

Using SQL in SAS

David Ghan
SAS Education
416 307-4515
David.ghan@sas.com

Using **SQL** in SAS
Victoria Area SAS User Group
February 12, 2004

1. What is SQL?
2. Coding an SQL Query
3. Advanced Examples
 - a. Creating macro variables with SQL
 - b. Using Indexing
 - c. Dictionary Tables
 - d. SQL pass-through
4. Online Help for Proc SQL

What is SQL?

Structured Query Language (SQL)

- is a standardized language that is widely used to retrieve and update data in tables and in views based on those tables
- was originally designed as a query tool for relational databases, but is now used by many software products.

What is SQL?

Timeline

- 1970 Conceptualized and proposed by Dr. E. F. Codd at the IBM Research Laboratory, San Jose, CA
- 1970-1980 Developed by IBM
- 1981 First commercial SQL-based product, the IBM SQL/DS System
- 1989 Over 75 SQL database management systems exist, including SAS Release 6.06.

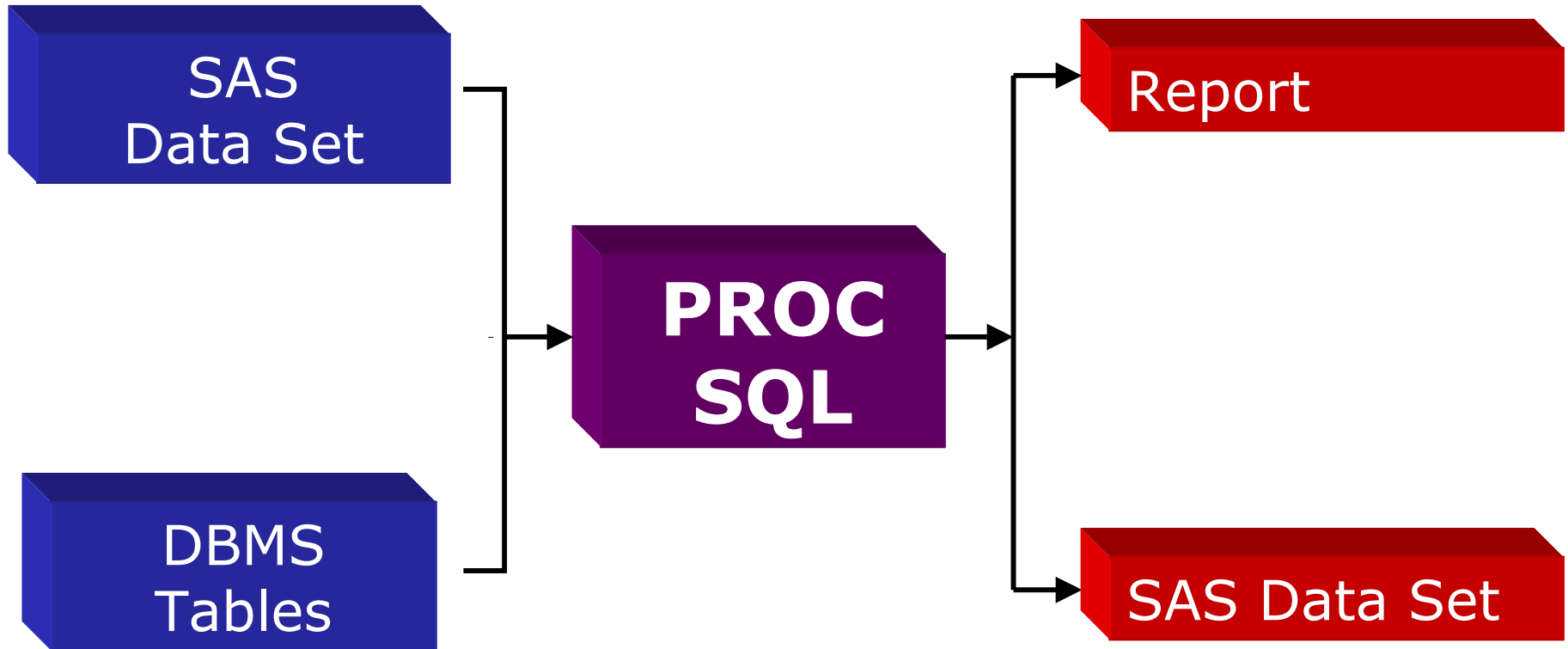
What is SQL?

In SAS, you can use the SQL Procedure to:

- select rows and column from datasets
- derive new values
- combine datasets
- store the result as a reports or output dataset
- summarize data
- data management: create, delete, insert, update ...

What is SQL?

Structured Query Language



What is SQL?

The SQL procedure

- enables you to use SQL within the SAS System
- follows the guidelines set by the American National Standards Institute (ANSI)
- an alternative for many common data manipulations using a “request-based” syntax

What is SQL?

The SQL Procedure

IS NOT

- a replacement for the DATA step
- a custom reporting tool.

IS

- a tool for queries
- for data manipulation
- an augmentation to the DATA step.

