SAS® In-Memory Analytics for Hadoop

Venkateswarlu Toluchuri, United Health Group (Optum), Hyderabad, India

ABSTRACT

SAS® In-Memory Statistics is an analytical programming environment that enables a user to address many components of an analytics project in a single environment, rather having to switch between different applications. Users can easily prepare the raw data for different types of analytics they can explore the data to enhance the information extractions. They can apply a large variety of statistical and machine learning techniques to the data to have different approaches strive. The model comparison capabilities let them quickly find the best model, which they can deploy and score in the Hadoop environment. All of these different components of the analytics project are supported in a distributed in-memory environment for lightning fast processing.

This paper aims to highlight the tips and my recent approaches on analytics experience to deal with SAS/HDFS data, it get loaded in to LASR Server to apply analytics models on top of that. Presented here are multiple scenarios and elementary but pragmatic approaches that will enable analytics programmers to work efficiently within SAS® In-Memory Statistics for Hadoop.

INTRODUCTION

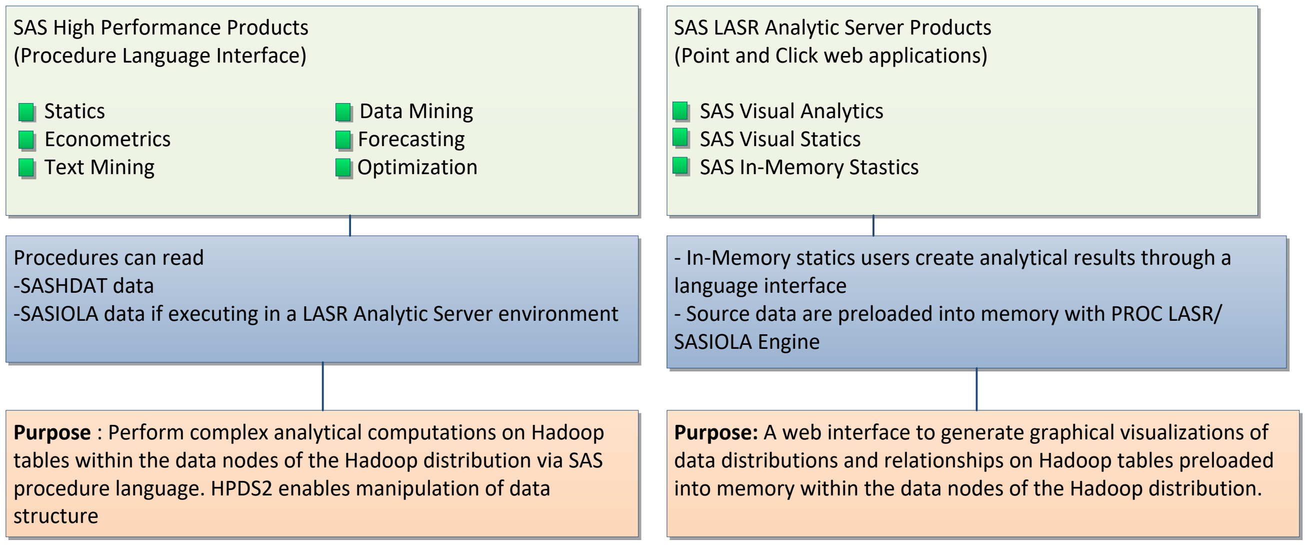
SAS® In-Memory Statistics enables multiple users to concurrently leverage fast in-memory data management, exploration, and statistical modeling capabilities in a highly interactive programming environment that seamlessly integrates with Hadoop. SAS® In-Memory Statistics loads all the data into memory once and can interact with the data without reloading the data each time. SAS High-Performance Analytics is not a server, it does not persist data in-memory. Instead data is lifted into memory for the duration of the high performance procedure. SAS High-Performance Analytics procedures lift data into memory, perform actions, and then shut down and free up resources. Each user makes his/her own copy of data in-memory without sharing and persistence. Thus, SAS High-Performance Analytics is not designed for high concurrency - multiple users do not share resources. In addition to Hadoop, the SAS High-Performance Analytics products also work on Teradata, Pivotal and Oracle. This analytic platform is not just for storage and return data it provides a platform for exploring, manipulating and analyzing data.

SAS In-Memory Statistics:

SAS In-Memory Statistics provides the data scientist or analytical expert with interactive programming access to large data sets stored in Hadoop.

