)

Proposition 8: Show me the largest to smallest orders along with the weight of the order

Model Diagrams:

Figure 8A: Key View Model for Proposition 8

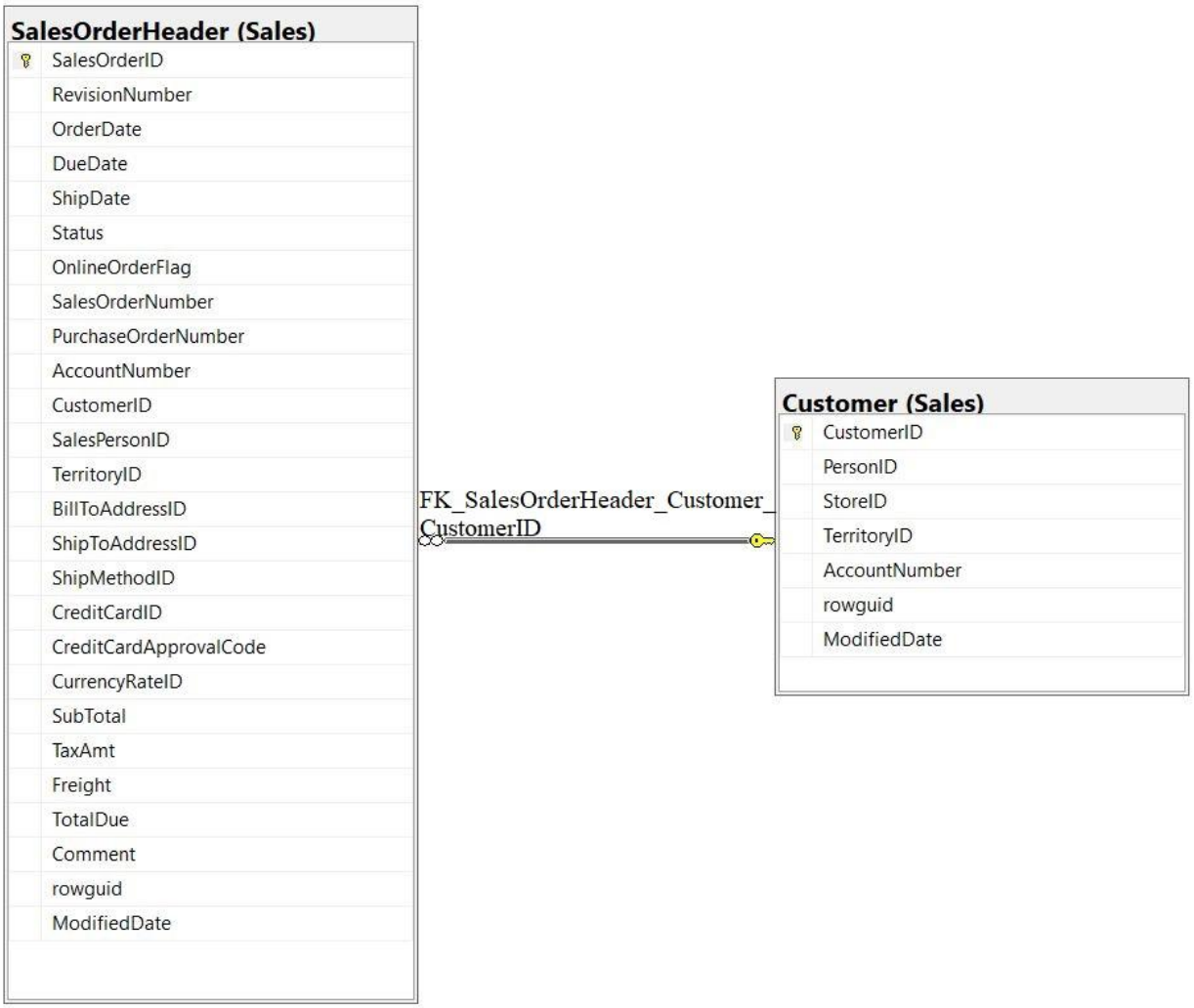


Figure 8B: Standard View Model for Proposition 8

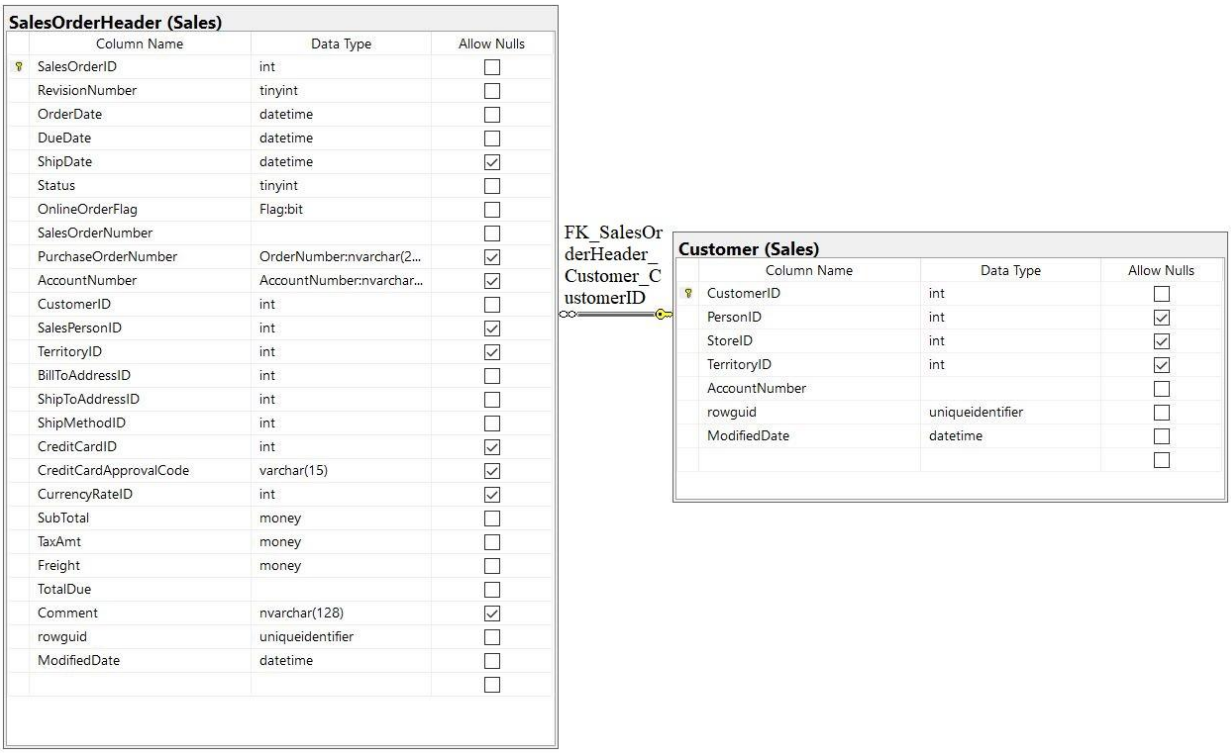


Figure 8C: Tables for SQL query components

Select clause

Table name:	Column name:
Example table	Example columns

Order by (optional, only if exist)

Table name	Column name	Sort order
Example table	Example column	asc/desc

Query:

All queries use ANSI 92 standard with type “safe” on, formatted using poorsql.com.

Figure 8D: Formatted SQL Query for Proposition 8

```sql
USE AdventureWorks2017

SELECT PersonID
        ,SubTotal
        ,a.total_weight
FROM Sales.SalesOrderHeader
JOIN Sales.Customer ON SalesOrderHeader.CustomerID = Customer.CustomerID
JOIN (
        SELECT SalesOrderDetail.SalesOrderID
                ,SUM(Product.Weight * SalesOrderDetail.OrderQty) AS total_weight
        FROM Production.Product
        JOIN Sales.SalesOrderDetail ON Product.ProductID = SalesOrderDetail.ProductID
        GROUP BY SalesOrderID
        ) AS a ON SalesOrderHeader.SalesOrderID = a.SalesOrderID
ORDER BY SalesOrderHeader.SubTotal DESC;
FOR

json path
        ,root('Total Sales')
        ,include_null_values;
```

