Chapter 9: Aggregating Data

Objectives

- Introduce the Aggregate node to create summary records
- Introduce the Set To Flag node to transform a single set field into a collection of flag fields
- Use the Merge node to combine the output from the Aggregate and Set To Flag nodes
- Introduce the Restructure node

Data

In this chapter we use the data file *fulldata.txt*. The file contains one record per account held by customers of a financial organization. The file contains demographic details on the customer and individual account information.

9.1 Introduction

As the previous chapters have illustrated, the general file structure of the data source may not be in the correct format for your analysis.

For example, purchase data is often held in a file containing one record per purchase. To analyze this at a customer level, the data must be restructured so that there is one record per customer, containing summary information. Using the Aggregate node, a new file can be created so that each record contains information such as total amount spent, average amount spent on certain goods, and total number of purchases.

Alternatively, it is often useful to have purchase data arranged so that each field represents a particular product and each record stores whether or not a customer has purchased that product. The Set To Flag node may be used to transform a single field containing a set of products into a collection of product flag fields within a record (or, generically, any symbolic field into a set of multiple fields).

In the following sections we introduce the Aggregate and Set To Flag nodes to perform such data file manipulation. We also introduce the Restructure node to transform each value of a set into a separate field, but this time each field will take on the value from another field. And finally, we will show how aggregation can be used in conjunction with each node.

9.2 Summarizing Data Using the Aggregate Node

It is often necessary to replace data with summary or aggregated information. For example:

- A data file, containing information on individual purchases can be replaced by a file containing summary information on individual customers, such as total amount spent.
- A data file in which a record contains end-of-month account balances for different accounts can be replaced by a file with one record per account containing average end-of-month balances for the year.
- Information held on individuals, such as age, can be summarized into information representing each household, such as average age.

The Aggregate node, located in the Record Ops palette, replaces a collection of input records with a summary, aggregated output record. During aggregation, the overall file structure normally changes because the case or record definition is altered.

During aggregation at least one key field must be specified. The key field defines the group of records that are to be replaced by one record, and multiple key fields may be used. For example, if summarizing purchase data into customer data, the key field will be a unique customer reference number/ID or a combination of fields that are used to uniquely identify each customer.

For each unique combination of the key field(s), values within an aggregate field (fields from which aggregate summaries are calculated) will be replaced by a single summary value. Summaries available include the mean, minimum, maximum, sum, and standard deviation. The aggregate fields must be numeric.

For example, Table 9.1 gives a set of possible input records taken from a personnel database:

Table 9.1 Input Records to the Aggregate Node

Employee Number	Department	Current Age	Initial Salary	Current Salary
10123	Sales	24	17,000	18,000
10124	Sales	29	16,500	20,000
10125	Sales	32	13,000	30,000
10126	Accounts	30	14,000	20,000
10127	Accounts	41	12,000	35,000
10128	Accounts	35	16,000	18,000
10129	R&D	29	10,000	14,000
10130	R&D	37	13,000	17,000
10131	R&D	45	12,500	27,500
10132	Marketing	35	25,000	26,000
10133	Marketing	21	12,000	14,000

Aggregating this information, using department as the key field, with aggregate fields age (with a mean summary), initial salary (with a minimum summary), and current salary (with a sum summary), will produce the data file in Table 9.2.

Table 9.2 Aggregated Data Using Department as the Key Field

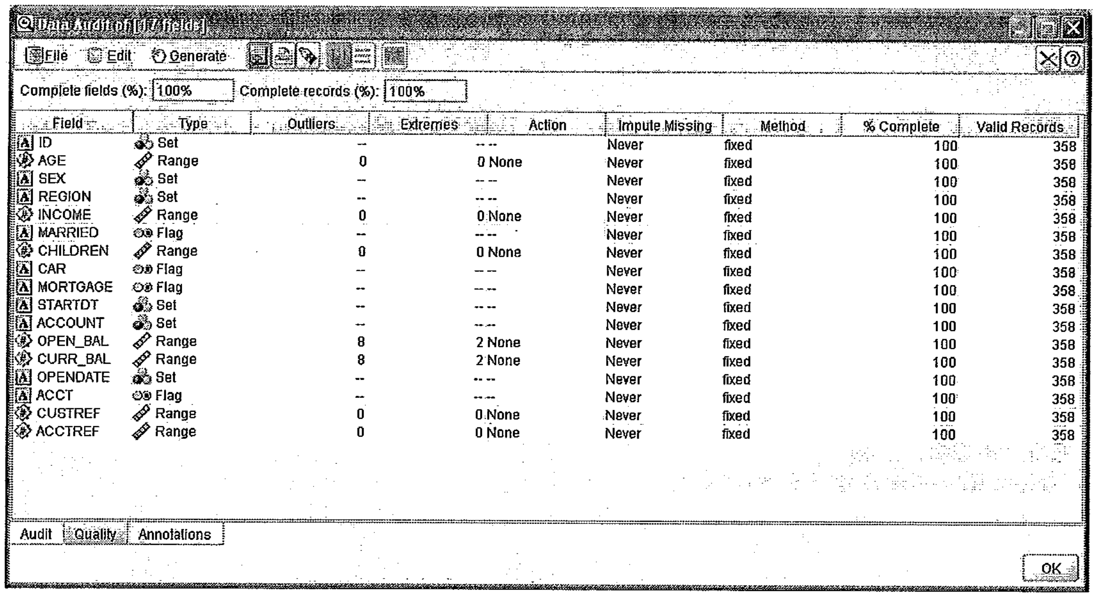
Department	Mean Age	Minimum Initial Salary	Total Current Salary
Sales	28.33	13,000	68,000
Accounts	35.33	12,000	73,000
R&D	37.00	10,000	58,500
Marketing	28.00	12,000	40,000

To illustrate the use of the Aggregate node, we will reduce the data file *fulldata.txt* from a set of records containing account information into a data stream containing one record per customer.

It is best to aggregate data after missing values have been removed or modified. Records with missing values (non-numeric or $null$) on aggregate fields will not contribute to the aggregate summaries for those fields. However, blanks (user defined missing values) do contribute to the aggregate summaries and these values should be replaced with $null$ (you can use the Filler node) before aggregation. We will begin with an existing stream.

Open the Stream file **Aggregate.str**, located in the **c:\Train\ClemIntro** directory
Execute the **Data Audit** node
Click the **Quality** tab

Figure 9.1 Quality Tab in Data Audit Node



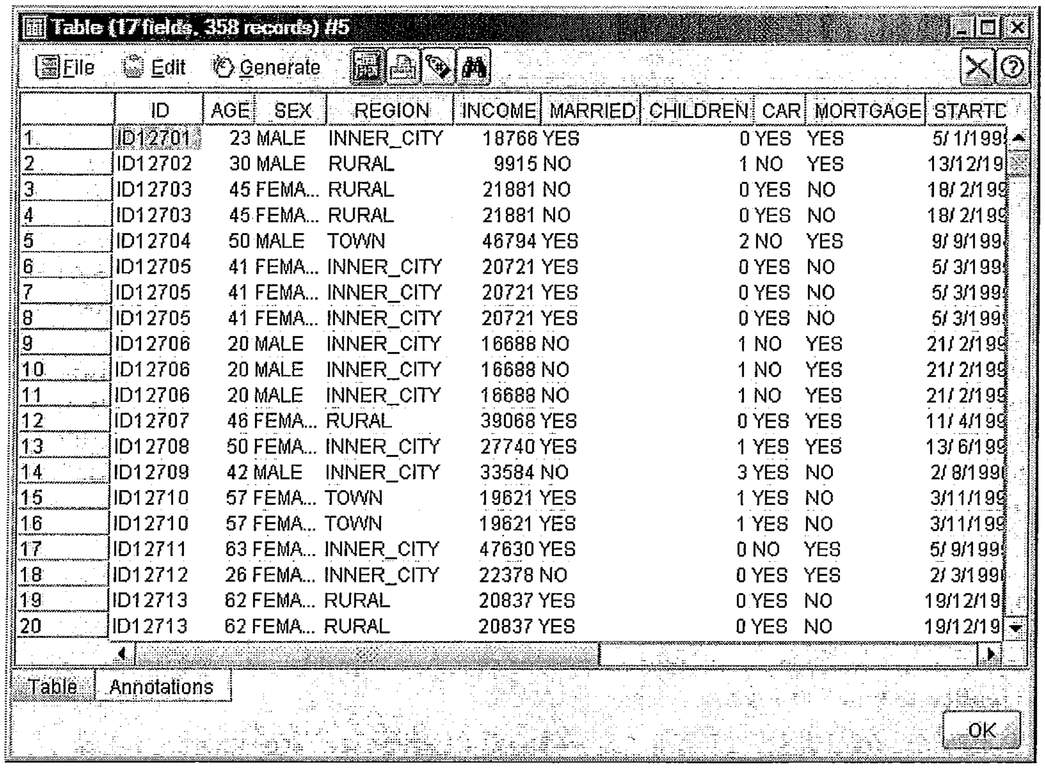
Complete fields (%): 100% Complete records (%): 100%

Field	Type	Outliers	Extremes	Action	Impute Missing	Method	% Complete	Valid Records
ID	Set	--	--	--	Never	fixed	100	358
AGE	Range	0	0	None	Never	fixed	100	358
SEX	Set	--	--	--	Never	fixed	100	358
REGION	Set	--	--	--	Never	fixed	100	358
INCOME	Range	0	0	None	Never	fixed	100	358
MARRIED	Flag	--	--	--	Never	fixed	100	358
CHILDREN	Range	0	0	None	Never	fixed	100	358
CAR	Flag	--	--	--	Never	fixed	100	358
MORTGAGE	Flag	--	--	--	Never	fixed	100	358
STARTDT	Set	--	--	--	Never	fixed	100	358
ACCOUNT	Set	--	--	--	Never	fixed	100	358
OPEN_BAL	Range	8	2	None	Never	fixed	100	358
CURR_BAL	Range	8	2	None	Never	fixed	100	358
OPENDATE	Set	--	--	--	Never	fixed	100	358
ACCT	Flag	--	--	--	Never	fixed	100	358
CUSTREF	Range	0	0	None	Never	fixed	100	358
ACCTREF	Range	0	0	None	Never	fixed	100	358

Note that the data are 100% complete, which means there are no missing data.

