Standardize Your Data Preparation in SAS: Use SQL!

May 2009

SYSTEMS SEMINAR CONSULTANTS, INC.
www.sys-seminar.com
2997 Yarmouth Greenway Drive
Madison, WI 53711
(608) 278-9964

Welcome to Systems Seminar Consultants, Inc.

Systems Seminar Consultants, Inc. is a SAS Alliance Quality Partner™ of SAS. Our team of SAS software experts has a broad base of knowledge and experience working with a variety of complex systems in a number of diverse industry settings. This knowledge and experience is leveraged to help you effectively achieve your business goals.

Free SAS Newsletter

Our popular publication, The Missing Semicolon™, shares SAS software solutions developed by our staff and provides additional technical assistance to our customers.

SAS Training Services

For over 1,000 students each year, we make SAS software easier to understand, use, and support.

- Public training schedules are posted on our web site.
- Private on-site training options are also available.

Additional SAS Services

SAS Consulting Services

Our staff of SAS consultants is well-versed in a variety of business areas. Our specialty areas include:

- Data systems development
- Decision support and business consultation
- Market research and analysis

SAS Help Desk Services

Our SSC team of experts is available to solve your company's daily SAS problems. Call us to develop a customized support plan that meets your company's needs.

For More Information

To receive additional information about our services or discuss a specific cost-effective solution for your company:

- Call us at: (608) 278-9964
- Check our Website at: www.sys-seminar.com
- Contact us at: 2997 Yarmouth Greenway Drive, Madison, WI 53711

Presenter

Steven J. First, President

- Over 30 years of SAS experience, including hundreds of manufacturing, retail, government, marketing, and financial applications.
- Over 25 years as President and Founder of SSC
- Founder of WISAS and WISUG
- Invited speaker at local, regional, and international user groups

Outline

- Data Preparation as a Separate Task
- What is SQL and Why is it Often Underused?
- Using SQL for Data Preparation

Data Preparation as a Separate Task

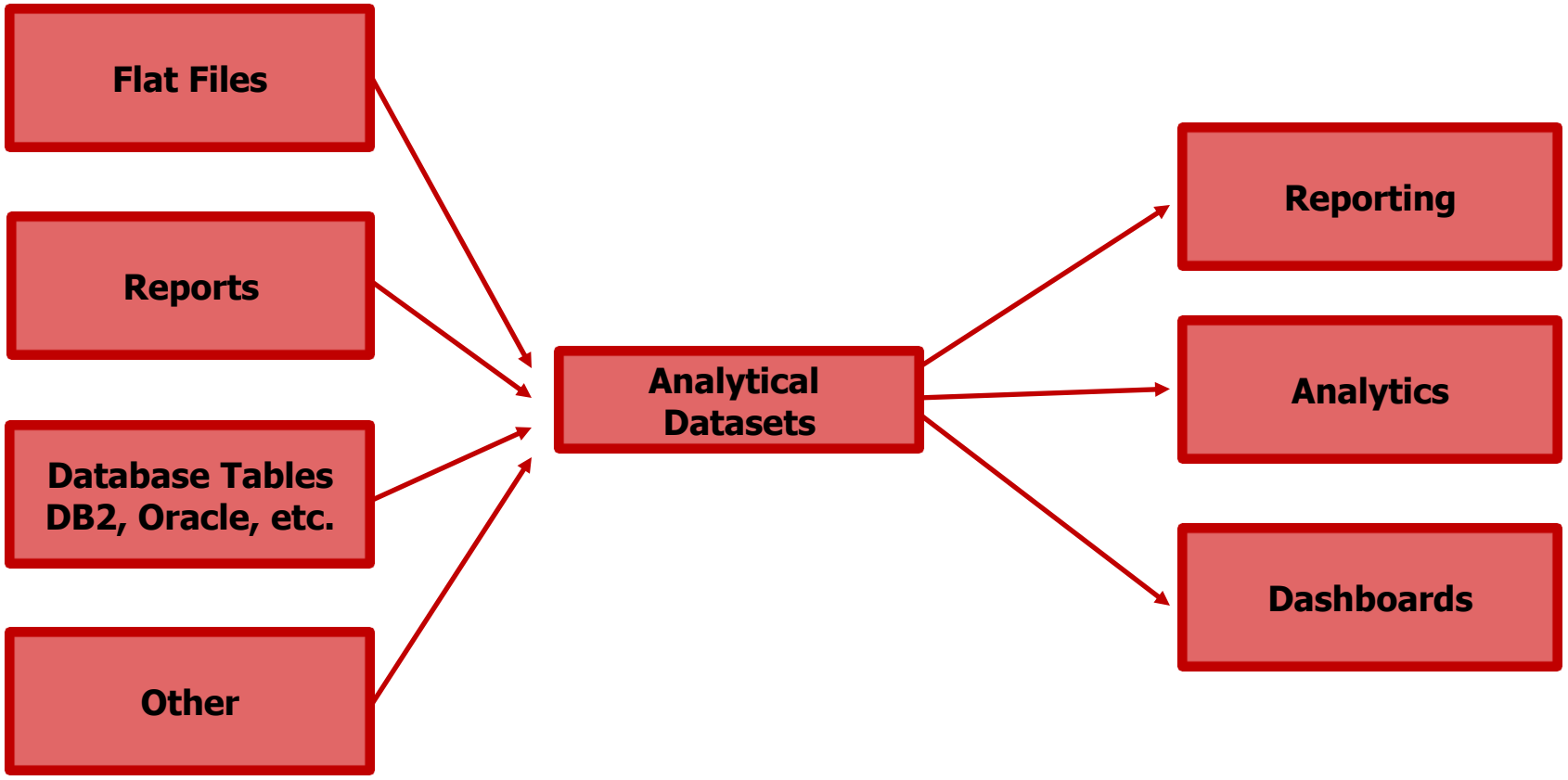
SAS Program Structure – What is Data Preparation?

