SQL Assignments

SQL related assignments will be on Wide World Importers Database if not otherwise introduced.

1. List of Persons’ full name, all their fax and phone numbers, as well as the phone number and fax of the company they are working for (if any).

Select FullName,PhoneNumber, FaxNumber From Application.People

Union All

Select CustomerName,PhoneNumber, FaxNumber From Sales.Customers Where CustomerCategoryID = 1

Graphical user interface, text, table, Excel

Description automatically generated

1. If the customer's primary contact person has the same phone number as the customer’s phone number, list the customer companies.

Select a.CustomerName From Sales.Customers as a

left Join Application.People as b on a.PrimaryContactPersonID = b.PersonID

where a.PhoneNumber = b.PhoneNumber

Graphical user interface, text, application

Description automatically generated

1. List of customers to whom we made a sale prior to 2016 but no sale since 2016-01-01.

Select c.CustomerName From Sales.Customers as c

Left Join Sales.Orders as o on c.CustomerID = o.CustomerID

And o.OrderDate>= '2016-01-01'

Where o.CustomerID is null

Graphical user interface, application

Description automatically generated

1. List of Stock Items and total quantity for each stock item in Purchase Orders in Year 2013.

Select si.StockItemName, il.Quantity as "Total Quantity"

from Sales.InvoiceLines il

join Warehouse.StockItems si on il.StockItemID = si.StockItemID

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

Join Purchasing.PurchaseOrders po on po.PurchaseOrderID = pol.PurchaseOrderID

where YEAR(po.OrderDate) = 2013

Group by si.StockItemName ,il.Quantity

Text, table

Description automatically generated with medium confidence

1. List of stock items that have at least 10 characters in description.

Select StockItemName From Warehouse.StockItems si

join Sales.InvoiceLines il on il.StockItemID = si.StockItemID

where LEN(Description) >= 10

Graphical user interface, text, application

Description automatically generated

1. List of stock items that are not sold to the state of Alabama and Georgia in 2014.

select distinct StockItemName from Warehouse.StockItems si

join Sales.InvoiceLines il on il.StockItemID = si.StockItemID

join Sales.CustomerTransactions ct on ct.InvoiceID = il.InvoiceID

join Sales.Customers c on c.CustomerID = ct.CustomerID

join Application.Cities cs on cs.CityID = c.DeliveryCityID

join Application.StateProvinces sp on cs.StateProvinceID = sp.StateProvinceID

where sp.StateProvinceName !='Alabama' and sp.StateProvinceName!='Georgia' and Year(ct.FinalizationDate)!='2014'

Graphical user interface, text, application

Description automatically generated

1. List of States and Avg dates for processing (confirmed delivery date – order date).

select sp.StateProvinceName, AVG(DATEDIFF(day, o.OrderDate, i.ConfirmedDeliveryTime)) as AvgDates

from Sales.Orders o join Sales.Invoices i on i.OrderID = o.OrderID

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

join Application.Cities ci on ci.CityID = c.DeliveryCityID

join Application.StateProvinces sp on sp.StateProvinceID = ci.StateProvinceID

group by sp.StateProvinceName;

Table

Description automatically generated

1. List of States and Avg dates for processing (confirmed delivery date – order date) by month.

select sp.StateProvinceName, AVG(DATEDIFF(day, o.OrderDate, i.ConfirmedDeliveryTime)) as 'Duration', MONTH(o.OrderDate) AS 'Month'

from Sales.Orders o join Sales.Invoices i on i.OrderID = o.OrderID

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

join Application.Cities ci on ci.CityID = c.DeliveryCityID

join Application.StateProvinces sp on sp.StateProvinceID = ci.StateProvinceID

group by sp.StateProvinceName, MONTH(o.OrderDate);

Table

Description automatically generated with medium confidence

1. List of StockItems that the company purchased more than sold in the year of 2015.

with Sales as (

select si.StockItemID, SUM(ol.Quantity) as SalesQuantity

from Warehouse.StockItems si

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

join Sales.Orders o on o.OrderID = ol.OrderID and YEAR(o.OrderDate) =2015

group by si.StockItemID

),

Purchases as (

select si.StockItemID, sum(pol.ReceivedOuters\*si.QuantityPerOuter) as PurchasesQuantity

from Warehouse.StockItems si

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

join Purchasing.PurchaseOrders po on po.PurchaseOrderID = pol.PurchaseOrderID and YEAR(po.OrderDate) = 2015

group by si.StockItemID

)

select p.StockItemID, p.PurchasesQuantity, s.SalesQuantity

from Purchases p

join Sales s on s.StockItemID = p.StockItemID

where PurchasesQuantity > SalesQuantity;

