Document Summary

|  |  |
| --- | --- |
| **Document Title** | Data Management Guide to Data Profiling |
| **Document Description** | Framework to guide data profiling work performed by Data Management |
| **Purpose** | To serve as an internal guide for working through data profiling tasks in an efficient and effective manner |
| **Internal or External Facing** | Internal |
| **Intended Audience** | Data architects, engineers, and analysts (anyone involved in data profiling) |

Release Notes

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Notes** |
| V1 | 2016/12/22 | Initial release |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Data Management

Guide to Data Profiling

# Overview

For anyone who has done even a modest amount of data profiling, it is notoriously easy to get lost in the data. The purpose of this guide is to provide a structured way for thinking about data profiling projects, whether small or large, that reduces wandering. In addition to this guide, a template is also available to help with organization and facilitate the easy communication of findings.

This guide consists of brief discussion of the difference between exploratory and outcome-driven data profiling, a step-by-step method for completing a data profiling project, and a list of data profiling tasks. The steps for performing data profiling should be relatively stable regardless of the project. The individual tasks, on the other hand, should serve as a picklist that can be drawn on to formulate a data profiling plan. Some tasks (such as identifying grain) may be needed more often than others (such as testing linkages), but the goal is to provide a fairly comprehensive list of basic data profiling tasks that can be drawn on and arranged into a plan that addresses a majority of data profiling projects.

It is important to emphasize that not all of the data profiling tasks provided in this document will be relevant to a given data profiling project, and it does not necessarily provide a comprehensive list of the tasks that will ultimately be needed. The tasks used to complete a data profiling project and the order in which they are executed depend on the nature of the project and the deliverables being produced. It is also important to think critically about the results, formulate follow-up questions at the end of every task, and prioritize (or deprioritize) those follow-up questions based on the overall requirements of the project. When doing so, however, it is important to align any follow-up work with project requirements and not get lost in the data.

# Types of Data Profiling

At a high level, we categorize data profiling tasks as either *exploratory* or *outcome-driven*. Most data profiling projects will be made up of some mixture of the two, but a data profiling project *as a whole* should always have an outcome-driven focus—that is, it should always be aimed at determining whether a specific set of requirements are supported by the data. Exploratory and outcome-driven data profiling are defined here as follows:

**Exploratory Data Profiling** – Data profiling intended to achieve a deeper understanding of a dataset or group of datasets. Exploratory data profiling is focused on answering general, open-ended questions about the data such as “What is contained in this dataset?” or “What can the distributions of these columns tell me about a particular group of users?”

**Outcome-Driven Data Profiling** – Data profiling intended to test a hypothesis, capability, or assumption. Outcome-driven data profiling is focused on answering specific, close-ended questions about the data such as “Can I join these two datasets using the user ID column?” or “How long does it take for the status of a user’s subscription to appear in the dataset after the actual change occurs?”

# Steps for Completing a Data Profiling Project

The steps outlined below are high level and do *not* replace the standard workflows used by Data Management. In particular, validation is as important in data profiling as it is in any other workflow, and it should be performed for every work step in a data profiling project. In addition, the below steps should be used as a high level framework (the “what”) while the actual implementation of these steps (the “how”) is left to the data profiler. Best practices, standards, and templates will be introduced over time wherever possible though in an effort to achieve greater standardization.

