PAPER SAS3019-2019

Boop-Oop-A-Doop! It's Showtime with SAS® on Apache Hadoop!

Cecily Hoffritz, SAS Institute Inc., Denmark

ABSTRACT

Iconic Betty Boop in the 1930s cartoon Boop Oop A Doop tamed a lion. Nowadays, SAS® has tamed the elephant, the yellow Apache Hadoop one, and this paper shows you how it is done! Some Hadoop elephants live on land and others in clouds, and with the right SAS tools, you can sneak up really close to tame that data of yours! This paper is your easy-to-read SAS on Hadoop jungle survival guide that explains Hadoop tools for SAS®9 and SAS® Viya®, the main Hadoop landscapes, and good practices to access and turn your Hadoop data into top-notch quality information. It is showtime with SAS on Hadoop!

INTRODUCTION

This paper is for beginners! Taming Hadoop elephants can be a challenge, but if you use a SAS application that suits your user profile and business requirements, you should end up with a docile elephant and a smooth ride!

This is a tale of three SAS applications! It is about SAS® Data Loader for Hadoop, SAS® Data Integration Studio and SAS® Data Preparation, each with unique benefits, target groups and purpose when Hadoop data is in play. There is a natural flow of data between the three SAS applications, blending SAS 9 and SAS Viya into one SAS platform, and this paper shows you how to accomplish this.

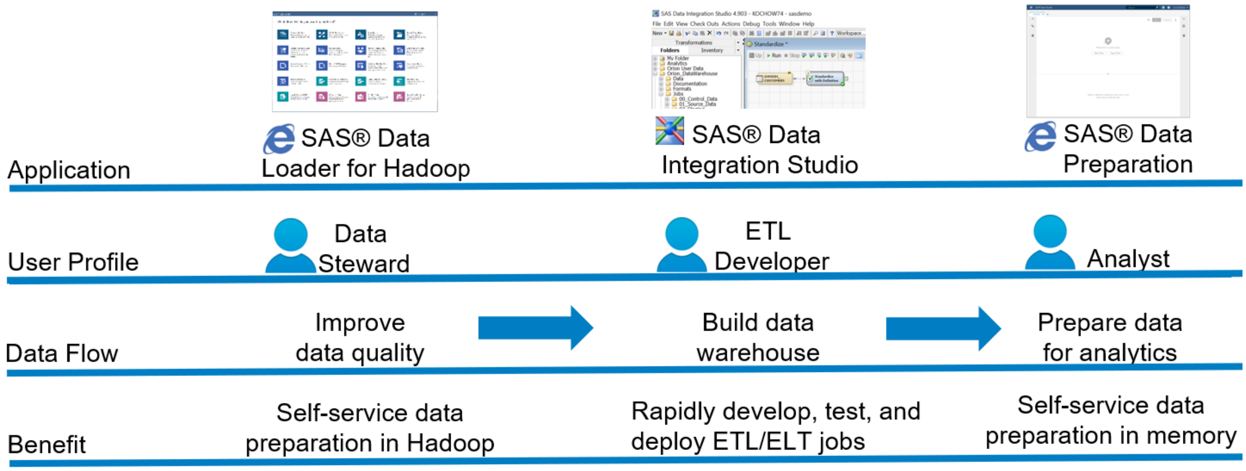

Application
SAS® Data Loader for Hadoop
SAS® Data Integration Studio
SAS® Data Preparation
User Profile
Data Steward
ETL Developer
Analyst
Data Flow
Improve data quality
Build data warehouse
Prepare data for analytics
Benefit
Self-service data preparation in Hadoop
Rapidly develop, test, and deploy ETL/ELT jobs
Self-service data preparation in memory

Figure 1. The tale of three SAS applications in a nutshell.

For each SAS application, there is a well-defined use case, and there is a natural flow between them. In short, it is all about removing data quality issues, calculating columns, combining data in Hive on Hadoop, and loading the results into memory for last-mile data preparation before analytics. The demo data contains customer contact information, where one of the tables contains existing Danish customers and the other table contains new Danish customers. Both tables have real data quality issues.

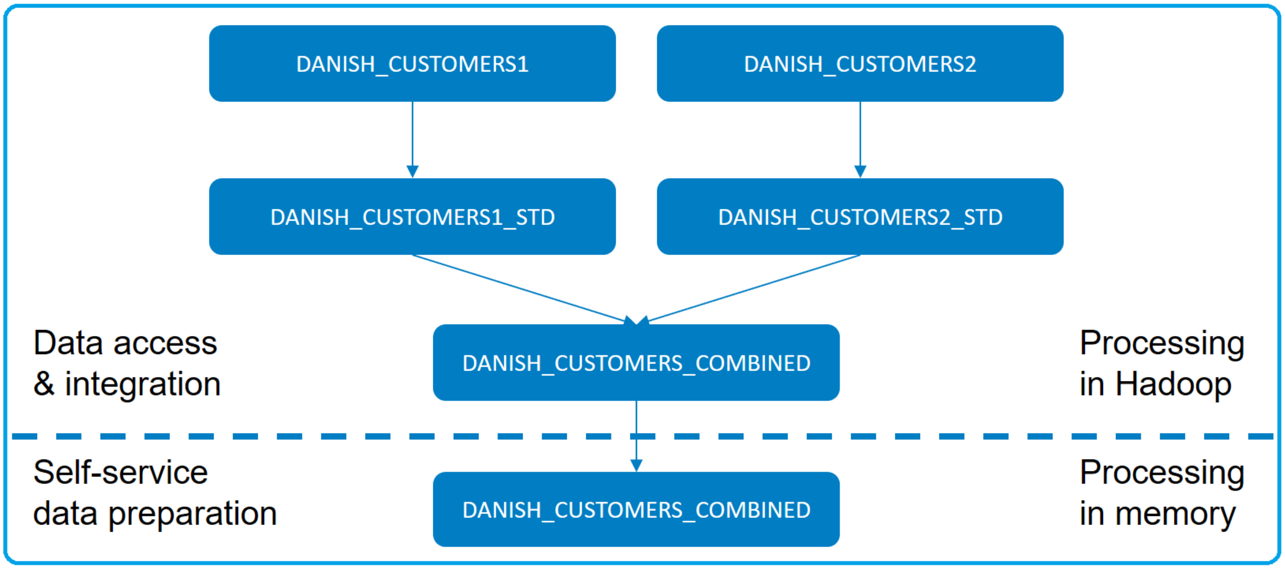


Figure 2. Overview of data flow.

The architecture for the use cases contains the SAS platform, with SAS 9 and SAS Viya providing a solid foundation for the end-to-end data and analytics life cycle, benefitting many types of users and business purposes.

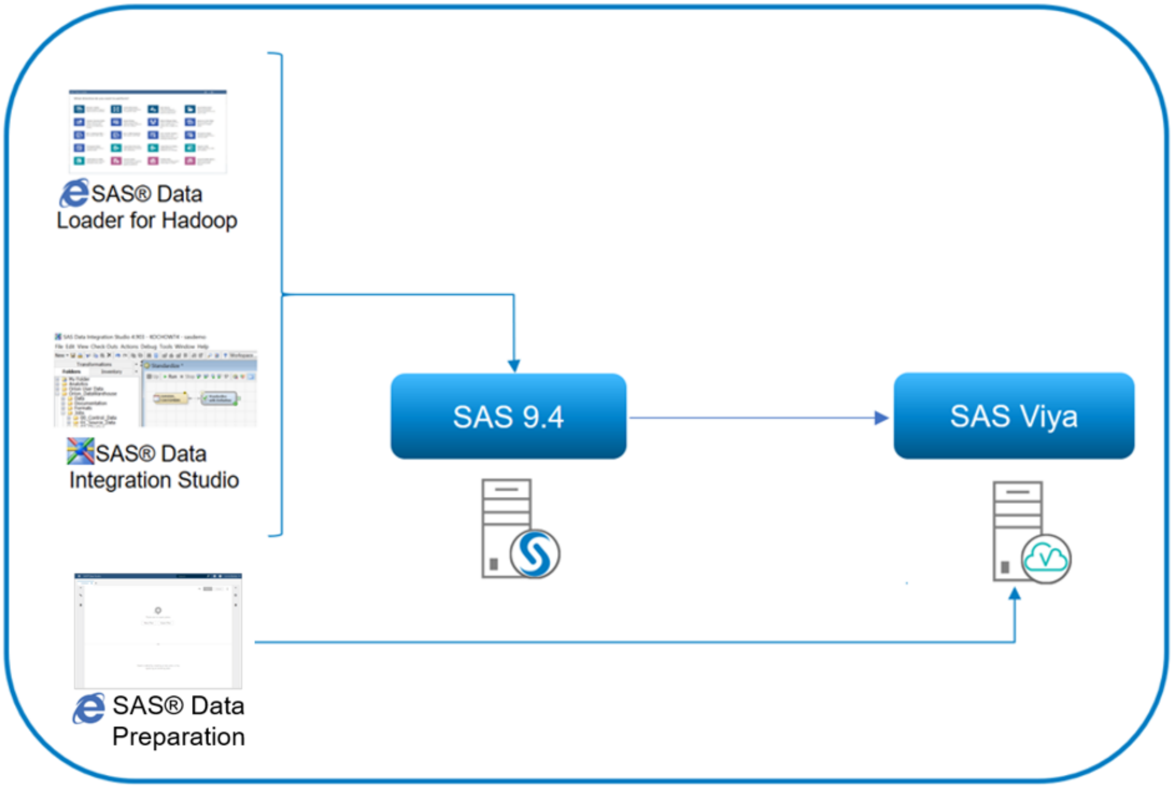


Figure 3. Architecture of the SAS platform with the three SAS applications in play.

SAS supports primarily these Hadoop distributions, but there is a variation of the support of SAS components for each distribution and SAS version that you can read more about in the documentation (links at the end): Cloudera, Hortonworks, MapR, Amazon EMR and Microsoft Azure HDInsight.

RIDING THE HADOOP ELEPHANT WITH SAS DATA LOADER FOR HADOOP

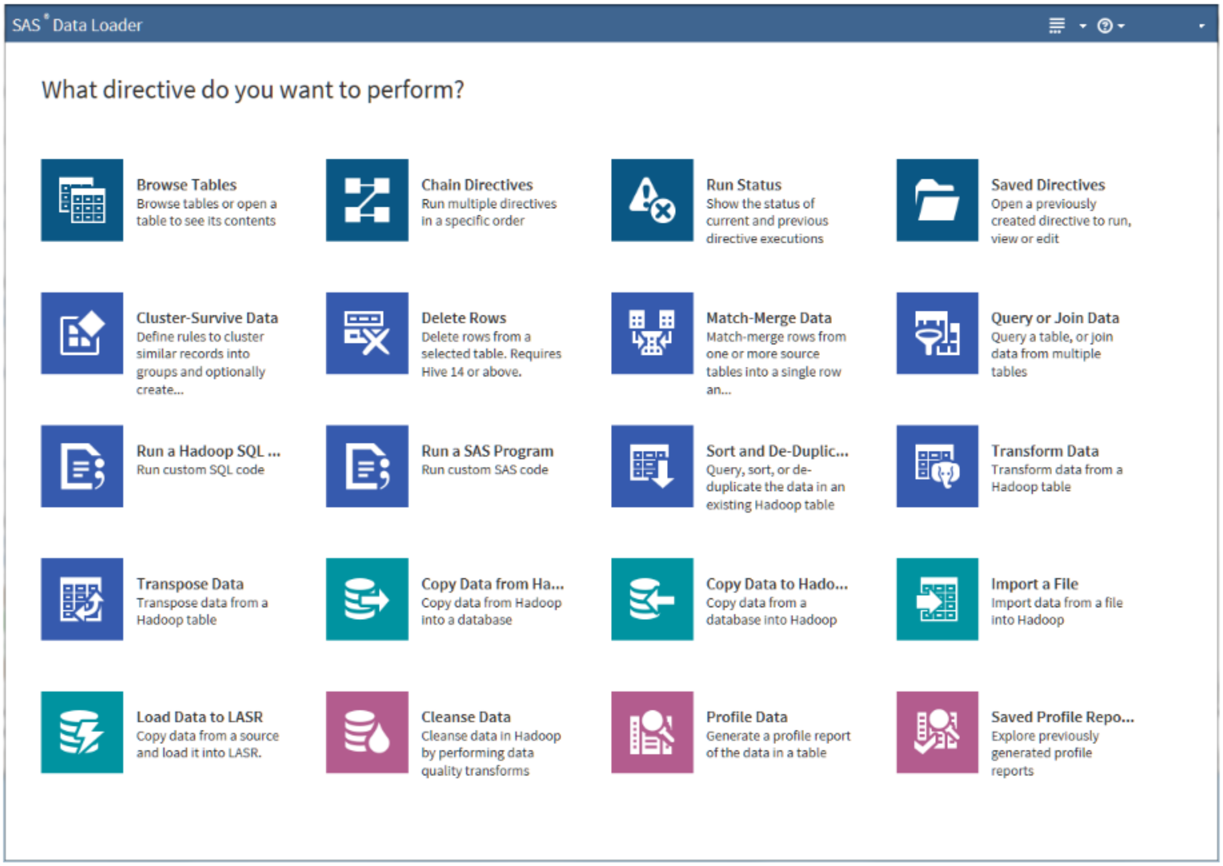
It is an uncomplicated and easy ride using SAS Data Loader for Hadoop, and this use case shows you how it is done!

USE CASE

Your data lake contains contact information for customers in many countries. As a data steward, you have noted that especially names, addresses, postal codes and cities for Danish customers need to be standardized. The original contact information resides in SAS tables on the Linux server, and it also is your job to ensure that they are copied to the lake prior to standardization.

SAS DATA LOADER FOR HADOOP – THE ULTIMATE SAS RIDE DIRECTLY IN THE LAKE

This use case focuses on SAS Data Loader for Hadoop, a self-service web application that helps you perform data quality and data manipulation tasks for Hive Hadoop data. These tasks are called *directives* that contain point-and-click menus, and some directives provide an array of transformations.



What directive do you want to perform?

Browse Tables
Browse tables or open a table to see its contents

Chain Directives
Run multiple directives in a specific order

Run Status
Show the status of current and previous directive executions

Saved Directives
Open a previously created directive to run, view or edit

Cluster-Survive Data
Define rules to cluster similar records into groups and optionally create...

