Leveraging IBM Netezza Data Warehouse Appliances with SAS

Best Practices Guide for SAS Programmers

11.11v1

Acknowledgements

Thomas Dinsmore, Analytic Solutions Architect, IBM

Pete Heywood, Analytics Systems Engineer, IBM

Paulinda Pangalda, Managing Consultant - Architect, IBM

Dana Rafiee, Destiny Corporation

William Zanine, Solution Executive - Analytic Solutions, IBM

Tracy Zerbin, Advanced Analytics Systems Engineer, IBM

Destiny Corporation is a Business and Technology Consulting Firm that engages in primarily project based Data Warehousing and Analytics initiatives. A dedicated staff works closely with clients to learn their business requirements, analyze the existing infrastructure, and the overall goals of the organization. The most appropriate business process and IT solutions are designed and developed, using industry best practices to meet client needs. For most clients, Destiny works in a data modeling, ETL, systems design, In-Database, Grid architecture, software development, and support role, assisting organizations in determining the appropriate target state architecture, data, analysis and business processes. Destiny provides an objective recommendation to support both the business and IT.

Established in 1987, Destiny's clients are primarily Fortune 500 Companies and Government Organizations. Destiny Corporation delivers a wide variety of solutions that enable its clients to achieve their goals, combining strategy, marketing, industry and sector specific knowledge with experience in advanced technology, software, and hardware solutions.

For more information, visit: http://www.destinycorp.com/Consulting/consulting.html

Table of Contents

Introduction

This document summarizes key best practices for SAS programmers seeking to leverage IBM Netezza data warehouse appliances with SAS. It is intended to supplement SAS documentation and assumes that the user has completed training in SAS/Access Interface to Netezza and in basic SQL. Refer to the “References” section of this document for details on pertinent SAS and Netezza documentation.

Use the SAS/Access Interface to Netezza

SAS/ACCESS interface to Netezza allows SAS programmers to reference database objects directly in a DATA step or SAS procedure using the SAS LIBNAME statement. New libraries can be defined to associate a libref with relational DBMS objects or a SAS data library. SAS reads the data values just as if they were in a SAS data file.

Structured Query Language (SQL) can also be used to update, delete, or insert data into a relational table using the SAS SQL procedure.

The ability to pass database queries, joins and many functions to a target data source for processing reduces network traffic and speeds data access. Faster load times are achieved with support for Netezza native bulk-load utilities.

There is a high degree of control over data security because SAS/ACCESS honors and augments the native security of the target data source. Access to the database can be provided to as many or few users as necessary.

SAS/Access for ODBC is also available, but provides lower performance as it does not ‘push down’ as much of the SAS code or PROC SQL as SQL to Netezza. SAS/Access for ODBC does not support SAS 9.3 in-database processing.

Minimize Data Returned to SAS

As a rule, data handling is faster and more efficient when performed in Netezza. Projects should be organized so the data set returned to SAS consists of the minimum number of rows and columns needed to perform analysis.

In an individual query, minimize the length of the record returned to SAS from Netezza by specifying column names in a SQL SELECT statement (instead of Select *) or use DROP and KEEP. DROP and KEEP must be used as data set options and not DROP and KEEP statements.

SAS applies Data set options on inbound data in the data step during the compilation phase when the Program Data Vector is defined, prior to moving data. SAS applies DROP and KEEP statements in the data step to the outbound data after extracting data and loading it into the Program Data Vector. Less variables in the PDV always make the data step run faster.

Migrate Large SAS Datasets to Netezza

With SAS/Access Interface to Netezza, Netezza can serve as the physical host for SAS datasets. This enables SAS programs to exploit Netezza's high performance architecture, and eliminates network latency from transporting data back to the SAS server.

Use BULK LOAD/UNLOAD to Insert and Retrieve

When working with Netezza, the fastest way to insert and retrieve is through the bulk load and unload facilities. Inserts and retrieves run 30% faster because this facility uses Netezza External Tables.   Specify the DATA set option BULKLOAD=YES or BULKUNLOAD=YES.Use BULKUNLOAD in a libname statement. Use BULKLOAD in a connection string.

Leverage SAS 9.3 In-Database PROCs

With Release 9.3, SAS supports the following PROCs running in-database in Netezza:

- FREQ
- REPORT
- SORT
- RANK
- MEANS/SUMMARY
- TABULATE

SAS converts the SAS syntax for these PROCS into SQL and submits that SQL to Netezza for processing. When used with SAS datasets hosted on Netezza, these PROCs run entirely inside Netezza, with significant performance gains.

