Project 1 Propositions

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

OCT 2021

ISSUED BY
10:45AM Group 4

REPRESENTATIVE
Lindita Kalaj

Table of Contents

Proposition 1 (Best Simple)

Proposition 1: Show work orders where products have been scrapped (greater than 70) and the reason as to why

Model Diagrams:

Figure 1A: Key View Model for Proposition 1

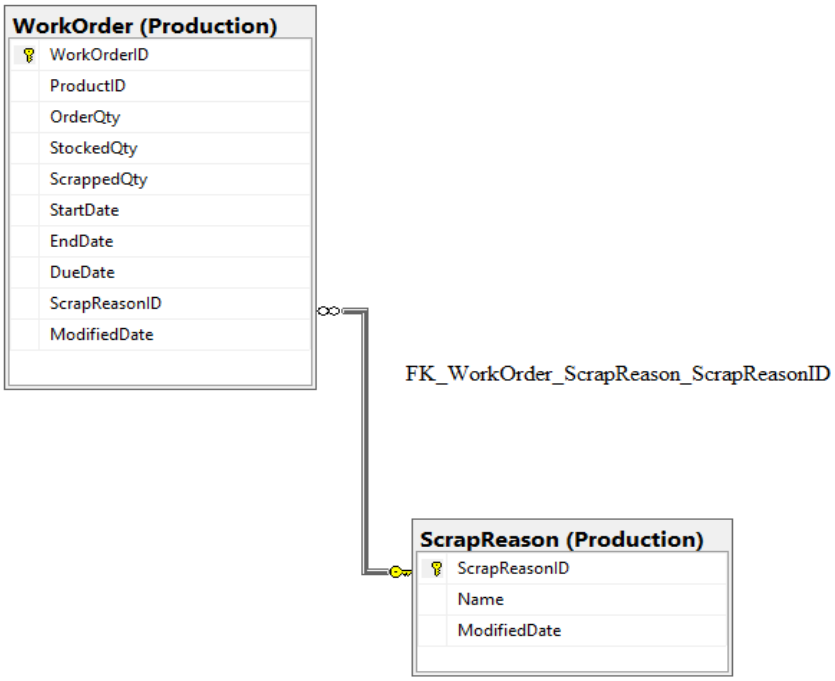
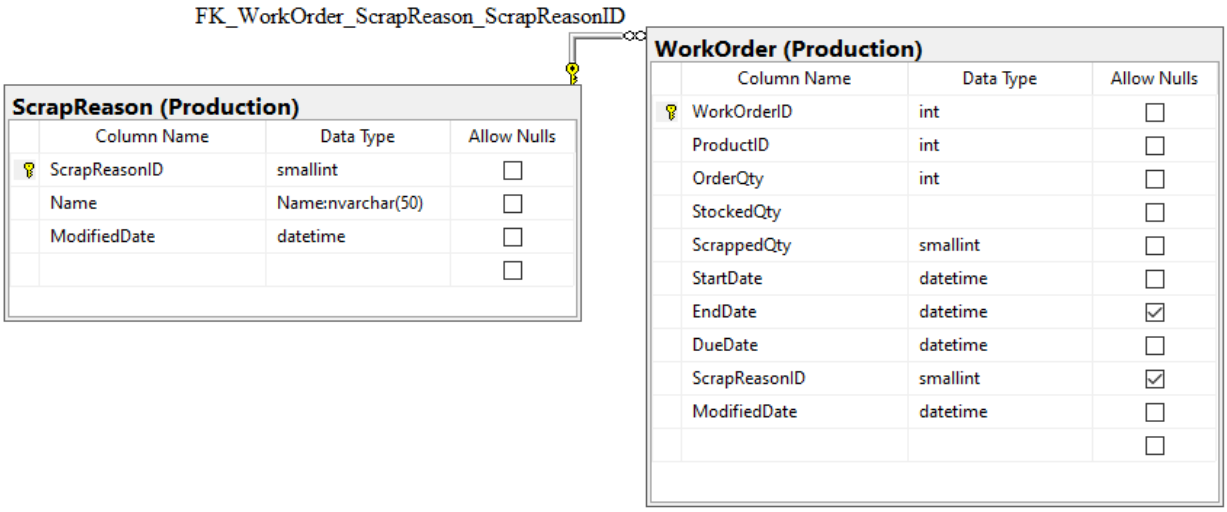


Figure 1B: Standard View Model for Proposition 1



Explanation:

Using adventure works, you must join work order to get the work order id, the quantity and the reasons id. The table to be joined with is scrap reason which will show the reason based on the foreign key relationship of scrap reason id. The quantity should also be filtered to show greater than 70 being scrapped

Figure 1C: Tables for SQL query components

Select clause

Table name:	Column name:
WorkOrder	WorkOrderID, ScrappedQty, ScrappedReasonID
ScrapReason	name

Order by (optional, only if exist)

Table name	Column name	Sort order
WorkOrder	ScrappedQTY	asc

Query:

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

Figure 1D: Formatted SQL Query for Proposition 1

```sql
USE AdventureWorks2017

SELECT wo.WorkOrderID
        ,wo.ScrappedQty
        ,wo.ScrapReasonID
        ,sr.name
FROM Production.WorkOrder AS WO
INNER JOIN production.ScrapReason AS SR ON wo.ScrapReasonID = sr.ScrapReasonID
WHERE ScrappedQty > 70
ORDER BY WO.ScrappedQty;
```

Figure 1E: Query Output for Proposition 1

Results | Messages

	WorkOrderID	ScrappedQty	ScrapReasonID	name
1	19480	72	13	Thermoform temperature too low
2	16079	73	13	Thermoform temperature too low
3	36755	78	9	Primer process failed
4	42071	83	12	Thermoform temperature too high
5	53865	83	11	Stress test failed
6	66140	83	6	Drill size too small
7	47670	84	12	Thermoform temperature too high
8	59820	85	3	Gouge in metal
9	50540	87	6	Drill size too small
10	39429	89	5	Drill size too large
11	24308	90	2	Color incorrect

Query executed successfully. | 192.168.69.58,12001 (15.0 RTM) | sa (55) | AdventureWorks2017 | 00:00:00 | 43 rows

JSON:

Sample JSON Output with total number of rows returned (43)

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

```sql
USE AdventureWorks2017

SELECT wo.WorkOrderID
        ,wo.ScrappedQty
        ,wo.ScrapReasonID
        ,sr.name
FROM Production.WorkOrder AS WO
INNER JOIN production.ScrapReason AS SR ON wo.ScrapReasonID = sr.ScrapReasonID
WHERE ScrappedQty > 70
ORDER BY WO.ScrappedQty
FOR json path
        ,root('Scrapped>70')
        ,include_null_values;
```

Figure 1G: Formatted JSON Output for Proposition 1


```
{
   "Scrapped&gt;70":[
      {
         "WorkOrderID":19480,
         "ScrappedQty":72,
         "ScrapReasonID":13,
         "name":"Thermoform temperature too low"
      },
      {
         "WorkOrderID":16079,
         "ScrappedQty":73,
         "ScrapReasonID":13,
         "name":"Thermoform temperature too low"
      },
      {
         "WorkOrderID":36755,
         "ScrappedQty":78,
         "ScrapReasonID":9,
         "name":"Primer process failed"
      },
      {
         "WorkOrderID":42071,
         "ScrappedQty":83,
         "ScrapReasonID":12,
         "name":"Thermoform temperature too high"
```

