Paper SAS5192-2020

Data Preparation Techniques for Cloud-Based Data Sources

Jennifer L. Nenadic, SAS Institute Inc.

ABSTRACT

Traditional data warehousing continues to evolve as the industry continues to embrace more and more cloud-based transactional systems. The days of accessing on-premises source system databases to extract data for your data warehouse, report, or analytics model are becoming more and more rare. In order to be ready to successfully navigate this new world, this session focuses on techniques and alterations needed to update your data preparation and data warehousing strategies. These strategic changes address the many differences that are presented by cloud-based source systems.

INTRODUCTION

From this paper you will gain a better understanding of what to expect when sourcing data from cloud-based third-party software vendors. You will also learn how cloud data access differs from traditional on-premise transactional systems that offer direct access to their underlying database. For those that leverage Extract, Transform, and Load (ETL) techniques for populating a data mart or data warehouse, this paper will discuss the impacts to the Extract layer of your process.

This paper will not only include technical guidelines and information, but also will include recommendations for how to equip your organization for the larger cultural and process changes associated with consuming data from third-party cloud-based applications.

Although this paper is primarily focused on third-party cloud-based software, many of the techniques described could also be applied to internally developed (i.e. non-3rd party) cloud-based applications. We suggest working directly with the owners of applications to differentiate what points in this paper are relevant.

This paper is intended for data management and data warehousing professionals and management. This includes data engineers, data architects, ETL developers, or anyone responsible for implementing and managing an Enterprise Data Warehouse (EDW) or data marts.

This paper is not intended to nor will address the follow auxiliary topics:

- In-Depth Cloud Architecture. Only information that is relevant and necessary to extract data from a cloud-based system will be included. This paper is not intended to serve as educational material on cloud-based system architectures and frameworks
- Migration path for moving from an on-premise data warehouse into a cloud-based data warehouse architecture.
- Guidelines and best practices for how to architect or implement an EDW or data mart. The content of this paper is applicable to both dimensional and normalized data models.

HIGH-LEVEL DIFFERENCES IN APPROACH AND DESIGN

Before exploring the detailed elements of how to source from a cloud-based application, it's important to first understand some high-level differences in approach as you shift into extracting data from third-party cloud applications.

ON-PREMISE TRANSACTIONAL SYSTEMS OVERVIEW

In traditional non-cloud data extraction techniques, data engineers have direct access to the transactional system's data source. Historically, data engineers would partner with source system experts to review the data model of the transactional system. Often this would include leveraging a SQL viewer to query, analyze, and profile the data in the source system. Before beginning any design work, data warehousing architects could get familiar with the data, explore it, and develop the full ETL data flow design and data model. One typical deliverable from the architecting and designing phase is a source-to-target document. The goal of this document is to exactly define the tables and columns that the development team should access within the source system's data model. The document also specifies how data should be transformed and loaded into the final target data model within the data mart or data warehouse.

Implementation generally consisted of developing jobs that connected directly to the database to source the data from the various transactional tables or from views that had been created. Once the database connection was achieved, standard ETL and data preparation techniques could be used to load the data into the data warehouse or data mart. All of this code and implementation logic would be executed within the internal firewall.

Please see Figure 1 below for the high-level conceptual architecture when sourcing data from a traditional on-premise application.

