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 p.fullname, p.faxnumber as personlfaxnumber, p.phonenumber as personlphonenumber, cmpphonenumber, cmpfaxnumber
from application.people p
left join
(select p.fullname, p.faxnumber as personlfaxnumber, p.phonenumber as personlphonenumber, s.phonenumber as cmpphonenumber, s.faxnumber as cmpfaxnumber
from application.people p inner join purchasing.suppliers s on (p.personid = s.primarycontactpersonid or p.personid = s.AlternateContactPersonID)
union
select p.fullname, p.faxnumber as personlfaxnumber, p.phonenumber as personlphonenumber, c.phonenumber as cmpphonenumber, c.faxnumber as cmpfaxnumber
from application.people p inner join sales.customers c on (p.personid = c.primarycontactpersonid or p.personid = c.AlternateContactPersonID)
) j
on p.fullname = j.fullname and p.faxnumber = j.cmpfaxnumber and p.phonenumber = j.cmpphonenumber
```

	fullname	personlfaxnumber	personlphonenumber	cmpphonenumber	cmpfaxnumber
1	Data Conversion Only	NULL	NULL	NULL	NULL
2	Kayla Woodcock	(415) 555-0103	(415) 555-0102	NULL	NULL
3	Hudson Onslow	(415) 555-0103	(415) 555-0102	NULL	NULL
4	Isabella Rupp	(415) 555-0103	(415) 555-0102	NULL	NULL
5	Eva Muirden	(415) 555-0103	(415) 555-0102	NULL	NULL
6	Sophia Hinton	(415) 555-0103	(415) 555-0102	NULL	NULL
7	Amy Trefl	(415) 555-0103	(415) 555-0102	NULL	NULL
8	Anthony Grosse	(415) 555-0103	(415) 555-0102	NULL	NULL

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

```
select c.CustomerName
from application.people p inner join sales.customers c on p.personid = c.primarycontactpersonid
where p.phonenumber = c.phonenumber
```

	CustomerName
1	Tailspin Toys (Head Office)
2	Tailspin Toys (Sylvanite, MT)
3	Tailspin Toys (Peeples Valley, AZ)
4	Tailspin Toys (Medicine Lodge, KS)
5	Tailspin Toys (Gasport, NY)
6	Tailspin Toys (Jessie, ND)
7	Tailspin Toys (Frankewing, TN)
8	Tailspin Toys (Bow Mar, CO)

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

```
select max(transactiondate) as lastdate, min(transactiondate) as earliestdate
from sales.CustomerTransactions
group by customerid
having year(max(transactiondate)) < 2016 and min(transactiondate) != null
```

	lastdate	earliestdate

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

```
select stockitemname, total_quant
from
(select ol.Stockitemid, sum(ol.quantity) total_quant
from sales.orderLines ol inner join sales.orders o on ol.orderid = o.orderid
where year(o.orderdate) = 2013
group by ol.stockitemid) j
inner join
warehouse.stockitems s
on j.stockitemid = s.stockitemid
```

	stockitemname	total_quant
1	"The Gu" red shirt XML tag t-shirt (Black) 3XL	18312
2	"The Gu" red shirt XML tag t-shirt (Black) 3XS	16692
3	"The Gu" red shirt XML tag t-shirt (Black) 4XL	19644
4	"The Gu" red shirt XML tag t-shirt (Black) 5XL	16764
5	"The Gu" red shirt XML tag t-shirt (Black) 6XL	17964
6	"The Gu" red shirt XML tag t-shirt (Black) 7XL	18708
7	"The Gu" red shirt XML tag t-shirt (Black) L	19164
8	"The Gu" red shirt XML tag t-shirt (Black) M	18648
9	"The Gu" red shirt XML tag t-shirt (Black) S	18024
10	"The Gu" red shirt XML tag t-shirt (Black) XL	20376
11	"The Gu" red shirt XML tag t-shirt (Black) XS	19092

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