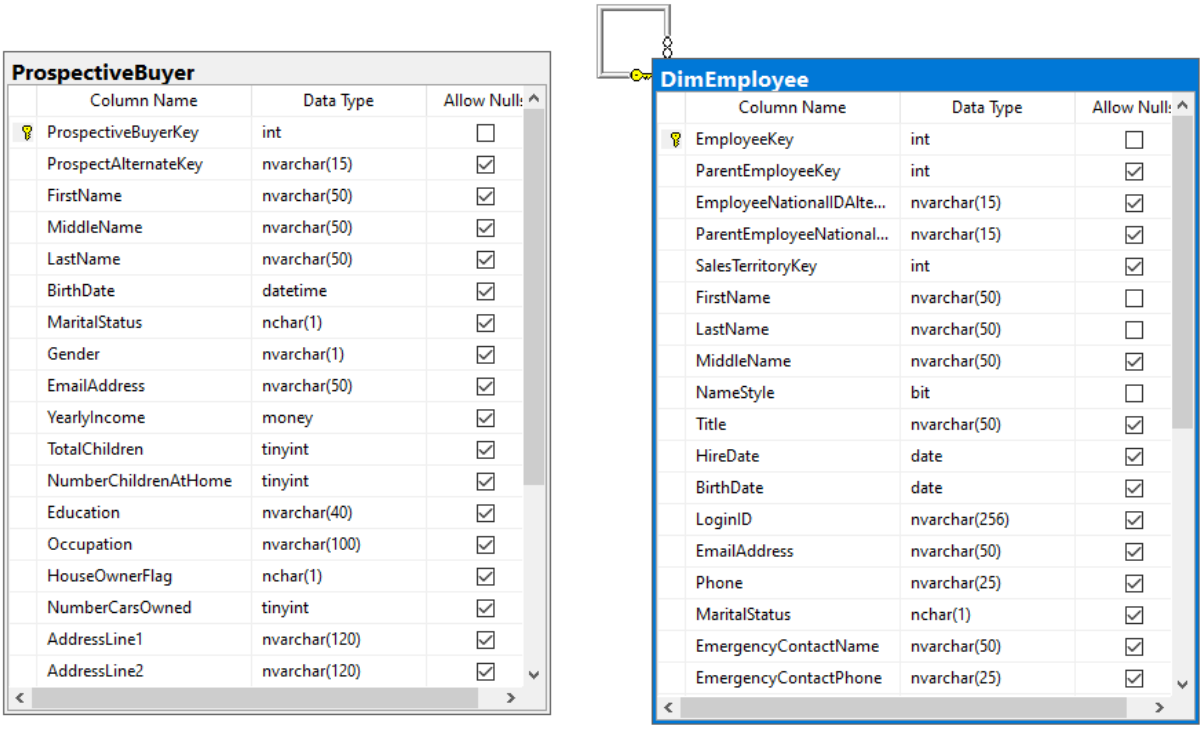
Proposition 2 (Best Medium)

Proposition 2: Show all single male potential buyers and match them up with the youngest single female employees

Model Diagrams:

Figure 2A: Key View Model for Proposition 2

Figure 2B: Standard View Model for Proposition 2



ProspectiveBuyer

Column Name	Data Type	Allow Null:
ProspectiveBuyerKey	int	☐
ProspectAlternateKey	nvarchar(15)	☑
FirstName	nvarchar(50)	☑
MiddleName	nvarchar(50)	☑
LastName	nvarchar(50)	☑
BirthDate	datetime	☑
MaritalStatus	nchar(1)	☑
Gender	nvarchar(1)	☑
EmailAddress	nvarchar(50)	☑
YearlyIncome	money	☑
TotalChildren	tinyint	☑
NumberChildrenAtHome	tinyint	☑
Education	nvarchar(40)	☑
Occupation	nvarchar(100)	☑
HouseOwnerFlag	nchar(1)	☑
NumberCarsOwned	tinyint	☑
AddressLine1	nvarchar(120)	☑
AddressLine2	nvarchar(120)	☑

DimEmployee

Column Name	Data Type	Allow Null:
EmployeeKey	int	☐
ParentEmployeeKey	int	☑
EmployeeNationalIDAlte...	nvarchar(15)	☑
ParentEmployeeNational...	nvarchar(15)	☑
SalesTerritoryKey	int	☑
FirstName	nvarchar(50)	☐
LastName	nvarchar(50)	☐
MiddleName	nvarchar(50)	☑
NameStyle	bit	☐
Title	nvarchar(50)	☑
HireDate	date	☑
BirthDate	date	☑
LoginID	nvarchar(256)	☑
EmailAddress	nvarchar(50)	☑
Phone	nvarchar(25)	☑
MaritalStatus	nchar(1)	☑
EmergencyContactName	nvarchar(50)	☑
EmergencyContactPhone	nvarchar(25)	☑

Explanation:
Join the two tables, prospective buyer and employee. Get all the information from potential buyer and filter to make sure they are single and make a yearly income over 100k. Find the youngest female employees and match them up by row number used in the select clause. There is no forein key relationship here

Figure 2C: Tables for SQL query components

Select clause

Table name:	Column name:
ProspectiveBuyer	Firstname, Lastname, yearlyincome, phone
DimEmployee	Firstname, Lastname, employee key

Order by (optional, only if exist)

Table name	Column name	Sort order
ProspectiveBuyer	YearlyIncome	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 AdventureWorksDW2017

DROP VIEW

IF EXISTS dbo.femp;GO
        CREATE VIEW dbo.femp
        AS
        SELECT FirstName
                ,LastName
                ,EmployeeKey
                ,ROW_NUMBER() OVER (
                        ORDER BY YEAR(BirthDate) DESC
                        ) AS rownum
        FROM dbo.DimEmployee
        WHERE MaritalStatus = 's'
                AND Gender = 'F'
                AND YEAR(BirthDate) > '1970'
GO

SELECT pb.firstname
        ,pb.LastName
        ,pb.YearlyIncome
        ,pb.Phone
        ,f.FirstName
        ,f.LastName
        ,f.EmployeeKey
FROM (
        SELECT FirstName
                ,LastName
                ,YearlyIncome
                ,Phone
                ,ROW_NUMBER() OVER (
                        ORDER BY YearlyIncome DESC
                        ) AS rownum