1. Provides a single interactive programming environment for analytics.
2. Processing is performed in-memory in the LASR Analytic Server
3. Data can also be loaded from any source that a SAS session can access
4. Leverages the SAS LASR Analytic Server for data exploration, modeling and scoring

Types of SAS In-Memory Analytics Products:



SAS High Performance Products
(Procedure Language Interface)

- 🟩 Statics
- 🟩 Econometrics
- 🟩 Text Mining
- 🟩 Data Mining
- 🟩 Forecasting
- 🟩 Optimization

Procedures can read
-SASHDAT data
-SASIOLA data if executing in a LASR Analytic Server environment

Purpose : Perform complex analytical computations on Hadoop tables within the data nodes of the Hadoop distribution via SAS procedure language. HPDS2 enables manipulation of data structure

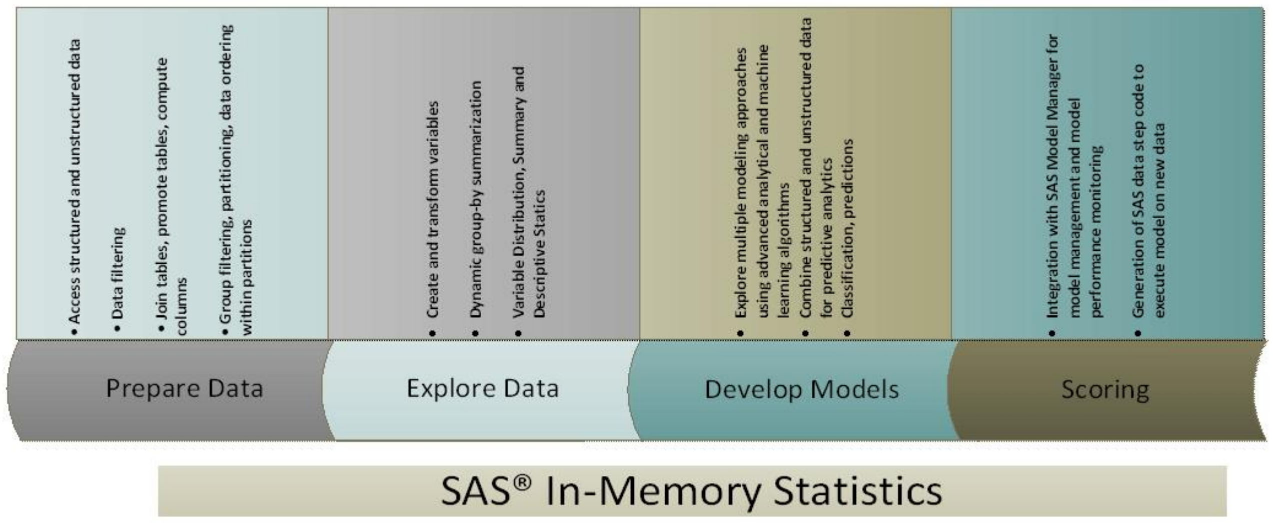
SAS LASR Analytic Server Products
(Point and Click web applications)

- 🟩 SAS Visual Analytics
- 🟩 SAS Visual Statics
- 🟩 SAS In-Memory Stastics

- In-Memory statics users create analytical results through a language interface
- Source data are preloaded into memory with PROC LASR/ SASIOLA Engine

Purpose: A web interface to generate graphical visualizations of data distributions and relationships on Hadoop tables preloaded into memory within the data nodes of the Hadoop distribution.

Analytic life cycle support SAS In-Memory:



• Access structured and unstructured data
• Data filtering
• Join tables, promote tables, compute columns
• Group filtering, partitioning, data ordering within partitions
• Create and transform variables
• Dynamic group-by summarization
• Variable Distribution, Summary and Descriptive Statics
• Explore multiple modeling approaches using advanced analytical and machine learning algorithms
• Combine structured and unstructured data for predictive analytics
• Classification, predictions
• Integration with SAS Model Manager for model management and model performance monitoring
• Generation of SAS data step code to execute model on new data
Prepare Data
Explore Data
Develop Models
Scoring
SAS® In-Memory Statistics

SAS LASR Server:

SAS LASR Analytic Server can work on unstructured text, such as social media feeds, news articles, and arbitrary collection of documents. The server uses text analytics to turn the unstructured text into numbers and counts and weights, and terms and topics with relationships, suitable for visualization and exploration. This also can take advantage of a distributed computing environment by distributing data and the workload among multiple machines and performing massively parallel processing in-memory

The SAS LASR Analytic Server can run in two different modes:

Distributed: This architecture is a distributed computing environment consists of multiple machines, each of which has some local storage and a great deal of memory. The local storage is important to store large data sets in distributed form. Loading data from disk is efficient when the LASR Analytic Server is co-located with a distributed data provider. The distributed data provider for SAS In-memory Statistics is Hadoop (HDFS). Data can be loaded direct from any data source that SAS can access.