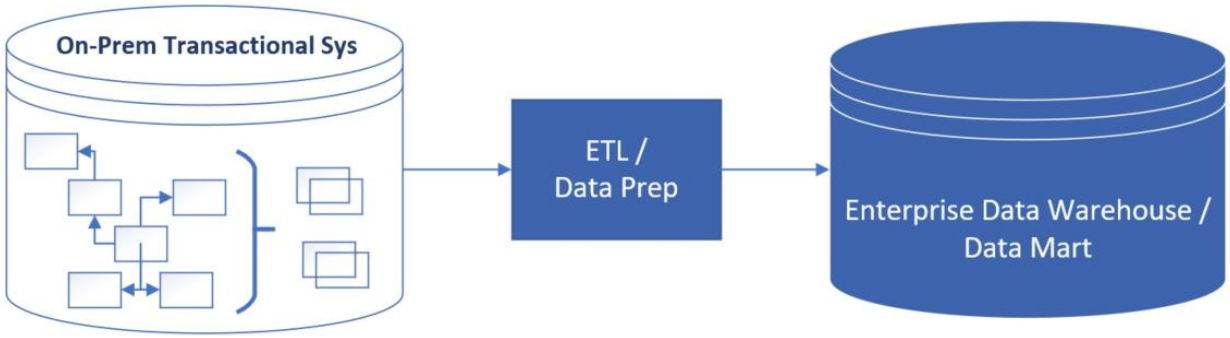

Figure 1 High-level data flow when leveraging on-premise transactional systems

3RD PARTY CLOUD APPLICATIONS OVERVIEW

When sourcing data from a third-party cloud application, the basic principles as described in the section above quickly change. In cloud-based applications, the published data architecture shifts from a data model focus to an object-oriented focus. Third-party vendors typically do not produce nor share the application's data model, as access is not granted to the underlying data model.[1] Data engineers and data architects are quickly met with puzzling looks when they ask to review the data model and access to the underlying table structures and views. Therefore, as shown in Figure 2 below, cloud applications can appear as more of a "black box" for traditional data engineers and data architects.

For implementation, data engineers no longer can connect directly to the cloud application's underlying tables. New techniques for accessing and sourcing the data must be leverages (including API, web services, SFTP, etc.). The techniques are described in more detail in the sections below. Further, data engineers now have to navigate passing data through

[1] Exceptions can occur for non-native cloud companies (i.e. companies that started as on-premise software providers that are migrating their offerings to the cloud). For companies that migrate their offerings to the cloud, they sometimes retain some of the on-premise architecture approaches within the cloud, including surfacing the underlying data model.

firewalls, which means the data warehousing team's engagement with the IT infrastructure and networking organizations often increase. Lastly, because cloud systems do not include direct database access, data engineers must build their expertise with manipulating and accessing data in new data formats (XML, JSON, etc.).Once the data engineering team has successfully addressed the Extract component of the data preparation process (i.e. ETL), including changing the data into the necessary format to conduct the remaining data processing, the rest of the ETL process can complete in the same way as with traditional on-premise data sources.[2]