        FROM dbo.ProspectiveBuyer
        WHERE MaritalStatus = 'S'
                AND Gender = 'M'
                AND YearlyIncome > 100000
        ) AS pb
INNER JOIN dbo.femp AS f ON f.rownum = pb.rownum
ORDER BY pb.YearlyIncome DESC;
```

Figure 2E: Query Output for Proposition 2

Results | Messages

	firstname	LastName	YearlyIncome	Phone	FirstName	LastName	EmployeeKey
1	Seth	Martinez	160000.00	835-555-0181	Angela	Barbariol	123
2	Tony	Goel	130000.00	277-555-0195	Diane	Tibbott	121
3	Todd	Ye	130000.00	1 (11) 500 555-0121	Kitti	Lertpiriyasuwat	211
4	Marvin	Castro	130000.00	1 (11) 500 555-0158	Shelley	Dyck	218
5	Alfredo	Alvarez	130000.00	1 (11) 500 555-0143	Kimberly	Zimmerman	238
6	Byron	Moreno	130000.00	1 (11) 500 555-0181	Danielle	Tiedt	256
7	Connor	Carter	130000.00	1 (11) 500 555-0136	Diane	Margheim	81
8	Damien	Gao	130000.00	851-555-0174	Wendy	Kahn	73
9	Harold	Verma	130000.00	1 (11) 500 555-0184	Katie	McAskill-White	186
10	Jesus	Jimenez	130000.00	1 (11) 500 555-0121	Stephanie	Conroy	154
11	Mathew	Gutierrez	130000.00	867-555-0119	Kim	Ralls	74
12	Micheal	Jimenez	130000.00	1 (11) 500 555-0141	Susan	Metters	52

Query executed successfully. | 192.168.69.58,12001 (15.0 RTM) | sa (57) | AdventureWorksDW2017 | 00:00:00 | 26 rows

JSON:

Sample JSON Output with total number of rows returned (26)

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

```sql
SELECT pb.firstname
        ,pb.LastName
        ,pb.YearlyIncome
        ,pb.Phone
        ,f.FirstName AS empFN
        ,f.LastName AS empLN
        ,f.EmployeeKey
FROM (
        SELECT FirstName
                ,LastName
                ,YearlyIncome
                ,Phone
                ,ROW_NUMBER() OVER (
                        ORDER BY YearlyIncome DESC
                        ) AS rownum
        FROM dbo.ProspectiveBuyer
        WHERE MaritalStatus = 'S'
                AND Gender = 'M'
                AND YearlyIncome > 100000
        ) AS pb
INNER JOIN dbo.femp AS f ON f.rownum = pb.rownum
ORDER BY pb.YearlyIncome DESC
FOR json path
        ,root('MaleBuyerFemaleEmp')
        ,include_null_values;
```

Figure 2G: Formatted JSON Output for Proposition 2

```
{
   "MaleBuyerFemaleEmp":[
      {
         "firstname":"Seth",
         "LastName":"Martinez",
         "YearlyIncome":160000.0000,
         "Phone":"835-555-0181",
         "empFN":"Angela",
         "empLN":"Barbariol",
         "EmployeeKey":123
      },
      {
         "firstname":"Tony",
         "LastName":"Goel",
         "YearlyIncome":130000.0000,
         "Phone":"277-555-0195",
         "empFN":"Diane",
         "empLN":"Tibbott",
         "EmployeeKey":121
      },
      {
         "firstname":"Todd",
         "LastName":"Ye",
         "YearlyIncome":130000.0000,
         "Phone":"1 (11) 500 555-0121",
         "empFN":"Kitti",
         "empLN":"Lertpiriyasuwat",
         "EmployeeKey":211
      },
```

Proposition 3 (Best Complex)

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

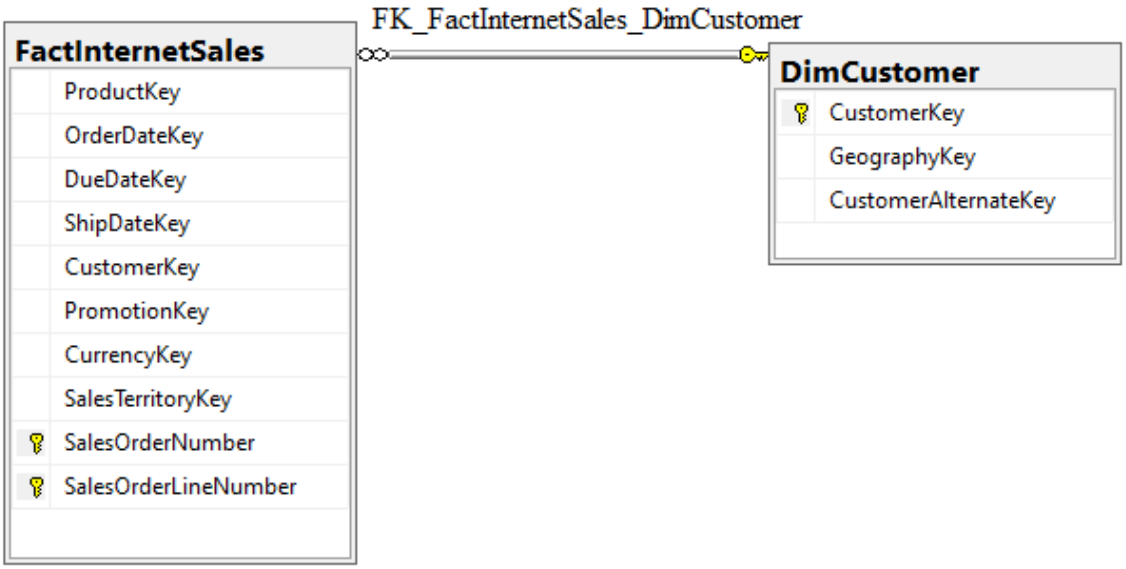
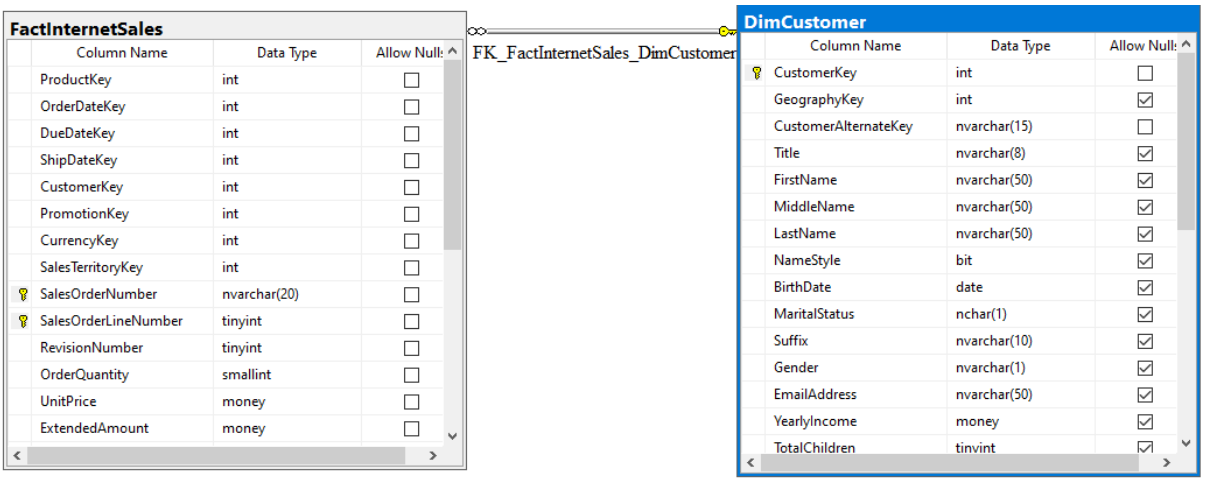


Figure 3B: Standard View Model for Proposition 3



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:
FactInernetSales	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)

Proposition 4: Show the top 5 items where you have the most stock on hand

Model Diagrams:

Figure 4A: Key View Model for Proposition 4

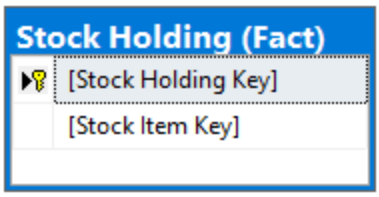


Figure 4B: Standard View Model for Proposition 4

Stock Holding (Fact)

Column Name	Data Type	Allow Nulls
[Stock Holding Key]	bigint	☐
[Stock Item Key]	int	☐
[Quantity On Hand]	int	☐
[Bin Location]	nvarchar(20)	☐
[Last Stocktake Quantity]	int	☐
[Last Cost Price]	decimal(18, 2)	☐
[Reorder Level]	int	☐
[Target Stock Level]	int	☐
[Lineage Key]	int	☐
		☐

Explanation:

Bring up stock holding table to get stockitemkey, quantity on hand and the last cost price.

Figure 4C: Tables for SQL query components

Select clause

Table name:	Column name:
Stock holding	Stockitemkey, quantityonhand, lastcostPrice

Order by (optional, only if exist)

Table name	Column name	Sort order
stockholding	quantityonhand	DESC

Query:

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

Figure 4D: Formatted SQL Query for Proposition 4

