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).
2. If the customer's primary contact person has the same phone number as the customer's phone number, list the customer companies.
3. List of customers to whom we made a sale prior to 2016 but no sale since 2016-01-01.
4. List of Stock Items and total quantity for each stock item in Purchase Orders in Year 2013.
5. List of stock items that have at least 10 characters in description.
6. List of stock items that are not sold to the state of Alabama and Georgia in 2014.
7. List of States and Avg dates for processing (confirmed delivery date – order date).
8. List of States and Avg dates for processing (confirmed delivery date – order date) by month.
9. List of StockItems that the company purchased more than sold in the year of 2015.
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.
11. List all the cities that were updated after 2015-01-01.
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.
13. List of stock item groups and total quantity purchased, total quantity sold, and the remaining stock quantity (quantity purchased – quantity sold)
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".
15. List any orders that had more than one delivery attempt (located in invoice table).
16. List all stock items that are manufactured in China. (Country of Manufacture)
17. Total quantity of stock items sold in 2015, group by country of manufacturing.
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]

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]
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.

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.