Sales Analysis

The Northwind database represents a fictional wholesale trading company that sells various products to customers.

Here's an overview of the tables included in the Northwind data set:

"**Customers**": Contains information about the company's customers, including their names, addresses, contact details, and other relevant information.

"**Employees**": Stores data about the employees working for the company, such as their names, titles, birth dates, hire dates, and other related details.

"**Orders**": Contains information about the orders placed by customers, including order IDs, order dates, customer IDs, employee IDs, and other relevant details.

"**Order Details**": Stores details about individual items within each order, such as product IDs, quantities, unit prices, discounts, and other related information.

"**Products**": Contains information about the products available for sale, including product names, suppliers, categories, unit prices, and other relevant details.

"**Suppliers**": Stores data about the suppliers who provide the products to the company, including supplier names, addresses, contact details, and other related information.

"**Categories**": Contains details about the categories to which the products belong, such as category names and descriptions.

"**Shippers**": Stores information about the shipping companies used by the company, including shipper names, phone numbers, and other related details.

"**Employees Territories**": Represents the relationship between employees and territories, linking each employee to the territories they are responsible for.

"**Region**": Contains information about different regions, such as region names.

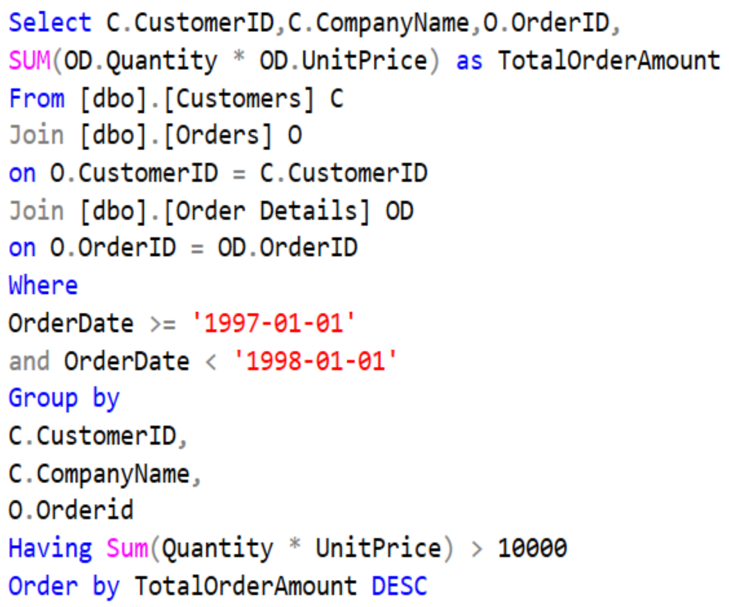
"**Territories**": Stores details about the territories covered by the company, including territory names and region IDs

Problem 1:

We want to send all of our high-value customers a special VIP gift. We're defining high-value customers as those who've made at least 1 order with a total value (not including the discount) equal to $10,000 or more. We only want to consider orders made in the year 2016.

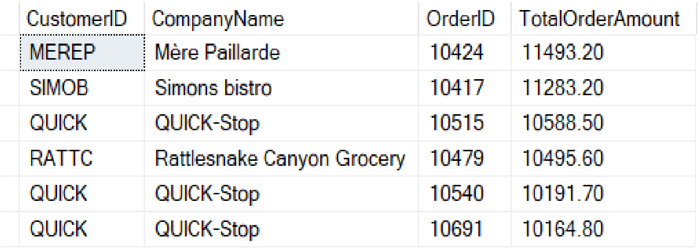
Solution: Three Tables are used: Customers, Orders, and Order Details

Code



```
Select C.CustomerID,C.CompanyName,O.OrderID,
SUM(OD.Quantity * OD.UnitPrice) as TotalOrderAmount
From [dbo].[Customers] C
Join [dbo].[Orders] O
on O.CustomerID = C.CustomerID
Join [dbo].[Order Details] OD
on O.OrderID = OD.OrderID
Where
OrderDate >= '1997-01-01'
and OrderDate < '1998-01-01'
Group by
C.CustomerID,
C.CompanyName,
O.Orderid
Having Sum(Quantity * UnitPrice) > 10000
Order by TotalOrderAmount DESC
```