Delete Rows
Delete rows from a selected table. Requires Hive 14 or above.

Match-Merge Data
Match-merge rows from one or more source tables into a single row an...

Query or Join Data
Query a table, or join data from multiple tables

Run a Hadoop SQL ...
Run custom SQL code

Run a SAS Program
Run custom SAS code

Sort and De-Duplic...
Query, sort, or de-duplicate the data in an existing Hadoop table

Transform Data
Transform data from a Hadoop table

Transpose Data
Transpose data from a Hadoop table

Copy Data from Ha...
Copy data from Hadoop into a database

Copy Data to Hado...
Copy data from a database into Hadoop

Import a File
Import data from a file into Hadoop

Load Data to LASR
Copy data from a source and load it into LASR.

Cleanse Data
Cleanse data in Hadoop by performing data quality transforms

Profile Data
Generate a profile report of the data in a table

Saved Profile Repo...
Explore previously generated profile reports

Display 1. The SAS Data Loader for Hadoop main page when you have logged on.

What I especially like about SAS Data Loader for Hadoop is that it is an application dedicated to executing your work as efficiently as possible in Hadoop. Once your data is in Hadoop, and if you stick to the directives where the underlying code is generated by the application, you can relax because your data is processed in the lake and not in SAS. Data movement is something you want to avoid as much as possible! If you use the directive where you create your own custom SQL code, you need more than minimal Hadoop knowledge to avoid your data being dredged out of the lake and into SAS for processing. You can read more about this in the next section on SAS Data Integration Studio.

YOUR ELEPHANT RIDING PROFILE

You are an analyst, business user, data steward or anyone else who needs to access and process data in Hadoop in an easy manner. Your knowledge of the inner workings of Hadoop, HiveQL and SAS can be very limited, but that doesn’t matter because the SAS Data Loader for Hadoop web application is a very user-friendly one.

YOUR ELEPHANT RIDING ACCESSORIES

SAS Data Loader for Hadoop supports in-database processing, which means that SAS processing is moved to the data source. For this to happen, SAS software is deployed to each node in the Hadoop cluster. These SAS components are deployed to the Hadoop nodes.

Component	Details
SAS Quality Knowledge Base for Contact Information.	This is important because it contains the standardization definition used in this use case. There are also definitions for gender analysis, personal data discovery, fuzzy matching and loads of other content. You can even customize the knowledge base by adding logic to quality assure car makes and parts, medical diagnoses, drug and telco products and many other areas, benefitting users in any SAS application that supports data quality.
SAS® Data Quality Accelerator for Hadoop.	This runs your data quality logic in Hadoop.
SAS® In-Database Code Accelerator for Hadoop.	This runs SAS Data Loader’s SAS programs in Hadoop.
SAS/ACCESS® Interface to Hadoop.	This allows you to connect to Hadoop.

Table 1. SAS Data Loader for Hadoop software components.

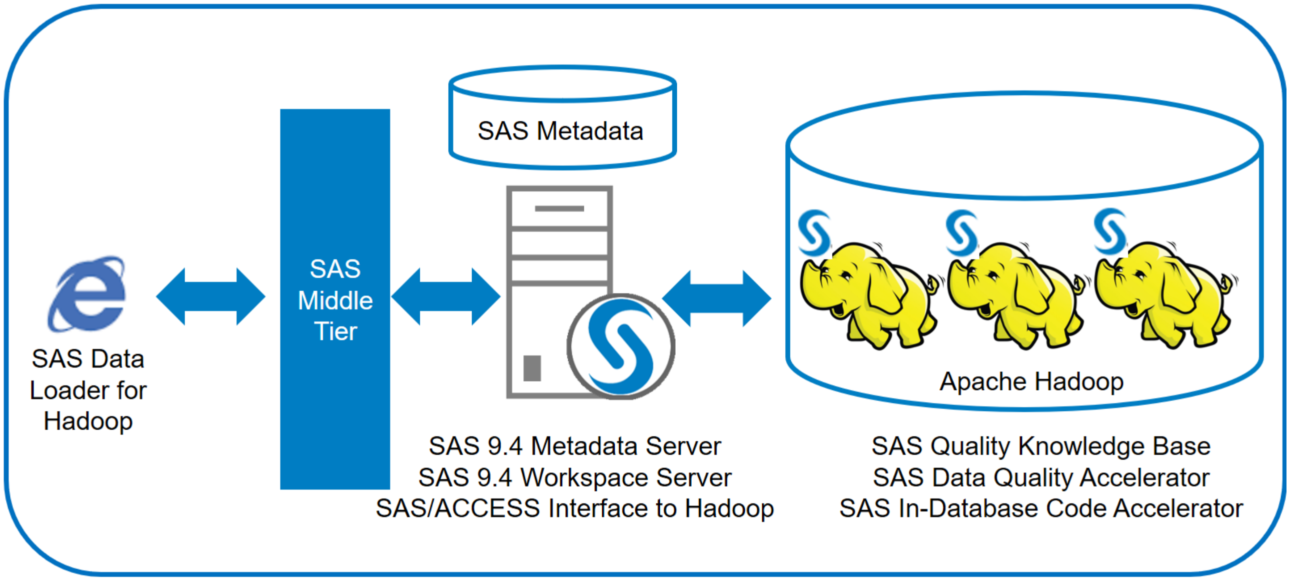
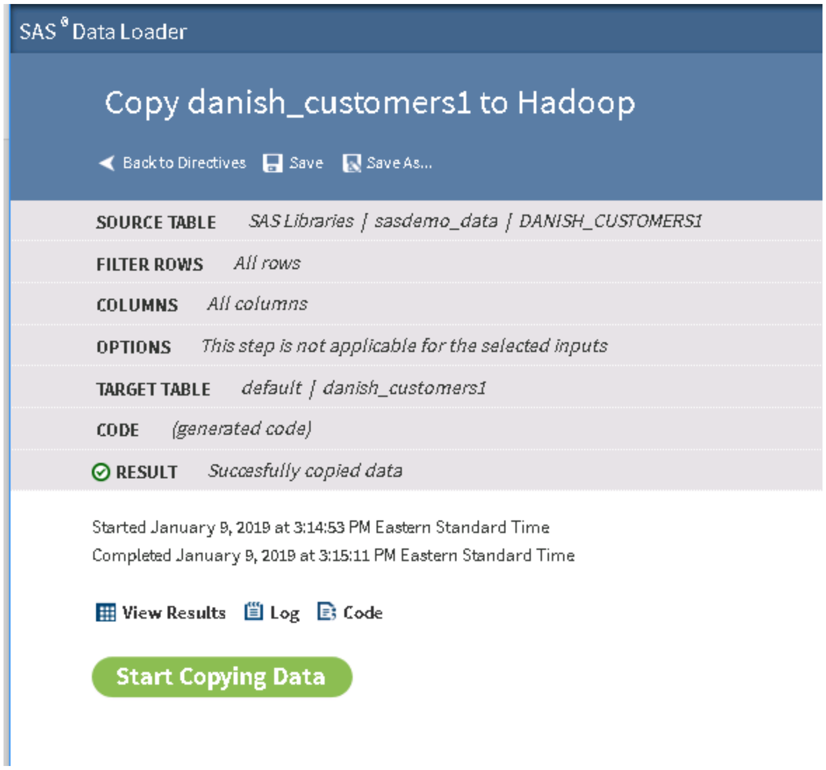

SAS Data Loader for Hadoop
SAS Middle Tier
SAS Metadata
SAS 9.4 Metadata Server
SAS 9.4 Workspace Server
SAS/ACCESS Interface to Hadoop
Apache Hadoop
SAS Quality Knowledge Base
SAS Data Quality Accelerator
SAS In-Database Code Accelerator

Figure 4. Architecture for SAS Data Loader for Hadoop.

MASTERING THE HADOOP ELEPHANT

Here are the overall steps that I took to solve my use case:

1. I used the Copy Data to Hadoop directive to copy to Hadoop the table DANISH_CUSTOMERS1 residing in the SAS library sasdemo_data on the Linux server. The destination for the target table is the default Hive schema that I have access to. I saved the directive so that I can rerun it when necessary.



Copy danish_customers1 to Hadoop

SOURCE TABLE SAS Libraries | sasdemo_data | DANISH_CUSTOMERS1

FILTER ROWS All rows

COLUMNS All columns

OPTIONS This step is not applicable for the selected inputs

TARGET TABLE default | danish_customers1

CODE (generated code)

RESULT Succesfully copied data

Started January 9, 2019 at 3:14:53 PM Eastern Standard Time
Completed January 9, 2019 at 3:15:11 PM Eastern Standard Time

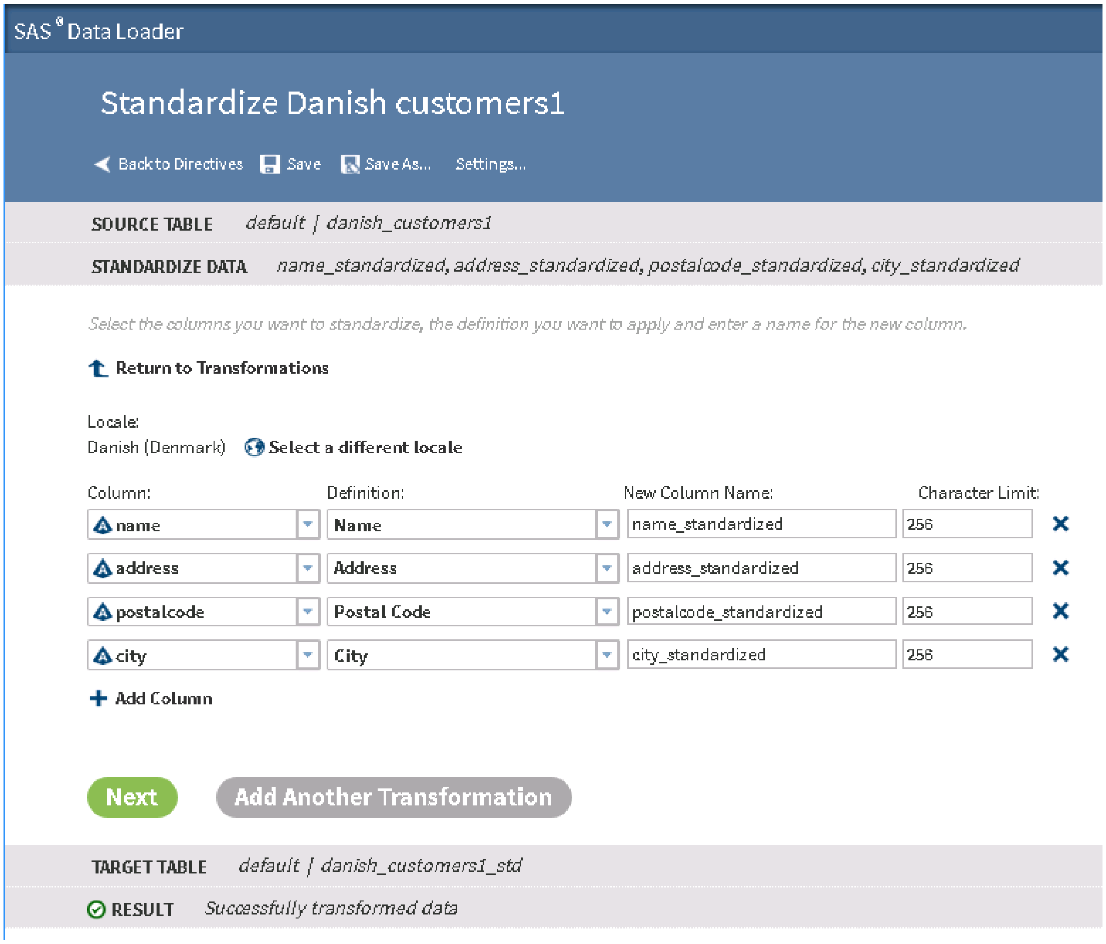
Display 2. Summary after copying data to Hadoop.

2. I followed the same process for the DANISH_CUSTOMERS2 table.

By the way, your Hadoop administrator is the one who provides you with the appropriate authorization to work with Hive schemas in your organization and your SAS administrator sets up libraries to your SAS tables and other data sources. You set up your own preferences for your run-time environment using Apache Spark or MapReduce. In my environment, Apache Spark is the preferred run-time environment because in-memory processing is usually much faster.

3. I used the Cleanse Data directive and the Standardize transformation to cleanse names, addresses, postal codes and cities for each Danish customer table. Because the column values are written in Danish, I used the Danish (Denmark) locale when standardizing. If the values had been written in American English, I would have used the US locale. For Danish postal codes, standardization means turning all deviating occurrences that are not just 4 digits into a 4-digit postal code (standard for Denmark). For example, a value, DK-7000, is transformed into 7000.

   I would also modify the standard length of new columns (for example, ensuring that the new postal code column was defined with a length of 4 characters and not 255). Modifying column attributes is beyond the scope here.


SAS® Data Loader
Back to Directives
Save
Save As...
Settings...
Return to Transformations
Select a different locale
Add Column
Next
Add Another Transformation

Standardize Danish customers1

SOURCE TABLE *default | danish_customers1*

STANDARDIZE DATA *name_standardized, address_standardized, postalcode_standardized, city_standardized*

Select the columns you want to standardize, the definition you want to apply and enter a name for the new column.

Locale:
Danish (Denmark)

Column:	Definition:	New Column Name:	Character Limit:
name	Name	name_standardized	256
address	Address	address_standardized	256
postalcode	Postal Code	postalcode_standardized	256
city	City	city_standardized	256

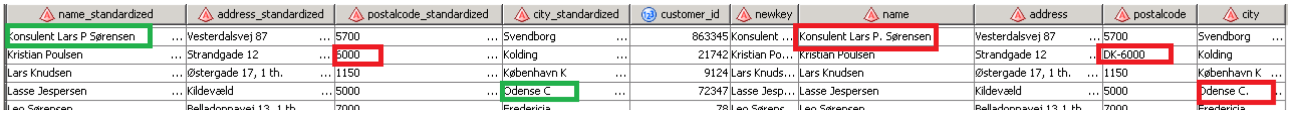
TARGET TABLE *default | danish_customers1_std*

RESULT *Successfully transformed data*

Display 3. Selections made in the Standardize transformation.