Table

Description automatically generated

1. List of Customers and their phone number, together with the primary contact person’s name, to whom we did not sell more than 10 mugs (search by name) in the year 2016.

select c1.CustomerName, c1.PhoneNumber, p.FullName, p.PhoneNumber, s2.SoldQuantity

from(

select c.CustomerID, SUM(ol.Quantity) as 'SoldQuantity'

from (

select StockItemID

from Warehouse.StockItems

where StockItemName like '%mug%') s1

join Sales.OrderLines ol on ol.StockItemID = s1.StockItemID

join Sales.Orders o on o.OrderID = ol.OrderID and YEAR(o.OrderDate)=2016

join Sales.Customers c on c.CustomerID = o.CustomerID

group by c.CustomerID

having SUM(ol.Quantity)<=10) s2

join Sales.Customers c1 on c1.CustomerID = s2.CustomerID

join Application.People p on p.PersonID = c1.PrimaryContactPersonID

Table

Description automatically generated

1. List all the cities that were updated after 2015-01-01.

select c.CityName

from Application.Cities c join sales.Customers cu

on c.CityID=cu.PostalCityID

where cu.ValidFrom > '2015-01-01'

Union

select c.CityName

from Application.Cities c join Purchasing.Suppliers s

on c.CityID=s.PostalCityID

where s.ValidFrom > '2015-01-01';

Graphical user interface, application, table

Description automatically generated

1. List all the Order Detail (Stock Item name, delivery address, delivery state, city, country, customer name, customer contact person name, customer phone, quantity) for the date of 2014-07-01. Info should be relevant to that date.

select si.StockItemName, cu.DeliveryAddressLine1, cu.DeliveryAddressLine2,

cu.DeliveryCityID, cu.CustomerName, p.FullName, cu.PhoneNumber, ol.Quantity

from Sales.OrderLines ol

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

join Sales.Customers cu on cu.CustomerID = o.CustomerID

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

join Application.People p on p.PersonID = cu.PrimaryContactPersonID

where o.OrderDate = '2014-07-01';

Graphical user interface

Description automatically generated

1. List of stock item groups and total quantity purchased, total quantity sold, and the remaining stock quantity (quantity purchased – quantity sold)

With Purchases as

(select SG.StockGroupID, sum(POL.OrderedOuters) as Purchase

from Purchasing.PurchaseOrderLines as POL

join Warehouse.StockItemStockGroups as SISG on SISG.StockItemID = POL.StockItemID

join Warehouse.StockGroups as SG on SG.StockGroupID = SISG.StockGroupID

group by SG.StockGroupID),

Sales as

(select SG.StockGroupID, sum(OL.Quantity) as Sale

from Sales.OrderLines as OL

join Warehouse.StockItemStockGroups as SISG on SISG.StockItemID = OL.StockItemID

join Warehouse.StockGroups as SG on SG.StockGroupID = SISG.StockGroupID

group by SG.StockGroupID)

select p.StockGroupID, p.Purchase, s.Sale, (p.Purchase - s.Sale) as RemainStock

from Purchases p

join Sales s on p.StockGroupID = s.StockGroupID

order by p.StockGroupID

Table

Description automatically generated

1. List of Cities in the US and the stock item that the city got the most deliveries in 2016. If the city did not purchase any stock items in 2016, print “No Sales”.

with

cte1 as (

select ol.StockItemID, c.DeliveryCityID, COUNT(\*) AS Delivery

FROM Sales.OrderLines ol

JOIN Sales.Orders o ON o.OrderID = ol.OrderID

JOIN sales.Customers c ON o.CustomerID = c.CustomerID

WHERE YEAR(o.OrderDate) = 2016

GROUP BY ol.StockItemID, c.DeliveryCityID),

cte2 AS(

SELECT StockItemID, DeliveryCityID

FROM (

SELECT StockItemID, DeliveryCityID,

DENSE\_RANK() OVER(PARTITION BY DeliveryCityId ORDER BY Delivery DESC) AS rnk

FROM cte1) a

WHERE rnk = 1

)