Close the Data Audit output window
Execute the **Table** node

Figure 9.2 Transaction Data (fulldata.txt)



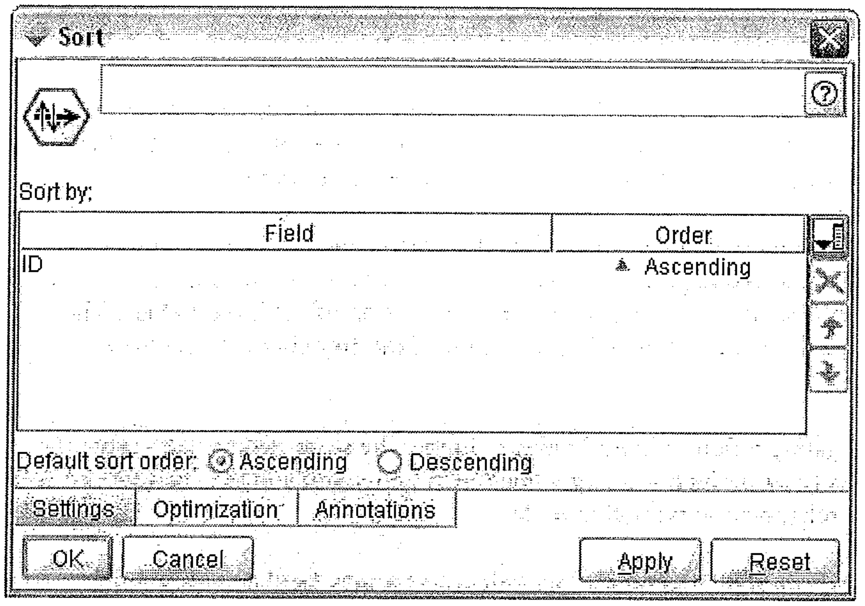
	ID	AGE	SEX	REGION	INCOME	MARRIED	CHILDREN	CAR	MORTGAGE	STARTD
1	ID12701	23	MALE	INNER_CITY	18766	YES	0	YES	YES	5/ 1/199
2	ID12702	30	MALE	RURAL	9915	NO	1	NO	YES	13/12/19
3	ID12703	45	FEMA...	RURAL	21881	NO	0	YES	NO	18/ 2/199
4	ID12703	45	FEMA...	RURAL	21881	NO	0	YES	NO	18/ 2/199
5	ID12704	50	MALE	TOWN	46794	YES	2	NO	YES	9/ 9/199
6	ID12705	41	FEMA...	INNER_CITY	20721	YES	0	YES	NO	5/ 3/199
7	ID12705	41	FEMA...	INNER_CITY	20721	YES	0	YES	NO	5/ 3/199
8	ID12705	41	FEMA...	INNER_CITY	20721	YES	0	YES	NO	5/ 3/199
9	ID12706	20	MALE	INNER_CITY	16688	NO	1	NO	YES	21/ 2/199
10	ID12706	20	MALE	INNER_CITY	16688	NO	1	NO	YES	21/ 2/199
11	ID12706	20	MALE	INNER_CITY	16688	NO	1	NO	YES	21/ 2/199
12	ID12707	46	FEMA...	RURAL	39068	YES	0	YES	YES	11/ 4/199
13	ID12708	50	FEMA...	INNER_CITY	27740	YES	1	YES	YES	13/ 6/199
14	ID12709	42	MALE	INNER_CITY	33584	NO	3	YES	NO	2/ 8/199
15	ID12710	57	FEMA...	TOWN	19621	YES	1	YES	NO	3/11/199
16	ID12710	57	FEMA...	TOWN	19621	YES	1	YES	NO	3/11/199
17	ID12711	63	FEMA...	INNER_CITY	47630	YES	0	NO	YES	5/ 9/199
18	ID12712	26	FEMA...	INNER_CITY	22378	NO	0	YES	YES	2/ 3/199
19	ID12713	62	FEMA...	RURAL	20837	YES	0	YES	NO	19/12/19
20	ID12713	62	FEMA...	RURAL	20837	YES	0	YES	NO	19/12/19

Note that *ID* is not unique, since a single customer can hold multiple accounts. This is a common phenomenon when dealing with transaction data.

We will aggregate the data using *ID* as the key field and request several summaries, including the total amount originally deposited and the number of accounts for each individual. We will also sort the data to make the data aggregation more efficient.

Close the Table window
Place a **Sort** node from the Record Ops palette to the right of the Type node
Connect the **Type** node to the **Sort** node
Edit the **Sort** node
Select **ID** in the **Sort by:** field box

Figure 9.3 Sort Node Dialog



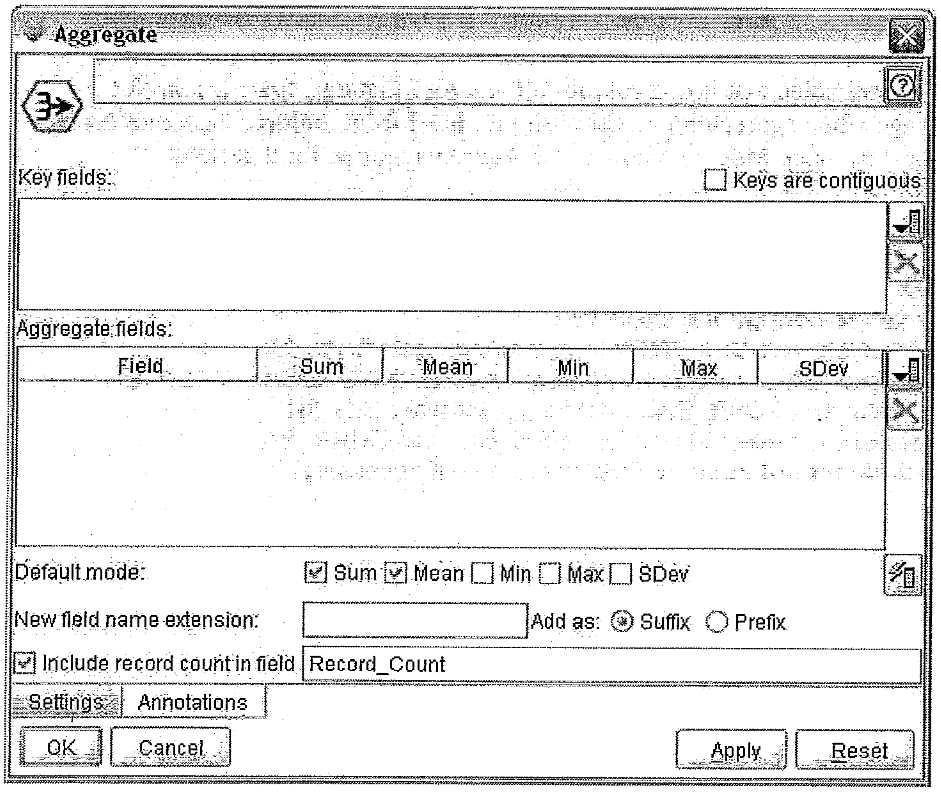
Sort by:

Field	Order
ID	Ascending

Default sort order: [✓ Ascending] Descending

Click **OK** to return to the Stream Canvas
Place an **Aggregate** node from the Record Ops palette to the right of the **Sort** node
Connect the **Sort** node to the **Aggregate** node
Edit the **Aggregate** node

Figure 9.4 Aggregate Node Dialog



Key fields: ☐ Keys are contiguous

Aggregate fields:

Field	Sum	Mean	Min	Max	SDev

Default mode: ☑ Sum ☑ Mean ☐ Min ☐ Max ☐ SDev

New field name extension: Add as: [✓ Suffix] Prefix

☑ Include record count in field Record_Count

As shown in the dialog box in Figure 9.4, the *Key fields* box is used to specify the key(s) for aggregation. If you select multiple key fields, the values of the fields are combined to produce a key value.

If the *Keys are contiguous* check box is selected, values for the key fields will only be treated as equal if they occur in adjacent records. If the data stream is already sorted in order of the key fields, checking *Keys are contiguous* will make the aggregation more efficient.

Fields whose values are to be aggregated are selected on the *Aggregate fields* list. To create aggregate summaries for the selected aggregate field, check one or more of the aggregate mode check boxes. Summaries will be created for each unique combination of key field values. The five available summaries are the Sum, Mean, Min (minimum), Max (maximum), and Sdev (standard deviation).

As a reminder, when aggregating a field, if a null value is found, the aggregation will ignore the null value. However, blanks (user defined missing values) will be included in the calculations and so should be excluded beforehand or converted to nulls.

Checking the *Include record count in field* check box will create a new field containing the number of records aggregated to form each output record. This field will be named *Record_Count* by default, but the name can be changed.

