Project 1 Propositions

9 queries selected from everyone: (3) worst (3) best (3) improved

OCT 2021

ISSUED BY
10:45AM Group 4

REPRESENTATIVE
Min Joo Jeong
Lindita Kalaj
Micheal Cao
Saqib Mahmood

Table of Contents

Proposition 1 (Best Simple)- Saqib

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

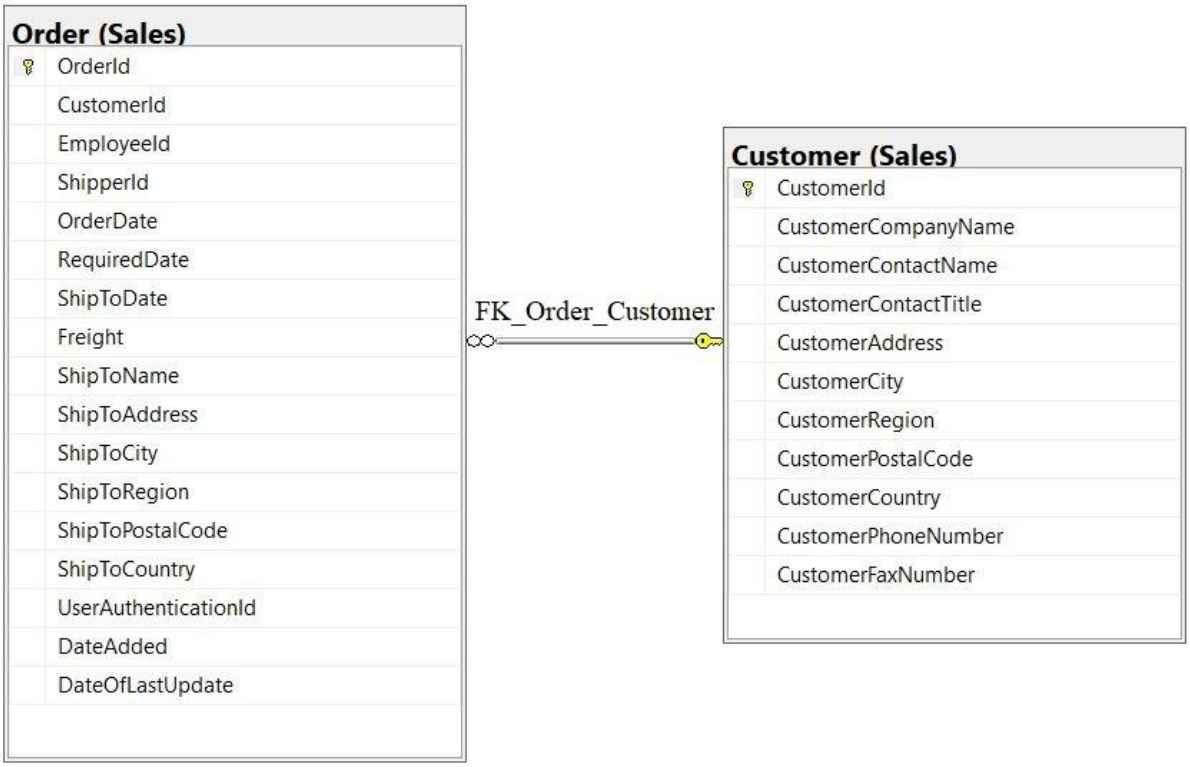
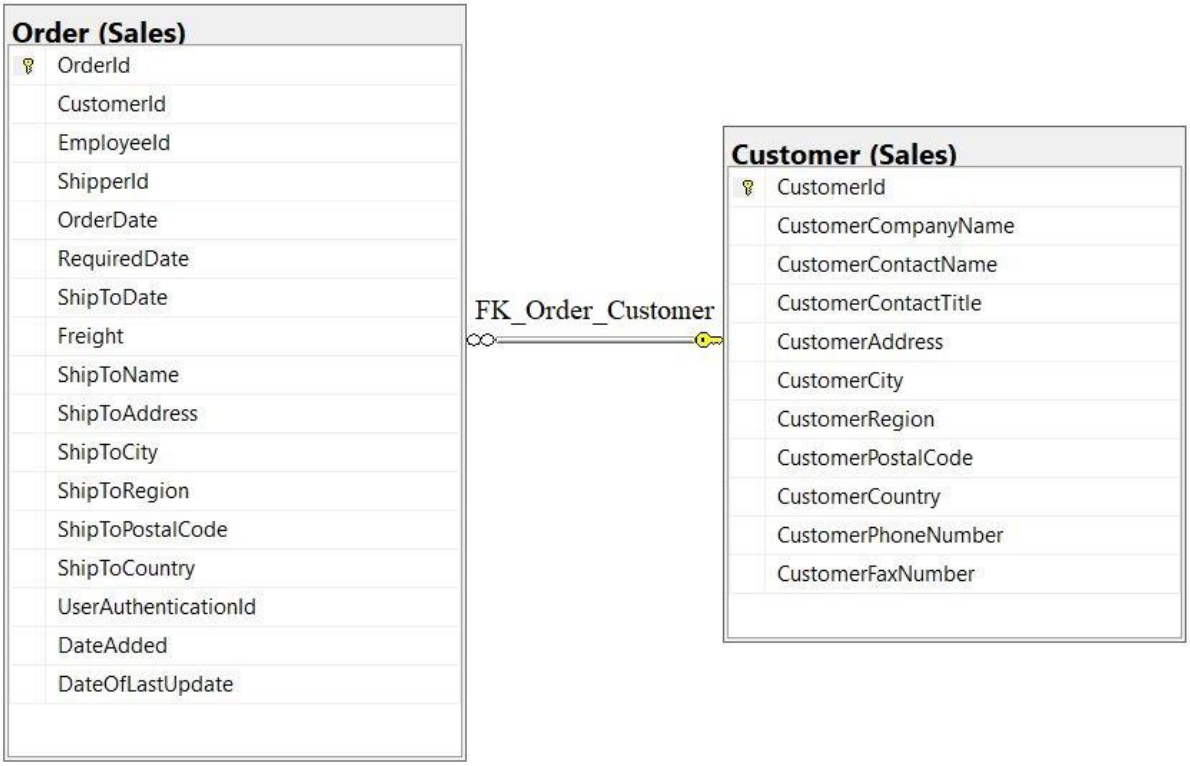


Figure 1B: Standard View Model for Proposition 1



Explanation:

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.

Figure 1C: Tables for SQL query components

Select clause

Table name:	Column name:
Sales. Order	CustomerID EmployeeID
Sales. Customer	CustomerID CustomerCity