Data Preparation = ETL (Extract, Transform, Load)

- Extracting Data
  - Profiling Data
  - Cleaning Data
- Transforming Data
  - Aggregation
  - Filtering
  - Joining
  - Sorting
- Loading Data to SAS Datasets, Data Marts, etc

i.e. Source to Target Processing

Typical Data Flow for SAS Processes

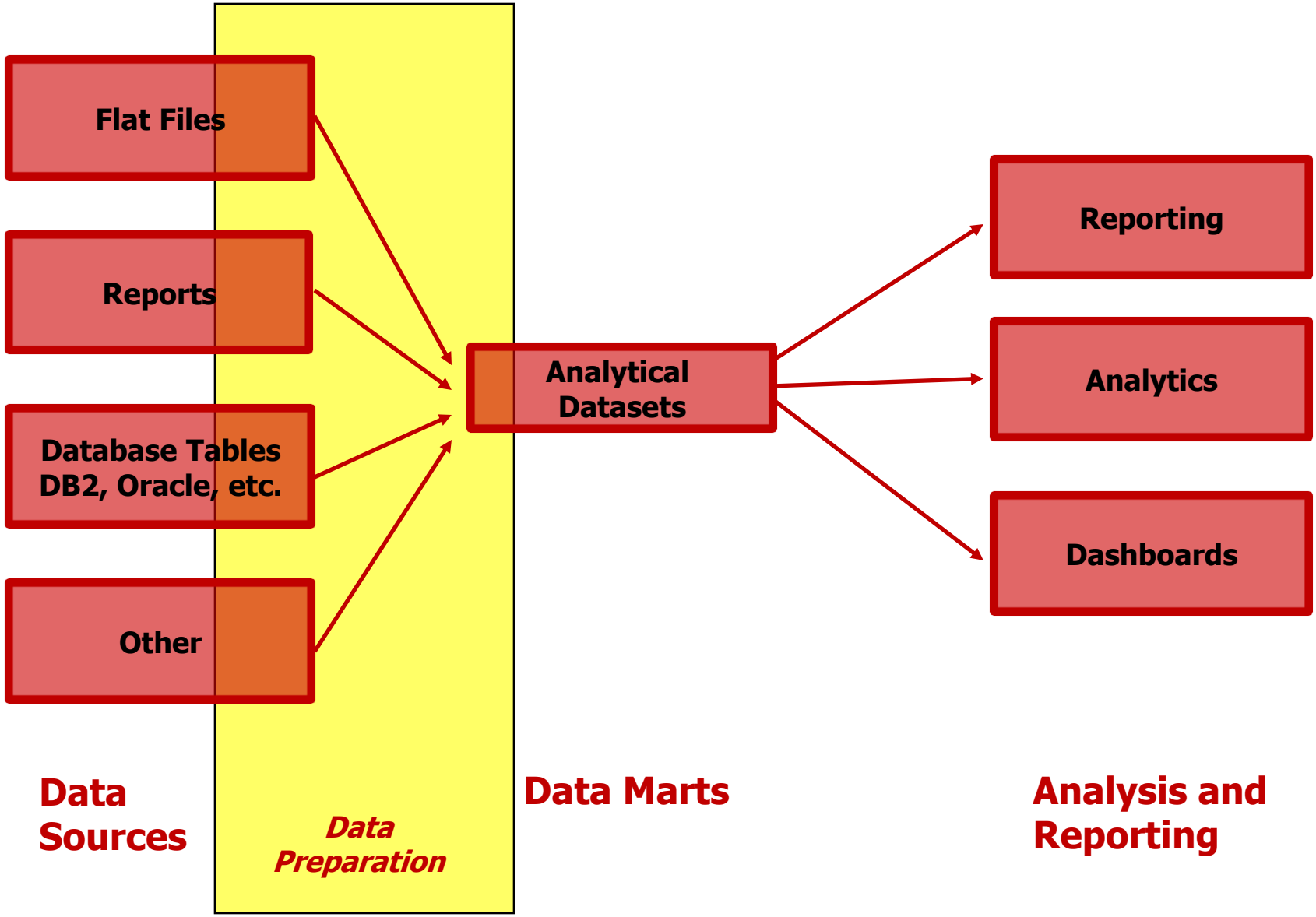


Data Sources

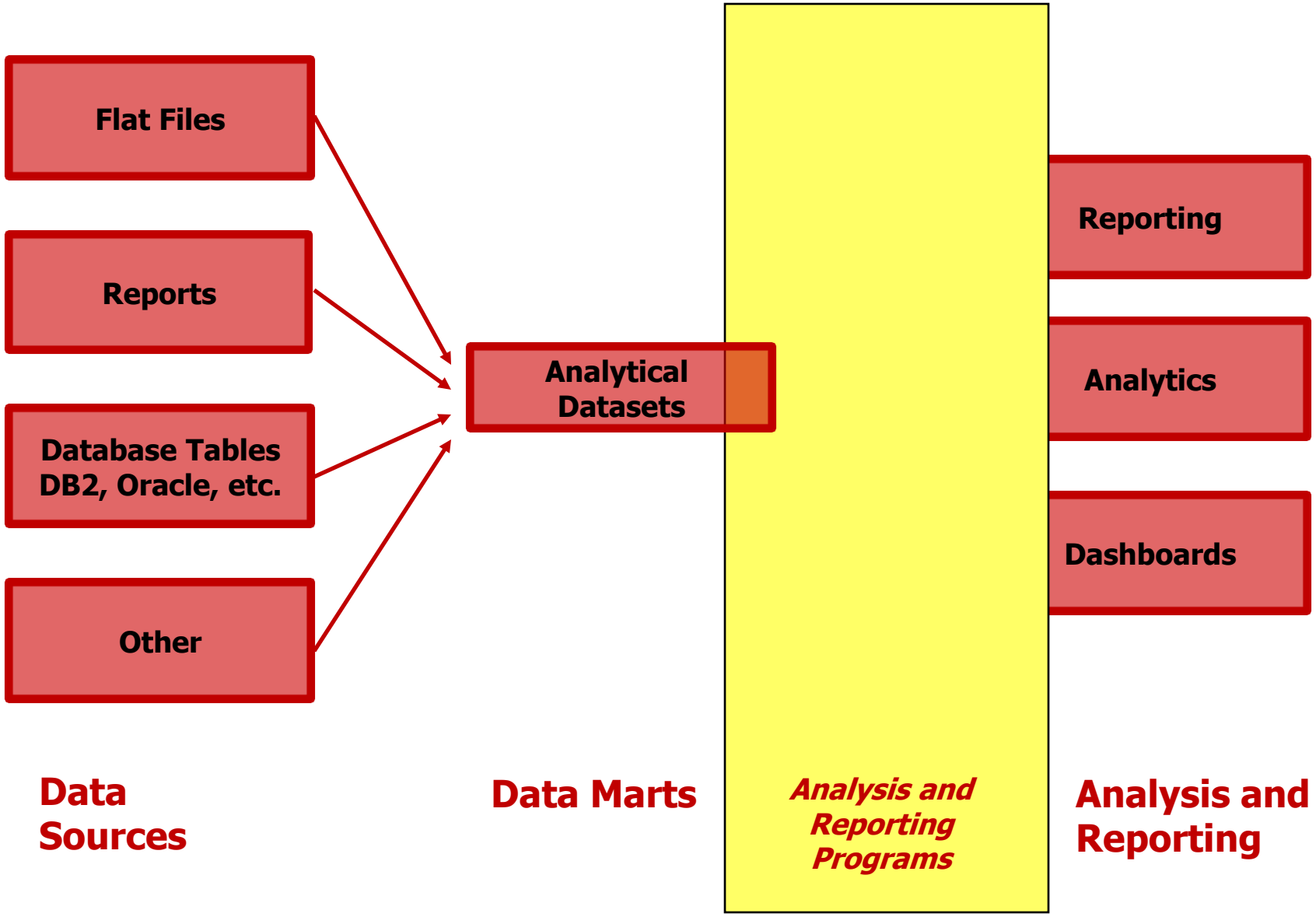
Data Marts

Analysis and Reporting

Typical Data Flow for SAS Processes - Data Preparation



