GENERALIZED METHOD FOR ANALYSIS OF PAVEMENT MANAGEMENT DATABASE

Final Repot

Submitted to:

Florida Department of Transportation
Pavement Management Office
605 Suwannee Street, Mail Station #70
Tallahassee, Florida

Min-Tang Li, Ph.D.
Research Associate

Lee-Fang Chow
Research Associate

and

Irtishad U. Ahmad, Ph.D., P.E.
Associate Professor

Department of Civil and Environmental Engineering
Florida International University
10555 West Flagler Street
Miami, Florida 33174
Tel: (305) 348-3045
E-mail: ahmadi@fiu.edu

Under
Contract No. BB883

May 2002

DISCLAMER AND ACKNOWLEDGEMENTS

The product of this study is a mainframe application with a graphical user interface that can be utilized as a tool to perform initial statistical analyses on the FDOT's pavement management system database. The program is not designed to perform sophisticated statistical analyses but be flexible enough to accommodate future enhancements to improve its capabilities.

The authors would like to thank Mr. Bruce Dietrich, Mr. Philip Pitts, and Mr. Winfred Langley at the Pavement Management Office of the FDOT for their input and help during the development of the SAS/AF application. The authors also thank Ms. Kathy Shirley at the Computer Services department for the assistance she has provided during the process of program development.

TABLE OF CONTENTS

LIST OF TABLES iii
LIST OF FIGURES iv
EXECUTIVE SUMMARY v
1. INTRODUCTION 1
2. CQR DATABASE 2
3. SAS/AF 5
3.1 Introduction 5
3.1 Program Export 5
3.2 Program Transfer 6
3.3 Program Restore 7
3.4 Limitations 8
4. PROGRAM INTERFACE 10
4.1 Execute SAS/AF Applications 10
4.2 Open Data File 10
4.3 Select Target Variable and Material Type 11
4.4 Select Class Variables and Values 13
5. SUMMARY AND FUTURE POSSIBLE ENHANCEMENTS 16
REFERENCES 17
APPENDIX PROGRAM SOURCE CODE 18

LIST OF TABLES

Table 1. Examples of Pavement Sampling Data by Material Number 3
Table 2. Names and Descriptions for Variables in CQR Database 4

LIST OF FIGURES

Figure 1. Create/Open a Section in EXTRA! 7
Figure 2. File Transfer Using EXTRA 8
Figure 3. File Menu 11
Figure 4. Select Target Variable 12
Figure 5. Define Material ID 12
Figure 6. Program Message Box 13
Figure 7. Example of Data Query 14
Figure 8. Example Output Screen 15

EXECUTIVE SUMMARY

In Florida, pavement materials are periodically and systematically sampled and tested. The test results are recorded in the databases of pavement management system (PMS). The PMS databases are large and are currently stored on a remote mainframe system that is usually not directly accessible by an end user at a PC terminal with a modem connection. This project involves the development of a mainframe computer tool to allow the remote users to conveniently and instantly perform initial on-line statistical analyses on the FDOT's PMS databases.

The SAS/AF software was used to create user-friendly interactive windowing applications. As a product of the Statistical Analysis System (SAS) software, SAS/AF provides interactive user interfaces to all the data access, management, analysis, and presentation features. The software uses the Frame class as the foundation for GUI-based applications. The Frame class provides windowing capabilities to SAS/AF applications. It can contain visual controls and non-visual components for creating a user interface. Through the Frame class, windows and dialog boxes, menu bars or banners, and application help can be created. However, SAS/AF versions 7 and up do not support build-time development of FRAME entries in the mainframe operating environments. Consequently, applications developed in other desktop environments must be ported to a mainframe, i.e., frames have to be created on a personal computer (PC) or other systems that support frame entries and then ported to the mainframe.

The mainframe SAS/AF application program was designed to query a subset of records for analysis. Currently, querying non-index variables is likely to consume more CPU time than a data source is allowed. As a result, users can only use index variables to query and create subset databases of less than 86,030 records. This report documents the development process and serves as the user manual for the SAS/AF mainframe program developed. The procedure for porting the SAS/AF code from a PC to the mainframe is also included.

1. INTRODUCTION

The evaluation and prediction of pavement performance is crucial to roadway's maintenance and rehabilitation activities since roadway pavement consumes intensive construction capital and thus has significant impacts on the overall planning and budgeting activities. Consequently, pavement materials are periodically and systematically sampled and tested, and the results are recorded in the database of pavement management systems (PMS). In the State of Florida, pavement performance information is available from pavement condition surveys back to 1976 [PING]. The Florida Department of Transportation (FDOT) applies the provisions in the Sampling, Testing, and Reporting Guide (STRG), or known as the Sampling Guide, to perform the necessary tests on pavement materials to assure that both materials and construction meet the minimum requirements of acceptance. The STRG states in detail what must be done and by whom in order for the FDOT to accept materials incorporated into construction work and for FDOT to accept the results of construction operations controlled by sampling and testing [MM].

The FDOT is currently utilizing the Construction Quality Reporting (CQR) System as the computerized PMS to collect, report, and store test results for pavement materials used on FDOT projects [STRG]. The CQR system is operated on the FDOT's mainframe with the IBM Multiple Virtual Systems (MVS) platform. To conveniently and instantly perform initial statistic analyses on the test results for selective sampled pavement materials is critical since the PMS data consist of a sizable amount of detail and are stored on the remote mainframe system that is usually not directly accessible by an end user at a PC terminal with a modem connection.

The Statistical Analysis System (SAS) software is an integrated system of software products that can be utilized to perform the following tasks [SASDOC]:

- Data entry, retrieval, and management;
- Report writing and graphics;
- Statistical and mathematical analysis;
- Business planning, forecasting, and decision support;
- Operations research and project management;
- Quality improvement; and
- Applications development.

The SAS/AF software, one of the SAS products, provides a set of tools for developing applications within the SAS software that can provide interactive user interfaces to all the data access, management, analysis, and presentation features. The objective of this study is thus to develop a SAS/AF computer tool to perform initial on-line statistical analyses on the FDOT's PMS database that is stored on their mainframe system. This report documents the development process and also serves as the user manual for the SAS/AF mainframe program. This report is organized as follows. A brief review of the PMS variables that are generally contained in a CQR database is presented in Section Two. Section Three addresses some of the key features in the SAS/AF software and how to transport SAS components between different computer platforms. Section Four describes an example for program implementation. Finally, Section Five addresses the disclamation and acknowledgement.

2. CQR DATABASE

The CQR database contains the essential information and test results for various types of pavement material that are sampled from the field. Table 1 respectively shows the material description, test name, test ID, sample frequency and size, and test report form for four common material types. The variables that consist of a regular CQR database are illustrated in Table 2.