```
select distinct stockitemname
from sales.orderlines ol inner join warehouse.stockitems s on ol.StockItemID = s.StockItemID
where len(description) >= 10
```

	stockitemname
1	DBA joke mug - you might be a DBA if (Black)
2	Ride on vintage American toy coupe (Red) 1/12 s...
3	Developer joke mug - understanding recursion re...
4	Chocolate beetles 250g
5	Halloween skull mask (Gray) S
6	"The Gu" red shirt XML tag t-shirt (White) 3XL
7	Ogre battery-powered slippers (Green) XL
8	RC toy sedan car with remote control (Pink) 1/5...
9	Halloween skull mask (Gray) L
10	Animal with big feet slippers (Brown) XL
11	10 mm Double sided bubble wrap 50m
12	Animal with big feet slippers (Brown) L

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

```
with cte as(
select customerid
from sales.customers
where sales.customers.deliverycityid not in
(select cityid
from application.stateprovinces s inner join application.cities c on s.StateProvinceID = c.StateProvinceID
where s.stateprovincename = 'Alabama' or s.stateprovincename = 'Georgia'
)),
cte1 as
(select distinct stockitemid
from warehouse.stockitemtransactions s inner join cte on s.CustomerID = cte.CustomerID
where year(s.transactionoccurredwhen) = 2014)
select stockitemname
from warehouse.stockitems inner join cte1 on warehouse.stockitems.stockitemid = cte1.StockItemID
```

	stockitemname
1	"The Gu" red shirt XML tag t-shirt (Black) 3XL
2	"The Gu" red shirt XML tag t-shirt (Black) 3XS
3	"The Gu" red shirt XML tag t-shirt (Black) 4XL
4	"The Gu" red shirt XML tag t-shirt (Black) 5XL
5	"The Gu" red shirt XML tag t-shirt (Black) 6XL
6	"The Gu" red shirt XML tag t-shirt (Black) 7XL
7	"The Gu" red shirt XML tag t-shirt (Black) L
8	"The Gu" red shirt XML tag t-shirt (Black) M
9	"The Gu" red shirt XML tag t-shirt (Black) S
10	"The Gu" red shirt XML tag t-shirt (Black) XL
11	"The Gu" red shirt XML tag t-shirt (Black) XS
12	"The Gu" red shirt XML tag t-shirt (Black) XXL
13	"The Gu" red shirt XML tag t-shirt (Black) XXS
14	"The Gu" red shirt XML tag t-shirt (White) 3XL

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

```
/* join one more to have name*/
with cte as(
select i.customerid, datediff(d, o.orderdate, i.confirmeddeliverytime) as processdate
from sales.invoices i inner join sales.orders o on i.orderid = o.orderid
)
select stateprovinceid, avg(processdate) as avgprocessdate
from cte inner join dbo.cuscitystate ccs on cte.CustomerID = ccs.customerid
group by stateprovinceid;
```

	stateprovinceid	avgprocessdate
1	40	5
2	2	5
3	5	5
4	12	4
5	29	3
6	38	4
7	50	3
8	14	4
9	15	7
10	23	6
11	36	5
12	52	5

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

```
/* join one more to have name*/
with cte1 as(
select i.customerid, cast(datediff(d, o.orderdate, i.confirmeddeliverytime) as decimal)/30.00 as processdate
from sales.invoices i inner join sales.orders o on i.orderid = o.orderid
)
select stateprovinceid, avg(processdate) as avgprocessmonth
from cte1 inner join dbo.cuscitystate ccs on cte1.CustomerID = ccs.customerid
group by stateprovinceid;
```

	stateprovinceid	avgprocessmonth
1	40	0.188741
2	2	0.182768
3	5	0.198417
4	12	0.163130
5	29	0.124395
6	38	0.159930
7	50	0.123658
8	14	0.144129
9	15	0.238228
10	23	0.212233
11	36	0.183858
12	52	0.177180

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