Output

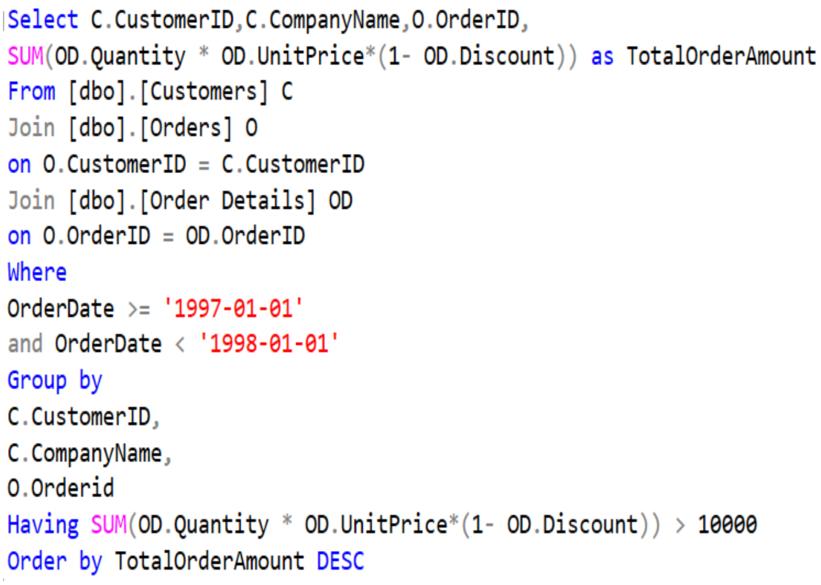


CustomerID	CompanyName	OrderID	TotalOrderAmount
MEREP	Mère Paillarde	10424	11493.20
SIMOB	Simons bistro	10417	11283.20
QUICK	QUICK-Stop	10515	10588.50
RATTC	Rattlesnake Canyon Grocery	10479	10495.60
QUICK	QUICK-Stop	10540	10191.70
QUICK	QUICK-Stop	10691	10164.80

Problem 2:
Change the above query to use the discount when calculating high-value customers. Order by the total amount which includes the discount.

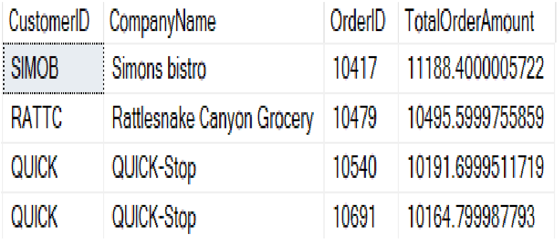
Solution: Three Tables are used: Customers, Orders, and Order Details

Code



```
Select C.CustomerID,C.CompanyName,O.OrderID,
SUM(OD.Quantity * OD.UnitPrice*(1- OD.Discount)) as TotalOrderAmount
From [dbo].[Customers] C
Join [dbo].[Orders] O
on O.CustomerID = C.CustomerID
Join [dbo].[Order Details] OD
on O.OrderID = OD.OrderID
Where
OrderDate >= '1997-01-01'
and OrderDate < '1998-01-01'
Group by
C.CustomerID,
C.CompanyName,
O.Orderid
Having SUM(OD.Quantity * OD.UnitPrice*(1- OD.Discount)) > 10000
Order by TotalOrderAmount DESC
```

