Paper 078-31

Efficient “One-Row-per-Subject” Data Mart Construction for Data Mining

Gerhard Svolba, PhD, SAS Austria

ABSTRACT

Creating a "one-row-per-subject" data mart is a fundamental task when preparing data for data mining. To answer the underlying business question, the analyst or data-mart programmer is challenged to distill the relevant information from various data sources.

Creating a data mart involves more than reading columns from a source table to the data mart. It also includes the aggregation or transposition of observations from "multiple-row-per-subject" tables like transactional tables and time histories. This process is a critical success factor for being able to answer the business question or to have good predictors available for a target event or target value.

This paper shows the details of input data structures for a "one-row-per-subject" data mart. It also discusses the "one-row-per-subject" paradigm from a technical and a business point of view, and shows how data, some of which include hierarchical dependencies, is aggregated into a “single-row-per-subject.” A comprehensive example shows how a "one-row-per-subject" data mart is created from various data sources.

INTRODUCTION

Besides selecting and tuning the appropriate statistical model, data availability and data preparation are fundamental to the successful performance of an analytical model. Depending on the business questions and on the properties of the statistical model, different data mart structures, like the one-row-per-subject data mart that is necessary for predictive modeling or the multiple-row-per subject data mart that is required for market basket analysis, must be assembled.

Answering a business problem usually includes the following steps:

1. Define the business problem. A lot of collaboration and discussion between business people, analysts, and IT people is needed in order to evaluate what exactly is needed.
2. Based on the definition of the business problem, determine what data is available to help you answer the business question(s), particularly observational data rather than data that are specifically gathered for an analysis.
3. Determine how to prepare the data for analytical modeling and the analytical data mart.
4. Feed the analytical data mart into the analytical procedure to analyze relationships, predict values and events, or segment subjects, to name a few tasks.

There are two main factors that influence the quality of an analytical model:

- The selection and tuning of the appropriate analytical model
- The availability and preparation of data that is suitable to answer the business question.

SUCCESS FACTORS FOR ANALYTICS

In statistical theory and literature there is much focus on the selection and tuning of an analytical model. There are many books and courses that focus on predictive modeling--training neural networks, creating decisions trees, or building regression models. For all these tasks a lot of knowledge is needed and has proven to be a critical success factor for good analytical results.

Besides the importance of analytical skills, however, having good and relevant data to help answer business questions is also very important.

SUCCESS FACTORS FOR DATA

The *availability of the data* for creating the analytical data mart is an important consideration to evaluate early in a project. Determining whether historic snapshots of the data that are necessary for an analysis will be available is a long-term planning issue and is not something that can be determined at the beginning of the data preparation process.

The *adequate preparation of the data* is an important success factor for analysis. Even if the right data sources are available and can be accessed, appropriately preparing the data is an essential pre-requisite for a good model. In the author's experience with numerous projects, a missing or improperly derived variable or aggregation cannot be covered up by any clever parameterization of the model.

This paper focuses on the adequate preparation of the data. This paper contains the content of selected chapters from a soon to be released book *Data Preparation for Analytics* written by Gerhard Svolba (read more about this book in the section "Recommended Reading"). Following is a discussion of conceptual relationships and data structures when creating a one-row-per-subject data mart. The creation process is further illustrated by a case study.

THE ONE-ROW-PER-SUBJECT PARADIGM

ANALYSIS SUBJECT

Analysis subjects are entities that are being analyzed and the analysis results are interpreted in their context. Analysis subjects are therefore also the basis for the structure of analysis tables.

Examples of analysis subjects are:

- Persons: Depending on the area of the analysis the analysis subjects have more specific names like patients in medical statistics, customers in marketing analytics, or applicants in credit scoring.
- Animals: Piglets, for example, are analyzed in feeding experiments; rats are analyzed in pharmaceutical experiments.
- Parts of the body system: In medical research analysis subjects can also be parts of the body system like arms (left arm compared to the right arm), shoulders or hips. Note that from a statistical point of view the validity of the assumptions of the respective analysis methods have to be checked, if dependent observations per person are used in the analysis.
- Objects: Things like cash machines in cash demand prediction, cars in quality control in the automotive industry, or products in product analysis can be subjects.
- Legal entities: Examples are companies, contracts, accounts, and applications.
- Geography: Regions or plots in agricultural studies or reservoirs in the maturity prediction of fields in the oil and gas industries are examples.

Analysis subjects are therefore the heart of each analysis. The features of analysis subjects are measured, processed, and analyzed. In deductive statistics the features of the analysis subjects in the sample are used to infer the properties of the similar subjects in the larger population.

IMPORTANCE AND FREQUENT OCCURRENCE OF THIS TYPE OF DATA MART

The one-row-per-subject data mart is important for statistical and data mining analyses. Many business questions are answered based on a data mart of this type. The following list presents business examples:

- Prediction of events such as
  - Campaign response
  - Contract renewal
  - Contract cancellation (attrition or churn)
  - Insurance claim
  - Default on credit or loan payback
  - Identification of risk factors for a certain disease
- Prediction of values such as
  - Amount of loan that can be paid back
  - Customer turnover and profit for the next period
  - Time intervals like the remaining lifetime of a customer or patient

- Time until next purchase
- Value of next insurance claim

- Segmentation of customers

The following analytical methods require a one-row-per-subject data mart:

- Regression analysis
- Analysis-of-variance analysis
- Neural networks
- Decision trees
- Survival analysis
- Cluster analysis
- Principle components analysis and factor analysis

PUTTING ALL INFORMATION INTO ONE

The above application examples coupled with the underlying algorithm requires that all the data source information be aggregated into one row. Central to this type is the fact that we have to put all information per analysis subject into one row. We will therefore call this the *one-row-per-subject data mart paradigm* in that all information has to be put into a single row. Multiple observations per analysis subject must not appear in additional rows, but have to be converted over into additional columns.

Obviously, putting all information into one row is very simple if we have no multiple observations per analysis subject. The values are then simply read into the analysis data mart and derived variables are being added.

If we have multiple observations per analysis subject it is more effort to put all information into one row. Here we have to cover two aspects:

- The technical aspect--how multiple-row-per-subject data can be converted into one-row-per-subject data.
- The business aspect--which aggregations, derived variables, and so on can best be condensed from the multiple rows into columns.

The process of taking data from tables that have one-to-many relationships and putting them into a rectangular one-row-per subject analysis table has many names: transposing, denormalizing, “making flat”, pivoting, and others.

Figure 1 illustrates the creation of a one-row-per-subject data mart in terms of an entity relationship diagram.