Due to the significant number of records that are sequentially stored in the pavement management database, variables in various datasets are indexed by FDOT in order to reduce the processing time for querying information and performing analysis. For example, variables such as MANDIST, WPITEM, WPITMSEG, WPPHAZE, FINPRJSQ, MATERLID, SAMPLEID, REPORTNO, COFRMSEC, and CQTSTQUA are usually indexed (refer to Table 2 for variable descriptions).

However, since each dataset stored on the FDOT's mainframe system is specified with a specific access time constraint, e.g., 6 seconds of CPU time, querying a large dataset which takes more CPU resources than it is granted will halt and terminate the program execution instantly (Error -905: Exceeding the number of CPU seconds permitted for each SQL statement). As a result, only the index variables are employed as the parameters in this study to select the subset of records for statistical analyses (refer to Section 4.4 for more information related to the index variables).

Table 1. Examples of Pavement Sampling Data by Material Number

Material No.[1]	Material Description[2]	Sample-Test Name[3]	Test[4]	Sample Frequency[5]		Sample Size[6]		Report No.[7]	
				English	Metric	English	Metric	English	Metric
120A	Bit Mix-Type S	Extraction	F5-544	1/1000 Tons	1/900 MT[9]	1 Box	1 Box	28402	28451
120G	Bit Mix-Type S	Extraction/Marshall	F5-511	1/Lot	1/Lot	N/A	N/A	28501	28550
123D	Bit Mix-Type SP	Core Density	F1-T166	1/1000 Ft/Lift	1/300 M1[8]/Lift	6" Core	15 cm Core	29101	29150
123G	Bit Mix-Type SP	Extr/Vol. Props.	5-334	1/Lot	1/Lot	Varies	Varies	28701	28751

1. An abbreviation for material number. This is an arbitrary number assigned to each material name as part of an identifying process when compiling the list of materials. An alpha-character appearing as part of a material number, however, denotes the number of samples or actions required. Each different letter indicates a separate action or sample. The appearance of the same letter four times (A1, A2, A3, A4) means four separate tests are performed on one sample - not that four individual samples are obtained.
2. A short description of the material requiring sampling and testing.
3. A description of the test required.
4. This is an alphanumeric field that identifies Department tests or specification requirements. Prefixes used have the following meanings:
   - 1 - AASHTO procedure
   - 2 - Federal Standards (procedures or requirements)
   - 3 - ASTM procedure
   - 4 - American Welding Society
   - 5 - FDOT specification
   - 7 - American Wood Preservers Association
   - 8 - The Asphalt Institute
   - F - Florida Method
5. This is a minimum sampling/testing frequency established to verify specification compliance for most state and all federal aid projects.
6. The minimum required sample size to perform the indicated test(s).
7. The number of the test report form or the method of documenting results of tests/inspection. Usually refers to the CQR (Construction quality Reporting) form number used to report test results using the CQRTS07A screen.
8. CQR format M1 is equivalent to meter in length.
9. CQR format MT is equivalent to metric ton in mass.

Table 2. Names and Descriptions for Variables in CQR Database

Variable	Description	Length	Format
EXTRACDT	System date that the extract file program was run	8	Date7
REPORTNO[3]	Test report form number	5	$5[1]
CQFRMSEC[3]	Section of form	1	$1
CQFRMSEQ[3]	Test results sequence on form	2	$2
CQTESTYP	Type of test/measurement	2	$2
CQTESTNM	Name of test/measurement	30	$30
CQRSLTMS	Test result unit of measure	3	$3
CQTSTMIN	Minimum allowable test result	8	12.4[2]
CQTSTMAX	Maximum allowable test result	8	12.4
CQTSTSTA	Test results status on form	1	$1
CQTSTCST	Standard cost for making test	8	7.2
SPECYEAR	Const. specification book year	2	$2
CQUPDFLG	Test results update flag	1	$1
MANDIST[3]	Managing district	2	$2
WPITEM[3]	Work program item - 1994	6	$6
WPITMSEG[3]	Work program item segment-1994	1	$1
WPPHAZE[3]	Work program phase - 1994	2	$2
FINPRJSQ[3]	Financial project sequence-1994	2	$2
MATERLID[3]	Material identifier	4	$4
SAMPLEID[3]	Sample identification	6	$6
SAMPLEDT	Date sample taken	6	$6
STATNFRM	Station no. Sample starts at	10	$10
STATNTO	Station no. Sample ends at	10	$10
LOCATION	Location sample taken from	10	$10
CQSOURCE	Source of material sample	2	$2
CQPLANT	Plant number sampling material	5	$5
SAMVEND	Samas vendor number	14	$14
CQMTLQTY	Quantity of material sampled	8	13.2
MEASCODE	Unit of measure code	2	$2
CQINUSE	Intended use of material	20	$20
PASSFAIL	Sample status	2	$2
CQMEMO	Notification memo flag	1	$1
CQMATLTS	Material test sequence number	1	$1
RDWYSIDE	Side of roadway or composite	1	$1
CQOFFSFT	Sample offset	4	$4
CQOFFSDR	Sample offset direction	1	$1
CQMAINFL	Mainline flag	1	$1
BASLNREF	Base line reference	20	$20
CQTSTQUA[3]	Test qualifier	4	$4
CQRSLTN	Result of test - numeric	8	12.4

1. Five-character alphabetic text
2. Twelve digit numeric value with 4 decimal
3. Index variable

3. SAS/AF

3.1 Introduction

SAS/AF software can be used to create user-friendly interactive windowing applications. The software uses the Frame class as the foundation for GUI-based applications. The Frame class provides windowing capabilities to SAS/AF applications and serves as a container where visual controls and non-visual components can be added to create a user interface. By utilizing the Frame class, windows and dialog boxes, menu bars or banners, and application help can be created. Although SAS/AF software is required for building applications, users do not need to license or install SAS/AF software in order to run the applications. The commands and procedure that launch SAS/AF applications are available in base SAS software.

The FDOT currently has SAS version 8.1 installed on their MVS mainframe system. However, since versions 7 and up of SAS/AF do not support build-time development of FRAME entries in the mainframe operating environments, applications developed in other desktop environments must be ported to a mainframe. In other words, frames have to be created on a personal computer (PC) or other system that supports frame entries and then ported to the mainframe. The code developed for the program is illustrated in the Appendix section. The following section depicts the file transform between the FDOT mainframe and a PC in the process of program development.