Output



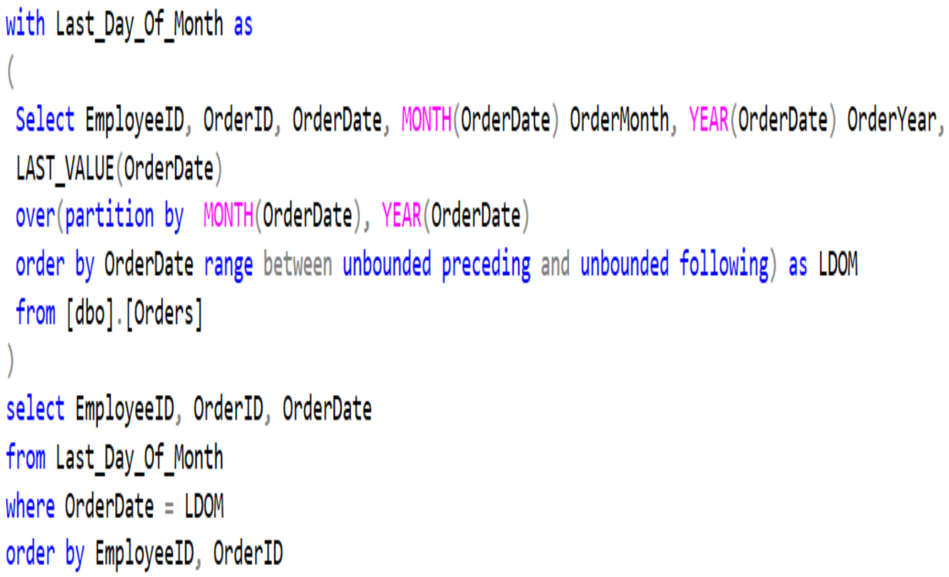
CustomerID	CompanyName	OrderID	TotalOrderAmount
SIMOB	Simons bistro	10417	11188.4000005722
RATTC	Rattlesnake Canyon Grocery	10479	10495.5999755859
QUICK	QUICK-Stop	10540	10191.6999511719
QUICK	QUICK-Stop	10691	10164.799987793

Problem 3:
At the end of the month, salespeople are likely to try much harder to get orders, to meet their month-end quotas. Show all orders made on the last day of the month. Order by EmployeeID and OrderID

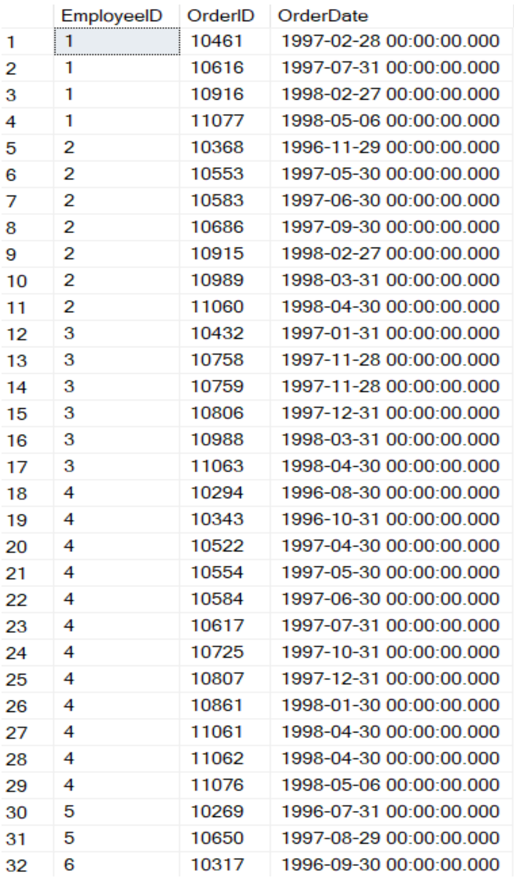
Solution: Orders Table is Used

Code

Output



```
with Last_Day_Of_Month as
(
 Select EmployeeID, OrderID, OrderDate, MONTH(OrderDate) OrderMonth, YEAR(OrderDate) OrderYear,
 LAST_VALUE(OrderDate)
 over(partition by  MONTH(OrderDate), YEAR(OrderDate)
 order by OrderDate range between unbounded preceding and unbounded following) as LDOM
 from [dbo].[Orders]
)
select EmployeeID, OrderID, OrderDate
from Last_Day_Of_Month
where OrderDate = LDOM
order by EmployeeID, OrderID
```



