PROC SQL or PROC FedSQL:

Which Should a Programmer Use?

T Winand

Senior Solutions Architect

SAS Global Customer Success Organization

What We Will Cover

Guidance to Answer the Questions:
- When do I use PROC FedSQL?
- When does PROC FedSQL offer benefits over PROC SQL?

Who will benefit?
- SAS®9 Programmers
- SAS® Viya® Programmers
- Anyone who accesses and queries data using DATA step or PROC SQL.

What Will You Learn?
- When and how to use PROC FedSQL.
- When to use PROC FedSQL versus PROC SQL or DATA step.
- The differences of running PROC FedSQL in SAS®9 (or the Viya Compute Server) versus in the CAS Server.

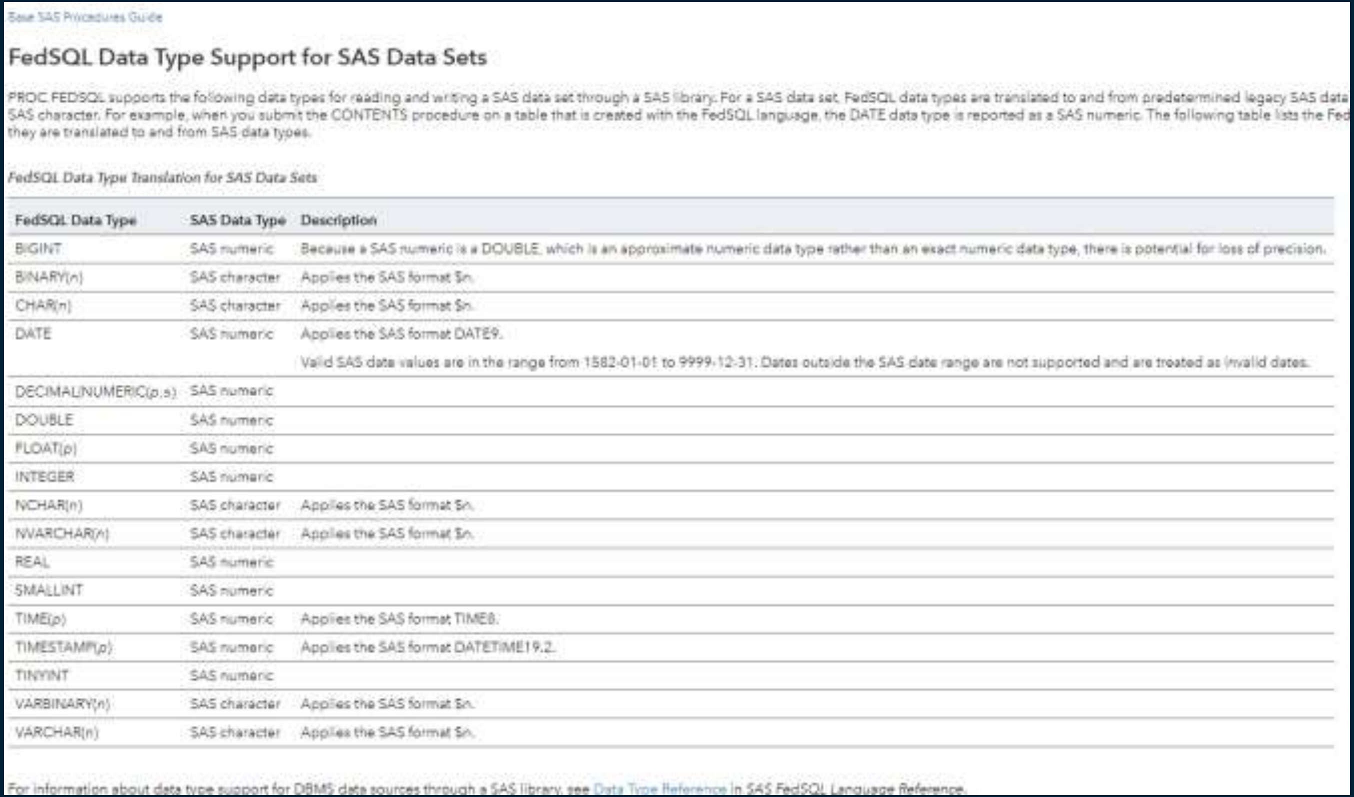
Introduction to the FedSQL Language

- SAS FedSQL is a SAS proprietary implementation of ANSI SQL:1999 core standard.
- Provides support for new data types.
- Scalable, high-performance access
- Common SQL (vendor neutral) syntax

Benefits of FedSQL

- Benefits:
  - Able to process queries in its own language
  - Able to process queries in native languages of other data sources
  - Supports more data types for greater precision
  - Handles federated queries
  - Can create data in any of the supported data sources

FedSQL Data Type Support



Base SAS Procedures Guide

FedSQL Data Type Support for SAS Data Sets

PROC FEDSQL supports the following data types for reading and writing a SAS data set through a SAS library. For a SAS data set, FedSQL data types are translated to and from predetermined legacy SAS data
SAS character. For example, when you submit the CONTENTS procedure on a table that is created with the FedSQL language, the DATE data type is reported as a SAS numeric. The following table lists the Fed
they are translated to and from SAS data types.

FedSQL Data Type Translation for SAS Data Sets

FedSQL Data Type	SAS Data Type	Description
BIGINT	SAS numeric	Because a SAS numeric is a DOUBLE, which is an approximate numeric data type rather than an exact numeric data type, there is potential for loss of precision.
BINARY(*n*)	SAS character	Applies the SAS format $*n*.
CHAR(*n*)	SAS character	Applies the SAS format $*n*.
DATE	SAS numeric	Applies the SAS format DATE9. Valid SAS date values are in the range from 1582-01-01 to 9999-12-31. Dates outside the SAS date range are not supported and are treated as invalid dates.
DECIMAL|NUMERIC(*p*,*s*)	SAS numeric	
DOUBLE	SAS numeric	
FLOAT(*p*)	SAS numeric	
INTEGER	SAS numeric	
NCHAR(*n*)	SAS character	Applies the SAS format $*n*.
NVARCHAR(*n*)	SAS character	Applies the SAS format $*n*.
REAL	SAS numeric	
SMALLINT	SAS numeric	
TIME(*p*)	SAS numeric	Applies the SAS format TIME8.
TIMESTAMP(*p*)	SAS numeric	Applies the SAS format DATETIME19.2.
TINYINT	SAS numeric	
VARBINARY(*n*)	SAS character	Applies the SAS format $*n*.
VARCHAR(*n*)	SAS character	Applies the SAS format $*n*.

For information about data type support for DBMS data sources through a SAS library, see Data Type Reference in SAS FedSQL Language Reference.

[FedSQL Data Type Support](#)

Federated Queries

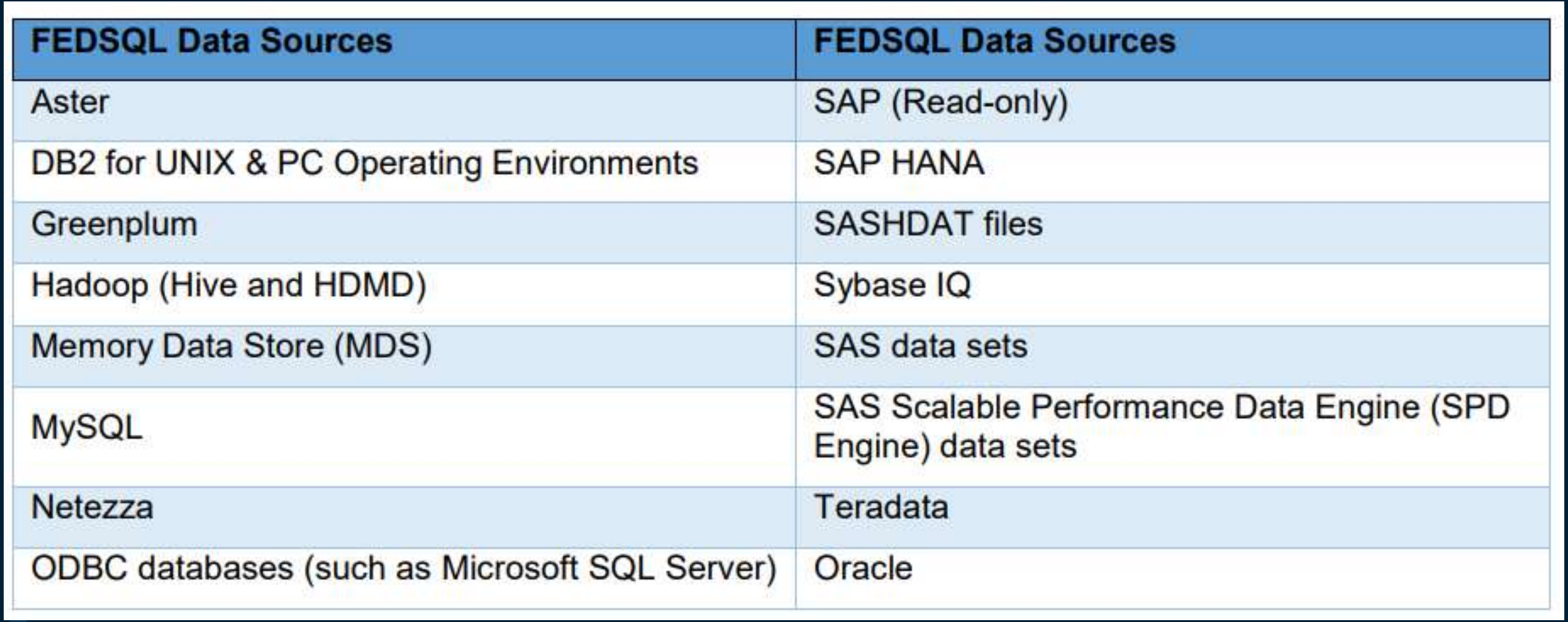
```
select
  Ora1.city Ora1.State, Ora1.zip from Oracle.Tbl1 Ora1,
  Teradata.Tbl2 Tera2,
  Teradata.Tbl3 Tera3
  where Ora1.zip = Tera2.zip and Tera2.zip = Tera3.zip;
```

Running FedSQL Programs

You can submit FedSQL programs in the following ways:

- Using the FedSQL procedure in SAS programs *Base SAS Procedures Guide*
- Using FedSQL.execDirect action in CAS (Viya) *SAS Viya: System Programming Guide*
- From a JDBC, ODBC or OLE DB client by using SAS Federation Server *SAS Federation Server: Administrator's Guide.*
- Using a SAS LIBNAME engine for the SAS Federation Server *SAS LIBNAME Engine for SAS Federation Server: User's Guide*
- From a DS2 Program Using FedSQL and DS2

Data Source Support



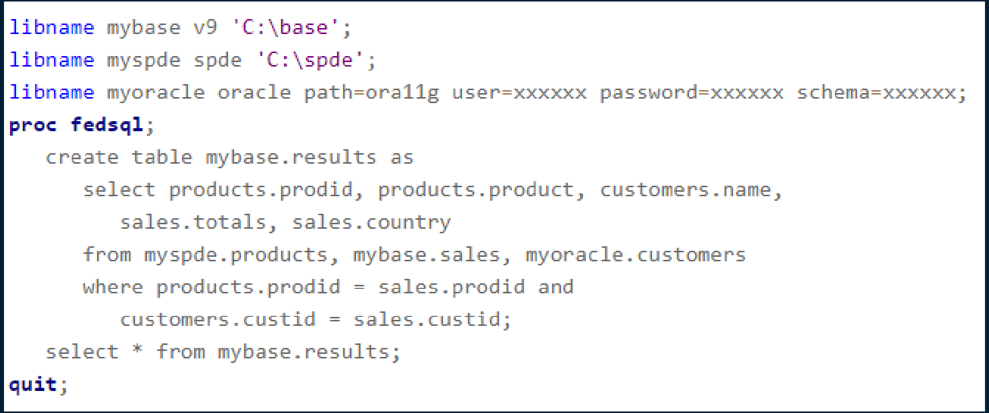
FEDSQL Data Sources	FEDSQL Data Sources
Aster	SAP (Read-only)
DB2 for UNIX & PC Operating Environments	SAP HANA
Greenplum	SASHDAT files
Hadoop (Hive and HDMD)	Sybase IQ
Memory Data Store (MDS)	SAS data sets
MySQL	SAS Scalable Performance Data Engine (SPD Engine) data sets
Netezza	Teradata
ODBC databases (such as Microsoft SQL Server)	Oracle

FedSQL Procedure

Data Source Connection

Establish a connection to a data source

- FedSQL Procedure - connection string generated from attributes of currently assigned librefs



```
libname mybase v9 'C:\base';
libname myspde spde 'C:\spde';
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx schema=xxxxxx;
proc fedsql;
   create table mybase.results as
      select products.prodid, products.product, customers.name,
         sales.totals, sales.country
      from myspde.products, mybase.sales, myoracle.customers
      where products.prodid = sales.prodid and
         customers.custid = sales.custid;
   select * from mybase.results;
quit;
```

- SAS Federation Server - LIBNAME engine obtains a data source connection by connecting to a SAS Federation Server and specifying a DSN.