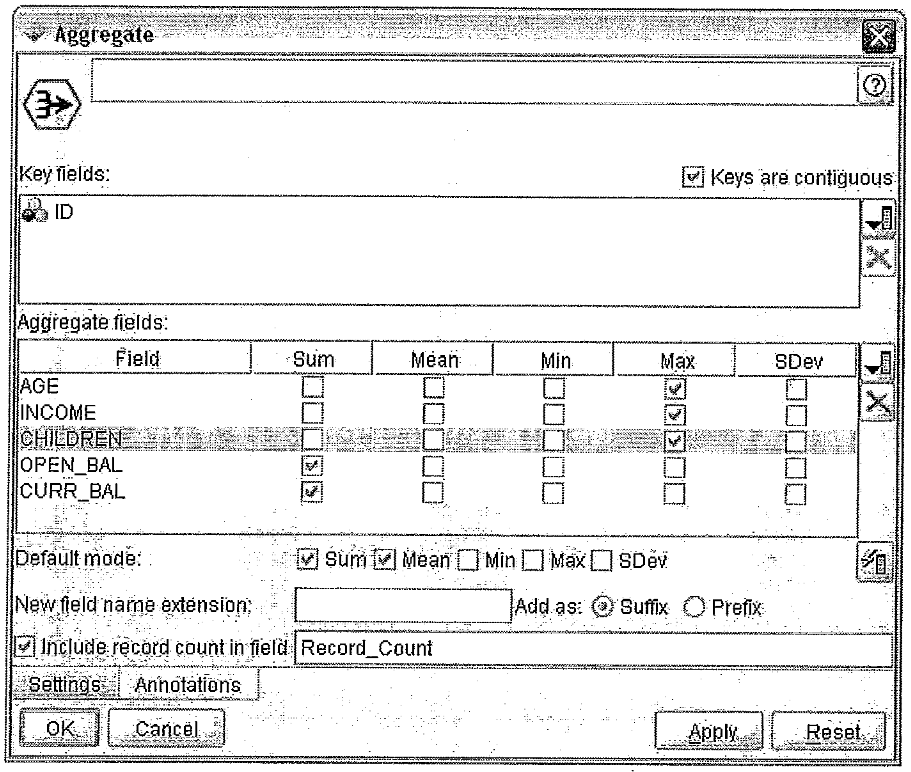
On execution, the Aggregate node reads all input records, extracting summary information. Each aggregate record is created with the aggregation key fields and summary values stored in new fields. Each new aggregate field is assigned a name that consists of the aggregate mode name appended (by default) to the original field name (for example, *BALANCE_Sum*). These aggregate records are then passed downstream in no defined order.

If you wish to retain a field value that is constant for all records in an aggregate group (for example, customer's age when aggregating to the customer level from multiple customer account records), simply request the Min, Max, or Mean as the Aggregate mode for that field.

To obtain records representing ID-level aggregates:

Place **ID** in the **Key fields** list
Check the **Keys are contiguous** check box
Select **AGE**, **INCOME** and **CHILDREN** from the Aggregate fields: list
Check **Max** statistic (deselect **Sum** and **Mean**) for AGE, INCOME and CHILDREN
Select **OPEN_BAL** and **CURR_BAL** from the Aggregate fields: list
Check **Sum** statistic (deselect **Mean**) for OPEN_BAL and CURR_BAL
Check the **Include record count in field** check box (if necessary)

Figure 9.5 Completed Aggregate Node Dialog



Key fields: [✓] Keys are contiguous

ID

Aggregate fields:

Field	Sum	Mean	Min	Max	SDev
AGE	☐	☐	☐	✓	☐
INCOME	☐	☐	☐	✓	☐
CHILDREN	☐	☐	☐	✓	☐
OPEN_BAL	✓	☐	☐	☐	☐
CURR_BAL	✓	☐	☐	☐	☐

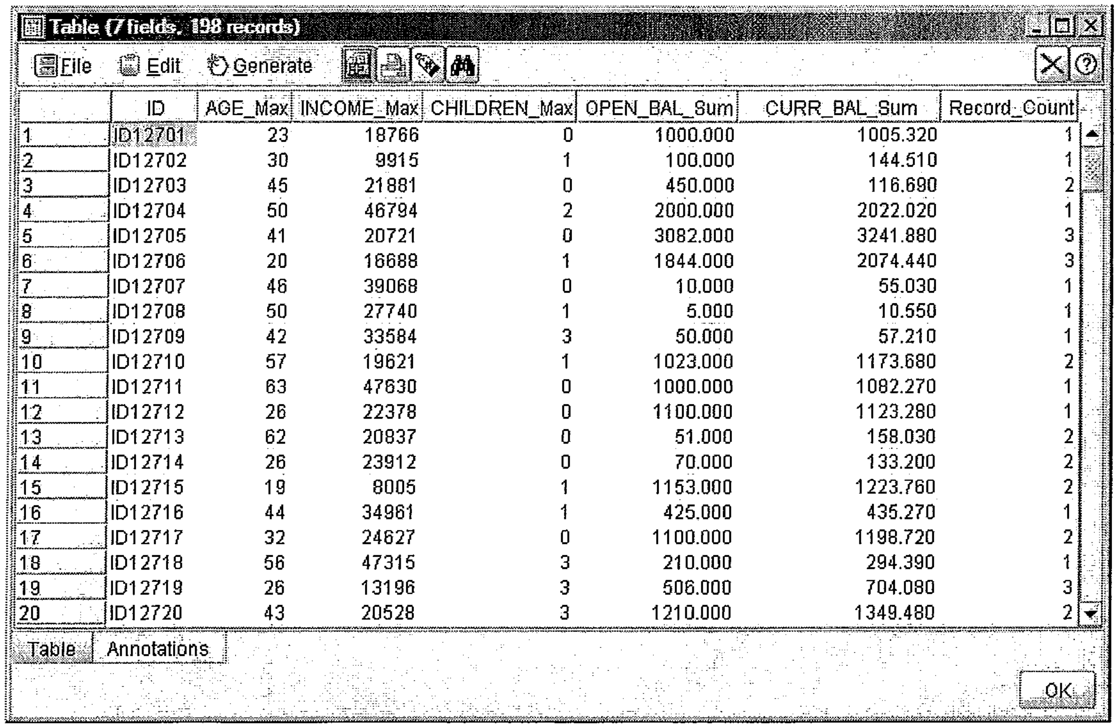
Default mode: [✓] Sum [✓] Mean ☐ Min ☐ Max ☐ SDev

New field name extension: Add as: [✓ Suffix] Prefix

[✓] Include record count in field Record_Count

Click **OK** to return to the Stream Canvas
Connect the **Aggregate** node to a **Table** node
Execute the **Table** node

Figure 9.6 Aggregated Data Containing One Record per Customer



	ID	AGE_Max	INCOME_Max	CHILDREN_Max	OPEN_BAL_Sum	CURR_BAL_Sum	Record_Count
1	ID12701	23	18766	0	1000.000	1005.320	1
2	ID12702	30	9915	1	100.000	144.510	1
3	ID12703	45	21881	0	450.000	116.690	2
4	ID12704	50	46794	2	2000.000	2022.020	1
5	ID12705	41	20721	0	3082.000	3241.880	3
6	ID12706	20	16688	1	1844.000	2074.440	3
7	ID12707	46	39068	0	10.000	55.030	1
8	ID12708	50	27740	1	5.000	10.550	1
9	ID12709	42	33584	3	50.000	57.210	1
10	ID12710	57	19621	1	1023.000	1173.680	2
11	ID12711	63	47630	0	1000.000	1082.270	1
12	ID12712	26	22378	0	1100.000	1123.280	1
13	ID12713	62	20837	0	51.000	158.030	2
14	ID12714	26	23912	0	70.000	133.200	2
15	ID12715	19	8005	1	1153.000	1223.760	2
16	ID12716	44	34961	1	425.000	435.270	1
17	ID12717	32	24627	0	1100.000	1198.720	2
18	ID12718	56	47315	3	210.000	294.390	1
19	ID12719	26	13196	3	506.000	704.080	3
20	ID12720	43	20528	3	1210.000	1349.480	2

The data have now been restructured so that one record contains the age, income, and number of children for each customer. The sums of the opening and current balance are given, together with the total number of accounts the customer holds. The name of the aggregate mode (summary) function applied to each aggregate field has been appended to the original field name (for example, *AGE_Max*). These names can be changed with a Filter node. Fields in the original data stream that were neither key nor aggregate fields have been dropped from the stream.

These data can now be used for modeling and analysis, or the aggregate data can be merged back into the original file, as we show below.

One limitation of the Aggregate node is that aggregate fields must be numeric. If, for example, the stream contains a field defining purchase type that is in a symbolic format, this information will be dropped when performing aggregation. In the next section we will introduce the Set To Flag node as a method of restructuring data held in a symbolic format.

9.3 Restructuring Set Fields Using the Set To Flag Node

It may be necessary to convert information held in a set field into a collection of flag fields. For example:

- A file containing one record per purchase may need to be analyzed at a customer level. A field containing a product identifier can be expanded across a number of flag fields indicating whether or not each product was purchased.

- To use a set field in regression or discriminant analysis, flags based on set members (dummy variables) must be used as inputs, as opposed to the original set field.

The Set To Flag node located in the Field Ops palette generates flag fields based on set members. On execution, each record received is checked for each value of the selected set field. If found, the appropriate flag field is set to true. The record is then passed down the stream.

The Set To Flag node has an optional aggregate setting that groups the records according to aggregate key field(s). Those flags with at least one true value within the group are set to true, and all others are set to false.

For example, if the data shown in Table 9.3 were passed through a Set To Flag node, with the product field as the set field, Table 9.4 shows the resulting data if aggregating were not used, while Table 9.5 shows the resulting data if aggregating were used.

