# BASUG 2020 Under the Hood Webinar

| Q# | Attendee Questions | Answer |
|---|---|---|
| 1 | I was just wondering if all of these optimization techniques work with all RDBMS?? For example, I notice I have some issues with some queries with POSTGRESQL | This is a great question. A number of the techniques I showed in my presentation are specific to the SAS environment (MSGLEVEL= and _METHOD). In POSTGRESQL, a keyword that allows users to better understand "under the hood" information and is called, EXPLAIN ANALYZE. The EXPLAIN keyword provides information about what the query planner plans to do with your query before it executes your query. The EXPLAIN ANALYZE keywords will also show the amount of time spent on executing the query (similar to FULLSTIMER and _METHOD) including sort and join usage. Another technique that is relatively universal across RDBMS platforms is to avoid specifying an ORDER BY clause when working with views. When an ORDER BY clause is specified in a view, the data must be sorted each time the view is referenced. Also, if data is accessed often in a program, then it can be more efficient to create a table opposed to a view. Let me know if you'd like additional assistance with query tuning. Here are a few links with guidance for constructing efficient queries for POSTGRESQL users. https://www.geekytidbits.com/performance-tuning-postgres/ and https://statsbot.co/blog/postgresql-query-optimization/ . |
| 2 | Will PROC SQL leverage the (sortedby=) dataset option when grouping variables ex( select state, county, count(distinct phone) from bigdata(sortedby=state) group by state, county) similar to a proc summary with a by statement to use less memory | This is a great question. When specifying an ORDER BY clause in SAS' SQL, the SQL optimizer first determines whether the data has already been sorted in the order of the variable(s). If the table has already been sorted in the requested order, then the optimizer will not (or should not) expend resources to sort the data. Another thing to keep in mind is when a GROUP BY clause is specified without a corresponding summary (or statistical) function. When this happens, your GROUP BY clause is automatically transformed into an ORDER BY clause and a message is sent to the SAS Log. The result of this action could result in more CPU usage and the results being compromised. |
| 3 | I use COMPLEV (name1, name2) function in WHERE clause in a fuzzy match. This query runs very slow. Any advice on how to accelerate this process? | This is a great question. Depending on what your query is designed to do (e.g., lookups, matches, etc.), I've found that the COMPLEV fuzzy matching function typically performs faster than the COMPGED function. One thing I try to do before using the COMPLEV and COMPGED functions is to perform my exact matches first using either PROC SQL or a DATA step MERGE. With the exact matches identified you can then perform the COMPLEV, and if necessary the COMPGED, to address fuzzy matching scenarios. You may also want to consider using the COMPLEV (and COMPGED) function's CUTOFF parameter. When a CUTOFF value is specified, SAS will automatically stop the calculation when reaching the desired "cutoff value". This should help to reduce CPU usage which often times helps with elapsed time. |