My First Attempt in Translating MS Access to SAS

Barry DeCicco
Nissan Technical Center North America
Situation

• There was an existing Access system (‘Durability’) to conduct some durability analyses.
• The queries took pre-processed data from a prior Access system (‘Warranty Analysis’), which in turn took data from the corporate warranty databases (Oracle). That system used over a dozen complex, debugged queries.
• The long-term goal was to run analyses on *all* models by *all* part numbers (1,000’s). The Durability system broke down at 50 combinations.
Limitations

• Existing systems were based on Access. Oracle was the ultimate source, but no programming could be on that end. The software choices were Excel, Access, Minitab and SAS.

• Limited SAS modules (e.g., no SAS/Access).

• All processing was to be one on a single PC.
Solution – Proc SQL

• The relevant tables from the Warranty Analysis system could be exported to .csv files.

• A set of SAS queries in Proc SQL could be written to duplicate the Durability system’s queries in Access.

• Since the Durability system was debugged, there was a ‘gold standard’ for comparison.

How hard could this be?
First problems: SQL’s, not SQL

• Unfortunately, ‘SQL’ does not stand for ‘Standardized Query Language’.

• SAS SQL and Access SQL are not the same SQL, and I had to learn that the hard way.
First step – IIF is not a SAS Command

- Parts of the Access queries looked like this:

\[
\text{IIf}([\text{Date}_1] \leq [\text{Date}_2], -1, \text{IIf}(([\text{Date}_1] - [\text{Date}_2]) \div 365 \times 12 = \text{Int}(([\text{Date}_1] - [\text{Date}_2]) \div 365 \times 12), \text{Int}(([\text{Date}_1] - [\text{Date}_2]) \div 365 \times 12), \text{Int}(([\text{Date}_1] - [\text{Date}_2]) \div 365 \times 12 + 1))) \text{ AS MIS,}
\]

- In SAS Proc SQL, the CASE statement was useful:

\[
\text{CASE WHEN } [\text{Date}_1] \leq [\text{Date}_2] \text{ THEN } 0 \\
\quad \text{WHEN } (([\text{Date}_1] - [\text{Date}_2]) \div 365 \times 12) = \\
\quad \quad \text{Int}(((\text{Date}_1 - \text{Date}_2) / 365) \times 12) \text{ THEN} \\
\quad \quad \quad \text{Int}(((\text{Date}_1 - \text{Date}_2) / 365) \times 12) \text{ THEN} \\
\quad \quad \quad \quad \text{ELSE } \text{Int}(((\text{Date}_1 - \text{Date}_2) / 365) \times 12) + 1 \\
\text{END AS MIS,}
\]
Second step – Calculated Fields

• In Access, to reuse a calculated field, the calculation has to be repeated. This meant that:
  • The opportunity for typo’s was doubled (or tripled,...)
  • The maintenance burden was increased, because multiple changes were always needed.

• This term occurred three times in one Access query:
  Round((DateSerial(Year([Date_3]),Month([Date_3]),1) - DateSerial(Year([Activity_Month]),Month([Activity_Month]),1))/365 *12,0)
SAS CALCULATED Statement

- In SAS Proc SQL, the CALCULATED statement is needed and convenient.

- The calculation is used once to create the field (variable), and after that referred to by: ‘CALCULATED [field alias]’.

- If the ‘CALCULATED’ statement is omitted, SAS will throw an error.
Third - Order of Operations

• I found that Access and SAS have different orders of operation. This took me a while to figure out. I wasn’t expecting it, and the error was not obvious. The numbers were wrong, but not in an obvious way.

  Access: ([Date_4]-[Date_2])/365*12

• To get the same numeric results in SAS, I had to add parentheses to force an order of operation:

  SAS: ((Date_4-Date_2)/365)*12
Fourth – Dates/Times

• I’ve messed around with several different software systems, and expect to mess around with several more. I do NOT expect to ever encounter one where dates and times are not a PITA.

• In Access, ‘DATE’ fields are actually date-time fields. They become DATETIME variables in SAS. Since I didn’t know that, I didn’t quickly understand the error messages.

• The DATEPART() function fixed that by extracting only the date part of a DATETIME variable. I plan on making a habit of using that at the start on all ‘DATE’ fields from other systems.
Conclusions

• The ‘S’ in SQL does not stand for ‘Standard’.

• Each version will have enhancements and ‘non-enhancements’; when you are stuck on an error, check for those.

• Basic things you’d expect to be unchangeable can be changed. Make very sure that it’s not your error, then

• When looking at numeric errors, look at ratios and differences for patterns. Especially since I knew that I was looking at month-year calculations, I should have checked for ratios of 365 and 12.