```
select *
from
(
select stockitemid, sum(quantity) as totalsale2015
from sales.orderlines ol inner join sales.orders o on ol.OrderID = o.orderid
where year(o.orderdate) = 2015
group by stockitemid
) t right join
(select stockitemid, sum(orderedouters) as totalbuy2015
from purchasing.purchaseorderlines
where year(purchasing.purchaseorderlines.LastReceiptDate) = 2015
group by stockitemid) t1 on t.stockitemid = t1.stockitemid  and totalbuy2015 > totalsale2015
```

	stockitemid	totalsale2015	stockitemid	totalbuy2015
1	86	21120	86	663371
2	204	17220	204	565462
3	98	22116	98	676689
4	184	47725	184	247714
5	77	19104	77	645687
6	95	19992	95	325593
7	78	21972	78	665104
8	80	21936	80	318353
9	193	40368	193	493027

10. 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.

```
with cte as(
select customerid
from
(select orderid
from sales.orderlines ol left join warehouse.stockitems s on ol.StockItemID = s.StockItemID
where s.StockItemName like '%mug%' and ol.quantity <= 10
) t left join sales.orders o on t.orderid = o.orderid
where year(o.orderdate) = 2016
)
select customername, c.phonenumber,  p.FullName
from cte inner join sales.customers c on cte.CustomerID = c.CustomerID
inner join application.people p on c.PrimaryContactPersonID = p.PersonID
```

	customername	phonenumber	FullName
1	Tailspin Toys (Hodgdon, ME)	(207) 555-0100	Rohana Kaskar
2	Tailspin Toys (Sans Souci, SC)	(803) 555-0100	Coralie Emond
3	Shyam Poddar	(218) 555-0100	Shyam Poddar
4	Tailspin Toys (Saint Louis Park, MN)	(218) 555-0100	Beatrise Bite
5	Dipti Shah	(216) 555-0100	Dipti Shah
6	Dhaatri Chavva	(207) 555-0100	Dhaatri Chavva
7	Tailspin Toys (Orrtanna, PA)	(215) 555-0100	Ngai Lam
8	Hai Banh	(252) 555-0100	Hai Banh
9	Emily Whittle	(231) 555-0100	Emily Whittle
10	Tailspin Toys (Tunnelhill, PA)	(215) 555-0100	Bhavani Bhowmick
11	Wingtip Toys (Plaquemine, LA)	(225) 555-0100	Nishant Menon
12	Wingtip Toys (Omer, MI)	(231) 555-0100	Javier Caraballo

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

```sql
select cityname, validfrom
from application.cities
where validfrom < '2015-01-01'
```

	cityname	validfrom
1	Aaronsburg	2013-01-01 00:00:00.0000000
2	Abanda	2013-01-01 00:00:00.0000000
3	Abbeville	2013-01-01 00:00:00.0000000
4	Abbeville	2013-01-01 00:00:00.0000000
5	Abbeville	2013-01-01 00:00:00.0000000
6	Abbeville	2013-01-01 00:00:00.0000000
7	Abbeville	2013-01-01 00:00:00.0000000
8	Abbotsford	2013-01-01 00:00:00.0000000
9	Abbott	2013-01-01 00:00:00.0000000
10	Abbott	2013-01-01 00:00:00.0000000
11	Abbott	2013-01-01 00:00:00.0000000

12. 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.