Coding an SQL Query

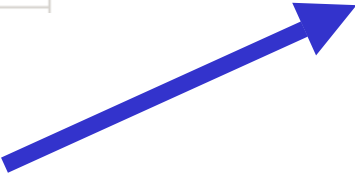
General form of the SELECT statement:

```
SELECT column <,column>...  
  FROM table|view<,table|view>...  
  <WHERE expression>  
  <GROUP BY column<,column>...>  
  <HAVING expression>  
  <ORDER BY column<,column>...>;
```

Coding an SQL Query

VIEWTABLE: Demodata.Payrollmaster						
	EmpID	Gender	JobCode	Salary	DateOfBirth	DateOfHire
22	1639	F	TA3	\$56,364	30JUN1955	31JAN1982
23	1269	M	NA1	\$58,366	07MAY1970	02DEC1990
24	1065	M	ME2	\$49,126	29JAN1942	10JAN1985
25	1876	M	TA3	\$55,545	23MAY1956	01MAY1983
26	1037	F	TA1	\$39,981	14APR1962	17SEP1990
27	1129	F	ME2	\$48,901	12DEC1959	20AUG1989
28	1988	M	FA3	\$45,104	03DEC1957	22SEP1982
29	1405	M	SCP	\$25,278	08MAR1964	29JAN1990
30	1430	F	TA2	\$46,095	03MAR1960	30APR1985
31	1983	F	FA3	\$46,787	03MAR1960	30APR1985
32	1134	F	TA2	\$46,847	09MAR1967	25DEC1986

EmpID	JobCode	Salary
1352	NA2	\$75,317
1417	NA2	\$73,178
1935	NA2	\$71,513
1839	NA1	\$60,806
1443	NA1	\$59,184
1332	NA1	\$59,049
1269	NA1	\$58,366
1111	NA1	\$56,820



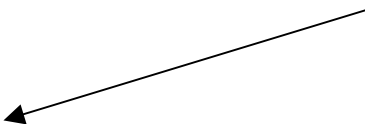
```
proc sql;  
    select EmpID, JobCode, Salary  
    from demodata.payrollmaster  
    where JobCode contains 'NA'  
    order by Salary desc;  
quit;
```

Coding an SQL Query

```
proc sql;  
    select EmpID, JobCode, Salary  
        from demodata.payrollmaster  
        where JobCode contains 'NA'  
        order by Salary desc;  
quit;
```

Coding an SQL Query

start the SQL procedure



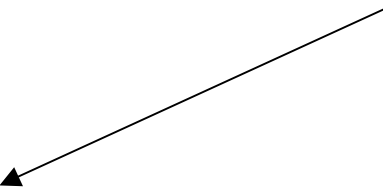
```
proc sql;
```

```
    select EmpID, JobCode, Salary  
    from demodata.payrollmaster  
    where JobCode contains 'NA'  
    order by Salary desc;
```

```
quit;
```

Coding an SQL Query

specify columns

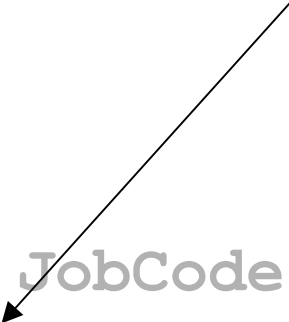


```
proc sql;  
  select EmpID, JobCode, Salary  
    from demodata.payrollmaster  
   where JobCode contains 'NA'  
   order by Salary desc;  
quit;
```

Coding an SQL Query

specify data set

```
proc sql;  
  select EmpID, JobCode, Salary  
    from demodata.payrollmaster  
  where JobCode contains 'NA'  
  order by Salary desc;  
quit;
```



Coding an SQL Query

specify which rows

```
proc sql;  
    select EmpID, JobCode, Salary  
        from demodata.payrollmaster  
    where JobCode contains 'NA'  
    order by Salary desc;  
quit;
```



Coding an SQL Query

order rows in results

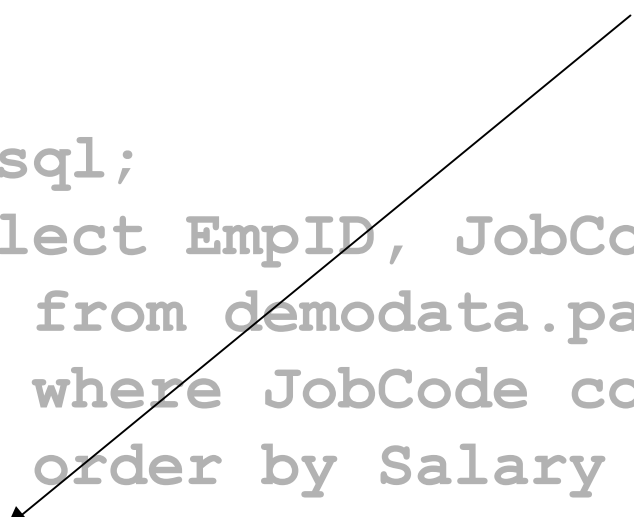
```
proc sql;  
    select EmpID, JobCode, Salary  
        from demodata.payrollmaster  
        where JobCode contains 'NA'  
        order by Salary desc;  
quit;
```