1. **Define requirements**
   1. The requirements of the data profiling project consist of the objectives for performing the data profiling, and should be derived from the business or functionalrequirements[[1]](#footnote-1) of the overall project
   2. Requirements should be outcome-driven, and focused the specific questions that need to be answered to enable subsequent work
   3. If the outcome is unclear, or what is being enabled is unclear, revisit the business or requirements or objectives for the overall project or engagement to better define the purpose of the data profiling exercise
2. **Identify datasets**
   1. Work with internal and external stakeholders and any available documentation to identify the datasets that will be needed to satisfy requirements
3. **Request access to the data**
   1. Obtaining access to the necessary data and systems can take a significant amount of time, especially when working directly in a client’s data environment
   2. Obtaining access should always be treated as a significant risk to the overall project timeline, and should be addressed as early in the project as possible
4. **Review the environment, documentation, and metadata**
   1. Review the environment in which the data sits, how that environment impacts the data, and whether the environment is conducive to data profiling work
   2. If the environment is not conducive, explore options for pulling data (whether a sample or a full copy of the data) into another environment that is conducive (e.g., SQL DB)
   3. Review any available documentation on the data to better understand what is contained in the dataset, issues impacting the data, known quality concerns, and so on
   4. Review any available information regarding the schema (column names, data types, and any other information that’s available on what is stored in the dataset)
   5. Review the typical size of the dataset in terms of file size, number of columns, and number of rows using multiple run dates if available
   6. Understand how the size of the dataset (file size, number of columns, and number of rows) grows or diminishes over time, and the rate at which this change occurs
5. **Create data profiling plan**
   1. Document the list of tasks that will be needed to support requirements
   2. Document assumptions and expectations for each task
   3. Revise the plan as you work through the data profiling, but stay focused on the requirements—any significant deviation from the original plan should be discussed with project leads, especially if it impacts the project timeline
   4. When providing timeline estimates for data profiling, always estimate from the start of the data profiling, not a specific date—specific dates are dependent upon access, and gaining access is usually beyond our control
6. **Obtain a copy or sample of the dataset, and load it into an environment that is conducive to data profiling (a SQL DB, for example, as opposed to a big data environment like Cosmos)**
   1. Generally speaking, data profiling should not be performed on a production dataset (a dataset that is used directly by business users or analysts, a dataset that enables some automated downstream process, etc.), with the following exceptions:
      1. The production dataset cannot be copied or sampled due to security restrictions
      2. There are very few dependencies on the dataset (such as a small dataset owned and managed by the Insights team) and profiling work will not disrupt other users or downstream processes
      3. Access restrictions prevent altering or deleting the production dataset, and therefore minimize risk to downstream users and processes
      4. Even in cases where profiling needs to be performed on a production dataset, a sample should still be used for developing and debugging the data profiling script whenever possible
      5. If a production dataset must be used, the data profiling script should be run on the production dataset only after it has been fully debugged and validated through the standard code review process used by Data Management
   2. When obtaining a sample, consider the following:
      1. Work with engineering to determine if any additional steps are required to obtain a random sample that is representative of the full dataset—depending on the environment, the data may not be ordered and simply selecting the top N rows (100, for example) will return a sufficiently random sample of data
      2. In some cases, questions can be answered to satisfaction using a sample, and scripts do not need to be run on the full dataset (an ID column that contains nulls in a sample will also contain nulls in the full dataset, for example)—use your best judgement to determine whether the full dataset is needed to answer a specific question, while also taking into account the level of effort required to deploy a script on the full dataset
      3. If a sample is random (i.e., representative of the full parent dataset) and sufficiently large, it should be enough to review distributions, answer questions about the proportion of records or IDs that are impacted by a given data quality issue, and so on—it may not be necessary to profile the full dataset
7. **Execute data profiling plan**
   1. If appropriate, review schema and select column subsets for profiling
      1. With the exception of identifying grain, it is often not necessary to perform profiling on every column in a dataset
      2. Once grain has been determined, use your best judgement to determine which columns should be included in the data profiling task or exercise
   2. Perform data profiling according to the data profiling plan and document findings
   3. Flag any items that require follow-up investigation, but whenever possible, hold off on implementing follow-up work items until the initial profiling plan is complete
8. **Summarize findings and assess whether requirements have been met**
   1. If requirements have not been met, identify the steps needed to meet them (if known) and work with project leads to determine whether and how work should proceed
9. **Share initial findings with internal stakeholders who are dependent on the findings**
10. **Summarize findings (including data quality issues) and surface as appropriate to the client**
    1. When communicating data profiling results to the client, it is important to answer the “so what?” question by recommending clear action items based on the results, follow-up research that needs to take place, client stakeholders that should be engaged (such as client engineering stakeholders, for example), etc.
    2. It is usually *not* necessary (or advisable) to surface *all* findings from a data profiling project to the client
       1. Instead, use your best judgement to identify items that are valuable to the client such as items that improve the client’s general understanding of the data or call attention to issues that pose a risk to information on which the client depends
       2. When calling attention to data quality issues or items that pose a risk, be sure to include recommendations on how to deal with those items and an assessment of overall impact (e.g., “3% of ID values are invalid” as opposed to “The ID column contains values that are invalid”)
11. **Work with project leads to prioritize any follow-up items for further profiling**
    1. Reconcile follow-up work items with requirements and prioritize
    2. For any questions that are prioritized for follow-up investigation, identify clear objectives for each question and ensure that these questions align with the requirements of the overall project or engagement
12. **Iterate**
    1. Create a new data profiling plan and w through the steps outlined in this document

# Data Profiling Tasks

The below tasks represent general activities that can be used to complete a data profiling project. These should serve as a picklist to help in generating a data profiling plan, but do not necessarily represent a comprehensive list of all the tasks that may ultimately be needed on a data profiling project.