Replace SAS Data Step MERGE with PROC SQL Joins

SAS Data Step MERGE operations can normally be replaced by PROC SQL join statements and executed in-database for more efficient processing. Identify all joins in existing PROC SQL and ensure both datasets are located on Netezza during the JOIN. Identify SAS variables used for MERGE statements and use them to convert MERGE sequences into PROC SQL joins executing on Netezza datasets. This will take advantage of Netezza's massively parallel processing (MPP) architecture, resulting in faster throughput.

Pass SQL Joins to Netezza

In the SAS LIBNAME statement, specify DIRECT_SQL=YES to force PROC SQL to pass SQL joins directly to the database for processing.

Leverage the SQL Pass-Through Facility

There are two ways to access a database with SAS; LIBNAME SQL and the Pass-Through SQL Facility. Using a LIBNAME reference allows reference to Netezza databases and tables with the same syntax used to access SAS datasets. This allows SAS tools and any existing SAS code to be quickly redirected to a Netezza data source. This means SAS code that uses DATA Step processing can be executed with a Netezza database.

LIBNAME references can also be used in PROC SQL that is interpreted 'implicitly'. When accessing Netezza data in this manner the SQL used in PROC SQL must be ANSI-standard. SAS/ACCESS for Netezza translates this into Netezza specific SQL syntax and passes it to Netezza for processing. Accessing data in this manner is as efficient as pass-through when certain conditions are met with respect to WHERE clauses. SAS/ACCESS for Netezza substitutes Netezza functions for SAS Functions where possible. If it cannot, it will bring the data back to SAS for processing. Use the system option SASTRACE= to determine if your ANSI SQL is executing completely in-database.

The Pass-Through SQL Facility interacts with the database objects by using SQL syntax supported by Netezza. This syntax allows SAS/ACCESS Interface for Netezza to directly pass the SQL to the database for in-database processing. The Netezza optimizer takes advantage of the table or view's structure to process joins efficiently. Aggregations that use SQL GROUP BY clauses are also implemented.

When optimizing existing SAS code for Netezza, the simplest method to deliver the greatest performance gains is to use the SQL pass-through facility. This is only applicable to SAS code that is doing SQL style data manipulation; this does not apply to statistical PROCs. For example, first dot/last dot processing would be best processed in SAS. There are conditions where it makes sense to process in Netezza and other conditions where it makes sense to process in SAS. The rule of thumb is if it can be replicated in SQL, Netezza is the answer, if it is SAS specific such as SAS Statistical Procedures or Cursor based Data Step style processing, then SAS is the best place for the logic. Note that IBM Netezza Analytics libraries ship with each appliance. For more information on their capabilities and use, see your IBM Netezza representative.

It is best practice to use SQL pass-through when reading or joining Netezza tables. When joining a Netezza table and a SAS table, it may be useful to move the SAS table into a Netezza table and then perform the join however, consider that each coding situation has its own requirements.

Conform Pass-Through SQL to Netezza SQL Standards

Netezza SQL is based on PostgreSQL. Netezza SQL differs slightly from SAS' version of SQL, which is a variant of the ANSI standard. Known differences include:

Do not use double quotes to enclose a literal; use single quotes instead. The double quotes are used to define mixed case identifiers.

Use || as the concatenation character.

The table below shows key Netezza limits:

Limit	Value
Maximum Database Size	Determined by system size
Maximum Table Size	Determined by system size
Maximum Row Size	64kB
Maximum Field Size	64,000
Maximum Rows per Table	Unlimited
Maximum Columns per Table	1600
Maximum Indexes per Table	Indexes are not required or supported

Understand Exceptions to SQL Pass-Through

SAS attempts to pass most SQL join statements to the native DBMS because direct processing can result to significant performance gains. However, the SQL statement passed by SAS may be affected by the option settings specified in SAS. As a result, the SQL statement submitted to Netezza may fail or produce unexpected results.

Join statements may not pass-through correctly under the following circumstances:

The generated SQL syntax is not accepted by Netezza because the syntax for the join is not ANSI-SQL compliant.

The query involves multiple data sources that do not share the same connection characteristics.

The query uses a SAS dataset option, which is not supported.

