Homework #6

**Question #1:**

List the names of all customers located in the state of Nevada. Sort the output by customer last name. The first and last name of the customer should be concatenated.

SELECT concat(FirstName, ' ', LastName ) as 'CustomerName',

Phone as 'CustomerPhone',

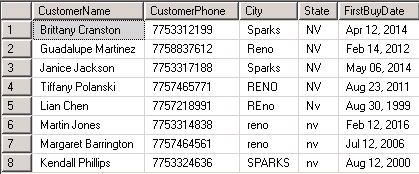
City,

State,

CONVERT(VARCHAR,FirstBuyDate, 107) 'FirstBuyDate'

FROM tblCustomer

WHERE State = 'NV';



**Question #2:**

Change the query to format the customer's phone number and first buy date. Change the structure of the customer name to make sure that there is a comma directly after the last name (no blank spaces between the last name and the comma) and that the first initial of the first name is separated by only one space from the comma. Make sure that there is a period directly after the initial. Change the format of the city and state so that they display in all upper case.

SELECT concat(LastName, ', ', left(FirstName, 1), '.') as 'CustomerName',

'(' + substring(Phone, 1, 3) + ') ' +

substring(Phone, 4, 3) + '-' +

substring(Phone, 7, 4) as 'Phone Number',

upper(City) as 'City',

upper(State) as 'State',

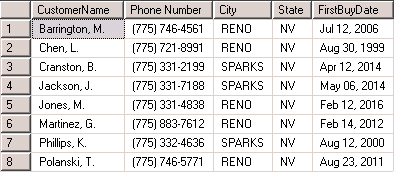
convert(varchar(12), FirstBuyDate, 107) as 'FirstBuyDate'

FROM tblCustomer

WHERE State = 'NV'

ORDER BY CustomerName;

**-TABLE ON NEXT PAGE-**



**Question #3: (Needs Table)**

List the orders in the order table that do NOT have a discount code (the discount code is a NULL value). Sort the output by orderdate.

SELECT convert(varchar(10), OrderDate, 101) as 'Date of Order',

OrderID as 'Order Number',

CustomerID as 'Customer Number',

CreditCode as 'Credit Code'

FROM tblOrder

WHERE DiscountCode IS NULL

ORDER BY OrderDate;

**Question #4:**

List order information placed for itemID A34665. Include the orderID, itemID, quantity ordered, price paid, and calculate the extended price (price \* quantity). Sort the output by orderID.

SELECT OrderID as 'OrderNumber',

ItemID as 'ItemNumber',

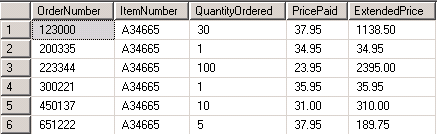
Quantity as 'QuantityOrdered',

Price as 'PricePaid',

Quantity \* Price as 'ExtendedPrice'

FROM tblOrderLine

WHERE itemId = 'A34665';



**Question #5:**

List the order information for all items in the OrderLine table that have an extended price over $800. Sort the rows by itemID within orderID.

SELECT OrderID as 'OrderNumber',

ItemID as 'ItemNumber',

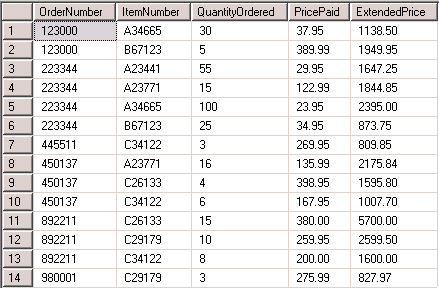
Quantity as 'QuantityOrdered',

Price as 'PricePaid',

Quantity \* Price as 'ExtendedPrice'

FROM tblOrderLine

WHERE Quantity \* Price > 800;



**Question #6:**

Modify query #5 to include a column with a message about the extended price.

