**Data Checks Pseudo Code**

**Purpose:** Provide team with straightforward direction for how data checks should be implemented, and what kinds of checks should be administered, as well as suggestions for correcting errors identified.

**Data Flow:**

1. Data entry and proofing are completed with ODIN.
2. Data ready to be processed in data checks should meet the following criteria:
   1. DataStatus = ‘Proofed’
   2. DateEntered, EnteredBy, DateProofed, ProofedBy = NOT NULL
      1. Currently ODIN appears to be filling these fields in correctly for all table except: TripInfo. This is likely because Proofing occurs on the SampleEvent level, so there is no clear place in the current process to move a TripInfo record to the next stage.
      2. Note that leaving DataStatus = ‘Not Entered’ does allow for staff to go back and reenter individual SampleEvents. This proved useful when reentering improperly saved data and might also be useful when entering especially large trips, like Shell Budget (one recent trip has 90 quadrats!). For now, I think we should leave that as is.
   3. There are a few solutions to ensuring that TripInfo meets these criteria:
      1. Data Managers could physically confirm that a trip is proofed and then use a short SQL script to fill and update these fields.
      2. We could use ODIN to fill and update these fields. We could add functionality to the ‘Create New Trip’ so that we can “move” or “update” a Trip’s status. Going this route, the button should be renamed ‘Trip Management’ or something similar.
         1. The responsibility for “moving/updating” a trip should probably be on the person proofing.
         2. It would only add a few steps to the end of the proofing process. After all data is proofed, the proofer would navigate to that screen, select the trip, and click “Proofing Complete” or something similar.
3. Data should now be ready to be processed in data checks.
4. There are two separate types of errors we will encounter. Each should be checked for in order.
   1. Database Errors – These are errors that are specific to database structure and formatting. These may not be visible to or fixable by ODIN users.
      1. Examples include duplicate records and PKs, orphan records (no corresponding FK record), childless records, Fields = “” instead of NULL, expected sample events not occurring.
   2. Data Errors – These are errors that do not fall outside of the realm of possibility but may either need extra documentation or deletion.
      1. Examples include weights being higher or lower than expected, unexpected NULL values without Comments, GPS coordinates outside of sample universes.
5. Data checks should create a report of both types of errors found. (Use RMarkdown here?).
6. Staff should check to correct or document these errors. This could be done either by Data Managers or by upper-level staff.
7. Data Managers should correct or document the errors identified.
8. Data Managers should run a secondary check to ensure none of the changes made in the previous step have caused new errors. This step may not be necessary, but we should discuss.
9. Data checks should update DataStatus = “Complete”, DateCompleted and CompletedBy = NOT NULL, and move records from dbo to hsdb.
   1. We should discuss moving the records. Originally, hsdb was intended to hold ALL our completed data regardless of entry method. Though one could make the argument that the schemas can safely hold all our data without moving and instead using UNION in SQL code to pull all data together.
10. Data is now complete.

**Database Errors:**

**ALL TABLES:** These are database errors we should check for in all tables. Actual code may treat each table separately.

