SQL Assignments

SQL related assignments will be on the Wide World Importers Database unless otherwise mentioned.

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, FaxNumber,PhoneNumber, SUBSTRING(EmailAddress,CHARINDEX('@', EmailAddress)+1 , CHARINDEX('.', EmailAddress )) AS company

From Application.People

JOIN Sales.Customers

-- WHERE EmailAddress IS NOT NULL

Sorry this question is sooooo hard, I don’t know how to match the email and the website URL..

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

SELECT CustomerName

FROM

(

SELECT PersonID, PhoneNumber FROM Application.People

)

AS People

JOIN

(

SELECT CustomerName, PrimaryContactPersonID, PhoneNumber FROM Sales.Customers

) as Company

on People.PersonID = Company.PrimaryContactPersonID where People.PhoneNumber = Company.PhoneNumber;

图形用户界面, 应用程序

描述已自动生成

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

SELECT t.CustomerID , c.CustomerName

FROM Sales.CustomerTransactions t

JOIN Sales.Customers c ON t.CustomerID = c.CustomerID

GROUP BY t.CustomerID, c.CustomerName , t.TransactionDate

HAVING max(TransactionDate) < '2016-01-01'

表格

描述已自动生成

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

SELECT p.StockItemID, SUM(Quantity) AS Total\_quantity

FROM Purchasing.PurchaseOrderLines p

JOIN Warehouse.StockItemTransactions s ON p.StockItemID = s.StockItemID

WHERE YEAR(LastReceiptDate) = 2013

GROUP BY p.StockItemID

表格

描述已自动生成

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

SELECT DISTINCT(StockItemName)

FROM Warehouse.StockItems s

JOIN Sales.OrderLines o ON s.StockItemID = o.StockItemID

WHERE LEN(o.Description) >= 10

文本

描述已自动生成

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

SELECT DISTINCT(s.StockItemName)

FROM Warehouse.StockItemTransactions t

JOIN Warehouse.StockItems s ON s.StockItemID = t.StockItemID

JOIN Sales.Invoices i ON t.InvoiceID = i.InvoiceID

JOIN Sales.Customers c ON c.CustomerID = i.CustomerID

JOIN Application.Cities a ON a.CityName = c.PostalAddressLine2

JOIN Application.StateProvinces p ON p.StateProvinceID = a.StateProvinceID

WHERE YEAR(t.TransactionOccurredWhen) = 2014 AND p.StateProvinceName != 'Alabama' AND p.StateProvinceName != 'Georgia'

图形用户界面, 文本

描述已自动生成

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

SELECT StateProvinceName, AVG(DATEDIFF(day, OrderDate, ConfirmedDeliveryTime)) AS Avg\_Processing

FROM Sales.Orders o

JOIN Sales.Invoices i ON o.OrderID = i.OrderID

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

JOIN Application.Cities a ON a.CityName = c.PostalAddressLine2

RIGHT JOIN Application.StateProvinces p ON p.StateProvinceID =a.StateProvinceID

GROUP BY StateProvinceName

表格

描述已自动生成

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

SELECT StateProvinceName, MONTH(OrderDate) AS MONTH ,AVG(DATEDIFF(day, OrderDate,ConfirmedDeliveryTime)) AS Avg\_Processing

FROM Sales.Orders o

JOIN Sales.Invoices i ON o.OrderID = i.OrderID

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

JOIN Application.Cities a ON a.CityName = c.PostalAddressLine2

JOIN Application.StateProvinces p ON p.StateProvinceID = a.StateProvinceID

GROUP BY StateProvinceName, MONTH(OrderDate)

ORDER BY MONTH(OrderDate),StateProvinceName

表格

描述已自动生成

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

SELECT p.StockItemID

FROM Purchasing.PurchaseOrderLines p

JOIN Sales.OrderLines s ON s.StockItemID = p.StockItemID

WHERE YEAR(LastReceiptDate) = '2015'

GROUP BY p.StockItemID

HAVING SUM(OrderedOuters) > SUM(s.Quantity)

表格

描述已自动生成

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 c.CustomerName,c.PhoneNumber,pp.FullName AS Contact