Typical Data Flow for SAS Processes - Analysis & Reporting



How do YOU develop SAS Programs?

- Formal Plan?
- Trial and Error?

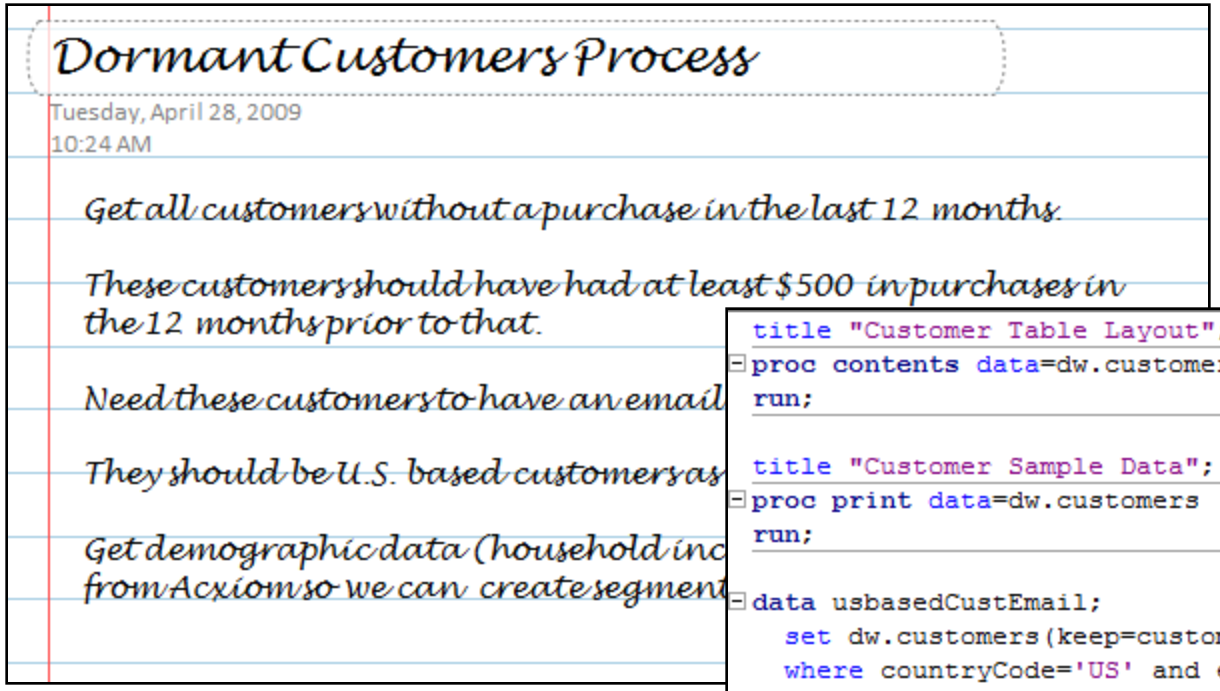
Example Process

- Create segmentation reports showing demographic information on customers who have not made a purchase in the last year, but have spent at least $500 the prior year.

- These U.S. based customers should have a valid email address, so we can also create an extract file to use to send out a special promotion.