Table 9.3 Input Data to Set To Flag Node

ID	AGE	PRODUCT
101	24	Bread
101	24	Milk
101	24	Cheese
102	32	Bread
102	32	Fruit
103	44	Milk

Table 9.4 Output Data from Set To Flag Node Without Aggregate

ID	AGE	PRODUCT	Bread	Milk	Cheese	Fruit
101	24	Bread	T	F	F	F
101	24	Milk	F	T	F	F
101	24	Cheese	F	F	T	F
102	32	Bread	T	F	F	F
102	32	Fruit	F	F	F	T
103	44	Milk	F	T	F	F

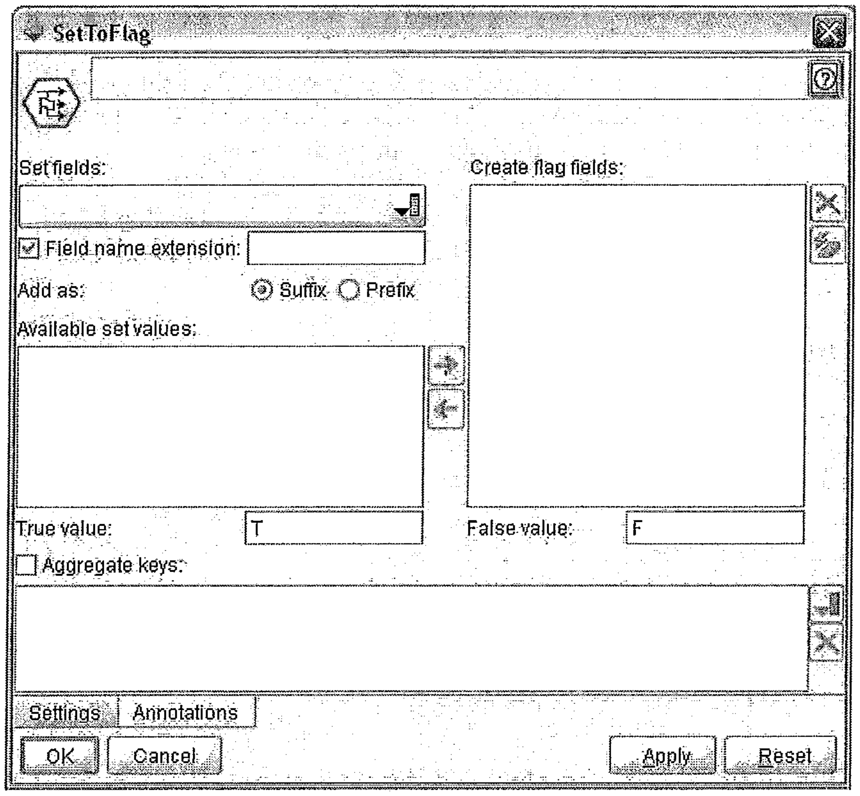
Table 9.5 Output Data from Set To Flag Node With Aggregation

ID	Bread	Milk	Cheese	Fruit
101	T	T	T	F
102	T	F	F	T
103	F	T	F	F

We will first demonstrate the Set To Flag node without aggregation, using the set field *ACCOUNT*, which contains four different account types, *CURRENT*, *ISA*, *PEP* and *SAVE*.

Close the Table window
Place the **Set To Flag** node from the Field Ops palette to the right of the **Sort** node
Connect the **Sort** node to the **Set To Flag** node
Edit the **Set To Flag** node

Figure 9.7 Set To Flag Dialog



The field to be expanded into a number of flag fields is selected from the *Set fields* list. The *Field name extension* text box allows you to specify an extension that will be added as a suffix or prefix to the new flag field names. By default, new field names are automatically created by combining the original field name with the field value as a suffix; for example, *ACCOUNT_CURRENT*, *ACCOUNT_ISA*, etc. When a field is selected, its members will appear in the *Available set values* list box.

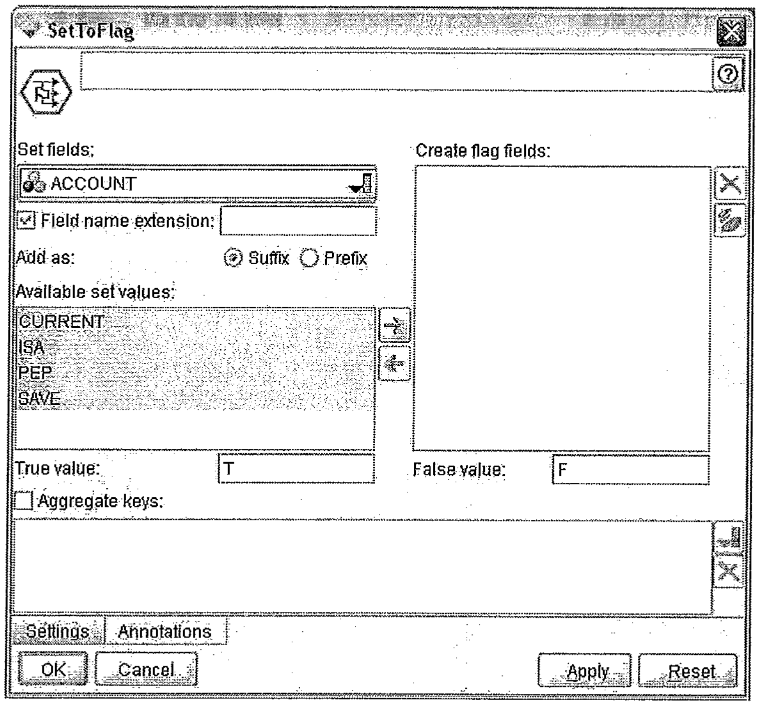
You then place the members for which you wish to create new flag fields in the *Create flag fields* list box. Not all possible flags need be created.

By default, if the set value is present in a record, the corresponding flag field will be set to the True value (T), otherwise it will be set to the False value (F). These can be changed if desired.

As explained earlier, output records can be aggregated by checking the *Aggregate keys* check box and selecting the appropriate key(s).

Select **ACCOUNT** in the **Set fields:** list. The list of values will appear in the **Available set values** list (Note that all values listed in the Type node are included, whether they exist in the current data file or not)

Figure 9.8 Set To Flag Dialog (Displaying Set Values for ACCOUNT)


SetToFlag

Set fields: ACCOUNT

[✓] Field name extension:

Add as: [✓ Suffix] Prefix

Available set values:
CURRENT
ISA
PEP
SAVE

Create flag fields:

True value: T

False value: F

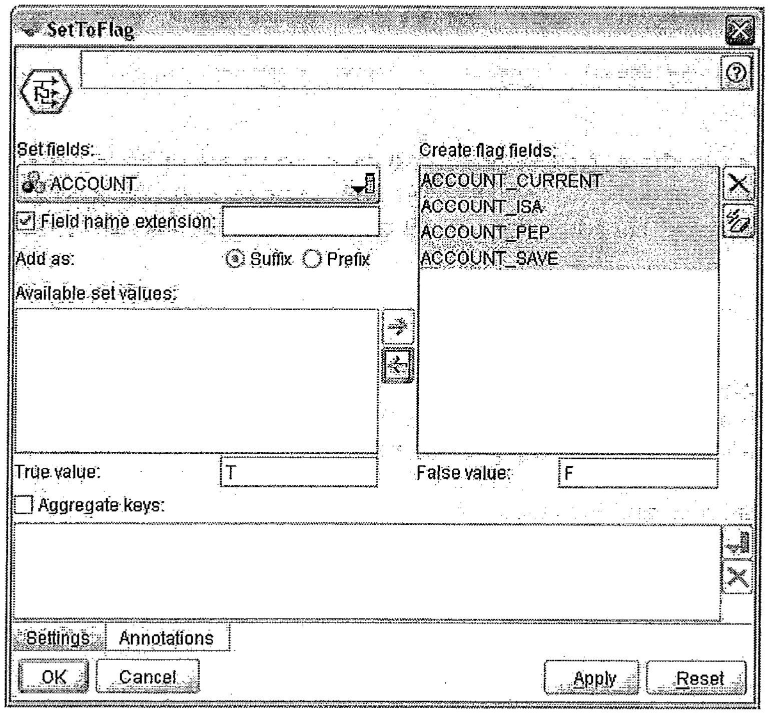
[ ] Aggregate keys:

Settings Annotations

OK Cancel Apply Reset

Click the button

Figure 9.9 Completed Set To Flag Dialog (Without Aggregation)


SetToFlag

Set fields: ACCOUNT

[✓] Field name extension:

Add as: [✓ Suffix] Prefix

Available set values:

Create flag fields:
ACCOUNT_CURRENT
ACCOUNT_ISA
ACCOUNT_PEP
ACCOUNT_SAVE

True value: T

False value: F

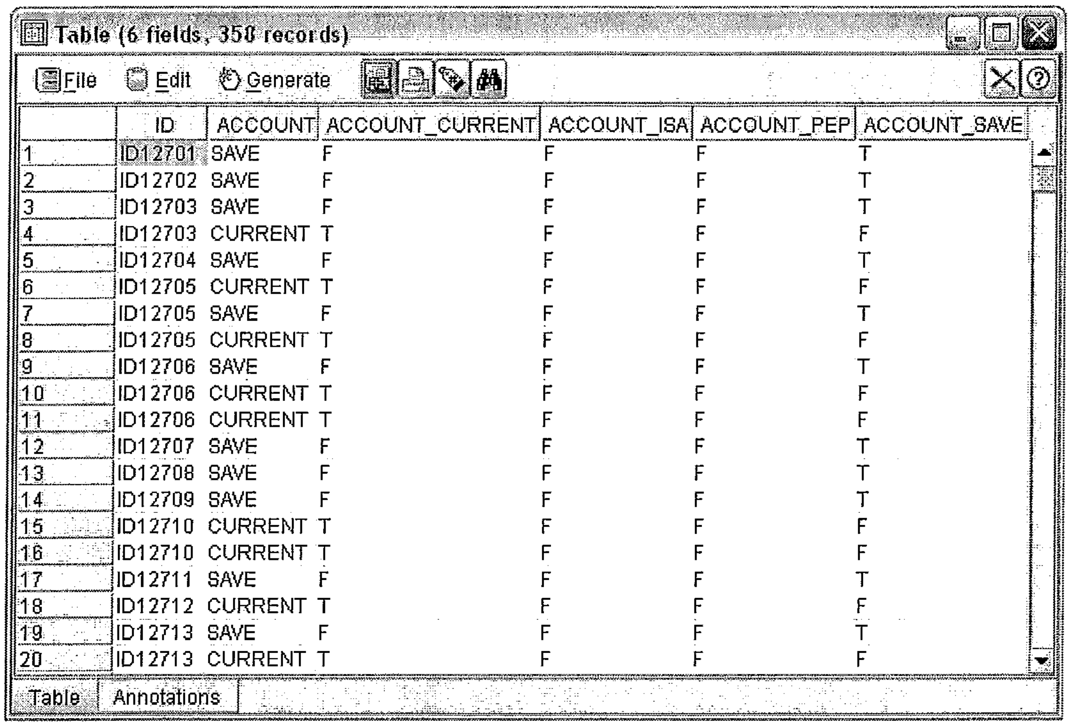
[ ] Aggregate keys:

Settings Annotations

OK Cancel Apply Reset

Click **OK** to return to the Stream Canvas
Connect the **Set to Flag** node to a **Filter** node
Edit the **Filter** node and filter all fields, except **ID**, **ACCOUNT**, and
ACCOUNT_CURRENT to **ACCOUNT_SAVE** (not shown)
Click **OK**
Connect the **Filter** node to a **Table** node
Execute the **Table** node

Figure 9.10 Data Showing Flag Fields Created by Set To Flag Node Without Aggregation

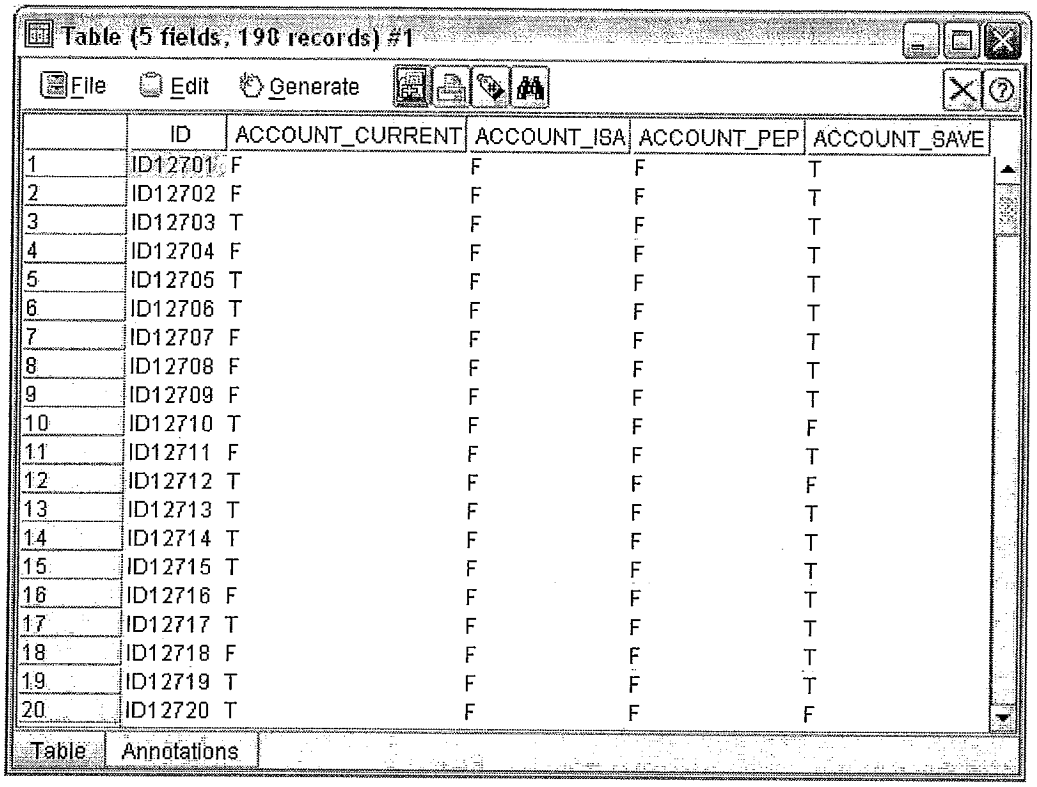


	ID	ACCOUNT	ACCOUNT_CURRENT	ACCOUNT_ISA	ACCOUNT_PEP	ACCOUNT_SAVE
1	ID12701	SAVE	F	F	F	T
2	ID12702	SAVE	F	F	F	T
3	ID12703	SAVE	F	F	F	T
4	ID12703	CURRENT	T	F	F	F
5	ID12704	SAVE	F	F	F	T
6	ID12705	CURRENT	T	F	F	F
7	ID12705	SAVE	F	F	F	T
8	ID12705	CURRENT	T	F	F	F
9	ID12706	SAVE	F	F	F	T
10	ID12706	CURRENT	T	F	F	F
11	ID12706	CURRENT	T	F	F	F
12	ID12707	SAVE	F	F	F	T
13	ID12708	SAVE	F	F	F	T
14	ID12709	SAVE	F	F	F	T
15	ID12710	CURRENT	T	F	F	F
16	ID12710	CURRENT	T	F	F	F
17	ID12711	SAVE	F	F	F	T
18	ID12712	CURRENT	T	F	F	F
19	ID12713	SAVE	F	F	F	T
20	ID12713	CURRENT	T	F	F	F

Four flag fields were created. In this file, the *ACCOUNT* field only takes on values of SAVE and CURRENT, so only the flags representing those values are coded T for some records. We now demonstrate the effect of setting the Aggregate option by returning to the dialog box of the Set To Flag node.

Close the Table window
Edit the **Set To Flag** node
Click the **Aggregate keys** check box (not shown)
Specify **ID** in the **Aggregate keys** field box, and then click **OK**
Connect a **Sort** node between the **Set To Flag** and **Filter** nodes
Edit the **Sort** node
Select **ID** in the **Sort by:** field box, and then click **OK**
Execute the **Table** node

Figure 9.11 Data Showing Flag Fields Created by Set To Flag Node With Aggregation



	ID	ACCOUNT_CURRENT	ACCOUNT_ISA	ACCOUNT_PEP	ACCOUNT_SAVE
1	ID12701	F	F	F	T
2	ID12702	F	F	F	T
3	ID12703	T	F	F	T
4	ID12704	F	F	F	T
5	ID12705	T	F	F	T
6	ID12706	T	F	F	T
7	ID12707	F	F	F	T
8	ID12708	F	F	F	T
9	ID12709	F	F	F	T
10	ID12710	T	F	F	F
11	ID12711	F	F	F	T
12	ID12712	T	F	F	F
13	ID12713	T	F	F	T
14	ID12714	T	F	F	T
15	ID12715	T	F	F	T
16	ID12716	F	F	F	T
17	ID12717	T	F	F	T
18	ID12718	F	F	F	T
19	ID12719	T	F	F	T
20	ID12720	T	F	F	F

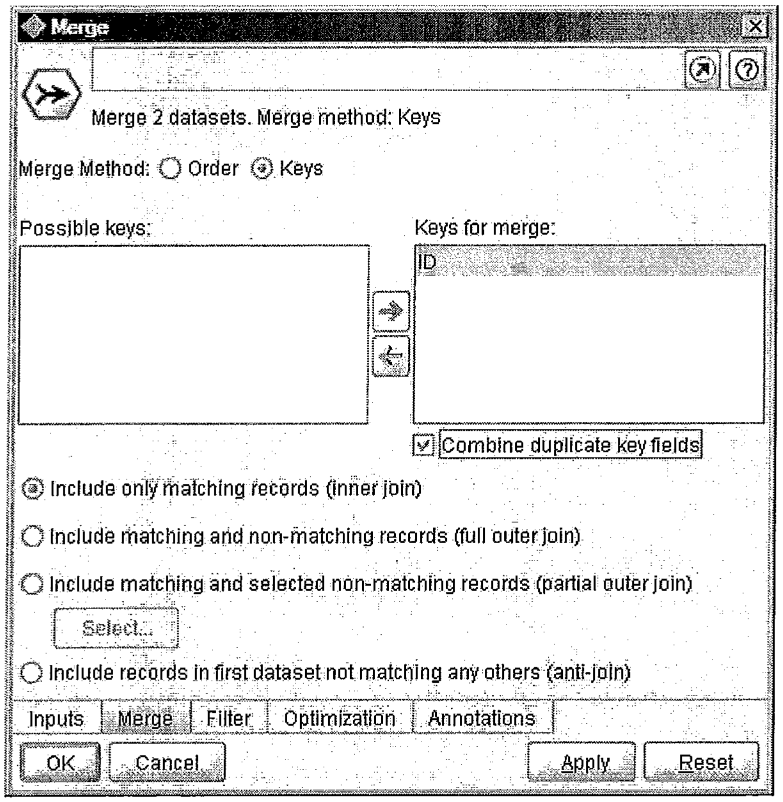
Now there is only one record per customer. One limitation of using the Aggregate option within the Set To Flag node is that all other fields are dropped from the output stream. If you wish to have the data in this case structure, but also have other fields of interest available to analyze, then the branches of the stream created in the previous two sections must be merged using the Merge node.

9.4 Combining Aggregation and Set To Flag Output

We now demonstrate how the two previous data manipulation exercises may be combined to form a data file that contains not only the output fields from the Set To Flag node with aggregation, but also summary information for the aggregated records.