Specifying member level control or table locks (READ_LOCK_TYPE= LIBNAME Option or UPDATE_LOCK_TYPE= LIBNAME Option), prohibits the statement from successfully passing to the DBMS for direct processing.

To find out how the SQL statement is submitted to Netezza, enable SQL tracing and refer to the SAS log.

Enable SQL Tracing

Specify the SAS System Option SASTRACE=',,,d' and SASTRACELOC=saslog to pipe SQL statements sent to Netezza to the SAS log. This is useful to understand how SAS interprets SQL commands and transmits them to Netezza.

Replace IF-THEN Processing With WHERE Clauses

If the WHERE clause contains a function that SAS cannot translate into a DBMS function, SAS retrieves all rows from the DBMS and then applies the WHERE clause.

References

SAS 9.3 In-Database Products User's Guide
http://support.sas.com/documentation/cdl/en/indbug/64162/HTML/default/viewer.htm#titlepage.htm

SAS/Access Interface to Relational Databases 9.3
http://support.sas.com/documentation/cdl/en/acreldb/63144/HTML/default/viewer.htm#titlepage.htm

SAS/Access Interface to Relational Databases 9.2
http://support.sas.com/documentation/cdl/en/acreldb/63647/HTML/default/viewer.htm#titlepage.htm

PROC SQL (SAS 9.3)
http://support.sas.com/documentation/cdl/en/sqlproc/63043/HTML/default/viewer.htm#n1oihmdy7om5rmn1aorxui3kxizl.htm

PROC SQL (SAS 9.2)
http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000086336.htm

