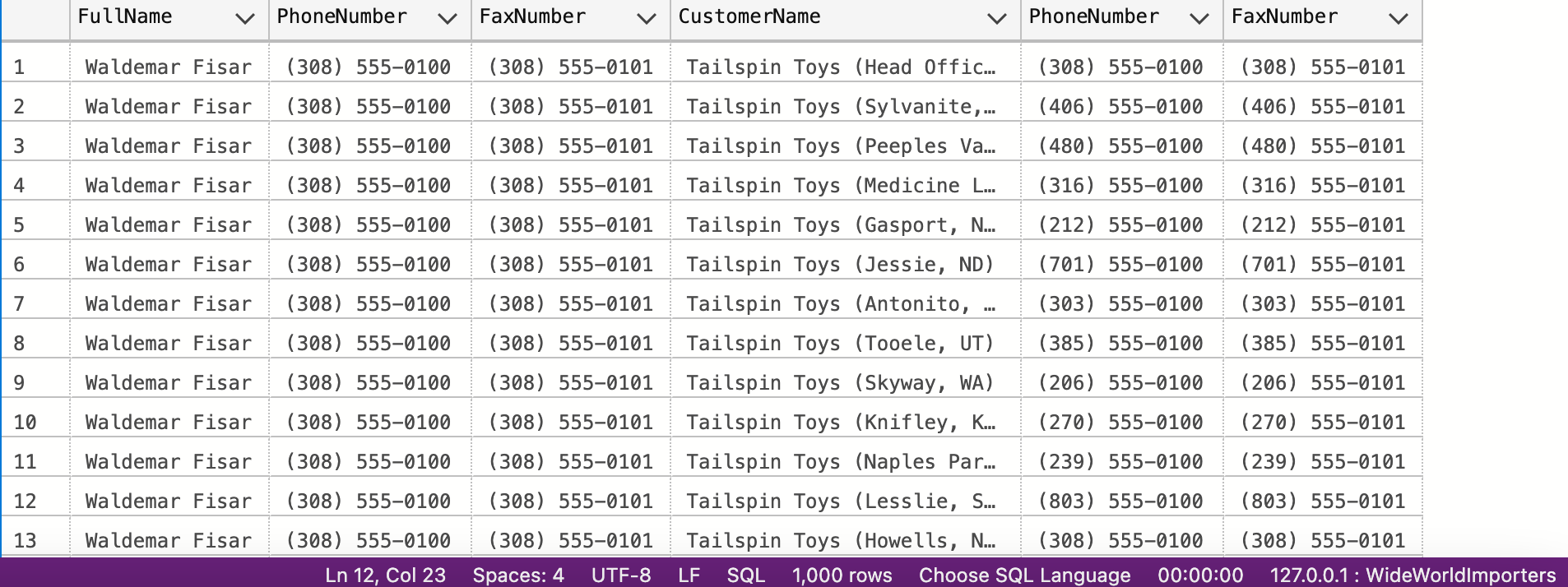
SQL Assignments

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).



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

A picture containing graphical user interface

Description automatically generated

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

Text

Description automatically generated with medium confidence

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

Graphical user interface, application, table

Description automatically generated

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

Table

Description automatically generated with medium confidence

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

Graphical user interface

Description automatically generated with medium confidence

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

Graphical user interface, application

Description automatically generated

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

Table

Description automatically generated

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

Graphical user interface, application, table, Excel

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.

Graphical user interface, application, table, Excel

Description automatically generated

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

A picture containing graphical user interface

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.

Graphical user interface, table, Excel

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)

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”.

Graphical user interface, application, table

Description automatically generated

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

A picture containing text

Description automatically generated

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

Table

Description automatically generated

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

A picture containing graphical user interface

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]

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]

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.
2. 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.
3. 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.
4. 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.
5. 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"

}

]

}

Graphical user interface

Description automatically generated with medium confidence

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.

A picture containing graphical user interface

Description automatically generated

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

A picture containing text

Description automatically generated

1. 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.
2. Write a short essay talking about your understanding of transactions, locks and isolation levels.

Transaction is a group of commands which change data stored in database and a transaction is treated as one single unit, transaction basically ensures that either all the commands are executed successfully or none of the commands executed. If one command fails then all the commands will fail, any data that was modified in the database will be rolled back, if all the commands runs successfully then the transaction will be committed. Locks are used to prevent concurrent use of resources by different transactions, for example dead lock occurs when several processes want to access the resources that are lock by the other process. When working with database transactions, we need to choose an isolation level for the application. There are four standard Isolation levels, first level is read committed, transaction in this level can see data written by other uncommitted transactions; the second level is read committed that the transactions can only see data has been committed by other transactions. The third level is repeatable read which ensures that the same select query will always return the same result. The fourth level is serializable level which is the highest level of isolation in the isolation level, in this level, concurrent transactions are guaranteed exists at least one way to order these concurrent transactions.

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.

Being able to improve and tunning SQL server performance is one of the professional skills that data developers should familiarize themselves with. When it comes to performance tuning in SQL Server, it was difficult to locate the query slowness by looking or manually modifying the big query that contains multiple sub-queries. I would use a tool that Azure Data Studio built in is called Explain, is equivalent to SSMS Actual Execution Plan. It is a tool to visualize and analysis the query pipeline and performance usage. I would start with the sub-query with the biggest performance usage. If I am using SSMS instead of Azure Data Studio, there is an additional tool SSMS provide is called Tuning Advisor, it is a tool to analysis a query with a short amount of time then generate a report with recommendations. Another tool that SSMS provided is Extended Events, it is mainly helpful to monitor and troubleshoot details about the inner operations of the SQL system. It is a better use tool to determine lock on process such as finding the object with the most locks taken on them or determine queries are holding locks.

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.
2. 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.

~~Diagram

Description automatically generated~~

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.