1.

Select P.SearchName as PersonName, P.FaxNumber as PersonFax, P.PhoneNumber as PersonNumber, S.PhoneNumber as CompanyNumber, S.FaxNumber as CompanyFax

From WideWorldImporters.Application.People\_Archive as P

JOIN WideWorldImporters.Purchasing.Suppliers\_Archive as S

on P.PersonID = S.PrimaryContactPersonID or P.PersonID = S.AlternateContactPersonID

2.

Select DISTINCT(CustomerName) from

Sales.Customers right join

(Select P.PhoneNumber

from Sales.Customers as C

join Application.People as P on C.PrimaryContactPersonID = P.PersonID) as contact

on Sales.Customers.PhoneNumber = contact.PhoneNumber

where CustomerCategoryID != '1'



3.

Select distinct(C.CustomerName)

from Sales.CustomerTransactions as S

join Sales.Customers as C on C.CustomerID = S.CustomerID

where S.TransactionAmount > 0 and S.TransactionDate < '2016-01-01' and S.CustomerID not in

(Select distinct(S1.CustomerID) from Sales.CustomerTransactions as S1

where S1.TransactionAmount > 0 and S1.TransactionDate >= '2016-01-01')

4.

Select I.StockItemName, ABS(stock.TotalAmount) as TotalAmount from

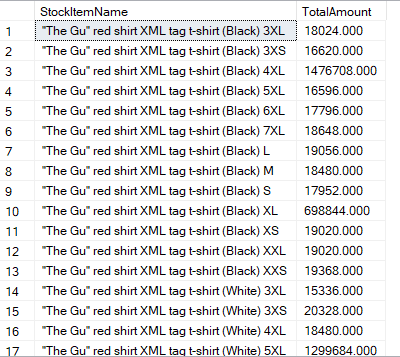
(SELECT StockItemID, sum(Quantity) as TotalAmount, DATEPART(year, TransactionOccurredWhen) as 'Year'

FROM Warehouse.StockItemTransactions as S

group by S.StockItemID, DATEPART(year, S.TransactionOccurredWhen)

having DATEPART(year, S.TransactionOccurredWhen) = 2013) as stock

join Warehouse.StockItems as I on I.StockItemID = stock.StockItemID



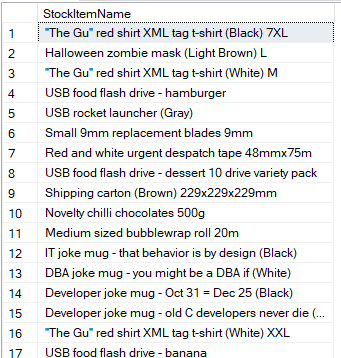
5.

SELECT DISTINCT(S.StockItemName)

FROM Warehouse.StockItems as S

join Sales.OrderLines as O on S.StockItemID = O.StockItemID

where LEN(O.Description) >= 10



6.

Select DISTINCT(SI.StockItemName)

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

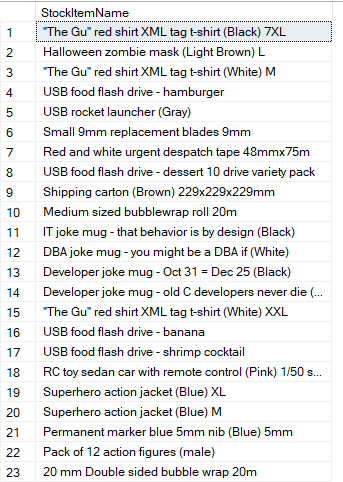
join Sales.Customers as C on C.CustomerID = O.CustomerID

join Application.Cities as City on C.PostalCityID = City.CityID

join Application.StateProvinces as S on S.StateProvinceID = City.StateProvinceID

join Warehouse.StockItems as SI on SI.StockItemID = OL.StockItemID

where DATEPART(year, O.OrderDate) = 2014 and S.StateProvinceName not in ('Alabama', 'Georgia')



7.