NOTE: The below tasks do not cover the topic of parity. Parity checks may sometimes be included in data profiling, however, and parity requirements are an important part of many data infrastructure projects. Parity checks are often performed during the QA phase of a project though, and will therefore be addressed in a separate framework.

* **Review environment, documentation, and metadata[[2]](#footnote-2)**
  + Review the environment in which the data sits and how it impacts the data
  + Review any available documentation on the data to better understand what is contained in the dataset, issues impacting the data, known quality concerns, and so on
  + Review any available information regarding the schema (column names, data types, and any other information that’s available on what is stored in the dataset)
  + Review the typical size of the dataset in terms of file size, number of columns, and number of rows using multiple run dates if available
  + Understand how the size of the dataset (file size, number of columns, and number of rows) grows or diminishes over time, and the rate at which this change occurs
  + Preview the data to get a sense of what values are contained in each column (in SQL, this can be done by using a “select top 100 \*” statement, for example)
  + Generate a high level summary of the dataset and what it contains, if known, or hypotheses and questions if unknown
  + Generate a high level summary of each column and what it contains, if known, or hypotheses and questions if unknown
  + Review where the dataset or tables sits in a broader data model if available, and use this information to provide context on how the dataset is used
  + Document any other items that may impact usability of the dataset (e.g., how structured the data is, whether column headers repeat themselves at various points in the data as with log files that have been combined into a single dataset, etc.)
* **Check for duplicate rows**
  + Check for duplicates using the following steps:
    - Perform initial row count
    - Group by or select distinct for all columns and obtain a second row count
    - Compare the second row count to the first row count
    - If the second row count is smaller than the first, there are duplicate rows
  + If duplicates exist, document the percentage of records for which there are duplicates
  + Duplicate rows may be easily addressed through de-duping (using group by or select distinct for all columns), but only if duplicate records are in fact duplicates, and not truly unique records that should be treated as such (included in aggregations, metrics, etc.)
    - One situation in which duplicates may actually be unique records is when a dataset is produced from a parent dataset by incorrectly dropping columns that are needed to define the grain of the dataset
    - How duplicates are dealt with should depend on how many rows are impact, and what information is available about the dataset and the upstream processes used to generate it
    - If the opportunity exists, check with client engineering stakeholders who are familiar with the dataset to clear up any questions surrounding duplication
    - If impact to requirements is minimal, the most efficient approach may be to simply drop duplicate columns
  + Regardless of the cause for the duplicate rows, it may be appropriate (or even necessary, as in the case of identifying potential grain) to drop them before other profiling tasks can be completed—in situations such as this, make a note of the issue and the extent to which the dataset is impacted, then remove the duplicate rows and proceed with the data profiling as planned
* **Review simple counts**
  + Generate a list of assumptions and expectations about the dataset (how many users should be represented, for example) that can be addressed with simple counts
    - This step may require involvement from the client or internal stakeholders who have prior experience working with the data
  + Review simple counts (all-up rows, unique IDs, etc.)
  + Check these against the assumptions that were previously generated
* **Review ID columns**
  + Check if ID columns are valid for all records by looking for nulls, empty strings, or unexpected numerical values (such as 0, negative values, or values that do not conform to expectations or a standard format that is apparent in the data)
  + Invalid ID values are not necessarily an issue depending on what the dataset is going to be used for, but they should be documented and flagged as a potential quality issue
  + Oftentimes unexpected values may be numerical codes used to indicate null or missing values (-1 is often used to indicate a null or missing value, for example)
  + If ID values are numeric, it may sometimes be useful to check if ID values are contiguous (or incremental), or if ranges are otherwise missing
* **Review univariate (single-column) distributions of discrete (or categorical) variables**
  + Generate a list of assumptions and expectations about any distributions of interest (e.g., how many users should fall into a given group)
  + Review univariate (single-column) distributions for discrete variables by looking at both row and unique ID counts (users, subscriptions, orders, etc.)
  + If the discrete variable is numerical in nature, review the average and range (min, max) of the variable of interest
  + Review cardinality (how many distinct values appear under a given column)
  + Compare distributions, averages, and ranges to any known assumptions about the data (this may require involvement from the client or analysts who have prior experience working with the data)
