1. [DONE] [YES] Should we add “ValueAsBoolean” as a calculated persisted indexed field on AttributeValue? It would help with BI, correct BooleanField values not stored as ‘True’ or ‘False’ (CCV has ‘Yes’ and ‘No’ stored in some of these), and also make some DataFilters faster.
2. [MAYBE NOT] [YES] Should we add “ValueAsGuid” as a calculated persisted indexed field on AttributeValue? It would help make some DataFilters faster.
3. [TODO] [DEPENDS as needed] What should the “Blank String” be for the various Analytic fields, grab from this list when obvious:
   1. ‘None’
   2. ‘Unknown’
4. [DONE][YES, 250 just in case other fields are part of the index] Should we limit the max length of Attribute Fields on AnalyticsDimPersonHistorical and AnalyticsDimPersonCurrent to nvarchar(250).  The advantage is that they could be indexed, but anything larger than 450 would have to be stored as NULL (since we probably want it to be either exact or nothing). Another thought is should anything larger than 250 even be in the AnalyticsPerson tables?
5. [DONE]Naming convention for the de-normalized Attribute value fields?
   1. Proposed:
      1. Option#1: “attribute\_” + attribute.Key.RemoveSpecialCharacters()
      2. [YES, try to do this.  Figure out a way to do the ‘DROP if doesn’t exist anymore’]Option#2: no prefix, Native Field wins when a conflict
6. [DONE]Do we want ALL the fields from Person to be in AnalyticsDimPersonHistorical and AnalyticsDimPersonCurrent? I assume we will do most of them, but the following differences
   1. [NO] IsSystem
   2. [ADD, but doesn’t trigger history] PhotoUrl as a field, would be handy for some BI stuff
   3. PhoneNumbers?
      1. [OK] Option#1: Flatten to 3 fields, HomePhoneNumber, CellPhoneNumber, WorkPhoneNumber?
      2. Option#2: just a PhoneNumber, and picks first in this priority: Cell, Home, Work
   4. [YES] Gender as String “Male” “Female”, “Unknown”
   5. Email Fields?
      1. [YES] Email
      2. [No] IsEmailActive
      3. [No] EmailNote
      4. [YES] EmailPreference (enum to string)
   6. Birthdate Fields
      1. [YES, which means we will need a AnalyticDimDateBirthDate view]BirthDateKey (Link to AnalyticDimDate??)
   7. NOTE: I have DIM tables for all the “Lookup” fields, such as AnalyticsDimPersonConnectionStatus, AnalyticsDimPersonMaritalStatus, etc
7. [DONE]What fields trigger history?
   1. [YES] [RecordStatusValueId],
   2. [RecordStatusLastModifiedDateTime],
   3. [RecordStatusReasonValueId],
   4. [YES] [ConnectionStatusValueId],
   5. [ReviewReasonValueId],
   6. [YES] [IsDeceased],
   7. [TitleValueId],
   8. [FirstName],
   9. [NickName],
   10. [MiddleName],
   11. [YES] [LastName],
   12. [SuffixValueId],
   13. [PhotoId],
   14. [BirthDay],
   15. [BirthMonth],
   16. [BirthYear],
   17. [Gender],
   18. [YES] [MaritalStatusValueId],
   19. [AnniversaryDate],
   20. [GraduationYear],
   21. [YES] [GivingGroupId],
   22. [GivingId],
   23. [GivingLeaderId],
   24. [Yes] [Email],
   25. DELETE[IsEmailActive],
   26. DELETE [EmailNote],
   27. [Yes][EmailPreference],
   28. [ReviewReasonNote],
   29. [InactiveReasonNote],
   30. [SystemNote],
   31. [ViewedCount],
   32. [Guid],
8. [DONE] What Attributes trigger history
   1. The ones with IsAnalyticHistory = TRUE
9. [TODO]Primary Family.  The GetFamily of Person (<https://github.com/SparkDevNetwork/Rock/blob/3ee719f7b68d273bf8a5b7ed2baf28ad00055ce7/Rock/Model/Person.cs#L2668>), isn’t guaranteed to give a consistent result in the cases of multiple active families.  I propose we add an “ORDER BY ID” so that it will be consistent. That way it would always be the “NEWEST” family record first (that’s what CCV’s datamart tables do).  Another solution is to introduce a “PrimaryFamilyId” to Person so that Rock no longer has to guess which family is their primary family (it would also speed up a lot of stuff).  It could default to whichever family they were added to first, but would be editable (sort of like GivingGroup). It might take a bit of work to figure out how we want to do this since a person can be moved from families, families can become inactive, removed, etc.  So, maybe we could go with guessing like we do for now and figure this out later.
   1. [THIS WAY]
      1. Sort so that [IsAdult] (Order so that Adult then Child), then by [OLDEST] (order by Group.Id ASC), wins.  In other words, if they are in two families, and they are an adult in one and a child in another, choose the family where they are an Adult.  If they are the same in both, choose the oldest family.
10. [TODO] [YES] Do we want AnalyticsDimPersonHistorical and AnalyticsDimPersonCurrent to be available to DataViews and Reports? I assume yes, but it would take some work to figure out how to deal with the Dynamic fields (Attributes)
    1. [YES] When you pick AttributeDimPerson for example, the field picker shows all the fields, including the attribute fields
    2. Random Notes
       1. Have a AttributeValuesXML column on AnalyticsDimPersonHistorical so that Attribute Values can be loaded efficiently from SQL without having to do a “One at a Time/Record By Record thingy”
       2. Have a “AnalyticsDynamicFields” Data Select that is smart enough to figure out how to create the Linq expression for SELECTs
       3. Have a “AnalyticsDynamicFields” Data Filter that is smart enough to figure out how to create the Linq expression for WHERE statements.
11. [TODO]Metrics
    1. We had briefly talked about making Metrics available to BI in a BI Friendly format.  I have solution that will work that we could talk about. Here are the basics
       1. Right now, I have a dev tool that does it, but I was thinking it would be in C# if we do it for realz, C:\Projects\Rock\Dev Tools\Research\Analytics\BuildAnalyticViewFromMetric.sql
       2. Each Metric would have a “Enable Analytics” setting[YES] ~~[NO](or maybe we just enable it for all of them regardless since it’ll just be a view)~~
       3. When a metric config is saved (MetricDetail), it will create a view that presents the MetricValues as a PIVOT table, for example
          1. SELECT pvt.Id