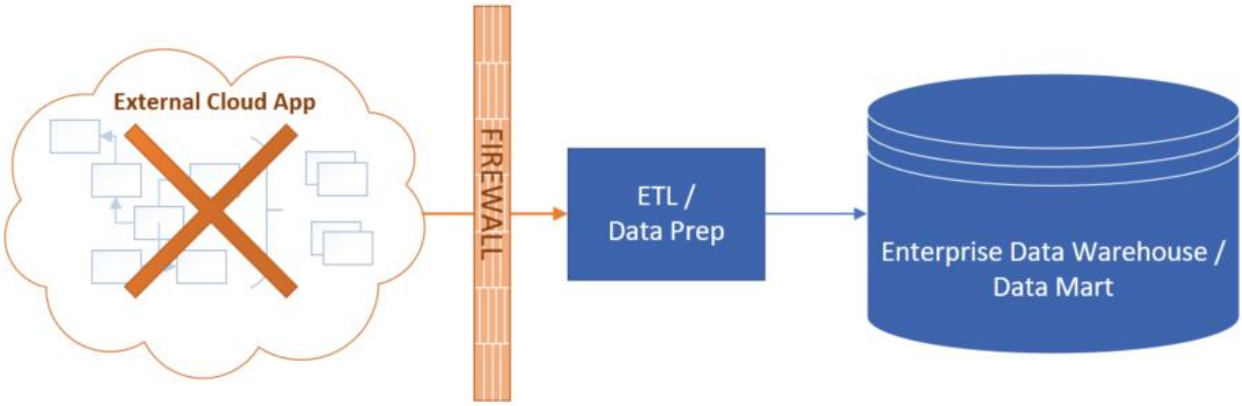

Figure 2 High-level data flow when leveraging 3rd party cloud-based applications

NEW WORLD DETAILS

More in-depth discussion on what to expect when sourcing from cloud

STANDARD EXTRACT CAPABILITIES

SAS® has an internal EDW that has been on a journey of sourcing data from third-party cloud applications for quite a few years. We've engaged with cloud-native applications and companies, cloud-offerings that were based on an original on-premise design architecture, newer players to the market, and well-established cloud providers. This section outlines the standard techniques for data extraction we have encountered across all our interactions, and guidelines/best practices for how to incorporate into your data processing ecosystem. third-party cloud applications may not leverage all the techniques and capabilities mentioned in this paper, but in our experience, if they provide data extraction capabilities, it generally aligns with at least one of approaches below.

Extracting Data via Reports

One common approach leveraged by third-party cloud applications to extract data relies on the use of their reporting interface. The same interface used to create reports to be showed within the application, can be used to build data extraction feeds.

The first thing you'll generally notice when interacting with the reporting interface is that it is generally object-oriented in its approach. You aren't joining tables and columns together, but rather the reporting interfaces allow you to join objects based on their relationships. This will limit how sophisticated you can get, and sometimes you may want to include information together on a report for which there is no relationship established through the

[2] If your data preparation techniques included regular calls out to the various source systems versus consolidating in an extract layer, wider changes may be required. All data pulls from cloud-based systems will be impacted, even if those steps occur outside the traditional Extract step of an ETL process.

report builder. In this case, you will need to create more than one report and ensure you have the data included to enable you downstream to perform the joins desired.

As with most reporting tools, typically you will find a reporting development interface and a report viewing interface. Depending on the security and responsibilities defined in your organization, you may need to engage another team to author reports.

Once a report is built, there are typically two techniques to access the data in the report for data warehousing purposes. Those are by scheduling a report via SFTP (using a data "push" approach) and calling it on-demand via a web service (using a data "pull" approach). third-party cloud applications may have only one or both of these capabilities available. The sections below outline the general guidelines and approaches for each.

Scheduling a Report

In this approach, cloud applications have a report scheduler to "push" the data. This enables you or a business super user to schedule a report to be executed at a specific time or on a specific cadence. The output is typically encrypted and transmitted to an SFTP drop zone location. You will want to validate with the third-party vendor the types of data formats that are used. You may find for some vendors the format is configurable via the cloud application's scheduler. Once the data has been dropped, it will have to be decrypted before you can leverage it in your data warehousing process.

Calling a Report via Web Service

In this approach, you will make an HTTP or an SHTTP call to execute and deliver the data in the report you have built within the third-party cloud application. Your environment will need to be set up with access to the internet to make these calls, which will go outside your internal firewall. Typically, the data will be delivered unencrypted, so you will not need to handle decryption techniques; however, you will want to confirm that with the third-party vendor. In our experience, we have found that the data typically is delivered using an XML or JSON format.

Extracting Data via APIs

APIs are different in that you are working within the confines of the methods and extractions provided by the third-party vendor. With the report method, you can custom build the exact layouts and attributes you wish to include. With APIs these decisions are made by the vendor. You will want to get very familiar with the API documentation to discover which APIs are the best fit for your needs. It is possible that multiple API calls may be needed to extract all the attributes you require. APIs tend to have similar data output formats as what are used when calling web services. The exact formats should be included in the API documentation.

Exceptions

The techniques mentioned above are the common approaches to extracting data from a third-party cloud application. However, there can be exceptions, especially for vendors that have migrated their applications and offerings to the cloud instead of developing cloud-native applications. For these situations, you can often find a hybrid approach, where some of the capabilities only available in on-premise applications are still available even though the application is being offered in the cloud. You may find cases where you can explore and write SQL queries against the backend tables or are able to find documentation on the backend data model. In cases where third-party application has been migrated to versus developed in the cloud, you will want to work directly with the vendor to understand their data extraction techniques as the hybrid approach can look different vendor to vendor.