RESULTS

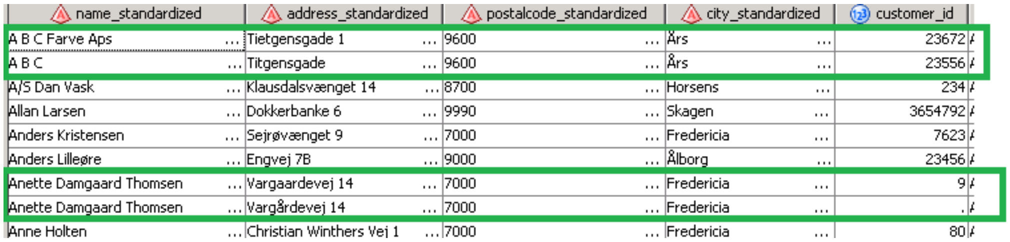
Notice in the table below the results after standardization. For each row, the value before standardization is boxed in red while the value after standardization is boxed in green. For example, in the first visible row, the dot after P in the name has been removed.



name_standardized	address_standardized	postalcode_standardized	city_standardized	customer_id	newkey	name	address	postalcode	city
Konsulent Lars P Sørensen	Vesterdalsvej 87	5700	Svendborg	863345	Konsulent ...	Konsulent Lars P. Sørensen	Vesterdalsvej 87	5700	Svendborg
Kristian Poulsen	Strandgade 12	6000	Kolding	21742	Kristian Po...	Kristian Poulsen	Strandgade 12	DK-6000	Kolding
Lars Knudsen	Østergade 17, 1 th.	1150	København K	9124	Lars Knuds...	Lars Knudsen	Østergade 17, 1 th.	1150	København K
Lasse Jespersen	Kildevæld	5000	Odense C	72347	Lasse Jesp...	Lasse Jespersen	Kildevæld	5000	Odense C.
Leo Sørensen	Belladonnavej 13, 1 th	7000	Fredericia	78	Leo Sørens	Leo Sørensen	Belladonnavej 13 1 th	7000	Fredericia

Output 1. Output showing before and after standardization.

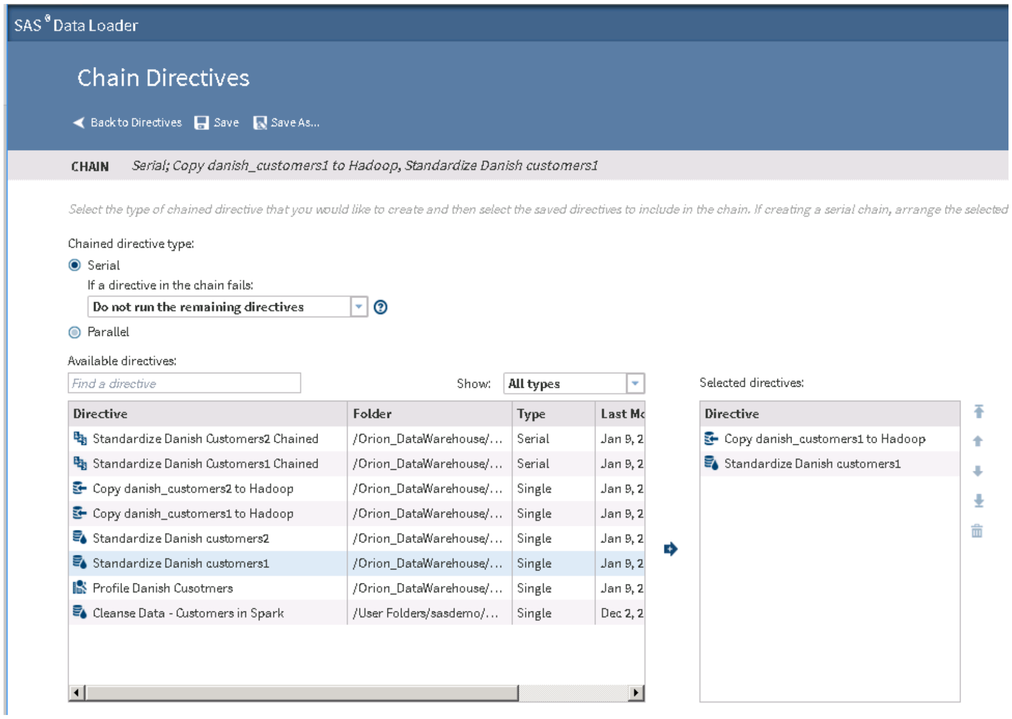
I also noticed in my data that I had other data quality issues. There appears to be more than one record for certain persons or organizations, and names of persons and organizations are jumbled together. To solve these issues, I would need to do fuzzy matching of values, set up rules to cluster records and then set up another set of survival rules to create the golden record. Alongside this, I would split the name column into two, with one containing individuals and the other containing organizations. It is of course possible for you to do all of this and more in SAS Data Loader for Hadoop! These tasks are beyond the scope of this paper.



name_standardized	address_standardized	postalcode_standardized	city_standardized	customer_id
A B C Farve Aps	Tietgensgade 1	9600	Års	23672
A B C	Titgensgade	9600	Års	23556
A/S Dan Vask	Klausdalsvænget 14	8700	Horsens	234
Allan Larsen	Dokkerbanke 6	9990	Skagen	3654792
Anders Kristensen	Sejrøvænget 9	7000	Fredericia	7623
Anders Lilleøre	Engvej 7B	9000	Ålborg	23456
Anette Damgaard Thomsen	Vargaardevej 14	7000	Fredericia	9
Anette Damgaard Thomsen	Vargårdevej 14	7000	Fredericia	.
Anne Holten	Christian Winthers Vej 1	7000	Fredericia	80

Output 2. Records that appear to be the same person or organization.

Because my directives are also going to be run by ETL developers working in SAS Data Integration Studio (see next section), I used the Chain Directive to chain my directives into logical flows. I chained the saved directives that involved DANISH_CUSTOMERS1 in one chain and the saved directives for DANISH_CUSTOMERS2 in another chain. It is also possible to decide to chain directives to run in sequence or in parallel.



CHAIN Serial; Copy danish_customers1 to Hadoop, Standardize Danish customers1

Select the type of chained directive that you would like to create and then select the saved directives to include in the chain. If creating a serial chain, arrange the selected

Chained directive type:

● Serial

If a directive in the chain fails: Do not run the remaining directives

○ Parallel

Available directives:

Show: All types

Directive	Folder	Type	Last Mc
Standardize Danish Customers2 Chained	/Orion_DataWarehouse/...	Serial	Jan 9, 2
Standardize Danish Customers1 Chained	/Orion_DataWarehouse/...	Serial	Jan 9, 2
Copy danish_customers2 to Hadoop	/Orion_DataWarehouse/...	Single	Jan 9, 2
Copy danish_customers1 to Hadoop	/Orion_DataWarehouse/...	Single	Jan 9, 2
Standardize Danish customers2	/Orion_DataWarehouse/...	Single	Jan 9, 2
Standardize Danish customers1	/Orion_DataWarehouse/...	Single	Jan 9, 2
Profile Danish Cusotmers	/Orion_DataWarehouse/...	Single	Jan 9, 2
Cleanse Data - Customers in Spark	/User Folders/sasdemo/...	Single	Dec 2, 2

Selected directives:

Directive
Copy danish_customers1 to Hadoop
Standardize Danish customers1

Display 4. Overview of directives chained together.

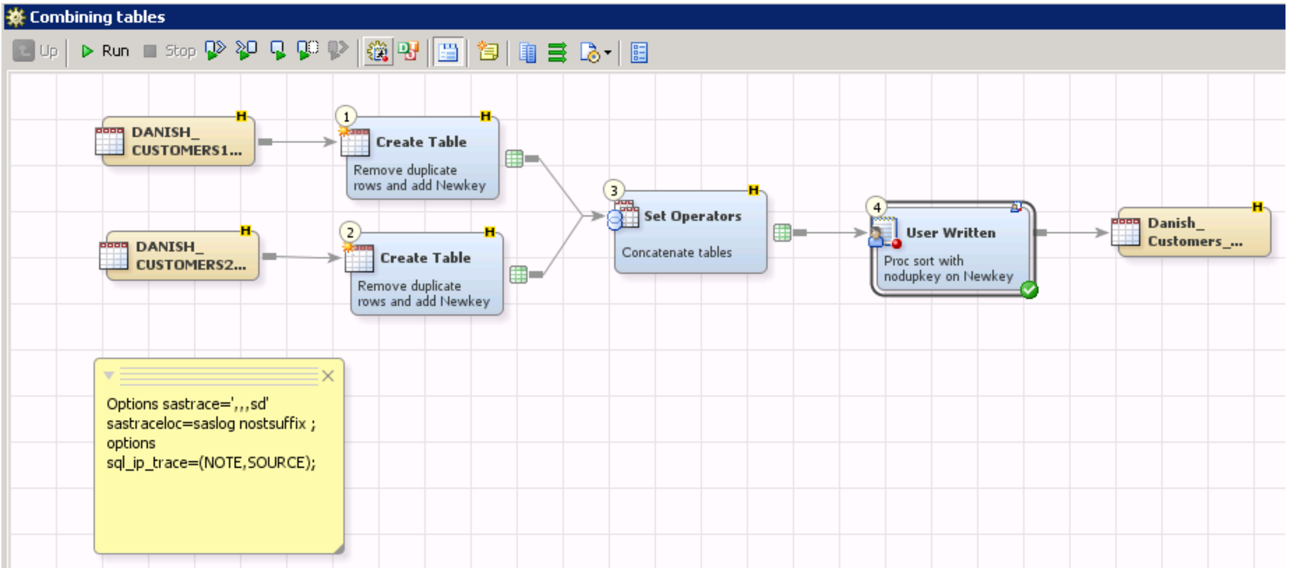
TAMING THE HADOOP ELEPHANT WITH SAS DATA INTEGRATION STUDIO

Understanding the elephant and its blusterous behavior helps you tame it when using SAS Data Integration Studio. This use case shows you how it is done!

USE CASE

You want to integrate the previously created SAS Data Loader for Hadoop chained directives into a SAS Data Integration Studio job to make it a part of your ETL flow.

You also want to use the SQL transformations in SAS Data Integration Studio to access and manage Hive Hadoop tables, working very consciously to ensure that you push as much processing as possible down to Hadoop for maximum performance. Your tasks for this use case involve removing duplicate rows and keys, combining tables and calculating columns using functions. In fact, you want to build a SAS Data Integration Studio job that resembles the one here:



```
Options sastrace=',,,sd'
sastraceloc=saslog nostsuffix ;
options
sql_ip_trace=(NOTE,SOURCE);
```

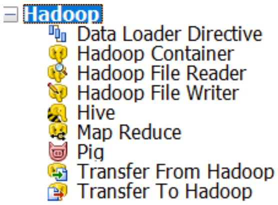
Display 5. SAS Data Integration Studio job solution.

SAS DATA INTEGRATION STUDIO IS FOR ELEPHANT HABITATS AND OTHER HABITATS

SAS Data Integration Studio is a sophisticated, professional visual design tool that you use to build, implement and manage data integration processes regardless of data sources, applications, or platforms.

In these privacy focused times, SAS Data Integration Studio is the perfect choice to build warehouses and marts due to its 100% metadata awareness, allowing high trackability for data flowing in and out of Hadoop and other data sources and providing lineage for the complete data life cycle.

SAS Data Integration Studio contains pre-built Hadoop transformations where you can pass through native HiveQL and write Pig, Map Reduce and HDFS commands, all of which require that you know the syntax. It also contains other transformations based on the SAS DATA step, SQL and other SAS procedures, the syntax of which you are most likely familiar with.



- Hadoop
  - Data Loader Directive
  - Hadoop Container
  - Hadoop File Reader
  - Hadoop File Writer
  - Hive
  - Map Reduce
  - Pig
  - Transfer From Hadoop
  - Transfer To Hadoop

Display 6. Overview of Hadoop transformations.

YOUR ELEPHANT TAMER PROFILE

Your job is to build data warehouses and data marts for reporting and analytics in batch. You are an ETL developer, data engineer or in a similar job, and you are very familiar with databases and SQL processing. You also have sufficient Hadoop knowledge to avoid getting snagged in SAS Data Integration Studio jobs involving Hadoop data.

YOUR ELEPHANT TAMING ACCESSORIES

To solve this use case without the first part involving SAS Data Loader for Hadoop, your minimum SAS software package contains SAS Data Integration Server and SAS/ACCESS® Interface to Hadoop.

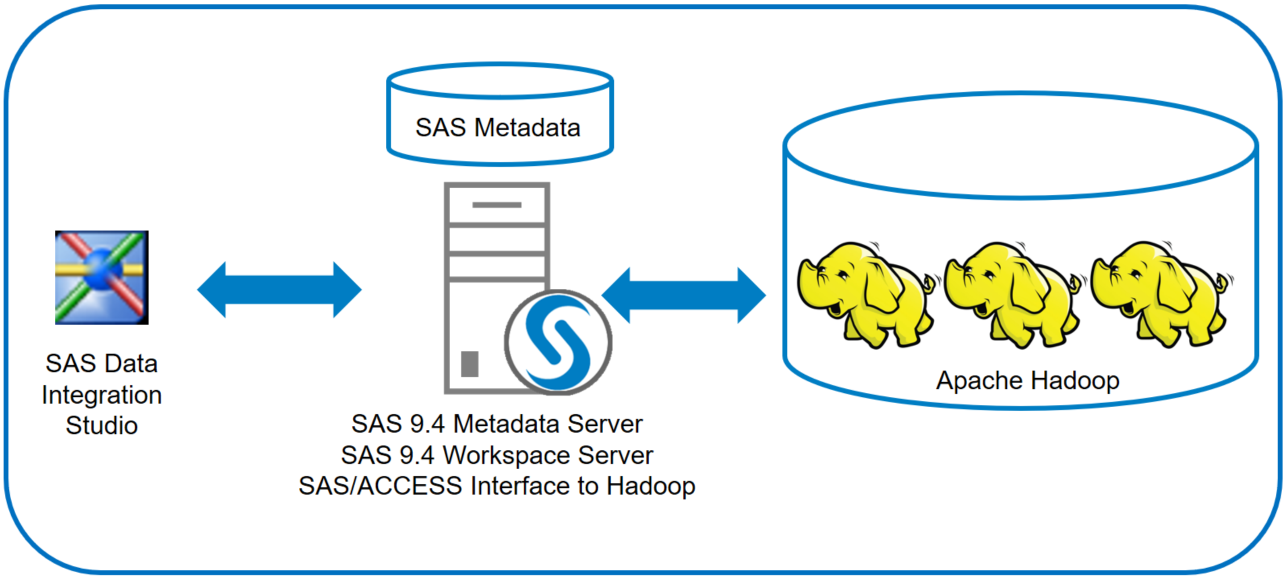


Figure 5. Depicting a SAS Data Integration architecture.