```
USE WideWorldImportersDW

SELECT TOP (5) [Stock Item Key]
        ,[Quantity On Hand]
        ,[Last Cost Price]
FROM Fact.[Stock Holding]
ORDER BY [Quantity On Hand] DESC;
```

Figure 4E: Query Output for Proposition 4

Results | Messages

	Stock Item Key	Quantity On Hand	Last Cost Price
1	639	1034169	0.00
2	637	723037	1.00
3	643	686530	1.00
4	648	654741	2.00

Query executed successfully. | 192.168.69.58,12001 (15.0 RTM) | sa (57) | WideWorldImportersDW | 00:00:00 | 5 rows

JSON:

Sample JSON Output with total number of rows returned (5)

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

```sql
USE WideWorldImportersDW

SELECT TOP (5) [Stock Item Key]
        ,[Quantity On Hand]
        ,[Last Cost Price]
FROM Fact.[Stock Holding]
ORDER BY [Quantity On Hand] DESC
FOR json path
        ,root('top5item')
        ,include_null_values;
```

Figure 4G: Formatted JSON Output for Proposition 4


```json
{
   "top5item":[
      {
         "Stock Item Key":639,
         "Quantity On Hand":1034169,
         "Last Cost Price":0.00
      },
      {
         "Stock Item Key":637,
         "Quantity On Hand":723037,
         "Last Cost Price":1.00
      },
      {
         "Stock Item Key":643,
         "Quantity On Hand":686530,
         "Last Cost Price":1.00
      },
      {
         "Stock Item Key":648,
         "Quantity On Hand":654741,
         "Last Cost Price":2.00
      },
      {
         "Stock Item Key":644,
         "Quantity On Hand":618169,
         "Last Cost Price":2.00
      }
   ]
}
```

Proposition 5 (Worst Medium)

Proposition 5: show all discontinued orders customer 5 placed

Model Diagrams:

Figure 5A: Key View Model for Proposition 5

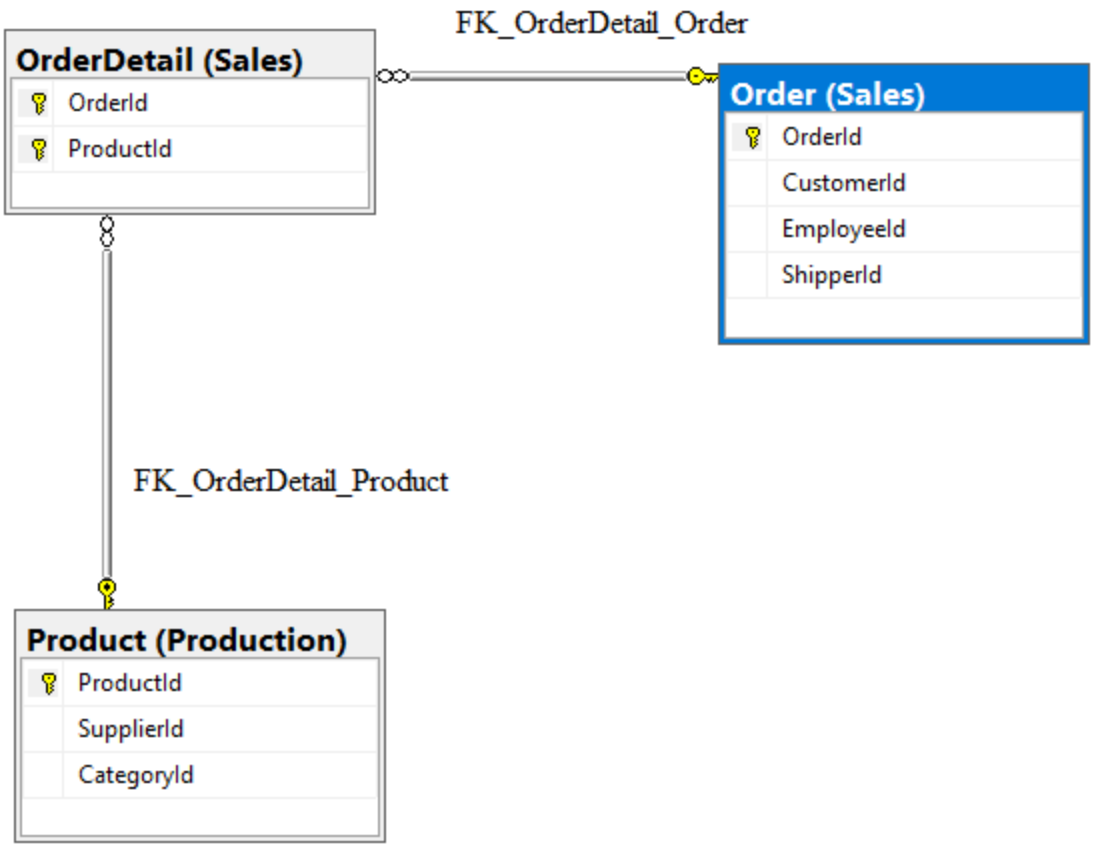
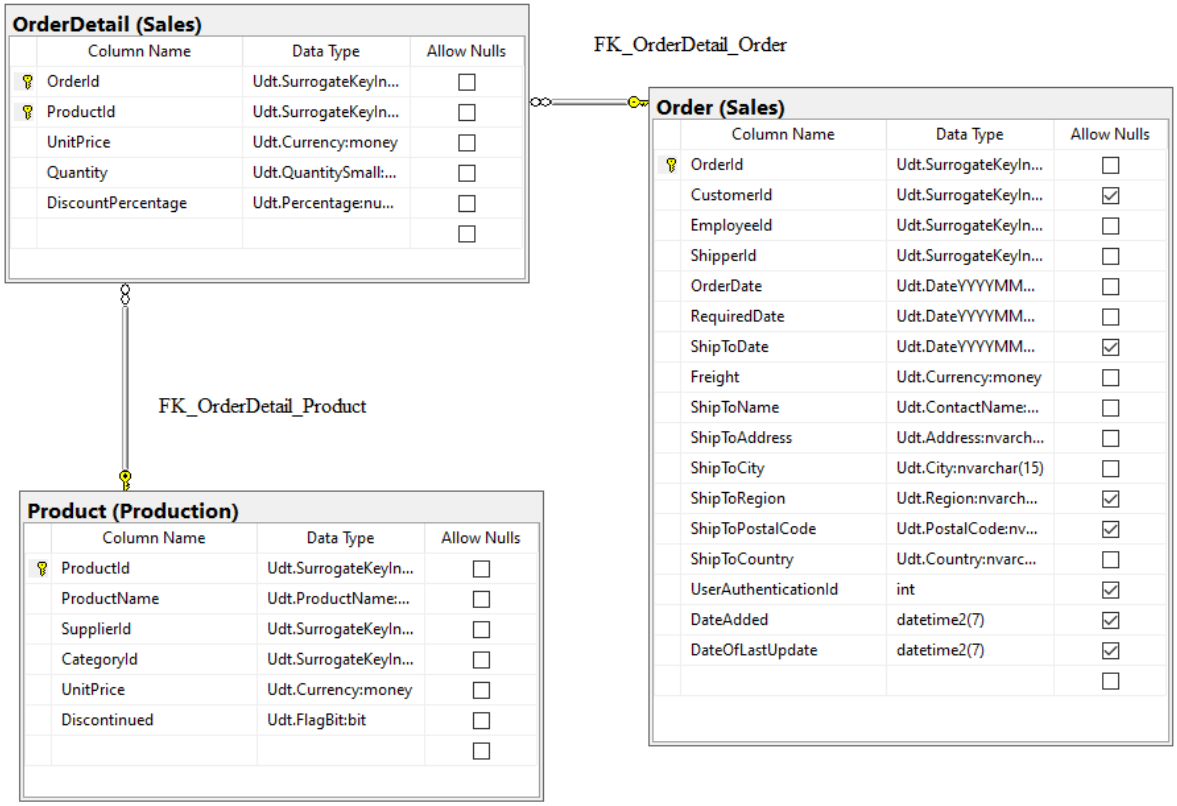


Figure 5B: Standard View Model for Proposition 5



Explanation:

Join the tables orderid to get the customer id on key orderid with orderdetail. Join with product to get discontinued on product id.

Figure 5C: Tables for SQL query components

Select clause

Table name:	Column name:
order	customerid
orderdetail	productid
product	discontinued

Query:

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

Figure 5D: Formatted SQL Query for Proposition 5