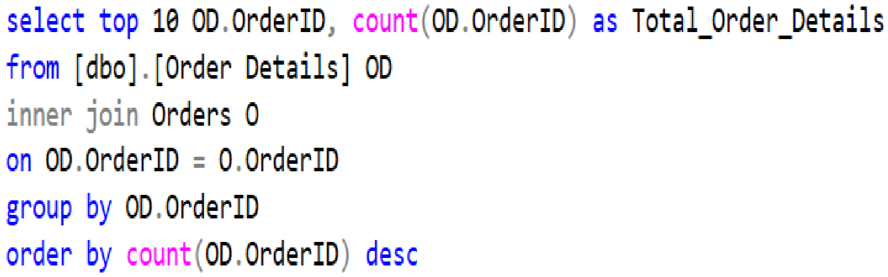
	EmployeeID	OrderID	OrderDate
1	1	10461	1997-02-28 00:00:00.000
2	1	10616	1997-07-31 00:00:00.000
3	1	10916	1998-02-27 00:00:00.000
4	1	11077	1998-05-06 00:00:00.000
5	2	10368	1996-11-29 00:00:00.000
6	2	10553	1997-05-30 00:00:00.000
7	2	10583	1997-06-30 00:00:00.000
8	2	10686	1997-09-30 00:00:00.000
9	2	10915	1998-02-27 00:00:00.000
10	2	10989	1998-03-31 00:00:00.000
11	2	11060	1998-04-30 00:00:00.000
12	3	10432	1997-01-31 00:00:00.000
13	3	10758	1997-11-28 00:00:00.000
14	3	10759	1997-11-28 00:00:00.000
15	3	10806	1997-12-31 00:00:00.000
16	3	10988	1998-03-31 00:00:00.000
17	3	11063	1998-04-30 00:00:00.000
18	4	10294	1996-08-30 00:00:00.000
19	4	10343	1996-10-31 00:00:00.000
20	4	10522	1997-04-30 00:00:00.000
21	4	10554	1997-05-30 00:00:00.000
22	4	10584	1997-06-30 00:00:00.000
23	4	10617	1997-07-31 00:00:00.000
24	4	10725	1997-10-31 00:00:00.000
25	4	10807	1997-12-31 00:00:00.000
26	4	10861	1998-01-30 00:00:00.000
27	4	11061	1998-04-30 00:00:00.000
28	4	11062	1998-04-30 00:00:00.000
29	4	11076	1998-05-06 00:00:00.000
30	5	10269	1996-07-31 00:00:00.000
31	5	10650	1997-08-29 00:00:00.000
32	6	10317	1996-09-30 00:00:00.000

Problem 4:
Show the 10 orders with the most line items, in order of total line items.

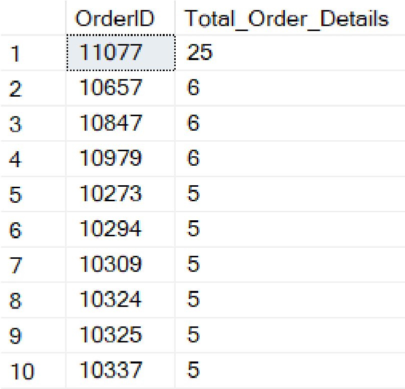
Solution: Orders and Orders Details Table are Used

Code

Output



```
select top 10 OD.OrderID, count(OD.OrderID) as Total_Order_Details
from [dbo].[Order Details] OD
inner join Orders O
on OD.OrderID = O.OrderID
group by OD.OrderID
order by count(OD.OrderID) desc
```

