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

# CS561 – Programming Assignment 2

**Due Dates: 4/28/2017 (Fri.)**

***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, product and state combination, compute (1) the customer's average sale of this product for the state, (2) the average sale of the product and the customer but for the other states and (3) the customer's average sale for the given state, but for the other products.
  2. For customer and product, show the average sales before and after each month (e.g., for February, show average sales of January and March. For "before" January and "after" December, display <NULL>. The "YEAR" attribute is not considered for this query – for example, both January of 2007 and January of 2008 are considered January regardless of the year.
  3. For customer and product, find the month by which time, 1/3 of the sales quantities have been purchased. Again for this query, the "YEAR" attribute is not considered. Another way to view this problem (problem #2 above) is to pretend all 500 rows of sales data are from the same year.

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 STATE CUST_AVG  OTHER_STATE_AVG OTHER_PROD_AVG
======== ======= ===== ========  ============== ==============
Helen    Bread   NY          243             268           1493
Emily    Milk    NJ         1426             478            926

. . . .
```

Report #2:

```
CUSTOMER PRODUCT  MONTH  BEFORE_AVG  AFTER_AVG
======== =======  =====  ==========  =========
Bloom    Bread        1      <NULL>       2434
Sam      Milk         3         254        325

. . . .
```

Report #3:

```
CUSTOMER PRODUCT  1/3 PURCHASED BY MONTH
======== =======  ======================
Emily    Bread    2
Bloom    Milk     3

. . . .
```

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***:

- (80 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.
- (10 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.

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** | | | **80%** | |
| ***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** | | | **10%** | |
| ***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*** | | | | | | |