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
```

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 (40)

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)-Micheal

Proposition 2: Return Customer ID, Order ID, Product ID, Quantity, and UnitPrice. Sorted by Customer ID

Model Diagrams:

Figure 2A: Key View Model for Proposition 2

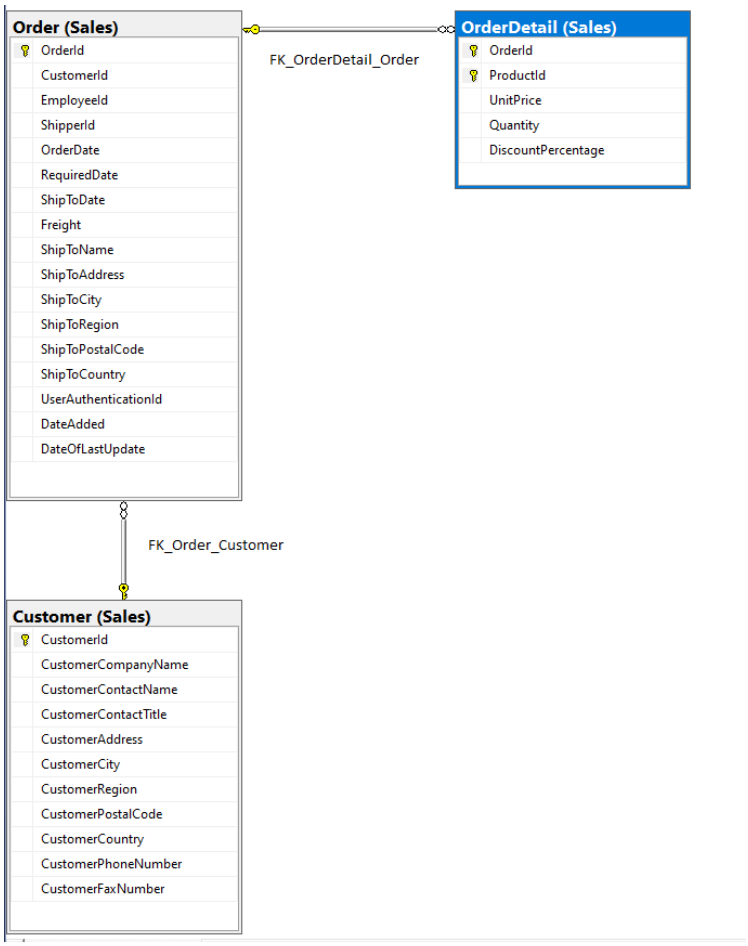
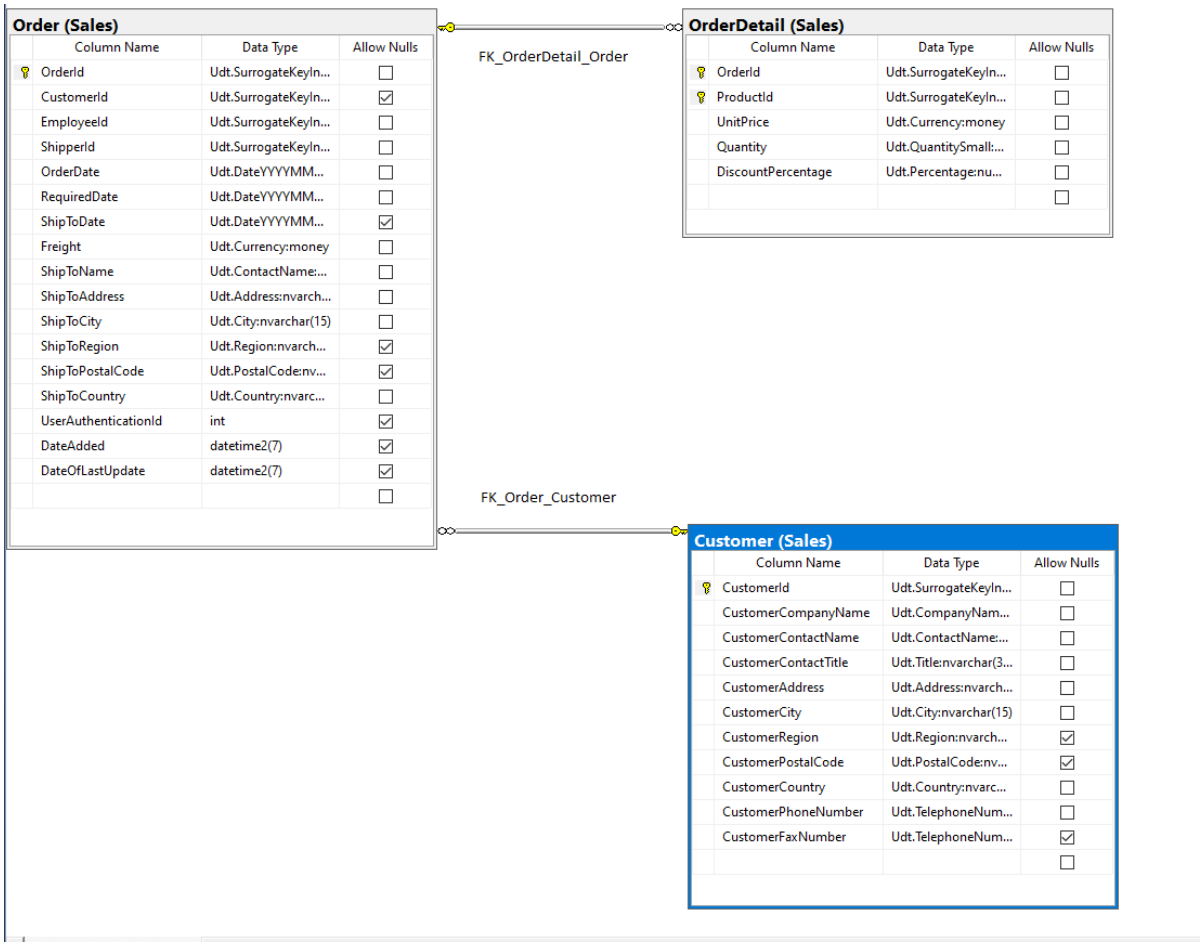


Figure 2B: Standard View Model for Proposition 2



Explanation:

Selected C.CustomerId, O.OrderId, OD.ProductId, OD.Quantity, OD.UnitPrice. Use Inner Join to combine the Sales.Order table and Sales.OrderDetails Table for the Quantity and UnitPrice. Also included in Sales.Customer for the Customer ID with the Right Outer Join

Figure 2C: Tables for SQL query components

Select clause

Table name:	Column name:
sales.order	o.orderid
sales.orderdetails	Od.productid, od.quantity, od.unitprice
Sales.customer	c.customerid

Order by (optional, only if exist)

Table name	Column name	Sort order
sales.customer	c.customerid	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
--12 Medium. Return Customer ID, Order ID, Product ID, Quantity, and UnitPrice. Sorted by Customer ID
USE Northwinds2020TSQLV6;
GO

SELECT C.CustomerId, O.OrderId, OD.ProductId, OD.Quantity, OD.UnitPrice
FROM Sales.[Order] AS O
INNER JOIN Sales.[OrderDetail] AS OD
ON O.orderid = OD.orderid
RIGHT OUTER JOIN Sales.[Customer] AS C
ON C.CustomerId = O.CustomerId
ORDER BY C.CustomerId
```

Figure 2E: Query Output for Proposition 2

100 %

Results | Messages

	CustomerId	OrderId	ProductId	Quantity	UnitPrice
1	1	10643	28	15	45.60
2	1	10643	39	21	18.00
3	1	10643	46	2	12.00
4	1	10692	63	20	43.90
5	1	10702	3	6	10.00
6	1	10702	76	15	18.00
7	1	10835	59	15	55.00
8	1	10835	77	2	13.00
9	1	10952	6	16	25.00
10	1	10952	28	2	45.60
11	1	11011	58	40	13.25
12	1	11011	71	20	21.50
13	2	10926	11	2	21.00
14	2	10926	13	10	6.00
15	2	10926	19	7	9.20
16	2	10926	72	10	34.80
17	2	10759	32	10	32.00
18	2	10625	14	3	23.25
19	2	10625	42	5	14.00

JSON:

Sample JSON Output with total number of rows returned (2,157)

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

```sql
--12 Medium. Return Customer ID, Order ID, Product ID, Quantity, and UnitPrice. Sorted by Customer ID
USE Northwinds2020TSQLV6;
GO

SELECT C.CustomerId, O.OrderId, OD.ProductId, OD.Quantity, OD.UnitPrice
FROM Sales.[Order] AS O
INNER JOIN Sales.[OrderDetail] AS OD
ON O.orderid = OD.orderid
RIGHT OUTER JOIN Sales.[Customer] AS C
ON C.CustomerId = O.CustomerId
ORDER BY C.CustomerId
for json path, root('CustomerOrders'), include_null_values;
```

Figure 2G: Formatted JSON Output for Proposition 2


```
{
 "CustomerOrders":[
  {
   "CustomerId":1,
   "OrderId":10643,
   "ProductId":28,
   "Quantity":15,
   "UnitPrice":45.6000
  },
  {
   "CustomerId":1,
   "OrderId":10643,
   "ProductId":39,
   "Quantity":21,
   "UnitPrice":18.0000
  },
  {
   "CustomerId":1,
   "OrderId":10643,
   "ProductId":46,
   "Quantity":2,
   "UnitPrice":12.0000
  },
  {
```

Proposition 3 (Best Complex)-Lindita

Proposition 3: Find caleb F carter and give me information on his properties and taxes he payed over each year.