Close the Table window
Place a **Merge** node from the Record Ops Palette on the Stream Canvas
Connect the **Aggregate** node to the **Merge** node
Connect the **Set To Flag** node to the **Merge** node
Edit the **Merge** node
Set the Merge Method to **Keys**
Select **ID** from the **Possible keys** list and move it to **Keys for merge:**

Figure 9.12 Merge Specifications



Merge 2 datasets. Merge method: Keys

Merge Method: Order [✓ Keys]

Possible keys:

Keys for merge: ID

[✓ Combine duplicate key fields]

[✓ Include only matching records (inner join)]

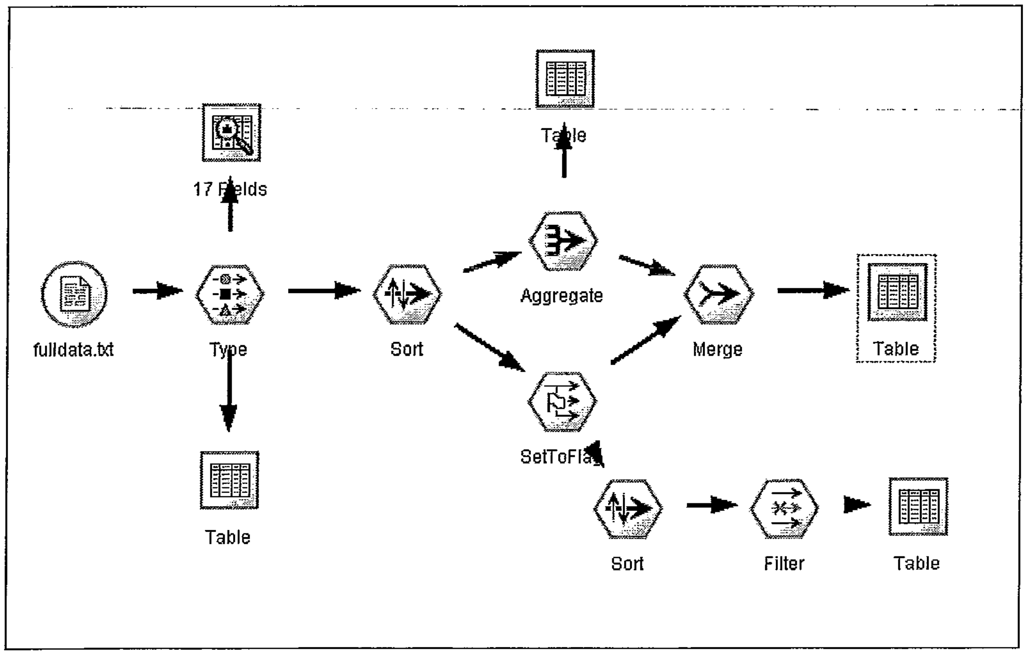
Include matching and non-matching records (full outer join)

Include matching and selected non-matching records (partial outer join)

Include records in first dataset not matching any others (anti-join)

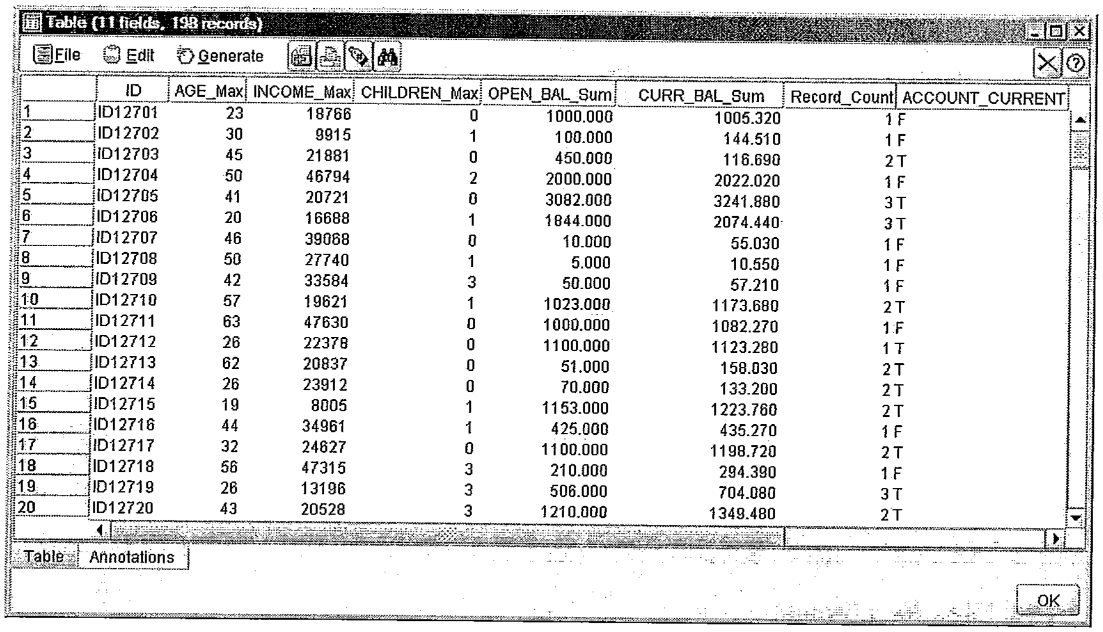
Click **OK**
Connect the **Merge** node to a **Table** node

Figure 9.13 Completed Stream (Merging Aggregate and Set To Flag Results)



Execute the **Table** node

Figure 9.14 Output Data after Aggregate and Set To Flag Streams Merged



	ID	AGE_Max	INCOME_Max	CHILDREN_Max	OPEN_BAL_Sum	CURR_BAL_Sum	Record_Count	ACCOUNT_CURRENT
1	ID12701	23	18766	0	1000.000	1005.320	1	F
2	ID12702	30	9915	1	100.000	144.510	1	F
3	ID12703	45	21881	0	450.000	116.690	2	T
4	ID12704	50	46794	2	2000.000	2022.020	1	F
5	ID12705	41	20721	0	3082.000	3241.880	3	T
6	ID12706	20	16688	1	1844.000	2074.440	3	T
7	ID12707	46	39068	0	10.000	55.030	1	F
8	ID12708	50	27740	1	5.000	10.550	1	F
9	ID12709	42	33584	3	50.000	57.210	1	F
10	ID12710	57	19621	1	1023.000	1173.680	2	T
11	ID12711	63	47630	0	1000.000	1082.270	1	F
12	ID12712	26	22378	0	1100.000	1123.280	1	T
13	ID12713	62	20837	0	51.000	158.030	2	T
14	ID12714	26	23912	0	70.000	133.200	2	T
15	ID12715	19	8005	1	1153.000	1223.760	2	T
16	ID12716	44	34961	1	425.000	435.270	1	F
17	ID12717	32	24627	0	1100.000	1198.720	2	T
18	ID12718	56	47315	3	210.000	294.390	1	F
19	ID12719	26	13196	3	506.000	704.080	3	T
20	ID12720	43	20528	3	1210.000	1349.480	2	T

9.5 Restructuring Data Using the Restructure Node

The Restructure node functions in much the same way as the Set To Flag node except that it is not limited to just creating flags; you can also create numeric fields using values from other fields. Also, Restructure can be used to restructure Flag as well as Set fields. However, unlike with Set To Flag, you cannot restructure your fields and aggregate all in one step. This is because the Restructure node does not have its own aggregation option. For that reason, Set To Flag node may be more convenient than Restructure if you are creating Flag fields.

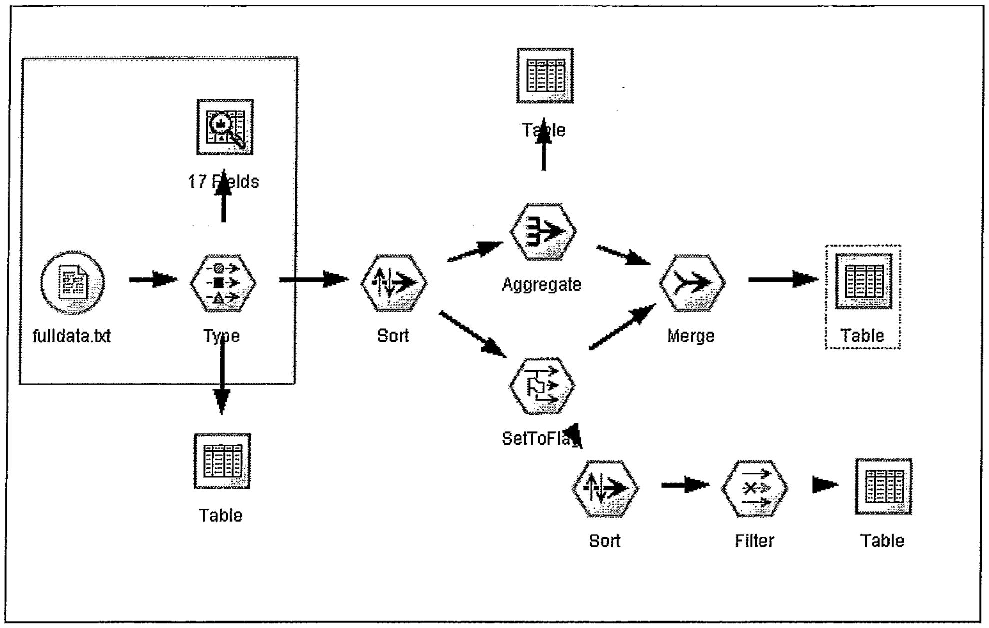
In this example, we will create a field for two of the accounts, *CURRENT* and *SAVE*, but instead of creating flag fields to indicate whether or not a person has these accounts, we will use the value from *CURR_BAL* to show how much money they have in each account.