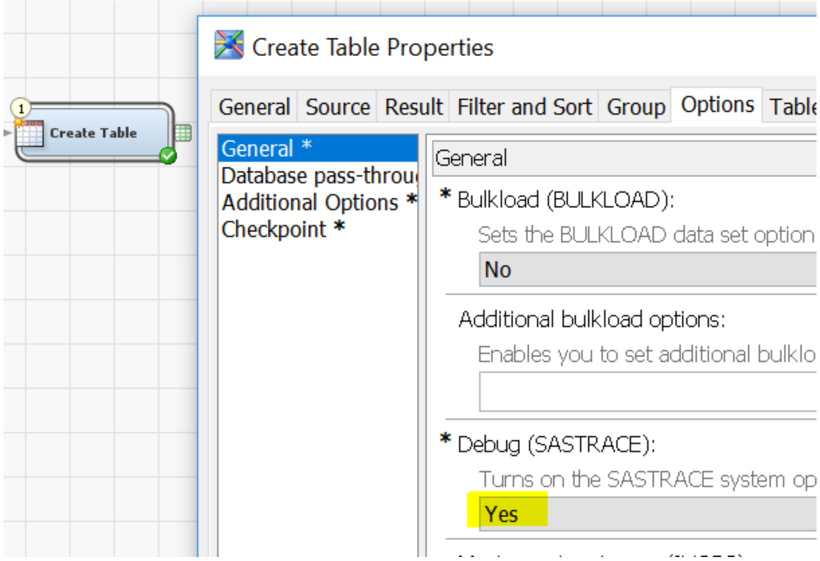
By the way, to get started, you are dependent on your SAS administrator configuring the SAS server so that there is a connection to the Hadoop server. You are also dependent on your Hadoop administrator providing you with the means to authenticate to the Hadoop server and with the proper authorizations to the Hadoop data locations, meeting your organization's stringent security requirements.

MASTERING THE HADOOP ELEPHANT

Mastering the Hadoop elephant involves certain sneaky tricks to track elephants and to avoid unintentional and interim lake draining.

A sneaky trick to know exactly how to behave in the vicinity of elephants

Do you know if you are staying put in the lake, frolicking with Hadoop elephants, or if you are unintentionally spending loads of effort dredging content out of the lake? To figure this out, you can add tracing options to your jobs when developing them. Some transformations include options for tracing while, for others, you can turn on SASTRACE in an OPTIONS statement in a job's pre-code. Once tracing is turned on, you consult the log to determine whether statements have been sent to Hadoop.

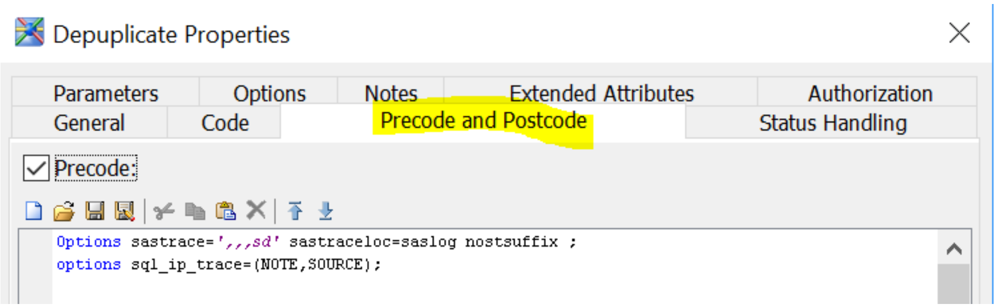


* Bulkload (BULKLOAD):
Sets the BULKLOAD data set option
No

Additional bulkload options:
Enables you to set additional bulklo

* Debug (SASTRACE):
Turns on the SASTRACE system op
Yes

Display 7. Turning on SASTRACE in a transformation.



☑ Precode:

```
Options sastrace=',,,sd' sastraceloc=saslog nostsuffix ;
options sql_ip_trace=(NOTE,SOURCE);
```

Display 8. Turning on SASTRACE in an OPTIONS statement in a job's pre-code.

Here is an explanation of arguments you can add to SASTRACE:

- S sets timers (to capture and display the amount of time spent on database activities)
- D is the database trace.
- SASTRACELOC sends the trace to a log.
- NOSTSUFFIX makes the log easier to read.
- FULLSTIMER collects performance statistics on each SAS step.

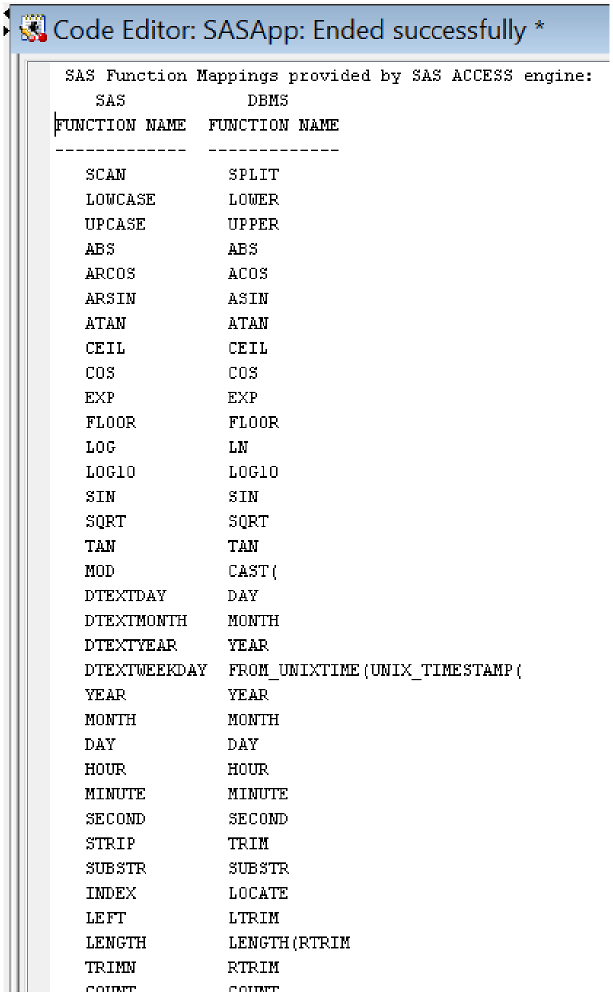
A sneaky trick to know about functions to avoid lake draining

The trick is to avoid non-mapped SAS functions when attempting SQL implicit pass-through because that will literally drag all the rows out of the Hadoop data lake and into SAS for processing.

To exercise caution and get an understanding of what is allowed between SAS and Hadoop, you need to map SAS functions with equivalent Hive functions. The way to get the full list is to add the options SQL_FUNCTIONS_COPY=SASLOG and SQL_FUNCTIONS=ALL to a Hadoop LIBNAME. I ran the following LIBNAME statement in the Code Editor, and the mapping results appeared in the log.

```
libname DL4HDQ Hadoop port=10000 server="sasdata1.demo.sas.com"
schema=default authdomain="DefaultAuth" SQL_FUNCTIONS_COPY=SASLOG and
SQL_FUNCTIONS=ALL;
```

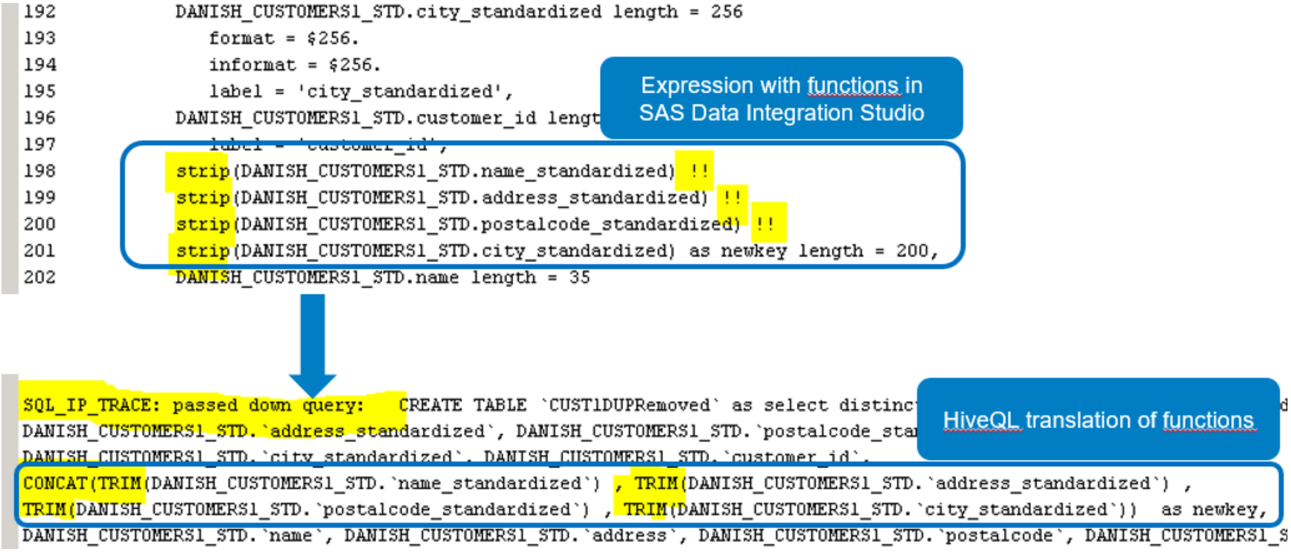
Output 3. Library definition to access Apache Hadoop Hive data.



```
 SAS Function Mappings provided by SAS ACCESS engine:
    SAS            DBMS
FUNCTION NAME  FUNCTION NAME
-------------  -------------
   SCAN           SPLIT
   LOWCASE        LOWER
   UPCASE         UPPER
   ABS            ABS
   ARCOS          ACOS
   ARSIN          ASIN
   ATAN           ATAN
   CEIL           CEIL
   COS            COS
   EXP            EXP
   FLOOR          FLOOR
   LOG            LN
   LOG10          LOG10
   SIN            SIN
   SQRT           SQRT
   TAN            TAN
   MOD            CAST(
   DTEXTDAY       DAY
   DTEXTMONTH     MONTH
   DTEXTYEAR      YEAR
   DTEXTWEEKDAY   FROM_UNIXTIME(UNIX_TIMESTAMP(
   YEAR           YEAR
   MONTH          MONTH
   DAY            DAY
   HOUR           HOUR
   MINUTE         MINUTE
   SECOND         SECOND
   STRIP          TRIM
   SUBSTR         SUBSTR
   INDEX          LOCATE
   LEFT           LTRIM
   LENGTH         LENGTH(RTRIM
   TRIMN          RTRIM
   COUNT          COUNT
```

Display 9. Log showing partial listing of mapped functions.

If you use appropriate functions when defining expressions, the query is passed down, and a behind-the-scenes translation of the functions in HiveQL takes place.



```
           DANISH_CUSTOMERS1_STD.city_standardized length = 256
              format = $256.
              informat = $256.
              label = 'city_standardized',
           DANISH_CUSTOMERS1_STD.customer_id lengt
              label = 'customer_id',
           strip(DANISH_CUSTOMERS1_STD.name_standardized) !!
           strip(DANISH_CUSTOMERS1_STD.address_standardized) !!
           strip(DANISH_CUSTOMERS1_STD.postalcode_standardized) !!
           strip(DANISH_CUSTOMERS1_STD.city_standardized) as newkey length = 200,
           DANISH_CUSTOMERS1_STD.name length = 35
```

Expression with functions in SAS Data Integration Studio

```
SQL_IP_TRACE: passed down query:   CREATE TABLE `CUST1DUPRemoved` as select distinc
DANISH_CUSTOMERS1_STD.`address_standardized`, DANISH_CUSTOMERS1_STD.`postalcode_sta
DANISH_CUSTOMERS1_STD.`city_standardized`, DANISH_CUSTOMERS1_STD.`customer_id`,
CONCAT(TRIM(DANISH_CUSTOMERS1_STD.`name_standardized`) , TRIM(DANISH_CUSTOMERS1_STD.`address_standardized`) ,
TRIM(DANISH_CUSTOMERS1_STD.`postalcode_standardized`) , TRIM(DANISH_CUSTOMERS1_STD.`city_standardized`))  as newkey,
DANISH_CUSTOMERS1_STD.`name`, DANISH_CUSTOMERS1_STD.`address`, DANISH_CUSTOMERS1_STD.`postalcode`, DANISH_CUSTOMERS1_S
```

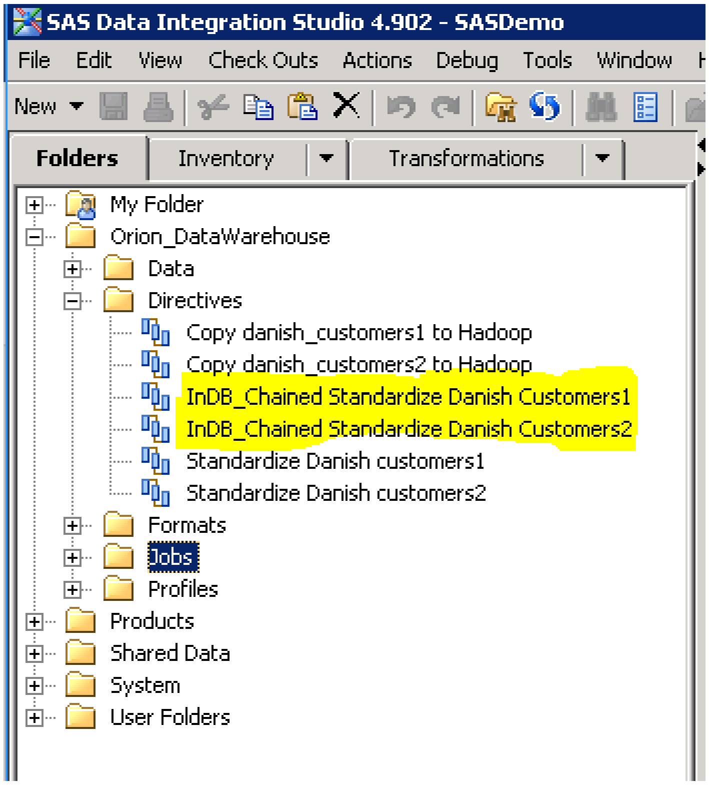
HiveQL translation of functions

Output 4. Log output showing that the query is passed down.