FROM Sales.Invoices i

JOIN Sales.Customers c ON c.CustomerID = i.CustomerID

LEFT JOIN Warehouse.StockItemTransactions t ON t.InvoiceID = i.InvoiceID

JOIN Warehouse.StockItems s ON s.StockItemID = t.StockItemID

JOIN Application.People pp ON pp.PersonID = c.PrimaryContactPersonID

WHERE StockItemName LIKE '%mug%'

GROUP BY c.CustomerName,c.PhoneNumber,pp.FullName

Having SUM(t.Quantity) <-10

图形用户界面, 应用程序

描述已自动生成

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

SELECT CityID

FROM Application.Cities

WHERE ValidFrom > '2015-01-01'

图形用户界面, 应用程序

描述已自动生成

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 i.OrderID, s.StockItemName,c.CustomerName,C.DeliveryAddressLine1,c.DeliveryAddressLine2,p.StateProvinceName,a.CityName,g.CountryName,pp.FullName AS Contact,C.PhoneNumber,T.Quantity

FROM Sales.Invoices i

LEFT JOIN Warehouse.StockItemTransactions t ON i.InvoiceID = t.InvoiceID

JOIN Warehouse.StockItems s ON s.StockItemID = t.StockItemID

JOIN Sales.Customers c ON c.CustomerID = i.CustomerID

JOIN Application.Cities a ON a.CityID = c.DeliveryCityID

JOIN Application.StateProvinces p ON p.StateProvinceID = a.StateProvinceID

JOIN Application.Countries g ON g.CountryID = p.CountryID

JOIN Application.People pp ON pp.PersonID = c.PrimaryContactPersonID

WHERE InvoiceDate = '2014-07-01'

图形用户界面, 文本, 应用程序

描述已自动生成

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

SELECT g.StockGroupID ,SUM(CAST(p.OrderedOuters AS BIGINT)) AS purchased,SUM(CAST(s.Quantity AS BIGINT)) AS Sold, SUM(CAST(p.OrderedOuters AS BIGINT))-SUM(CAST(s.Quantity AS BIGINT)) AS Remained

FROM Purchasing.PurchaseOrderLines p

JOIN Sales.OrderLines s ON s.StockItemID = p.StockItemID

JOIN Warehouse.StockItemStockGroups g ON g.StockItemID = p.StockItemID

GROUP BY g.StockGroupID

表格

描述已自动生成

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 CTE (Stock\_Item\_Name, City\_ID, Total\_Sales, Ranking)

AS(

SELECT OL.Description AS Stock\_Item\_Name, C.PostalCityID, SUM(OL.Quantity) AS Total\_Sale,

RANK() OVER (PARTITION BY C.PostalCityID ORDER BY SUM(OL.Quantity) DESC) AS RK

FROM Sales.OrderLines AS OL

JOIN Sales.Orders AS O ON O.OrderID = OL.OrderID

JOIN Sales.Customers AS C ON C.CustomerID = O.CustomerID

WHERE YEAR(O.OrderDate) = 2016

GROUP BY OL.Description, C.PostalCityID

)

SELECT CT.CityName,

ISNULL(CTE.Stock\_Item\_Name, 'No Sales') AS Stock\_Item\_Name

FROM CTE

JOIN Application.Cities AS CT ON CT.CityID = CTE.City\_ID

JOIN Application.StateProvinces AS SP ON SP.StateProvinceID = CT.StateProvinceID

JOIN Application.Countries AS CY ON CY.CountryID = SP.CountryID

WHERE CY.CountryName = 'United States'

AND CTE.Ranking = 1

GROUP BY CT.CityName, CTE.Stock\_Item\_Name

图形用户界面, 应用程序, 表格, Excel

描述已自动生成

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

SELECT I.OrderID

FROM Sales.Invoices AS I

WHERE JSON\_VALUE(I.ReturnedDeliveryData, '$.Events[1].Status') IS NULL

表格

低可信度描述已自动生成

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

SELECT StockItemID

FROM Warehouse.StockItems

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

表格

描述已自动生成

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

SELECT JSON\_VALUE(CustomFields, '$.CountryOfManufacture') AS Country, SUM(Quantity) AS Total\_Quantity