SELECT OrderID as OrderNumber,

ItemID as ItemNumber,

Quantity as QuantityOrdered,

Price as PricePaid,

(Quantity \* Price) as ExtendedPrice,

CASE

WHEN 1000 > (Quantity \* Price)

THEN NULL

WHEN 1500 > (Quantity \* Price)

THEN 'Medium Order'

WHEN 2000 > (Quantity \* Price)

THEN 'Large Order - Monitor'

WHEN 5000 > (Quantity \* Price)

THEN 'Very Large Order – Watch Dates'

WHEN 5000 < (Quantity \* Price)

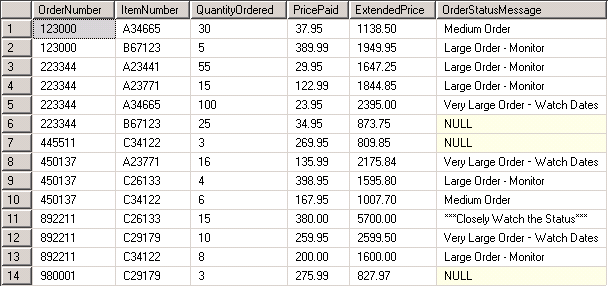
THEN '\*\*\*Closely Watch the Status\*\*\*'

END OrderStatusMessage

FROM tblOrderLine

WHERE Quantity \* Price > 800;

**-TABLE ON NEXT PAGE-**



**Question #7:**

List all the shipments in the ShipLine table that were shipped in January of the current year. For full credit on the answer, the query must be flexible enough to compare the year of the DateShipped to the year of the current date. Thus, you must use the GETDATE() function to determine the correct year for comparison. Sort the result table in order by ItemID within OrderID.

SELECT OrderID,

ItemID,

convert(varchar(10), DateShipped, 101) as DateShipped,

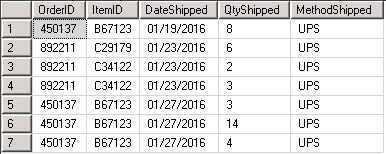
QtyShipped,

upper(MethodShiped) 'MethodShipped'

FROM tblShipLine

WHERE DateShipped BETWEEN ( '01/01/' + cast(year(getdate()) as varchar(4)))

AND ( '01/31/' + cast(year(getdate()) as varchar(4)));



**Question #8:**

What is the earliest FirstBuyDate for a customer in the CUSTOMER table? You do not have to display any other information – just the value of the earliest (or you might consider the earliest to be the minimum) FirstBuyDate – for the answer to this query.

SELECT TOP 1 convert(varchar(12), FirstBuyDate, 107) as 'Earliest First Buy Date'

FROM tblCustomer

ORDER BY FirstBuyDate;



**Question #9:**

What is the average selling price for itemID A34665? Use the OrderLine table for the table in this query. Round the result.

SELECT ROUND(AVG(Price), 2) as 'Average Selling Price'

FROM tblOrderLine

WHERE itemID = 'A34665';



**Question #10:**

Create a grouped report that summarizes data and provides information about each item that is on an order in the OrderLine table. The report should include the itemID, the number of rows in the OrderLine table for that item, total quantity on order for that item, the minimum price that the item sold for, the maximum price that the item sold for, and the average price that the item sold for. The report should be ordered by itemID.

SELECT itemID,

COUNT (itemID) NumberOfRows,

SUM (Quantity) QuantitySold,

MIN (Price) MinPrice,

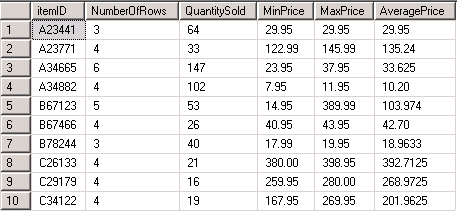
MAX (Price) MaxPrice,

AVG (Price) AveragePrice

FROM tblOrderLine

GROUP BY itemID;

**-TABLE ON NEXT PAGE-**



**Question #11:**

Modify query #10 so that it only displays the rows where the MaximumPrice is greater than 50% more than the MinimumPrice. Add a column to the output that displays the percentage difference between the minimum and maximum price.