Informal Plan with Trial and Error

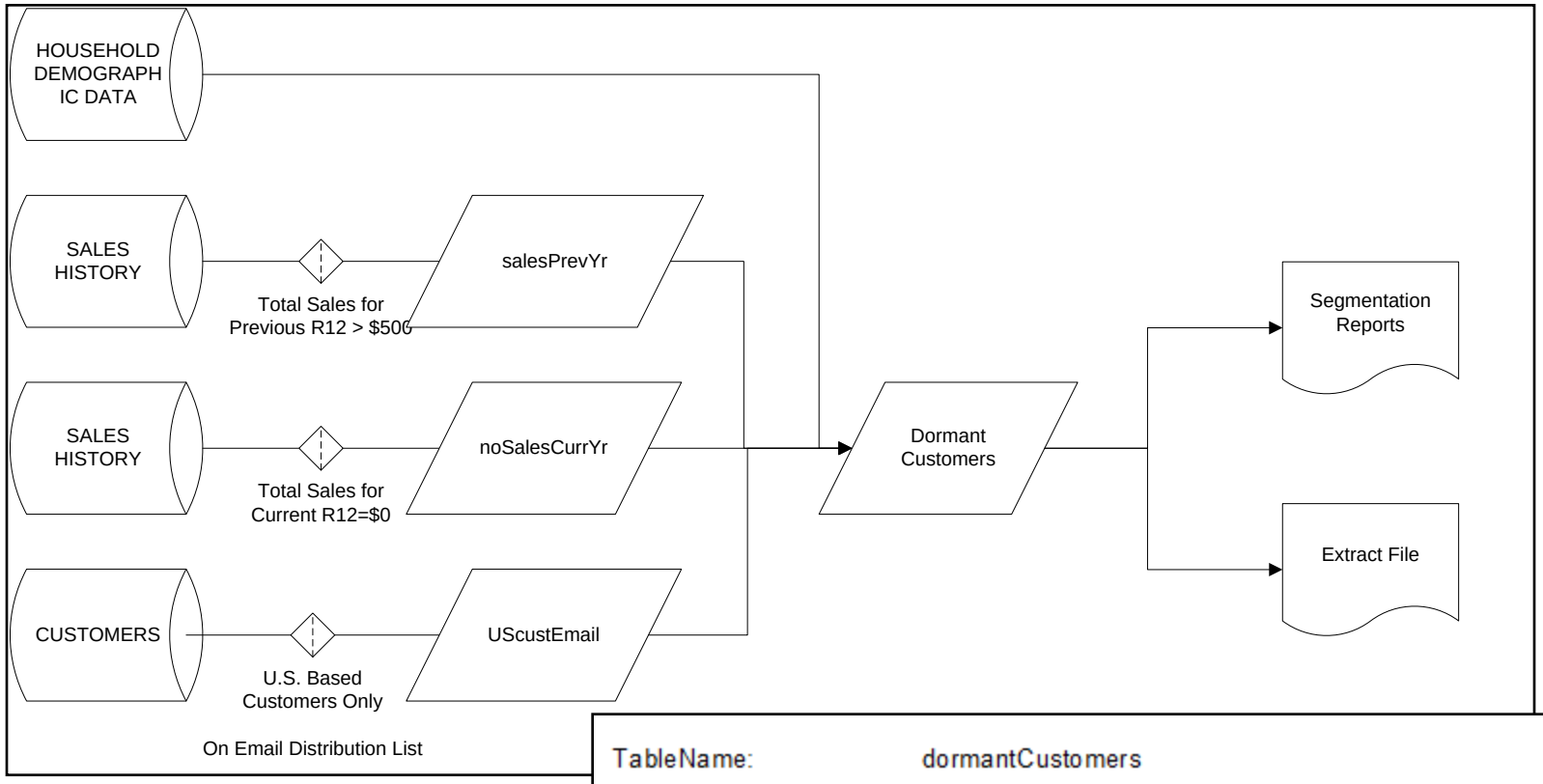
Informal Specifications and Trial and Error Programming




```
title "Customer Table Layout";
proc contents data=dw.customers;
run;

title "Customer Sample Data";
proc print data=dw.customers (obs=10);
run;

data usbasedCustEmail;
  set dw.customers(keep=customerID countryCode emailAddress1);
  where countryCode='US' and emailAddress1 is not missing;
  drop countryCode emailAddress1;
run;

title "Sales Table Layout";
proc contents data=dw.sales;
run;

title "Sales Sample Data";
proc print data=dw.sales (obs=10);
run;
```

Formal Plan with Flow Charts & Sourcing Documents

- Formal Plan
  - Flow Chart with Table Names
  - Source to Target Documents with Column Names

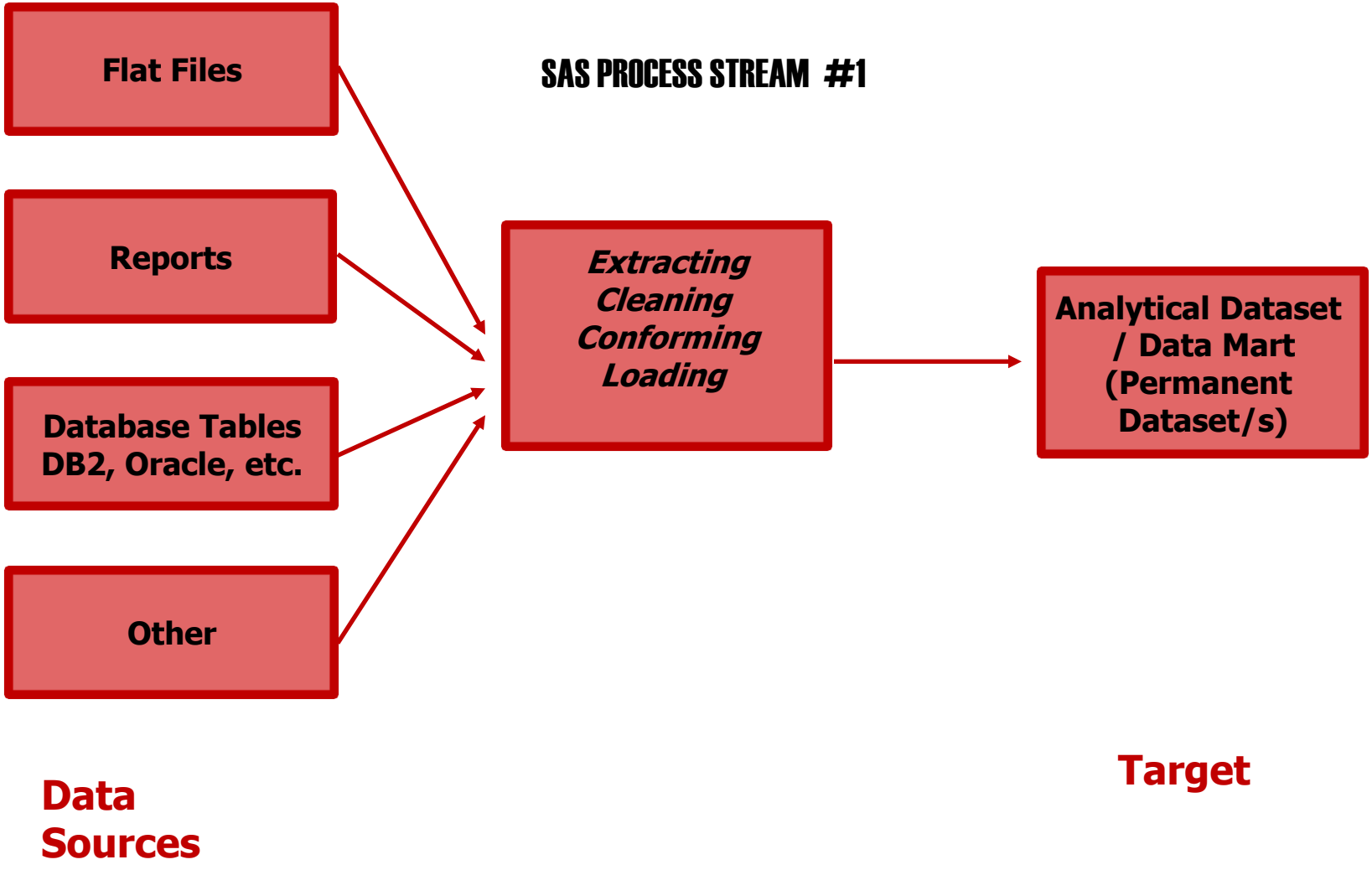


TableName: dormantCustomers

Variable	Description	Field Name	Source Name	Derived / Raw
customer ID	unique customer identifier	custID	customers	raw
.....				