Select S.StateProvinceName, AVG(DATEDIFF(day, O.OrderDate, I.ConfirmedDeliveryTime)) as AVG\_Time

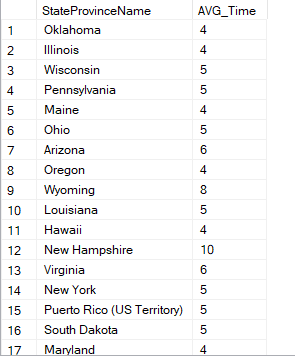
from Sales.Orders as O join Sales.Invoices as I on O.OrderID = I.OrderID

join Sales.Customers as C on C.CustomerID = I.CustomerID

join Application.Cities as City on C.PostalCityID = City.CityID

join Application.StateProvinces as S on S.StateProvinceID = City.StateProvinceID

group by S.StateProvinceName



8.

Select S.StateProvinceName, DATEPART(month, O.OrderDate) as 'Month', AVG(DATEDIFF(day, O.OrderDate, I.ConfirmedDeliveryTime)) as AVG\_Time

from Sales.Orders as O join Sales.Invoices as I on O.OrderID = I.OrderID

join Sales.Customers as C on C.CustomerID = I.CustomerID

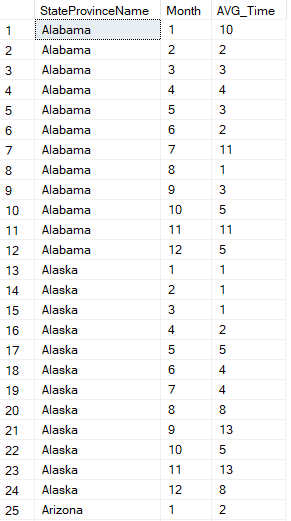
join Application.Cities as City on C.PostalCityID = City.CityID

join Application.StateProvinces as S on S.StateProvinceID = City.StateProvinceID

group by S.StateProvinceName, DATEPART(month, O.OrderDate)

having S.StateProvinceName is not null

order by S.StateProvinceName, DATEPART(month, O.OrderDate)



9.

Select Distinct(SI.StockItemName)

from Purchasing.PurchaseOrders as PO

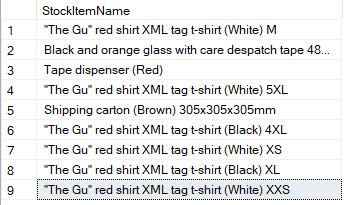
join Purchasing.PurchaseOrderLines as POL on PO.PurchaseOrderID = POL.PurchaseOrderID

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

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

join Warehouse.StockItems as SI on SI.StockItemID = OL.StockItemID

where POL.OrderedOuters-OL.Quantity>0 and DATEPART(year, PO.OrderDate) = 2015 and DATEPART(year, PO.OrderDate) = DATEPART(year, O.OrderDate)



10.

SELECT DISTINCT(C.CustomerName), C.PhoneNumber, P.FullName as ContactPerson

from Sales.Orders as O

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

join Warehouse.StockItems as SI on SI.StockItemID = OL.StockItemID

join Sales.Customers as C on C.CustomerID = O.CustomerID

join Application.People as P on P.PersonID = C.PrimaryContactPersonID

where SI.StockItemName like '%mug%' and DATEPART(year, O.OrderDate) = '2016'

group by C.CustomerName, OL.Quantity, C.PhoneNumber, P.FullName

having sum(OL.Quantity) < 10

Graphical user interface, application

Description automatically generated with medium confidence

11.

DECLARE @CurrentDate date = getDate()

SELECT CityName

from Application.Cities for SYSTEM\_TIME FROM '2015-01-01' to @CurrentDate

Group by CityName

Having count(\*) > 1



12.

Select SI.StockItemName, I.DeliveryInstructions, S.StateProvinceName, City.CityName, CT.CountryName,

C.CustomerName, P.FullName as ContactName, C.PhoneNumber, OL.Quantity

from Sales.Orders as O join Sales.Invoices as I on O.OrderID = I.OrderID

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

join Warehouse.StockItems for SYSTEM\_TIME as of '2014-07-01' as SI on SI.StockItemID = OL.StockItemID

join Sales.Customers as C on C.CustomerID = I.CustomerID

join Application.Cities for SYSTEM\_TIME as of '2014-07-01' as City on C.PostalCityID = City.CityID

join Application.StateProvinces for SYSTEM\_TIME as of '2014-07-01' as S on S.StateProvinceID = City.StateProvinceID

join Application.Countries for SYSTEM\_TIME as of '2014-07-01' as CT on CT.CountryID = S.CountryID

join Application.People for SYSTEM\_TIME as of '2014-07-01' as P on P.PersonID = C.PrimaryContactPersonID

where O.OrderDate = '2014-07-01'



13.