LESSONS LEARNED

This section includes some general lessons we have learned in our years of extracting data from the cloud. These lessons were very costly when they were discovered, but if they are known early can help you avoid delays and risks of not being able to extract what you need.

- Customizations – Third-party cloud applications are generally not customizable, which means you may need to prepare to handle some situations where you historically may have been able to address directly within the application.
    - Cloud applications can allow unexpected things to happen from a data quality perspective. These can include situations that may be detrimental to your end user's analysis or reporting. Be prepared to incorporate a lot more data quality checks on your side because you likely will not be able to incorporate or address them directly in the application.
    - You cannot make changes to improve performance. The performance is what it is and can often be throttled by the application or the vendor without your awareness or insight. If your extracts take a long time, you cannot ask to have indices created or the load balancing techniques altered. You need to test early and be prepared for possible performance issues that you will need to circumvent on your side.
- Upgrades - Clouds have a set upgrade cadence set by the vendor and often have multiple environments (also called instances). You typically do not have control over the upgrade schedule, so you will want to plan cycles to test the upgrades and ensure you have time to remediate any issues discovered before the upgrade date. You cannot postpone an upgrade based on your readiness or even due to issues discovered. Upgrades are often made available early for testing in a special upgrade instance. This is an instance that you would want to switch to for your extracts only for testing in preparation for an upgrade.
- API Changes – APIs can change and change frequently. APIs you may have used historically can become deprecated with an upgrade (or even outside the upgrade schedule). If you use APIs, you will want to stay informed of any planned changes. Further, you may find some companies have API versions. Although an API may become deprecated in a newer version, there may not always be a replacement. When this happens, you may need different extract designs that allow you to call APIs from multiple versions to meet all your needs.
- Limited data warehousing capabilities - Data warehousing needs are typically a lower priority use case for third-party cloud applications and as a result you may not have the core capabilities needed or in the way you'd prefer.
    - Cloud applications can be sold without any data extraction techniques at all or those techniques can be very immature.
    - It's possible that some but not all of the data is extractable.
    - There may or may not be capabilities to do delta extracts (i.e. only pull changes), or the capabilities are available through one extraction technique but not others.
    - May or may not be possible to go back and extract data as of a point-in-time.
    - Most APIs are not designed for data warehousing needs (i.e. bulk extraction), but rather for operational needs where they transmit only 1 record at a time. Do not assume that all APIs are a good fit for data warehousing purposes. If you choose the wrong ones, you will experience performance issues, and possibly worse, encounter API call limits and/or overage fees.
- Hard Deletes - Cloud applications can often have "hard delete" capabilities where data is permanently removed from the system, meaning even the cloud provider cannot get it back. This can happen even for critical transactional data. If you find this is possible, you will want to ensure your design can handle programmatically determining what data has been permanently removed from your system. Sometimes third-party cloud

applications will use soft deletes by default but also have out of the box capabilities that allow permanent deletes as well. You cannot rely on soft deletes alone.

- Web service and API call failures – when you leverage "pull" methods to extract the data, you may find failures are not handled gracefully. It's possible that data is returned without error but that the data is not complete (i.e. only a small subset of rows are returned). Be prepared to have more robust testing in your extraction layer of the data preparation process to confirm you are getting a complete extract and incorporate your own error handing techniques if you assess you may not have received all the data.