Coding an SQL Query

stop SQL

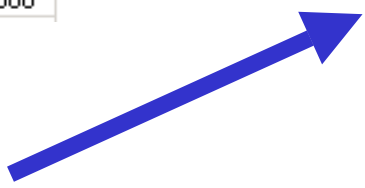
```
proc sql;  
    select EmpID, JobCode, Salary  
        from demodata.payrollmaster  
        where JobCode contains 'NA'  
        order by Salary desc;  
quit;
```



Coding an SQL Query

VIEWTABLE: Demodata.Payrollmaster						
	EmpID	Gender	JobCode	Salary	DateOfBirth	DateOfHire
22	1639	F	TA3	\$56,364	30JUN1955	31JAN1982
23	1269	M	NA1	\$58,366	07MAY1970	02DEC1990
24	1065	M	ME2	\$49,126	29JAN1942	10JAN1985
25	1876	M	TA3	\$55,545	23MAY1956	01MAY1983
26	1037	F	TA1	\$39,981	14APR1962	17SEP1990
27	1129	F	ME2	\$48,901	12DEC1959	20AUG1989
28	1988	M	FA3	\$45,104	03DEC1957	22SEP1982
29	1405	M	SCP	\$25,278	08MAR1964	29JAN1990
30	1430	F	TA2	\$46,095	03MAR1960	30APR1985
31	1983	F	FA3	\$46,787	03MAR1960	30APR1985
32	1134	F	TA2	\$46,847	09MAR1967	25DEC1986

EmpID	JobCode	Salary
1352	NA2	\$75,317
1417	NA2	\$73,178
1935	NA2	\$71,513
1839	NA1	\$60,806
1443	NA1	\$59,184
1332	NA1	\$59,049
1269	NA1	\$58,366
1111	NA1	\$56,820



```
proc sql;  
    select EmpID, JobCode, Salary  
    from demodata.payrollmaster  
    where JobCode contains 'NA'  
    order by Salary desc;  
quit;
```

Coding an SQL Query

Select statement returns results to the output window:

```
proc sql;  
    select EmpID, JobCode, Salary  
        from demodata.payrollmaster  
        where JobCode contains 'NA'  
        order by Salary desc;  
quit;
```

To store results in a SAS dataset place a Create table in front:

```
proc sql;  
  
    create table work.Navigators as  
    select EmpID, JobCode, Salary  
        from demodata.payrollmaster  
        where JobCode contains 'NA'  
        order by Salary desc;  
quit;
```

Coding an SQL Query

Summary Functions

Demodata.PAYROLLMASTER

EmplD	Gender	JobCode	Salary
1919	M	TA2	\$48,126
1653	F	ME2	\$49,151
1400	M	ME1	\$41,677
1350	F	FA3	\$46,040
1401	M	TA3	\$54,351
1499	M	ME3	\$60,235
1101	M	SCP	\$26,212
1333	M	PT2	\$124,048
1402	M	TA2	\$45,661
1479	F	TA3	\$54,299
1403	M	ME1	\$39,301
1739	M	PT1	\$93,124



```
proc sql;  
    select avg(Salary) as MeanSalary  
    from demodata.payrollmaster;
```



The SAS System

MeanSalary

56852.13

Coding an SQL Query

Summary Functions

Demodata.PAYROLLMASTER

EmpID	Gender	JobCode	Salary
1919	M	TA2	\$48,126
1653	F	ME2	\$49,151
1400	M	ME1	\$41,677
1350	F	FA3	\$46,040
1401	M	TA3	\$54,351
1499	M	ME3	\$60,235
1101	M	SCP	\$26,212
1333	M	PT2	\$124,048
1402	M	TA2	\$45,661
1479	F	TA3	\$54,299
1403	M	ME1	\$39,301
1739	M	PT1	\$93,124



```
proc sql;  
    select gender, avg(Salary) as MeanSalary  
    from demodata.payrollmaster  
    group by gender;
```



Gender	MeanSalary
F	49830.2
M	59192.78

Coding an SQL Query

SQL Joins

Demodata.PAYROLLMASTER

Gender	JobCode	Salary	DateOfBirth	DateOfHire	EmpID
M	TA1	\$40,432	05MAR1957	30MAR1990	1009
M	TA3	\$57,201	01JAN1956	20OCT1979	1017
F	TA3	\$55,149	23MAY1963	27OCT1982	1036
F	TA1	\$39,981	14APR1962	17SEP1990	1037
F	TA1	\$37,146	13NOV1967	26NOV1989	1038
M	ME2	\$49,234	17JUL1961	27AUG1984	1050
M	ME2	\$49,126	29JAN1942	10JAN1985	1065
M	PT1	\$93,181	18OCT1970	06OCT1989	1076
M	FA1	\$31,175	05APR1968	20APR1989	1094
M	BCK	\$35,006	05DEC1958	11MAY1986	1100
M	SCP	\$26,212	09JUN1960	04OCT1988	1101
M	TA2	\$48,359	04OCT1957	18APR1989	1102