SELECT \*,

CONVERT(VARCHAR(50),(MaxPrice-MinPrice) / ((MaxPrice+MinPrice)/2)\*100)+'%' Diff

FROM (

SELECT itemID,

COUNT (itemID) NumberOfRows,

SUM (Quantity) QuantitySold,

MIN (Price) as MinPrice,

MAX (Price) as MaxPrice,

AVG (Price) AveragePrice

FROM tblOrderLine

GROUP BY itemID) sub

WHERE MinPrice\*1.5 < MaxPrice



**Question #12: (Needs Table)**

Assume just for this query that all orders in the Order table are waiting to be shipped. That isn’t true for this database, but assume it is anyway for this question. Calculate the number of days each order is overdue and display those orders that are 40 days overdue from the day you run your query. I’m running this query on 2/26/2016, so my output is based on that run date. Use only the Order table for this query.

SELECT OrderID as 'Order Number',

CustomerID as 'Customer Number',

OrderDate as 'Date Ordered',

DATEADD(day,40,OrderDate) as '40 Days After Date Ordered',

DATEDIFF(day, OrderDate, GETDATE()) as 'Number Of Days Difference',

GETDATE () as 'Current Date and Time'

FROM tblOrder

WHERE DATEDIFF(day, OrderDate, GETDATE()) >= 40

GROUP BY OrderID,

CustomerID,

OrderDate

**Question #13:**

It is possible in this database to have partial shipments. The goal of this query is to see only those OrderID and ItemID combinations that were shipped in partial shipments. Count the number of shipments, as represented by rows in the ShipLine table, sum the quantity shipped in the ShipLine table by orderID and itemID, displaying only those combinations of orderID and itemID that have more than one row in the ShipLine table.

SELECT OrderID,

ItemID,

COUNT(QtyShipped) NumberOfShipments,

SUM(QtyShipped) TotalShipped

FROM TblShipLine

GROUP BY OrderID,

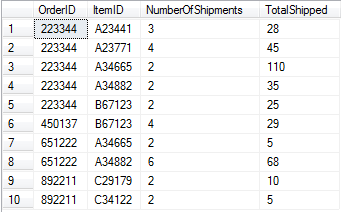
ItemID

HAVING COUNT(QtyShipped) > 1

ORDER BY OrderID,

ItemID

**-TABLE ON NEXT PAGE-**



**Question #14:**

Which customers do not have any orders in the Order table? I recommend using a non-correlated sub-query for this query.

SELECT CustomerID,

Lastname +', ' + SUBSTRING(FirstName, 1,1) + '.' EmployeeName,

'(' + SUBSTRING(Phone, 1,3) + ') ' + SUBSTRING(Phone, 4,3) + '-' + SUBSTRING(Phone, 7,4) PhoneNumber,

City,

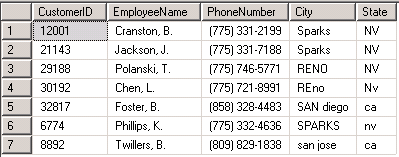
State

FROM TblCustomer

WHERE CustomerID NOT IN

(SELECT CustomerID

FROM TblOrder)



**Question #15:**

Which customer is the one who has the most recent FirstBuyDate? I recommend using a non-correlated sub-query for this query.

SELECT CustomerID,

Lastname +', ' + SUBSTRING(FirstName, 1,1) + '.' 'Employee Name',

'(' + SUBSTRING(Phone, 1,3) + ') ' + SUBSTRING(Phone, 4,3) + '-' + SUBSTRING(Phone, 7,4) 'Phone Number',

City,

State,

CONVERT(VARCHAR,FirstBuyDate, 107) 'First Buy Date'

FROM TblCustomer

WHERE FirstBuyDate in

(SELECT TOP 1 FirstBuyDate

FROM TblCustomer

ORDER BY FirstBuyDate DESC)

