IS475/675 HW#6

Each of the questions in this homework assignment asks you to write a SELECT statement to satisfy the request. There are 15 questions. Complete result tables are provided for all of the questions to help you check your work.

Deliverable

After reading this homework assignment, estimate how long in hours it will take you to complete it. Write your estimate on the time sheet provided as the last sheet of this assignment.

For each question turn in the SQL code and then the output from that code for that question directly under the code. **I want to first see the SQL code, and then see the output generated from that code directly under the code.** For example, let's say that one of the questions was to find the last name of all customers who live in the city of Sparks. Let's assume that this was question number 3 (it isn't - this is just an example to show what I want turned in for the assignment). The deliverable would look like this:

#3. Code

```
SELECT      lastname     "Customer Last Name",
            UPPER(City)  "Customer City"
FROM        TblCustomer
WHERE city = 'sparks';
```

#3. Output

	Customer Last Name	Customer City
1	Phillips	SPARKS
2	Cranston	SPARKS
3	Jackson	SPARKS

Guidelines for SQL code:

- Start SQL reserved words in the first column of the line.
- Put SQL reserved words in all uppercase (or an easily identifiable combination of upper and lower case).
- Put attribute and table names in all lower case, or a combination of upper and lower case.
- Place no more than one SQL command (SELECT, FROM, etc.) per line. Feel free to combine functions such as SUBSTRING, POSITION, AVG on a single line.
- Leave spaces after the SQL reserved word and the directives of the command. Use the TAB key to improve readability.

The SQL code must be written in SQL Server Management Studio. The result table should be snipped and copied from the result table generated by SQL Server Management Studio. You may have to use landscape output in MS Word for some of the wider reports as I have in this assignment. Please make the result tables and SQL code large enough to be readable. The samples provided on this document are readable – please do not make yours smaller. Use landscape formatting in Word if necessary to keep it readable.

The code and output for each question should be identifiable - please write down which question is satisfied with the code and output. Please turn in printed output as the main deliverable for grading. Also turn in a time sheet with the original time estimate in hours, and the actual time spent to complete the assignment.

Make sure your query will work correctly with any input data. Our test dataset, like all test datasets, is not comprehensive. Try and think of situations where your query could produce an incorrect result table and make sure your SQL code will handle those situations.

Make queries that continue to work when a year changes. For example, if a query asks to see all purchase orders placed in January of the current year, don't limit it to the purchase orders placed in January of 2016. Make your SQL code work for **any** current year so that the query will work in the future without modification.

Here are the query questions:

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 so that the format of the result table is the same as that shown below:

	CustomerName	CustomerPhone	City	State	FirstBuyDate
1	Margaret Barrington	7757464561	reno	nv	2006-07-12 00:00:00.000
2	Lian Chen	7757218991	REno	Nv	1999-08-30 00:00:00.000
3	Brittany Cranston	7753312199	Sparks	NV	2014-04-12 00:00:00.000
4	Janice Jackson	7753317188	Sparks	NV	2014-05-06 00:00:00.000
5	Martin Jones	7753314838	reno	nv	2016-02-12 00:00:00.000
6	Guadalupe Martinez	7758837612	Reno	NV	2012-02-14 00:00:00.000
7	Kendall Phillips	7753324636	SPARKS	nv	2000-08-12 00:00:00.000
8	Tiffany Polanski	7757465771	RENO	NV	2011-08-23 00:00:00.000

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. Here is the result table:

	Customer Name	Phone Number	City	State	FirstBuyDate
1	Barrington, M.	(775) 746-4561	RENO	NV	Jul 12, 2006
2	Chen, L.	(775) 721-8991	RENO	NV	Aug 30, 1999
3	Cranston, B.	(775) 331-2199	SPARKS	NV	Apr 12, 2014
4	Jackson, J.	(775) 331-7188	SPARKS	NV	May 06, 2014
5	Jones, M.	(775) 331-4838	RENO	NV	Feb 12, 2016
6	Martinez, G.	(775) 883-7612	RENO	NV	Feb 14, 2012
7	Phillips, K.	(775) 332-4636	SPARKS	NV	Aug 12, 2000
8	Polanski, T.	(775) 746-5771	RENO	NV	Aug 23, 2011

3. 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. Here is the complete result table:

	Date of Order	Order Number	Customer Number	Credit Code
1	12/28/2015	675990	00625	111
2	12/28/2015	892211	00625	111
3	12/29/2015	450137	07831	444
4	01/26/2016	567123	07831	444
5	01/26/2016	200335	07831	111
6	01/29/2016	651222	12006	111
7	02/09/2016	223344	21142	231
8	02/19/2016	983983	32018	111
9	02/19/2016	671100	32018	111

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. Here is the result table:

	OrderNumber	ItemNumber	QuantityOrdered	PricePaid	ExtendedPrice
1	123000	A34665	30	37.95	1138.50
2	200335	A34665	1	34.95	34.95
3	223344	A34665	100	23.95	2395.00
4	300221	A34665	1	35.95	35.95
5	450137	A34665	10	31.00	310.00
6	651222	A34665	5	37.95	189.75

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. Here is the result table:

	OrderNumber	ItemNumber	QuantityOrdered	PricePaid	ExtendedPrice
1	123000	A34665	30	37.95	1138.50
2	123000	B67123	5	389.99	1949.95
3	223344	A23441	55	29.95	1647.25
4	223344	A23771	15	122.99	1844.85
5	223344	A34665	100	23.95	2395.00
6	223344	B67123	25	34.95	873.75
7	445511	C34122	3	269.95	809.85
8	450137	A23771	16	135.99	2175.84
9	450137	C26133	4	398.95	1595.80
10	450137	C34122	6	167.95	1007.70
11	892211	C26133	15	380.00	5700.00
12	892211	C29179	10	259.95	2599.50
13	892211	C34122	8	200.00	1600.00
14	980001	C29179	3	275.99	827.97

6. Modify query #5 to include a column with a message about the extended price. The table below provides the messages appropriate for an extended price:

Value of Extended Price	**Contents of Order Message Column**
< 1000	No message (null)
>= 1000 and < 1500	Medium Order
>= 1500 and < 2000	Large Order - Monitor
>= 2000 and < 5000	Very Large Order – Watch Dates
>= 5000	***Closely Watch the Status***

The result table is provided at the top of the next page.

