USE wideworldimporters

GO

/\* see number of tables\*/

SELECT \*

FROM sys.Tables;

/\* see data tables info\*/

select schema\_name(t.schema\_id) as schema\_name,

t.name as table\_name,

t.create\_date,

t.modify\_date

from sys.tables t

order by schema\_name,

table\_name;

/\* question 1 \*/

-- ######################################################################################################################

/\* Three types of people in the application.person table: Stuff, Customer, Supplier;

We will have to use Purchasing.suppliers, Sales.customers to figure out the company info for customers and suppliers;

we will then find out the stuff (salesperson or other stuff) by connecting to the application.people itself

NOT EFFICIENT THO\*/

SELECT FullName, FaxNumber, PhoneNumber, PhoneNumber AS CompanyPhone, FaxNumber AS CompanyFax

FROM Application.People

WHERE IsEmployee = 1

UNION

SELECT p.FullName, p.FaxNumber, p.PhoneNumber, c.PhoneNumber AS CompanyPhone, c.FaxNumber AS CompanyFax

FROM Application.People as p Inner join sales.Customers as c

ON p.PersonID = c.PrimaryContactPersonID or p.PersonID = c.AlternateContactPersonID

UNION

SELECT p.FullName, p.FaxNumber, p.PhoneNumber, s.PhoneNumber AS CompanyPhone, s.FaxNumber AS CompanyFax

FROM Application.People as p Inner join Purchasing.Suppliers as s

ON p.PersonID = s.PrimaryContactPersonID or p.PersonID = s.AlternateContactPersonID;

/\* question 2 \*/

-- ######################################################################################################################

/\* we need to combine application.people and sales.customers and find out if primary contact person has the

same phone Number as the customer\*/

SELECT c.CustomerName as CompanyName

FROM SALES.Customers as c inner join Application.People as p

ON p.PersonID = c.PrimaryContactPersonID

WHERE p.PhoneNumber = c.PhoneNumber;

/\* question 3 \*/

-- ######################################################################################################################

/\* first find customer had orders prior 2016\*/

/\* Then find customer ordered in and after 2016\*/

-- method 1

SELECT DISTINCT c.CustomerID, c.CustomerName

from sales.Customers as c inner join sales.Orders as o

ON c.CustomerID = o.CustomerID

WHERE o.OrderDate < '2016-01-01' AND c.CustomerID NOT in (select DISTINCT CustomerID from sales.orders

where OrderDate >= '2016-01-01');

-- method 2

select DISTINCT c.CustomerID, c.CustomerName

from sales.Customers as c inner join sales.Orders as o

ON c.CustomerID = o.CustomerID

WHERE o.CustomerID NOT in (select DISTINCT CustomerID from sales.orders

where OrderDate >= '2016-01-01');

/\* question 4 \*/

-- #######################################################################################################################

/\* what does the purcahse order mean here? purchased from suppliers or sold to customers? if it is purchase orders,

then what is the quantity in this table? OrderedOuter? \*/

-- if sold orders then

SELECT s.StockItemID, s.StockItemName, sum(ol.Quantity) as TotalQuantity

FROM Warehouse.StockItems AS s Inner join sales.orderlines as ol ON s.stockItemID = ol.stockitemID

inner join sales.orders as o ON o.OrderID = ol.OrderID

WHERE year(o.OrderDate) = 2013

GROUP BY s.StockItemID, s.StockItemName;

-- if purchased orders then

SELECT s.StockItemID, s.StockItemName, sum(pl.OrderedOuters) as TotalQuantity

FROM Warehouse.StockItems AS s Inner join Purchasing.PurchaseOrderLines as pl

ON s.stockItemID = pl.stockitemID

inner join Purchasing.PurchaseOrders as p

ON p.PurchaseOrderID = pl.PurchaseOrderID

WHERE year(p.OrderDate) = 2013

GROUP BY s.StockItemID, s.StockItemName;

/\* Question 5 \*/

-- #######################################################################################################################