RECOMMENDATIONS AND GETTING PREPARED

PARTICIPATE IN THIRD-PARTY SOFTWARE REQUEST FOR PROPOSAL (RFP) PROCESSES

If you intend, either now or in the future, to source data from a third-party cloud application, it is crucial that you or your organization is involved in the RFP or vendor selection process. We recommend building relationships with your procurement and/or vendor management organization to help educate them on why your role is so critical in the vendor selection process. Below are some points to help with this education:

- Cloud applications can be developed and sold without any data extraction capabilities. If there is an expectation that you are adding this data to your data mart or data warehouse, you will want to validate that you can get the data out of the third-party cloud application. If the vendor has data extraction capabilities on their roadmap, this will impact your timeline for adding it to your data warehouse or data mart.
- Cloud applications do not always provide capabilities for extracting all data fields. Sometimes data extraction capabilities can only include basic information and will exclude key fields you may need for reporting and analytics. It will be important to understand the coverage of attributes included in data extraction capabilities and amend the contract with the vendor as needed to ensure will be able to successfully extract the data you need.
- Cloud applications can develop data extraction techniques focused exclusively on real-time operational system needs. For example, a financial system often needs information from the HR system. Specific data interfaces are often delivered with the third-party application to enable these real-time, single transaction data feeds. However, operational API's and data connectors are not designed with data warehousing and bulk data extraction needs in mind. Therefore, you will likely experience substantial performance impacts if these APIs are used, unless your data warehousing is designed using a queuing / message bus architecture.
- Often data extraction capabilities are offered by vendors through add-on fees, forcing you to buy their version of a data warehouse in order to extract the data. This can be expensive, inflexible, and duplicate the efforts of your data warehousing organization. Vendors will rarely call this offering a data warehouse, but if you are involved in the RFP process, you can ask questions to understand the details of the offering and assess the suitability of this approach.
  Some key limitations of a vendor data warehouse offering that I like to explore include:
    - refresh frequency,
    - uptime and availability SLAs, configurability (i.e. can I modify the warehouse? What if data elements I need aren't included? Do I have to pay professional services to get extra columns added?),
    - point-in-time capabilities and identification of changes,
    - any risk of primary key regeneration,
    - bulk extract capabilities and limits

- Cloud applications can have contractual limitation on data extraction. This can include maximum thresholds on the number of API calls allowed (with either overage fees or expectations that API calls will not be executed one the threshold has been met). If the vendor only has single transaction APIs (and not bulk data extraction APIs), then the data warehouse will be making high volumes of API calls.

CREATE ARCHITECTURE AND DESIGN STANDARDS

To help expedite adoption and implementations where data is being sourced from third-party cloud applications, it can be very helpful to create basic architectural standards and guidelines for your organization. When embarking on this step, it's important to understand all the groups and individuals that will be engaged in extracting data from the third-party cloud application.

It can be helpful to have a holistic data flow design that highlights the various groups with roles and responsibilities outlined, so that everyone is clear on how the larger process will work. This will also make it easier in the future if your company chooses to engage additional third-party software vendors in the future.

Further, it can also be very helpful to have a detailed design created specifically for the data engineer to outline the details of what a data extraction job(s) should entail when data is sourced from a third-party cloud application. This would also include naming conventions and error handling technique.

Report-based Data Extraction Capabilities

When thinking about all data extraction techniques based on building a report in the cloud application, you'll want to work to develop some guidelines and standards.

First, who is responsible for developing and authoring reports? Is this something the data engineers will be able to do or is this owned by a business superuser or subject matter expert (SME)? When reports are developed, is there a naming convention for the reports to help identify which ones are developed for use by the data warehousing team? How will columns be named in the report? How much transformation are you comfortable undertaking within the report itself?[3] Do you want to try to build reports that reflects and aligns with your target data mart or data warehouse design or not?

Using SFTP

These standard architectures will likely be developed in conjunction with the IT networking and infrastructure team. Below are some questions to consider:

- Report scheduling. Who is responsible for scheduling the reports built to be SFTP? Given this will be a "push" model instead of a pull, special consideration is needed depending on when you want to load and refresh this data mart or area of your data warehouse.
- Standard location for SFTP drop zones to be created. Who all has access to them? Can there be a standard folder layout for development, test, and production instances of the source system?
- Decryption Standards. How are the files to be decrypted? Is there a timing element to decryption (i.e. a listener that pings the SFTP location on a regular cadence to look for new files then automatically decrypt and move to the decrypted location)?