Model Diagrams:

Figure 3A: Key View Model for Proposition 3

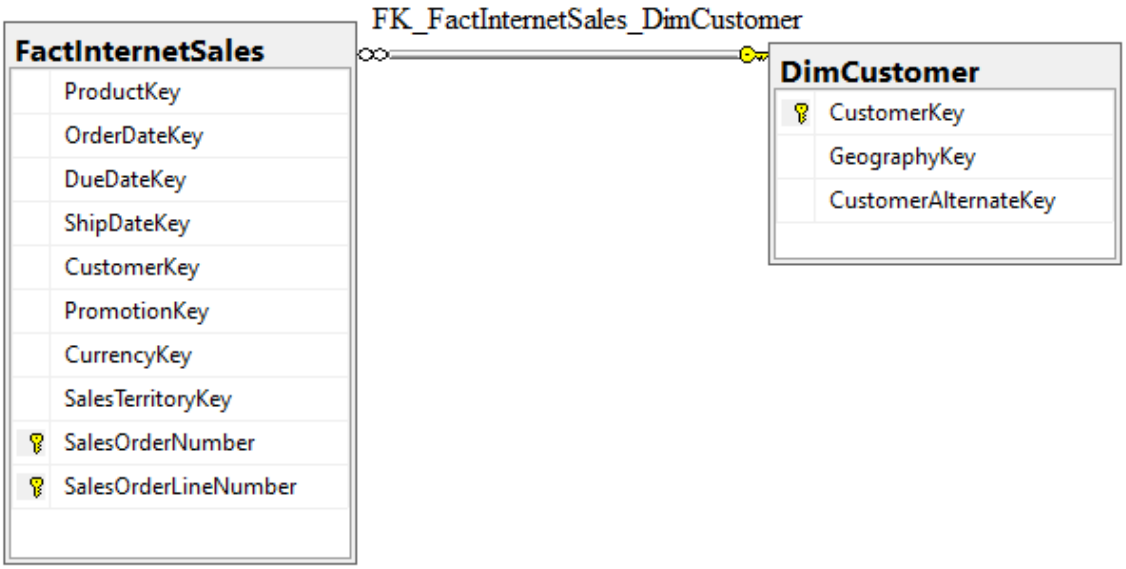
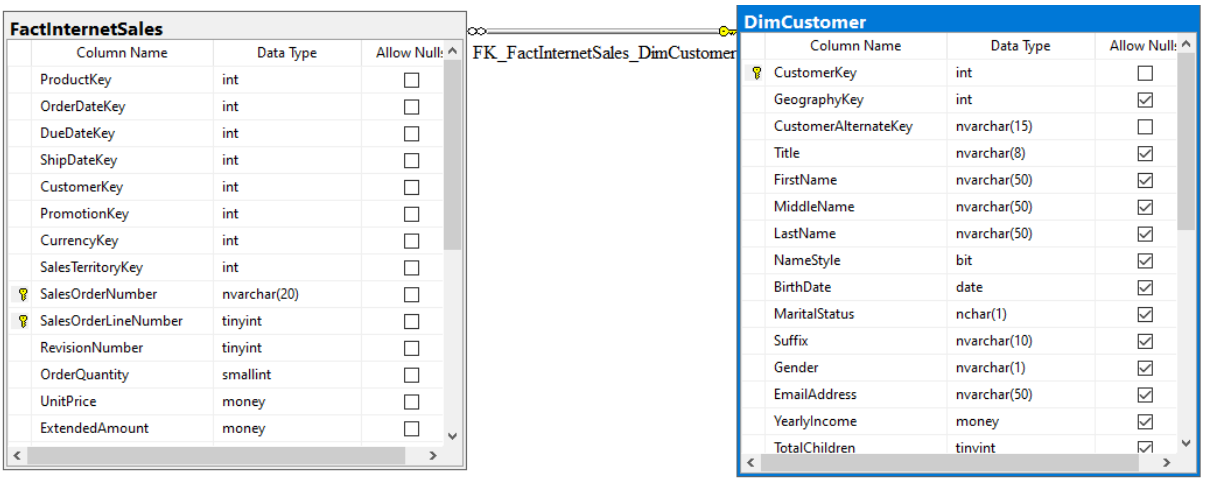


Figure 3B: Standard View Model for Proposition 3



Explanation:
Join factinternetsales with dim customer on customer id. From there you can pull out the customer key, the orderdate(year), the sum of tax amount, cars owned and if hes a homeowner. You can also get his full name from the customer table.

Figure 3C: Tables for SQL query components

Select clause

Table name:	Column name:
factinternetsales	Customerkey, orderdate as order year, sum of taxamt, cars owned, homeowner
dimcustomer	Firstname, lastname, middlename

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

Figure 3D: Formatted SQL Query for Proposition 3

```
USE AdventureWorksDW2017

DROP FUNCTION

IF EXISTS dbo.taxassets;GO
        CREATE FUNCTION dbo.taxassets (@custid AS INT)
        RETURNS TABLE
        AS
        RETURN

        SELECT fis.CustomerKey
                ,YEAR(OrderDate) AS orderyear
                ,SUM(TaxAmt) AS sumtax
                ,c.NumberCarsOwned
                ,CASE
                        WHEN c.HouseOwnerFlag = 1
                                THEN 'Yes'
                        ELSE 'No'
                        END AS homeowner
        FROM dbo.FactInternetSales AS fis
        INNER JOIN dbo.DimCustomer AS c ON c.CustomerKey = fis.CustomerKey
        WHERE fis.CustomerKey = @custid
        GROUP BY fis.CustomerKey
                ,YEAR(OrderDate)
                ,c.NumberCarsOwned
                ,c.HouseOwnerFlag
GO

DROP FUNCTION
```

```sql
IF EXISTS dbo.findcustid;GO
        CREATE FUNCTION dbo.findcustid (
                @personfn AS NVARCHAR(50)
                ,@personmn AS NVARCHAR(50)
                ,@personln AS NVARCHAR(50)
                )
        RETURNS TABLE
        AS
        RETURN

        SELECT CustomerKey
        FROM dbo.DimCustomer
        WHERE FirstName = @personfn
                AND (
                        MiddleName = @personmn
                        OR MiddleName IS NULL
                        )
                AND LastName = @personln
GO

DECLARE @custkey AS INT = (
                SELECT CustomerKey
                FROM dbo.findcustid('Caleb', 'F', 'Carter')
                )

SELECT *
FROM dbo.taxassets(@custkey);
```

Figure 3E: Query Output for Proposition 3

Results | Messages

	CustomerKey	orderyear	sumtax	NumberCarsOwned	homeowner
1	11067	2013	0.5024	2	Yes
2	11067	2014	7.9576	2	Yes

Query executed successfully. | 192.168.69.58,12001 (15.0 RTM) | sa (57) | AdventureWorksDW2017 | 00:00:00 | 2 rows

JSON:

Sample JSON Output with total number of rows returned (2)

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

```sql
USE AdventureWorksDW2017

DECLARE @custkey AS INT = (
                SELECT CustomerKey
                FROM dbo.findcustid('Caleb', 'F', 'Carter')
                )

SELECT *
FROM dbo.taxassets(@custkey)
FOR json path
        ,root('findemptax')
        ,include_null_values;
```

Figure 3G: Formatted JSON Output for Proposition 3


```
{
   "findemptax":[
      {
         "CustomerKey":11067,
         "orderyear":2013,
         "sumtax":0.5024,
         "NumberCarsOwned":2,
         "homeowner":"Yes"
      },
      {
         "CustomerKey":11067,
         "orderyear":2014,
         "sumtax":7.9576,
         "NumberCarsOwned":2,
         "homeowner":"Yes"
      }
   ]
}
```

Proposition 4 (Worst Simple)-Min Joo

Proposition 4: Amount of suppliers from each country (Northwinds2020TSQLV6)

Model Diagrams:

Figure 4A: Key View Model for Proposition 4

Figure 4B: Standard View Model for Proposition 4

Supplier (Production)

Column Name	Data Type	Allow Nulls
SupplierId	Udt.SurrogateKeyInt:int	☐
SupplierCompanyName	Udt.CompanyName:nvarch...	☐
SupplierContactName	Udt.ContactName:nvarchar...	☐
SupplierContactTitle	Udt.ContactTitle:nvarchar(...	☐
SupplierAddress	Udt.Address:nvarchar(60)	☐
SupplierCity	Udt.City:nvarchar(15)	☐
SupplierRegion	Udt.Region:nvarchar(15)	☑
SupplierPostalCode	Udt.PostalCode:nvarchar(10)	☑
SupplierCountry	Udt.Country:nvarchar(15)	☐
SupplierPhoneNumber	Udt.TelephoneNumber:nva...	☐
SupplierFaxNumber	Udt.TelephoneNumber:nva...	☑
		☐

Explanation:

The worst simple query because of how convoluted table use and select statements became with miscellaneous CTE use and output column renaming, despite how simple it could have been with only two columns from the table (Production.Supplier). D(SupplierCountry, supplierid)'s "D" table expression was never used in the query despite creation. No order to the count of suppliers.

