TO: NatureServe Network Program data managers

FROM: Karen Cieminski and Jodi Shippee

DATE: 22 May 2020

TOPIC: Help implementing the upcoming “Detected” field in the Biotics Visit grid

WHO IS THIS MEMO FOR?

Programs that could use a little headstart on SQL queries to identify negative Visit data in Biotics.

WHAT THE FOLLOWING SCRIPTS AND INSTRUCTIONS DO

Find Visit records that *may* contain negative data, provide a template for reviewing and documenting them, and provide a script for updating the new Detected field.

* first identifying the records that are most likely to be negative data and marking them as Detected = 'N',
* then moving on to records for which it is more difficult to identify by automated script if they are negative data.

NOTES OF CAUTION

Everything that follows are SUGGESTIONS and will likely need to be modified to fit each individual program.

Please note that neither of us are professional programmers, so if you have better or more queries that you have written for your program, please consider sharing them with the Network.

Also note that CASE WHEN THEN statements (see the Query) return a value when the *first* condition is met, then stops evaluating that row of data for other conditions. In other words, if your data meets more than one of the WHEN statements, it will return a value as if it meets only the first that it encounters.

One possible process:

1. Run the DETECTED=FALSE QUERY (below) in Biotics Query Builder and review the output.
2. Read the blue highlighted comments in the query and modify the query as desired to fit your program’s data. If you need help modifying the script, feel free to contact Karen Cieminski (karen.cieminski@state.mn.us) or Jodi Shippee (jodi.shippee@vermont.gov). For the CASE statement #10 in the DETECTED=FALSE QUERY, it may be useful to run this script to identify some of your program’s most popular phrases, adding phrases you suspect may indicate negative data in your database:

select visit\_notes, count (\*) from visit WHERE

UPPER(VISIT\_NOTES) LIKE '%NO %'

OR UPPER(VISIT\_NOTES) LIKE '%NOT %'

OR UPPER(VISIT\_NOTES) LIKE '%NONE %'

OR UPPER(VISIT\_NOTES) LIKE '%UNKNOWN%'

OR UPPER(VISIT\_NOTES) LIKE '%GONE%'

group by visit\_notes

/\*NOTE: The following line excludes visit\_notes that are unique (count=1). You may wish to remove this line if you suspect unique visit\_notes may contain negative data. Alternatively, you may wish to run “SELECT DISTINCT visit\_notes, COUNT (\*) FROM visit GROUP BY visit\_notes ORDER BY COUNT (\*) DESC” to view all data.\*/

HAVING COUNT(\*) > 1 order by count (\*) DESC

1. Once you are happy with the DETECTED=FALSE QUERY, do a final output to an Excel file.
2. Sort the data (if it’s not already) so that the batch of data most likely to be negative data sort together. For example, in Minnesota, this will be the Visits that have the phrase “Negative data”. As you review and confirm that batch of data, enter “N” in the NEW\_DETECTED\_IND column. If you find any records that are actually NOT negative data, enter “Y”.
3. Continue re-sorting (if necessary) and reviewing batches of data and filling out the NEW\_DETECTED\_IND column.
4. When you have completed filling out the NEW\_DETECTED\_IND column (or completed a batch of records), fill in your initials, or other desired text, where it says ‘xxx’ in the line below, then copy the entire string (quotes included) and paste it into row 2 in the “UPDATE\_SCRIPT” column. If you did not add or delete fields in the DETECTED=FALSE QUERY (e.g. commenting in DETECTED\_IND as in note 11), Column U will be the NEW\_DETECTED\_IND column and column A will be the VISIT\_ID column. If this is not the case, change the “U” and “A” below to the correct columns:

**="update visit set detected\_ind='"&U2&"', rec\_last\_mod\_user='xxx' where visit\_id="&A2&";"**

1. After checking that the UPDATE\_SCRIPT now in row 2 runs correctly, pull down to paste it in the rest of the rows. Then copy the update scripts from the UPDATE\_SCRIPT column and run them in SQLDeveloper, SQL Plus, Toad, etc. If you are running the update scripts in batches, fill out the DETECTED\_UPDATED field with an “X”, “Yes”, or date you ran the update in Biotics, to keep track.
2. After you believe you have marked all your Detected = FALSE data in Biotics, you may want to run the DETECTED=FALSE QUERY again, but this time commenting *in* the “WHERE detected\_ind = 'Y'” statement at the end (#12) and commenting *out* the other WHERE statements (#13).
3. If you process this task in batches, please note that other than adding a phrase like “Detected confirmed” to the Visit Notes, neither of us have thought of a good way to document within Biotics which Visits you have already reviewed and found NOT to contain negative data. So keep your spreadsheet(s) that mark positive records you have already reviewed in a safe place ☺ so you don’t need to review those records multiple times. And if you have thought of a way to record which positive-data records do not need to be looked at again (in Biotics), please share!