With P 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),

S 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 P join S on P.StockGroupID = S.StockGroupID

order by P.StockGroupID

Table

Description automatically generated

14.

With CC as

(Select DISTINCT(CityName) from Application.Cities as City

join Application.StateProvinces as SP on SP.StateProvinceID = City.StateProvinceID

join Application.Countries as CC on CC.CountryID = SP.CountryID

where CC.CountryName = 'United States'),

R as (Select temp.CityName, ISNULL(temp.StockItemName, 'No Sales') as StockName from

(Select City.CityName, sum(OL.Quantity) as TotalAmount, SI.StockItemName, Rank() over (partition by City.CityName order by sum(OL.Quantity) desc) ranking

from Sales.Orders as O join Sales.Invoices as I on O.OrderID = I.OrderID

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

join Warehouse.StockItems as SI on SI.StockItemID = OL.StockItemID

join Sales.Customers as C on C.CustomerID = I.CustomerID

join Application.Cities as City on C.PostalCityID = City.CityID

join Application.StateProvinces as SP on SP.StateProvinceID = City.StateProvinceID

join Application.Countries as CC on CC.CountryID = SP.CountryID

where CC.CountryName = 'United States' and DATEPART(year, O.OrderDate) = 2016

group by City.CityName, SI.StockItemName) temp

where temp.ranking = 1)

Select CC.CityName, ISNULL(R.StockName, 'No Sales') as StockItem from CC left join R on R.CityName = CC.CityName

Graphical user interface, text

Description automatically generated

15.

With O as

(Select I.CustomerPurchaseOrderNumber, count(I.CustomerPurchaseOrderNumber) as 'Count'

from Sales.Invoices as I

group by I.CustomerPurchaseOrderNumber

having count(I.CustomerPurchaseOrderNumber) > 1)

Select I.OrderID, O.Count

from O join Sales.Invoices as I on O.CustomerPurchaseOrderNumber = I.CustomerPurchaseOrderNumber

Chart

Description automatically generated with medium confidence

16.

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

from Warehouse.StockItems as SI

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

Text

Description automatically generated with low confidence

17.

Select JSON\_VALUE(SI.CustomFields, '$.CountryOfManufacture') as Country, sum(SIT.Quantity) as TotalSale

from Warehouse.StockItems as SI

join Warehouse.StockItemTransactions as SIT on SI.StockItemID = SIT.StockItemID

where DATEPART(year, SIT.TransactionOccurredWhen) = 2015

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

Table

Description automatically generated

18.

CREATE VIEW [Sale of Stock Group] 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 pvt

order by pvt.StockGroupID

Table

Description automatically generated

19.

CREATE VIEW [Sale of Stock Group] as

Select OrderDate, [Novelty Items], [Clothing], [Mugs],

[T-Shirts], [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], [Computing Novelties], [USB Novelties], [Furry Footwear], [Toys], [Packaging Materials])) as pvt

order by pvt.OrderDate

Graphical user interface

Description automatically generated with low confidence

20.

CREATE FUNCTION Sales.AttachOrder (

@long INT

)

RETURNS DECIMAL(18,2) AS

Begin

RETURN (Select t.Total from

(Select OL.OrderID as OID, sum(OL.Quantity\*OL.UnitPrice) as Total

from Sales.OrderLines as OL

where OL.OrderID = @long

group by OL.OrderID)t

join Sales.Invoices as I on I.OrderID = t.OID)

END

A picture containing table

Description automatically generated

21.

CREATE PROCEDURE Sales.OrderCaculate

@Date datetime2

AS

SET NOCOUNT OFF;

BEGIN TRY

BEGIN TRANSACTION

INSERT INTO ods.StockItem

Select O.OrderID, O.OrderDate, sum(OL.Quantity\*OL.UnitPrice) AS TotalOrder, O.CustomerID

from Sales.Orders as O

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

where O.OrderDate = @Date;

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

END CATCH

22.

