IS475/675 HW#7: Accessing Data from one table at a time with SQL

Each of the questions in this homework assignment requires you to create one SELECT statement to satisfy the request. There are 15 questions for this assignment. Questions start on pg. 3 of this document.

Deliverable

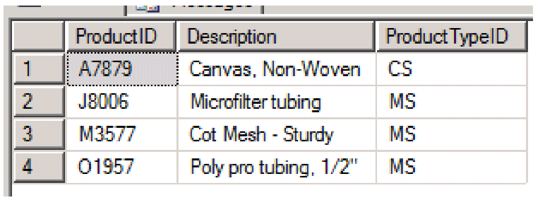
Create a single Word document (or a .pdf) with the answers to the questions and upload it to WebCampus. There should be one answer per question, but that should be only one document for the entire deliverable. Each answer should produce one SELECT statement and one result table per question. Please use the format for the answers that is discussed below.

For each question turn in the SQL code and then the result table produced from that code for that question directly under the code. **I want to first see the SQL code, and then see the result table generated from that code directly under the code.**

For example, let's say that one of the questions is to find the Products that have a UOM equal to the value "feet". Let's assume that this is question #2 (it isn't - this is just an example to show what I want turned in for the assignment). The deliverable would look like this:

Question #2.

```
SELECT        ProductID,
              Description,
              ProductTypeID
FROM          tblproduct
WHERE         uom = 'feet'
```



	ProductID	Description	ProductTypeID
1	A7879	Canvas, Non-Woven	CS
2	J8006	Microfilter tubing	MS
3	M3577	Cot Mesh - Sturdy	MS
4	01957	Poly pro tubing, 1/2"	MS

The SQL code should be a copy of the code you write and execute to produce your result table. The SQL code should not be snipped as a picture – just copy and paste the SQL code from Management Studio to a MS Word document.

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, ISNULL, and 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 **result table** should be snipped as a picture and copied from the result table generated by SQL Server Management Studio. Do NOT turn in the result table provided in this document. Turn in the result table that is generated by your SQL code.

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.

Your query should produce accurate output. I have provided the result tables for the queries so that you can more easily test your work. It is possible that your output may look a little different from mine. I have leading zeros in some of my data, and you may not. That is OK, but keep in mind that the output may sort differently in the result table.

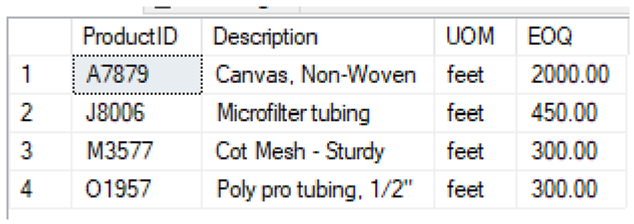
Make sure your query will work correctly with any input data. Our test dataset, like all test datasets, is not comprehensive. Try and think of situations where your query could produce an incorrect result table and make sure your SQL code will handle those situations.

Make queries that continue to work when a year changes. For example, if a query asks to see all purchase orders placed in September of the current year, don't limit it to the purchase orders placed in September of 2020 – instead make your SQL code work for **any** current year so that the query will work in the future without modification.

Here are the questions:

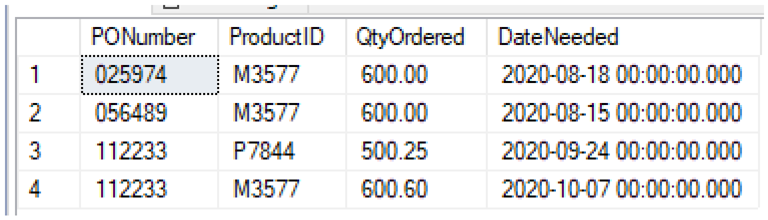
Full result tables are provided for all questions to help you become familiar with SQL.

1. List the data shown below from the Product table. These are the products that have a UOM (unit of measure) equal to ‘feet’. Sort the result table by ProductID.