Demodata.STAFFMASTER

EmpID	LastName	FirstName	City	State
1009	MORGAN	GEORGE	NEW YORK	NY
1017	WELCH	DARIUS	NEW YORK	NY
1036	WONG	LESLIE	NEW YORK	NY
1037	CHOW	JANE	STAMFORD	CT
1038	RODRIGUEZ	MARIA	BRIDGEPORT	CT
1050	TUTTLE	THOMAS	WHITE PLAINS	NY
1065	CHAPMAN	NEIL	NEW YORK	NY
1076	VENTER	RANDALL	NEW YORK	NY
1094	GOMEZ	ALAN	BRIDGEPORT	CT
1100	VANDEUSEN	RICHARD	NEW YORK	NY
1101	BAUCOM	WALTER	NEW YORK	NY
1102	HOWARD	LEONARD	WHITE PLAINS	NY
1103	MCDANIFI	RONDA	NEW YORK	NY

Coding an SQL Query

SQL Joins

Demodata.PAYROLLMASTER

Gender	JobCode	Salary	DateOfBirth	DateOfHire	EmpID
M	TA1	\$40,432	05MAR1957	30MAR1990	1009
M	TA3	\$57,201	01JAN1956	20OCT1979	1017
F	TA3	\$55,149	23MAY1963	27OCT1982	1036
F	TA1	\$39,981	14APR1962	17SEP1990	1037
F	TA1	\$37,146	13NOV1967	26NOV1989	1038
M	ME2	\$48,334	17JUL1961	27JUL1994	1050

Demodata.STAFFMASTER

EmpID	LastName	FirstName	City	State
1009	MORGAN	GEORGE	NEW YORK	NY
1017	WELCH	DARIUS	NEW YORK	NY
1036	WONG	LESLIE	NEW YORK	NY
1037	CHOW	JANE	STAMFORD	CT
1038	RODRIGUEZ	MARIA	BRIDGEPORT	CT



```
select LastName, JobCode, int((today()-DateOfBirth)/365.25) as Age
from demodata.payrollmaster, demodata.staffmaster
where payrollmaster.empID=staffmaster.empID and State='NY';
```



LastName	JobCode	Age
MORGAN	TA1	46
WELCH	TA3	48
WONG	TA3	40
TUTTLE	ME2	42
CHAPMAN	ME2	62
VENTER	PT1	33
VANDEUSEN	PT1	45

• note the derived column

Advanced Examples

Create macro variables with SQL

PROC SQL can create or update macro variables using an INTO clause. This clause can be used in three ways.

Advanced Examples

Create macro variables with SQL

Method 1:

```
SELECT col1, col2, ...  
INTO :mvar1, :mvar2, ...  
FROM ...
```

Method 1 extracts values **only** from the **first** row of the query result.

Advanced Examples

Create macro variables with SQL

Method 1

```
select avg(salary) ,  
       min(salary) ,  
       max(salary)  
into :mean, :min, :max  
from airline.payrollmaster;  
%put &mean &min &max;
```

Log

54079.65	25120.2	155930.6
----------	---------	----------

Advanced Examples

Create macro variables with SQL

Method 2

```
SELECT a, b, ...  
      INTO :a1-:an, :b1-:bn  
      FROM ...
```

Method 2 extracts values from the first n rows of the query result, and puts them into a series of n macro variables.

Advanced Examples

Create macro variables with SQL

Method 2

```
select distinct destination  
    into :destlist1 - :destlist9  
    from demodata.marchflights;  
%put &destlist1  
%put &destlist2  
%put &destlist3;
```

Log

CDG
CPH
DFW

Advanced Examples

Create macro variables with SQL

Method 3

```
SELECT col1, col2, ...  
      INTO :macrovar1, :macrovar2, ...  
          SEPARATED BY 'delimiter'  
FROM ...
```

Method 3 extracts values from all rows of the query result, and puts them into a single macro variable, separated by the specified delimiter.

Advanced Examples

Create macro variables with SQL

Method 3

```
select distinct destination into :destlist  
    separated by ' '  
    from demodata.marchflights;  
%put &destlist;
```

Log

CDG CPH DFW FRA LAX LHR ORD WAS YYZ

Advanced Examples

Create macro variables with SQL

Example:

FlightNumber	Date	DepartureTime	Origin	Destination	Distance
182	01MAR2000	8:21	LGA	YYZ	365
114	01MAR2000	7:10	LGA	LAX	247
202	01MAR2000	10:43	LGA	ORD	74
219	01MAR2000	9:31	LGA	LHR	344
439	01MAR2000	12:16	LGA	LAX	247
387	01MAR2000	11:40	LGA	CPH	385
290	01MAR2000	6:56	LGA	WAS	22
523	01MAR2000	15:19	LGA	ORD	74
982	01MAR2000	10:28	LGA	DFW	136
622	01MAR2000	12:19	LGA	FRA	385

**DFW
dataset**

**CPH
dataset**

**LAX
dataset**

**LHR
dataset**

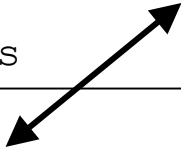
**FRA
dataset**

Etcetera ...