We will use a subset of the data that has already been modified with the History node so that each customer does not have the same number of monthly balances. One limitation of the History node is that you can specify only one span and offset value, which would not work in this situation because you would end up spanning across observations. We will restructure the file so that the monthly balance for each individual is stored on the same record.

We will also use some of the nodes from the existing stream since we will use the same data file.

Close the Table window
Use the mouse to **draw a rectangle** around the **three nodes** shown in Figure 9.15 to select them
Click **Edit...Copy**

Figure 9.15 Nodes to Select for Copy Operation

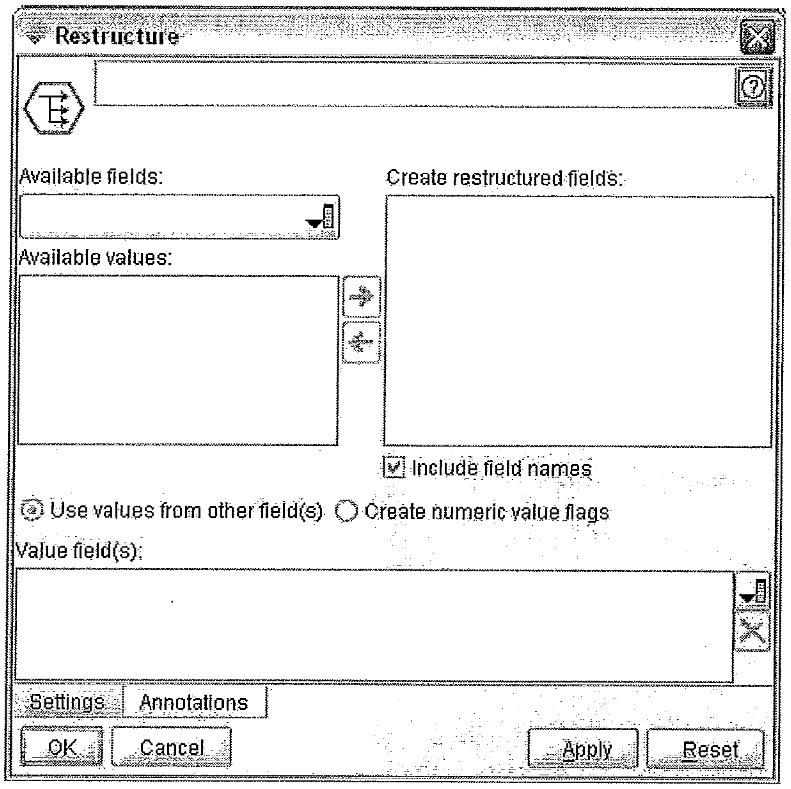


Click **File...New Stream**
Click **Edit...Paste**

These actions will place the three nodes into a new stream.

Place a **Restructure** node from the Field ops palette to the right of the Type node
Connect the **Type** node to the **Restructure** node
Edit the **Restructure** node

Figure 9.16 Restructure Dialog

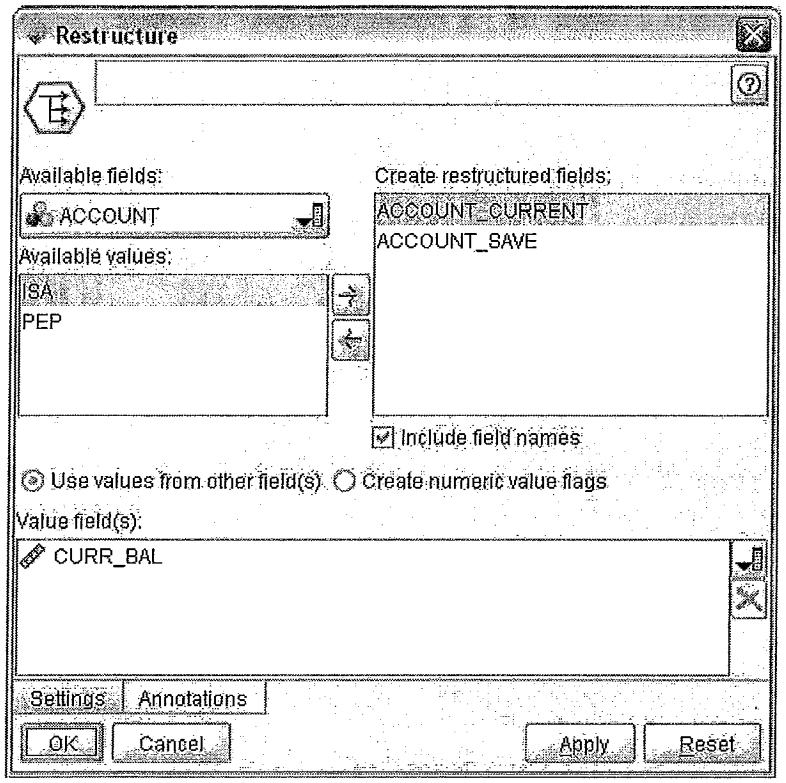


The *Available fields:* option lists all the fields that are either *Set* or *Flag.* When a field is selected, the values will be displayed in the *Available values:* box. You can create separate fields for a subset of these values or all of them by moving them into the *Create restructured fields* box. Note that the data must be fully instantiated using an upstream Type node before you can see a list of the available fields and their values. The *Include field names* box should be checked if you want to include the original field name as a prefix for the new fields.

The Create *numeric value flags* option should be checked if you want to create a numeric Flag for each field (0 for false and 1 for true) to indicate in this instance whether or not the person had a current balance that particular month. The *Use values from other field(s)* option should be used instead if you want the person's balance for that month to become the value of the newly restructured field.

Select **ACCOUNT** from the **Available fields:** list. The list of values will appear in the Available values: box
Move **CURRENT** and **SAVE** into the **Create restructured fields** box
Check to be sure that the **Use values from other field(s)** box is checked
Select **CURR_BAL** from the **Value field(s)** list

Figure 9.17 Completed Restructure Dialog

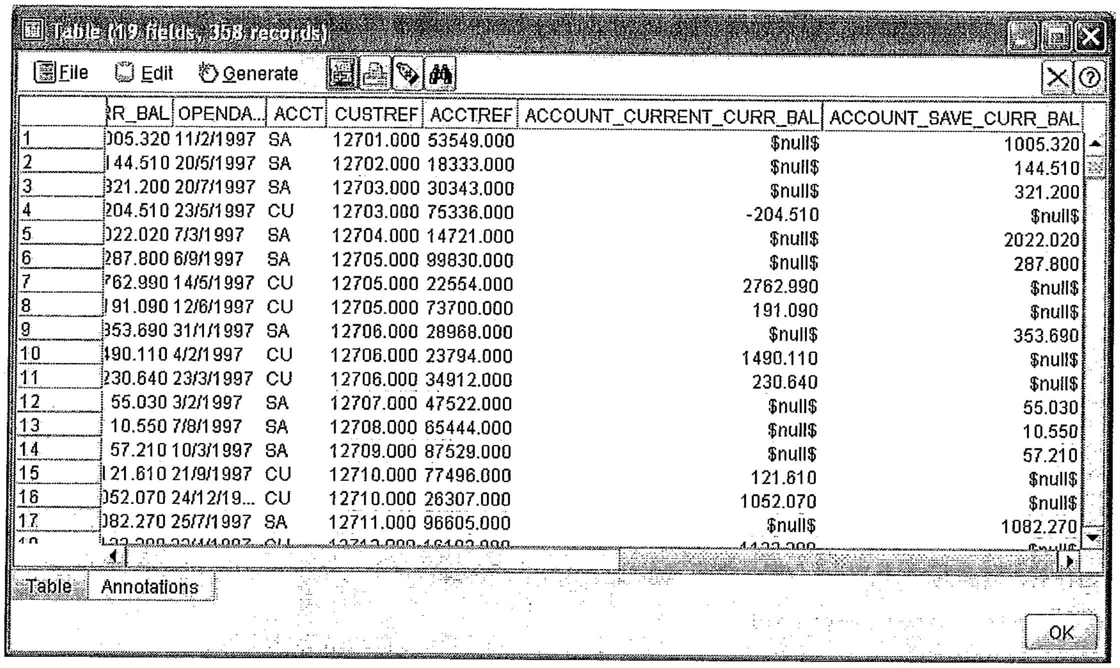


Click **OK**

To see how the Restructure node operates we will look at the data in a Table downstream from this node.