Figure 4C: Tables for SQL query components

Select clause

Table name:	Column name:
Production.Supplier	Suppliercountry, supplierid

Query:

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

Figure 4D: Formatted SQL Query for Proposition 4

```
USE Northwinds2020TSQLV6;

SELECT SupplierCountry
        ,COUNT(DISTINCT supplierid) AS NumberSupplier
FROM (
        SELECT suppliercountry
                ,supplierid
        FROM Production.Supplier
        ) AS D(SupplierCountry, supplierid)
GROUP BY SupplierCountry
```

Figure 4E: Query Output for Proposition 4

Results | Messages

	SupplierCountry	NumberSupplier
1	Australia	2
2	Brazil	1
3	Canada	2
4	Denmark	1
5	Finland	1
6	France	3
7	Germany	3
8	Italy	2
9	Japan	2
10	Netherlands	1
11	Norway	1
12	Singapore	1
13	Spain	1
14	Sweden	2

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (77) | Northwinds2020TSQLV6 | 00:00:00 | 16 rows

JSON:

Sample JSON Output, 7 rows displayed here. Total number of rows returned (16)

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

```
USE Northwinds2020TSQLV6;

SELECT SupplierCountry
        ,COUNT(DISTINCT supplierid) AS NumberSupplier
FROM (
        SELECT suppliercountry
                ,supplierid
        FROM Production.Supplier
        ) AS D(SupplierCountry, supplierid)
GROUP BY SupplierCountry
FOR JSON path
        ,ROOT('SupplierCountry');
```

Figure 4G: Formatted JSON Output for Proposition 4


```
{
   "SupplierCountry":[
      {
         "SupplierCountry":"Australia",
         "NumberSupplier":2
      },
      {
         "SupplierCountry":"Brazil",
         "NumberSupplier":1
      },
      {
         "SupplierCountry":"Canada",
         "NumberSupplier":2
      },
      {
         "SupplierCountry":"Denmark",
         "NumberSupplier":1
      },
      {
         "SupplierCountry":"France",
         "NumberSupplier":3
      },
         "SupplierCountry":"UK",
         "NumberSupplier":2
      },
      {
         "SupplierCountry":"USA",
         "NumberSupplier":4
      }
   ]
}
```

Proposition 5 (Worst Medium)-Saqib

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

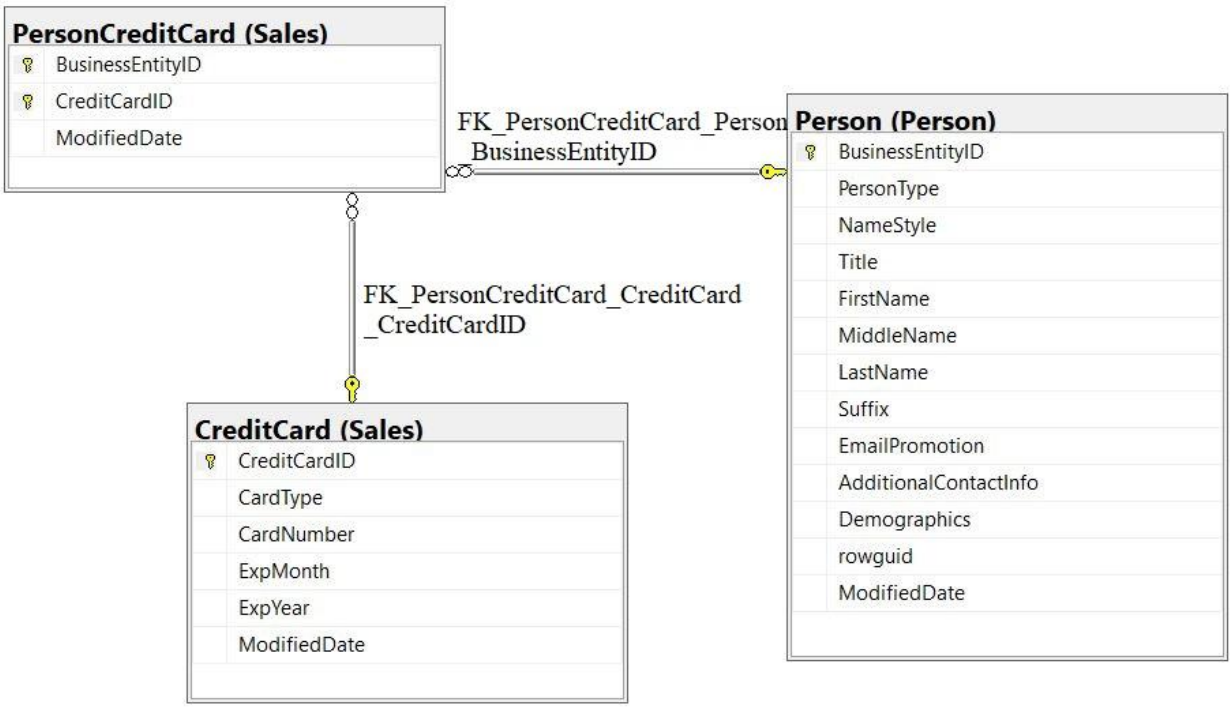
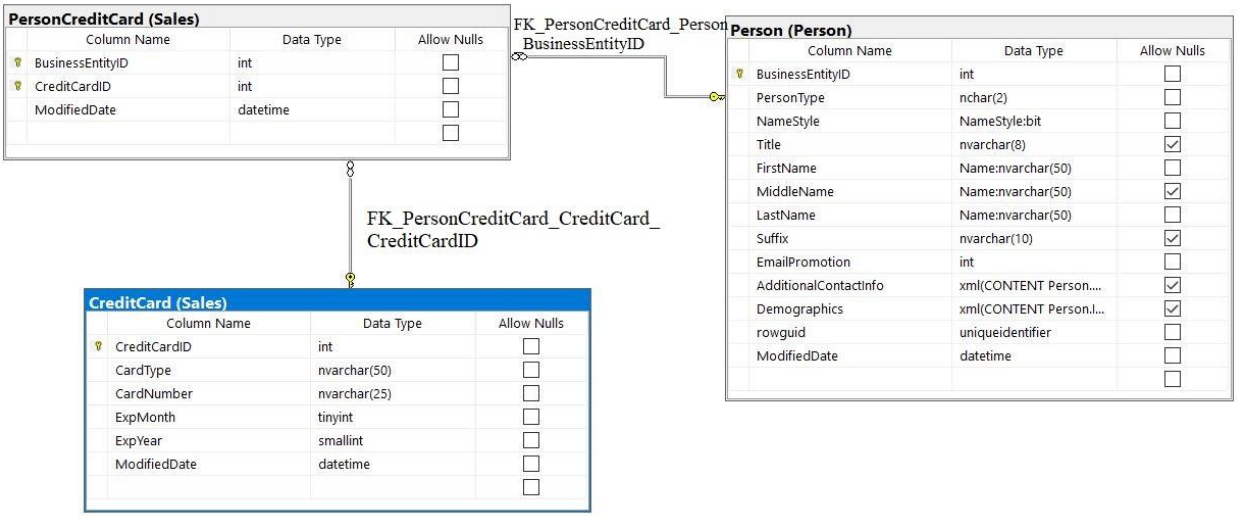


Figure 5B: Standard View Model for Proposition 5



Explanation:

This query uses two inner joins to find out the people's credit card that expires after February 2007. This is done through the use of subqueries as well as ORDER BY and WHERE statements. This can be simplified as well.