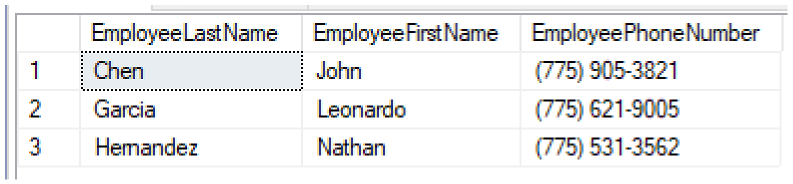
	ProductID	Description	UOM	EOQ
1	A7879	Canvas, Non-Woven	feet	2000.00
2	J8006	Microfilter tubing	feet	450.00
3	M3577	Cot Mesh - Sturdy	feet	300.00
4	O1957	Poly pro tubing, 1/2"	feet	300.00

2. List the data shown below from the PurchaseOrderLine table. These are the PurchaseOrderLines that have a dateneeded that is prior to the current date and a qtyordered that is greater than 500. I ran this query on 10/14/2020, so your output may have a few more rows depending on when you run your query. Make sure that you use the current date in the GETDATE() function in your WHERE statement. Sort the result table by PONumber.



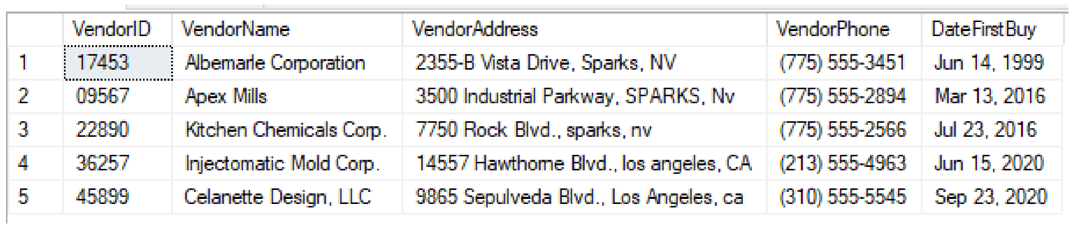
	PONumber	ProductID	QtyOrdered	DateNeeded
1	025974	M3577	600.00	2020-08-18 00:00:00.000
2	056489	M3577	600.00	2020-08-15 00:00:00.000
3	112233	P7844	500.25	2020-09-24 00:00:00.000
4	112233	M3577	600.60	2020-10-07 00:00:00.000

3. List the data shown below about employees in the Employee table who don’t have an email address (the email is null). Be sure to format the phone number as shown.



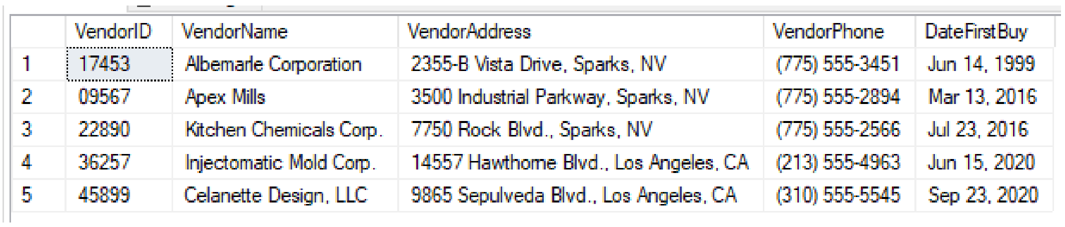
	EmployeeLastName	EmployeeFirstName	EmployeePhoneNumber
1	Chen	John	(775) 905-3821
2	Garcia	Leonardo	(775) 621-9005
3	Hernandez	Nathan	(775) 531-3562

4. List information about vendors in the Vendor table who are located in either the state of California or the state of Nevada. Concatenate the Address1, City, and State columns into a single column. Format the phone number using standard U.S. formatting. Format the FirstBuyDate as shown in the result table. Sort the result table by FirstBuyDate. The column in the result table below called “DateFirstBuy” is created from the FirstBuyDate in the Vendor table. Here is the result table:



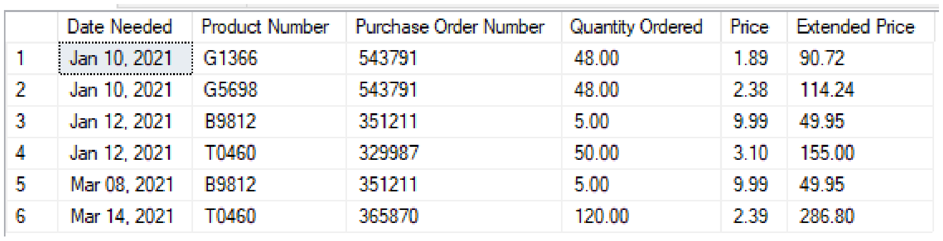
	VendorID	VendorName	VendorAddress	VendorPhone	DateFirstBuy
1	17453	Albemarle Corporation	2355-B Vista Drive, Sparks, NV	(775) 555-3451	Jun 14, 1999
2	09567	Apex Mills	3500 Industrial Parkway, SPARKS, Nv	(775) 555-2894	Mar 13, 2016
3	22890	Kitchen Chemicals Corp.	7750 Rock Blvd., sparks, nv	(775) 555-2566	Jul 23, 2016
4	36257	Injectomatic Mold Corp.	14557 Hawthorne Blvd., los angeles, CA	(213) 555-4963	Jun 15, 2020
5	45899	Celanette Design, LLC	9865 Sepulveda Blvd., Los Angeles, ca	(310) 555-5545	Sep 23, 2020

4a. Extra Credit + 3 points for correcting the capitalization of the city – as long as you input the capitalization "incorrectly" in your table as provided in HW#6 as depicted in the "standard" result table above. Show both result tables and both queries to earn the extra credit. Here is the result table for the extra credit for question #4a:



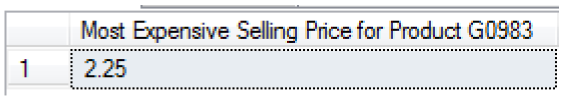
	VendorID	VendorName	VendorAddress	VendorPhone	DateFirstBuy
1	17453	Albemarle Corporation	2355-B Vista Drive, Sparks, NV	(775) 555-3451	Jun 14, 1999
2	09567	Apex Mills	3500 Industrial Parkway, Sparks, NV	(775) 555-2894	Mar 13, 2016
3	22890	Kitchen Chemicals Corp.	7750 Rock Blvd., Sparks, NV	(775) 555-2566	Jul 23, 2016
4	36257	Injectomatic Mold Corp.	14557 Hawthorne Blvd., Los Angeles, CA	(213) 555-4963	Jun 15, 2020
5	45899	Celanette Design, LLC	9865 Sepulveda Blvd., Los Angeles, CA	(310) 555-5545	Sep 23, 2020

5. List the DateNeeded, ProductID, PONumber, QtyOrdered and Price for all purchase order lines that have a DateNeeded that is not in the current year. Calculate the extended price (QuantityOrdered multiplied by Price). Sort the output by DateNeeded. For full credit on the answer, the query must compare the year of the DateNeeded to the year of the current date. Thus, you must use the GETDATE() function to determine the correct year for comparison. Use the format below for the result table.



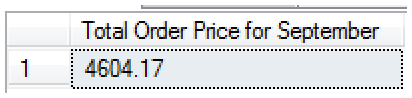
	Date Needed	Product Number	Purchase Order Number	Quantity Ordered	Price	Extended Price
1	Jan 10, 2021	G1366	543791	48.00	1.89	90.72
2	Jan 10, 2021	G5698	543791	48.00	2.38	114.24
3	Jan 12, 2021	B9812	351211	5.00	9.99	49.95
4	Jan 12, 2021	T0460	329987	50.00	3.10	155.00
5	Mar 08, 2021	B9812	351211	5.00	9.99	49.95
6	Mar 14, 2021	T0460	365870	120.00	2.39	286.80

6. What is the highest selling price for productID ‘G0983’? Use the PurchaseOrderLine table for this query. Result table:



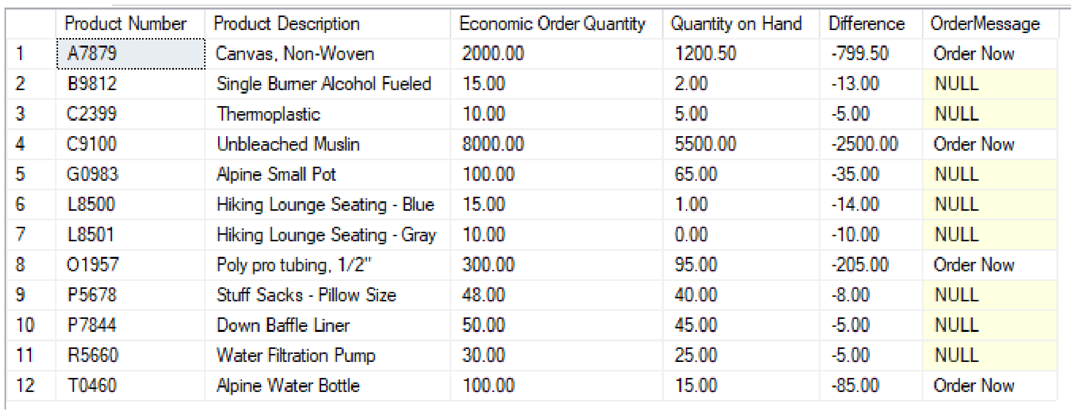
	Most Expensive Selling Price for Product G0983
1	2.25