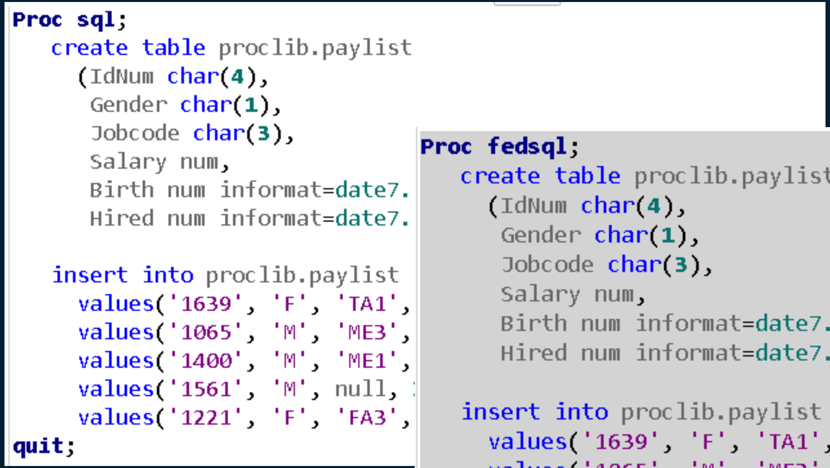
FedSQL and SAS Cloud Analytic Services

- FedSQL functionality in CAS is limited
- FedSQL supports same functions and formats in CAS, SAS 9.4, and SAS Viya

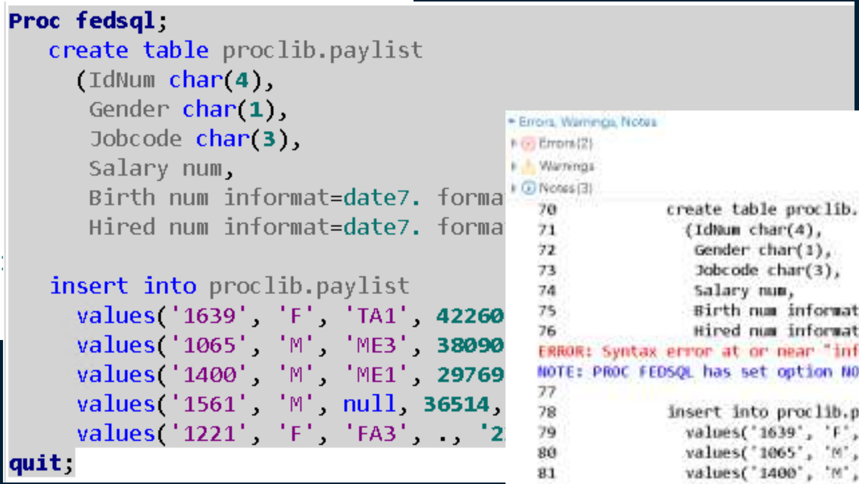
SAS Viya: FedSQL Programming for SAS Cloud Analytic Services

Performing PROC SQL Tasks with FedSQL

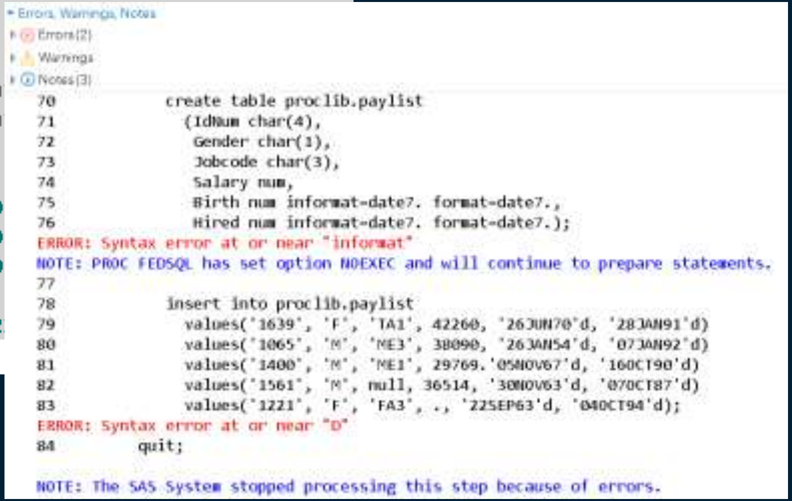
Can I just add “Fed” and use the same code?



```
Proc sql;
   create table proclib.paylist
     (IdNum char(4),
      Gender char(1),
      Jobcode char(3),
      Salary num,
      Birth num informat=date7.
      Hired num informat=date7.

   insert into proclib.paylist
     values('1639', 'F', 'TA1',
     values('1065', 'M', 'ME3',
     values('1400', 'M', 'ME1',
     values('1561', 'M', null,
     values('1221', 'F', 'FA3',
quit;
```



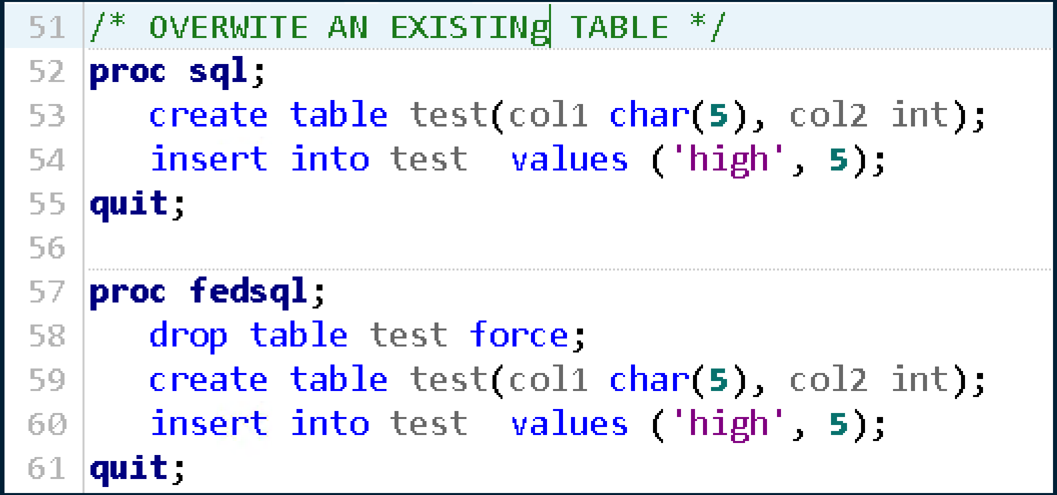
```
Proc fedsql;
   create table proclib.paylist
     (IdNum char(4),
      Gender char(1),
      Jobcode char(3),
      Salary num,
      Birth num informat=date7. forma
      Hired num informat=date7. forma

   insert into proclib.paylist
     values('1639', 'F', 'TA1', 42260
     values('1065', 'M', 'ME3', 38090
     values('1400', 'M', 'ME1', 29769
     values('1561', 'M', null, 36514,
     values('1221', 'F', 'FA3', ., '2
quit;
```



```
70             create table proclib.paylist
71               (IdNum char(4),
72                Gender char(1),
73                Jobcode char(3),
74                Salary num,
75                Birth num informat=date7. format=date7.,
76                Hired num informat=date7. format=date7.);
ERROR: Syntax error at or near "informat"
NOTE: PROC FEDSQL has set option NOEXEC and will continue to prepare statements.
77
78             insert into proclib.paylist
79               values('1639', 'F', 'TA1', 42260, '26JUN70'd, '28JAN91'd)
80               values('1065', 'M', 'ME3', 38090, '26JAN54'd, '07JAN92'd)
81               values('1400', 'M', 'ME1', 29769.'05NOV67'd, '16OCT90'd)
82               values('1561', 'M', null, 36514, '30NOV63'd, '07OCT87'd)
83               values('1221', 'F', 'FA3', ., '22SEP63'd, '04OCT94'd);
ERROR: Syntax error at or near "D"
84          quit;

NOTE: The SAS System stopped processing this step because of errors.
```

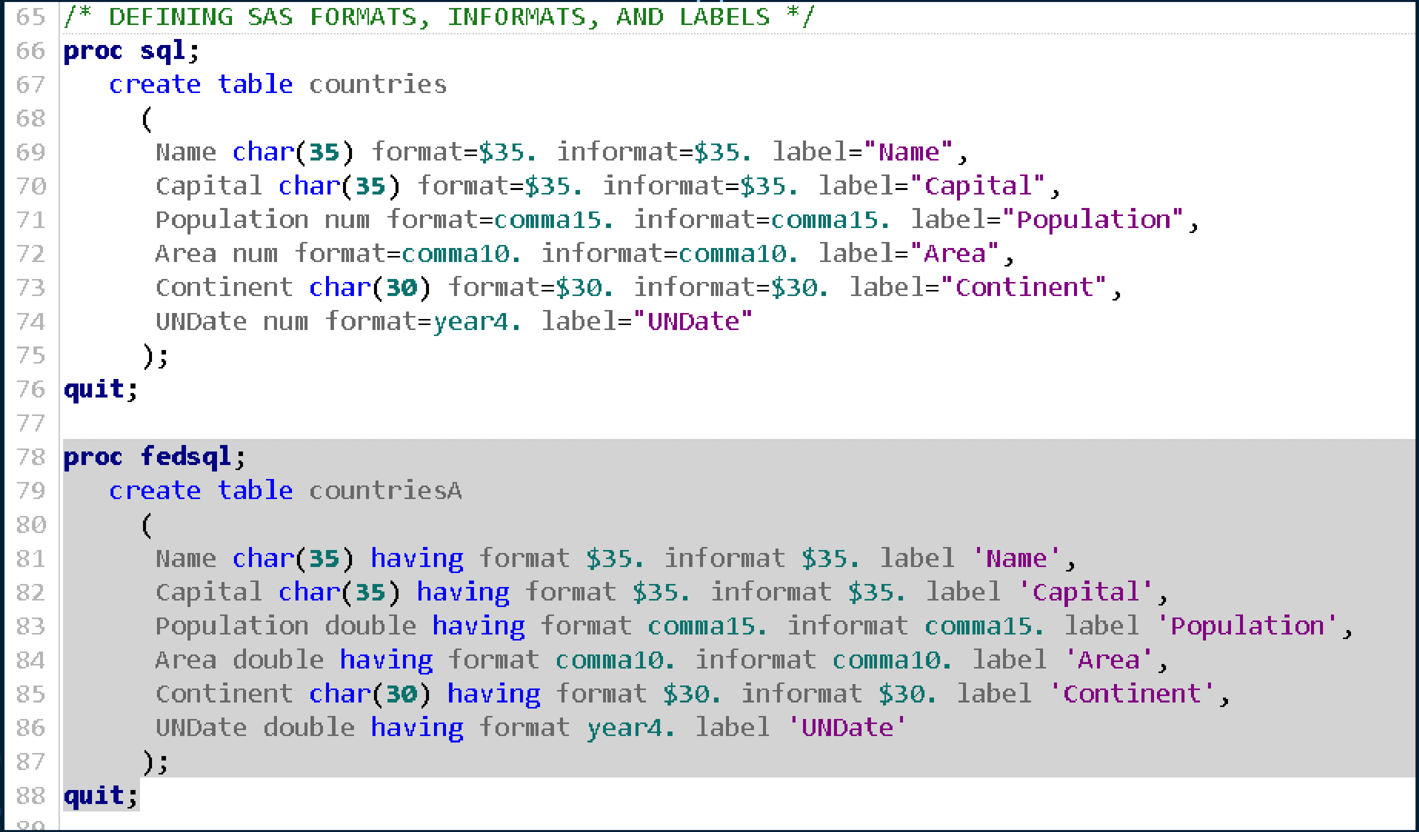
Performing PROC SQL Tasks with FedSQL

Overwriting an Existing Table



```
/* OVERWITE AN EXISTING TABLE */
proc sql;
   create table test(col1 char(5), col2 int);
   insert into test  values ('high', 5);
quit;

proc fedsql;
   drop table test force;
   create table test(col1 char(5), col2 int);
   insert into test  values ('high', 5);
quit;
```