```
USE Northwinds2020TSQLV6

SELECT o.customerid
        ,od.productid
        ,count(p.Discontinued) AS discontamount
FROM sales.[Order] AS o
INNER JOIN sales.OrderDetail AS od ON od.OrderId = o.OrderId
INNER JOIN production.Product AS p ON p.ProductId = od.ProductId
WHERE p.Discontinued = 1
        AND o.CustomerId = 5
GROUP BY o.CustomerId
        ,od.ProductId;
```

Figure 5E: Query Output for Proposition 5

Results | Messages

	customerid	productid	discontamount
1	5	17	1
2	5	24	2
3	5	28	2
4	5	29	1
5	5	53	1

Query executed successfully. | 192.168.69.58,12001 (15.0 RTM) | sa (57) | Northwinds2020TSQLV6 | 00:00:00 | 5 rows

JSON:

Sample JSON Output with total number of rows returned (5)

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

```sql
USE Northwinds2020TSQLV6

SELECT o.customerid
        ,od.productid
        ,count(p.Discontinued) AS discontamount
FROM sales.[Order] AS o
INNER JOIN sales.OrderDetail AS od ON od.OrderId = o.OrderId
INNER JOIN production.Product AS p ON p.ProductId = od.ProductId
WHERE p.Discontinued = 1
        AND o.CustomerId = 5
GROUP BY o.CustomerId
        ,od.ProductId
FOR json path
        ,root('Customer5disc')
        ,include_null_values;
```

Figure 5G: Formatted JSON Output for Proposition 5


