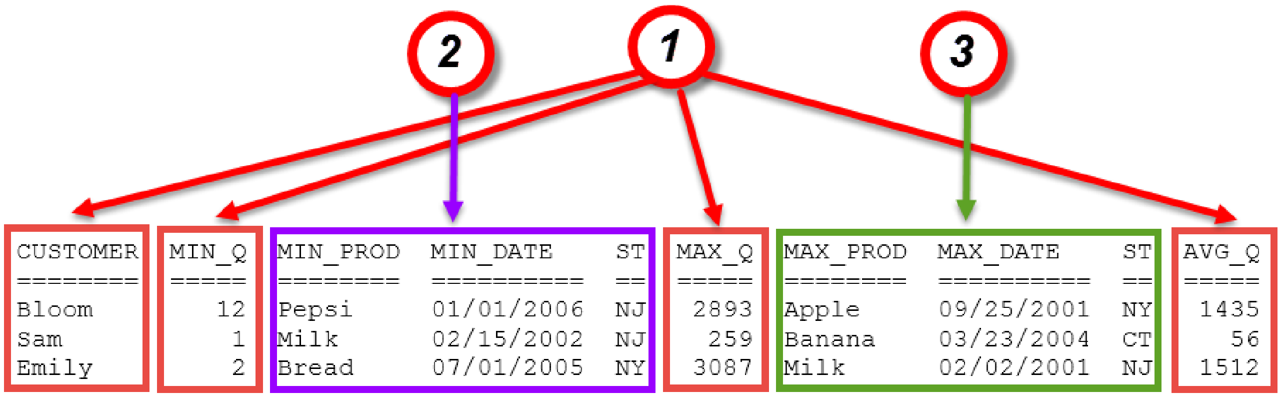
Stevens Institute of Technology
Castle Point on Hudson
Hoboken, NJ 07030

CS561 – Programming Assignment 1

“The Idea”

Due Dates: 10/22/2019 (Tue) for Sec. A & 10/24/2019 (Thu) for Sec. B

To help you understand the mechanics of writing the SQL queries for the assignment, I have outlined below the “structure” of the query results, and how each part is produced and how the three parts are combined to produce the final result set – Please note the output below does not match exactly the output of Query #1 (in your output, MAX comes before MIN), but the idea is the same.



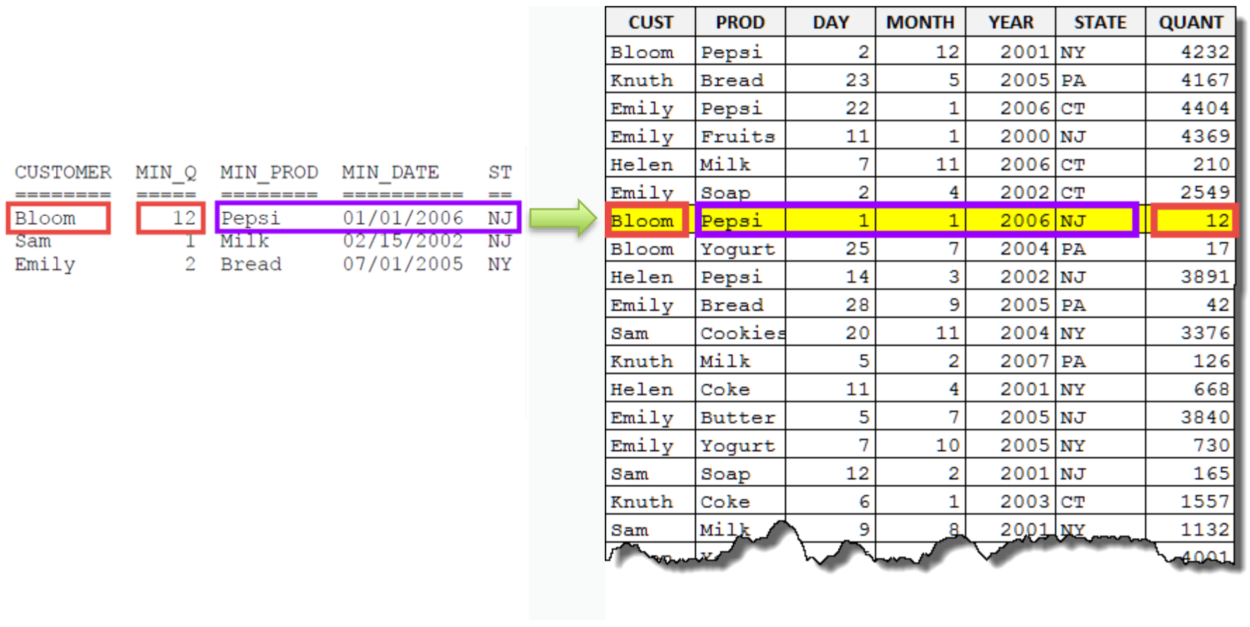
```
CUSTOMER  MIN_Q  MIN_PROD  MIN_DATE    ST  MAX_Q  MAX_PROD  MAX_DATE    ST  AVG_Q
========  =====  ========  ==========  ==  =====  ========  ==========  ==  =====
Bloom        12  Pepsi     01/01/2006  NJ   2893  Apple     09/25/2001  NY   1435
Sam           1  Milk      02/15/2002  NJ    259  Banana    03/23/2004  CT     56
Emily         2  Bread     07/01/2005  NY   3087  Milk      02/02/2001  NJ   1512
```