Advanced Examples

Create macro variables with SQL

```
Select distinct destination into :destlist  
separated by ' '  
from demodata.marchflights
```



```
data CDG CPH DFW FRA LAX LHR ORD WAS YYZ;  
set demodata.marchflights;  
  if destination='CPH'  
    then output CPH;  
  else if destination='CPG'  
    then output CPG;  
  else if destination='DFW'  
    then output DFW;  
  ...  
  ...  
run;
```

```
Select distinct destination  
      into :dest1-:dest9  
      from demodata.marchflights
```

Create macro variables with SQL

```
data CDG CPH DFW FRA LAX LHR ORD WAS YYZ;  
set demodata.marchflights;  
if destination='CPH'  
then output CPH;  
else if destination='CPG'  
then output CPG;  
else if destination='DFW'  
then output DFW;  
...  
...  
run;
```

Select distinct destination into **:destlist**
from demodata.marchflights

```
data &destlist;  
set demodata.marchflights;  
if destination="&dest1"  
then output &dest1;  
else if destination="&dest2"  
then output &dest2;  
else if destination="&dest3"  
then output &dest3;  
...  
...  
run;
```

Select distinct destination
into **:dest1-:dest9**
from demodata.marchflights

SAS Log:

```
proc sql;
  select distinct destination into :destlist
    separated by ' '
    from demodata.marchflights
    where destination is not missing;
  %put Destination list is: &destlist;
Destination list is: CDG CPH DFW FRA LAX LHR ORD WAS YYZ

  select count(distinct destination) into :count
    from demodata.marchflights
    where destination is not missing;
  %let count=%left(&count);
  %put count=&count;
count=9
  select distinct destination
    into :dest1-:dest&count
    from demodata.marchflights
    where destination is not missing;
  %put dest1=&dest1;
dest1=CDG
  %put dest2=&dest2;
dest2=CPH
  quit;
NOTE: PROCEDURE SQL used:
      real time          0.04 seconds
      cpu time           0.04 seconds
```

```

117 %macro parsout;
118     data &destlist;
119         set demodata.marchflights;
120         if destination="&dest1"
121             then output &dest1;
122     %do i=2 %to &count;
123         else if destination = "&&dest&i"
124             then output &&dest&i;
125     %end;
126 run;
127 %mend;
128 %parsout;

```

NOTE: There were 635 observations read from the data set DEMODATA.MARCHFLIGHTS.

NOTE: The data set WORK.CDG has 27 observations and 13 variables.

NOTE: The data set WORK.CPH has 27 observations and 13 variables.

NOTE: The data set WORK.DFW has 62 observations and 13 variables.

NOTE: The data set WORK.FRA has 27 observations and 13 variables.

NOTE: The data set WORK.LAX has 123 observations and 13 variables.

NOTE: The data set WORK.LHR has 58 observations and 13 variables.

NOTE: The data set WORK.ORD has 93 observations and 13 variables.

NOTE: The data set WORK.WAS has 155 observations and 13 variables.

NOTE: The data set WORK.YYZ has 62 observations and 13 variables.

NOTE: DATA statement used:

real time	0.56 seconds
-----------	--------------

cpu time	0.38 seconds
----------	--------------

Advanced Examples

Indexing

Objective: To increase speed for inner join of a small table(5 rows) and a large table (1,000,000 rows)

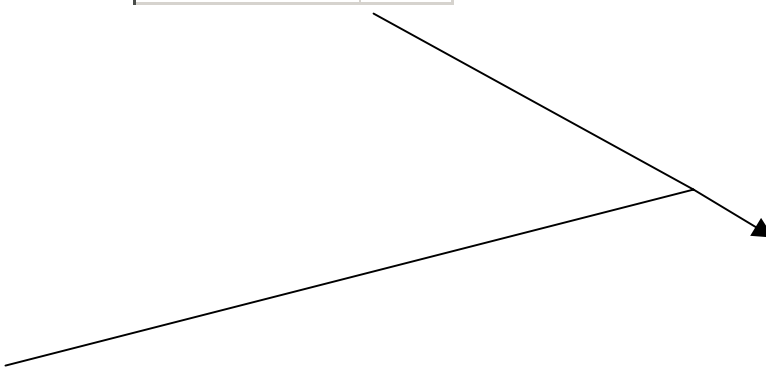
Demodata.large

id	salary
1	57398
2	88803
3	65992
4	60375
5	86864
6	88771
7	71719
8	71267
9	51991
10	52662
11	82772
12	70954
13	84135
14	52687
15	88280
16	61887
17	60904
18	77597
19	89070
20	59060
21	77529