BUILDING JOBS

Job 1 – Creating a job with SAS Data Loader for Hadoop saved directives

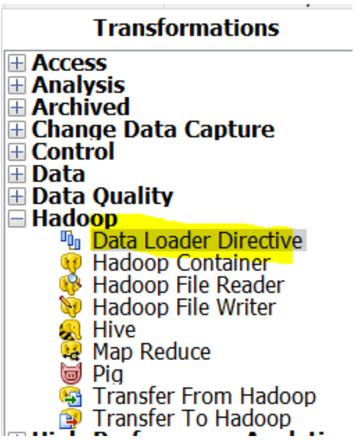
Because the SAS® Metadata Server is common to both SAS Data Loader for Hadoop and SAS Data Integration Studio in my environment, the metadata folder with the saved directives is visible in SAS Data Integration Studio.


SAS Data Integration Studio 4.902 - SASDemo
File
Edit
View
Check Outs
Actions
Debug
Tools
Window
New
Folders
Inventory
Transformations
My Folder
Orion_DataWarehouse
Data
Directives
Copy danish_customers1 to Hadoop
Copy danish_customers2 to Hadoop
InDB_Chained Standardize Danish Customers1
InDB_Chained Standardize Danish Customers2
Standardize Danish customers1
Standardize Danish customers2
Formats
Jobs
Profiles
Products
Shared Data
System
User Folders

Display 10. Saved directives in metadata visible in SAS Data Integration Studio.

These are the overall steps that I took to create the job:

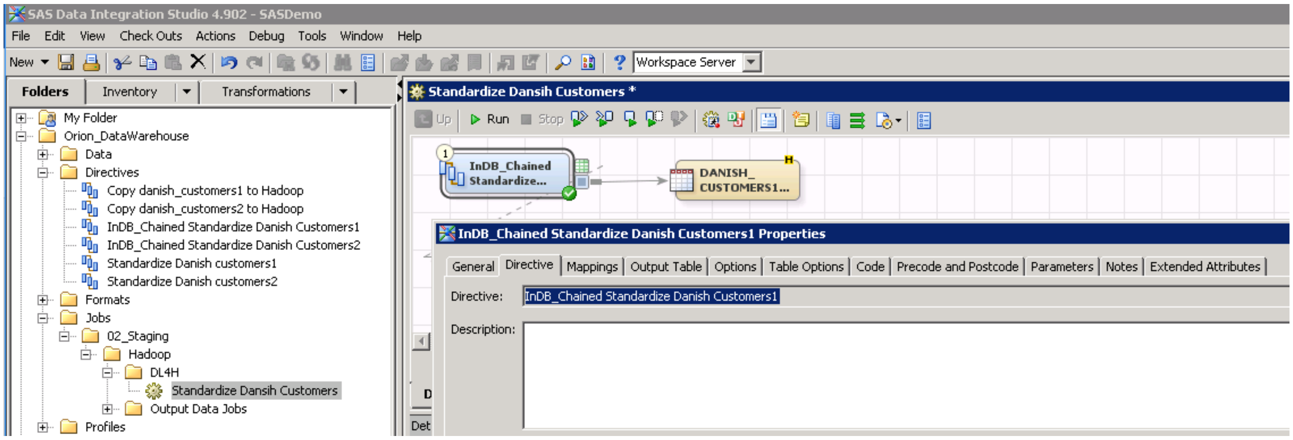
1. I created a new job named Standardize Danish Customers and added the Data Loader Directive.



Display 11. Overview of Hadoop transformations in SAS Data Integration Studio.

2. In the Directives tab in Properties of the transformation, I added the InDB_Chained Standardize Danish Customers 1 directive. In the Options tab, you can specify whether you want to wait for the saved directive to finish before processing the rest of the job.

3. I also added a metadata registered table to visualize the table output of the Directives transformation.


SAS Data Integration Studio 4.902 - SASDemo
File Edit View Check Outs Actions Debug Tools Window Help
New
Workspace Server
Folders
Inventory
Transformations
My Folder
Orion_DataWarehouse
Data
Directives
Copy danish_customers1 to Hadoop
Copy danish_customers2 to Hadoop
InDB_Chained Standardize Danish Customers1
InDB_Chained Standardize Danish Customers2
Standardize Danish customers1
Standardize Danish customers2
Formats
Jobs
02_Staging
Hadoop
DL4H
Standardize Dansih Customers
Output Data Jobs
Profiles
Standardize Dansih Customers *
Up
Run
Stop
1
InDB_Chained Standardize...
DANISH_ CUSTOMERS1...
InDB_Chained Standardize Danish Customers1 Properties
General
Directive
Mappings
Output Table
Options
Table Options
Code
Precode and Postcode
Parameters
Notes
Extended Attributes
Det

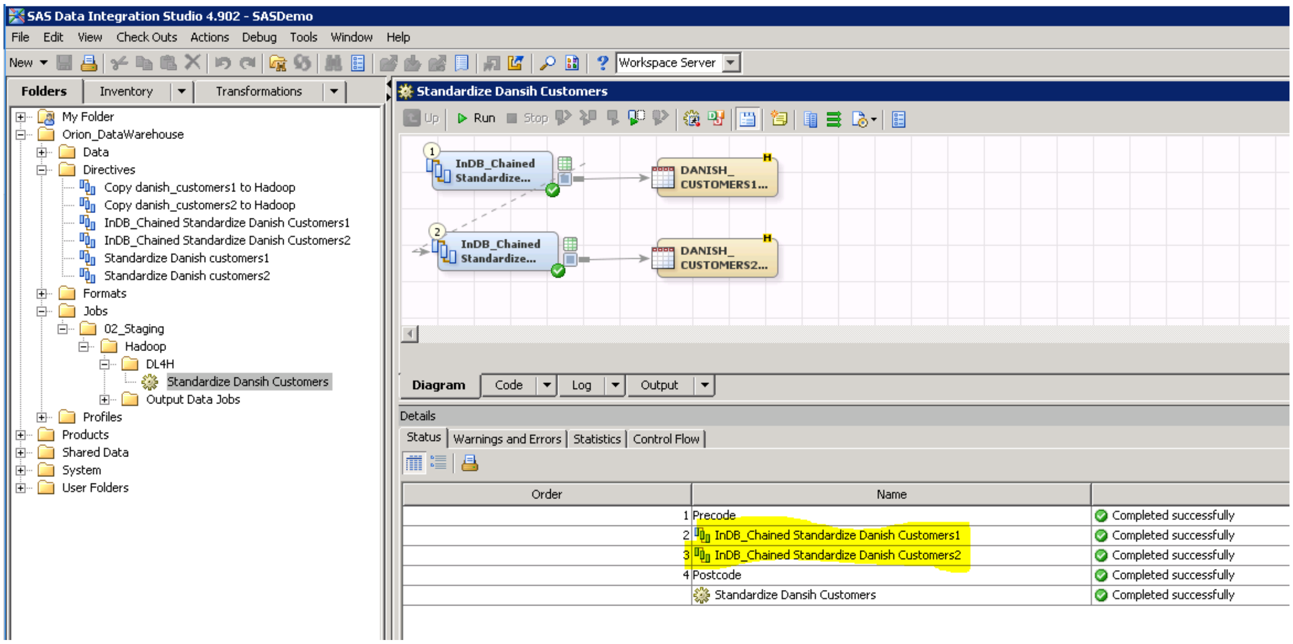
Directive: InDB_Chained Standardize Danish Customers1

Description:

Display 12. Job with saved directive and its properties.

4. I repeated the process for the InDB_Chained Standardize Danish Customers 2 directive so that the job now contains two Data Loader Directives transformations with the relevant Properties specified.

5. I then modified the Control Flow of the job and that resulted in the hyphened arrow between the two directives. As you can see here, I ran the job and it completed successfully.


SAS Data Integration Studio 4.902 - SASDemo
File Edit View Check Outs Actions Debug Tools Window Help
New
Workspace Server
Folders
Inventory
Transformations
My Folder
Orion_DataWarehouse
Data
Directives
Copy danish_customers1 to Hadoop
Copy danish_customers2 to Hadoop
InDB_Chained Standardize Danish Customers1
InDB_Chained Standardize Danish Customers2
Standardize Danish customers1
Standardize Danish customers2
Formats
Jobs
02_Staging
Hadoop
DL4H
Standardize Dansih Customers
Output Data Jobs
Profiles
Products
Shared Data
System
User Folders
Standardize Dansih Customers
Up
Run
Stop
1
InDB_Chained Standardize...
DANISH_ CUSTOMERS1...
2
InDB_Chained Standardize...
DANISH_ CUSTOMERS2...
Diagram
Code
Log
Output
Details
Status
Warnings and Errors
Statistics
Control Flow

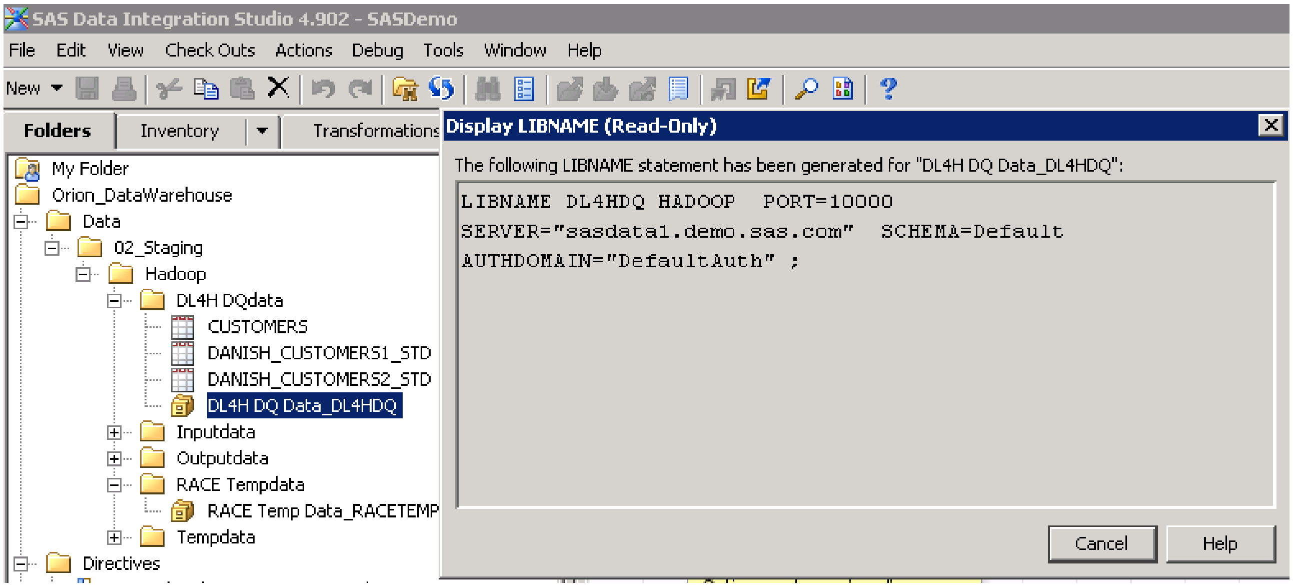
Order	Name	
1	Precode	✓ Completed successfully
2	InDB_Chained Standardize Danish Customers1	✓ Completed successfully
3	InDB_Chained Standardize Danish Customers2	✓ Completed successfully
4	Postcode	✓ Completed successfully
	Standardize Dansih Customers	✓ Completed successfully

Display 13. A job with saved SAS Data Loader for Hadoop saved directives that ran successfully.

Job 2 – Creating a job with SQL transformations and user-written transformation

The tasks involved removing duplicate rows and keys, combining Hive tables and calculating columns using functions. These are the overall steps I took to complete the job:

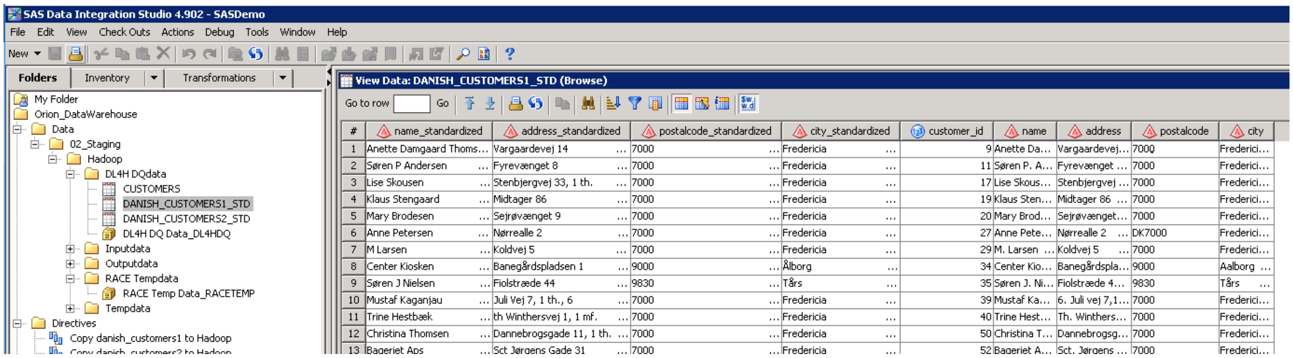
1. I created two Hadoop libraries. One of them is a temporary one where I used the library options DBMSTEMP=Yes and CONNECTION=Global. Depending on your setup and security, your Hadoop LIBNAME statements might contain more options than shown here. The online documentation provides information about this (see the links at the end of the paper).