	OrderNumber	ItemNumber	QuantityOrdered	PricePaid	ExtendedPrice	OrderStatusMessage
1	123000	A34665	30	37.95	1138.50	Medium Order
2	123000	B67123	5	389.99	1949.95	Large Order - Monitor
3	223344	A23441	55	29.95	1647.25	Large Order - Monitor
4	223344	A23771	15	122.99	1844.85	Large Order - Monitor
5	223344	A34665	100	23.95	2395.00	Very Large Order - Watch Dates
6	223344	B67123	25	34.95	873.75	NULL
7	445511	C34122	3	269.95	809.85	NULL
8	450137	A23771	16	135.99	2175.84	Very Large Order - Watch Dates
9	450137	C26133	4	398.95	1595.80	Large Order - Monitor
10	450137	C34122	6	167.95	1007.70	Medium Order
11	892211	C26133	15	380.00	5700.00	***Closely Watch the Status***
12	892211	C29179	10	259.95	2599.50	Very Large Order - Watch Dates
13	892211	C34122	8	200.00	1600.00	Large Order - Monitor
14	980001	C29179	3	275.99	827.97	NULL

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. Result table:

	OrderID	ItemID	DateShipped	QtyShipped	MethodShipped
1	450137	B67123	01/19/2016	8	UPS
2	450137	B67123	01/27/2016	3	UPS
3	450137	B67123	01/27/2016	14	UPS
4	450137	B67123	01/27/2016	4	UPS
5	892211	C29179	01/23/2016	6	UPS
6	892211	C34122	01/23/2016	2	UPS
7	892211	C34122	01/23/2016	3	UPS

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. Result table:

	Earliest First Buy Date
1	Aug 15, 1999

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

	Average Selling Price
1	33.63

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. Result table:

	itemid	NumberOfRows	QuantitySold	MinimumPrice	MaximumPrice	AveragePrice
1	A23441	3	64	29.95	29.95	29.95
2	A23771	4	33	122.99	145.99	135.24
3	A34665	6	147	23.95	37.95	33.63
4	A34882	4	102	7.95	11.95	10.20
5	B67123	5	53	14.95	389.99	103.97
6	B67466	4	26	40.95	43.95	42.70
7	B78244	3	40	17.99	19.95	18.96
8	C26133	4	21	380.00	398.95	392.71
9	C29179	4	16	259.95	280.00	268.97
10	C34122	4	19	167.95	269.95	201.96

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. Here is the result table:

	itemid	NumberOfRows	QuantitySold	MinimumPrice	MaximumPrice	AveragePrice	Diff
1	A34665	6	147	23.95	37.95	33.63	58.45%
2	A34882	4	102	7.95	11.95	10.20	50.31%
3	B67123	5	53	14.95	389.99	103.97	2508.62%
4	C34122	4	19	167.95	269.95	201.96	60.73%

12. 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. Here is the result table with a current date of 2/26/2016, but your output will be slightly different because you will be running it on a different day. Make sure that your "current date and time" column is the actual current date when you run your query.

	Order Number	Customer Number	Date Ordered	40 Days After Date Ordered	Number of Days Difference	Current Date and Time
1	450137	07831	Dec 29, 2015	Feb 07, 2016	59	2016-02-26 21:43:10.970
2	675990	00625	Dec 28, 2015	Feb 06, 2016	60	2016-02-26 21:43:10.970
3	892211	00625	Dec 28, 2015	Feb 06, 2016	60	2016-02-26 21:43:10.970

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. The result table is on the top of the next page.

	OrderID	ItemID	NumberOfShipments	TotalShipped
1	223344	A23441	3	28
2	223344	A23771	4	45
3	223344	A34665	2	110
4	223344	A34882	2	35
5	223344	B67123	2	25
6	450137	B67123	4	29
7	651222	A34665	2	5
8	651222	A34882	6	68
9	892211	C29179	2	10
10	892211	C34122	2	5

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

	CustomerID	CustomerName	PhoneNumber	City	State
1	06774	Phillips, K.	(775) 332-4636	SPARKS	NV
2	08892	Twillers, B.	(809) 829-1838	SAN JOSE	CA
3	12001	Cranston, B.	(775) 331-2199	SPARKS	NV
4	21143	Jackson, J.	(775) 331-7188	SPARKS	NV
5	29188	Polanski, T.	(775) 746-5771	RENO	NV
6	30192	Chen, L.	(775) 721-8991	RENO	NV
7	32817	Foster, B.	(858) 328-4483	SAN DIEGO	CA

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

	CustomerID	Customer Name	Phone Number	City	State	First Purchase Date
1	32018	Jones, M.	(775) 331-4838	RENO	NV	Feb 12, 2016

IS 475/675 HW#6 Time Sheet

Name: **Working in a Group? Yes/No**

Time Estimate:

How many hours do you think it will take you to finish this assignment?

Date	Hours		Date	Hours

Total Time Spent =