Demodata.small

id	name
13	Lee
205	Wil
3187	Fer
29999	Nel
497981	Yel

```
select large.id, name, salary
from demodata.large, demodata.small
where large.id=small.id;
```



id	name	salary
13	Lee	84135
205	Wil	86422
3187	Fer	69954
29999	Nel	83184
497981	Yel	74666

Advanced Examples

Indexing

Demodata.large

id	salary
1	57398
2	88803
3	65992
4	60375
5	86864
6	88771
7	71719
8	71267
9	51991
10	52662
11	82772
12	70954
13	84135
14	52687
15	88280
16	61887
17	60904
18	77597
19	89070
20	59060
21	77529

Demodata.small

id	name
13	Lee
205	Wil
3187	Fer
29999	Nel
497981	Yel

Without indexing, SAS must process all rows in demodata.large dataset to find matching ID values.

With indexing, SAS could go directly to the rows in demodata.large that contain the ID values 13,205,3187, 29999, and 497981

Advanced Examples

Indexing

An *index* is an auxiliary data structure that specifies the location of rows based on the values of one or more **key** columns.

You can use indexes for subsetting, grouping, and joining tables.

Advanced Examples

Indexing

Indexed SAS Data Set

Row	EmpID	Gender	Jobcode
1	1001	F	FA1
2	1012	F	FA3
3	1015	M	FA2
.			
.			
.			
11	1104	M	FA3
.			
.			
.			

DATA or PROC Step
where Jobcode='FA3' ;

Index File

Key Column=Jobcode			
Key	Location		
Value	Page (row, row...)		
FA1	1 (1, 4, ...)	2 (...)	...
FA2	1 (3, 6, ...)	2 (...)	...
FA3	1 (2, 11, ...)	2 (...)	...

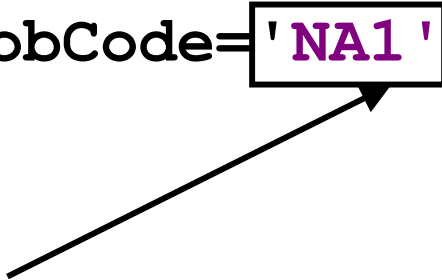
Data Processed			
ROW	EmpID	Gender	Jobcode
2	1012	F	FA3
11	1104	M	FA3
		.	
		.	
		.	

Advanced Examples

Indexing

Indexes provide fast access to small subsets of data...

```
proc sql;  
  select *  
    from airline.payrollmaster  
   where JobCode='NA1';
```



One of many values
of the variable JobCode

Advanced Examples

Indexing

... and also enhance join performance.

```
proc sql;  
    select *  
        from demodata.large, demodata.small  
        where large.ID=small.ID;
```

The user creates the index for the variable in the dataset. Once the index is created, the SQL Procedure will use the index automatically where appropriate.

Advanced Examples

Indexing

```
proc sql;  
  create index id  
    on demodata.large(id) ;  
  
  select large.id, name, salary  
    from demodata.large, demodata.small  
     where large.id=small.id;  
quit;
```

Advanced Examples

Indexing

Join Without index

```
156  proc sql stimer;  
157      select large.id, name, salary  
158          from demodata.large, demodata.small  
159          where large.id=small.id;
```

NOTE: SQL Statement used:

real time 1.17 seconds

cpu time 1.17 seconds

Create index

```
160  create index id  
161      on demodata.large(id);
```

NOTE: Simple index id has been defined.

Join with index

```
162  select large.id, name, salary  
163      from demodata.large, demodata.small  
164      where large.id=small.id;
```

NOTE: SQL Statement used:

real time 0.01 seconds

cpu time 0.01 seconds

Advanced Examples

Dictionary Tables

You can retrieve information about SAS session metadata by querying *dictionary tables* with PROC SQL. Dictionary tables are

- created at initialization
- updated automatically
- limited to read-only access.

Advanced Examples

Dictionary Tables

The metadata available in dictionary tables includes

- SAS files
- external files
- system options, macros, titles, and footnotes.

Advanced Examples

Dictionary Tables

SAS File Metadata

DICTIONARY.MEMBERS	general information about data library members
DICTIONARY.TABLES	detailed information about data sets
DICTIONARY.COLUMNS	detailed information on variables and their attributes
DICTIONARY.CATALOGS	information about catalog entries
DICTIONARY.VIEWS	general information about data views
DICTIONARY.INDEXES	information on indexes defined for data files

Advanced Examples

Dictionary Tables

Other Metadata

DICTIONARY.EXTFILES	currently assigned filerefs
DICTIONARY.OPTIONS	current settings of SAS system options
DICTIONARY.MACROS	information about macro variables
DICTIONARY.TITLES	text assigned to titles and footnotes

Advanced Examples

Dictionary Tables

Exploring Dictionary Tables

describe table dictionary.tables;