CREATE TABLE ods.StockItem

(

StockItemID int 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] int not null,

Shelflife int not null

)

23.

CREATE PROCEDURE Sales.OrderCaculate

@Date datetime2

AS

SET NOCOUNT OFF;

BEGIN TRY

BEGIN TRANSACTION

INSERT INTO ods.StockItem

Select O.OrderID, O.OrderDate, sum(OL.Quantity\*OL.UnitPrice) AS TotalOrder, O.CustomerID

from Sales.Orders as O

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

where O.OrderDate BETWEEN DATEADD(day, 1, @Date) AND DATEADD(day, 8, @Date);

DELETE FROM ods.StockItem

where ods.StockItem.OrderDate < @Date

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

END CATCH

24.

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"

}

]

}';

INSERT INTO Warehouse.StockItems

Select \*

FROM OPENJSON(@json)

WITH (StockItemName nvarchar(100),

Supplier int,

UnitPackageId int,

OuterPackageId int,

Brand nvarchar(50),

LeadTimeDays int,

QuantityPerOuter int,

TaxRate decimal(18,3),

UnitPrice decimal(18,2),

RecommendedRetailPrice decimal(18,2),

TypicalWeightPerUnit decimal(18,3))

INSERT INTO Purchasing.PurchaseOrders

Select \*

FROM OPENJSON(@json)

WITH (Supplier int,

ExpectedDeliveryDate date)

INSERT INTO Purchasing.PurchaseOrderLines

Select \*

FROM OPENJSON(@json)

WITH (StockItemName nvarchar(100))

25.

Select OrderDate, [Novelty Items], [Clothing], [Mugs],

[T-Shirts], [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], [Computing Novelties], [USB Novelties], [Furry Footwear], [Toys], [Packaging Materials])) as pvt

order by pvt.OrderDate

FOR JSON AUTO

26.

Select OrderDate, [Novelty Items], [Clothing], [Mugs],

[T-Shirts], [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], [Computing Novelties], [USB Novelties], [Furry Footwear], [Toys], [Packaging Materials])) as pvt

order by pvt.OrderDate

FOR XML AUTO

27.

CREATE TABLE ods.ConfirmedDeliveryJson

(

id int not null,

[date] datetime2 null,

value int null

)

CREATE FUNCTION Sales.Invoices.JsonTransform (

@date datetime2

)

RETURNS TABLE AS

RETURN

Select I.\*, IL.\* from Sales.Invoices as I

join Sales.InvoiceLines as IL on I.InvoiceID = IL.InvoiceID

where @date= I.ConfirmedDeliveryTime

FOR JSON AUTO

CREATE PROCEDURE Sales.DeliverySummary

@Date datetime2

AS

SET NOCOUNT OFF;

BEGIN TRY

BEGIN TRANSACTION

INSERT INTO ods.ConfirmedDeliveryJson

Select I.InvoiceID as 'id', I.ConfirmedDeliveryTime as 'date',

Sales.Invoices.JsonTransform(I.ConfirmedDeliveryTime) as 'value'

from Sales.Invoices as I

join Sales.InvoiceLines as IL on I.InvoiceID = IL.InvoiceID

where @Date = I.ConfirmedDeliveryTime

COMMIT TRANSACTION

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION

END CATCH

Query:

….

28.

Transaction is like a protected wall for any process we want to make for the database. We can have any tries for the database and do not need to worry about harming it by using the transaction because we can roll back if there are any errors happening. On the other hand, if the process is running smoothly, it will be automatically committed.

29.

The first thing I will do is to email boss that I will investigate the issue immediately. Then I will try to simulate the error on the website first and see how slow the website it. After checking the details of error, I will stop the service first and have a basic guess. Then I will go to the relevant database and check the issue through the error log. There are several common errors we have for database related issue, and I will try to figure out which one it is. Finally, I will estimate the time of fixing the errors and give the schedule to my boss. For the short term, I can think about a hot fix if the user wants the error fix as soon as possible but for the long term, I have to think the source of error and design a stable solution for that to avoid happening similar issue in future.