-- method 1

select distinct s.StockItemName

from Purchasing.PurchaseOrderLines as p inner join Warehouse.StockItems as s

on p.StockItemID = s.StockItemID where len(p.Description) >= 10;

-- method 2

select distinct s.StockItemName

from Sales.OrderLines as ol inner join Warehouse.StockItems as s

on ol.StockItemID = s.StockItemID where len(ol.Description) >= 10;

/\* question 6 \*/

-- #######################################################################################################################

select StockItemID, StockItemName from Warehouse.StockItems

WHERE StockItemID NOT IN

(select s.StockItemID

from warehouse.StockItems as s

inner join Sales.InvoiceLines as il on s.StockItemID = il.StockItemID

inner join sales.Invoices as i on il.InvoiceID = i.InvoiceID

inner join sales.customers as c on i.CustomerID = c.CustomerID

inner join Application.Cities as ct on c.DeliveryCityID = ct.CityID

inner join Application.StateProvinces as st on ct.StateProvinceID = st.StateProvinceID

where year(i.InvoiceDate) = 2014 AND (st.StateProvinceName = 'Alabama' or st.StateProvinceName = 'Georgia')

);

/\* question 7 \*/

-- #######################################################################################################################

/\* following is the answer\*/

select st.StateProvinceID as StateID, avg(datediff(day, o.OrderDate, convert(date, i.ConfirmedDeliveryTime))) as AvgTime

from sales.Orders as o inner join sales.Invoices as i on o.OrderID = i.OrderID

inner join sales.Customers as c on c.CustomerID = i.CustomerID

inner join Application.Cities as ct on c.DeliveryCityID = ct.CityID

inner join Application.StateProvinces as st on st.StateProvinceID = ct.StateProvinceID

Group by st.StateProvinceID;

/\* question 8\*/

-- ######################################################################################################################

select st.StateProvinceID as StateID, Month(o.OrderDate) MonthInAYear,

avg(datediff(day, o.OrderDate, convert(date, i.ConfirmedDeliveryTime))) as AvgTime

from sales.Orders as o inner join sales.Invoices as i on o.OrderID = i.OrderID

inner join sales.Customers as c on c.CustomerID = i.CustomerID

inner join Application.Cities as ct on c.DeliveryCityID = ct.CityID

inner join Application.StateProvinces as st on st.StateProvinceID = ct.StateProvinceID

Group by st.StateProvinceID, Month(o.OrderDate)

order by st.StateProvinceID, Month(o.OrderDate);

/\* question 9\*/

-- #####################################################################################################################

select sold.ID, sold.quantity AS SoldQuantity, purchased.ID, purchased.quantity as PurchasedQuantity

from

(select s.StockItemID as ID, sum(ol.Quantity) as quantity

from Warehouse.StockItems as s inner join Sales.OrderLines as ol on ol.StockItemID = s.StockItemID

inner join Sales.Orders as o on o.OrderID = ol.OrderID

where YEAR(o.OrderDate) = 2015

group by s.StockItemID) as sold

full outer join

(select s.StockItemID as ID, sum(pl.OrderedOuters) as quantity

from Purchasing.PurchaseOrderLines as pl inner join Purchasing.PurchaseOrders as po on po.PurchaseOrderID = pl.PurchaseOrderID

inner join Warehouse.StockItems as s on s.StockItemID = pl.StockItemID

where year(po.OrderDate) = 2015

Group by s.StockItemID) as purchased

on sold.ID = purchased.ID

where sold.quantity < purchased.quantity;

/\* question 10 \*/

-- #####################################################################################################################

select CustomerName, CustomerID, customer.PhoneNumber, People.FullName as PrimaryPersonName

from Sales.Customers as customer INNER JOIN Application.People AS People

on customer.PrimaryContactPersonID = People.PersonID

where customer.CustomerID IN

(select c.CustomerID

from sales.Customers as c inner join Sales.Orders as o on c.CustomerID = o.CustomerID

inner join sales.OrderLines as ol on o.OrderID = ol.OrderID

inner join Warehouse.StockItems as si on si.StockItemID = ol.StockItemID

Where si.StockItemName like '%mug%' AND year(o.OrderDate) = 2016

Group by c.CustomerID

HAVING COUNT(si.StockItemID) < 10);