Non-Distributed: A grid of computers is not required in this mode, the LASR Analytic Server can execute on a single machine using multi-threading instead of the hybrid gridding-and-multi-threading model in Distributed mode

This server processes client requests at extraordinarily high speeds due to the combination of hardware and software that is designed for rapid access to tables in memory. By loading tables into memory for analytic processing, the server enables business analysts to explore data and discover relationships in data at the speed of RAM.
Data and Analytic process happens in parallel on the server. In order to share the workload and still act as a single server, the SAS software that is installed on each blade implements the Message Passing Interface (MPI). The MPI implementation is used to enable communication between the blades.

Distributed LASR Analytic Server	Non-Distributed LASR Analytic Server
• Multiple Machines • SAS High Performance Analytics Environment • Data Provider	• Single machine • No SAS High-Performance Analytics Environment • No data provider

Why LASR Server so faster:

1. Serial Computing: Traditionally software has been written for serial computation. That is instructions are executed sequentially one after another. Execute on a sing machine. Only one instruction may execute any moment.
2. Distributed/Parallel Execution: parallel computing is the simultaneous use of multiple compute resources to solve a computational problem: A problem is broken into discrete parts that can be solved concurrently. Each part is further broken down to a series of instructions. Instructions from each part execute simultaneously on different machines/processors. An overall control/coordination mechanism is employed to return the result.
3. Distributed Data Loading: Big data can be loaded very quickly because data is read in parallel from the Hadoop file system.

How/who start LASR Server:

Pre-requisite: Any user who needs to start/stop a LASR Analytic Server or load/unload/reload tables to the server must have an account on each machine in the cluster and the account must have password less SSH configured across all machines.

To connect to the SAS LASR Analytic Server session, a user must have Read access to this signature file.

How to stop/start LASR Server using SAS Code:
1. PROC LASR
2. SASIOLA libname (non-distributed only).

You start a distributed server with the LASR procedure. See the following example:

```
Options set=GRIDHOST="xxxxxxxxxxx";
Options set=GRIDINSTALLOC="/xxxx/TKGrid_REP";
Options set=GRIDMODE=asym;

proc lasr create port=10011;
  Performance nodes=all;
run;
```

Using a TKGrid_REP installation location and the GRIDMODE=ASYM options enable reading data in parallel with SAS/ACCESS engines from distributed databases and Hadoop clusters.

We can use the LIFETIME option to have a server expire after a certain amount of time.
For example, if you specify LIFETIME=3600(1800), you request that the server is up for at least one hour(3600), afterwards it terminates if no actions are received in any 30 minute(1800) interval

Different data load techniques in to LASR Server:

There are several options available for writing data into memory on the SAS LASR Analytic Server.

- ➢ Writing data to a location that is accessible to SAS VA and manually upload data to memory
- ➢ Writing data to the autoload directory for SAS VA, it gets automatically uploaded to memory. This can be scheduled to run every 30 Min.
- ➢ Use a shared metadata and SAS application server and write data to a location shared by both applications
- ➢ Transformations in the High-Performance Analytics folder of Data Integration Studio

PROC LASR: When reading SASHDAT files, PROC LASR is more efficient than HPDS2 because it moves the individual blocks straight in to memory. HPDS2 coordinate through the head node.

1. BASE SAS and SASIOLA Libname
2. PROC HPDS2.

SAS Engines for SAS In-Memory Analytics:

SASIOLA: Data read in to memory and held in memory for use by multiple processes or users. This functions in a SAS LASR Analytics server environment.

SASHDAT: Data stored permanently in HDFS in the distributed data nodes. Data read in to memory in parallel by SAS High Performance procedures and released from memory when completed execution.