Partial Log

```
create table DICTIONARY.TABLES (  
  libname char(8) label='Library Name',  
  memname char(32) label='Member Name',  
  memtype char(8) label='Member Type',  
  memlabel char(256) label='Dataset Label',  
  typemem char(8) label='Dataset Type',  
  crdate num format=DATETIME informat=DATETIME  
  label='Date Created', ...);
```

Advanced Examples

Dictionary Tables

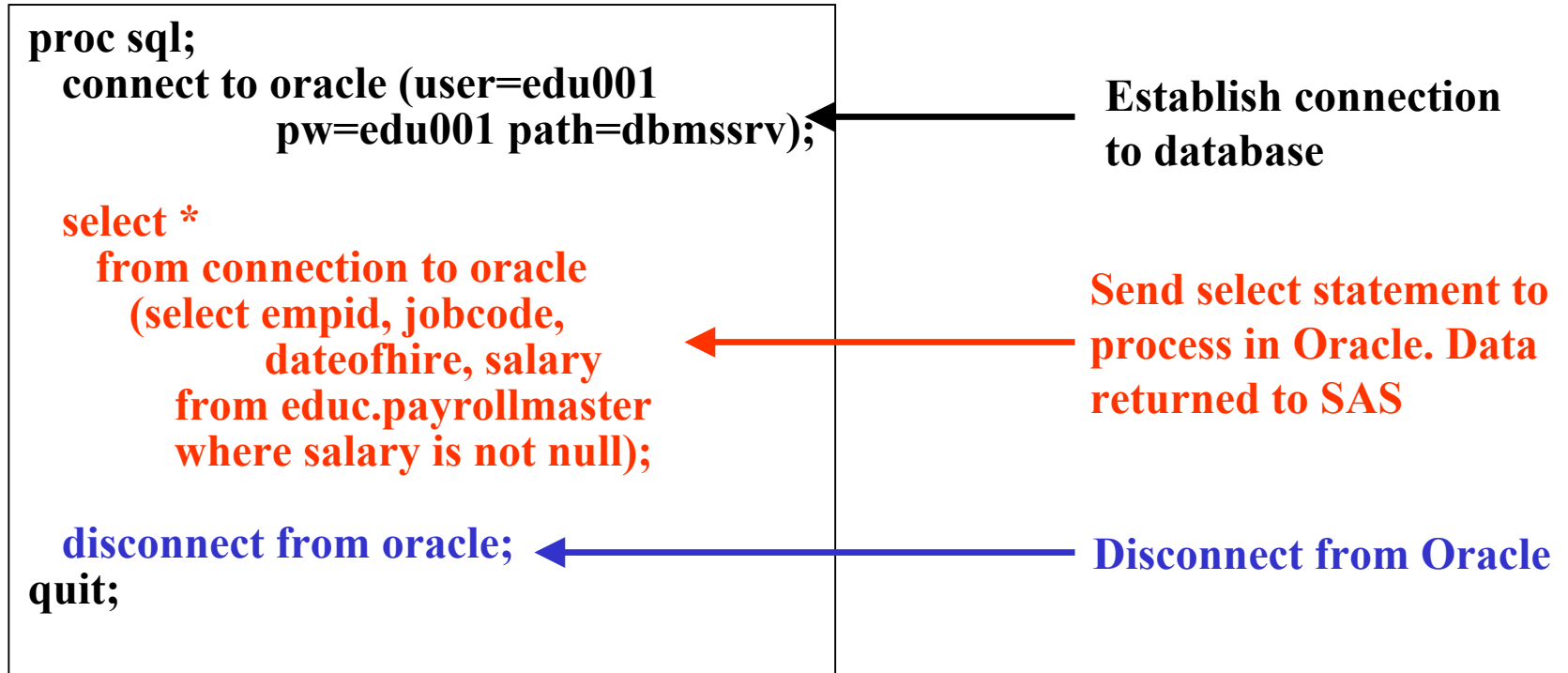
Using Dictionary Information

Example: Display information about the files in the DEMODATA library.

```
select memname format=$20.,  
       nobs,  
       nvar,  
       crdate  
from dictionary.tables  
where libname='DEMODATA' ;
```