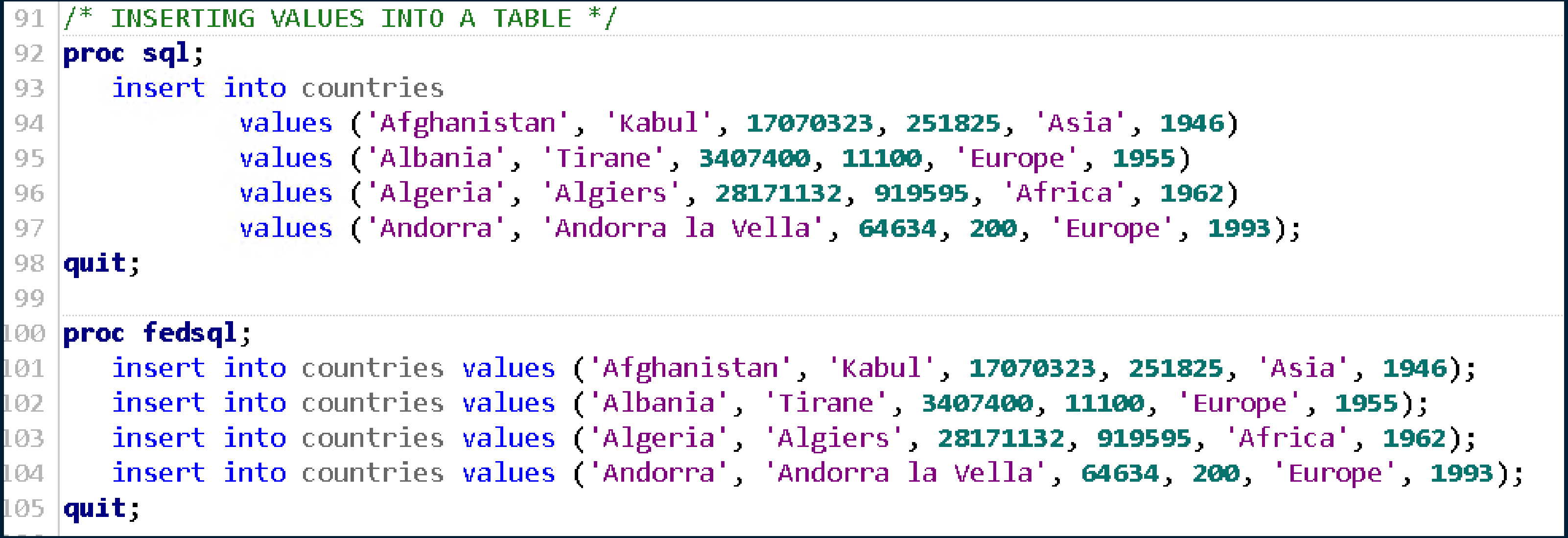
Performing PROC SQL Tasks with FedSQL



```
/* DEFINING SAS FORMATS, INFORMATS, AND LABELS */
proc sql;
   create table countries
     (
      Name char(35) format=$35. informat=$35. label="Name",
      Capital char(35) format=$35. informat=$35. label="Capital",
      Population num format=comma15. informat=comma15. label="Population",
      Area num format=comma10. informat=comma10. label="Area",
      Continent char(30) format=$30. informat=$30. label="Continent",
      UNDate num format=year4. label="UNDate"
     );
quit;

proc fedsql;
   create table countriesA
     (
      Name char(35) having format $35. informat $35. label 'Name',
      Capital char(35) having format $35. informat $35. label 'Capital',
      Population double having format comma15. informat comma15. label 'Population',
      Area double having format comma10. informat comma10. label 'Area',
      Continent char(30) having format $30. informat $30. label 'Continent',
      UNDate double having format year4. label 'UNDate'
     );
quit;
```

Performing PROC SQL Tasks with FedSQL

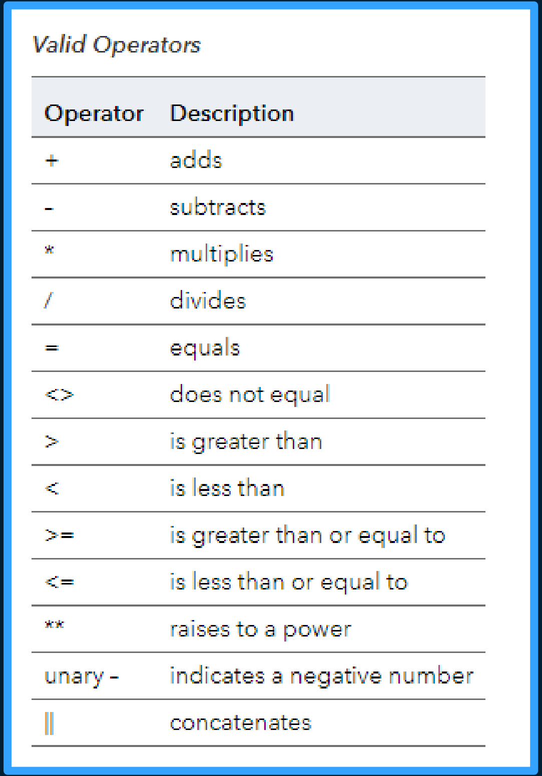
Inserting Values into a Table



```
/* INSERTING VALUES INTO A TABLE */
proc sql;
   insert into countries
          values ('Afghanistan', 'Kabul', 17070323, 251825, 'Asia', 1946)
          values ('Albania', 'Tirane', 3407400, 11100, 'Europe', 1955)
          values ('Algeria', 'Algiers', 28171132, 919595, 'Africa', 1962)
          values ('Andorra', 'Andorra la Vella', 64634, 200, 'Europe', 1993);
quit;

proc fedsql;
   insert into countries values ('Afghanistan', 'Kabul', 17070323, 251825, 'Asia', 1946);
   insert into countries values ('Albania', 'Tirane', 3407400, 11100, 'Europe', 1955);
   insert into countries values ('Algeria', 'Algiers', 28171132, 919595, 'Africa', 1962);
   insert into countries values ('Andorra', 'Andorra la Vella', 64634, 200, 'Europe', 1993);
quit;
```

Performing PROC SQL Tasks with FedSQL

Use of Comparison Operators



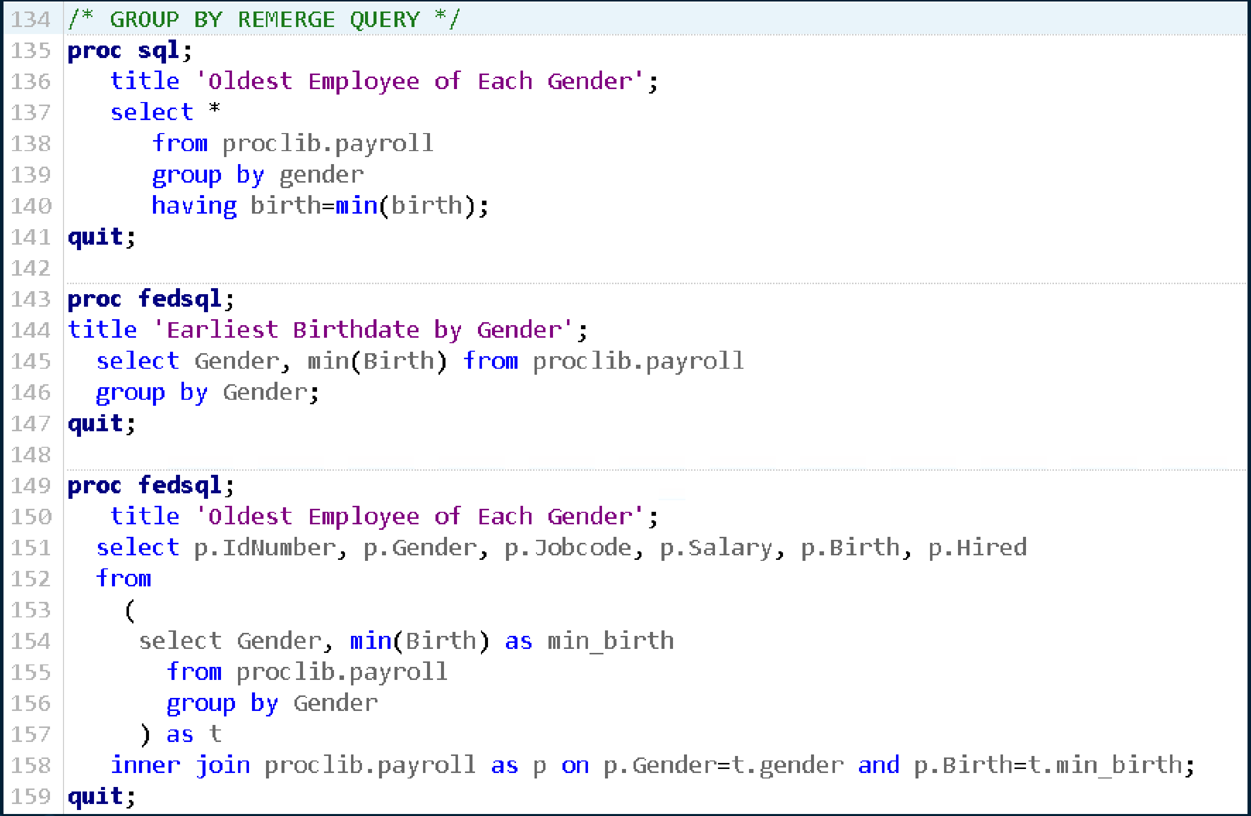
Valid Operators

Operator	Description
+	adds
-	subtracts
*	multiplies
/	divides
=	equals
<>	does not equal
>	is greater than
<	is less than
>=	is greater than or equal to
<=	is less than or equal to
**	raises to a power
unary -	indicates a negative number
||	concatenates