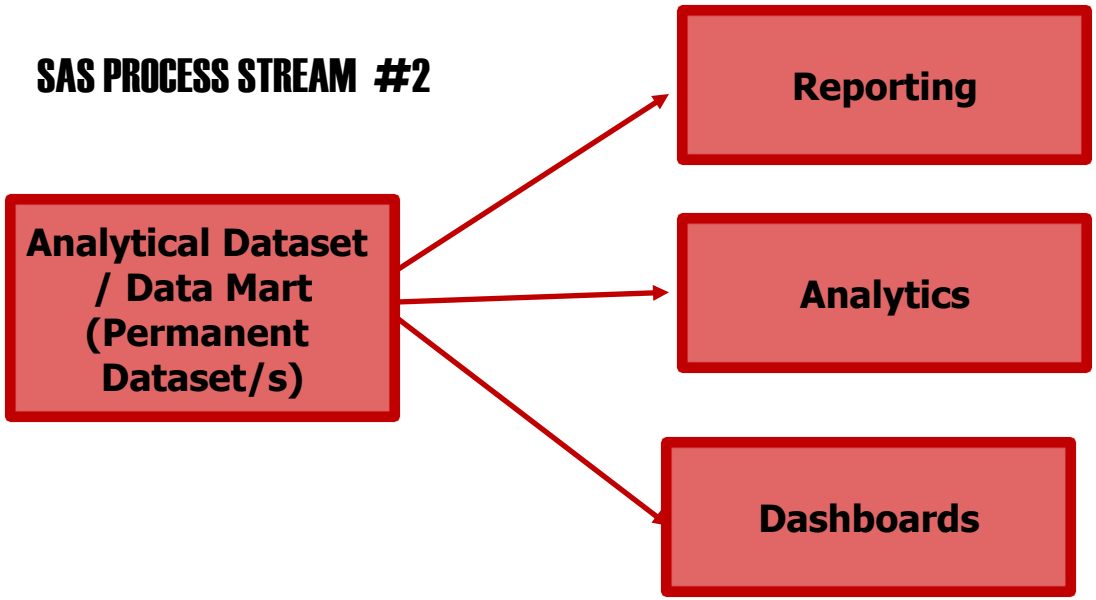
Source to Target Processing

Logically separating data preparation processes from data analysis and reporting makes system design and maintenance easier.

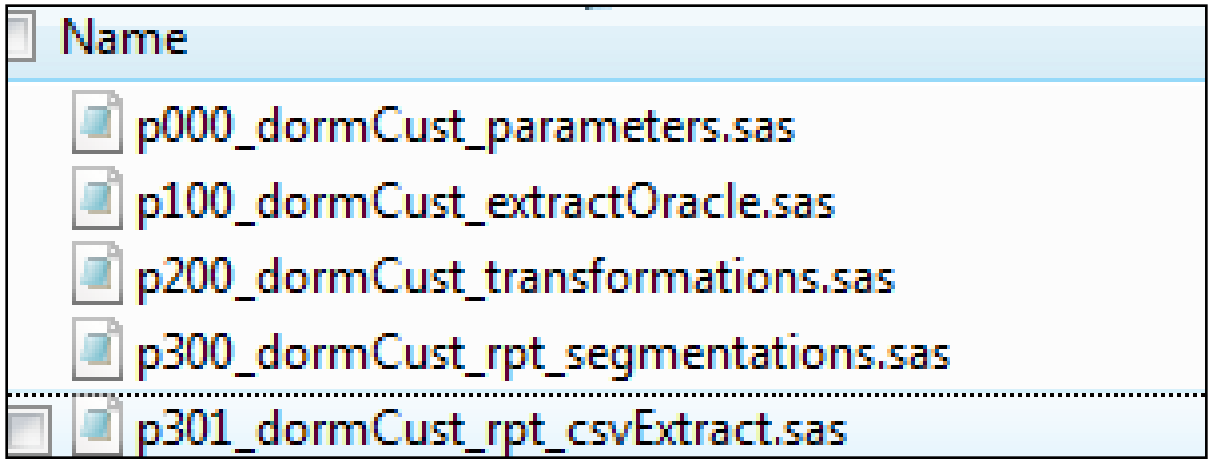


Analytic Processing

Logically separating data preparation processes from data analysis and reporting makes system design and maintenance easier.



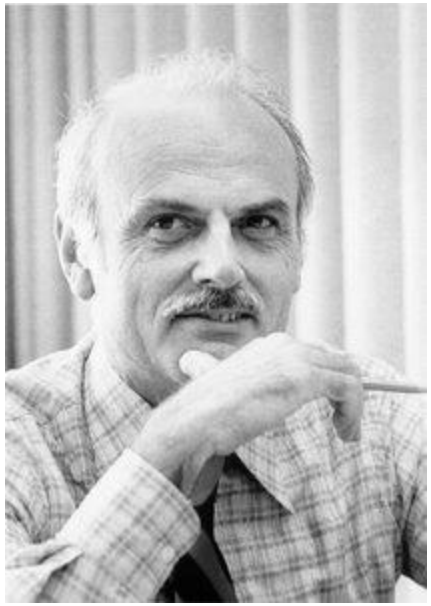
Final Program Lists



What is SQL, and Why is it Often Underused within SAS?

What is SQL?