Figure 5C: Tables for SQL query components

Select clause

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

Order by (optional, only if exist)

Table name	Column name	Sort order
Sales.CreditCard	ExpMonth,ExpYear	asc

Query:

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

Figure 5D: Formatted SQL Query 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
```

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 (X)

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",
```

Proposition 6 (Worst Complex)-Lindita

Proposition 6: get customer 4s latest order, when the order was picked, the expected delivery and check the warehouse

Model Diagrams:

Figure 6A: Key View Model for Proposition 6

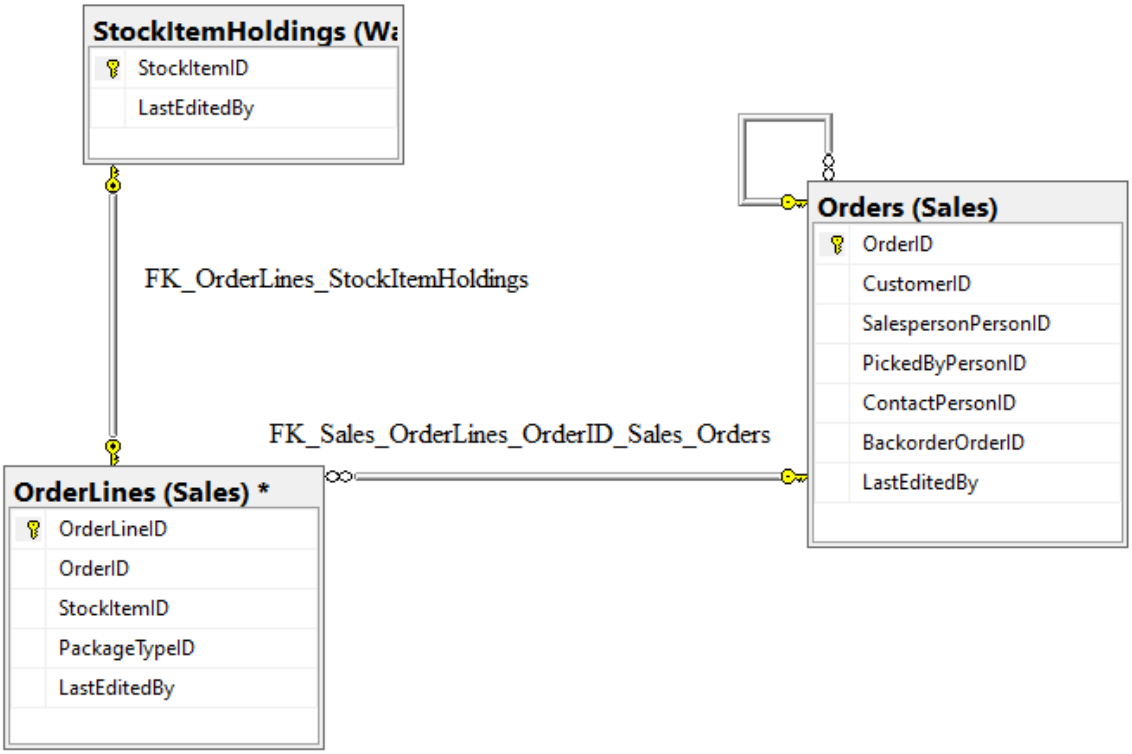
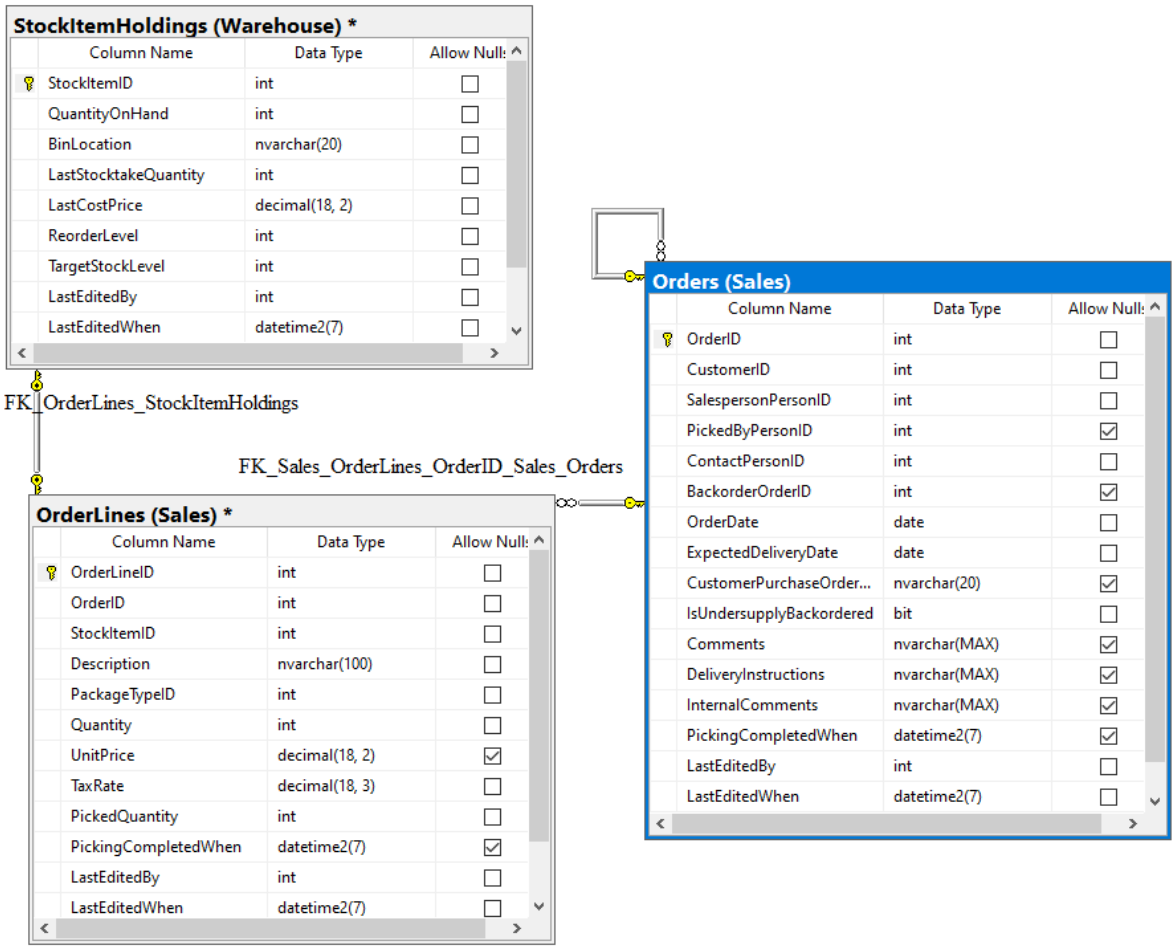


Figure 6B: Standard View Model for Proposition 6



Explanation:

Join tables orders to get the customerid, orderdate, orderid, when the picking was completed and expected delivery. Join orderlines to get stockitem id and match its order id with orders orderid. And join stock item holdings to get the quantity on hand on the stock item id with orderlines stockitemid.

Figure 6C: Tables for SQL query components

Select clause

Table name:	Column name:
orders	max(orderdate), customerid, orderdate, orderid, pickingcompletewhen,

	expecteddelivery
orderline	stockitemid
stockitemholdings	quantityonhand

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 6D: Formatted SQL Query for Proposition 6

```
USE WideWorldImporters

DECLARE @custkey AS INT = (4)
DECLARE @maxod AS DATETIME = (
                SELECT MAX(orderdate)
                FROM sales.Orders
                WHERE CustomerID = @custkey
                )

SELECT o.customerid
        ,o.OrderDate
        ,o.OrderID
        ,CAST(o.PickingCompletedWhen AS SMALLDATETIME) AS pickingcomplete
        ,o.ExpectedDeliveryDate
        ,ol.StockItemID
        ,sih.QuantityOnHand
        ,CASE
                WHEN o.ExpectedDeliveryDate < SYSDATETIME()
                        THEN 'late'
                ELSE 'ontime'
                END AS STATUS
FROM sales.Orders AS o
INNER JOIN sales.OrderLines AS ol ON ol.OrderID = o.OrderID
INNER JOIN Warehouse.StockItemHoldings AS sih ON sih.StockItemID = ol.StockItemID
WHERE o.customerid = @custkey
        AND o.OrderDate = @maxod;
```

Figure 6E: Query Output for Proposition 6

Results | Messages

	customerid	OrderDate	OrderID	pickingcomplete	ExpectedDeliveryDate	StockItemID	QuantityOnHand	status
1	4	2016-04-28	71366	2016-04-28 11:00:00	2016-04-29	22	72747	late
2	4	2016-04-28	71366	2016-04-28 11:00:00	2016-04-29	42	61075	late
3	4	2016-04-28	71366	2016-04-28 11:00:00	2016-04-29	100	277863	late

Query executed successfully. | 192.168.69.58,12001 (15.0 RTM) | sa (57) | WideWorldImporters | 00:00:00 | 3 rows

JSON:

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

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

```sql
USE WideWorldImporters

DECLARE @custkey AS INT = (4)
DECLARE @maxod AS DATETIME = (
                SELECT MAX(orderdate)
                FROM sales.Orders
                WHERE CustomerID = @custkey
                )

SELECT o.customerid
        ,o.OrderDate
        ,o.OrderID
        ,CAST(o.PickingCompletedWhen AS SMALLDATETIME) AS pickingcomplete
        ,o.ExpectedDeliveryDate
        ,ol.StockItemID
        ,sih.QuantityOnHand
        ,CASE
                WHEN o.ExpectedDeliveryDate < SYSDATETIME()
                        THEN 'late'
                ELSE 'ontime'
                END AS STATUS
FROM sales.Orders AS o
INNER JOIN sales.OrderLines AS ol ON ol.OrderID = o.OrderID
INNER JOIN Warehouse.StockItemHoldings AS sih ON sih.StockItemID = ol.StockItemID
WHERE o.customerid = @custkey
        AND o.OrderDate = @maxod
FOR json path
        ,root('CustomerOrders')
        ,include_null_values;
```