**DETECTED=FALSE QUERY start**

SELECT VISIT\_ID, VISIT.SOURCE\_FEATURE\_ID,

/\*In the following CASE statement, modify or add to the WHEN statements to include any additional species or taxonomic groups that will help you triage your records.(A function that returns an element’s taxonomic group based on its element\_global\_id can be found at <https://bioticssupport.natureserve.org/support/solutions/articles/201956-function-informal-taxonomy>.).\*/

CASE WHEN HIGHER\_CLASS\_UNIT.CONCATENATED\_CD LIKE 'C%' THEN 'Community'

   WHEN HIGHER\_CLASS\_UNIT.CONCATENATED\_CD LIKE 'IMBIV%' THEN 'Mussel'

   WHEN HIGHER\_CLASS\_UNIT.CONCATENATED\_CD LIKE 'VB%' THEN 'Bird'

   ELSE NULL END AS taxa,

SCIENTIFIC\_NAME, S\_PRIMARY\_COMMON\_NAME COMMON\_NAME,

VISITED\_BY, VISIT\_DATE,

/\*1-The phrase “Negative data” is a code that the MN program used to label our Visits where the species was not detected. For us, these Visits can be automatically set to Detected = FALSE and not reviewed further. If you have a phrase like this, put it (in all caps) in the place of “NEGATIVE DATA” in the CASE statement on the line below. If not, you can delete the whole statement BUT REMEMBER to also remove the column name SAYS\_NEGDATA\_KEYWORD in the ORDER BY statement at the end of this query.\*/

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NEGATIVE DATA%' THEN 'Yes' ELSE NULL END AS says\_negdata\_keyword,

/\*2-The following CASE statement finds Visit records that are likely to be POSITIVE. Depending on your data, this can be more conclusive than the scripts trying to find the negative data. If this seems to be true for your program, you can leave these to review last, as they are likely NOT going to contain your negative data. This worked well for VT data but not as well for MN data, so you should look carefully at your data before relying on this field too much.\*/

CASE WHEN (UPPER(VISIT\_NOTES) LIKE '%FOUND%' and UPPER (VISIT\_NOTES) not like '%NO%')

 OR (UPPER(VISIT\_NOTES) LIKE '%OBSERVED%' and UPPER (VISIT\_NOTES) not like '%NO%')

 OR (UPPER(VISIT\_NOTES) LIKE '%SEEN%' and UPPER (VISIT\_NOTES) not like '%NO%')

 OR (UPPER(VISIT\_NOTES) LIKE '%PRESENT%' and UPPER (VISIT\_NOTES) not like '%NO%')

 OR (UPPER(VISIT\_NOTES) LIKE '%DETECTED%' and UPPER (VISIT\_NOTES) not like '%NO%')

 OR (UPPER(VISIT\_NOTES) LIKE '%NOTED%' and UPPER (VISIT\_NOTES) not like '%NO%')

 OR (UPPER(VISIT\_NOTES) LIKE '%CAPTURED%' and UPPER (VISIT\_NOTES) not like '%NO%')

 OR (UPPER(VISIT\_NOTES) LIKE '%COLLECTED%' and UPPER (VISIT\_NOTES) not like '%NO%')

 THEN 'likely positive'

 ELSE 'could be negative' END AS likely\_positive,

/\*3-The following CASE statement finds all Visit Notes containing phrases with the words “none were…” (found/observed/ etc.) and produces a column with the full phrase (none were found, none were observed, etc.). This group of records is very likely to be negative data, UNLESS you do a lot of combining of locations or dates in one Visit record, e.g. “none were seen on 7/3 but we did see some on 7/4.”\*/

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NONE WERE FOUND%' THEN 'none were found'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE WERE OBS%' THEN 'none were observed'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE WERE SEEN%' THEN 'none were seen'

 WHEN UPPER(VISIT\_NOTES) LIKE '%NONE WERE PRESENT%' THEN 'none were present'

 WHEN UPPER(VISIT\_NOTES) LIKE '%NONE WERE DETECTED%' THEN 'none were detected'

 WHEN UPPER(VISIT\_NOTES) LIKE '%NONE WERE NOTED%' THEN 'none were noted'

  ELSE NULL END AS says\_none\_were\_found,

/\*4-The following CASE statement finds all Visit Notes containing phrases with the words “none …” (found/observed/ etc.) and produces a column with the full phrase. This group of records is very likely to be negative data, UNLESS you do a lot of combining of locations or dates in one Visit record, e.g. “none seen on 7/3 but we did see some on 7/4.”\*/