<sql-expression> Expression

Performing PROC SQL Tasks with FedSQL

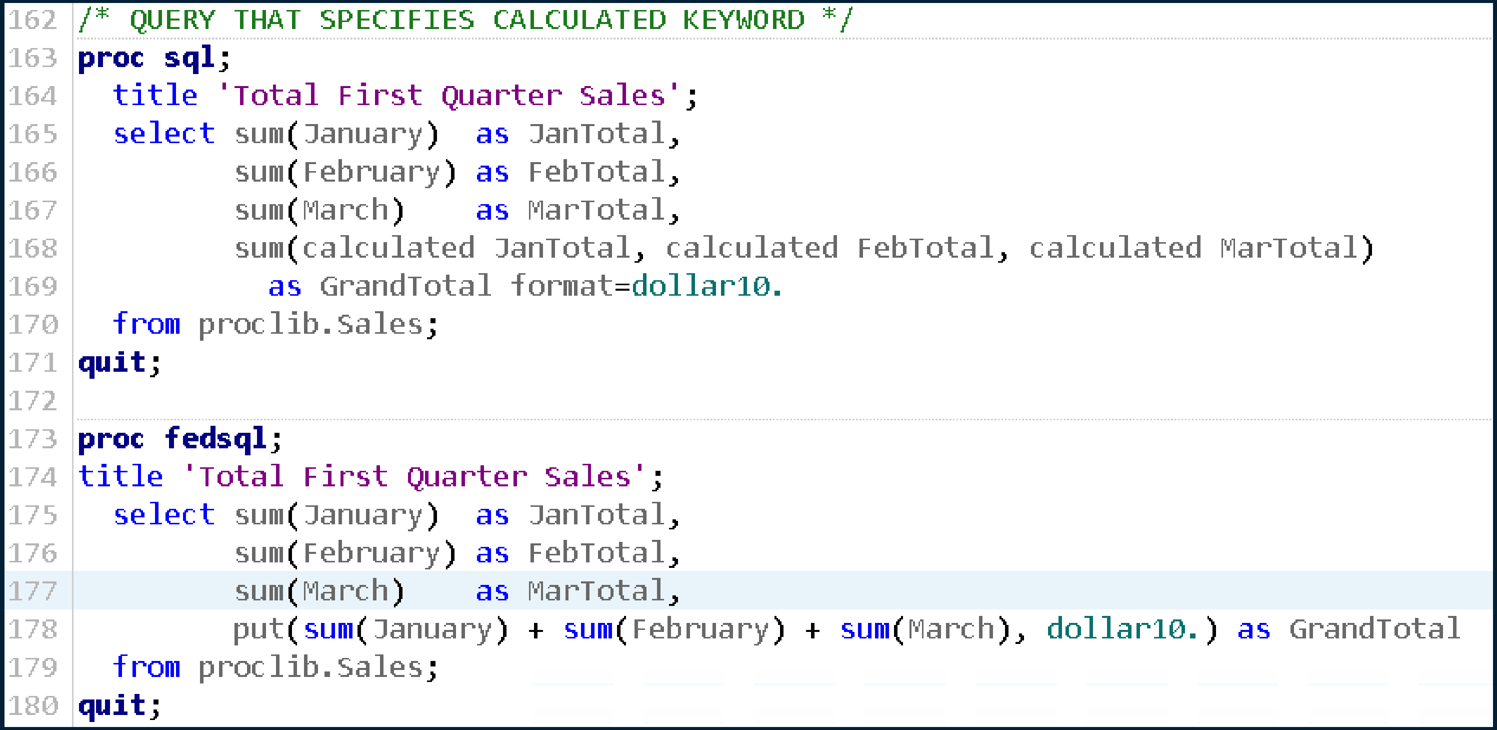
Group By Remerge Query



```
/* GROUP BY REMERGE QUERY */
proc sql;
   title 'Oldest Employee of Each Gender';
   select *
      from proclib.payroll
      group by gender
      having birth=min(birth);
quit;

proc fedsql;
title 'Earliest Birthdate by Gender';
  select Gender, min(Birth) from proclib.payroll
  group by Gender;
quit;

proc fedsql;
   title 'Oldest Employee of Each Gender';
  select p.IdNumber, p.Gender, p.Jobcode, p.Salary, p.Birth, p.Hired
  from
    (
     select Gender, min(Birth) as min_birth
       from proclib.payroll
       group by Gender
     ) as t
   inner join proclib.payroll as p on p.Gender=t.gender and p.Birth=t.min_birth;
quit;
```

Performing PROC SQL Tasks with FedSQL

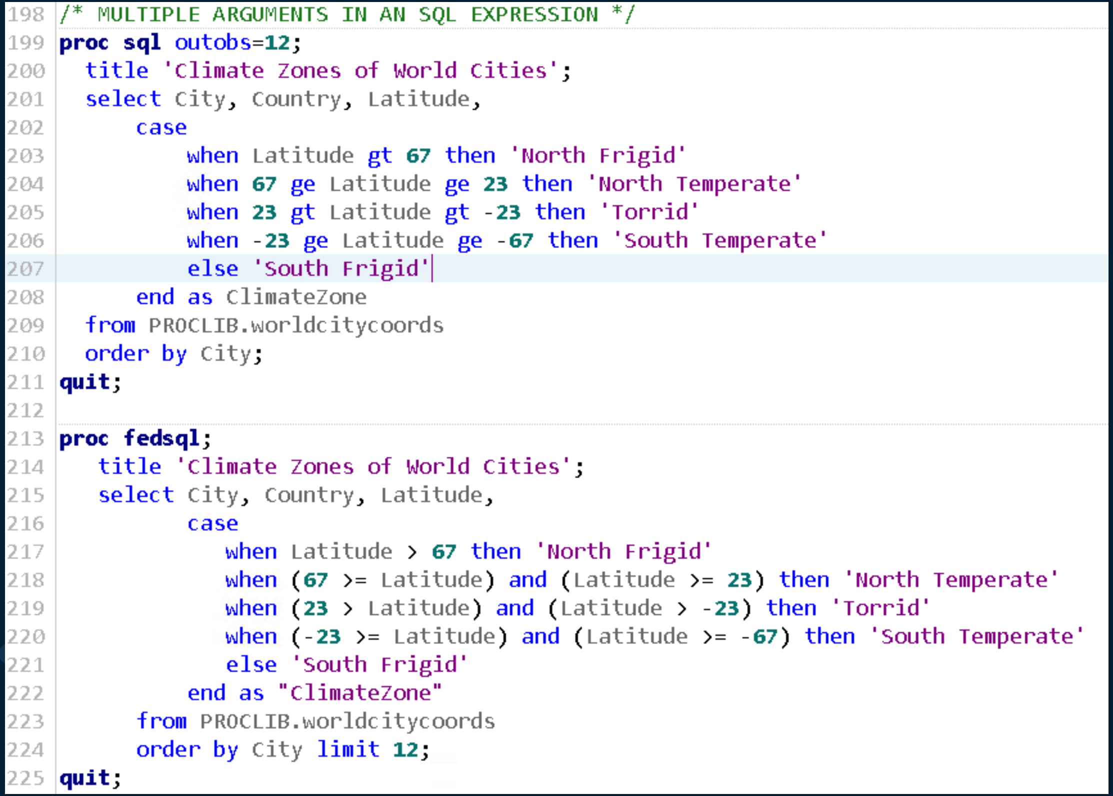
Query the Specifies CALCULATED Keyword



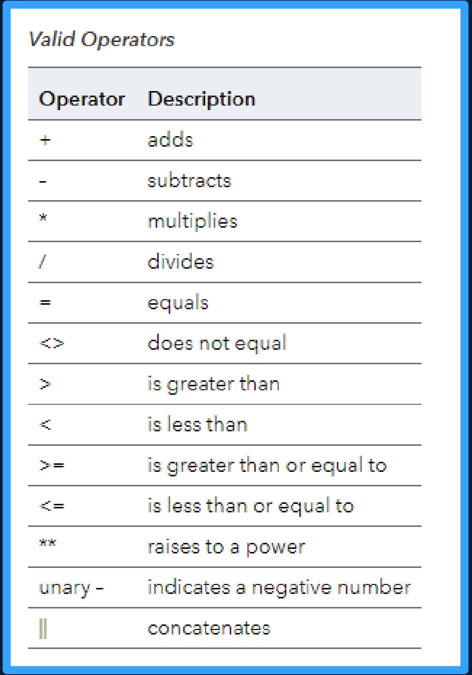
```
/* QUERY THAT SPECIFIES CALCULATED KEYWORD */
proc sql;
  title 'Total First Quarter Sales';
  select sum(January)  as JanTotal,
         sum(February) as FebTotal,
         sum(March)    as MarTotal,
         sum(calculated JanTotal, calculated FebTotal, calculated MarTotal)
           as GrandTotal format=dollar10.
  from proclib.Sales;
quit;

proc fedsql;
title 'Total First Quarter Sales';
  select sum(January)  as JanTotal,
         sum(February) as FebTotal,
         sum(March)    as MarTotal,
         put(sum(January) + sum(February) + sum(March), dollar10.) as GrandTotal
  from proclib.Sales;
quit;
```

Performing PROC SQL Tasks with FedSQL

Specifying Multiple Arguments



```
/* MULTIPLE ARGUMENTS IN AN SQL EXPRESSION */
proc sql outobs=12;
  title 'Climate Zones of World Cities';
  select City, Country, Latitude,
      case
          when Latitude gt 67 then 'North Frigid'
          when 67 ge Latitude ge 23 then 'North Temperate'
          when 23 gt Latitude gt -23 then 'Torrid'
          when -23 ge Latitude ge -67 then 'South Temperate'
          else 'South Frigid'
      end as ClimateZone
  from PROCLIB.worldcitycoords
  order by City;
quit;

proc fedsql;
   title 'Climate Zones of World Cities';
   select City, Country, Latitude,
          case
             when Latitude > 67 then 'North Frigid'
             when (67 >= Latitude) and (Latitude >= 23) then 'North Temperate'
             when (23 > Latitude) and (Latitude > -23) then 'Torrid'
             when (-23 >= Latitude) and (Latitude >= -67) then 'South Temperate'
             else 'South Frigid'
          end as "ClimateZone"
       from PROCLIB.worldcitycoords
       order by City limit 12;
quit;
```

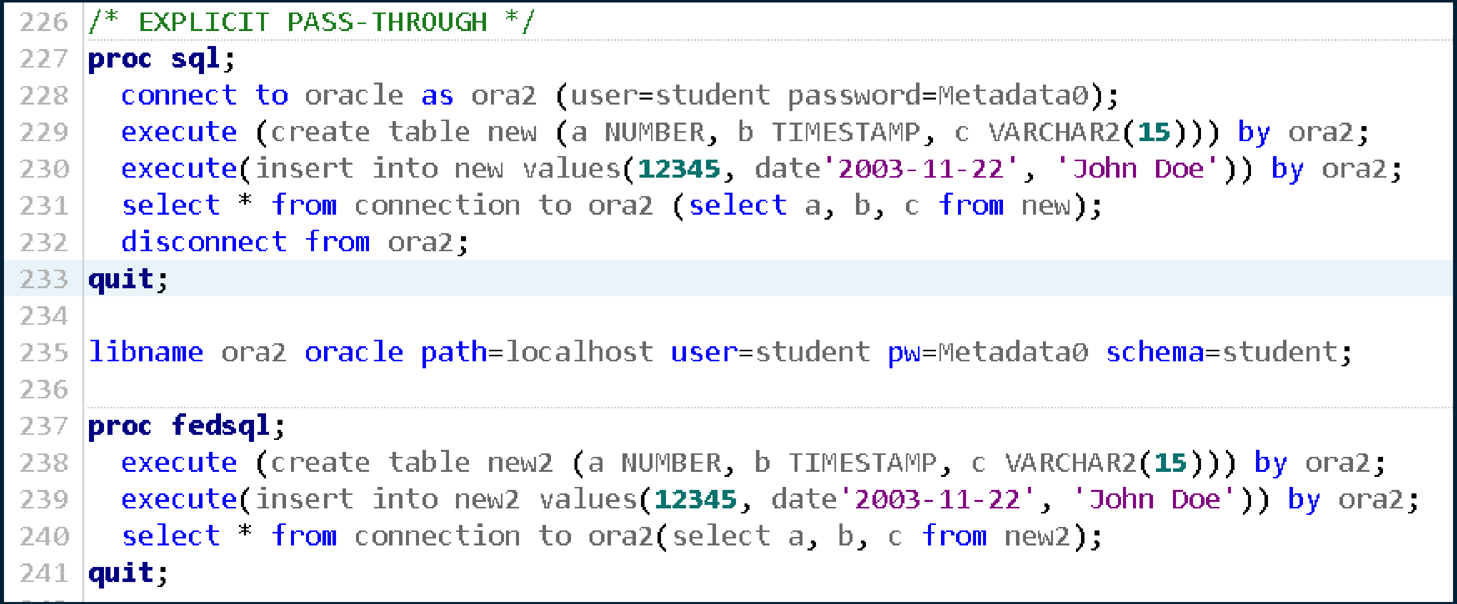


Valid Operators

Operator	Description
+	adds
-	subtracts
*	multiplies
/	divides
=	equals
<>	does not equal
>	is greater than
<	is less than
>=	is greater than or equal to
<=	is less than or equal to
**	raises to a power
unary -	indicates a negative number
||	concatenates