The following LIBNAME statement has been generated for "DL4H DQ Data_DL4HDQ":

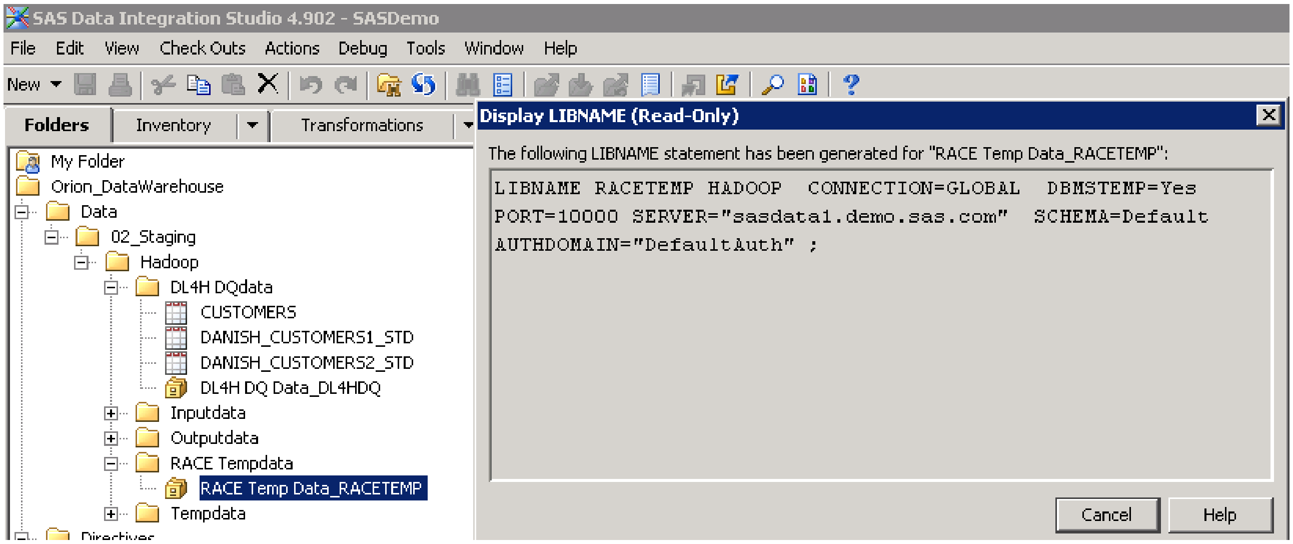
```
LIBNAME DL4HDQ HADOOP  PORT=10000
SERVER="sasdata1.demo.sas.com"  SCHEMA=Default
AUTHDOMAIN="DefaultAuth" ;
```

Display 14. Permanent Hadoop library definition.



#	name_standardized	address_standardized	postalcode_standardized	city_standardized	customer_id	name	address	postalcode	city
1	Anette Damgaard Thoms…	Vargaardevej 14	7000	Fredericia	9	Anette Da…	Vargaardevej…	7000	Frederici…
2	Søren P Andersen	Fyrevænget 8	7000	Fredericia	11	Søren P. A…	Fyrevænget …	7000	Frederici…
3	Lise Skousen	Stenbjergvej 33, 1 th.	7000	Fredericia	17	Lise Skous…	Stenbjergvej …	7000	Frederici…
4	Klaus Stengaard	Midtager 86	7000	Fredericia	19	Klaus Sten…	Midtager 86 …	7000	Frederici…
5	Mary Brodesen	Sejrøvænget 9	7000	Fredericia	20	Mary Brod…	Sejrøvænget…	7000	Frederici…
6	Anne Petersen	Nørrealle 2	7000	Fredericia	27	Anne Pete…	Nørrealle 2 …	DK7000	Frederici…
7	M Larsen	Koldvej 5	7000	Fredericia	29	M. Larsen …	Koldvej 5 …	7000	Frederici…
8	Center Kiosken	Banegårdspladsen 1	9000	Ålborg	34	Center Kio…	Banegårdspla…	9000	Aalborg …
9	Søren J Nielsen	Fiolstræde 44	9830	Tårs	35	Søren J. Ni…	Fiolstræde 4…	9830	Tårs …
10	Mustaf Kaganjau	Juli Vej 7, 1 th., 6	7000	Fredericia	39	Mustaf Ka…	6. Juli vej 7,1…	7000	Frederici…
11	Trine Hestbæk	th Winthersvej 1, 1 mf.	7000	Fredericia	40	Trine Hest…	Th. Winthers…	7000	Frederici…
12	Christina Thomsen	Dannebrogsgade 11, 1 th.	7000	Fredericia	50	Christina T…	Dannebrogsg…	7000	Frederici…
13	Bageriet Aps	Sct Jørgens Gade 31	7000	Fredericia	52	Bageriet A…	Sct. Jørgens …	7000	Frederici…

Output 5. Viewing a table in the Hadoop library.

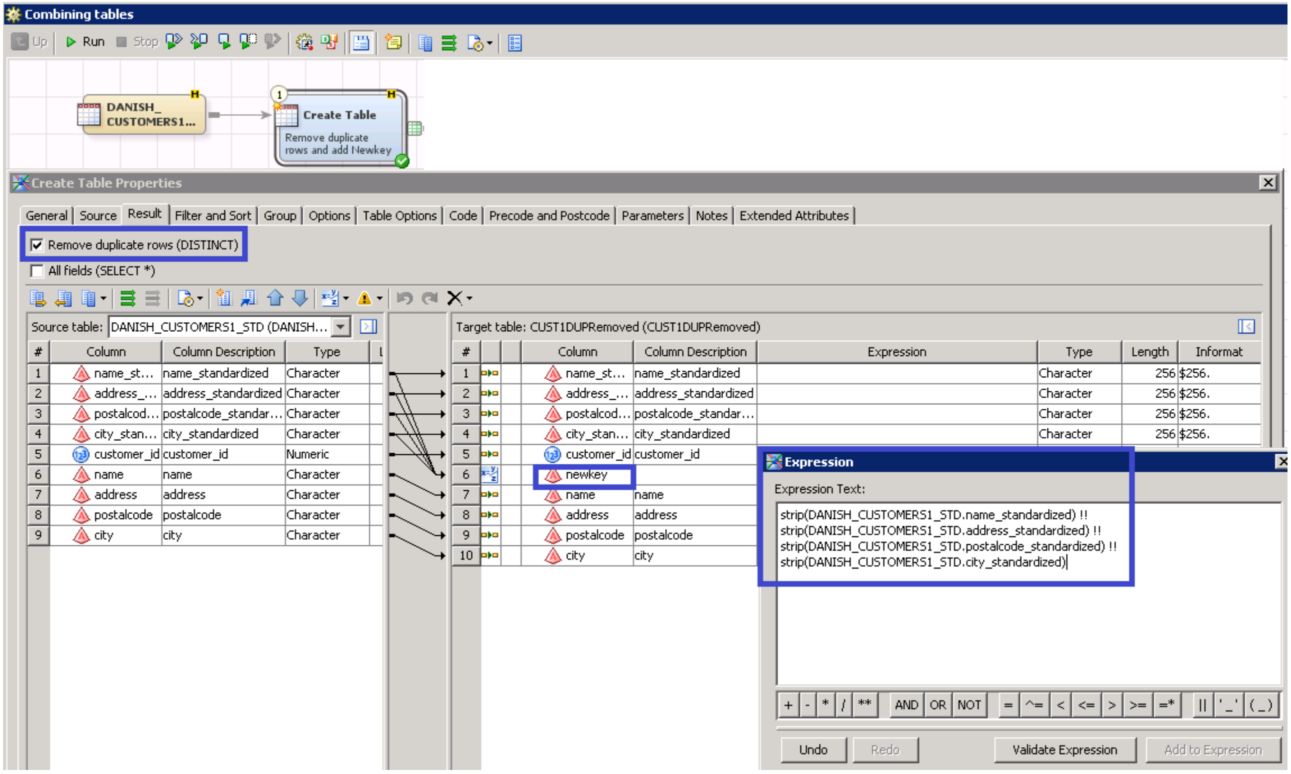


The following LIBNAME statement has been generated for "RACE Temp Data_RACETEMP":

```
LIBNAME RACETEMP HADOOP  CONNECTION=GLOBAL  DBMSTEMP=Yes
PORT=10000 SERVER="sasdata1.demo.sas.com"  SCHEMA=Default
AUTHDOMAIN="DefaultAuth" ;
```

Display 15. Temporary Hadoop library definition.

2. I added the Create Table definition to the job and connected the table DANISH_CUSTOMERS1_STD.
3. In Create Table Properties, I ticked Remove duplicate rows and created a column called newkey with an expression containing the STRIP function and a concatenation of four variables. I did this because there are no CUSTOMER_IDs in the second table and I needed a key to compare the tables with each other.
4. I repeated the process for DANISH_CUSTOMERS2_STD.



Combining tables

DANISH_ CUSTOMERS1... → Create Table: Remove duplicate rows and add Newkey

Create Table Properties

General | Source | Result | Filter and Sort | Group | Options | Table Options | Code | Precode and Postcode | Parameters | Notes | Extended Attributes

☑ Remove duplicate rows (DISTINCT)

□ All fields (SELECT *)

Source table: DANISH_CUSTOMERS1_STD (DANISH...

#	Column	Column Description	Type
1	name_st…	name_standardized	Character
2	address_…	address_standardized	Character
3	postalcod…	postalcode_standar…	Character
4	city_stan…	city_standardized	Character
5	customer_id	customer_id	Numeric
6	name	name	Character
7	address	address	Character
8	postalcode	postalcode	Character
9	city	city	Character

Target table: CUST1DUPRemoved (CUST1DUPRemoved)

#	Column	Column Description	Expression	Type	Length	Informat
1	name_st…	name_standardized		Character	256	$256.
2	address_…	address_standardized		Character	256	$256.
3	postalcod…	postalcode_standar…		Character	256	$256.
4	city_stan…	city_standardized		Character	256	$256.
5	customer_id	customer_id				
6	newkey					
7	name	name				
8	address	address				
9	postalcode	postalcode				
10	city	city				

Expression

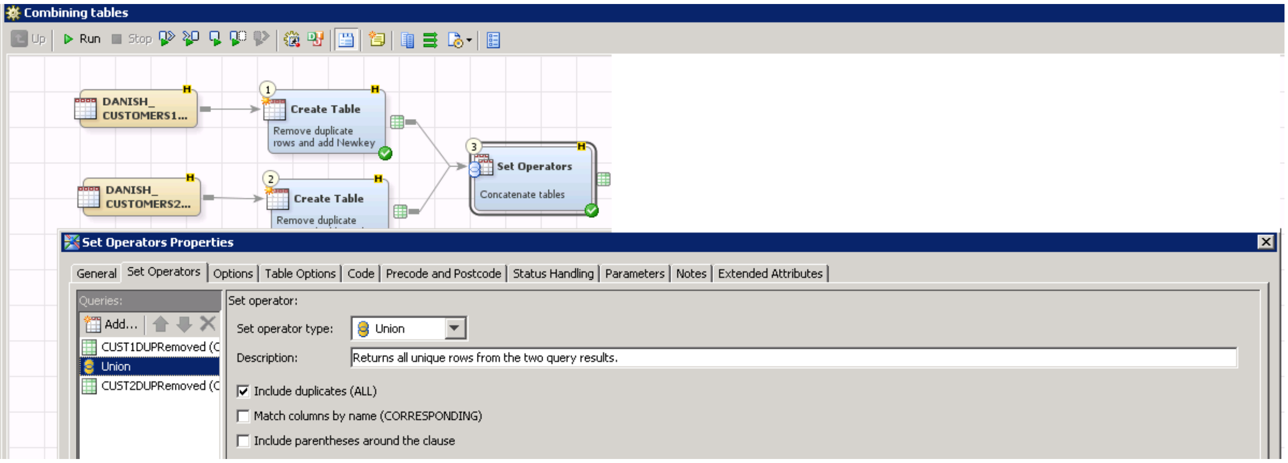
Expression Text:

```
strip(DANISH_CUSTOMERS1_STD.name_standardized) !!
strip(DANISH_CUSTOMERS1_STD.address_standardized) !!
strip(DANISH_CUSTOMERS1_STD.postalcode_standardized) !!
strip(DANISH_CUSTOMERS1_STD.city_standardized)
```

Display 16. Create Table Properties with visible expression content with mapped function.

5. I added the Set Operators transformation so that I could concatenate the temporary output Hadoop tables created in the Create Table transformation.

6. I selected a Union All in the Properties of the Set Operators transformation. With Set Operators, you need to be very wary when it comes to Hive because not all features are supported. This is a Hive limitation.



Combining tables

DANISH_ CUSTOMERS1... → Create Table: Remove duplicate rows and add Newkey → Set Operators: Concatenate tables

DANISH_ CUSTOMERS2... → Create Table: Remove duplicate ... → Set Operators

Set Operators Properties

General | Set Operators | Options | Table Options | Code | Precode and Postcode | Status Handling | Parameters | Notes | Extended Attributes

Queries: Add...

- CUST1DUPRemoved (C
- Union
- CUST2DUPRemoved (C

Set operator:

Set operator type: Union

Description: Returns all unique rows from the two query results.