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NONE FOUND%' THEN 'none found'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE OBS%' THEN 'none observed'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE SEEN%' THEN 'none seen'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE PRESENT%' THEN 'none present'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE DETECTED%' THEN 'none detected'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE NOTED%' THEN 'none noted'

  ELSE NULL END AS says\_none\_found,

/\*5-The following CASE statement finds all Visit Notes containing phrases with the words “none … found/observed/ etc.” (such as “searched for a nest but *none was found*” and produces a column with “none” and the keyword occurring in the record. Due to the wildcard in the middle, this group of records has a moderate likelihood of getting false hits (such as “*none* of the orchids *found* had fruiting heads”), \*/

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NONE%FOUND%' THEN 'none\*found'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE%OBS%' THEN 'none\*observed'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE%SEEN%' THEN 'none\*seen'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE%PRESENT%' THEN 'none\*present'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE%DETECTED%' THEN 'none\*detected'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NONE%NOTED%' THEN 'none\*noted'

  ELSE NULL END AS says\_none\_x\_found,

/\*6-The following CASE statement finds all Visit Notes containing phrases with the words “no …s were found/observed/ etc.” (such as “*no birds were found*”) and produces a column with “no” and the key phrase occurring in the record. This group of records is very likely to be negative data, UNLESS you do a lot of combining of locations or dates in one Visit record, e.g. “no birds were seen on 7/3 but we did see some on 7/4.”\*/

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S WERE FOUND%' THEN 'no \*s were found'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S WERE OBS%' THEN 'no \*s were observed'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S WERE SEEN%' THEN 'no \*s were seen'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S WERE PRESENT%' THEN 'no \*s were present'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S WERE DETECTED%' THEN 'no \*s were detected'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S WERE NOTED%' THEN 'no \*s were noted'

   ELSE NULL END AS says\_no\_s\_were\_found,

/\*7-The following CASE statement finds all Visit Notes containing phrases with the words “no …s found/observed/ etc.” (such as “no individuals found”) and produces a column with “no … were” and the key phrase occurring in the record. This group of records has a moderate likelihood of getting false hits (such as “parents at nest but *no* juvenile*s seen*”)\*/

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S FOUND%' THEN 'no \*s found'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S OBS%' THEN 'no \*s observed'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S SEEN%' THEN 'no \*s seen'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S PRESENT%' THEN 'no \*s present'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S DETECTED%' THEN 'no \*s detected'

  WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S NOTED%' THEN 'no \*s noted'

   ELSE NULL END AS says\_no\_s\_found,

/\*8-The following CASE statement finds all Visit Notes containing phrases with the words “no …s” (such as “No Pipits”). This group of records has a high likelihood of getting false hits.\*/

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NO %S' OR UPPER(VISIT\_NOTES) LIKE '%NO %S %'

  OR UPPER(VISIT\_NOTES) LIKE '%NO %S.%' OR UPPER(VISIT\_NOTES) LIKE '%NO %S;%'

  OR UPPER(VISIT\_NOTES) LIKE '%NO %S,%' THEN 'no \*s'

   ELSE NULL END AS says\_no\_s,

/\*9-The following CASE statement finds all Visit Notes containing the phrase “not found”.\*/

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NOT FOUND%' THEN 'not found' ELSE NULL END AS says\_not\_found,

/\*10-The following CASE statement finds all Visit Notes containing miscellaneous phrases that may be useful (especially in conjunction with a species or taxon group) to find batches of data you can easily confirm as negative data. You can add additional WHEN statements for additional key phrases your program uses. Also add them to the final WHERE statement (#13)\*/

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%UNDETECTED%' THEN 'undetected'

 WHEN UPPER(VISIT\_NOTES) LIKE '%NO DETECTION%' THEN 'no detection'

 WHEN UPPER(VISIT\_NOTES) LIKE '%NO ACTIVITY%' THEN 'no activity'

 WHEN UPPER(VISIT\_NOTES) LIKE '%NO RESPONSE%'

 OR UPPER(VISIT\_NOTES) LIKE '%NO BIRDS RESPONDED%' THEN 'no response'

 WHEN UPPER(VISIT\_NOTES) LIKE '%NO NEST%' THEN 'no nest'

 WHEN UPPER(VISIT\_NOTES) LIKE '%NEST GONE%' OR UPPER(VISIT\_NOTES) LIKE '%NEST IS GONE%' THEN 'nest gone'

 WHEN UPPER(VISIT\_NOTES) LIKE '%INACTIVE%' THEN 'inactive'

 WHEN UPPER(VISIT\_NOTES) LIKE '%SUBFOSSIL%' THEN 'subfossil'

 WHEN UPPER(VISIT\_NOTES) LIKE '%0 %' THEN 'zero'