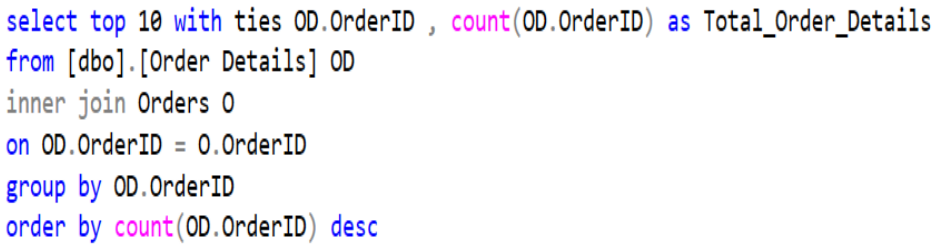


	OrderID	Total_Order_Details
1	11077	25
2	10657	6
3	10847	6
4	10979	6
5	10273	5
6	10294	5
7	10309	5
8	10324	5
9	10325	5
10	10337	5

Problem 5:
There might be a chance that ties in the number of order details for the top 10 orders,
If yes, which orders are tied and how many order details do they have

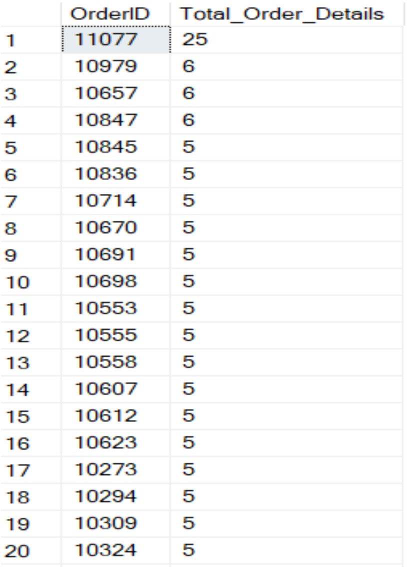
Solution: Orders and Orders Details Table are Used

Code



```
select top 10 with ties OD.OrderID , count(OD.OrderID) as Total_Order_Details
from [dbo].[Order Details] OD
inner join Orders O
on OD.OrderID = O.OrderID
group by OD.OrderID
order by count(OD.OrderID) desc
```

Output

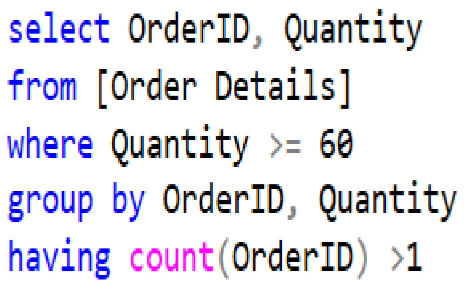


	OrderID	Total_Order_Details
1	11077	25
2	10979	6
3	10657	6
4	10847	6
5	10845	5
6	10836	5
7	10714	5
8	10670	5
9	10691	5
10	10698	5
11	10553	5
12	10555	5
13	10558	5
14	10607	5
15	10612	5
16	10623	5
17	10273	5
18	10294	5
19	10309	5
20	10324	5

Problem 6:
Janet Leverling, one of the salespeople, has come to you with a request. She thinks that she accidentally double-entered a line item on an order, with a different ProductID, but the same quantity. She remembers that the quantity was 60 or more. Show all the OrderIDs with line items that match this, in order of OrderID

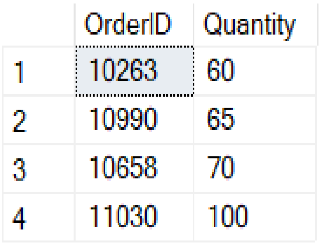
Solution: Orders Details Table are Used

Code



```
select OrderID, Quantity
from [Order Details]
where Quantity >= 60
group by OrderID, Quantity
having count(OrderID) >1
```

Output

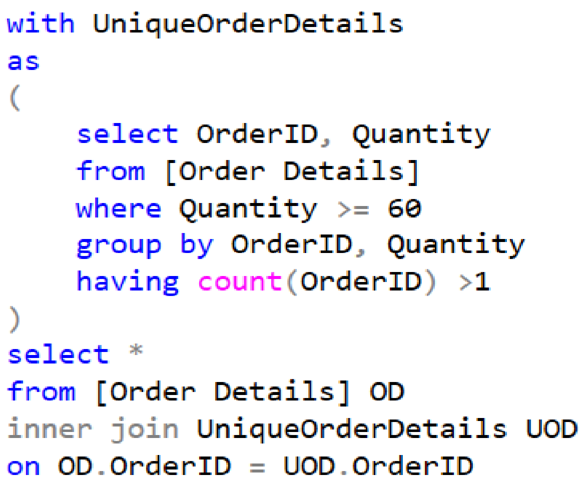


	OrderID	Quantity
1	10263	60
2	10990	65
3	10658	70
4	11030	100

Problem 7:
Based on the previous question, we now want to show details of the order, for orders that match the above criteria