* **Review averages and ranges of continuous variables**
  + Generate a list of assumptions and expectations about any continuous variables of interest (e.g., expected averages, how many users should fall into a given range of values, etc.)
  + Review averages and ranges to look for unexpected values
  + Review distributions if appropriate (and feasible)
    - In some cases, it may be useful to convert continuous variables to discrete variables by bucketing values into predefined ranges to make running or reviewing distributions easier
* **Identify grain (or possible grains if more than one)**
  + Identify the sets of columns that can be used to identify a unique row (focusing on dates, IDs, and discrete variables) using the following steps:
    - Ensure duplicate records have been removed
    - Obtain an initial (all-up) row count for the entire dataset
    - Identify a subset of columns that might constitute a potential grain
    - Group by or select distinct for the column subset and obtain a second row count
    - Compare the row counts
    - If the second row count is equal to the first row count, then the column subset constitutes a potential grain for the dataset
    - Iterate as needed
  + It is often helpful to first review the columns and look for logical groups
    - This is especially true for de-normalized datasets that may contain numerous columns with one-to-one relationships (this is often done in parallel environments to avoid having to join multiple datasets which can be expensive from a performance standpoint)
    - If column groups having a one-to-one relationship can be identified, you may be able to shorten your investigation into grain by only including one column from the group in your column subset, since the other columns in the group have a one-to-one relationship and do not need to be tested individually
    - Note that the larger the column group having a one-to-one relationship that can be identified, the more impactful this initial step will be
  + When dealing with multiple potential grains, the important thing is to identify a grain that you can work with and that will support the project requirements
  + Be aware that some datasets may have a *mixed* grain (metrics at the level of day and week in the same dataset, for example, either in the same row or in separate rows that have been stacked together)
    - Mixed datasets usually contain columns that cannot be added because doing so would lead to duplication or inflation of metrics, and must be treated with care
    - Note that it is a best practice to avoid mixed grain tables whenever possible, but these may sometimes be encountered
* **Review multivariate (multiple-column) distributions of discrete (or categorical) variables**
  + Review distributions for specific groups of discrete variables to answer questions *as needed* related to the proportion of the data that falls into a given category
  + Due to the many possible combinations that can result from looking at distributions across multiple columns, reviewing multivariate distributions should be an outcome-driven task focused on answering a specific question related to requirements
* **Assess latency and availability**
  + Track how long it takes for data to populate
    - If data files are overwritten during each refresh, use available timestamp or date columns to identify the interval between when an event or change occurs and when that event or change becomes reflected in the dataset
  + Check for gaps in date ranges to determine if there are availability issues
    - If historical data files are available, check run dates for gaps in coverage
    - If historical data files are not available, check available timestamp or date columns for gaps in the data (this can be done by reviewing distributions)
  + Compare these findings to latency and availability requirements and flag any issues
* **Check for curing issues**
  + Review available timestamp and date columns to identify potential curing issues (additional time required for data from a prior date to populate or update after the data for that date has already appeared)
    - One approach that can be used to check for curing issues is to look for records, users, or unique ID values corresponding to a prior date appearing in the data file for a more recent date, and then look back over past run dates to see if the data associated with the prior date changes in any way or remains constant after the data for the prior date first appeared
    - In aggregate or report level data, it may also be possible to identify curing issues by tracking what happens to metrics or dimensions for prior dates each time the data is refreshed—if metrics for prior dates continue to change or dimension values are added each time the data is refreshed (on a daily basis, for example), then there is likely some type of curing issue
  + If curing issues are identified, perform the following:
    - Look for the magnitude of the change to data for prior dates each time a refresh occurs (this can be done using historical data files or tracking what happens over the course of several days, assuming a daily refresh)
    - Identify whether there is a date interval after which no changes occur (2-3 days for example), excluding periodic restatements of past data (these should not be considered a curing issue, but taken into account as part of the restatement process)
    - Assess the impact on project requirements and risk to final deliverables
