SAS® Viya® Intermediate Programming Exam

Programming in SAS Viya Concepts (5-10%)

Describe the SAS Viya architecture.

- Compute Server vs. Cloud Analytics Server (CAS)
- Serial vs parallel processing
- In-memory processing
- Open source integration

Explain when to use the CAS server for programming tasks.

- Size of data
- Type of SAS procedure used

Managing Data with CAS Enabled Procedures (10-15%)

Explain how to access and use CAS Libraries (caslibs).

- Establish CAS sessions with the CAS statement.
- Caslib attributes (Session, local, active, personal)
- Properties of the casuser caslib
- Use the CASLIB statement to assign session-scope caslibs
- Assign a libref to a caslib with the LIBNAME statement and CAS engine
- View the contents of a caslib with PROC CASUTIL

Describe how to load data into In-Memory Tables.

- Load data files into memory
- Client-side vs server-side files
- Loading client-side data (PROC CASUTIL)
    - LOAD DATA= statement
- In-memory table scope (Session vs Global, promoting tables)
- Loading server-side data sources (PROC CASUTIL)
    - LOAD CASDATA= statement
    - ALTERTABLE statement
- Alternate data loading methods (DATA step, PROC SQL, PROC IMPORT)

Describe how to save and drop In-Memory Tables.

- SASHDAT files
- PROC CASUTIL (SAVE and DROPTABLE statements)

Describe CAS column data types.

- Properties of character column variable types
  - CHAR
  - VARCHAR()
  - Determine when to use CHAR vs VARCHAR()
- Properties of numeric column variable types
  - DOUBLE
  - INT32
  - INT64
- Create varchar column variables with the LENGTH statement
- Determine appropriate column data types for example data

DATA Step and SQL programming in CAS (10-15%)

Explain how SAS determines where code executes.

- Location of the input/output data
- What procedures are being run
- What statements/functions are used
- SESSREF= option on the DATA statement
- SESSREF= option within FedSQL
- MSGLEVEL= system option

Explain threading within the SAS DATA step.

- Where code executes: CAS, Compute Server
- Effect of threads on the DATA step
- _THREADID_ and _NTHREADS_ automatic variables
- SINGLE= DATA step option
- Adjust DATA Step code when accumulating totals
- Explain how BY groups are processed in CAS enabled DATA step code
  - Relationship between the distribution of threads and BY GROUP variables
  - DATA step BY GROUP processing and sorting

Update DATA step code to run in CAS.

- DESCENDING keyword
- WHERE= option
- INFILE/INPUT/DATALINES statements
- MODIFY/REMOVE/REPLACE statements
- DATALIMIT= option
- Functions not supported in CAS (Examples: RANBIN, RANUNI, SYMGET, FILEREF, GIT functions)

Update PROC SQL code to run as PROC FEDSQL code.

- Data types
- Supported statements
- Mnemonics vs operators
- SESSREF= option
- Remerge
- Calculated keyword
- SET operations
- Correlated subqueries
- Dictionary tables
- Views
- LIMIT clause
- FORMAT=, LABEL= vs PROC CASUTIL ALTERTABLE CASDATA statemen

CAS-Enabled Procedures and User Defined Formats (5-10%)

Identify common procedures that run only on the Compute Server.

- PROC FREQ and UNIVARIATE
- SG Graphics procedures

Use common procedures that run in both the CAS and Compute Server.

- How SAS determines where the procedure runs
  - Location of the input/output data
  - Which functions/options are used in the code
- PROC MEANS & PROC SUMMARY
  - Common Supported Statements: CLASS/BY/VAR/WHERE/FORMAT
  - Common Supported Statistics: N, NMISS, MIN, MAX, RANGE, MEAN, SUM, STDERR, VAR)
  - Common Unsupported Statistics: MEDIAN, MODE, percentiles
- PROC TRANSPOSE
- BY GROUP processing in CAS
- Use the log file to identify where code executed

Use Common summary procedures that run only in CAS.

- PROC FREQTAB
  - TABLE statement
  - BY statement
- PROC MDSUMMARY
  - VAR statement
  - OUTPUT statement
  - GROUPBY statement

Discuss how user-defined formats are used and stored in CAS.

- Location where formats are stored within CAS
- Saving formats to caslibs with the CASFMTLIB= option
- Save formats to and retrieve from permanent SASHDAT files with a CAS statement
- Assigning formats to in-memory tables

CAS Language Basics (15-20%)

Describe the CASL programming language.

- Action Sets
- Actions
- Parameters
- Statements

Create and manipulate CASL variables.

- CASL variables vs. SAS variables
- CASL Variable Data Types (int32, int64, double, string)
- DESCRIBE statement
- PRINT statement
- Built-in functions vs common functions

Use arrays in CASL programs.

- Define arrays
  - Array element data types
  - Nested arrays
- Retrieve values from arrays
- Array operators (||, &, /, ==)
- Array functions (DIM, SORT, SORT_REV)
- Use DO-OVER loops to process arrays

Use dictionaries in CASL programs.

