Stevens Institute of Technology
Castle Point on Hudson
Hoboken, NJ 07030

CS561 – Programming Assignment 2

Due Dates: 11/29/2017 (Wed.) for Sec. A & 11/30/2017 (Thu.) for Sec. B

Objectives:

- You will continue with evaluating simple report queries and produce the output. As with the assignment #1, you will also express the queries in SQL. The reports below are similar in nature with the reports from the assignment #1; however, there are two main differences between the two: (1) the new reports will require aggregation "outside" the groups (in assignment #1, all of the aggregates were computed for the rows within the groups); (2) some of the aggregates in the new reports will be computed based on other aggregates of the same reports – they are known as "dependent aggregates".

Description:

"Simple Database Application Program #2" (`sdap2.cpp`)

- Generate reports based on the following queries:
  1. For each customer and product, compute (1) the *customer's average sale* of this product, (2) the customer's average sale of the *other products* (3) the average sale of the product for the *other customers* and.
  2. For customer and product, show the average sales before and after each quarter (e.g., for Q2, show average sales of Q1 and Q3. For "before" Q1 and "after" Q4, display <NULL>). The "YEAR" attribute is not considered for this query – for example, both Q1 of 2007 and Q1 of 2008 are considered Q1 regardless of the year.
  3. For customer and product, count for each quarter, how many sales of the previous and how many sales of the following quarter had quantities between that quarter's average sale and minimum sale. Again for this query, the "YEAR" attribute is not considered.

For this assignment, you can write either 3 separate programs, one for each of the 3 reports, or one program generating all of the 3 reports.

Again, the only SQL statement you're allowed to use for your program is:

```
select * from sales;
```

That is, **no `where`** clauses, **no aggregate functions** (e.g., `avg`, `sum`, `count`), etc.

And, you cannot store the 'sales' table in memory.

The following are sample report output (NOTE: the numbers shown below are not the actual aggregate values. You can write simple SQL queries to find the actual aggregate values).

Report #1:

```
CUSTOMER PRODUCT  THE_AVG  OTHER_PROD_AVG  OTHER_CUST_AVG
======== =======  =======  ==============  ==============
Helen    Bread        243             268            1493
Emily    Milk        1426             478             926

. . . .
```

Report #2:

```
CUSTOMER PRODUCT  QUARTER  BEFORE_AVG  AFTER_AVG
======== =======  =======  ==========  =========
Bloom    Bread    Q1           <NULL>       2434
Sam      Milk     Q3              254        325

. . . .
```

Stevens Institute of Technology
Castle Point on Hudson
Hoboken, NJ 07030

Report #3:

```
CUSTOMER PRODUCT  QUARTER  BEFORE_TOT  AFTER_TOT
======== =======  =======  ==========  =========
Emily    Bread    Q4               23     <NULL>
Bloom    Milk     Q2               45         35

. . . .
```

Make sure that:

1. Character string data (e.g., customer name and product name) are left justified.
2. Numeric data (e.g., Maximum/minimum Sales Quantities) are right justified.
3. The Date fields are in the format of MM/DD/YYYY (i.e., 01/02/2002 instead of 1/1/2002).

Grading:

- (50 pts.) Logic/Correctness
- (10 pts.) Programming Style (e.g., comments, indentation, use of functions, etc.). You must include a program header, function header, etc. to clearly state what your program and functions are designed to do. Also for inline comments, please state clearly the purpose of those statements – for you as the programmer and to help others better understand your programming logic.
- (40 pts.) SQL statements to generate the same two reports

NOTE: A program with compilation errors will earn no more than 50 points.

Sample Command Line

```
$ sdap2 [sales], where 'sales' is an optional argument for the table name.
```

Submission: Submit your source code (file) (with your name and CWID on it) on Canvas.

Please include a "README" file with detailed instructions on how to compile and run the code, especially if you are using a language other than C, C++ or Java.

Please remember the following points when you're working on your programming assignments:

1. Your program must compile and execute based on the instructions provided in the README file (i.e., if your programs contain special functions for other compilers and does not compile based on README, you WILL lose 50% of the grade for the assignment).
2. Programming style is 10% of the grade. Please make sure to provide comments for the program, functions, etc. as well as in-line comments as needed. Also, make sure to use meaningful names for your classes, variables, methods/functions, etc. Use proper indentation.
3. In the header comments for your program (i.e., at the beginning of your program), please provide:
    a. General instructions on how to execute your program (e.g., command line for the program and whatever arguments it requires). This can be a simple copy & paste of the README file, or you can provide a simplified bullet listing of the steps for compiling and executing the code.
    b. Justification of your choice of data structures for your program – e.g., if you're using a linked list to maintain whatever information necessary for your program, justify why it's a data structure of your choice, as opposed to, say, arrays. If you're using other more sophisticated data structures, please provide a brief description of the data structures and again justify as to why you chose the data structures for your program.
    c. A detailed description of the algorithm of your program, e.g., how you're computing and maintaining the aggregates (e.g., min, max, avg) for your query output. You can do this with a detailed pseudo code.
4. Remember the only SQL statement allowed in your program is the simple select statement, "**`select * from sales`**". Points will be deducted if you use any other SQL statements in your programs.
5. You are NOT allowed to read in the entire table ('sales') and store them in memory before processing the rows. Instead, you need to read each row (one row at a time), process it and discard it. 50% of the overall grade (50 points out of 100) will be deducted, if any of the rows, other than the current row, are saved in memory (e.g., in a simple variable or an array).

Most importantly, **make sure it's your own work**! If we determine that your program is a copy of someone else's, both you and that someone else will receive 0 for the assignment and possibly additional penalties for the course.

Stevens Institute of Technology
Castle Point on Hudson
Hoboken, NJ 07030

Student's Name:_____

Major Area	Item	Max	Deduct	Score	%	Total
Compilation	If fails, subtract …	**50**				
Logic	Query/Report #1	30				
	Query/Report #2	20				
	Query/Report #3	50				
	"Minimal Scan" implementation (YES/NO)					
	e.g., 1 scan for queries 1 & 2 and 2 scans for query 3					
	Total	**100**			**50%**	
Style	Header Comment	20				
	Function Comment	20				
	Line Comment	20				
	Indentation	10				
	Strings – Left Justified	15				
	Numbers – Right Justified	15				
	Total	**100**			**10%**	
SQL	**Total**	**100**			**40%**	
Sub-Total		**100**				
Penalties	If compilation fails or 'sales' table is cached into memory (subtract 50); For using anything more than 'select * from sales' for programming (vs. for your SQL queries), 25 points will be deducted.					**- 50**
Total						