SELECT c.CityName, ISNULL(s.StockItemName, 'No Sale') AS MostDelivery

FROM cte2 c1 JOIN Warehouse.StockItems s ON c1.StockItemID = s.StockItemID

RIGHT JOIN Application.Cities c ON c1.DeliveryCityID = c.CityID

Table

Description automatically generated

1. List any orders that had more than one delivery attempt (located in invoice table).

select OrderID

from Sales.Invoices

where JSON\_VALUE(ReturnedDeliveryData, '$.Events[1].Comment') IS NOT NULL

A picture containing table

Description automatically generated

1. List all stock items that are manufactured in China. (Country of Manufacture)

select SI.StockItemName, JSON\_VALUE(SI.CustomFields, '$.CountryOfManufacture') as Country

from Warehouse.StockItems as si

where JSON\_VALUE(si.CustomFields, '$.CountryOfManufacture') = 'China'

Text, table

Description automatically generated

1. Total quantity of stock items sold in 2015, group by country of manufacturing.

select JSON\_VALUE(si.CustomFields, '$.CountryOfManufacture') as Country, SUM(ol.Quantity) as TotalQuantity

from sales.Orders o

join Sales.OrderLines ol ON o.OrderID = ol.OrderID

join Warehouse.StockItems si ON ol.StockItemID = si.StockItemID

where YEAR(o.OrderDate) = 2015

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

Table

Description automatically generated

1. Create a view that shows the total quantity of stock items of each stock group sold (in orders) by year 2013-2017. [Stock Group Name, 2013, 2014, 2015, 2016, 2017]

create view [StockGroup] as

select StockGroupName, [2013], [2014], [2015], [2016], [2017]

from

(

select sg.StockGroupName, sisg.StockGroupID, ol.Quantity, DATEPART(year, O.OrderDate) as OrderDate

from Warehouse.StockItems as si join Warehouse.StockItemStockGroups as sisg on sisg.StockItemID = si.StockItemID

join Warehouse.StockGroups as sg on SISG.StockGroupID = sg.StockGroupID

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

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

where ol.Quantity > 0) s

PIVOT

(

sum(s.Quantity)

for s.OrderDate in ([2013], [2014], [2015], [2016], [2017])

) as pt

Table

Description automatically generated

1. Create a view that shows the total quantity of stock items of each stock group sold (in orders) by year 2013-2017. [Year, Stock Group Name1, Stock Group Name2, Stock Group Name3, … , Stock Group Name10]

create view [StockGroup2] as

select pvt.OrderDate as [Year], [Novelty Items], [Clothing], [Mugs], [T-Shirts],

isnull([Airline Novelties], 0) as [Airline Novelties], [Computing Novelties], [USB Novelties], [Furry Footwear], [Toys], [Packaging Materials]

from

(select SG.StockGroupName, OL.Quantity, DATEPART(year, O.OrderDate) as OrderDate

from Warehouse.StockItems as SI join Warehouse.StockItemStockGroups as SISG on SISG.StockItemID = SI.StockItemID

join Warehouse.StockGroups as SG on SISG.StockGroupID = SG.StockGroupID

join Sales.OrderLines as OL on OL.StockItemID = SI.StockItemID

join Sales.Orders as O on OL.OrderID = O.OrderID

where OL.Quantity > 0) s

PIVOT

(sum(s.Quantity)

for s.StockGroupName in ([Novelty Items], [Clothing], [Mugs], [T-Shirts],

[Airline Novelties], [Computing Novelties], [USB Novelties],

[Furry Footwear], [Toys], [Packaging Materials])) as pvt

Table

Description automatically generated

1. Create a function, input: order id; return: total of that order. List invoices and use that function to attach the order total to the other fields of invoices.

create function Sales.OrderTotal (@orderid INT)

returns decimal(18,2) AS

begin

return (select sum(OL.Quantity\*OL.UnitPrice) as Total

from Sales.OrderLines as OL

where OL.OrderID = @orderid)

end

Table

Description automatically generated

1. Create a new table called ods.Orders. Create a stored procedure, with proper error handling and transactions, that input is a date; when executed, it would find orders of that day, calculate order total, and save the information (order id, order date, order total, customer id) into the new table. If a given date is already existing in the new table, throw an error and roll back. Execute the stored procedure 5 times using different dates.
2. Create a new table called ods.StockItem. It has following columns: [StockItemID], [StockItemName] ,[SupplierID] ,[ColorID] ,[UnitPackageID] ,[OuterPackageID] ,[Brand] ,[Size] ,[LeadTimeDays] ,[QuantityPerOuter] ,[IsChillerStock] ,[Barcode] ,[TaxRate] ,[UnitPrice],[RecommendedRetailPrice] ,[TypicalWeightPerUnit] ,[MarketingComments] ,[InternalComments], [CountryOfManufacture], [Range], [Shelflife]. Migrate all the data in the original stock item table.