FROM Sales.InvoiceLines i

JOIN Sales.Invoices ii ON ii.InvoiceID = i.InvoiceID

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

JOIN Warehouse.StockItems s ON i.StockItemID = s.StockItemID

WHERE YEAR(o.OrderDate) = 2015

GROUP BY JSON\_VALUE(CustomFields, '$.CountryOfManufacture')

图形用户界面, 应用程序, 表格

描述已自动生成

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 Sales.Sales\_for\_stockgroup

AS

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

FROM

(

SELECT sg.StockGroupName,YEAR(od.OrderDate) AS 'Year' ,o.Quantity

FROM Sales.OrderLines o

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

JOIN Warehouse.StockItemStockGroups g ON o.StockItemID =g.StockItemID

RIGHT JOIN Warehouse.StockGroups sg ON sg.StockGroupID = g.StockGroupID

WHERE YEAR(od.OrderDate) IN (2013,2014,2015,2016,2017)

)AS SourceTable

PIVOT

(

SUM(Quantity)

For Year IN ( [2013],[2014],[2015],[2016],[2017])

) AS Pivottable

图形用户界面, 应用程序, 表格, Excel

描述已自动生成

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]

Feel like my 18 is wrong..so I give up 19

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.total\_price(@orderid int)

RETURNS int

AS

BEGIN

DECLARE @tt int

SELECT @tt = SUM(o.Quantity \* o.UnitPrice)

FROM Sales.OrderLines o

WHERE OrderID = @orderid

RETURN @tt

END;

SELECT InvoiceID, OrderID, Sales.total\_price(InvoiceID) AS Total

FROM Sales.Invoices;

表格

描述已自动生成

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.

GO

CREATE SCHEMA ods;

GO

DROP TABLE IF EXISTS ods.Orders;

GO

CREATE TABLE ods.Orders(

orderID int PRIMARY KEY,

orderDate date,

orderTotal float,

customerID int);

DROP PROCEDURE IF EXISTS p\_DayOrders;

GO

CREATE PROCEDURE p\_DayOrders @Day date

AS BEGIN

BEGIN TRY

BEGIN TRANSACTION

INSERT INTO ods.Orders

SELECT S.OrderID, S.OrderDate, SUM(Quantity\*UnitPrice)OrderTotal, S.CustomerID

FROM Sales.Orders S LEFT JOIN Sales.OrderLines OL ON S.OrderID=OL.OrderID

WHERE S.OrderDate=@Day

GROUP BY S.OrderID, S.OrderDate,S.CustomerID;

COMMIT

END TRY

BEGIN CATCH

SELECT ERROR\_MESSAGE() AS ErrorMessage;

ROLLBACK;

END CATCH

END

GO

EXEC p\_DayOrders @day = '2013-01-01';

EXEC p\_DayOrders @day = '2014-01-12';

EXEC p\_DayOrders @day = '2014-02-04';

EXEC p\_DayOrders @day = '2015-11-12';

EXEC p\_DayOrders @day = '2015-11-13';

EXEC p\_DayOrders @day = '2015-11-13';

SELECT \* FROM ods.Orders;

表格

描述已自动生成

1. 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.

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, NULL AS Range, NULL AS Shelflife

INTO ods.StockItem

FROM Warehouse.StockItems

SELECT \* FROM ods.StockItem

图形用户界面, 应用程序, 表格

描述已自动生成

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.

GO

ALTER PROCEDURE p\_DayOrders @day date

AS

DELETE FROM ods.Orders WHERE OrderDate < @day;

INSERT INTO ods.Orders

SELECT S.OrderID, S.OrderDate, SUM(Quantity\*UnitPrice)OrderTotal, S.CustomerID

FROM Sales.Orders S LEFT JOIN Sales.OrderLines OL ON S.OrderID=OL.OrderID

WHERE S.OrderDate>@Day AND S.OrderDate<=DATEADD(day,7,@Day)

GROUP BY S.OrderID, S.OrderDate,S.CustomerID;

GO

EXEC p\_DayOrders @day='2015-01-12';

SELECT \* FROM ods.Orders

表格

描述已自动生成

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.