3.1 Program Export

The process of moving a file between incompatible hosts is known as transporting; for example, from CMS to Windows. Such a process is necessary in order to account for binary incompatibilities between different host architectures. Since directly developing a SAS/AF application on the FDOT mainframe system is currently not feasible, the SAS/AF application was initially developed on a PC with the Microsoft Windows 98 operating system and then uploaded to the mainframe system after completion, i.e., the program is transported from the PC (source host) to the mainframe (target host). The process of the program transport involves exporting the program from PC, transferring the resulting binary files to the mainframe disk and then running the program after it is restored to the mainframe's native format. FDOT is currently using the CPORT and CIMPORT procedures to create and restore transport files. Specifically, a transport file is first created using PROC CPORT at the source host and then imported to the target host using PROC CIMPORT. The following sections describe in detail the procedures involved in each step in the program transporting process.

The first step in moving SAS data and catalog files between different operating environments is to convert these transport files to a common language format, known as transport format using Procedure CPORT. A transport file contains one or more data sets or catalogs or both in transport format. The format includes a header to describe the content of the file and the contents of the member types that are represented in binary format. In procedure CPORT, export means to put a SAS data library, a SAS catalog, or a SAS data set into transport format. The transport format that PROC CPORT writes is the same for all environments and for many release of the SAS System. PROC CPORT exports catalogs and data sets, either singly or as a SAS data

library. In this project, only a SAS catalog containing FRAME, MENU and SCL entries needs to be transported. To export the SAS catalog, the following code was executed in SAS Program Editor window:

```
LIBNAME CQRCODE 'SAS-library-name';
FILENAME TRANFILE 'transport-file-name';
PROC CPORT CATALOG = cqrcode.mainframe FILE = tranfile;
Run;
```

First, a LIBNAME statement is used to define the physical location of the source file in native format for which the transport file is created. By default, SAS automatically selects the base engine to write data libraries to disk format when no engine name is specified on the LIBNAME statement. Second, a FILENAME statement is used to define a physical location to store the transport file that is created, followed by the PROC CPORT statement to read the source file from native format and to write its content in transport format. Finally, use the RUN statement to execute the SAS statements.

3.2 Program Transfer

Transfer is the process of conveying a file between hosts across a network. Various products are available for performing such an operation. In this project, the *EXTRA!* program, Version 6.5 was employed to move the transport-format SAS catalog from the source host to the target host. To transfer files, *EXTRA!* must communicate with a file transfer tool on the host. In the three available file transfer tools in *EXTRA!*, the Internet standard File Transfer Protocol (FTP) was used because it is generally the fastest method for transferring files. FTP copies files across a network connection between the source host and a target host. FTP runs from the initiating host, which can be either the source host or the target host. In order to transfer a file to a target host across a network, the binary (or image) format transfer is specified. This format guarantees a consistent file structure for any operating environment that runs SAS.

To transfer files, first create a session for the FDOT's installed host type by selecting "Creating a New Session" on the dialog box in Figure 1 and the New Session wizard will be displayed. The wizard's function is to create a session with many default settings, which determine how the session appears, operates, and interacts with users. To create a session, the following information must be provided to the wizard:

- Session type — display or printer.
- Connection type and settings.
- File transfer type applicable to the selected host type.

The following procedures explicitly show the steps needed to create a new section via the New Section wizard:

- Select a host type and click Next.
- Select a session type and click Next.
- Select a connection type and click Next.
- Set the options for the selected connection and click Next.

- Select a file transfer type.
- Click Finish for EXTRA! to display the new session or choose Save from the File menu. A session can be saved with any name. The default file extensions indicate the session type, i.e., .EDP for display sessions and .EPP for printer sessions.

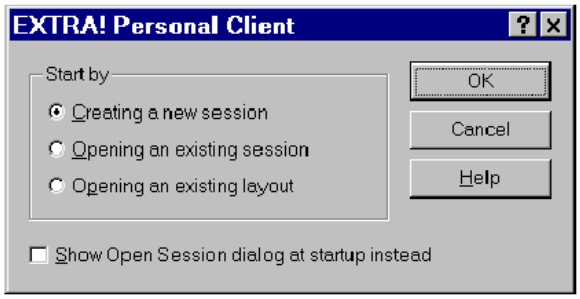

Figure 1. Create/Open a Section in EXTRA!

Second, select Transfer File from the Tool menu to display the Transfer File dialog box shown in Figure 2 and then specify the path and file name of the exported SAS file name in the host environment on this dialog box. Specify FTP (File Transfer Protocol) and choose Session Type from the Options menu and change File Transfer Protocol, then choose Settings. Choose and set options from the following tabbed pages in Setting:

- General: Set options for connecting host. Use host name of DOT1 and FTP port of 21.
- Proxies: Configure proxy server option. Choose "No Proxy".
- File Handling: Save or delete your file transfer scheme.
- Folders: Set options for determining the destination location of files. Choose the receive option of "Create path if it does not already exist".
- File Types: Set option for determining the file type during file transfer. Use Binary file type.

Choose Transfer from the Transfer File dialog box to start the transfer. A File Transfer Progress dialog box displays the transfer progress followed by a transfer Summary dialog box when the transfer is completed.

3.3 Program Restore

To restore the transport-format SAS catalog in SAS system on mainframe, CIMPORT procedure was used. The CIMPORT procedure imports a transport file that was created (exported) by the CPORT procedure. PROC CIMPORT, which can only read transport files that PROC CPORT creates, restores the transport file to its original form as a SAS catalog, SAS data set, or SAS data library. The following code was executed under the SAS software environment on the FDOT mainframe system:

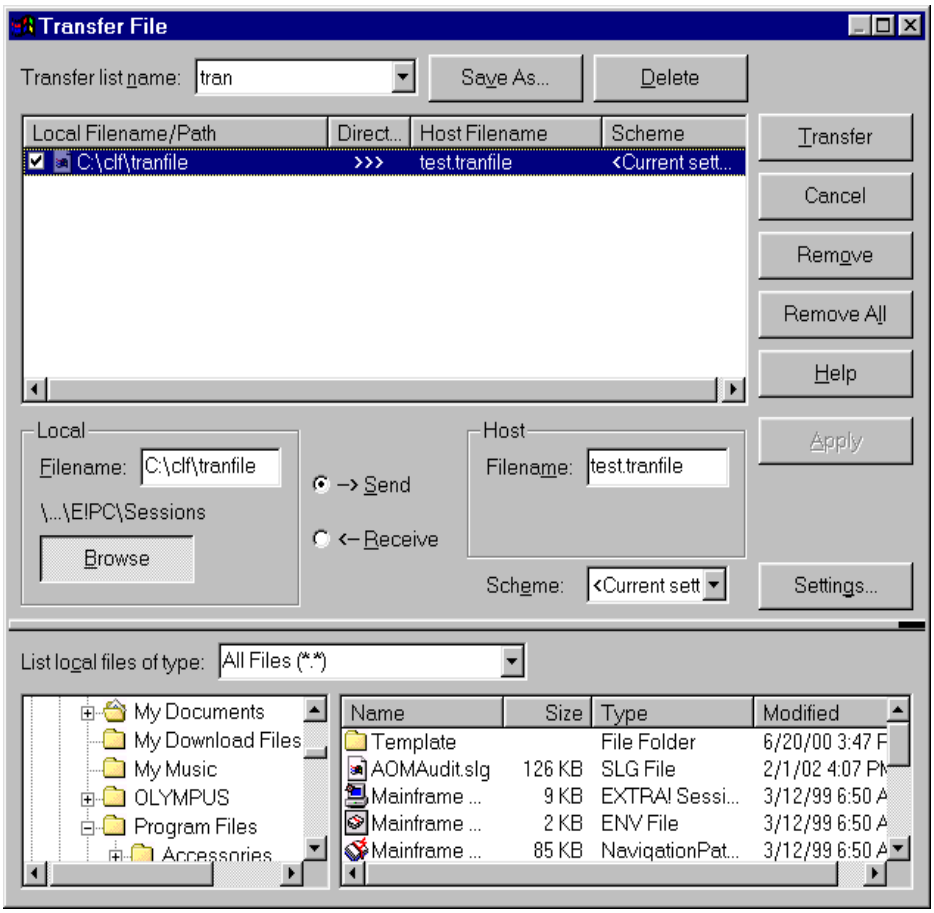

Figure 2. File Transfer Using EXTRA

LIBNAME CQRCODE *'SAS-library-name';*
FILENAME TRANFILE *'transport-file-name';*
PROC CIMPORT CATALOG = *cqrcode.mainframe* INFILE = *tranfile*;
Run;

First, a LIBNAME statement is used to define the physical location in which the file to be imported into the target host native format is stored. Second, use a FILENAME statement to define a physical location for the transport file that was transferred to the target host. Third, use PROC CIMPORT to read in the transport file and to write out its content in the target host native format. Finally, use the RUN statement to execute the SAS statements.

3.4 Limitations

The disadvantage of using PROC CPORT and PROC CIMPORT is that they do not allow the file transport operation from a later version to an earlier version, which is known as regressing (e.g., from Version 8 to Version 6). PROC CPORT and PROC CIMPORT move files only from an earlier version to a later version (e.g., from Version 6 to Version 8) or between the same versions (e.g., from one Version 8 host to another Version 8 host). However, SAS files can be moved between releases of Version 6, e.g., from Release 6.12 to Release 6.08. In addition,

PROC CPORT and PROC CIMPORT do not support the transport of any type of view, multi-dimensional databases (MDDBs), or data mining databases (DMDBs).

4. PROGRAM INTERFACE

This section describes how to run the SAS/AF interface program that is created to access the pavement management data stored on the FDOT mainframe system.

4.1 Execute SAS/AF Applications

Although SAS/AF software is needed to create applications for the SAS software, only base SAS product is required to run the interface program. Therefore, users can run the transferred SAS catalog in the SAS system on the mainframe environment by entering the following AF command:

AF Catalog = *libname.catalogname.framename.FRAME*

Users can also launch a SAS/AF application without opening any intervening SAS System windows such as the PROGRAM EDITOR, LOG, or OUTPUT windows. The SAS system option INITCMD can be used to open an AF window when a SAS session starts. The INITCMD system option must either be used in conjunction with the command that starts the SAS session or be specified in the SAS configuration file. When an application is invoked with the INITCMD system option, the SAS session automatically ends when the application ends.

In this project, the INITCMD system option is issued in conjunction with the command that starts the SAS session with another system option, AUTOEXEC. The AUTOEXEC system option specifies the autoexec file. The autoexec file contains SAS statements that are executed automatically when the SAS System is invoked. The autoexec file contains two LIBNAME statements for the SAS/AF program and for SAS data library where users retrieve the databases. The autoexec file that specifies the AF application and SAS data library is shown as follows:

LIBNAME CQRCODE 'kn983ml.cqrcode'; (AF application)
LIBNAME *db2real* '*sasreal.sassql.view*' DISP = *shr*; (SAS data library)
RUN;

Under TSO, a SAS CLIST command can be used to launch SAS session and specify any SAS system options. CLIST is a command list composed of a planned, executable sequence of TSO commands, subcommands, and command procedure statements that control various system and program operations. The following SAS CLIST command was specifically created and saved in the FDOTSAS file on the FDOT mainframe system:

SASV8 options ('Autoexec = "Kn983ml.tets.fdotdsas.ini" INICMD "AF cat = cqrcode.new.frmmenu.frame"')

4.2 Open Data File

After executing the CLIST command under TSO (i.e., *EX FDOTSAS*), the AF application will open a window with a FILE menu as illustrated in Figure 3. Pull down the FILE menu and select the option of OPEN DATABASE. An Open File dialog box will pop up to allow users browse and select the database for analysis. Users can choose SAS data file, which is a type of SAS data set that contains both the data values and the descriptor information, or SAS data view, which is

a SAS data set that uses descriptor information and data from other files. While a SAS data file actually contains data values, a SAS data view contains only references to data stored elsewhere. The default type of data set is SAS data view.

Figure 3. File Menu

4.3 Select Target Variable and Material Type

Select a specific data source will pop up a list box, i.e., Box Column as shown in Figure 4, for users to select the target variable for statistical analysis. Currently, only CQRSLTN is available for selection. Click the variable in the Column box and then click SELECT. Once the target variable has been identified, the Class, Choices, Conditions, and Results boxes will be popped up along with a text entry box requesting users to enter a material ID. Users have to specify the material ID (see Figure 5) before proceeding to the next step.