create table StockItem

(

StockItemID int PRIMARY KEY not null,

StockItemName nvarchar(100) not null,

SupplierID int not null,

ColorID int null,

UnitPackageID int not null,

OuterPackageID int not null,

Brand nvarchar(50) null,

Size nvarchar(20) null,

LeadTimeDays int not null,

QuantityPerOuter int not null,

IsChillerStock bit not null,

Barcode nvarchar(50) null,

TaxRate decimal(18,3) not null,

UnitPrice decimal(18,2) not null,

RecommendedRetailPrice decimal(18,2) not null,

TypicalWeightPerUnit decimal(18,3) not null,

MarketingComments nvarchar(max) null,

InternalComments nvarchar(max) null,

CountryOfManufacture nvarchar(20) null,

[Range] nvarchar(100) null,

ShelfLife nvarchar(100) null

)

insert into StockItem

Select

si.StockItemID,

si.StockItemName,

si.SupplierID,

si.ColorID,

si.UnitPackageID,

si.OuterPackageID,

si.Brand,

si.Size,

si.LeadTimeDays,

si.QuantityPerOuter,

si.IsChillerStock,

si.Barcode,

si.TaxRate,

si.UnitPrice,

si.RecommendedRetailPrice,

si.TypicalWeightPerUnit,

si.MarketingComments,

si.InternalComments,

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

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

JSON\_VALUE(SI.CustomFields, '$.ShelfLife') as ShelfLife

from Warehouse.StockItems as si

Table

Description automatically generated

1. Rewrite your stored procedure in (21). Now with a given date, it should wipe out all the order data prior to the input date and load the order data that was placed in the next 7 days following the input date.

CREATE PROCEDURE ods.NewOrderTotalOfDate

@OrderDate DATE

AS

BEGIN TRANSACTION

DELETE FROM ods.Orders

WHERE OrderDate < @OrderDate

COMMIT

BEGIN TRANSACTION

MERGE ods.Orders T

USING (

SELECT o.OrderID, o.OrderDate, f.Total, o.CustomerID

FROM Sales.Orders o

CROSS APPLY Sales.OrderTotal(OrderID) f

WHERE DATEDIFF(d, @OrderDate, OrderDate) BETWEEN 1 AND 7

) R

ON T.OrderID = R.OrderID

WHEN NOT MATCHED

THEN INSERT VALUES (R.OrderID, R.OrderDate, R.Total, R.CustomerID);

COMMIT

Table

Description automatically generated

1. Consider the JSON file:

{

"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"

}

]

}

Looks like that it is our missed purchase orders. Migrate these data into Stock Item, Purchase Order and Purchase Order Lines tables. Of course, save the script.

DECLARE @json NVARCHAR(max) = 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"

}

]

}';

with a as

(Select \*

FROM OPENJSON(@json, '$.PurchaseOrders')

WITH (

StockItemID int,

StockItemName nvarchar(100),

SupplierID int '$.Supplier',

ColorID int,

UnitPackageID int '$.UnitPackageId',

OuterPackageID int '$.OuterPackageId',

Brand nvarchar(50),

Size nvarchar(20),

LeadTimeDays int,

QuantityPerOuter int,

IsChillerStock bit,

Barcode nvarchar(50),

TaxRate decimal(18,3),

UnitPrice decimal(18,2),

RecommendedRetailPrice decimal(18,2),

TypicalWeightPerUnit decimal(18,3),

MarketingComments nvarchar(max),

InternalComments nvarchar(max),

Photo varbinary(max),

CustomFields nvarchar(max),

Tags nvarchar(max),

SearchDetails nvarchar(max),

LastEditedBy int,

ValidFrom datetime2(7),

ValidTo datetime2(7)))

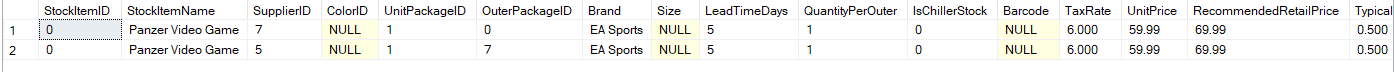
INSERT INTO Warehouse.StockItems

select isnull(StockItemID, 0) as StockItemID, StockItemName, SupplierID, ColorID,

isnull(UnitPackageID, 0) as UnitPackageID, isnull(OuterPackageID, 0) as OuterPackageID, Brand, Size, isnull(LeadTimeDays, 0) as LeadTimeDays, isnull(QuantityPerOuter, 0) as QuantityPerOuter,

isnull(IsChillerStock, 0) as IsChillerStock, Barcode, TaxRate, UnitPrice, RecommendedRetailPrice, TypicalWeightPerUnit, MarketingComments, InternalComments, Photo, CustomFields, Tags, isnull(SearchDetails, 0) as SearchDetails, isnull(LastEditedBy, 0) as LastEditedBy,

isnull(ValidFrom, GETDATE()) as ValidFrom, isnull(ValidTo, '9999-12-30 23:59:59') as ValidTo from a



with b as

(Select \*

FROM OPENJSON(@json, '$.PurchaseOrders')