Netezza SQL
http://www.enzeecommunity.com/docs/DOC-1161

Appendix: Code Samples

Create a SAS Dataset from a Netezza table. The following is a SAS script that creates a dataset, pushes it into a Netezza database, and then pulls it back into SAS. It provides a starting point for understanding how to write SAS code to implement some of the methods mentioned in this document.

```
/* assigning connection info to memory variables  */
/* !CHANGE TO YOUR CONNECTION INFORMATION! */

%let NZServer = 'nzserver'; /*  can be ###.###.###.### or DNS Short Name */
%let NZUser = 'nzuser'; /*This is the Netezza account you use to access Netezza
data */
%let NZDatabase = 'nzdatabase'; /* The name of the Netezza database your using */
%let NZPassword = 'nzpassword'; /* nzuser's Netezza password */

/* Netezza Library - Using Parameterized memory variables*/
LIBNAME nzData netezza server = &NZServer user= &NZUser database=&NZDatabase
password=&NZPassword BULKUNLOAD=YES;

/* Local Library */
LIBNAME SASDATA "C:\Users\IBM_ADMIN\Desktop\Desktop\Projects\SAS\SASDATA";

/* Create a SAS Data Set in a local library */
DATA nzData.SASFoo(BULKLOAD=YES DISTRIBUTE_ON=ID);
INPUT ID GENDER $ AGE HEIGHT WEIGHT;
DATALINES;
1 M 23 68 155
2 F . 61 102
3 M 55   70   202
run;


/*Clean up after*/
PROC SQL;
      drop table nzData.NzFoo;
quit;

/*PROC SQL using a LIBNAME statement
    (with NZ SQL Specific clause 'passed' as a parameter, and a SAS DATA Option) */
Proc SQL;
      create table nzData.NzFoo (DBCREATE_TABLE_OPTS='DISTRIBUTE on (AGE)'
BULKLOAD=YES)
            as (select * from SASDATA.SASFoo);
quit;

/*Clean up after*/
Proc SQL;
```

```
      drop table nzData.NzFoo;
quit;

/*DATA Step using a LIBNAME statement and BULKLOAD*/
DATA nzData.NzFoo (BULKLOAD=YES BL_OPTIONS="logdir 'c:\temp'");
      set SASDATA.SASFoo;
RUN;

LIBNAME nzData netezza server = &NZServer user= &NZUser database=&NZDatabase
password=&NZPassword BULKUNLOAD=YES;
/*PROC SQL creating a SAS table in the temp SAS default library (Called WORK)
this ANSI SQL statement uses the 'implicit' method and the BULKUNLOAD option
   assigned in the above LIBNAME statement*/
PROC SQL;
      create table test as
            select * from nzData1.NzFoo;
quit;

/* PROC SQL works on SAS Data Sets just fine!*/
PROC SQL;
      drop table test;
quit;

/*PROC SQL using a LIBNAME statement 'implicit' ANSI SQL creating a table on
    Netezza (with NZ SQL Specific clause 'passed' as a parameter,
        and the dataset option BULKLOAD) */
proc sql;
      create table nzData.test (DBCREATE_TABLE_OPTS='DISTRIBUTE on (ID)'
BULKLOAD=YES)
            as (select * from SASDATA.SASFoo);
quit;

/*Clean up after - SAS Old school style*/
PROC DATASETS library=work;
    delete TEST;
run;

/*PROC SQL using a execute clause (NZ specific SQL as expected!) table is created
on Netezza*/
proc sql stimer;
      connect to netezza (server = &NZServer user= &NZUser database=&NZDatabase
                  password=&NZPassword BULKUNLOAD=YES);
      execute (create table test
            as (select * from NzFoo) DISTRIBUTE on (ID)) by netezza;
quit;

/* Example of using pass-through NZ SQL and bring in the result into SAS */
proc sql;
      connect to netezza as dbcon (server = &NZServer user= &NZUser
database=&NZDatabase
```

```
            password=&NZPassword BULKUNLOAD=YES);
       create table test as
       select * from connection to dbcon
         (
         /*Netezza SQL goes here!*/
       select * from test
       );
quit;

Additional examples below.


/* Here is an example of using a FORMAT Statement */
/* to control the data type created on Netezza    */

/* assigning connection info to memory variables  */
%let NZServer = '172.20.5.122';
%let NZUser = 'sasadmin';
%let NZDatabase = 'sastest';
%let NZPassword = 'nzpass';

/* Netezza Library - Using Parameterized memory variables*/
LIBNAME nzData netezza server = '172.20.5.122' user= sasadmin database=sastest
password=nzpass;


/* assigning connection info to memory variables  */
/* !CHANGE TO YOUR CONNECTION INFORMAITON! */

%let NZServer = 'nzserver'; /*  can be ###.###.###.### or DNS Short Name */
%let NZUser = 'nzuser'; /*This is the Netezza account you use to access Netezza
data */
%let NZDatabase = 'nzdatabase'; /* The name of the Netezza database your using */
%let NZPassword = 'nzpassword'; /* nzuser's Netezza password */

/* Netezza Library - Using Parameterized memory variables*/
LIBNAME nzData netezza server = &NZServer user= &NZUser database=&NZDatabase