Performing PROC SQL Tasks with FedSQL

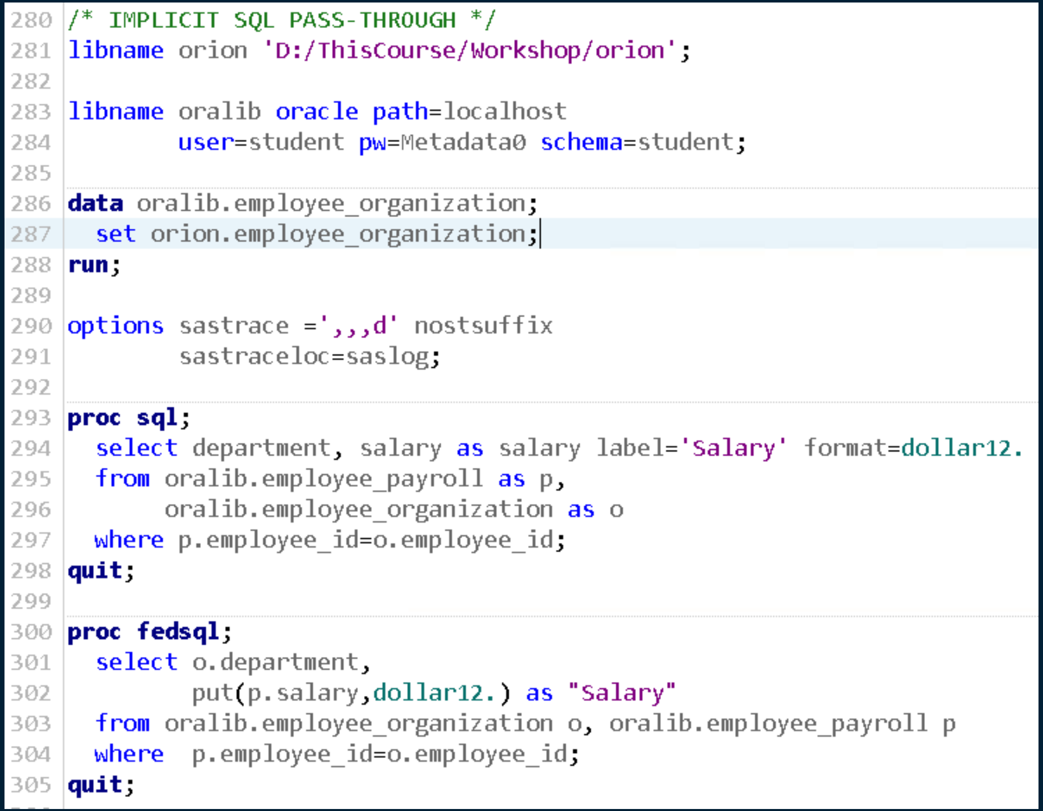
Using Explicit Pass-Through



```
/* EXPLICIT PASS-THROUGH */
proc sql;
  connect to oracle as ora2 (user=student password=Metadata0);
  execute (create table new (a NUMBER, b TIMESTAMP, c VARCHAR2(15))) by ora2;
  execute(insert into new values(12345, date'2003-11-22', 'John Doe')) by ora2;
  select * from connection to ora2 (select a, b, c from new);
  disconnect from ora2;
quit;

libname ora2 oracle path=localhost user=student pw=Metadata0 schema=student;

proc fedsql;
  execute (create table new2 (a NUMBER, b TIMESTAMP, c VARCHAR2(15))) by ora2;
  execute(insert into new2 values(12345, date'2003-11-22', 'John Doe')) by ora2;
  select * from connection to ora2(select a, b, c from new2);
quit;
```

Performing PROC SQL Tasks with FedSQL

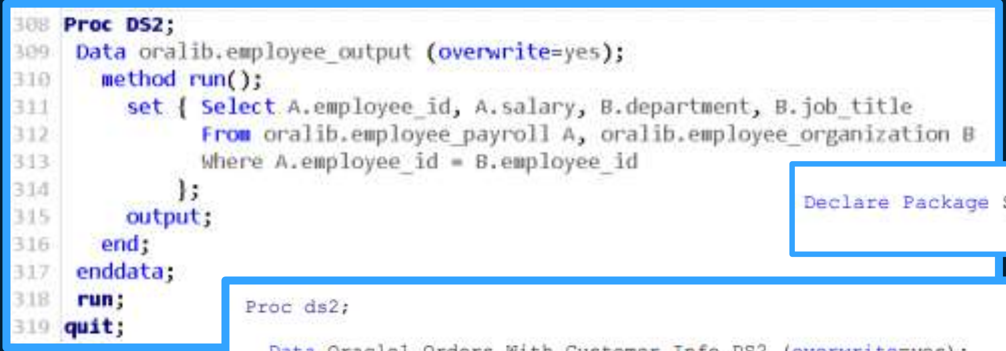
Performing Implicit Pass-Through



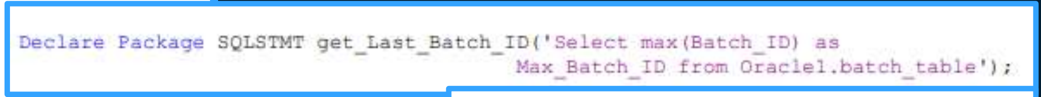
```
/* IMPLICIT SQL PASS-THROUGH */
libname orion 'D:/ThisCourse/Workshop/orion';

libname oralib oracle path=localhost
        user=student pw=Metadata0 schema=student;

data oralib.employee_organization;
  set orion.employee_organization;
run;

options sastrace =',,,d' nostsuffix
        sastraceloc=saslog;

proc sql;
  select department, salary as salary label='Salary' format=dollar12.
  from oralib.employee_payroll as p,
       oralib.employee_organization as o
  where p.employee_id=o.employee_id;
quit;

proc fedsql;
  select o.department,
         put(p.salary,dollar12.) as "Salary"
  from oralib.employee_organization o, oralib.employee_payroll p
  where  p.employee_id=o.employee_id;
quit;
```

Performing PROC SQL Tasks with FedSQL

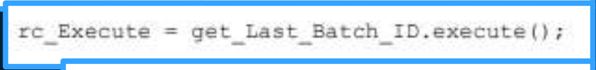
FedSQL and DS2



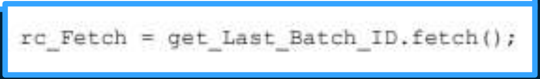
```
Proc DS2;
 Data oralib.employee_output (overwrite=yes);
   method run();
     set { Select A.employee_id, A.salary, B.department, B.job_title
           From oralib.employee_payroll A, oralib.employee_organization B
           Where A.employee_id = B.employee_id
         };
     output;
   end;
 enddata;
 run;
quit;
```



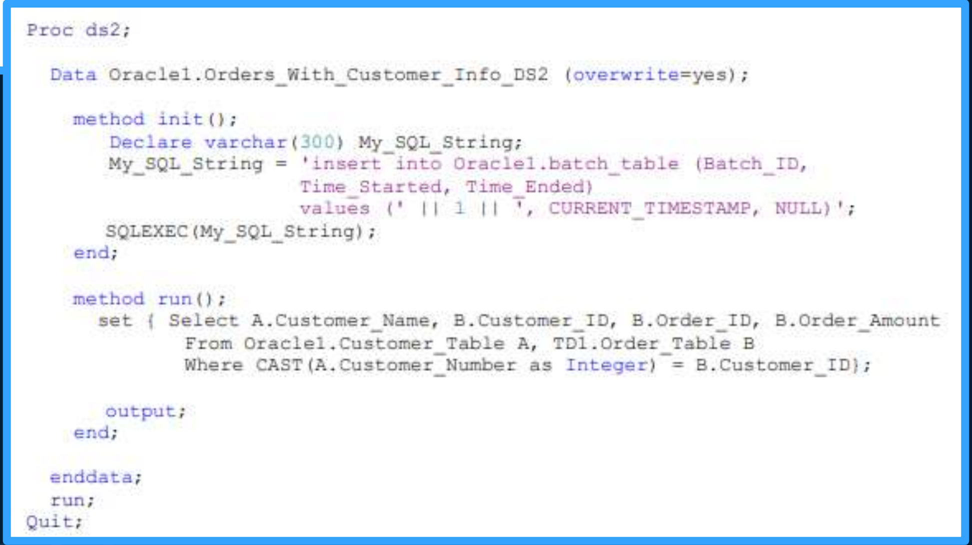
```
Declare Package SQLSTMT get_Last_Batch_ID('Select max(Batch_ID) as
                                 Max_Batch_ID from Oracle1.batch_table');
```



```
rc_Execute = get_Last_Batch_ID.execute();
```



```
rc_Fetch = get_Last_Batch_ID.fetch();
```



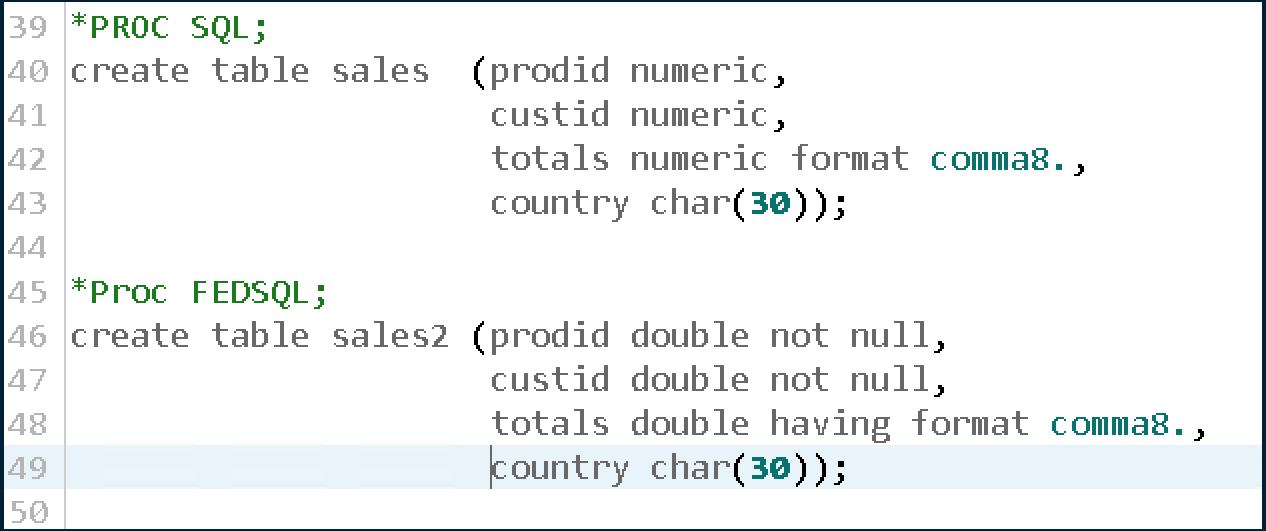
```
Proc ds2;

  Data Oracle1.Orders_With_Customer_Info_DS2 (overwrite=yes);

    method init();
       Declare varchar(300) My_SQL_String;
       My_SQL_String = 'insert into Oracle1.batch_table (Batch_ID,
                        Time_Started, Time_Ended)
                        values (' || 1 || ', CURRENT_TIMESTAMP, NULL)';
       SQLEXEC(My_SQL_String);
    end;

    method run();
      set { Select A.Customer_Name, B.Customer_ID, B.Order_ID, B.Order_Amount
            From Oracle1.Customer_Table A, TD1.Order_Table B
            Where CAST(A.Customer_Number as Integer) = B.Customer_ID};

       output;
    end;

  enddata;
  run;
Quit;
```