    - Identify whether there are any steps that can be taken to mitigate or resolve any curing issues that have been identified, or whether client stakeholders should be engaged to troubleshoot
* **Assess data quality and usability**
  + Review consistency of values stored in discrete columns
  + Review presence of nulls, missing data, and unexpected values
  + Review comprehensibility of table names, column names, and dimension values, and whether additional information or tables (such as lookup tables) are needed to make the data easier to use and understand—any such tables should not be created during profiling, but they can be flagged for consideration during design work
  + Review timestamp and date columns to ensure they meet expectations (e.g., dates contained in a date column match the date range of the data that is supposed to be contained in the dataset)
  + Review additivity for any metrics that need to be produced from the data (whether these metrics are already stored in the data or need to be derived from it)
    - Checking for additivity can be complicated, however, and it may help to compare aggregations produced using the dataset against aggregations obtained from a trusted set of reports or other source of truth (reports or aggregate data obtained through a third-party UI, for example)
      * If the numbers do not match (specifically, if the aggregations obtained from the dataset are greater than those obtained from the source of truth), then the columns being aggregated may not be additive
      * It should *not* be assumed, however, that reports or data obtained from a third-party UI are necessarily correct, and in some cases, this task may surface quality issues with existing tools or processes
    - If a given column is needed to support requirements and exhibits additivity issues, assess the impact of the additivity issue on aggregate level results—some variance (less than 3% for example) may be acceptable if the data is only being used for reporting purposes, but any such decisions need to discussed and agreed upon with the client
  + Check for rows containing column headers, rows that do not conform to the table schema, or any other anomalies that may impact data quality or interfere with usability
  + For any issues that are identified, perform the following:
    - Assess the impact on requirements and final deliverables
    - Document transformations and any additional processing needed to address data quality issues or meet usability standards, or identify issues that require further research to resolve
    - Surface issues to project and client stakeholders as appropriate
* **Check for conformity**
  + If there is a need to join or combine multiple datasets, it may be necessary to check for conformity of dimensions including datatypes, ID formats (numerical versus hexadecimal, encrypted versus unencrypted, etc.), format and existence of discrete variable values, use of upper and lower case values in strings, etc.
  + If conformity issues exist, document those differences so that any necessary transformations can be specified during the design phase
* **Test linkages (i.e., matching) across multiple datasets**
  + If multiple datasets need to be integrated at a logical level (by joining the datasets on a particular column or set of columns, for example), assess whether datasets can be linked using available columns
  + Assess to what extent the data can support deterministic and probabilistic matching techniques if applicable
  + Use set analysis to assess coverage (overlap, match, etc.) between the datasets
    - Start with row counts and unique ID counts for each dataset individually
    - Once datasets are linked (or joined), assess the intersection and compare to the original counts
  + Document any issues with conformity that may need to be addressed before the datasets can be linked (in terms of grain, data types, value formats, etc.)
  + If gaps exist between the datasets, assess impact (percentage of records or unique IDs failing to match or appear in a complementary dataset, for example) as it pertains to requirements or final deliverables
* **Assess project-specific requirements**
  + Assess any additional project-specific requirements to determine whether they are supported by the data
  + Tasks may include additional analyses related to distributions (measures of variation, correlation, etc.), additional set analyses, investigations involving client engineering teams, and so on
  + Assess any risk pertaining to these requirements if gaps exist between the data that is needed and the data that is available

1. Requirements can be classified as business, functional, and technical. Business requirements consist of the high level goals, objectives, or capabilities the project is meant to achieve or advance—these can be thought of as the reason for the project. Functional requirements consist of the specific capabilities, use cases, and logical inputs and outputs enabled by the project. Technical requirements consist of the data and systems requirements that need to be met by the design and implementation of the project. [↑](#footnote-ref-1)
2. Some of the items listed here are a repetition of step 4 from the previous section *Steps for Completing a Data Profiling Project*. The reason is that some tasks performed during the initial phase of a data profiling project are needed to inform the data profiling plan, but these tasks can also be considered preliminary data profiling in that they impact our understanding of the data. Such tasks are therefore listed in both sections. [↑](#footnote-ref-2)