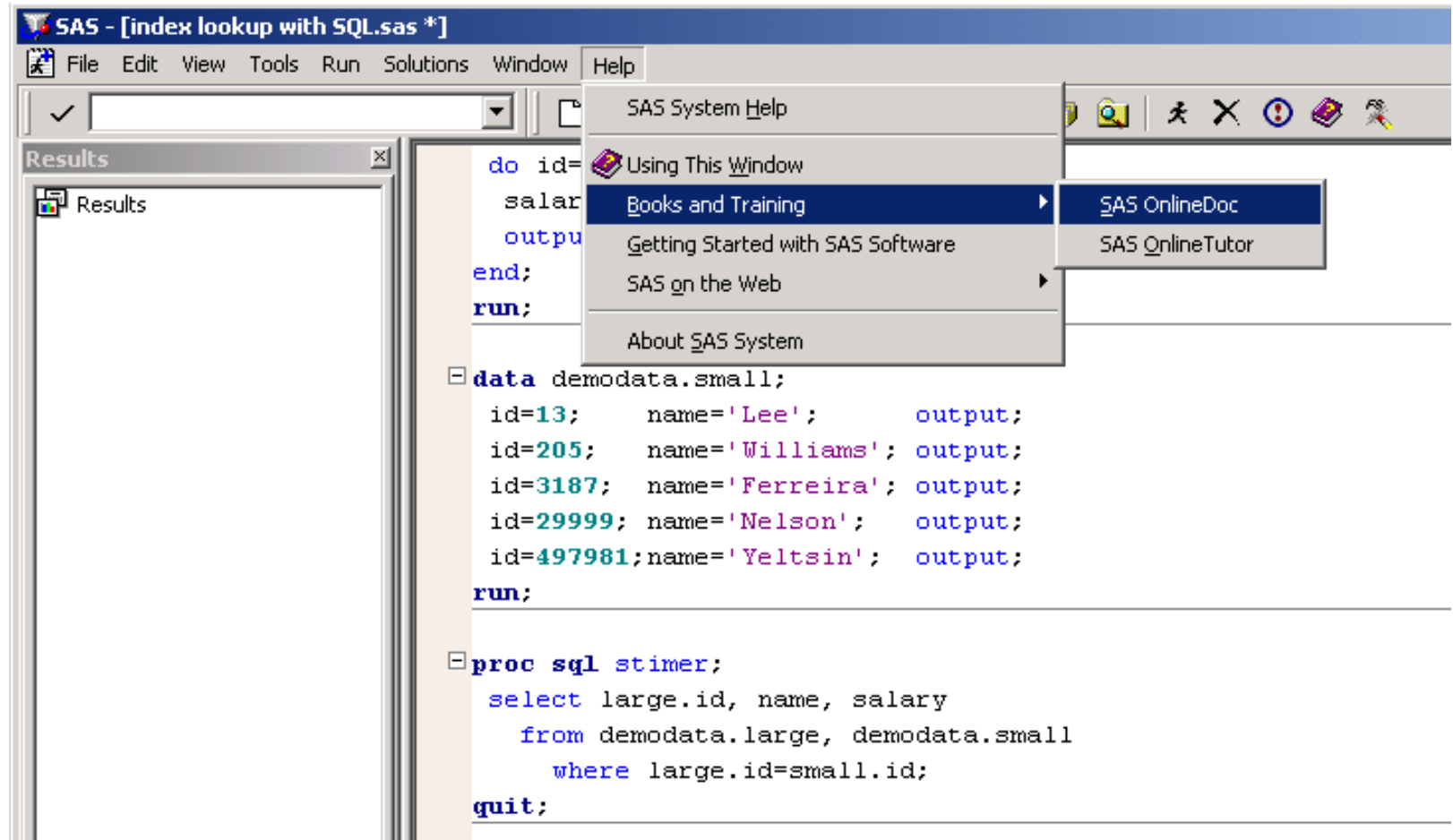
Advanced Examples

SQL Pass-through



- from SAS session, send SQL to database for processing
- results returned to SAS for reporting, or store as a dataset

Online Help



Online Help

SAS OnlineDoc, V8 - MSIE provided by SAS Institute Canada

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Media

Address <C:\Program Files\SASOnlineDocV8\sasdoc\sashtml\onldoc.htm>

Links [Customize Links](#) [Windows](#) [RFS Helpdesk](#) [Outlook Web Access](#) [SAS Canada InfoSite](#)

Contents Index Search Help

- [-] SAS OnlineDoc
 - Overview
 - + What's New in SAS Software for V8
 - Hardcopy Documentation Update
 - [-] Base SAS Software
 - + SAS Language Reference: Conventions
 - + SAS Language Reference: Dictionary
 - + SAS Macro Language Reference
 - + Moving and Accessing SAS Files
 - [-] SAS Procedures Guide
 - + Changes and Enhancement
 - + Concepts
 - + **Procedures**
 - + Appendices
 - + Guide to the SAS Output Delivery System
 - + SAS SQL Query Window User's Guide
 - + Host Specific Information
 - + SAS/ACCESS
 - + SAS/AF

[The OPTIONS Procedure](#)

[The PLOT Procedure](#)

[The PMENU Procedure](#)

[The PRINT Procedure](#)

[The PRINTTO Procedure](#)

[The RANK Procedure](#)

[The REGISTRY Procedure](#)

[The REPORT Procedure](#)

[The SORT Procedure](#)

[The SQL Procedure](#)

[The STANDARD Procedure](#)

[The SUMMARY Procedure](#)

[The TABULATE Procedure](#)

[The TIMEPLOT Procedure](#)

Courses

Many courses offered including:

SQL Processing with the SAS System

Programming 1: Essentials

Programming 2: Manipulating Data with the Data Step
offered in Vancouver March 24-26

Programming 3: Advanced Techniques
offered in Vancouver March 29-30

For course information and registration

www.sas.com/training

or call

Gloria Pierre 416 307-4535