Figure 6G: Formatted JSON Output for Proposition 6

```
{
   "CustomerOrders":[
      {
         "customerid":4,
         "OrderDate":"2016-04-28",
         "OrderID":71366,
         "pickingcomplete":"2016-04-28T11:00:00",
         "ExpectedDeliveryDate":"2016-04-29",
         "StockItemID":22,
         "QuantityOnHand":72747,
         "status":"late"
      },
      {
         "customerid":4,
         "OrderDate":"2016-04-28",
         "OrderID":71366,
         "pickingcomplete":"2016-04-28T11:00:00",
         "ExpectedDeliveryDate":"2016-04-29",
         "StockItemID":42,
         "QuantityOnHand":61075,
         "status":"late"
      },
      {
         "customerid":4,
         "OrderDate":"2016-04-28",
         "OrderID":71366,
         "pickingcomplete":"2016-04-28T11:00:00",
         "ExpectedDeliveryDate":"2016-04-29",
         "StockItemID":100,
         "QuantityOnHand":277863,
         "status":"late"
      }
   ]
}
```

Proposition 7 (Improved Simple) -Min Joo

Proposition 7: Count of unique customers for days in 2016 (WideWorldImporters)

Model Diagrams:

Figure 7A: Key View Model for Proposition 7

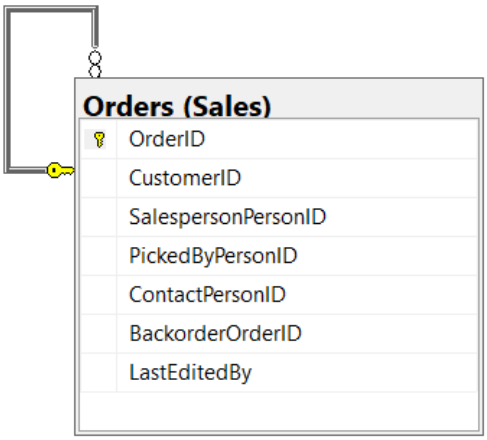
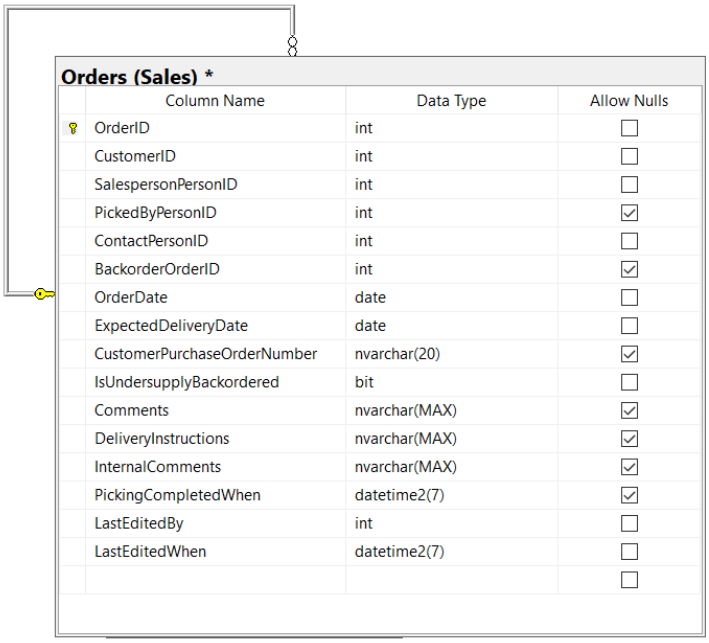


Figure 7B: Standard View Model for Proposition 7



Orders (Sales) *

Column Name	Data Type	Allow Nulls
OrderID	int	☐
CustomerID	int	☐
SalespersonPersonID	int	☐
PickedByPersonID	int	☑
ContactPersonID	int	☐
BackorderOrderID	int	☑
OrderDate	date	☐
ExpectedDeliveryDate	date	☐
CustomerPurchaseOrderNumber	nvarchar(20)	☑
IsUndersupplyBackordered	bit	☐
Comments	nvarchar(MAX)	☑
DeliveryInstructions	nvarchar(MAX)	☑
InternalComments	nvarchar(MAX)	☑
PickingCompletedWhen	datetime2(7)	☑
LastEditedBy	int	☐
LastEditedWhen	datetime2(7)	☐
		☐

Explanation:

Use a CTE to simplify the processing of relevant data to find out the count of unique customers per day in 2016. It is an improved query because only relevant columns and information from Sales.Orders are kept with the CTE. COUNT of distinct customers is done in the query. Output is ordered by date.

Figure 7C: Tables for SQL query components

Select clause

Table name:	Column name:
Sales.Orders	orderdate, customerid

Order by (optional, only if exist)

Table name	Column name	Sort order
C (Sales.Orders CTE)	days2016	asc

Query:

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

Figure 7D: Formatted SQL Query for Proposition 7

```
USE WideWorldImporters;

WITH C (
        days2016
        ,customerid
        )
AS (
        SELECT orderdate
                ,customerid
        FROM Sales.Orders
        WHERE YEAR(orderdate) = 2016
        )
SELECT days2016
        ,COUNT(DISTINCT customerid) AS numcusts
FROM C
GROUP BY days2016
ORDER BY days2016 ASC;
```

Figure 7E: Query Output for Proposition 7

Results | Messages

	days2016	numcusts
1	2016-01-01	41
2	2016-01-02	43
3	2016-01-04	84
4	2016-01-05	80
5	2016-01-06	99
6	2016-01-07	100
7	2016-01-08	76
8	2016-01-09	54
9	2016-01-11	39
10	2016-01-12	80
11	2016-01-13	62
12	2016-01-14	46
13	2016-01-15	85
14	2016-01-16	35
15	2016-01-18	82
16	2016-01-19	52

Query executed successfully. | localhost, 12001 (15.0 RTM) | sa (82) | WideWorldImporters | 00:00:00 | 130 rows

JSON:

Sample JSON Output with total number of rows returned (130), 6 displayed

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

```
USE WideWorldImporters;

WITH C (
        days2016
        ,customerid
        )
AS (
        SELECT orderdate
                ,customerid
        FROM Sales.Orders
        WHERE YEAR(orderdate) = 2016
        )
SELECT days2016
        ,COUNT(DISTINCT customerid) AS numcusts
FROM C
GROUP BY days2016
ORDER BY days2016 ASC
FOR JSON PATH
        ,ROOT('Customers2016');
```

Figure 7G: Formatted JSON Output for Proposition 7


```
{
   "Customers2016":[
      {
         "days2016":"2016-01-01",
         "numcusts":41
      },
      {
         "days2016":"2016-01-02",
         "numcusts":43
      },
      {
         "days2016":"2016-01-04",
         "numcusts":84
      },
      {
         "days2016":"2016-01-11",
         "numcusts":39
      },
      {
         "days2016":"2016-01-12",
         "numcusts":80
      },
      {
         "days2016":"2016-01-13",
         "numcusts":62
      }   ]
}
```