Figure 4. Select Target Variable

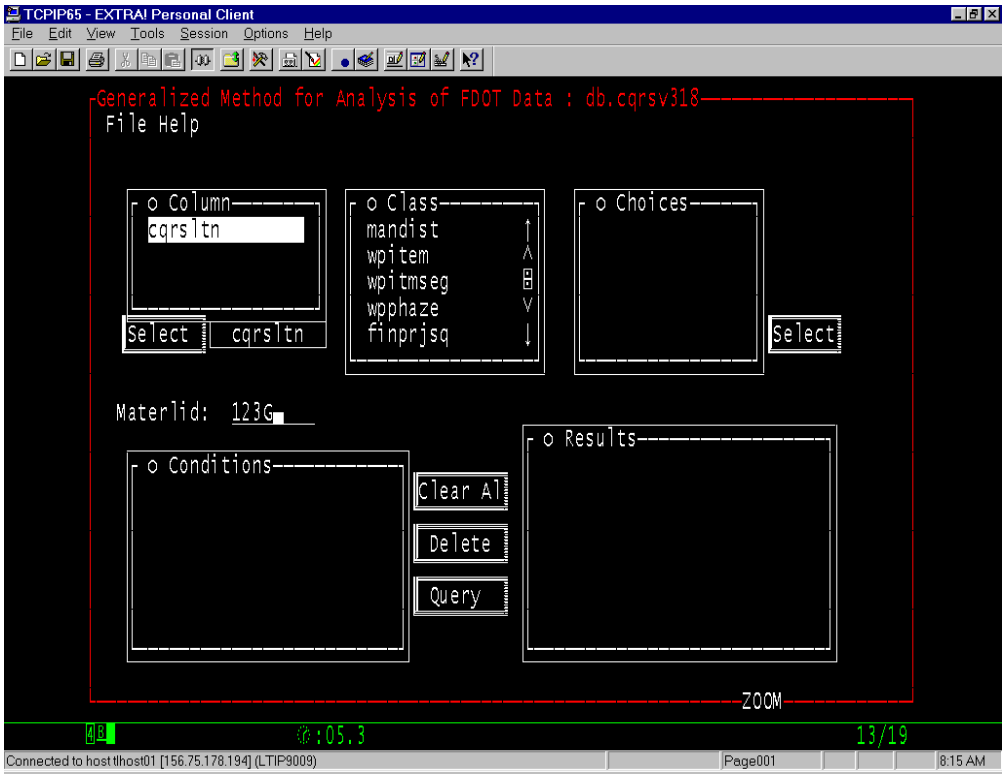

Figure 5. Define Material ID

4.4 Select Class Variables and Values

Once the material ID is specified, the next step is to define the class variables that will be used to query a subset of records for analysis. The program will display the index variables contained in the selected data source in the Class box. Currently, querying non-index variables is likely to consume more CPU time than a data source is allowed. In fact, querying an index variable from a large data set such as CQRSV318 already exceeds the system resource that can be allocated. Consequently, users can only used index variables to query and create subset data.

Once users click any index variable in the Class box, the program will search the dataset for all of the possible values for the highlighted class variable. All unique values associated with the selected class variable will be listed in the CHOICE box. As previously mentioned, the data retrieving process requires significant CPU time. An INFORMATION message box, as shown in Figure 6, is thus created. Click OK to close the message box and proceed to next step for further class specification and click cancel to abort.

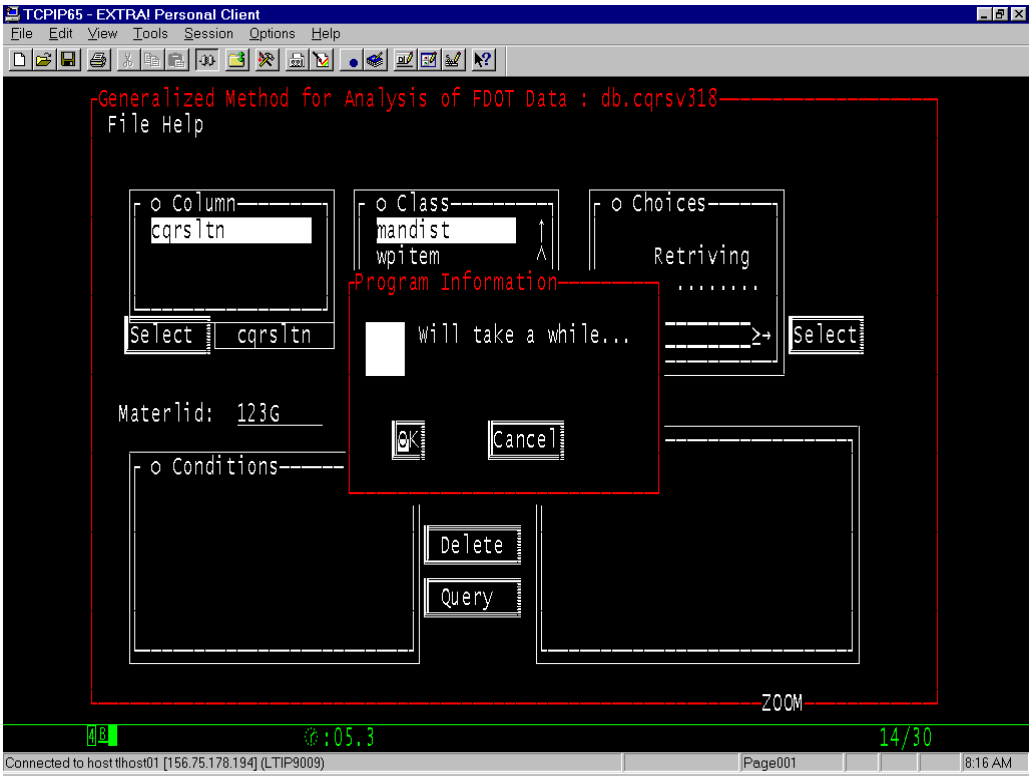

Figure 6. Program Message Box

Once all of the possible values for the class variables are shown in the CHOICES box, users can then click and highlight the choice value to query the records with their class variables set to the desired value. Clicking Select will display the query condition that is specified, as illustrated in Figure 7, in the CONDITION box. Users can follow the steps described in this section to define more query conditions. They can also delete or clear all of the query conditions that are previous defined by clicking the Delete or Clear All buttons.