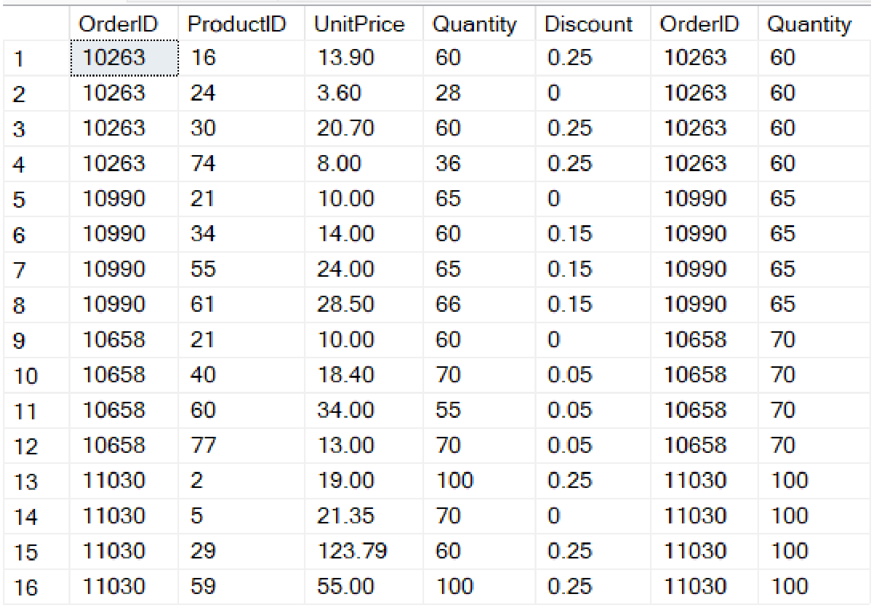
Solution: Orders Details Table are Used

Code



```
with UniqueOrderDetails
as
(
     select OrderID, Quantity
     from [Order Details]
     where Quantity >= 60
     group by OrderID, Quantity
     having count(OrderID) >1
)
select *
from [Order Details] OD
inner join UniqueOrderDetails UOD
on OD.OrderID = UOD.OrderID
```

Output

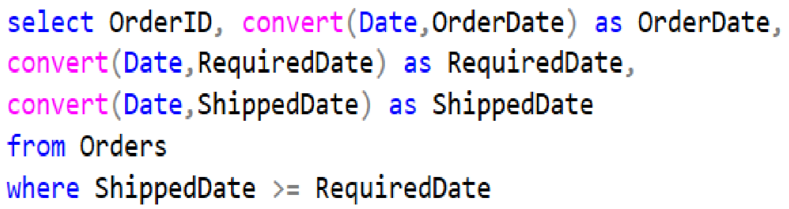


	OrderID	ProductID	UnitPrice	Quantity	Discount	OrderID	Quantity
1	10263	16	13.90	60	0.25	10263	60
2	10263	24	3.60	28	0	10263	60
3	10263	30	20.70	60	0.25	10263	60
4	10263	74	8.00	36	0.25	10263	60
5	10990	21	10.00	65	0	10990	65
6	10990	34	14.00	60	0.15	10990	65
7	10990	55	24.00	65	0.15	10990	65
8	10990	61	28.50	66	0.15	10990	65
9	10658	21	10.00	60	0	10658	70
10	10658	40	18.40	70	0.05	10658	70
11	10658	60	34.00	55	0.05	10658	70
12	10658	77	13.00	70	0.05	10658	70
13	11030	2	19.00	100	0.25	11030	100
14	11030	5	21.35	70	0	11030	100
15	11030	29	123.79	60	0.25	11030	100
16	11030	59	55.00	100	0.25	11030	100