```
{
   "Customer5disc":[
      {
         "customerid":5,
         "productid":17,
         "discontamount":1
      },
      {
         "customerid":5,
         "productid":24,
         "discontamount":2
      },
      {
         "customerid":5,
         "productid":28,
         "discontamount":2
      },
      {
         "customerid":5,
         "productid":29,
         "discontamount":1
      },
      {
         "customerid":5,
         "productid":53,
         "discontamount":1
      }
   ]
}
```

Proposition 6 (Worst Complex)

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

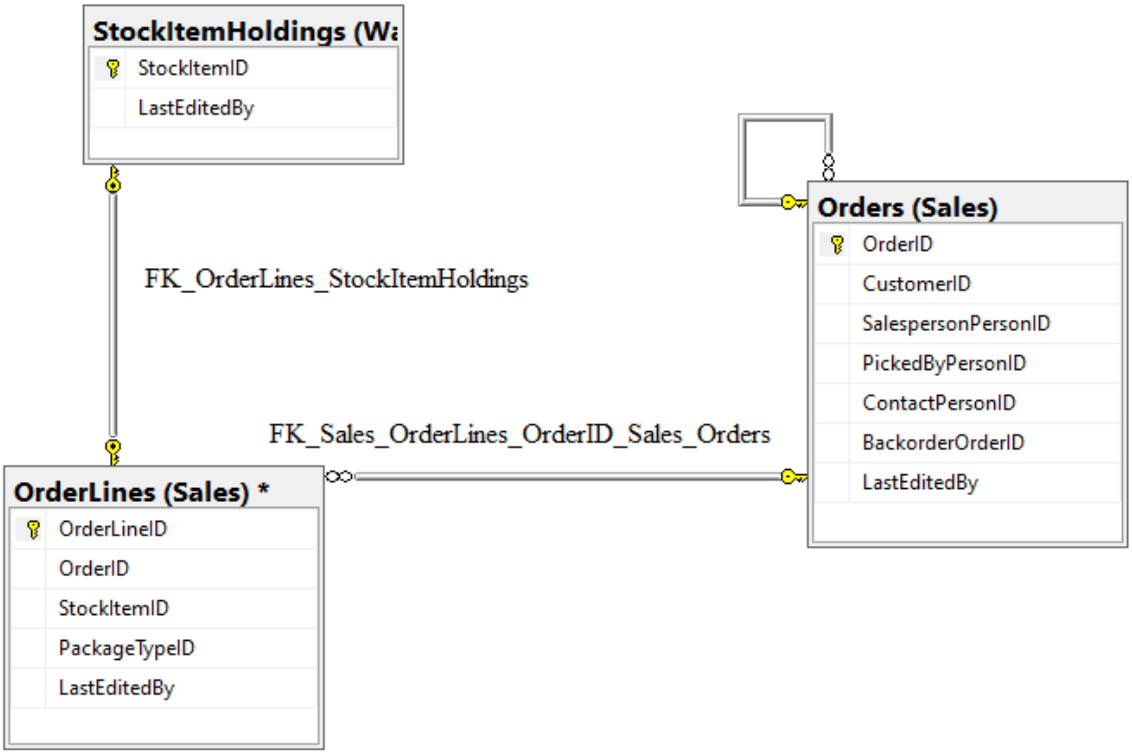
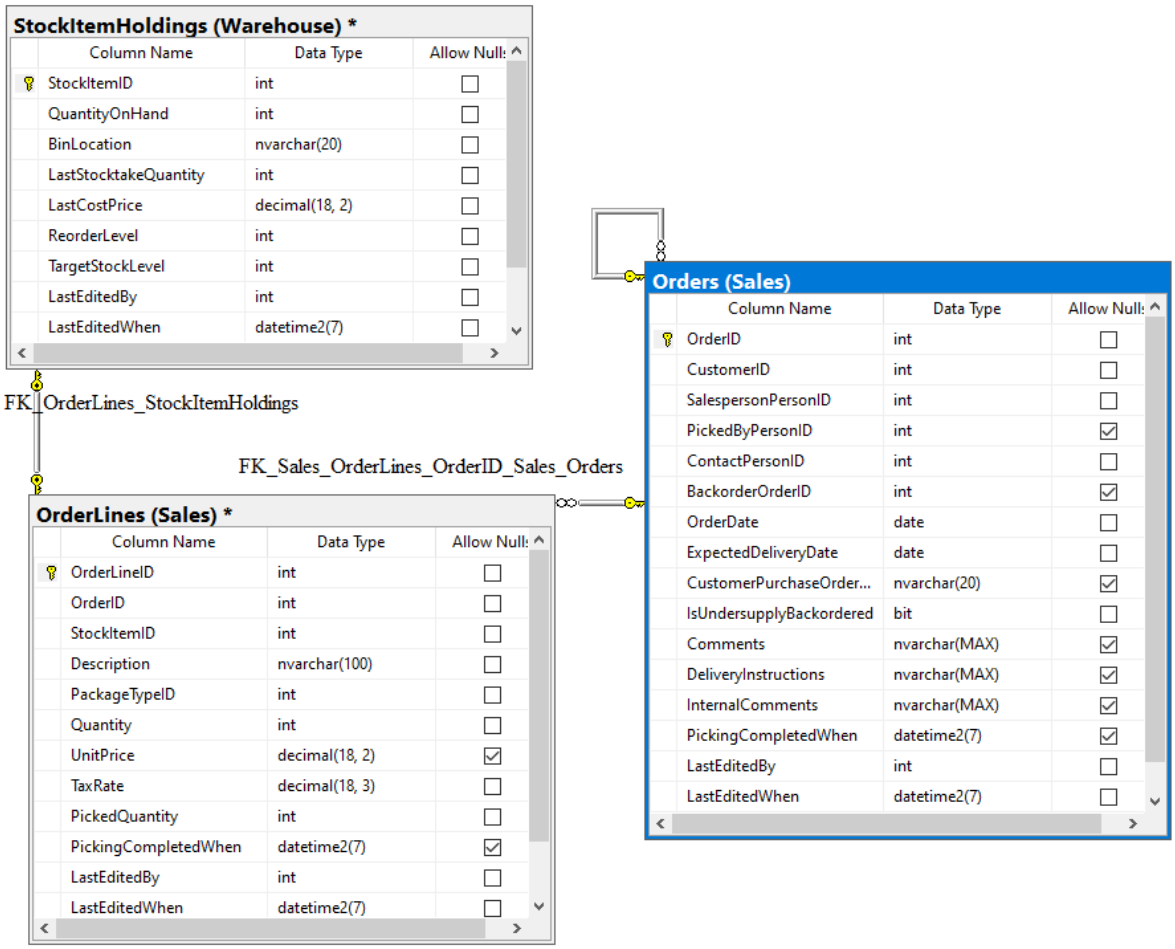


Figure 6B: Standard View Model for Proposition 6



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

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)

Proposition 7: show the top 5 items where we have the most stock on hand and the total price

Model Diagrams:

Figure 7A: Key View Model for Proposition 7

Figure 7B: Standard View Model for Proposition 7

Stock Holding (Fact)

Column Name	Data Type	Allow Nulls
[Stock Holding Key]	bigint	☐
[Stock Item Key]	int	☐
[Quantity On Hand]	int	☐
[Bin Location]	nvarchar(20)	☐
[Last Stocktake Quantity]	int	☐
[Last Cost Price]	decimal(18, 2)	☐
[Reorder Level]	int	☐
[Target Stock Level]	int	☐
[Lineage Key]	int	☐
		☐

Explanation:

Use select top 5 to get the top 5 highest quantity. Multiply quantity on hand and last cost price to get the total price. Also show orders with last cost price above 0

Figure 7C: Tables for SQL query components

Select clause

Table name:	Column name:
stockholding	Stock item key, quantity on hand, last cost price, (quantity on hand * last cost price)

Order by (optional, only if exist)

Table name	Column name	Sort order
stockholding	Quantity on hand	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 WideWorldImportersDW

SELECT TOP (5) [Stock Item Key]
        ,[Quantity On Hand]
        ,[Last Cost Price]
        ,([Quantity On Hand] * [Last Cost Price]) AS total_price
FROM Fact.[Stock Holding]
WHERE [Last Cost Price] > 0
ORDER BY [Quantity On Hand] DESC;
```

Figure 7E: Query Output for Proposition 7

Results | Messages

	Stock Item Key	Quantity On Hand	Last Cost Price	total_price
1	637	723037	1.00	723037.00
2	643	686530	1.00	686530.00
3	648	654741	2.00	1309482.00
4	644	618169	2.00	1236338.00
5	443	570360	1.00	570360.00

Query executed successfully. | 192.168.69.58,12001 (15.0 RTM) | sa (57) | WideWorldImportersDW | 00:00:00 | 5 rows

JSON:

Sample JSON Output with total number of rows returned (5)

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

```sql
USE WideWorldImportersDW

SELECT TOP (5) [Stock Item Key]
        ,[Quantity On Hand]
        ,[Last Cost Price]
        ,([Quantity On Hand] * [Last Cost Price]) AS total_price
FROM Fact.[Stock Holding]
WHERE [Last Cost Price] > 0
ORDER BY [Quantity On Hand] DESC
FOR json path
        ,root('top5moststock')
        ,include_null_values;
```

Figure 7G: Formatted JSON Output for Proposition 7

```
{
   "top5moststock":[
      {
         "Stock Item Key":637,
         "Quantity On Hand":723037,
         "Last Cost Price":1.00,
         "total_price":723037.00
      },
      {
         "Stock Item Key":643,
         "Quantity On Hand":686530,
         "Last Cost Price":1.00,
         "total_price":686530.00
      },
      {
         "Stock Item Key":648,
         "Quantity On Hand":654741,
         "Last Cost Price":2.00,
         "total_price":1309482.00
      },
      {
         "Stock Item Key":644,
         "Quantity On Hand":618169,
         "Last Cost Price":2.00,
         "total_price":1236338.00
      },
      {
         "Stock Item Key":443,
         "Quantity On Hand":570360,
         "Last Cost Price":1.00,
         "total_price":570360.00
      }
   ]
}
```

Proposition 8 (Improved Medium)

Proposition 8: show all discontinued orders customer 5 placed and show amount

Model Diagrams:

Figure 8A: Key View Model for Proposition 8

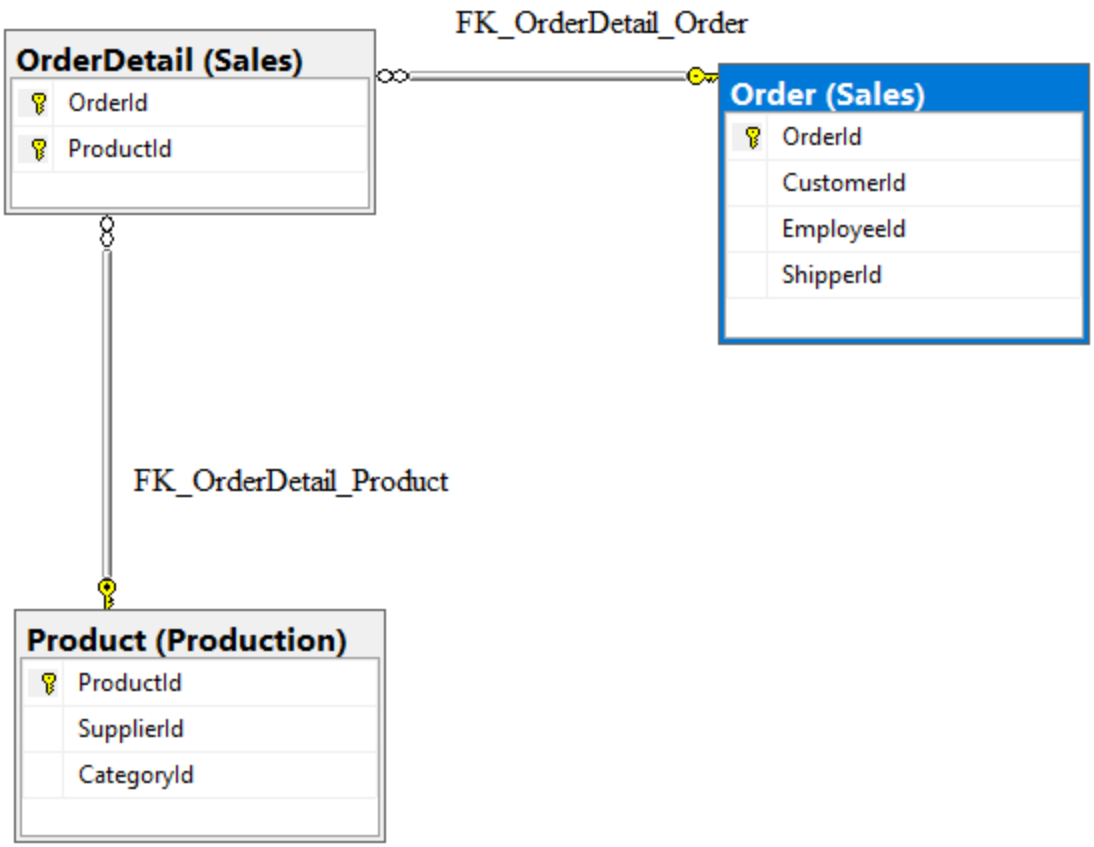
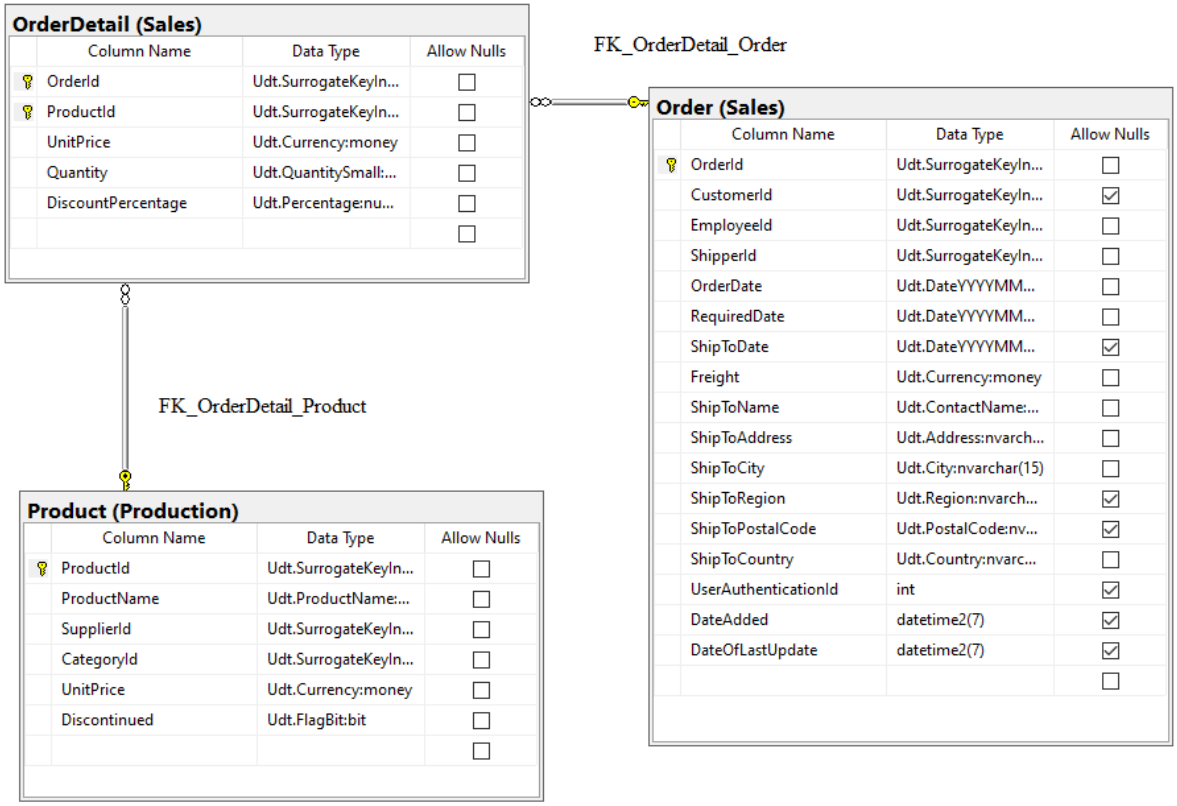


Figure 8B: Standard View Model for Proposition 8



Explanation:

Join order, orderdetail and product to bring up the customer details and see if he ordered any discontinued items along with the price, but put it in a function so youre able to check any customer

Figure 8C: Tables for SQL query components

Select clause

Table name:	Column name:
order	customerid
orderdetail	product id
product	discontinued

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 Northwinds2020TSQLV6

DROP FUNCTION

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

        SELECT o.customerid
                ,od.productid
                ,count(p.Discontinued) AS discontamount
        FROM sales.[Order] AS o
        INNER JOIN sales.OrderDetail AS od ON od.OrderId = o.OrderId
        INNER JOIN production.Product AS p ON p.ProductId = od.ProductId
        WHERE p.Discontinued = 1
                AND o.CustomerId = @custid
        GROUP BY o.CustomerId
                ,od.ProductId
GO

SELECT *
FROM sales.viewcustdiscontinue(5);
```

Figure 8E: Query Output for Proposition 8

Results | Messages

	customerid	productid	discontamount
1	5	17	1
2	5	24	2
3	5	28	2
4	5	29	1
5	5	53	1

Query executed successfully. | 192.168.69.58,12001 (15.0 RTM) | sa (57) | Northwinds2020TSQLV6 | 00:00:00 | 5 rows

JSON:

Sample JSON Output with total number of rows returned (5)

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

```sql
USE Northwinds2020TSQLV6

DROP FUNCTION

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

        SELECT o.customerid
                ,od.productid
                ,count(p.Discontinued) AS discontamount
        FROM sales.[Order] AS o
        INNER JOIN sales.OrderDetail AS od ON od.OrderId = o.OrderId
        INNER JOIN production.Product AS p ON p.ProductId = od.ProductId
        WHERE p.Discontinued = 1
                AND o.CustomerId = @custid
        GROUP BY o.CustomerId
                ,od.ProductId
GO