Figure 8E: Query Output for Proposition 8

Results | Messages

	PersonID	SubTotal	total_weight
1	651	163930.3943	4603.66
2	651	160378.3913	3815.21
3	591	150837.4387	16480.77
4	1961	147390.9328	11675.43
5	785	146154.5653	12272.46
6	1425	140078.3959	19035.59
7	1335	129261.254	19397.84
8	1241	128873.2206	5945.30
9	615	126198.3362	1261.70
10	1261	122285.724	13948.29
11	1299	122284.4578	4087.84
12	615	121761.9396	1206.95
13	1417	120182.185	21531.87
14	661	117274.3453	11898.80
15	651	116153.8278	3292.47
16	1125	115696.3313	1124.96
17	1843	115310.4777	17993.17
18	813	112722.8945	14623.92
19	599	112611.9697	4908.16

Query executed successfully. | 10.0.0.16,12001 (15.0 RTM) | sa (72) | AdventureWorks2017 | 00:00:00 | 31,465 rows

JSON:

Sample JSON Output with total number of rows returned (31,456 rows)

Figure 8F: Formatted SQL Query with JSON for Proposition 8

```sql
USE AdventureWorks2017

SELECT PersonID
        ,SubTotal
        ,a.total_weight
FROM Sales.SalesOrderHeader
JOIN Sales.Customer ON SalesOrderHeader.CustomerID = Customer.CustomerID
JOIN (
        SELECT SalesOrderDetail.SalesOrderID
                ,SUM(Product.Weight * SalesOrderDetail.OrderQty) AS total_weight
        FROM Production.Product
        JOIN Sales.SalesOrderDetail ON Product.ProductID = SalesOrderDetail.ProductID
        GROUP BY SalesOrderID
        ) AS a ON SalesOrderHeader.SalesOrderID = a.SalesOrderID
ORDER BY SalesOrderHeader.SubTotal DESC;
FOR

json path
        ,root('Total Sales')
        ,include_null_values;
```

Figure 8G: Formatted JSON Output for Proposition 8


```
{
   "Total Sales":[
      {
         "PersonID":651,
         "SubTotal":163930.3943,
         "total_weight":4603.66
      },
      {
         "PersonID":651,
         "SubTotal":160378.3913,
         "total_weight":3815.21
      },
      {
         "PersonID":591,
         "SubTotal":150837.4387,
         "total_weight":16480.77
      },
      {
         "PersonID":1961,
         "SubTotal":147390.9328,
         "total_weight":11675.43
      },
      {
         "PersonID":785,
         "SubTotal":146154.5653,
         "total_weight":12272.46
      },
      {
         "PersonID":1425,
         "SubTotal":140078.3959,
         "total_weight":19035.59
      },
      {
         "PersonID":1335,
         "SubTotal":129261.2540,
         "total_weight":19397.84
      },
      {
         "PersonID":1241,
         "SubTotal":128873.2206,
         "total_weight":5945.30
      },
```

Proposition 9 (Improved Complex)

Proposition 9: Show customer 30, see if they belong to the US region and show which employee helped.

Model Diagrams:

Figure 9A: Key View Model for Proposition 9

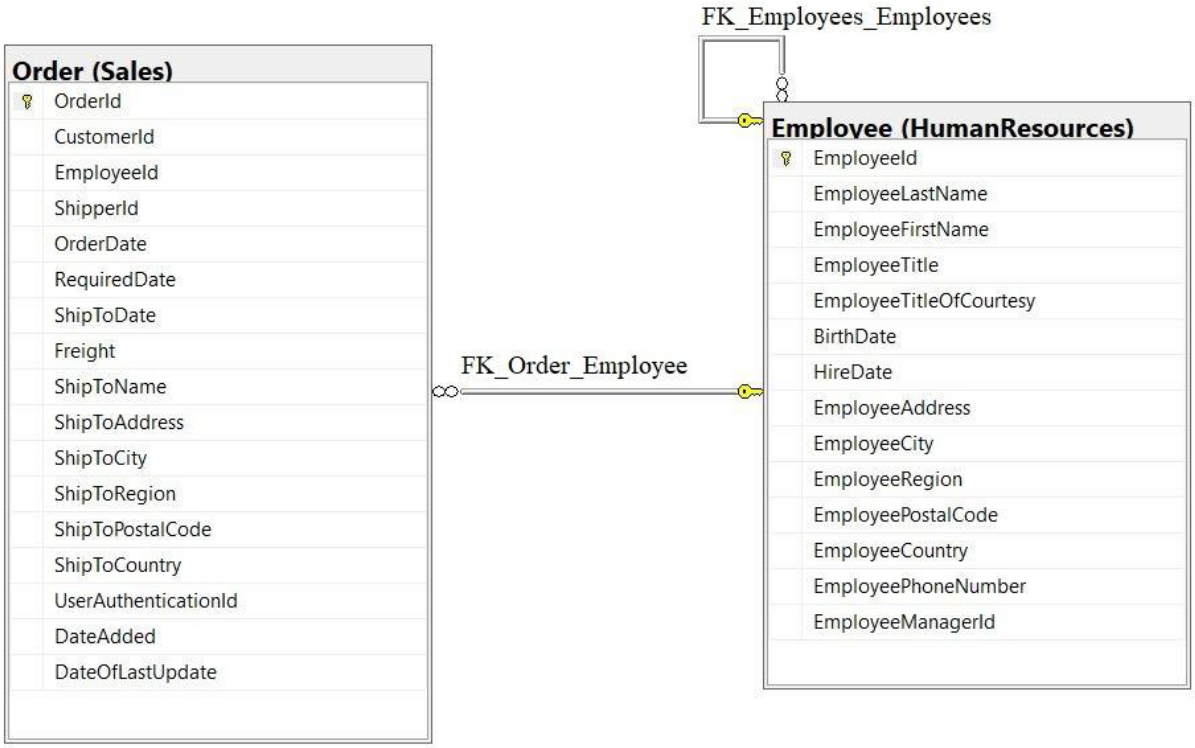
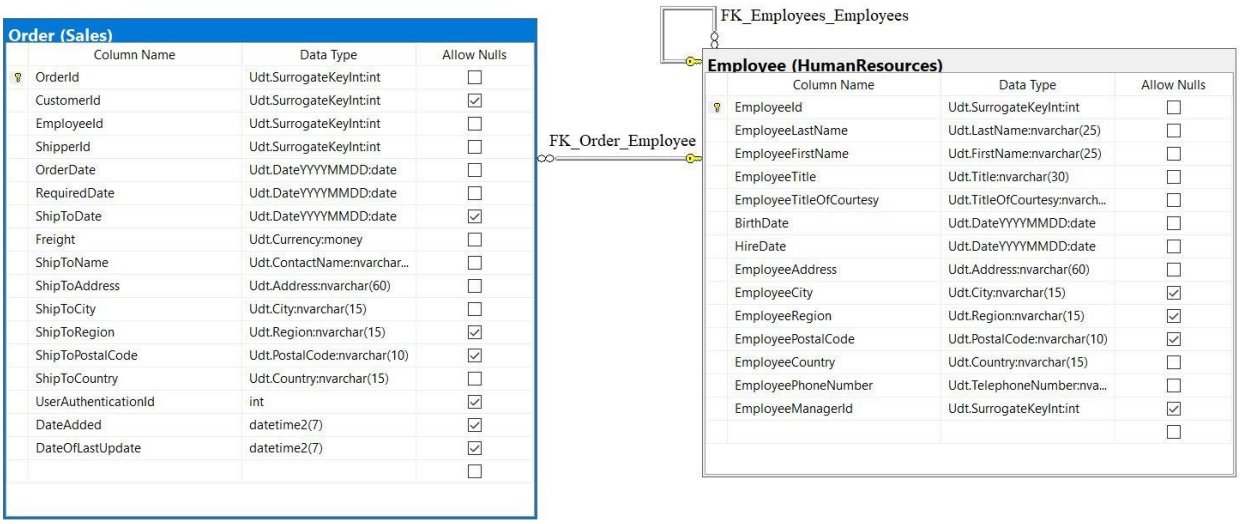


Figure 9B: Standard View Model for Proposition 9



Explanation:

summary explanation that will help the developer with the proposition.