 ELSE NULL END AS key\_phrases,

/\*11-The following 3 CASE statements produce 3 columns identifying Visit Notes with the words “none”, “not”, or “no”. Visits with entries in one of these 3 columns **but not** any of the previous columns are most likely to be false hits (so it is recommended you review them last).\*/

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NONE %' THEN 'NONE' ELSE NULL END AS says\_none,

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NOT %' THEN 'NOT' ELSE NULL END AS says\_not,

CASE WHEN UPPER(VISIT\_NOTES) LIKE '%NO %' THEN 'NO' ELSE NULL END AS says\_no,

/\*The line below can be commented in once the Detected field is added to Biotics (comment it in by removing the 2 dashes at the start of the line). I am assuming the field will be named “detected\_ind”, but if it is not, you will need to edit this.\*/

--DETECTED\_IND,

/\*The next line puts a placeholder column in your spreadsheet where you can fill in “N” for negative data rows and “Y” for rows you reviewed but found to have positive data.\*/

NULL as NEW\_DETECTED\_IND,

 /\*The next row puts a placeholder column in your spreadsheet where you will paste in the Excel formula (from the instructions above) that will create the Update statement for you to run in Biotics.\*/

NULL as UPDATE\_SCRIPT,

/\*The next row puts a placeholder column in your spreadsheet where you can fill in “X”, “Yes”, or the date you updated the Detected field in Biotics. This column will only be needed for record-keeping if you intend to process your data in batches.\*/

NULL as DETECTED\_UPDATED,

VISIT\_NOTES

FROM VISIT

JOIN SOURCE\_FEATURE ON VISIT.SOURCE\_FEATURE\_ID = SOURCE\_FEATURE.SOURCE\_FEATURE\_ID

JOIN ELEMENT\_SUBNATIONAL ON SOURCE\_FEATURE.ELEMENT\_SUBNATIONAL\_ID = ELEMENT\_SUBNATIONAL.ELEMENT\_SUBNATIONAL\_ID

JOIN ELEMENT\_NATIONAL ON ELEMENT\_SUBNATIONAL.ELEMENT\_NATIONAL\_ID = ELEMENT\_NATIONAL.ELEMENT\_NATIONAL\_ID

JOIN ELEMENT\_GLOBAL ON ELEMENT\_NATIONAL.ELEMENT\_GLOBAL\_ID = ELEMENT\_GLOBAL.ELEMENT\_GLOBAL\_ID

LEFT JOIN HIGHER\_CLASS\_UNIT ON ELEMENT\_GLOBAL.HIGHER\_CLASS\_UNIT\_ID = HIGHER\_CLASS\_UNIT.HIGHER\_CLASS\_UNIT\_ID

JOIN SCIENTIFIC\_NAME ON

ELEMENT\_SUBNATIONAL.SNAME\_ID =

SCIENTIFIC\_NAME.SCIENTIFIC\_NAME\_ID

WHERE

/\*12- If you are re-running this script after marking a bunch of records Detected = FALSE, you can comment in the following line to eliminate those from your results. I am assuming the field will be named “detected\_ind”, but if it is not, you will need to edit this.\*/

--DETECTED\_IND = 'Y' and

/\*13- The following WHERE statement limits your results to only records with key words used above, such as “none”, “no”, etc. If you want to run the query for all your Visits, you can comment out these clauses. However, note that for MN the result was larger than the 25MB limit in Query Builder and so was truncated.\*/

 (UPPER(VISIT\_NOTES) LIKE '% NO %'

or UPPER(VISIT\_NOTES) LIKE '% NOT %'

or UPPER(VISIT\_NOTES) LIKE '% NONE %'

or UPPER(VISIT\_NOTES) LIKE '% NEGATIVE %'

or UPPER(VISIT\_NOTES) LIKE '% INACTIVE %'

or UPPER(VISIT\_NOTES) LIKE '% UNDETECTED %'

or UPPER(VISIT\_NOTES) LIKE '% GONE %'

or UPPER(VISIT\_NOTES) LIKE '% SUBFOSSIL %'

or UPPER(VISIT\_NOTES) LIKE '% 0 %'

)

ORDER BY

SAYS\_NEGDATA\_KEYWORD,

LIKELY\_POSITIVE,

SAYS\_NONE\_WERE\_FOUND,

SAYS\_NONE\_FOUND,

SAYS\_NONE\_X\_FOUND,

SAYS\_NO\_S\_WERE\_FOUND,

SAYS\_NO\_S\_FOUND,

SAYS\_NO\_S,

SAYS\_NOT\_FOUND,

KEY\_PHRASES,

SAYS\_NO,

SAYS\_NOT,

SAYS\_NONE,

VISIT\_NOTES