1. Revisit your answer in (19). Convert the result in JSON string and save it to the server using TSQL FOR JSON PATH.
2. Revisit your answer in (19). Convert the result into an XML string and save it to the server using TSQL FOR XML PATH.
3. Create a new table called ods.ConfirmedDeviveryJson with 3 columns (id, date, value) . Create a stored procedure, input is a date. The logic would load invoice information (all columns) as well as invoice line information (all columns) and forge them into a JSON string and then insert into the new table just created. Then write a query to run the stored procedure for each DATE that customer id 1 got something delivered to him.
4. Write a short essay talking about your understanding of transactions, locks and isolation levels.

Transactions, locks and isolation levels play an important role in maintaining the concurrency for the Database. There are several types of isolation Levels which can prevent different kinds of concurrency issues. Locks are be used differently in different isolation levels. Sometimes we will meet deadlock problem which happens on multiple transactions, but it can be resolved automatically.

1. Write a short essay, plus screenshots talking about performance tuning in SQL Server. Must include Tuning Advisor, Extended Events, DMV, Logs and Execution Plan.

Assignments 30 - 32 are group assignments.

1. Write a short essay talking about a scenario: Good news everyone! We (Wide World Importers) just brought out a small company called “Adventure works”! Now that bike shop is our sub-company. The first thing of all works pending would be to merge the user logon information, person information (including emails, phone numbers) and products (of course, add category, colors) to WWI database. Include screenshot, mapping and query.

We create 2 tables to store the information from AW, and we want to merge those table in to WWI

SELECT \* INTO person\_info

FROM(

SELECT a.FirstName,a.LastName,b.PhoneNumber,c.EmailAddress,d.PasswordHash

FROM AdventureWorks2019.Person.Person a

left join AdventureWorks2019.Person.PersonPhone b

on a.BusinessEntityID = b.BusinessEntityID

left join AdventureWorks2019.Person.EmailAddress c

on a.BusinessEntityID =c.BusinessEntityID

left join AdventureWorks2019.Person.Password d

on a.BusinessEntityID = d.BusinessEntityID) AS tt

SELECT \* INTO product\_info

FROM(

SELECT a.color,b.ProductSubcategoryID

FROM AdventureWorks2019.Production.Product a

left join AdventureWorks2019.Production.ProductSubcategory b

on a.ProductSubcategoryID = b.ProductSubcategoryID) AS tb

1. Database Design: OLTP db design request for EMS business: when people call 911 for medical emergency, 911 will dispatch UNITs to the given address. A UNIT means a crew on an apparatus (Fire Engine, Ambulance, Medic Ambulance, Helicopter, EMS supervisor). A crew member would have a medical level (EMR, EMT, A-EMT, Medic). All the treatments provided on scene are free. If the patient needs to be transported, that’s where the bill comes in. A bill consists of Units dispatched (Fire Engine and EMS Supervisor are free), crew members provided care (EMRs and EMTs are free), Transported miles from the scene to the hospital (Helicopters have a much higher rate, as you can image) and tax (Tax rate is 6%). Bill should be sent to the patient insurance company first. If there is a deductible, we send the unpaid bill to the patient only

. Don’t forget about patient information, medical nature and bill paying status.

1. Remember the discussion about those two databases from the class, also remember, those data models are not perfect. You can always add new columns (but not alter or drop columns) to any tables. Suggesting adding Ingested DateTime and Surrogate Key columns. Study the Wide World Importers DW. Think the integration schema is the ODS. Come up with a TSQL Stored Procedure driven solution to move the data from WWI database to ODS, and then from the ODS to the fact tables and dimension tables. By the way, WWI DW is a galaxy schema db. Requirements:
   1. Luckly, we only start with 1 fact: Purchase. Other facts can be ignored for now.
   2. Add a new dimension: Country of Manufacture. It should be given on top of Stock Items.
   3. Write script(s) and stored procedure(s) for the entire ETL from WWI db to DW.