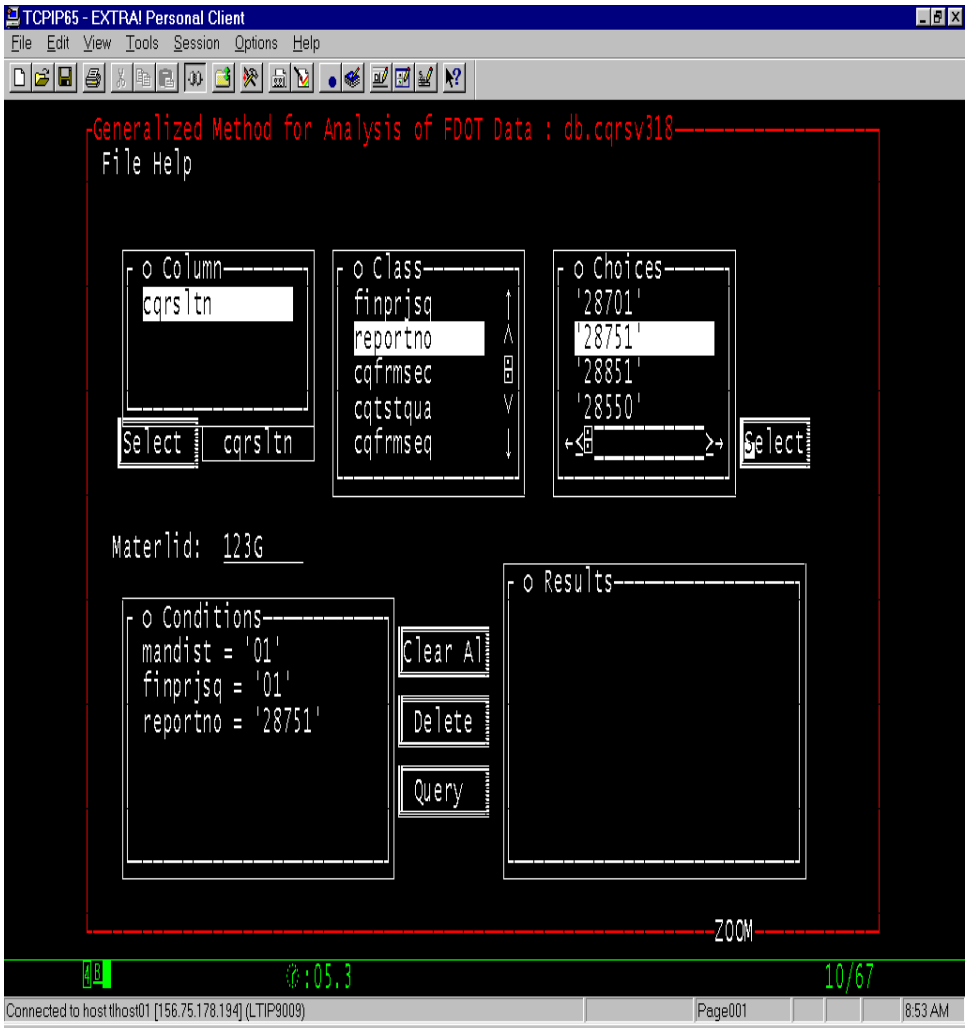

Figure 7. Example of Data Query

Click Query after all of the desired query conditions are specified to display the following statistic parameters for the target variable: mean, standard deviation, variance, maximum value, minimum value, summation, and total number of records in the subset. Figure 8 shows the final results for CQRSLTN for material type 123G in the CQR318 data view. The queries are set to include data collected from Managing District I (MANDIST = '01') at Financial Project Sequence one (FINPRJSQ = '01') with test results recorded on Test Report Form 28751 (REPORTNO = '28751').

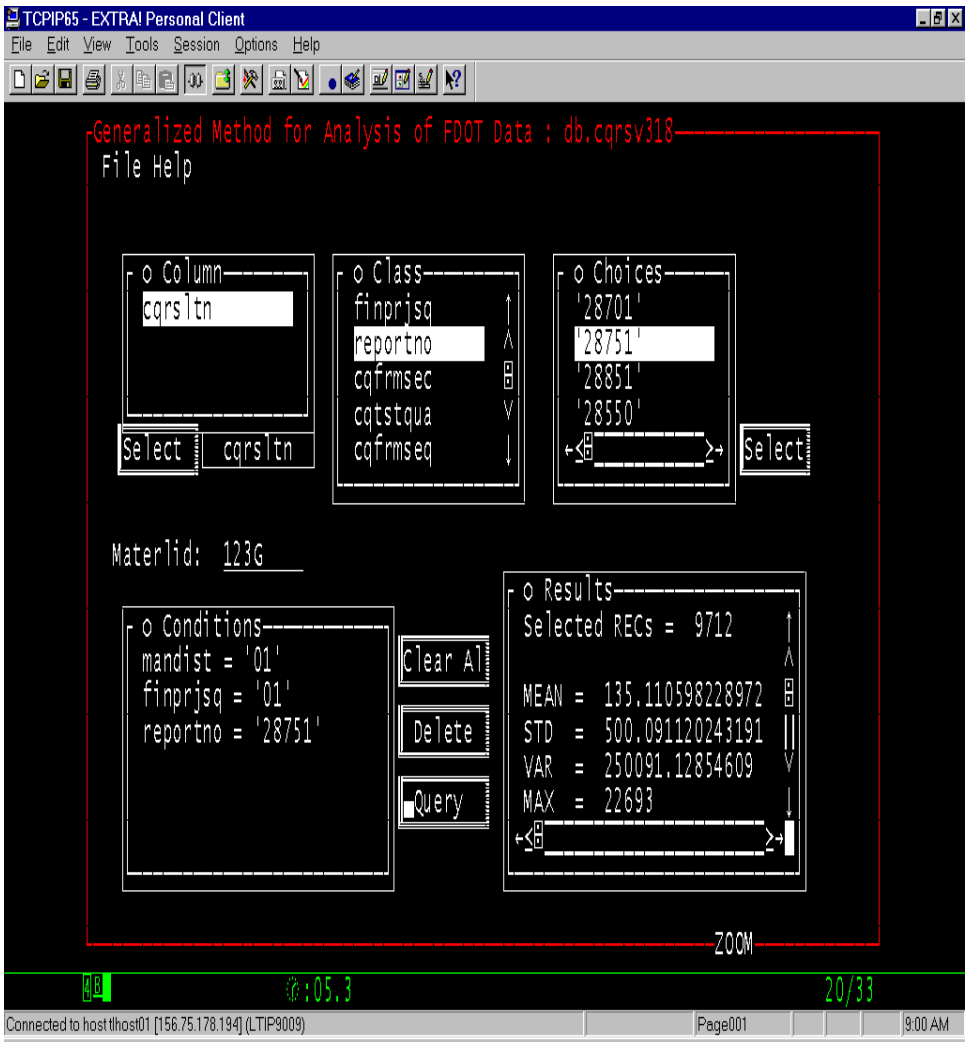

Figure 8. Example Output Screen

5. SUMMARY AND FUTURE POSSIBLE ENHANCEMENTS

This study developed a SAS/AF computer tool to perform initial on-line statistical analyses on the FDOT's PMS mainframe database. The program was not designed to perform sophisticated statistical analyses but flexible enough to accommodate future enhancements to improve its capabilities. Future possible enhancements to the program may include the following:

- Advance the pre-process module to include more data manipulation functions for data retrieval;
- Develop a display module to present the data in graphics;
- Ability to conduct more complicated statistical analysis; and
- Enhance the output module to display results in a more visually appealing and organized format.

REFERENCES

[MM] *Material Manual*, Florida Department of Transportation, Tallahassee, FL, March 2000.

[PING] Ping, V.W. and H. He, *Evaluation of Flexible Pavement Performance Life in Florida*, Final Report, Florida Department of Transportation, Tallahassee, FL, June 1998.