1. All required fields should be NOT NULL. Records lacking these fields shouldn’t exist. Example, a trip with no date.
   1. CageCount
      1. CageCountID, Data Type, Cage Color, Total Count
   2. CageSH
      1. Data Type, Cage Color, #
   3. ConditionIndex
      1. OysterID, SampleEventID, ShellHeight
   4. Dermo
      1. OysterID, SampleEventID, ShellHeight
   5. Recruitment
      1. ShellID, SampleEventID, DeployedDate, ShellReplicate, ShellPosition
   6. Repro
      1. OysterID, SampleEventID, Sex, ReproStage, Parasite, BadSlide
   7. SampleEvent
      1. SampleEventID, TripID, FixedLocationID
   8. SampleEventWQ
      1. SampleEventWQID, SampleEventID
   9. SedimentTrap
      1. CupSampleID, SampleEventID, DeployedDate
   10. ShellBudgetQuadrat
       1. QuadratID, SampleEventID, QuadratNumber, TotalSampleVolume, TotalSamleWeight, NumLive
   11. ShellBudgetSH
       1. ShellHeightID, QuadratID, LiveOrDead
   12. ShellPest
       1. ShellPestID, OysterID, SampleEventID, PhotoSurface
   13. SurveyQuadrat
       1. QuadratID, SampleEventID, QuadratNumber, NumLive
   14. SurveySH
       1. ShellHeightID, QuadratID
   15. TripInfo
       1. TripID, TripType, TripDate
   16. Wave
       1. Wave ID
2. Primary Key should be unique. Make sure to search both dbo and hsdb schemas for duplicates.
3. Orphan records, check to ensure that each FK has a corresponding PK in the appropriate table.
   1. Example: in SampleEvent the TripID should have a matching TripID in TripInfo.
4. Childless records, check to ensure that each PK occurs as an FK in the appropriate table(s).
   1. Example: in TripInfo the TripID should occur in at least 1 record in SampleEvent
5. Blanks not NULL. I’m seeing a lot of instances where what should be NULL is saved as “”. We can probably sot that out in the future with ODIN and flows. But we should still check.
   1. Example: SampleEvent.HarvestStatus = “”.
6. Checking that expected sampling occurred. This may be optional or estuary specific.
   1. Example A: in AB for a given month, there should be 15 Recruitment Sample Events. Less than that indicates missed stations, more than that indicates a station double sampled or double entered.
   2. Example B: in AB for a given month, there can be 3 – 15 Collection Sample Events. There are many reasons why a sample Event did or didn’t happen in each month. So, this might be an unhelpful check for this trip type in that estuary.

**Cages:** These are database errors specific to this trip type.

**Collections:** These are database errors specific to this trip type.

1. Primary Keys are longer or shorter than expected.

**Recruitment:** These are database errors specific to this trip type.

1. ShellReplicate should be between 1 – 3.
2. ShellPosition should be between 1 – 12.

**Sediment Traps:** These are database errors specific to this trip type.

**Shell Budget:** These are database errors specific to this trip type.

**Survey:** These are database errors specific to this trip type.

**Wave:** These are database errors specific to this trip type.

**Data Errors:**

**ALL TABLES:** These are data errors we should check for in all tables. Actual code may treat each table separately.

\*\*\* Value ranges below represent historically high or low values. Actual values can be outside that range and may or may not require a Comment, unless noted otherwise. \*\*\*

**CageCount:**

1. Total Count should be between 0 and 30.

**CageSH**

1. Shell Height should be between 10.00 and 200.00 when not NULL.

**ConditionIndex:**

1. ShellHeight should be between 10.00 and 150.00.
2. ShellLength should be between 20.00 and 90.00. AND less than ShellHeight.
3. ShellWidth should be between 8.00 and 50.00. AND less than ShellLength
4. TotalWeight and ShellHeight have a non-linear relationship. 94.4% of our ConditionIndex oysters can be predicted using the following process:
   1. Use this equation to determine a PredictedWeight (PW) using ShellHeight (SH): PW = 0.0005 \* (SH ^ 2.6434)
   2. Use this equation to determine the difference (d) between PW and the observed TotalWeight (TW): d = (PW – TW) / TW
   3. If the value of d is between -0.65 and 1.65, then TotalWeight passes, otherwise, it should be flagged.
5. TarePanWeight should be between 0.90 and 5.00.
6. TissueWetWeight should be between 5% and 40% of the sum of TotalWeight and TarePanWeight.
7. ShellWetWeight should be between 50% and 99% of TotalWeight.
8. TissueDyWeight should be between TarePanWeight and 90% of TissueWetWeight
9. ShellDryWeight should be between 80% and 100% of ShellWetWeight

**Dermo**

1. ShellHeight should be between 10.00 and 150.00.
2. ShellLength should be between 20.00 and 90.00. AND less than ShellHeight.
3. ShellWidth should be between 8.00 and 50.00. AND less than ShellLength
4. TotalWeight and ShellHeight have a non-linear relationship. 94.9% of our Dermo oysters can be predicted using the following process:
   1. Use this equation to determine a PredictedWeight (PW) using ShellHeight (SH): PW = 0.0005 \* (SH ^ 2.6434)
   2. Use this equation to determine the difference (d) between PW and the observed TotalWeight (TW): d = (PW – TW) / TW
   3. If the value of d is between -0.65 and 1.65, then TotalWeight passes, otherwise, it should be flagged.
5. ShellWetWeight should be between 50% and 99% of TotalWeight.