SELECT *
FROM sales.viewcustdiscontinue(5)
FOR json path
        ,root('Customerdiscontord')
        ,include_null_values;
```

Figure 8G: Formatted JSON Output for Proposition 8

```
{
   "Customerdiscontord":[
      {
         "customerid":5,
         "productid":17,
         "discontamount":1
      },
      {
         "customerid":5,
         "productid":24,
         "discontamount":2
      },
      {
         "customerid":5,
         "productid":28,
         "discontamount":2
      },
      {
         "customerid":5,
         "productid":29,
         "discontamount":1
      },
      {
         "customerid":5,
         "productid":53,
         "discontamount":1
      }
   ]
}
```

Proposition 9 (Improved Complex)

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

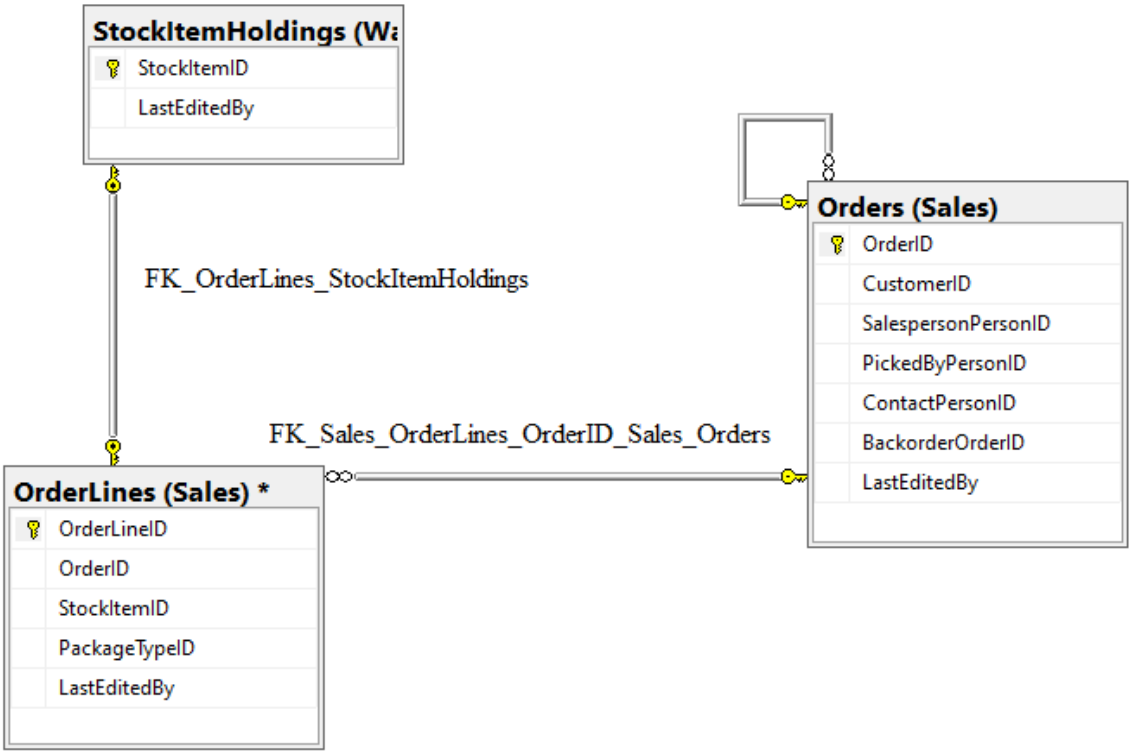
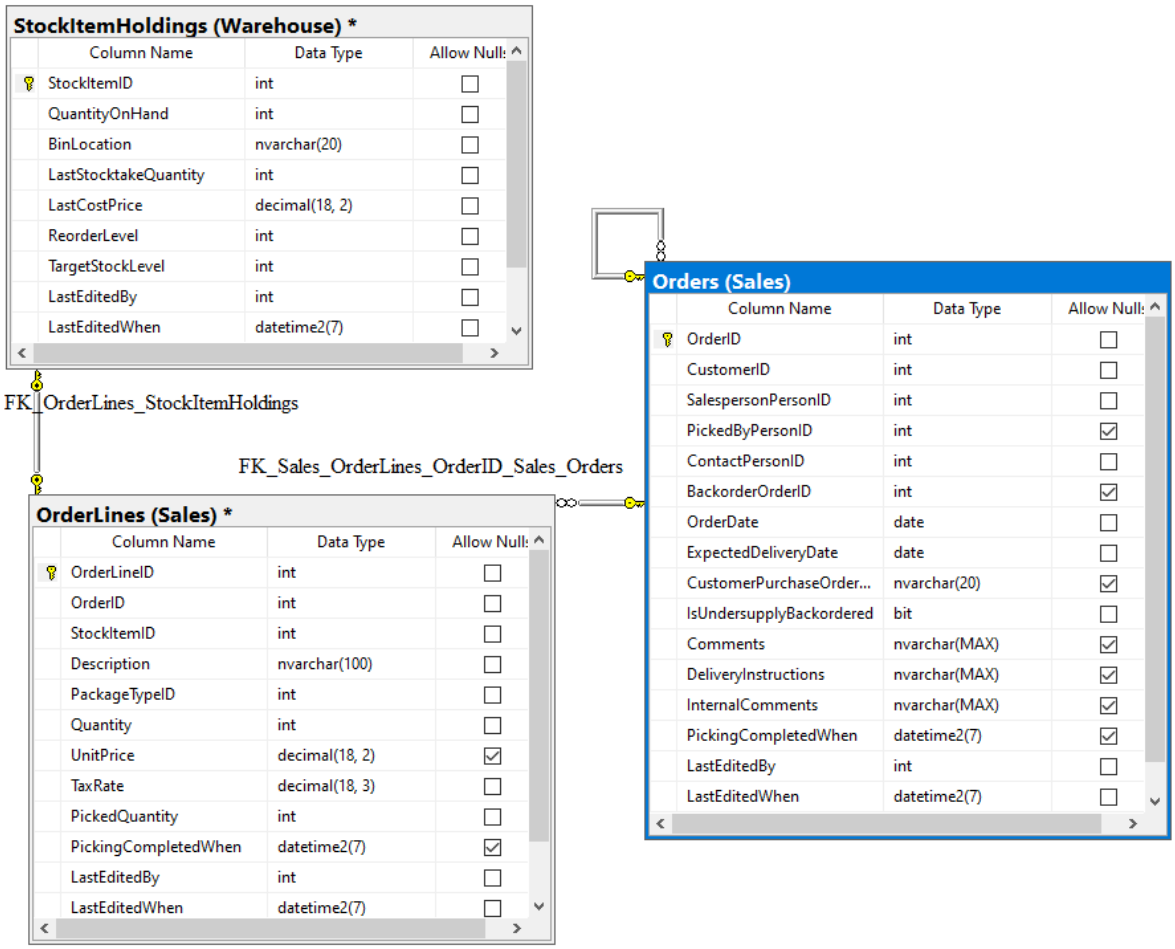


Figure 9B: Standard View Model for Proposition 9



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

```
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"
      }
   ]
}
```