password=&NZPassword BULKUNLOAD=YES;

/* Local Library */
LIBNAME SASDATA "C:\Users\IBM_ADMIN\Desktop\Desktop\Projects\SAS\SASDATA";

/* Create a SAS Data Set in a local library */
DATA SASDATA.SASFoo;
INPUT ID GENDER $ AGE HEIGHT WEIGHT;
DATALINES;
1 M 23 68 155
2 F . 61 102
3 M 55   70   202
;
```

```
/*DATA Step using a LIBNAME statement to create a table on NZ from the local SAS
Data Set*/
DATA nzData.NzFoo;
      set SASDATA.SASFoo;
RUN;

/*Clean up after*/
PROC SQL;
      drop table nzData.NzFoo;
quit;

/*PROC SQL using a LIBNAME statement
    (with NZ SQL Specific clause 'passed' as a parameter, and a SAS DATA Option) */
PROC SQL;
      create table nzData.NzFoo (DBCREATE_TABLE_OPTS='DISTRIBUTE on (AGE)'
BULKLOAD=YES)
            as (select * from SASDATA.SASFoo);
quit;

/*Clean up after*/
PROC SQL;
      drop table nzData.NzFoo;
quit;

/*DATA Step using a LIBNAME statement and BULKLOAD*/
DATA nzData.NzFoo (BULKLOAD=YES BL_OPTIONS="logdir 'c:\temp'");
      set SASDATA.SASFoo;
RUN;

LIBNAME nzData netezza server = &NZServer user= &NZUser database=&NZDatabase
password=&NZPassword BULKUNLOAD=YES;
/*PROC SQL creating a SAS table in the temp SAS default library (Called WORK)
this ANSI SQL statement uses the 'implicit' method and the BULKUNLOAD option
   assigned in the above LIBNAME statement*/
PROC SQL;
      create table test as
            select * from nzData1.NzFoo;
quit;

/* PROC SQL works on SAS Data Sets just fine!*/
PROC SQL;
      drop table test;
quit;

/*PROC SQL using a LIBNAME statement 'implicit' ANSI SQL creating a table on
    Netezza (with NZ SQL Specific clause 'passed' as a parameter,
        and the dataset option BULKLOAD) */
proc sql;
```

```
      create table nzData.test (DBCREATE_TABLE_OPTS='DISTRIBUTE on (ID)'
BULKLOAD=YES)
            as (select * from SASDATA.SASFoo);
quit;

/*Clean up after - SAS Old school style*/
PROC DATASETS library=work;
    delete TEST;
run;

/*PROC SQL using a execute clause (NZ specific SQL as expected!) table is created
on Netezza*/
proc sql stimer;
      connect to netezza (server = &NZServer user= &NZUser database=&NZDatabase
                  password=&NZPassword BULKUNLOAD=YES);
      execute (create table test
            as (select * from NzFoo) DISTRIBUTE on (ID)) by netezza;
quit;

/* Example of using pass-through NZ SQL and bringin the result into SAS */
proc sql;
      connect to netezza as dbcon (server = &NZServer user= &NZUser
database=&NZDatabase
            password=&NZPassword BULKUNLOAD=YES);
     create table test as
     select * from connection to dbcon
       (
       /*Netezza SQL goes here!*/
     select * from test
     );
quit;

/* Some log tracking options */

OPTIONS DEBUG=dbms_timers sastrace=',,t,s' sastraceloc=saslog  no$stsuffix
FULLSTIMER STIMEFMT=S;
OPTIONS NOTES STIMER SASTRACE=',,,db' sastraceloc=saslog  nostsuffix;
OPTIONS STIMER sastrace=',,,db' sastraceloc=saslog nostsuffix; /* My Favorite! */

/* Create a SAS Data Set in a local library with a date in it*/
DATA SASDATA.SASFoo;
INPUT ID GENDER $ AGE HEIGHT WEIGHT HIRED anydtdte8.; /* Date format, creates a
number */
DATALINES;
1 M 23 68 155 11-01-06
2 F . 61 102 10-24-09
3 M 55   70   202 6-24-09
;

PROC SQL; drop table nzData.NzFoo; quit;
```

```
/*Pushing data using SAS Formats and implicit SQL */
/*Age and Hired are not formatted so they are both created as DOUBLE */
PROC SQL;
      create table nzData.NzFoo (BULKLOAD=YES)
            as (select * from SASDATA.SASFoo);
quit;

/* If you use ANSI SQL you can use an in-line FORMAT Statements and SAS Functions
*/
/* This implicit SQL applies the SAS Formats and 'converts' the pulled data */
PROC SQL;
      create table SASDATA.test as
            select ID, GENDER, AGE format 3., HEIGHT, WEIGHT, HIRED format
mmddyy10.
            from nzData.nzFoo;
quit;

OPTIONS DATESTYLE=mdy;
DATA SASDATA.SASFoo;
INPUT ID GENDER $ AGE HEIGHT WEIGHT HIRED anydtdte8.;
FORMAT AGE 3. HIRED mmddyy10.;/* Format statement is added to change the output
appearence in SAS */
DATALINES;
1 M 23 68 155 11-01-06
2 F . 61 102 10-24-09
3 M 55   70   202 6-24-09
;

PROC SQL; drop table nzData.NzFoo; quit;

/*Pushing data using SAS Formats and implicit SQL */
/* After Formatting Hired now is a DATE in Netezza, and AGE is a SMALLINT */
PROC SQL;
      create table nzData.NzFoo (BULKLOAD=YES)
            as (select * from SASDATA.SASFoo);
quit;


/* Since the data is in a Netezza data type the fields do not need
    to be formatted on retrieval using implicit SQL*/
PROC SQL;