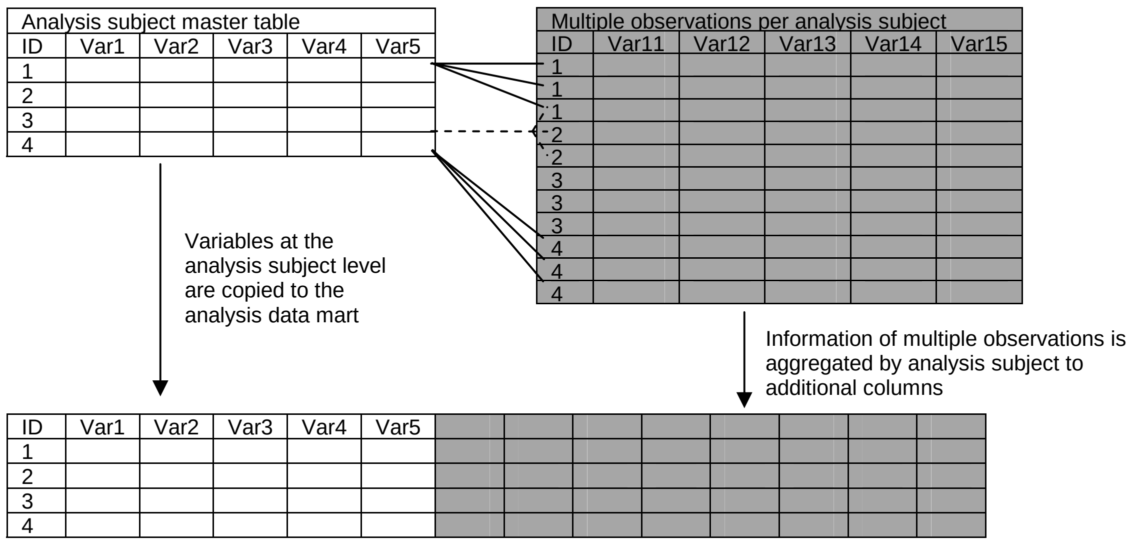

Analysis subject master table
ID Var1 Var2 Var3 Var4 Var5
1 2 3 4
Multiple observations per analysis subject
ID Var11 Var12 Var13 Var14 Var15
1 1 1 2 2 3 3 3 4 4 4
Variables at the analysis subject level are copied to the analysis data mart
Information of multiple observations is aggregated by analysis subject to additional columns
ID Var1 Var2 Var3 Var4 Var5
1 2 3 4

Figure 1. Creation of a One-Row-Per-Subject Data Mart

THE TECHNICAL POINT OF VIEW

From a technical perspective, you can solve the task of putting all information into one row using two main techniques:

- Transposing. Here we transpose the multiple rows per subject into columns. This technique can also be named as the "pure way," as we take all data from the rows and represent it in columns.
- Aggregating. Here we aggregate the information from the columns into an aggregated value per analysis subject. We perform information reduction by trying to express the content of the original data into descriptive measures that are derived from the original data.

Transposing

Here is a very simple example with three multiple observations per subject (Table 1).

PatNr	Gender
1	Male
2	Female
3	Male

Table 1. Base Table with Static Information per Patient (analysis subject)

Next, we have a table with repeated measurements per patient (Table 2).

PatNr	Measurement	Cholesterol
1	1	212
1	2	220
1	3	240
2	1	150
2	2	145
2	3	148
3	1	301
3	2	280
3	3	275

Table 2. Table with Multiple Observations per Patient (analysis subject)

We have to bring data from the table with multiple observations per patient into a form that can be joined to the base table on a 1:1 basis. We therefore transpose Table 2 by patient number and bring all repeated elements into columns (Table 3).

PatNr	Cholesterol 1	Cholesterol 2	Cholesterol 3
1	212	220	240
2	150	145	148
3	301	280	275

Table 3. Multiple Observations in a Table Structure with One Row per Patient

This information is then joined with the base table and we retrieve our analysis data mart in the requested data structure (Table 4).

PatNr	Gender	Cholesterol 1	Cholesterol 2	Cholesterol 3
1	Male	212	220	240
2	Female	150	145	148
3	Male	301	280	275

Table 4. The Final One-Row-Per-Subject Data Mart

Notice that we brought all information 1:1 from the multiple-row-per-subject data set to the final data set by transposing the data. This data structure is suitable for analyses such as repeated measurements for analysis of variance.

Aggregation

If we aggregate the information from the multiple-row-per-subject data set, for example, by using descriptive statistics such as the median, the minimum and maximum, or the interquartile range, we do not bring the original data 1:1 to the one-row-per-subject data set, but instead analyze a condensed version of the data. The data can then look like that shown in Table 5.

PatNr	Gender	Cholesterol_Mean	Cholesterol_Std
1	Male	224	14.4
2	Female	148	2.5
3	Male	285	13.8

Table 5. Aggregated Data from Tables 1 and 2

We see that aggregations do not produce as many columns as transpositions do when we condense the information. The forms of aggregations, however, have to be carefully selected, because omitting an important and--from a business point of view--relevant aggregation means that information is lost. There are no “rules” for selecting the best aggregation measures for all the different business questions. Domain-specific knowledge is key when selecting them. In predictive analysis, for example, we try to create candidate predictors that reflect the properties of the underlying subject and its behavior as well as possible.

THE BUSINESS POINT OF VIEW - TRANSPOSING OR AGGREGATING ORIGINAL DATA

In the above section we saw two ways to bring multiple observation data into a one-row-per-subject structure: transposing and aggregating. Selecting the appropriate method depends on the business question and the analytical method.

We have already mentioned in the section above that there are analyses in which all data for a subject is needed in columns but no aggregation of observations makes sense. For example the repeated measurement analysis of variance needs the repeated measurement values transposed into columns. In cases where we want to calculate indicators describing the course of a time series--for example, the mean usage in months 1 to 3 before cancellation divided by the mean usage in months 4 to 6--the original values are needed in columns.

There are, however, a lot of situations in which a 1:1 transfer of data is technically not possible nor is it practical. In these situations aggregations make much more sense. Let’s look at the following questions:

- If we have the cholesterol values expressed in columns, is this the information we need for further analysis?
- What shall we do if we have not only three observations for one measurement variable per subject, but 100 repetitions for 50 variables? Transposing all these data would lead to 5000 variables.
- Do we in the analysis data mart really need the data 1:1 from the original table, or do we more precisely need the information in concise aggregations?
- In predictive analysis, do we need data that has good predictive power for the modeling target and suitably describes the relationship between target and input variables instead of the original data?

The answers to these questions depend on the business questions, the modeling technique, the data itself, and so on. However there are many cases, especially in data mining analysis, where clever aggregations of the data make much more sense than a 1:1 transposition. Determining how to best aggregate the data requires business and domain knowledge and a good understanding of the data.

Having said that, putting all information into one row is not just data management; we show in this paper ways to cleverly prepare data from multiple observations. Following is an overview of methods that enable the information to be extracted and represented from the multiple observations per subject.

When aggregating *measurement data* for a one-row-per-subject data mart we can use the following:

- simple descriptive statistics like sum, minimum, or maximum
- frequency counts and number of distinct values
- measures of location or dispersion like mean, median, standard deviation, the quartiles, or special quantiles
- concentration measures on the basis of cumulative sums
- statistical measures describing trends and relationships like regression coefficients or correlations coefficients

When analyzing multiple observations with *categorical data* that is aggregated per subject we can use the following:

- total counts
- frequency counts per category and percentages per category
- distinct counts
- concentration measures on the basis of cumulative frequencies

HIERARCHIES: “AGGREGATING UP” AND “COPYING DOWN”

General

In the case of *repeated measurements* over time we are usually aggregating this information by using the summary measurements described above. The aggregated values, such as means or sums, are then used at the subject level as a property of the corresponding subject.

In the case of *multiple observations because of hierarchical relationships*, we are also aggregating information from a lower hierarchical level to a higher one. However, we can also have information at a higher hierarchical level that has to be available or expressed to lower hierarchical levels. Also, information has to be shared between members of the same hierarchy, like the number of members at this hierarchy level.