SASHDAT files use a proprietary SAS format that High Performance Procedures and PROC LASR can read. To understand SASHDAT files, let’s consider a SAS dataset broken up into blocks and distributed across a multi-machine Hadoop file system. Each block contains its subset of the data in along with header and metadata information that allows the LASR Analytic Server to read it very efficiently.

The SASHDAT Libname engine deals with SASHDAT files and CSV files. The engine is uni-directional for data; it does not bring data from a SASHDAT file back to the SAS session. Data moves only from the SAS session into the SASHDAT format.

How to list the Data in Hadoop

```
LIBNAME libhdfs SASHDAT PATH="xxxxxxx " SERVER="xxxxx " INSTALL="xxxxx " ;
proc lasr add port=10090  data=test;
performance host="xxxxxx ";
```

run;

HADOOP: Data stored permanently in HDFS in the distributed data nodes. Read and write Hive tables exactly like SAS Datasets.

Load BASE SAS table in to LASR Server:

```
libname lasrlib sasiola host="xxxxxxx" port=10090;
data lasrlib.sashelp.class;
set sashelp.class; run;
```

Typically, large tables are first loaded to a co-located data provider, and then loaded into memory from there, in parallel, by worker nodes. This enables a quick load, because each worker node is able to read in parallel, a portion of the data from local storage.

The Hadoop Name Node maintains a virtual directory along with the location of every physical data block stored in the system i.e metadata. No data is stored on the Hadoop Name Node. Data sources that have been loaded into memory will remain in memory until they are unloaded or the LASR server connection terminates. Usually users that are authorized to access the in-memory table can send requests to the server and receive the results from the server. Data can be loaded from any data source that SAS can access. When you store a table in a distributed LASR Analytic Server, by default the observations are distributed in round-robin fashion. Each worker node on the distributed system will have an equal number of observations and the data is in no particular order.

All data sources that a SAS procedure or a DATA step can consume can be loaded into a LASR Analytic Server by this technique. The Libname could point to SAS Tables, Hadoop, Oracle, DB2 etc.

Data loading via SASIOLA does not happen in parallel. It is therefore not an efficient way to load big data.
However data loading via SASIOLA you can:

- Append
- Partition
- Join

We cannot load data from SASHDAT with this method because the SAS Data in HDFS engine will only write to SASHDAT. To read data from SASHDAT use PROC HPDS2

LASR Analytic Server integrates with Hadoop by reading and writing SAS data in SASHDAT format in the Hadoop Distributed File System (HDFS).

The LASR server is designed to read data in parallel from HDFS. When used to read data from HDFS, the LASR procedure causes the worker nodes to read the blocks of data that are local to the machine.

SAS Studio is the preferred client can be used. By default SAS Studio does not run in interactive mode this will toggle interactive mode on and off. Some SAS procedures like PROC IMSTAT are interactive, which means they remain active until you submit a QUIT statement, or until you submit a new PROC or DATA step. In SAS Studio, you can use the code editor to run these procedures, as well as other SAS procedures, in interactive mode.

SAS Workspace Server which is used to submit SAS Code and connect to data sources i.e Hadoop and the LASR Analytic Server. Which is hosted on SAS Middle Tier?

SAS LASR Analytic Server Tags:

A TAG is a reference to an in-memory table or a group of in-memory tables. TAGS are used in LASR to identify tables because:

- Hadoop files can have many level names but SAS requires one or two level names
- Different users can load tables with the same name.

In BASE SAS all tables have a unique two level name consisting of libname.memname. The Libname and memname have an 8 character limit. There is no such constraint in the LASR server where tables have multi-level names.

However with the introduction of the SASIOLA Libname and the ability to read LASR tables from BASE SAS a problem arose. BASE SAS can only process two-level names. To solve this problem the concept of TAGS was introduced.

The tag is used to replace everything but the last level of the name. It is possible that multiple users could load data into the server with the same table names. Tags facilitate this and prevent naming conflicts.