**Recruitment:**

1. Difference between DeployedDate and Retrieved Date [] should be between 21 and 36.
   1. select datediff(day, DeployedDate, cast(substring(ShellID, 8, 8) as date)) as DateDifference from Recruitment;
2. NumTop should be between 0 and 50; or NULL (max recorded = 258; only 4 (0.001%) out of 266,720 records have been over 200; only 97 (0.04%) over 100, only 1,012 (0.38%) over 50)
3. NumBottom should be between 0 and 50; or NULL (max recorded = 251; only 7 (0.003%) out of 266,716 records have been over 200; only 227 (0.08%) over 100, only 1,893 (0.71%) over 50)
4. NumBottom is NULL where ShellPosition = 2 – 5 or 8 – 11 need a comment.

**Repro**

1. If Sex = M/F, then ReproStage must = NULL
2. If Parasite = Buceph, then ReproStage must = NULL
3. If ReproStage = 4, then Sex must = Z

**SampleEvent**

1. Cages
2. Collections (WHERE SampleEventID like ‘%COLL%’)
   1. If Estuary = AB, then LatitudeDec and LongitudeDec should not be NULL
   2. We should check that Lat and Lon are near Lat/Lon for the FixedLocationID.
      1. [SQL code here]
3. Recruitment (WHERE SampleEventID like ‘%RCRT%’)
   1. There are no fields that require a Comment if they are NULL.
4. Sediment Traps (WHERE SampleEventID like ‘%SDTP%’)
   1. There are no fields that require a Comment if they are NULL.
5. Shell Budget (WHERE SampleEventID like ‘%SHBG%’)
   1. If Estuary = AB, then LatitudeDec and LongitudeDec should not be NULL
   2. We should check that Lat and Lon are near Lat/Lon for the FixedLocationID.
      1. [SQL code here]
6. Survey (WHERE SampleEventID like ‘%SRVY%’)
   1. If Estuary = AB, then LatitudeDec and LongitudeDec should not be NULL
   2. We should check that Lat and Lon are near Lat/Lon for the FixedLocationID.
      1. [SQL code here]
7. Wave (WHERE SampleEventID like %WAVE%)

**SampleEventWQ**

1. Water Quality should be taken at each sample event, so any NULL should have a Comment for these fields:
   1. Temperature, Salinity, DissolvedOxygen, pH, Depth, SampleDepth.
   2. NULL for other fields may or may not need Comments as some labs take WQ metrics others don’t.
   3. Also, Since some are duplicates there can be records where all values are NULL. These **do** need a comment.
2. ALL
   1. Temperature should be between 10.0 and 33.0.
   2. Salinity should be between 0.10 and 35.00.
   3. DissolvedOxygen should be between 1.00 and 13.00.
      1. Max DO and Temperature are linked (higher Temp -> lower DO) we can calculate this to determine if a value falls outside of range. FIM has a calculation they use.
   4. pH should be between 6.50 and 8.50.
   5. Depth should be between 0.00 and 3.50.
   6. SampleDepth should be between 0.00 and Depth.
   7. Secchi should be between 0.00 and Depth.
   8. TurbidityYSI should be between -2.00 and 30.00. (Yes, negatives are possible).
   9. TurbidityHach should be between -2.00 and 30.00.