Example

Consider an example in which we have the following relationships:

- In one HOUSHOLD lives one or more CUSTOMER(s).
- One CUSTOMER can have one or more CREDIT CARD(s).
- On each CREDIT CARD a number of TRANSACTION(s) are associated.

For marketing purposes we do a segmentation of customers; therefore, CUSTOMER is chosen as the appropriate subject analysis level. When looking at the hierarchies, we aggregate information from the entity of CREDIT CARD and TRANSACTION to the CUSTOMER level and make information that is stored on the HOUSEHOLD level available to the entity CUSTOMER.

We next show how data from different hierarchical levels is made available to the analysis data mart. Note that for didactical purposes, we reduce the number of variables to a few.

Information from the Same Level

At the customer level we have the following variables directly available to our analysis data mart:

- customer birthday (on that basis the derived variable AGE is calculated)
- gender
- customer since (on that basis the duration of customer relationship is calculated)

Aggregating Up

In our example, we create the following variables on customer level that are aggregations from the lower levels CREDIT CARD and TRANSACTION.

- number of cards per customer
- number of transactions over all cards for the last 3 months
- sum of transaction values over all cards for the last 3 months
- average transaction value for the last 3 months (calculated on the basis of the values over all cards)

Dividing Down

At the HOUSEHOLD level the following information is available, that is copied down to the customer level.

- number of persons in the household
- geographic region type of the household with the values RURAL, HIGHLY RURAL, URBAN, HIGHLY URBAN.

Graphical Representation

Figure 2 shows the graphical representation in the form of an entity relationship diagram and the respective variable flows.

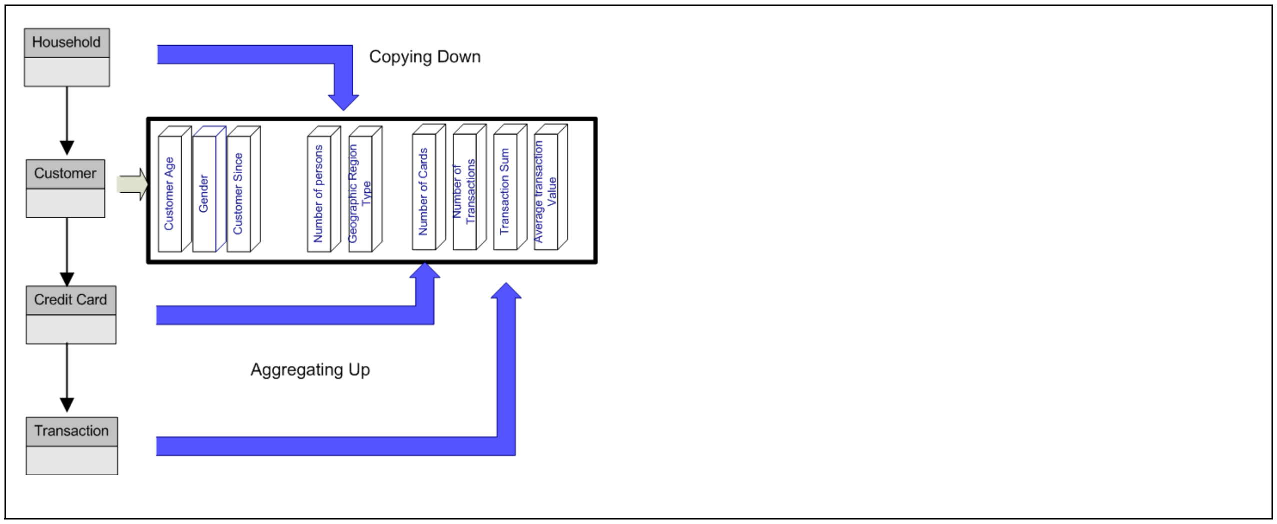


Figure 2. Entity Relationship Diagram - Information Flows Between Hierarchies

Conclusion

We see that information that is finally used at the analysis level CUSTOMER is retrieved from several levels. We have used simple statistical aggregations to illustrate this. In the following case study we go into more detail on this topic and also present the respective coding examples.

CASE STUDY – BUILDING A CUSTOMER DATA MART

INTRODUCTION

This section provides an example of how a one-row-per-subject table can be built using a number of data sources. In these data sources we only have in the CUSTOMER table one row per subject; in the other tables we have multiple rows per subject.

The task in this example is to combine these data sources to create a table that holds one row per customer. Therefore we have to aggregate the multiple observations per subject to one observation per subject

Tables like this are frequently needed in data mining analyses such as prediction or segmentation. In our example, we assume that we can create data to predict the cancellation event of a customer.

THE BUSINESS CONTEXT

General

We want to create a data mart that reflects the properties and behaviors of customers. This table will be used to predict which customer has a high probability to leave the company.

Data Sources

In this example we will build a customer data mart for marketing purposes. Our basis is data from the following data sources:

- *Customer data*: demographic and customer baseline data
- *Account data*: information customer accounts
- *Leasing data*: data on leasing information
- *Call Center data*: data on Call center contacts
- *Score data*: data of value segment scores

From these data sources we want to build one table with one row per customer. We therefore have the classic one-row-per-subject paradigm in which we need to bring the repeated information from, for example, the account data to several columns.

Considerations for Predictive Modeling

Determining whether a customer has left the company can be derived from its value segment. The SCORES table (see Table 10) holds for each customer the ValueSegment for the previous month. If ValueSegment has the value '8. LOST' then the customer cancelled his contract in the previous month. Otherwise ValueSegment has values such as '1. GOLD', '2. SILVER' which represent the customer value grading.

We define our snapshot date as June 1, 2003. This means that in our analysis we are allowed to use all information that is known on or before this date. We use the month June 2003 as the *offset window* and the month July 2003 as the *target window*. This means we want to predict which data cancellation events took place in July 2003. Therefore we have to consider the ValueSegment in the SCORES table per August 1, 2003. The offset window is used to train the model to predict an event that does not take place immediately after the snapshot date but after some time. This makes sense, because retention actions can not start immediately after the snapshot data. They start after data is loaded into the data warehouse and scored. Then the retention actions are defined.

Reasons for Multiple Observations per Customer

We have different reasons for multiple observations per customer in the different tables:

- In the SCORES table data, for example, we have multiple observations because of scores for different points in time.
- In the ACCOUNT table data we have already aggregated data over time, but we have multiple observations per customer because of different contract types.
- The CALLCENTER table is a classic form of a transactional data structure. We have on row per call center contact.

THE DATA

We have five tables available:

1. CUSTOMER
2. ACCOUNTS
3. LEASING
4. CALLCENTER
5. SCORES

These tables can be merged by the unique CUSTID variable. Notice that, for example, the ACCOUNT table already holds data that are aggregated from a transactional ACCOUNT table. We already have one row per account type that holds aggregated data over time.

The data model and the list of available variables can be seen in Figure 3.