Proposition 8 (Improved Medium)- Saqib

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

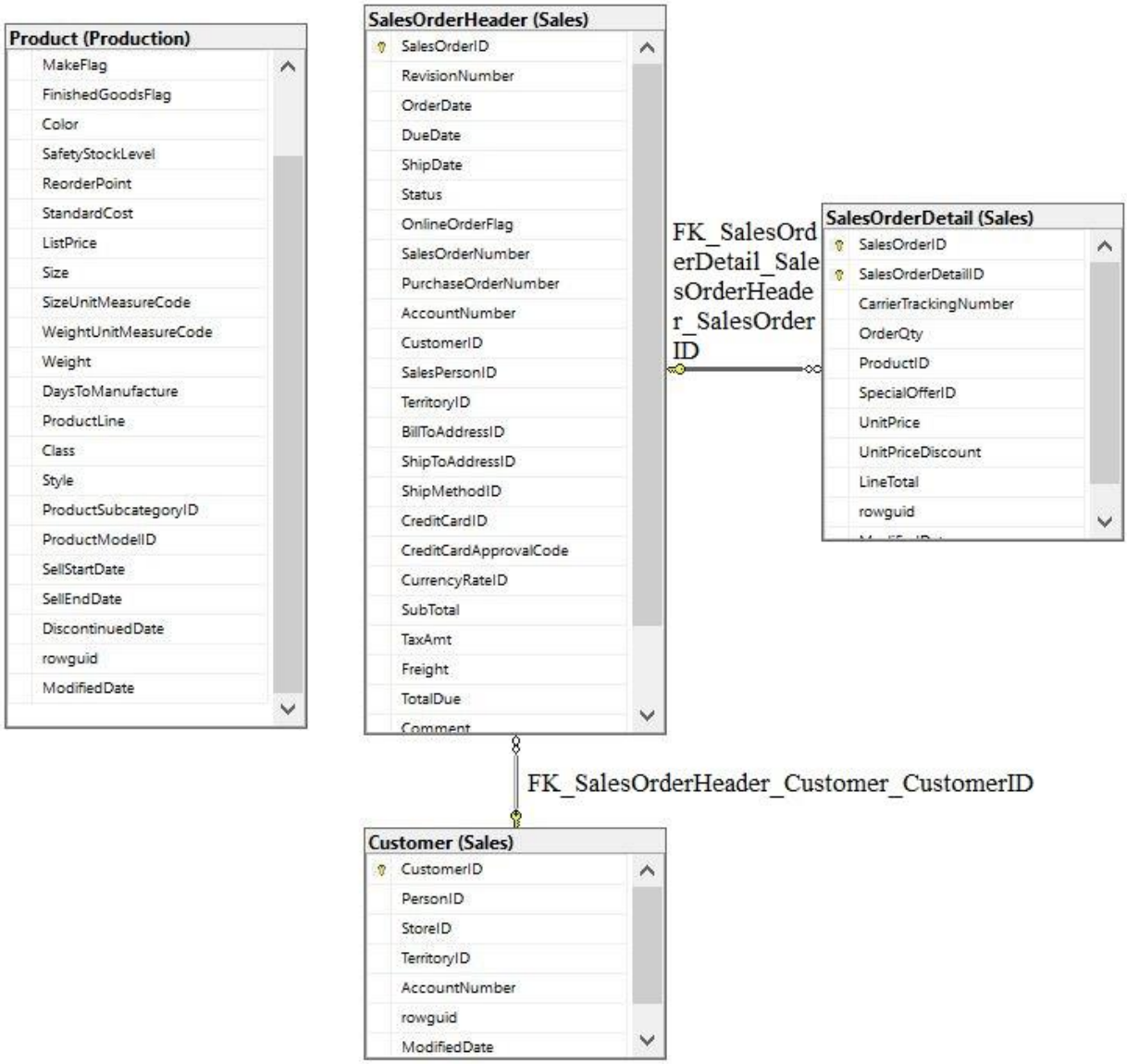
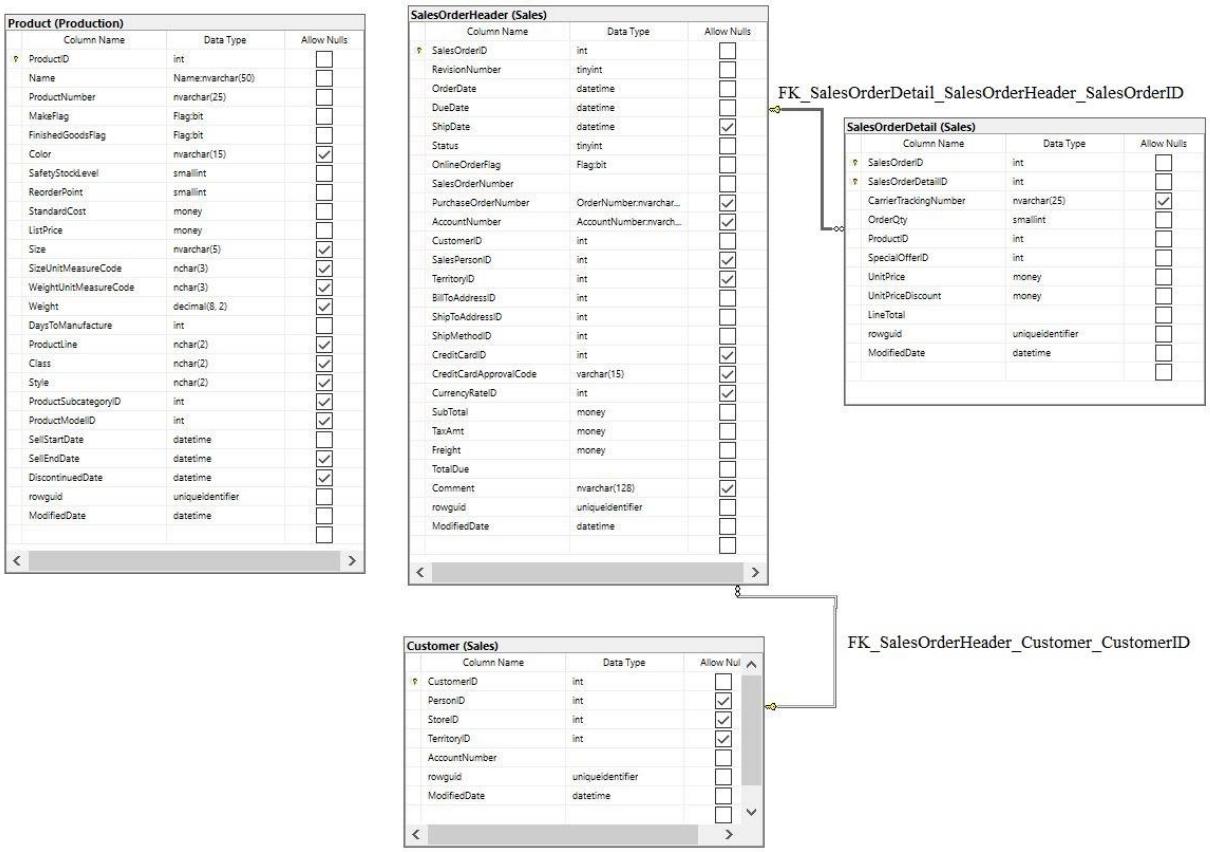


Figure 8B: Standard View Model for Proposition 8



Explanation:

summary explanation that will help the developer with the proposition.

Figure 8C: Tables for SQL query components

Select clause

Table name:	Column name:
Production.Product Sales.Customer	Weight CustomerId
Sales.SalesOrderDetail Sales.SalesOrderHeader	ProductId OrderQty

Order by (optional, only if exist)

Table name	Column name	Sort order
Sales.SalesOrderHeader	SubTotal	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;
```

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 (31,465 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,
```

Proposition 9 (Improved Complex)-Lindita

Proposition 9: find out what happened with customer 4s latest order and show when the order was picked and expected delivery and check the warehouse for quantity.

Model Diagrams:

Figure 9A: Key View Model for Proposition 9

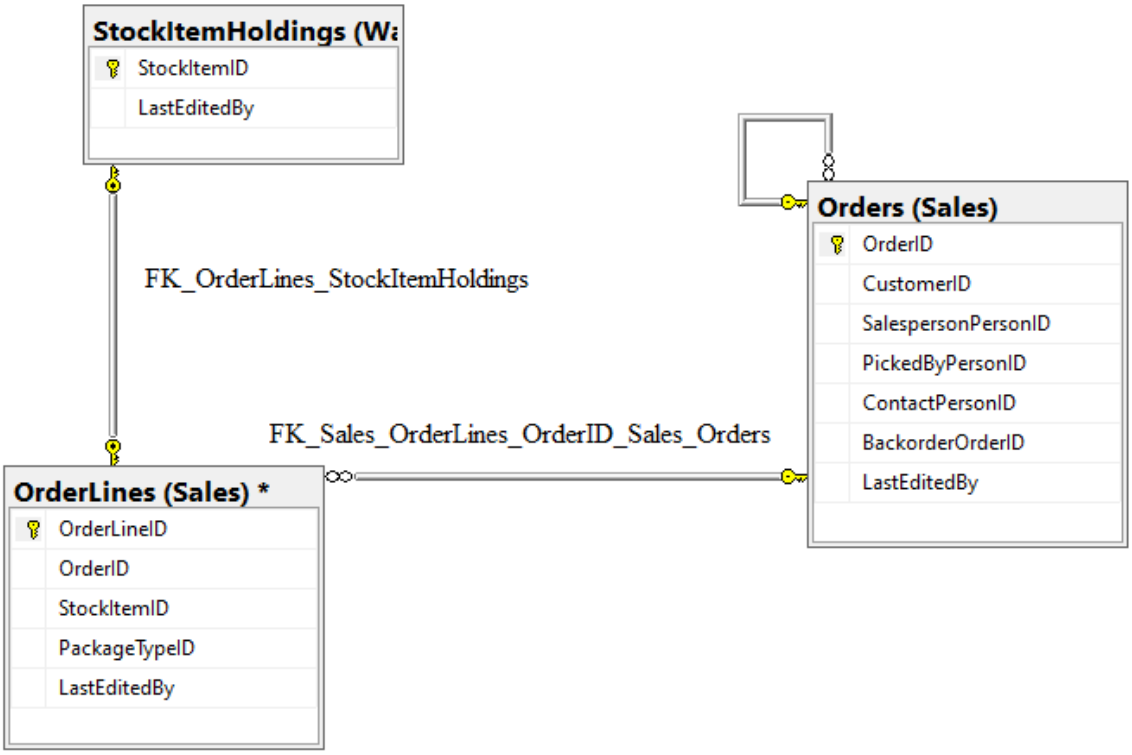
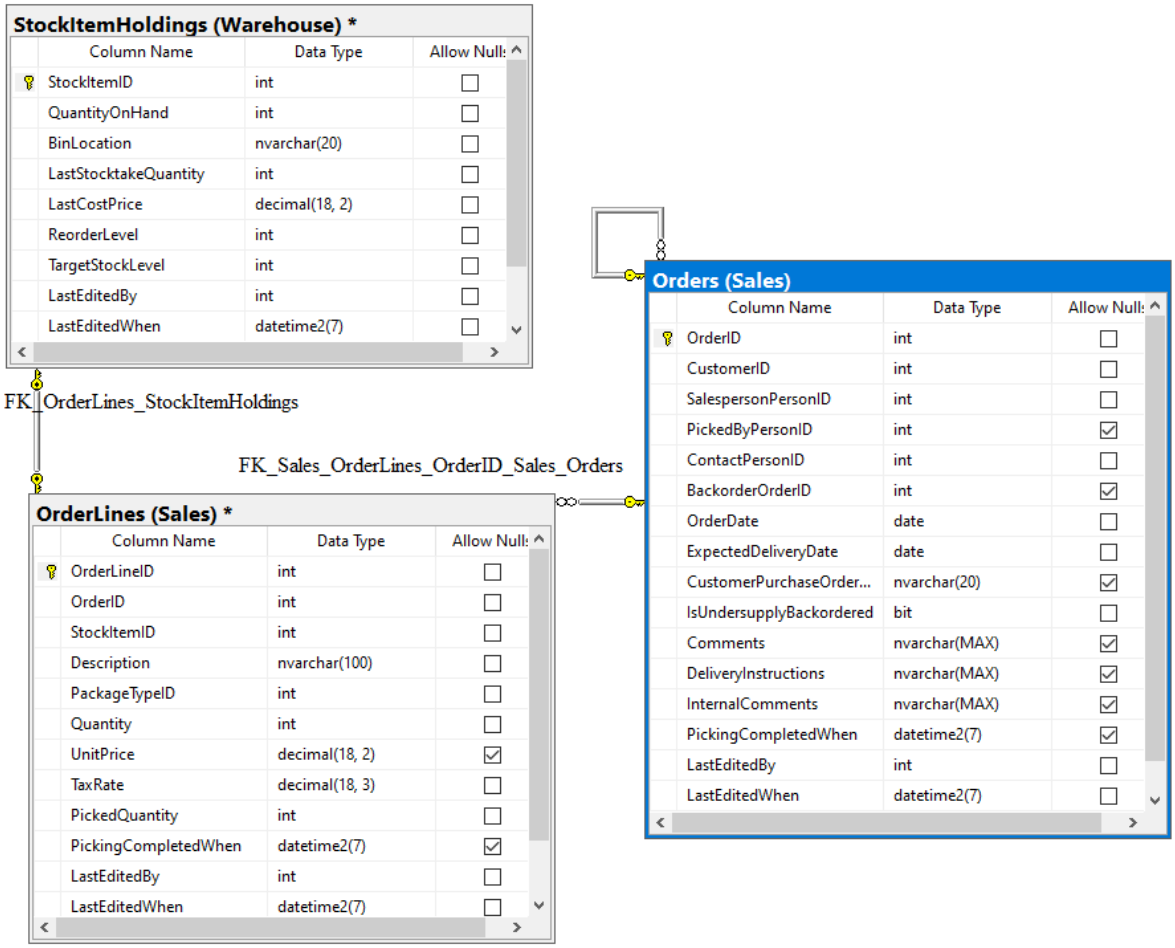


Figure 9B: Standard View Model for Proposition 9



Explanation:

Create a function where youre able to input any customer, then join tables orders orderlines and stockitem holding. To check if somethings late compare it to sysdatetime, cast picking complete as smalldate. To get the max orderdate, you can put it in the where clause instead of defining the variable.

Figure 9C: Tables for SQL query components

Select clause

Table name:	Column name:
orders	Customerid, orderdate, orderid, picking completedwhen, expecteddeliverydate

orderlines	stockitemid
stockitemholdings	quantityonhand

Query:

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

Figure 9D: Formatted SQL Query for Proposition 9

```sql
USE WideWorldImporters;


DROP FUNCTION IF EXISTS Sales.custorderdelivery;
GO
CREATE FUNCTION Sales.custorderdelivery
(
    @custkey AS INT
)
RETURNS TABLE
AS
RETURN SELECT o.CustomerID,
              o.OrderDate,
              o.OrderID,
              CAST(o.PickingCompletedWhen AS SMALLDATETIME) AS pickingcomplete,
              o.ExpectedDeliveryDate,
              ol.StockItemID,
              sih.QuantityOnHand,
              CASE
                  WHEN o.ExpectedDeliveryDate > SYSDATETIME() THEN
                      'late'
                  ELSE
                      'ontime'
              END AS status
       FROM Sales.Orders AS o
           INNER JOIN Sales.OrderLines AS ol
               ON ol.OrderID = o.OrderID
           INNER JOIN Warehouse.StockItemHoldings AS sih
               ON sih.StockItemID = ol.StockItemID
       WHERE o.CustomerID = @custkey
             AND o.OrderDate =
             (
                 SELECT MAX(OrderDate)FROM Sales.Orders WHERE CustomerID = @custkey
             );
GO

SELECT *
FROM Sales.custorderdelivery(4);
```

Figure 9E: Query Output for Proposition 9

Results | Messages

	customerid	OrderDate	OrderID	pickingcomplete	ExpectedDeliveryDate	StockItemID	QuantityOnHand	status
1	4	2016-04-28	71366	2016-04-28 11:00:00	2016-04-29	22	72747	late
2	4	2016-04-28	71366	2016-04-28 11:00:00	2016-04-29	42	61075	late
3	4	2016-04-28	71366	2016-04-28 11:00:00	2016-04-29	100	277863	late

Query executed successfully. | 192.168.69.58,12001 (15.0 RTM) | sa (57) | WideWorldImporters | 00:00:00 | 3 rows

JSON:

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

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

```sql
USE WideWorldImporters;

SELECT *
FROM Sales.custorderdelivery(4)
FOR json path
        ,root('CustomerOrderdelivery')
        ,include_null_values;
```

Figure 9G: Formatted JSON Output for Proposition 9


```
{
   "CustomerOrderdelivery":[
      {
         "customerid":4,
         "OrderDate":"2016-04-28",
         "OrderID":71366,
         "pickingcomplete":"2016-04-28T11:00:00",
         "ExpectedDeliveryDate":"2016-04-29",
         "StockItemID":22,
         "QuantityOnHand":72747,
         "status":"late"
      },
      {
         "customerid":4,
         "OrderDate":"2016-04-28",
         "OrderID":71366,
         "pickingcomplete":"2016-04-28T11:00:00",
         "ExpectedDeliveryDate":"2016-04-29",
         "StockItemID":42,
         "QuantityOnHand":61075,
         "status":"late"
      },
      {
         "customerid":4,
         "OrderDate":"2016-04-28",
         "OrderID":71366,
         "pickingcomplete":"2016-04-28T11:00:00",
         "ExpectedDeliveryDate":"2016-04-29",
         "StockItemID":100,
         "QuantityOnHand":277863,
         "status":"late"
      }
   ]
}
```