/\* question 11 \*/

-- ####################################################################################################################

SELECT distinct CityID, CityName, ValidFrom FROM Application.Cities

except

select distinct CityID, CityName, ValidFrom from application.Cities

FOR SYSTEM\_TIME AS OF '2015-01-01 00:00:00.0000000';

/\* question 12 \*/

-- ####################################################################################################################

Select StockItemName,deliveryAddress, StateName, CityName, CountryName, CustomerName, CustomerPhone, Quantity,

p.PhoneNumber as ContactPersonPhone

from

(select si.StockItemName as StockItemName, CONCAT(c.DeliveryAddressLine1, c.DeliveryAddressLine2) as deliveryAddress,

sp.StateProvinceName as StateName, ct.CityName as CityName, cty.CountryName as CountryName,

c.CustomerName as CustomerName, c.PhoneNumber as CustomerPhone, ol.Quantity as Quantity,

i.ContactPersonID as ContactPersonID

from Warehouse.StockItems as si inner join sales.OrderLines as ol on si.StockItemID = ol.StockItemID

inner join sales.Orders as o on ol.OrderID = o.OrderID

inner join sales.Invoices as i on o.OrderID = i.OrderID

inner join sales.Customers as c on i.CustomerID = c.CustomerID

inner join Application.Cities as ct on c.PostalCityID = ct.CityID

inner join Application.StateProvinces as sp on ct.StateProvinceID = sp.StateProvinceID

inner join Application.Countries as cty on sp.CountryID = cty.CountryID

where o.OrderDate = '2014-07-01') as t

inner join Application.People as p on t.ContactPersonID = p.PersonID;

/\* question 13\*/

-- ########################################################################################################################################

SELECT sold.id, sold.tol as sold, purchased.tol as purchased,

(purchased.tol - sold.tol) as remaining\_stock

FROM

(

/\*sold\*/

select sg.StockGroupID as id, sum(ol.Quantity) as tol

from Warehouse.StockGroups as sg inner join

Warehouse.StockItemStockGroups as sisg on sg.StockGroupID=sisg.StockGroupID

inner join Warehouse.StockItems as si on si.StockItemID = sisg.StockItemID

inner join sales.OrderLines as ol on ol.StockItemID = si.StockItemID

group by sg.StockGroupID) AS sold

INNER JOIN

/\*purchased\*/-- how to calculate the purchase quantity?

(select sg.StockGroupID as id, sum(pol.OrderedOuters) as tol

from Warehouse.StockGroups as sg inner join

Warehouse.StockItemStockGroups as sisg on sg.StockGroupID=sisg.StockGroupID

inner join Warehouse.StockItems as si on si.StockItemID = sisg.StockItemID

inner join Purchasing.PurchaseOrderLines as pol on pol.StockItemID = si.StockItemID

group by sg.StockGroupID) AS purchased

ON sold.id = purchased.id;

-- negative?????

/\* question 14\*/

-- ###############################################################################################################

with rank\_table as

(

select ct.CityID as CityID, si.StockItemID as StockItemID, count(si.StockItemID) as sales,

RANK() over (partition by ct.cityid order by count(si.StockItemID) desc) as ranks

from application.Cities as ct

left join sales.Customers as c on ct.CityID = c.DeliveryCityID

inner join sales.Orders as o on c.CustomerID = o.CustomerID

inner join Sales.OrderLines ol on o.OrderID = ol.OrderID

inner join Warehouse.StockItems as si on si.StockItemID = ol.StockItemID

WHERE year(o.orderdate) = '2016'

group by ct.CityID, si.StockItemID

)

select CityID, StockItemID, coalesce(sales, 'No Sales')

from rank\_table where ranks = 1;