[3] As a reminder, you cannot create indices to help improve performance of reports developed within the cloud applications; therefore, guidelines on the level of complexity you are comfortable incorporating into the authored report will be important and often can be vendor-specific or application-specific.

Who owns the process and code to decrypt data?[4] Can the location of the decrypted files also follow a standard folder layout? How and when are files deleted from the decrypted location?[5]

- Extract Design Data Process Flow – As mentioned above, taking the time to create detailed design documents to outline exactly what should occur can be very helpful.

Using Web Service Calls

When reports are extracted using web service calls, reports do not need to be scheduled. Instead, data is extracted on-demand. Using web service calls can add some additional design and architectural considerations.

- Web service calls can fail and return no data.
    - Temporary failures – Your designs will need to include a looping mechanism when the web service fails because often the web service will complete when called again.
    - Extended failures - Depending on the vendor and your contractual SLAs, these outages can continue for extended periods of time - for multiple runs and, though less common, even for days. This is an important design consideration for mission critical data that needs to be refreshed within a specific time period. If your organization cannot function without this data being updated, you'll want to have a back-up architecture and work-around. This could mean you also setup an SFTP drop zone and report schedule that is only used in these situations. This is where naming conventions are so important, as you'll want the naming conventions of the web service to closely align with those of the SFTP file drop.
- Data extraction limitations – Ensure your design addresses any data extract limitations or vendor-imposed restrictions.
    - Incomplete file received - Web service calls can return successfully, but without all the data. Therefore, t's important that your extract job includes basic checks for how many records you expected to receive and aborts if those minimum thresholds are not met.
    - Max data extraction limit reached - Some third-party cloud applications contain thresholds on the maximum data you can extract at a time. This can be based on the number of records or file size. This may or may not be included in the vendor's documentation (and is another great reason to be engaged in the RFP process). If you discover a cloud application has these restrictions, your data extraction design needs to incorporate a two-step process. First, a call to identify how many records need to be extracted, which may need to be addressed via development of an additional report within the cloud application. Then, second, a looping mechanism needs to be implemented to make multiple calls to extract all the data. This second step must also include aggregating the results of the various calls.

[4] Some cloud providers will use SFTP even for transactional needs, so it may be helpful to have a larger conversation as this process may impact more than the data warehousing team

[5] This is especially important when sensitive, confidential, or personally identifiable information (PII) is being transmitted. To be consistent, I recommend keeping your decrypted file directory clean, with only the minimum files retained until loaded into the warehouse, and leveraging your standard data warehousing archiving and data retention processes to store extracted data.

API-base Data Extraction Capabilities

Many of the design considerations of API-based data extraction capabilities align with those of web-service based extractions. Therefore, please review the Using Web Service Calls section above for general considerations.

With that said, APIs can have some differences to web service calls. As mentioned in the Extracting Data via APIs section above, a key difference is that with an API capability there is no report built within the cloud application. Your extraction design and implementations are dependent on the outputs defined and determined by the vendor. It's a good idea to have links directly to the API documentation in your design documents, especially if the API's change over time to ensure you can adapt quickly.

In those cases where the only option to extract data is using single transaction APIs (i.e. bulk data extraction options are not available), your designs need to incorporate performance-related considerations

- Cloud vendor performance considerations − Making potential thousands or tens of thousands of API calls cannot only impact your batch performance but could have performance impacts on the cloud application itself.
- Data Warehouse performance considerations − An elongated extract process can have impacts on your data warehouse, consuming compute cycles for extended periods of time and impacting other data processes. In some circumstances, this impact can be reduced by enhancing hardware and infrastructure. These enhancements ensure, for example, that you have sufficient CPU and I/O bandwidth for your API calls. However, I would not anticipate this being a silver bullet to fully negate the impact of thousands of web-service calls.