```sql
with base as(
select orderid
from sales.orders
where orderdate = '2014-07-01')
select base.orderid, customername, fullname as contactpersonname, deliveryinstructions as deliaddress, stockitemid, quantity
from base left join sales.invoices i on base.orderid = i.orderid
left join sales.invoicelines il on i.invoiceid = il.invoiceid
left join sales.customers c on c.customerid = i.customerid
left join application.people p on p.personid = i.contactpersonid
```

	orderid	customername	contactpersonname	deliaddress	stockitemid	quantity
1	29914	Wingtip Toys (Rose Tree, PA)	Lan Chu	Unit 212, 1837 Vasiljevic Boulevard	42	7
2	29937	Wingtip Toys (Rosa Sánchez, PR)	Selma Seppanen	Unit 193, 583 Aluri Road	33	7
3	29917	Tailspin Toys (Lavon, TX)	Alba Ponce	Suite 258, 771 Kidambi Road	44	2
4	29927	Tailspin Toys (Stonefort, IL)	Razeena Hosseini	Suite 185, 1492 Shah Road	28	2
5	29933	Wingtip Toys (East Fultonham, OH)	Masa Buecek	Shop 20, 1046 Saucier Road	41	2
6	29910	Tailspin Toys (Indios, PR)	Roxane Rastgu	Unit 50, 519 Jogi Street	16	2
7	29924	Kertu Sokk	Kertu Sokk	Shop 11, 965 Horackova Lane	21	2
8	29917	Tailspin Toys (Lavon, TX)	Alba Ponce	Suite 258, 771 Kidambi Road	18	9
9	29909	Manca Hrastovsek	Manca Hrastovsek	Unit 15, 685 Pavel Road	36	9
10	29931	Tailspin Toys (Sans Souci, SC)	Coralie Emond	Shop 104, 1889 Smirnov Road	37	9
11	29911	Wingtip Toys (Bell Acres, PA)	Abhra Thakur	Suite 8, 1761 Nastase Avenue	20	9

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

```
with cte as(
select stockgroupid, sum(quantity) as soldQuantities
from warehouse.stockitemstockgroups sg inner join warehouse.stockitemtransactions st on sg.stockitemid = st.stockitemid
where quantity < 0
group by stockgroupid),
cte2 as(
select stockgroupid, sum(quantity) as purchaseQuantities
from warehouse.stockitemstockgroups sg inner join warehouse.stockitemtransactions st on sg.stockitemid = st.stockitemid
where quantity > 0
group by stockgroupid)
select cte.stockgroupid, soldQuantities, purchaseQuantities, purchaseQuantities - soldQuantities*-1 as remain
from cte inner join cte2 on cte.StockGroupID = cte2.StockGroupID
```

	stockgroupid	soldQuantities	purchaseQuantities	remain
1	1	-1168318.000	11018.000	-1157300.000
2	3	-244036.000	1459.000	-242577.000
3	4	-1581499.000	90920724.000	89339225.000
4	9	-121363.000	1316.000	-120047.000
5	2	-2624358.000	90923761.000	88299403.000
6	8	-395858.000	1398.000	-394460.000
7	6	-1962761.000	90923916.000	88961155.000
8	7	-81066.000	1583.000	-79483.000
9	10	-5696362.000	48485036.000	42788674.000

14. 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 as(
select orderid, customerid
from sales.orders
where year(orderdate) = 2016
),
cte2 as(
select stockitemid, cityid, sum(quantity) as totalsold
from cte inner join sales.orderlines ol on cte.orderid = ol.orderid
right join cuscitystate ccs on ccs.customerid = cte.CustomerID
group by stockitemid, cityid
),
cte3 as (
select stockitemid, cityid, max(totalsold) as mostsoldtocity
from cte2
group by stockitemid, cityid
)
select coalesce(stockitemid, 'no sale') as result, c.cityid
from application.cities c left join cte3 on c.cityid = cte3.cityid /*Conversion failed when converting the ****** value '******' to data type ******.*/
```

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

```
adSimulation.ReactivateTemporalTablesAfterDataLoad
from sales.invoices
where JSON_VALUE(ReturnedDeliveryData,'$.Events[1].Event') = 'DeliveryAttempt'
group by orderid
having count(ReturnedDeliveryData) > 1
```

```
orderid
```

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

```
select *
from warehouse.stockitems
where JSON_VALUE(customfields,'$.CountryOfManufacture') = 'China'
```