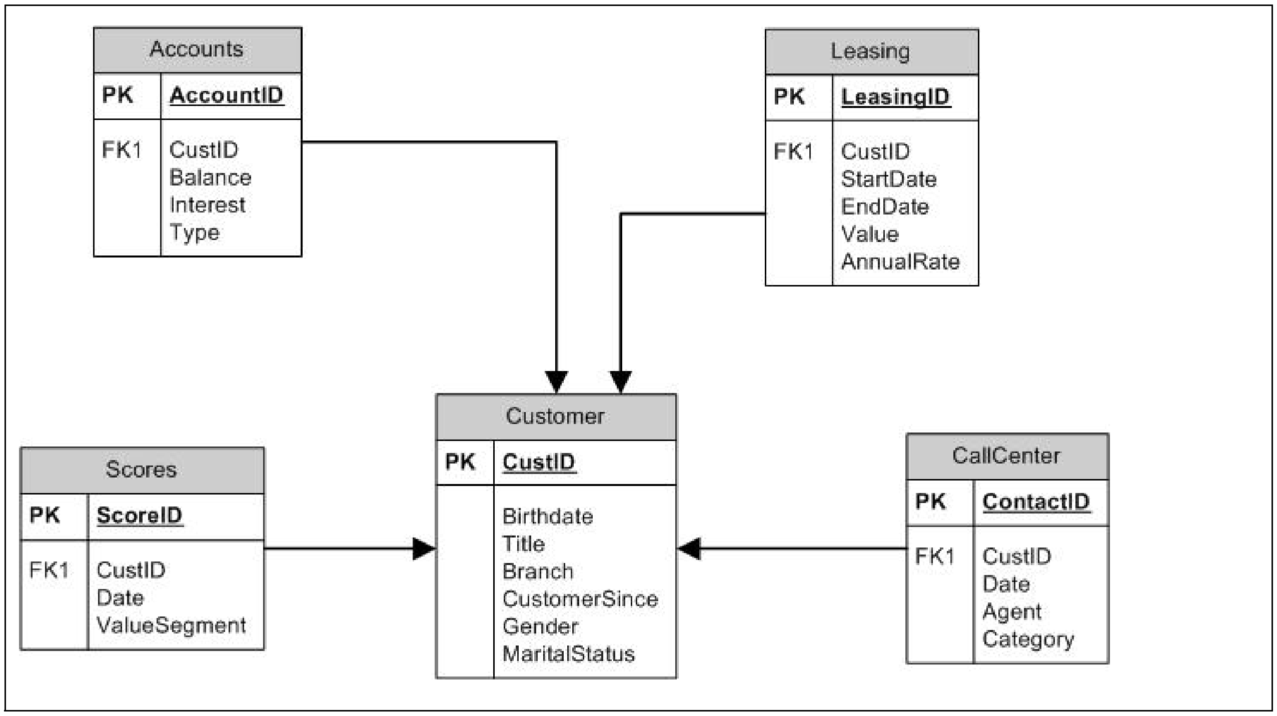
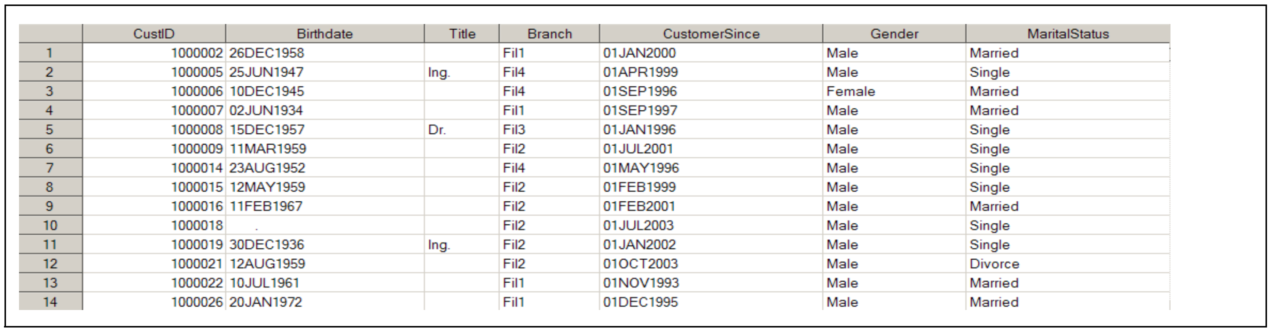


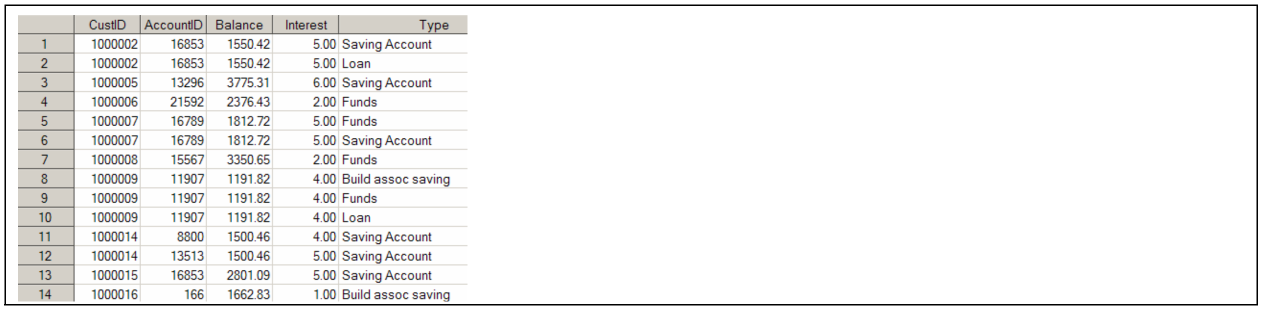
Figure 3. Data Model for Creating the CUSTOMER DATA MART

The following tables give a snapshot of the contents of the tables (Tables 6 to 10).



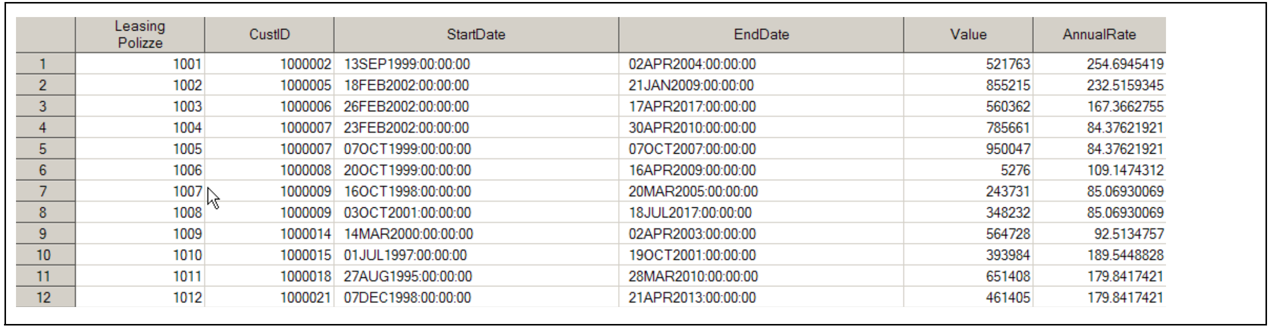
	CustID	Birthdate	Title	Branch	CustomerSince	Gender	MaritalStatus
1	1000002	26DEC1958		Fil1	01JAN2000	Male	Married
2	1000005	25JUN1947	Ing.	Fil4	01APR1999	Male	Single
3	1000006	10DEC1945		Fil4	01SEP1996	Female	Married
4	1000007	02JUN1934		Fil1	01SEP1997	Male	Married
5	1000008	15DEC1957	Dr.	Fil3	01JAN1996	Male	Single
6	1000009	11MAR1959		Fil2	01JUL2001	Male	Single
7	1000014	23AUG1952		Fil4	01MAY1996	Male	Single
8	1000015	12MAY1959		Fil2	01FEB1999	Male	Single
9	1000016	11FEB1967		Fil2	01FEB2001	Male	Married
10	1000018	.		Fil2	01JUL2003	Male	Single
11	1000019	30DEC1936	Ing.	Fil2	01JAN2002	Male	Single
12	1000021	12AUG1959		Fil2	01OCT2003	Male	Divorce
13	1000022	10JUL1961		Fil1	01NOV1993	Male	Married
14	1000026	20JAN1972		Fil1	01DEC1995	Male	Married