Figure 9C: Tables for SQL query components

Select clause

Table name:	Column name:
Example table	Example columns

Order by (optional, only if exist)

Table name	Column name	Sort order
Example table	Example column	asc/desc

Query:

All queries use ANSI 92 standard with type "safe" on, formatted using poorsql.com.

Figure 9D: Formatted SQL Query for Proposition 9

```
USE Northwinds2020TSQLV6

DROP FUNCTION

IF EXISTS sales.custwho GO
        CREATE FUNCTION sales.custwho (@custid AS INT)
        RETURNS TABLE
        AS
        RETURN

        SELECT CustomerId
                ,orderid
                ,employeeid
                ,ShipToCountry
        FROM Sales.[Order]
        WHERE @custid = CustomerId
GO

DROP VIEW

IF EXISTS Sales.country;GO
        CREATE VIEW sales.country
        AS
        SELECT ShipToCountry
        FROM sales.[Order]
        WHERE ShipToCountry = 'USA'
GO

;

WITH EMP
AS (
        SELECT EmployeeId
        FROM HumanResources.Employee
        )

SELECT DISTINCT c.customerid
        ,EMP.EmployeeId
FROM sales.custwho(65) AS c
LEFT OUTER JOIN EMP AS EMP ON EMP.EmployeeId = c.EmployeeId
INNER JOIN sales.country AS r ON c.ShipToCountry = r.ShipToCountry
```

Figure 9E: Query Output for Proposition 9

Results | Messages

	customerid	EmployeeId
1	65	1
2	65	2
3	65	3
4	65	4
5	65	5
6	65	6
7	65	8
8	65	9

Query executed successfully. | 10.0.0.16,12001 (15.0 RTM) | sa (71) | Northwinds2020TSQLV6 | 00:00:00 | 8 rows

JSON:

Sample JSON Output with total number of rows returned (8)

Figure 9F: Formatted SQL Query with JSON for Proposition 9

```
USE Northwinds2020TSQLV6

DROP FUNCTION

IF EXISTS sales.custwho GO
        CREATE FUNCTION sales.custwho (@custid AS INT)
        RETURNS TABLE
        AS
        RETURN

        SELECT CustomerId
                ,orderid
                ,employeeid
                ,ShipToCountry
        FROM Sales.[Order]
        WHERE @custid = CustomerId
GO

DROP VIEW

IF EXISTS Sales.country;GO
        CREATE VIEW sales.country
        AS
        SELECT ShipToCountry
        FROM sales.[Order]
        WHERE ShipToCountry = 'USA'
GO

;

WITH EMP
AS (
        SELECT EmployeeId
        FROM HumanResources.Employee
        )
  SELECT DISTINCT c.customerid
         ,EMP.EmployeeId
  FROM sales.custwho(65) AS c
  LEFT OUTER JOIN EMP AS EMP ON EMP.EmployeeId = c.EmployeeId
  INNER JOIN sales.country AS r ON c.ShipToCountry = r.ShipToCountry
SELECT DISTINCT c.customerid
        ,EMP.EmployeeId
FROM sales.custwho(65) AS c
LEFT OUTER JOIN EMP AS EMP ON EMP.EmployeeId = c.EmployeeId
INNER JOIN sales.country AS r ON c.ShipToCountry = r.ShipToCountry
FOR json path
        ,root('Customer30')
        ,include_null_values;
```

Figure 9G: Formatted JSON Output for Proposition 9


```
{
    "Customer30":[
        {
            "customerid":65,
            "EmployeeId":1
        },
        {
            "customerid":65,
            "EmployeeId":2
        },
        {
            "customerid":65,
            "EmployeeId":3
        },
        {
            "customerid":65,
            "EmployeeId":4
        },
        {
            "customerid":65,
            "EmployeeId":5
        },
        {
            "customerid":65,
            "EmployeeId":6
        },
        {
            "customerid":65,
            "EmployeeId":8
        },
        {
            "customerid":65,
            "EmployeeId":9
        }
    ]
}
```