7. What is the sum of the qtyordered * price for all purchase order lines in the PurchaseOrderLine table that have a DateNeeded in September of the current year? Make sure that your query looks for the current year using the GETDATE() function. The test dataset doesn’t contain data for September in another year, but make sure the query works with both month and year. Result table:



	Total Order Price for September
1	4604.17

8. Create a result table from the Product table to list which products have a quantity on hand (QOH) less than the economic order quantity (EOQ). For products where the QOH is more than 50 units less than the EOQ, display a message that says “Order Now” – for those greater than or equal to 50, just leave the message NULL. (Hint: use the CASE statement to generate the order message). Sort the output by Product Number. The result table should look like the one provided on the top of the next page:

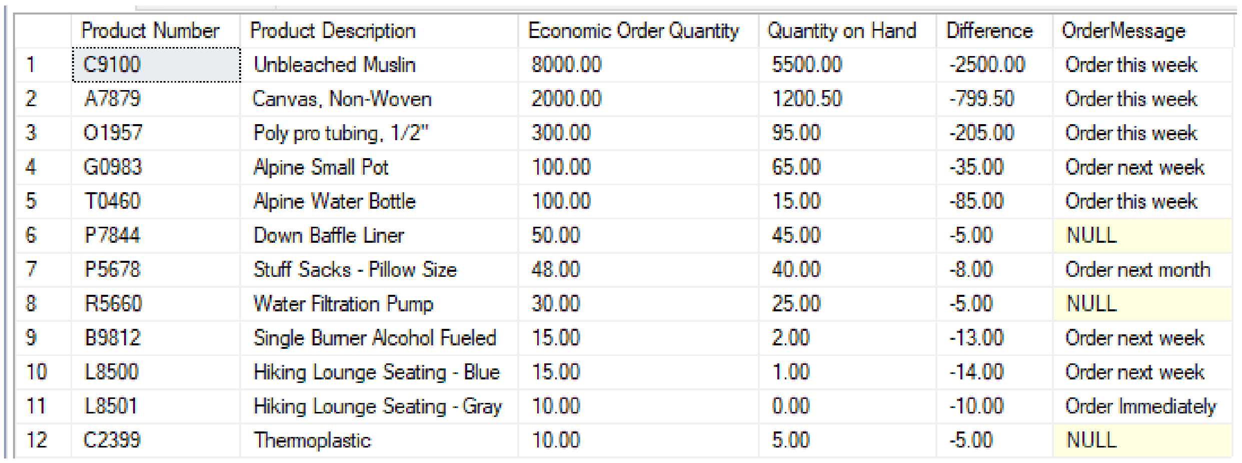


	Product Number	Product Description	Economic Order Quantity	Quantity on Hand	Difference	OrderMessage
1	A7879	Canvas, Non-Woven	2000.00	1200.50	-799.50	Order Now
2	B9812	Single Burner Alcohol Fueled	15.00	2.00	-13.00	NULL
3	C2399	Thermoplastic	10.00	5.00	-5.00	NULL
4	C9100	Unbleached Muslin	8000.00	5500.00	-2500.00	Order Now
5	G0983	Alpine Small Pot	100.00	65.00	-35.00	NULL
6	L8500	Hiking Lounge Seating - Blue	15.00	1.00	-14.00	NULL
7	L8501	Hiking Lounge Seating - Gray	10.00	0.00	-10.00	NULL
8	O1957	Poly pro tubing, 1/2"	300.00	95.00	-205.00	Order Now
9	P5678	Stuff Sacks - Pillow Size	48.00	40.00	-8.00	NULL
10	P7844	Down Baffle Liner	50.00	45.00	-5.00	NULL
11	R5660	Water Filtration Pump	30.00	25.00	-5.00	NULL
12	T0460	Alpine Water Bottle	100.00	15.00	-85.00	Order Now

9. Modify the query written for #8 to have additional messages possible to display in the Order Message column. Remember that we are still only looking at the rows where the QOH is less than the EOQ. Use the table below to see what message should be displayed based on the difference between QOH and EOQ.