☑ Include duplicates (ALL)

□ Match columns by name (CORRESPONDING)

□ Include parentheses around the clause

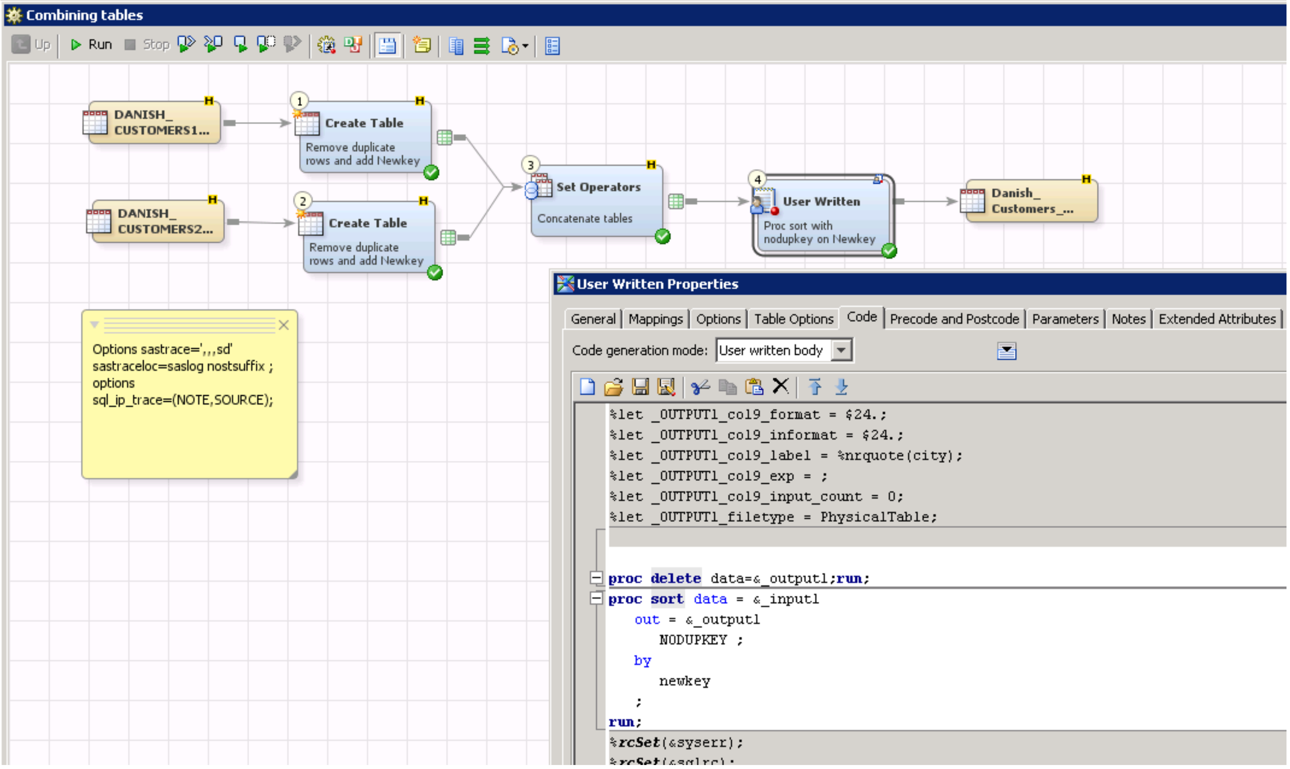
Display 17. Set operator properties with a Union All.

7. I added a User-Written transformation. In the user-written body, I added my own SAS program that removes duplicate keys.

I could have used the Sort transformation, but with that you need to heed the prerequisites described here to avoid the underlying automatic code from adding a step involving a SAS WORK table or view that would be detrimental to performance. It is always good practice to open the underlying code of a transformation to check whether WORK tables or views are being created.

Here are the Sort transformation prerequisites:

- The number of columns in the source table and target table must be the same.
- All columns must be mapped.
- All of the following attributes must be the same between mapped columns: name, format, informat, and description.



```
Options sastrace=',,,sd'
sastraceloc=saslog nostsuffix ;
options
sql_ip_trace=(NOTE,SOURCE);
```

Code generation mode: User written body

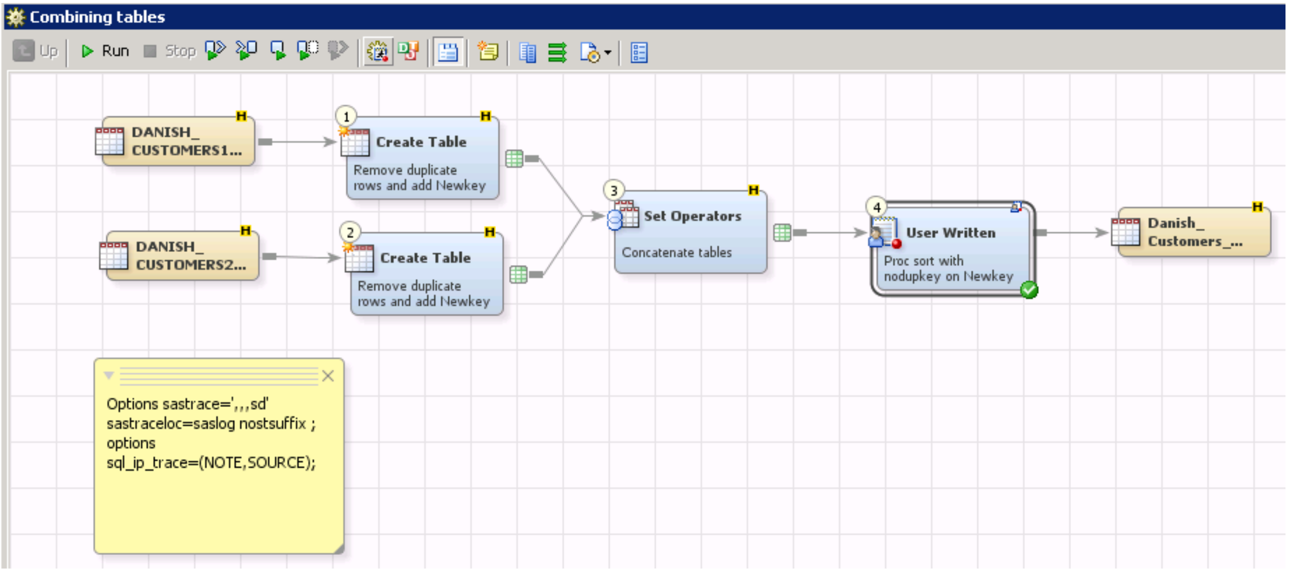
```
%let _OUTPUT1_col9_format = $24.;
%let _OUTPUT1_col9_informat = $24.;
%let _OUTPUT1_col9_label = %nrquote(city);
%let _OUTPUT1_col9_exp = ;
%let _OUTPUT1_col9_input_count = 0;
%let _OUTPUT1_filetype = PhysicalTable;


proc delete data=&_output1;run;
proc sort data = &_input1
   out = &_output1
      NODUPKEY ;
   by
      newkey
   ;
run;
%rcSet(&syserr);
%rcSet(&sqlrc);
```

Display 18. User-written body with custom SAS code.

8. I added the final output table that is registered in metadata, DANISH_CUSTOMERS_COMBINED, and ran the job.

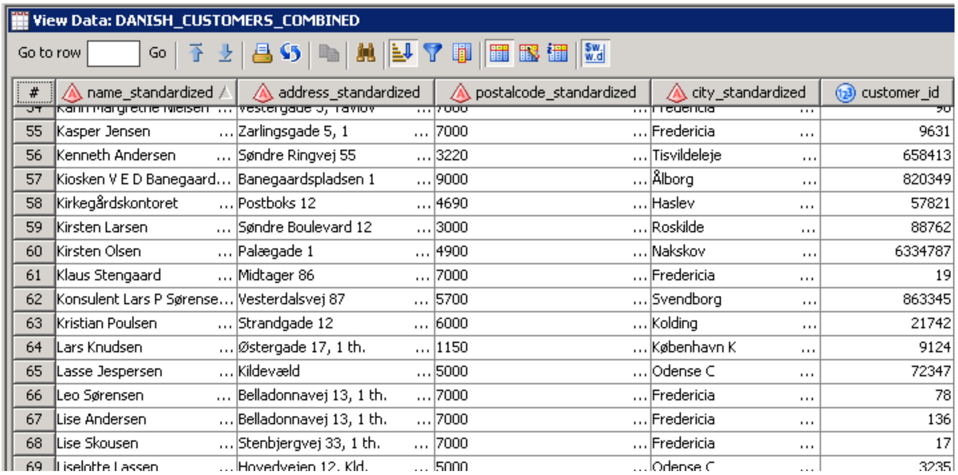
   Notice the little “H” on each transformation and table. This tells you that you are pushing code down to Hadoop. The user-written one lacks the “H” because it contains custom code, but the log trace and the “H” on the output table tells me that sorting was pushed down.



```
Options sastrace=',,,sd'
sastraceloc=saslog nostsuffix ;
options
sql_ip_trace=(NOTE,SOURCE);
```

Display 19. Job ran successfully and in Hadoop.

The output table is now ready to be used by the analysts!



#	name_standardized	address_standardized	postalcode_standardized	city_standardized	customer_id
54	Karin Margrethe Nielsen	Vestergade 5, Tavlov	7000	Fredericia	90
55	Kasper Jensen	Zarlingsgade 5, 1	7000	Fredericia	9631
56	Kenneth Andersen	Søndre Ringvej 55	3220	Tisvildeleje	658413
57	Kiosken V E D Banegaard…	Banegaardspladsen 1	9000	Ålborg	820349
58	Kirkegårdskontoret	Postboks 12	4690	Haslev	57821
59	Kirsten Larsen	Søndre Boulevard 12	3000	Roskilde	88762
60	Kirsten Olsen	Palægade 1	4900	Nakskov	6334787
61	Klaus Stengaard	Midtager 86	7000	Fredericia	19
62	Konsulent Lars P Sørense…	Vesterdalsvej 87	5700	Svendborg	863345
63	Kristian Poulsen	Strandgade 12	6000	Kolding	21742
64	Lars Knudsen	Østergade 17, 1 th.	1150	København K	9124
65	Lasse Jespersen	Kildevæld	5000	Odense C	72347
66	Leo Sørensen	Belladonnavej 13, 1 th.	7000	Fredericia	78
67	Lise Andersen	Belladonnavej 13, 1 th.	7000	Fredericia	136
68	Lise Skousen	Stenbjergvej 33, 1 th.	7000	Fredericia	17
69	Liselotte Lassen	Hovedvejen 12, Kld.	5000	Odense C	3235

Output 6. View of table results after running job.

GAINING FROM AMAZING MEMORY WITH SAS DATA PREPARATION

It is said that elephants have an amazing memory and intelligence, remembering the many times they leave the lake to discover land untrodden, gaining invaluable insight about hidden watering holes and bountiful pastures. This use case is a walk down memory lane!

USE CASE

Using SAS Data Preparation, you want to do some last-mile data preparation using a Hive Hadoop table as source before moving on to analytics, which ultimately is your primary focus. To do that, you need to load the Hive table to memory. In SAS terminology, it's called SAS® Cloud Analytic Services (CAS). You do a quick profile and discover that 9 rows of contact information lack a customer ID. You decide to investigate further, and match coding reveals that there is another data quality issue. There seem to be customers who are duplicated in the data. The ETL developers creating the data mart have now become aware of this, and the issues are resolved during the next batch load. In the interim, you could resolve the quality issues yourself using SAS Data Preparation's many transforms. You might have to do this in other cases when the data is completely raw, and there are no well-defined data integration processes supporting your data needs. Fixing data quality issues in SAS Data Preparation is beyond the scope of this paper.

SAS DATA PREPARATION IS TAMING DATA IN CAS

SAS Data Preparation offers an easy way for you to prepare data, providing an interactive, self-service environment for users who need to access, blend, shape and cleanse data to prepare it for reporting or analytics. There is also a seamless integration with analytics applications such as SAS® Visual Data Mining and Machine Learning, SAS® Visual Statistics, SAS® Decision Manager and SAS® Model Studio, providing a continuous flow throughout the analytical life cycle.

YOUR CAS TAMER PROFILE

You are an analyst, business user, data scientist or anyone else who performs tasks involving data preparation and analytics.

YOUR CAS TAMER ACCESSORIES

To solve this use case, you need the SAS software packages SAS® Visual Analytics and SAS Data Preparation on SAS Viya. For the full analytical life cycle, you might consider SAS Visual Statistics, SAS Visual Data Mining and Machine Learning, SAS® Visual Text Analytics and other analytics packages.

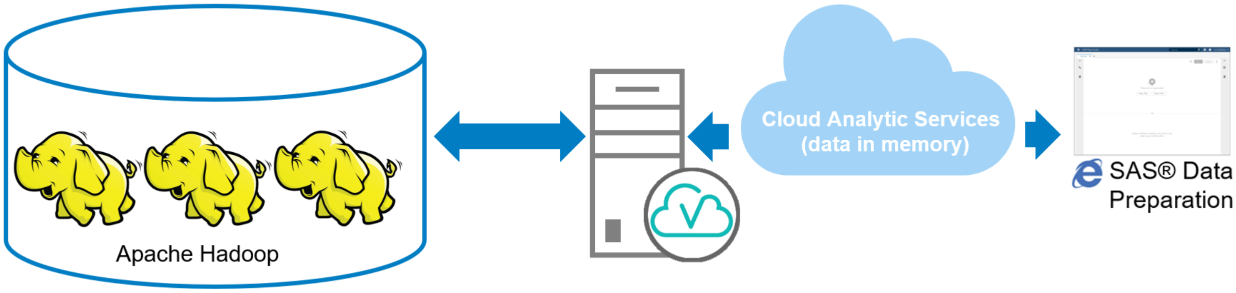
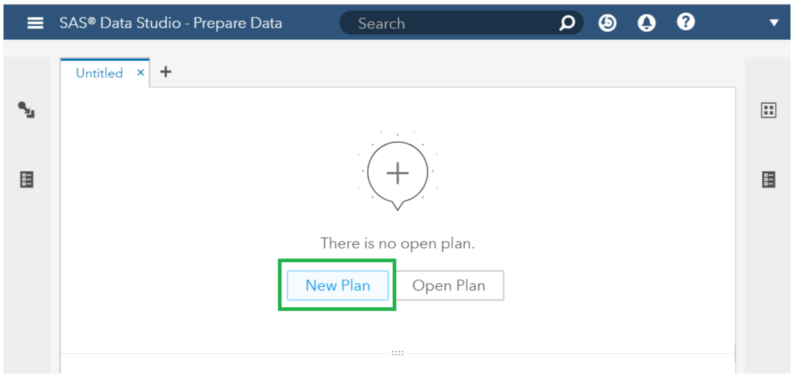


Figure 6. A simple architecture for SAS Data Preparation on SAS Viya.