Place a **Table** node on the stream to the right of the **Restructure** node
Connect the **Restructure** node to the **Table** node
Execute the **Table** node, then scroll to the right to the new fields

Figure 9.18 Two New Variables Created by the Restructure Node



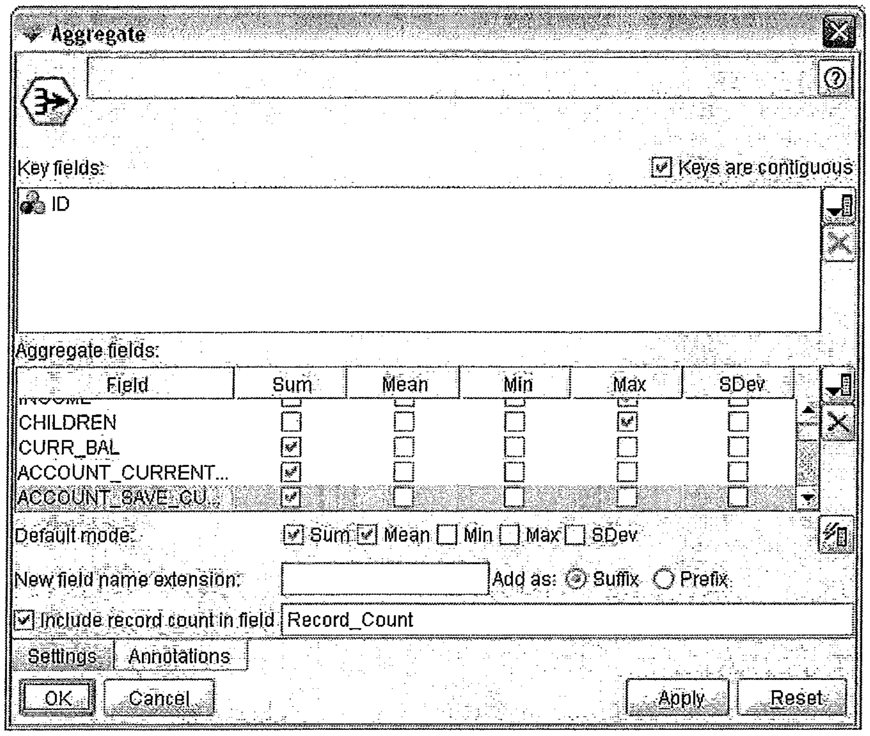
	R_BAL	OPENDA...	ACCT	CUSTREF	ACCTREF	ACCOUNT_CURRENT_CURR_BAL	ACCOUNT_SAVE_CURR_BAL
1	005.320	11/2/1997	SA	12701.000	53549.000	$null$	1005.320
2	144.510	20/5/1997	SA	12702.000	18333.000	$null$	144.510
3	321.200	20/7/1997	SA	12703.000	30343.000	$null$	321.200
4	204.510	23/5/1997	CU	12703.000	75336.000	-204.510	$null$
5	022.020	7/3/1997	SA	12704.000	14721.000	$null$	2022.020
6	287.800	6/9/1997	SA	12705.000	99830.000	$null$	287.800
7	762.990	14/5/1997	CU	12705.000	22554.000	2762.990	$null$
8	191.090	12/6/1997	CU	12705.000	73700.000	191.090	$null$
9	353.690	31/1/1997	SA	12706.000	28968.000	$null$	353.690
10	490.110	4/2/1997	CU	12706.000	23794.000	1490.110	$null$
11	230.640	23/3/1997	CU	12706.000	34912.000	230.640	$null$
12	55.030	3/2/1997	SA	12707.000	47522.000	$null$	55.030
13	10.550	7/8/1997	SA	12708.000	65444.000	$null$	10.550
14	57.210	10/3/1997	SA	12709.000	87529.000	$null$	57.210
15	121.610	21/9/1997	CU	12710.000	77496.000	121.610	$null$
16	052.070	24/12/19...	CU	12710.000	26307.000	1052.070	$null$
17	082.270	25/7/1997	SA	12711.000	96605.000	$null$	1082.270

The Restructure node has created two new fields because there were two values ("CURRENT" and "SAVE") of *ACCOUNT* that we specified. Their names are a combination of the variables and values used to create the field values. In each of these new fields, the value of *CURR_BAL* is placed in the appropriate field, depending on whether the value of *ACCOUNT* for that case was "CURRENT" or "SAVE." The other field gets the value $null$, the missing value for numeric fields.

We can now proceed to the aggregate.

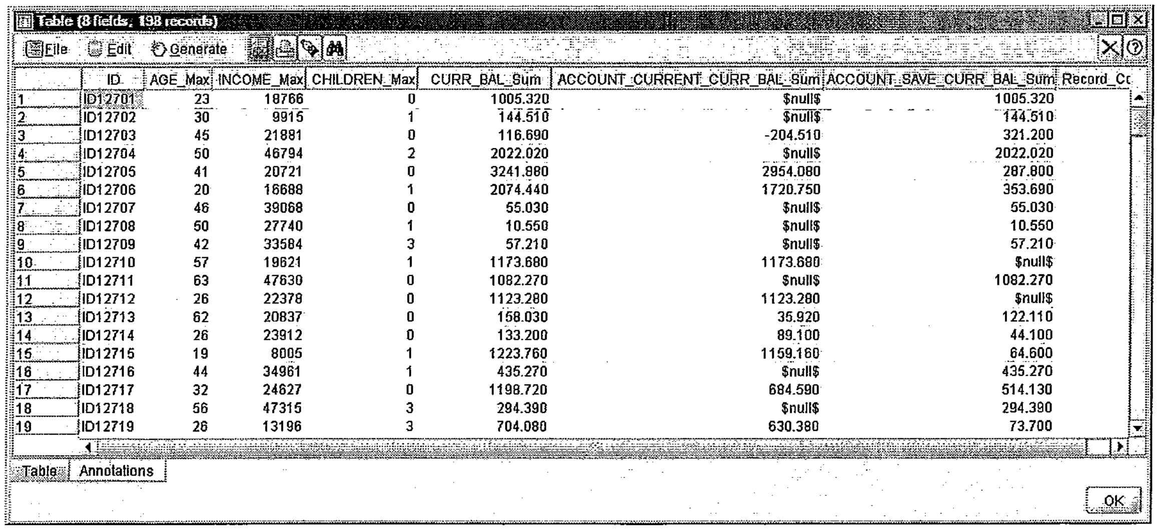
Close the Table window
Add an **Aggregate** node to the stream
Connect the **Restructure** node to the **Aggregate** node
Edit the **Aggregate** node
Select the **ID** field in the **Key fields** list
Check the **Keys are contiguous** check box
Select **AGE**, **INCOME** and **CHILDREN** from the Aggregate fields: list
Check **Max** statistic (deselect **Sum** and **Mean**) for AGE, INCOME and CHILDREN
Select **CURR_BAL**, **ACCOUNT_CURRENT_CURR_BAL**, and
ACCOUNT_SAVE_CURR_BAL from the Aggregate fields: list
Check **Sum** statistic (deselect **Mean**) for these three balance fields

Figure 9.19 Completed Aggregate Node Dialog



Click **OK** to return to the Stream Canvas
Connect the **Aggregate** node to a **Table** node
Execute the **Table** node

Figure 9.20 Data Showing New Fields Created by Restructure With Aggregation



	ID	AGE_Max	INCOME_Max	CHILDREN_Max	CURR_BAL_Sum	ACCOUNT_CURRENT_CURR_BAL_Sum	ACCOUNT_SAVE_CURR_BAL_Sum	Record_Cc
1	ID12701	23	18766	0	1005.320	$null$	1005.320	
2	ID12702	30	9915	1	144.510	$null$	144.510	
3	ID12703	45	21881	0	116.690	-204.510	321.200	
4	ID12704	50	46794	2	2022.020	$null$	2022.020	
5	ID12705	41	20721	0	3241.880	2954.080	287.800	
6	ID12706	20	16688	1	2074.440	1720.750	353.690	
7	ID12707	46	39068	0	55.030	$null$	55.030	
8	ID12708	50	27740	1	10.550	$null$	10.550	
9	ID12709	42	33584	3	57.210	$null$	57.210	
10	ID12710	57	19621	1	1173.680	1173.680	$null$	
11	ID12711	63	47630	0	1082.270	$null$	1082.270	
12	ID12712	26	22378	0	1123.280	1123.280	$null$	
13	ID12713	62	20837	0	158.030	35.920	122.110	
14	ID12714	26	23912	0	133.200	89.100	44.100	
15	ID12715	19	8005	1	1223.760	1159.160	64.600	
16	ID12716	44	34961	1	435.270	$null$	435.270	
17	ID12717	32	24627	0	1198.720	684.590	514.130	
18	ID12718	56	47315	3	294.390	$null$	294.390	
19	ID12719	26	13196	3	704.080	630.380	73.700	

We now see only one record for each customer. The two fields created by the Restructure node sum to the value of *CURR_BAL_Sum*.

Summary

In this chapter we introduced methods to manipulate the data structure.

You should now be able to:

- Use the Aggregate node to create summary records
- Use the Set To Flag node to expand a set field into a collection of flag fields
- Combine the outputs from the Set to Flag and Aggregate nodes using the Merge node
- Use the Restructure node to as an alternative to the Set to Flag node

Exercises

In this exercise you will restructure the data that was merged in the exercises for Chapter 9 to create a file that has one record per holiday code. Each new record will contain information including total number of holidaymakers, total cost, average length of stay etc.

1. If it is not already loaded, load the stream created in the previous chapter exercise, *ExerChapter8.str*. (If that is unavailable, load the *Backup_ExerChapter8.str* from the course folder.)

2. First, check to see if the data stream is sorted by *HOLCODE*. Hint: Add a Table node to the Merge node. Is the stream sorted?

3. Connect an Aggregate node to the Merge node. Edit the Aggregate node. Use *HOLCODE* as the key field, and create the following aggregate fields:
   Mean number of people in party (*NUMPARTY*)
   Total holiday cost (*HOLCOST*)
   Mean number of nights stayed (*NIGHTS*)
   Retain the distance to the beach (*DIST_TO_BEECH*)– use Min mode
   Total number of bookings for this holiday (Include record count).

   Use the check box that tells Aggregate that the key values are contiguous.

4. Connect a Table node to the Aggregate node and execute this section of the stream. Has the data aggregation worked? How much, in total, has been spent by individuals for their holidays in code CAF3108? How many records/holidays were aggregated to create this information?

5. Use a Set to Flag node (connect to Sort node) to create three new fields in the original merged data that represent whether each customer requested Full board (FB), Half board (HB) or self-catering (SC) in field *ACCOM*. Connect a Table node to verify the results.

6. Edit the Set to Flag node to create these three variables aggregated on *HOLCODE*.

7. Merge the Aggregate and Set to Flag fields in one data stream and review the data with a Table.

8. Save the stream as *ExerChapter9.str*.