Table 6. CUSTOMER Table



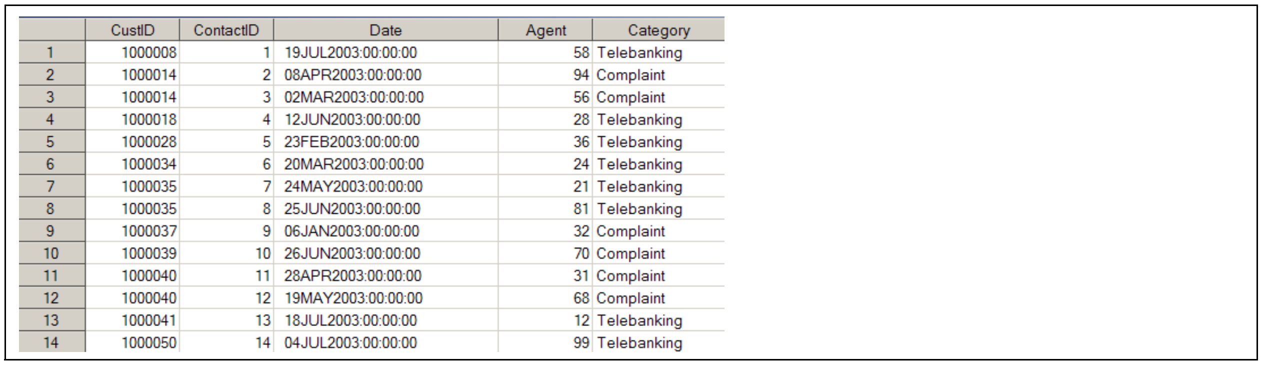
	CustID	AccountID	Balance	Interest	Type
1	1000002	16853	1550.42	5.00	Saving Account
2	1000002	16853	1550.42	5.00	Loan
3	1000005	13296	3775.31	6.00	Saving Account
4	1000006	21592	2376.43	2.00	Funds
5	1000007	16789	1812.72	5.00	Funds
6	1000007	16789	1812.72	5.00	Saving Account
7	1000008	15567	3350.65	2.00	Funds
8	1000009	11907	1191.82	4.00	Build assoc saving
9	1000009	11907	1191.82	4.00	Funds
10	1000009	11907	1191.82	4.00	Loan
11	1000014	8800	1500.46	4.00	Saving Account
12	1000014	13513	1500.46	5.00	Saving Account
13	1000015	16853	2801.09	5.00	Saving Account
14	1000016	166	1662.83	1.00	Build assoc saving

Table 7. ACCOUNTS Table



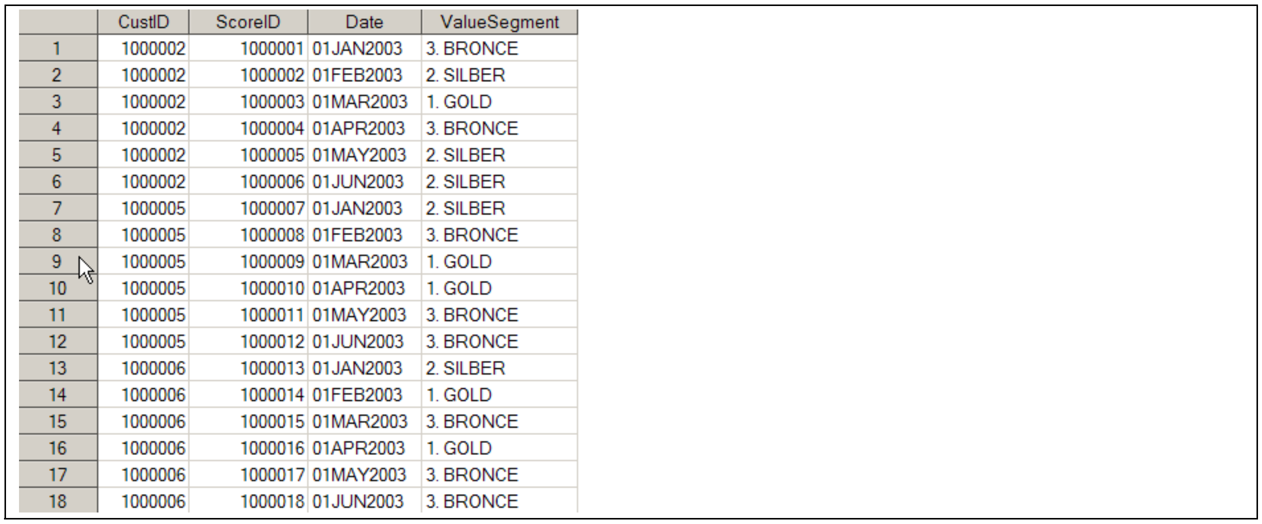
	Leasing Polizze	CustID	StartDate	EndDate	Value	AnnualRate
1	1001	1000002	13SEP1999:00:00:00	02APR2004:00:00:00	521763	254.6945419
2	1002	1000005	18FEB2002:00:00:00	21JAN2009:00:00:00	855215	232.5159345
3	1003	1000006	26FEB2002:00:00:00	17APR2017:00:00:00	560362	167.3662755
4	1004	1000007	23FEB2002:00:00:00	30APR2010:00:00:00	785661	84.37621921
5	1005	1000007	07OCT1999:00:00:00	07OCT2007:00:00:00	950047	84.37621921
6	1006	1000008	20OCT1999:00:00:00	16APR2009:00:00:00	5276	109.1474312
7	1007	1000009	16OCT1998:00:00:00	20MAR2005:00:00:00	243731	85.06930069
8	1008	1000009	03OCT2001:00:00:00	18JUL2017:00:00:00	348232	85.06930069
9	1009	1000014	14MAR2000:00:00:00	02APR2003:00:00:00	564728	92.5134757
10	1010	1000015	01JUL1997:00:00:00	19OCT2001:00:00:00	393984	189.5448828
11	1011	1000018	27AUG1995:00:00:00	28MAR2010:00:00:00	651408	179.8417421
12	1012	1000021	07DEC1998:00:00:00	21APR2013:00:00:00	461405	179.8417421