      create table SASDATA.test as
            select * from nzData.nzFoo;
quit;

/* Since the data is in a Netezza data type the fields do not need
    to be formatted on retrieval even when using explicit SQL*/
/* Using Explicit SQL you can use Netezza SQL cast statements to change
the data types rendered in SAS within the passthrough SQL */
```

```
proc sql;
      connect to netezza as dbcon (server = &NZServer user= &NZUser
database=&NZDatabase
            password=&NZPassword BULKUNLOAD=YES);
     create table SASDATA.test as
     select * from connection to dbcon
      (select * from nzFoo
     );
quit;

/* What follows are some examples of formats using date handling in SAS and Netezza
*/

/*Clean up after*/ PROC SQL; drop table nzData.test; quit;
proc sql;
      connect to netezza (server = &NZServer user= &NZUser database=&NZDatabase
                  password=&NZPassword BULKUNLOAD=YES);
      execute (
            create table test as
                  select now() as datetime_value,
                  current_date as date_value,
                  'nzData1' as test) netezza;
quit;
proc sql;
      connect to netezza (server = &NZServer user= &NZUser database=&NZDatabase
                  password=&NZPassword BULKUNLOAD=YES);
      execute (
      insert into test values (now()-1, current_date-1, 'nzData2')
                  ) netezza;
quit;

/* If you use ANSI SQL you can use an in-line FORMAT Statement and SAS Functions
            The NZ data is projected/restricted (only selected fields/records are
                  pulled accross)and SAS applies the functions in SAS */
PROC SQL;
      create table SASDATA.test as
            select datetime_value,
               datepart(datetime_value) as trunc_date format mmddyy10.,
               date_value format mmddyy10. ,
                     'SASData' as test
            from nzData.test
            where date_value < today();
;
quit;

/* This example is NZ pass-through emulating the SAS date format MMDDYY10., in this
case
      the return value is a string. SAS has no way of knowing it's a date. */
proc sql;
```

```
      connect to netezza as dbcon (server = &NZServer user= &NZUser
database=&NZDatabase
            password=&NZPassword BULKUNLOAD=YES);
     create table SASDATA.test as
     select * from connection to dbcon
      (select extract(MONTH from datetime_value)||'/'||
           extract(DAY from datetime_value)||'/'||
           extract(YEAR from datetime_value) as field1 from test
     );
quit;

proc sql;
      connect to netezza as dbcon (server = &NZServer user= &NZUser
database=&NZDatabase
            password=&NZPassword BULKUNLOAD=YES);
     create table test as
     select * from connection to dbcon
      (select * from test
     );
quit;

/*=====================*/
/* USER DEFINED FORMATS */
/*=====================*/

/* Local Library */
LIBNAME SASDATA "C:\Users\IBM_ADMIN\Desktop\Desktop\Projects\SAS\SASDATA";

/* Create a SAS Data Set in a local library */
DATA SASDATA.zipsales;
INPUT ID ZIPCODE $ SALES;
DATALINES;
1 02129 11
2 03755 29
3 10005 38
4 27513 42
5 27511 57
6 27705 63
7 92173 76
8 97214 85
9 94105 90
;

DATA nzData.zipsales;
      set SASDATA.zipsales;
RUN;

PROC FORMAT format library = SASDATA;
   value $region
   '02129', '03755', '10005' = 'Northeast'
   '27513', '27511', '27705' = 'Southeast'
   '92173', '97214', '94105' = 'Pacific';
run;

PROC FORMAT;
   value $region
   '02129', '03755', '10005' = 'Northeast'
   '27513', '27511', '27705' = 'Southeast'
   '92173', '97214', '94105' = 'Pacific';
run;

/* assigning connection info to memory variables  */
%let NZServer = '172.20.5.122';
%let NZUser = 'sasadmin';
%let NZDatabase = 'saslib';
%let NZPassword = 'nzpass';

%let indconn=%str(server=&NZserver user=&NZUser database=&NZDatabase
password=&NZpassword);
%indnzpj;
%INDNZ_PUBLISH_JAZLIB(ACTION=CREATE DATABASE=SASTEST OUTDIR=c:\temp);
%indnzpc;
%INDNZ_PUBLISH_COMPILEUDF(DATABASE=SASTEST);
%INDNZ_PUBLISH_FORMATS (fmtcat=SASDATA.formats );

PROC SQL;
select put(zipcode,$region.) as region,
   sum(sales) as sum_sales from nzData.zipsales
   group by region;
quit;

PROC SQL;
      connect to netezza as nzcon (server = &NZServer user= &NZUser
database=&NZDatabase
                  password=&NZPassword);
      execute(
select sas_put("ZIPCODE",'$REGION9.0') as region,
       sum(sales) as sum_sales
       from nzfoo group by region) by nzcon;
quit;
```

For more information

http://www.ibm.com/software/data/netezza/

http://www.netezza.com