Accessing Tagged In-Memory Tables to access a specific table use the TAG= option. This option specifies the tag to use for identifying the tables in the server instance.
Use the FULLCOPYTO option on the LASR ADD or the data step.

```
proc lasr add data=hdfs.iris fullcopyto=3 port=10010  ; run;
libname example sasiola port=10010  tag="hps";
data example.prdsale(fullcopyto=5);
set sashelp.prdsale; run;
```

Data Partitioning and Ordering:

By default, partitioning is not used and data distributed in round-robin algorithm. This applies to SAS Data in HDFS engine as well as SAS LASR Analytic Server. We need to partition the table in a distributed server (HDFS) using DATA Step to take the advantage of partition. Usually, Data analysis can be done frequently on subset of the data table by few variables. If you partition the data table by few variables, you speed up processing in an exploration /report because the server can direct all of the work to the one machine containing the relevant fields. However, if the data table is not partitioned by those fields which used for analysis, performance can be negatively impacted. Once the table is portioned in HDFS and same loaded in to LASR Server, same partition can be maintained in LASR.

If you also specify the ORDERBY= option, then the ordering is preserved when the table is loaded to memory too.

Partition keys are derived based on the formatted values in the order of the variable names in the variable-list. All of the rows with the same partition key are stored in a single block. This ensures that all the data for a partition is loaded into memory on a single machine. The blocks are replicated according to the default replication factor or the value that you specify for the COPIES= option.

If user-defined formats are used, then the format name is stored with the table, but not the format. The format for the variable must be available to the SAS LASR Analytic Server when the table is loaded into memory. This can be done by having the format in the format catalog search path for the SAS session.

SAS High-Performance Deployment of Hadoop:

SAS High-Performance Deployment of Hadoop enables a homogeneous block distribution, resulting in balanced memory utilization across the SAS LASR Analytic Server. Hadoop Distributed File System (HDFS) stores files in blocks of a certain size using the local disks on the host of a cluster. Hadoop arranges replication of the blocks in the case of node failure.

Writing Data to Hadoop via SAS Data in HDFS engine

We can use the SAS Data in HDFS engine to add tables to HDFS. When a table is added to HDFS, it is divided into number of blocks that are distributed across the machines in the cluster. The SASHDAT libname engine deals with SASHDAT files and CSV files. The engine is uni-directional for data it does not carry data from a SASHDAT file back to the SAS session. Data moves only from the SAS session into the SASHDAT format.

HPDS2 PROCEDURE

This method uses multi-threaded distributed processing from HDFS to LASR through the LASR head node. It is useful when there is a large amount of intensive, row-independent computations to be performed. It is also useful if you need to filter the data in HDFS and load a subset into LASR.

This method support to Load the data in parallel and Subset of data load for HDFS engine. Append and partition does not support the same procedure.

Parallel Loading from Remote or Non-SASHDAT Sources:

The SAS Embedded Process supports Asymmetric mode to load data in parallel.

1. LASR is co-located with the data that you want to analyze, but your data is not CSV or SASHDAT. For example: Hive tables or sequence files.
2. The data cluster that has the data you want to analyze is not co-located with the LASR Analytic Server

```
options set=GRIDHOST= ‘xxxxxxxx’;
options set=GRIDINSTALLLOC="/xxx/xxxxx/TKGrid_REP";
options set=GRIDMODE="ASYM";
proc lasr add port=10090 data=xxxxx;
performance mode=ASYM;
run;
```

TKGrid_REP is the remote embedded process.

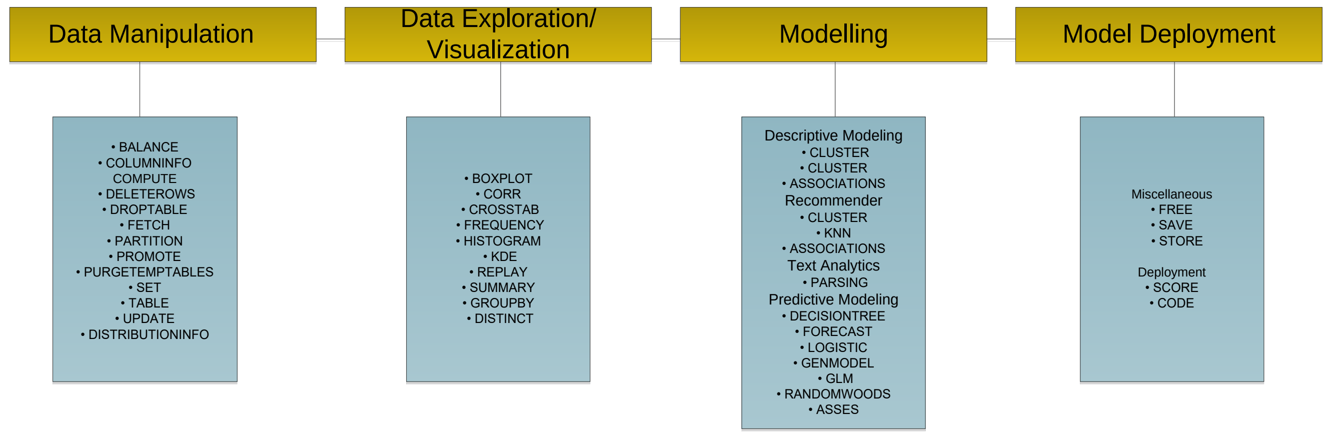
Allows a Base SAS session to interact with a LASR Server via the SASIOLA Libname.