Table 8. LEASING Table



	CustID	ContactID	Date	Agent	Category
1	1000008	1	19JUL2003:00:00:00	58	Telebanking
2	1000014	2	08APR2003:00:00:00	94	Complaint
3	1000014	3	02MAR2003:00:00:00	56	Complaint
4	1000018	4	12JUN2003:00:00:00	28	Telebanking
5	1000028	5	23FEB2003:00:00:00	36	Telebanking
6	1000034	6	20MAR2003:00:00:00	24	Telebanking
7	1000035	7	24MAY2003:00:00:00	21	Telebanking
8	1000035	8	25JUN2003:00:00:00	81	Telebanking
9	1000037	9	06JAN2003:00:00:00	32	Complaint
10	1000039	10	26JUN2003:00:00:00	70	Complaint
11	1000040	11	28APR2003:00:00:00	31	Complaint
12	1000040	12	19MAY2003:00:00:00	68	Complaint
13	1000041	13	18JUL2003:00:00:00	12	Telebanking
14	1000050	14	04JUL2003:00:00:00	99	Telebanking

Table 9. CALLCENTER Table



	CustID	ScoreID	Date	ValueSegment
1	1000002	1000001	01JAN2003	3. BRONCE
2	1000002	1000002	01FEB2003	2. SILBER
3	1000002	1000003	01MAR2003	1. GOLD
4	1000002	1000004	01APR2003	3. BRONCE
5	1000002	1000005	01MAY2003	2. SILBER
6	1000002	1000006	01JUN2003	2. SILBER
7	1000005	1000007	01JAN2003	2. SILBER
8	1000005	1000008	01FEB2003	3. BRONCE
9	1000005	1000009	01MAR2003	1. GOLD
10	1000005	1000010	01APR2003	1. GOLD
11	1000005	1000011	01MAY2003	3. BRONCE
12	1000005	1000012	01JUN2003	3. BRONCE
13	1000006	1000013	01JAN2003	2. SILBER
14	1000006	1000014	01FEB2003	1. GOLD
15	1000006	1000015	01MAR2003	3. BRONCE
16	1000006	1000016	01APR2003	1. GOLD
17	1000006	1000017	01MAY2003	3. BRONCE
18	1000006	1000018	01JUN2003	3. BRONCE

Table 10. SCORES Table

Transactional Data or Snapshot

As we can see from the data examples only tables SCORES and CALLCENTER are real transactional tables. When processing data from these tables, we need to make sure that no observations after our snapshot data 01JUN2003 are used.

The other tables we consider as snapshots from 01JUN2003--we do not have to worry about the correct time span for these tables.

THE PROGRAMS

We will create the CUSTOMER DATA MART in two steps:

1. We will create aggregations on a subject level of the ACCOUNT, LEASING, CALLCENTER, and SCORES tables.
2. We will join the resulting tables together and create the CUSTOMER DATA MART table.

Transforming the ACCOUNTS Table

With the ACCOUNTS table, we want to perform two aggregations:

- derive sums per customer
- calculate an account type split based on the balance

We use PROC MEANS to create the output data set ACCOUNTMTP that contains both aggregations per customer and aggregations per customer and account type. Notice that we have intentionally omitted the NWAY option in order to have the full set of _TYPE_ categories in the output data set.

```
PROC MEANS DATA = accounts NOPRINT;
 CLASS CustID Type;
 VAR Balance Interest;
 OUTPUT OUT = AccountTmp(RENAME = (_FREQ_ = NrAccounts))
              SUM(Balance) = BalanceSum
              MEAN(Interest) = InterestMean;
RUN;
```

In the next step we use a DATA step with multiple output data sets to create:

- ACCOUNTSUM: that holds the aggregations on the customer level
- ACCOUNTTYPETMP: that is our basis for the account type split, which we need to transpose later on.

Notice that the creation of two output data sets is very elegant and we can rename the respective variables with a data set option. Based on the value of the _TYPE_ variable, we decide to which data set we output the respective observations.

```
DATA AccountSum(KEEP = CustID NrAccounts BalanceSum InterestMean)
     AccountTypeTmp(KEEP = CustID Type BalanceSum);
 SET AccountTmp;
 Type = COMPRESS(Type);
 IF _TYPE_ = 2 THEN OUTPUT AccountSum;
 ELSE IF _TYPE_ = 3 THEN OUTPUT AccountTypeTmp;
RUN;
```

Finally we need to transpose the data set with the ACCOUNTTYPES to have a one-row-per-customer structure:

```
PROC TRANSPOSE DATA = AccountTypeTmp
               OUT = AccountTypes(DROP = _NAME_ _LABEL_);
 BY CustID;
 VAR BalanceSum;
 ID Type;
RUN;
```

Transforming the LEASING Table

For the leasing data we simply summarize the value and the annual rate per customer with the following statement:

```
PROC MEANS DATA = leasing NWAY NOPRINT;
 CLASS CustID;
 VAR Value AnnualRate;
 OUTPUT OUT = LeasingSum(DROP = _TYPE_
                         RENAME = (_FREQ_ = NrLeasing))
                        SUM(Value) = LeasingValue
                        SUM(AnnualRate) = LeasingAnnualRate;
RUN;
```

Transforming the CALLCENTER Table

We first define the snapshot date for the data in a macro variable.

```
%let snapdate = "01JUN2003"d;
```

Then we aggregate the CALLCENTER data over all customers using PROC FREQ.

```
PROC FREQ DATA = callcenter NOPRINT;
 TABLE CustID / OUT = CallCenterContacts(DROP = Percent RENAME = (Count = Calls));
WHERE datepart(date) < &snapdate;
RUN;
```

Next we calculate the frequencies for the category COMPLAINT.

```
PROC FREQ DATA = callcenter NOPRINT;
 TABLE CustID / OUT = CallCenterComplaints(DROP = Percent RENAME = (Count =
Complaints));
 WHERE Category = 'Complaint' and datepart(date) < &snapdate;
RUN;
```

Transforming the SCORES Table

We create three output data sets, one with the value segment in the actual (snapshot date) month, one with the value segment of one month ago, and one with the value segment in a future month in order to calculate the target variable.

```
DATA ScoreFuture(RENAME = (ValueSegment = FutureValueSegment))
     ScoreActual
     ScoreLastMonth(RENAME = (ValueSegment = LastValueSegment));
 SET Scores;
 DROP Date;
 IF Date = &snapdate THEN OUTPUT ScoreActual;
 ELSE IF Date = INTNX('MONTH',&snapdate,-1) THEN OUTPUT ScoreLastMonth;
 ELSE IF Date = INTNX('MONTH',&snapdate,2) THEN OUTPUT ScoreFuture;
RUN;
```

Notice that we used the INTNX function, which enables us to define the relevant points in time in a very elegant way. In this example the first INTNX function evaluates to 01MAY2003 and the second evaluates to 01AUG2003. We have already explained in the “Considerations for Predictive Modeling” section that this is needed for an offset-month.

Creating the CUSTOMER DATA MART

Finally we create the CUSTOMER DATA MART table by joining all intermediate result tables to the CUSTOMER table.