	StockItemID	StockItemName	SupplierID	ColorID	UnitPackageID	OuterPackageID	Brand	Size	LeadTimeDays	QuantityPerOuter	IsChillerStock
1	1	USB missile launcher (Green)	12	NULL	7	7	NULL	NULL	14	1	0
2	2	USB rocket launcher (Gray)	12	12	7	7	NULL	NULL	14	1	0
3	3	Office cube periscope (Black)	12	3	7	6	NULL	NULL	14	10	0
4	16	DBA joke mug - mind if I join you? (White)	5	35	7	7	NULL	NULL	12	1	0
5	17	DBA joke mug - mind if I join you? (Black)	5	3	7	7	NULL	NULL	12	1	0
6	18	DBA joke mug - daaaaaa-ta (White)	5	35	7	7	NULL	NULL	12	1	0
7	19	DBA joke mug - daaaaaa-ta (Black)	5	3	7	7	NULL	NULL	12	1	0
8	20	DBA joke mug - you might be a DBA if (White)	5	35	7	7	NULL	NULL	12	1	0
9	21	DBA joke mug - you might be a DBA if (Black)	5	3	7	7	NULL	NULL	12	1	0

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

```
with t as(
select stockitemid, quantity
from sales.orders o inner join sales.orderlines ol on o.orderid = ol.orderid
where year(o.orderdate) = 2015
)
select JSON_VALUE(customFields,'$.CountryOfManufacture') as country, sum(quantity) as totalsale
from t inner join warehouse.stockitems s on t.stockitemid = s.stockitemid
group by JSON_VALUE(customFields,'$.CountryOfManufacture')
```

	country	totalsale
1	Japan	22365
2	China	2850885

18. 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 stockgroupsbyyear as
select stockgroupname AS stockGroupSoldByYears,
  [2013], [2014], [2015], [2016], [2017]
FROM
(
  SELECT year(orderdate) as aggyear, quantity, stockgroupname
  FROM sales.orders o inner join sales.orderlines ol on o.orderid = ol.orderid
  inner join warehouse.stockitemstockgroups s on ol.stockitemid = s.stockitemid
  inner join warehouse.stockgroups sg on s.StockGroupID = sg.StockGroupID
) AS SourceTable
PIVOT
(
  sum(quantity)
  FOR aggyear IN ([2013], [2014], [2015], [2016], [2017])
) AS PivotTable
```

19. 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 yearbystockgroup as
select aggyear AS yearsSoldBystockgroup,
  [Novelty Items], [Clothing], [Mugs], [T-Shirts], [Airline Novelties],
  [Computing Novelties], [USB Novelties], [Furry Footwear], [Toys], [Packaging Materials]
FROM
(
  SELECT year(orderdate) as aggyear, quantity, stockgroupname
  FROM sales.orders o inner join sales.orderlines ol on o.orderid = ol.orderid
  inner join warehouse.stockitemstockgroups s on ol.stockitemid = s.stockitemid
  inner join warehouse.stockgroups sg on s.StockGroupID = sg.StockGroupID
) AS SourceTable
PIVOT
(
  sum(quantity)
  FOR stockgroupname IN ([Novelty Items], [Clothing], [Mugs], [T-Shirts], [Airline Novelties],
  [Computing Novelties], [USB Novelties], [Furry Footwear], [Toys], [Packaging Materials])
) AS PivotTable
```

20. 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.ufn_SalesByorder(@argorderid int)
RETURNS decimal
AS
begin
    declare @ret decimal;
    SELECT @ret = quantity*unitprice
    from sales.orderlines
    where orderid = @argorderid
    if (@ret is null)
        set @ret = 0;
    return @ret
end
go
```

```
select i.*, Sales.ufn_SalesByorder(i.orderid) as total
from sales.invoices i
```

21. 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.

22. 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.
23. 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.
24. 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.

25. Revisit your answer in (19). Convert the result in JSON string and save it to the server using TSQL FOR JSON PATH.
26. Revisit your answer in (19). Convert the result into an XML string and save it to the server using TSQL FOR XML PATH.
27. 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.
28. Write a short essay talking about your understanding of transactions, locks and isolation levels.

29. 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.

30. 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.
31. 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.

32. 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:
    a. Luckly, we only start with 1 fact: Order. Other facts can be ignored for now.
    b. Add a new dimension: Country of Manufacture. It should be given on top of Stock Items.
    c. Write script(s) and stored procedure(s) for the entire ETL from WWI db to DW.