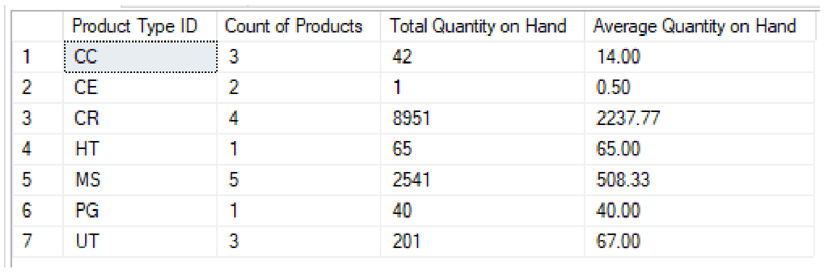
Difference between QOH and EOQ	**Contents of Order Message Column**
-1 to -5	No message (null)
-6 to -10	Order next month
-11 to -35	Order next week
Lower than -35	Order this week
If the QOH is equal to zero	Order Immediately

Sort the result table by EOQ in descending order so that it looks like the table shown below



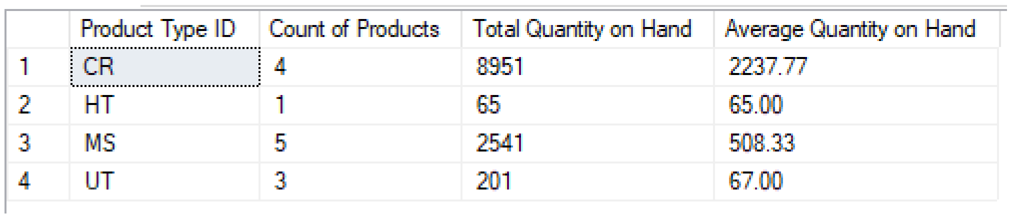
	Product Number	Product Description	Economic Order Quantity	Quantity on Hand	Difference	OrderMessage
1	C9100	Unbleached Muslin	8000.00	5500.00	-2500.00	Order this week
2	A7879	Canvas, Non-Woven	2000.00	1200.50	-799.50	Order this week
3	O1957	Poly pro tubing, 1/2"	300.00	95.00	-205.00	Order this week
4	G0983	Alpine Small Pot	100.00	65.00	-35.00	Order next week
5	T0460	Alpine Water Bottle	100.00	15.00	-85.00	Order this week
6	P7844	Down Baffle Liner	50.00	45.00	-5.00	NULL
7	P5678	Stuff Sacks - Pillow Size	48.00	40.00	-8.00	Order next month
8	R5660	Water Filtration Pump	30.00	25.00	-5.00	NULL
9	B9812	Single Burner Alcohol Fueled	15.00	2.00	-13.00	Order next week
10	L8500	Hiking Lounge Seating - Blue	15.00	1.00	-14.00	Order next week
11	L8501	Hiking Lounge Seating - Gray	10.00	0.00	-10.00	Order Immediately
12	C2399	Thermoplastic	10.00	5.00	-5.00	NULL

10. Create a **grouped** report summarizing data about product types in the PRODUCT table. The report should include ProductTypeID, the count of items in the PRODUCT table of that product type, the total quantity on hand for that product type, and the average quantity on hand for that product type. Sort the output by ProducttTypeID. Result table:



	Product Type ID	Count of Products	Total Quantity on Hand	Average Quantity on Hand
1	CC	3	42	14.00
2	CE	2	1	0.50
3	CR	4	8951	2237.77
4	HT	1	65	65.00
5	MS	5	2541	508.33
6	PG	1	40	40.00
7	UT	3	201	67.00