**SedimentTrap**

1. Difference between DeployedDate and Retrieved Date should be between 21 and 36.
   1. select datediff(day, DeployedDate, cast(substring(SampleEventID, 8, 8) as date)) as DateDifference from SedimentTrap;
2. If any of these fields are NULL, there should be a Comment.
   1. FilterTareWeight, PanTareWeight, FilterDryWeight, PanDryWeight, NumDrills, NumCrabs, NumHermitCrabs, NumFish, NumOtherBiota.
3. FilterTareWeight should be between 1.30 and 2.00.
   1. If no filter used, trap missing, etc. value should be NULL not 0.
4. PanTareWeight should be between 2.00 and 400.00.
   1. If no pan used, trap missing, etc. value should be NULL not 0.
5. FilterDryWeight should be between FilterTareWeight and 10.00.
   1. If no filter used, trap missing, etc. value should be NULL not 0.
6. PanDryWeight should be between PanTareWeight and 600.00.
   1. If no pan used, trap missing, etc. value should be NULL not 0.
7. NumDrills should be between 0 and 5.
   1. If trap missing value should be NULL.
8. NumCrabs should be between 0 and 2.
   1. If trap missing value should be NULL.
9. NumHermitCrabs should be between 0 and 10.
   1. If trap missing value should be NULL.
10. NumFish should be between 0 and 2.
    1. If trap missing value should be NULL.
11. NumOtherBiota should be between 0 and 1.
    1. If trap missing value should be NULL.

**ShellBudgetQuadrat**

1. TotalSampleVolume should be between 0.05 and 8.00.
2. TotalSampleWeight should be between 0.10 and 16.00.
3. LiveOysterVolume should be between 0.00 and 2.00.
4. LiveOysterWeight should be between 0.00 and 4.00.
5. NumDrills should be between 0 and 5.
6. DrillWeight should be between 0.00 and 0.10.
7. OtherBiotaWeight should be between 0.00 and 0.20.
8. NumLiveOysters should be between 0 and 1000.
9. NumDeadOysters should be between 0 and 500.
10. OysterShellVolume should be between 0.00 and 1.00.
11. OysterShellWeight should be between 0.00 and 2.00.
12. PlantedShellVolume should be between 0.00 and 5.00.
13. PlantedShellWeight should be between 0.00 and 10.00.
14. ShellHashVolume should be between 0.00 and 1.00.
15. ShellHashWeight should be between 0.00 and 2.00.
16. BlackAndOtherSubstrateVolume should be between 0.00 and 0.50.
17. BlackAndOtherSubstrateWeight should be between 0.00 and 1.00.
18. The sum of the following fields should be 90 – 110% of TotalSampleWeight
    1. LiveOysterWeight, DrillWeight, OtherBiotaWeight, OysterShellWeight, PlantedShellWeight, ShellHashWeight, BlackAndOtherSubstrateWeight
19. The sum of the following fields should be 90 – 110% of TotalSampleVolume
    1. LiveOysterVolume, OysterShellVolume, PlantedShellVolume, ShellHashVolume, BlackAndOtherSubstrateVolume
20. Volume and Weight are related. Corresponding volume should be 40 – 60 % of corresponding weight.

**ShellBudgetSH**

1. ShellHeight should be between 1 and 150.
2. ShellHeight can be NULL **only if** there are no other **LiveOrDead** records in the quadrat.
   1. For Live: must be [LiveOrDead = ‘Live’ AND ShellHeightID is like ‘%\_001’].
   2. For Dead: must be [LiveOrDead = ‘Dead’ AND no other ‘Dead’ in Quadrat].

**ShellPest**

1. TotalArea should be between between 50% and 90% of {TotalHeight x TotalLength}
2. TotalHeight should be between 60% and 130% of ConditionIndex.ShellHeight
3. TotalLength should be between 60% and 130% of ConditionIndex.ShellLength
4. ClionaArea should be between 0.00 and 90% of TotalArea.
5. PolydoraArea should be between 0.00 and 90% of TotalArea.
6. ClamCount should be between 0 and 10.
7. ClamAverageDiameter should be between 0.00 and 6.00.

**SurveyQuadrat**

1. NumLive should be between 0 and 500.
2. NumDead should be between 0 and 300.
3. TotalVolume should be between 0.05 and 5.00.
4. TotalWeight should be between 0.10 and 10.00.
5. Volume and Weight are related. Volume should be 40 – 60 % of weight.
6. NumDrills should be between 0 and 5.

**SurveySH**

1. ShellHeight should be between 1 and 150.
2. ShellHeight can be NULL **only if** there are no other records in the quadrat. ShellHeightID must be like ‘%\_001’.

**TripInfo**

1. TripDate and TripType don’t match TripID

**Wave**

1. All weights should be between 0 and 500.