First we define a macro variable for the snapshot date in order to have a reference point for the calculation of age. (Notice that we repeated this macro variable assignment for didactical purposes here.)

```
%let snapdate = "01JUN2003"d;
```

Next we start defining our CUSTOMER DATA MART talbe. We use ATTRIB statements to have a detailed definition of the data set variables by specifying their order, their format, and their label. Notice that we aligned the FORMAT and LABEL statements for a clearer visual picture.

```
DATA CustomerMart;

ATTRIB /* Customer Baseline */
CustID         FORMAT  = 8.    LABEL = "Customer ID"
Birthdate      FORMAT = DATE9. LABEL = "Date of Birth"
Alter          FORMAT = 8.     LABEL = "Age (years)"
Gender         FORMAT = $6.    LABEL = "Gender"
MaritalStatus  FORMAT = $10.   LABEL = "Marital Status"
Title          FORMAT = $10.   LABEL = "Academic Title"
HasTitle       FORMAT = 8.     LABEL = "Has Title? 0/1"
Branch         FORMAT = $5.    LABEL = "Branch Name"
CustomerSince  FORMAT = DATE9. LABEL = "Customer Start Date"
CustomerMonths FORMAT = 8.     LABEL ="Customer Duration (months)"
;
ATTRIB /* Accounts */
HasAccounts      FORMAT = 8.  LABEL ="Customer has any accounts"
NrAccounts       FORMAT = 8.  LABEL ="Number of Accounts"
BalanceSum       FORMAT = 8.2 LABEL ="All Accounts Balance Sum"
InterestMean     FORMAT = 8.1 LABEL ="Average Interest"
Loan             FORMAT = 8.2 LABEL ="Loan Balance Sum"
SavingAccount    FORMAT = 8.2 LABEL ="Saving Account Balance Sum"
Funds            FORMAT = 8.2 LABEL ="Funds Balance Sum"
LoanPct          FORMAT = 8.2 LABEL ="Loan Balance Proportion"
SavingAccountPct FORMAT = 8.2 LABEL ="Saving Account Balance Proportion"
FundsPct         FORMAT = 8.2 LABEL ="Funds Balance Proportion"
;
ATTRIB /* Leasing */
HasLeasing        FORMAT = 8.  LABEL ="Customer has any leasing contract"
NrLeasing         FORMAT  = 8. LABEL ="Number of leasing contracts"
LeasingValue      FORMAT = 8.2 LABEL ="Totals leasing value"
LeasingAnnualRate FORMAT = 8.2 LABEL ="Total annual leasingrate"
;
ATTRIB /* Call Center */
HasCallCenter FORMAT =8.   LABEL ="Customer has any call center contact"
Calls         FORMAT = 8.  LABEL ="Number of call center contacts"
Complaints    FORMAT = 8.  LABEL ="Number of complaints"
ComplaintPct  FORMAT = 8.2 LABEL ="Percentage of complaints"
;
ATTRIB /* Value Segment */
ValueSegment       FORMAT =$10.  LABEL ="Currently Value Segment"
LastValueSegment   FORMAT =$10.  LABEL ="Last Value Segment"
ChangeValueSegment FORMAT =8.2   LABEL ="Change in Value Segment"
```

```
   Cancel              FORMAT =8.    LABEL ="Customer cancelled"
   ;
```

Next we merge the data set that we created in the previous steps and create logical variables for some tables.

```
   MERGE Customer (IN = InCustomer)
          AccountSum (IN = InAccounts)
          AccountTypes
          LeasingSum (IN = InLeasing)
          CallCenterContacts (IN = InCallCenter)
          CallCenterComplaints
          ScoreFuture
          ScoreActual
          ScoreLastMonth;
    BY CustID;
    IF InCustomer;
```

In the next step we replace missing values with zero (0). These can be causally interpreted as zeros, for example, when transposing account types per customer and a customer does not have a certain account type.

```
    ARRAY vars {*} Calls Complaints LeasingValue LeasingAnnualRate
                   Loan SavingAccount Funds Buildassocsaving NrLeasing;
    DO i = 1 TO dim(vars);
     IF vars{i}=. THEN vars{i}=0;
     END;
    DROP i;
```

Finally we calculate derived variables for the data mart.

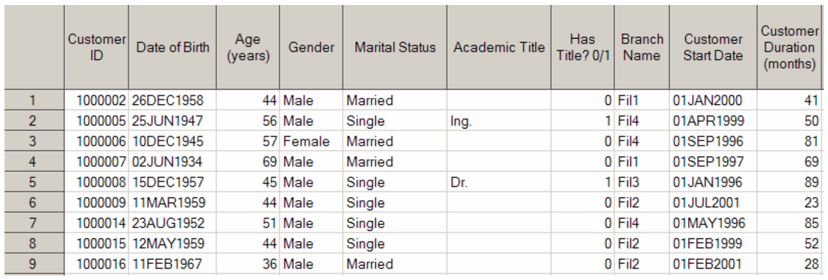
```
   /* Customer Baseline */
   HasTitle = (Title ne "");
   Alter = (&Snapdate-Birthdate)/365.25;
   CustomerMonths = (&Snapdate- CustomerSince)/(365.25/12);
   /* Accounts */
   HasAccounts = InAccounts;
   LoanPct = Loan / BalanceSum * 100;
   SavingAccountPct = SavingAccount / BalanceSum * 100;
   FundsPct = Funds / BalanceSum * 100;
   /* Leasing */
   HasLeasing = InLeasing;
   /* Call Center */
   HasCallCenter = InCallCenter;
   ComplaintPct = Complaints / Calls *100;
   /* Value Segment */
   Cancel = (FutureValueSegment = '8. LOST');
   ChangeValueSegment = (ValueSegment = LastValueSegment);
   RUN;
```

Notice that we calculate the target variable CANCEL based on the fact that a customer has the ValueSegment entry ‘8. LOST’.

When ValueSegment entries change, we simply compare the current and the last ValueSegment entry. It would be more accurate to differentiate between an increase and a decreased ValueSegment entry. This would, however, be more complicated because we would need to consider all segment codes manually. Here it is more accurate to have the underlying numeric score available, which can be used for a simple numeric comparison.

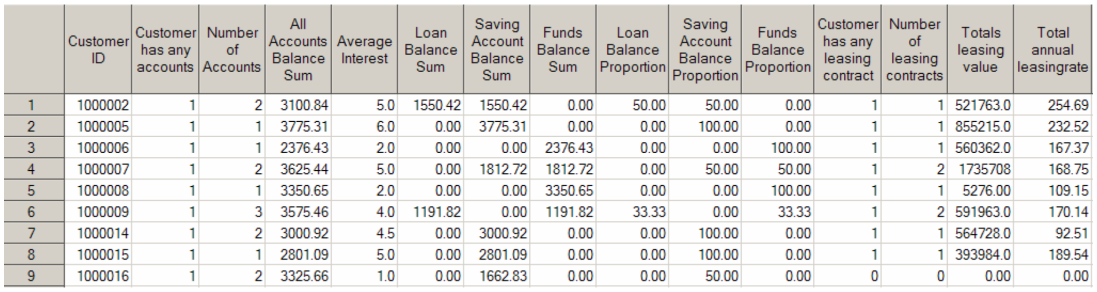
The Resulting Data Set

The resulting data set looks like the following (Tables 11 to 13):