- Origins – Authored by Dr. E.F. Codd of IBM
- ANSI Standards 1986, 1989, 1992, 199, 2003
- DDL (data definition language) and DML (data manipulation language). *We are concentrating on DML.*
- Simple Syntax
  - Easy to understand data flow (multiple tables in, one table out)
  - Small number of verbs (clauses)



History of SQL in SAS

- First release of SAS in 1976
- Proc SQL was first introduced in SAS Version 6 (~ 1988)
  - *SQL conceptually different*
  - *Syntax completely different*
- The Pass through Facility in SAS/ACCESS introduced shortly thereafter. (~1990)
- SAS/Access Views were also used to access RDBM data
- SAS/Access LIBNAME engine not introduced until SAS V7 (~ 1999)

Many SAS Users Introduced to SQL Through the Pass-Through Facility

For many users, DB2, Oracle, and SQL was the first exposure to SQL.

To access RDBMS. Required either a SAS/ACCESS view or SQL pass through facility.

- Somewhat complicated
- Users would pass data back to SAS as quickly possible
- V7 and higher, LIBNAME engine simplified syntax

Notes: PROC SQL provides much more: SQL for SAS datasets!

Many Use SQL Exclusively to Access Data from RDMS

Sample pass-through facility syntax

```
proc sql;
  connect to oracle ( user=sscdata
                      orapw=sscpw
                       path="ssctrain");
  create table work.claimData as        /* create sas dataset   */
  select patientID,
         patname,                       /* sas select           */
         serviceDate,
         billAmt                        /*                      */
  from connection to oracle             /* get data from oracle */
  (select m.patientID,
          m.patName,                    /* start pass-thru query*/
          c.serviceDate,
          c.billAmt                     /* select fields        */
   from  members m left join            /* left join tables     */
         claims  c
     on m.patientID=c.patientID
     );                                 /* end  pass-thru query */
  %put &sysdbrc &sysdbmsg;              /* return codes         */
  disconnect from oracle;
quit;
```

Can Access SAS Datasets

Querying a local SAS dataset

```
proc SQL;
  create table work.claimData as      /* create sas dataset   */
  select patientID,
         patname,                     /* sas select           */
         serviceDate,
         billAmt                      /*                      */
  from work.sasclaims
  where year(serviceDate)=2009;
quit;
```

Fully Functional Query Language

Example: Gather U.S. states with total orders over 100,000 and group midwest states together.

```
proc sql;
  create table midwestsummary as
  select case when c.state in ('IL','MN','WI','IA','MI')
                then 'Midwest'
              else c.state
  end as region,
         sum(o.orderAmt) as TotalOrderAmt,
         mean(o.orderAmt) as AvgOrderAmt
  from datamart.customers c inner join
       datamart.orders    o on c.custid=o.custid
  where c.countrycode='US'
  group by 1
  having sum(o.orderAmt) > 100000
  order by totalOrderAmt desc;
quit;
```

SELECT Clause

The SELECT clause is used to:

- List all the columns and transformations.
- Specify CASE-WHEN logic to conditionally set values.
- Invoke all SAS functions (with a few exceptions).

Example:

```
select custid,
       case when orderAmt < 100
                        then 'LOW'
            when orderAmt < 250
                        then 'AVERAGE'
            else 'LARGE'
       end as transactionSize
```

SELECT Clause – INTO – Create Macro Variables

The SELECT clause can also create macro variables easily.

Example: Create a single macro variable containing all codes found and separate them with commas.

```
proc sql noprint;
  select code
    into :mincodes separated by ','
      from codes;
quit;

%PUT MACRO VARIABLE 'MINCODES' = &MINCODES;
```

SAS LOG:

```
MACRO VARIABLE 'MINCODES' = 123,456,789
```

SELECT Clause – INTO – Use Macro Variable

Use the Macro Variable in a WHERE statement.

Examples:

```
* COULD USE IN PROC PRINT;
PROC PRINT DATA=NAMES;
  WHERE NUMBER IN (&MINCODES);
RUN;
```

```
Results:
Obs    NUMBER    NAME
 1       123     DAVE
 4       456     MARY
 7       789     LINDA
```

```
* COULD ALSO USE IN SQL QUERY;
Proc SQL;
  SELECT *
    FROM NAMES
      WHERE NUMBER IN (&MINCODES);
Quit;
```

```
Results:
 NUMBER   NAME
_____________________

    123   DAVE
    456   MARY
    789   LINDA
```

SELECT Clause – INTO – Additional Examples

```
/* 1. to select the first value from dataset into macro variable: */
proc sql noprint;
  select Doctor
  INTO :frstdoc
  from  Ervisits;
quit;
%put The First ER Visit on Record was treated by &frstdoc;
```

SAS LOG:

```
The First ER Visit on Record was treated by Jones
```

```
/* 2. to select all values into one macro variable:                */
proc sql noprint;
  SELECT DISTINCT Doctor
  INTO :docs separated by ', '
  from Ervisits;
Quit;
%put 'docs' Macro Variable:  "&docs";
```

SAS LOG:

```
'docs' Macro Variable:  "Jones, Smith, White"
```

SELECT Clause – INTO – Additional Examples

```
/* 3. to select all values into macro variable and quote values   */
proc sql noprint;
  select DISTINCT QUOTE(COMPBL(Doctor))
  INTO :docs separated by ', '
  from Ervisits;
quit;

%put docs2=&docs2;
%let docs2=%sysfunc(tranwrd(%bquote(&docs2),%bquote("),%bquote(')));
%put docs2=&docs2;
```

SAS LOG:

```
docs2= "Jones ", "Smith ", "White "
docs2='Jones ', 'Smith ', 'White '
```

```
/* 4. writing the result of a function to a macro variable:       */
proc sql noprint;
  select MAX(Charged) format=dollar8. ,
         MIN(Charged) format=dollar8.
  INTO :maxchrg, :minchrg
  from  Ervisits;
quit;
%put minchrg=&minchrg maxchrg=&maxchrg;
```

SAS LOG:

```
minchrg=$105 maxchrg=$1,234
```

SELECT Clause – INTO – Additional Examples

```
/* 5. writing the number of rows to a macro variable:                */
proc sql noprint;
  select COUNT(DISTINCT Doctor)
  INTO :numrows
  from Ervisits;
quit;
%let numrows=&numrows;
%put numrows: &numrows;
```