PROC SQL versus PROC FEDSQL

- PROC SQL
  - Write queries or execute statements against SAS dataset or in a database
  - Combine functionality of DATA Step & multiple PROC steps in one call
  - Can only handle one database connection per query
  - Is not compliant with ANSI SQL syntax (…)
- PROC FEDSQL
  - Faster performance
  - Ability to connect to multiple databases in one query
  - Increased security
  - Support for new data types
  - Compliance with ANSI SQL: 1999 core standards

Why (and When to) Use FedSQL

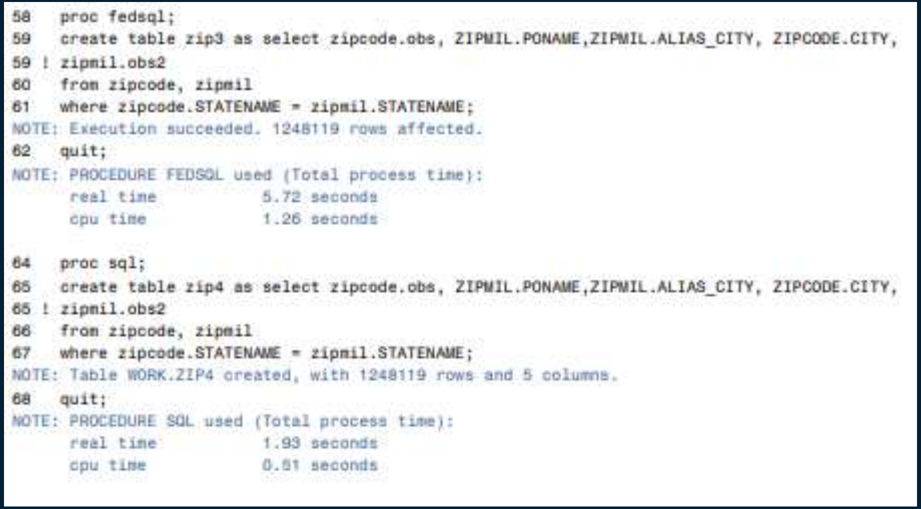
Greater Precision



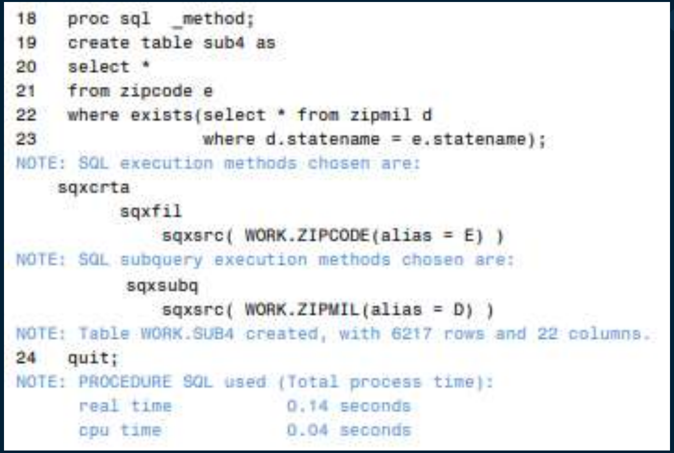
```
*PROC SQL;
create table sales  (prodid numeric,
                     custid numeric,
                     totals numeric format comma8.,
                     country char(30));

*Proc FEDSQL;
create table sales2 (prodid double not null,
                     custid double not null,
                     totals double having format comma8.,
                     country char(30));
```

Why (and When to) Use FedSQL (or NOT)

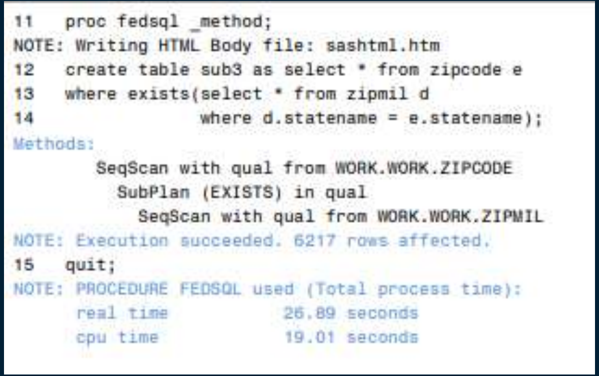
Performance



```
58     proc fedsql;
59     create table zip3 as select zipcode.obs, ZIPMIL.PONAME,ZIPMIL.ALIAS_CITY, ZIPCODE.CITY,
59 !   zipmil.obs2
60     from zipcode, zipmil
61     where zipcode.STATENAME = zipmil.STATENAME;
NOTE: Execution succeeded. 1248119 rows affected.
62     quit;
NOTE: PROCEDURE FEDSQL used (Total process time):
      real time           5.72 seconds
      cpu time            1.26 seconds

64     proc sql;
65     create table zip4 as select zipcode.obs, ZIPMIL.PONAME,ZIPMIL.ALIAS_CITY, ZIPCODE.CITY,
65 !   zipmil.obs2
66     from zipcode, zipmil
67     where zipcode.STATENAME = zipmil.STATENAME;
NOTE: Table WORK.ZIP4 created, with 1248119 rows and 5 columns.
68     quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           1.93 seconds
      cpu time            0.61 seconds
```



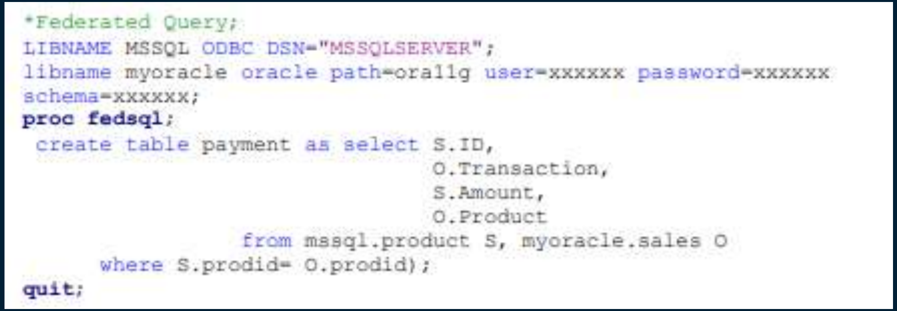
```
18     proc sql _method;
19     create table sub4 as
20     select *
21     from zipcode e
22     where exists(select * from zipmil d
23                  where d.statename = e.statename);
NOTE: SQL execution methods chosen are:
      sqxcrta
          sqxfil
              sqxsrc( WORK.ZIPCODE(alias = E) )
NOTE: SQL subquery execution methods chosen are:
          sqxsubq
              sqxsrc( WORK.ZIPMIL(alias = D) )
NOTE: Table WORK.SUB4 created, with 6217 rows and 22 columns.
24     quit;
NOTE: PROCEDURE SQL used (Total process time):
      real time           0.14 seconds
      cpu time            0.04 seconds
```



```
11     proc fedsql _method;
NOTE: Writing HTML Body file: sashtml.htm
12     create table sub3 as select * from zipcode e
13     where exists(select * from zipmil d
14                  where d.statename = e.statename);
Methods:
        SeqScan with qual from WORK.WORK.ZIPCODE
          SubPlan (EXISTS) in qual
            SeqScan with qual from WORK.WORK.ZIPMIL
NOTE: Execution succeeded. 6217 rows affected.
15     quit;
NOTE: PROCEDURE FEDSQL used (Total process time):
      real time           26.89 seconds
      cpu time            19.01 seconds
```

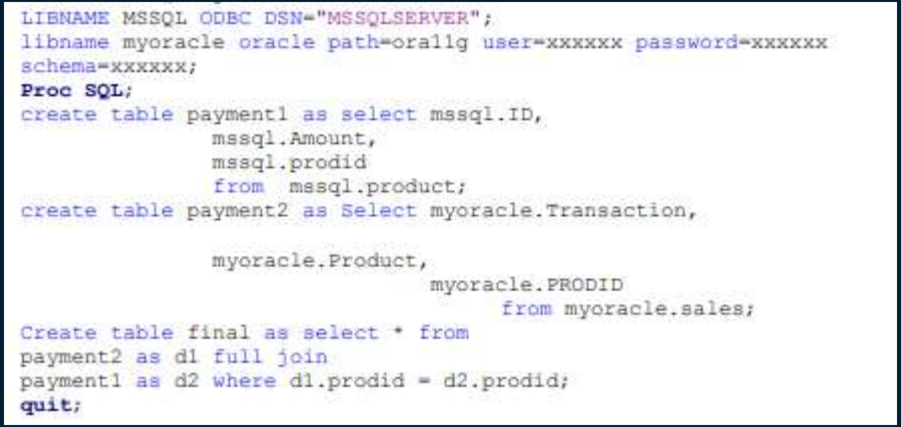
Performing PROC SQL Tasks with FedSQL

Federated Queries



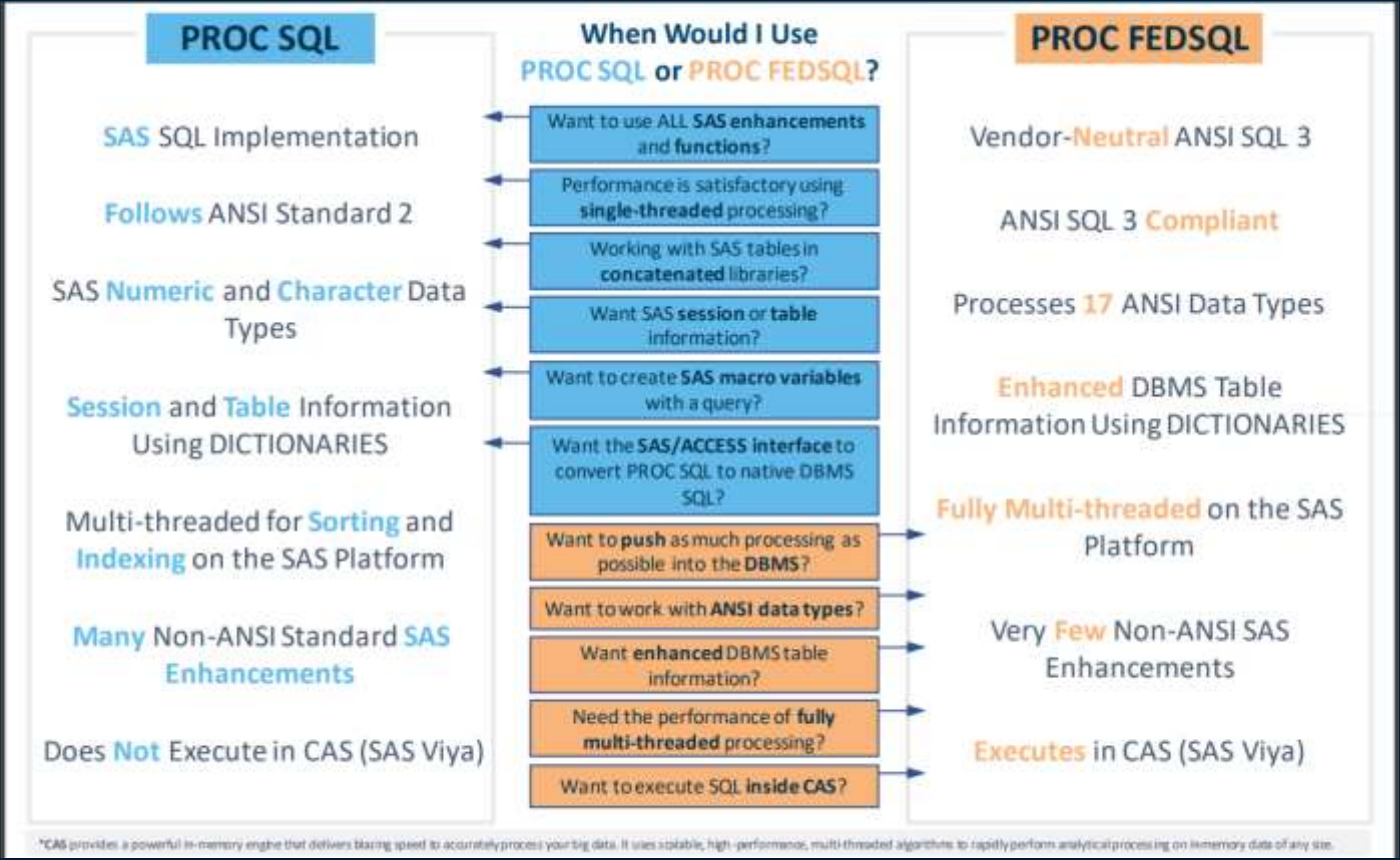
```
*Federated Query;
LIBNAME MSSQL ODBC DSN="MSSQLSERVER";
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx
schema=xxxxxx;
proc fedsql;
 create table payment as select S.ID,
                                O.Transaction,
                                S.Amount,
                                O.Product
                  from mssql.product S, myoracle.sales O
      where S.prodid= O.prodid);
quit;
```