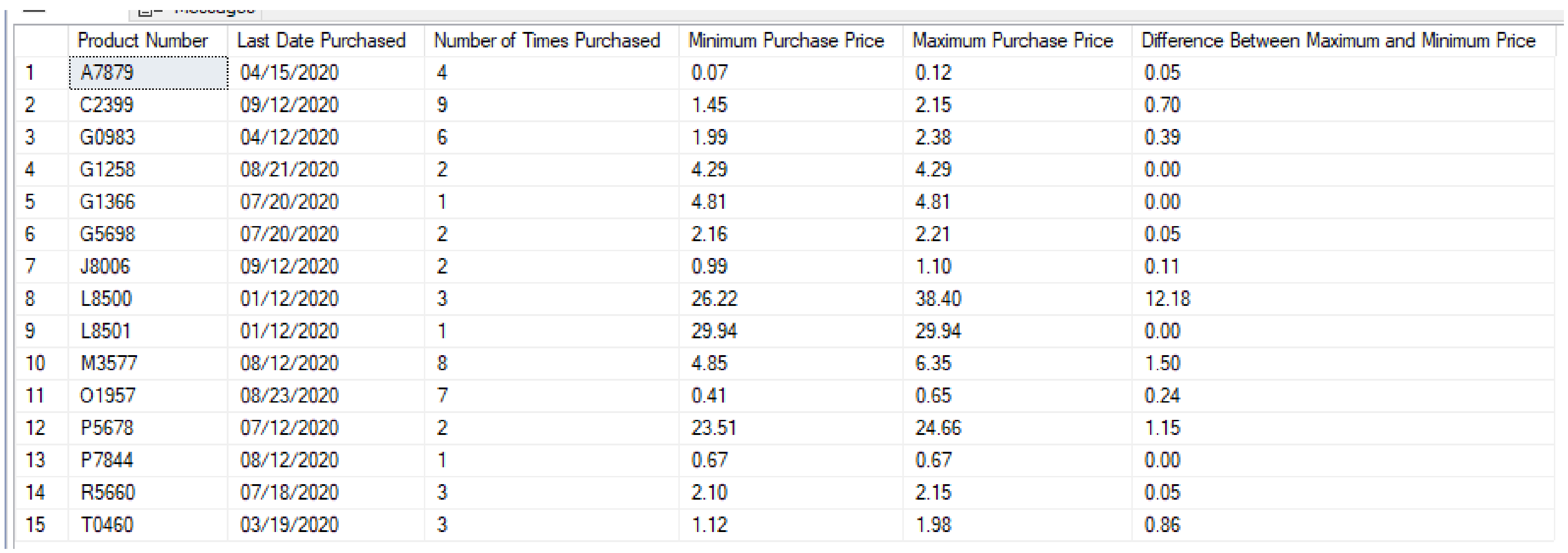
11. Modify the query created for question #10 to display only those rows where the average quantity on hand is greater than 50. Result table:



	Product Type ID	Count of Products	Total Quantity on Hand	Average Quantity on Hand
1	CR	4	8951	2237.77
2	HT	1	65	65.00
3	MS	5	2541	508.33
4	UT	3	201	67.00

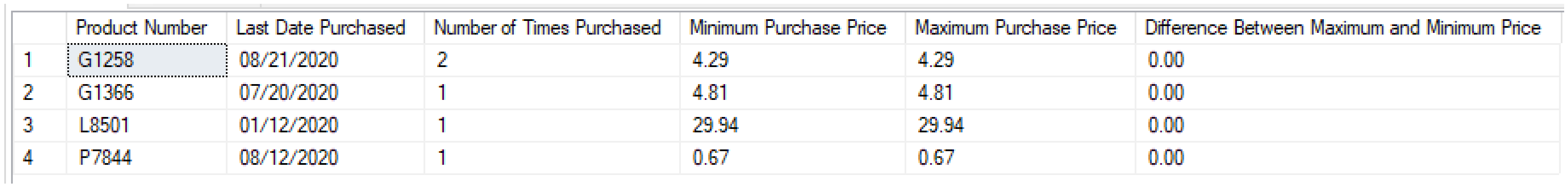
Keep going – there are more questions on the next page!!