[SASDOC] *SAS-OnlineDOC*, Version 8, SAS Institute Inc., Cary, NC, 1999.

[STRG] *Sampling, Testing, and Reporting Guide*, State Materials Office, Florida Department of Transportation, Tallahassee, FL, November 1991.

APPENDIX PROGRAM SOURCE CODE

```
DCL CHAR mytable;
DCL CHAR rc_char;
DCL CHAR(1500) cc;
DCL CHAR name1;
DCL CHAR name2;
DCL NUM  message_ID = MAKELIST(1);
DCL NUM  rc_num;
DCL NUM  table_ID;
DCL NUM  varlist = MAKELIST();
DCL num  box2list = MAKELIST();
DCL NUM  numtemp;


INIT:

  /* Startup Information */
  CONTROL ALWAYS;
   _FRAME_.title = 'Generalized Method for Analysis of FDOT Data';


  LstBoxTargetVariable.visible = 'No';

  Pushbutton_TargetVariable.visible = 'No';
  txtlbl_TargetVariable.visible = 'No';
  /* add dual selector control from sashelp.classes.Dualselector_c.class */

  CALL EXECCMDI('NUMBER OFF;ZOOM ON;PMENU ON');
  rc_num = PMENU('Cqrcode.new.Frame.Pmenu');
  rc=clearlist(listbox3.items,'y');
  table_ID=0;

RETURN;

MAIN:


RETURN;

TERM:
  IF (table_ID > 0) THEN DO;
      rc_num = CLOSE(table_ID);
  END;
  IF (varlist > 0) THEN DO;
      rc_num = DELLIST(varlist);
  END;
  CALL EXECCMDI('ZOOM OFF;PMENU OFF');
RETURN;

OpenFileDialog:
  DCL CHAR temp;

  mytable = OPENSASFILEDIALOG('VIEW DATA');

  temp = 'Generalized Method for Analysis of FDOT Data :  ';
```

```
  SUBSTR(temp, LENGTH(temp, 'NOTRIM')) = LOWCASE(mytable);
  _FRAME_.TITLE = temp;
  LINK OpenFile;

RETURN;


EXITFileDialog:
 rc=close(table_ID);

RETURN;


OpenFile:
  DCL CHAR vname;
  if(table_ID>0) then rc=close(table_ID);
  table_ID = OPEN(mytable,'in');

  allowvarlist=makelist();
  rc_num = INSERTC(allowvarlist,'CQRSLTN', -1);

  allowclasslist=makelist();
  rc_num = INSERTC(allowclasslist,'MANDIST', -1);
  rc_num = INSERTC(allowclasslist,'WPITEM', -1);
  rc_num = INSERTC(allowclasslist,'WPITMSEG', -1);
  rc_num = INSERTC(allowclasslist,'WPPHAZE', -1);
  rc_num = INSERTC(allowclasslist,'FINPRJSQ', -1);
  rc_num = INSERTC(allowclasslist,'REPORTNO', -1);
  rc_num = INSERTC(allowclasslist,'CQFRMSEC', -1);
  rc_num = INSERTC(allowclasslist,'CQTSTQUA', -1);
  rc_num = INSERTC(allowclasslist,'CQFRMSEQ', -1);

  IF (table_ID > 0) THEN
    DO;
       ListBox1.items =makelist();
       ListBox2.items =makelist();
       ListBox3.items =makelist();
       ListBox4.items =makelist();

       TxtLbl_TargetVariable.label='';

      /* get number of columns in the SAS table */
      nv = ATTRN(table_ID, 'NVARS');
      varlist = MAKELIST();
      invarlist= MAKELIST();


     DO j = 1 to listlen(allowvarlist);
      allowvar=getitemc(allowvarlist,j);
      DO i = 1 TO nv;
        vname = VARNAME(table_ID, i);
        IF (allowvar=upcase(vname)) THEN rc_num = INSERTC(varlist,
LOWCASE(vname), -1);
      END;
     END;
```

```
     DO j = 1 to listlen(allowclasslist);
      allowclass=getitemc(allowclasslist,j);
      DO i = 1 TO nv;
        vname = VARNAME(table_ID, i);
        IF (allowclass=upcase(vname)) THEN rc_num = INSERTC(invarlist,
LOWCASE(vname), -1);
      END;
     END;
      LstBoxTargetVariable.visible = 'Yes';
      LstBoxTargetVariable.items = varlist;
      Pushbutton_TargetVariable.visible = 'Yes';
      txtlbl_TargetVariable.visible = 'Yes';
    END;

  ELSE
    DO;
        RETURN;
    END;

RETURN;


Pushbutton_TargetVariable:

  Pushbutton_TargetVariable._onClick();
    IF(LstBoxTargetVariable.selectedcount=1) THEN
     DO;
       TxtLbl_TargetVariable.label = LstBoxTargetVariable.selecteditem;
       link QuaryVariable;
     END;
    ELSE
     DO;
       numtemp = SETITEMC(message_ID, 'Please select a variable.');
       rc_char = MESSAGEBOX(numtemp, 'I', 'O', 'Program Information', 'O',
'');
     END;

RETURN;

LstBoxTargetVariable:

RETURN;


QuaryVariable:

  IF (length(trim(TxtLbl_TargetVariable.label)) NE 0) THEN
    DO;
     textlabel1.visible='Yes';
     textentry1.visible='Yes';
     listbox1.visible = 'Yes';
     listbox2.visible = 'Yes';
     listbox3.visible = 'Yes';
     listbox4.visible = 'Yes';
     pushbutton2.visible = 'Yes';
     pushbutton1.visible='yes';
     pushbutton3.visible='yes';
```

```
     pushbutton4.visible='yes';
     ListBox1.items = invarlist;
    END;
  ELSE
    DO;
      listbox1.visible = 'Yes';
    END;

RETURN;


textentry1:

 textentry1._onclick();
 hh=textentry1.text;

RETURN;


listbox1:

  listbox1._onclick(listbox1.selectedindex);
  processlist=makelist();
  item1='   ';
  item2='    Retriving data';
  item3='      ................';
  rc_num = INSERTC( processlist, item1, -1);
  rc_num = INSERTC( processlist, item2, -1);
  rc_num = INSERTC( processlist, item3, -1);

  listbox2.items=processlist;

  numtemp = SETITEMC(message_ID, 'Will take a while...');
  rc_char = MESSAGEBOX(numtemp, 'I', 'OC', 'Program Information', 'O', '');
  IF (rc_char ='OK') THEN link variablerange;

RETURN;


listbox2:

  listbox2._onclick(listbox2.selectedindex);

RETURN;


VariableRange:

 rc=where(table_ID);

 vindex=varnum(table_ID,listbox1.selecteditem);
 vindex1=varnum(table_ID,TxtLbl_TargetVariable.label);