DEMONSTRATION:
SQL_Server, Oracle & DB2



```
LIBNAME MSSQL ODBC DSN="MSSQLSERVER";
libname myoracle oracle path=ora11g user=xxxxxx password=xxxxxx
schema=xxxxxx;
Proc SQL;
create table payment1 as select mssql.ID,
               mssql.Amount,
               mssql.prodid
               from  mssql.product;
create table payment2 as Select myoracle.Transaction,

               myoracle.Product,
                                myoracle.PRODID
                                      from myoracle.sales;
Create table final as select * from
payment2 as d1 full join
payment1 as d2 where d1.prodid = d2.prodid;
quit;
```

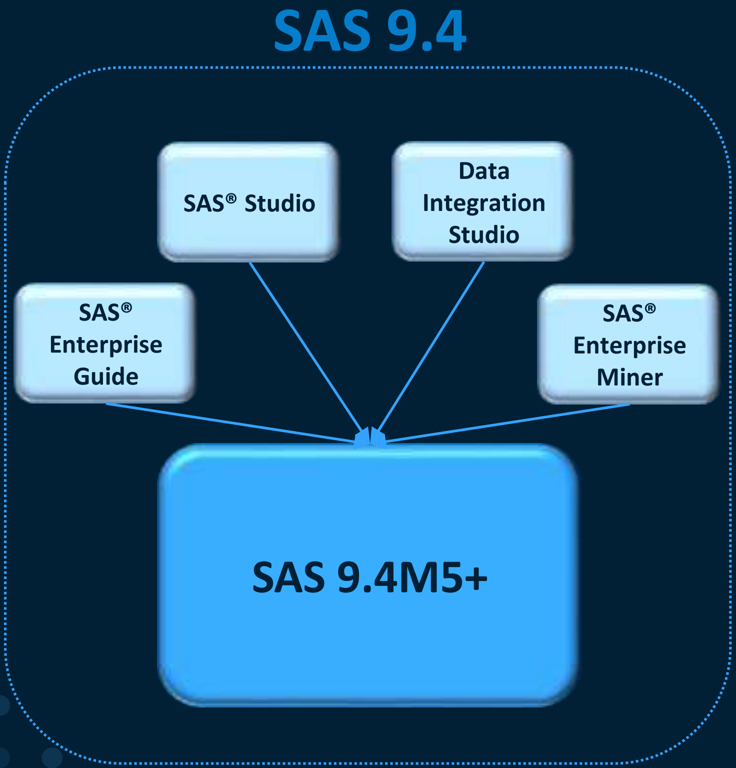
Comparison Table

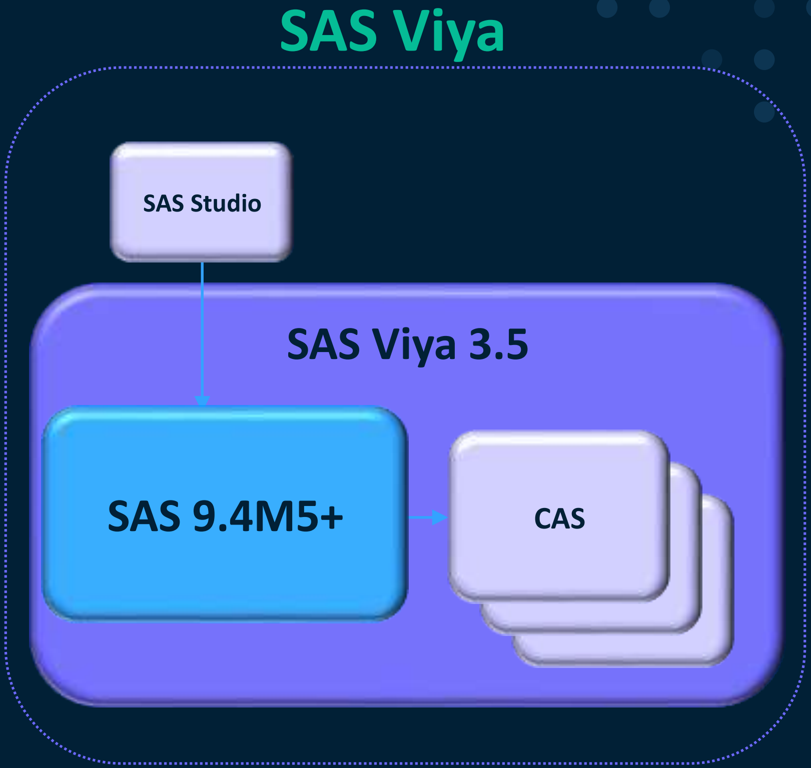


PROC SQL	When Would I Use PROC SQL or PROC FEDSQL?	PROC FEDSQL
SAS SQL Implementation	Want to use ALL **SAS enhancements** and **functions**?	Vendor-Neutral ANSI SQL 3
Follows ANSI Standard 2	Performance is satisfactory using **single-threaded** processing?	ANSI SQL 3 Compliant
SAS Numeric and Character Data Types	Working with SAS tables in **concatenated** libraries?	Processes 17 ANSI Data Types
Session and Table Information Using DICTIONARIES	Want SAS **session** or **table** information?	Enhanced DBMS Table Information Using DICTIONARIES
Multi-threaded for Sorting and Indexing on the SAS Platform	Want to create **SAS macro variables** with a query?	Fully Multi-threaded on the SAS Platform
Many Non-ANSI Standard SAS Enhancements	Want the **SAS/ACCESS interface** to convert PROC SQL to native DBMS SQL?	Very Few Non-ANSI SAS Enhancements
Does Not Execute in CAS (SAS Viya)	Want to **push** as much processing as possible into the **DBMS**?	Executes in CAS (SAS Viya)
	Want to work with **ANSI data types**?	
	Want **enhanced** DBMS table information?	
	Need the performance of **fully multi-threaded** processing?	
	Want to execute SQL **inside CAS**?	

*CAS provides a powerful in-memory engine that delivers blazing speed to accurately process your big data. It uses scalable, high-performance, multi-threaded algorithms to rapidly perform analytical processing on in-memory data of any size.

Running PROC FedSQL in SAS Viya





SAS Viya Procedures

Running FedSQL Programs in CAS

How to Run FedSQL in CAS

- Using the FedSQL Procedure
- Using the fedsql.execDirect action

Support FedSQL Statements

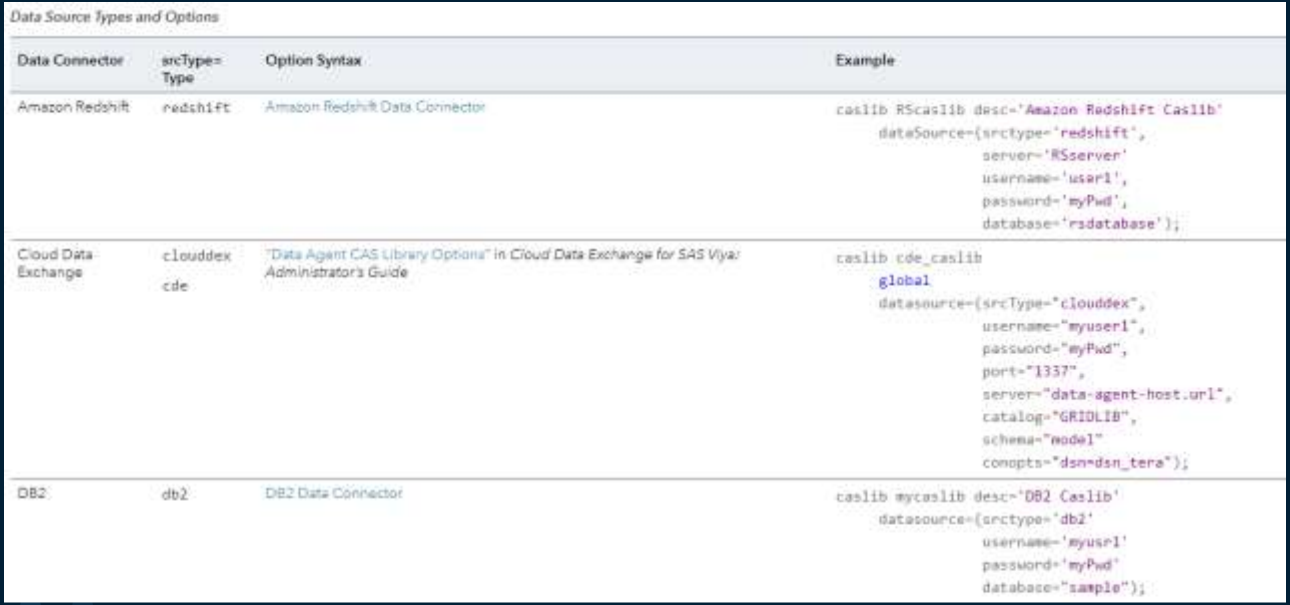
The following FedSQL statements are supported in CAS:

- CREATE TABLE, with the AS query expression
- DROP TABLE
- SELECT

Supported Data Sources

fedSQL.execDirect action uses SAS Data Connectors

Quick Reference for Data Connector Syntax



Data Source Types and Options

Data Connector	srcType= Type	Option Syntax	Example

Amazon Redshift	`redshift`	Amazon Redshift Data Connector	`caslib RScaslib desc='Amazon Redshift Caslib'`
			`dataSource=(srctype='redshift',`
			`server='RSserver'`
			`username='user1',`
			`password='myPwd',`
			`database='rsdatabase');`

Cloud Data Exchange	`clouddex`
	`cde`
		"Data Agent CAS Library Options" in *Cloud Data Exchange for SAS Viya: Administrator's Guide*	`caslib cde_caslib`
			`global`
			`datasource=(srcType="clouddex",`
			`username="myuser1",`
			`password="myPwd",`
			`port="1337",`
			`server="data-agent-host.url",`
			`catalog="GRIDLIB",`
			`schema="model"`
			`conopts="dsn=dsn_tera");`

DB2	`db2`	DB2 Data Connector	`caslib mycaslib desc='DB2 Caslib'`
			`datasource=(srctype='db2'`
			`username='myusr1'`
			`password='myPwd'`
			`database="sample");`

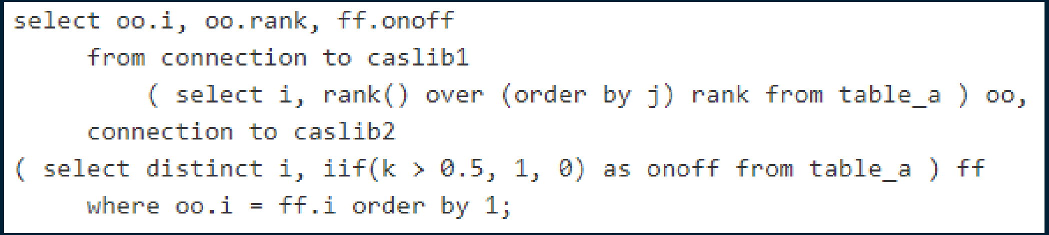
Implicit & Explicit Pass-Through Support in CAS