As shown in the diagram above, the output of the query #1 consists of 3 main parts, and they are constructed in the order listed (query #2 works in a very similar fashion):

1. This is the easy part, where you are simply computing MIN_Q (minimum quantity), MAX_Q (maximum quantity) and AVG_Q (average quantity) based on the grouping attribute, CUSTOMER. It is a simple “group-by” query (or “aggregation operator” in relational algebra).

2. Next step is to capture the corresponding information for each pair of (CUSTOMER, MIN_Q) – that is, MIN_PROD, MIN_DATE and ST (that is, the corresponding PRODUCT, DATE and STATE for the given (CUSTOMER, MIN_Q)). The “challenge” is that the corresponding information is only available in the SALES table; therefore, you will need to “join” the result of Part #1 with the SALES table to pull the corresponding information from the SALES table.

   As you can see below, if you join the results of Part #1 with the SALES table using the condition (predicate) of “**`part1.CUSTOMER = sales.CUST and part1.MIN_Q = sales.QUANT`**”, you will find the corresponding information (highlighted in YELLOW, with a PURPLE box around them) in the SALES table (NOTE: the data you see in the diagrams below are made up and do not match the actual contents of the SALES table you have – this is just for illustrative purposes).



```
CUSTOMER  MIN_Q  MIN_PROD  MIN_DATE    ST
========  =====  ========  ==========  ==
Bloom        12  Pepsi     01/01/2006  NJ
Sam           1  Milk      02/15/2002  NJ
Emily         2  Bread     07/01/2005  NY
```

CUST	PROD	DAY	MONTH	YEAR	STATE	QUANT
Bloom	Pepsi	2	12	2001	NY	4232
Knuth	Bread	23	5	2005	PA	4167
Emily	Pepsi	22	1	2006	CT	4404
Emily	Fruits	11	1	2000	NJ	4369
Helen	Milk	7	11	2006	CT	210
Emily	Soap	2	4	2002	CT	2549
Bloom	Pepsi	1	1	2006	NJ	12
Bloom	Yogurt	25	7	2004	PA	17
Helen	Pepsi	14	3	2002	NJ	3891
Emily	Bread	28	9	2005	PA	42
Sam	Cookies	20	11	2004	NY	3376
Knuth	Milk	5	2	2007	PA	126
Helen	Coke	11	4	2001	NY	668
Emily	Butter	5	7	2005	NJ	3840
Emily	Yogurt	7	10	2005	NY	730
Sam	Soap	12	2	2001	NJ	165
Knuth	Coke	6	1	2003	CT	1557
Sam	Milk	9	8	2001	NY	1132

Results of Part #1 SALES table

3. Next step is to capture the corresponding information for each pair of (CUSTOMER, MAX_Q) – that is, MAX_PROD, MAX_DATE and ST (that is, the corresponding PRODUCT, DATE and STATE for the given (CUSTOMER, MAX_Q), and the idea is exactly the same as the Part #2.

Things to keep in mind:

- The result of each part needs to be "saved" (or stored) in a temporary result set (e.g., a table, view, etc.) – for this, the best method to use the "WITH" clause. WITH is a syntactic construct that allows you to define a "temporary view" (a derived relation/table), and you can find the details in slides 3.43 and 3.44. You can also use "VIEWS" (3.45 – 3.47); however, it's a bit cumbersome to use VIEWS (as compared to WITH clauses). Also note that the WITH syntax for PostgreSQL is slightly different from the one presented in the slides (e.g., in PostgreSQL, you don't need to list the column names when you're defining the temporary results).

- The three parts of the query results are built on top of the results of the previous Parts – that is, the results of Part #2 is created using the results of Part #1, and similarly, the results of Part #3 is created based on the results of Part #3.

- For the date columns of the result set (output), you don't need to concatenate the month, day and year into a date column (e.g., 01/01/2006; instead, just project MONTH, DAY and YEAR columns separately, e.g., 1 | 1 | 2006.