/\* question 15\*/

-- ###############################################################################################################

-- if orders are shipped one than once, then this order with have more than one invoice id

select i.OrderID

from sales.Invoices as i

group by i.OrderID

having count(i.invoiceid) > 1;

/\* question 16\*/

-- ###############################################################################################################

-- use json function to find out the items made in china

select s.StockItemID, s.StockItemName

from Warehouse.StockItems as s

WHERE JSON\_VALUE(s.CustomFields, '$.CountryOfManufacture') = 'China';

/\* question 17\*/

-- ###############################################################################################################

SELECT JSON\_VALUE(si.CustomFields, '$.CountryOfManufacture') as country,

case count(quantity)

WHEN null then 0

ELSE count(quantity)

end as quantity

from Warehouse.StockItems as si

left join sales.OrderLines as ol on si.StockItemID=ol.StockItemID

inner join sales.Orders as o on ol.OrderID = o.OrderID

where year(o.OrderDate) = 2015

group by JSON\_VALUE(si.CustomFields, '$.CountryOfManufacture')

UNION

SELECT 'USA', 0;

/\* question 18 \*/

-- ################################################################################################################

CREATE or ALTER view TolSalesForGroups as

(select stockgroup, [2013], [2014], [2015], [2016], 0 as [2017]

from (

select groups as StockGroup,

[2013], [2014], [2015], [2016]

from

(select quantity, groups, years

from (

select sg.StockGroupID, sg.StockGroupName as groups, year(o.OrderDate) as years, ol.Quantity as quantity

from Warehouse.StockGroups as sg inner join Warehouse.StockItems as si on sg.StockGroupID = sg.StockGroupID

inner join Sales.OrderLines as ol on ol.StockItemID = si.StockItemID

inner join sales.Orders as o on o.OrderID = ol.OrderID

where YEAR(o.OrderDate) between 2013 and 2017

) as table1

) as sourcetable

Pivot

(

SUM(quantity)

for years in

([2013], [2014], [2015], [2016])

) as pvt) as table2

)

GO

/\* test if we create the view successfully\*/

select \* from TolSalesForGroups;

/\* question 19 \*/

-- ##################################################################################################################

CREATE OR ALTER view TolSalesForGroups as

(select years, [Airline Novelties], [Clothing], [Computing Novelties], [Furry Footwear],

[Mugs], [Novelty Items], [Packaging Materials], [Toys], [T-Shirts], [USB Novelties]

from (

select years,

[Airline Novelties], [Clothing], [Computing Novelties], [Furry Footwear],

[Mugs], [Novelty Items], [Packaging Materials], [Toys], [T-Shirts], [USB Novelties]

from

(select quantity, groups, years

from (

select sg.StockGroupID, sg.StockGroupName as groups, year(o.OrderDate) as years, ol.Quantity as quantity

from Warehouse.StockGroups as sg inner join Warehouse.StockItems as si on sg.StockGroupID = sg.StockGroupID

inner join Sales.OrderLines as ol on ol.StockItemID = si.StockItemID

inner join sales.Orders as o on o.OrderID = ol.OrderID

where YEAR(o.OrderDate) between 2013 and 2017

) as table1

) as sourcetable

Pivot

(

SUM(quantity)

for groups in

([Airline Novelties], [Clothing], [Computing Novelties], [Furry Footwear],

[Mugs], [Novelty Items], [Packaging Materials], [Toys], [T-Shirts], [USB Novelties])

) as pvt) as table2

UNION

select 2017, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

);

-- Test the view created

SELECT \* from TolSalesForGroups;

/\* question 20 \*/

-- ###################################################################################################################

-- total of what?