For both situations described above, there are some common techniques to consider that will help reduce the performance impacts. First, consider changing the time of the extract to non-peak hours. Second, consider leveraging a design that makes calls throughout the day instead of a one-time process. This would reduce the number of API calls made at each run.

EDUCATE AND TRAIN STAFF

If your organization has historically interacted with databases and traditional data-related programing languages (SQL, SAS, etc.), you will need to help educate your team on how to navigate this new world. Education and training do not always need to be formal, but your team does need to understand how their design and implementation approaches are changing, and how to interact with these new technologies. To help with this journey, SAS® has created new capabilities to ease this transition for your team. Please see the HOW SAS® CAN HELP section below for details. Below are some suggested areas of focus for your team:

- XML − How to read and understand an XML file, including how XML files interact with and rely upon an XML Schema Definitions (XSD). Team members will also need to be equipped with skills for how to programmatically ingest XML data and convert into a traditional table format, and how to navigate errors in this process (including blank files).
- JSON − Similar to XML file formats, team members will need to understand how to read and ingest JSON files. Team members need to understand expectations on attribute lengths in a JSON files and how to ensure they are not truncating data values. Similarly, team members will need to understand how to convert JSON data into a more structured table and column format.

- APIs – If your team has not interacted with APIs before, they will need to understand the basics of an API and how it works. Another important area of emphasis is on the difference between APIs designed for transactions versus APIs designed for data warehousing and bulk extraction capabilities. Error handling can be different and team members need to be educated on the types of errors and how to navigate them in their code.
    - General architectural differences when sourcing data from third-party cloud applications – A great way to expedite this is to create standard designs and architectures as mentioned in the sections above.

HOW SAS® CAN HELP

Below are some suggestions of capabilities available in SAS® to ease your transition to extracting data from third-party cloud applications.

- PROC HTTP - You can make web service and API calls directly from SAS® using PROC HTTP
- JSON engine – If licensed, the JSON engine can be a great way to ingest JSON data.
- XML Mappers – If you have SAS® Data Integration Studio licensed, there are transforms available to expedite and auto map data in an XML format into a table format.
- Custom Transformations – If you have SAS® Data Integration Studio licensed, you can leverage custom transformations to create a template for your cloud application process. This allows you to set up a code template based on specific inputs that will reduce the amount of time your staff needs to learn and implement code. Below are a few examples where custom transforms can be very useful.
    - For vendor-specific API or web service calls.
    - For standard checks and transformations to ensure you have full data returned

CONCLUSION

The evolution and the movement to the cloud is no longer just an idea; it's happening and the speed at which companies are beginning to leverage cloud-based applications for their operational needs continues to increase. In order to keep pace with this technological direction and reality, data management professionals need to incorporate new data extraction techniques to ensure they can continue to provide the critical data necessary to enable their business.

Lastly, getting prepared for sourcing data from the cloud is not exclusively about coding. New relationships and considerations are needed when purchasing cloud applications. New skills are needed by data engineers and data management professionals who will need to interact and source data from these cloud systems, and new architectural principals need to be established and applied.

Whether you are already started on your journey of source data from cloud applications or just beginning, getting educated and ensuring you have a solid holistic foundation will help expedite your effectiveness and ensure that you are set up for success.

RECOMMENDED READING

- *The ABCs of PROC HTTP* - https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3232-2019.pdf
- LIBNAME Statement: JSON Engine - https://go.documentation.sas.com/?docsetId=proc&docsetTarget=n0bdg5vmrpyi7jn1pbgbje2atoov.htm&docsetVersion=9.4&locale=en

- Creating and Using a Generated Transformation - https://go.documentation.sas.com/?docsetId=etlug&docsetTarget=n0iu33h8k5gxhon0za9f10hfe0pk.htm&docsetVersion=4.904&locale=en

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Jennifer L. Nenadic
Director, Data and Analytics Center of Excellence, SAS®
Jennifer.Nenadic@sas.com
http://www.linkedin.com/in/jennifer-nenadic