- Define dictionaries
- Retrieve values from dictionaries with bracket and dot notation
- Retrieve nested dictionary values with dot notation
- Use DO-OVER loops to process dictionaries

Capture the results returned from CAS actions.

- Capture results from CAS actions as variables/objects/dictionaries
- Verify the return status to check for a successful action
- Use DO-OVER loops to process the rows of a result table
- Save results tables:
  - to In-memory tables with the SAVERESULT statement
  - to caslib data sources with the table.save action
  - to SAS data sets with the SAVERESULT statement
  - to CSV files with the SAVERESULT statement

Use source blocks in CASL programs.

- Identify when SOURCE blocks are required for code substitution
- use SOURCE blocks for DATA step and FedSQL code substitutions
- Use SOURCE blocks for code substitution in the computedVarsProgram= parameter of CAS actions

Access Data with CAS Actions (5-10%)

Use CAS actions to access and explore data sources.

- Create caslibs with the table.addCaslib action
- View available caslib information with the table.caslibInfo action
- View data source files information with the table.fileInfo action
- Load server side files into memory with the table.loadTable action
    - parameters: path, caslib, casOut, importOptions

Use CAS actions to manage in-memory tables.

- View in-memory table information with the table.tableInfo action
- Load client side files into memory with the table.upload action
    - parameters: path, casOut
- Explain how database files load into memory with data connectors
- Promote in-memory tables
- Save tables with table.save
    - parameters caslib= and table=
- Remove tables from memory with the table.dropTable action

Explore and Validate Data with CAS actions(5-10%)

Investigate in-memory data table properties and contents.

- Table action set
    - columnInfo action
    - fetch action (parameters table=, fetchVars=, sortBy=, from=, to=)
    - WHERE clause
- Simple action set
    - numRows action
    - distinct action
- Identify duplicate values within in-memory table variables
    - deduplication.deduplicate action
- Compare table values with expected values to identify data that does not comply with business rules

Investigate results table properties and contents.

- Access results tables property values (nrows, ncols, name, title, attrs)
- Create an array from a single results table column
- Use functions with results table content (SUM, EXISTS)
- Filter results tables with the WHERE operator
- Create computed columns with the COMPUTE operator

Prepare Data with CAS Actions (20-25%)

Update the contents of in-memory tables with the table.update action.

- table= and set= parameters
  - WHERE= subparameter
- Use arrays of dictionaries as values for the set= parameter
- Use IFC and IFN functions to use conditional logic when updating tables
- Benefits and considerations of the table.update action

Create a copy of in-memory tables with the table.copyTable action.

- table= and casOut= parameters to define input and output data sets
- computedVars= parameter to set column attributes
- computedVarsProgram= parameter to set column values
- Benefits and considerations of the table.copyTable action
- Promoting the copied table

Convert character to numeric columns.

- Convert character to numeric columns with the inputn function
- Use informats
- Cast data types and the CAST function

Use data preparation action sets.

- dataStep action set
  - runCode action
  - RunCodeTable action
- fedSQL action set
  - execDirect action
  - CREATE TABLE, SELECT, DROP TABLE
  - query= parameter

Modify table attributes with the table.alterTable action.

- Update table attributes with rename=, label= parameters
- Change included columns with keep=, drop= parameters
- Change column attributes with the columns= parameter

Resolve missing values in tables with the dataPreprocess action set.

- Use the impute action to impute missing values
  - inputs= parameter
  - copyAllVars= parameter
  - MethodInterval= & valuesInterval= parameters
  - treatment of nominal and continuous variables

Transpose tables with the transpose.transpose action

- table= and casOut= to specify input and output tables
  - groupBy= subparameter to specify by groups
- parameters: transpose, ID=, NAME=, IDLABEL=, PREFIX=

Analyse and Summarize Data with CAS Actions (5-10%)

Summarize data with CAS actions.

- Produce summary statistics with the simple.summary action
  - table= and casOut= parameters
  - inputs= parameter
  - subSet= parameter
- Produce summary statistics with the aggregation.aggregate action
  - table= and casOut= parameters
  - varSpecs= parameter
    - name=, subset=, agg= subparameters
- Produce summary statistics with the dataPreprocess.rustats action
  - table=, inputs=, RequestPackages=, casOutStats= parameters
- Create one-way and two-way frequency tables
  - simple.freq
    - table= and inputs= parameters
  - freqTab.freqTab
    - table= and tabulate= parameters
    - vars= and cross= subparameters
  - simple.crossTab
    - table=, row=, col=, aggregator=, weight= parameters

Create Visualizations and Reports.

- Run CAS actions to produce summarized or subsets of results tables
- Use visualization procedures to produce graphics from summarized results tables
- Use SAS Output Delivery System (ODS)
  - CSVALL, EXCEL, POWERPOINT, RTF, PDF destinations

Note: All 33 main objectives will be tested on every exam. The additional details provide for additional explanation and define the entire domain that could be tested.