CREATE or ALTER FUNCTION OrderTotal (

@OrderIDInput int

)

RETURNS TABLE

AS

RETURN

select o.OrderID as orderID, sum(ol.Quantity\*ol.UnitPrice) as total

from Sales.Orders as o

inner join sales.orderlines as ol on o.OrderID = ol.OrderID

WHERE o.OrderID = @OrderIDInput

Group by o.OrderID;

/\* test if function works \*/

SELECT \* FROM ORDERtotal(1);

select \* from OrderTotal(2);

/\* use apply to join them together, we can only use apply here cuz we have table valued function\*/

select \* from Sales.Invoices as i

OUTER APPLY OrderTotal(i.OrderID);

/\* question 21 \*/

-- ########################################################################################################################

DROP TABLE [ods.ORDERS]

CREATE TABLE [ods.ORDERS]

(

orderID int not null PRIMARY KEY,

orderDate Date,

orderTotal int,

customerID int

);

CREATE OR ALTER Proc procedure21

@dateinput DATE

AS

SET nocount on;

Begin try

BEGIN TRANSACTION

insert into [ods.ORDERS]

select o.OrderID, o.OrderDate, SUM(ol.Quantity\*ol.UnitPrice) as total, o.CustomerID

from sales.Orders as o inner join sales.OrderLines as ol on o.OrderID = ol.OrderID

where o.OrderDate = @dateinput

group by o.OrderDate, o.OrderID, o.CustomerID

COMMIT TRANSACTION;

END TRY

BEGIN CATCH

PRINT 'Order already exists in the table.'

ROLLBACK TRANSACTION;

END CATCH

RETURN

GO

--test the proc

-- 1 TRY

exec procedure21 @dateinput = '2015-01-01'

SELECT \* FROM [ods.orders]

-- 2 TRY

exec procedure21 @dateinput = '2015-01-02'

-- 3 TRY

EXEC procedure21 '2016-06-17'

-- 4 TRY

exec procedure21 '2016-09-03'

-- 5 try

exec procedure21 '2014-04-11'

-- if we use 5 different dates, how are we gonna test the error handling statement, or we just dont?

/\* question 22 \*/

-- ###############################################################################################################################

-- [RANGE], [SHELFLIFE], no these two in the tables: add two new columns?

DROP table [ods.StockItem]

SELECT [StockItemID], [StockItemName],[SupplierID],

[ColorID],[UnitPackageID],[OuterPackageID],[Brand],[Size],[LeadTimeDays],

[QuantityPerOuter],[IsChillerStock],[Barcode],

[TaxRate],[UnitPrice],[RecommendedRetailPrice],

[TypicalWeightPerUnit],[MarketingComments],[InternalComments],

JSON\_VALUE(CustomFields, '$.CountryOfManufacture') as [CountryOfManufacture]

INTO [ods.StockItem]

FROM Warehouse.StockItems;

ALTER TABLE [ods.StockItem]

ADD [Range] Varchar(255), [Shelflife] Varchar(255);

SELECT \* FROM [ods.StockItem];

/\* Question 23\*/

-- ##############################################################################################################################

CREATE OR ALTER Proc procedure23

@dateinput DATE

AS

SET nocount on;

-- create table

select o.OrderID as OrderID, o.OrderDate as OrderDate, SUM(ol.Quantity\*ol.UnitPrice) as total, o.CustomerID as CustomerID

into [#ods.ORDERS]

from sales.Orders as o inner join

sales.OrderLines as ol on o.OrderID = ol.OrderID

group by o.OrderDate, o.OrderID, o.CustomerID

--delete from table

DELETE FROM [#ods.ORDERS]

WHERE OrderDate < @dateinput

--return orders in the next 7 days

SELECT \* FROM [#ods.ORDERS]

WHERE OrderDate > @dateinput AND OrderDate <= DATEADD(DAY, 7, @dateinput)

RETURN

GO

EXEC PROCEDURE23 @dateinput = '2014-01-01';

/\* question 24 \*/

-- ##############################################################################################################################

DECLARE @json nvarchar(MAX)

DECLARE @jsons nvarchar(MAX);

SET @json = N'