- Manipulate Data
- Compute statistics
- Create new tables in-memory
- Return results/data to Base SAS.

PROC IMSTAT is Interactive RUN-group processing enables you to submit a PROC step with a RUN statement without ending the procedure. You can continue to use the procedure without issuing another PROC statement. To end the procedure, use a RUN CANCEL or a QUIT statement.

```
proc imstat ;
table lasrsrv.mega;
summary expenses revenue /
groupby=(productline) aggregate=(sum) ;
run;
crosstab productline * facilityregion /
measures chisq nomiss;
run; quit;
```

PROC IMSTAT Statements:



The IMSTAT procedure is used to perform in-memory analytics on tables that are loaded in a SAS LASR Analytic Server instance. When you use the IMSTAT procedure the:

- LASR Analytic Server performs the analytic processing
- Results of the analytic operations can be returned to the SAS session and some results can be saved to a temporary table in memory.

LASR Server Informational Statements:

IMSTAT statements provide information on a LASR Analytic Server and its tables.

SERVERINFO returns info about the SAS LASR Analytic Server including host, port, and number of tables.

TABLEINFO returns info about an in-memory table includes table name, label, no. of rows and columns.

COLUMNINFO returns info for all the columns in an in-memory table includes name, type, length, formats etc.

The **TABLE** statement sets, active table in-memory on which future IMSTAT statements will operate on.

Like in SAS Formats, PROC IMSTAT and the LASR Analytic Server support SAS formats and user-defined formats. Every variable is associated with a format when it is loaded into memory. If a variable does not have a format it is assigned a default format when it is loaded into memory that is numeric best 12. , character$.

User defined formats must be made available to the LASR server in XML format.

There are two ways in which formats are made available:

Automatic: when user-defined formats are used with the data, the formats are converted to an XML representation and transferred to the server by the SASIOLA libname, PROC IMSTAT and PROC LASR.

Manual: Create XML with PROC FORMAT and pass to LASR with the FMTLIBXML= option.

The FETCH statement is used to retrieve rows from an in-memory table.

IMSTAT and ODS Results by default are output ODS destination Results it can also bea PDF, SAS Dataset etc. Results are stored with the SAVE option and printed later with the REPLAY statement.

Codes are stored to macro variables with the STORE statement and read by later tasks in the same session.

SAVE Option saves a ODS output object generated by an IMSTAT statement for later use.
SAVE Statement saves an in-memory table to HDFS.

The REPLAY statement enables you to display a previously saved result table or set of tables.

CONCLUSION

It is necessary to know the analyst/data scientist about the LASR server distribution types and the components with In-memory and in-database when dealt with SAS and HDFS data with improved options and tips can leverage for exploration and reports on SAS Visual Analytics.

REFERENCES

https://support.sas.com/documentation/cdl/en/inmsref/68736/HTML/default/viewer.htm#titlepage.htm

http://support.sas.com/documentation/cdl/en/inmsref/67597/HTML/default/viewer.htm#p1ohlb0pdcdsj4n1oym3pbnov0sw.htm

ACKNOWLEDGMENTS

I would like to thank Sedhou Ranganathan, Chaturvedi Ashish for his constant support and encouragement. I also like to thank my manager Shaun Maloney for their valuable inputs and support.

CONTACT INFORMATION

We encourage you to kindly share your valuable comments and questions. Please contact the author at:

Name: Venkateswarlu Toluchuri
Enterprise: United Health Group
Address: Hi-Tech City 2 Building H09, Hyderabad, India-500081
E-mail: venkat.toluchuri@optum.com, venkateswarlutoluchuri@gmail.com