  ,cast(pvt.MetricValueDateTime AS DATE) AS [MetricValueDateTime]

  ,pvt.YValue

  ,pvt.[67] AS [CampusId]

  ,pvt.[54] AS [ServiceValueId]

  ,pvt.[31] AS [AreaId]

FROM (

    SELECT

          mv.Id

      ,mv.YValue

      ,mv.MetricValueDateTime

      ,mvp.EntityId

      ,mp.EntityTypeId

    FROM MetricValue mv

    JOIN MetricValuePartition mvp ON mvp.MetricValueId = mv.Id

    JOIN MetricPartition mp ON mvp.MetricPartitionId = mp.Id

    WHERE mv.MetricId = 73

    ) src

pivot(min(EntityId) FOR EntityTypeId IN ([31]

,[54]

,[67]

)) pvt

* + 1. Naming Convention for Metric Analytic Views would be
       1. [YES] ‘AnalyticsFactMetric’ + Metric.Name
    2. ~~The Views would be maintained by a Rock Job that would detect if the View needs to be regenerated~~
    3. MetricDetail takes care to ensure that the View is consistent with the Metric configuration
    4. Delete Metric cleans up orphaned MetricViews

1. Rock.Model.Attribute
   1. We had talked about adding an “IsAnalytic” as a field Rock.Model.Attribute,  but that would only be shown in the Attribute Config UI when the Model implement IAnalytic or IAnalyticHistorical.  I assume that would default to false for all existing Person/Family attributes? Or do we need to talk about that on a case by case (for the core ones)
   2. [DONE]Two new fields on Rock.Model.Attribute table
      1. Bool: IsAnalytic (Adds the field to the AnalyticDimPerson tables)
      2. Bool: IsAnalyticHistory (Triggers a AnalyticDimXXXHistory record if changed)
   3. What should we store in the “Analytic” fields in the AnalyticDimPerson and AnalyticDimFamily tables?
      1. [DONE] [YES] Option #1: Do “Whatever FormatValue does”, but that would mean that at least some of the ETL would probably have to be done in C# with lots of traffic between the Rock server and the SQL Server
         1. It could be done where a SQL statement or StoredProc could do most of the heavy data work for normal Person fields and maybe some of the straight forward attributes (text, Datetime, Numeric, Boolean), and then a Rock Job would use C# to fill in the AttributeValues that have to be done using the FieldType.FormatValue
      2. [NO] Option #2: We support a limited number of FieldTypes, and add support as needed.
      3. ~~Other Random Mike Notes~~
         1. ~~There are some pretty straight forward answers for some Attribute FieldTypes~~
            1. ~~Text:  the Raw value as a nvarchar(max)~~
            2. ~~Boolean: a string of ‘True’ or ‘False’ (it would normalize all the variations on True/False “Yes,No,1,0,True,False,true,false,yes,no,Y,N,T,F,etc)~~

~~Option #1: ‘True’ or ‘False’ as strings~~

~~Option #2: 1 or 0 as SQL Bit fields (which PowerBI interprets as True/False fields)~~

* + - * 1. ~~Date/DateTime: The ValueAsDateTime as a SQL DateTime field~~
        2. ~~Integer/Decimal/Currency: The ValueAsNumeric as a SQL NUMERIC field~~
      1. ~~Then is gets a little fuzzier, a couple of biggies would be~~
         1. ~~DefinedValue~~

~~The DefinedValueId? Or the DefinedValue.Value? or the DefinedValue.Description?~~

* + - * 1. ~~Campus~~

~~Campus.Name?~~

* + - * 1. ~~GroupField~~
        2. ~~GroupType~~
        3. ~~Location~~
        4. ~~FinancialAccount~~
        5. ~~BinaryFile~~
        6. ~~Campuses~~

~~Comma-Delimited Campus.Names?~~

* + - * 1. ~~Category~~
        2. ~~PhoneNumber~~
      1. ~~Then there are a bunch more, plus any that come from Plugins, or future releases of Rock~~