{"PurchaseOrders":[

{

"StockItemName":"Panzer Video Game",

"Supplier":"7",

"UnitPackageId":"1",

"OuterPackageId":[

6,

7

],

"Brand":"EA Sports",

"LeadTimeDays":"5",

"QuantityPerOuter":"1",

"TaxRate":"6",

"UnitPrice":"59.99",

"RecommendedRetailPrice":"69.99",

"TypicalWeightPerUnit":"0.5",

"CountryOfManufacture":"Canada",

"Range":"Adult",

"OrderDate":"2018-01-01",

"DeliveryMethod":"Post",

"ExpectedDeliveryDate":"2018-02-02",

"SupplierReference":"WWI2308"

},

{

"StockItemName":"Panzer Video Game",

"Supplier":"5",

"UnitPackageId":"1",

"OuterPackageId":"7",

"Brand":"EA Sports",

"LeadTimeDays":"5",

"QuantityPerOuter":"1",

"TaxRate":"6",

"UnitPrice":"59.99",

"RecommendedRetailPrice":"69.99",

"TypicalWeightPerUnit":"0.5",

"CountryOfManufacture":"Canada",

"Range":"Adult",

"OrderDate":"2018-01-025",

"DeliveryMethod":"Post",

"ExpectedDeliveryDate":"2018-02-02",

"SupplierReference":"269622390"

}

]

}';

-- split into two rows

With table1 (col) as (

select json\_query(@json, '$.PurchaseOrders[0]')

),

table2 (col) as (

select json\_query(@json, '$.PurchaseOrders[1]')

)

SELECT \* INTO #temp\_table

FROM (

SELECT \* FROM table1

UNION all

SELECT \* FROM table2) as table3;

SELECT \* FROM #temp\_table;

DECLARE @var1 VARCHAR(MAX);

SET @var1 = (SELECT top 1 \* FROM #temp\_table);

SELECT \*

FROM OPENJSON(@var1);

/\* question 25 \*/

-- #########################################################################################################################

-- FOR JSON PATH

SELECT \* from TolSalesForGroups ORDER BY 1

FOR JSON PATH;

/\* question 26 \*/

-- #########################################################################################################################

-- FOR XML PATH

SELECT years, [Airline Novelties] as AirlineNovelties,

[Clothing], [Computing Novelties] as ComputingNovelties, [Furry Footwear] as FurryFootwear,

[Mugs], [Novelty Items] as NoveltyItems, [Packaging Materials] as PackagingMaterials, [Toys],

[T-Shirts], [USB Novelties] as USBNovelties

from TolSalesForGroups ORDER BY 1

FOR XML PATH;

/\* question 27\*/

-- ##########################################################################################################################

-- create table

DROP table [ods.confirmedDeliveryJson];

CREATE TABLE [ods.confirmedDeliveryJson] (

ID INT identity(1, 1),

DelDate DATETIME,

TableValue NVarchar(MAX));

-- create procedure

CREATE PROC procedure27

@dateinput date

as

-- select all columns from both tables and convert into JSON

with table1 (the\_value) as (

SELECT \* FROM sales.Invoices as i inner join sales.InvoiceLines as il on i.InvoiceID = il.InvoiceID

WHERE i.InvoiceDate = @dateinput

FOR JSON AUTO)

-- insert into new table

INSERT INTO [ods.confirmedDeliveryJson]

SELECT @dateinput, the\_value

FROM table1

GO

-- run the SP

DECLARE @inputdate DATE

DECLARE cur CURSOR LOCAL FOR

SELECT DISTINCT InvoiceDate FROM Sales.Invoices WHERE CustomerID = 1

OPEN cur

FETCH NEXT FROM cur INTO @inputdate

WHILE @@FETCH\_STATUS = 0

BEGIN

EXEC procedure27 @inputdate

FETCH NEXT FROM cur INTO @inputdate

END

CLOSE cur

DEALLOCATE cur

-- return the table

SELECT \* FROM [ods.confirmedDeliveryJson];