SAS LOG:

```
numrows: 3
```

```
/* 6. writing out a range of values into multiple macro variables:*/
proc sql noprint;
  select DISTINCT Doctor
    INTO :doc1-:doc&numrows
  from Ervisits;
quit;
%put doc Variables: doc1=&doc1 doc2=&doc2 doc3=&doc3;
```

SAS LOG:

```
doc Variables: doc1=Jones doc2=Smith doc3=White
```

FROM Clause

The FROM clause is used to:

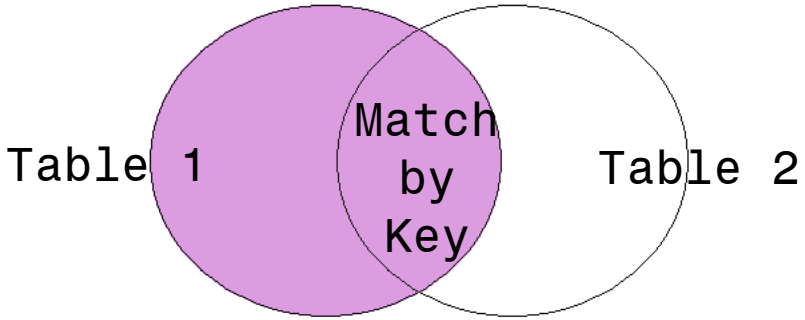
- Select source data
- Join tables (FULL, INNER, LEFT and RIGHT)
- Concatenate tables

Example:

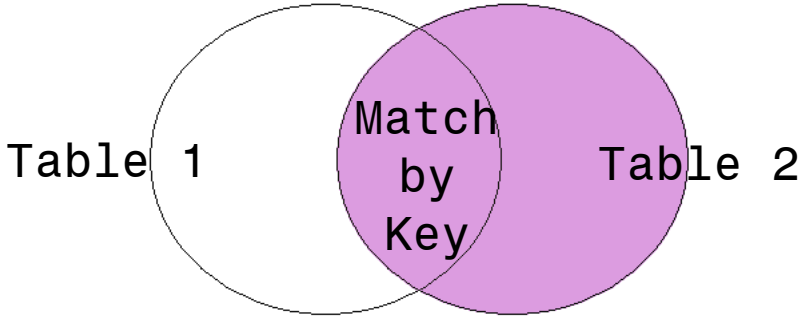
```
FROM datamart.customers c INNER JOIN
     datamart.orders    o on c.custid=o.custid
```

FROM Clause – Outer Joins in SQL

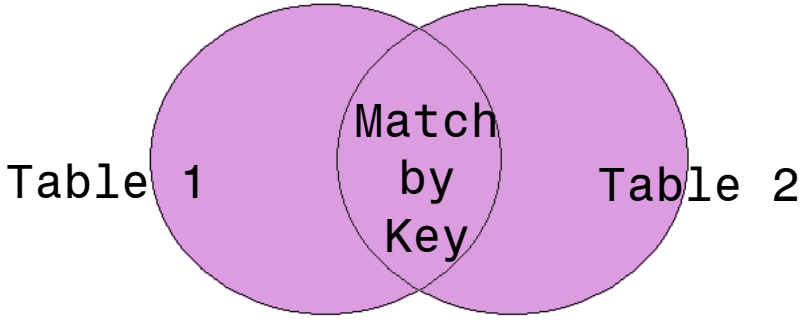
Left Join



Right Join



Full Join



FROM Clause – Full Join Example - Outer Joins in SQL

LEFT, RIGHT or FULL Joins

```
Proc SQL;
 SELECT *
 FROM GIRLS A
      FULL JOIN
      BOYS B
 ON A.STATE=B.STATE;
Quit;
```

GNAME	STATE	BNAME	STATE
		ADAM	CA
AMELIA	IL		
NANCY	WI	NED	WI
NANCY	WI	GENE	WI
JEAN	WI	NED	WI
JEAN	WI	GENE	WI

Notes:

- **Full** Join keeps records from both data sets

FROM Clause – Left Join Example - Outer Joins in SQL

LEFT, RIGHT or FULL Joins

```
Proc SQL;
 SELECT *
 FROM GIRLS A
      LEFT JOIN
      BOYS B
 ON A.STATE=B.STATE;
Quit;
```

GNAME	STATE	BNAME	STATE
AMELIA	IL		
NANCY	WI	NED	WI
NANCY	WI	GENE	WI
JEAN	WI	NED	WI
JEAN	WI	GENE	WI

Notes:

- **Left** Join keeps all records from left side of From (If ONA) Lists all the columns and transformations.

FROM Clause – Right Join Example - Outer Joins in SQL

LEFT, RIGHT or FULL Joins

```
Proc SQL;
 SELECT *
 FROM GIRLS A
      RIGHT JOIN
      BOYS B
 ON A.STATE=B.STATE;
Quit;
```

GNAME	STATE	BNAME	STATE
		ADAM	CA
NANCY	WI	NED	WI
NANCY	WI	GENE	WI
JEAN	WI	NED	WI
JEAN	WI	GENE	WI

Notes:

- **Right** Join keeps all records from right side of From (If ONB)

WHERE Clause

- The WHERE clause is used to subset the rows in our source data.
- Subqueries can be used on the FROM clause (and in many other clauses).

Example:

```
WHERE state='NY' and
      custid in (SELECT custid
                 from datamart.orders
                 group by custid
                 having sum(orderAmt) > 5000)
```

GROUP BY Clause

The GROUP BY clause is used to:

- Summarize rows by the columns listed.
- Create aggregated statistics using summary functions

Example:

```
select custid,
       SUM(orderAmt) as TotalOrderAmt,
       MEAN(orderAmt) as AvgOrderAmt
From datamart.orders
GROUP BY custid
```

HAVING Clause

The HAVING clause is:

- Used to Subset on summarized information.
- Different from the FROM clause that operates on individual rows.

Example:

```
select custid,
       sum(orderAmt) as TotalOrderAmt,
       mean(orderAmt) as AvgOrderAmt
from   datamart.orders
group  by custid
HAVING mean(orderAmt) > 100
```

ORDER BY Clause

The ORDER BY clause is used to sequence query results in ascending or descending order (sort).