 vtype=VARTYPE(table_ID,vindex);
 box1list = MAKELIST();

 rc=where(table_ID);
```

```
 cc='materlid='||quote(hh);
 rc=where(table_ID,cc);
 rc=fetch(table_ID);
 IF (rc=-1) THEN
  DO;
    numtemp = SETITEMC(message_ID, 'Invalid Materlid.');
    rc_char = MESSAGEBOX(numtemp, 'I', 'O', 'Program Information', 'O', '');
    RETURN;
  END;

 IF(vtype='C') THEN vvalue="'"||getvarc(table_ID,vindex)||"'";
 IF(vtype='N') THEN vvalue=getvarn(table_ID,vindex);
 rc_num = INSERTC(box1list, vvalue, -1);
 qq=listbox1.selecteditem;
 cc=cc||' and '||qq||'<>'||vvalue;
 rc=where(table_ID,trim(cc));

 i=1;
 DO WHILE((fetch(table_ID) ne -1) and (i < 50));
   i=i+1;
   IF(vtype='C') THEN vvalue="'"||getvarc(table_ID,vindex)||"'";
   IF(vtype='N') THEN vvalue=getvarn(table_ID,vindex);
   rc_num = INSERTC(box1list, vvalue, -1);
   rc=where(table_ID);
   cc=cc||' and ' ||qq||'<>'||vvalue;

   rc=where(table_ID,trim(cc));
 END;

 IF(i< 50) THEN
   DO;
     ListBox2.items = box1list;
   END;
 ELSE
   DO;
     numtemp = SETITEMC(message_ID, 'Too many choices.');
     rc_char = MESSAGEBOX(numtemp, 'I', 'O', 'Program Information', 'O', '');
   END;

RETURN;


pushbutton2:

  pushbutton2._onclick();
  box2list=listbox3.items;

  IF(listbox2.selectedcount>0) THEN
   DO;

    DO i = 1 TO listbox2.selectedcount;
       name1=listbox1.selecteditem;
       name2=getitemc(listbox2.selecteditems,i);
       name3=name1||' = '||name2;
       duplic1=0;
        IF( LISTLEN(listbox3.items)>0) THEN
          Do;
```

```
           Do j = 1 TO LISTLEN(listbox3.items);
              IF(name3 = getitemc(listbox3.items,j)) THEN duplic1=1;
            END;
         END;


        IF(duplic1=0) THEN
          Do;
             rc_num = INSERTC(box2list,name3 , -1);
          END;
        ELSE
          Do;
           numtemp = SETITEMC(message_ID, name3||'   exists');
           rc_char = MESSAGEBOX(numtemp, 'I', 'O', 'Program Information',
'O', '');
          END;
     END;

     ListBox3.items = box2list;
   END;
   ELSE
     DO;
       numtemp = SETITEMC(message_ID, 'No item selected.');
       rc_char = MESSAGEBOX(numtemp, 'I', 'O', 'Program Information', 'O',
'');
     END;

RETURN;


pushbutton3:

  pushbutton3._onclick();
  n=listlen(listbox3.items);
  IF(n=0) THEN
  DO;
     numtemp = SETITEMC(message_ID, 'Query without condition?');
     rc_char = MESSAGEBOX(numtemp, 'I', 'OC', 'Program Information', 'O',
'');
     IF (rc_char ='CANCEL') THEN RETURN;
  END;

     processlist=makelist();
     item1='   ';
     item2='   Processing.................';
     item3='      Please wait...';
     rc_num = INSERTC( processlist, item1, -1);
     rc_num = INSERTC( processlist, item2, -1);
     rc_num = INSERTC( processlist, item3, -1);
     listbox4.items=processlist;
     numtemp = SETITEMC(message_ID, 'Will take a while...');
     rc_char = MESSAGEBOX(numtemp, 'I', 'OC', 'Program Information', 'O',
'');
     IF (rc_char ='OK') THEN link doquery;

RETURN;
```

```
pushbutton1:

  pushbutton1._onclick();
  templist3=makelist();
  listbox3.items=templist3;

RETURN;


pushbutton4:

  pushbutton4._onclick();
  IF(listbox3.selectedcount=1) THEN
    DO;
      templist= MAKELIST();
      templist=DELITEM(listbox3.items,listbox3.selectedindex);
      listbox3.items=templist;
    END;

  ELSE
    DO;
      numtemp = SETITEMC(message_ID, 'No item selected.');
      rc_char = MESSAGEBOX(numtemp, 'I', 'O', 'Program Information', 'O',
'');
    END;

RETURN;


Doquery:

 rc=where(table_ID);
 rc=where(table_ID,'materlid='||quote(hh));
 rc=fetch(table_ID);
 IF (rc=-1) THEN
  DO;
    numtemp = SETITEMC(message_ID, 'Invalid Materlid.');
    rc_char = MESSAGEBOX(numtemp, 'I', 'O', 'Program Information', 'O', '');
    RETURN;
  END;

 IF(listlen(listbox3.items) >0) THEN
  DO;
    cond1=getitemc(listbox3.items,1);
    rc=where(table_ID,'also '||cond1);

    IF(listlen(listbox3.items) >1) THEN
      DO;
        DO i = 2 TO listlen(listbox3.items);
          condi=getitemc(listbox3.items,i);
          rc=where(table_ID,'also '||condi);
        END;
      END;
   END;
```

```
  i=0;
  DO WHILE((fetch(table_ID) ne -1) and (i<86030));
     i=i+1;
  END;

  IF(i<86030) THEN
    DO;
      statcode=varstat(table_ID,TxtLbl_TargetVariable.label,'mean std var max
min sum n', sta1,sta2,sta3,sta4,sta5,sta6,rns);
      templist=makelist();
      rc_num = INSERTC(templist,'Selected RECs =  '||rns , -1);
      rc_num = INSERTC(templist,'  ' , -1);
      rc_num = INSERTC(templist,'MEAN =  '||sta1 , -1);
      rc_num = INSERTC(templist,'STD  =  '||sta2 , -1);
      rc_num = INSERTC(templist,'VAR  =  '||sta3 , -1);
      rc_num = INSERTC(templist,'MAX  =  '||sta4 , -1);
      rc_num = INSERTC(templist,'MIN  =  '||sta5 , -1);
      rc_num = INSERTC(templist,'SUM  =  '||sta6 , -1);
      listbox4.items=templist;
    END;
   ELSE
    DO;
      numtemp = SETITEMC(message_ID, 'Too many records.  Please be more
selecticve.');
      rc_char = MESSAGEBOX(numtemp, 'I', 'O', 'Program Information', 'O',
'');
    END;

RETURN;
```