MASTERING CAS

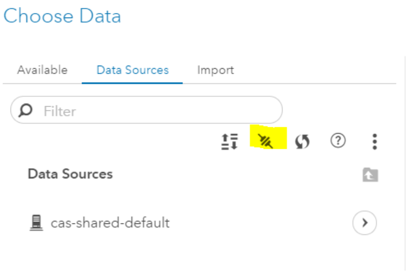
SAS Data Preparation is about creating plans that are a collection of transforms performed on a table. These are the overall steps I took to create and populate a plan:

1. I created a new plan.



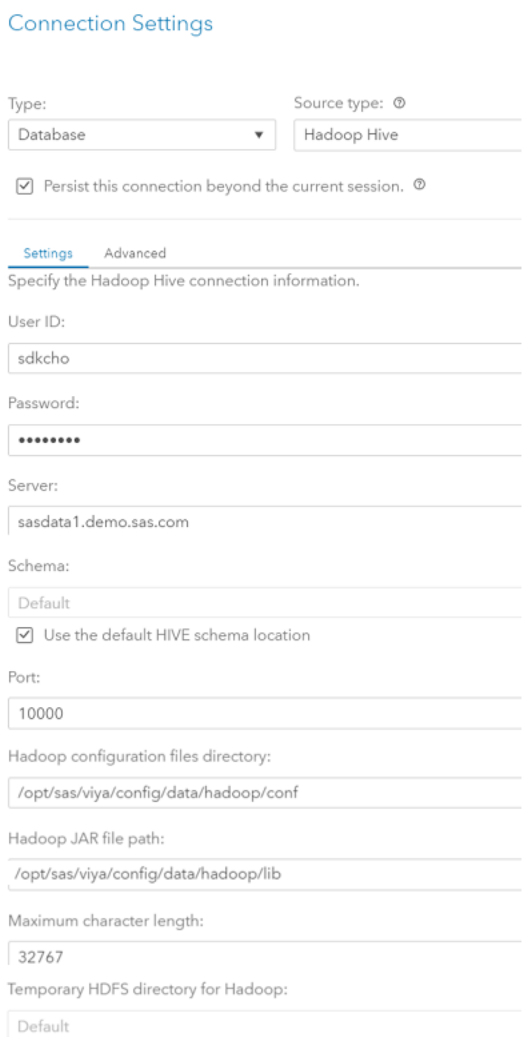
Display 20. Starting out with a new plan in SAS Data Preparation.

2. To set up a new connection, I clicked the Connect tool.



Display 21. Overview of Connect tool.

3. I added connection settings to the Hive data source. Your Hadoop administrator can provide you with the appropriate connection details.



Connection Settings

Type: Database ▾

Source type: Hadoop Hive

☑ Persist this connection beyond the current session.

Settings | Advanced

Specify the Hadoop Hive connection information.

User ID: sdkcho

Password: ••••••••

Server: sasdata1.demo.sas.com

Schema: Default

☑ Use the default HIVE schema location

Port: 10000

Hadoop configuration files directory: /opt/sas/viya/config/data/hadoop/conf

Hadoop JAR file path: /opt/sas/viya/config/data/hadoop/lib

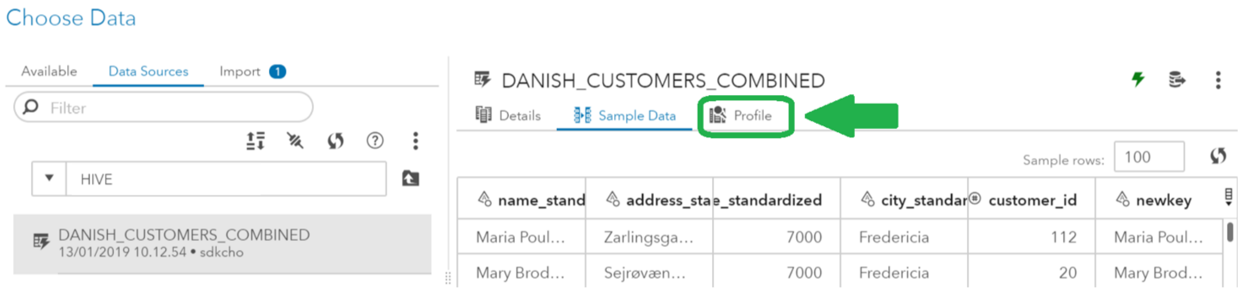
Maximum character length: 32767

Temporary HDFS directory for Hadoop: Default

Display 22. Connection settings to Hive Hadoop tables.

4. I loaded my Hive table into CAS. In my case, I performed a serial load, but depending on your deployment and your data, you can also load data to CAS in parallel. CAS loading techniques are very much a topic worth investigating, and the online documentation is a must-read!

   Once my Hive table was loaded to CAS, I highlighted it to create a profile.



Choose Data

Available | Data Sources | Import 1

Filter

HIVE

DANISH_CUSTOMERS_COMBINED
13/01/2019 10.12.54 • sdkcho

DANISH_CUSTOMERS_COMBINED

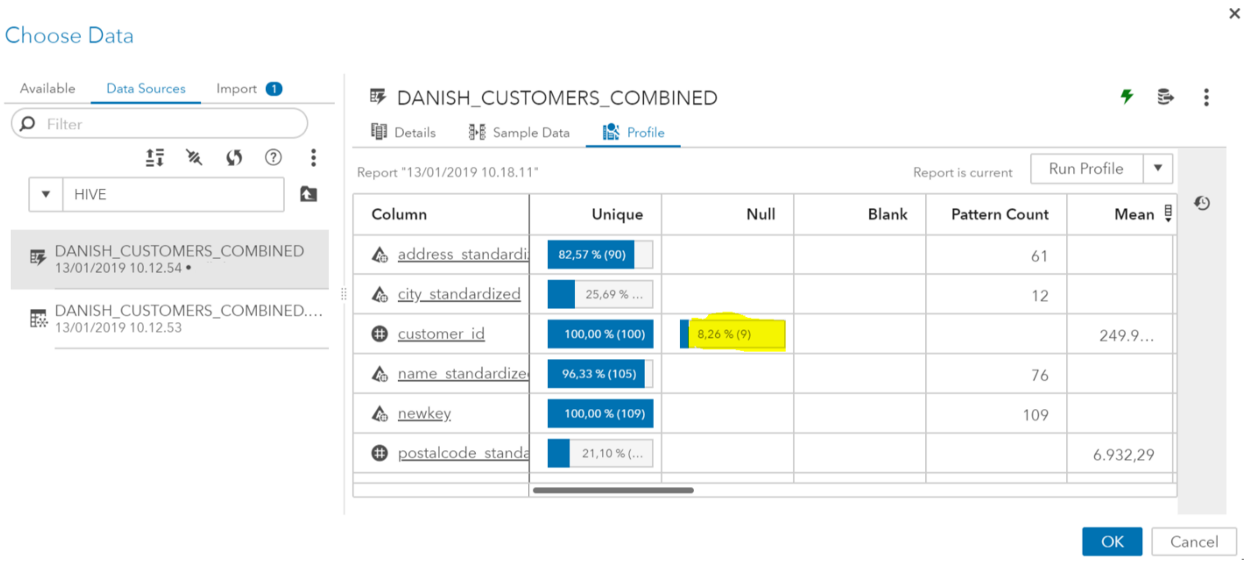
Details | Sample Data | Profile

Sample rows: 100

name_stand	address_sta	e_standardized	city_standar	customer_id	newkey
Maria Poul…	Zarlingsga…	7000	Fredericia	112	Maria Poul…
Mary Brod…	Sejrøvæn…	7000	Fredericia	20	Mary Brod…

Display 23. Highlighted CAS table and Profile menu.

The profile revealed that there were missing customer IDs.



DANISH_CUSTOMERS_COMBINED

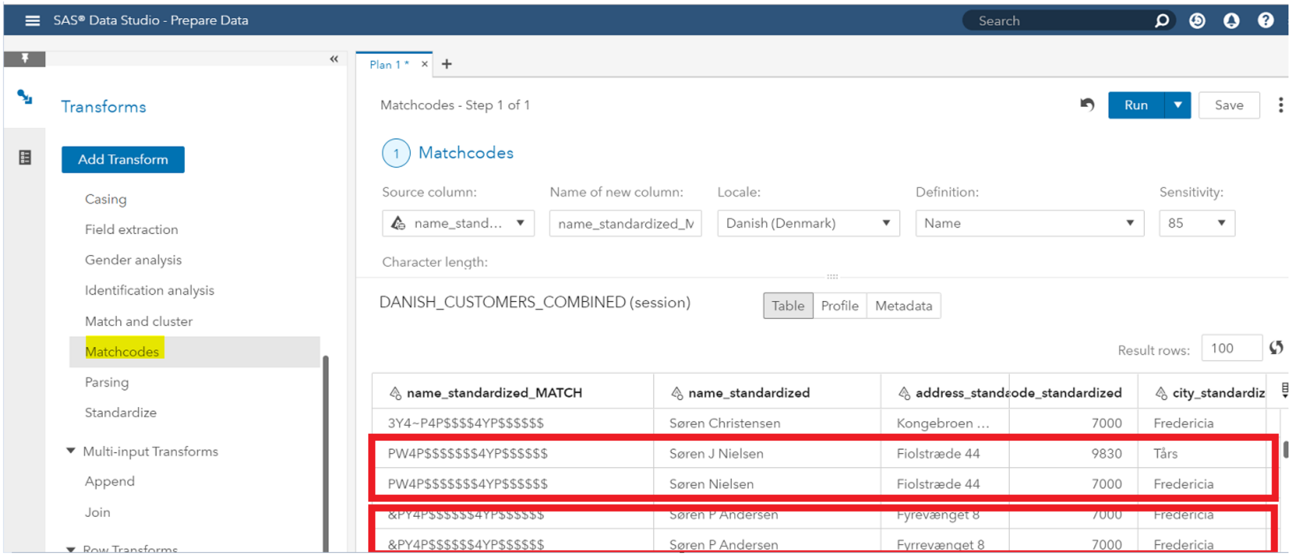
Report "13/01/2019 10.18.11"

Report is current

Column	Unique	Null	Blank	Pattern Count	Mean
address_standardi	82,57 % (90)			61	
city_standardized	25,69 % ...			12	
customer_id	100,00 % (100)	8,26 % (9)			249.9...
name_standardize	96,33 % (105)			76	
newkey	100,00 % (109)			109	
postalcode_standa	21,10 % (...				6.932,29

Display 24. Profile revealing nine missing occurrences for customer_id.

5. I investigated the data further using the Matchcodes transform. Here I discover that there seems to be more than one row for certain customers.



Matchcodes - Step 1 of 1

1 Matchcodes

Source column: name_stand...

Name of new column: name_standardized_M

Locale: Danish (Denmark)

Definition: Name

Sensitivity: 85

Character length:

DANISH_CUSTOMERS_COMBINED (session)

Result rows: 100

name_standardized_MATCH	name_standardized	address_standa	ode_standardized	city_standardiz
3Y4~P4P$$$$4YP$$$$$$	Søren Christensen	Kongebroen ...	7000	Fredericia
PW4P$$$$$$$4YP$$$$$$	Søren J Nielsen	Fiolstræde 44	9830	Tårs
PW4P$$$$$$$4YP$$$$$$	Søren Nielsen	Fiolstræde 44	7000	Fredericia
&PY4P$$$$$$4YP$$$$$$	Søren P Andersen	Fyrevænget 8	7000	Fredericia
&PY4P$$$$$$4YP$$$$$$	Søren P Andersen	Fyrrevænget 8	7000	Fredericia

Display 25. Match codes reveal data quality issues in customer information records.

Once you have tamed your data, or someone tames it for you, you can head on to analytics using SAS Visual Statistics, SAS Visual Data Mining and Machine Learning, SAS Visual Text Analytics and other SAS analytics applications. After all, analytics is your primary focus, and CAS has an amazing memory for analytics capabilities!

CONCLUSION

This tale of three SAS® Data Management applications with interconnected use cases and a natural data flow between them serves as your inspiration and motivation to implement the right technology that supports your organization's data management requirements for data residing in a Hadoop data lake. Different user profiles require applications that suit their business requirements and proficiency, and this leads to faster insight about your data.

Data scientists are a rare breed, and the SAS Data Management applications discussed here help fill the personnel gap!

It is all about doing the right thing with the right tool with the right knowledge for the right reason in aid of your business!

ACKNOWLEDGMENTS

Many thanks to Jens Dahl Mikkelsen, Principal Advisor, SAS Institute Denmark, for reviewing this paper.

RECOMMENDED READING

SAS Support for Hadoop:

https://support.sas.com/en/documentation/third-party-software-reference/9-4/support-for-hadoop.html

Hadoop prerequisites:

https://go.documentation.sas.com/?docsetId=dplyml0phy0lax&docsetTarget=n10oo4yrm0x0ygn1g8kug0yw12gg.htm&docsetVersion=3.4&locale=en

Introduction to SAS and Hadoop technology:

https://go.documentation.sas.com/?docsetId=hadoopov&docsetTarget=p1d3oooypq5aemn1e3t2cbkvxm6p.htm&docsetVersion=9.4&locale=en

Working with Hive:

https://go.documentation.sas.com/?docsetId=hadoopbacg&docsetTarget=n1xshq8hrrog8sn1oo7h65792c71.htm&docsetVersion=9.4&locale=en

In-database processing:

http://support.sas.com/documentation/onlinedoc/indbtech/

Blog on SQL implicit pass-through:

https://blogs.sas.com/content/sgf/2018/03/29/implicit-sql-pass-through-to-hive-in-sas-viya/

SAS Data Loader for Hadoop:

http://support.sas.com/documentation/onlinedoc/dmdd/index.html

SAS Data Integration Studio techniques for ELT and ETL:

https://support.sas.com/resources/papers/proceedings10/116-2010.pdf

SGF paper on in-database processing:

https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2823-2018.pdf

QKB locale ISO codes:

http://support.sas.com/documentation/onlinedoc/qkb/28/QKBCI28/Help/qkb-help.html#qkbci-generaldoc/qkbci-locisocodes.html%3FTocPath%3DAppendix%7C_____1

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Cecily Hoffritz

SAS Institute Denmark

cecily.hoffritz@sas.com

http://www.sas.com