WITH (

PurchaseOrderID int,

SupplierID int '$.Supplier',

OrderDate date,

DeliveryMethodID int,

ContactPersonID int,

ExpectedDeliveryDate date '$.ExpectedDeliveryDate',

SupplierReference nvarchar(20),

IsOrderFinalized bit,

Comments nvarchar(max),

InternalComments nvarchar(max),

LastEditedBy int,

LastEditedWhen datetime2))

INSERT INTO Purchaing.PurchaseOrderLines

Select isnull(PurchaseOrderID, 0) as PurchaseOrderID, SupplierID, isnull(OrderDate, 0) as OrderDate,

isnull(DeliveryMethodID, 0) as DeliveryMethodID, isnull(ContactPersonID, 0) as ContactPersonID,

ExpectedDeliveryDate, SupplierReference, isnull(IsOrderFinalized, 0) as IsOrderFinalized,

Comments, InternalComments, isnull(LastEditedBy, 0) as LastEditedBy,

isnull(LastEditedWhen, GETDATE()) as LastEditedWhen from b

with c as

(Select \*

FROM OPENJSON(@json, '$.PurchaseOrders')

WITH (

PurchaseOrderLineID int,

PurchaseOrderID int,

StockItemID int,

OrderedOuters int,

[Description] nvarchar(100),

ReceivedOuters int,

PackageTypeID int,

ExpectedUnitPricePerOuter decimal(18,2),

LastReceiptDate date,

IsOrderLineFinalized bit,

LastEditedBy int,

LastEditedWhen datetime2))

Select isnull(PurchaseOrderLineID, 0) as PurchaseOrderLineID, isnull(PurchaseOrderID, 0) as PurchaseOrderID,

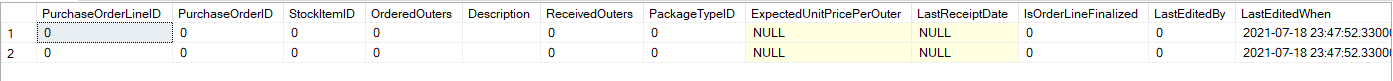
isnull(StockItemID, 0) as StockItemID, isnull(OrderedOuters, 0) as OrderedOuters,

isnull([Description], '') as [Description], isnull(ReceivedOuters, 0) as ReceivedOuters,

isnull(PackageTypeID, 0) as PackageTypeID, ExpectedUnitPricePerOuter, LastReceiptDate,

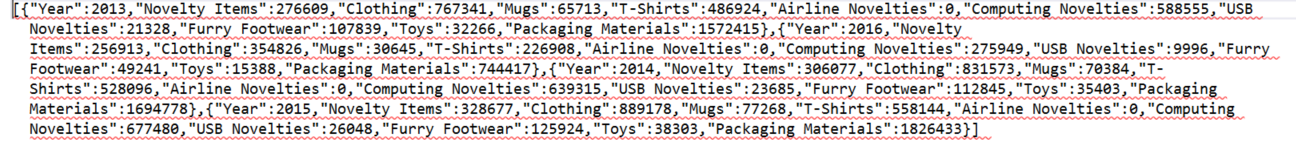
isnull(IsOrderLineFinalized, 0) as IsOrderLineFinalized, isnull(LastEditedBy, 0) as LastEditedBy,

isnull(LastEditedWhen, GETDATE()) as LastEditedWhen from c



1. Revisit your answer in (19). Convert the result in JSON string and save it to the server using TSQL FOR JSON PATH.

select \* from dbo.[StockGroup2] for json auto



1. Revisit your answer in (19). Convert the result into an XML string and save it to the server using TSQL FOR XML PATH.

select Year AS '@Year',

[Novelty Items] AS NoveltyItems,

[Clothing],

[Mugs],

[T-Shirts],

[Airline Novelties] AS AirlineNovelties,

[Computing Novelties] AS ComputingNovelties,

[USB Novelties] AS USBNovelties,

[Furry Footwear] AS FurryFootwear,

[Toys],

[Packaging Materials] AS PackagingMaterials

from dbo.[StockGroup2]

FOR XML PATH('StockItems')

Timeline

Description automatically generated with medium confidence