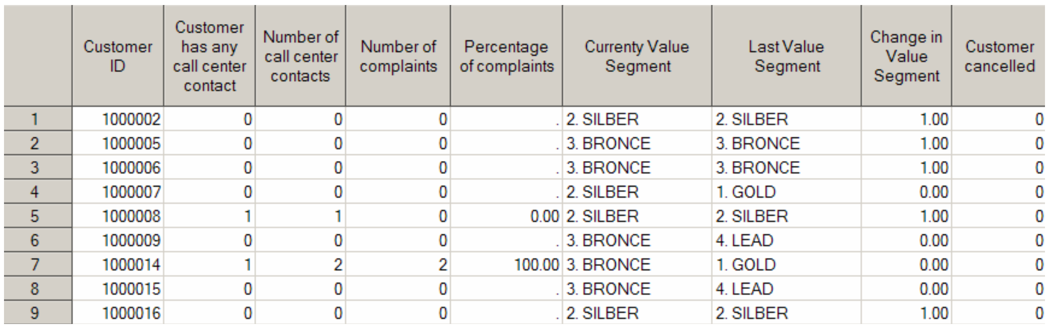
	Customer ID	Date of Birth	Age (years)	Gender	Marital Status	Academic Title	Has Title? 0/1	Branch Name	Customer Start Date	Customer Duration (months)
1	1000002	26DEC1958	44	Male	Married		0	Fil1	01JAN2000	41
2	1000005	25JUN1947	56	Male	Single	Ing.	1	Fil4	01APR1999	50
3	1000006	10DEC1945	57	Female	Married		0	Fil4	01SEP1996	81
4	1000007	02JUN1934	69	Male	Married		0	Fil1	01SEP1997	69
5	1000008	15DEC1957	45	Male	Single	Dr.	1	Fil3	01JAN1996	89
6	1000009	11MAR1959	44	Male	Single		0	Fil2	01JUL2001	23
7	1000014	23AUG1952	51	Male	Single		0	Fil4	01MAY1996	85
8	1000015	12MAY1959	44	Male	Single		0	Fil2	01FEB1999	52
9	1000016	11FEB1967	36	Male	Married		0	Fil2	01FEB2001	28

Table 11. Customer Baseline Data in the CUSTOMER DATA MART



	Customer ID	Customer has any accounts	Number of Accounts	All Accounts Balance Sum	Average Interest	Loan Balance Sum	Saving Account Balance Sum	Funds Balance Sum	Loan Balance Proportion	Saving Account Balance Proportion	Funds Balance Proportion	Customer has any leasing contract	Number of leasing contracts	Totals leasing value	Total annual leasingrate
1	1000002	1	2	3100.84	5.0	1550.42	1550.42	0.00	50.00	50.00	0.00	1	1	521763.0	254.69
2	1000005	1	1	3775.31	6.0	0.00	3775.31	0.00	0.00	100.00	0.00	1	1	855215.0	232.52
3	1000006	1	1	2376.43	2.0	0.00	0.00	2376.43	0.00	0.00	100.00	1	1	560362.0	167.37
4	1000007	1	2	3625.44	5.0	0.00	1812.72	1812.72	0.00	50.00	50.00	1	2	1735708	168.75
5	1000008	1	1	3350.65	2.0	0.00	0.00	3350.65	0.00	0.00	100.00	1	1	5276.00	109.15
6	1000009	1	3	3575.46	4.0	1191.82	0.00	1191.82	33.33	0.00	33.33	1	2	591963.0	170.14
7	1000014	1	2	3000.92	4.5	0.00	3000.92	0.00	0.00	100.00	0.00	1	1	564728.0	92.51
8	1000015	1	1	2801.09	5.0	0.00	2801.09	0.00	0.00	100.00	0.00	1	1	393984.0	189.54
9	1000016	1	2	3325.66	1.0	0.00	1662.83	0.00	0.00	50.00	0.00	0	0	0.00	0.00

Table 12. Account Data and Leasing Data in the CUSTOMER DATA MART



	Customer ID	Customer has any call center contact	Number of call center contacts	Number of complaints	Percentage of complaints	Currenty Value Segment	Last Value Segment	Change in Value Segment	Customer cancelled
1	1000002	0	0	0	.	2. SILBER	2. SILBER	1.00	0
2	1000005	0	0	0	.	3. BRONCE	3. BRONCE	1.00	0
3	1000006	0	0	0	.	3. BRONCE	3. BRONCE	1.00	0
4	1000007	0	0	0	.	2. SILBER	1. GOLD	0.00	0
5	1000008	1	1	0	0.00	2. SILBER	2. SILBER	1.00	0
6	1000009	0	0	0	.	3. BRONCE	4. LEAD	0.00	0
7	1000014	1	2	2	100.00	3. BRONCE	1. GOLD	0.00	0
8	1000015	0	0	0	.	3. BRONCE	4. LEAD	0.00	0
9	1000016	0	0	0	.	2. SILBER	2. SILBER	1.00	0

Table 13. CALLCENTER Data, SCORE Data and the Target Variable in the CUSTOMER DATA MART

THE RESULTS AND THEIR USAGE

We created a data mart that can be used to predict which customers have a high probability to leave the company and to understand the reasons why customers cancel their service. We can perform univariate and multivariate descriptive analysis on the customer attributes. We can develop predictive modeling to determine the likelihood that future customers will cancel.

Notice that we only used data that we found in our five tables. We aggregated and transposed some of the data to a one-row-per-subject structure and joined these tables together. The derived variables that we defined give a good picture of the customer attributes and provide the best possible inference of customer behavior.

CONCLUSION

This paper shows that data preparation is a key success factor for analytical projects. Besides multiple-row-per-subject data marts or longitudinal data marts, the one-row-per-subject data mart is frequently used and is central for statistical and data mining analyses.

In many cases the available source data have a one-to-many relationship to the subject itself. Therefore data needs to be transposed or aggregated. The selection of the transposition and aggregation methods is primarily driven by business considerations of what will give meaningful derived variables.

This paper shows you ways to create a one-row-per-subject data mart from a conceptual and a coding point of view.

Anyone who is interested in more details on *Data Preparation for Analytics* is referred to my book with the same name, which will be published by SAS Press (Book# 60502) in September 2006 (see the “Recommended Reading” section). In this book the business point of view, the concepts, and coding examples for data preparation in analytics is extensively discussed.

RECOMMENDED READING

The author of this paper has written a SAS Press publication titled *Data Preparation for Analytics.* It is currently under review. The book provides a more detailed discussion on the topic of this paper. This book is written for SAS users and anyone involved in the data preparation process for analytics. The focus of the book is to give practical advice in the form of SAS coding tips and tricks, as well as conceptual background on data structures and considerations from the business point of view.

This book will enable the reader to do the following:

- view analytic data preparation in light of its business environment and consider the underlying business questions in data preparation
- learn about data models and relevant analytic data structures like the one-row-per-subject data mart, the multiple-row-per-subject data mart, and the longitudinal data mart
- use powerful SAS macros to change between the various data mart structures
- consider the specifics of predictive modeling for data mart creation
- learn how to create meaningful derived variables for all data mart types
- illustrate how to create a one-row-per-subject data mart from various data sources and data structures
- learn the concepts and considerations when preparing data for time series analysis
- illustrate how scoring can be done with various SAS procedures themselves and with SAS® Enterprise Miner™
- learn about the power of SAS for analytic data preparation and the data mart requirements of various SAS procedures
- benefit from many do’s and don’ts and practical examples for data mart building

CONTACT INFORMATION

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

Gerhard Svolba
SAS Austria
Mariahilfer Str. 116
Vienna, Austria
Email: Gerhard.Svolba@aut.sas.com