Implicit Pass-Through

- Single-source, full-query implicit pass-through
- Is on by default
- Performs automatically if the IP conditions are met

Explicit Pass-Through

- Connect to data source
- Send SQL statements directly to that data source
- Use data source specific syntax

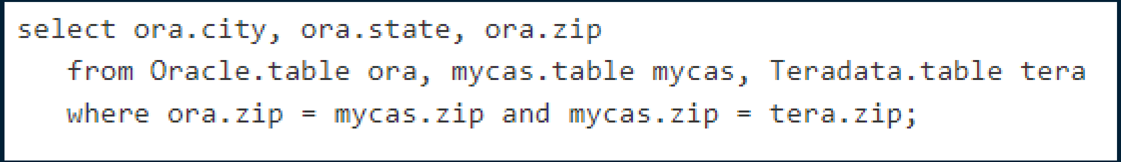


```
select oo.i, oo.rank, ff.onoff
     from connection to caslib1
         ( select i, rank() over (order by j) rank from table_a ) oo,
     connection to caslib2
( select distinct i, iif(k > 0.5, 1, 0) as onoff from table_a ) ff
     where oo.i = ff.i order by 1;
```

FedSQL Federated Queries in CAS

FedSQL request against multiple data sources

- Use two-part table name (caslib.table-name)
- Caslibs must be previously assigned & reference a data connector
- Tables are then loaded into CAS

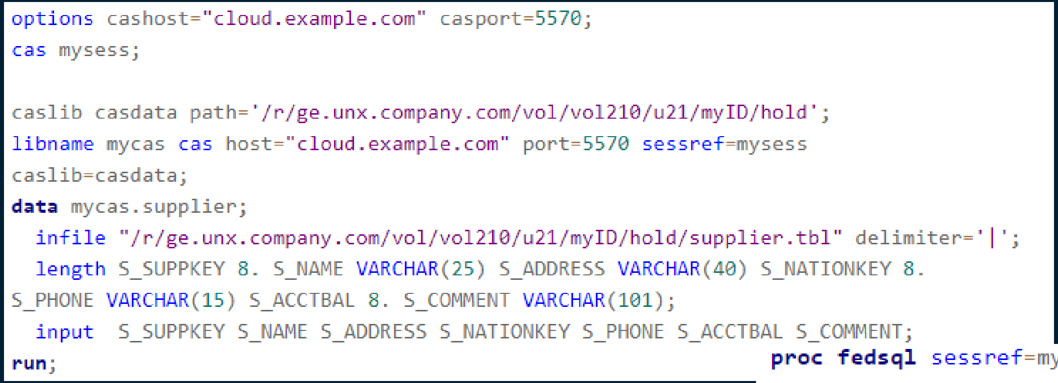


```
select ora.city, ora.state, ora.zip
   from Oracle.table ora, mycas.table mycas, Teradata.table tera
   where ora.zip = mycas.zip and mycas.zip = tera.zip;
```

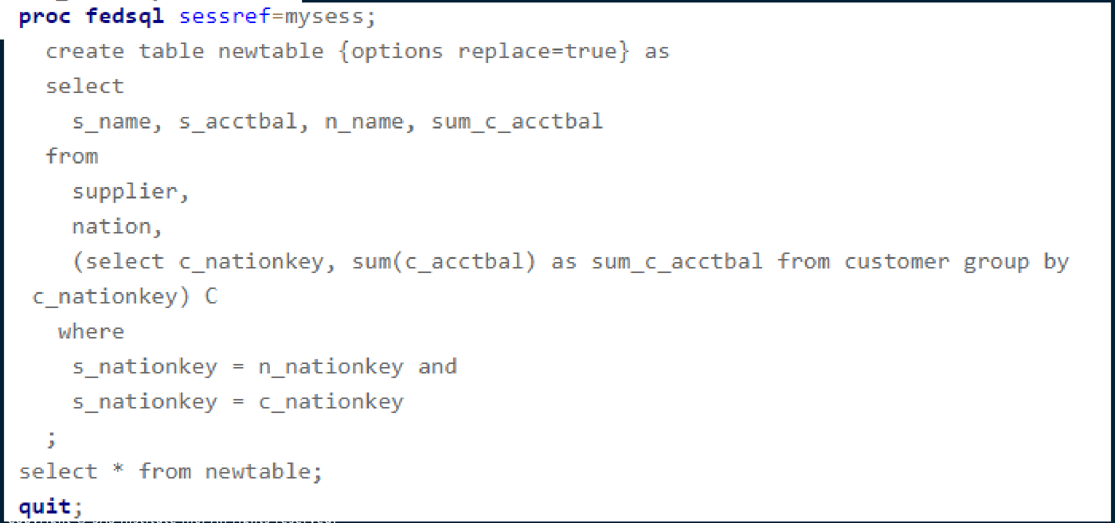
Example 1: Querying a DBMS Table

```
options cashost="cloud.example.com" casport=5570;
cas mysess;

caslib castera desc='Teradata Caslib'
   datasource=(srctype='teradata',
      dataTransferMode='serial',
      username='myname',
      password='mypw',
      server='testserver',
      db='test');

proc fedsql sessref=mysess;
select Pos, count(Pos) as Count_Pos
    from castera.employees
    group by Pos
    having count(Pos) >= 2;

quit;
```

Example 2: Explicitly Loading & Joining Tables in CAS

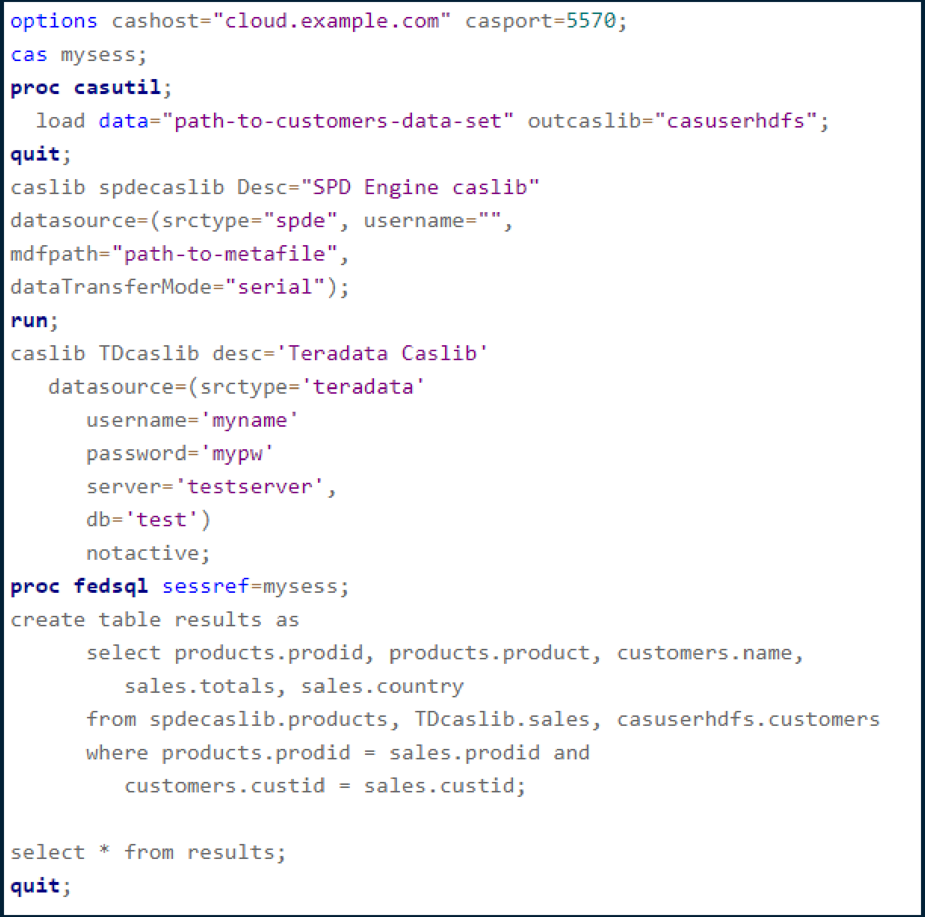


```
options cashost="cloud.example.com" casport=5570;
cas mysess;

caslib casdata path='/r/ge.unx.company.com/vol/vol210/u21/myID/hold';
libname mycas cas host="cloud.example.com" port=5570 sessref=mysess
caslib=casdata;
data mycas.supplier;
  infile "/r/ge.unx.company.com/vol/vol210/u21/myID/hold/supplier.tbl" delimiter='|';
  length S_SUPPKEY 8. S_NAME VARCHAR(25) S_ADDRESS VARCHAR(40) S_NATIONKEY 8.
S_PHONE VARCHAR(15) S_ACCTBAL 8. S_COMMENT VARCHAR(101);
  input  S_SUPPKEY S_NAME S_ADDRESS S_NATIONKEY S_PHONE S_ACCTBAL S_COMMENT;
run;
```



```
proc fedsql sessref=mysess;
  create table newtable {options replace=true} as
  select
    s_name, s_acctbal, n_name, sum_c_acctbal
  from
    supplier,
    nation,
    (select c_nationkey, sum(c_acctbal) as sum_c_acctbal from customer group by
 c_nationkey) C
   where
    s_nationkey = n_nationkey and
    s_nationkey = c_nationkey
  ;
select * from newtable;
quit;
```

Example 3: Joining Tables from Multiple CAS Libraries



```
options cashost="cloud.example.com" casport=5570;
cas mysess;
proc casutil;
  load data="path-to-customers-data-set" outcaslib="casuserhdfs";
quit;
caslib spdecaslib Desc="SPD Engine caslib"
datasource=(srctype="spde", username="",
mdfpath="path-to-metafile",
dataTransferMode="serial");
run;
caslib TDcaslib desc='Teradata Caslib'
   datasource=(srctype='teradata'
      username='myname'
      password='mypw'
      server='testserver',
      db='test')
      notactive;
proc fedsql sessref=mysess;
create table results as
      select products.prodid, products.product, customers.name,
         sales.totals, sales.country
      from spdecaslib.products, TDcaslib.sales, casuserhdfs.customers
      where products.prodid = sales.prodid and
         customers.custid = sales.custid;

select * from results;
quit;
```

DEMO

Conclusion

Use the Right Tool for the Job

Syntax, Performance, Output

- PROC FEDSQL does not do everyday tasks as well as PROC SQL
- FEDSQL is excellent at connecting to multiple different databases at once

Resources for Learning More

Documentation

- Introduction to the FedSQL Language
- How to Perform Common PROC SQL Tasks in FedSQL
- PROC FedSQL and the ANSI Standard

Papers

- High-Performance Data Access with FedSQL and DS2
- Anything You Can Do I Can Do Better: PROC FEDSQL VS PROC SQL
- Working with PROC FEDSQL in SAS® 9.4

Q&A

sas.com

Explore Helpful Resources

Ask the Expert
View other user webinars that provide insights into using SAS products to make your job easier.

FREE Training
Learn from home – free for 30 days. Get software labs to practice and online support if needed.

SAS Support Communities
Ask questions, get answers and share insights with SAS users.

SAS Analytics Explorers
An exclusive platform to collaborate, learn and share your expertise. Gain access to a diverse network to advance your career. Special rewards and recognition exclusively for SAS users.

SAS Users YouTube Channel
A plethora of videos on hundreds of topics, just for SAS users.

Newsletters
Get the latest SAS news plus tips, tricks and more.

Users Groups
Meet local SAS users, network and exchange ideas – virtually.

SAS Profile
If you haven't already done so, create your SAS Profile to access free training, SAS Support Communities, technical support, software downloads, newsletters and more.