12. Create a grouped report summarizing data in the PurchaseHistory table by ProductNumber. The report should provide the last date (most recent) that a product was purchased, a count of the number of times a product was purchased, the minimum price and the maximum price as well as the difference between the two. There should be one row in the result table for each product in the PurchaseHistory table. Result table:



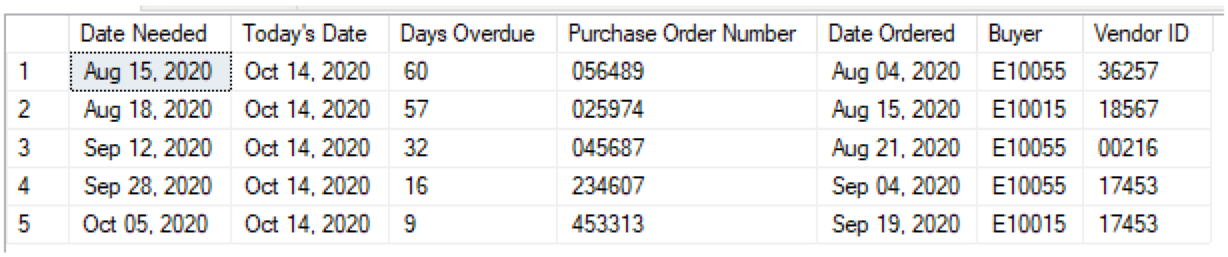
	Product Number	Last Date Purchased	Number of Times Purchased	Minimum Purchase Price	Maximum Purchase Price	Difference Between Maximum and Minimum Price
1	A7879	04/15/2020	4	0.07	0.12	0.05
2	C2399	09/12/2020	9	1.45	2.15	0.70
3	G0983	04/12/2020	6	1.99	2.38	0.39
4	G1258	08/21/2020	2	4.29	4.29	0.00
5	G1366	07/20/2020	1	4.81	4.81	0.00
6	G5698	07/20/2020	2	2.16	2.21	0.05
7	J8006	09/12/2020	2	0.99	1.10	0.11
8	L8500	01/12/2020	3	26.22	38.40	12.18
9	L8501	01/12/2020	1	29.94	29.94	0.00
10	M3577	08/12/2020	8	4.85	6.35	1.50
11	O1957	08/23/2020	7	0.41	0.65	0.24
12	P5678	07/12/2020	2	23.51	24.66	1.15
13	P7844	08/12/2020	1	0.67	0.67	0.00
14	R5660	07/18/2020	3	2.10	2.15	0.05
15	T0460	03/19/2020	3	1.12	1.98	0.86

13. Modify the query created for question #12 to display only those rows with zero difference between the minimum and maximum purchase price. Result table:



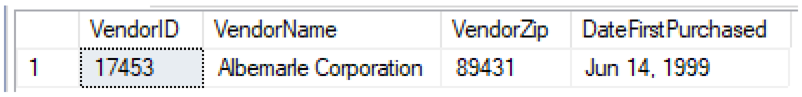
	Product Number	Last Date Purchased	Number of Times Purchased	Minimum Purchase Price	Maximum Purchase Price	Difference Between Maximum and Minimum Price
1	G1258	08/21/2020	2	4.29	4.29	0.00
2	G1366	07/20/2020	1	4.81	4.81	0.00
3	L8501	01/12/2020	1	29.94	29.94	0.00
4	P7844	08/12/2020	1	0.67	0.67	0.00

14. Which purchase orders are overdue by more than 7 days? Use the PODateNeeded attribute in the PurchaseOrder table to make this determination. Assume that none of the purchase orders are filled if they are in the PurchaseOrder table in your database. Use the current date (obtained through GETDATE()) to make the determination of whether a purchase order is overdue. Sort the output by PODateNeeded. The result table provided below was run with a current date (using the GETDATE() function) that was equal to 10/14/2020 (as you can see on the result table), so your current date will be different when you run your SQL program. Hint: The DATEDIFF function is an easy way to calculate the difference between dates.



	Date Needed	Today's Date	Days Overdue	Purchase Order Number	Date Ordered	Buyer	Vendor ID
1	Aug 15, 2020	Oct 14, 2020	60	056489	Aug 04, 2020	E10055	36257
2	Aug 18, 2020	Oct 14, 2020	57	025974	Aug 15, 2020	E10015	18567
3	Sep 12, 2020	Oct 14, 2020	32	045687	Aug 21, 2020	E10055	00216
4	Sep 28, 2020	Oct 14, 2020	16	234607	Sep 04, 2020	E10055	17453
5	Oct 05, 2020	Oct 14, 2020	9	453313	Sep 19, 2020	E10015	17453

15. Which vendor has the earliest FirstBuyDate in the vendor table? This question is answered either through the TOP 1 option, or through a non-correlated sub-query. Result table:



	VendorID	VendorName	VendorZip	DateFirstPurchased
1	17453	Albemarle Corporation	89431	Jun 14, 1999