Problem 8:
Some customers are complaining about their orders arriving late. Which orders are late

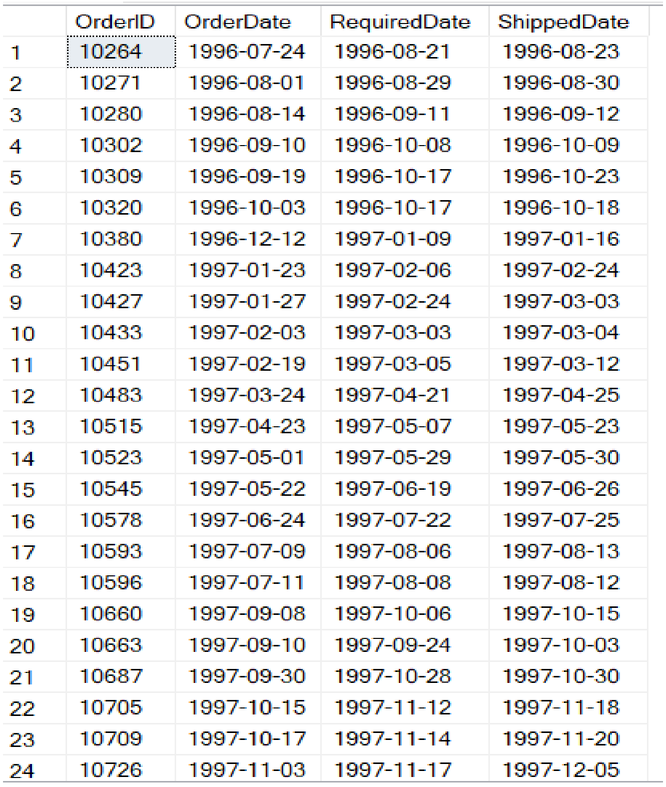
Solution: Orders Table are Used p

Code



```
select OrderID, convert(Date,OrderDate) as OrderDate,
convert(Date,RequiredDate) as RequiredDate,
convert(Date,ShippedDate) as ShippedDate
from Orders
where ShippedDate >= RequiredDate
```

Output



	OrderID	OrderDate	RequiredDate	ShippedDate
1	10264	1996-07-24	1996-08-21	1996-08-23
2	10271	1996-08-01	1996-08-29	1996-08-30
3	10280	1996-08-14	1996-09-11	1996-09-12
4	10302	1996-09-10	1996-10-08	1996-10-09
5	10309	1996-09-19	1996-10-17	1996-10-23
6	10320	1996-10-03	1996-10-17	1996-10-18
7	10380	1996-12-12	1997-01-09	1997-01-16
8	10423	1997-01-23	1997-02-06	1997-02-24
9	10427	1997-01-27	1997-02-24	1997-03-03
10	10433	1997-02-03	1997-03-03	1997-03-04
11	10451	1997-02-19	1997-03-05	1997-03-12
12	10483	1997-03-24	1997-04-21	1997-04-25
13	10515	1997-04-23	1997-05-07	1997-05-23
14	10523	1997-05-01	1997-05-29	1997-05-30
15	10545	1997-05-22	1997-06-19	1997-06-26
16	10578	1997-06-24	1997-07-22	1997-07-25
17	10593	1997-07-09	1997-08-06	1997-08-13
18	10596	1997-07-11	1997-08-08	1997-08-12
19	10660	1997-09-08	1997-10-06	1997-10-15
20	10663	1997-09-10	1997-09-24	1997-10-03
21	10687	1997-09-30	1997-10-28	1997-10-30
22	10705	1997-10-15	1997-11-12	1997-11-18
23	10709	1997-10-17	1997-11-14	1997-11-20
24	10726	1997-11-03	1997-11-17	1997-12-05