Example:

```
ORDER BY orderAmt desc
```

Notes: Columns do not have to appear in query results

Use Query Clauses Together for Maximum Flexibility

All query clauses shown:

```
proc sql options;
  SELECT column(s)
  FROM table-name | view-name
  WHERE expression
  GROUP BY column(s)
  HAVING expression
  ORDER BY column(s)
   ;
quit;
```

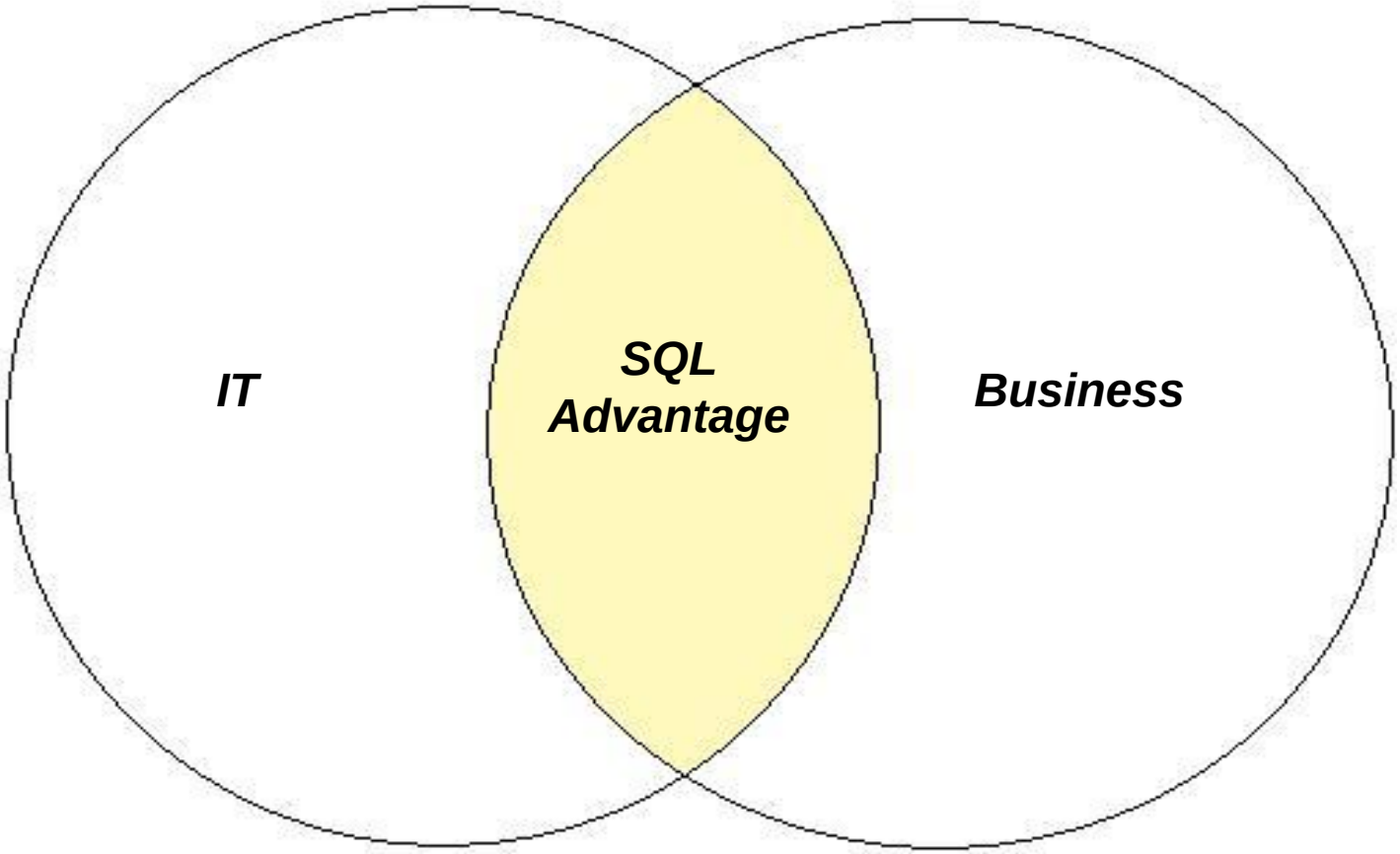
Using SQL for Data Preparation

Why Use SQL?

- Standardized Language Many Can Understand
- Intuitive
- Others More Likely to Pick up and Maintain Code
- Consistent style = Easier to Maintain Code
- Other SAS Products Moving That Way
- Simpler Macro language
- Flexible Joining Logic

Useful Tool for the Business Areas and IT

SAS is often not supported by IT staff, though all IT departments use and support SQL.



Often IT Supported

- Production Process / Problems

- Future Conversion to Other ETL Tool.

Coding Efficiencies Using SQL

- Flexibility of language allows the combination of many regular PROC and DATA steps. Not just DATA ACCESS!
  - SELECT
  - FROM
  - WHERE
  - GROUP BY
  - HAVING
  - ORDER BY

- This often results in faster run times.

Possible Rules for Proc SQL Only Programs

- Separate data processing from reporting and statistical analysis. SQL only rules apply primarily to data processing, though reporting is also a possibility.
- LIBNAME, FILENAME, OPTION statements okay.
- Macro programs and variables are okay with minimal if-then logic.
- Minimize data steps and proc steps for data processing.
- Almost always can find SQL equivalent.

SQL Exceptions

Some things that SQL cannot easily do are:

- Arrays
- Flat Files
- Some Functions
  - SOUNDEX
  - LAG
  - DIF
- Complicated DO Looping
- Some FIRST. LAST. Processing
- Others?

Challenging Programming Examples in Other Presentations

- Creating Buckets in SQL
- Advanced Subqueries
- More to Come

Summary

SSC thinks that SQL is:

- Consistent
- Easy to learn
- Intuitive
- Promotes good design
- Another tool!

Contact Us

SYSTEMS SEMINAR CONSULTANTS, INC.

SAS® Training, Consulting, & Support Services

2997 Yarmouth Greenway Drive • Madison, WI 53711

(608) 278-9964 x302 • Fax (608) 278-0065

www.sys-seminar.com

SAS Alliance Silver Member

Steven First

President

sfirst@sys-seminar.com

SAS Certified Professional Version 8